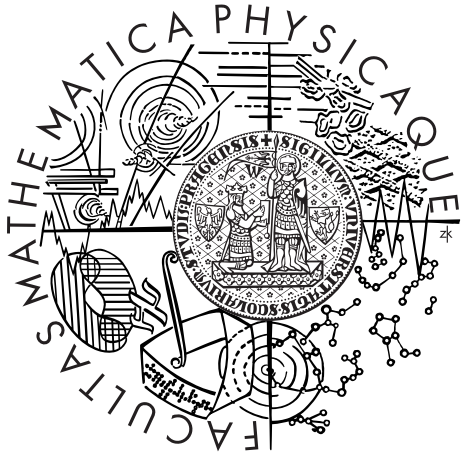
Charles University in Prague
Faculty of Mathematics and Physics

MASTER THESIS



Tomáš Hlaváček

Routing policies and real paths in the Internet

Department of Software Engineering

Supervisor of the master thesis: RNDr. Ing. Jiří Peterka
Study programme: Informatics
Specialization: Software Systems

Prague 2015

My sincere gratitude goes to my supervisor RNDr. Ing. Jiří Peterka and external counselor Mgr. Ondřej Filip, MBA for valuable advice and to companies CZ.NIC, z.s.p.o., Ignum, s.r.o. and Trenka Informatik AG for providing data feeds and substantial computing resources.

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular the fact that the Charles University in Prague has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 paragraph 1 of the Copyright Act.

In Prague, September 23, 2015 Tomáš Hlaváček

Název práce: Routing policies and real paths in the Internet

Autor: Tomáš Hlaváček

Katedra: Katedra softwarového inženýrství

Vedoucí diplomové práce: RNDr. Ing. Jiří Peterka, Katedra softwarového inženýrství

Abstrakt: Směrovací politiky jsou v současnosti reprezentovány jazykem RPSL a jeho evolučním rozšířením RPSLng. Tyto jazyky slouží k popisu jednotné směrovací politiky každého autonomního systému. Praxe ukazuje, že navzdory existenci převodních nástrojů z RPSL a RPSLng do konfiguračních formátů pro běžně používané routery, skutečná konfigurace se z RPSL generuje málokdy a směrovací politika se tak stává jen opomíjenou evidencí, která často neodpovídá skutečnému nastavení sítí. Pro případnou nápravu nesrovnalostí bude zapotřebí pravděpodobně opatření na úrovni změny formátu RPSL. Předkládám zde výsledky dlouhodobého měření výskytu nepřesností ve směrovacích politikách, které se rozchází se skutečnými cestami v Internetu. Vedle toho předkládám seznam nejčastějších problémů a návrhy řešení, které by mohlo situaci dlouhodobě zlepšit.

Klíčová slova: Internet, BGP, RPSL, routing database, DFZ

Title: Routing policies and real paths in the Internet

Author: Tomáš Hlaváček

Department: Department of Software Engineering

Supervisor: RNDr. Ing. Jiří Peterka, Department of Software Engineering

Abstract: Routing policies are now represented by RPSL and by its evolutionary extension called RPSLng. These languages can be used for describing unique routing policy of each autonomous system. Experience shows that even though there are translation tools from RPSL and RPSLng to configuration formats of commonly used routers, the actual network configuration is rarely generated from RPSL sources and routing policy is then perceived as marginal paperwork, which often does not reflect the real network settings. There will be most likely a need for RPSL format change in order to remedy the discrepancies. To support this I present long-term measurements of inaccuracies in routing policies compared to real paths in the Internet. I also present a list of the most frequent problems, and I offer suggestions, how to reform RPSL to improve situation in the long term.

Keywords: Internet, BGP, RPSL, routing database, DFZ

Contents

Introduction

0.1 Thesis topic

The topic of this thesis is loosely based on my previous Bachelor's thesis as well as on my experience that comes from operating several Autonomous Systems. Few important ideas that concern routing policies also came from my work on research into Internet security in CZ.NIC, z.s.p.o. It seems to me that majority of security problems, operation difficulties and network development issues are based on inherent features of the Internet routing system. These problems can and do affect almost everybody that participates in the Internet and include all types of DoS attacks, route-hijacking, either intentional or accidental, route instability problems as well as non-optimal routing and unnecessarily unreachable routes.

My Bachelor's thesis [1] has been focused on operation of a single AS and the intention was to support administrators in their effort to generate a router configuration from Routing Policy Specification Language. In this work I created a RPSL translator, which was unique in certain aspects and it was perhaps easier to use than another ones, but still it was only yet another RPSL software. It seems that most people do not complete even the step that is required before any RPSL translator could be applied - describe the routing policy of their autonomous systems.

In the meantime there was a brief discussion in RIPE community about the future of Internet routing registry (IRR) in RIPE DB and it proved that speaking about IRR inevitably leads to discussion about the Routing Policy Specification Language itself. The problem is that in last two or three years there was almost no interest in RPSL and IRR improvement, because the RIPE and even the world-wide community was busy with IPv4 address space exhaustion, procedures for the last */8* prefix, transition to IPv6 and fighting fraudulent behavior in IPv4 registrations.

In the same time I had an opportunity to move along from the AS operation business to research and development and focus on core Internet technologies. I realized that the most serious problems we, as the whole Internet community, have to deal with, are somehow linked to the Internet routing and therefore to routing policies.

From minimal discussion about routing policies both on the RIPE meetings as well as on mailing lists I managed to figure out that the community needs hard data about the Internet routing system issues with special attention to routing policies. This is rather obvious idea and I decided to undertake this difficult task. I have to admit that I underestimated an extent of this topic in the first place even though I had previous experience with implementing RPSL parser. The complexity of the task is most likely the reason why nobody else has done the same thing before.

On the other hand the experience of doing the whole thing by myself is valuable asset, because I can report on the whole topic at once. Implementing old standards is always a good starting point for improving them or inventing new ones. This thesis presents my measurements on the topic as well as my opinions

and ideas how to improve routing policy system. The results and recommendations are still only a starting point for future work. There are many things to be done in order to change current standards, reach consensus on new ones, change general attitude towards routing policies, and provide tools for the Internet community to mend the current unsatisfactory system. Aim of this work is primarily to start the discussion and provide arguments that support the need for change.

1. Internet routing

1.1 Internet resources

Internet routing is a massively distributed cooperative process that supports data transmission by means of IP (Internet Protocol). The IP needs resources for moving packets from one point to another using certain pre-computed paths. Most notably it needs IP addresses, which are in fact simple integer numbers. It also needs AS numbers, which are also only integers. Apart from these virtual resources the protocol needs physical resources like processing power, packet buffers, configuration and pre-computed routing tables that direct packets along their way towards the destination.

There are few organizations that coordinate use of the number resources in the Internet, most notably IP addresses and ASNs. The use of these resources in the DFZ (Default Free Zone), which is a synonym to the Internet, is a subject of policies that are being developed in open communities, like RIPE community, by the "bottom-up" process.

There is of course no technical nor formal obstacle for anybody to configure anything that is technically possible on any router or host in his or her network. But when the host or, more importantly, the router connects to the Internet it has to follow certain rules established by the Internet community. The idea is that the Internet itself is defined by the community that connects together and that forms a huge inter-network. The coordination role is being performed by service organizations that follow policies that have been created by the community. The coordination bodies do not have an authority over the resources in the enforcement sense. Since it is not possible for any single organization or company to enforce even the most fundamental policies, it is up to the community to adhere to policies. In fact policies become reality by themselves to certain degree, because there has been a consensus on them in the policy development process, that involves the parties that are expected to implement them. When a majority of the Internet, or at least when a majority of a local network environment implement the policy, it becomes mandatory for everybody in their network-wise proximity. Alternative is to cease the interconnection and effectively stop the data flow for those parties, which decide not to accept the policy.

In fact it is technically possible to disobey community rules and ignore any authority and coordination bodies to certain degree. Some people actually do that in order to gain some extra profit while still not breaking the whole thing, so nobody notices. And even if somebody learned about the wrongdoing, he or she would not be able to do anything about it.

1.2 Hierarchy of coordinators

To operate a huge network, as today's Internet is, the community of network operators established a hierarchy of coordination bodies that both support the community in creating the policies for operating the network and execute the policies.

The coordination of the most important resources - IP addresses and AS numbers starts at the top level in IANA (Internet Assigned Numbers Authority), which is a department of ICANN, a not-for-profit private American corporation that oversees not only IP address and AS number allocation but also different IP constants and DNS resources. IANA operates according to policies that are created on basis of consensus among RIRs (Regional Internet Registry), the only direct recipients of resources that IANA hands out.

There are five RIRs, each serves its own service region that is loosely equivalent to a continent: AfriNIC for Africa, ARIN for North America, APNIC for Asia and Pacific region, LACNIC for South America and RIPE NCC for Europe, Middle East and parts of Central Asia. All RIRs operate according to policies that are being developed and agreed upon by their communities and based on consensus amongst the community members. In case of RIPE NCC the community is open to everyone, regardless of network size or status of LIR. The RIPE NCC is the executive body, based in Amsterdam, Netherlands, while the community that meets twice a year on RIPE meetings and develop policies in several working groups and mailing lists is simply called RIPE.

Even though the community that creates the policies is open to everyone, there are, on contrary, formal requirements for any company to receive resources from RIPE NCC. The company needs a status of LIR (Local Internet Registry) to deal with RIPE NCC. The LIR can obtain IP addresses either for their own networks and their customers, which they directly manage (meaning they route the traffic for these IP addresses) and that is called PA (Provider Aggregatable) resource. Or the LIR can obtain PI (Provider Independent) resources for any other network that fulfills certain criteria. The PI addressing resources are being used by the end customer's network and the end customer is also responsible for routing the traffic and the LIR's role is usually limited to dealing with RIPE NCC during the allocation process. The two types of addressing resources, PA and PI, are not technically distinct, but there is a substantial difference in their formal handling in policies.

In case of PA allocations, another level of authority transfer is possible: The LIR might make sub-allocation out of its allocation to another party. The recipient of the sub-allocation does not need to have any formal relationship with RIPE NCC and the LIR is still responsible for the sub-allocated IP addressing resources.

The same process works for ASNs (Autonomous System numbers) with the difference that the ASN can not be partitioned. IANA allocates ASN blocks to RIRs and RIRs allocate single ASNs to LIRs or their customers one by one. ASN allocation process is in fact more like the PI allocation, even in case when the final recipient of the resource is in fact the requesting LIR.

I have described basics of the process in RIPE NCC, but it is roughly the same for all RIRs. Though there are differences among RIRs in the rules, fees, requirements, time frames, in terminology and evaluation procedures to certain extent, the basic idea I described in this paragraph, stays the same.

1.3 Resource utilization

The most important resources for operating IP protocol are the IP addresses and ASNs. Each party that wish to connect to an IP network need at least IP addresses for addressing the devices and a way to route traffic. Apart from physical interconnection to the network the connected parties require routing protocol to generate routing tables in the network. There have been several smaller, but still large-scale, private networks that operated IP with various routing protocols. And there might be some isolated networks with exotic routing protocols running even today, though operation of isolated networks is considered much more expensive than simply participating in the Internet.

Today's standard for routing traffic in the Internet is BGP (Border Gateway Protocol). It is an EGP (Exterior Gateway Protocol), that is being used for exchanging routing information among autonomous systems. Internet is technically an interconnection of large amount of networks that use common routing infrastructure called DFZ (Default Free Zone).

In order to use an addressing resource in the Internet it has to be announced by some autonomous system to DFZ. BGP is being used for transmitting announcements as well as forwarding other parties' announcements learned from another BGP neighbors. This process cooperatively creates DFZ routing table that contains, in theory, all information needed to route traffic to any possible destination in the Internet. In fact there is a minority of addressing resources that fluctuates in DFZ or are even unreachable due to technical, political or economical reasons.

1.4 Records and public databases

The resources needed for operation of IP are coordinated and allocated by IANA, RIRs and LIRs and used by end users. The whole process is documented and there there are publicly available records about it. The records contain information that is needed to certain extent to operate IP correctly and to troubleshoot the routing system or services running above the network layer. The validity and accuracy of the records are subjects of shared responsibility of the parties that allocate the resources and the parties that use them.

On the top there are lists of IP address blocks and ASN blocks that IANA has allocated to RIRs or that has been allocated directly to end users before the RIR system was invented back in 1996. The current system that works since then is described in RFC 2050/BCP 12 [2]. IANA lists are publicly available on the web site of IANA in form of plain text file and CSV [3]. It is actually quite short list considering the fact that it describes usage of 2^{32} IPv4 addresses, certain portion of IPv6 addresses, which may top 2^{125} of individual addresses under current policies and up to 2^{32} ASNs.

On the RIR level the records are much more interesting, because RIRs have to deal with substantially more information that captures resource usage in much more detail and that are changing much faster than the top-level allocations. Besides that the records of RIRs usually contain some personal and confidential information. The RIR allocation data has to be structured and indexed in order to allow searching for records, using certain criteria for them in order to support

operation and troubleshooting of IP in scale of the Internet. The scale and detail level as well as the way of storing records is based on policy created by the RIR's community.

On the LIR level the record keeping is mandatory because of rules imposed by the community through the coordination bodies. The rules are enforced as a part of the contractual relationship among the RIRs and the LIRs. There are both mandatory public records that LIRs are obliged to publish as well as mandatory records that the LIRs have to keep internally and provide upon request. Mandatory internal records policy applies to IPv6 protocol family, because prefix boundaries are too small compared to the allocated address space to the LIRs. It would cause huge administrative burden and it would be technically challenging if the RIRs had to keep these records in the public databases operated by the LIRs themselves to the level of detail that might reach subnet allocations for the end networks.

On the sub-allocation level there are generally no additional rules and the overall responsibility for keeping records up to date stays with the LIR. Despite that there are mechanisms that make it possible to transfer some authority and responsibilities towards the customer or downstream network in general.

1.5 Routing registry databases

Keeping records that describe resource assignments and link resources to their users is a mandatory process, though differences in policies of RIRs exist in extent required, format and another constants. The obligation to keep records up to date is based on contractual relationship among RIR and LIR and among LIR and the end user.

On contrary, records that describe technical details of resource usage, most importantly routing, are not mandatory or at least not to that extent and not because of the same relationships among RIRs and LIRs and/or end networks. The resource usage on the Internet is represented in routing policies that network operators can publish in routing registry databases. There are several routing registry databases, some of them are accessible to general public and some require certain form of membership (both paid and formal) in order to publish policies, but are usually accessible to general public for reading. And there are certainly some private databases that are used for internal processes inside network operators' organizations or for keeping records of customers in a transit network. Public databases (along with certain processes and rules that are bound to them) are commonly referred as IRR (Internet Routing Registry).

The public databases were at certain point able to mirror each others' data, so the IRR was or maybe still is sensible name for the whole system. But mirroring of data is rather complicated thing because different databases keep data in slightly different forms despite that theoretically a common format - RPSL is defined in documents RFC 2622 [4] and RFC 4012 [5]. Another problem is that even though operators are strongly discouraged from registering their routing policies in more than one routing registry database and the result, when the same routing policy is registered in two different places, is not defined, it might still happen and in fact it happens. Document RFC 2650 [11] explicitly says that this situation may lead to confusion of tools and therefore mislead anybody, who tries to verify the

routing policy.

The document RFC 2650 was created in 1999 and since than the differences among routing databases deepened. Some even ceased to exist and new ones have been created. Nowadays (June 2015) there are at least 34 known routing databases. Comprehensive, though not authoritative list is published on web by Merit Network, Inc. [6]. Relations among them, methods of replication and rules and guidelines for each database are generally unknown, but the Merit's website offers valuable hints.

Among the most important routing registry database are the databases operated by RIRs, and the most important for us is RIPE DB.

1.6 Network operation process

Operating IP network is a task that needs many different resources, among them the number resources plays important role. Some of the resources have to be obtained prior to physical network setup and another could be added in increments to meet demand of the network's users. The operator of the network is responsible for obtaining the resources from RIR that serves his region through the LIR, that has the contractual relationship with the RIR for this reason. Task of the LIR is to provide help to the network operator with requesting resources and with technical and administrative part of resource usage. The LIR have to keep track of PA resources and remains responsible for keeping the records in public databases up-to-date. In case of PI resources the responsibility stays with the resource and therefore it is up to the network operator, rather then the LIR, which has provided help only with setting up of the resource. When the resources are allocated, the network operator can commence using them in DFZ, which means announcing the IP address blocks to DFZ and using AS number as an originator of the IP address block announcements. The LIR in cooperation with the operator, or the operator itself is supposed to fill in objects that describe assignment and purpose of resources into the RIR database. Then the operator is supposed to create and publish routing policy in one of the public routing databases. And both the LIR and the operator should keep the public information up to date.

Reason for having LIRs in the loop is that the LIR is supposed to be well equipped with knowledge of RIR's policies and certain experience that makes the request process, as well as dealing with another situations that concern number resources, more efficient. LIR is supposed to speak on behalf of the network operator in resource request and allocation procedure and is also supposed to help the operator both technically and administratively throughout the process.

In case of PI resources the bond between the resource holder and the LIR is deliberately loose. This option allows certain IP resource holders to be technically independent on the LIR. But the LIR is still needed in the process of resource set up. And in case of RIPE there is also a contractual relationship requirement that demands holders to keep certain relationship with any LIR of their choice and pay small yearly fee to RIPE NCC. This requirement has been imposed in RIPE region in early stage or IPv4 address pool exhaustion. There has been a concern about the abandoned PI space that has been virtually irreclaimable. The contractual relationship serves as an assurance of the continuing need for the PI space as well as a mean of keeping at least some contact with the PI space holder.

Before that the contact information in the RIPE DB often proved to be invalid and there was nothing that RIPE could do to find the resource holder in that case.

2. Routing policies and language to describe them

Routing policy in general is a set of high level mechanisms for describing configuration of an element in routing system, such as a routing protocol or a SDN controller. But the only one current standard for representing the routing policy tool - Routing Policy Specification Language (RPSL) is created only for BGP. The most important documents for the RPSL - RFC 2622 [4] and RFC 4012 [5] repeatedly refer to specific BGP attributes, best-path selection algorithm and even mechanisms like flap dampening. Because of that I have to describe basics of BGP design and operation procedures before continuing to analysis of the RPSL.

2.1 BGP routing parameters

BGP stands for Border Gateway Protocol. It is a path-vector routing protocol, that essentially executes distributed version of a specialized shortest path algorithm with certain unique features that reach well beyond simple graph model.

A vertex in BGP protocol is the Autonomous System, which is identified by an integer, called Autonomous System Number. An autonomous system might consist of several routers that runs BGP with the same ASN. In that case these routers need to cooperate with each other to form a single consistent entity. An edge in BGP is called a session. It is in fact a TCP connection that the router uses to exchange routing information with a neighbor. There are two different modes of BGP sessions: Internal BGP session is a session that links two routers with the same ASN and it is somehow hidden inside the AS, which externally acts as one vertex and its internal structure is not exposed. On contrary external BGP session is configured between two routers with different ASNs.

The BGP configuration reflects point to point nature of BGP relationships as well as the most common physical topology in Internet backbone, which is a point to point physical link between two routers either in the same AS or among different ASes. The BGP session is configuration-defined adjacency (a vertex in the graph) and it is sometimes called informally "peering".

When the BGP session is set on both sides, the routers connect over TCP and after start-up procedure finished, the routing information exchange commences. The most important message in that phase is update that can transmit new information, update existing one or signalize withdraw of a path vector.

Routing information piece in BGP is called a path vector or formally Network Layer Reachability Information (NLRI). It has certain mandatory and optional elements, known as attributes. The most important elements of path vector is a prefix - network subnet that is a subject of the update and an *AS_path* attribute. The *AS_path* contains a list of all ASes that the path vector traversed from its originator. The *AS_path* serves as a loop prevention mechanism - when path vector enters the AS, its *AS_path* is checked for occurrence of the AS and path vectors with the AS in *AS_path* are discarded. Number of ASNs in *AS_path* is also a default metric for BGP best-path selection algorithm.

There are many different mandatory and even more optional attributes. Some

of them are well-known, which means that all implementations of BGP have to support them. Some attributes might be just passed on or discarded when the the BGP implementation does not support them. These attributes allows administrators to define comprehensive rules that modify best path selection process, local BGP state and data in generated or forwarded path vectors to BGP neighbors.

BGP internals are defined in RFC 4271 [7] and many other additional RFCs. Despite that there are variants and additional non-standard features specific to certain implementations. The best-path selection algorithm is one of the major spots for adding vendor-specific features. But the basic operation principle is always the same: Received path vectors are directed through series of filters that removes looping routes, routes with unreachable next-hop and user-defined lists of allowed or denied routes. After that the path-vectors are passed to a series of rules that makes it possible for the user to match arbitrary path-vector and alter some of the attributes. The implementation of this mechanism, its limitations and straight-forward approach, is a major competition point of various BGP implementations. The path-vectors are then stored in input tables (known as Adj-RIB-In) and all available path vectors to each destination are compared by the best-path selection process that selects path with the highest *LOCAL_PREF* attribute. When there is a tie among two or more path vectors, it uses another attributes such as *ORIGIN* or *MED* in predetermined order and when there is still a tie it selects path vector with the shortest *AS_path*. If it is still the same it uses a last resort rule that picks some path vector pseudo-randomly. The resulting routes forms the Routing Information Base (RIB) that transforms into Forwarding Information Base (FIB), which then directs actual packet flow.

The best-path selection process can be altered by two major mechanisms: Either filtering out path vectors, or changing the metric attributes. Apart from the actual path length there are basically two major metrices: Local Preference and Multi-Exit Discriminator (MED). The traffic flows according to the routes in local routing table and therefore the direction of the traffic flow is opposite of the BGP path vector transmission. Local Preference is usually assigned to incoming path vectors by the local configuration, so it prioritizes incoming routes and therefore directs outgoing traffic. MED is usually assigned to the outgoing routes and therefore it is being used for altering path selection process of the path vectors in the neighboring AS in certain situation, which is in fact rather common. Apart from this BGP can carry "community" attribute which is 32-bit integer without a defined meaning (there are few exceptions) and the administrators are free to define values and their meaning by themselves. Community attribute can be also matched by filters and freely manipulated and/or passed to another hops or cleared on the spot. Therefore fairly complicated actions can be achieved by utilizing BGP communities for arbitrary signalization.

Process of sending a path vector to the BGP neighbor is almost the opposite: The path vector is picked from the RIB, directed through series of rules and filters and then transmitted over the BGP session. The level of control over the outgoing path vectors is even greater than over the incoming ones. And there is a mechanism for injecting (originating) new path vectors. By originating the path vector the AS injects a route into DFZ, provided the route is not filtered out along its way. The announcement actually summons traffic destined to the subnet that is a subject of the originating announcement. Basically any router can send out

any prefix and it would eventually spread out through the BGP connected nodes to the all routers in the network, unless some filter denies it. This feature of BGP is the reason why originating a new prefix is somehow delicate operation.

BGP has comprehensive instrumentation and debugging tools. The path vectors inherently carry a lot of system information. Although the data from BGP reflects current operational status of the network and in case of DFZ it means the current state of the Internet, preliminary critical analysis is needed when reasoning about the data. The reason for caution is that the DFZ is massive distributed database of path vectors that contains valuable information that in fact drives packets over the Internet. Changes in it might be monetized, either by respectable ways, simply because optimization pay off, or because there is wide range of possible malicious tampering with BGP.

2.2 Routing Policy Specification Language

The first version of the language, which evolved over time to the current RPSL, was defined by the document ripe-81 [9] in 1993. The definition refers to the document ripe-60 [8] from 1992 that describes routing and associated issues in evolving European Internet at that time. Both documents reflects early phase of the Internet development in Europe. Some of ideas about routing in these documents are outdated from today's point of view. And there are some entirely missing mechanisms that current Internet could not live without.

It is fairly hard to interpret obsolete standard from 1993, but it is obvious that the very basic view of the Internet was quite different than today's reality. It seems that ambition of the RIPE community was to control BGP routing in the Internet in truly open a public manner. And it seems that this was the reason for developing comprehensive routing policy specification language that could serve as public statement of AS's intentions and at the same time also as a primary data source for configuration of the Autonomous System Border Routers (ASBR). The document states that the main requirements for the new RPSL are clarity, translatability, checkability, applicability and generality and defines these features in detail.

RPSL development continued and the result was the document ripe-181 [10] from 1994 that brought the RPSL to better shape. It contained more features than ripe-81 and more closely resembled nowadays standard RFC 2622, except for several constructs that has shifted its meaning, has been renamed or replaced since then. The changes are too substantial to summarize but the trend is obvious: The language has become more complex to allow creating compact descriptions of complex networks. Main tools for that are recursion and *-set* objects. These, combined together, make possible to logically partition the network and use chains of symbolic names to define filters.

RFC 2622 [4] is the current standard for RPSL that has been amended by RFC 4012 [5], sometimes called RPSLng, in order to add support for IPv6 and any possible future protocol or address family. The basic principle stays the same in both of these documents and it still derives from ripe-181. Basically it has a bit more levels of indirection and there is greater expressive power of the language since ripe-181.

2.3 RPSL elements

Routing policies are expected to be stored in public databases. Basic unit of RPSL information is called an object. The object is a piece of data that has its type, primary identifier that is being used for searching and referring to the object in database, optional secondary identifiers that might be also used as lookup keys and a set of mandatory or optional attributes that carry additional information.

Objects are represented in text format that is supposed to be easy to read for humans as well as easy to parse for programs. The text format consists of inactive lines (comments), object separators, which are empty lines and object lines. Object line is either attribute and value line or it might be a continuation of a multi-line value. Objects are separated by one or more empty lines.

Set of allowed attributes and mandatory attributes can be determined based on the object type. Object type is actually the name of the first attribute and its value is usually the primary identifier of the object. Exceptions exist in objects derived from RPSL format, though the exceptions are not present in RPSL, as it is defined in RFC 2622.

Example of a simple RPSL object is:

```
route:          217.31.48.0/20
descr:          Network of Ignum s.r.o.
descr:          Czech Republic
descr:          http://www.ignum.cz/
origin:         AS29134
mnt-by:         IGNUM-MNT
created:        2003-06-12T11:37:52Z
last-modified:  2008-04-16T21:25:49Z
source:         RIPE # Filtered
```

The objects that contains sensitive information, like personal data, might be filtered by the database output front-end, which is usually a *whois* server or some sort of RPC server, in order to remove private data from them. Write access to the object is determined by the *mnt-by* attribute, but this mechanism depends on database implementation and it is not important for RPSL itself.

2.4 RPSL object types

The basic types of RPSL objects that forms the core of routing policy expressions are *route* objects and *aut-num* objects. The *route* object carries information about the announcement of a prefix into the Internet routing system (DFZ) and states which AS is expected to originate the announcement of the particular prefix. The *aut-num* object serves two purposes: It is a record that describes administrative assignment of an autonomous system number to a specific organization and it links proper contact information. It also represents a starting point for definition of the autonomous system's routing policy. The routing policy is described by *import*, *export* and *default* attributes (sometimes called "peering expressions"). The *import* and *export* attributes combined together contains potentially full information needed to set up a BGP session for each partner of the AS and configure the

filters and route-maps for BGP, the mechanism for altering various metrices and attributes in path vectors, that can customize the BGP operation.

The peering expressions in *aut-num* objects might contain symbolic names that has to be resolved to sets of subjects that carry the referred information. These symbolic names are references to *-set* objects. A *-set* object might contain one or more references to another objects of the specific type or to another *-set* objects. In that case the objects form a graph that has to be traversed and all information has to be collected from its nodes before the filter might be applied.

Possible loops in these object graphs are not addressed in the the standards. But even if the loop is not a syntax error it is almost certainly a semantic error. Despite that, there are several loops in the RIPE DB data and therefore parsing software has to be able to deal with that.

2.5 Filters in RPSL

The routing policy in RPSL for a single AS has to fit in *import* and *export* multi-value attributes in *aut-num* object of the AS. Both *import* and *export* attribute types have two basic parts: Peering selector and action. The basic example follows:

```
aut-num:  AS29134
as-name:  IGNUM-AS
descr:    Czech Republic
...
export:   to AS6939 announce AS-IGNUM-OUT
import:   from AS6939 action pref=384; accept ANY AND
 NOT fltr-bogons
export:   to AS5580 announce AS-IGNUM-OUT
import:   from AS5580 action pref=384; accept ANY AND
 NOT fltr-bogons
```

The idea of the first *export* line is to define a BGP filter in the outgoing direction from the **AS29134** to the **AS6939** (selector part), stating that the **AS29134** announces or intends to announce prefixes that match the filter defined by the reference to *as-set* object **AS-IGNUM-OUT**.

The first *import* line says that **AS29134** accepts or would accept routes from **AS6939** with *local preference* set to **384** that match the filter **ANY AND NOT fltr-bogons** which is simple compound filter that references *filter-set* object **fltr-bogons**.

These basic principles are extended in several ways that works accordingly for both directions. Both *import* and *export* objects could contain information about peering configuration:

```
export:  to AS6939 216.66.80.241 at 216.66.80.242 announce
 AS-IGNUM-OUT
import:  from AS6939 216.66.80.241 at 216.66.80.242 action
 pref=384; accept ANY AND NOT fltr-bogons
```

The selectors might be extended multiple statements that shares the same filter:

```
export: to AS6939 216.66.80.241 at 216.66.80.242;
        to AS6939 216.66.81.129 at 216.66.81.130;
        to AS5580
        announce AS-IGNUM-OUT
import: from AS6939 216.66.80.241 at 216.66.80.242 action pref=384;
        from AS6939 216.66.81.129 at 216.66.81.130 action pref=256;
        from AS5580 action pref=128;
        accept ANY AND NOT fltr-bogons
```

The peering selectors can be encapsulated into *peering-set* objects:

```
aut-num:  AS29134
...
export:   to PRNG-IGNUM-UP announce AS-IGNUM-OUT
import:   from PRNG-IGNUM-UP accept ANY AND NOT fltr-bogons

peering-set: PRNG-IGNUM-UP
peering:     AS6939 216.66.80.241 at 216.66.80.242
peering:     AS6939 216.66.81.129 at 216.66.81.130
```

The IP addresses of the routes could be encapsulated in *inet-rtr* objects. And there are *rtr-set* objects for making lists of router references. Examples of these constructs are in the RFC 2622.

There are three basic filtering mechanisms that can be used in filter statement: Origin AS match, IP prefix filter and AS path regular expression match. These components can be encapsulated into *filter-set* objects and the origin AS matches into *as-set* objects. Following example shows compound filters that references a recursive *filter-set* object:

```
aut-num:  AS29134
...
export:   to PRNG-IGNUM-UP announce AS-IGNUM-OUT
import:   from PRNG-IGNUM-UP accept ANY AND NOT fltr-bogons

as-set:   AS-IGNUM-OUT
members:  AS29134
members:  AS51278
...

filter-set:  fltr-bogons
filter:      fltr-unallocated OR fltr-martian

filter-set:  fltr-unallocated
filter:      {}

filter-set:  fltr-martian
```

```
filter:       { 0.0.0.0/8^+, 10.0.0.0/8^+, 100.64.0.0/10^+,
              127.0.0.0/8^+, 169.254.0.0/16^+, 172.16.0.0/12^+,
              192.0.0.0/24^+, 192.0.2.0/24^+ , 192.168.0.0/16^+,
              198.18.0.0/15^+, 198.51.100.0/24^+, 203.0.113.0/24^+,
              224.0.0.0/3^+ }
```

There is also a possibility to define refinement and exception to the filters and to nest them. It is therefore possible to define refinements to refinements, exceptions to exceptions, exceptions to refinements and so on. In principle *refine* statement means that both filters - left and right side of the *refine* keyword have to match in order to execute action in the refinement. Example from RFC 2622 shows simple use of refinement:

```
import: { from AS-ANY action pref = 1; accept community(3560:10);
            from AS-ANY action pref = 2; accept community(3560:20);
          } refine {
             from AS1 accept AS1;
             from AS2 accept AS2;
             from AS3 accept AS3;
         }
```

Which is brief version of the following equivalent filter:

```
import: {
   from AS1 action pref = 1; accept community(3560:10) AND AS1;
   from AS1 action pref = 2; accept community(3560:20) AND AS1;
   from AS2 action pref = 1; accept community(3560:10) AND AS2;
   from AS2 action pref = 2; accept community(3560:20) AND AS2;
   from AS3 action pref = 1; accept community(3560:10) AND AS3;
   from AS3 action pref = 2; accept community(3560:20) AND AS3;
}
```

The function of *except* operator is to selectively change the result of more general filter on the left of the keyword and add a finer filter that precedes the general one in evaluation order. To set relations among the filter expressions there are quite complex rules, how to apply multiple levels of nested *refine* and *except* operators.

Last important group of element in filters are pre-defined words. The most important filter is perhaps *ANY*. Another important and widely-used keyword is *PeerAS*, which matches only the same origin AS as the AS in peering selector. But there are more additional actions and matches that are not essential for the language itself. We are going to discuss some of them later, rest is nicely described in RFC 2622.

3. RPSL basic problems

The documents RFC 2622 [4], RFC 4012 [5], RFC 2650 [11], the IRRToolset software [12] and another libraries and documents, that concern RPSL, formed an impressive ecosystem. Though, there are quite severe problems: The whole idea, standards and software have been created back in 1990's with respect to the Internet at that time. The only update of the RPSL since then was RFC 4012 that added support for IPv6, multicast and a new level of extensibility that was actually never used. The fundamental approach, basic syntax and semantics of RPSL stays almost the same as in the ripe-181 [10] that came out even before RFC 2622. But the Internet, routing and associated technologies evolved rapidly since then. Therefore operation principles and best practices as well as business environment nowadays is completely different from the mid-1990's, when the RPSL was created.

3.1 Real-world usage of RPSL and existing tools

The most prominent tool for processing RPSL is IRRToolSet [12]. It was originally written at the Information Sciences Institute at the University of Southern California, as part of the Routing Arbiter project. Responsibility for the IRRToolSet project passed to the RIPE NCC in 2001 and multiprotocol support, which is basically the support for IPv6, was added during the RIPE NCC period. Then the Internet Systems Consortium, took over the project in 2004 in order to provide long term support of the stable organization.

The IRRToolSet has been written in C++ and it consists of several tools and common libraries shared among them. There is a basic low-level tool intended for reuse in scripting and for debugging called *peval*. There is also syntax checking tool called *rpslcheck*. The most important tool in the suite is *RtConfig* that translates routing policies into configuration formats for several router operating systems, most notably Cisco IOS and JunOS. There are also tools that can be used to compare routing paths in BGP with RPSL policies or construct such tools. Most notably *prtraceroute* that can validate actual routing in an IP network discovered by traceroute-like method and *prpath*, which lists available paths that can be matched in BGP tables.

Apart from IRRToolSet there is a PERL module called RPSL::Parser [13], which is only basic syntax parser and it lacks mechanism for full interpretation of the filters as well as mechanism for recursive resolutions of the filter elements.

Another attempt to create complete library and tools in PERL for parsing and transforming RPSL into router configurations was my own project called BGF [14]. The project is basically abandoned now despite the fact it reached fairly mature status at its time.

There is also (presumably incomplete) support for RPSL build in NOC project [15]. And there is a myriad of small tools, written for single purpose, put to GitHub or kept without proper licensing somewhere on web, that can parse RPSL, reuse any mentioned general tool or do something that concerns RPSL.

To summarize the paragraph: The first and most important implementation of the RPSL is IRRToolSet that sets de-facto standard for routing policy format

and interpretation as well as the API standard for another tools and scripts that cooperate with a RPSL parsers.

I think it is worth noting here that creation of RFC documents and creation of the software that implements them is somehow connected process. Basically a document that defines technical standard is highly unlikely to become RFC unless there is one or more existing software projects that implement the standard. This methodology is as the matter of fact an integral part of procedures in IETF described in RFC 7282 [16]. The statement literally says: "We believe in: rough consensus and running code.".

3.2 Using RPSL in actual network operation

The previous sections described the BGP routing and its parameters, the language for creating descriptions of fairly complex BGP routing settings, the routing databases and basics of data maintenance procedures. The last and the most important point is combining it all together and making use of routing policies for configuring the routers and checking the routes along their way.

The idea of above mentioned RFC documents was to use RPSL as a primary configuration source of all external relations of an AS. It means that the AS operators are supposed to create their routing policies by filling in their *aut-num* objects and creating proper *-set* objects that might reference sets of other parties. Then they should create records for all their routes that are going to be announced from the AS. Final steps should be checking of the configuration and then generating of configuration files for their ASBRs out of the RPSL.

In reality it works differently: AS operators are obliged to keep records of their resources either directly because of their LIR status or indirectly because they have a contractual relationship with some LIR that mandates them to do so. Certain AS operators feel no need to keep any other records in addition to the obligatory ones. This attitude usually slightly changes when the AS becomes a member of some internet exchange (IX) and the AS operator is confronted with rather chaotic environment of multiple peering sessions that changes rapidly. And most importantly the AS is confronted with dozen peering partners that have different peering policies and requirements. Some of them require the AS to have a defined routing policy to a certain degree of detail in order to set up the peering. Few major transit network operators and some internet exchanges require connected ASes to have defined routing policy quite thoroughly. But usually it is sufficient to have either published *route* objects that the AS originate in some routing database and explicitly list the prefixes that the AS is going to announce - either because the AS is the originator or because the AS is transit path for that prefixes. Form of the required statement varies, but it is usually sufficient to send the list by e-mail in an arbitrary form.

Even though the same information should be provided by *aut-num* objects in conjunction with *route* and all *-sets* object in public database, it is possible that the ISPs and IXes that require their peering partners to send them lists of their prefixes by e-mail might use RPSL as storage format and might even put these filters in a public database, creating effectively, but not technically, a duplicate. Or they can do something completely different and it is hard to tell whether there is some common practice or how does it work in reality across

the Internet community. Moreover, the public databases are probably flooded with unused, aged and defunct data that their authors failed to remove when the data became outdated. My educated guess, based on discussions about the routing policies with representatives of different ISPs and IXes of various size and backgrounds, is that only very few of them use RPSL as their primary storage format. Others keep some proprietary databases in various forms ranging from simple Excel tables to some enterprise network management and orchestration software and database. But even the few companies that use RPSL probably do not reuse information that is already present in public databases. Either because they want to be totally in control of their own routing, so they are unwilling to link any data that somebody else might change, or because they rely on one particular database and the information they need might be potentially spread across another ones. Actually people seem not to expect their *-set* objects might get linked by another parties and therefore they make the reusing of the information harder. This works like a spiral - harder the data are to reuse, more bloated and unreusable the database content gets.

There are reportedly few ISPs that actually use RPSL as their primary data source for provisioning, assets management and network configuration. Among them the most interesting is Deutsche Telekom AG [17], which is considered to be a pioneer of RPSL usage, at least in RIPE NCC service region. As far as I know they use their own private database and software for network management and perhaps some in-house created extensions to RPSL, along with several custom procedures concerning its usage.

Though the move to the RPSL-centric network management was strongly encouraged in the RIPE community in the past, there are only a few companies that did the transition at the time, when the RPSL was new and promising. Since then the complexity of RPSL language and operation procedures proved to be too complicated for small and even bigger ISPs. And on contrary the limitations of RPSL discouraged huge international and intercontinental carriers from implementing it, so RPSL never took off and never reached popularity that has been anticipated by its creators.

Besides the RPSL language specifications there are also several RFC documents that describe RPSL usage procedures, most notably RFC 2650 [11]. The document describes couple of basic scenarios for RPSL usage and it also provides a guide to *RtConfig* as a part of the RPSL-centric network management. Unfortunately it seems that most network administrators do not really understand RPSL features and do not appreciate existence of the explanatory documents, like the above mentioned RFC. Next chapter provides detailed analysis of RPSL usage and confronts RPSL data with the actual BGP state. I think that we can assume that apart from extra-small ASes, where administration of the network might be partially outsourced, any reasonable ISP has some department which is responsible for BGP border routers and it should be responsible for publishing routing policies in public databases. But it seems that it does not work like that even in the biggest ISPs. The routing policies are underestimated and viewed as an extra "paperwork" that the company need to do without any apparent reason.

3.3 RPSL parsing

The approach to parsing RPSL data and interpreting routing policies described in RPSL depends on required output data and on input data source. Two types of possible sources are either online service - usually *whois* server or more modern variant that operates over HTTP in form of a *RESTful* service. Other possibility is to have data in form of files that contains snapshot of the database in some known form. Both data sources have specific advantages and downsides. The remote access over *whois* or HTTP is better for small number of requests that are needed for constructing a configuration of ASBRs in one AS or for checking some particular path. Online protocols provide actually some help for these tasks by sending referenced objects along with requested objects semi-automatically. And the access protocols also offers attribute search, which is content-specific or at least aware of the content.

On contrary local data are better for bulk requests and there is an obvious advantage in that it makes it possible to create own index or even transform data to more efficient data structures. This makes sense only in case we can amortize the data processing in the subsequent tasks.

Generating router configuration for an AS or IXP means retrieving the routing policy for the AS, which might refer to a huge graph of *-set* objects. The graph might contain cycles because the language allows them, but cycles do not make sense semantically. Therefore possible cycles are irrelevant for resolving the routing policy, because we need to traverse each vertex in the graph exactly once and put information from the vertex into a flat data structure. Network transactions for resolving routing policy of one AS are essentially cheap for the client of the online data source, because the server might anticipate next queries for linked objects and send more replies at once, saving time and bandwidth of unnecessary requests.

Verification of a real BGP path is a different story, because we need to resolve certain portions of routing policies that concern many different parties along the path for deciding on each step. Generally BGP path verification is a bit more complex from the data processing point of view, but it is still possible to verify an individual path by requesting data remotely. But it is not possible to do the same thing for bulk processing of many or even all paths in the Internet. Either because the network and server latency would inflate time to unacceptable length or because public databases usually impose some limits on request rate and total amount of requests per day, which would not certainly be enough for that.

For verification of all BGP paths in DFZ we would need complete data from all routing databases. The data are hard to obtain from certain databases. On the other hand RIPE DB is publicly available and it is also the most relevant routing database from our point of view. Anonymized content of RIPE DB, with e-mail addresses and other sensitive personal information removed, is available through FTP server of RIPE NCC and it is easy to download and make a snapshot every day. Therefore it is possible not only to verify current status of BGP routes with regard to current routing policies, but we can do the that for each day in past, for which we have the relevant data.

RIPE DB content is being published in form of text files. Each file contains RPSL objects of a single type. The input format is favorable for parsing each text

file and constructing internal data structure that support quick search according to certain attributes. It makes sense to use hash tables for searching in flat namespaces, like *-set* object names, but for IPv4 and IPv6 prefixes that are one of possible *route* and *route6* object identifiers, it is more complicated. It seems that best solution is to use prefix tree with IP routing aware traversing routines. It means it has to provide matches in the same manner as IP routing mechanism does, most importantly selecting the most specific route for a destination.

Apart from data structures that supports fast lookups for required data, we also need to understand semantics of RPSL objects, which is quite easy with one exception: Filters in *aut-num* objects. The filters are the one single point where the most of RPSL's complexity lies. Filters are hard to parse from text forms, because there are many possible distinct syntaxes for the same semantic meaning. Filters are recursive in the same manner as algebraic expressions. And they can contain unlimited number of references to another objects that can contain fragments of filters, as well as lists of subjects to match. And in fact it is possible to mix filters for importing or exporting routes from or to an AS with filters for origins, AS-paths and lists of prefixes and all of them might contain recursive references. Any filter can be modified by a sub-filter using *refine* or *except* keywords, and of course any subfilter is also a filter, therefore refinements and exceptions might be recursive.

My previous experience and also code of other authors show common patterns in RPSL parsing and usage. Basically it is a layered design with high level of code reuse. At the lowest level are routines for RPSL data retrieval either over network or from local sources. In case of local database, the data management is usually combined with some search engine that keeps indexes and provide basic lookup functions. On top of this level is an internal object representation of RPSL text objects, that parse basic RPSL syntax, but does not understand object semantics. On top of this lies RPSL semantic engine that need not only to parse the filters, but it needs to actively resolve links to another objects along the way. And on top of these libraries sits an application that actually use RPSL data either for generating router configuration, or verifying BGP paths, or for doing analysis of any kind.

4. Evaluation of RPSL usage

4.1 Current attitude towards RPSL

Since the creation of RPSL in mid-1990's the standard has been updated in order to support multiple protocols. It was in fact a generalization of need for IPv6 support. But the basic idea of RPSL has not changed since its creation.

The Internet community started to acknowledge the fact that acceptance level of RPSL is low and that, apart from few exceptions, nobody is using RPSL according to its creators' intentions. And the scale of usage is also quite low. But all these opinions are based on personal experience and examples of networks that community members manage or have to deal with and that exhibit inaccuracies or even lack of properly published routing policies. Even though the community seems to be well aware of problems concerning the routing policies, RPSL and the whole Internet routing system and connected record-keeping, there are still problems that have obviously higher priority. It seems that the community is still focused on IPv4 address run-out and a lot of phenomenons connected with that. Internet routing system is partially frozen by the IPv4 concerns and my personal opinion is that everybody is waiting for IPv6 to take up and to see impact on their businesses rather than putting an effort into network maintenance and automation. Therefore development of new technologies as well as new market opportunities stalled in this particular field, or it has been redirected to another places in networking business.

Despite that there were few attempts to start discussion about future of RPSL. On On RIPE 61 in 2010 Nick Hilliard presented [18] his opinion on RPSL usefulness and expressed his wish to start a discussion about RPSL's reform in order to RPSL become a truly useful tool for future.

But it seems that almost the same concerns about usefulness of RPSL and Routing Registries have been with us from the very beginning. In 2000 there was a discussion that took place in NANOG list [19] and it started by a simple questions "Who should use Route Registries? Why?" and more importantly "Is it worth the time?". Though the responses directly from people involved in Route Registry operation were smart, complete and provided a source of education, it showed that Routing Registries does not have enough data and the accuracy is questionable. The replies are stated in manner, that gives hope for future, when the data will presumably be complete and accurate enough to control the Internet routing. The hope, that has never fulfilled because 15 years later nobody can give better answer to the same questions. And in the following part I am going to prove that data in RIPE DB, which servers also as a Routing Registry, are indeed not complete nor accurate at all. And note that RIPE DB is considered to be one of the most accurate and complete Routing Registries, at least compared to other ones. In fact RIPE NCC service region is supposed to be the most educated about existence and proper usage or RPSL and Routing Registry.

IPv4 run-out and transition to IPv6 is in fact huge challenge for Internet routing system. It seemed that IPv6 is a better environment that would allow network operators to do things in cleaner and in more broad-minded way. But in fact transition to IPv6 is terribly painful process for everybody involved, which

is caused by the fact that IPv6 needs extra expenses and work. And it brings no immediate business opportunity and no immediate revenue, so the companies do not like it. The managers' dislike for the technology projects to the whole process extensively. In fact there are transition mechanisms that are difficult to grasp even for technical staff. For example things like *6rd* and *NAT-PT* are much harder to operate than dual stack network itself. Provided that dual stack management means having both the old, complex and difficult IPv4 as well as new, clean and easier IPv6 on the same physical network, which obviously causes some degradation of IPv6 advantage, because the network has to be build and has to accommodate to IPv4 in the first place.

Because of the obvious challenges in IPv6 deployment and because we are now in the middle of transition period I decided to gather and process data for both IP protocols separately and I will present both protocols side by side so we can compare trends and discuss possible influences among them.

The latest development in RPSL that intends to make use of RPSL for configuring route servers in IXPs or ASBRs of transit ISPs is the draft *draft-snijders-rpsl-via-00* [20]. The draft attempts to solve a problem with configuration of routing policy in an adjacent AS, which have to act on behalf of its customer by applying routing policy according to wishes of the customer. This is particularly interesting for route servers in IXPs, because RS customers physically loose their ability to filter outgoing routes and differentiate among peering partners by sending their complete announcement to the RS. Route Server is expected to apply some filtering rules to send out the learned path vectors only to the proper peering partners. For the time being this filtering is usually configurable by using quite complex BGP communities that encode what to send and what to filter out in certain directions. But there is a problem with BGP community length - 32 bits. It works only for 16 bits long ASN because RS operators and customers can use the higher 16 bits for destination ASN and lower 16 bits for encoding an action or any other additional information. But with 32 bit ASNs, that are becoming more and more common, that simply does not fit anymore. And even extended communities might prove insufficient in certain situations when there is a need for encoding two ASNs into one rule. The draft brings the idea of using out-of-band signaling in form of RPSL, which could be utilized to set routing policy and therefore create filters for intermediate ASes.

4.2 Data accuracy measurement

4.2.1 Route objects and origins

Among two major points of interest, that the routing policies in form of RPSL objects can describe, the most important is the route origin. When an IP prefix is being announced by its holder into the Internet routing system, the route for the prefix is originated by an AS of the holder or by an AS that the holder of the prefix has appointed. The origin AS is always the first in the AS path, with an exception of quite rare case of route aggregation that happens along the path. It means that in most cases it is possible to tell who announced the route at any place in the Internet, which received the route. And it is possible to bind IP prefix, that has been allocated by RIR or any subsequent coordination level, to an AS,

that has the authority to announce the prefix into the Internet routing system, by the corresponding RPSL *route* or *route6* object for IPv4 or IPv6 respectively.

It is also possible to check validity of routes in BGP with respect to RPSL *route* and *route6* objects. Provided that we are checking only RIPE objects, we have to select routes that are highly likely to be described in RPSL routing policy, which is stored in RIPE DB. Even though it is technically possible to keep RPSL routing policy records in any other public or even private routing database, the RIPE DB should either be the primary source for the data concerning resources allocated through RIPE NCC or at least mirror data from another source that the resource holder found fit to use instead of RIPE DB. Therefore we can filter the BGP table according to IANA list of prefixes and we get status for each prefix. Possible states are **RIPE**, **another RIR** or **legacy**. The prefixes from another LIR are unlikely to have its *route* objects in RIPE DB and therefore we can ignore them at the moment. Legacy result means that the prefix has been allocated prior to the RIR system existence and the resource holder decided to stick with old rules that were valid to date of allocation. Actually it says that they decided not to submit to the new and much more strict regulations of the current RIR system. When it comes to legacy IP space, RIRs offer them a possibility to use resources of the RIR to keep their records using standard tools for free, but they are not obliged to do so, as LIRs are. Luckily both parties feel that it is good to keep records for each and any network that participate in the Internet. But for us it means that we can ignore legacy prefixes as well. The remaining prefixes are supposed to have corresponding *route* object in RIPE DB. For IPv6 there are no legacy prefixes, because the protocol has been invented only after the introduction of the RIR system.

There is an obvious question: Whether the network operators keep the records of their DFZ announcements in form of *route* or *route6* objects in public routing databases updated and in correspondence with the actual state? The question is actually too complex to simply answer yes or no, but we can focus on visible BGP routes and match the routes to RIPE DB. The basic idea of BGP prefix checking against *route* or *route6* objects in RIPE DB is to take each prefix from BGP dump and resolve its IANA allocation. Possible outcomes are:

- Prefix is a part of a RIR allocation other than RIPE (ARIN, APNIC, ...)
- Prefix is a part of legacy allocation
- Prefix is a part of RIPE address pool

We can assume that prefixes that are parts of RIPE NCC pool should have the records in RIPE DB. Then we try to match the prefix to a corresponding *route* object for IPv4 prefixes or *route6* object for IPv6 prefixes. For the prefixes that are supposed to have the route object in RIPE DB there are following possibilities:

- The *route* or *route6* object for the prefix does not exist in RIPE DB.
- There is at least one corresponding object for the prefix but none matches the origin AS.
- There is at least one corresponding object that matches both the prefix and the origin AS.

Apart from purely academic interest in measurement of correspondence among *route* and *route6* objects and routes in DFZ, there are practical reasons for doing exactly these checks. For instance Internet Exchange Points, that operate route servers, often impose filters on imported prefixes in order to check route origins on the route servers. Since the IXPs are truly important Internet nodes, there is a huge pressure on network operators to keep their *route* or *route6* objects in good shape in order to pass through IXPs. The reason why IXPs are checking that is in the fact that these checks are pretty easy to perform and it can prevent route hijacking, which might be result of a dangerous mistake or even a deliberate attack on the Internet routing system.

Example of matching IP prefix and *route* objects follows. The BGP dump:

```
     Network          Next Hop      Metric LocPrf Weight Path
*   217.31.192.0/20 91.210.16.16        11    240      0 15685 25192 i
*                   91.210.16.205        1    240      0 25248 25192 i
*                   91.210.16.3          1    240      0 25192 i
*>i                 217.31.48.125        1    256      0 25192 i
```

There is a route that originates in **AS25192**. The route is a part of **217.0.0.0/8** RIPE NCC pool. And the corresponding *route* object in RIPE DB is:

```
route:          217.31.192.0/20
descr:          CZNIC - NET
origin:         AS25192
member-of:      RS-CZ-NIC
mnt-by:         CZ-NIC-MNT
created:        2002-08-28T14:54:30Z
last-modified:  2009-10-08T11:26:22Z
source:         RIPE # Filtered
```

The *origin* attribute matches the actual origin of the route in BGP.

On contrary randomly picked example of a route that do not have corresponding *route* object in RIPE DB is **217.31.94.0/24**. The route has been announced from **AS29140** throughout the whole period. Following route timeline shows that even though actual path to the observation point changed over time, the announcing AS was still the same and no *route* object existed at any time. Please note that the timeline shows only the first state and any subsequent state changes - either the changes in path or in the *route* object validity. It means that the following timeline says that the object was missing throughout the whole time period:

```
2012-11-04 217.31.94.0/24 (6939 29066 29140 i) route obj not found
2012-12-05 217.31.94.0/24 (5580 29066 29140 i) route obj not found
2013-05-04 217.31.94.0/24 (5588 29066 29140 i) route obj not found
2013-05-10 217.31.94.0/24 (5580 29066 29140 i) route obj not found
2013-10-09 217.31.94.0/24 (6939 29066 29140 i) route obj not found
2013-10-10 217.31.94.0/24 (5580 29066 29140 i) route obj not found
2013-10-11 217.31.94.0/24 (6939 29066 29140 i) route obj not found
2013-11-15 217.31.94.0/24 (5580 29066 29140 i) route obj not found
```

```
2013-12-04 217.31.94.0/24 (6939 29066 29140 i) route obj not found
2013-12-10 217.31.94.0/24 (5580 29066 29140 i) route obj not found
2013-12-14 217.31.94.0/24 (6939 29066 29140 i) route obj not found
2014-01-16 217.31.94.0/24 (5580 29066 29140 i) route obj not found
2014-02-06 217.31.94.0/24 (6939 29066 29140 i) route obj not found
2014-02-12 217.31.94.0/24 (5580 29066 29140 i) route obj not found
2014-03-24 217.31.94.0/24 (6939 29066 29140 i) route obj not found
2014-04-02 217.31.94.0/24 (5580 29066 29140 i) route obj not found
2014-04-27 217.31.94.0/24 (6939 29066 29140 i) route obj not found
2014-05-19 217.31.94.0/24 (5580 29066 29140 i) route obj not found
2014-06-06 217.31.94.0/24 (6939 29066 29140 i) route obj not found
```

It is possible to find more interesting results, where the status changes over time in more aspects of the *route* verification, like in the following example:

```
...
2014-10-27 217.31.112.0/20 (6939 8220 29312 i) OK
2014-11-02 217.31.112.0/20 (6939 12874 29312 i) route obj not found
2014-11-05 217.31.112.0/20 (5580 8220 29312 29312 29312 29312 29312 i)
 route obj not found
2014-11-06 217.31.112.0/20 (5580 5400 8968 29312 i) OK
```

Different route timeline shows that it might change from one error status to another periodically:

```
...
2014-11-03 217.31.128.0/19 (6939 2529 5587 i) AS not match: ripe-db
 orig: ['AS16063']
2014-11-04 217.31.128.0/19 (6939 2529 5587 i) route obj not found
2015-01-21 217.31.128.0/19 (5580 3356 2529 5587 i) route obj not found
2015-01-22 217.31.128.0/19 (5588 2529 5587 i) route obj not found
2015-02-12 217.31.128.0/19 (5588 3356 2529 5587 i) route obj not found
2015-02-13 217.31.128.0/19 (5588 2529 5587 i) route obj not found
2015-02-26 217.31.128.0/19 (5588 3356 2529 5587 i) route obj not found
2015-02-27 217.31.128.0/19 (5588 2529 5587 i) route obj not found
```

This examples are meant only for illustration of the problems. Summary analysis of the verification results is going to be the topic of the next chapter. And the possible failure modes are going to be described in more detail as well.

4.2.2 Path validation

A path, that is bound in BGP to a certain prefix, can be validated not only at its very beginning, as in case of route origin matching, but at each AS along its way from the originator to the observation point that provided the BGP dump. Autonomous System operators are supposed to declare their routing policies and collectively describe any possible path the prefix can take. Of course there are many possible paths, but BGP selects only one, the best at the moment, according to the BGP best path selection algorithm that is part of the BGP specification.

It means that we can see only the best route for each prefix for sure and sometimes a few additional alternatives, if we are lucky. The actual available paths to any destination are obviously unique for each observation point, but there is still some value in validating the paths from a single observation point and there is even more value in creating and publishing OSS software, which does the job. In the Internet, there is some sort of hierarchy, which dictates that geographically near originators are usually reachable through many different ways, especially over Internet Exchange Points, while remote originators, typically from overseas locations, are reachable only over transit networks. The transit networks translates to upstream connectivity for the Tier 2 ISP, which we use for collecting data. And since there are only a few transit providers with world-wide coverage, we can assume that what applies to our observation point, holds for certain portion of networks with the same or convergent transit path, that are in the same locality, which might be Prague, Central-Europe, or the whole Europe. And in fact we want to find rather errors than validate all paths, which would be rather nonsense, because results in next chapter are going to show that it is quite uncommon to have valid paths that matches routing policies over the entire AS path.

The simplest way to describe what we want to validate is to use an example. We have a prefix from BGP: **2.5.0.0/16** with AS path **6939 1299 5511 3215 i**. The prefix is a part of a RIPE NCC address pool according to IANA address space list. We can assume that RIPE DB is the correct place to look for record for this network. First we should do the check for *route* object. It exists and seems to be valid:

```
route:          2.5.0.0/16
descr:          France Telecom Orange
origin:         AS3215
mnt-by:         RAIN-TRANSPAC
mnt-by:         FT-BRX
created:        2012-11-22T09:32:07Z
last-modified:  2012-11-22T09:32:07Z
source:         RIPE # Filtered
```

Then we can examine *aut-num* objects for each AS in the AS-path to see whether the ASes are supposed to announce the prefix to their neighbors and whether the recipient ASes are supposed to accept the prefix before forwarding it further. The first AS to check is in fact the *aut-num* of the originator - **AS3215** and we are looking for *export* line for **AS5511**. The line is present in the *aut-num* object:

```
export:          to AS5511 announce AS-FTFR
```

Remaining step for validating this point is to resolve the filter and decide whether the prefix matches the filter. The filter in our case is a simple reference to an *as-set* object **AS-FTFR**. The object is present in RIPE DB and contains following data:

```
as-set:    AS-FTFR
descr:     ASes announced by France Telecom - AS3215
```

```
members:  AS-RAIN
members:  AS-RBCI
admin-c:  PHC
tech-c:   FTDI1-RIPE
tech-c:   GAB1
mnt-by:   RAIN-TRANSPAC
source:   RIPE # Filtered
```

So we have to resolve the members **AS-RAIN** and **AS-RBCI** to see whether the **AS3215** occurs in their *members* list or in any other nested *as-set* object. After one recursion step we find that the **AS5511** is listed in **AS-RAIN** object, which means that the filter allows **AS3215** to announce the prefix **2.5.0.0/16** to **AS5511**. It means that we have validated this step successfully.

Continuing with the same prefix we can examine *import* filter of the next AS in the AS-path, which is **AS5511**. There is proper *import* line for **AS3215** with exactly the same filter as it was on *export* side of **AS3215** and therefore the filter can be resolved and validated in the same manner. Then we have to check the *export* filter from **AS5511** to next AS in AS-path which is **AS1299**, and so on.

The problem emerges at the last point of the AS-path, because **AS6939** is an American company and its routing policies are unknown for the RIPE DB. There is only some sort of "dummy" *aut-num* object for the AS:

```
as-name:         HURRICANE
descr:           Hurricane Electric, Inc.
admin-c:         DUMY-RIPE
tech-c:          DUMY-RIPE
remarks:         For information on "status:" attribute read
  https://www.ripe.net/data-tools/db/faq/
  faq-status-values-legacy-resources
status:          OTHER
mnt-by:          RIPE-NCC-RPSL-MNT
source:          RIPE
```

This object does not even contain any *import* nor *export* lines. It is questionable whether it would be more proper to mark this kind of foreign AS that has only partial and probably auto-generated RIPE DB record as "missing" or "unknown" or whether it is more proper to try to check the filters and report failure. In my humble opinion the RIPE DB tries to mirror routing policy data that are available and that can be extracted from other databases. Therefore when there is a record, I think I can try to perform validation. In fact we are unable to validate the path trough this AS anyway and it is only a matter of terminology whether we put this to *unknown*, *failed* or to some special bucket for these ASes. As for now, we mark this class of objects as unknown, but the ultimate solution is to take all other databases into account. It would add another level of complexity because despite the theoretical assumption that each object resides in a single database and gets eventually mirrored to another ones it is not always the case in the real world. There is a lot of data duplication among routing databases and even different versions of the objects for the same resources can be found, so we will have to decide which object to use. And it is anything but easy task,

because regions differs and people have unexpected affiliations. This is the reason why this thesis focuses on RIPE DB data and why I keep global data analysis for future, hoping that I would be able to put it aside, if the RIPE DB analysis proves to be sufficient to show the point.

I am going to describe possible outcomes from the path validation in more detail and present validation results in the next chapter.

4.3 Complexity of RPSL

At this point I need to describe typical use of the data in *aut-num* and in subsequently referenced *-set* objects. It allows us not only to check real paths in the Internet, which is in fact only a minor application, but the intended use was to generate filters from perspective of an AS operator, which has also the authority over his/her own records in RIPE DB. In that case, the loop of creating routing policy and using it for configuring routers closes and the publishing of the routing policy for others' reference would be a useful side-effect. However in reality only few networks use RPSL in that way and there is no pressure to keep the routing policies in accordance with reality just for informing others. And there is a positive-feedback: When the data in public databases are incomplete or inaccurate, people tend to perceive them as useless and ignore its existence either for use as well as for keeping their own data up to date.

Another problem with the *aut-num* objects is high complexity of the filters. Even simple constructs in filters are hard to resolve by hand, because there is inherent recursion inside, so checking it is a time consuming and exceptionally dull job. Therefore debugging and maintaining RPSL data is everything but easy task. There are tools for this job, most notably utilities from IRRToolSet, but still the tools are complex and quite hard to use. I think, that complexity of learning RPSL and the tools for using it is comparable to learning a new programming language. It explains why people are somehow reluctant to dive into that.

The RPSL is undoubtedly a complex standard. The contributing factors are:

1. The basic text format does not conform to any modern and widely-accepted data markup standard like XML, JSON, YAML, etc.

2. The language is case insensitive and there are ambiguities in delimiters and white space handling.

3. The language is inherently recursive and it forces linearity of resolution - it is extremely hard to resolve filters sensibly in parallel.

4. Data are spread through different databases that are not linked. There are databases that mirrors another ones, which might help making information accessible, but it can also cause duplication and hold different versions of objects. Actually data might be changed intentionally in the process of mirroring or replaced by some kind of placeholder objects.

5. The language allows numerous ways of expressing a single thing.

6. The language allows to express policies that are not possible to configure on a particular, network, but are possible to check from other party perspective.

7. The language allows to link members both from the *-set* objects in forward direction and in the objects that can be members of any *-set* object by a *member-of* attribute.

8. The filters are difficult to decode, interpret and match BGP path vectors to them. From theoretical point of view it is based on quite simple idea of compound expressions that could match the BGP path-vector values. In reality there are ambiguity elements even in simple constructs, like that the filters can stand next to each other without any logical operator, which is usually interpreted as **or**, though it is not clearly and explicitly stated in the definition documents.

9. Only one type of parentheses is being used both for delimiting the filters and for enclosing filter elements in order to override operator precedence. It means that list of items enclosed in curly braces is in fact a list, which is equivalent to a filter that consists of sub-filters with imaginary **or** operator in between each two elements. On the other hand the curly braces are being used for enclosing prefix filters.

10. Selectors can be also compound expressions that might need recursion to resolve. There might be the compound filter instead of two or more *import* or *export* lines with the same filter but different selector, which is another way of expanding the selector list with **OR** operator.

11. There might be two lines with the same selector and different filters. It seems that this is not an error, but I am unaware of any explicit mention in standard definition documents as well as in best practices and recommendations documents, how to interpret it. Answer on the relation question observed in the reference implementation is: **OR**.

12. Filter lines in *aut-num* objects contains too much optional data fields. In fact the idea is to cram configuration that usually consists of tens of lines in average Cisco configuration to one long line in the routing database.

13. Mandatory but loosely defined naming conventions are being used to distinguish object type in references. For example objects that have name starting with **AS-** are supposed to be *as-set* objects and the similar rules apply to many different types of the objects.

14. AFI specification in RPSLng (RFC 4012) is clearly retro-fitted feature to the existing RPSL and two flavors of RPSL blends together. Fortunately the lines are easy to distinguish because RPSLng lines use *mp-* prefix, but still it adds another way of expressing IPv4 unicast policy - either by old *import* or *export* lines as well as new *mp-import* or *mp-export* lines.

Few examples follow to illustrate some of mentioned factors. Especially recursive manner and backward links:

```
aut-num:    AS29134
export:     to AS6939 announce AS-IGNUM-OUT
import:     from AS6939 action pref=384;
```

```
           accept ANY AND NOT fltr-bogons
mp-export: afi ipv6.unicast to AS6939
           announce AS-IGNUM-OUT
mp-import: afi ipv6.unicast from AS6939 action pref=384;
           accept ANY

as-set:    AS-IGNUM-OUT
members:   AS29134
members:   AS-GSNET-SET-TR-OUT-IGNUM

as-set:    AS-GSNET-SET-TR-OUT-IGNUM
members:   AS47949
members:   AS61047
members:   AS199607
members:   AS61370

route:     217.31.48.0/20
origin:    AS29134

route:     62.109.128.0/19
origin:    AS29134

aut-num:   AS51278
member-of: AS-IGNUM-OUT

route:     195.226.217.0/24
origin:    AS51278
```

In order to find out what the **AS29134** announces to **AS6939** according to the *export* and *mp-export* rules, we have to resolve the filters. The filter is extremely simple in that case - it consists of only one *as-set* object reference. The reference has to be resolved and it contains one ASN and one recursive reference which has to be resolved as well. But there is another ASN, which is a member of the **AS-IGNUM-OUT** object, even though it is not directly listed, because the object **AS51278** contains the *member-of* attribute. This type of filter is quite easy to read and understand, but it forces us to index objects and pre-create links to *-set* objects. And there are not only *as-set* objects but also *filter-set*, *peering-set*, *route-set* and *rtr-set* objects that might be back-referenced from many different objects that are semantically compatible.

Another example of filter complexity has been taken from RIPE DB, more precisely from *aut-num* object for **AS20535**:

```
import: {   from AS-ANY accept NOT { 0.0.0.0/0 }
        AND NOT { 0.0.0.0/0^25-32 };
        } refine {
        from AS-ANY action pref=40; accept community(20535:60);
        from AS-ANY action pref=30; accept community(20535:70);
        from AS-ANY action pref=0; accept ANY;
        } refine {
```

```
from AS13099 accept AS-AET and <AS-AET$>;
from AS28910 accept AS-INTAL AND <AS-INTAL$>;
from AS34639 accept AS-TOTEL and <AS-TOTEL$>;
from AS39214 accept as-comintech and <as-comintech$>;
from AS-INSAT accept PeerAS and <PeerAS$>;
}
```

The filter is quite understandable and readable, despite the fact that it is complex and uses special keywords like **PeerAS**, compound expressions and logical operators, set selector **AS-ANY**, refinement, IP address filter in curly braces, AS path regular expressions enclosed in <> blocks, even *community* matches and the **PeerAS** key in AS path regular expression, which is definitely not common use for the keyword. Even though the syntax and basic ideas seem to be readable, it is hard to resolve the filter by hand, because it references six *-set* objects that might be recursive. The use of *-set* objects in the regular expression is also far from being straight-forward and it is not completely obvious from RFC documents, how it should be handled. Another concern raises because of multiple refinements. The RFC documents do not explicitly state what does it mean and there are two possible interpretations - either the refinement chains or the first rule is refined twice and the first/best/last match wins. To find out we have to examine reference implementation - IRRToolSet.

And the readability of the filters is heavily dependent not only on complexity of the utilized elements and their combinations, but also on text formatting like in the following example that has been taken also from RIPE DB *aut-num* object for (**AS12695**):

```
import: {  from AS8395 accept AS-EAST; from AS8592
 accept AS8592; from AS8752 accept AS8752; from
 AS15672 accept AS15672; from AS16231 accept
 AS16231 OR AS28736; from AS21085 accept AS21085;
 from AS21225 accept AS-AMTKOM; from AS25032
 accept AS25032; from AS25251 accept AS-ARTCON;
 from AS25308 accept AS25308; from AS29124 accept
 AS29124; from AS29182 accept AS-ISPSYSTEM; from
 AS31494 accept AS-INFOSETI; from AS31720 accept
 AS31720; from AS33842 accept AS33842; from
 AS33902 accept AS33902; from AS34121 accept
 AS34121; from AS34211 accept AS34211; from
 AS34352 accept AS34352; from AS34682 accept
 AS34682; from AS34687 accept AS34687; from
 AS34690 accept AS34690; from AS35178 accept
 AS-TELART; from AS35374 accept AS35374; from
 AS35750 accept AS35750; from AS35755 accept
 AS35755; from AS38922 accept AS-Wiland-TP;
 from AS38964 accept AS-ADTEL; from AS39034
 accept AS39034; from AS39150 accept AS39150;
 from AS39165 accept AS39165; from AS41667
 accept AS41667; from AS41917 accept AS41917;
 from AS41947 accept AS41947; from AS42533 accept
```

```
AS42533; from AS42569 accept AS42569; from
AS43327 accept AS-REDLINE-NEW; from AS24758
accept AS24758; from AS48552 accept AS48552;
from AS43666 accept AS-CTS; from AS47711 accept
AS47711; from AS48050 accept AS48050; from AS48946
accept AS48946; from AS43816 accept AS43816; from
AS16300 accept AS16300; from AS43993 accept AS43993;
from AS43414 accept AS43414; from AS39596 accept
AS39596; from AS47839 accept AS47839; from AS34249
accept AS34249; from AS43221 accept AS43221; from
AS33902 accept AS33902; from AS34123 accept AS34123;
from AS5531 accept AS-TEZTOUR; from AS48535 accept
AS48535; from AS49060 accept AS-UNIONLINE; from
AS49400 accept AS49400; from AS49371 accept AS49371;
from AS49779 accept AS49779; from AS50212 accept
AS50212; from AS50265 accept AS-GT; from AS51410
accept AS51410; from AS51464 accept AS51464; from
AS51814 accept AS-KZNET; from AS48147 accept
AS48147; from AS39272 accept AS39272; from AS42293
accept AS42293; from AS52112 accept AS52112;} refine
{ from AS-ANY action pref=700; accept ANY; }
```

The problem is not only lack of spacing and reasonable styling but also abuse of the fact that many filters might be expressed on one line separated by semicolon, which is equivalent to more common practice of using multiple *import* lines - usually one for each peering. In fact the filter was even harder to read in the original form because of less intelligent line wrap in the *whois* output in a terminal.

When similar obfuscation combines with use of complex mechanisms, as in the following example, it is really hard to decode and say for sure what the filter means:

```
import:  { from AS-ANY action community(6774:65231);
 accept ANY AND NOT AS6774:FLTR-BOGONS; } refine {
 from AS6774:PRNG-BE-BNIX action
 community.append(6774:65100); from AS6774:PRNG-DE-DECIX
 action community.append(6774:65104); from
 AS6774:PRNG-FR-SFINX
 action community.append(6774:65102); from
 AS6774:PRNG-NL-AMSIX action
 community.append(6774:65101);
 from AS6774:PRNG-UK-LINX action
 community.append(6774:65103); accept
 (PeerAS OR AS6774:AS-PEERS:PeerAS); }
```

The problem is use of multiple filters in the first curly braces, followed by quite complex refinement that contains many actions. Apart from that, it is hard to resolve the filter because of different *-set* objects are being referenced. And there is a serious naming convention abuse that make it hard to read it even

for machine, because the *-set* objects uses hierarchical naming with colons as delimiters. The reference to the object **AS6774:AS-PEERS:PeerAS** is also tricky, because *PeerAS* is a keyword. The RFC documents do not explicitly forbid keywords from being used as a part of an identifier. Despite that it is quite uncommon and it is in fact the only one instance of this naming mismatch in the entire RIPE DB.

4.4 Expressive power of RPSL

Despite the unquestionable complexity of the RPSL language there are still gaps in the expressive power of RPSL that prevents RPSL from use in certain situations, or it makes the use much less straight-forward or painful in any other way.

When we accept the idea, that the RPSL has been designed to become the primary source for inter-AS BGP peering configuration and examine what parameters the BGP sessions need to be set, we can find that not each parameter can be put to RPSL. Maximum prefix count, which is an important shared parameter, for instance can not be stored in RPSL. It would make sense to publish it in order to prevent quite common problems with BGP session flapping due to incorrect maximum prefix setting. Another example is a BGP password. This is a challenging issue, because it is inherently a security concern. Current common practice regarding the BGP passwords is to exchange them by e-mail and set them manually once for all, which is exceptionally poor method, both from security point of view as well as from network operation point of view. Another example is the peculiar issue of BGP TTL security mechanism, which is known under many different names, based on terminology of different router vendors, The idea is to set the TTL to maximum, which is 255 on both ends and expect exact value, usually 254 on the receiving side, instead of sending with $TTL = 1$ and expecting it to be 0 at the receiving side. It would be useful to be able to announce to peering partners that an AS intend to use this settings, because it is quite hard to debug, especially when a new administrator comes to an undocumented network and is supposed to fix BGP sessions that can not connect without any apparent reason that turns out to be misconfiguration of this mechanism.

The RPSL has been designed with only inter-AS routing in mind and therefore there are virtually no provisions to support intra-AS routing. Even though I am far from suggesting to use RPSL for whole network configuration, I think that inter-AS and intra-AS routing merged to some extent when MPLS has been invented and maybe due to the way how it is being used in service providers' networks. The RPSL lacks tools for describing route propagation in different VRFs within one AS and expects the AS to decide on one single particular best route, that it announces further to another parties, which might not be the case with MPLS and multiple VRFs. Not to mention current heavy use of MPLS-TE and VPLS networks that alters traditional networking paradigm in many ways and it is completely unanticipated by the RPSL standard.

Another possible source of problems with creating meaningful RPSL descriptions is onset of SDN controllers with more advanced routing logic that extends the problem of describing routing policies to even greater scale. On the other hand even the SDN controllers struggle to define a common language for describ-

ing their actions, decisions and algorithms. There are projects that try to create standards or provide compilers for transforming controller programs, most notably the NetIDE project [21]. Aims of the project are nicely described in very educational paper [22]. It is more and more obvious, that the scale of network description itself surpassed a potential of 20 years old RPSL.

In last few years, several ideas have been brought up in certain groups, that BGP could be used to distribute access control lists or any kind of filters in general. The basic idea was to broaden spectrum of tools for customers of ISPs to semi-automatically populate filters to the upstream ISP when the customer's network is paralyzed by some sort of DoS attack that passes through the ISP in question. The idea is to extend Remote Triggered Blackhole mechanism, which is now usually performed by BGP communities that trigger remote blackholing (discarding) of certain traffic. The mechanism is there for suppressing DoS attacks that heavily affects the end network, but passes unnoticed through the upstream because of sufficient oversubscription. In both cases, even in the hypothetical case of using future provisions for ACL distribution over BGP as well as today's RTBH mechanism, it would be nice to be able to describe the mechanism in the standardized manner. In fact it is possible to describe the configuration of the RTBH receiving side in RPSL, but it would end up in quite complex filters.

The last and the most problematic RPSL issue was the support for route servers and transit ASes that are possible to configure or willing to implement custom routing policy for the end network. The original RPSL does not support this. But it is possible for the customer to use BGP communities and the upstream ISP or RS to match the communities and use them in the routing policy. This case is harder to describe in RPSL and understand than the actual configuration of the network equipment and therefore it is hardly justified to put extra effort into writing complex routing policies for this case. Since this problem is known for long time, there is an Internet Draft [20] for expanding RPSL by *import-via* and *export-via* rules in the *aut-num* objects. Even though the draft has not been promoted into RFC / BCP status the RIPE DB supports the rules according to this standard and there is some usage in the actual *aut-num* objects of few RS operators.

Just for sake of completeness I have to comment on an interaction among RPSL, RPKI and in general family of technologies for cryptographically secure BGP, most notably RFC 6480 [23]. Even though I personally do not believe in future proliferation of RPKI in the public Internet, I think that description of the RPKI setup and possible alterations of routing policies according to RPKI verification results should be standardized and put into RPSL. Unfortunately it is not the case. Both standards do not define relation to each other.

5. RPSL and BGP data analysis

This chapter presents selected results of the data analysis, referenced in the previous chapters. The results came from a software called **bgpcrunch**, which has been created as a part of this work. Unlike the results, I do not consider the software to be a particularly important part of the thesis. There is going to be only a brief description of the software internals. But I have published the software under OSS license on GitHub (`https://github.com/tmshlvck/bgpcrunch`) for reference and for possible re-use.

Please note that only the electronic version of this thesis contains appendix with most of the visualized results for the last 3 years. The reason for not including the entire appendix in the printed version is its size - over 3000 pages. Even more results in text form has been included on the attached disc, but it is still not everything. I published the comprehensive results on the web site `http://aule.elfove.cz/~brill/bgpcrunch`. The reason is still the size of the full data set that reaches several hundreds of gigabytes. See the appendix manifest on page 94 for the details on the complete data.

5.1 Available data

The idea of conducting an Internet routing system analysis from two different points of view arose in fall 2011 during an informal discussion on RIPE meeting. The primary concern was to measure not only the meaningfulness of routing policy data in RIPE DB, but also find, what trends can be observed in turbulent times of IPv6 transition. In 2011 the IPv6 was generally deployed in backbone networks, but it was still disabled by content providers and everybody was a bit afraid of turning it on and, at the same time, everybody was anxious about the future of the Internet. The IPv4 address pool exhaustion was imminent at that time and despite the fact IPv6 was around from late 1990's, the community did not have wide and consistent experience with end-to-end IPv6 services. This Internet state was interesting not only from operational point of view, but also as an opportunity to capture dynamics of IPv6 deployment and observe practices, concerning the new protocol in the Internet routing system.

In the beginning the scale of the experiment was constrained by the available resources that I have been provided by my former employer for research purposes. The basic idea was to collect data and store them until the software for its analysis is finished and then seek for more support if it proves to be necessary for processing the collected data.

The data collection started on November 11, 2011. There has been a brief discussion with the thesis supervisor, the representatives of the resource provider and few people who showed some interest in this topic about what data to collect and what to do with them afterwards. I created an outline for the data analysis, which is also the abstract for this thesis. The collection process required some adjustments afterwards. This is the reason why we have the first data that are useful for the subsequent analysis from March 3, 2012. Collected data in the period from November 11, 2011 to March 3, 2012 are incomplete and do not provide enough matter for comprehensive analysis, though some information

could be extracted. But later, the analysis proved to be time-consuming and requiring a lot of resources in general, which discouraged me from putting an extra effort in analysis of partial and possibly inconsistent data from the beginning.

The collected data consists of daily snapshots of BGP table in text form and snapshots of RIPE DB contents. The BGP snapshot is a simple output capture, generated by Cisco commands *show bgp ipv4 unicast* and *show bgp ipv6 unicast*. Example of small portion of the output capture follows:

```
     Network          Next Hop            Metric LocPrf Weight Path
 *>i 1.0.0.0/24       217.31.48.125            0    128      0 6939 15169 i
 *                    195.39.49.133                 127      0 5588 15169 i
 *>i 1.0.4.0/24       217.31.48.125            1    256      0 6939 4826
                                                               38803 56203 i
 *                    91.210.16.201            1    240      0 6939 4826
                                                               38803 56203 i
 *                    195.39.49.133                 127      0 5588 6939
                                                               4826 38803
                                                               56203 i
 *>i 1.0.5.0/24       217.31.48.125            1    256      0 6939 4826
                                                               38803 56203 i
 *                    91.210.16.201            1    240      0 6939 4826
                                                               38803 56203 i
 *                    195.39.49.133                 127      0 5588 6939
                                                          4826 38803 56203 i
 *>i 1.0.6.0/24       217.31.48.125            1    256      0 6939 4826
                                                     38803 56203 56203 56203 i
 *                    91.210.16.201            1    240      0 6939 4826
                                                     38803 56203 56203 56203 i
 *                    195.39.49.133                 127      0 5588 6939
                                                4826 38803 56203 56203 56203 i
 *>i 1.0.64.0/18      217.31.48.125            1    256      0 6939 4725
                                                   4725 7670 7670 7670 18144 i
 *                    91.210.16.201            1    240      0 6939 4725
                                                   4725 7670 7670 7670 18144 i
 *                    195.39.49.133                 127      0 5588 6939
                                              4725 4725 7670 7670 7670 18144 i
```

The first two lines, that follows the header, describe the two possible paths to prefix **1.0.0.0/24**. The most interesting part for routing policy analysis is the path column. In the example it contains two distinct paths over two different upstreams **AS6939** and **AS5588**. We can also see that the originator of the prefix is **AS15169**, BGP *origin* attribute is **i**. The path over **AS6939** is being used, because it has greater *local preference* and therefore it has been resolved by the BGP best path selection algorithm as the *best path*, which is expressed by the $>$ character in the first column. The complete input contains information about all prefixes in the Internet that are reachable from our observation point at the time of capture. Though some data might be less clear and less straight-forward to read, because there are few prefixes that has been created by route summarization, redistribution from EGP or even manipulated by any other mechanism

(which actually should not happen in DFZ). Moreover, for the first three months the router, which has been used for obtaining the data, did not support 32-bit ASNs and therefore path data have been mangled by the backward compatibility mechanism, described in RFC 6793 [24]. But because this data has not been used for the analysis, it does not have any impact on results in the end.

The size of the table dumps varies from 783 555 lines and 67 250 680 bytes in the beginning, up to 1 220 336 lines and 105 552 786 bytes nowadays for IPv4 table. For IPv6 the data size is considerably lower. It consisted of only 30 491 lines in November 2011 and now (June 2015) it contains 181 681 lines. Please note that the BGP table size is larger than the RIB table size because the BGP table contains multiple paths for the prefixes in order to be able to decide on best path and to switch to second-best path when the best path fails. Generally speaking, the BGP table in our case contains at least two possible paths for each prefix, but for some prefixes we might have as much as 10 competing BGP paths.

In order to conduct analysis of the routing policies I have decided to collect RIPE DB snapshots. The RIPE DB contents is available on FTP server of RIPE NCC which makes it easy to create daily snapshots. The FTP files are anonymized in order to protect personal data in the RIPE DB, but the anonymization does not interfere with our intentions. The RIPE DB format is based on text files that contains RPSL objects grouped by the object type.

Overall input data size in compressed form was 198 GB to the date of this thesis compilation. Major part of the data was the RIPE DB dump archive. Results and byproducts of the analysis took 776 GB, while the actual results without byproducts take a little bit over 26 GB. Obviously, it is not possible to compile all the results into the thesis, but I am going to pick the important ones. The rest is in the appendix or on the web site, as described in appendix manifest.

5.2 Data processing

The processing of that amount of rather poorly structured data deserves some attention. The basic decision I made in the beginning was to use scripting language that interact well with many different data analysis tools and that is reasonably fast and support distributed processing to certain degree. After thorough deliberation I chose Python, even though it does not excel in all the mentioned requirements. It proved to be a good choice, because I was able to create many components needed for the data analysis in the consistent environment. And I was able to scale the analysis execution to many computers when I realized that the data processing is really CPU-intensive task and I would not be able to get results in reasonable time on one machine. On contrary I have not utilized any other data processing tools, apart from **R** and **Gnuplot**, both only for visualization and simple reasoning on basic features of the results.

The figure 5.1 shows the basic architecture of the analysis software - **bgpcrunch**.

The basic idea is to take the data through three steps: The first step is called "pre-processing". Scripts take the raw text data, parse them and create few text outputs and very basic Gnuplot charts as well as Python lookup data structures that are being stored into *pickle* files. The saved files are actually much larger than the input files mostly because inputs are compressed and since the raw data

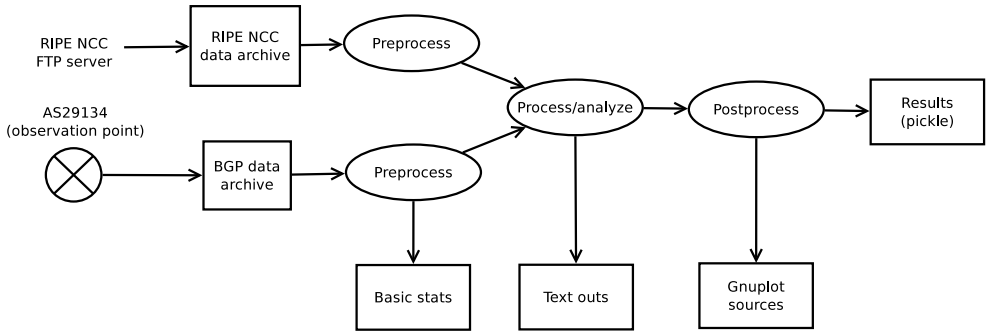


Figure 5.1: BGP crunch architecture

are text files with a lot of repetitions, the compression ratio was pretty high. In fact decompressing the raw input was time consuming and it generated quite a bit of I/O load as well, so I was able to do this on the main server, where all the data resides, in only 6 threads and it took few days to complete.

The resulting *pickle* files with the pre-processed data structures are being used as the input for the main and the most CPU intensive part of the analysis called simply "processing". At this point we can take the path vectors from the pre-processed BGP dumps and match the prefixes and corresponding paths to the *route* or *route6* objects in the pre-created lookup structures and record results to proper *pickle* file. Then we analyze each hop in the AS paths of each path vector in the BGP table and we try to validate the hops in AS path according to filters in the corresponding *aut-num* objects in the pre-created lookup structures. The difficult part is to resolve recursive selectors and then the recursive filters that might refer to many different objects, that are in another pre-processed lookup structures. And the depth of the recursion might be considerably high, which puts strain not only on CPU, but also on memory and I/O as well. This procedure is extremely time consuming. Analysis of the data, covering one day, takes almost a day on a single nowadays state-of-the-art CPU core. Therefore I had to utilize as many CPUs I possibly could and for doing so I had had to make some provisions to be able to distribute the work. The data has been shared among the servers via NFS. Distribution of the work has been semi-automatic in an aspect of distributing work among the servers and it has been fully-automatic at the servers in the aspect of distributing work among all the available CPU cores.

Results of the "processing" phase are still only a few text outputs and several *pickle* files. It is up to the last portion of the **bgpcrunch** software that does "post-processing" to take the analysis results in the *pickle* files and create main text outputs, count totals and generate sources of charts, that can be later processed by Gnuplot. Unfortunately this part has to run in a single thread, because it needs to process the data sequentially and generate the output in the precise order. Of course it could be parallelized to certain degree, but the running time is not that long even on one CPU core to justify any extra effort into that.

The table 5.1 shows running time of the analysis of the data, that covers 1 119 days of BGP and RIPE DB snapshots, starting at March 22, 2012 and spanning up to June 21, 2015.

More interesting aspect of the analysis was the size of the data involved in the process. In the table 5.2 there is an overview of data types and approximate

Phase	Running time	No. of CPU cores
pre-processing	127 hours	6
processing	247 hours	56
post-processing	27 hours	1

Table 5.1: Analysis running time

Data	Compressed	Uncompressed
IANA IP space map	N/A	44 kB
BGP snapshots	4 GB	91 GB
RIPE DB snapshots	194 GB	5.1 TB
pre-processing results	N/A	522 GB
processing results	N/A	306 GB
post-processing daily results	N/A	132 GB
post-processing summary results	N/A	9 MB
post-processing timeline results	N/A	2 GB
Total	198 GB	6.2 TB

Table 5.2: Size of data and products

sizes.

During the data pre-processing I also discovered a bug in RIPE DB software that permitted a syntactically invalid object into RIPE DB. It was unfortunate coincidence that the offending object **aut-num: AS2852** belongs to CESNET z.s.p.o. The problem was the following line:

```
mp-import: afi ipv6.unicastfrom AS39790 action pref=150; accept AS39790
```

The `ipv6.unicastfrom` fragment consists of two keywords `ipv6.unicast` and `from`, that should be separated by one or more white space characters. The bug has been reported and acknowledged by RIPE NCC staff, pending correction.

Apart from this obvious syntax error I encountered two other types of more complex syntax errors that, I believe, breach the standard to certain degree and I noted them for later discussion in mailing lists.

5.3 Changes and trends in the dataset

5.3.1 Global changes

The period, captured by the available data, has been eventful for the entire Internet, because the time frame covers a culmination of IPv4 scarcity and subsequential run-out in all but one RIRs. It also captures major IPv6 events like World IPv6 Launch [26] on June 6, 2012, which is unfortunately not directly visible on the BGP propagation plots, because the IPv6 BGP announcements of the participating parties had been in place long time before the World IPv6 Launch day and the flip of the switch happened rather in DNS.

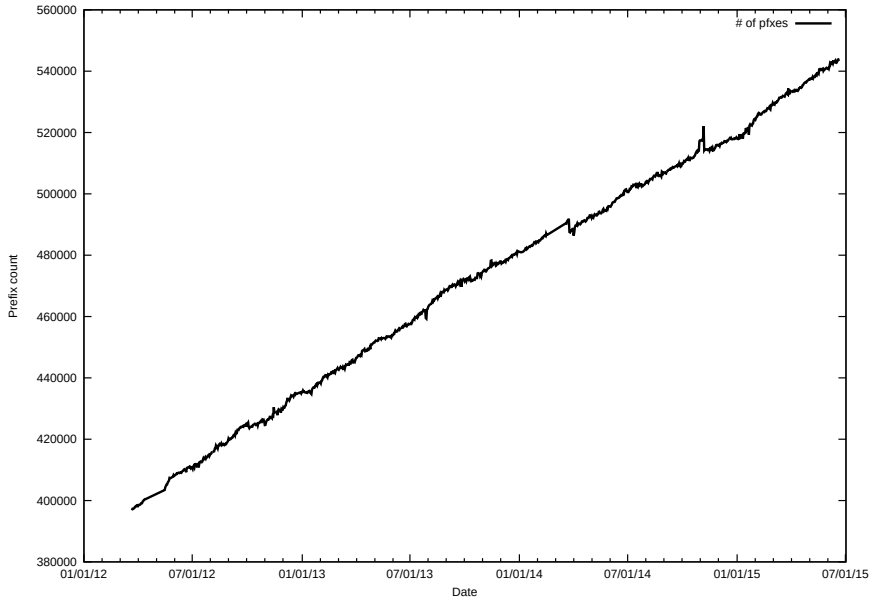


Figure 5.2: IPv4 prefixes in BGP

5.3.2 IPv4

Despite of IPv4 run-out the IPv4 BGP table has been growing steadily during the whole period, as shows the figure 5.2.

The wide-accepted answer to the question how it is possible, that IPv4 table is growing despite the fact that the IPv4 source has been depleted in major part of the world, is that three factors allows ISPs to add new prefixes to the DFZ: First possibility is to announce old and previously un-announced prefixes that some of the ISPs had allocated in past and have not used it yet. Problem is that, except the prefixes allocated in pre-RIR era, it would be breach of RIR's policies. But still it is not impossible.

Second possibility is obtaining small prefixes from RIRs' last /8. The IPv4 run-out is not yet total (in July 2015): RIRs decided few years ago, that they need to change IPv4 distribution policy, when they hit some level of remaining addresses. It means that most RIRs halted IPv4 distribution according to "justifiable need" when they reached their last */8* IPv4 prefix. New distribution policy in RIPE allows each and every LIR to obtain a single uniform-size */22* prefix from the last */8* under similar conditions regarding the justification of need, as for prefixes allocated before this policy. The intention was to support newcomers with some amount of IPv4 addresses to allow basic IPv4 presence and deployment of IPv6 transition mechanisms like NAT-PT.

Third possibility is de-aggregation of existing prefixes in the DFZ. This is actually the most frightening consequence of IPv4 depletion for most ISPs, because prefix de-aggregation on large scale can cause substantial increase of IPv4 routes in DFZ in a short time. Since BGP routers are usually hardware-assisted routing platforms, there is some hard limitation of maximum routes in FIB and overcoming the limitation implies that the router can not operate in the Internet correctly anymore and has to be replaced or carefully configured to drop certain routes, which has huge potential to cause problems.

The possible de-aggregation and injection of a large number of */22* routes

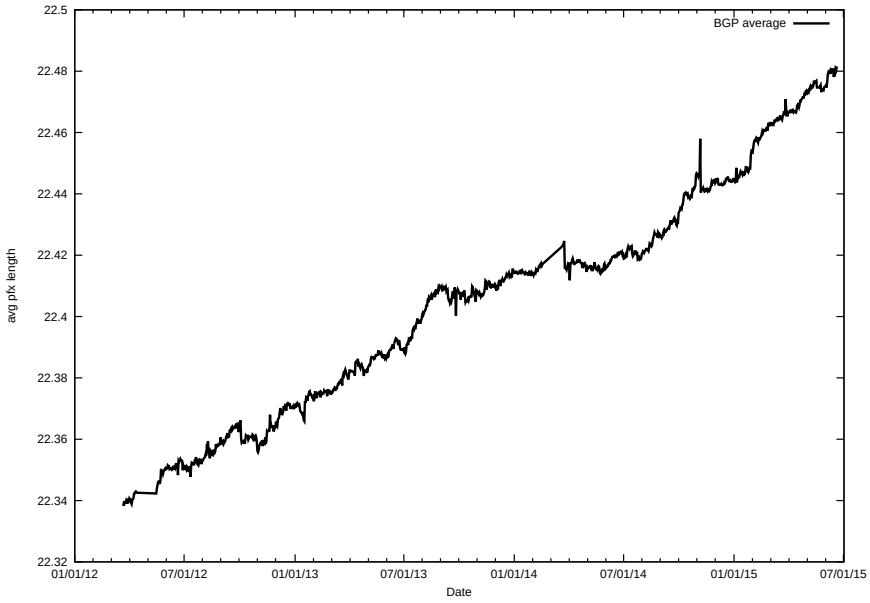


Figure 5.3: IPv4 average prefix length in BGP

at the same time should be visible. But in fact the timeline of average prefix length displayed in the figure 5.3 shows rather steady growth throughout the whole period. But the expected trend is visible in longer time frame in BGP analysis reports created by Geoff Houston from APNIC, that are published on website [27].

Another interesting point is the possible increase of */22* prefixes in routing table due to policy for handling the last */8*. Problem is that the RIPE NCC started to allocate IPv4 address from the last */8* on September 14, 2012 which means that my data are not old enough to capture the change in trend. The figure 5.4 shows rather steady growth and once again I have to refer to BGP analysis reports by Geoff Houston [27].

But it fact the IPv4 depletion consequences are visible in plots of IPv4 prefix counts for lower prefix lengths, which translates to greater portion of IPv4 address space. For instance share of one of the most common IPv4 prefixes - */20* is still increasing but it decelerates, as the figure 5.5 shows.

And even shorter prefixes started dropping. The most visible case is */16* in the figure 5.6.

5.3.3 IPv6

While IPv6 is definitely more interesting for the rest of the research effort, the IPv6 growth does not show anything exceptional or surprising. The growth in the captured period is steady and the shape in the figure 5.7 looks like it is linear in time.

The growth is pretty much concentrated in the default allocation units, which are */32* for PA allocations and anything between */32* and */48* for PI allocations in most RIRs. The figures for */32* - 5.8 and */48* - 5.9 shows the growth throughout the captured period. It is also interesting that because IPv6 allocation policies allow allocation of even shorter prefixes than */32*, there are prefixes as short as */20*, which is visible in the figure 5.10.

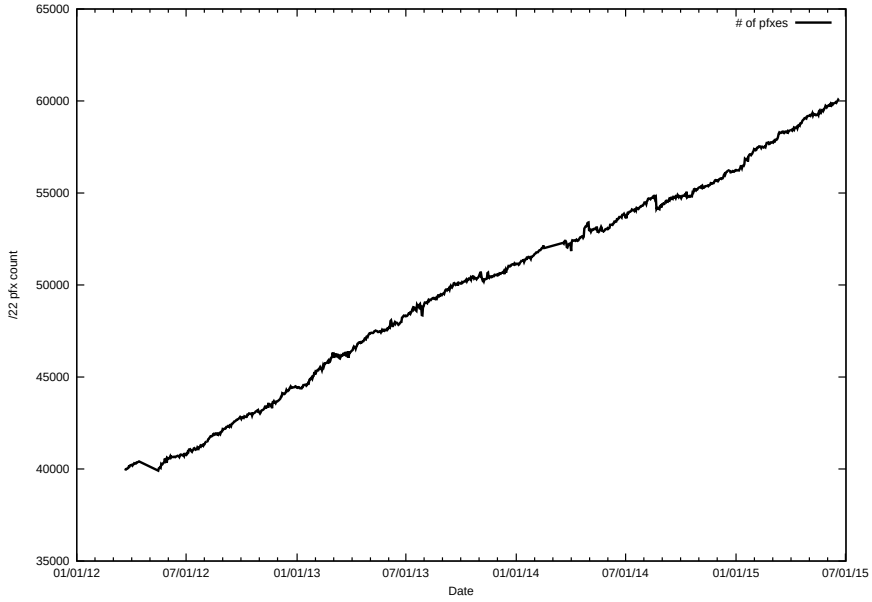


Figure 5.4: Number of IPv4 /22 prefixes in BGP

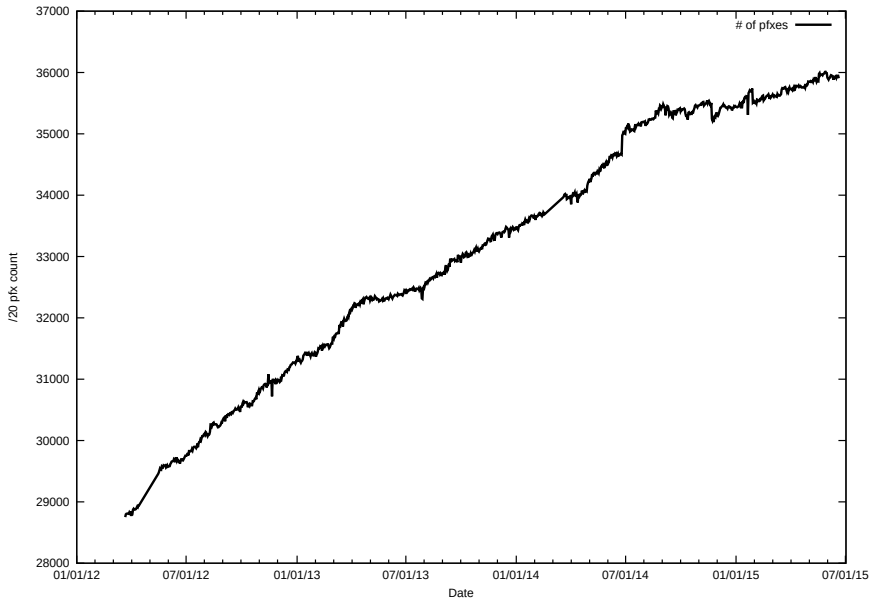


Figure 5.5: Number of IPv4 /20 prefixes in BGP

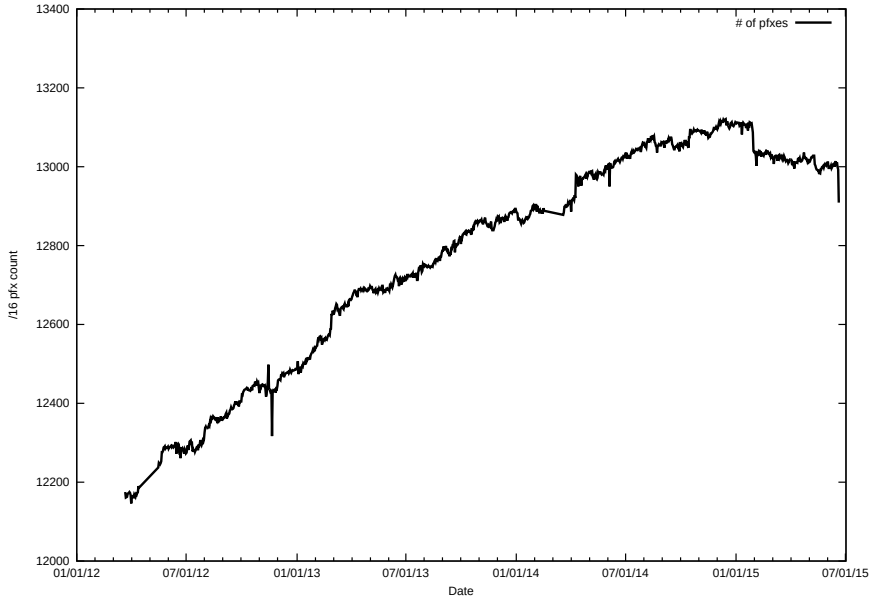


Figure 5.6: Number of IPv4 /16 prefixes in BGP

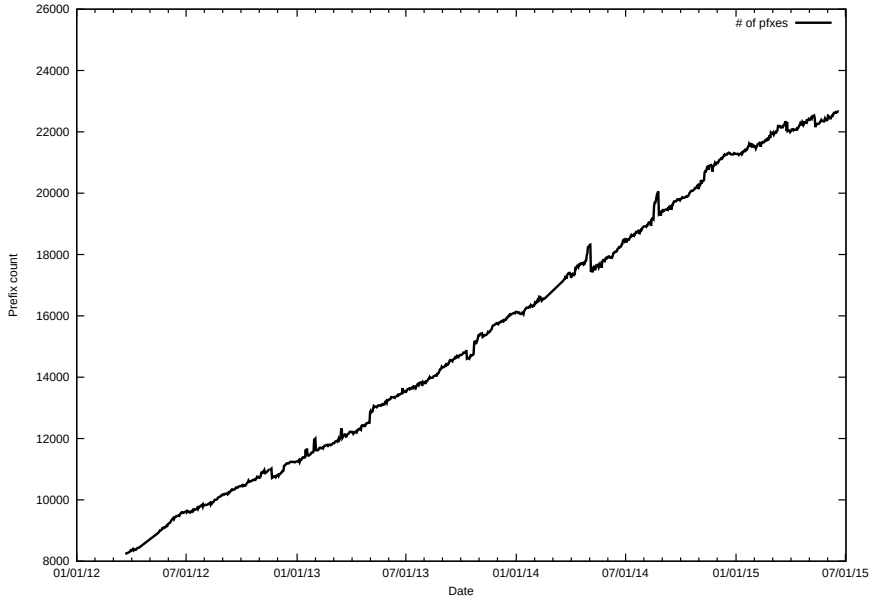


Figure 5.7: IPv6 prefixes in BGP

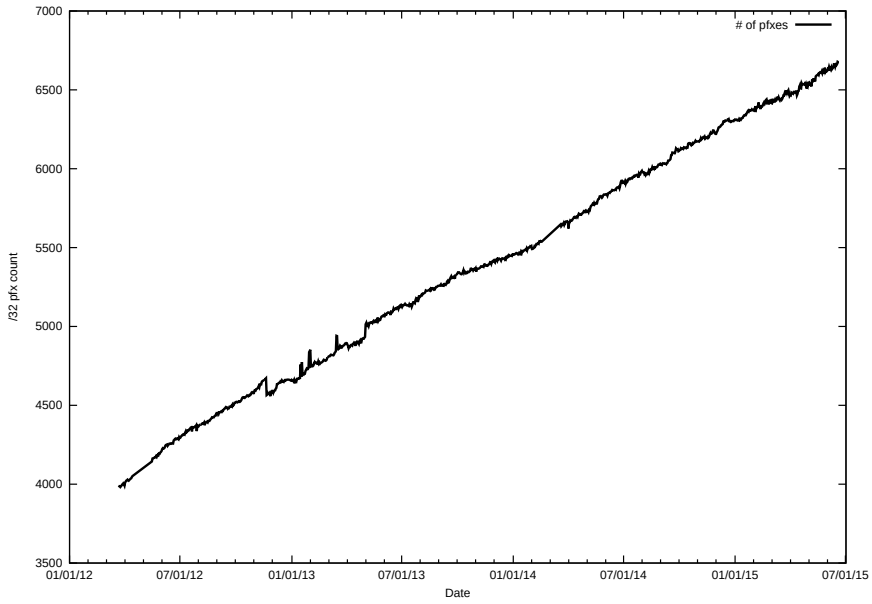


Figure 5.8: Number of IPv6 /32 prefixes in BGP

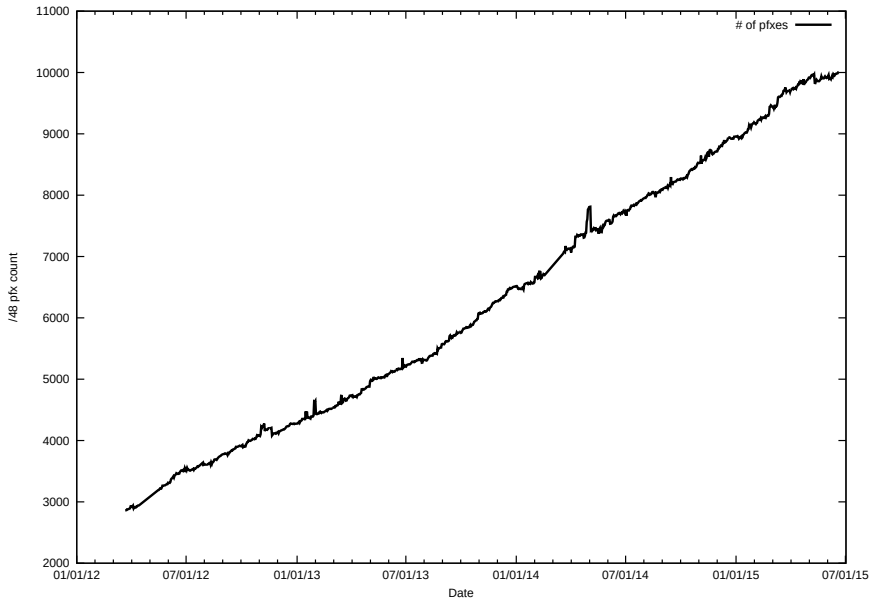


Figure 5.9: Number of IPv6 /48 prefixes in BGP

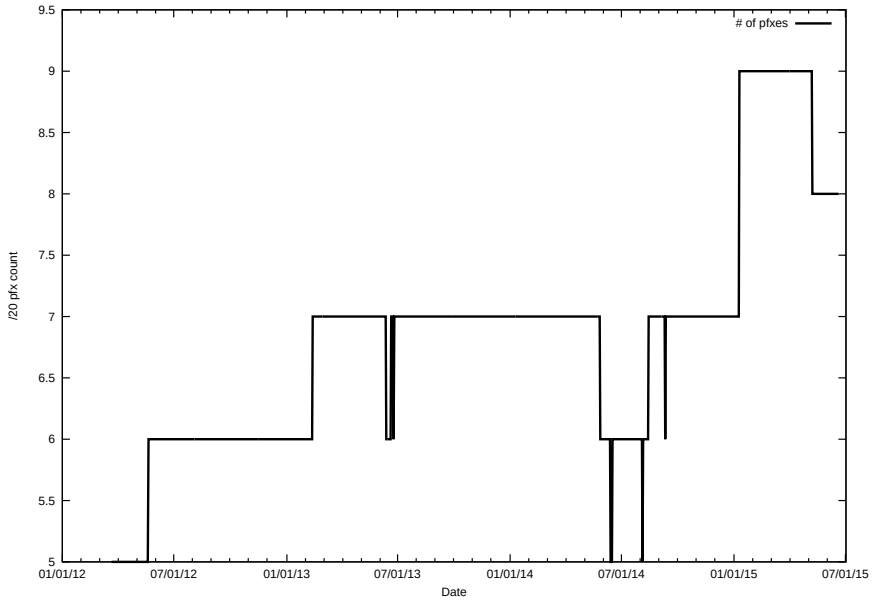


Figure 5.10: Number of IPv6 /20 prefixes in BGP

The average prefix length in IPv6 was also growing for most of the captured period but it seems that it is slowing down and there is a hope that it might be turning for good to decreasing trend in future as the figure 5.11 shows.

5.3.4 Changes near the observation point

The captured period covers not only global changes in scale of entire RIR service regions and in scale of the whole Internet but the input data are also affected by local changes. The observation point for data collection was one ASBR in **AS29134**. The primary purpose of this router is providing connectivity for an important Czech ISP that focuses on server hosting business. Advantage of using this observation point is that we have first-hand data from the real ISP, which is involved in daily operation of own backbone network and an autonomous system on full scale. But at the same time the whole network is fairly simple, because there are only two BGP routers and certain amount of IGP routers connected to them, so the design is literally a textbook case.

On the other hand disadvantage of having this particular observation point, at this place and at this particular time is that the European and Czech ISP business underwent certain transformation and restructuralization during the captured period. Analysis of what happened in the European ISP business is clearly beyond the scope of this thesis, but the consequences for the **AS29134** were several changes of upstream providers and experiments with extra-low-cost services, provided by certain networks that lately started operating in the Czech Republic. The consequences were rapid changes in BGP feed quality that are visible the figure 5.12. To interpret the figure I have to note that the simplest idea, that longer paths need to be worse than shorter ones does not hold. Nonetheless fluctuations and rapid path changes in BGP translates to lower quality after all. In fact quality of internet connection means throughput, lack of blind sports (unreachable or badly reachable places) and consistency. The rapid changes in path length might contribute to breach of consistency.

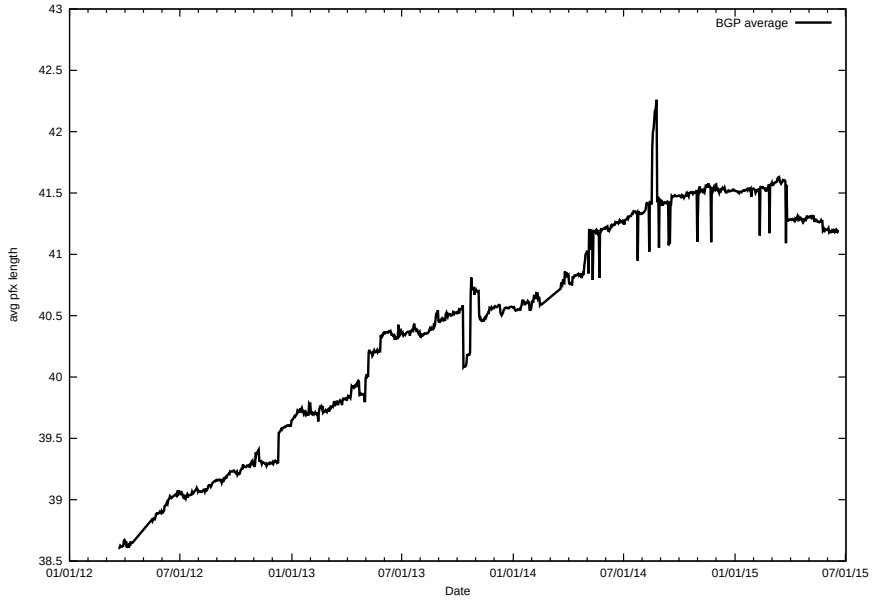


Figure 5.11: IPv6 average prefix length in BGP

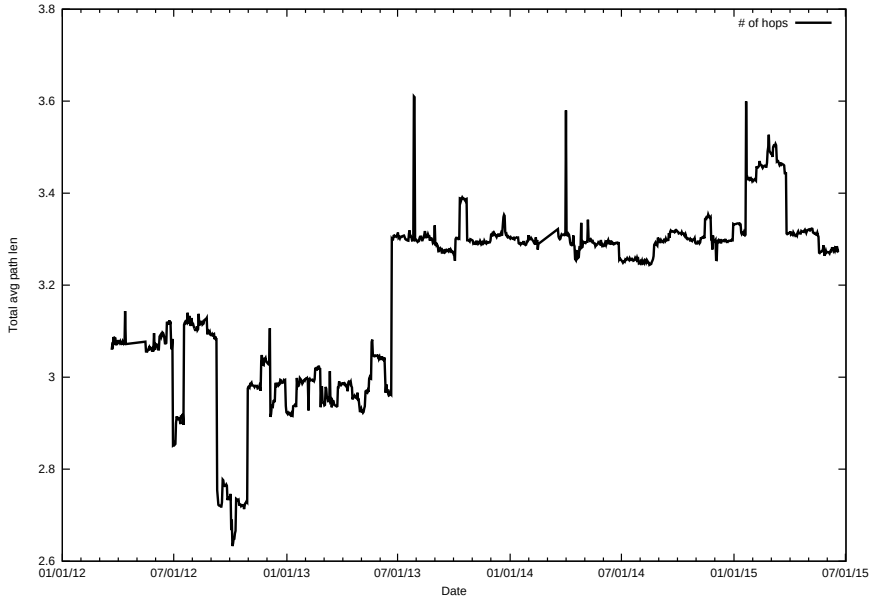


Figure 5.12: IPv4 BGP average path length

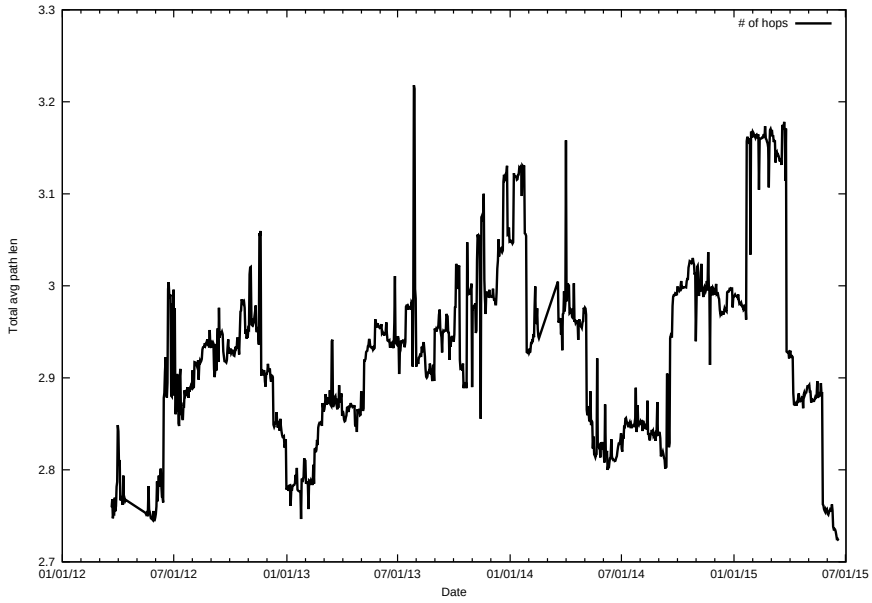


Figure 5.13: IPv6 BGP average path length

Even more BGP feed fluctuations can be seen in the figure 5.13 for IPv6.

The biggest and most visible changes in both protocols that happened in the mid-2013 were the consequence of a migration to another upstream provider. At that time the **AS29134** terminated an upstream connection to **AS174** (Cogent Com., Inc.) and started using **AS6939** (Hurricane Electric). Two most distinctive spikes in IPv4 in October 2013 and in April 2014 are two major Internet outages that happened in remote networks.

Apart from discussing the spikes and peaks in the charts and correlating them with actual changes in the **AS29134** connectivity it is interesting to look at plots of the path length with respect to prefix length and time for IPv4 in the figure 5.14.

Looking further into the figure it seems like there are two lines forming from left to right in the chart and the lines inverts each other. One at */16* mark that shows decrease of the average path length for this particular prefix length and other line shows increase at *24* mark. It is definitely worth examining detailed figures for */16* - 5.15 and for */24* - 5.16. Problem is that de-aggregations and propagation of de-aggregated paths in BGP are erratic process, so even though the charts sums up to quite smooth lines, the plots of path length in time for particular prefix length look strange.

The 3D plot reveals the rough profile of the IPv4 path length plotted against prefix length, but it is imprecise and hard to see. The actual daily plot for the last day I am taking into account for this thesis - June 21, 2015: figure 5.17 shows the profile in more detail. It is in fact only Y - Z projection of the rightmost X point in the 3D plot.

The IPv4 path length profile have actually changed substantially several times because of upstream changes, or because new connections have been created either locally or remotely, but near enough to the observation point, so it affected the local view of the Internet. For instance we can see completely different profile in the chart for the first day of the observations - March 22, 2012 in the figure 5.18.

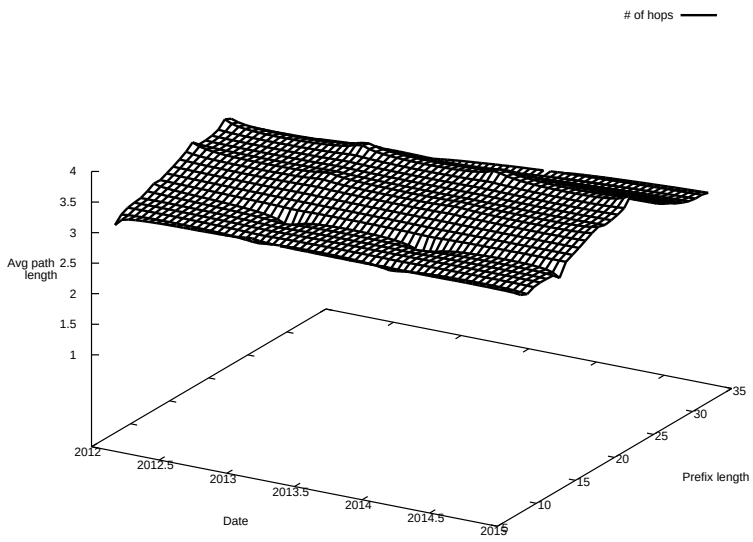


Figure 5.14: IPv4 BGP path length per prefix length

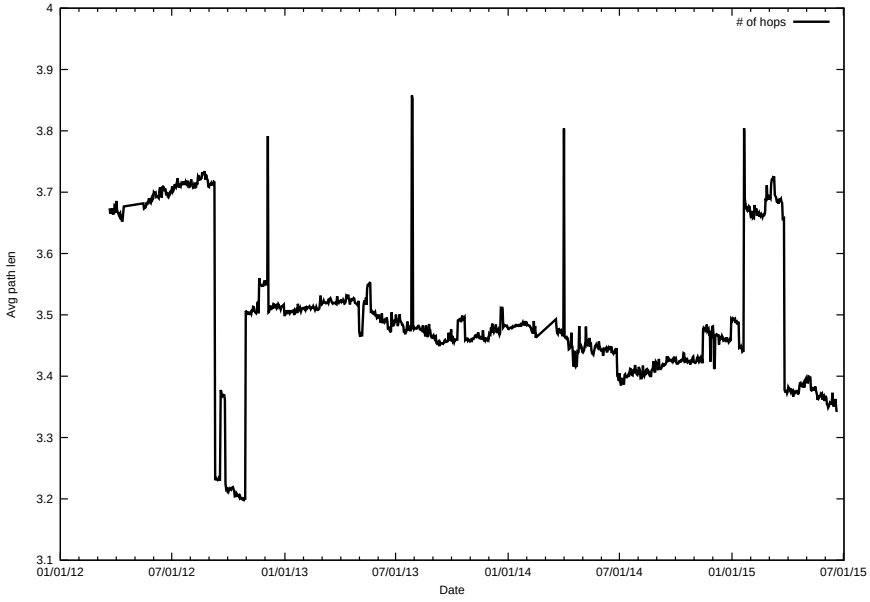


Figure 5.15: IPv4 BGP path length timeline for /16

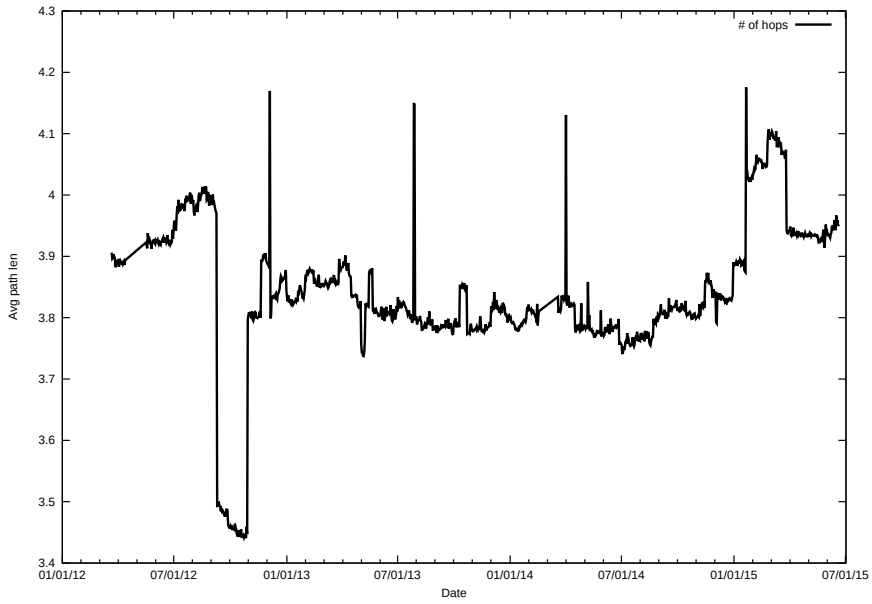


Figure 5.16: IPv4 BGP path length timeline for /24

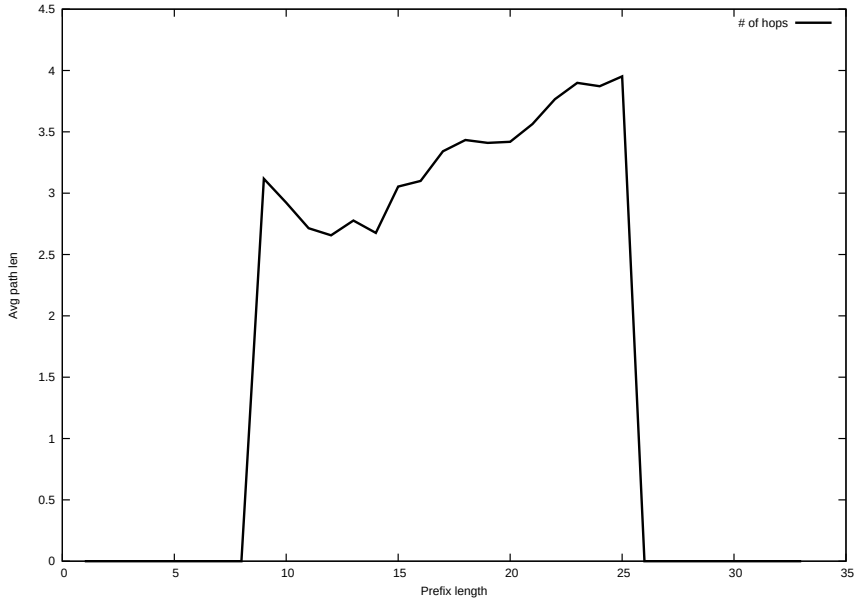


Figure 5.17: 2015-06-21 - IPv4 BGP path length per prefix length

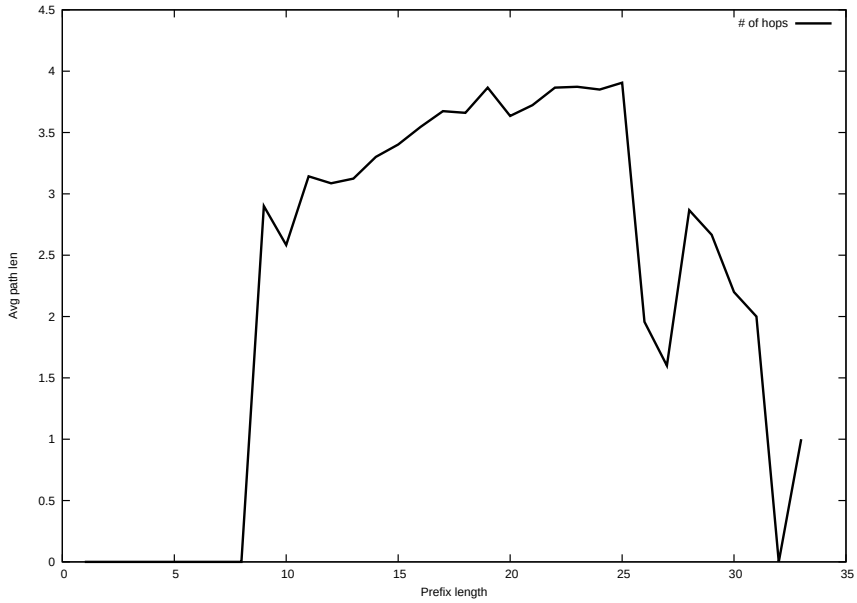


Figure 5.18: 2012-03-22 - IPv4 BGP path length per prefix length

These charts shows the local quality of the IPv4 connection to certain level, but interpretation or reasoning about the captured data heavily depend on context and side-channel information. I present the figures in this chapter, because it is important to get some picture of the scale and speed of changes in the BGP over the captured period before I present the main results.

Quality of IPv6 connectivity depends to certain degree on IPv4 service provider because the **AS29134** has a policy to operate always in fully dual-stacked mode. The same requirement is enforced in all contractual relationships with upstream providers or any other service providers. But still, the IPv4 is a bit more important so far from both technical and business point of view. The figure 5.19 shows the basic BGP path lengths plotted in time and by the prefix length.

Basically the major changes caused by migrations to different service providers are almost the same as in case of IPv4. It is worth looking into detail for IPv6 path length timelines of the most common prefix lengths. The figure 5.20 shows the path length timeline for */28* prefixes, which are a bit shorter than the preferred default. It is actually quite easy to justify need for such a large IPv6 prefix, especially for large ISPs that have plans for deployment of *6rd* (IPv6 Rapid Deployment) transition mechanism, which tends to consume a large amount of IPv6 addresses.

The figure 5.21 shows the */32* prefixes that are the preferred default allocation for PA IPv6 space.

Even though IPv6 is expected to provide better opportunity to aggregate addresses, it seems that a lot of networks announce the longest practically usable prefixes */48*. The figure 5.22 shows the path length timeline for these prefixes and we know from the figure 5.9, which shows number of */48* prefixes in time, that this group forms a considerable portion of IPv6 DFZ and it is growing quickly. It is actually not surprising because in IPv4 the */24*, the longest and therefore smallest prefixes that are practically usable, form the most ample group of prefixes in DFZ. In facts more than half of all IPv4 prefixes are the */24*s. And it seems

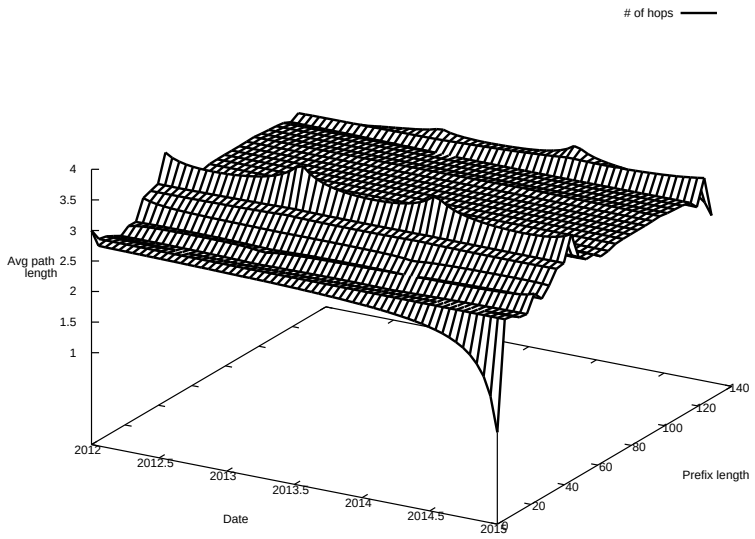


Figure 5.19: IPv6 BGP path length per prefix length

that this bad habit of using longest possible prefixes transitioned to IPv6 as well.

Just to clarify: The practical usability limit is a matter of best practice, unwritten rule and common belief in the Internet community, that there are and should be filters on certain boundaries. The boundary happen to be */24* for IPv4 and */48* for IPv6, though there are never-ending discussions about those numbers. In the end of the day each AS operator is free to set his own rules and therefore his own limits. Effects of these limits are actually quite interesting and since the **AS29134** does not impose any limits on incoming prefixes there are occasionally some shorter prefixes visible in corresponding timelines. Please refer to the timeline figures on the website `http://aule.elfove.cz/~brill/bgpcrunch` or in the appendix for more data about this topic.

5.3.5 RIR service region differences

Even though this thesis is focused on RIPE NCC service region that covers Europe, the Middle East and parts of Central Asia, the Internet is global network and we get all the routes mixed in the BGP. The figure 5.23 shows what portion of the routes in DFZ each RIR contributed to the IPv4 Internet over the captured period. The figure 5.24 shows the same for the IPv6 AFI. I find interesting that RIPE NCC service region clearly surpassed the ARIN region in IPv6 deployment and it is also fastest growing and consistently increasing region. On contrary the IPv4 chart shows huge gap among ARIN region and any other regions.

The table 5.3 shows the share of IPv4 DFZ by the RIR regions and the average prefix length in each region. The average prefix length indicates the level of de-aggregation within the region.

The same table for IPv6 - 5.4 shows much lower numbers and much less favorable picture of the de-aggregation. The level of de-aggregation in IPv6 DFZ is a huge disappointment, because in IPv6 the common allocation for a LIR is */32*, but the numbers are attacking rather */48*, which is the suggested minimum

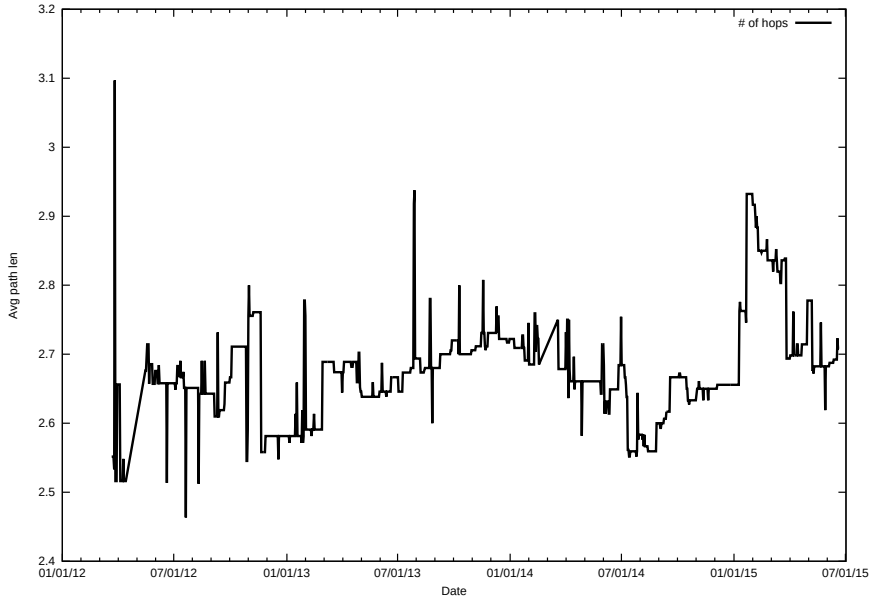


Figure 5.20: IPv6 BGP path length timeline for /28

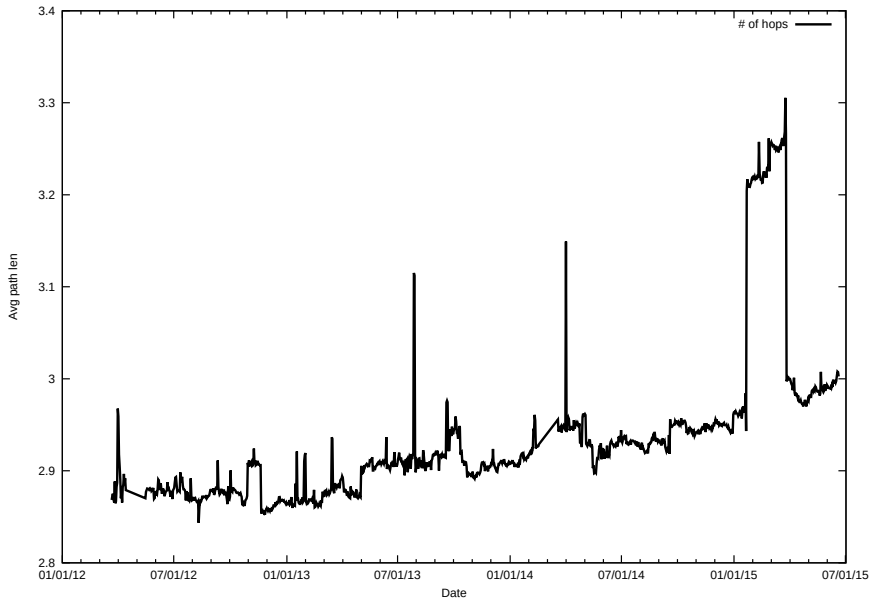


Figure 5.21: IPv6 BGP path length timeline for /32

RIR	No. of prefixes	Average prefix length
LACNIC	65045	22.33
APNIC	133151	22.33
ARIN	189472	22.33
RIPE NCC	136158	22.33
AFRINIC	12273	22.33

Table 5.3: IPv4 RIR regions size - 2015-06-21

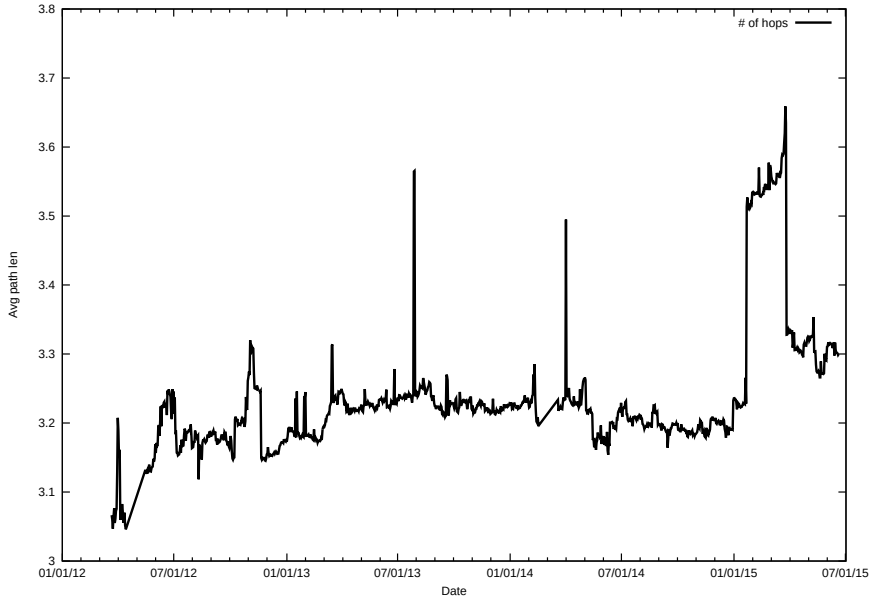


Figure 5.22: IPv6 BGP path length timeline for /48

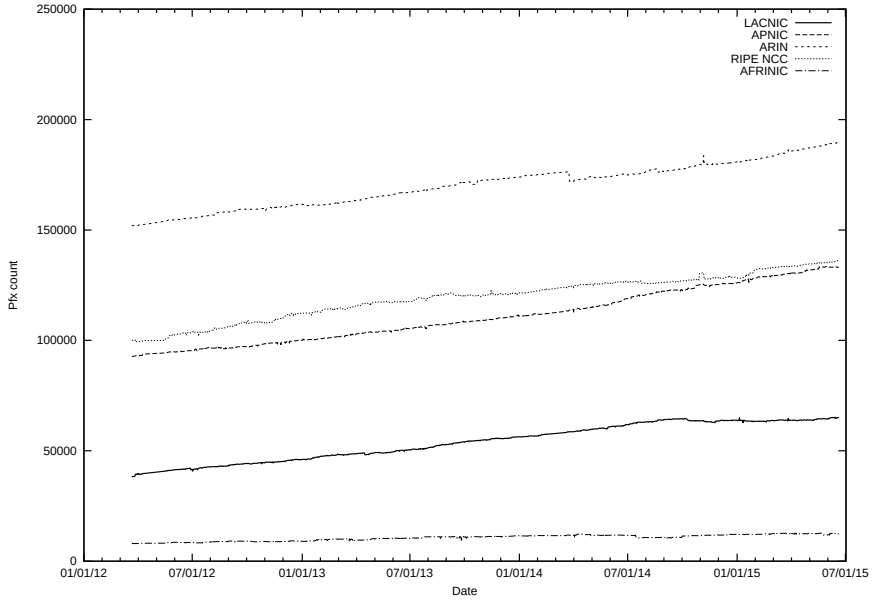


Figure 5.23: IPv4 RIR DFZ share

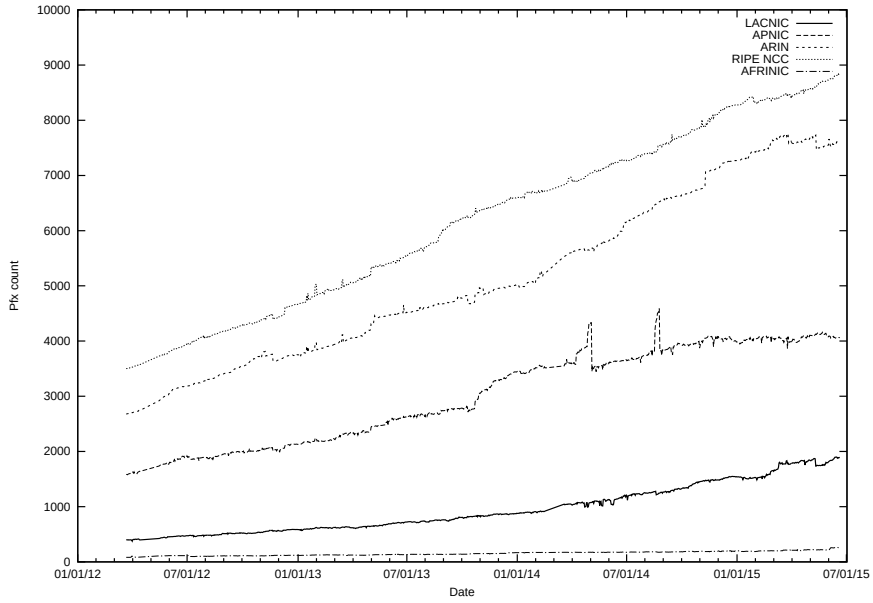


Figure 5.24: IPv6 RIR DFZ share

RIR	No. of prefixes	Average prefix length
LACNIC	1890	38.19
APNIC	4060	43.61
ARIN	7629	43.25
RIPE NCC	8822	38.99
AFRINIC	258	40.05

Table 5.4: IPv6 RIR regions size - 2015-06-21

assignment for the end network. Still the end networks are supposed to get aggregated at the ISP level, but it is obviously not happening.

The most discomforting figure of this section is the RIR average path length timeline for IPv4 - 5.25. Although I know that service region of AFRINIC comes through troubled times and I can imagine the reasons for de-aggregations and and changes in subnet announcements, this exceeded any fears and worst expectations. The chart proves that Internet is extremely dynamic even in its technical base and it seems that people are trying hard to squeeze more performance, money or advantage form their resources by constant and occasionally pretty violent changes.

The same chart for IPv6 in the figure 5.26 shows a bit more consistent but still shocking numbers. In case of IPv6 I find alarming mainly the scale of changes within small periods of time. In case of APNIC it seems like there is amplitude over 2 in the last year and it oscillates within days or weeks. Especially in IPv6 it is highly unlikely for one single ISP to affect the whole region because even the major ISPs are supposed to have a single PA allocation. They might de-aggregate but it is hard to imagine one single ISP to announce one short prefix one day and thousands of longer prefixes next day. More research into this matter will be needed in future.

This section intentionally presents only few selected charts. I tried to rephrase

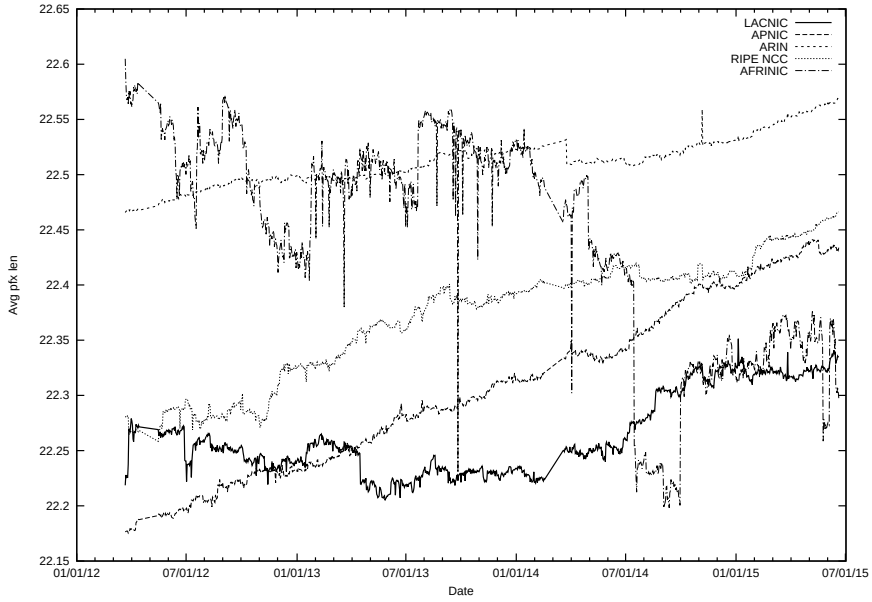


Figure 5.25: IPv4 RIR average prefix length

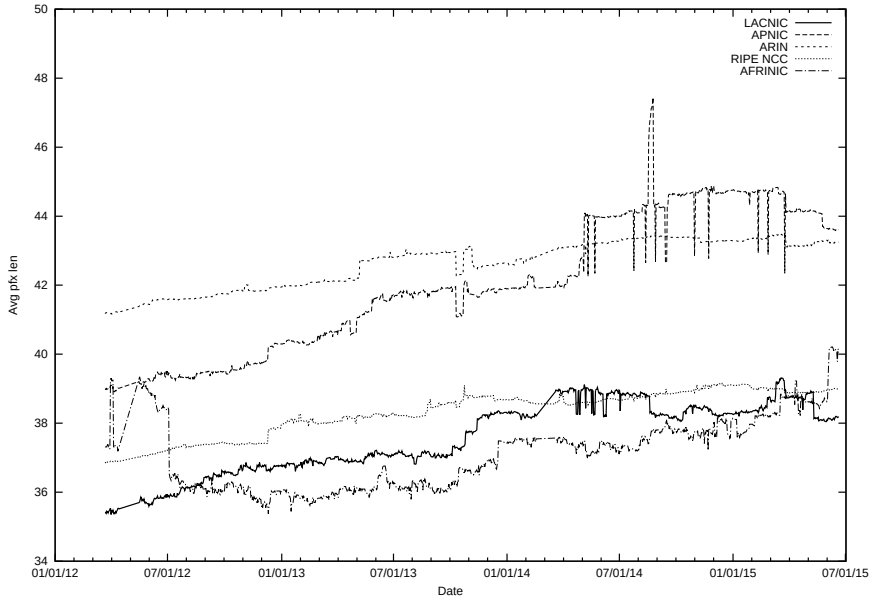


Figure 5.26: IPv6 RIR average prefix length

Prefix	Announcing AS	*route* object	Result
RIPE	RIPE	missing	fail (missing *route* obj)
RIPE	RIPE	prefix match, no *origin* match	fail (AS not match)
RIPE	RIPE	prefix match, *origin* match	OK
RIPE	no AS (local prefix)	N/A	N/A (origin missing)
RIPE	anything, but the path is summarized	N/A	N/A (no-search aggregate)
RIPE	non-RIPE	missing	fail (missing *route* obj)
RIPE	non-RIPE	prefix match, no *origin* match	fail (AS not match)
RIPE	non-RIPE	prefix match, *origin* match	OK
non-RIPE/legacy	anything	anything	N/A (non-RIPE)

Table 5.5: Origin validation states

textual output and interpret the results in a readable form. There are over 1800 charts that contain results, relevant to this section in the appendix in the electronic version of the thesis. The same charts with even more data points and the full text outputs are on the above mentioned website.

5.4 Origin validation

The origin of BGP routes is quite easy to validate. The idea, roughly described in the previous chapter, is to find *route* or *route6* objects for each path vector in DFZ BGP table and check the *origin* attribute. Apart from the object type in the routing database, the checking process is the same for both IPv4 and IPv6.

It is possible to have more *route* or *route6* objects for one prefix in one routing database and the meaning is logical OR. The path vector is valid when it matches at least one *route* or *route6* object, according to the proper AFI.

Since we are limited to European prefixes and to matching in RIPE DB, we do not have to resolve the fundamental problem of multiple Internet routing registries - deciding what data are authoritative. In case of *route* or *route6* objects the problem seems to be simple enough because we can keep the rule of a single match sufficiency. But the problem is that more routing registries might contain duplicates (which is not a problem) or obviously outdated data, which would be an issue. With only one database the possible situations are described in the table 5.5.

This resolution procedure has one hidden limitation with respect to conflicting path vectors. It is possible for a prefix to be present in BGP table of the observation point multiple times, which effectively means that there are more path

vectors with the same prefix component and different remaining components. It is perfectly normal and desirable, to have alternate paths to one destination and BGP selects one of these path vectors as the best, installs the corresponding route into FIB and propagates the best path vector further. It is even possible to have one prefix with different origins, though it is highly unusual and each case raises severe suspicion, because there is practically no valid reason to do that intentionally. On contrary, it might be a symptom of ongoing prefix hijacking attack.

Since we have so many different paths to each prefix in the Internet and I wanted to save CPU time spent on matching them to the origins, we actually examine only the best paths. It means we have to examine only approximately 500 000 path vectors for each day instead of over two millions. Obvious downside is that we would not spot possible prefix hijacking incidents. Therefore I searched through data specifically for this incidents and I found none. But I have to say that is is not likely to spot this kind attacks, because they are rare and it would have to happen near (BGP-wise) from the observation point. And we would need to have diverse connectivity to the victim AS as well as to the attacker to be able to see the incident.

Decision rules for the origin validation procedure can be a subject for long discussion, because the rules derive not only from standards and RFC documents, but also from current best practices and IP allocation policies of RIRs, that might be a subject for interpretation. The main issue is whether we should ignore path vectors that are parts of IP space allocated through RIPE NCC, but the announcing AS is not a part of RIPE NCC blocks. I examined current RIPE NCC allocation policies in order to research into this question and I think that it is proper to count failed path vectors that falls within RIPE IP space and foreign announcing ASes into errors, because policies forbids using majority of resources allocated by RIPE NCC outside of the RIPE NCC region and therefore it seems that ASes of different provenience should operate in RIPE NCC service region and submit to the same rules. Moreover, it is perfectly possible to register foreign AS into RIPE DB and use it as an *origin* in *route* or *route6* objects.

The figure 5.27 shows the BGP origin validation timeline for IPv4. The IPv6 counterpart is in the figure 5.28.

The complete results in text form are too large to be presented here or attached to the thesis in appendix, but the results are accessible on the above mentioned web site in the sections called **Daily IPv4 BGP route matching report (text)** and **Daily IPv6 BGP route matching report (text)**. The most interesting output is in section **Route violations timeline** on the web site and the same results are attached on the disc that comes with the printed version of this thesis. The timeline contains one file for each prefix that failed the BGP origin validation test at least once in the whole captured period. It shows all the changes both in BGP and in IRR that concerns the prefix and its relation with the IRR contents. The table 5.6 shows totals for IPv4 and the table 5.7 shows results for IPv6 from June 21, 2015, the last day of the captured period.

The results proved that even though the RIPE community encourages LIRs, ISPs and network operators within the RIPE NCC service region to keep extensive records and use *route* and *route6* objects to express their most basic BGP routing behavior, almost one third of prefixes in the RIPE NCC service region are invalid.

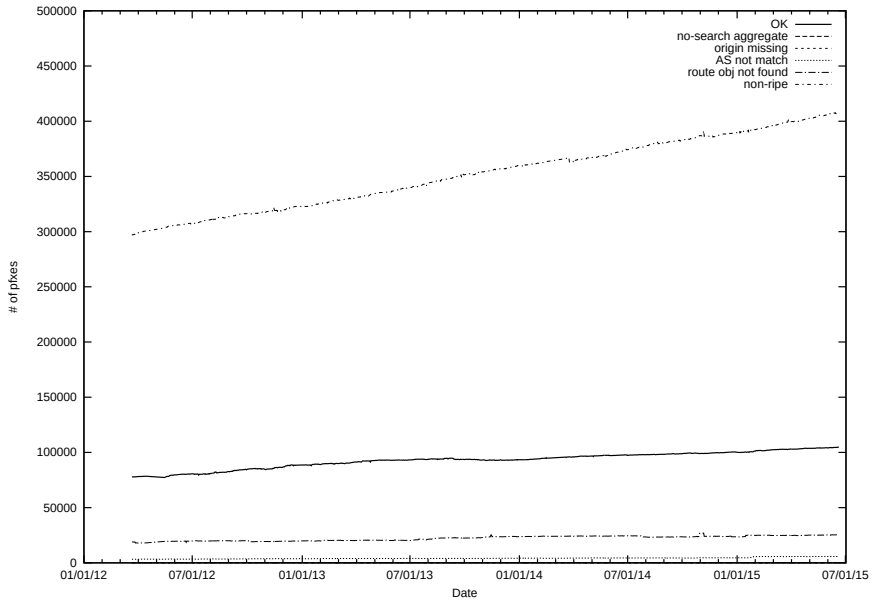


Figure 5.27: IPv4 BGP origin validation timeline

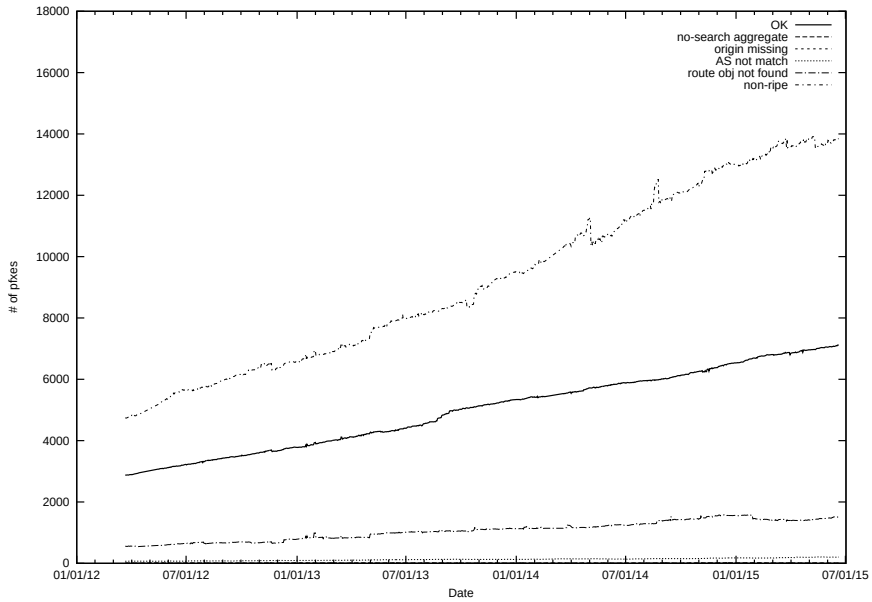


Figure 5.28: IPv6 BGP origin validation timeline

Status	No. of path vectors
OK	104778
no-search aggregate	164
origin missing	3
AS not match	5778
route obj not found	25555
non-ripe	407049

Table 5.6: IPv4 Origin validation states

It is hard to say whether is it worse not to have any *route* or *route6* object at all (which is the *route obj not found* status) or have some object with different *origin*. Actually it depends on the reason, why there is the discrepancy among BGP path vector and available objects in Internet routing databases. It might be temporary state, when the prefix is being migrated from one AS to another. Though in this case the correct procedure is to create new object with the new *origin*, then migrate the prefix in BGP and only after that remove the old *route* or *route6* object, which would avoid failing the origin validation tests.

Another possibility is that there is the corresponding *route* or *route6* object completely missing during the entire history of the prefix. This is not rare situation even though prefix origins are being checked, using very similar methods in IXPs, especially when the prefixes traverse route servers. Prefixes that fails origin validation are usually dropped from forwarding through route servers. It effectively means that the failed prefixes, which is quite substantial number, as we can see, are more likely to take longer and more expensive paths and are generally limited in number of alternate paths in the entire Internet, even in remote places from the point of view of the prefix originator.

It is also possible that some of the prefixes that fails validation are actually hijacked, which simply means that the prefix is being announced to DFZ by another user than the user that is entitled to do so. The most common type of hijacking is that somebody picks an unused *legacy* prefix that has not been reclaimed or even some unused prefix that is not announced for any other reason and simply starts announcing the prefix into DFZ. If the malicious announcement manages to pass filters of upstreams and spread to the Internet, it can be used as ordinary IP space for initiating TCP sessions or running public servers. Hijacked prefixes are usually source of spam or sophisticated DoS attacks that need TCP connections and their lifespan is usually limited to hours or days at most.

It is also possible to hijack existing and utilized prefix. It usually causes denial of service for the legal user of the IP space. These hijacking attacks, either malicious or unintentional, should be prevented by the upstream and peering filters. Even though current practice uses many different side channels to distribute the information about the announcements, the primary source for the filters should be IRR, more precisely *route* or *route6* objects in IRR. But it does not work obviously, if it is possible to have over 30 000 prefixes in RIPE NCC service region that do not have valid *route* or *route6* and the prefixes are still visible and work well on the Internet. It, in fact, means that great number of prefixes work well enough without the proper origin records, so their originators feel no need to take care of *route* and *route6* objects, even though it would be extremely simple to rectify them.

On the other hand the timeline shows that the number of failures is steady and actually slightly declining in IPv4. In IPv6 the situation is a bit worse: Timeline shows that the number of failures is increasing, but it is much slower than the growth of the correct prefixes. The table 5.7 shows the newest numbers for the IPv6. Still, in IPv6 keeping the origin records should be extremely straightforward and any discrepancy is simple to fix, especially because even the biggest networks usually need no more than one single IPv6 prefix.

Status	No. of path vectors
OK	7113
no-search aggregate	14
origin missing	2
AS not match	204
route obj not found	1498
non-ripe	13837

Table 5.7: IPv6 Origin validation states

5.5 Path validation

Path validation is the most complex and the novel part of this work. Unlike the basic data analysis and *origin* matching that has been done before by another researchers, I am not aware of any work that attempted the path validation.

The basic idea is to take each path vector in DFZ and traverse the entire path from the originator up to the observation point. In the beginning of the path we have to check the *origin* and decide what to do with paths that fails the *origin* validation test, described in the previous section. Basically the origin validation error is yet another failure mode for the path verification. For each AS, apart from the first and the last one in the AS path, we can attempt to lookup and verify the routing policy conformance of each prefix that transits the AS. The basic source of the routing policy specification for each AS is the corresponding *aut-num* object. The object should define *import* and *default* filters that contains selector for the neighboring AS, which announces the path vector and the AS in question can import it if the filter matches. The opposite direction works in the same manner: There are *export* lines. Each contains a selector and a filter that should accept the path vectors that the AS in question announces to the neighbors, picked by the selector.

The selector might be simply an AS number of the neighbor, which is actually the most common case. But the selector can contain list of selector terms, that can be ASNs, lists, or references to *-set* objects that need to be recursively resolved.

In order to evaluate the path vector imported into an AS we have to search through all selectors and eventually the corresponding filters in the *aut-num* object's *import* lines, until we find a selector and corresponding filter that accepts the path vector, or until we reach the end of the *aut-num* object. The exhaustive search is necessary, because finding a selector with a filter that rejects the prefix does not mean that there can not be another selector that matches the neighboring AS as well and the corresponding filter would accept the path vector in question. Therefore we need to potentially examine each *import* and even *default* lines in the *aut-num* object and verify filter portion of the lines that match in the selectors.

Exactly same procedure works for *export* filters. Though the checking of the directions is asymmetric in certain aspect: We can spot only validation errors in path vectors that are visible at our observation point in BGP and violate routing policy at some point. It means that routing policy effectively does not allow propagation of these particular path vectors, but they are propagated nonetheless.

AS	*aut-num*	filter	result
RIPE	present	match	OK
RIPE	present	not-match	fltr fail
RIPE	missing	N/A	fltr not found
non-RIPE	present	match	OK
non-RIPE	present	not-match	fltr fail
non-RIPE	dummy	not-match	unknown
non-RIPE	missing	N/A	unknown

Table 5.8: Path validation states

On the other hand we can not observe the opposite situation, when there is a prefix propagation suppressed in BGP in spite of that the prefix should be allowed to propagate, according to the routing policy.

In fact, this could be theoretically possible to do. In theory, we could build a graph of all relations among all autonomous systems in the Internet according to routing policies, compute a best path matrix in this enormous graph and then compare actual paths in BGP with the theoretically obtained best paths. There are obvious technical problems with that thought experiment and it is far beyond the scope of this work to attempt something alike. I am mentioning that just to discuss shortcomings of our validation approach.

Another drawback is that I was able to validate only the best paths in the BGP table of the observation point, given the available resources and finite time. In fact it would be easy to merely flip a switch and instantly evaluate three to five times more path vectors for each prefix, by taking into account also the path vectors that took different paths, than the best ones. But I would not be able to compile the results in reasonable time if the "processing" phase take 3 to 5 times more resources or time.

There are many more possible failure modes in this validation process. The table 5.8 shows possible results from validation of one hop (one AS in the AS path) in both directions - *import* as well as *export*. There is notable special case for transit autonomous systems that do not operate within RIPE NCC service region and decided to create *aut-num* object in RIPE DB for sake of registration and authorization. These objects might be also generated automatically, which leaves them with virtually no useful information. This is, strictly speaking, a routing policy failure, but I believe that it is a different kind of failure and it is more like a missing *aut-num* object in RIPE DB than a missing filter.

Basically checking the hop with corresponding *aut-num* object consists of:

1. Taking proper *import* and *default* lines if some exists for the previous AS in the AS path, and verifying the corresponding filters until the first match is found or the end of the object is reached.

2. Taking proper *export* lines (if exists) for the next AS in the AS path and verifying corresponding filters until the first match is found or the end of the object is reached.

Obviously there are special cases: The first and last autonomous systems in

import or *default*	*export*	result
match	match	OK
no-match	match	import fltr fail
match	no-match	export fltr fail
no-match	no-match	import fltr fail
N/A, originating AS	match	OK
N/A, originating AS	no-match	export fltr fail
match	N/A, last AS	OK
no-match	N/A, last AS	import fltr fail

Table 5.9: Hop validation states

the AS path. We can check only the available portion of the filter - *export* for the right-most AS and *import* for the left-most one.

The possible results of the whole process are listed in table 5.9. Please note that the result for case when both *import* and *export* filters fail is actually "import filter failure". This is because the prefix should not have been imported in the first place and therefore the export filter is actually right when it does not accept the stray prefix that already violated the *import* filter.

The reason for categorizing various failure modes is because we have so many of them at different points of AS-paths and therefore it is not enough to mark any path that contains any error as "invalid", but we have to break down the errors by their type and even by their position in AS path relative to the observation point.

5.5.1 IPv4

The figure 5.29 shows the summary of path validation results for IPv4 through the entire captured period. We can see from the chart that judging the result based on one single failed filter in the AS path is not enough. The fluctuations are heavily dependent on network paths, that the prefixes take from the originator to the observation point at the time of the BGP table snapshot capture. It changes over time and we can see that the changes correlate with the major fluctuations of the average BGP path presented in figure 5.12, that depends on already discussed local changes in BGP setup. It is obvious that the timeline is too coarse and too dependent on the observation point location and network connectivity, that we can not make any judgments on trends or global tendencies based only on that.

We can decompose the validation results based on detailed verification state, but it would not still be enough to see clearly how many errors are in the routing policies, when we compare them to the actual BGP state. The figure 5.30 shows results of another promising approach: It measures how many errors of each type occurred during the entire validation process. It means that each traversed AS that failed to validate counts once. To interpret this chart we have to keep in mind the number of IPv4 prefixes in DFZ from the figure 5.2 as well as the average path length that has been shown in the figure 5.3.

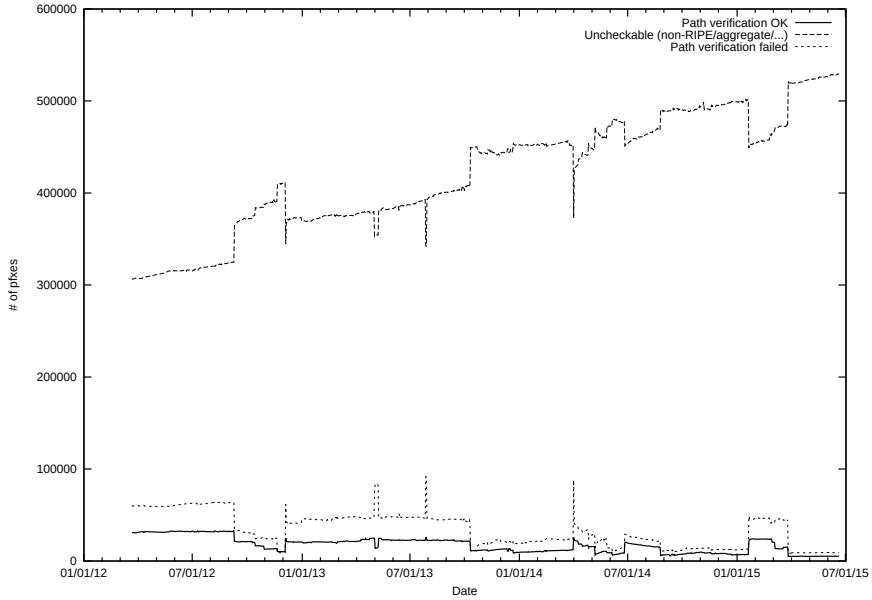


Figure 5.29: IPv4 BGP path validation timeline

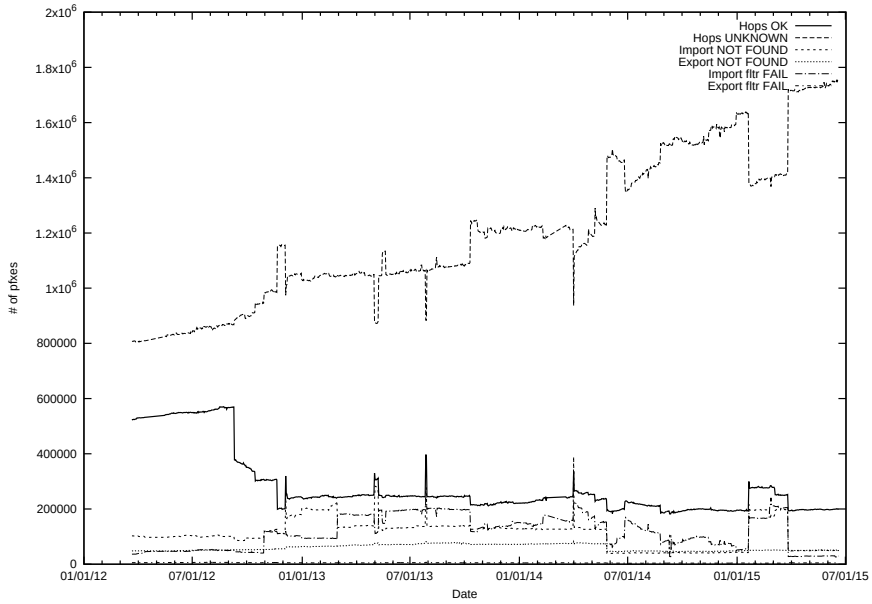


Figure 5.30: IPv4 BGP path validation details

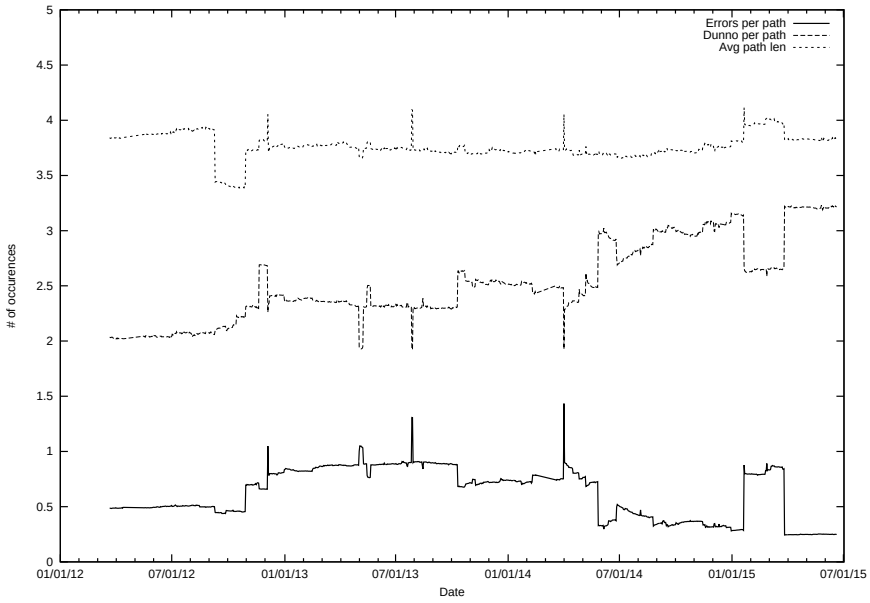


Figure 5.31: IPv4 BGP validation errors per path

It is definitely interesting to compute combined timeline from the previously mentioned charts. The figure 5.31 shows the timeline of average path length and compare it with the timeline of average number of errors per path and with timeline of average number of correctly validated prefixes per path.

The detailed results for each prefix in DFZ and for each day within the captured period are published in text form on above mentioned web site in section **Daily IPv4 BGP path matching report (text)**. The latest summary report for the last day of the captured period - June 21, 2015 follows:

```
Total prefixes: 543327
Path verification OK: 5234
Uncheckable (non-RIPE/aggregate/...): 529193
Path verification failed: 8900

Total hops observed: 2083177
Total hops valid: 199971
Total hops unknown (non-RIPE/aggregate/filter-dunno): 1748076
Total hops unknown due to filter syntax error (included in dunno): 90
Total import filter not-found: 49342
Total import filter invalid: 29628
Total export filter not-found: 50933
Total export filter invalid: 5227
```

Please note the low number of the decidable paths - both the correctly verified and the failed. The reason for that is the **AS29134** uses upstream connection to the **AS6939**, which is an international carrier that is based in the USA and it operates withing ARIN service region and therefore the *aut-num* object for the main upstream provider went out of our scope. The path that can be completely verified to the date consists only of the ASes within Europe that can be reached form **AS29134** via NIX.CZ and over direct peerings.

Still we can perform meaningful analysis of the data despite the fact we have to ignore the first hop, which is undecidable in most cases.

The number of observed prefixes and verification status with respect to the length of the path from the observation point to the originators are displayed by the following output:

```
Hop 0 : 543324 pfx traversed, 8841 ok, 6554 errors, 527929 dunnos
Hop 1 : 540934 pfx traversed, 60375 ok, 74560 errors, 405999 dunnos
Hop 2 : 450513 pfx traversed, 53877 ok, 33922 errors, 362714 dunnos
Hop 3 : 259788 pfx traversed, 30320 ok, 13476 errors, 215992 dunnos
Hop 4 : 116076 pfx traversed, 17952 ok, 3781 errors, 94343 dunnos
Hop 5 : 59055 pfx traversed, 9718 ok, 858 errors, 48479 dunnos
Hop 6 : 36046 pfx traversed, 6255 ok, 422 errors, 29369 dunnos
Hop 7 : 23248 pfx traversed, 3776 ok, 316 errors, 19156 dunnos
Hop 8 : 16240 pfx traversed, 2821 ok, 218 errors, 13201 dunnos
Hop 9 : 10421 pfx traversed, 1892 ok, 161 errors, 8368 dunnos
Hop 10 : 6765 pfx traversed, 1333 ok, 206 errors, 5226 dunnos
Hop 11 : 5281 pfx traversed, 942 ok, 181 errors, 4158 dunnos
Hop 12 : 3971 pfx traversed, 516 ok, 191 errors, 3264 dunnos
Hop 13 : 3225 pfx traversed, 405 ok, 200 errors, 2620 dunnos
Hop 14 : 2422 pfx traversed, 294 ok, 57 errors, 2071 dunnos
Hop 15 : 1945 pfx traversed, 117 ok, 27 errors, 1801 dunnos
Hop 16 : 735 pfx traversed, 73 ok, 0 errors, 662 dunnos
Hop 17 : 688 pfx traversed, 60 ok, 0 errors, 628 dunnos
Hop 18 : 596 pfx traversed, 49 ok, 0 errors, 547 dunnos
Hop 19 : 227 pfx traversed, 25 ok, 0 errors, 202 dunnos
Hop 20 : 186 pfx traversed, 24 ok, 0 errors, 162 dunnos
Hop 21 : 134 pfx traversed, 23 ok, 0 errors, 111 dunnos
Hop 22 : 116 pfx traversed, 22 ok, 0 errors, 94 dunnos
Hop 23 : 108 pfx traversed, 20 ok, 0 errors, 88 dunnos
Hop 24 : 101 pfx traversed, 20 ok, 0 errors, 81 dunnos
Hop 25 : 100 pfx traversed, 19 ok, 0 errors, 81 dunnos
Hop 26 : 97 pfx traversed, 19 ok, 0 errors, 78 dunnos
Hop 27 : 96 pfx traversed, 18 ok, 0 errors, 78 dunnos
Hop 28 : 96 pfx traversed, 18 ok, 0 errors, 78 dunnos
Hop 29 : 77 pfx traversed, 18 ok, 0 errors, 59 dunnos
Hop 30 : 49 pfx traversed, 18 ok, 0 errors, 31 dunnos
Hop 31 : 49 pfx traversed, 18 ok, 0 errors, 31 dunnos
Hop 32 : 49 pfx traversed, 18 ok, 0 errors, 31 dunnos
Hop 33 : 48 pfx traversed, 18 ok, 0 errors, 30 dunnos
Hop 34 : 25 pfx traversed, 3 ok, 0 errors, 22 dunnos
Hop 35 : 25 pfx traversed, 3 ok, 0 errors, 22 dunnos
Hop 36 : 25 pfx traversed, 3 ok, 0 errors, 22 dunnos
Hop 37 : 25 pfx traversed, 3 ok, 0 errors, 22 dunnos
Hop 38 : 25 pfx traversed, 3 ok, 0 errors, 22 dunnos
Hop 39 : 25 pfx traversed, 3 ok, 0 errors, 22 dunnos
Hop 40 : 24 pfx traversed, 3 ok, 0 errors, 21 dunnos
Hop 41 : 24 pfx traversed, 3 ok, 0 errors, 21 dunnos
Hop 42 : 23 pfx traversed, 3 ok, 0 errors, 20 dunnos
```

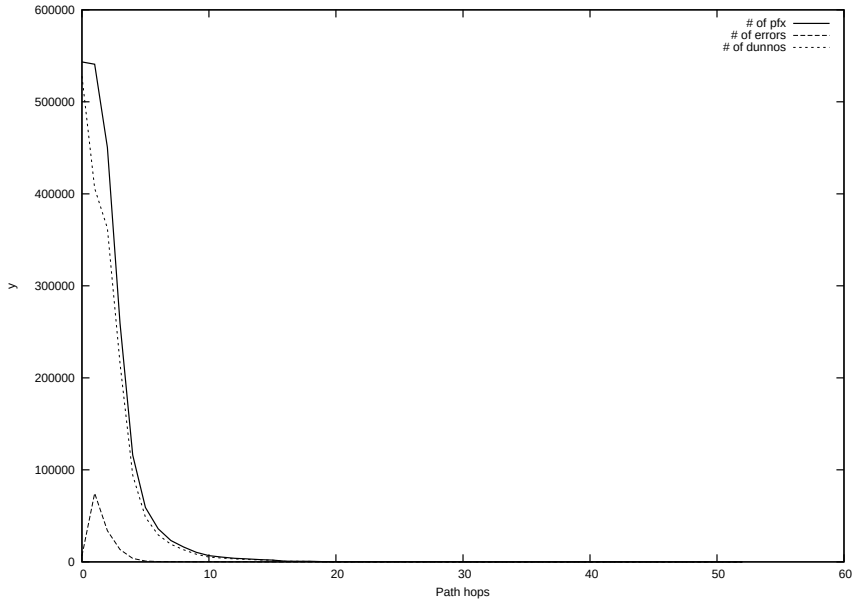


Figure 5.32: IPv4 BGP filter matching along the paths

```
Hop 43 : 23 pfx traversed, 3 ok, 0 errors, 20 dunnos
Hop 44 : 23 pfx traversed, 3 ok, 0 errors, 20 dunnos
Hop 45 : 23 pfx traversed, 3 ok, 0 errors, 20 dunnos
Hop 46 : 23 pfx traversed, 3 ok, 0 errors, 20 dunnos
Hop 47 : 23 pfx traversed, 3 ok, 0 errors, 20 dunnos
Hop 48 : 23 pfx traversed, 3 ok, 0 errors, 20 dunnos
Hop 49 : 3 pfx traversed, 3 ok, 0 errors, 0 dunnos
Hop 50 : 3 pfx traversed, 3 ok, 0 errors, 0 dunnos
Hop 51 : 3 pfx traversed, 3 ok, 0 errors, 0 dunnos
Hop 52 : 3 pfx traversed, 3 ok, 0 errors, 0 dunnos
```

Please note that the first hop, that is for vast majority of prefixes unfortunately the American company, which actually breaks the validation of the entire prefixes. But the rest of the paths shows more reasonable numbers of validated as well as failed prefixes and far less unknown results.

The last portion of the summary text output is actually the decomposed right most point of the chart 5.31:

```
Avg path length: 3.83
Avg dunnos per path: 3.22
Avg errors per path: 0.25
```

The presented text outputs can be visualized to make the point clear. The figure 5.32 shows the number of errors in relation to number of prefixes and number of undecidable hops along the path lengths on June 21, 2015.

It is also interesting to analyze the structure of prefixes with respect to path validation failures and see whether there is some interesting pattern. The figure 5.33 shows the number of valid hops, failed hops and undecidable hops for all possible prefix lengths on June 21, 2015.

The table 5.10 summarizes hop failure observations in this chapter. It shows the three different times when the observation point utilized different upstream

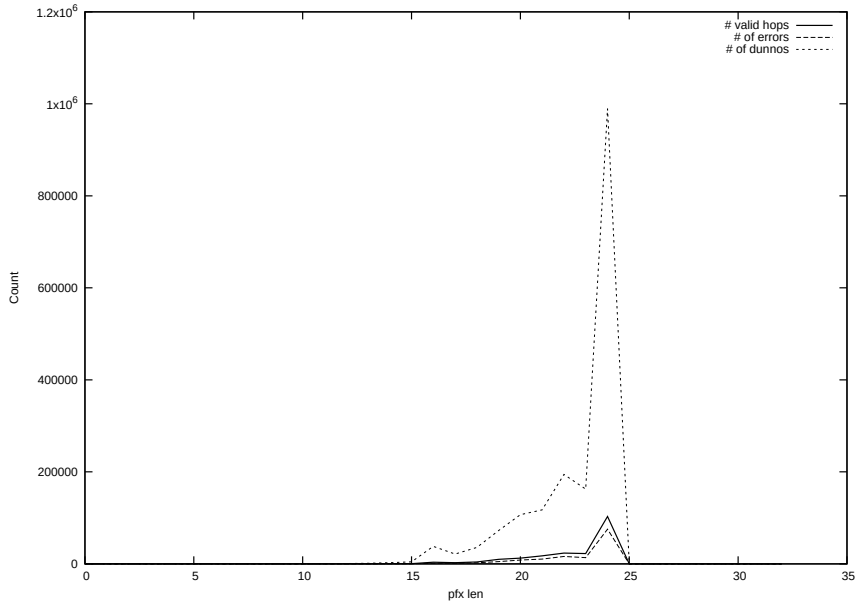


Figure 5.33: IPv4 BGP filter matching results per prefix length

	2012-03-22	2014-03-20	2015-06-21
Valid	524723	243988	199971
Unknown	807522	1227030	1748076
import not-found	102792	126817	49342
import invalid	36181	156308	29628
export not-found	47749	75866	50933
export invalid	5543	5195	5227
Total hops	1524510	1835204	2083177

Table 5.10: IPv4 Path AS transition validation results

providers. At the beginning it was European leading ISP, then in the middle the routing table consisted mostly of an European low cost and the USA based low cost providers. In the end there is only one low cost provider based in the USA and any other providers are kept in backup mode. It is clearly visible from the table and from the timelines as well, that the validation errors are persistent problem. But it actually matters the most what upstream provider is being used because some have more documented paths and some do not actually bother with the record keeping, which has great impact on our results.

The table 5.11 shows summary of entire path validation results in the three selected times.

Though the presented numbers prevent us from judging on global trends it is still valuable observation. It clearly shows that over the time the changes made it for worse in the **AS29134** that hosted the observation point, even though it is pretty decent and respectable Czech ISP. It also shows that despite high number of undecidable paths and hops, we can say that the portion of IPv4 DFZ that we could decide on, shows large number of errors of various kinds. And it is definitely consistent and long-term issue.

Please refer to the appendix in the electronic version of this thesis for the rest

	2012-03-22	2014-03-20	2015-06-21
Valid	31063	11699	5234
Filter fail	59756	22799	8900
Unknown	306015	455926	529193
Total prefixes	396834	490424	543327

Table 5.11: IPv4 Path validation results

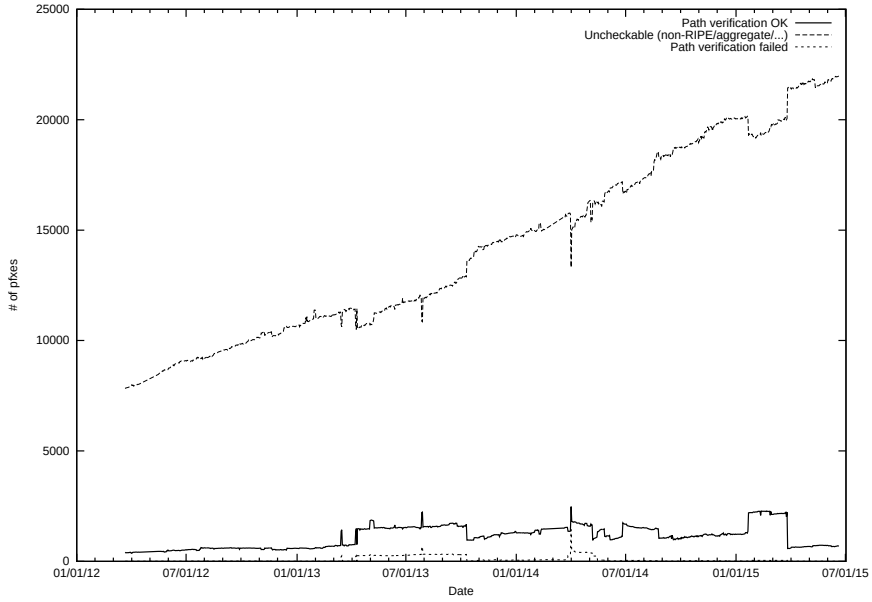


Figure 5.34: IPv6 BGP path validation timeline

of the daily results. And even more fine-grained and detailed information are published in text form on the above mentioned web site.

5.5.2 IPv6

In the following section on IPv6 I am going to present the same type of data from the same points of view and at the same points in time as for the IPv4 for easy comparison.

The figure 5.34 shows the summary of path validation results for IPv6 through the entire captured period. There holds the same observation as for IPv4 - the chart is too coarse to judge any trends or global tendencies. We can also refer to the figure 5.13 to see possible correlation with the IPv6 average path length timeline.

Validation results in the more detailed figure 5.35 show how many errors of each type occurred during the entire validation process. It works in exactly the same manner as in the IPv4 case. I also encourage to compare the chart with the numbers for IPv4 prefixes in DFZ as well as the corresponding numbers of IPv6 prefixes in the DFZ in the figure 5.7 and the average IPv6 path length in the figure 5.11.

The figure 5.36 shows the timeline of average path length and it allows comparison with the timeline of average number of errors per path and average number

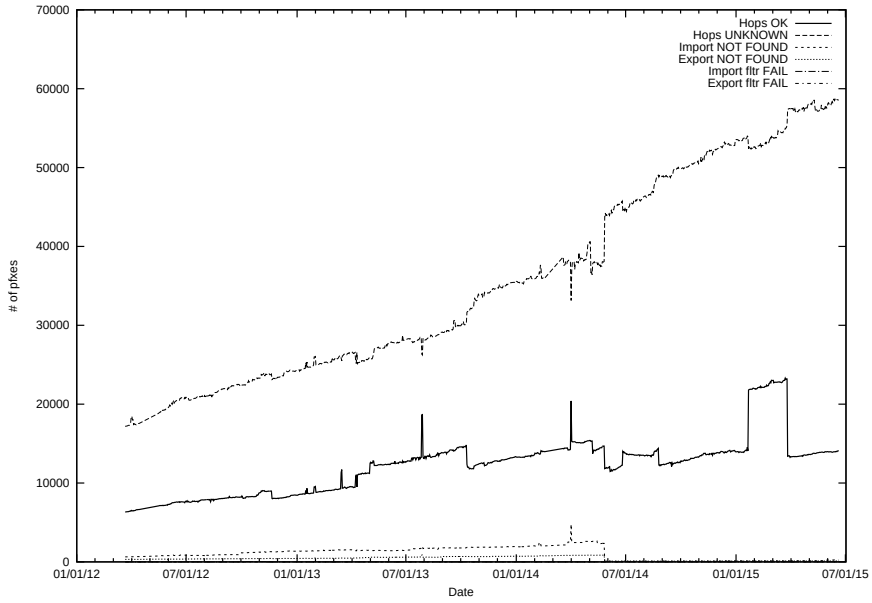


Figure 5.35: IPv6 BGP path validation details

of correctly validated hops per path.

The detailed results for each prefix in DFZ and for each day within the captured period are in text form on above mentioned web site in section **Daily IPv6 BGP path matching report (text)**. The latest summary report for the last day of the captured period - June 21, 2015 follows:

```
Total prefixes: 22668
Path verification OK: 680
Uncheckable (non-RIPE/aggregate/...): 21984
Path verification failed: 4

Total hops observed: 72978
Total hops valid: 14063
Total hops unknown (non-RIPE/aggregate/filter-dunno): 58558
Total hops unknown due to filter syntax error (included in dunno): 2
Total import filter not-found: 224
Total import filter invalid: 5
Total export filter not-found: 125
Total export filter invalid: 3
```

Please note the extremely low number of the decidable paths - both the correctly verified and the failed. The problem is that the IPv6 network is less dense than the IPv4 and this is the reason why it depends even more on upstream, which is entirely undecidable for us, because it is the USA based company that operates in ARIN service region. On the other hand it is pleasant surprise that peerings and directly connected networks, the only decidable ones, are almost completely correct.

And still, we can find interesting information in the following outputs from the very same day. The number of observed prefixes and verification status based on the length of the path from the observation point to the originators are displayed in the following output:

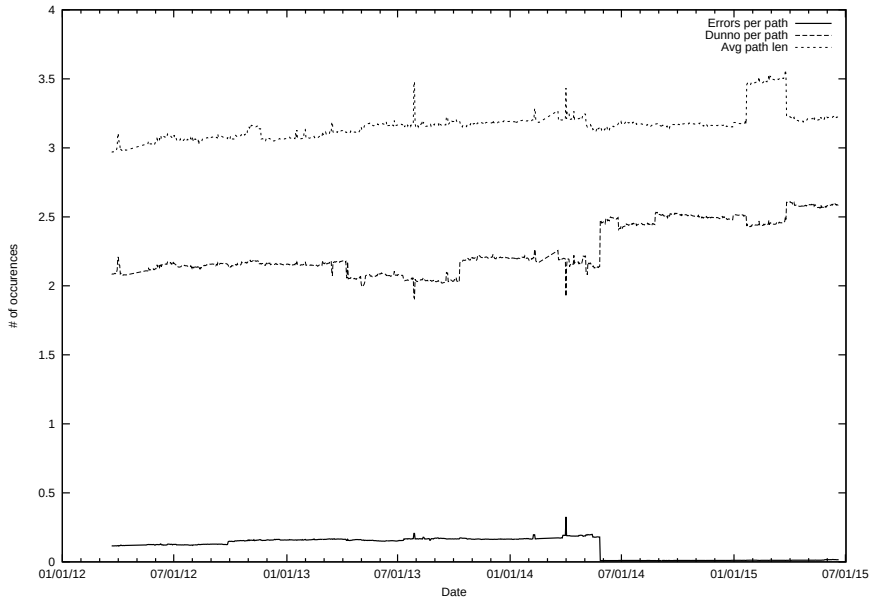


Figure 5.36: IPv6 BGP validation errors per path

```
Hop 0 : 22666 pfx traversed, 1333 ok, 0 errors, 21333 dunnos
Hop 1 : 22436 pfx traversed, 6420 ok, 286 errors, 15730 dunnos
Hop 2 : 16165 pfx traversed, 4164 ok, 43 errors, 11958 dunnos
Hop 3 : 6958 pfx traversed, 1431 ok, 22 errors, 5505 dunnos
Hop 4 : 2306 pfx traversed, 354 ok, 6 errors, 1946 dunnos
Hop 5 : 1131 pfx traversed, 169 ok, 0 errors, 962 dunnos
Hop 6 : 433 pfx traversed, 71 ok, 0 errors, 362 dunnos
Hop 7 : 292 pfx traversed, 38 ok, 0 errors, 254 dunnos
Hop 8 : 193 pfx traversed, 29 ok, 0 errors, 164 dunnos
Hop 9 : 145 pfx traversed, 22 ok, 0 errors, 123 dunnos
Hop 10 : 75 pfx traversed, 14 ok, 0 errors, 61 dunnos
Hop 11 : 45 pfx traversed, 6 ok, 0 errors, 39 dunnos
Hop 12 : 37 pfx traversed, 2 ok, 0 errors, 35 dunnos
Hop 13 : 35 pfx traversed, 1 ok, 0 errors, 34 dunnos
Hop 14 : 26 pfx traversed, 1 ok, 0 errors, 25 dunnos
Hop 15 : 8 pfx traversed, 1 ok, 0 errors, 7 dunnos
Hop 16 : 7 pfx traversed, 1 ok, 0 errors, 6 dunnos
Hop 17 : 7 pfx traversed, 1 ok, 0 errors, 6 dunnos
Hop 18 : 7 pfx traversed, 1 ok, 0 errors, 6 dunnos
Hop 19 : 3 pfx traversed, 1 ok, 0 errors, 2 dunnos
Hop 20 : 1 pfx traversed, 1 ok, 0 errors, 0 dunnos
Hop 21 : 1 pfx traversed, 1 ok, 0 errors, 0 dunnos
Hop 22 : 1 pfx traversed, 1 ok, 0 errors, 0 dunnos
```

I would say that this exhibits much more favorable numbers than the IPv4 case. Though it might be a temporary thing, because we are still in the middle of IPv6 transition period and it probably means that people, who do not generally care about technical details, technological correctness and rules in general, have not yet joined the IPv6 DFZ.

The last portion of the summary text output is actually the right most point

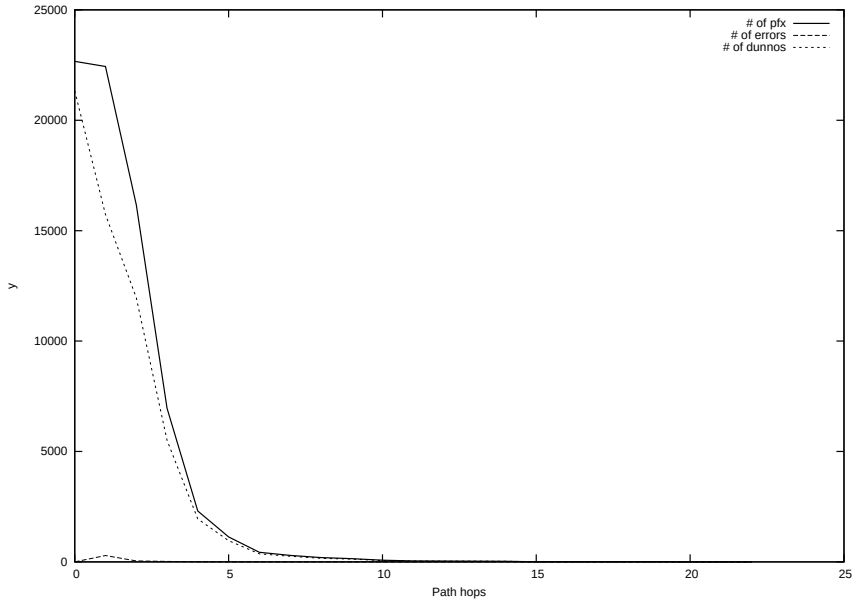


Figure 5.37: IPv6 BGP filter matching along the paths

	2012-03-22	2014-03-20	2015-06-21
Valid	6369	14389	14063
Unknown	17175	38674	58558
import not-found	634	2151	224
import invalid	0	3	5
export not-found	314	823	125
export invalid	0	2	3
Total hops	24492	56042	72978

Table 5.12: IPv6 Path AS transition validation results

of the chart 5.36:

```
Avg path length: 3.22
Avg dunnos per path: 2.58
Avg errors per path: 0.02
```

Visualization in the same manner as in IPv4 case follows. The figure 5.37 shows the number of errors in relation to number of prefixes and number of undecidable hops along the path lengths on June 21, 2015.

The figure 5.38 shows the number of valid hops, failed hops and undecidable hops for all possible prefix lengths on June 21, 2015.

The table 5.12 summarizes hop observations in this chapter. It shows the three different points in time, the same as in IPv4 case.

The table 5.13 shows summary of entire path validation results in the same selected time points.

The presented numbers shows the IPv6, still in its infancy, from three different points of view or better through three different upstream mixes in three different points in time. I must say that I am pretty pleased by the low number of hard failures in the results, but still it might be the result of the low popularity of IPv6

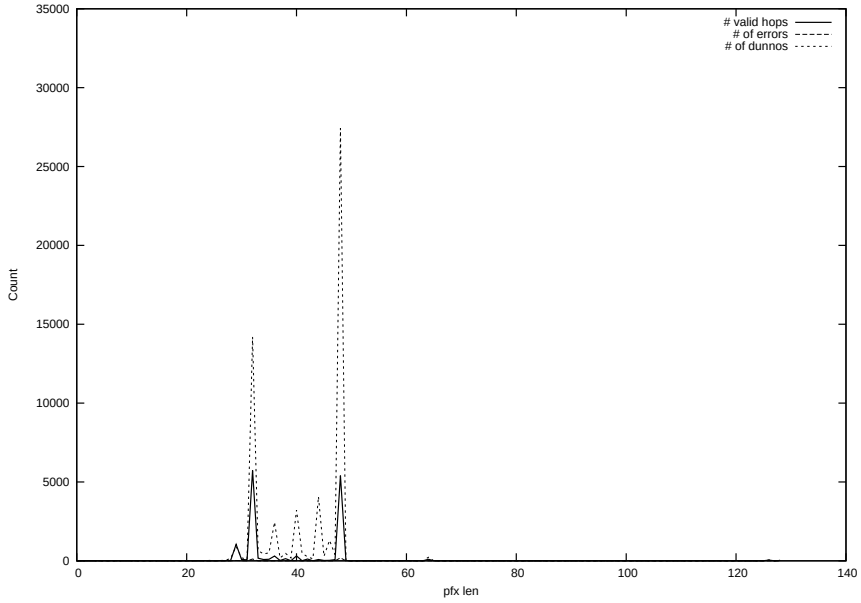


Figure 5.38: IPv6 BGP filter matching results per prefix length

	2012-03-22	2014-03-20	2015-06-21
Valid	415	1505	680
Filter fail	6	63	4
Unknown	7825	15573	21984
Total prefixes	8246	17141	22668

Table 5.13: IPv6 Path validation results

among common wrong-doers. The people, who undermine the accuracy of IPv4 routing policies, are perhaps yet to come.

At the end of this chapter I have to emphasize that we discussed only a small portion of the results that are sufficient to prove the point, which is that the routing policies in RIPE DB are inaccurate and incomplete to the point, where their overall value becomes questionable.

The complete results, published on the web site `http://aule.elfove.cz/~brill/bgpcrunch`, contains over 7000 charts and several gigabytes of text outputs to read through. It is extremely interesting and sometimes very helpful to browse through some of the results and even look up the data, subjected to the analysis, in raw form in various sources, to learn about the Internet routing system in depth.

6. Future of RPSL

6.1 Verdict on RPSL state

In the previous chapters I have presented the description of RPSL standard as well as the history of the standard development and a few notes on connection between the RPSL standard evolution and major changes in the Internet. Even though there are pretty extensive uses of the RPSL and despite the huge amount of data collected in the RIPE DB and in other routing databases as well, we were able to see in the previous chapter that the Internet is definitely not being configured from RPSL. Moreover, we can see that despite the great effort put in writing the parser software, implementing the majority of RPSL features and using the most extensive data source for RIPE NCC service region - RIPE DB, we were not able to completely decide on majority of path vectors in BGP, that are observable almost in the geographical center of the service region.

In the minority of data, we were able to decide upon, which still represents substantial amount of Internet resources, we found out that while the *origin* attributes in *route* and *route6* objects match the real originator in major portion of decidable DFZ table records for both protocols, the failed routes are still no exceptions. For IPv4 it is over a quarter of the decidable routes that failed to validate and for IPv6 it is slightly less then a quarter. It means that filtering routes from all directions, based on *origin* stated in RPSL routing policies, is totally out of question, because it would simply break the Internet connectivity. It is possible to do that in Internet Exchange Points because the IXP usually brings only better connectivity to already connected Autonomous Systems and therefore dropping routes, that do not pass RPSL filters, is possible, sometimes desirable and it would not break the network connectivity in most cases.

The verification of the entire paths ended up in an absolute catastrophe for both IPv4 and IPv6 AFIs. We can ignore the first hop - upstream of the observation point that actually obfuscated the results in summaries, because it was the AS from ARIN service region and validation of that is out of scope in this work. But we can judge from the absolute number of failed hops, that the filters simply does not work. I tried to dig deeper into that problem and infer what is mostly wrong, but despite the effort I was unable to pinpoint one or a few common errors, because the range of possible differences among the RPSL policy and the real BGP state is so broad, the paths are changing constantly and the *aut-num* objects exhibit almost all thinkable discrepancies with respect to the real DFZ. It simply does not work at all. There are Autonomous Systems that use RPSL seriously for generating configurations of their routers, but they keep maintaining their own objects and sometimes even their customers's objects, but they definitely do not trust and reference other sides, especially remote ones.

It seems that our observation proved the grim assumption form the beginning: A lot of people, even in the RIPE NCC service region, think of the RPSL records as of a nuisance. A lot of them fill the objects in RIPE DB for the first time with some temporary data at the beginning, when the ASN is allocated and then forget about it and never update. Another people simply misinterpret the standard or do not have time to read through RFCs and take care of their records, especially

when they think that nobody actually needs them.

I have to say that even though the concept of RPSL is technically advanced and despite the great effort RIPE NCC has put into educating the community in the service region about RPSL, its actual operational value seems to be quite low.

6.2 Relation among RPSL, RPKI and NetConf

RPSL language raised pretty high expectations back in 1990's, when it was invented. The expectations was, among others, improvement of BGP security and support for automated configuration of the routers. It turns out that after over twenty years of RPSL existence the routers are still decoupled from routing policies. The creation and maintenance of the network configurations is much more complex process than writing down routing policies and translating them to the router configuration. Currently it involves more stake holders, than simply technical staff. The routers themselves, with constantly evolving features, are still the most important factor in network operation, rather than abstract policies and high-level approach.

In fact unfulfilled expectations in configuration automation and growing operation costs of heterogeneous networks led to NetConf initiative in mid-2000. The basic protocol is described in RFC 6241 [28], but there are more derived standards that forms the NetConf stack. Basic idea is to define the protocol and a basic data model for router configuration and still leave some room for vendors to define their specific extensions. As far as I know the protocol is a moderate success, it has numerous implementations and major networking gear vendors support it both on implementation side as well as throughout the standardization process.

The security of BGP is more peculiar problem, because there are two opposing ideas constantly clashing: The need for more security and more control encounters the desire for preservation of freedom of speech in the Internet, technical robustness and inherent technical independence on governments. RPSL itself lacks dedicated security mechanism and the whole technology is based on dated protocols like *whois*. Even RPSL object modification in public routing databases are being authenticated by ancient methods, like appending *crypt* function result to the modification request. The only serious attempt to secure Internet BGP routing infrastructure is a technology called Resource Public Key Infrastructure - RPKI. The standard, or rather a family of standards, that describe provisions and operation of possible Resource Certification consists of quite a lot RFC documents. Among the most important are RFC 6810 [29] and RFC 6811 [30]. Detailed description of this protocol stack and mechanisms is clearly beyond scope of this work, but the basic idea is to build a PKI resource certification tree. The tree parallels addressing resource allocations and the coordination bodies as well as Internet community members could transfer authority over parts of the addressing space, that has been allocated to them by superior party, to some subsequent resource holder, forming a tree of delegations. The protocol is currently implemented in software, but apart from experiments and proof-of-concepts, it has not been seriously deployed anywhere in the Internet. The reason is apparent high complexity, need for additional resources and fear of giving up the decision authority on routing to the automated system that is hierarchically controlled.

Even though the RPKI is designed as opt-in solution, the network operator should be still in charge and should be able to easily override the decisions of the RPKI tools, some people still fears the censorship opportunity and possible loss of freedom on the Internet due to RPKI. The opponents of RPKI argues that when the system is really deployed, it would be possible for the courts in certain countries to order the coordination bodies to revoke certificates of certain resources, which would effectively kill the switch for those resource holders. And I personally doubt that the RPKI technology is going to be widely adopted anytime soon, if ever, because several aspects of the system obviously provoke aversion and resistance of the Internet community members. And the BGP security issues seems to be only a minor problem so far.

I think that RPSL-like system still has its place in the Internet operational infrastructure but it has to become more specific and specialized. In my opinion there is no point in competing or supplementing some functionality of RPKI or NetConf. In fact I believe that RPSL failed also as a data model, because of its complexity and unusual format. There are currently ongoing disputes in RIPE mailing lists about the format of RPSL and even the most experienced people are unsure where to put it in Chomsky hierarchy. (It is most likely that RPSL is context-sensitive language, which translates to type-1 grammar.)

Apart from functionality issues I think that there are more problems that need attention. The format is quite hard to parse and there are few or actually no libraries for doing that. Associated protocols are quite outdated and even though some of routing databases started offering *RESTful* APIs, the basic protocols for obtaining RPSL data are still *whois* and *FTP*. There are many different databases but the language does not provide any mechanism for linking them together. And it actually creates another problem with duplicate objects and conflicting information in different databases. This particular problem is not covered by any standard and not even discussed in best practices documents and therefore it is up to the implementation to find a solution.

6.3 Possible improvements and actions

I believe that in order to rectify RPSL routing policies the Internet community has to make it more attractive to its own members. The routing policy definition has to become less troublesome and more transparent. And routing policy evaluation has to be easy to do both in practice as well as in simulation.

For achieving that we can:

1. Create comprehensive and truly easy to use tools for the current RPSL version.

2. Keep the basics of the language and formats, but drop a lot of features, to make it really simple and easy to use.

3. Re-design the format while keeping the most of the features and even add some new ones.

4. Both radically redesign the format and drop features to reduce complexity.

In my opinion the reduction of complexity is necessary, because I find it intriguing even after years spent by operating various Autonomous Systems in different European countries and writing two theses on that. And I know about very few people who truly understand RPSL into the detail, needed for writing down real-life routing policy for medium-sized ISP from scratch.

Another approach that could possibly improve the usability of current RPSL is creating an application stack that would effectively isolate users from the RPSL language and provide an interface that would be more pleasant to use and more transparent for the users. There are few parties that use RPSL extensively and that might be interested in doing that. I am aware of an effort ongoing in NLNet Labs that focuses on creating modern application toolset in Python, that should help with RPSL but as far as I know they still struggle with efficient RPSL parsing. The obvious problem is that the only complete parser and interpreter of RPSL is in fact the IRRToolSet, which is the reference implementation of the RPSL stack. And there are no complete and supported RPSL libraries written in scripting languages so far. I think that creating one in Python, for instance, could help substantially because if there is the library with complete functionality and easy-to-use interface, applications would follow. Another problem is that the language parser for RPSL is pretty hard to create. And besides that, there are not many linguists and experts on formal languages in the networking community, who would be willing to put the effort in creating new software for an old and unpopular language. At least it should be possible to use general language parsers that accept metasyntax rules in Extended Backus–Naur Form to do the parsing work. But still parsing of RPSL is only the beginning, because there is no obvious and straight-forward way of transforming RPSL data into some data model. The problem is potential size range of data as well as links among objects and different searching mechanisms. And it is not easy to interpret even the parsed RPSL data correctly afterwards. Therefore creating the RPSL library would be definitely a huge endeavor and would take a lot of time and effort to do it and to do it right.

Redesigning the language is a tempting option because there is no doubt that lot of people would be happy for any reform of RPSL. But at the same time it would certainly make some people extremely unhappy, especially those who have invested huge money and effort into writing software that use the current version of RPSL. The main problem would be what to adopt in the new language and what to drop. There are two different approaches possible: Adopt more features than the current RPSL offers and pack them in a better format like XML, YAML or JSON for instance. Or create a new description language that would be super-simple to parse and to understand with much less features than the RPSL have today. Unfortunately, it seems to me that both possible ways have strong supporters as well as opponents and it will be extremely hard to reach some consensus on the way forward with RPSL reform. Besides people seem to be generally reluctant to put any effort into reforming something that unpopular as RPSL.

6.4 Possible RPSL reform

To enjoy the privilege of doing academic work to its full extent I am extending this thesis with a rough outline of possible RPSL reform.

Apart from choosing proper format for the data, which, I believe, should be one of the above mentioned standard data markup languages, we have to discuss features that the new routing policy language should have and the features that we could possibly sacrifice for simplicity, readability and usability.

6.4.1 Useful features to adopt

The following list contains mainly features I find successful in the current RPSL as well as features I deem exceptionally useful.

1. Binding of prefixes to their originators. Still, I believe it should use simpler structure than current *route* or *route6* objects. I would prefer the origin autonomous systems to list their prefixes in one place.

2. Autonomous system neighbor lists and import as well as export filters. The filter format should be more transparent and force uniform markup of the routing policies.

3. There has to be a provision for recursions but I think that it should be limited to the filter parts (terms) and lists of objects, most notably lists of originating autonomous systems.

4. Means for linking objects among different databases.

5. Tools for inheriting pieces of routing policies from remote autonomous systems and authority delegation. This feature should allow direct configuration of route servers and infrastructure of upstream providers from the customers' autonomous systems.

6. Personalized authentication, finer user rights and stronger authorization model and even some sort of object signing.

A language for specifying the routing policies is still necessary, but the methods for stating routing policies to peering partners or to service providers utilized in reality are often quite rudimentary, like sending mails with lists of prefixes or pieces of vendor-specific configurations. The ability to accurately specify routing policy is going to be even more important in future. The reason for that is likely the pressure on cutting costs of network operation and on more automation. It is also likely, that the backbone networking is going to be more integrated into service provisioning systems and therefore the rate of changes in the traditionally stable and slowly-evolving systems is going to raise. The prerequisite for doing a lot of changes and automating them is to have a proper data model and a standard way of sending information amongst cooperating parties. Another challenge in this area is perhaps going to be the onset of SDN. It is likely to bring the same problems with proprietary blobs of different kind to the networking, as the software-defined-* technology has brought proprietary lock-in to other fields, that implemented the software defined functionality earlier, like Software Defined Radio for instance. And the same problem lies within provisioning and orchestration systems, that might bring vendor lock-in and proprietary data formats even to places where already a viable standard exists. In fact I think that it is

highly unlikely that any commercial vendor would be willing to embrace and use RPSL instead of inventing own format, based on XML, for instance.

Even though the projected scenarios towards SDN and automated provisioning systems might look like thoughts about future corporate application equipment, that merely helps to isolate people from hands-on configuration on the real network, there are places that would strongly benefit from easy way for producing the structured routing policies today: IXPs. The problem with IXPs is that there are a lot of different parties that generally cooperate and share information but the many-to-many connection model that dictates to set up $N * (N - 1)$ BGP sessions in total is exceptionally unpleasant for the administrators. To alleviate the problem IXPs offer the route server service. It is often perceived as an alternative to setting up many BGP sessions that takes away not only the unnecessary workload but also the control over the routing policy. People tend to either use the entire route server BGP feed without any filtering or not to use the route server at all. The reason is that there is no common and easy way for defining and adopting the routing policy. Using custom BGP communities for BGP propagation control, which is quite common nowadays, adds unnecessary complexity, it is not transparent enough and it is quite hard to debug.

6.4.2 Features to drop

On the other hand I believe that there are features in the current RPSL that makes things unnecessarily complicated. Even though I think that the right way forward is to create a new data model and use existing markup language to write down or transfer the data, I feel need to explicitly state what not to do and what I consider to be the most troublesome in current RPSL.

1. Unlimited recursion. I think that recursion in current RPSL can be quite nice feature of the language but it do not serve nicely to neither party involved in the process. The author of the RPSL object has to keep track of the recursive graph he refers to. The *whois* server need to understand object semantics in order to return the linked objects from the requested one to save time, resources and speed up processing. And the user that resolves the recursive graph usually need to repeatedly ask for more objects even though the *whois* server helps quite a lot with the additional answers. In fact the recursion helps to keep data amount down to several hundreds of megabytes for the whole RIPE DB, which is nice, but not necessary. Sure, it is really helpful for people writing down the routing policies by hand. But I think that software should take over in the creation part and the potential for recursion should be limited to minimum, though links among objects have to persist.

2. Unnecessary *-set* objects. The most notable examples are *rtr-set* and *peering-set* objects that are quite rare and when being used, the objects allow to substitute several hundreds of bytes by a symbolic name, which is several bytes or tens of bytes long, at the expense of recursive lookup. Actually I think that RPSL took the abstraction level to unnecessary extreme and that might be one of the reasons why it is so unpopular.

3. The backward-pointing references using the *member-of* attribute are not particularly popular feature of RPSL, but it makes everything much more complicated. The *whois* servers have to keep special indexes for them and resolution of certain *-set* objects is hard to do without indexes or searching through a lot of another object of different kind. The most visible example is the *member-of* attribute in *route* objects. On June 21, 2015 there were only 4147 objects with *member-of* attribute but resolving each *as-set* means searching through all 261 187 *route* objects in RIPE DB, or creating a special index for them.

4. AS path filters that might be defined by an expression, which consists of a subset of the extended regular expressions. It represents a problem, because it is quite hard to interpret them. The idea behind is that the RPSL regular expressions are likely to be more restricted than any other regular expressions that are implemented in router operating systems. When this presumption does not hold, which is more likely with the nowadays' fragmented network gear market, the regular expressions often can not be simply translated or modified to the required form. Another problem is that evaluating regular expressions is slow and it even poses a threat to the destination system if the AS path filters get simply copied from some public database to the router. I think that this needs more formalization and maybe more restrictions.

5. Overall complexity of the filters with *refine* and *except* statements is perhaps the worst problem, especially because it uses algebraic notation, while in the end of the day, the filters have to be translated into lists of matches and actions, like for instance *route-maps* in Cisco IOS configurations. Problem with algebraic notation is that it is inherently harder to understand and evaluate it by hand. On the other hand the algebraic notation allows easy substitution of certain blocks, which sounds like a nice thing, but it brings complexity, destroys readability and needs recursions, which was the topic discussed above. Even though there are parties that use filters extensively, the most common case is to use of simple lists, few links to *-set* objects and only keywords like *ANY* or *PeerAS* instead of complex custom filters.

To sum up this section, I want to say that should the RPSL undergo transformation, I believe, it need to accent readability and simplicity rather than effectivity of manual markup. The reason is, that the information has to be easy to understand and easy to write tools for processing it. Without that it will not serve the purpose. Some repetition in markup seems to be more acceptable than a complex language, because repetitions are easy to alleviate by trivial scripts, but learning yet another complex language is simply too much for the most of network administrators.

6.5 Possible shape of RPSL reform

Even though this thesis has described a lot of serious problems in RPSL and the surrounding infrastructure, it is only the first step in a long and complicated process of creating a new technology. The possible reform of RPSL would affect

a lot of people with different needs and views. The obvious points of interests are:

1. *whois* protocol, which is now de-facto standard for requesting data concerning the IP resources as well as domain registry data, but at the same time new RESTful alternatives emerge and are being deployed alongside the *whois*, to help integrate new tools, add transactions, add national charsets and UTF-8 support and provide secure write access in the same channel as read access.

2. Text data format, which now suits well the *whois* protocol but it makes it hard to structure data and parse them afterwards.

3. Links among routing databases. This feature is not present in the current routing system and it is one of the most visible shortcoming of RPSL. I think that the future reform should establish a hierarchy of data sources in accordance with the resource distribution tree. On the other hand it have to be possible to create and maintain private or alternate data sources or even alternative hierarchy in the same manner as DNS hierarchy works.

4. Data authentication. The current protocol does not provide any mechanism for that, but I think that although I do not consider that absolutely necessary, it will be one of the requirements. The idea behind is that the routing database could be used for security purposes, both in forensic analysis as well as in filter generation and in many security incident prevention mechanisms, but the data need to be authenticated and verifiable before we can to do that.

I would say that departure from *whois* protocol would not be a huge source of controversy, even though RIR databases are now converged in *whois* service, where the resource allocation and registration data are mixed with routing policy data and even with additional administrative data.

The new text format and authentication are perhaps the most visible suggested modifications and I think those will have the greatest potential for generating controversy, because brand new format means that all current tools have to be reworked or even thrown out and a substantial investment into the new ones will be required. The authentication has to be carefully crafted in order not to threaten freedom and cooperative nature of the Internet, but at the same time not to be merely an opt-in feature. And since the routing information authentication is a subject of RPKI standard, I think that certain amount of people would be against that feature or even the whole idea of RPSL reform because of that.

Another issue is the amount of existing data in RPSL format. It would be exceptionally useful to convert the existing data to the new format. But it might prove impossible if the new format would lack some of RPSL's features. In fact there will be needed a substantial research effort into the question, how to parse existing RPSL data along with all the extensions that are generally not supported by the current tools, but the data using them are present in routing databases. One example for all are the *import-via* and *export-via* attributes from draft [20].

And even before the technical details can be discussed, the entire Internet community has to discuss the more fundamental questions regarding the routing

policies. The most important question is whether we need to develop yet another routing policy specification language or define rather different aim like "network description language" or "network documentation language". In fact, working group for the possible RPSL reform would need a charter with the aim carefully crafted and clearly stated, discussed and generally agreed upon, before any work on a new language shape starts. The proposed change in one of the Internet's basic technologies need solid and proven reasons. I believe, that this thesis, especially the previous chapter with BGP to RPSL comparison results, could provide them.

7. Future work

7.1 Community role

The research into RPSL and its deficiencies has been a process that I was able to do generally on my own. Discussions about the available tools, procedures and results on mailing lists provided some help, but still it had only a limited impact on the work. I also published my results on the web and notified Database Working Group in RIPE, mostly because I wanted the feedback and their opinion on critical parts of my work.

Unfortunately this topic stands out of most networking professionals' expertise and therefore the community that appreciate the work is fairly small. Academia tends to take Internet standards as granted and seems to prefer focusing on research into new ideas and technologies, rather then reforming old ones. This is the reason why there is no proper research effort into this topic and no recent scientific papers. This topic and the connected issues are subject of RIPE Database Working Group charter and it seems that currently it is the best place to discuss the future of RPSL.

In order to bring up the topic to the community I submitted an abstract for next RIPE meeting in November 2015. The proposal has been accepted and the presentation of my results is scheduled for the RIPE DB-WG.

I have to say that in the Internet community it is needed not only to prove the existence of a problem and offer a technically sound solution but it is also needed to draw attention of substantial amount of people in the community, in order to help with writing draft documents and createing reference implementation of any proposed standard. Then the community have to reach consensus in the standardization bodies in order to adopt the new technology as an Internet standard. All of these steps are exceptionally demanding and hard to achieve even in case when there is no opposing group of people or any competing proposal. I think that in the current situation it is the most important step to exhibit my results to the community, start the discussion of the RPSL reform and seek support for the possible future standardization effort.

I think that bottom line of current RPSL problems was, that the development of the language and tools for RPSL processing by the original authors finished shortly after the standard has been accepted and when they successfully defended their academic works about the standard. It seems that there was no business case for the development of the standard afterwards and so it stopped. Even the history of the IRRToolSet looks like that it was rather charity than a serious support of the fundamental project that belongs to the Internet infrastructure. Obvious question and issue that troubles me a lot is, how to avoid this in future even with a new possible standard.

7.2 Turning ideas into an Internet standard

I would like to outline the standard development process in IETF which is, I believe, the proper place for standardizing the new routing policy description format and system. As I said, I think that it is not enough to have a standard.

It has to become popular in the Internet community and the parties, that deal with large amount of registration and routing policy data, need to cooperate and embrace the new standard in order it to succeed. I believe that it is possible only if the standard is created in cooperation with all these parties and after thorough consideration of all technical details. Trying to set off these discussions is perhaps the best thing I can do in the near future for the RPSL.

The standards development process in IETF is described in the RFC 2026 (BCP 9) [31]. Basically the process of creating an Internet Standard is divided into several phases. A specification undergoes a period of development and several iterations of review by the Internet community and revision based upon experience. Then it is adopted as a Standard by the appropriate body and published as RFC. In practice, the process is more complicated, due to the difficulty of creating specifications of high technical quality, the need to consider the interests of all of the affected parties, the importance of establishing widespread community consensus and the difficulty of evaluating the utility of a particular specification for the Internet community.

I have to admit that I do not have any prior experience with SDP in IETF and therefore I will need to learn a lot before possibly starting that effort. I hope for drawing wider attention to this problem and to the possible reform on RIR meetings and networking conferences, like the RIPE meeting, and for receiving help or guidance along the way.

The first step - showing, how terribly needed the RPSL reform is, made by this thesis and the following presentations will, hopefully, deliver a strong message. And I believe that there is a bright future for routing policies and for the entire Internet routing system, that will benefit from possible RPSL reform in terms of better security, more robustness and greater operational transparency.

Conclusion

The objectives, set in the thesis abstract, have been fulfilled in following aspects: I have thoroughly studied the RPSL standard and many other Internet standards, that are of concern for the topic of this work. I examined the data in RIPE DB, studied the real-life policies and compared the standards with the interpretation of the data by the reference implementation and a few other tools to find discrepancies and possible issues. On basis of this preliminary research I created a software that collects and analyzes the data from BGP and RIPE DB, or potentially from any other routing database, that is compliant to the RPSL standard. The software has been published under OSS license. Thanks to generous support of my former employer, I was able to use the software for conducting a large-scale analysis. Huge amount of computational power allowed me to generate an outstanding set of results that cover more than three years of the Internet routing system operation history.

The thesis gave me opportunity to work on a problem that stands aside from the current interest of the networking research bodies, but it is exceptionally important for the entire Internet. There is no doubt that accurate and up-to-date routing policies represent a fundamental mechanism that divide well operated and maintained network from chaos and disorder. Now, when the most threatening issue for the Internet, the IPv4 address space exhaustion, has its resolution and nobody seriously challenge the transition to IPv6 anymore, it is perhaps the right time to help consolidate the Internet and make it better, safer and more robust.

Bibliography

[1] Hlaváček, T., *Routing policies* Bachelor's thesis, 2011, Prague

[2] Hubbard, K., Kosters, M., Conrad, D., Karrenberg, D., Postel, J. *Internet Registry IP Allocation Guidelines*, RFC 2050, November 1996
http://www.ietf.org/rfc/rfc2050.txt

[3] *Number Resources* Website of Internet Assigned Numbers Authority
http://www.iana.org/numbers

[4] Alaettinoglu, C., Villamizar, C., Gerich, E., Kessens, D., Meyer, D., Bates, T., Karrenberg, D., Terpstra, M. *Routing Policy Specification Language (RPSL)*, RFC 2622, June 1999
http://www.ietf.org/rfc/rfc2622.txt

[5] Blunk, L., Damas, J., Parent, F., Robachevsky, A. *Routing Policy Specification Language next generation (RPSLng)*, RFC 4012, March 2005
http://www.ietf.org/rfc/rfc4012.txt

[6] *List of Routing Registries* by Merit Network, Inc.
http://www.irr.net/docs/list.html

[7] Rekhter, Y., Ed., Li, T., Ed., Hares, S., Ed. *"A Border Gateway Protocol 4 (BGP-4)*, RFC 4271, January 2006
http://www.ietf.org/rfc/rfc4271.txt

[8] Jouanigot, J., Bonito, A., Dupont, F., Fassbender, S., Hillbo, A. Hommes. F., Klein, L., Porten, W., Stikvoort, D., Terpstra, M., Volk, R. *Policy based routing within RIPE*, ripe-60, May 1992
ftp://ftp.ripe.net/ripe/docs/ripe-060.txt

[9] Bates, T., Jouanigot, J., Karrenberg, D., Lothberg, P., Terpstra, M. *Representation of IP Routing Policies in the RIPE Database*, ripe-81, February 1993
ftp://ftp.ripe.net/ripe/docs/ripe-081.txt

[10] Bates, T., Gerich, E., Joncheray, L., Jouanigot, J., Karrenberg, D. Terpstra, M., Yu, J. *Representation of IP Routing Policies in a Routing Registry*, ripe-181, October 1994
ftp://ftp.ripe.net/ripe/docs/ripe-181.txt

[11] Meyer, D., Schmitz, J., Orange, C., Prior, M., Alaettinoglu, C. *Using RPSL in Practice*, RFC 2650, August 1999
http://www.ietf.org/rfc/rfc2650.txt

[12] *IRRtoolset* Trac home
http://irrtoolset.isc.org/

[13] *PERL RPSL::Parser* module in CPAN
http://search.cpan.org/~lmc/RPSL-Parser-0.04000/lib/RPSL/Parser.pm

[14] *BGF project page - SourceForge*
http://sourceforge.net/projects/bgflib/

[15] *NOC project webpage*
https://kb.nocproject.org/display/DOC/Home

[16] Resnick, P. *On Consensus and Humming in the IETF*, RFC 7282, June 2014
http://www.ietf.org/rfc/rfc7282.txt

[17] *IRRToolSet Use @ Deutsche Telekom* ToolSet BOF at RIPE 64
https://ripe64.ripe.net/presentations/210-ToolSet-BOF.pdf

[18] Hilliard N. *Whither RPSL?* Routing WG at RIPE 61
http://ripe61.ripe.net/presentations/231-228-inex-ripe-rome-routingwg-whitherrpsl-2010-11-17.pdf

[19] *Route Registry: who uses them?* Thread in NANOG Mailing List
http://seclists.org/nanog/2000/Oct/271

[20] Snijders, J., Hilliard, N. Internet-Draft, IEEE, Network Working Group *The 'via' keyword in RPSL Policy Specifications*
https://tools.ietf.org/html/draft-snijders-rpsl-via-00

[21] *Web page of NetIDE project*
http://www.netide.eu/

[22] Doriguzzi-Corin, R., Salvadori, E., Aranda Gutierrez A., Stritzke, C., Leckey, A., Phemius, K., Rojas, E., Guerrero, C. *NetIDE: removing vendor lock-in in SDN* NetSoft 2015, London (UK)
http://www.netide.eu/sites/www.netide.eu/files/publications/netsoft2015_demo.pdf

[23] Lepinski, M. Kent, S. *An Infrastructure to Support Secure Internet Routing*, RFC 6480, February 2012
http://www.ietf.org/rfc/rfc6480.txt

[24] Vohra, Q., Chen E. *BGP Support for Four-Octet Autonomous System (AS) Number Space*, RFC 6793, December 2012
http://www.ietf.org/rfc/rfc6793.txt

[25] *bgpcrunch* GitHub project page
https://github.com/tmshlvck/bgpcrunch

[26] *IPv6 Worl Launch website*
http://www.worldipv6launch.org/

[27] Houston, G. *BGP Routing Table Analysis Reports*
http://bgp.potaroo.net/

[28] Enns, R., Ed. *NETCONF Configuration Protocol*, RFC 4741, December 2006
http://www.ietf.org/rfc/rfc6241.txt

[29] Bush, R. Austein, R. *The Resource Public Key Infrastructure (RPKI) to Router Protocol*, RFC 6810, January 2013
http://www.ietf.org/rfc/rfc6810.txt

[30] Mohapatra, P., Scudder, J., Ward, D., Bush, R., Austein, R. *BGP Prefix Origin Validation*, RFC 6811, January 2013
http://www.ietf.org/rfc/rfc6811.txt

[31] Bradner, S. *The Internet Standards Process – Revision 3*, BCP 9, RFC 2026, October 1996
http://www.ietf.org/rfc/rfc2026.txt

List of Tables

List of Figures

List of Abbreviations

AS Autonomous System

ASBR Autonomous System Border Router

ASN Autonomous System Number

BGP Border Gateway Protocol

DoS Denial of Service

FIB Forwarding Information Base

IP Internet Protocol

ISP Internet Service Provider

IX Internet Exchange

MED Multi-Exit Discriminator

NLRI Network Layer Reachability Information

PA Provider Aggregatable

PI Provider Independent

RIB Routing Information Base

RIR Regional Internet Registry

RPSL Routing Policy Specification Language

RS Route Server

RTBH Remote Triggered Blackhole

SDN Software Defined Network

SDR Software Defined Radio

VRF Virtual Routing and Forwarding

A. Appendix

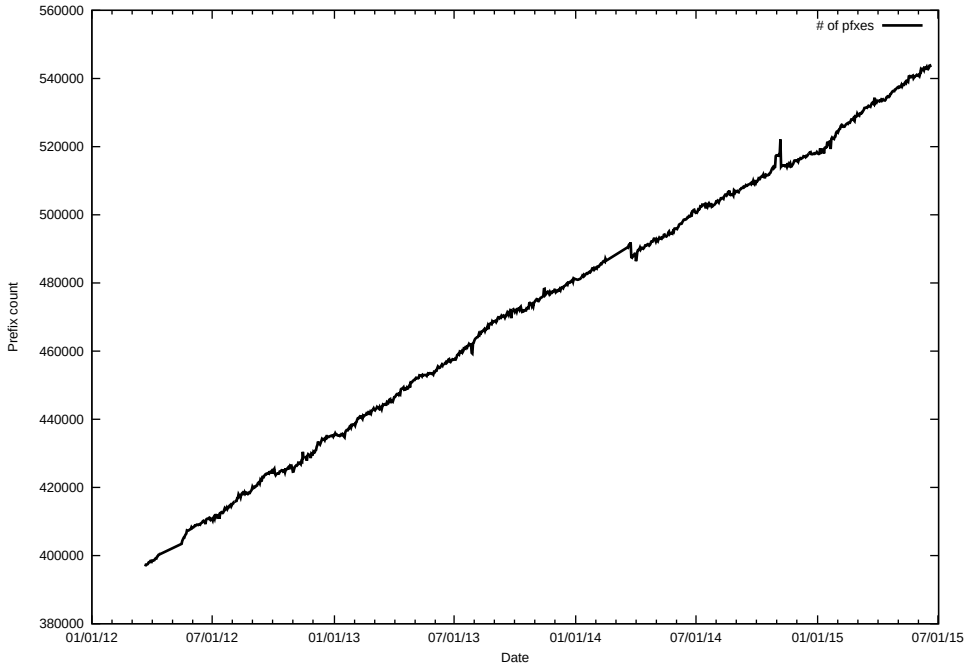
Contents

The appendix is present only in the electronic version of this thesis. It contains all figures generated by the **bgpcrunch** software. The complete results, including text outputs, are published on web page: `http://aule.elfove.cz/~brill/bgpcrunch`. The **bgpcrunch** software itself is available on GitHub: `https://github.com/tmshlvck/bgpcrunch`.

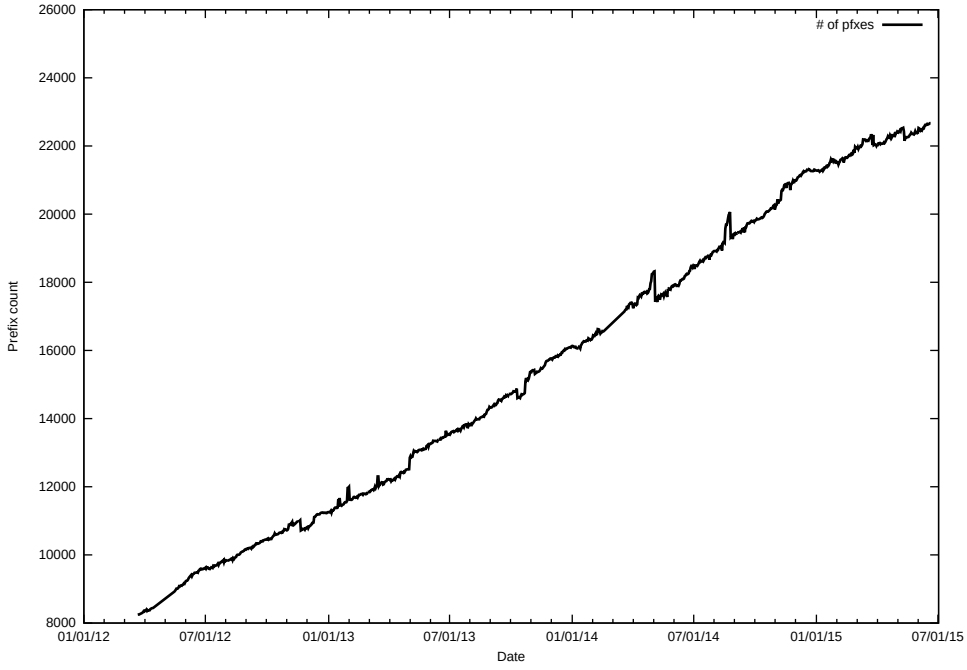
The disc, attached to the printed version, contains still only selected results, usually those, I referenced in the thesis. Contents of the disc is described in the **README** file placed in the media root directory. The sole reason for attaching only limited portion of the results is the size. The complete results are over 750 gigabytes.

A.1 BGP table timeline

A.1.1 Prefix count

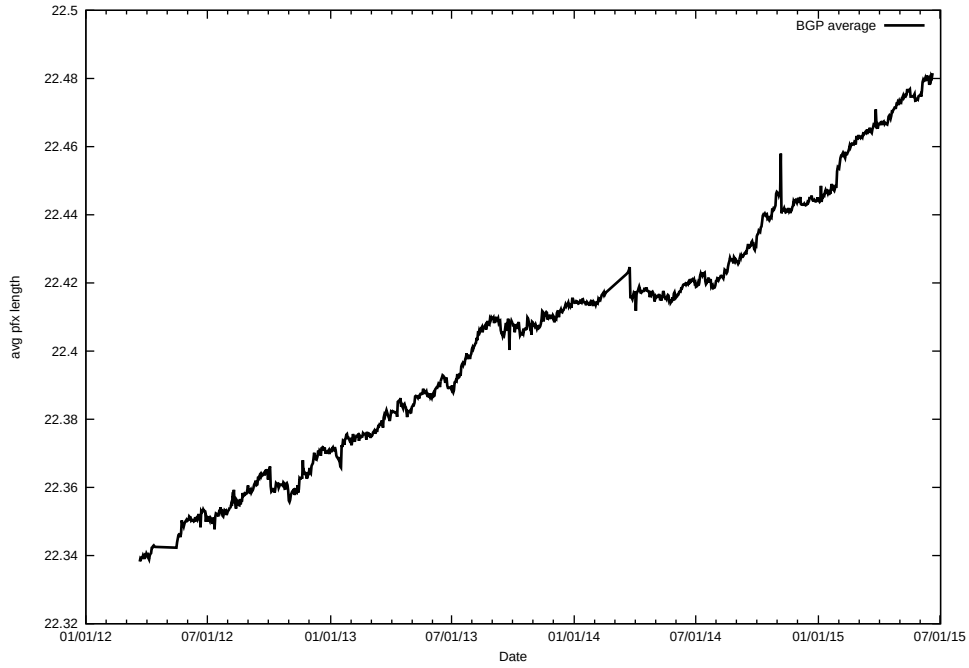


IPv4 prefixes in BGP

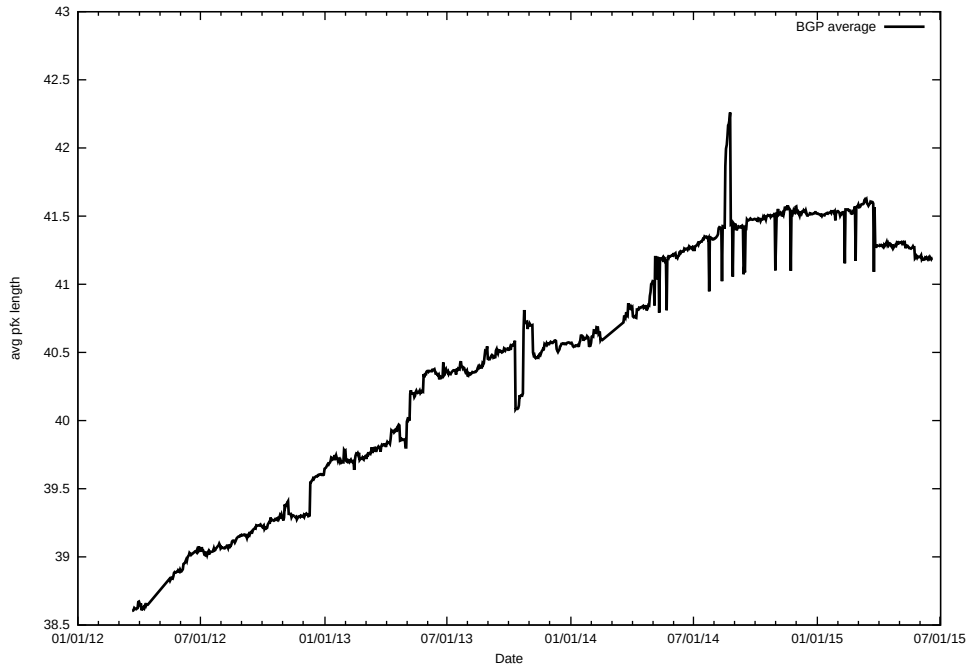


IPv6 prefixes in BGP

A.1.2 Average prefix length

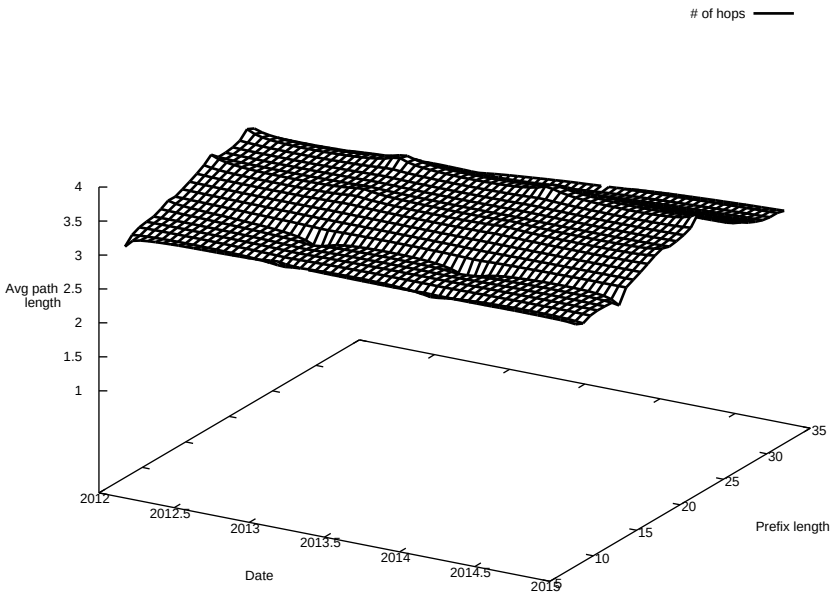


IPv4 average prefix length in BGP

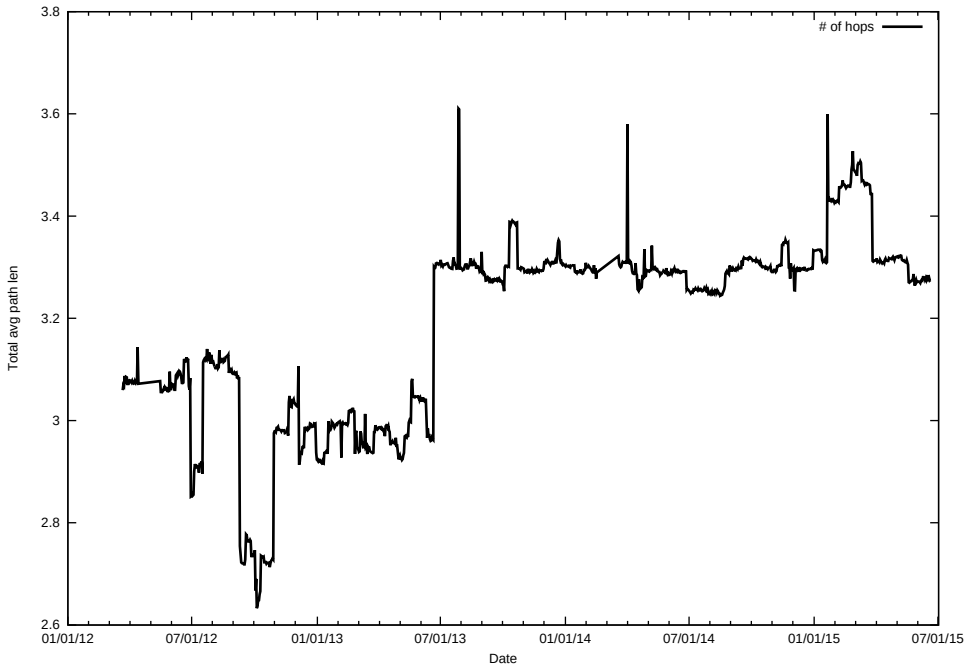


IPv6 average prefix length in BGP

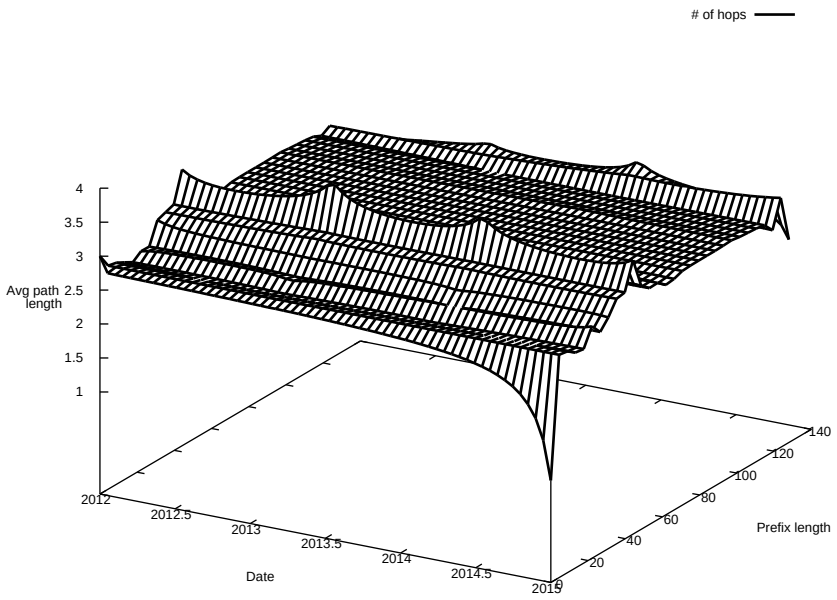
A.1.3 Average path length



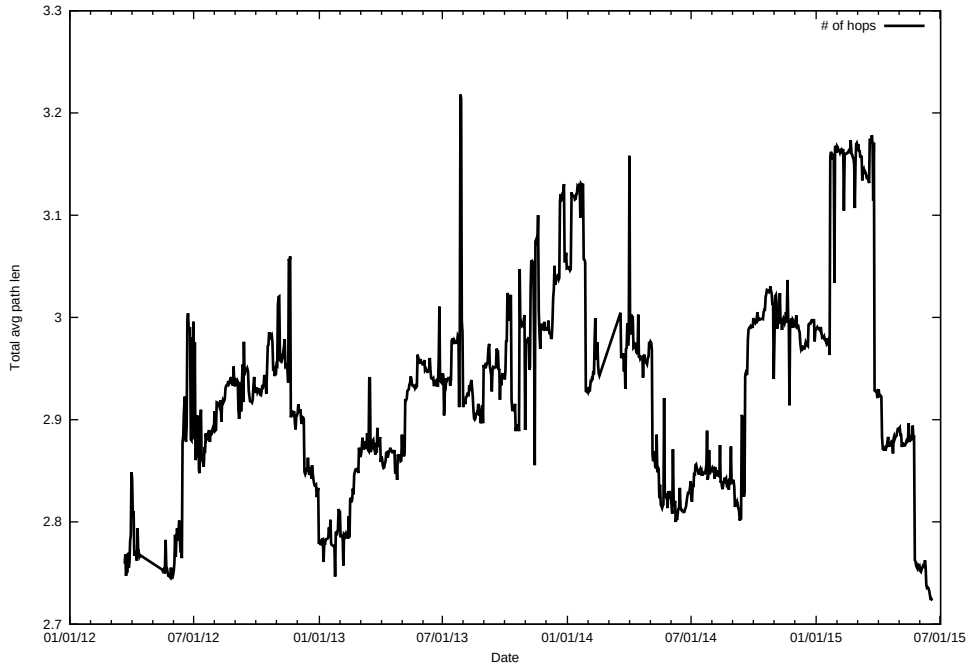
IPv4 BGP path length per prefix length



IPv4 BGP average path length

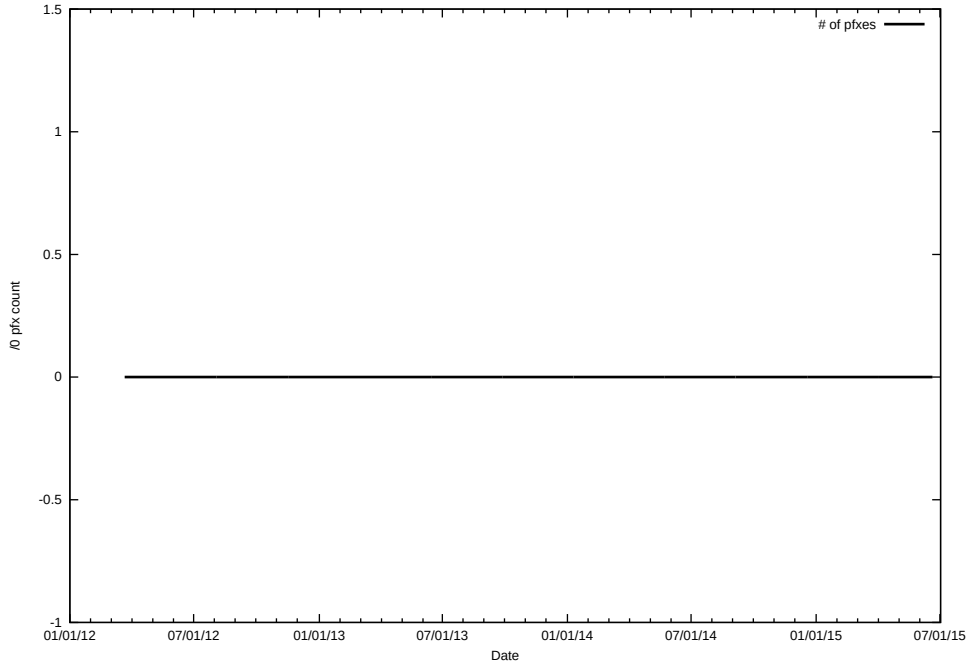


IPv6 BGP path length per prefix length

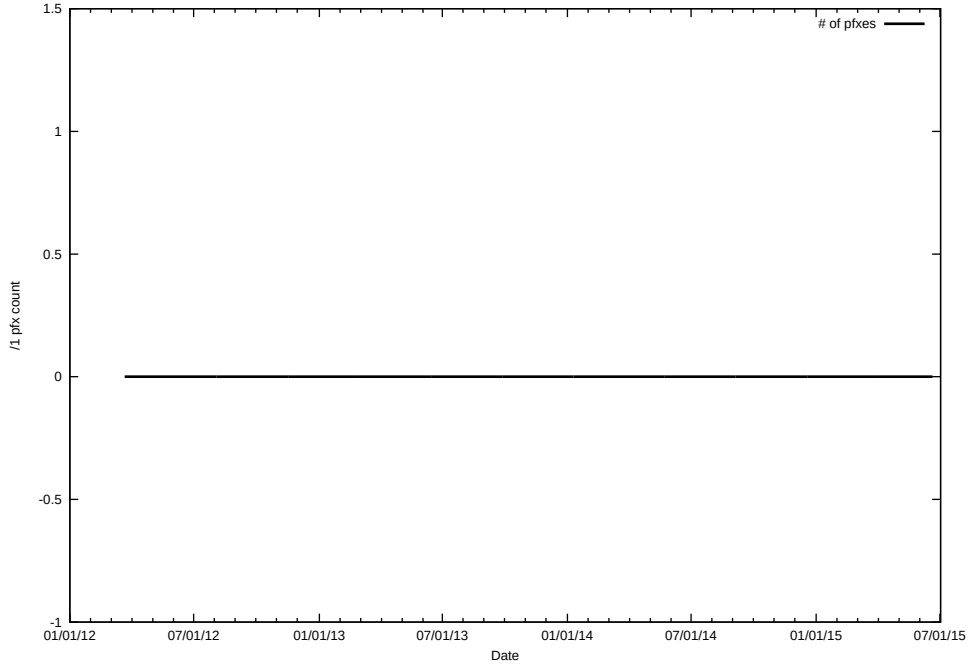


IPv6 BGP average path length

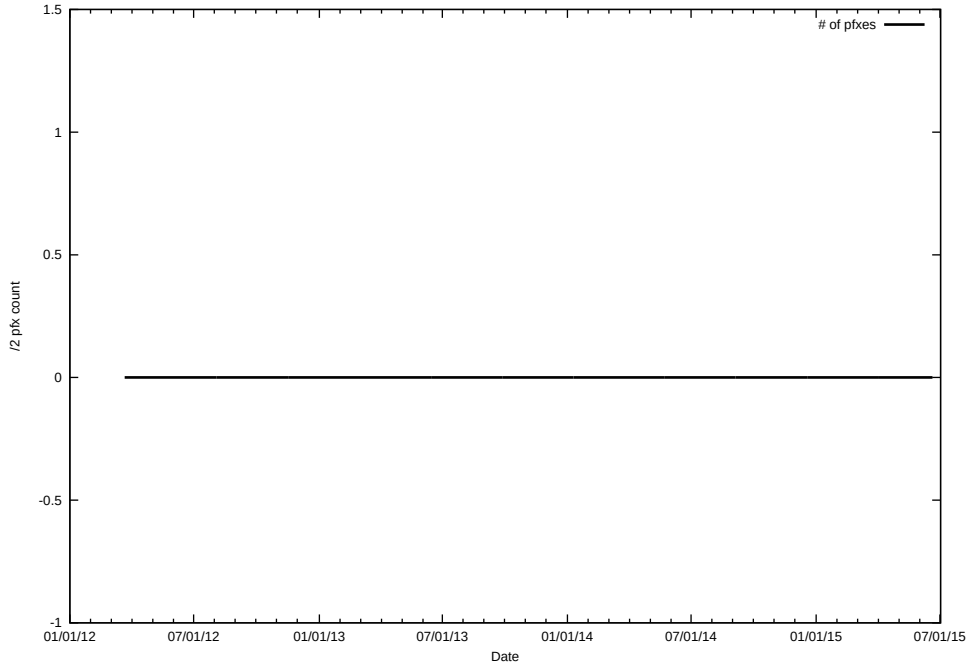
A.2 IPv4 BGP timeline of prefix count per prefix length



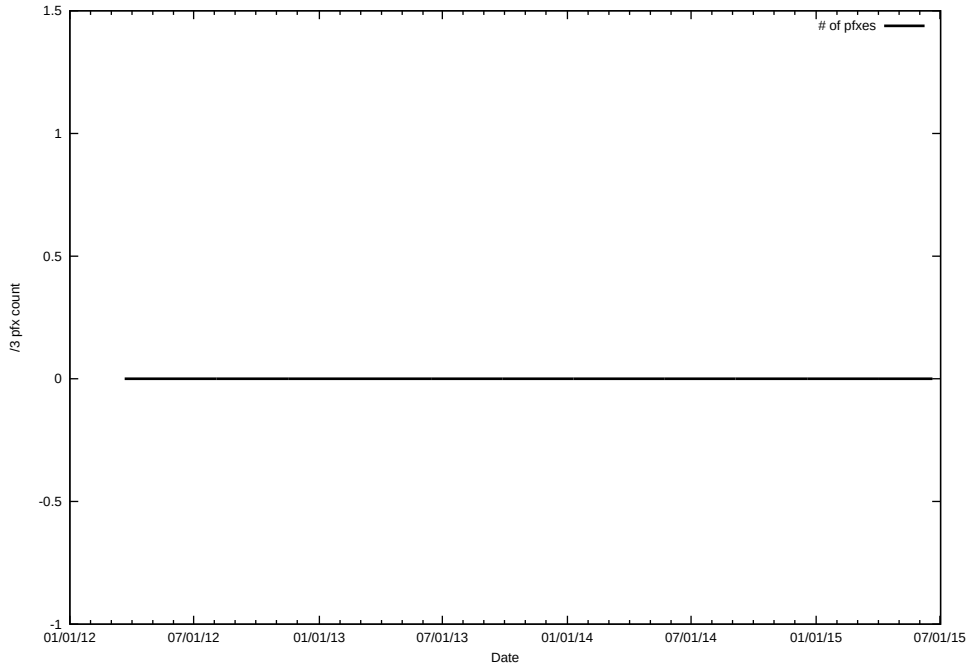
/0



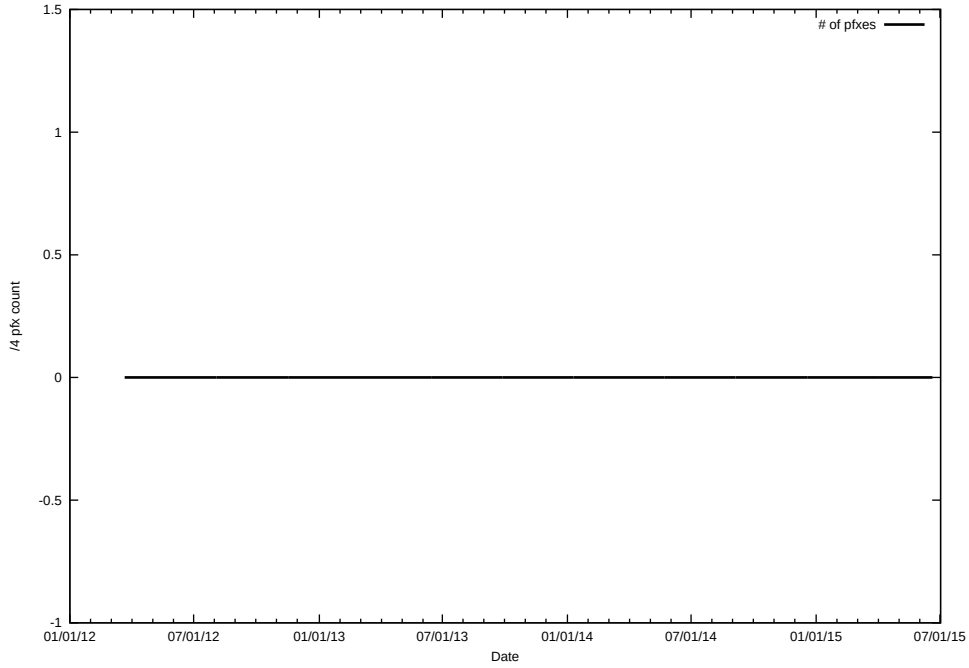
/1


1.5
1
0.5
0
-0.5
-1
/2 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

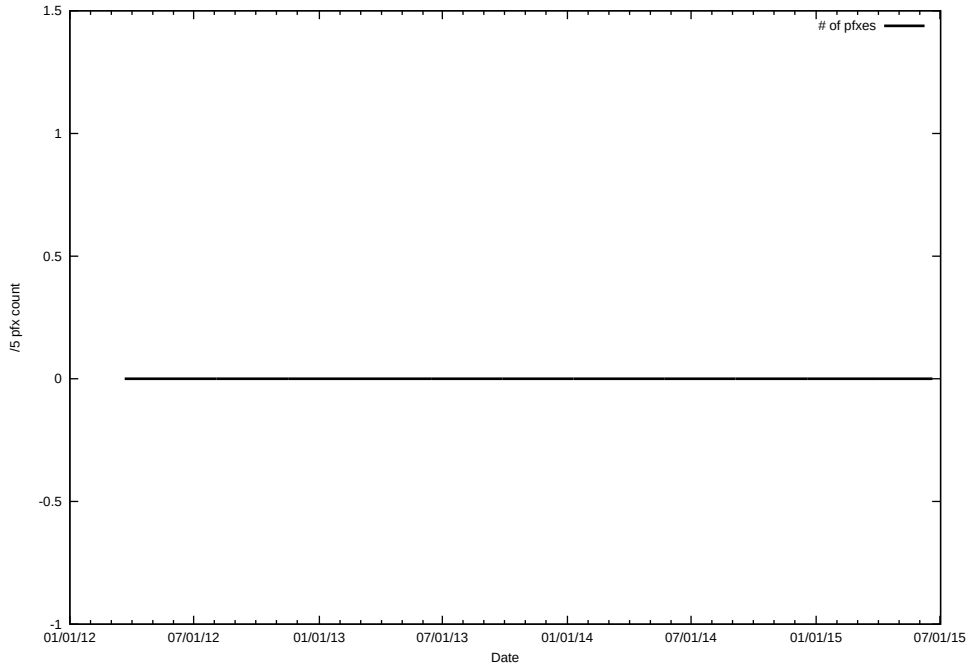
/2


1.5
1
0.5
0
-0.5
-1
/3 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

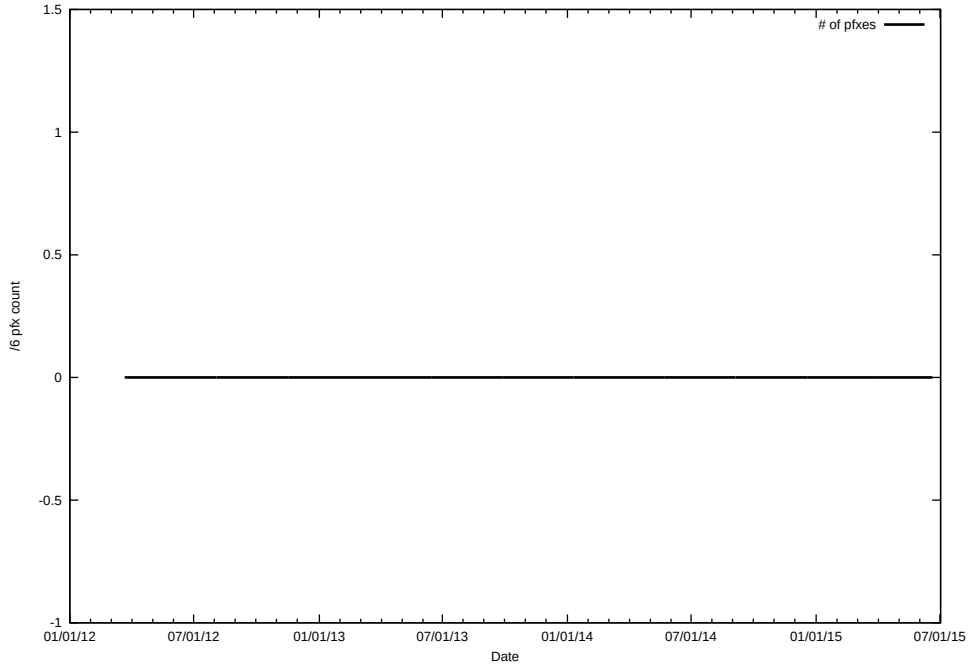
/3


1.5
1
0.5
0
-0.5
-1
/4 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

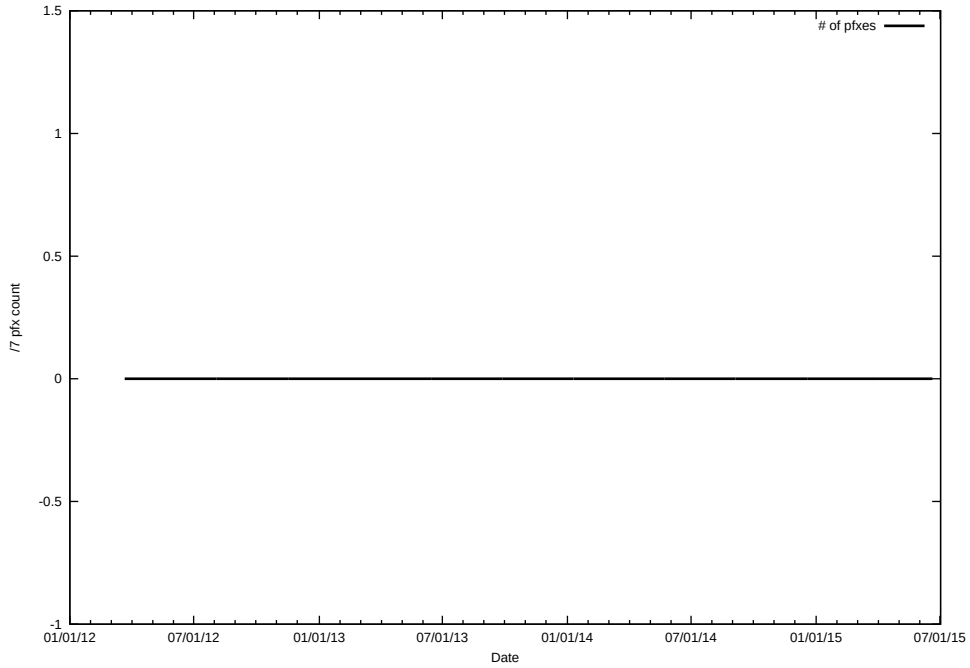
/4


1.5
1
0.5
0
-0.5
-1
/5 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

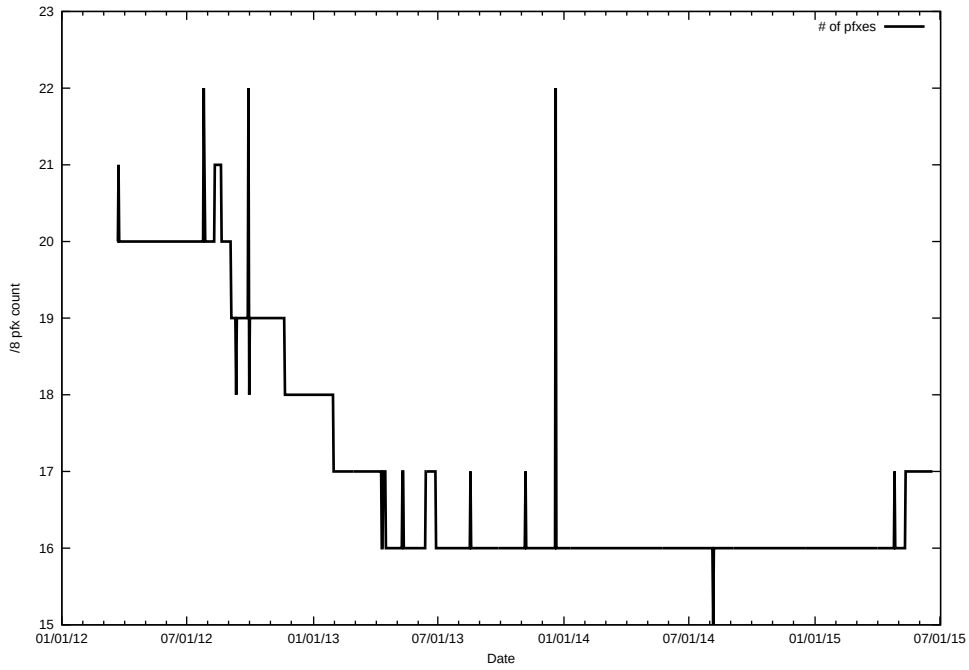
/5


1.5
1
0.5
0
-0.5
-1
/6 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

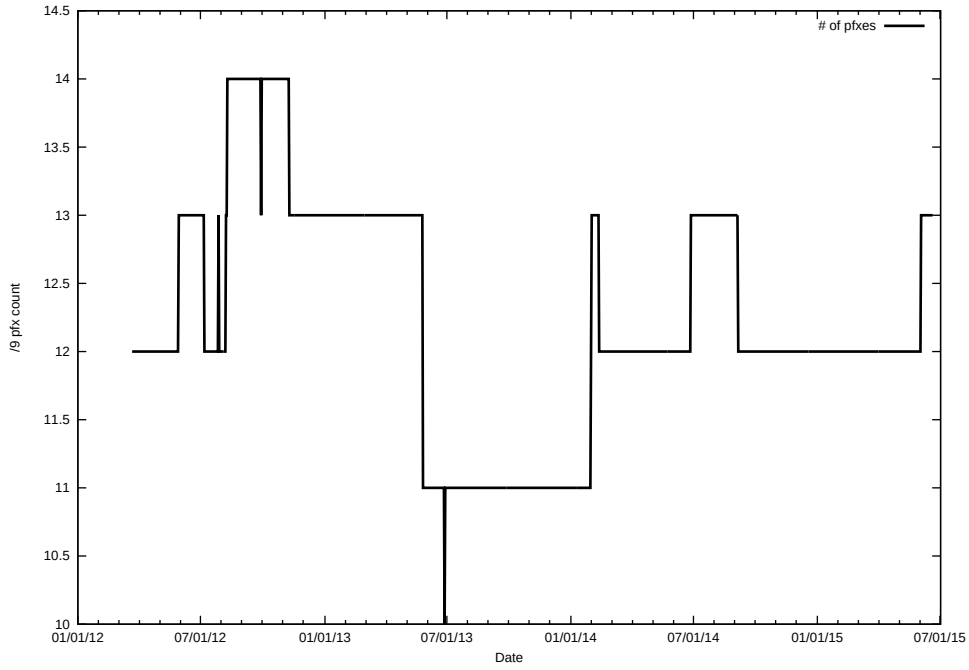
/6


1.5
1
0.5
0
-0.5
-1
/7 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

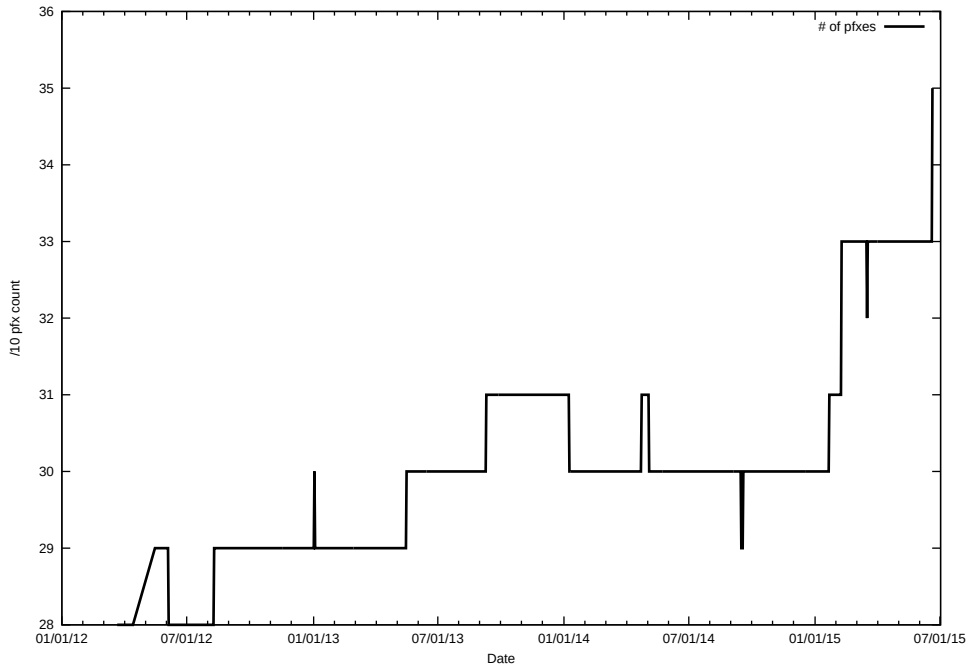
/7


/8 pfx count
Date
# of pfxes
23
22
21
20
19
18
17
16
15
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15

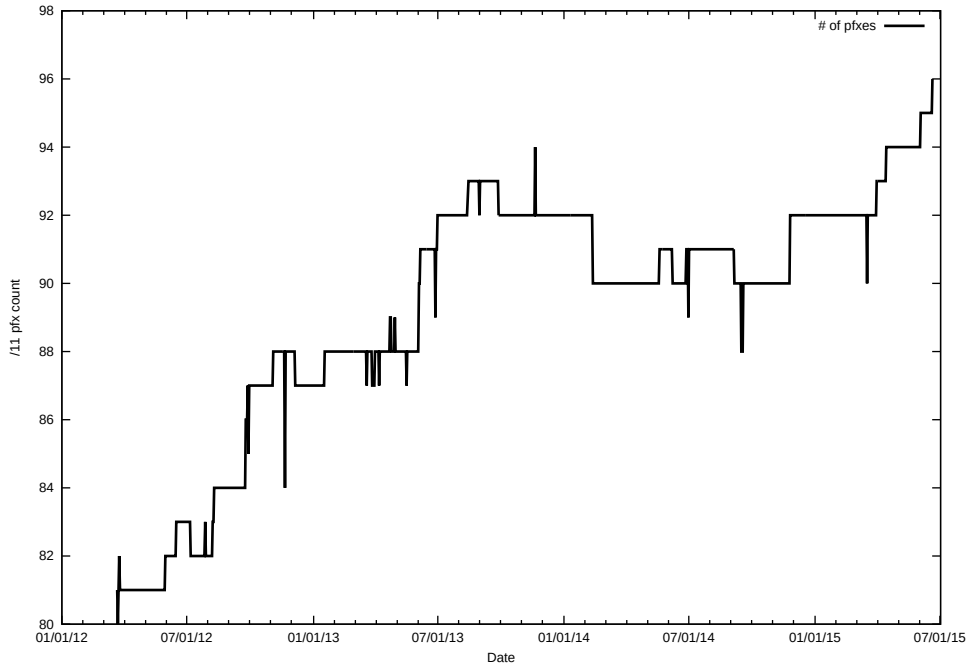
/8


/9 pfx count
Date
# of pfxes
14.5
14
13.5
13
12.5
12
11.5
11
10.5
10
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15

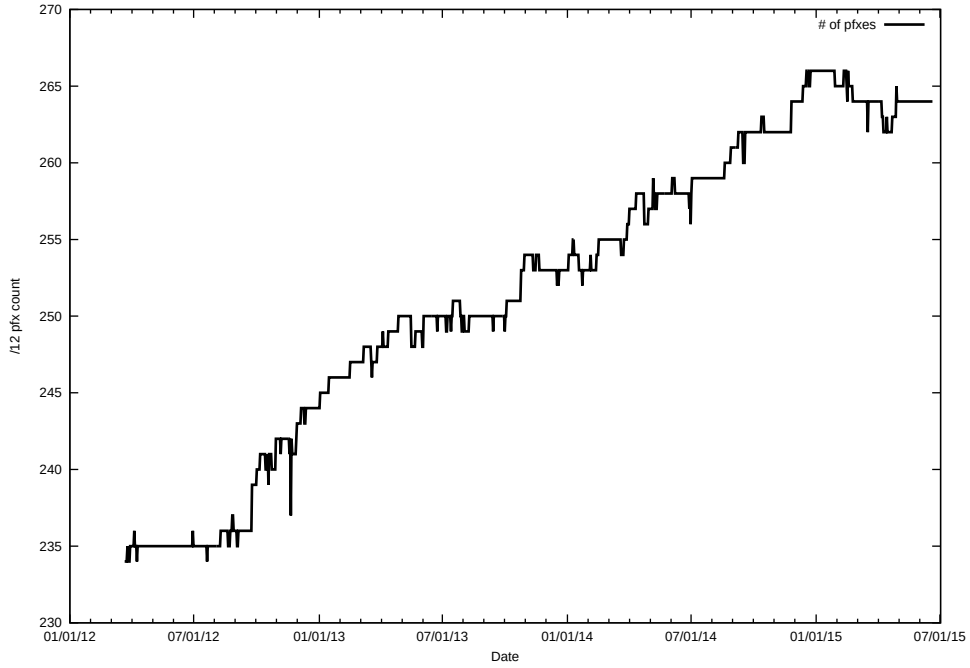
/9


/10 pfx count
Date
# of pfxes
36
35
34
33
32
31
30
29
28
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15

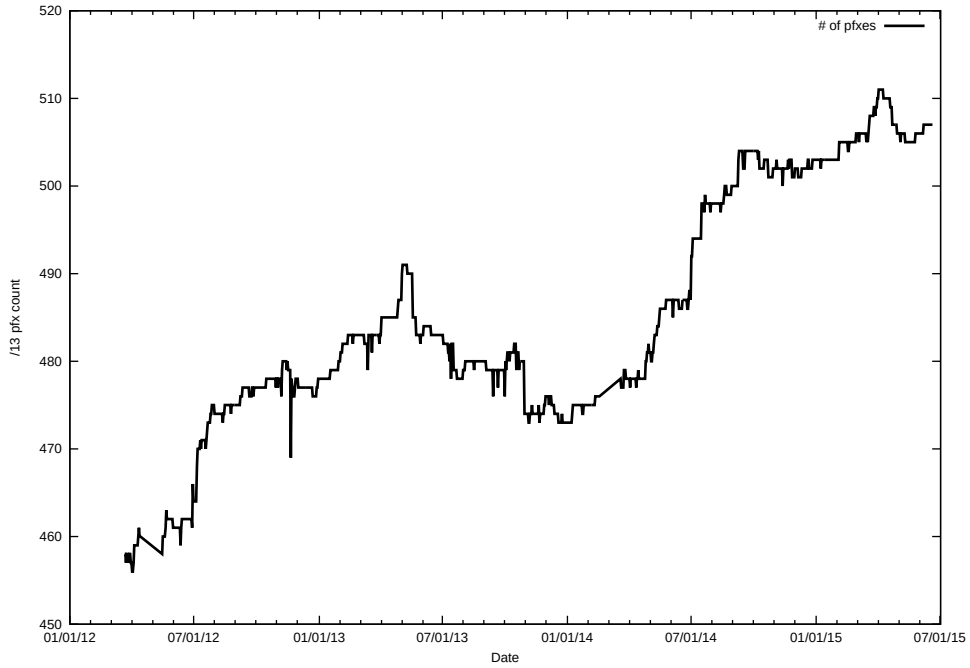
/10


/11 pfx count
Date
# of pfxes
98
96
94
92
90
88
86
84
82
80
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15

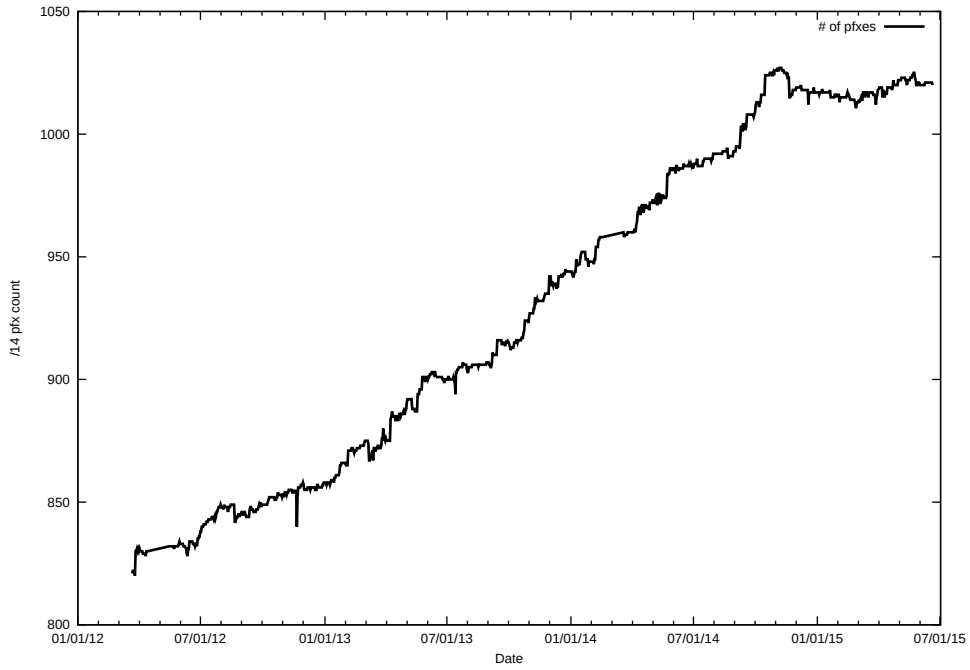
/11


270
265
260
255
250
245
240
235
230
/12 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

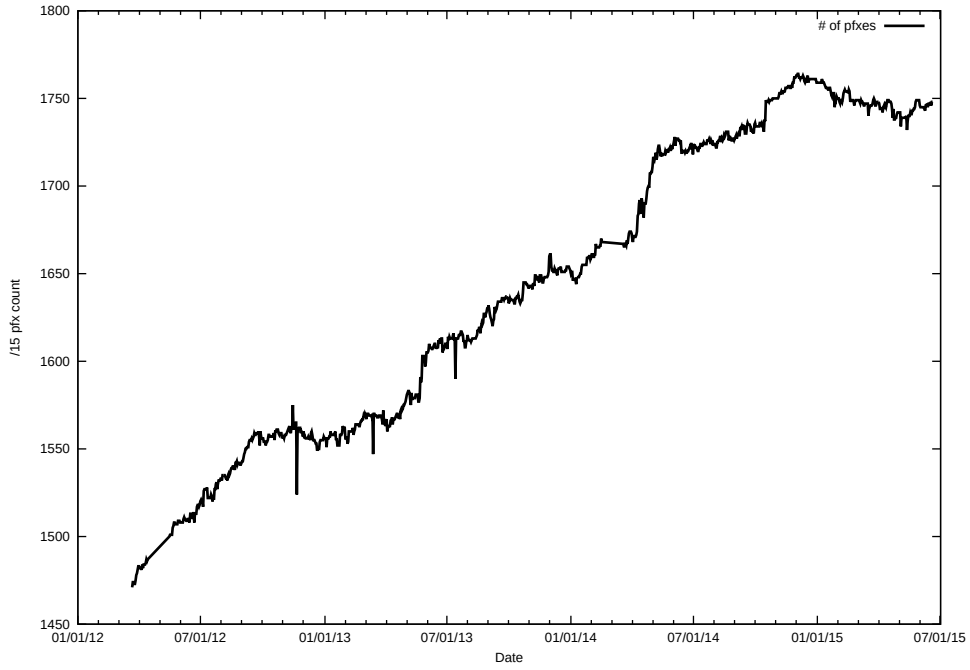
/12


520
510
500
490
480
470
460
450
/13 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

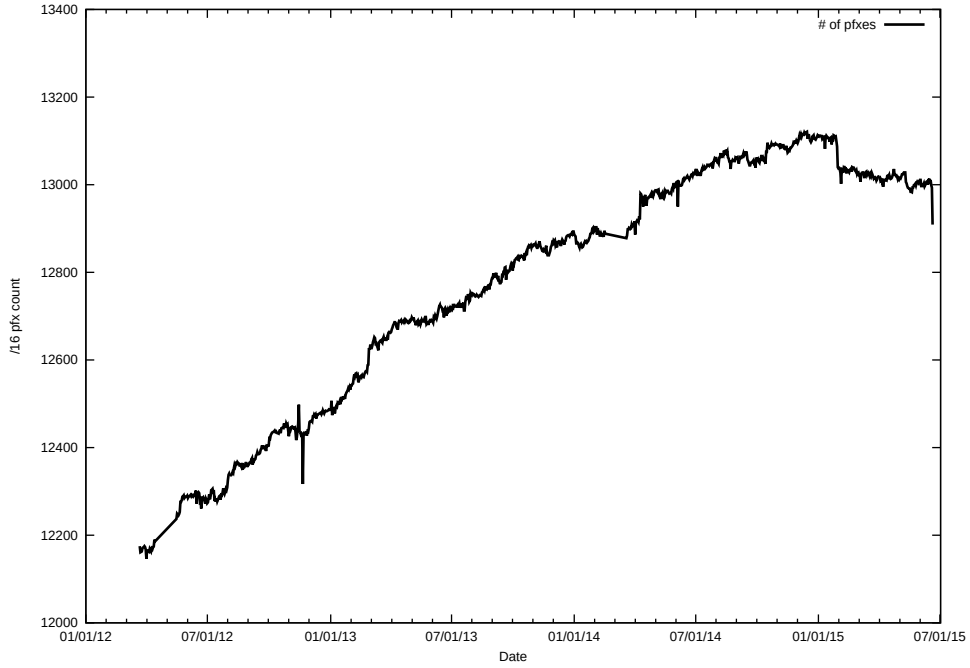
/13


1050
1000
950
900
850
800
/14 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

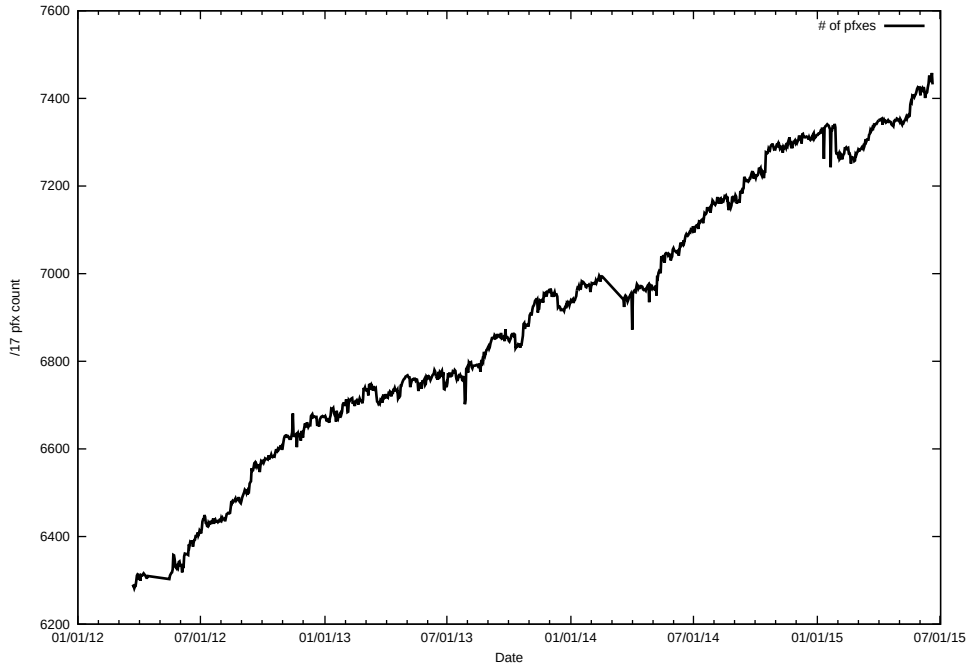
/14


1800
1750
1700
1650
1600
1550
1500
1450
/15 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

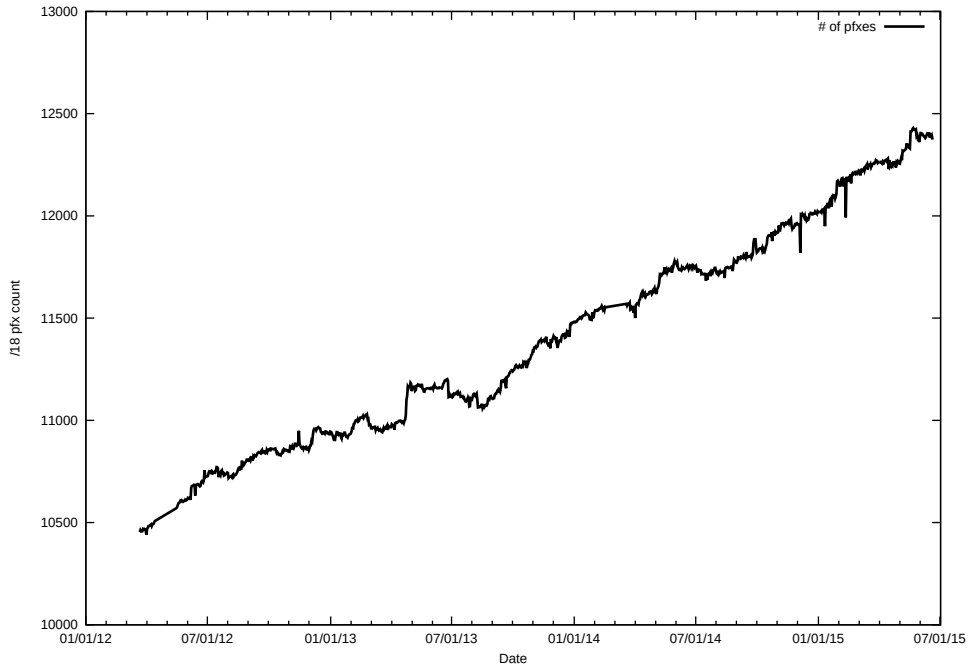
/15


13400
13200
13000
12800
12600
12400
12200
12000
/16 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

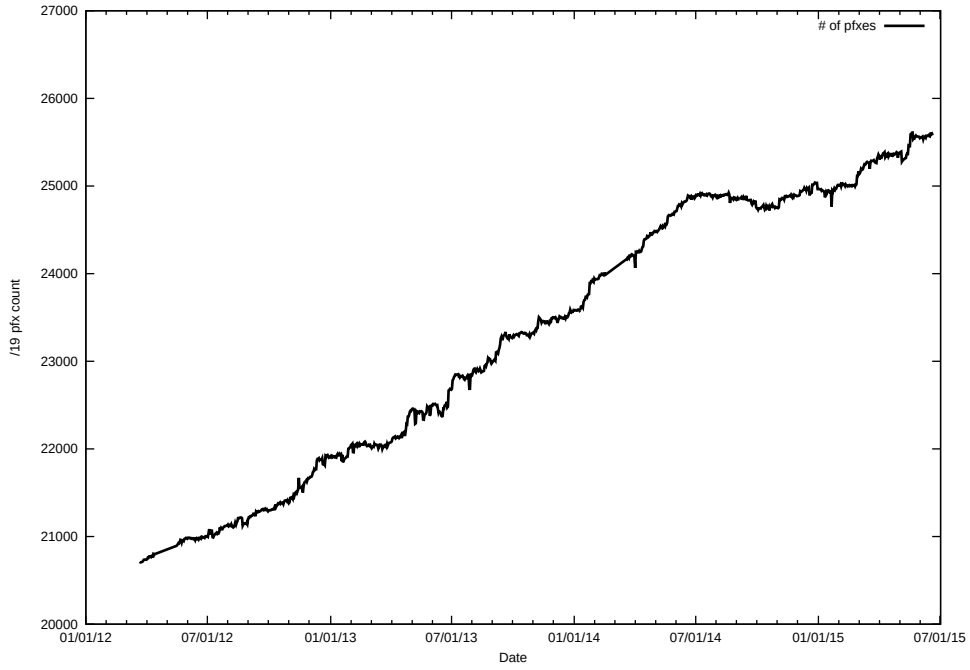
/16


7600
7400
7200
7000
6800
6600
6400
6200
/17 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

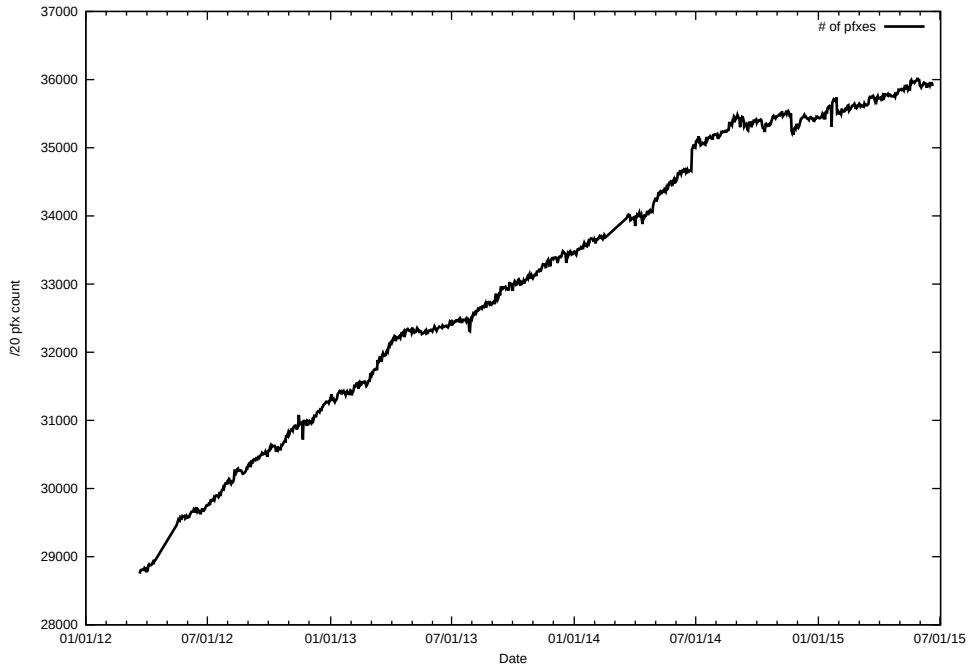
/17


13000
12500
12000
11500
11000
10500
10000
/18 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

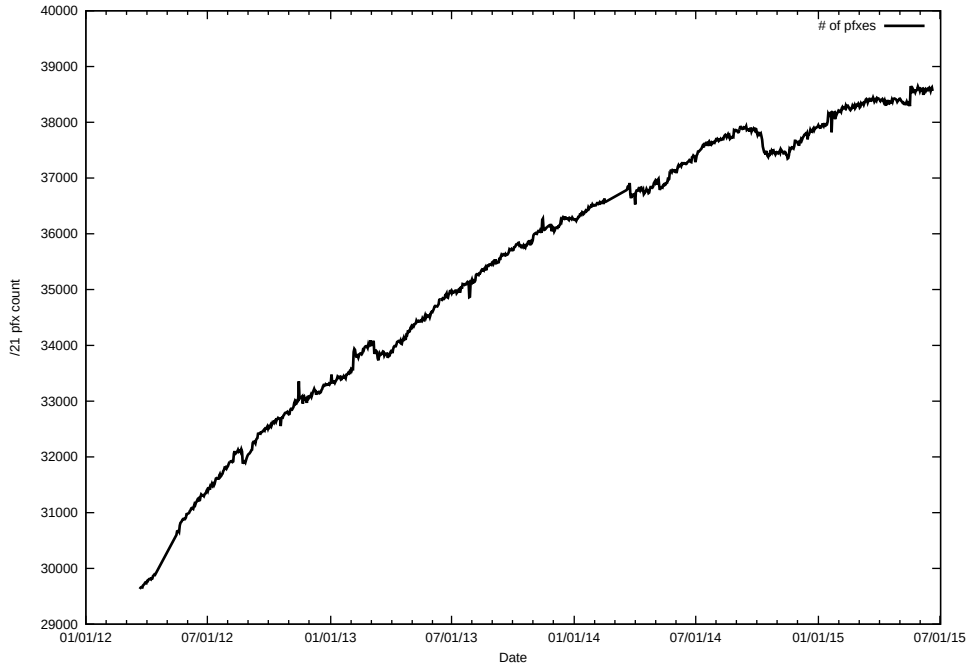
/18


27000
26000
25000
24000
23000
22000
21000
20000
/19 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

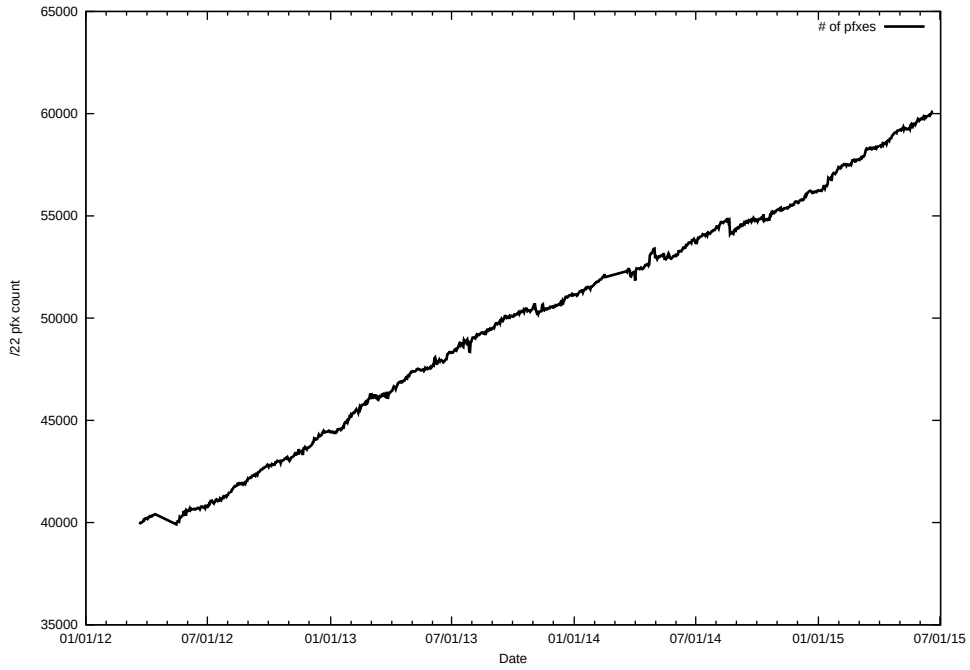
/19


37000
36000
35000
34000
33000
32000
31000
30000
29000
28000
/20 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

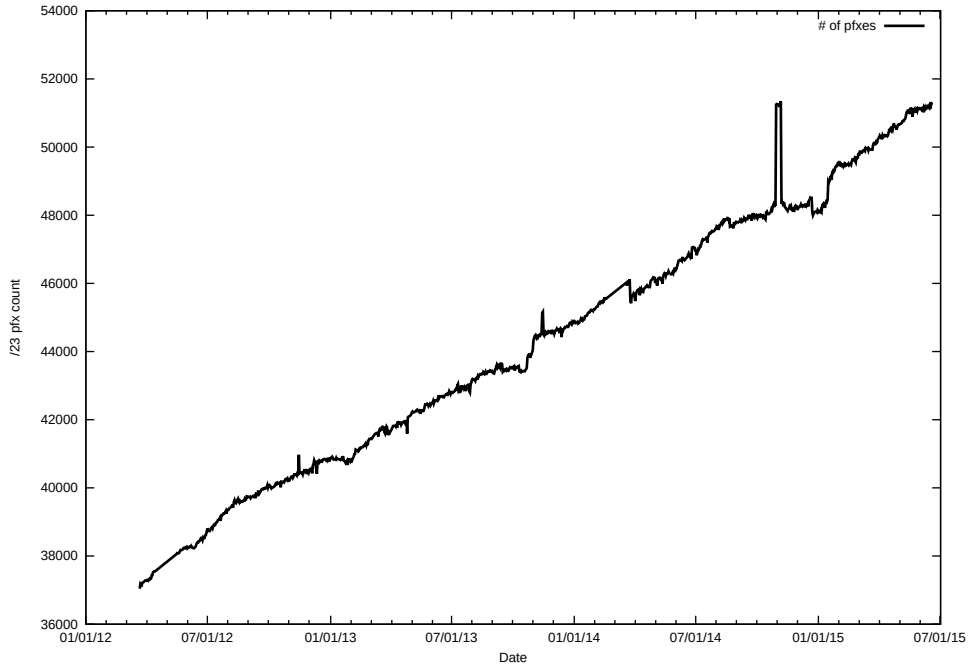
/20


40000
39000
38000
37000
36000
35000
34000
33000
32000
31000
30000
29000
/21 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

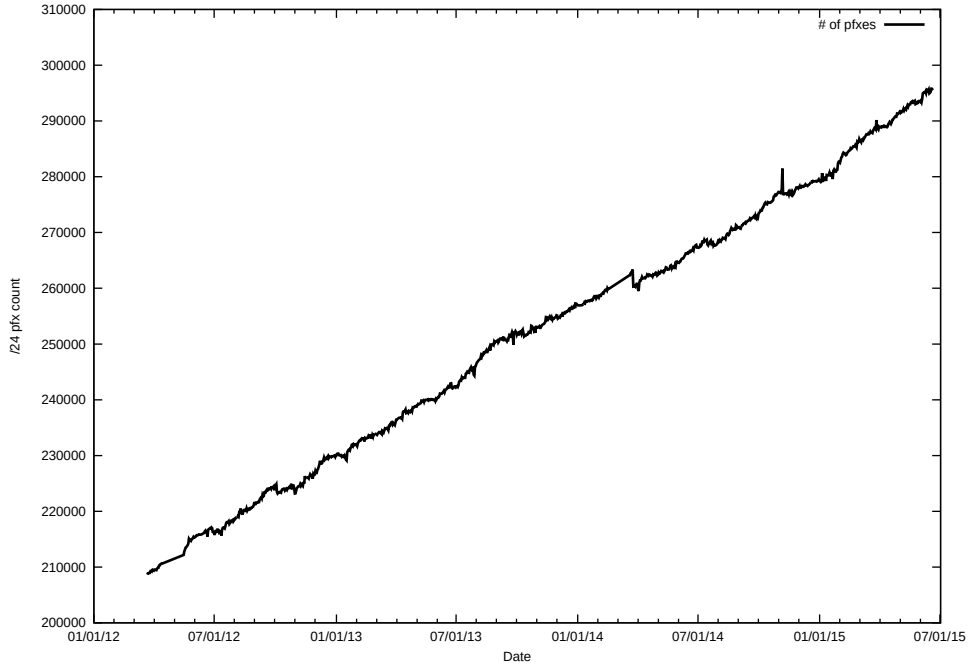
/21


65000
60000
55000
50000
45000
40000
35000
/22 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

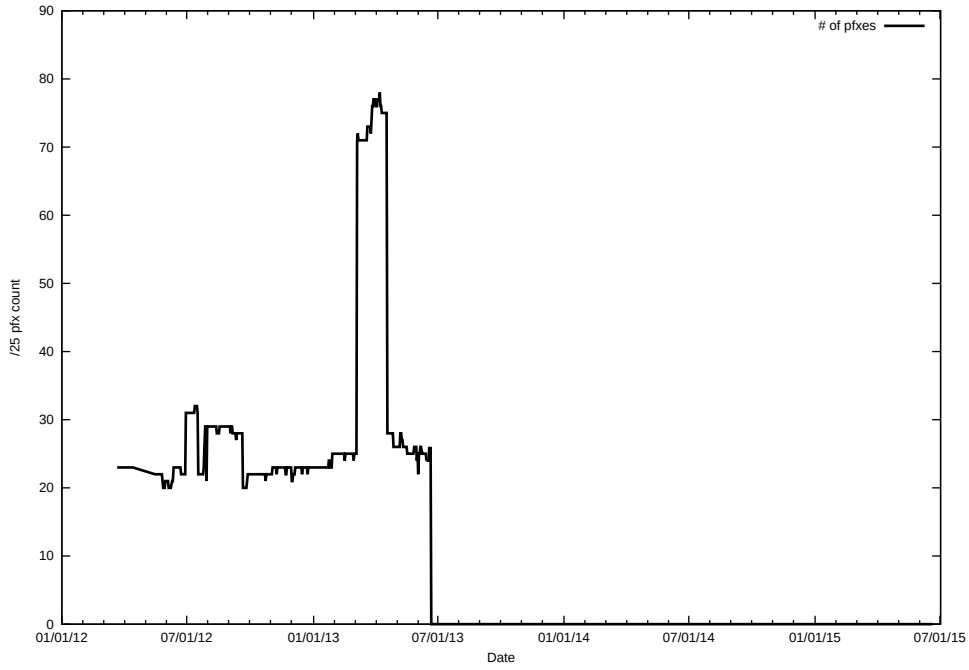
/22


54000
52000
50000
48000
46000
44000
42000
40000
38000
36000
/23 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

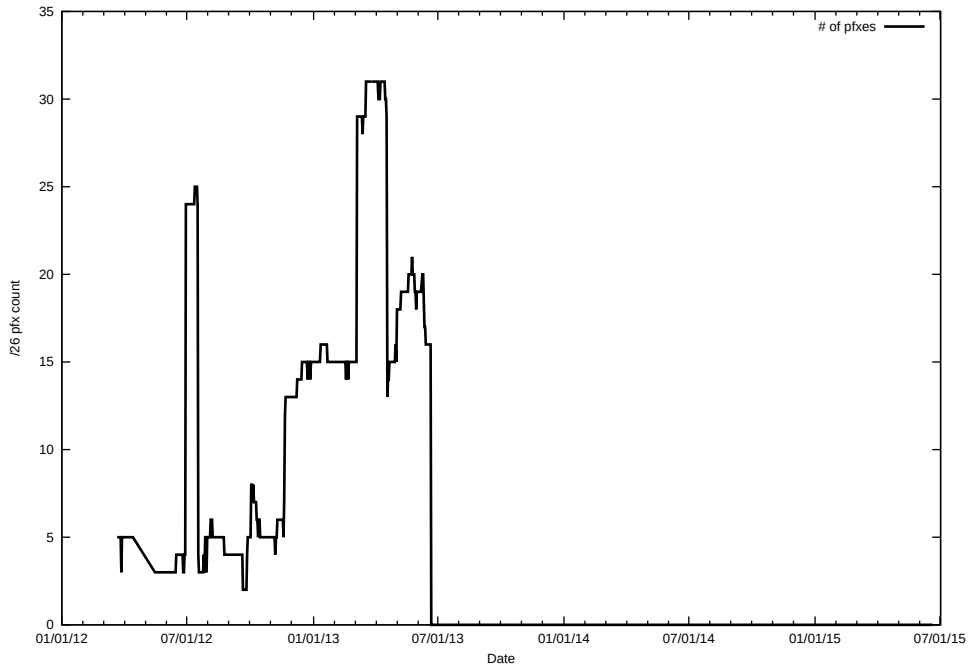
/23


310000
300000
290000
280000
270000
260000
250000
240000
230000
220000
210000
200000
/24 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

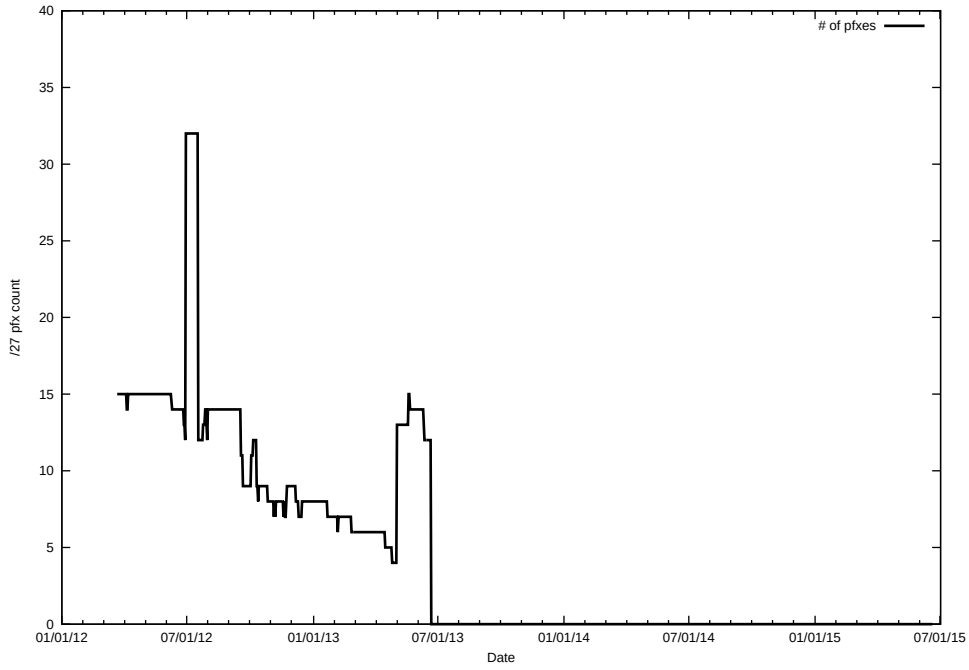
/24


90
80
70
60
50
40
30
20
10
0
/25 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

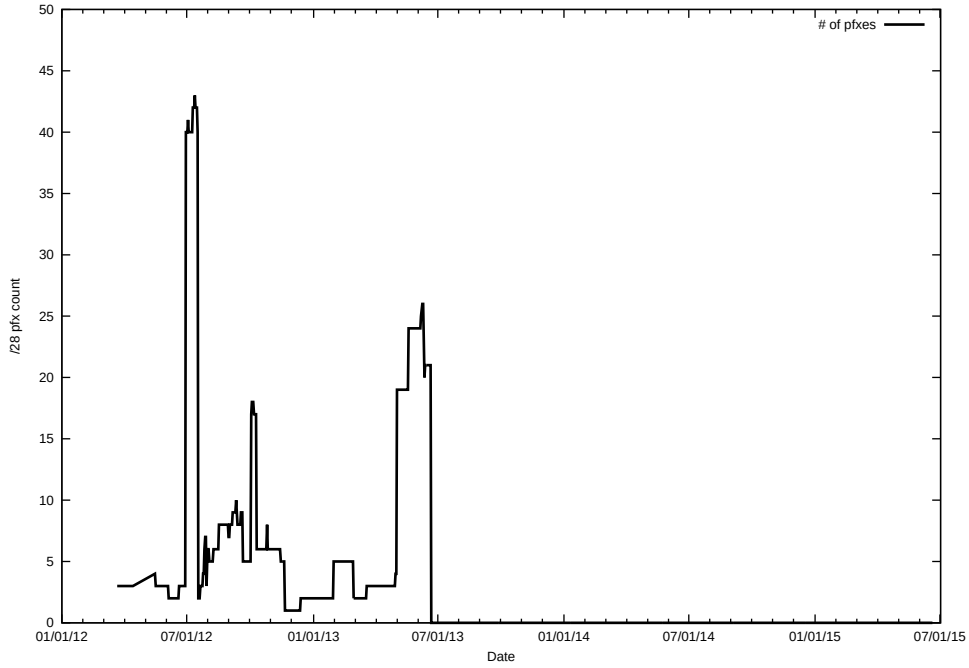
/25


35
30
25
20
15
10
5
0
/26 pfx count
# of pfxes
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

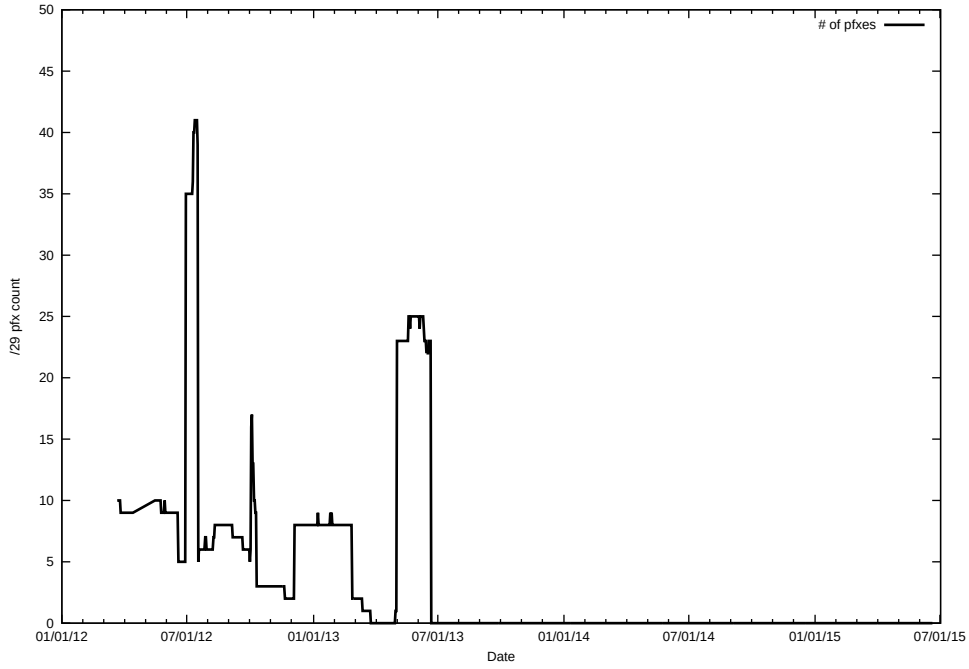
/26


40
35
30
25
20
15
10
5
0
/27 pfx count
# of pfxes
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

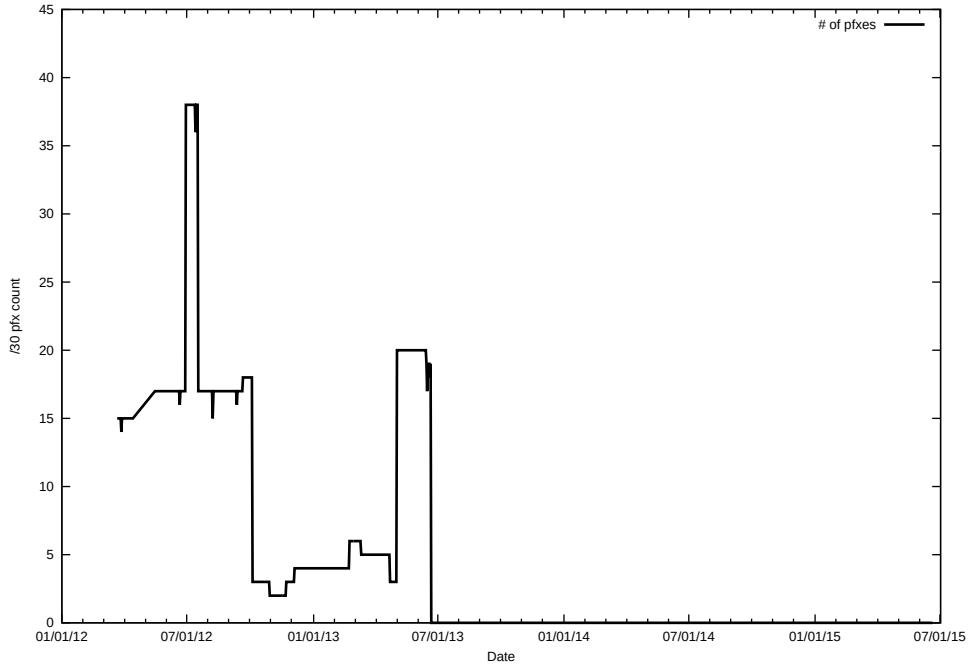
/27


/28 pfx count
Date
# of pfxes
0
5
10
15
20
25
30
35
40
45
50
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15

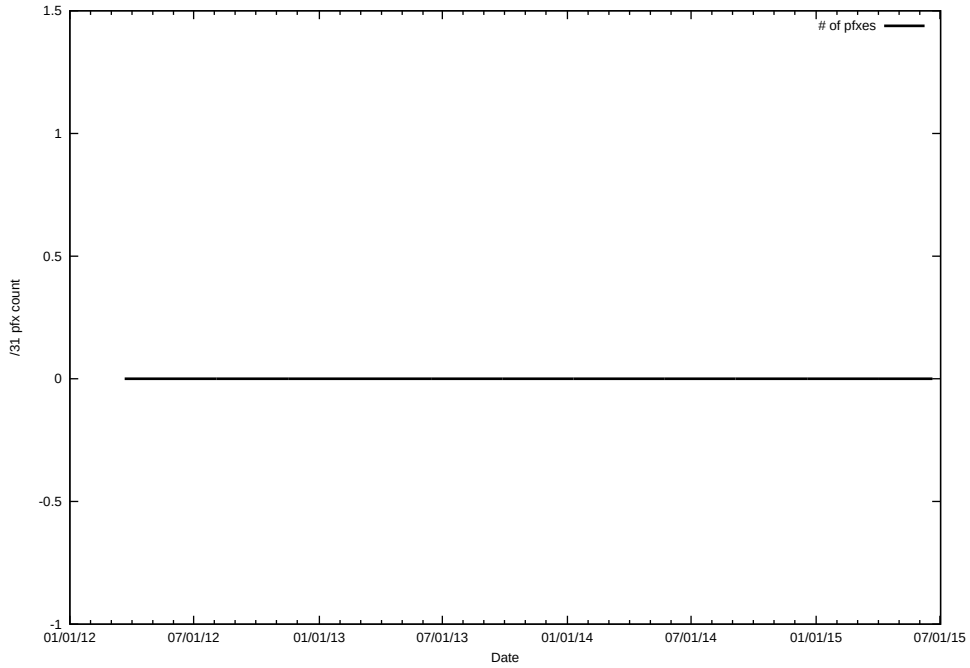
/28


/29 pfx count
Date
# of pfxes
0
5
10
15
20
25
30
35
40
45
50
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15

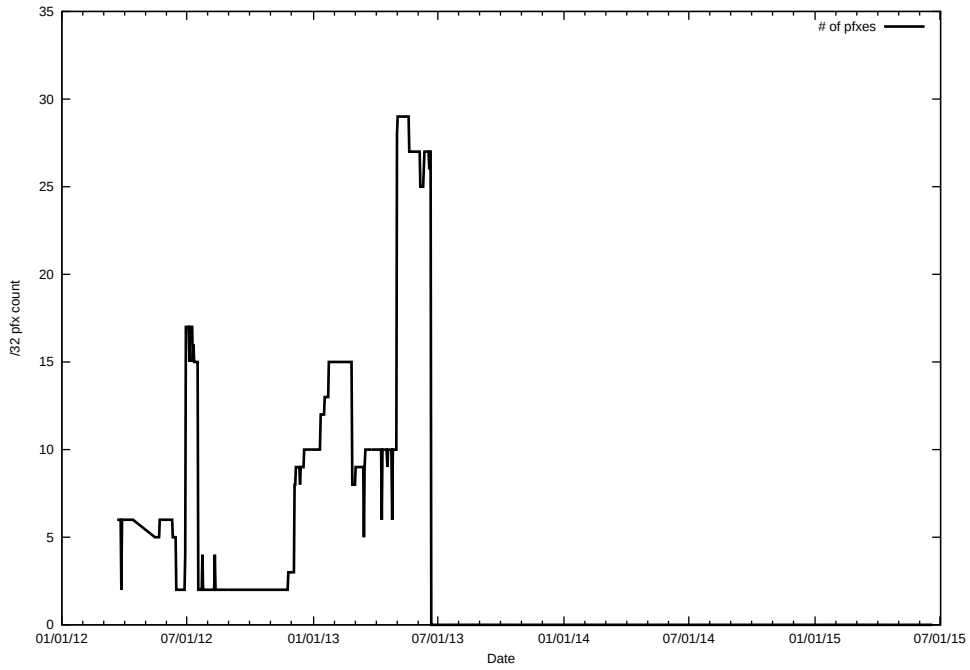
/29


45
40
35
30
25
20
15
10
5
0
/30 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

/30

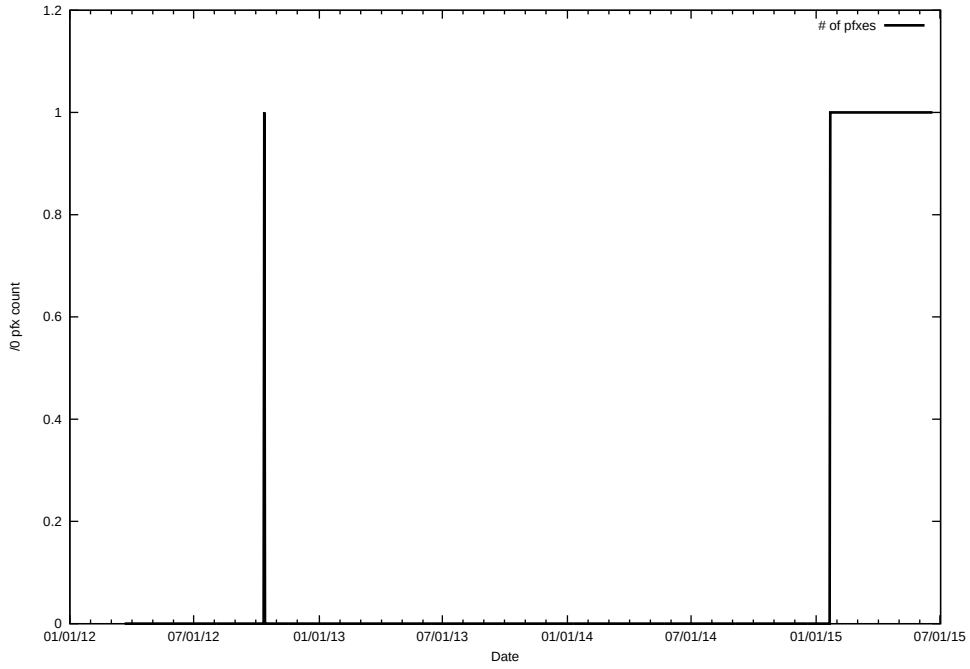

1.5
1
0.5
0
-0.5
-1
/31 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

/31

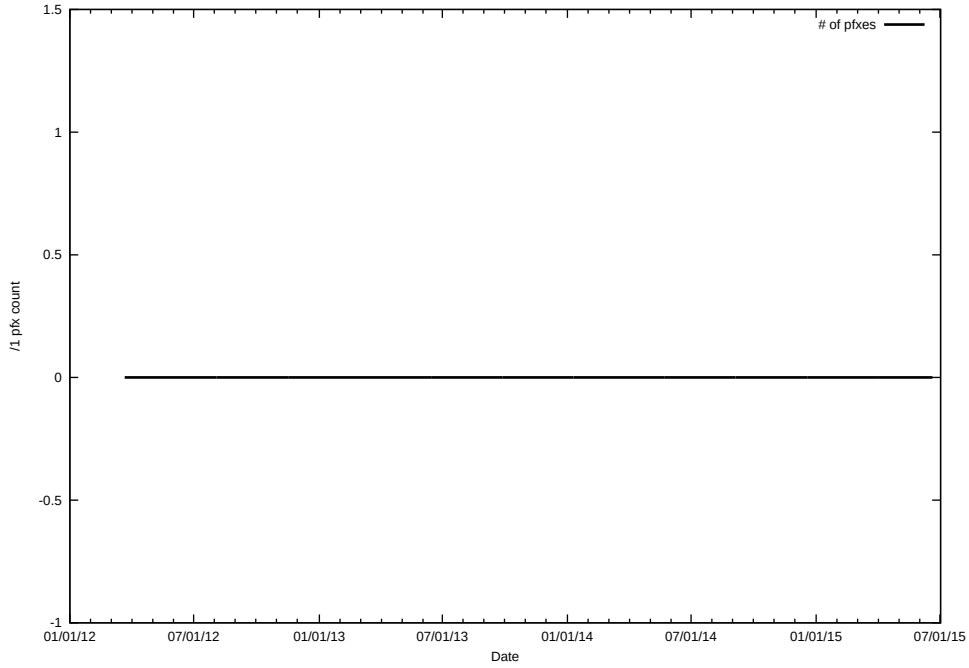


/32

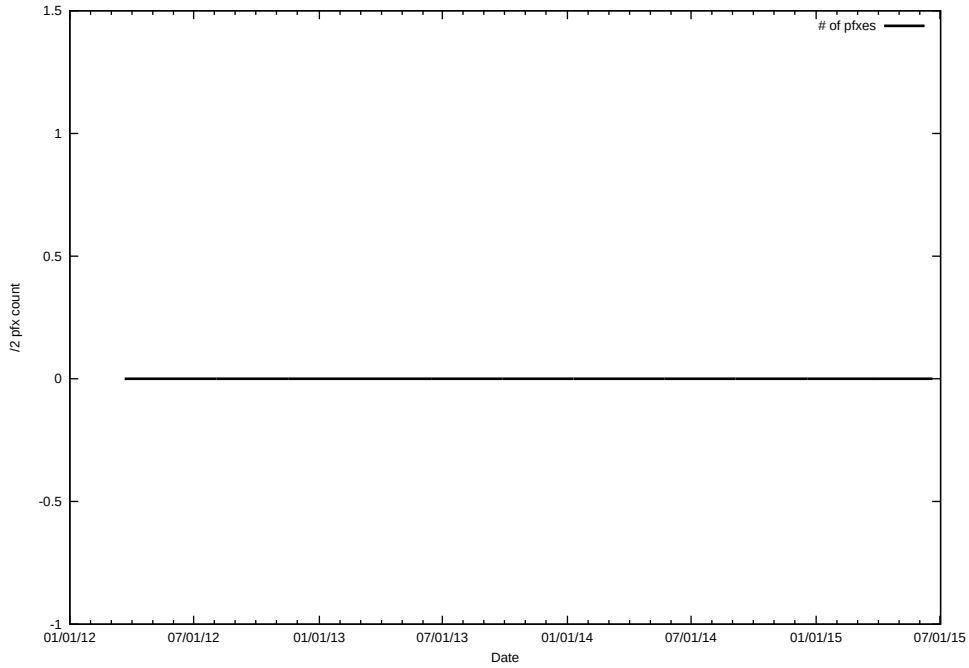
A.3 IPv6 BGP timeline of prefix count per prefix length



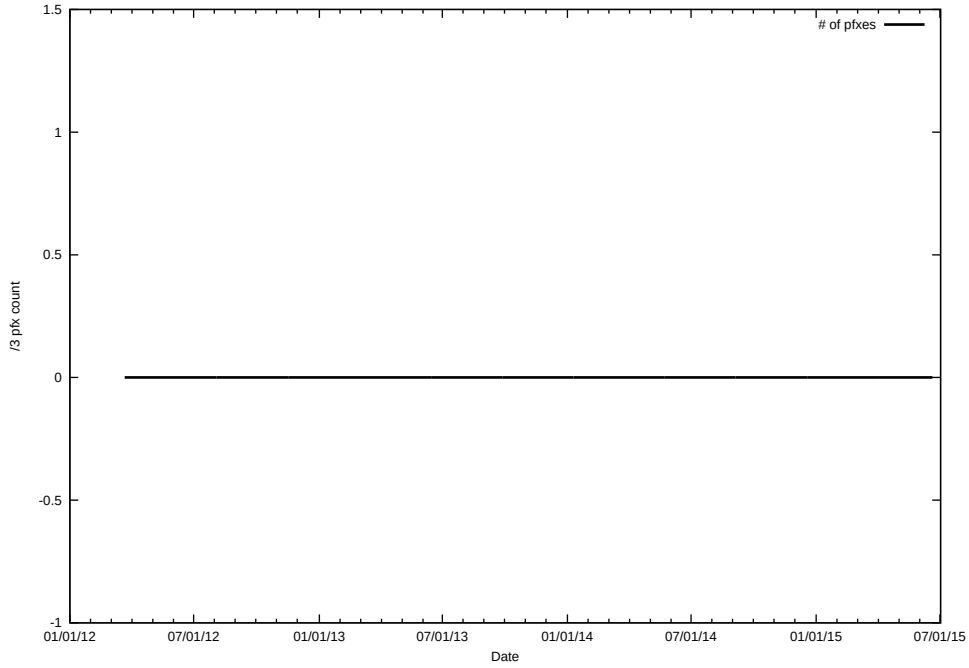
/0


1.5
1
0.5
0
-0.5
-1
/1 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

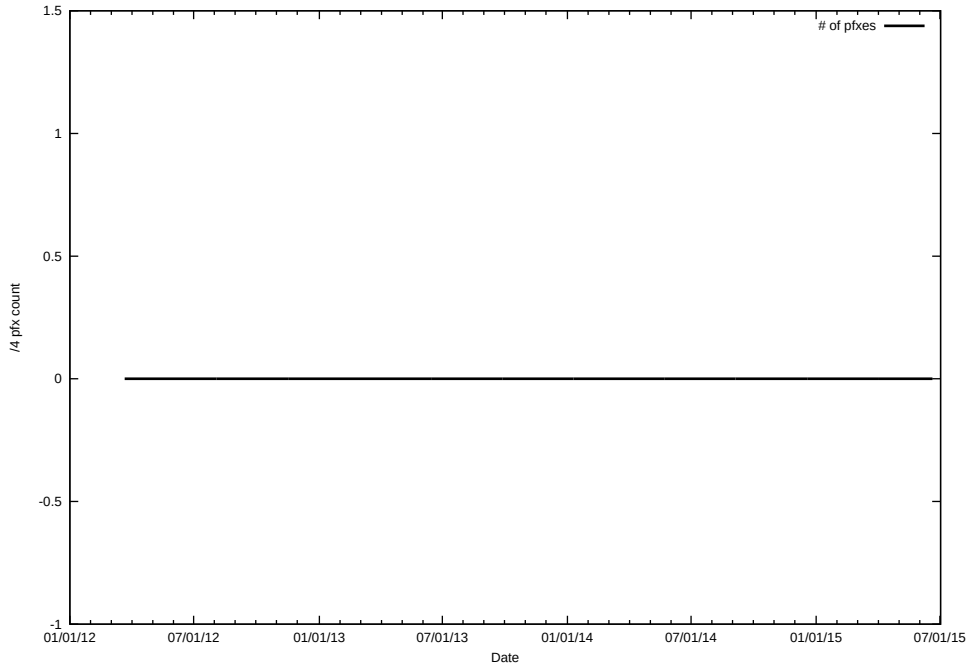
/1


1.5
1
0.5
0
-0.5
-1
/2 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

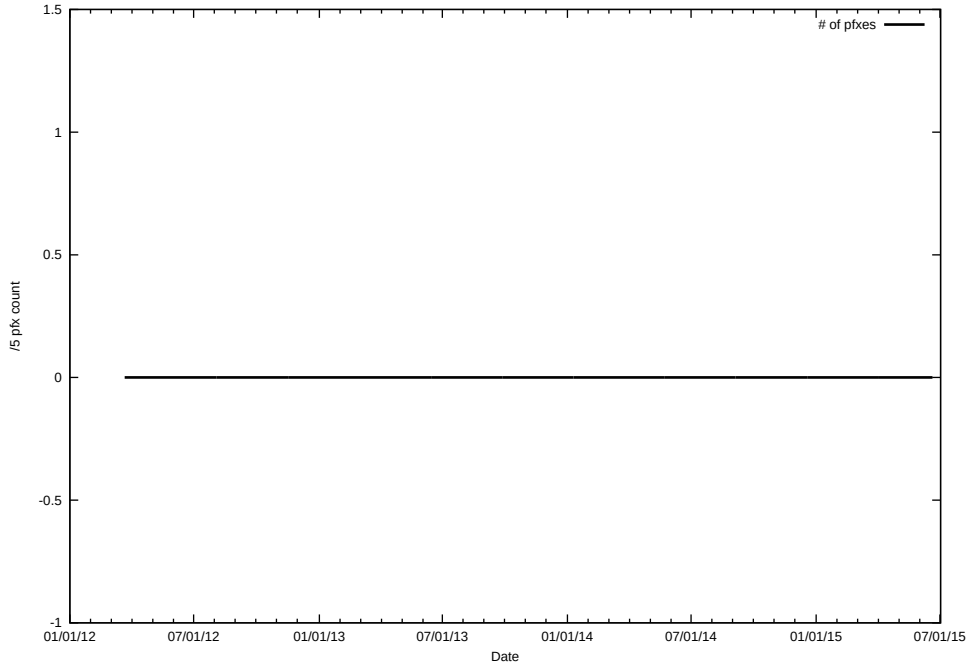
/2


1.5
1
0.5
0
-0.5
-1
/3 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

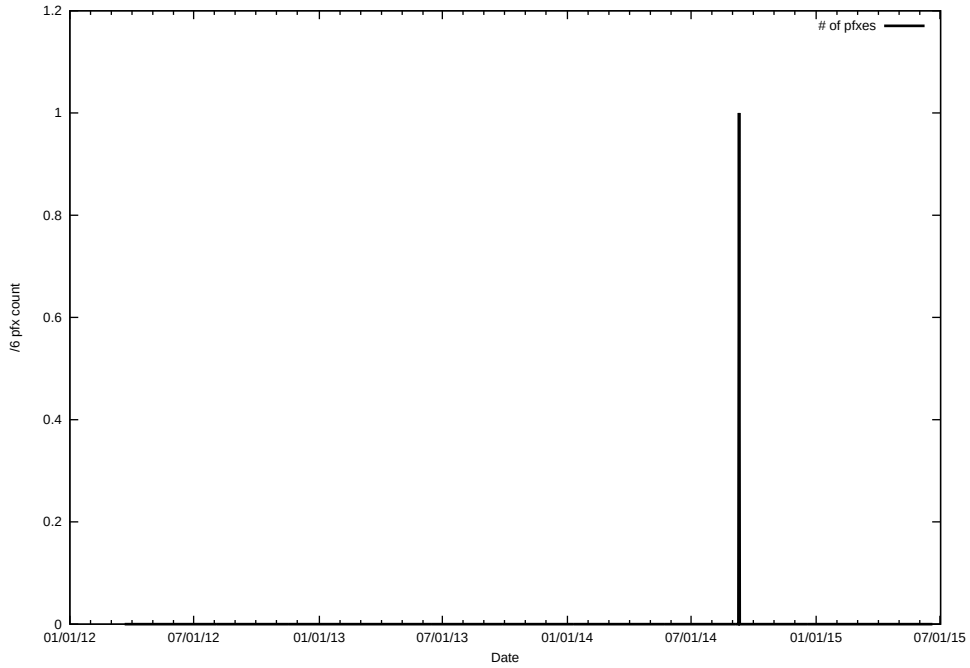
/3


1.5
1
0.5
0
-0.5
-1
/4 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

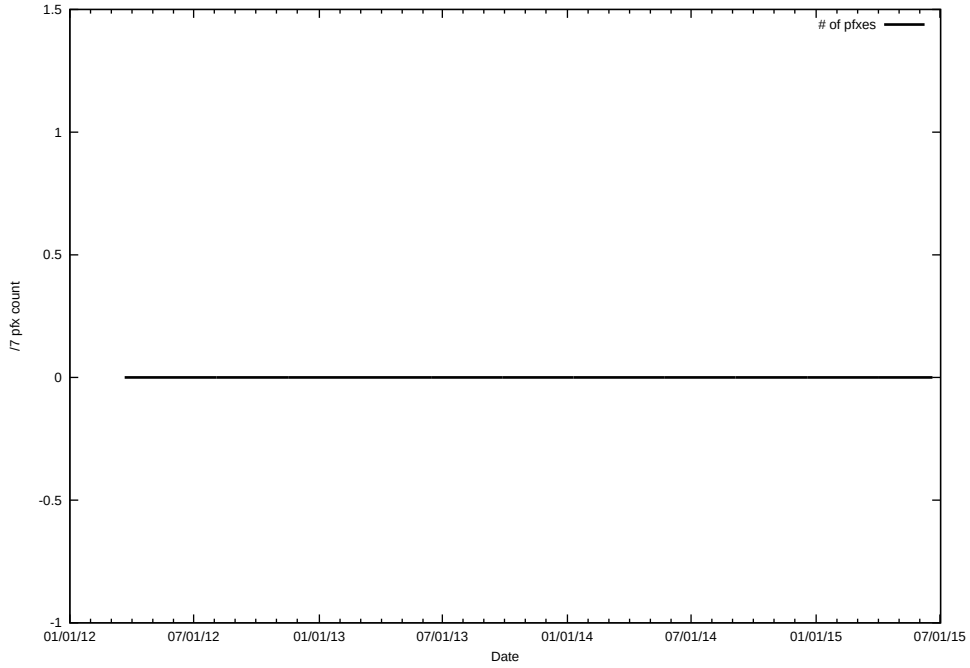
/4


1.5
1
0.5
0
-0.5
-1
/5 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

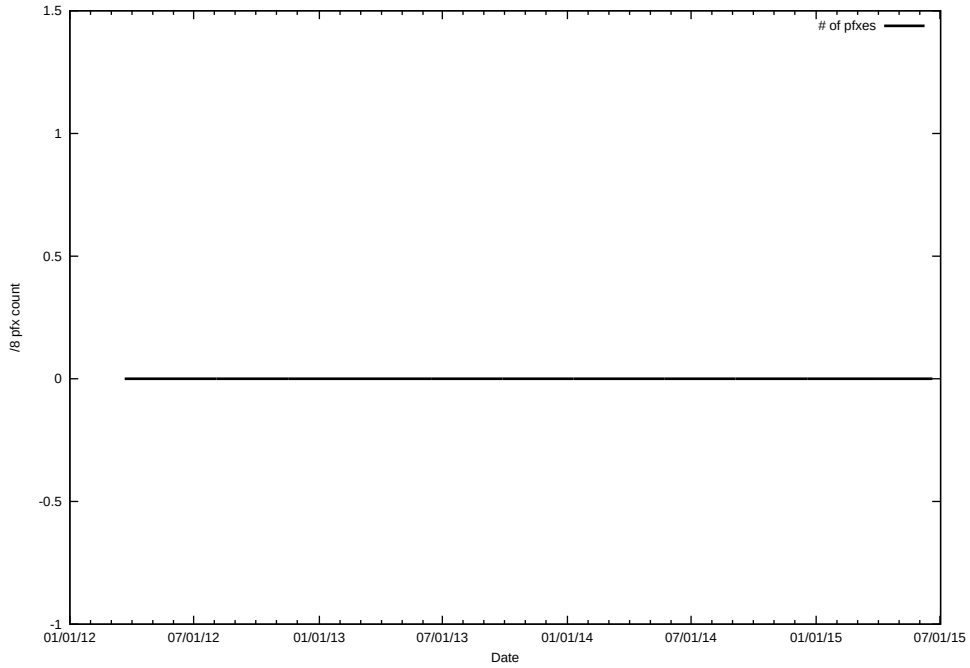
/5


1.2
1
0.8
0.6
0.4
0.2
0
/6 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

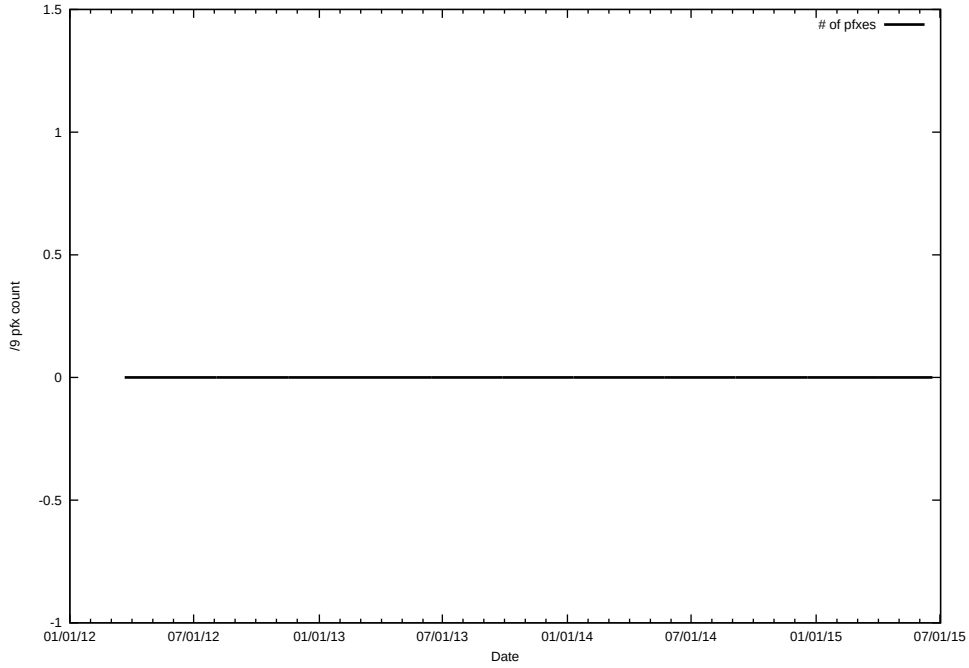
/6


1.5
1
0.5
0
-0.5
-1
/7 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

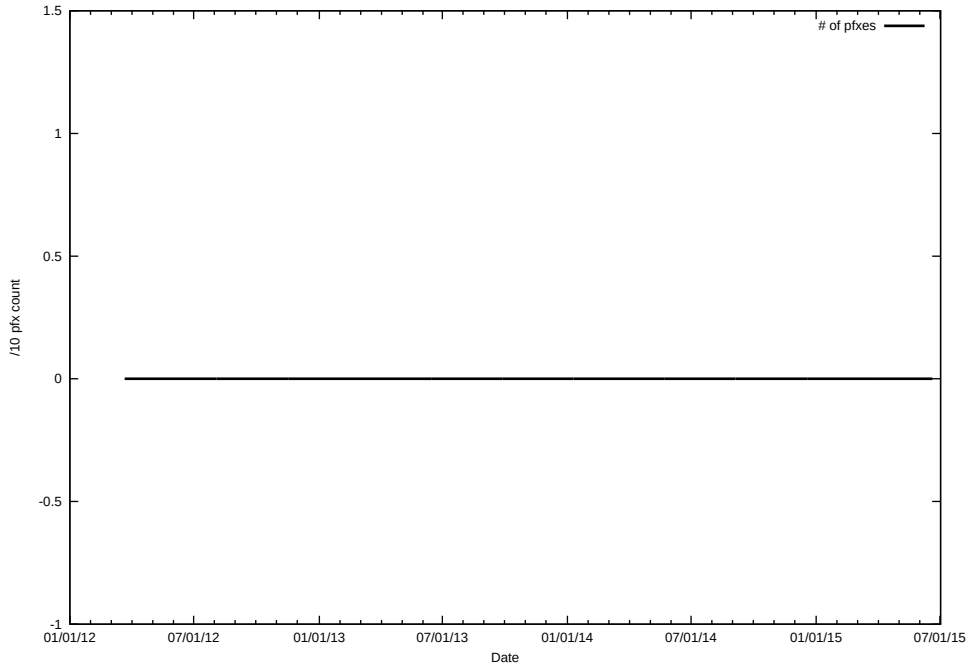
/7


1.5
1
0.5
0
-0.5
-1
/8 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

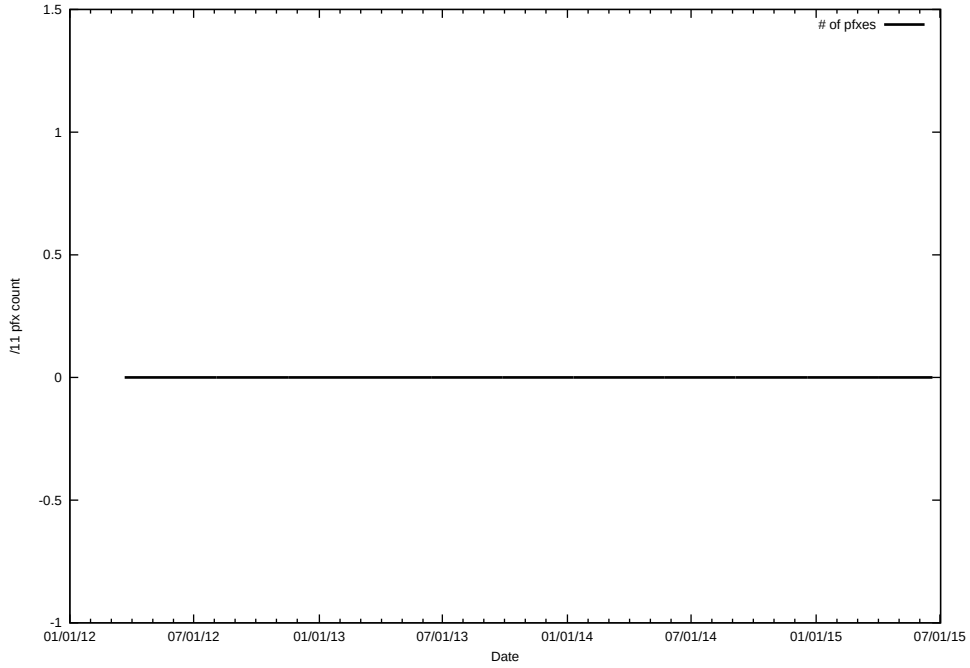
/8


1.5
1
0.5
0
-0.5
-1
/9 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

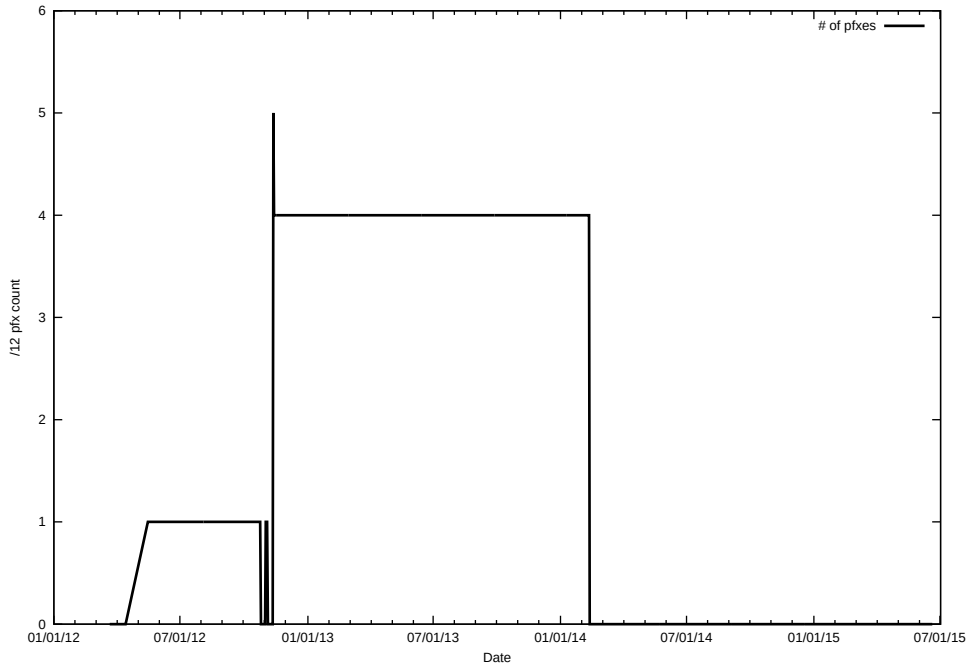
/9


1.5
1
0.5
0
-0.5
-1
/10 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

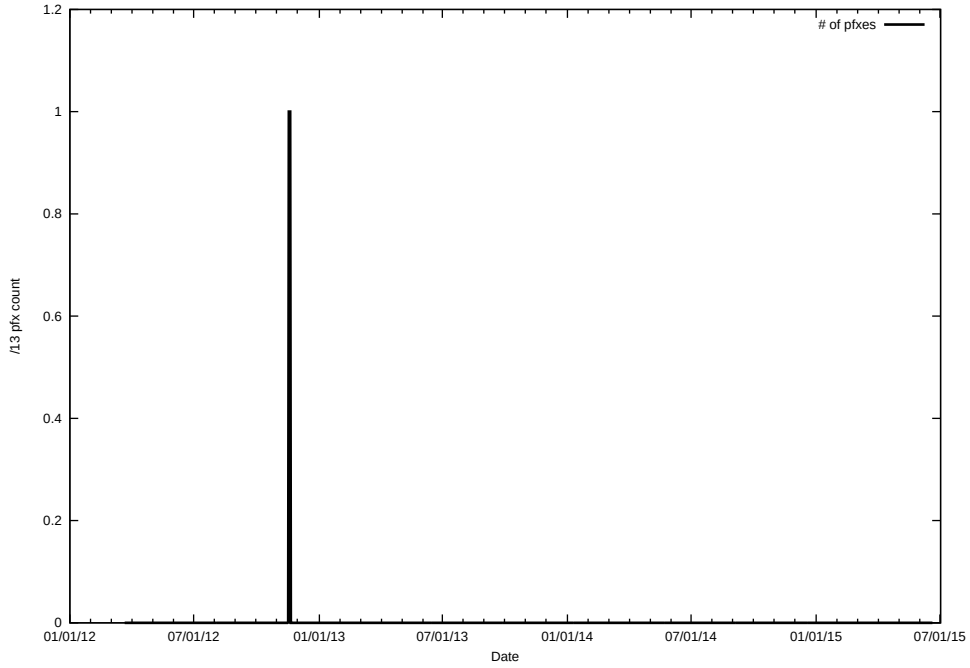
/10


1.5
1
0.5
0
-0.5
-1
/11 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

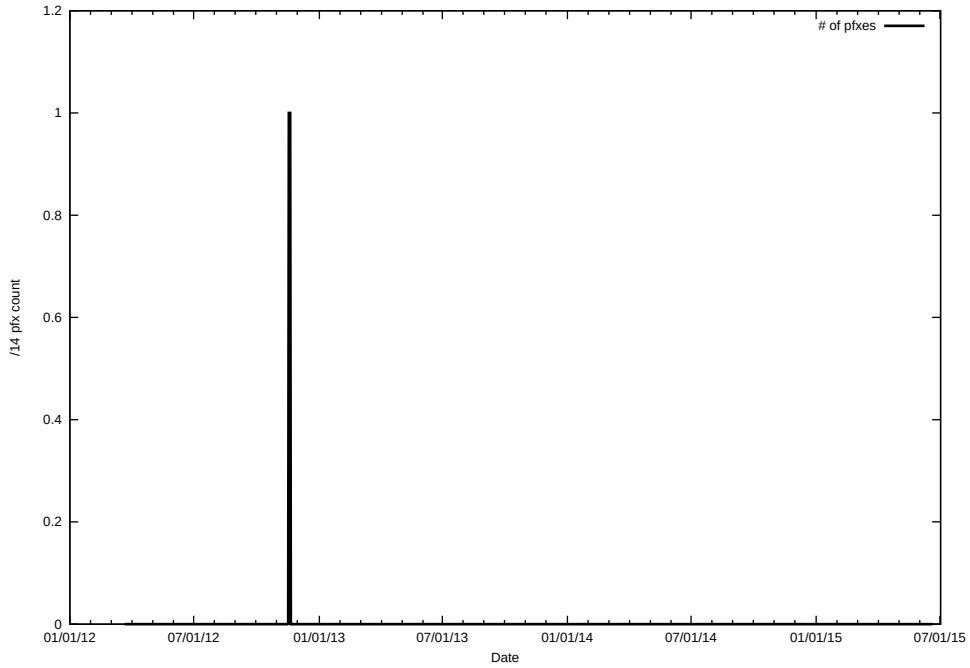
/11


6
5
4
3
2
1
0
/12 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

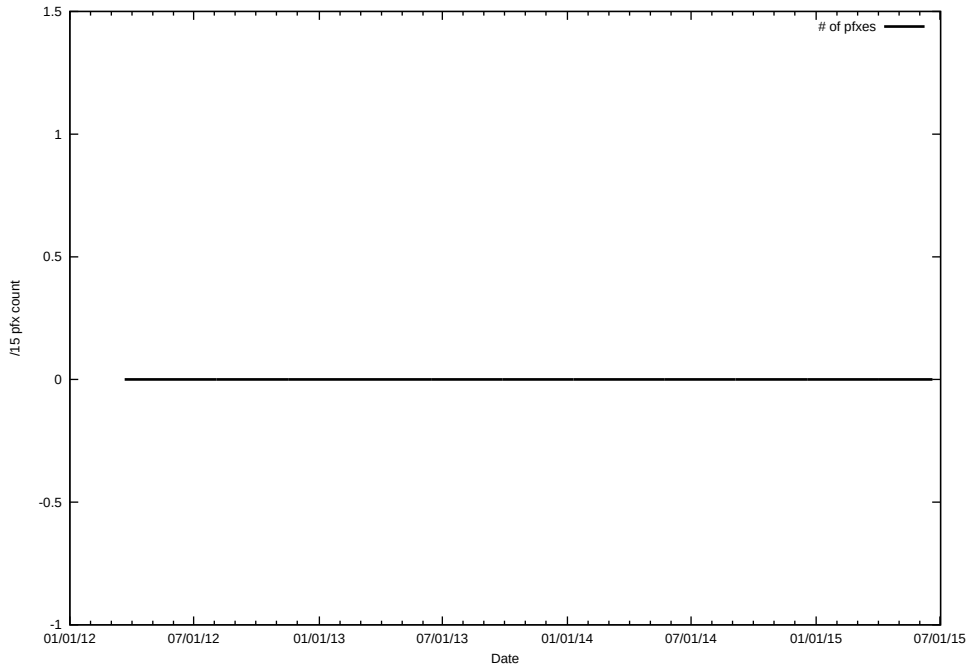
/12


1.2
1
0.8
0.6
0.4
0.2
0
/13 pfx count
# of pfxes
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

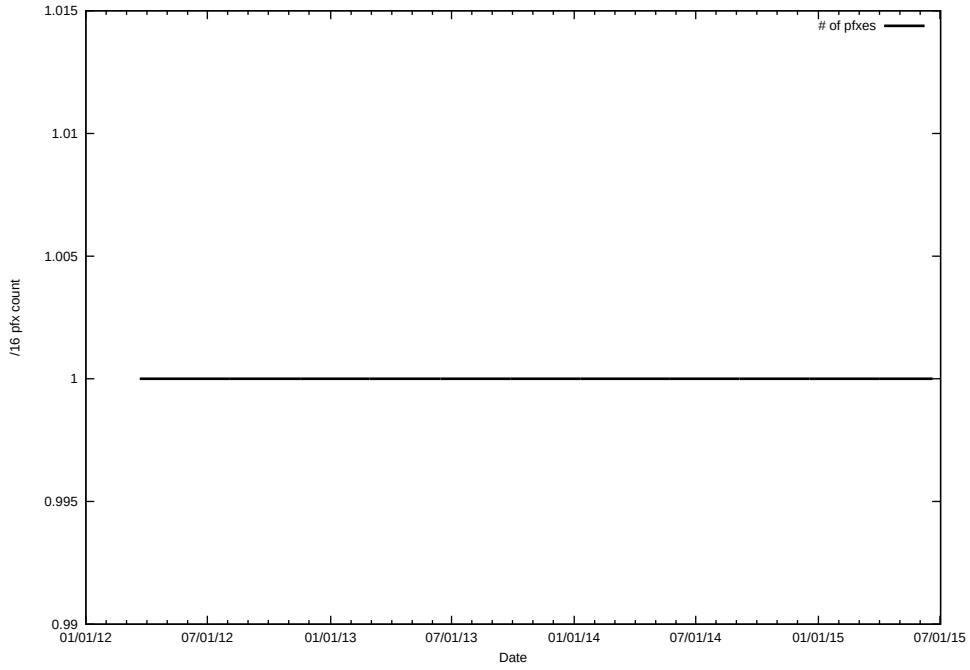
/13


1.2
1
0.8
0.6
0.4
0.2
0
/14 pfx count
# of pfxes
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

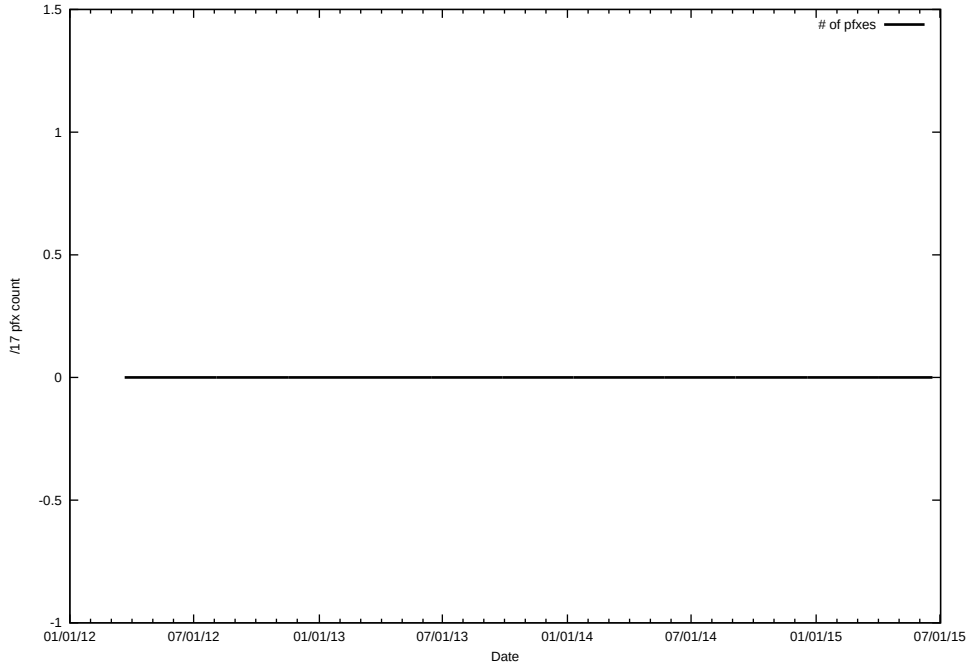
/14


1.5
1
0.5
0
-0.5
-1
/15 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

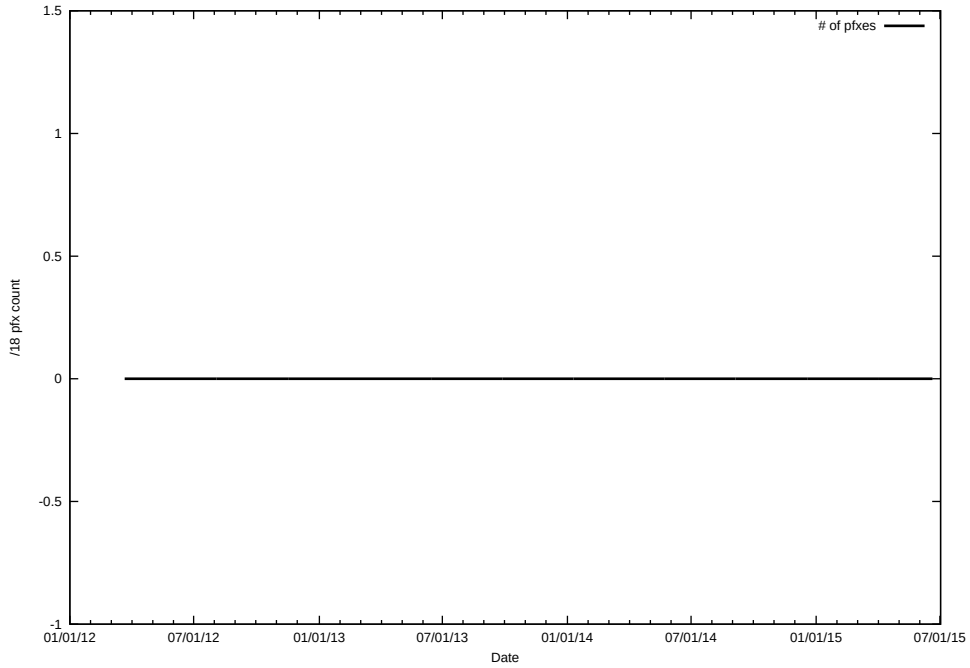
/15


1.015
1.01
1.005
1
0.995
0.99
/16 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

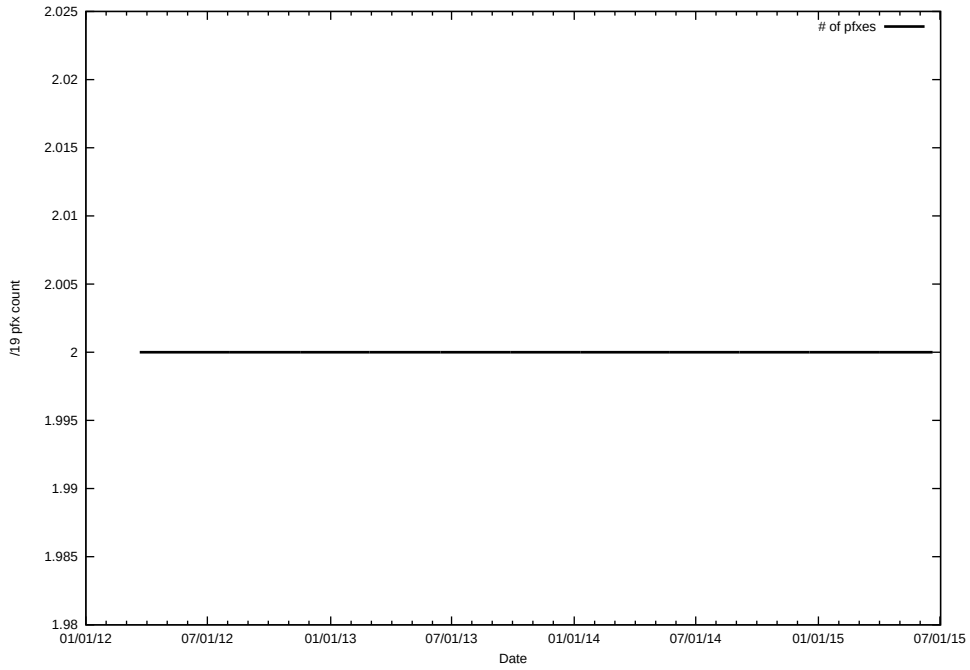
/16


1.5
1
0.5
0
-0.5
-1
/17 pfx count
# of pfxes
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

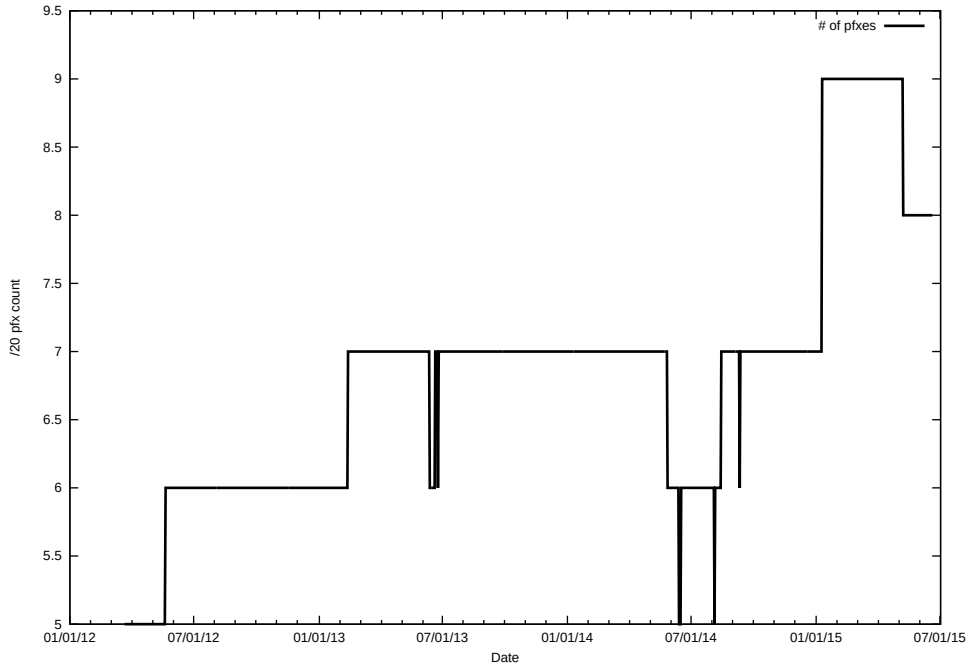
/17


1.5
1
0.5
0
-0.5
-1
/18 pfx count
# of pfxes
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

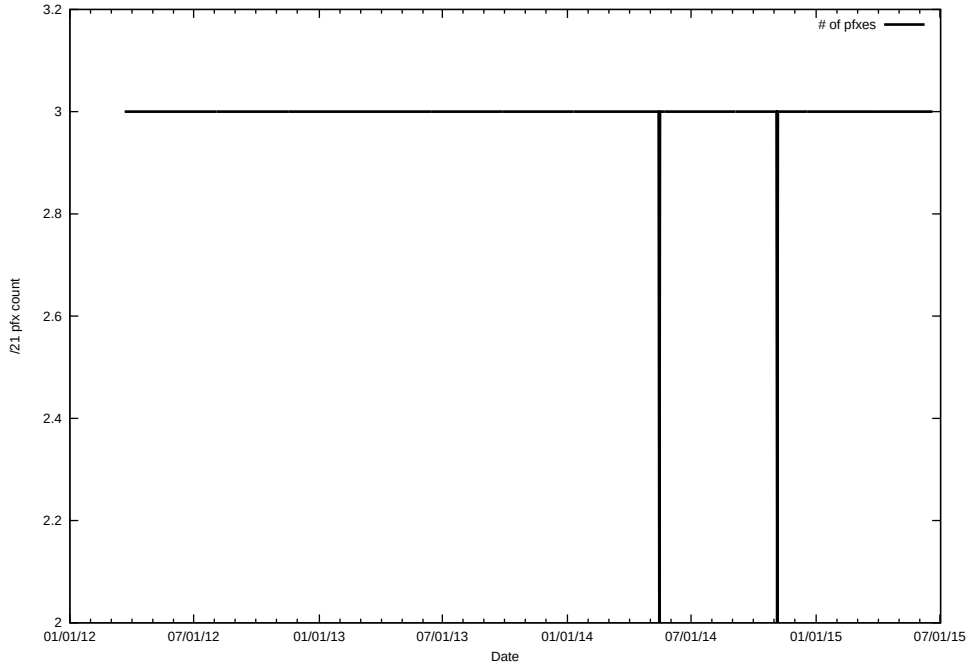
/18


2.025
2.02
2.015
2.01
2.005
2
1.995
1.99
1.985
1.98
/19 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

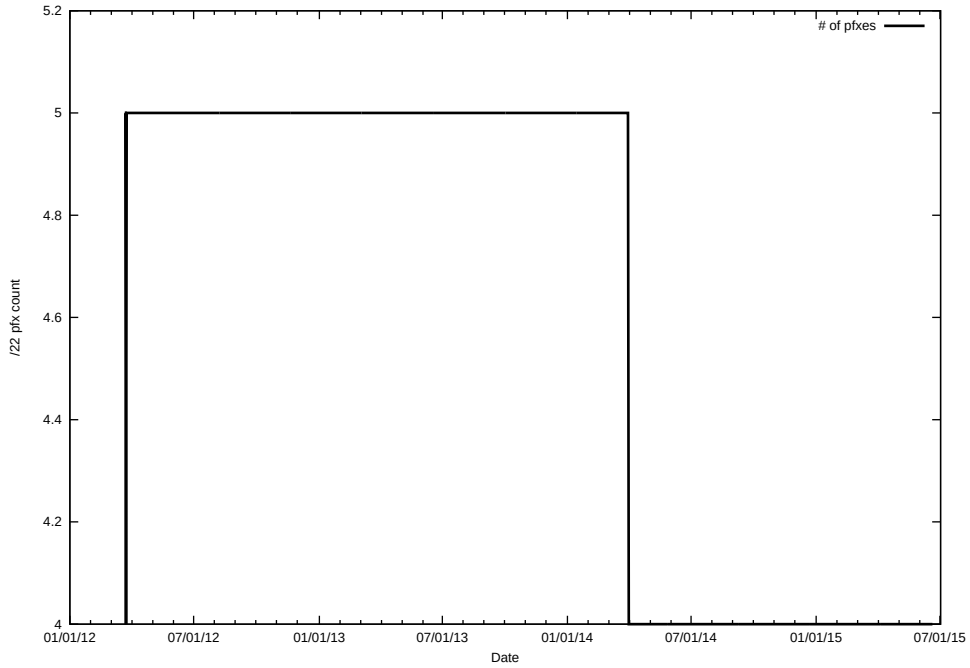
/19


9.5
9
8.5
8
7.5
7
6.5
6
5.5
5
/20 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

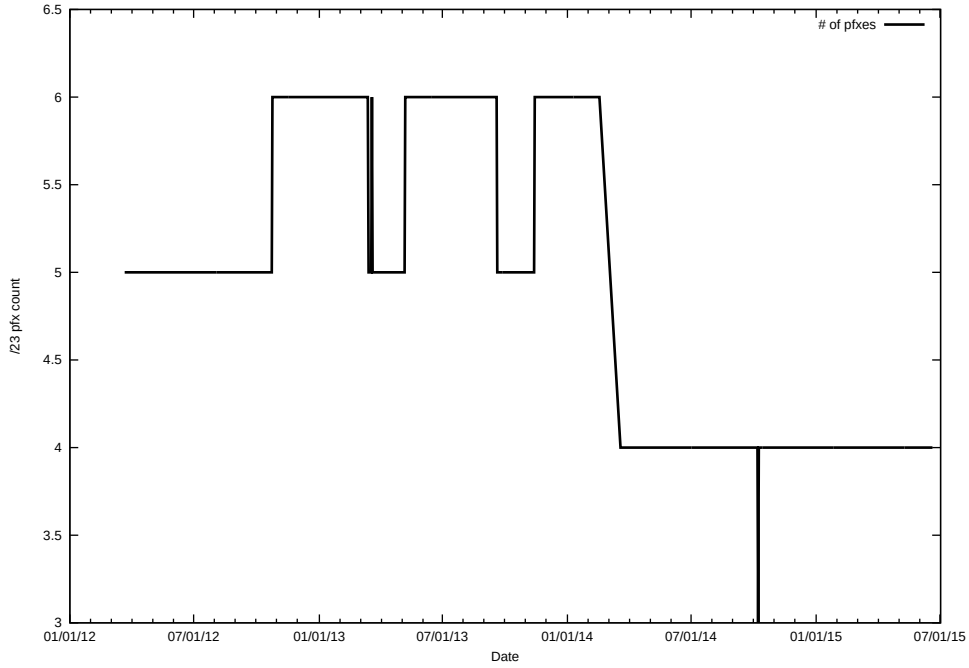
/20


3.2
3
2.8
2.6
2.4
2.2
2
/21 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

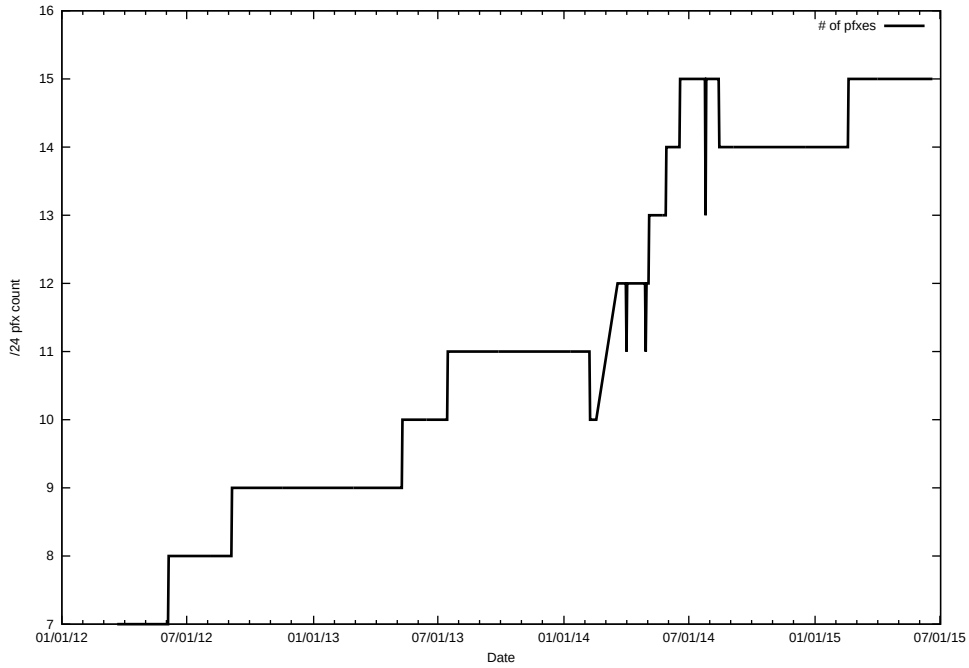
/21


5.2
5
4.8
4.6
4.4
4.2
4
/22 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

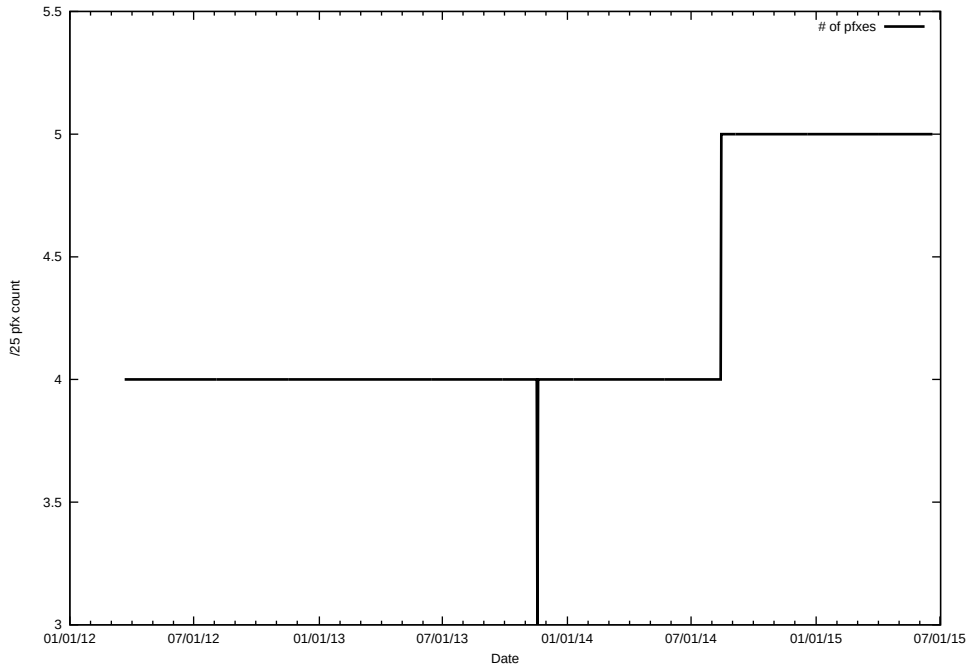
/22


/23 pfx count
Date
# of pfxes
6.5
6
5.5
5
4.5
4
3.5
3
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15

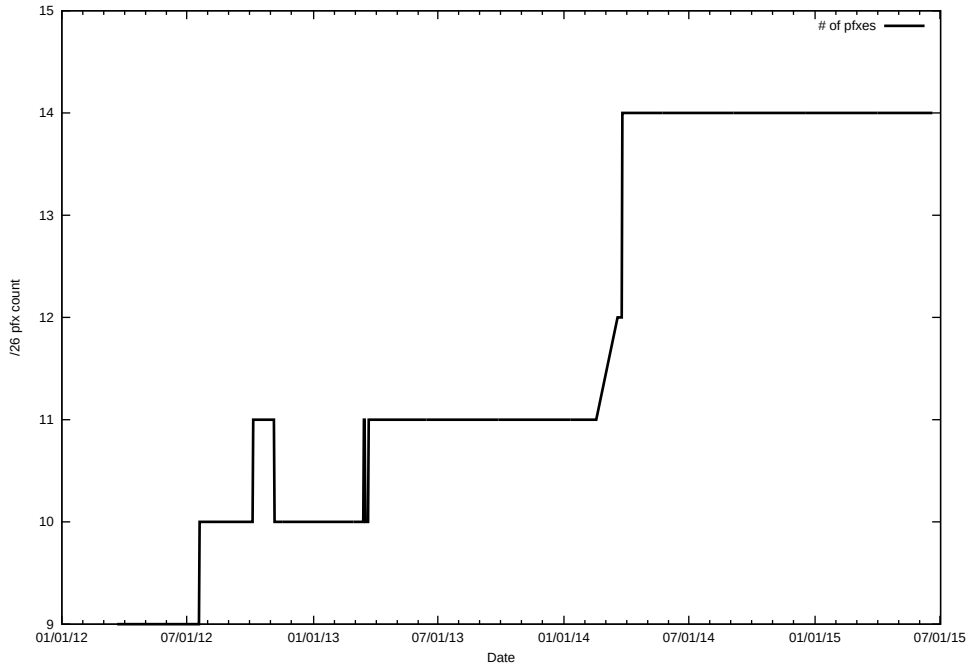
/23


/24 pfx count
Date
# of pfxes
16
15
14
13
12
11
10
9
8
7
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15

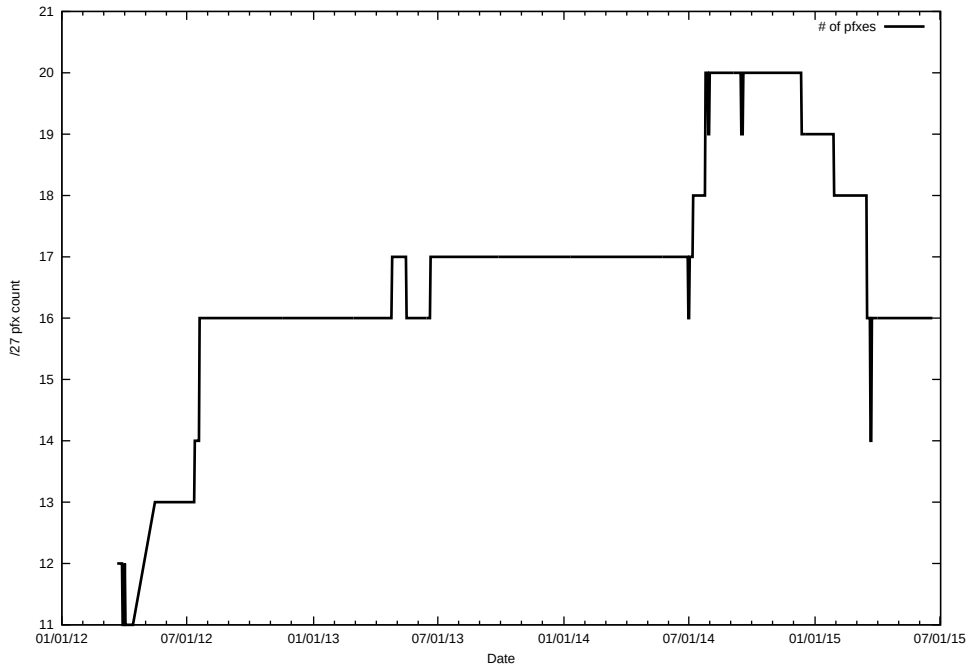
/24


5.5
5
4.5
4
3.5
3
/25 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

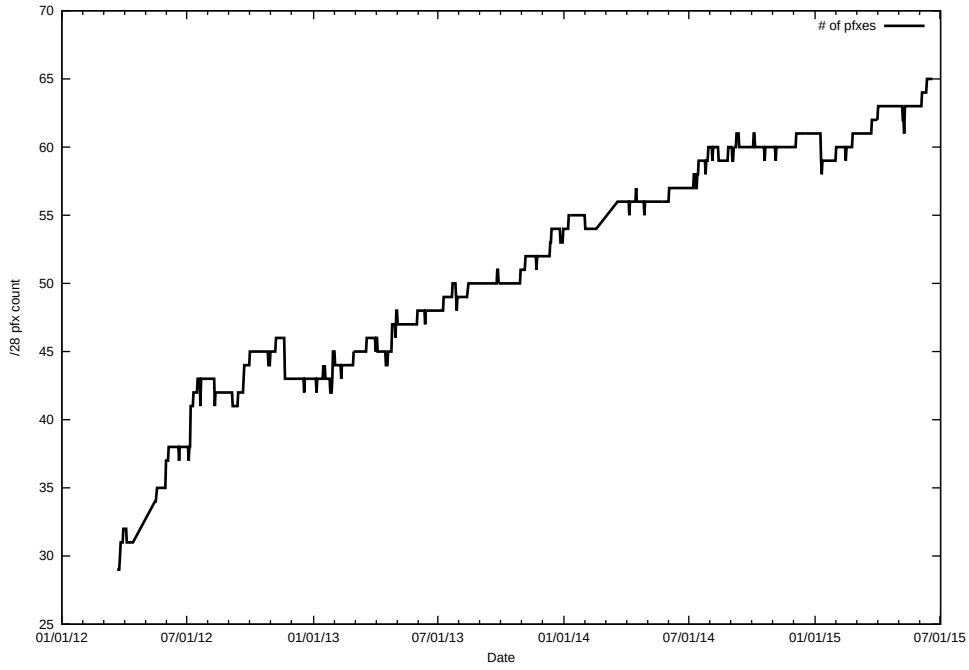
/25


15
14
13
12
11
10
9
/26 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

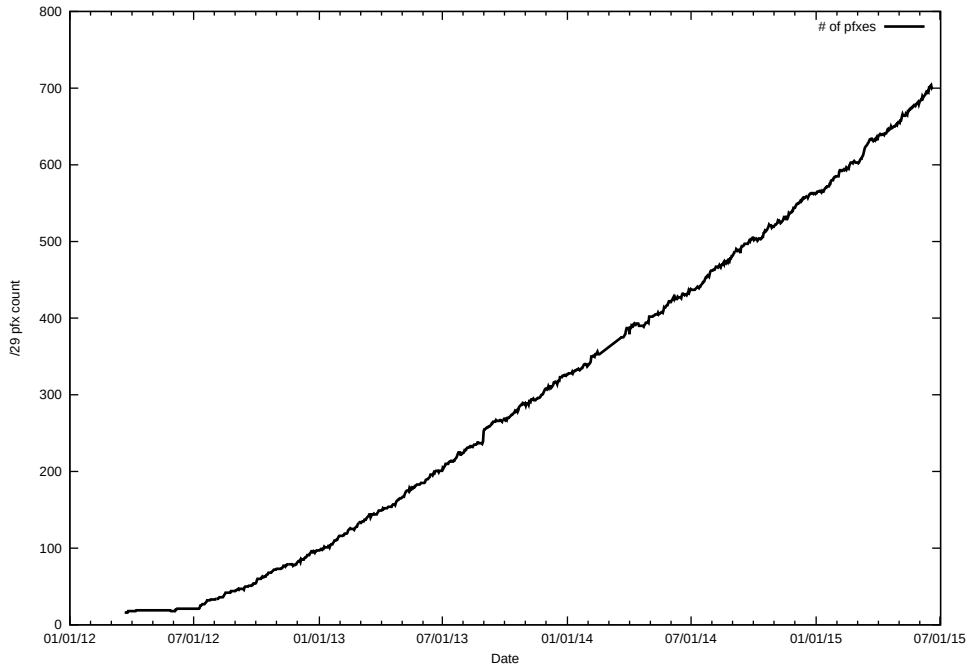
/26


/27 pfx count
Date
# of pfxes
21
20
19
18
17
16
15
14
13
12
11
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15

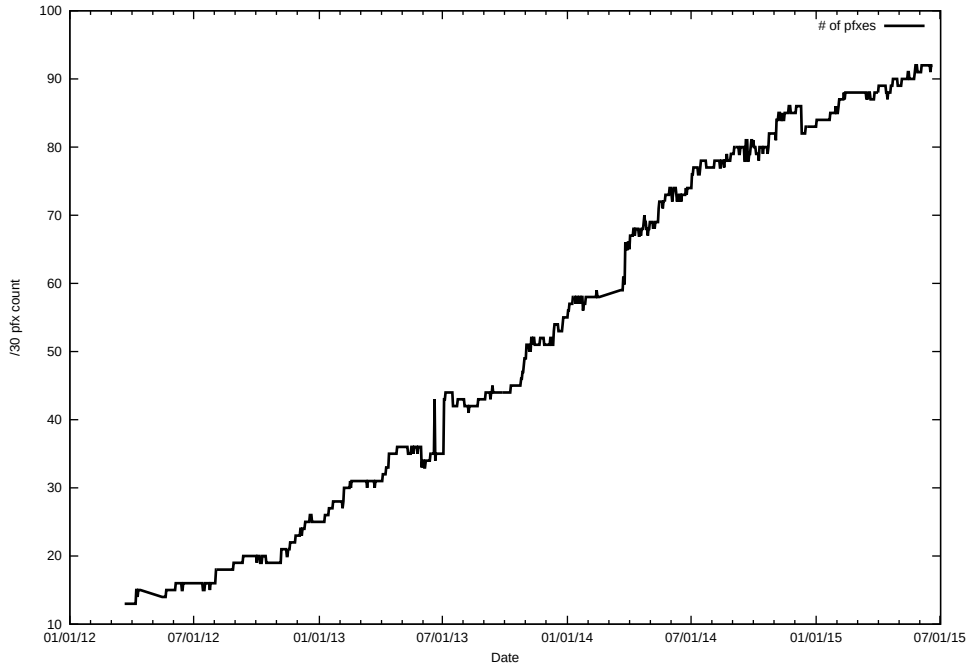
/27


/28 pfx count
Date
# of pfxes
70
65
60
55
50
45
40
35
30
25
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15

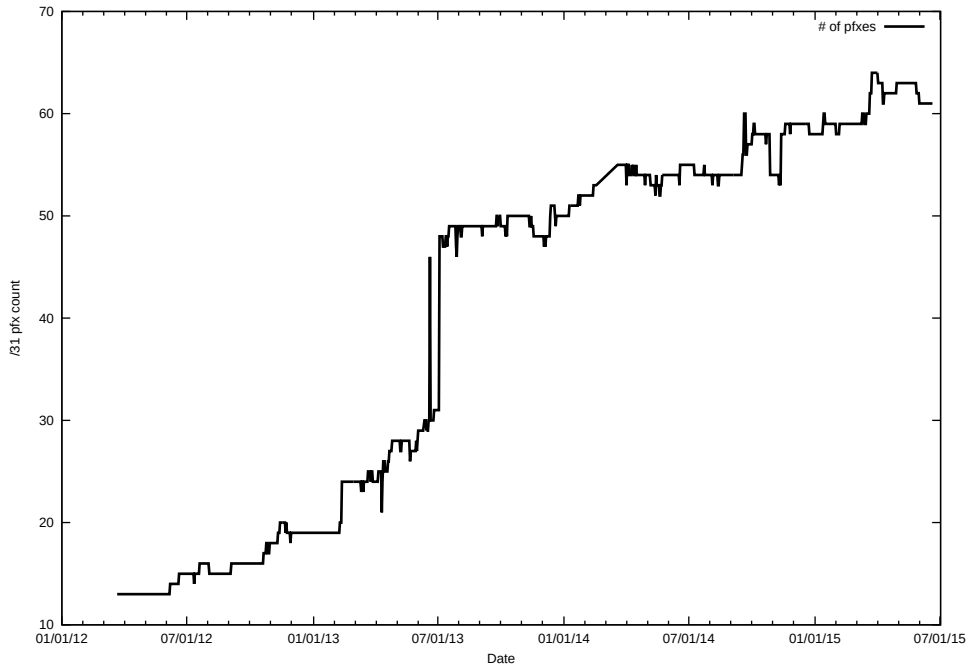
/28


800
700
600
500
400
300
200
100
0
/29 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

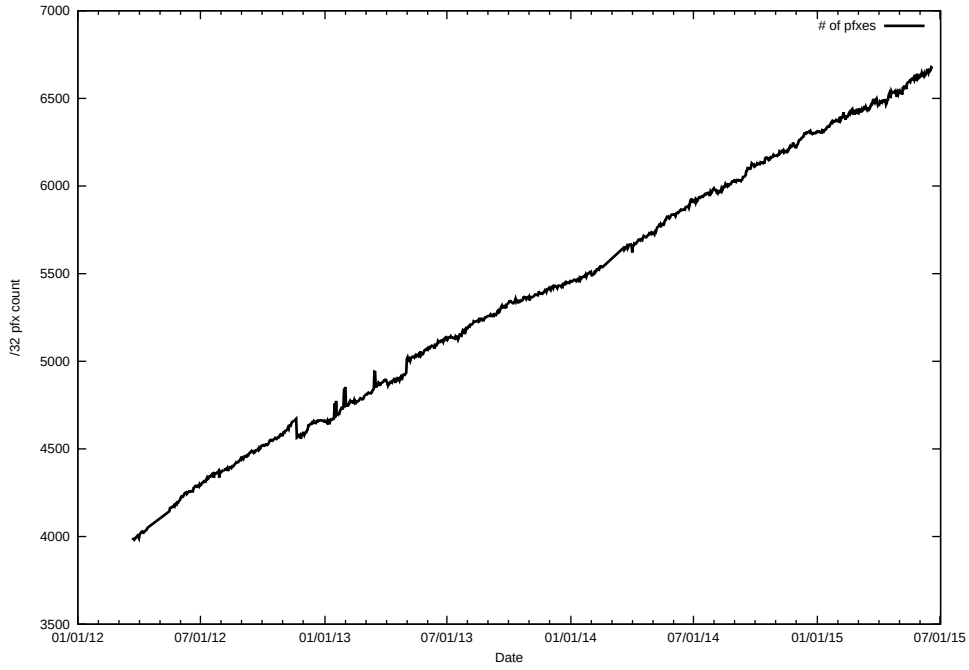
/29


100
90
80
70
60
50
40
30
20
10
/30 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

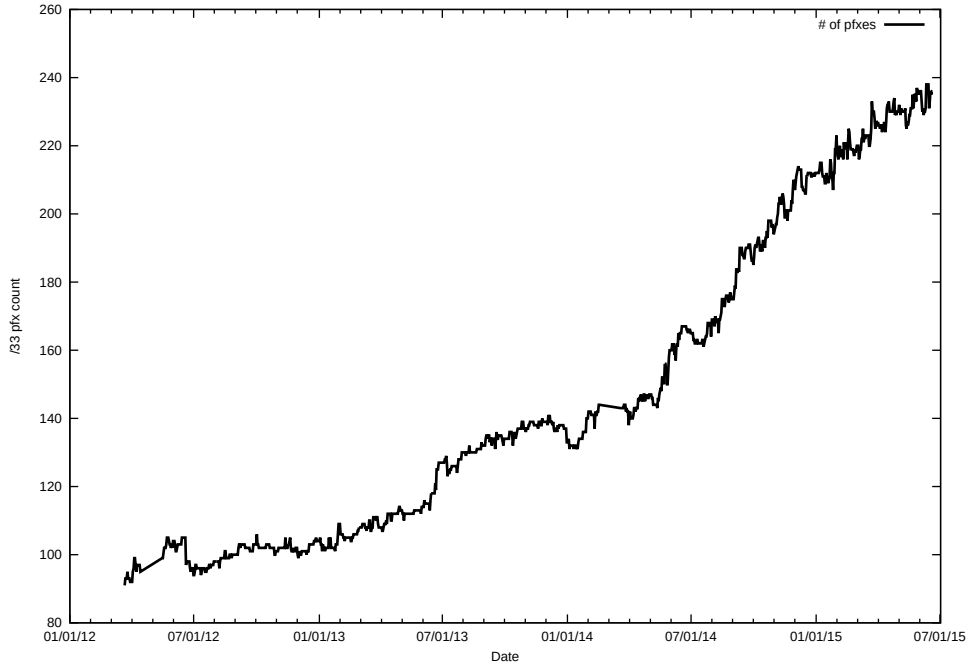
/30


70
60
50
40
30
20
10
/31 pfx count
# of pfxes
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

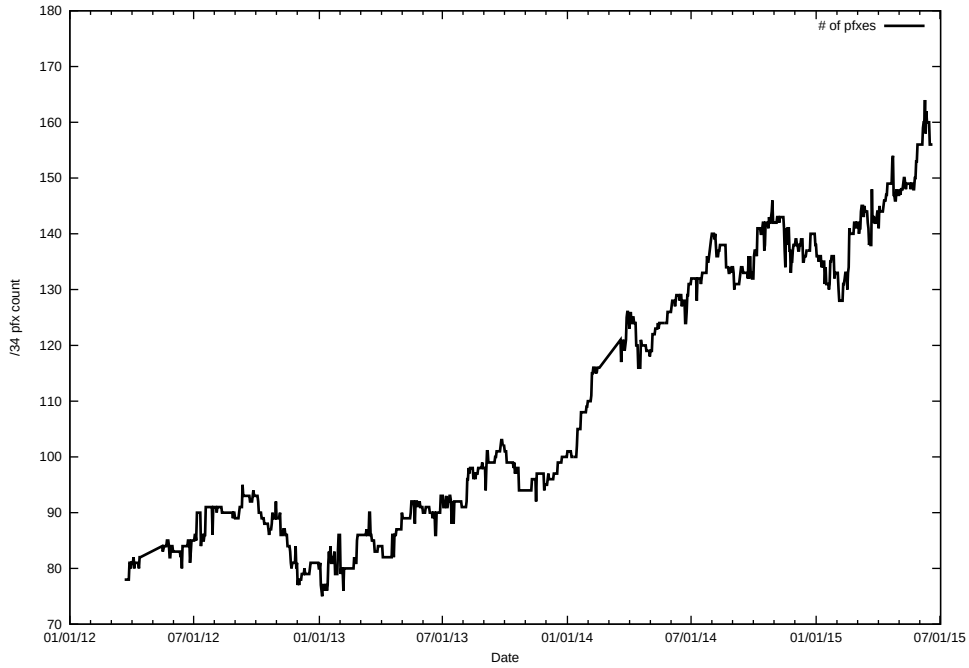
/31


7000
6500
6000
5500
5000
4500
4000
3500
/32 pfx count
# of pfxes
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

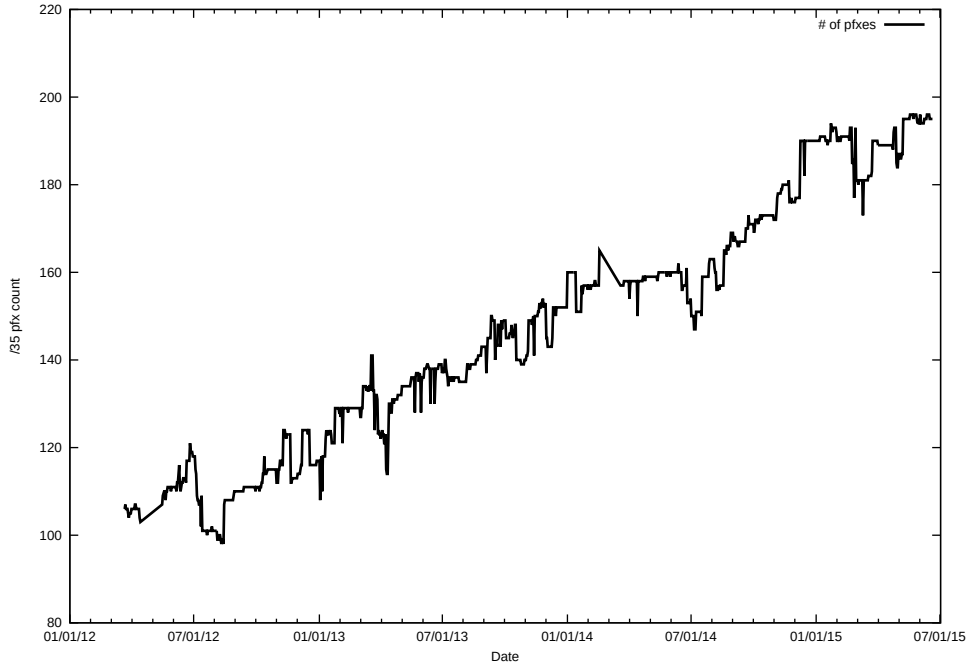
/32


/33 pfx count
Date
# of pfxes
260
240
220
200
180
160
140
120
100
80
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15

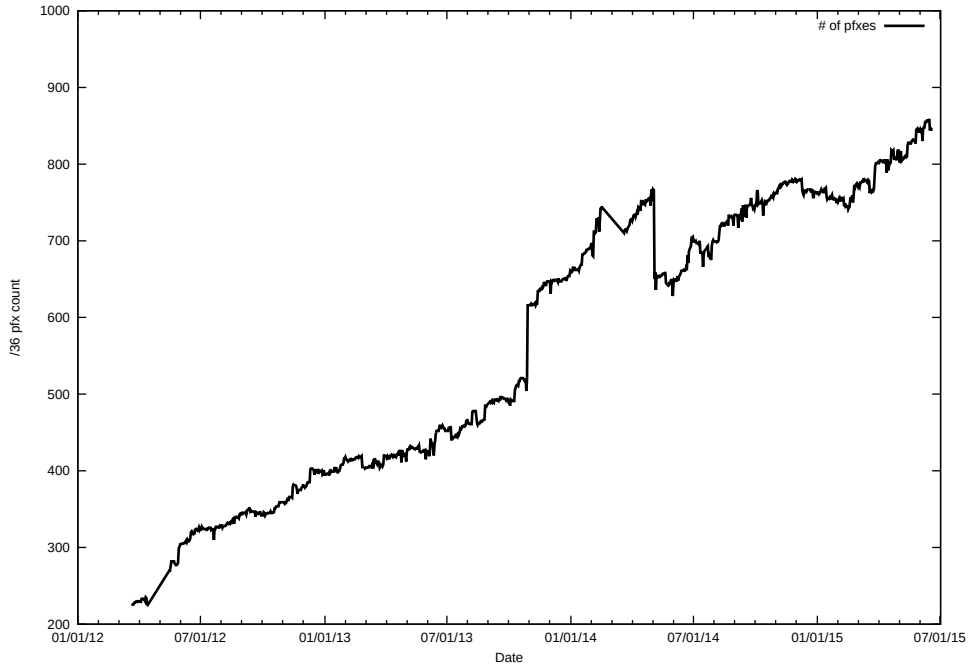
/33


/34 pfx count
Date
# of pfxes
180
170
160
150
140
130
120
110
100
90
80
70
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15

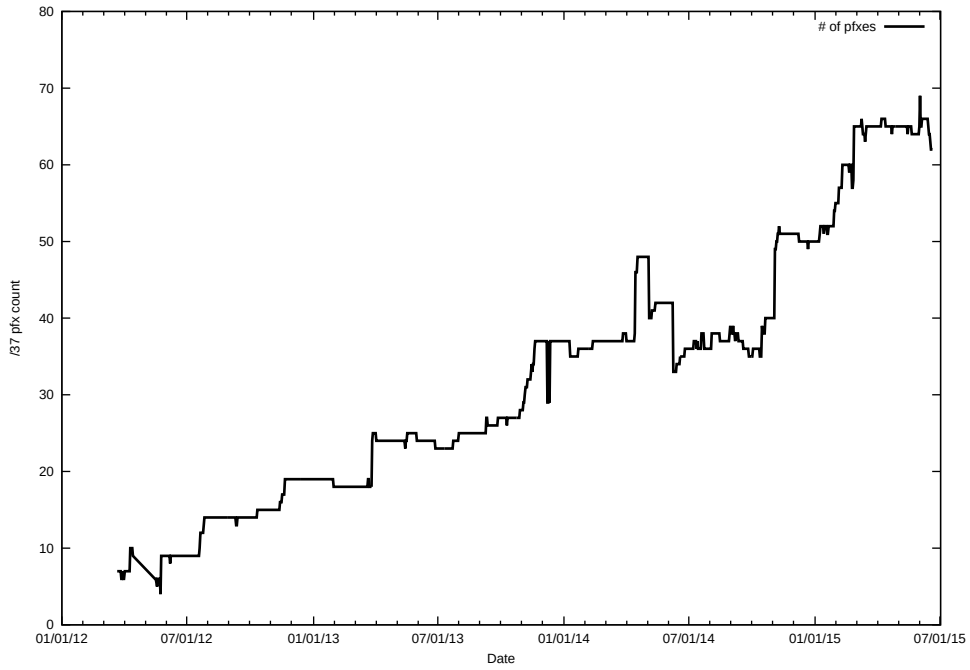
/34


220
200
180
160
140
120
100
80
/35 pfx count
# of pfxes
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

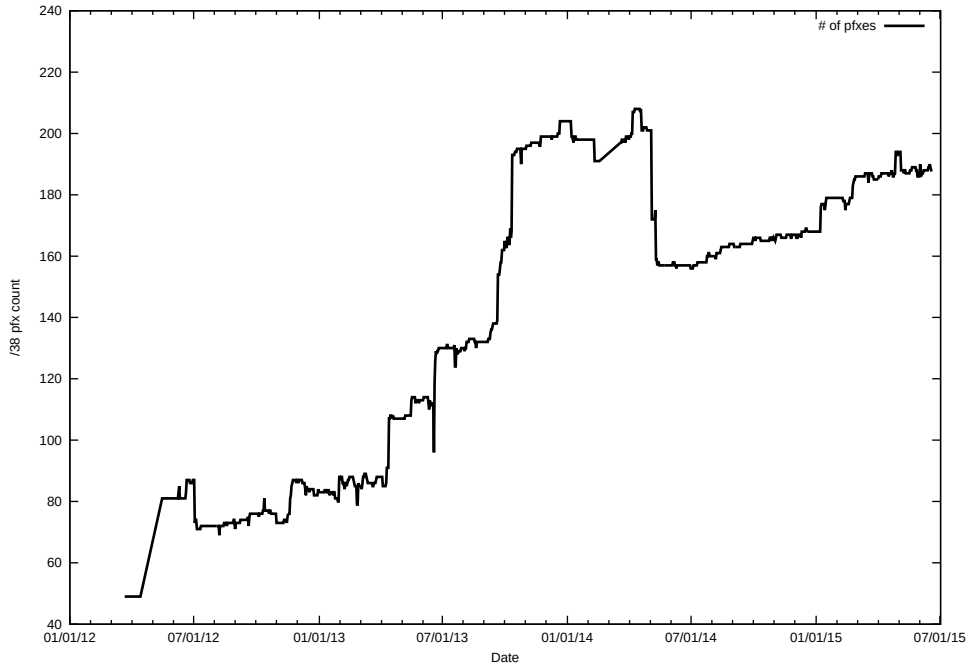
/35


1000
900
800
700
600
500
400
300
200
/36 pfx count
# of pfxes
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

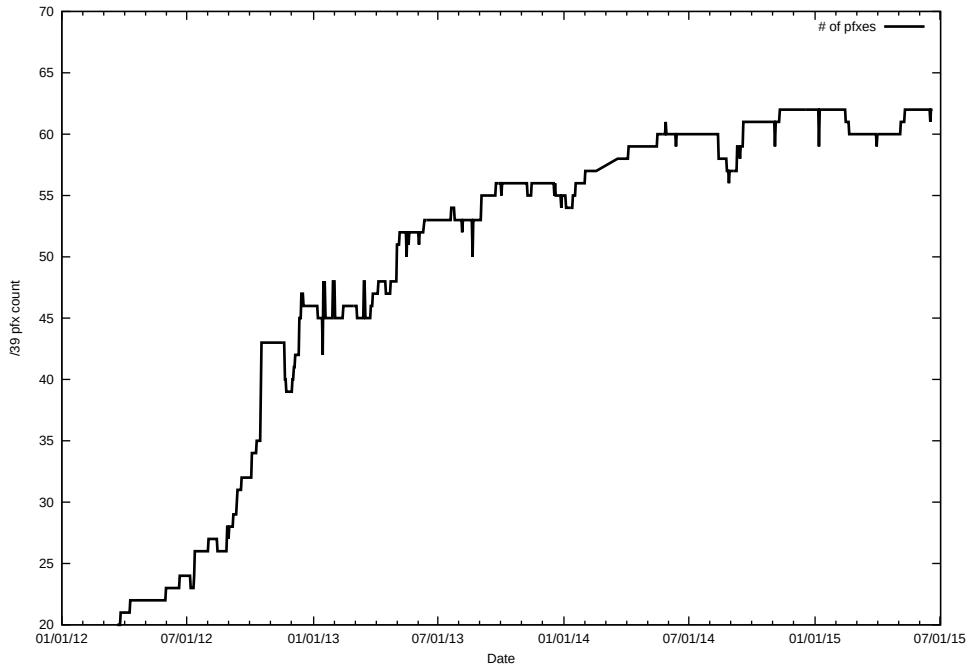
/36


/37 pfx count
Date
# of pfxes
0
10
20
30
40
50
60
70
80
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15

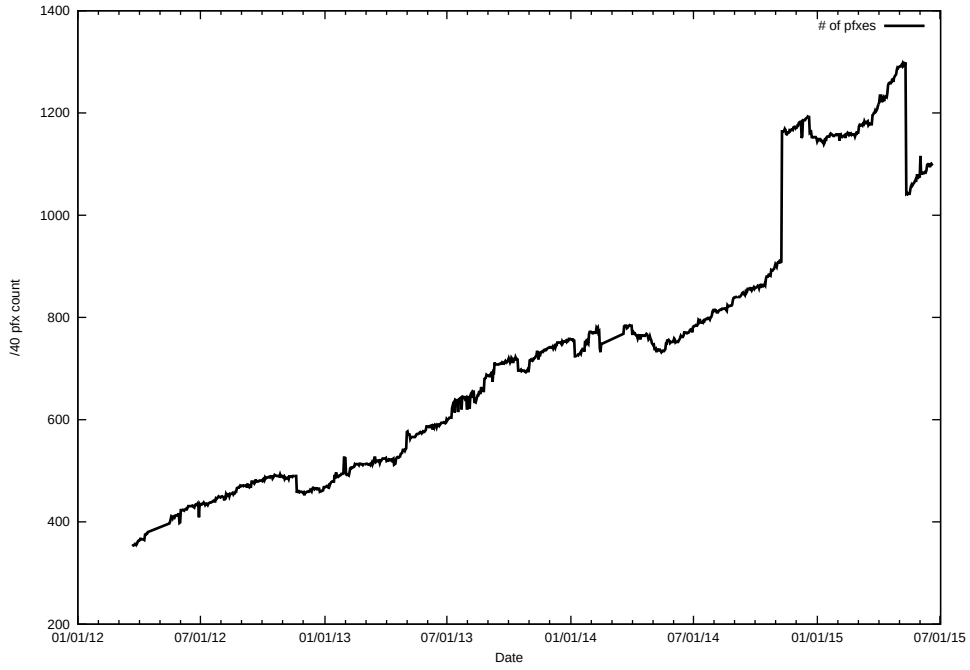
/37


/38 pfx count
Date
# of pfxes
40
60
80
100
120
140
160
180
200
220
240
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15

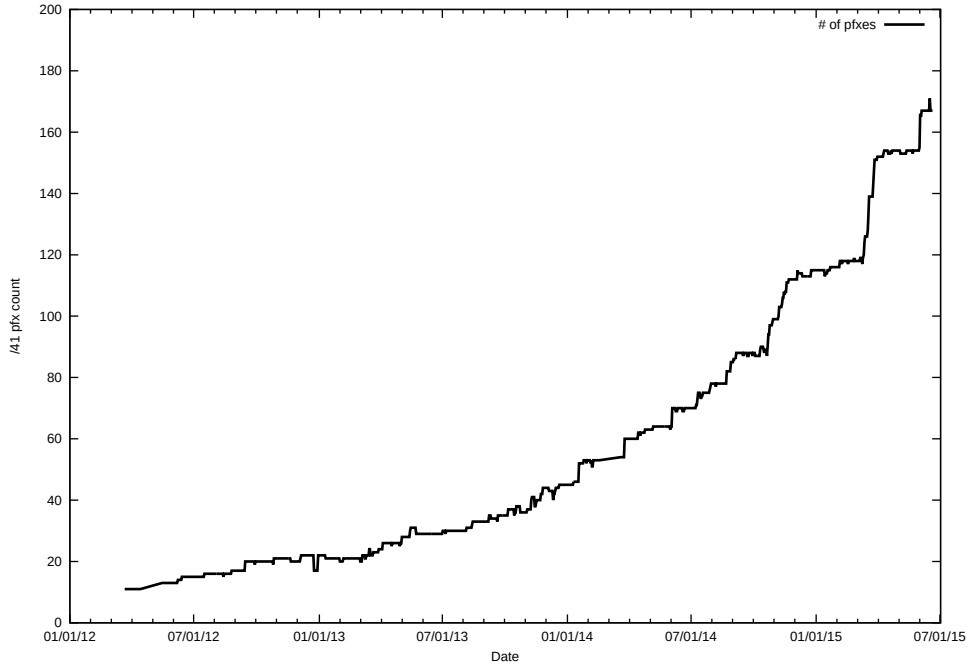
/38


70
65
60
55
50
45
40
35
30
25
20
/39 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

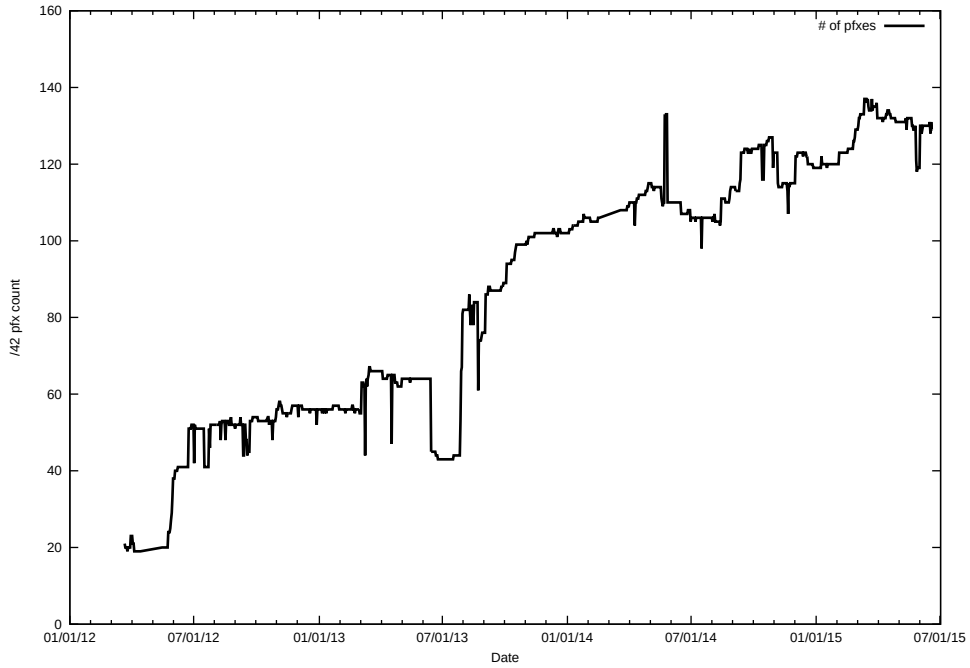
/39


1400
1200
1000
800
600
400
200
/40 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

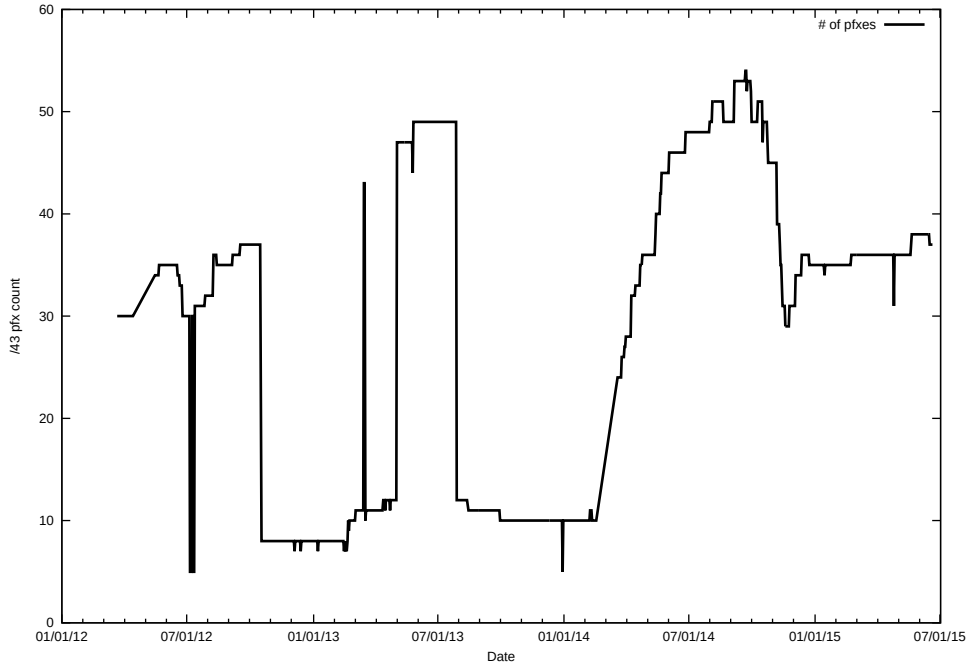
/40


/41 pfx count
Date
# of pfxes
0
20
40
60
80
100
120
140
160
180
200
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15

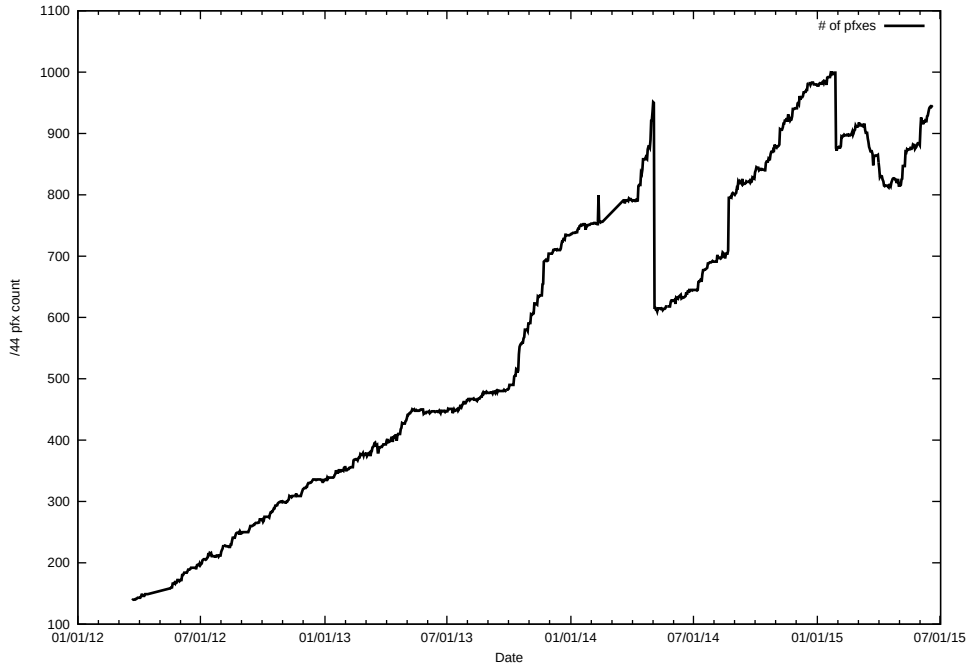
/41


/42 pfx count
Date
# of pfxes
0
20
40
60
80
100
120
140
160
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15

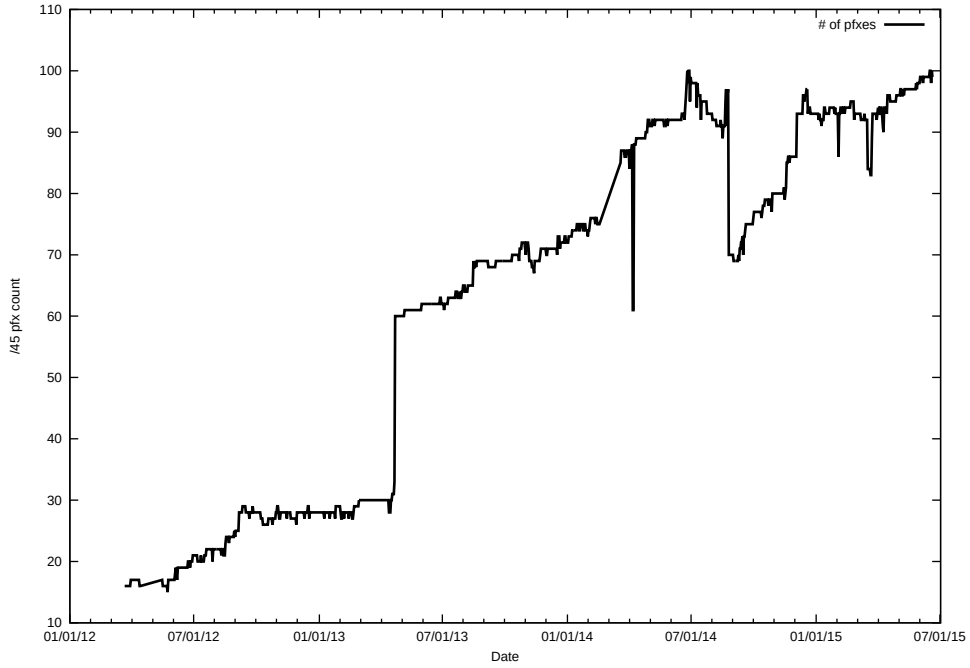
/42


/43 pfx count
Date
# of pfxes
0
10
20
30
40
50
60
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15

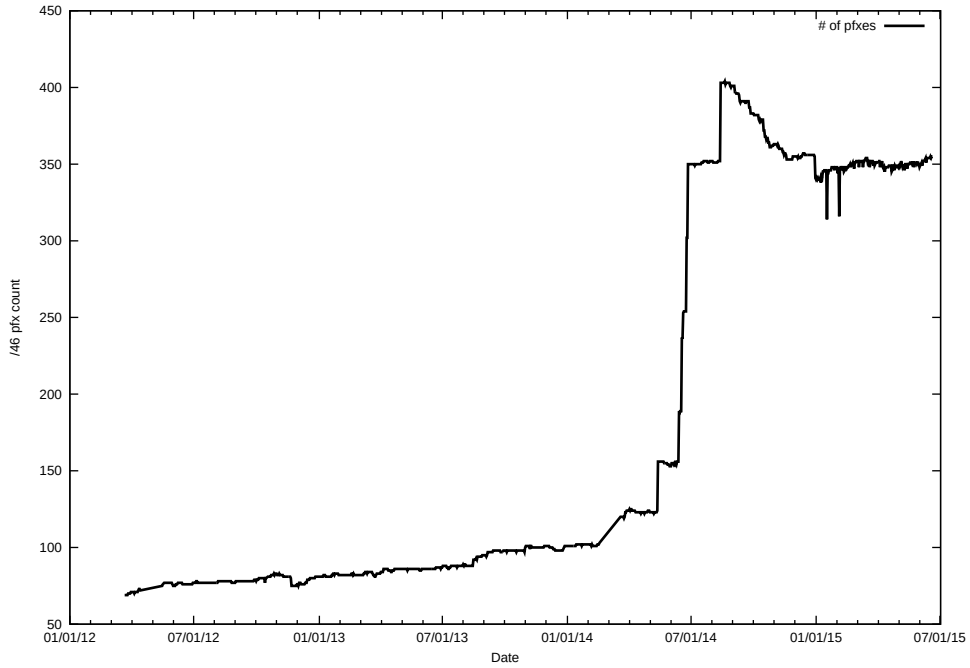
/43


/44 pfx count
Date
# of pfxes
100
200
300
400
500
600
700
800
900
1000
1100
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15

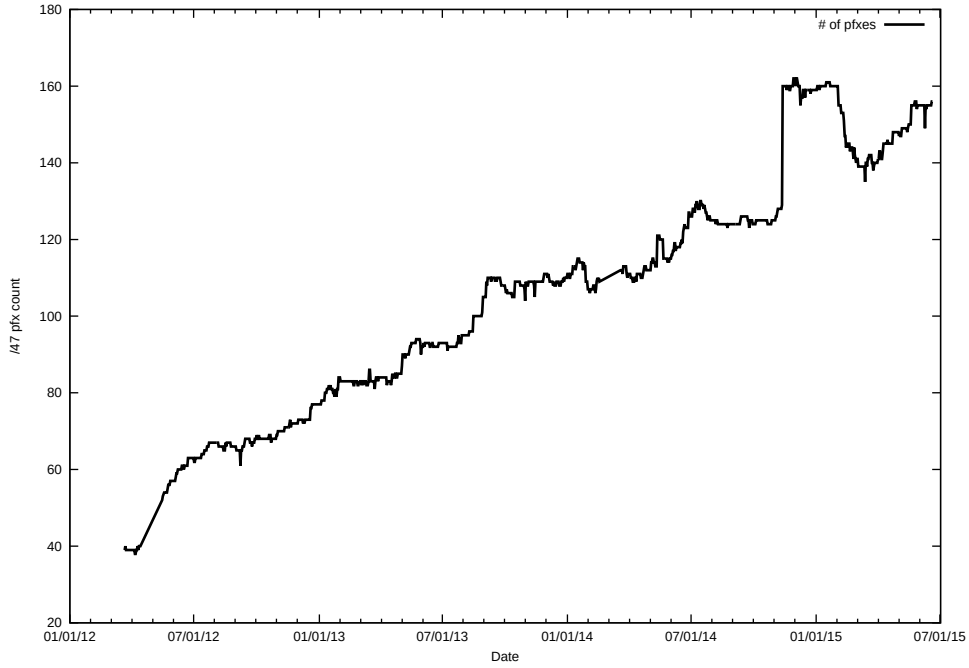
/44


110
100
90
80
70
60
50
40
30
20
10
/45 pfx count
# of pfxes
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

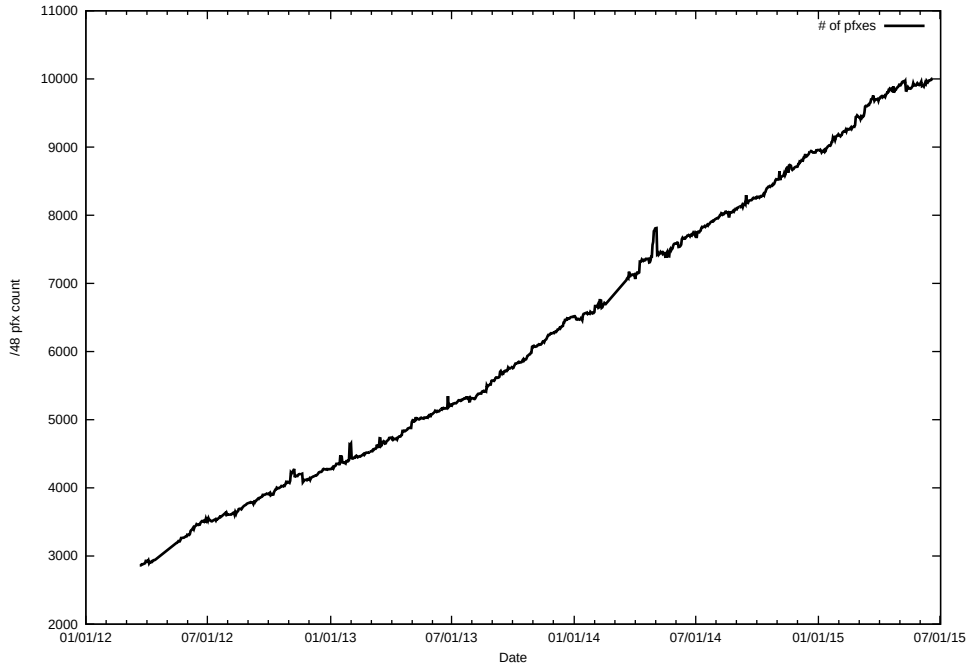
/45


450
400
350
300
250
200
150
100
50
/46 pfx count
# of pfxes
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

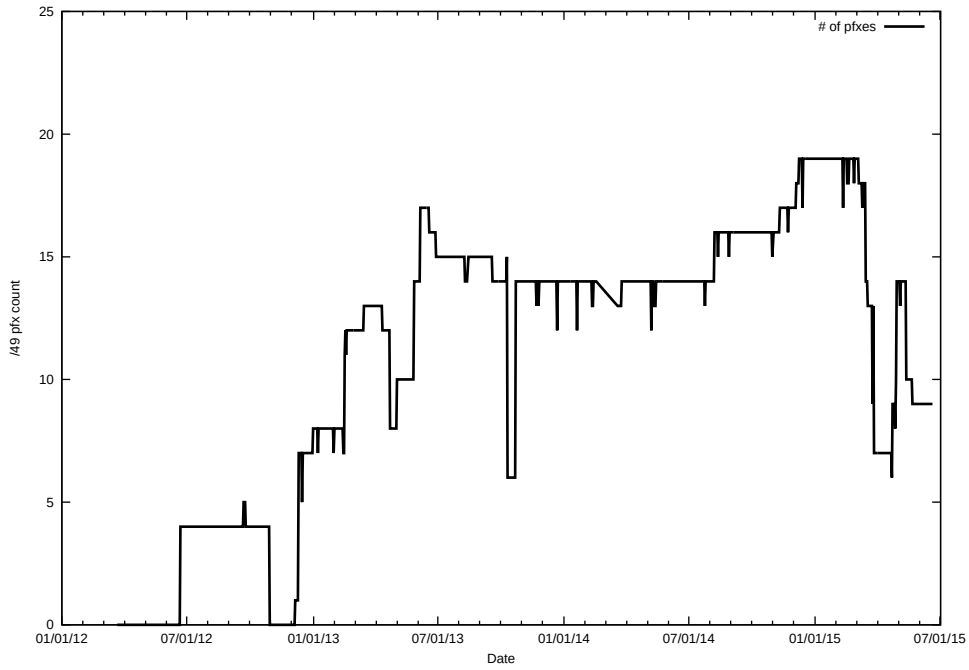
/46


/47 pfx count
Date
# of pfxes
180
160
140
120
100
80
60
40
20
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15

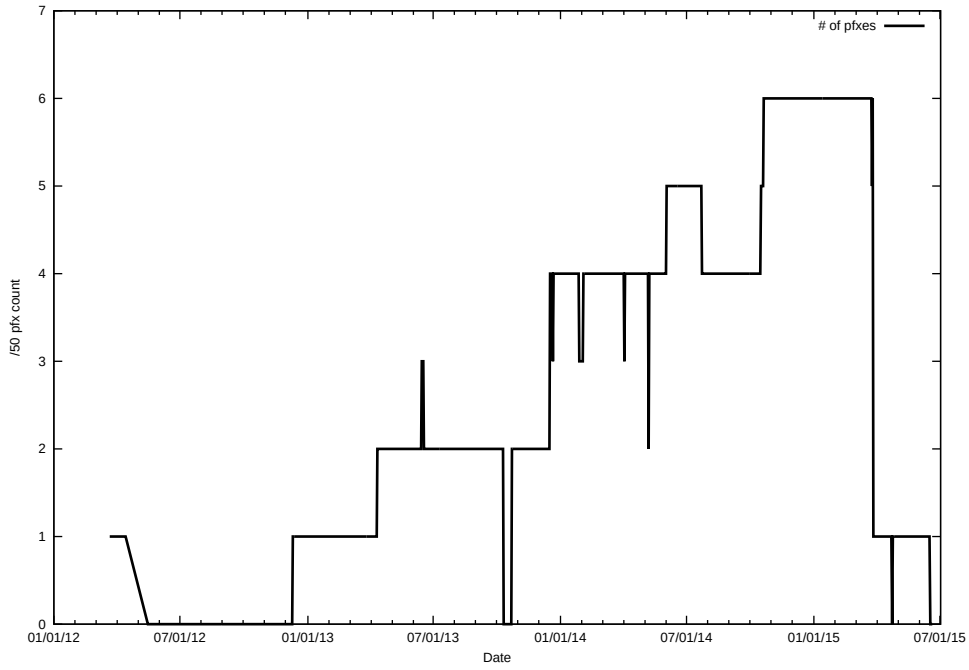
/47


/48 pfx count
Date
# of pfxes
11000
10000
9000
8000
7000
6000
5000
4000
3000
2000
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15

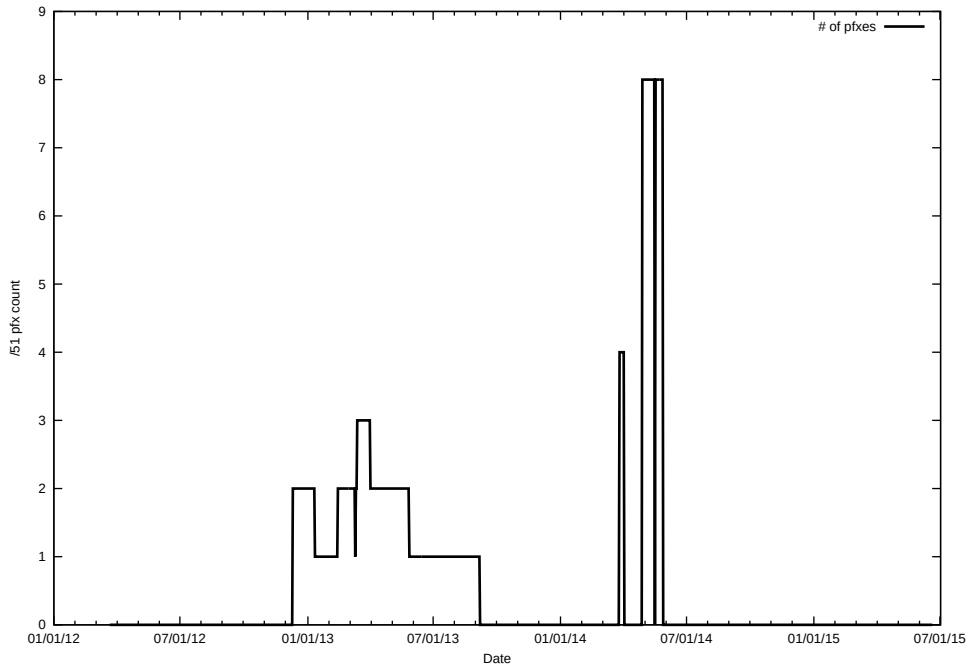
/48


25
20
15
10
5
0
/49 pfx count
# of pfxes
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

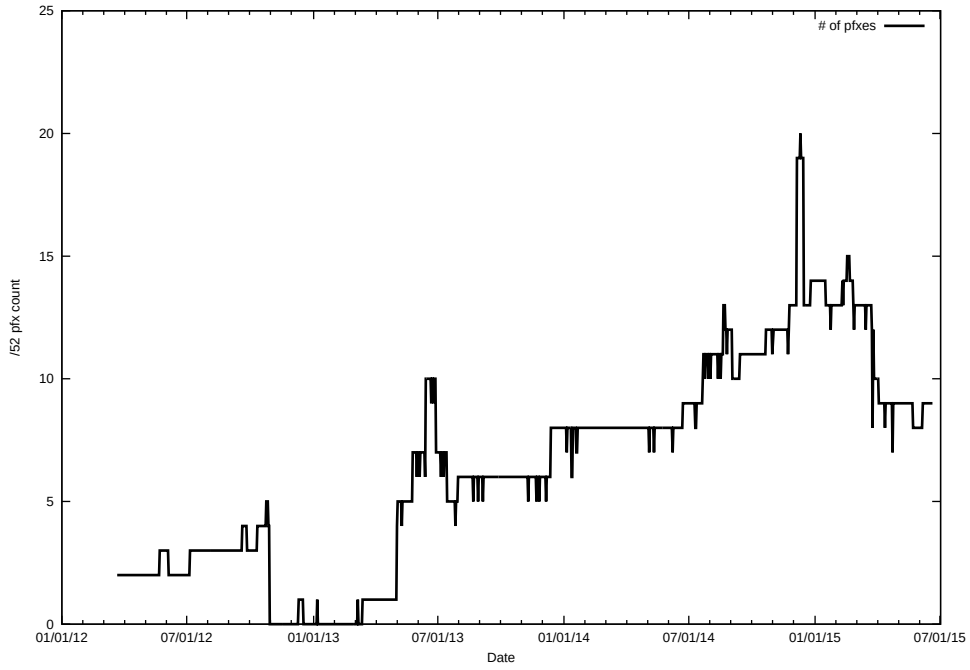
/49


7
6
5
4
3
2
1
0
/50 pfx count
# of pfxes
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

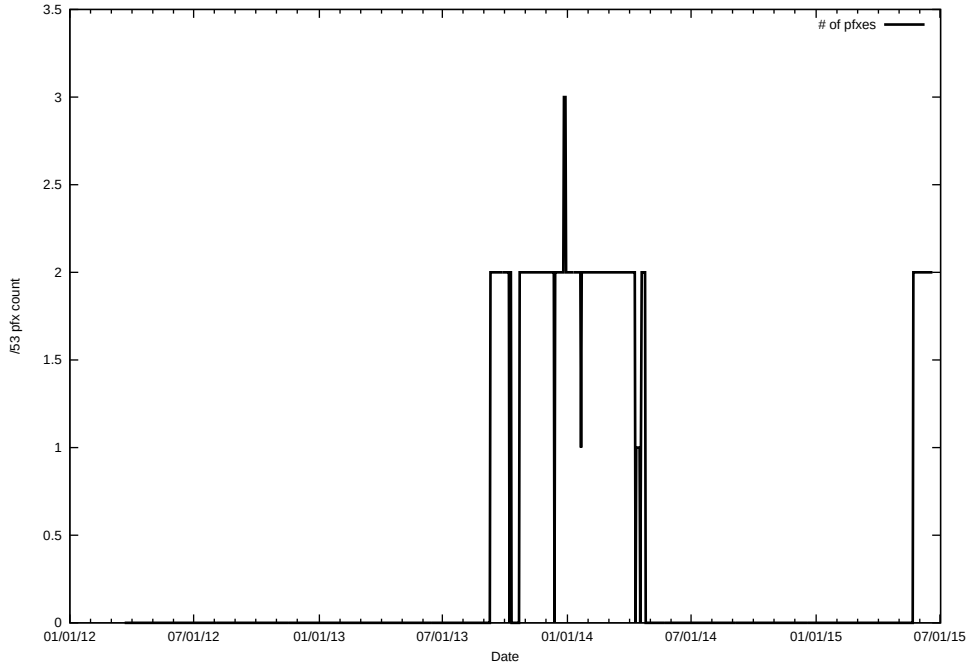
/50


/51 pfx count
Date
# of pfxes
0
1
2
3
4
5
6
7
8
9
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15

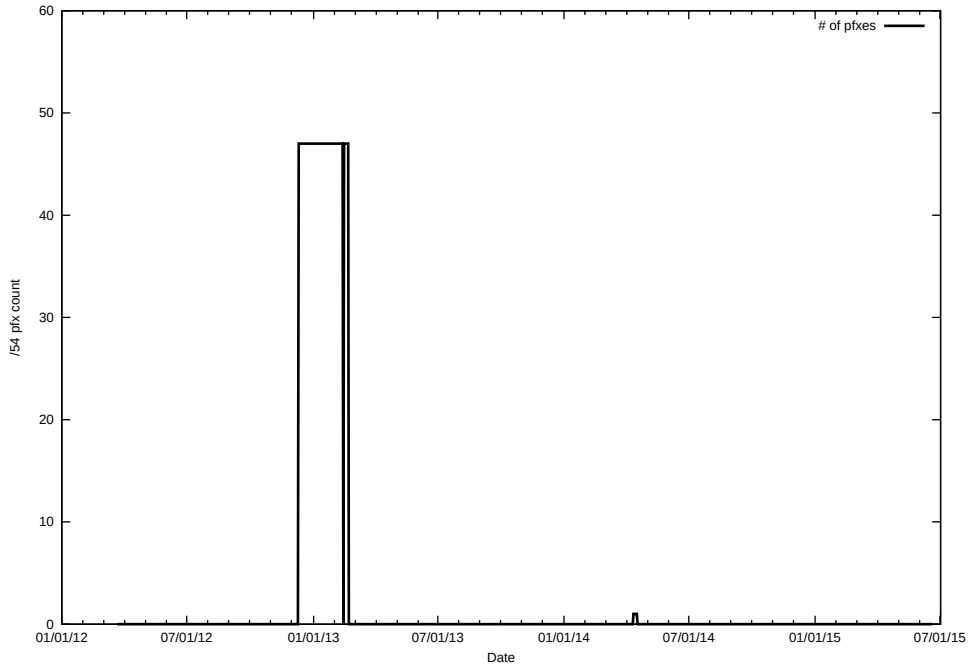
/51


/52 pfx count
Date
# of pfxes
0
5
10
15
20
25
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15

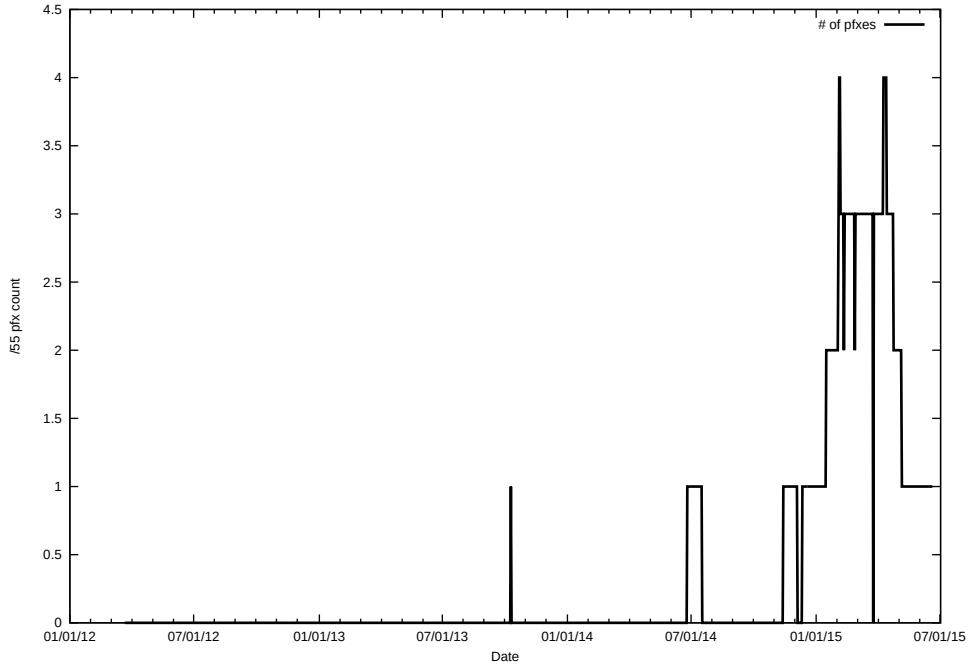
/52


3.5
3
2.5
2
1.5
1
0.5
0
/53 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

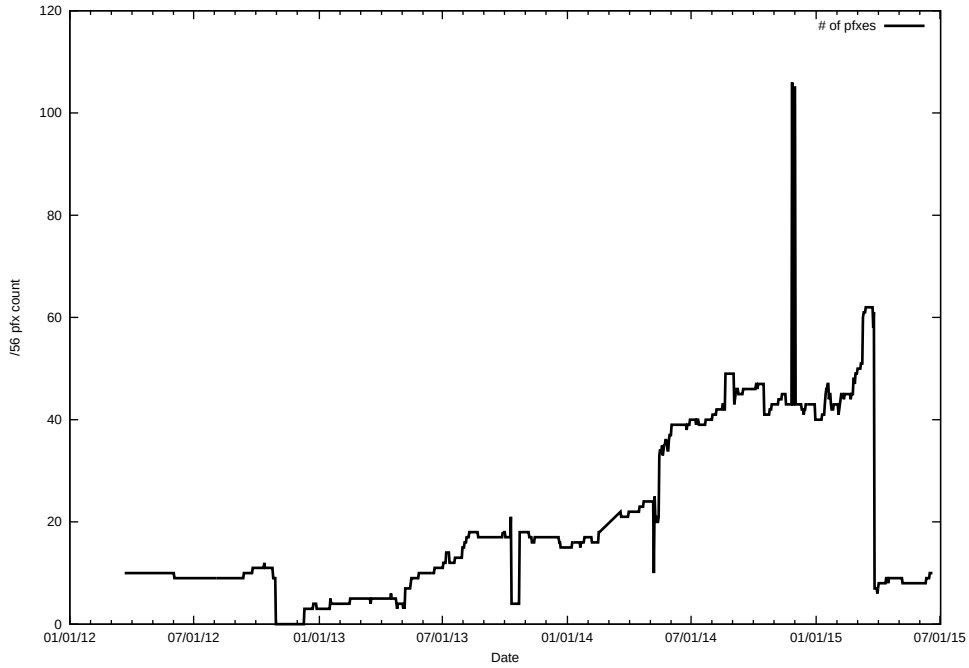
/53


60
50
40
30
20
10
0
/54 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

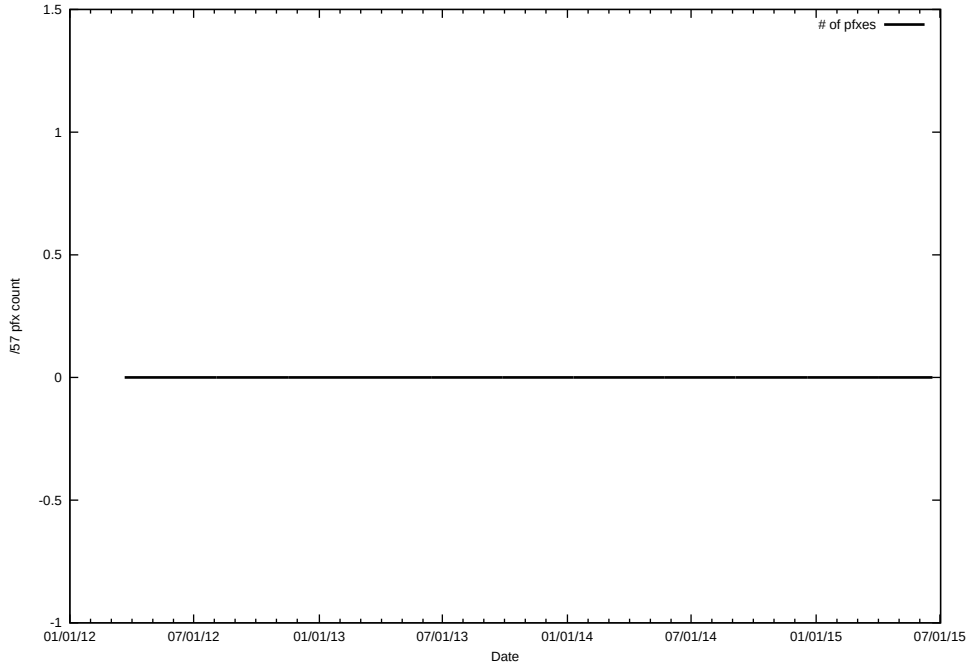
/54


/55 pfx count
Date
# of pfxes
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15

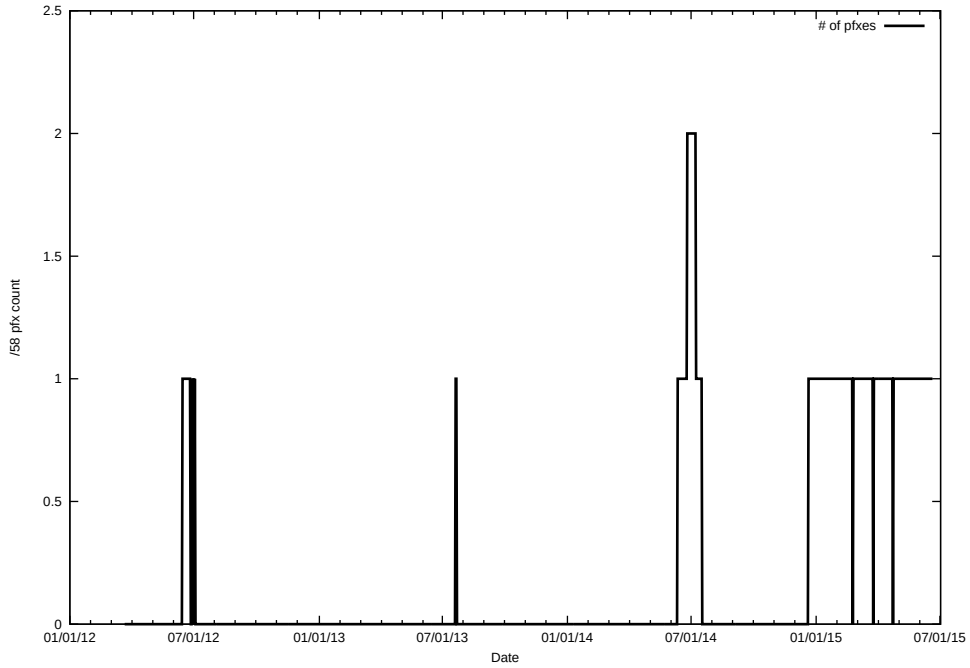
/55


/56 pfx count
Date
# of pfxes
120
100
80
60
40
20
0
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15

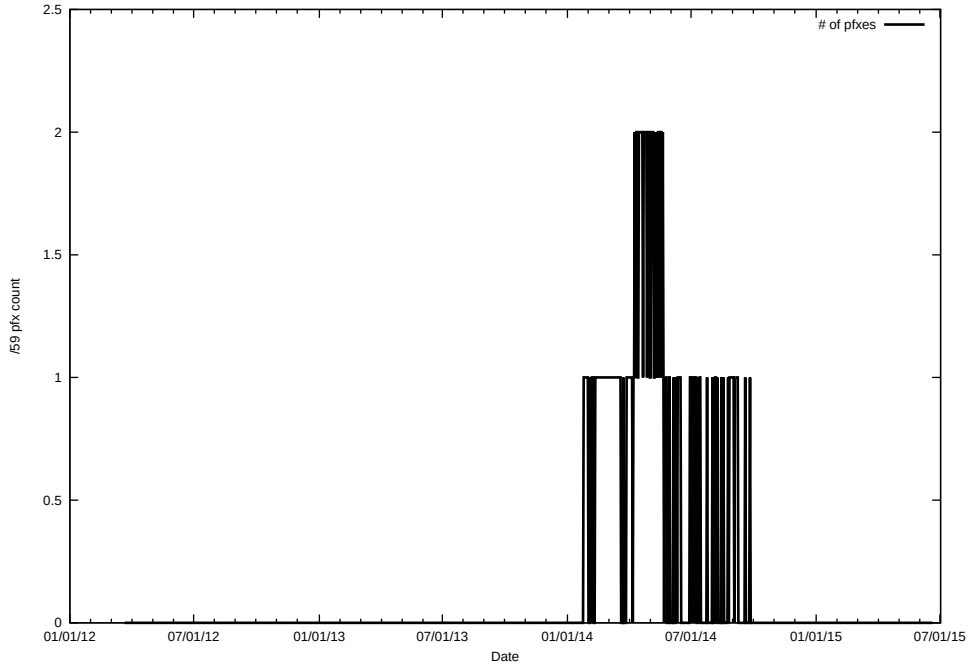
/56


1.5
1
0.5
0
-0.5
-1
/57 pfx count
# of pfxes
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

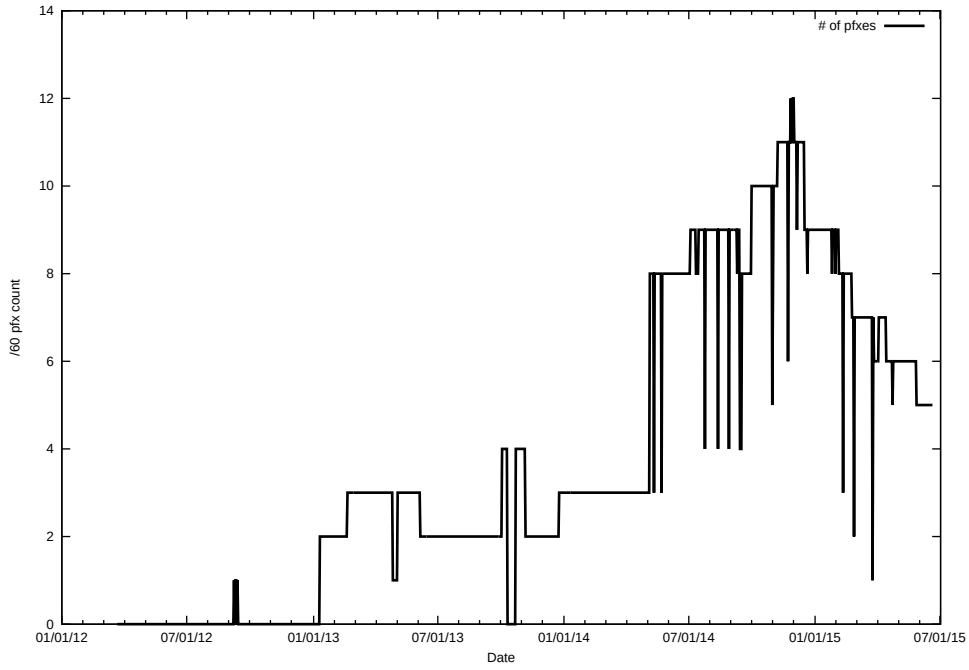
/57


2.5
2
1.5
1
0.5
0
/58 pfx count
# of pfxes
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

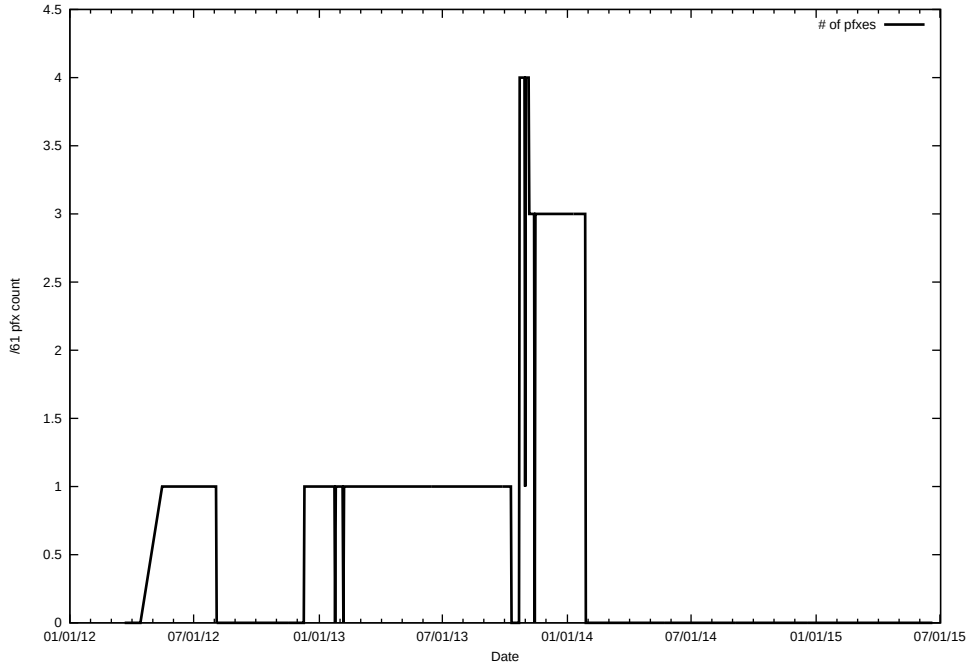
/58


/59 pfx count
Date
# of pfxes
2.5
2
1.5
1
0.5
0
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15

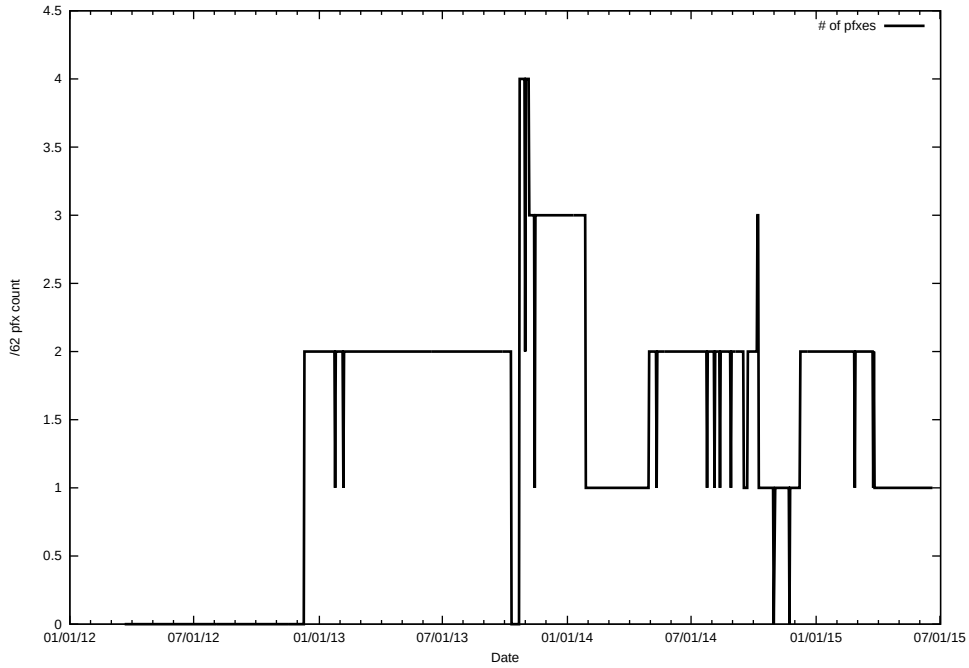
/59


/60 pfx count
Date
# of pfxes
14
12
10
8
6
4
2
0
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15

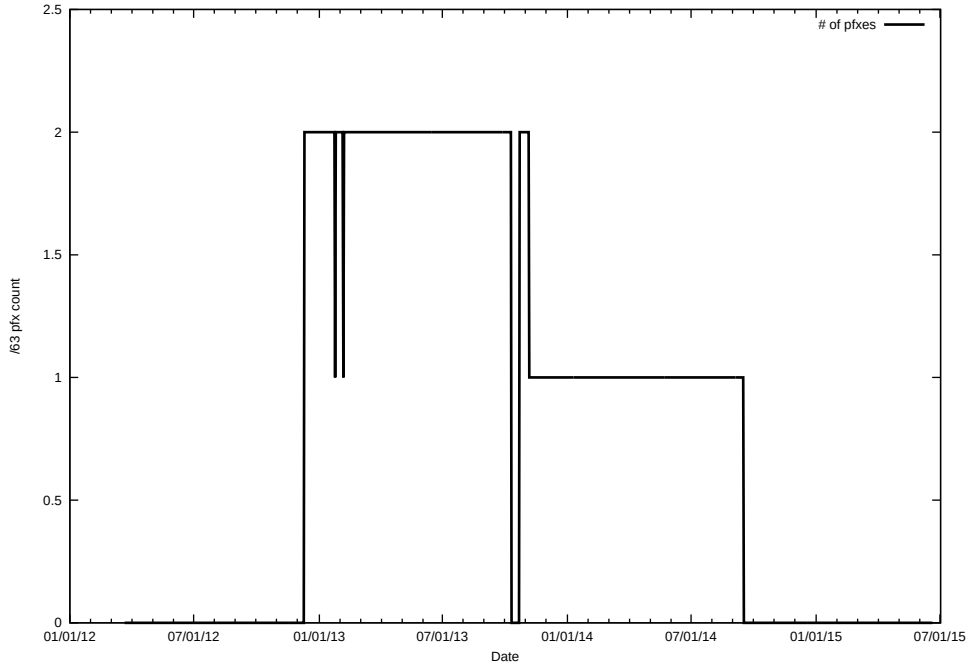
/60


/61 pfx count
Date
# of pfxes
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15

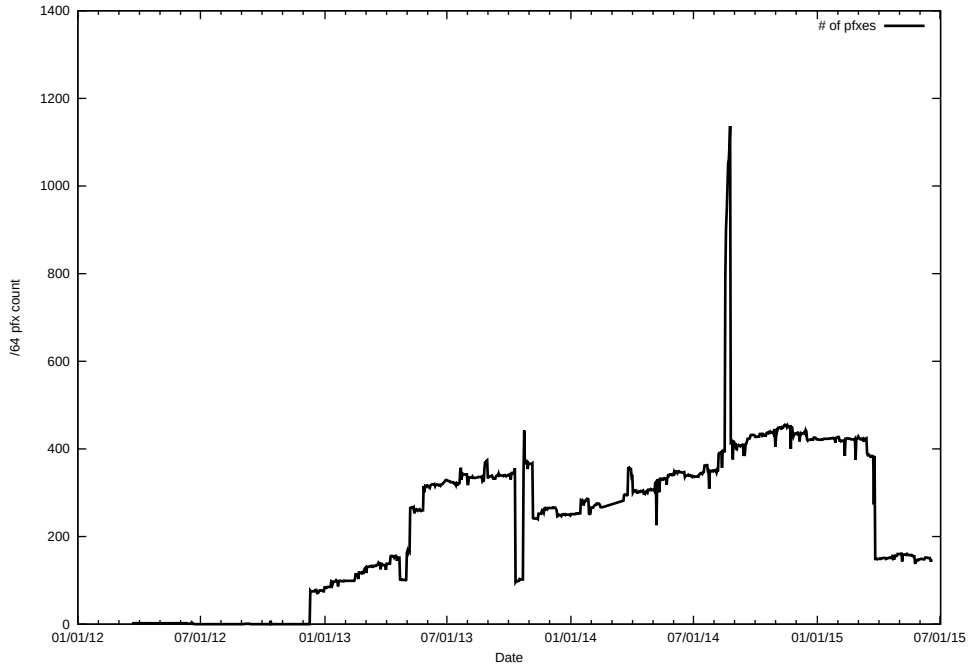
/61


/62 pfx count
Date
# of pfxes
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15

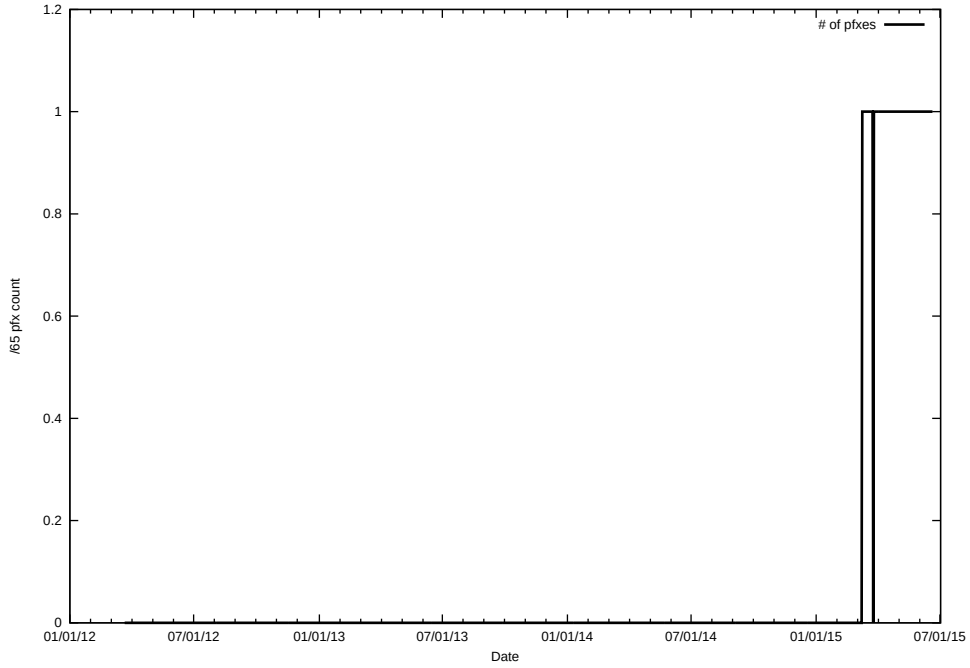
/62


/63 pfx count
Date
# of pfxes
2.5
2
1.5
1
0.5
0
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15

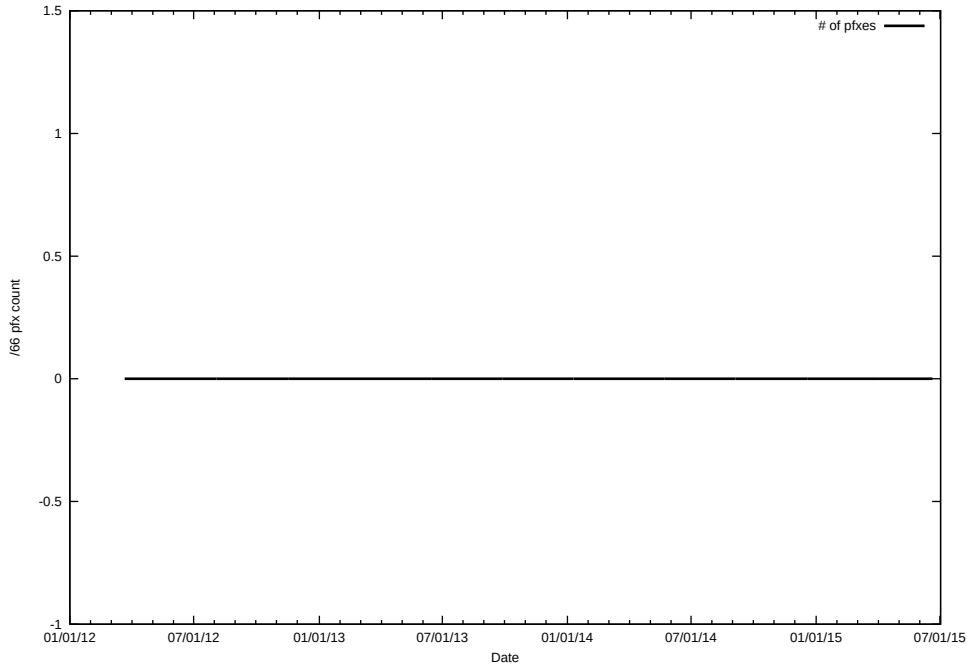
/63


/64 pfx count
Date
# of pfxes
1400
1200
1000
800
600
400
200
0
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15

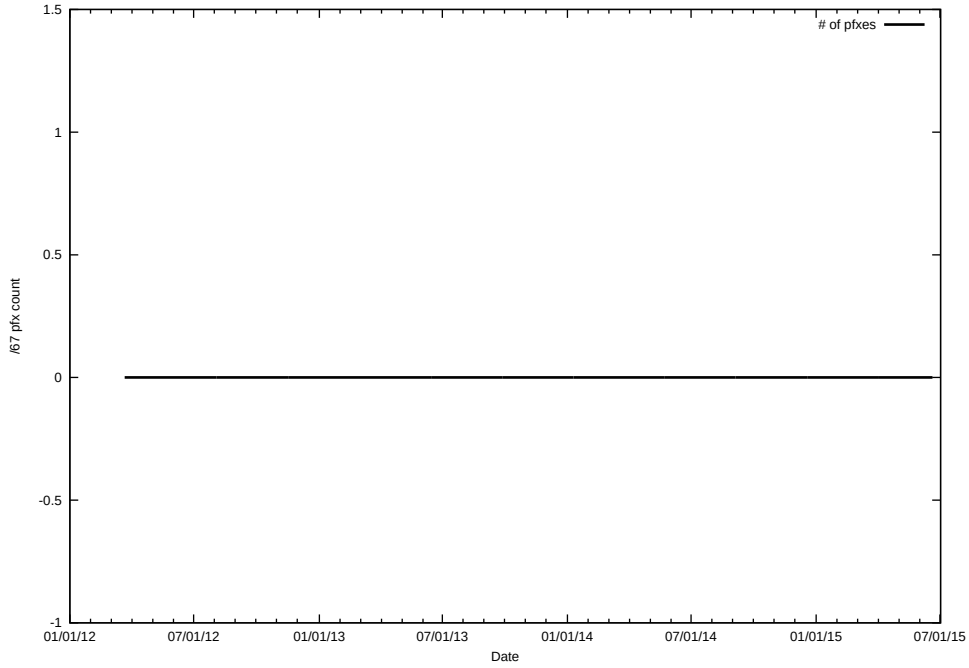
/64


1.2
1
0.8
0.6
0.4
0.2
0
/65 pfx count
# of pfxes
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

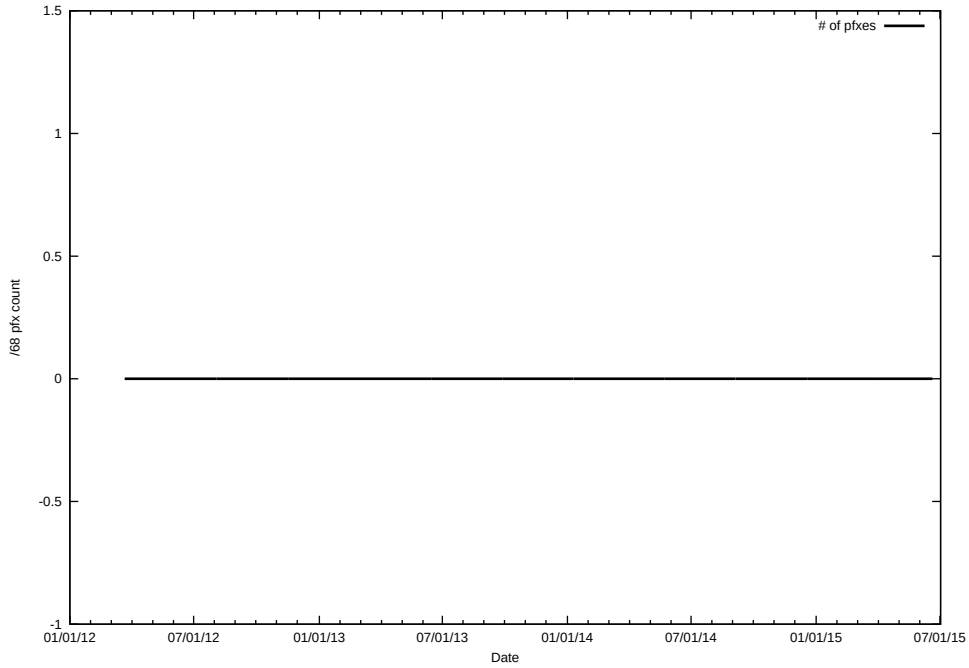
/65


1.5
1
0.5
0
-0.5
-1
/66 pfx count
# of pfxes
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

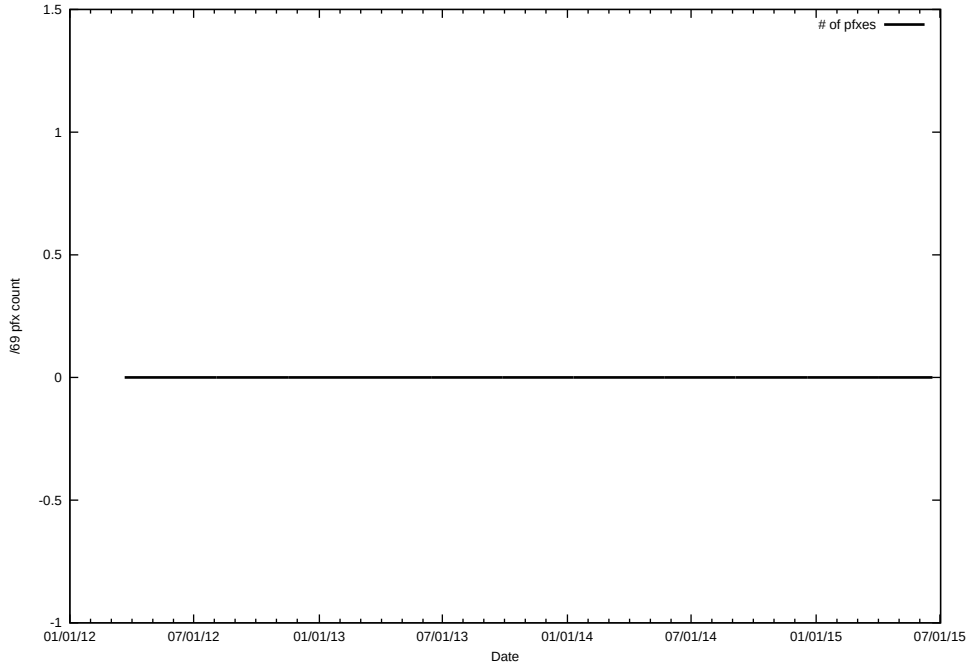
/66


1.5
1
0.5
0
-0.5
-1
/67 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

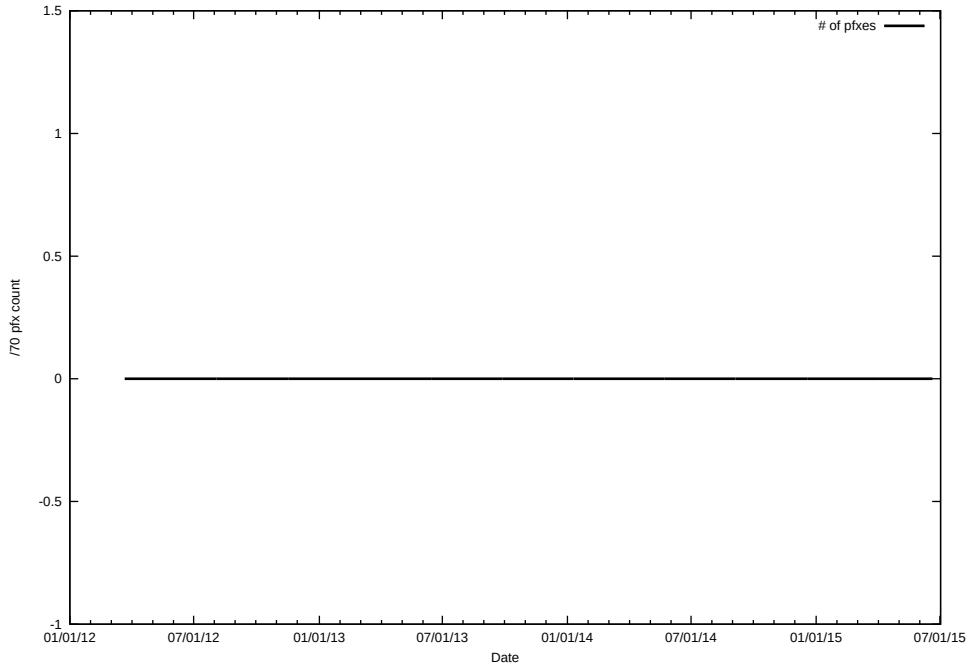
/67


1.5
1
0.5
0
-0.5
-1
/68 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

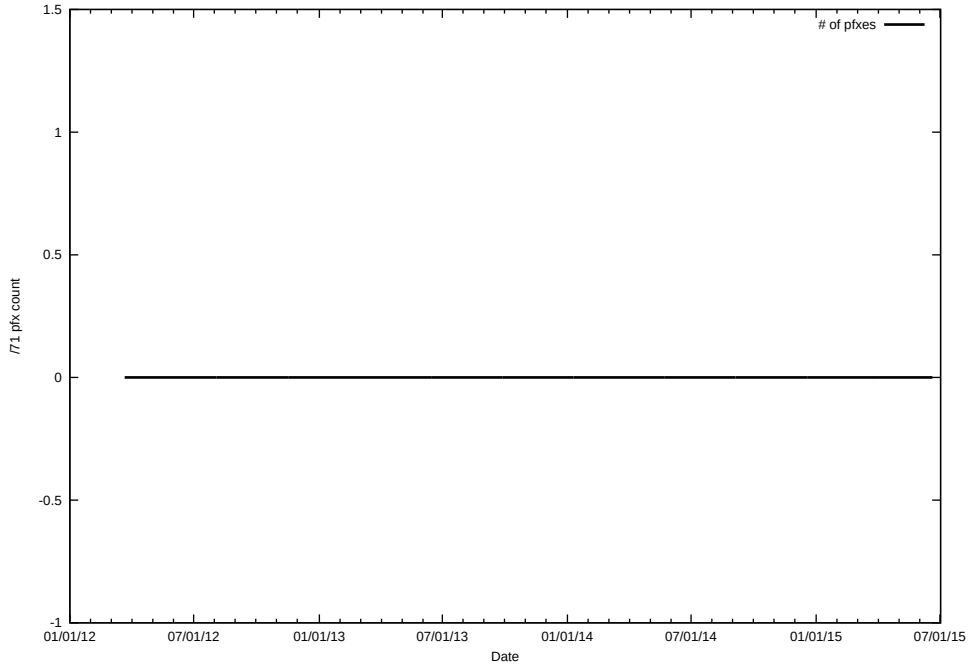
/68


1.5
1
0.5
0
-0.5
-1
/69 pfx count
# of pfxes
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

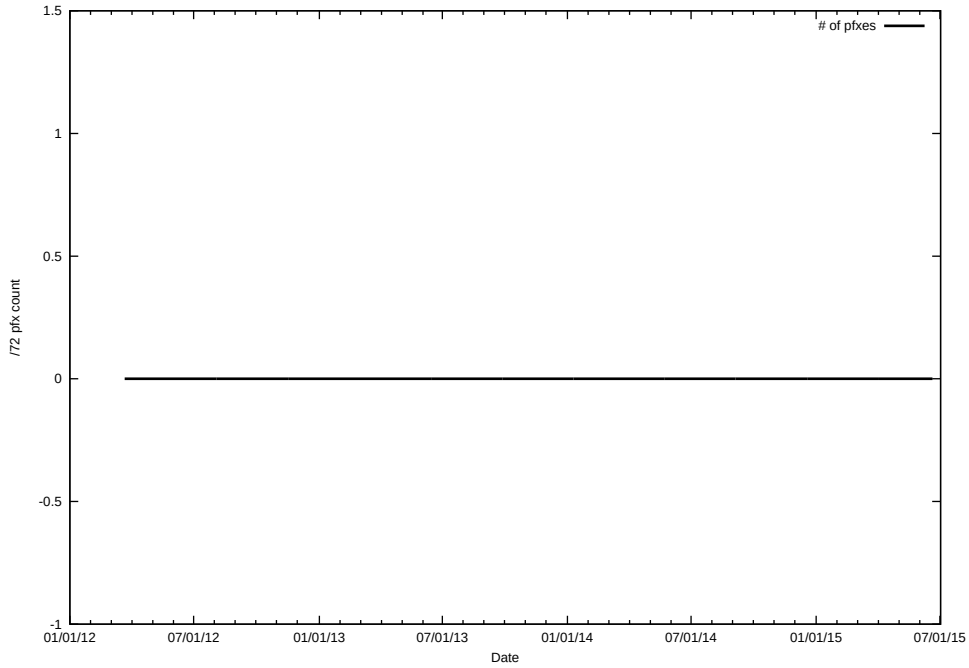
/69


1.5
1
0.5
0
-0.5
-1
/70 pfx count
# of pfxes
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

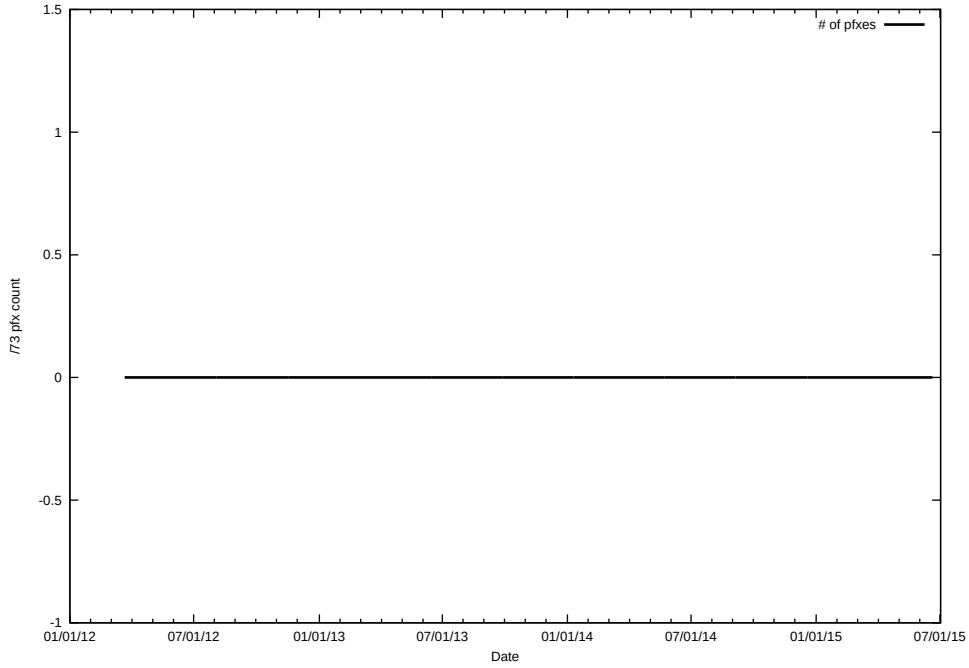
/70


1.5
1
0.5
0
-0.5
-1
/71 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

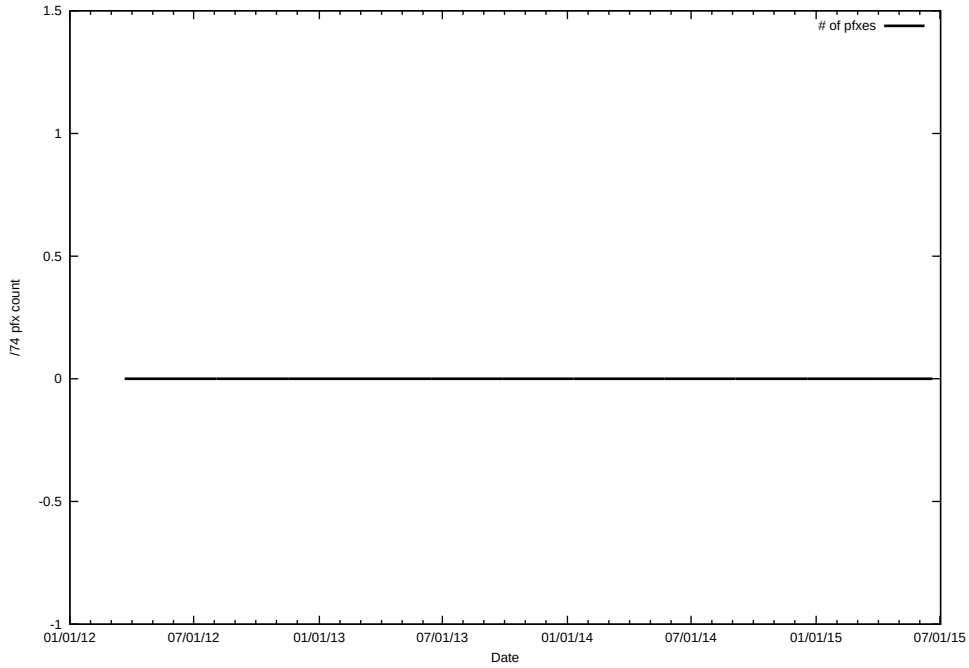
/71


1.5
1
0.5
0
-0.5
-1
/72 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

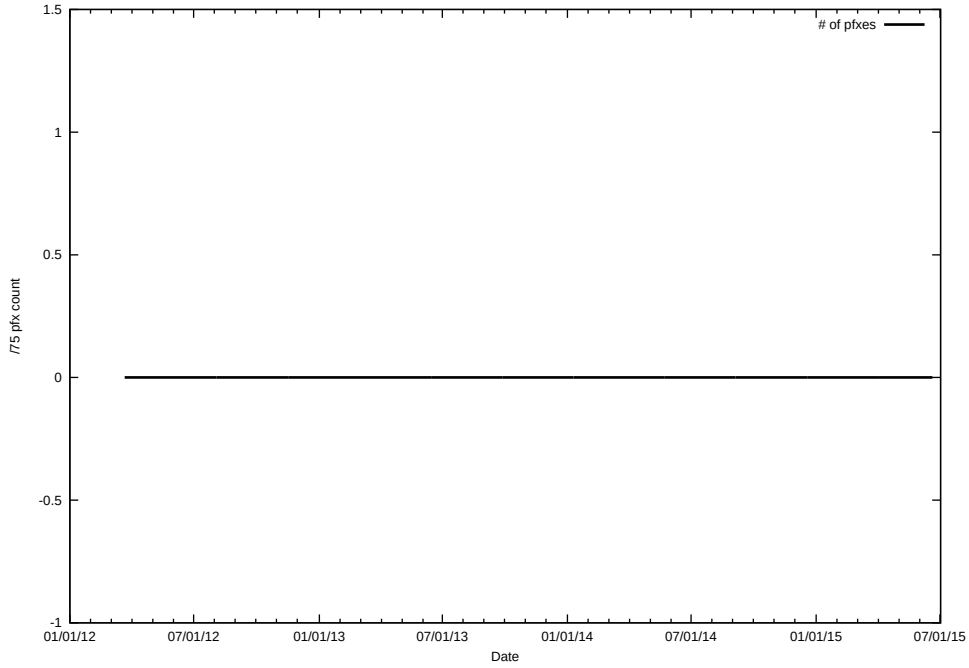
/72


1.5
1
0.5
0
-0.5
-1
/73 pfx count
# of pfxes
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

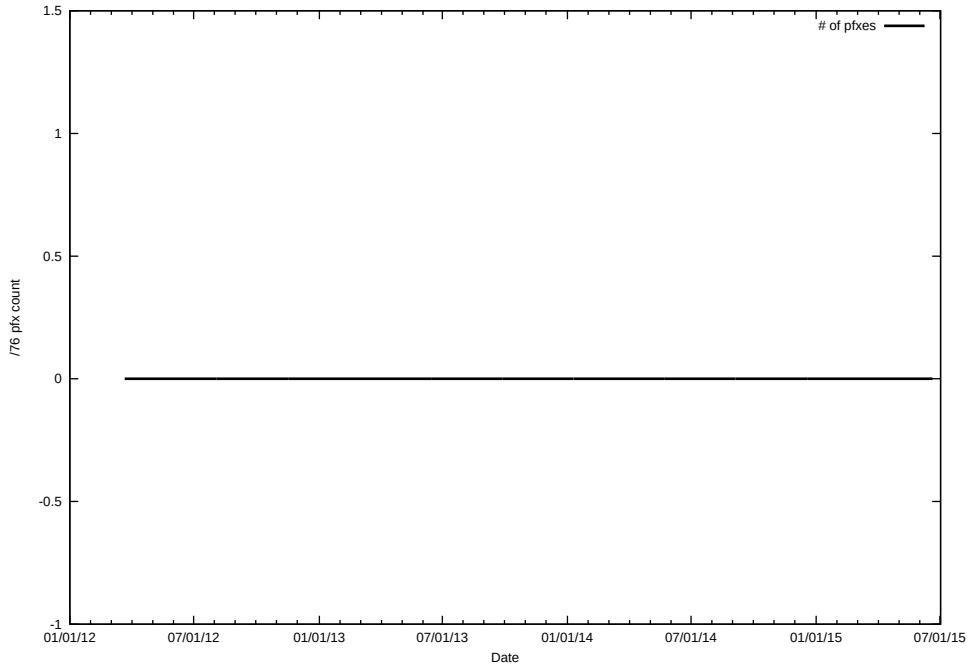
/73


1.5
1
0.5
0
-0.5
-1
/74 pfx count
# of pfxes
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

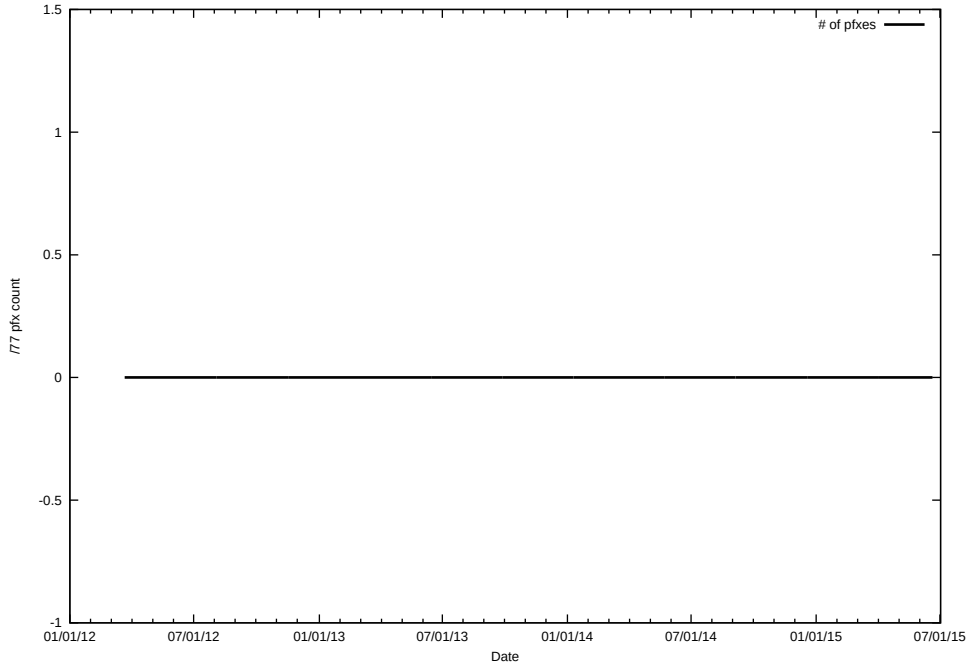
/74


1.5
1
0.5
0
-0.5
-1
/75 pfx count
# of pfxes
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

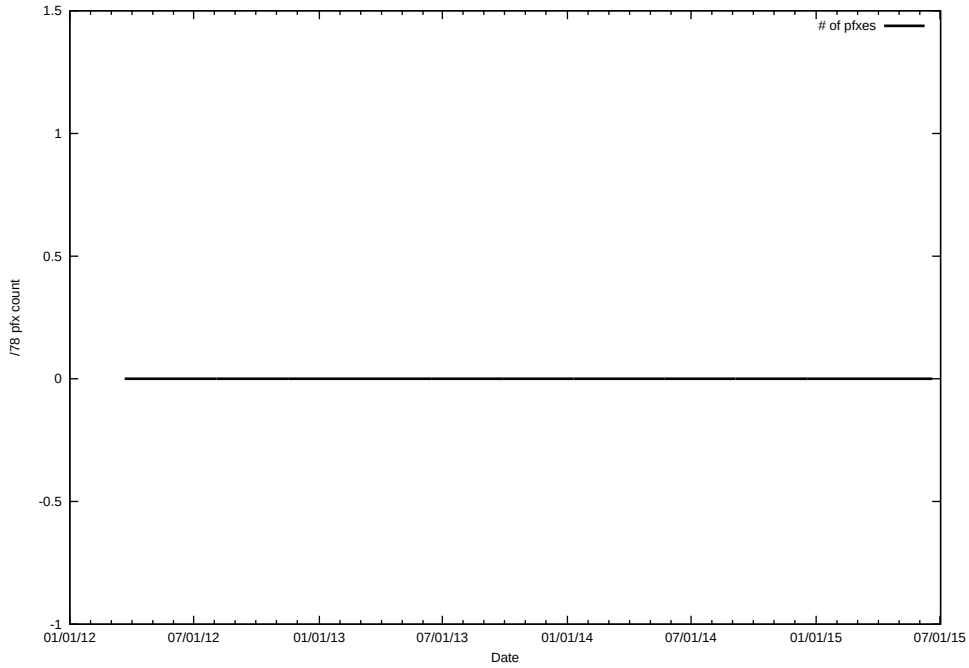
/75


1.5
1
0.5
0
-0.5
-1
/76 pfx count
# of pfxes
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

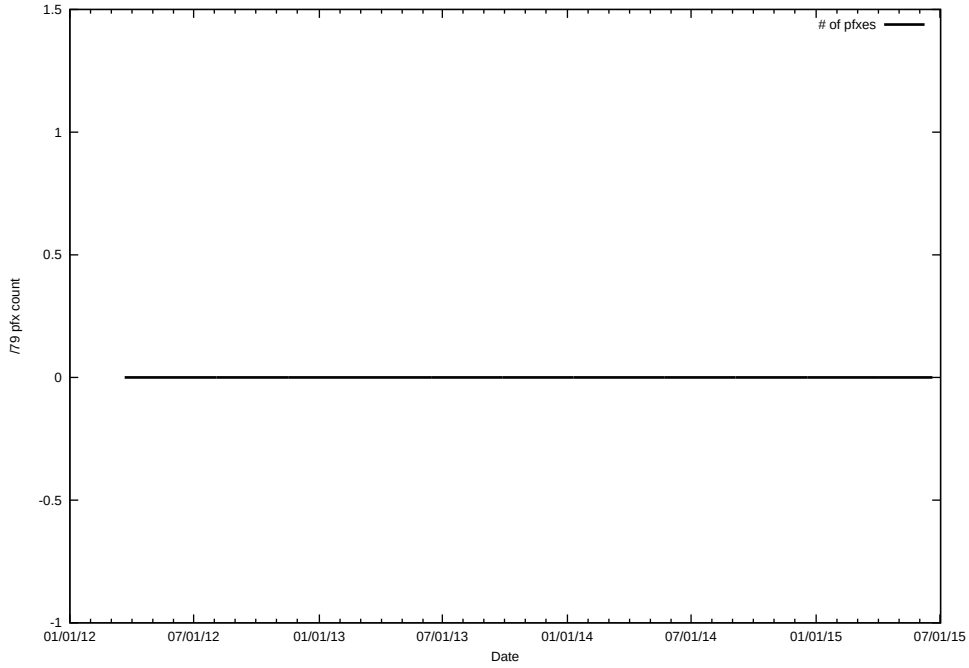
/76


1.5
1
0.5
0
-0.5
-1
/77 pfx count
# of pfxes
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

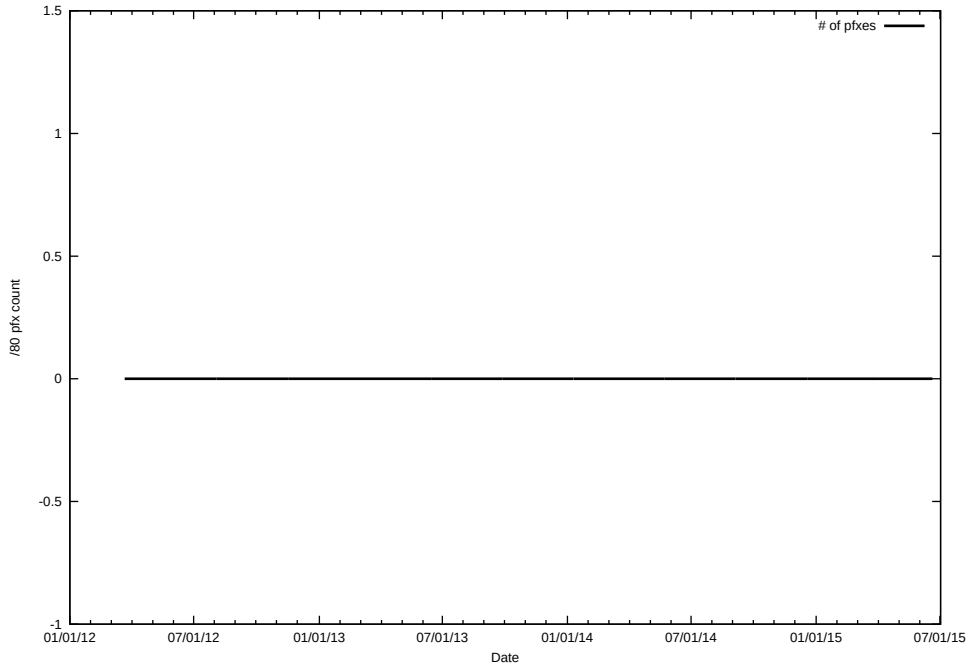
/77


1.5
1
0.5
0
-0.5
-1
/78 pfx count
# of pfxes
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

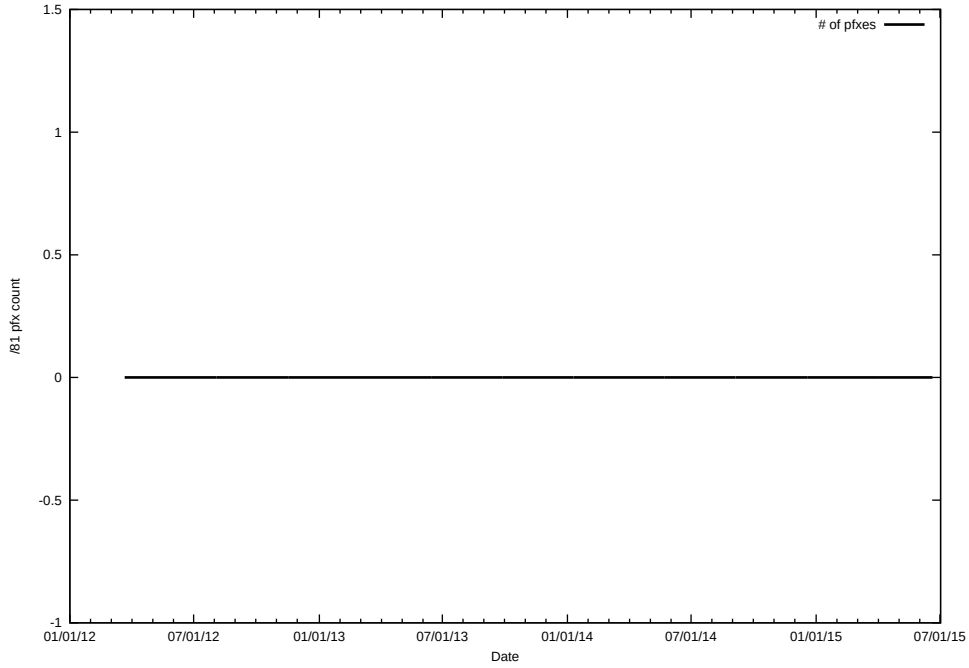
/78


1.5
1
0.5
0
-0.5
-1
/79 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

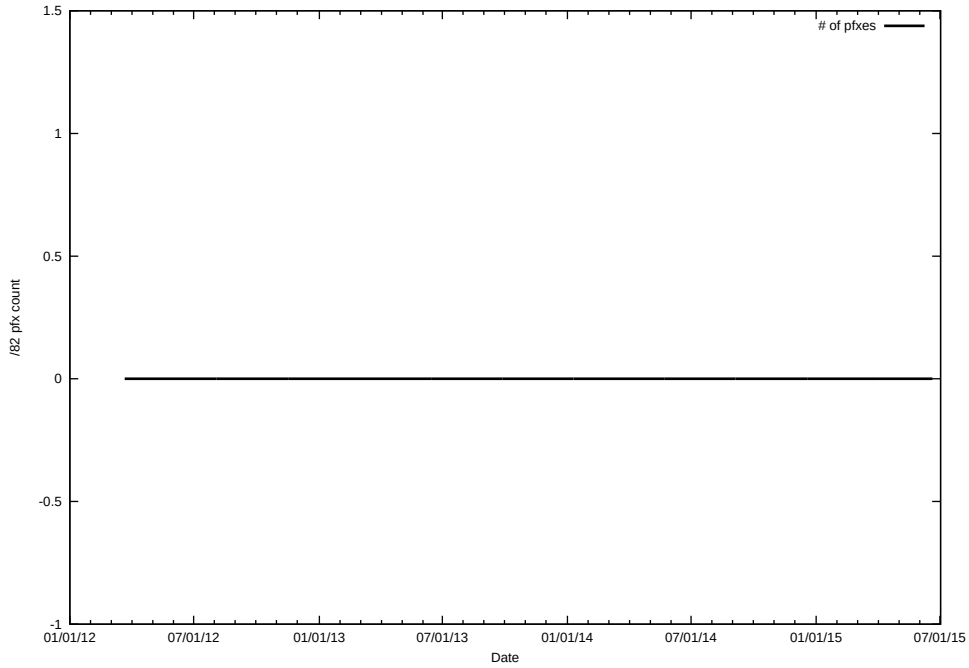
/79


1.5
1
0.5
0
-0.5
-1
/80 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

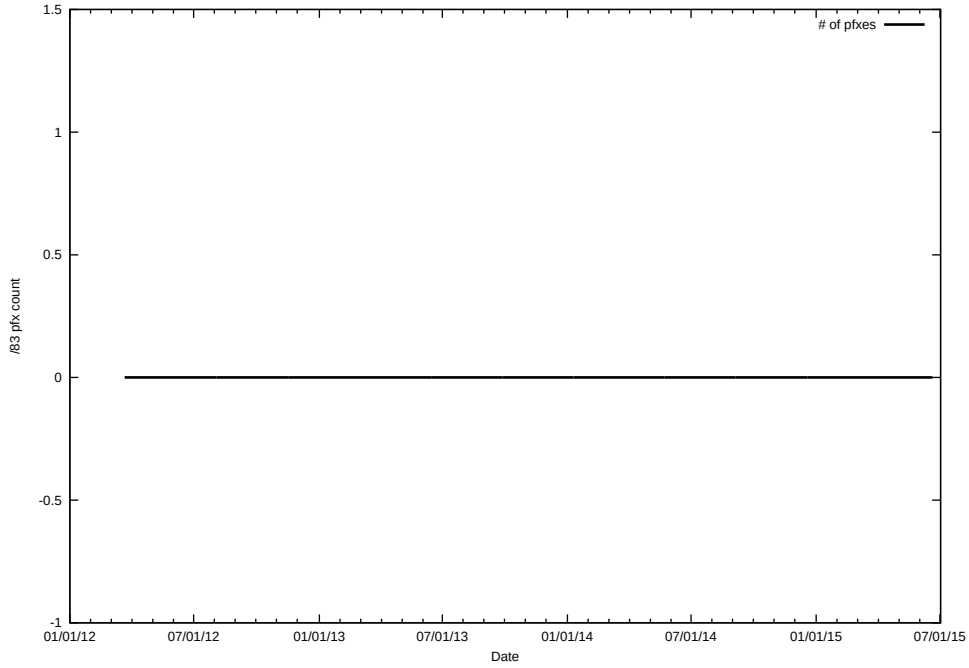
/80


1.5
1
0.5
0
-0.5
-1
/81 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

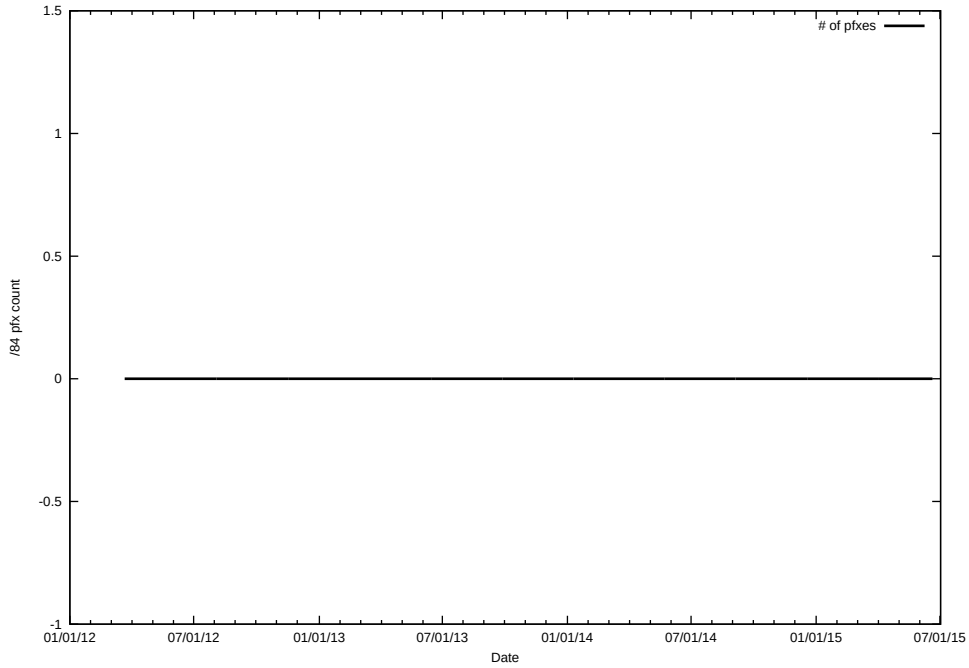
/81


1.5
1
0.5
0
-0.5
-1
/82 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

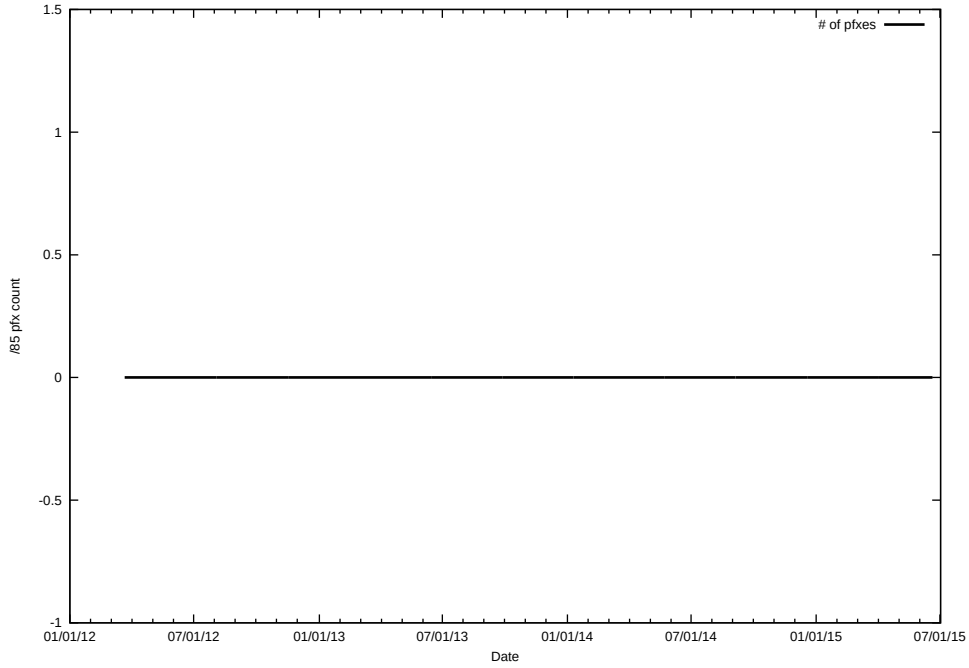
/82


1.5
1
0.5
0
-0.5
-1
/83 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

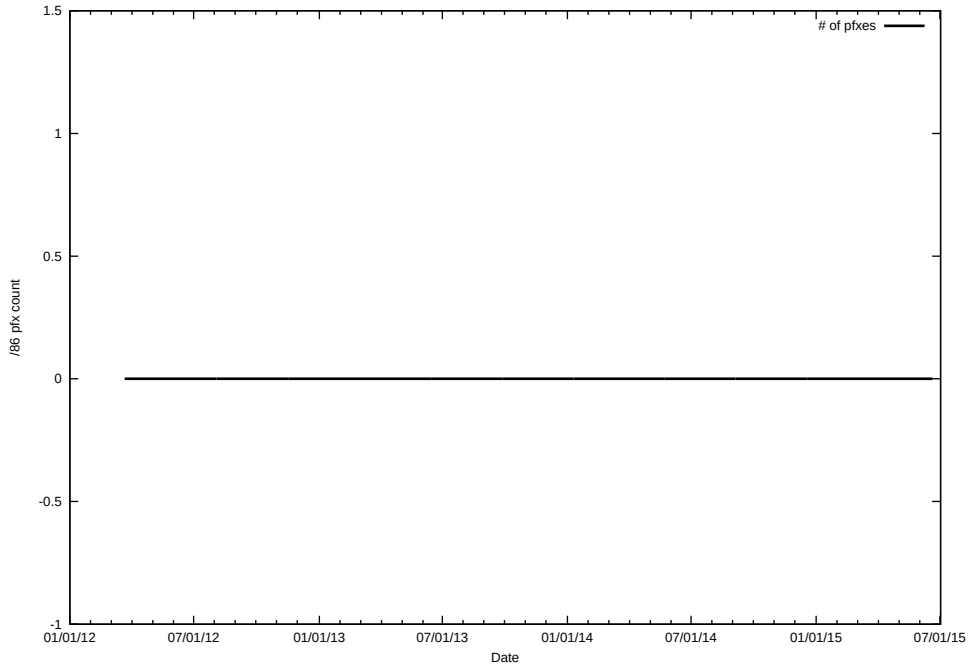
/83


1.5
1
0.5
0
-0.5
-1
/84 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

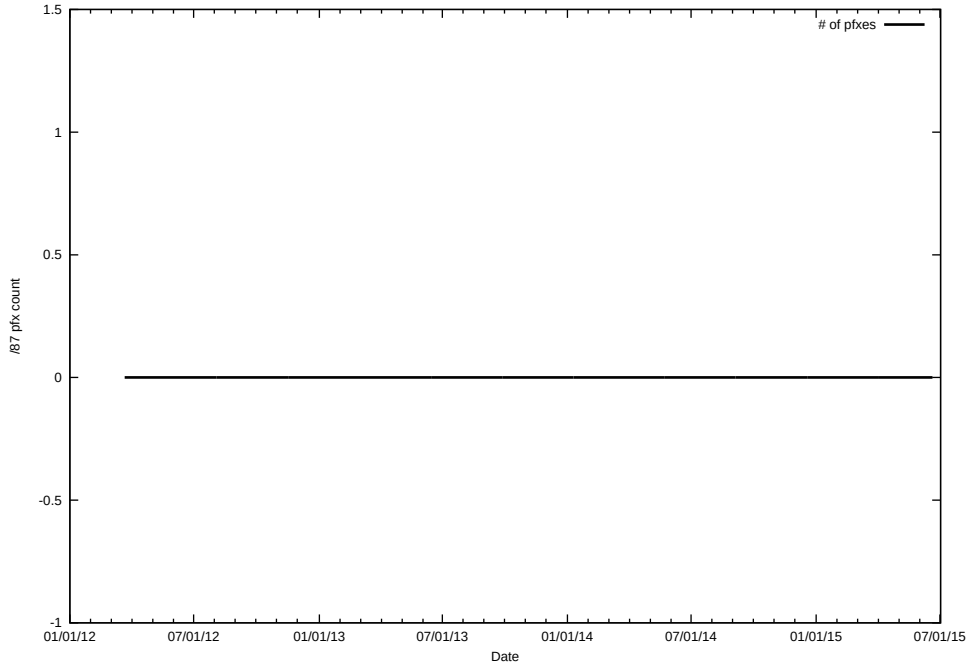
/84


1.5
1
0.5
0
-0.5
-1
/85 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

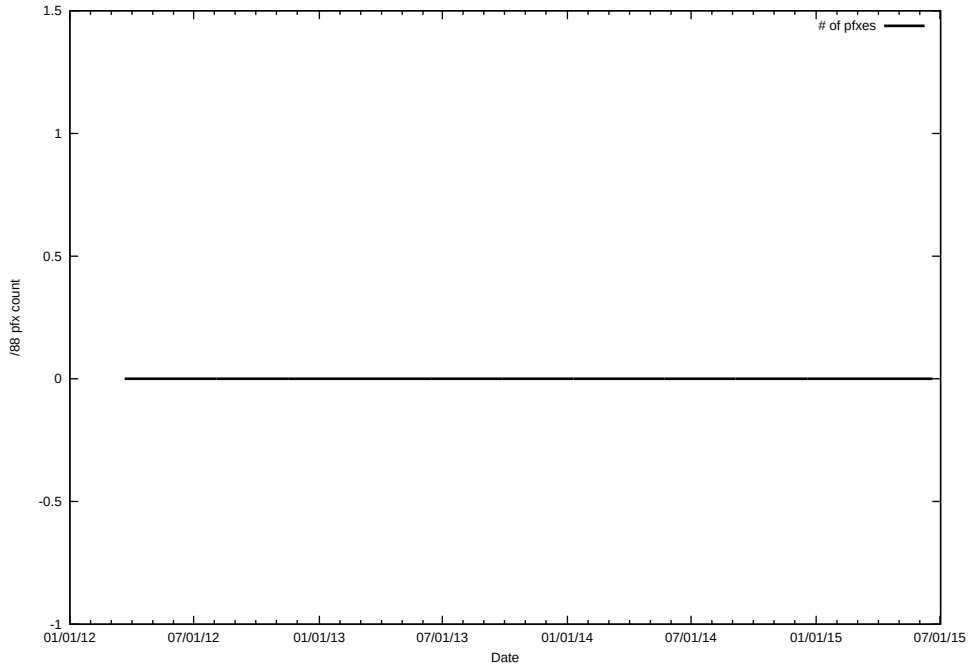
/85


1.5
1
0.5
0
-0.5
-1
/86 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

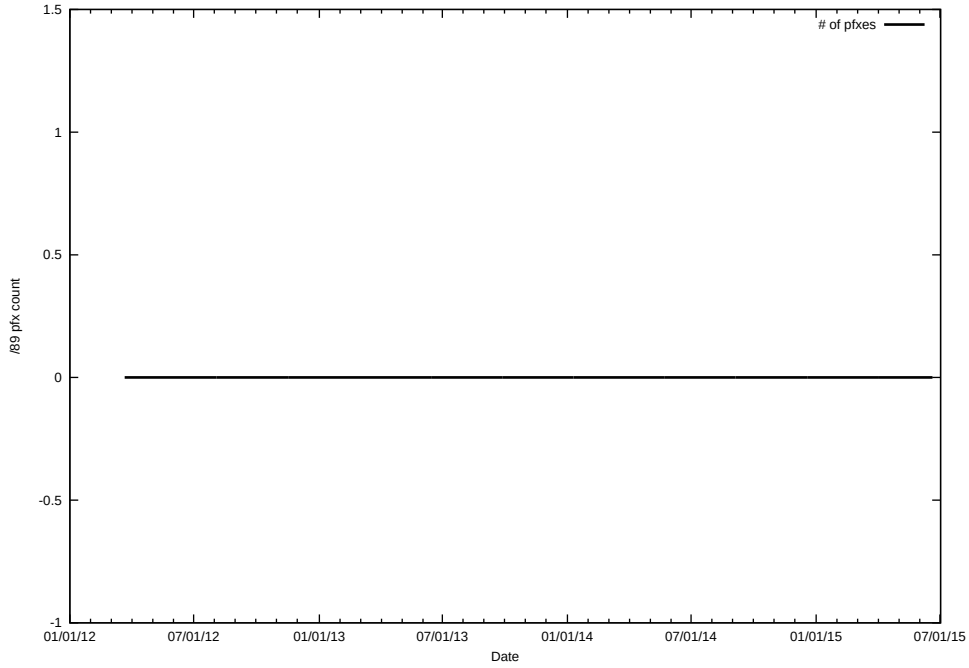
/86


1.5
1
0.5
0
-0.5
-1
/87 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

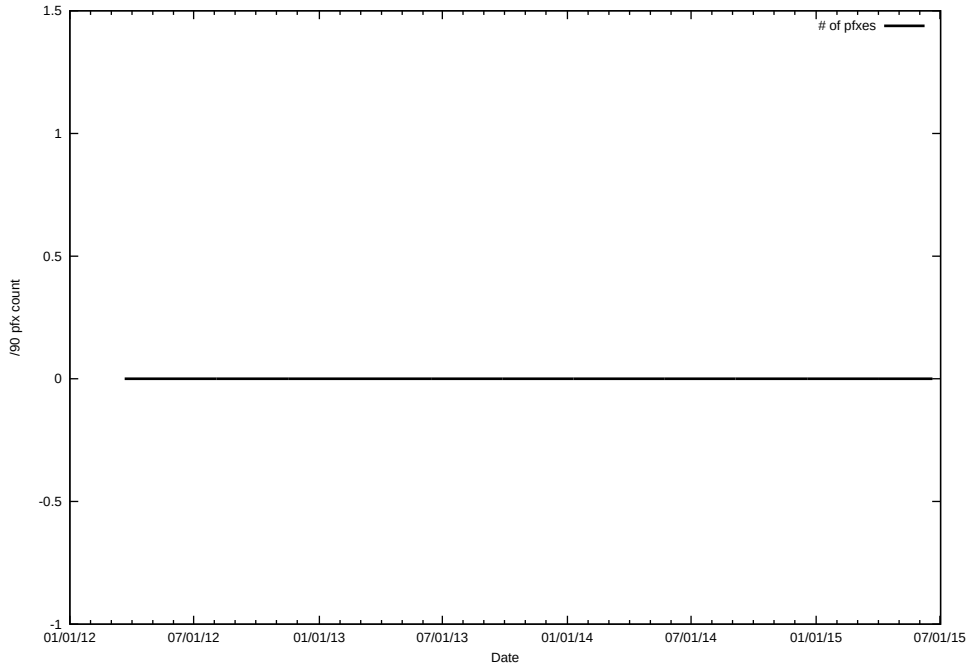
/87


1.5
1
0.5
0
-0.5
-1
/88 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

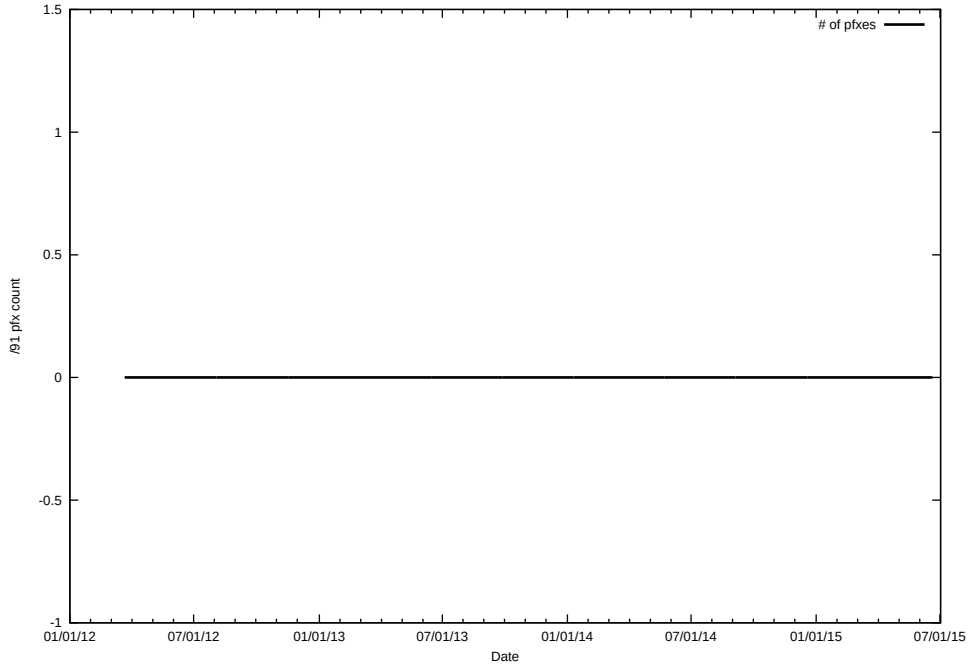
/88


1.5
1
0.5
0
-0.5
-1
/89 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

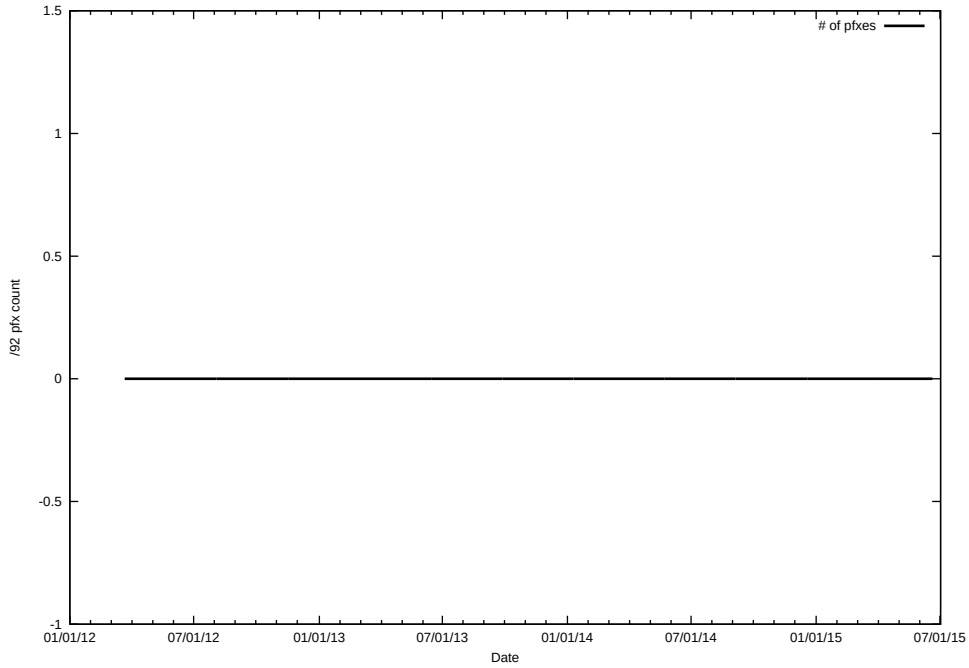
/89


1.5
1
0.5
0
-0.5
-1
/90 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

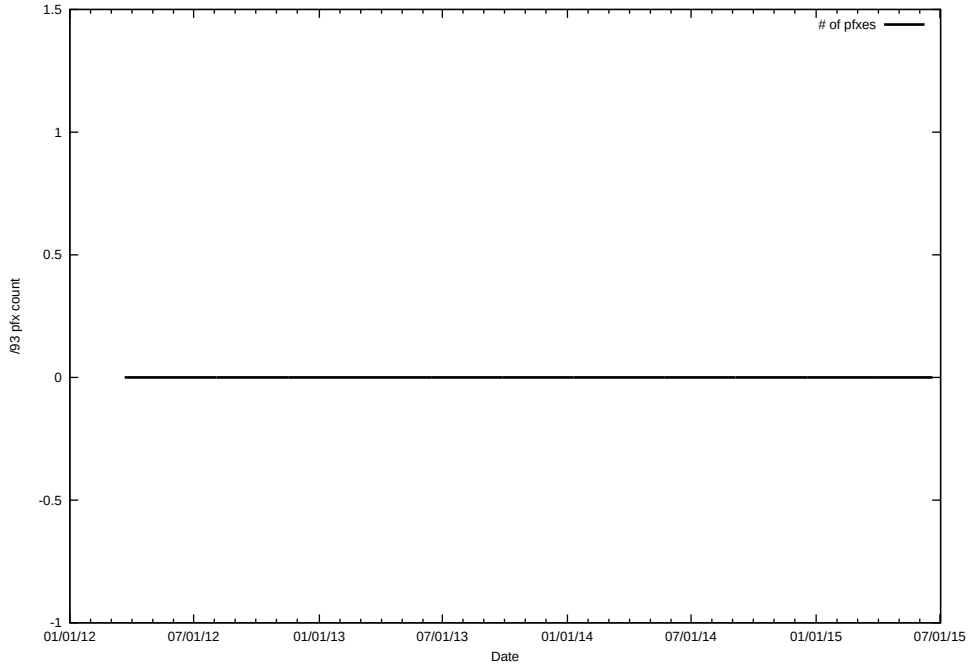
/90


1.5
1
0.5
0
-0.5
-1
/91 pfx count
# of pfxes
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

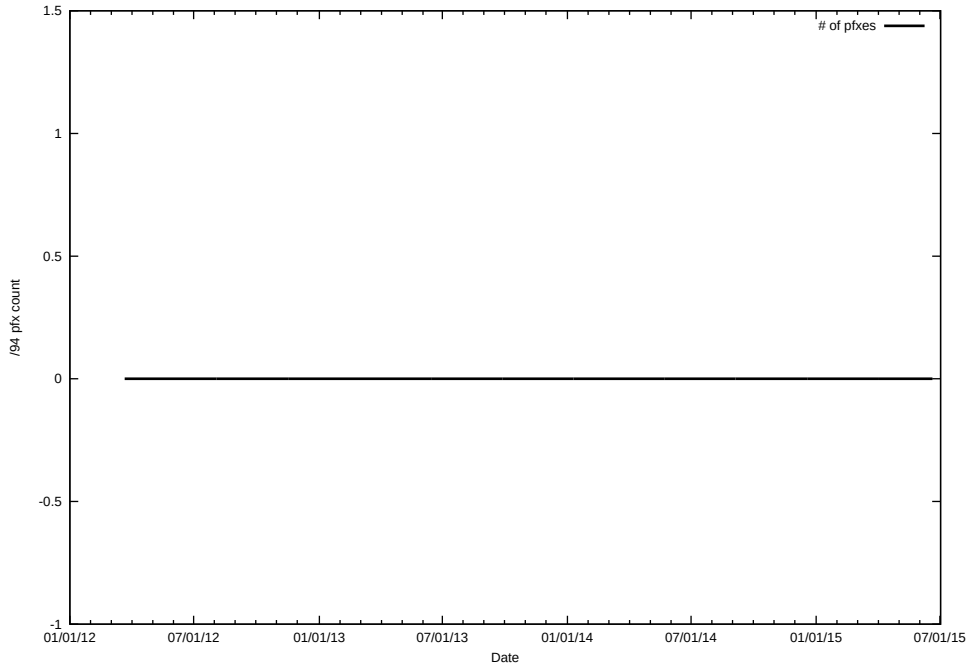
/91


1.5
1
0.5
0
-0.5
-1
/92 pfx count
# of pfxes
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

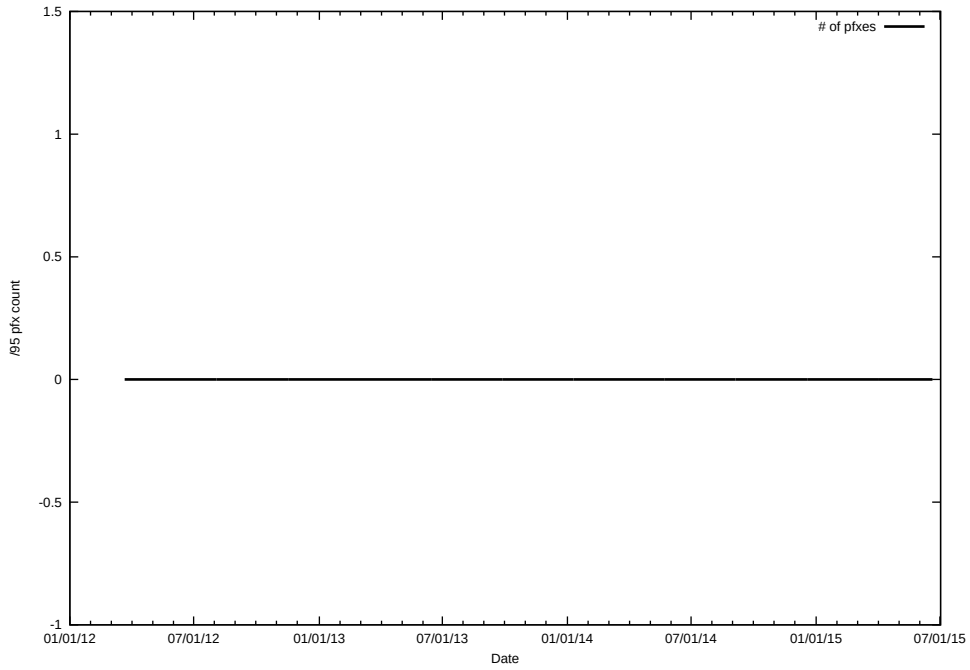
/92


1.5
1
0.5
0
-0.5
-1
/93 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

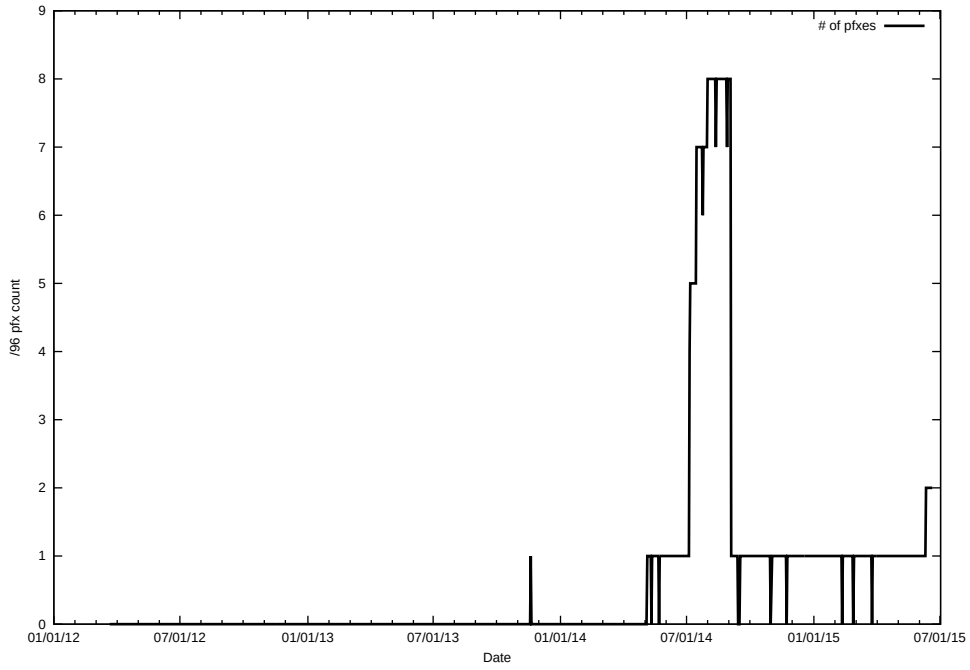
/93


1.5
1
0.5
0
-0.5
-1
/94 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

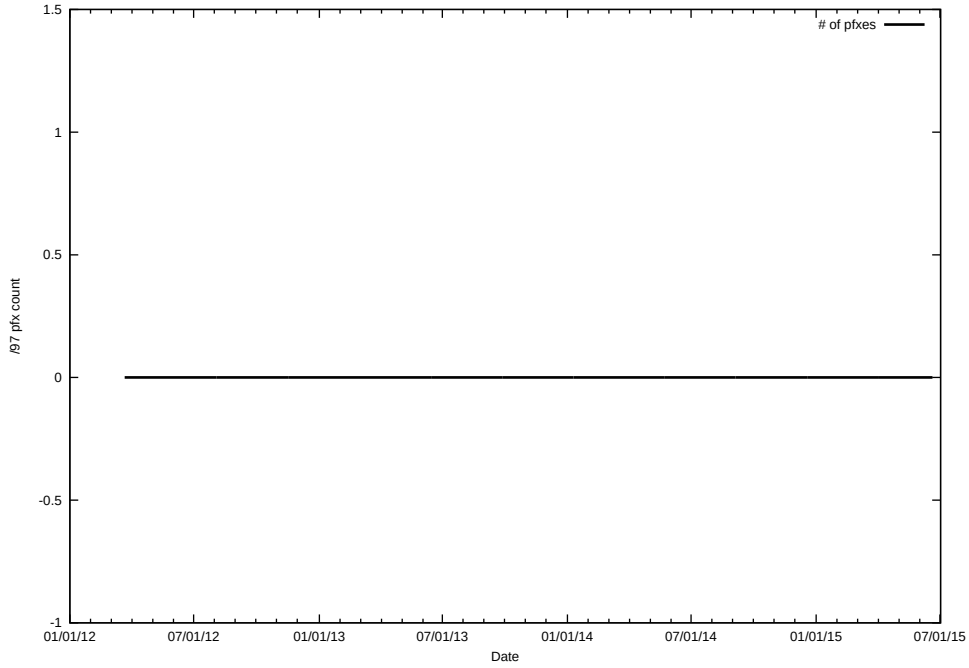
/94


1.5
1
0.5
0
-0.5
-1
/95 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

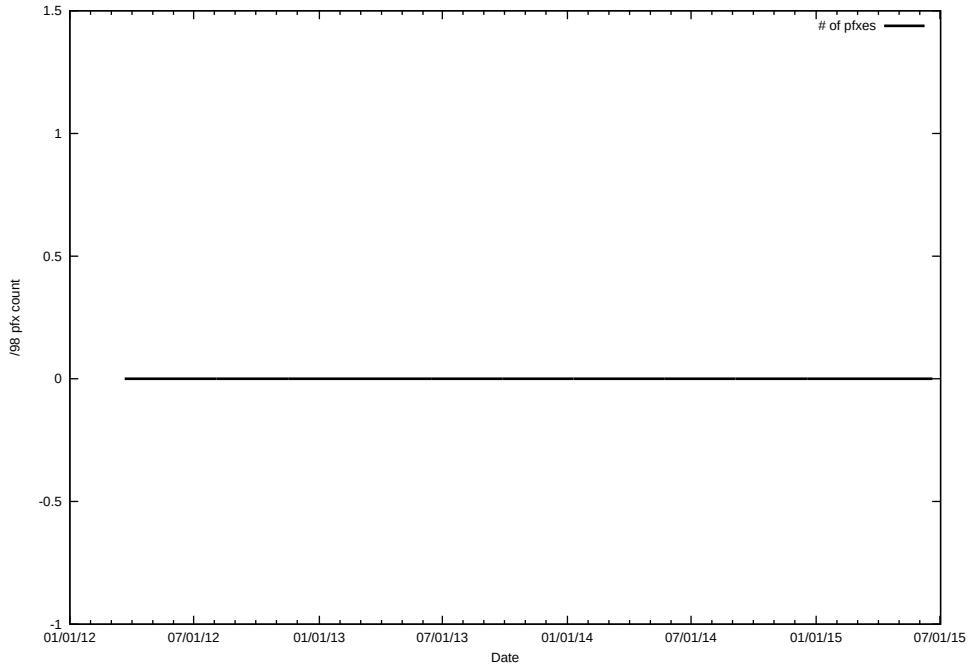
/95


9
8
7
6
5
4
3
2
1
0
/96 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

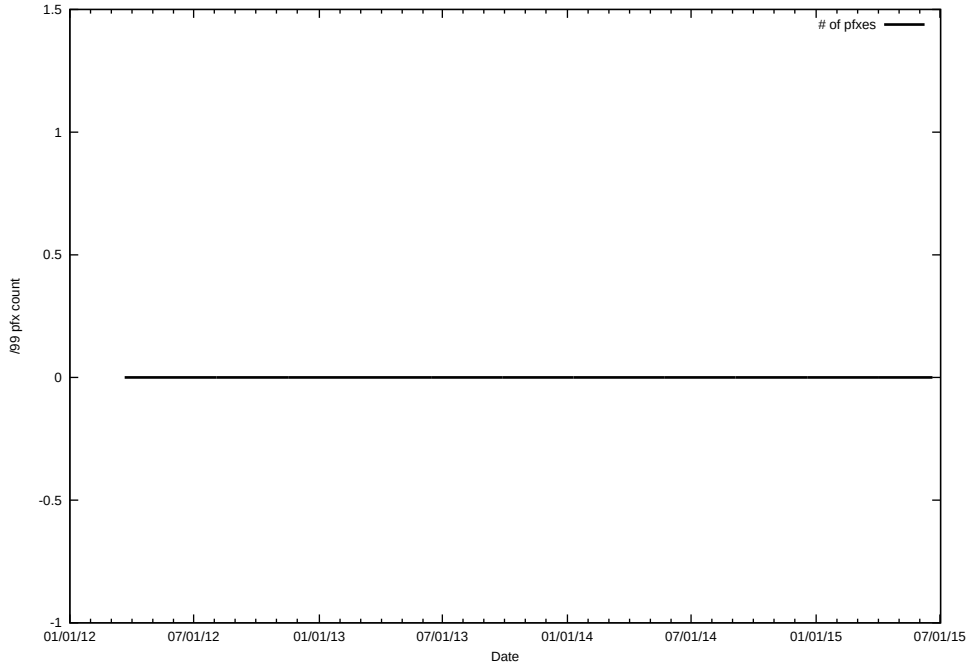
/96


1.5
1
0.5
0
-0.5
-1
/97 pfx count
# of pfxes
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

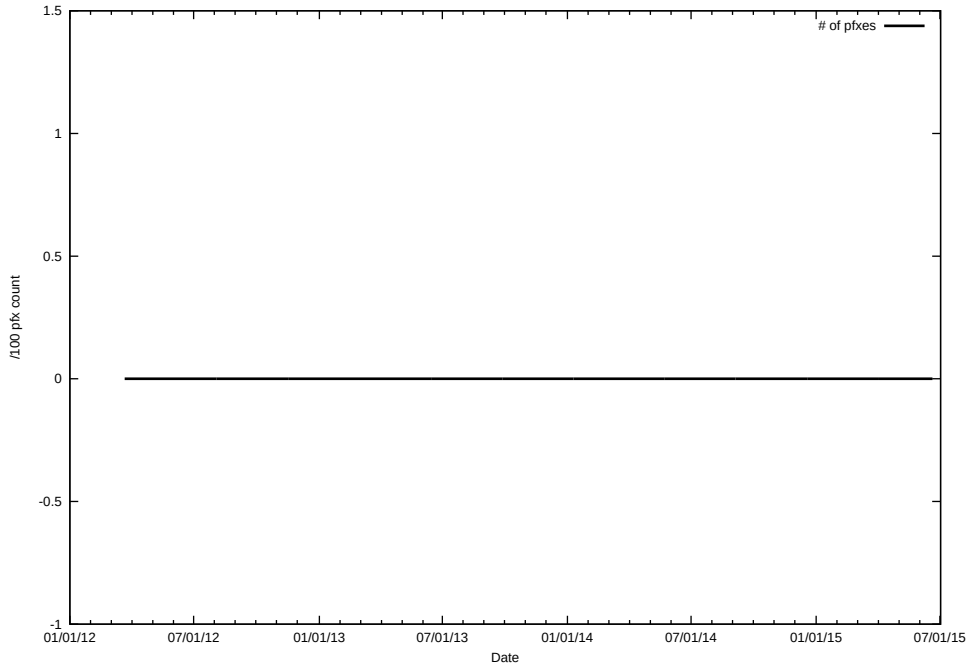
/97


1.5
1
0.5
0
-0.5
-1
/98 pfx count
# of pfxes
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

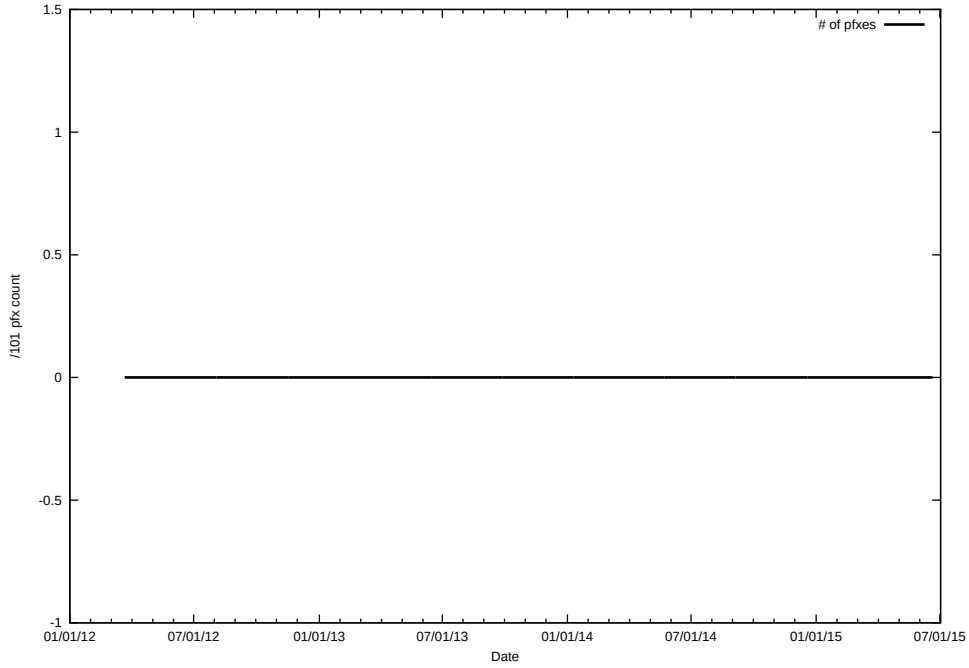
/98


1.5
1
0.5
0
-0.5
-1
/99 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

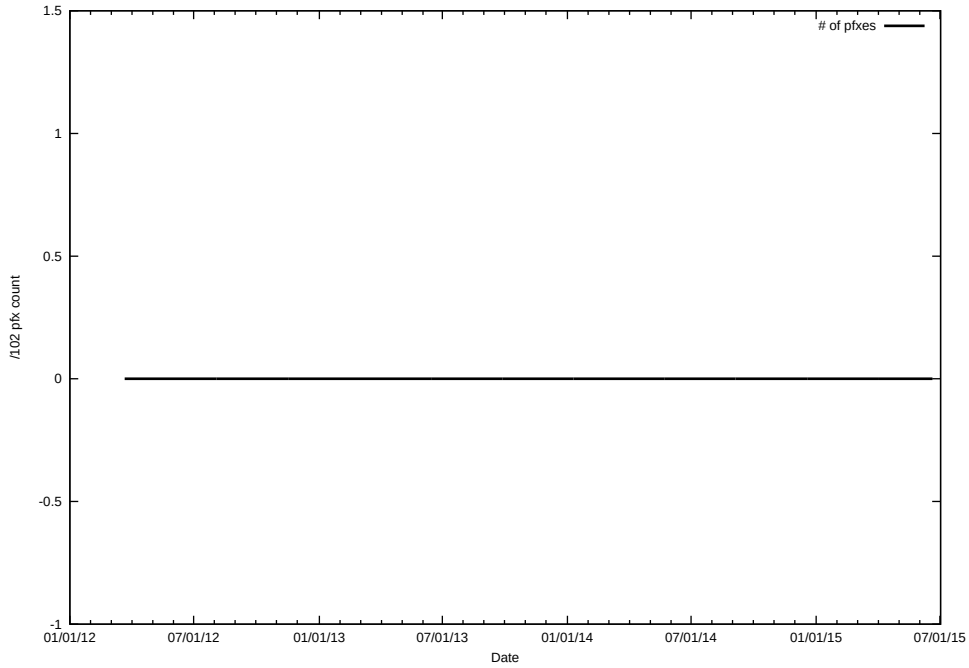
/99


1.5
1
0.5
0
-0.5
-1
/100 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

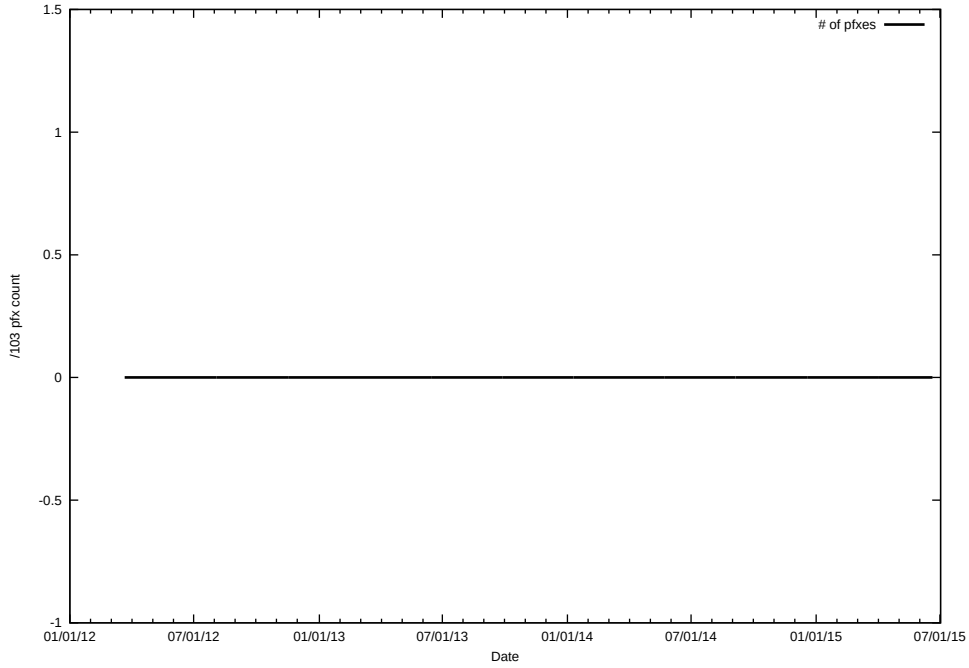
/100


1.5
1
0.5
0
-0.5
-1
/101 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

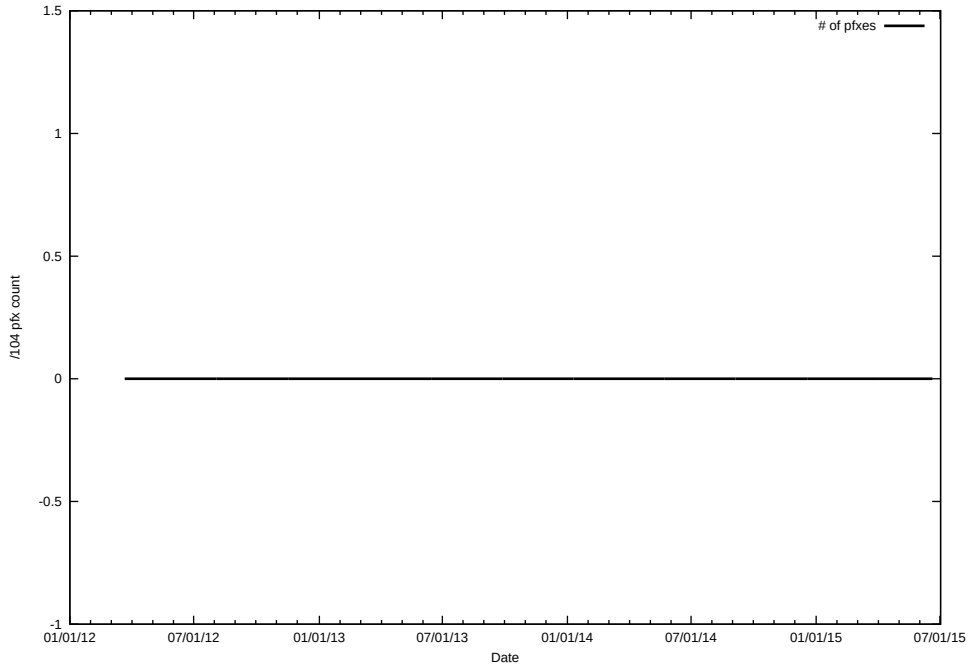
/101


1.5
1
0.5
0
-0.5
-1
/102 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

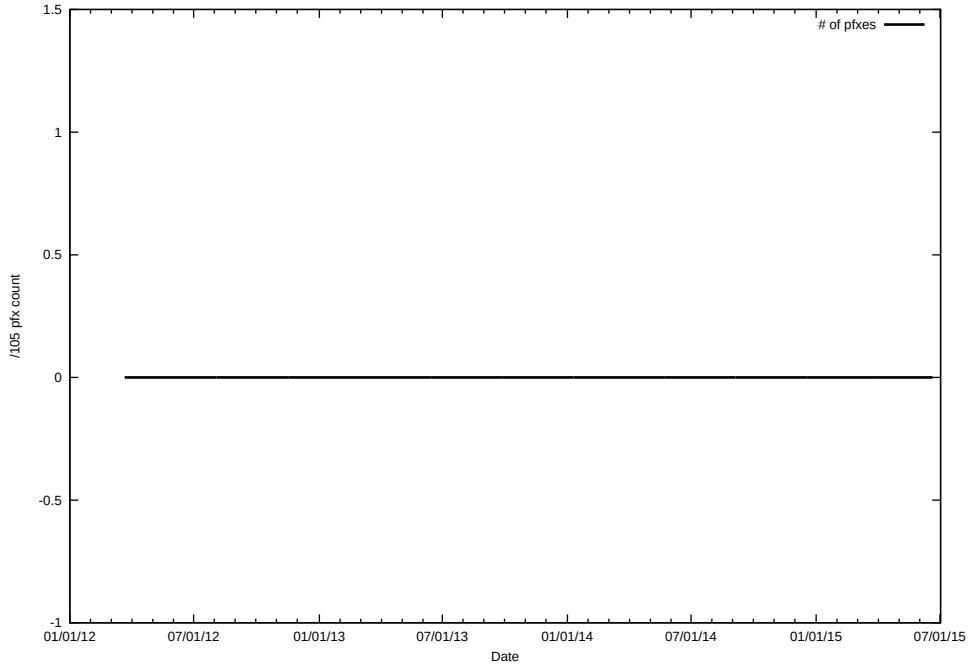
/102


1.5
1
0.5
0
-0.5
-1
/103 pfx count
# of pfxes
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

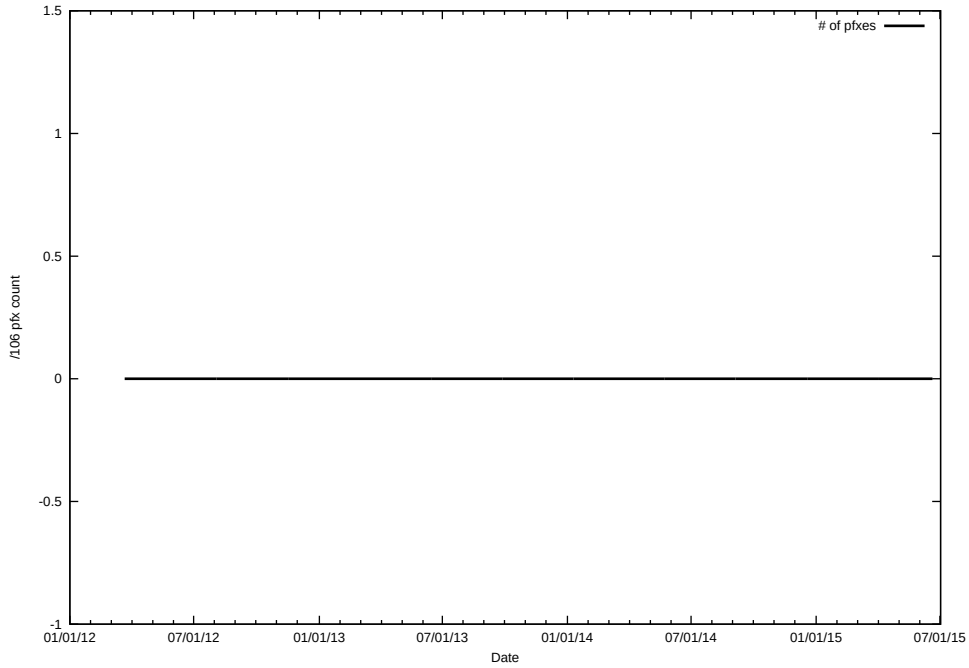
/103


1.5
1
0.5
0
-0.5
-1
/104 pfx count
# of pfxes
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

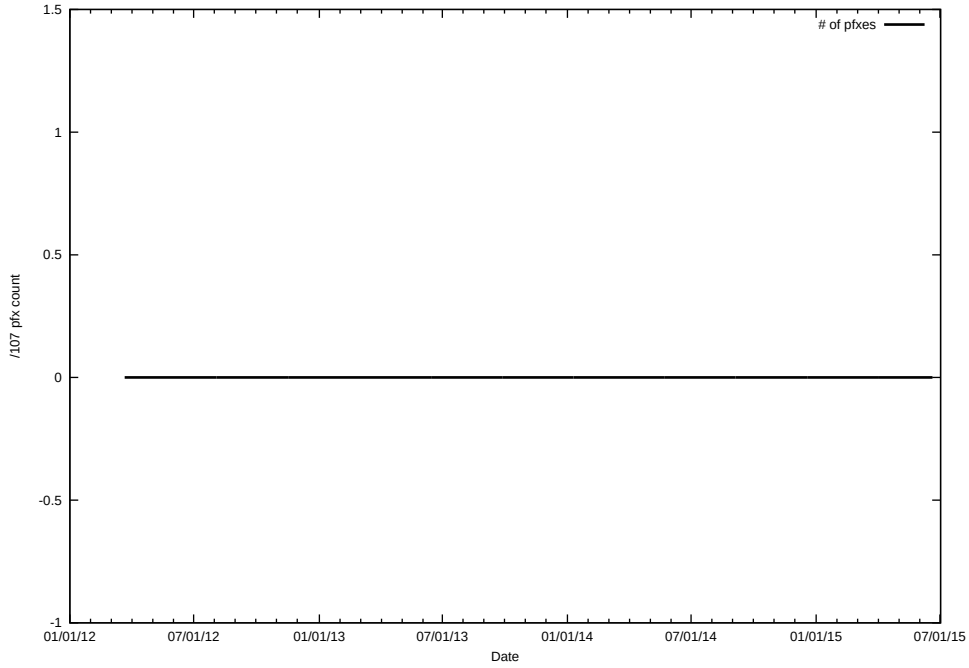
/104


1.5
1
0.5
0
-0.5
-1
/105 pfx count
# of pfxes
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

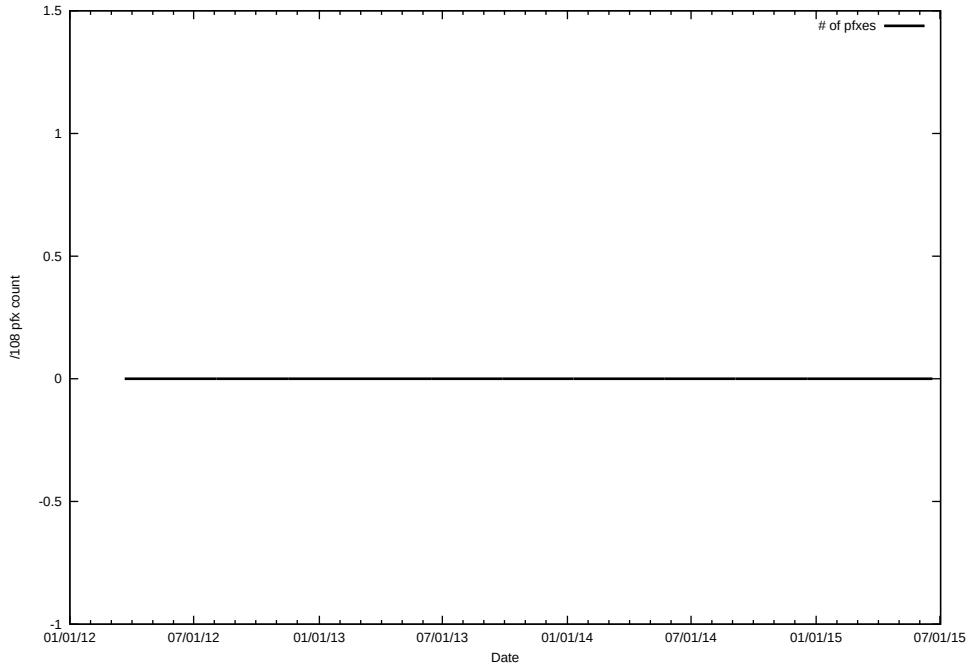
/105


1.5
1
0.5
0
-0.5
-1
/106 pfx count
# of pfxes
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

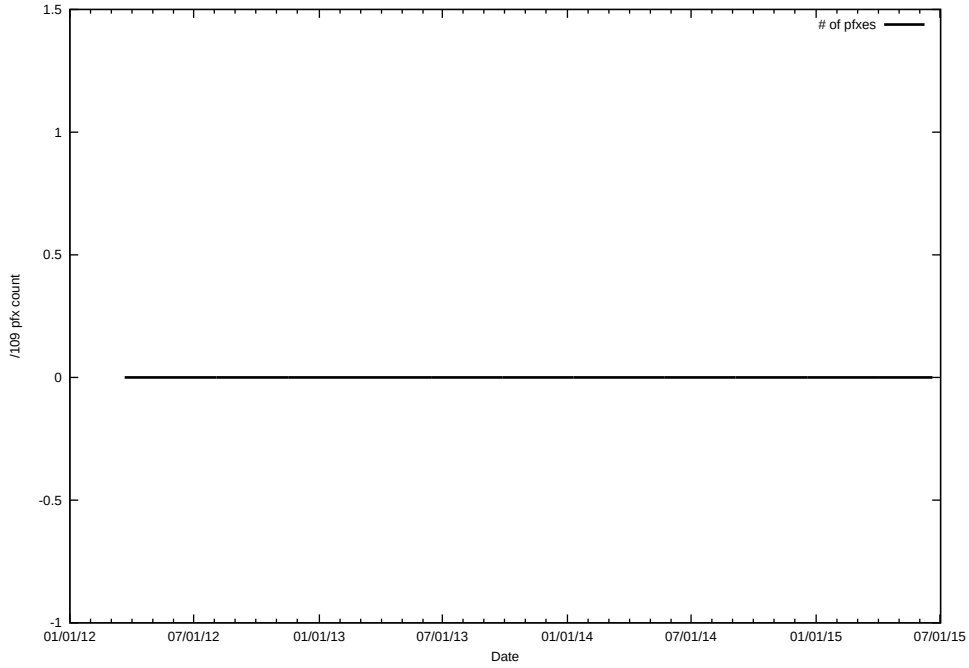
/106


1.5
1
0.5
0
-0.5
-1
/107 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

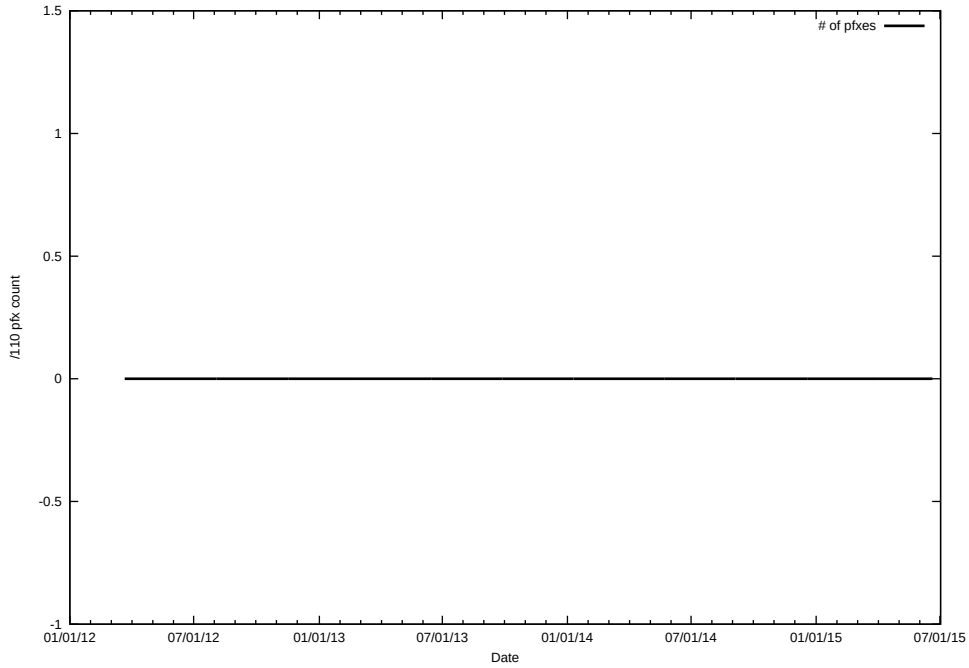
/107


1.5
1
0.5
0
-0.5
-1
/108 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

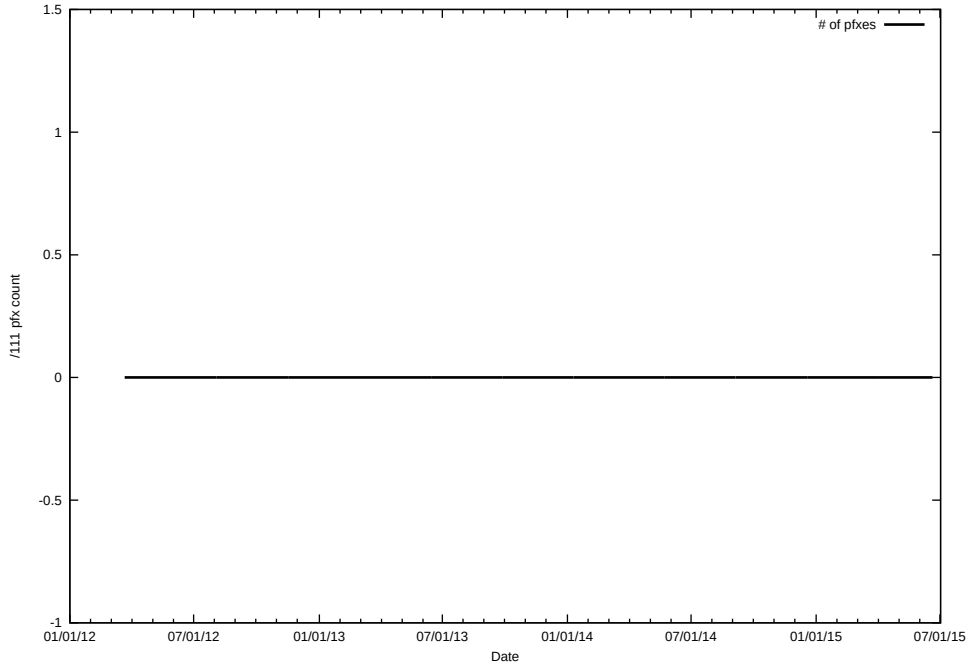
/108


1.5
1
0.5
0
-0.5
-1
/109 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

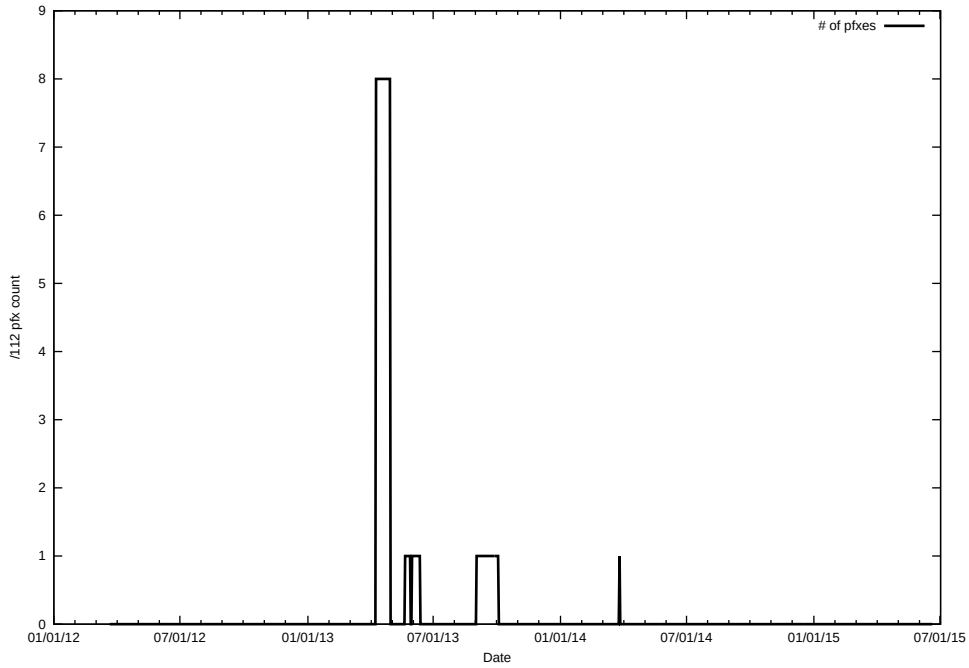
/109


1.5
1
0.5
0
-0.5
-1
/110 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

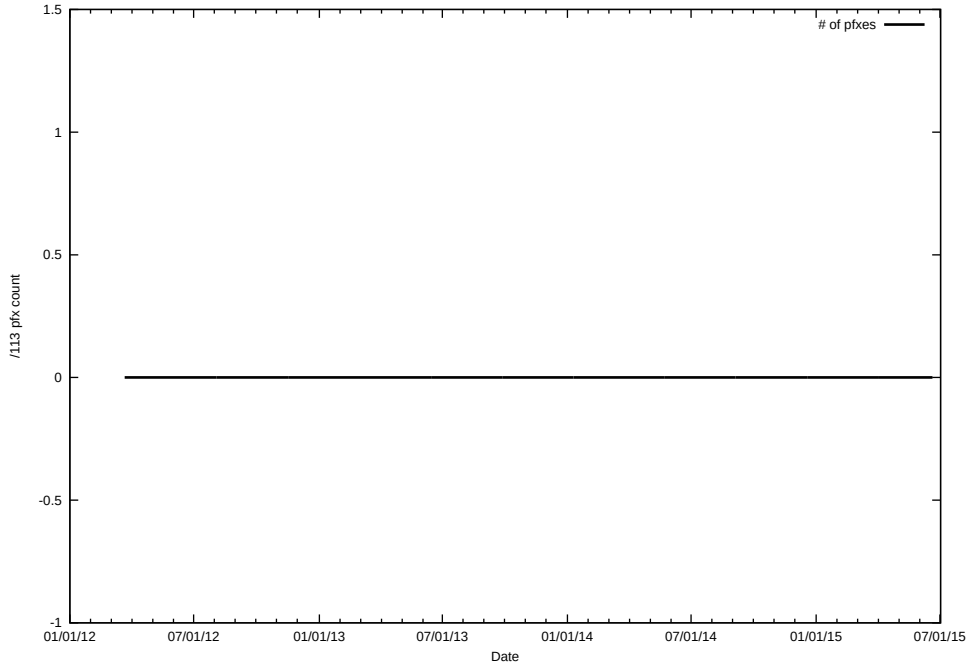
/110


1.5
1
0.5
0
-0.5
-1
/111 pfx count
# of pfxes
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

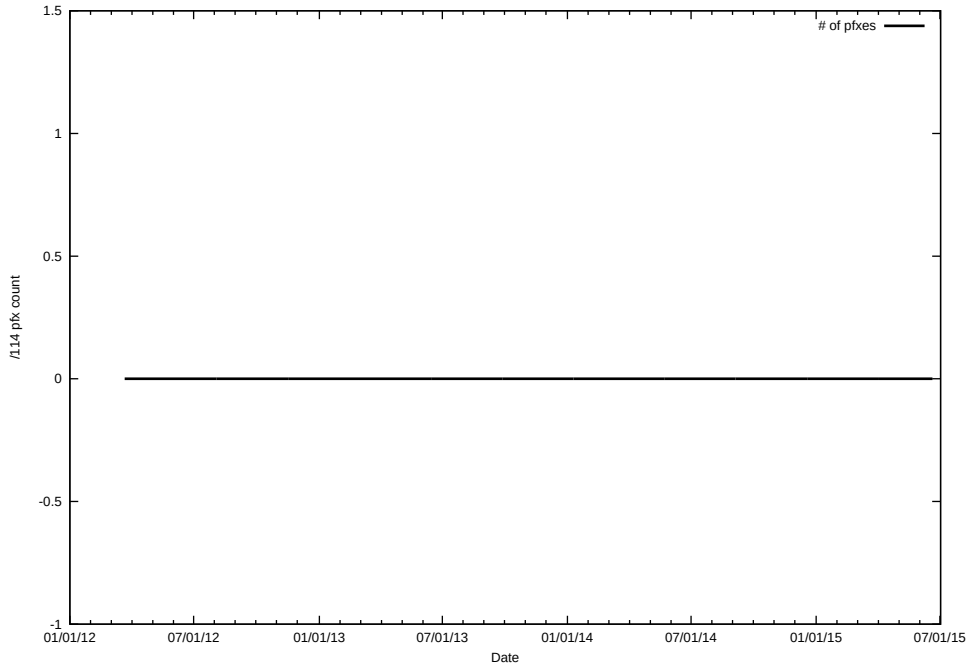
/111


9
8
7
6
5
4
3
2
1
0
/112 pfx count
# of pfxes
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

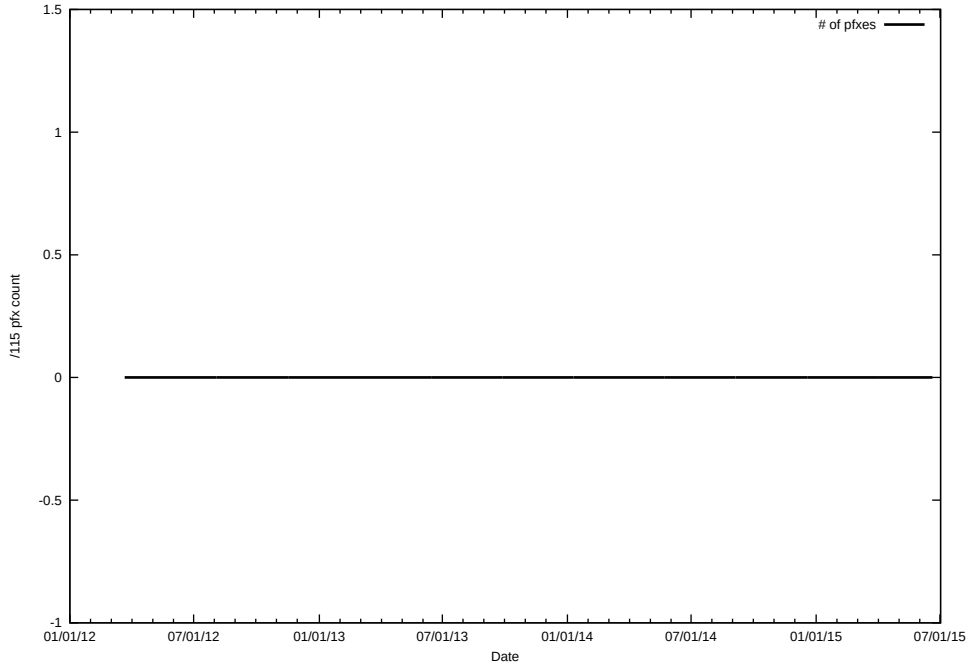
/112


1.5
1
0.5
0
-0.5
-1
/113 pfx count
# of pfxes
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

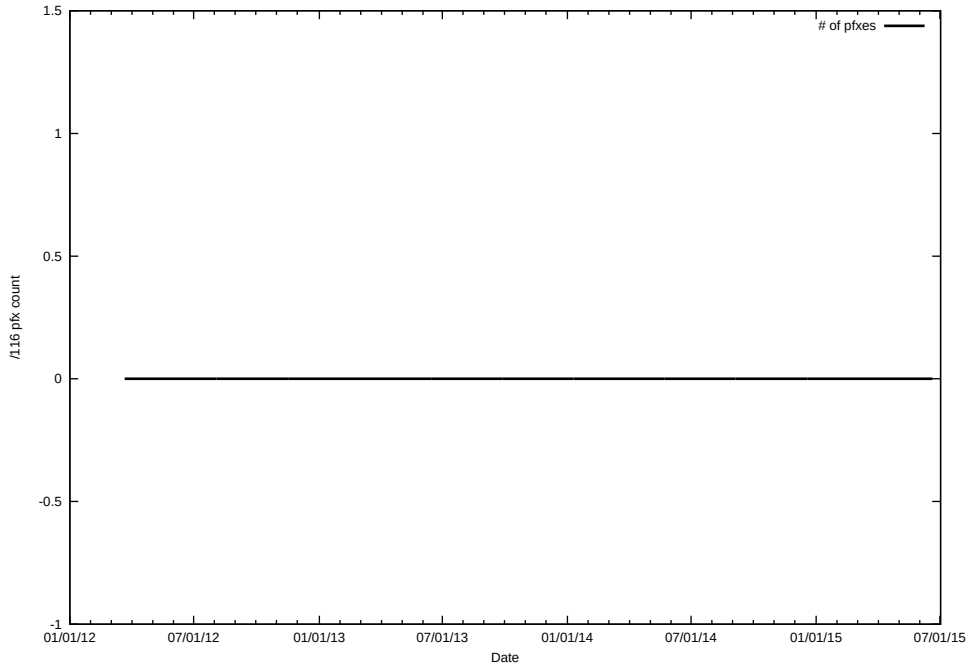
/113


1.5
1
0.5
0
-0.5
-1
/114 pfx count
# of pfxes
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

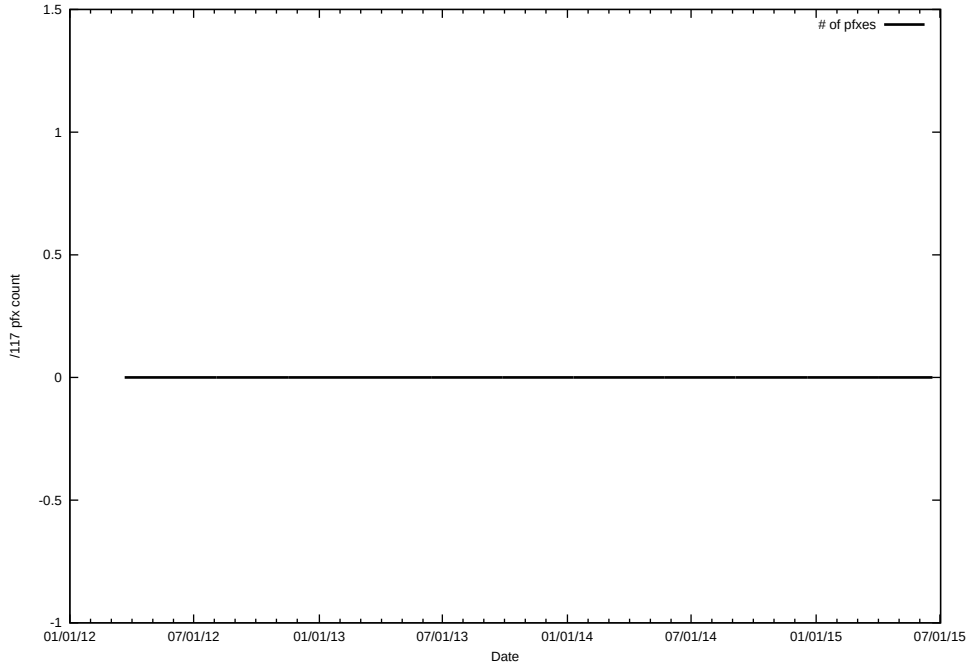
/114


1.5
1
0.5
0
-0.5
-1
/115 pfx count
# of pfxes
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

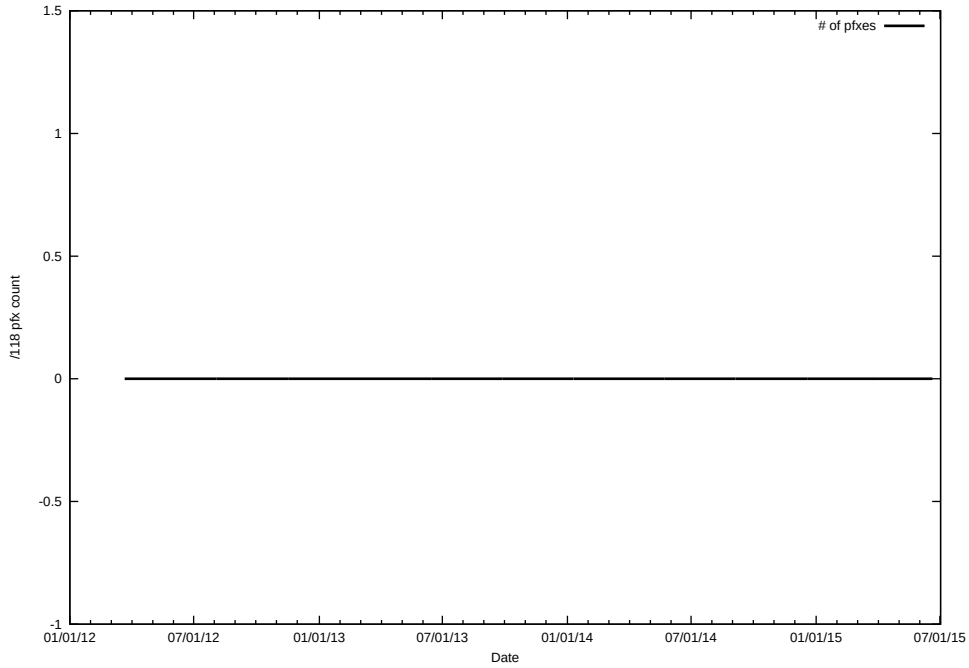
/115


1.5
1
0.5
0
-0.5
-1
/116 pfx count
# of pfxes
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

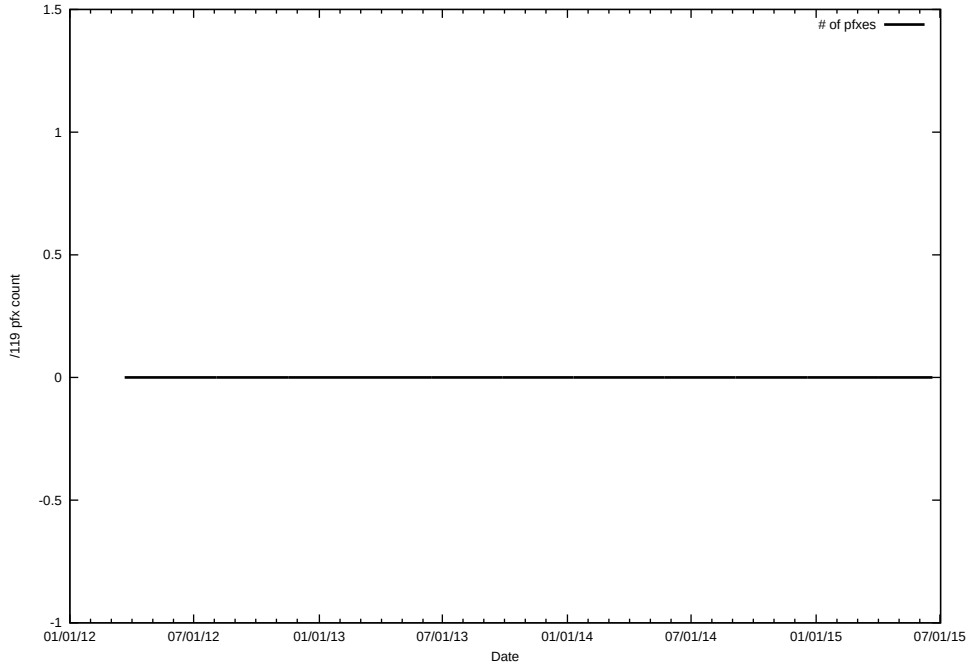
/116


1.5
1
0.5
0
-0.5
-1
/117 pfx count
# of pfxes
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

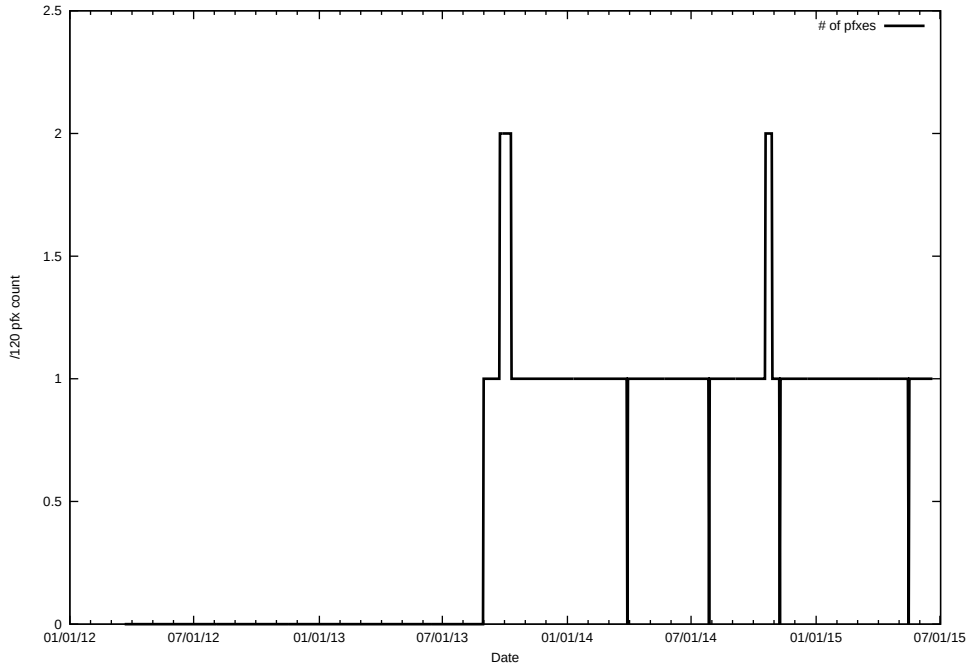
/117


1.5
1
0.5
0
-0.5
-1
/118 pfx count
# of pfxes
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

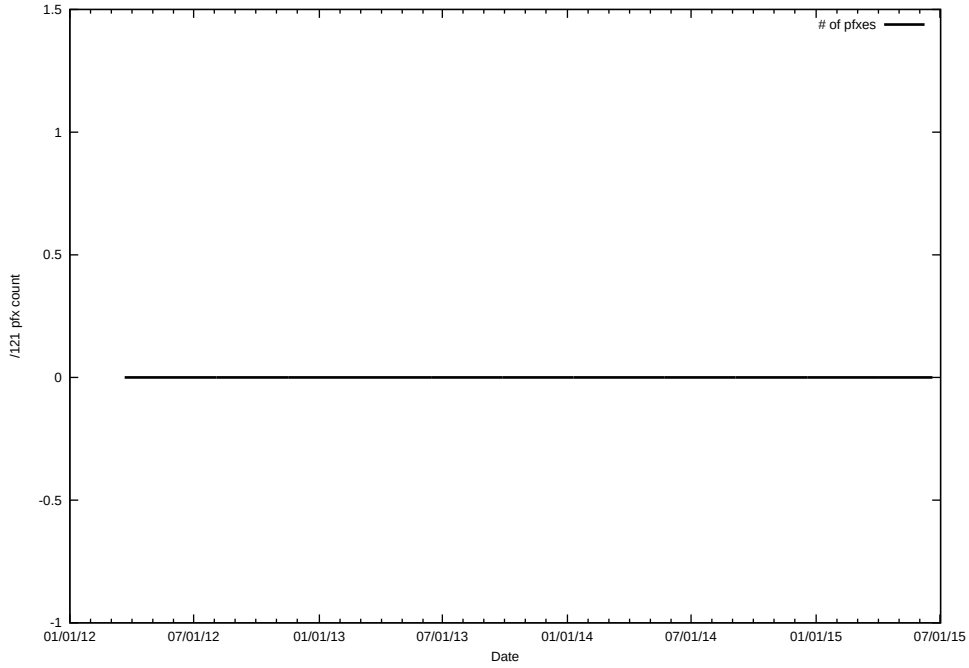
/118


1.5
1
0.5
0
-0.5
-1
/119 pfx count
# of pfxes
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

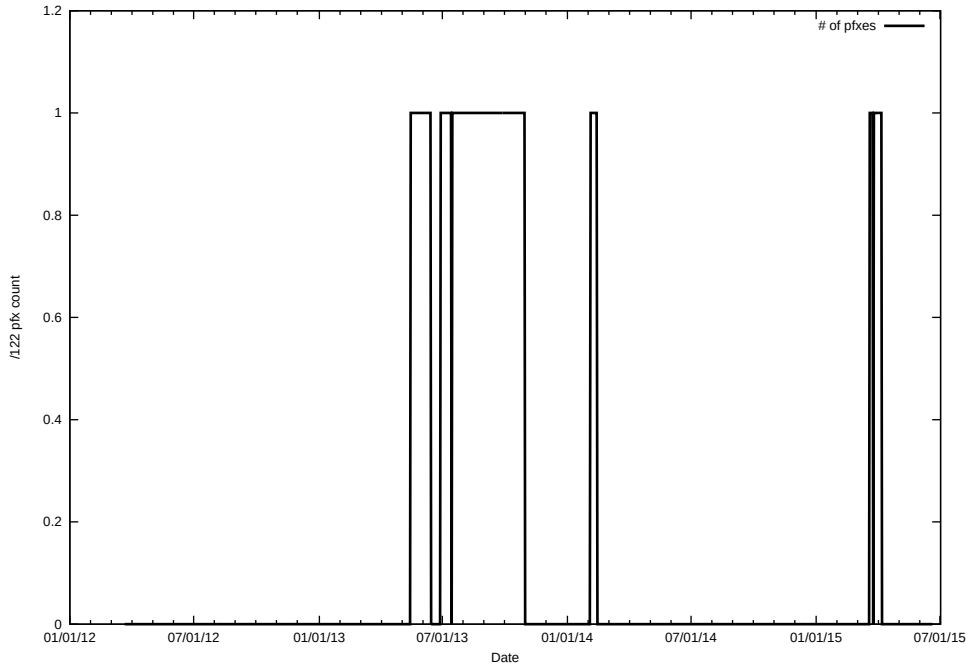
/119


2.5
2
1.5
1
0.5
0
/120 pfx count
# of pfxes
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

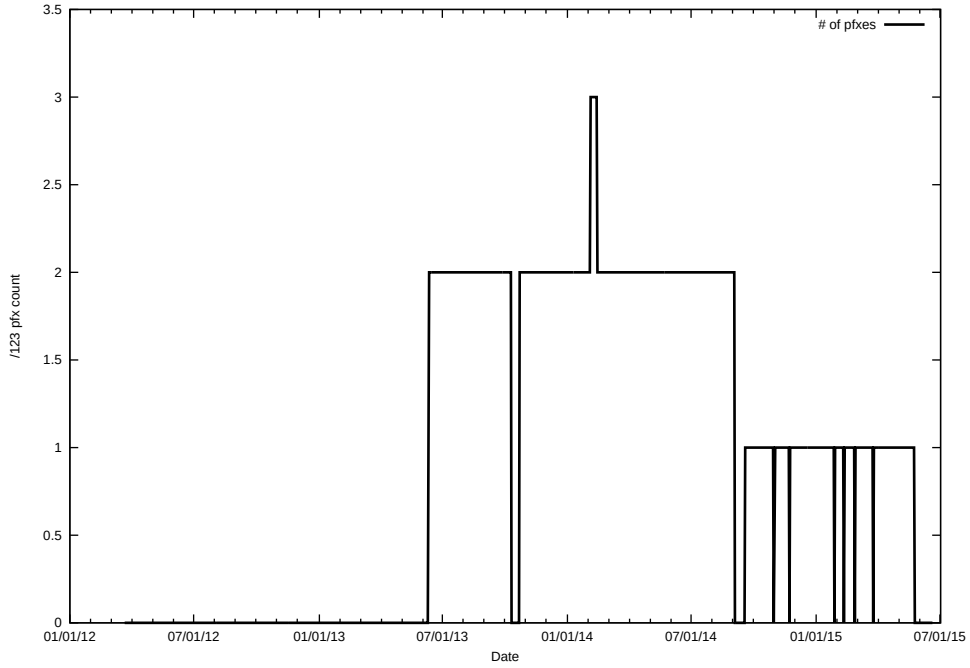
/120


1.5
1
0.5
0
-0.5
-1
/121 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

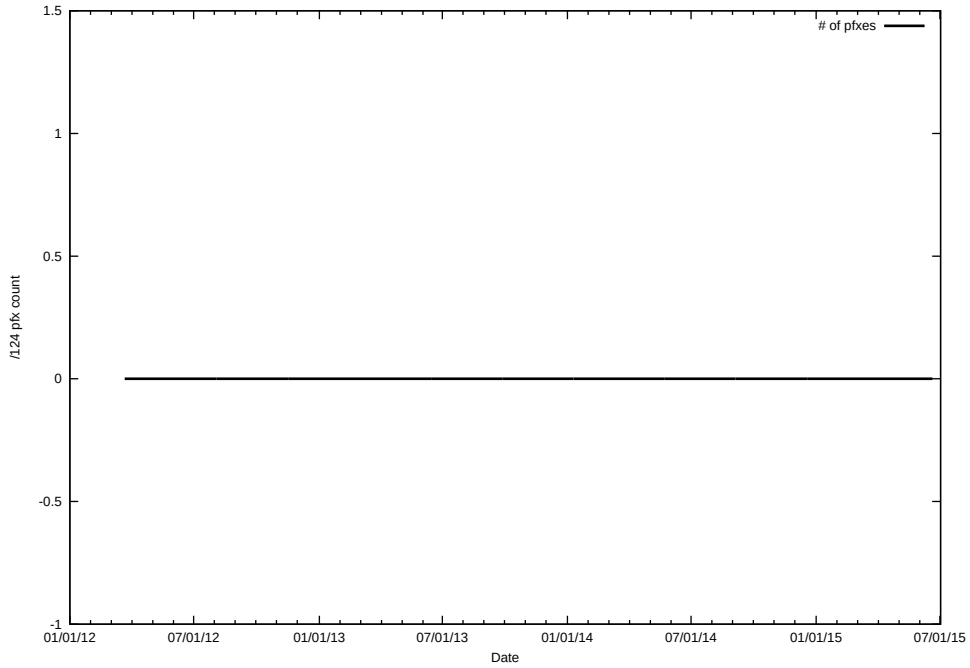
/121


1.2
1
0.8
0.6
0.4
0.2
0
/122 pfx count
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of pfxes

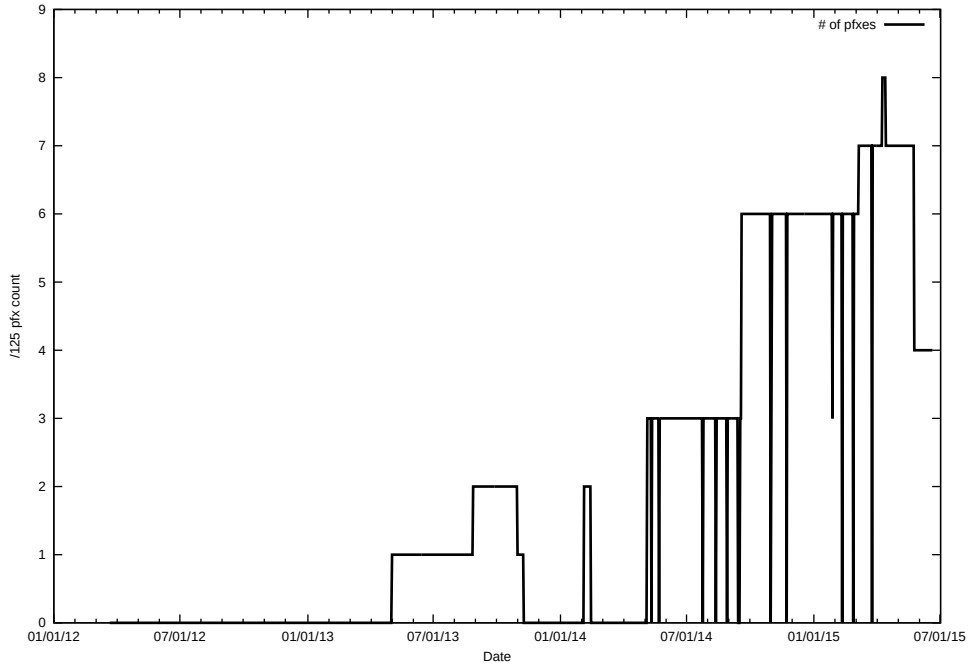
/122


3.5
3
2.5
2
1.5
1
0.5
0
/123 pfx count
# of pfxes
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

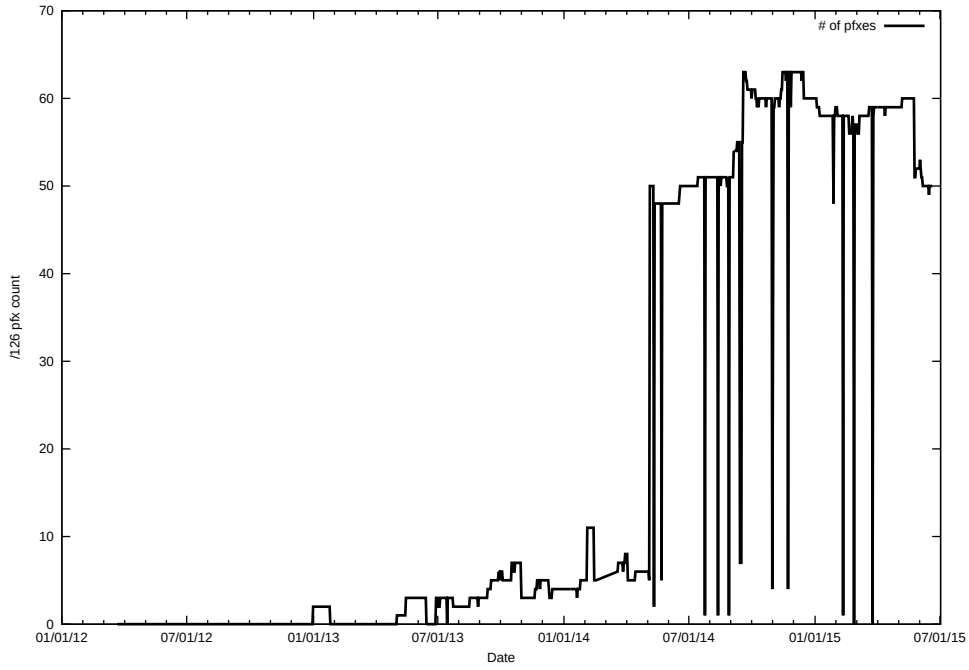
/123


1.5
1
0.5
0
-0.5
-1
/124 pfx count
# of pfxes
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

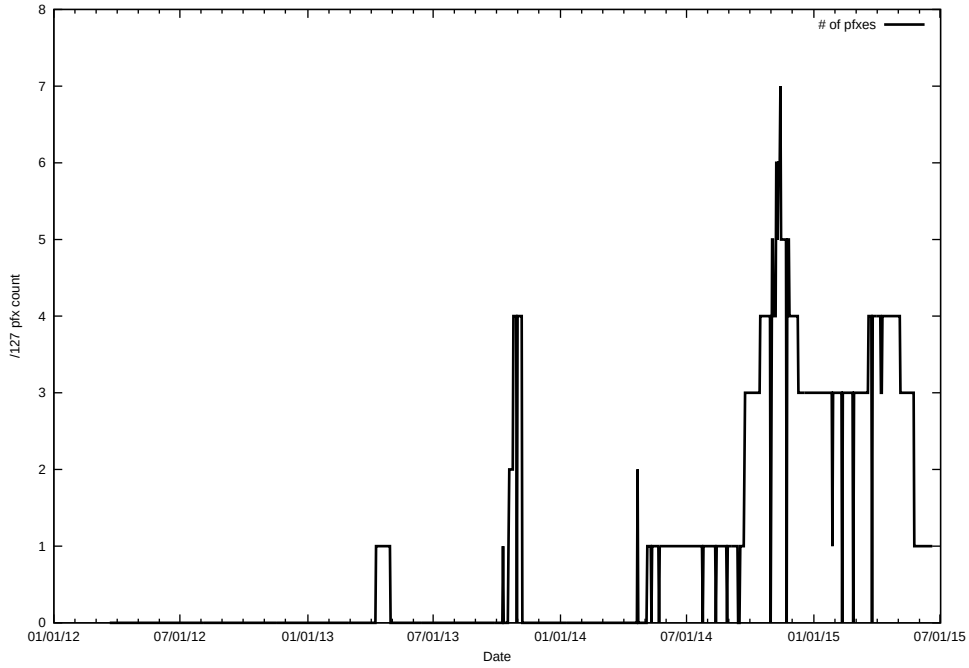
/124


# of pfxes
/125 pfx count
Date
0
1
2
3
4
5
6
7
8
9
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15

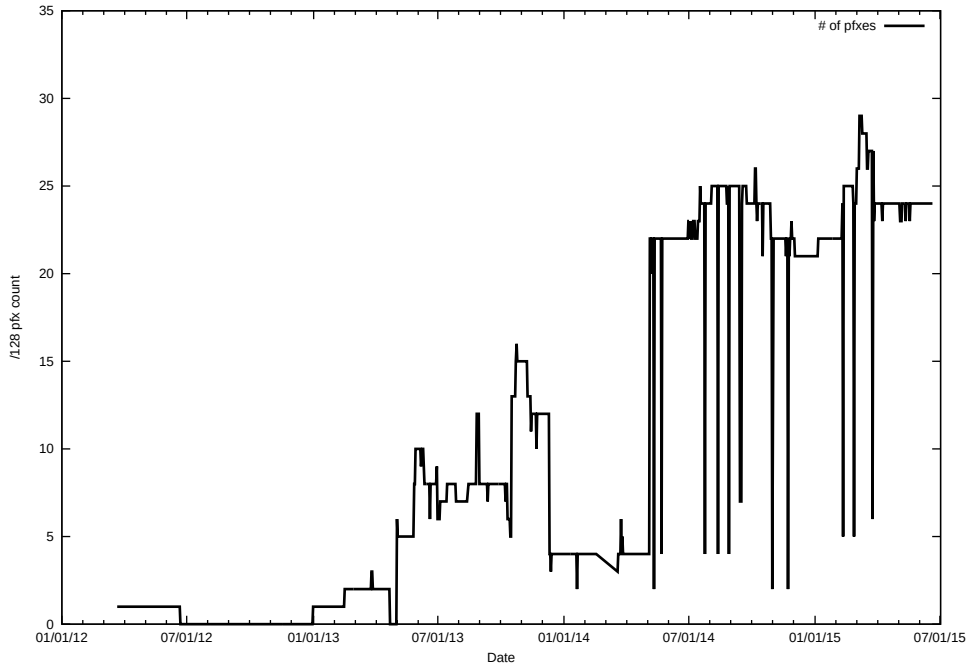
/125


# of pfxes
/126 pfx count
Date
0
10
20
30
40
50
60
70
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15

/126

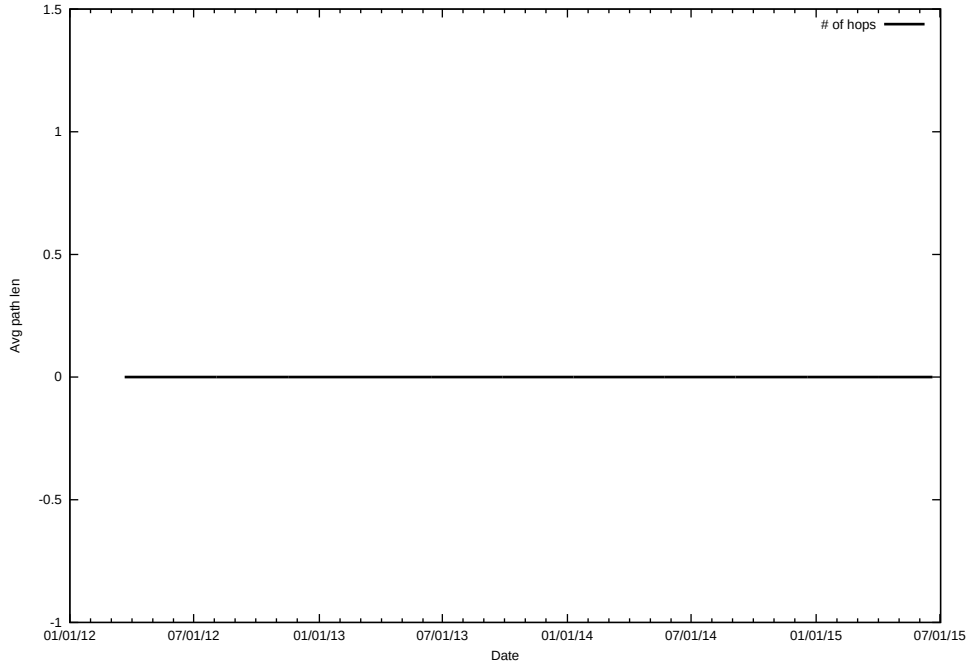

# of pfxes
/127 pfx count
Date
0
1
2
3
4
5
6
7
8
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15

/127

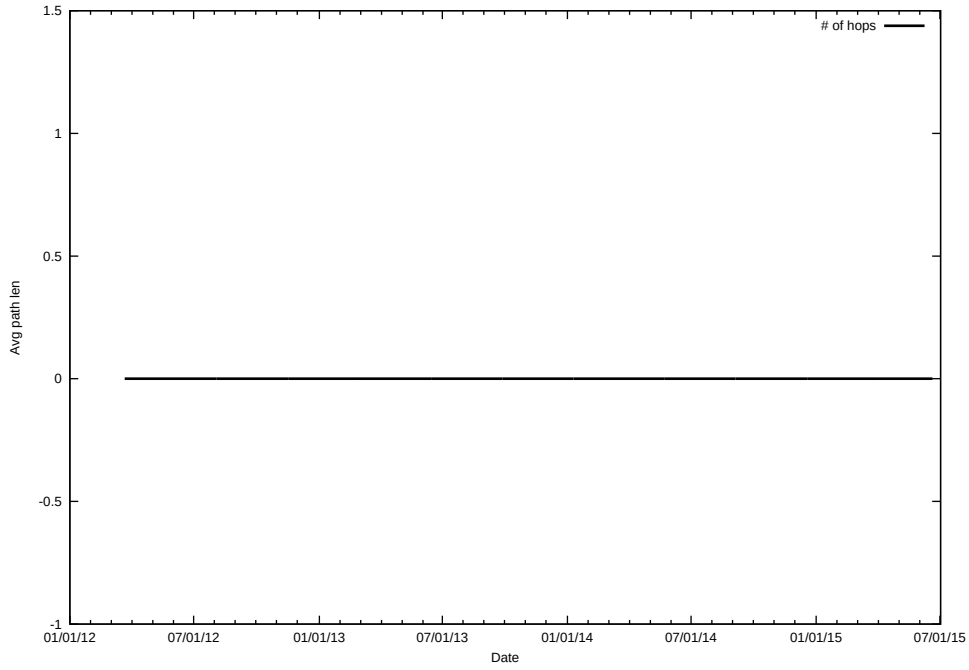

# of pfxes
/128 pfx count
Date
0
5
10
15
20
25
30
35
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15

/128

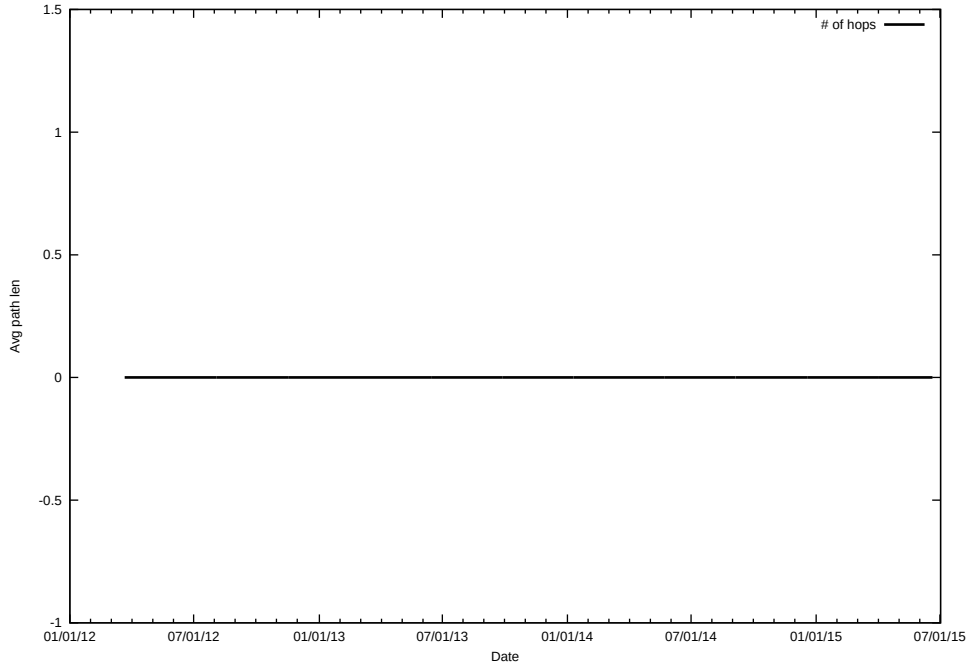
A.4 IPv4 BGP timeline of path length per prefix length



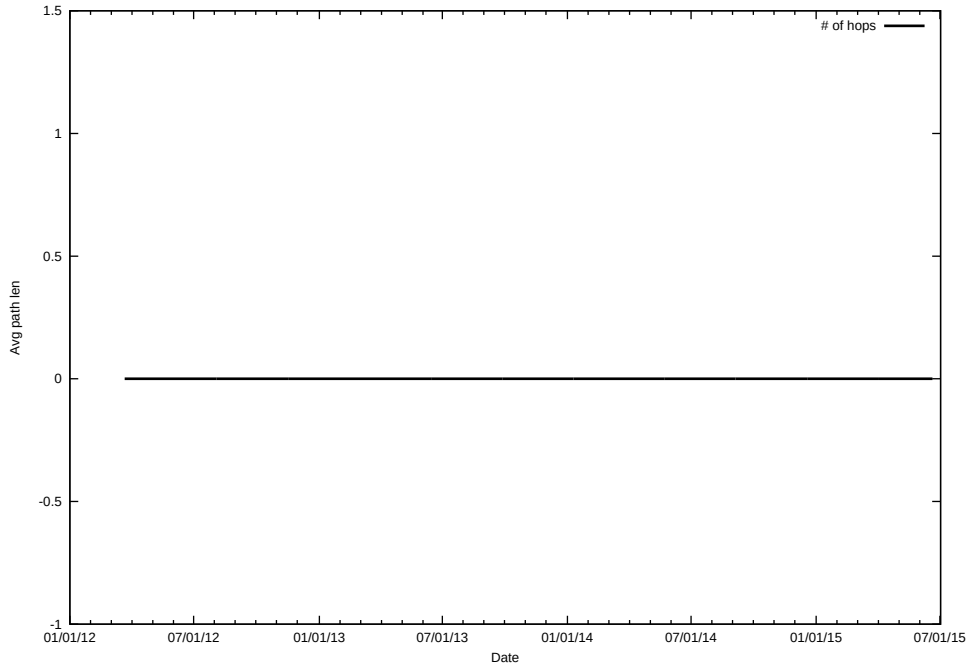
/0



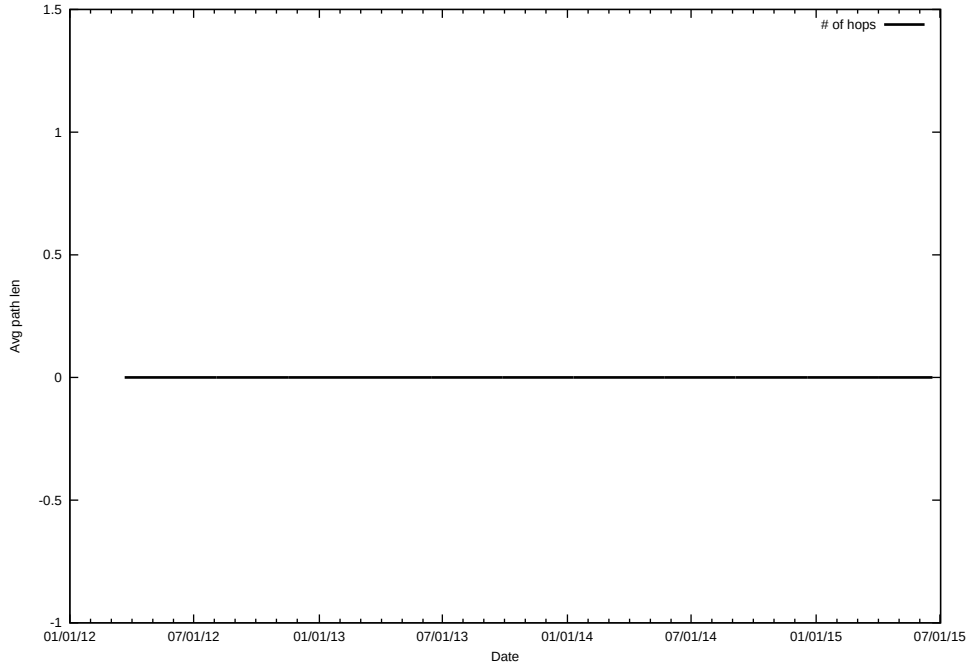
/1


1.5
1
0.5
0
-0.5
-1
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

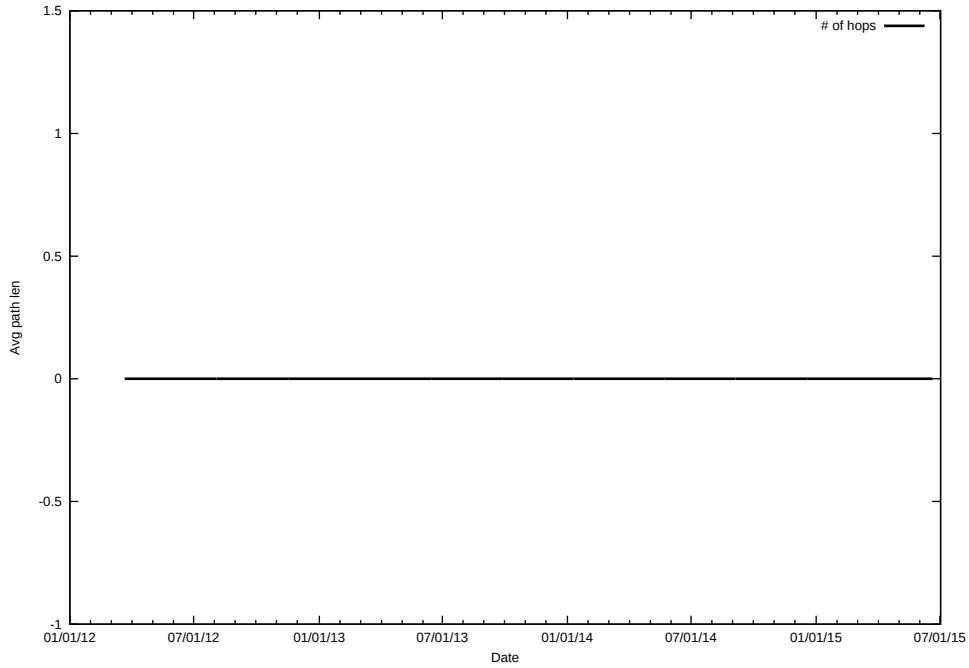
/2


1.5
1
0.5
0
-0.5
-1
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

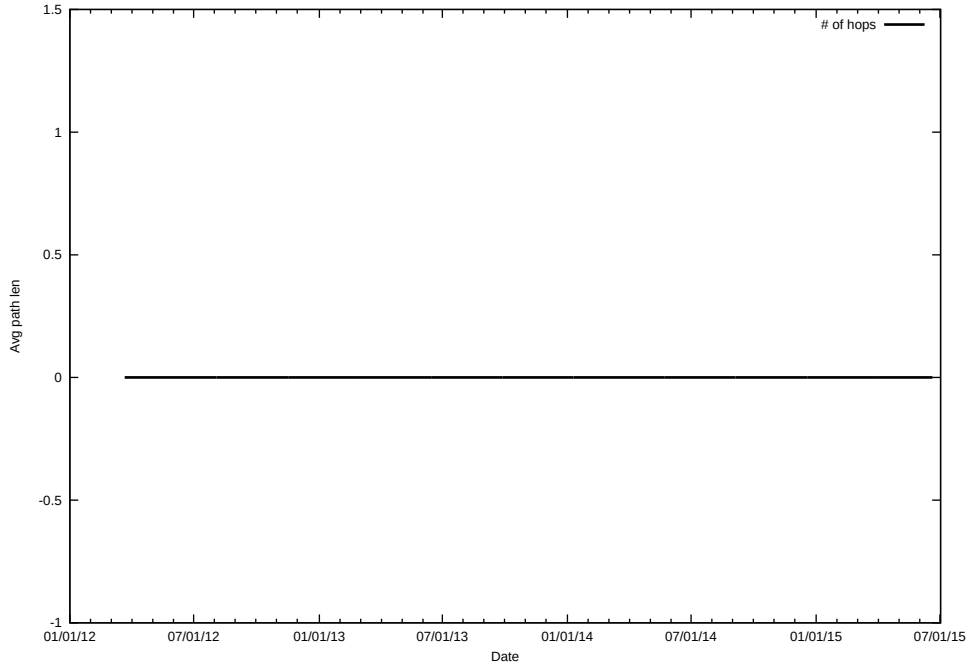
/3


1.5
1
0.5
0
-0.5
-1
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

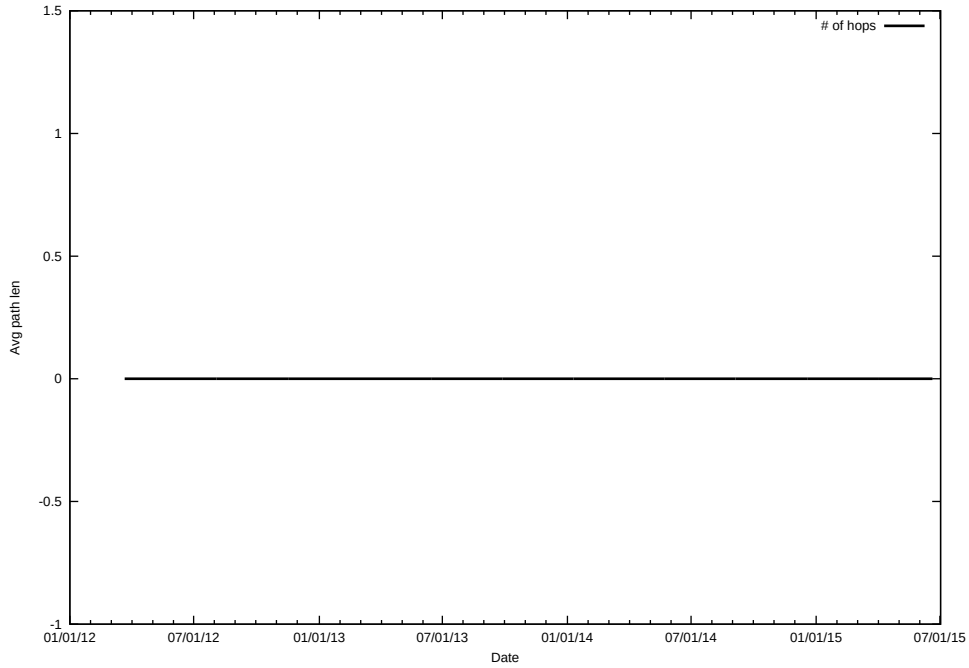
/4


1.5
1
0.5
0
-0.5
-1
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

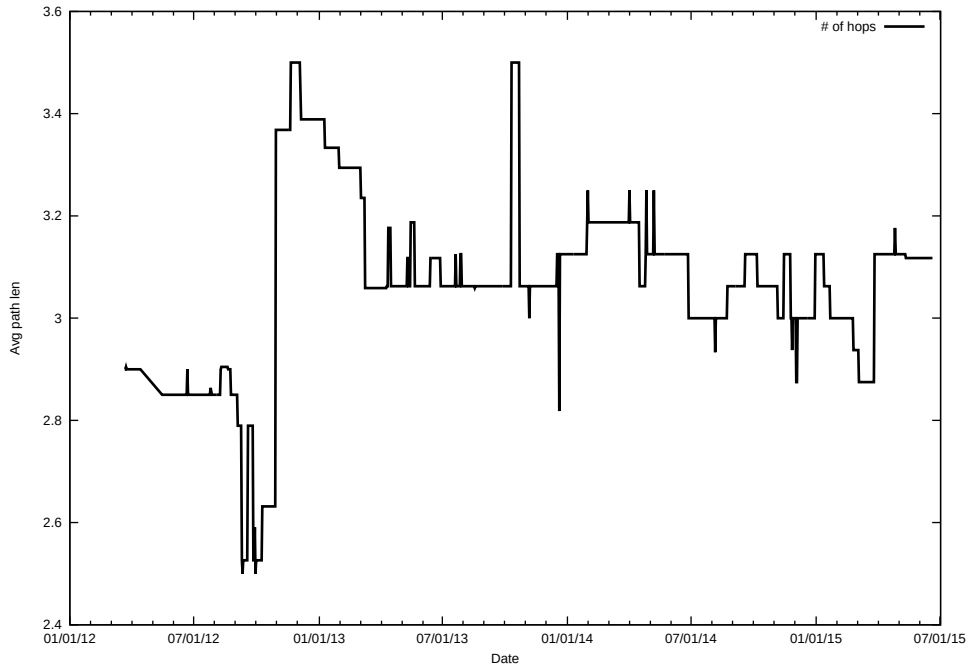
/5


1.5
1
0.5
0
-0.5
-1
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

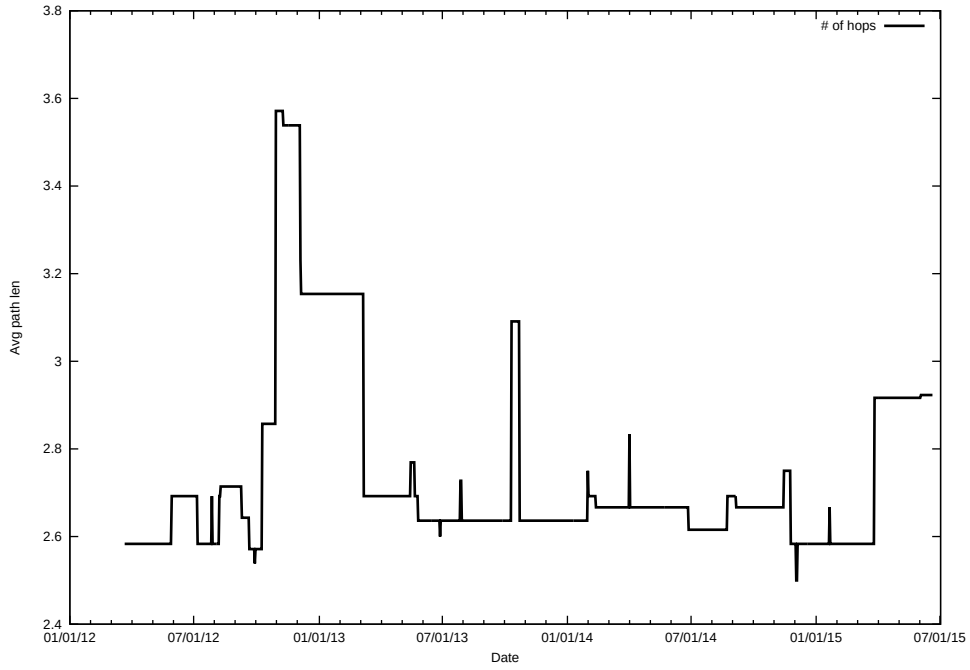
/6


1.5
1
0.5
0
-0.5
-1
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

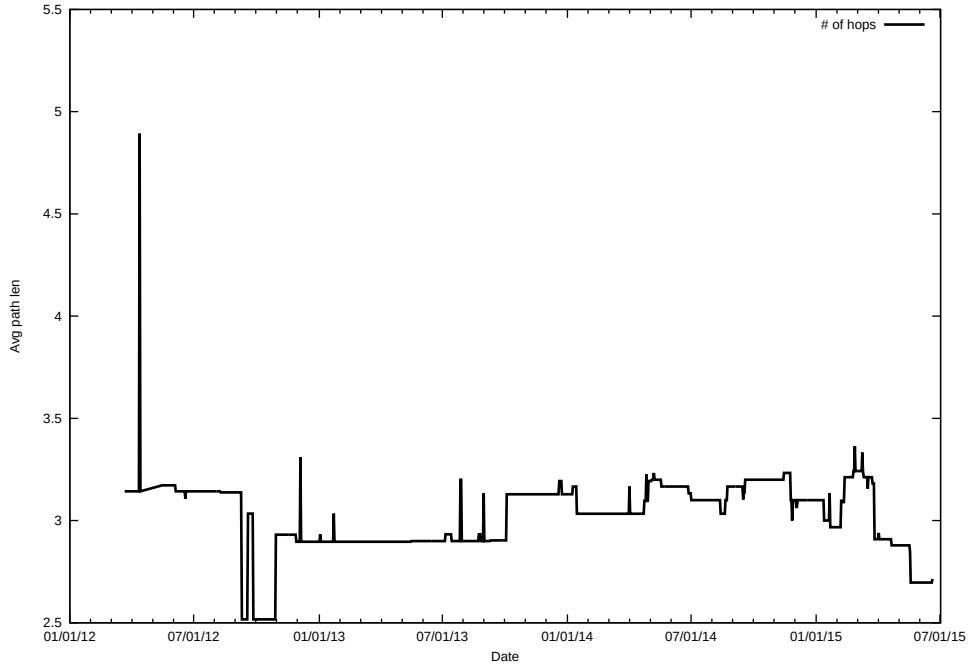
/7


3.6
3.4
3.2
3
2.8
2.6
2.4
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

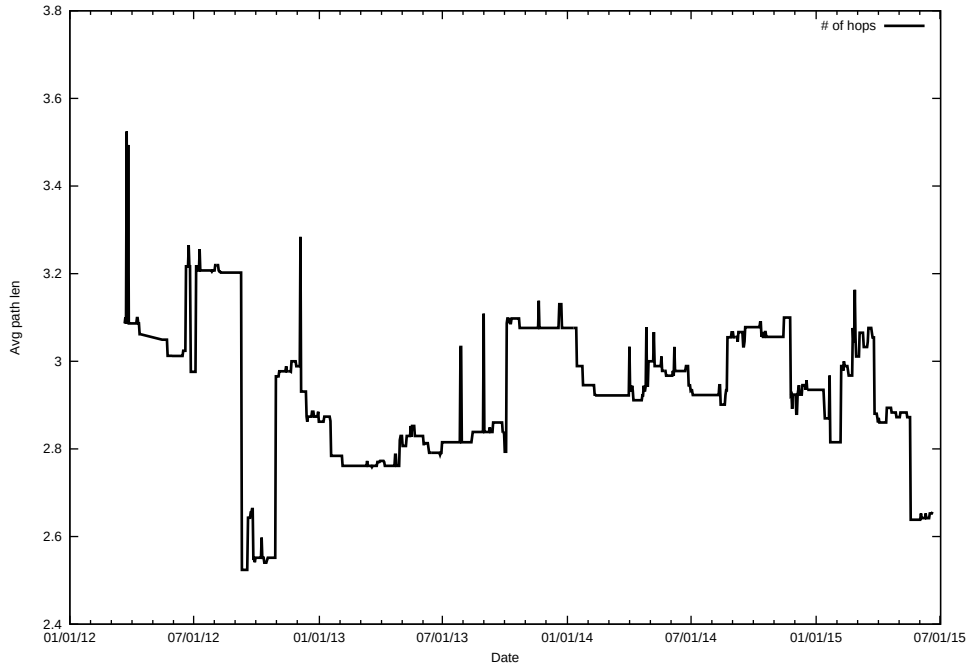
/8


3.8
3.6
3.4
3.2
3
2.8
2.6
2.4
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

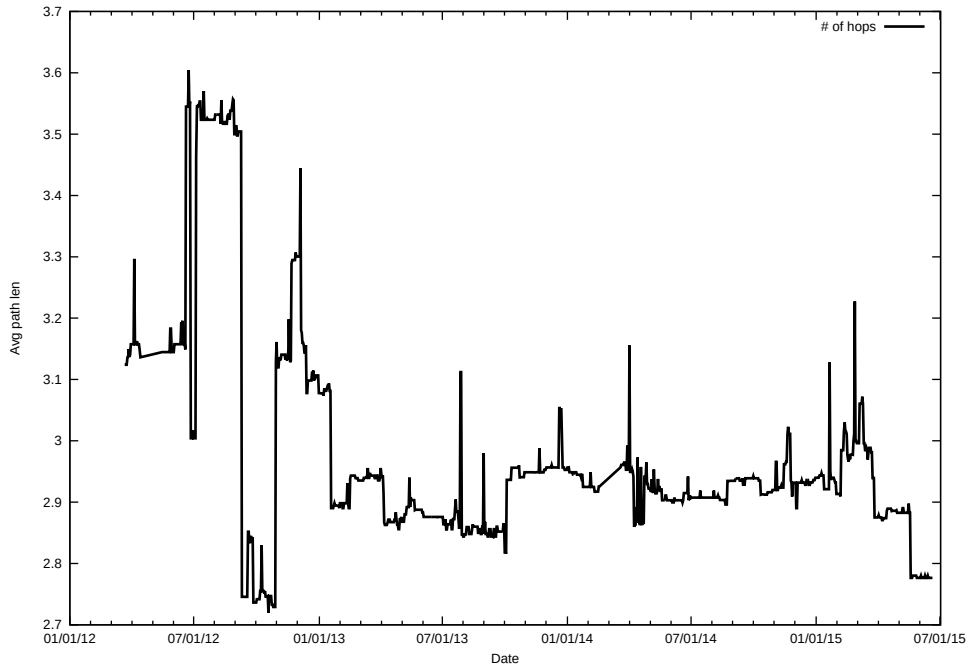
/9


5.5
5
4.5
4
3.5
3
2.5
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

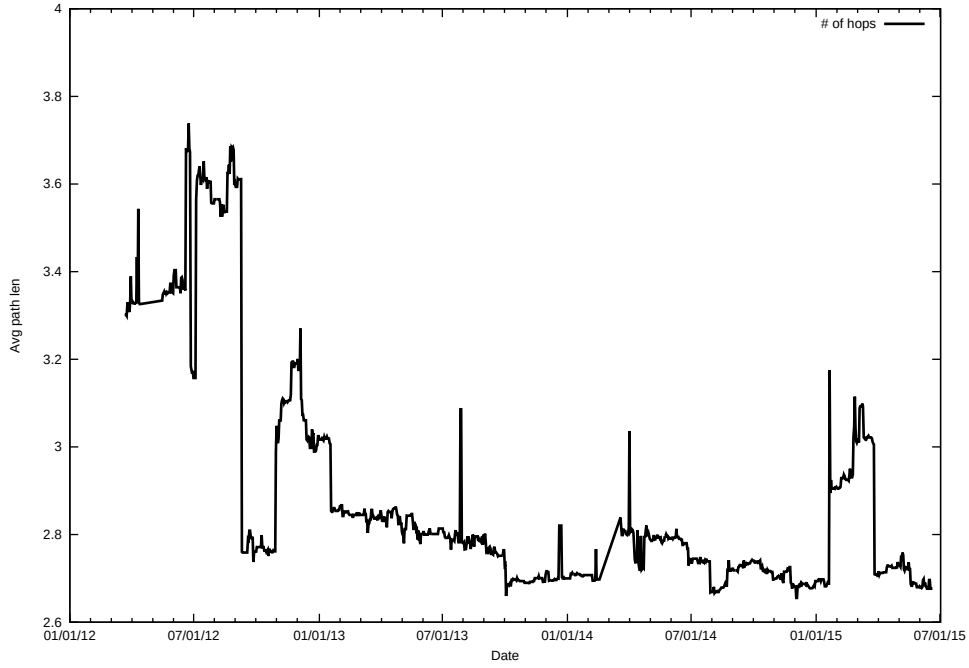
/10


3.8
3.6
3.4
3.2
3
2.8
2.6
2.4
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

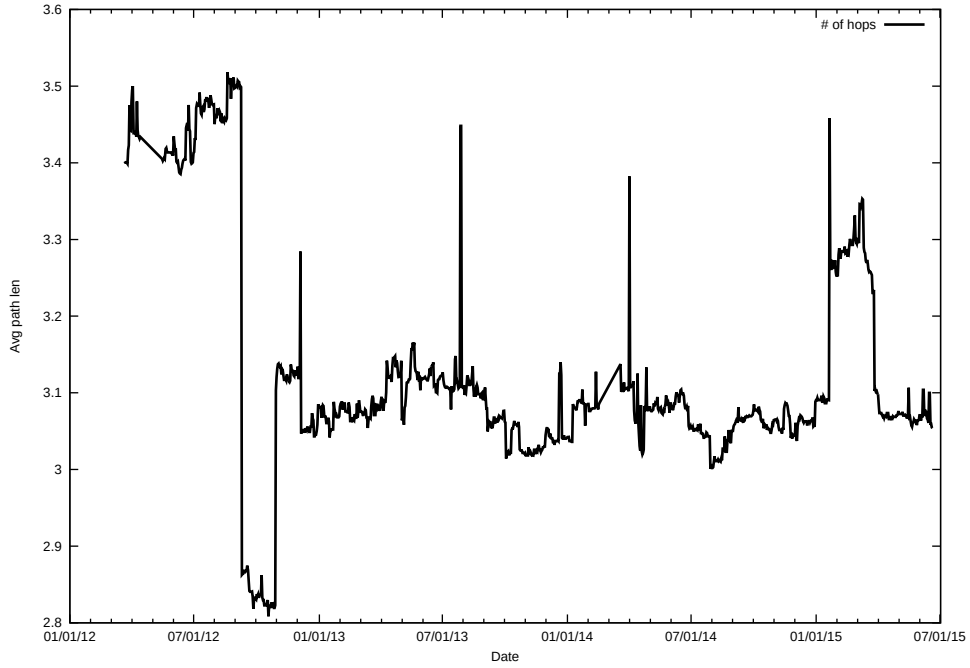
/11


3.7
3.6
3.5
3.4
3.3
3.2
3.1
3
2.9
2.8
2.7
Avg path len
# of hops
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

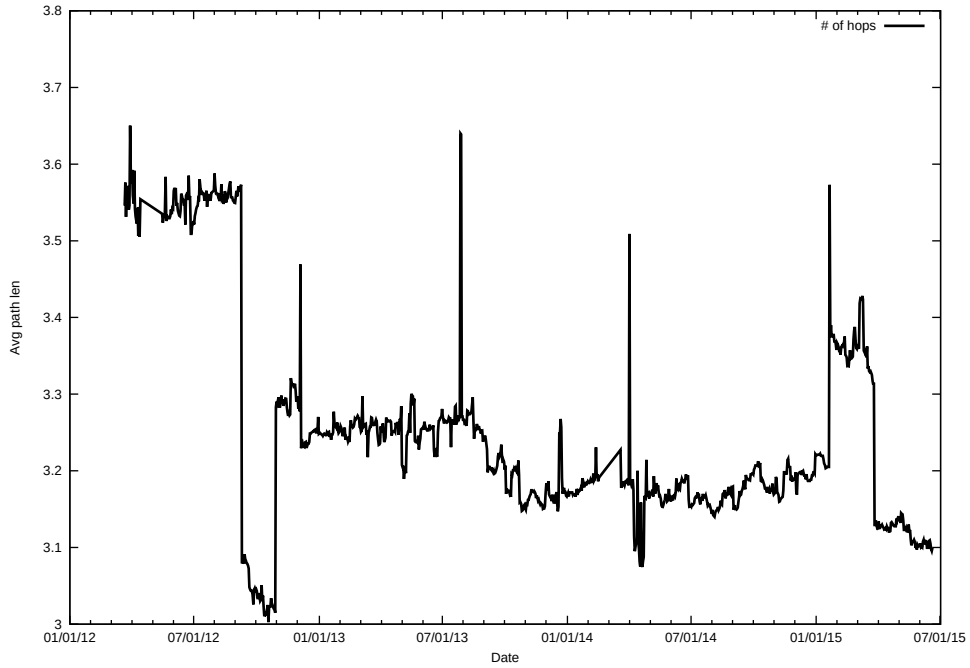
/12


4
3.8
3.6
3.4
3.2
3
2.8
2.6
Avg path len
# of hops
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

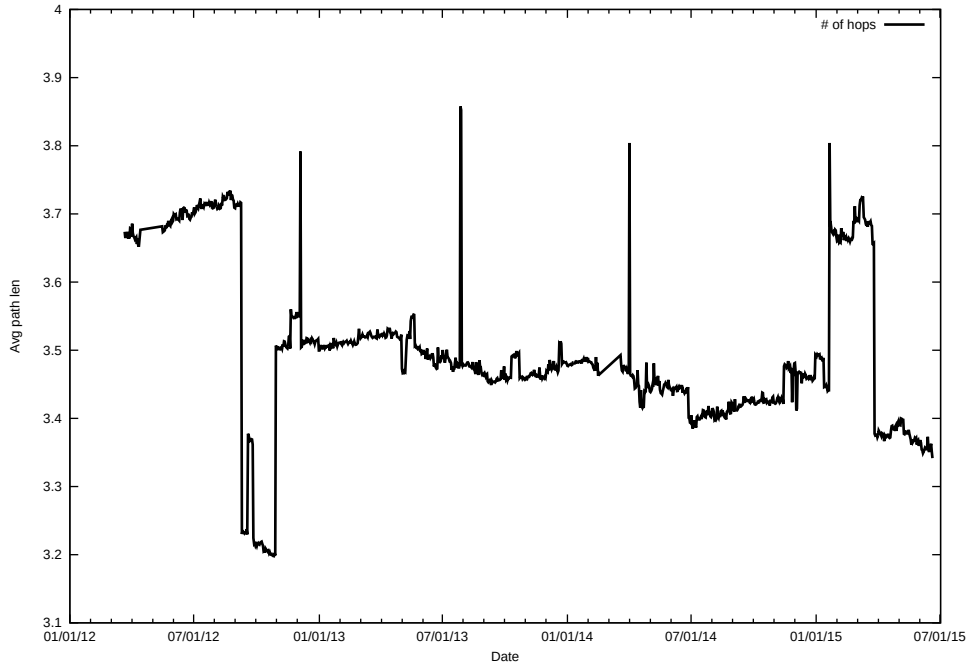
/13


3.6
3.5
3.4
3.3
3.2
3.1
3
2.9
2.8
Avg path len
# of hops
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

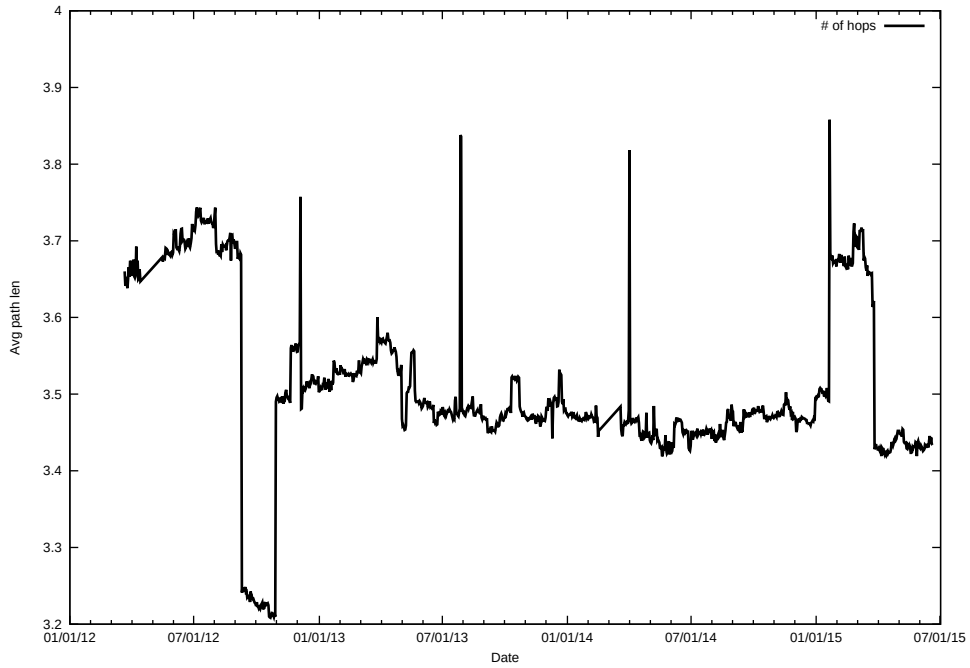
/14


3.8
3.7
3.6
3.5
3.4
3.3
3.2
3.1
3
Avg path len
# of hops
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

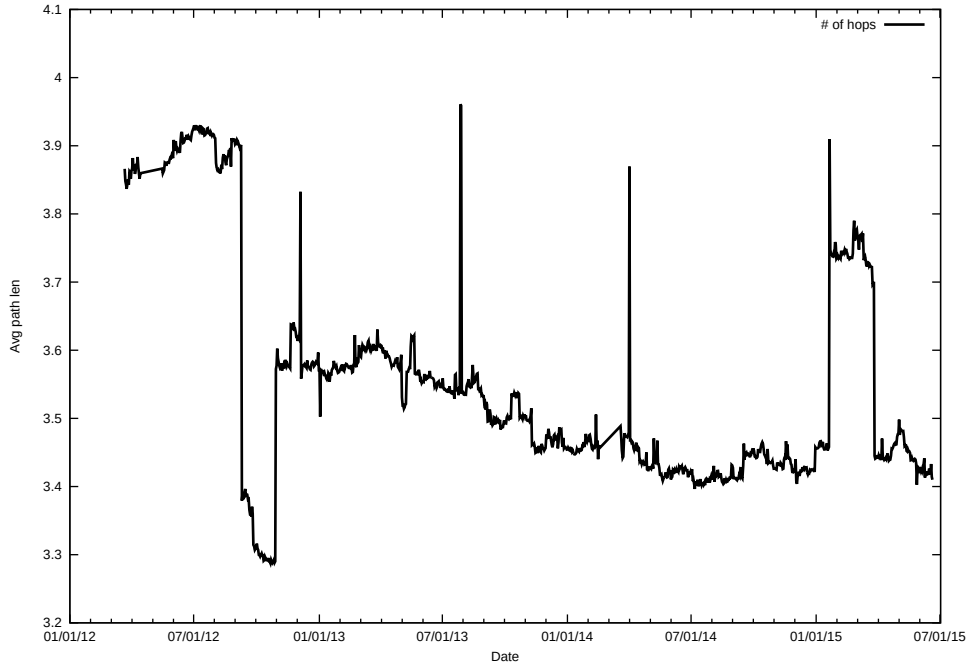
/15


# of hops
Avg path len
Date
4
3.9
3.8
3.7
3.6
3.5
3.4
3.3
3.2
3.1
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15

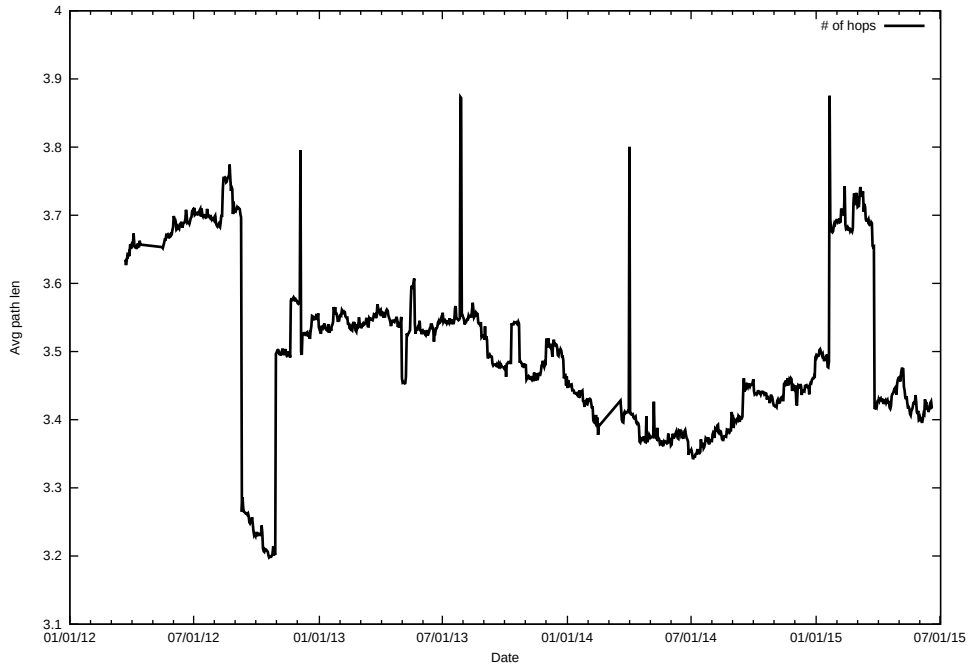
/16


# of hops
Avg path len
Date
4
3.9
3.8
3.7
3.6
3.5
3.4
3.3
3.2
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15

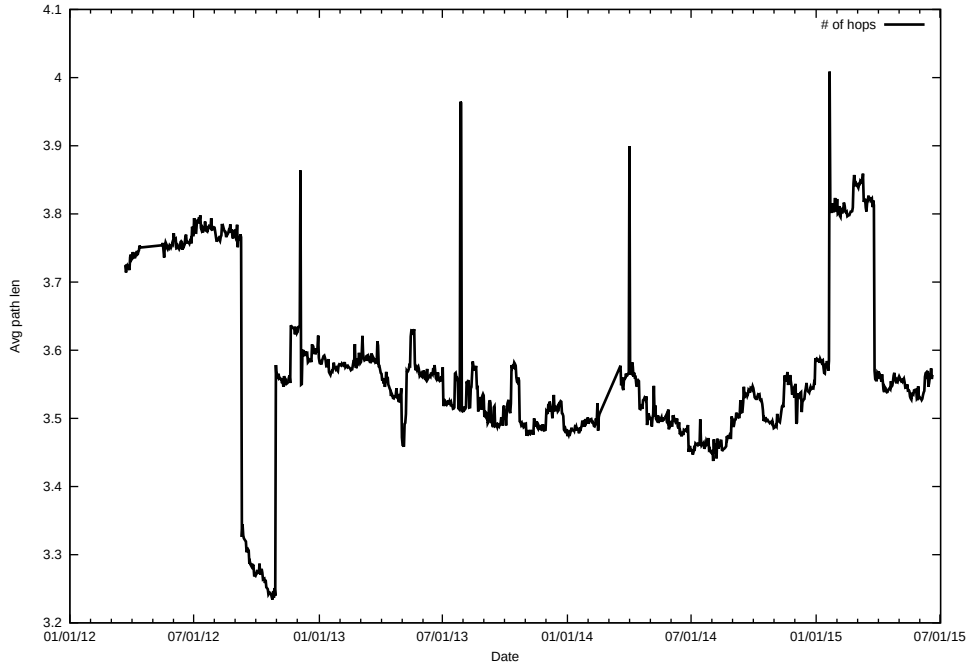
/17


4.1
4
3.9
3.8
3.7
3.6
3.5
3.4
3.3
3.2
Avg path len
# of hops
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

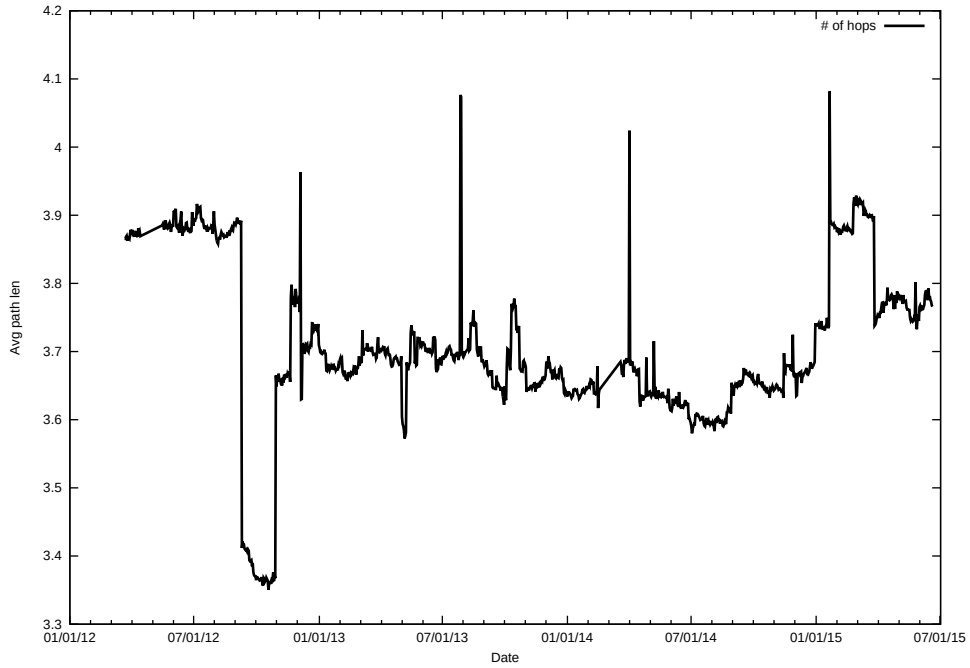
/18


4
3.9
3.8
3.7
3.6
3.5
3.4
3.3
3.2
3.1
Avg path len
# of hops
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

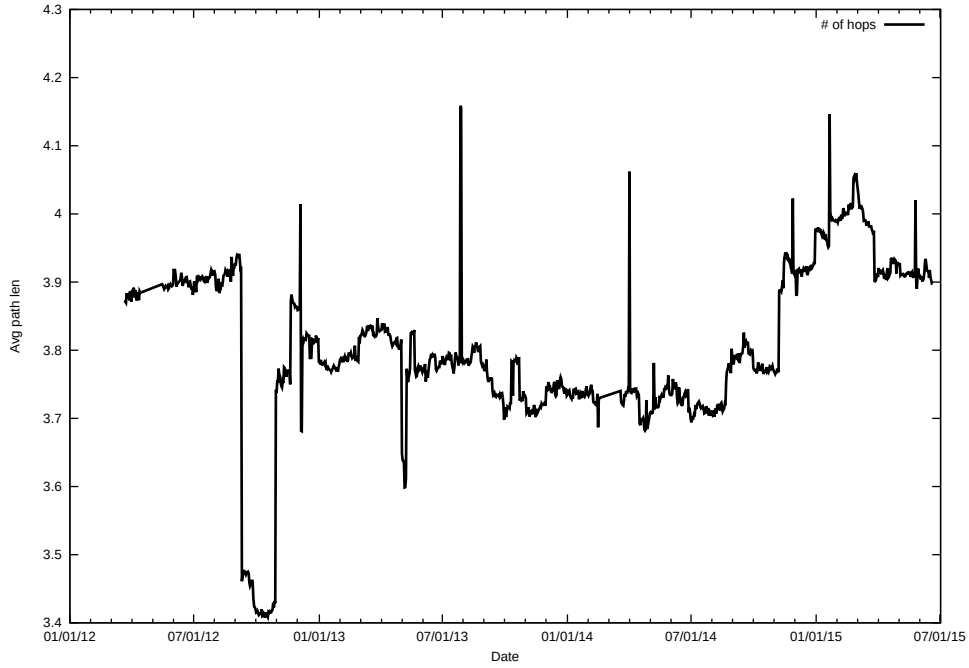
/19


4.1
4
3.9
3.8
3.7
3.6
3.5
3.4
3.3
3.2
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

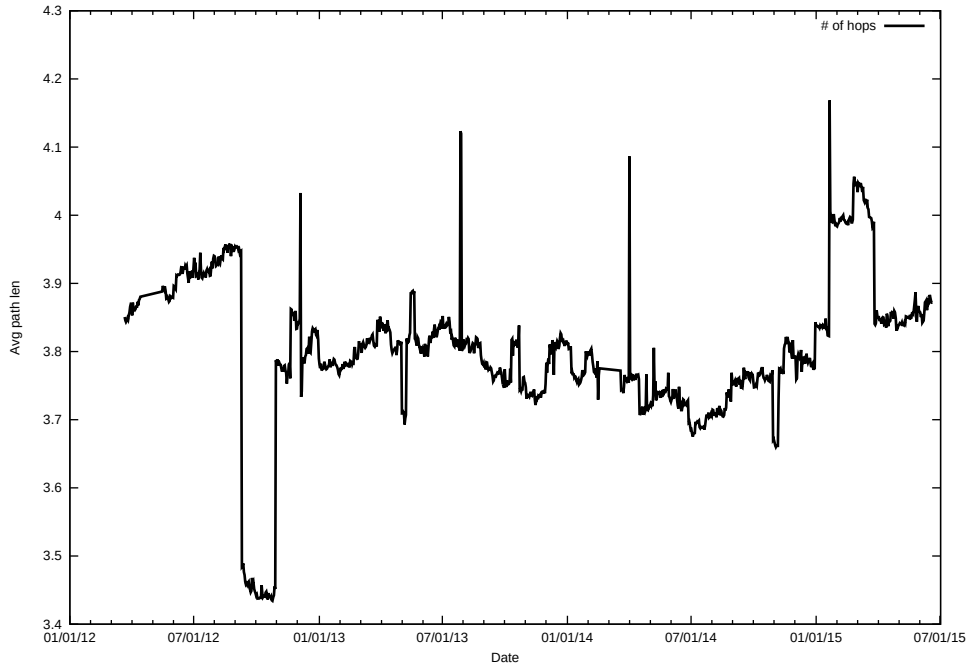
/20


4.2
4.1
4
3.9
3.8
3.7
3.6
3.5
3.4
3.3
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

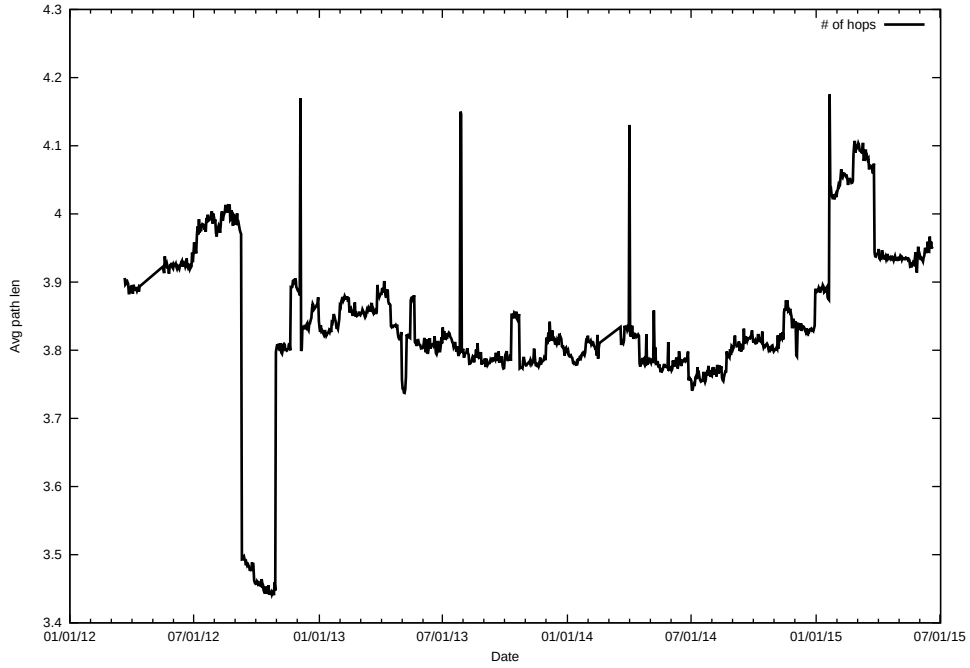
/21


# of hops
Avg path len
Date
4.3
4.2
4.1
4
3.9
3.8
3.7
3.6
3.5
3.4
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15

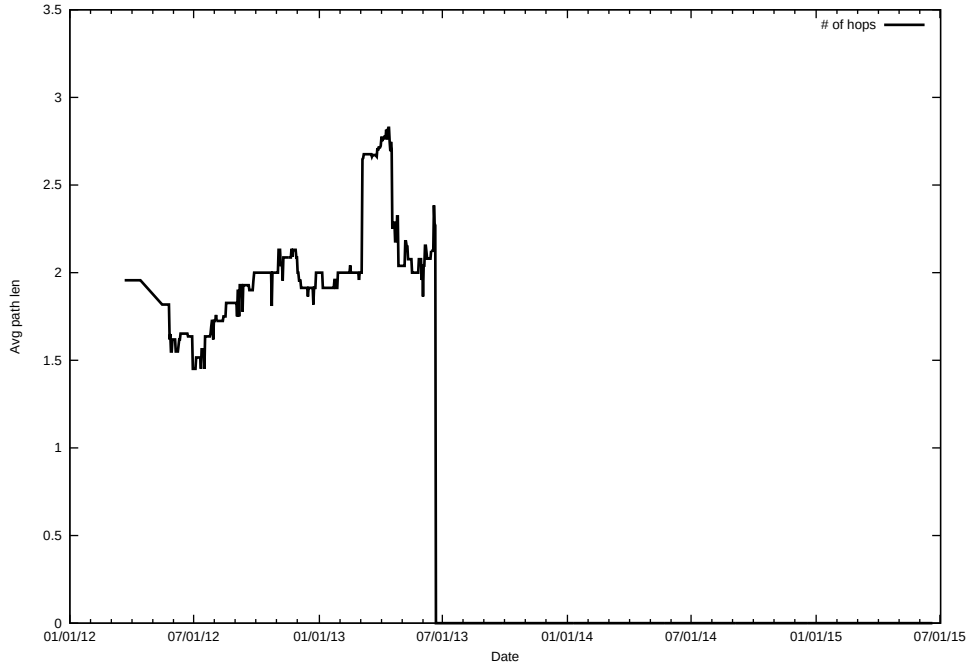
/22


# of hops
Avg path len
Date
4.3
4.2
4.1
4
3.9
3.8
3.7
3.6
3.5
3.4
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15

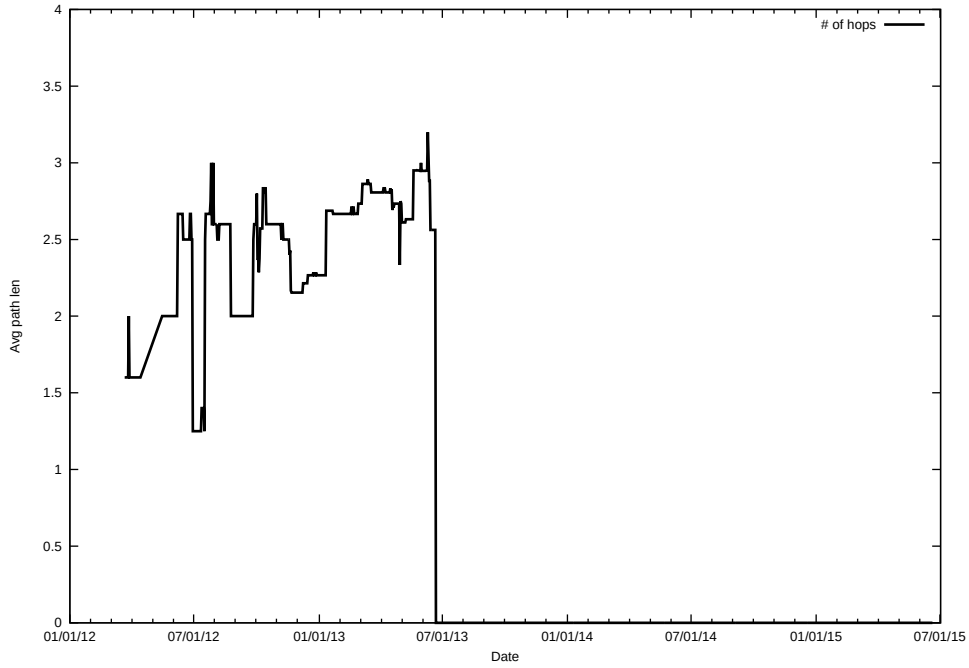
/23


# of hops
Avg path len
Date
4.3
4.2
4.1
4
3.9
3.8
3.7
3.6
3.5
3.4
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15

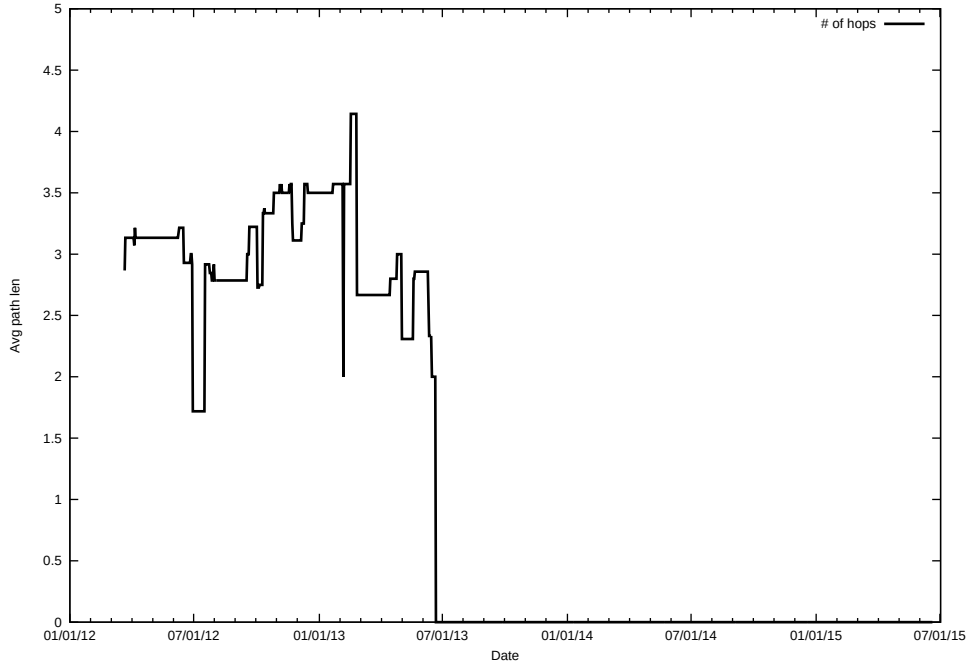
/24


# of hops
Avg path len
Date
3.5
3
2.5
2
1.5
1
0.5
0
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15

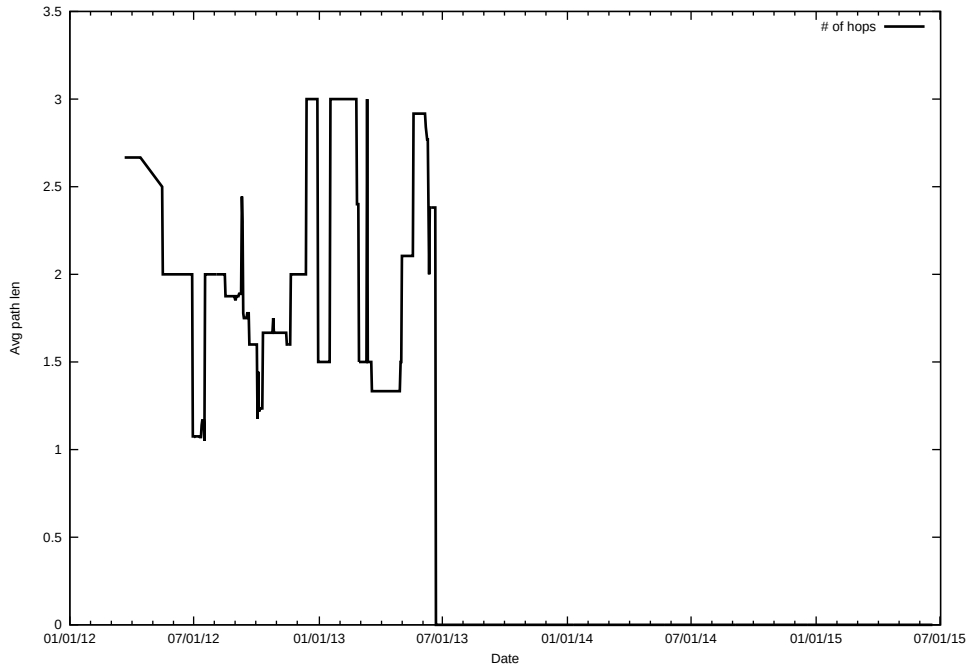
/25


# of hops
Avg path len
Date
0
0.5
1
1.5
2
2.5
3
3.5
4
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15

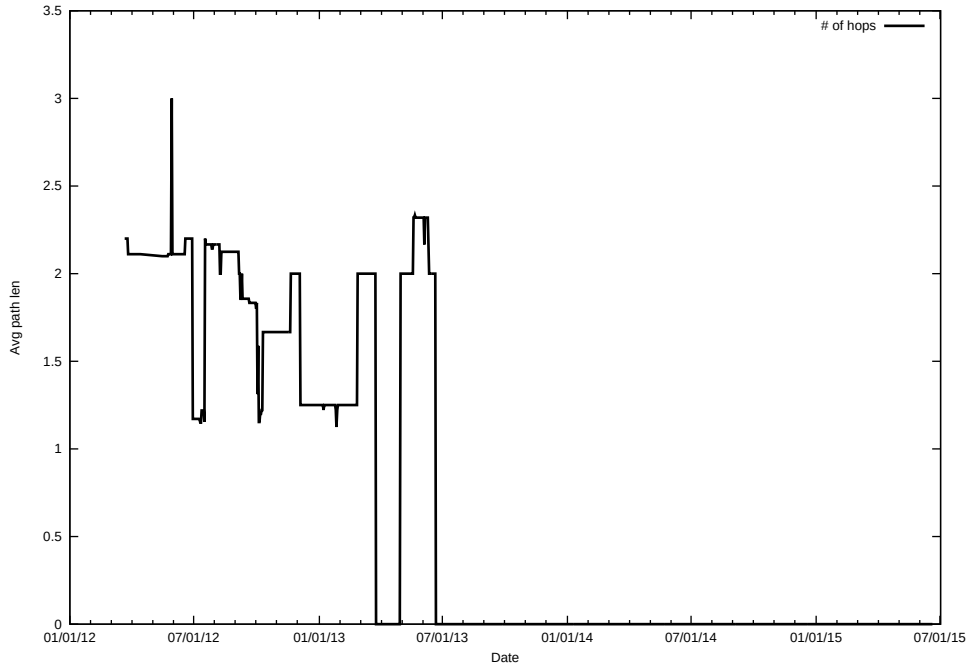
/26


# of hops
Avg path len
Date
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15

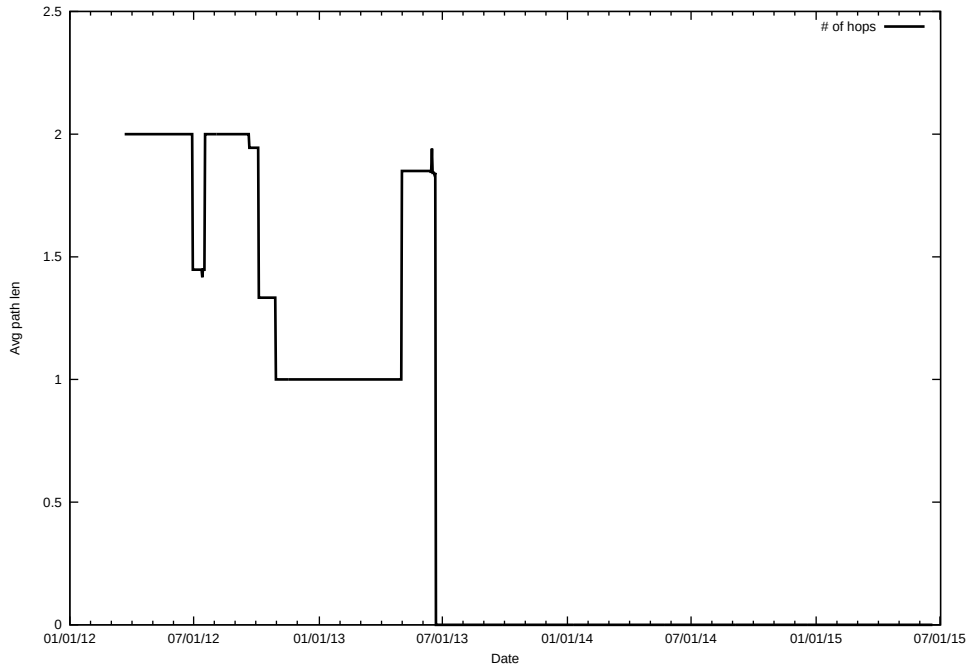
/27


3.5
3
2.5
2
1.5
1
0.5
0
Avg path len
# of hops
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

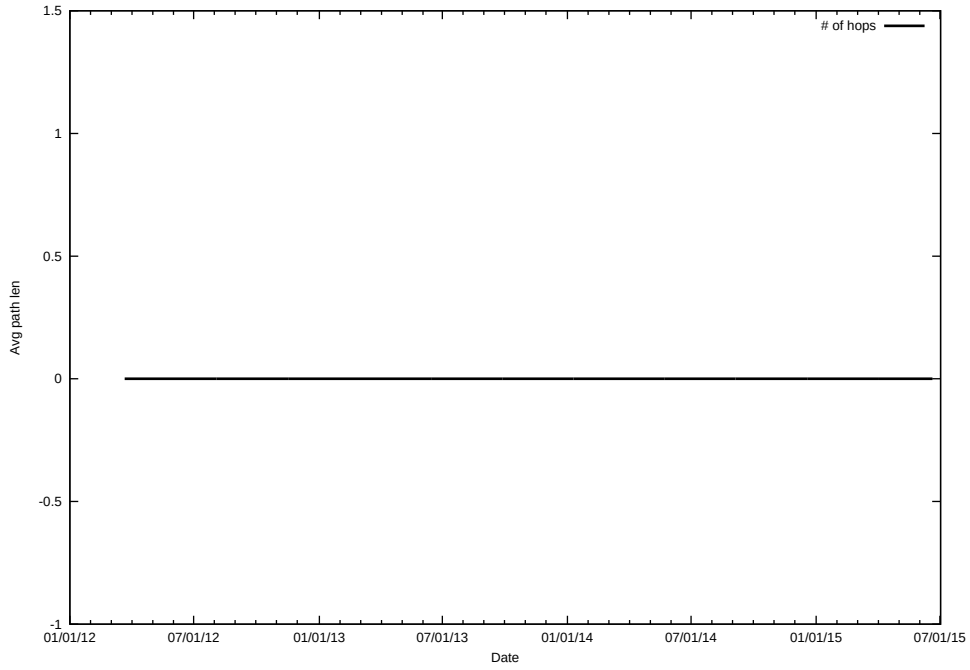
/28


3.5
3
2.5
2
1.5
1
0.5
0
Avg path len
# of hops
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

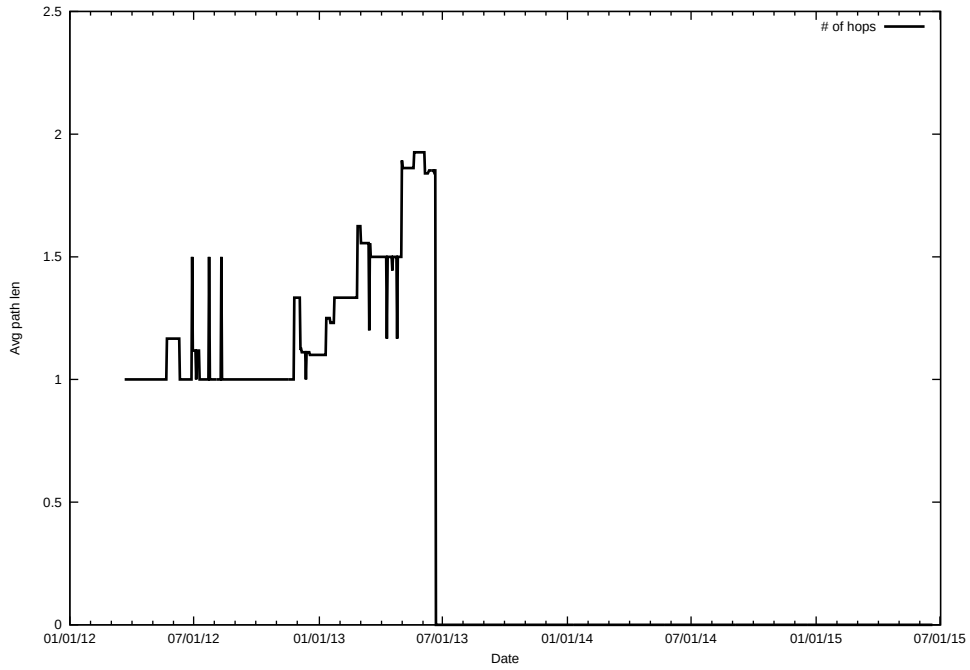
/29


2.5
2
1.5
1
0.5
0
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

/30

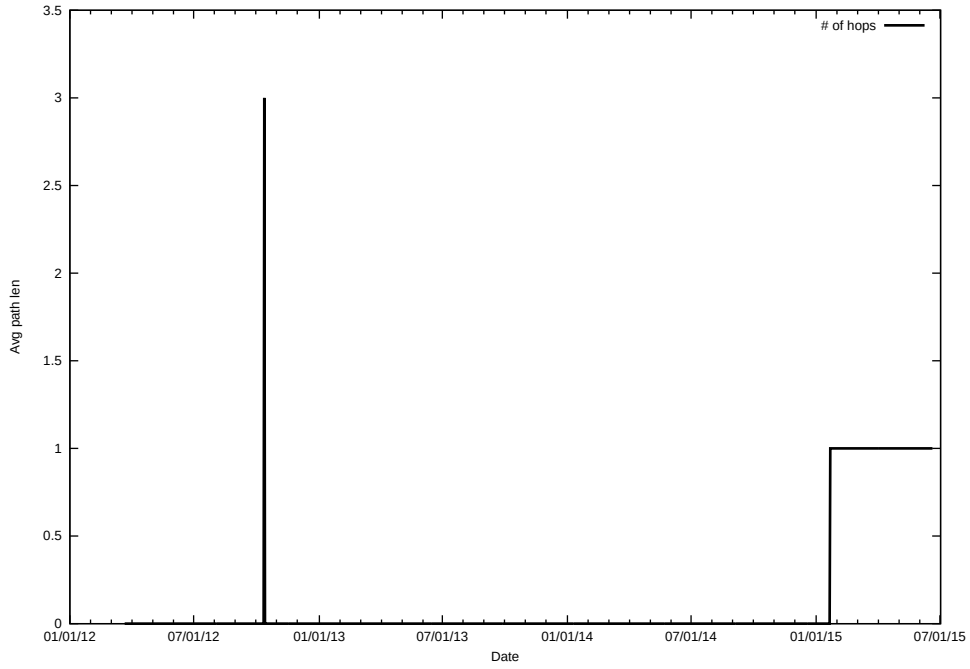

1.5
1
0.5
0
-0.5
-1
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

/31

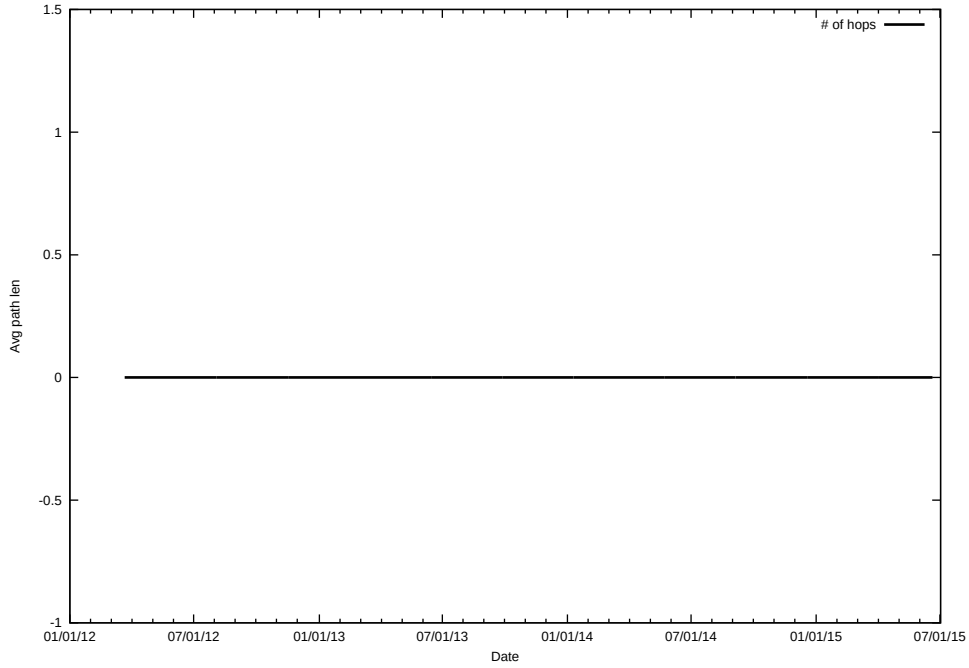


/32

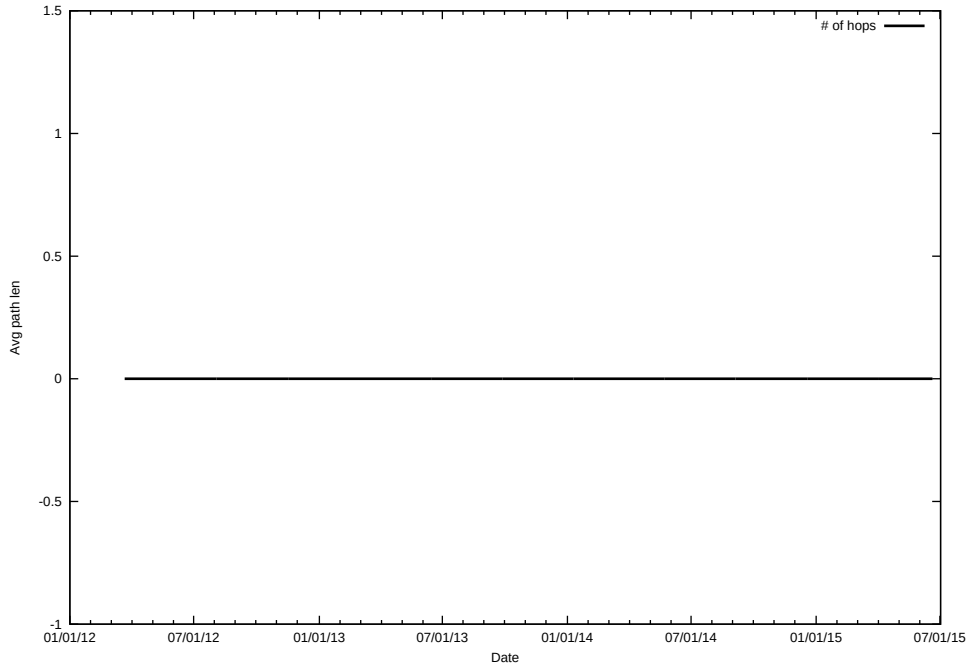
A.5 IPv6 BGP timeline of path length per prefix length



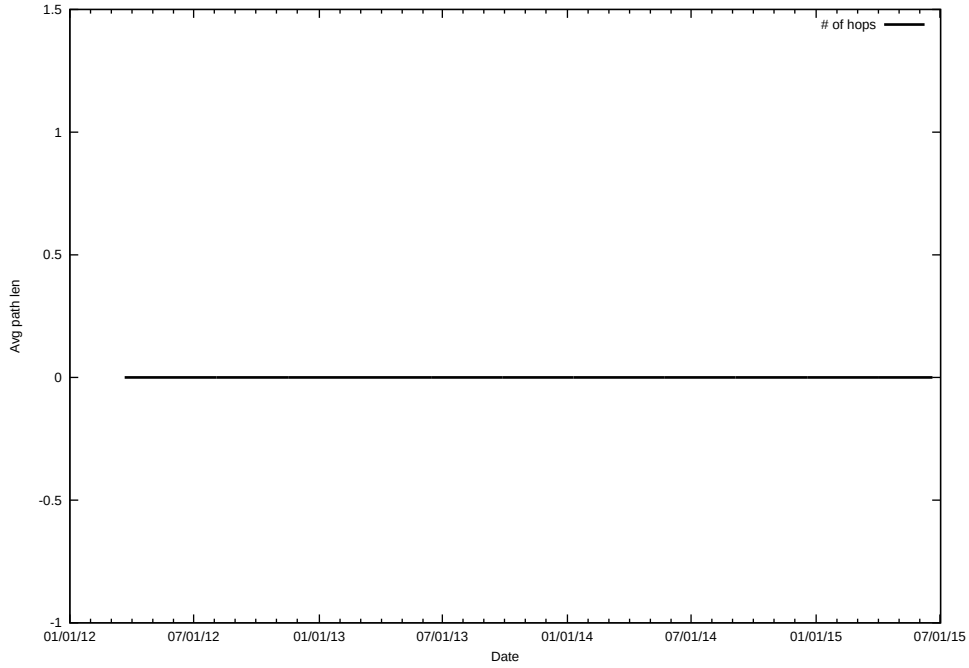
/0


1.5
1
0.5
0
-0.5
-1
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

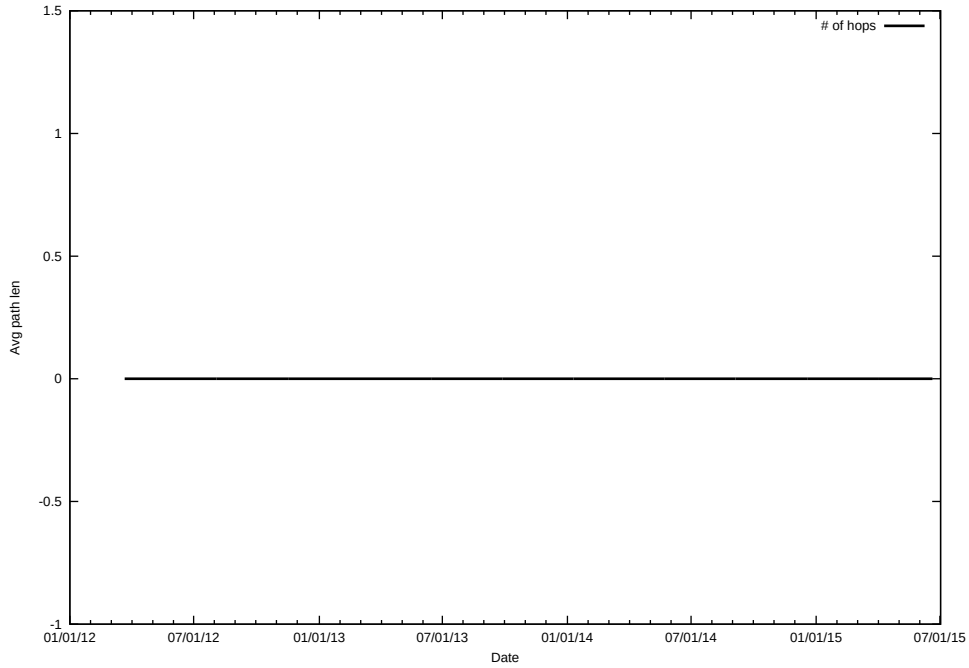
/1


1.5
1
0.5
0
-0.5
-1
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

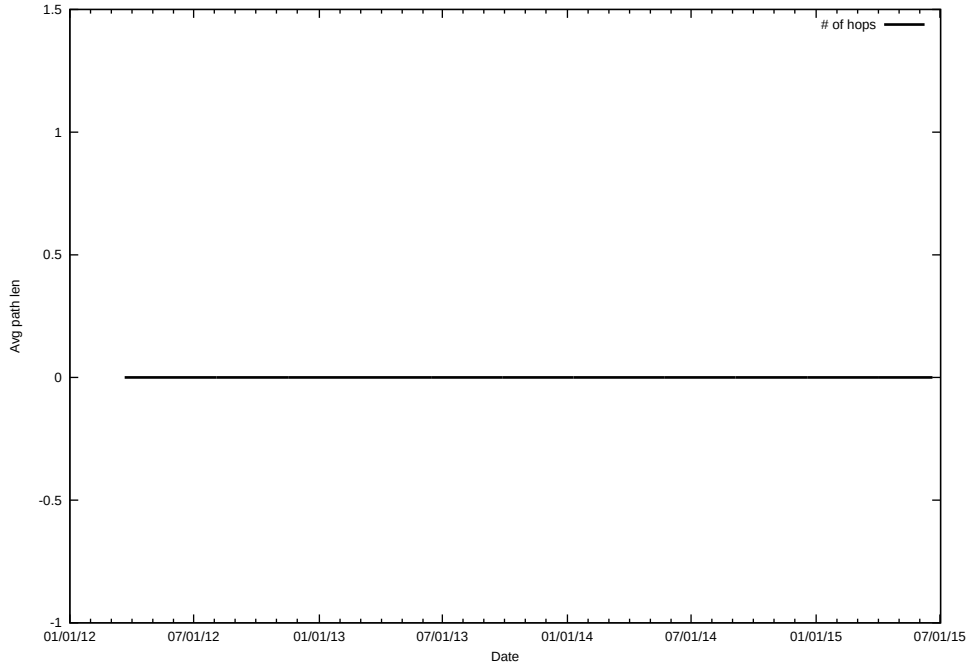
/2


1.5
1
0.5
0
-0.5
-1
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

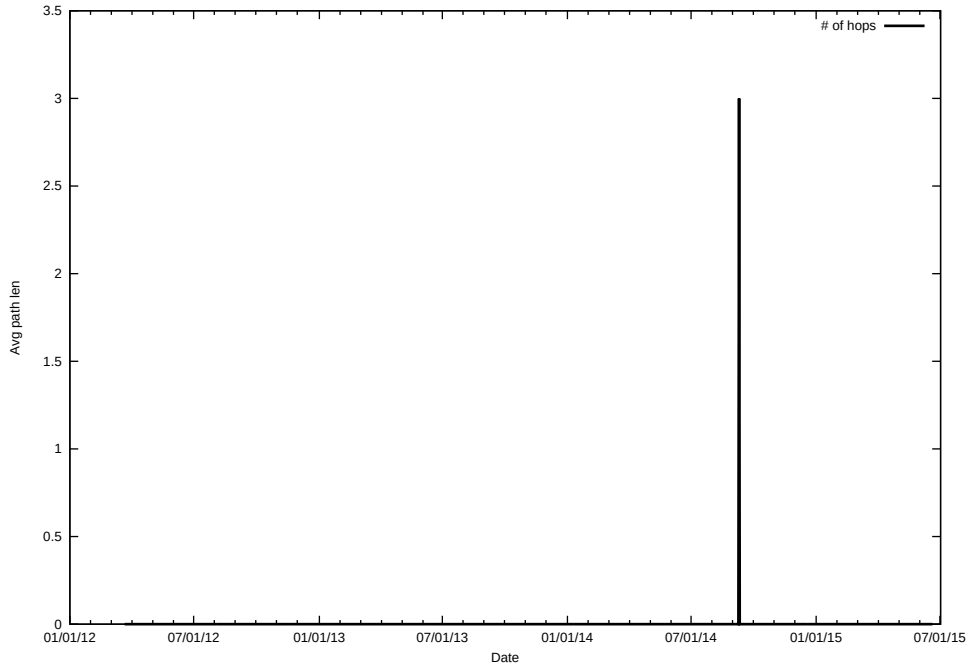
/3


1.5
1
0.5
0
-0.5
-1
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

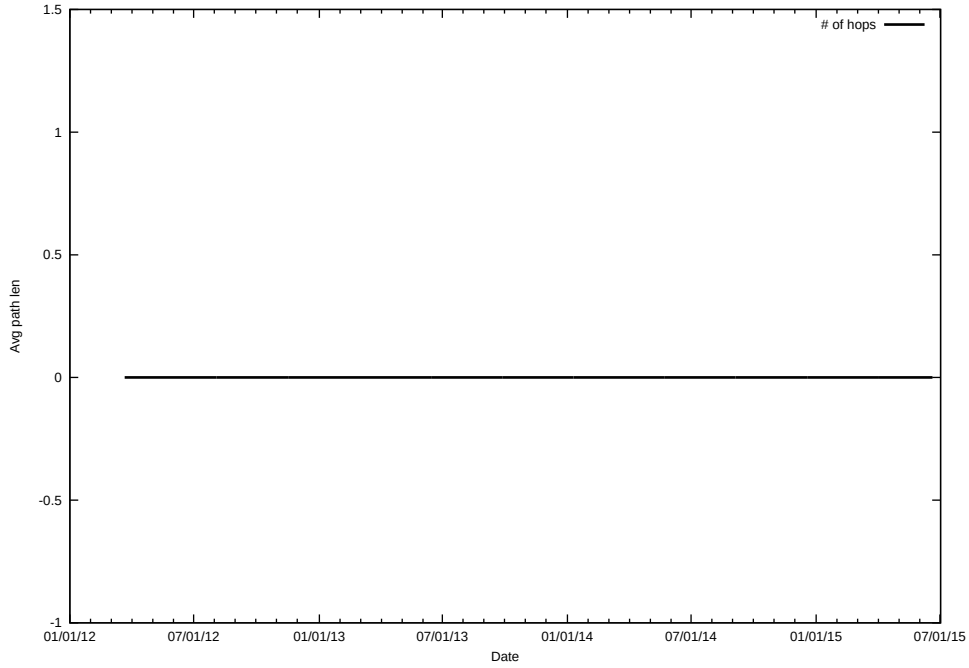
/4


1.5
1
0.5
0
-0.5
-1
Avg path len
# of hops
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

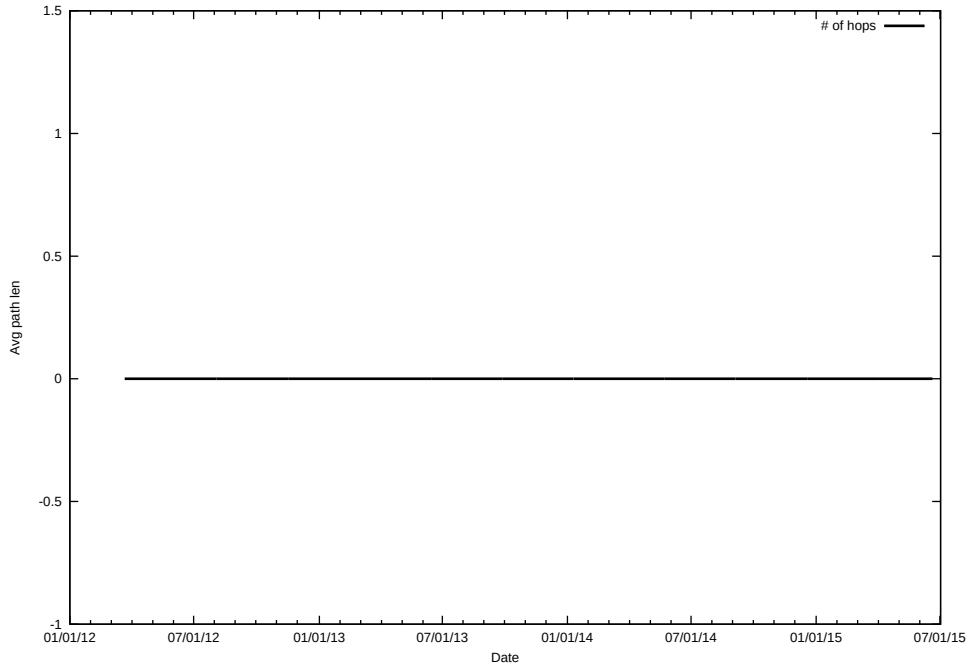
/5


3.5
3
2.5
2
1.5
1
0.5
0
Avg path len
# of hops
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

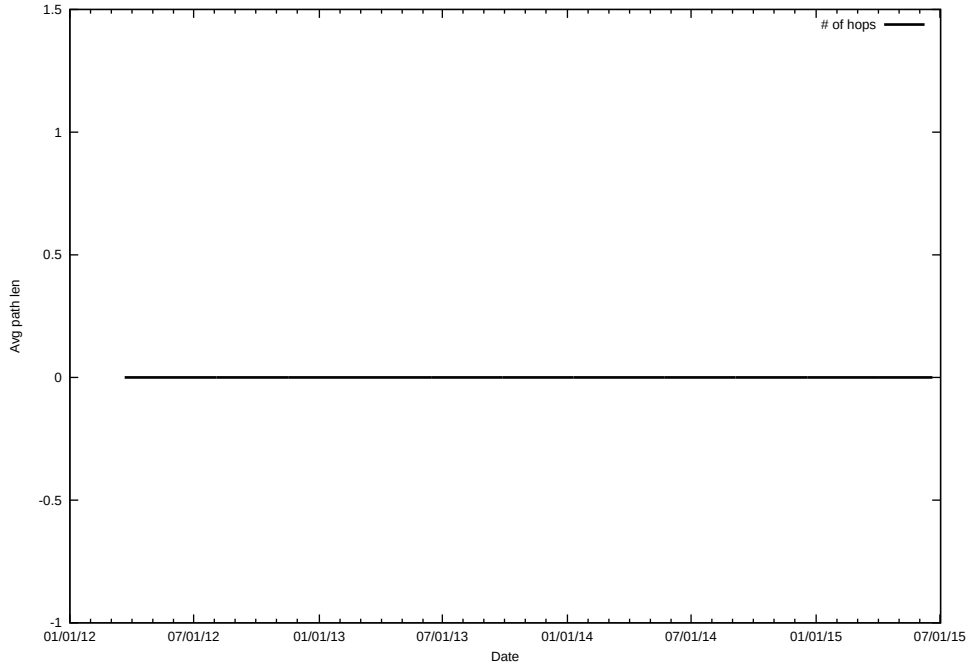
/6


1.5
1
0.5
0
-0.5
-1
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

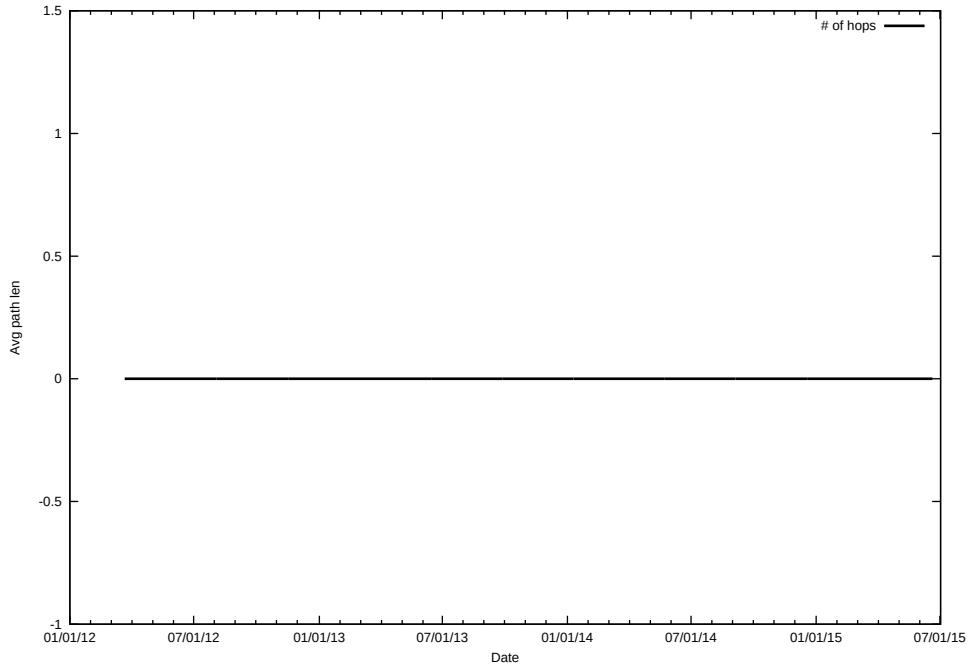
/7


1.5
1
0.5
0
-0.5
-1
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

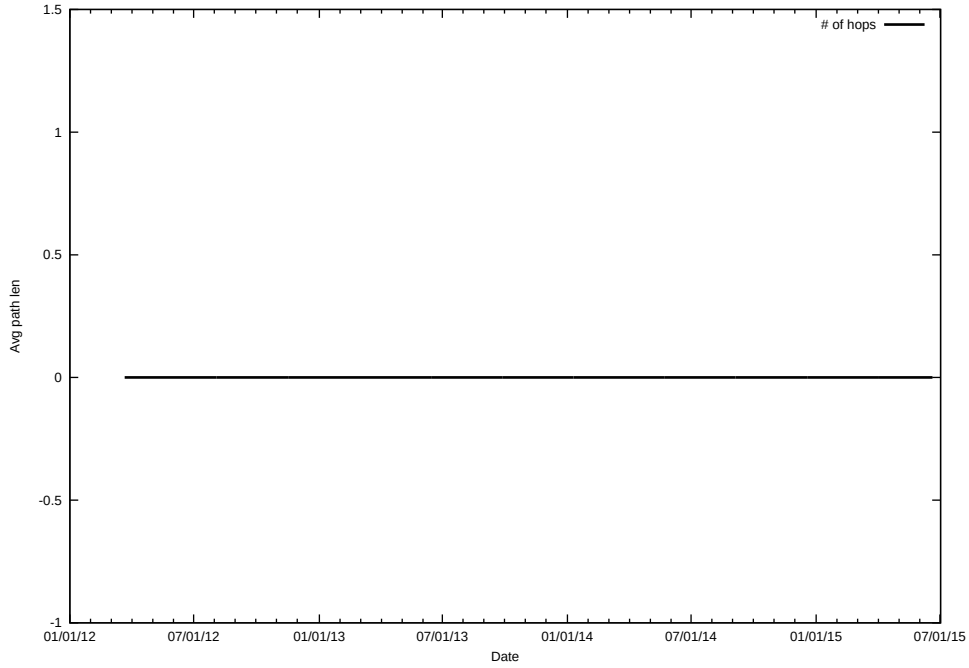
/8


1.5
1
0.5
0
-0.5
-1
Avg path len
# of hops
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

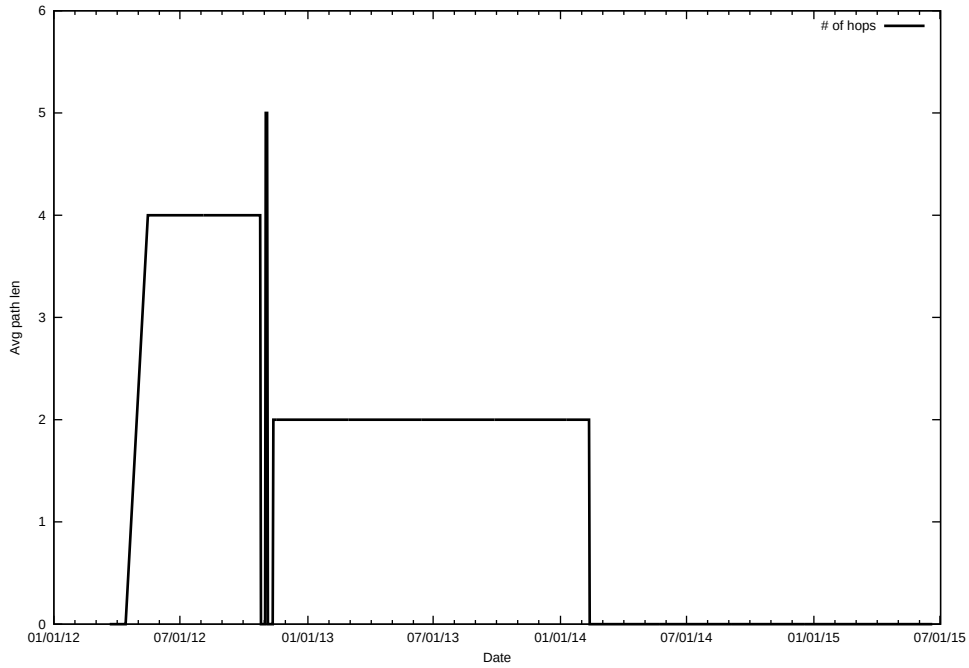
/9


1.5
1
0.5
0
-0.5
-1
Avg path len
# of hops
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

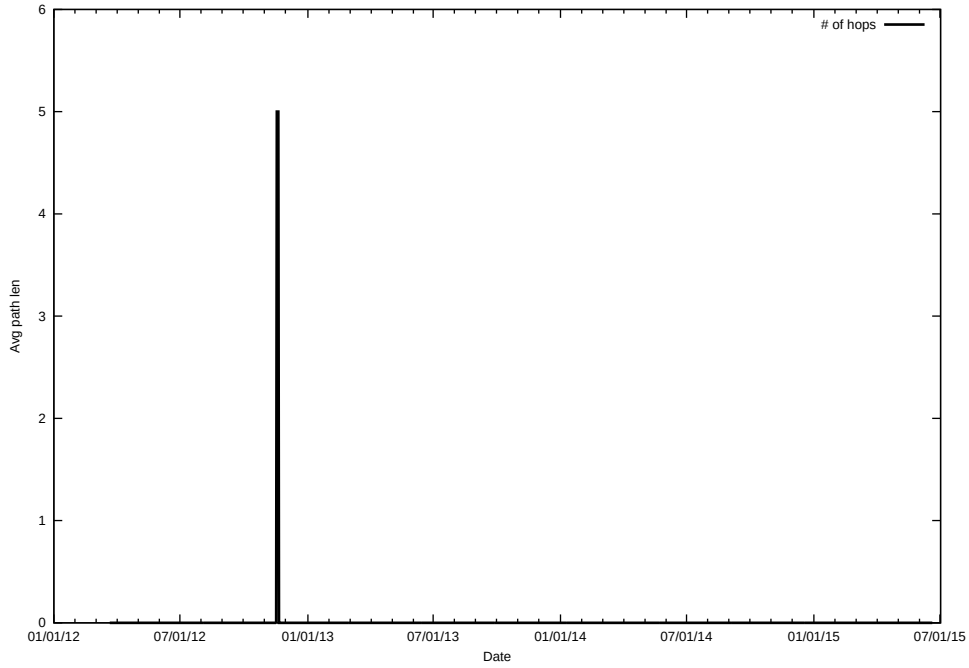
/10


1.5
1
0.5
0
-0.5
-1
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

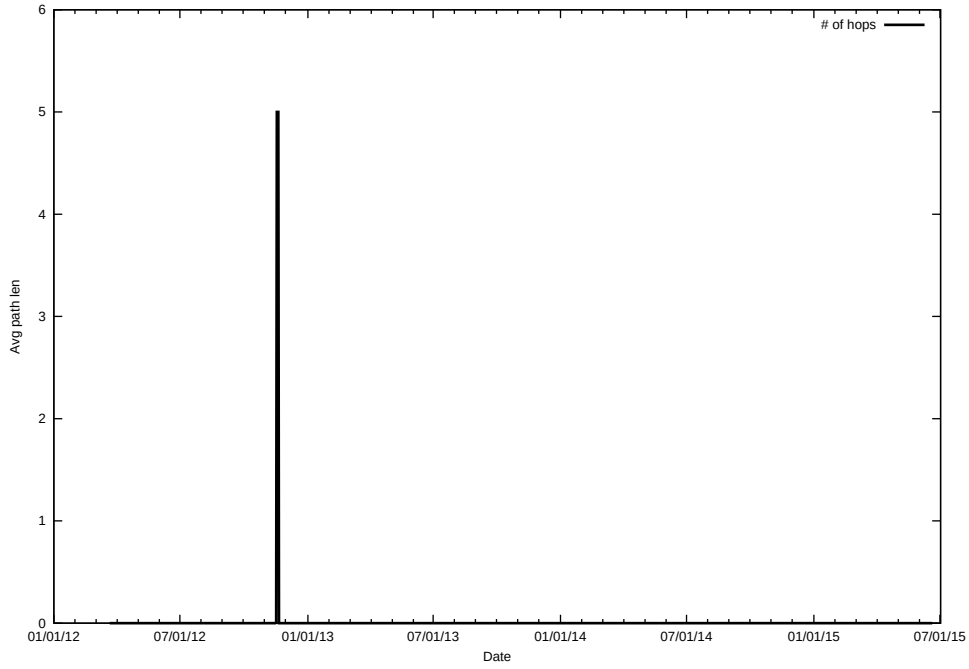
/11


6
5
4
3
2
1
0
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

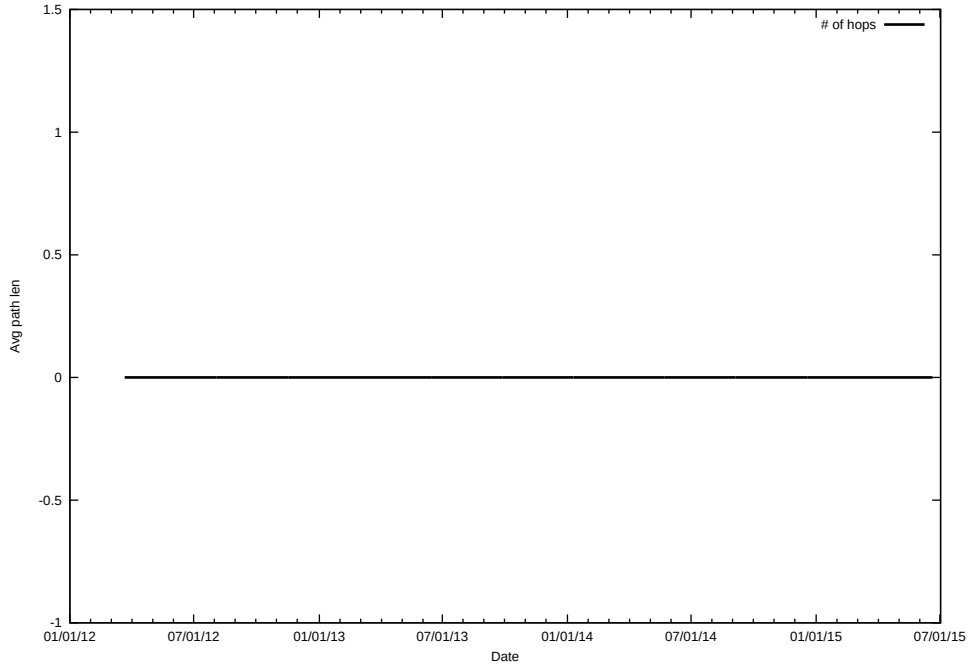
/12


6
5
4
3
2
1
0
Avg path len
# of hops
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

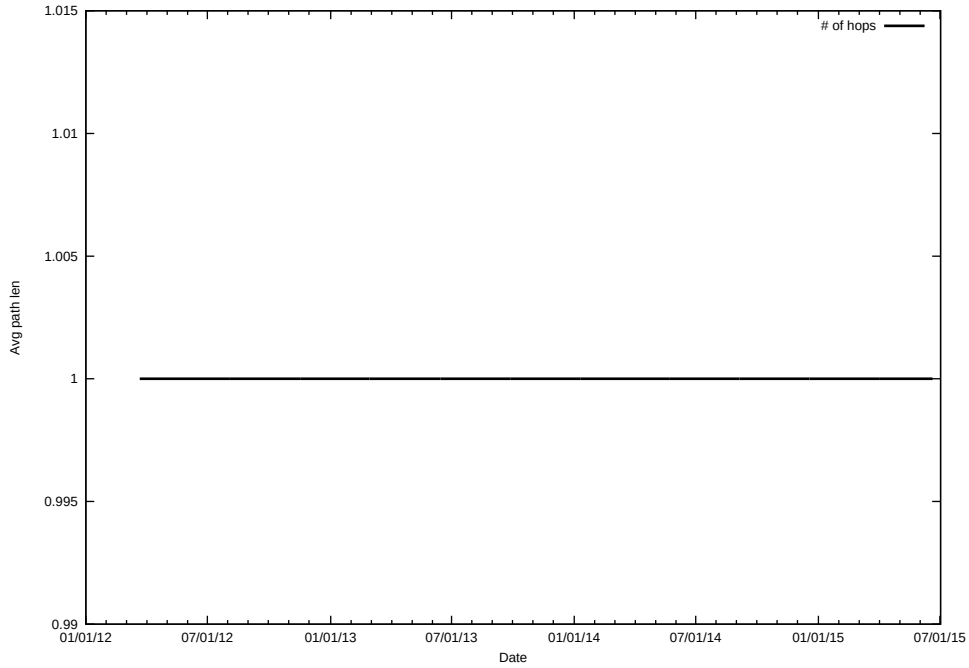
/13


6
5
4
3
2
1
0
Avg path len
# of hops
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

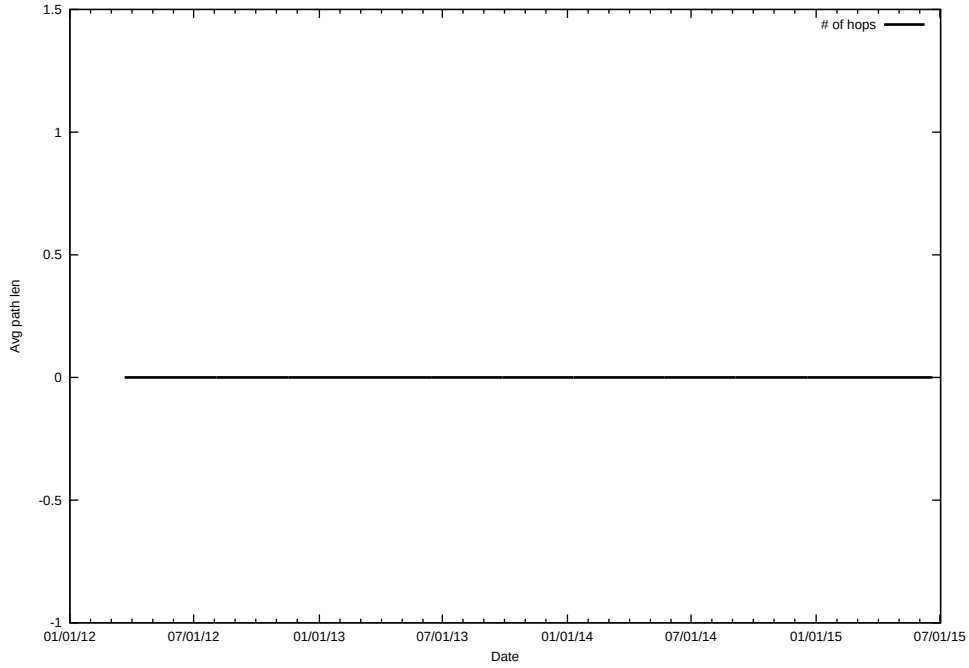
/14


1.5
1
0.5
0
-0.5
-1
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

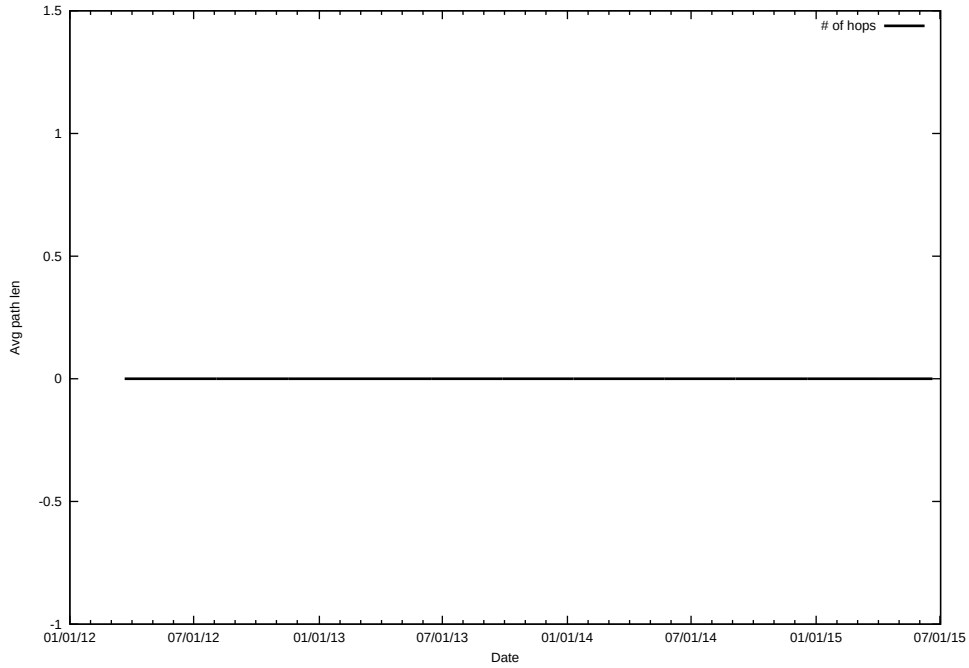
/15


1.015
1.01
1.005
1
0.995
0.99
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

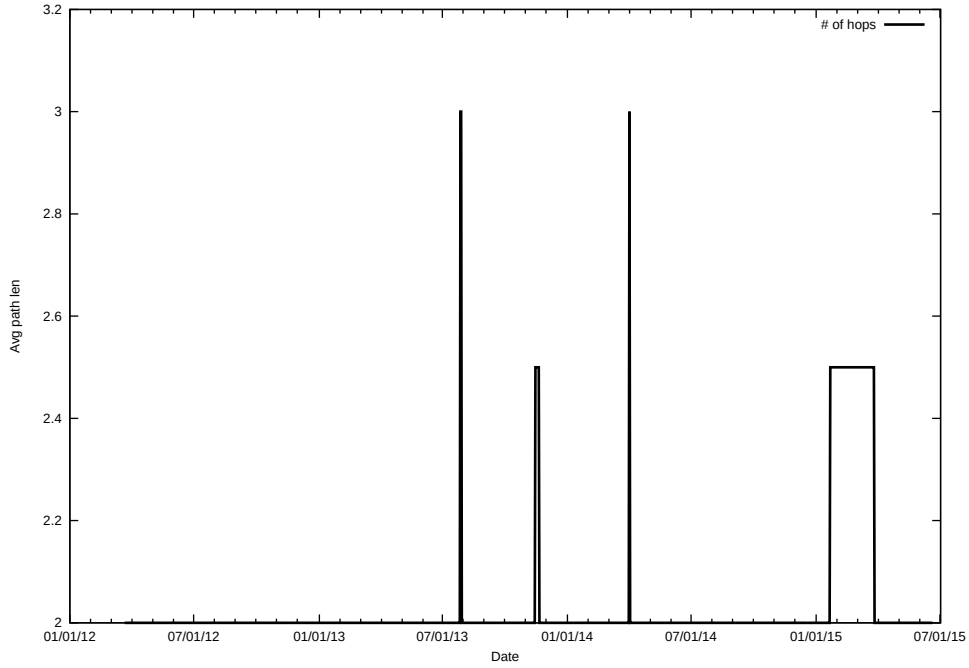
/16


1.5
1
0.5
0
-0.5
-1
Avg path len
# of hops
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

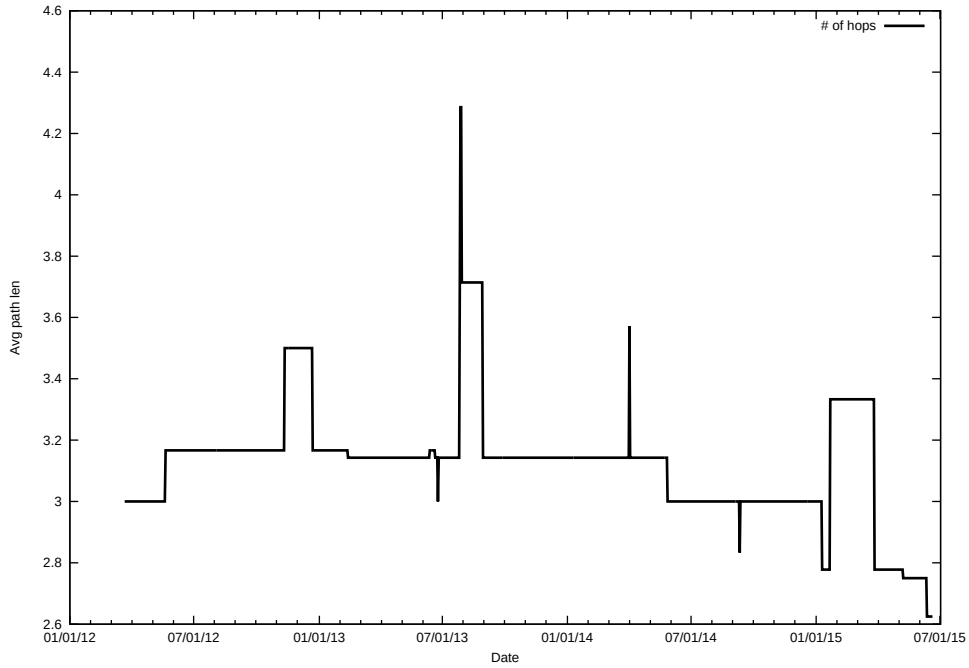
/17


1.5
1
0.5
0
-0.5
-1
Avg path len
# of hops
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

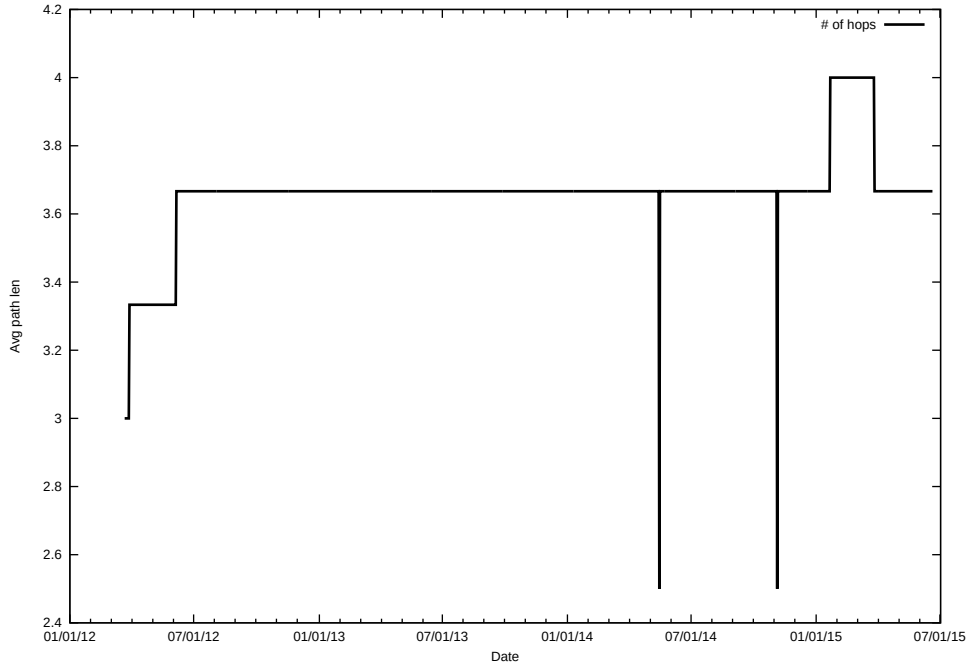
/18


3.2
3
2.8
2.6
2.4
2.2
2
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

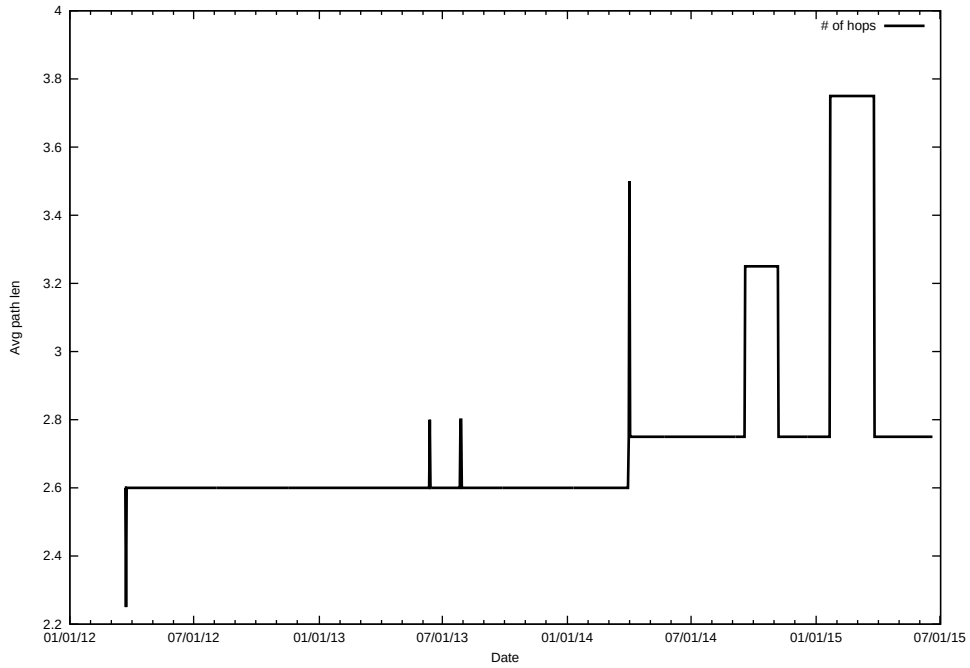
/19


4.6
4.4
4.2
4
3.8
3.6
3.4
3.2
3
2.8
2.6
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

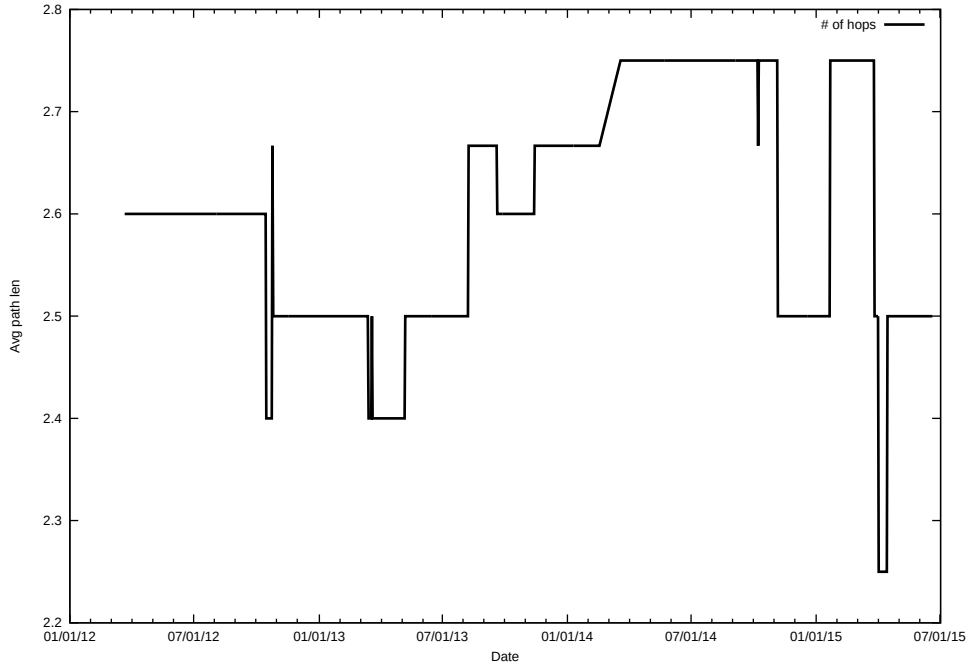
/20


# of hops
Avg path len
Date
4.2
4
3.8
3.6
3.4
3.2
3
2.8
2.6
2.4
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15

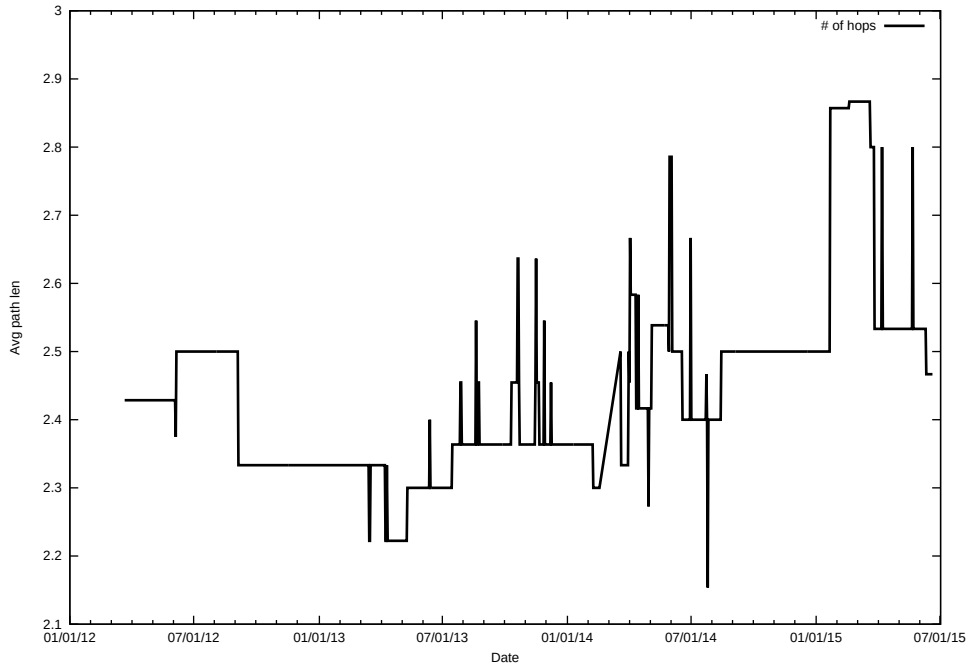
/21


# of hops
Avg path len
Date
4
3.8
3.6
3.4
3.2
3
2.8
2.6
2.4
2.2
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15

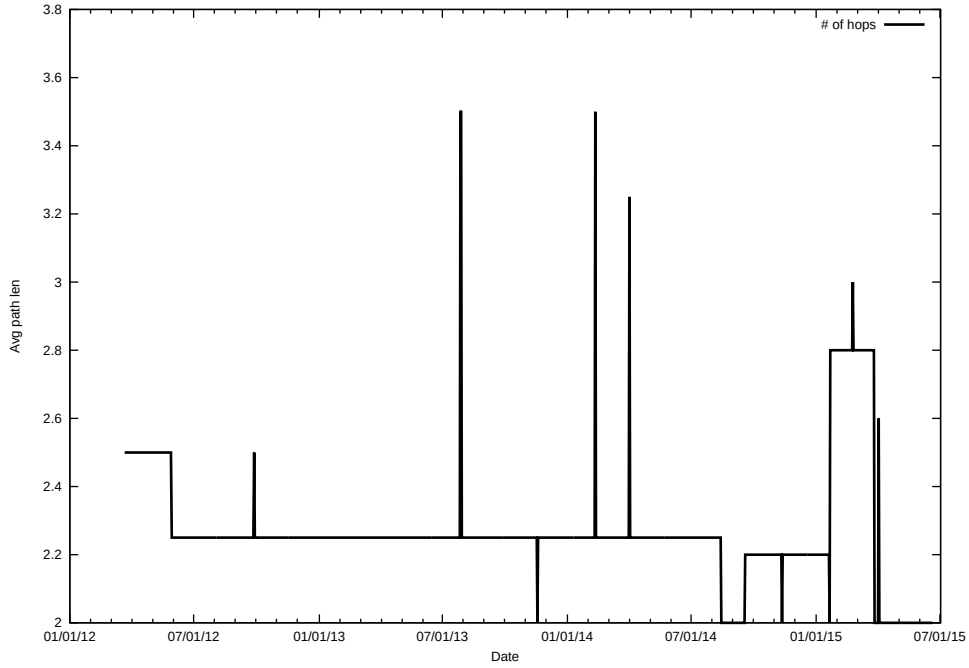
/22


2.8
2.7
2.6
2.5
2.4
2.3
2.2
Avg path len
# of hops
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

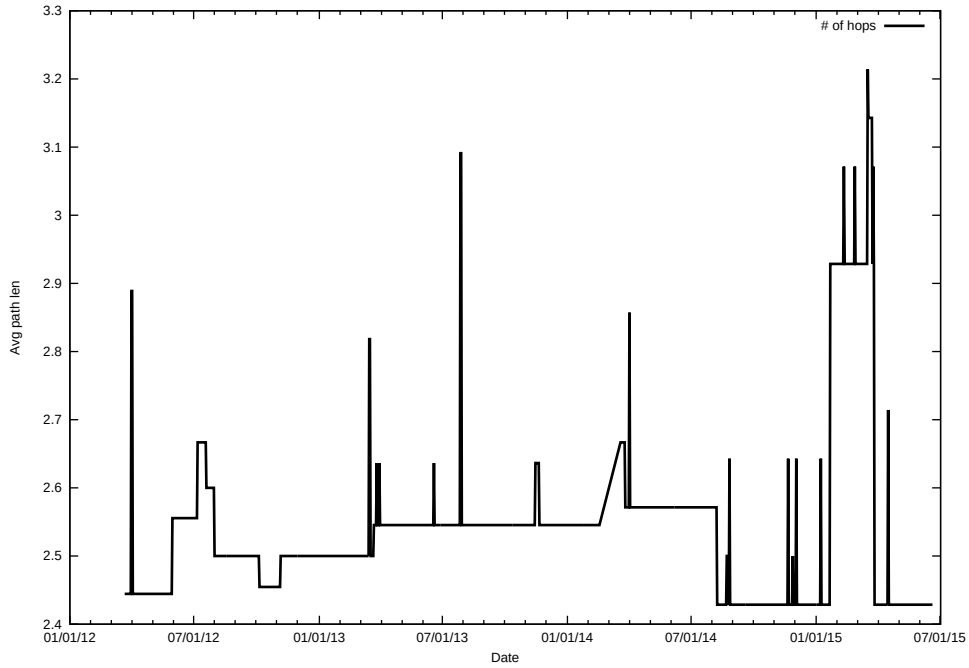
/23


3
2.9
2.8
2.7
2.6
2.5
2.4
2.3
2.2
2.1
Avg path len
# of hops
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

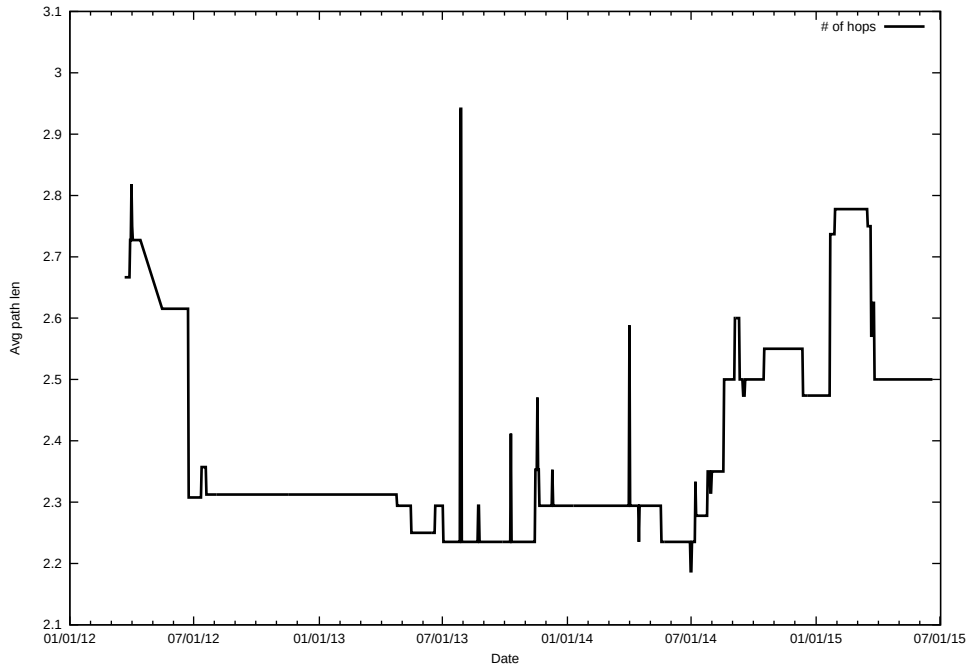
/24


3.8
3.6
3.4
3.2
3
2.8
2.6
2.4
2.2
2
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

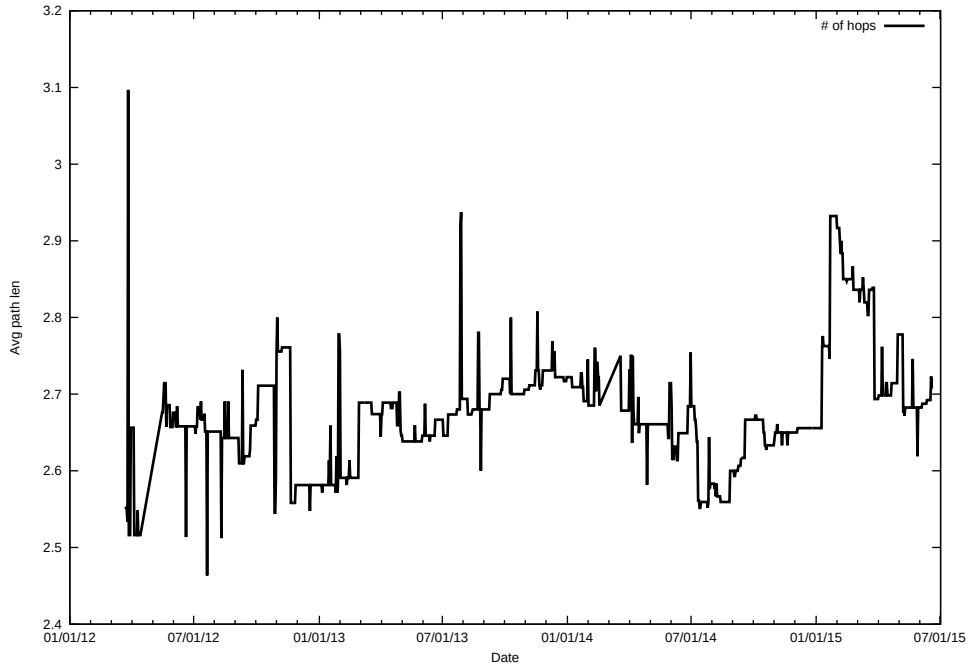
/25


3.3
3.2
3.1
3
2.9
2.8
2.7
2.6
2.5
2.4
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

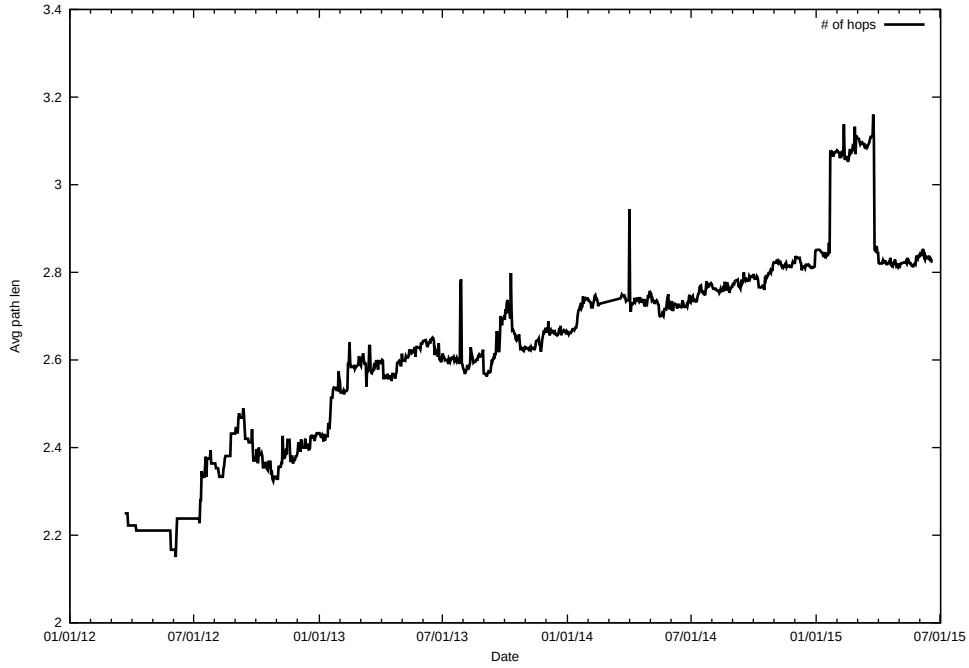
/26


3.1
3
2.9
2.8
2.7
2.6
2.5
2.4
2.3
2.2
2.1
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

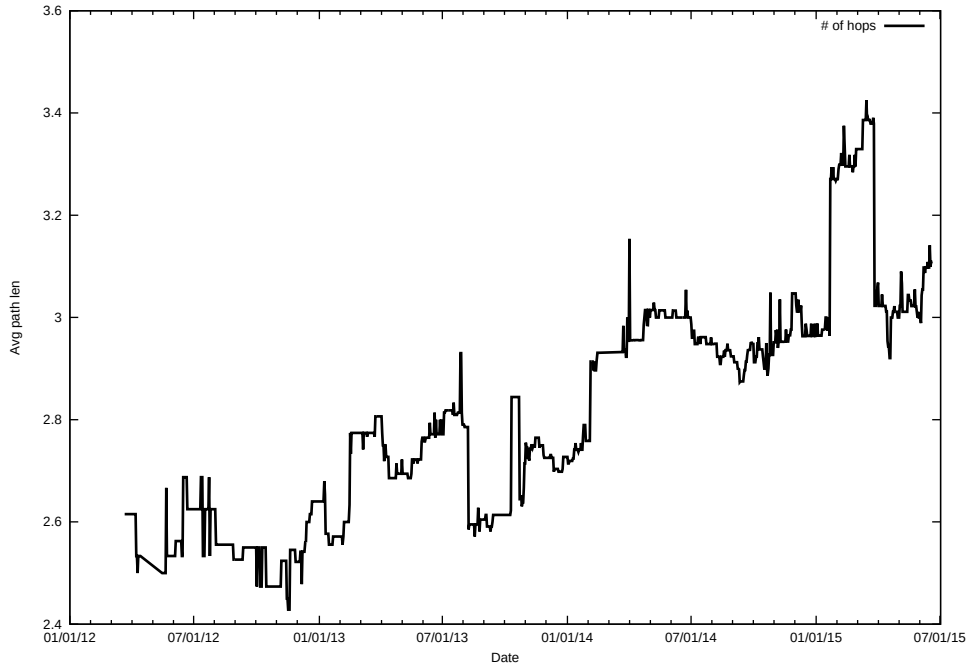
/27


3.2
3.1
3
2.9
2.8
2.7
2.6
2.5
2.4
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

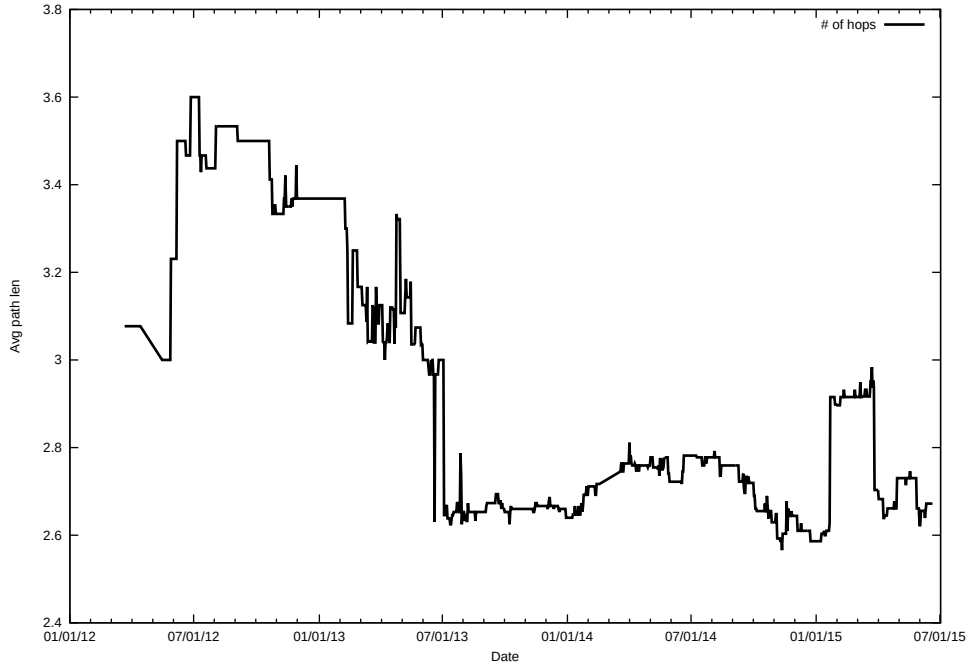
/28


3.4
3.2
3
2.8
2.6
2.4
2.2
2
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

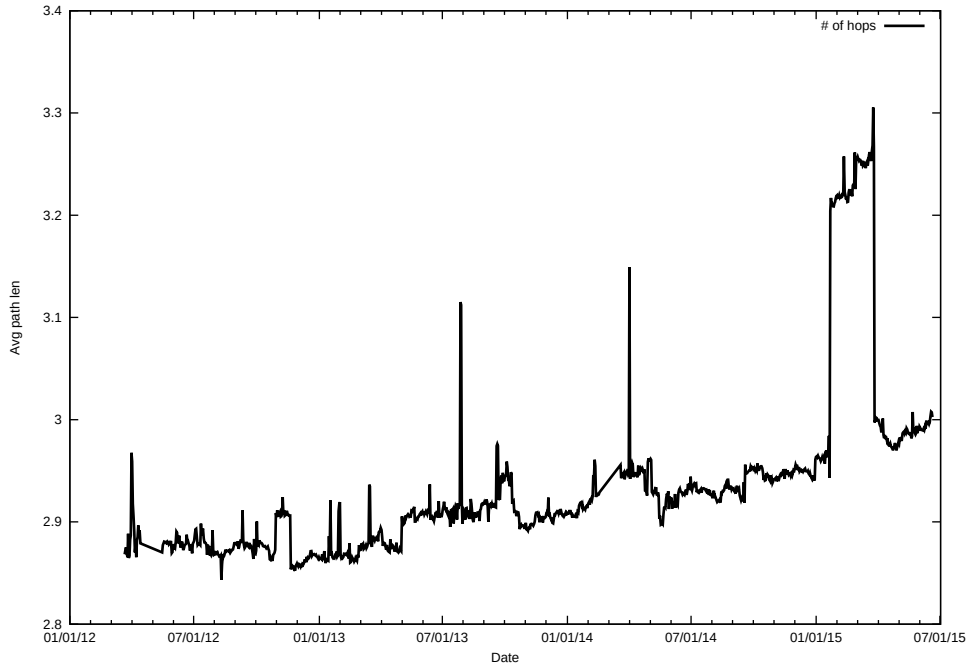
/29


3.6
3.4
3.2
3
2.8
2.6
2.4
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

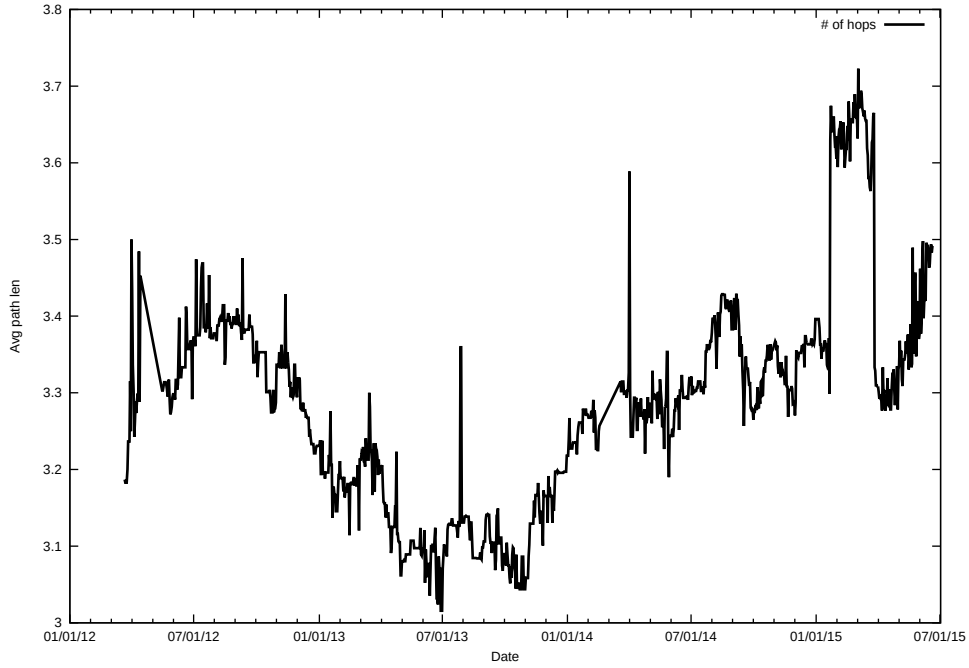
/30


3.8
3.6
3.4
3.2
3
2.8
2.6
2.4
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

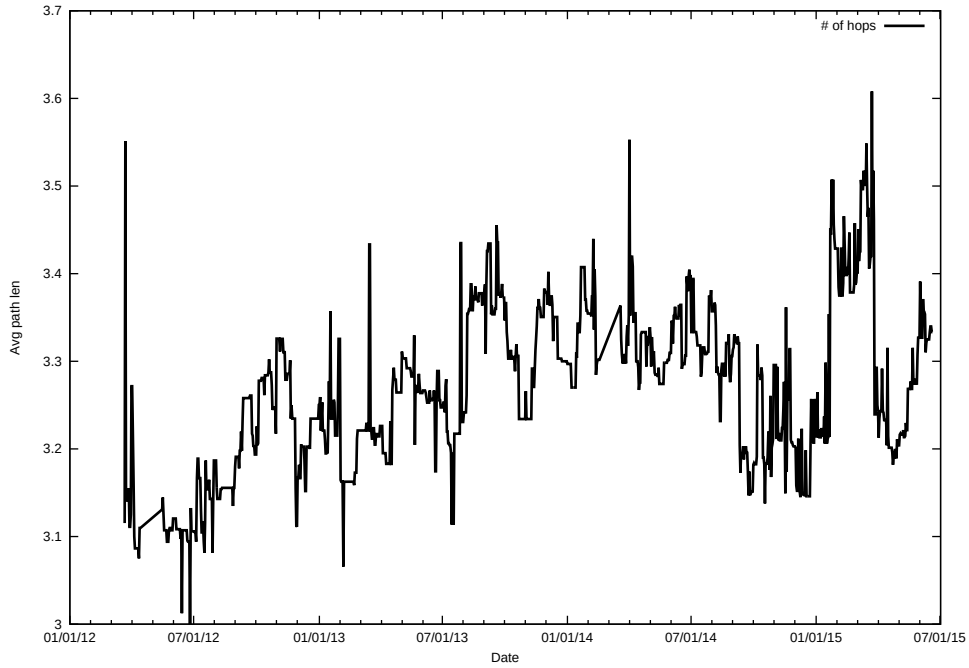
/31


3.4
3.3
3.2
3.1
3
2.9
2.8
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

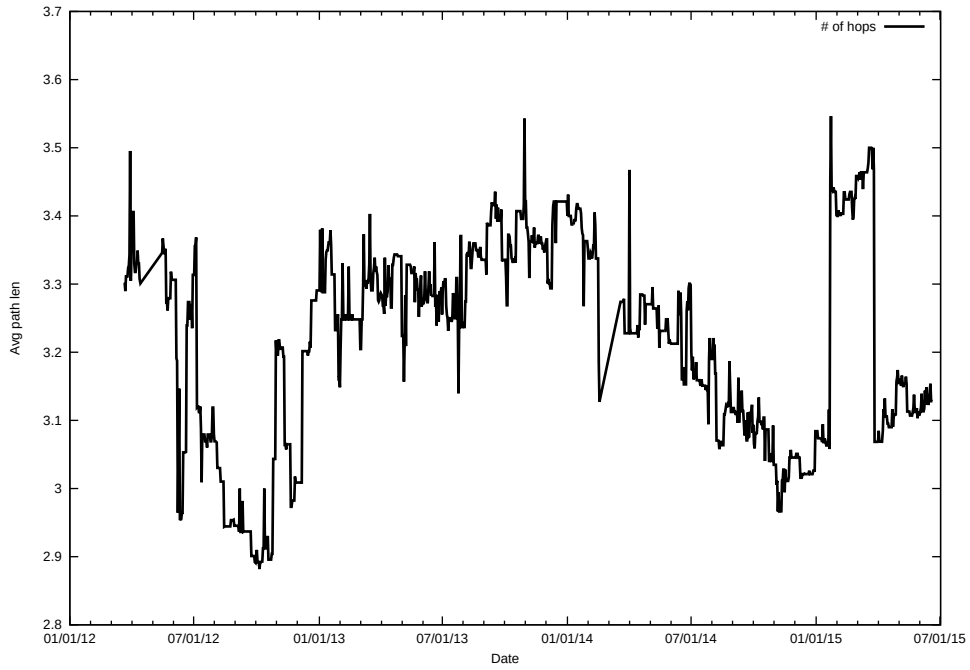
/32


3.8
3.7
3.6
3.5
3.4
3.3
3.2
3.1
3
Avg path len
# of hops
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

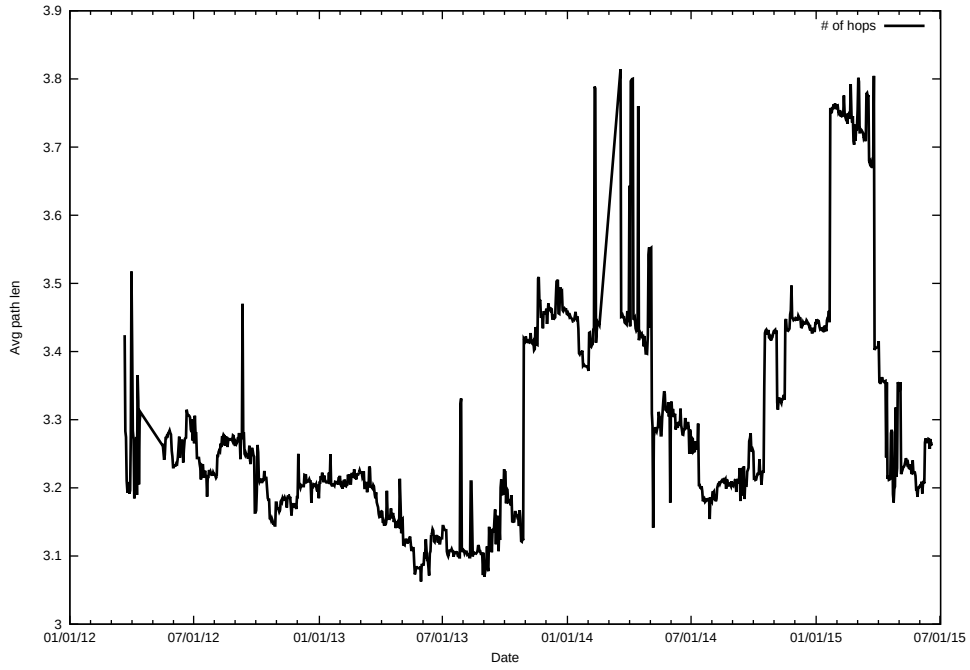
/33


3.7
3.6
3.5
3.4
3.3
3.2
3.1
3
Avg path len
# of hops
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

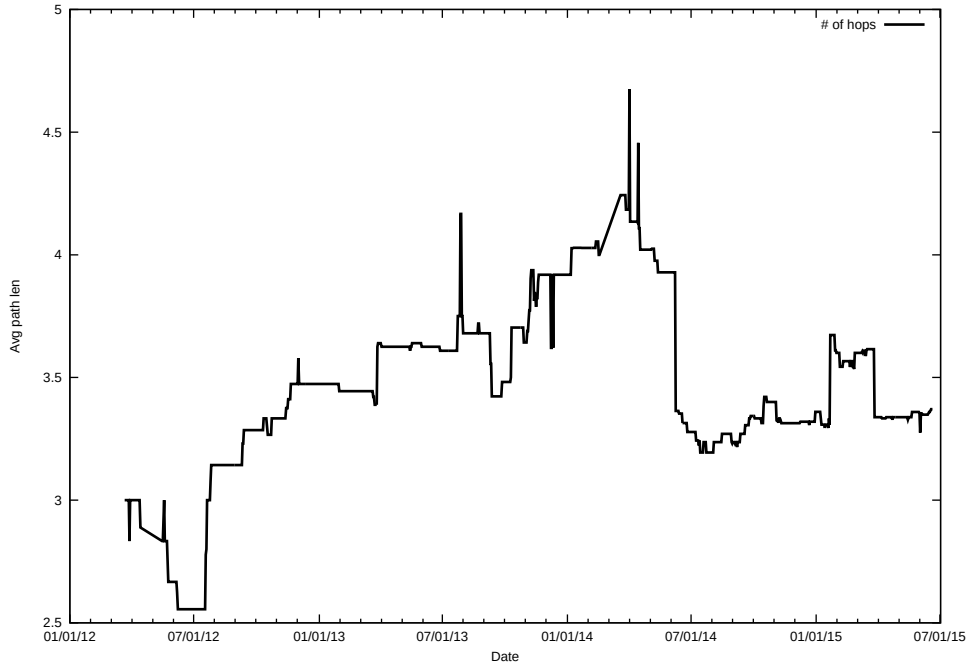
/34


3.7
3.6
3.5
3.4
3.3
3.2
3.1
3
2.9
2.8
Avg path len
# of hops
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

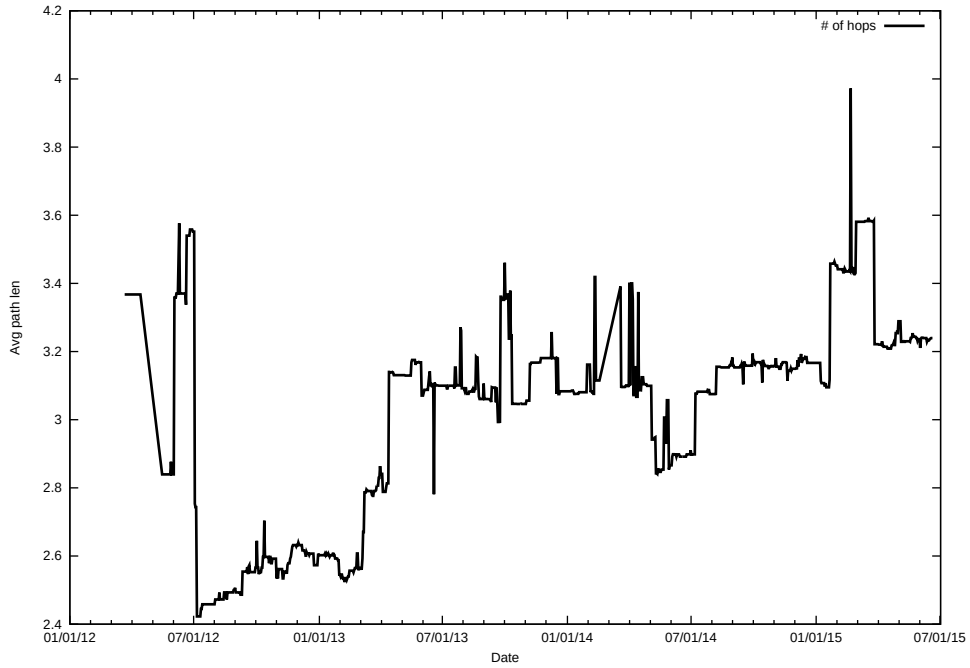
/35


3.9
3.8
3.7
3.6
3.5
3.4
3.3
3.2
3.1
3
Avg path len
# of hops
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

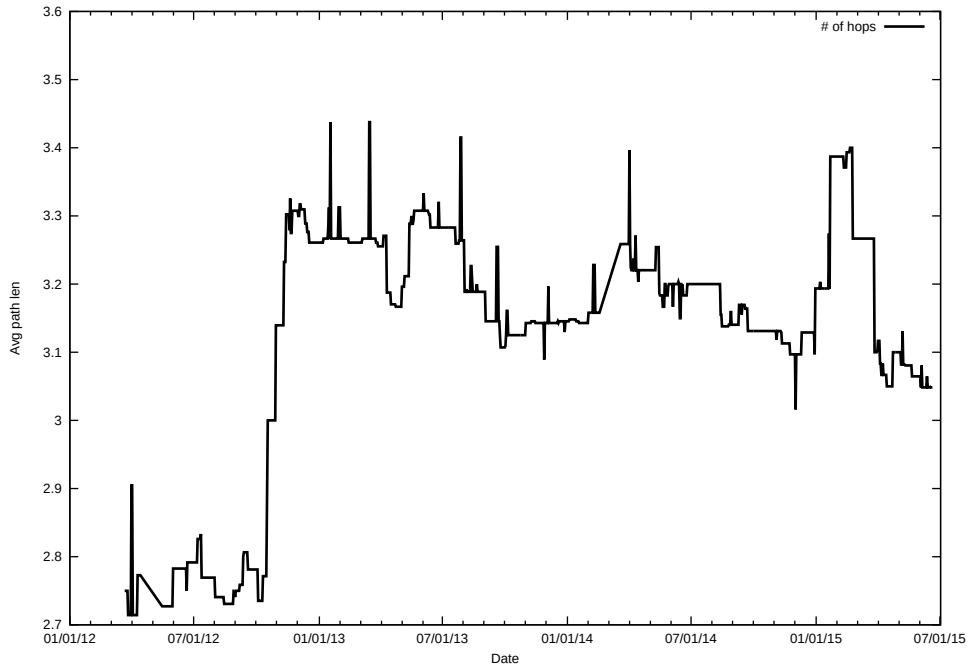
/36


5
4.5
4
3.5
3
2.5
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

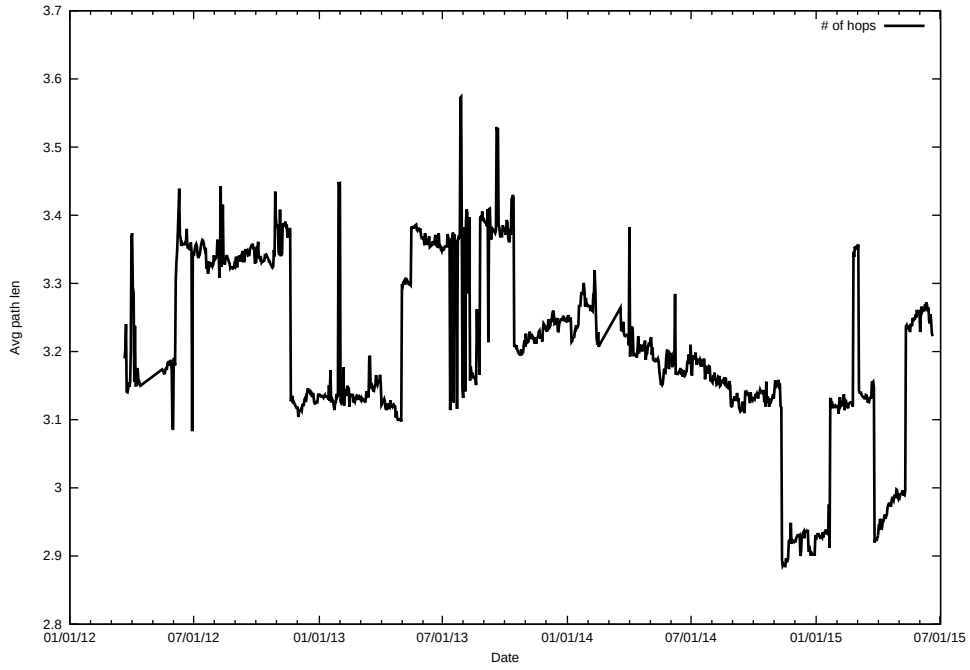
/37


4.2
4
3.8
3.6
3.4
3.2
3
2.8
2.6
2.4
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

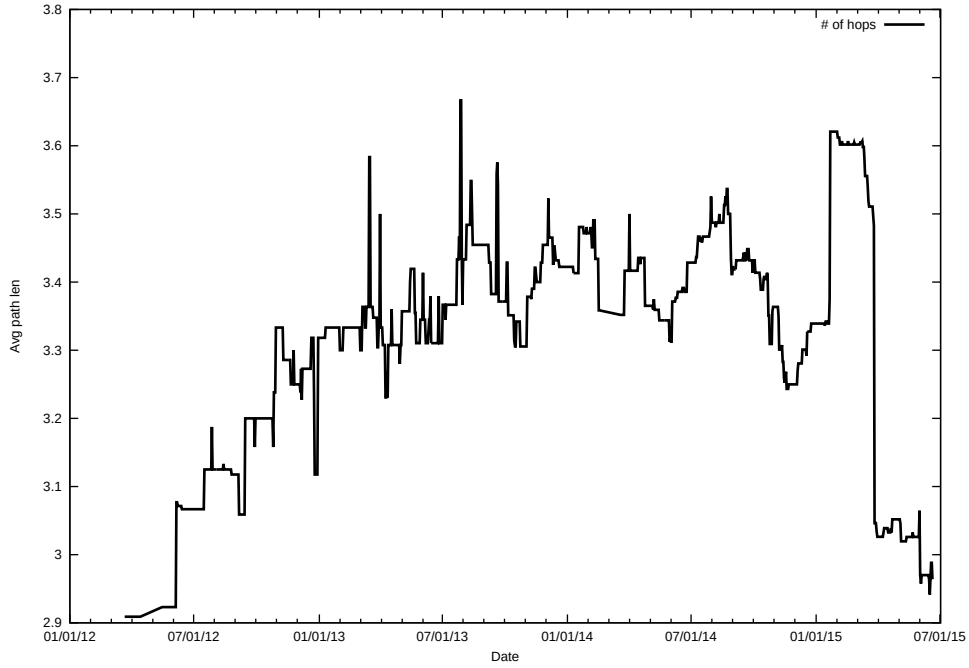
/38


# of hops
Avg path len
Date
3.6
3.5
3.4
3.3
3.2
3.1
3
2.9
2.8
2.7
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15

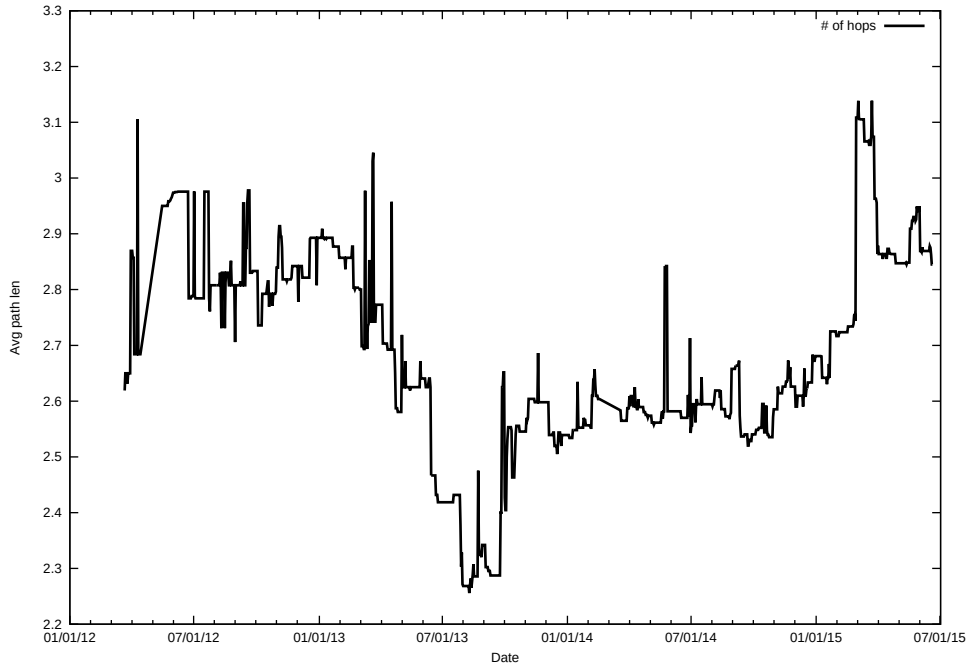
/39


# of hops
Avg path len
Date
3.7
3.6
3.5
3.4
3.3
3.2
3.1
3
2.9
2.8
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15

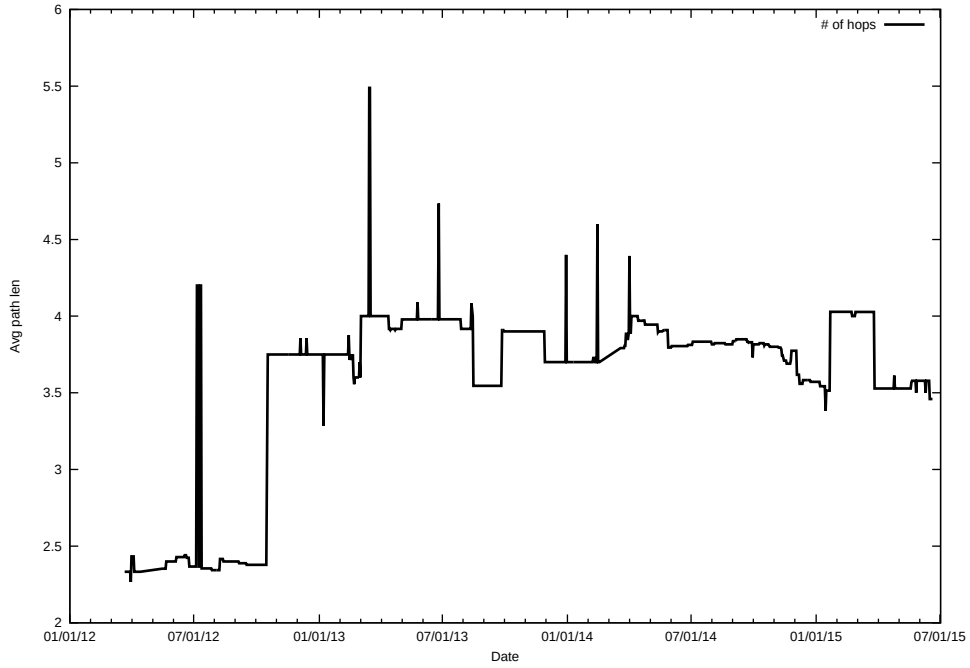
/40


# of hops
Avg path len
Date
3.8
3.7
3.6
3.5
3.4
3.3
3.2
3.1
3
2.9
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15

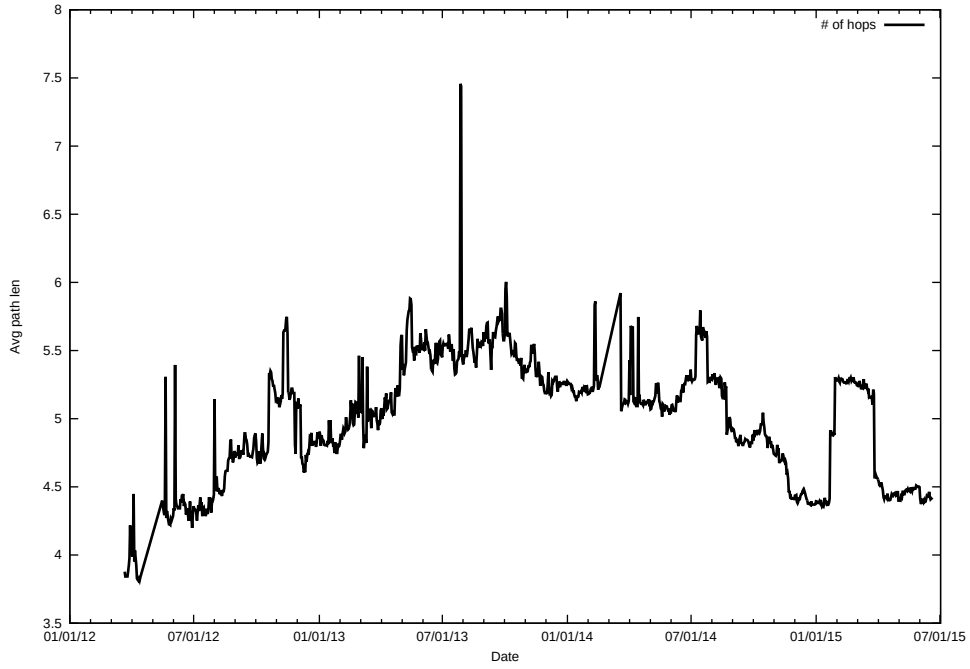
/41


# of hops
Avg path len
Date
3.3
3.2
3.1
3
2.9
2.8
2.7
2.6
2.5
2.4
2.3
2.2
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15

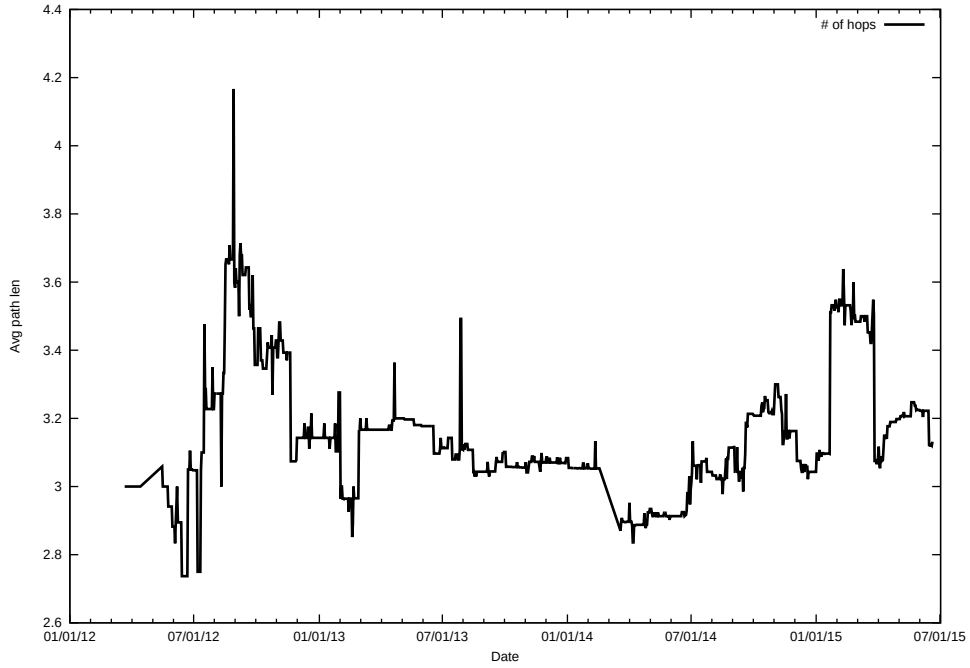
/42


Avg path len
Date
# of hops
6
5.5
5
4.5
4
3.5
3
2.5
2
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15

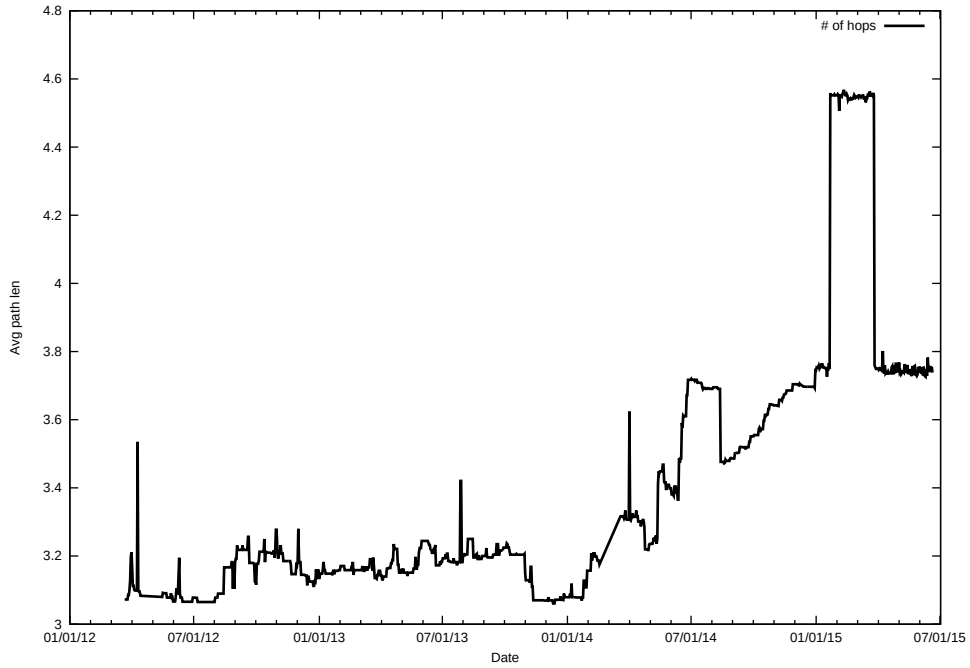
/43


Avg path len
Date
# of hops
8
7.5
7
6.5
6
5.5
5
4.5
4
3.5
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15

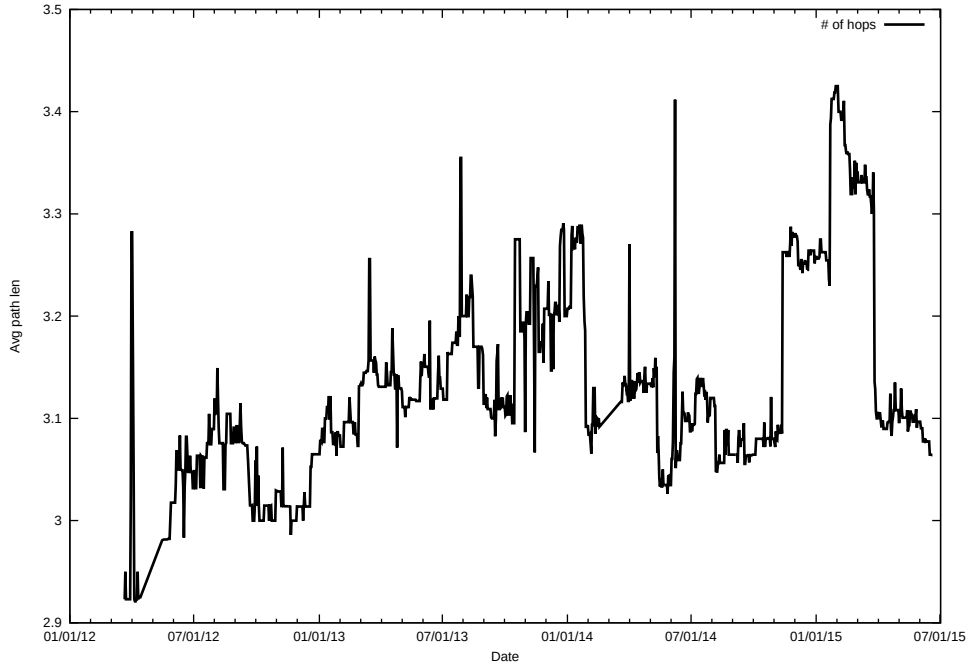
/44


# of hops
Avg path len
Date
4.4
4.2
4
3.8
3.6
3.4
3.2
3
2.8
2.6
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15

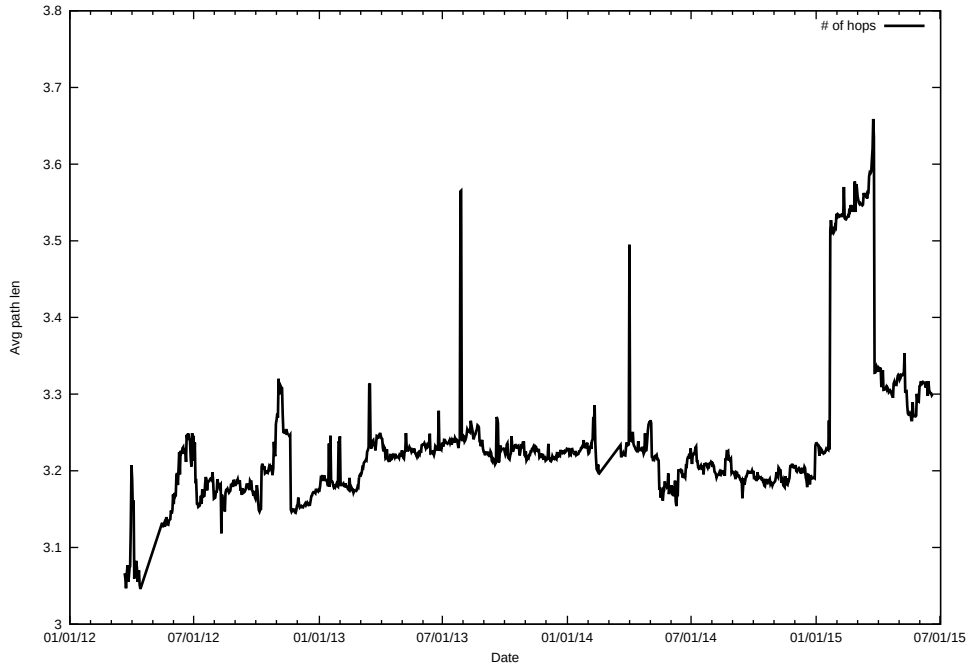
/45


# of hops
Avg path len
Date
4.8
4.6
4.4
4.2
4
3.8
3.6
3.4
3.2
3
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15

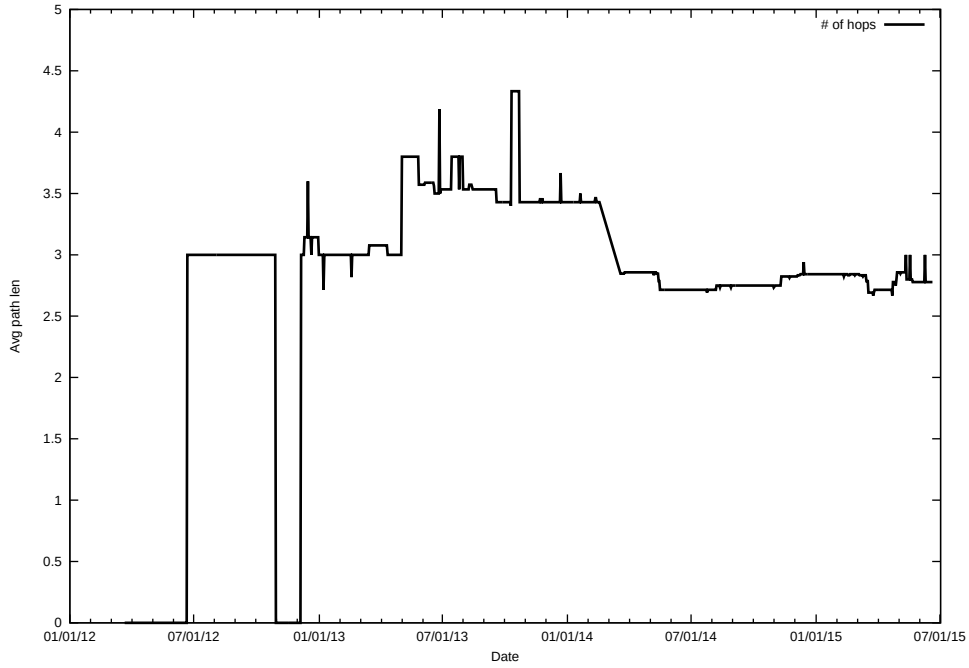
/46


3.5
3.4
3.3
3.2
3.1
3
2.9
Avg path len
# of hops
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

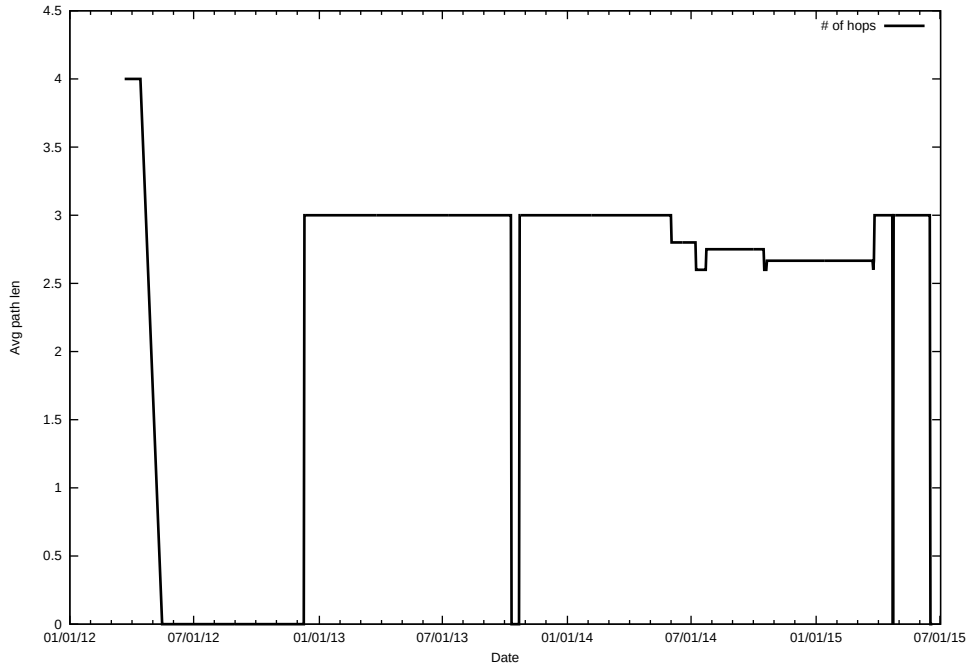
/47


3.8
3.7
3.6
3.5
3.4
3.3
3.2
3.1
3
Avg path len
# of hops
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

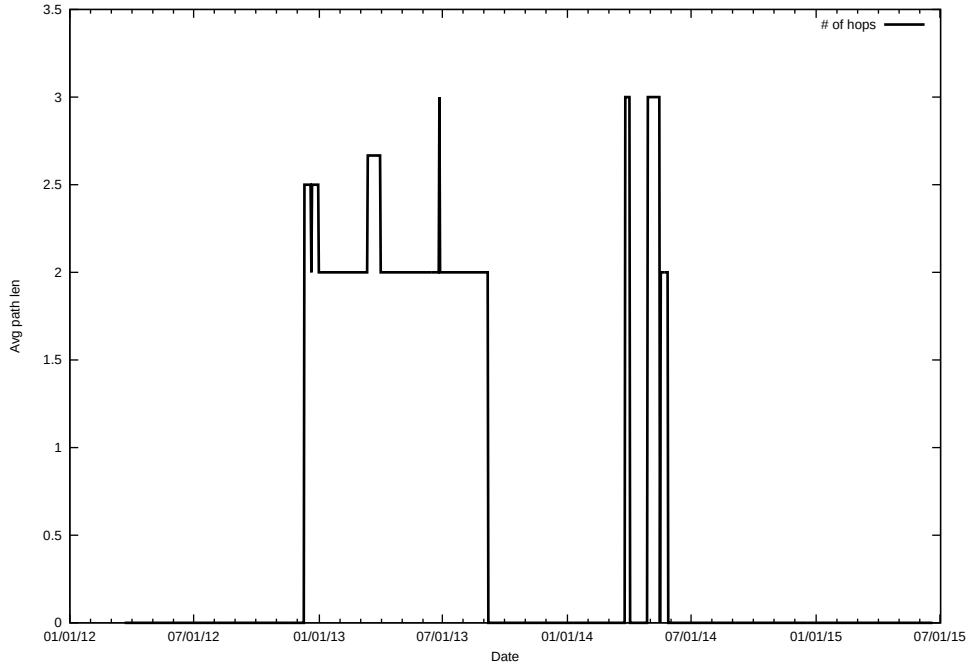
/48


Avg path len
Date
# of hops
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15

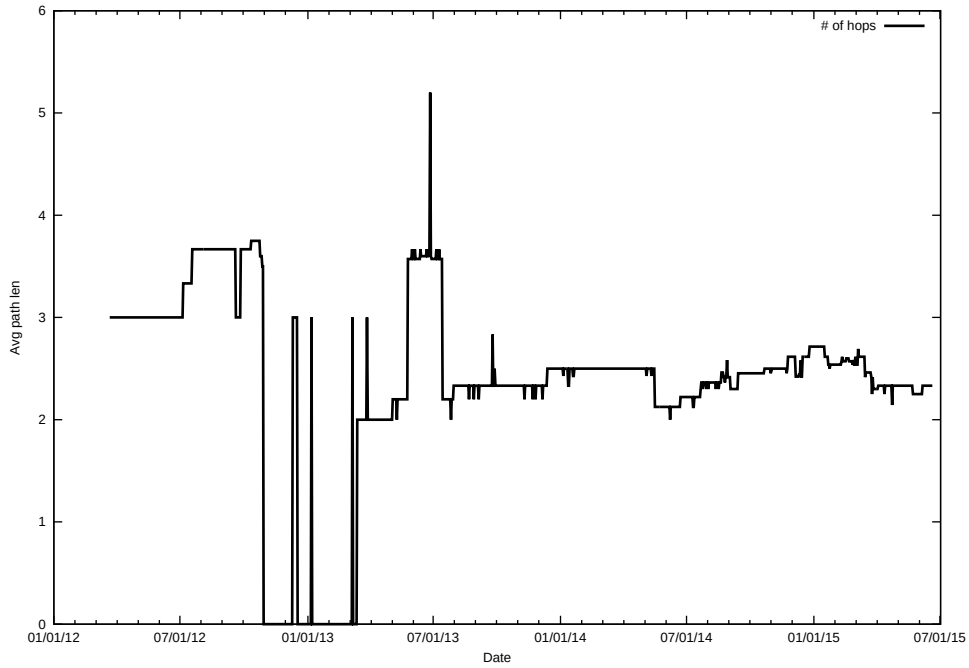
/49


Avg path len
Date
# of hops
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15

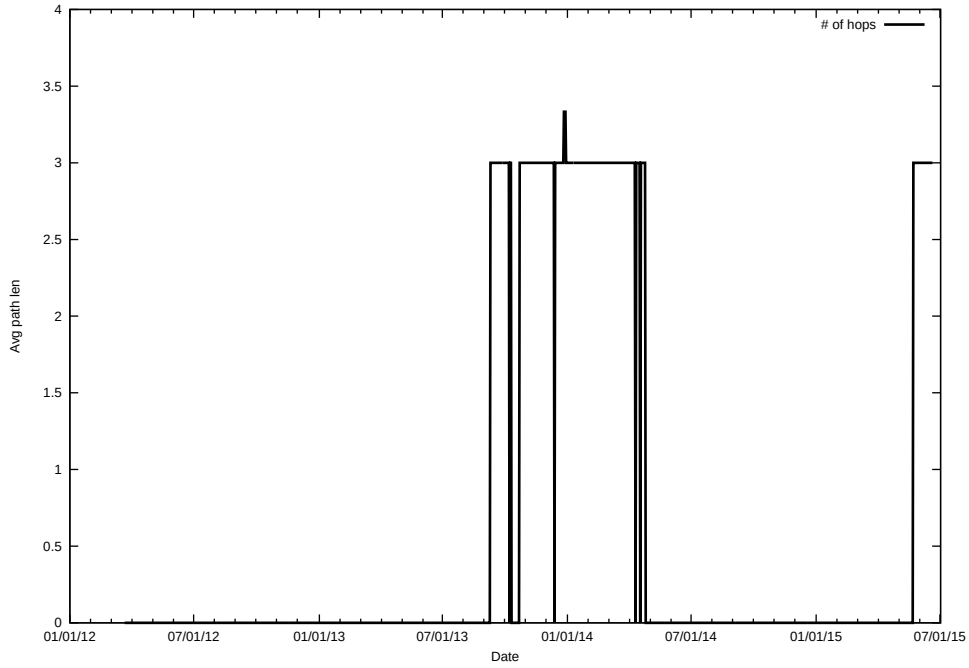
/50


3.5
3
2.5
2
1.5
1
0.5
0
Avg path len
# of hops
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

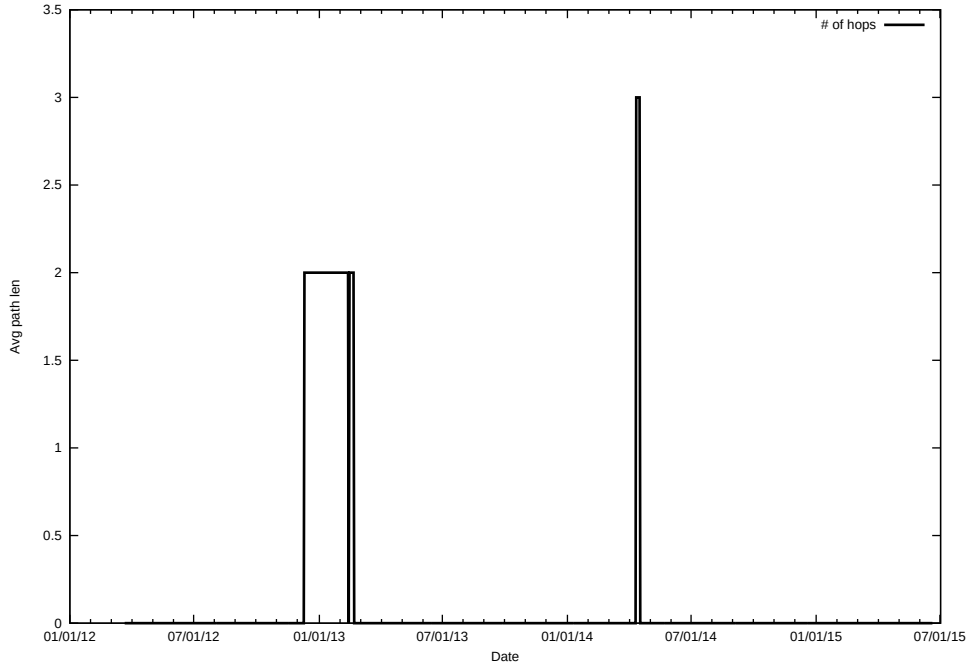
/51


6
5
4
3
2
1
0
Avg path len
# of hops
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

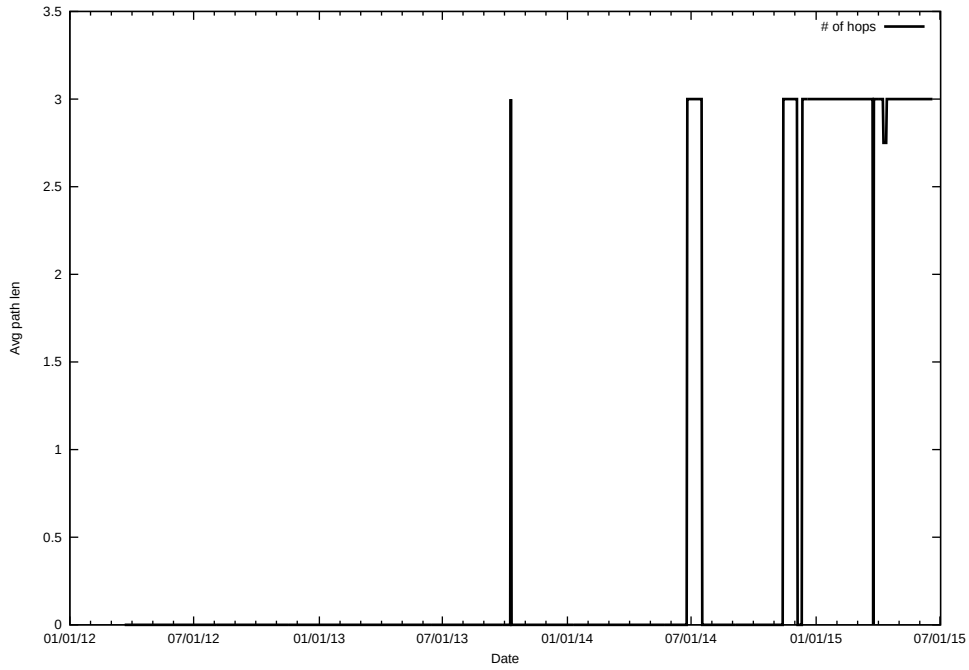
/52


4
3.5
3
2.5
2
1.5
1
0.5
0
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

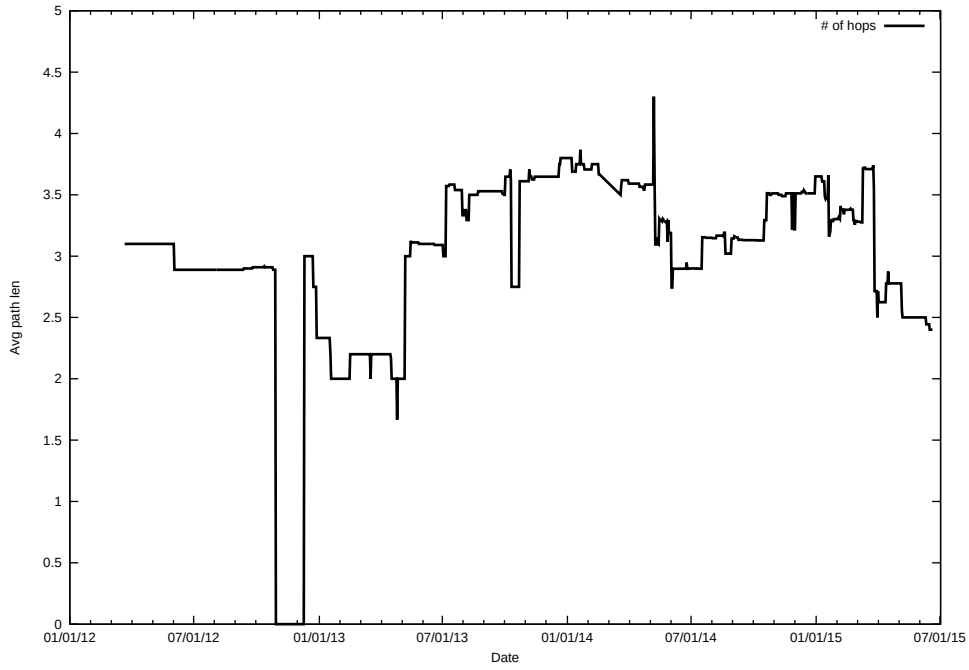
/53


3.5
3
2.5
2
1.5
1
0.5
0
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

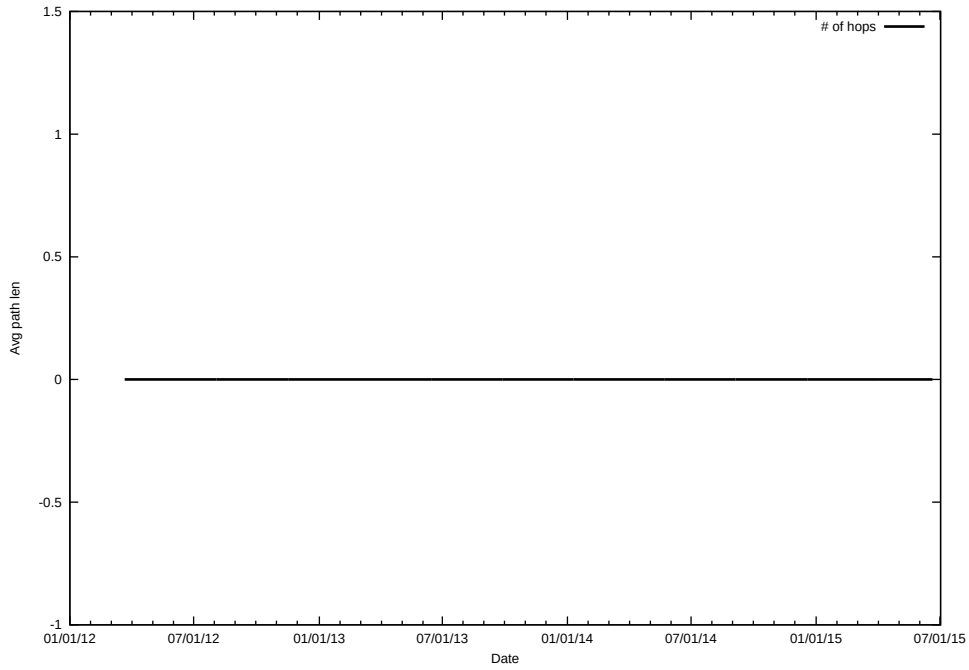
/54


3.5
3
2.5
2
1.5
1
0.5
0
Avg path len
# of hops
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

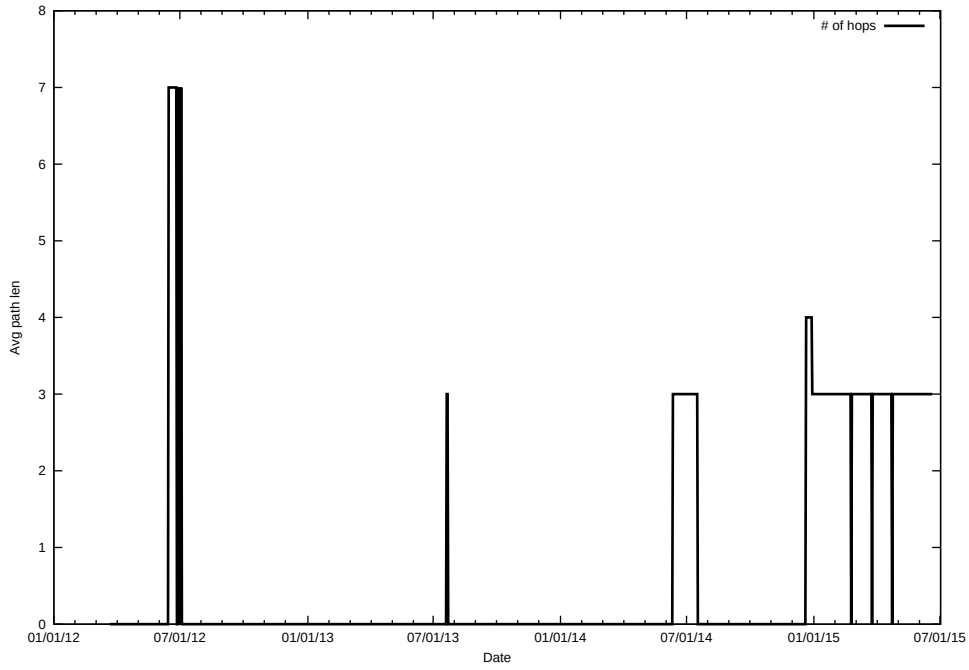
/55


5
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
Avg path len
# of hops
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

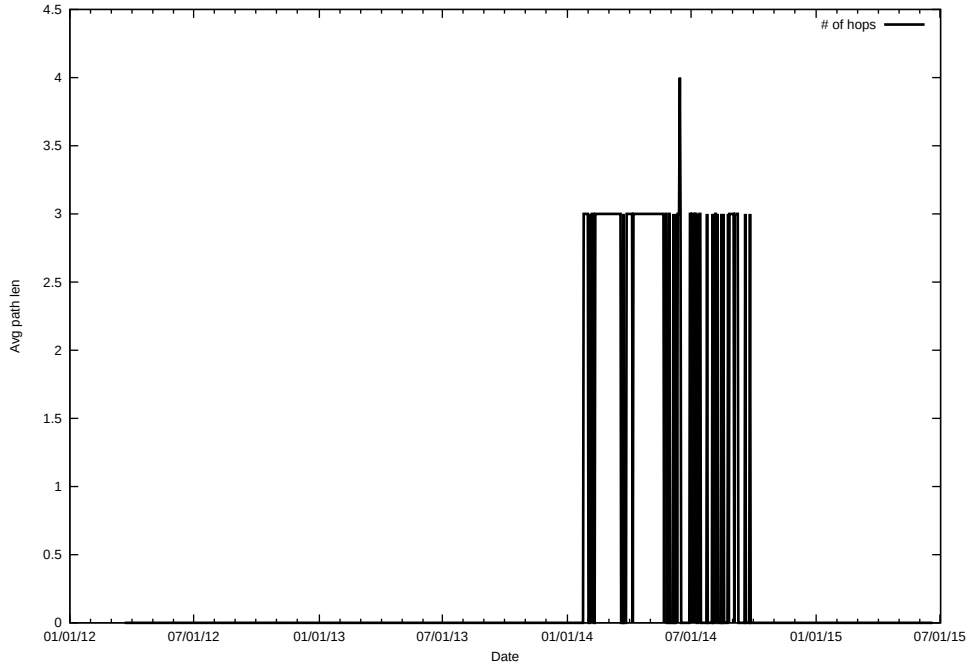
/56


1.5
1
0.5
0
-0.5
-1
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

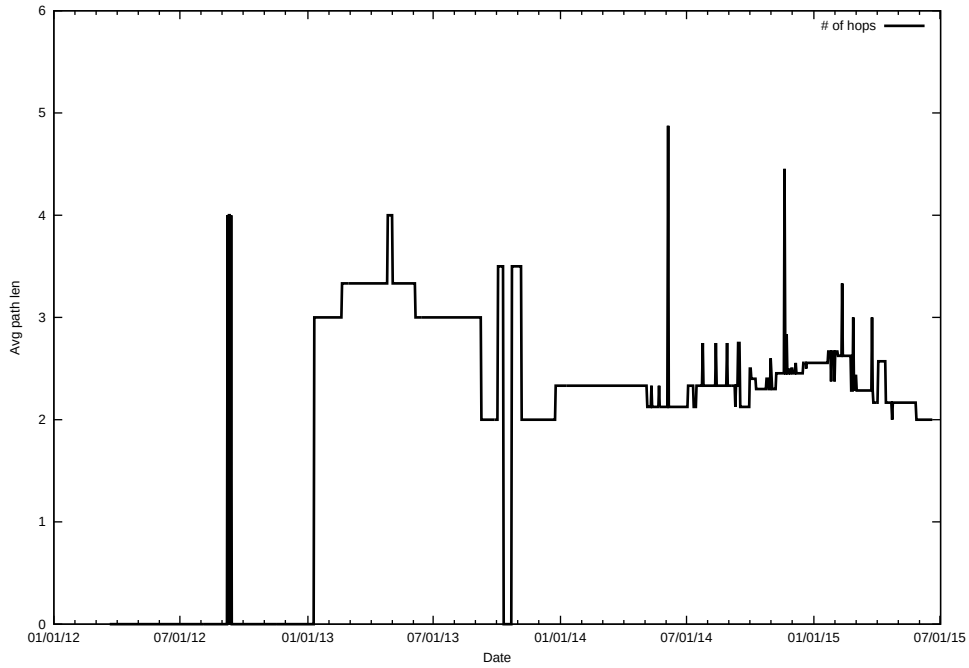
/57


8
7
6
5
4
3
2
1
0
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

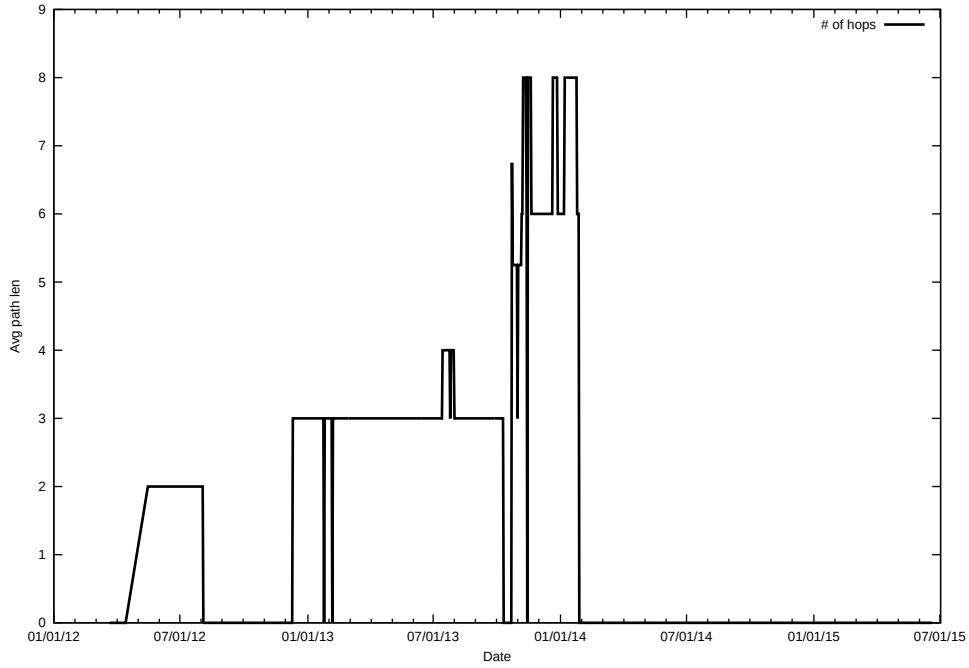
/58


4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

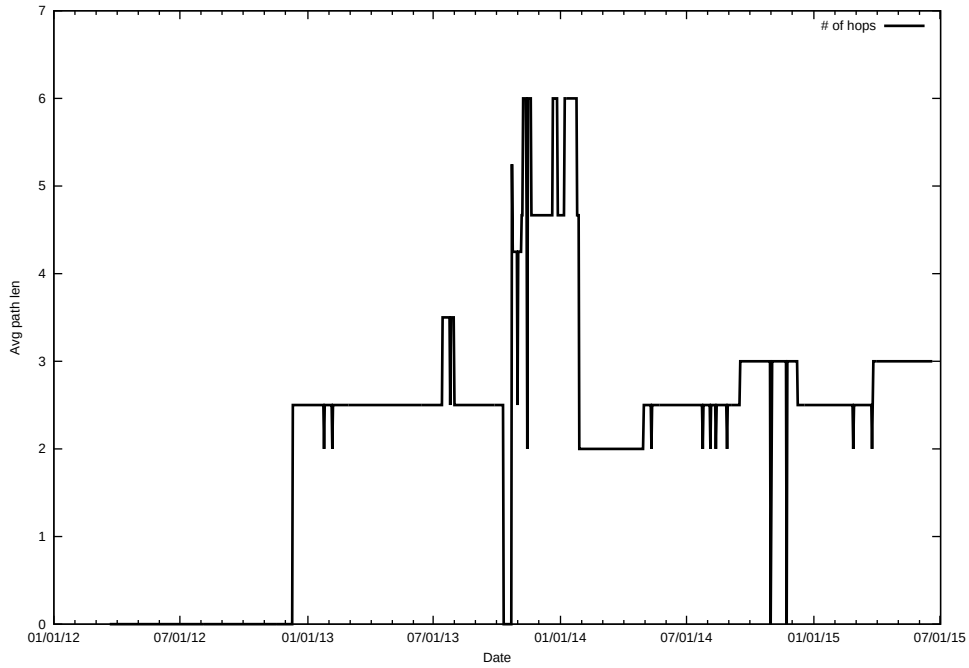
/59


6
5
4
3
2
1
0
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

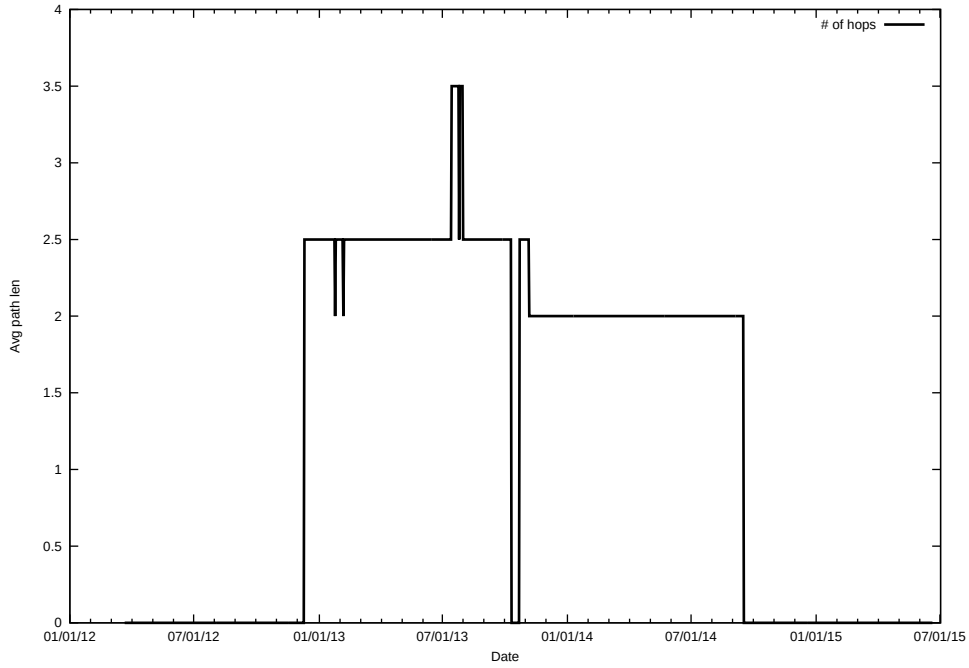
/60


Avg path len
Date
# of hops
0
1
2
3
4
5
6
7
8
9
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15

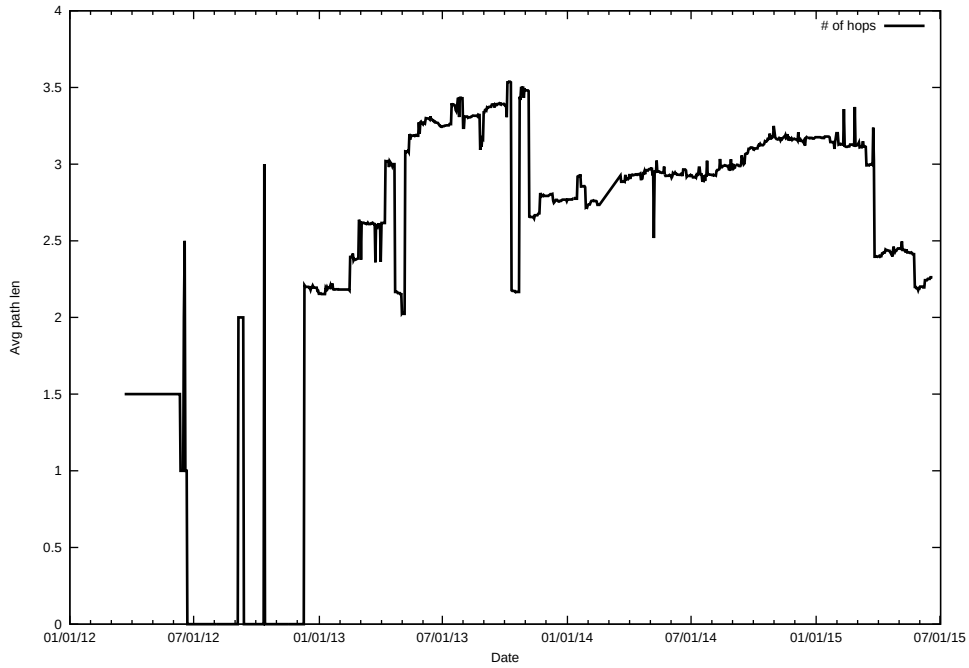
/61


Avg path len
Date
# of hops
0
1
2
3
4
5
6
7
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15

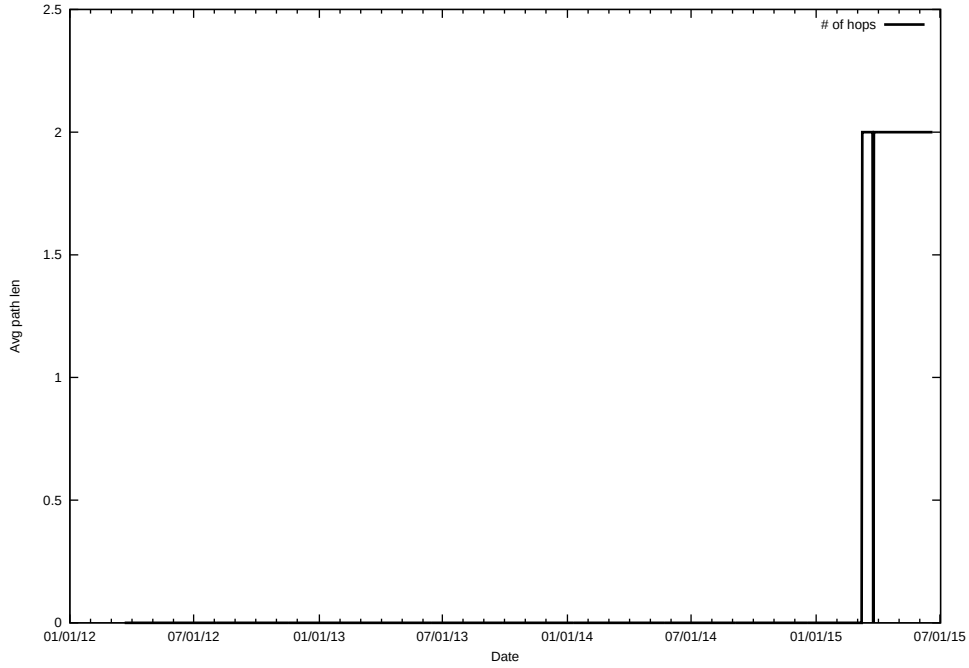
/62


# of hops
Avg path len
Date
0
0.5
1
1.5
2
2.5
3
3.5
4
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15

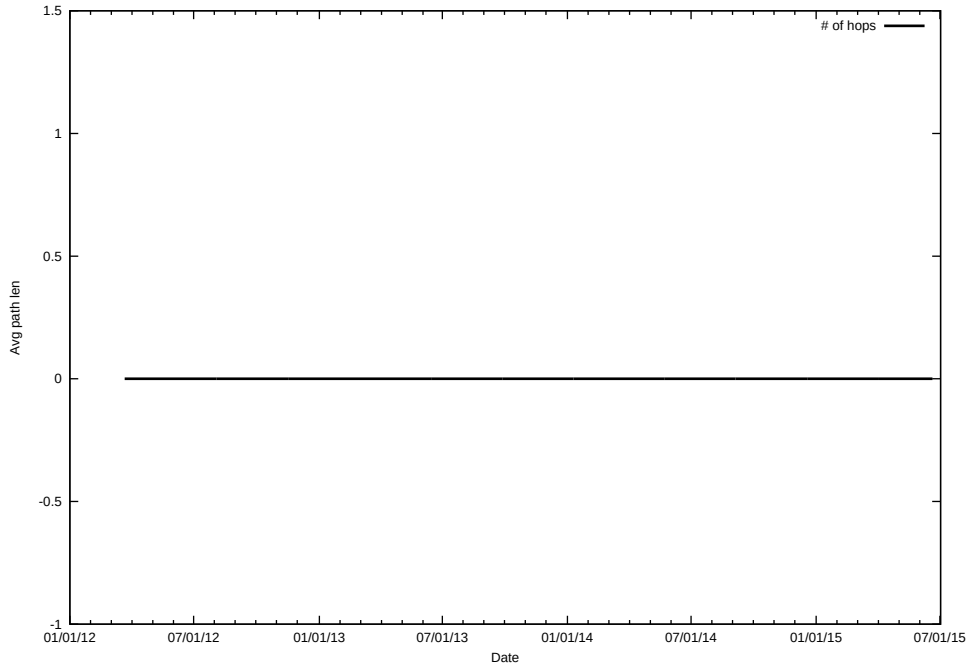
/63


# of hops
Avg path len
Date
0
0.5
1
1.5
2
2.5
3
3.5
4
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15

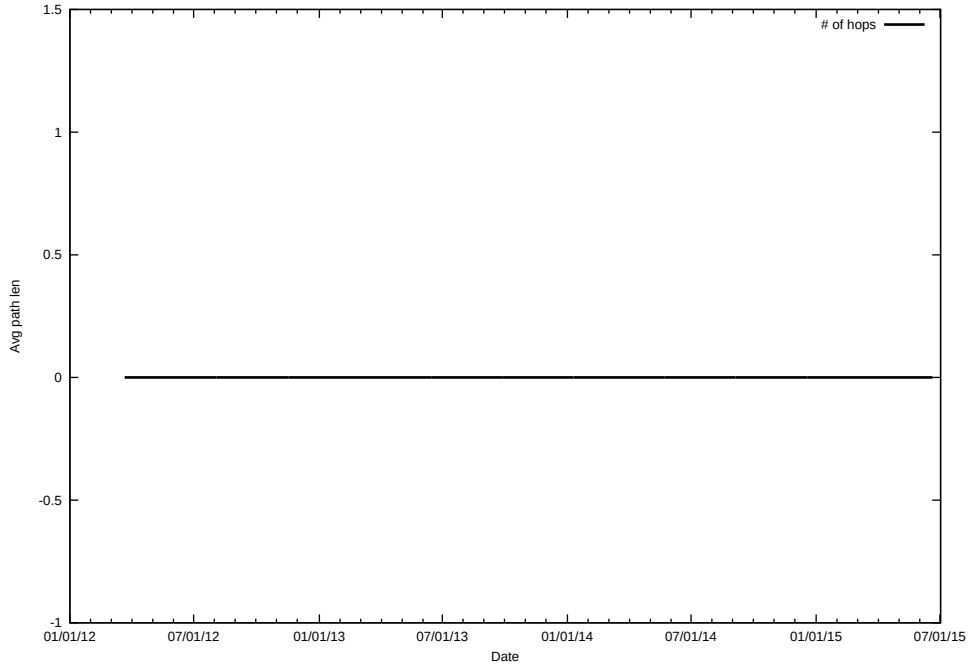
/64


2.5
2
1.5
1
0.5
0
Avg path len
# of hops
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

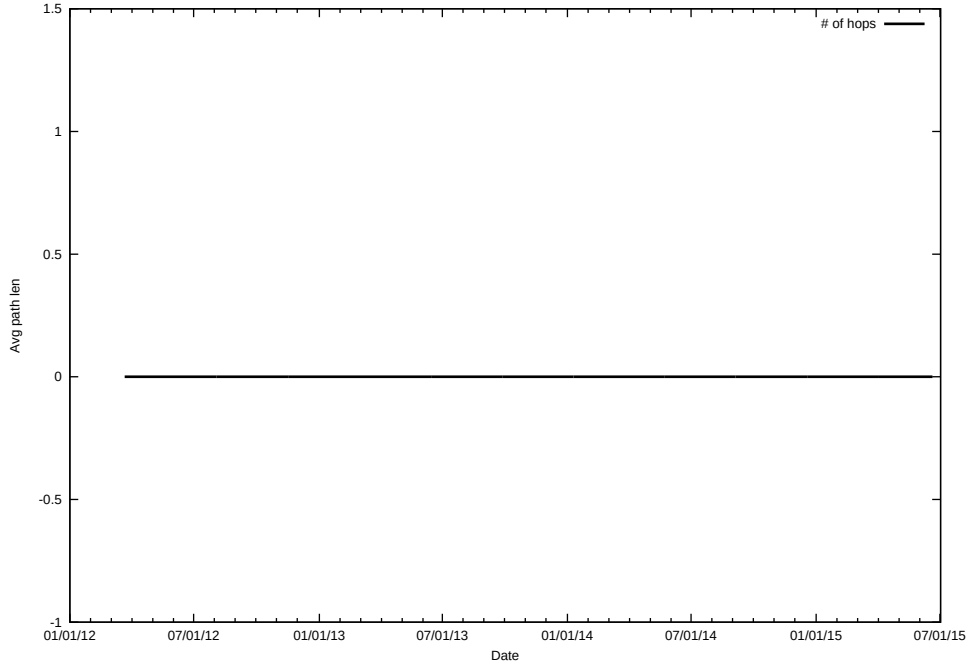
/65


1.5
1
0.5
0
-0.5
-1
Avg path len
# of hops
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

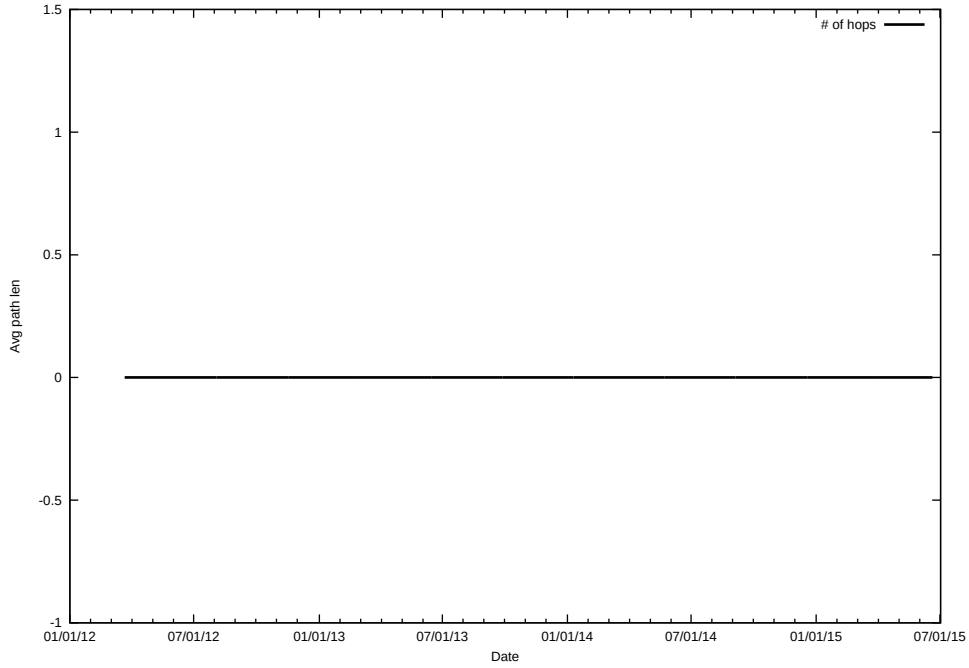
/66


1.5
1
0.5
0
-0.5
-1
Avg path len
# of hops
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

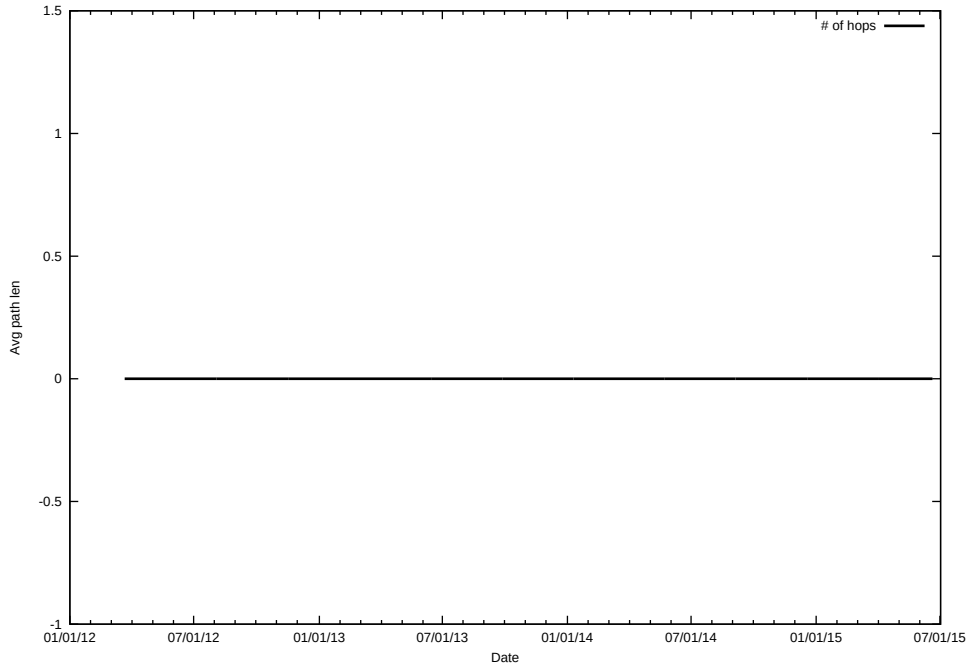
/67


1.5
1
0.5
0
-0.5
-1
Avg path len
# of hops
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

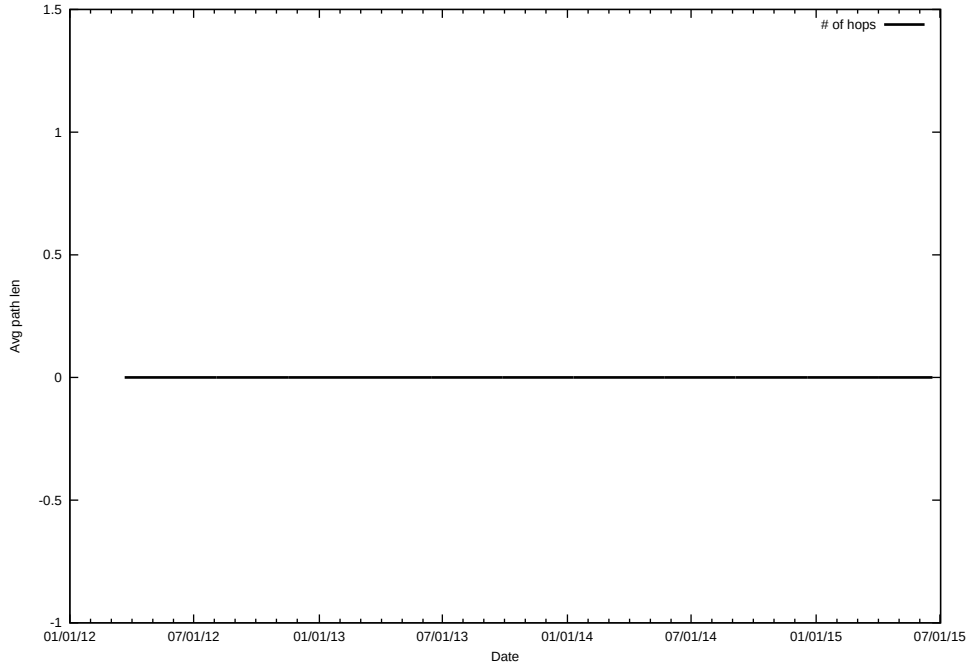
/68


1.5
1
0.5
0
-0.5
-1
Avg path len
# of hops
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

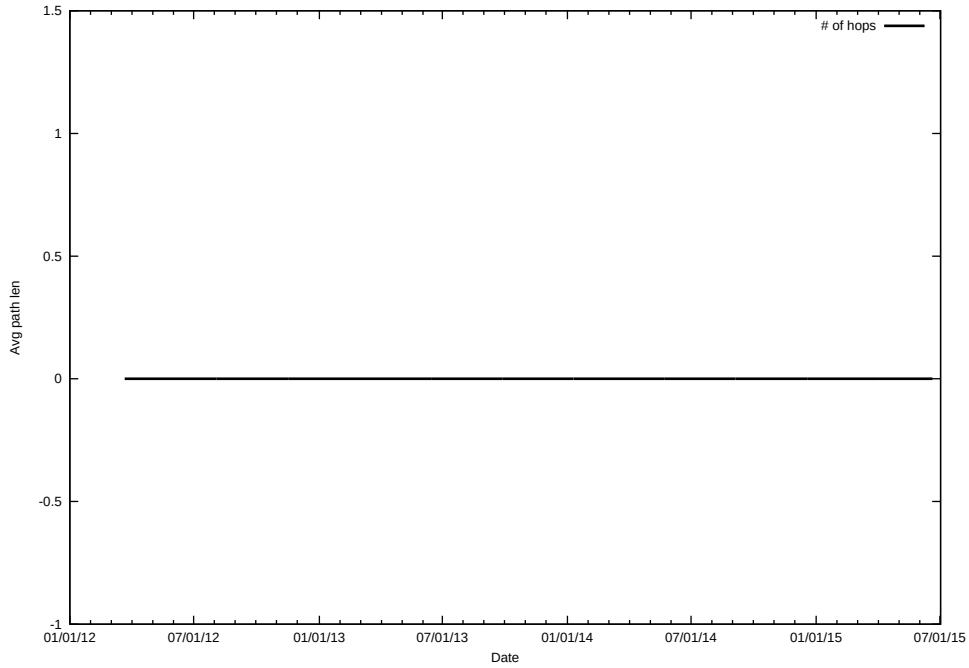
/69


1.5
1
0.5
0
-0.5
-1
Avg path len
# of hops
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

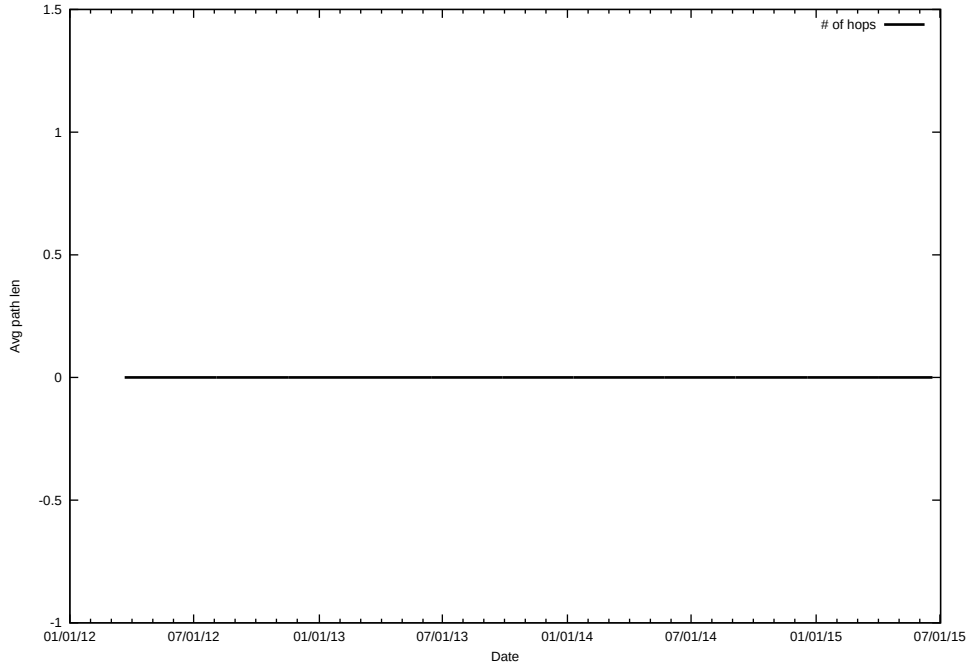
/70


1.5
1
0.5
0
-0.5
-1
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

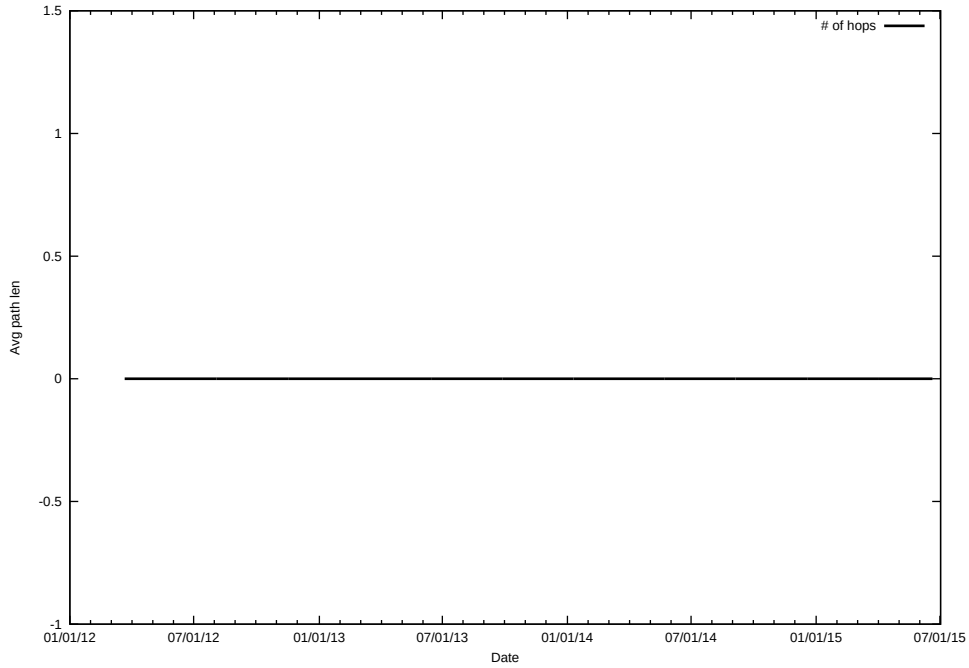
/71


1.5
1
0.5
0
-0.5
-1
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

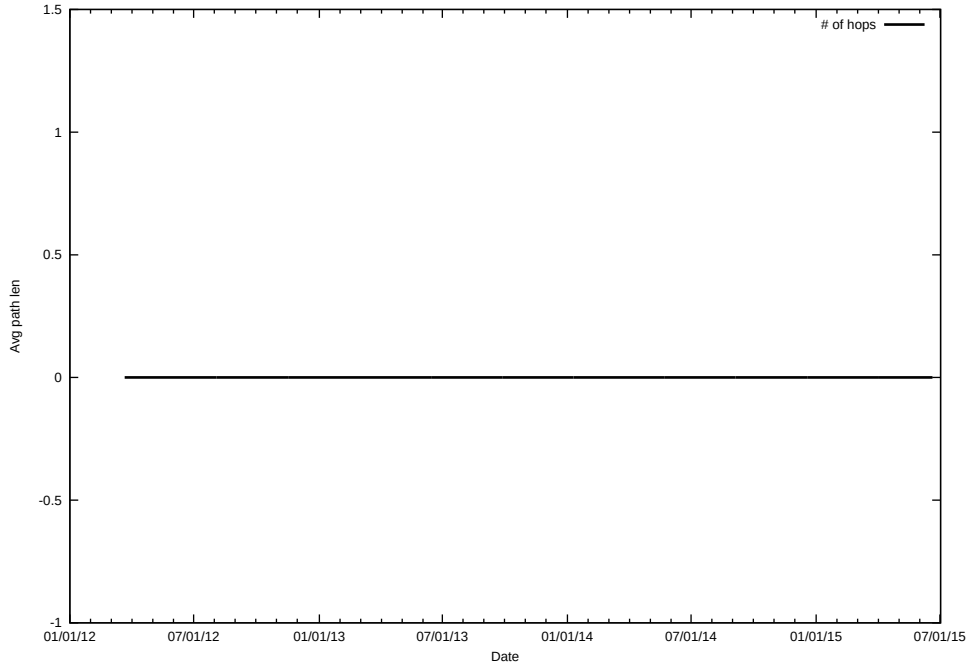
/72


1.5
1
0.5
0
-0.5
-1
Avg path len
# of hops
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

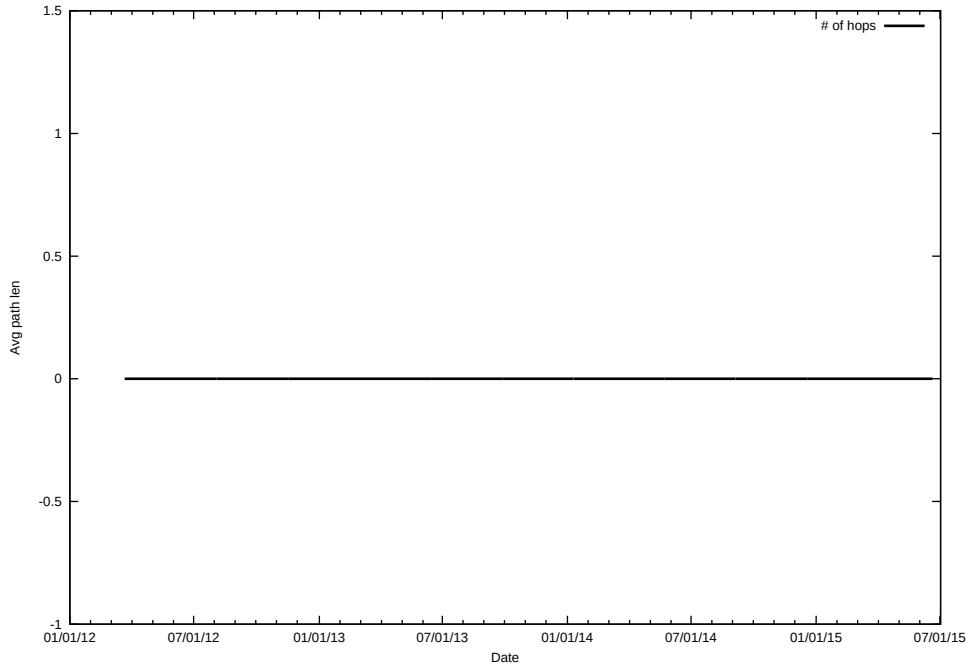
/73


1.5
1
0.5
0
-0.5
-1
Avg path len
# of hops
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

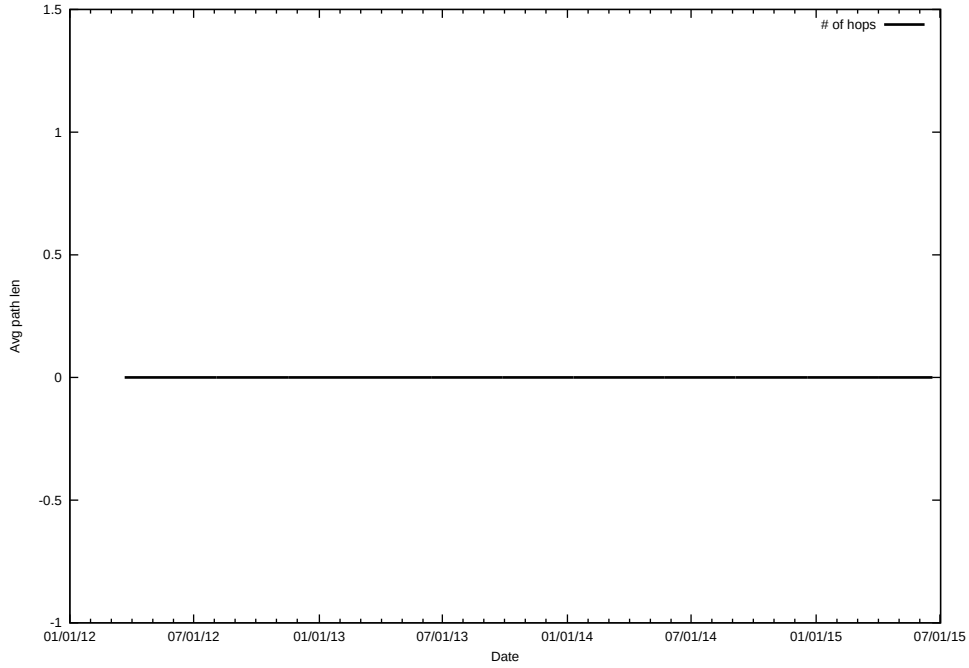
/74


1.5
1
0.5
0
-0.5
-1
Avg path len
# of hops
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

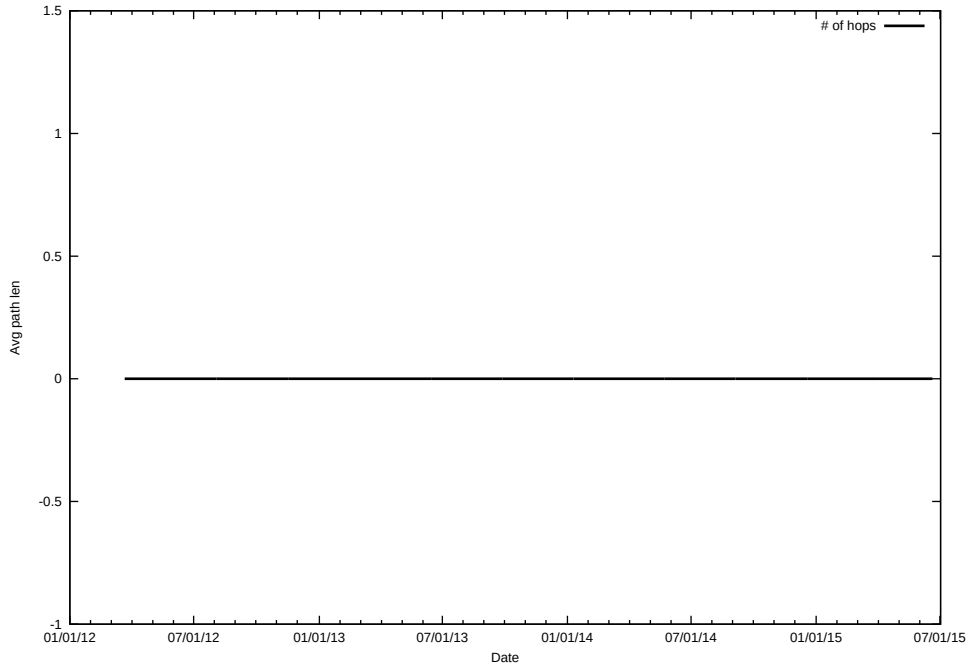
/75


1.5
1
0.5
0
-0.5
-1
Avg path len
# of hops
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

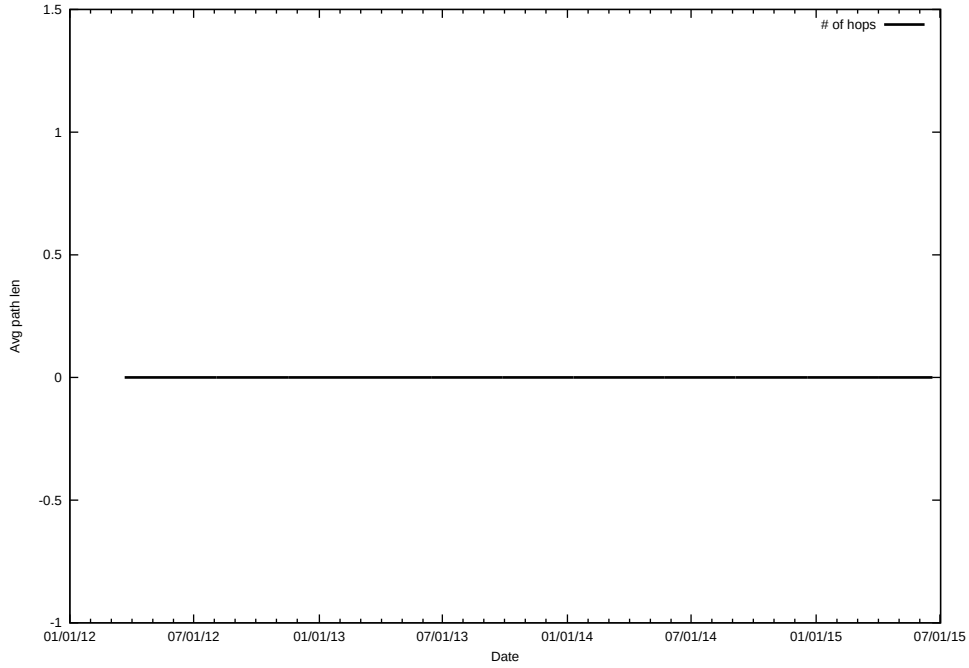
/76


1.5
1
0.5
0
-0.5
-1
Avg path len
# of hops
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

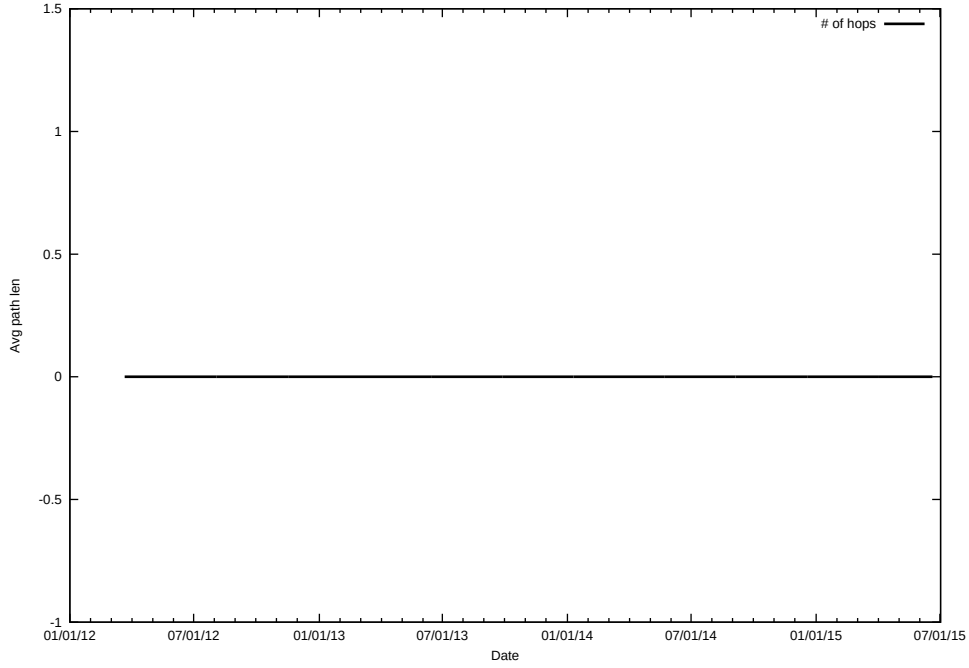
/77


1.5
1
0.5
0
-0.5
-1
Avg path len
# of hops
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

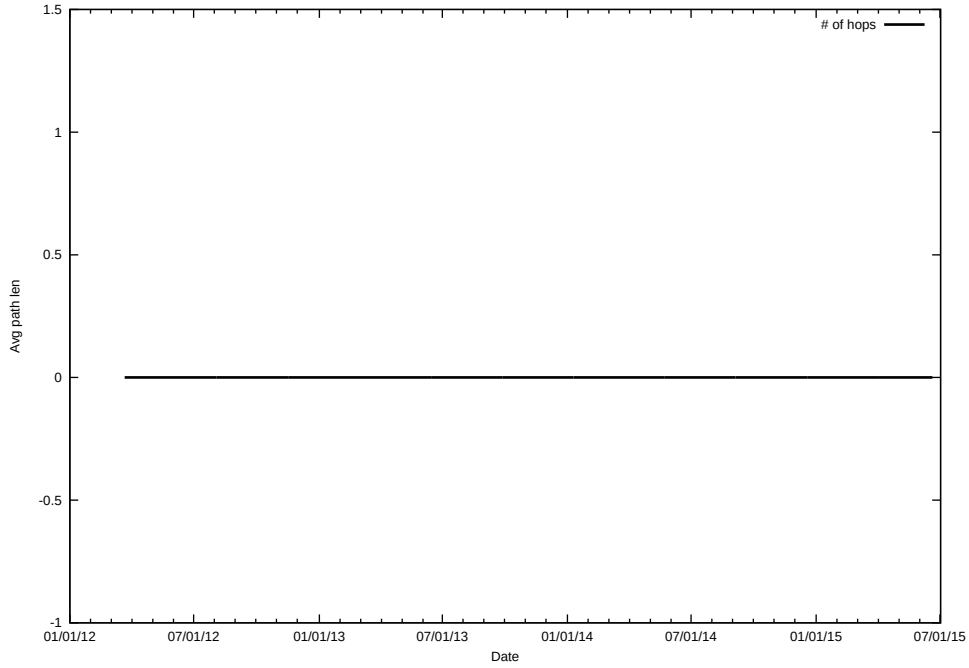
/78


1.5
1
0.5
0
-0.5
-1
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

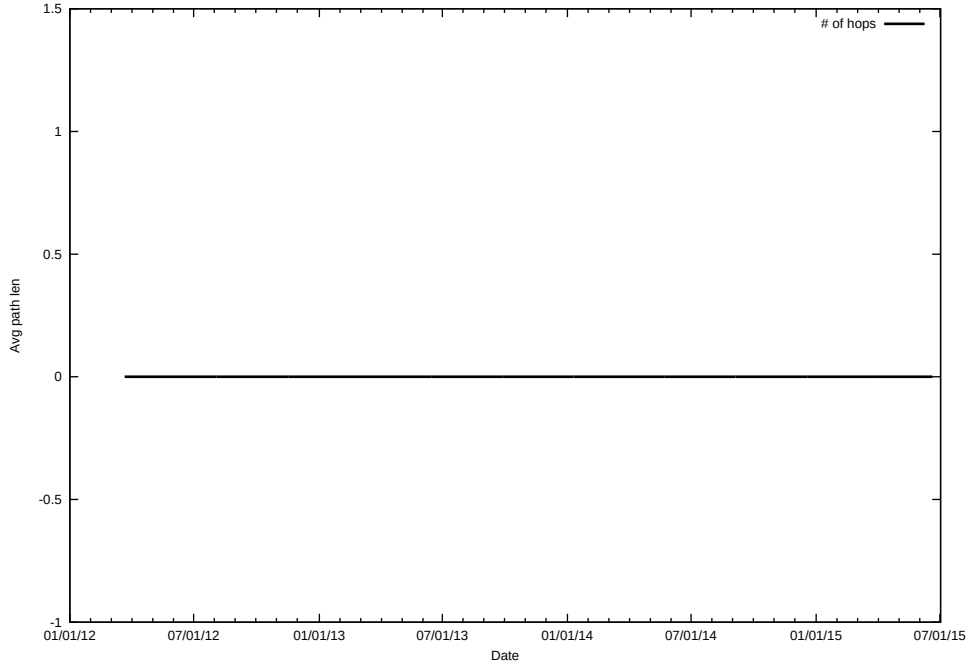
/79


1.5
1
0.5
0
-0.5
-1
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

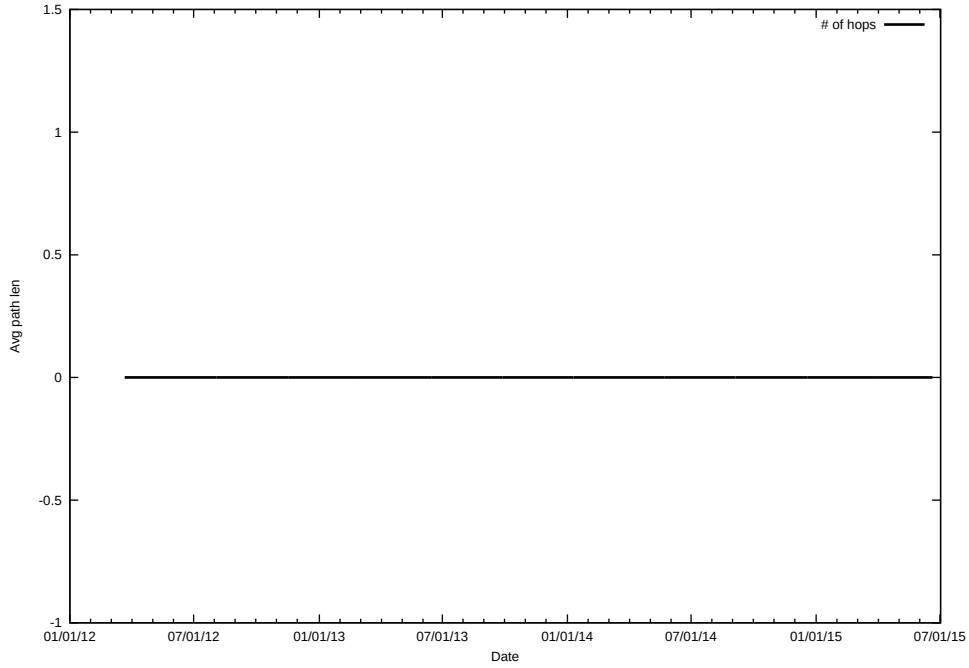
/80


1.5
1
0.5
0
-0.5
-1
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

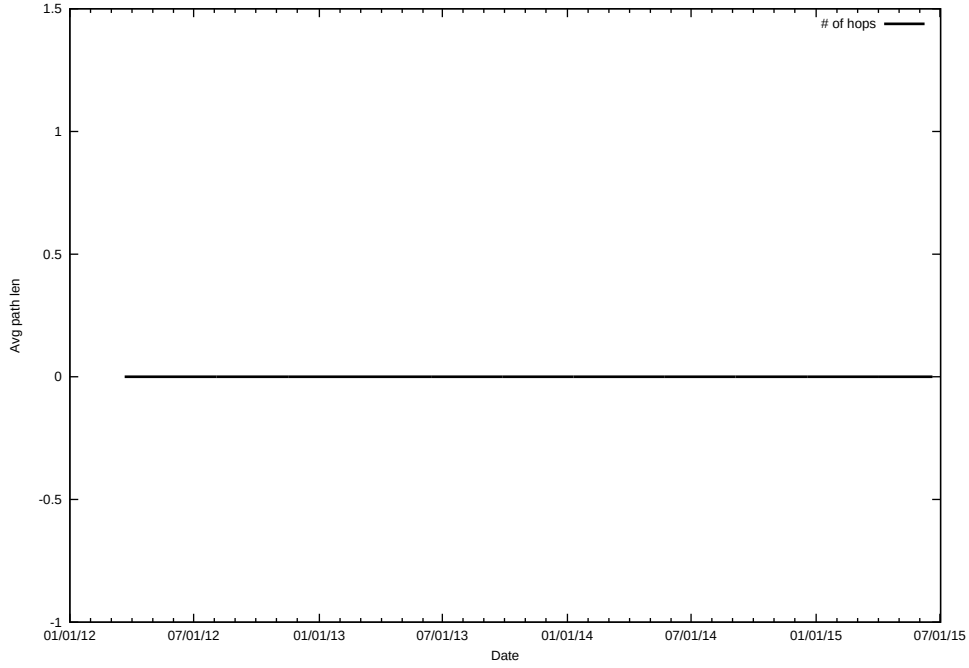
/81


1.5
1
0.5
0
-0.5
-1
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

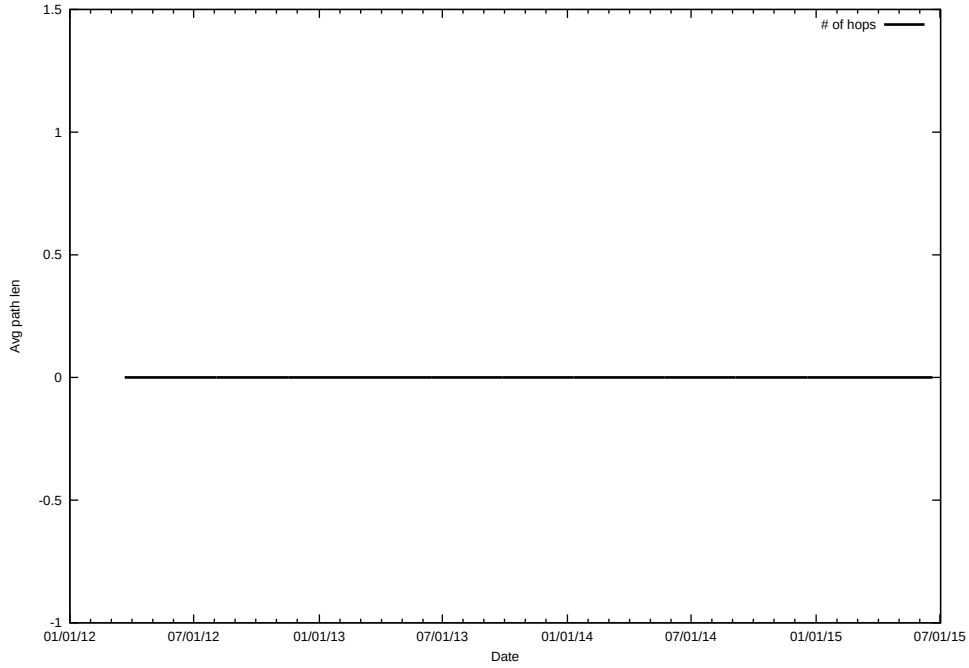
/82


1.5
1
0.5
0
-0.5
-1
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

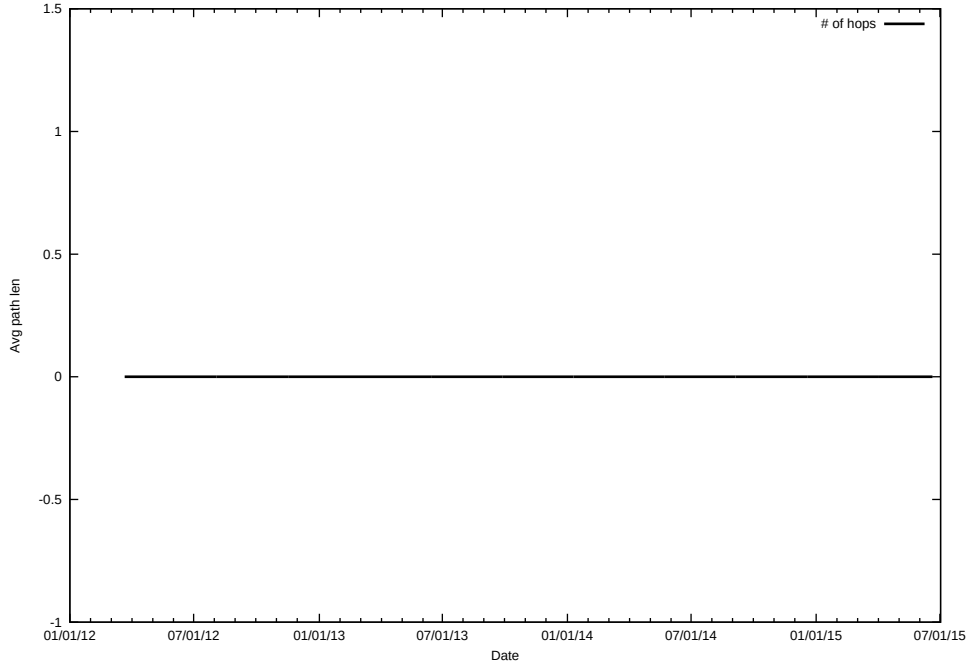
/83


1.5
1
0.5
0
-0.5
-1
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

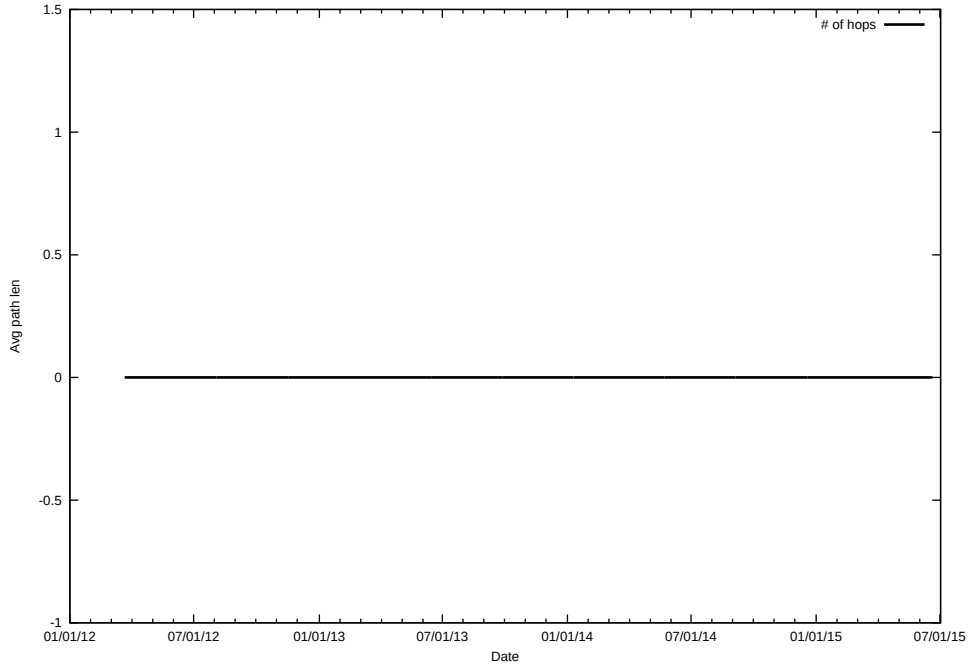
/84


1.5
1
0.5
0
-0.5
-1
Avg path len
# of hops
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

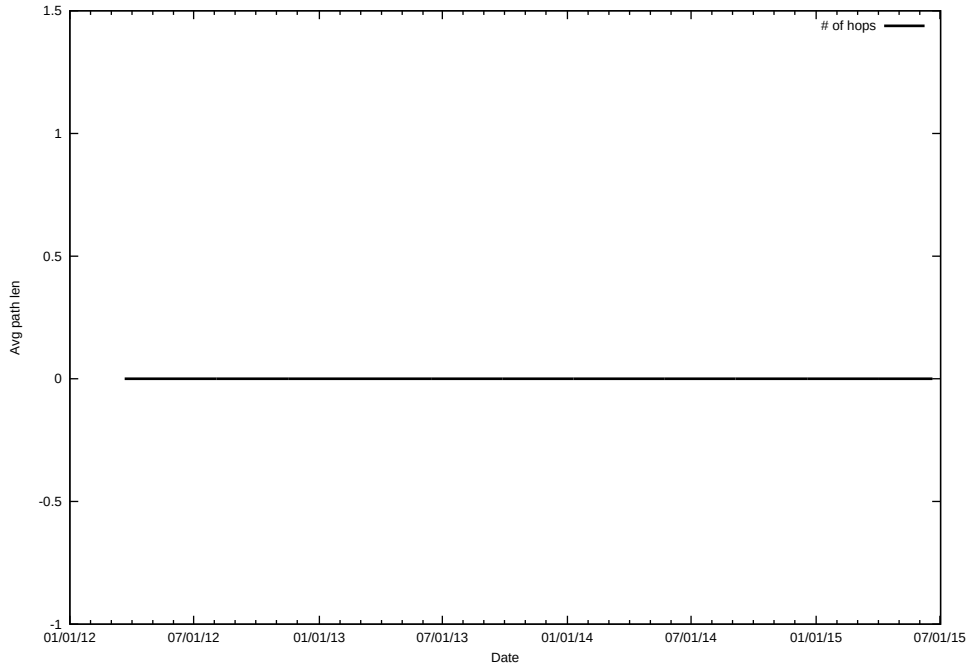
/85


1.5
1
0.5
0
-0.5
-1
Avg path len
# of hops
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

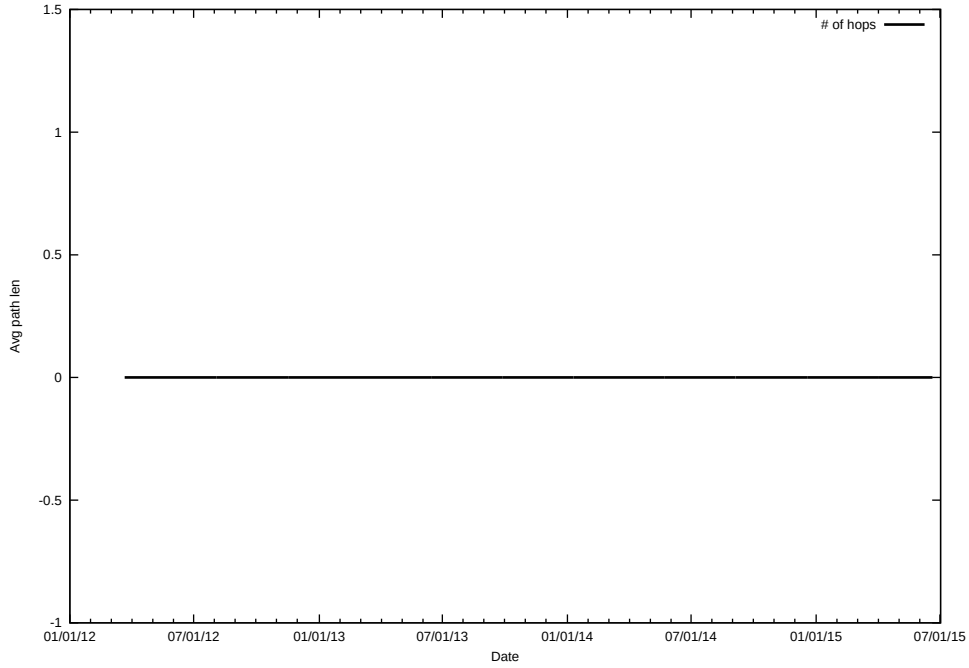
/86


1.5
1
0.5
0
-0.5
-1
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

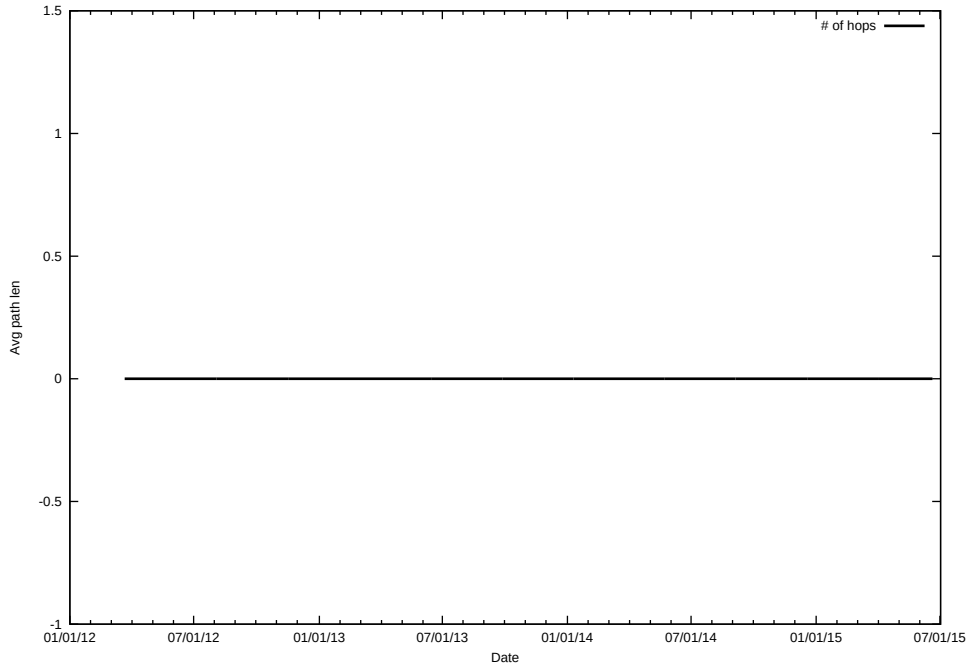
/87


1.5
1
0.5
0
-0.5
-1
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

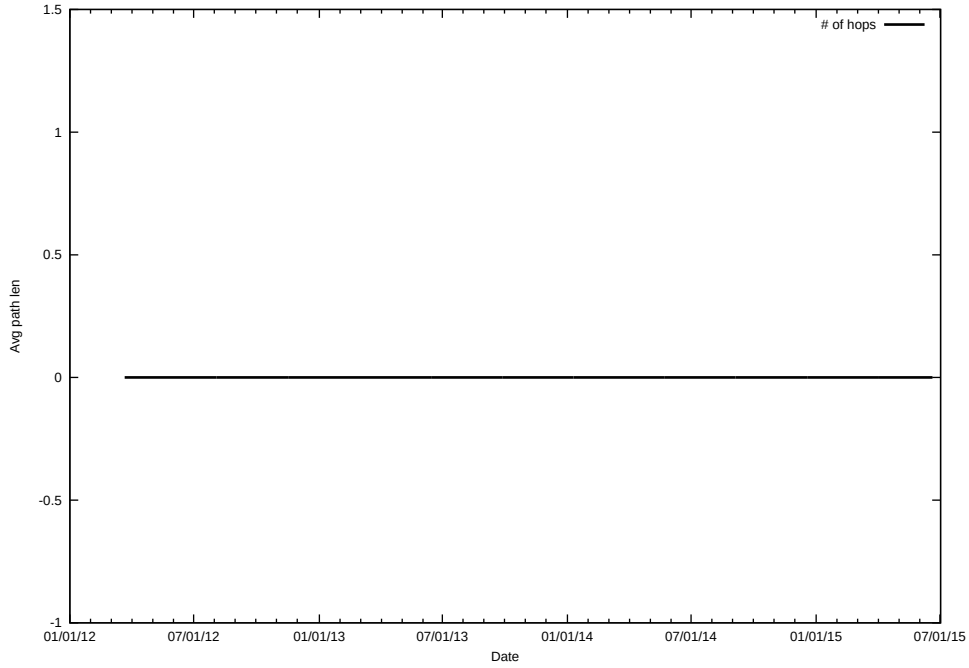
/88


1.5
1
0.5
0
-0.5
-1
Avg path len
# of hops
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

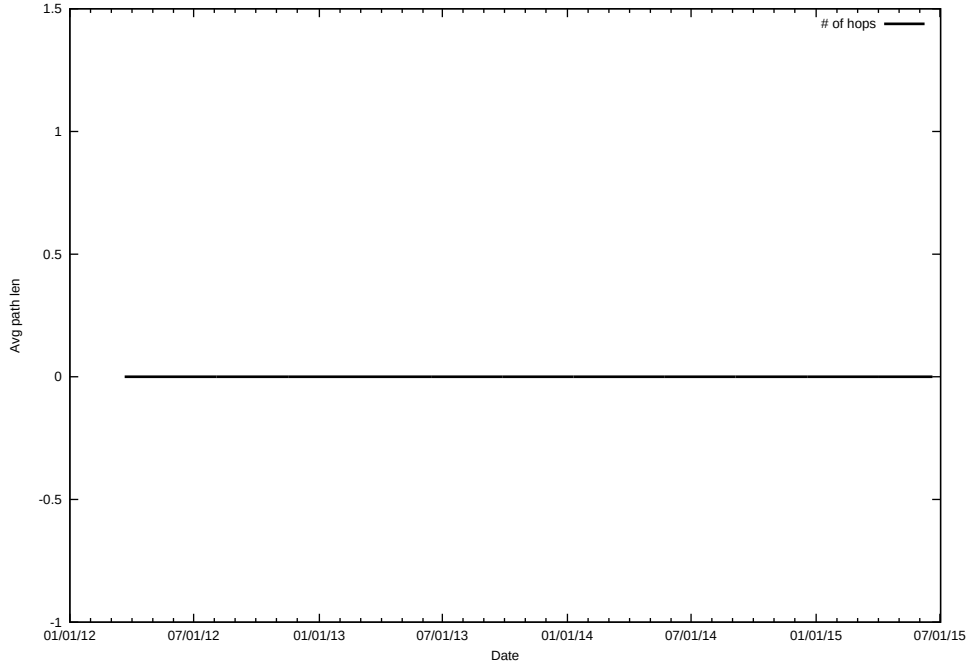
/89


1.5
1
0.5
0
-0.5
-1
Avg path len
# of hops
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

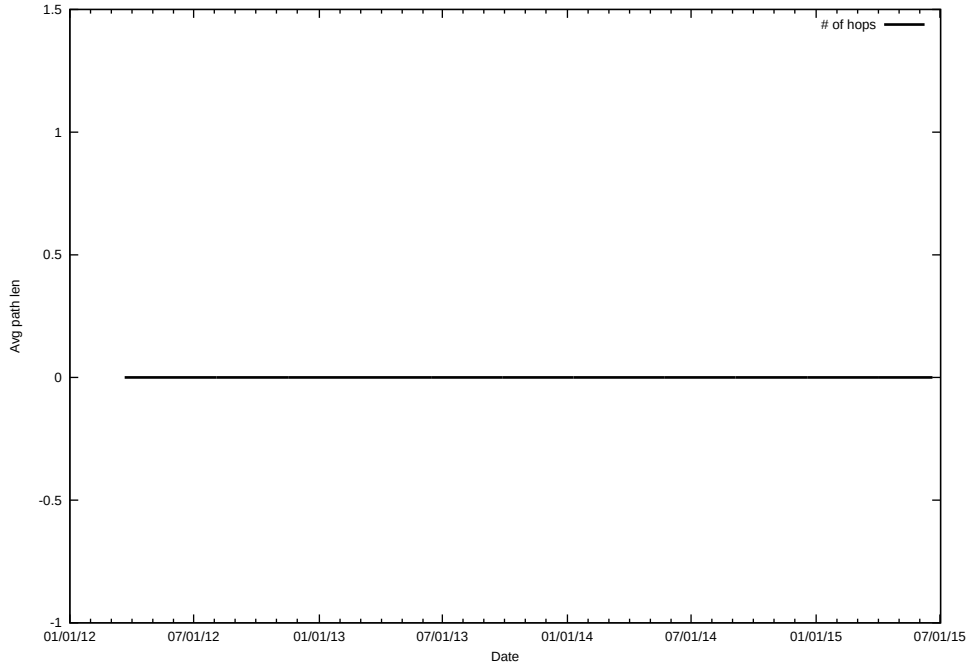
/90


1.5
1
0.5
0
-0.5
-1
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

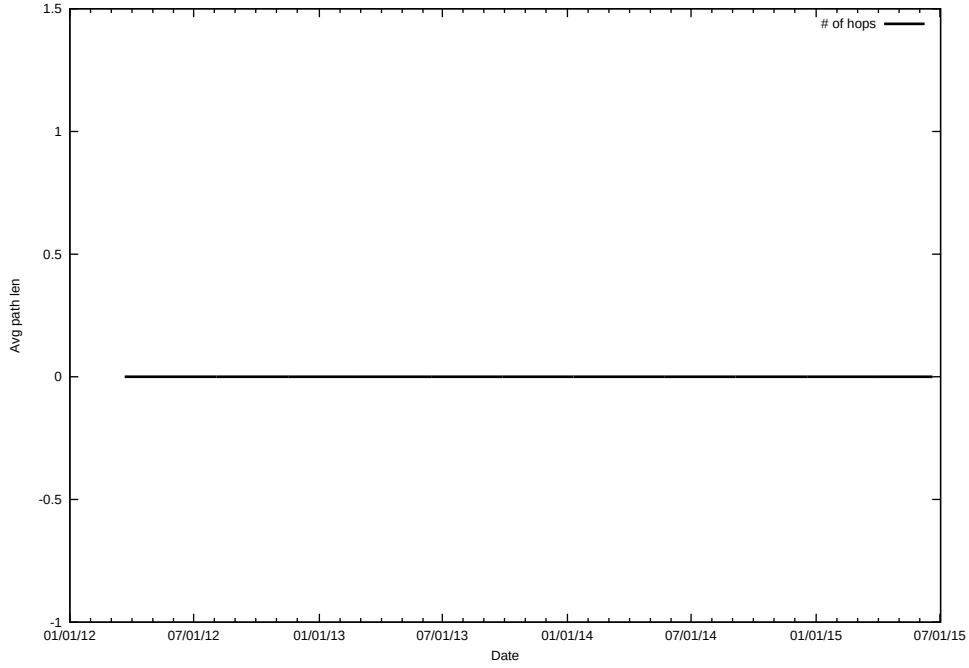
/91


1.5
1
0.5
0
-0.5
-1
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

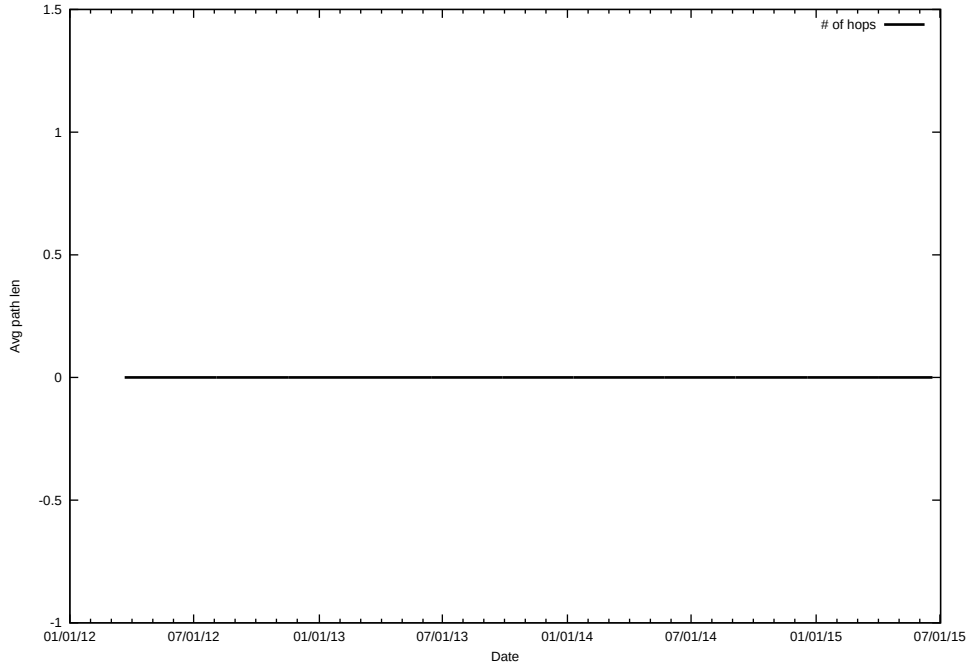
/92


1.5
1
0.5
0
-0.5
-1
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

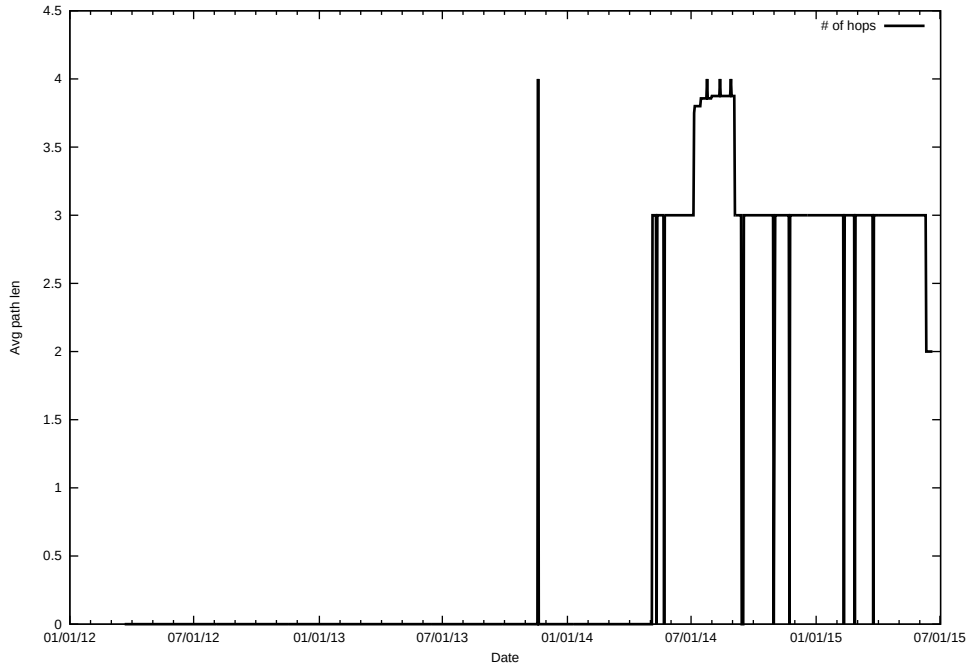
/93


1.5
1
0.5
0
-0.5
-1
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

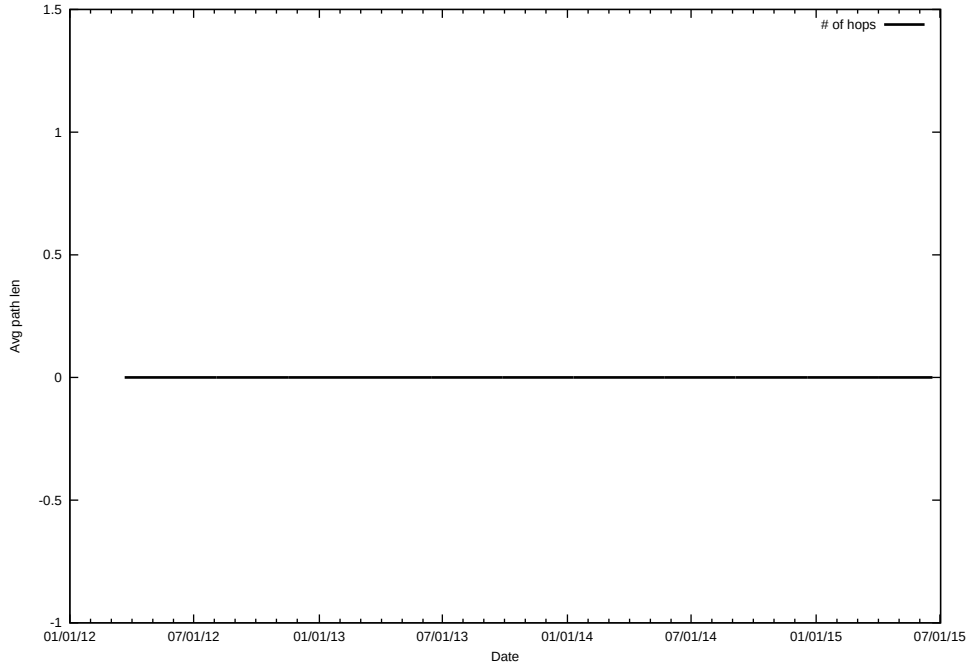
/94


1.5
1
0.5
0
-0.5
-1
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

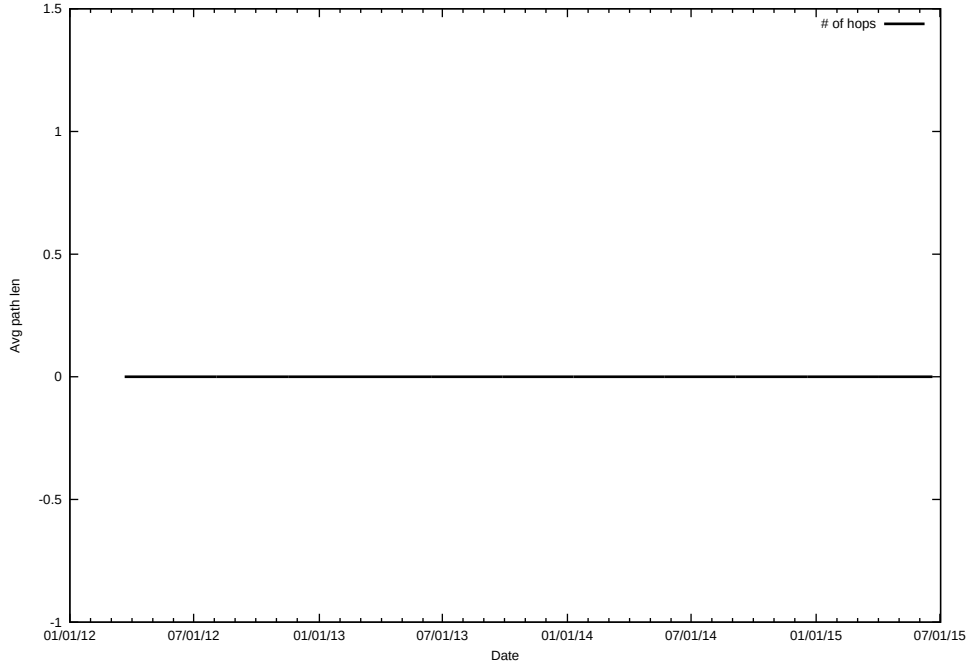
/95


4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

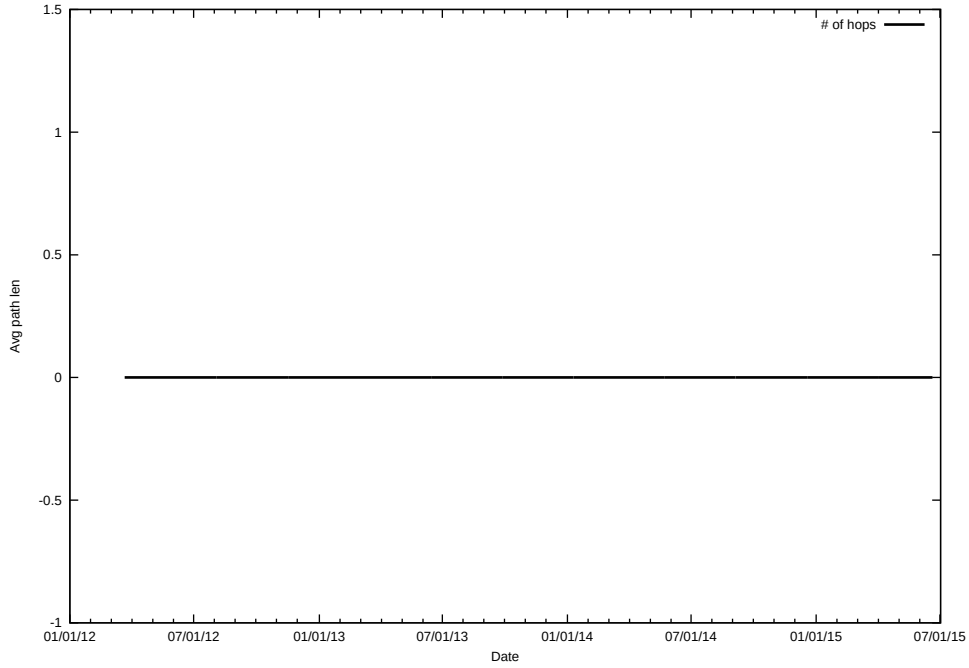
/96


1.5
1
0.5
0
-0.5
-1
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

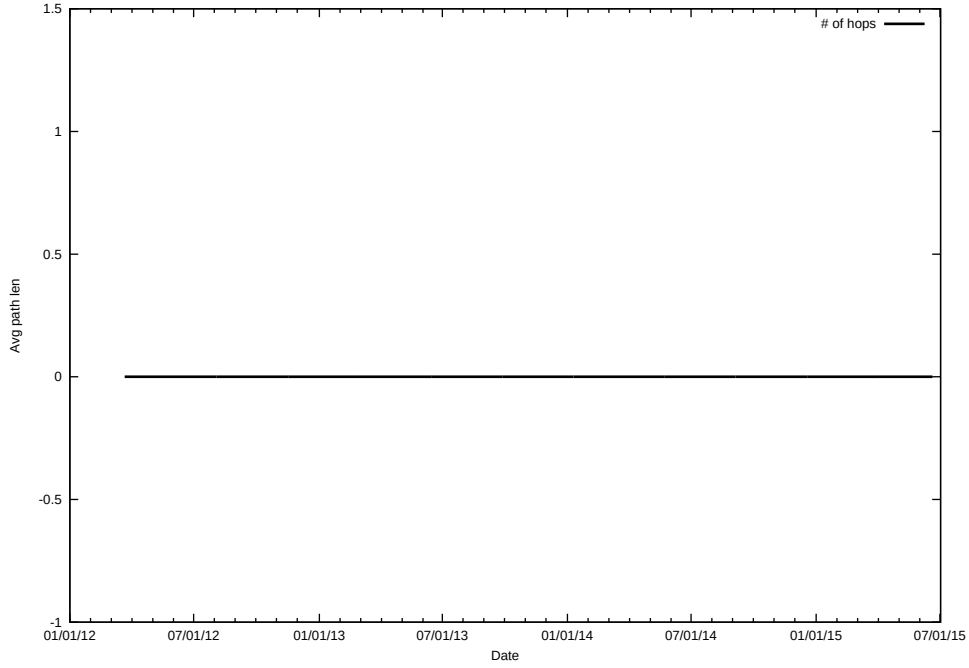
/97


1.5
1
0.5
0
-0.5
-1
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

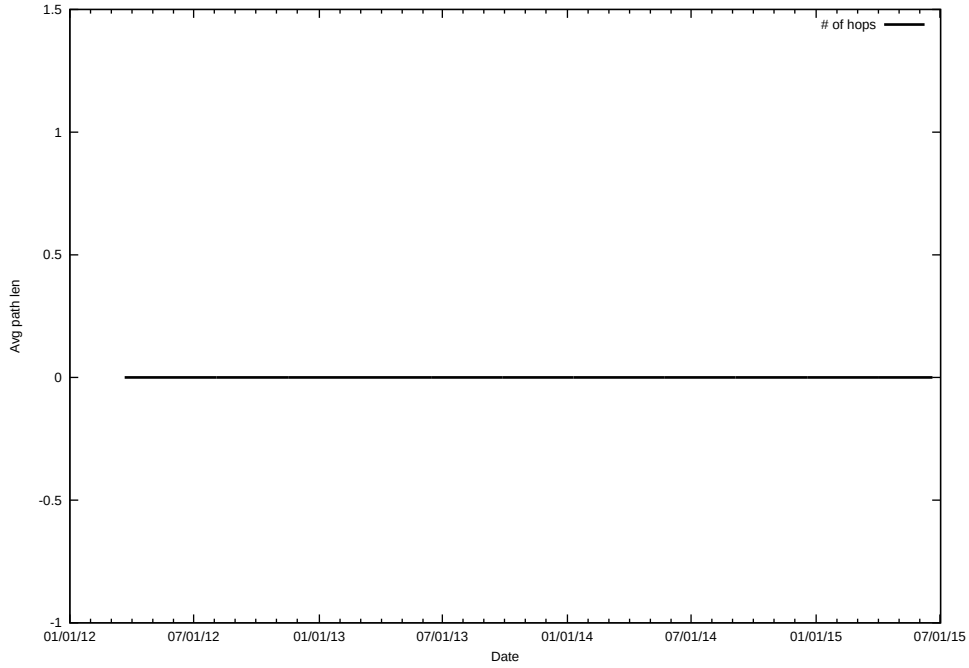
/98


1.5
1
0.5
0
-0.5
-1
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

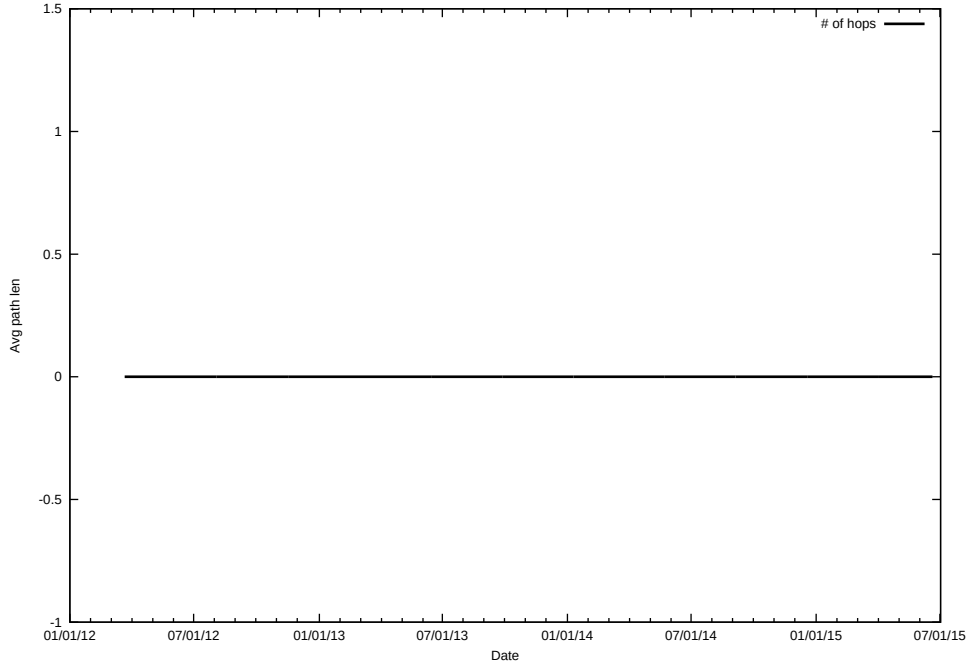
/99


1.5
1
0.5
0
-0.5
-1
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

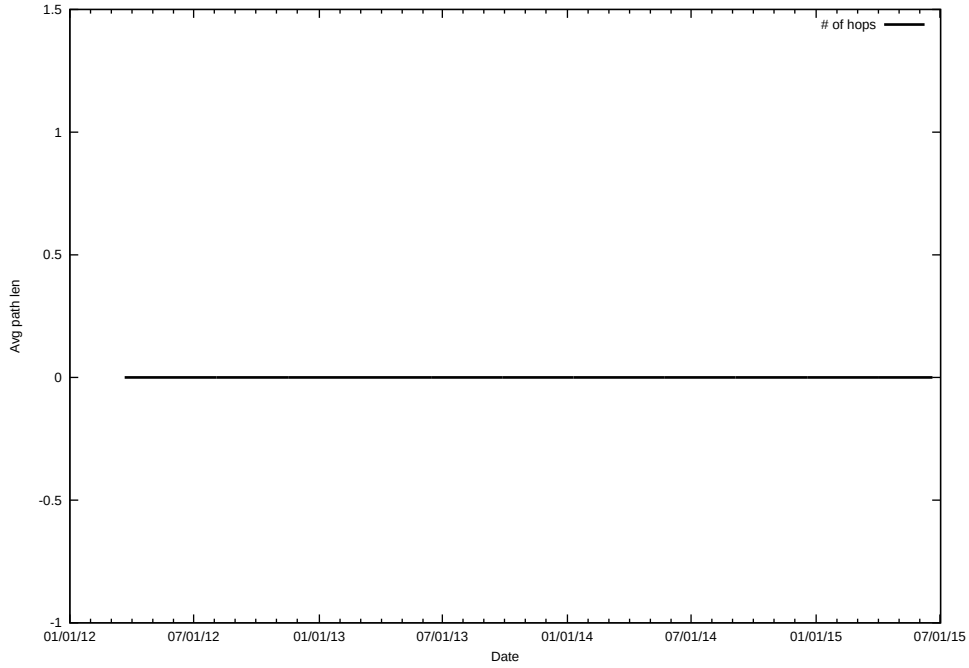
/100


1.5
1
0.5
0
-0.5
-1
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

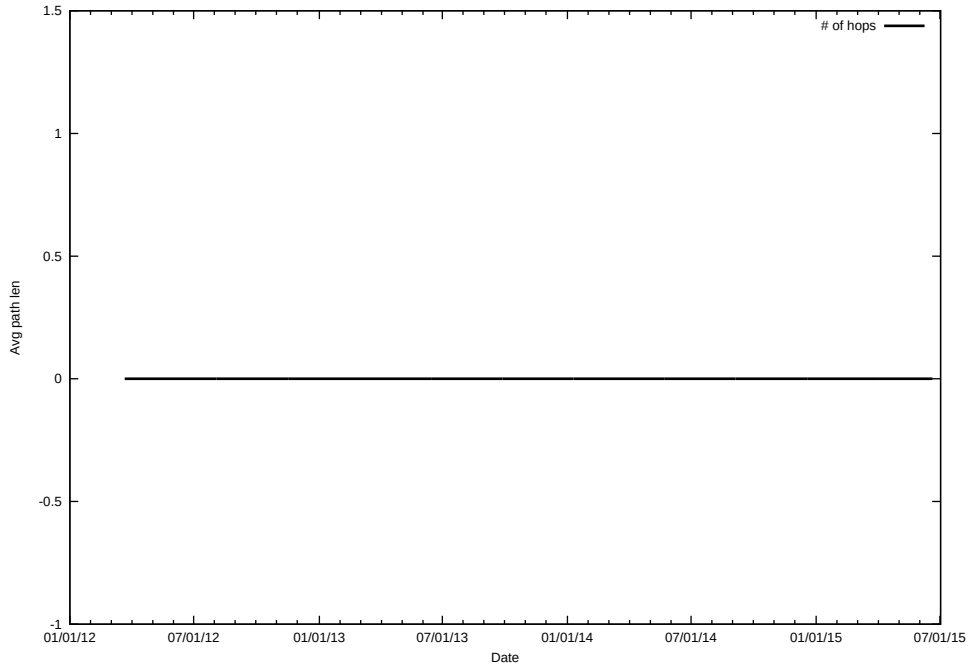
/101


1.5
1
0.5
0
-0.5
-1
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

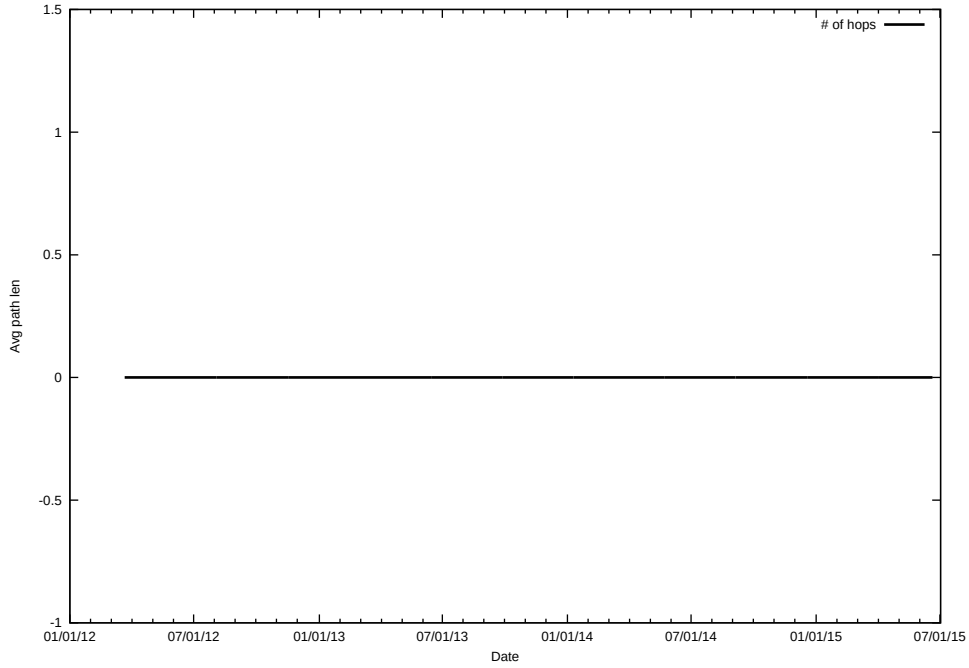
/102


1.5
1
0.5
0
-0.5
-1
Avg path len
# of hops
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

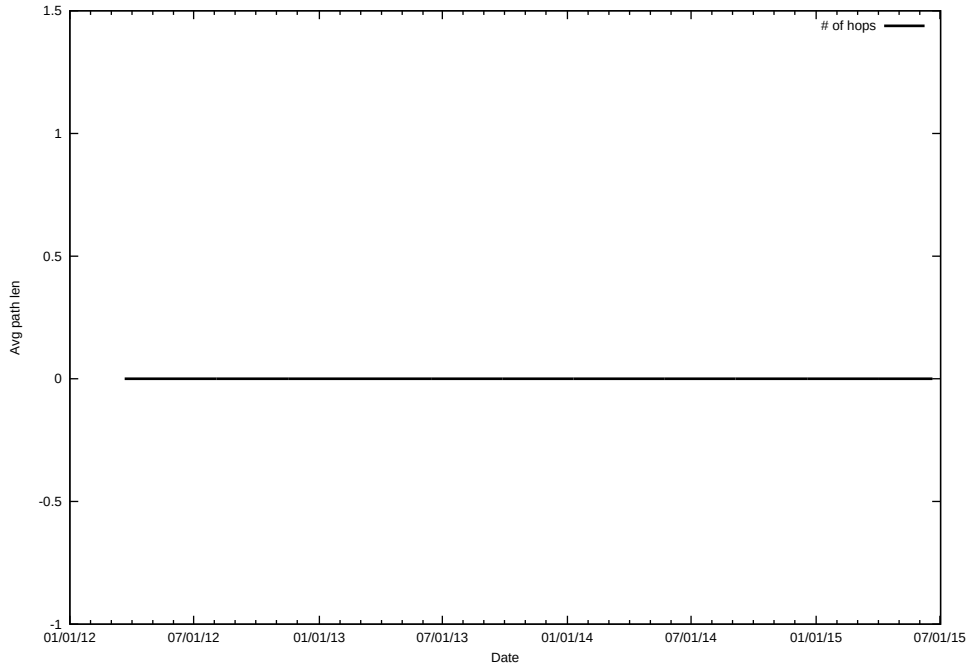
/103


1.5
1
0.5
0
-0.5
-1
Avg path len
# of hops
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

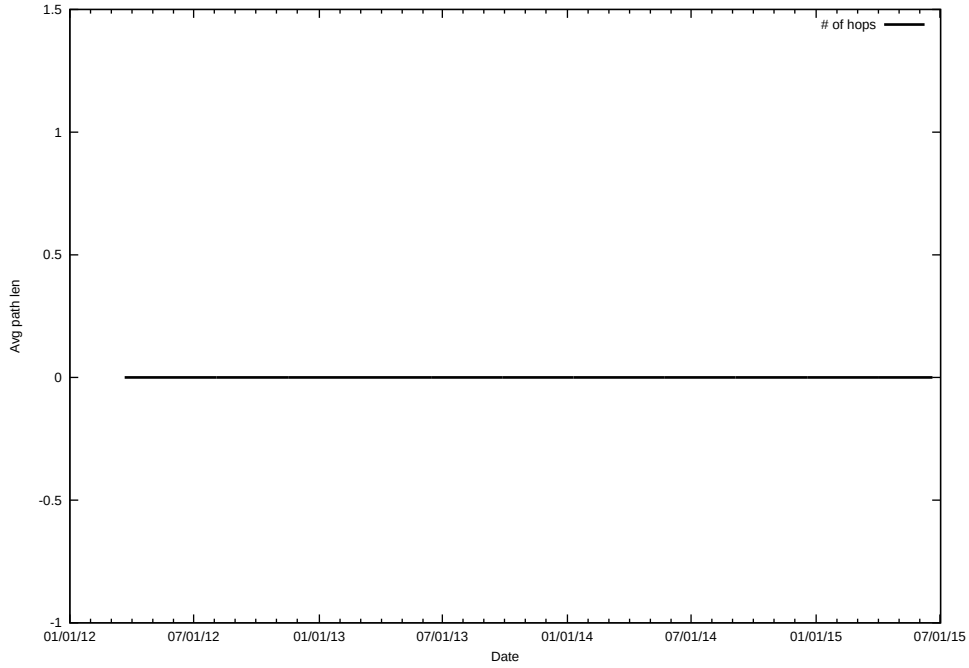
/104


1.5
1
0.5
0
-0.5
-1
Avg path len
# of hops
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

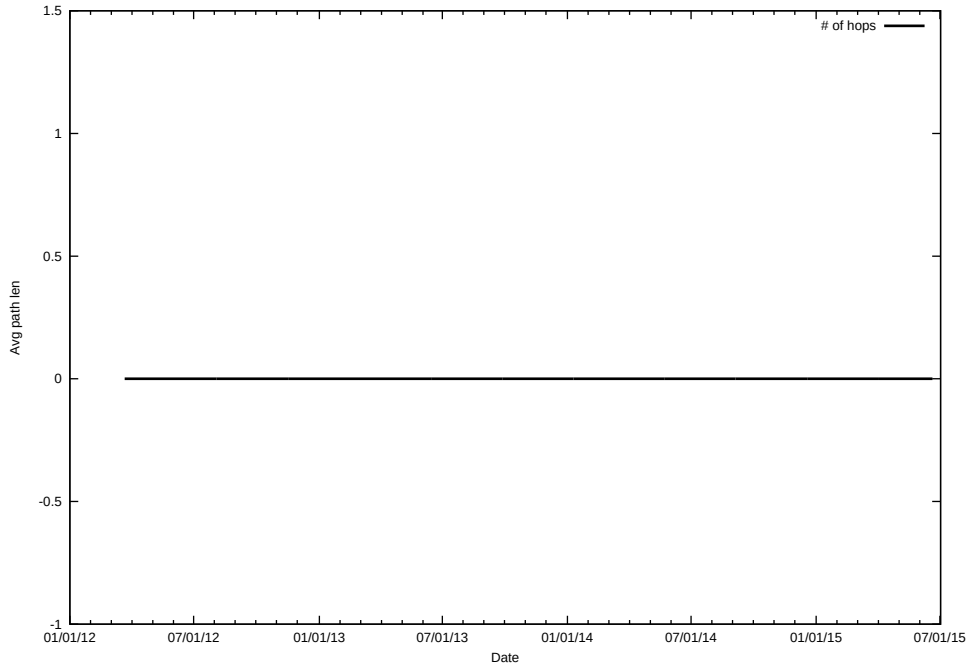
/105


1.5
1
0.5
0
-0.5
-1
Avg path len
# of hops
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

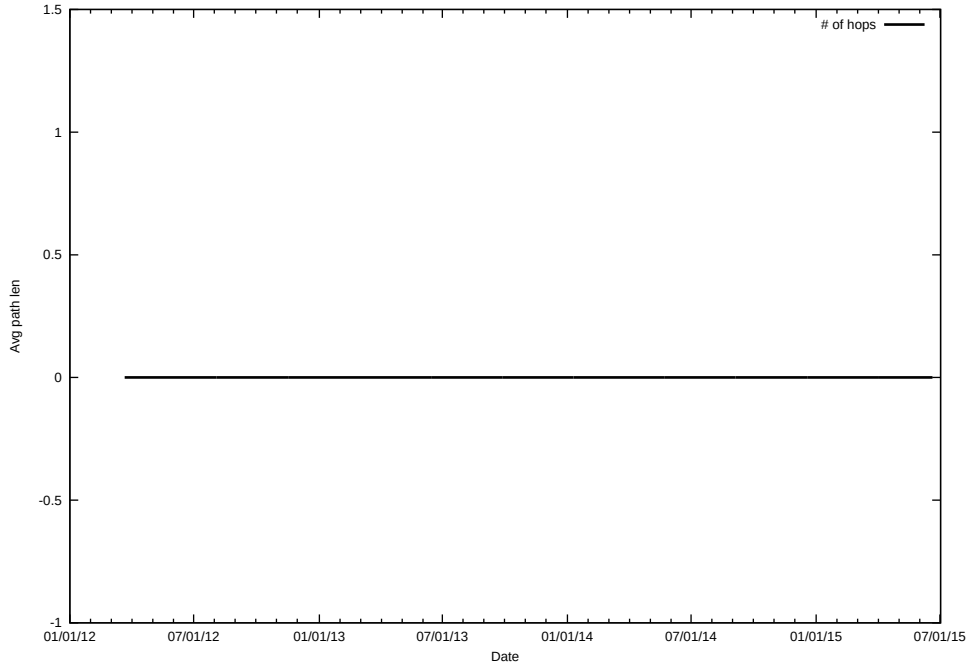
/106


1.5
1
0.5
0
-0.5
-1
Avg path len
# of hops
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

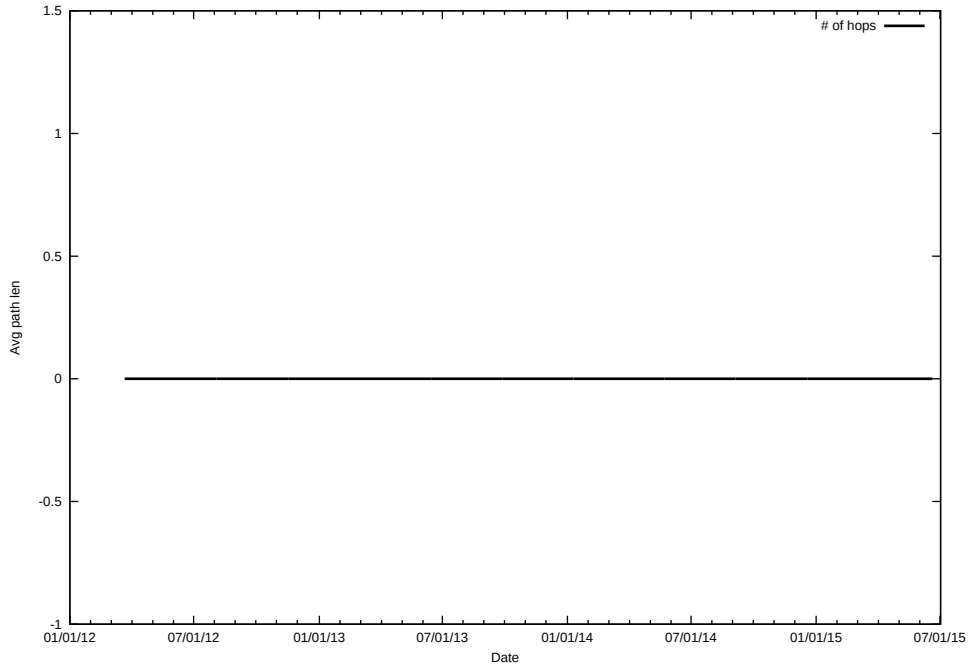
/107


1.5
1
0.5
0
-0.5
-1
Avg path len
# of hops
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

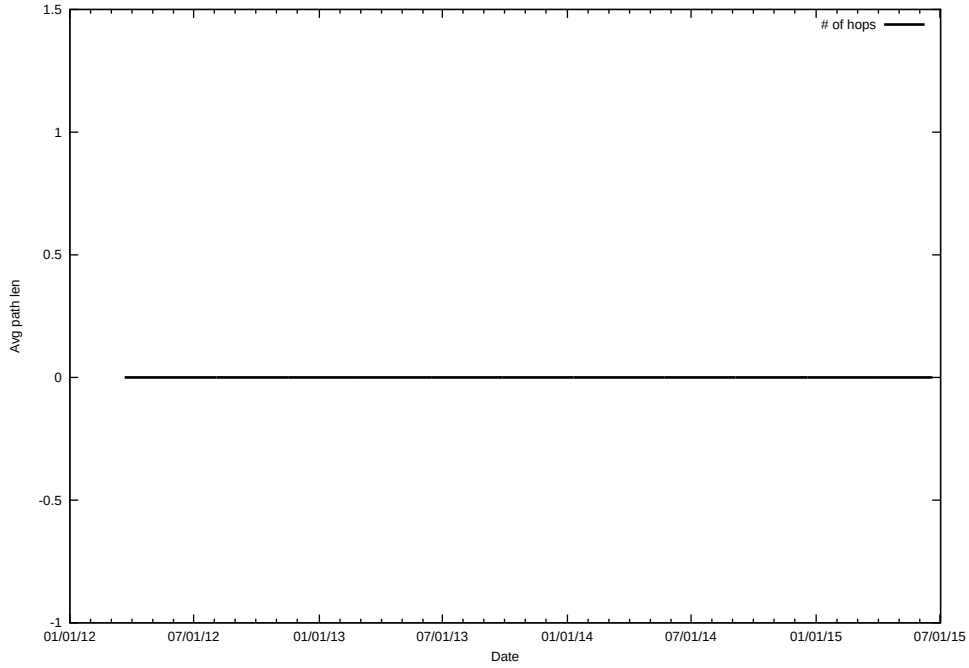
/108


1.5
1
0.5
0
-0.5
-1
Avg path len
# of hops
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

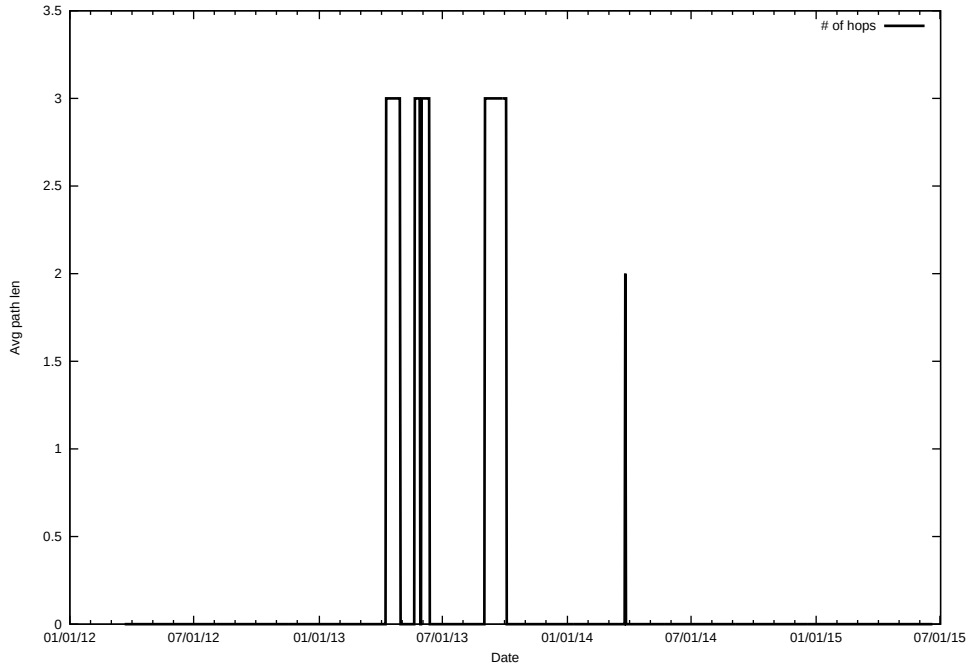
/109


1.5
1
0.5
0
-0.5
-1
Avg path len
# of hops
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

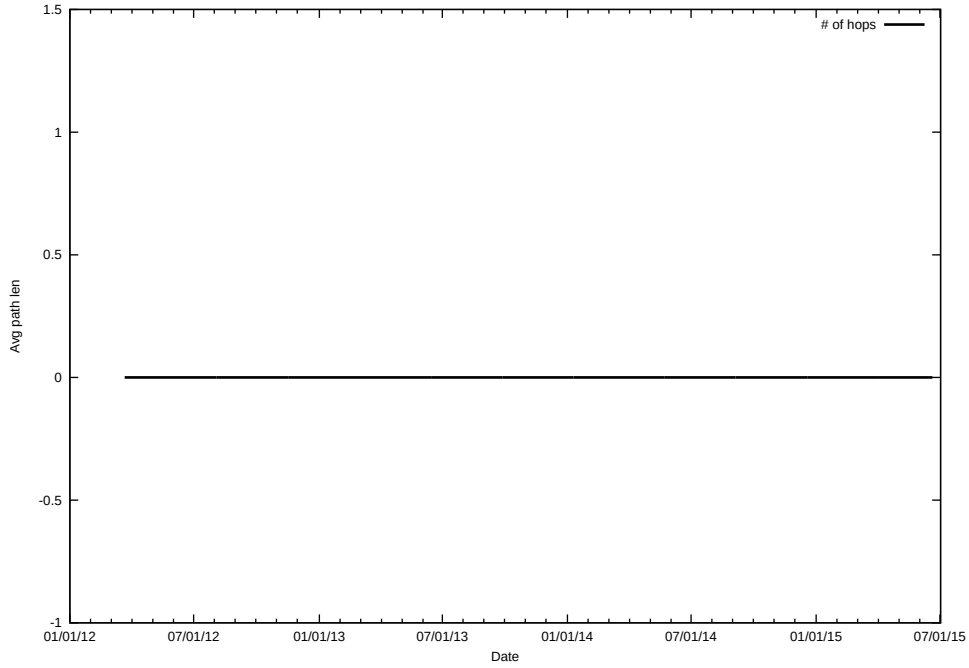
/110


1.5
1
0.5
0
-0.5
-1
Avg path len
# of hops
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

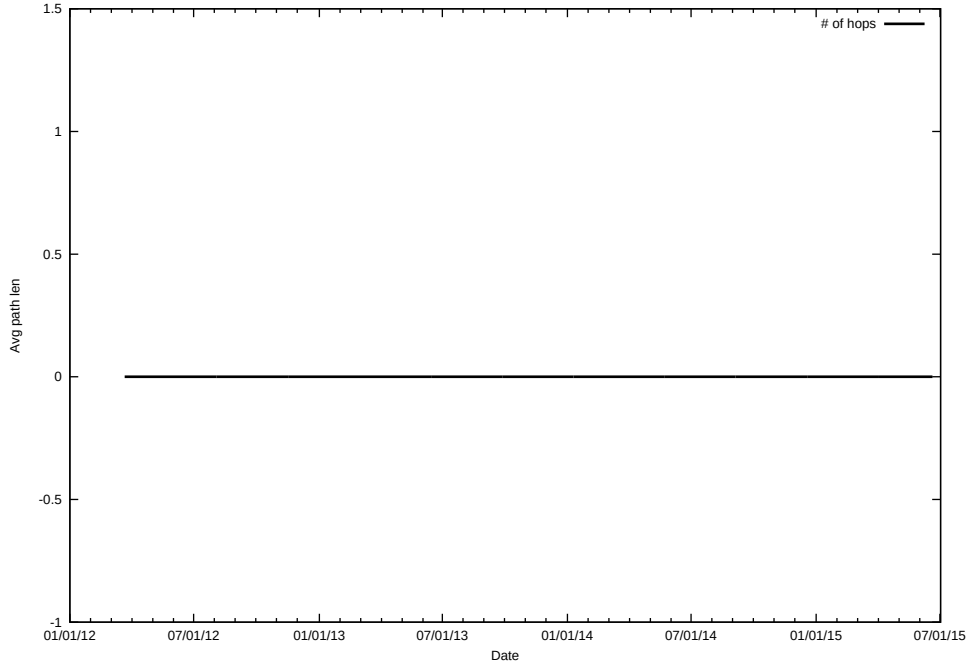
/111


3.5
3
2.5
2
1.5
1
0.5
0
Avg path len
# of hops
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

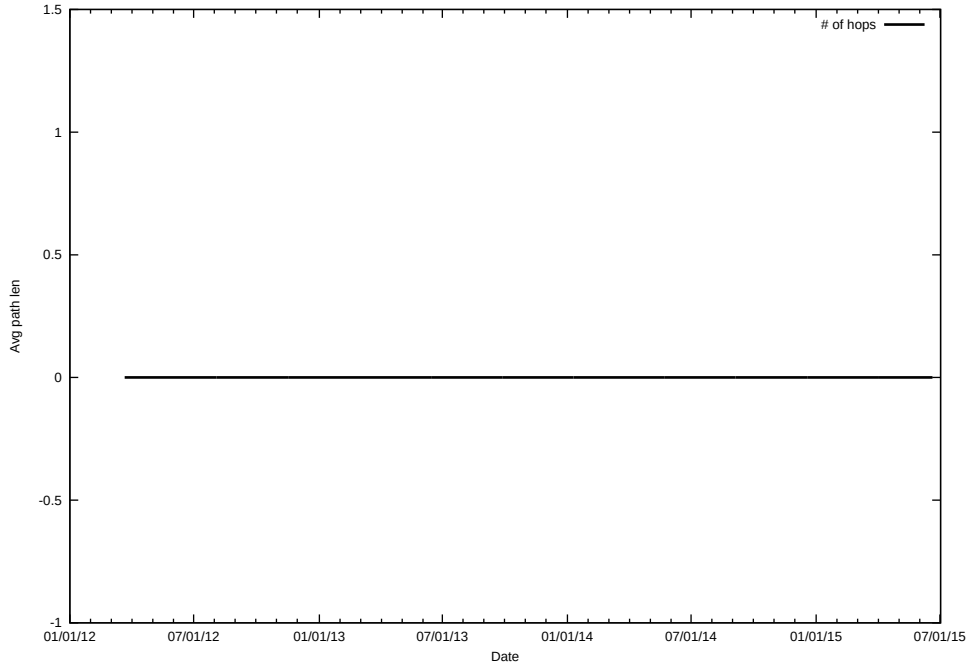
/112


1.5
1
0.5
0
-0.5
-1
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

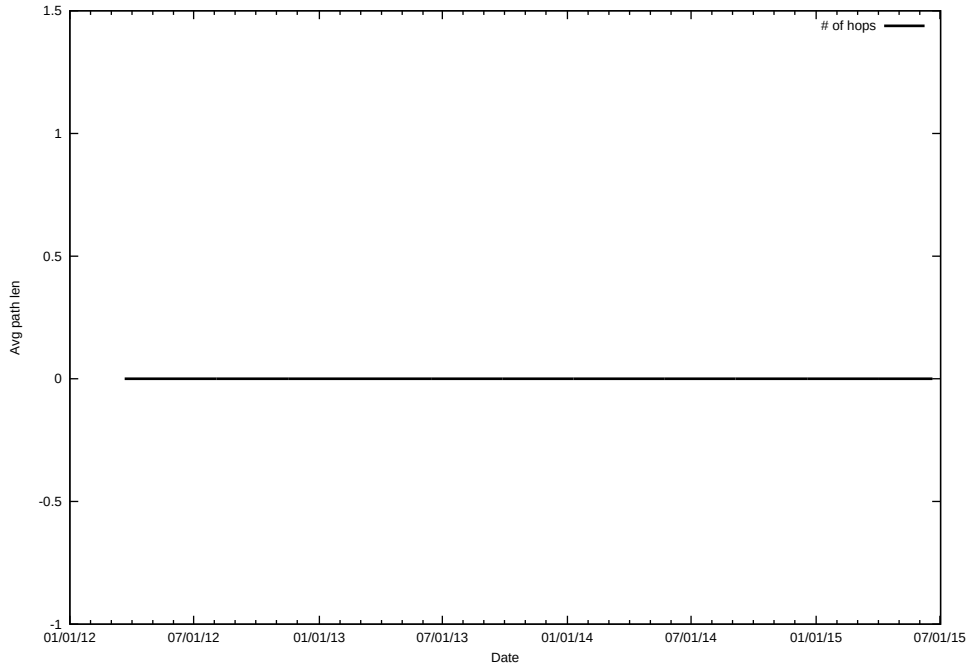
/113


1.5
1
0.5
0
-0.5
-1
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

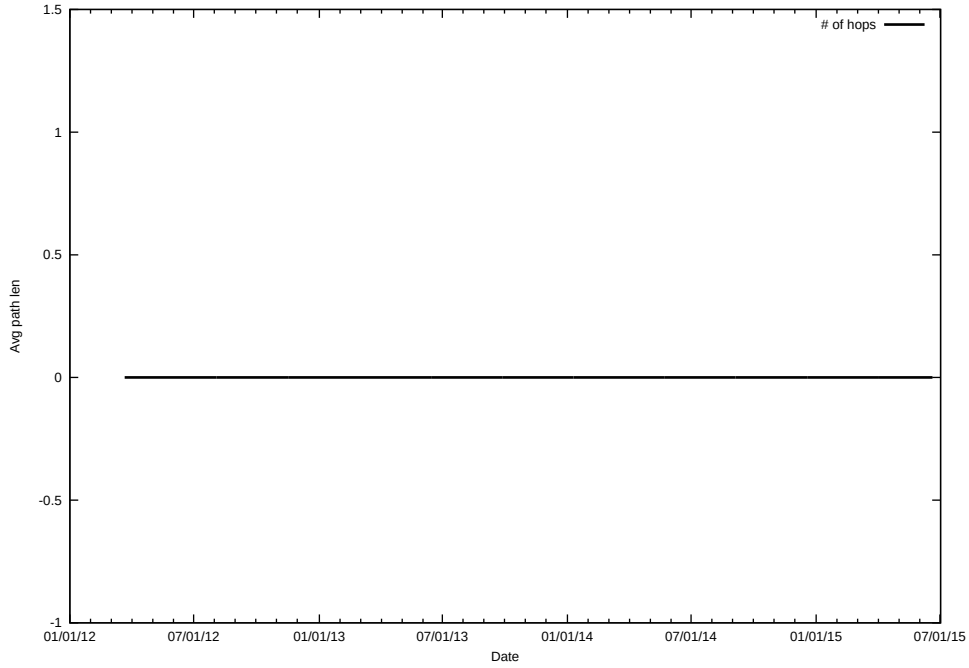
/114


1.5
1
0.5
0
-0.5
-1
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

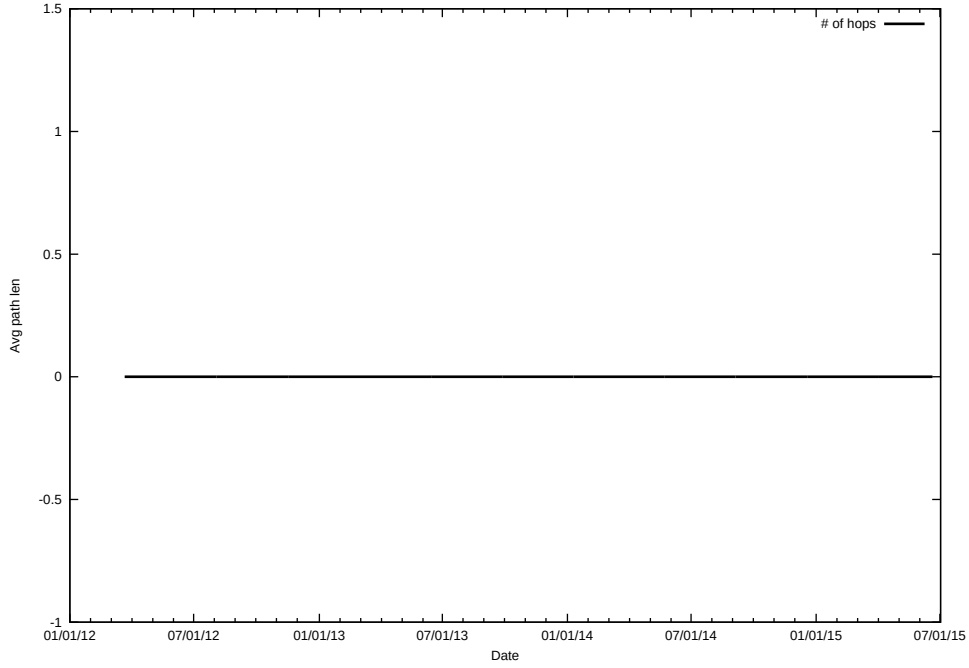
/115


1.5
1
0.5
0
-0.5
-1
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

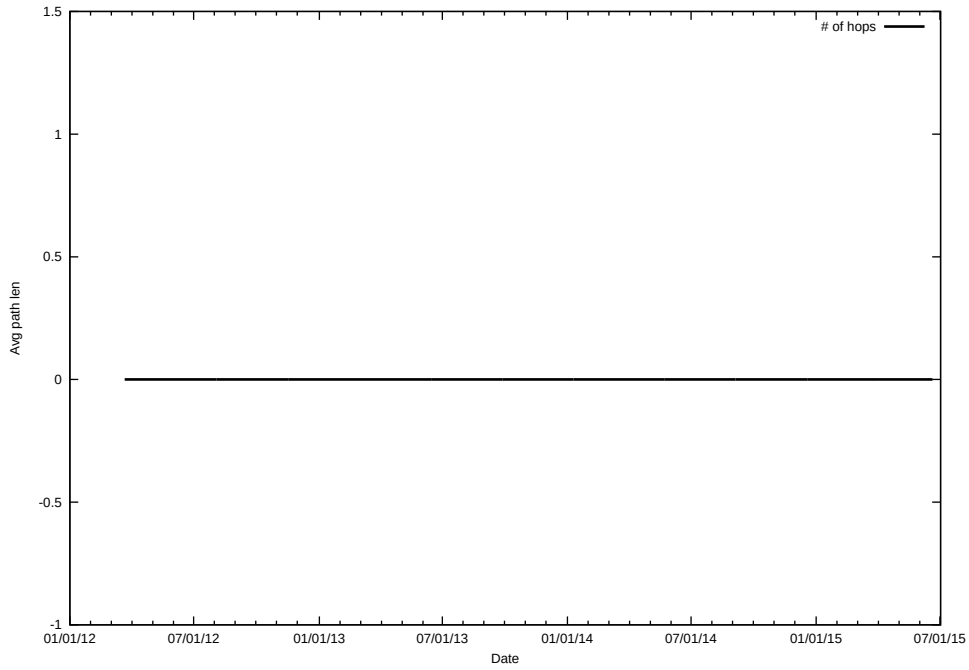
/116


1.5
1
0.5
0
-0.5
-1
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

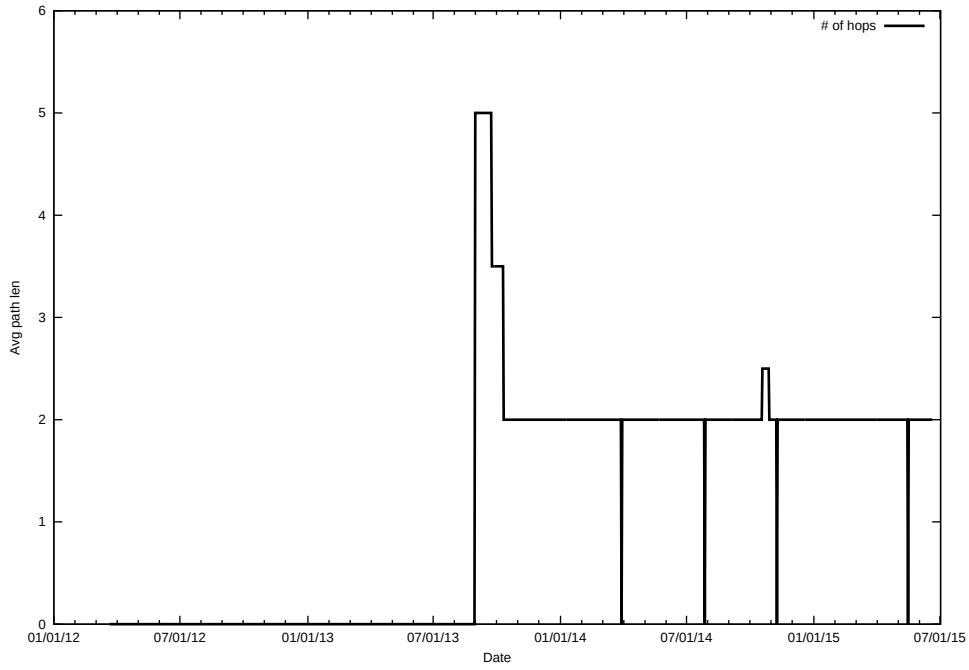
/117


1.5
1
0.5
0
-0.5
-1
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

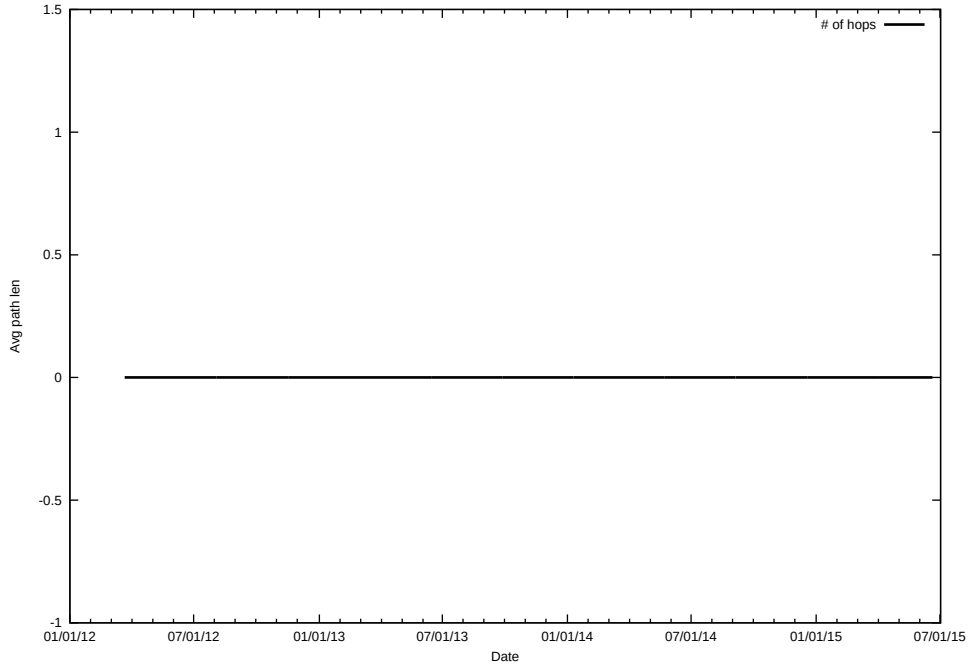
/118


1.5
1
0.5
0
-0.5
-1
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

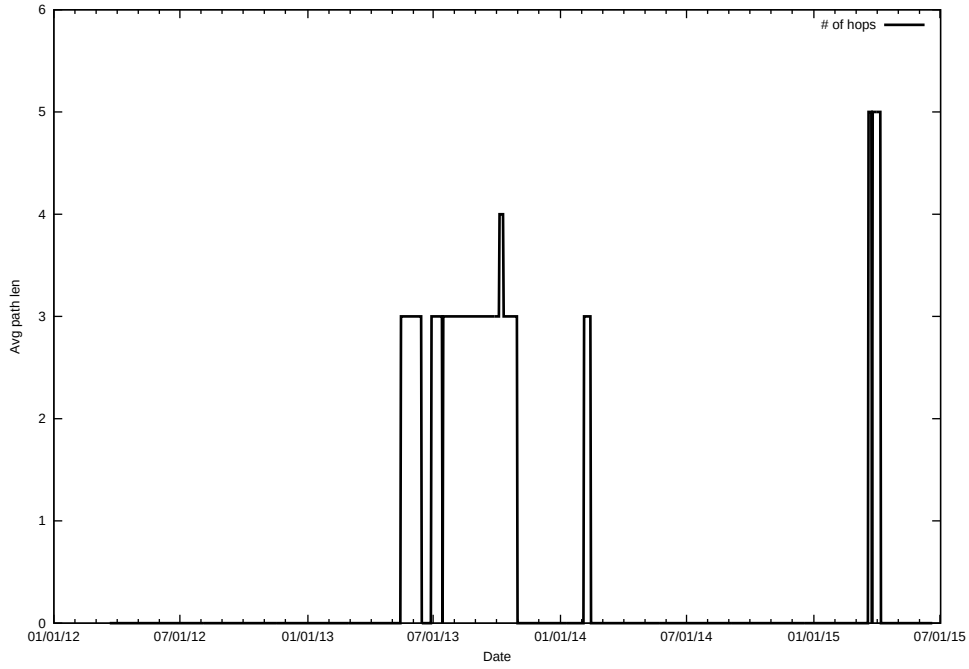
/119


6
5
4
3
2
1
0
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

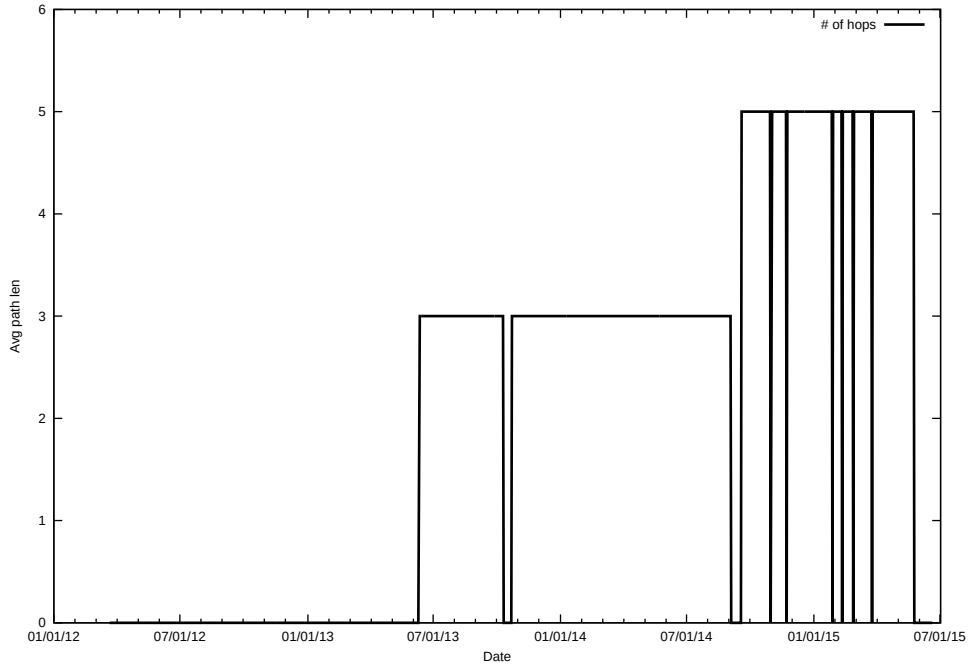
/120


1.5
1
0.5
0
-0.5
-1
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

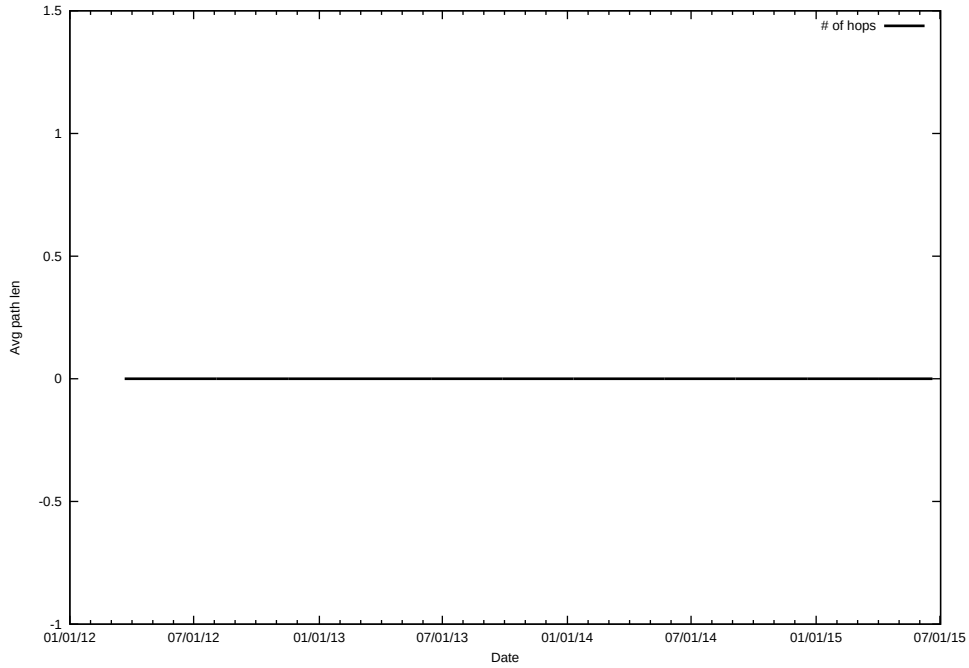
/121


6
5
4
3
2
1
0
Avg path len
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date
# of hops

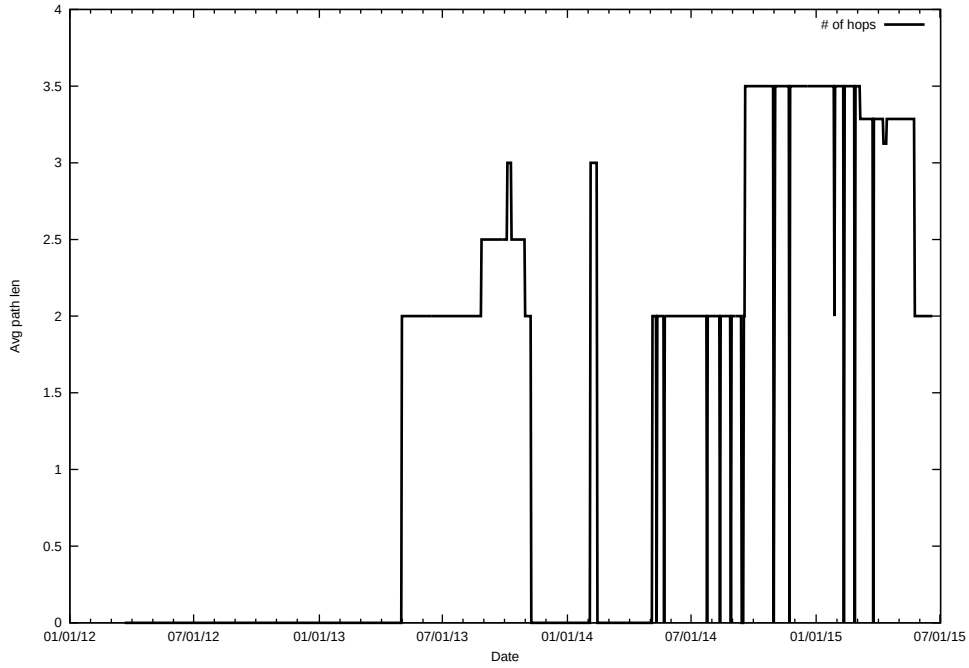
/122


6
5
4
3
2
1
0
Avg path len
# of hops
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

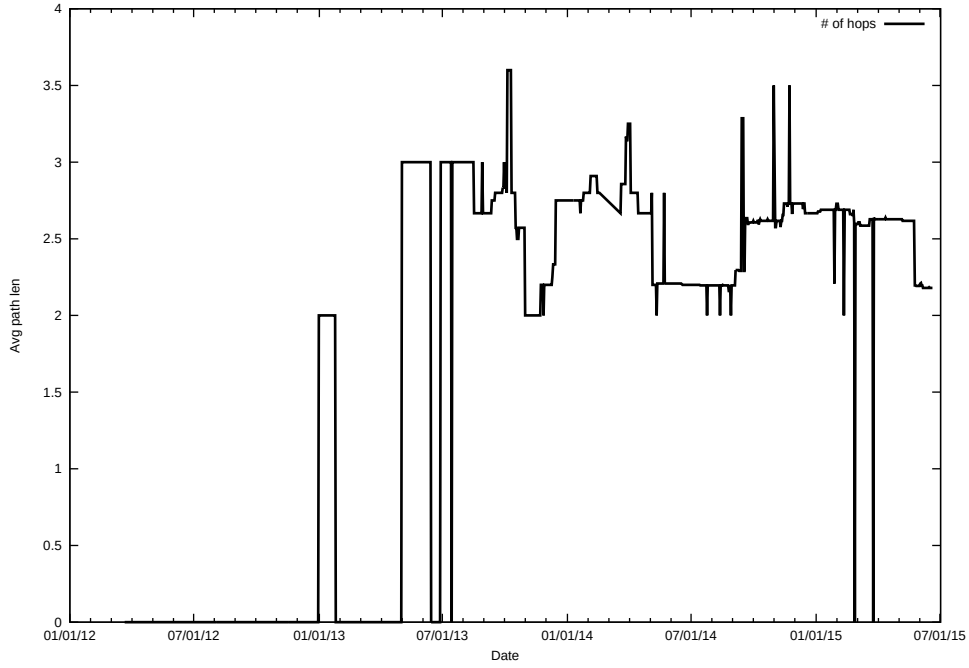
/123


1.5
1
0.5
0
-0.5
-1
Avg path len
# of hops
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15
Date

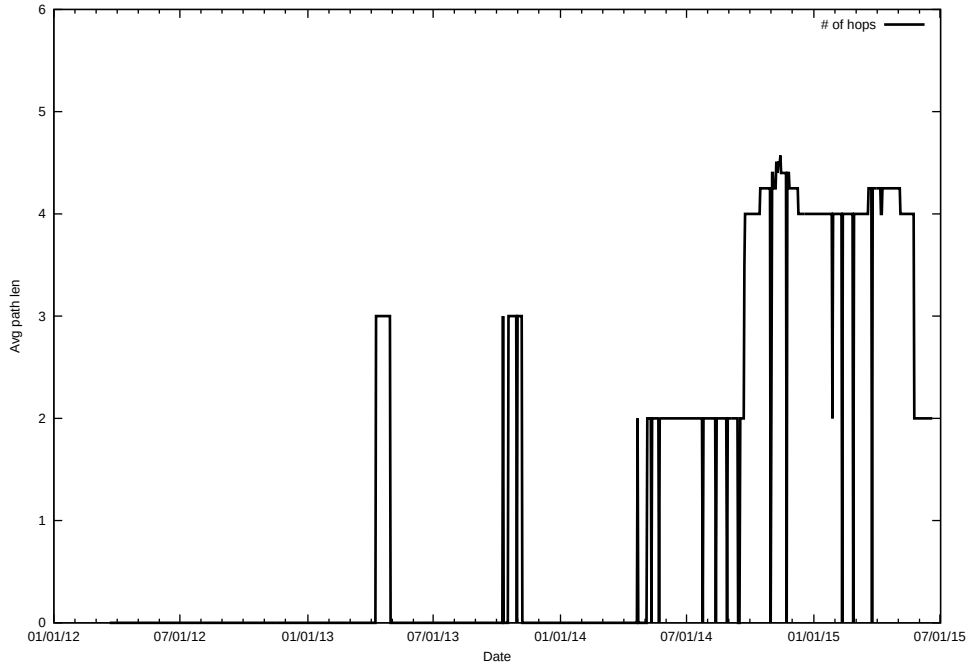
/124


# of hops
Avg path len
Date
0
0.5
1
1.5
2
2.5
3
3.5
4
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15

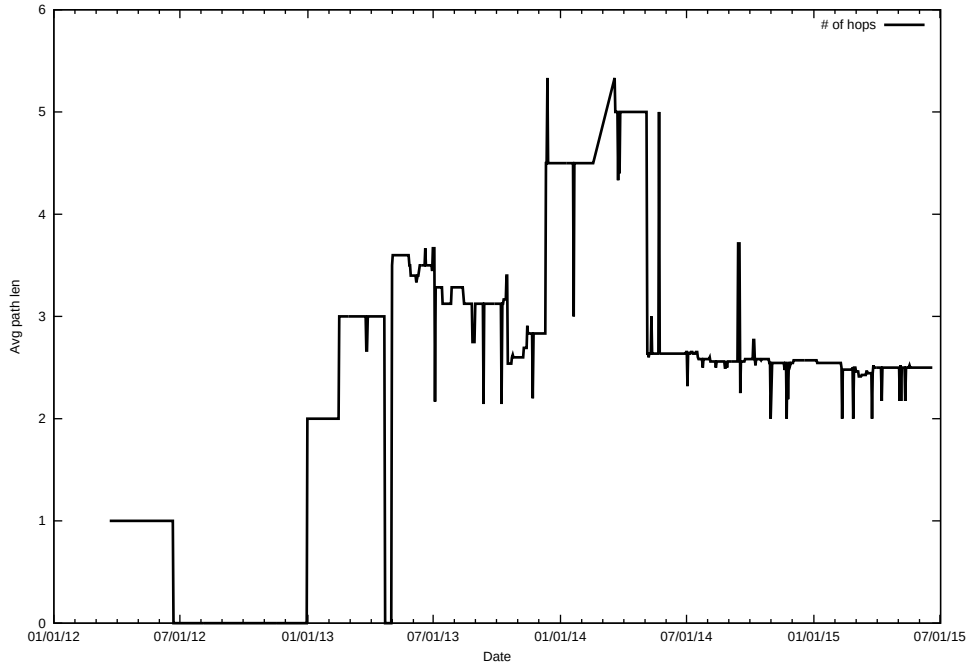
/125


# of hops
Avg path len
Date
0
0.5
1
1.5
2
2.5
3
3.5
4
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15

/126

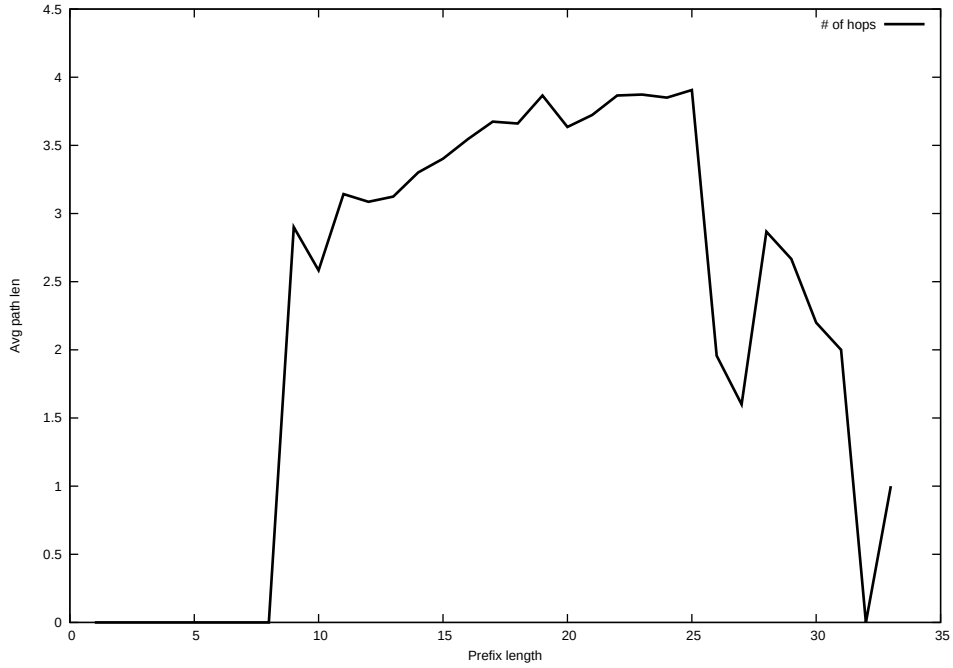

Avg path len
Date
# of hops
0
1
2
3
4
5
6
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15

/127

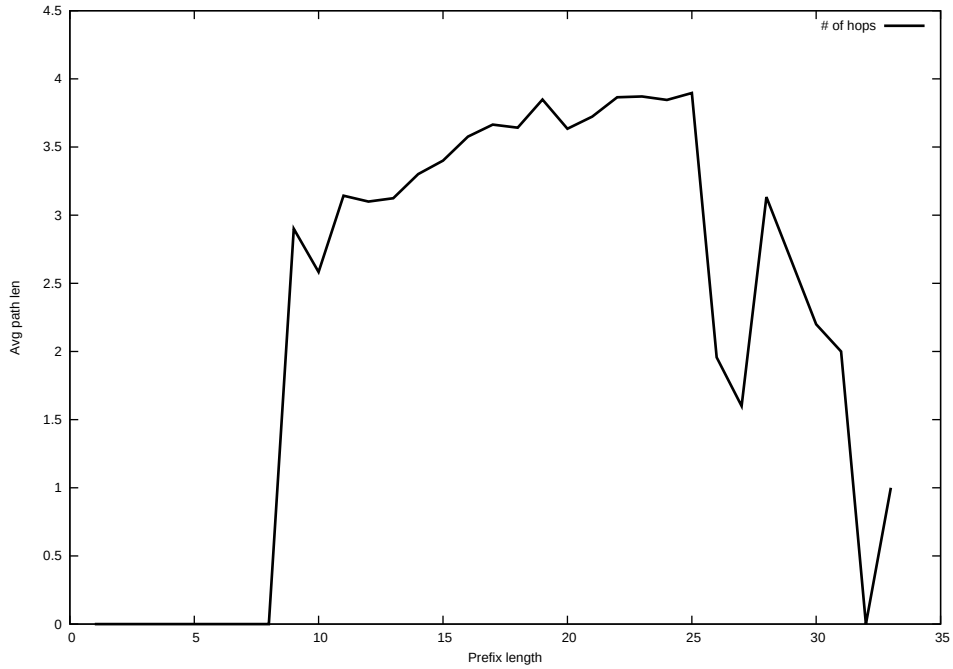

Avg path len
Date
# of hops
0
1
2
3
4
5
6
01/01/12
07/01/12
01/01/13
07/01/13
01/01/14
07/01/14
01/01/15
07/01/15

/128

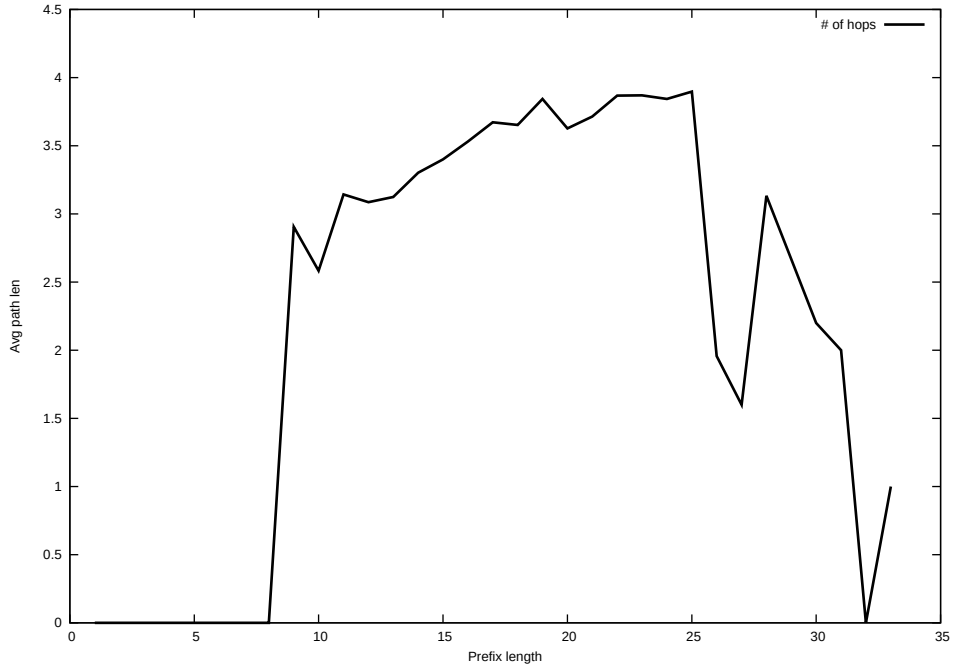
A.6 Daily IPv4 BGP avg path length by prefix length



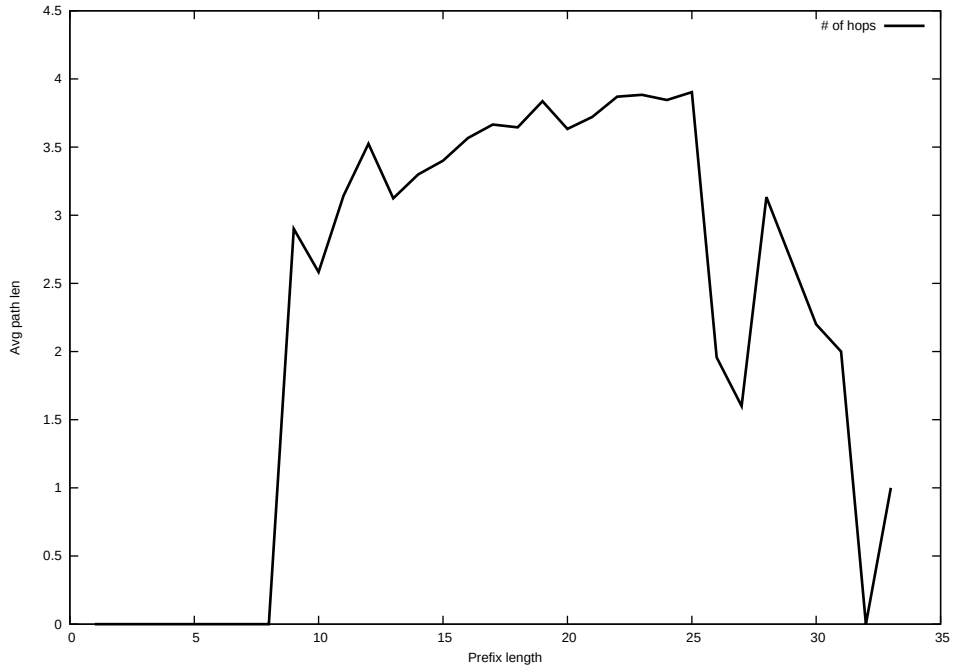
2012-03-22



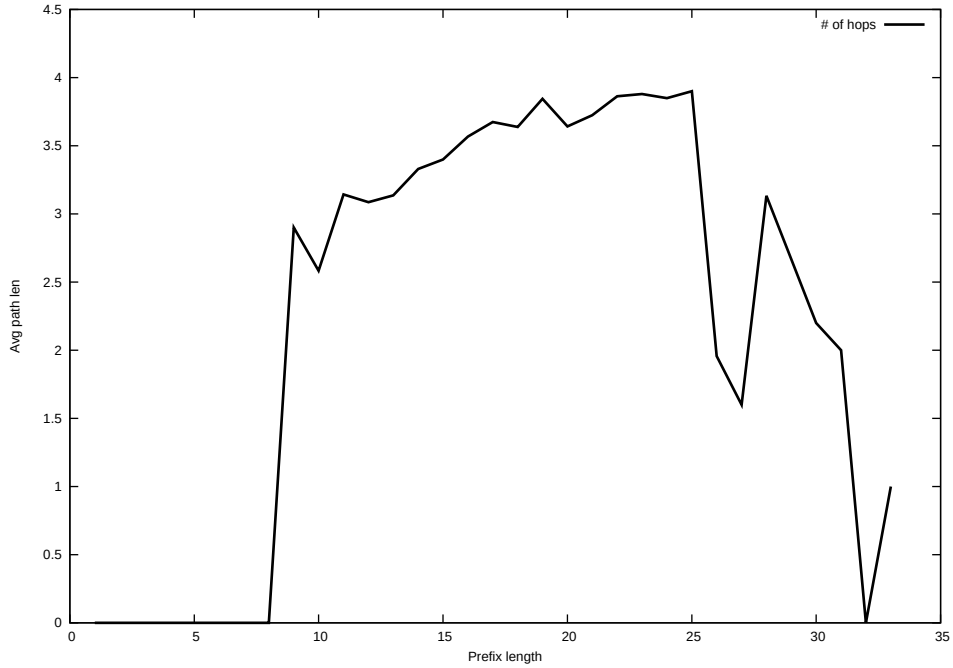
2012-03-23



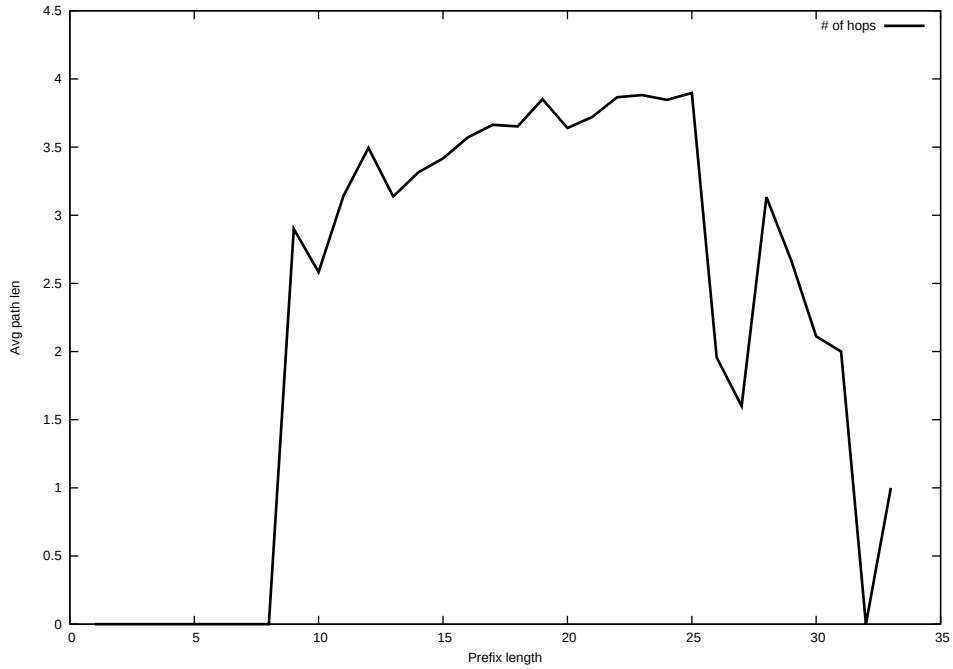
2012-03-24



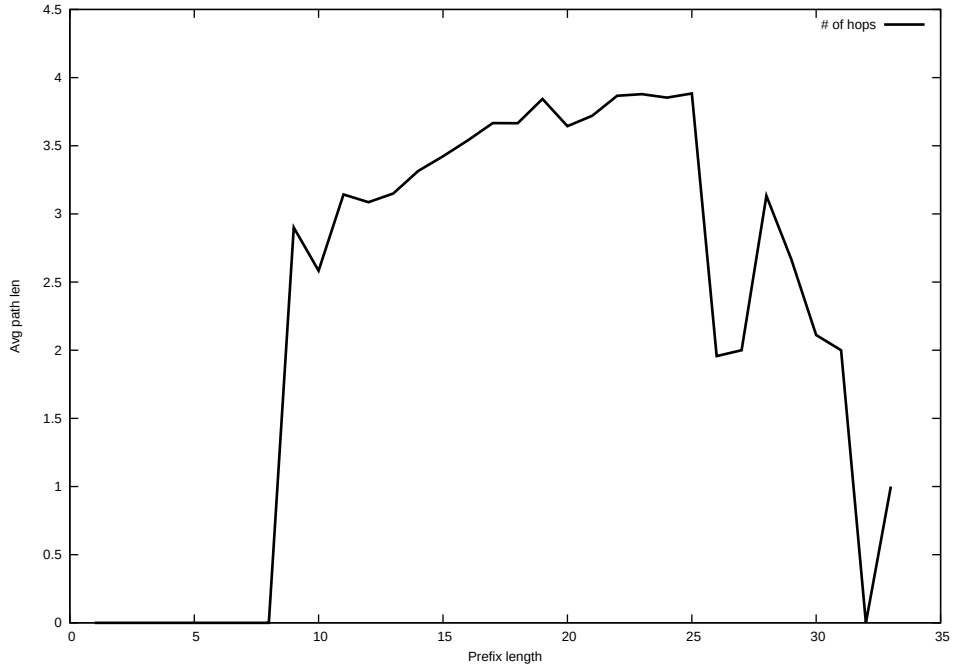
2012-03-25



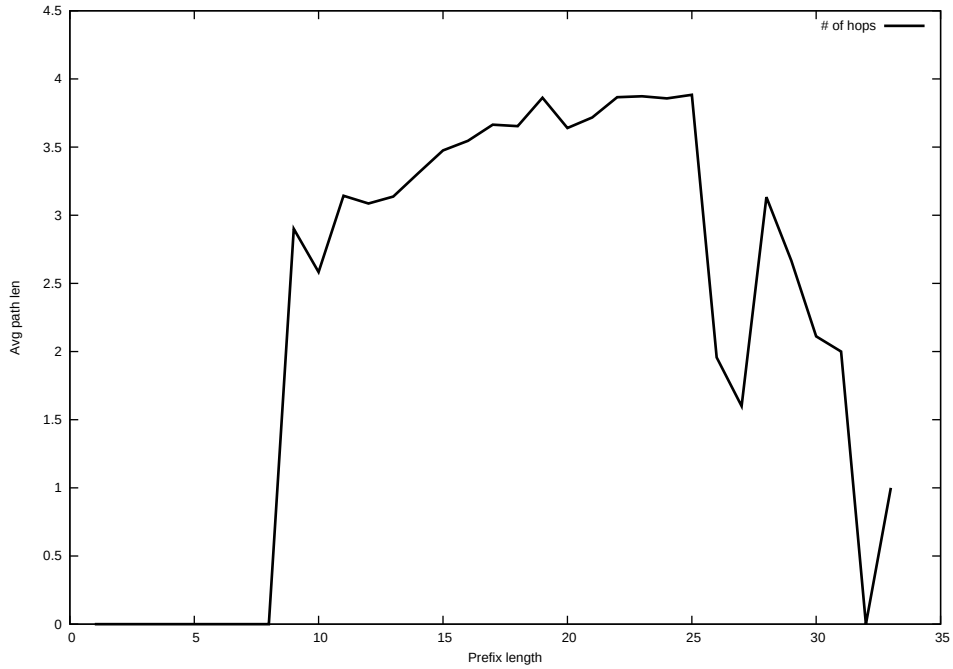
2012-03-26



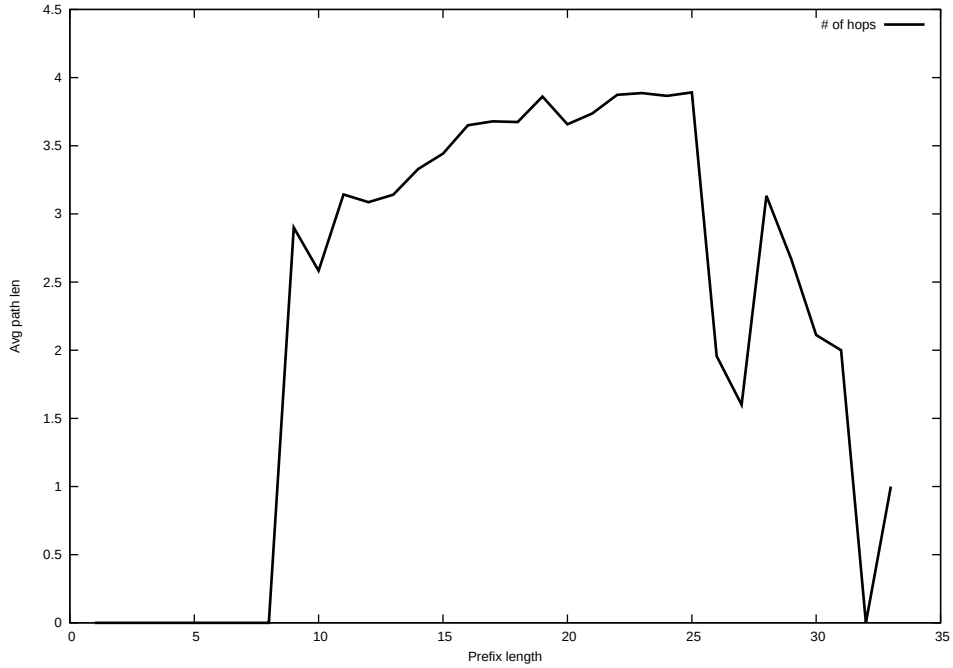
2012-03-27



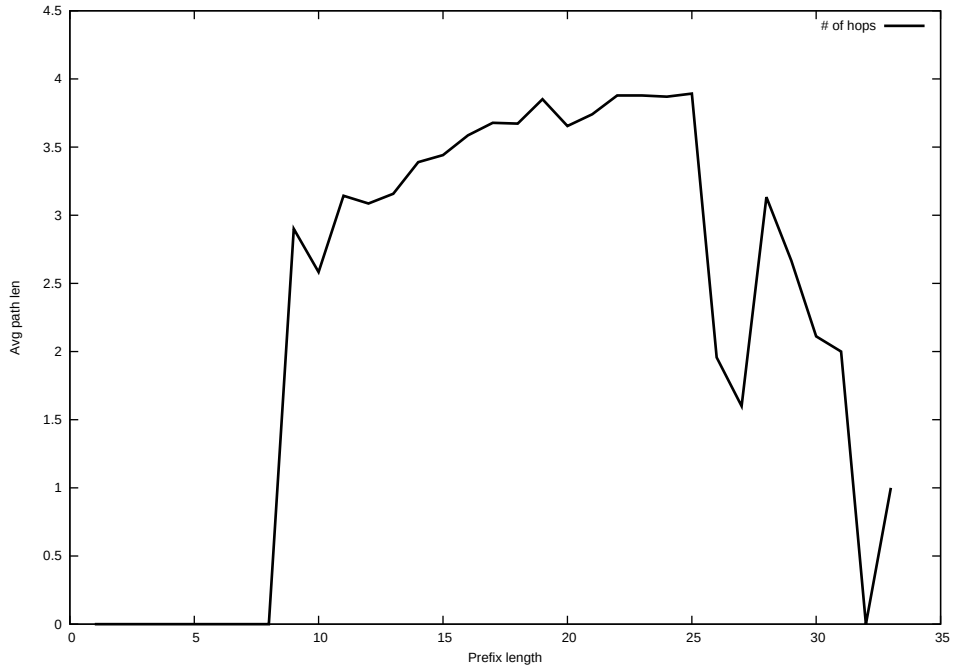
2012-03-28



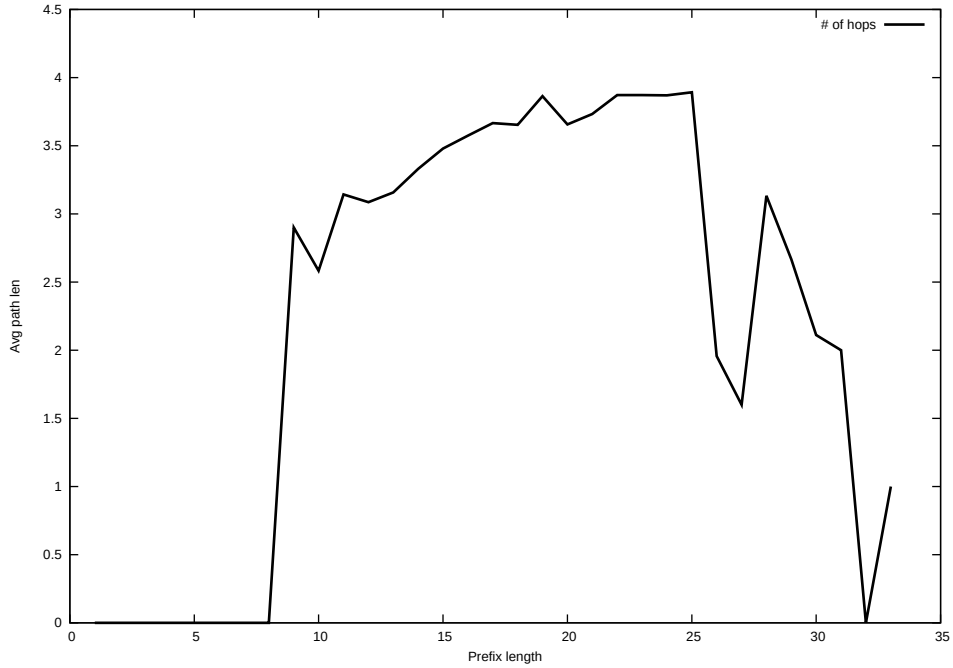
2012-03-29



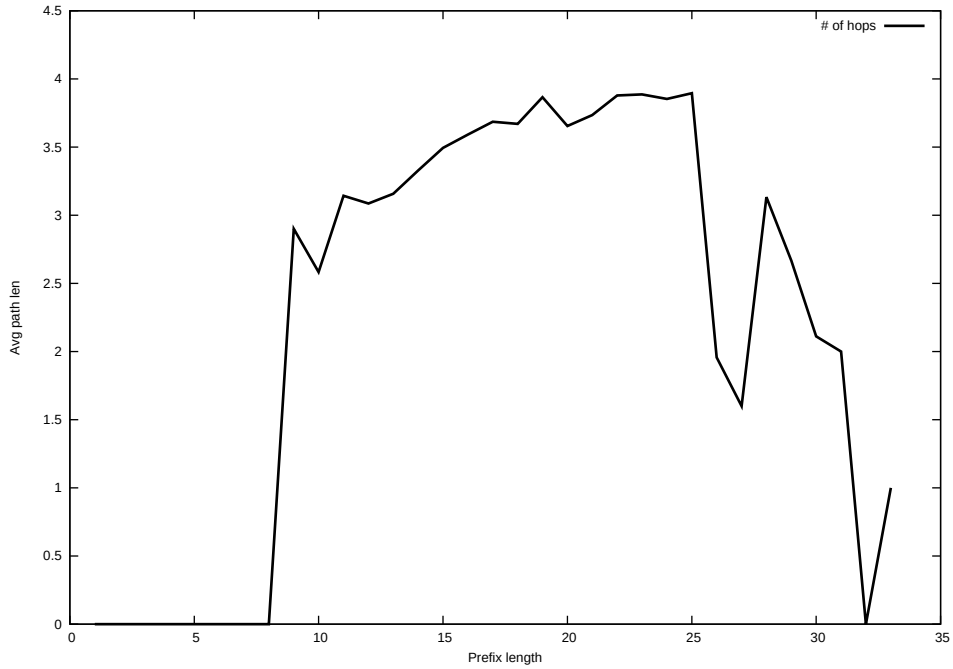
2012-03-30



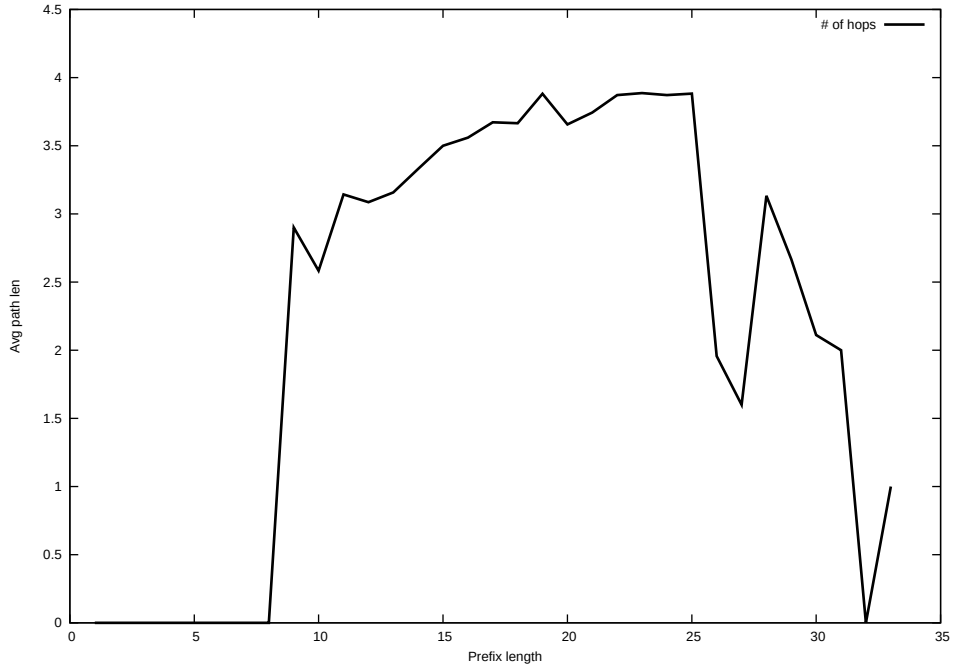
2012-03-31



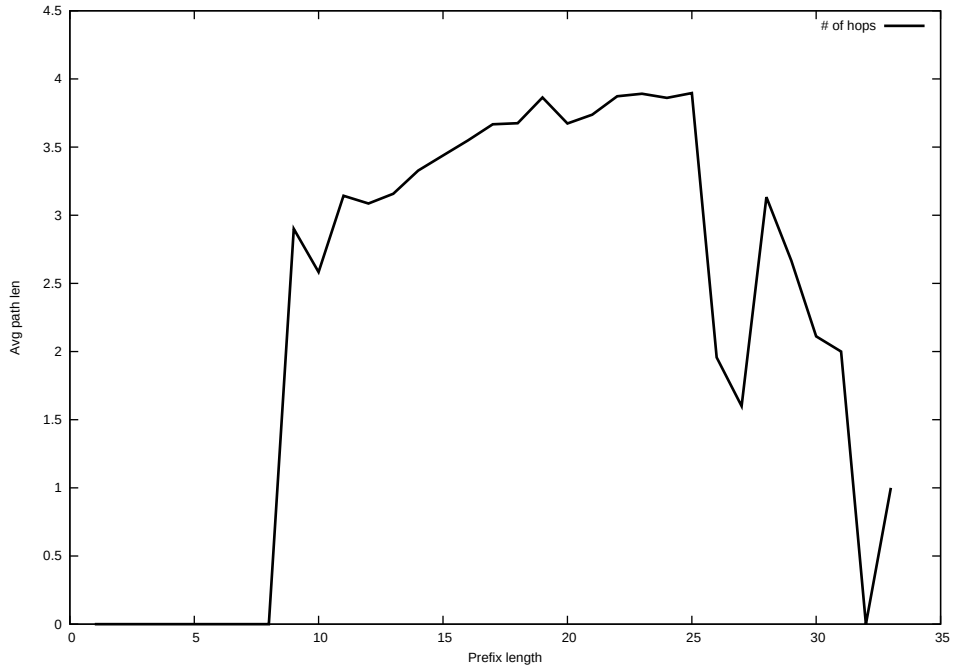
2012-04-01



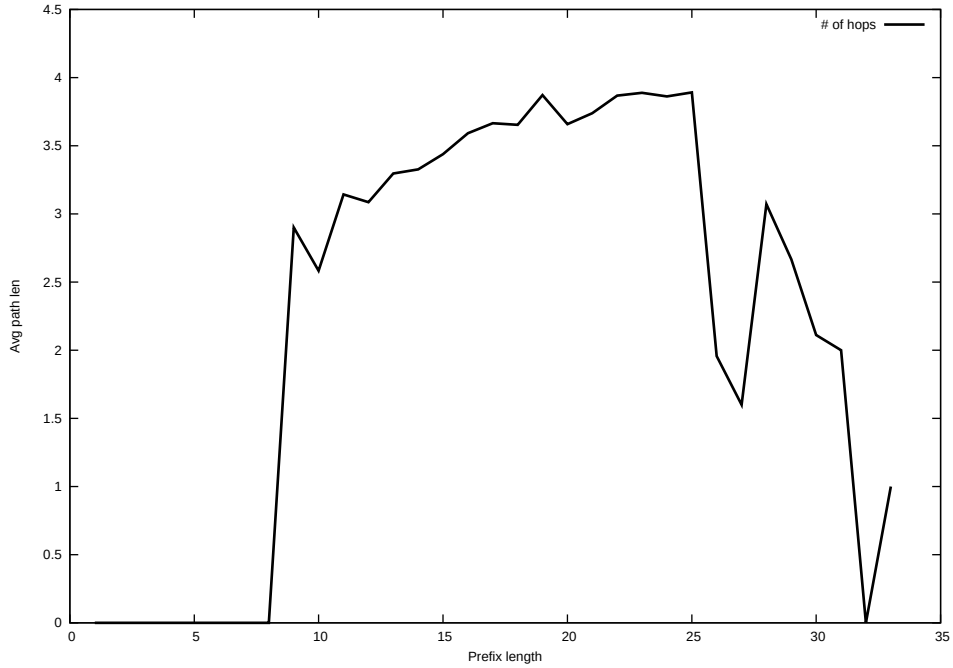
2012-04-02



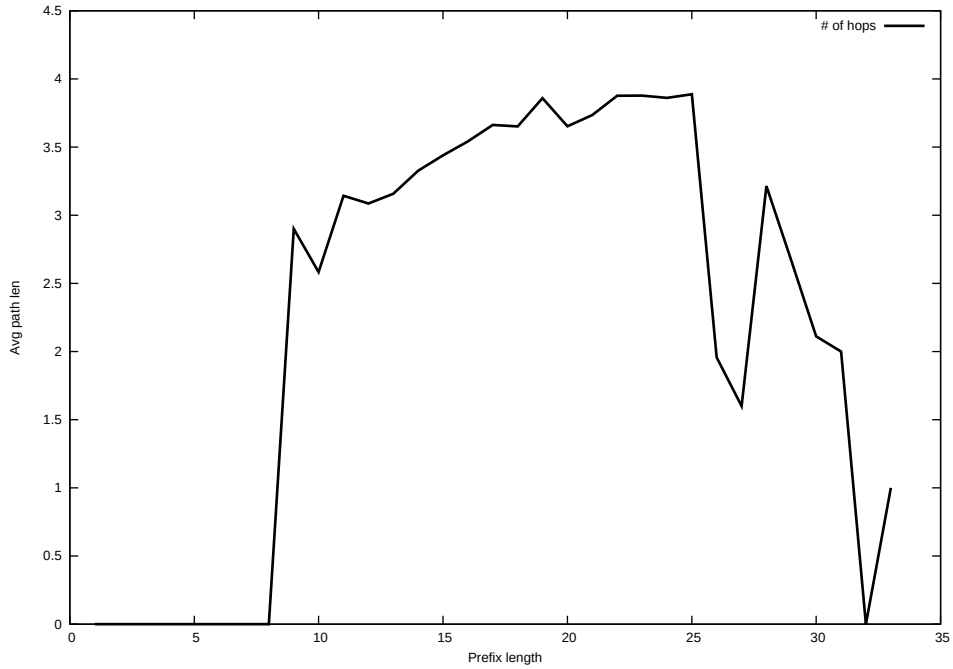
2012-04-03



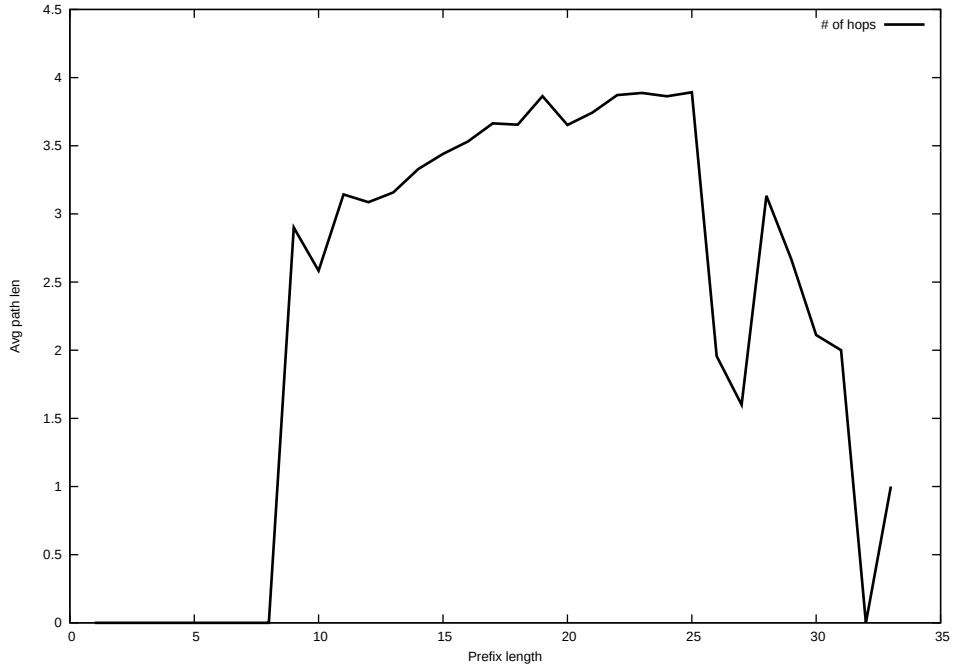
2012-04-04



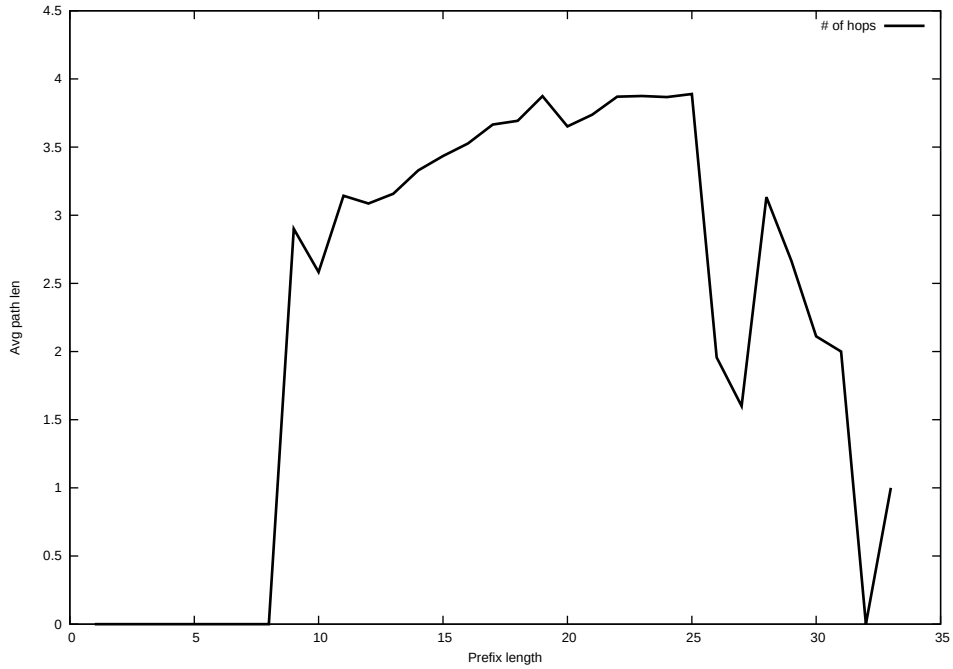
2012-04-05



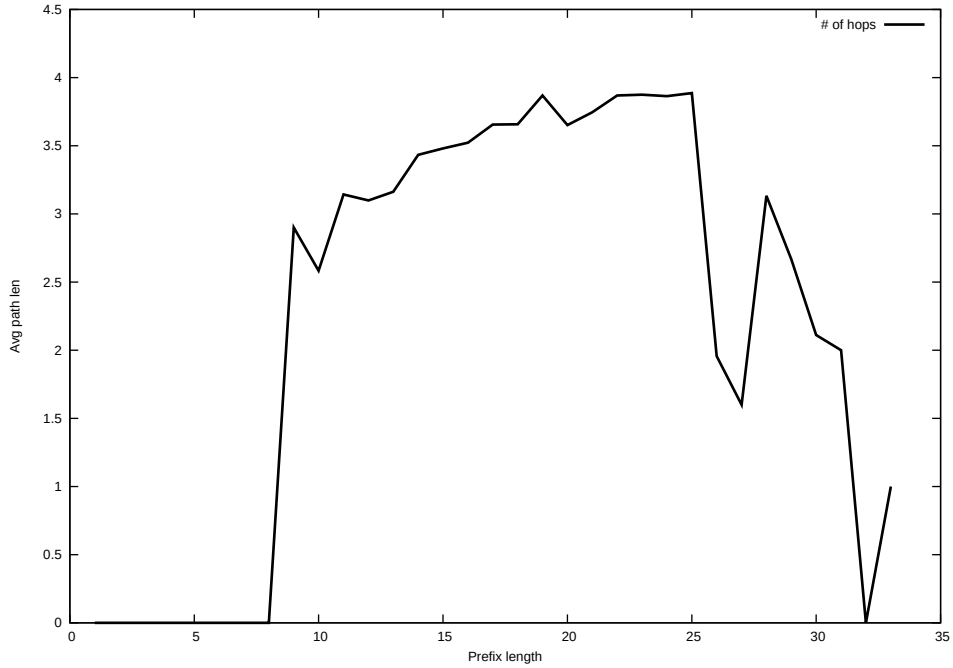
2012-04-06



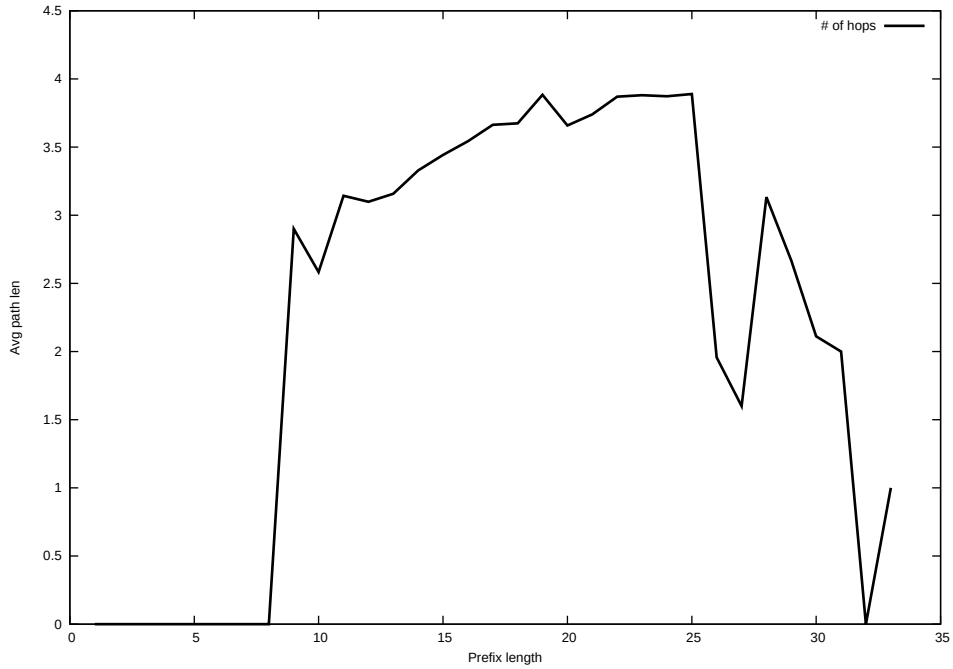
2012-04-07



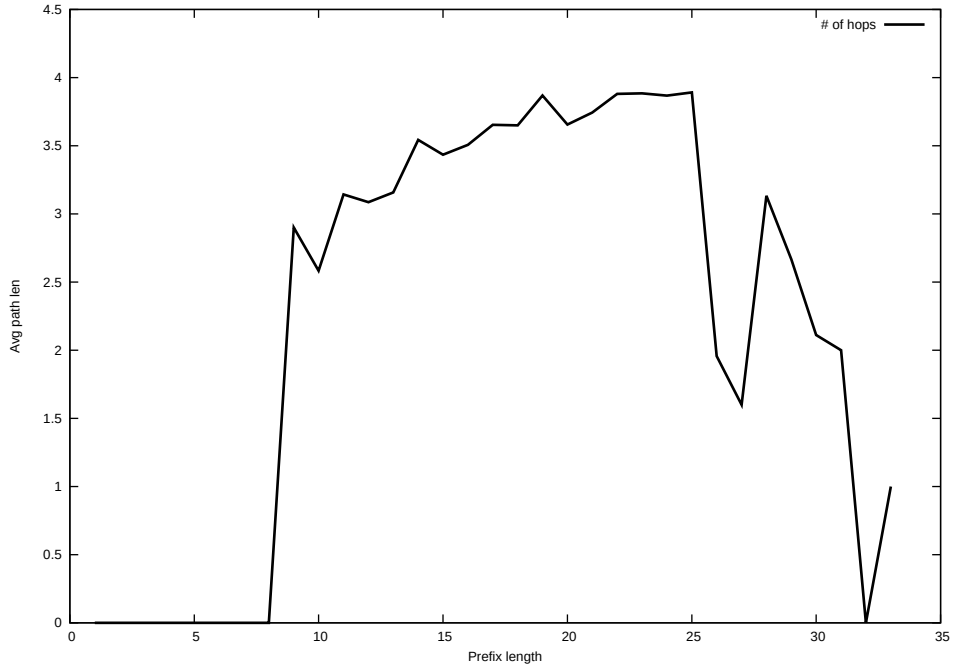
2012-04-08



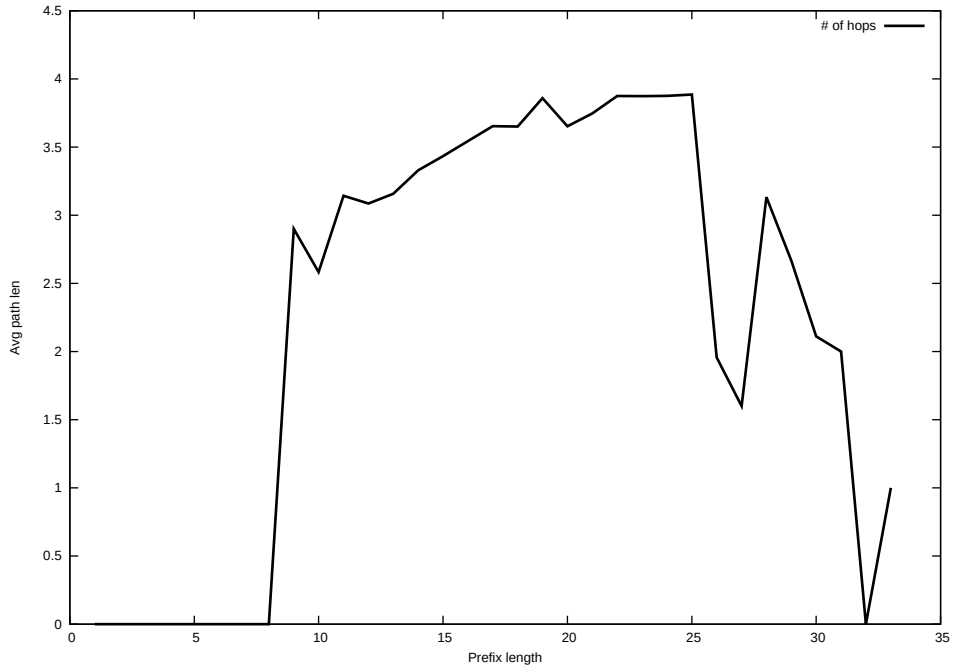
2012-04-09



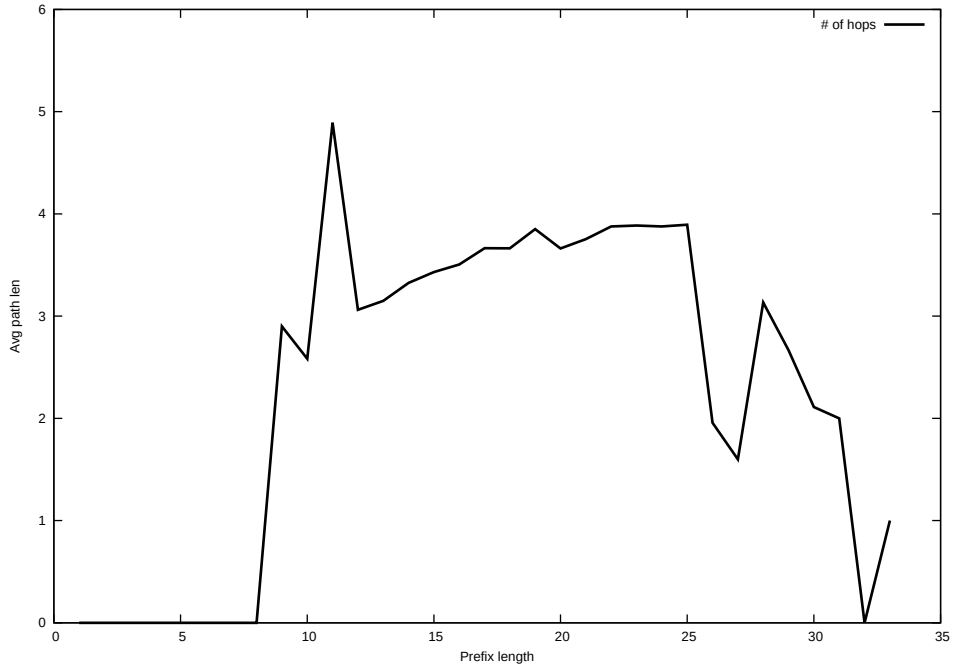
2012-04-10



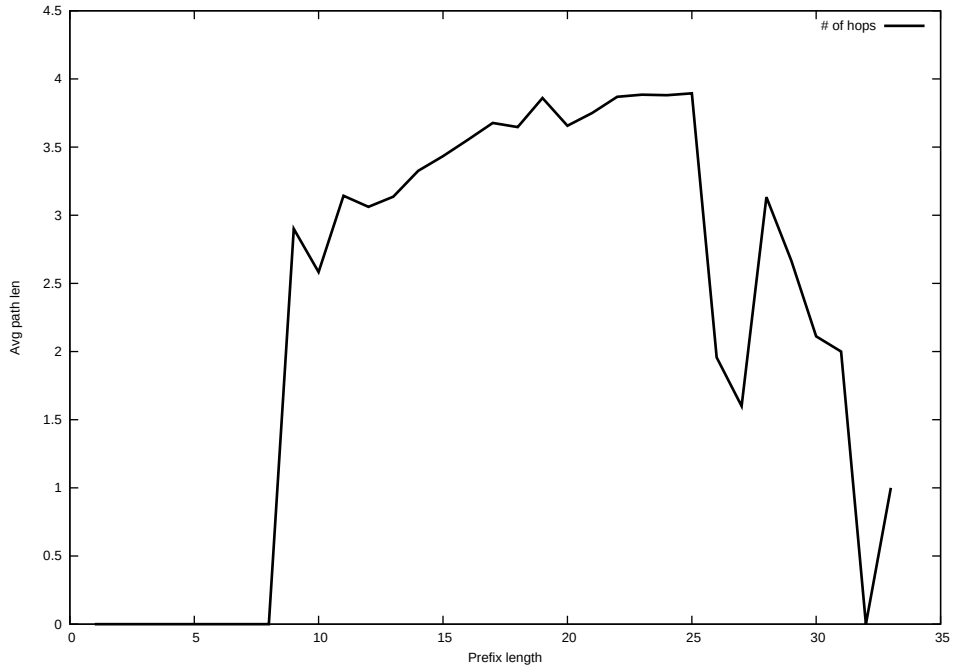
2012-04-11



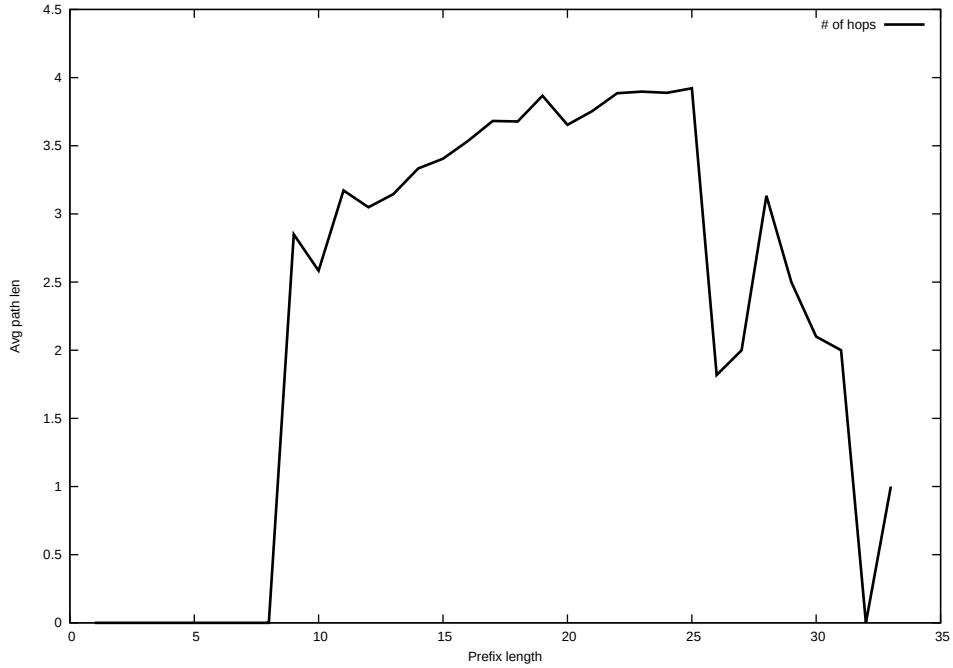
2012-04-12



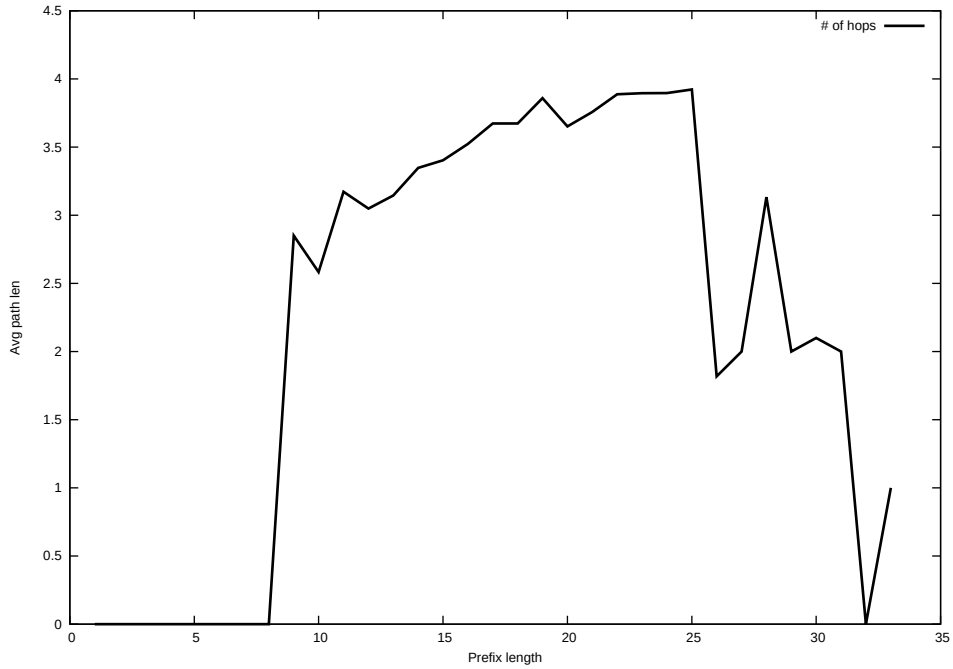
2012-04-13



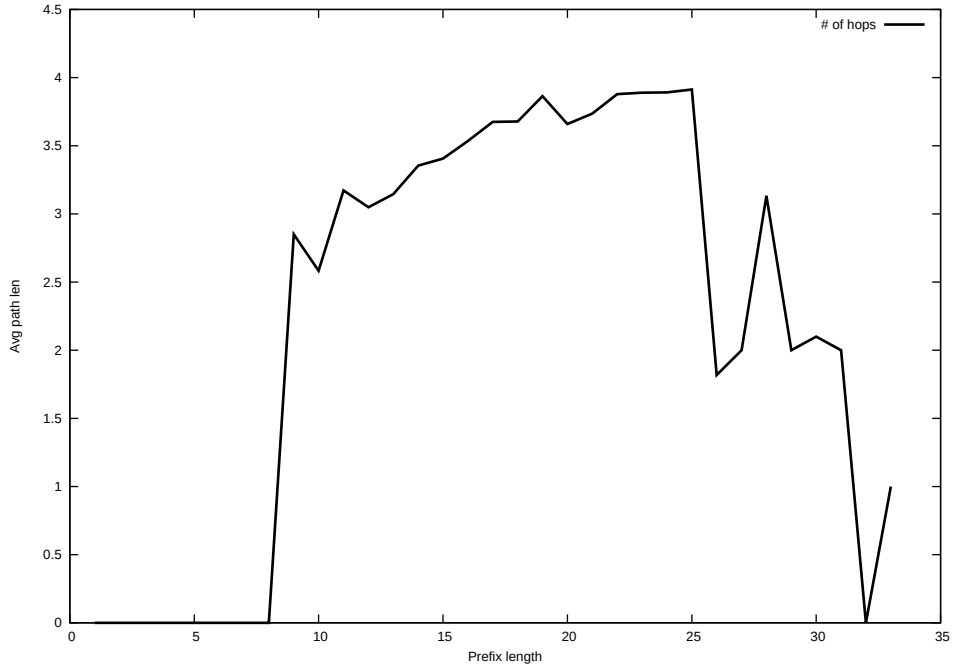
2012-04-14



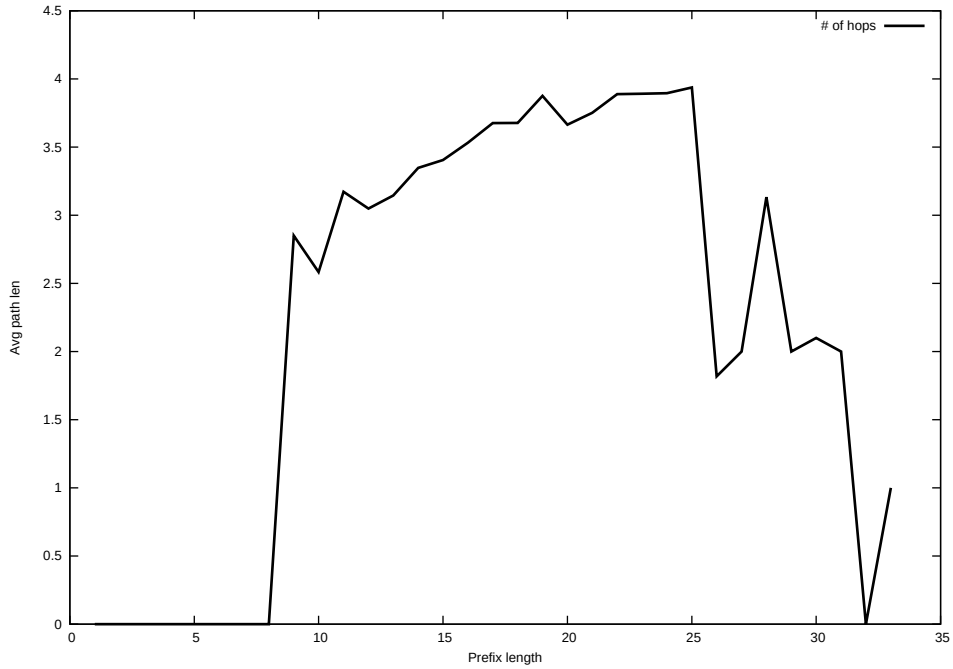
2012-05-16



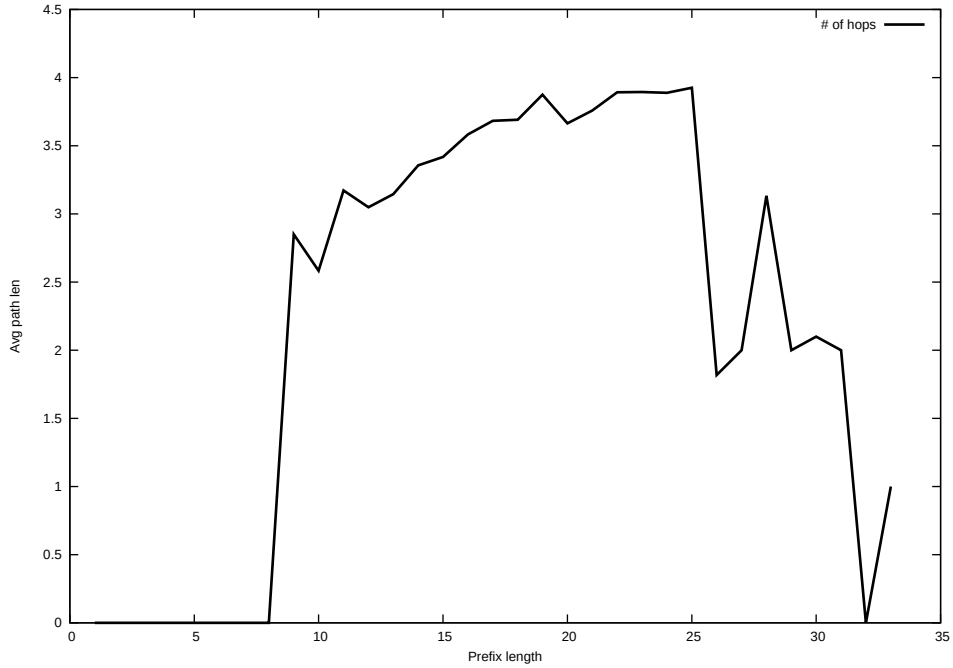
2012-05-17



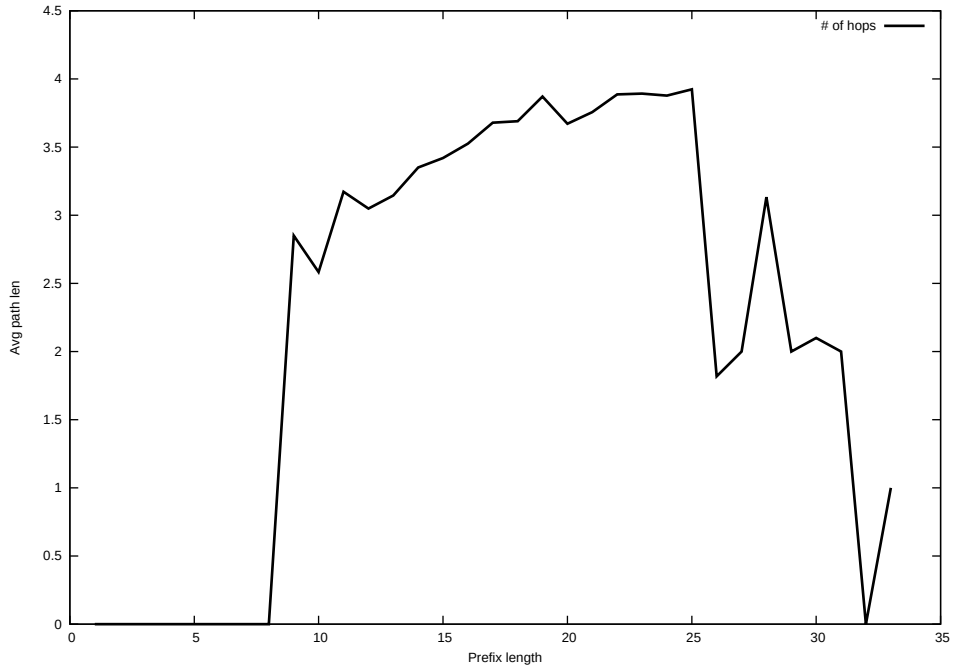
2012-05-19



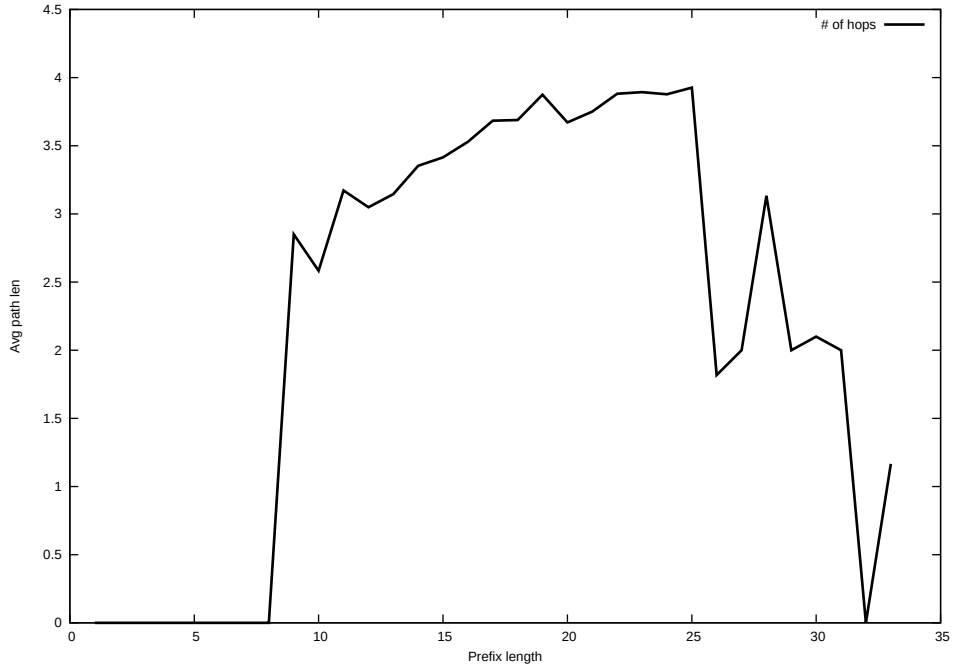
2012-05-20



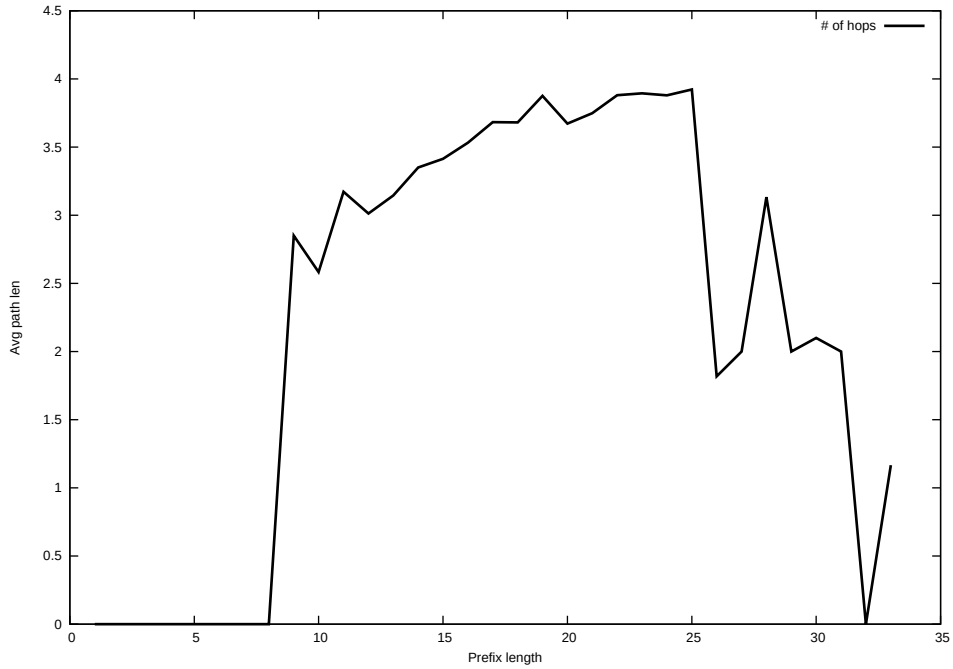
2012-05-21



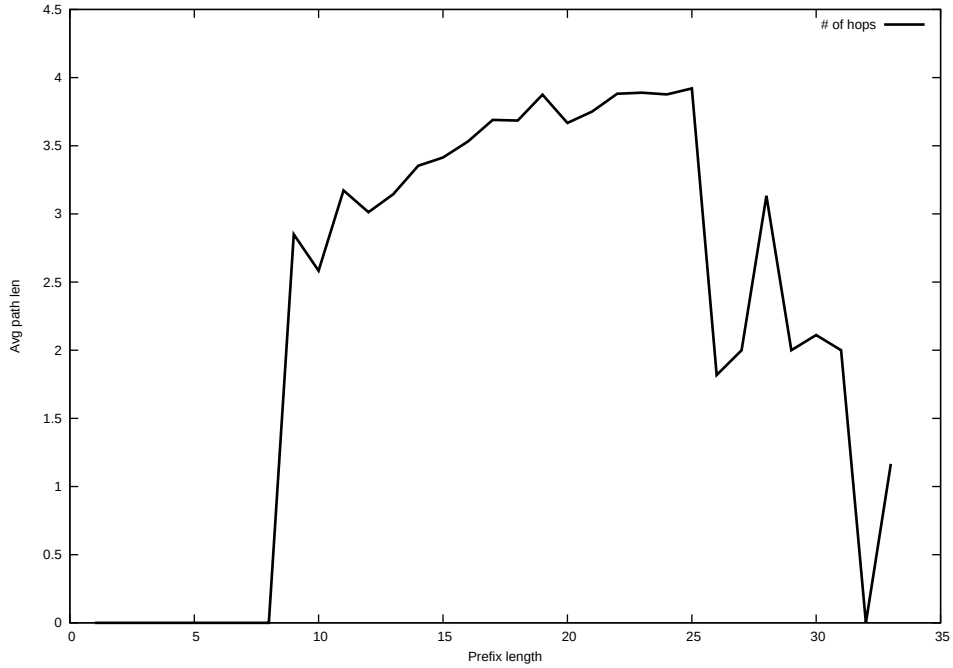
2012-05-22



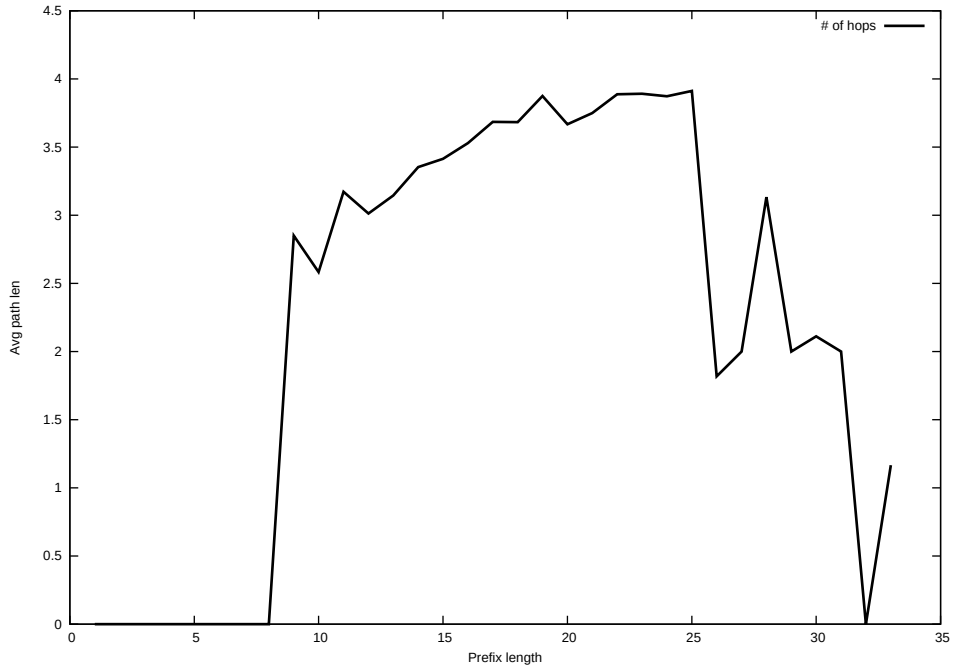
2012-05-23



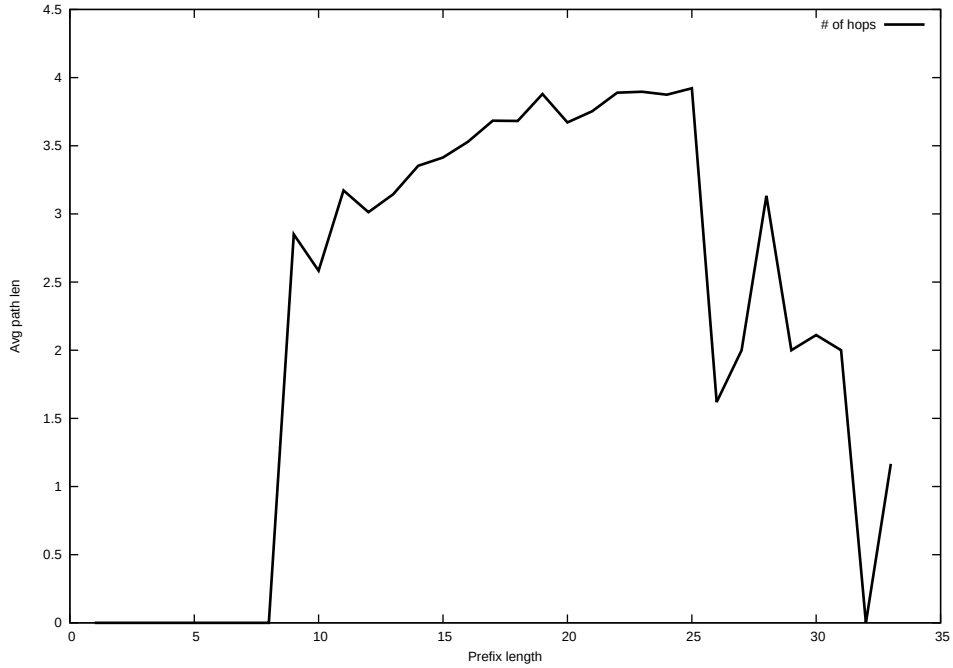
2012-05-24



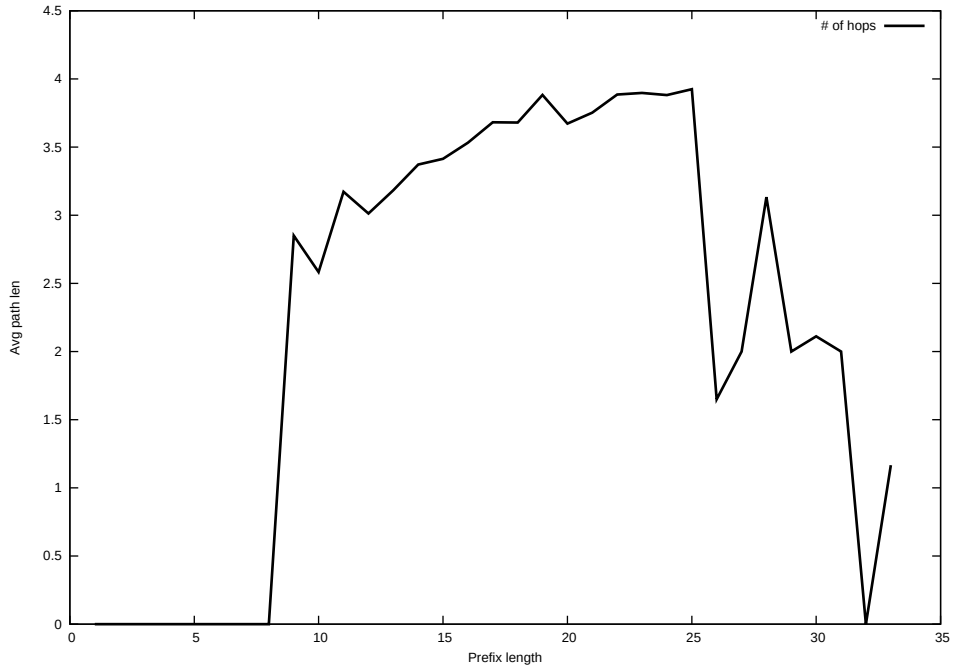
2012-05-25



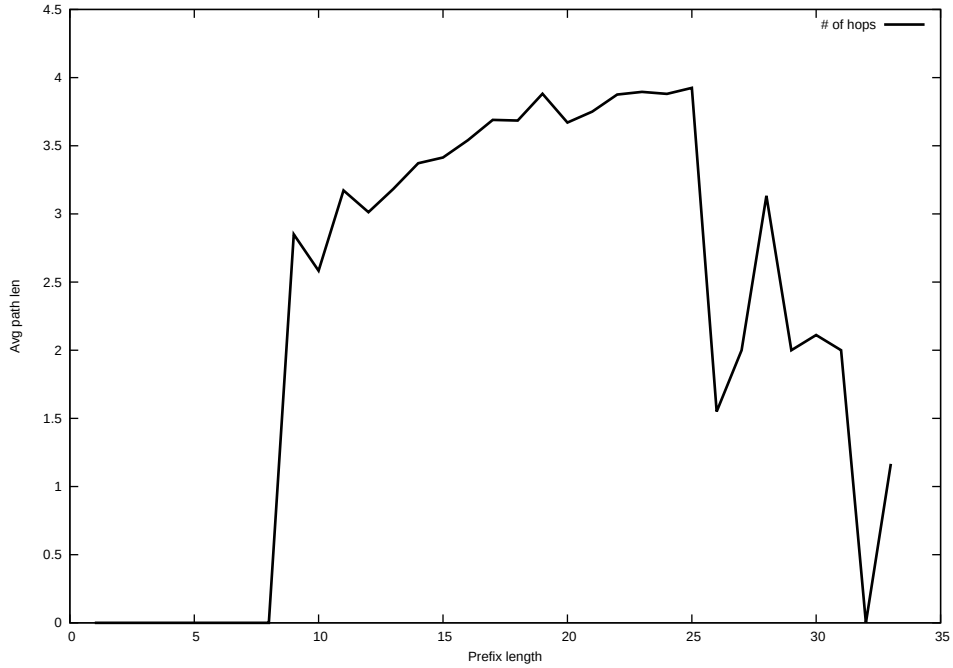
2012-05-26



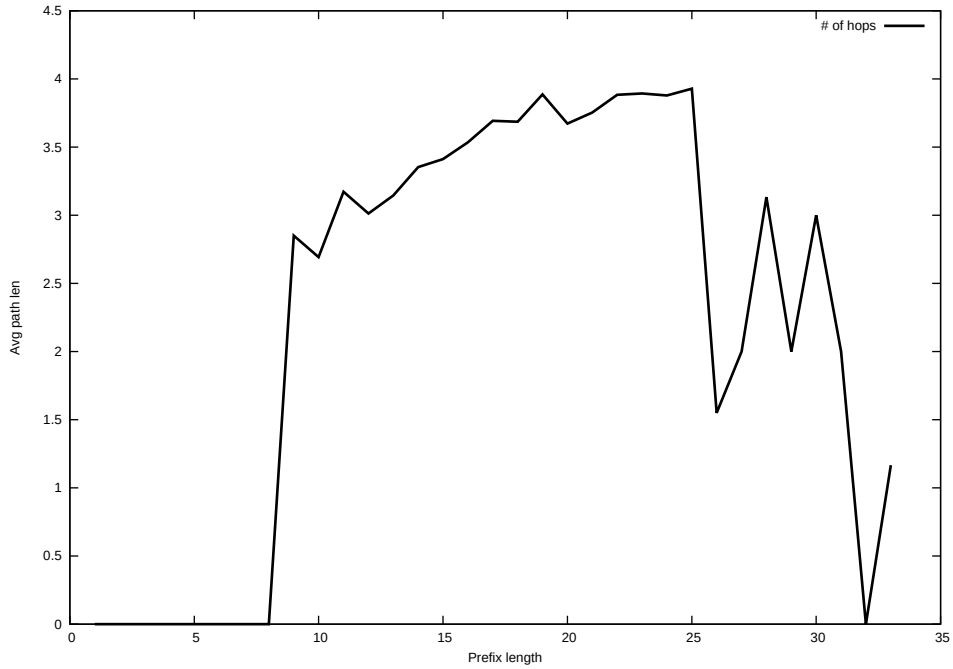
2012-05-27



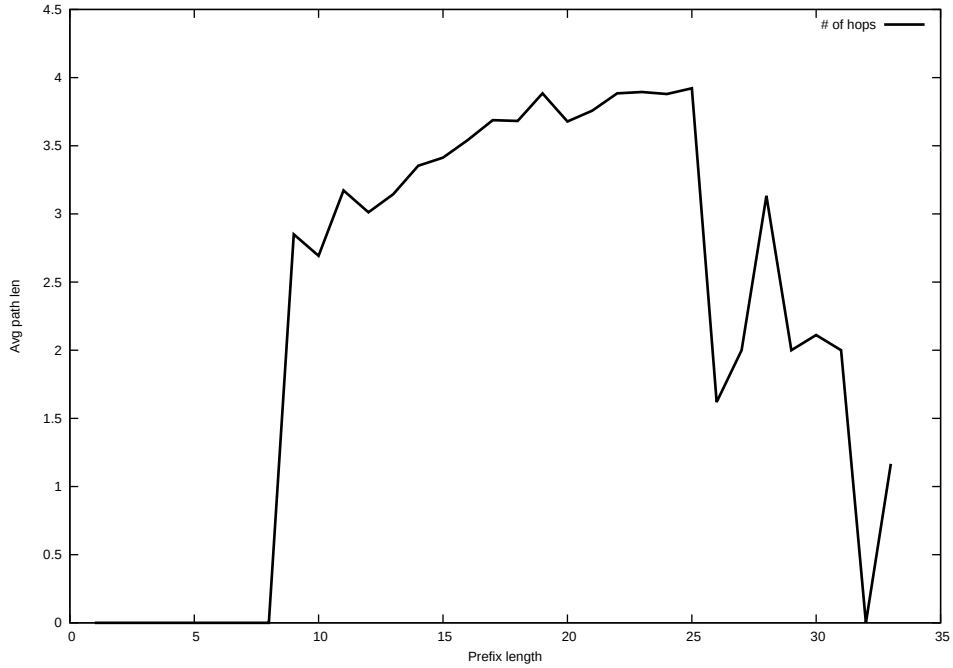
2012-05-28



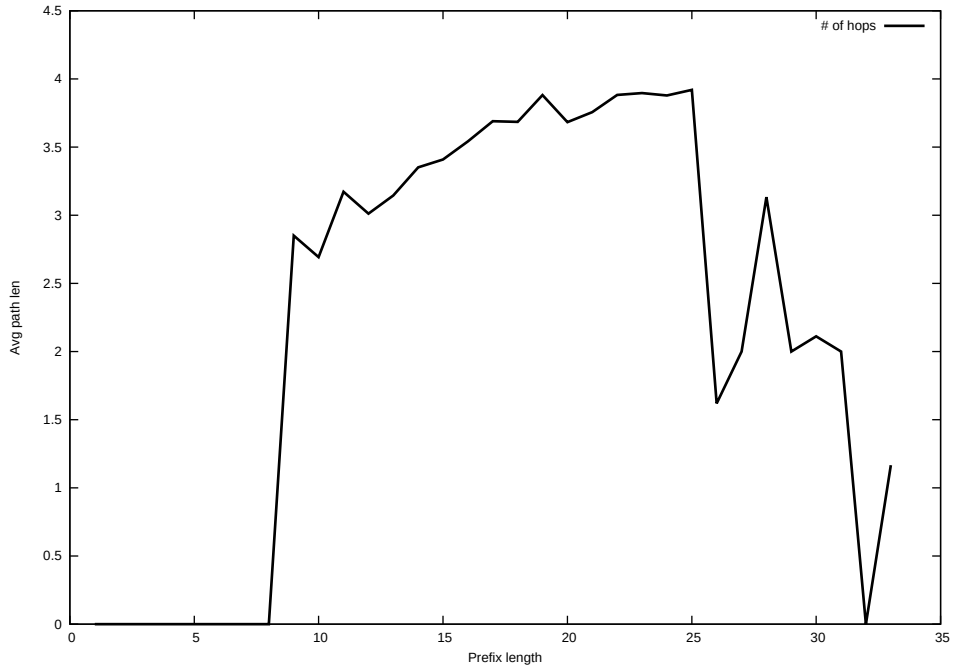
2012-05-29



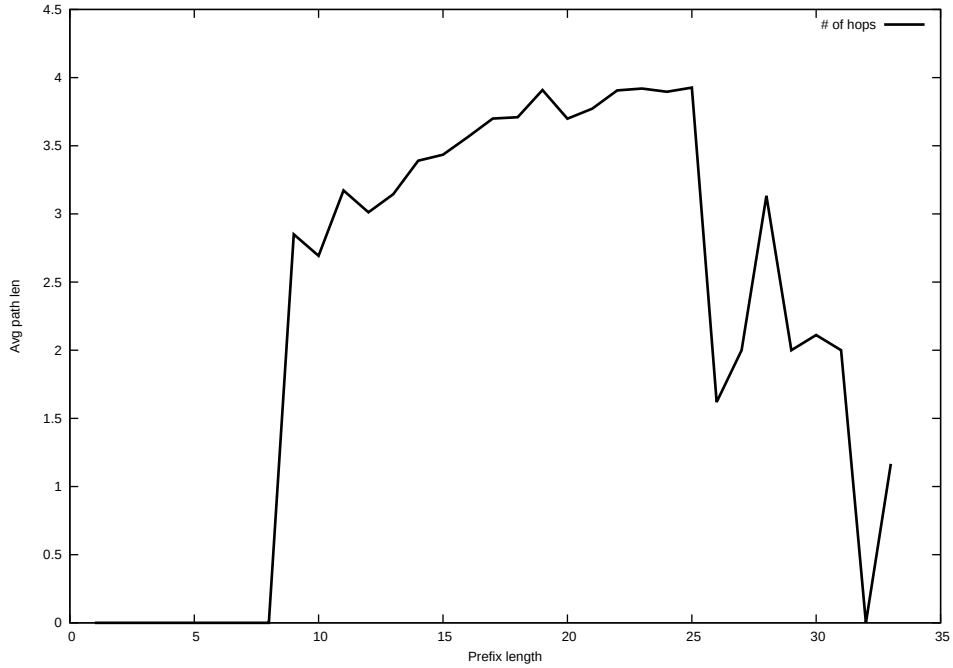
2012-05-30



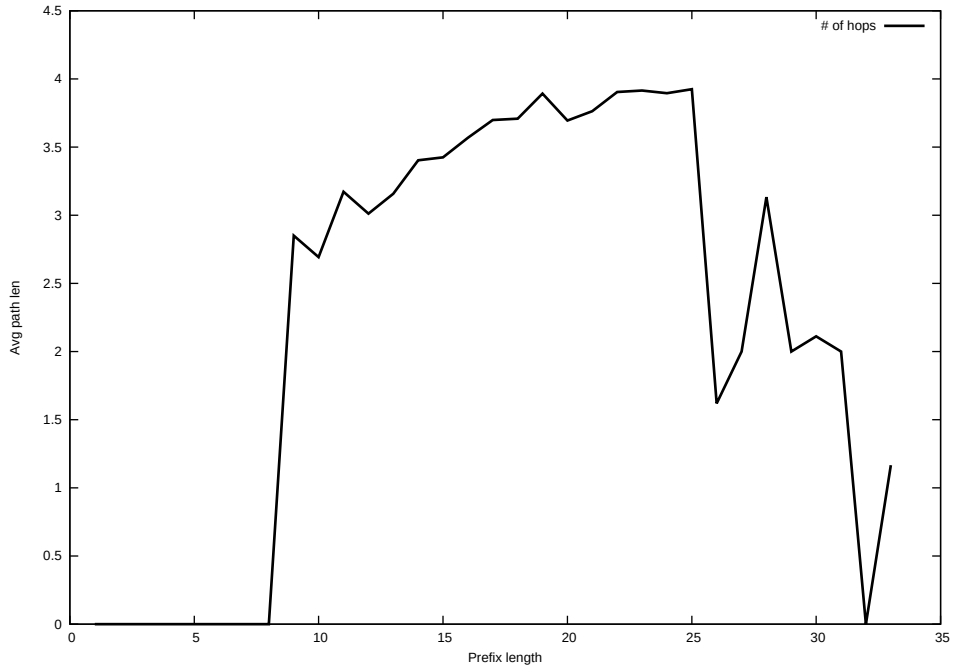
2012-05-31



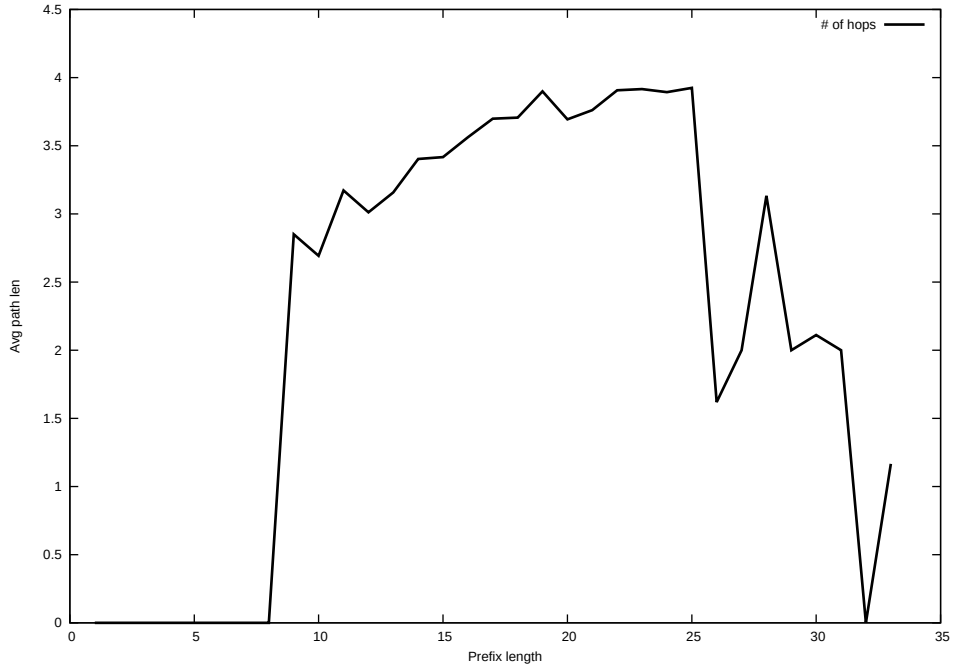
2012-06-01



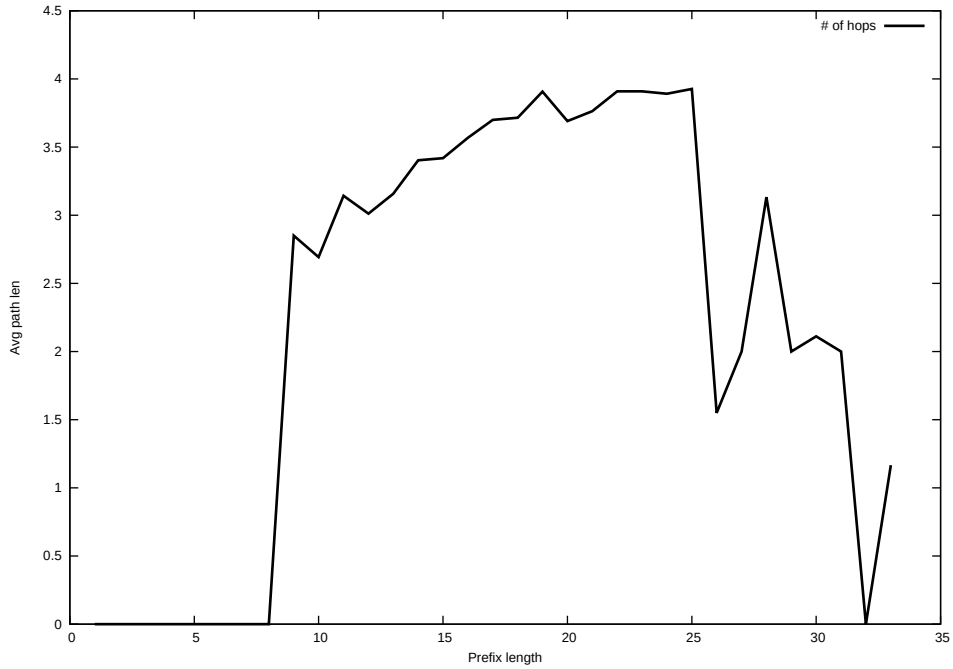
2012-06-02



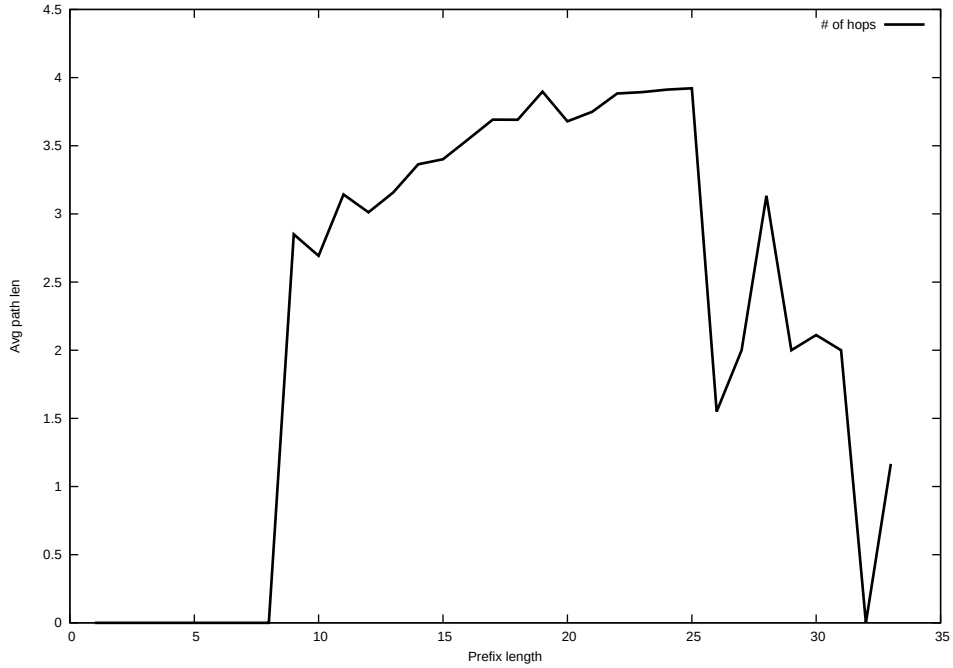
2012-06-03



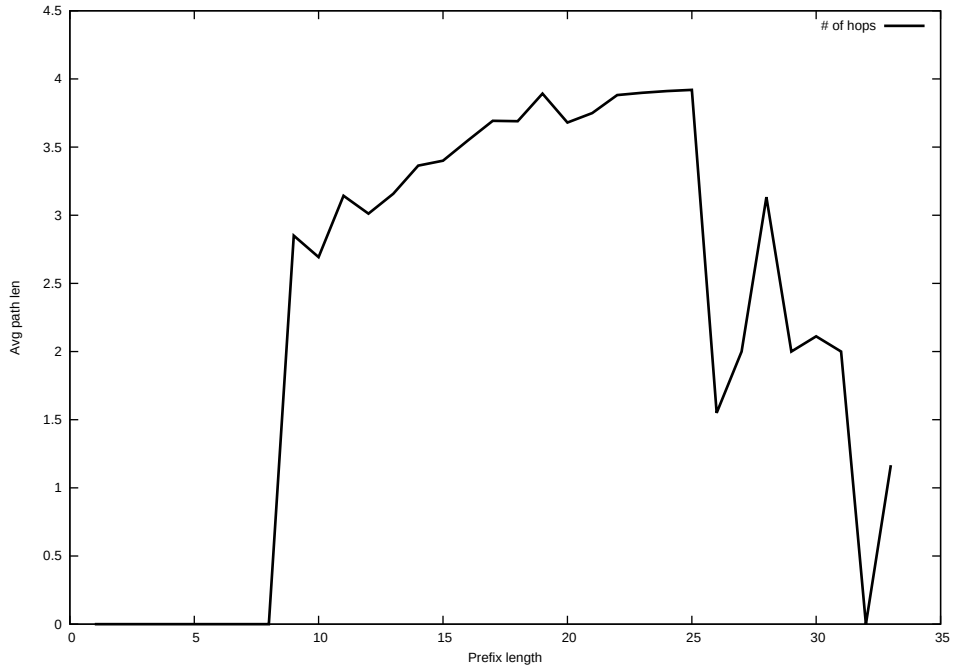
2012-06-04



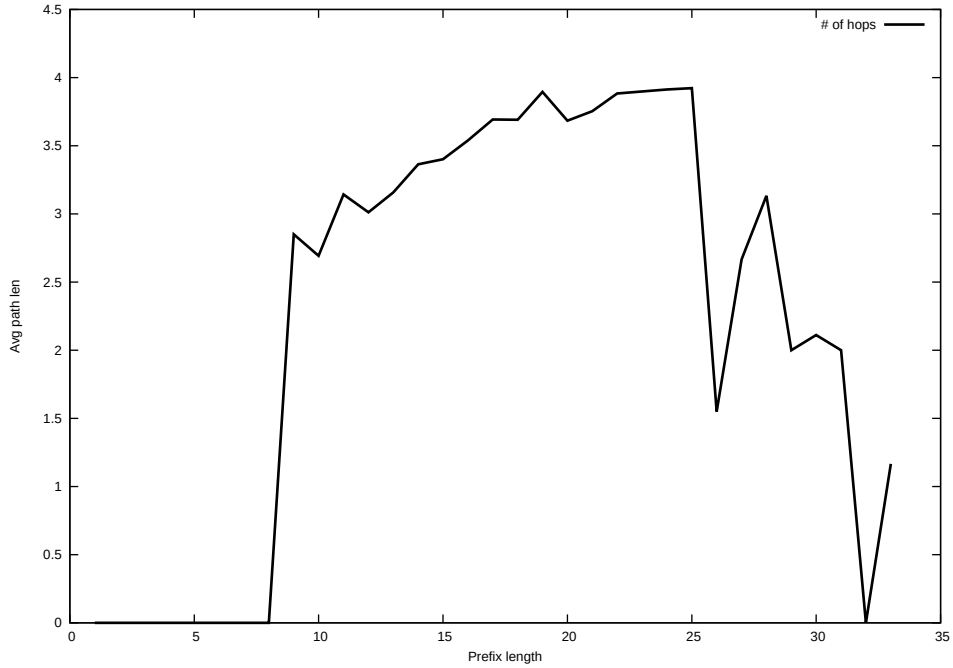
2012-06-05



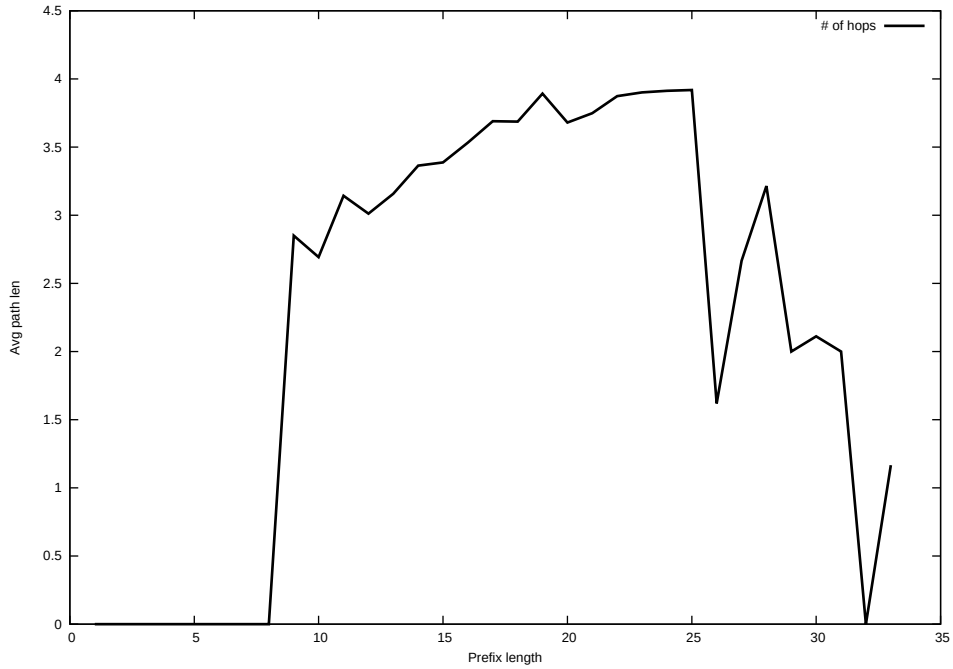
2012-06-06



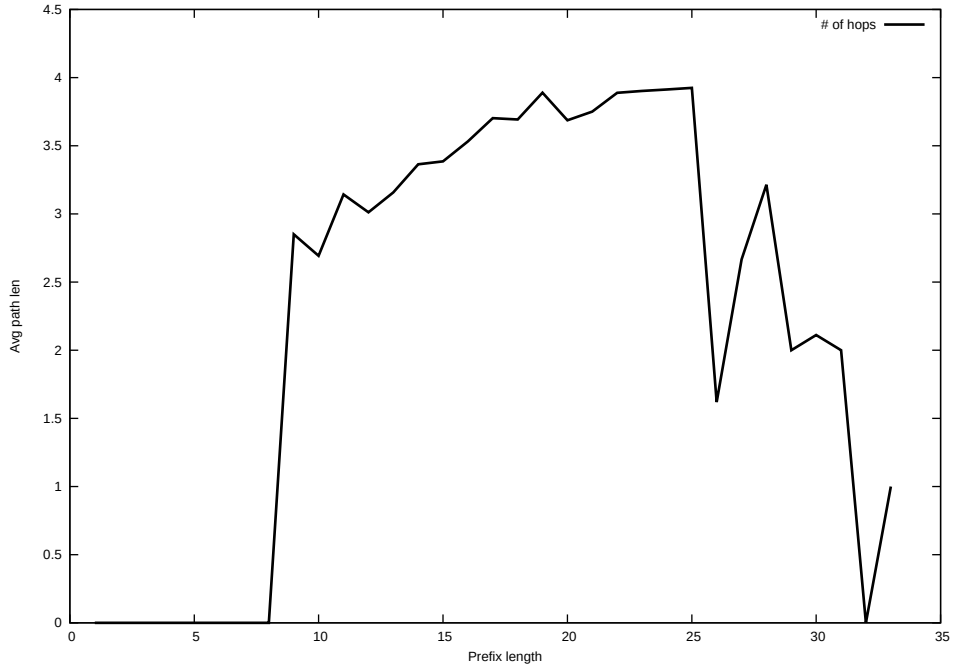
2012-06-07



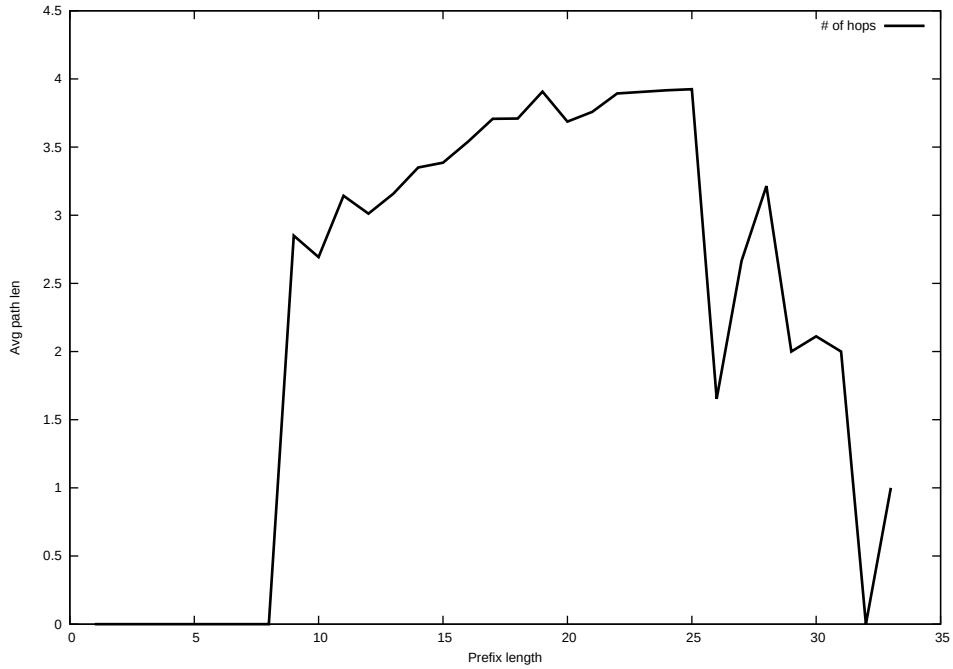
2012-06-08



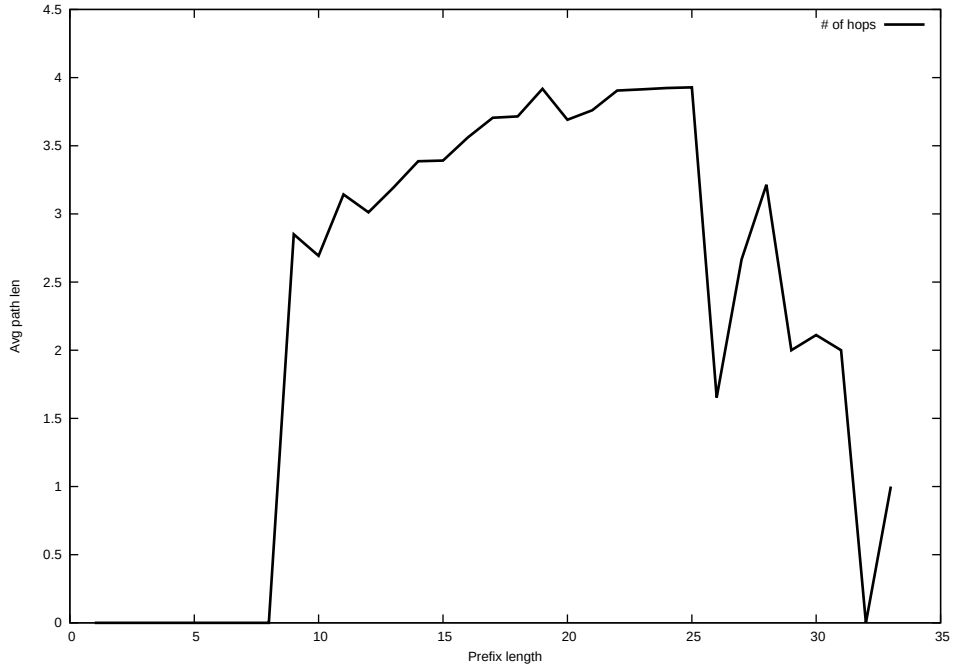
2012-06-10



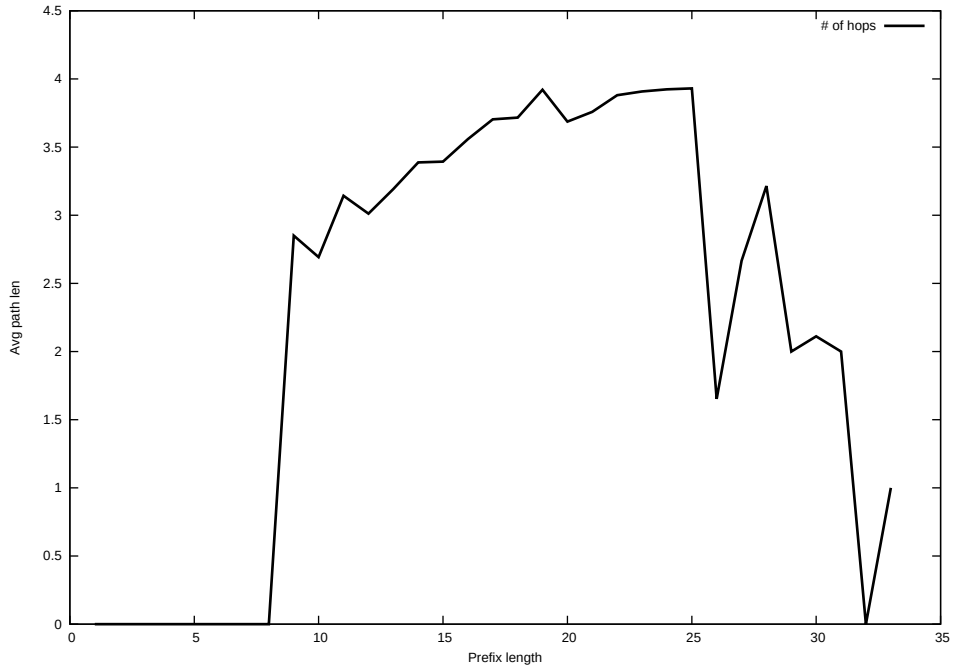
2012-06-11



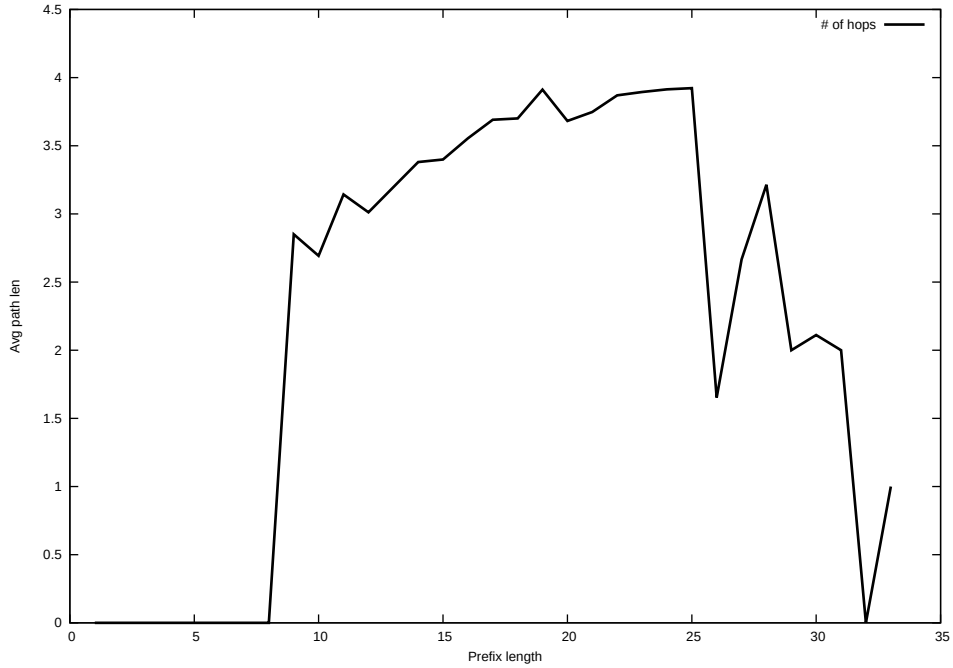
2012-06-12



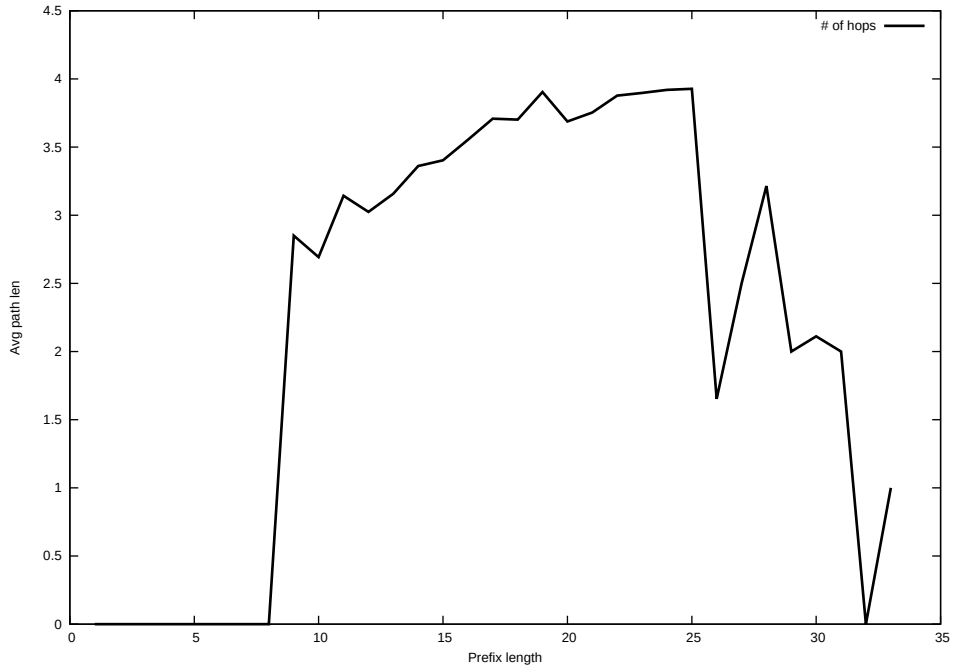
2012-06-13



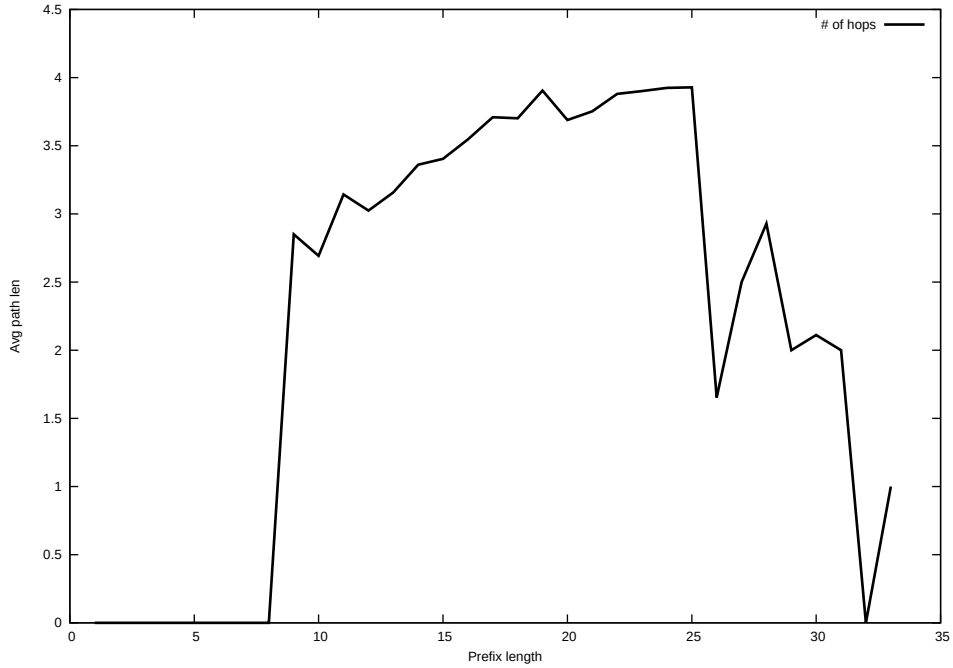
2012-06-14



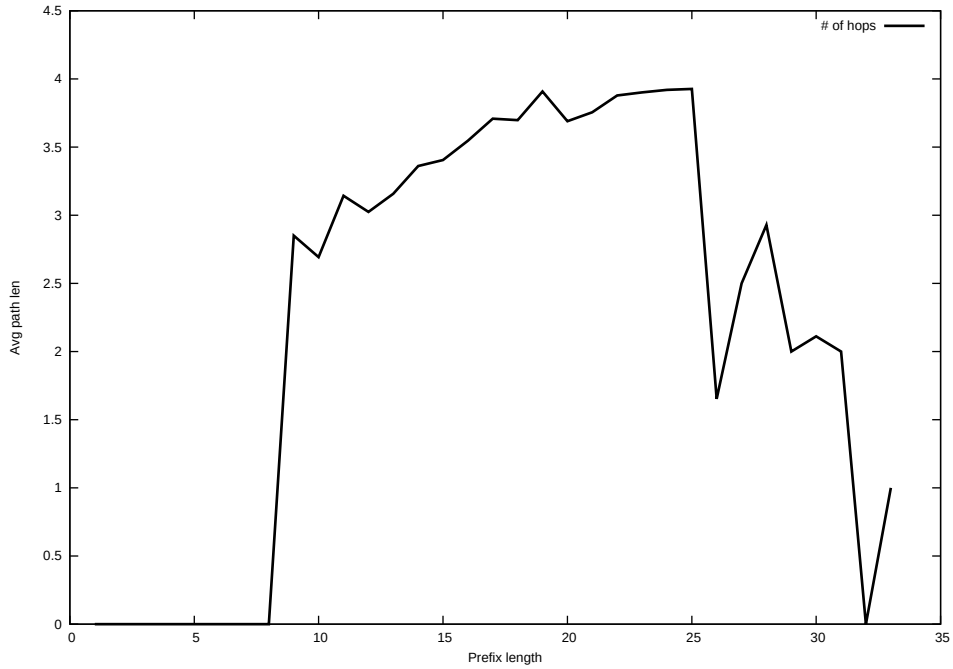
2012-06-15



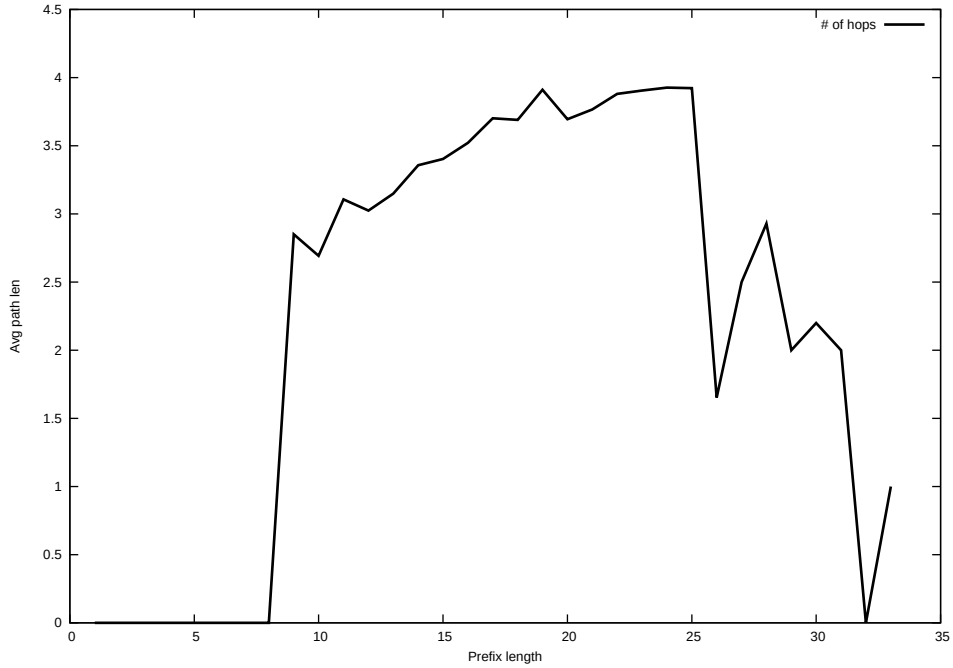
2012-06-16



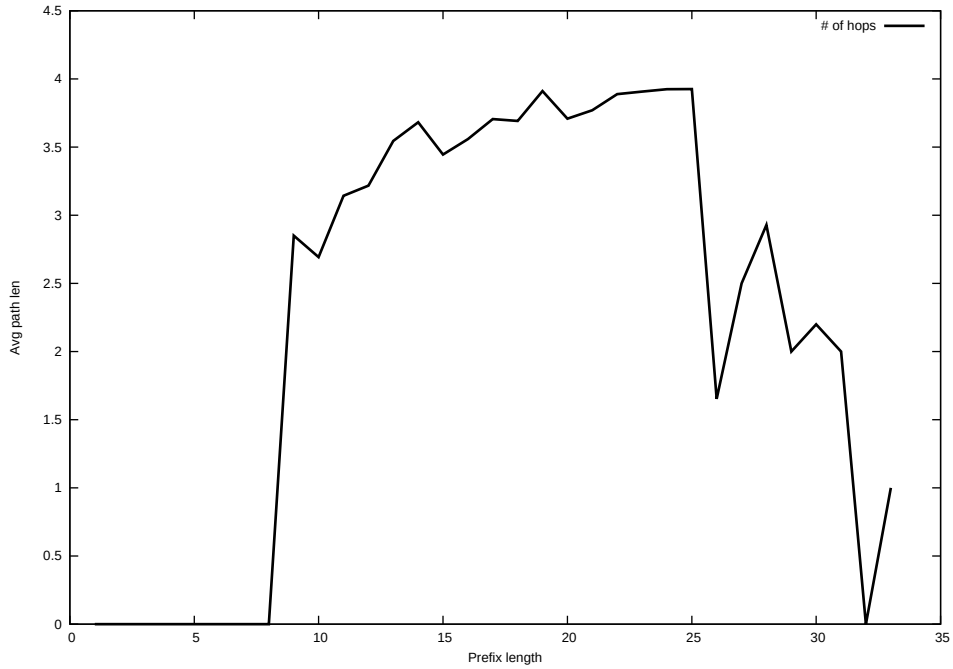
2012-06-17



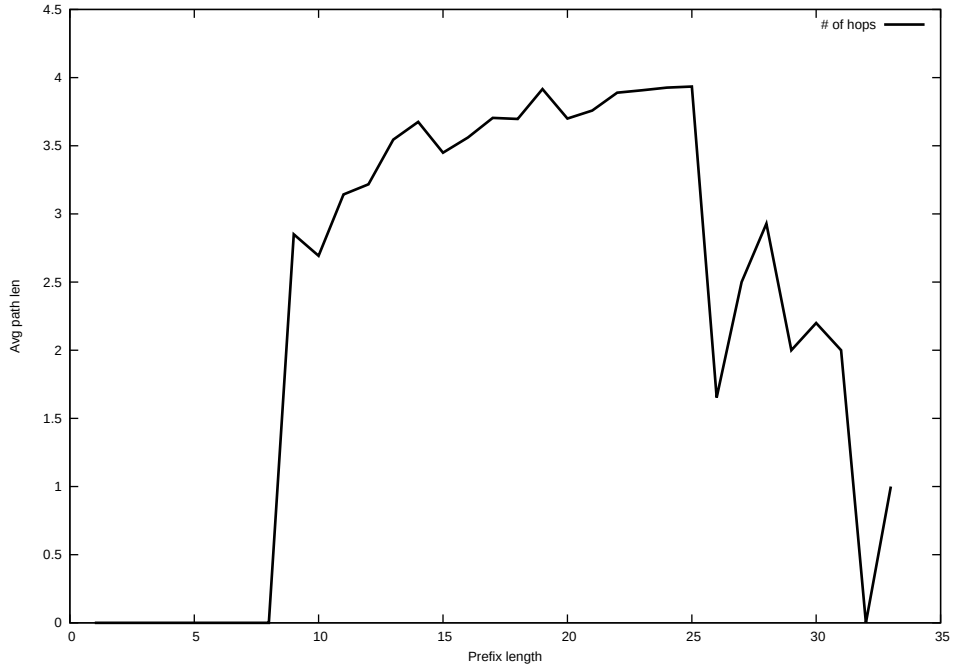
2012-06-18



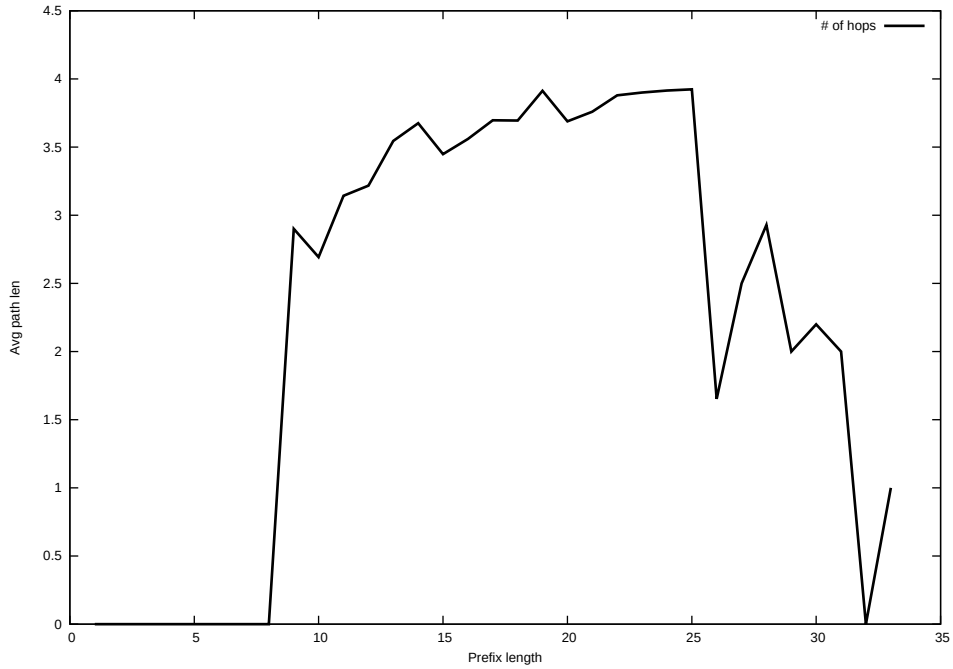
2012-06-19



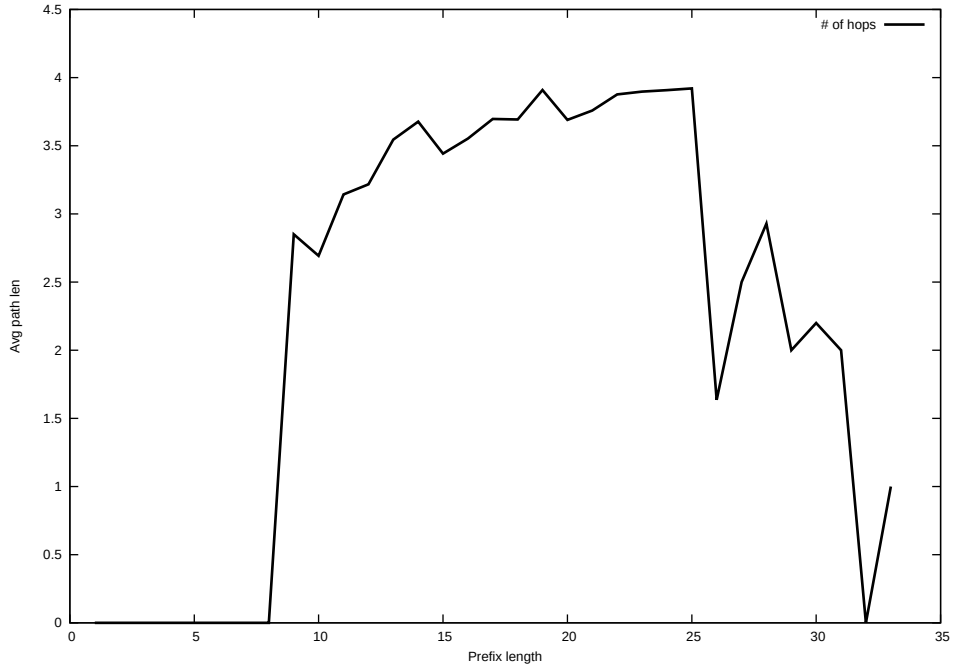
2012-06-20



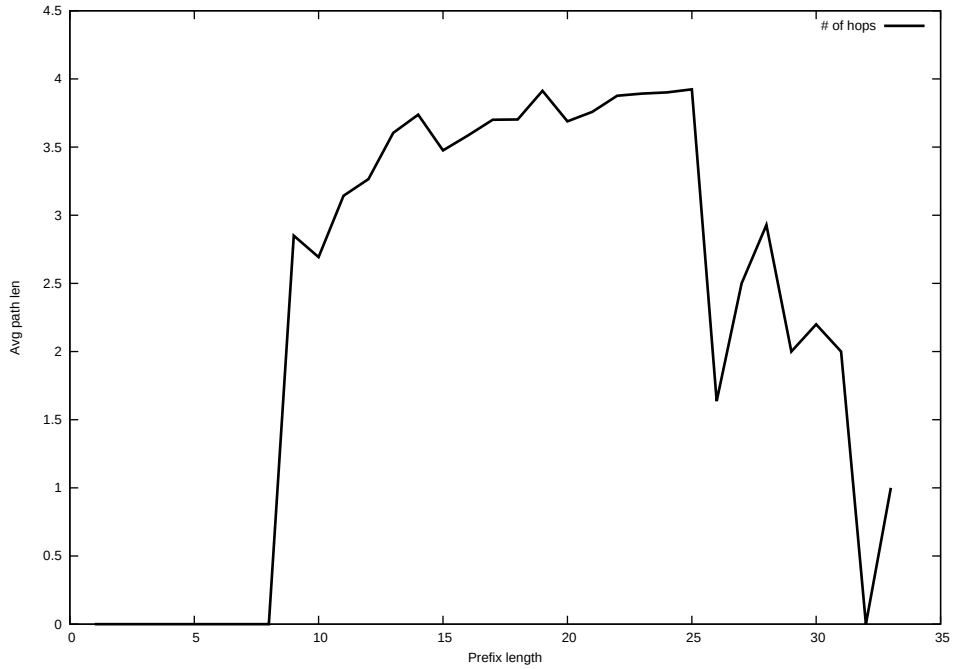
2012-06-21



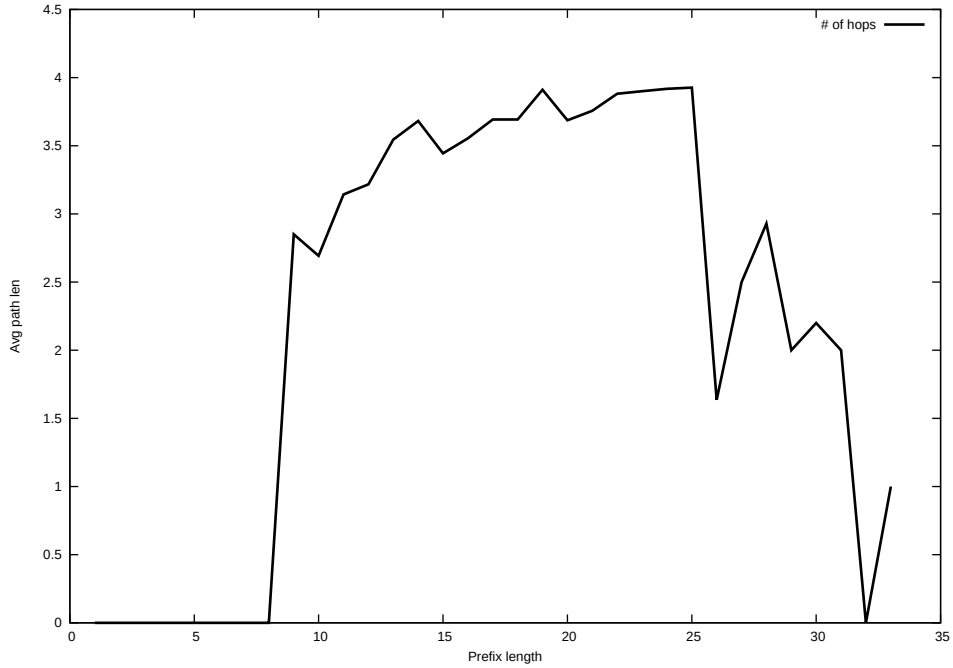
2012-06-22



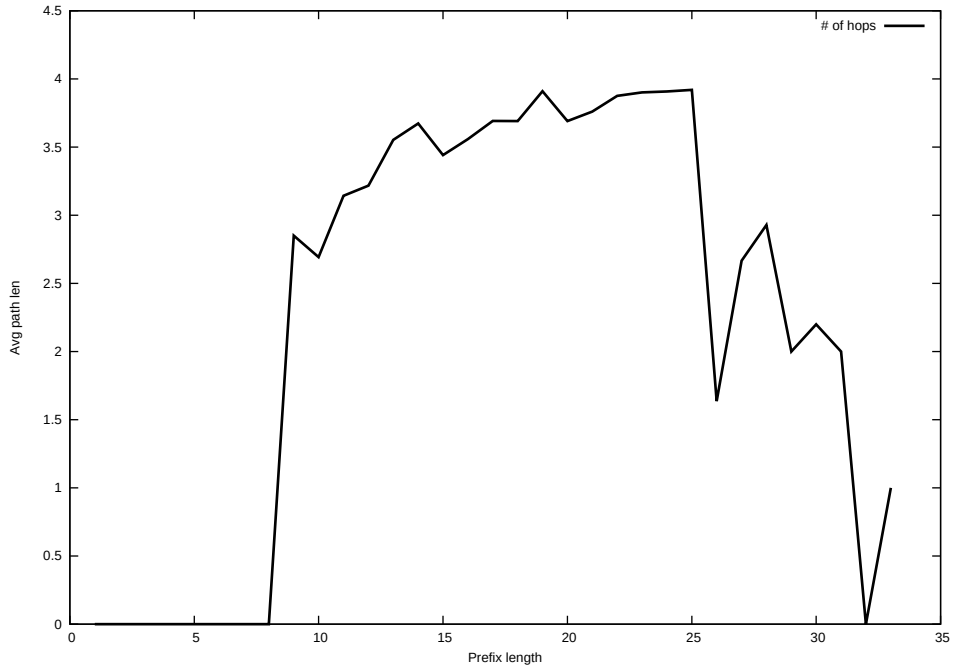
2012-06-23



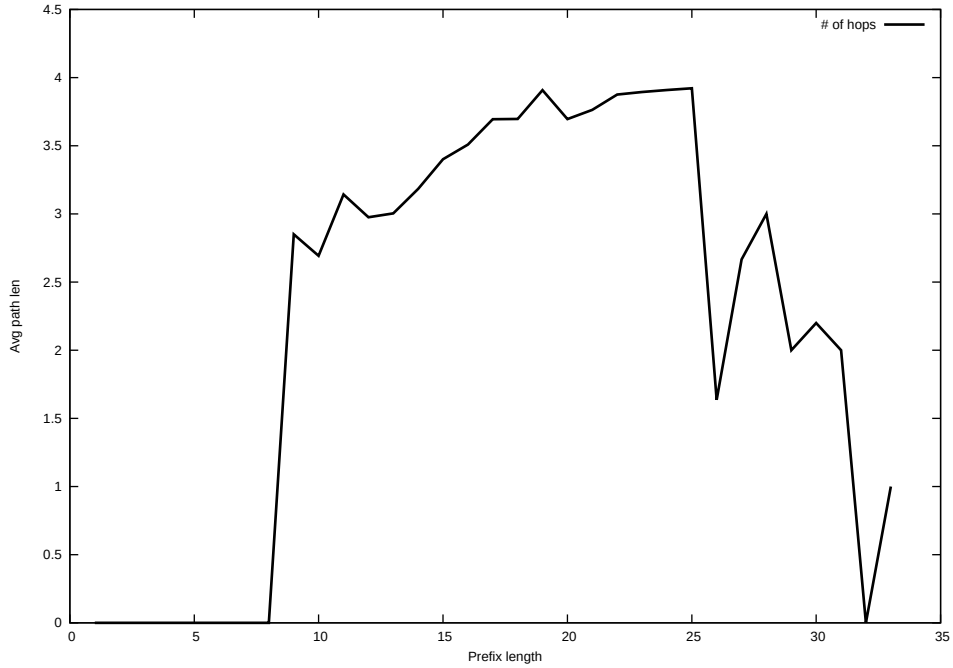
2012-06-24



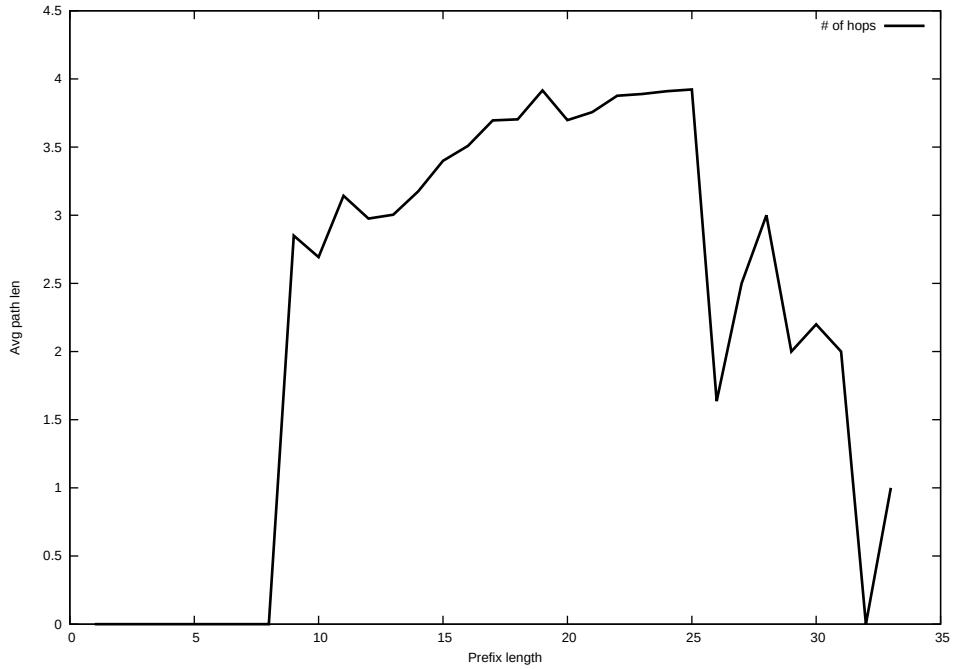
2012-06-25



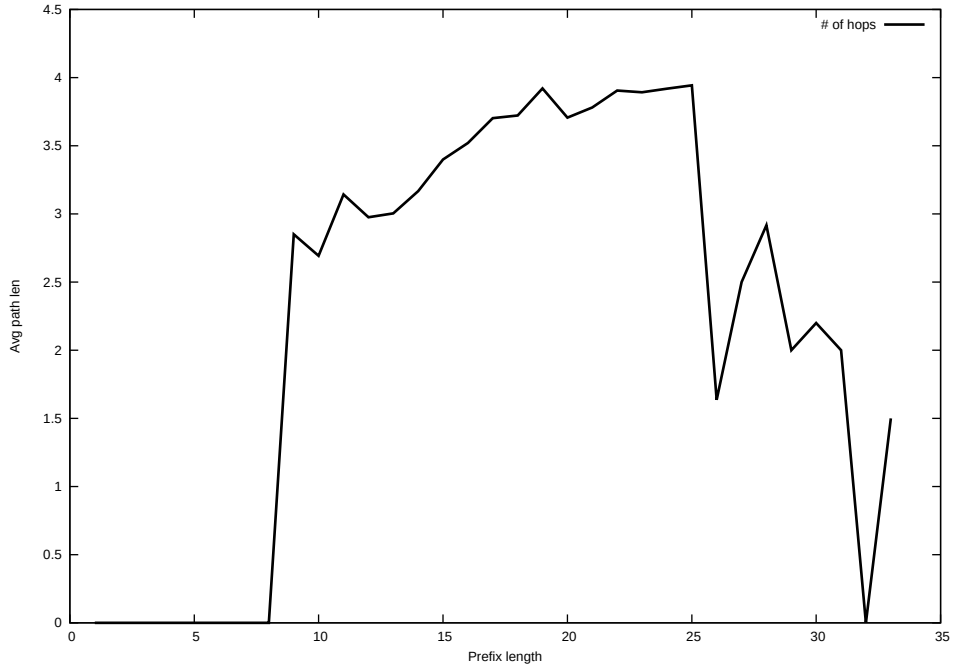
2012-06-26



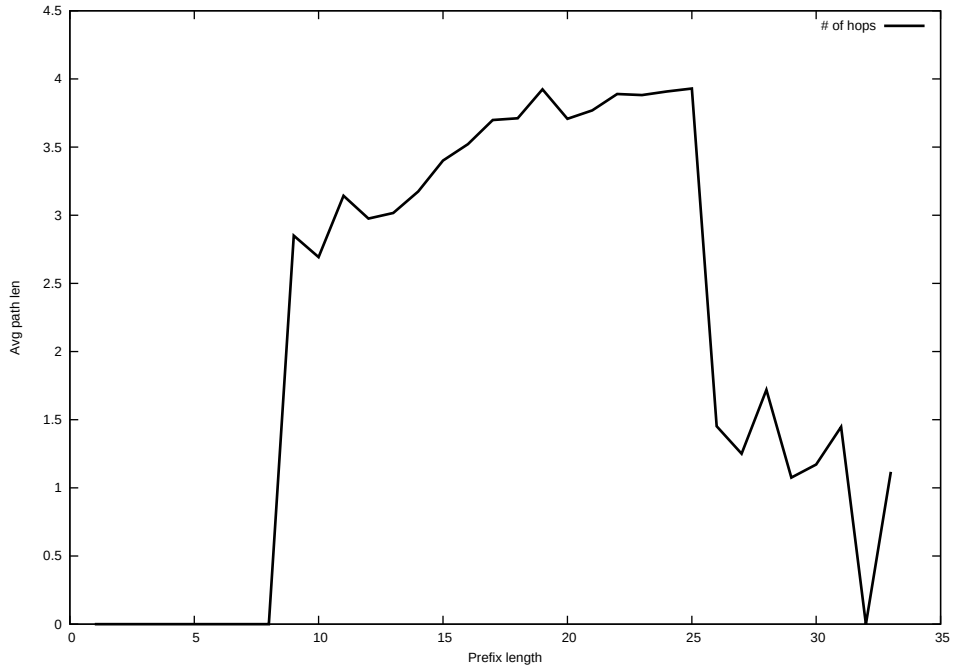
2012-06-27



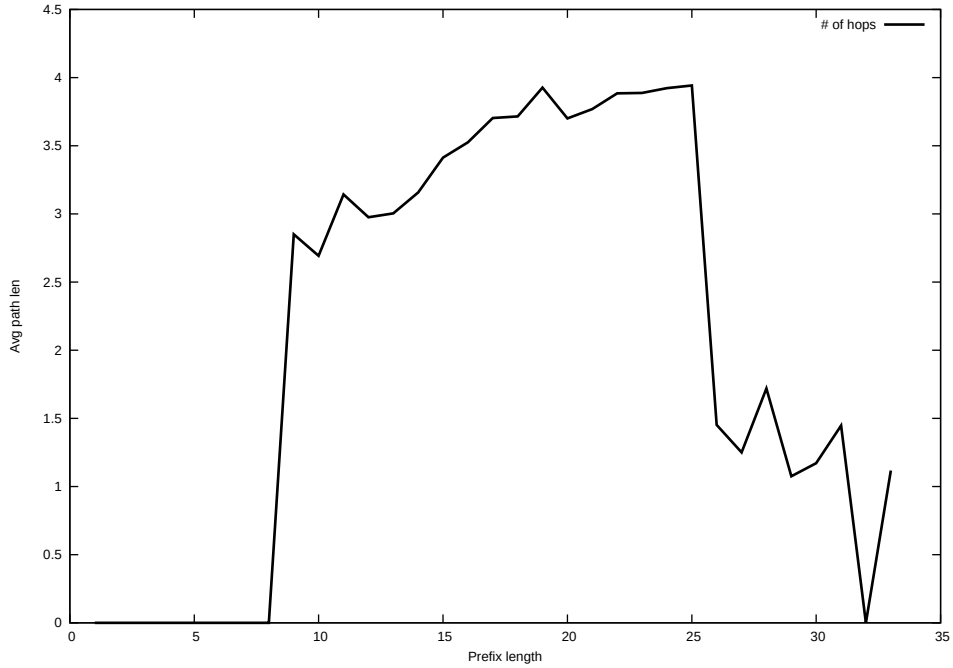
2012-06-28



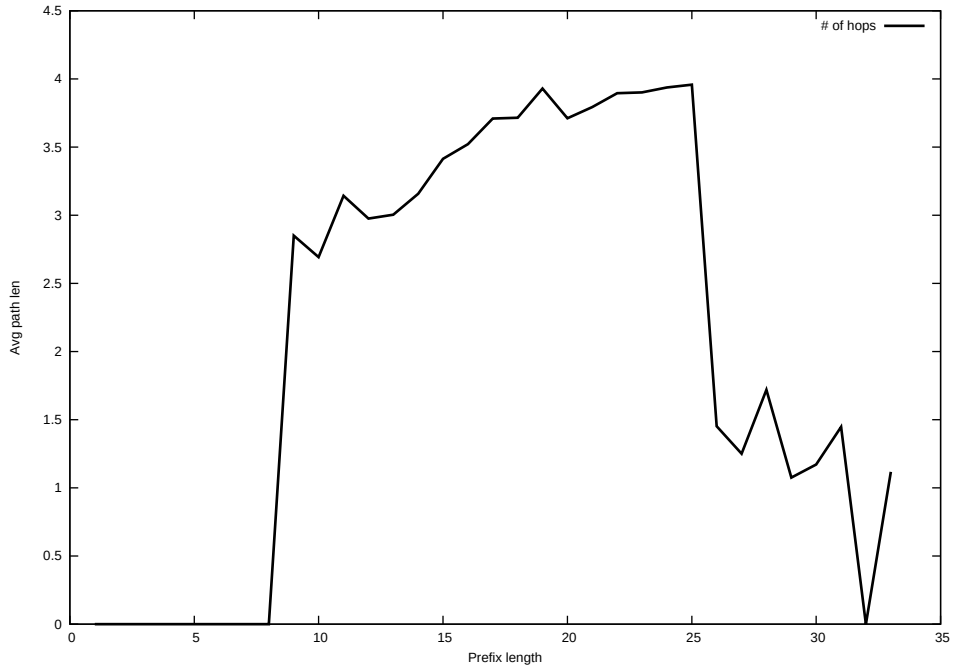
2012-06-29



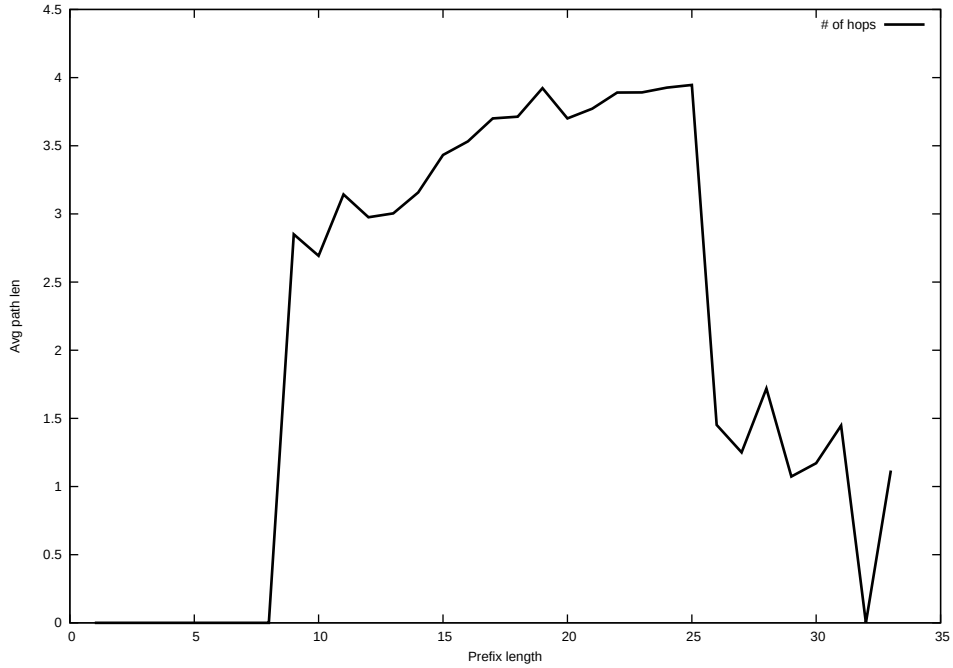
2012-06-30



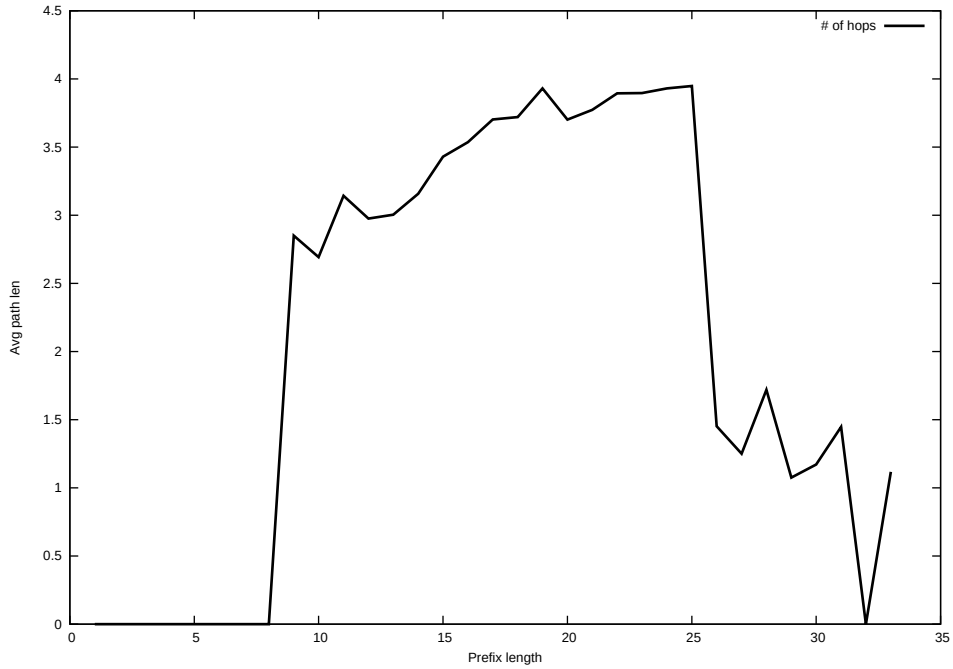
2012-07-01



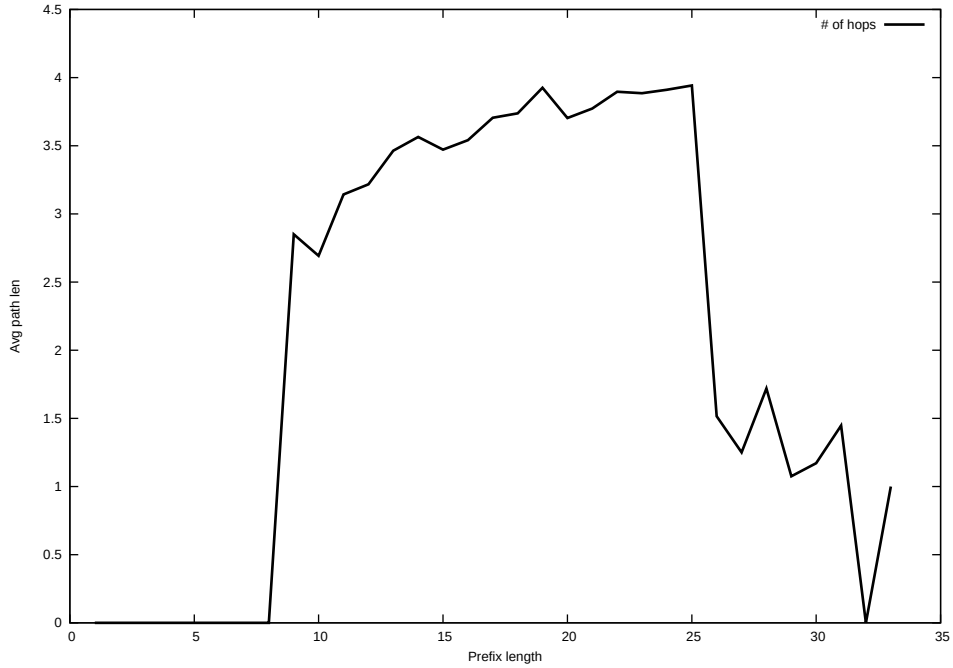
2012-07-02



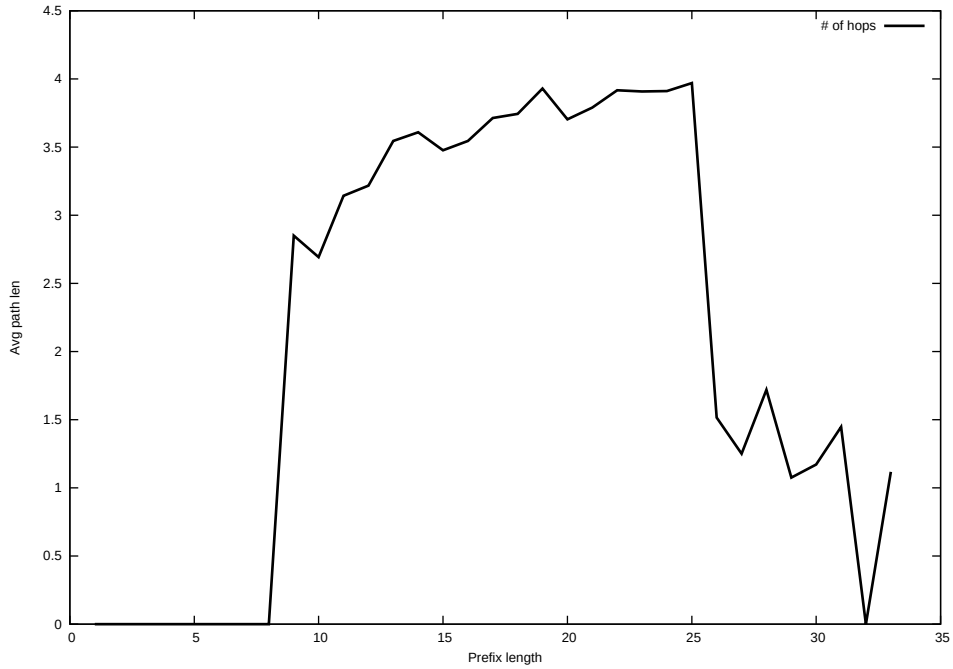
2012-07-03



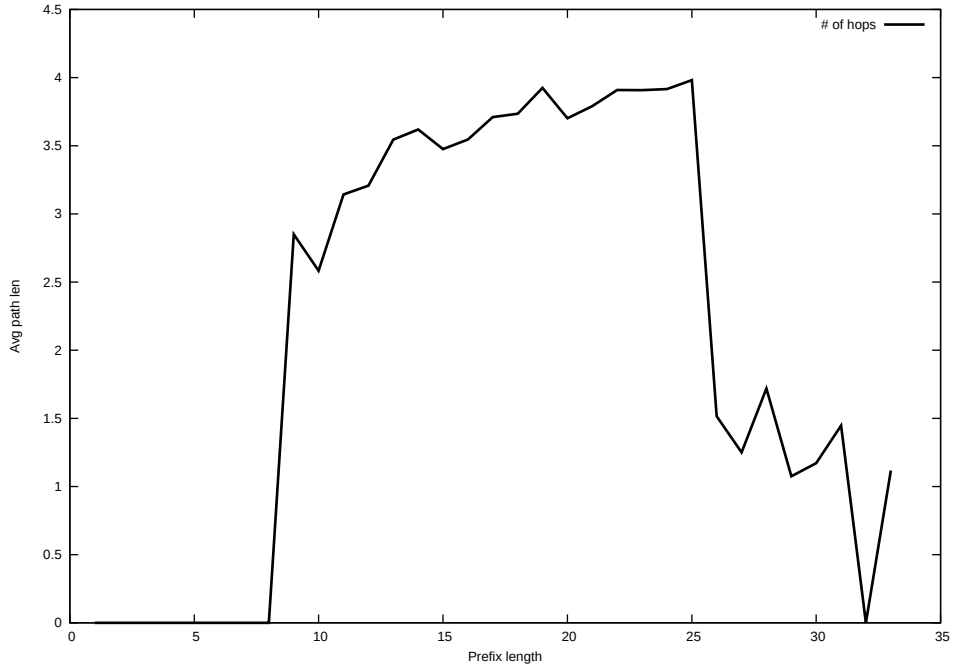
2012-07-04



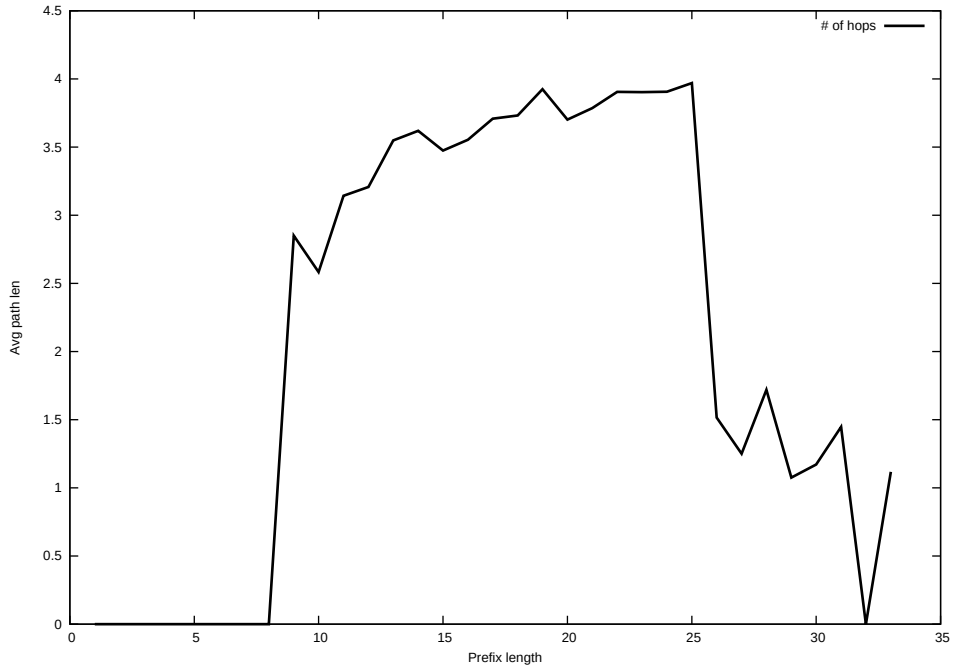
2012-07-05



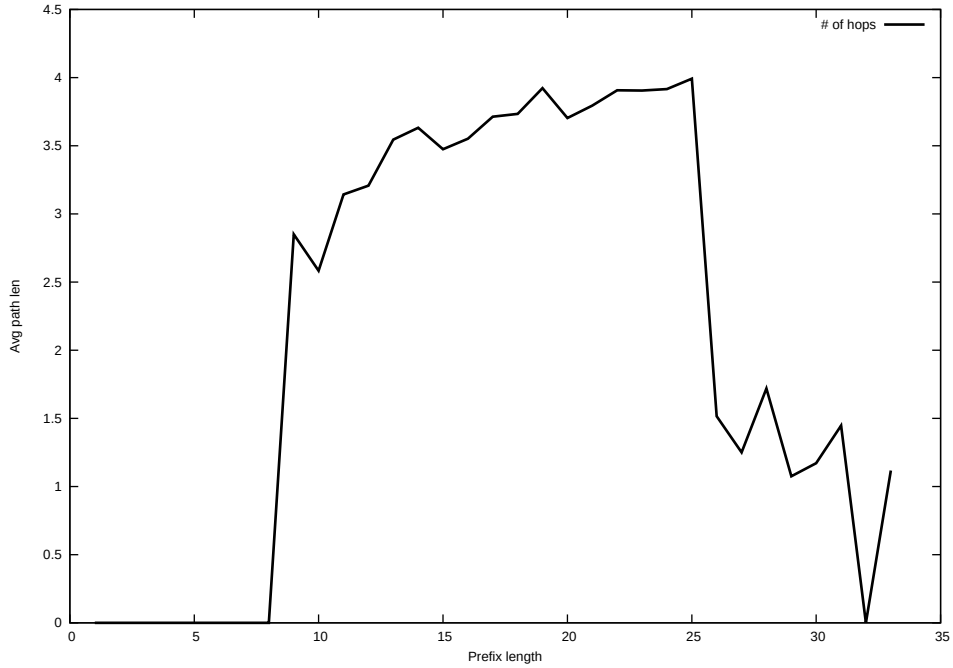
2012-07-06



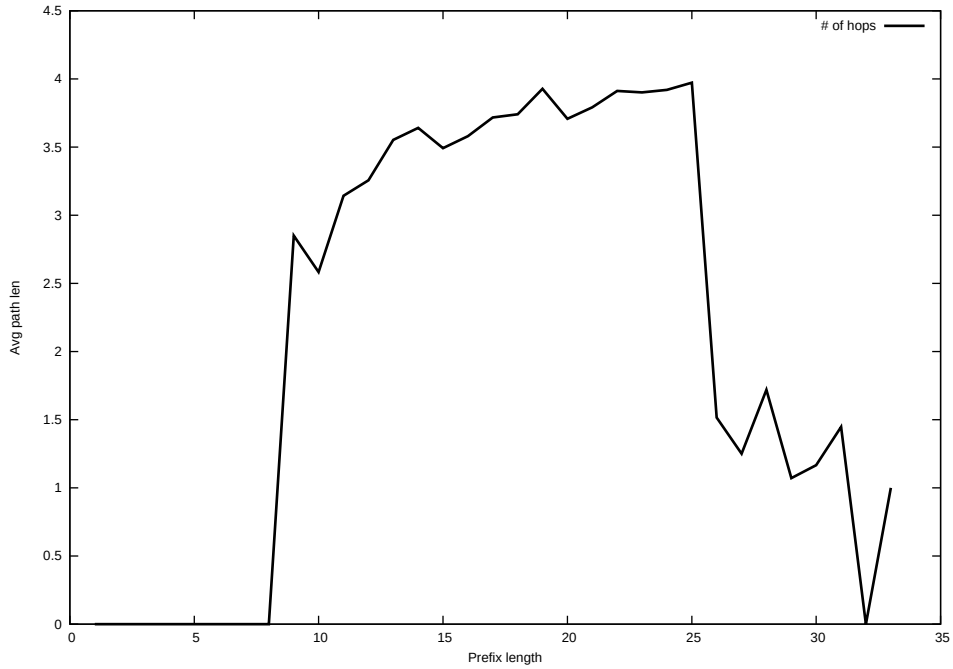
2012-07-07



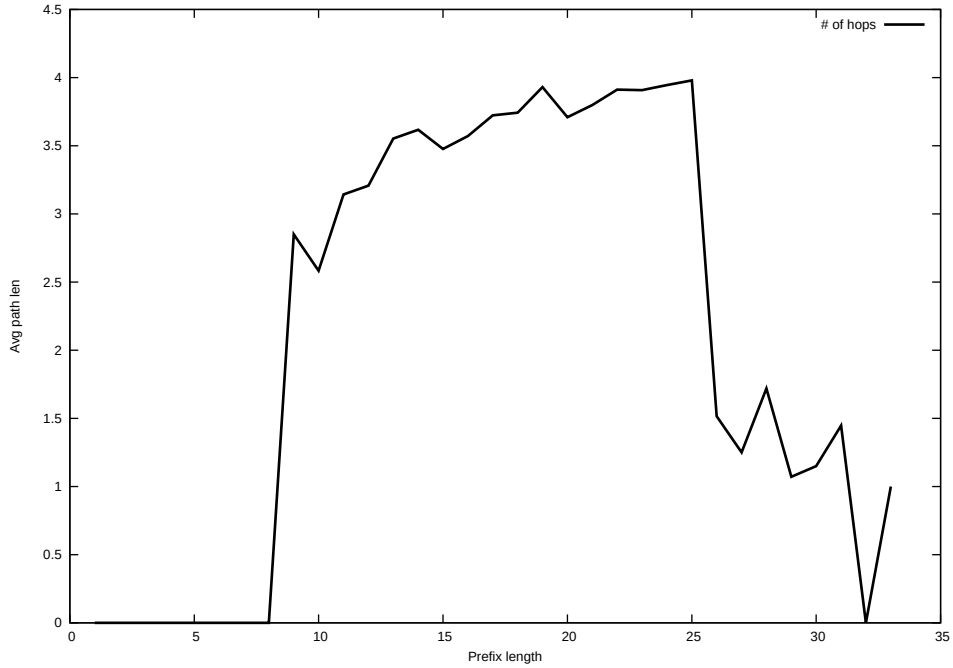
2012-07-08



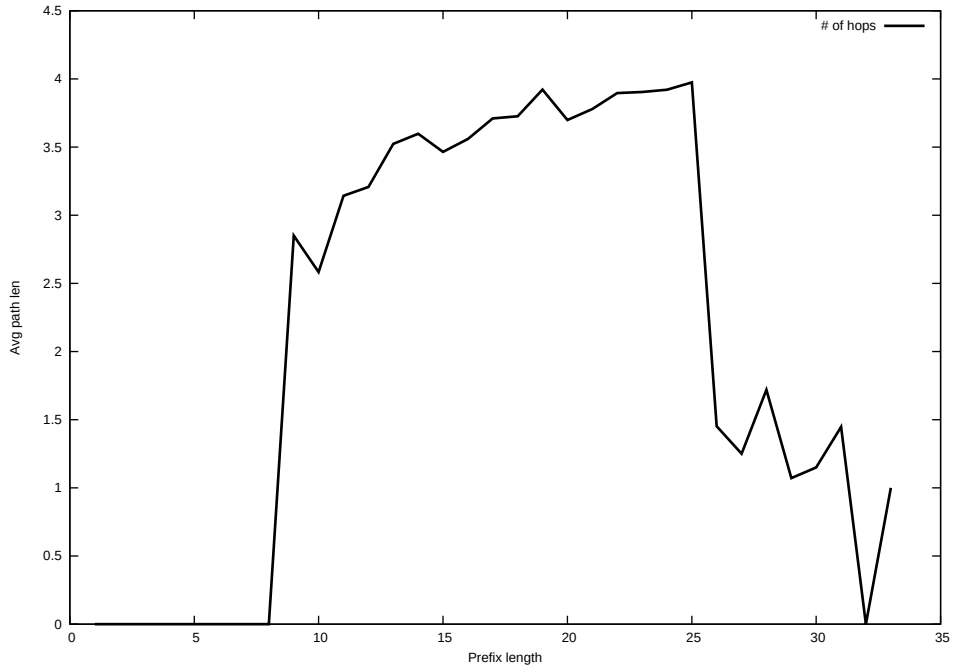
2012-07-09



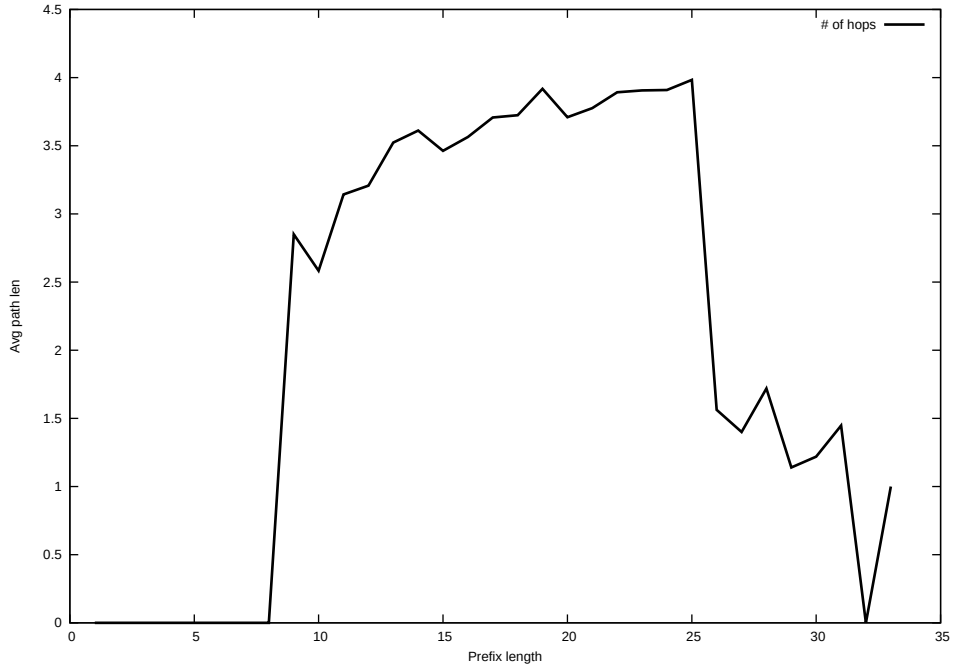
2012-07-10



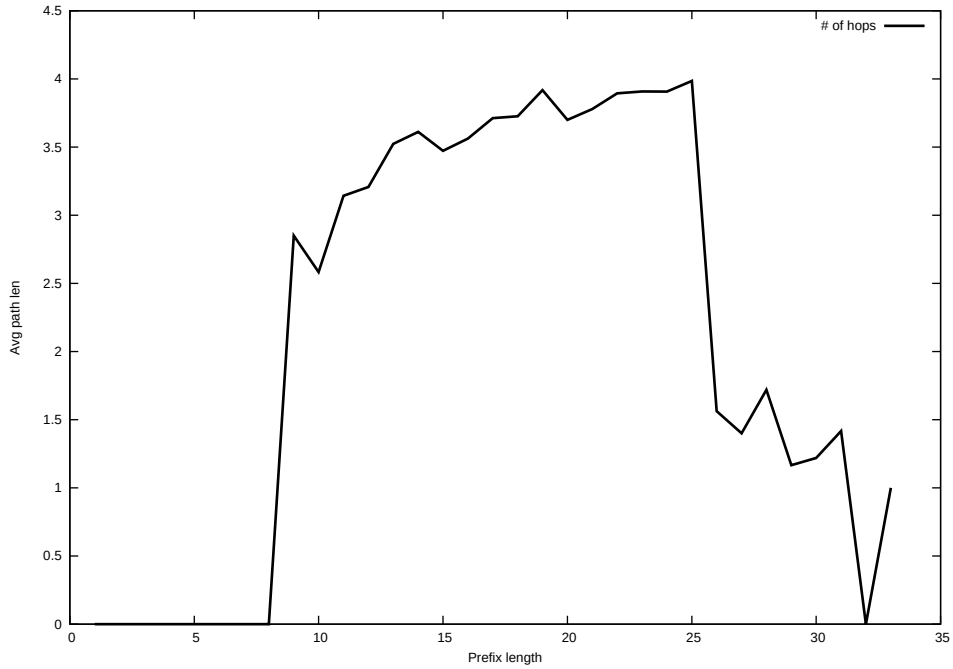
2012-07-11



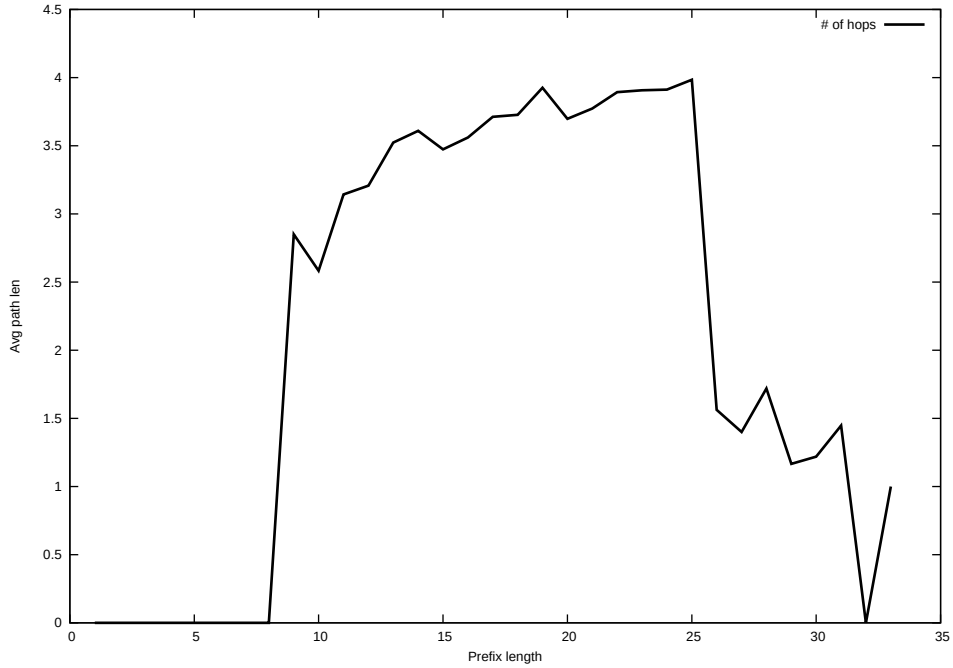
2012-07-12



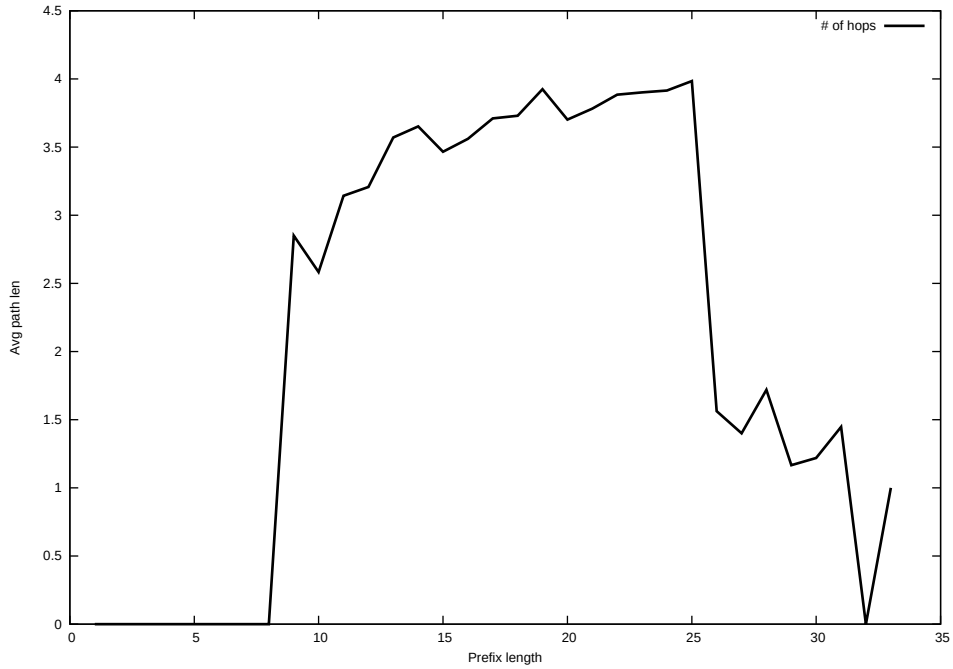
2012-07-13



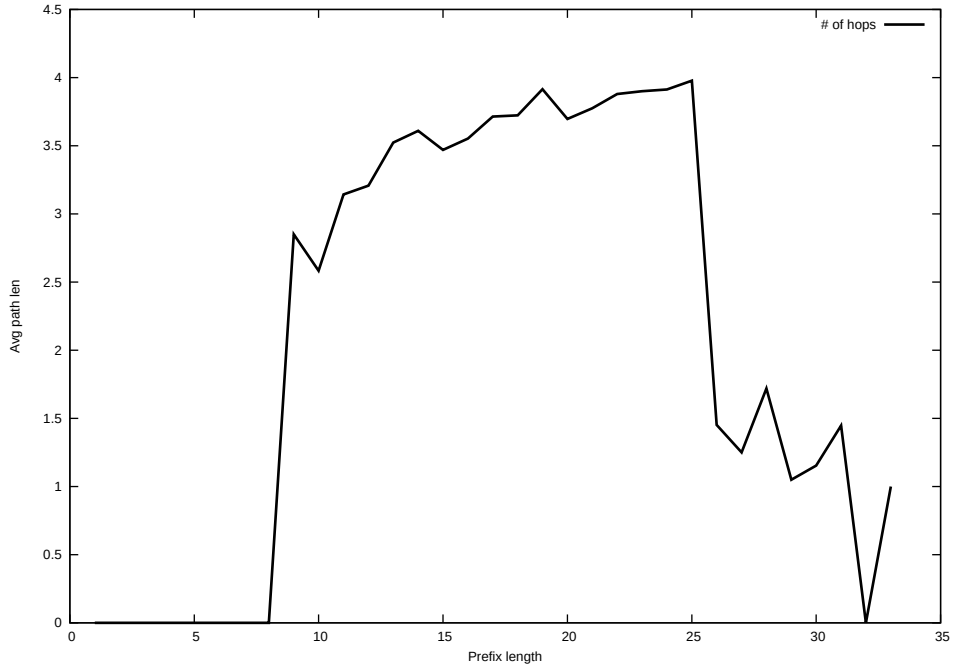
2012-07-14



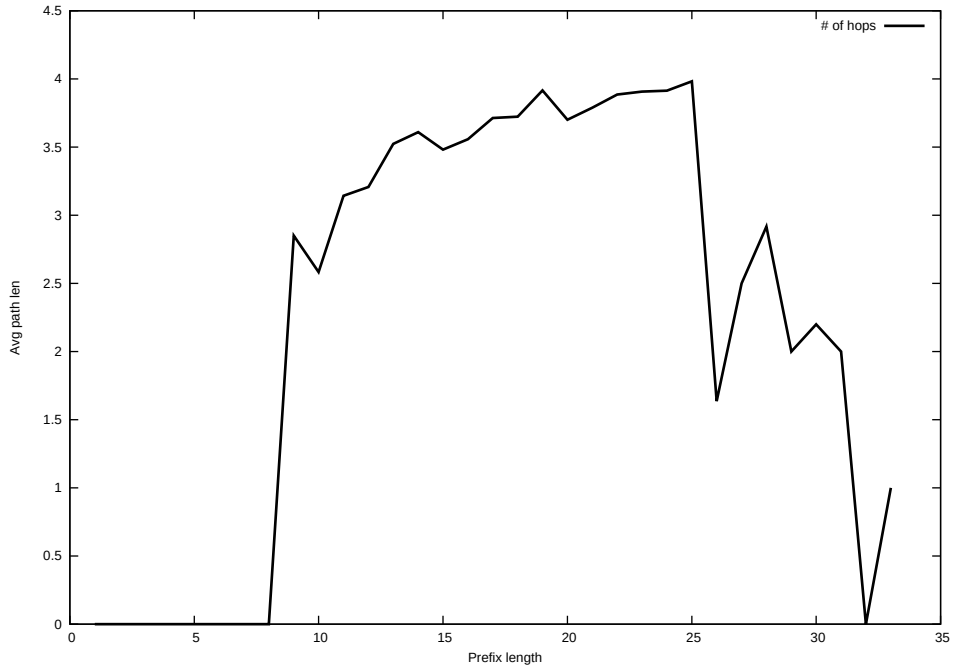
2012-07-15



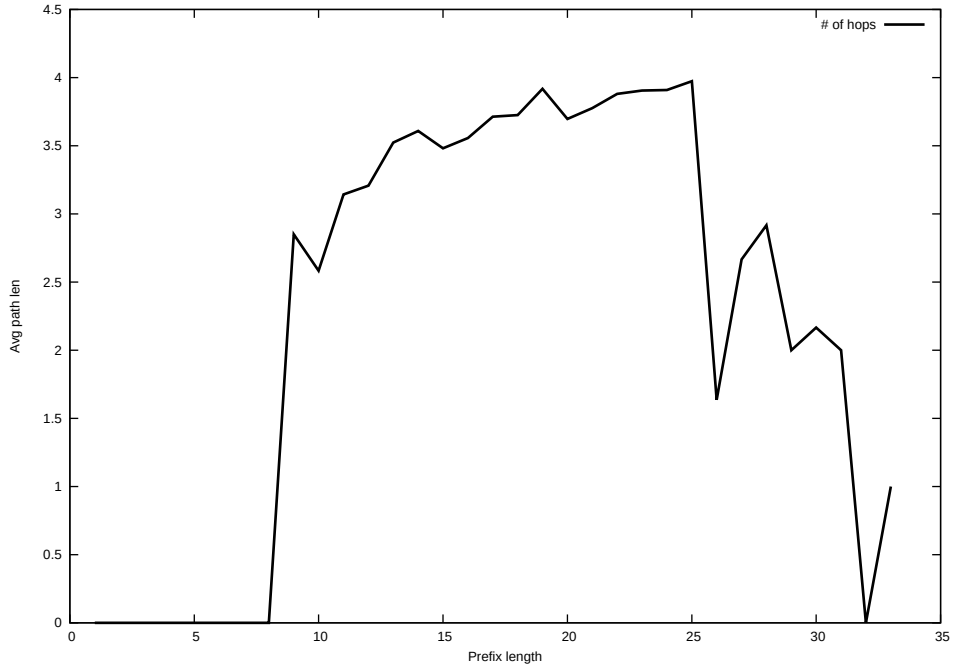
2012-07-16



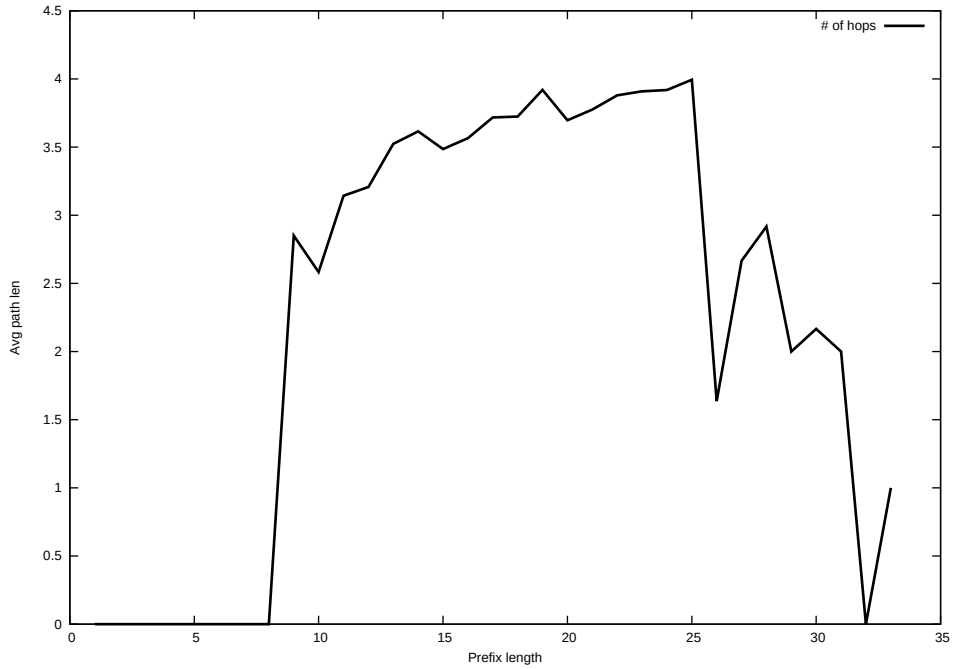
2012-07-17



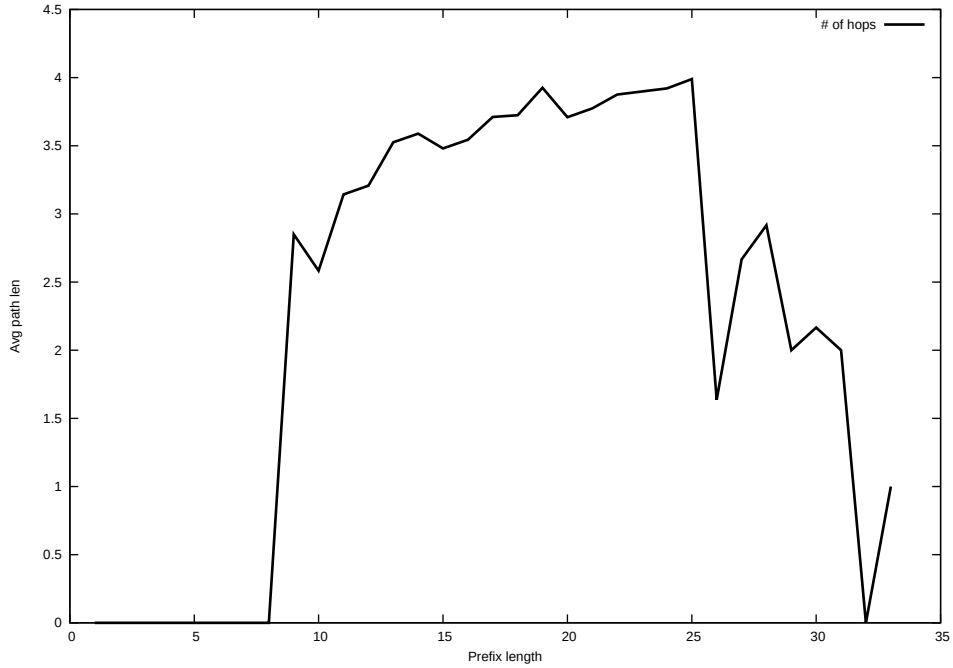
2012-07-18



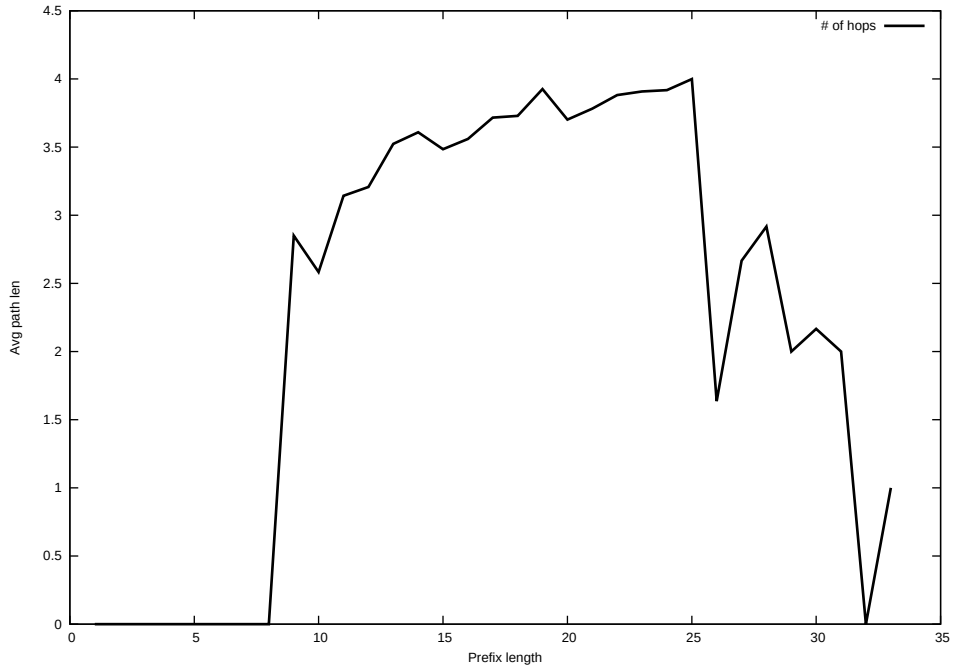
2012-07-19



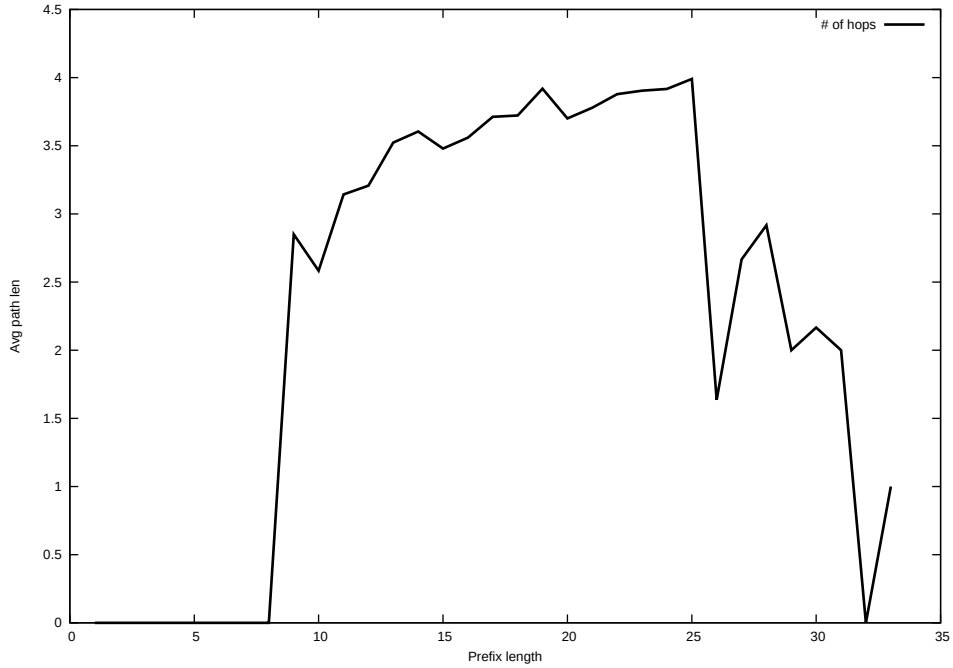
2012-07-20



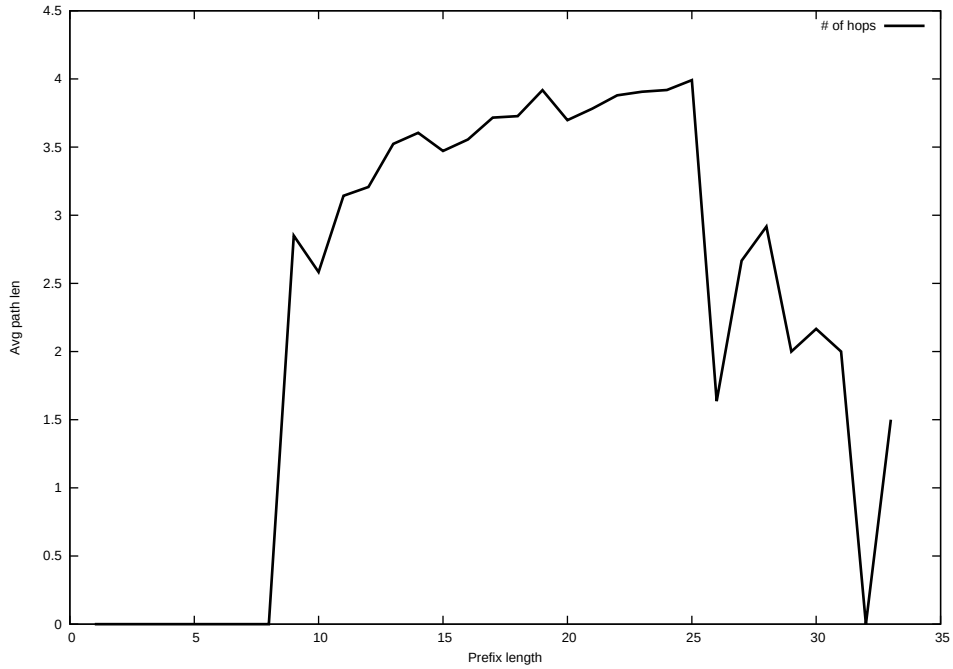
2012-07-21



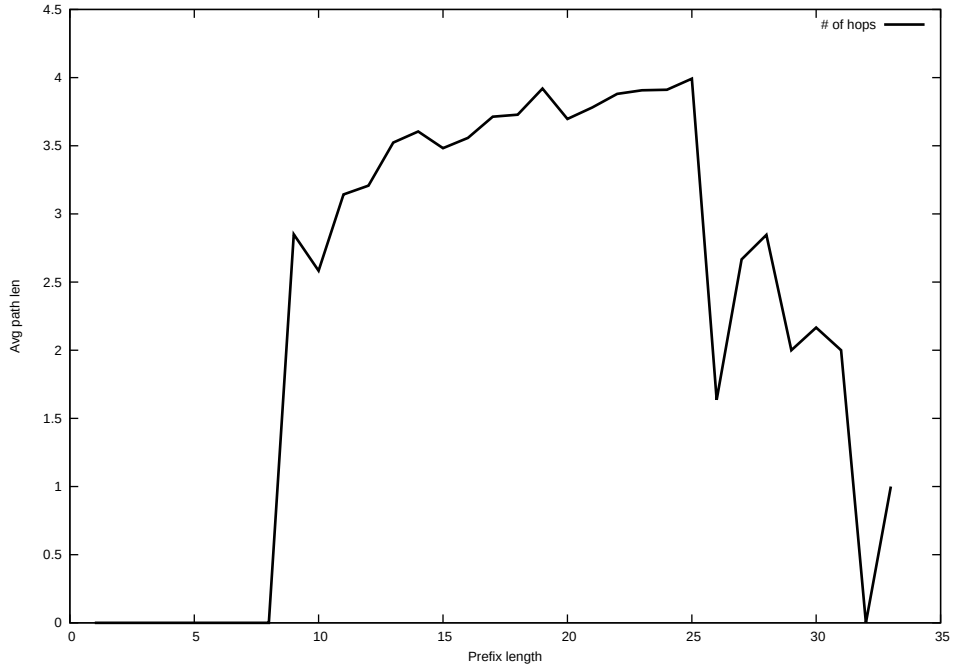
2012-07-22



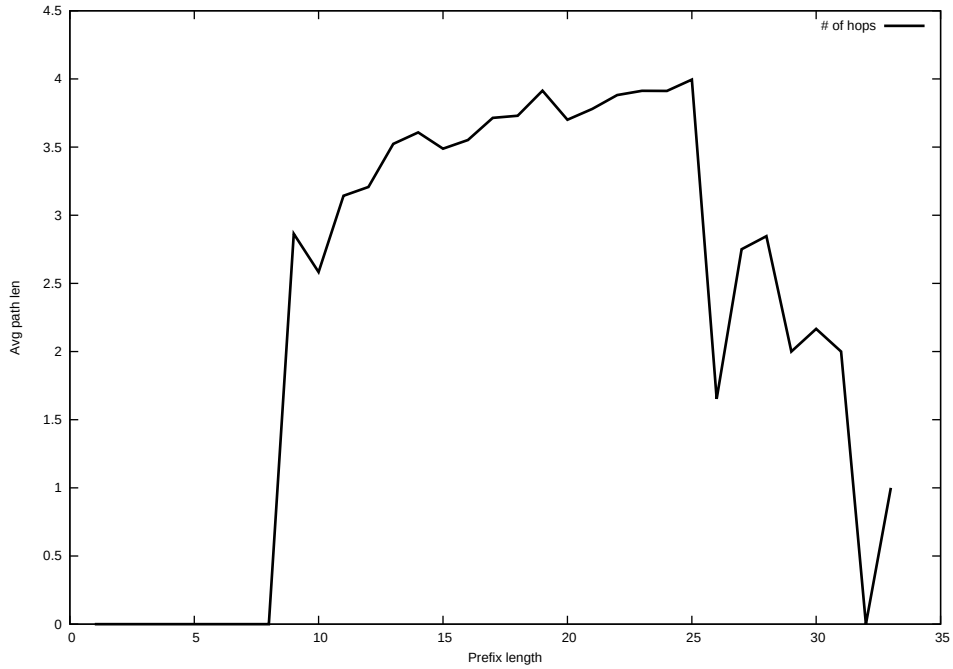
2012-07-23



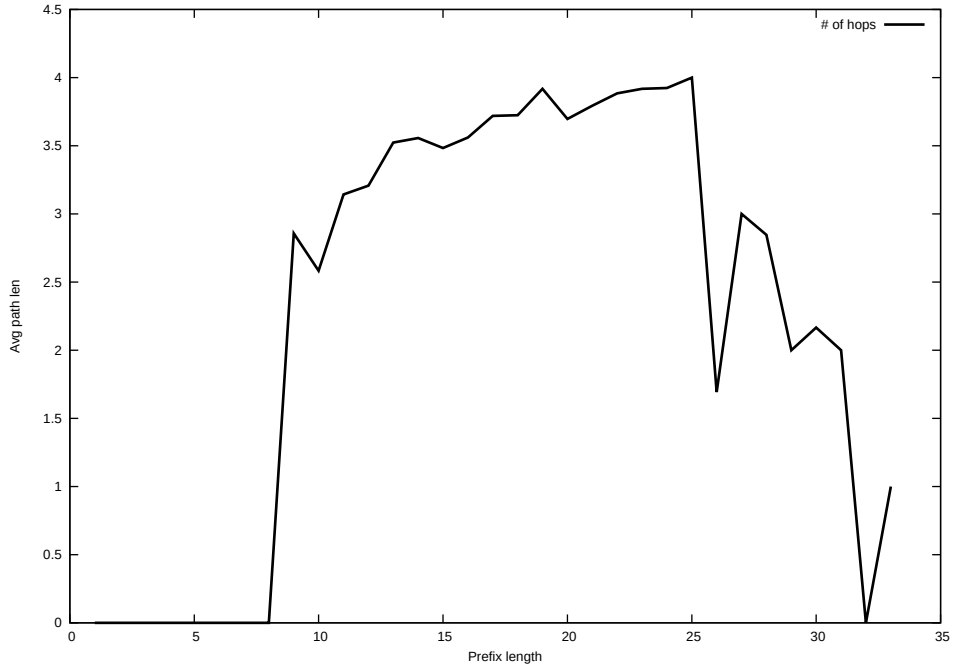
2012-07-24



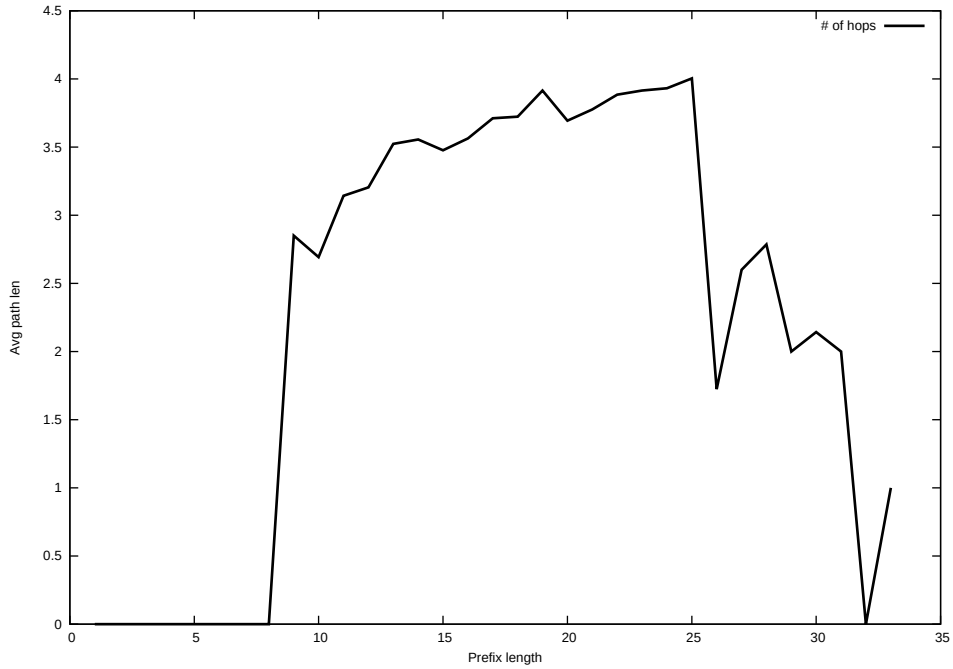
2012-07-25



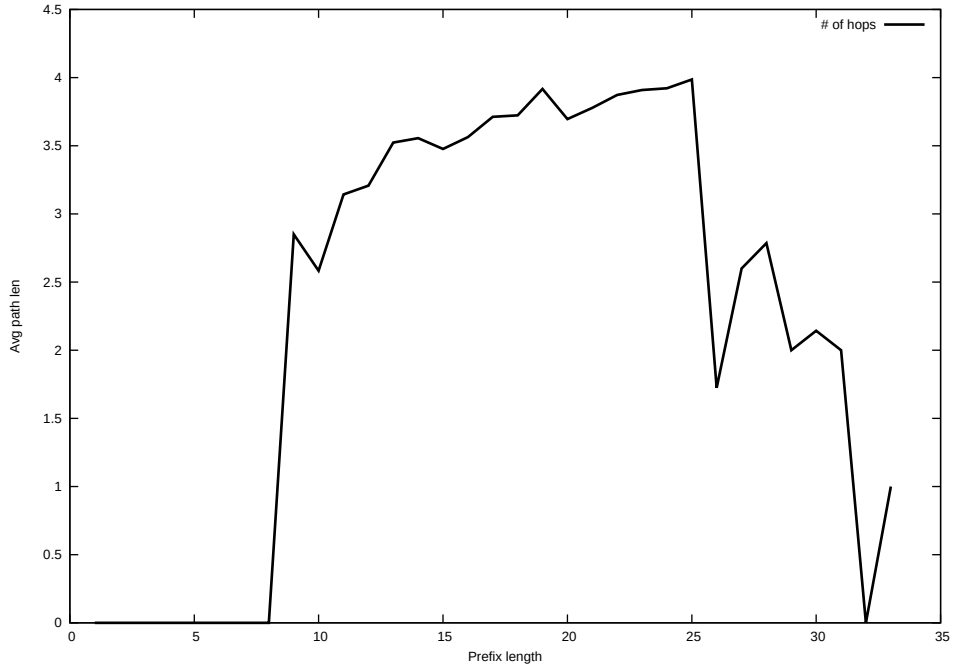
2012-07-26



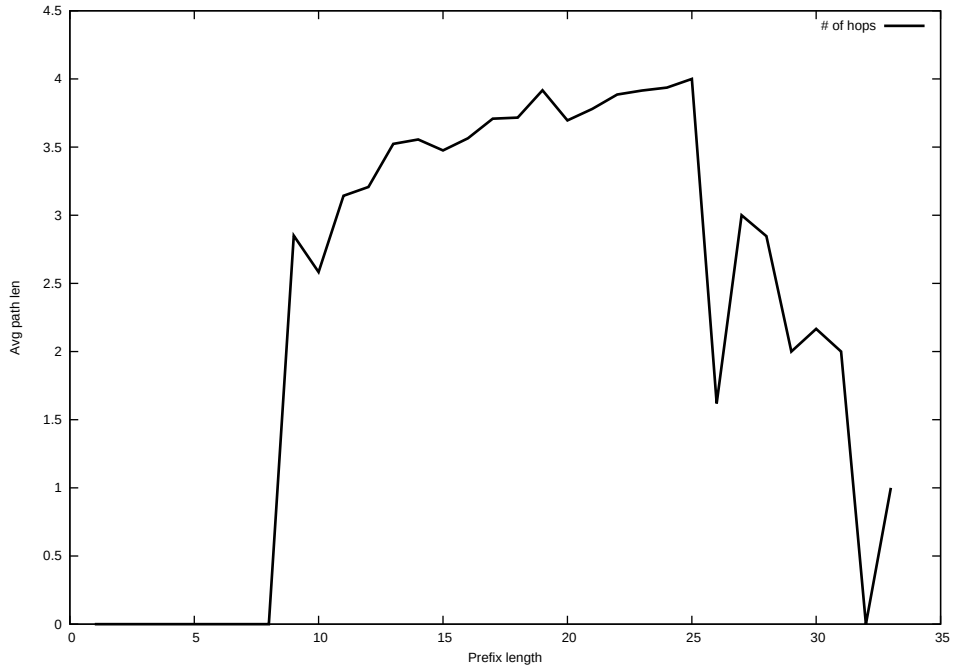
2012-07-27



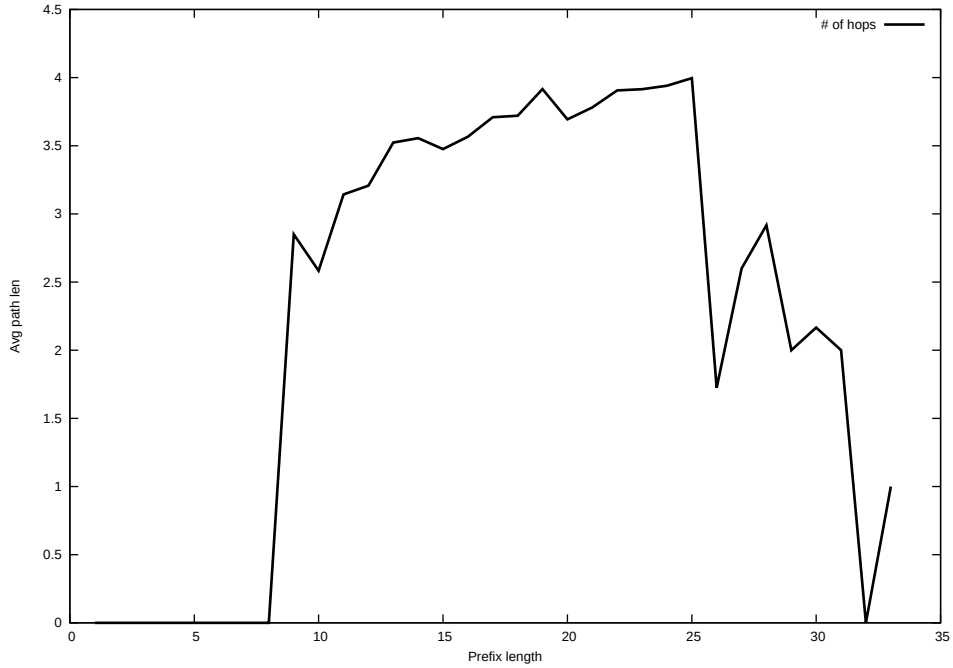
2012-07-28



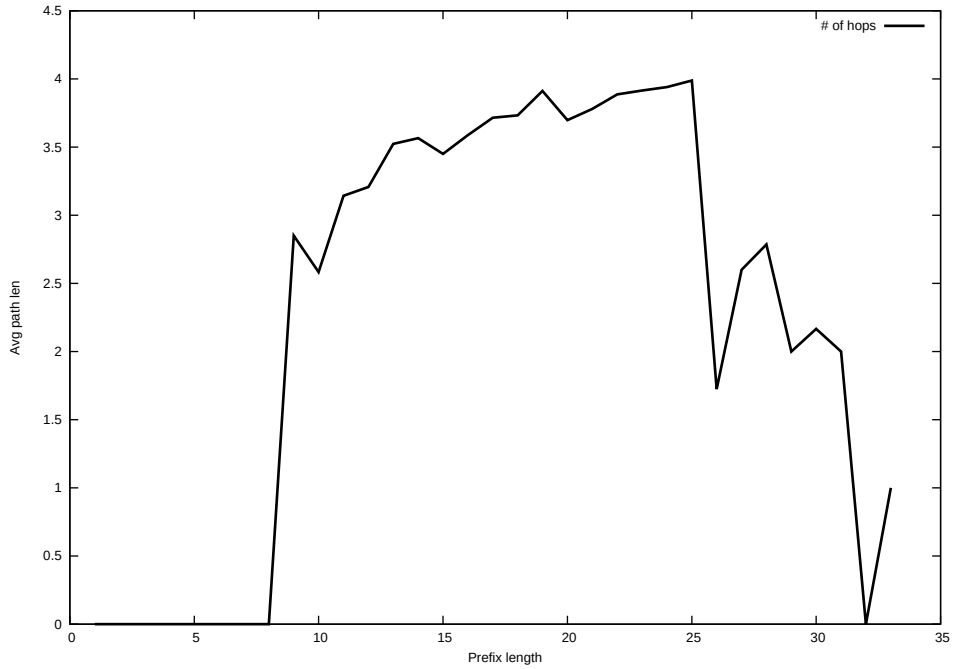
2012-07-29



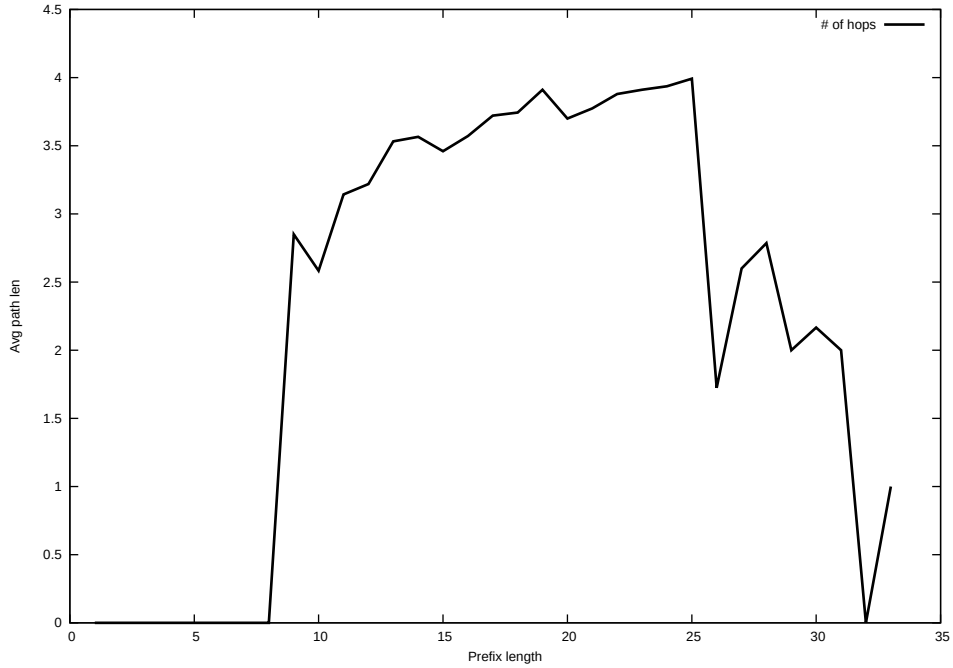
2012-07-30



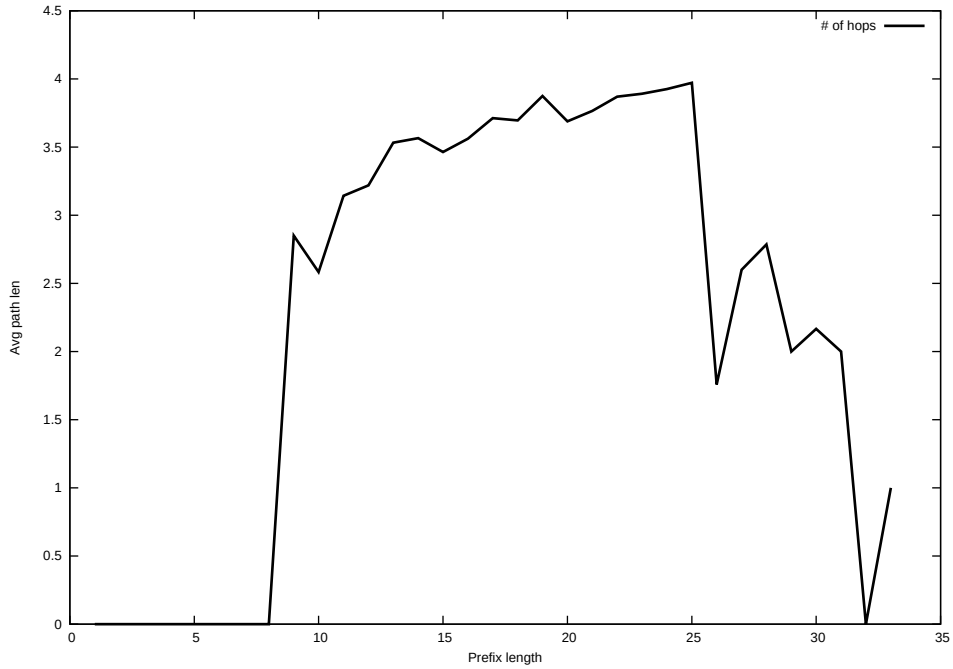
2012-07-31



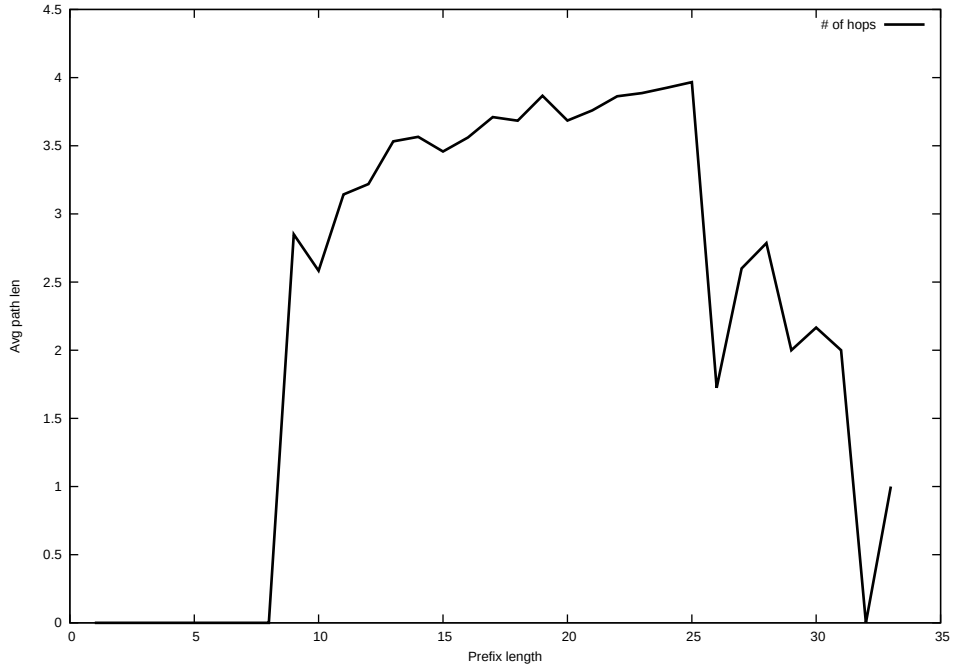
2012-08-01



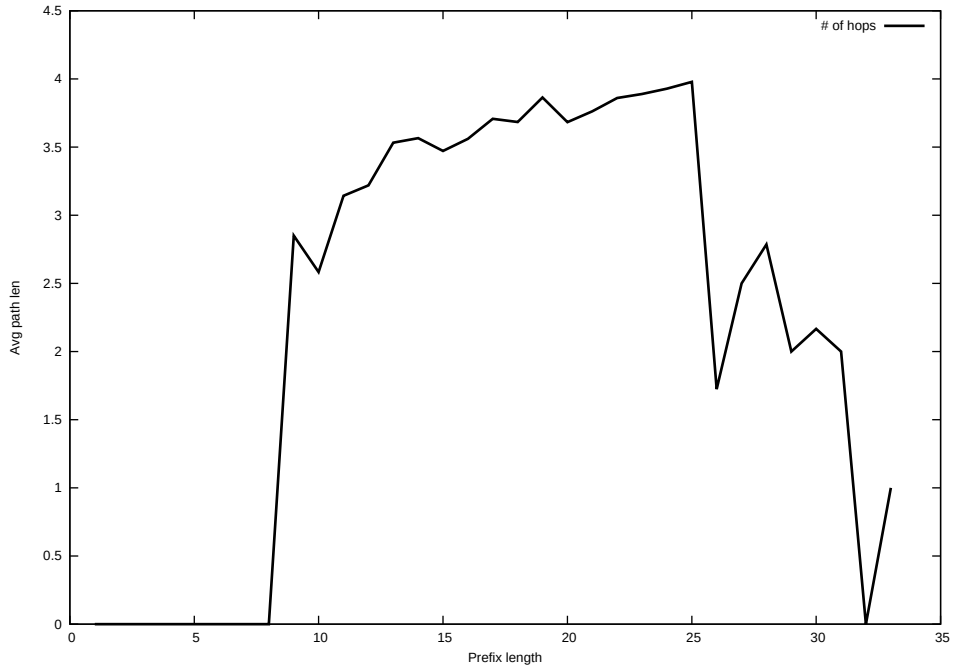
2012-08-02



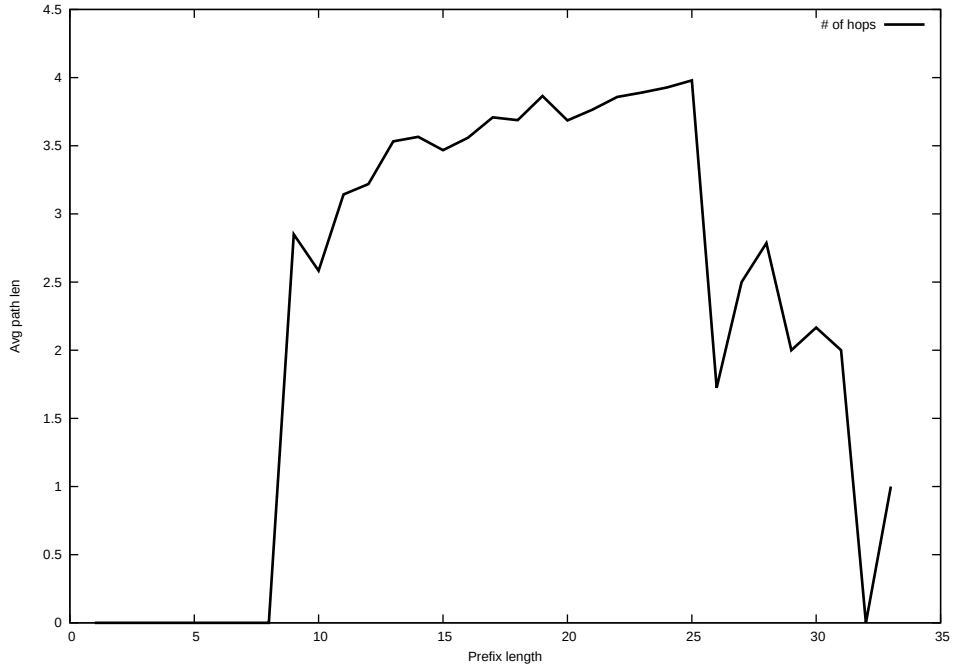
2012-08-03



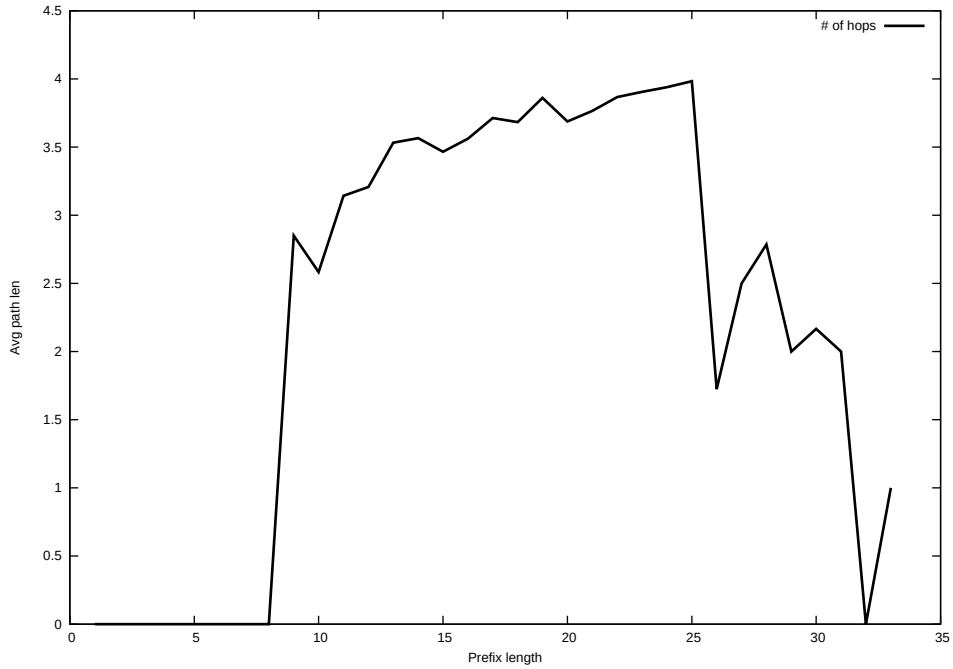
2012-08-04



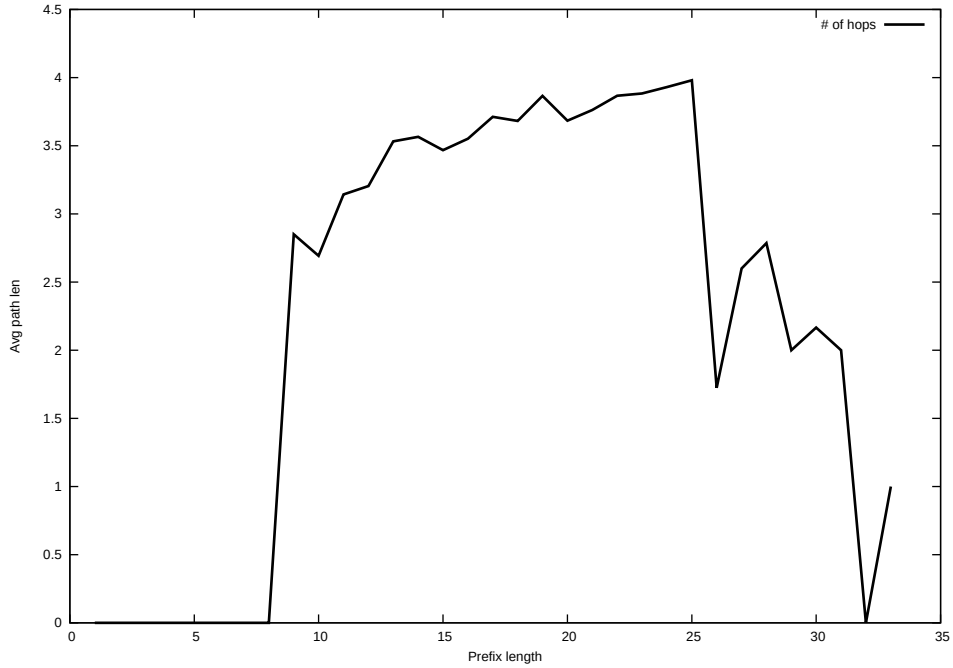
2012-08-05



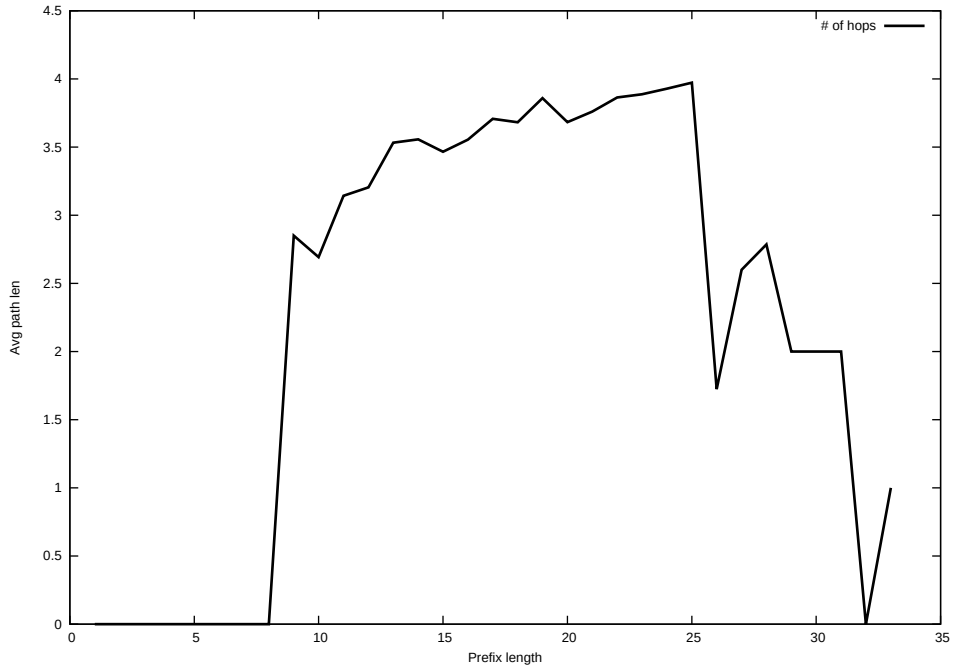
2012-08-06



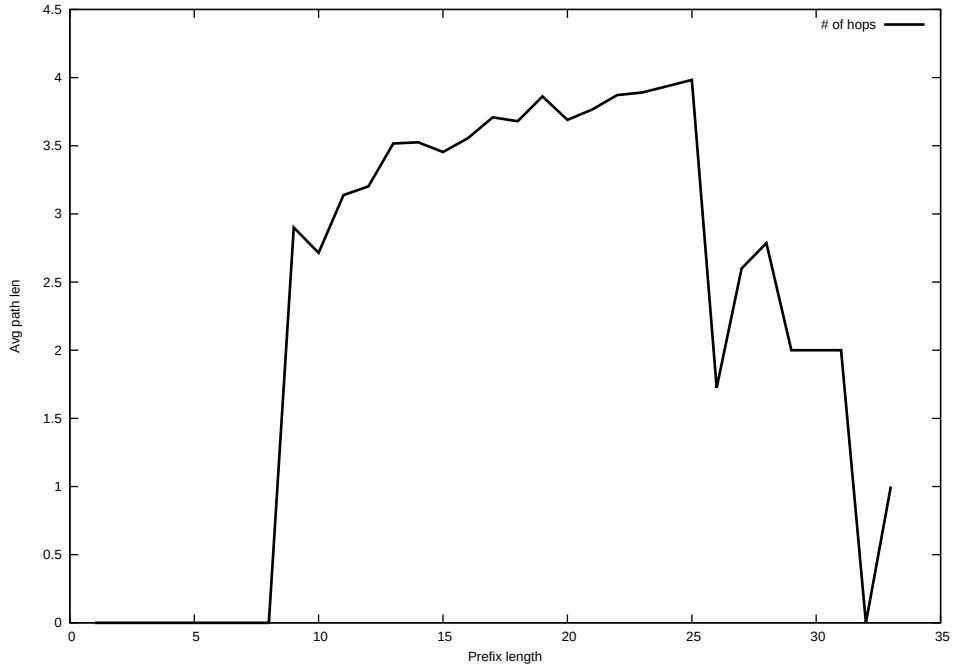
2012-08-07



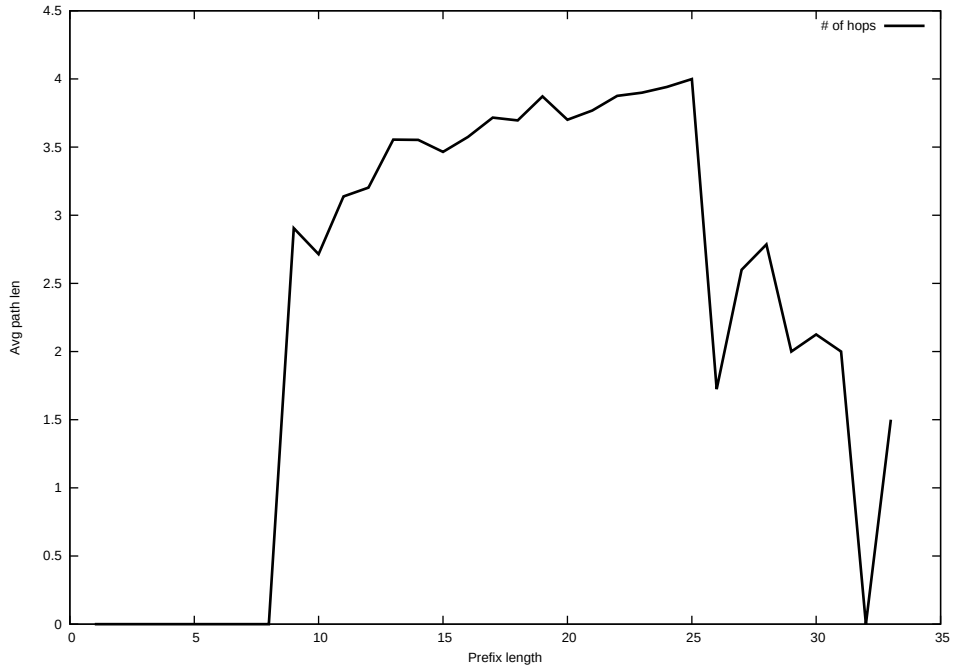
2012-08-08



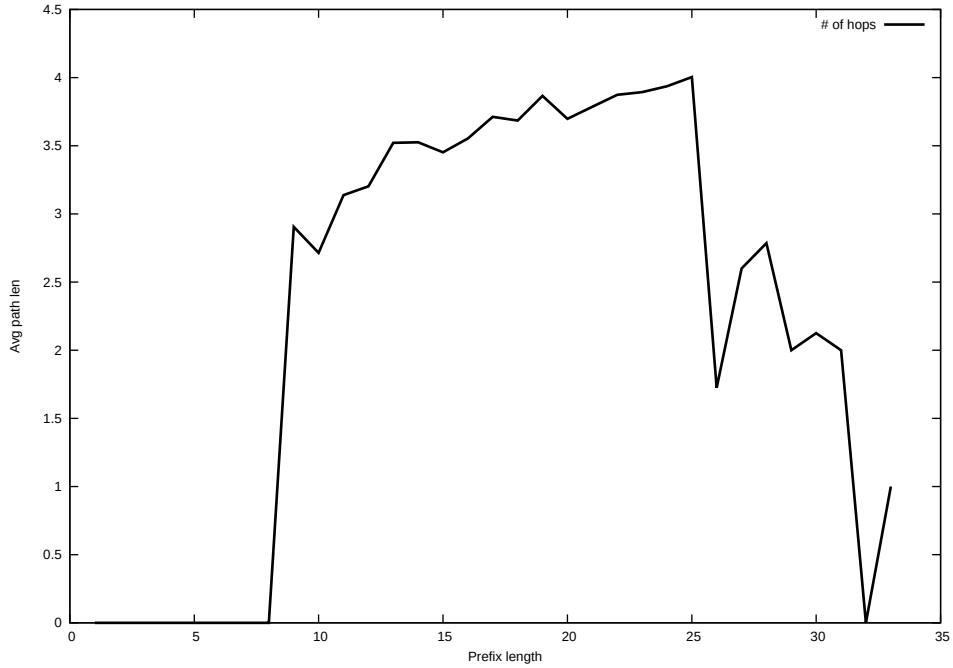
2012-08-09



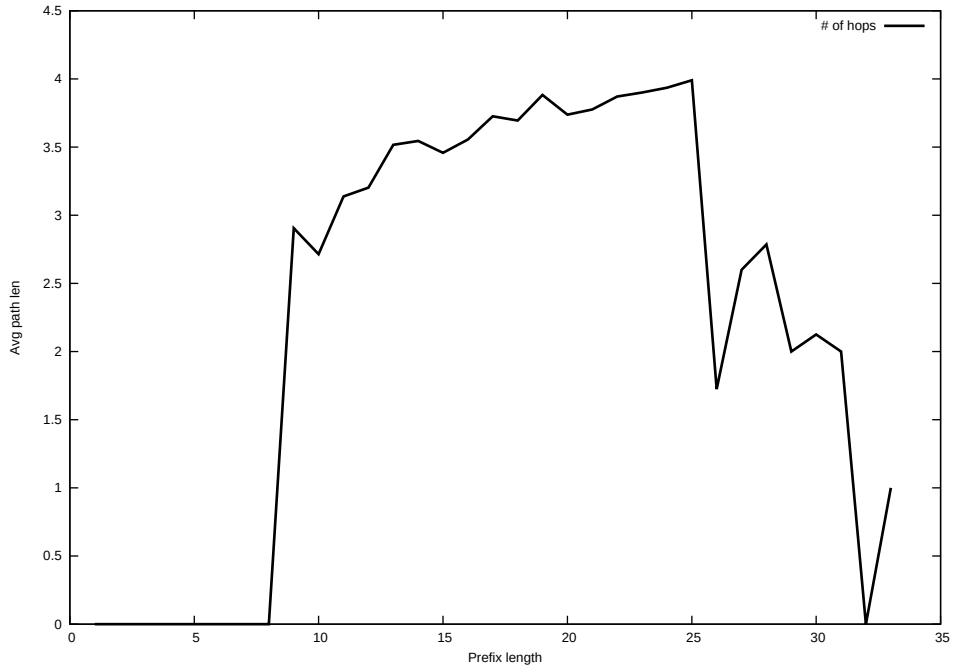
2012-08-10



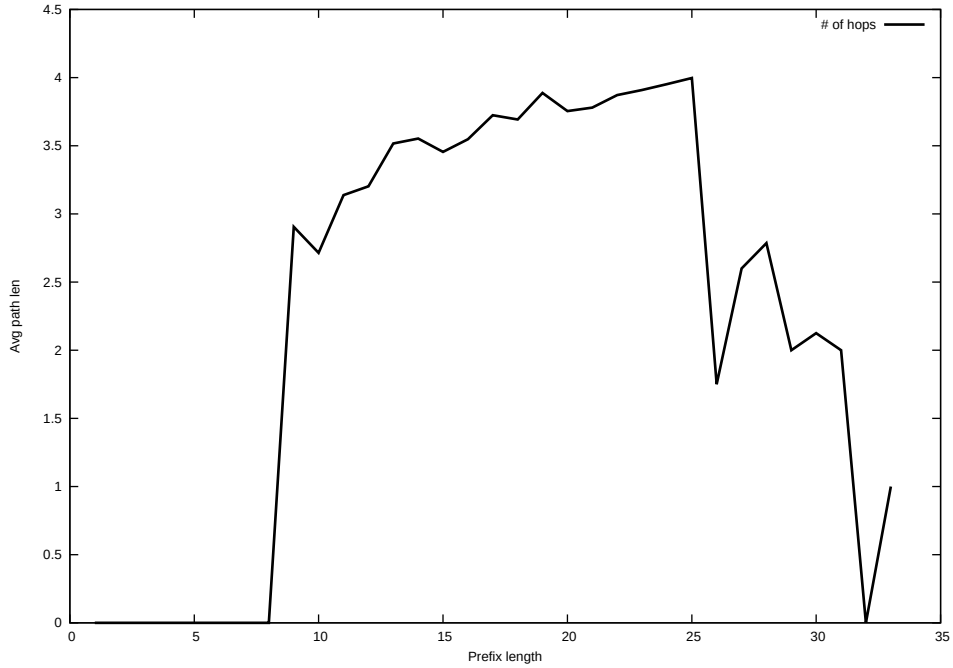
2012-08-11



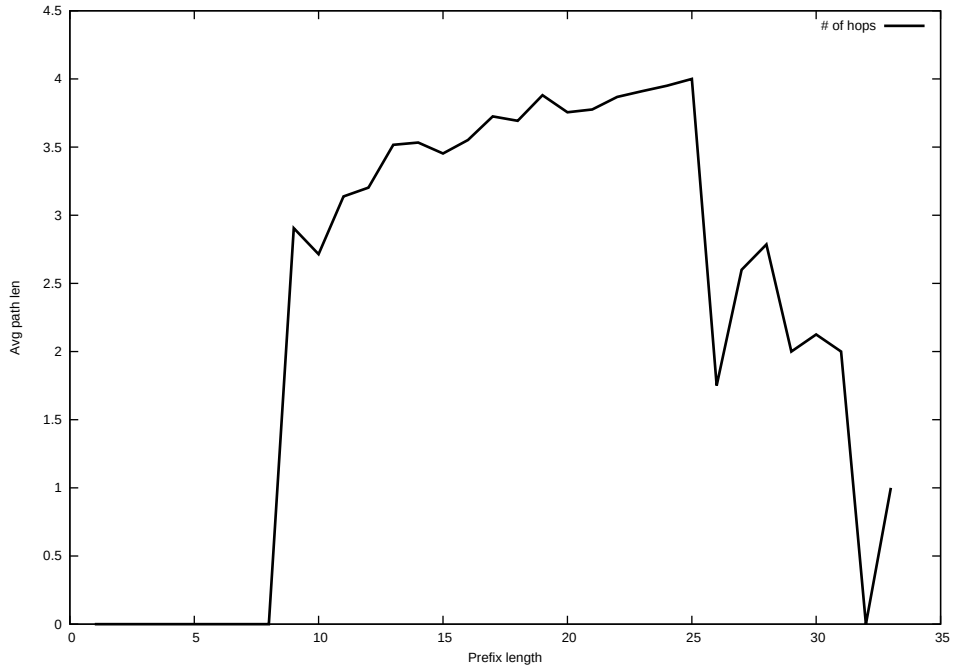
2012-08-12



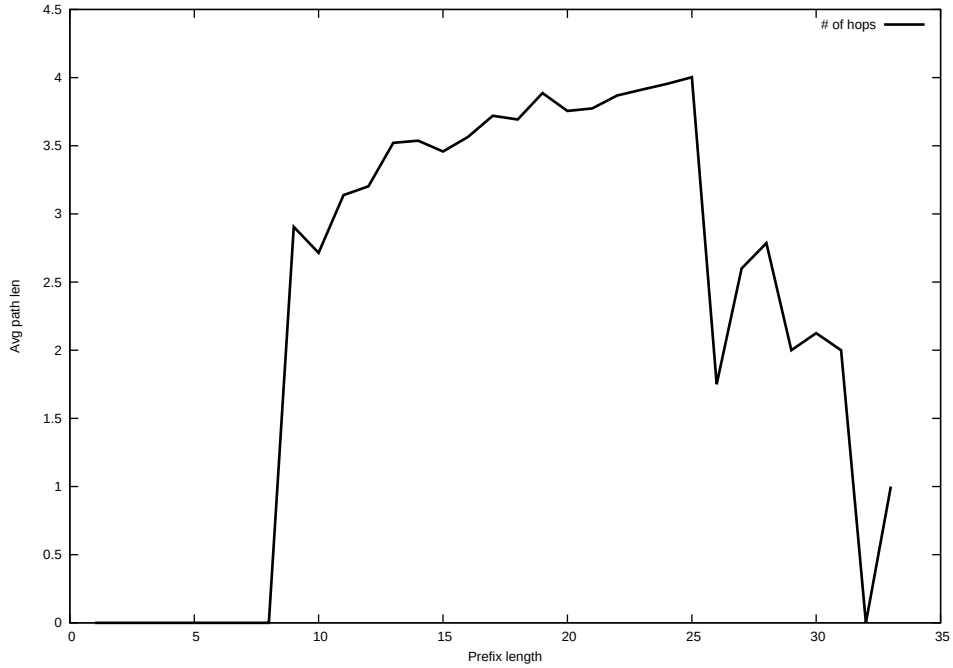
2012-08-13



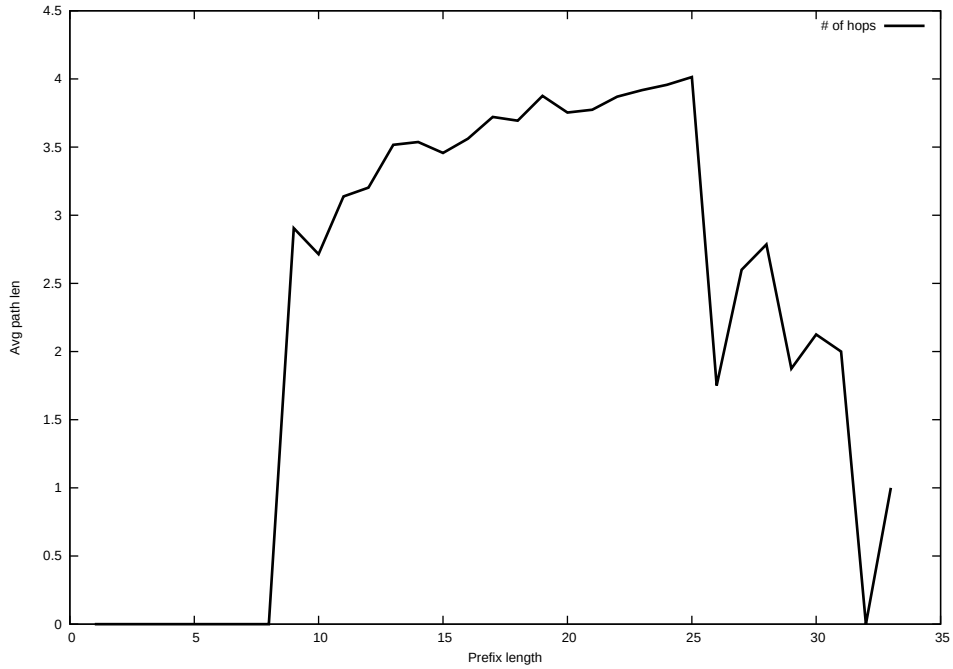
2012-08-14



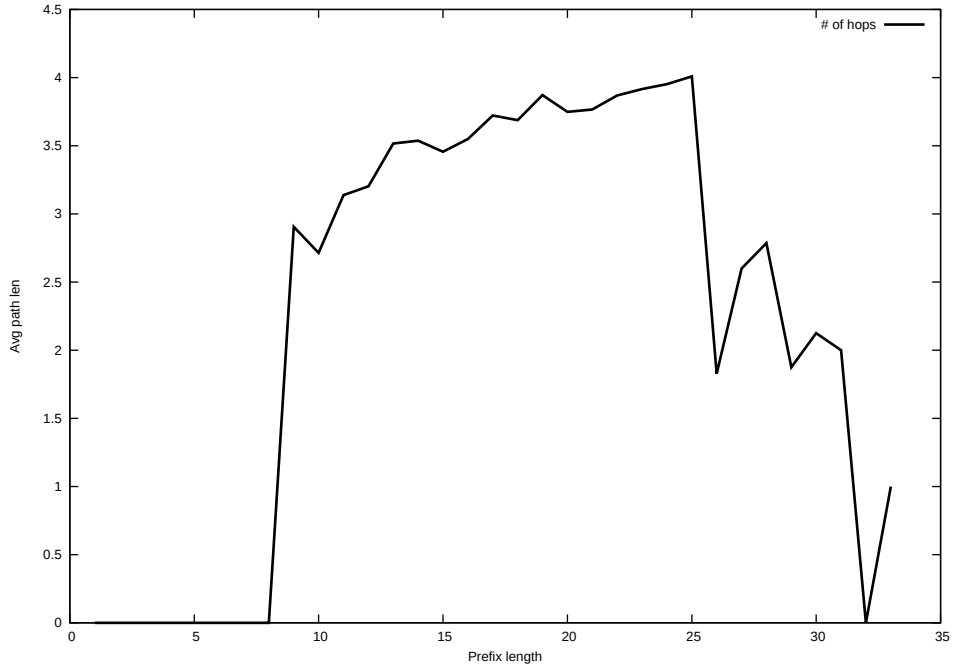
2012-08-15



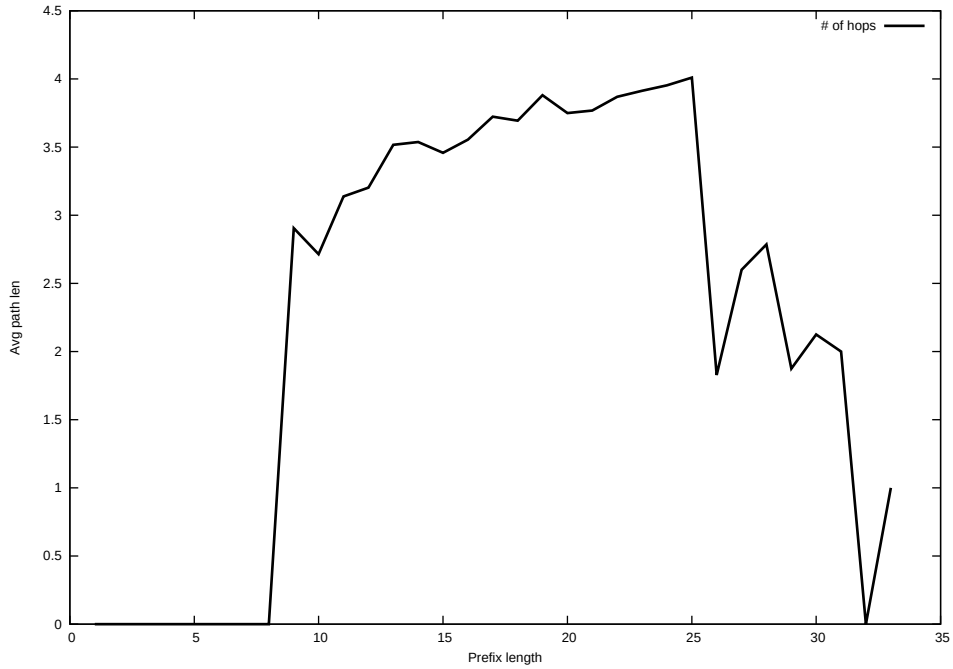
2012-08-16



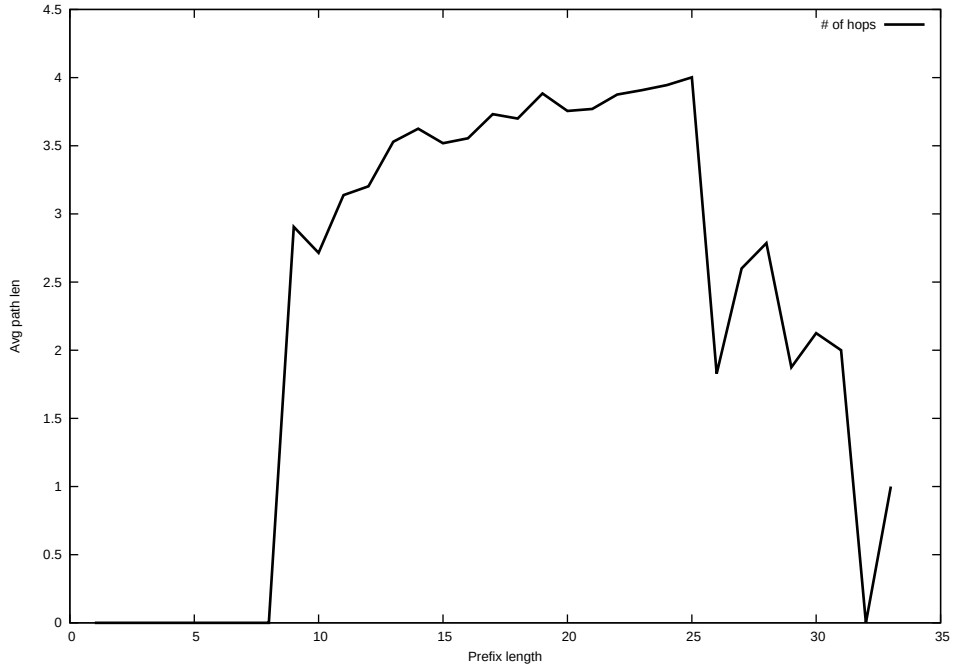
2012-08-17



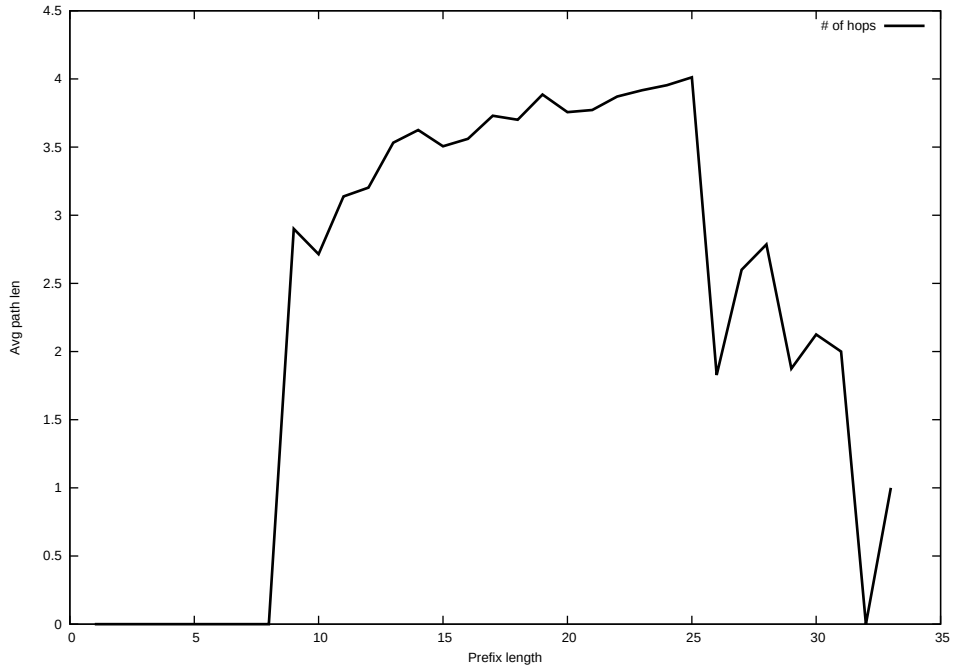
2012-08-18



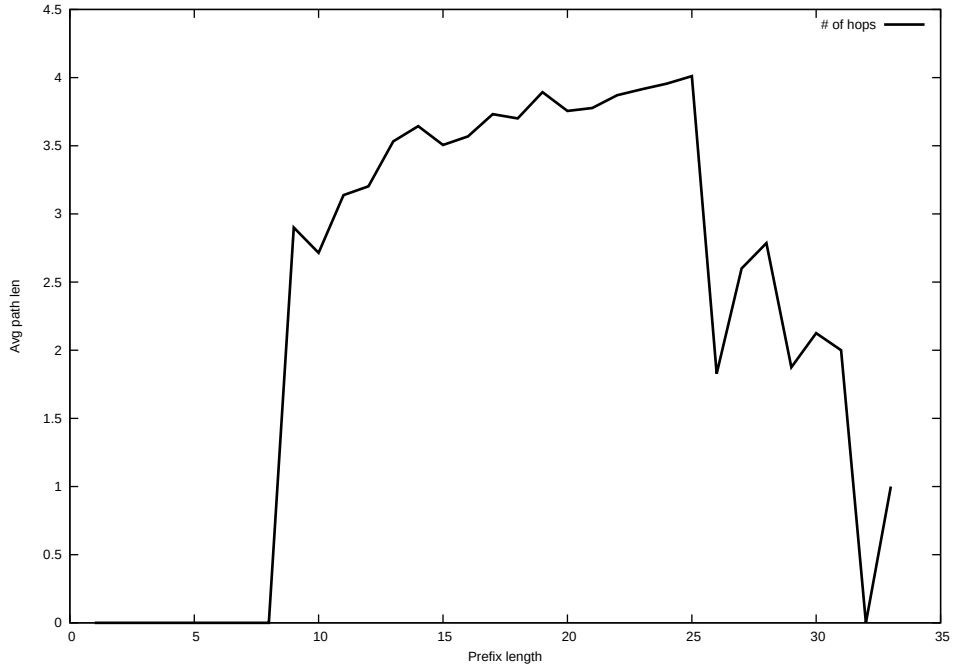
2012-08-19



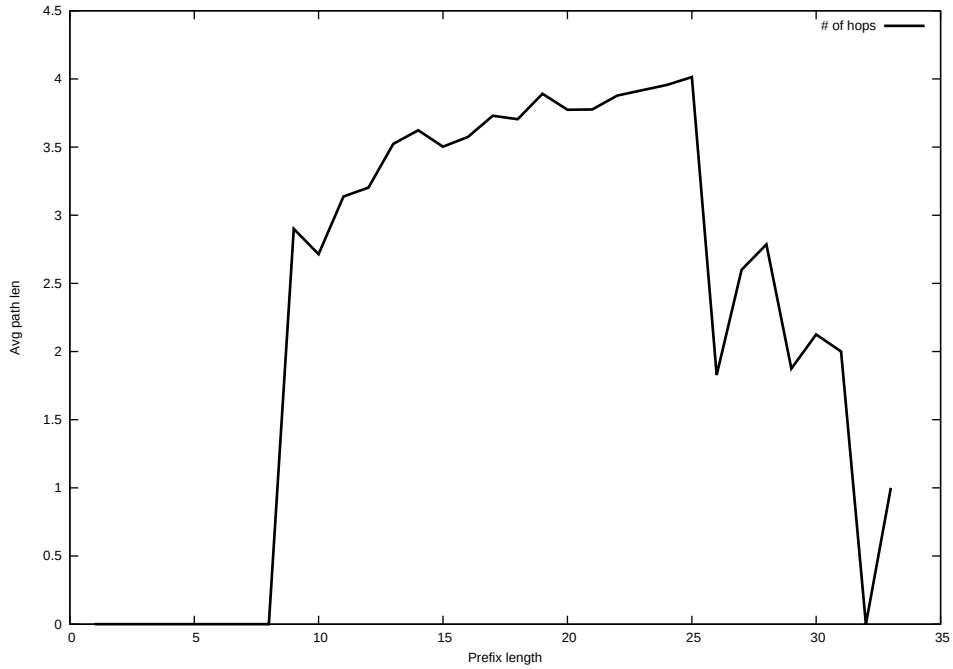
2012-08-20



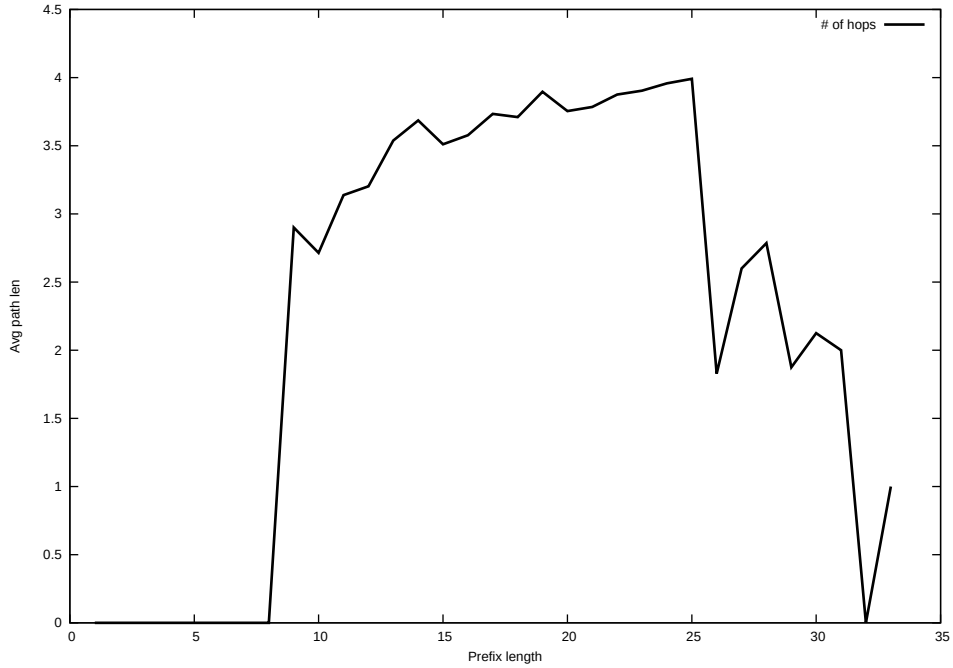
2012-08-21



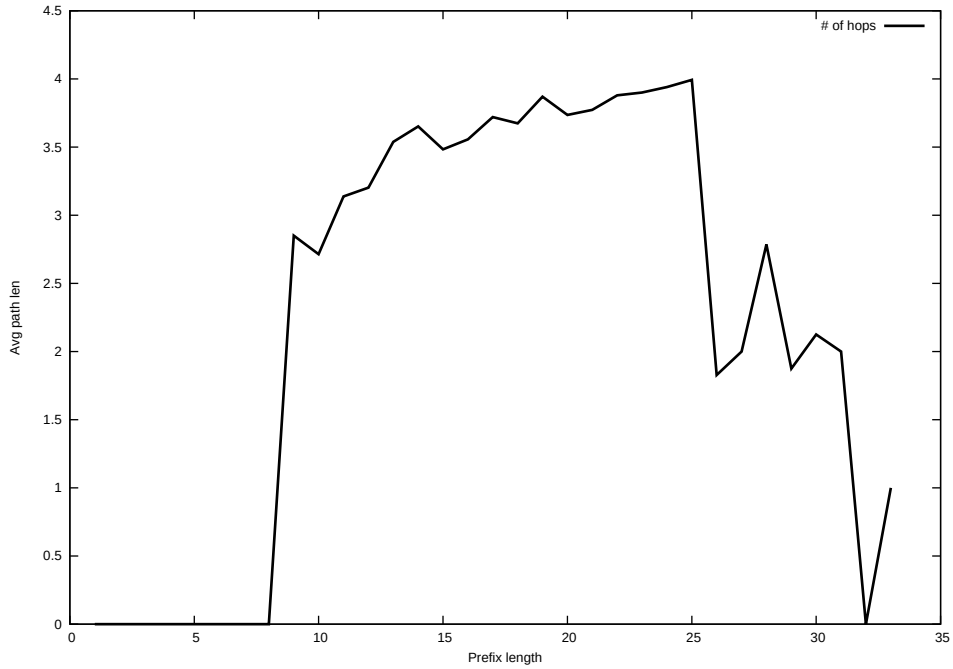
2012-08-22



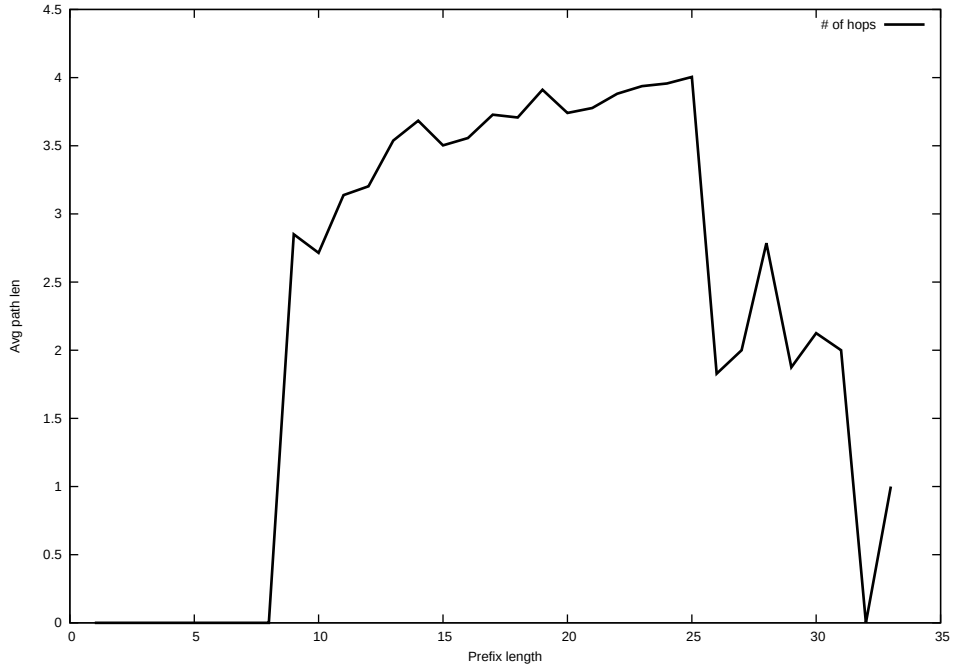
2012-08-23



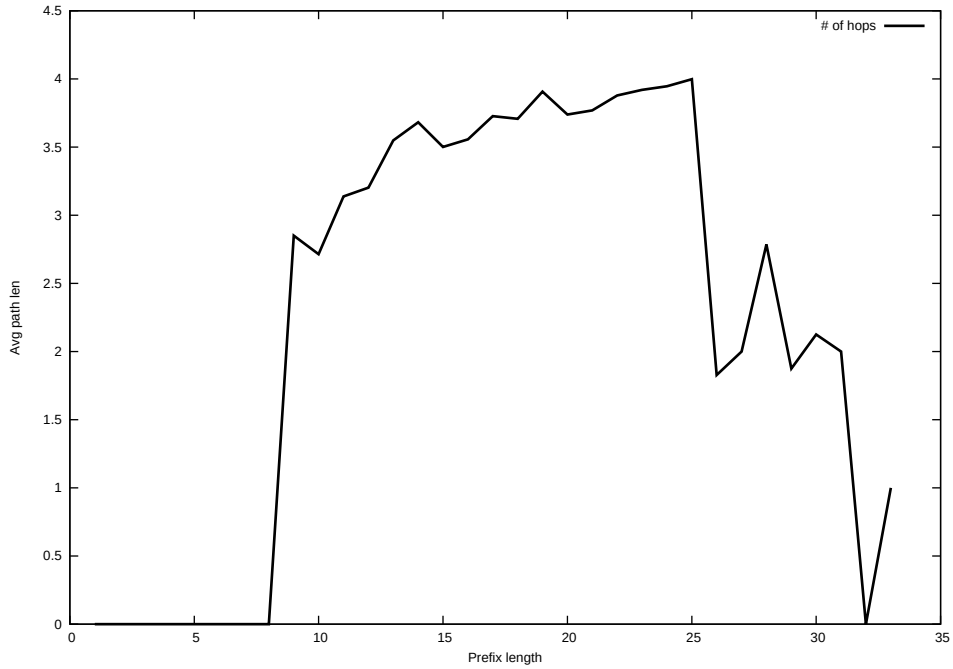
2012-08-24



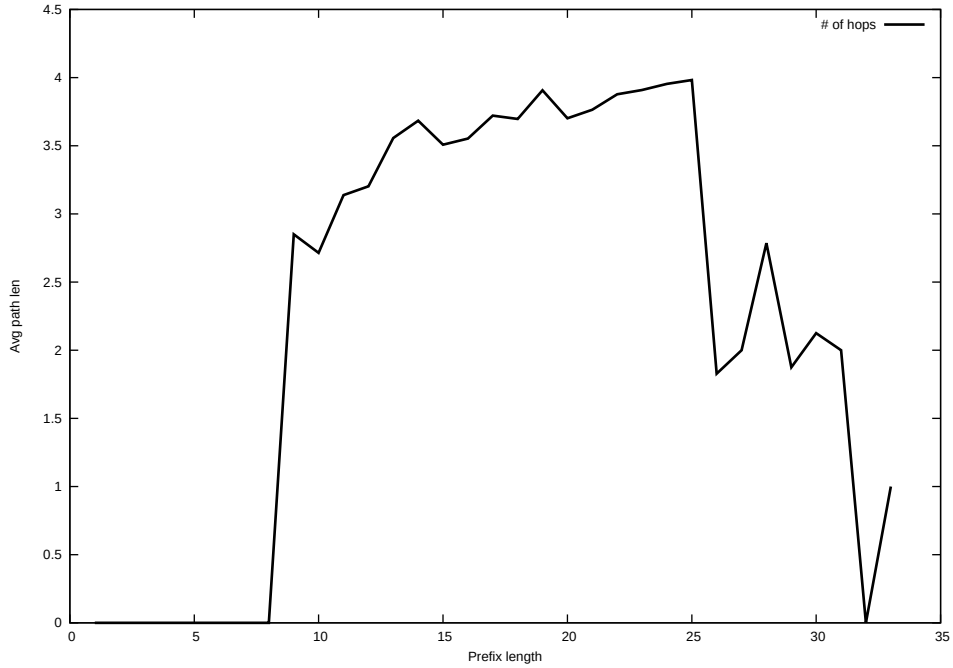
2012-08-25



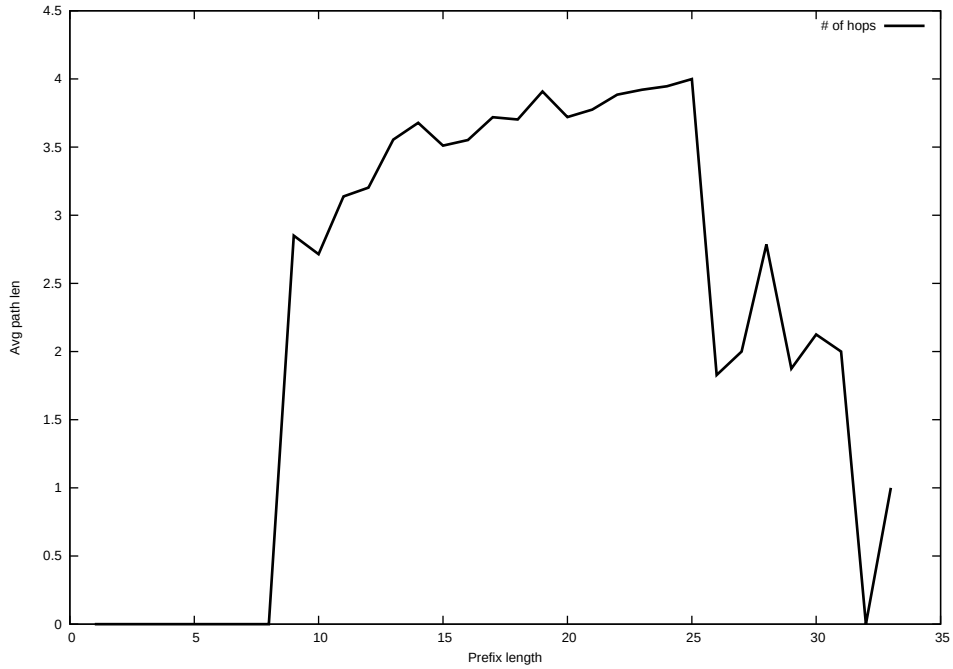
2012-08-26



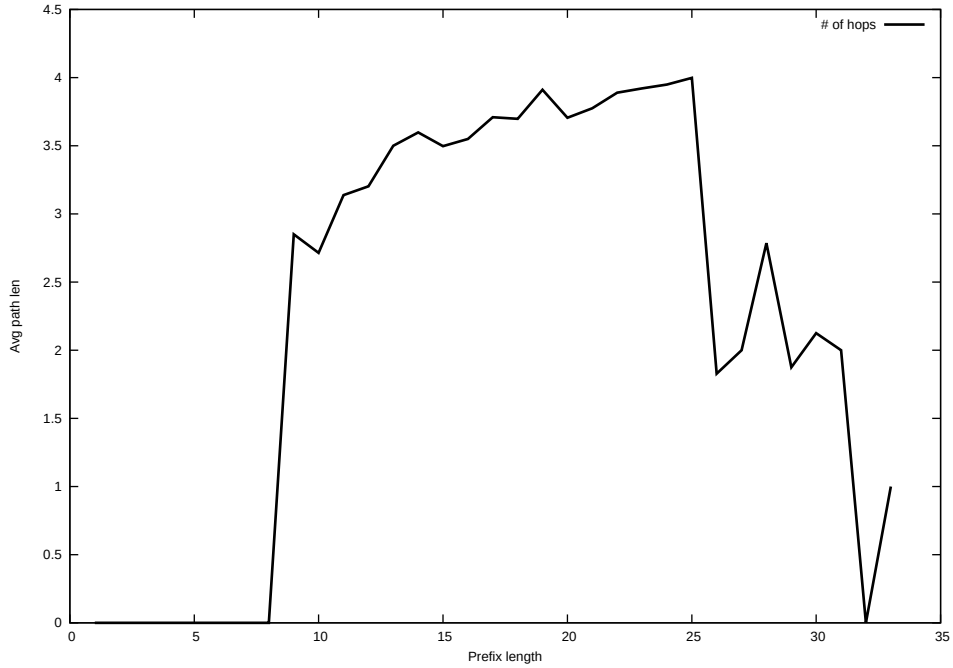
2012-08-27



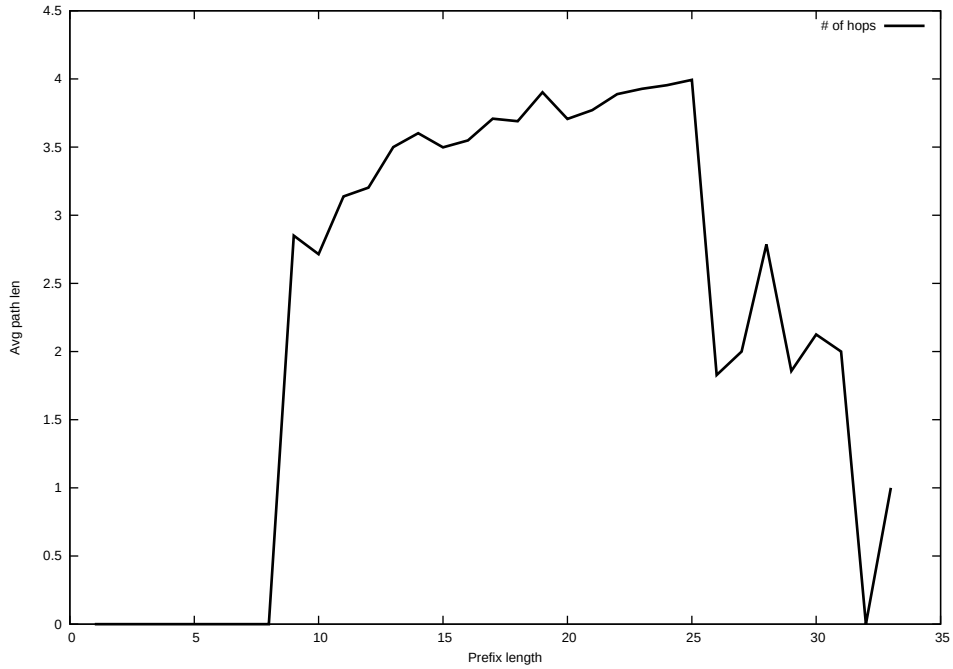
2012-08-28



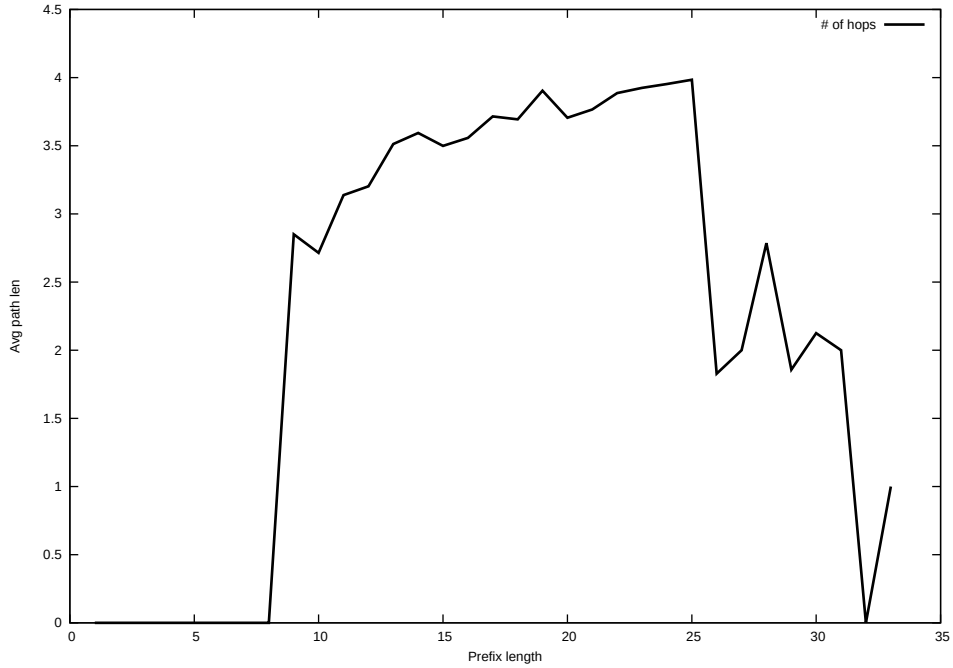
2012-08-29



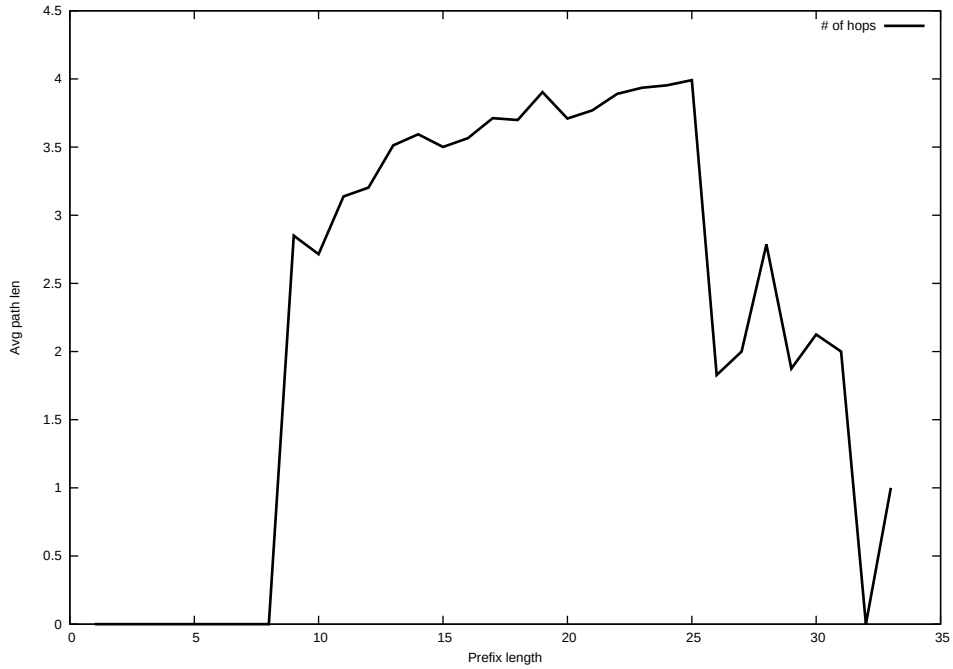
2012-08-30



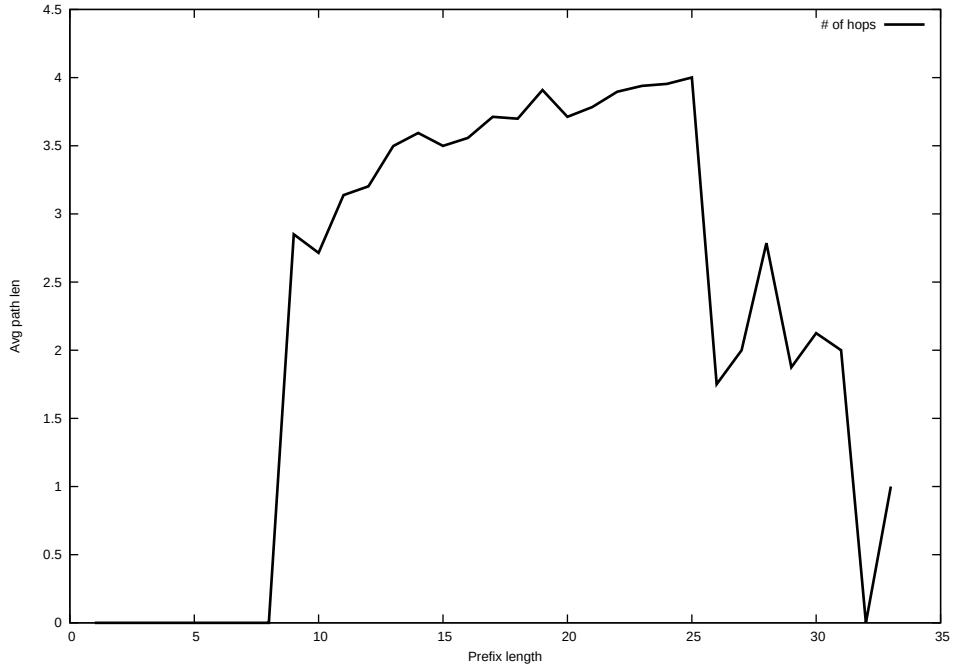
2012-08-31



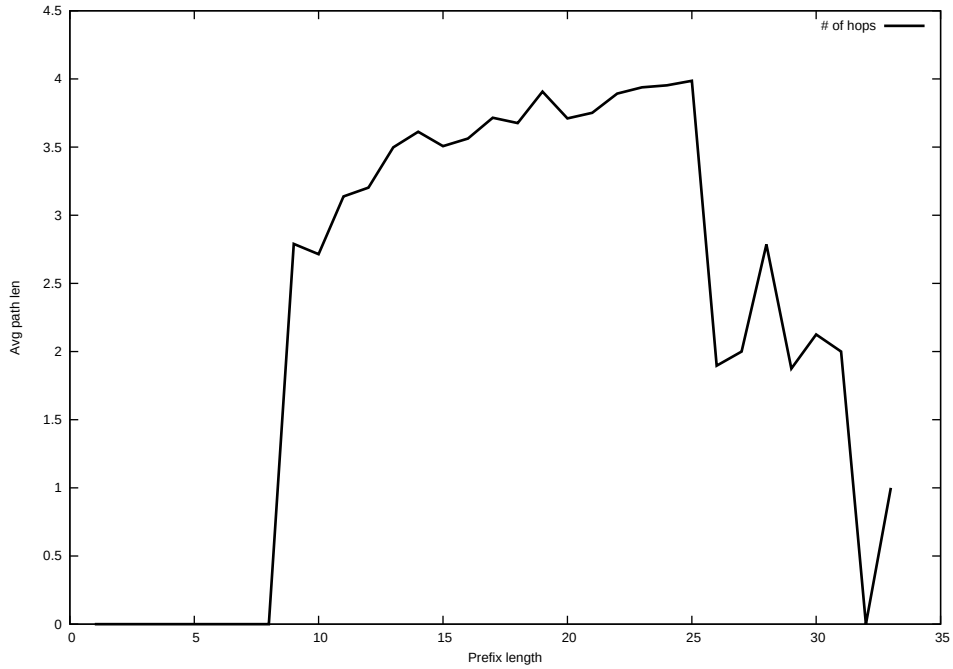
2012-09-01



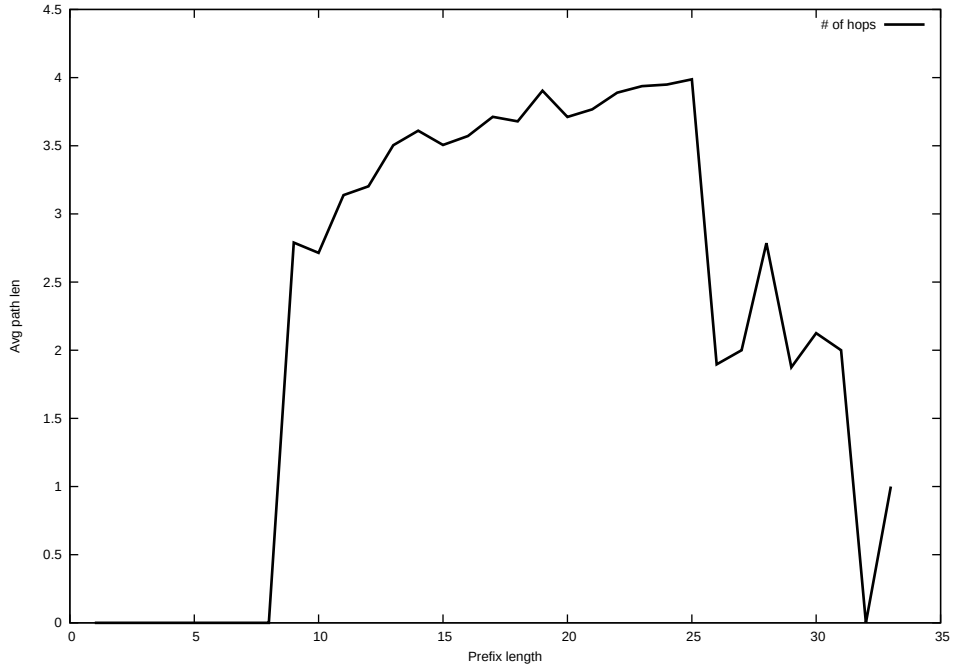
2012-09-02



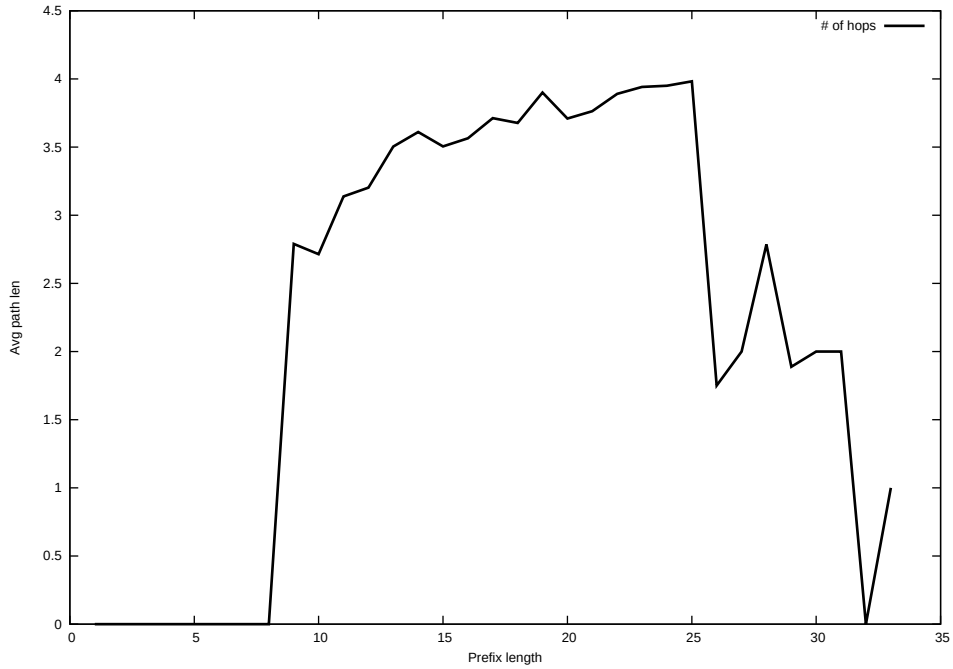
2012-09-03



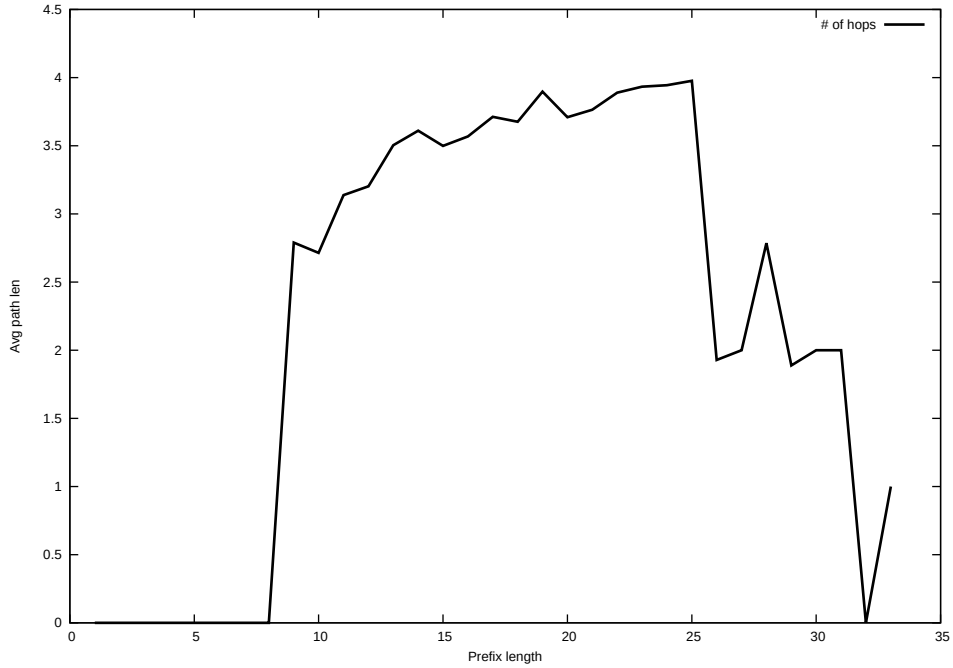
2012-09-04



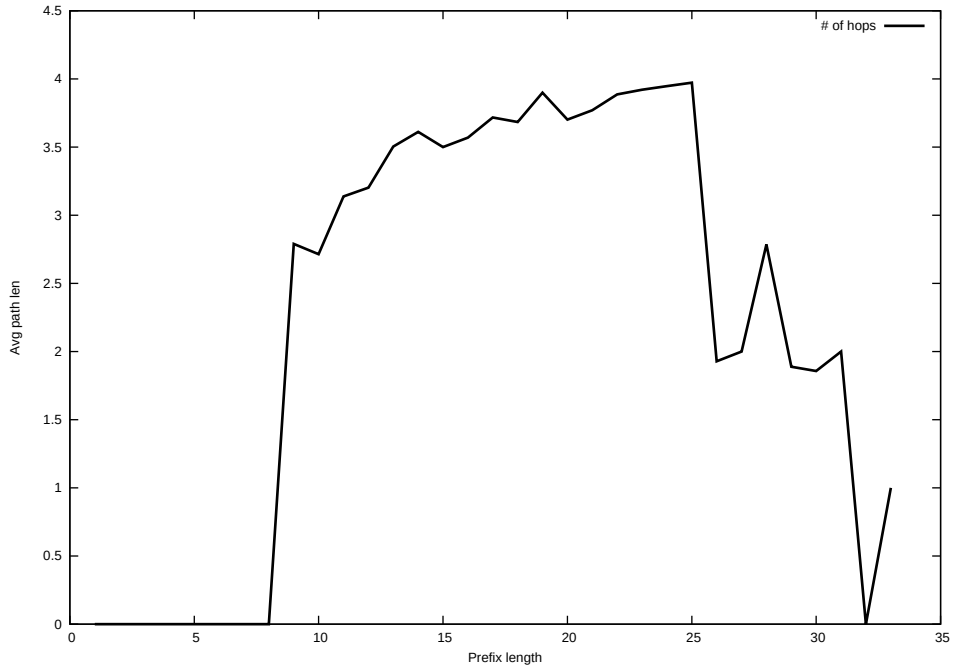
2012-09-05



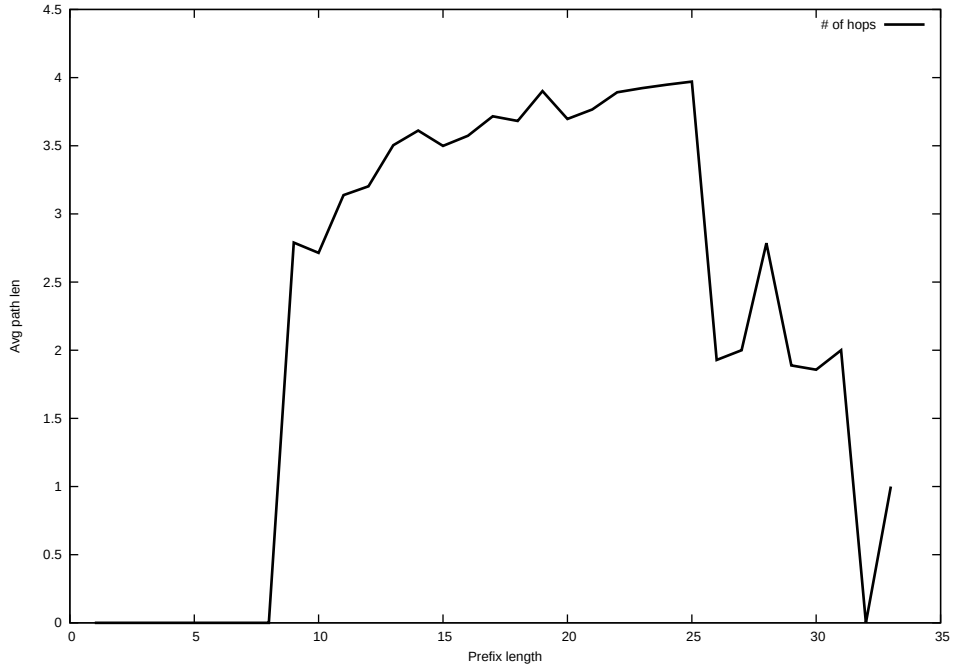
2012-09-06



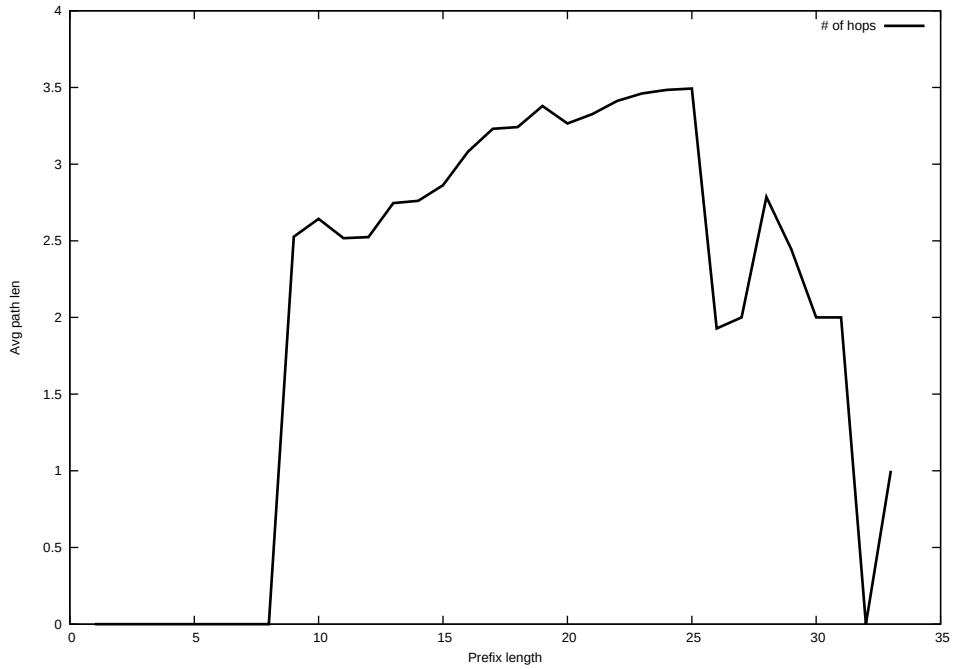
2012-09-07



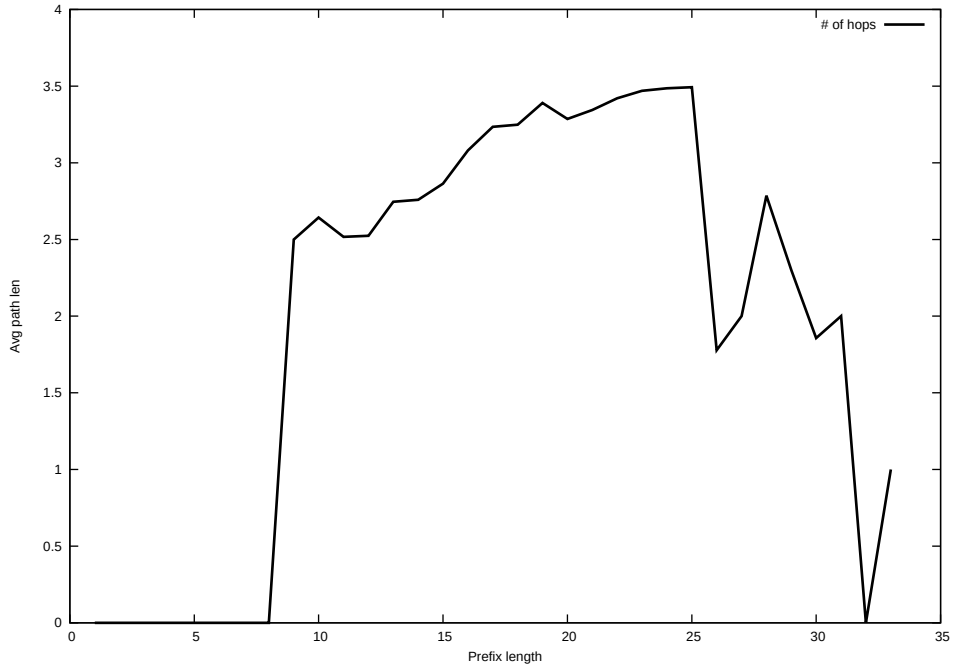
2012-09-08



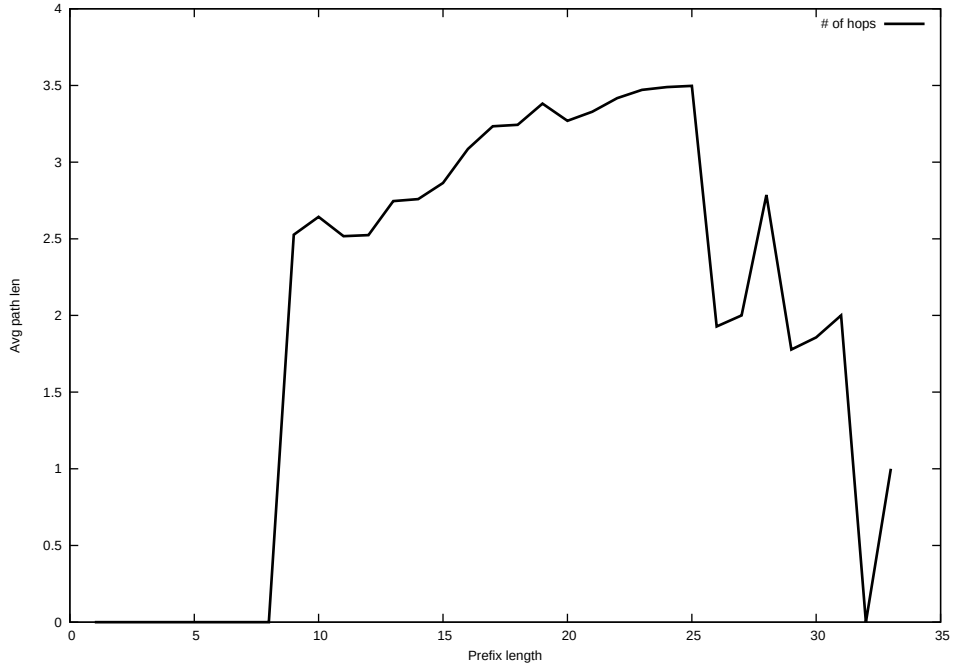
2012-09-09



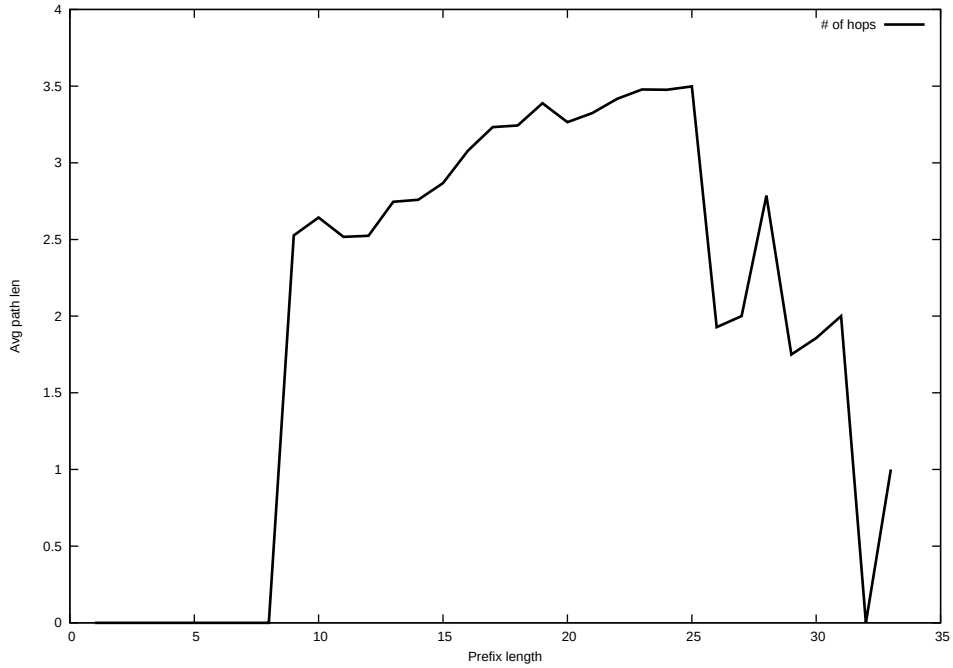
2012-09-10



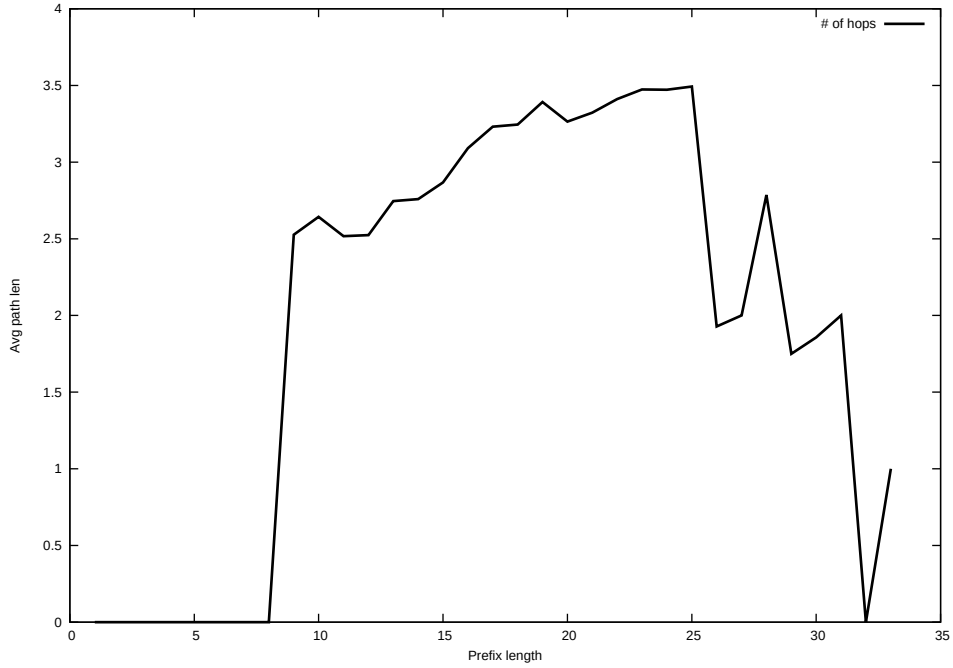
2012-09-11



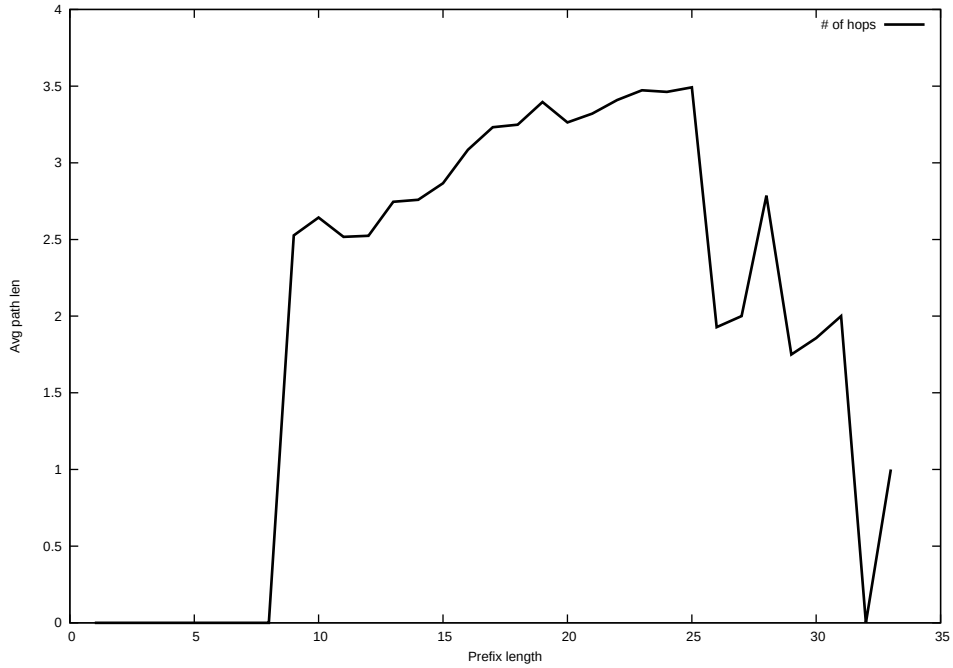
2012-09-12



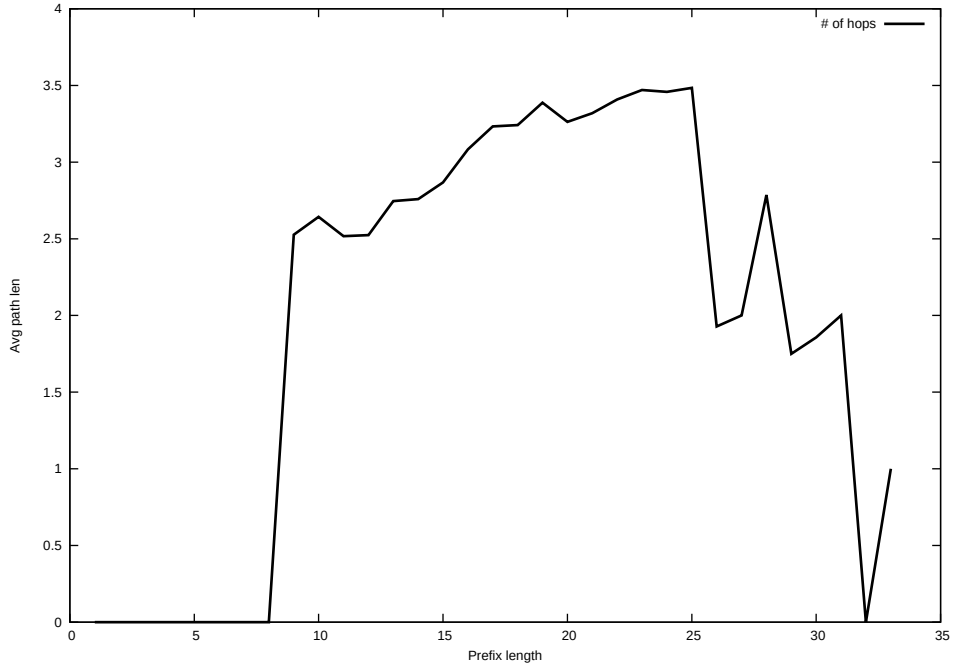
2012-09-13



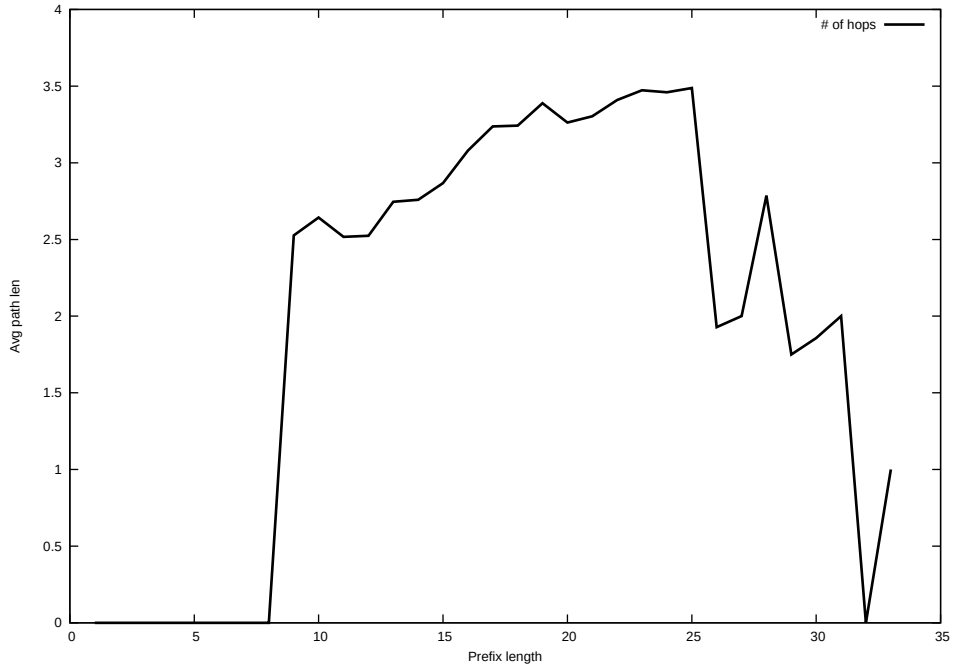
2012-09-14



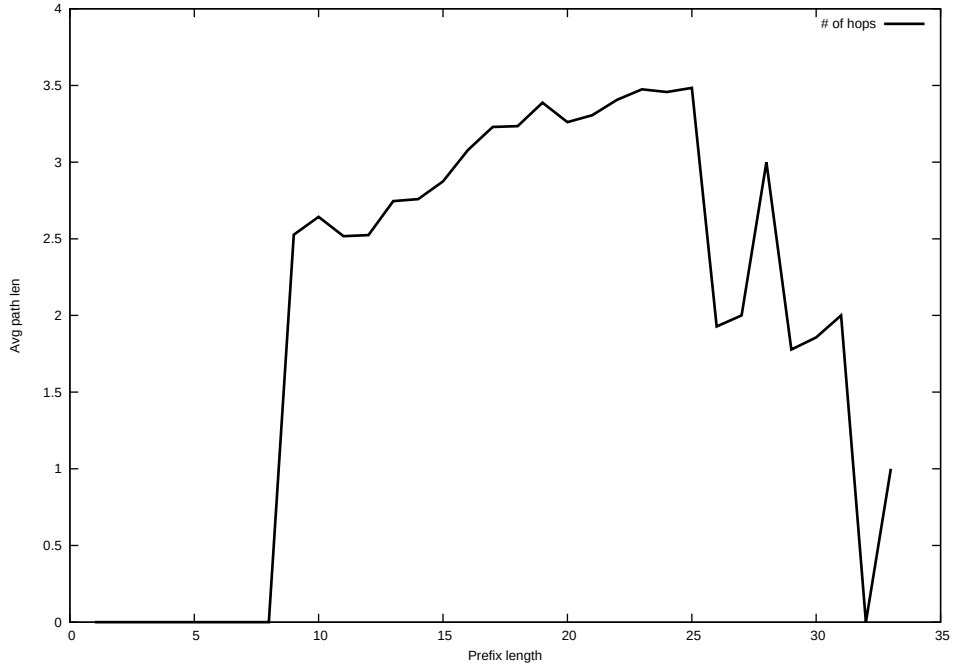
2012-09-15



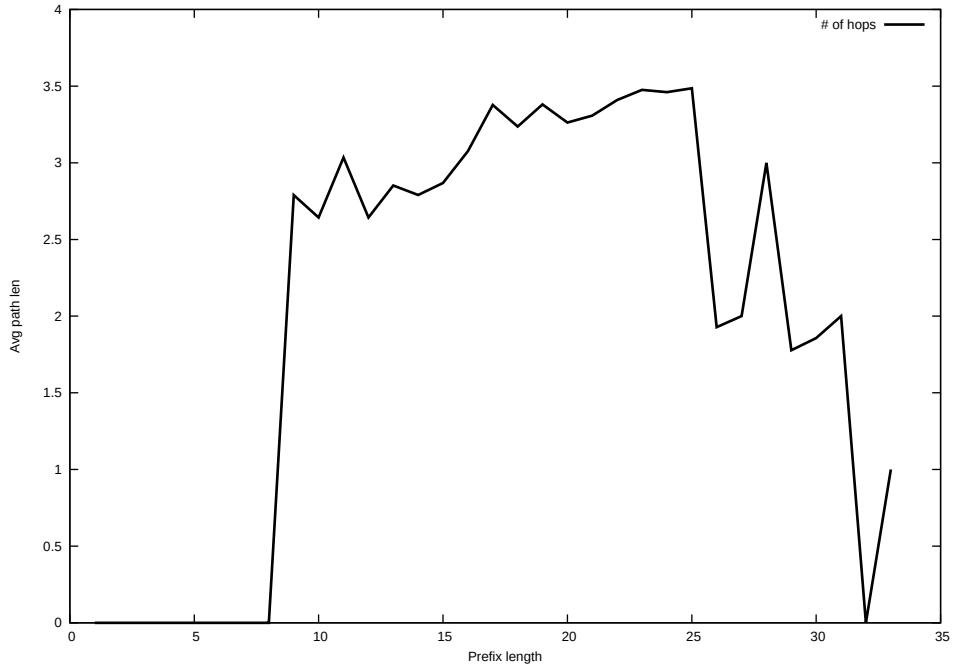
2012-09-16



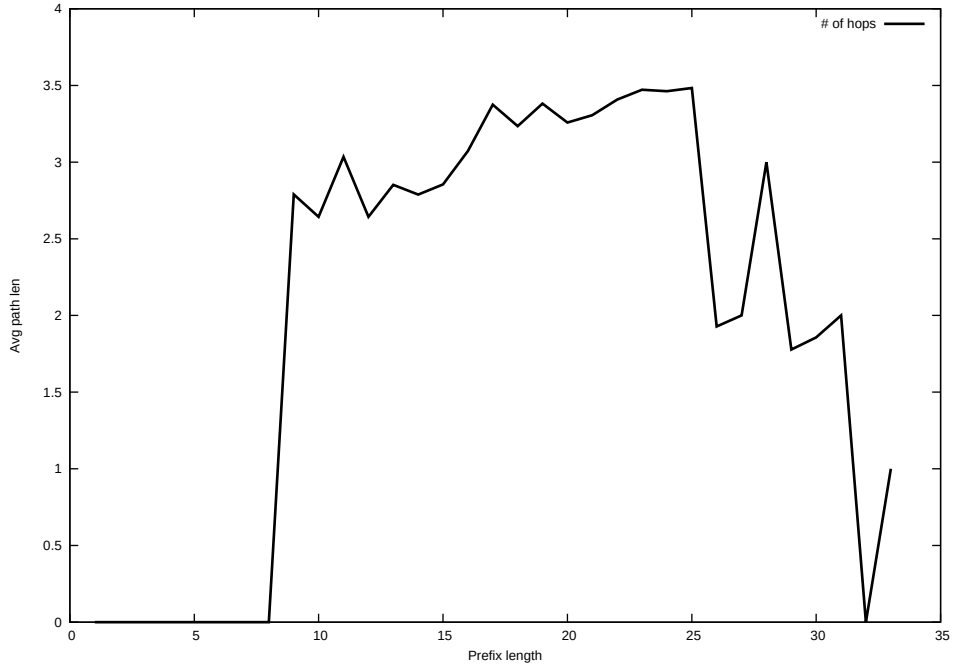
2012-09-17



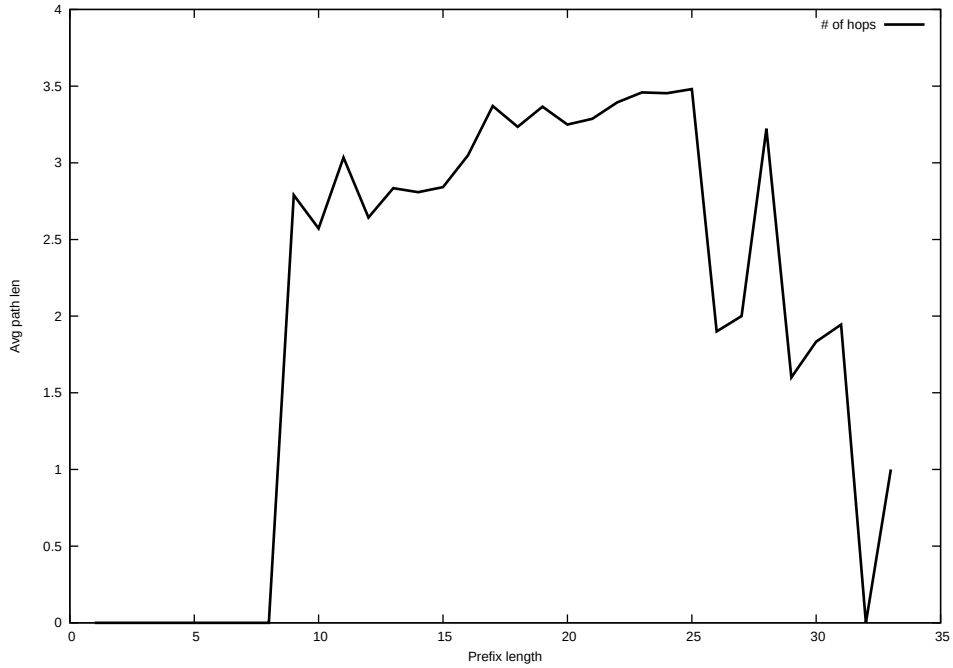
2012-09-18



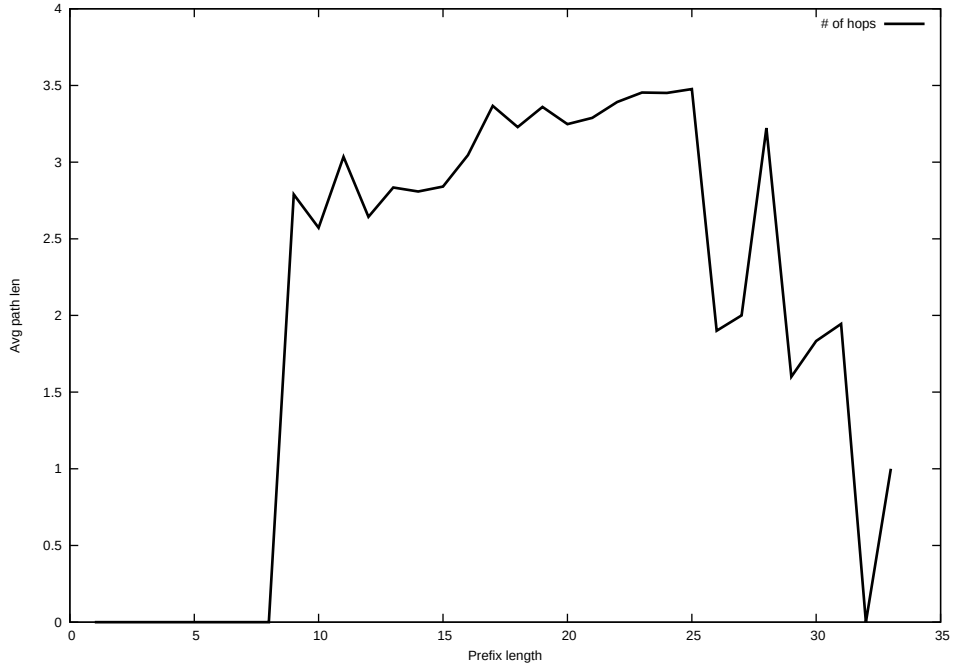
2012-09-19



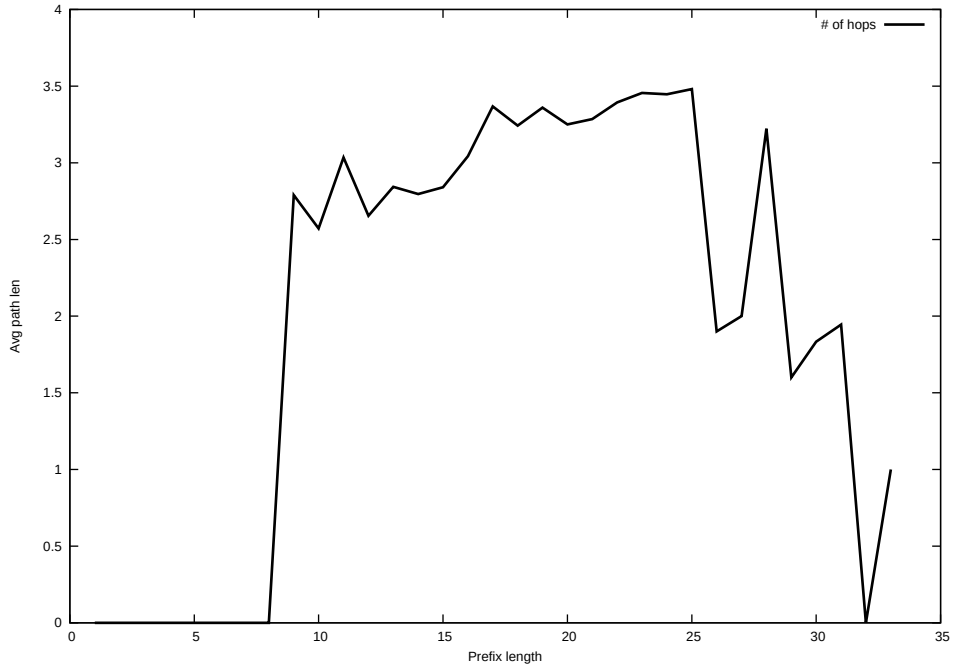
2012-09-20



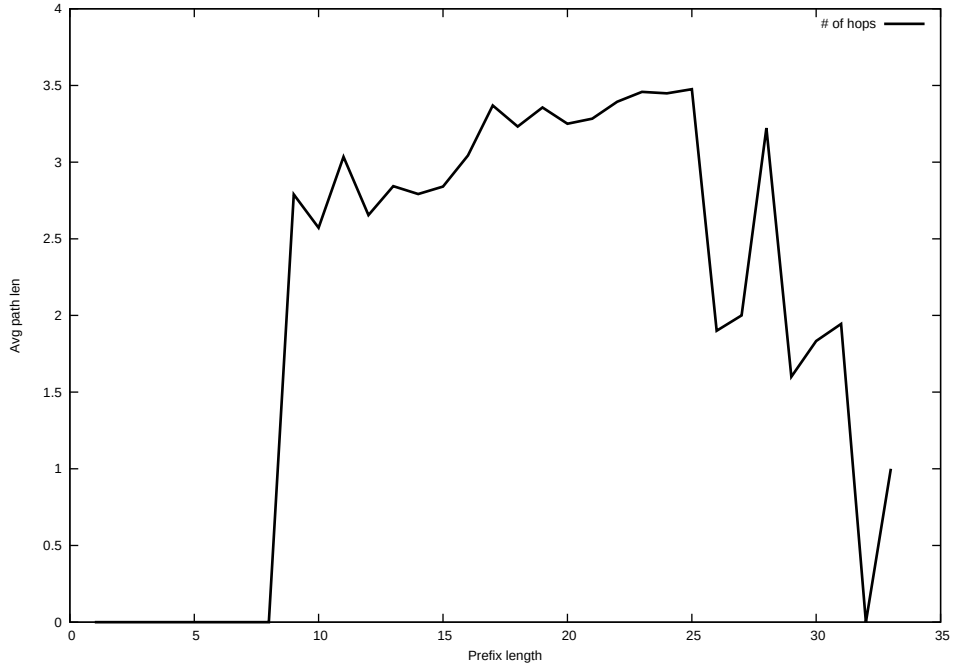
2012-09-21



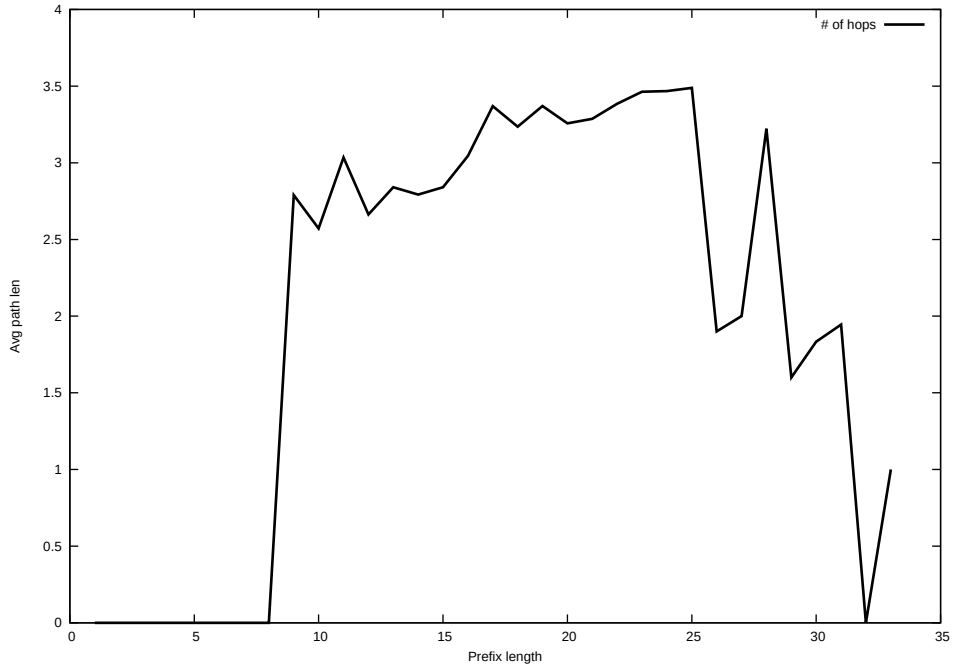
2012-09-22



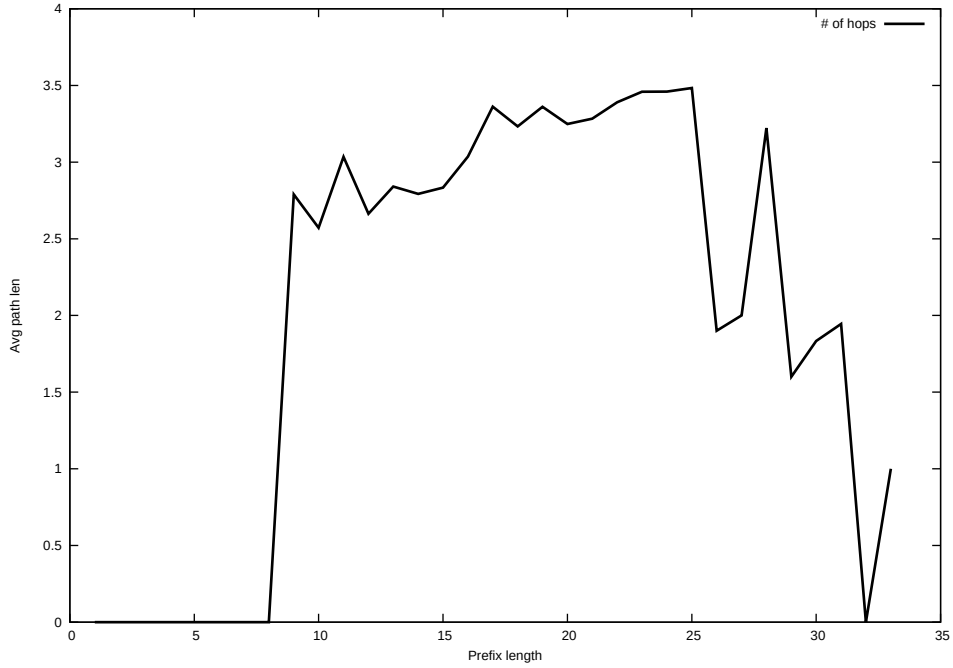
2012-09-23



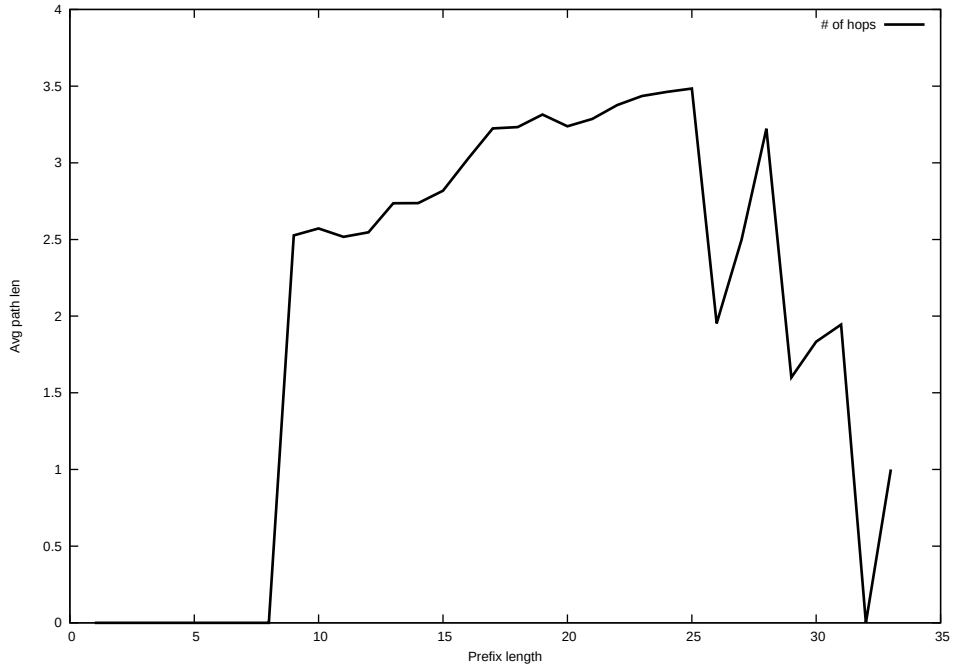
2012-09-24



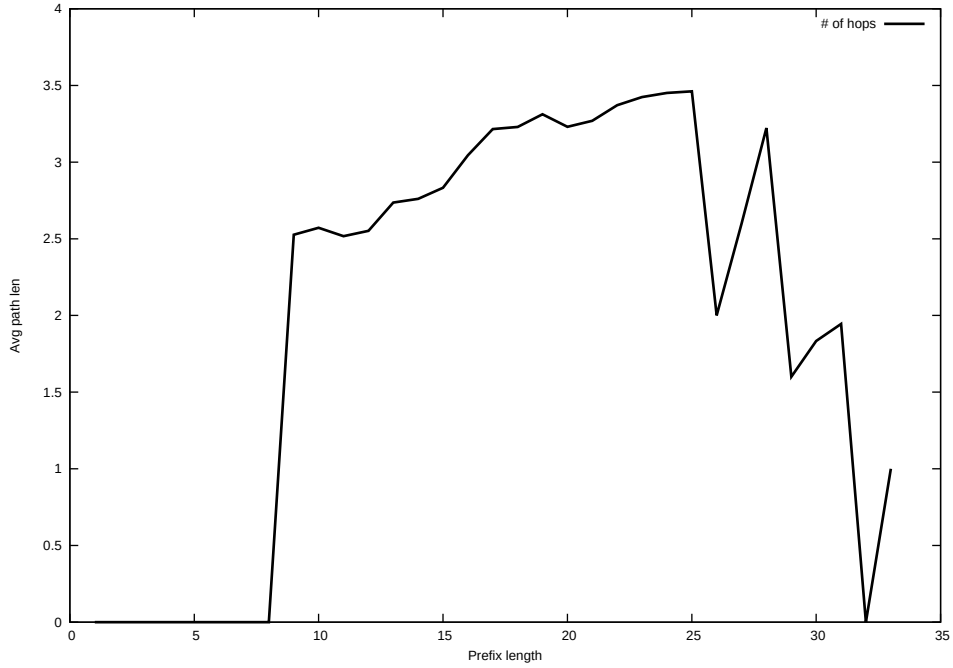
2012-09-25



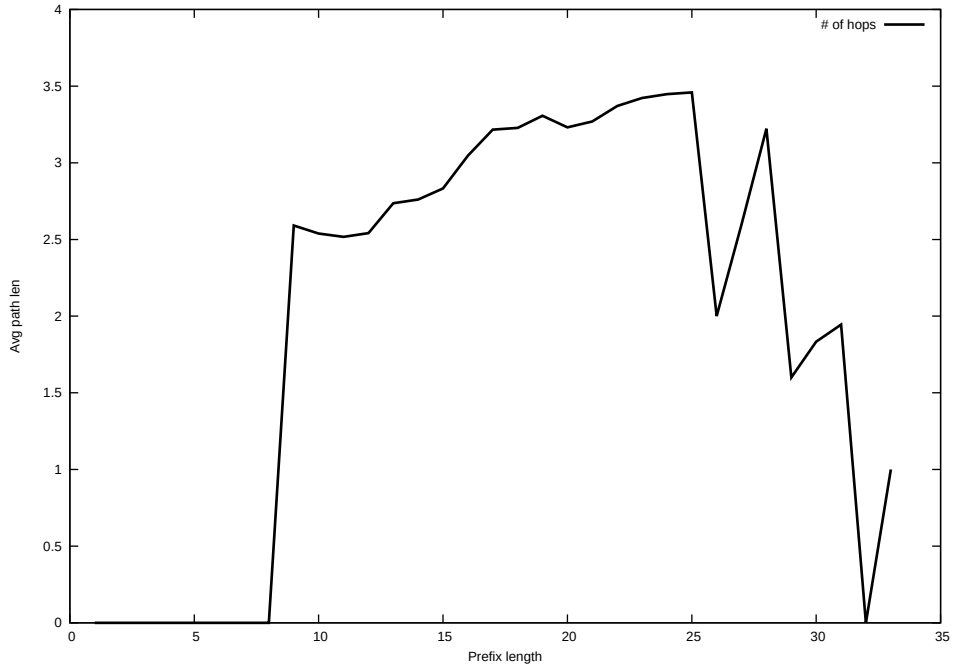
2012-09-26



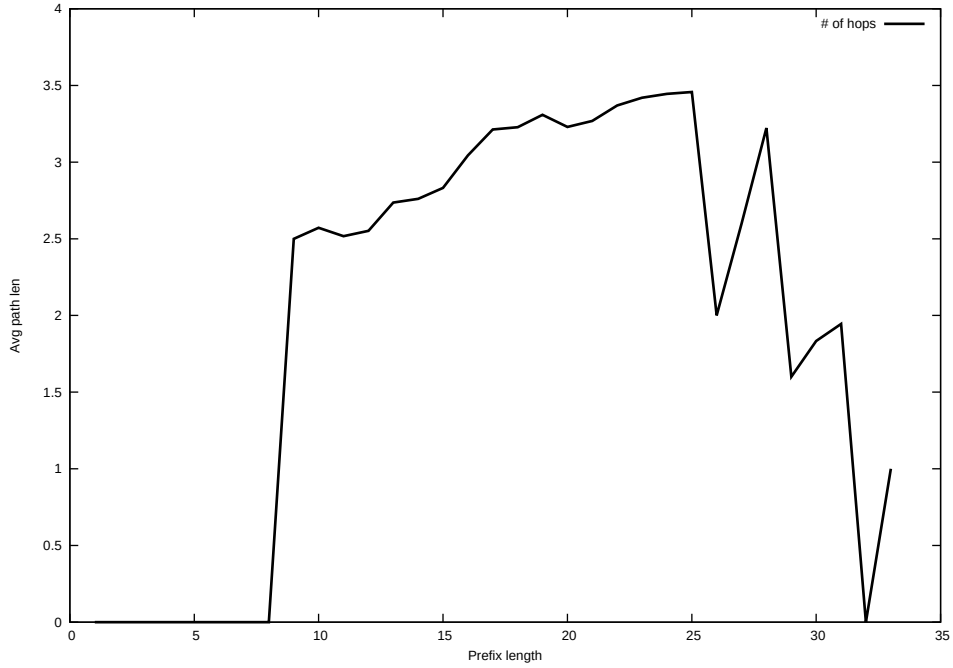
2012-09-27



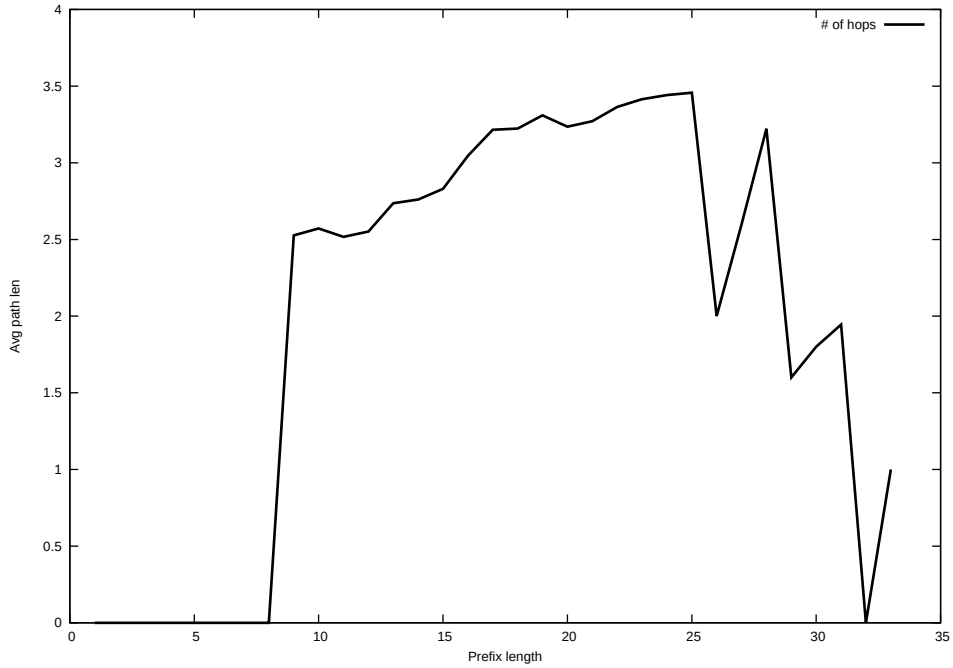
2012-09-28



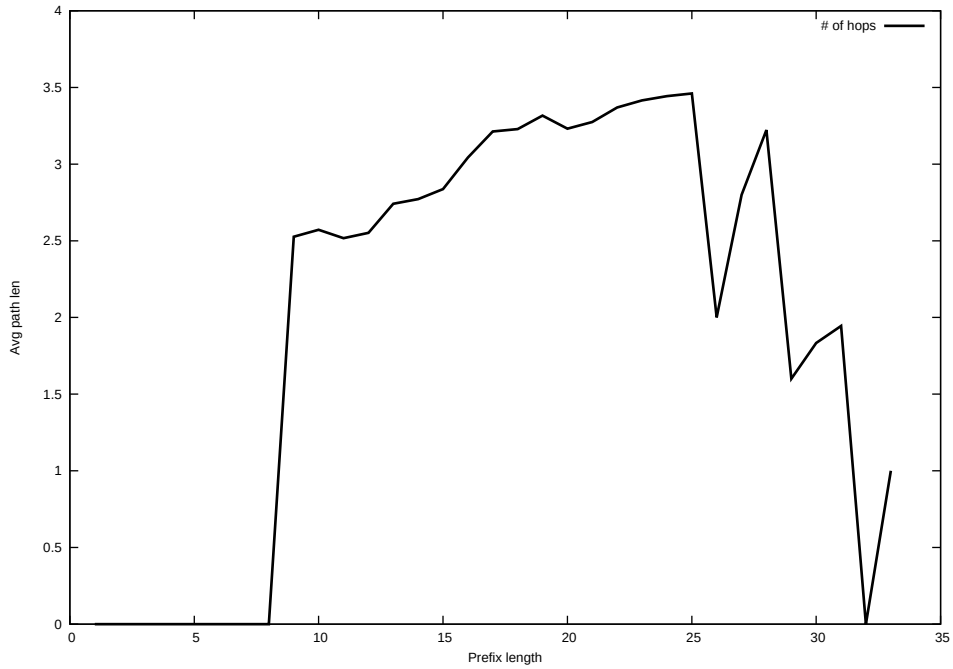
2012-09-29



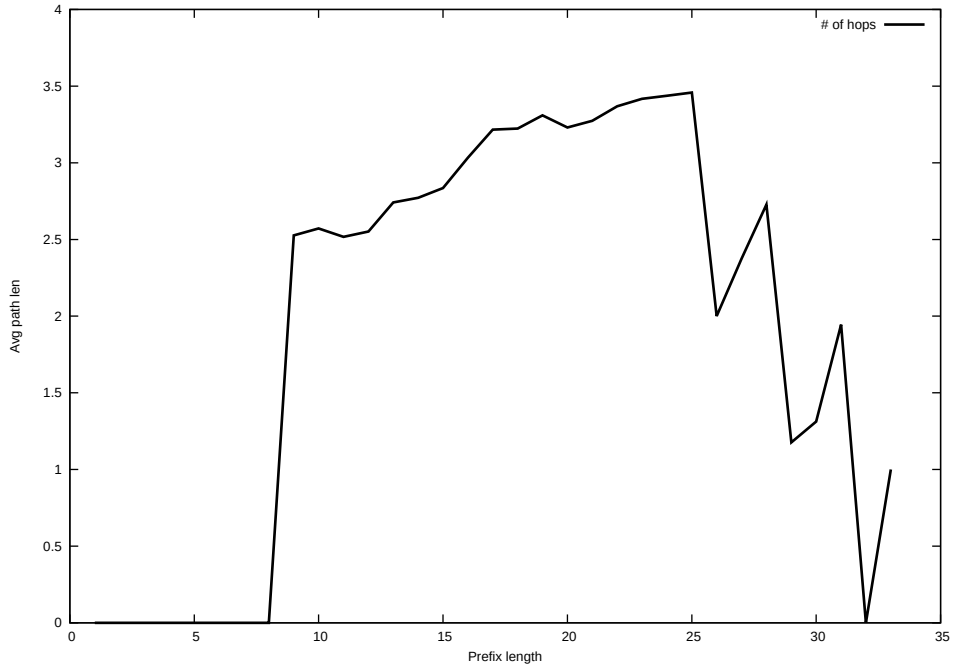
2012-09-30



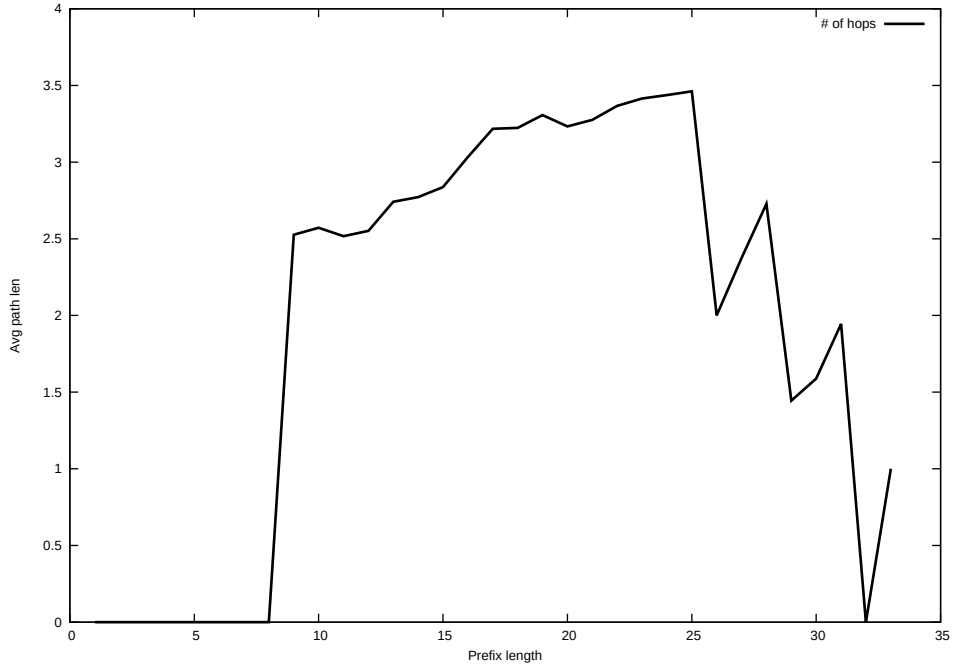
2012-10-01



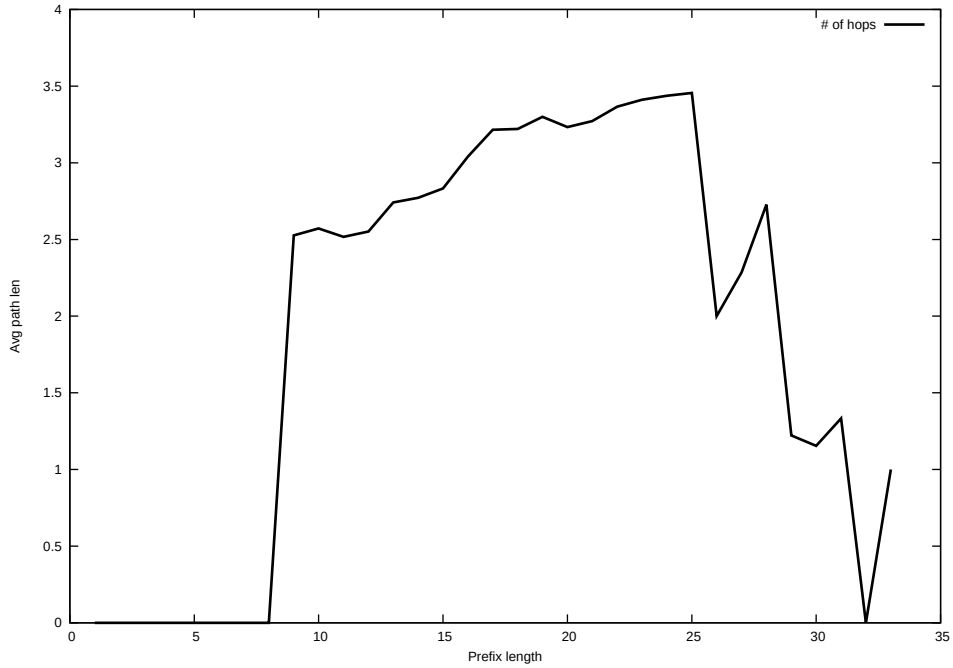
2012-10-02



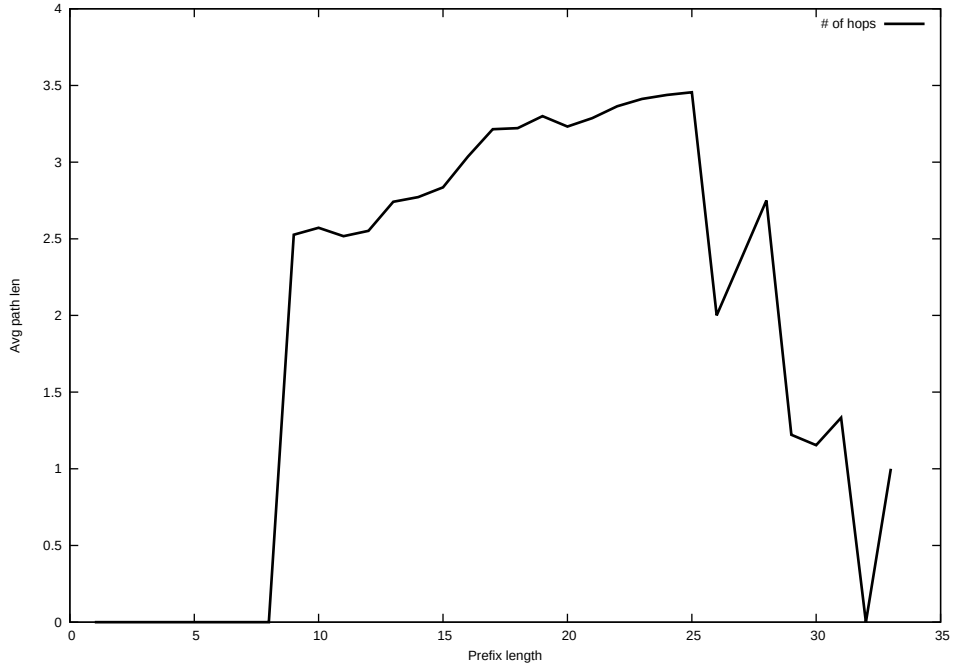
2012-10-03



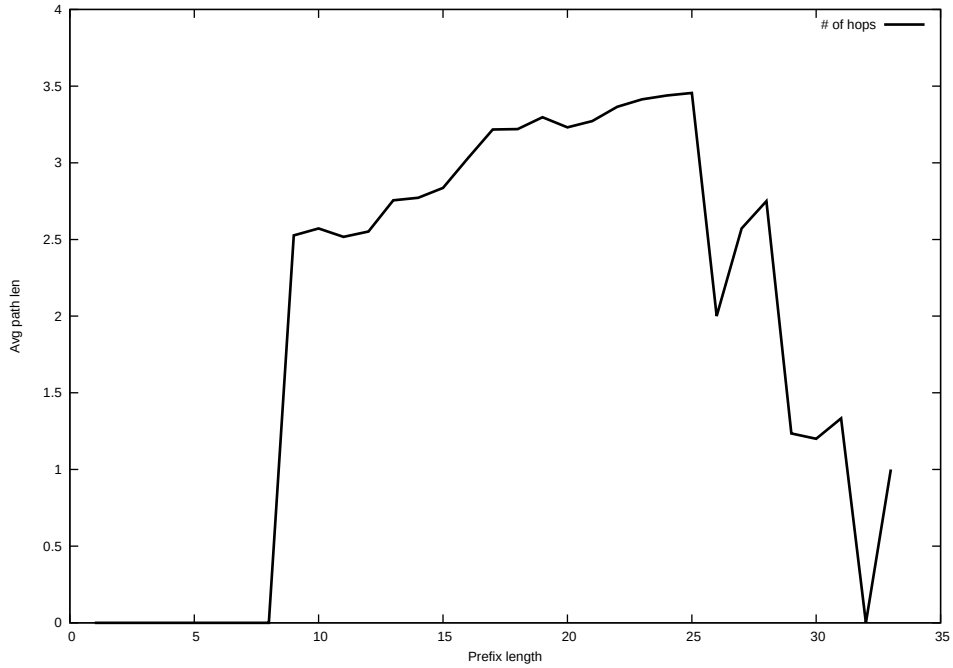
2012-10-04



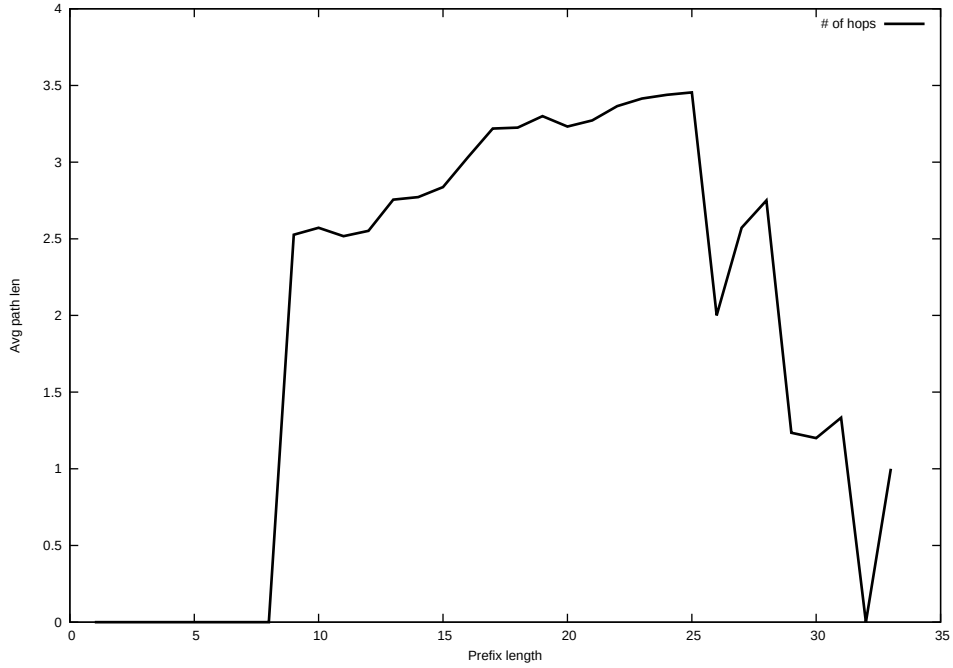
2012-10-05



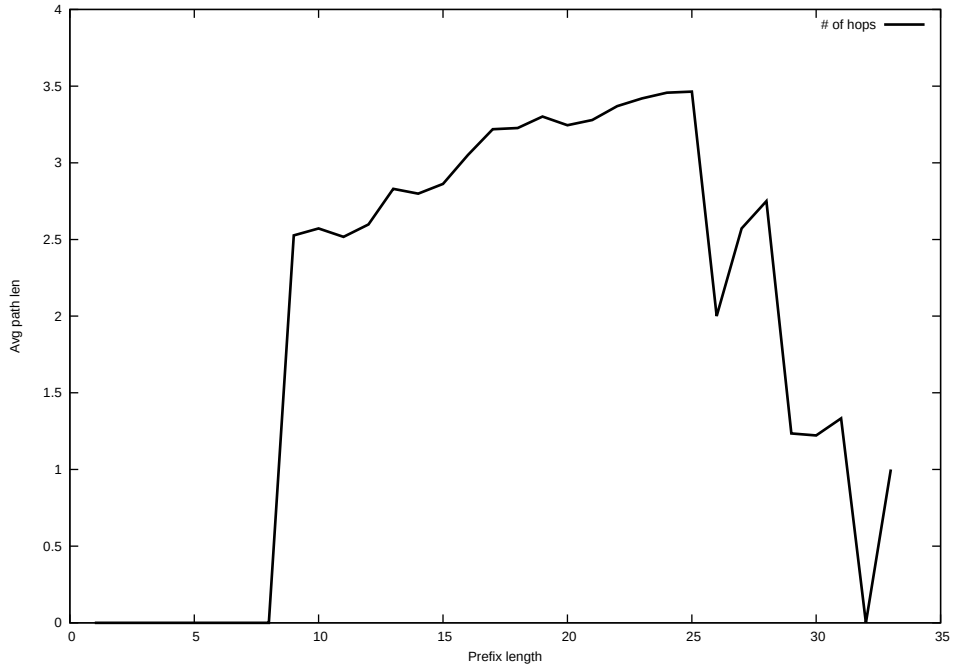
2012-10-06



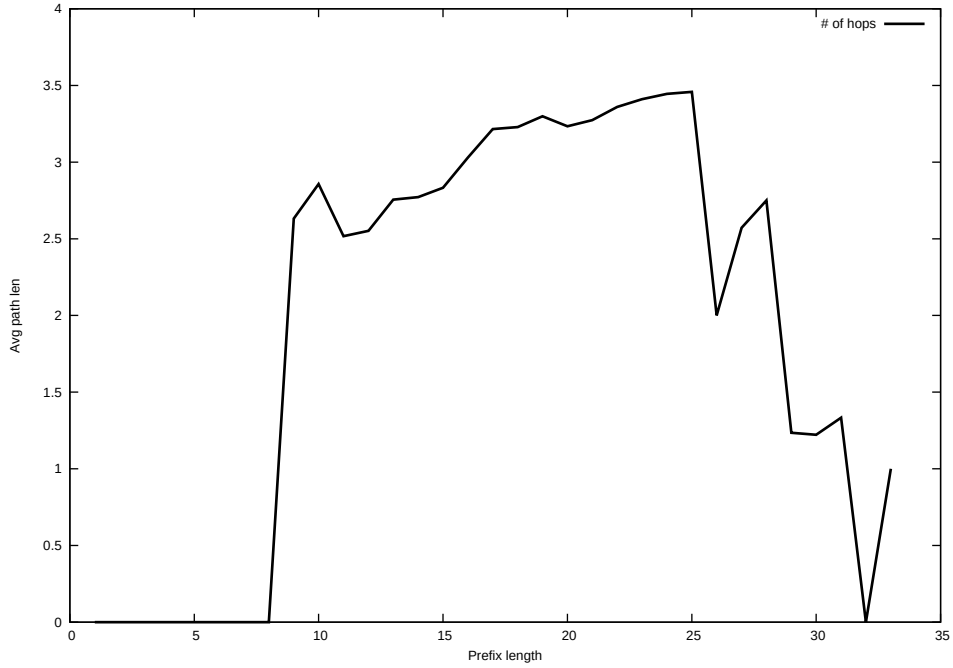
2012-10-07



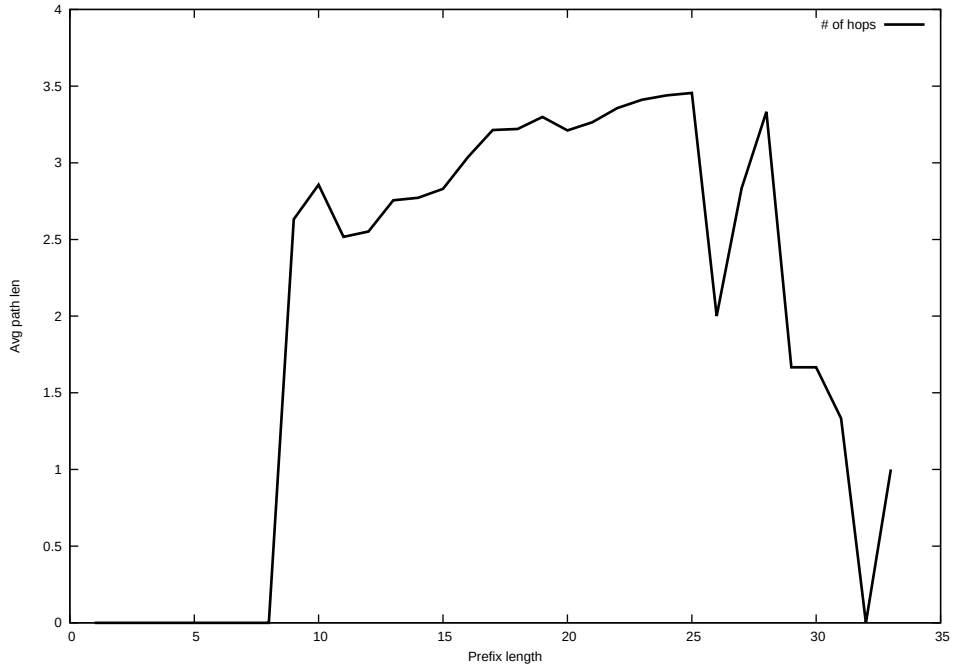
2012-10-08



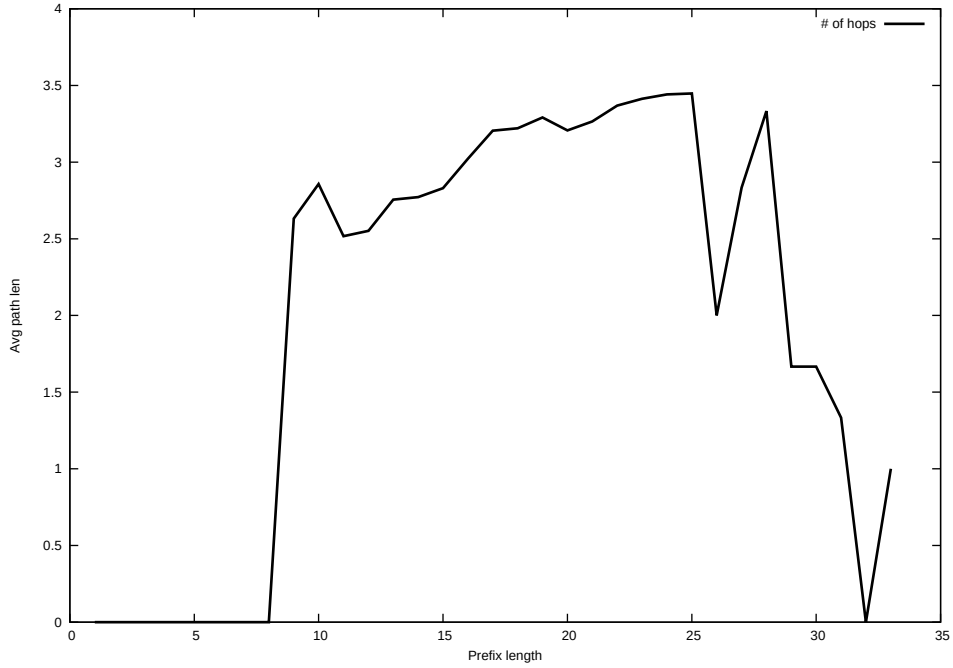
2012-10-09



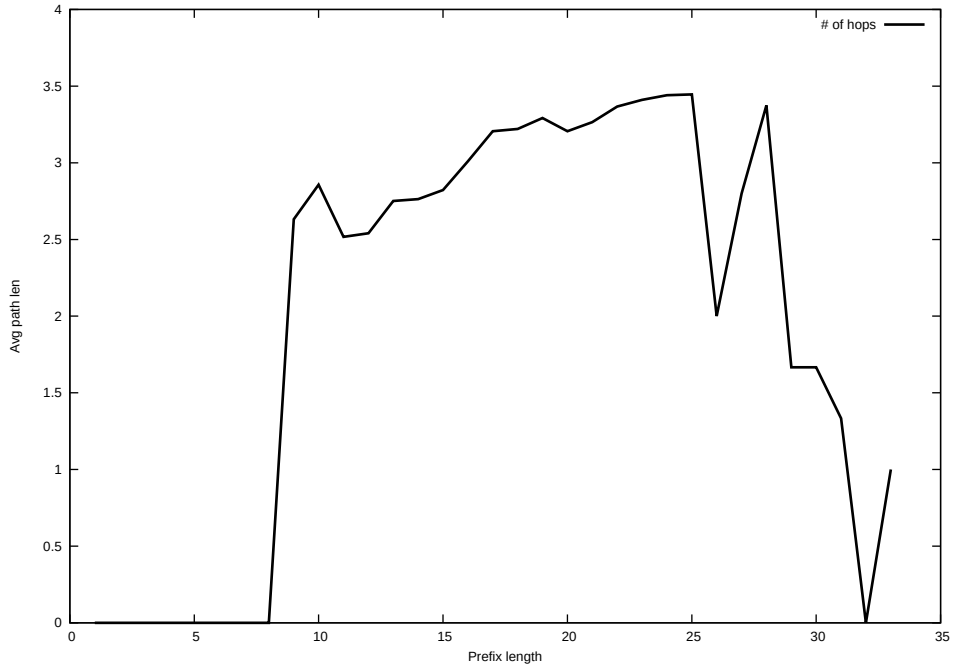
2012-10-10



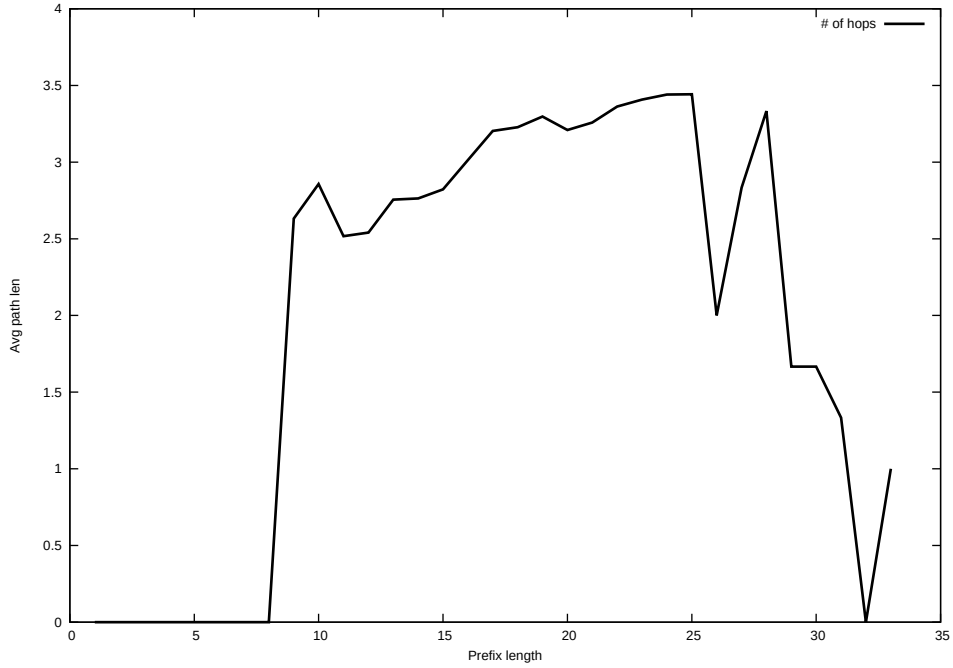
2012-10-11



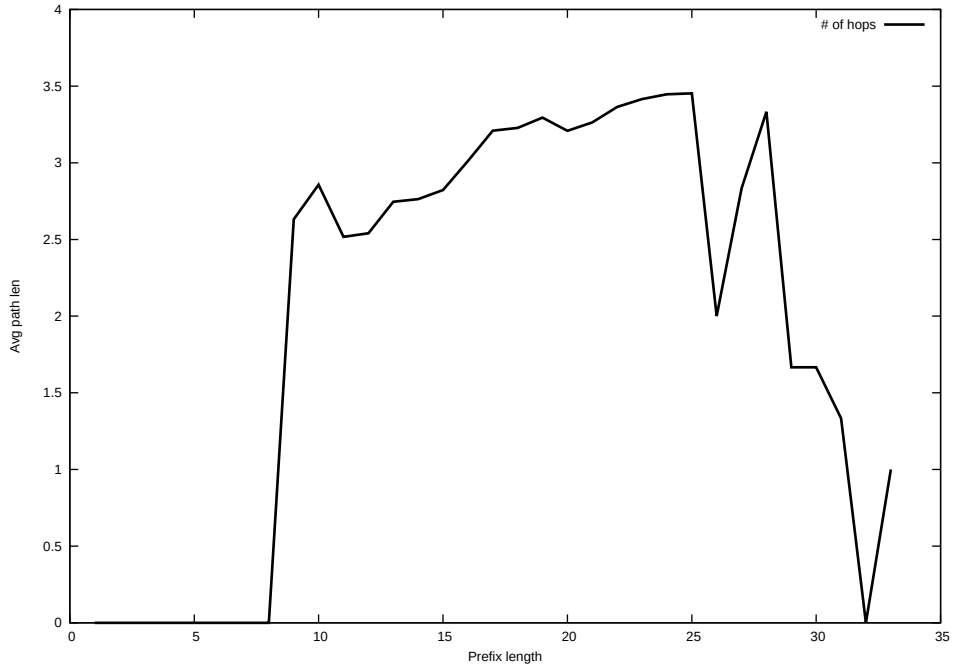
2012-10-12



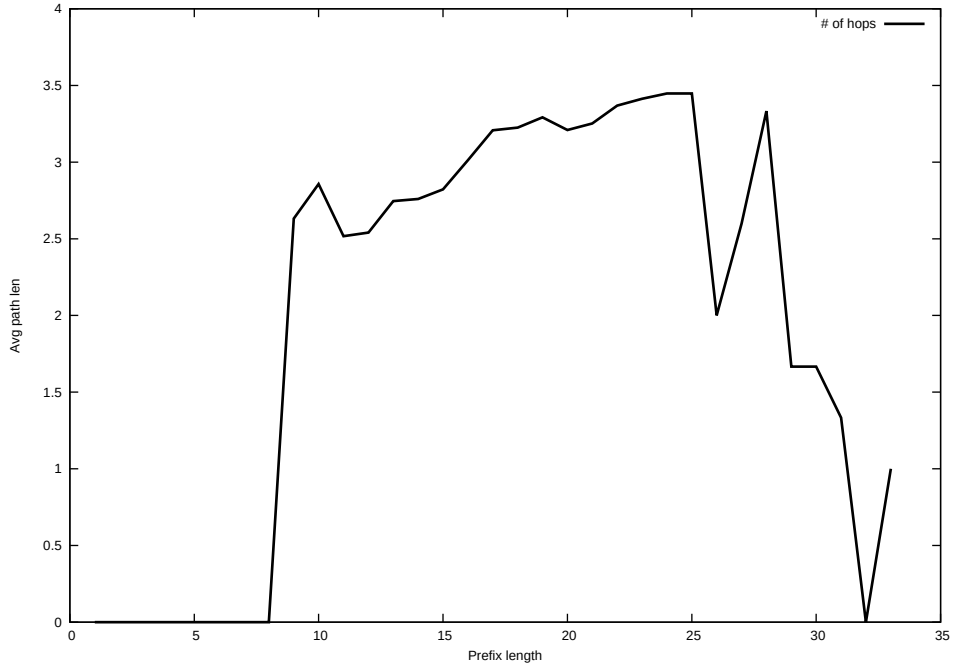
2012-10-13



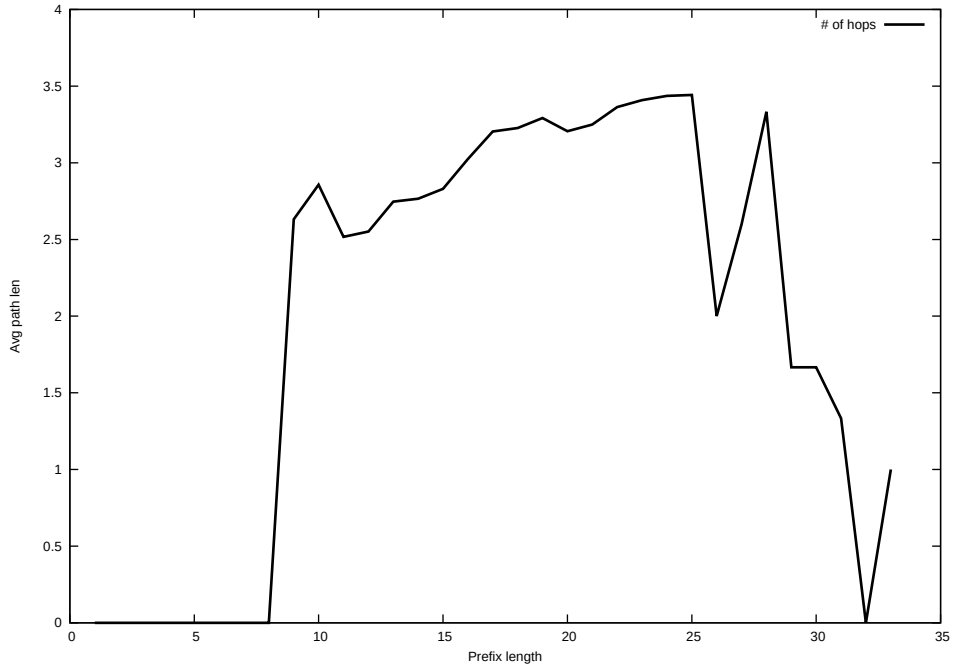
2012-10-14



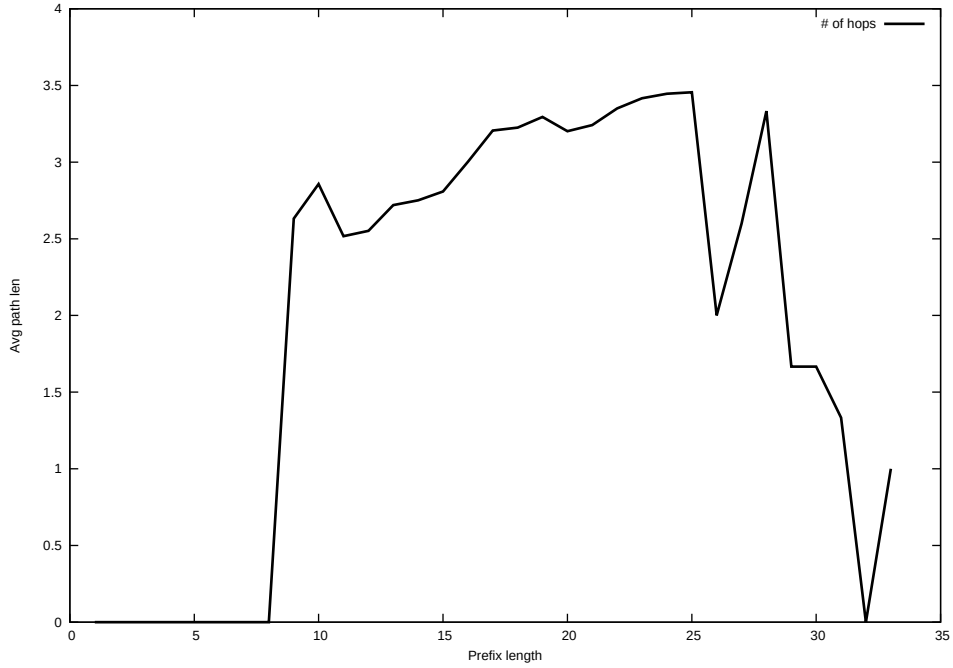
2012-10-15



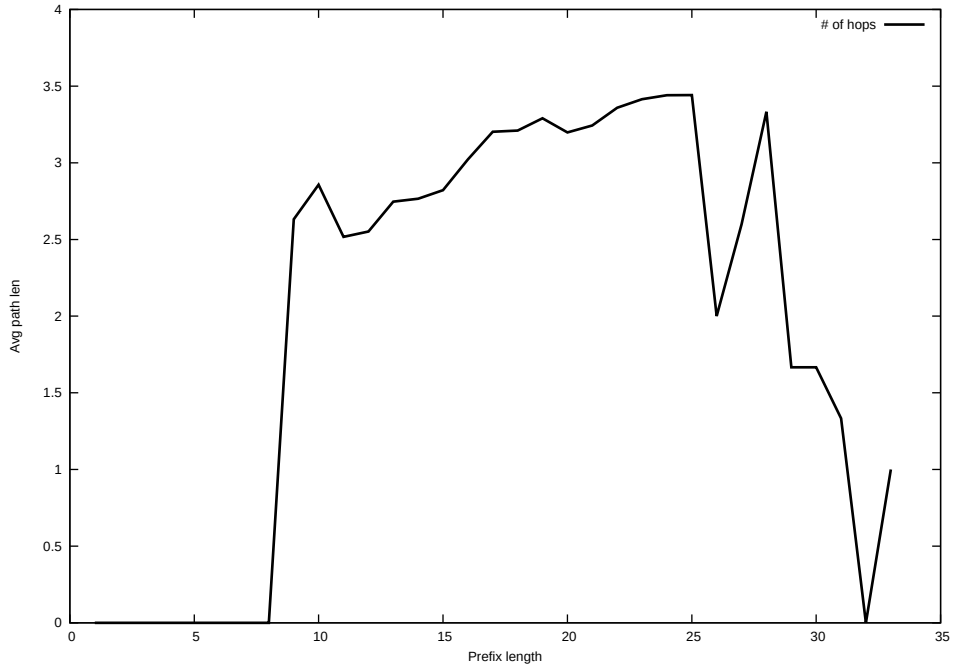
2012-10-16



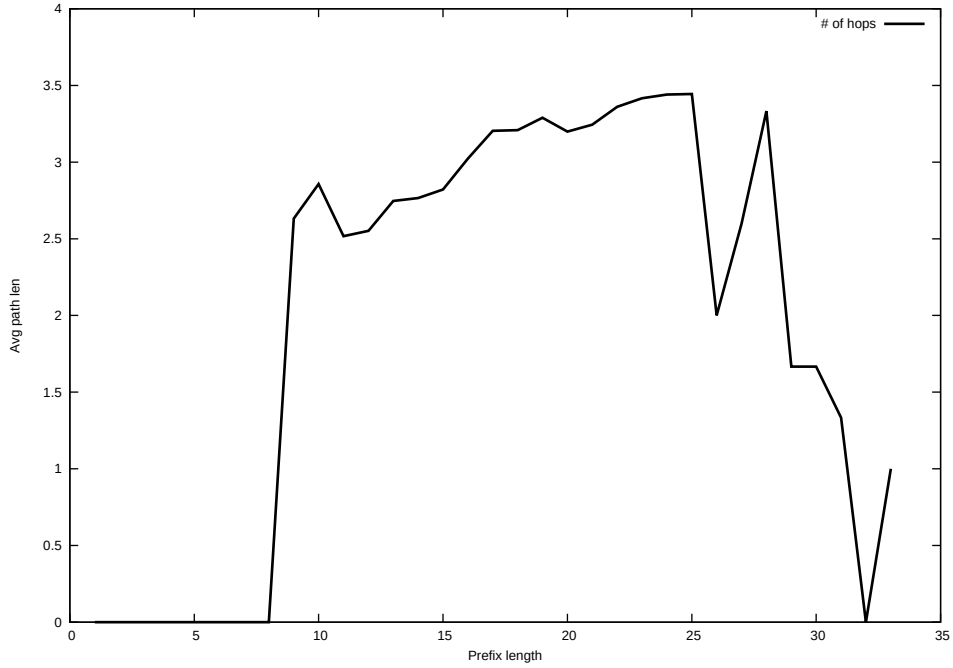
2012-10-18



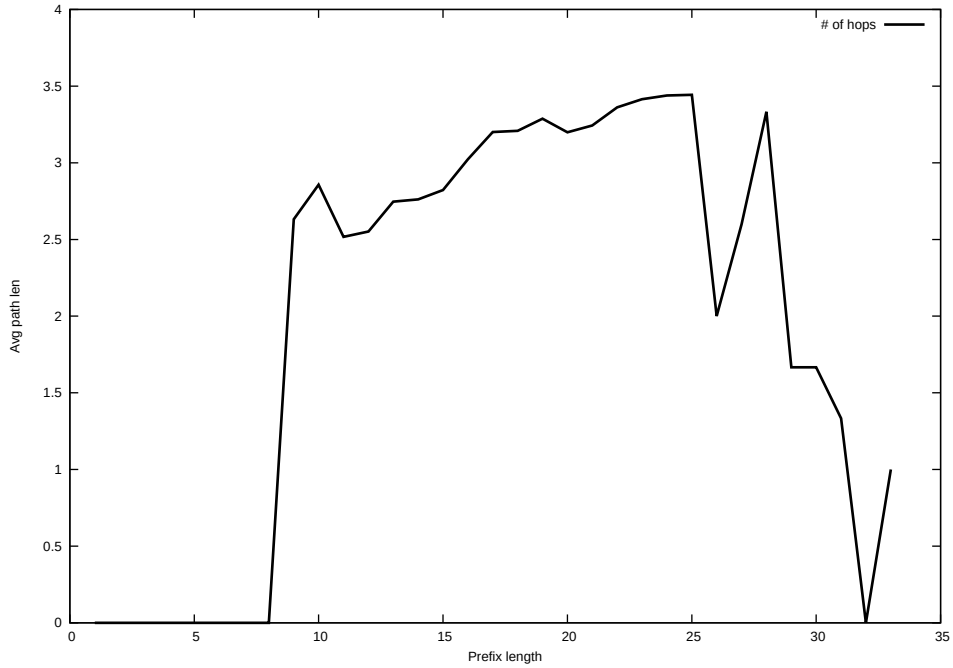
2012-10-19



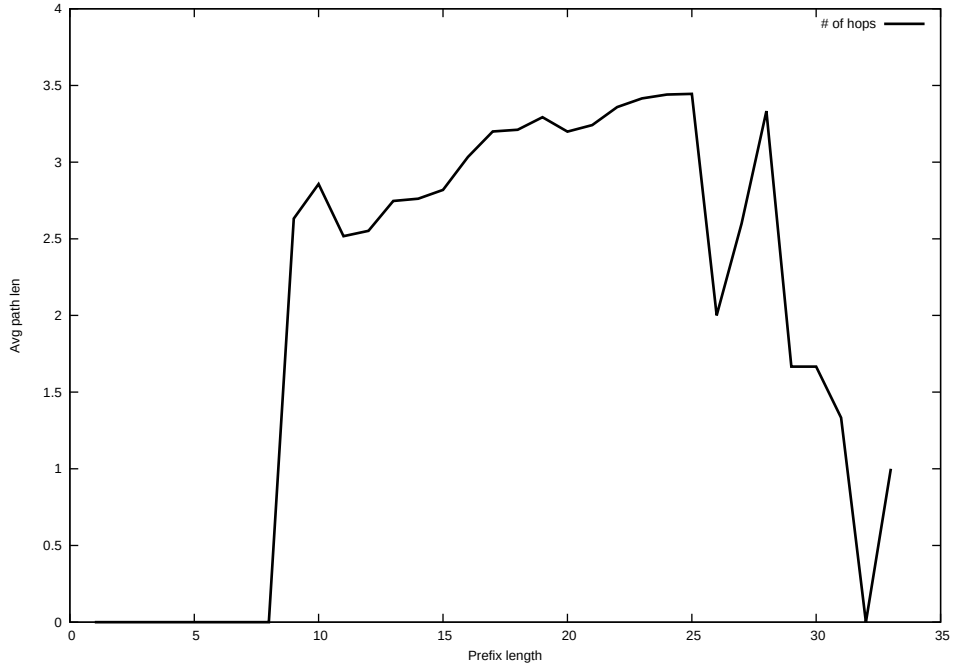
2012-10-20



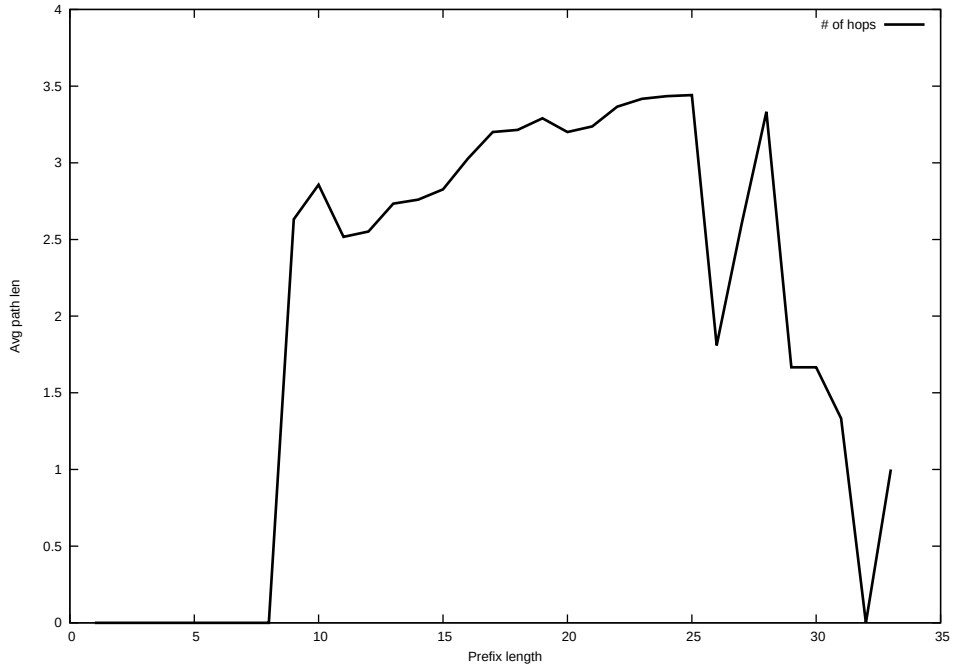
2012-10-21



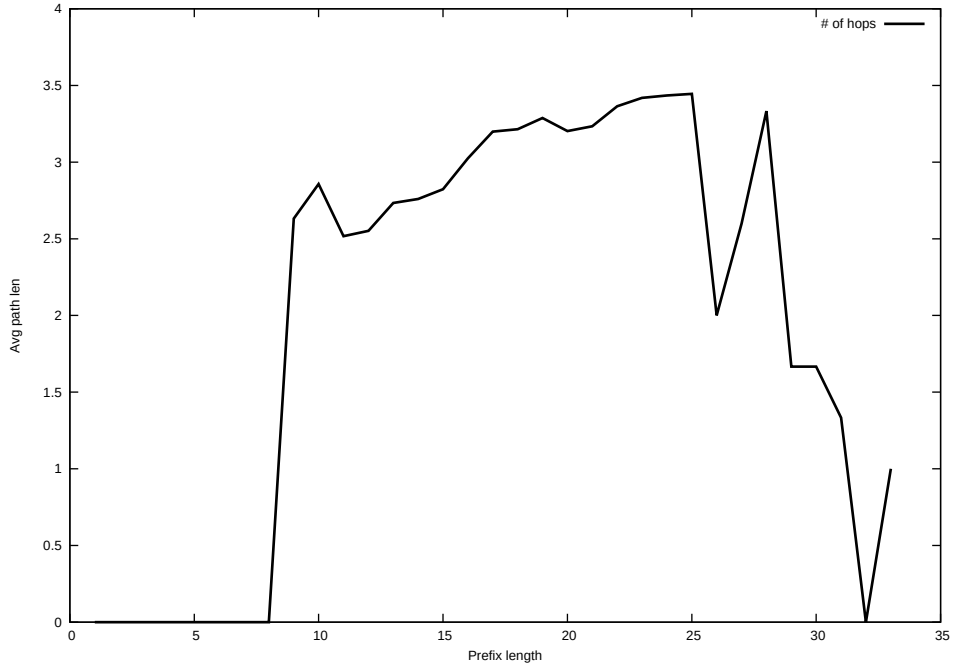
2012-10-22



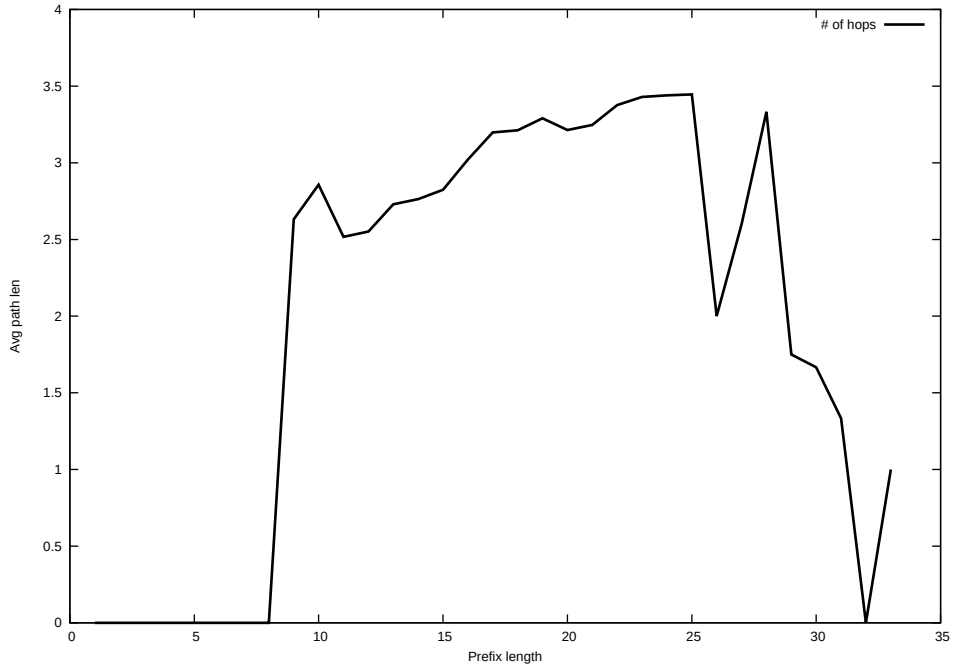
2012-10-23



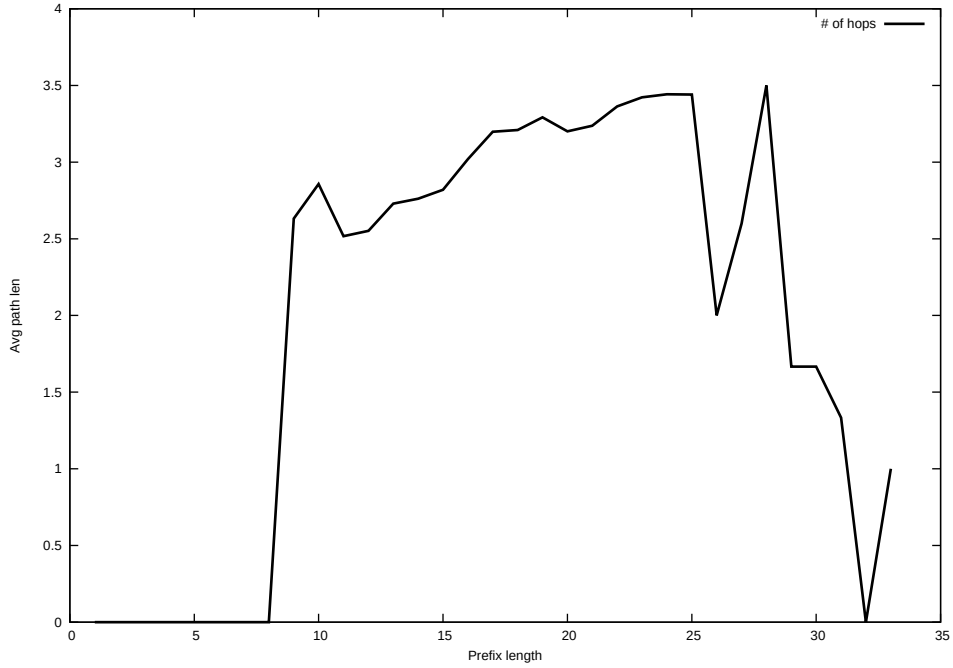
2012-10-24



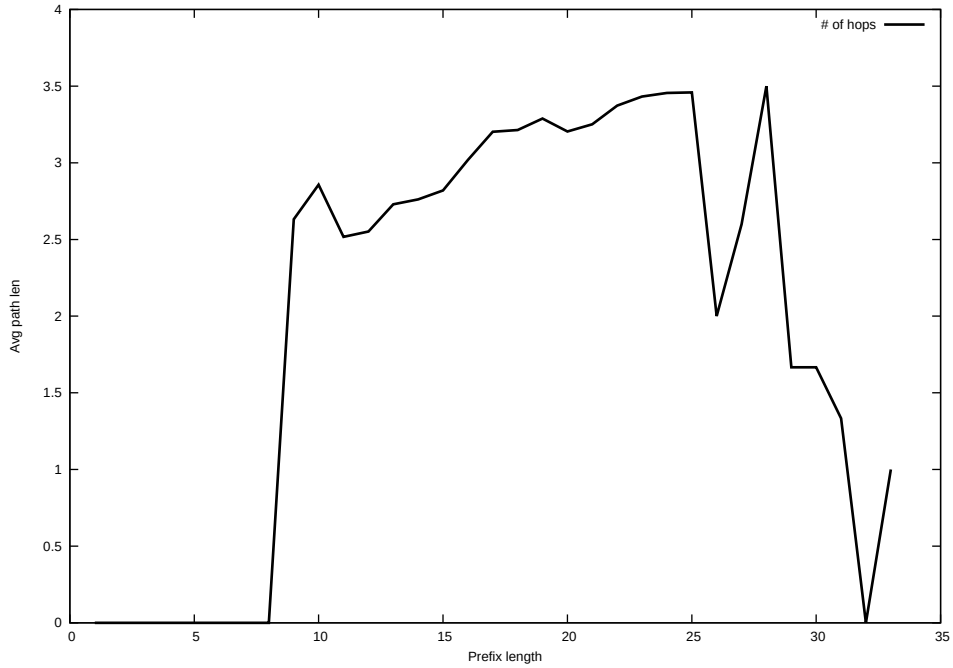
2012-10-25



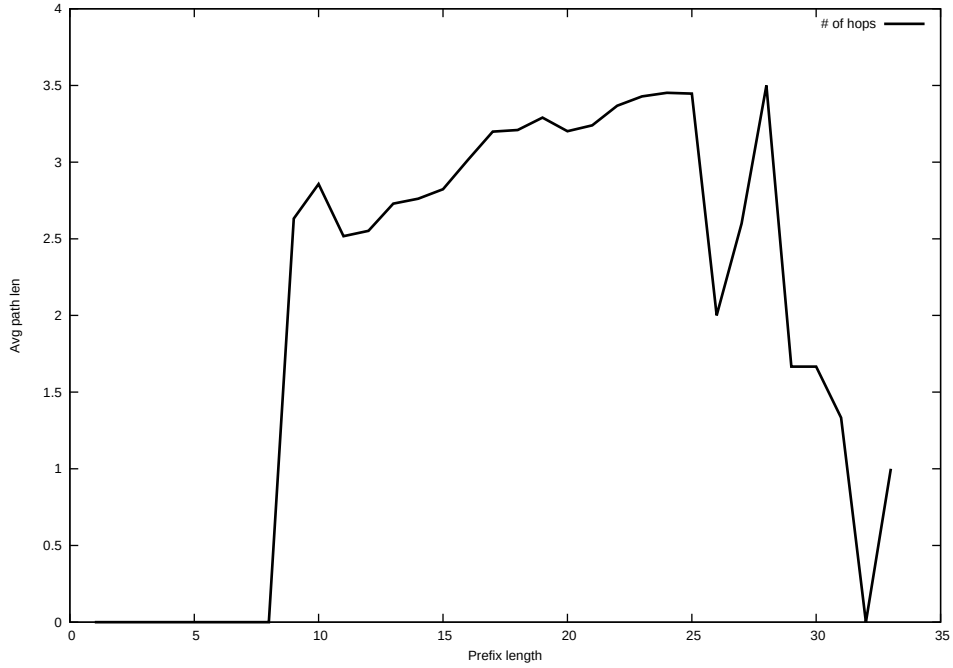
2012-10-26



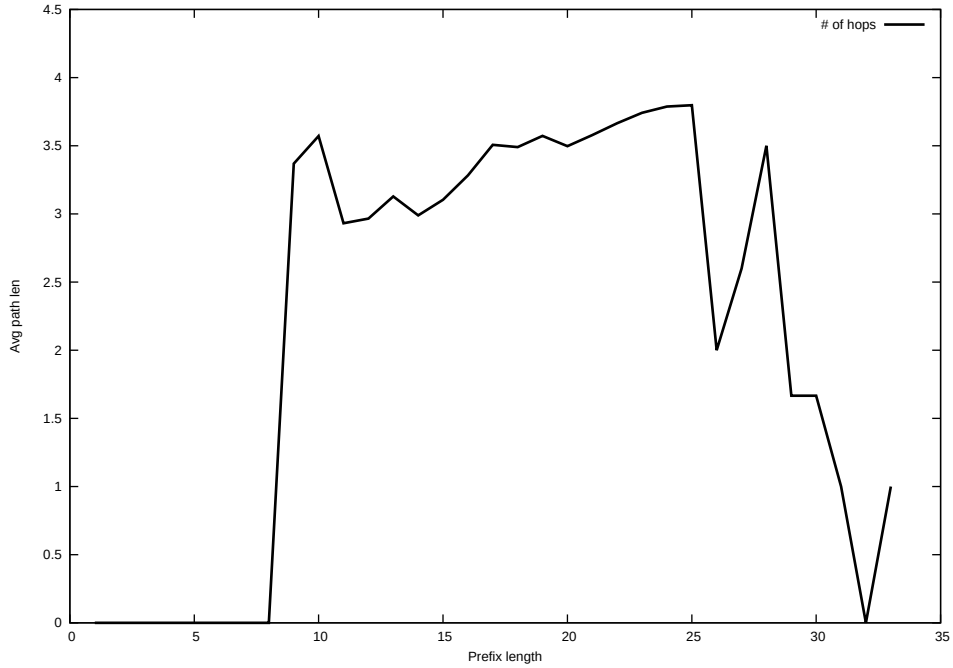
2012-10-27



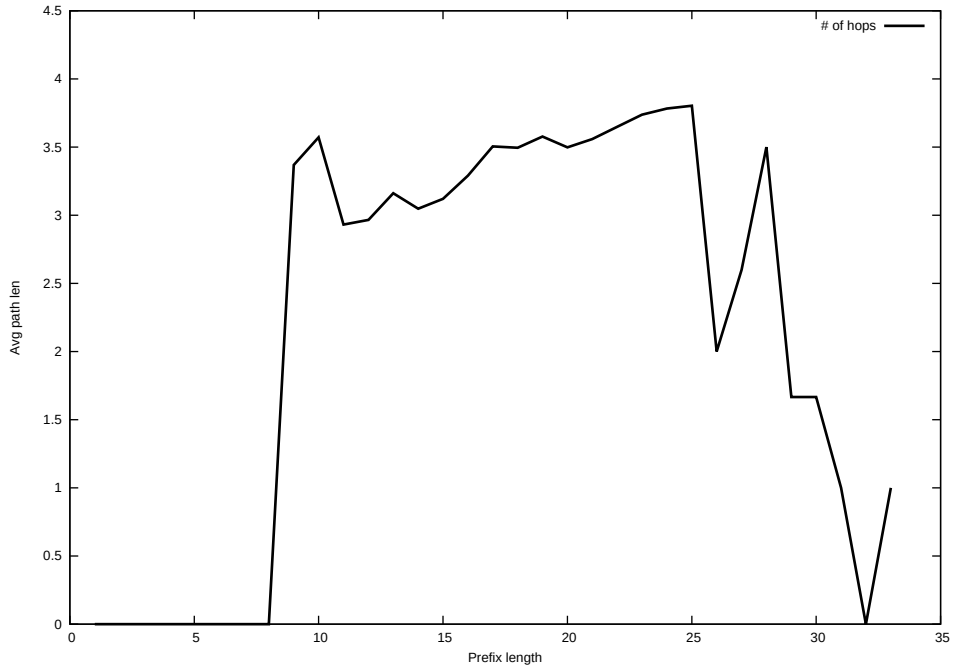
2012-10-28



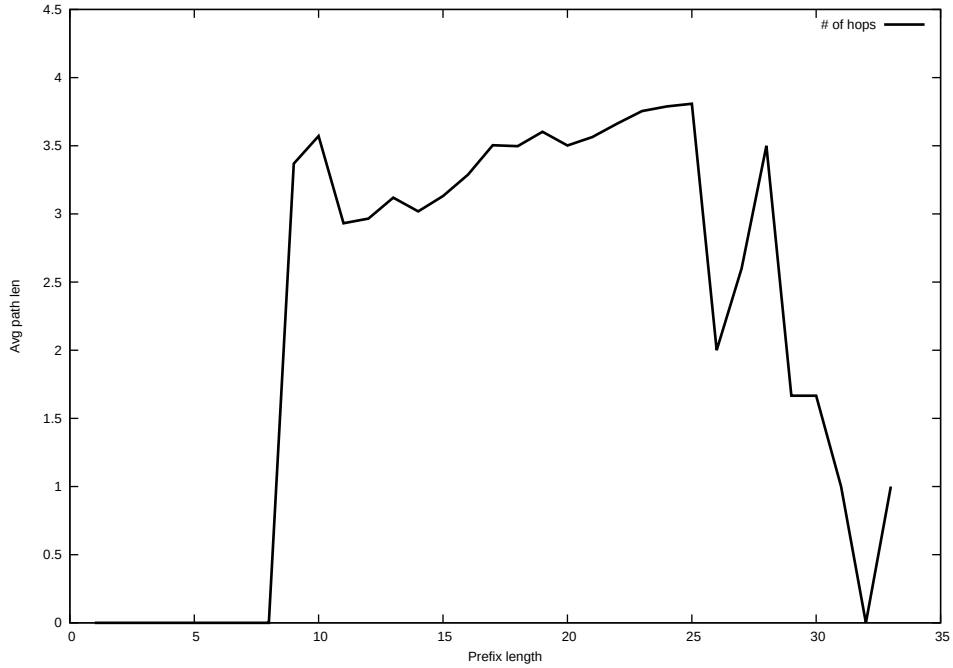
2012-10-29



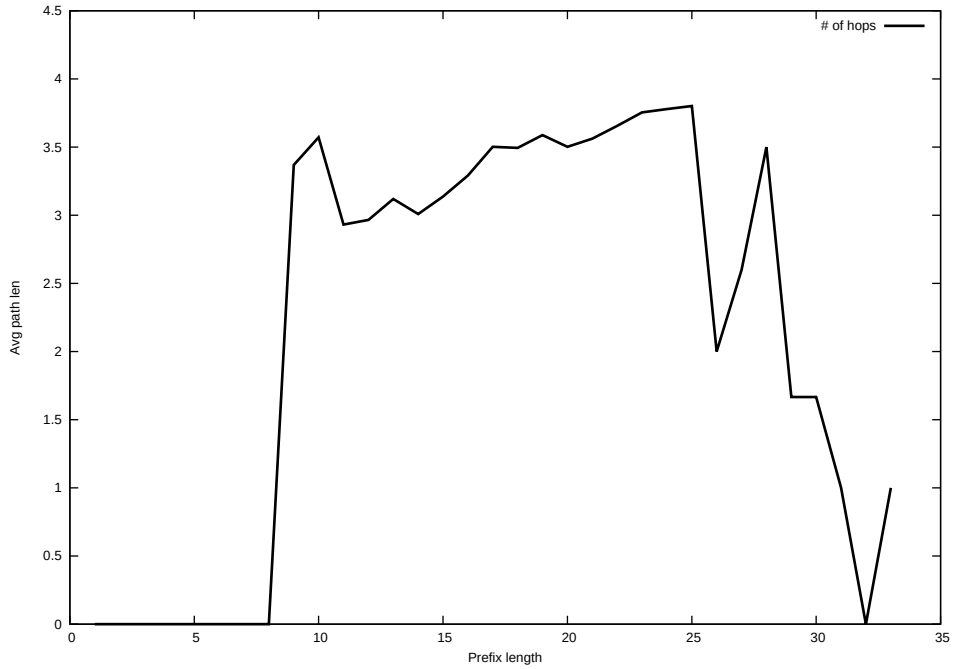
2012-10-30



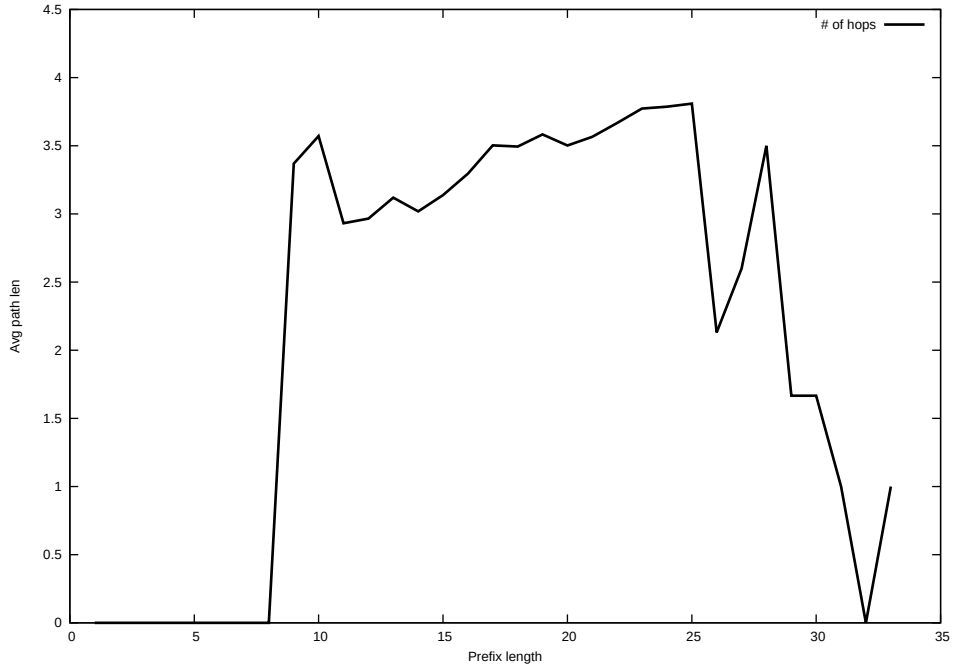
2012-10-31



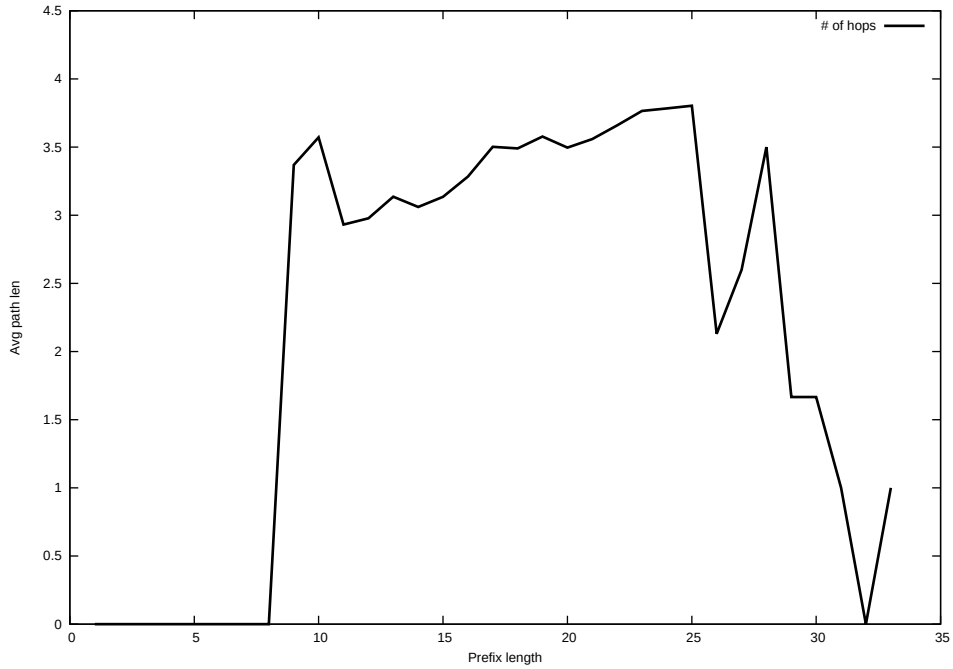
2012-11-01



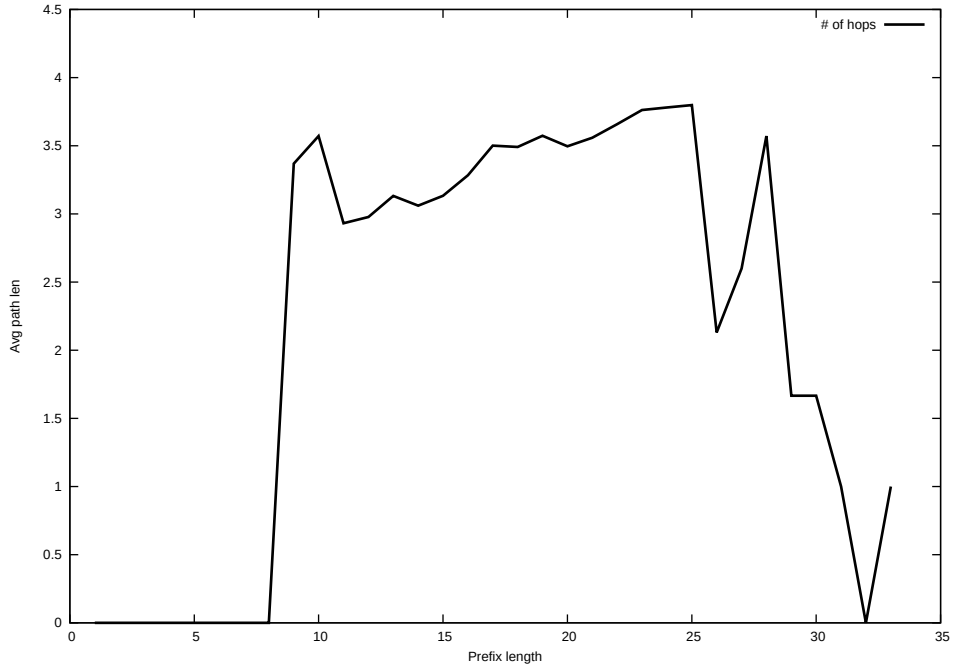
2012-11-02



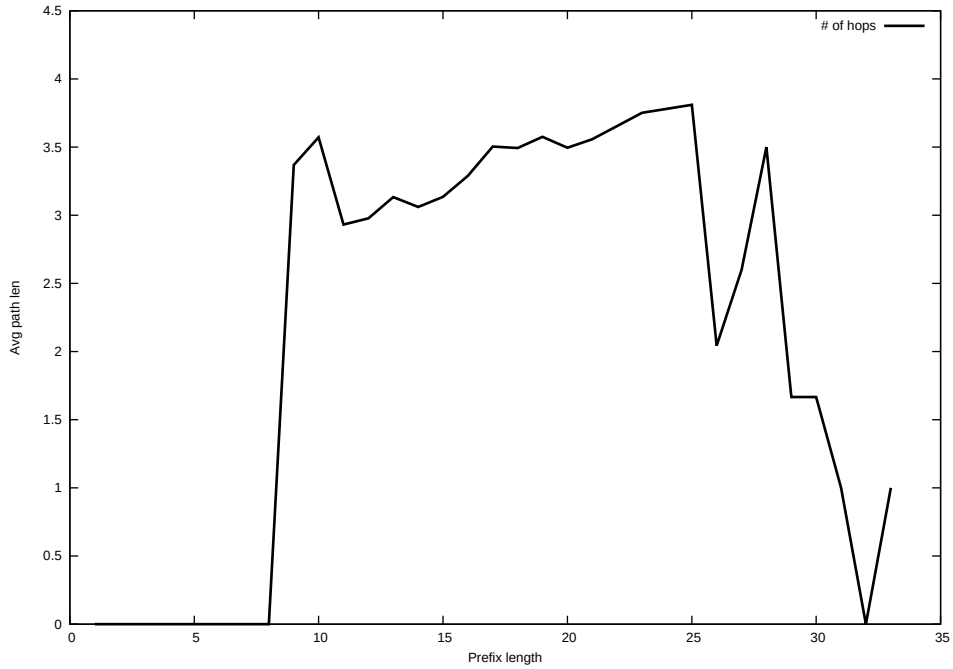
2012-11-03



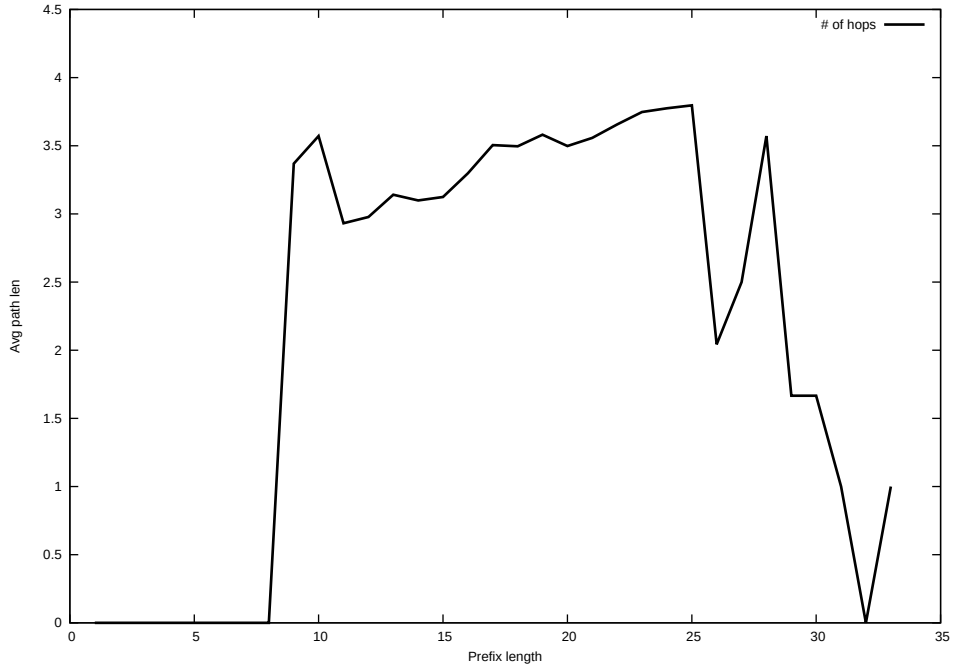
2012-11-04



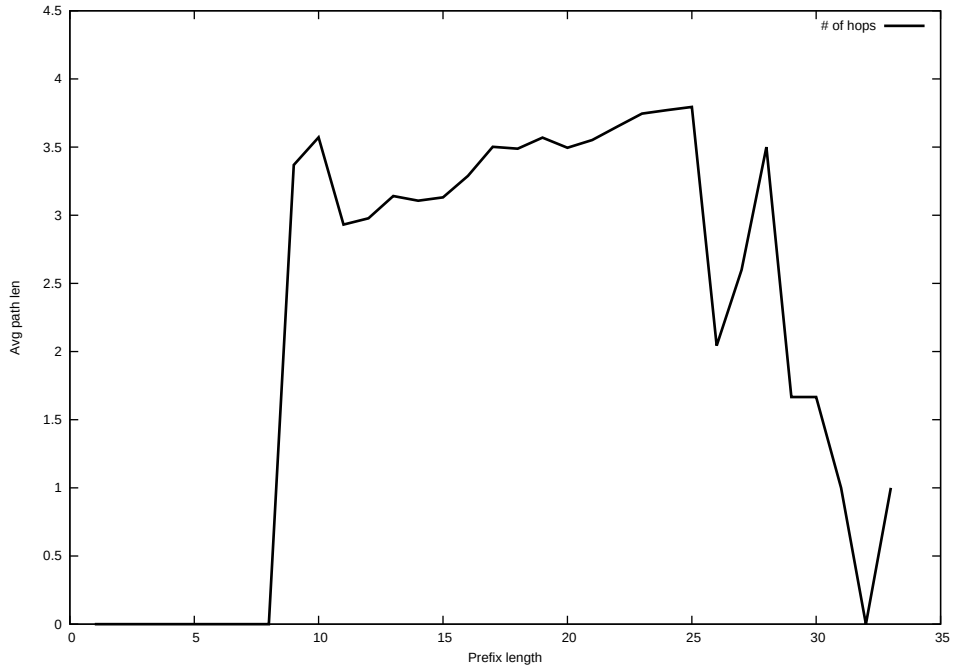
2012-11-05



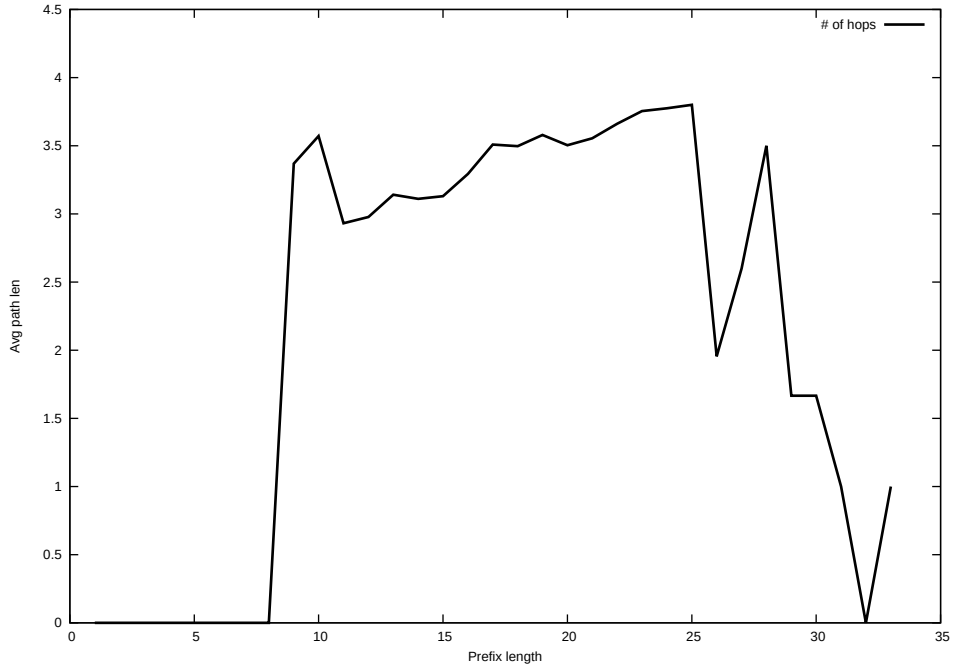
2012-11-06



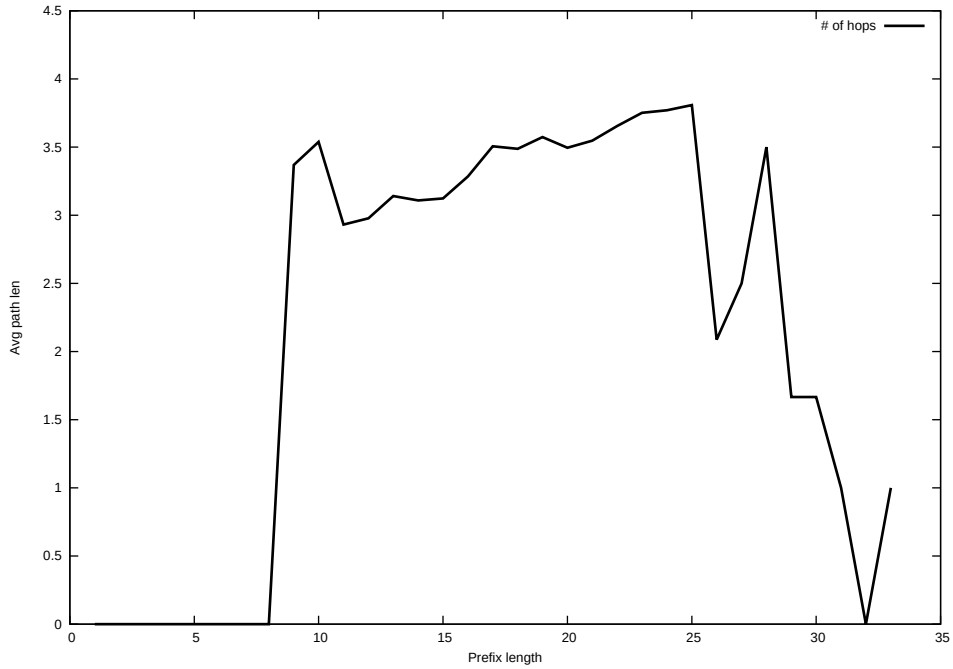
2012-11-07



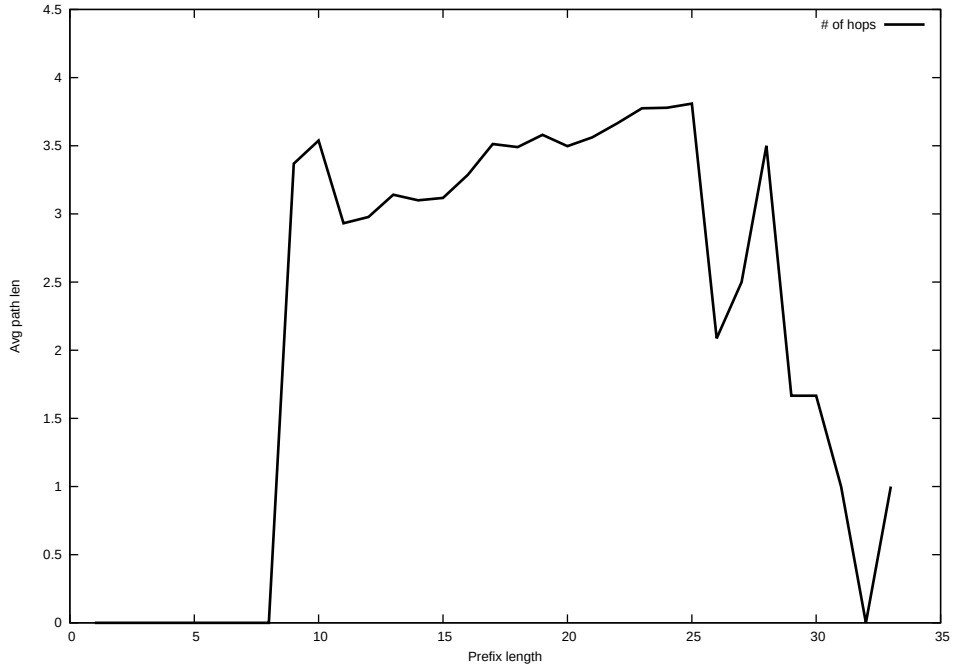
2012-11-08



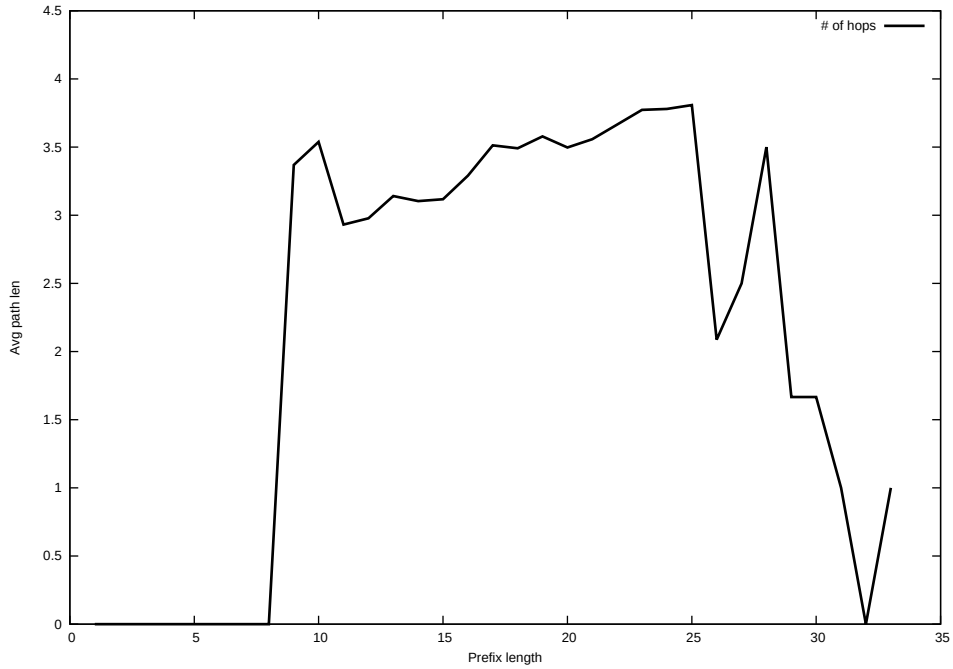
2012-11-09



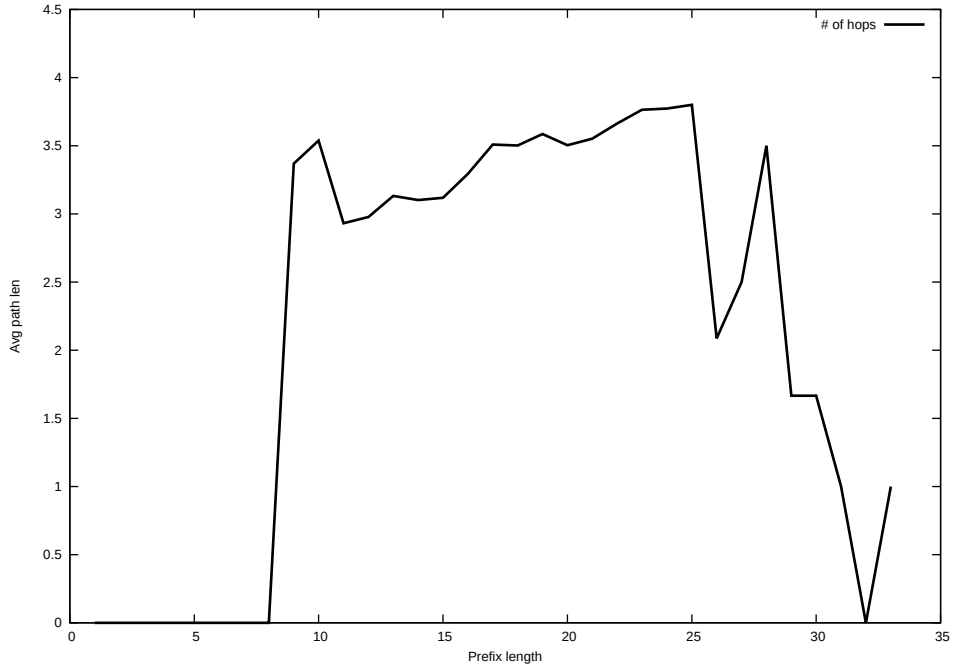
2012-11-10



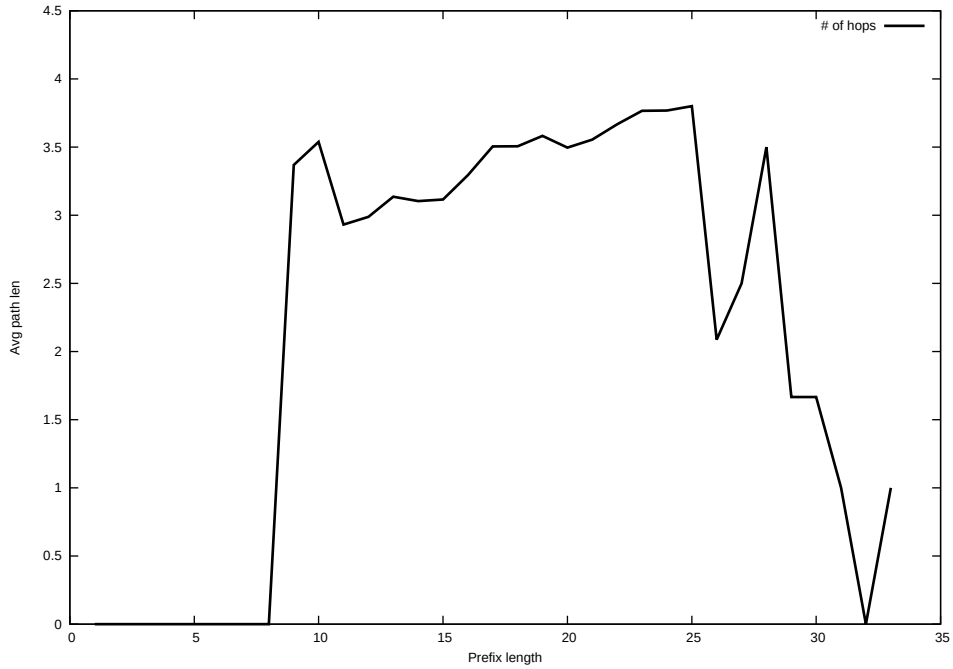
2012-11-11



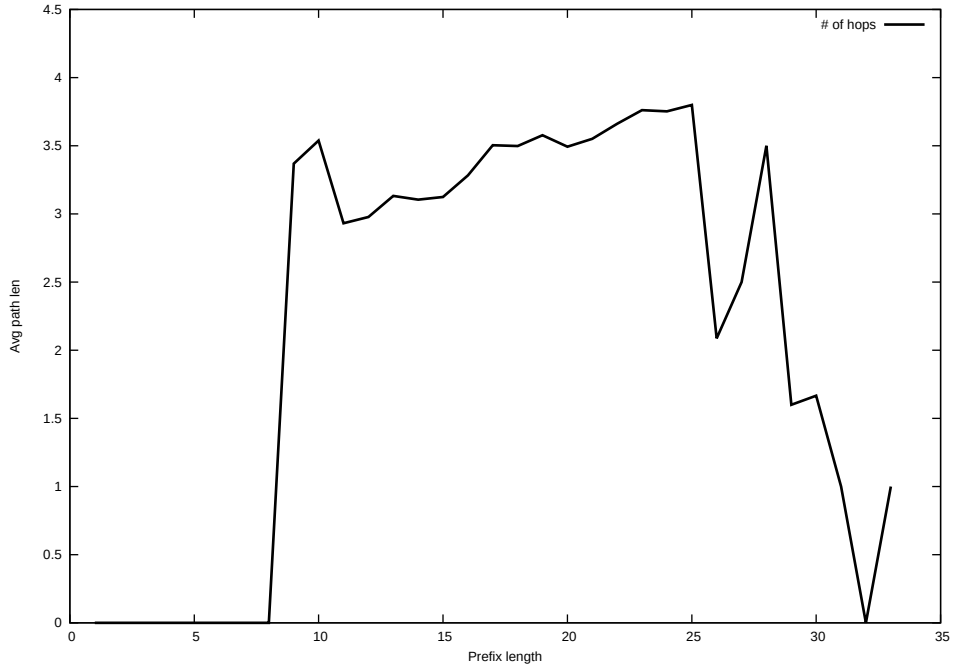
2012-11-12



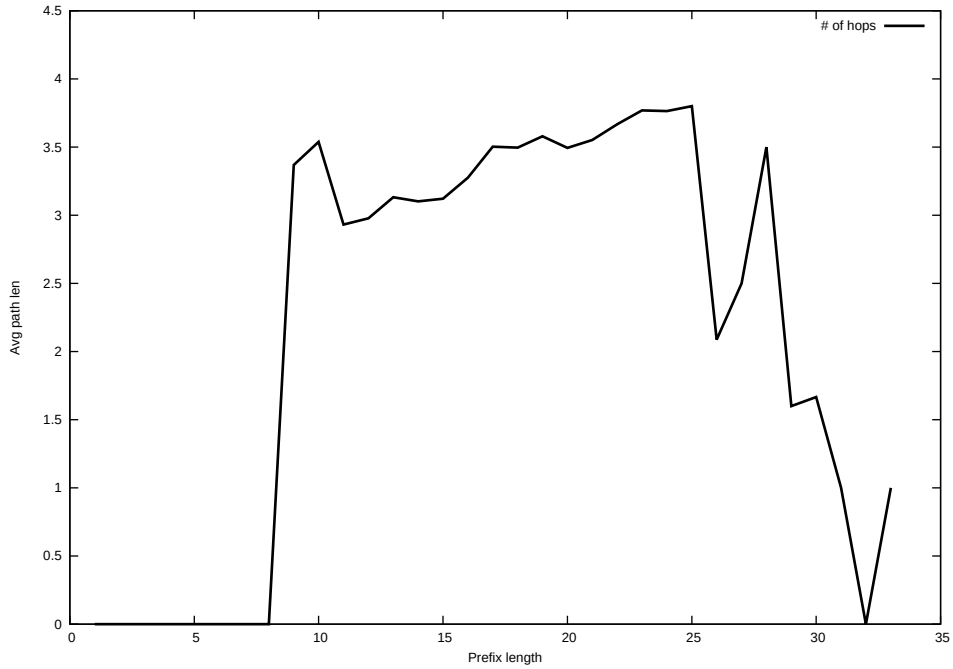
2012-11-13



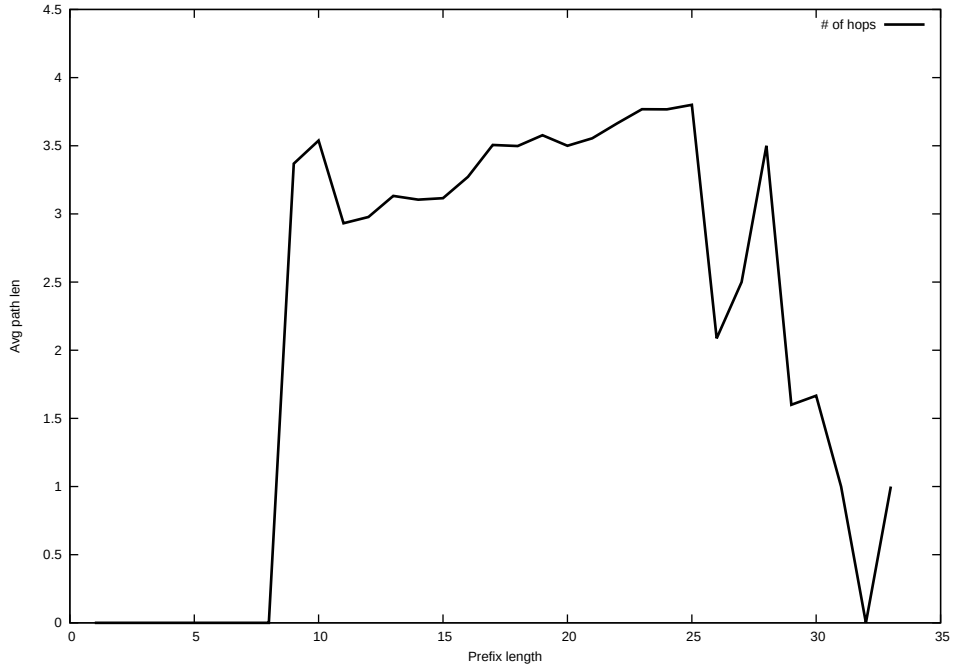
2012-11-14



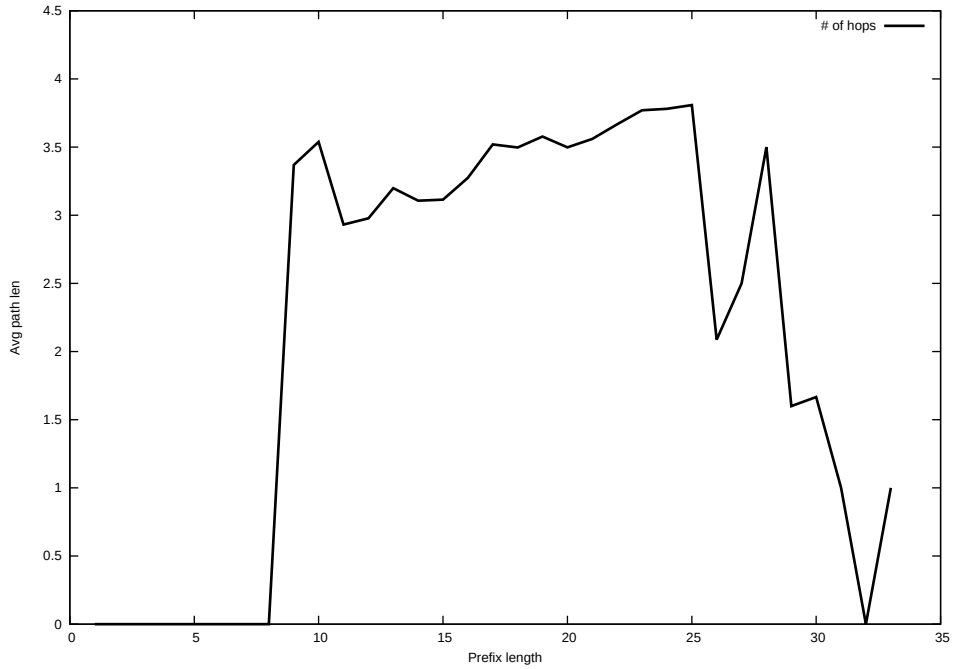
2012-11-15



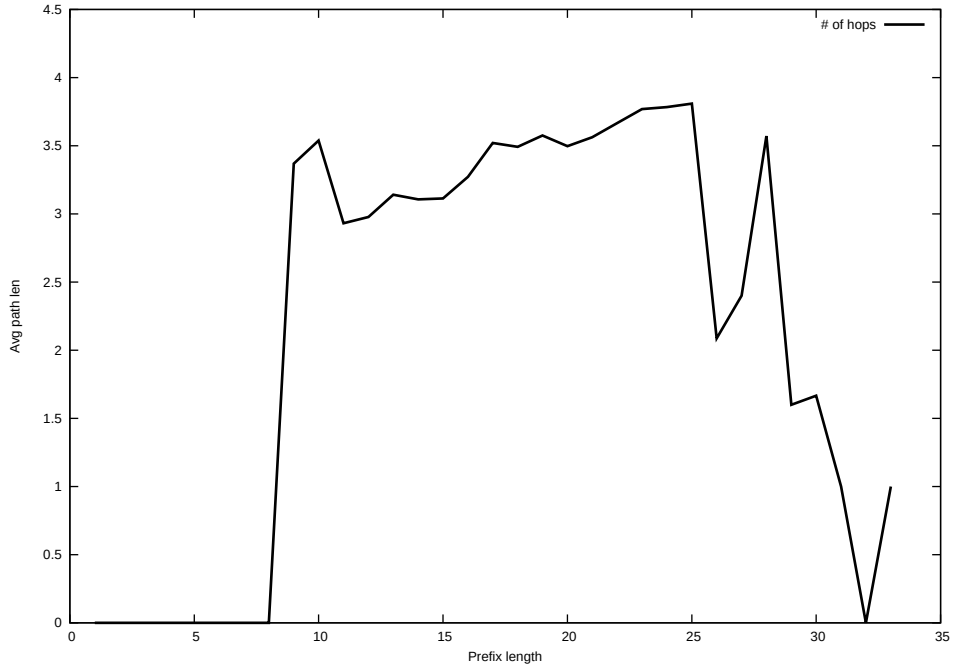
2012-11-16



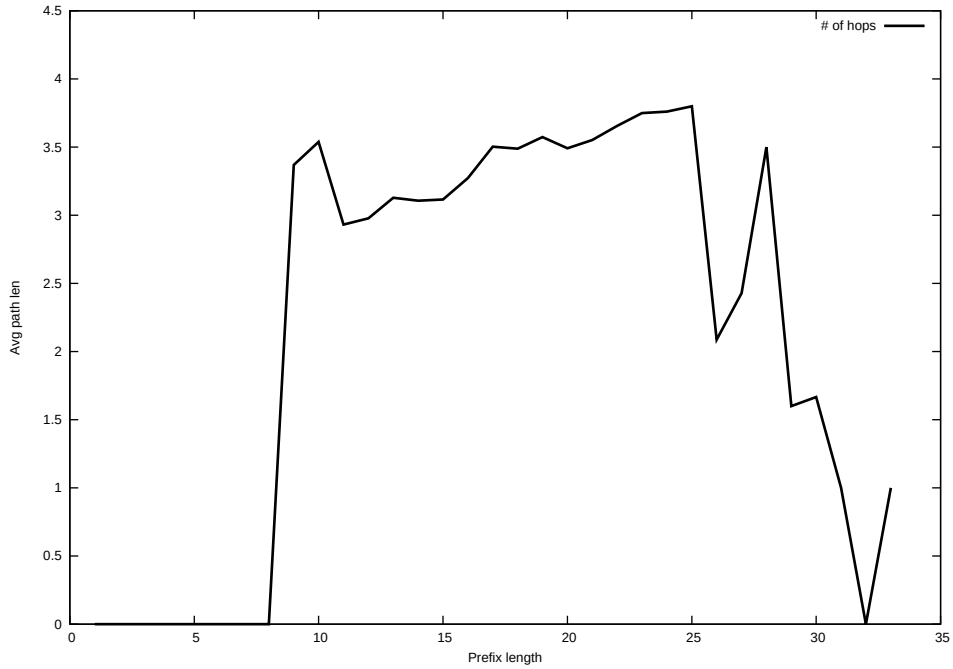
2012-11-17



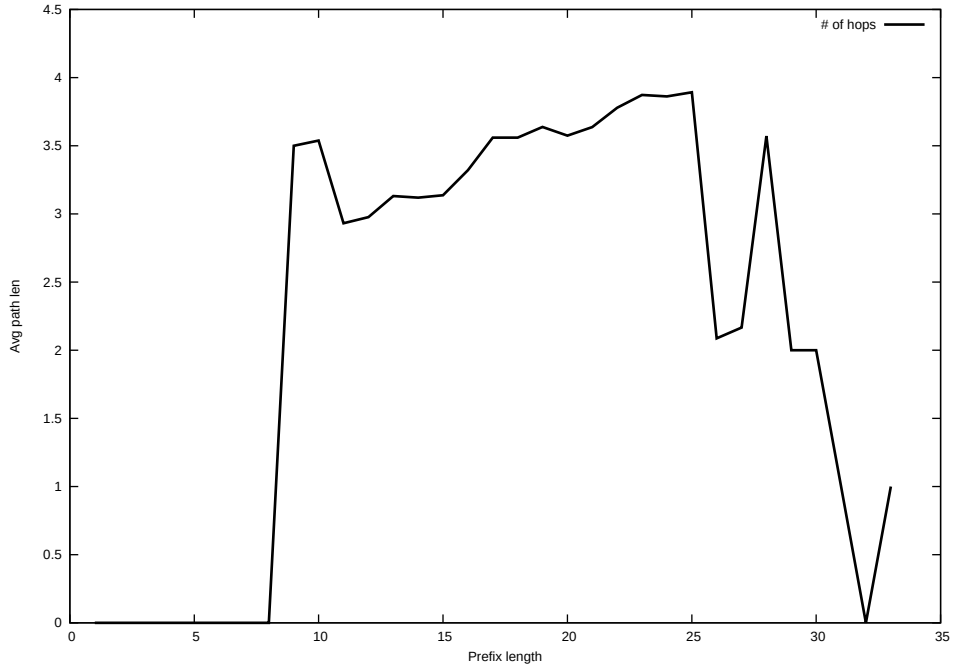
2012-11-18



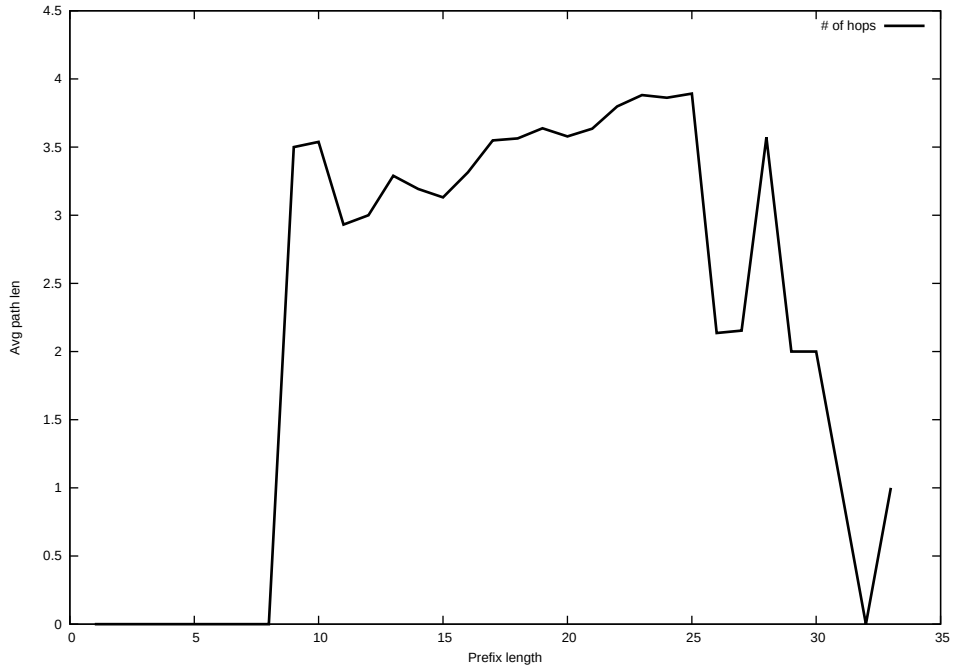
2012-11-19



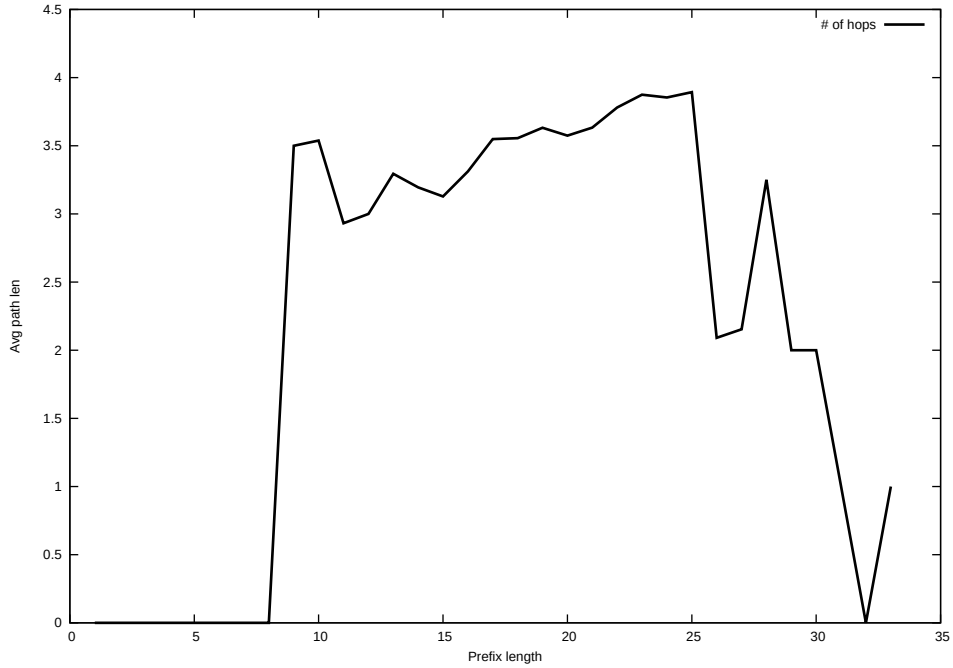
2012-11-20



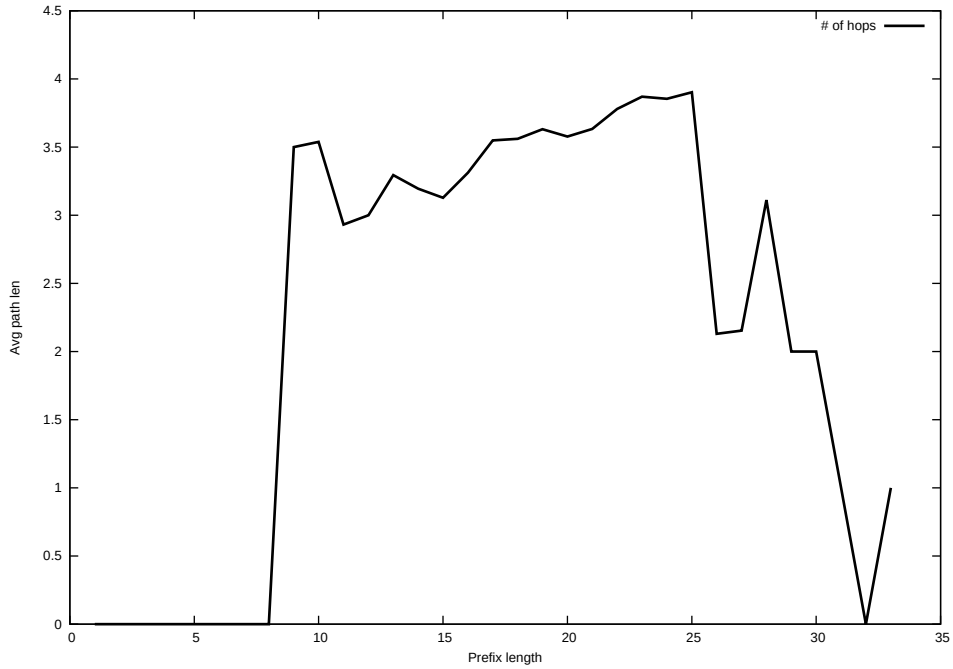
2012-11-21



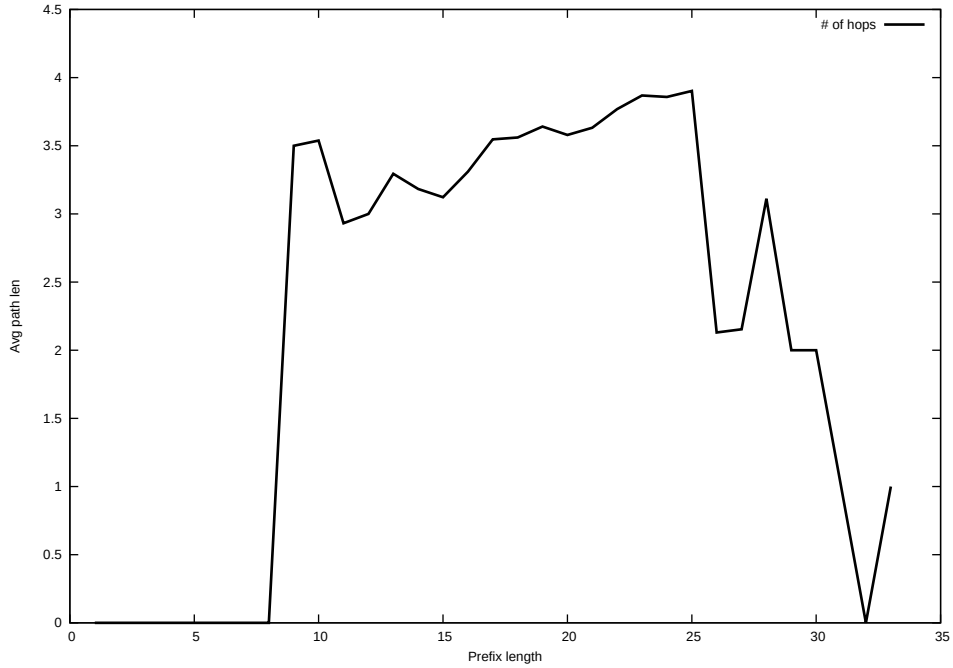
2012-11-22



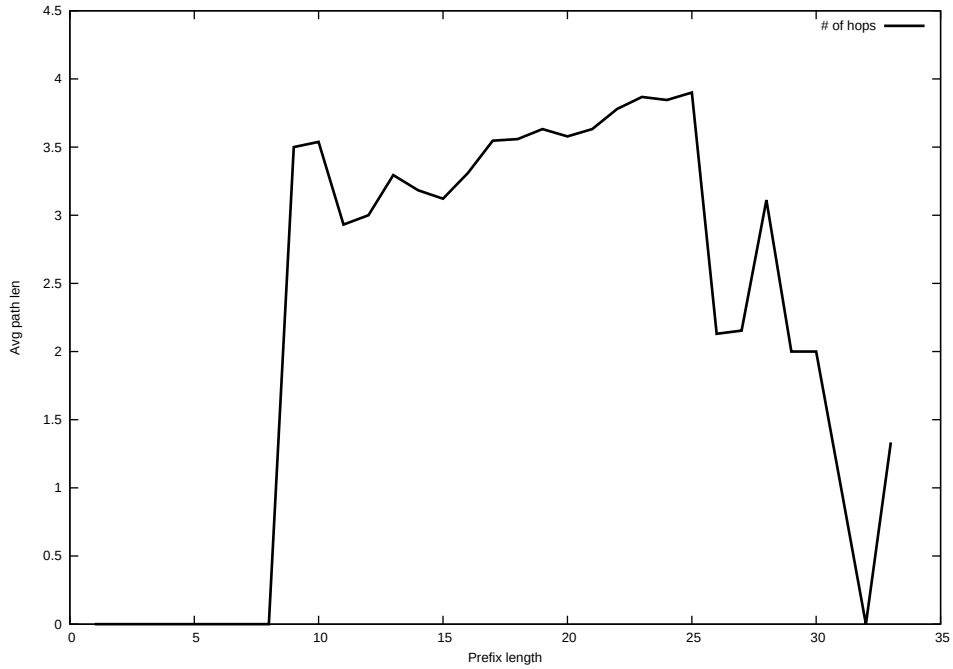
2012-11-23



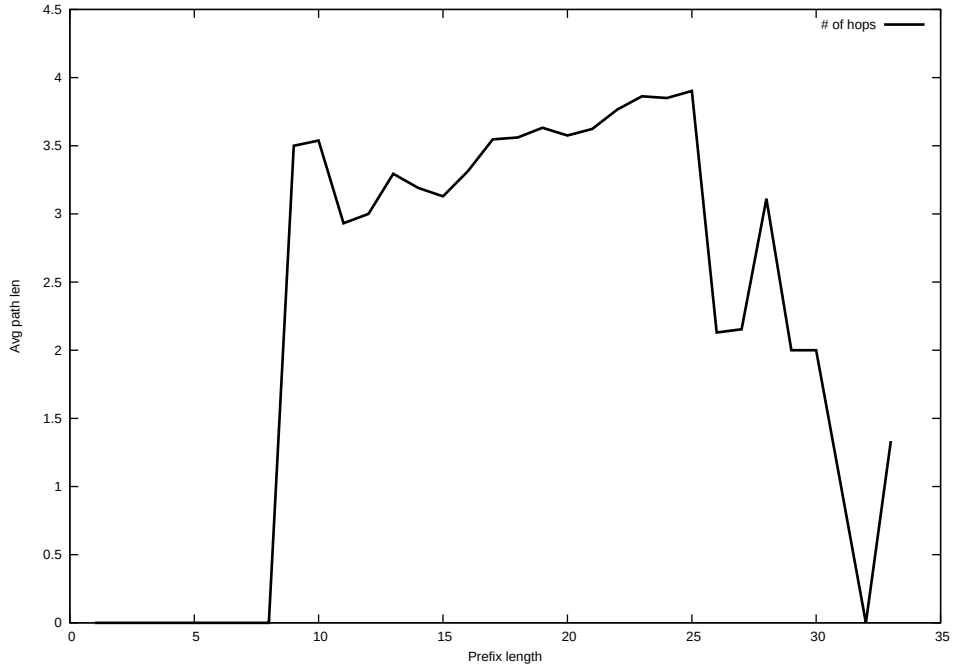
2012-11-24



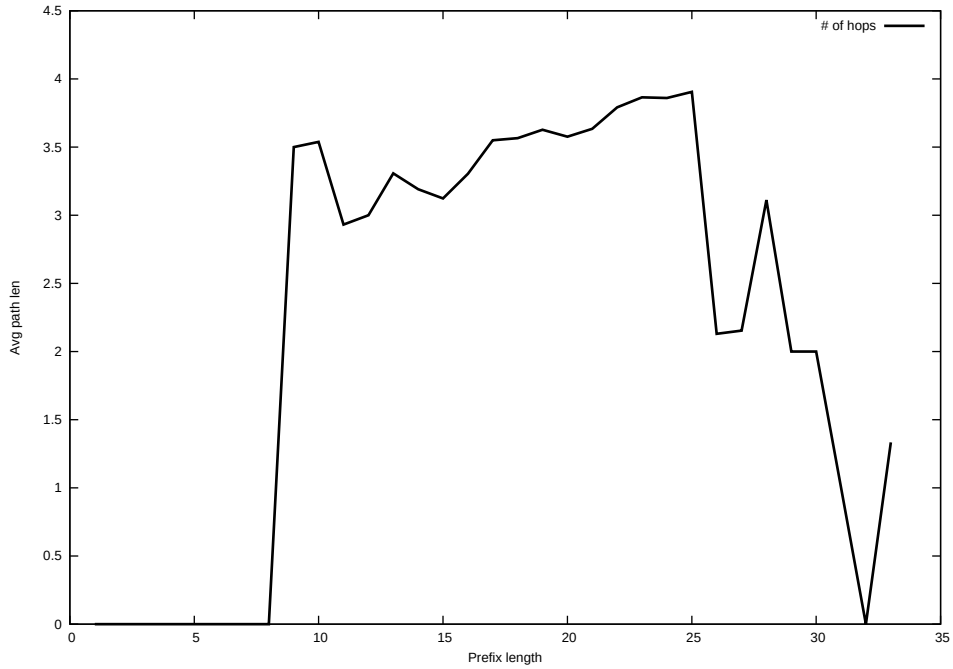
2012-11-25



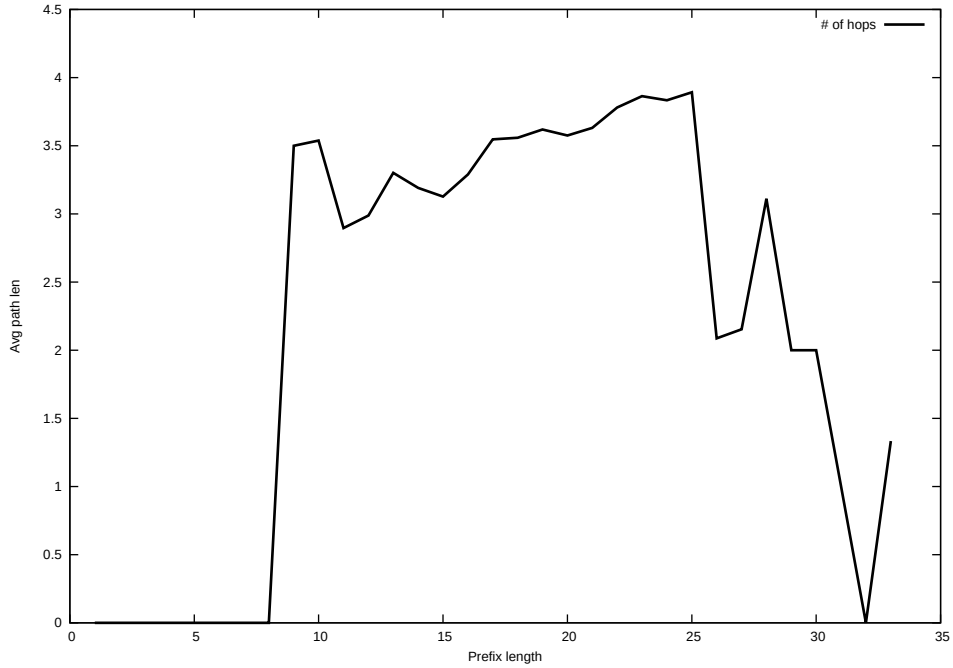
2012-11-26



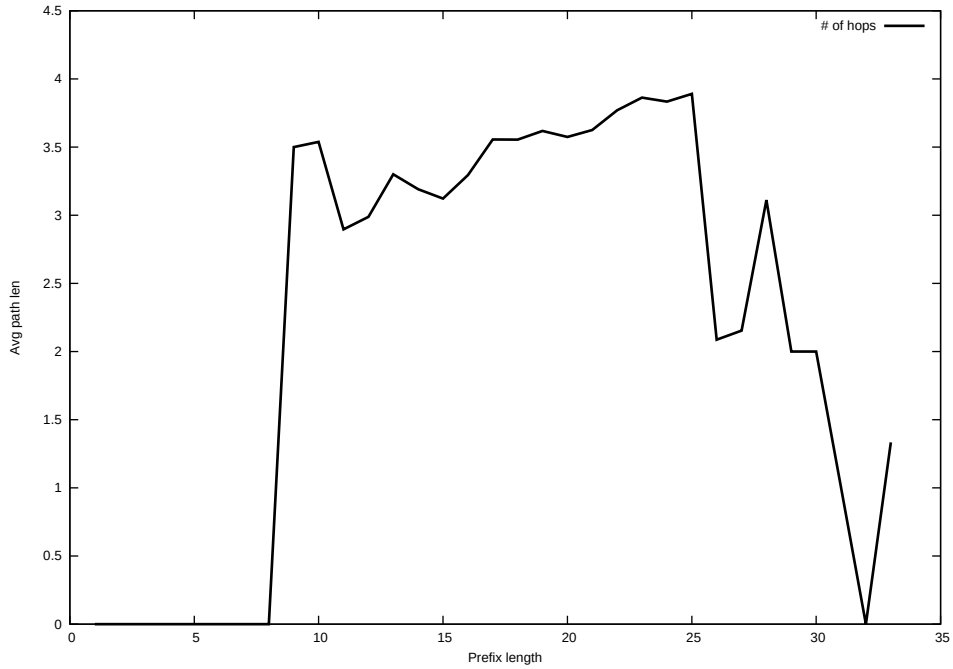
2012-11-27



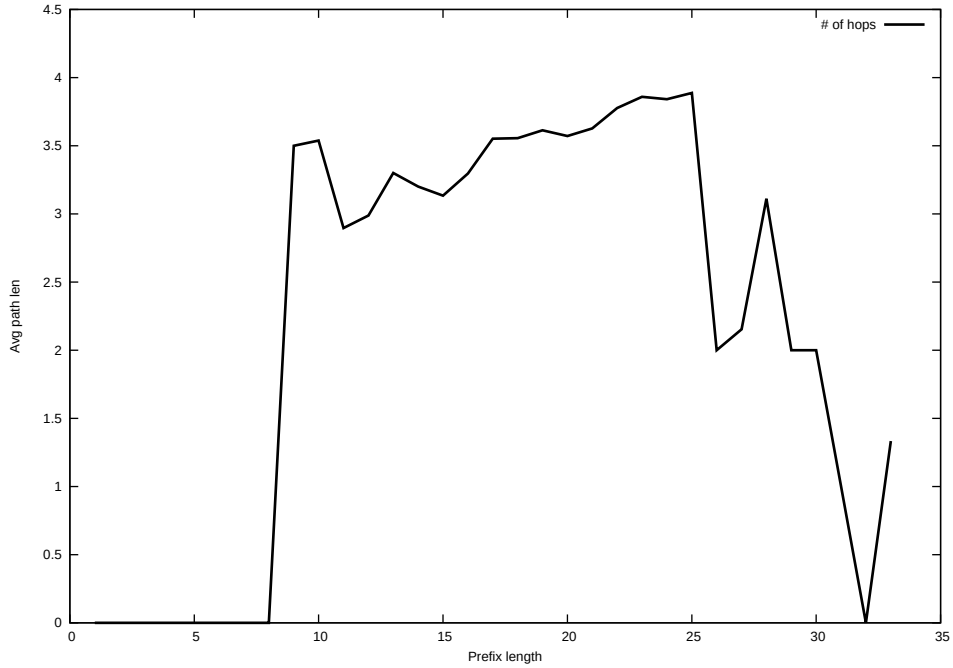
2012-11-28



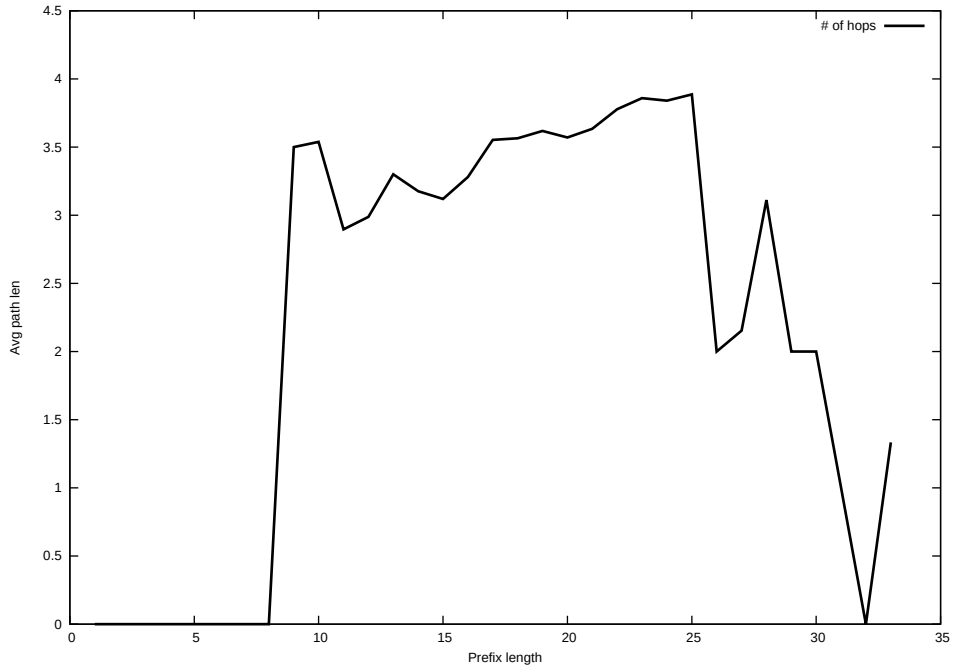
2012-11-29



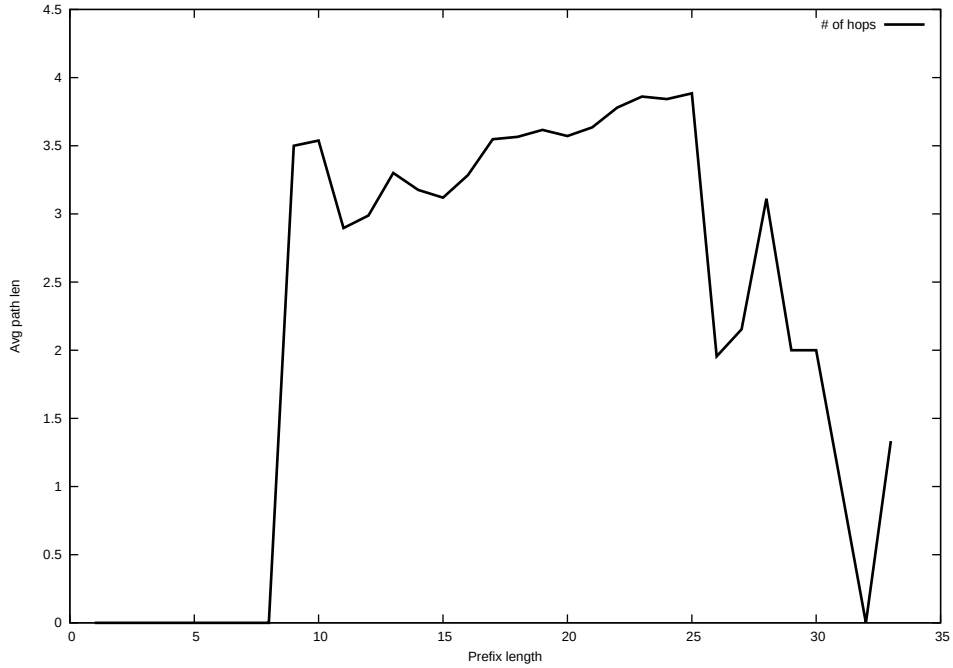
2012-11-30



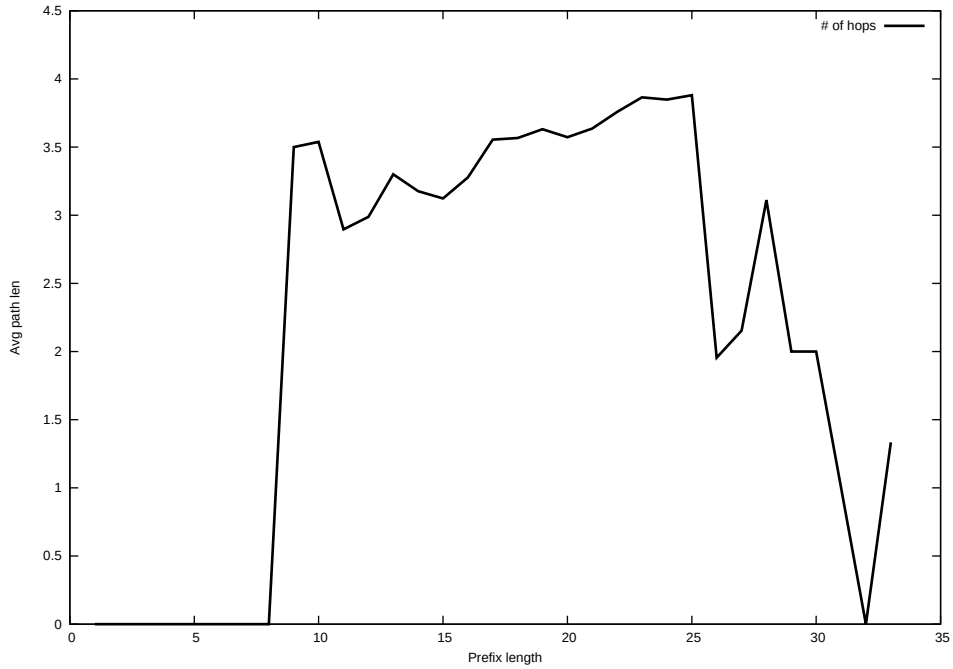
2012-12-01



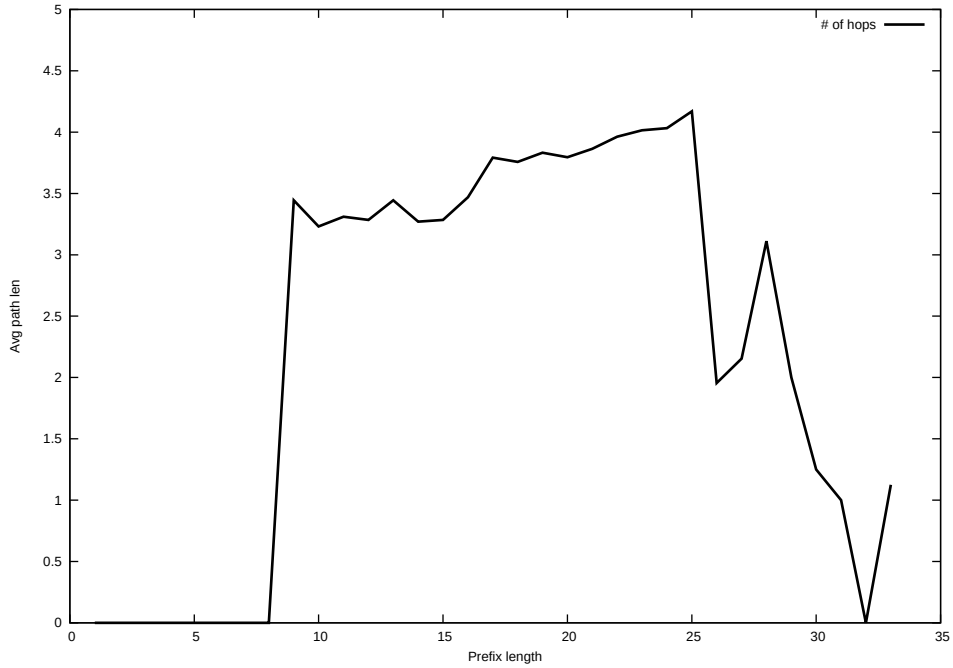
2012-12-02



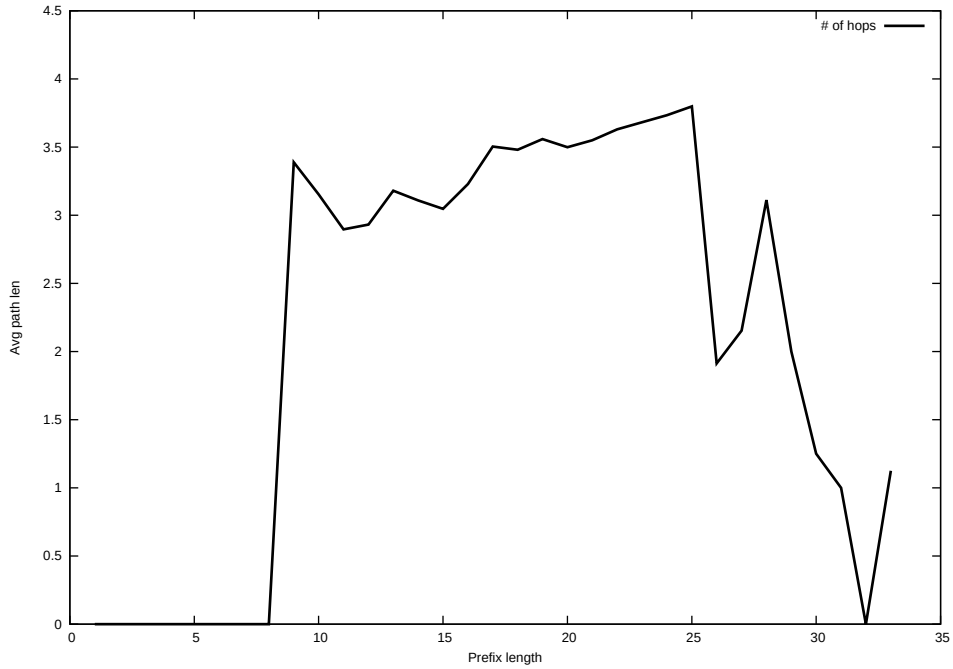
2012-12-03



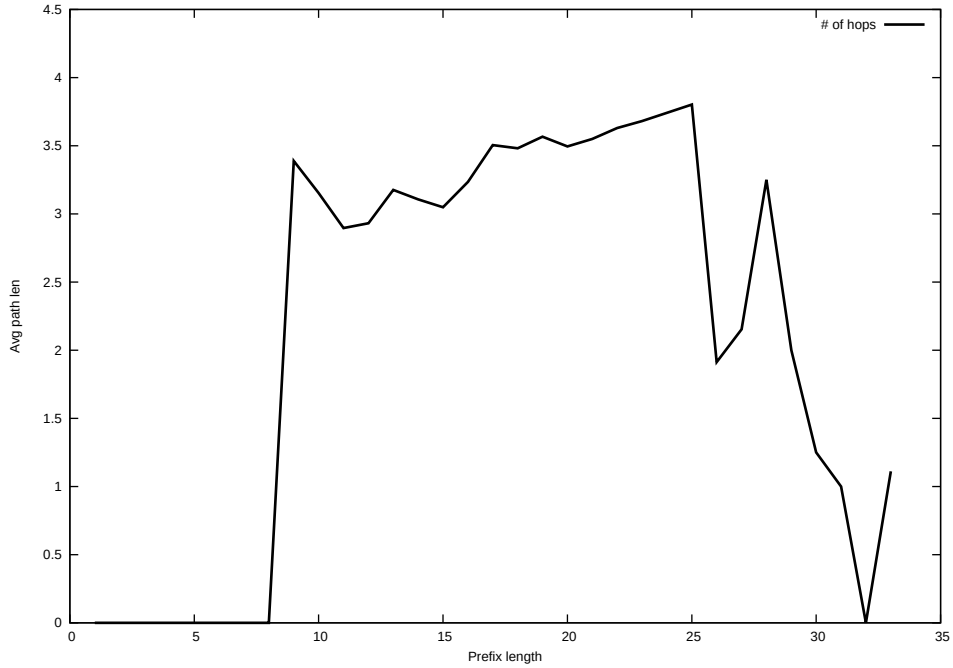
2012-12-04



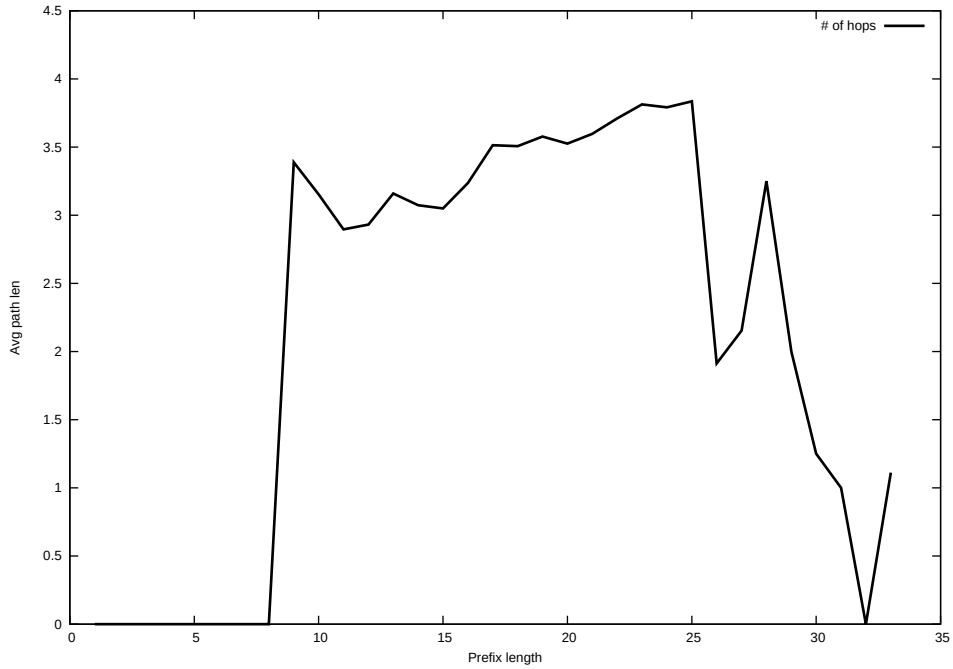
2012-12-05



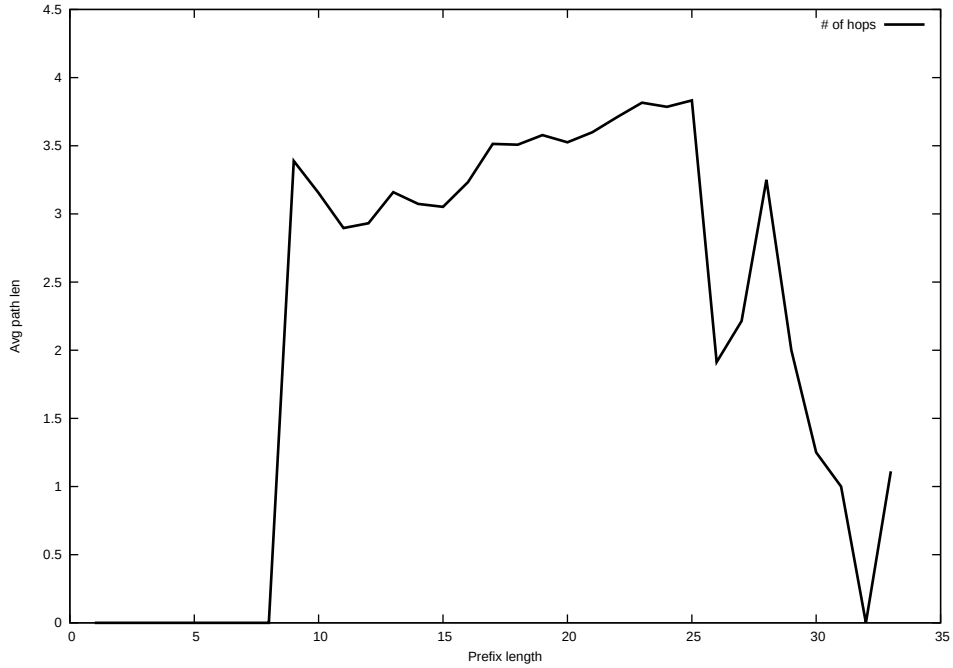
2012-12-06



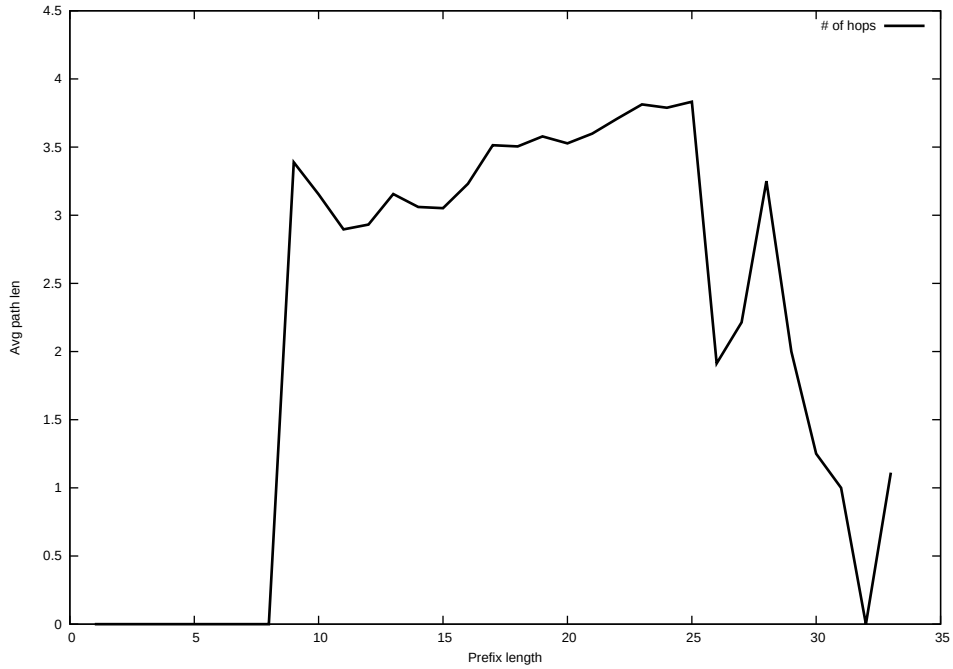
2012-12-07



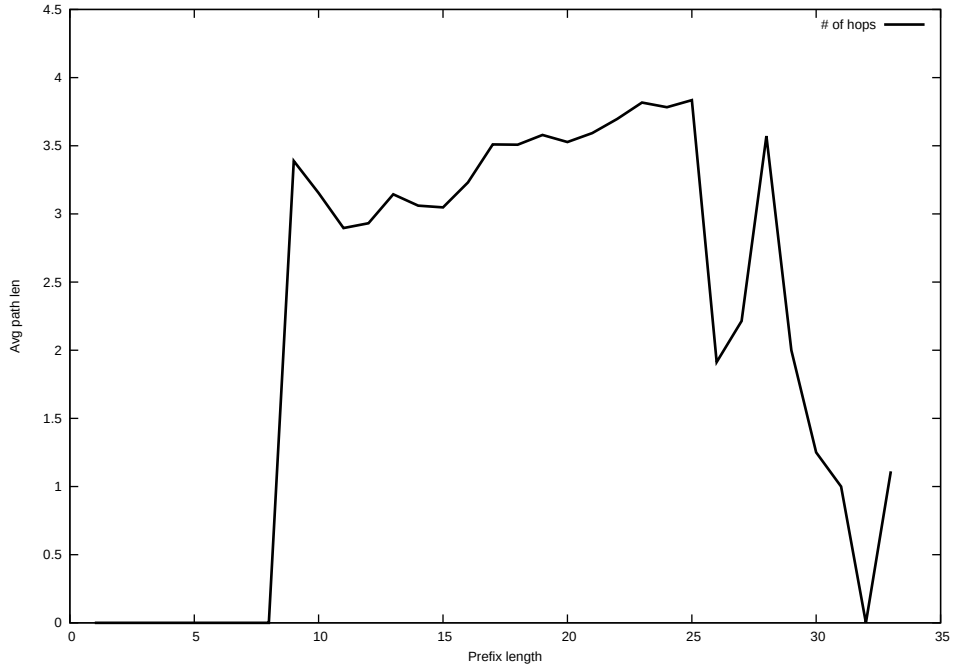
2012-12-08



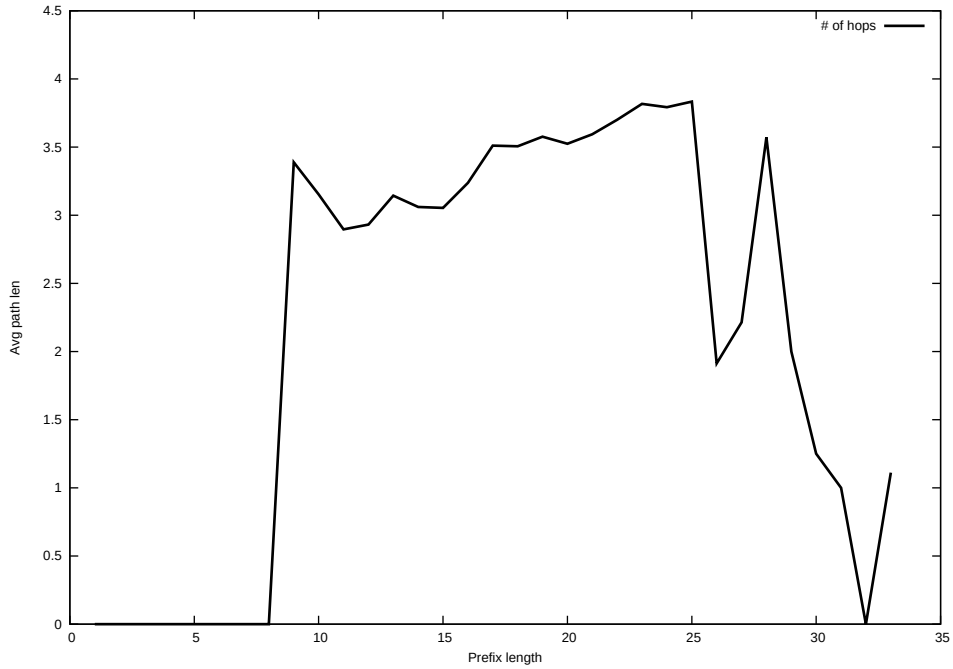
2012-12-09



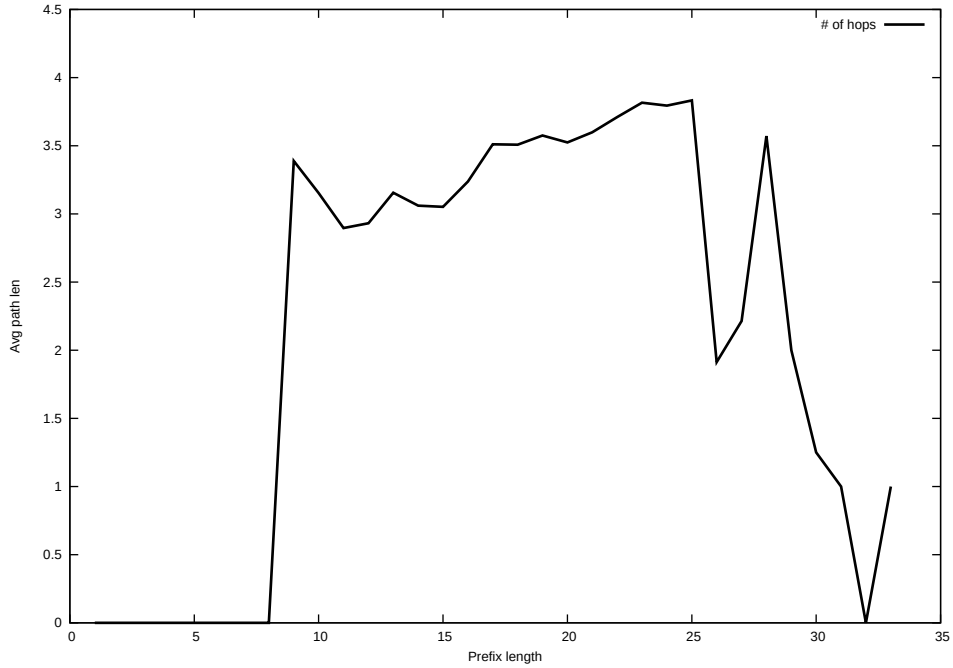
2012-12-10



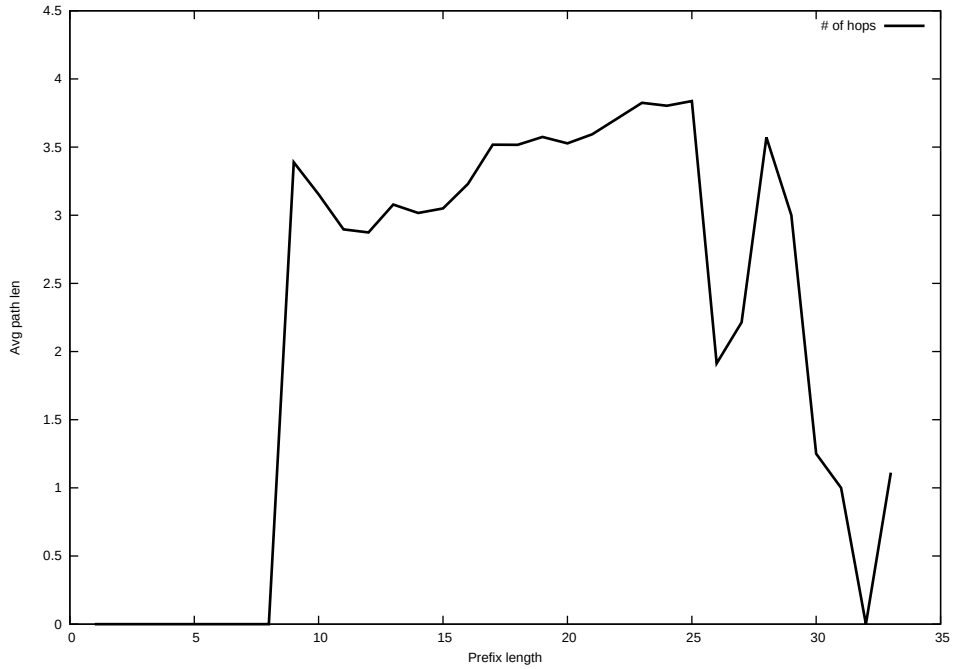
2012-12-11



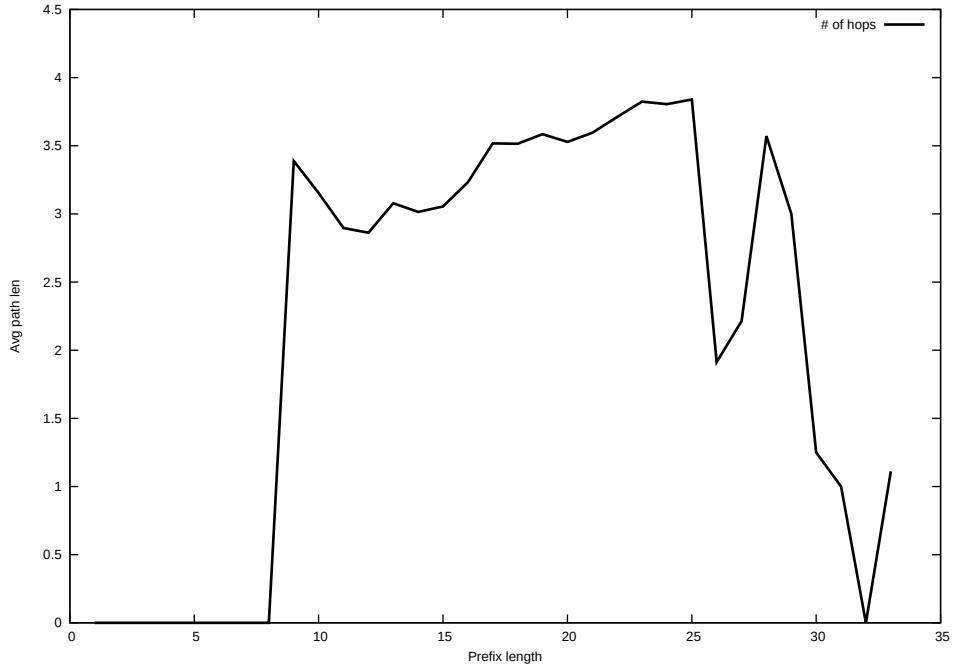
2012-12-12



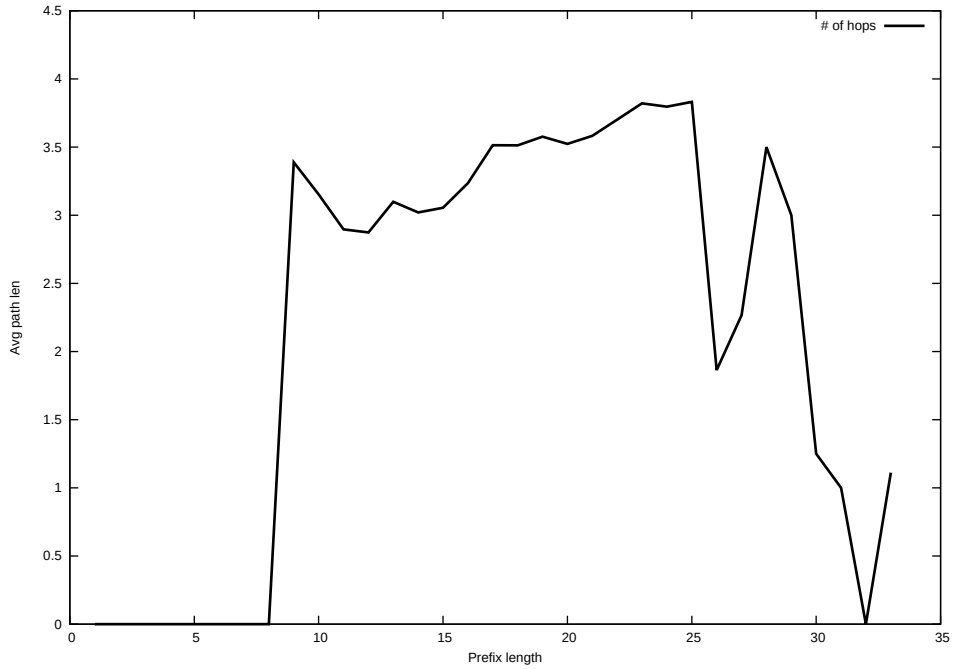
2012-12-13



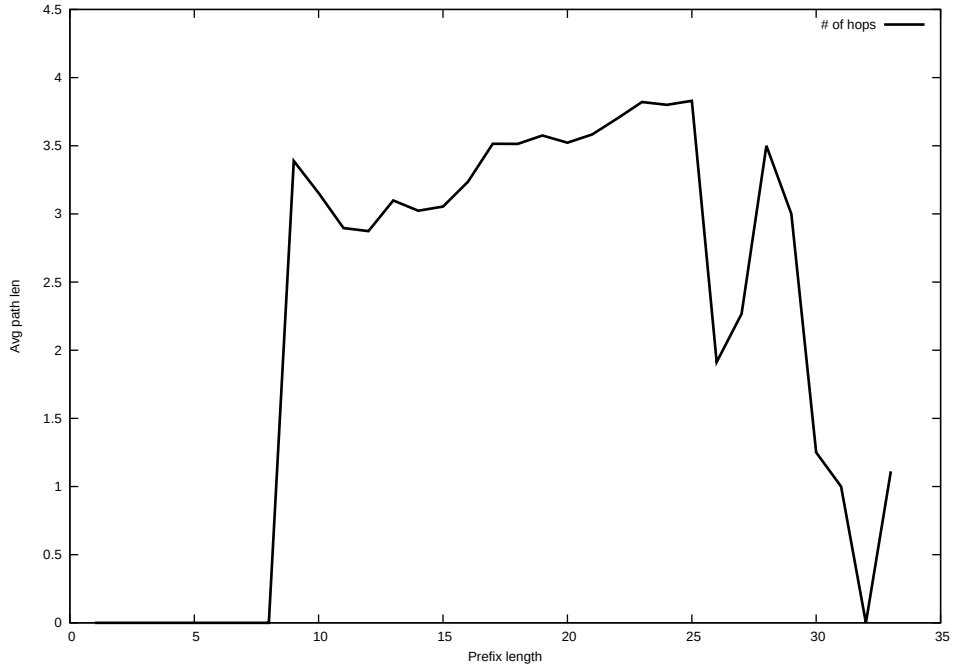
2012-12-14



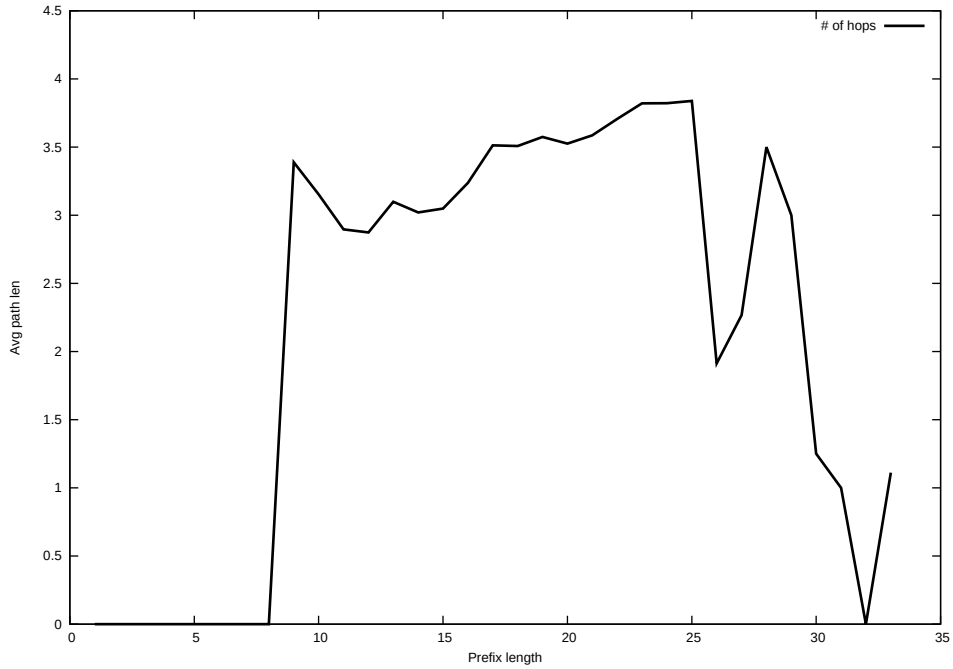
2012-12-15



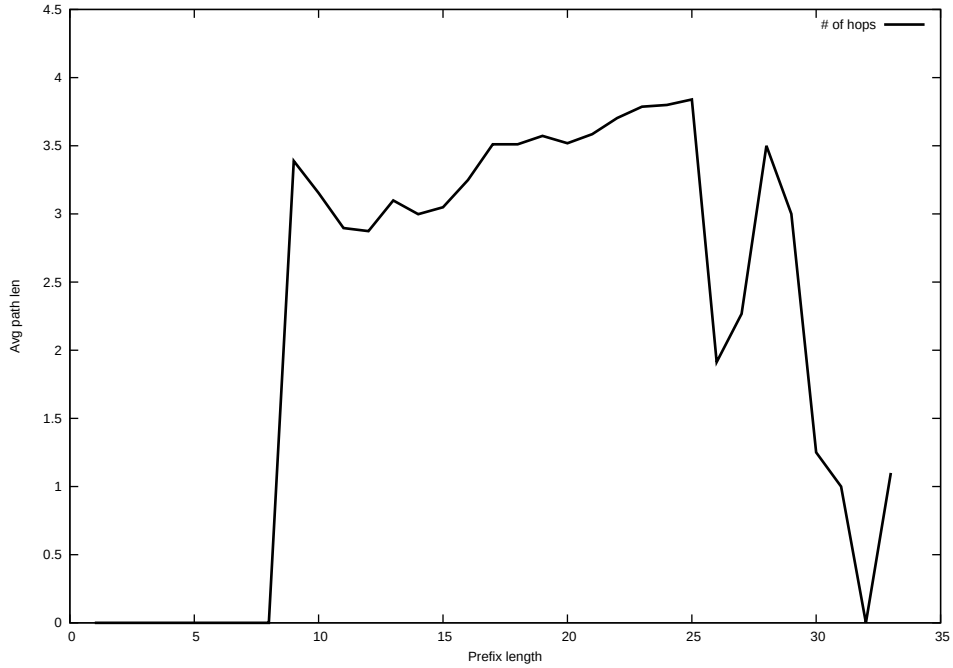
2012-12-16



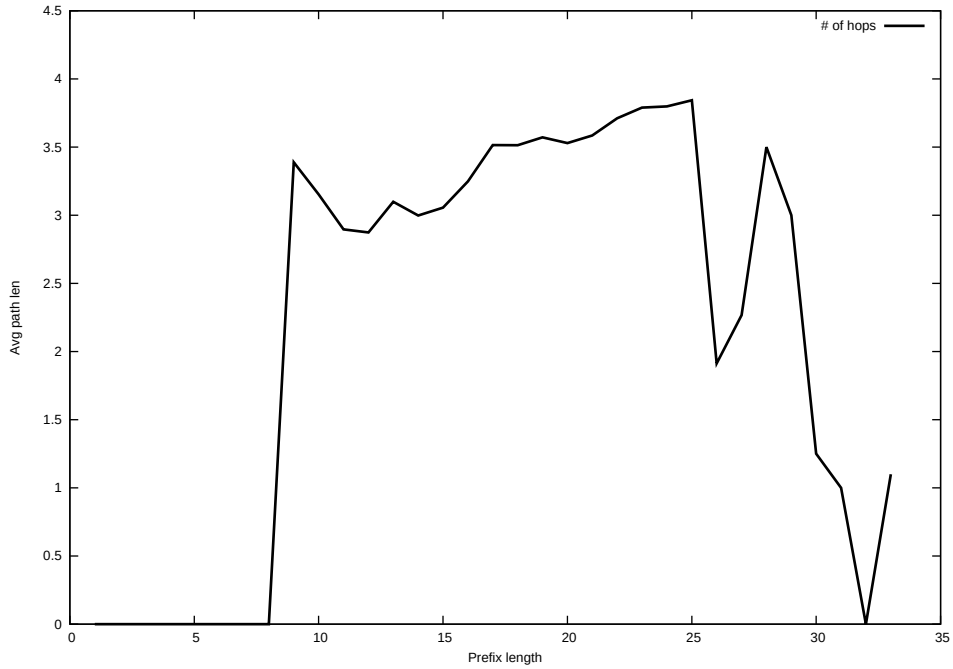
2012-12-17



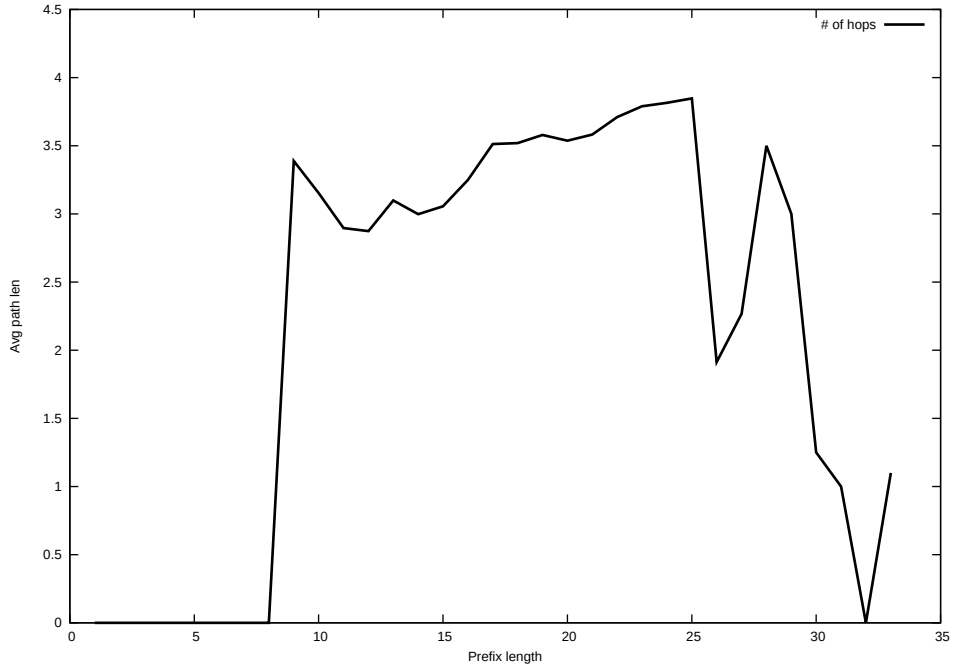
2012-12-18



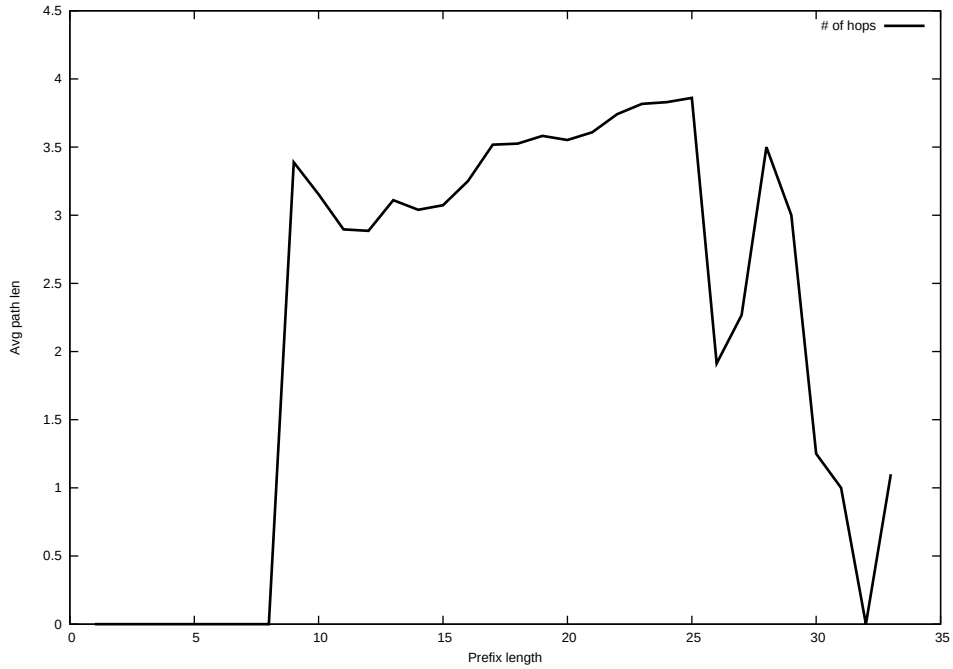
2012-12-19



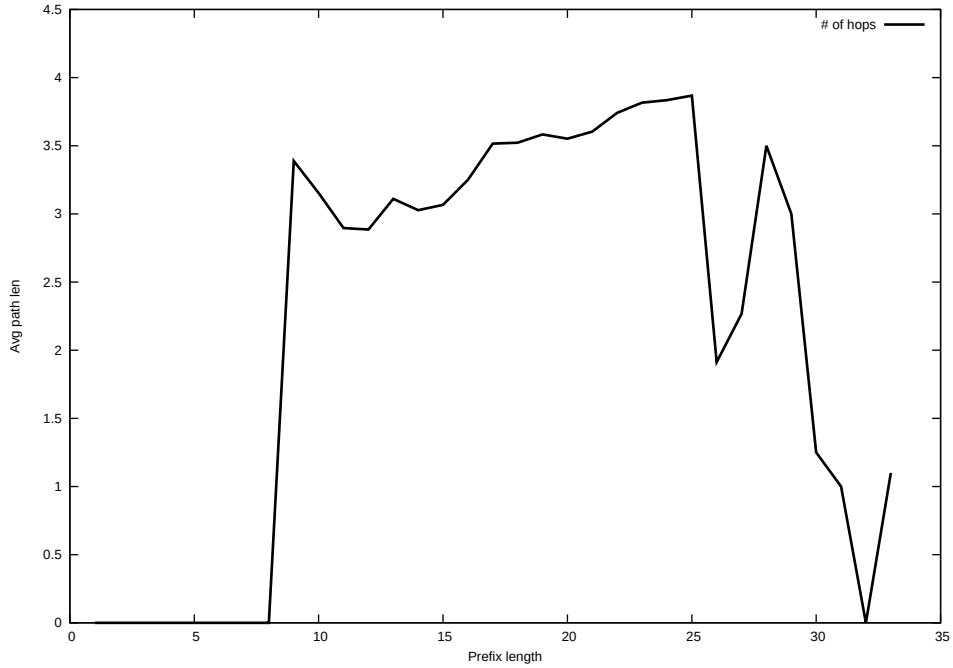
2012-12-20



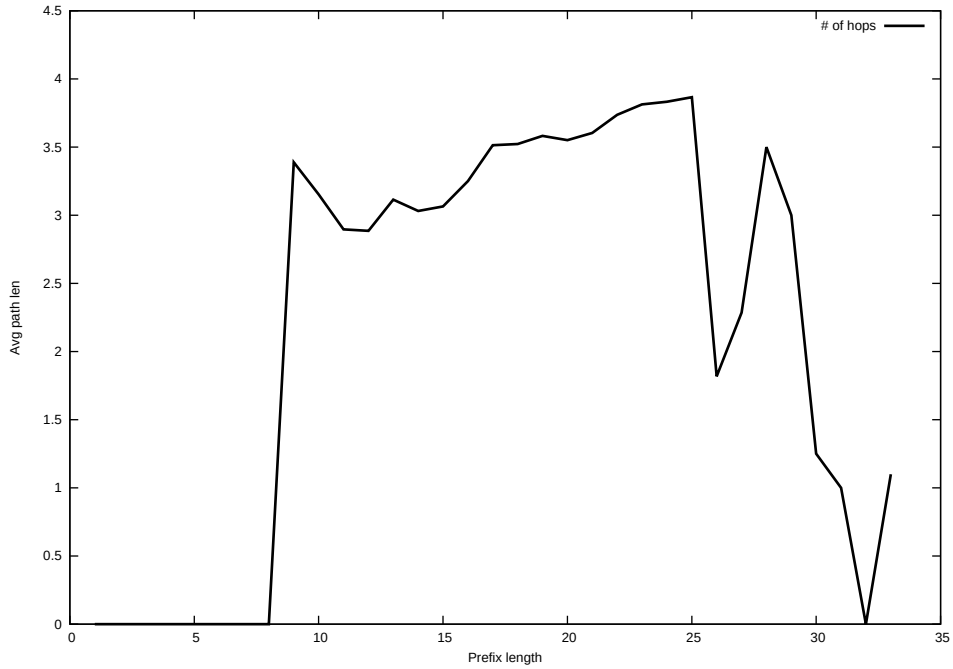
2012-12-21



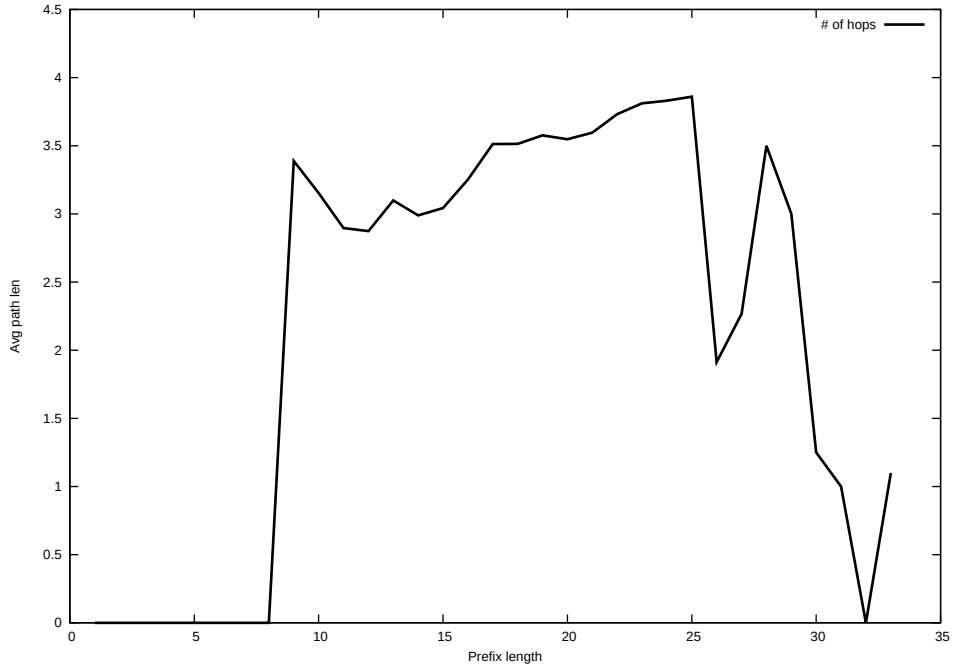
2012-12-22



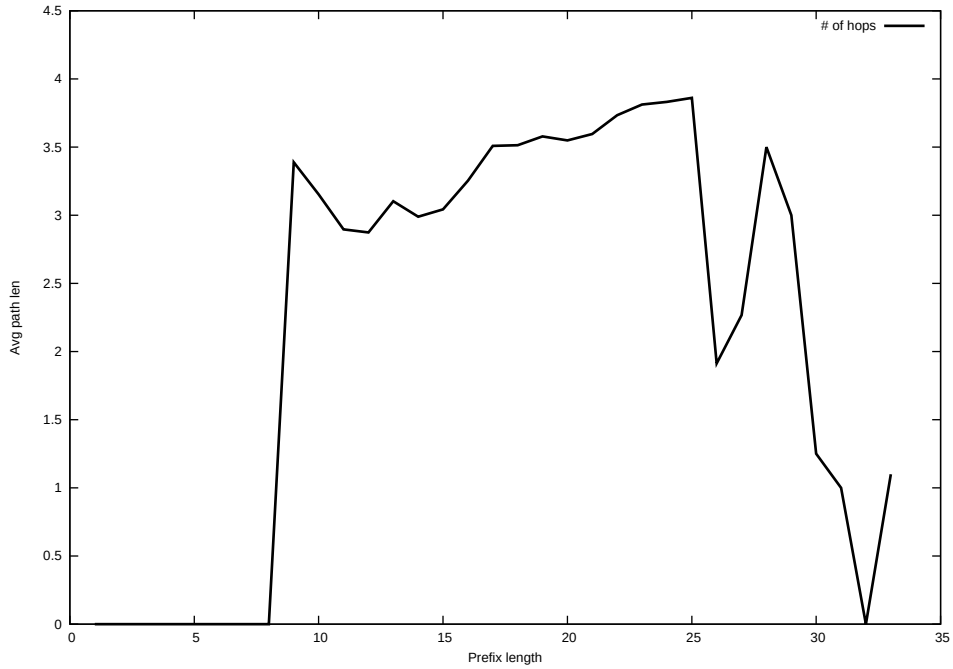
2012-12-23



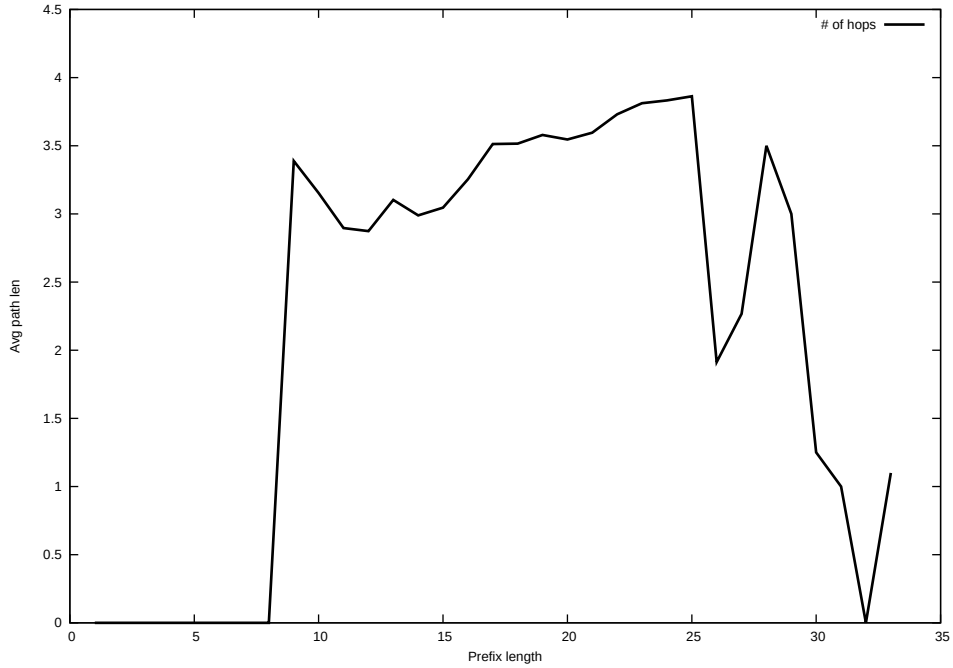
2012-12-24



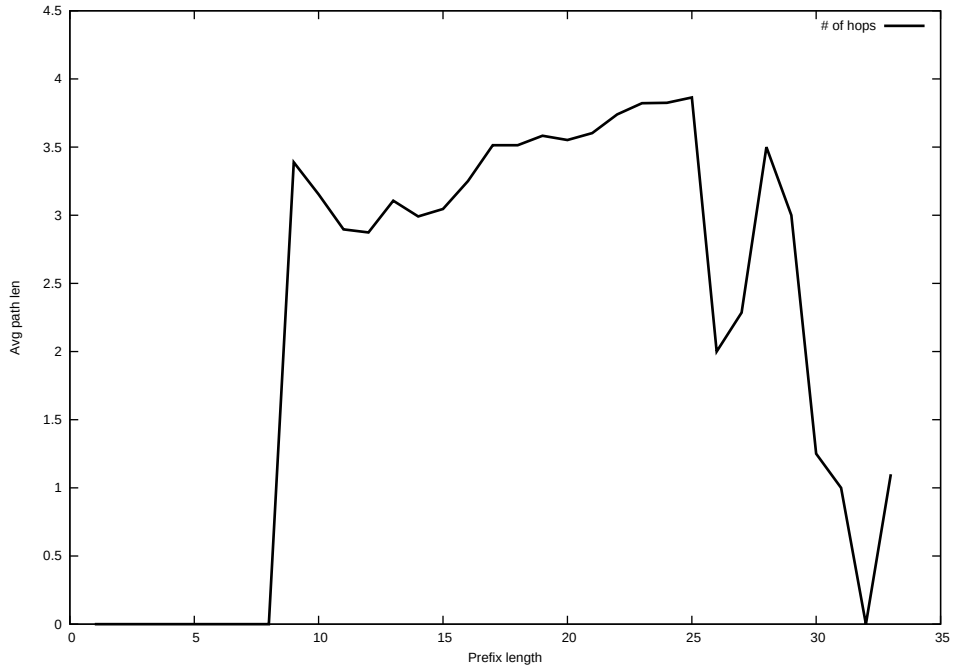
2012-12-25



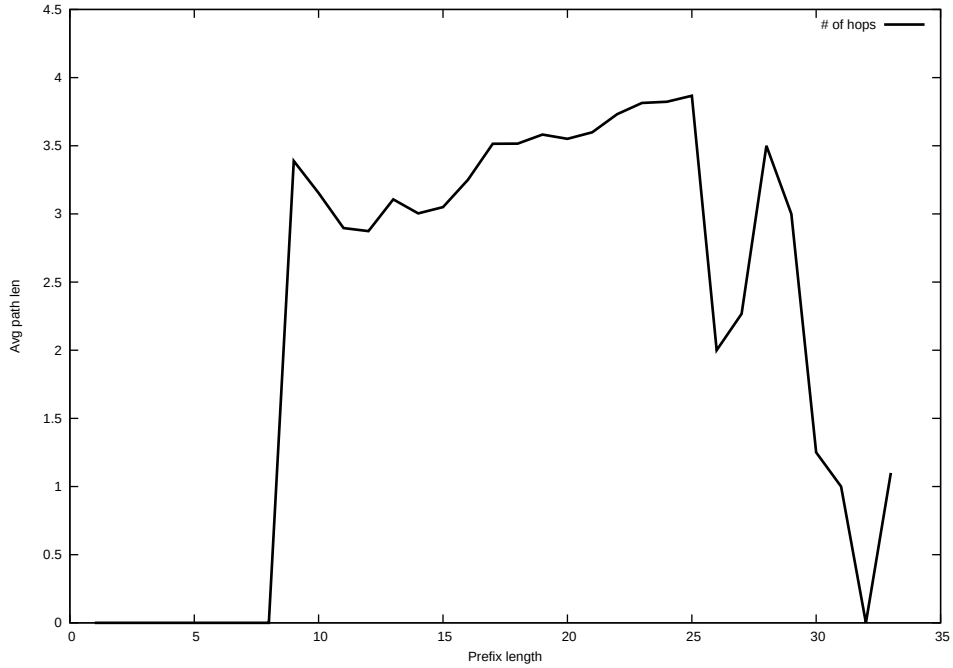
2012-12-26



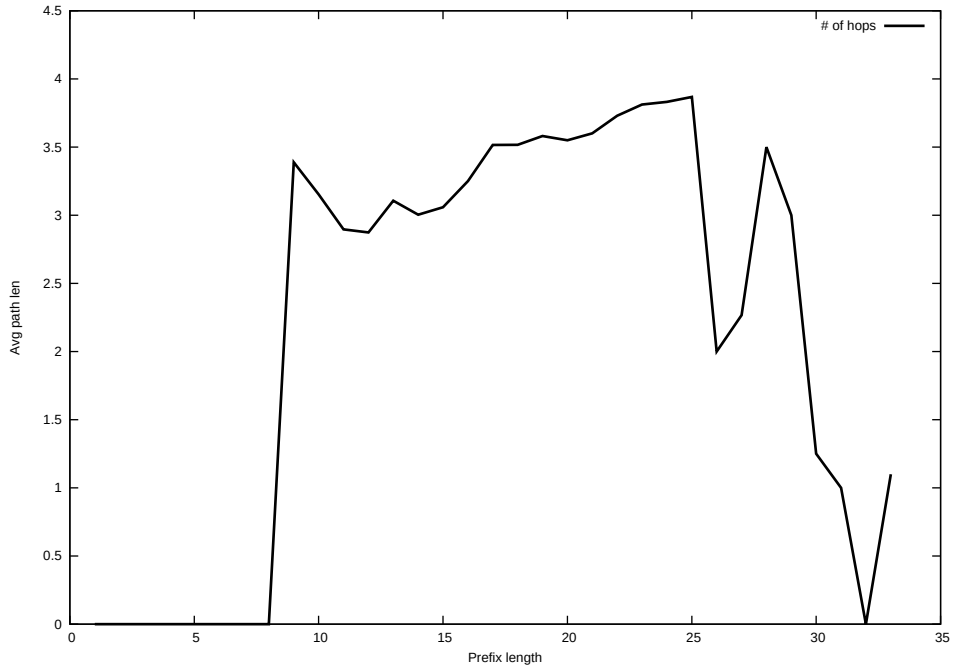
2012-12-27



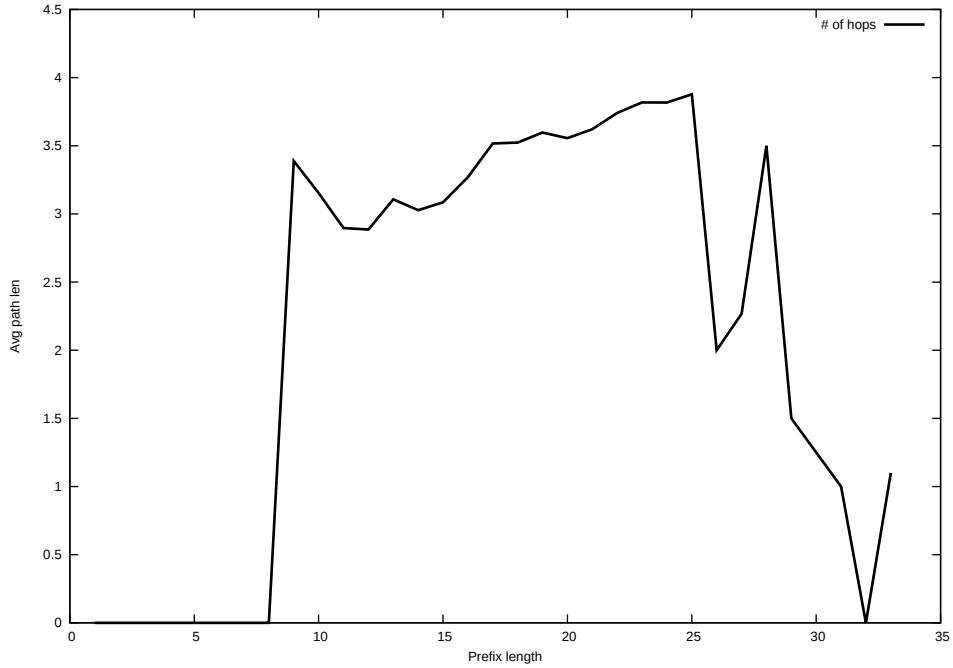
2012-12-28



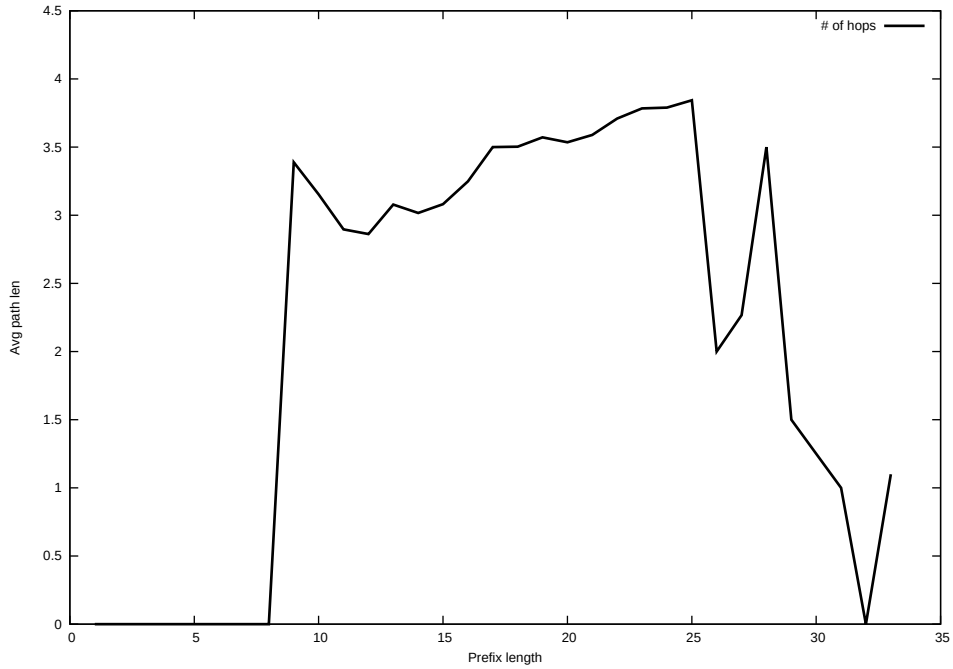
2012-12-29



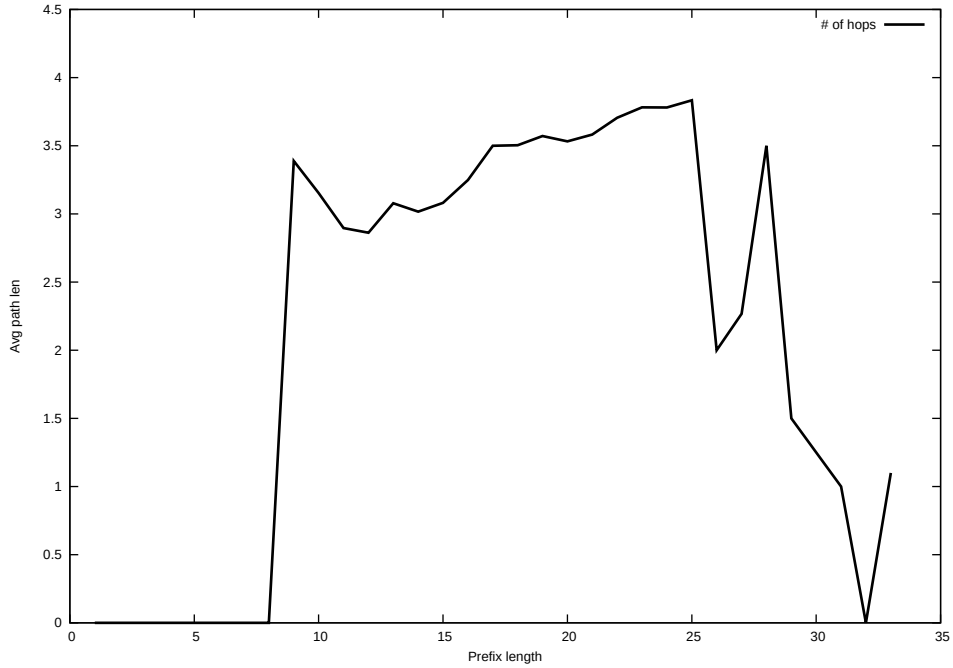
2012-12-30



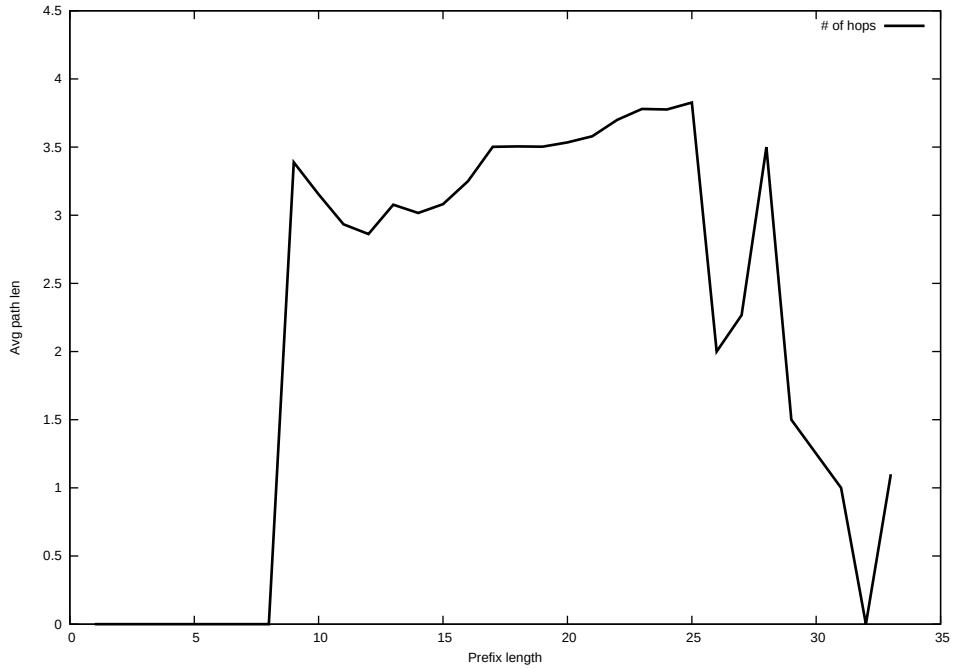
2012-12-31



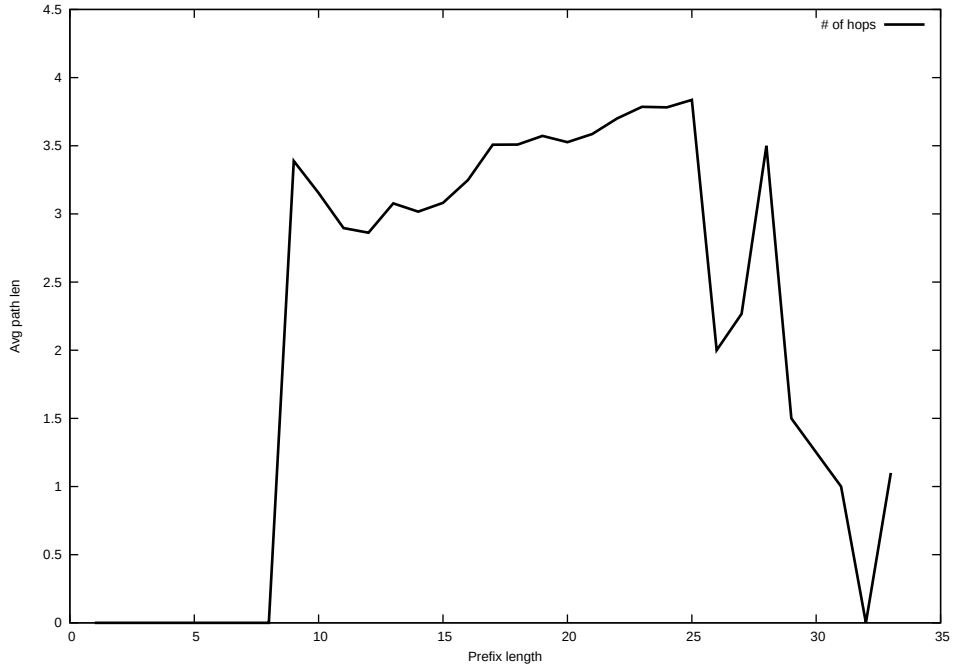
2013-01-01



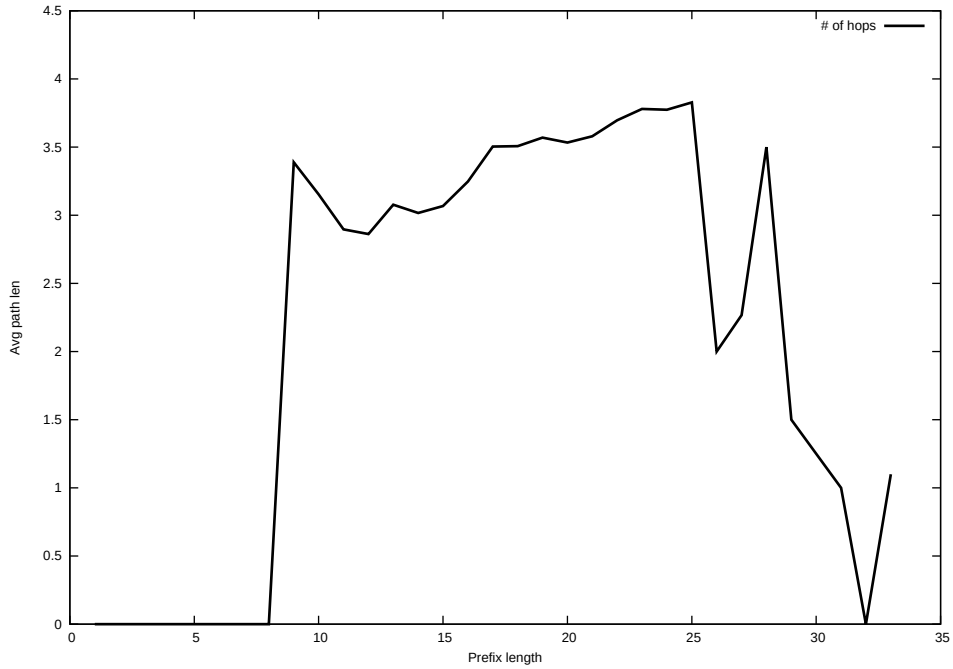
2013-01-02



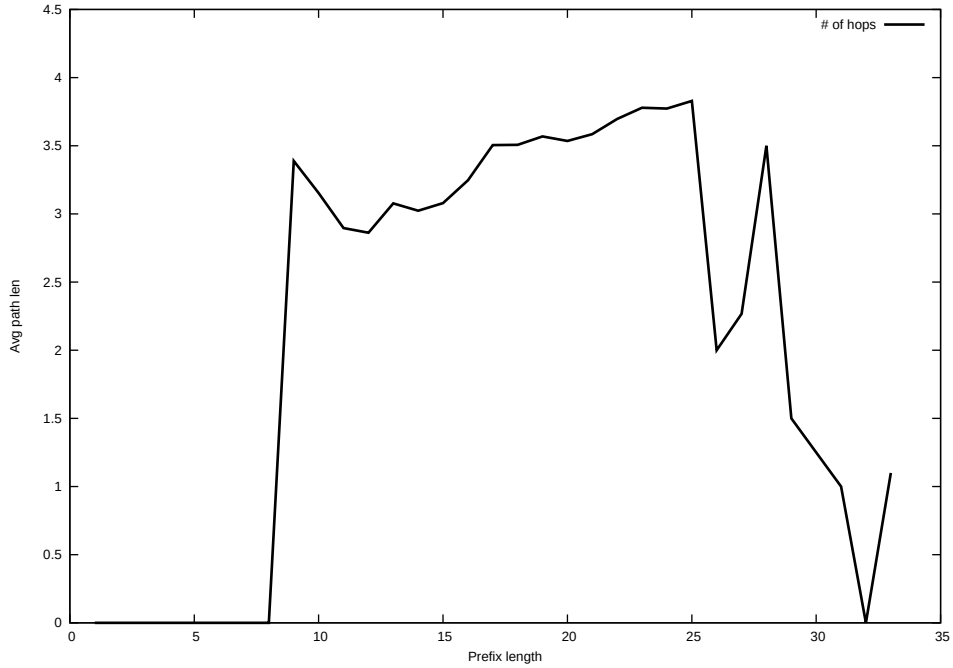
2013-01-03



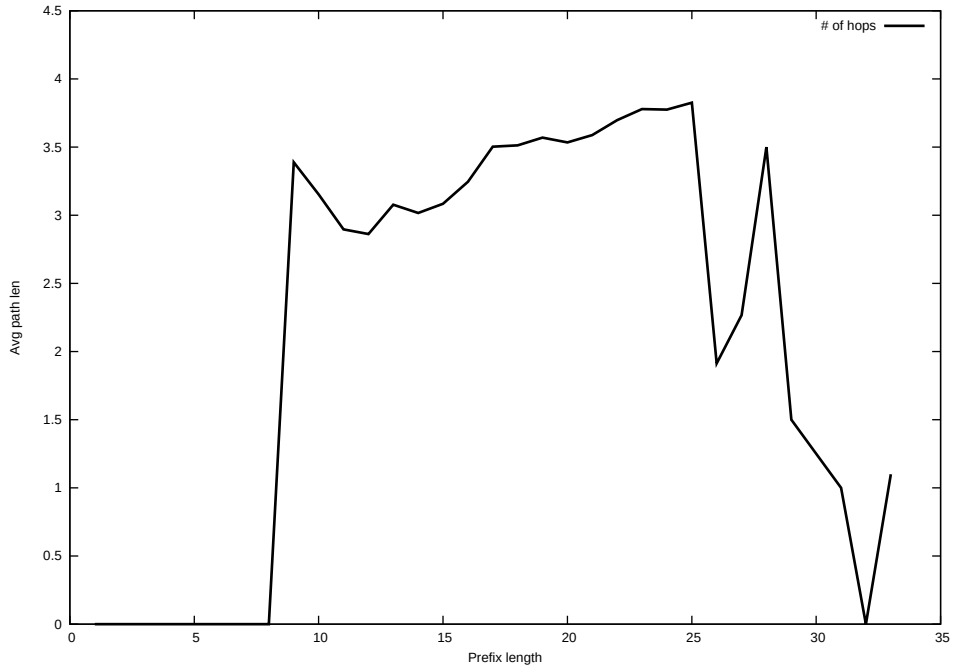
2013-01-04



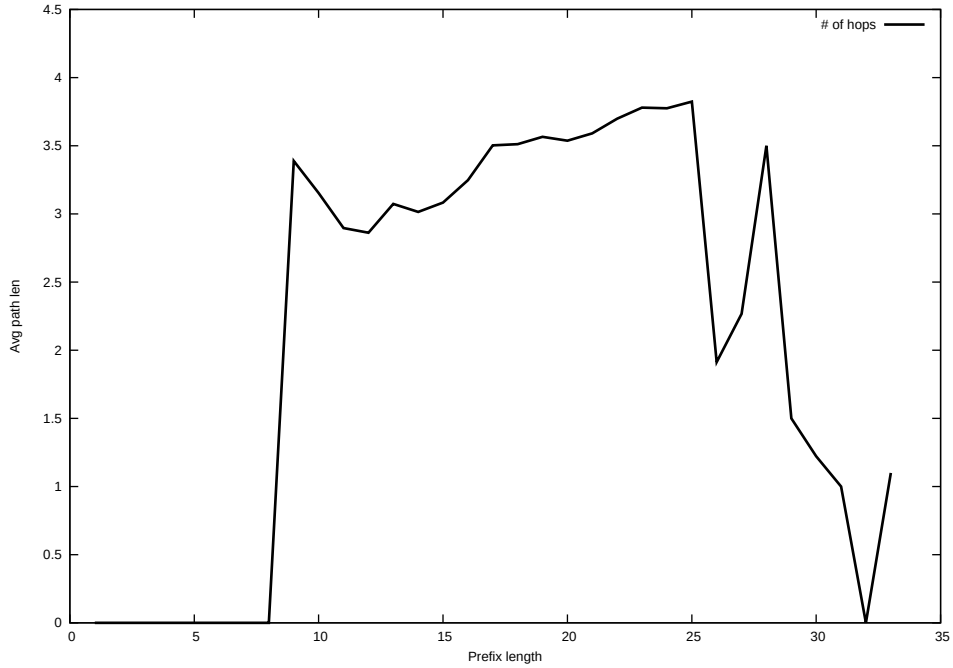
2013-01-05



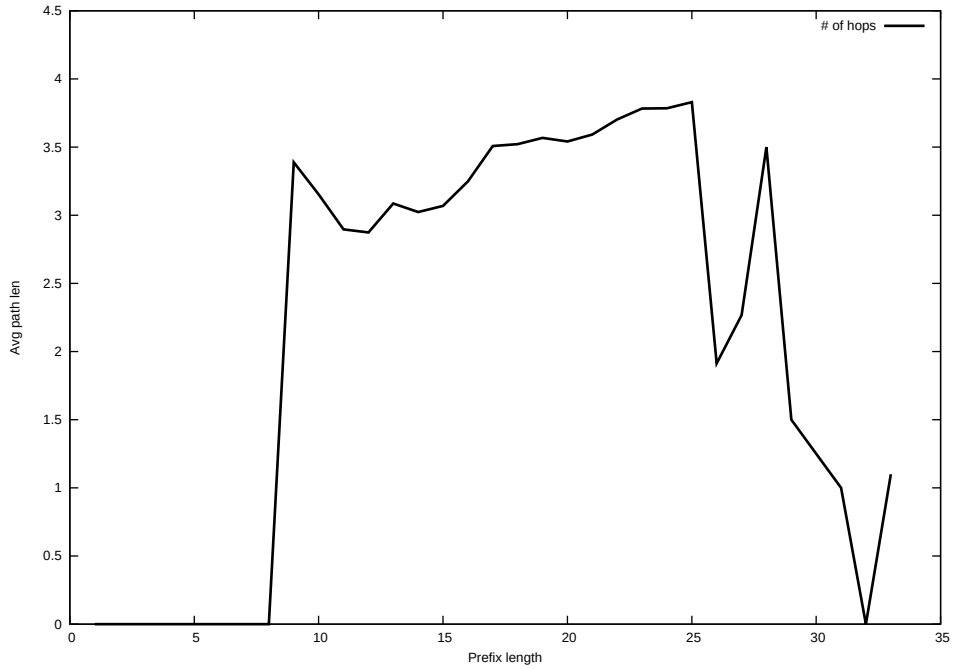
2013-01-06



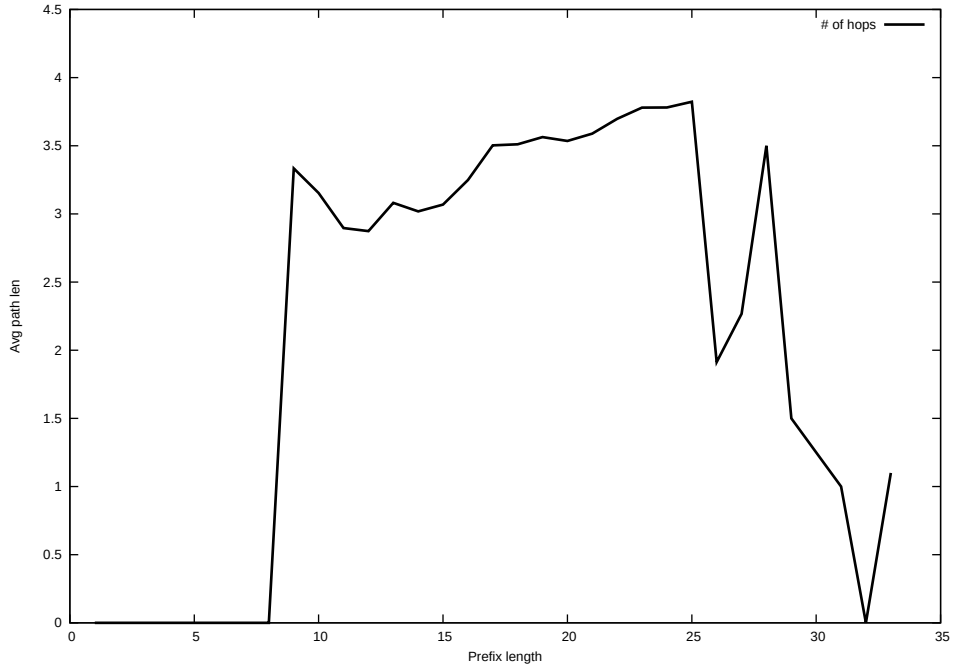
2013-01-07



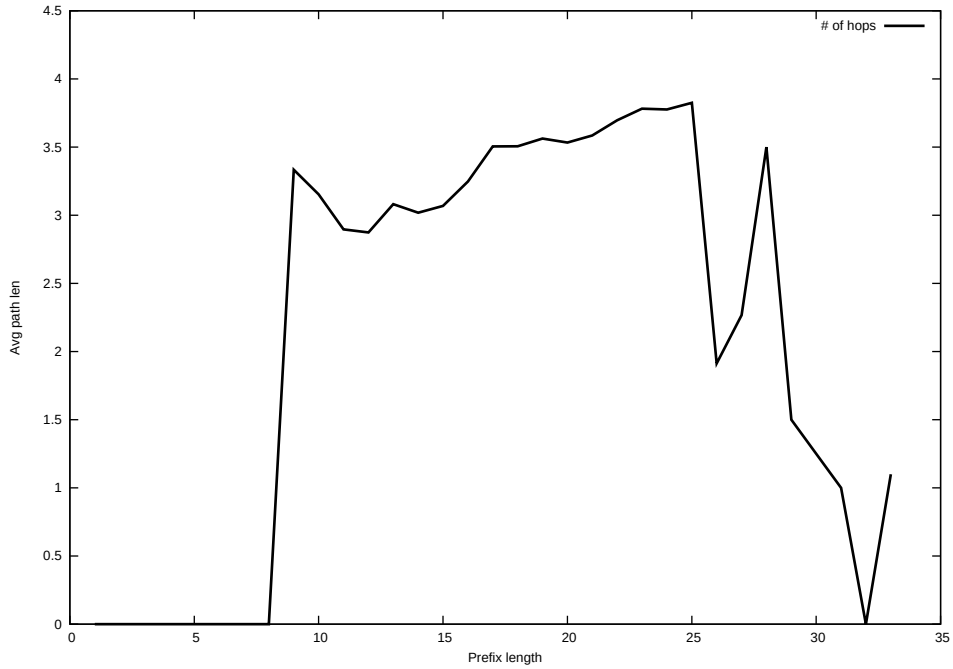
2013-01-08



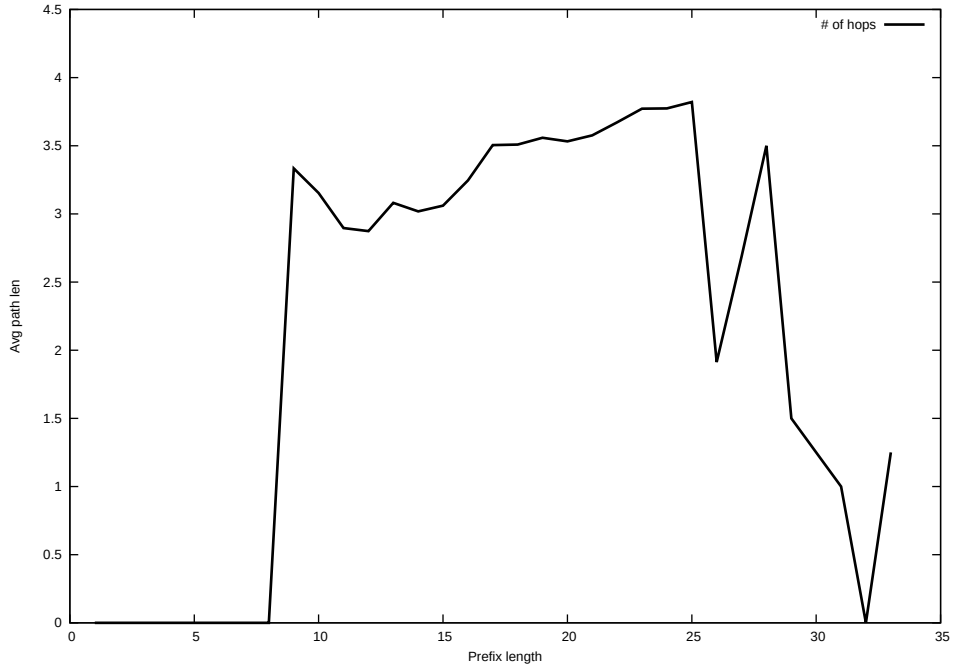
2013-01-09



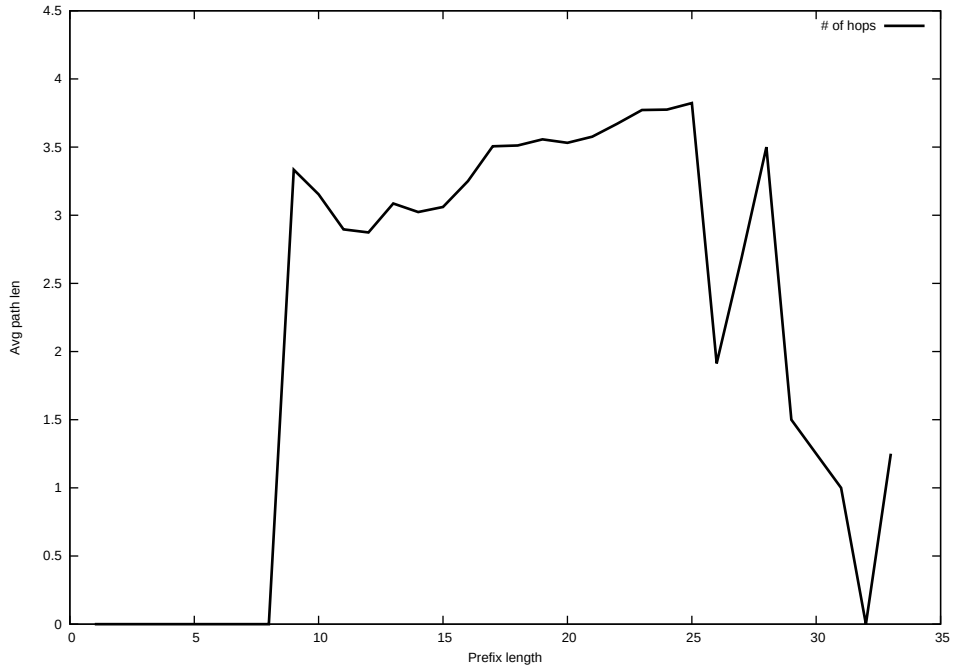
2013-01-10



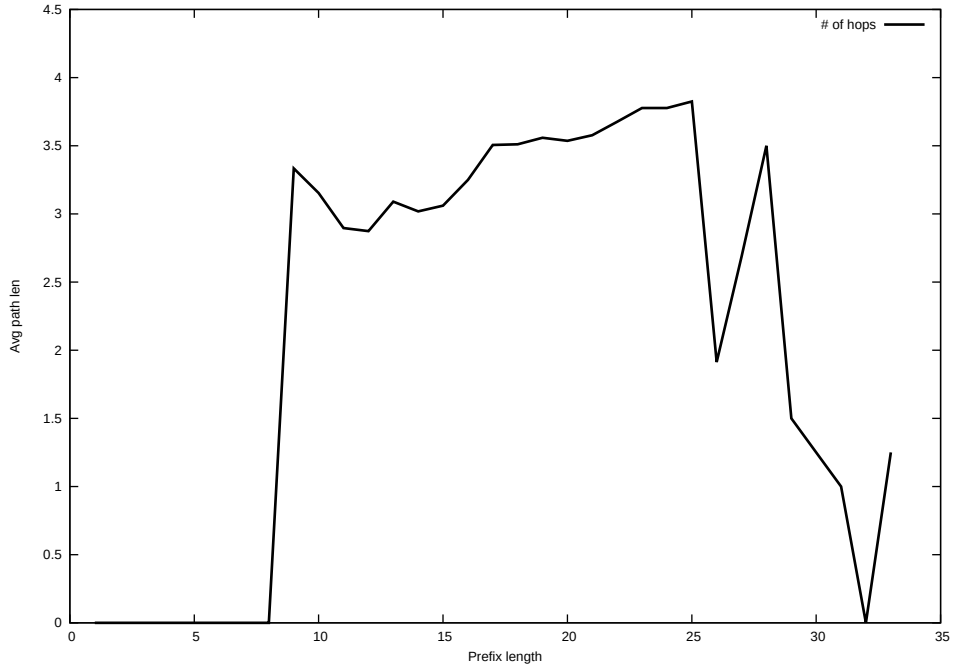
2013-01-11



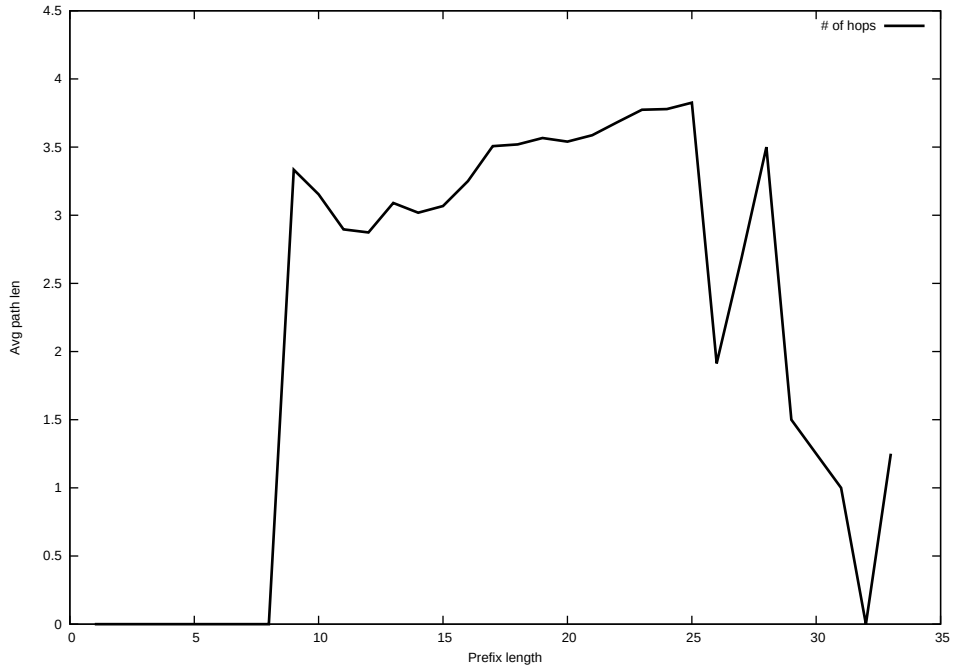
2013-01-12



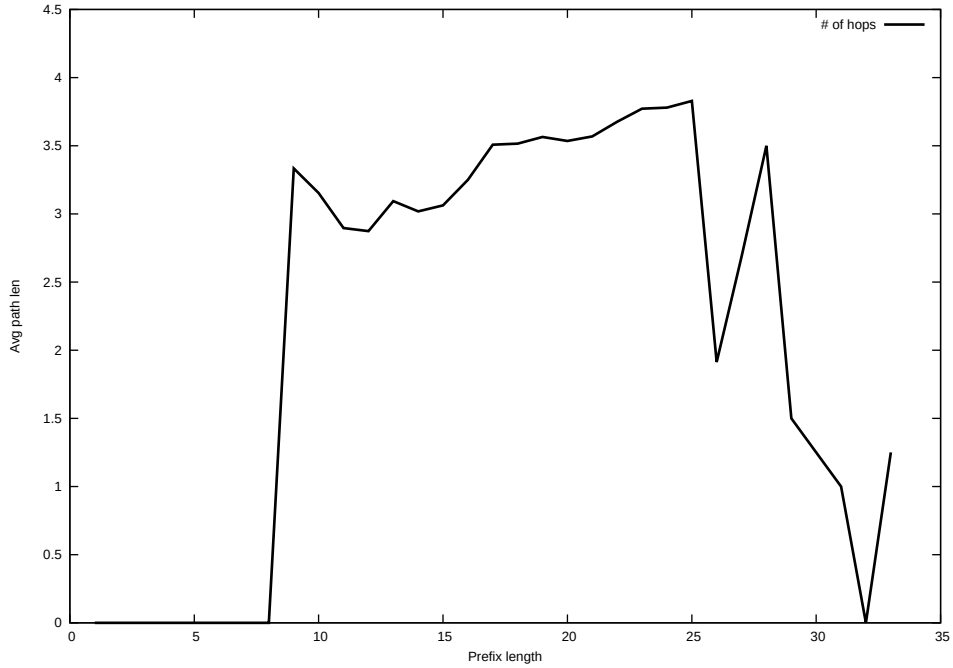
2013-01-13



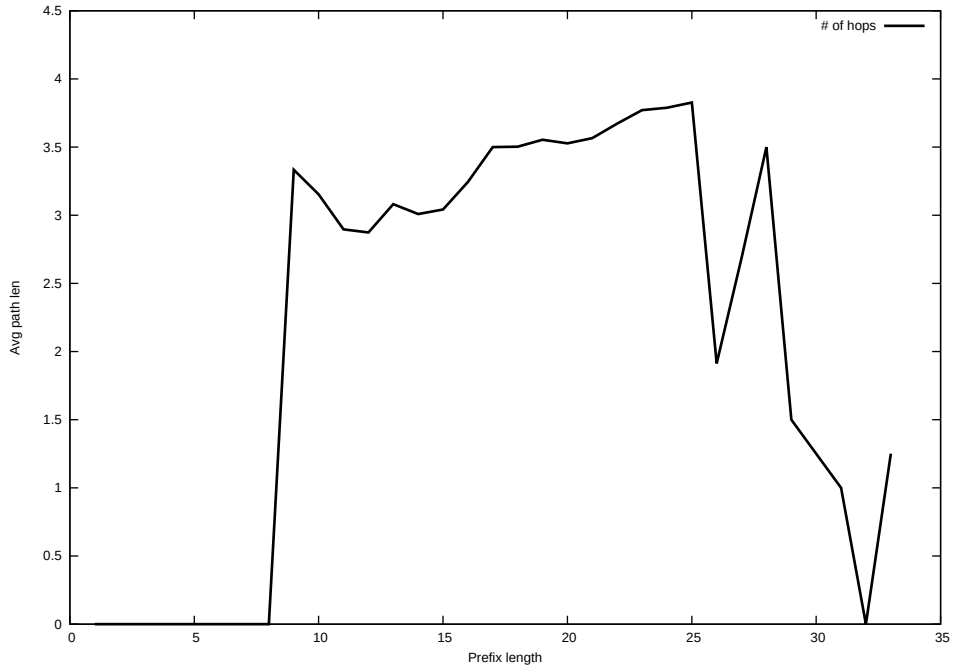
2013-01-14



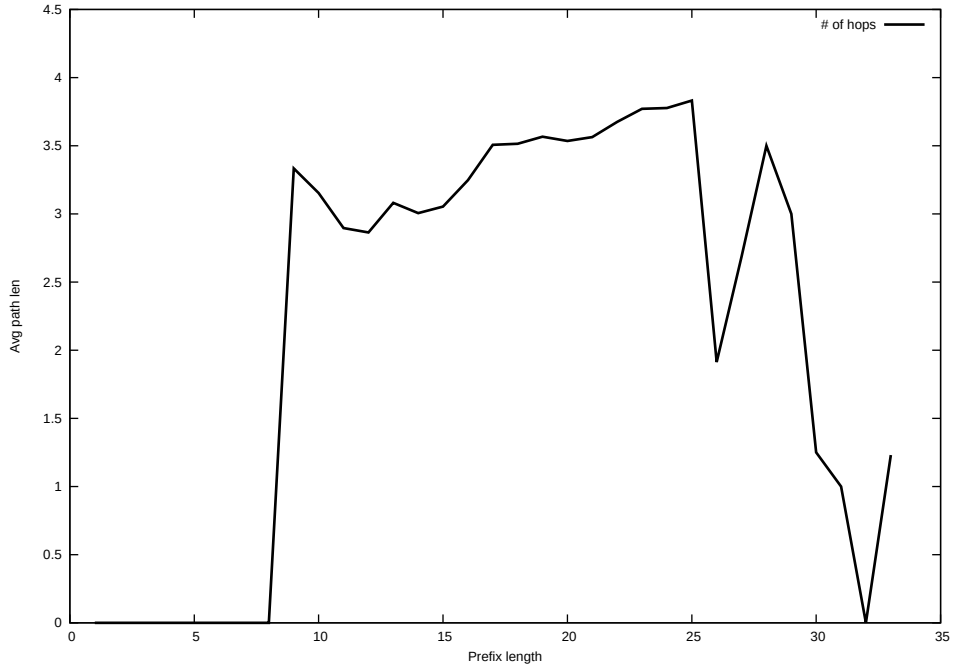
2013-01-15



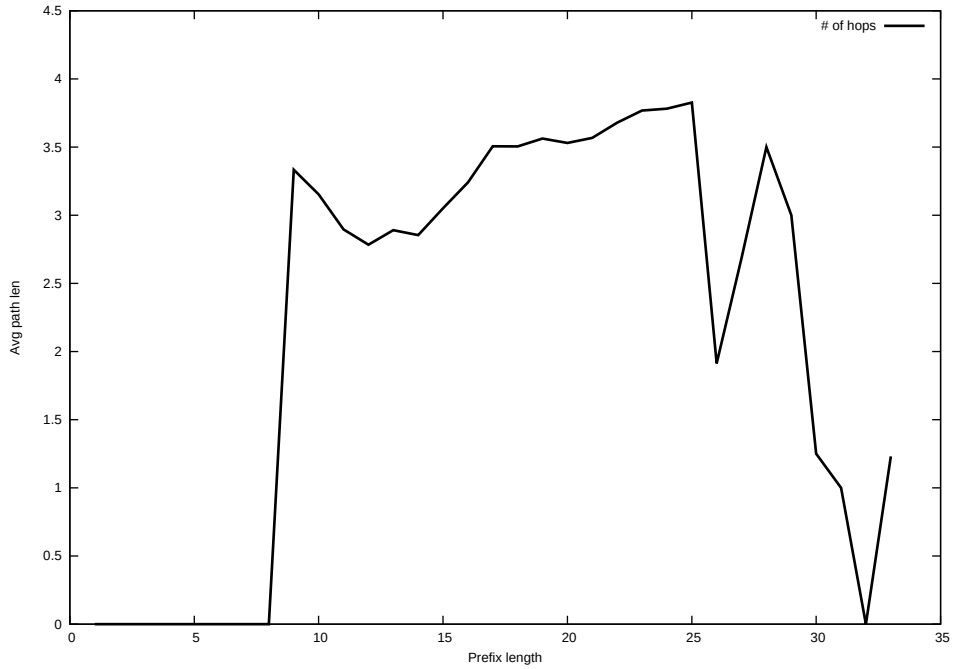
2013-01-16



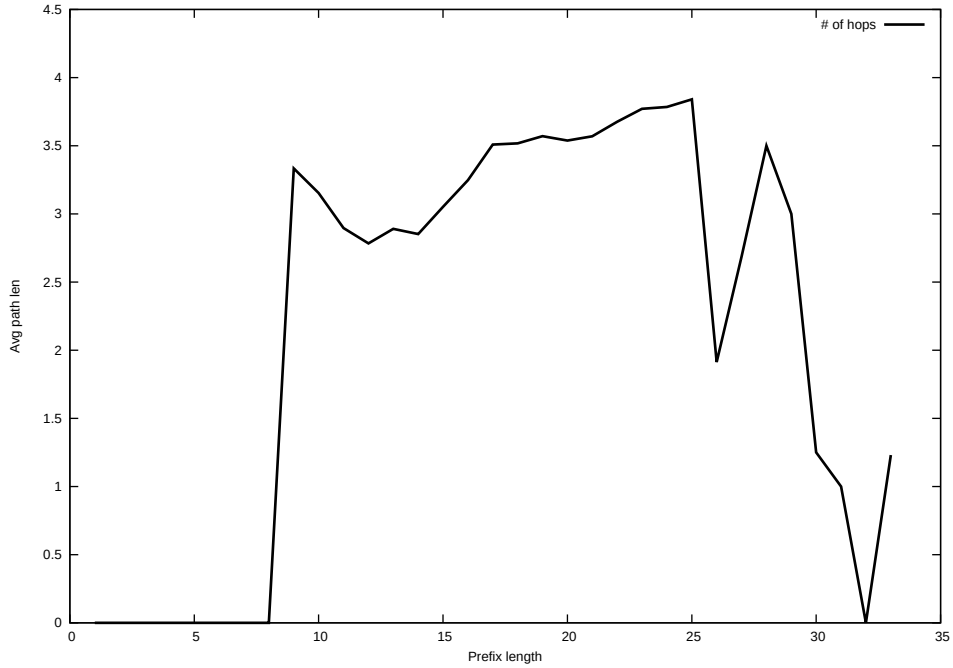
2013-01-17



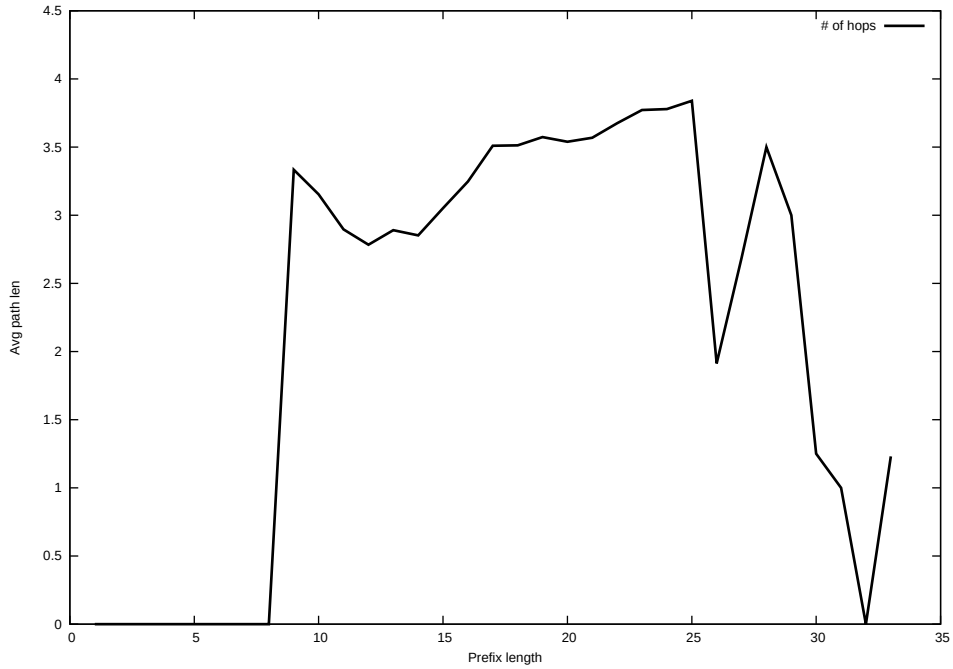
2013-01-18



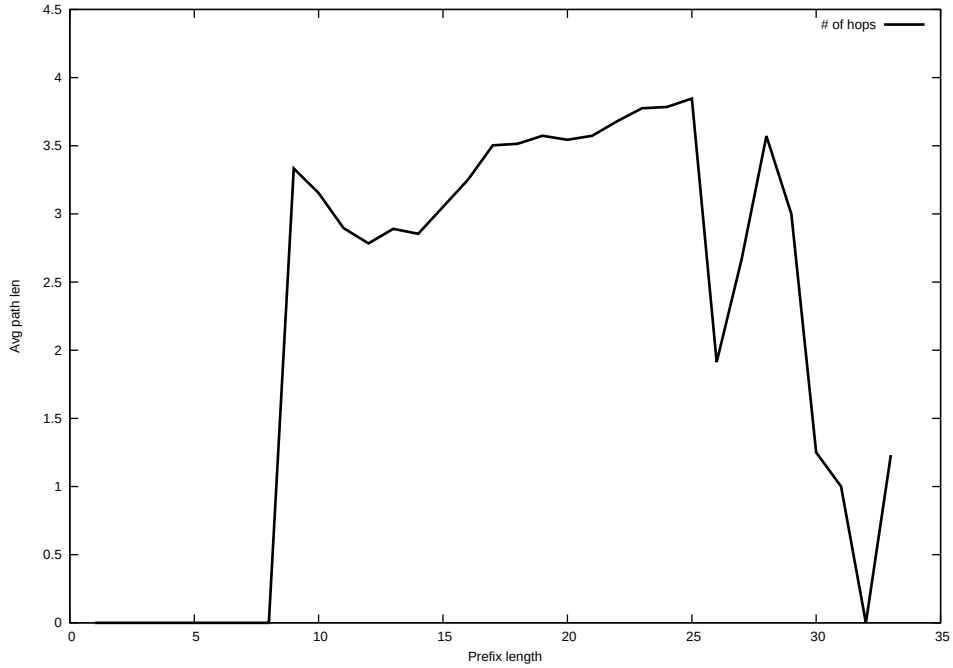
2013-01-19



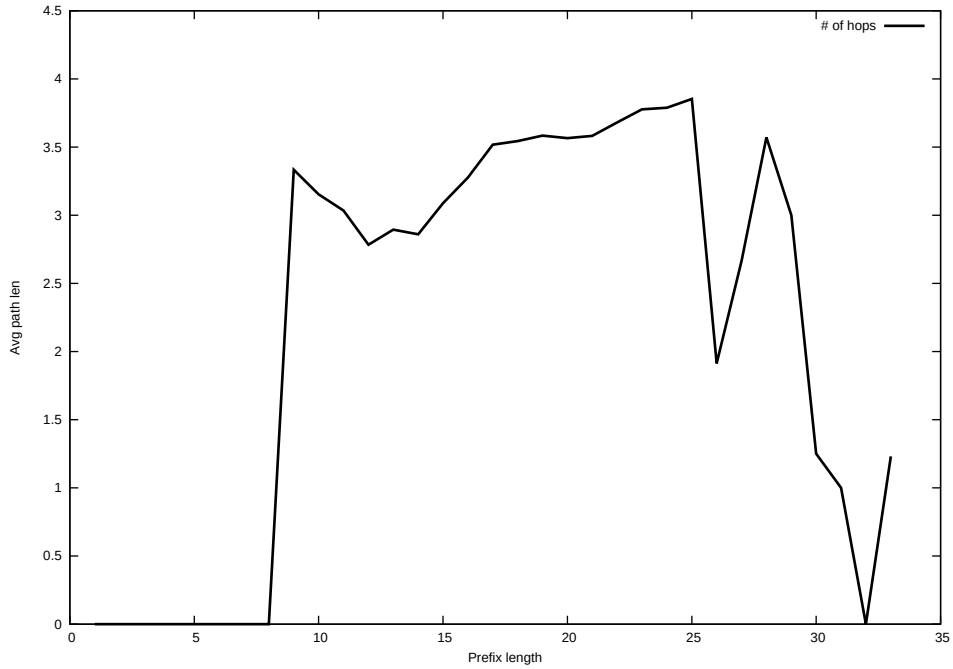
2013-01-20



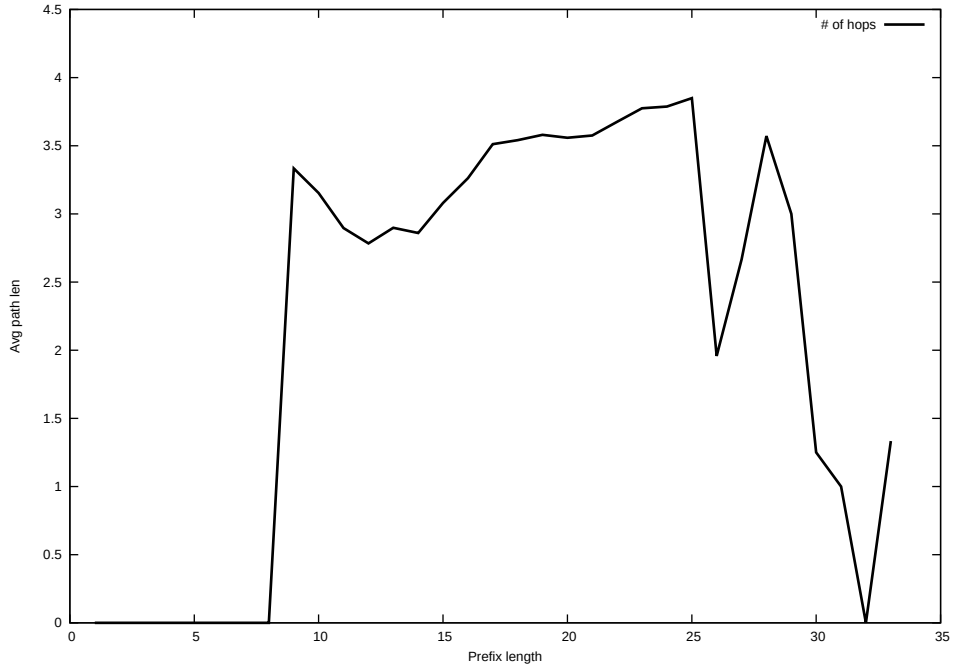
2013-01-21



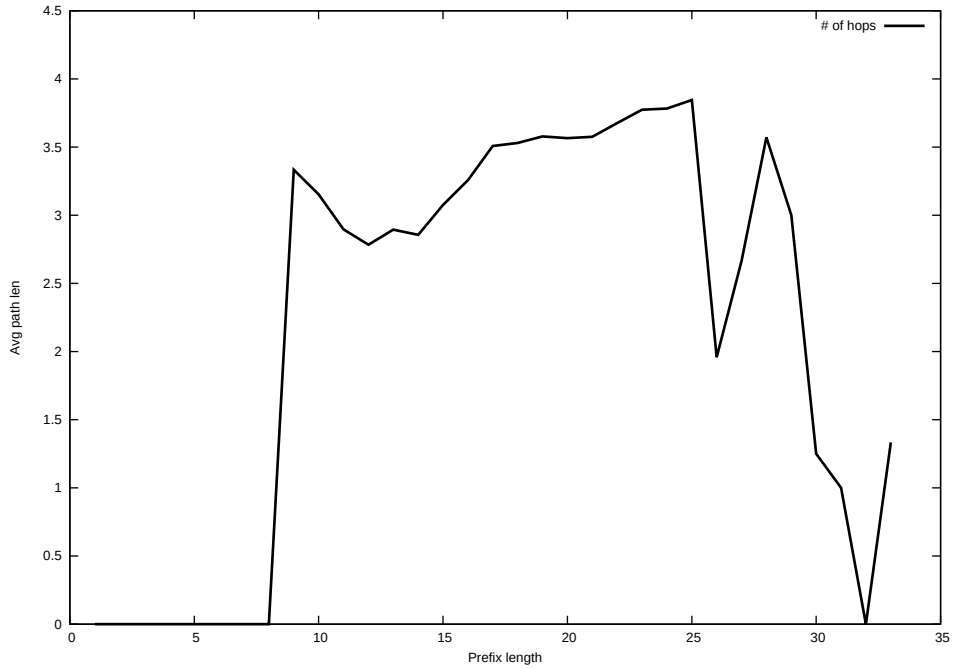
2013-01-22



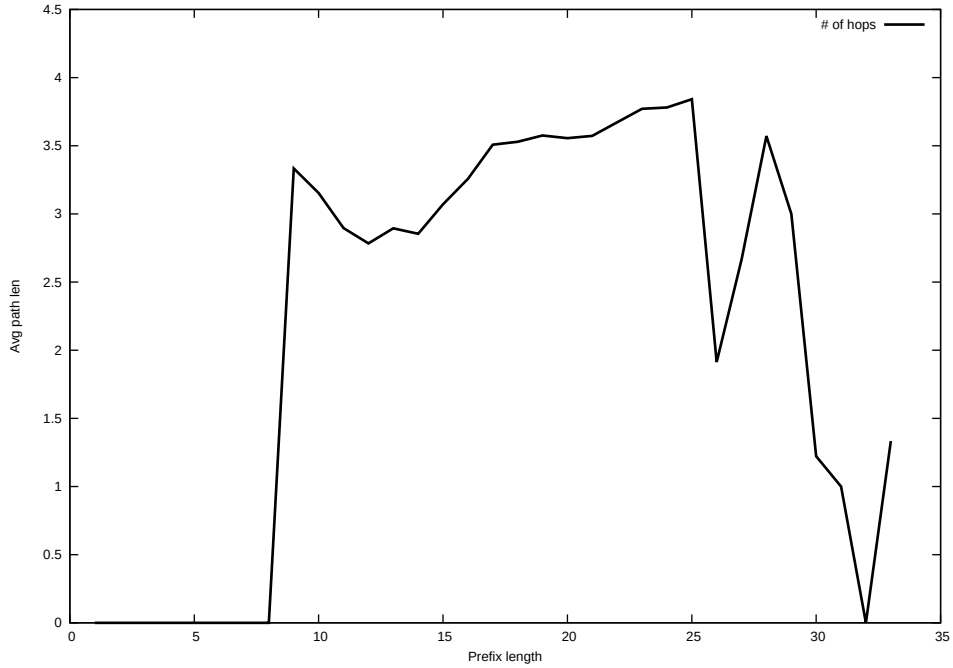
2013-01-23



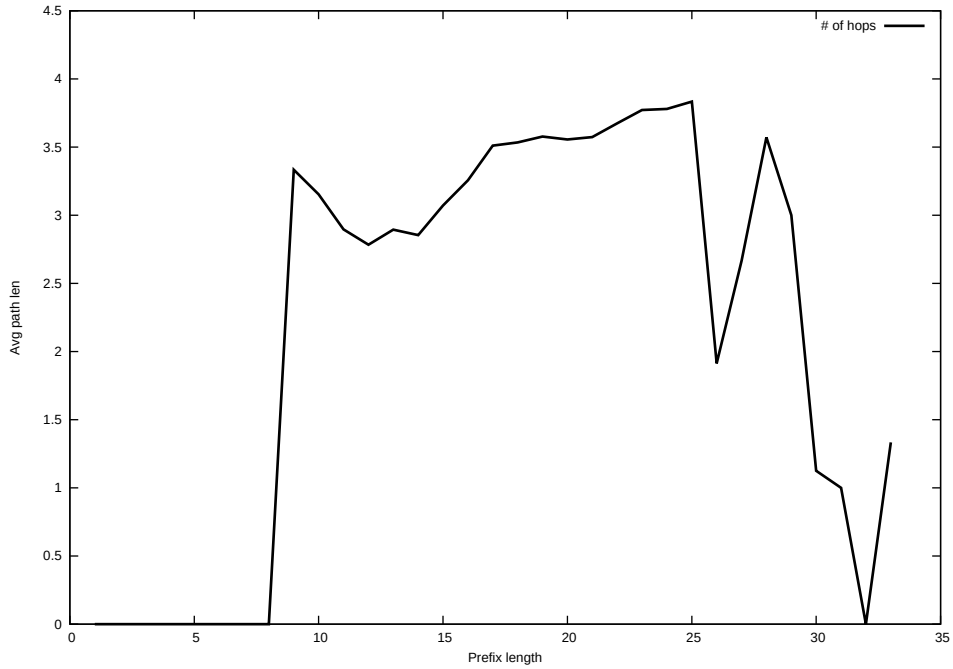
2013-01-24



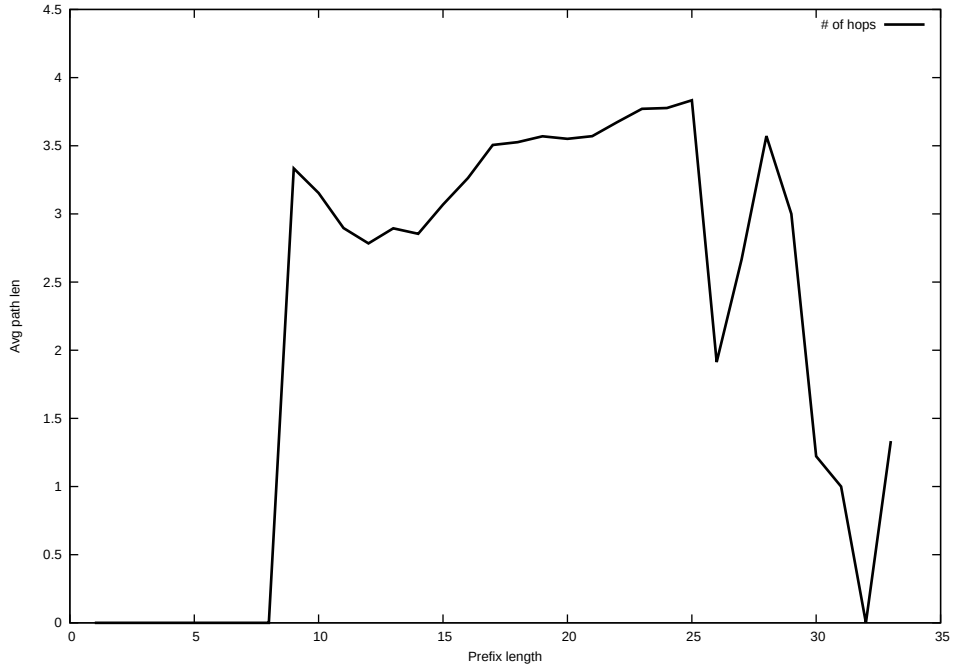
2013-01-25



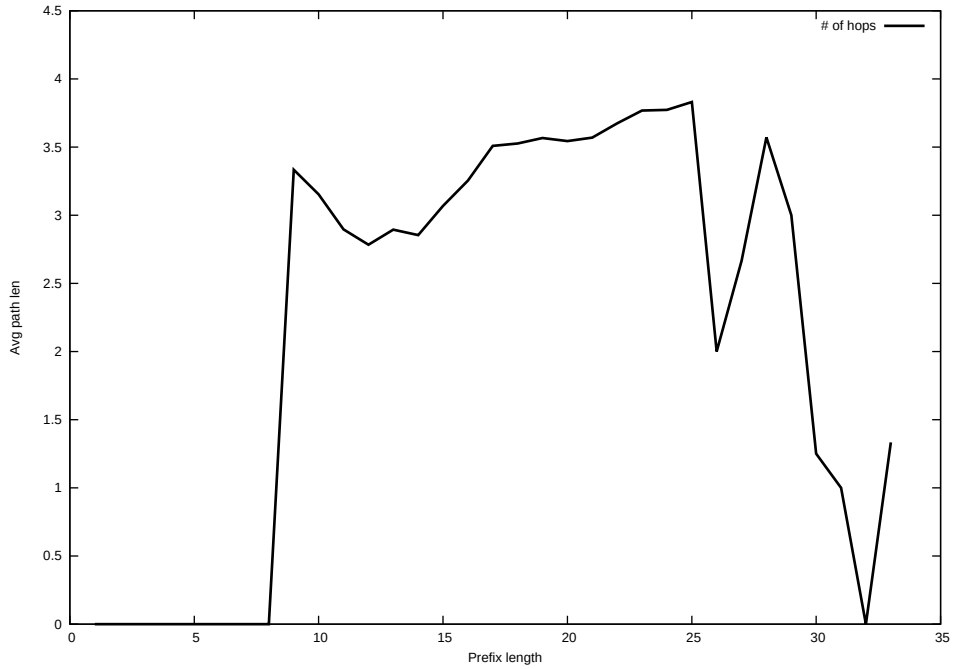
2013-01-26



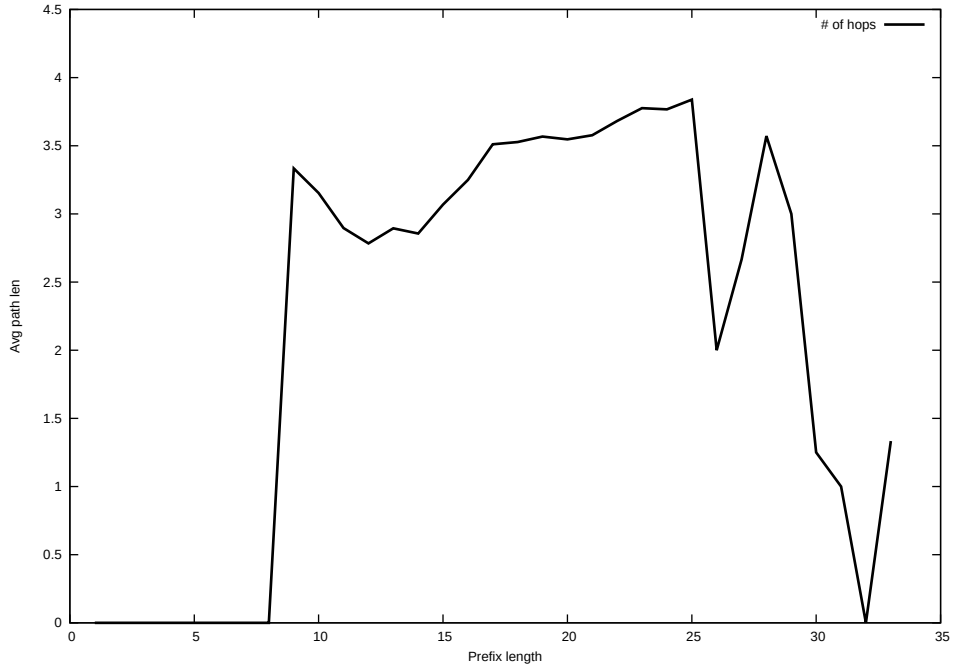
2013-01-27



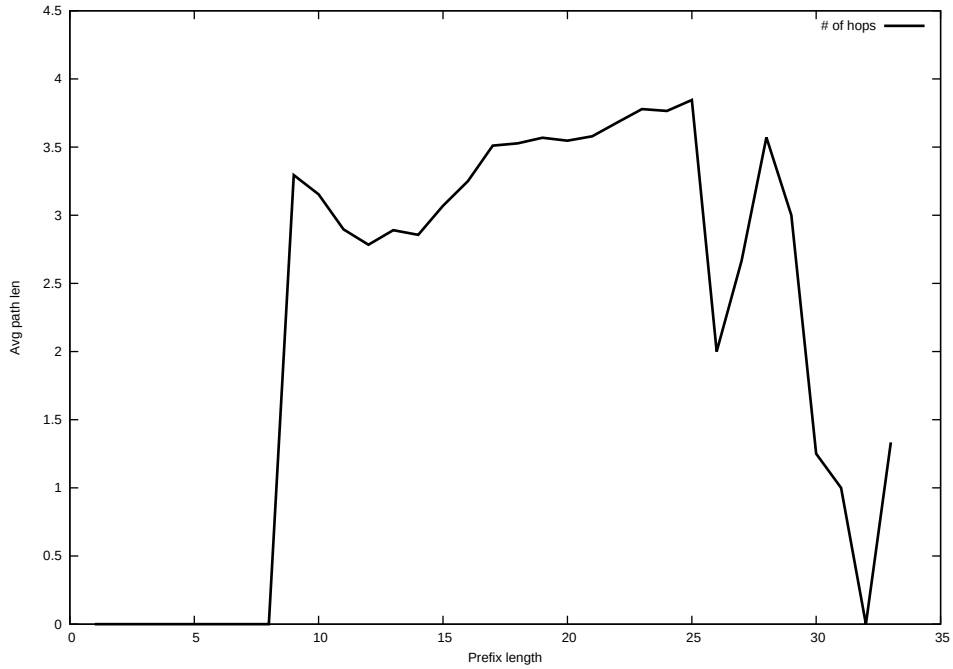
2013-01-28



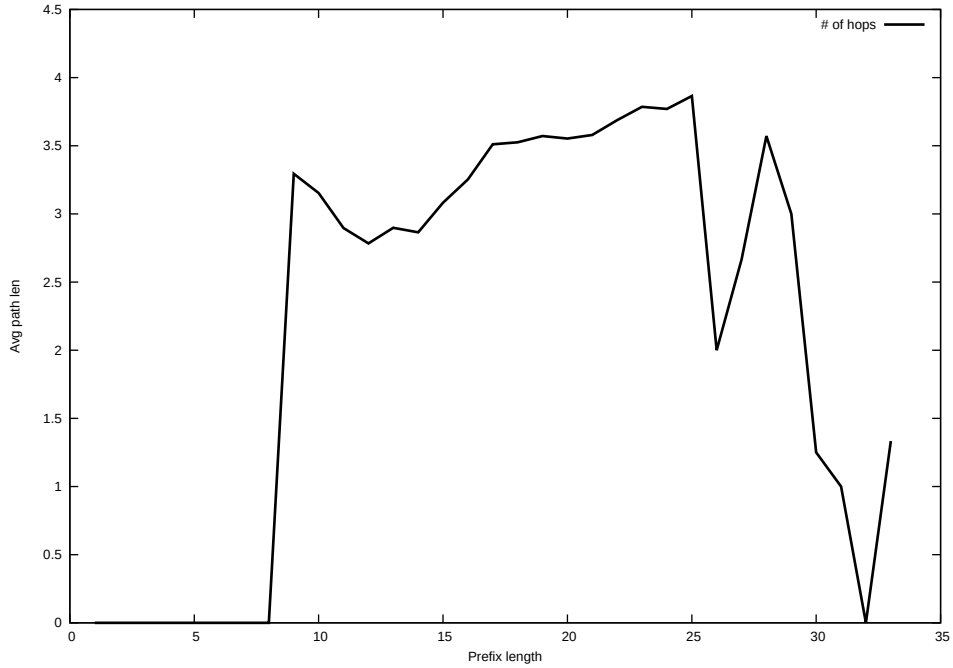
2013-01-29



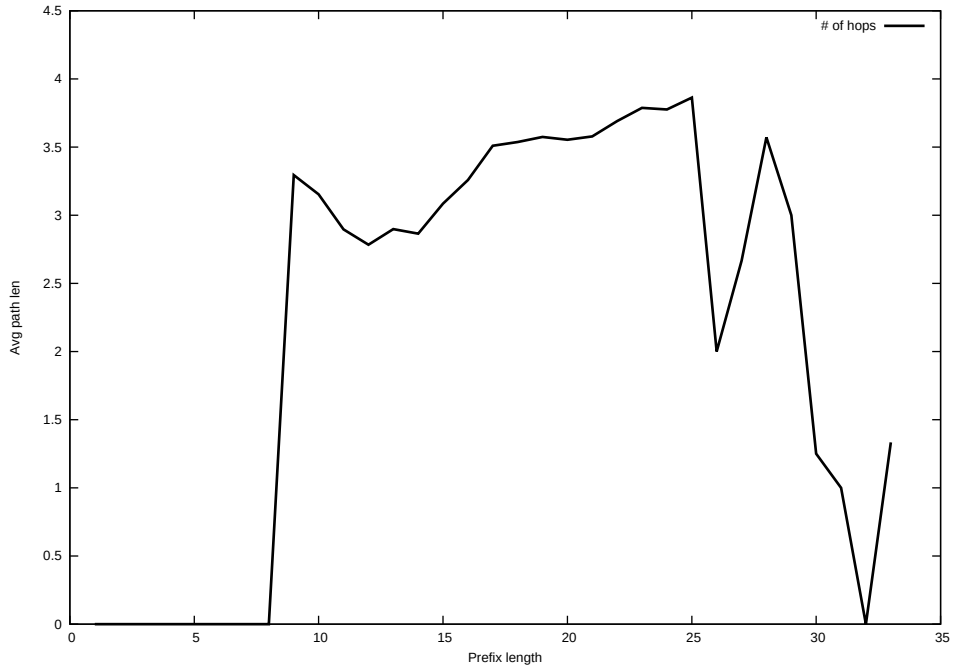
2013-01-30



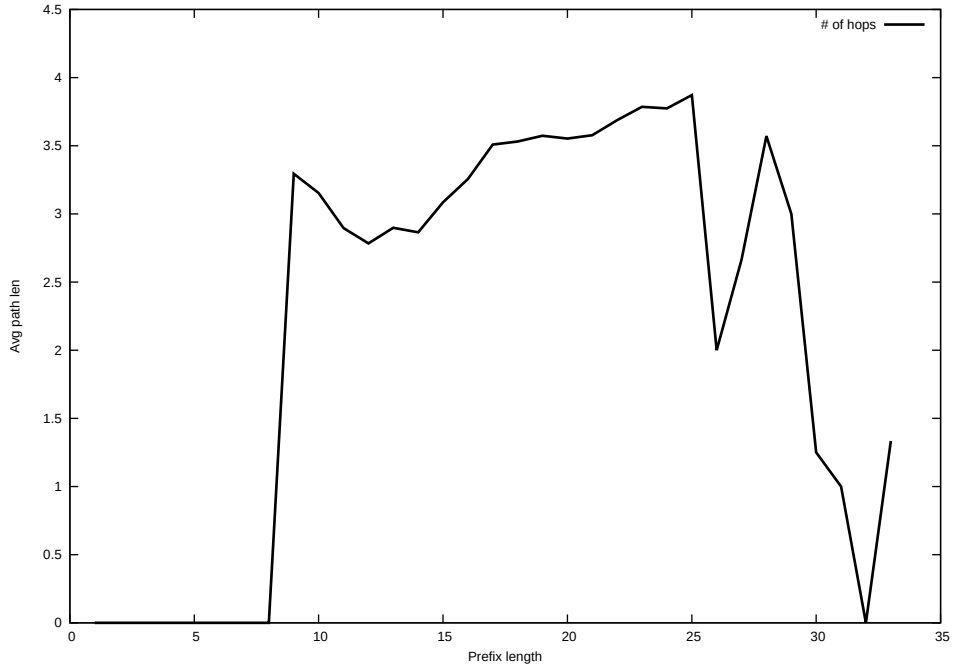
2013-01-31



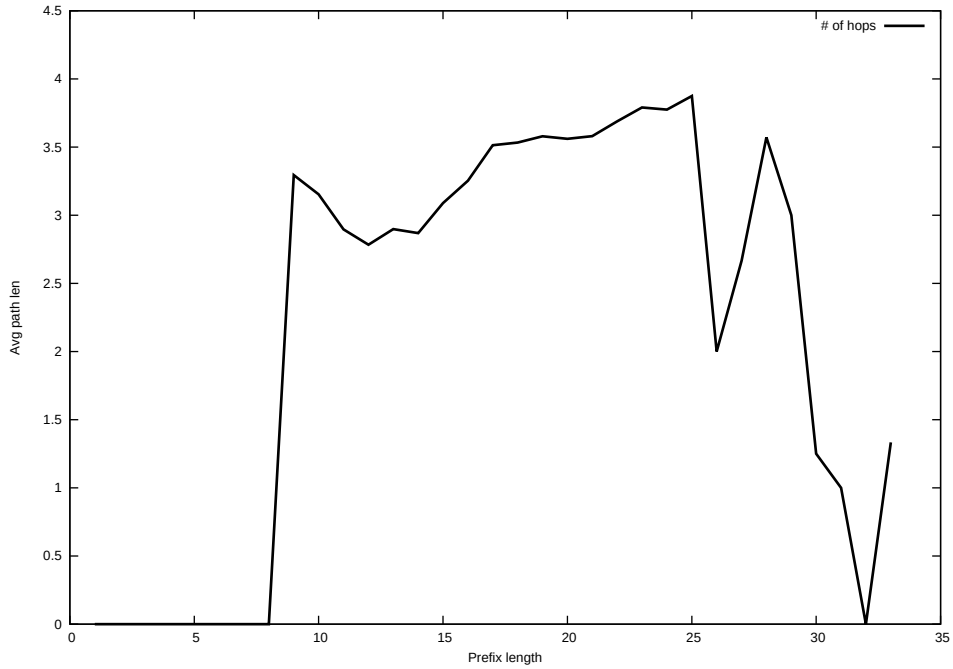
2013-02-01



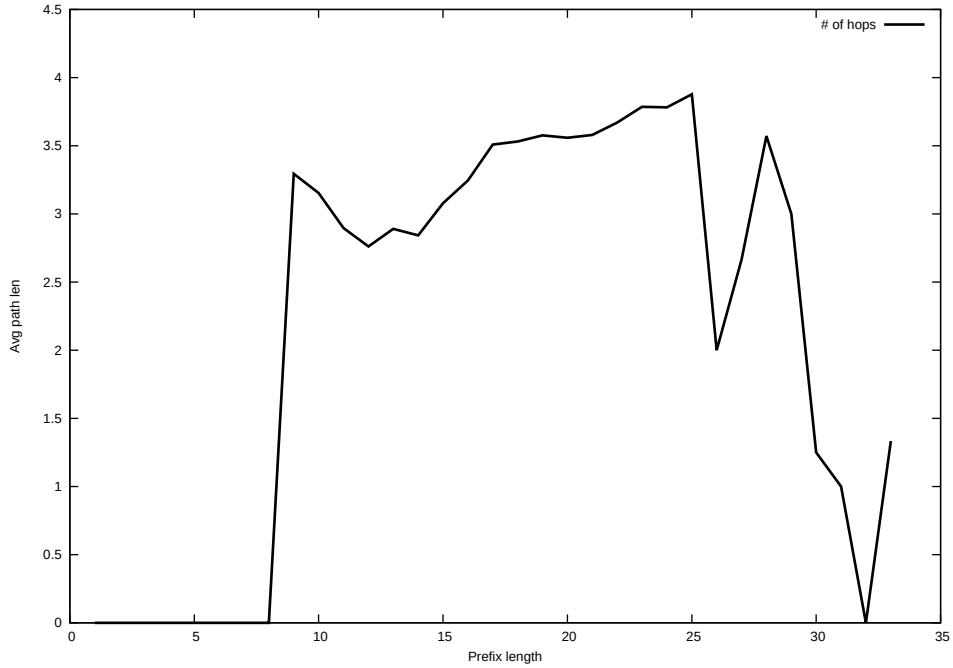
2013-02-02



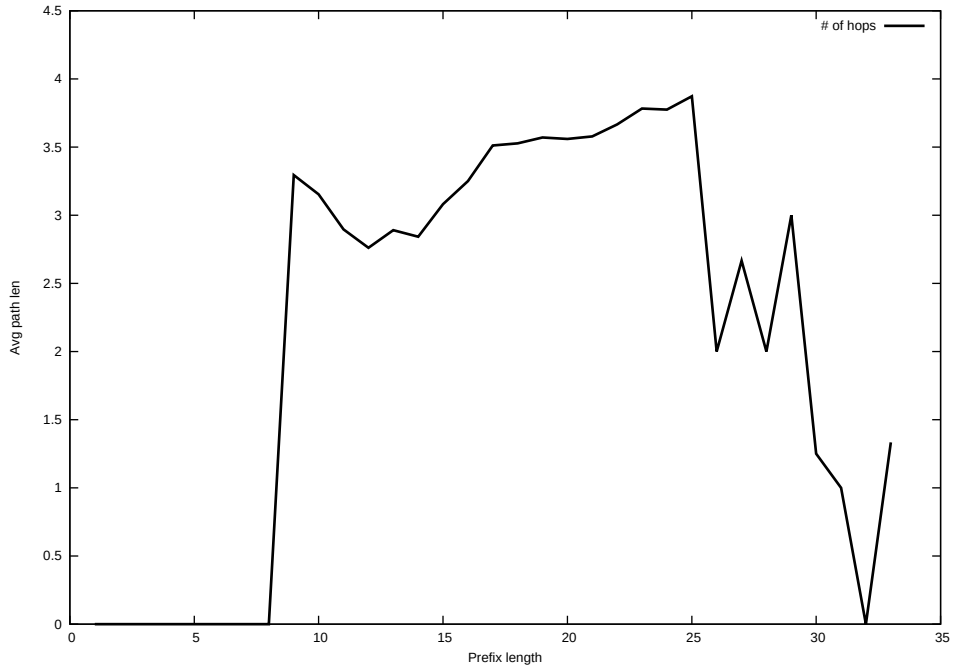
2013-02-03



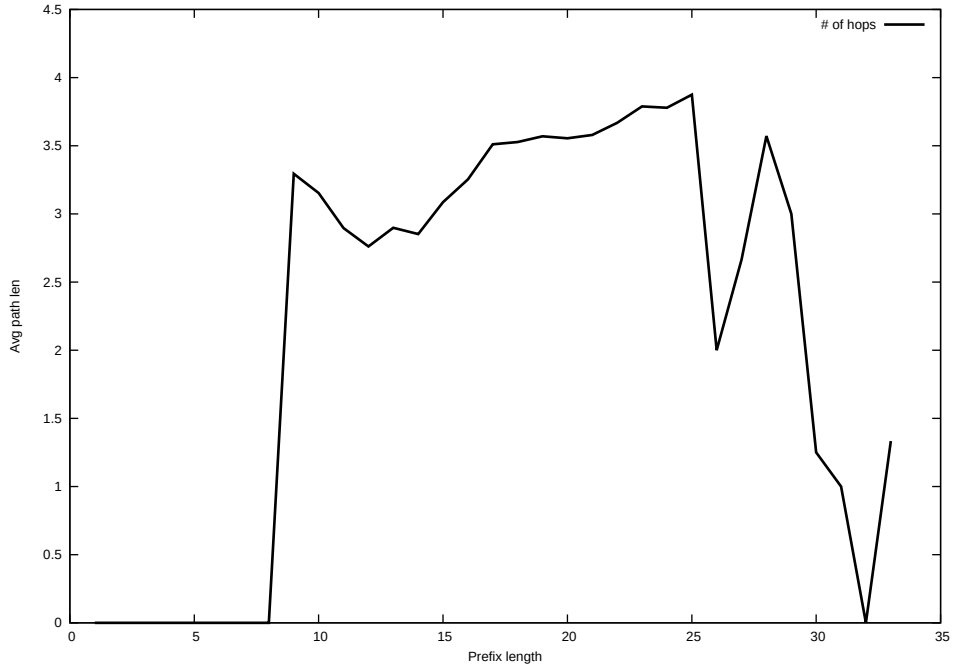
2013-02-04



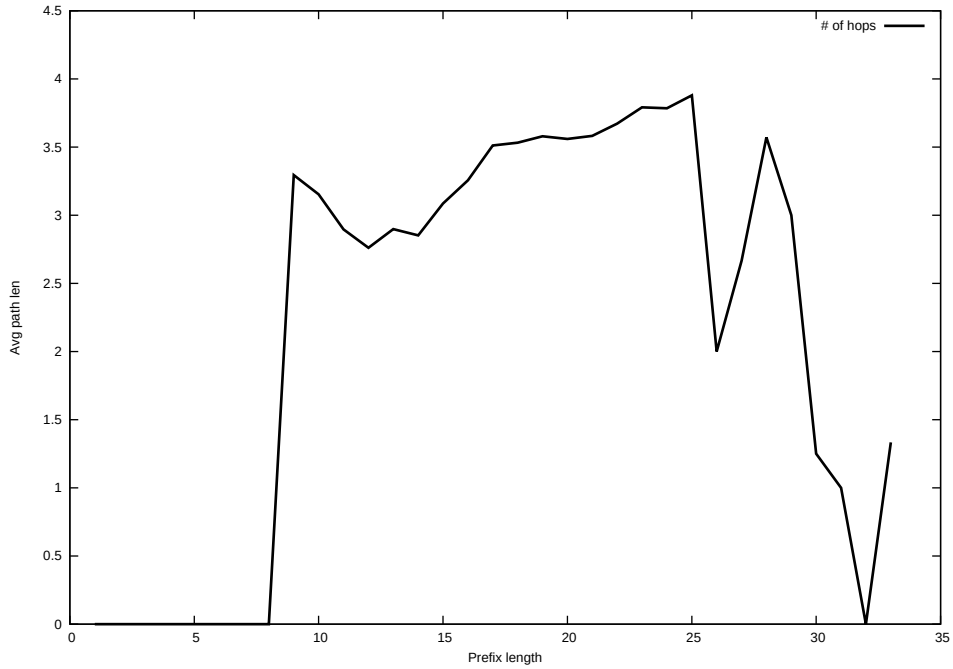
2013-02-05



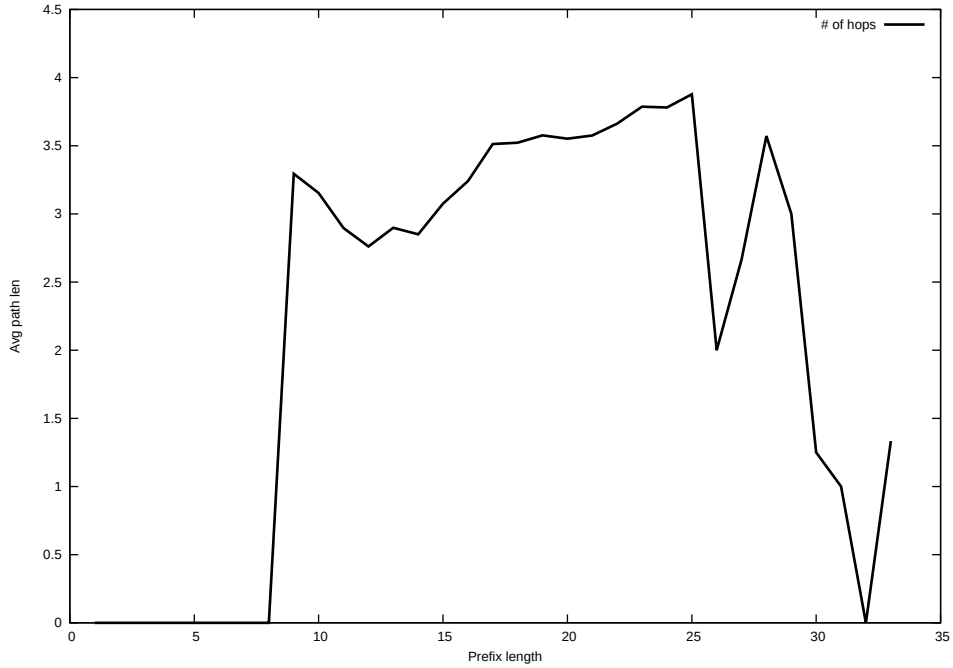
2013-02-06



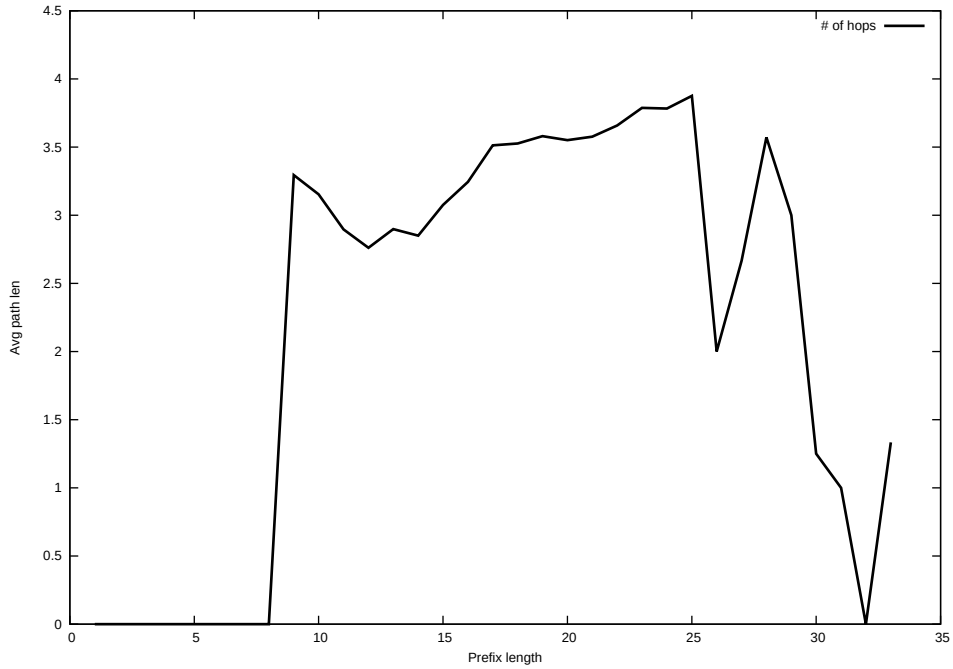
2013-02-07



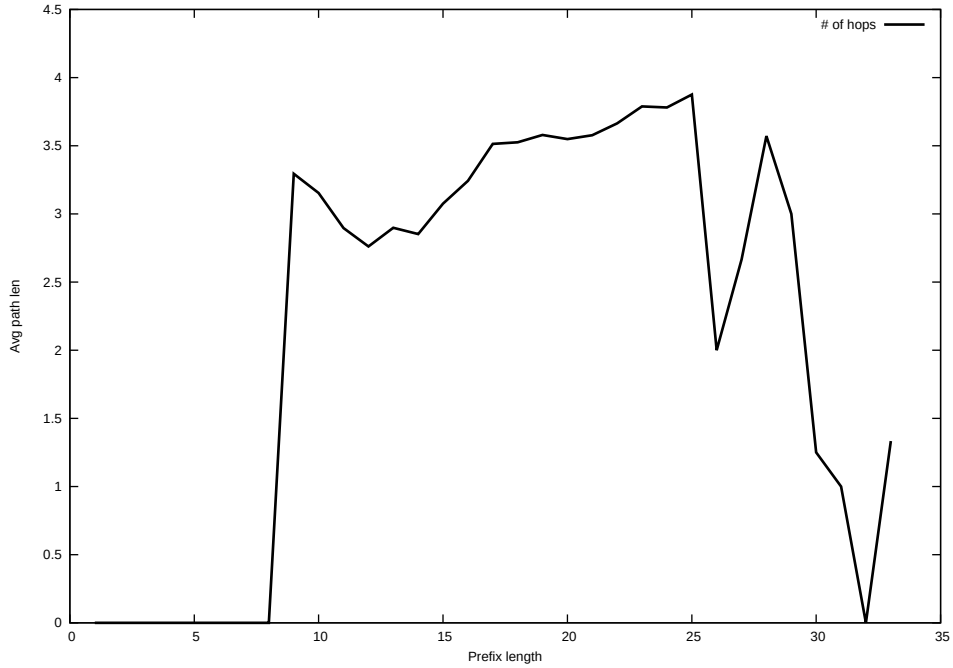
2013-02-08



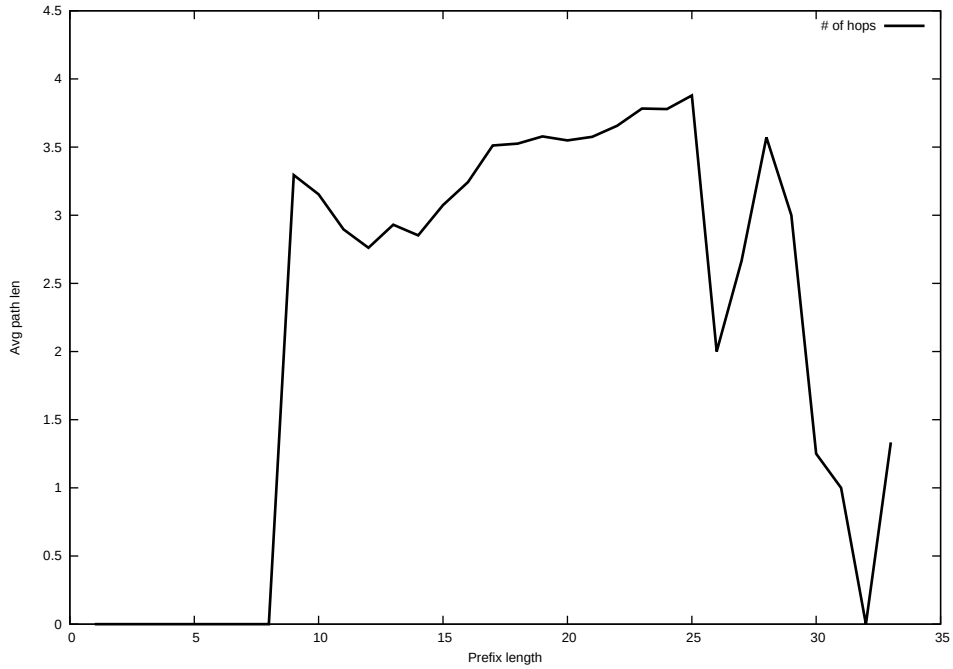
2013-02-09



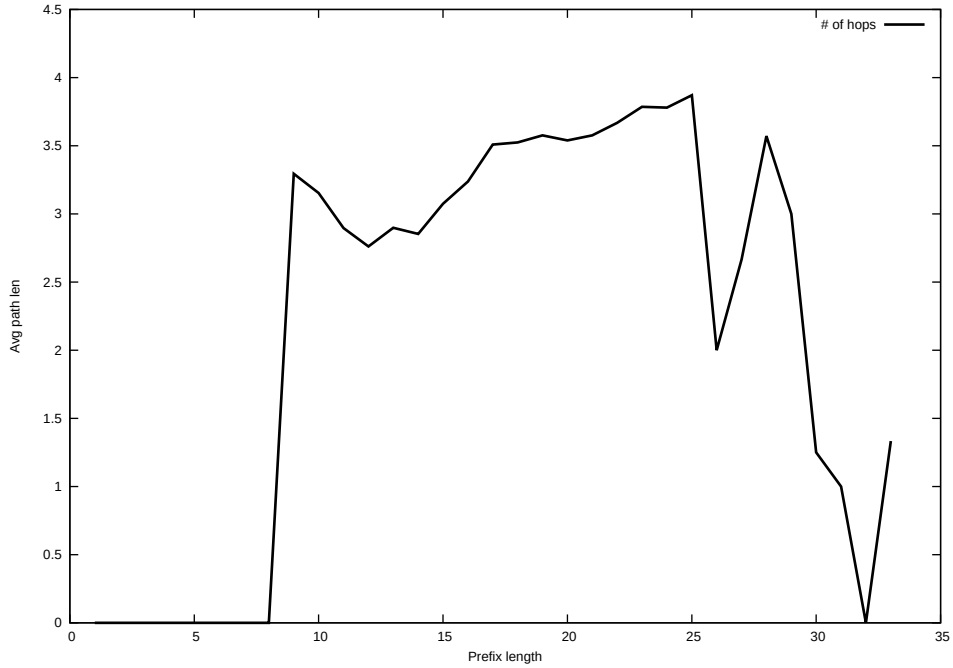
2013-02-10



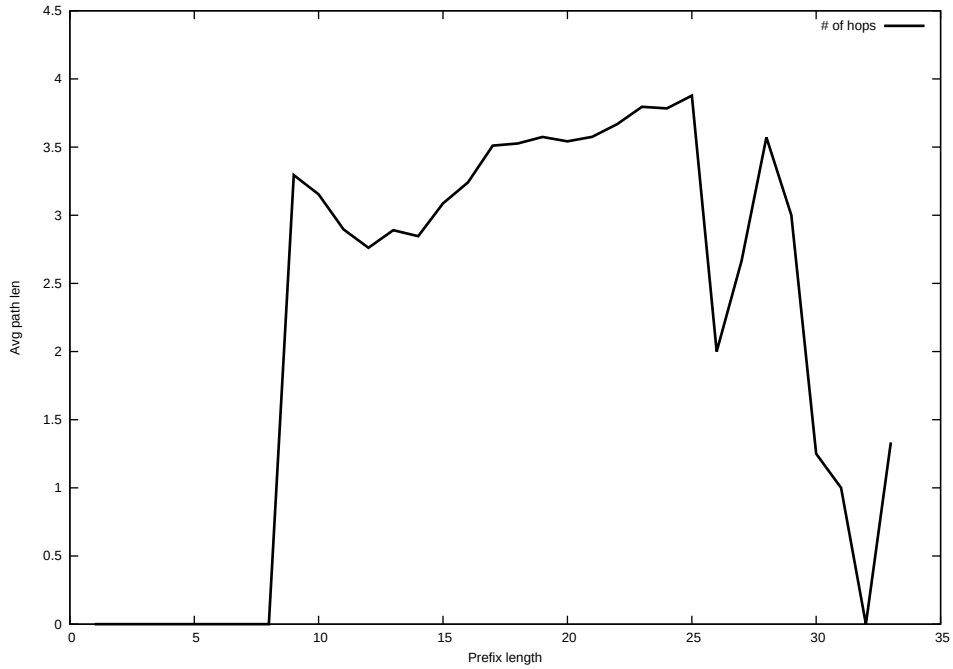
2013-02-11



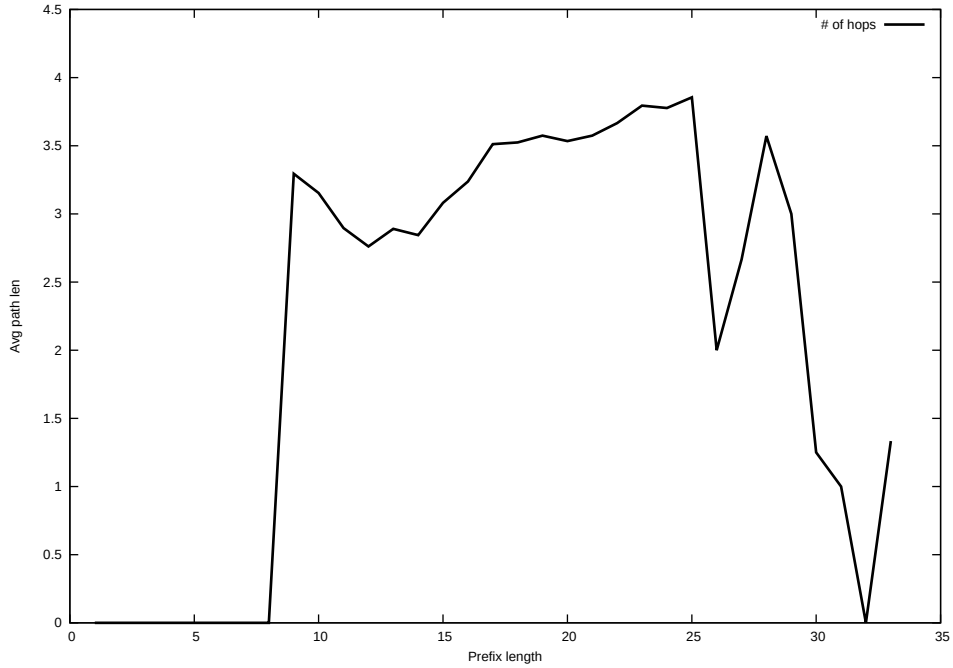
2013-02-12



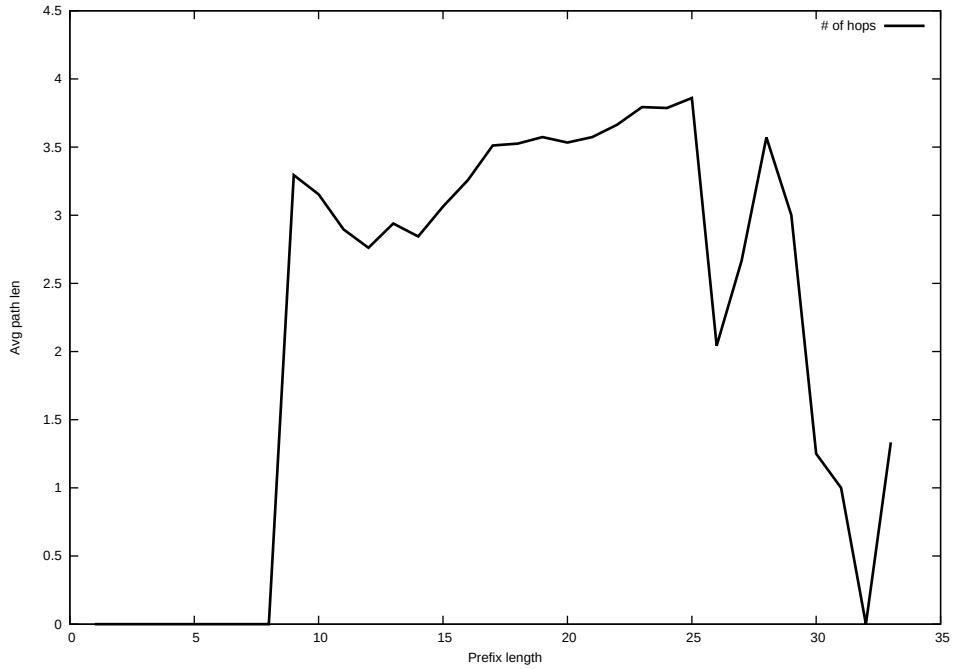
2013-02-13



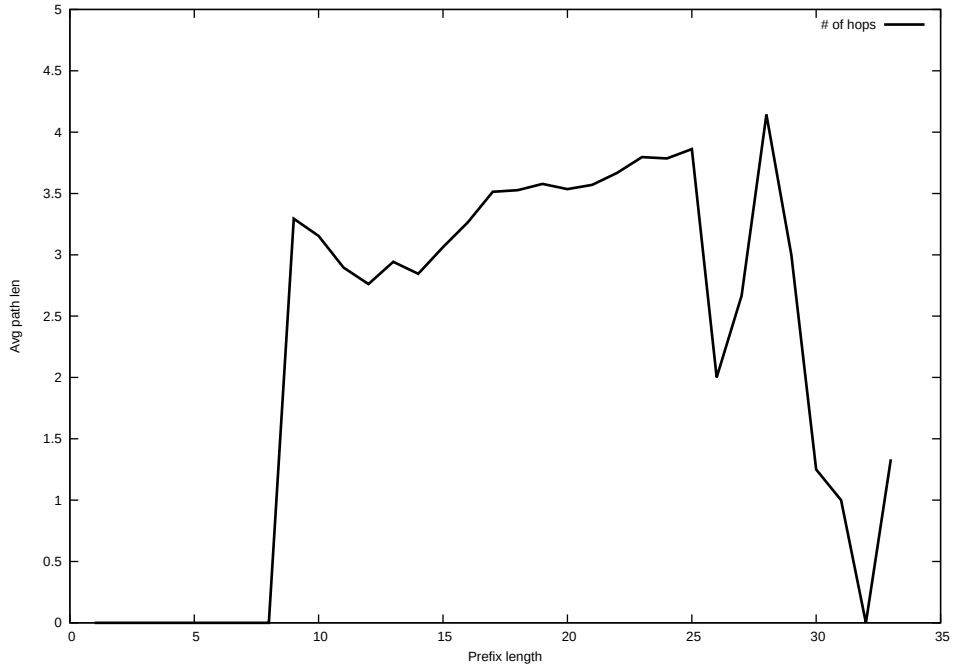
2013-02-14



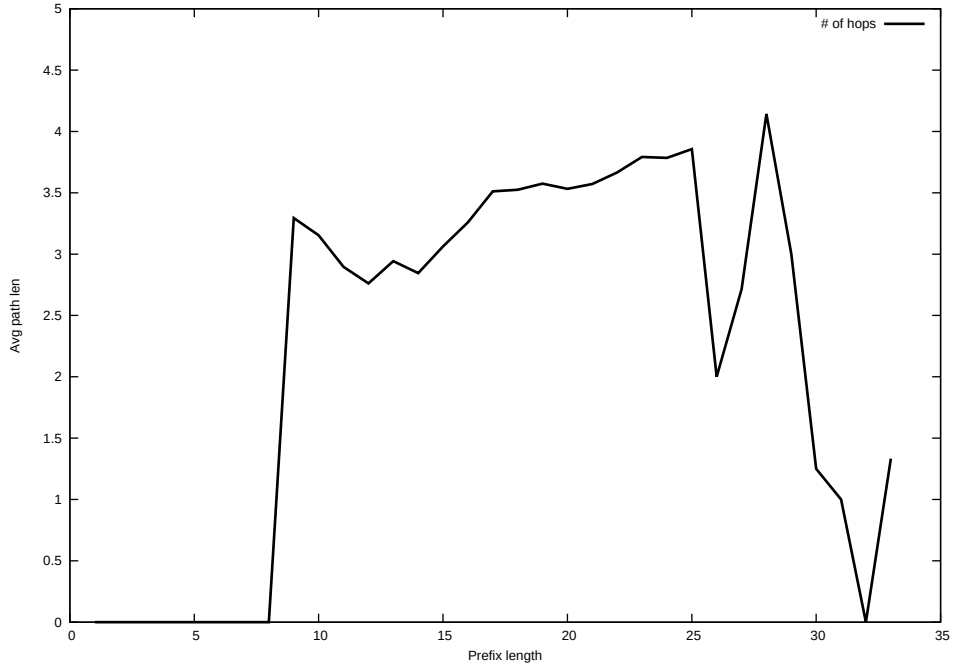
2013-02-15



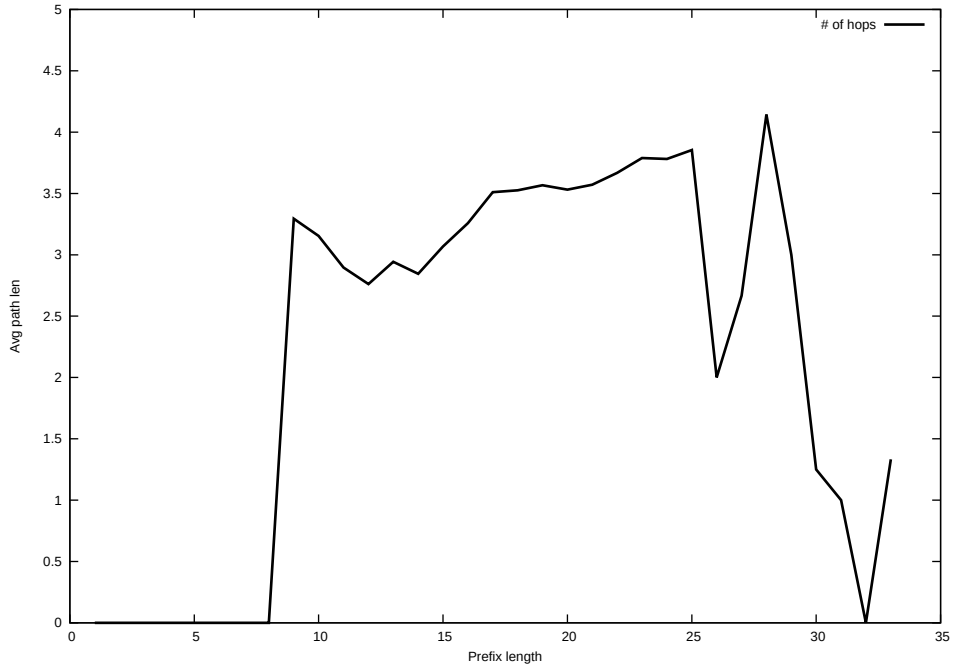
2013-02-16



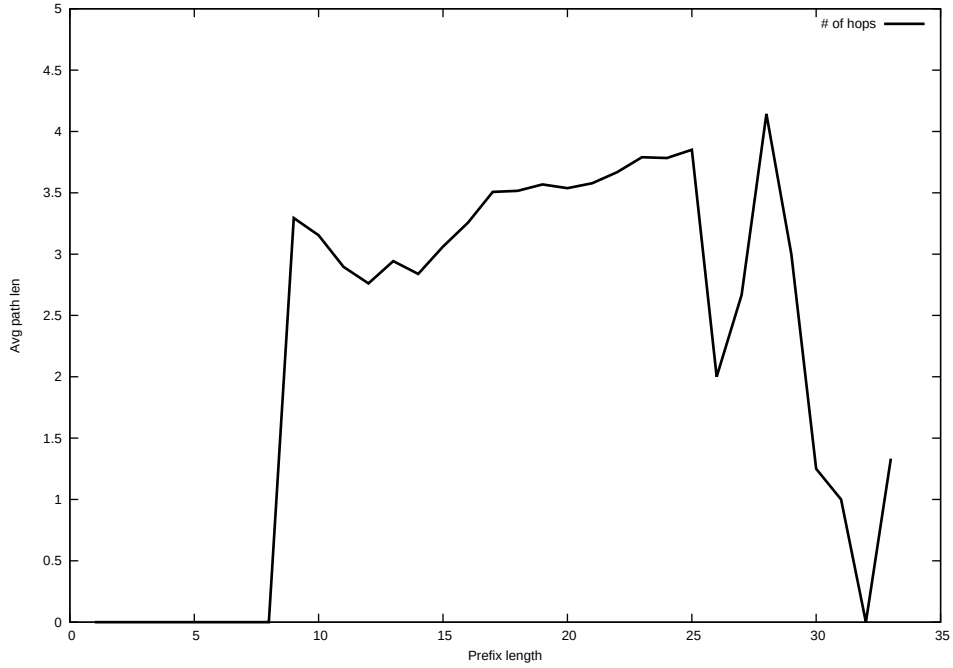
2013-02-17



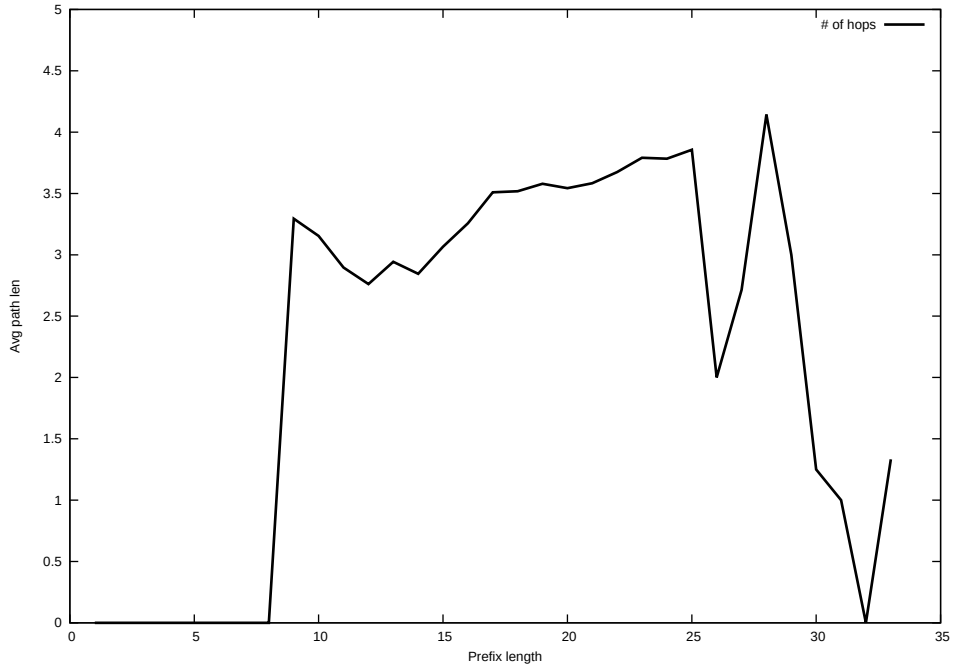
2013-02-18



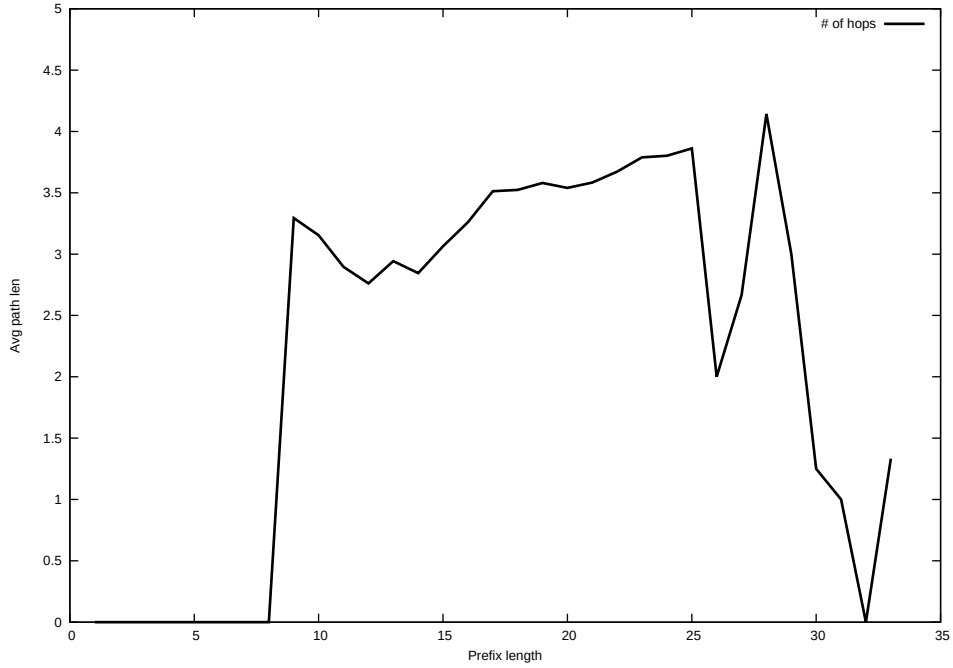
2013-02-19



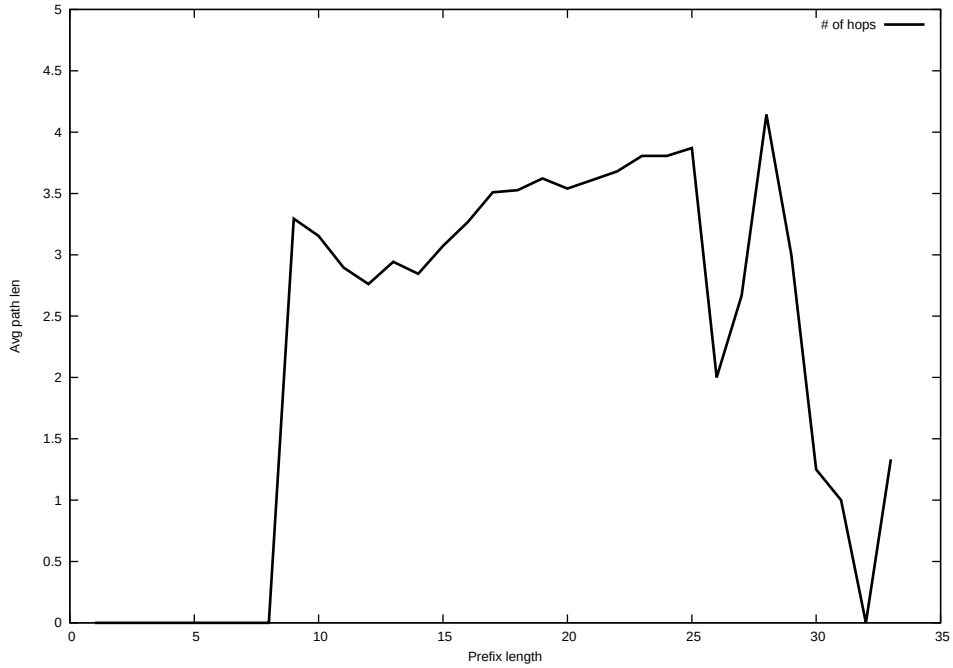
2013-02-20



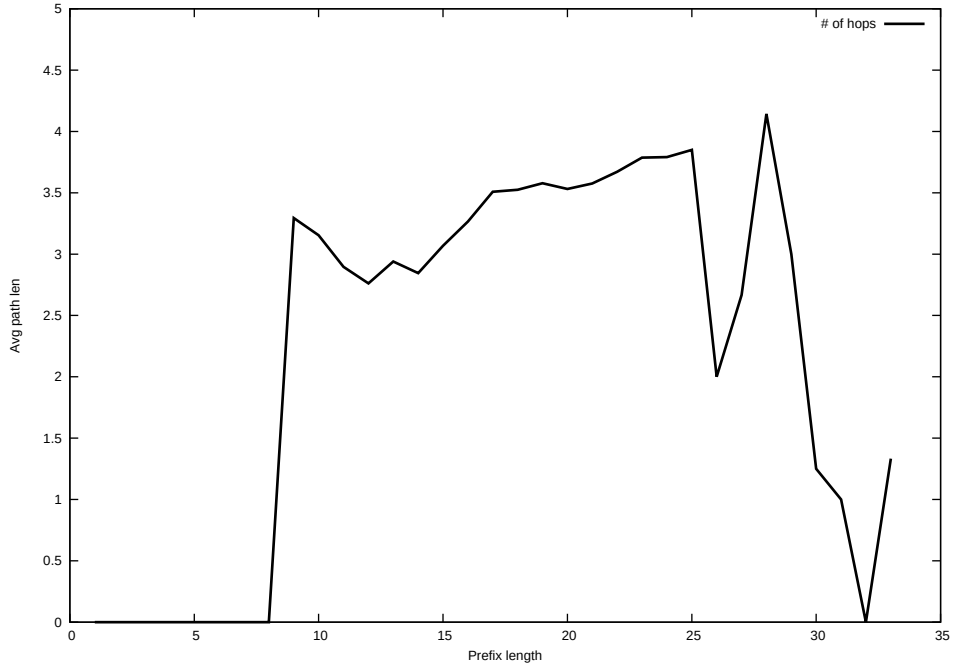
2013-02-21



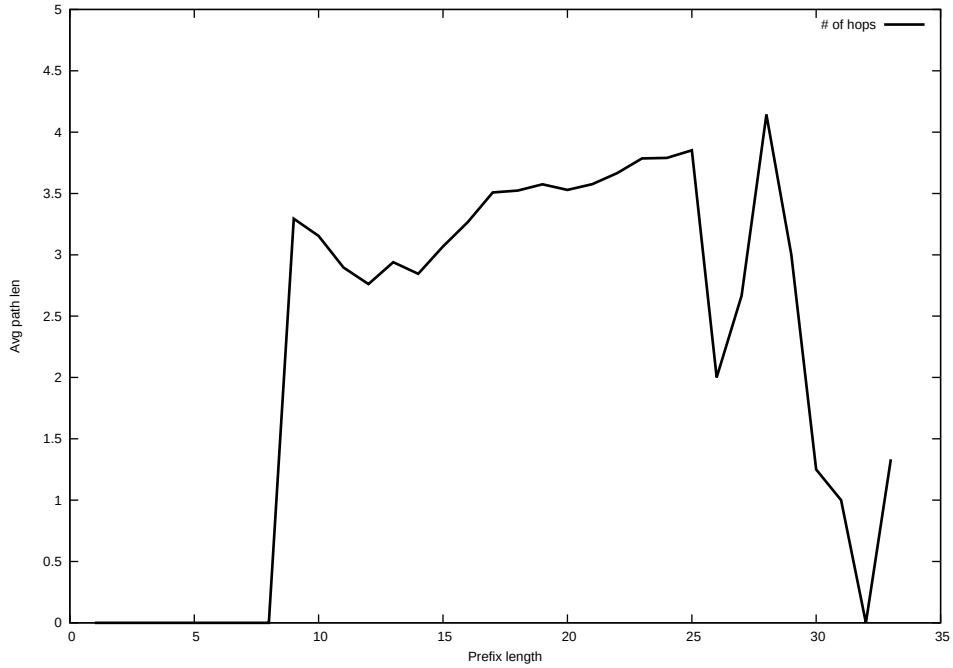
2013-02-22



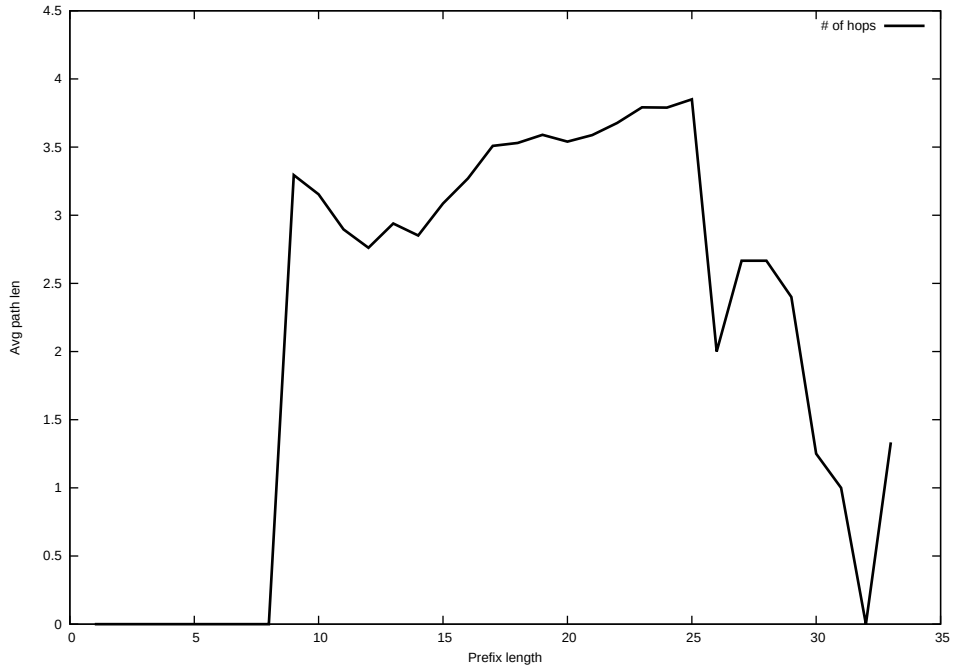
2013-02-23



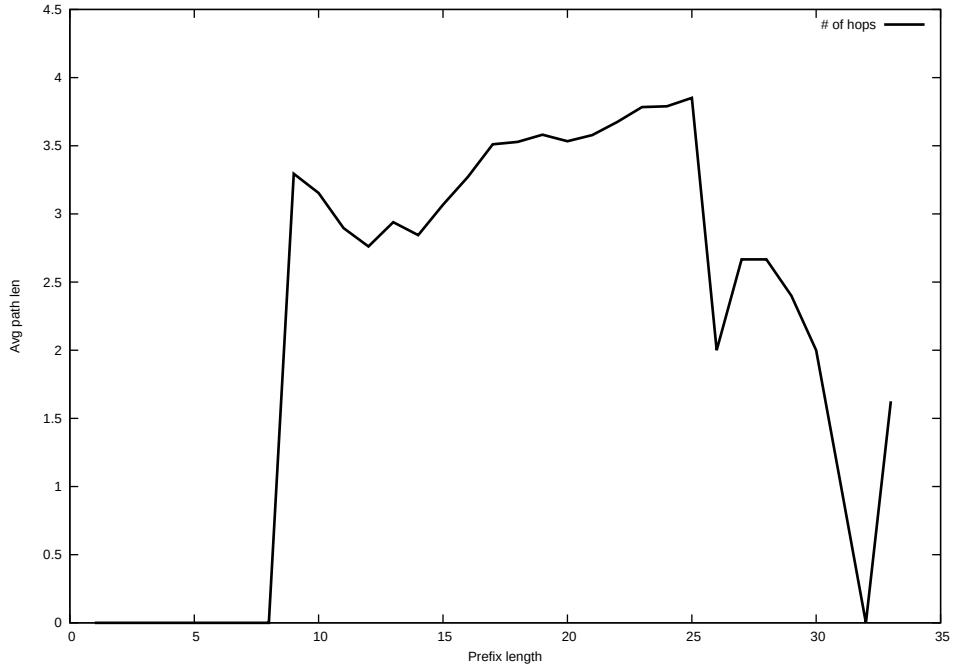
2013-02-24



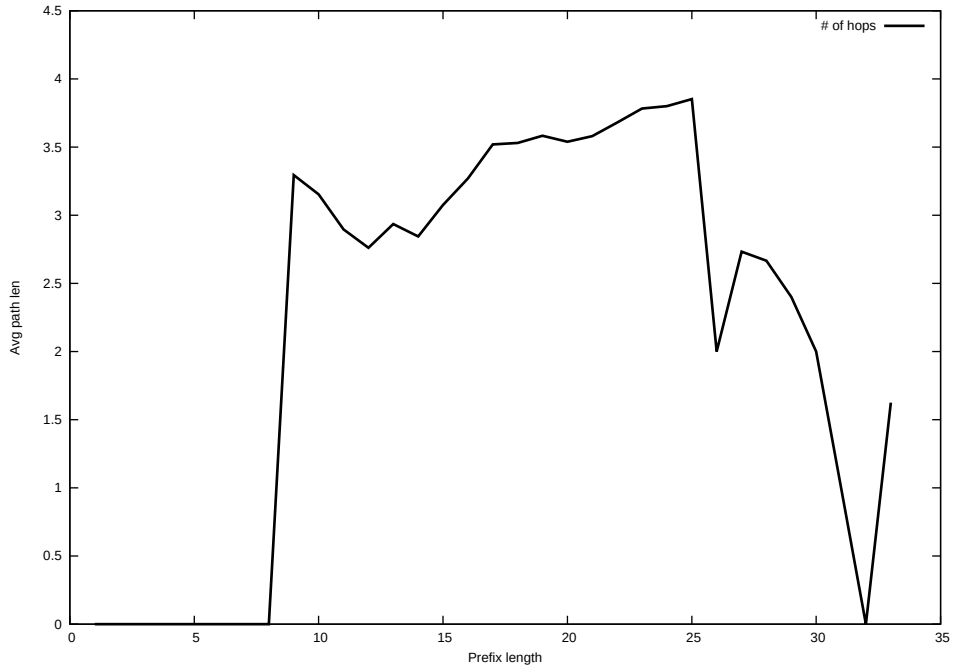
2013-02-25



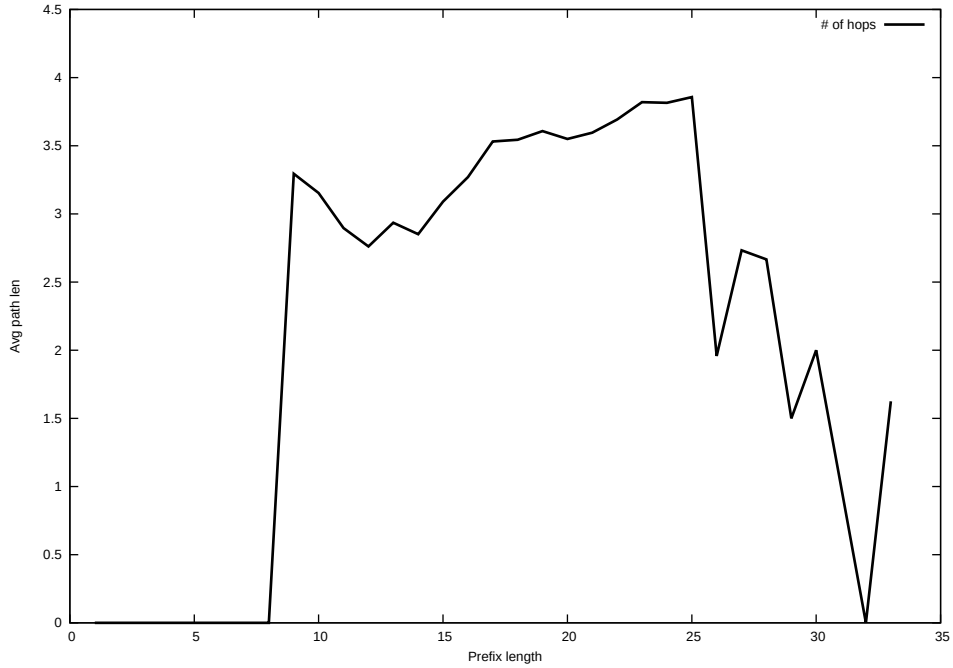
2013-02-26



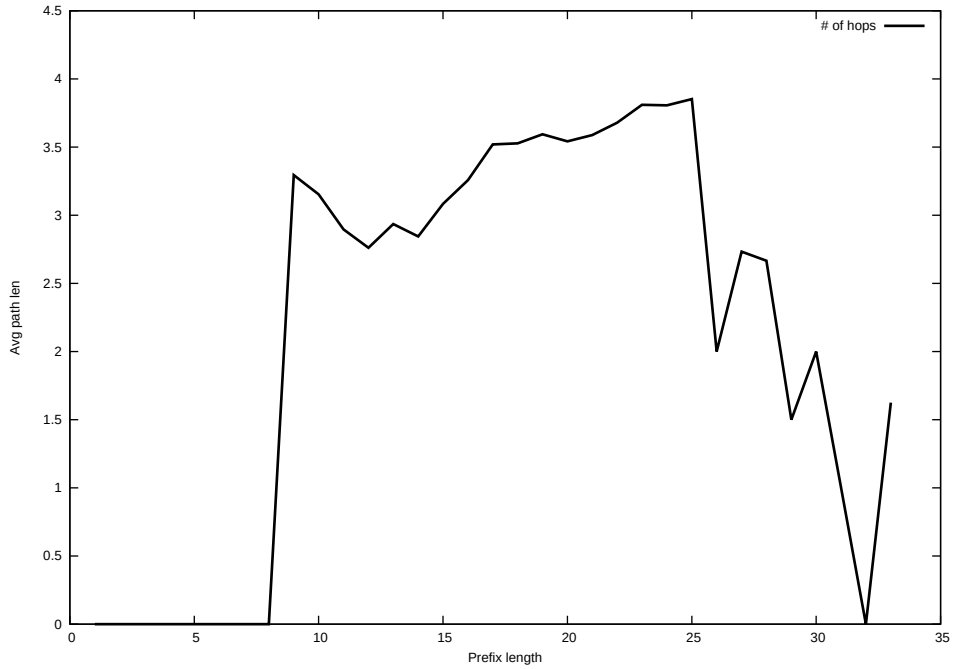
2013-02-27



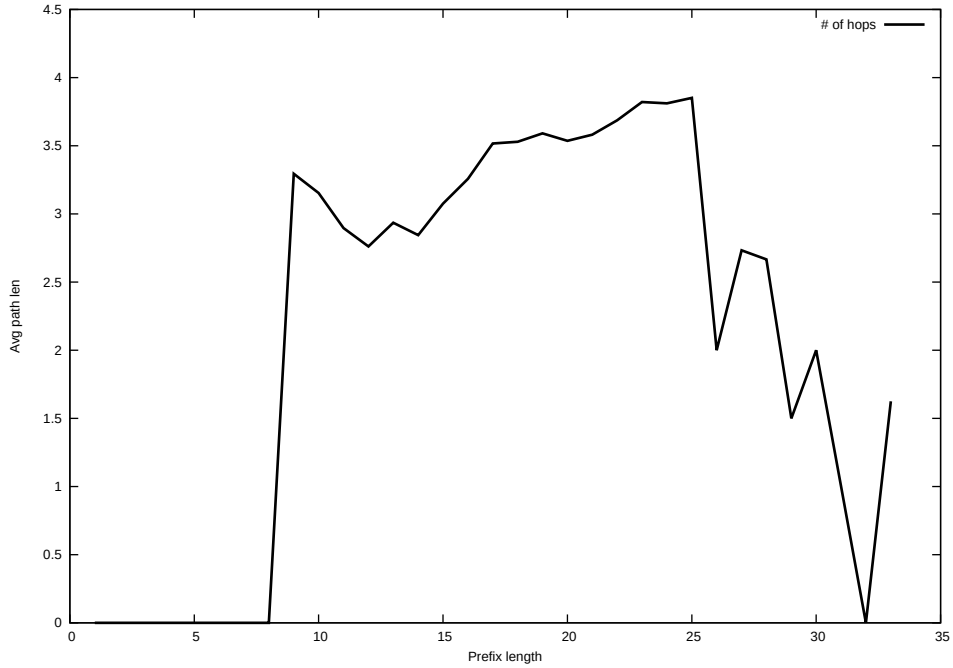
2013-02-28



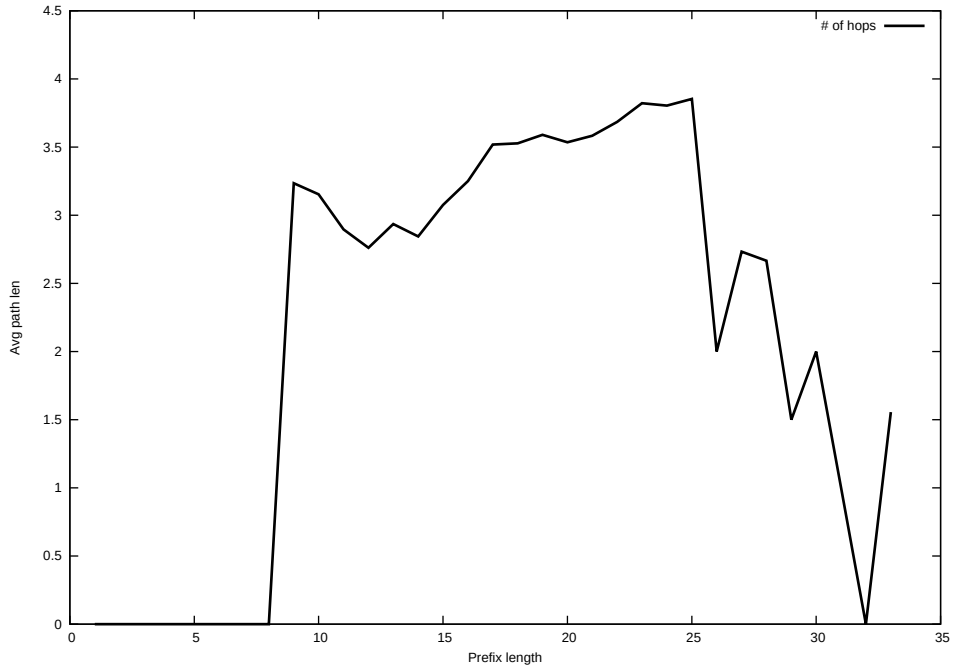
2013-03-01



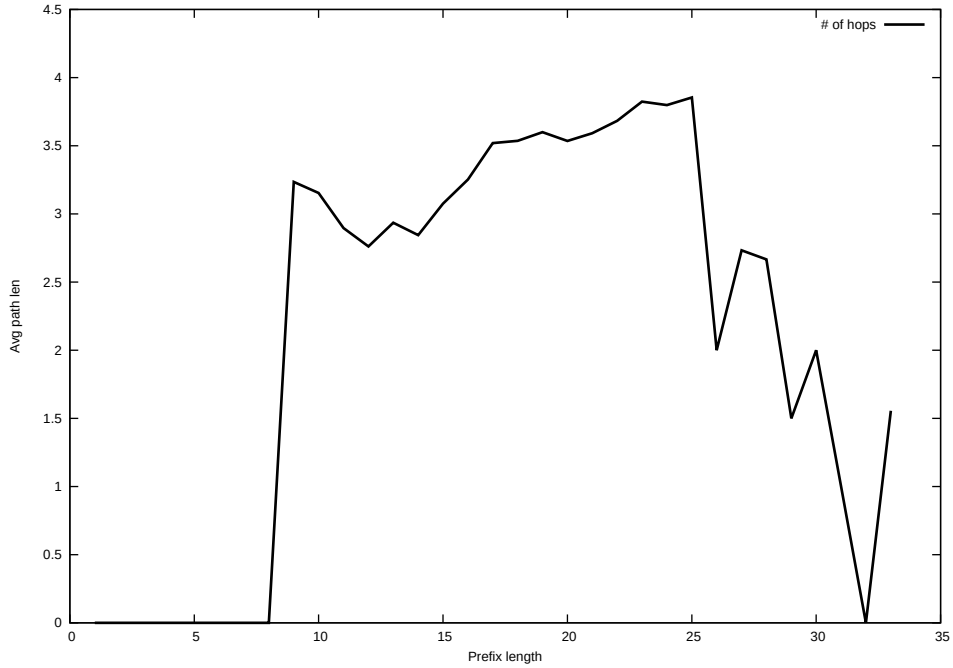
2013-03-02



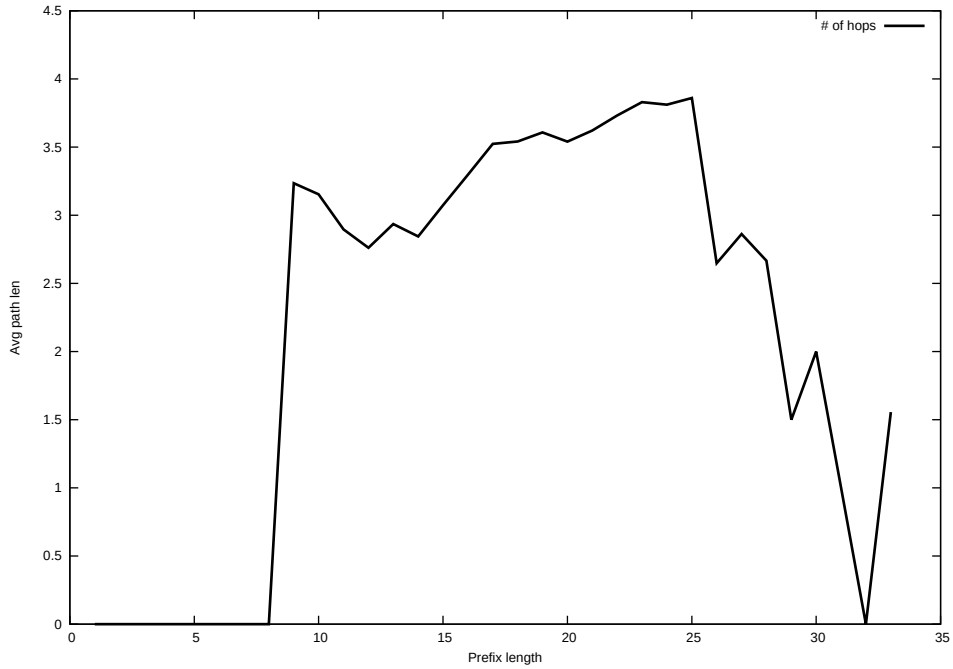
2013-03-03



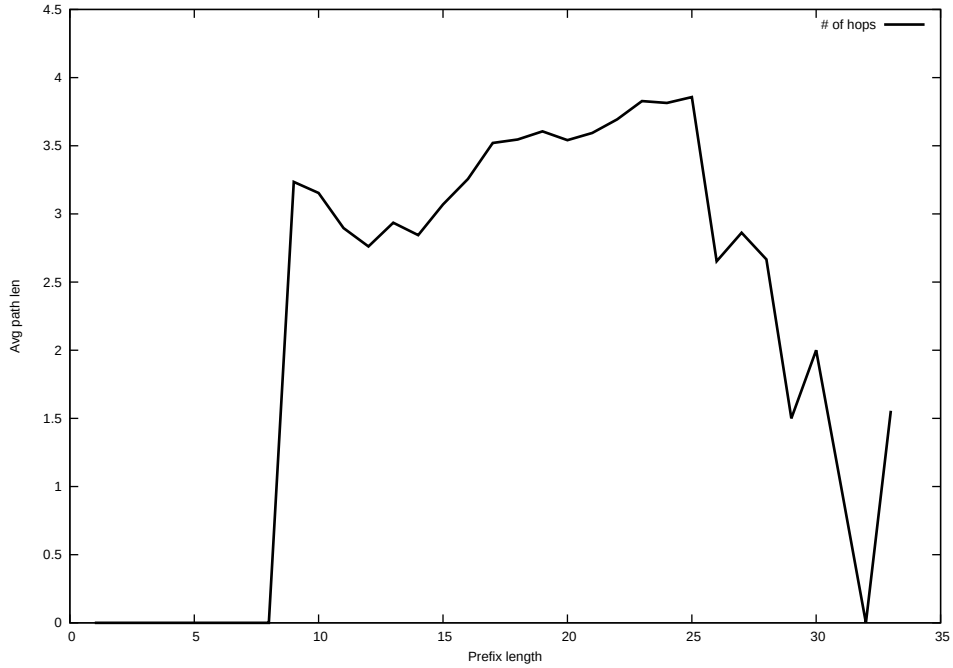
2013-03-04



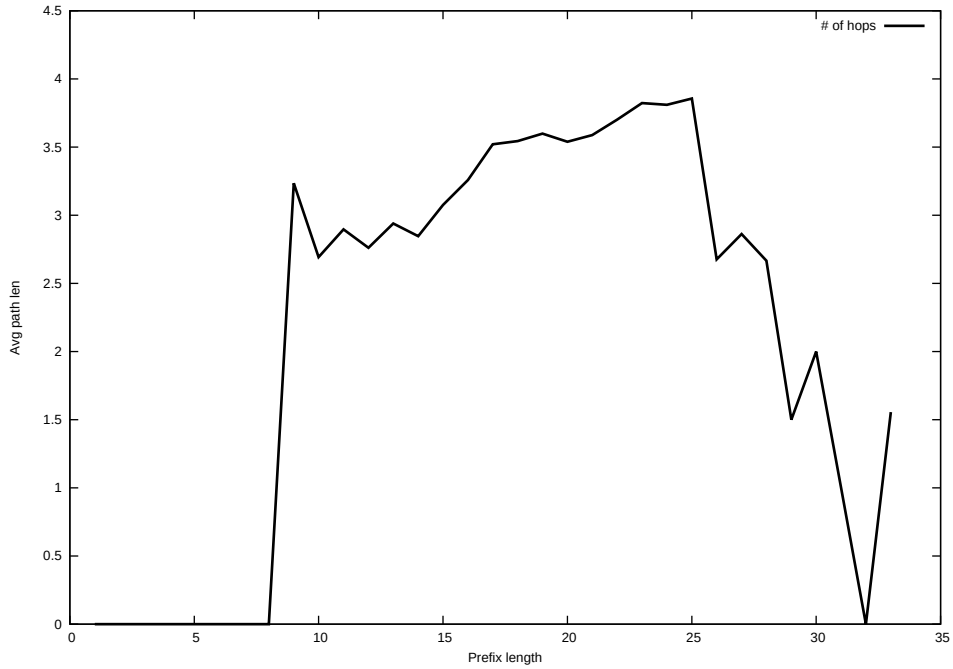
2013-03-05



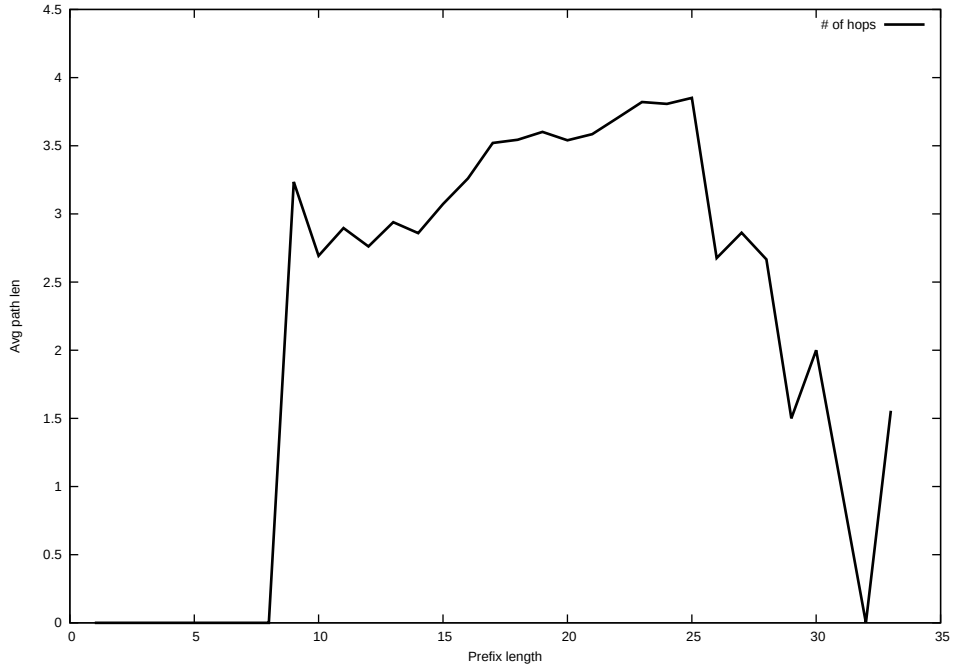
2013-03-06



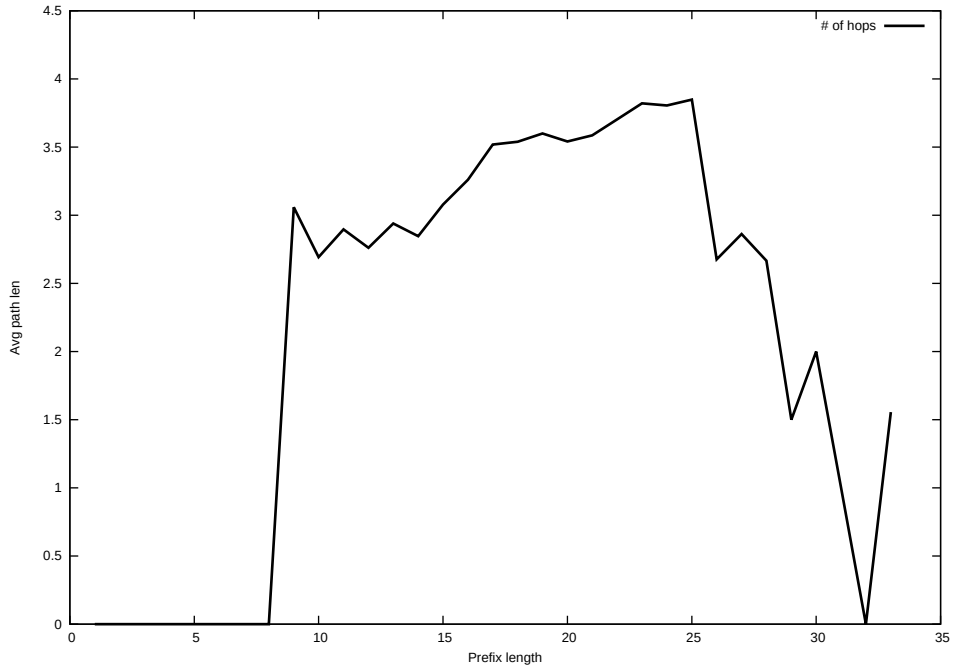
2013-03-07



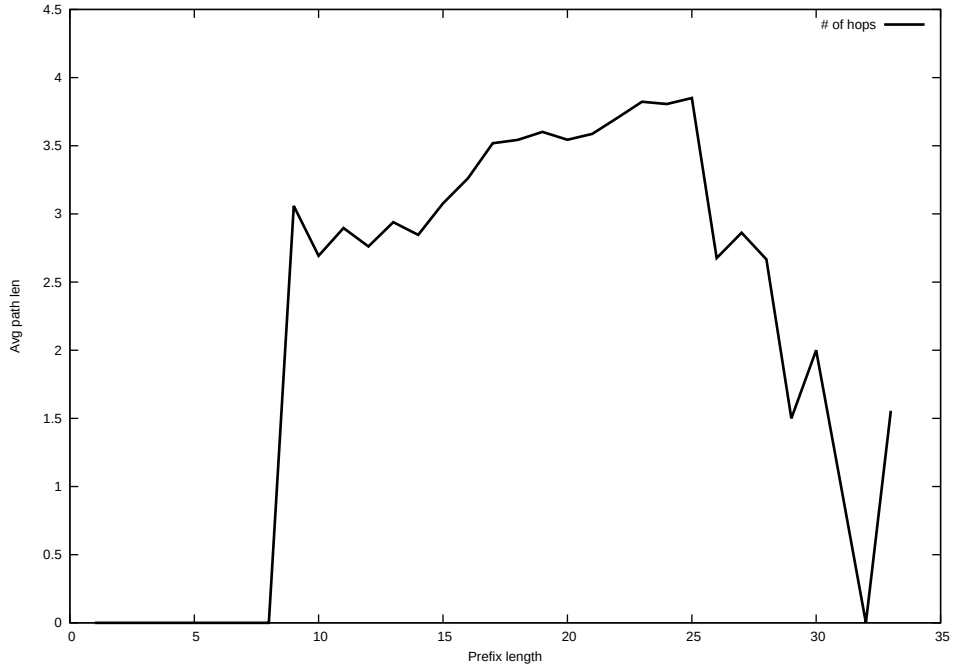
2013-03-08



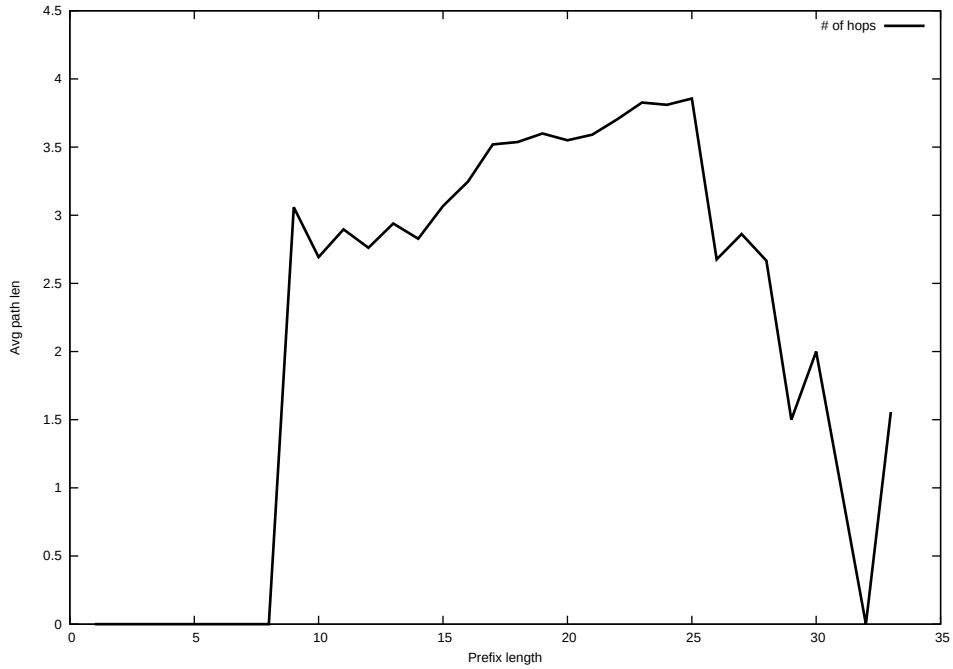
2013-03-09



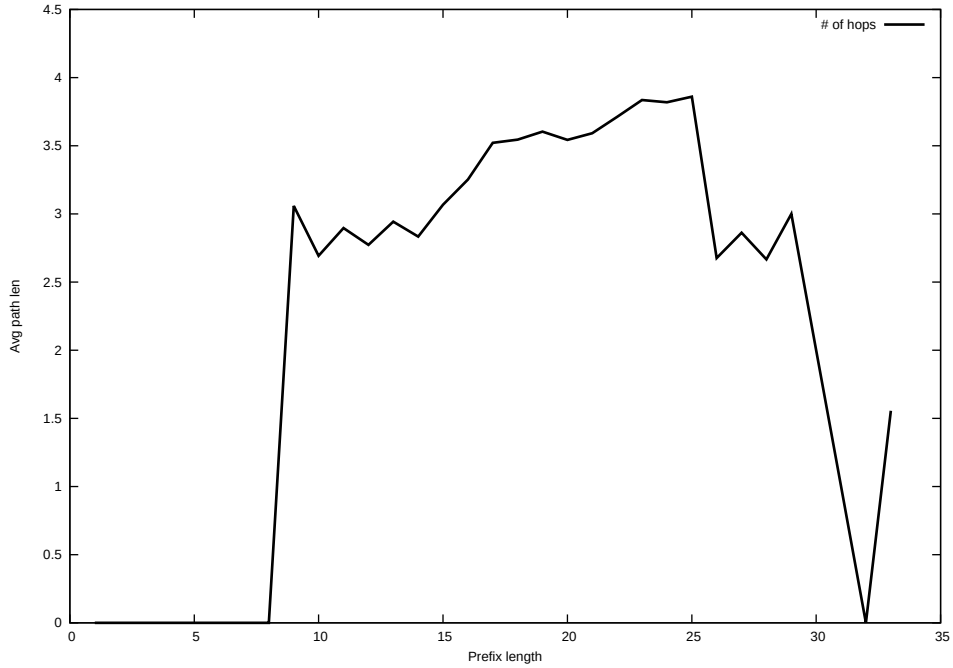
2013-03-10



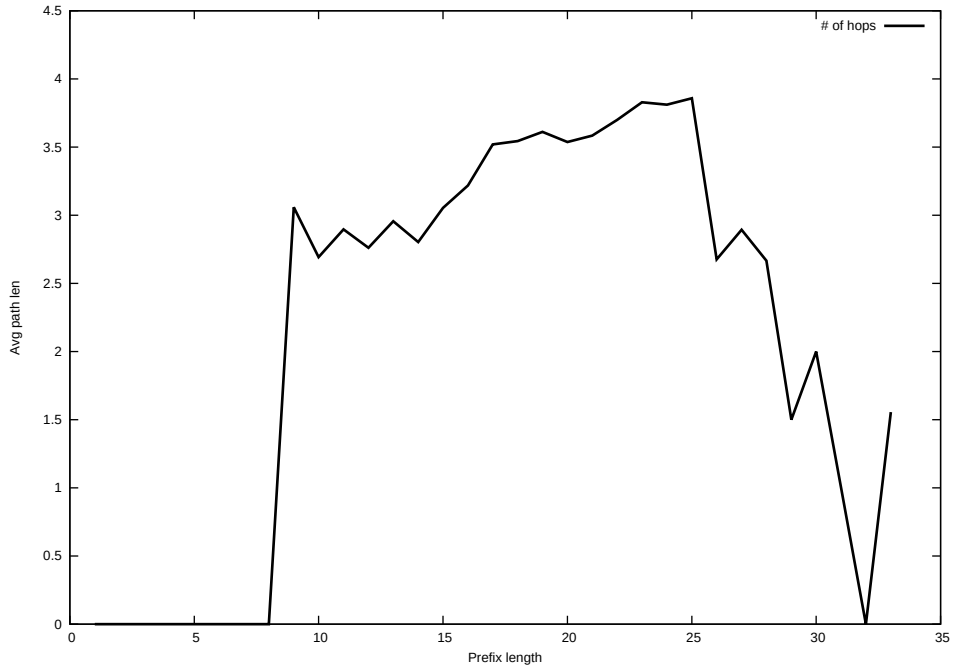
2013-03-11



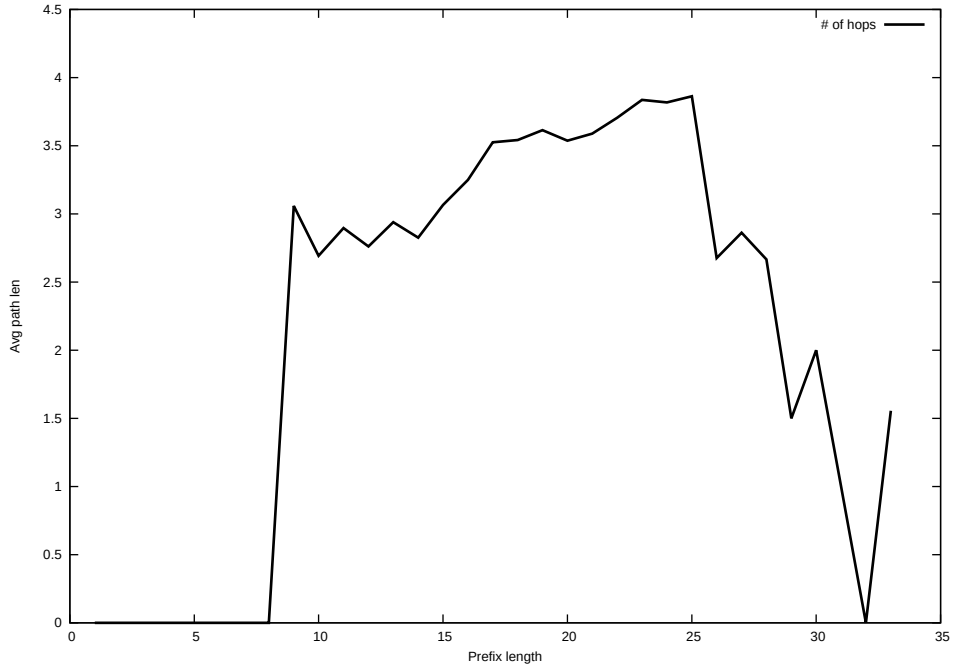
2013-03-12



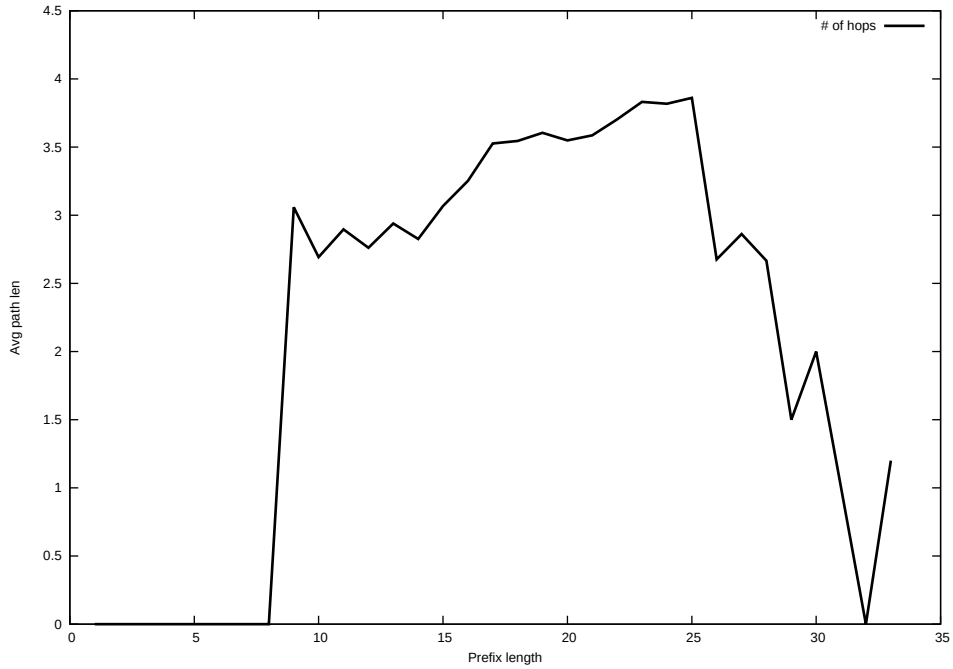
2013-03-13



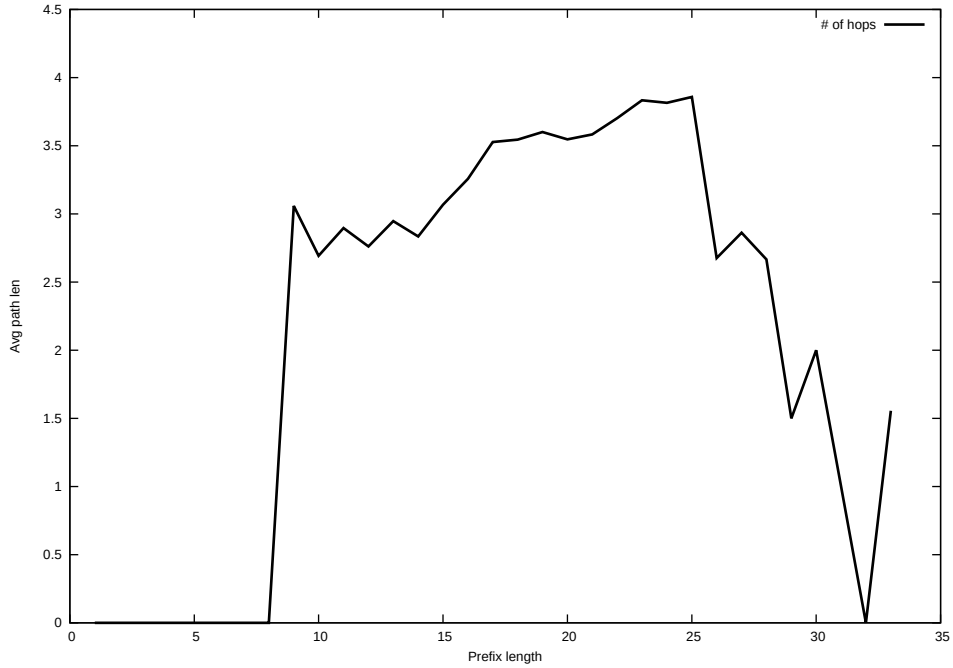
2013-03-14



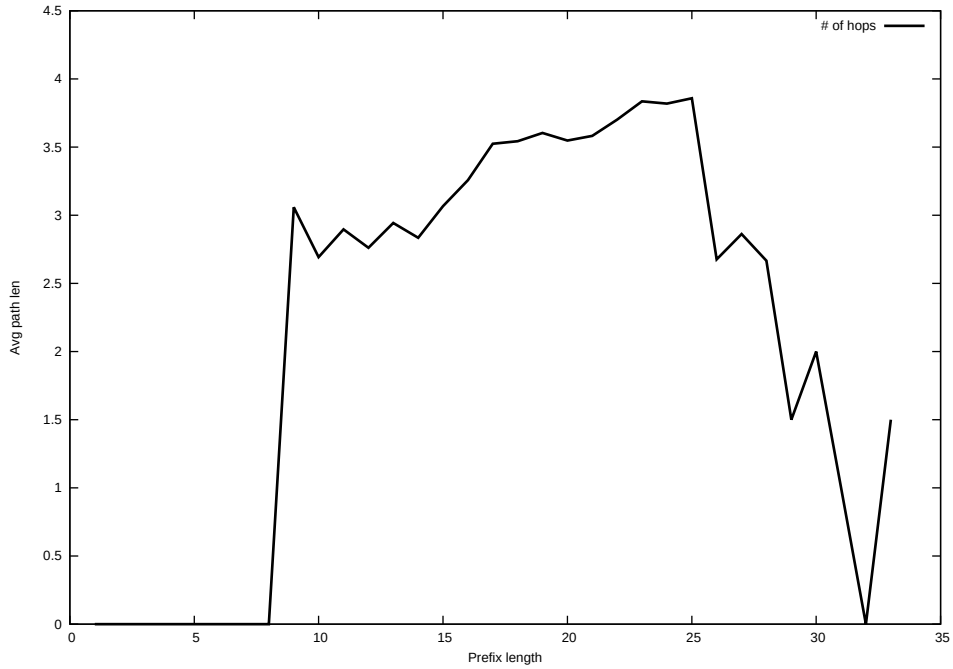
2013-03-15



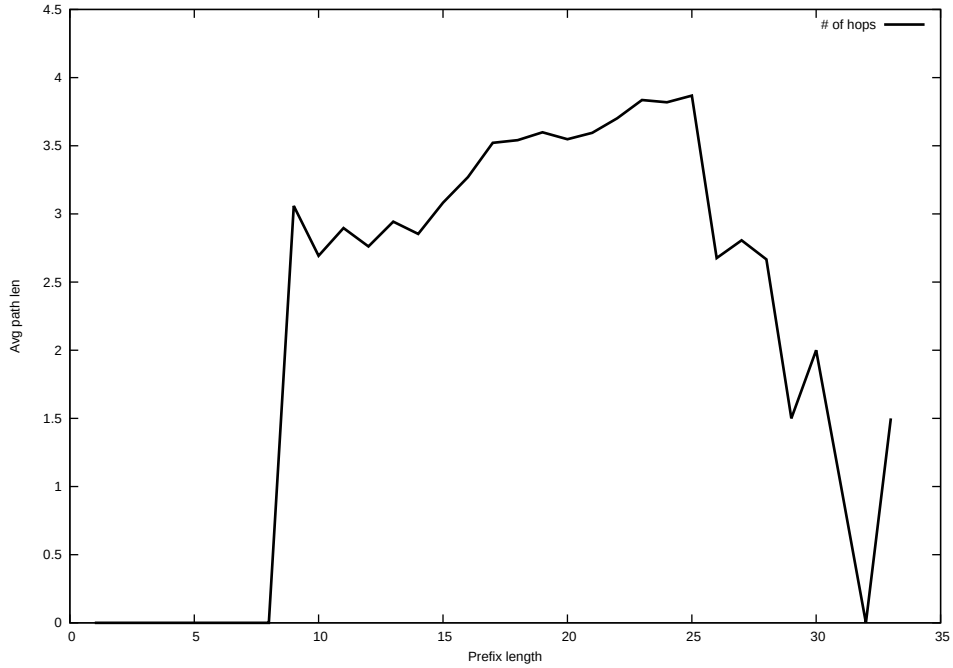
2013-03-16



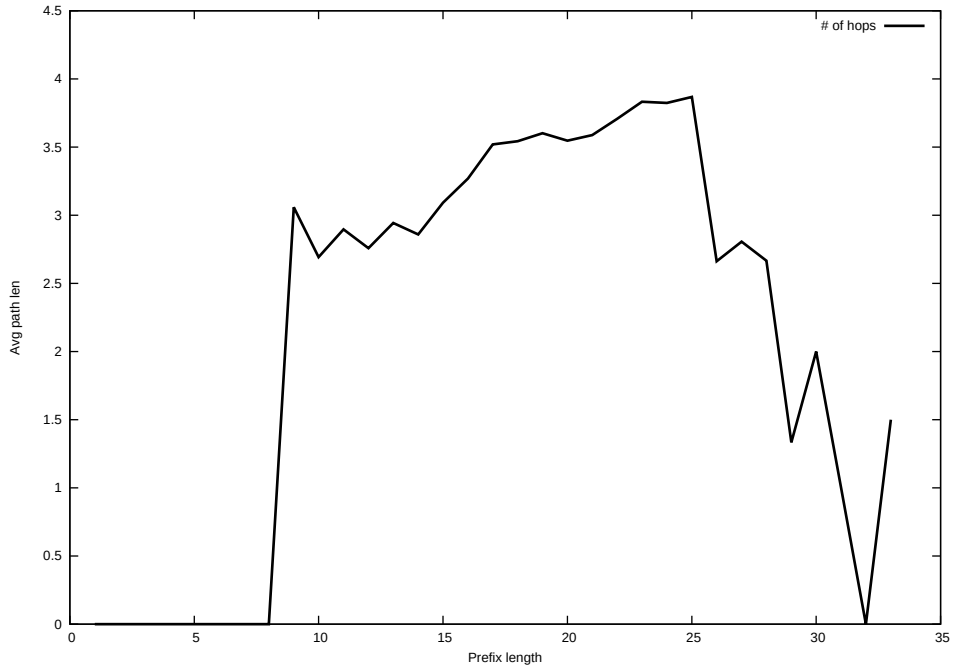
2013-03-17



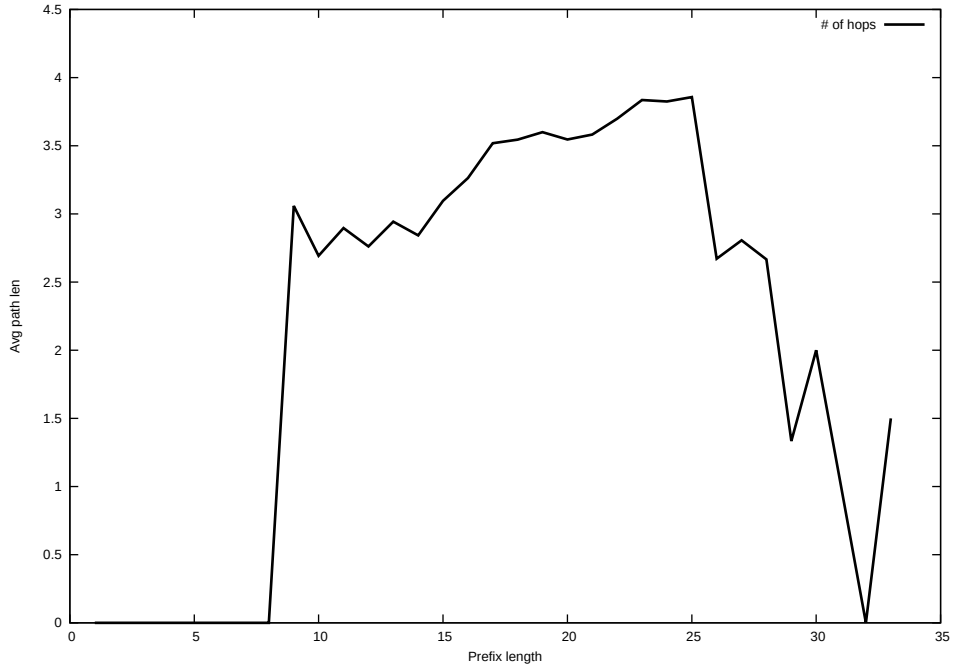
2013-03-18



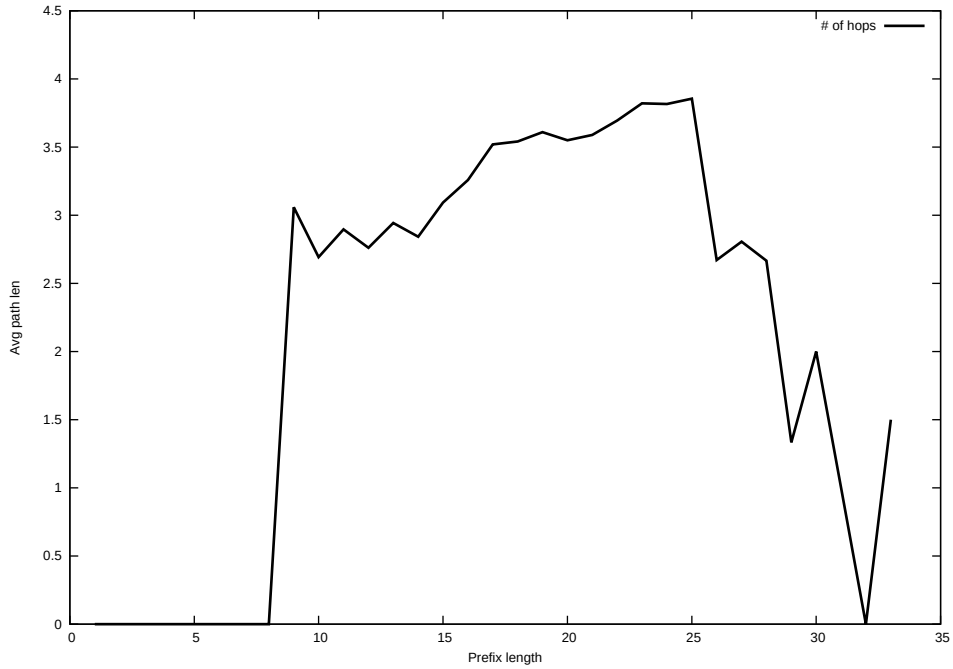
2013-03-19



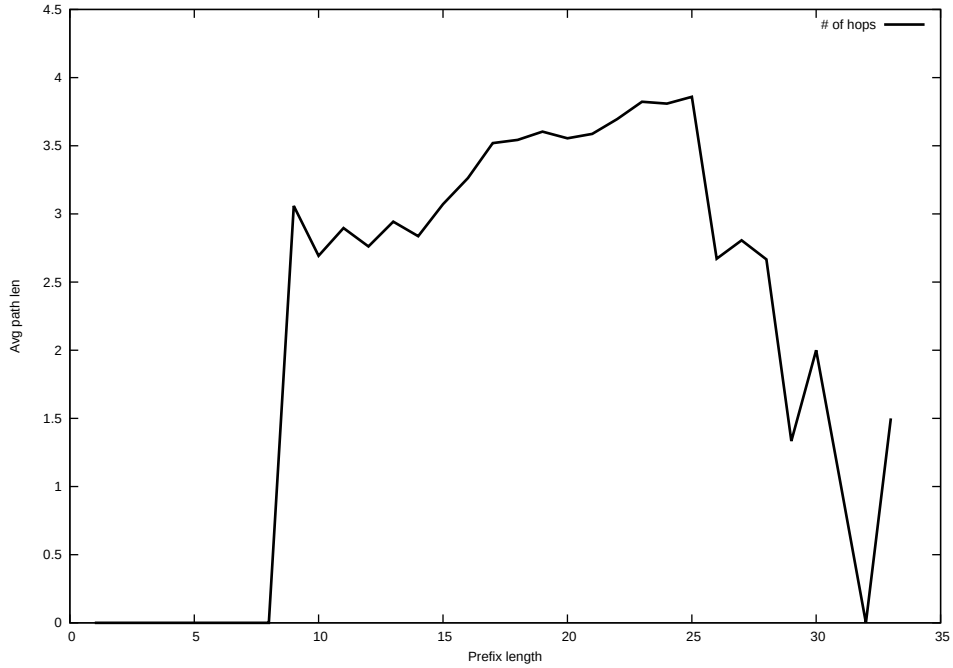
2013-03-20



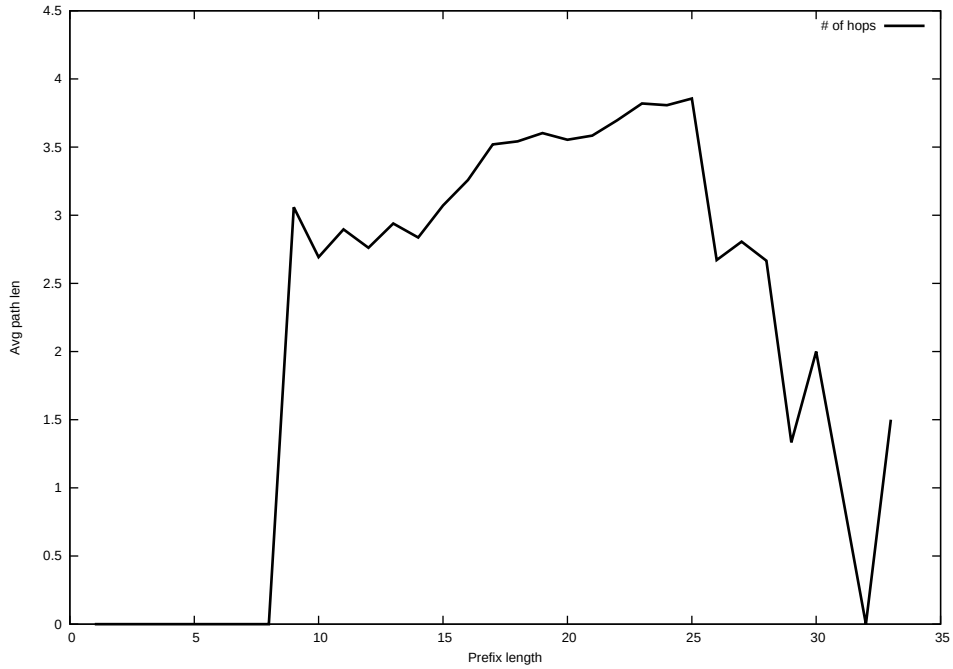
2013-03-21



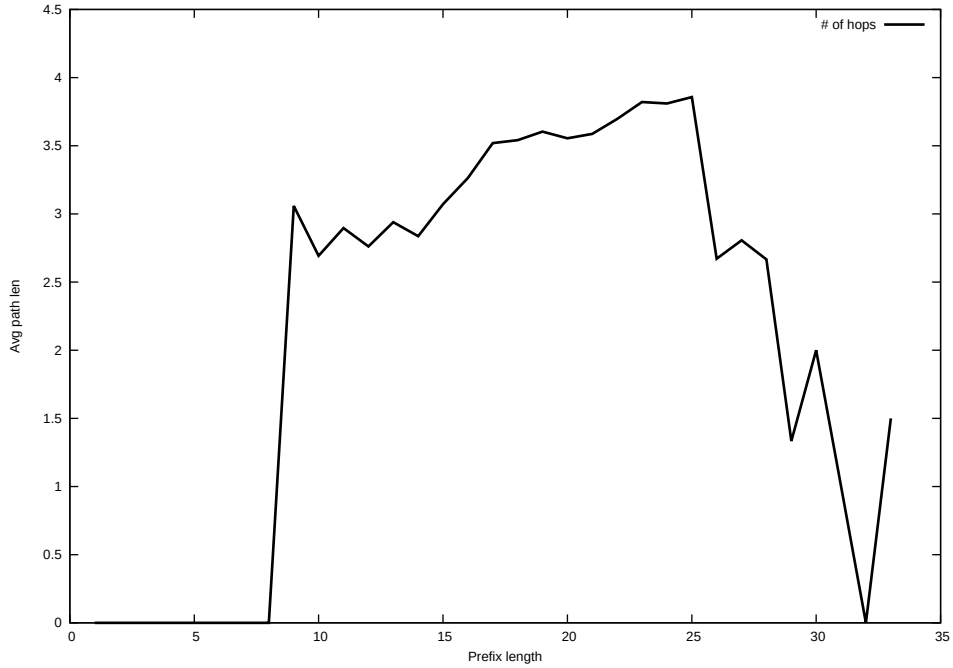
2013-03-22



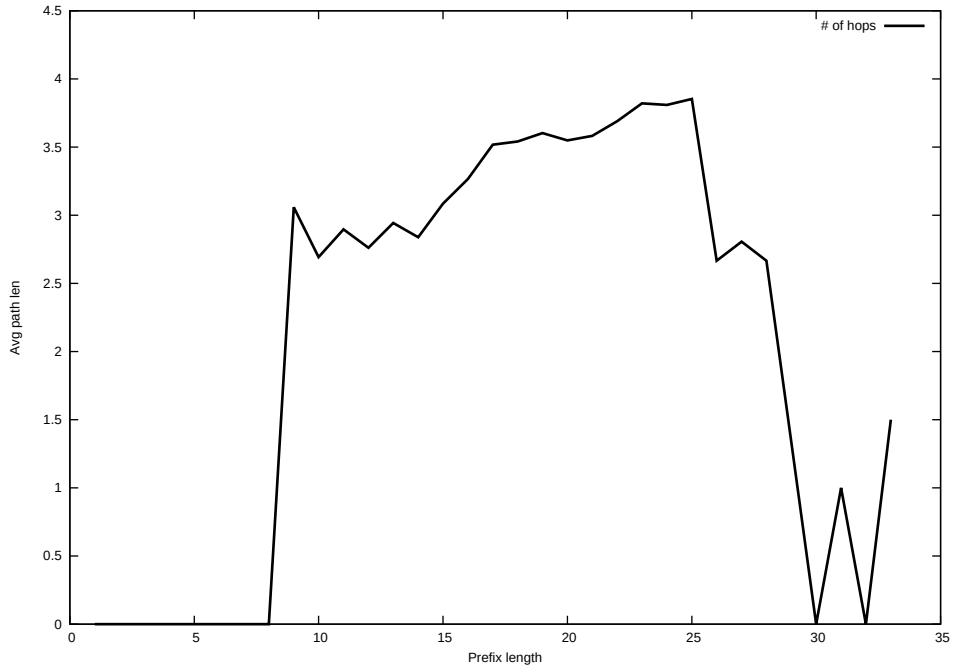
2013-03-23



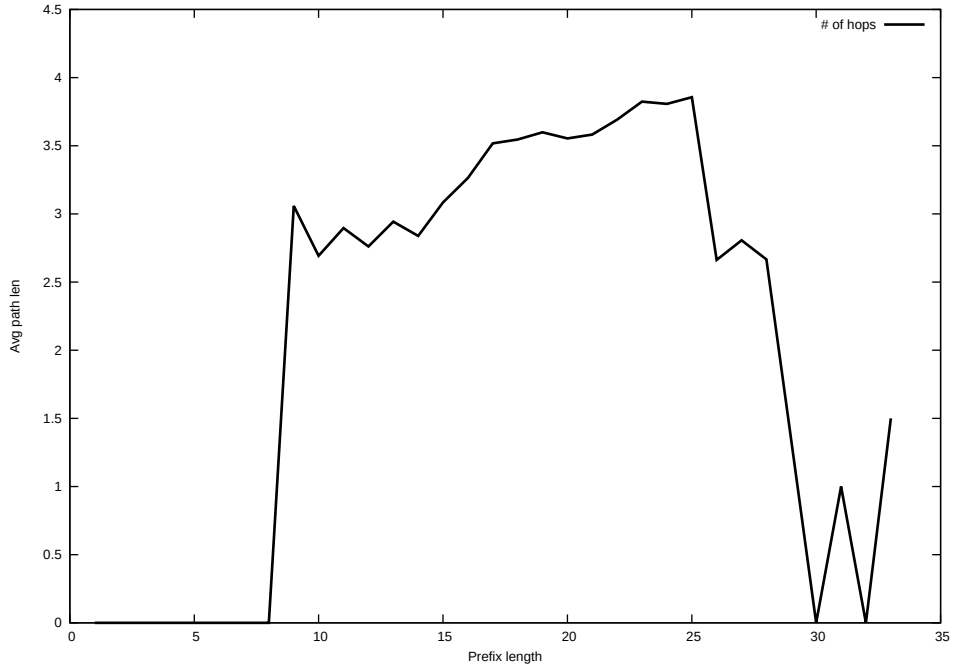
2013-03-24



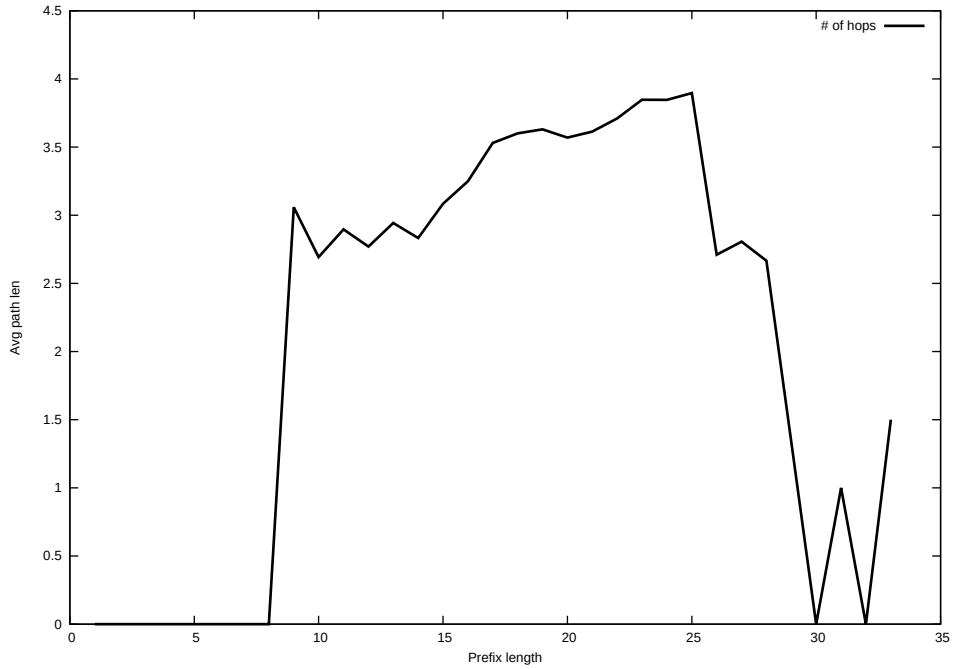
2013-03-25



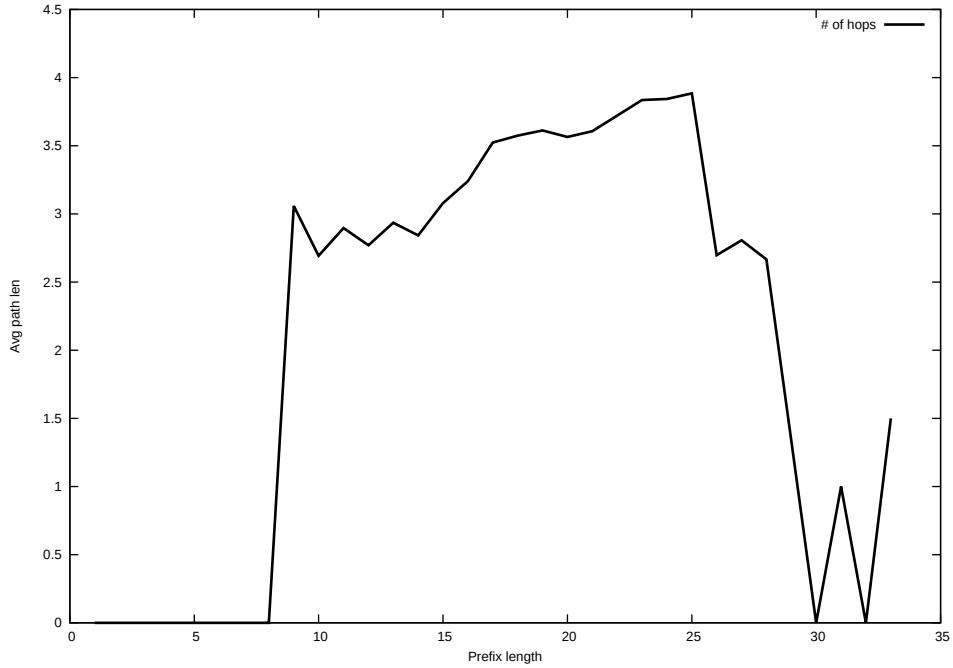
2013-03-26



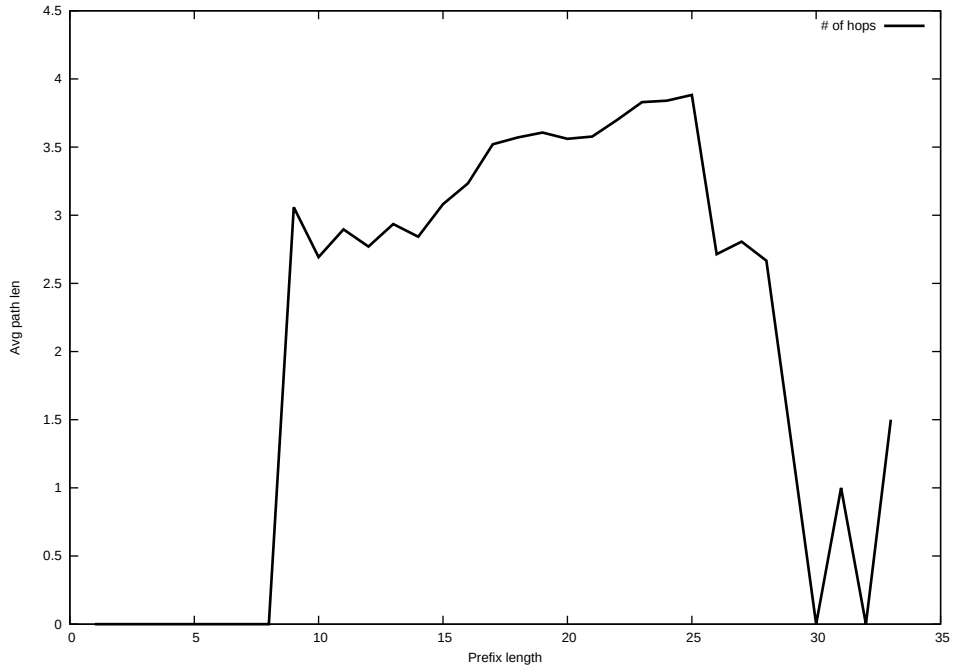
2013-03-27



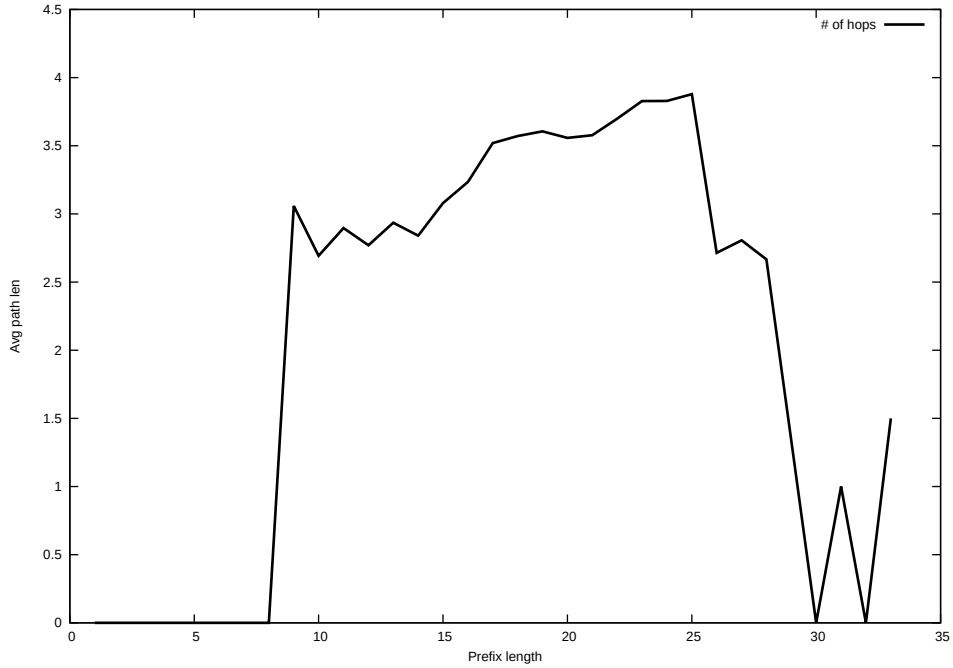
2013-03-28



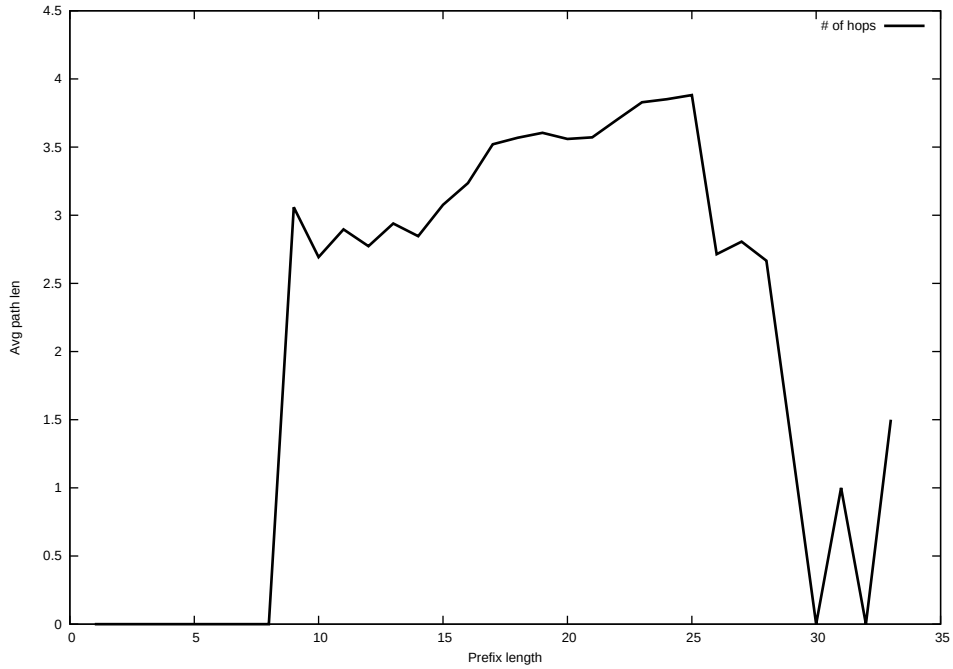
2013-03-29



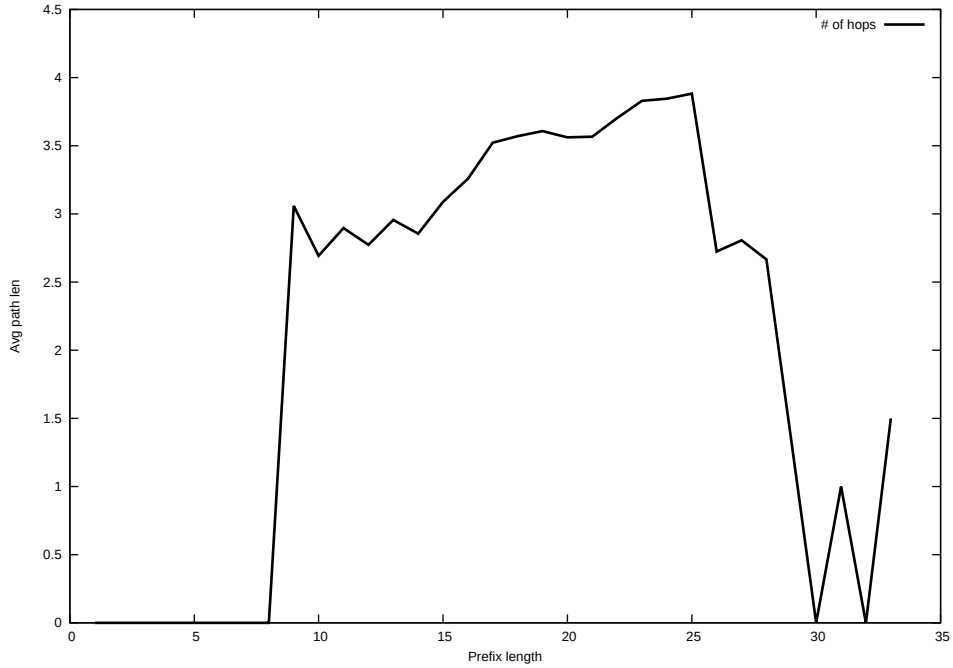
2013-03-30



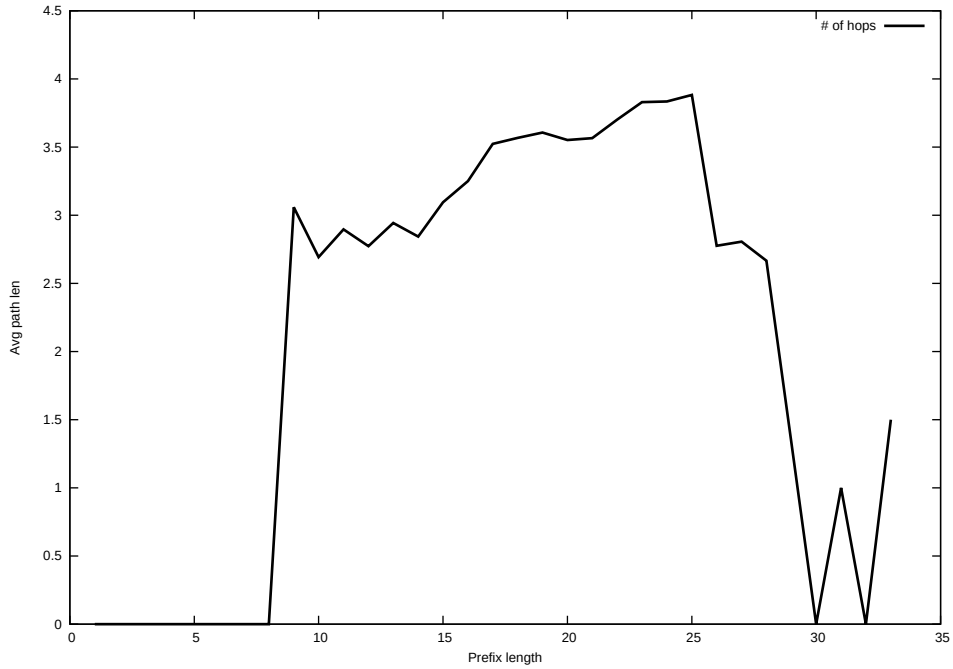
2013-03-31



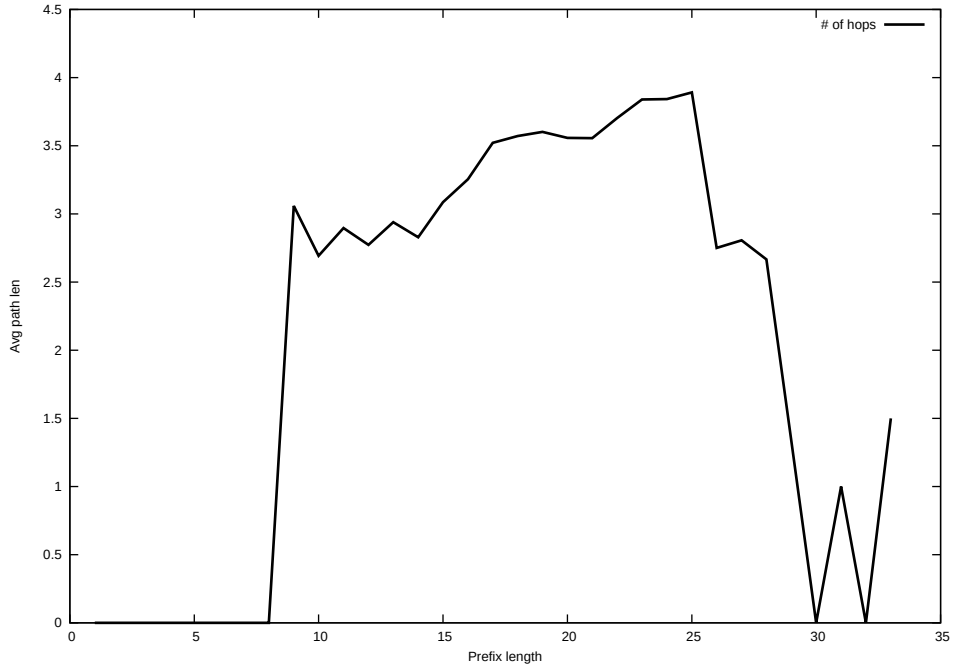
2013-04-01



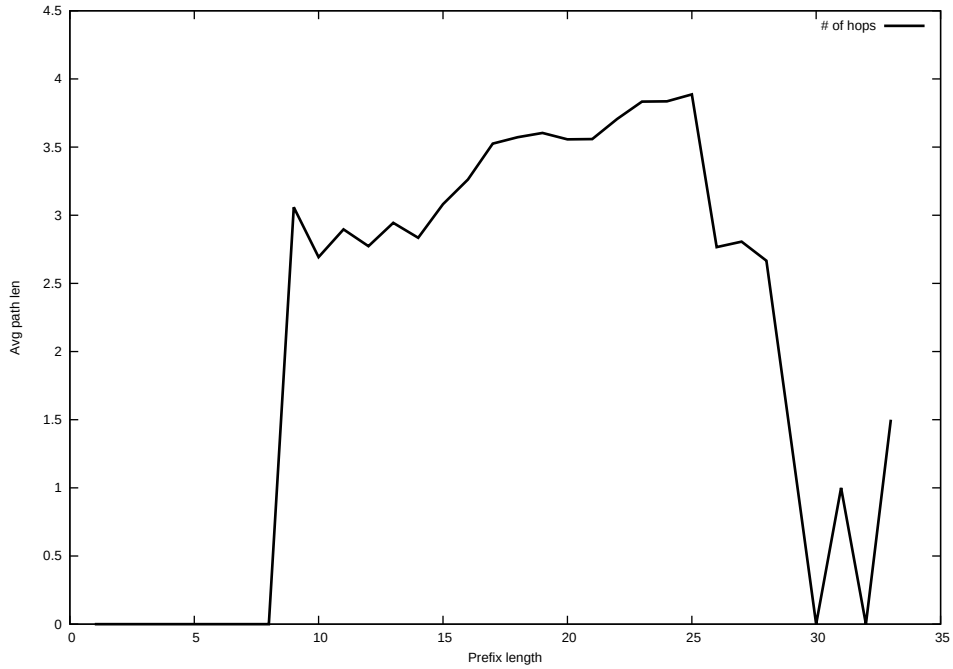
2013-04-02



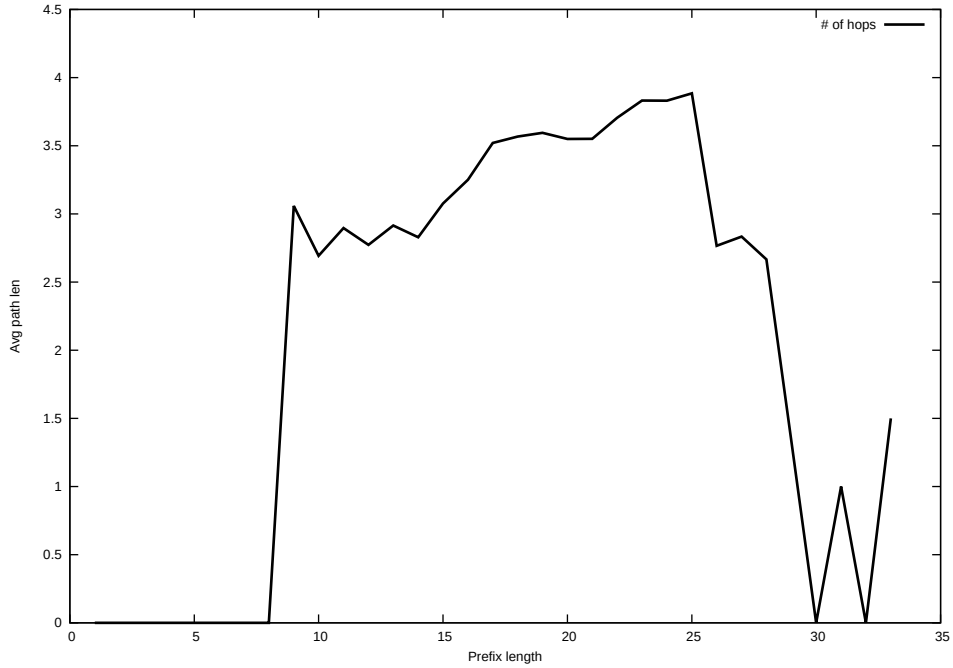
2013-04-03



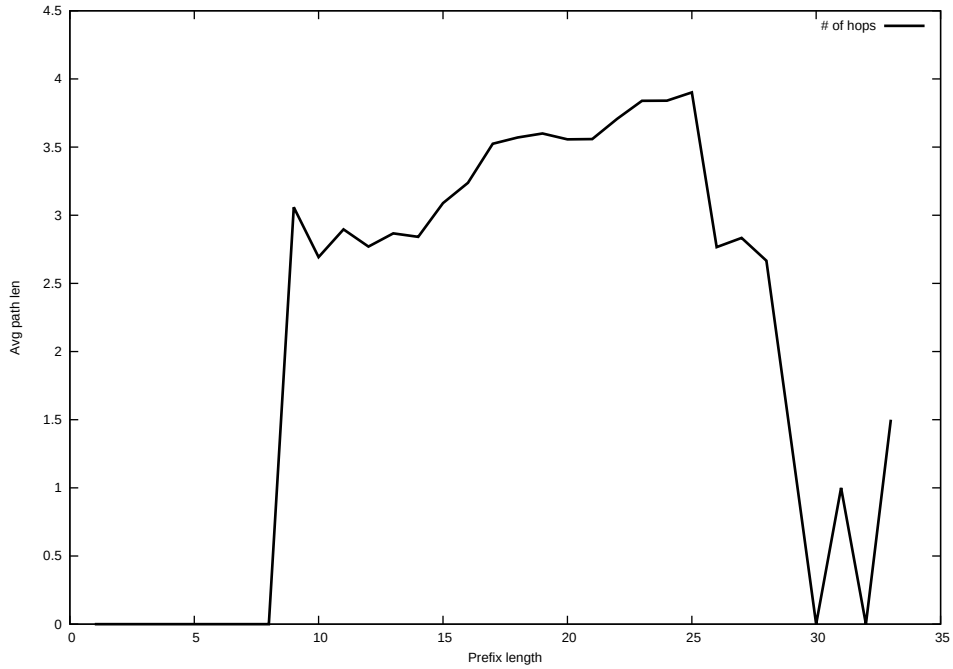
2013-04-04



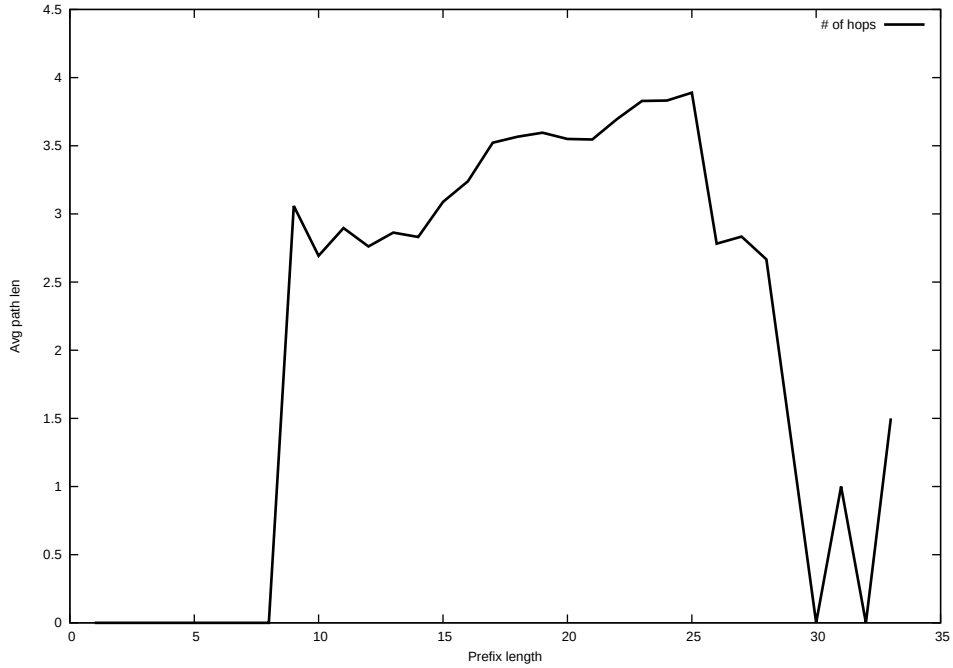
2013-04-05



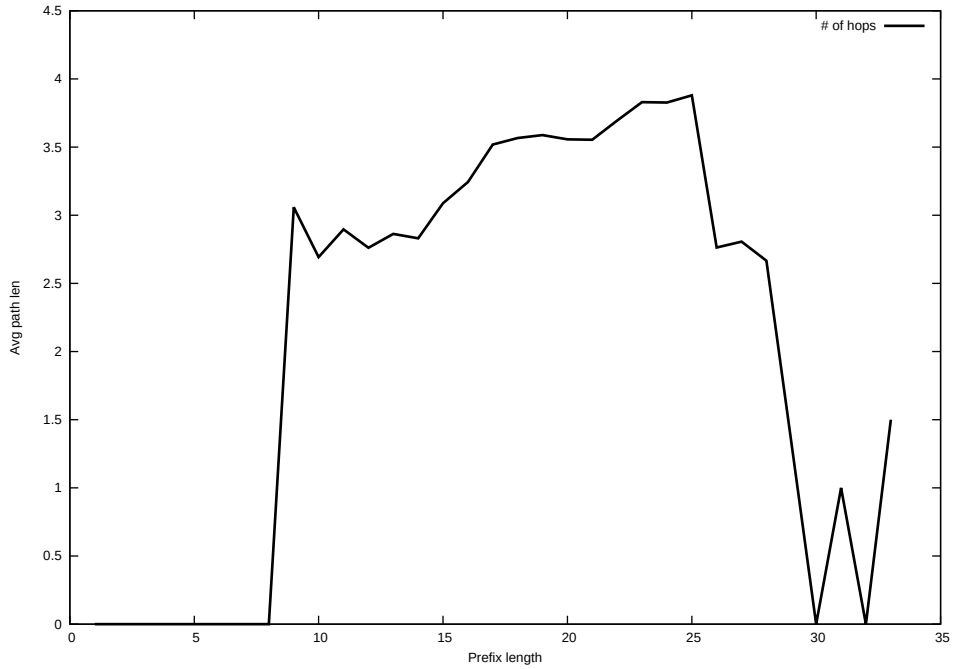
2013-04-06



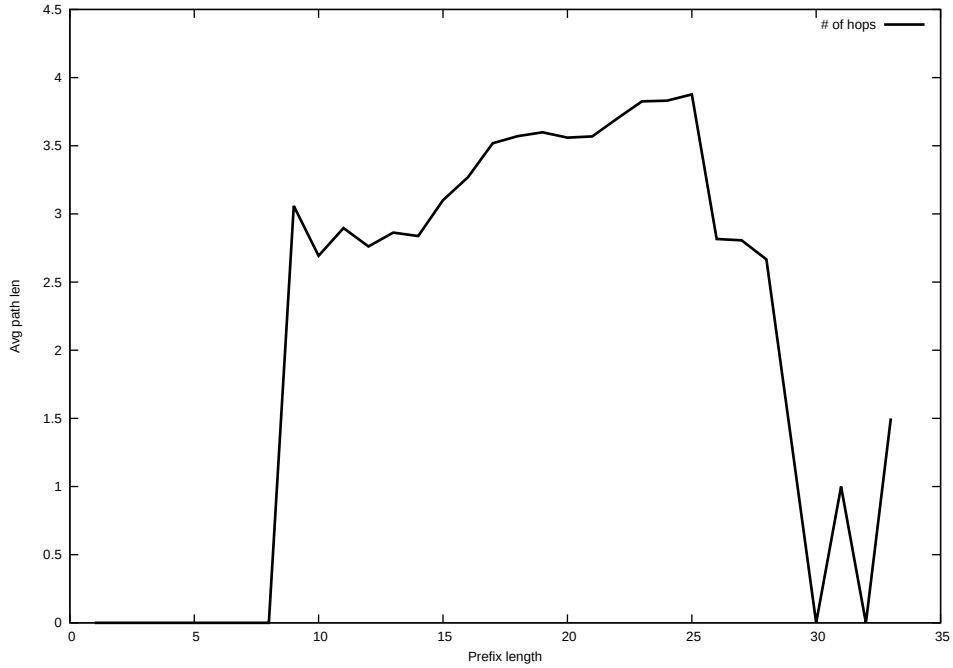
2013-04-07



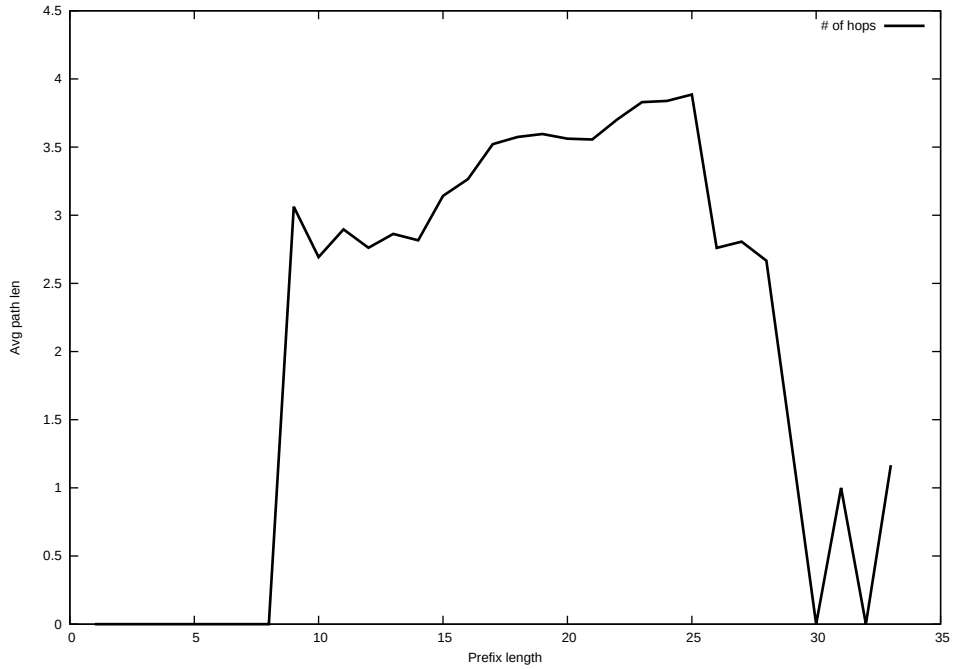
2013-04-08



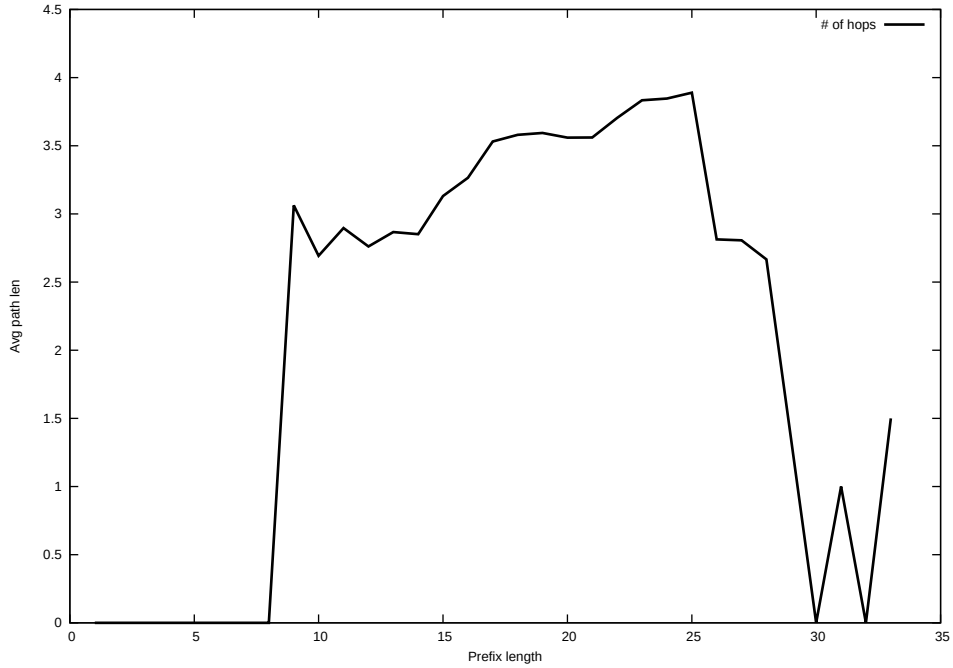
2013-04-09



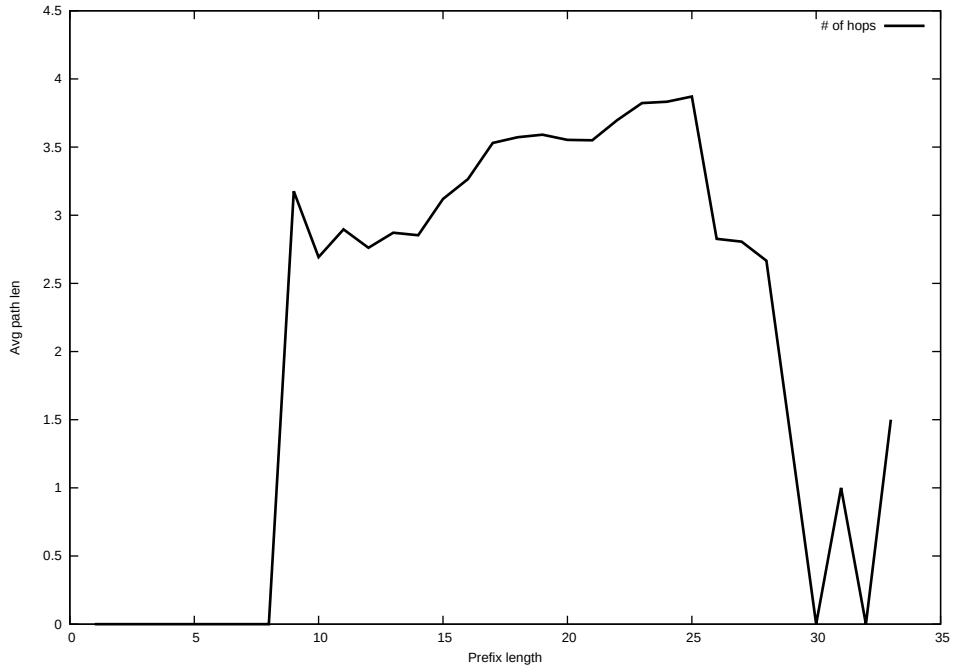
2013-04-10



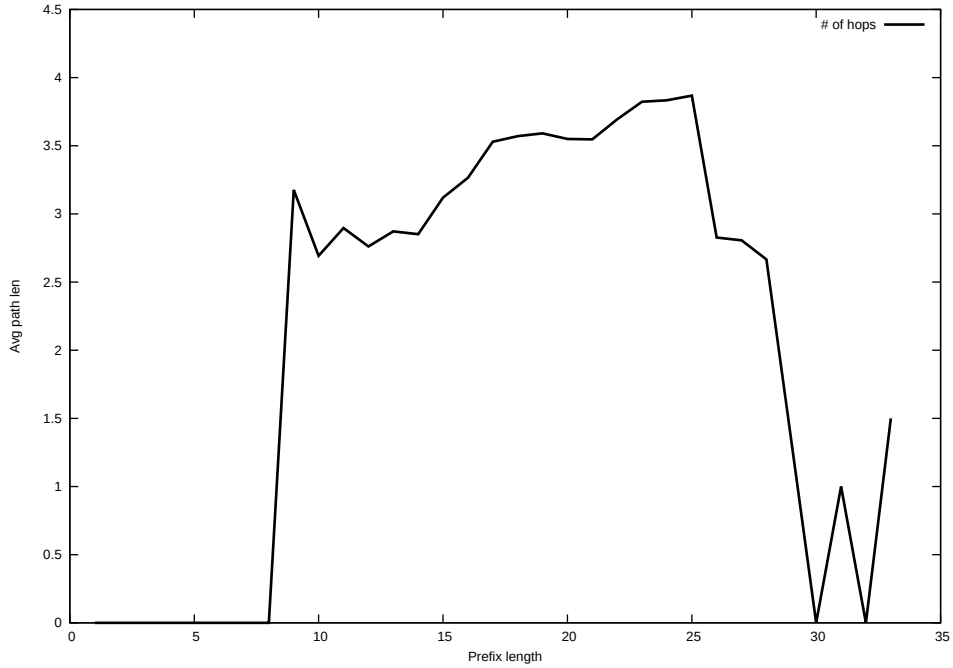
2013-04-11



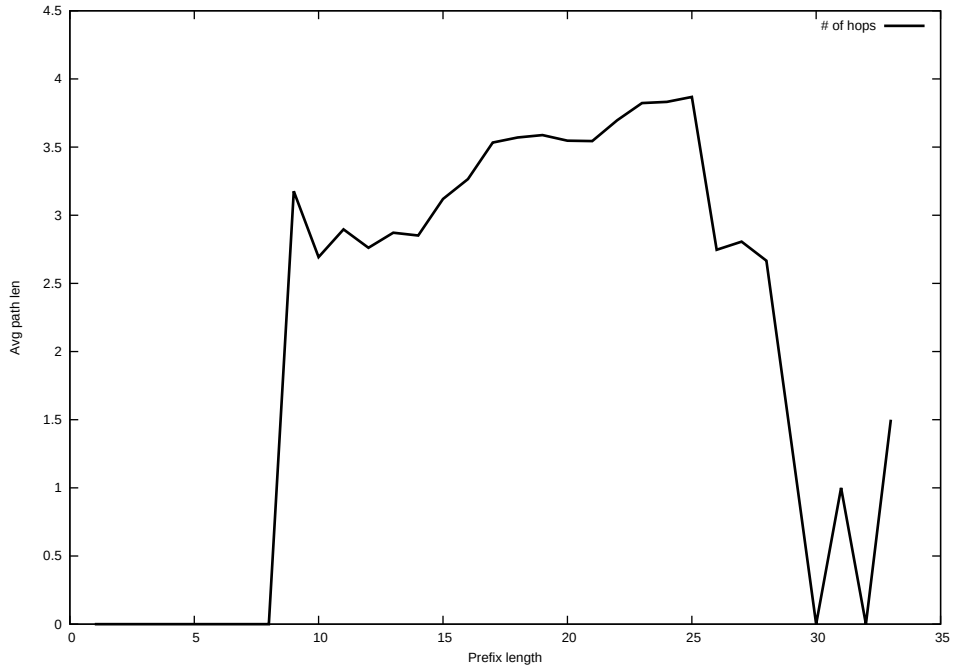
2013-04-12



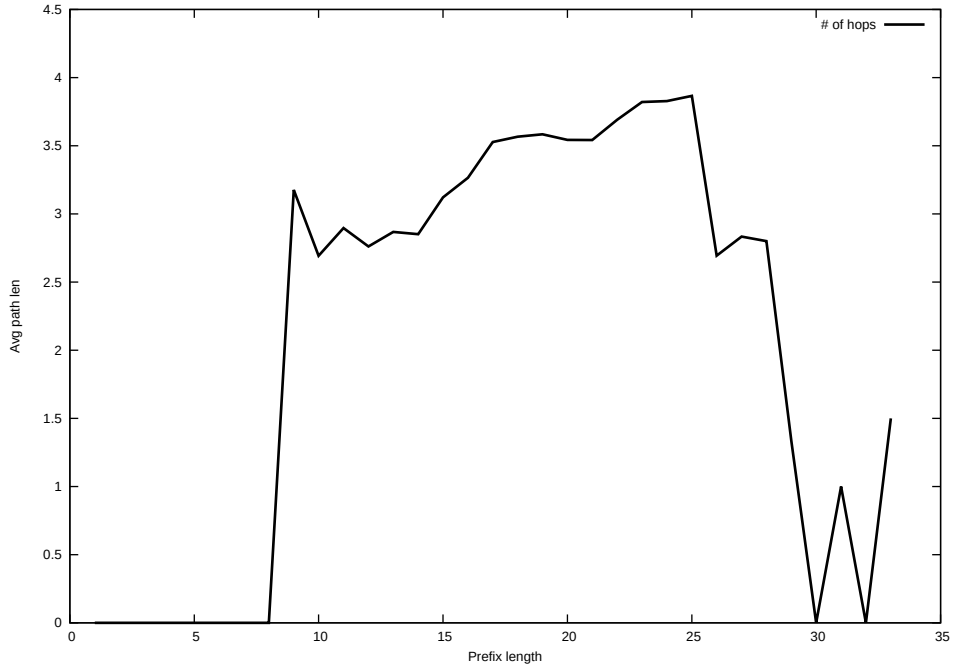
2013-04-13



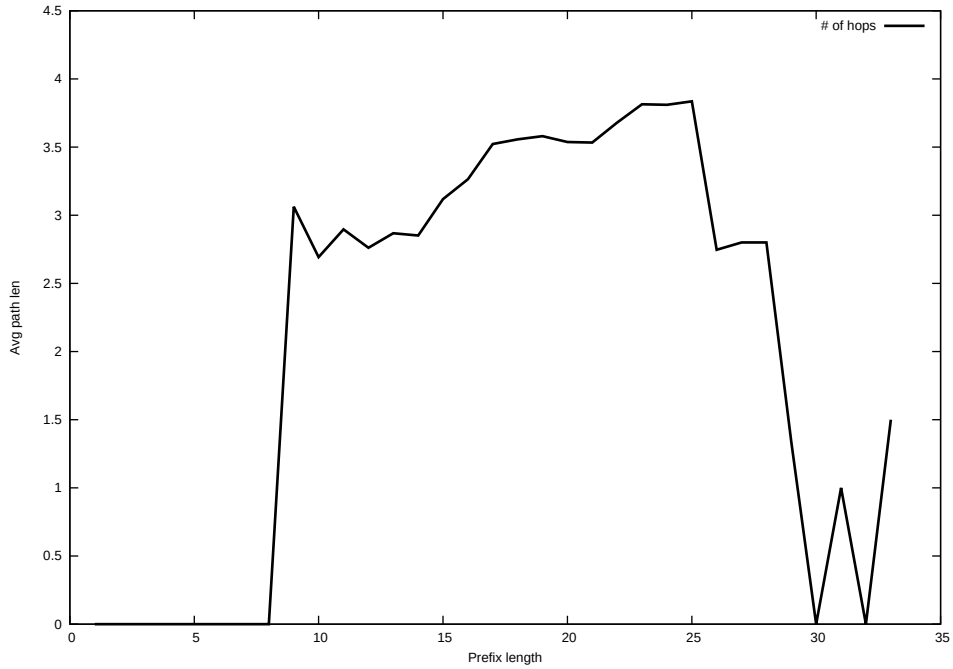
2013-04-14



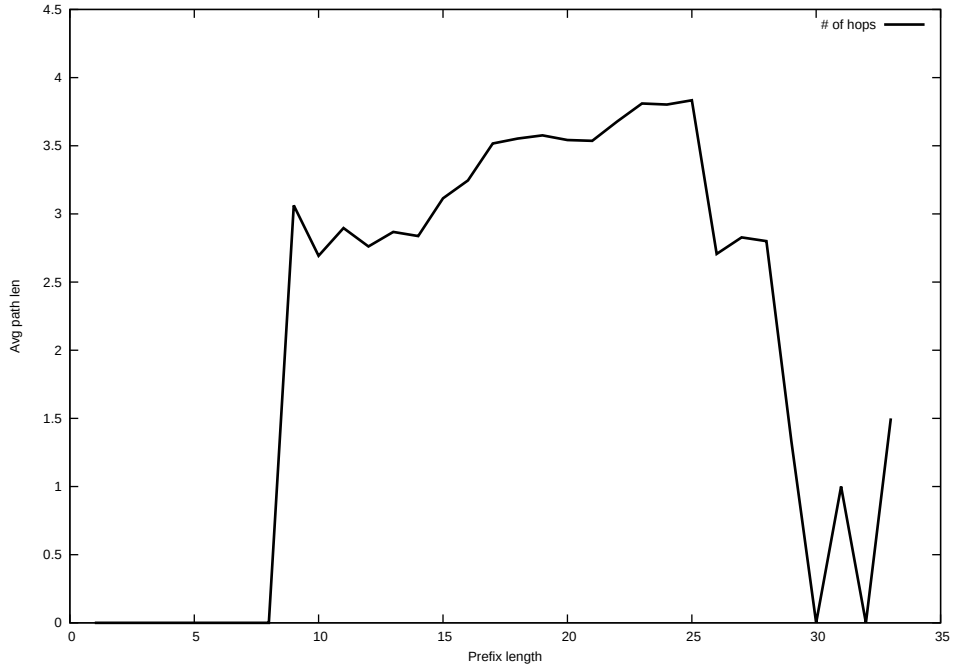
2013-04-15



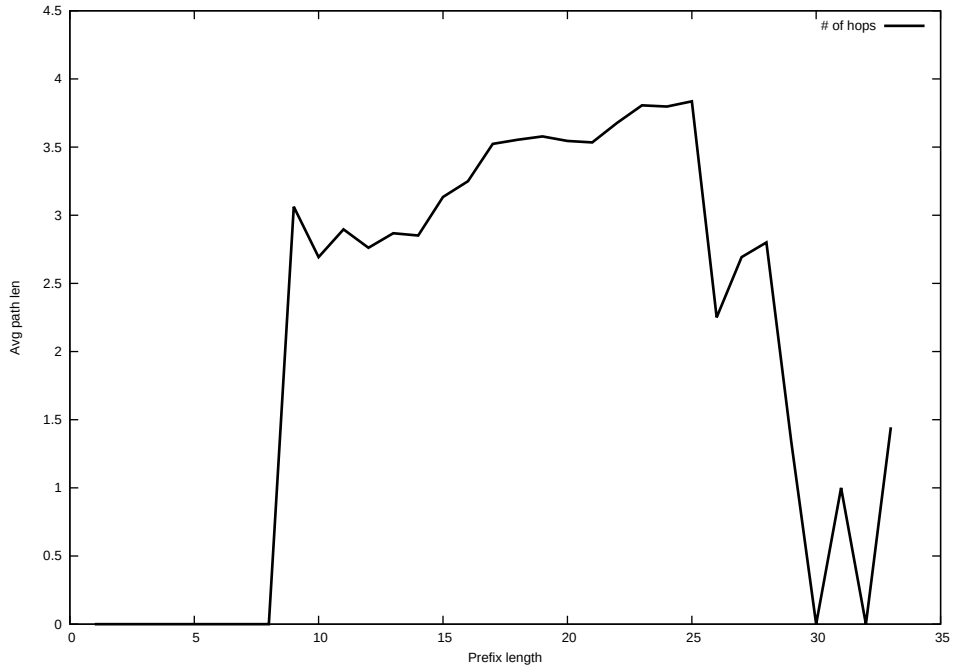
2013-04-16



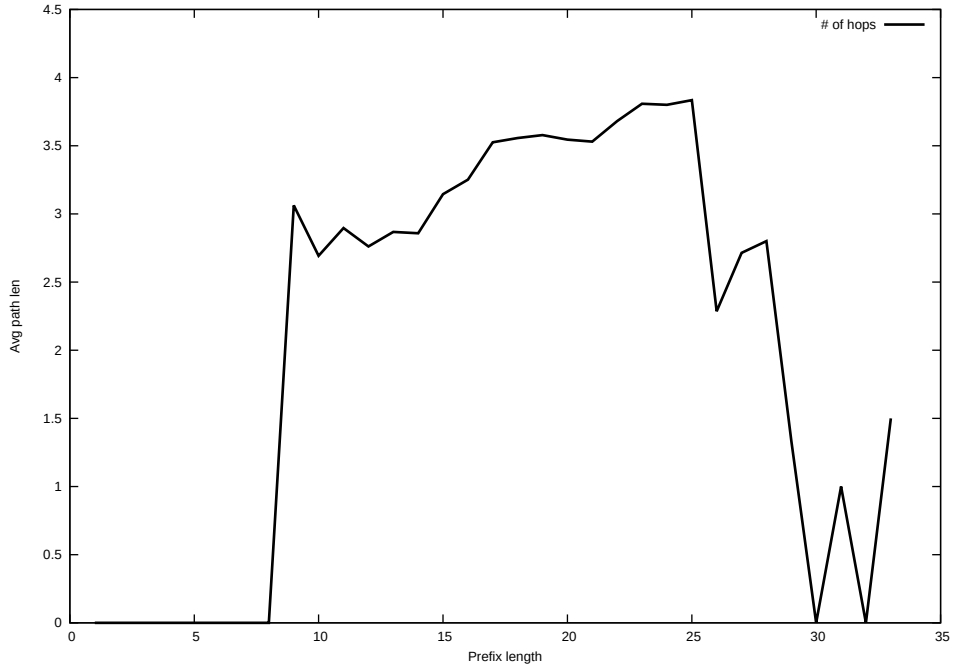
2013-04-17



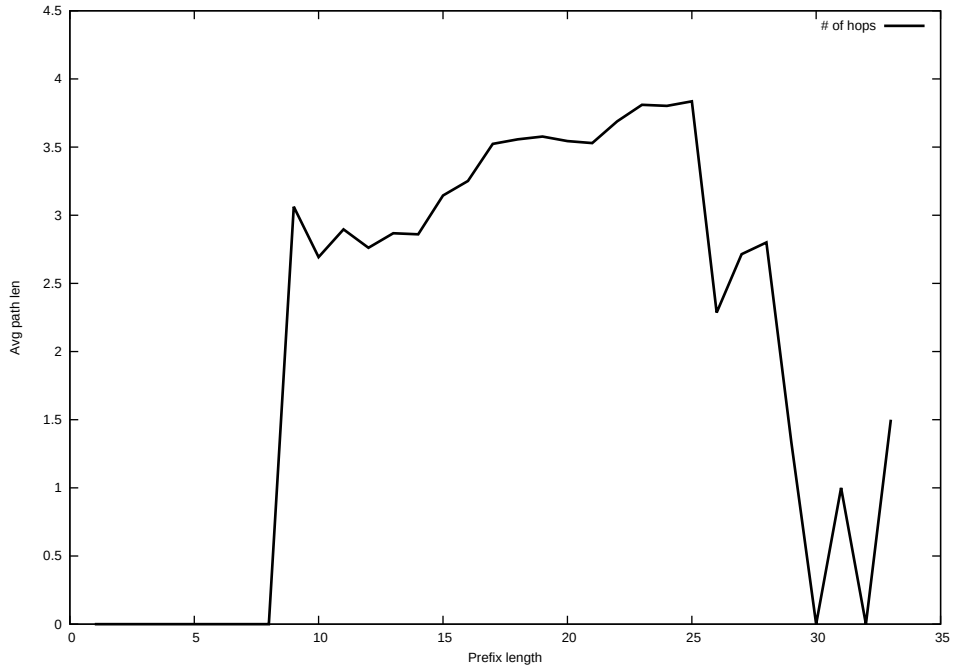
2013-04-18



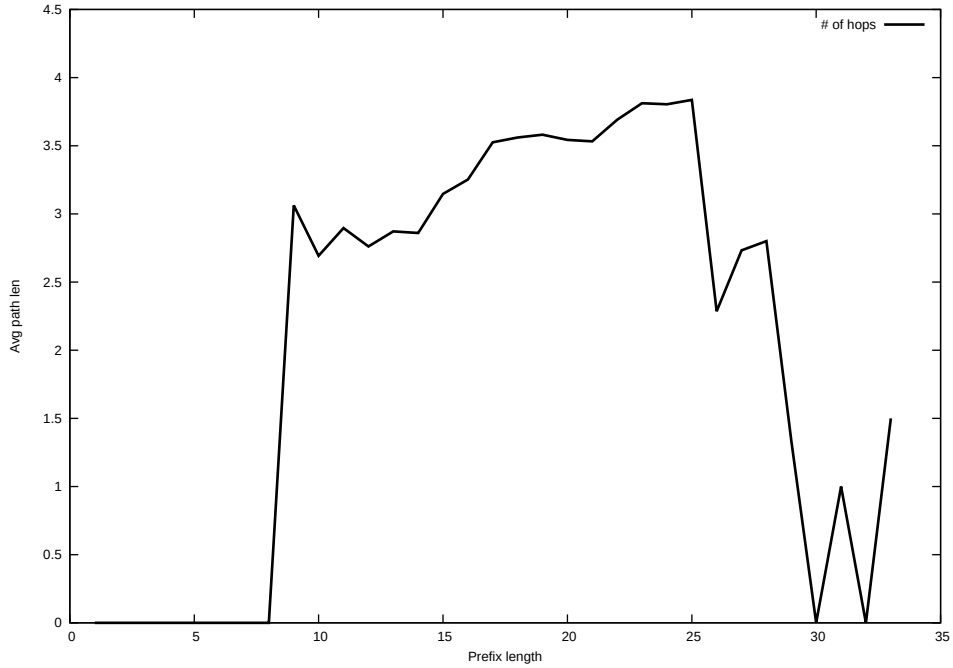
2013-04-19



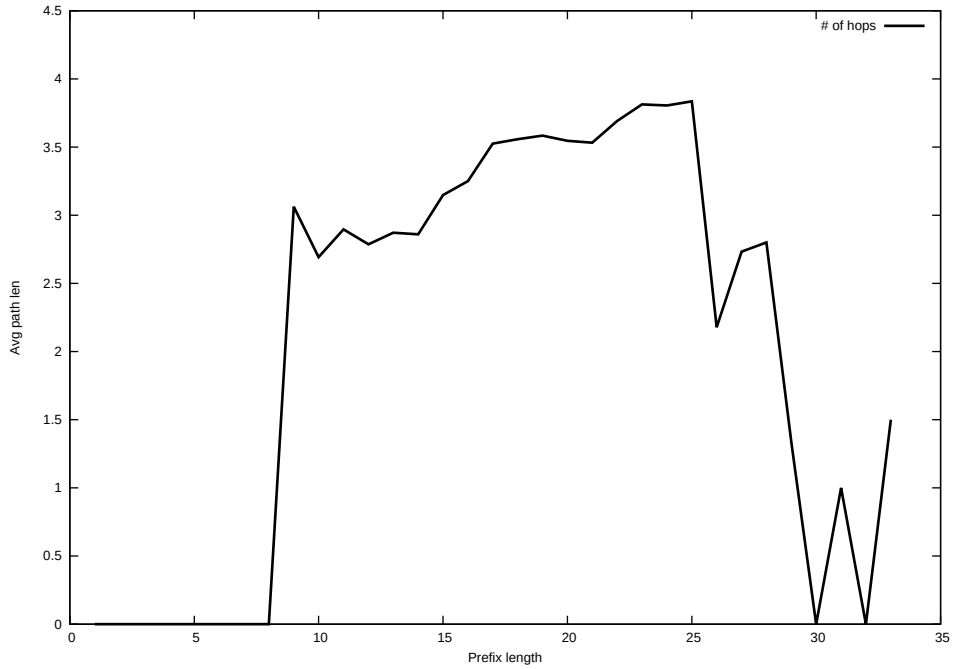
2013-04-20



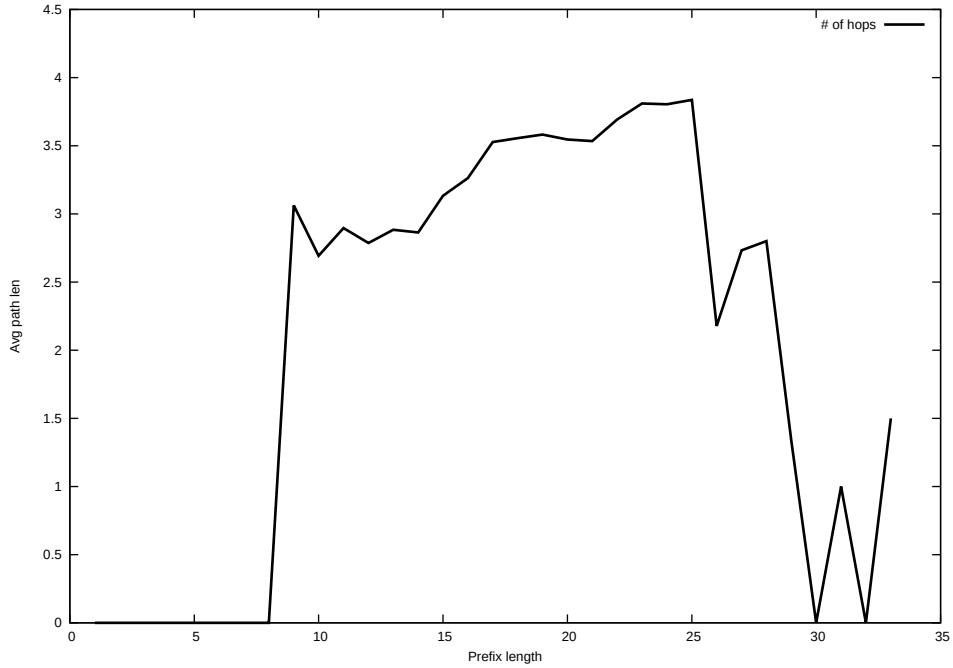
2013-04-21



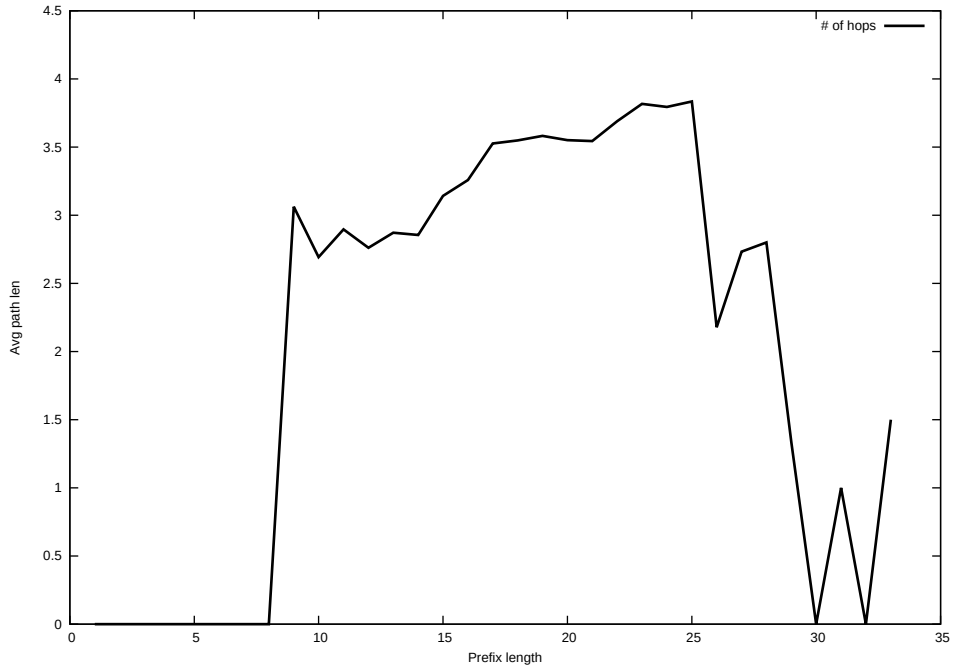
2013-04-22



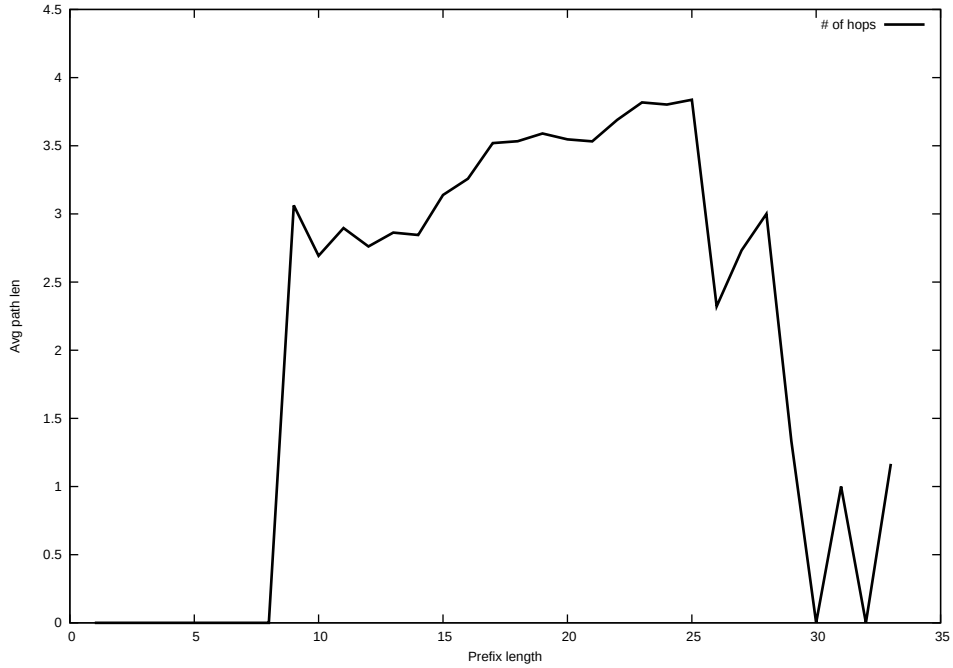
2013-04-23



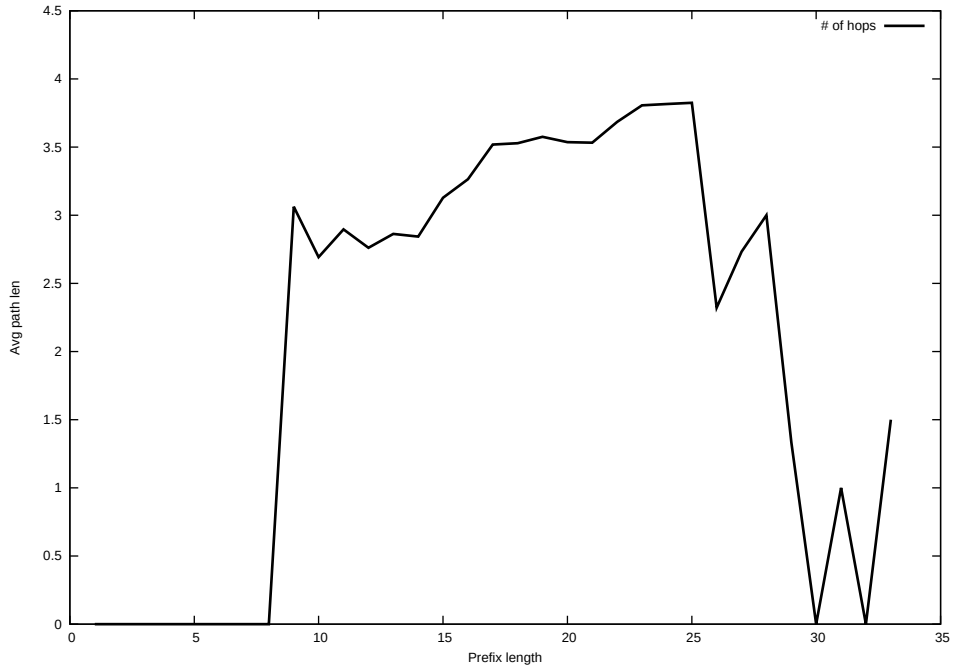
2013-04-24



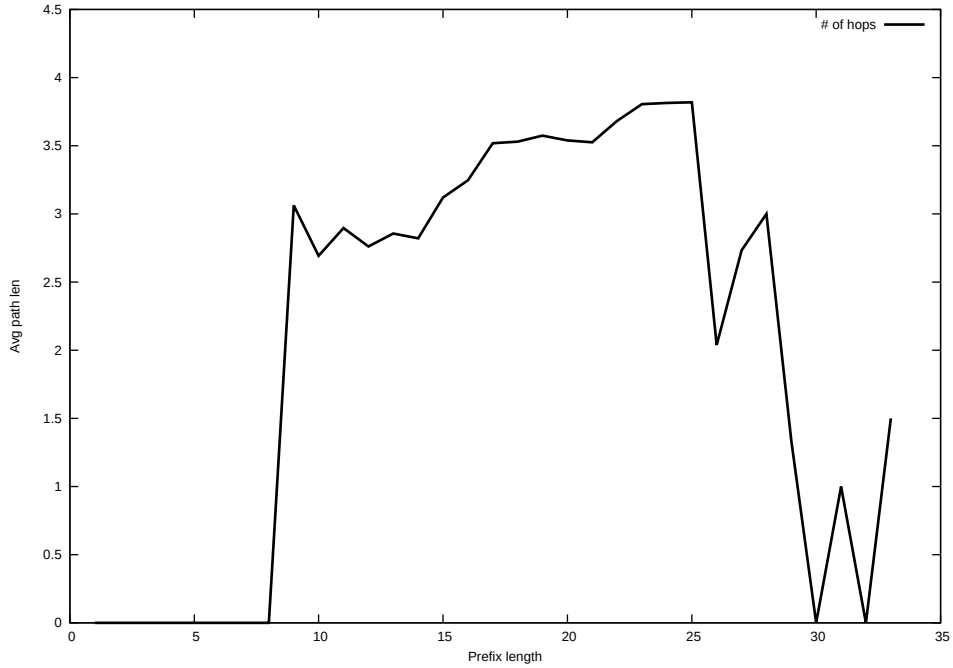
2013-04-25



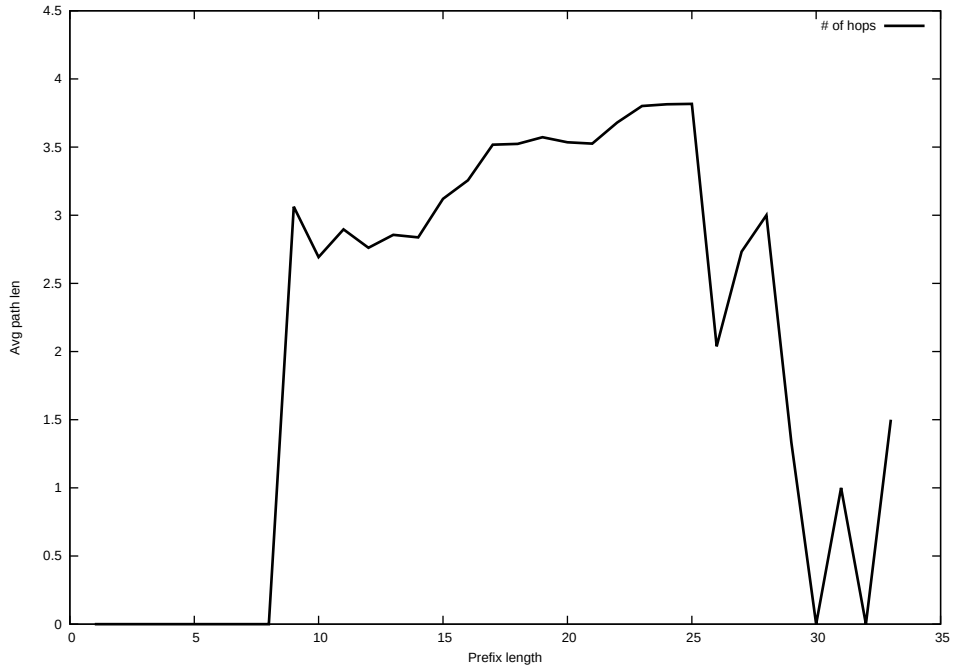
2013-04-26



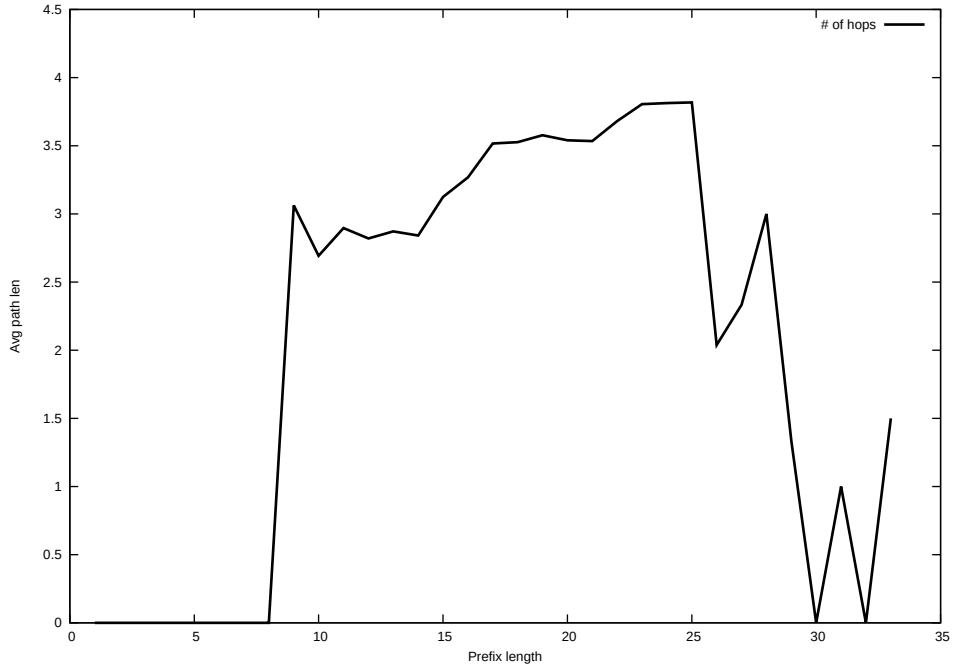
2013-04-27



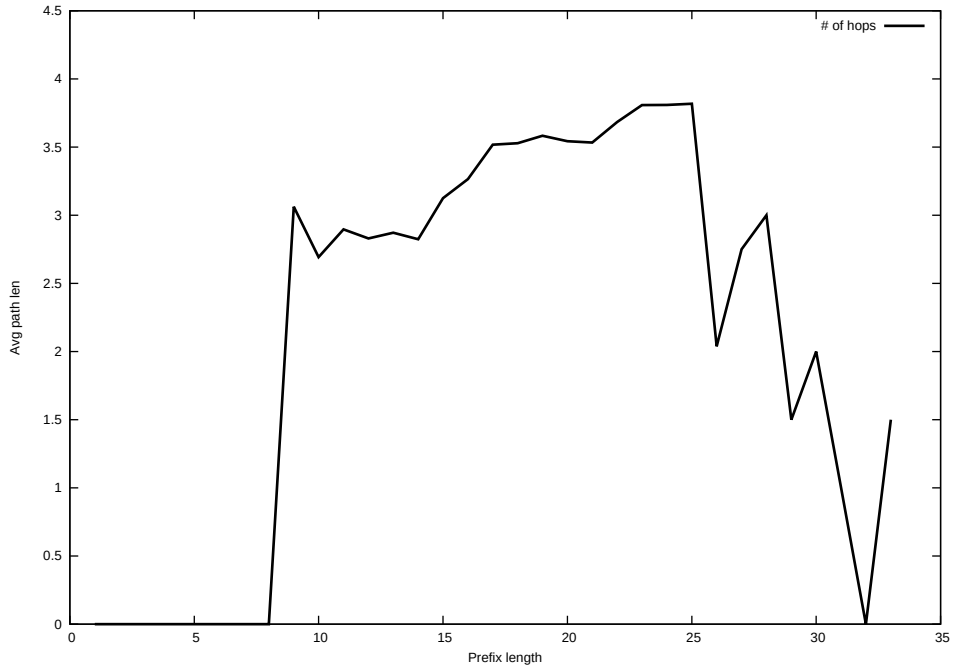
2013-04-28



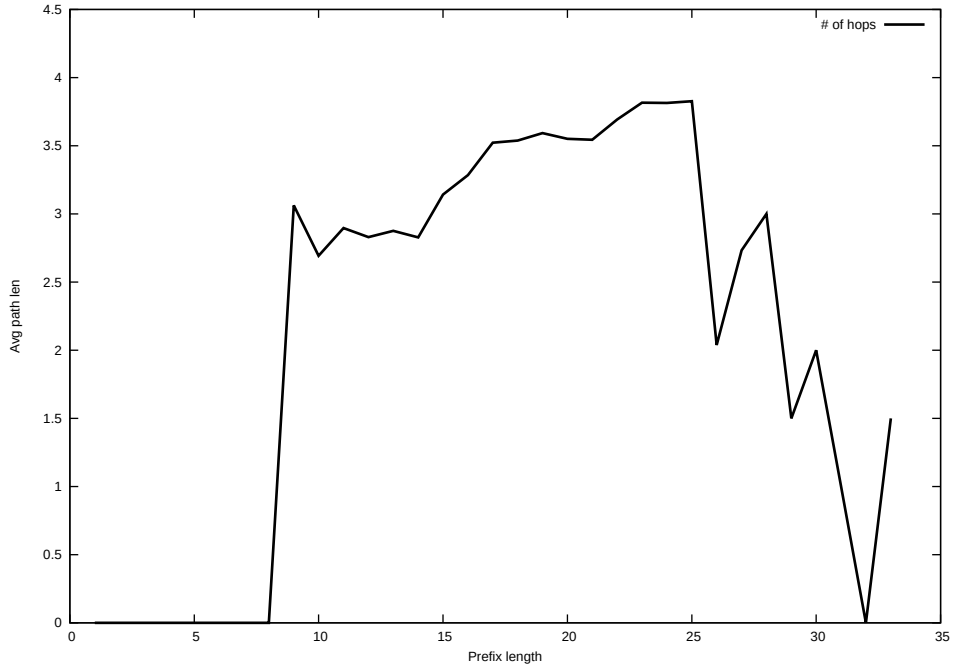
2013-04-29



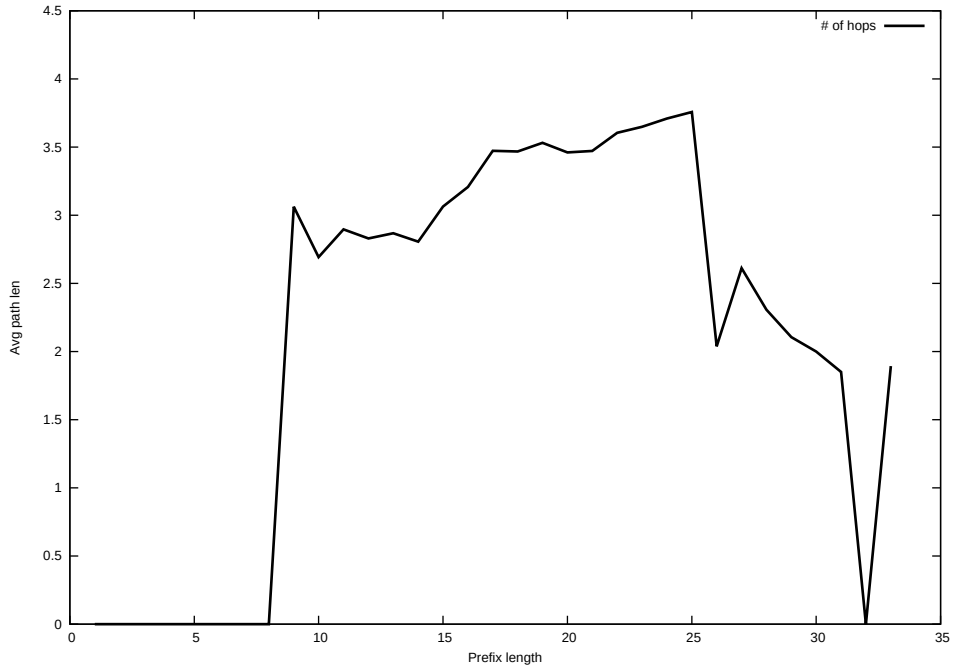
2013-04-30



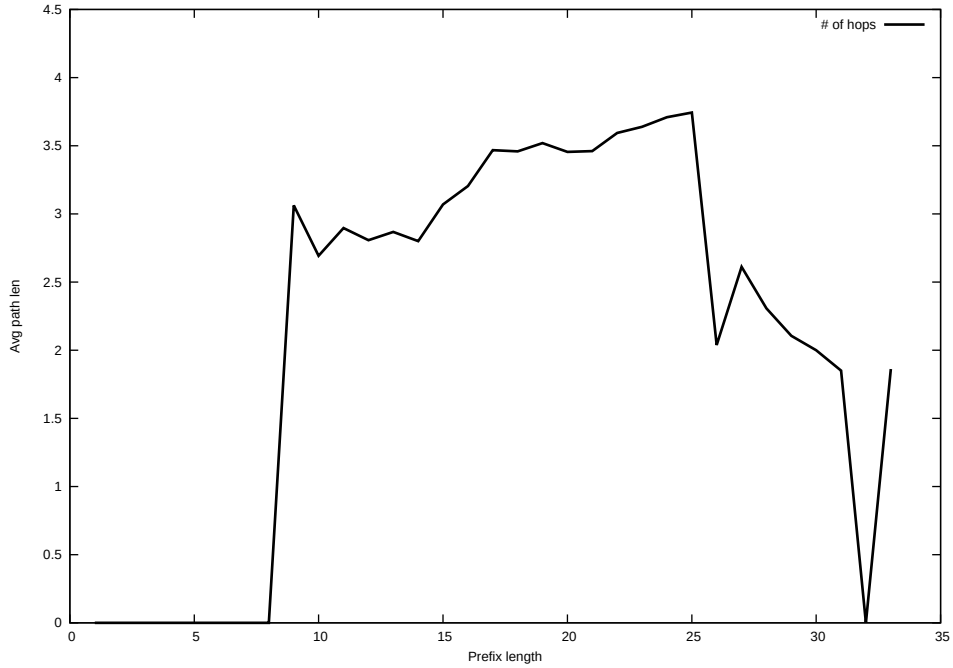
2013-05-01



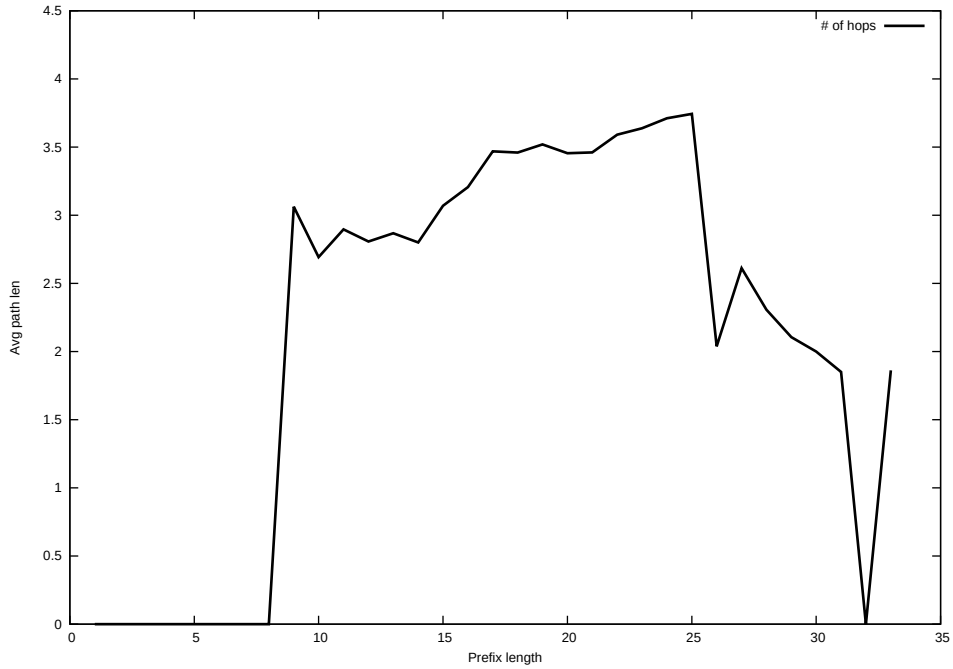
2013-05-02



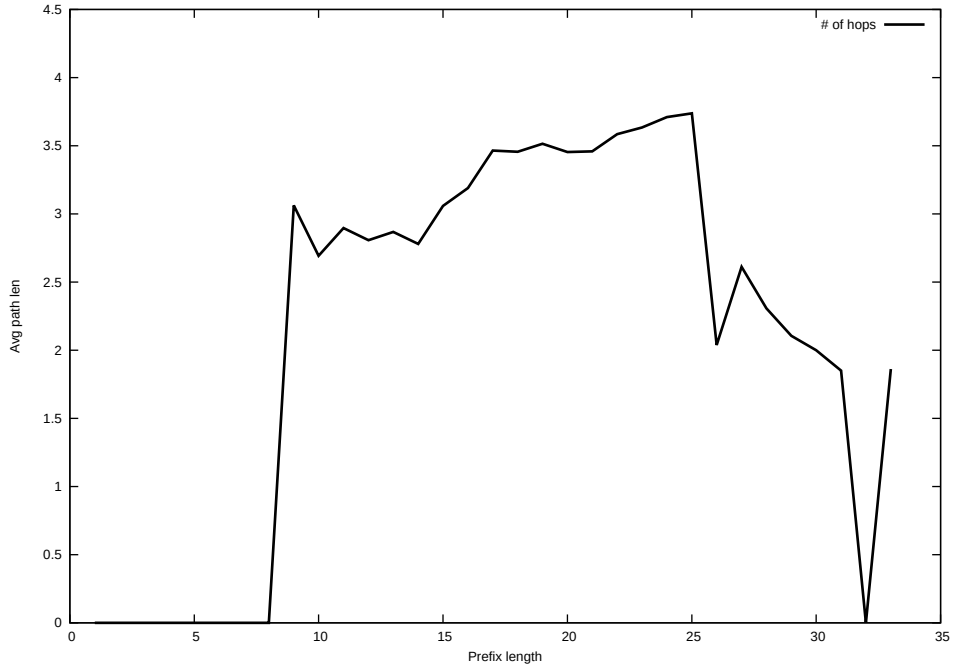
2013-05-03



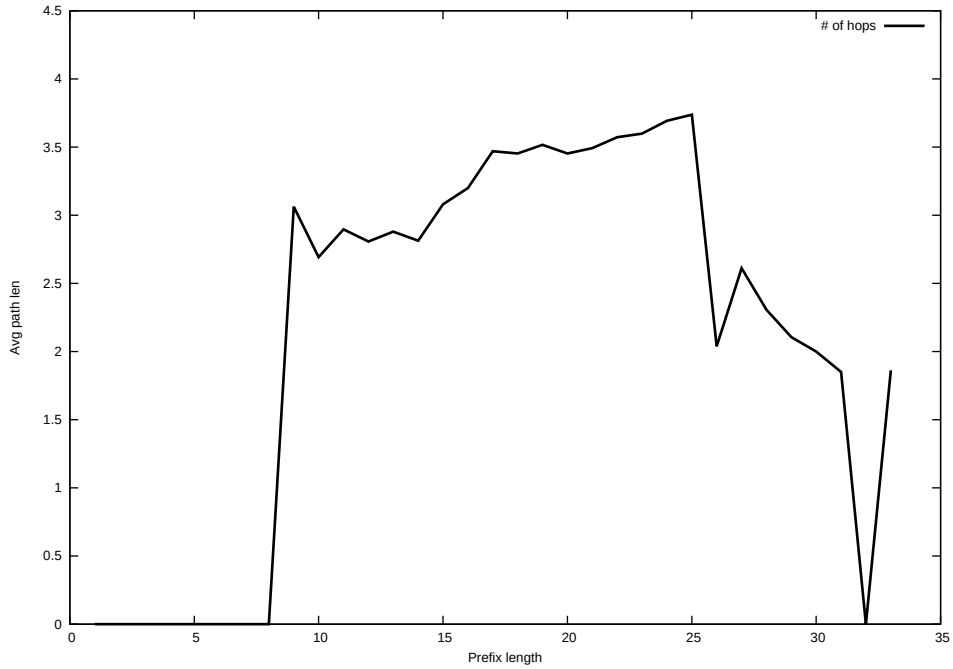
2013-05-04



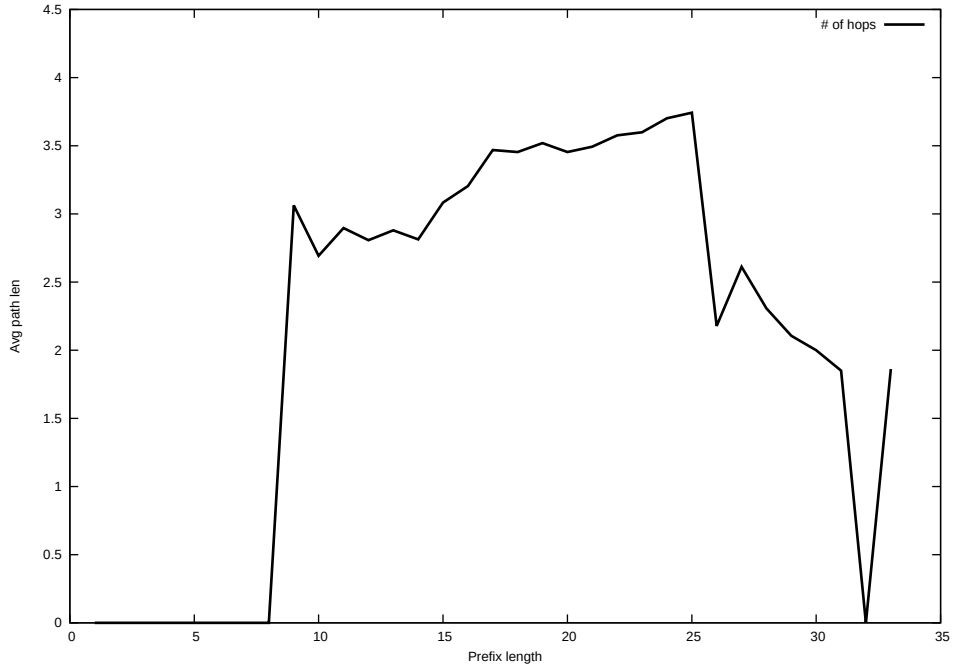
2013-05-05



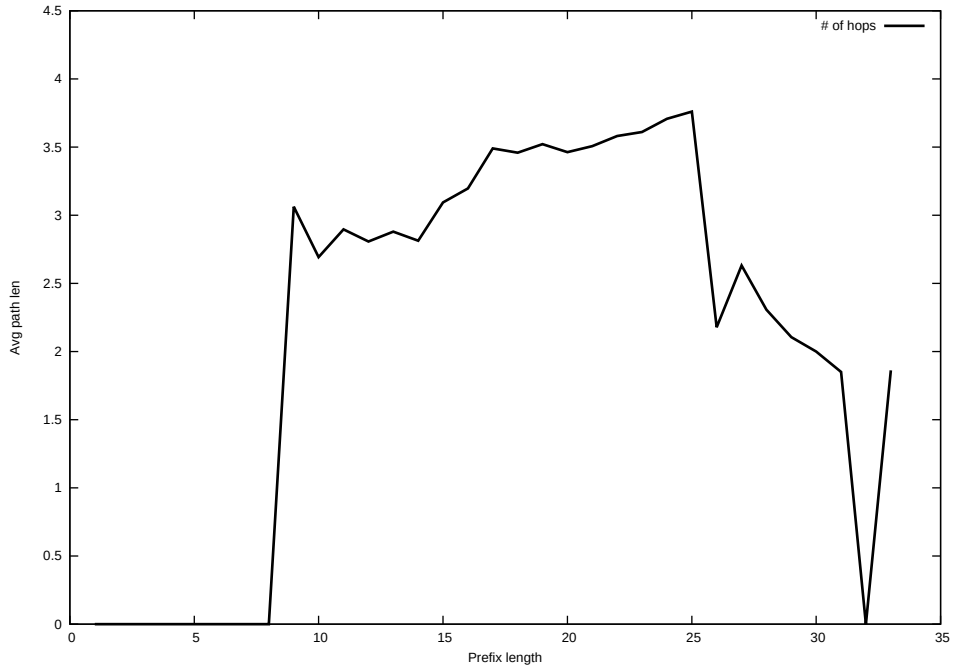
2013-05-06



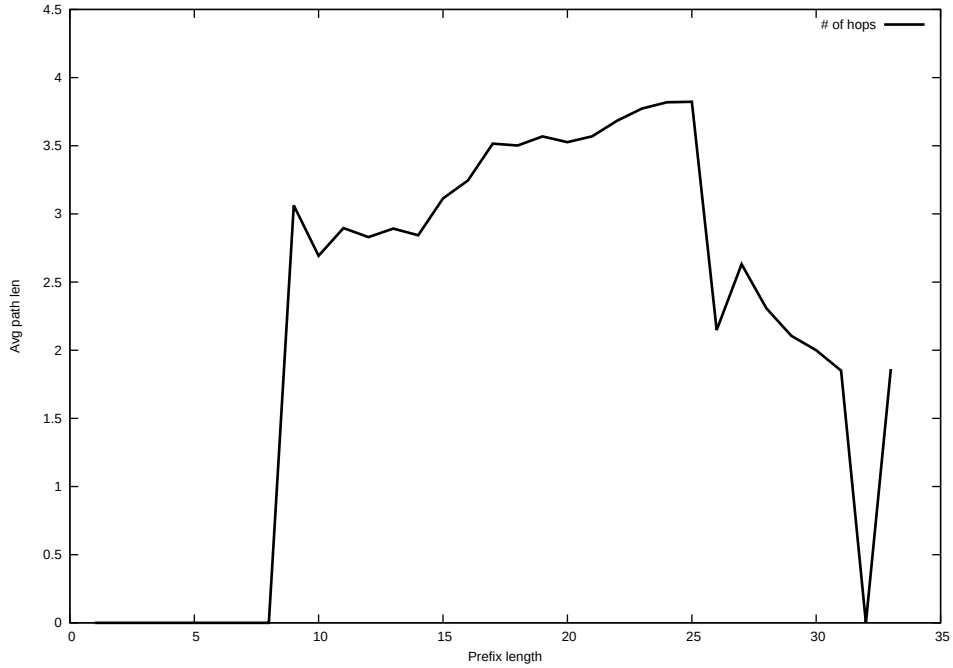
2013-05-07



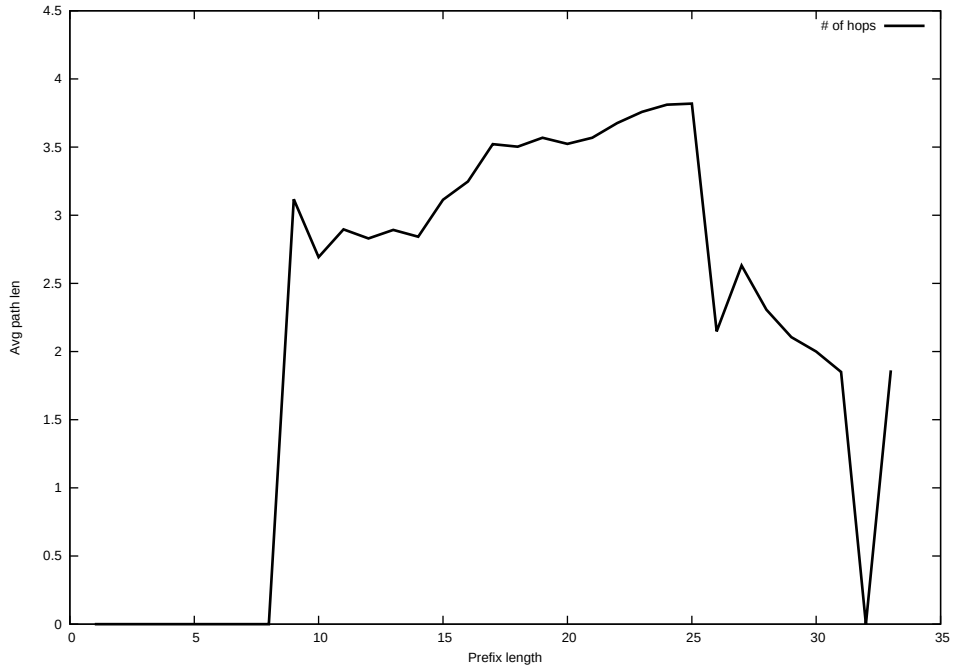
2013-05-08



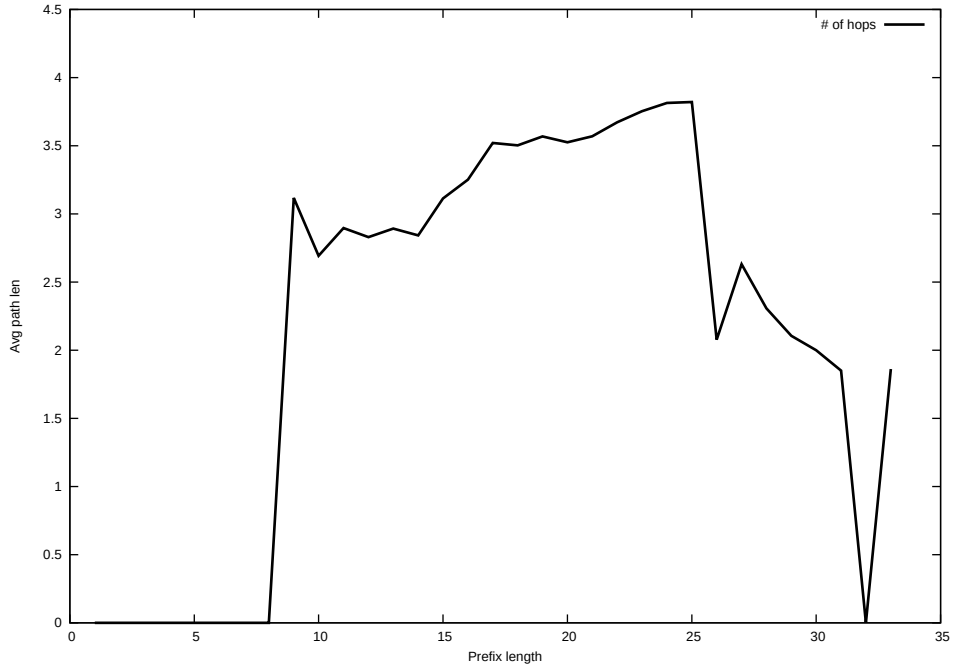
2013-05-09



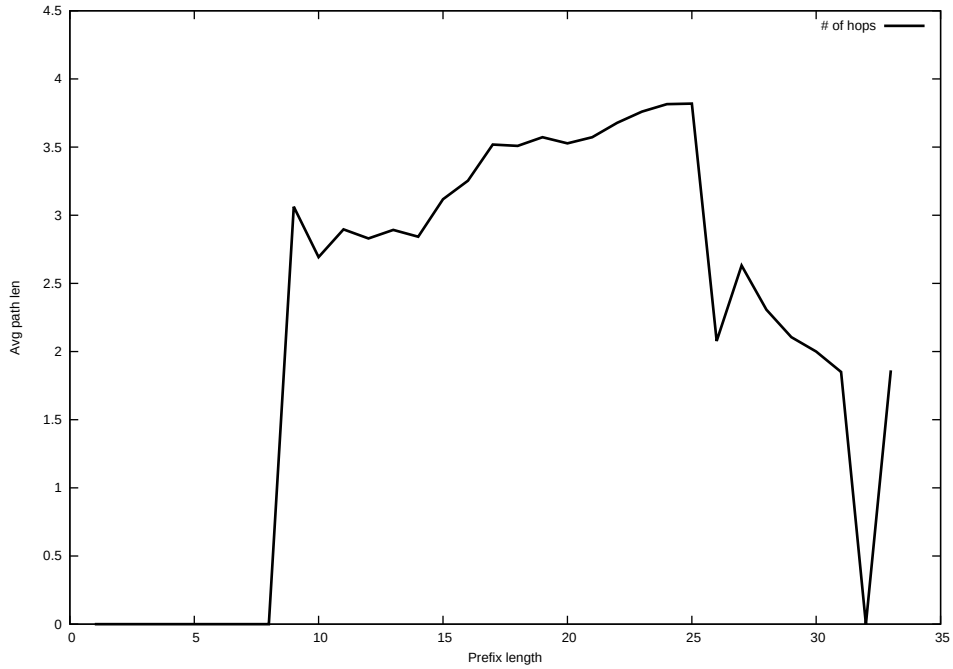
2013-05-10



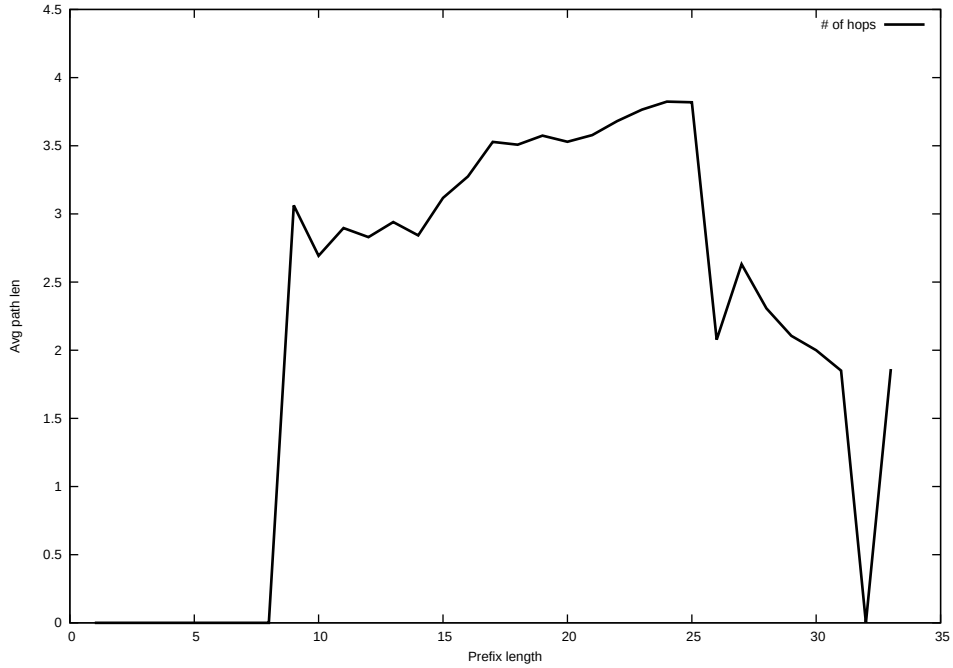
2013-05-11



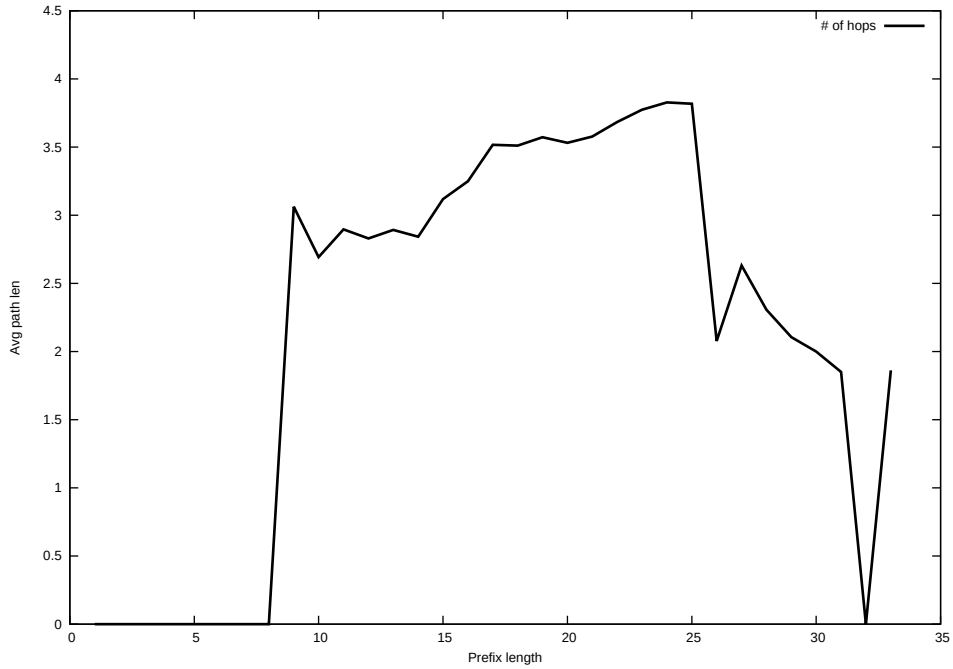
2013-05-12



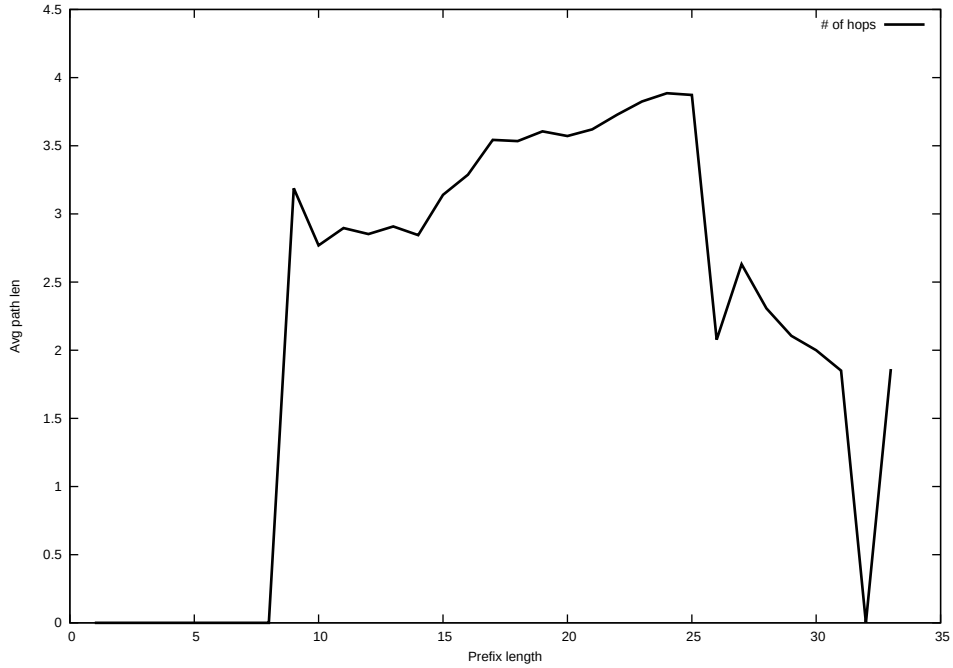
2013-05-13



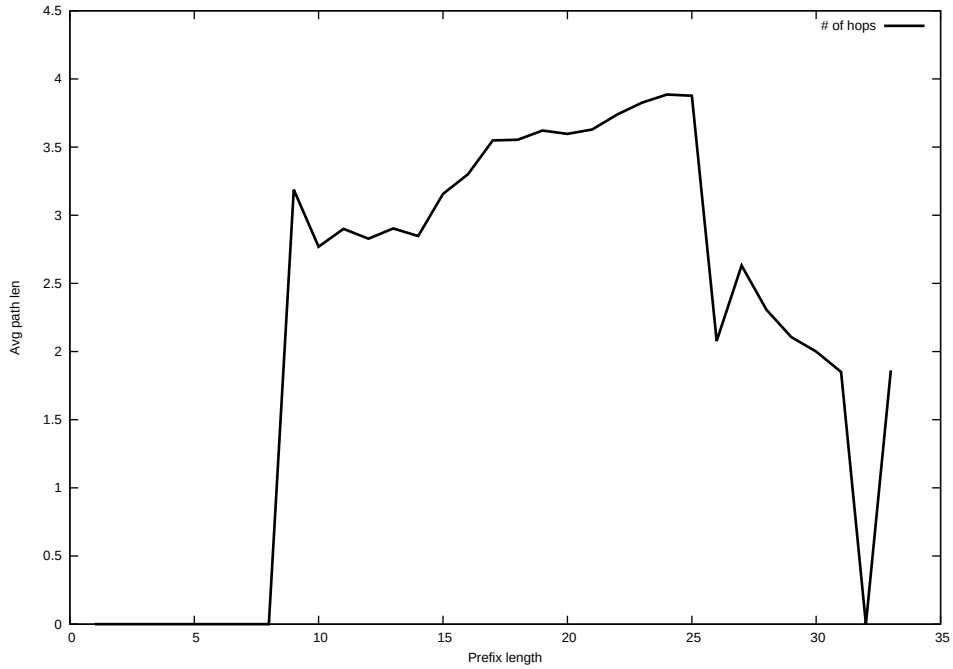
2013-05-14



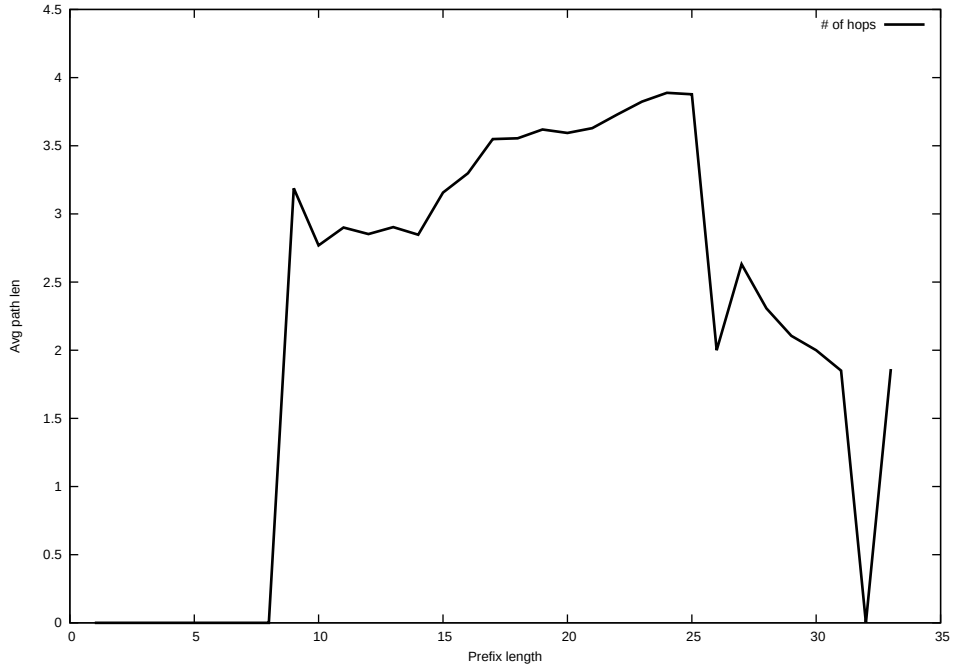
2013-05-15



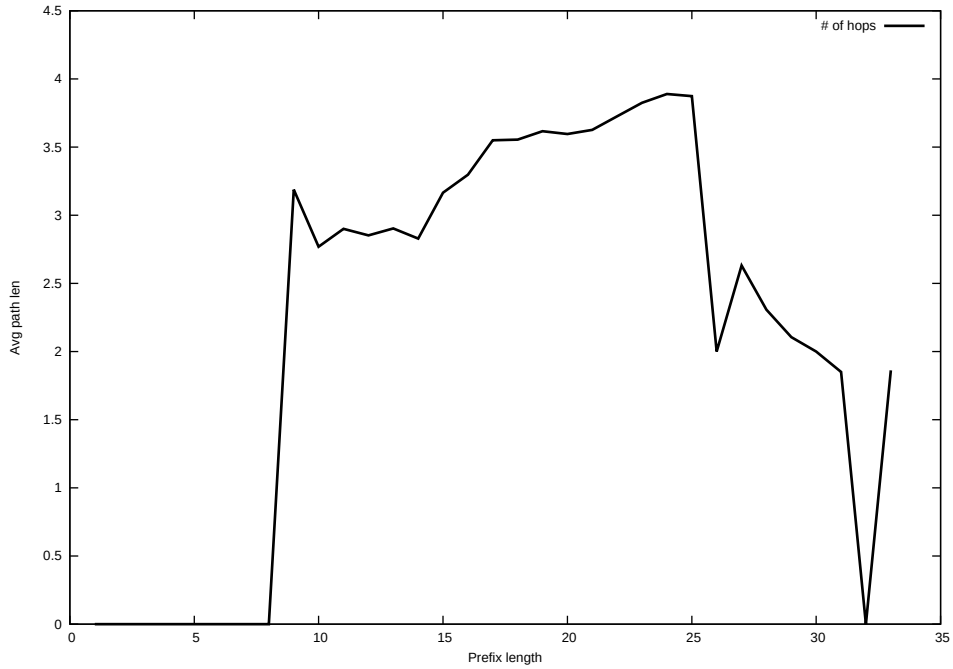
2013-05-16



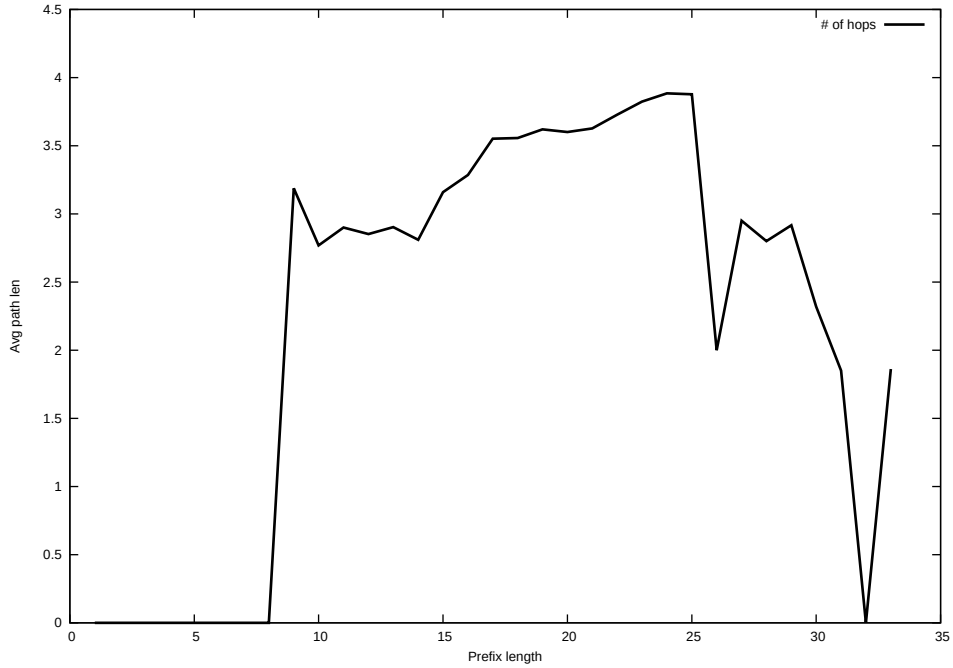
2013-05-17



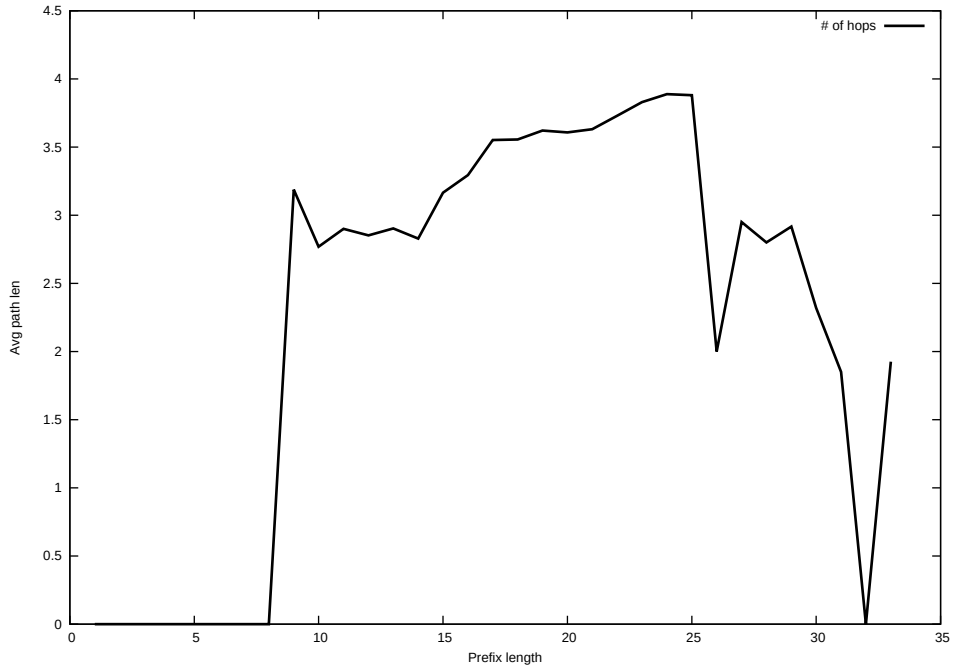
2013-05-18



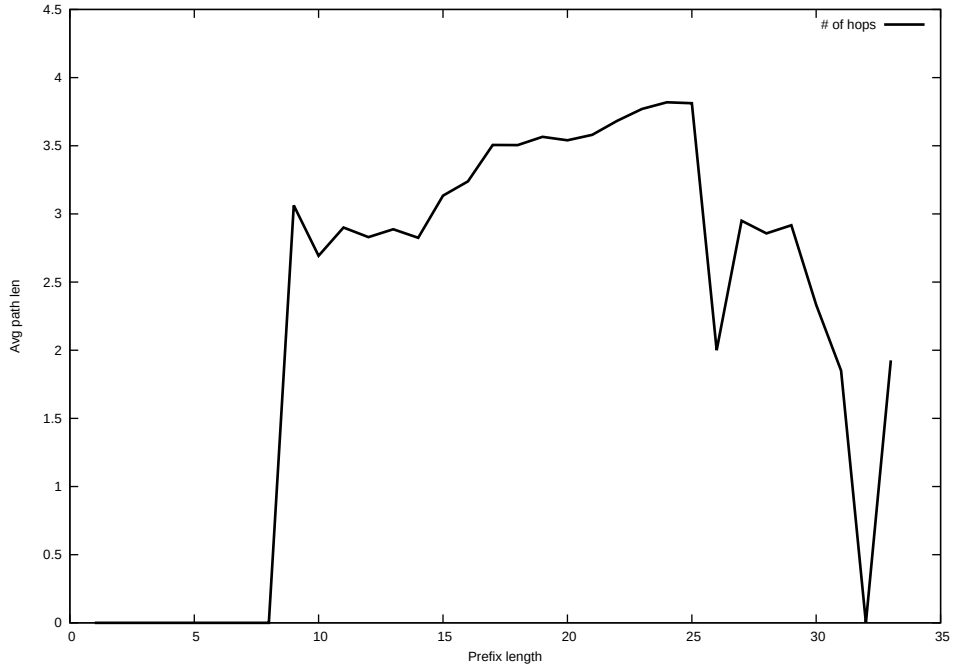
2013-05-19



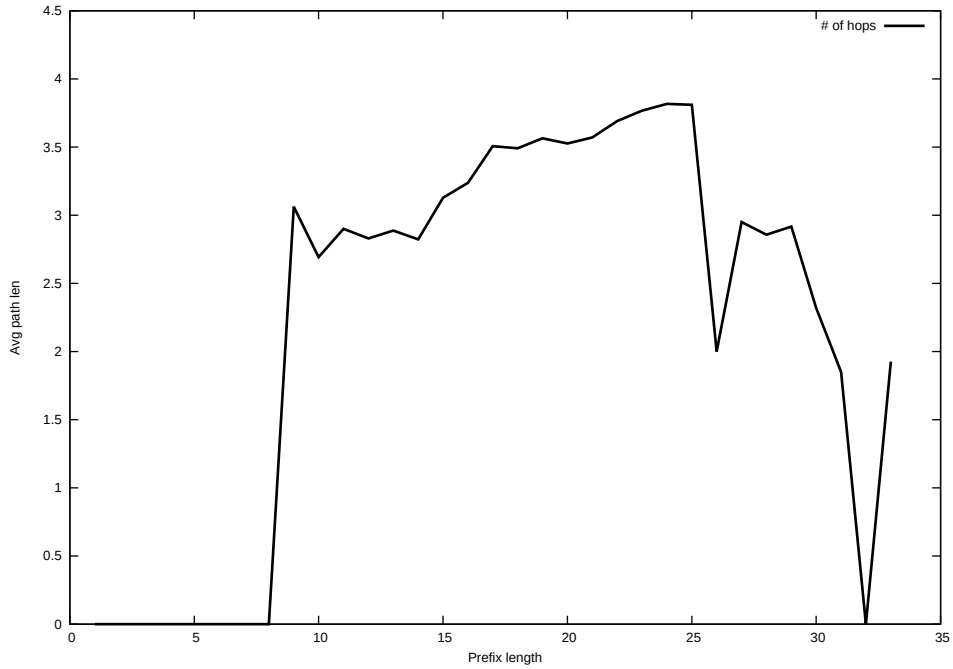
2013-05-20



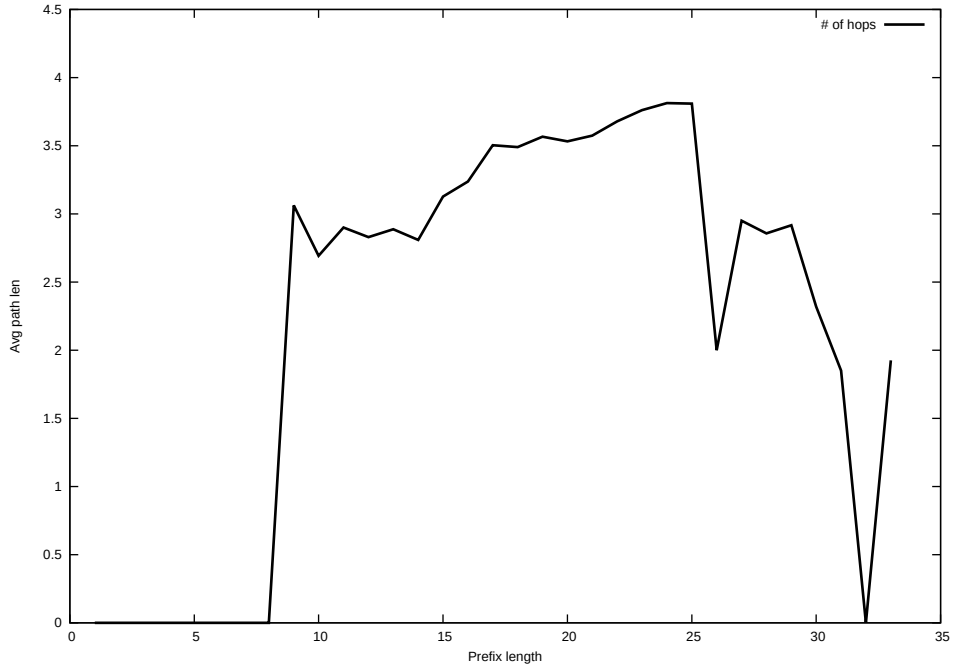
2013-05-21



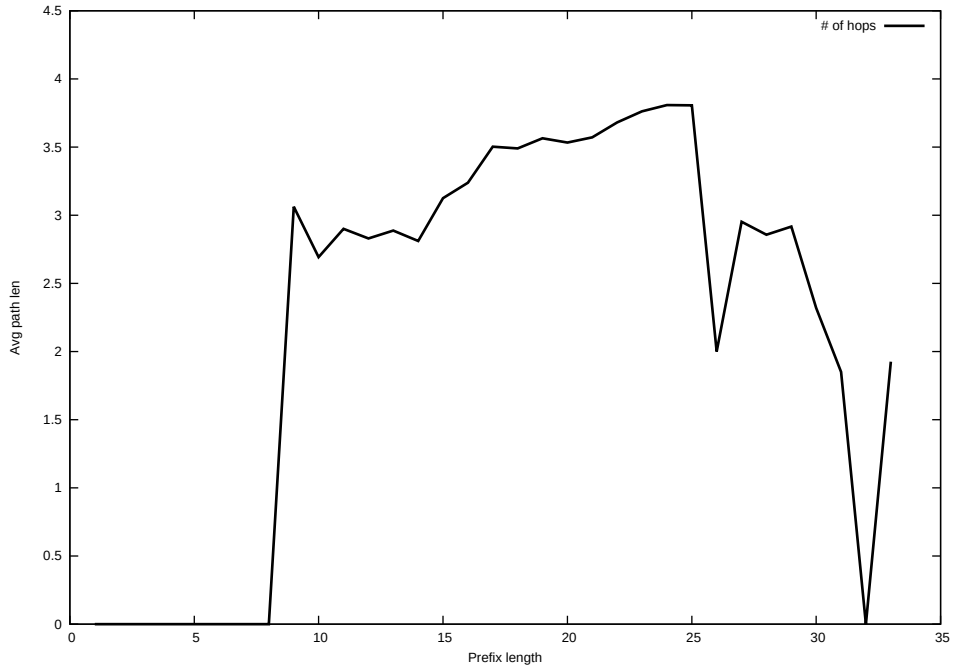
2013-05-22



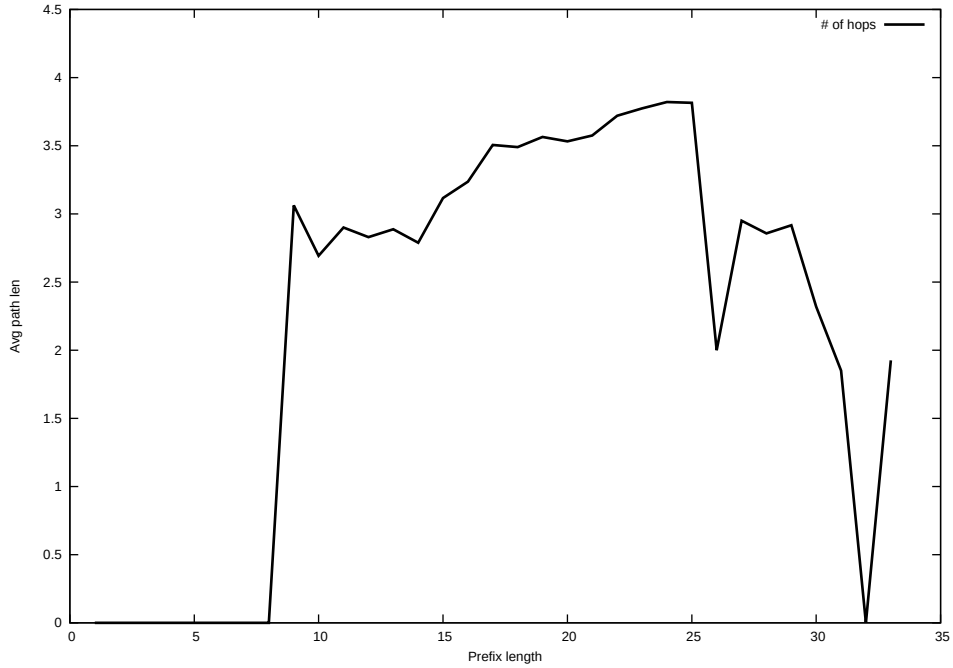
2013-05-23



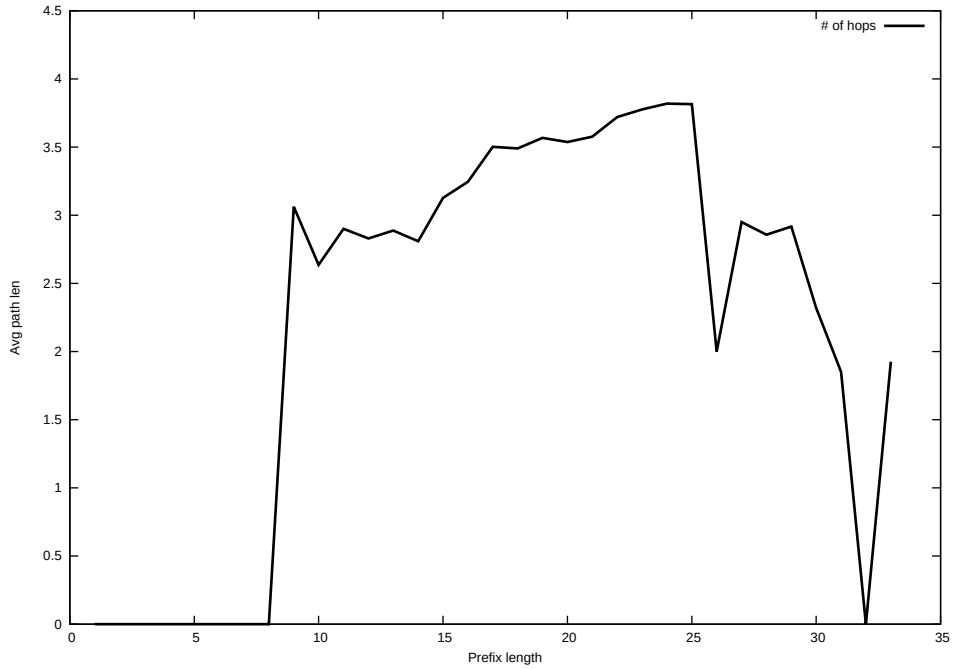
2013-05-24



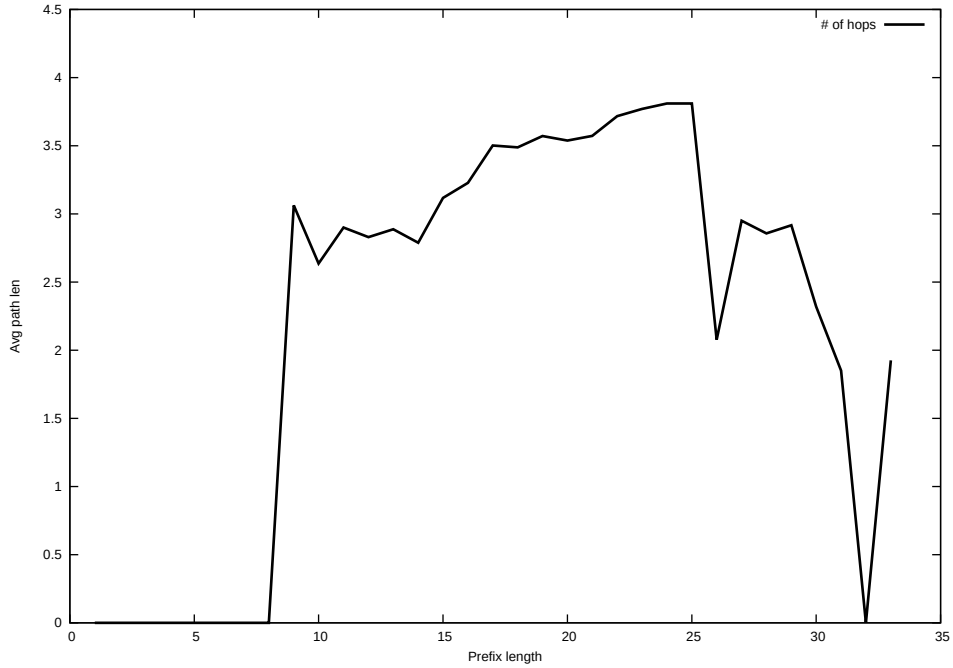
2013-05-25



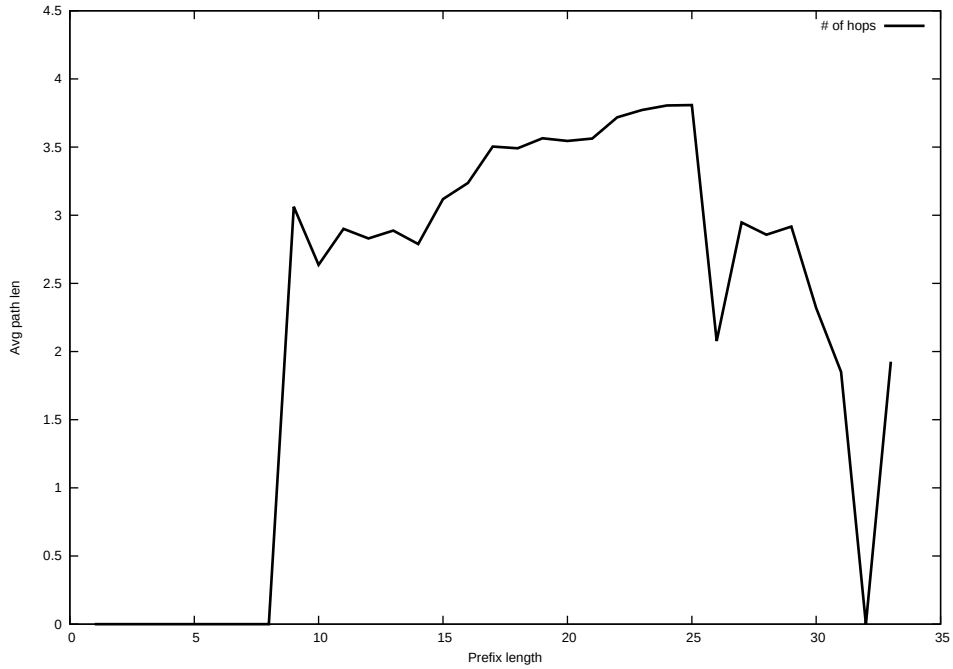
2013-05-26



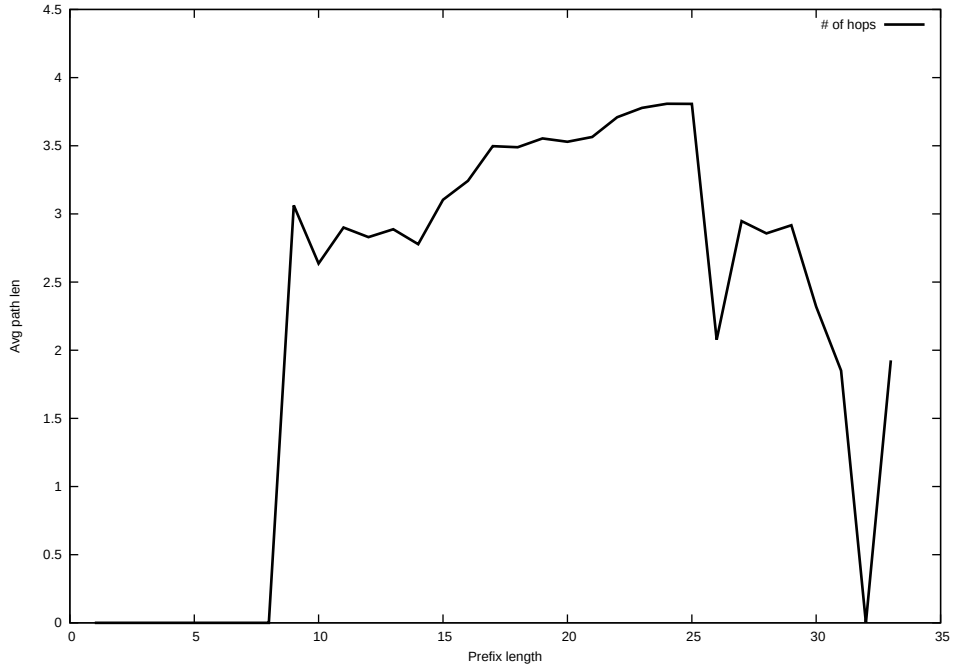
2013-05-27



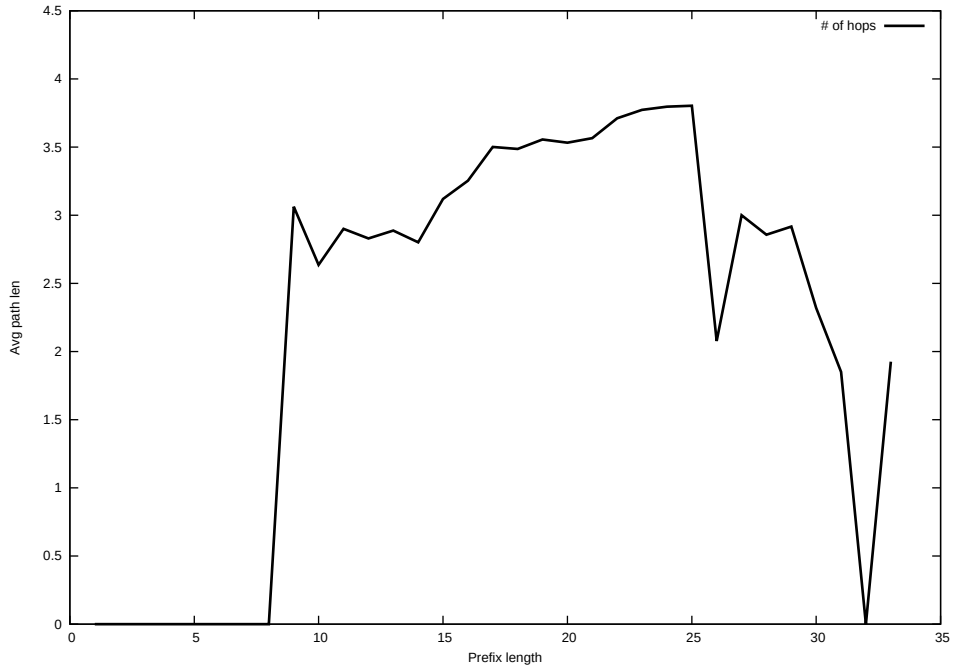
2013-05-28



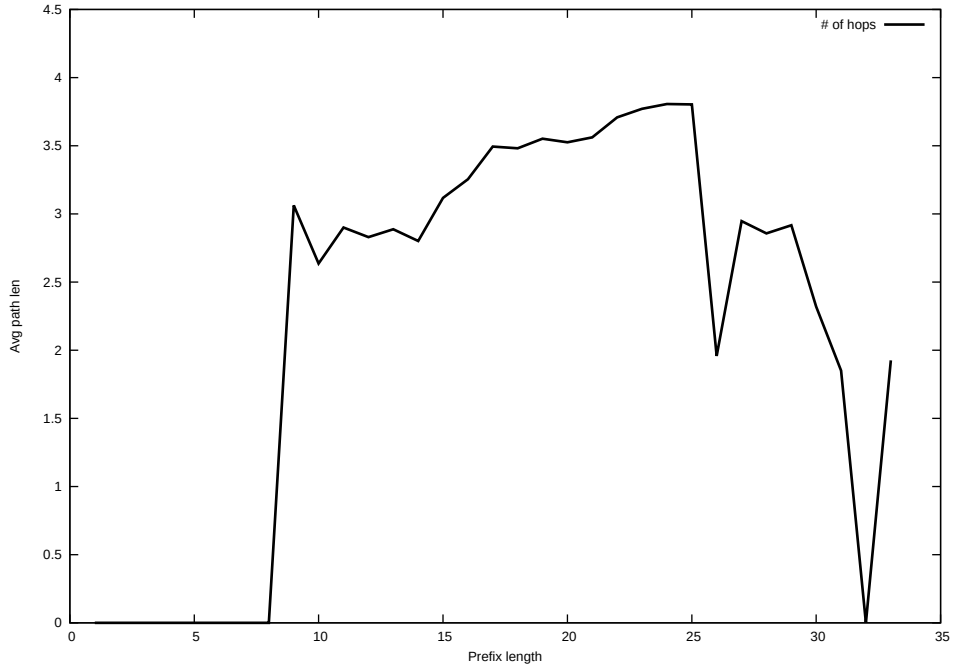
2013-05-29



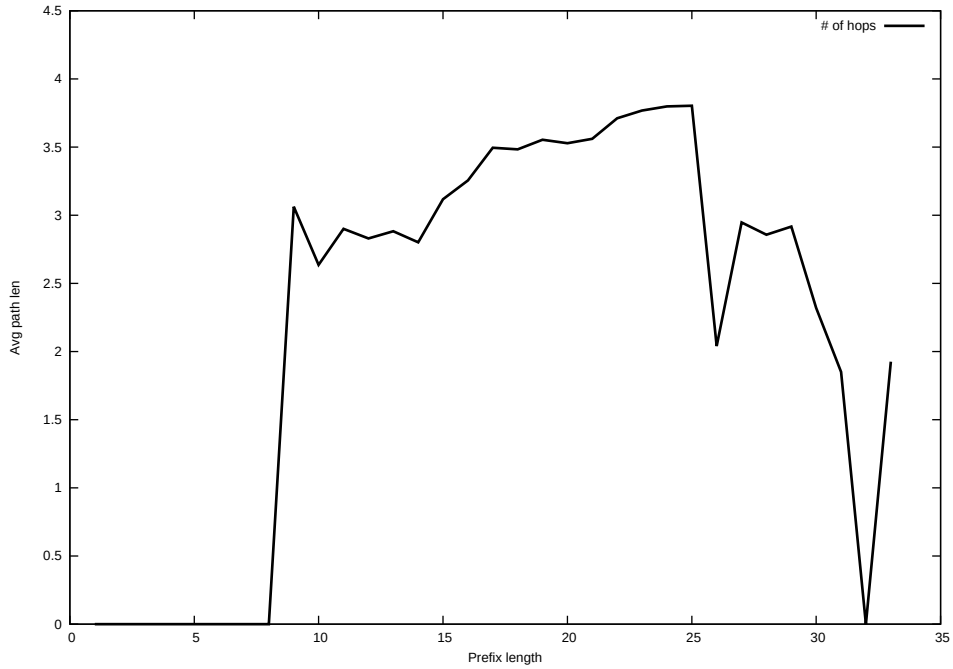
2013-05-30



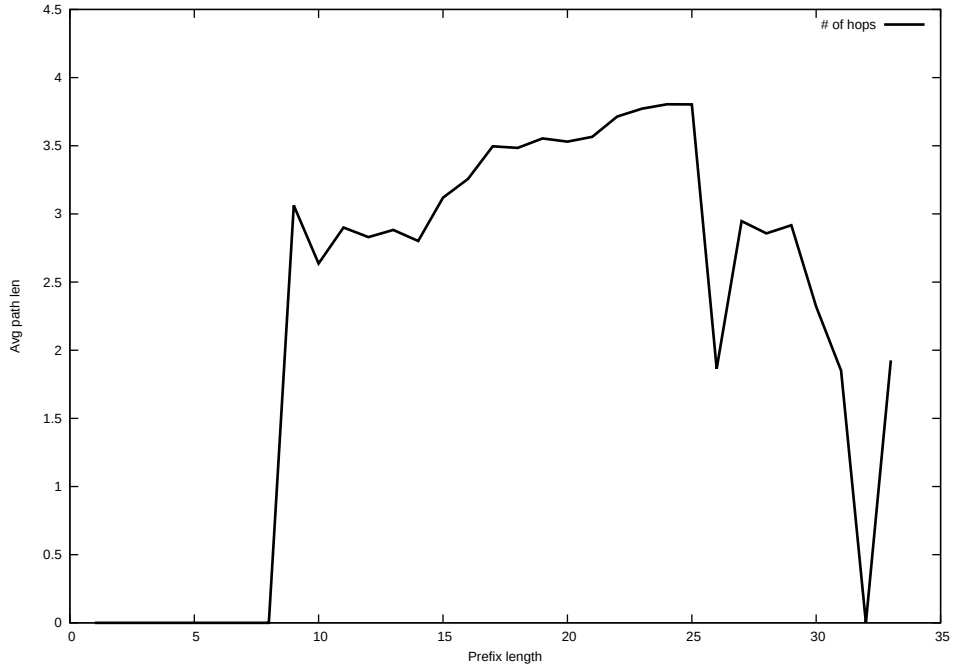
2013-05-31



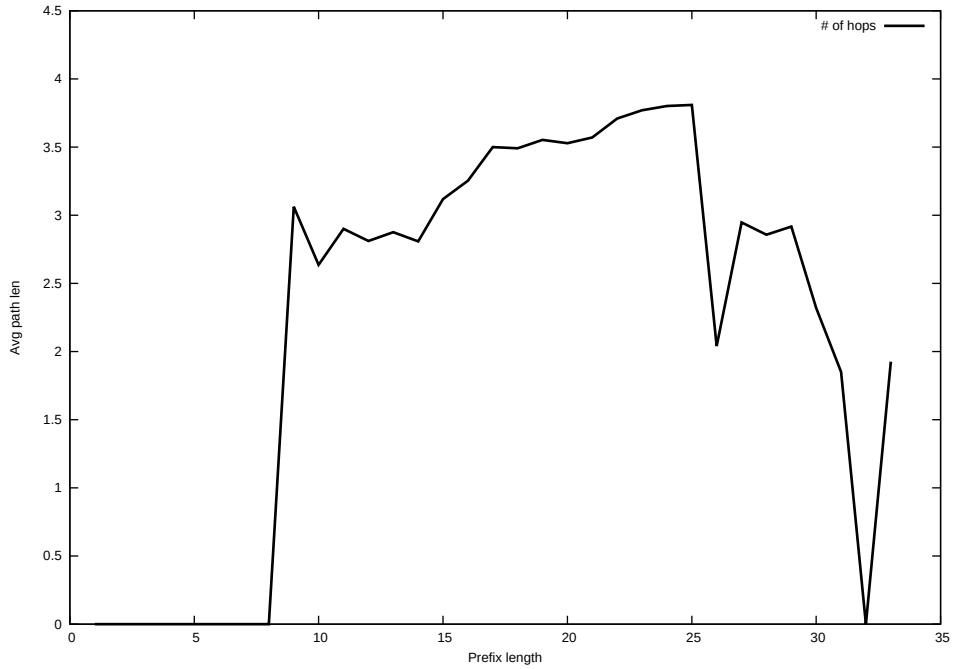
2013-06-01



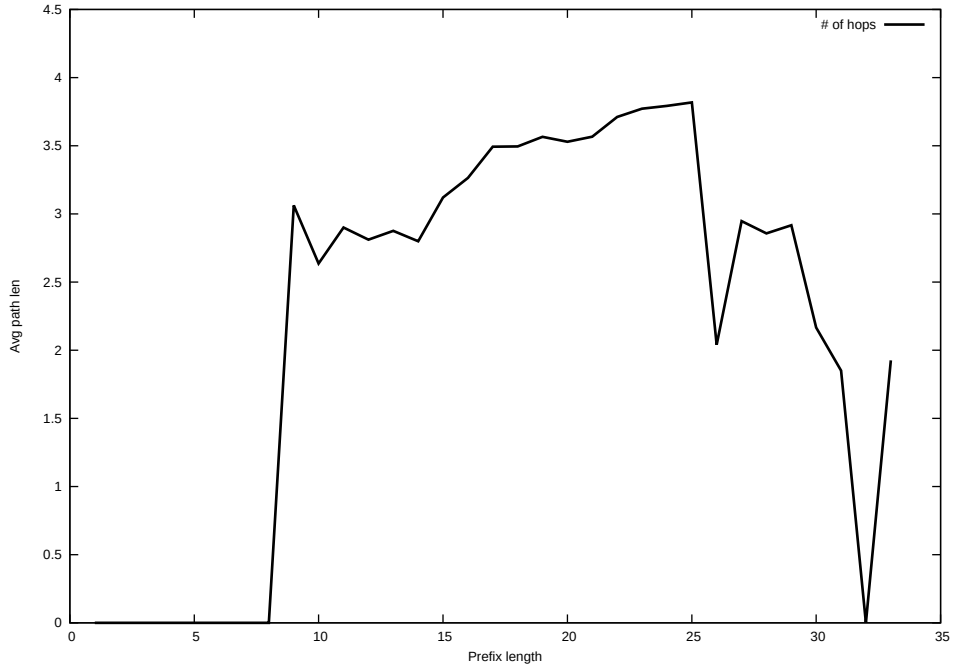
2013-06-02



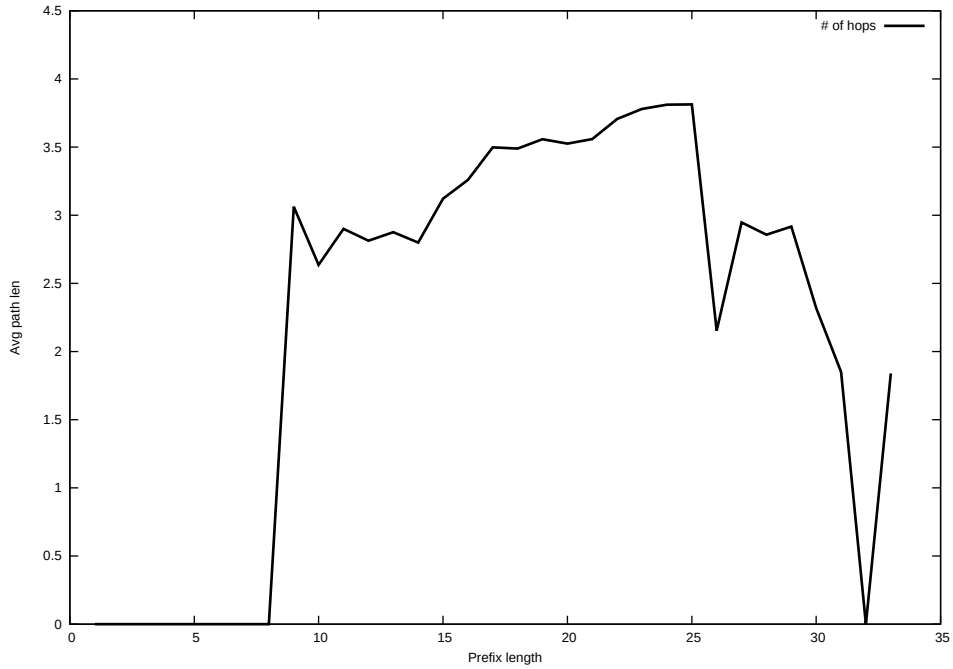
2013-06-03



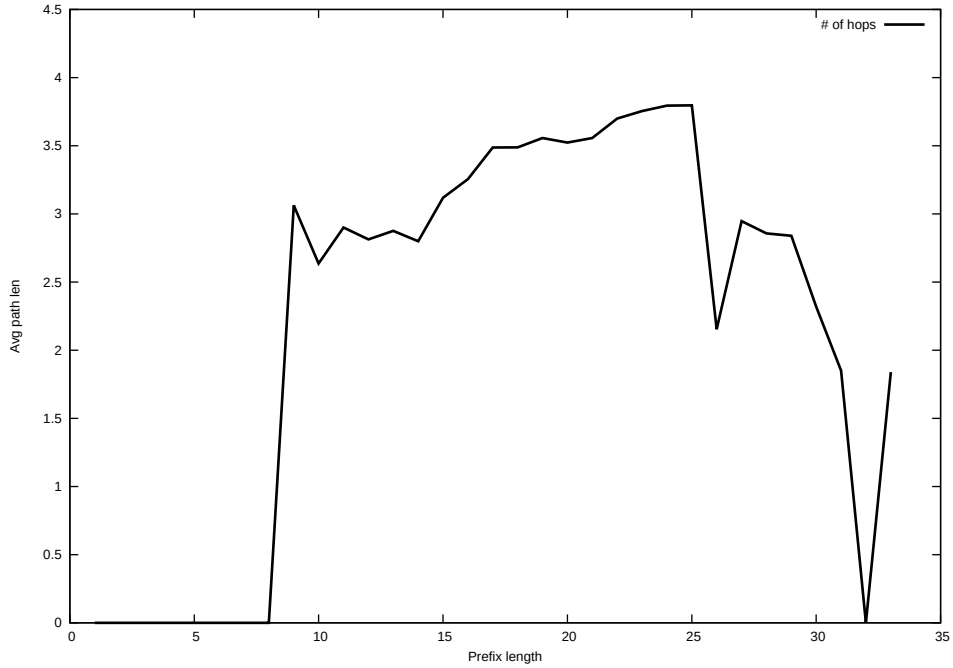
2013-06-04



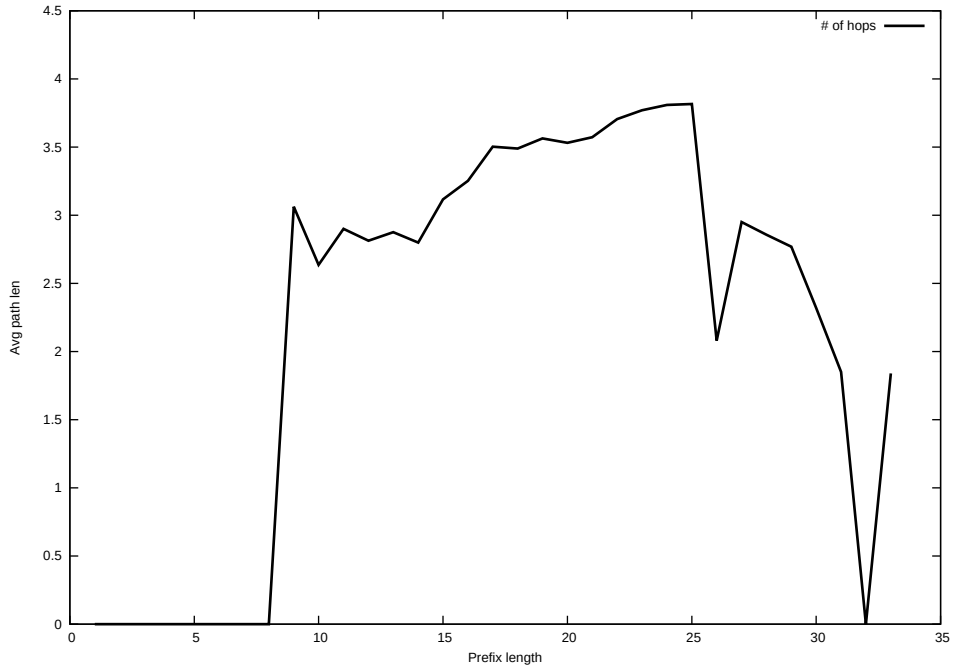
2013-06-05



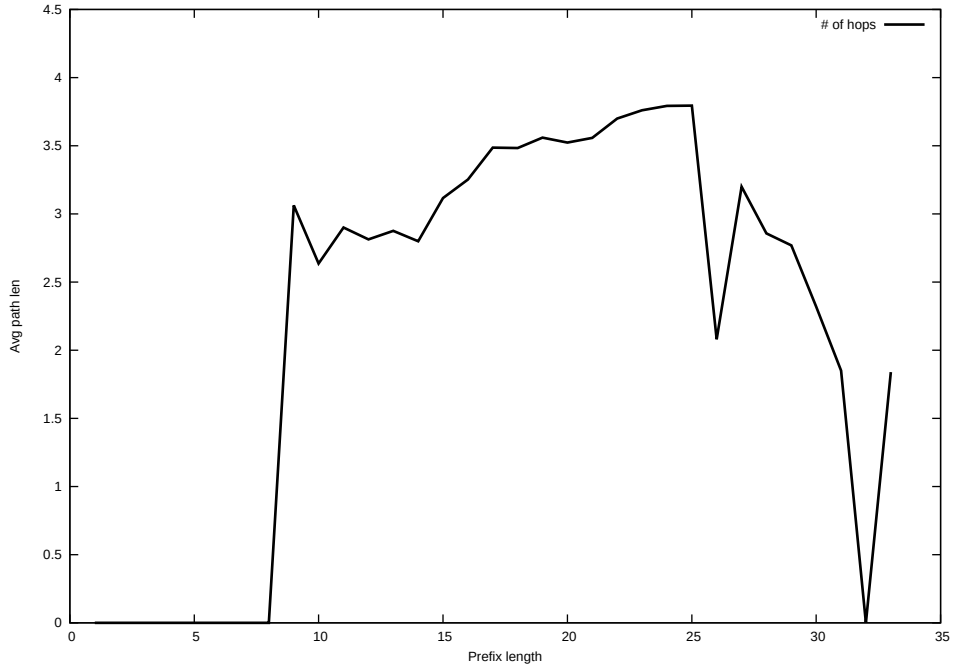
2013-06-06



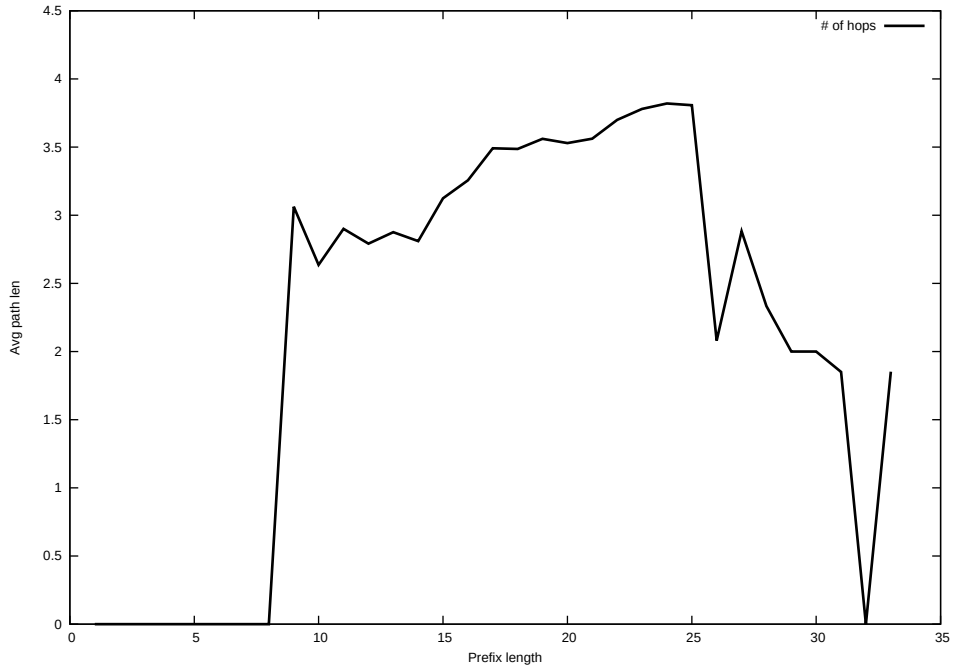
2013-06-07



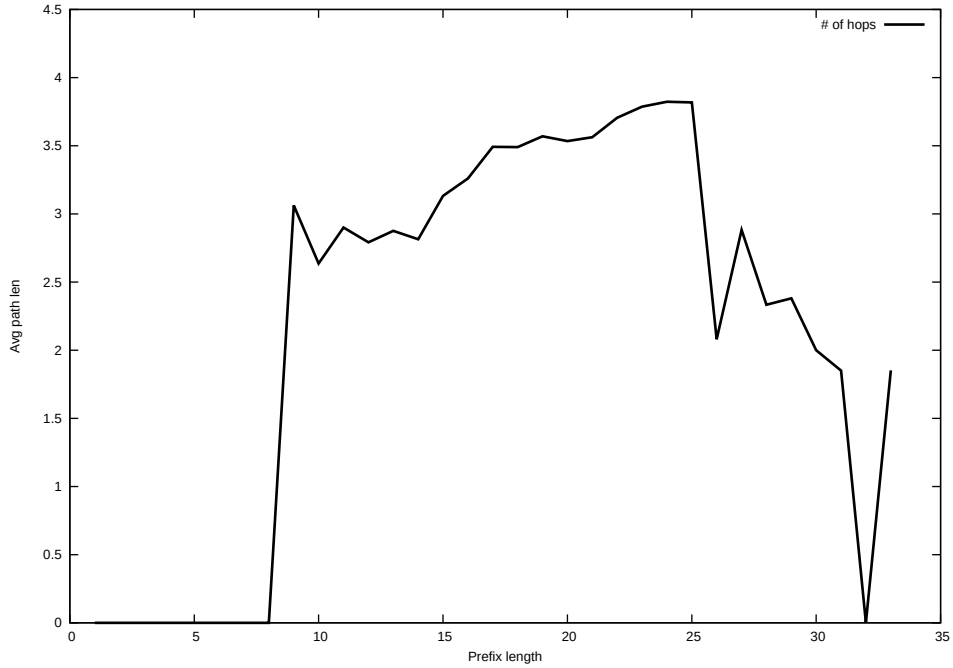
2013-06-09



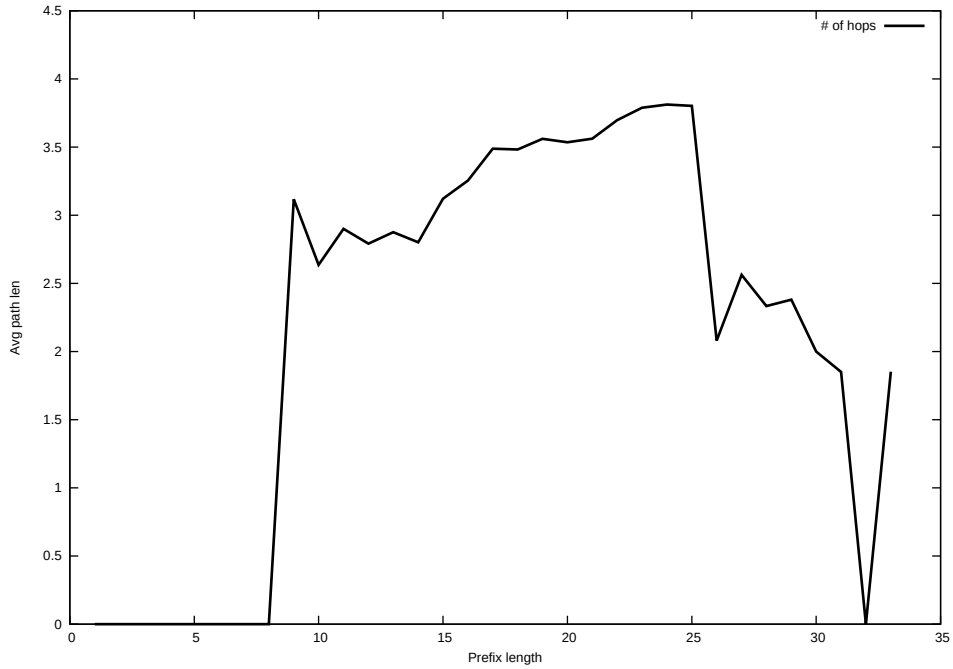
2013-06-10



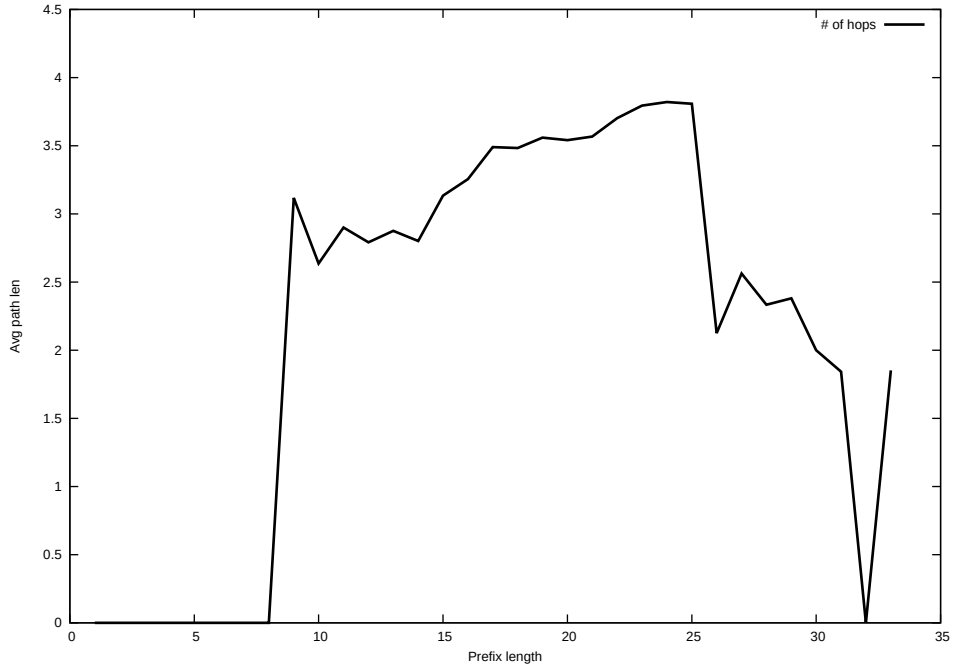
2013-06-12



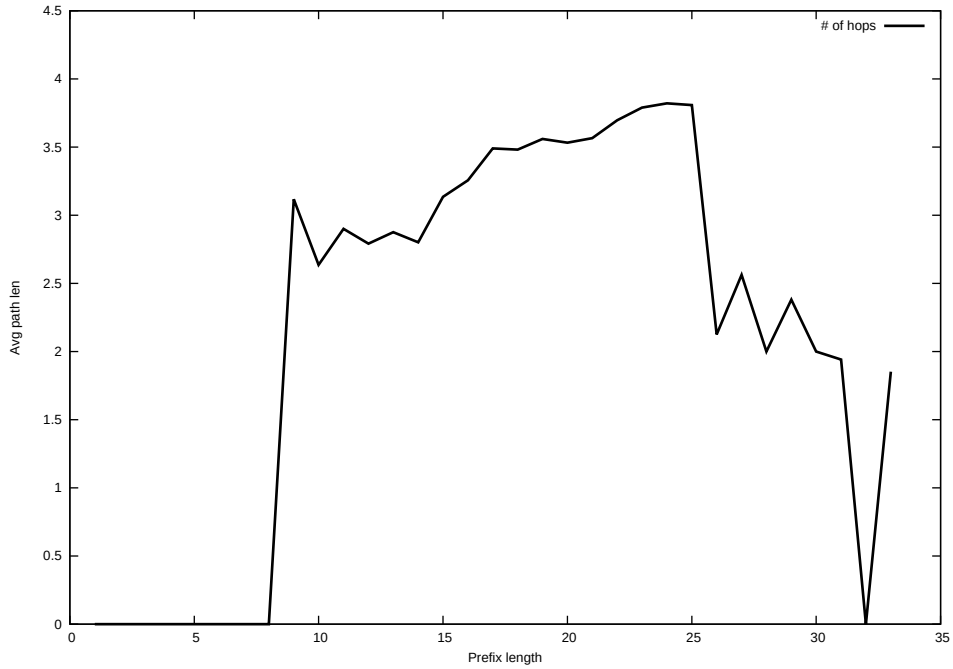
2013-06-13



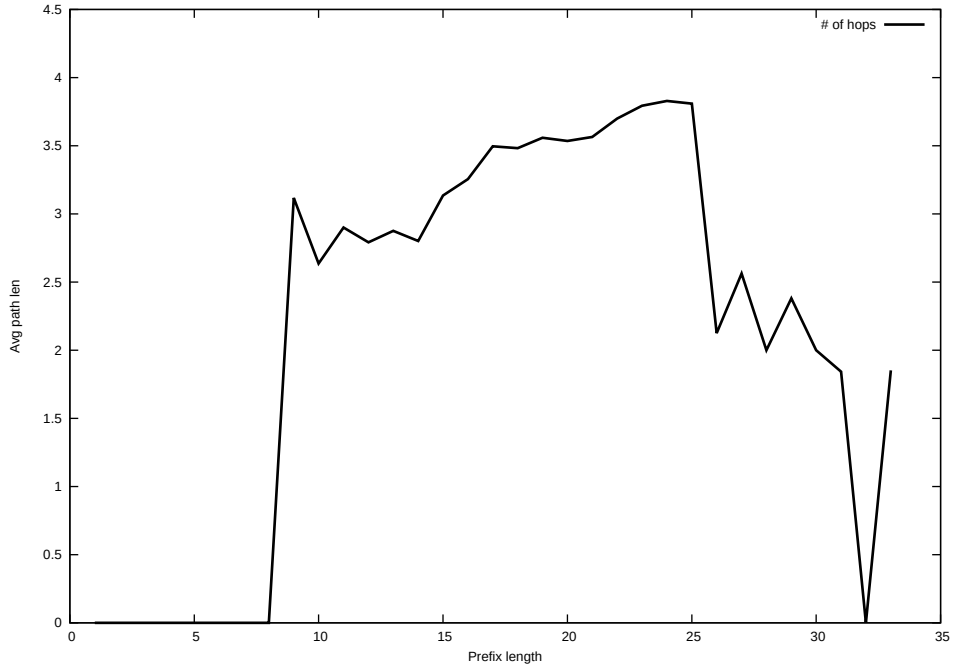
2013-06-14



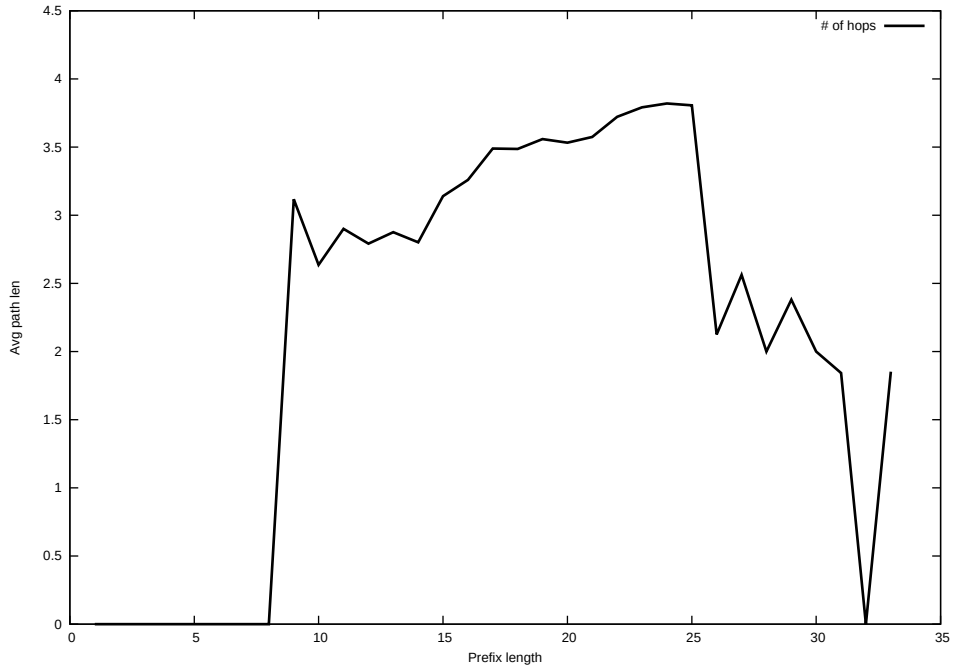
2013-06-15



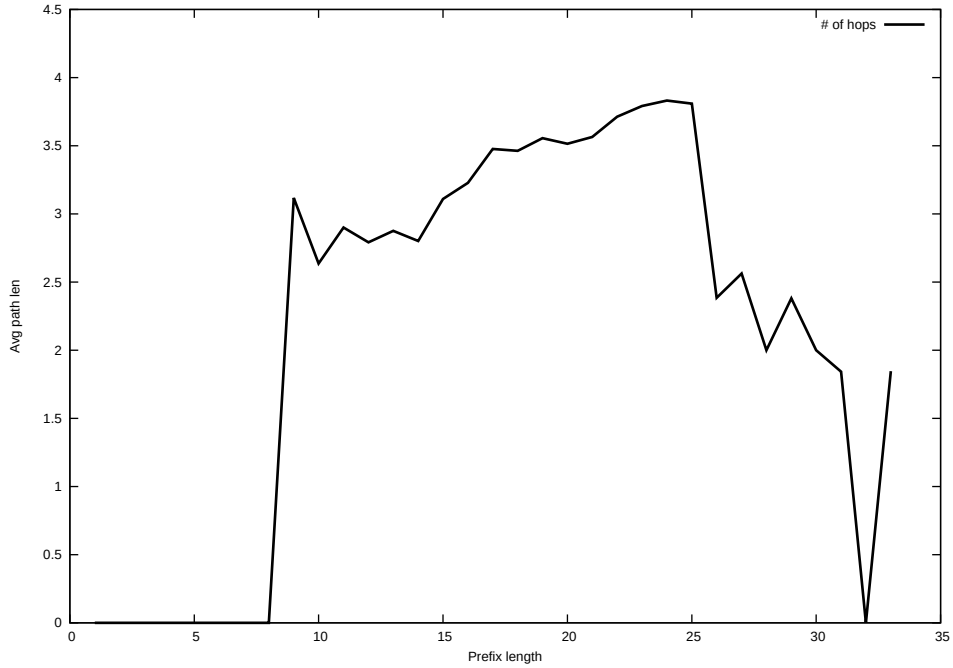
2013-06-16



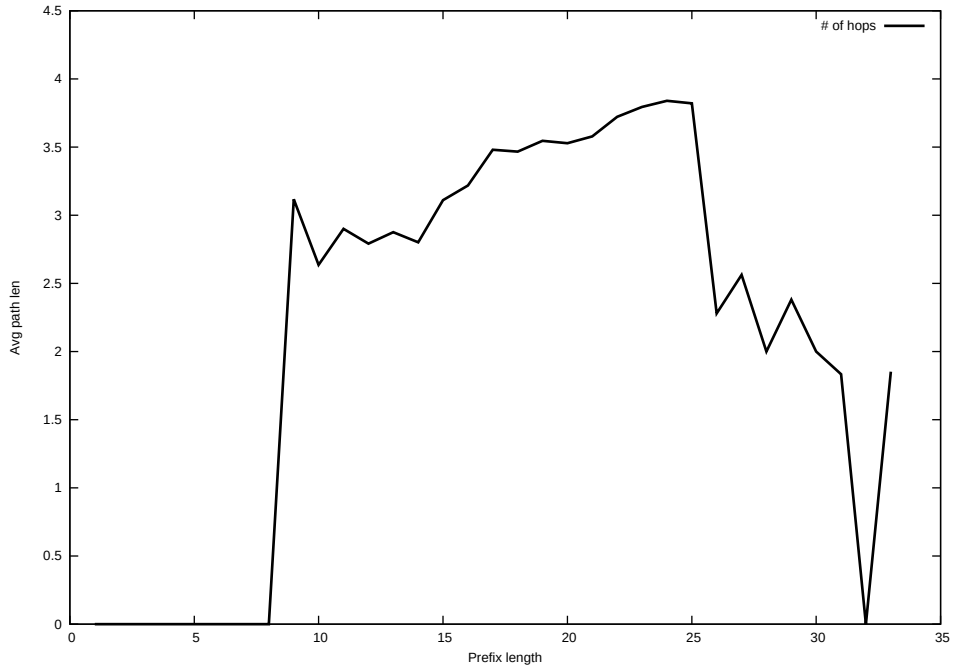
2013-06-17



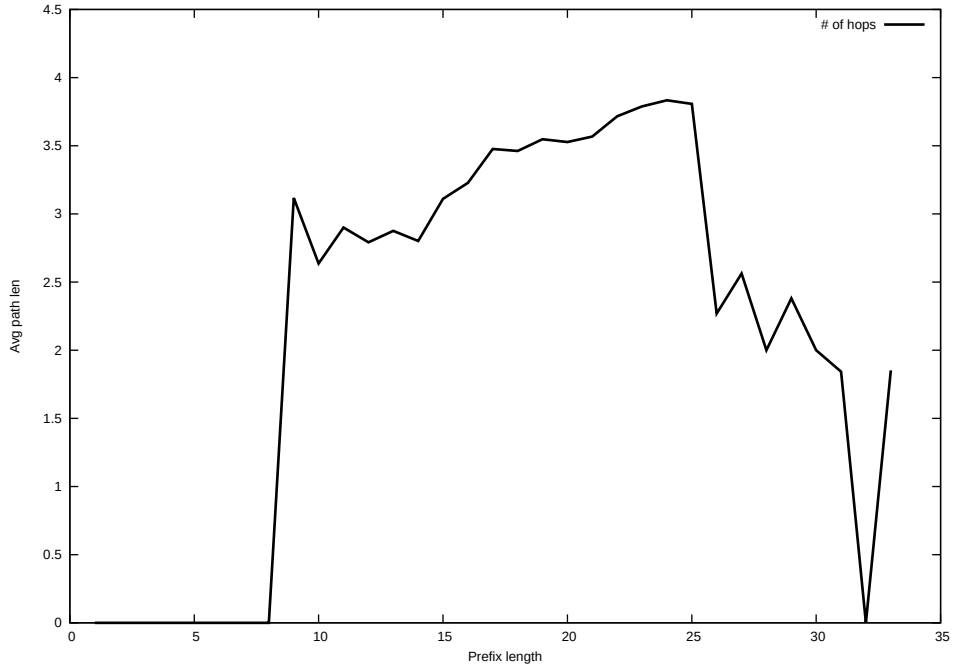
2013-06-18



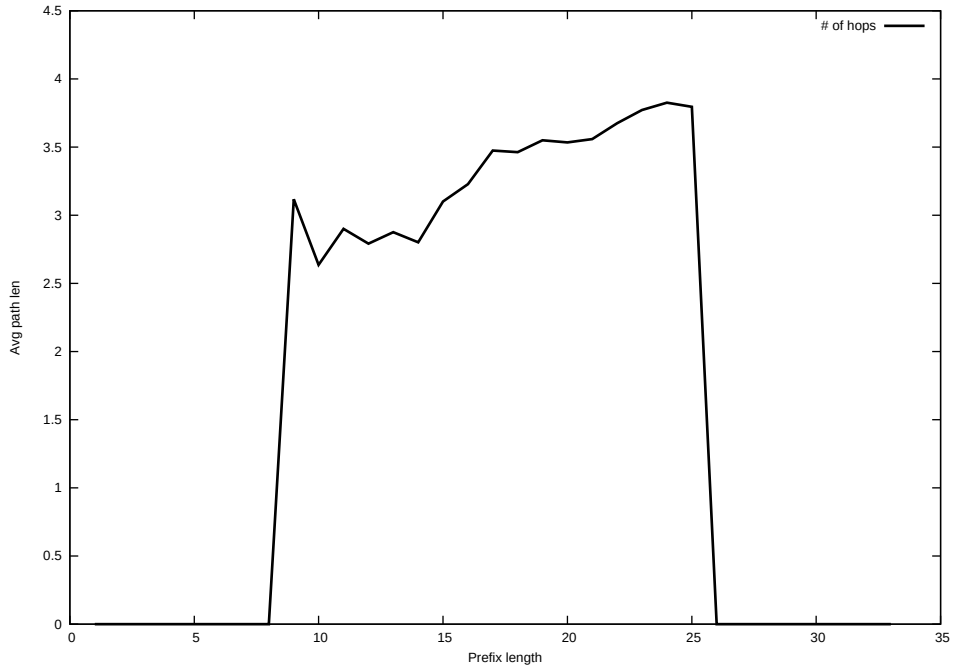
2013-06-19



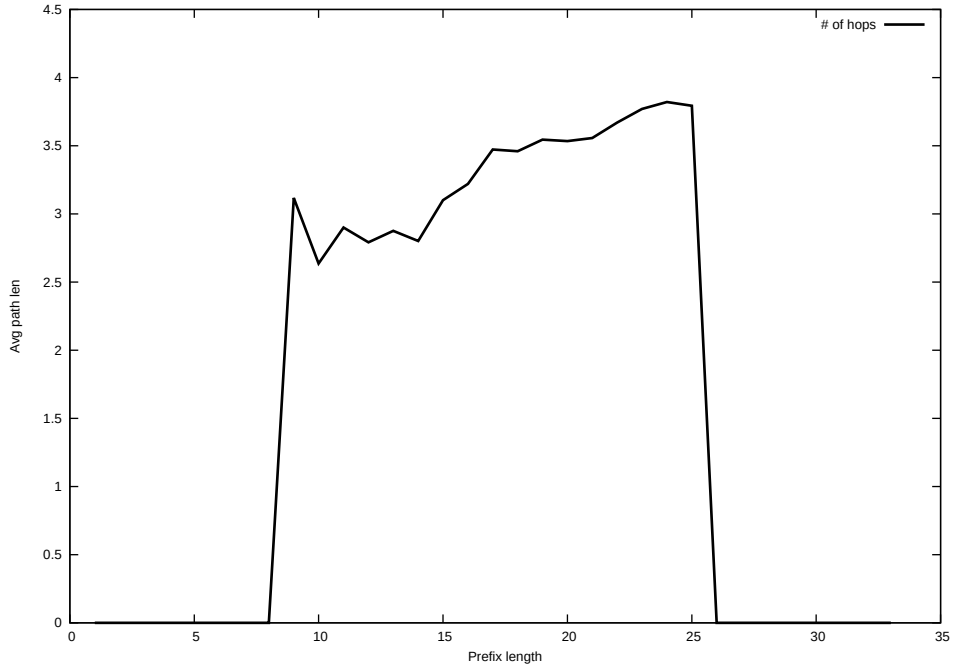
2013-06-20



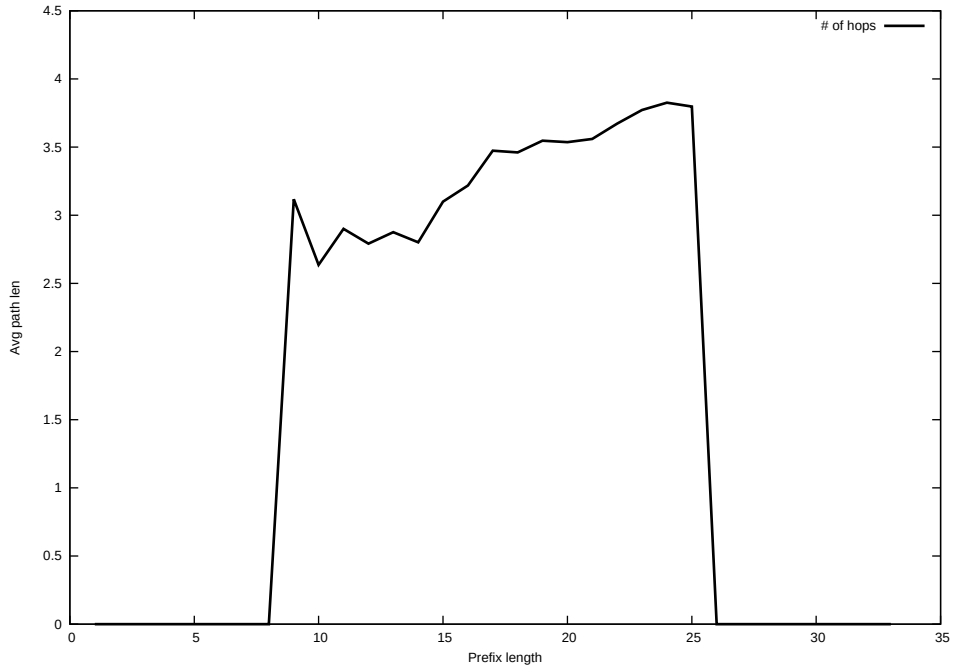
2013-06-21



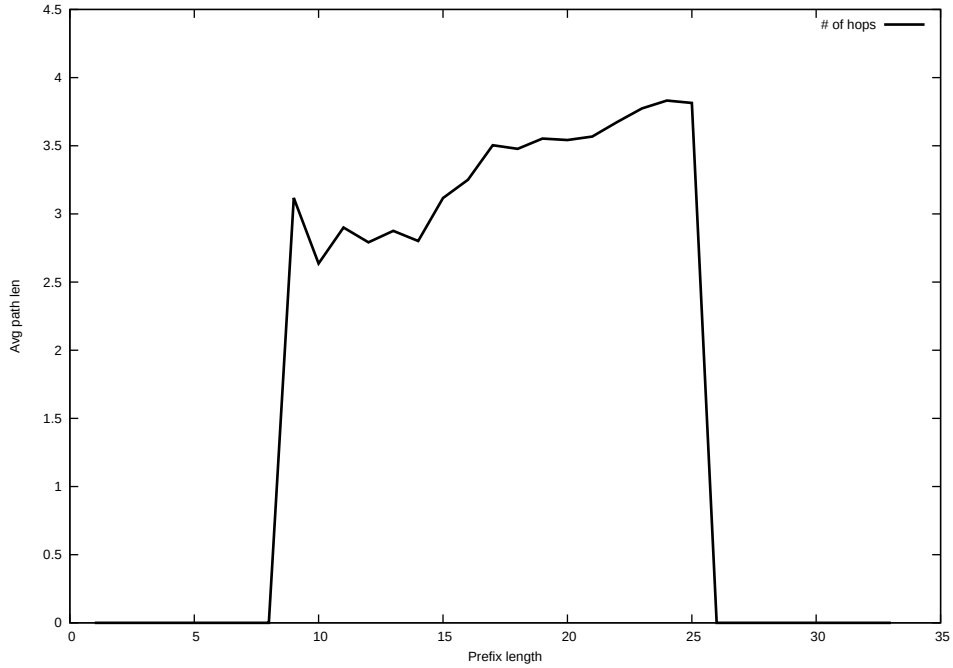
2013-06-22



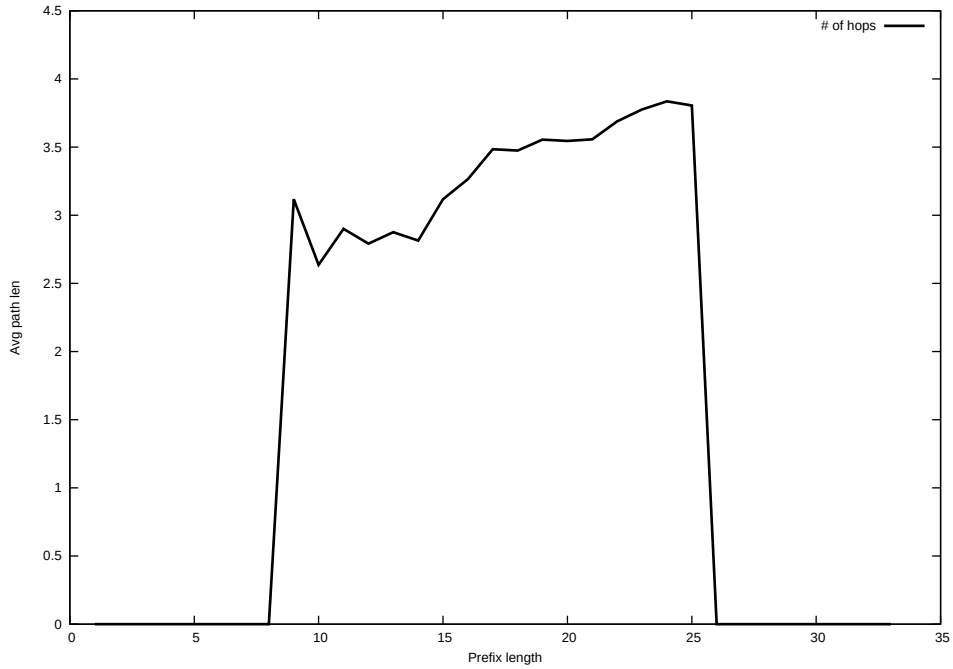
2013-06-23



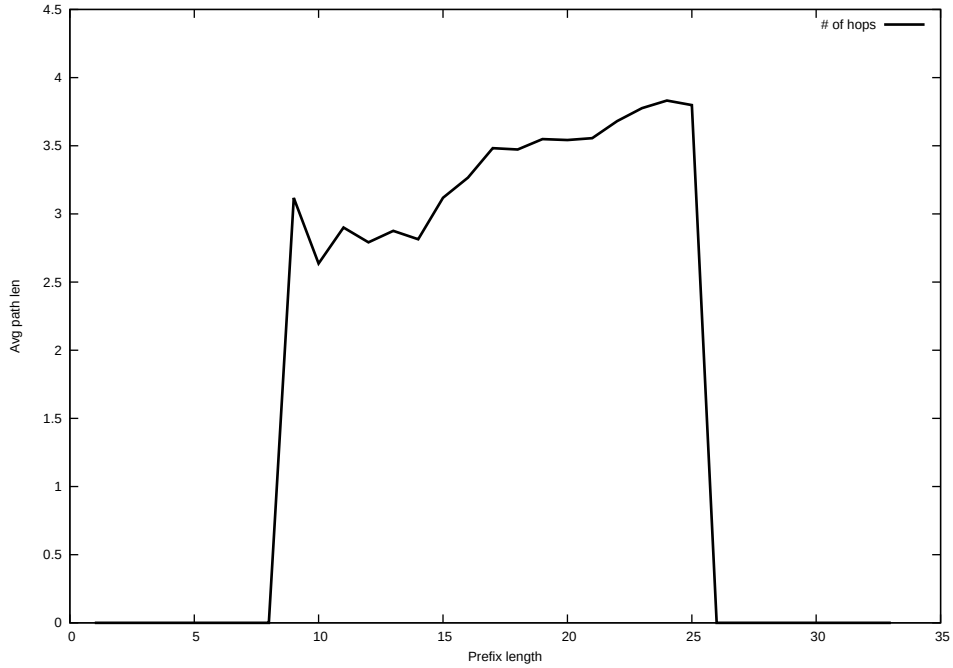
2013-06-24



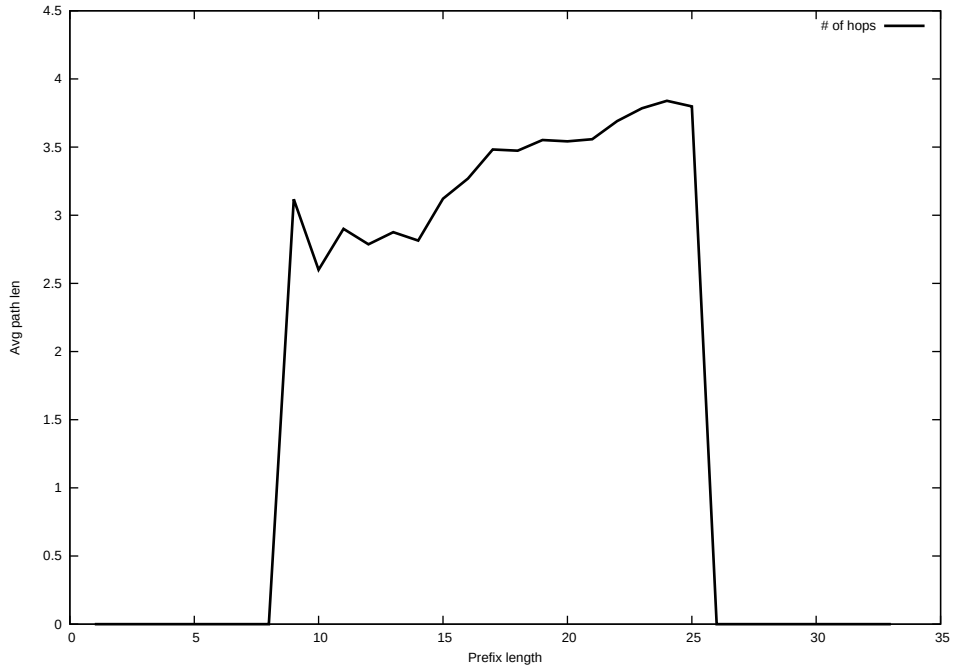
2013-06-25



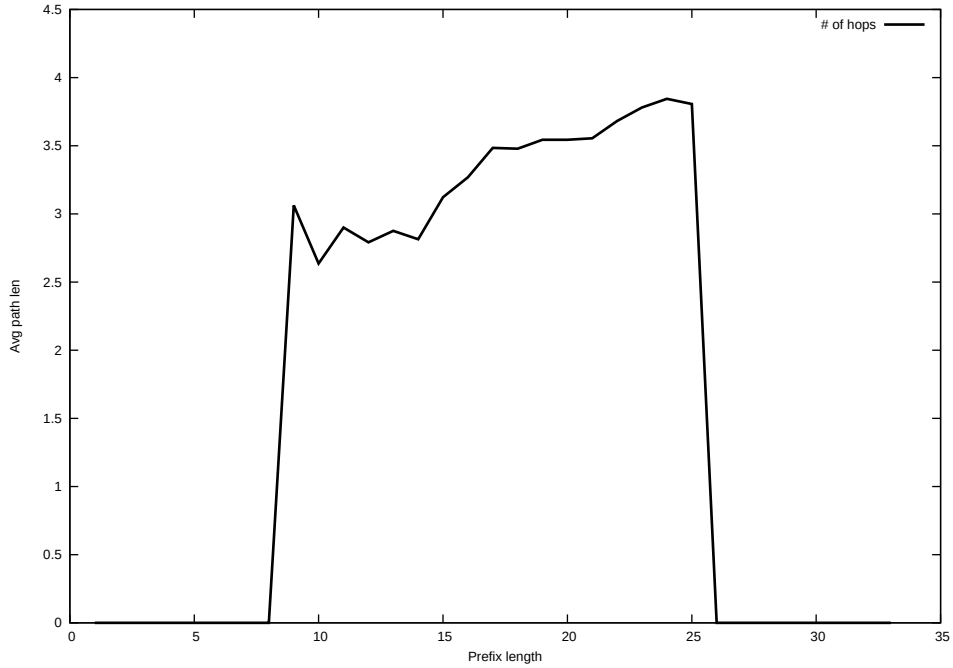
2013-06-26



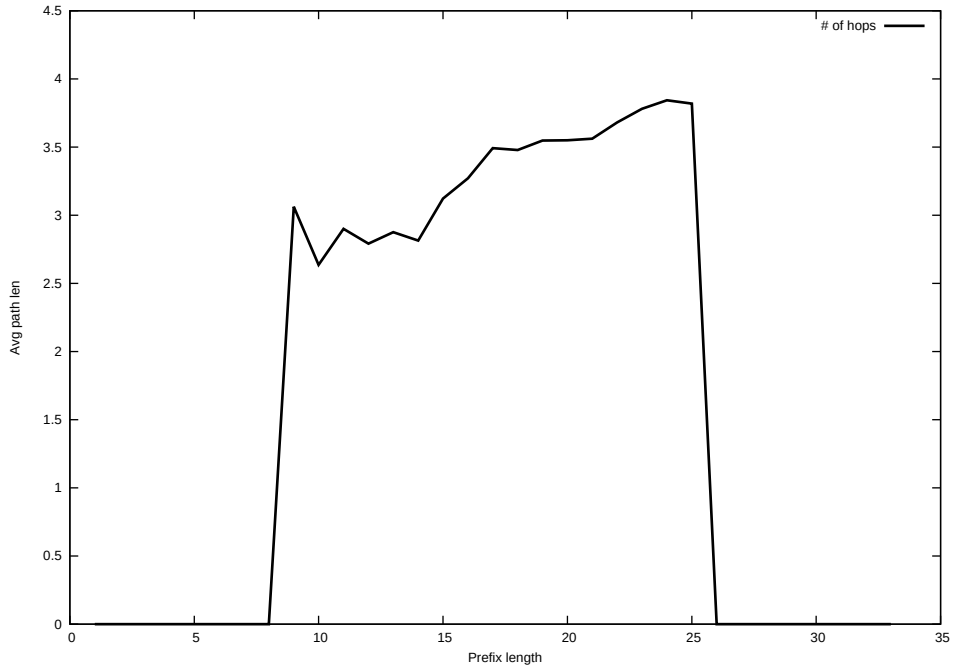
2013-06-27



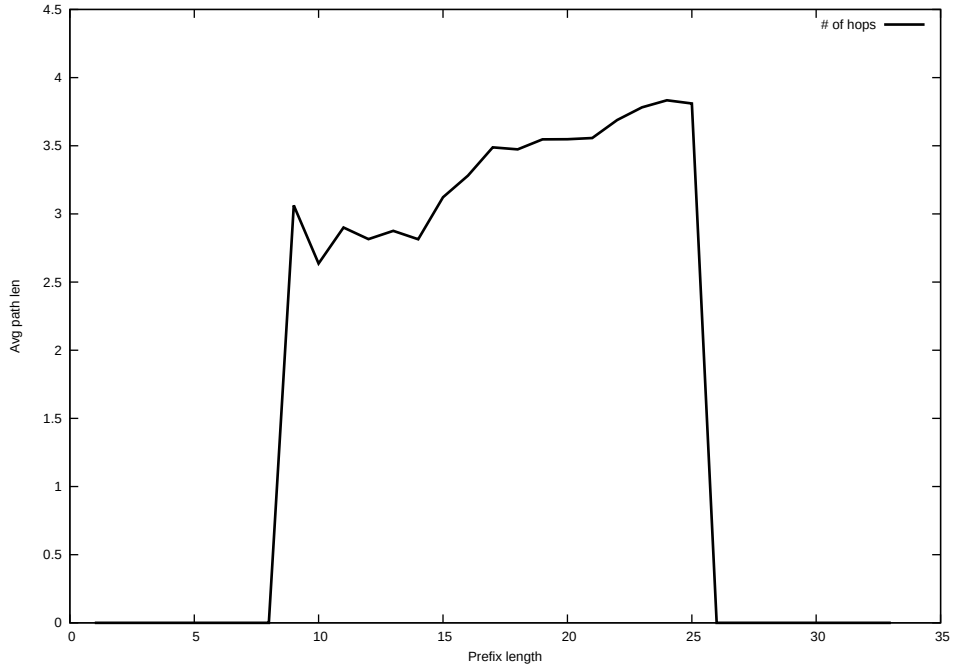
2013-06-28



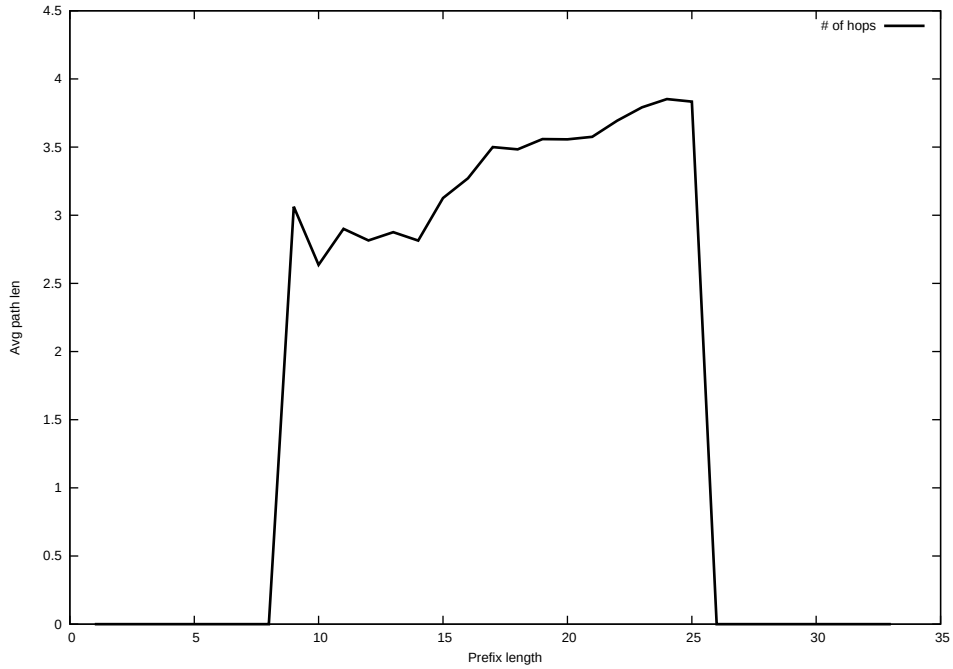
2013-06-29



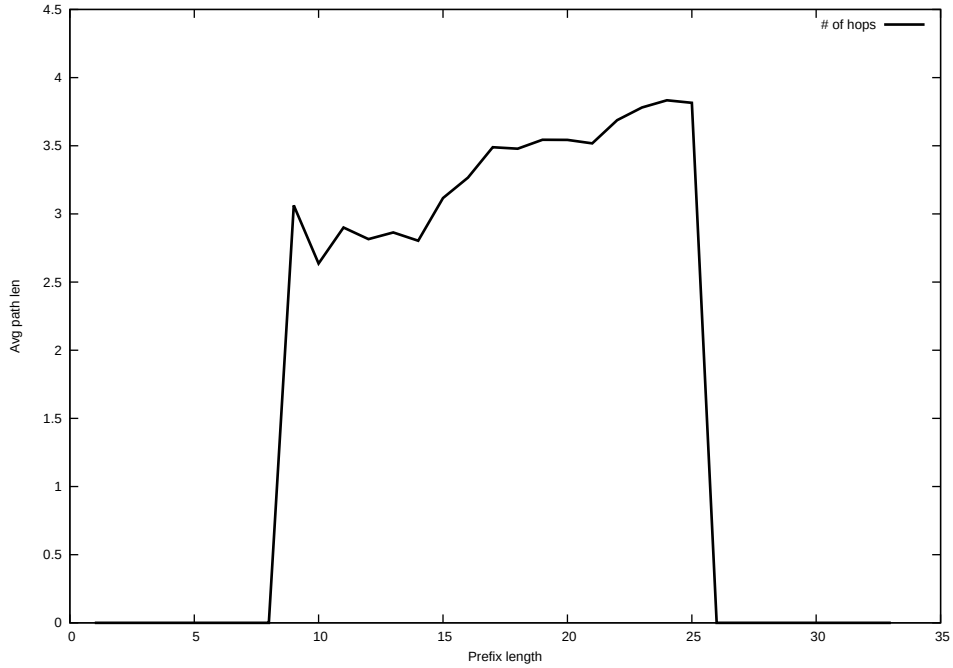
2013-06-30



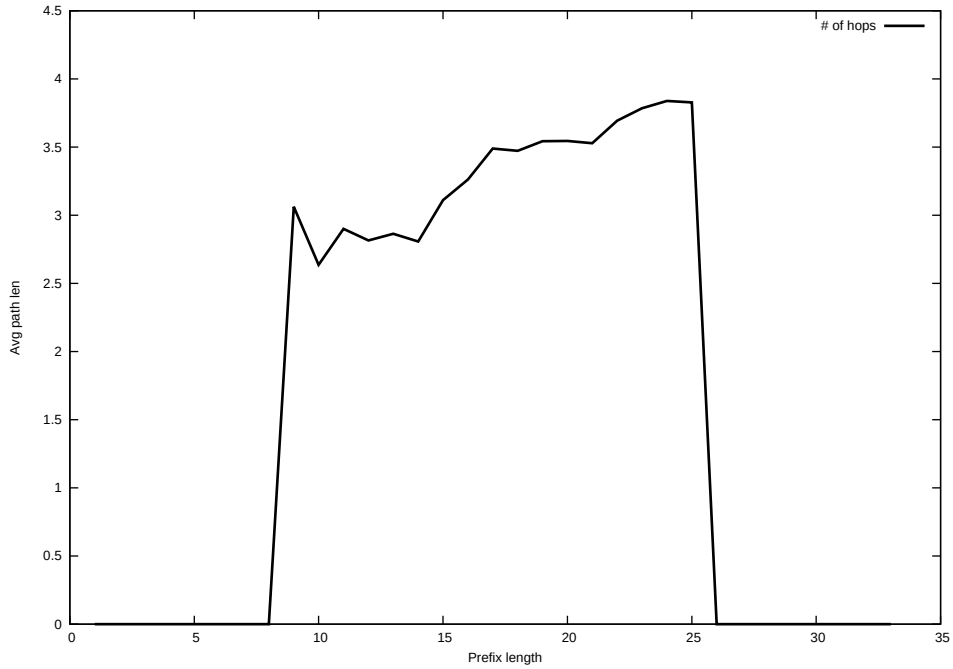
2013-07-01



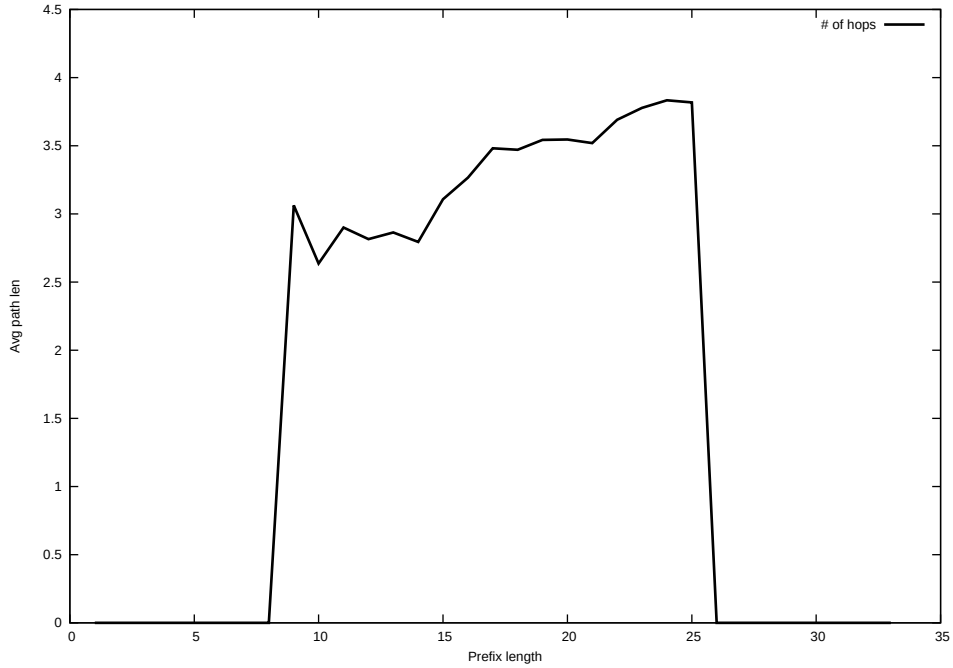
2013-07-02



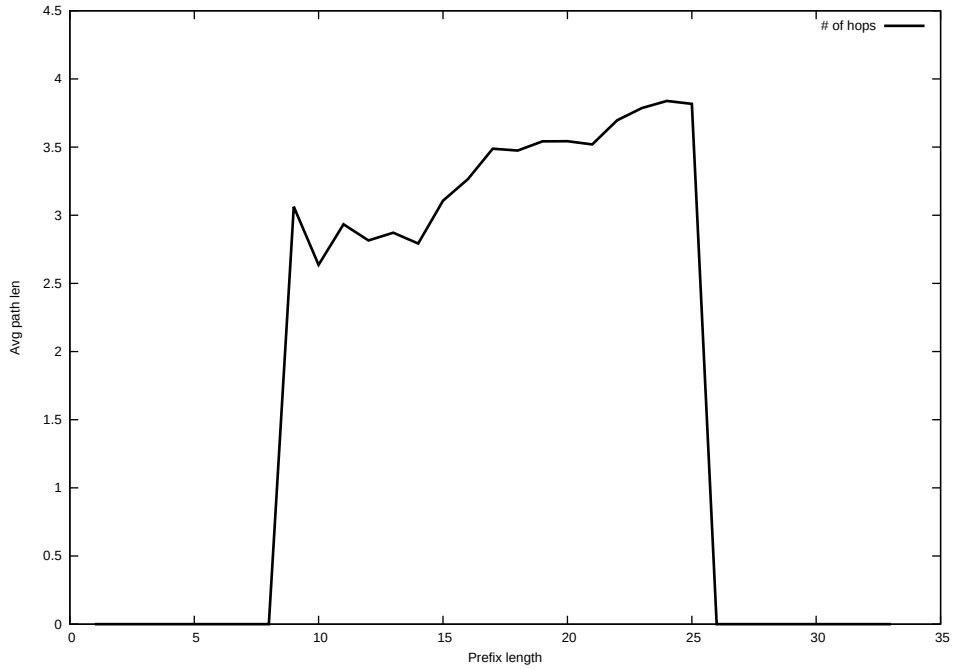
2013-07-03



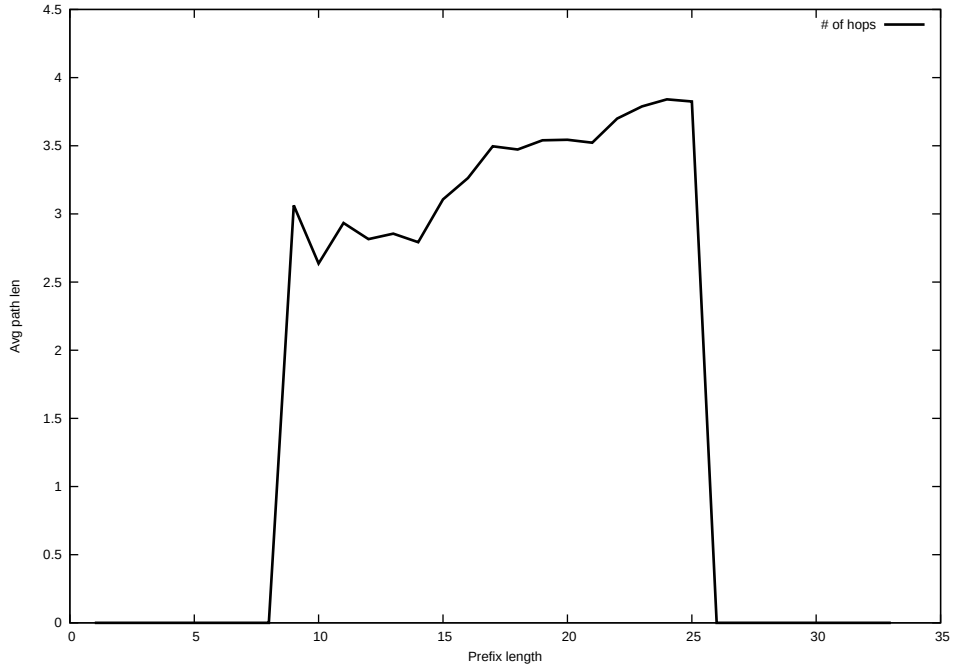
2013-07-04



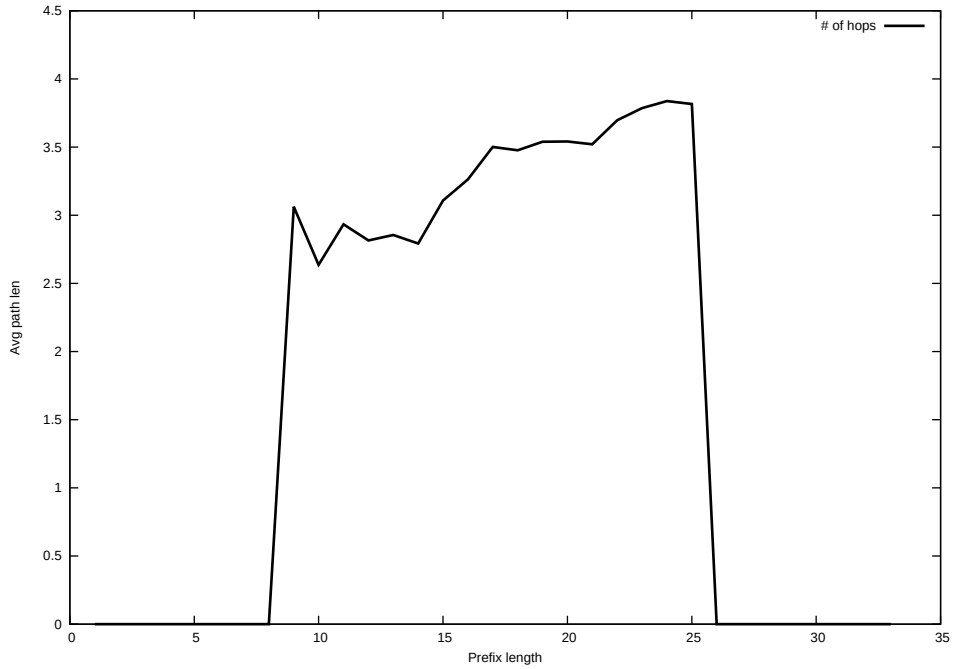
2013-07-05



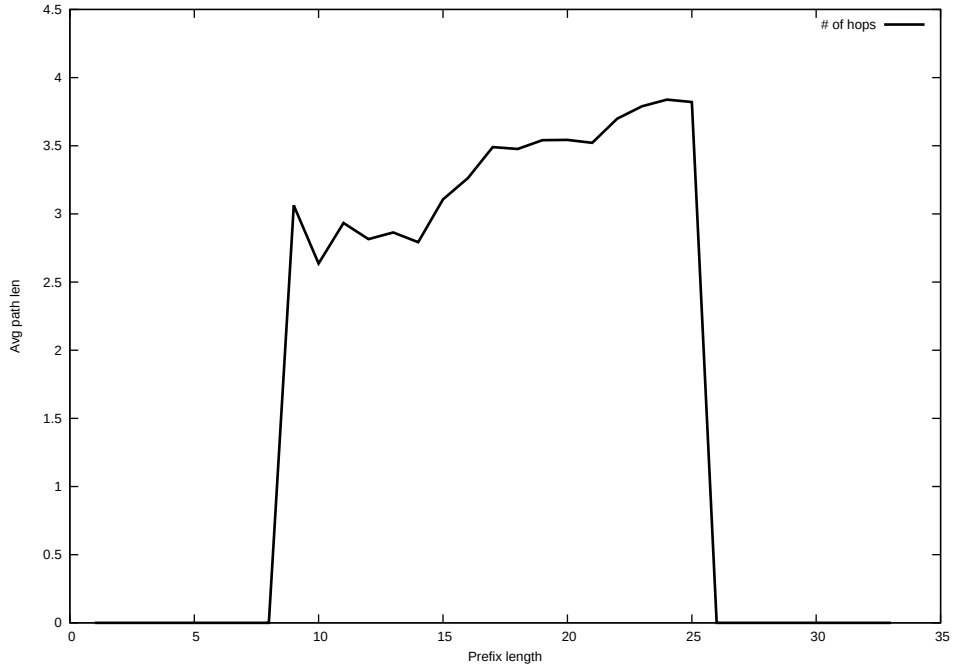
2013-07-06



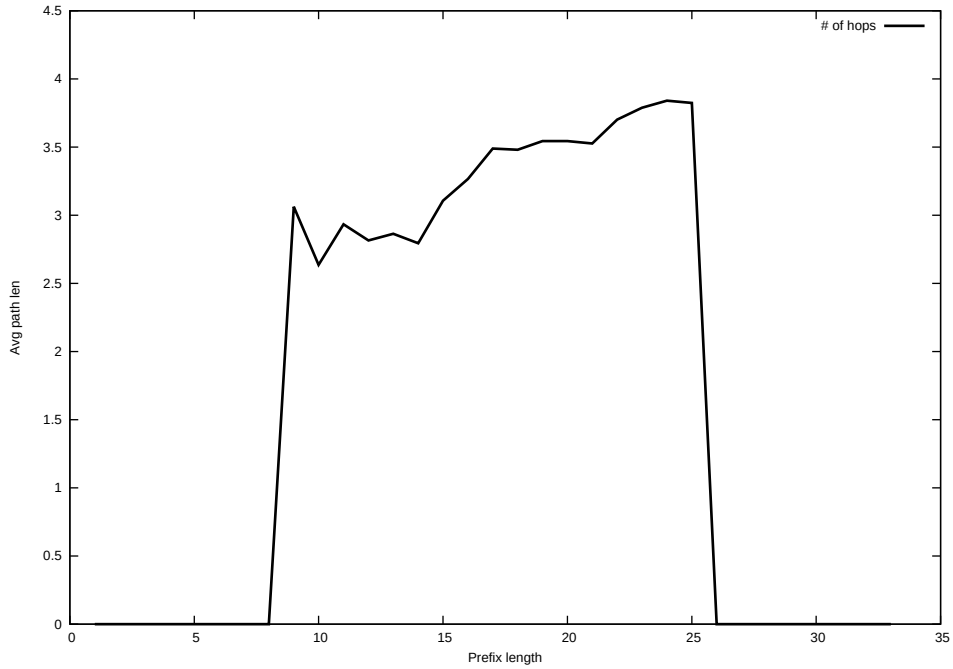
2013-07-07



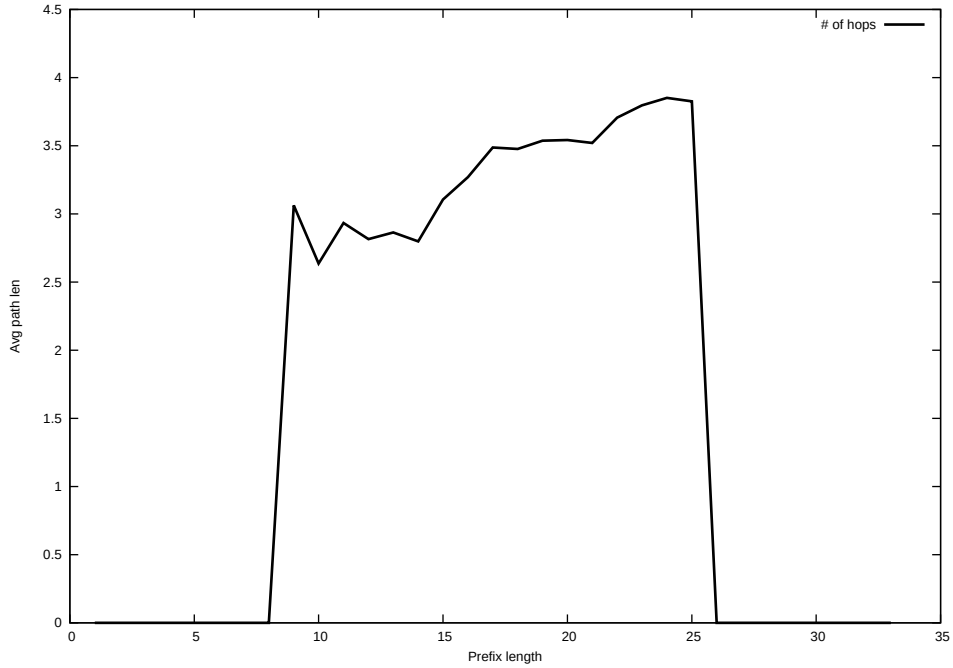
2013-07-08



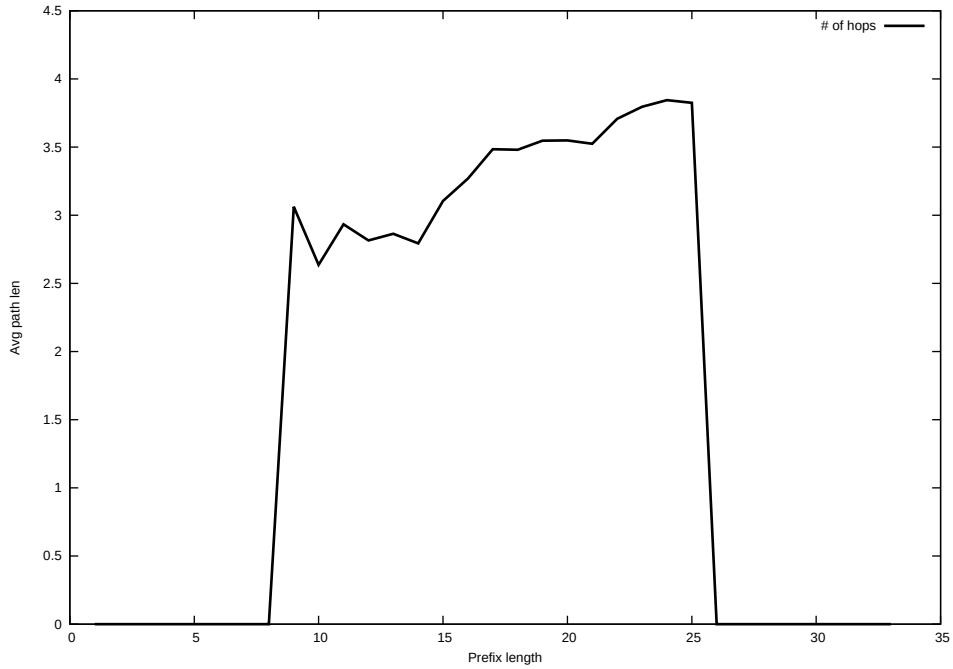
2013-07-09



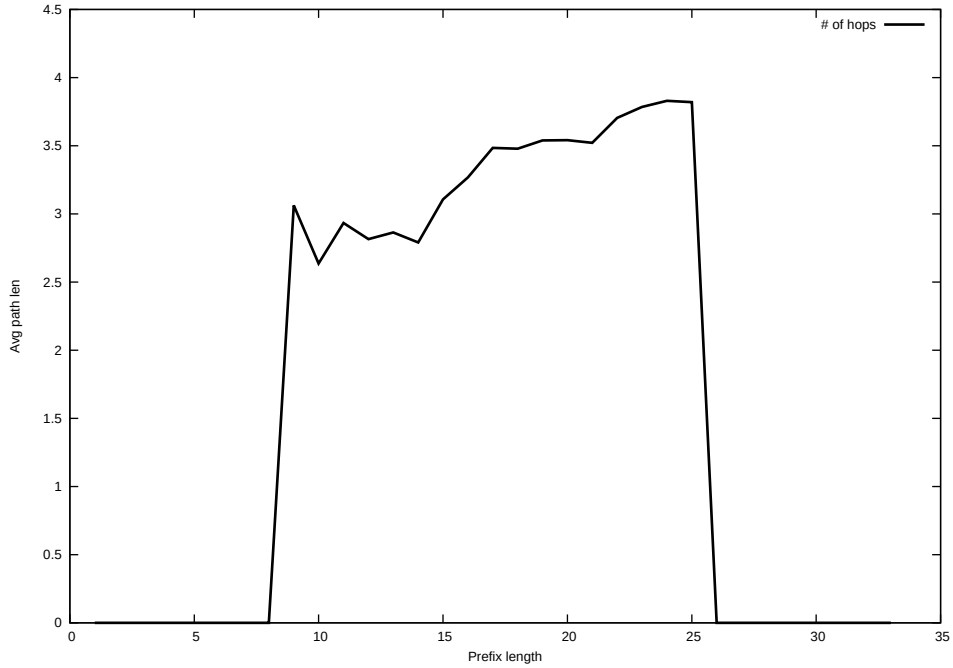
2013-07-10



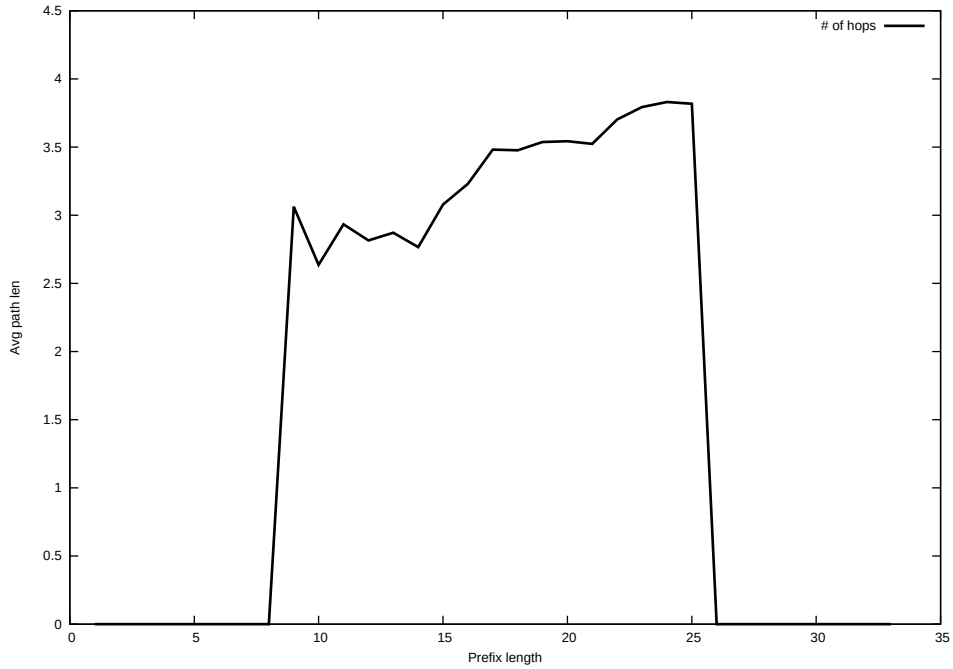
2013-07-11



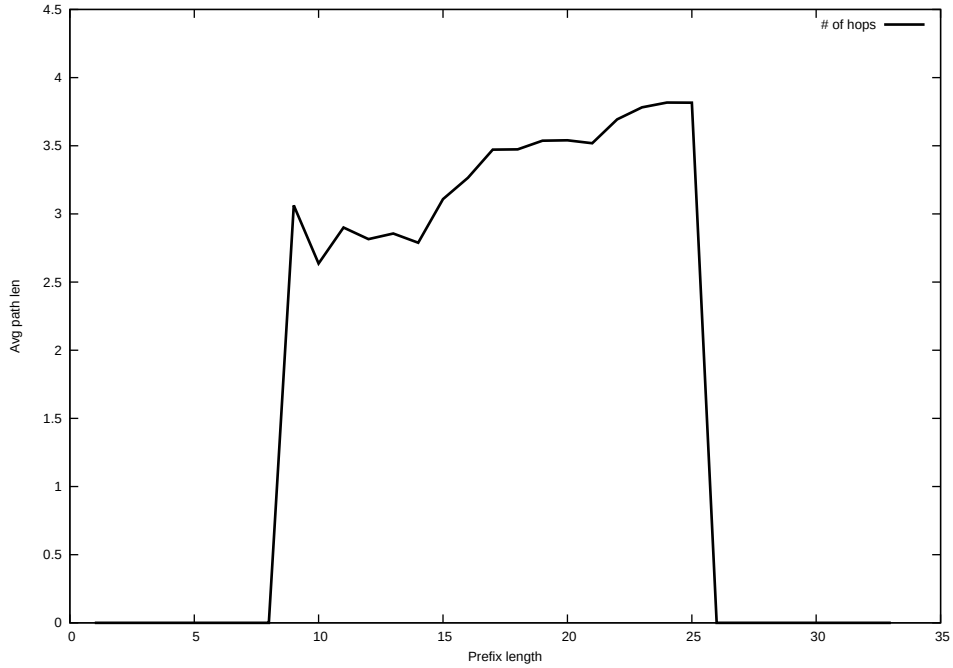
2013-07-12



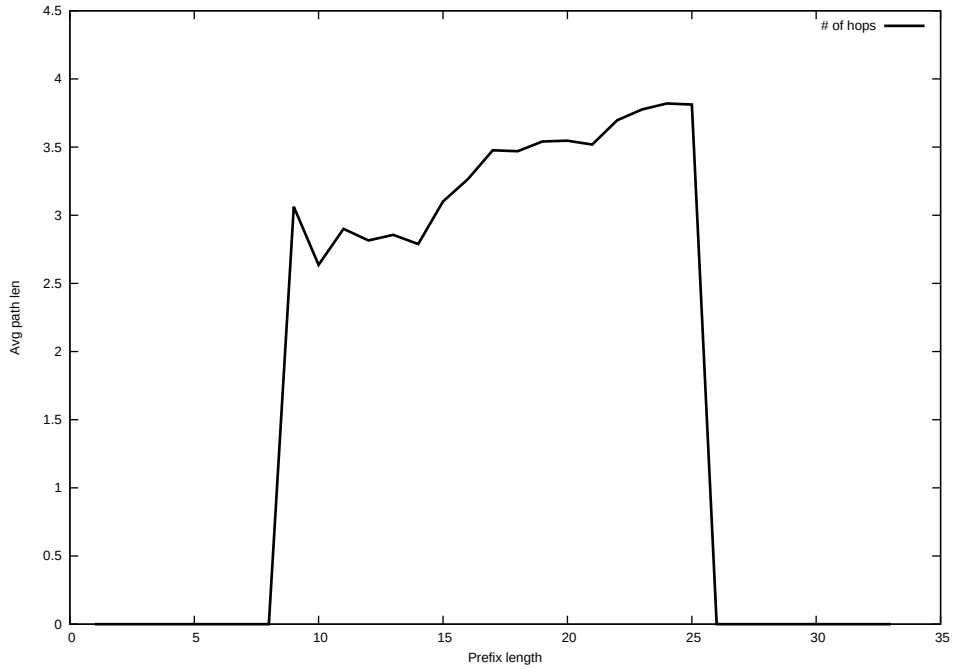
2013-07-13



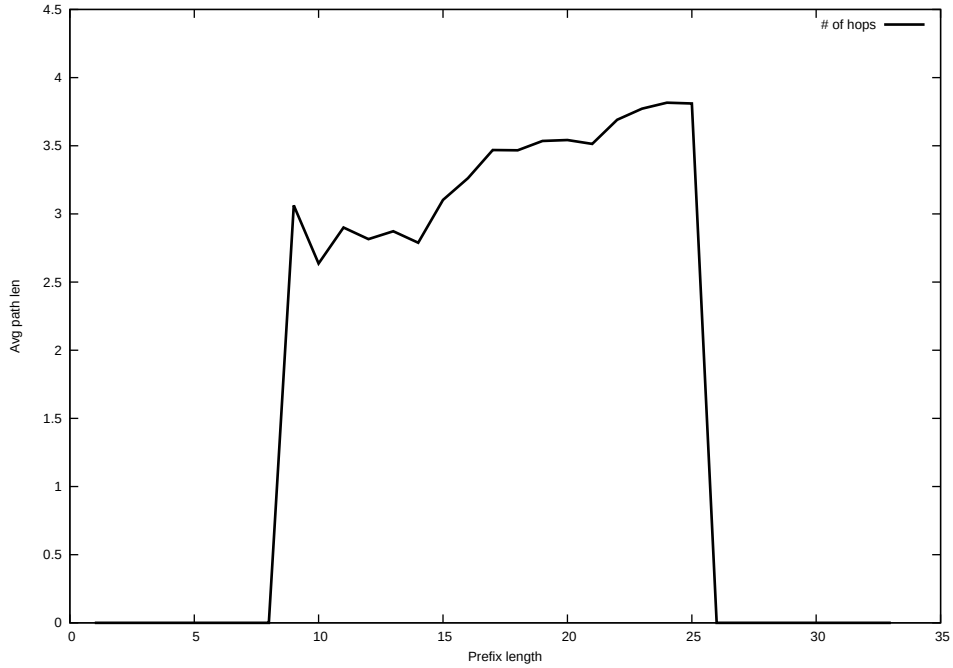
2013-07-14



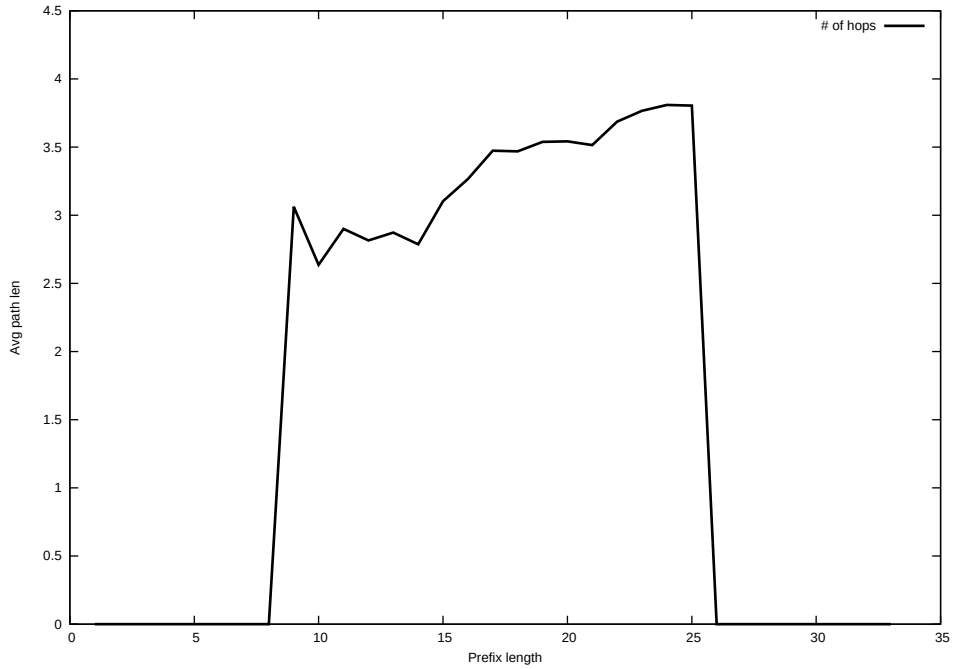
2013-07-15



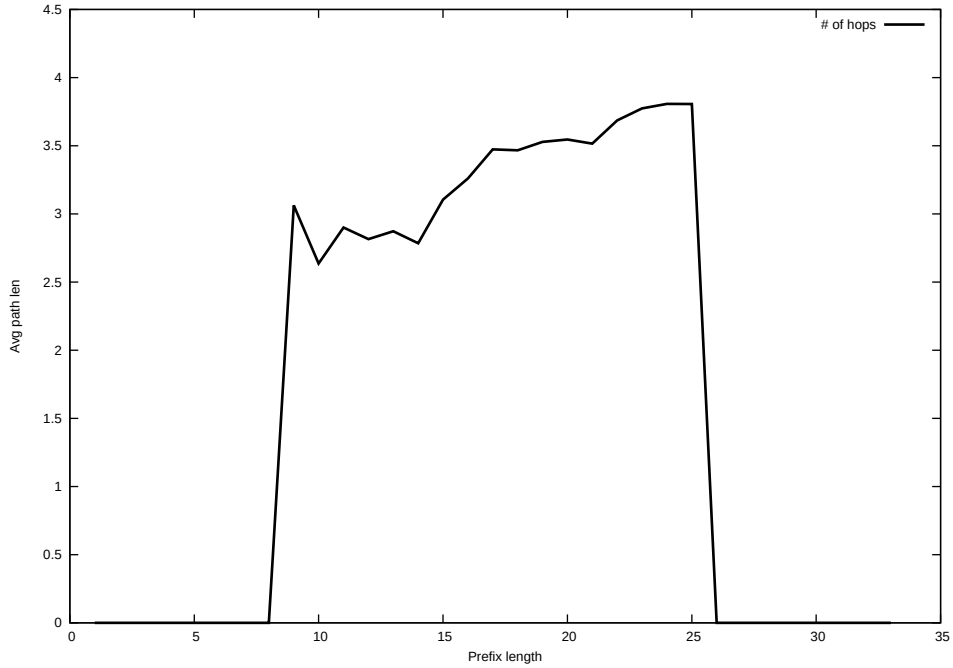
2013-07-16



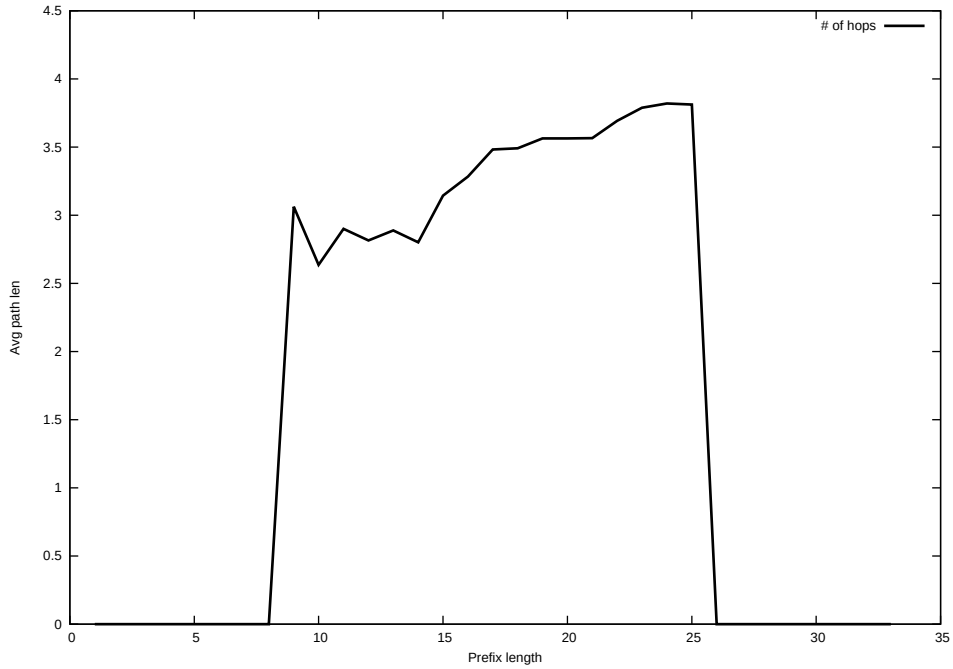
2013-07-17



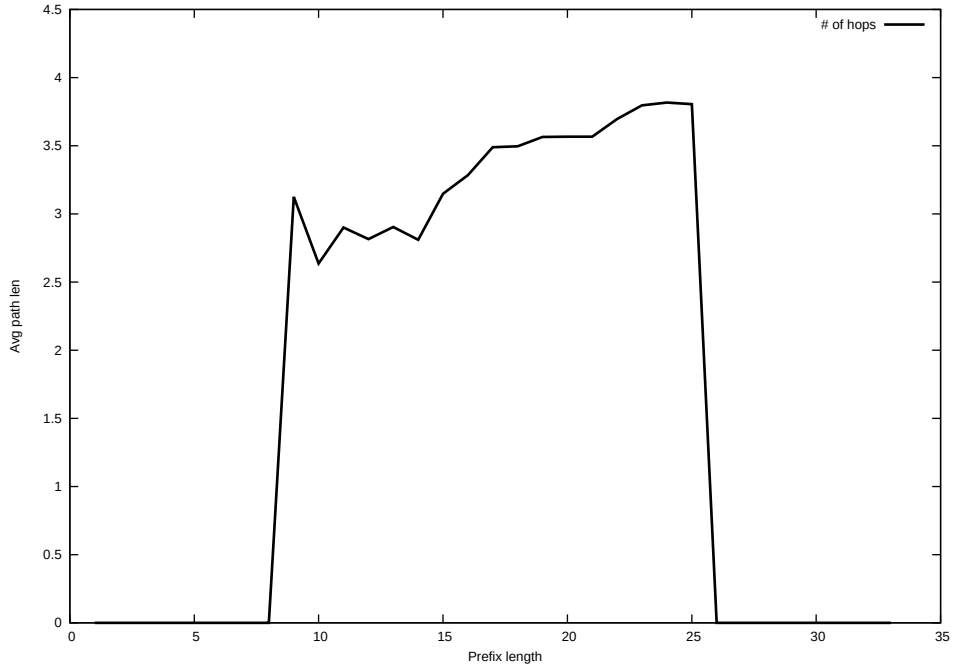
2013-07-18



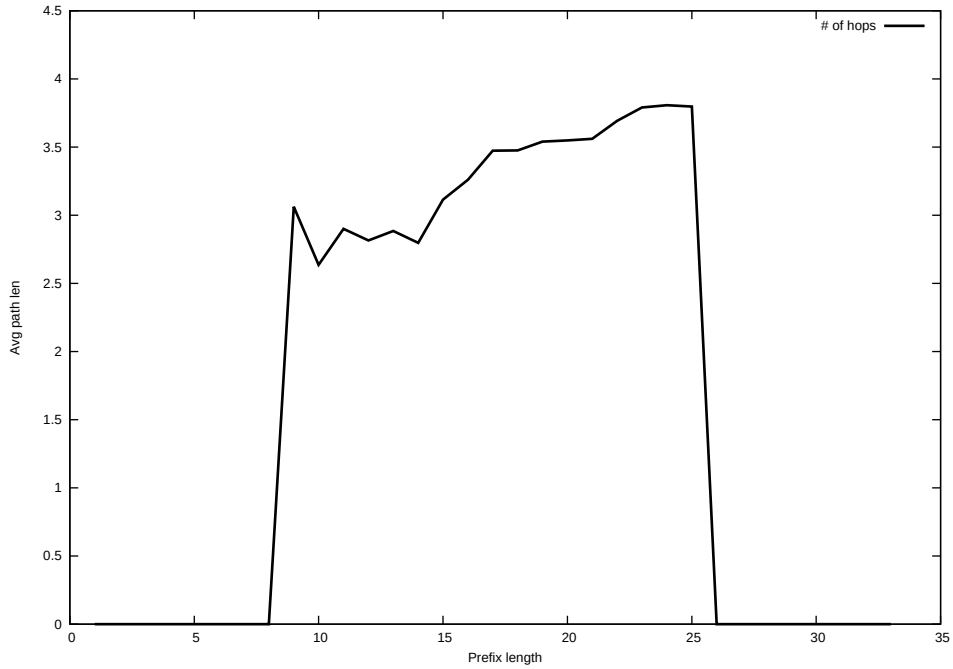
2013-07-19



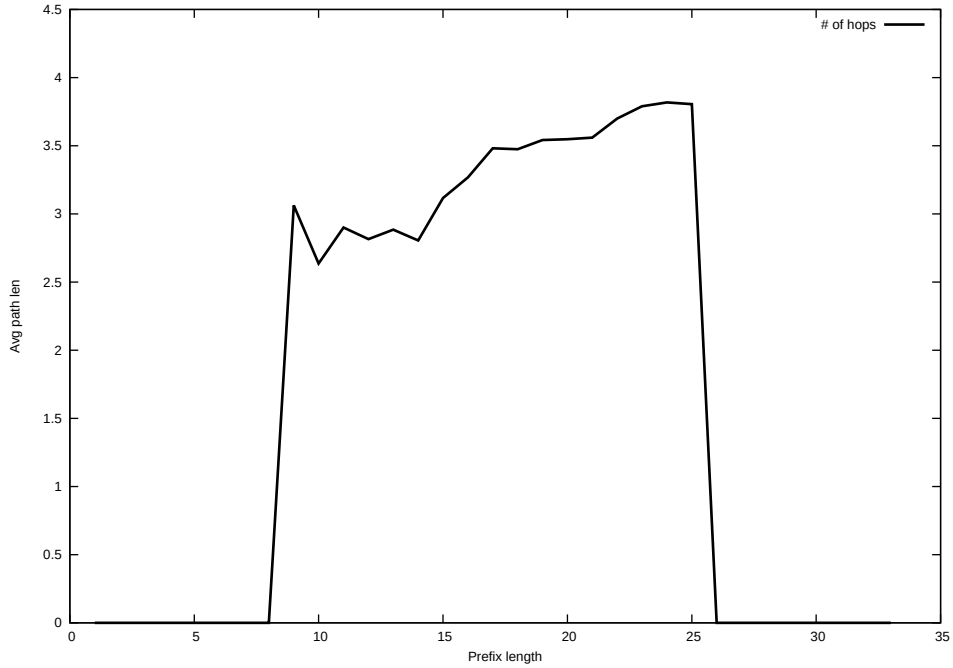
2013-07-20



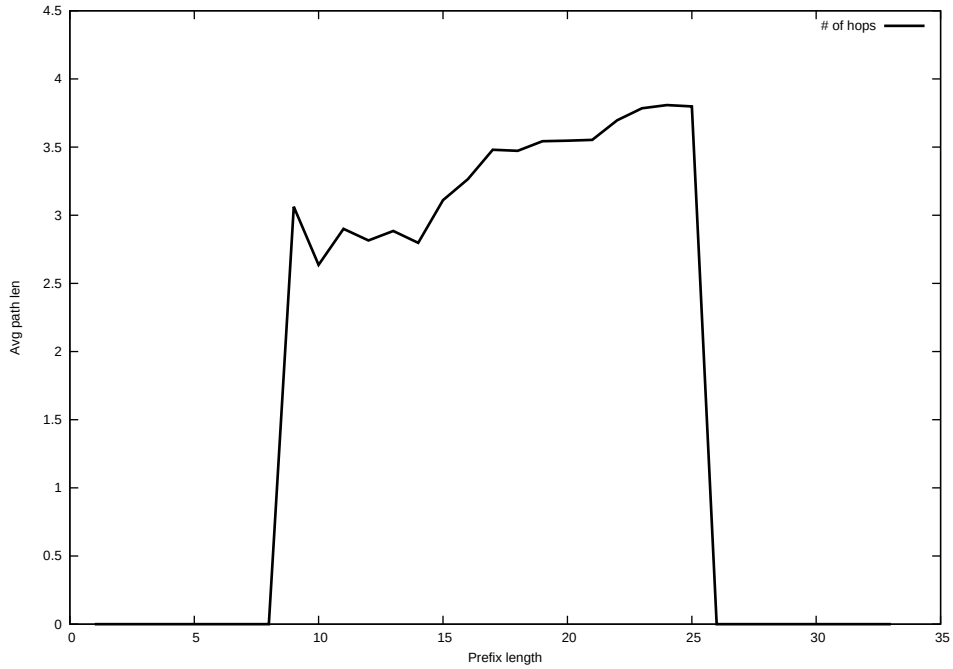
2013-07-21



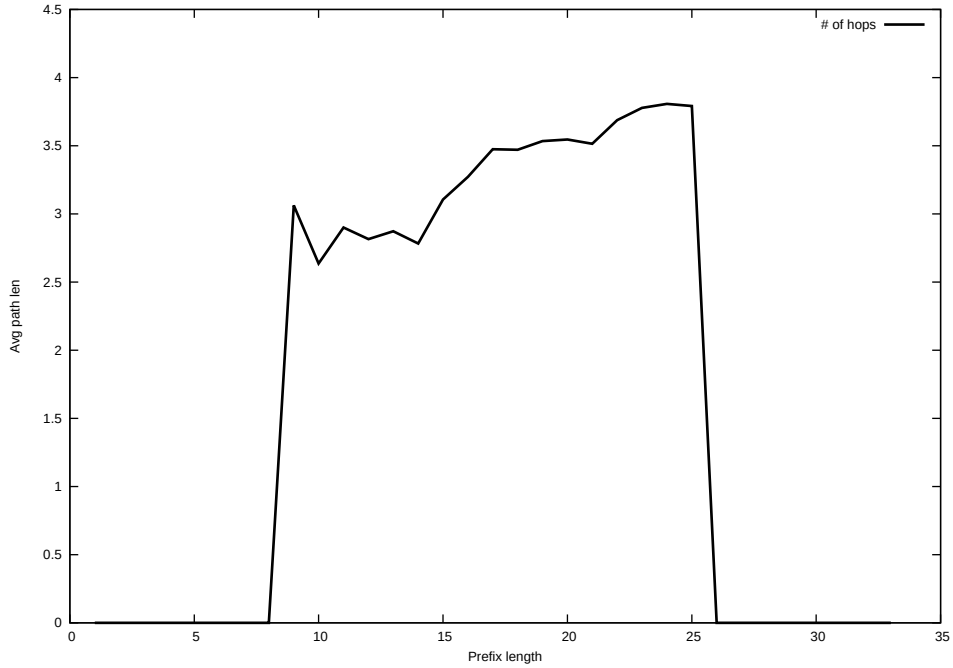
2013-07-22



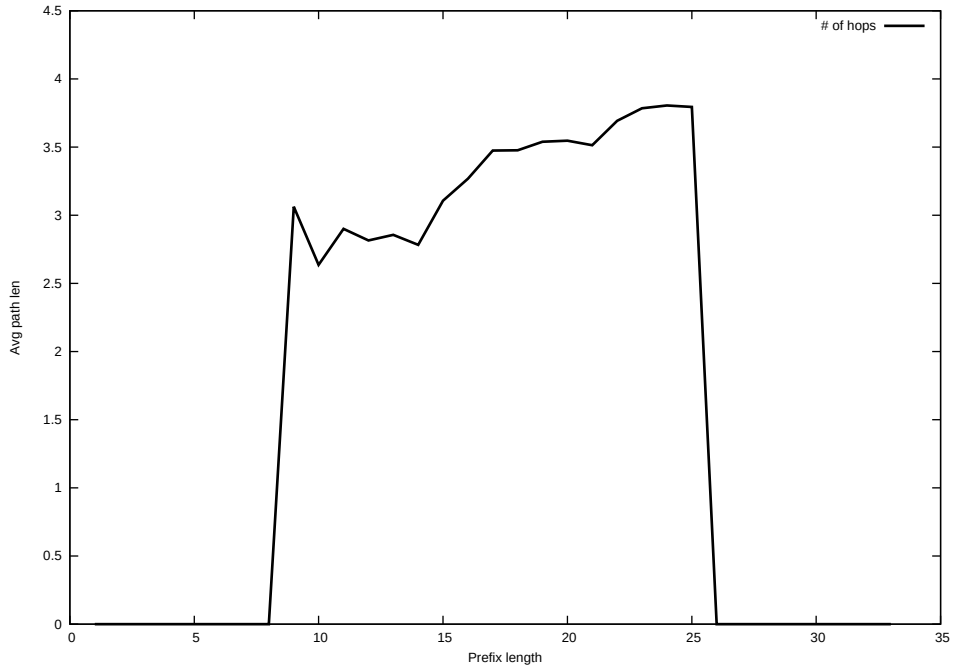
2013-07-23



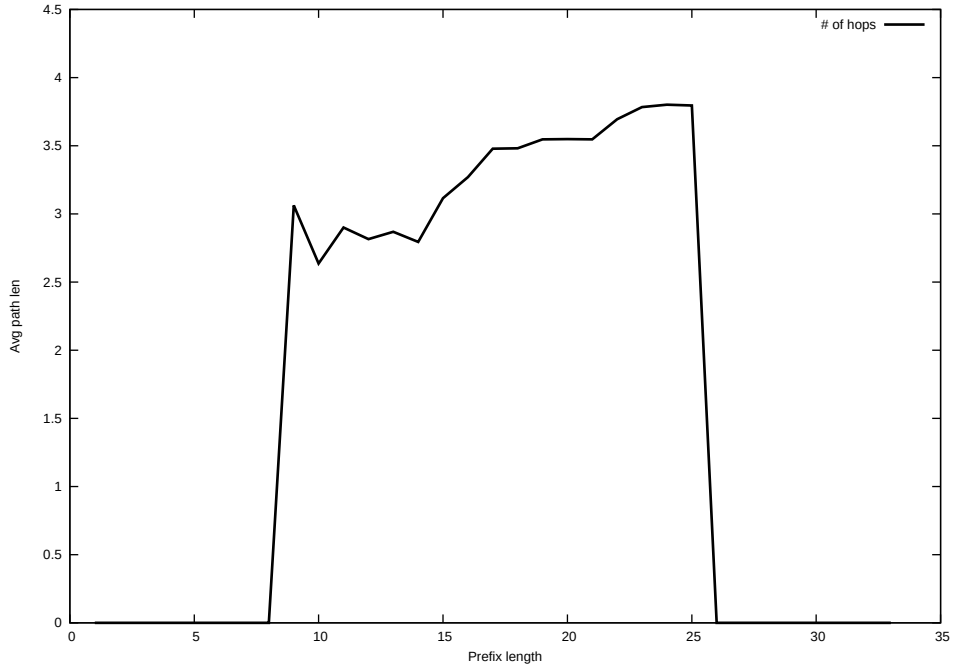
2013-07-24



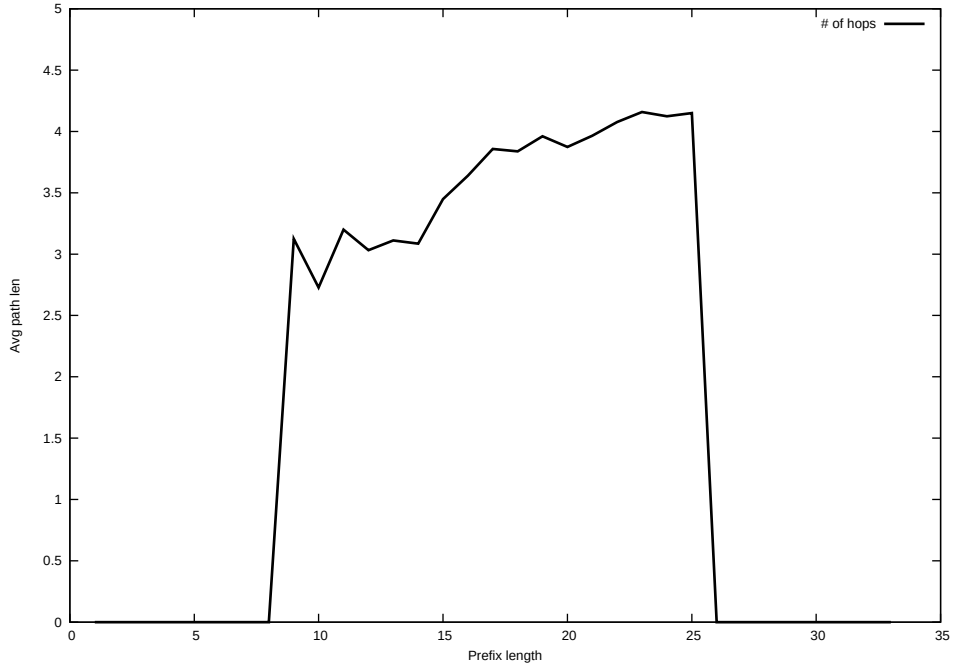
2013-07-25



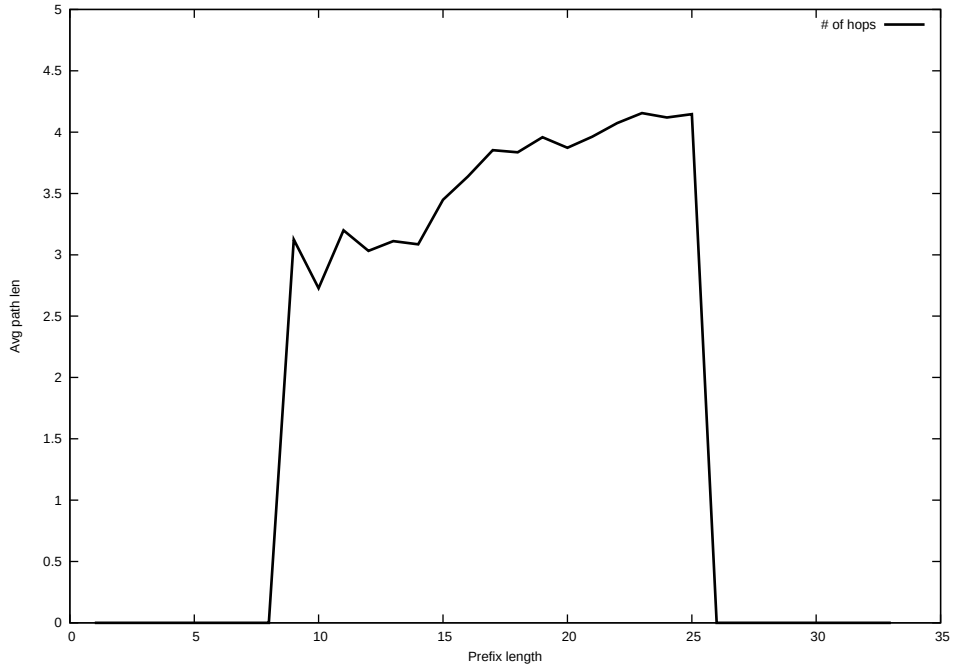
2013-07-26



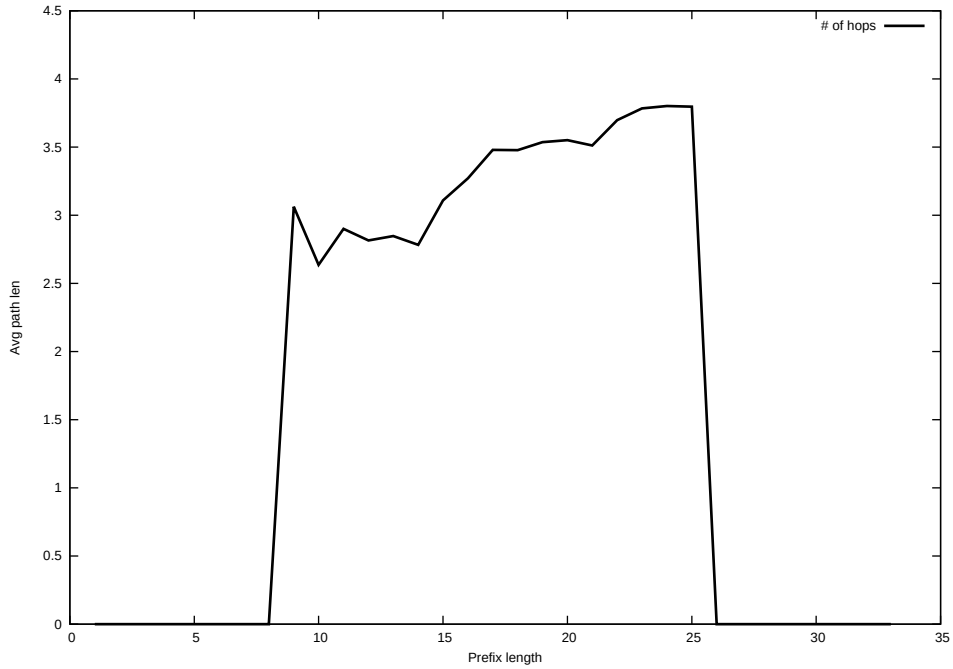
2013-07-27



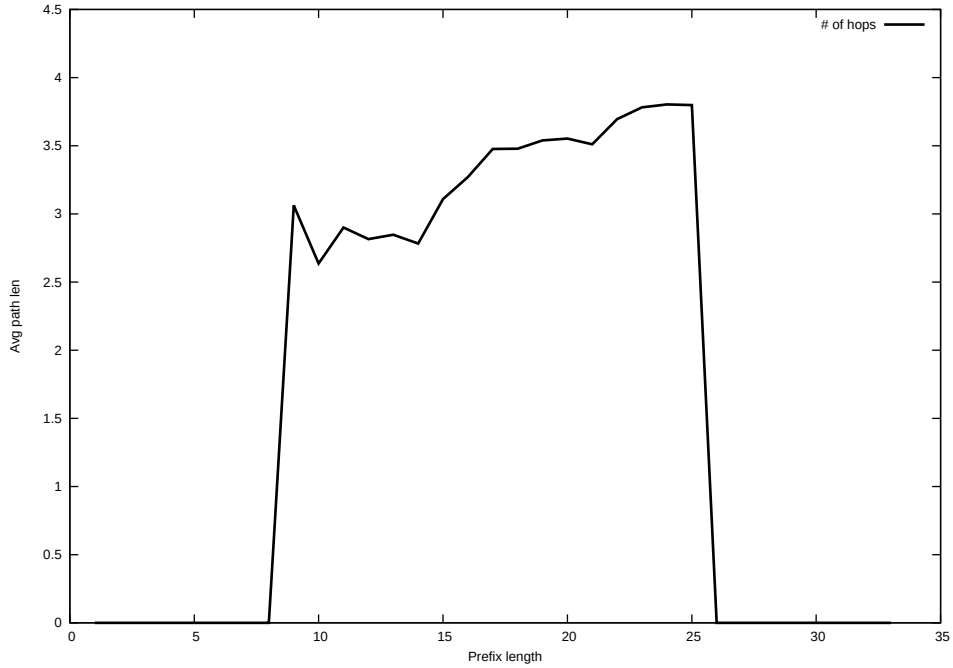
2013-07-28



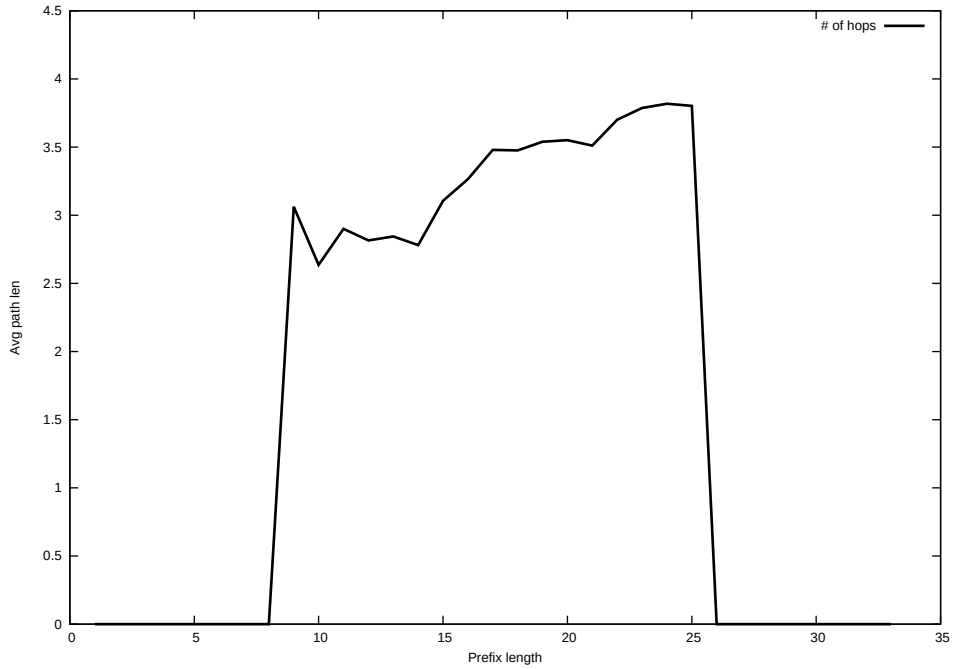
2013-07-29



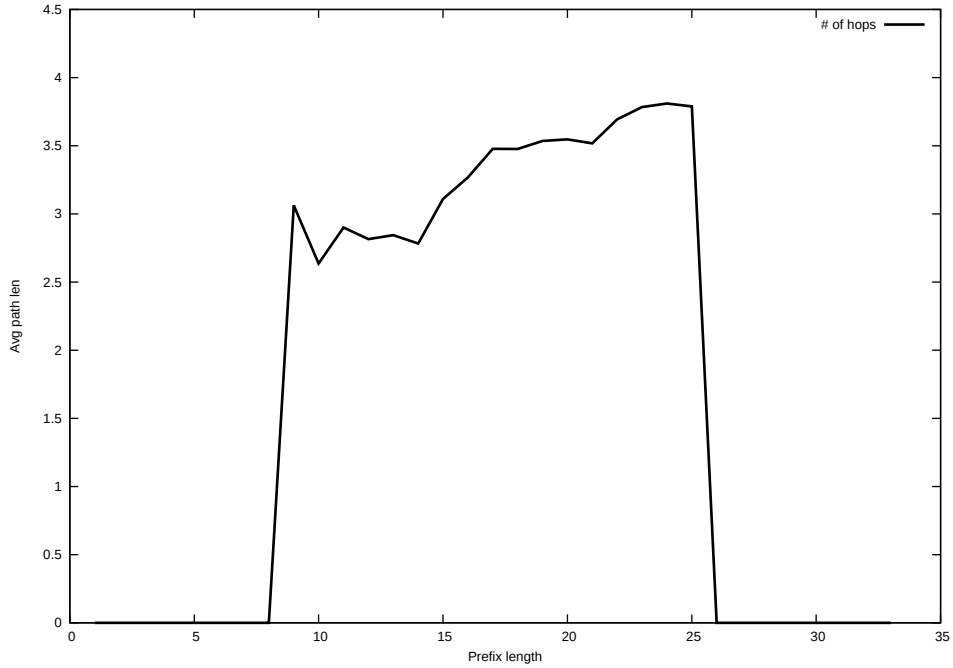
2013-07-30



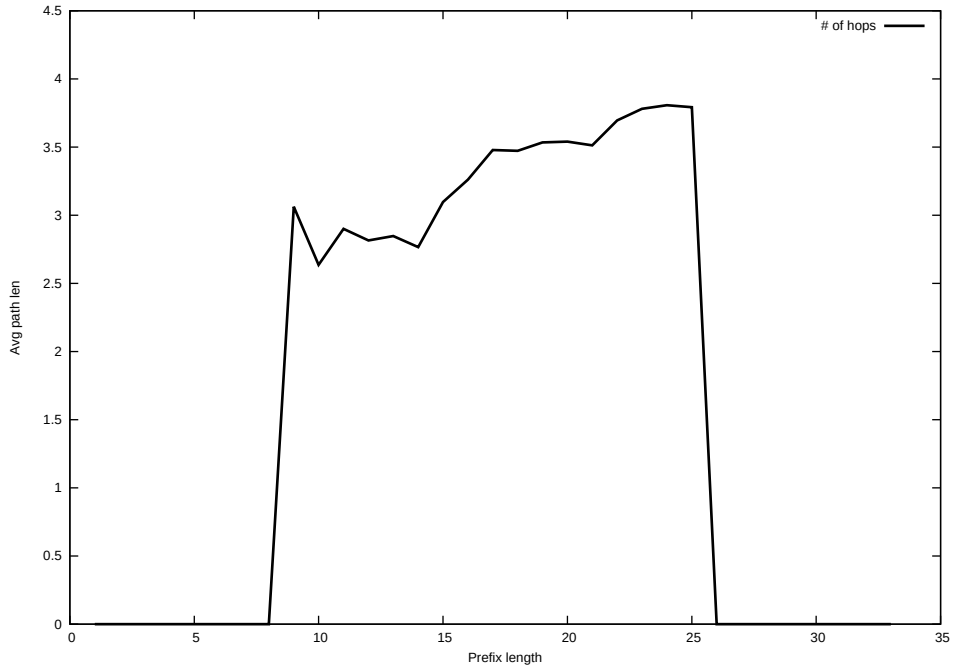
2013-07-31



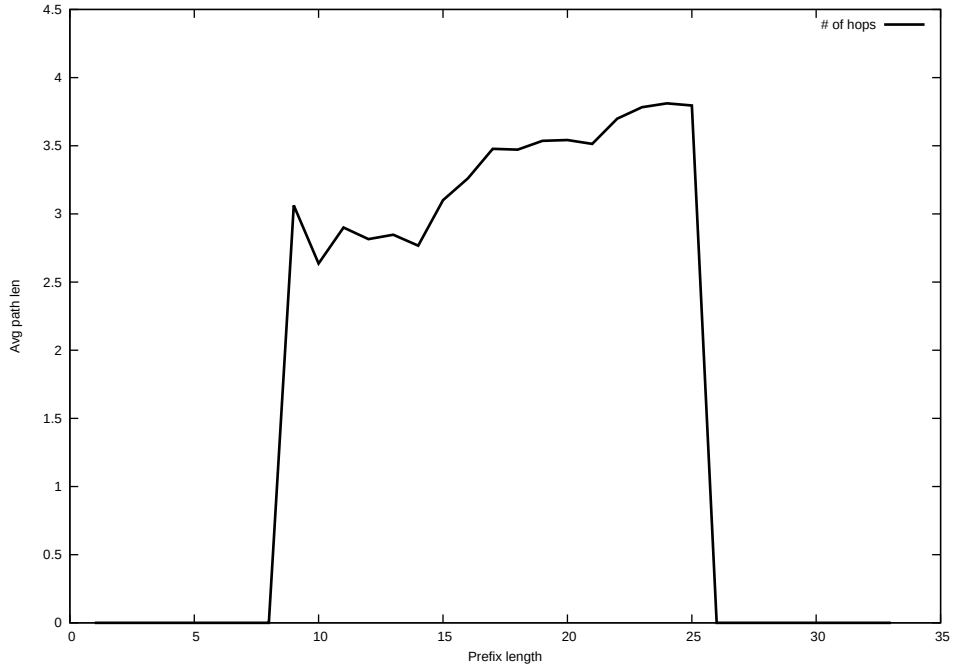
2013-08-01



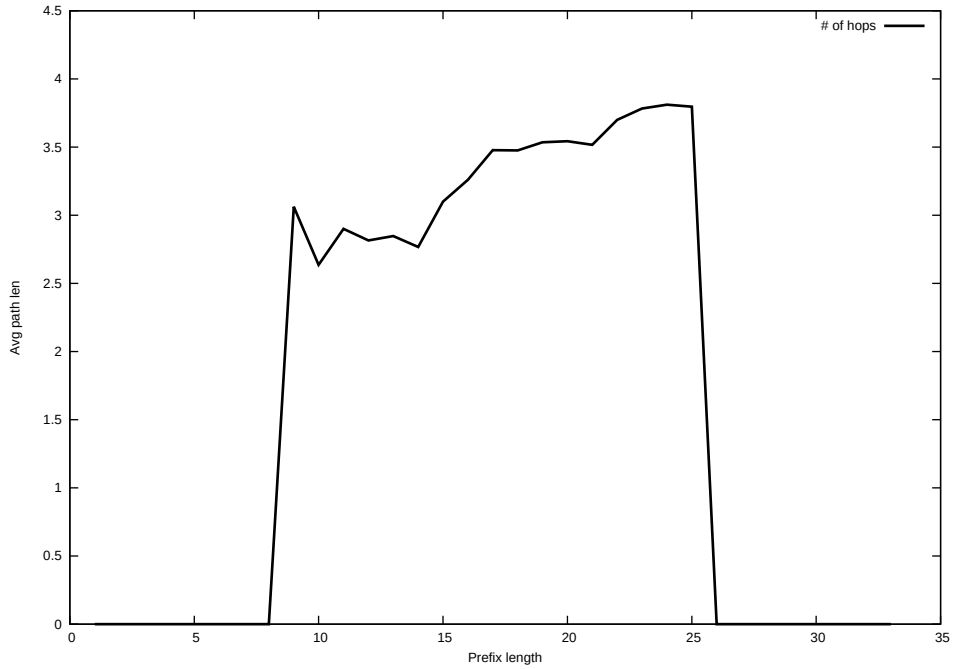
2013-08-02



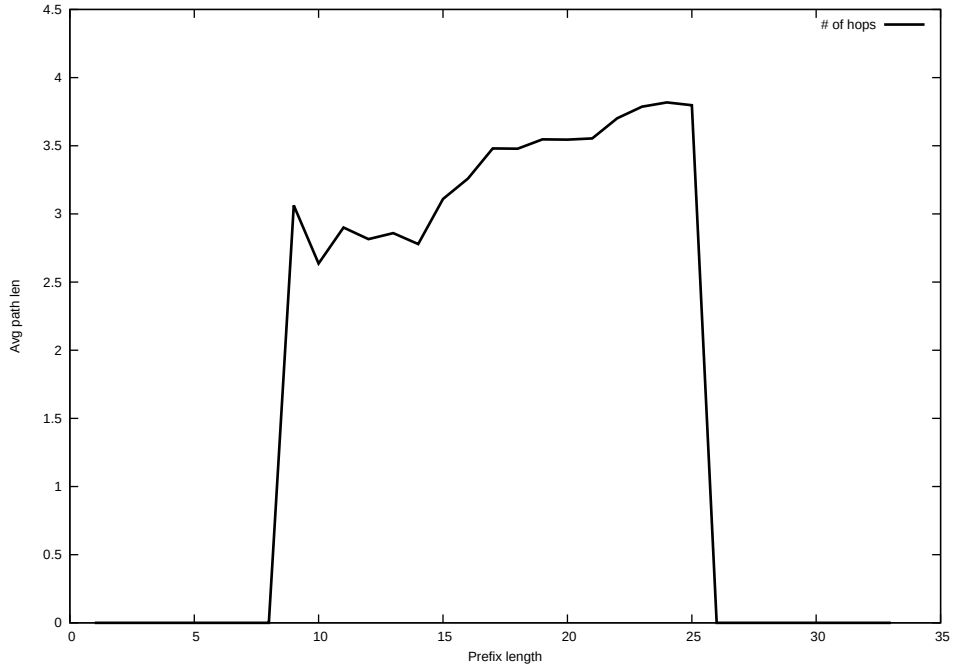
2013-08-03



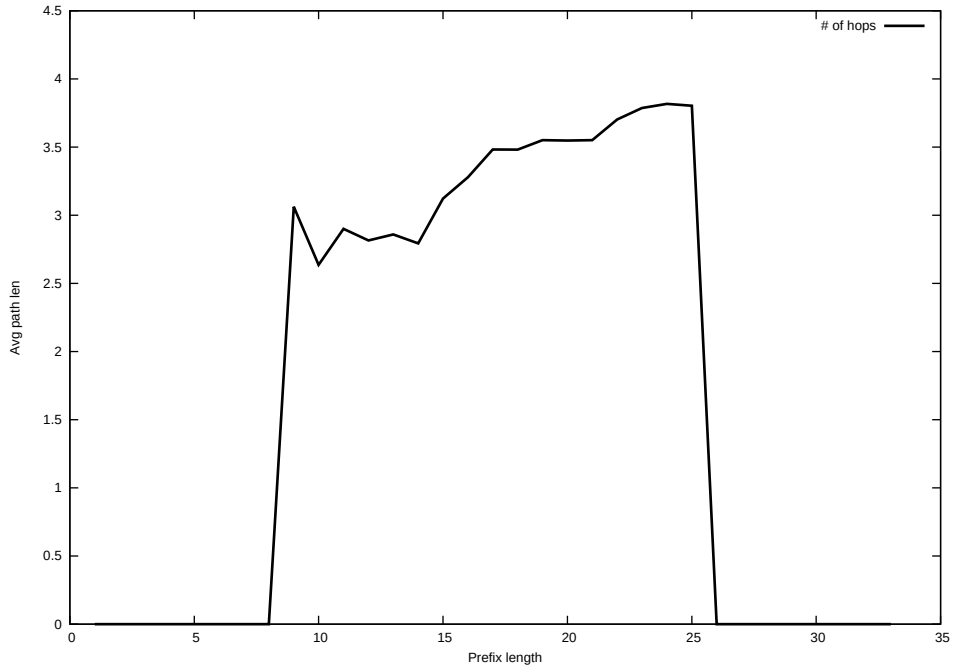
2013-08-04



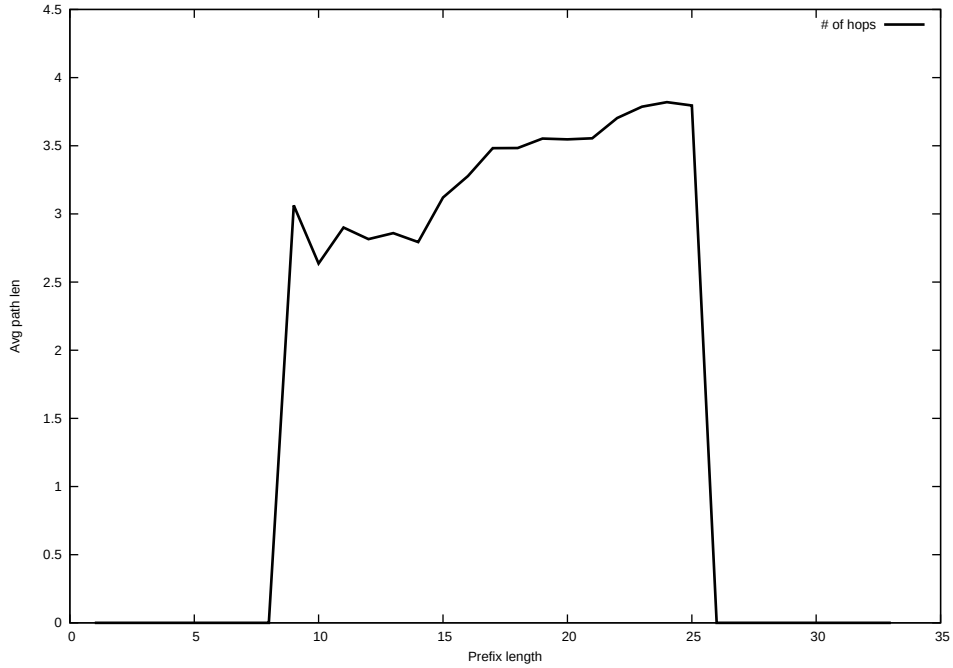
2013-08-05



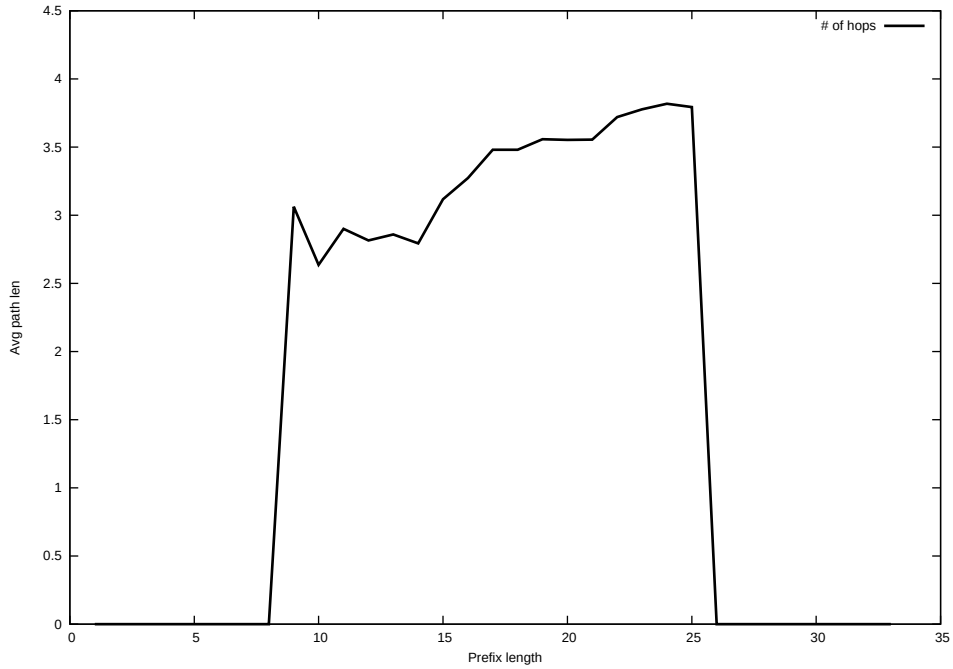
2013-08-06



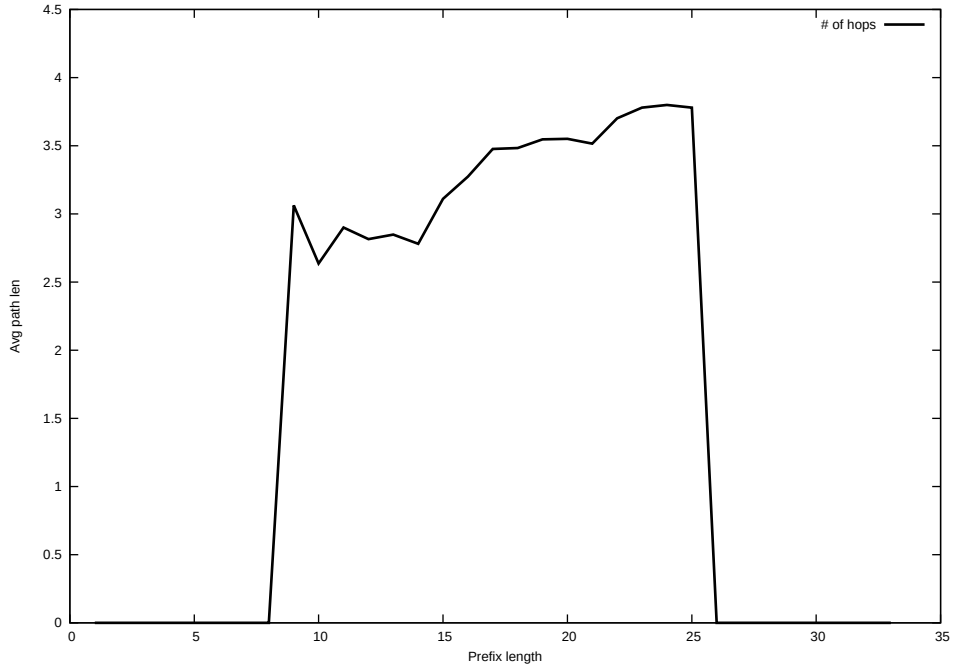
2013-08-07



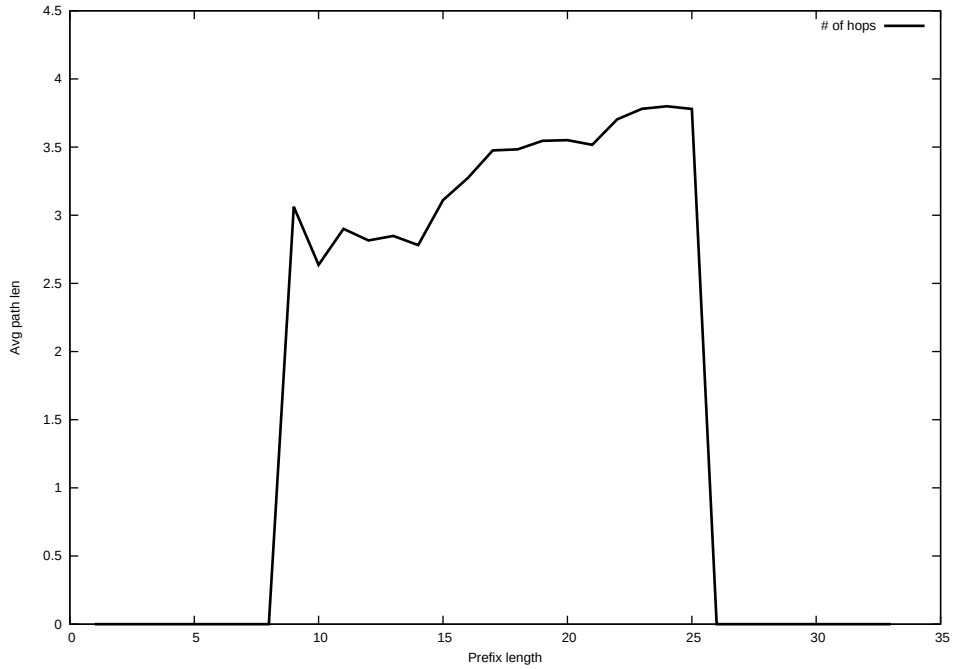
2013-08-08



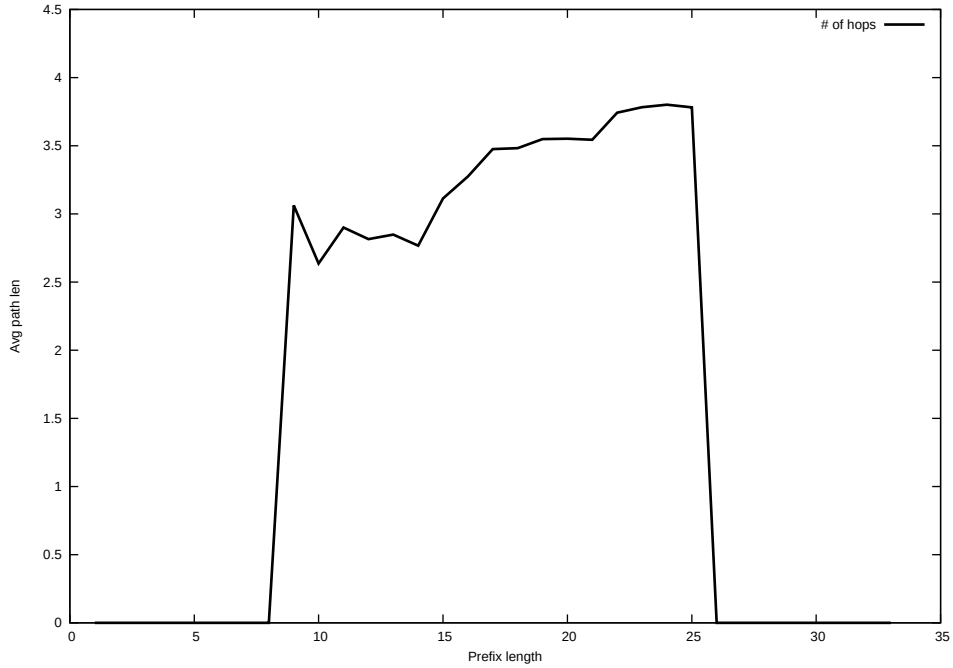
2013-08-09



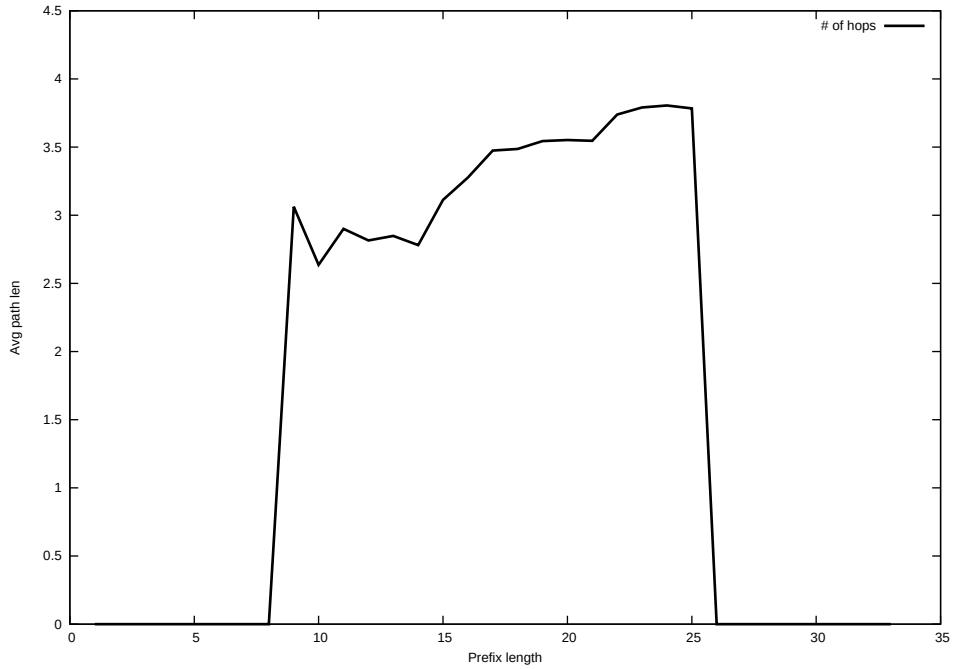
2013-08-10



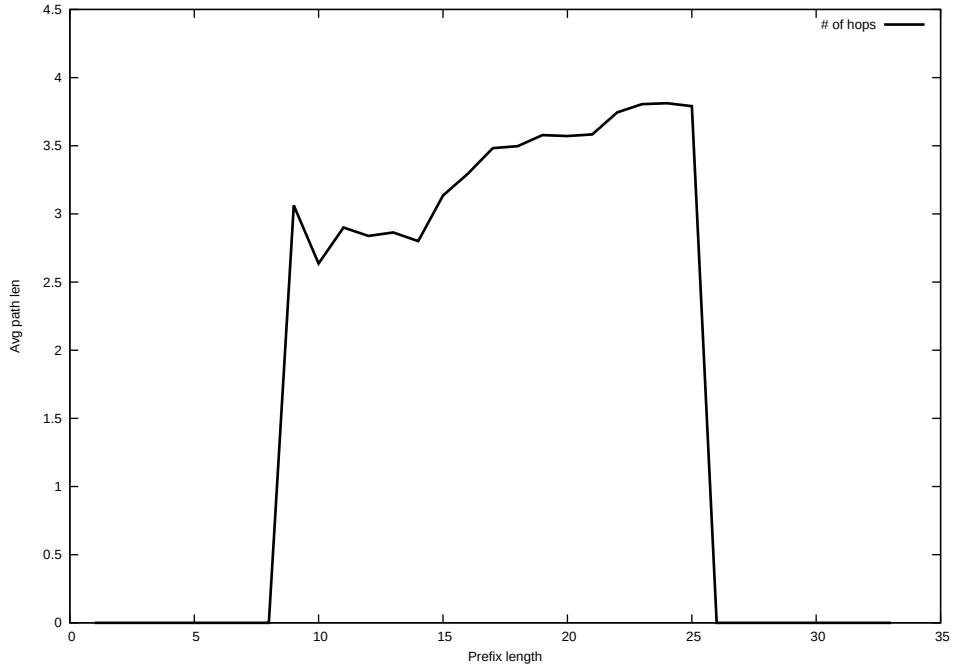
2013-08-11



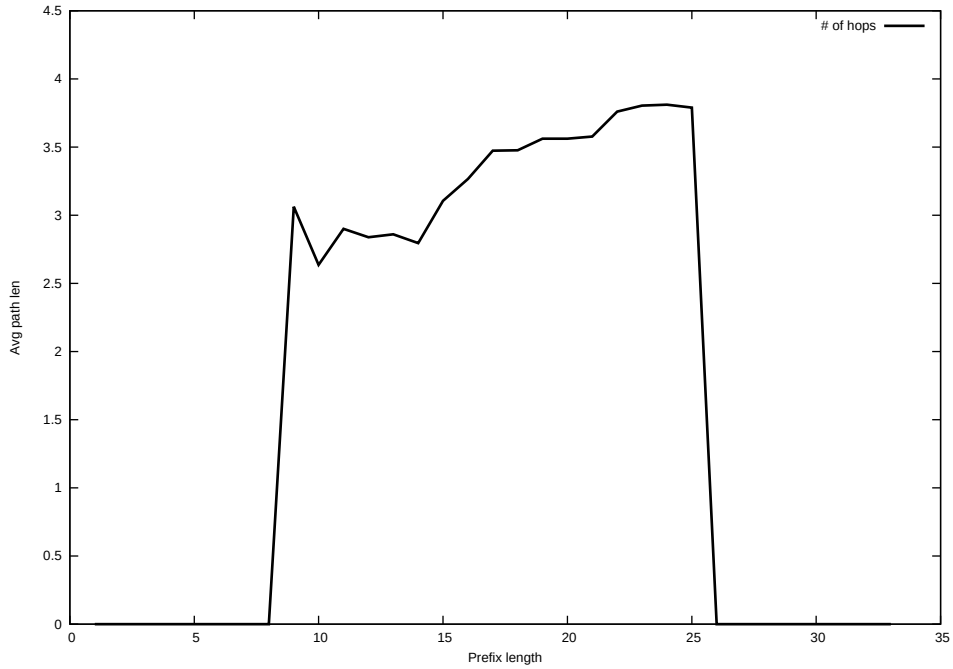
2013-08-12



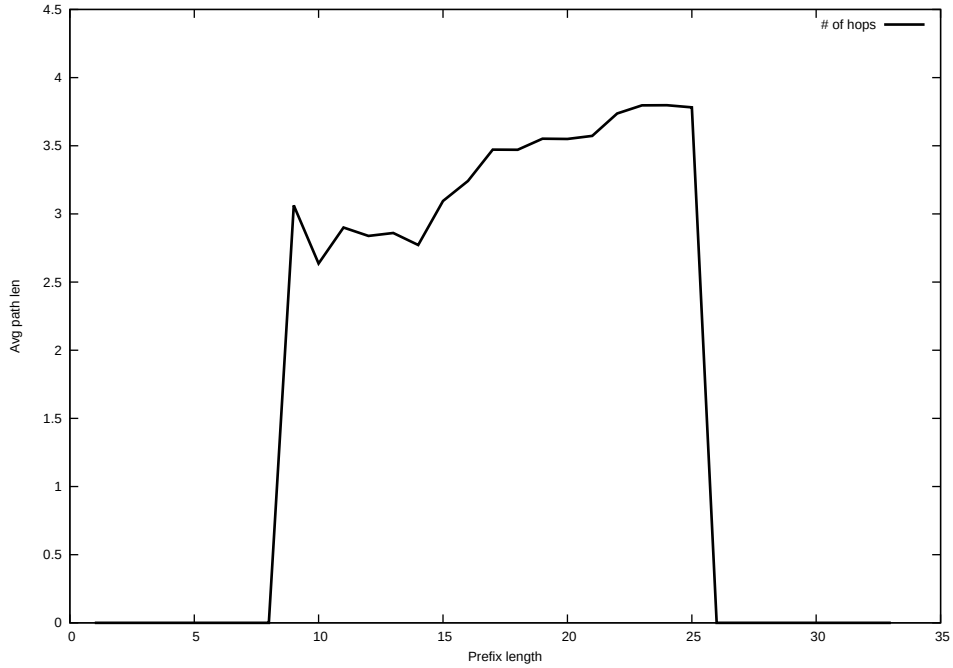
2013-08-13



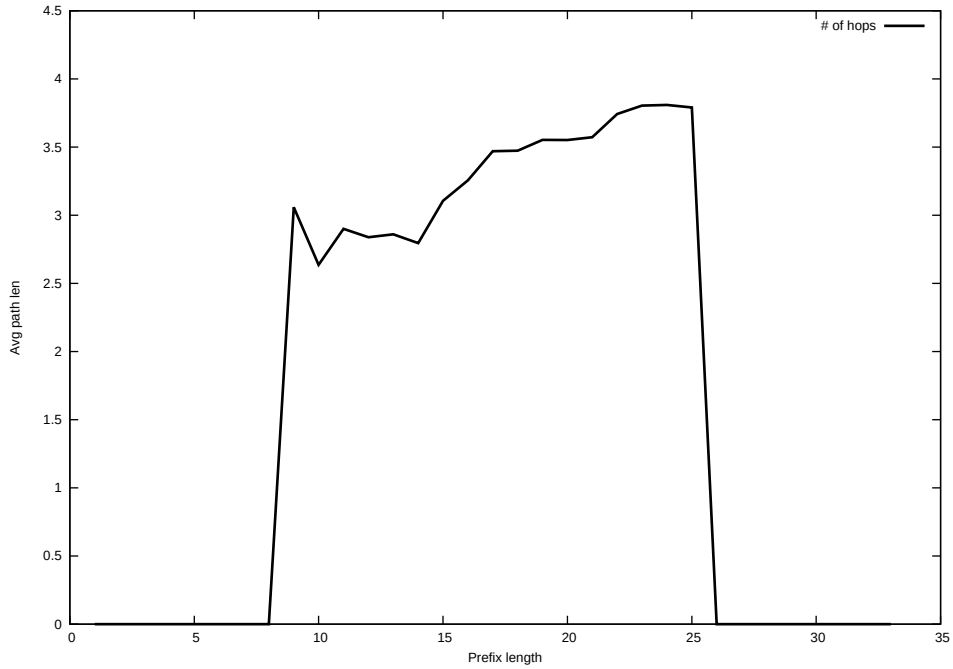
2013-08-15



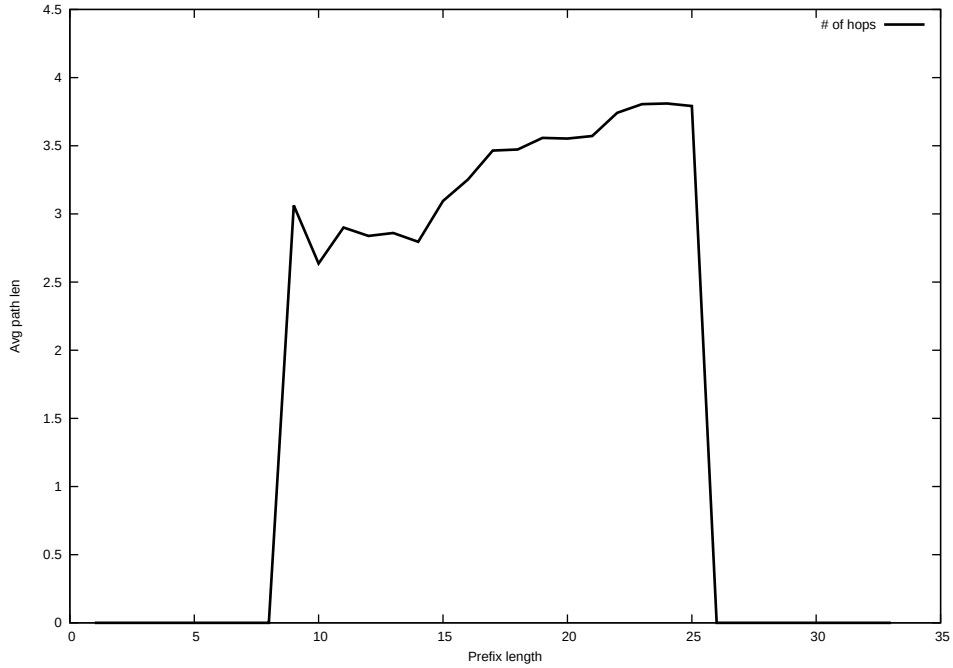
2013-08-16



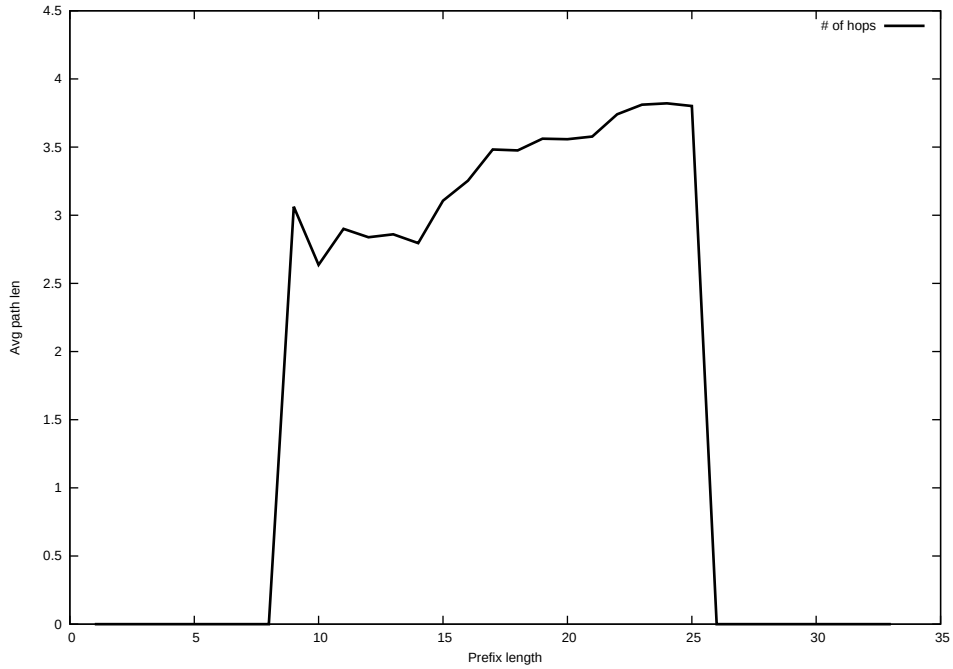
2013-08-17



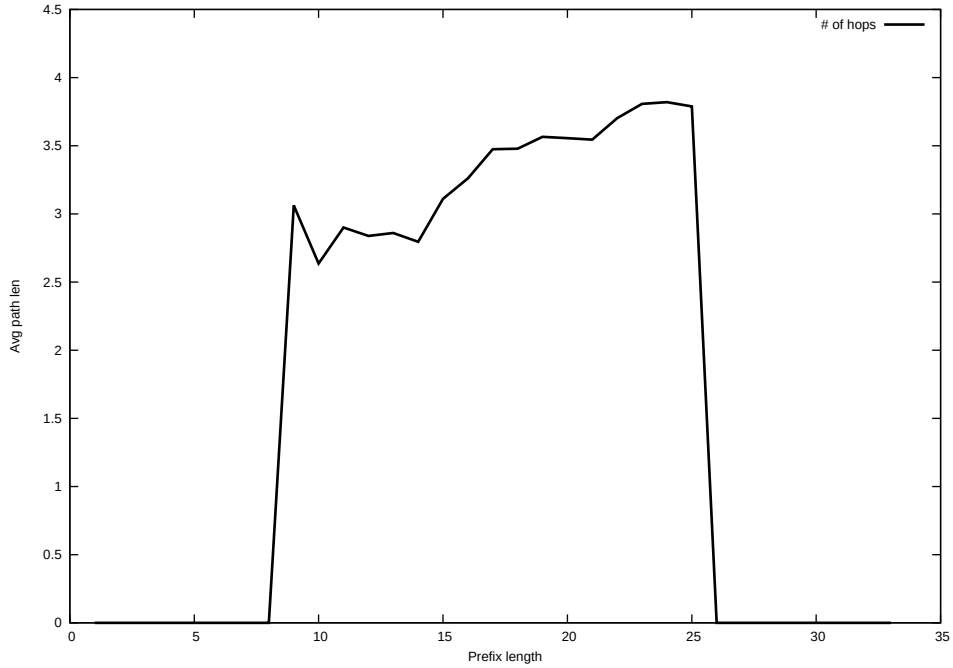
2013-08-18



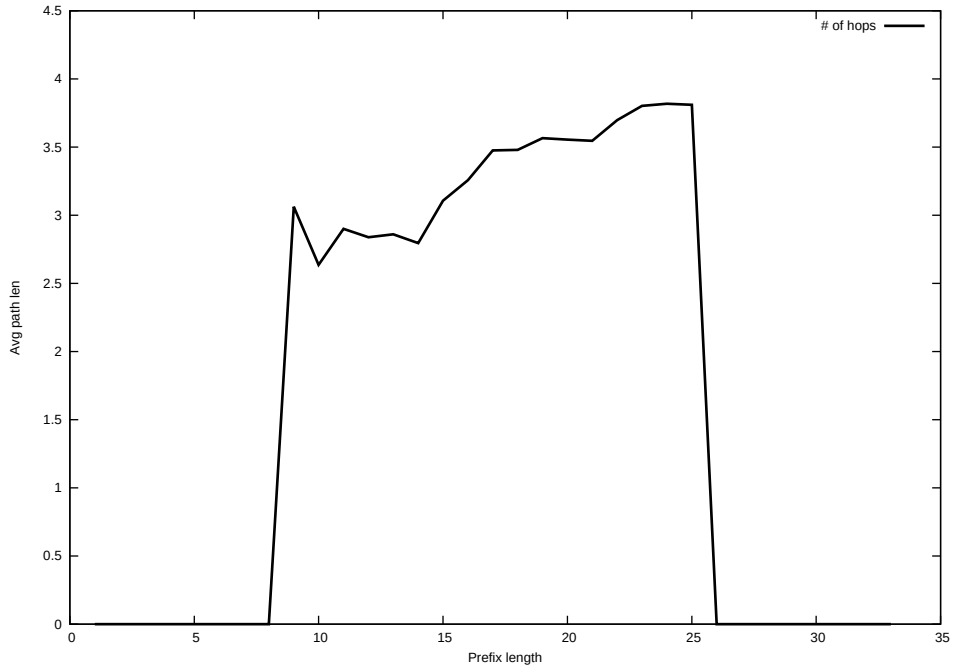
2013-08-19



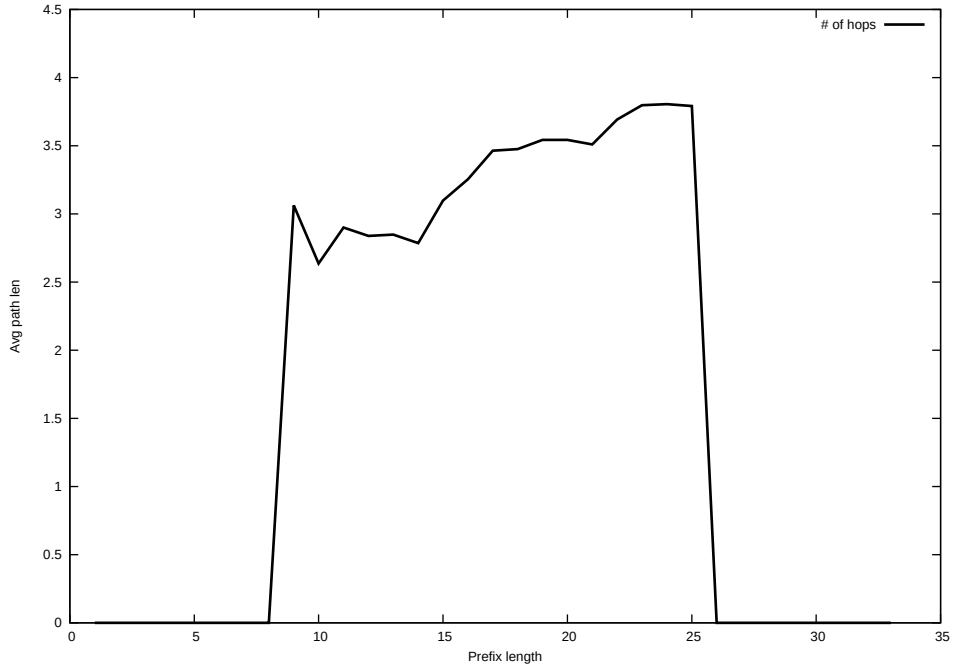
2013-08-20



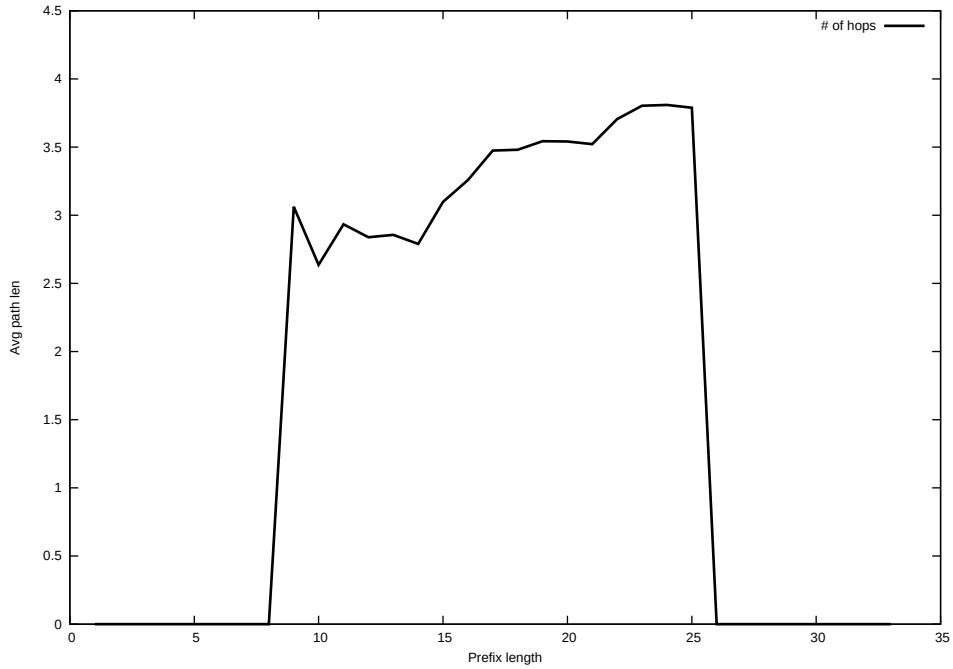
2013-08-21



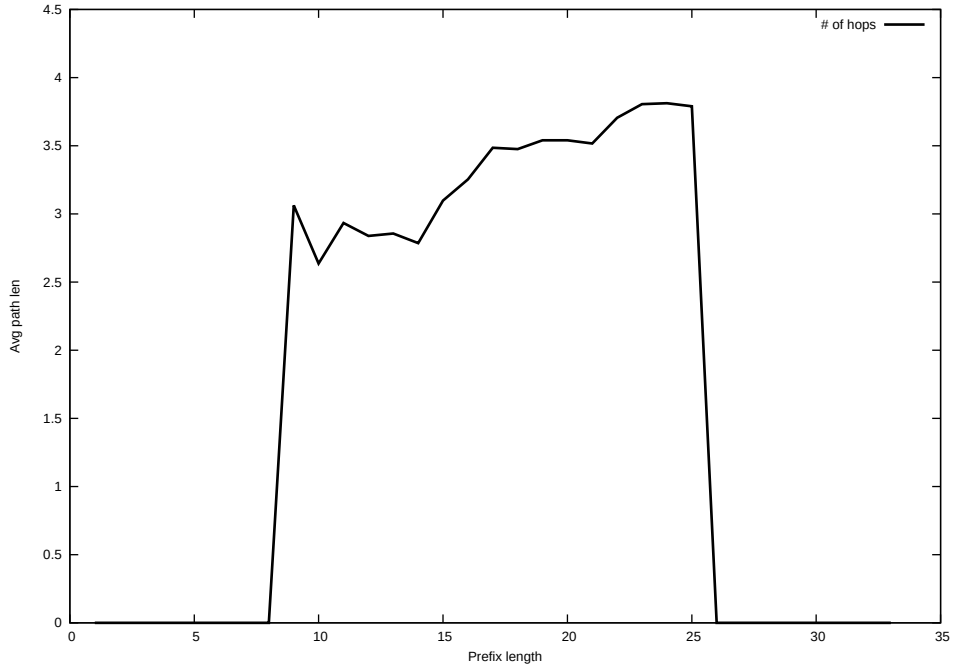
2013-08-22



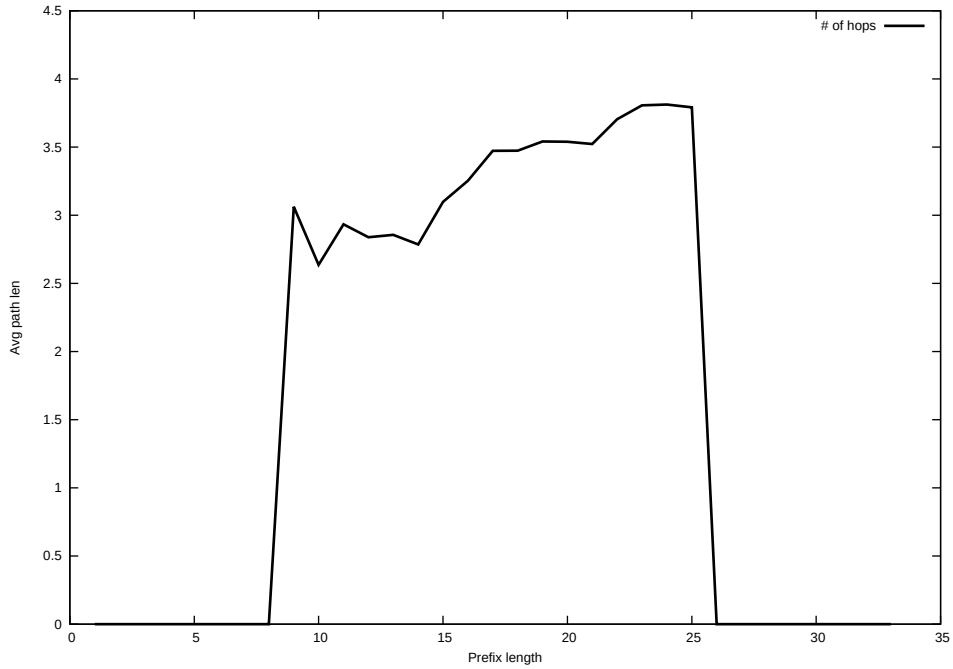
2013-08-23



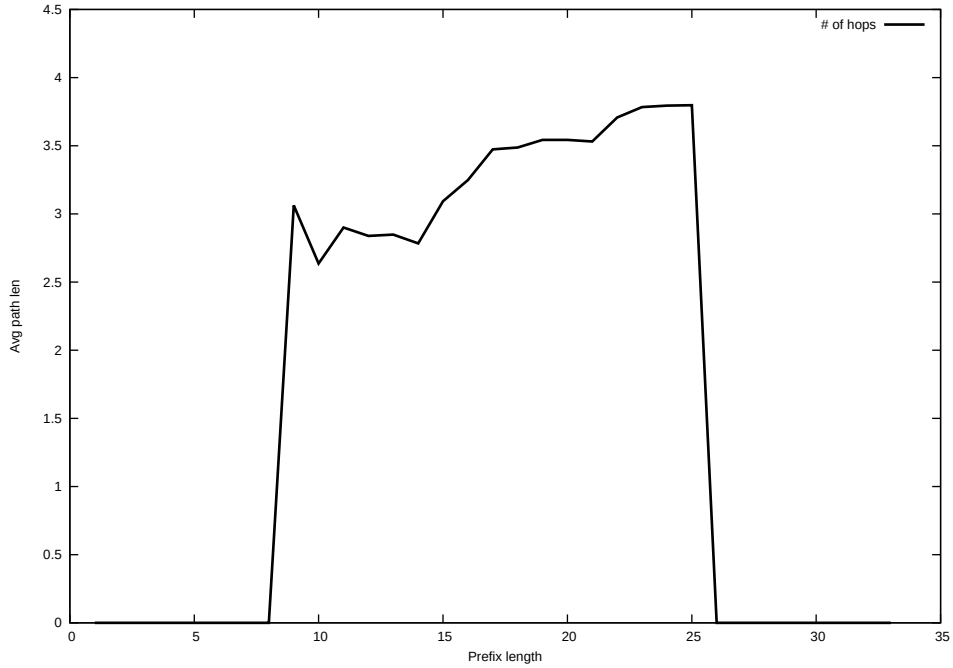
2013-08-24



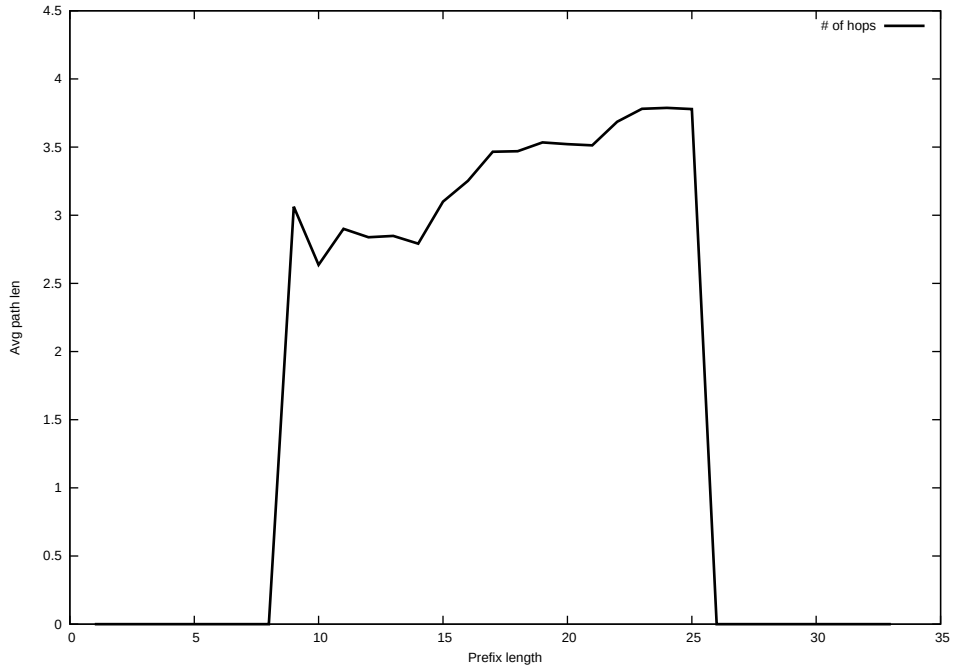
2013-08-25



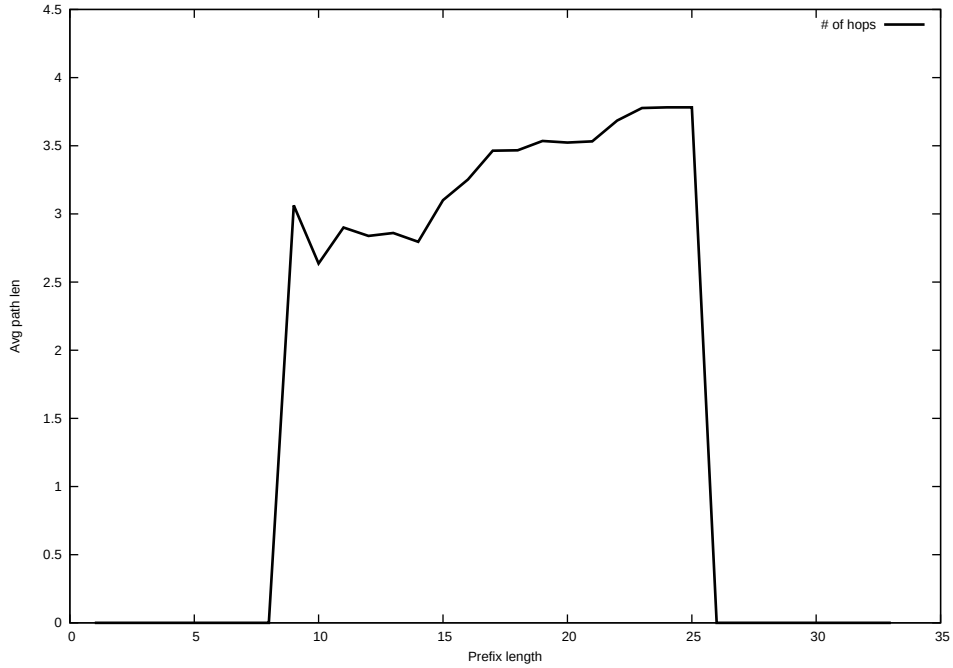
2013-08-26



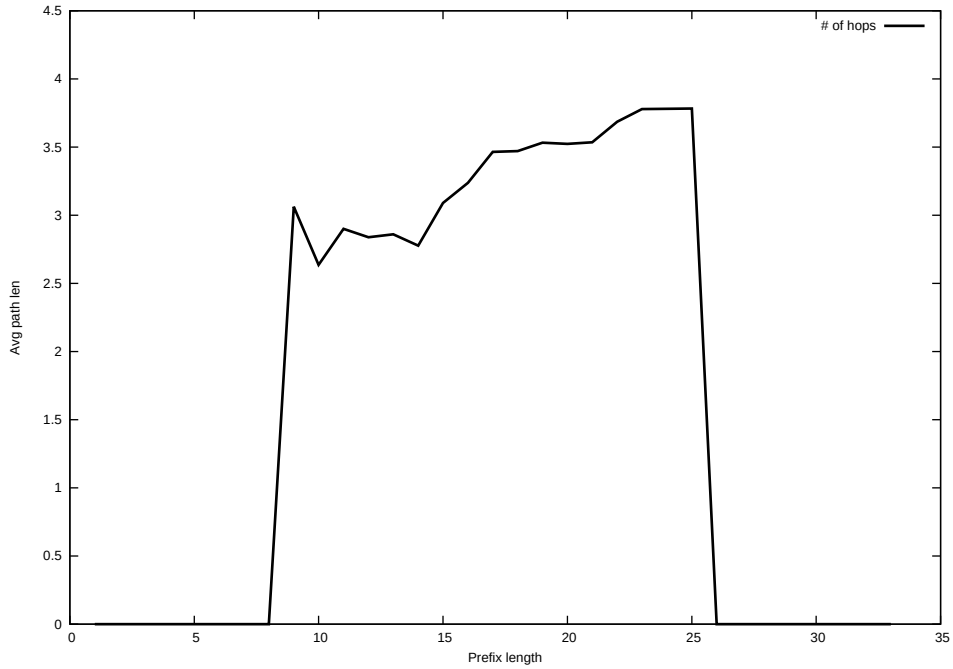
2013-08-27



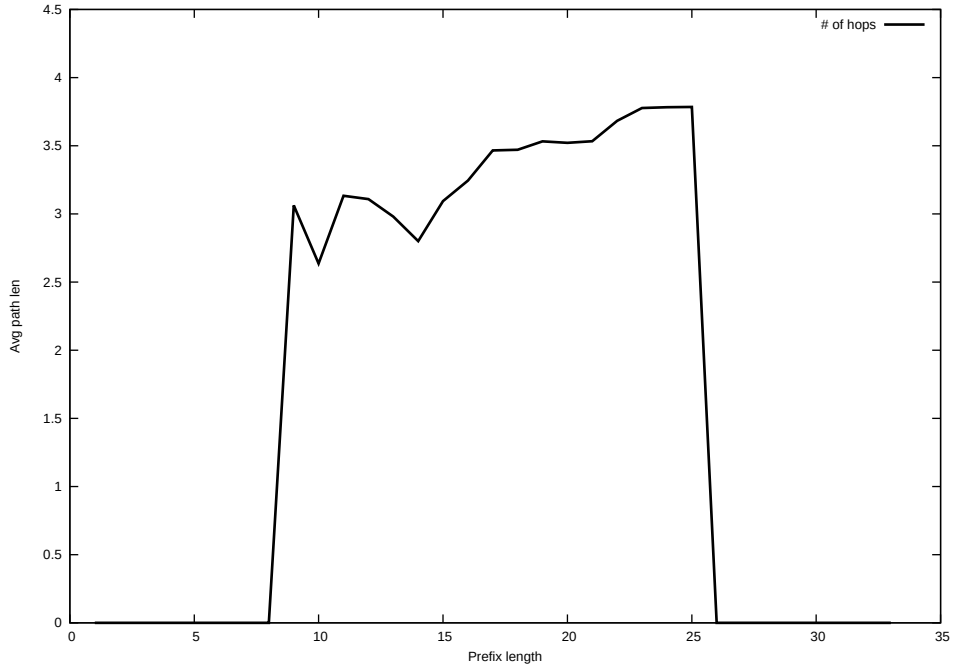
2013-08-28



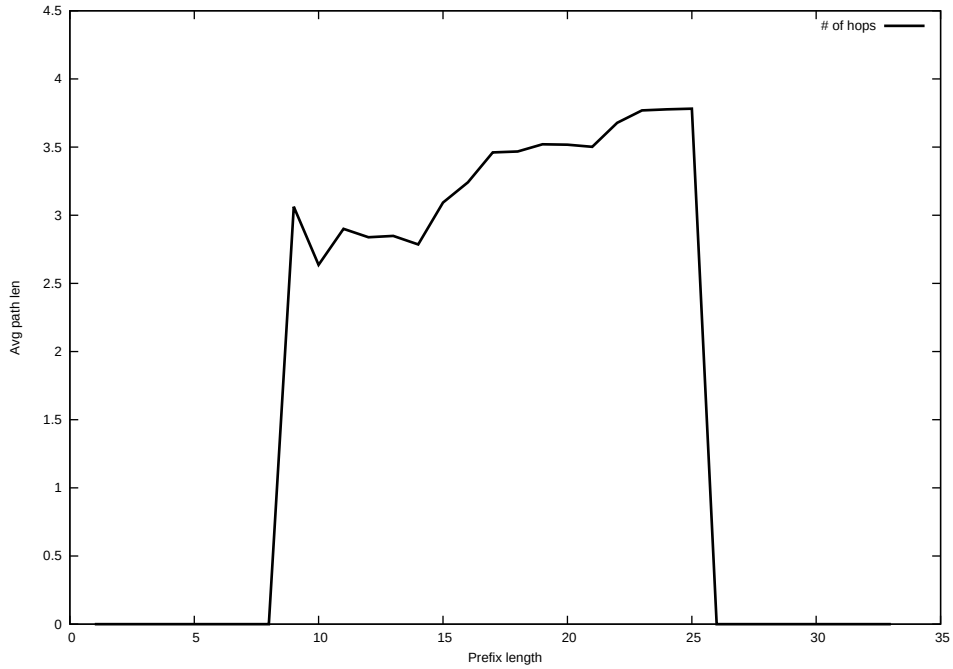
2013-08-29



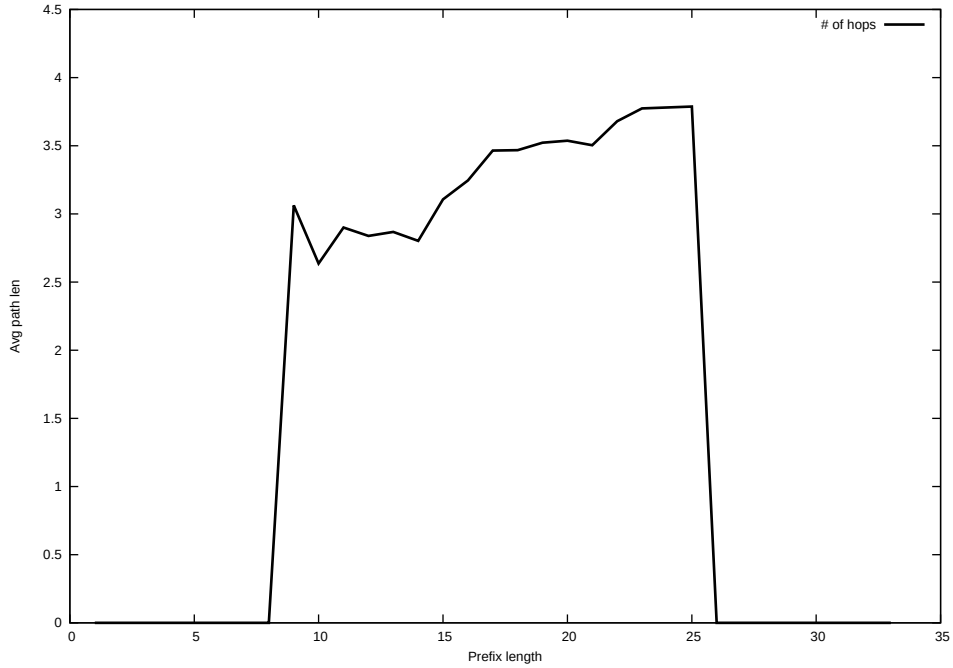
2013-08-30



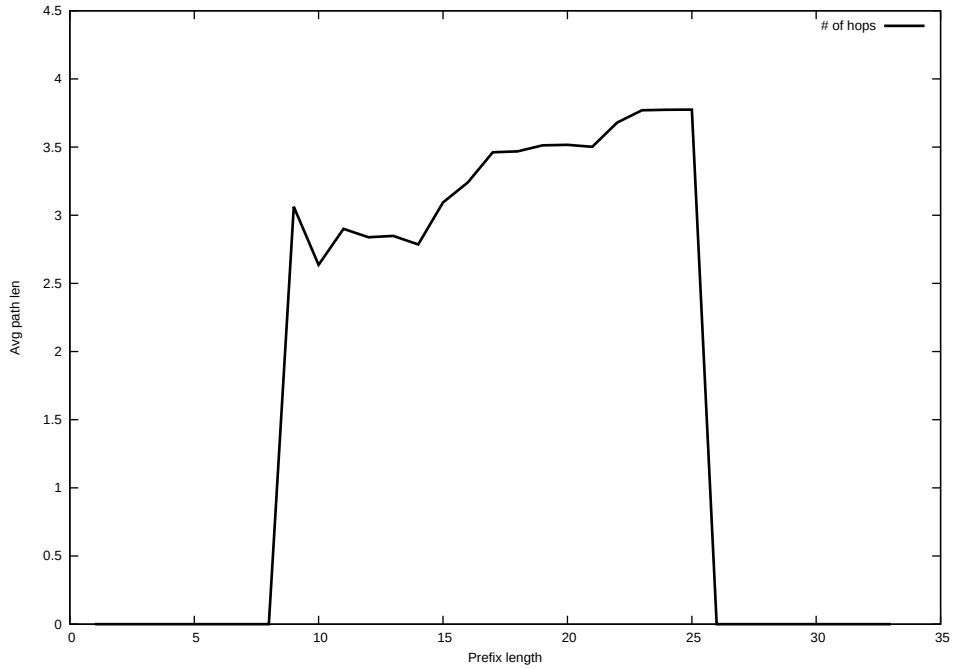
2013-08-31



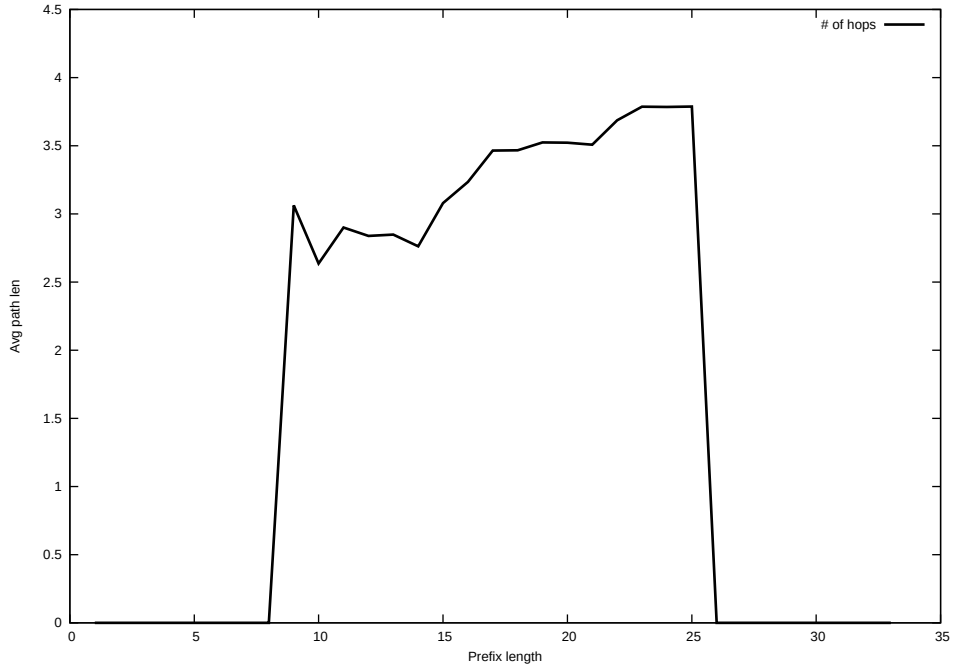
2013-09-01



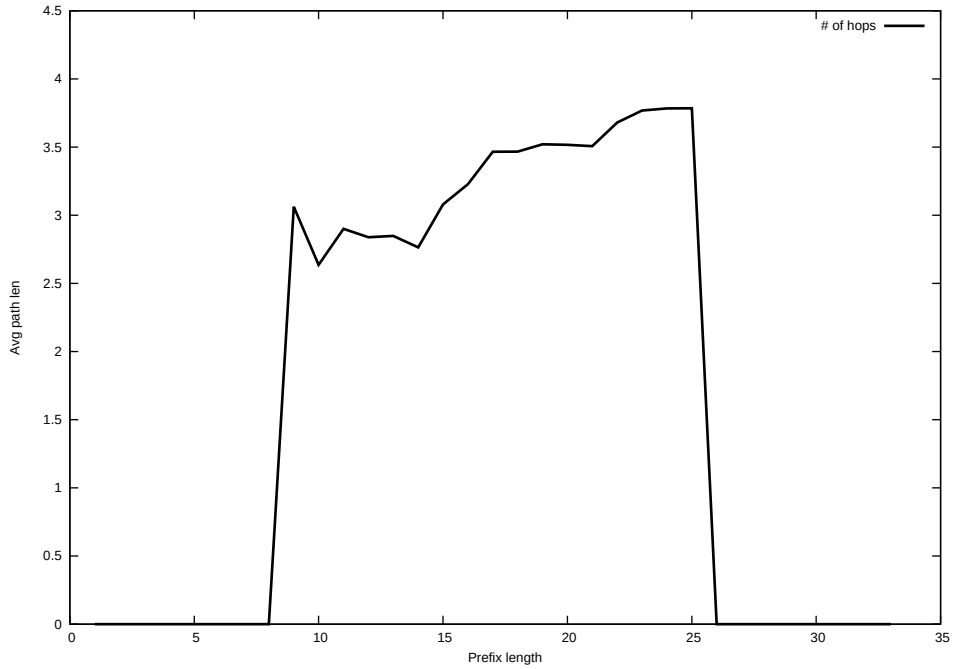
2013-09-02



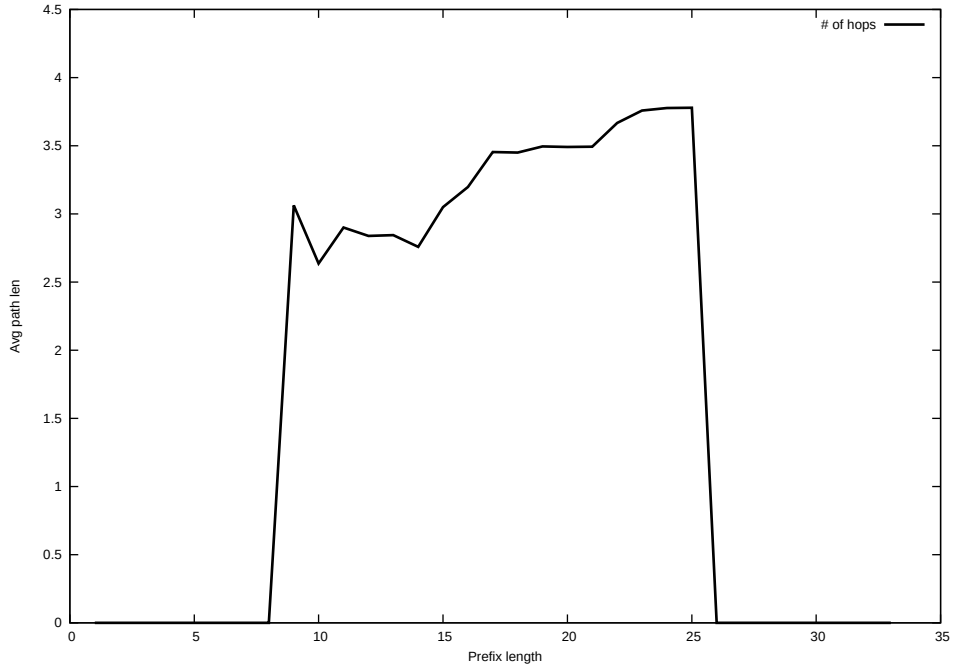
2013-09-03



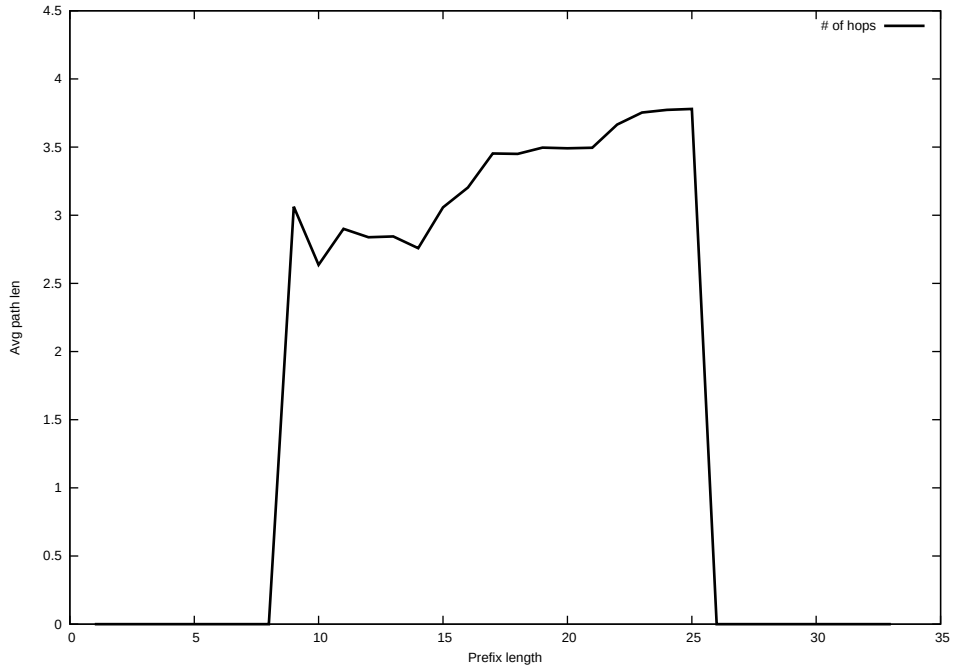
2013-09-04



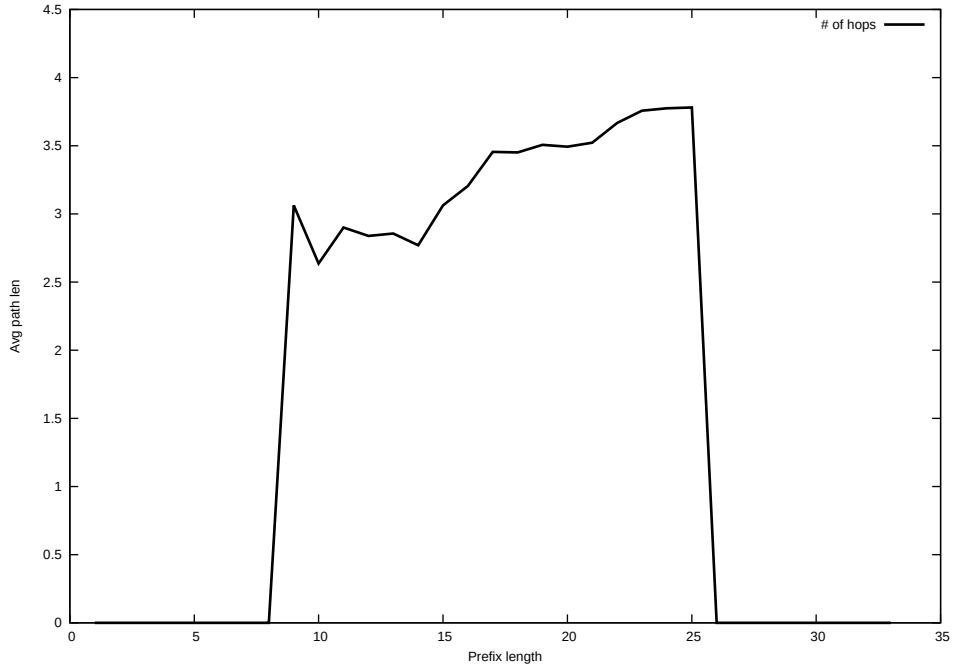
2013-09-05



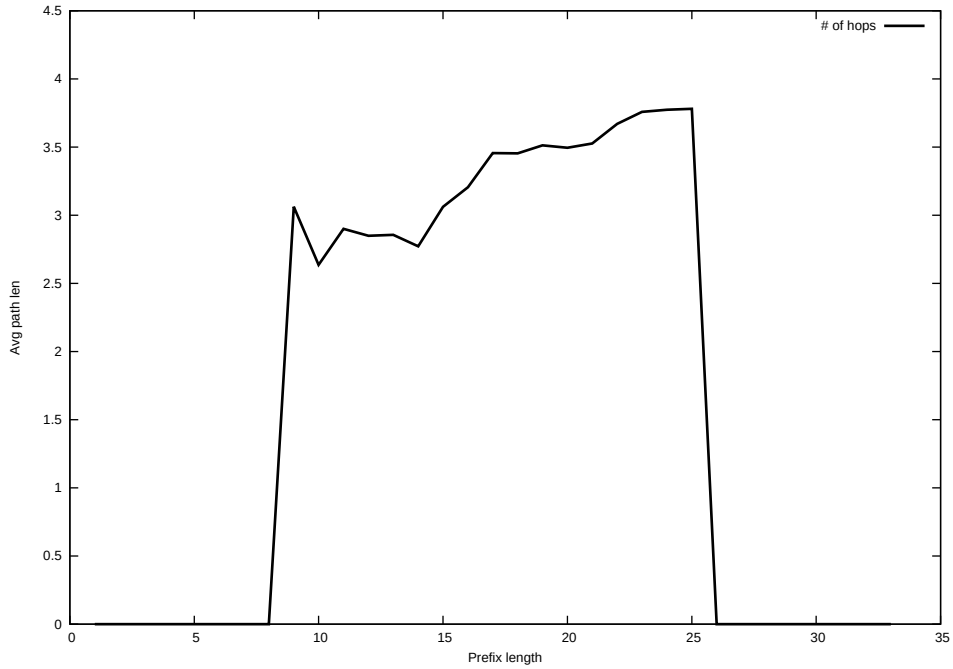
2013-09-06



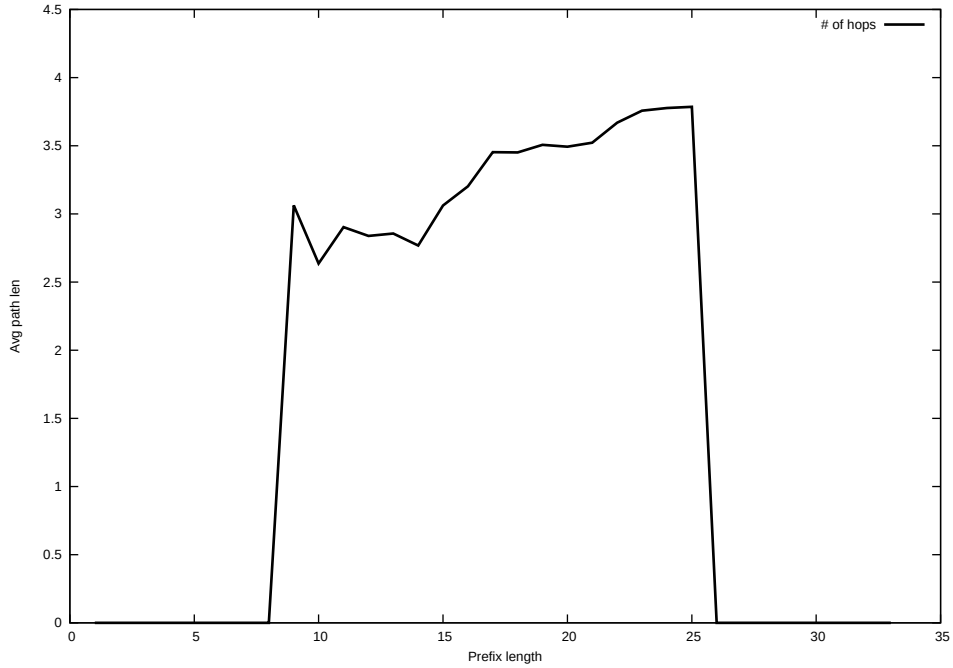
2013-09-07



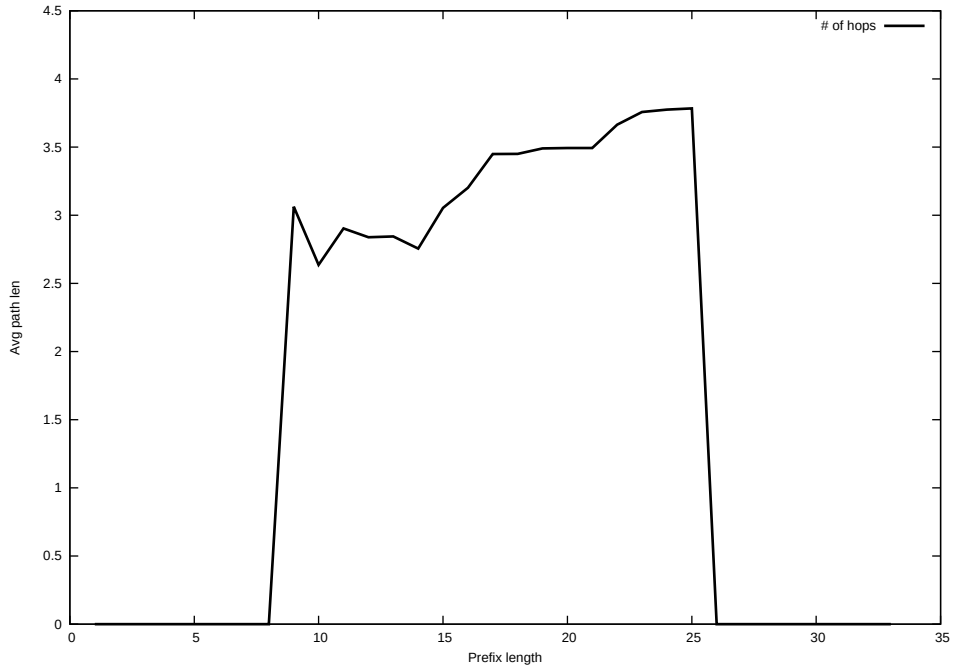
2013-09-08



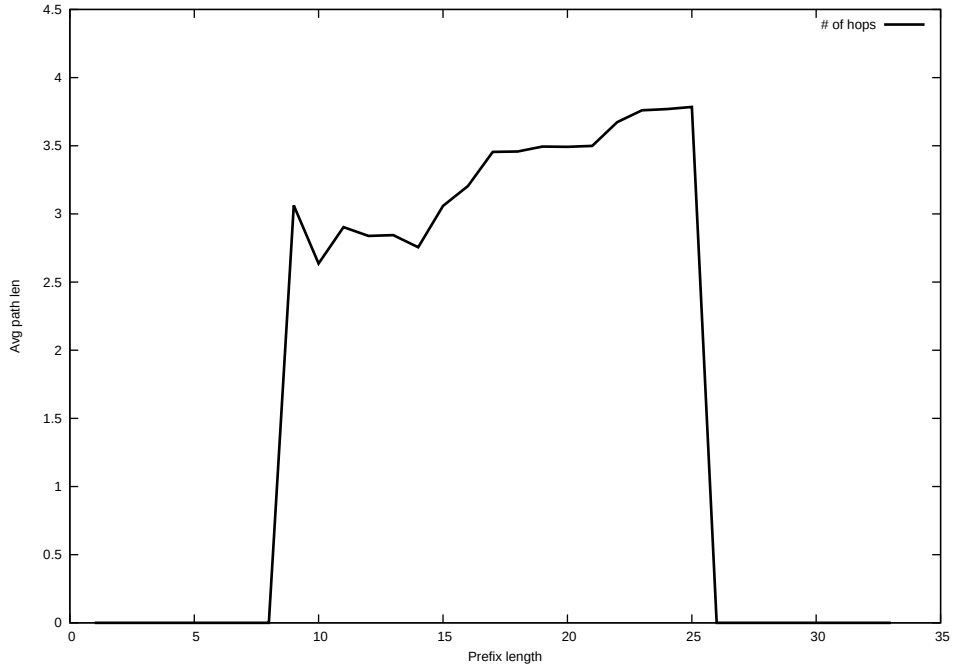
2013-09-09



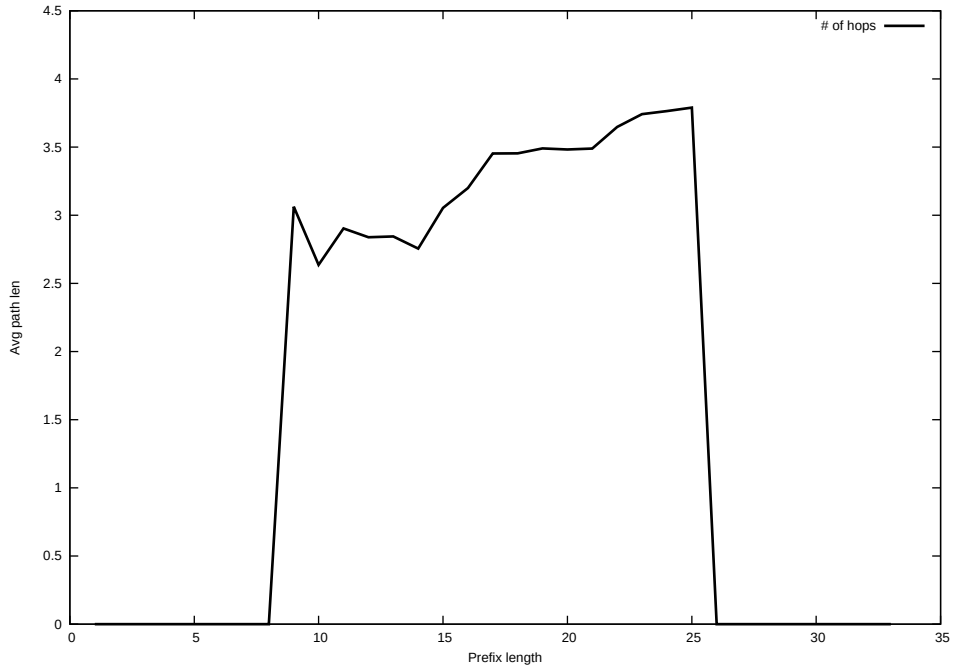
2013-09-10



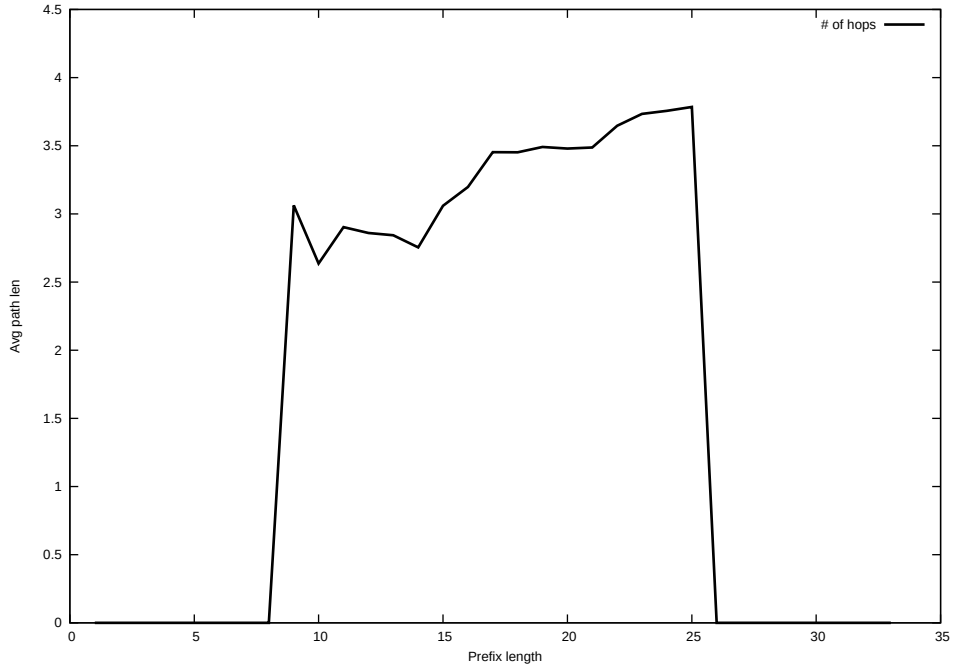
2013-09-11



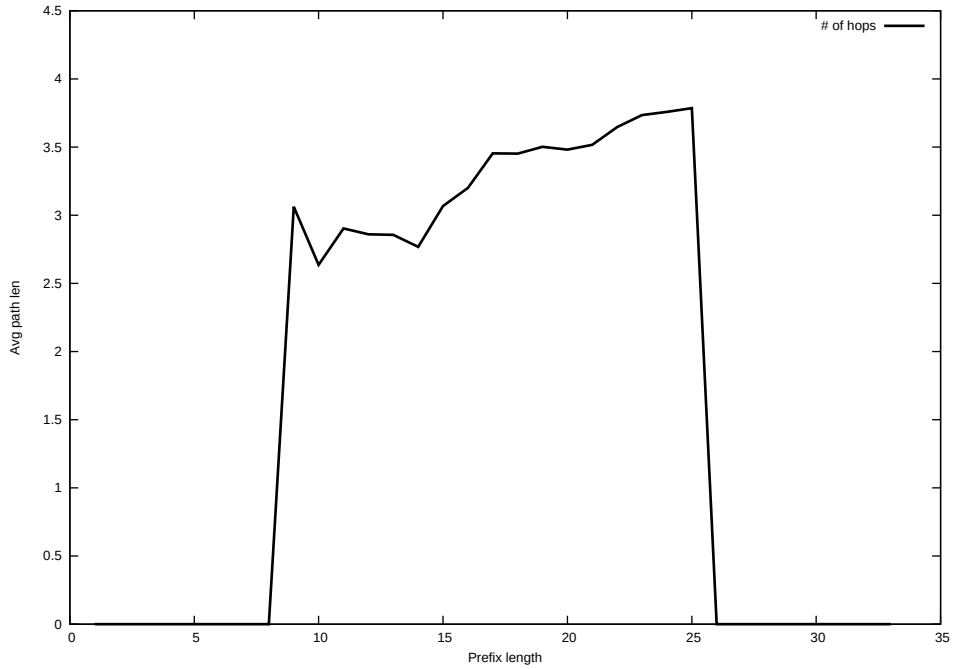
2013-09-12



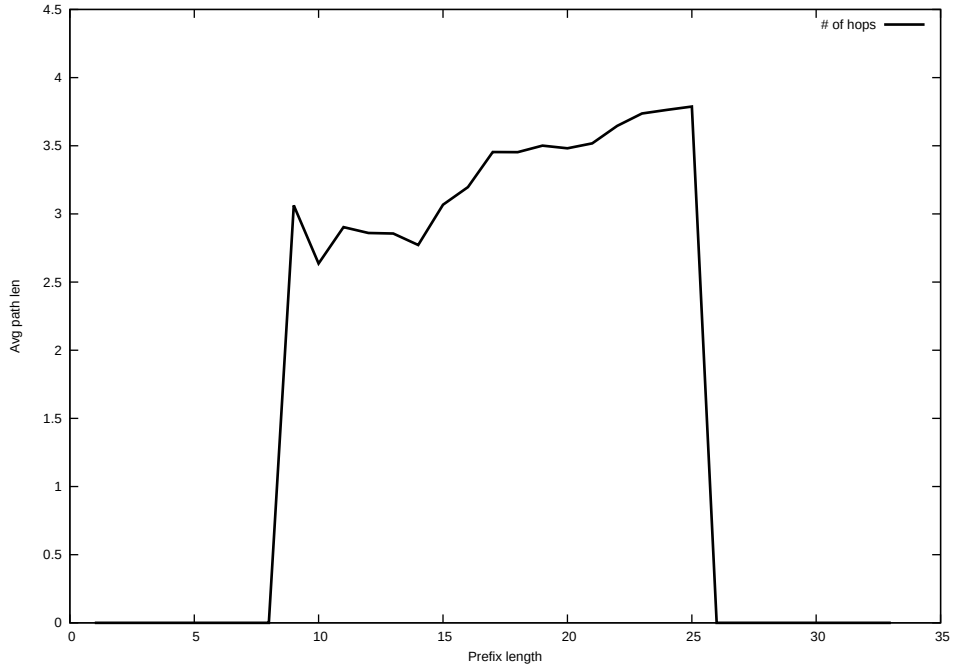
2013-09-13



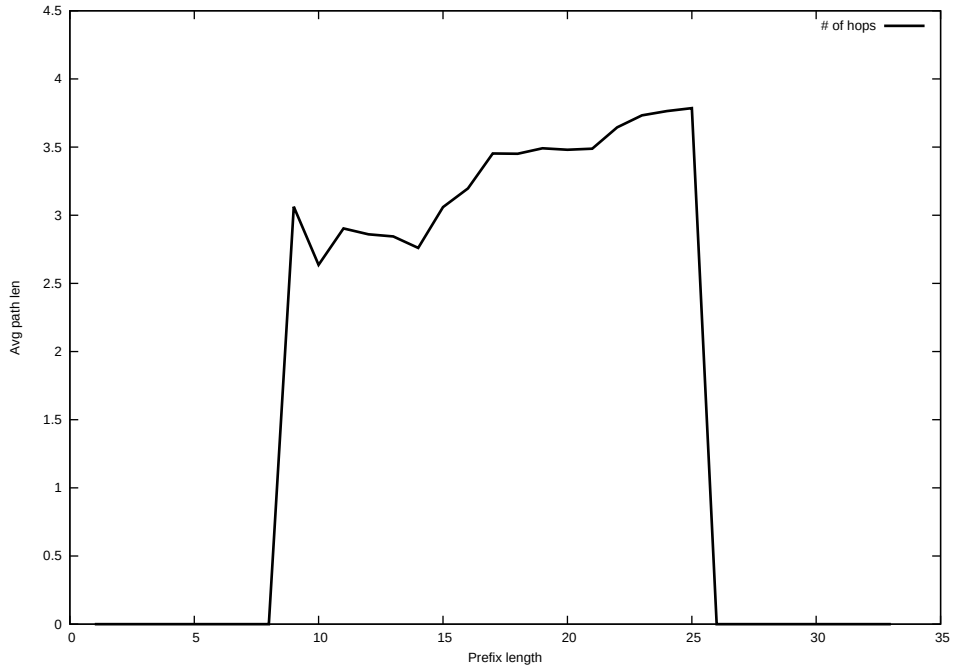
2013-09-14



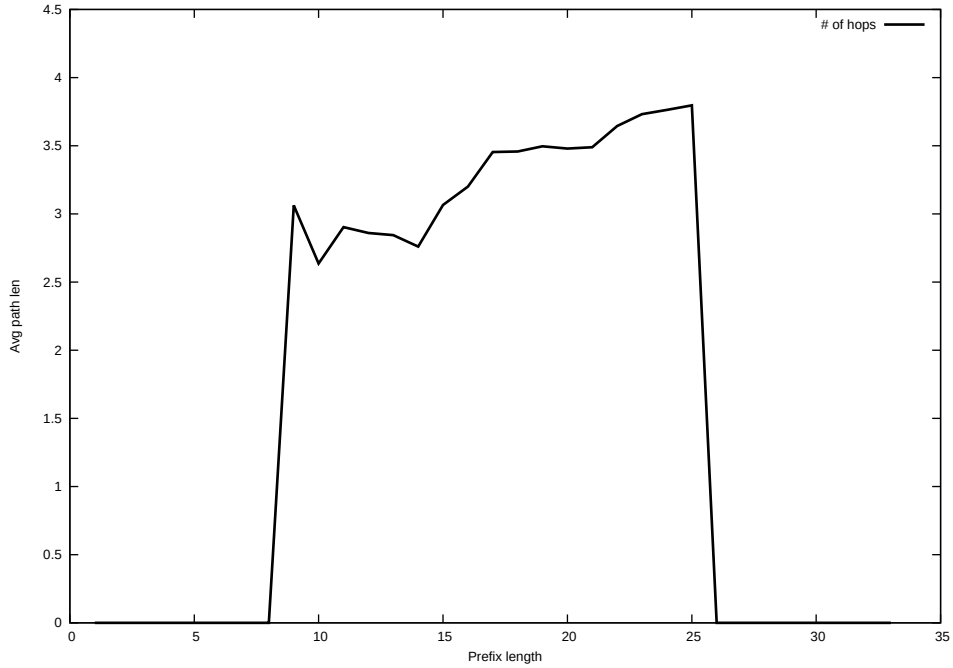
2013-09-15



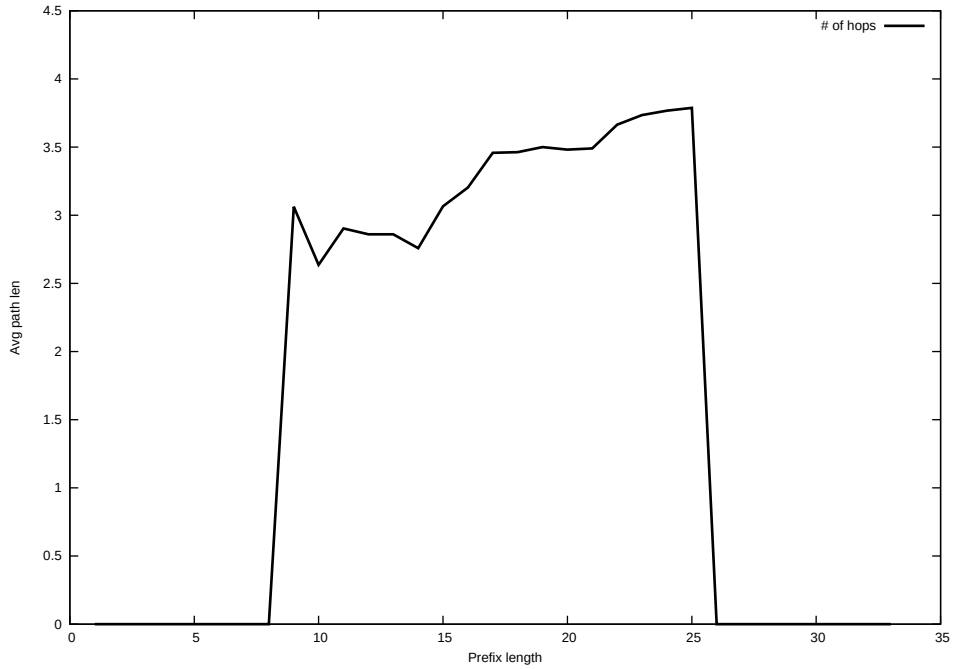
2013-09-16



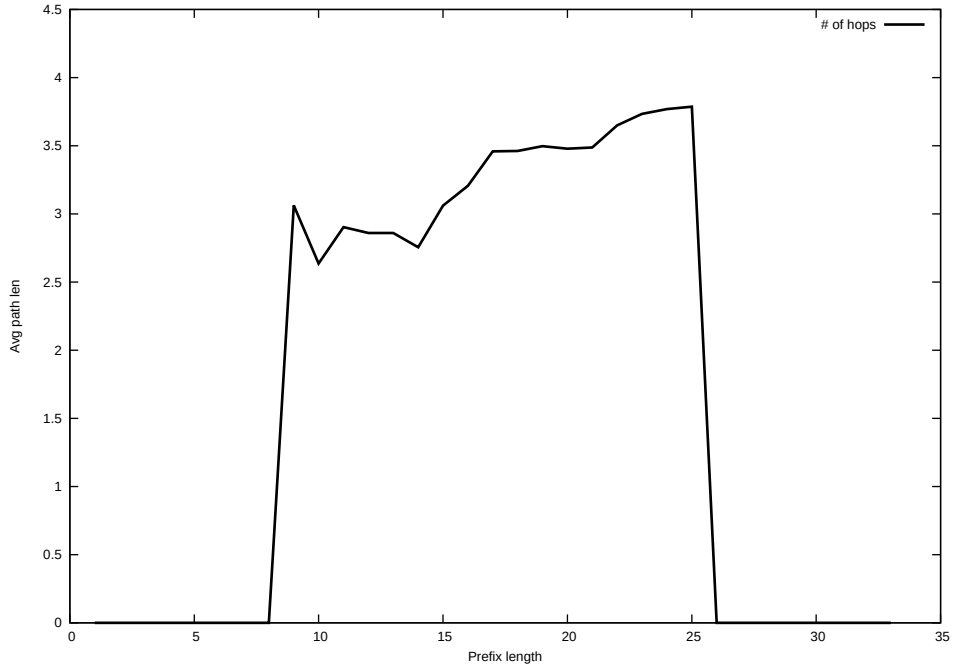
2013-09-17



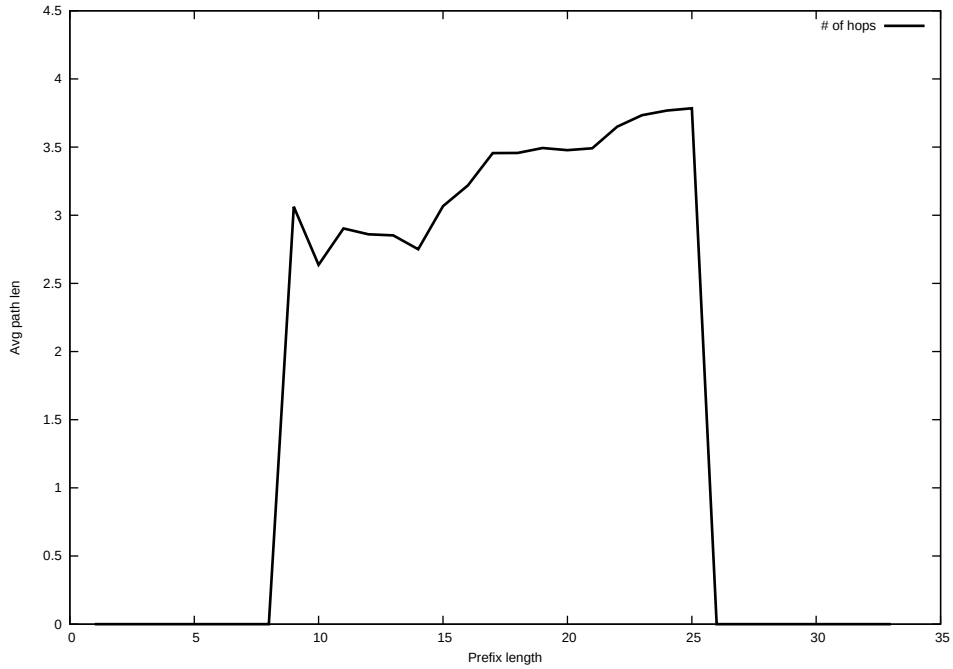
2013-09-18



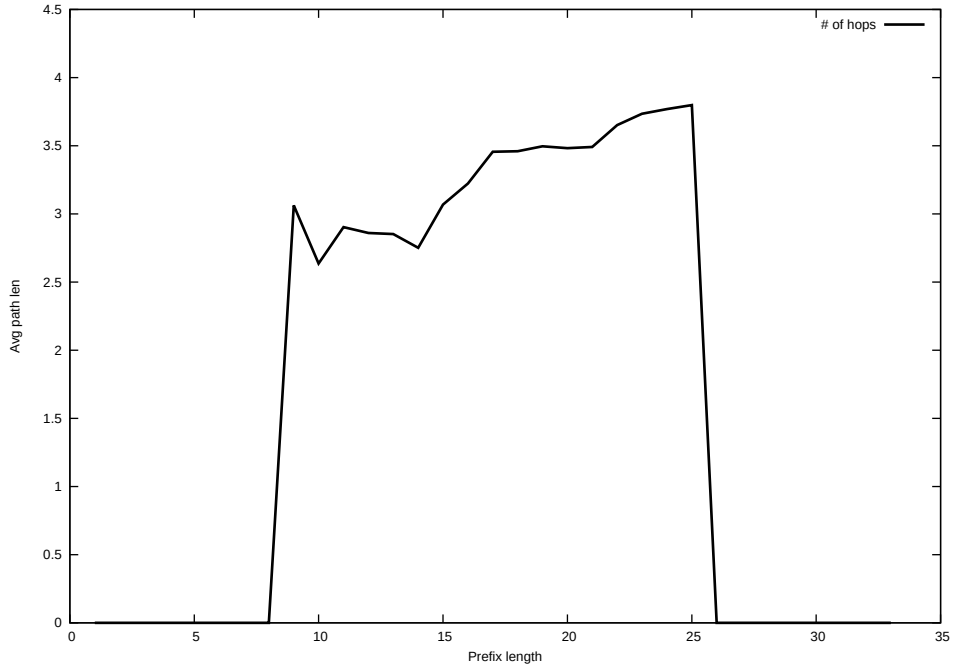
2013-09-19



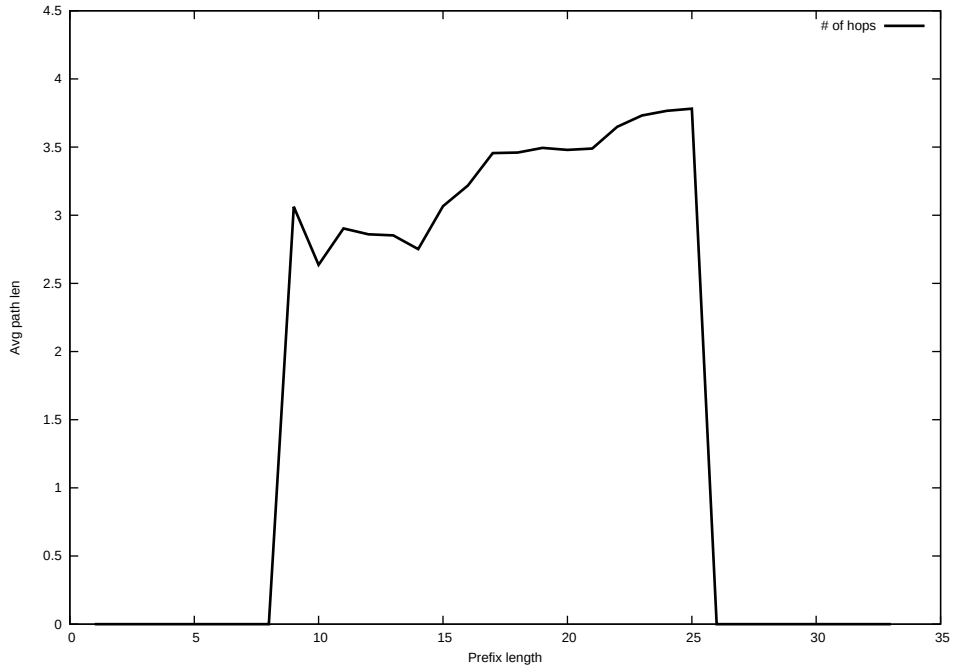
2013-09-20



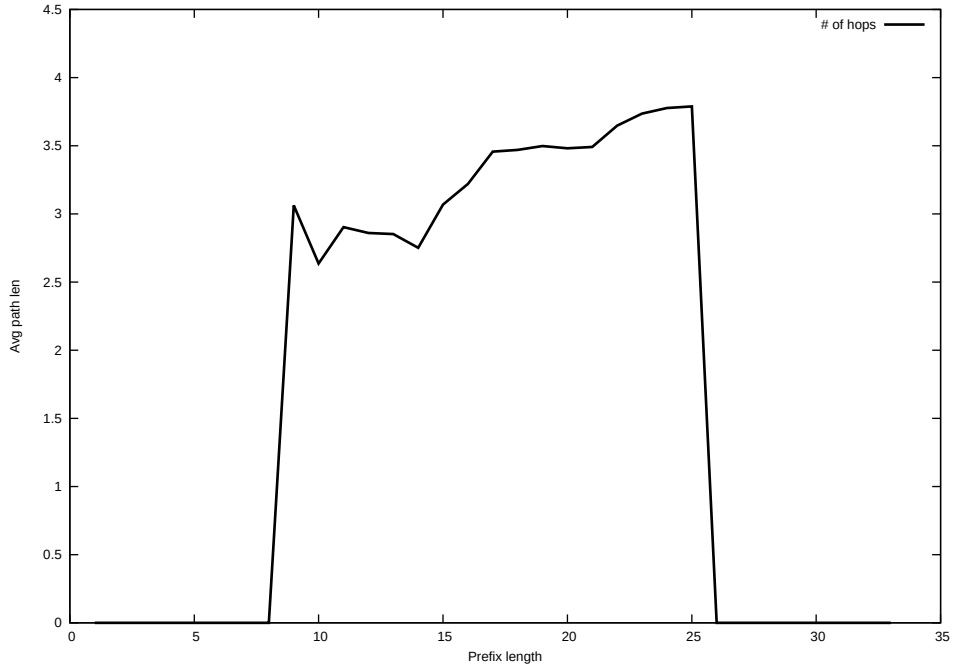
2013-09-21



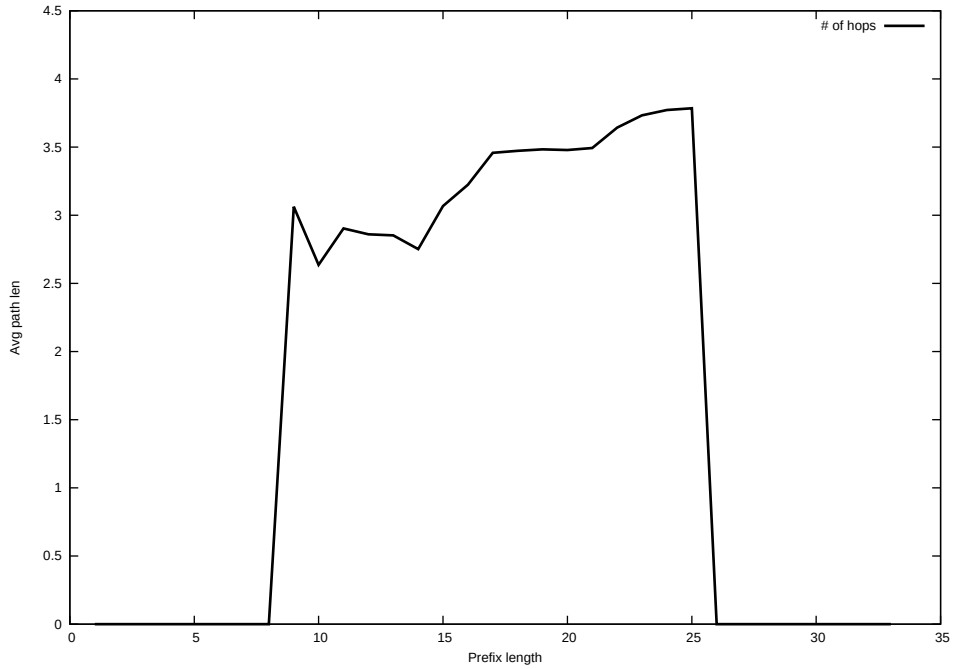
2013-09-22



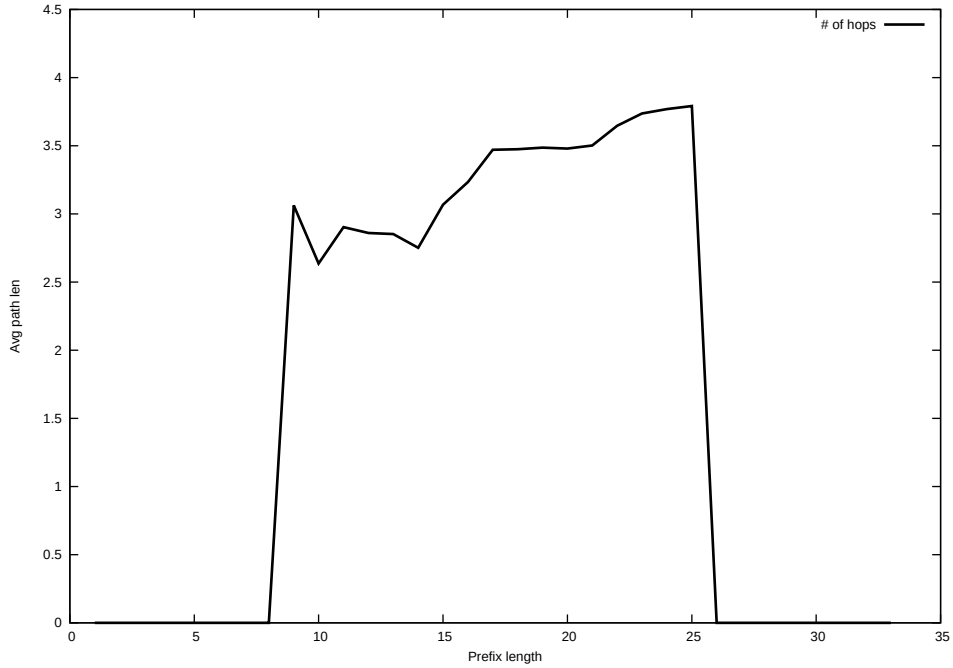
2013-09-23



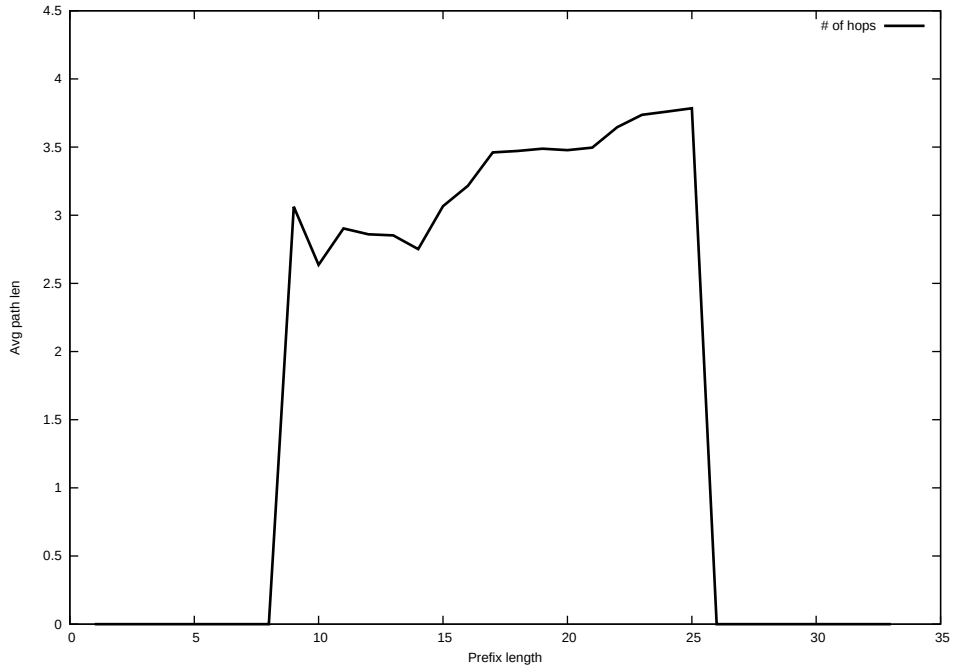
2013-09-24



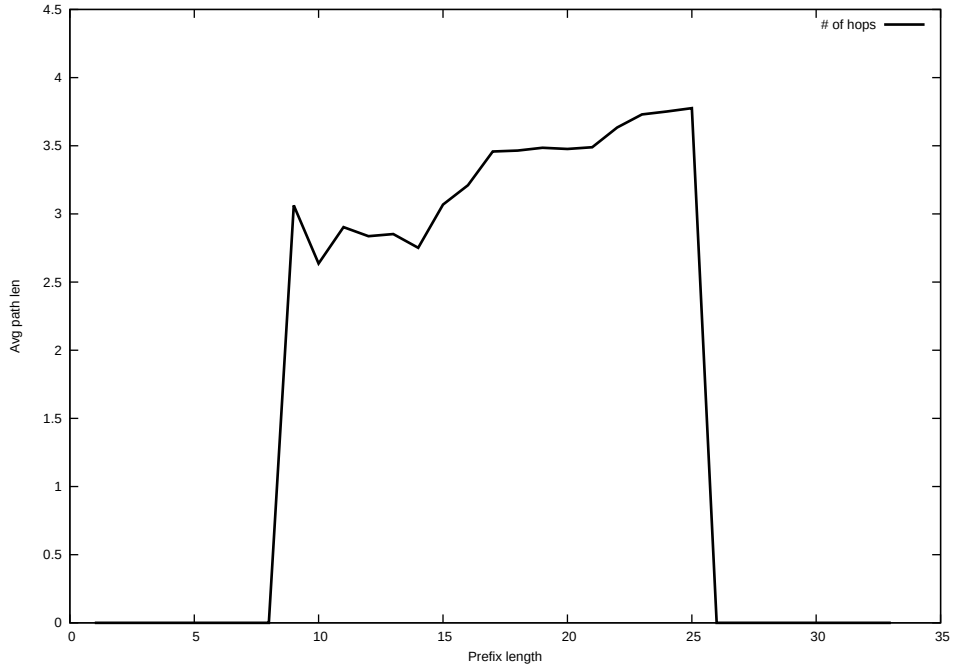
2013-09-25



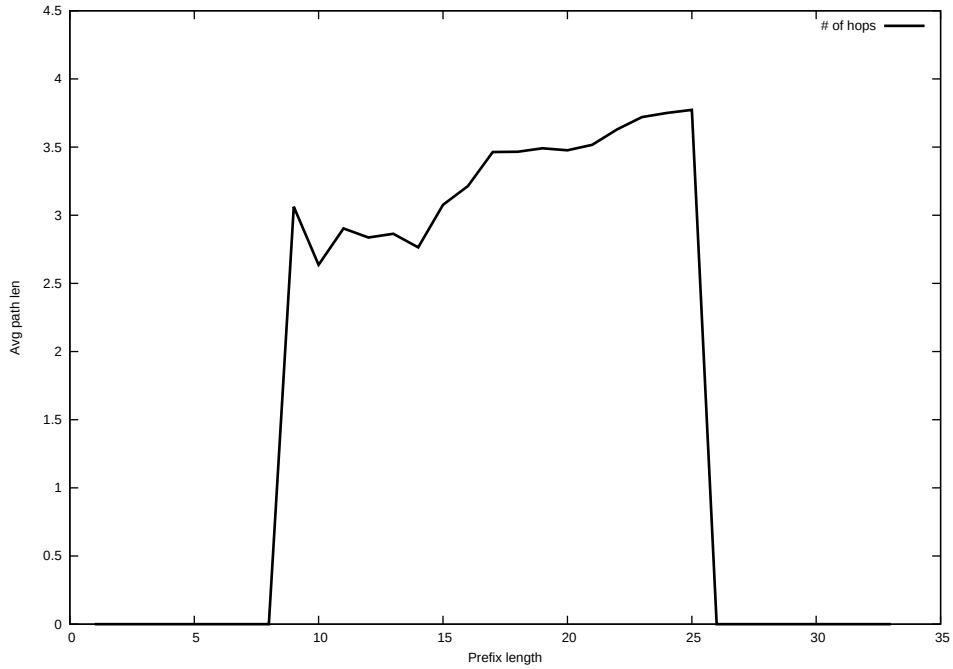
2013-09-26



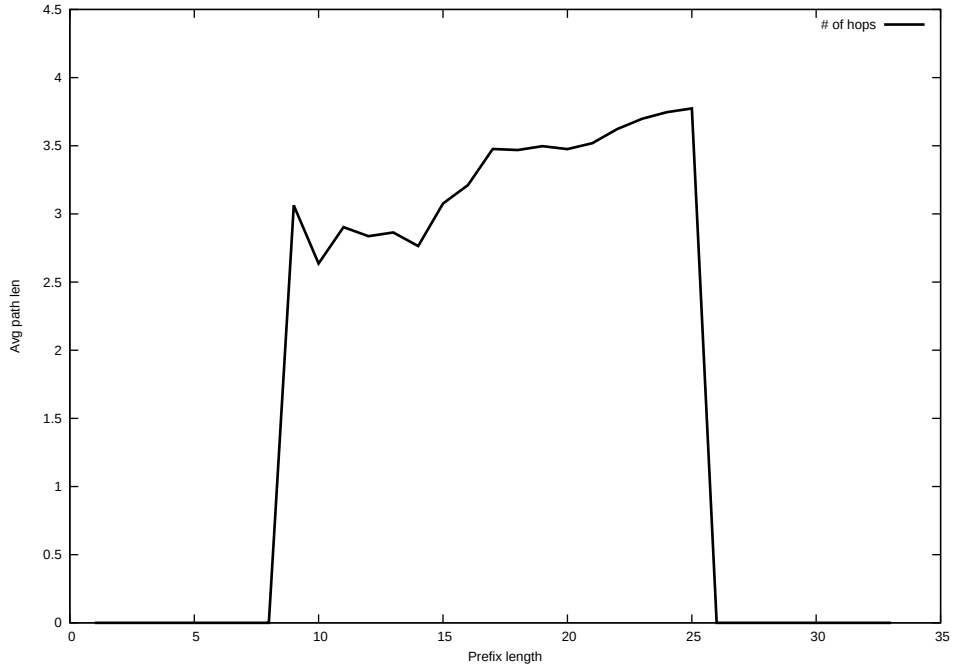
2013-09-27



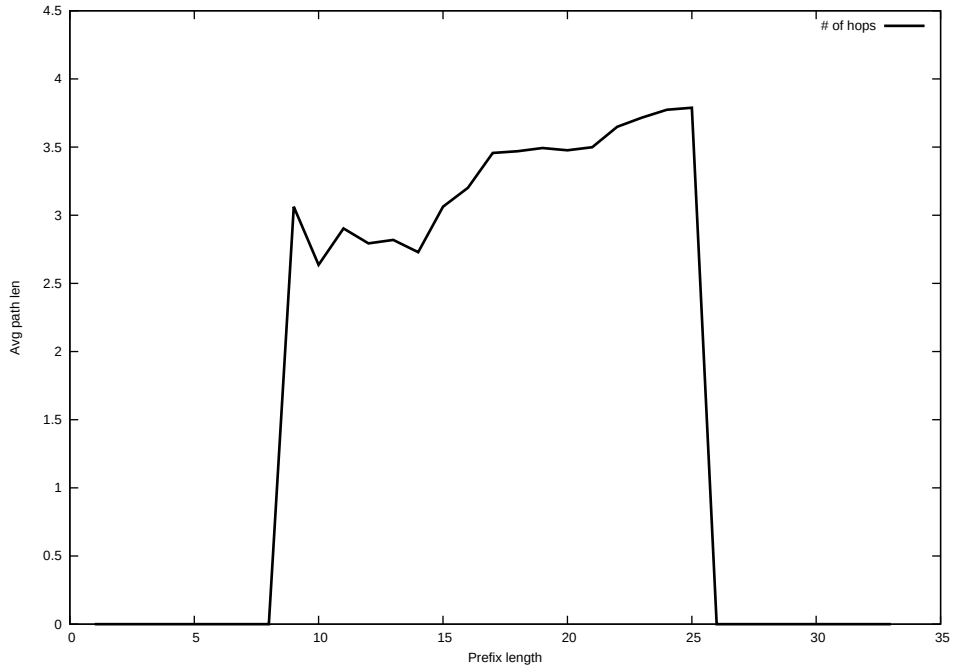
2013-09-28



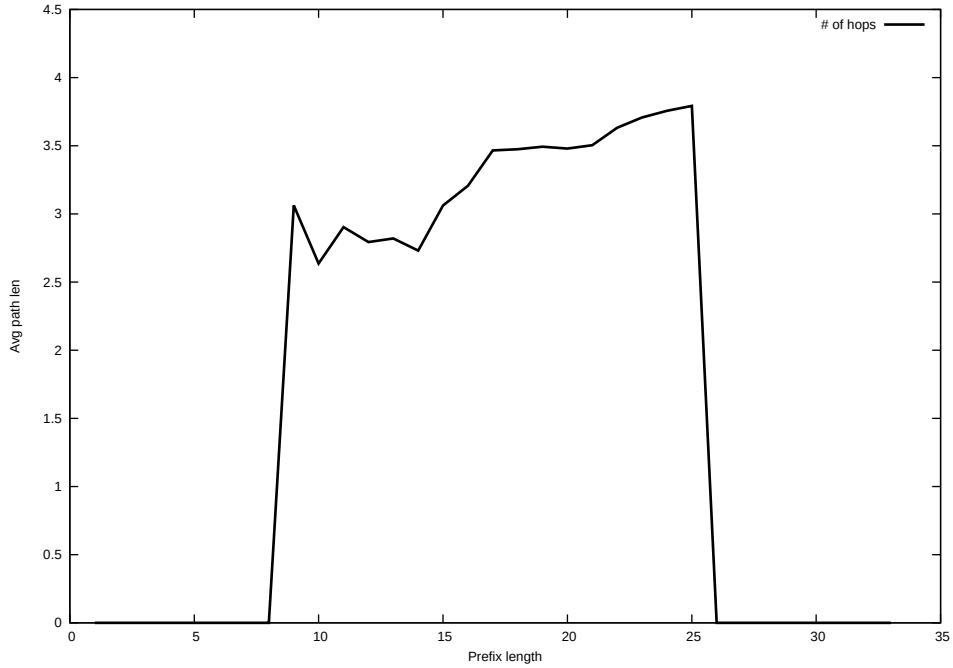
2013-09-29



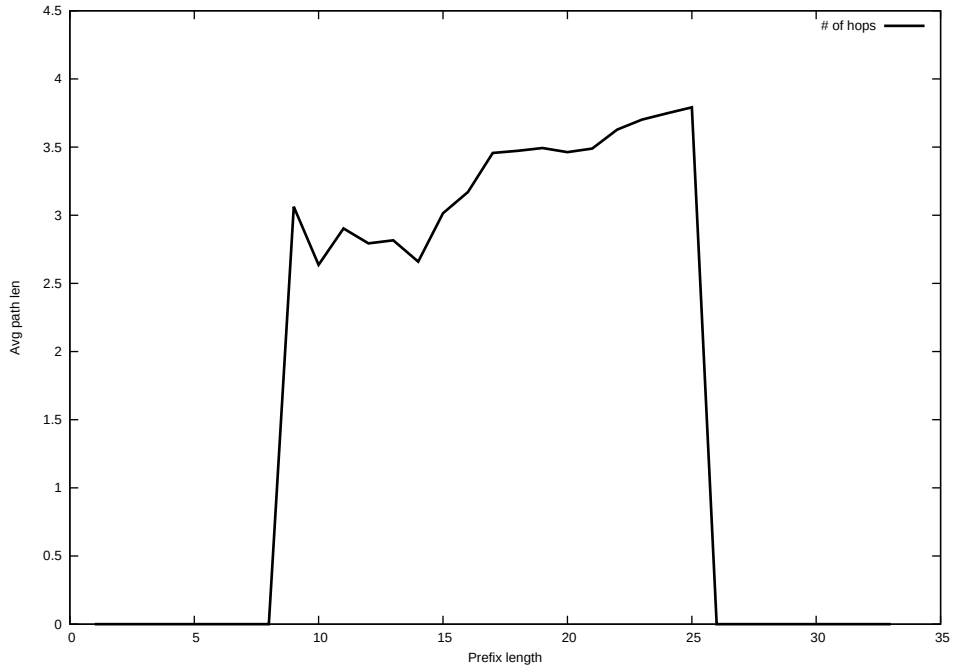
2013-09-30



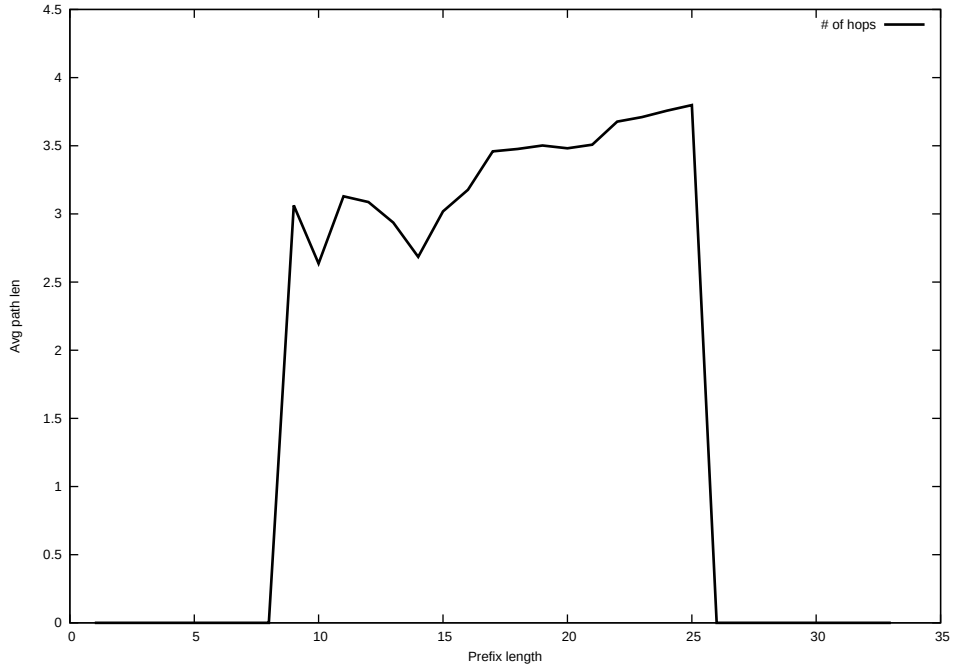
2013-10-01



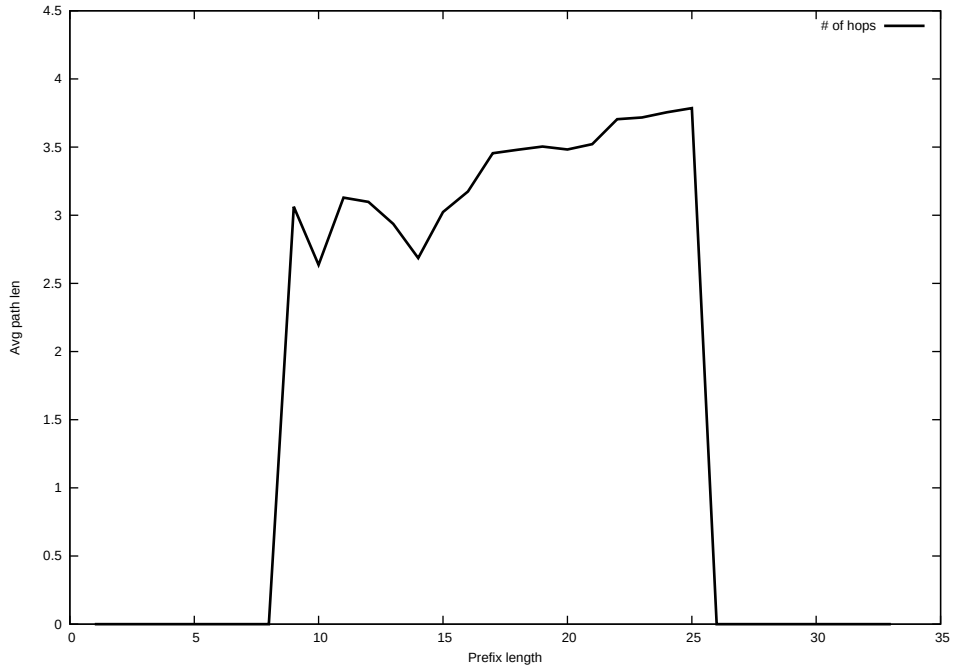
2013-10-02



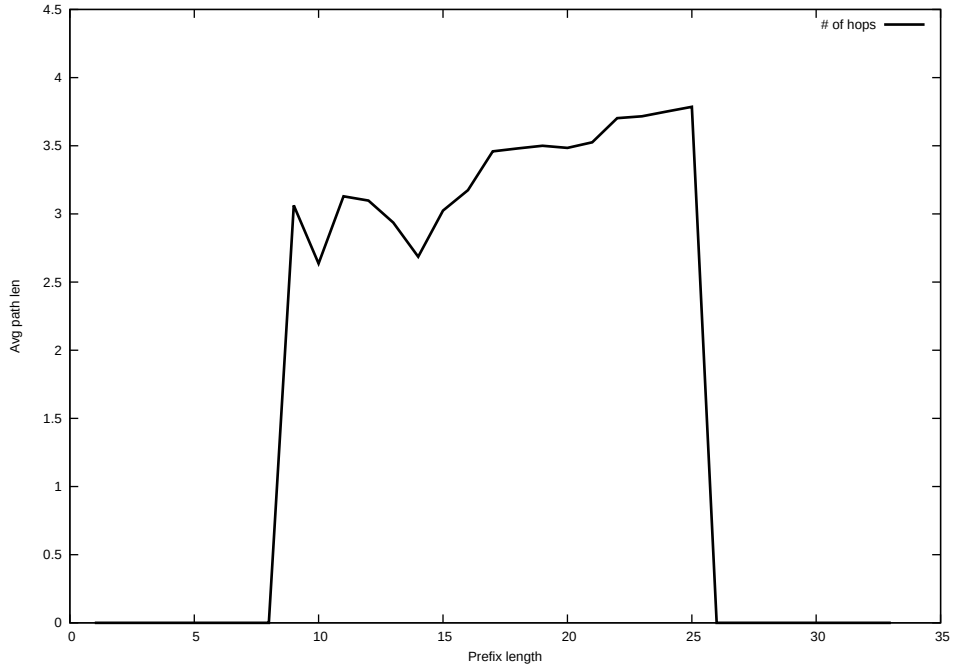
2013-10-03



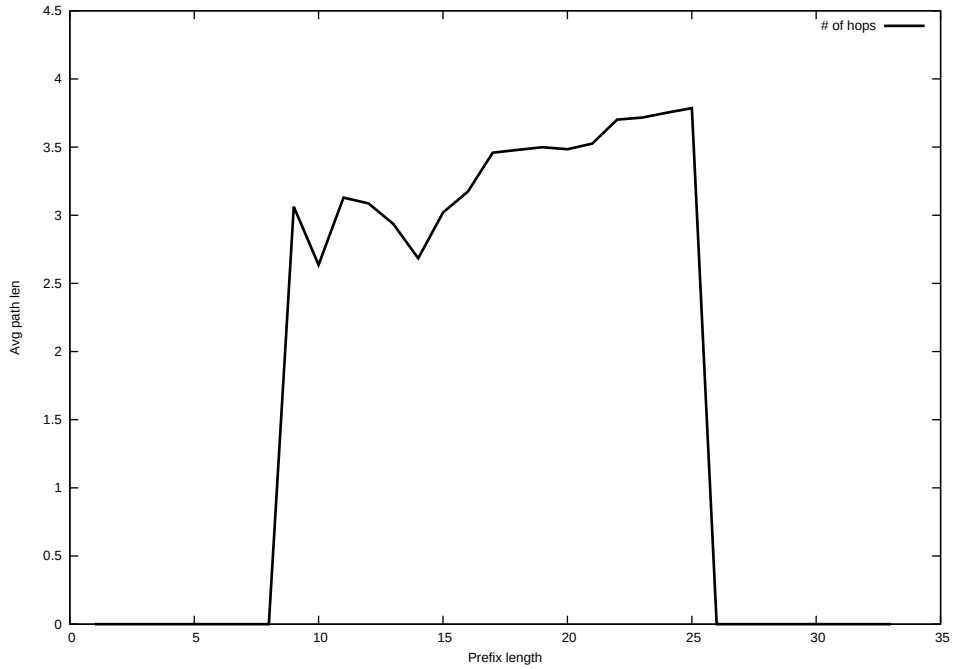
2013-10-04



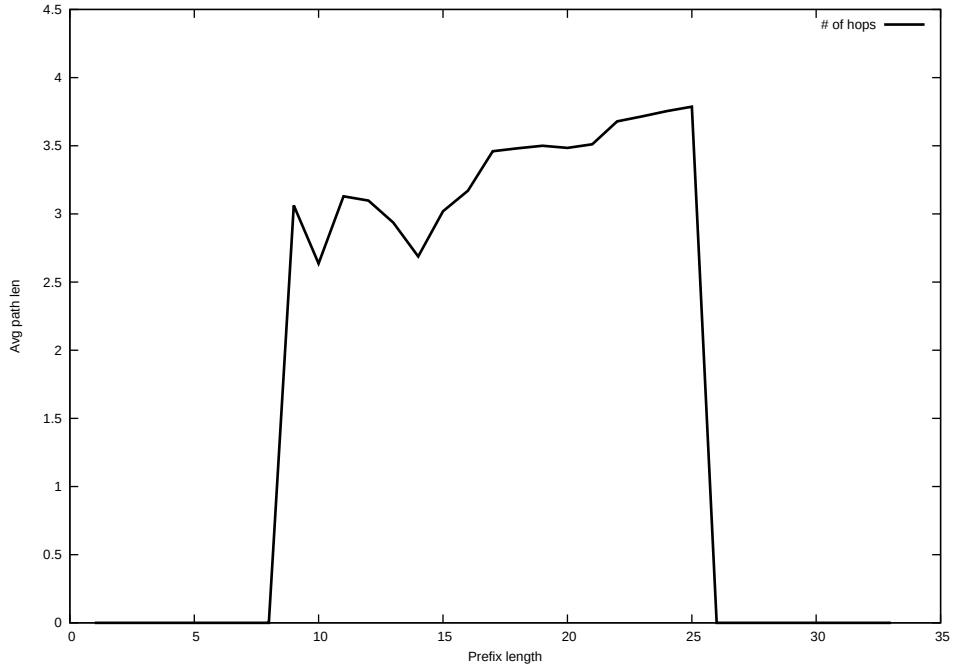
2013-10-05



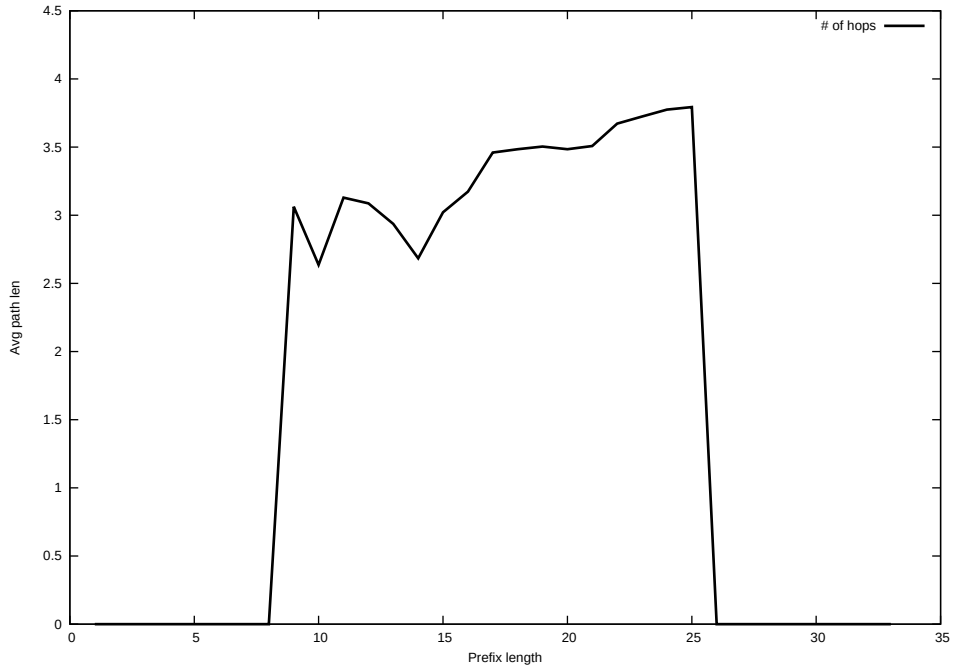
2013-10-06



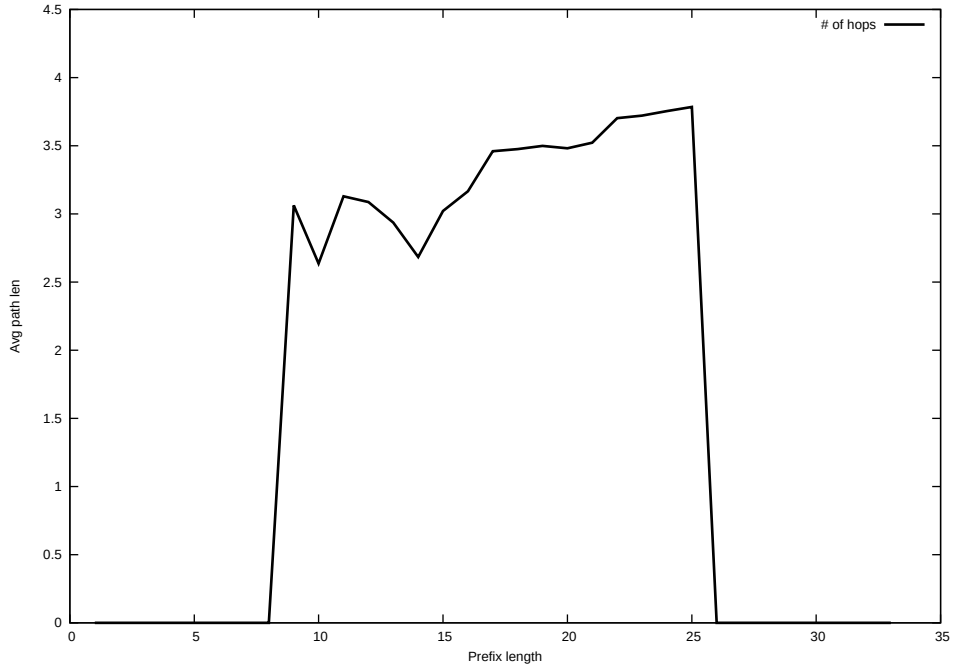
2013-10-07



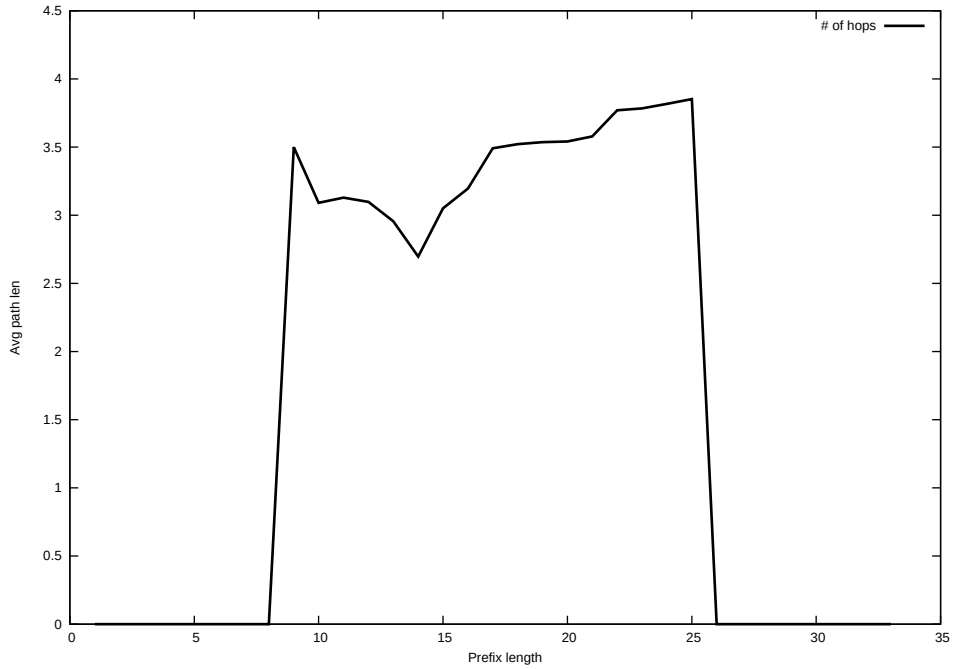
2013-10-08



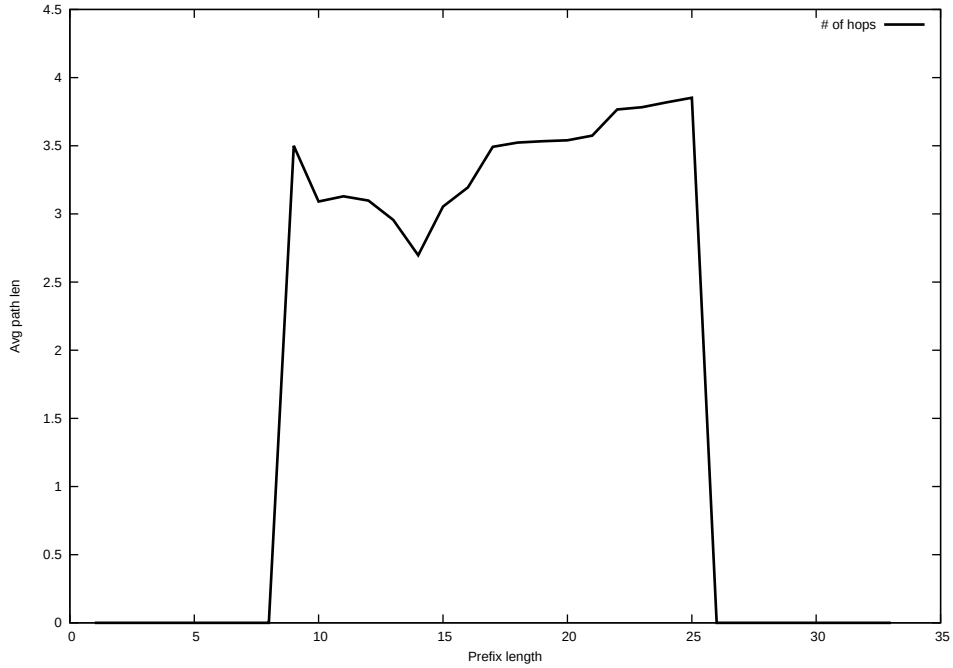
2013-10-09



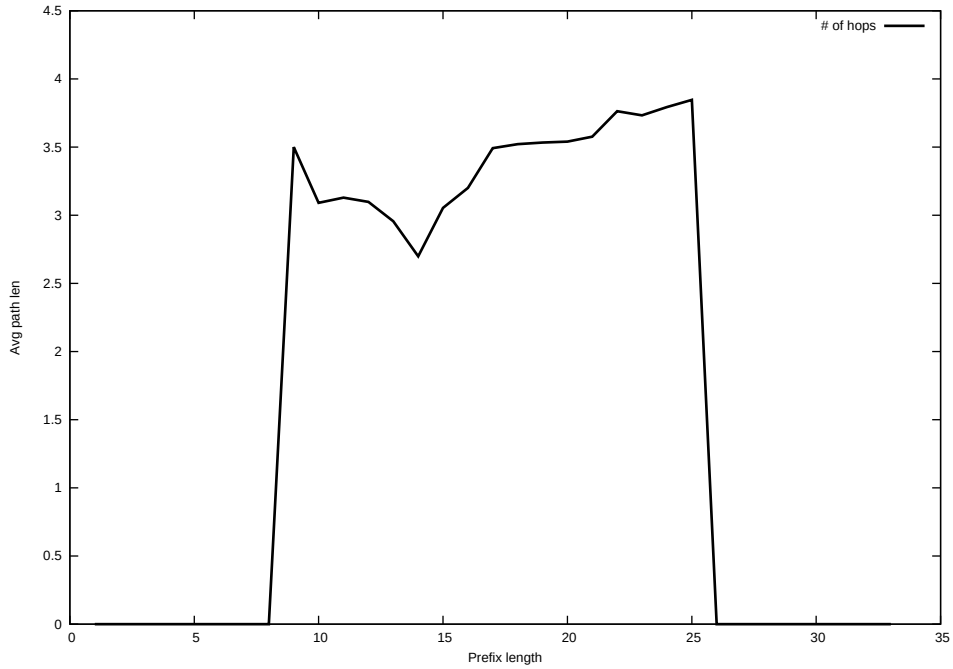
2013-10-10



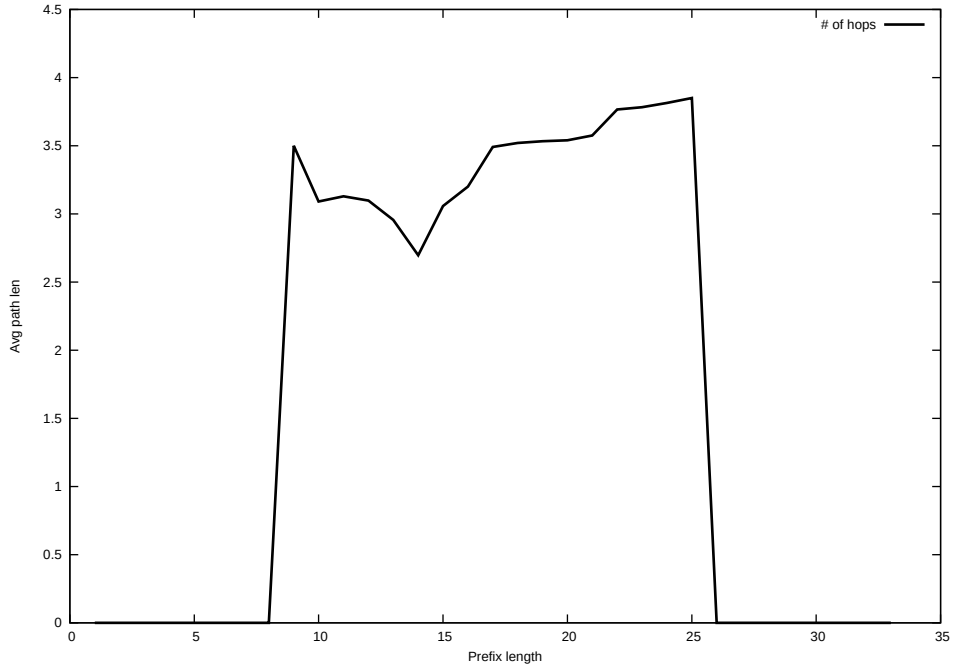
2013-10-11



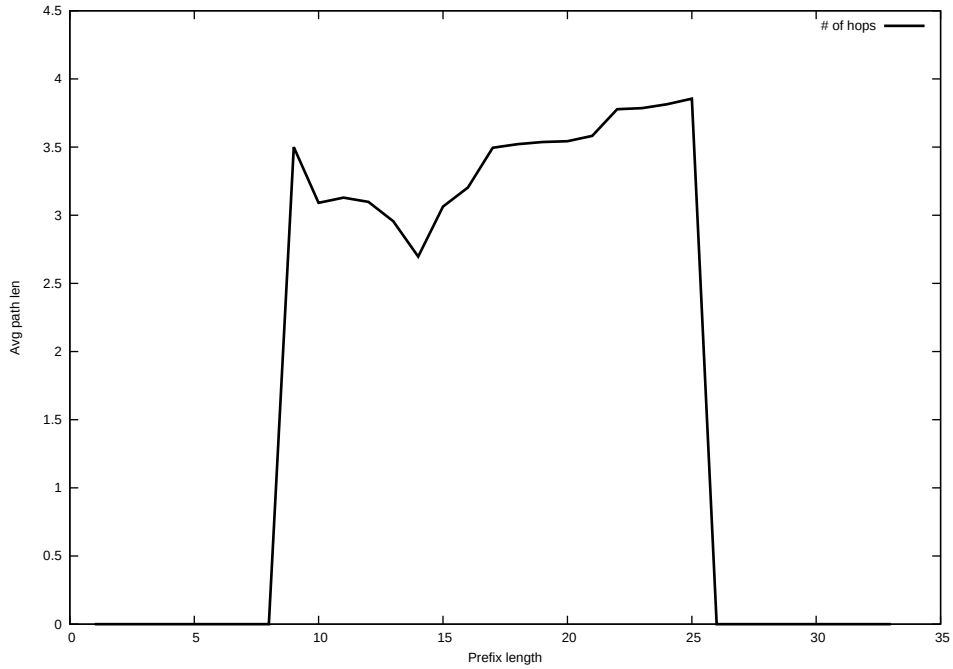
2013-10-12



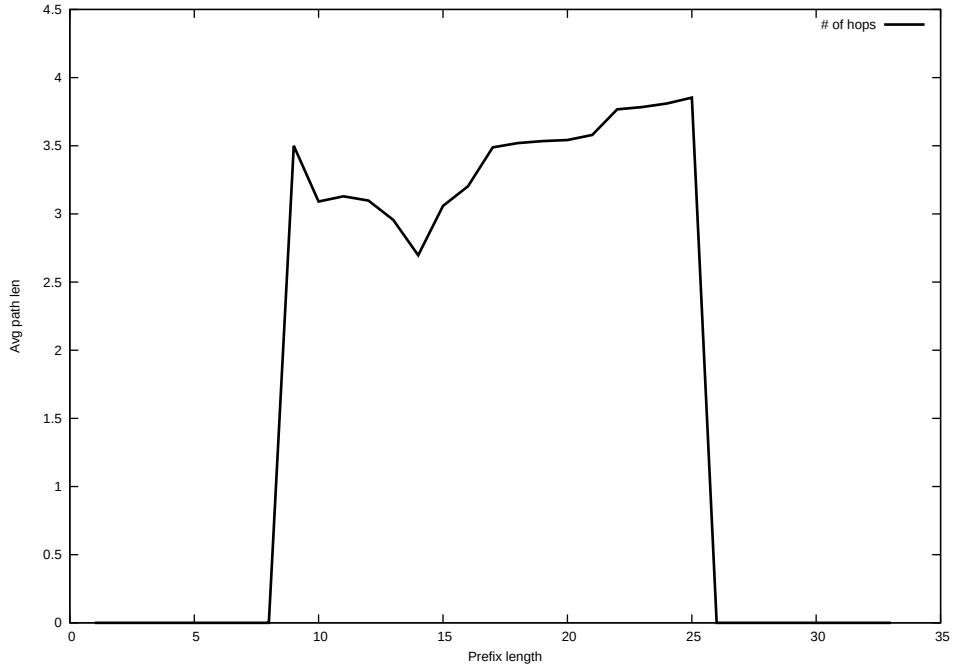
2013-10-13



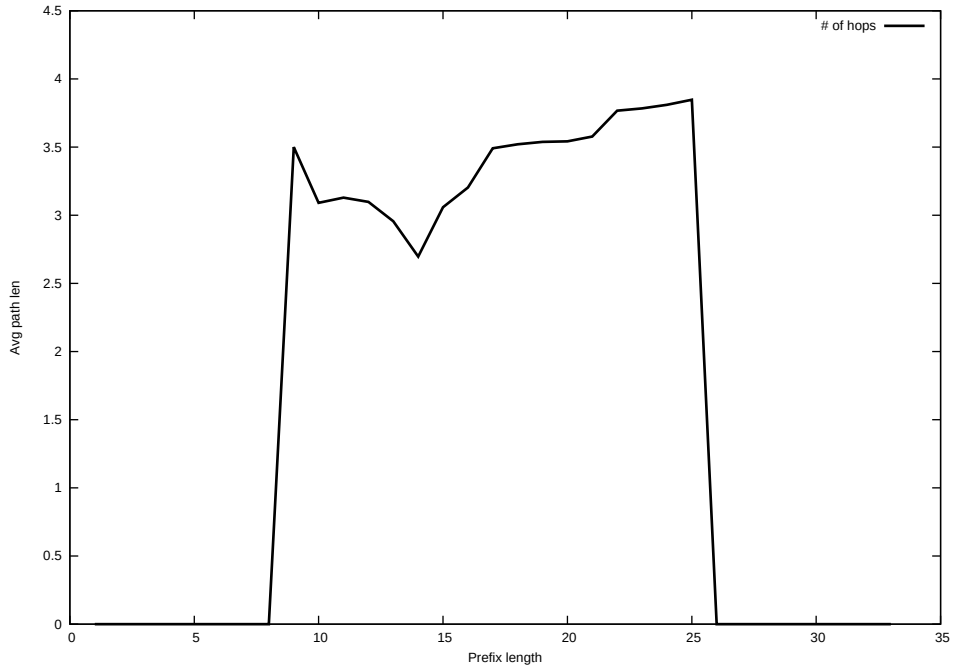
2013-10-14



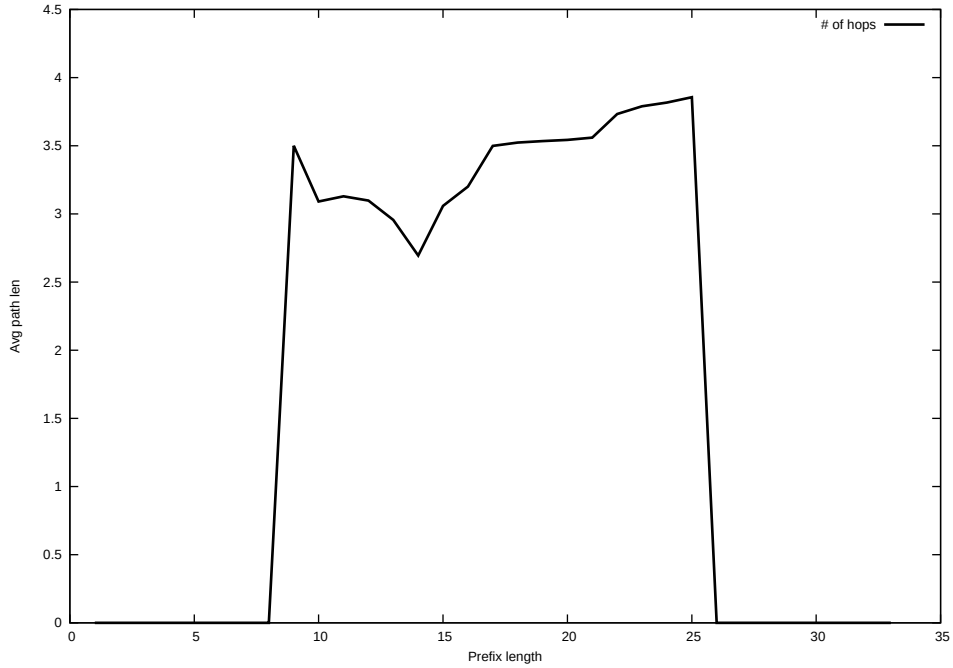
2013-10-15



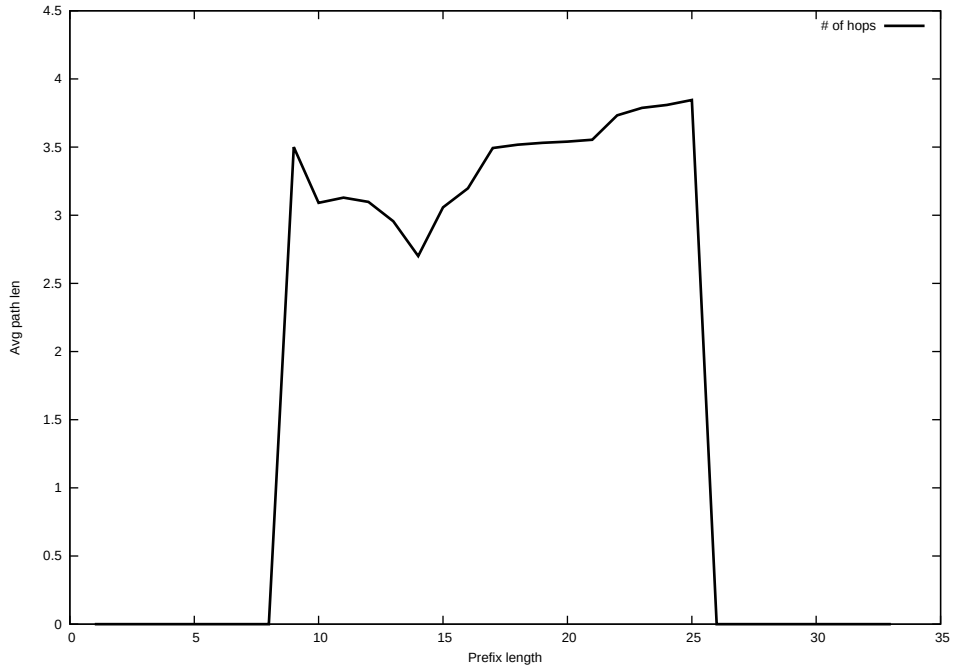
2013-10-16



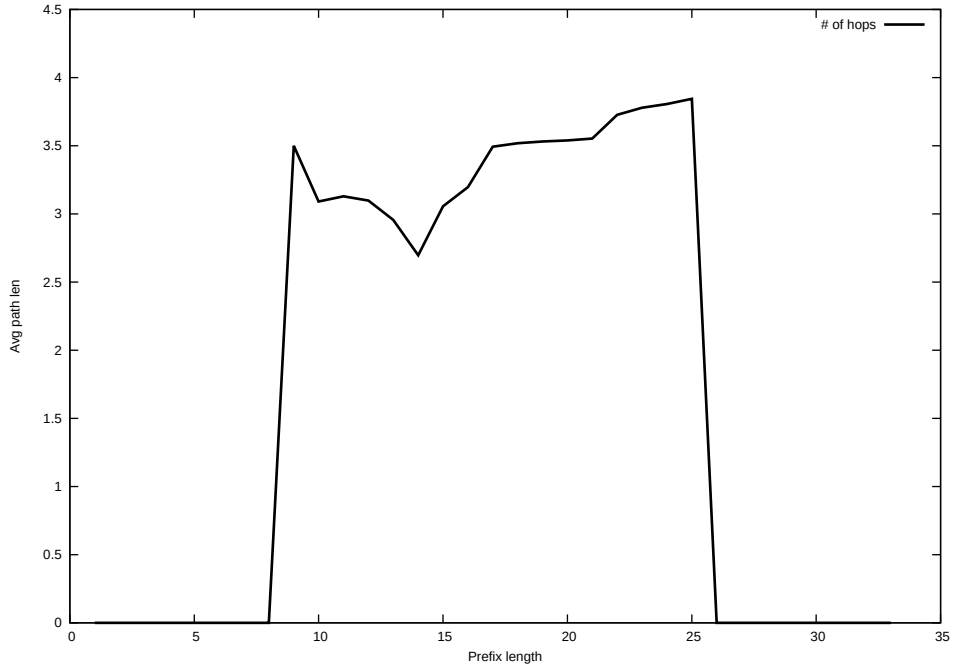
2013-10-17



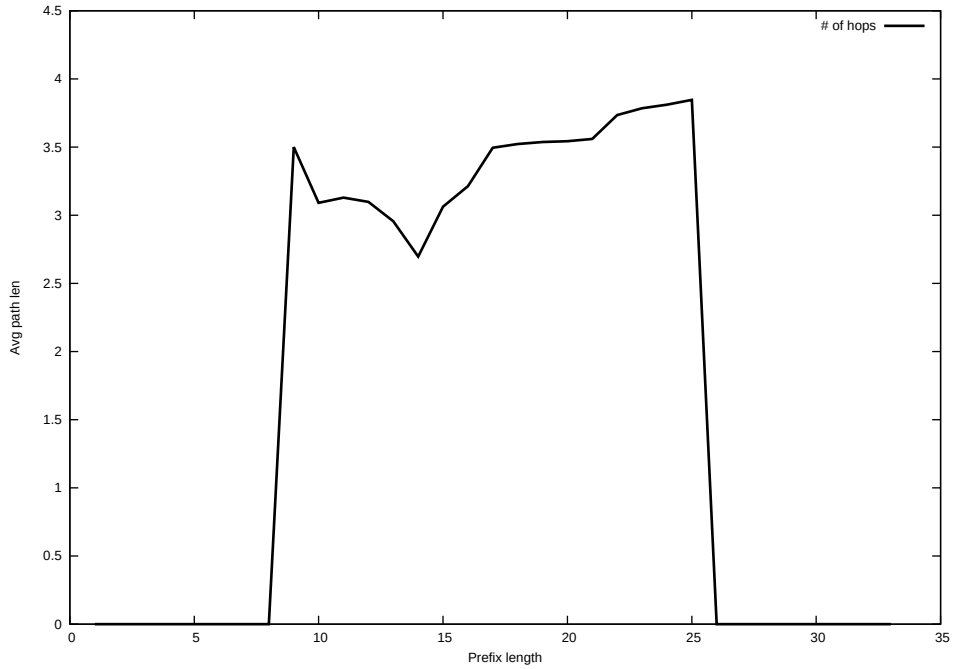
2013-10-18



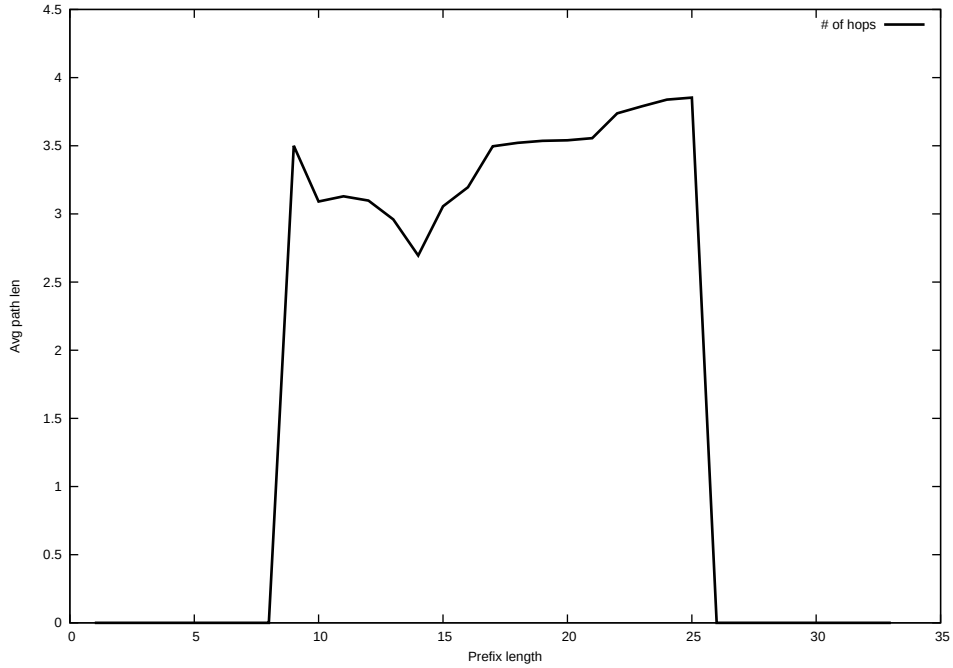
2013-10-19



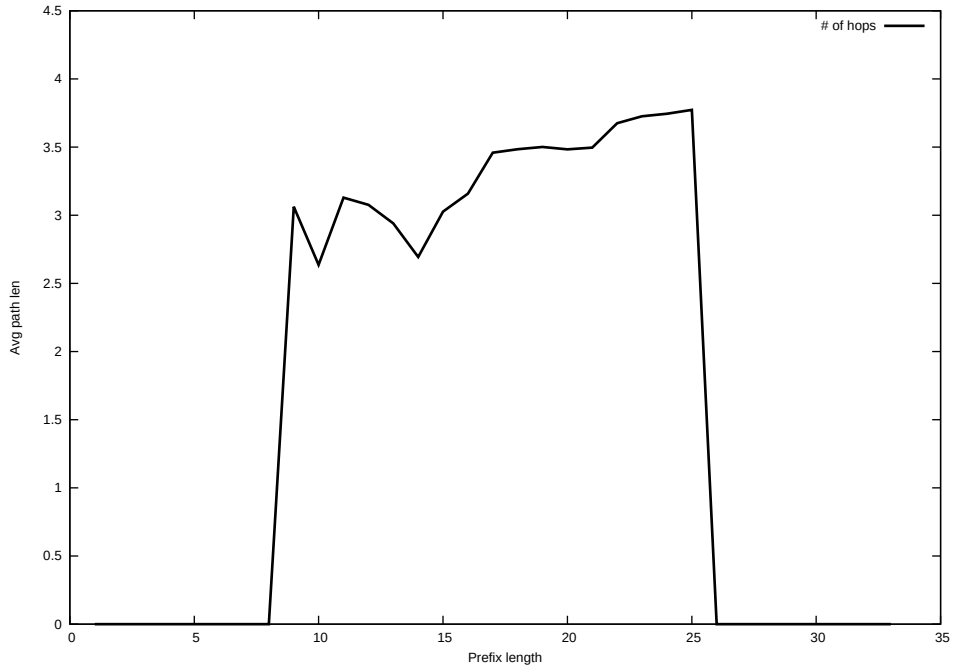
2013-10-20



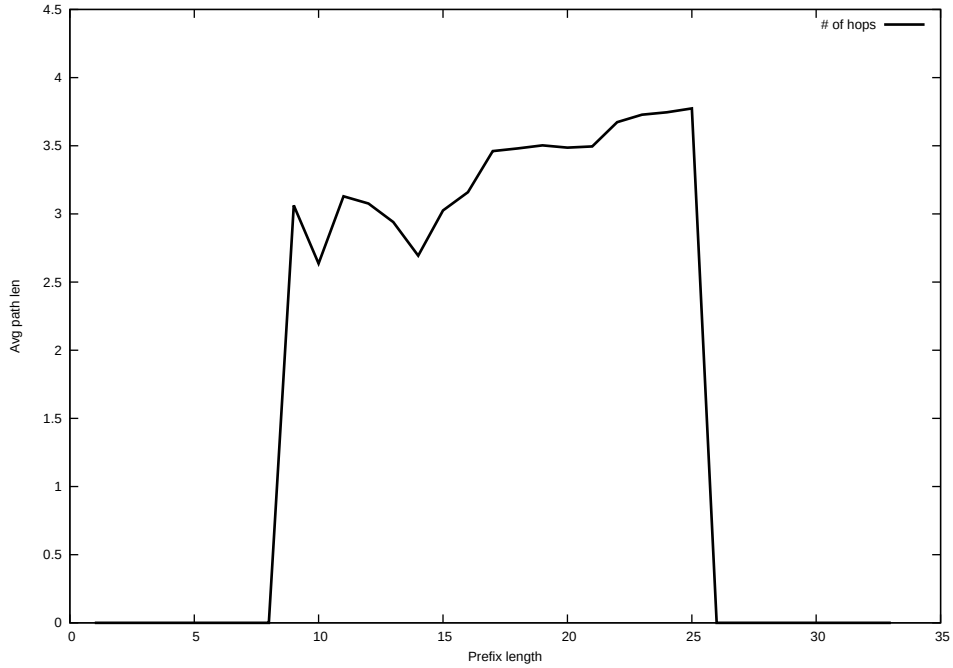
2013-10-21



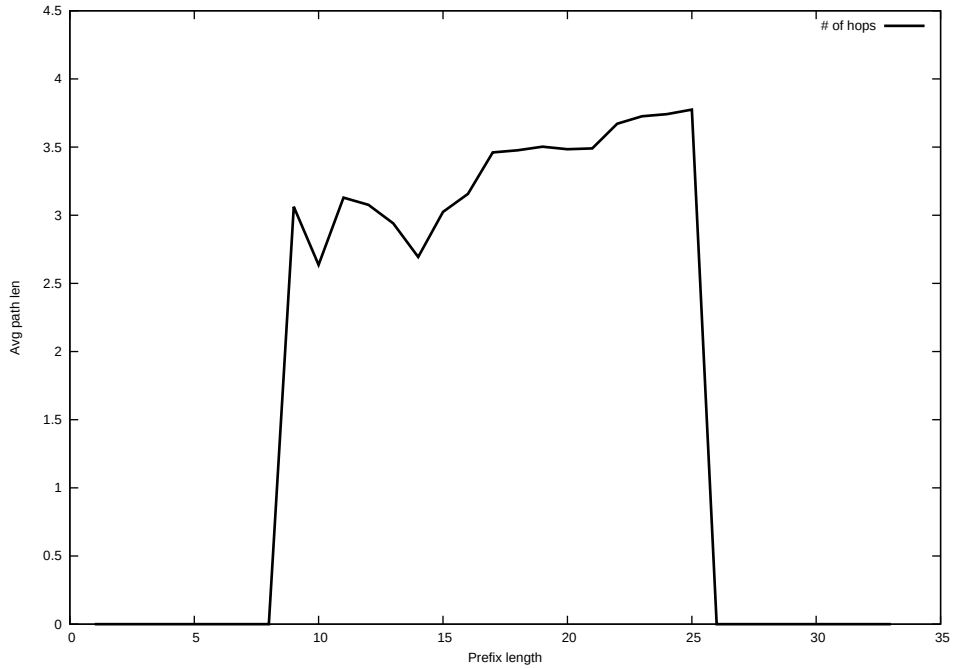
2013-10-22



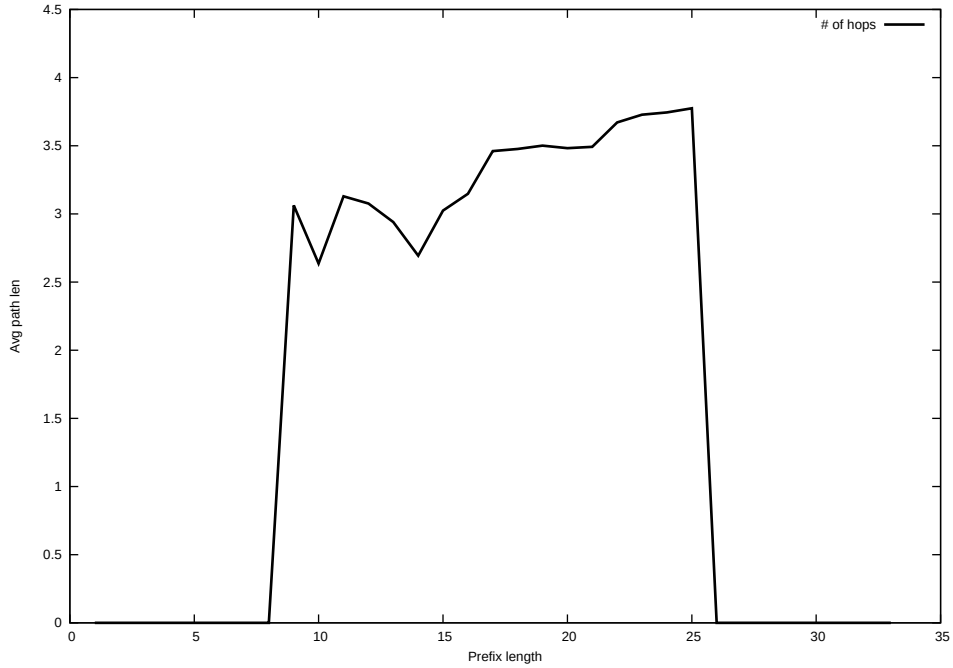
2013-10-23



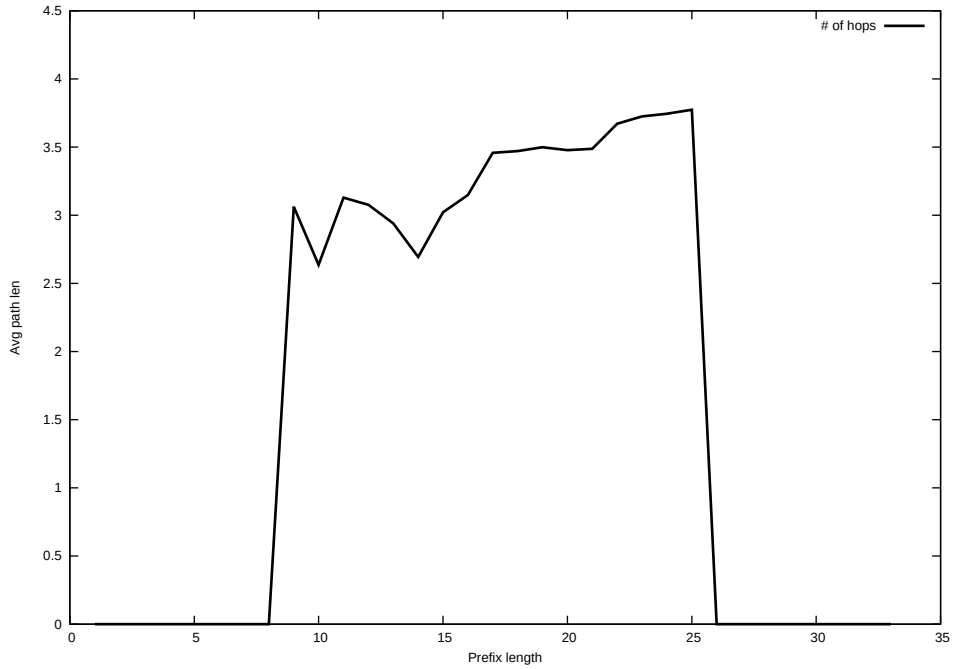
2013-10-24



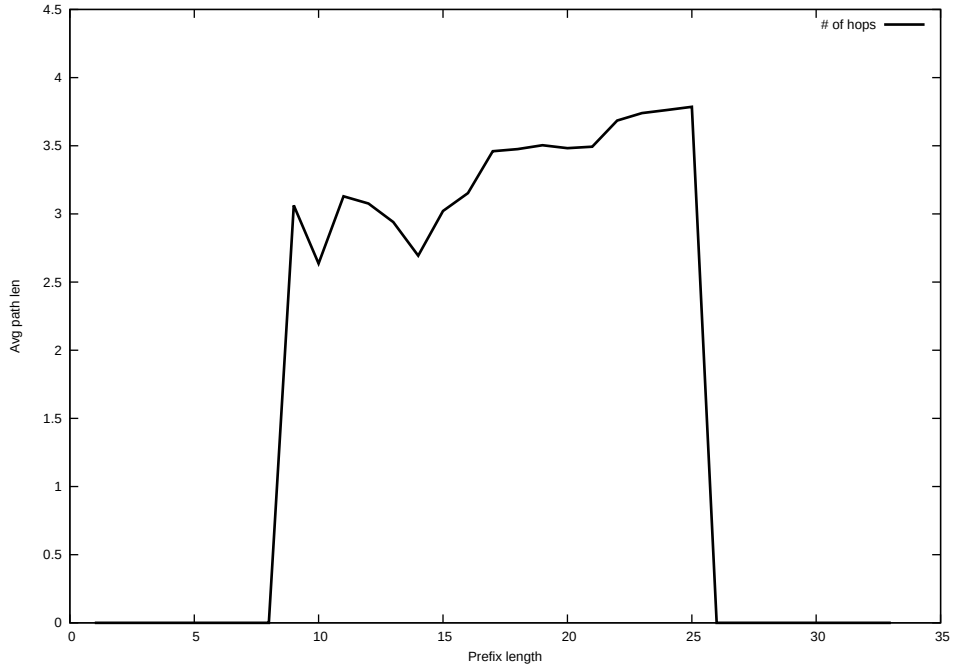
2013-10-25



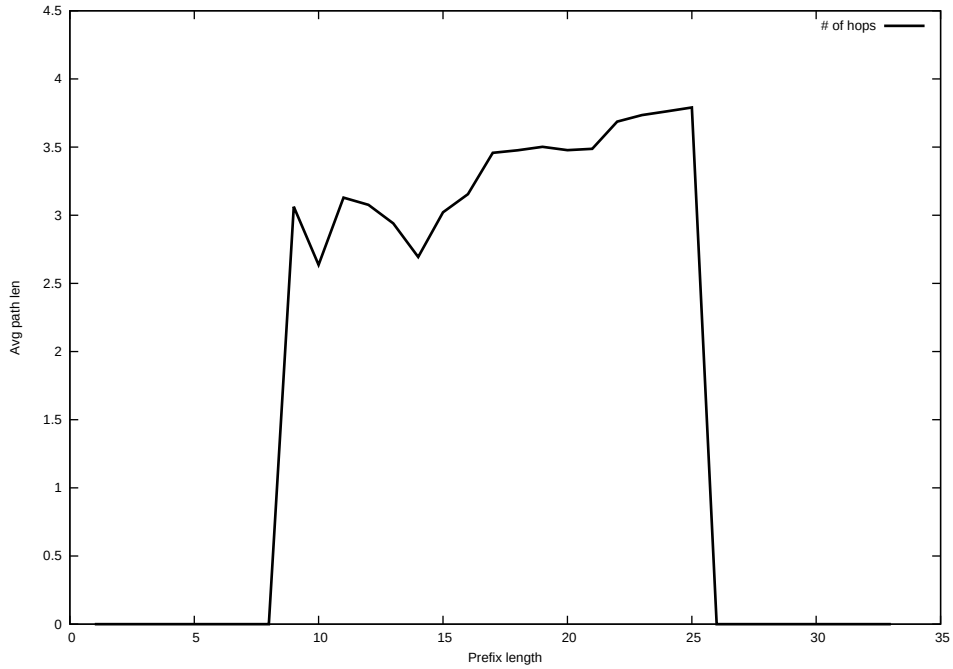
2013-10-26



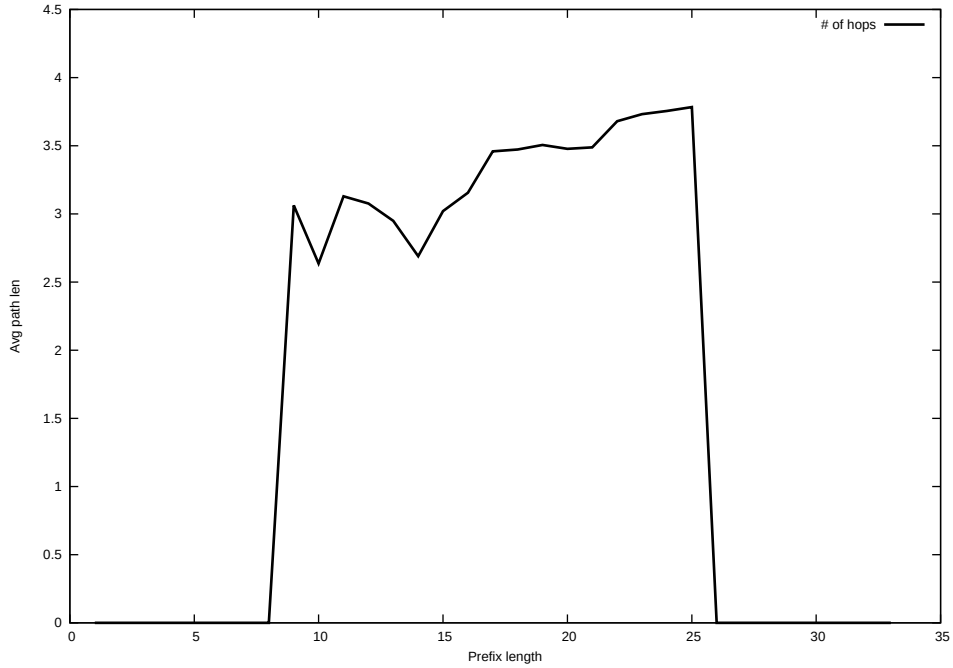
2013-10-27



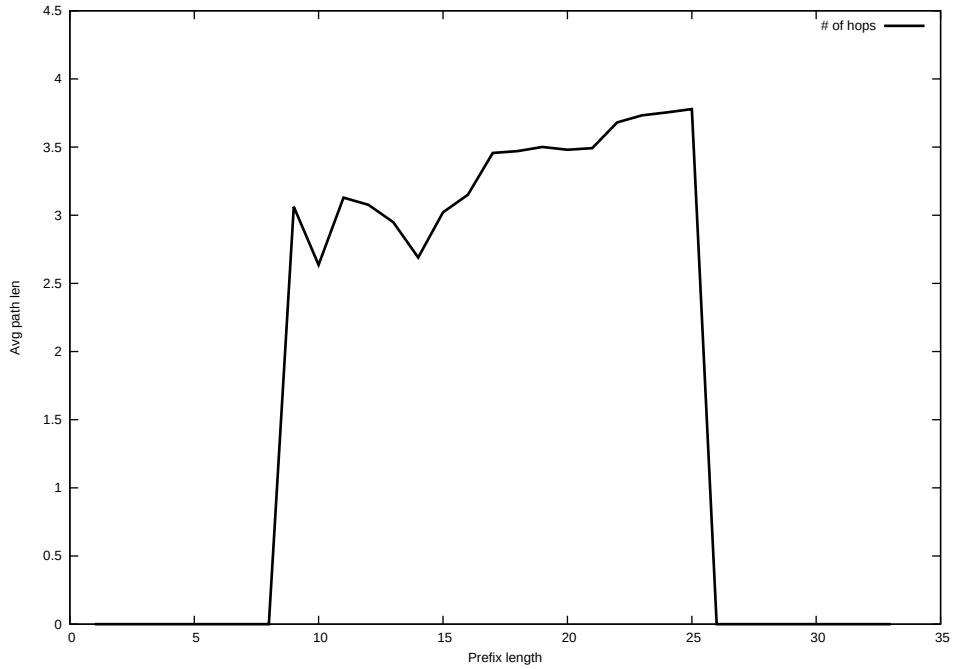
2013-10-28



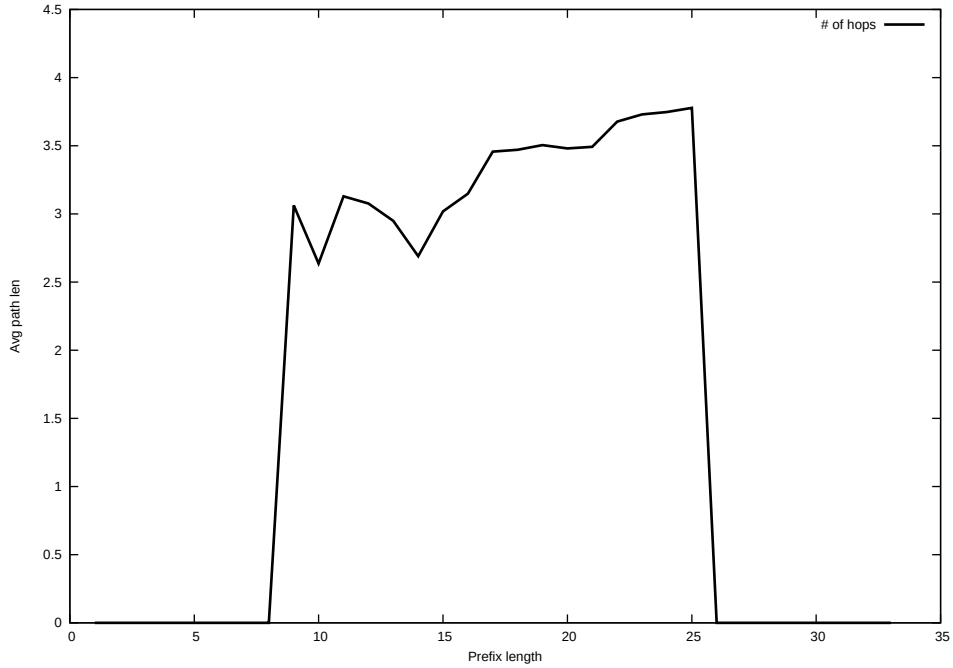
2013-10-29



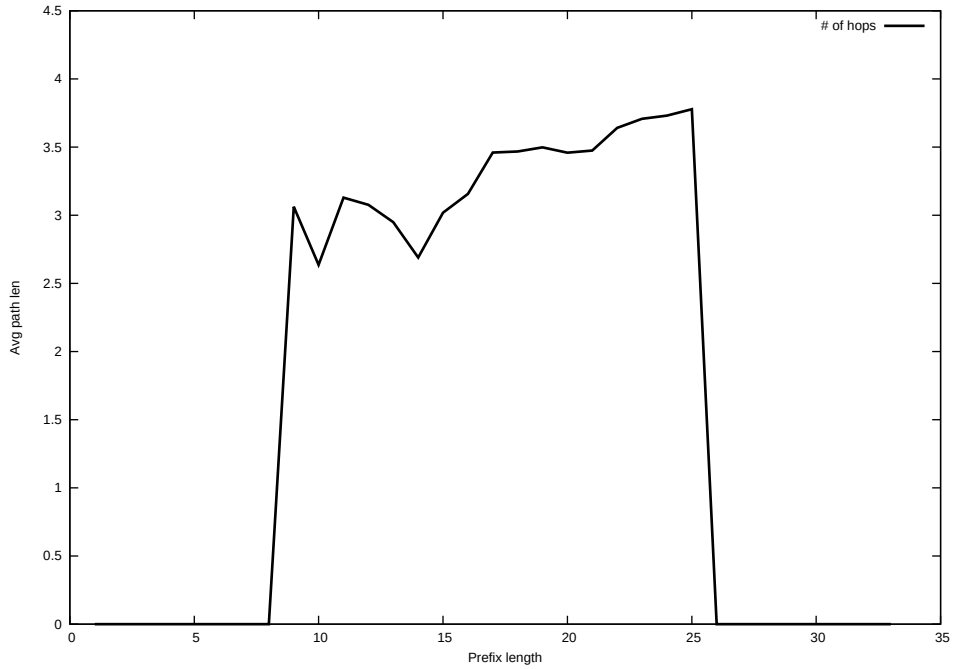
2013-10-30



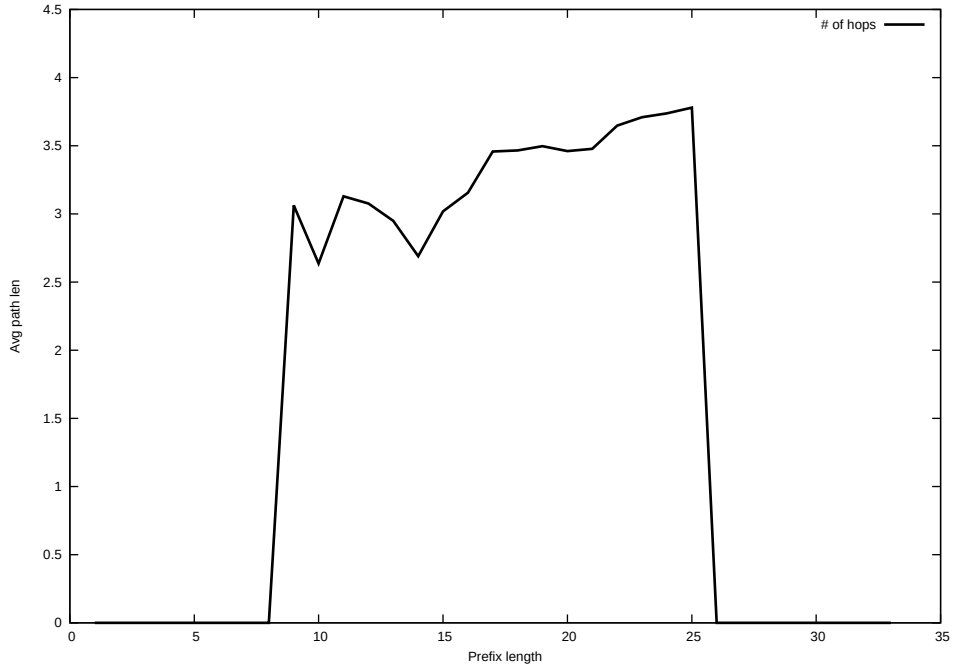
2013-10-31



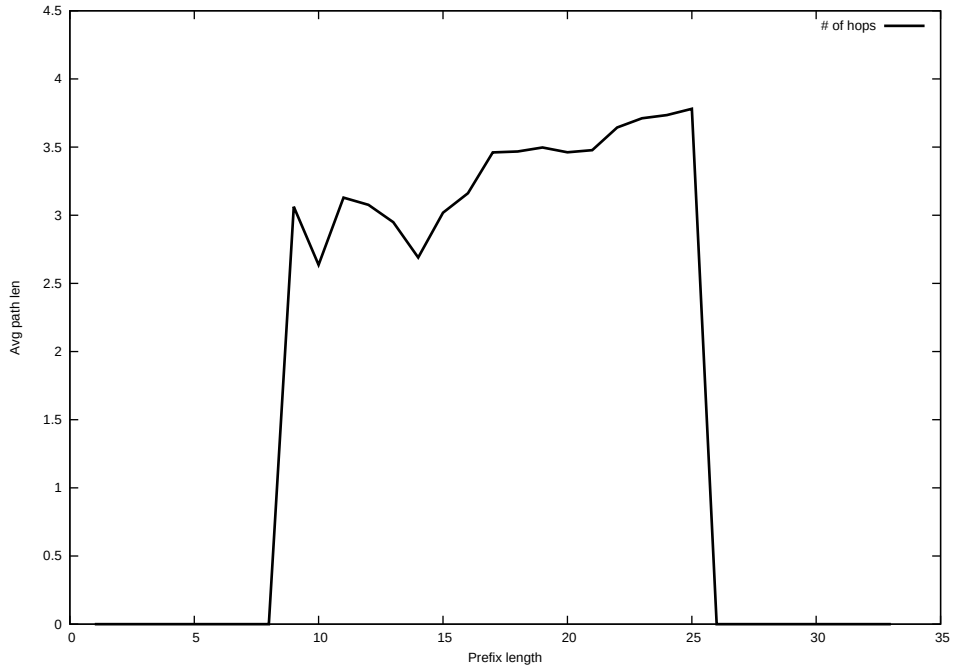
2013-11-01



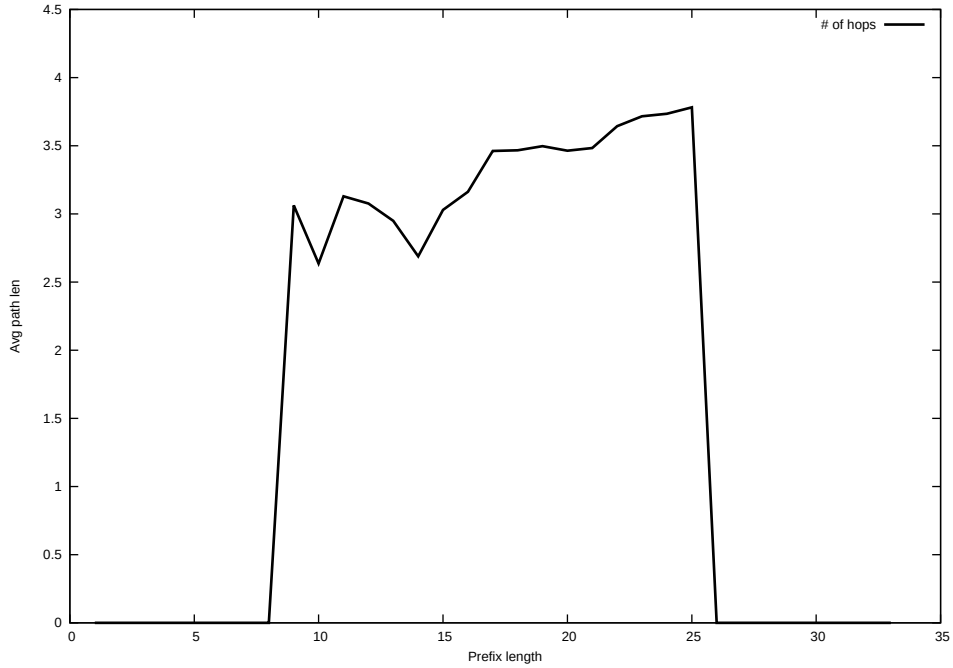
2013-11-02



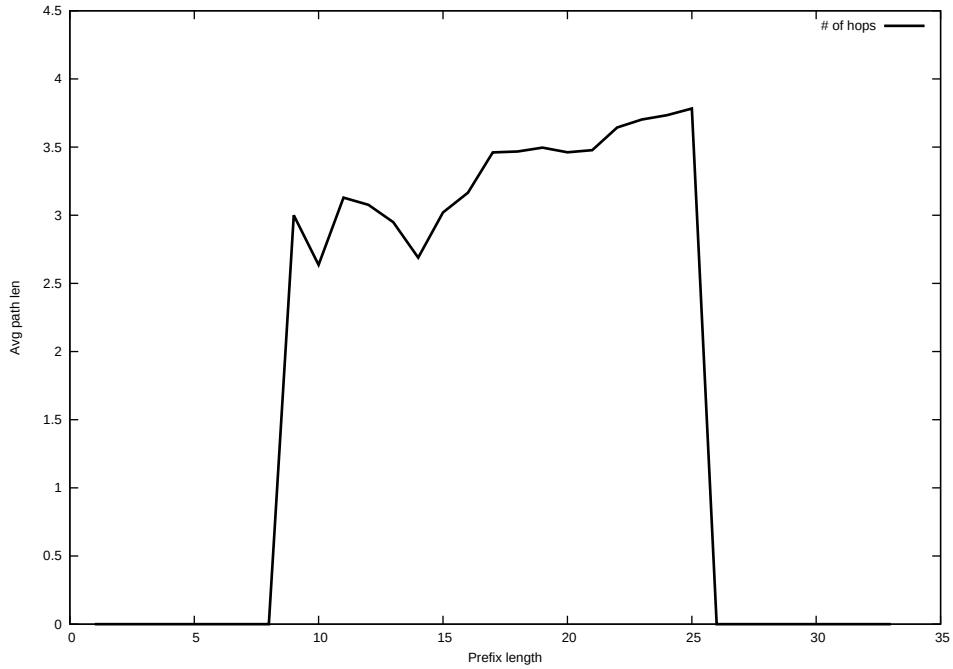
2013-11-03



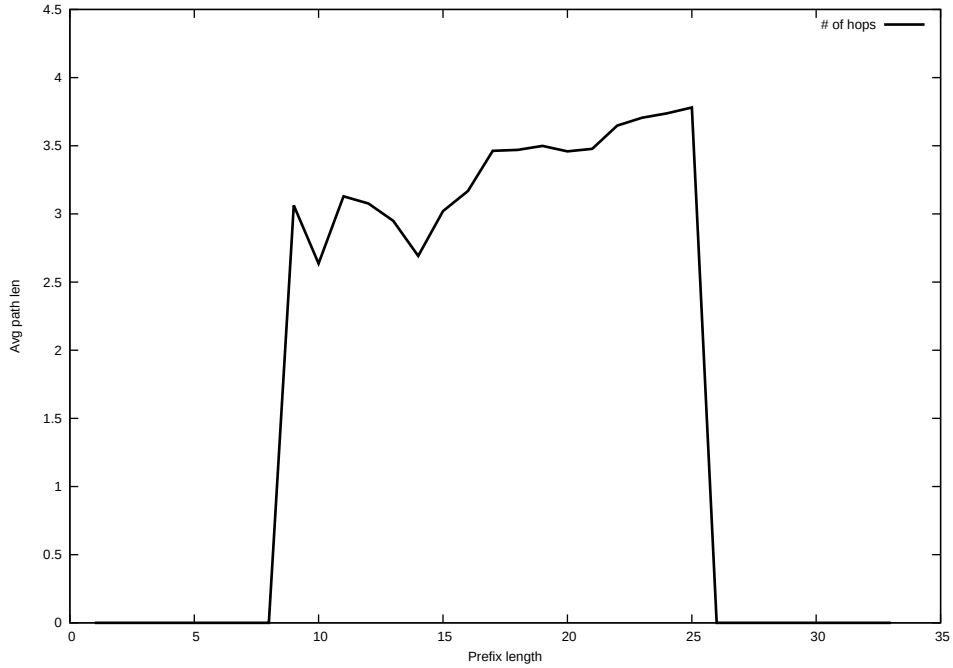
2013-11-04



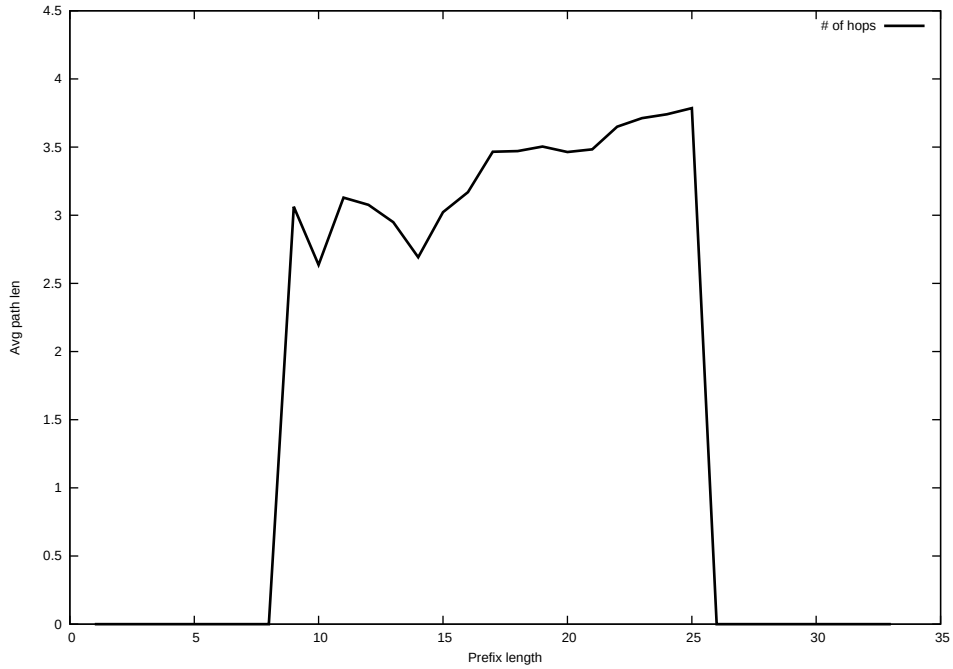
2013-11-05



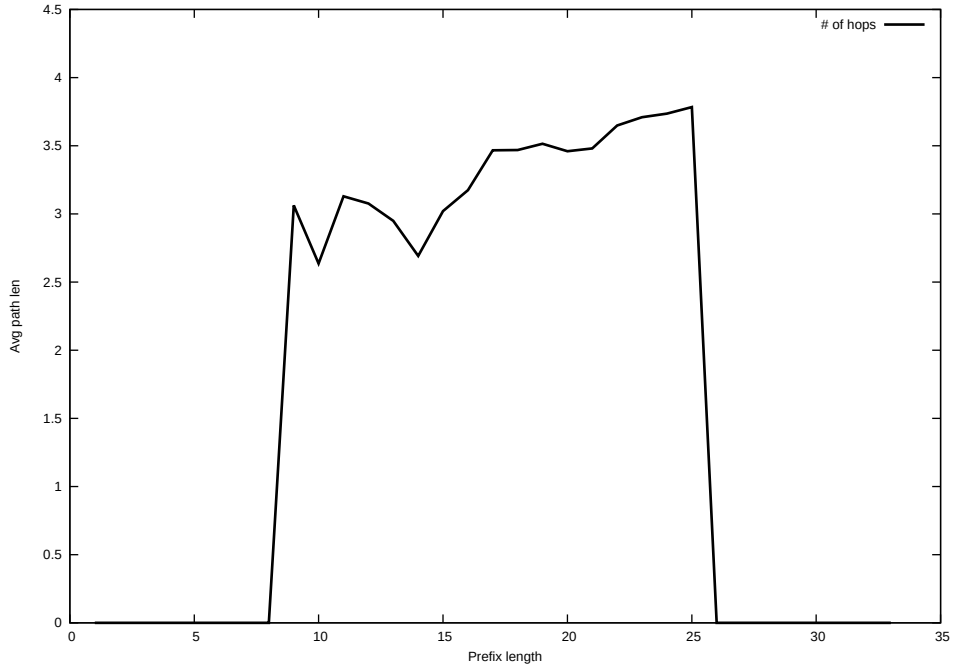
2013-11-06



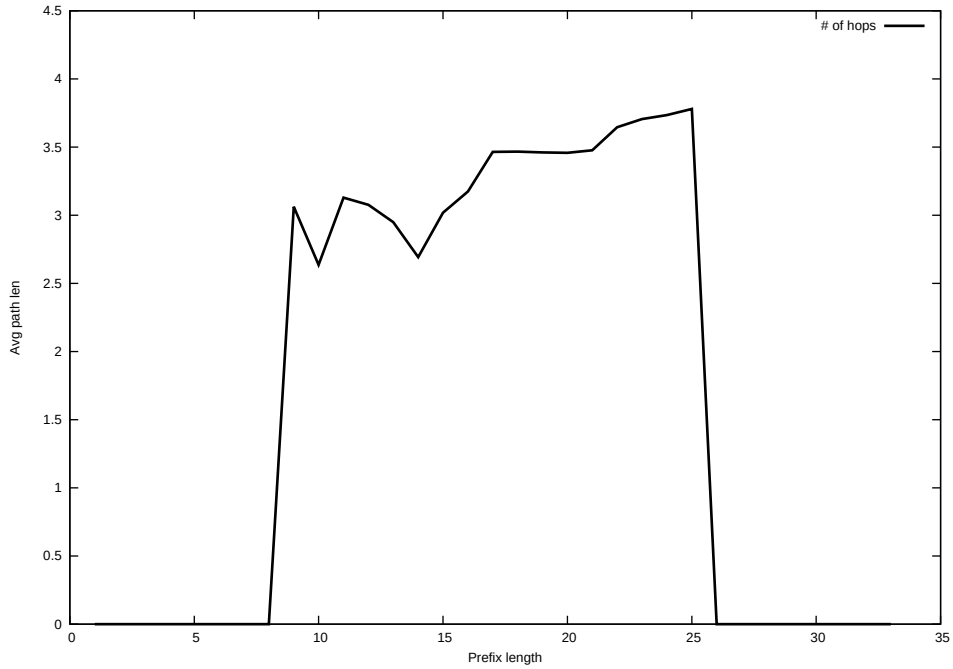
2013-11-07



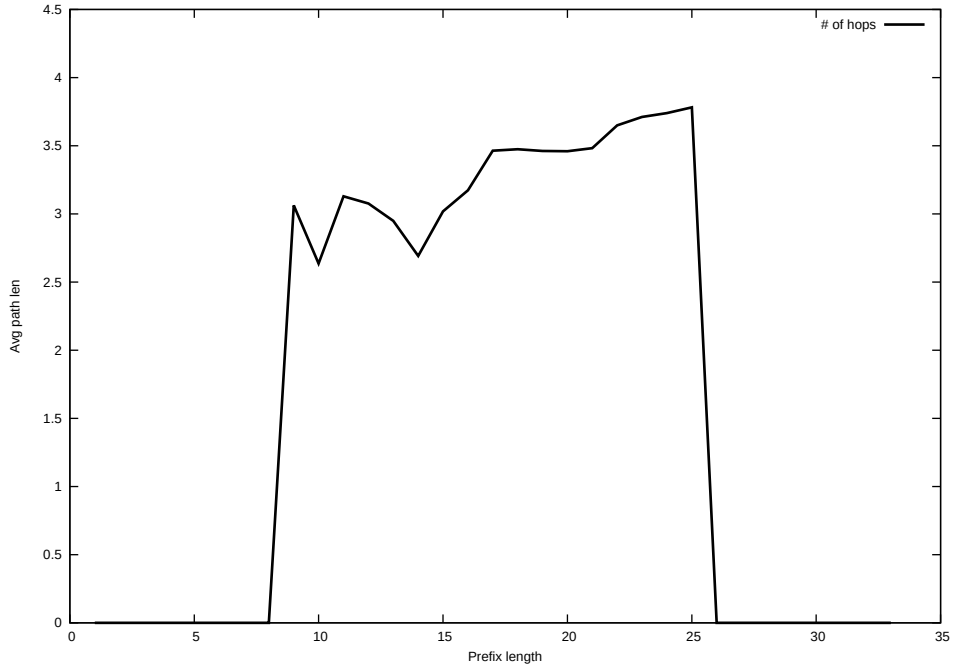
2013-11-08



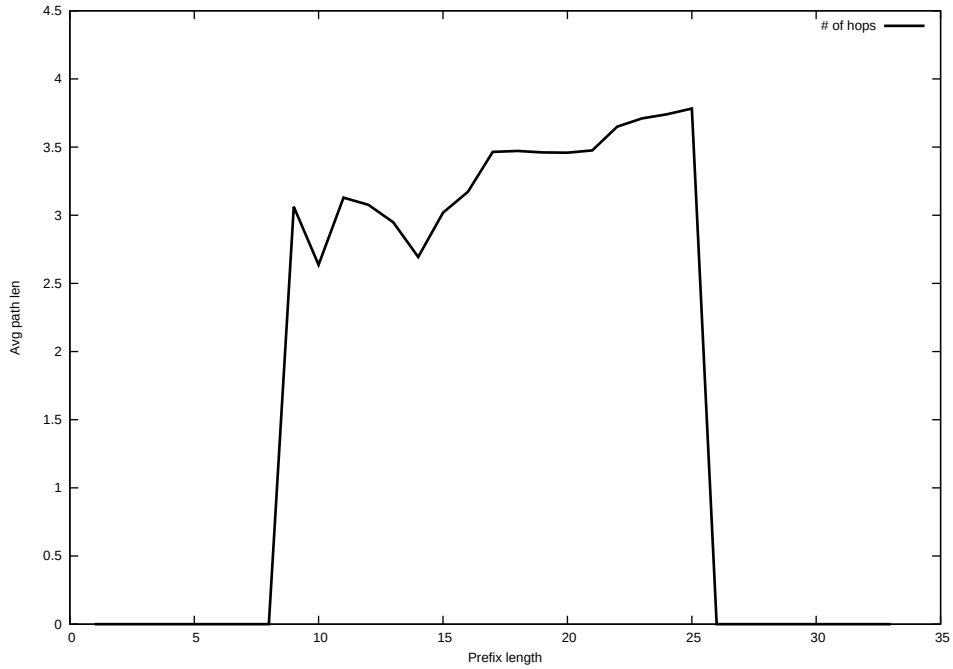
2013-11-09



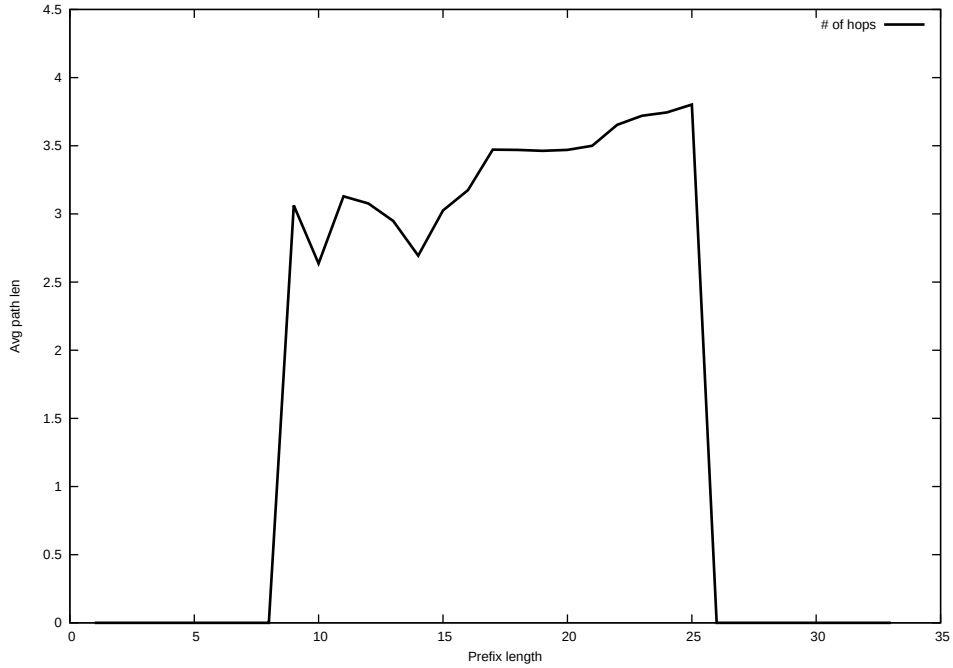
2013-11-10



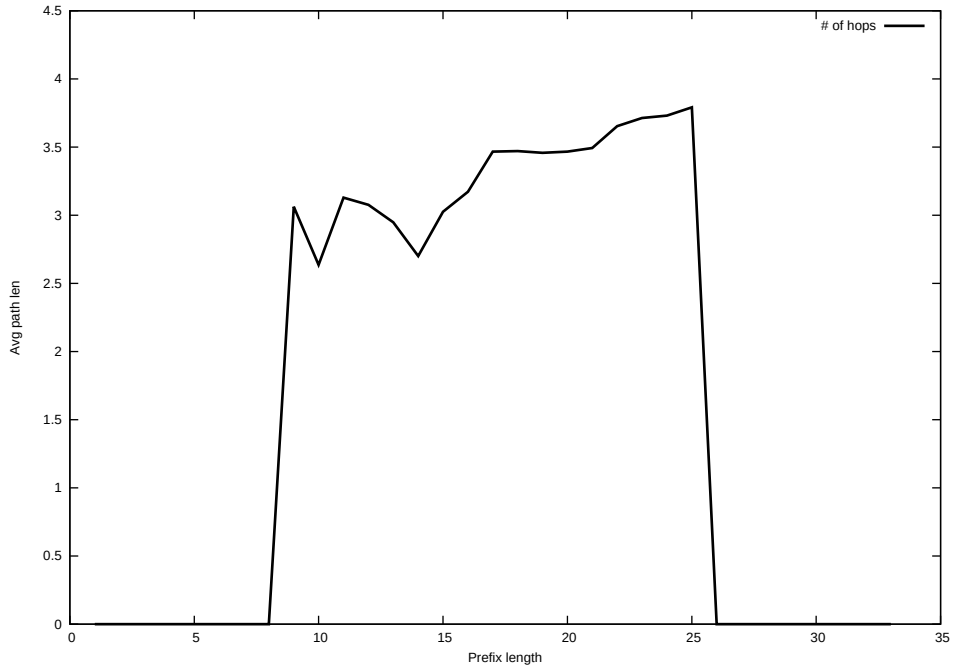
2013-11-11



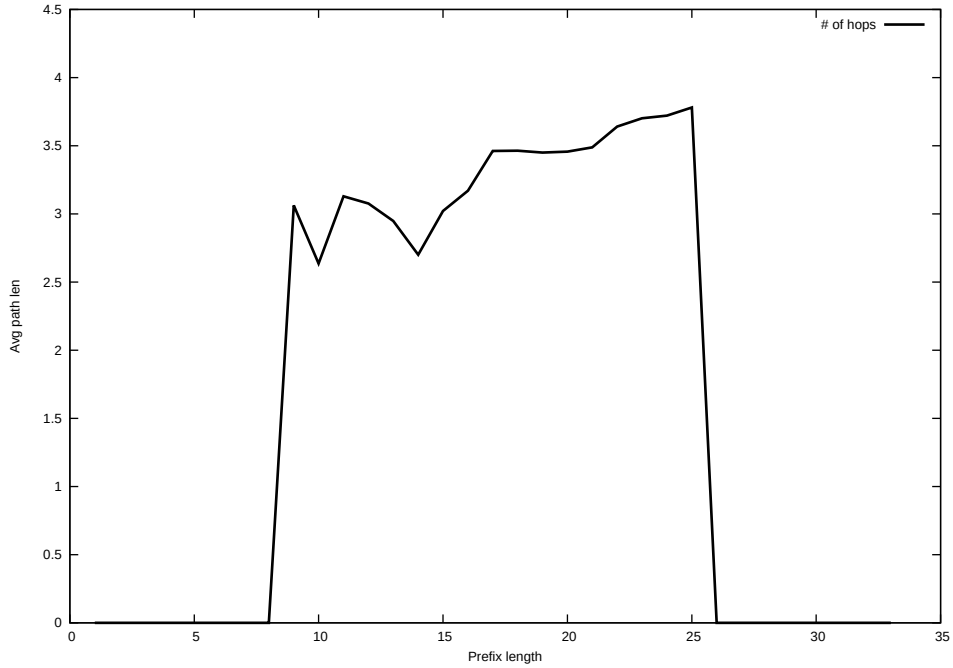
2013-11-12



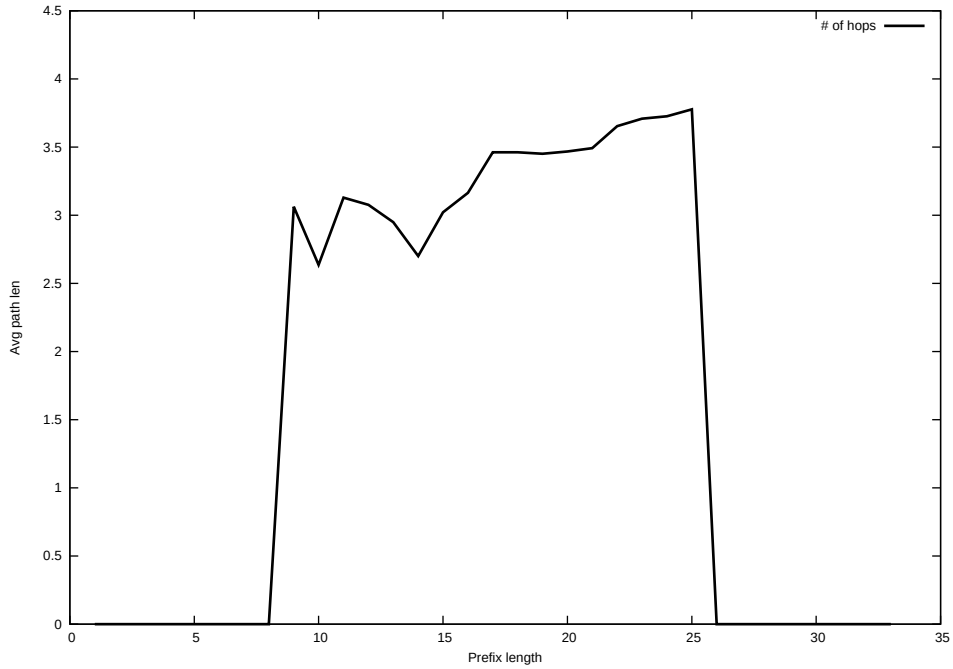
2013-11-13



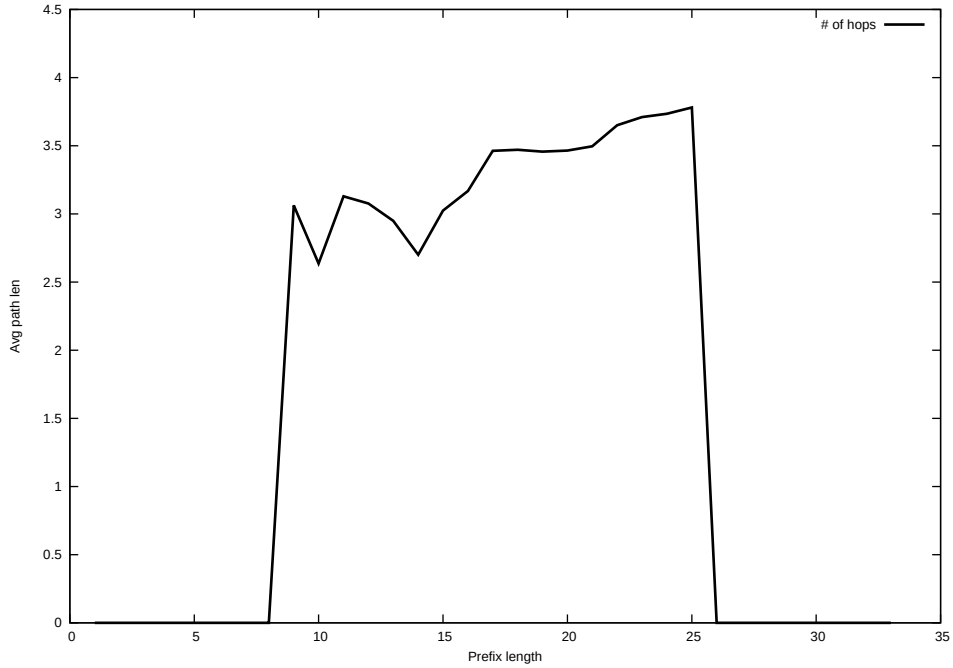
2013-11-14



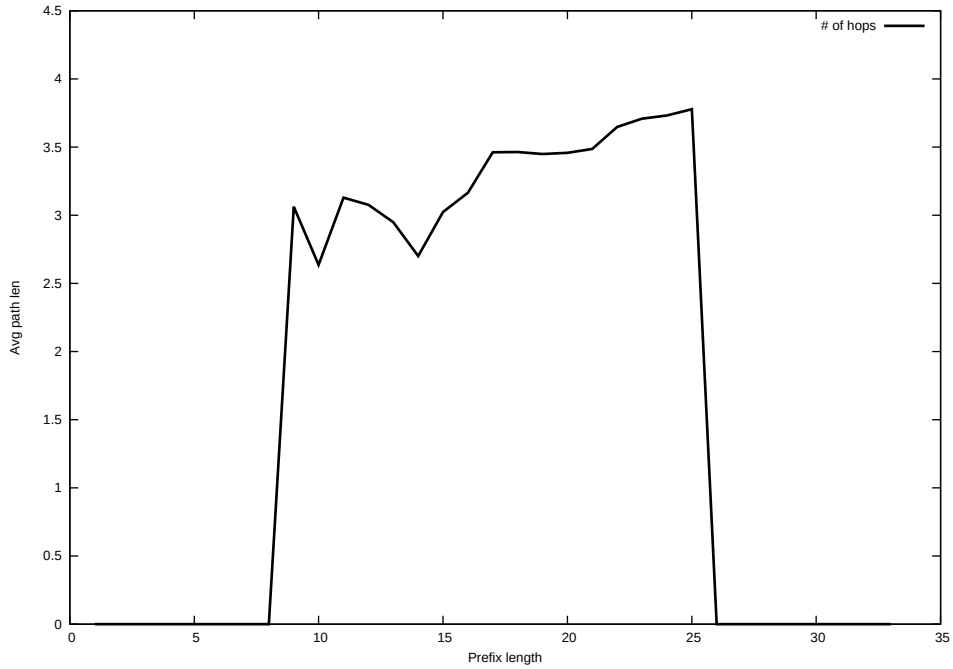
2013-11-15



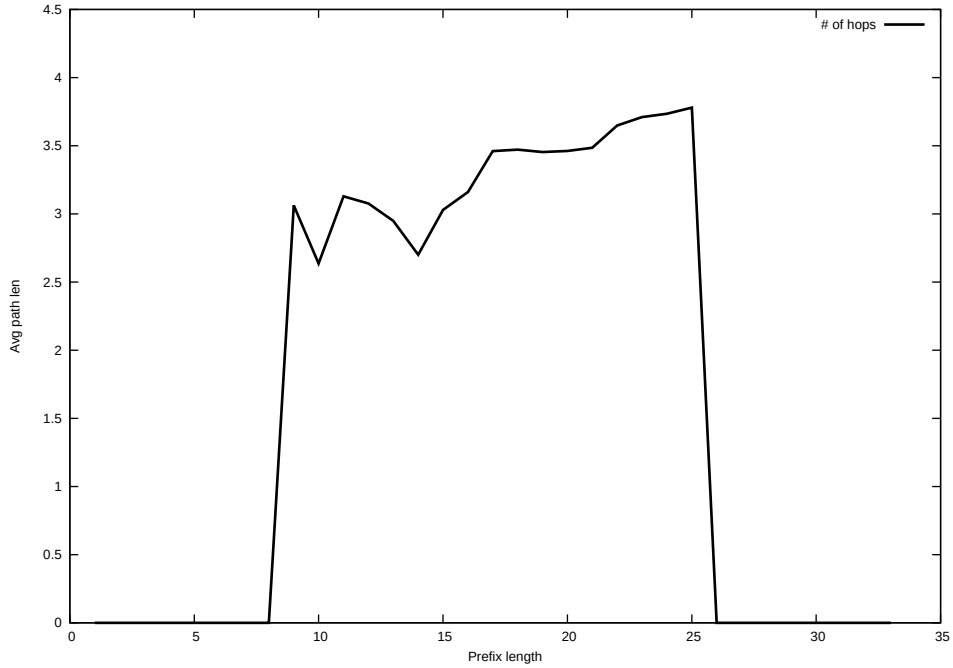
2013-11-16



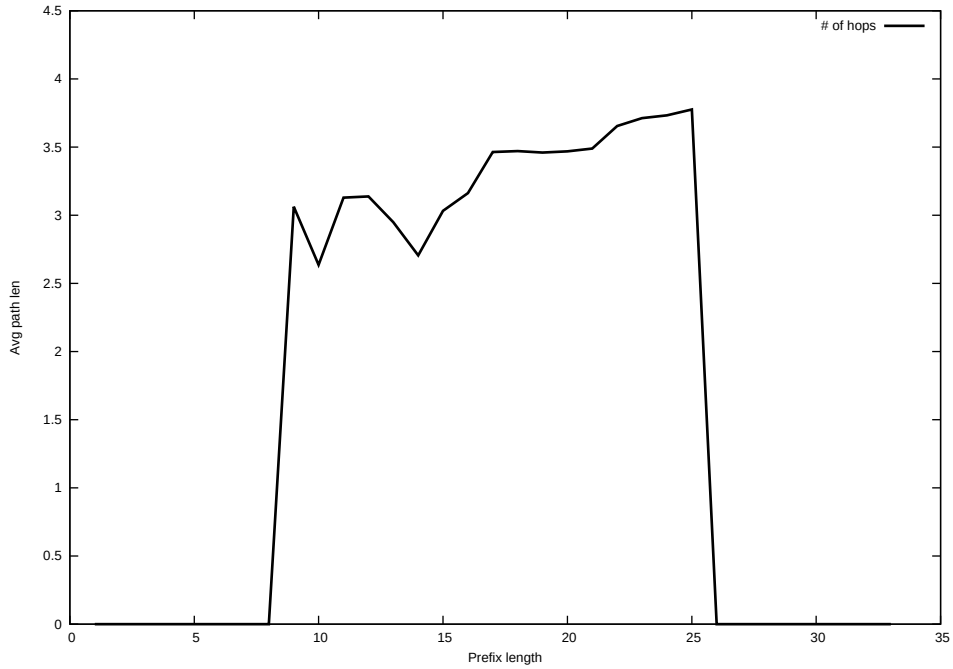
2013-11-17



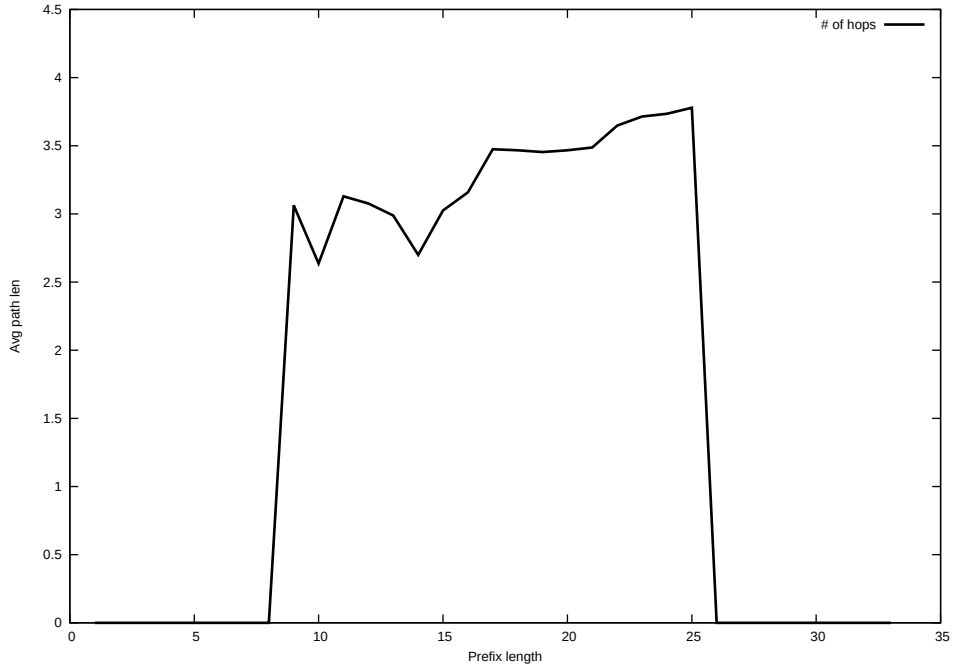
2013-11-18



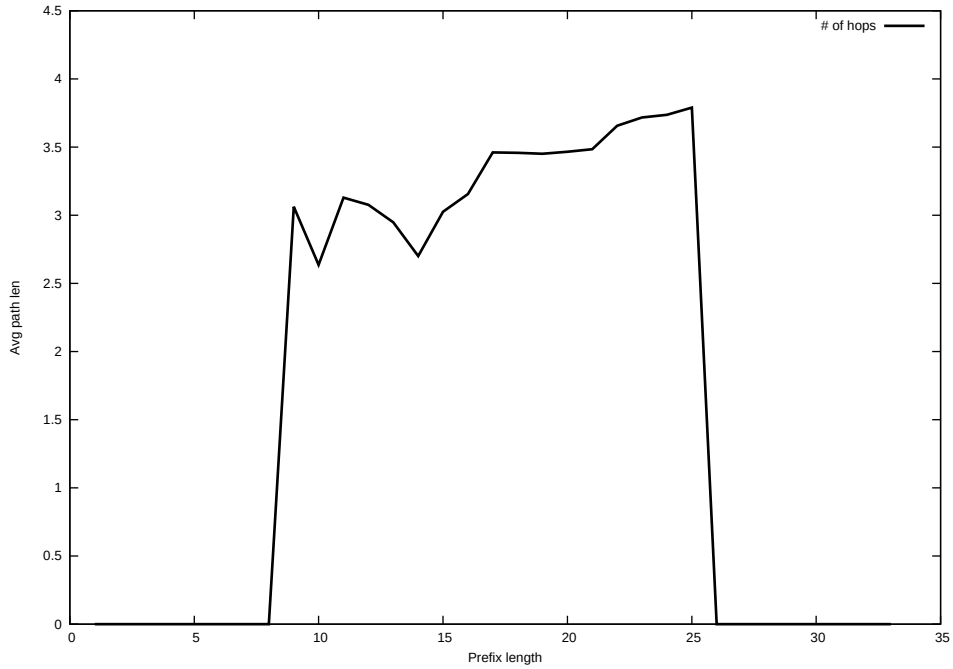
2013-11-19



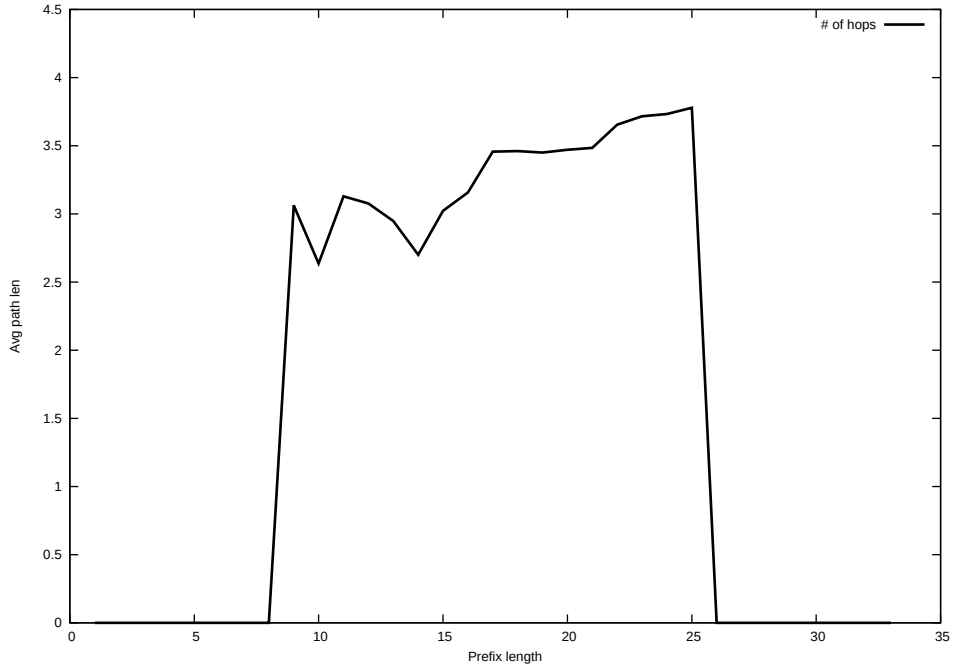
2013-11-20



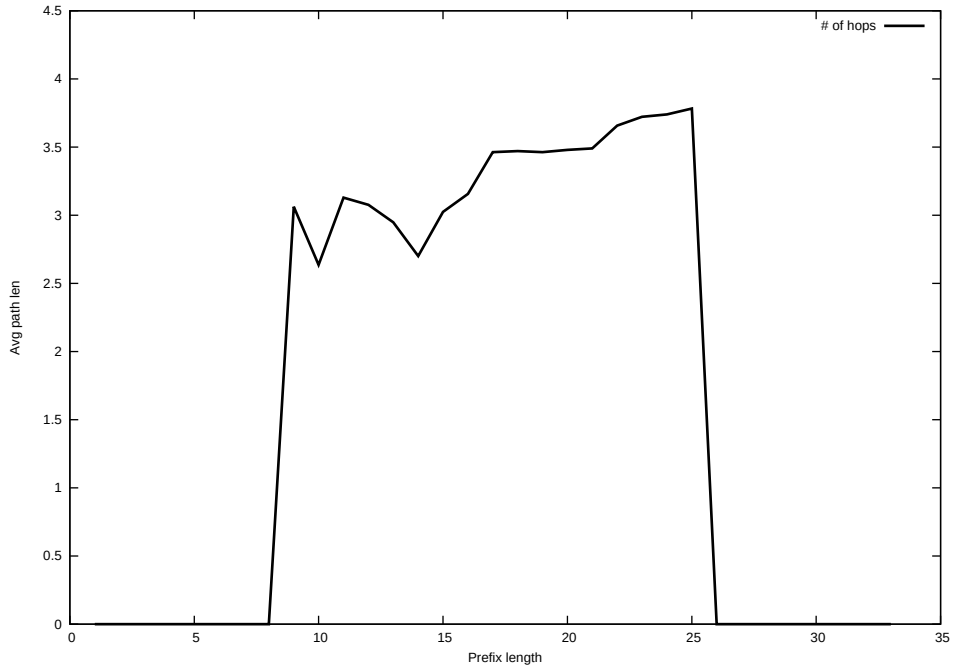
2013-11-21



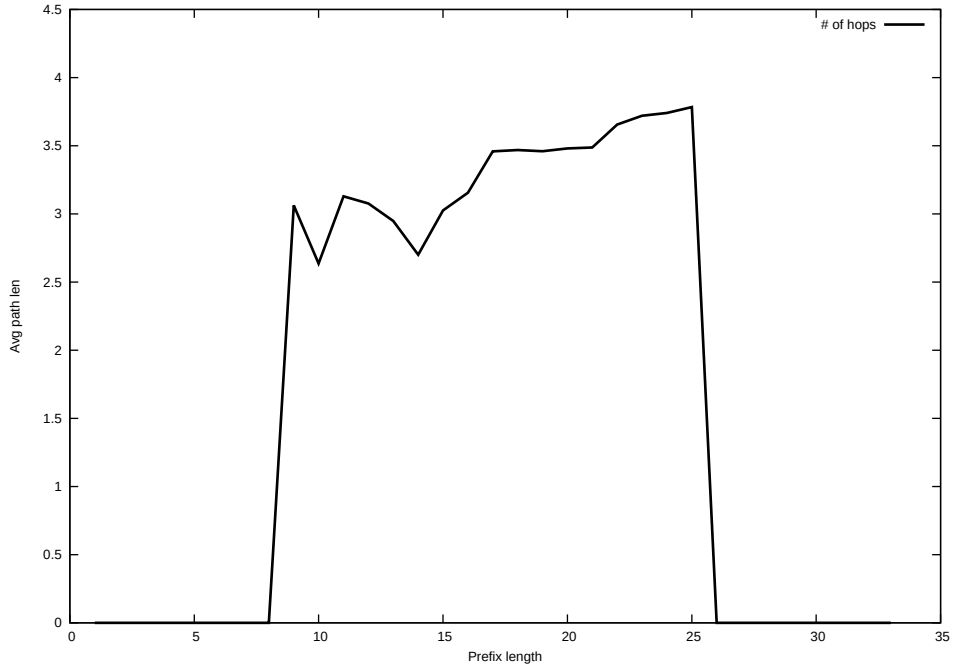
2013-11-22



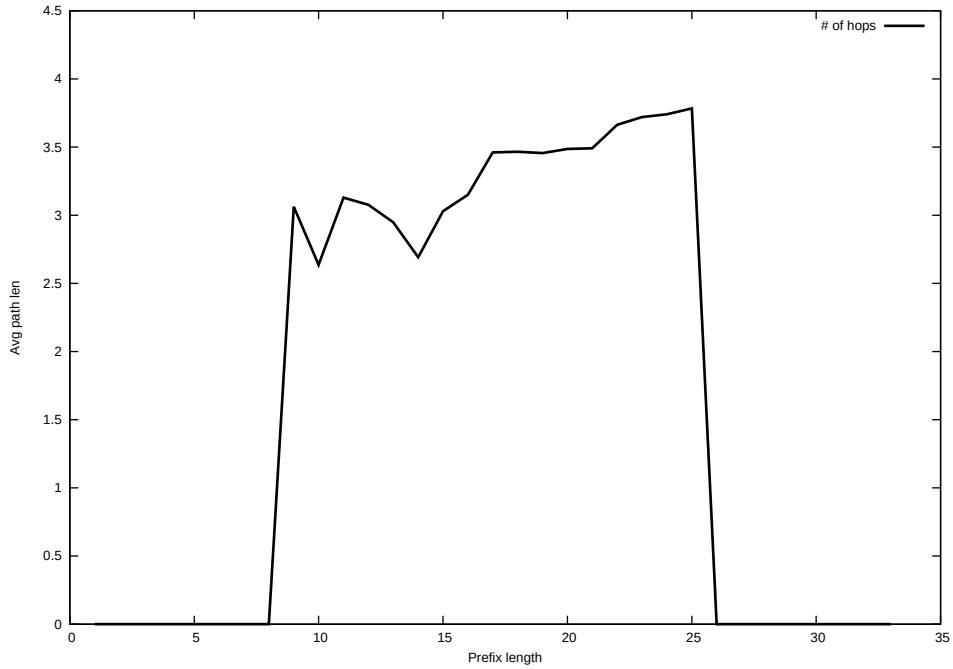
2013-11-23



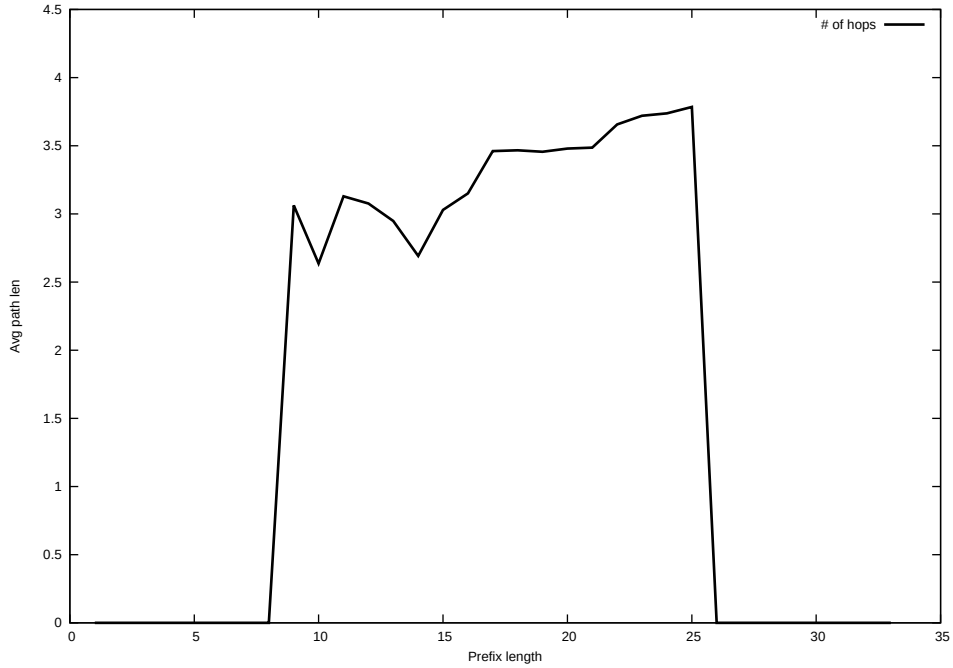
2013-11-24



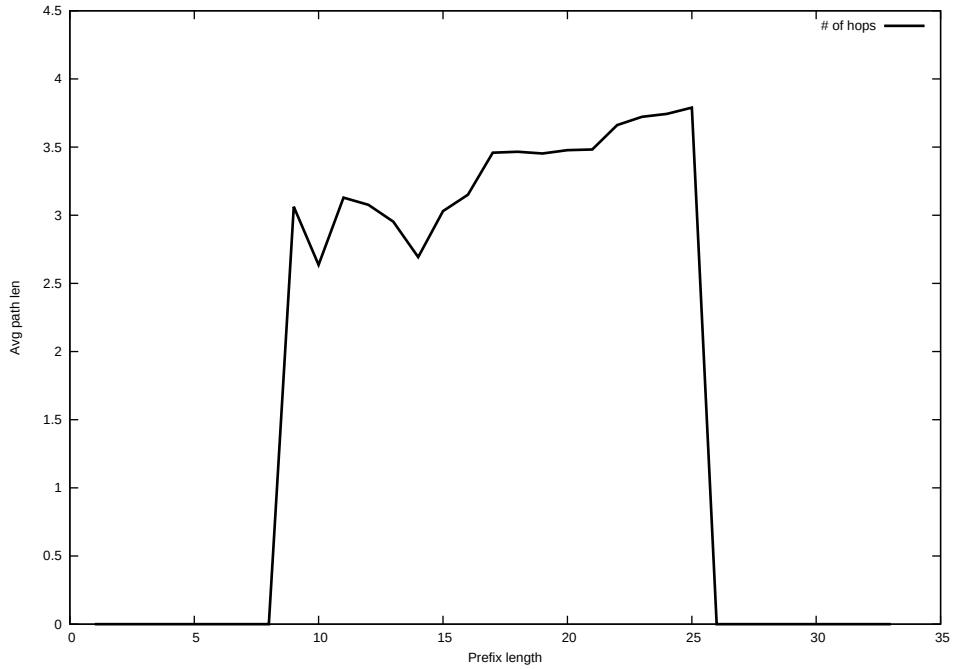
2013-11-25



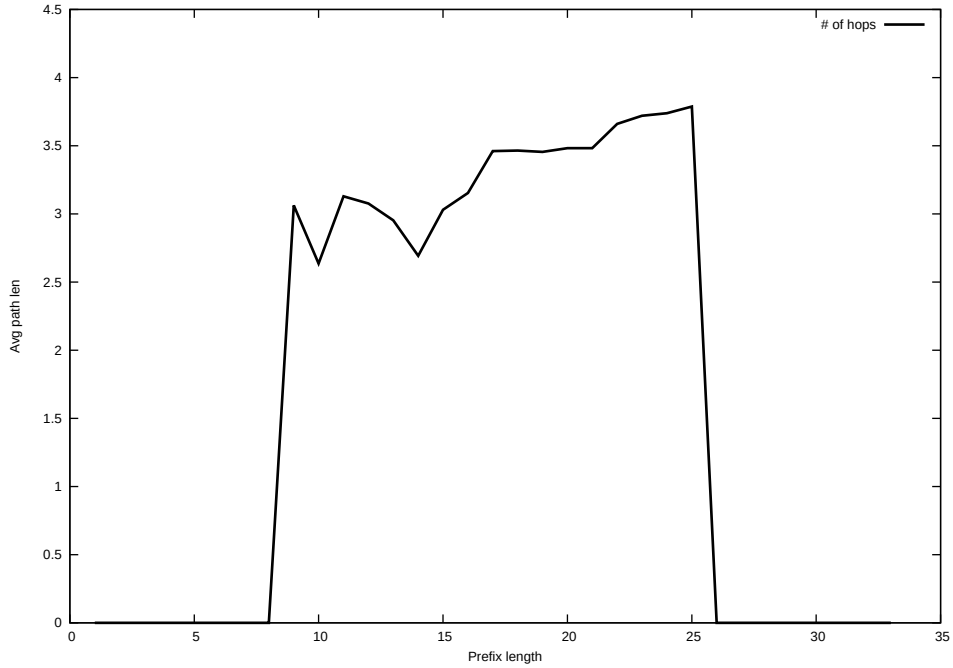
2013-11-26



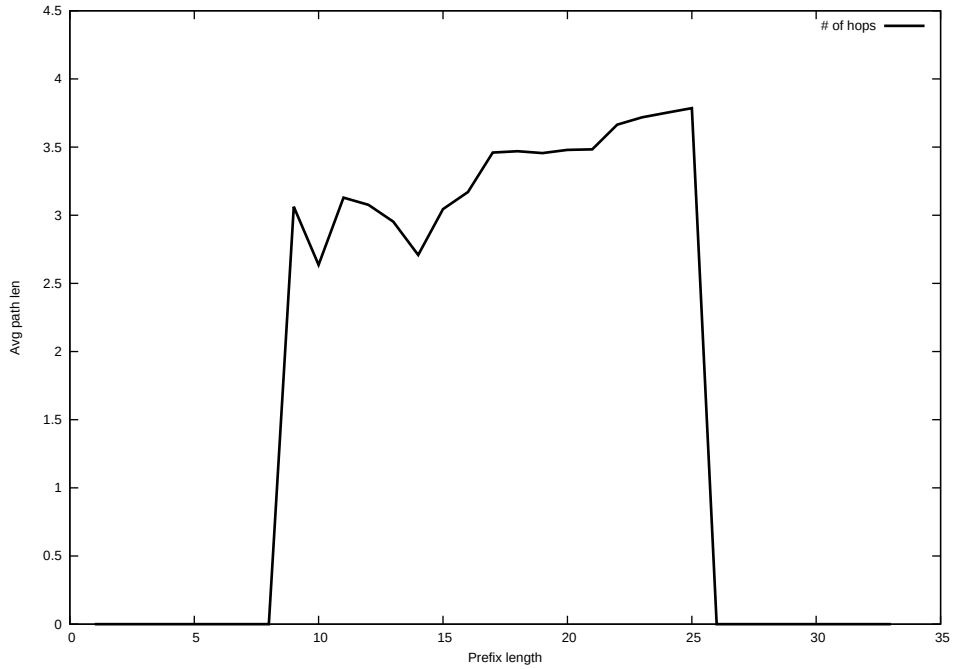
2013-11-27



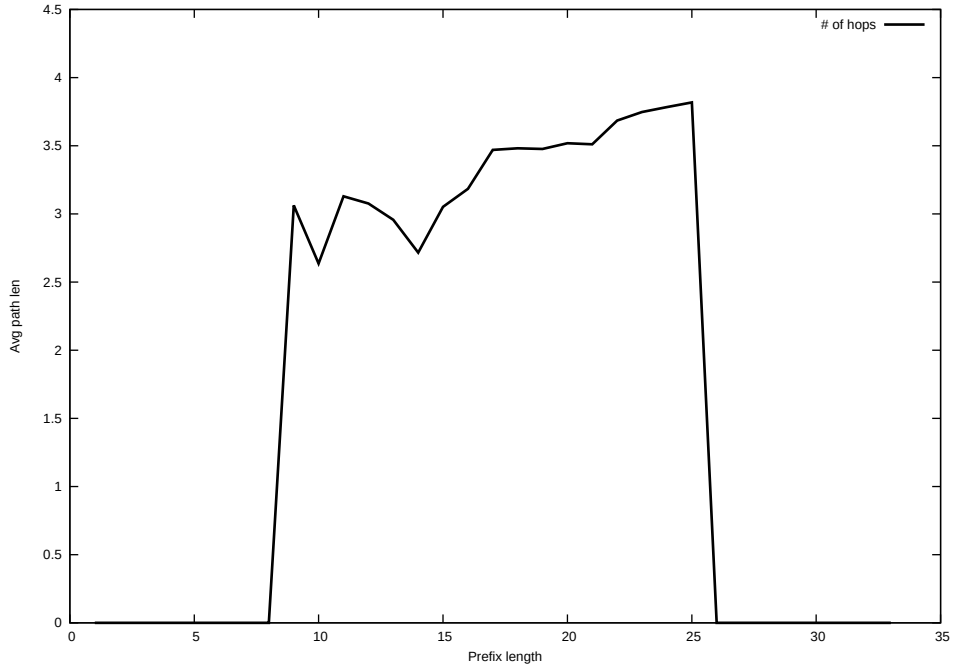
2013-11-28



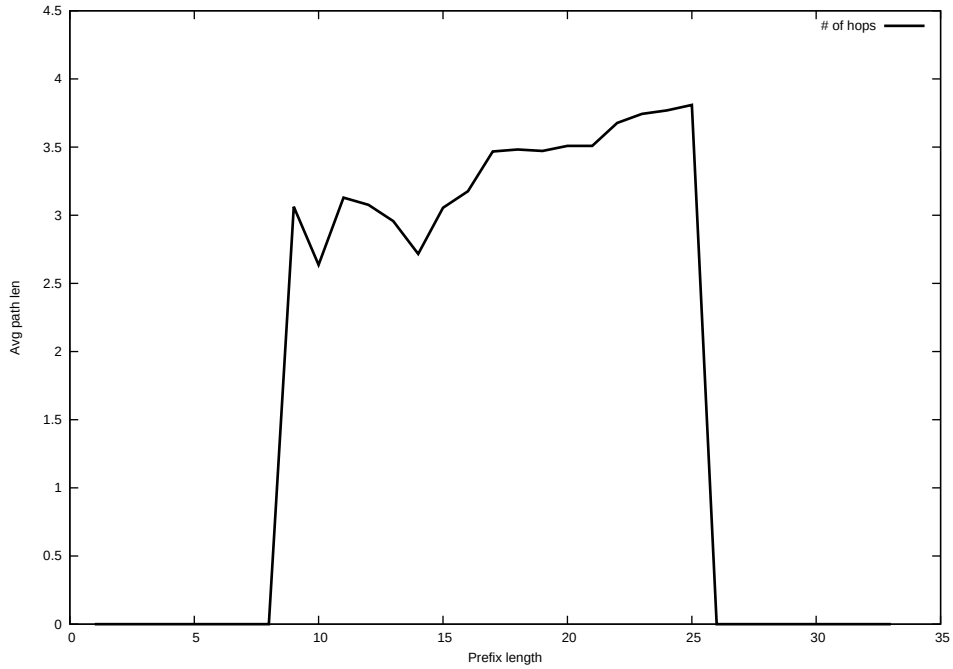
2013-11-29



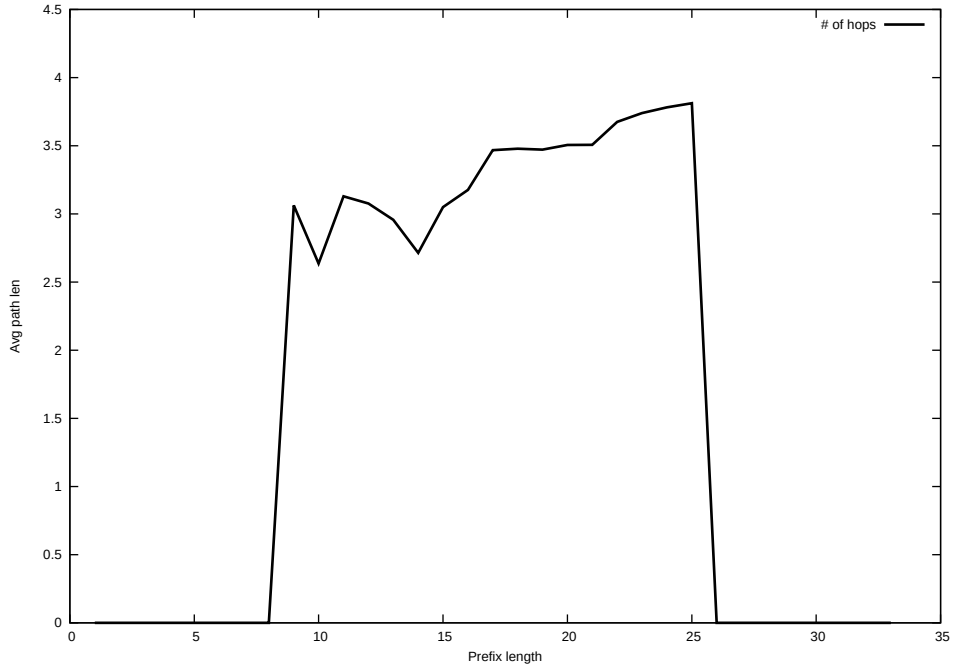
2013-11-30



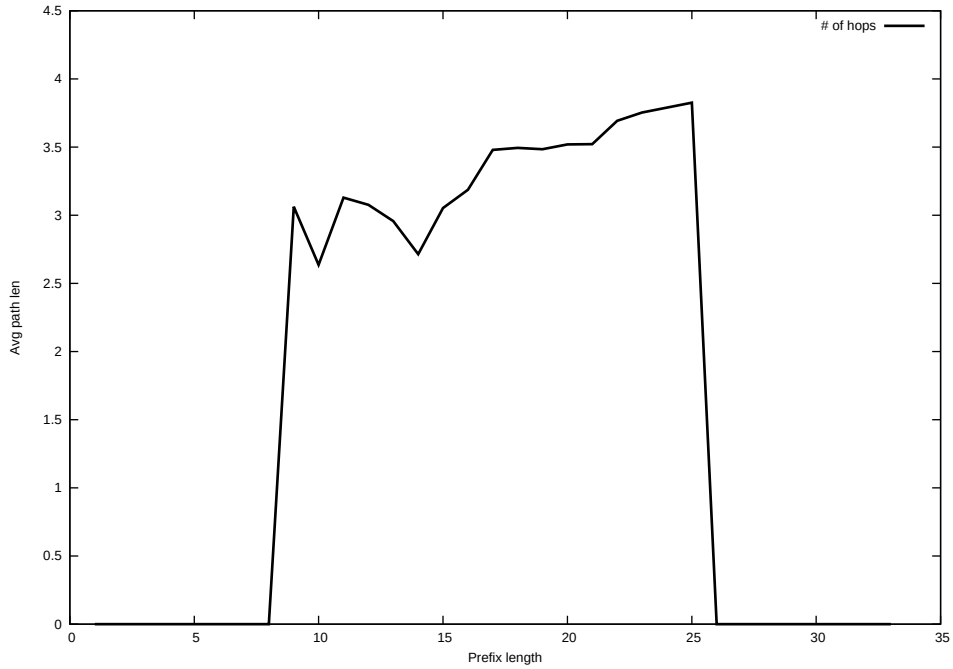
2013-12-01



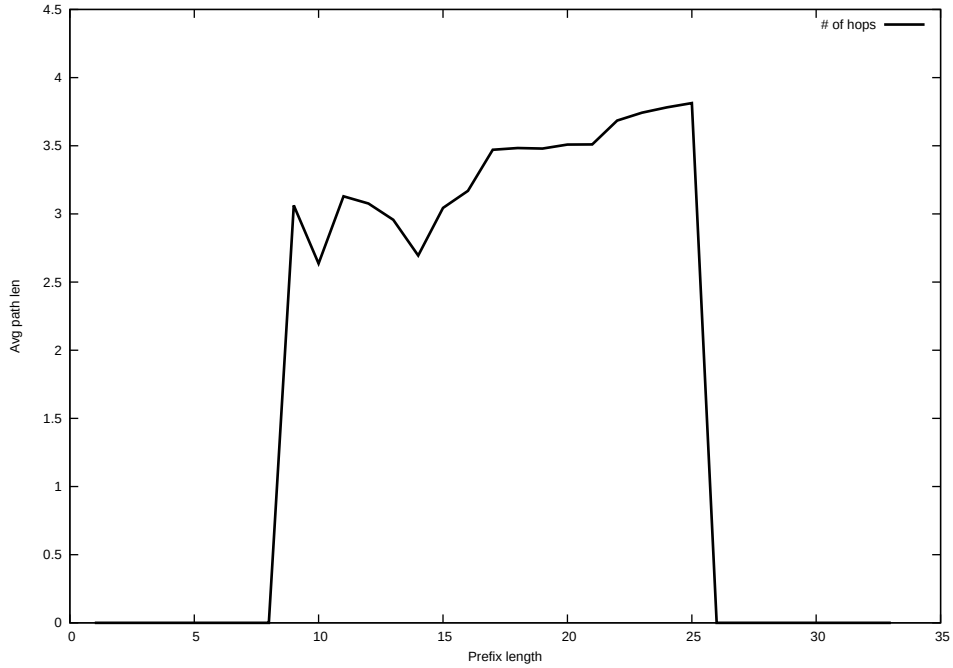
2013-12-02



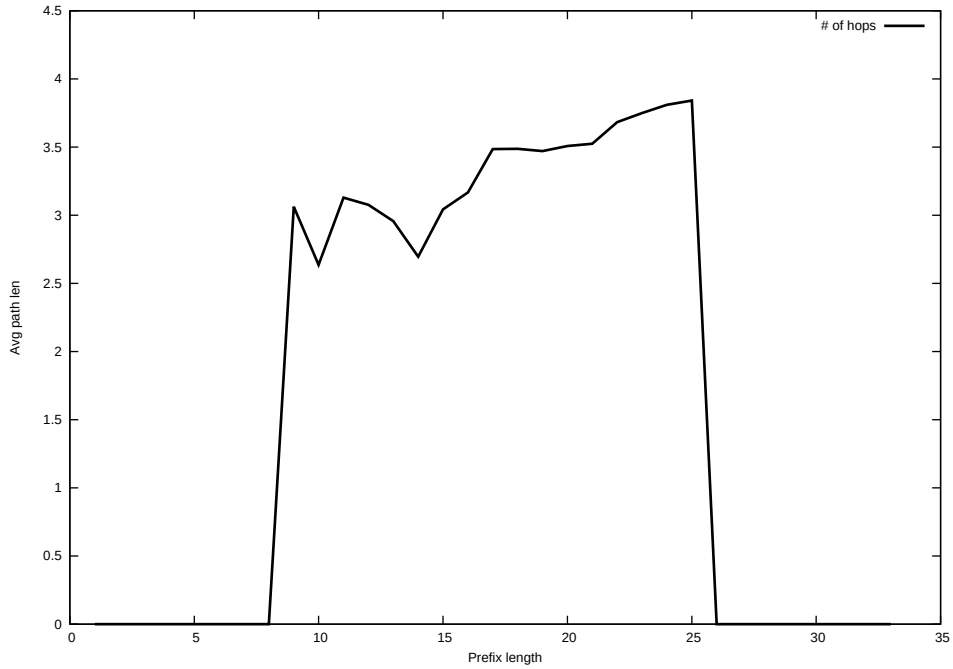
2013-12-03



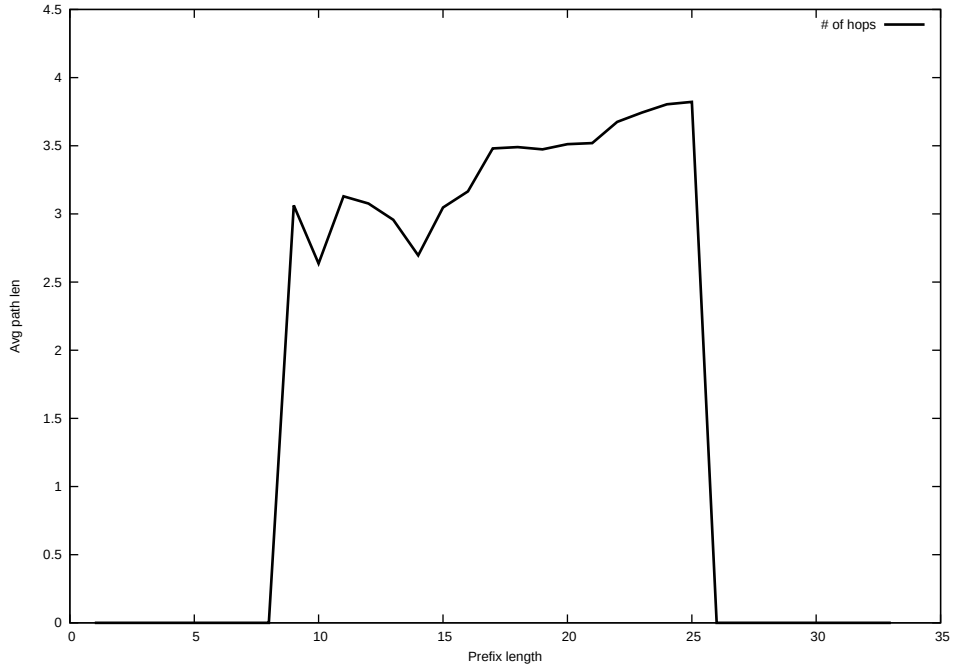
2013-12-04



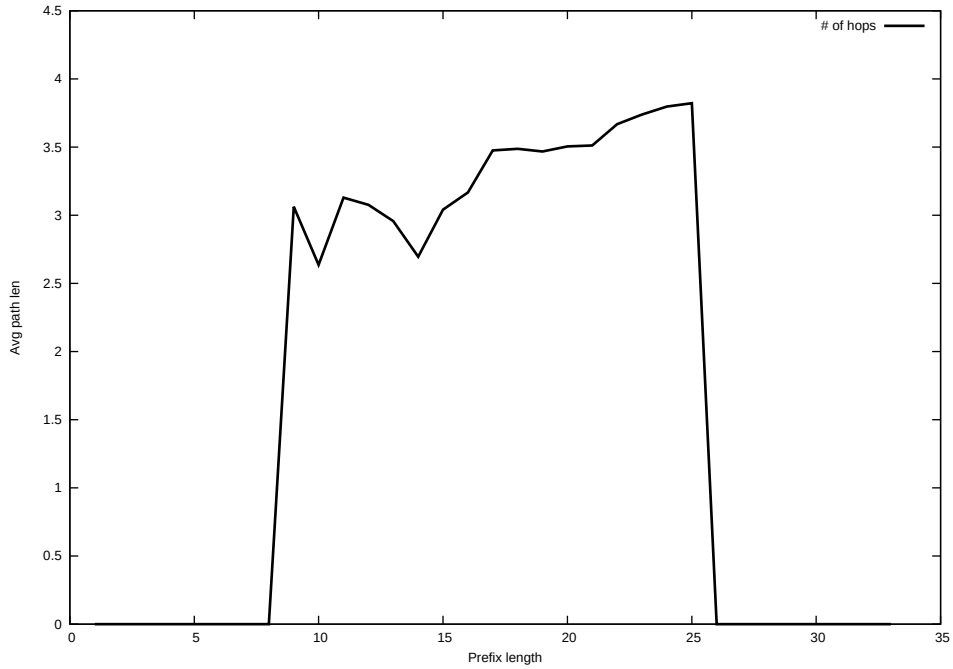
2013-12-05



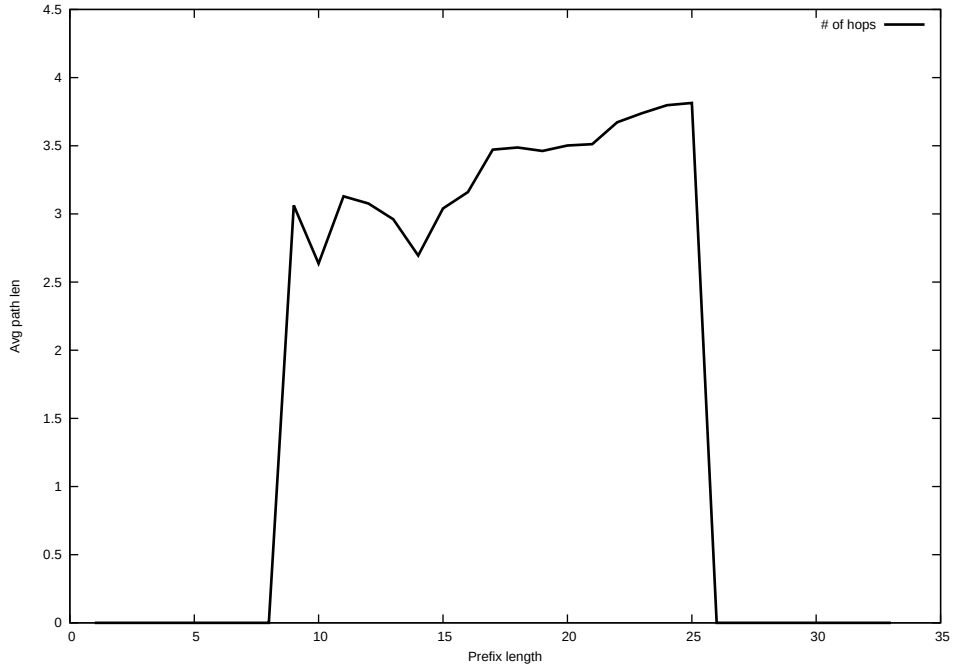
2013-12-06



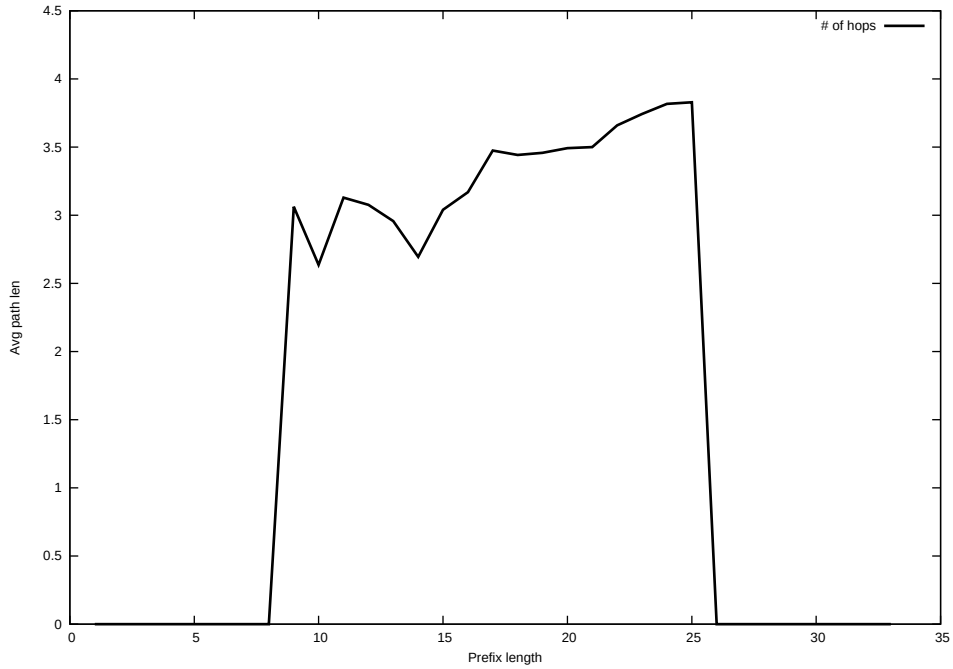
2013-12-07



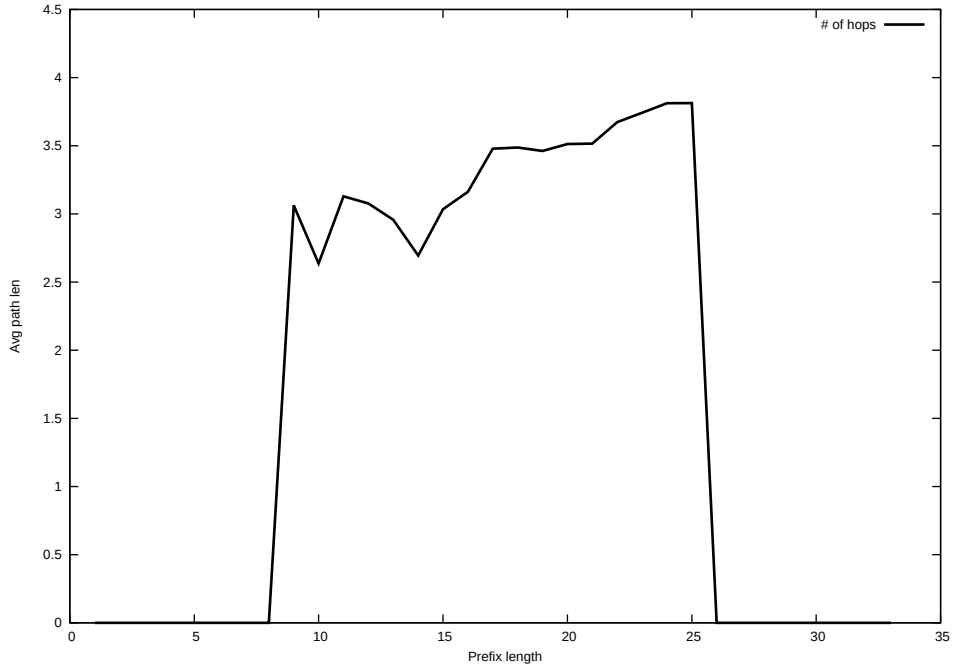
2013-12-08



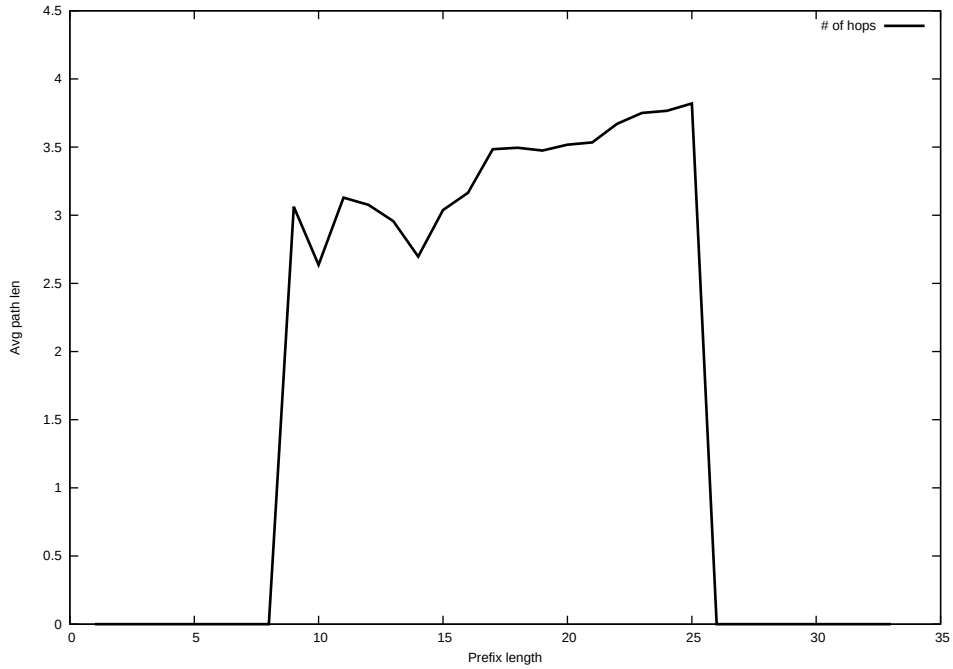
2013-12-09



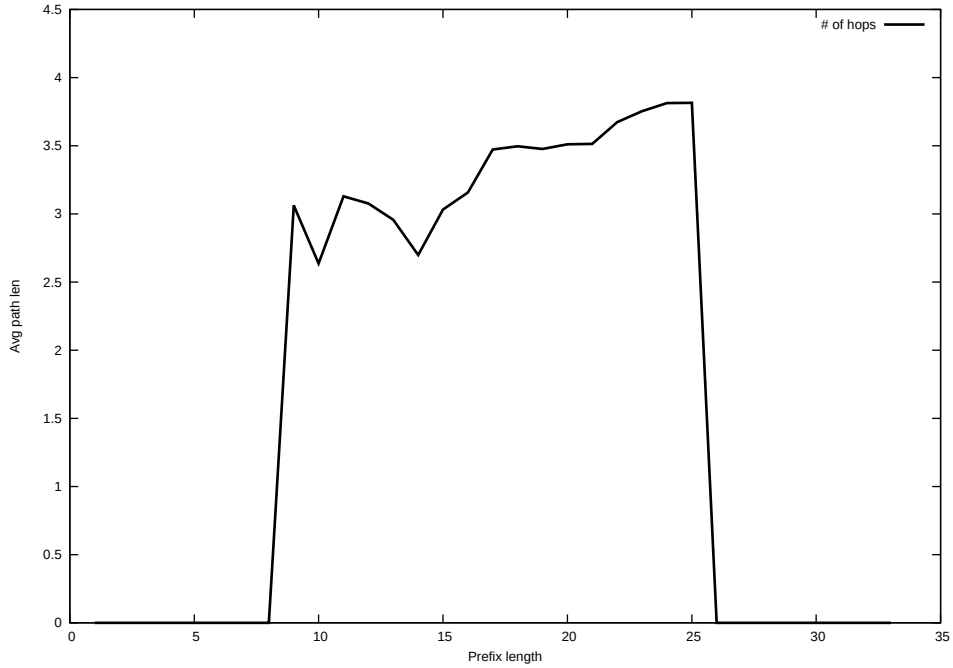
2013-12-10



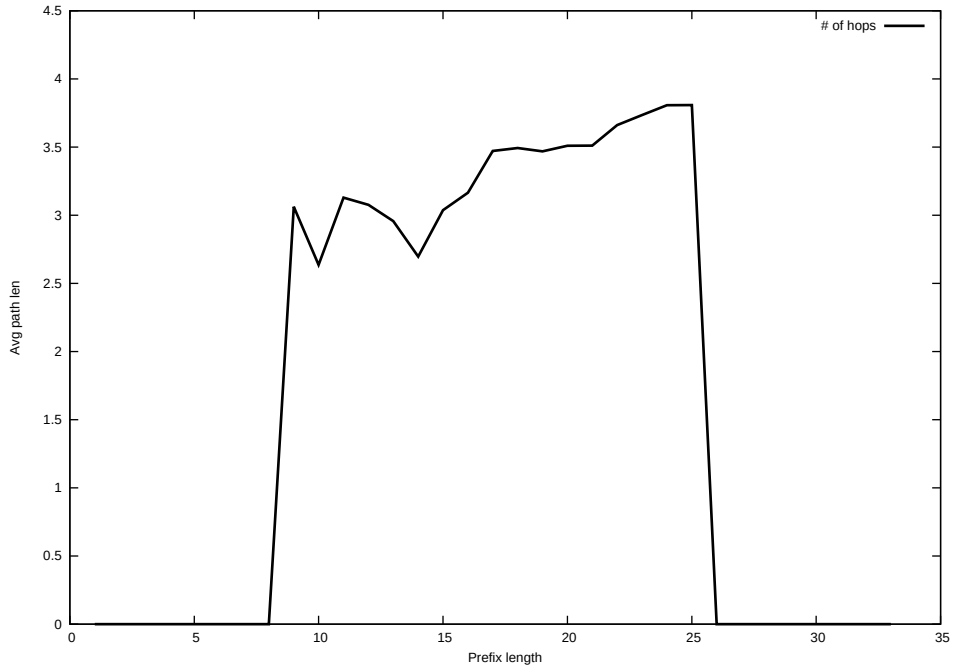
2013-12-11



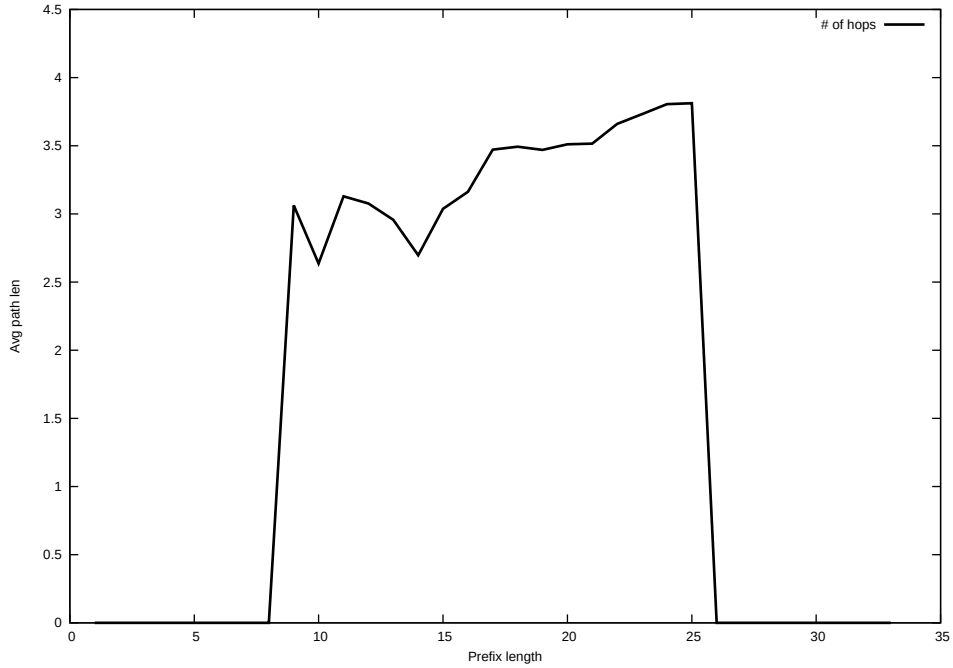
2013-12-12



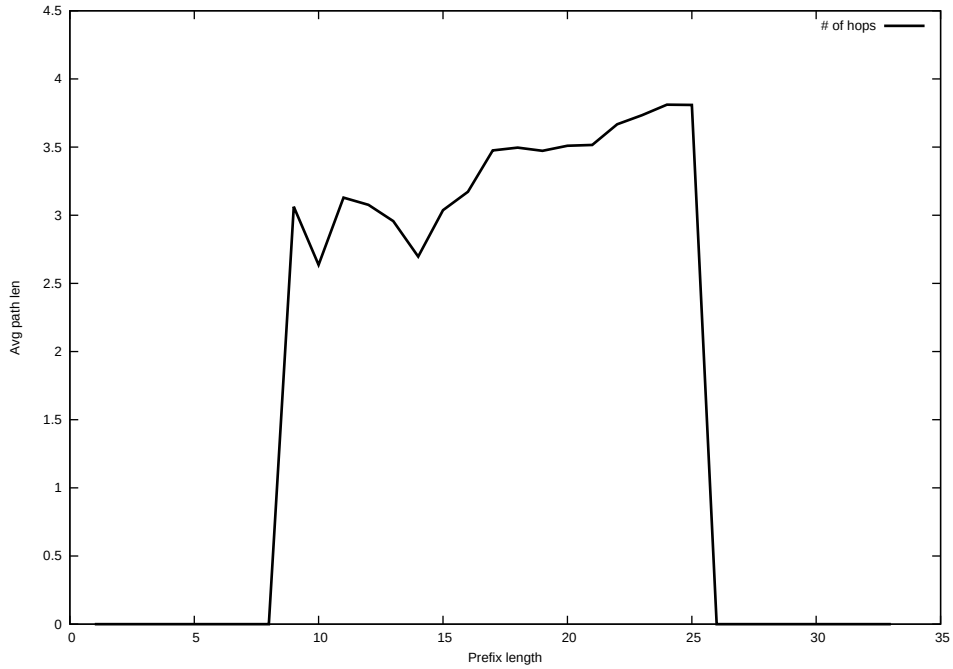
2013-12-13



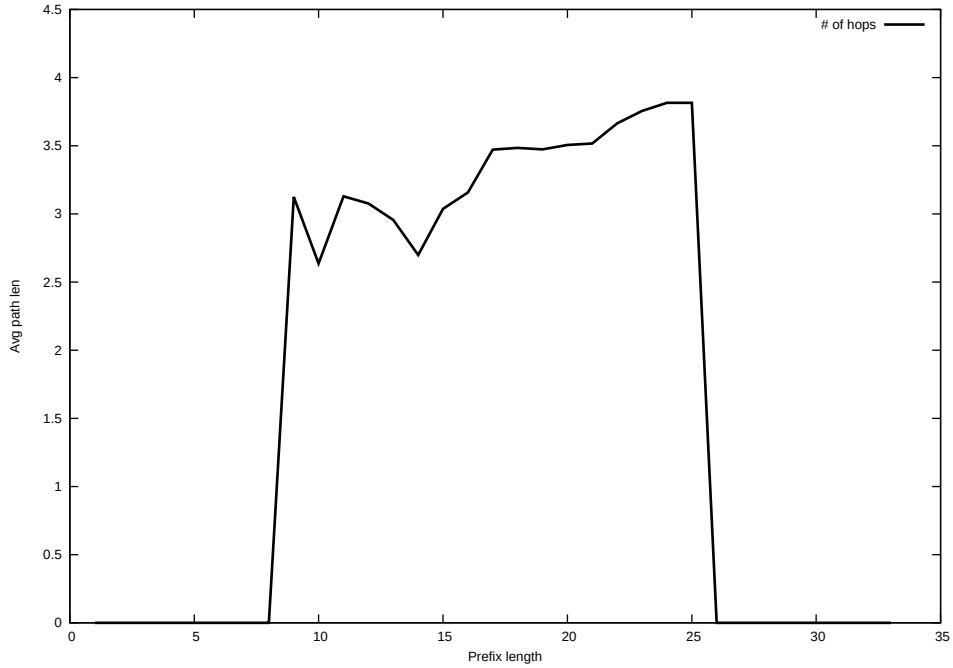
2013-12-14



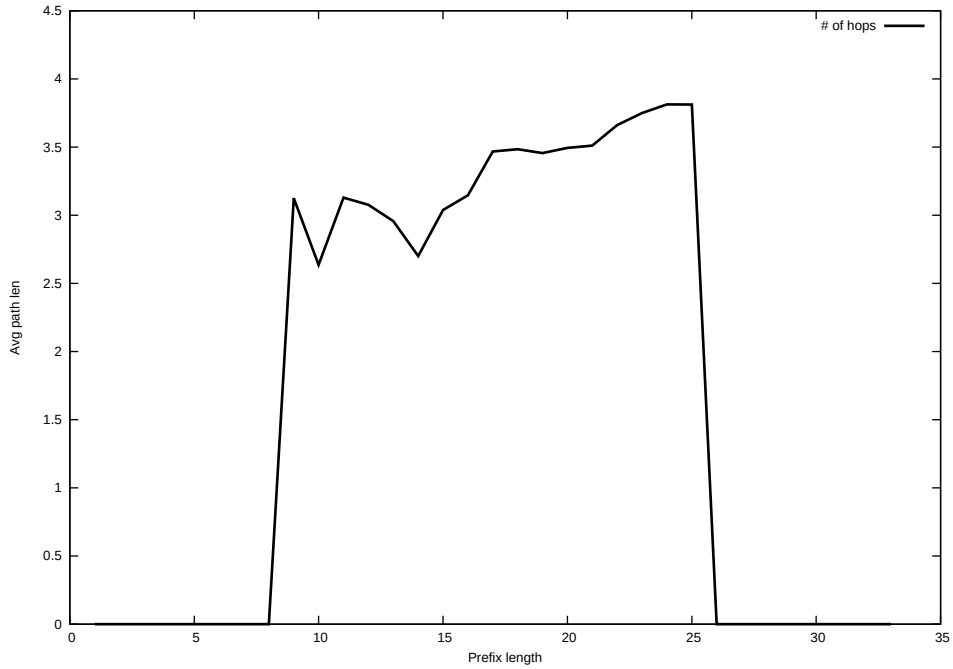
2013-12-15



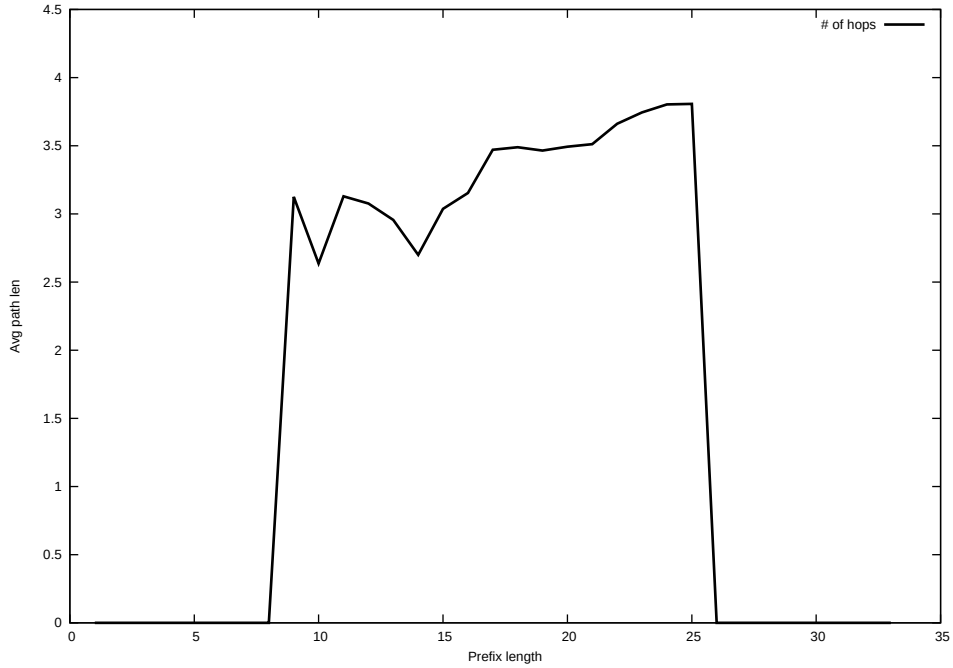
2013-12-16



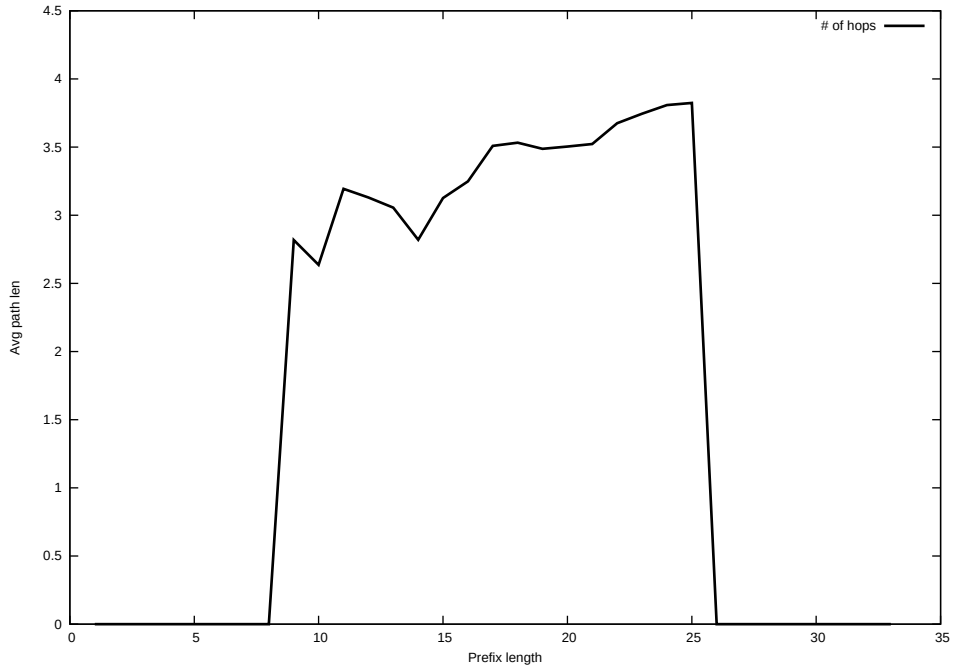
2013-12-17



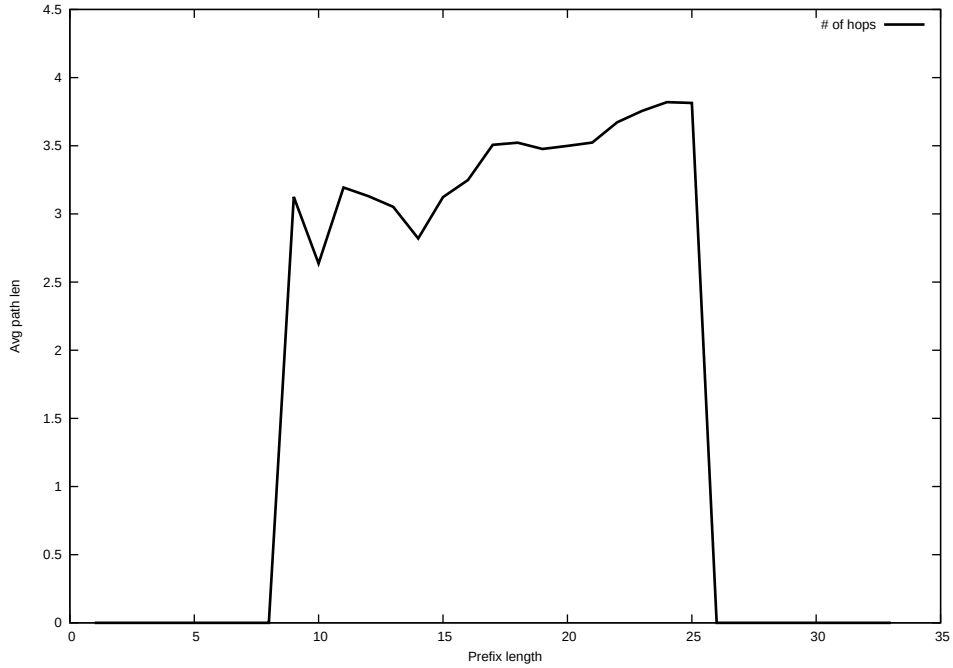
2013-12-18



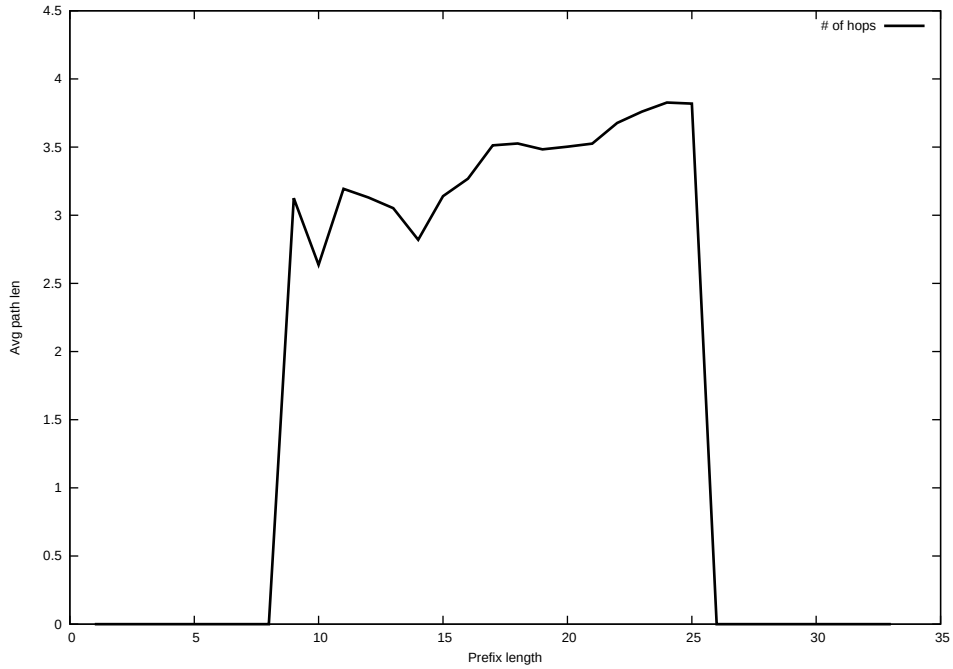
2013-12-19



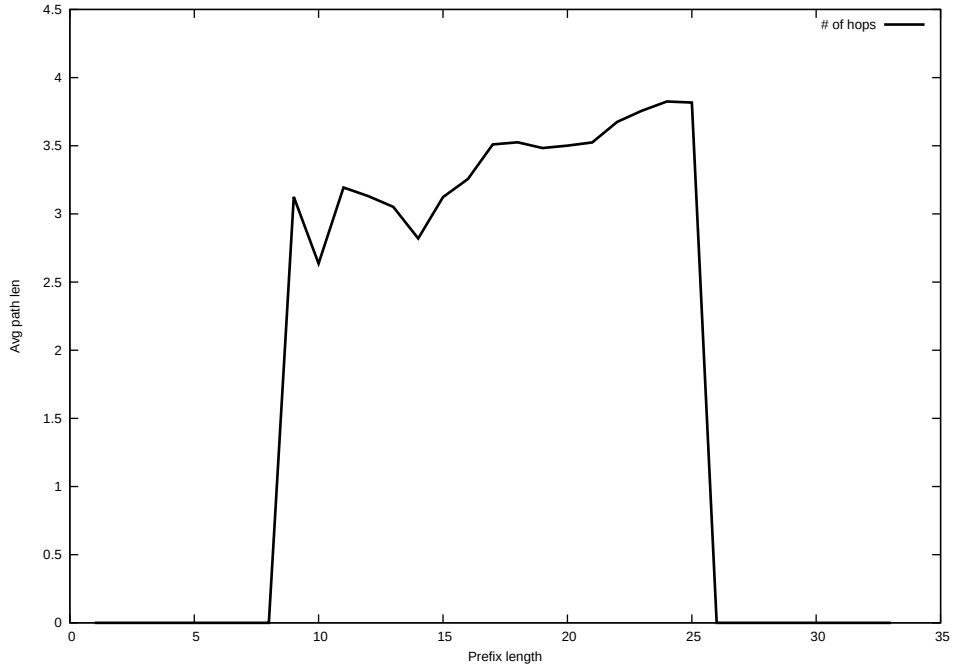
2013-12-20



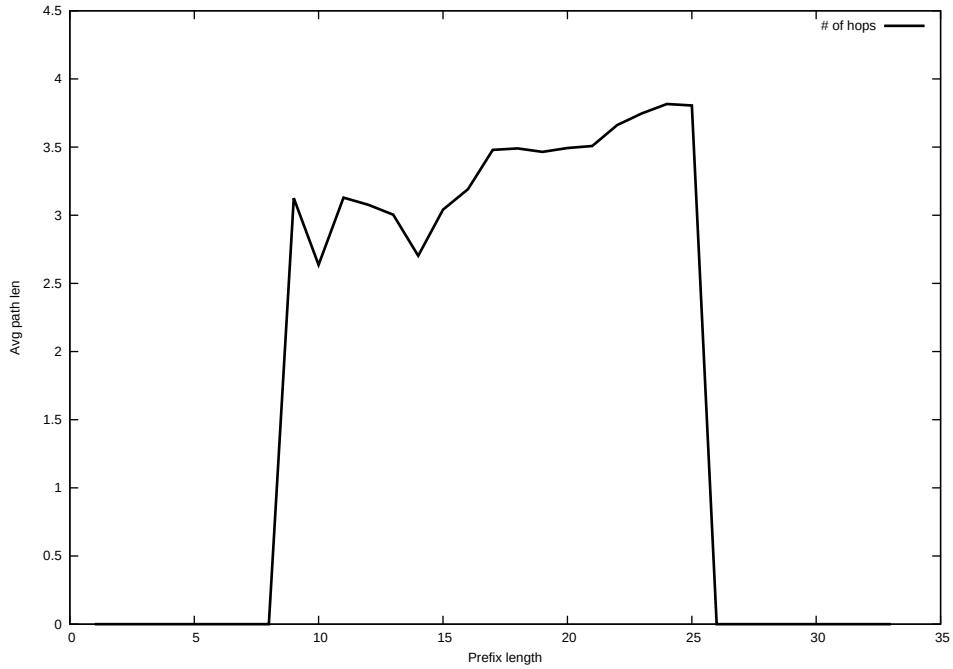
2013-12-21



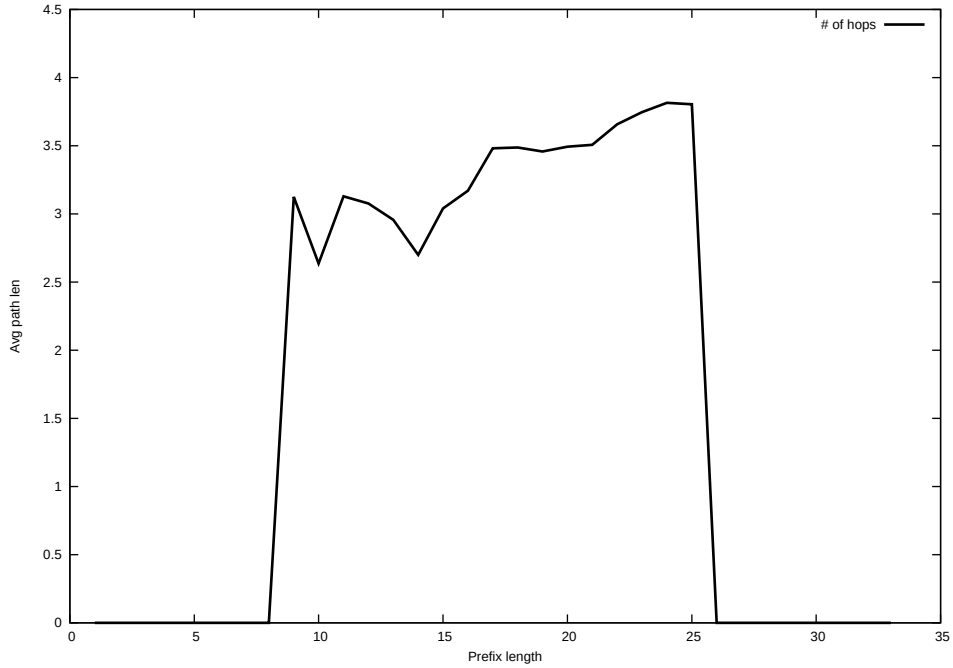
2013-12-22



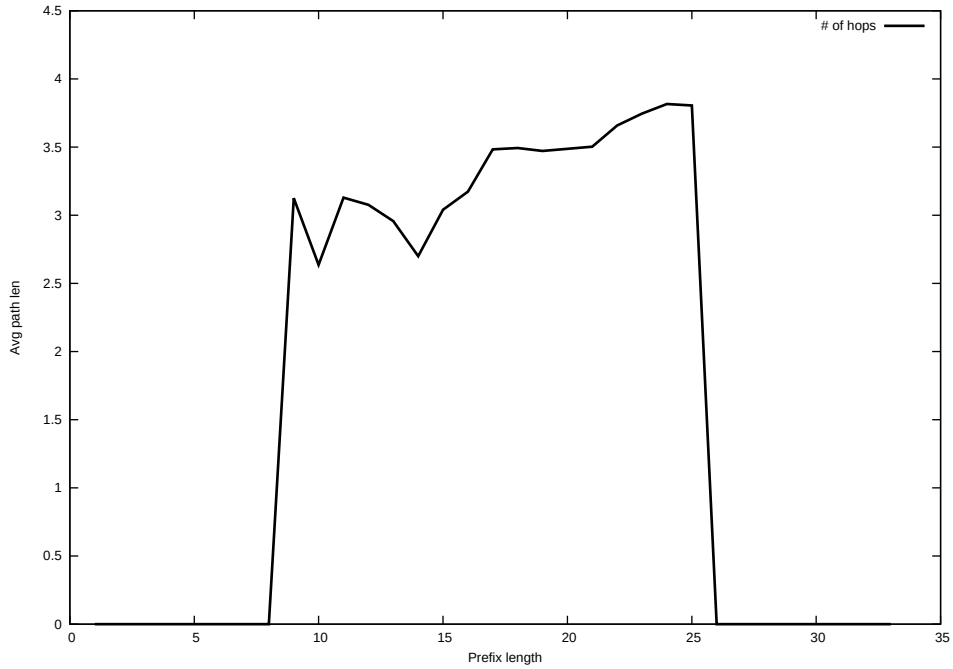
2013-12-23



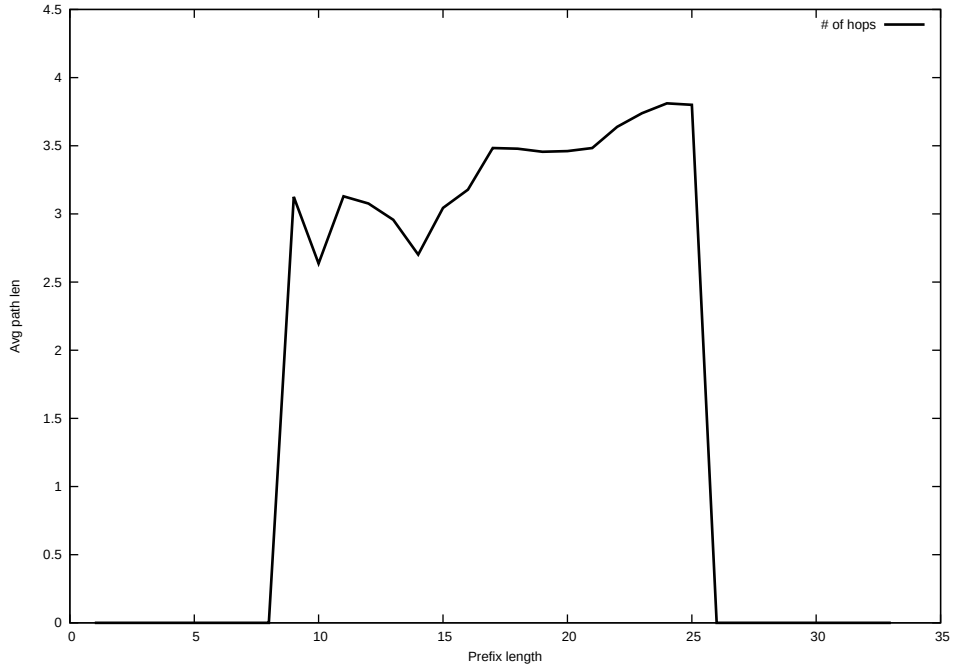
2013-12-24



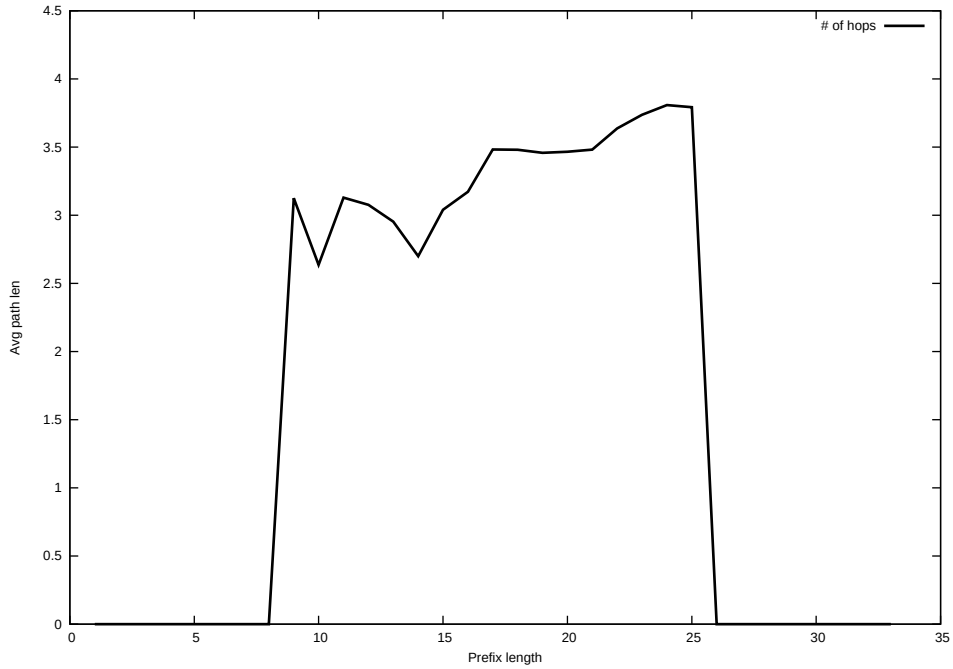
2013-12-25



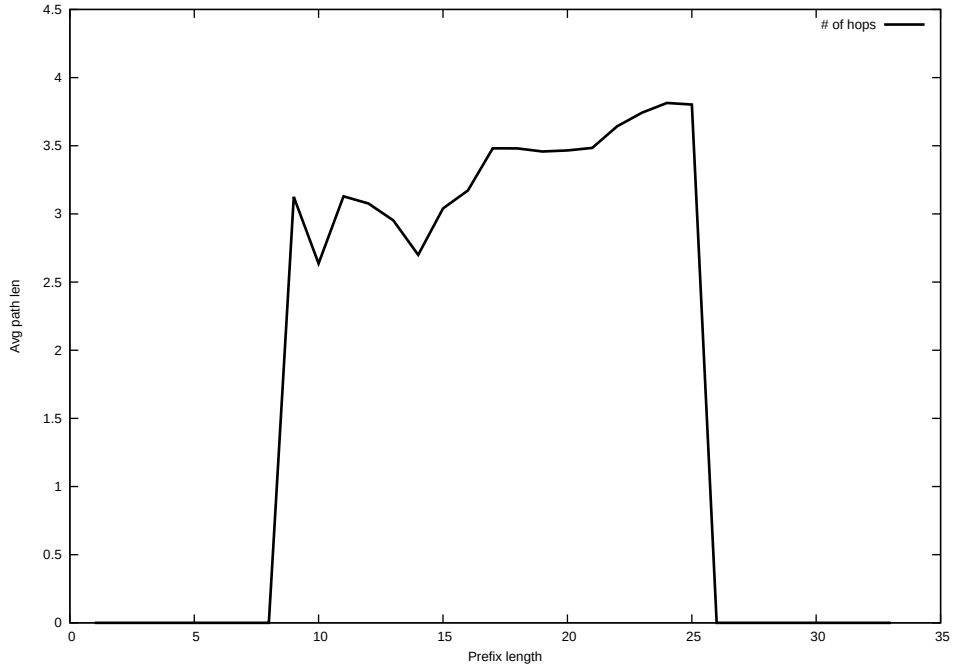
2013-12-26



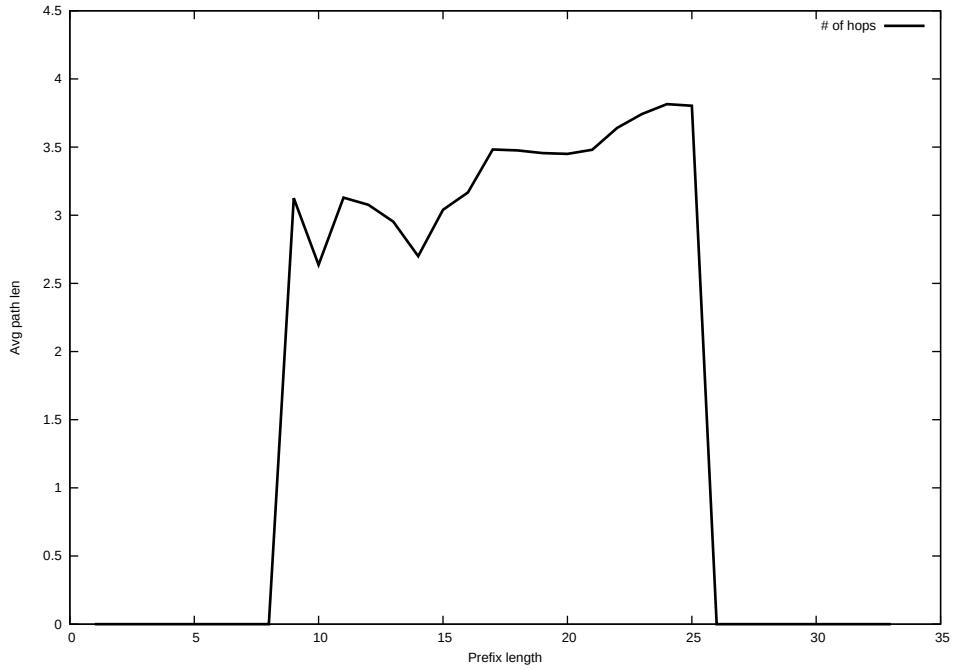
2013-12-27



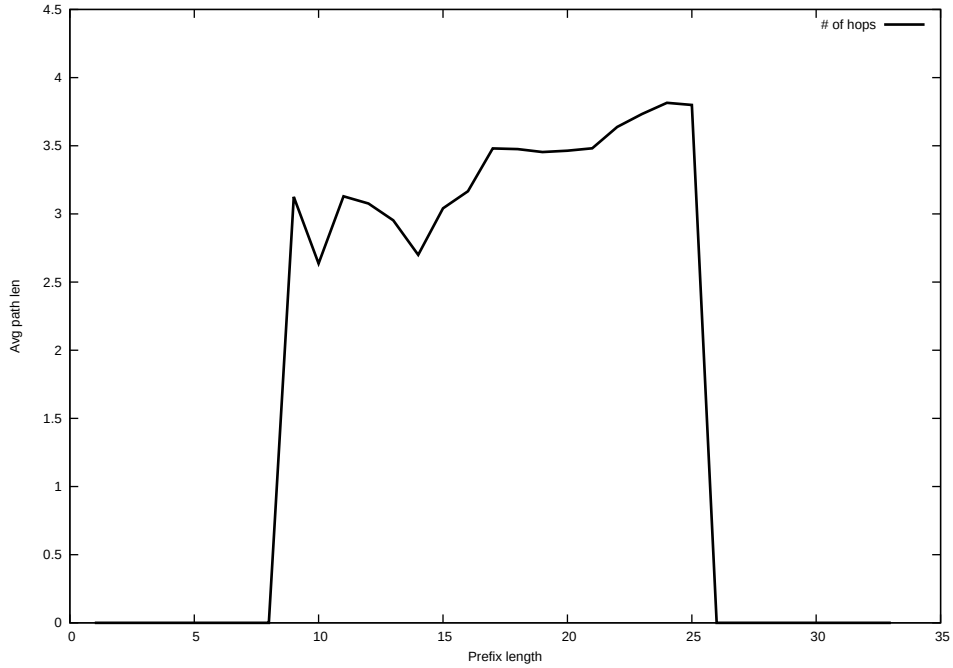
2013-12-28



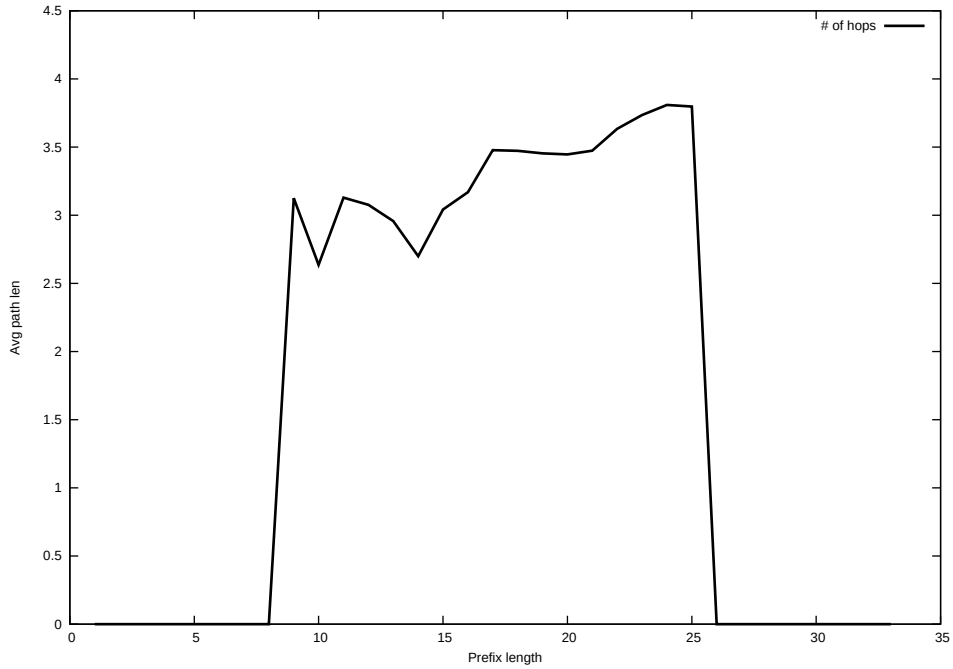
2013-12-29



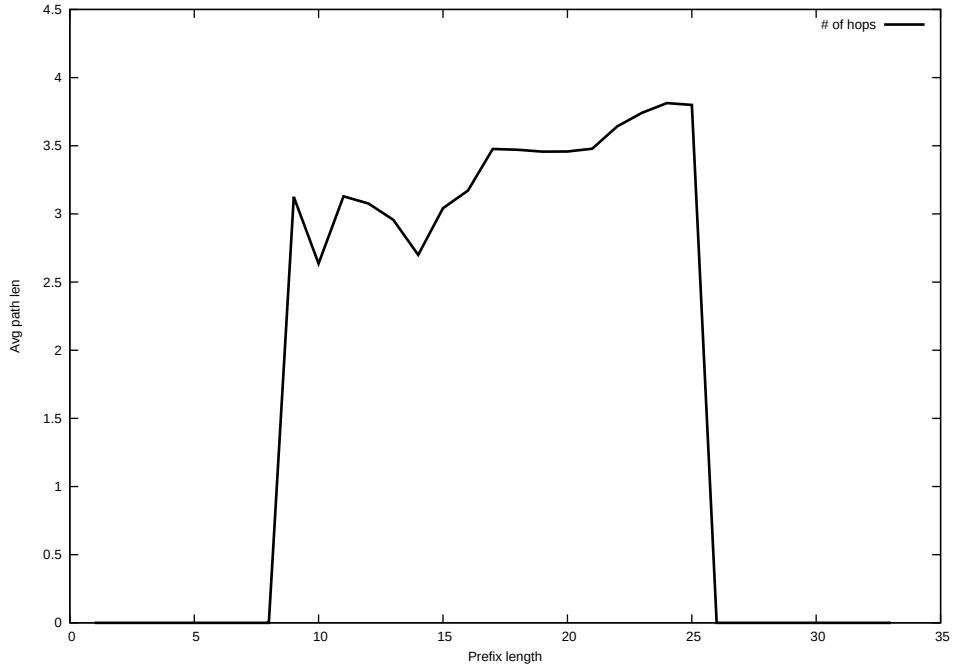
2013-12-30



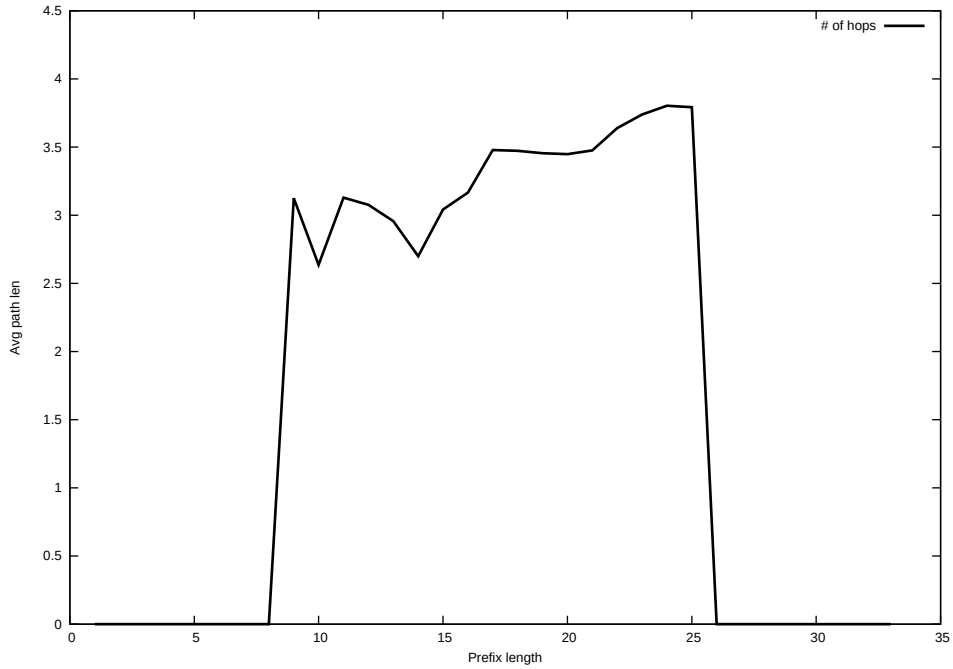
2013-12-31



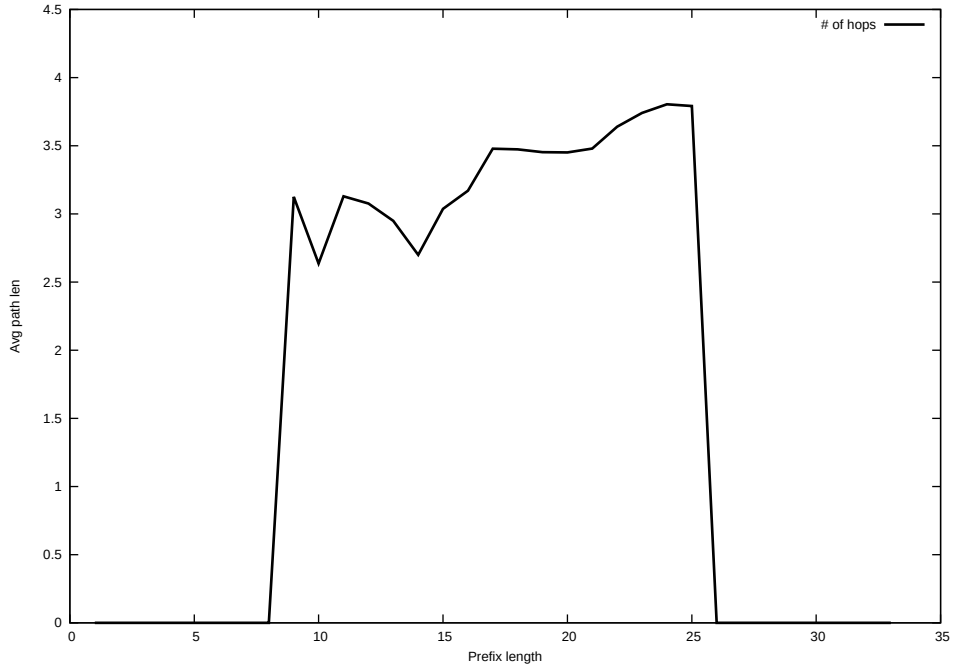
2014-01-01



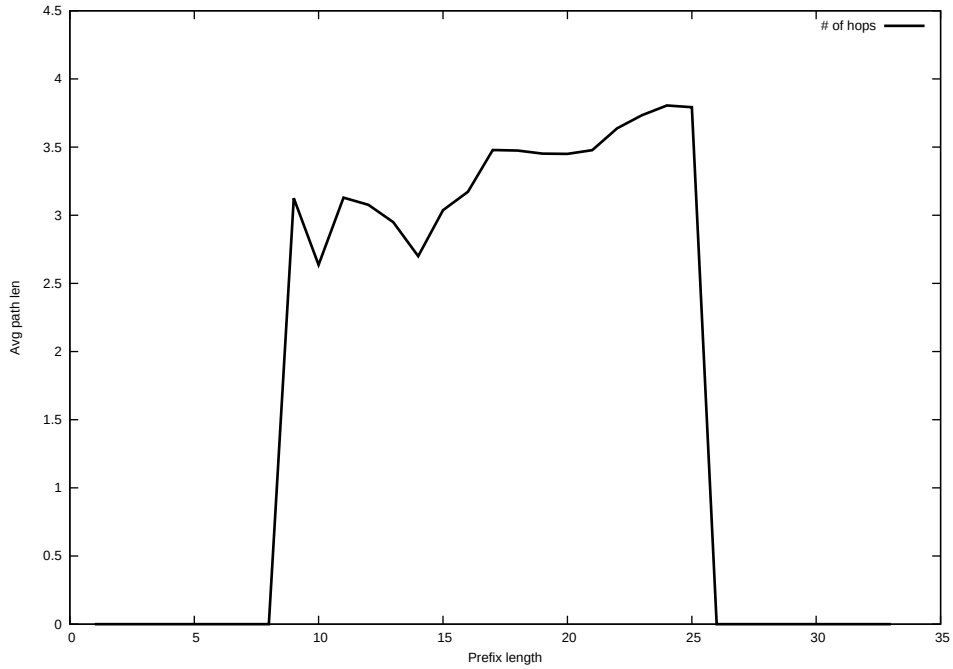
2014-01-02



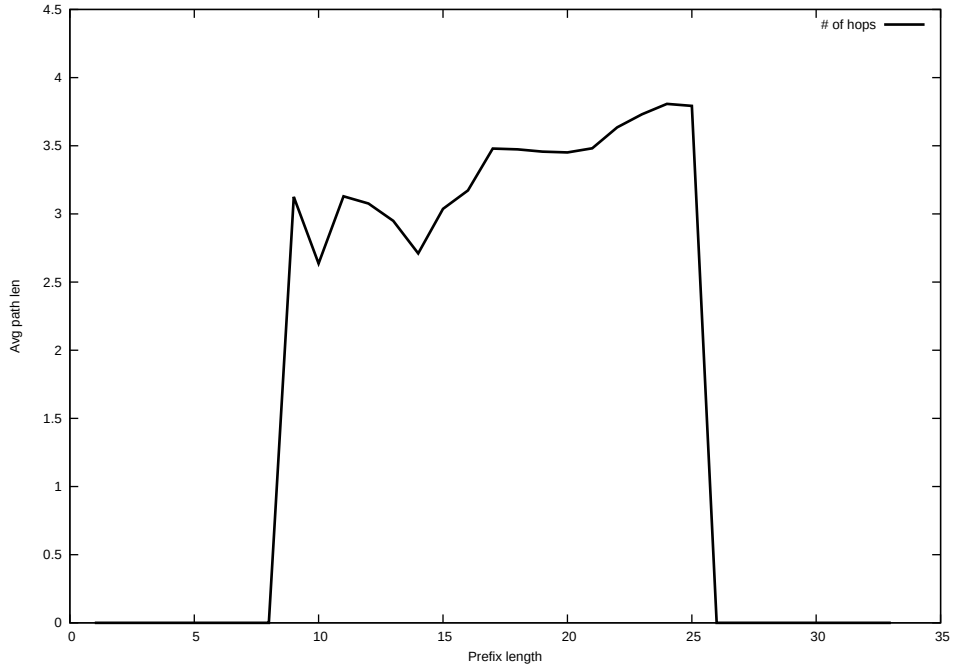
2014-01-03



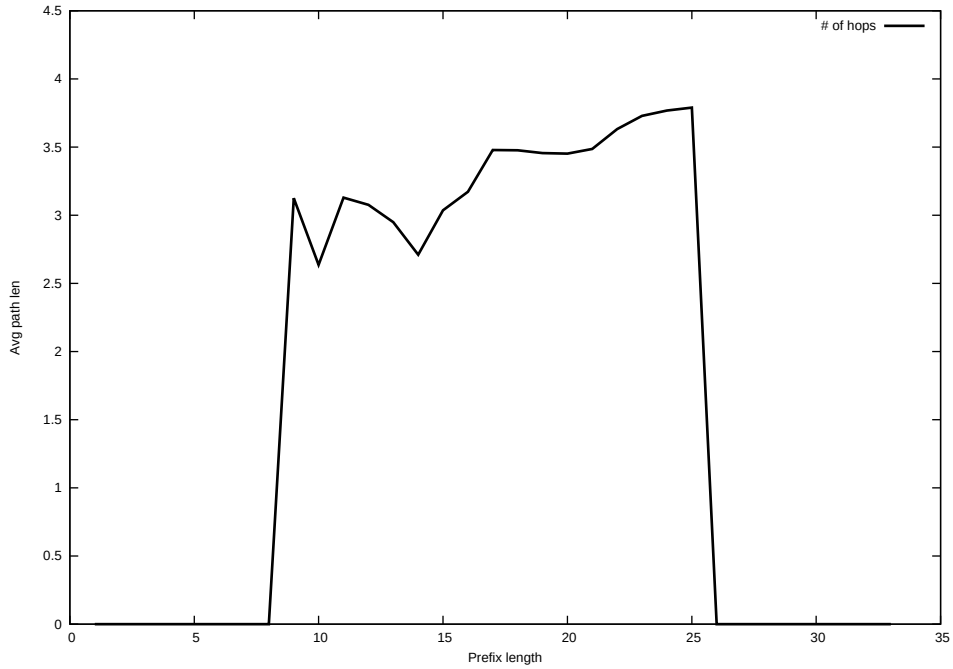
2014-01-04



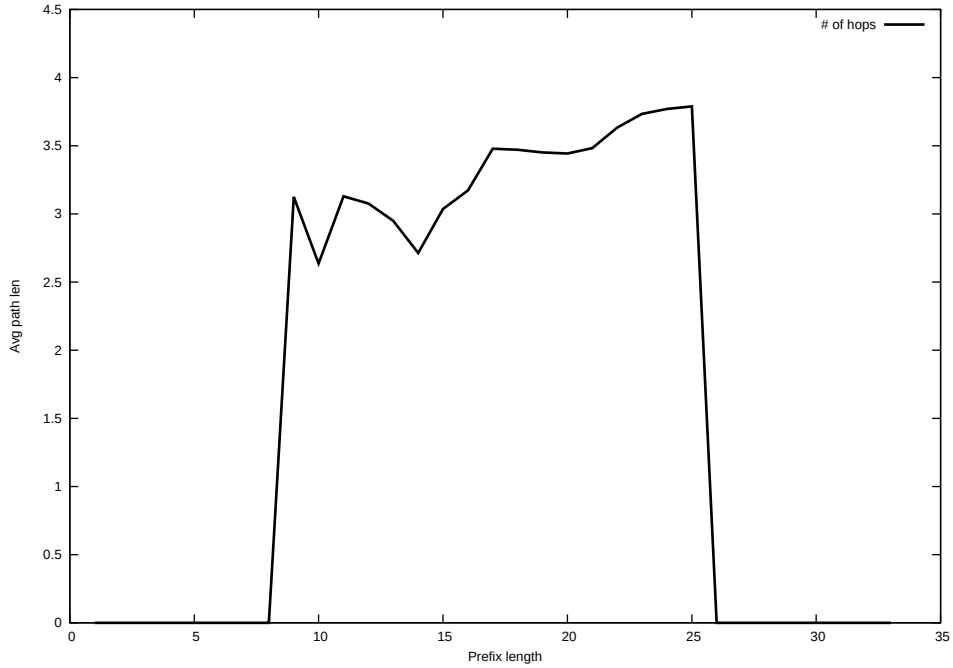
2014-01-05



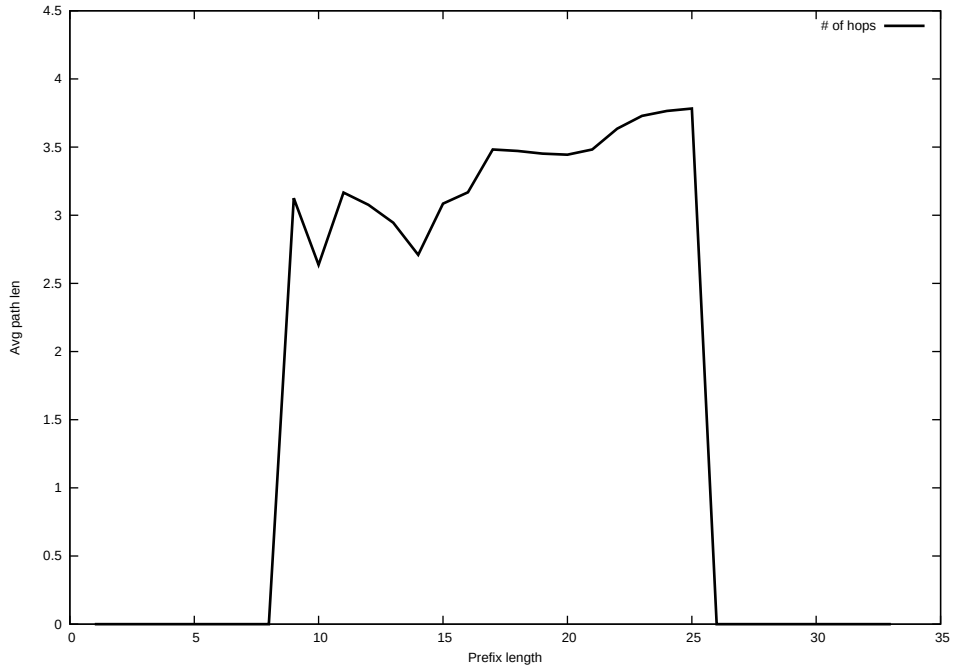
2014-01-06



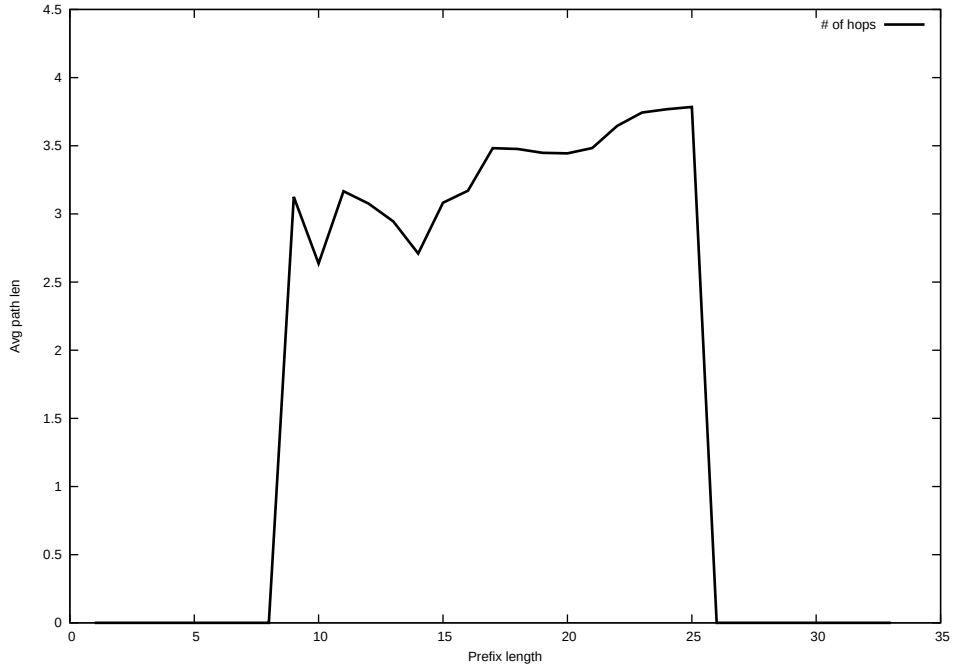
2014-01-07



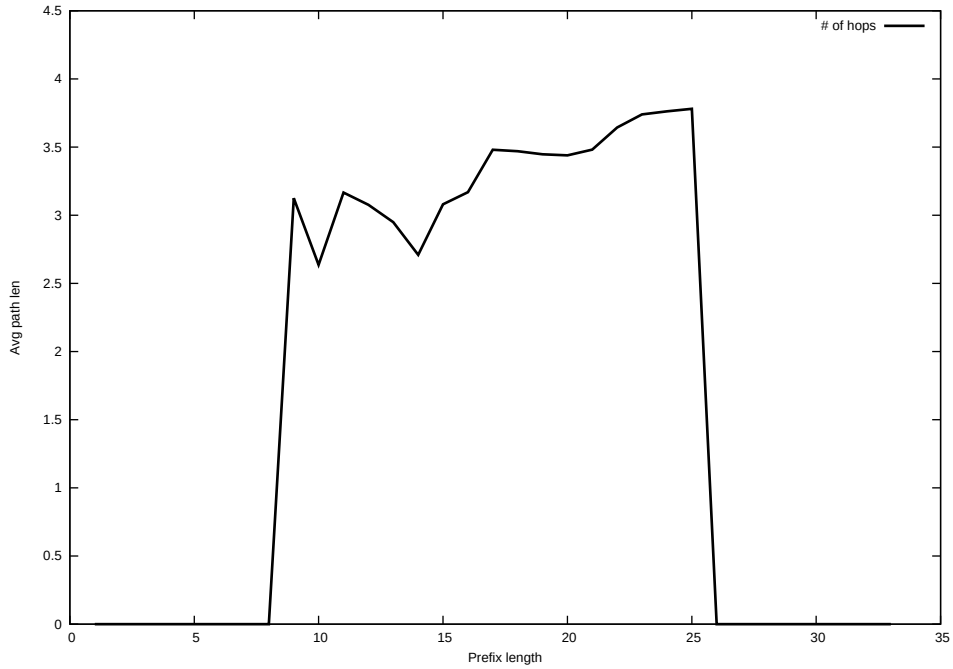
2014-01-08



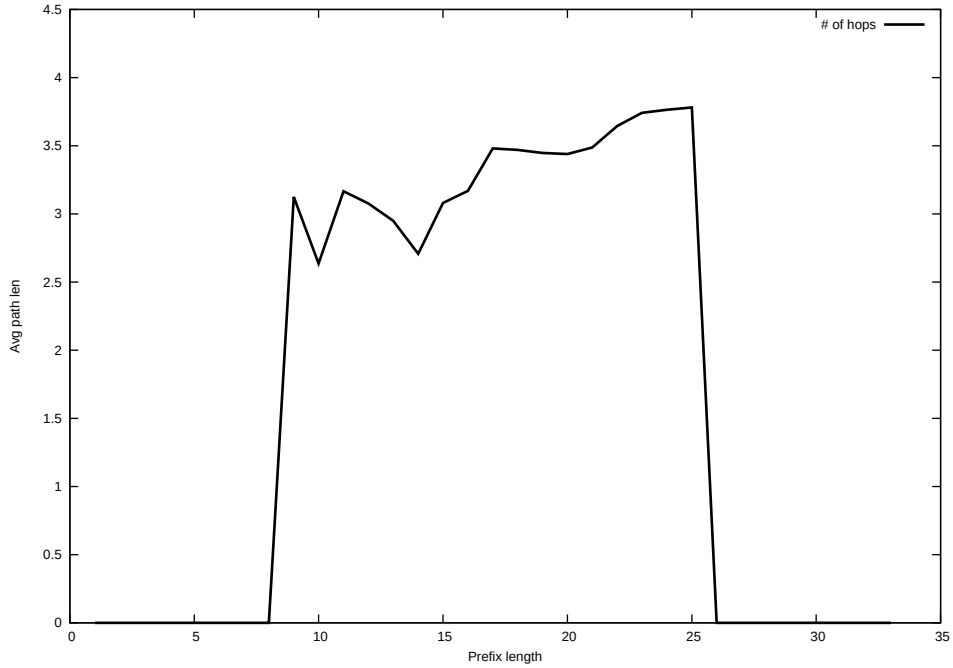
2014-01-09



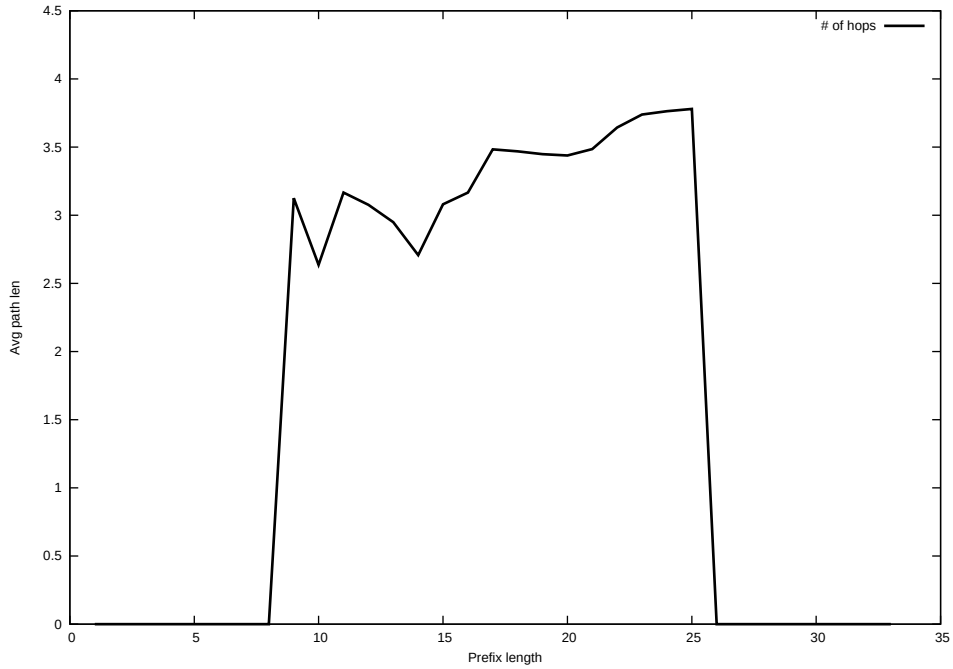
2014-01-10



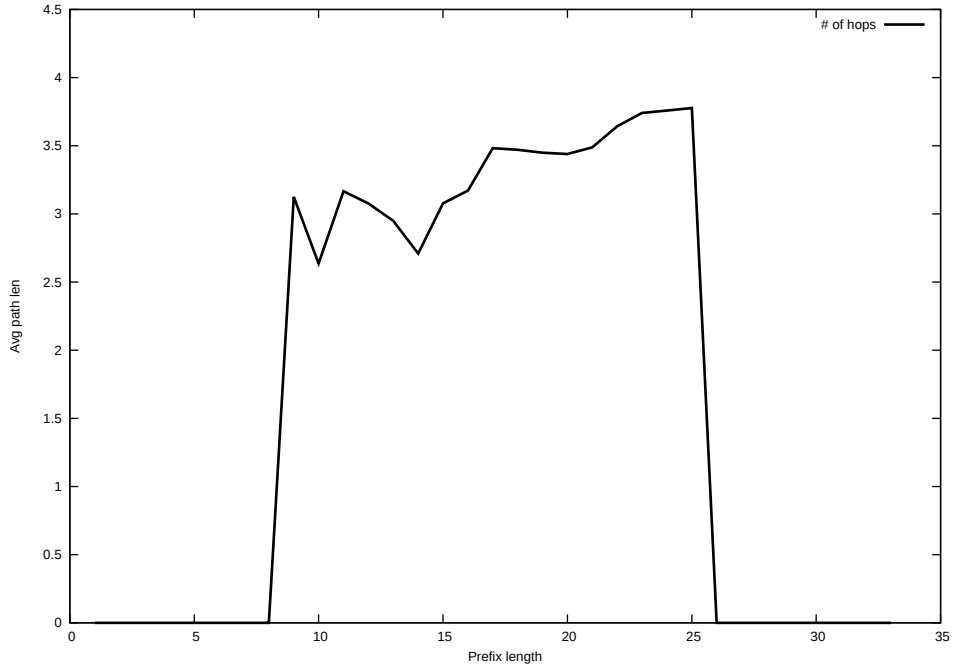
2014-01-11



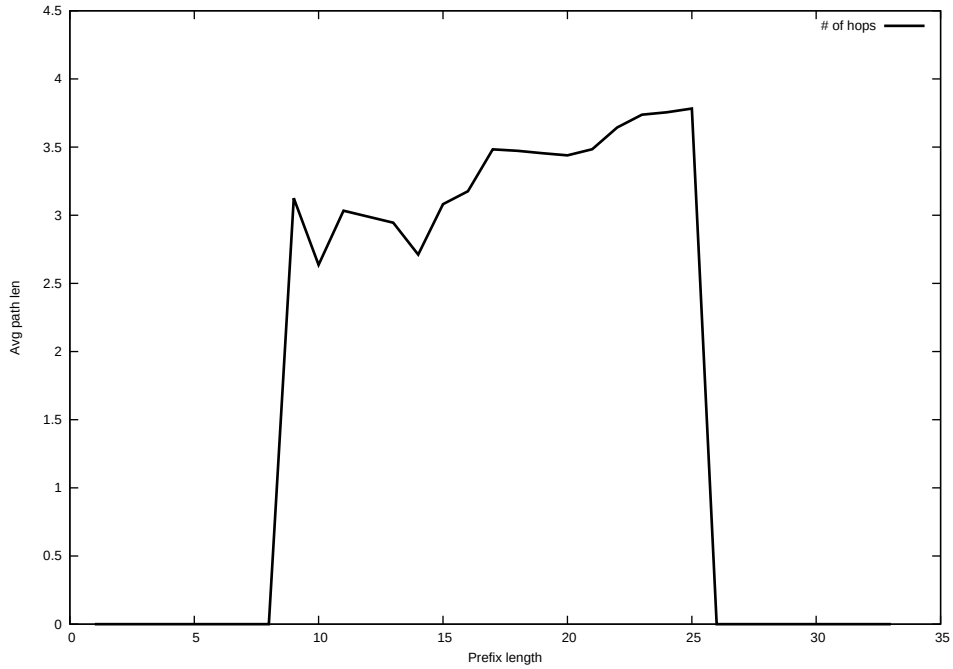
2014-01-12



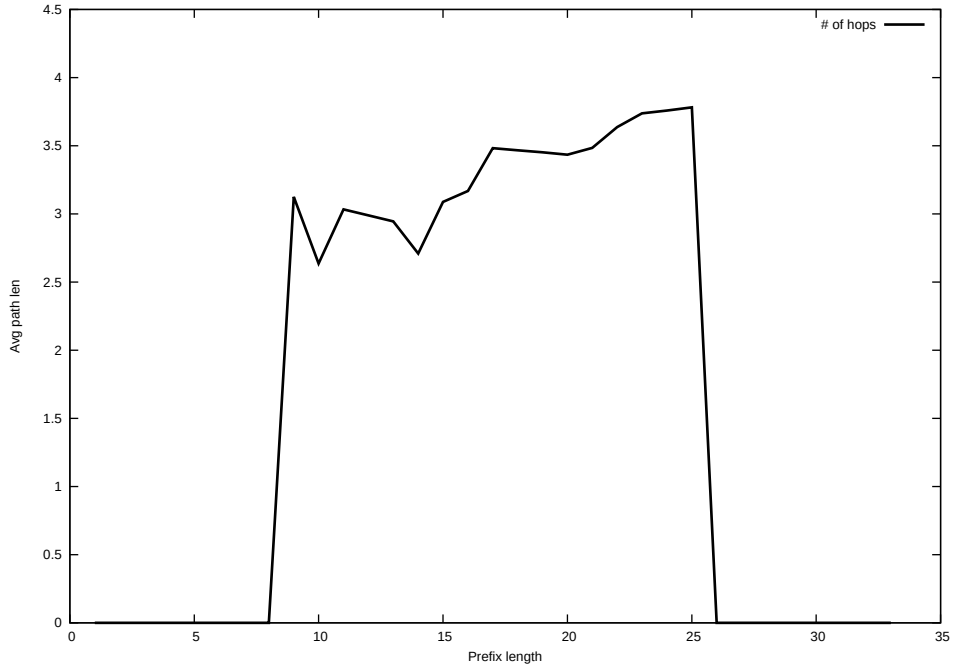
2014-01-13



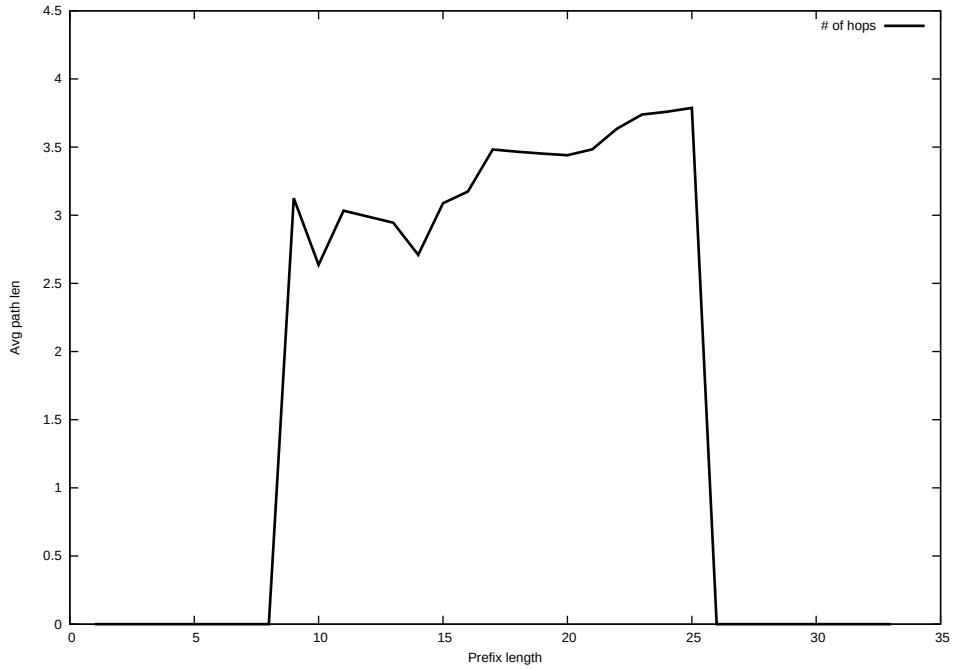
2014-01-14



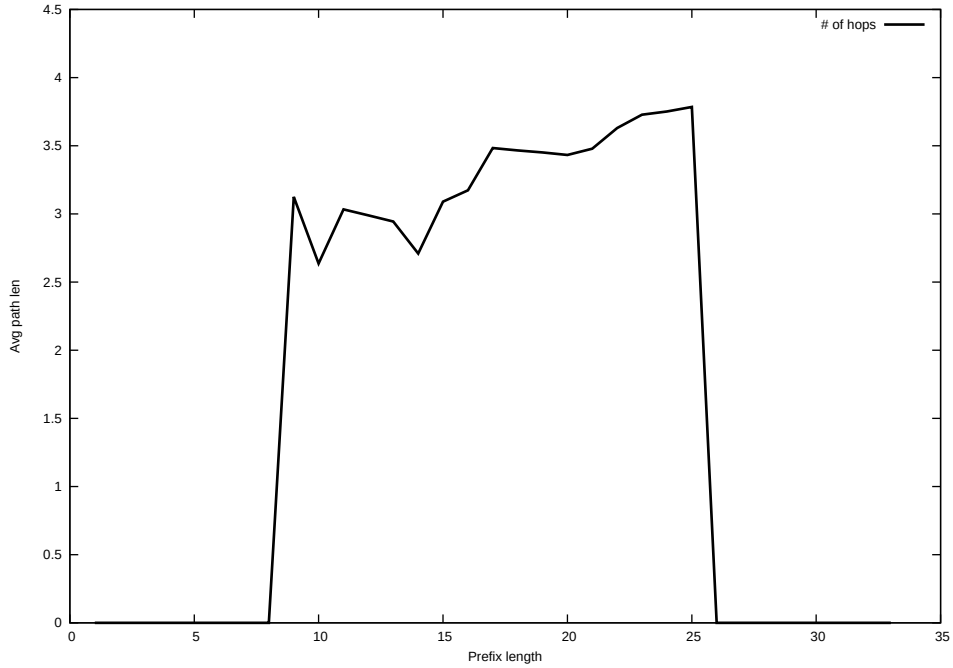
2014-01-15



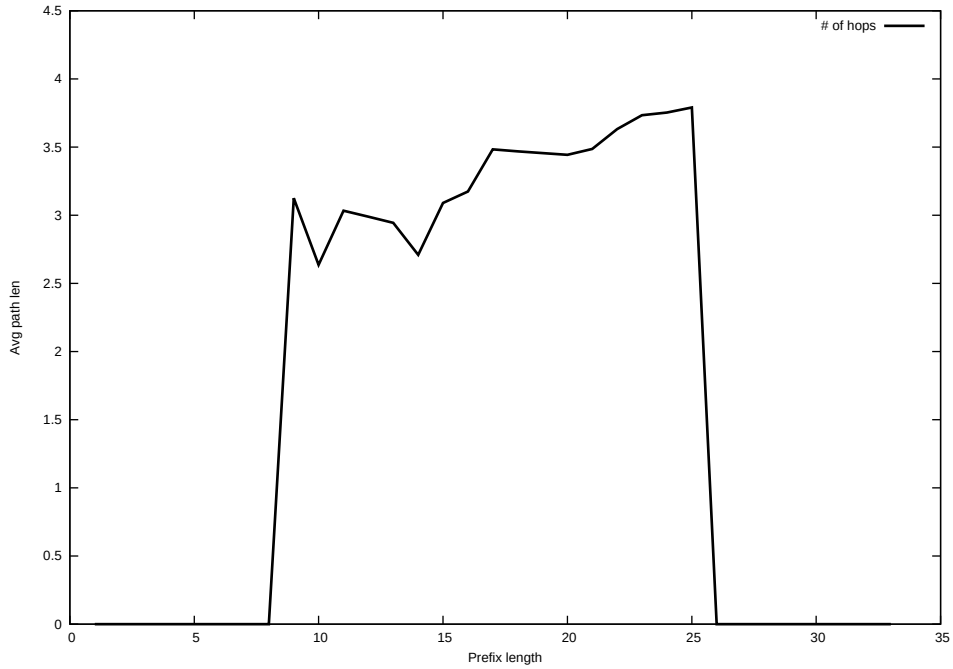
2014-01-16



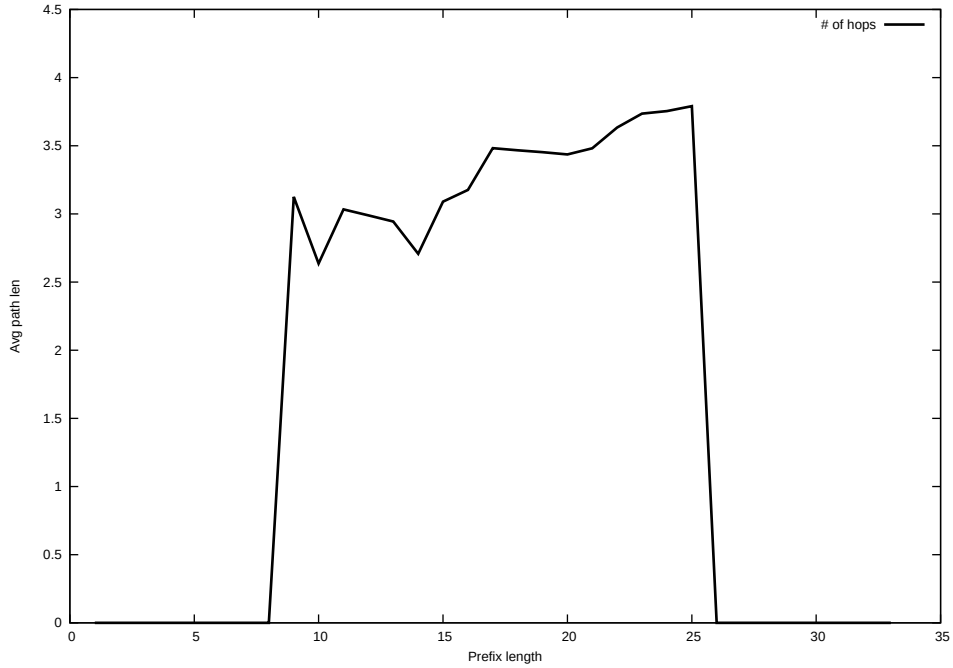
2014-01-17



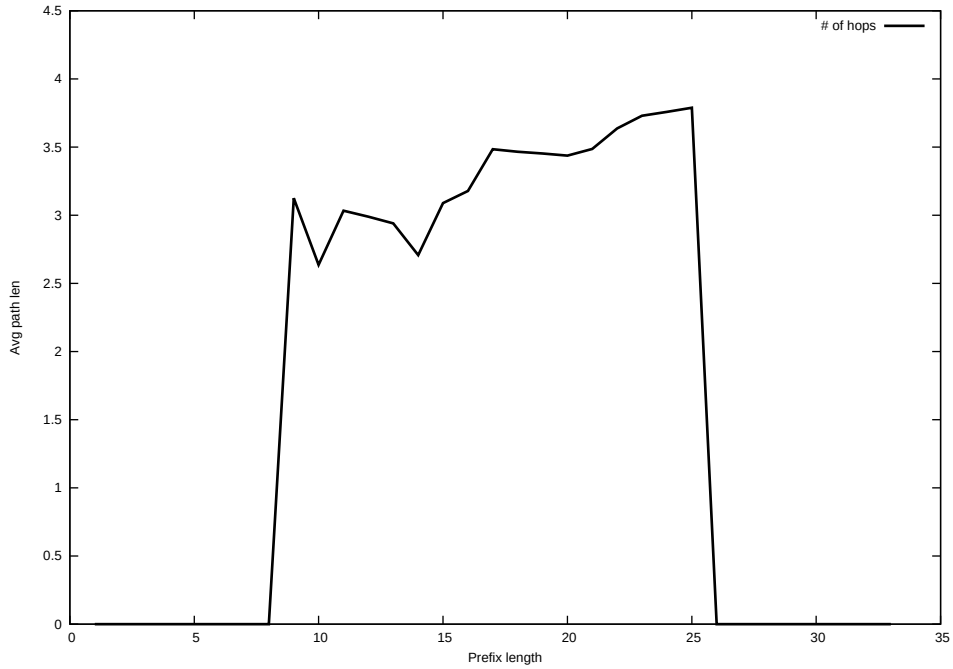
2014-01-18



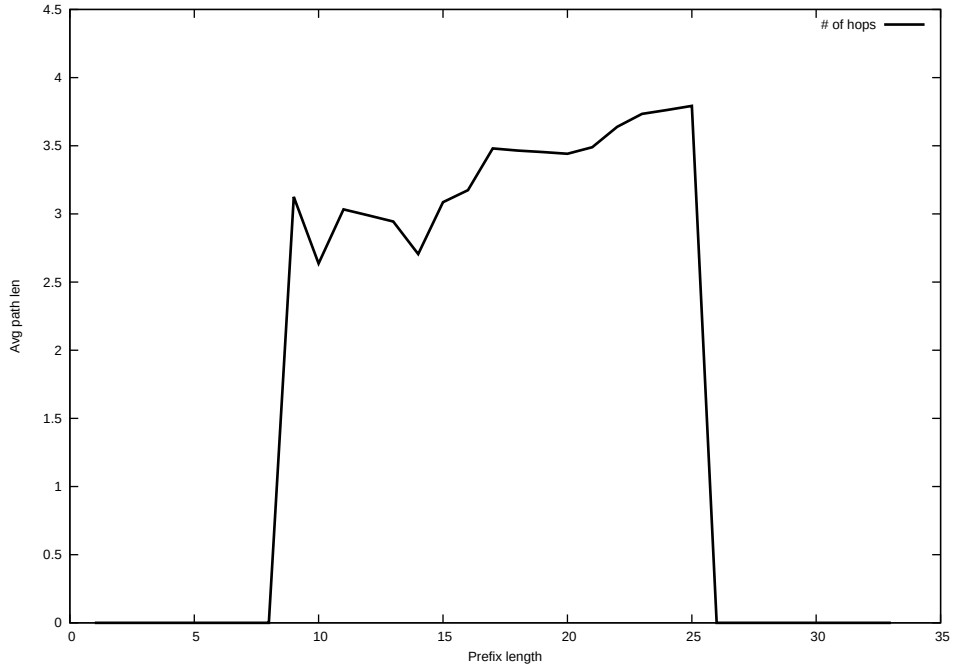
2014-01-19



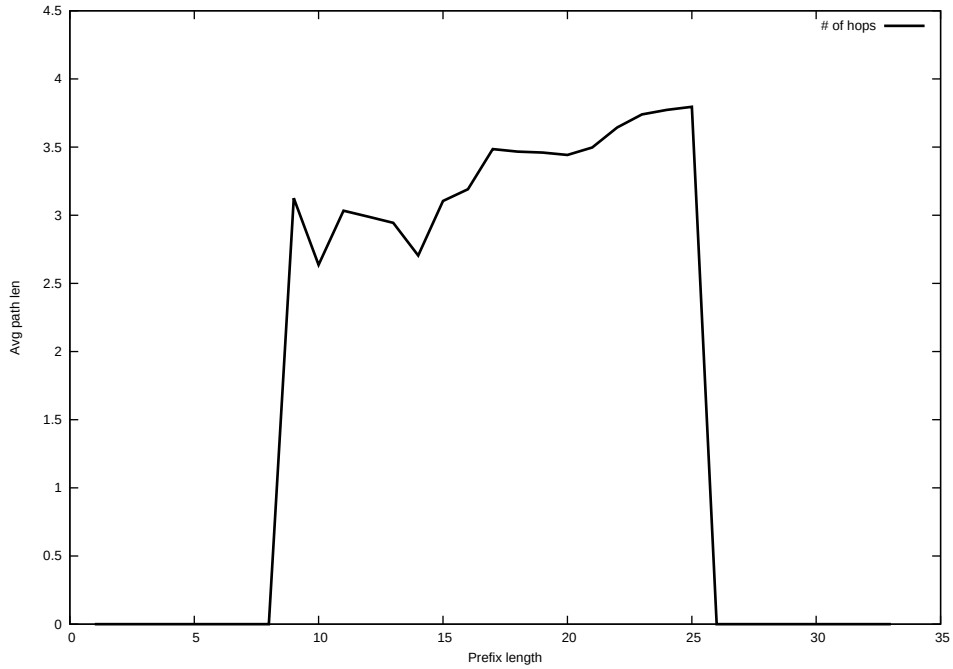
2014-01-20



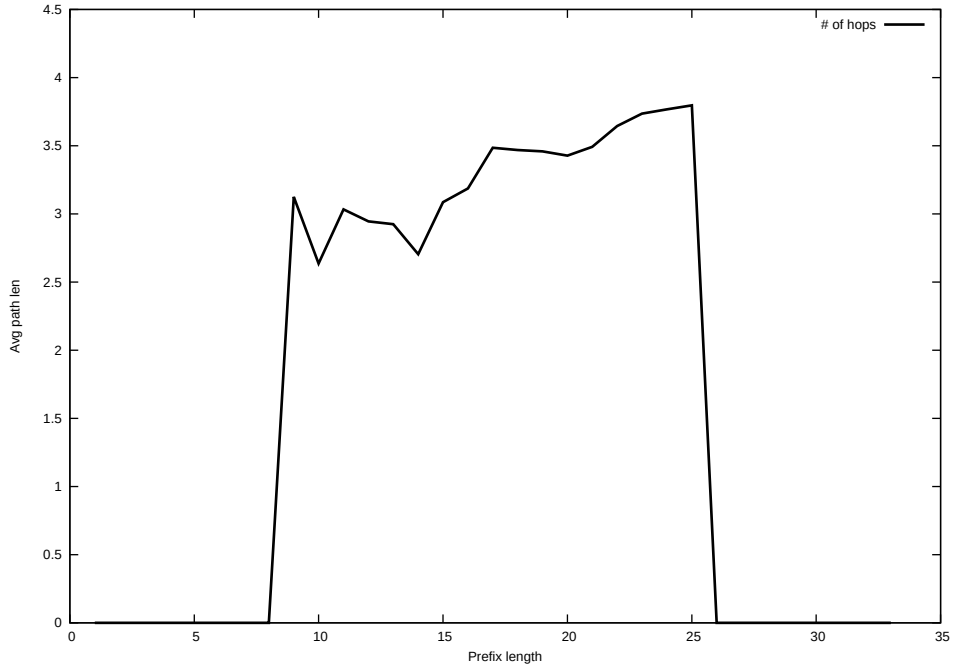
2014-01-21



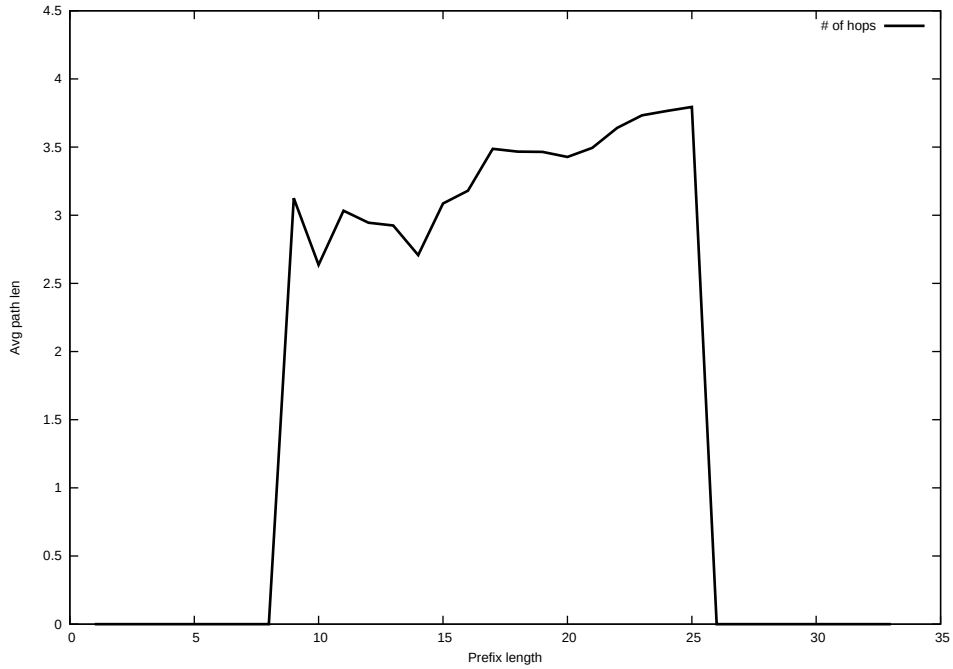
2014-01-22



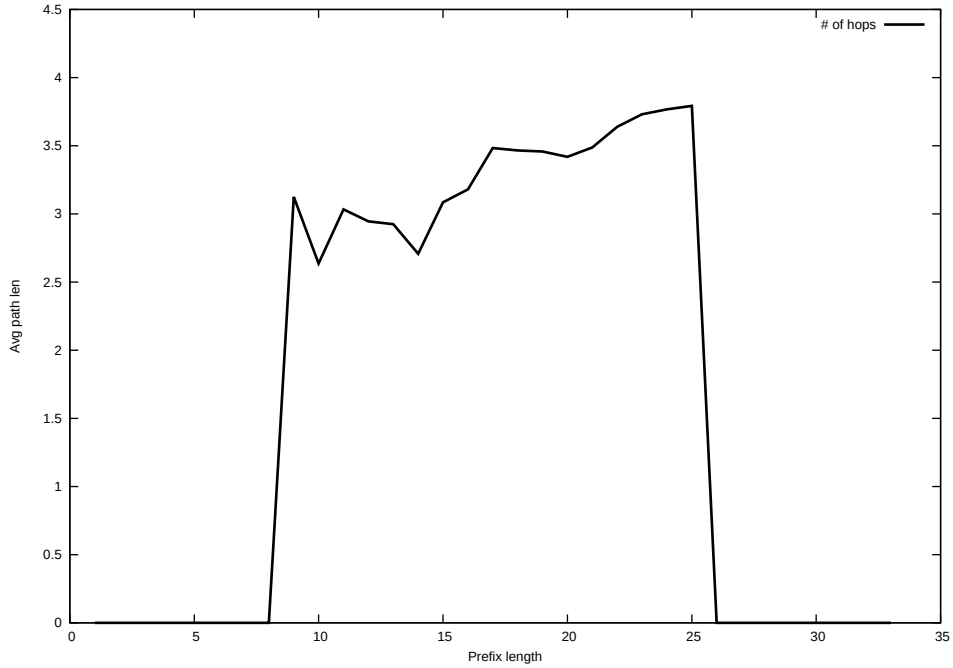
2014-01-23



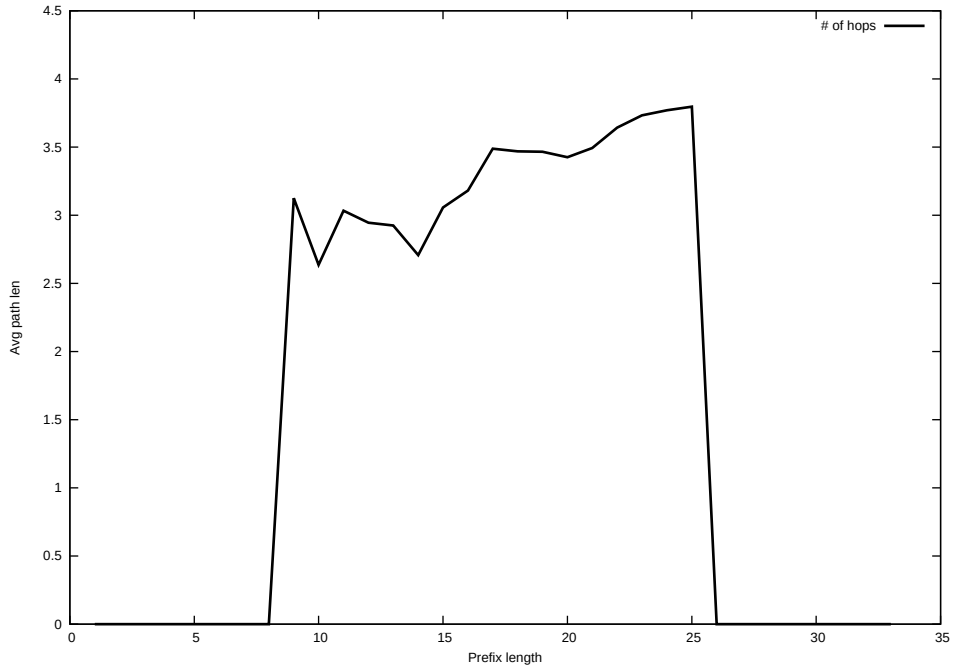
2014-01-24



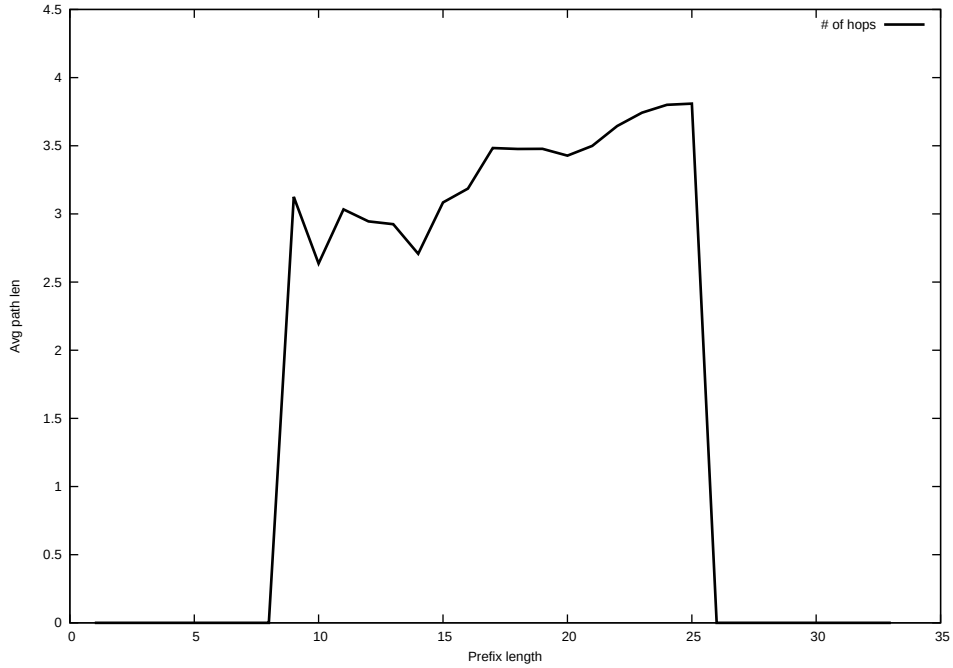
2014-01-25



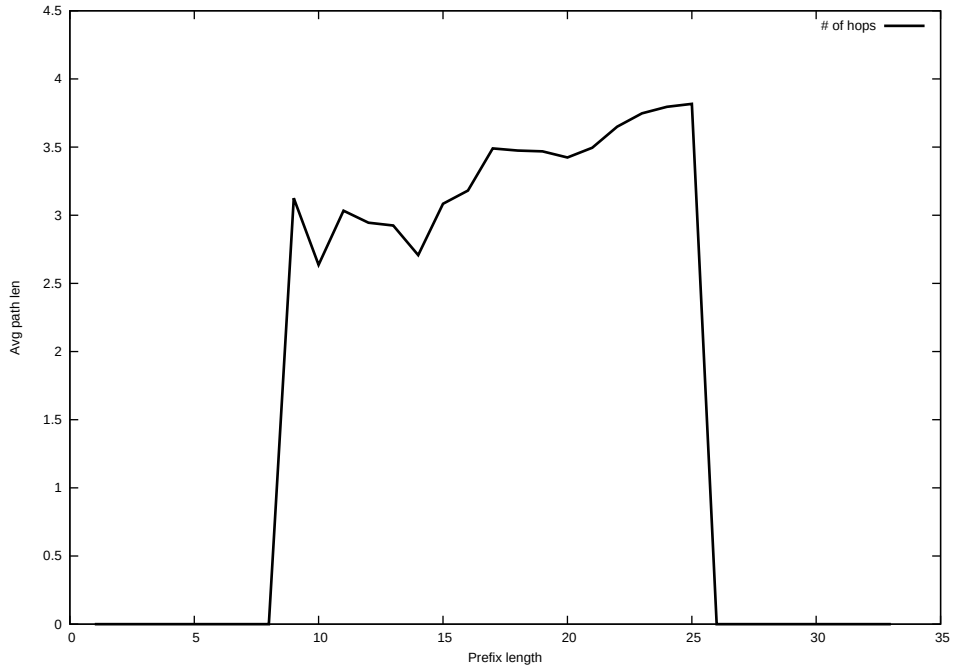
2014-01-26



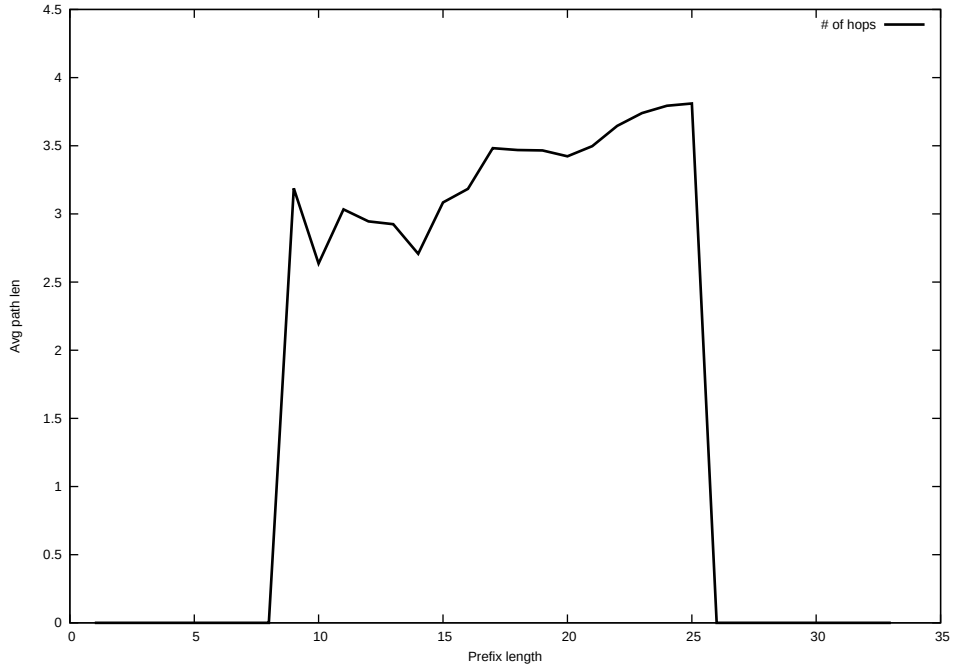
2014-01-27



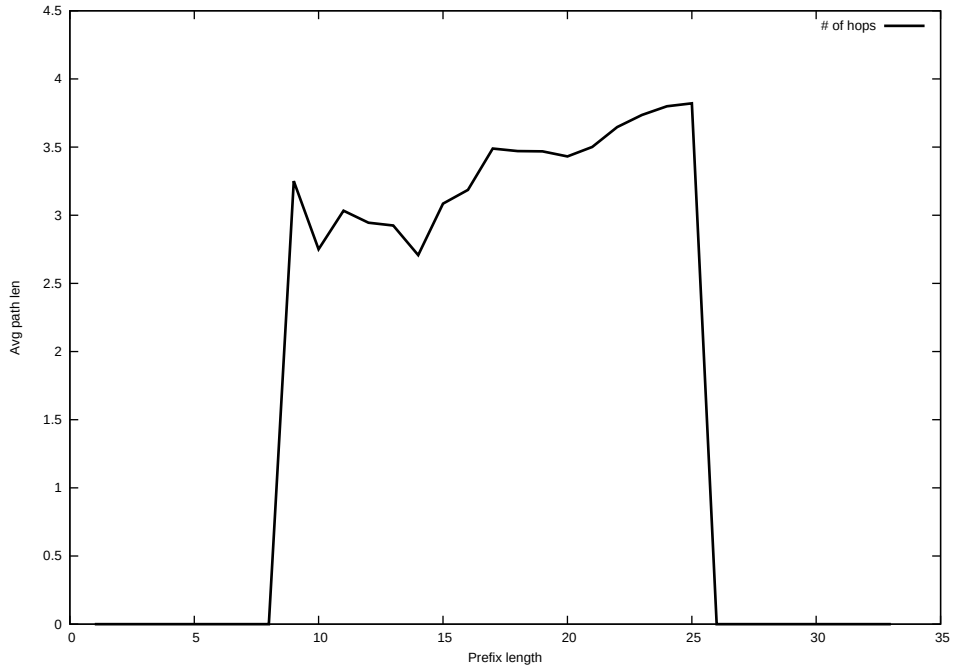
2014-01-28



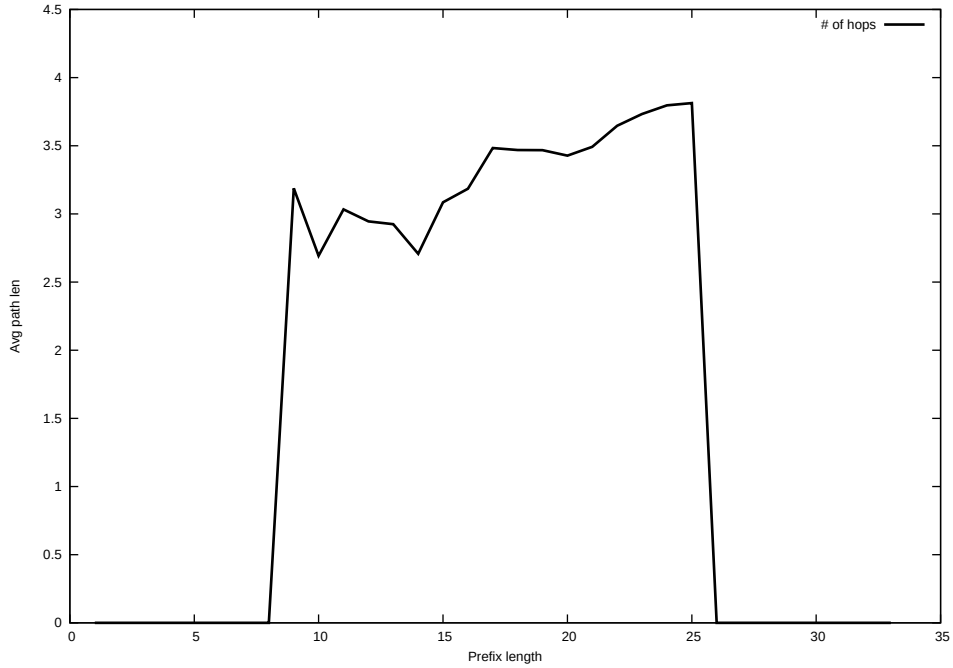
2014-01-29



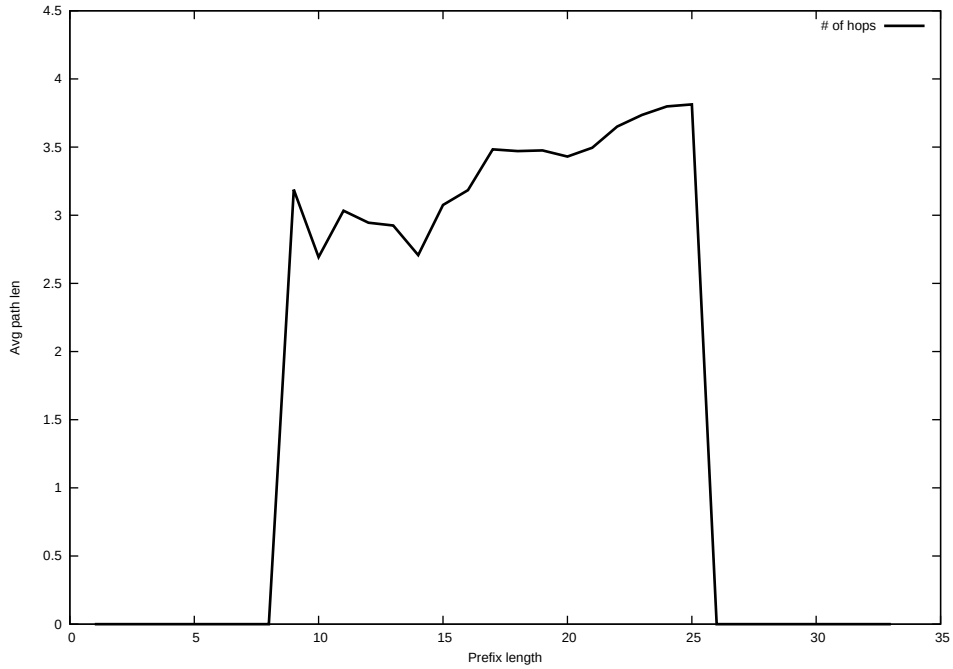
2014-01-30



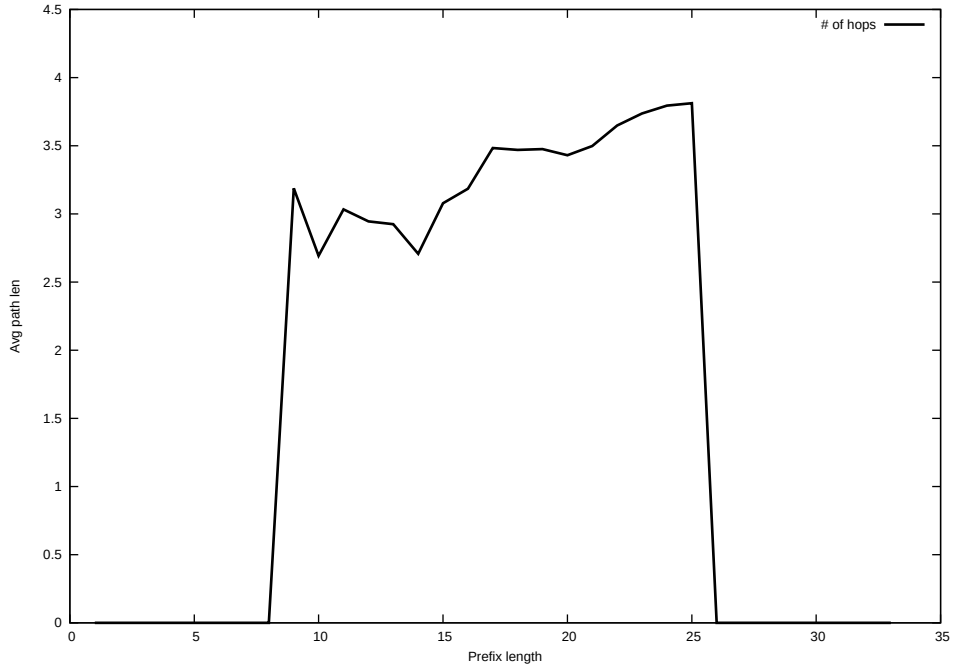
2014-01-31



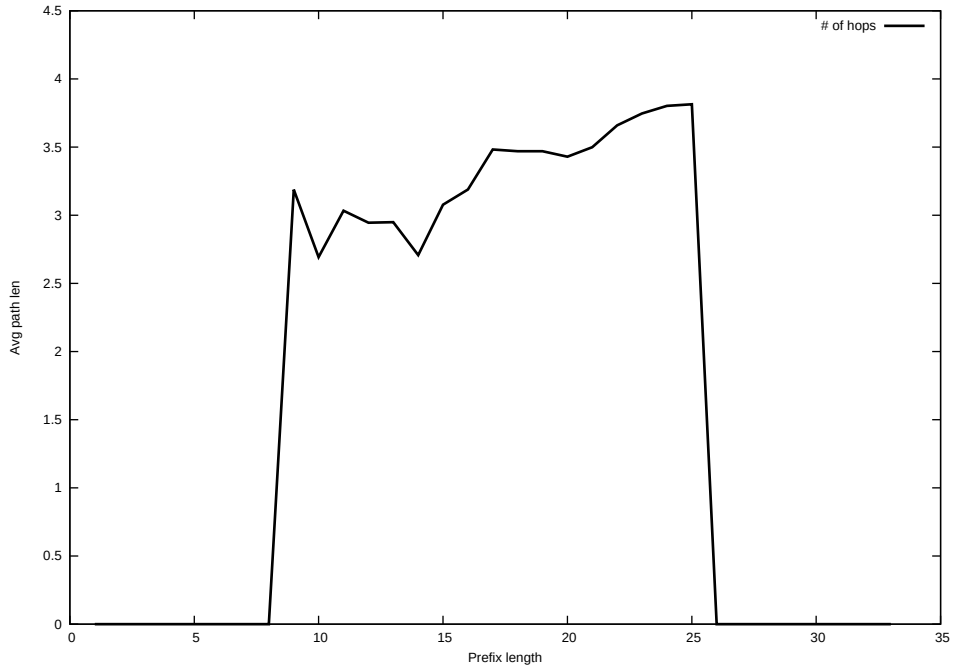
2014-02-01



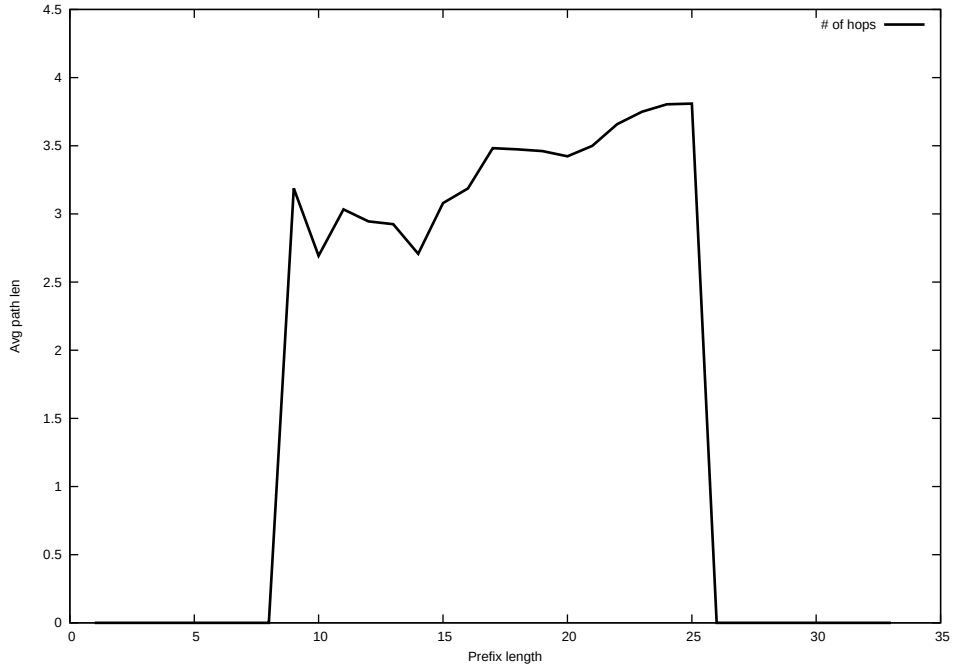
2014-02-02



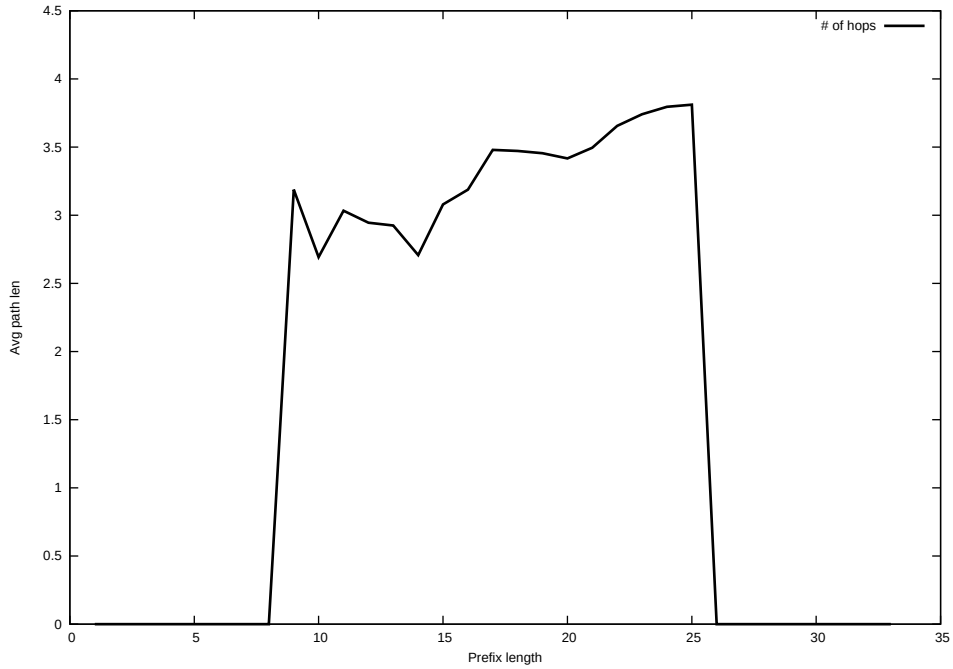
2014-02-03



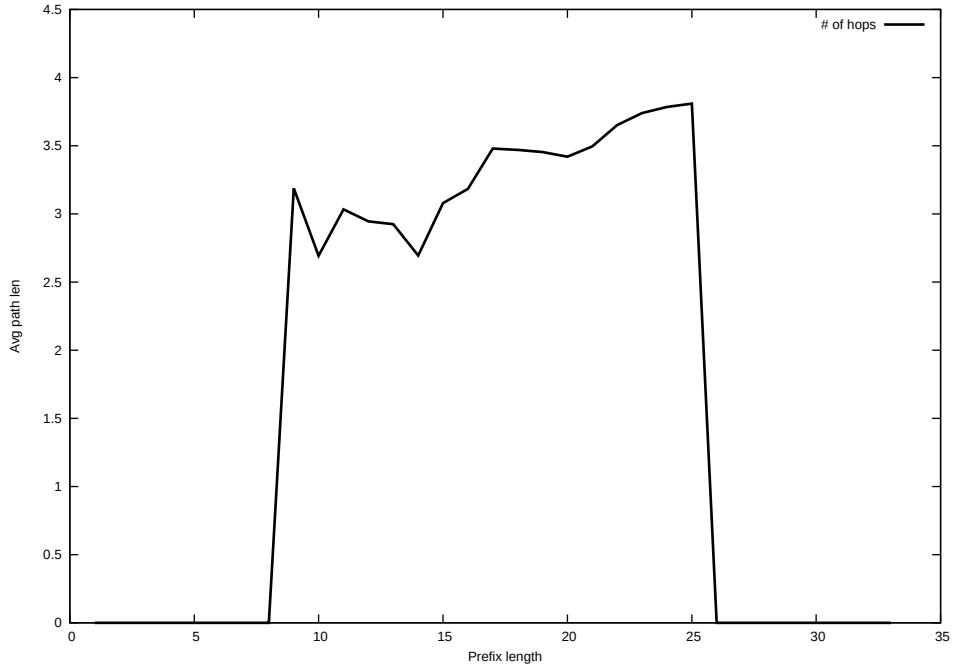
2014-02-04



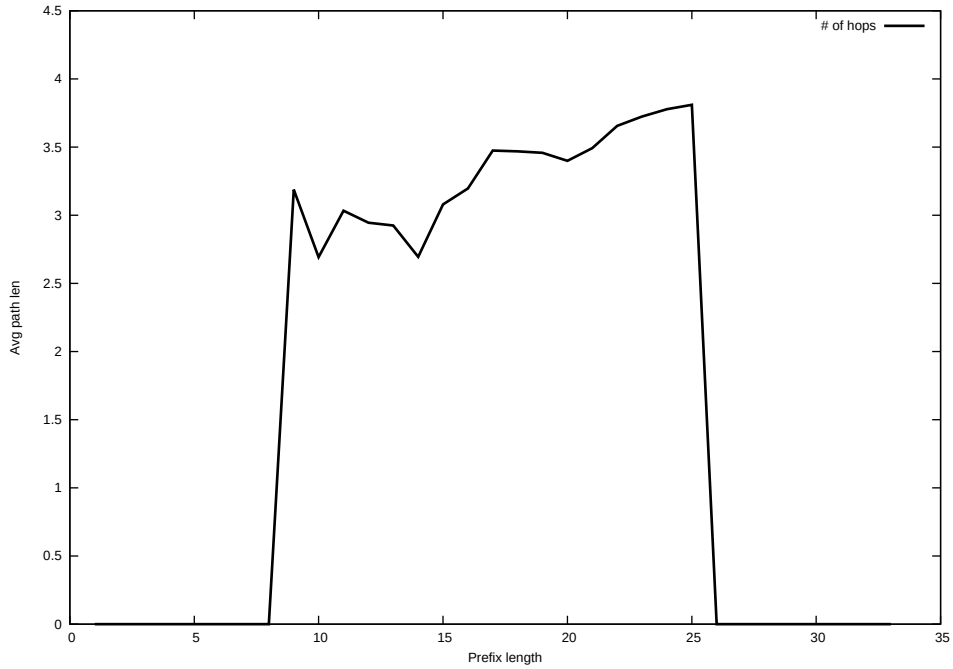
2014-02-05



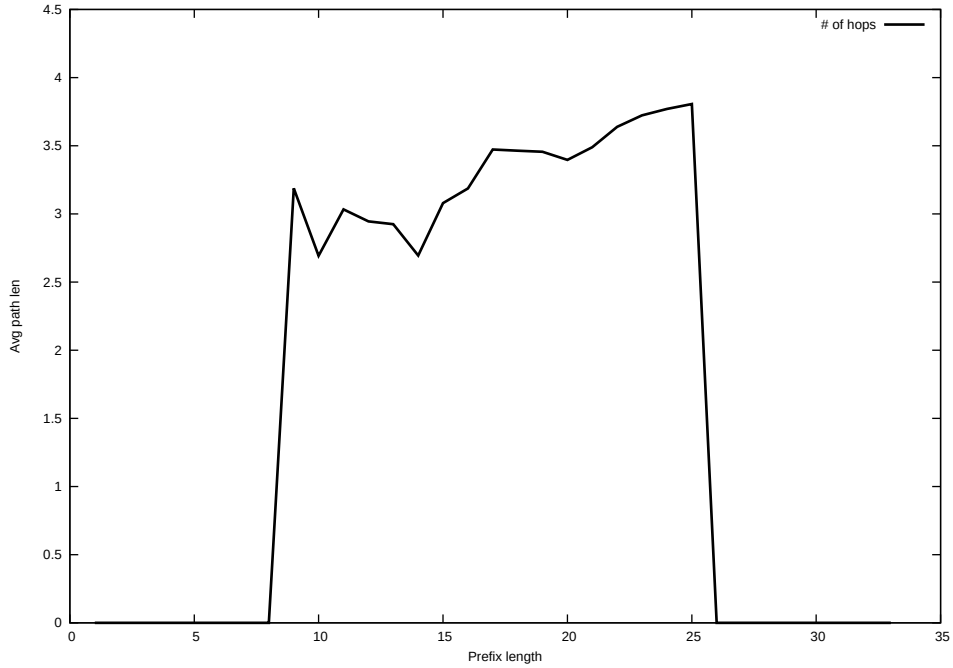
2014-02-06



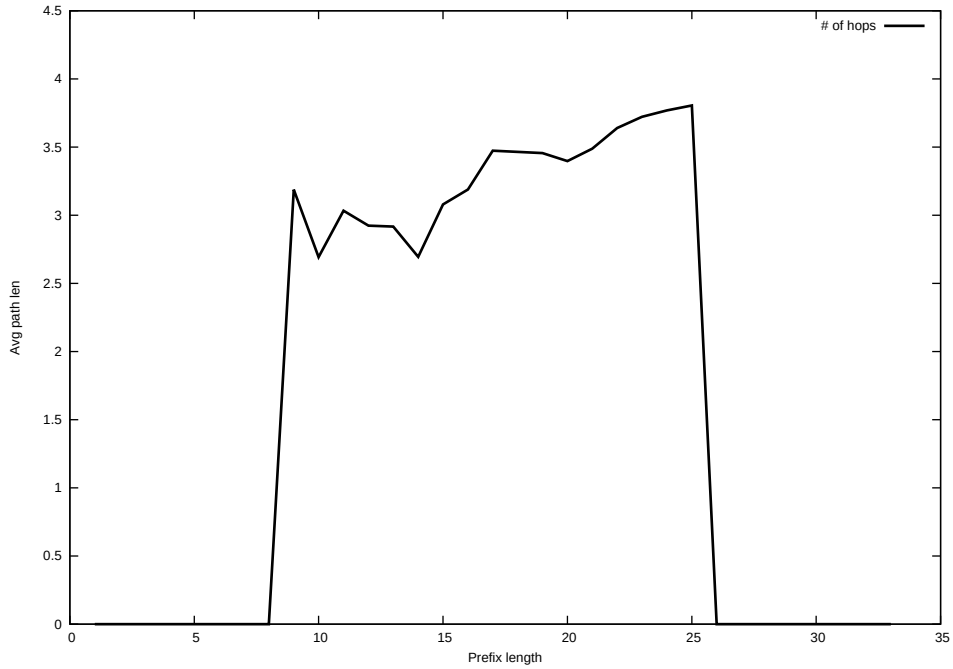
2014-02-07



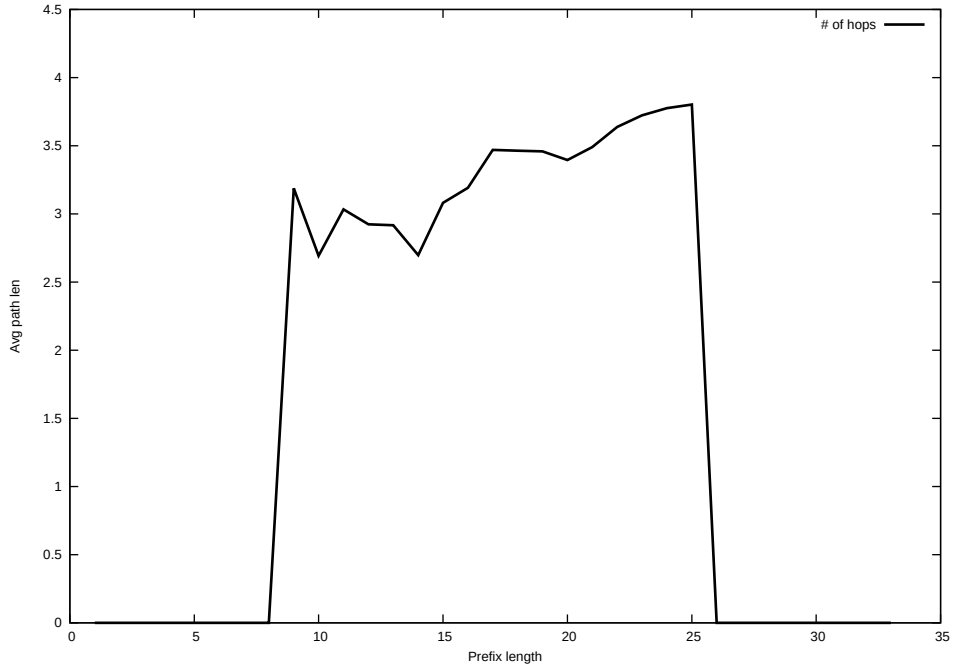
2014-02-08



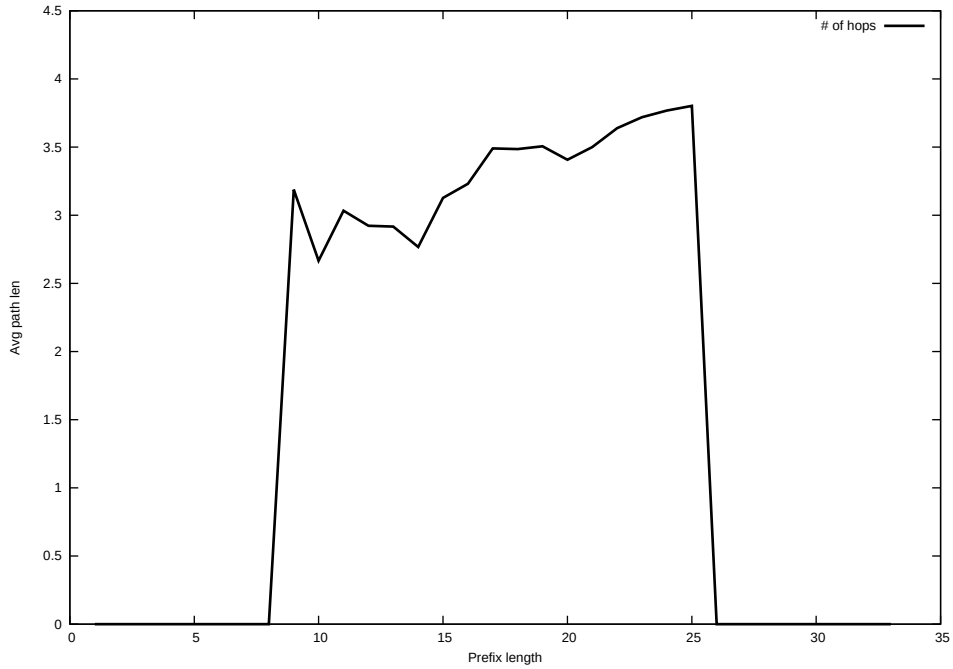
2014-02-09



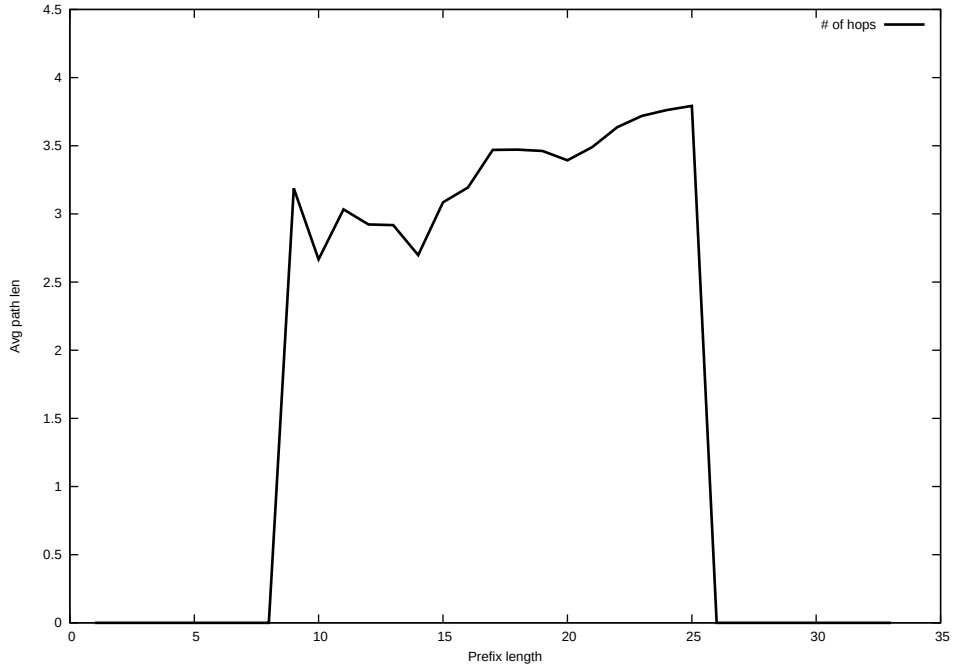
2014-02-10



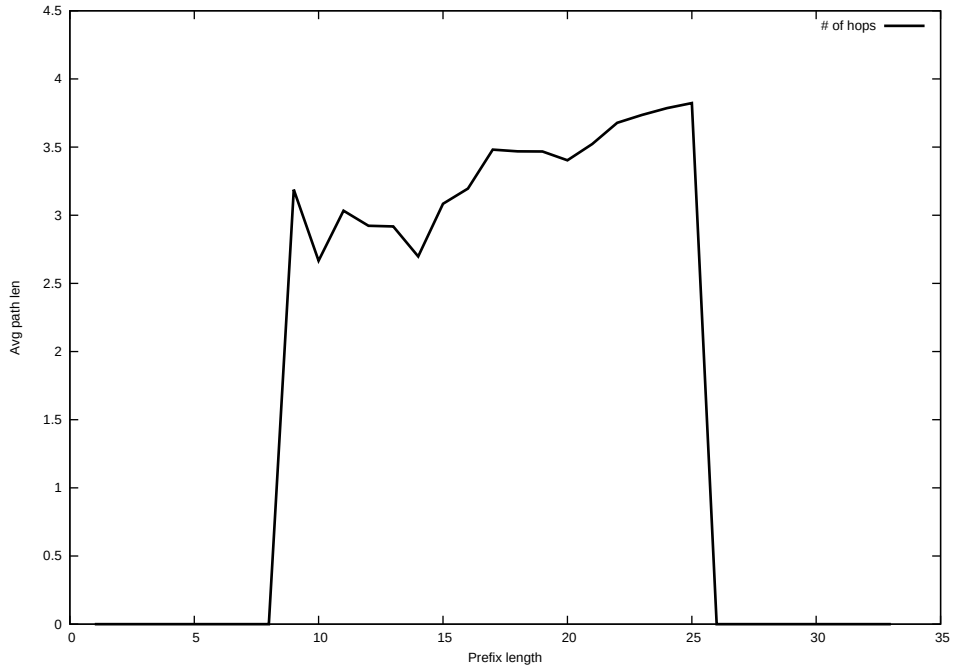
2014-02-11



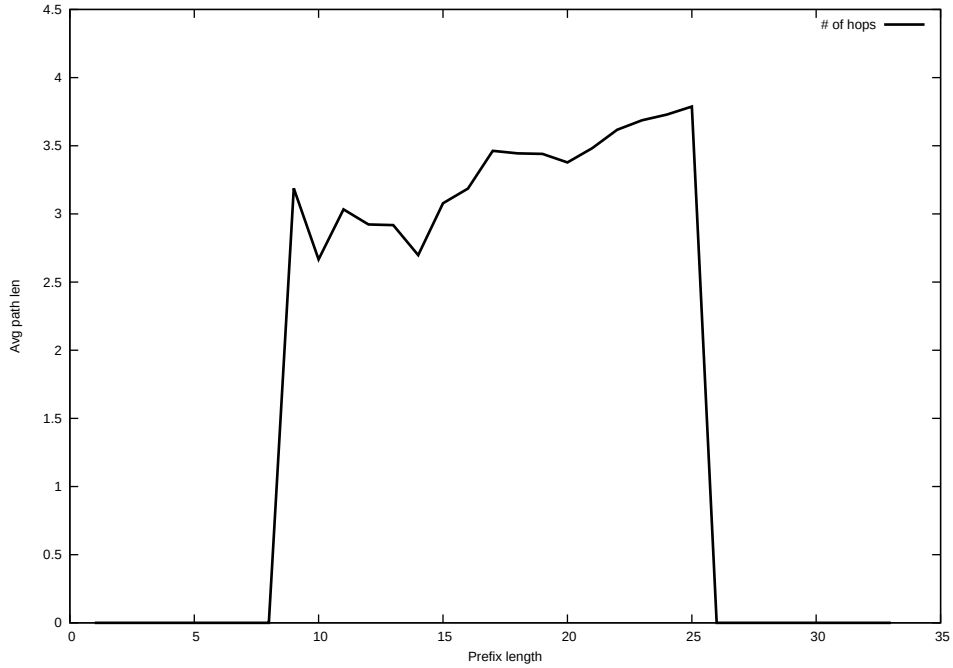
2014-02-12



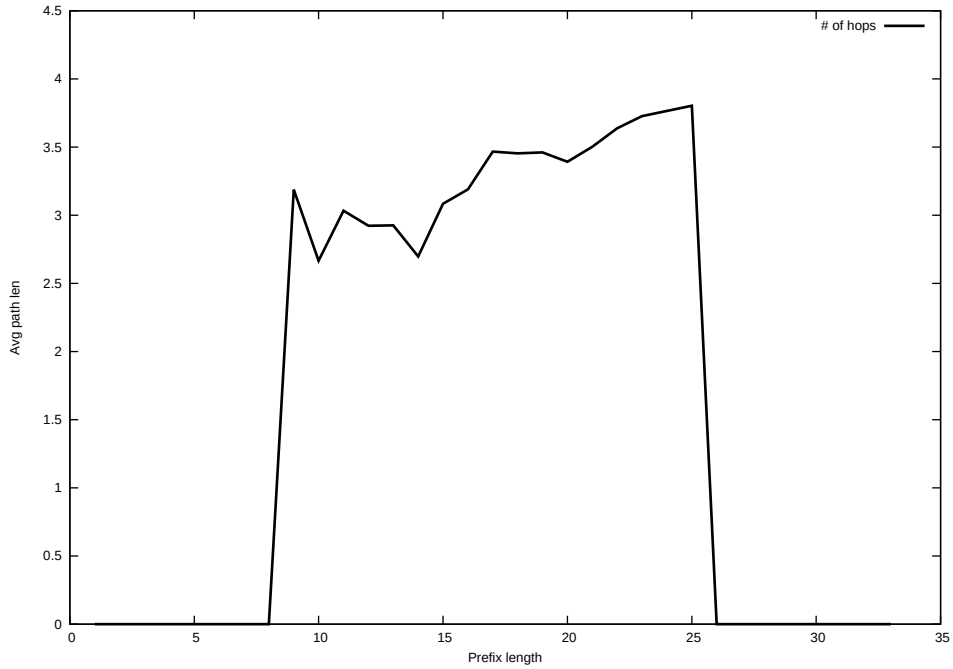
2014-02-13



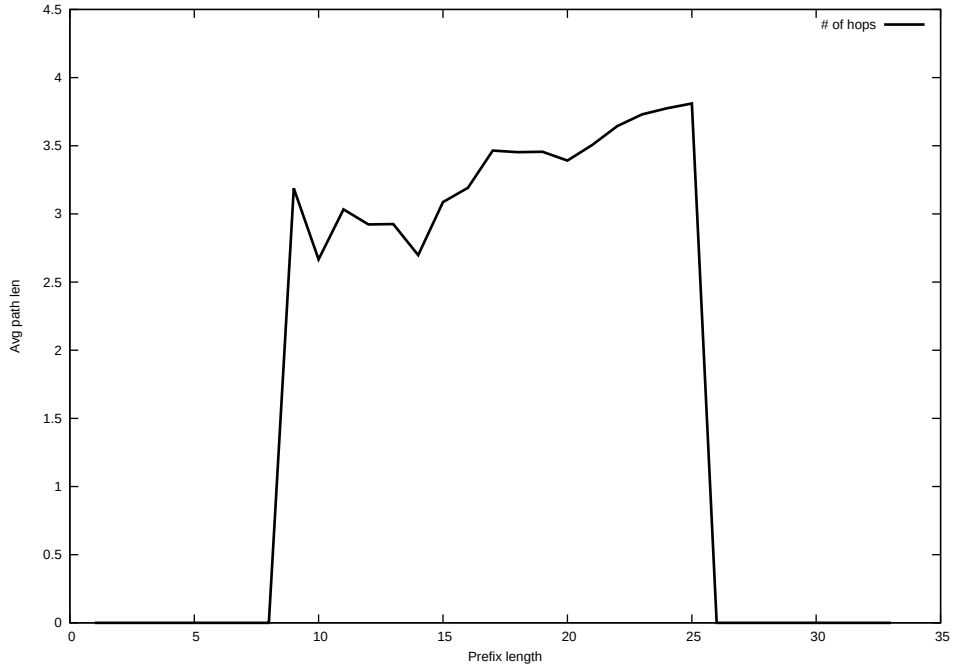
2014-02-14



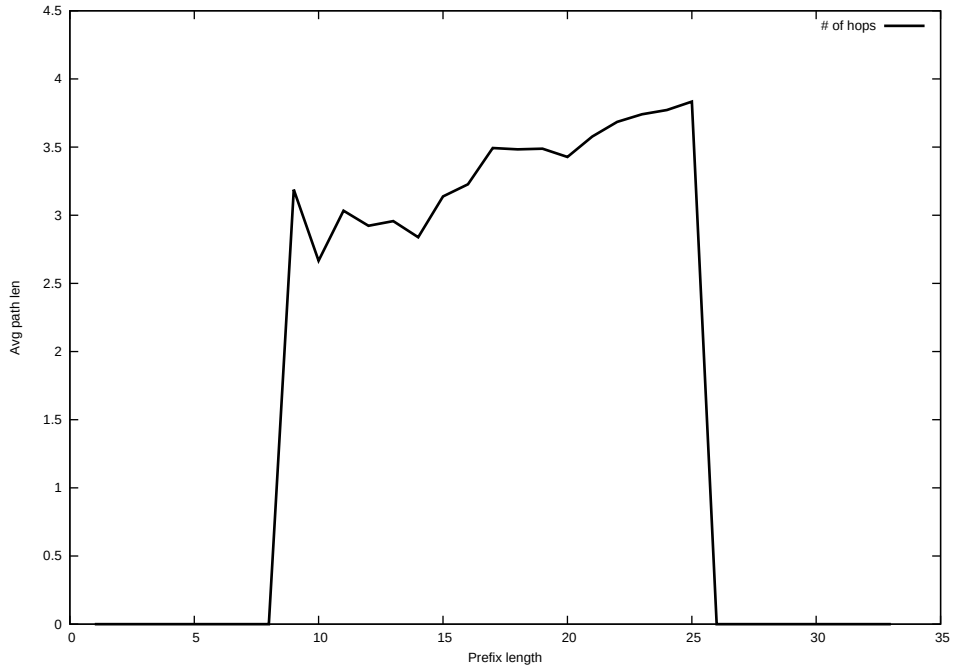
2014-02-15



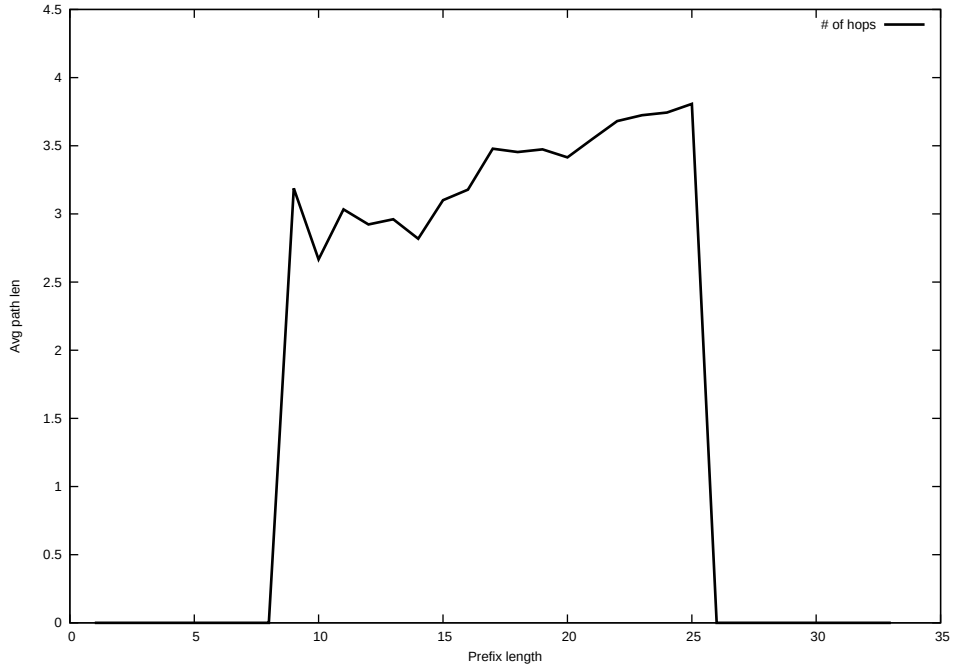
2014-02-16



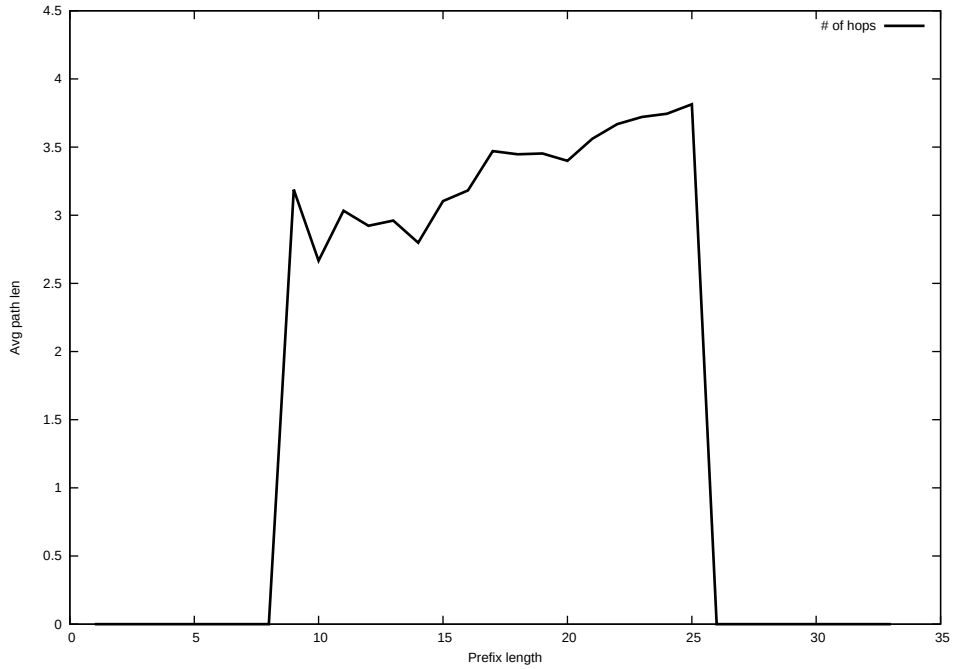
2014-02-17



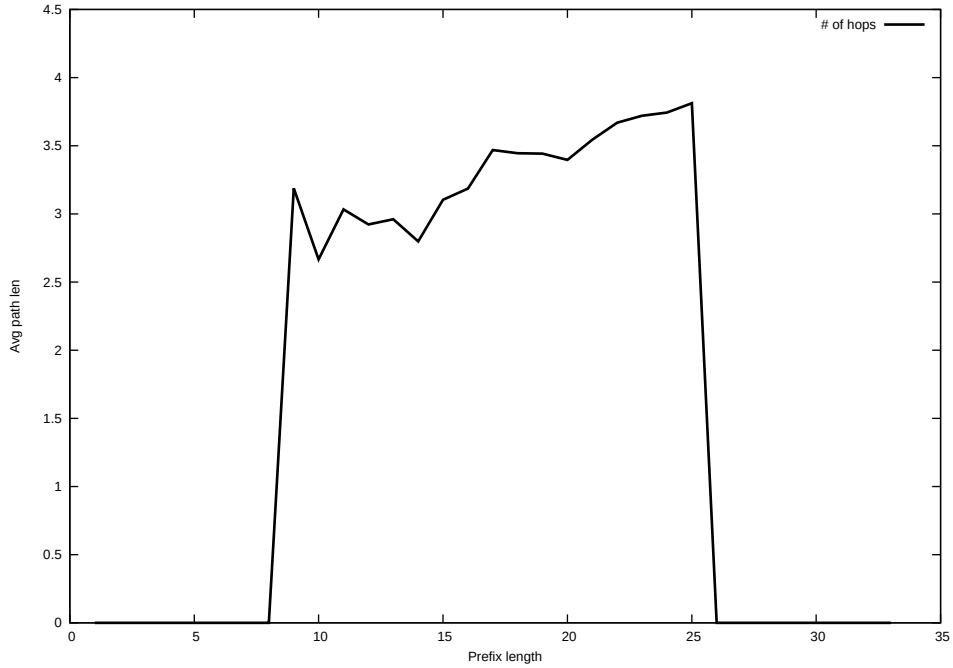
2014-03-20



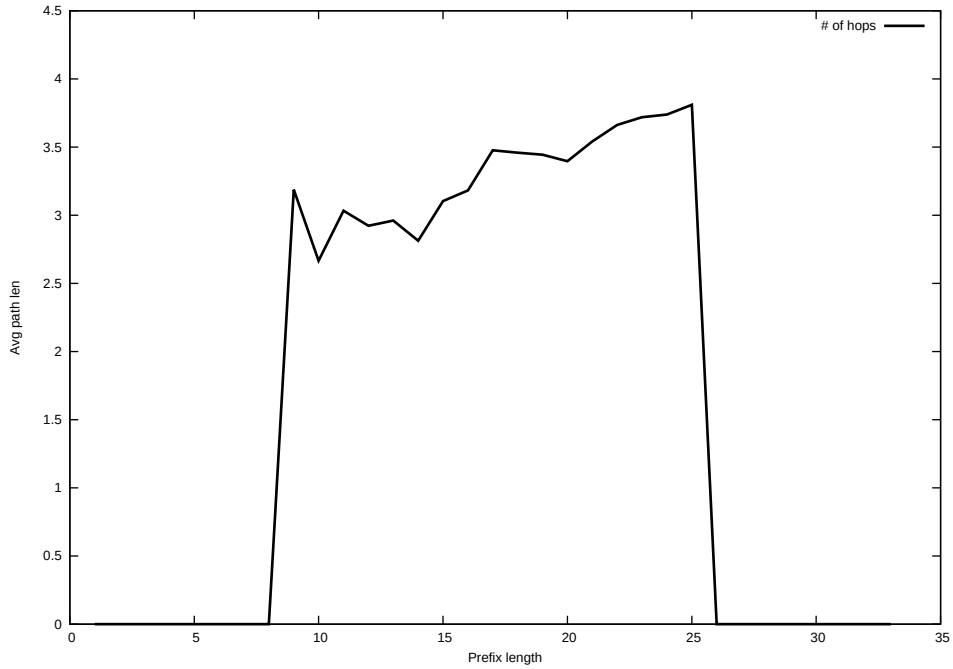
2014-03-21



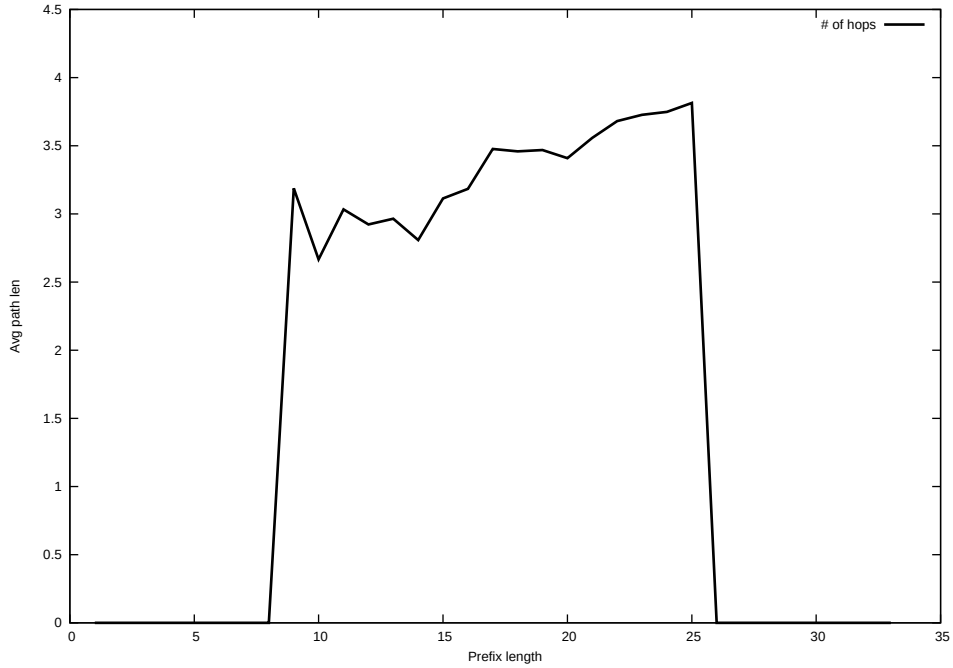
2014-03-22



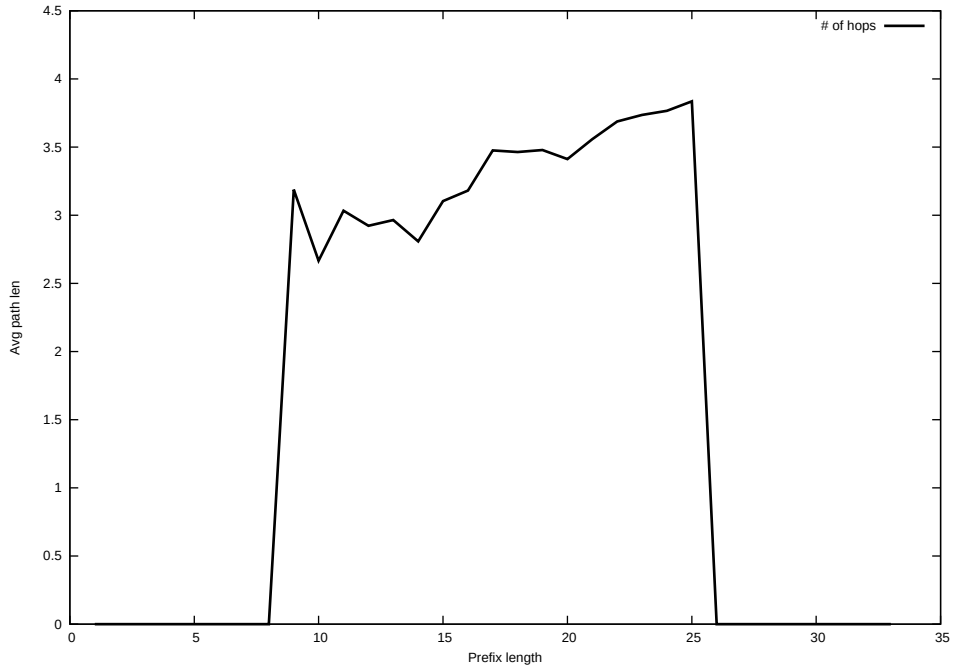
2014-03-23



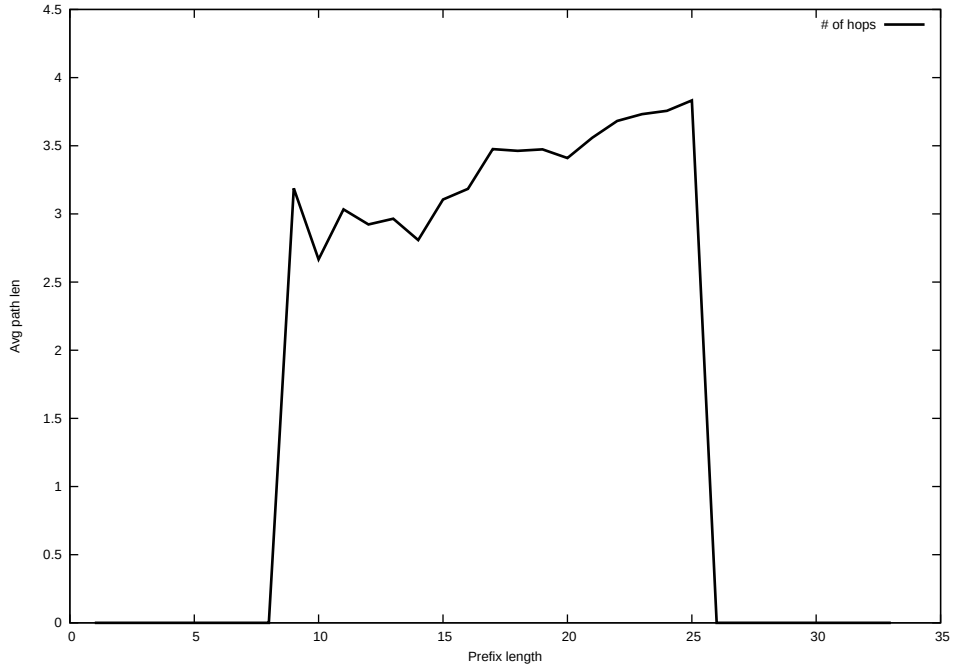
2014-03-24



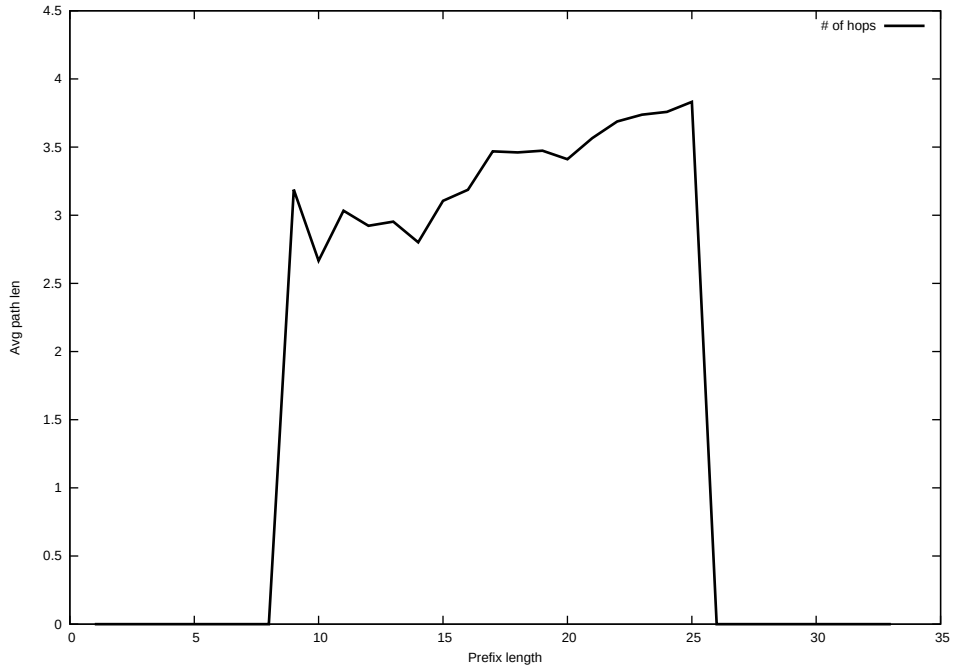
2014-03-25



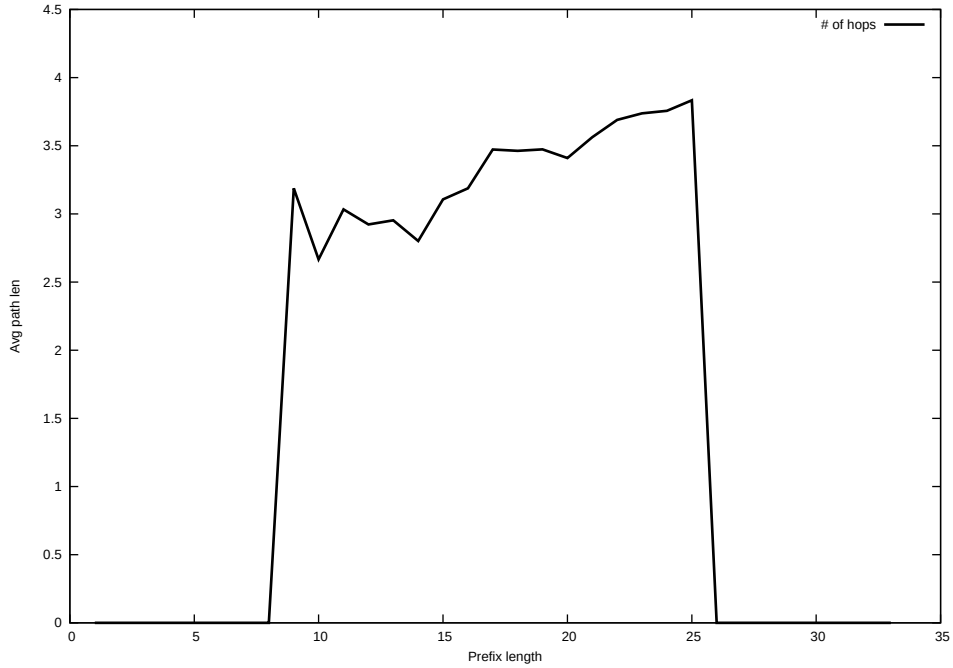
2014-03-26



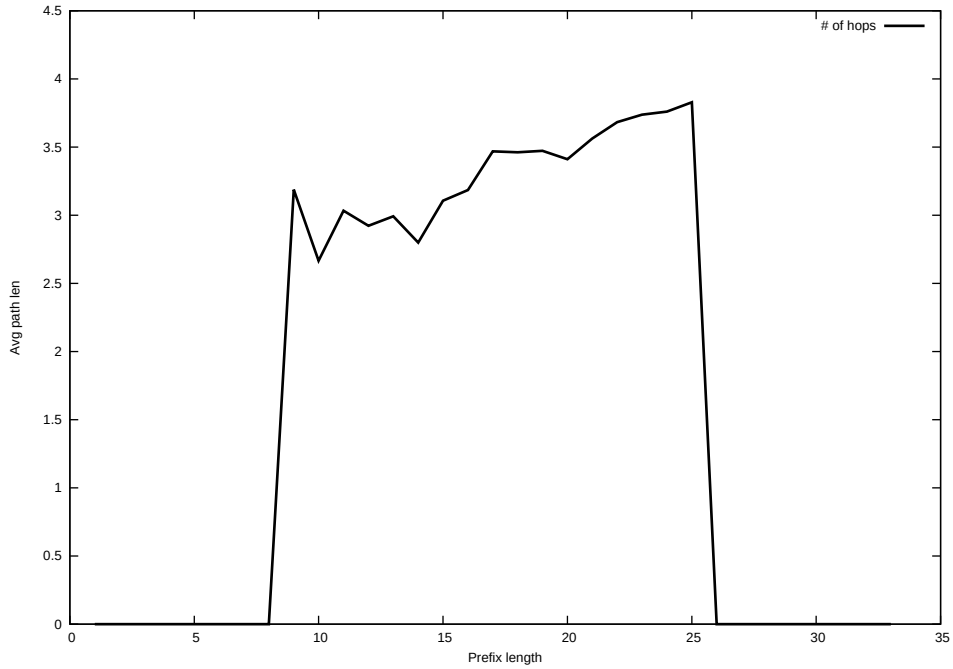
2014-03-27



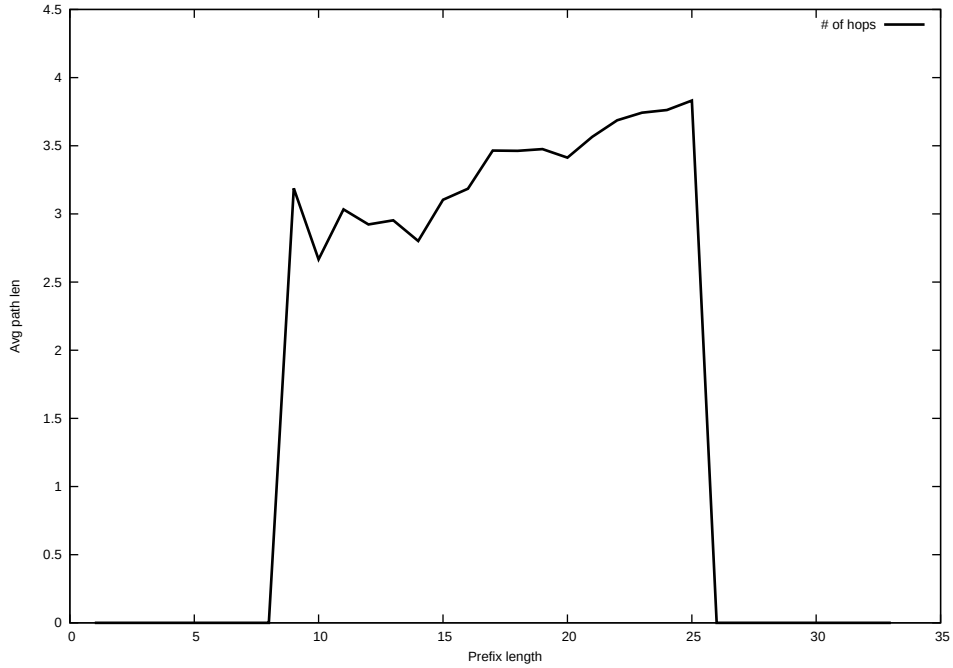
2014-03-28



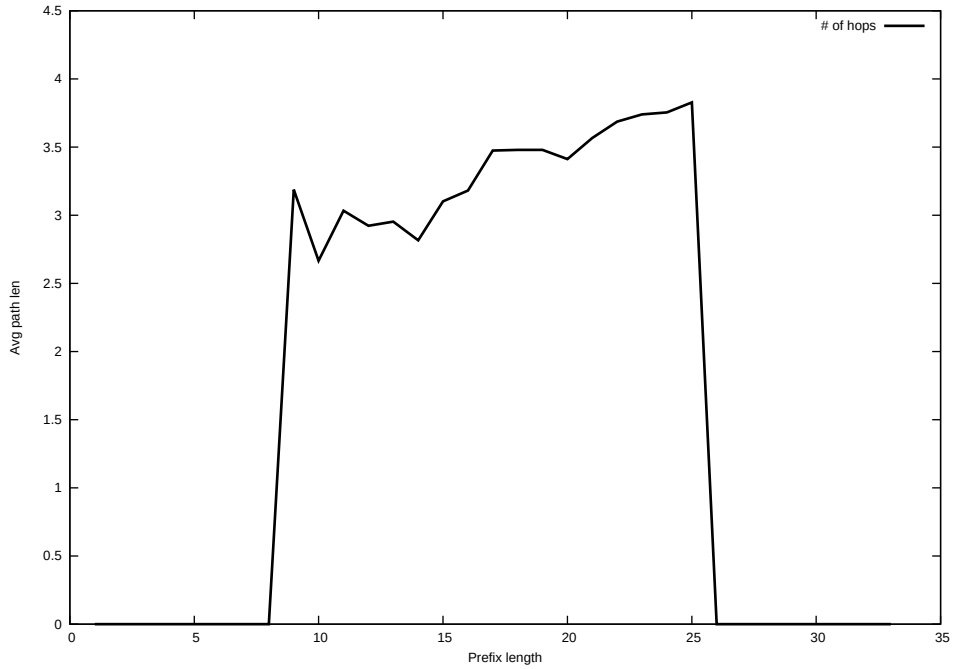
2014-03-29



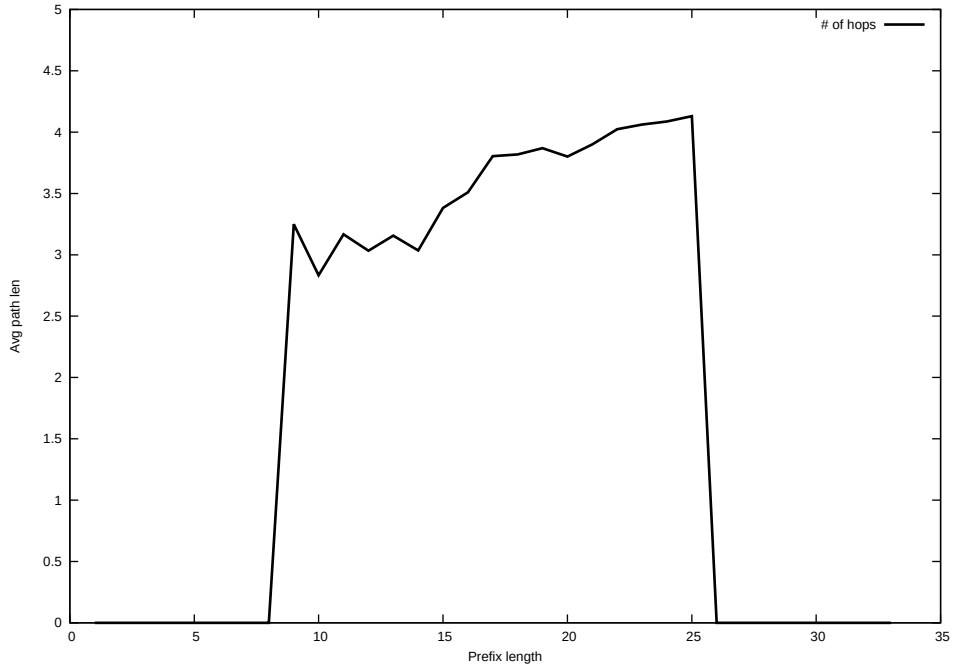
2014-03-30



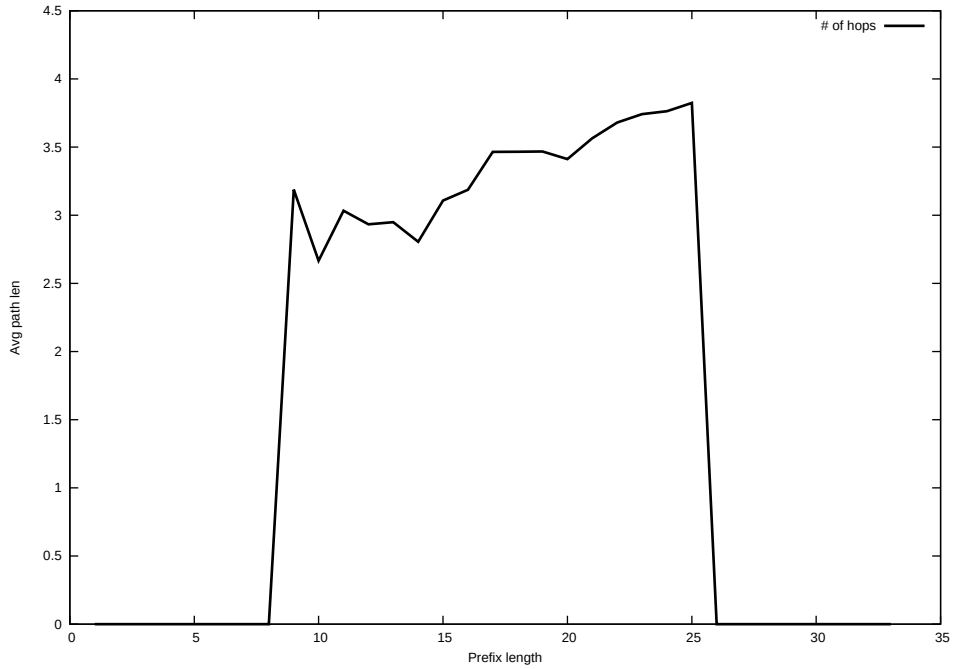
2014-03-31



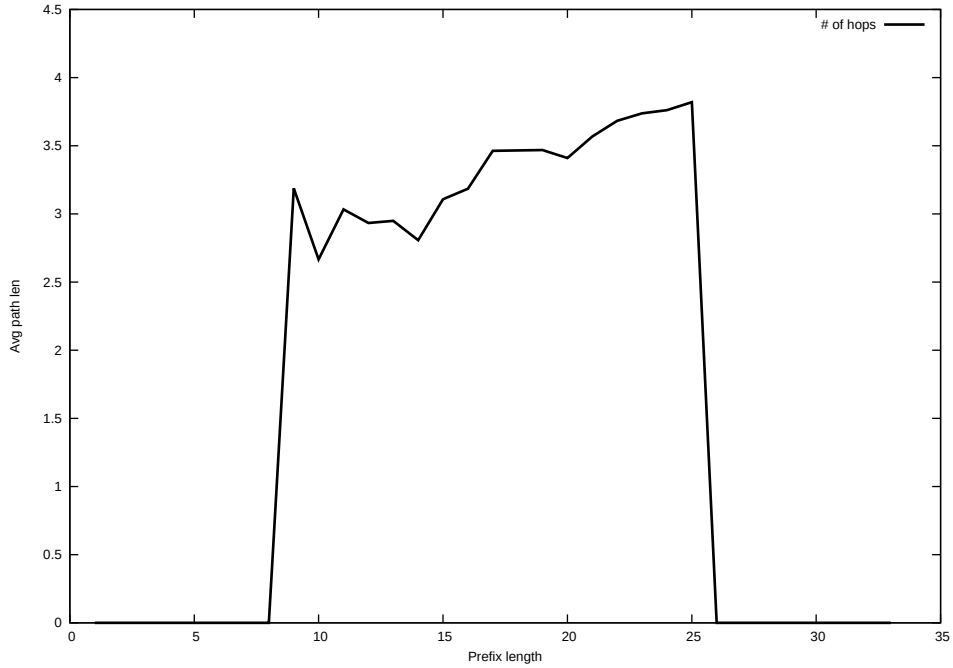
2014-04-01



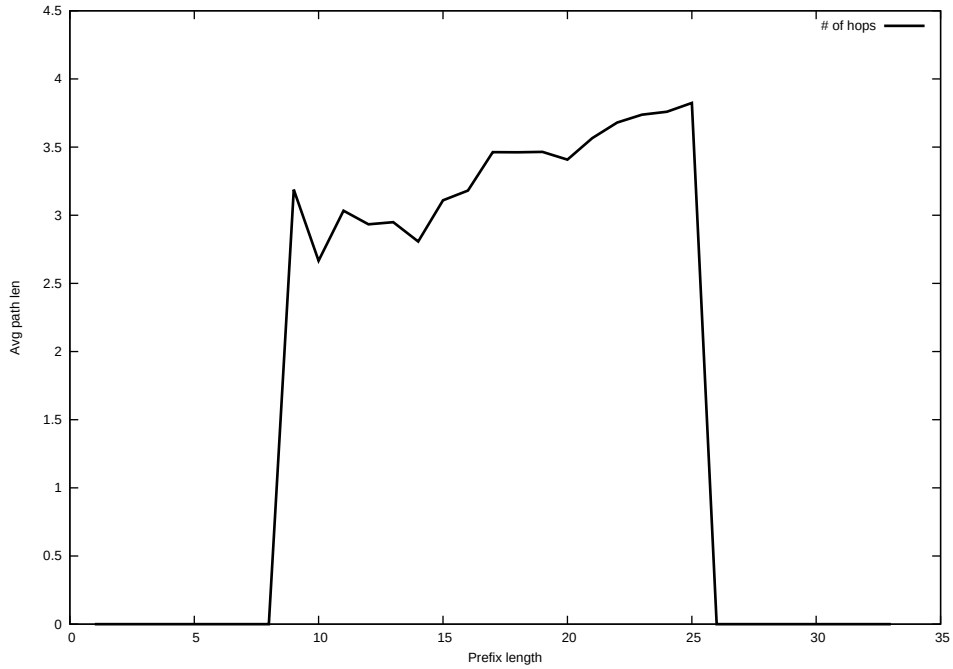
2014-04-02



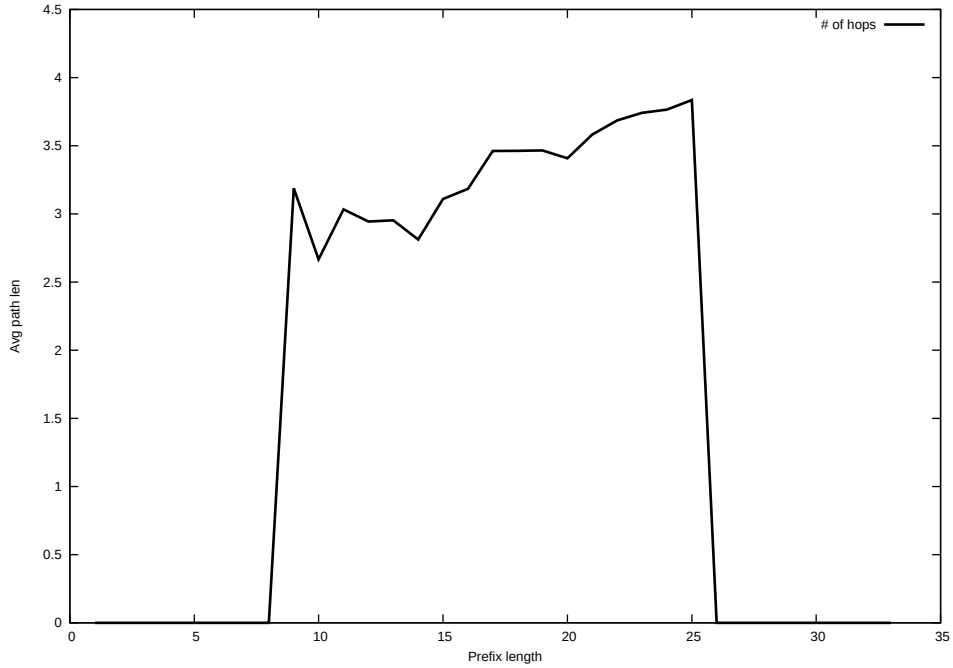
2014-04-03



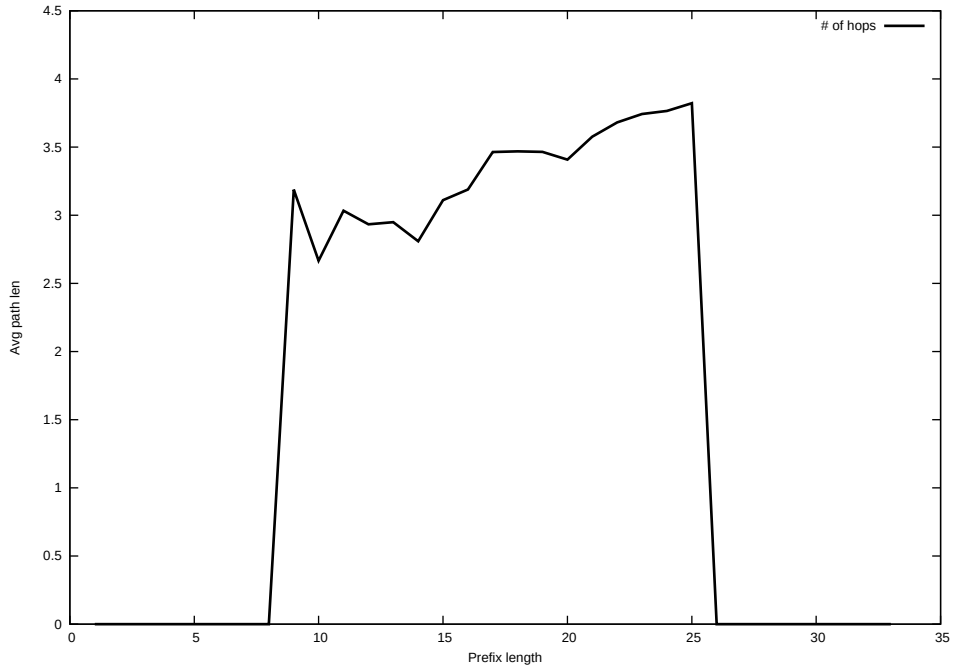
2014-04-04



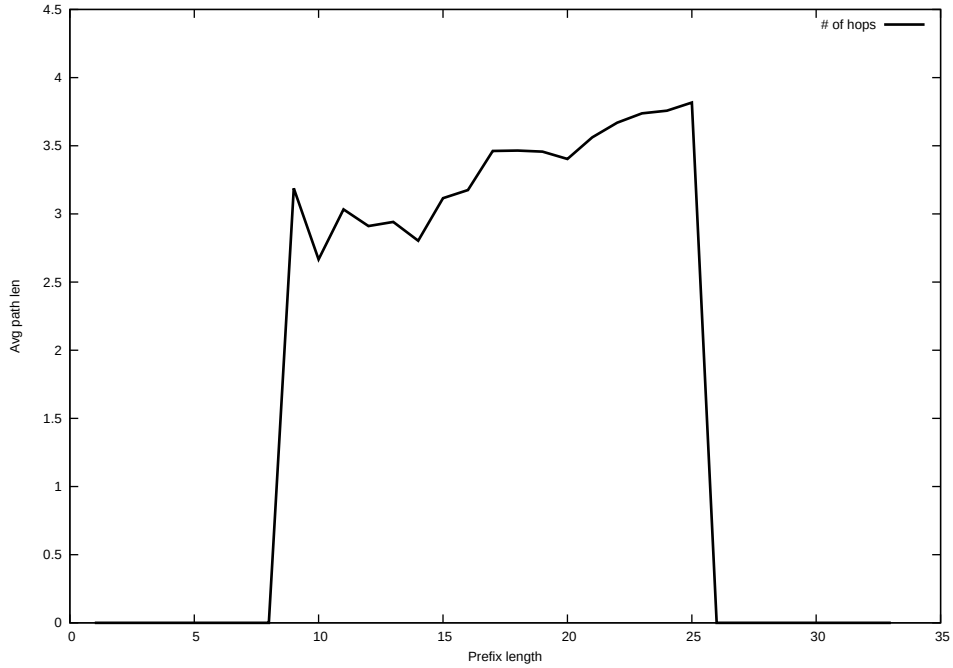
2014-04-05



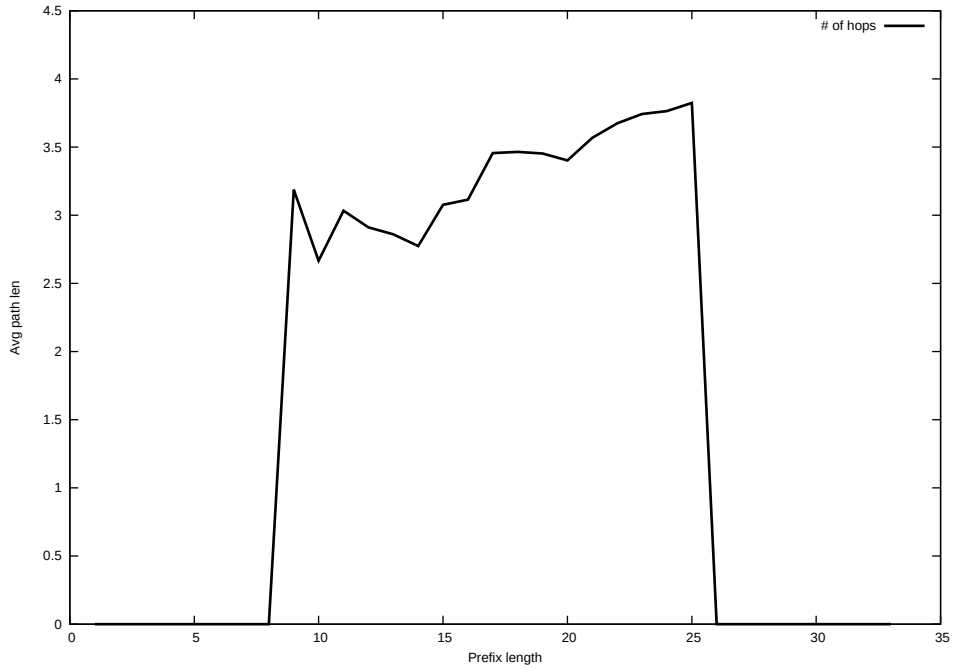
2014-04-06



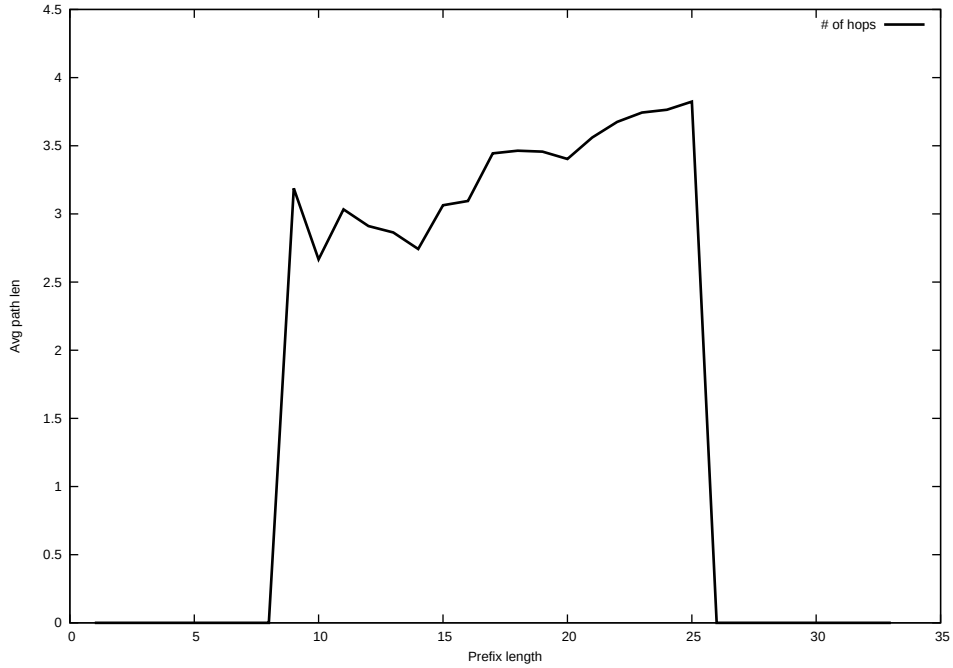
2014-04-07



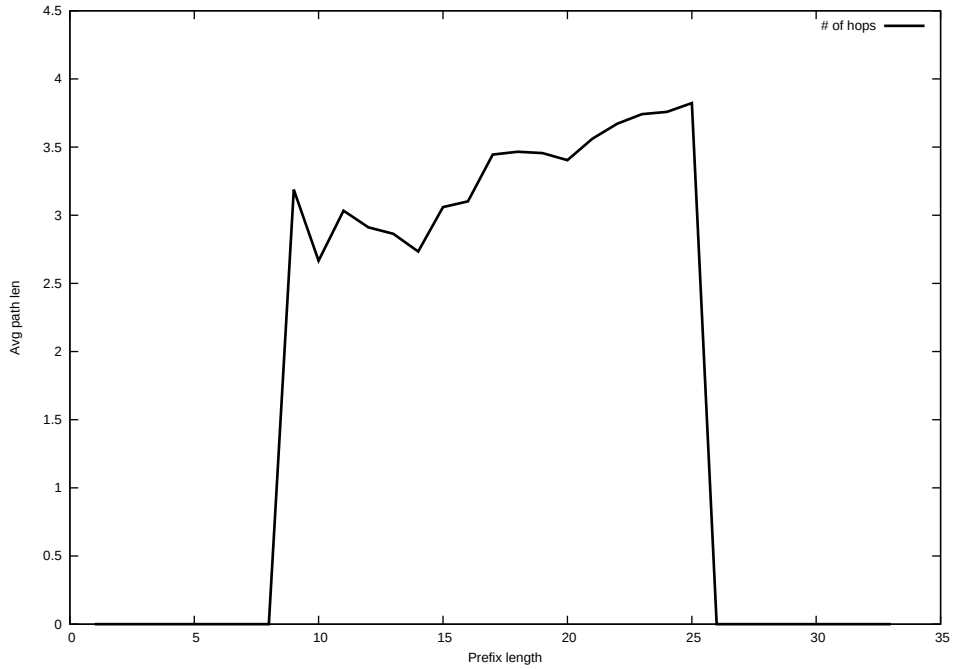
2014-04-08



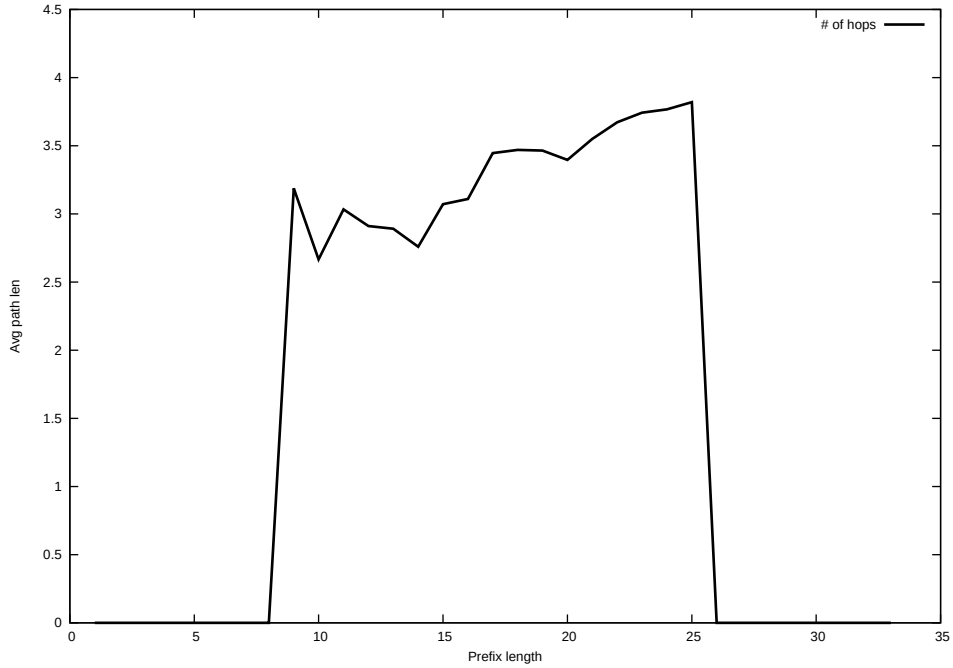
2014-04-09



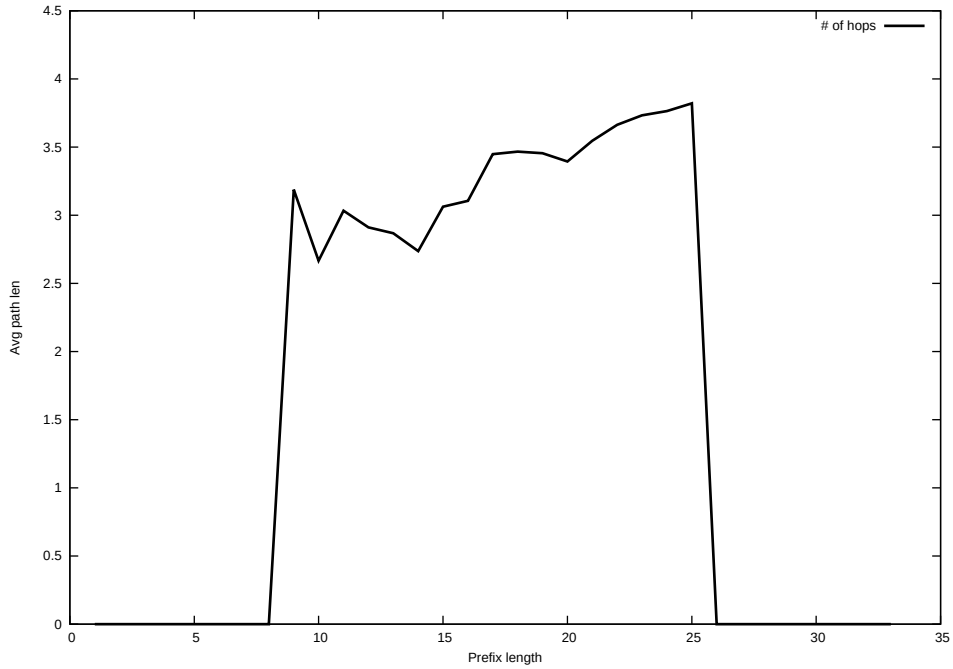
2014-04-10



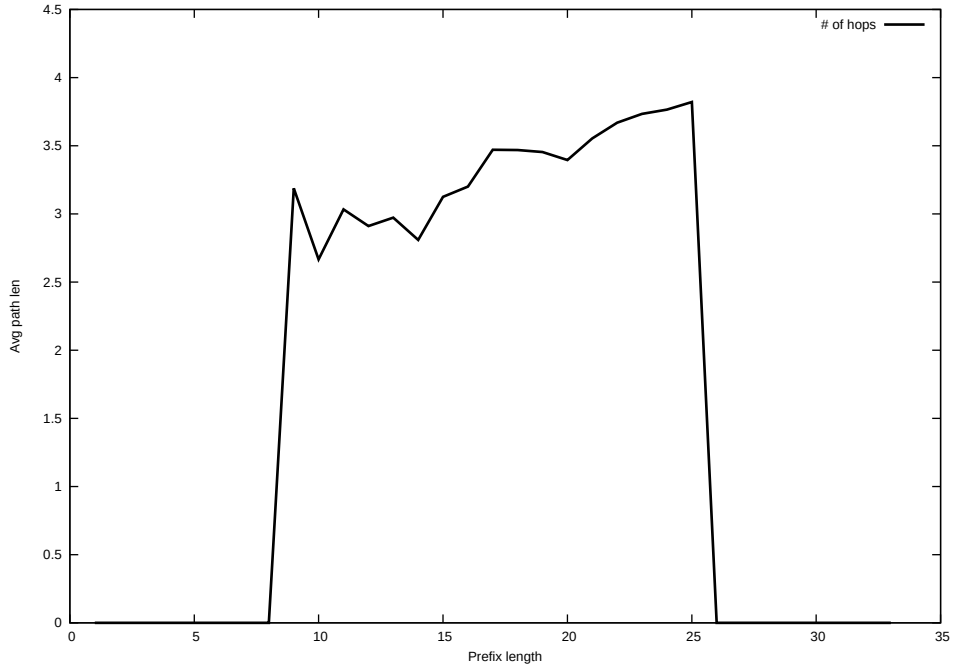
2014-04-11



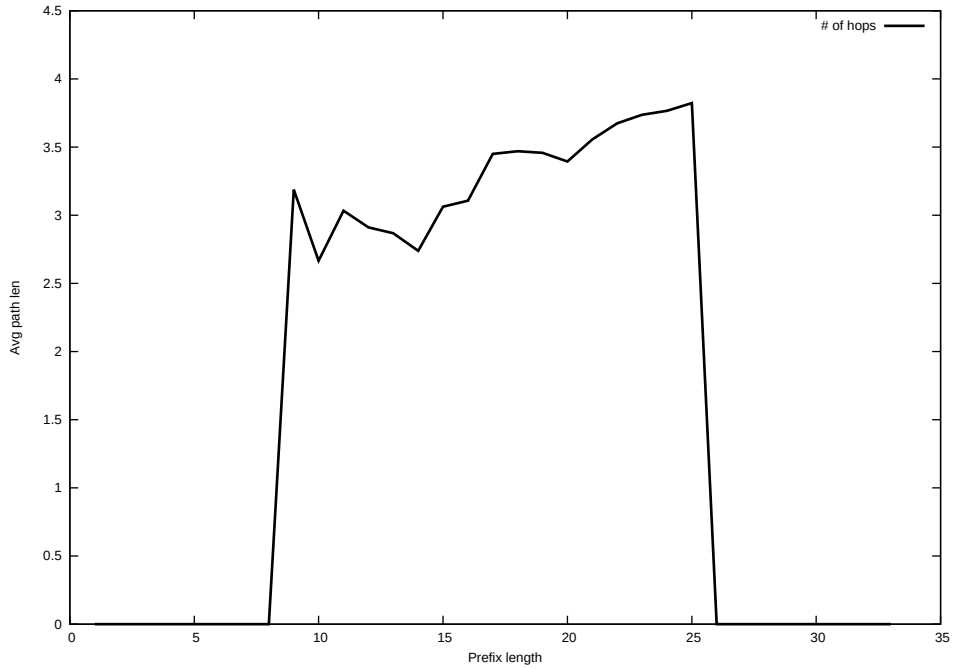
2014-04-12



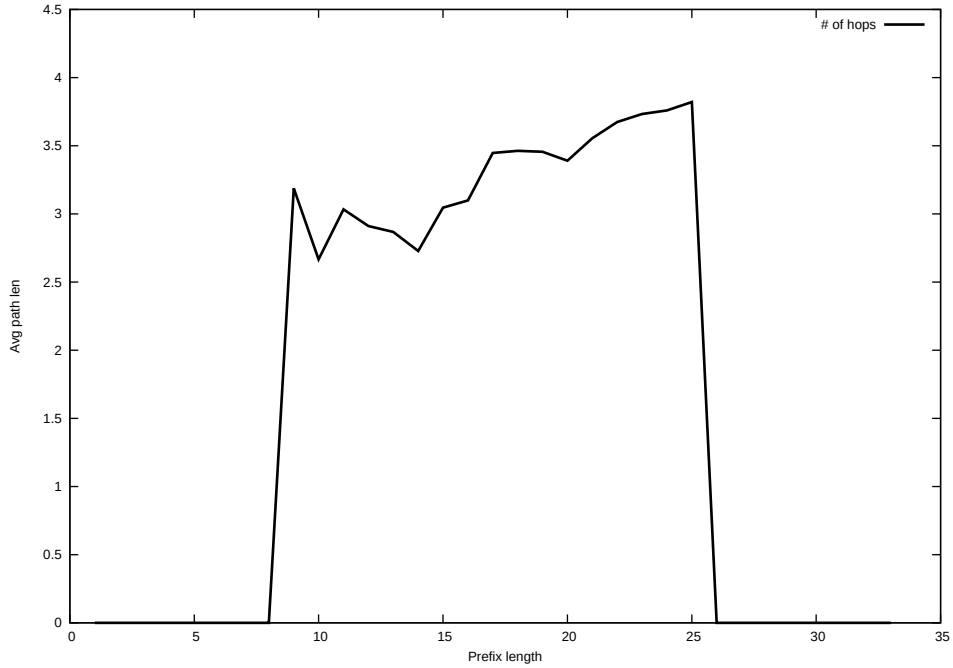
2014-04-13



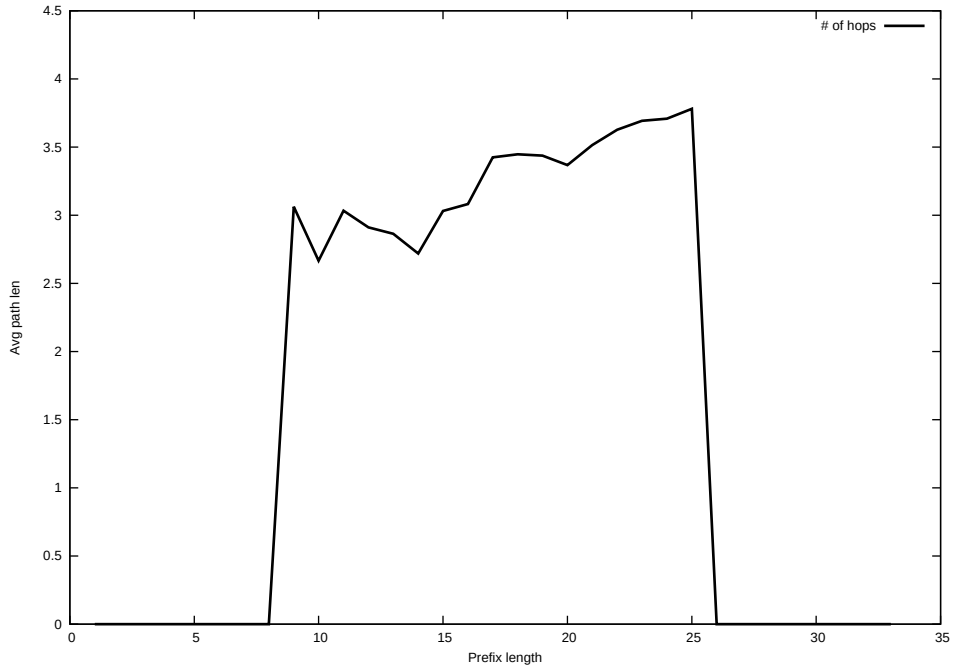
2014-04-14



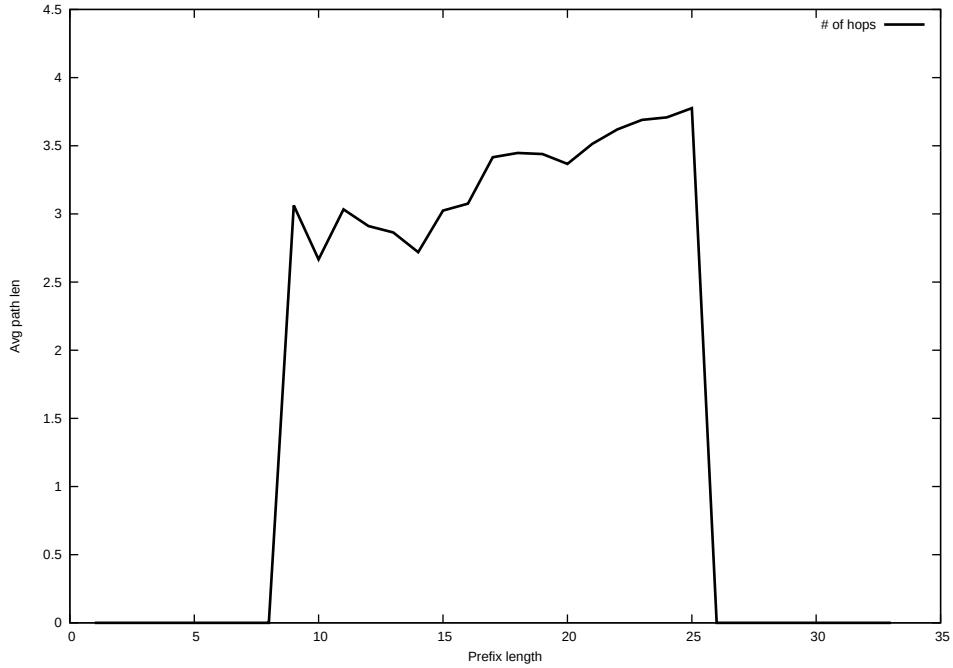
2014-04-15



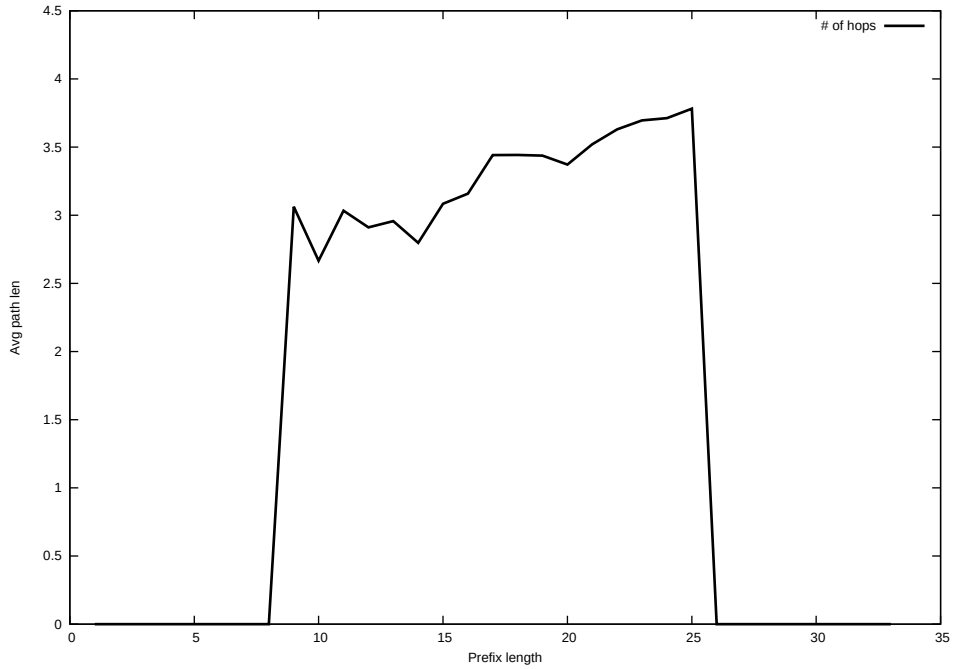
2014-04-16



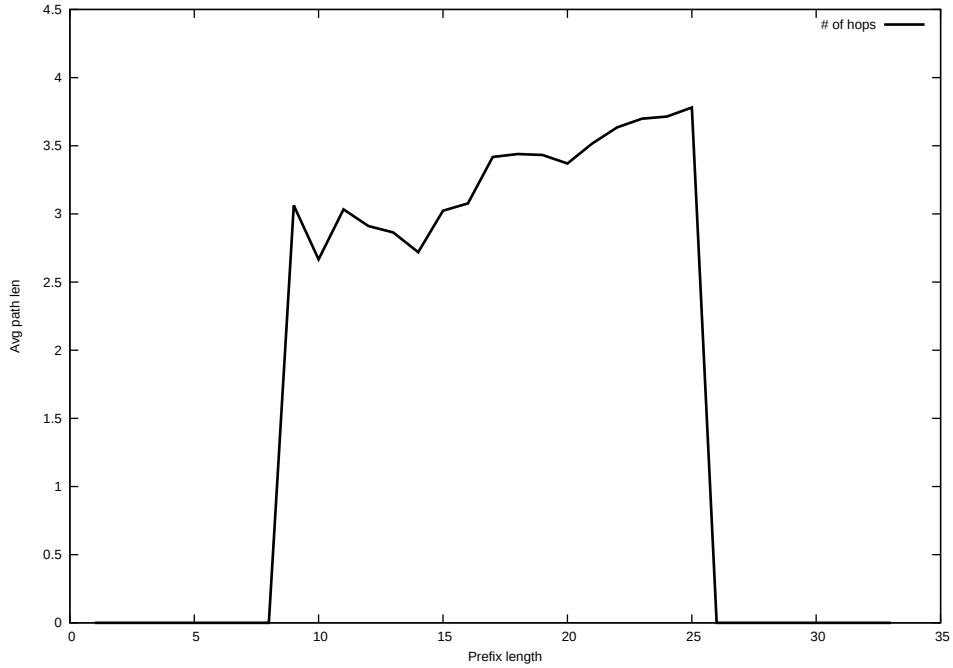
2014-04-17



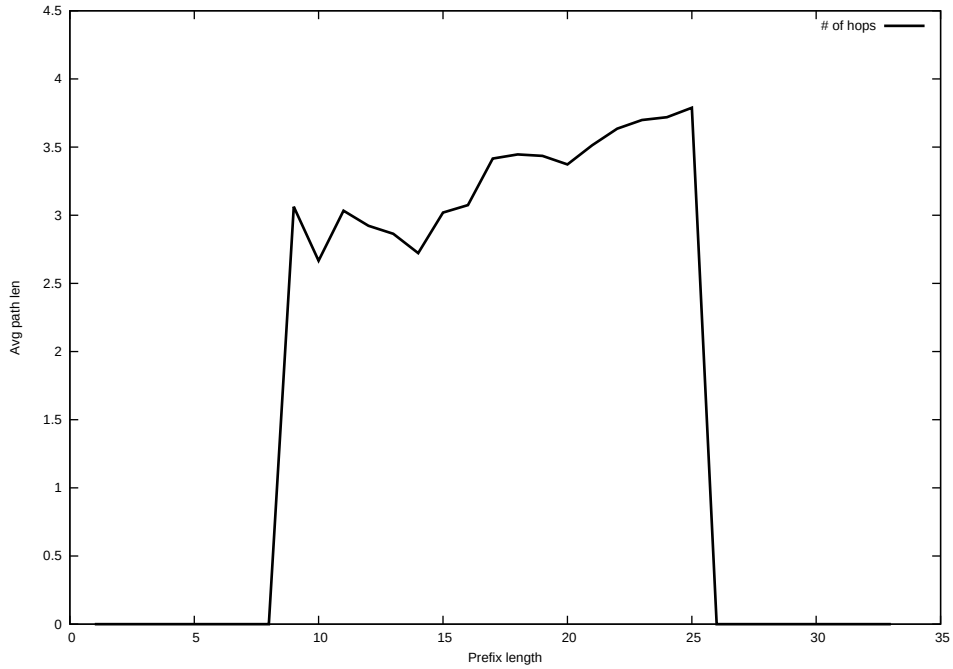
2014-04-18



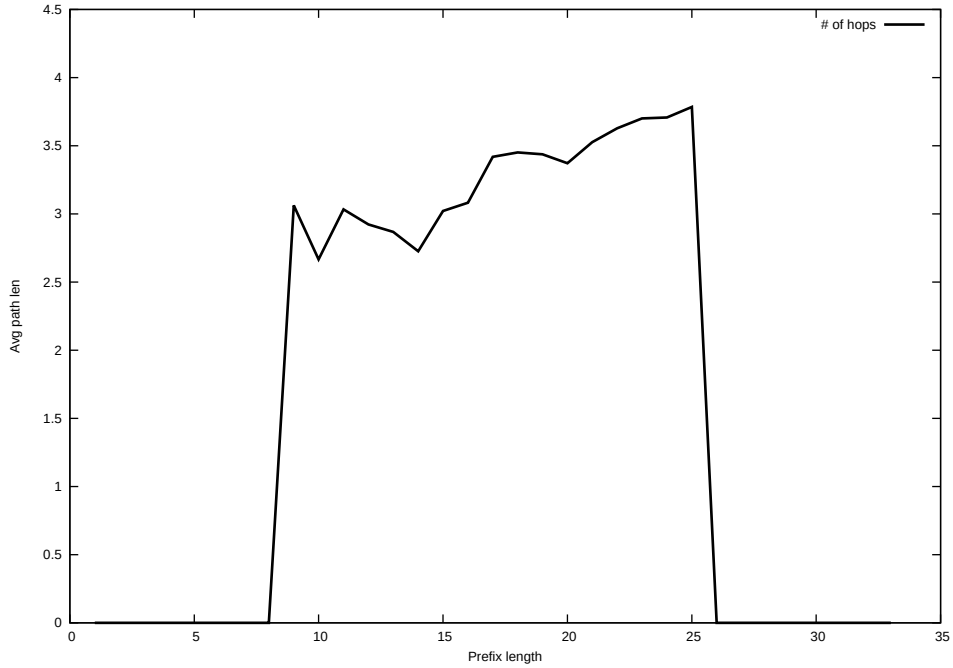
2014-04-19



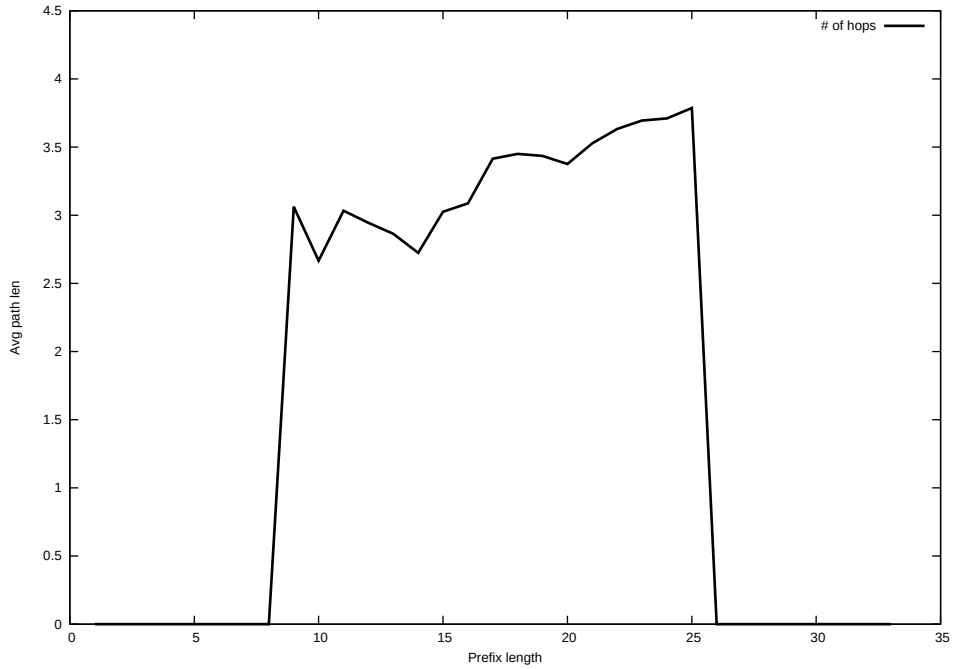
2014-04-20



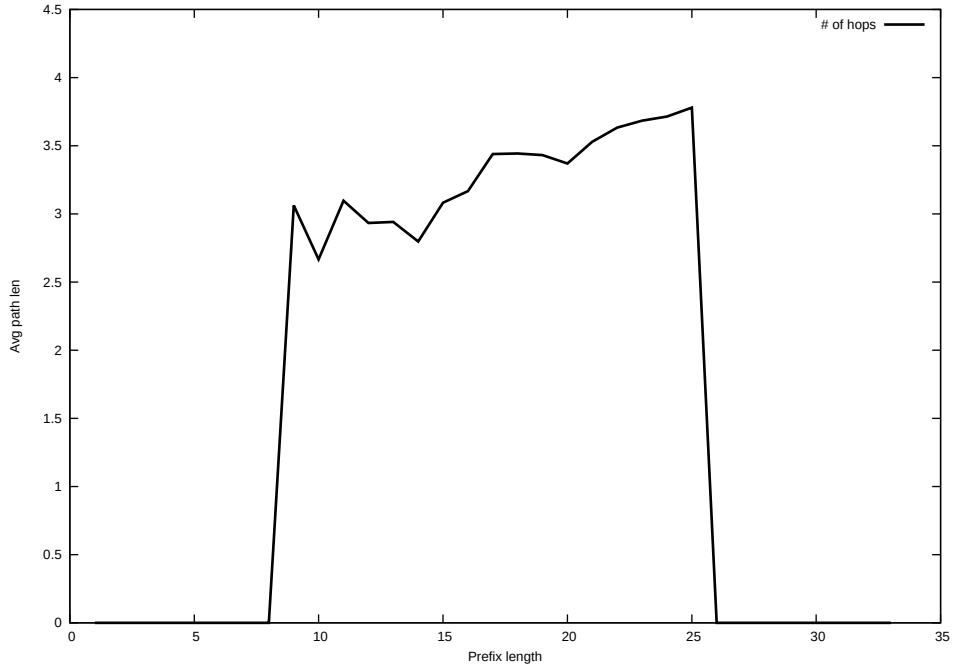
2014-04-21



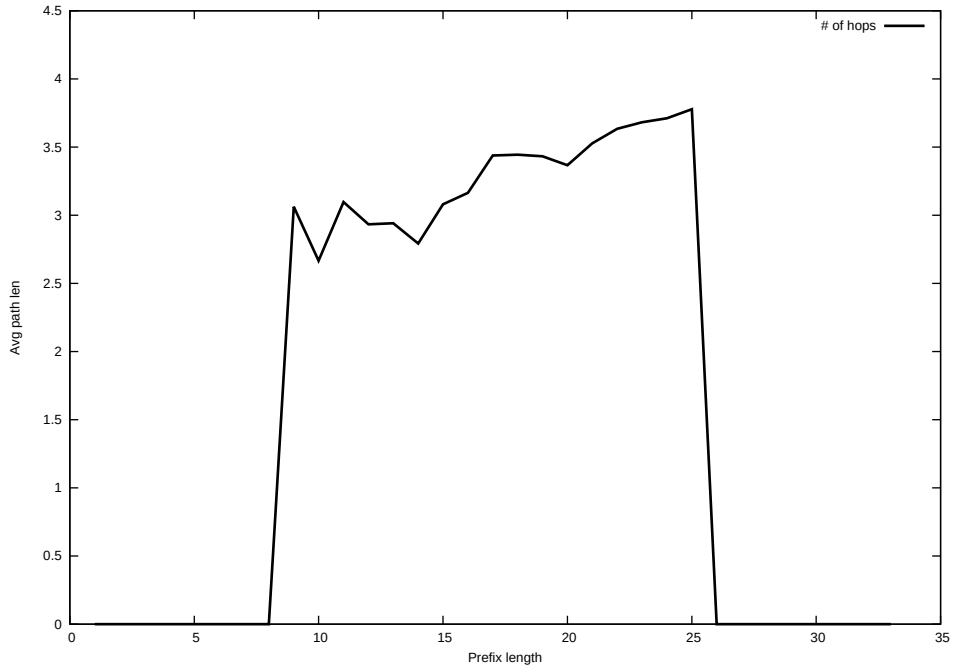
2014-04-22



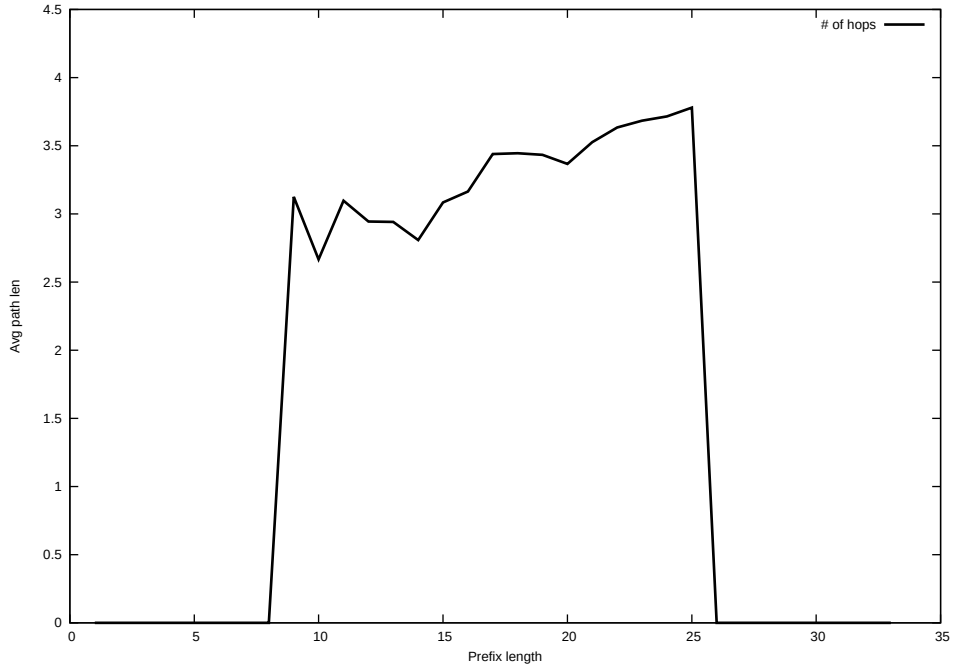
2014-04-23



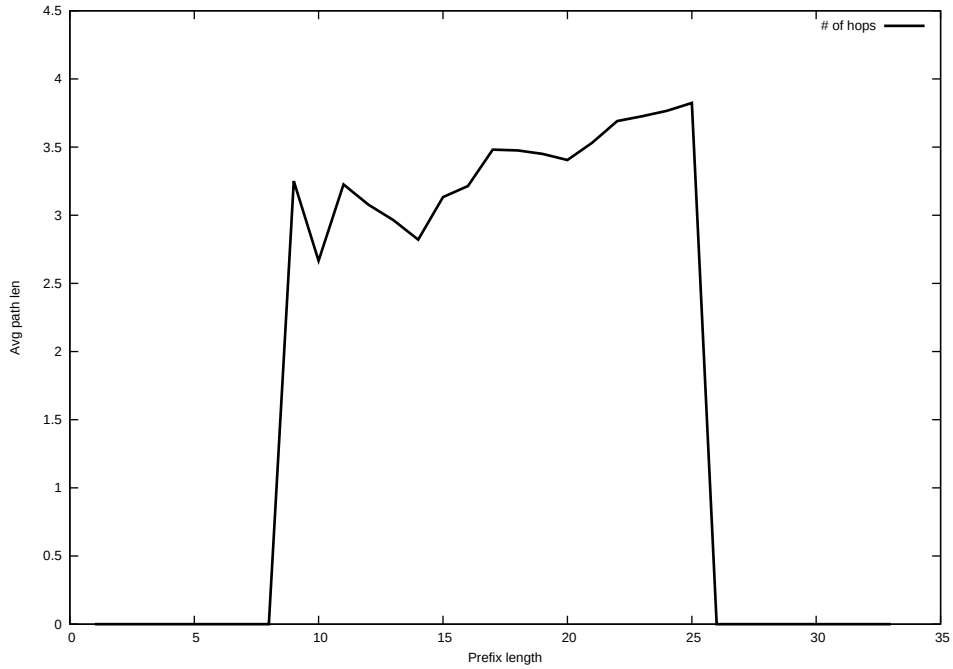
2014-04-24



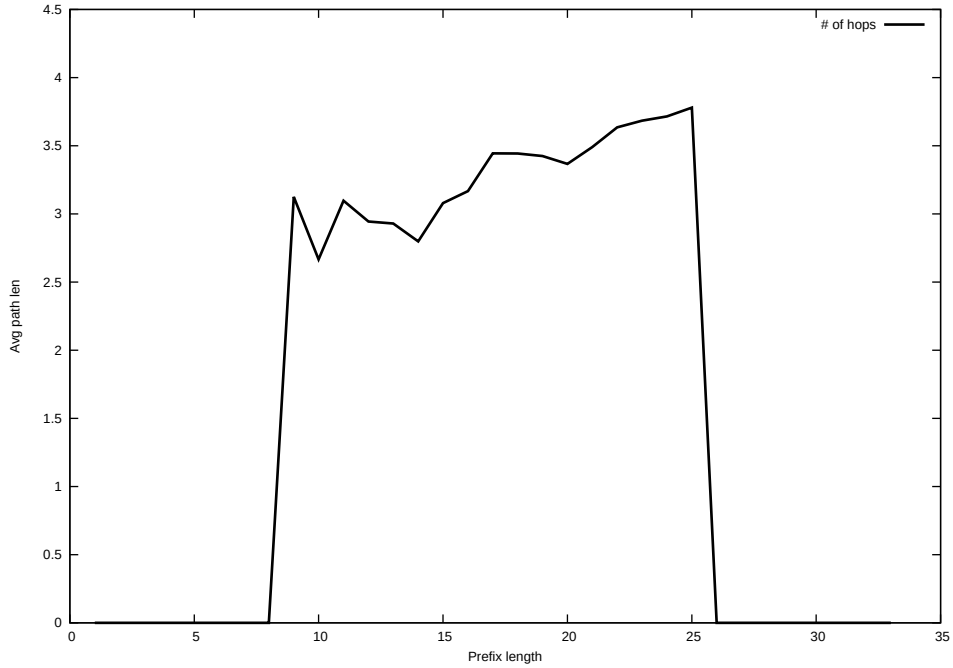
2014-04-25



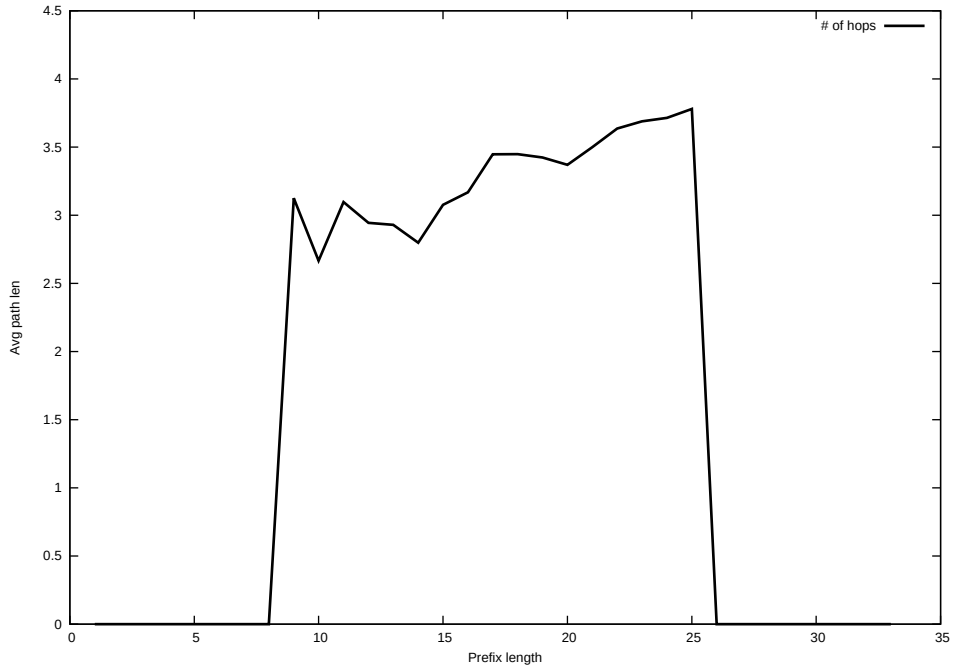
2014-04-26



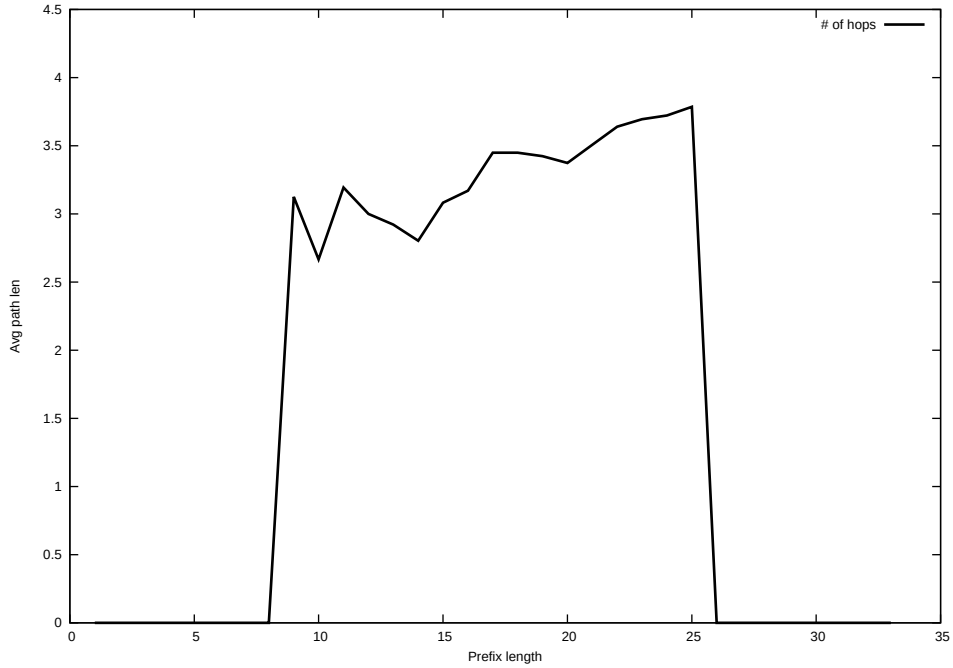
2014-04-27



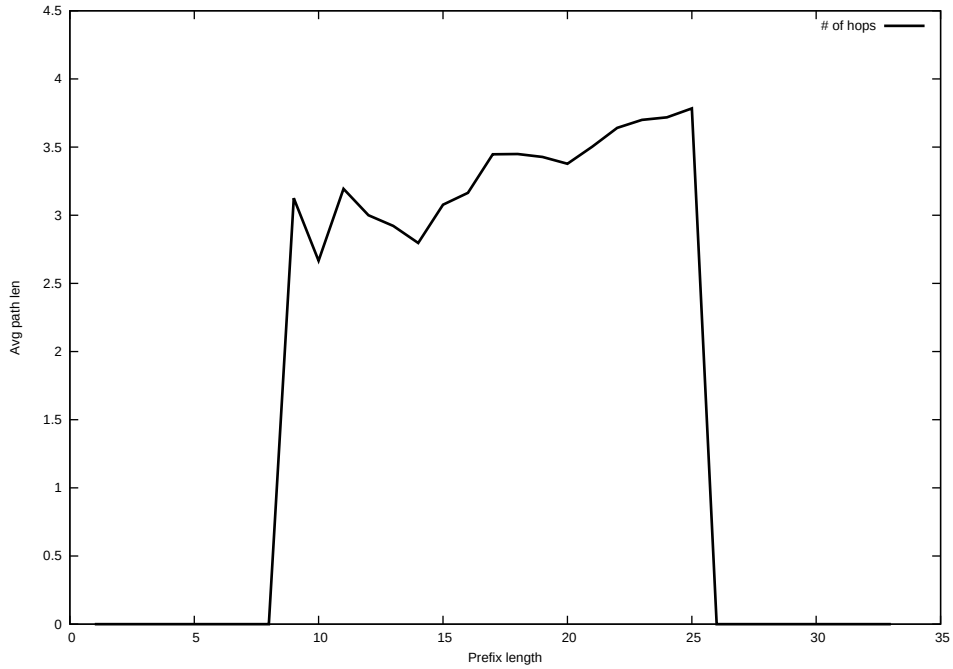
2014-04-28



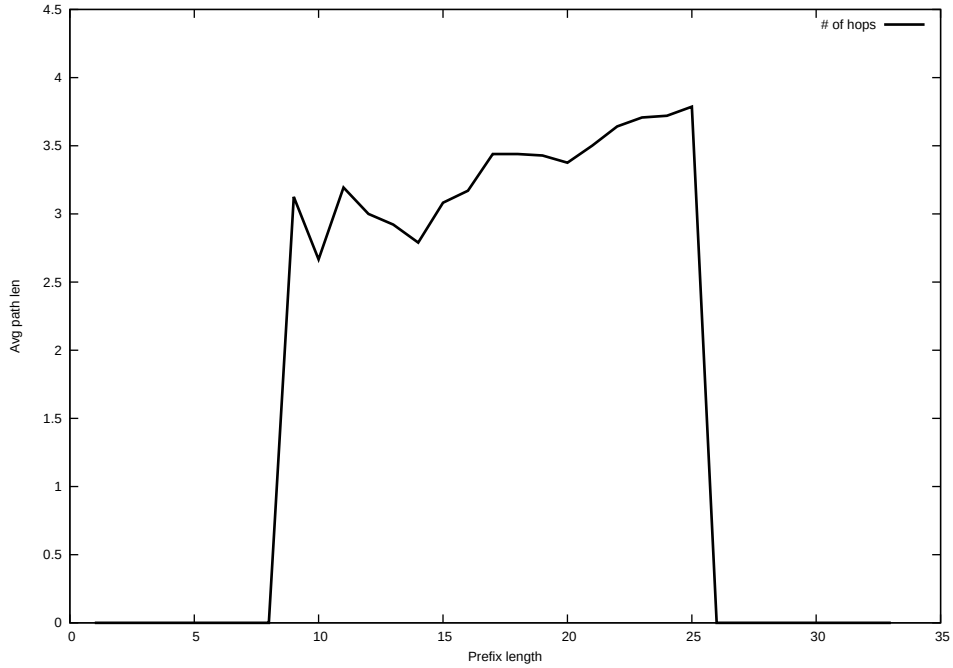
2014-04-29



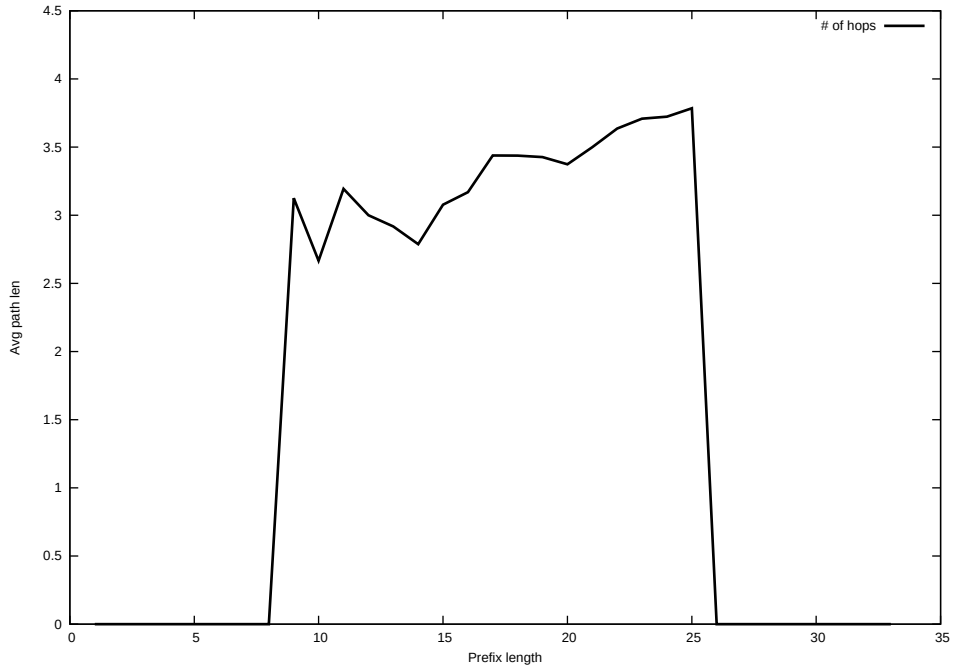
2014-04-30



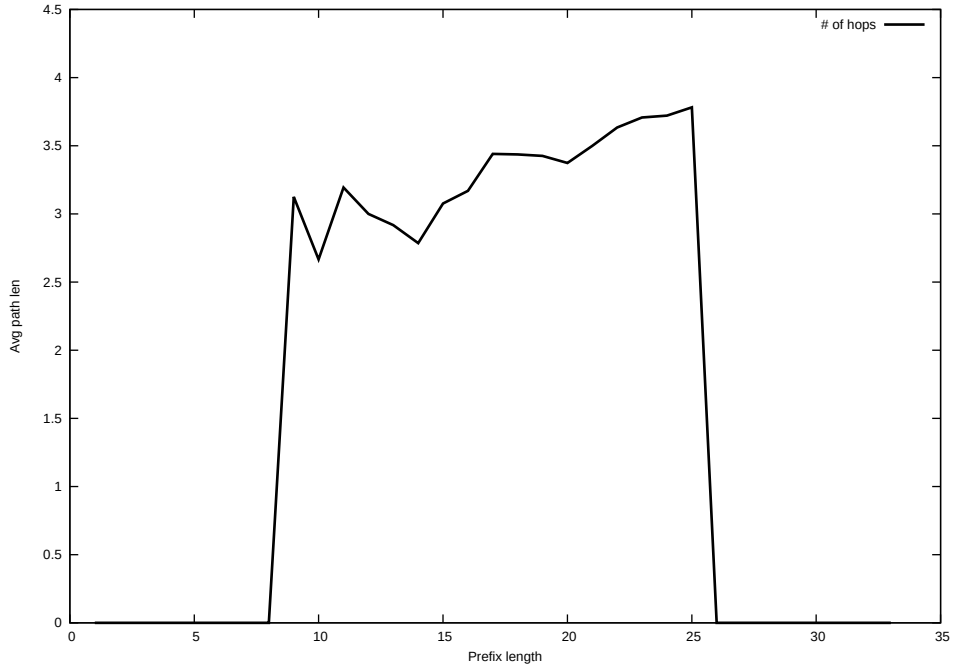
2014-05-01



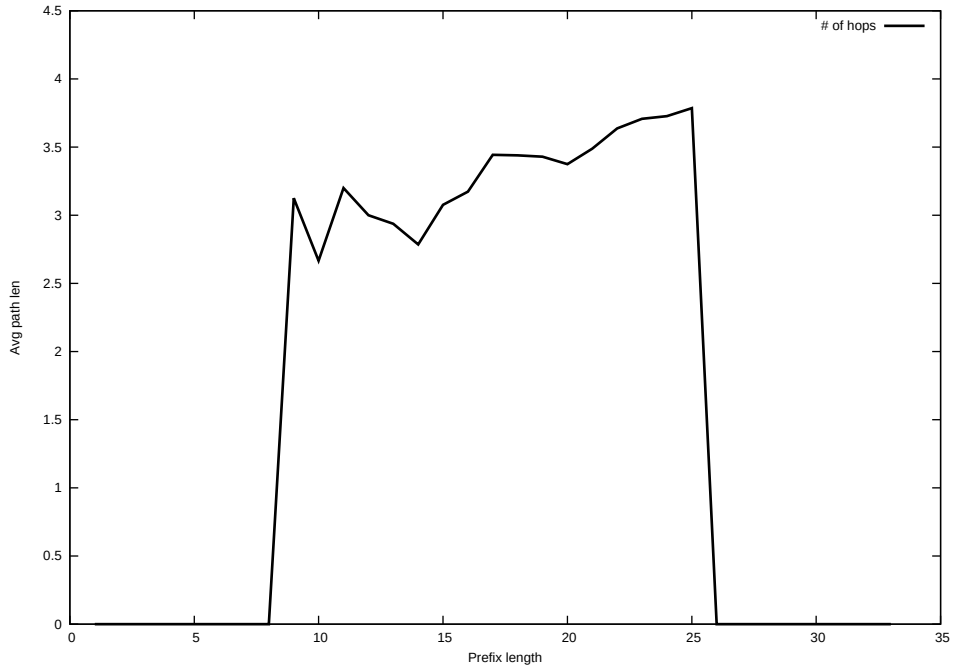
2014-05-02



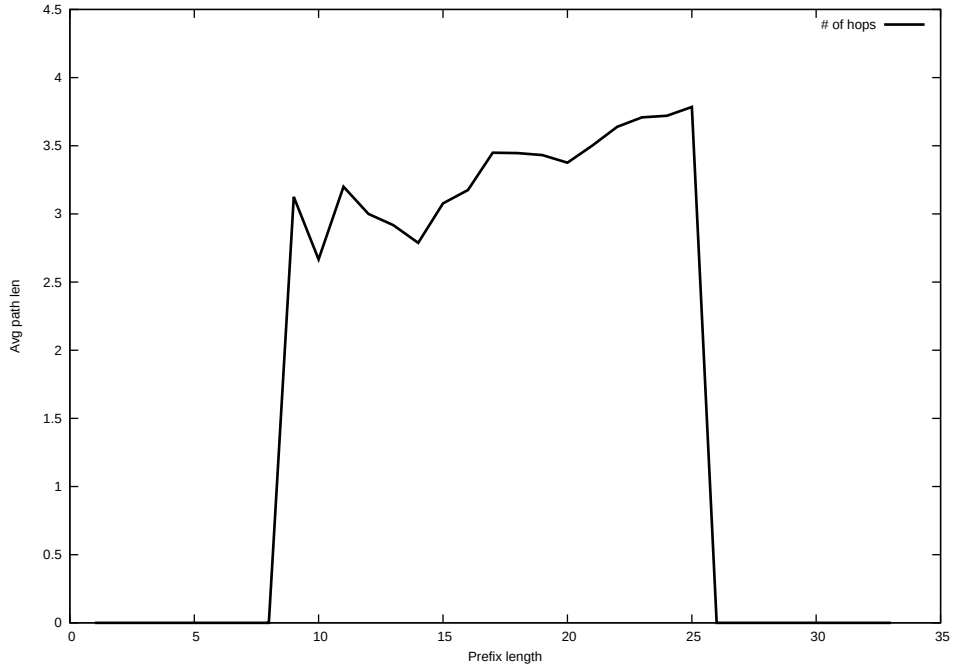
2014-05-03



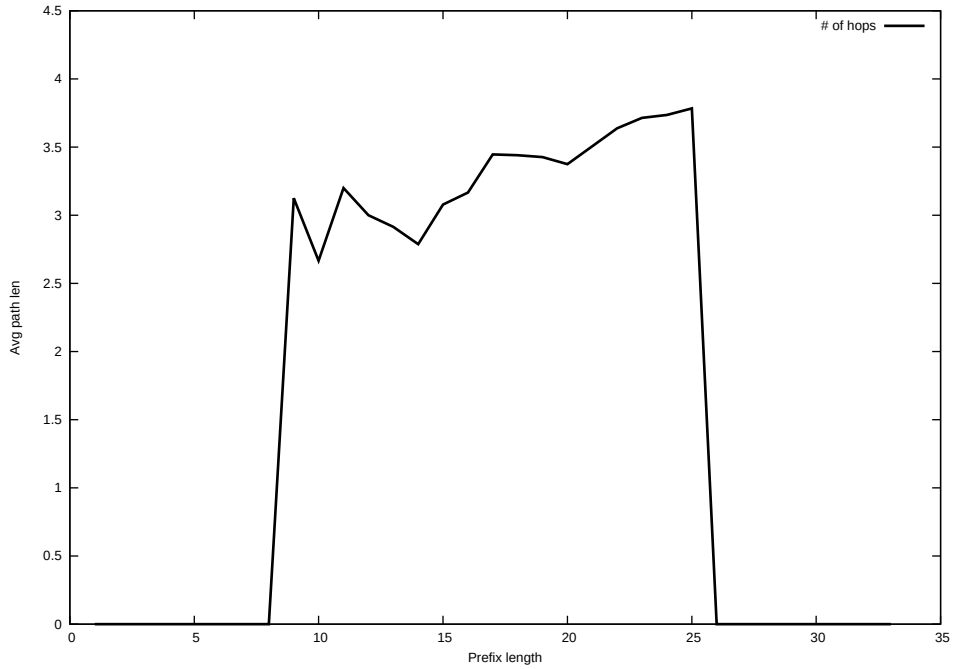
2014-05-04



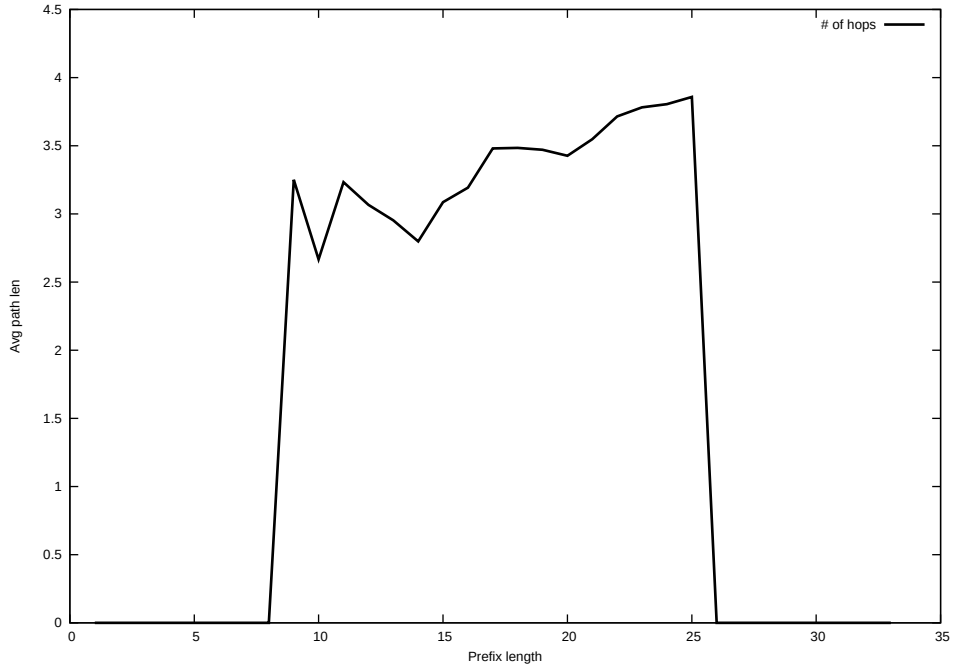
2014-05-05



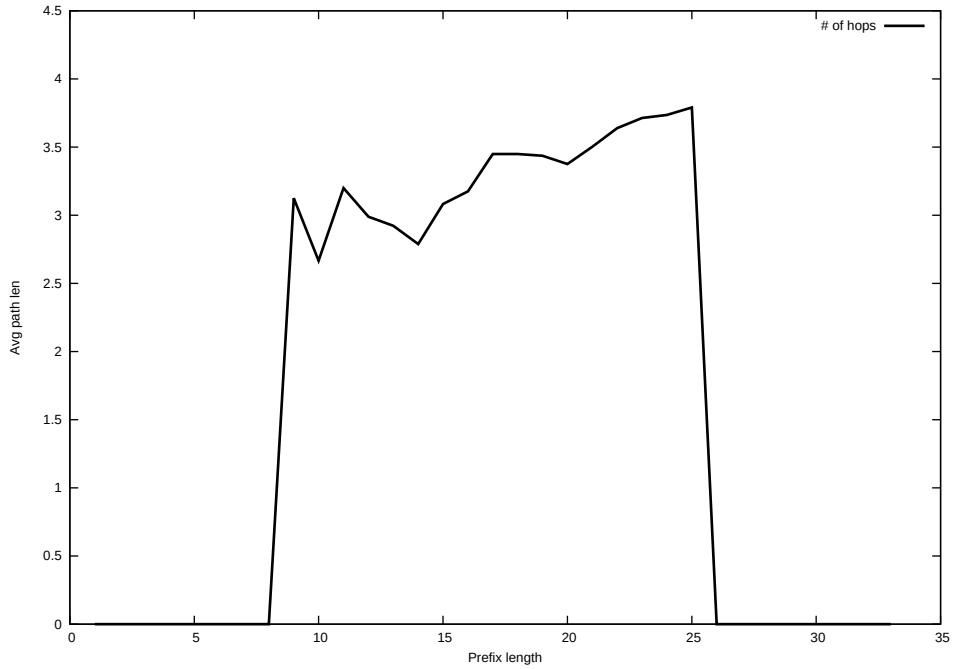
2014-05-06



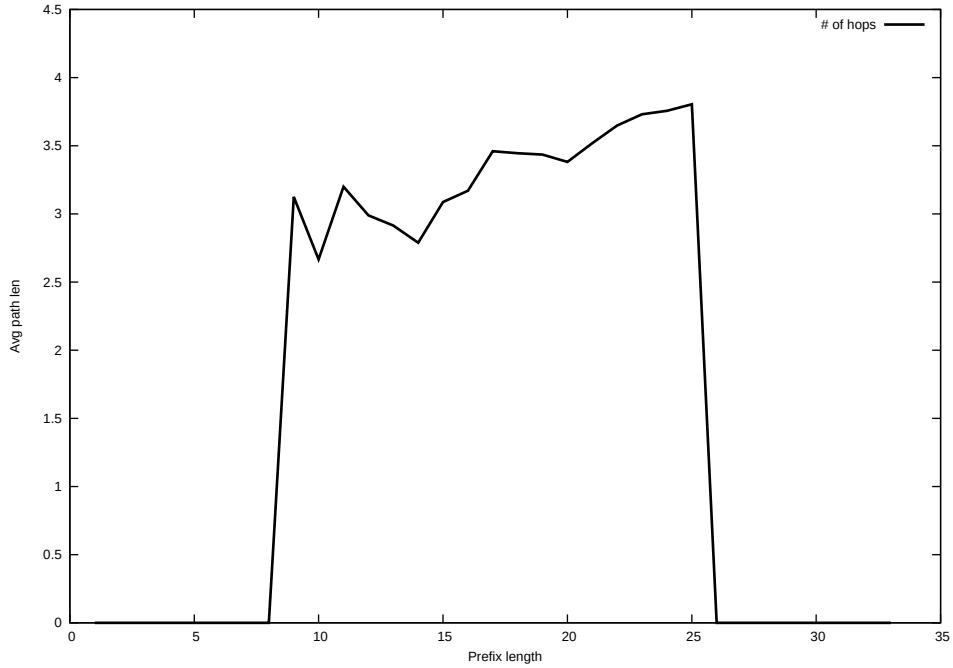
2014-05-07



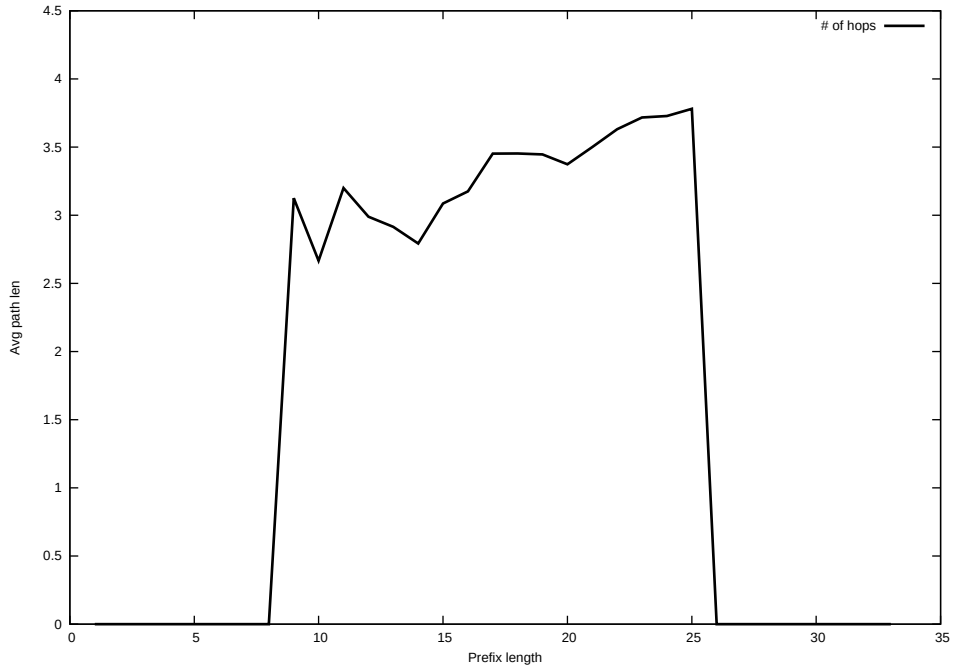
2014-05-08



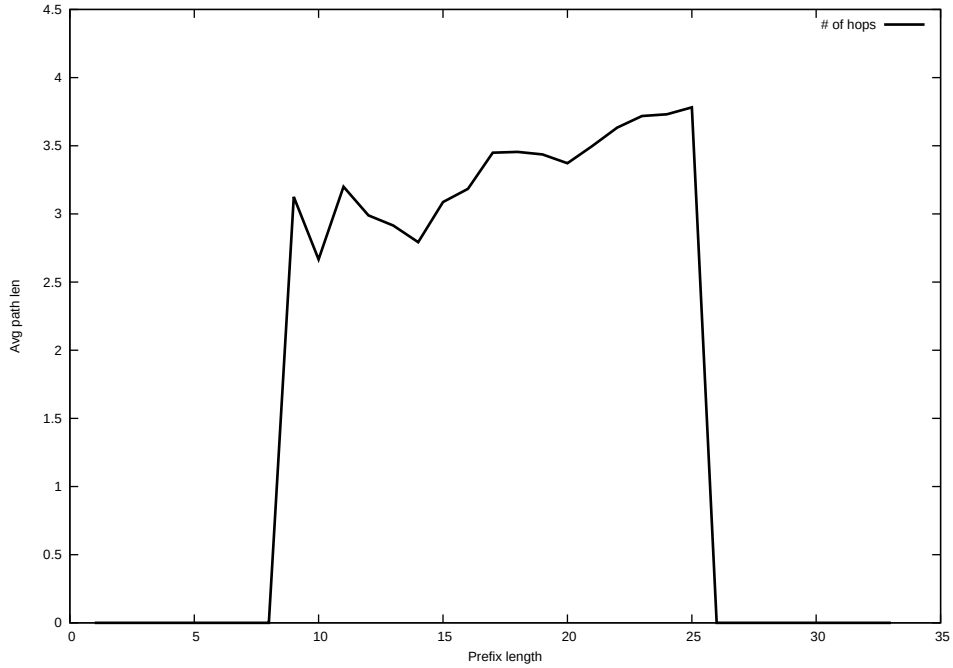
2014-05-09



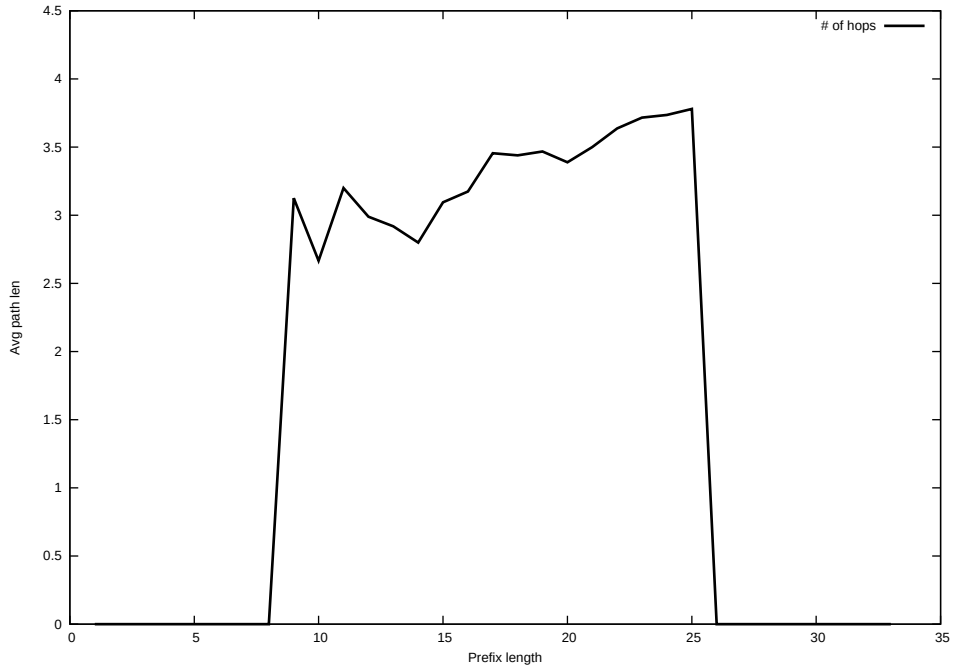
2014-05-10



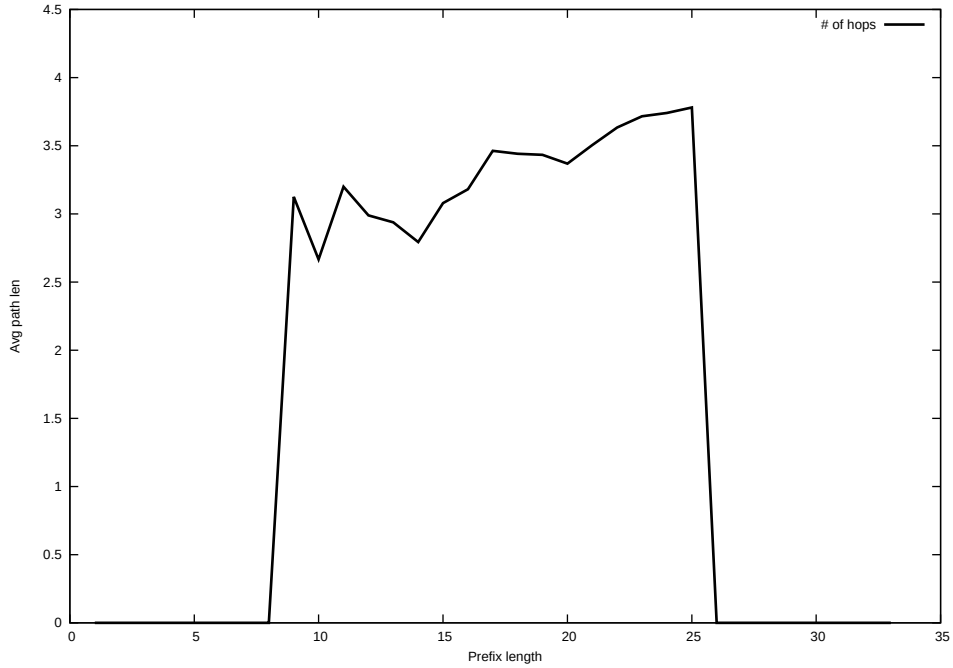
2014-05-11



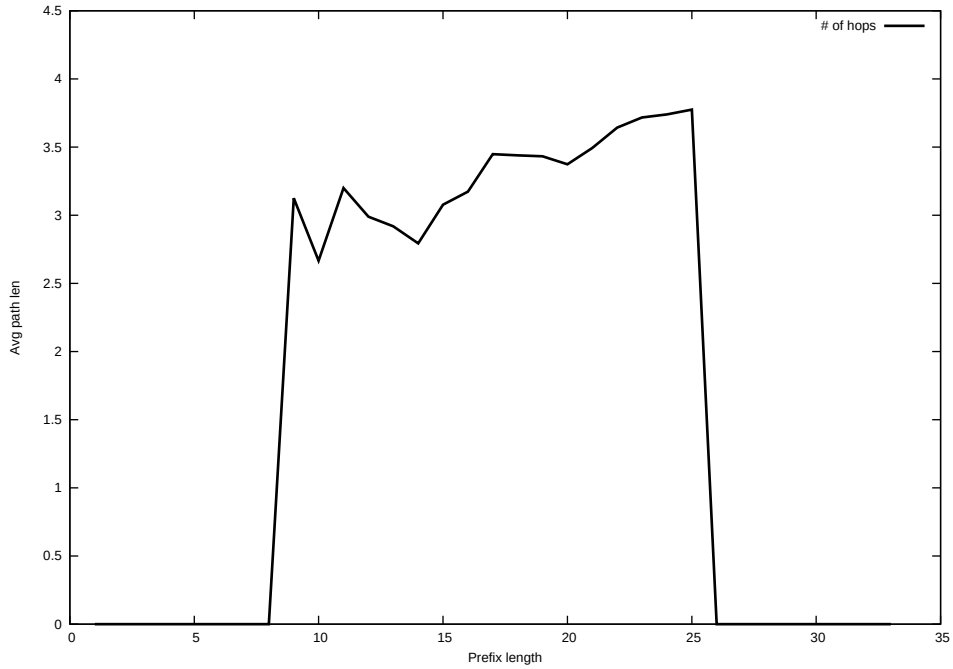
2014-05-12



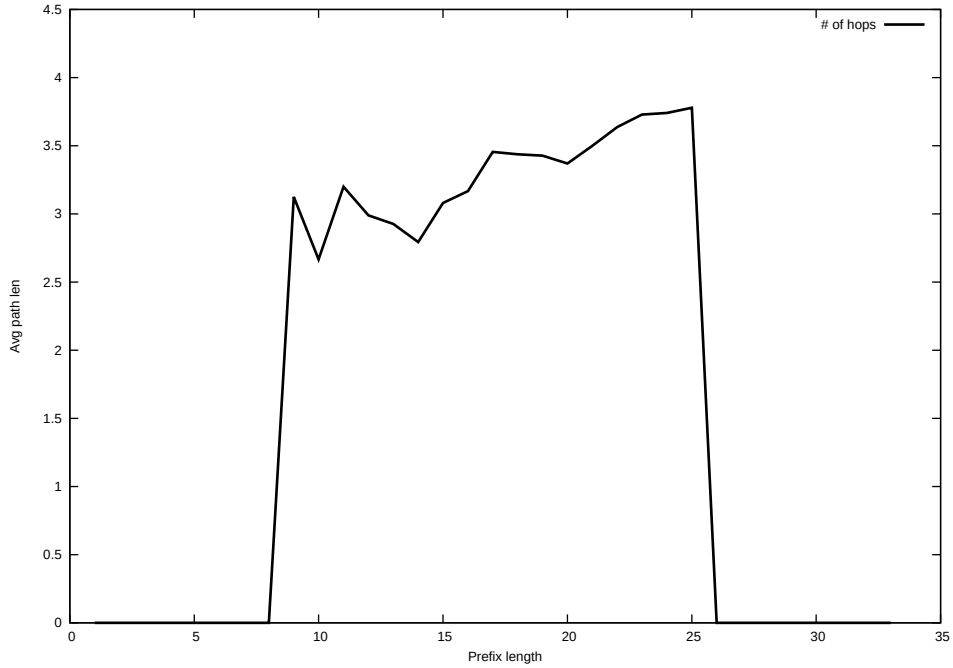
2014-05-13



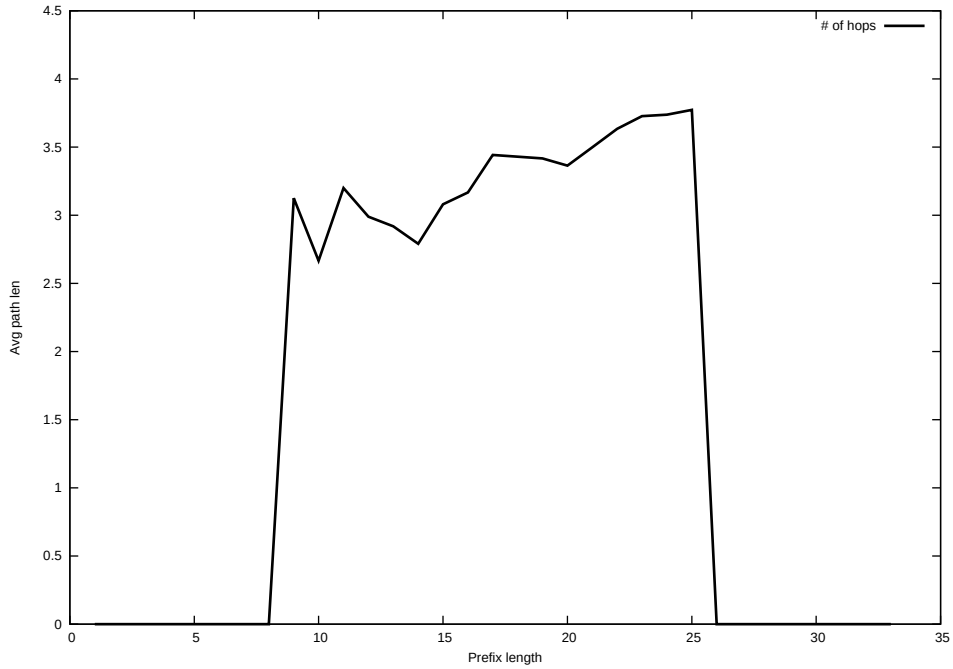
2014-05-14



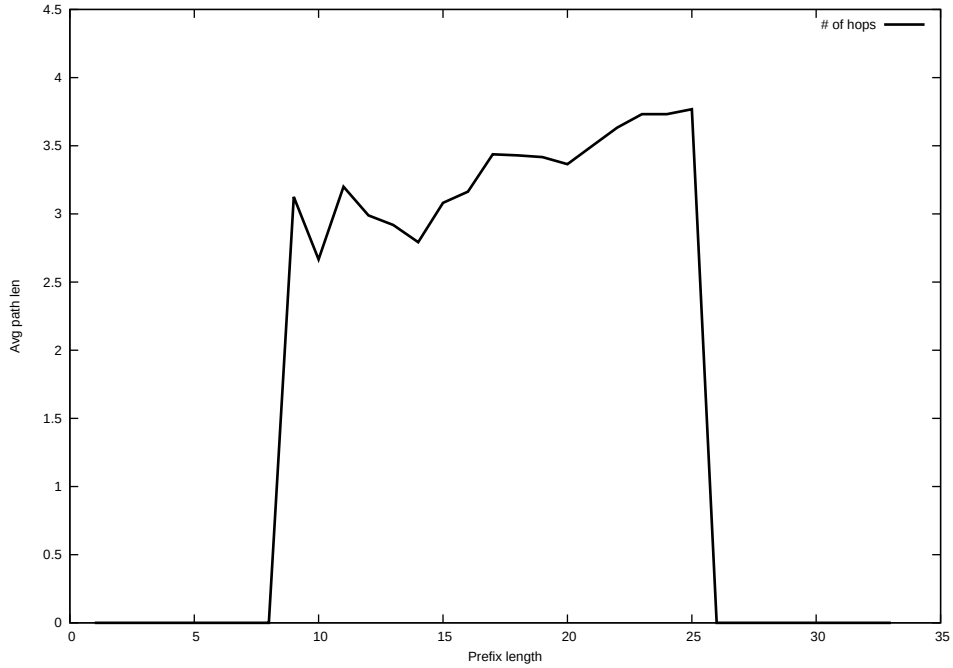
2014-05-15



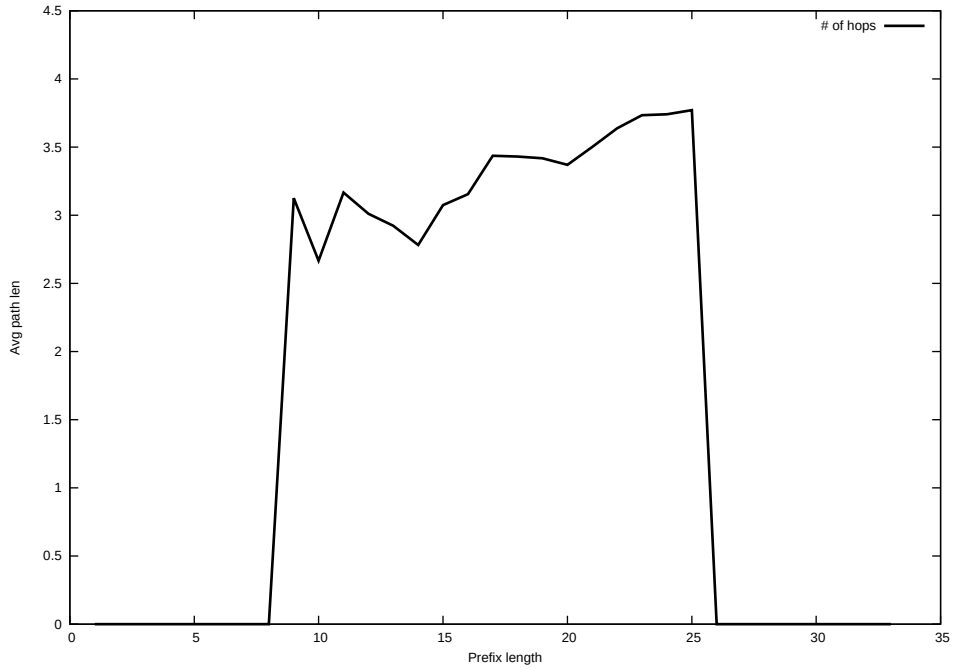
2014-05-16



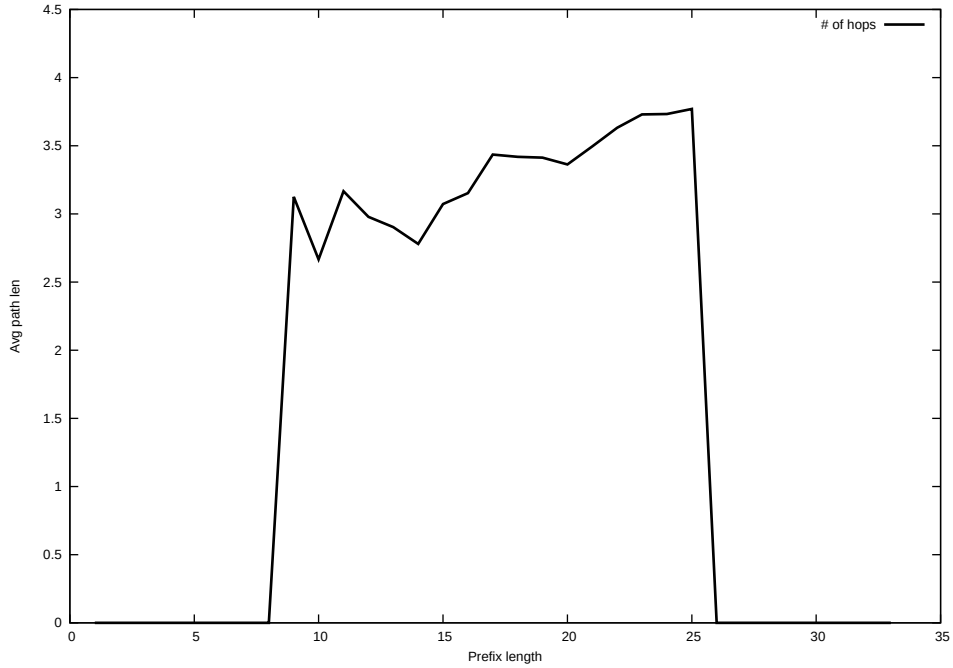
2014-05-17



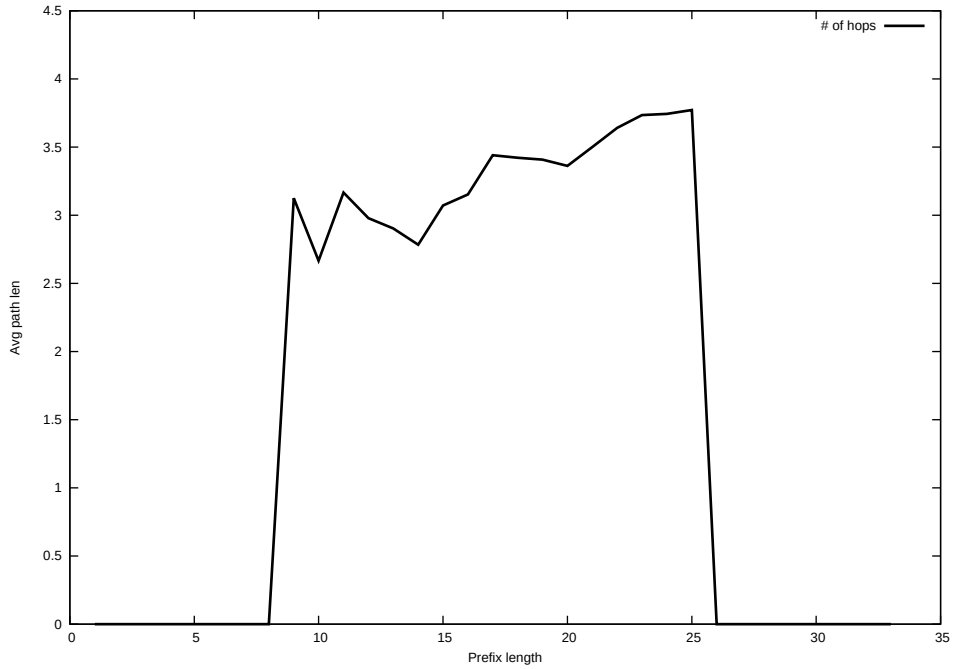
2014-05-18



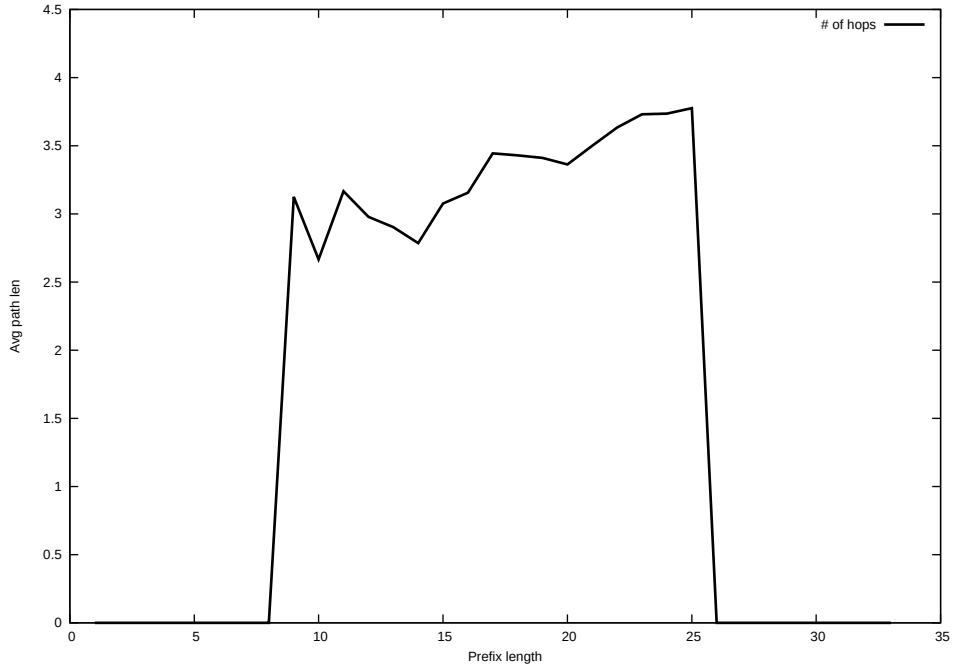
2014-05-19



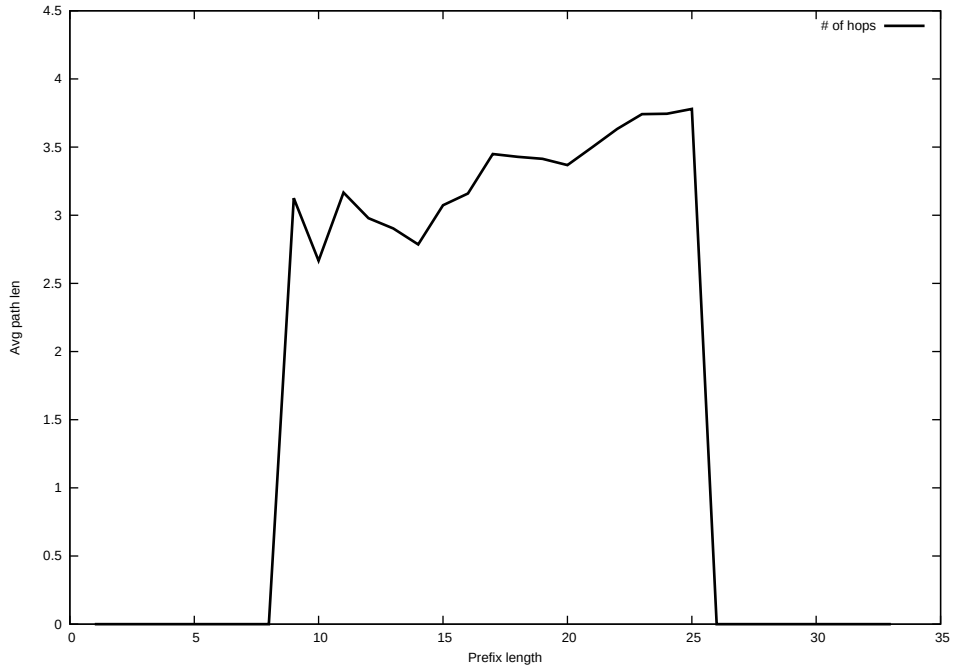
2014-05-20



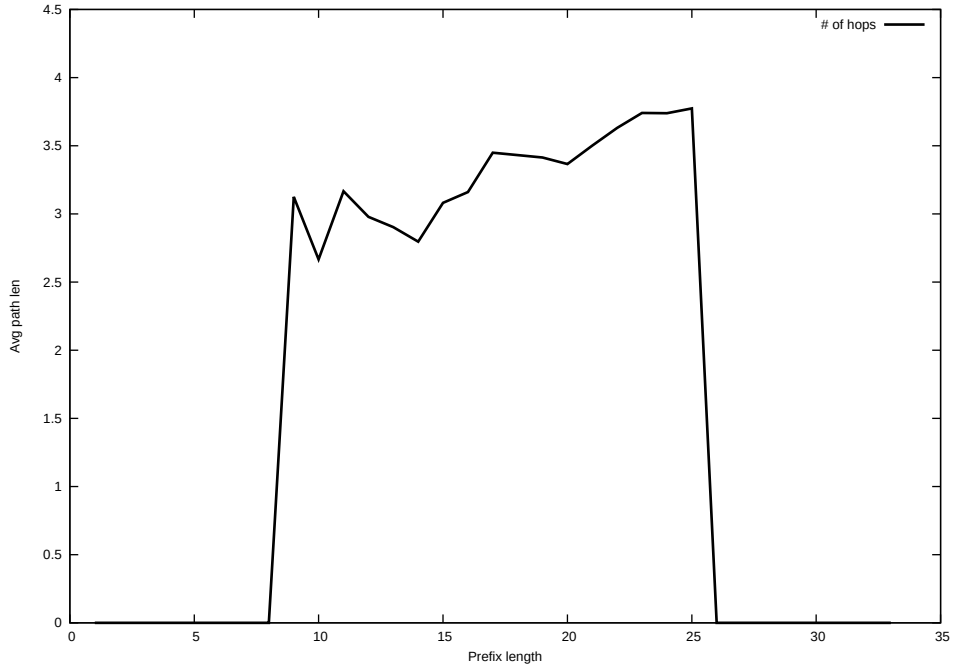
2014-05-21



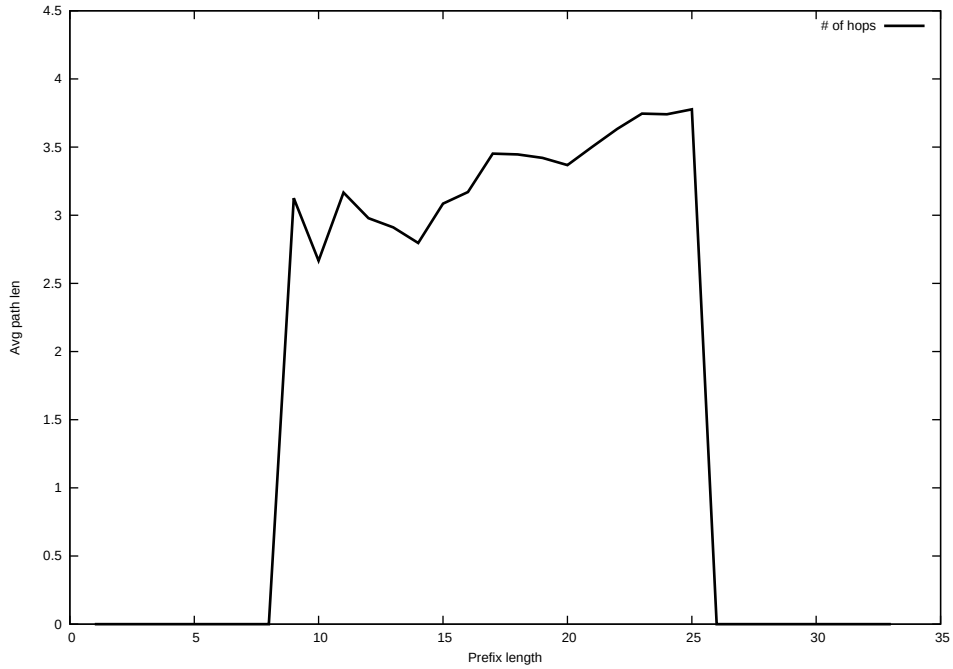
2014-05-22



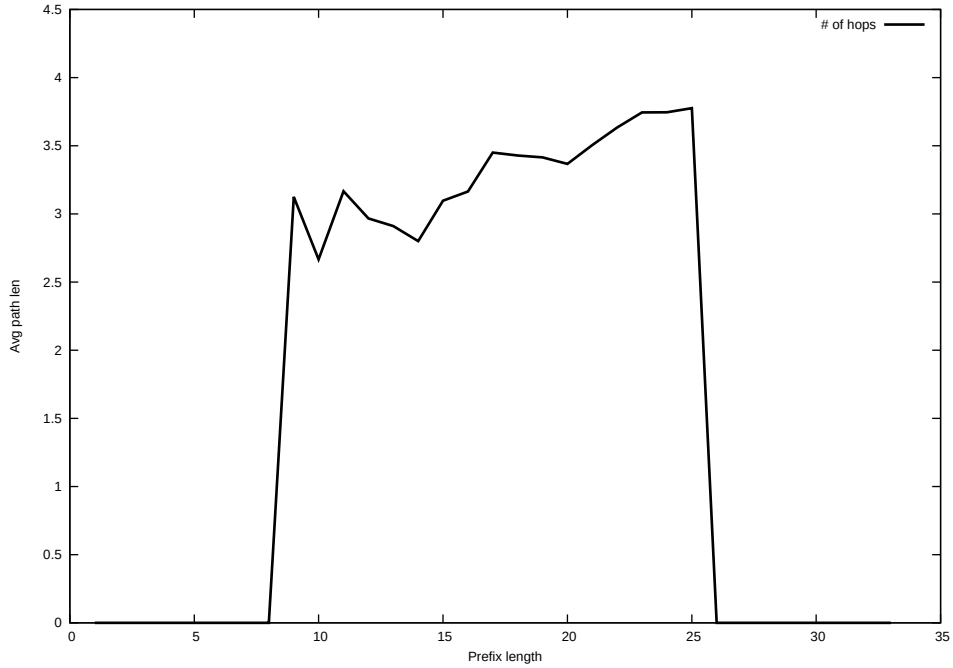
2014-05-23



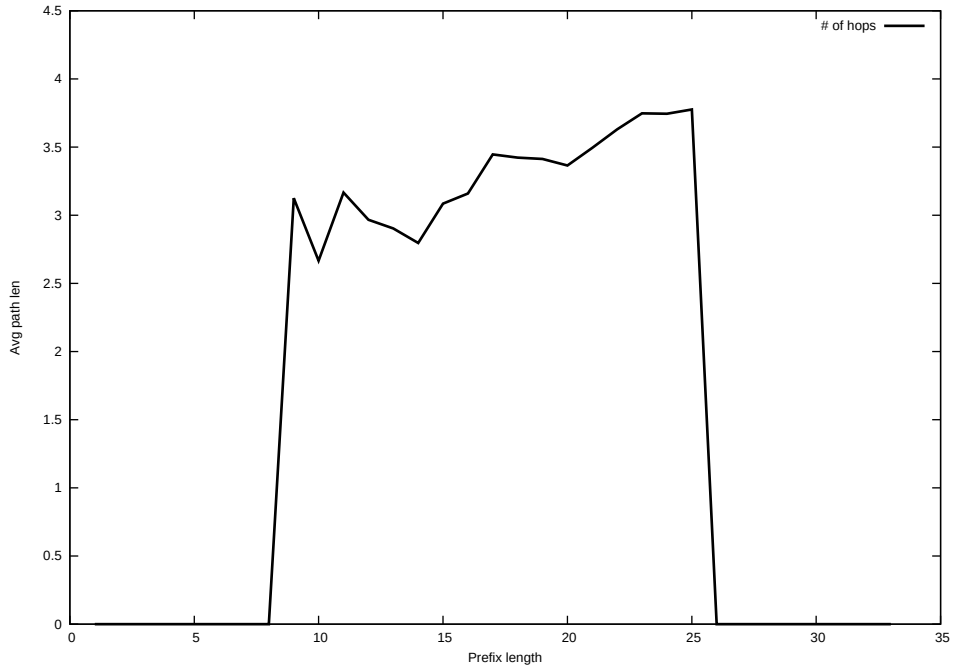
2014-05-24



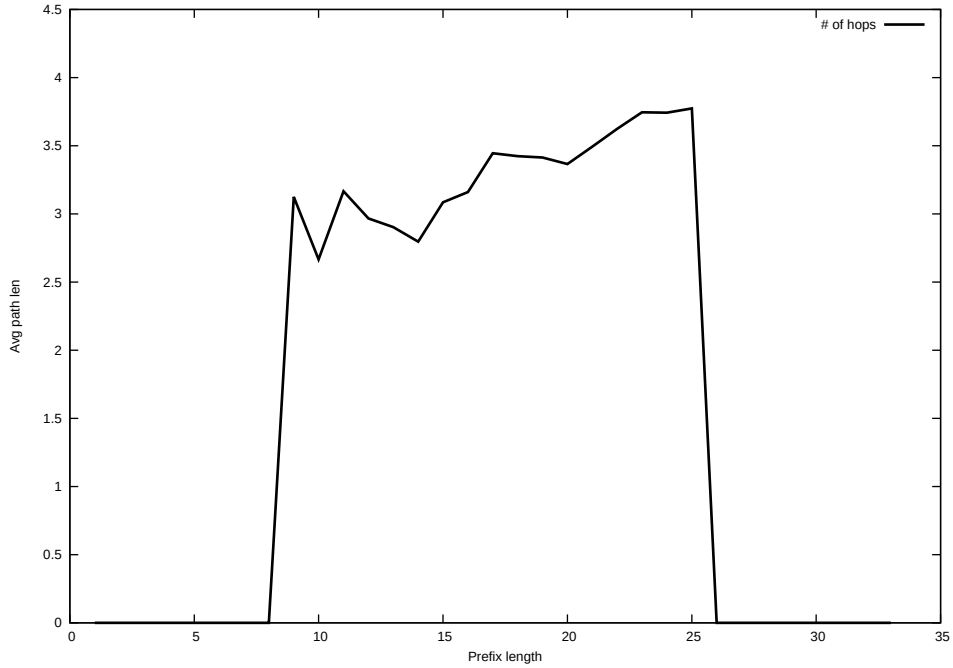
2014-05-25



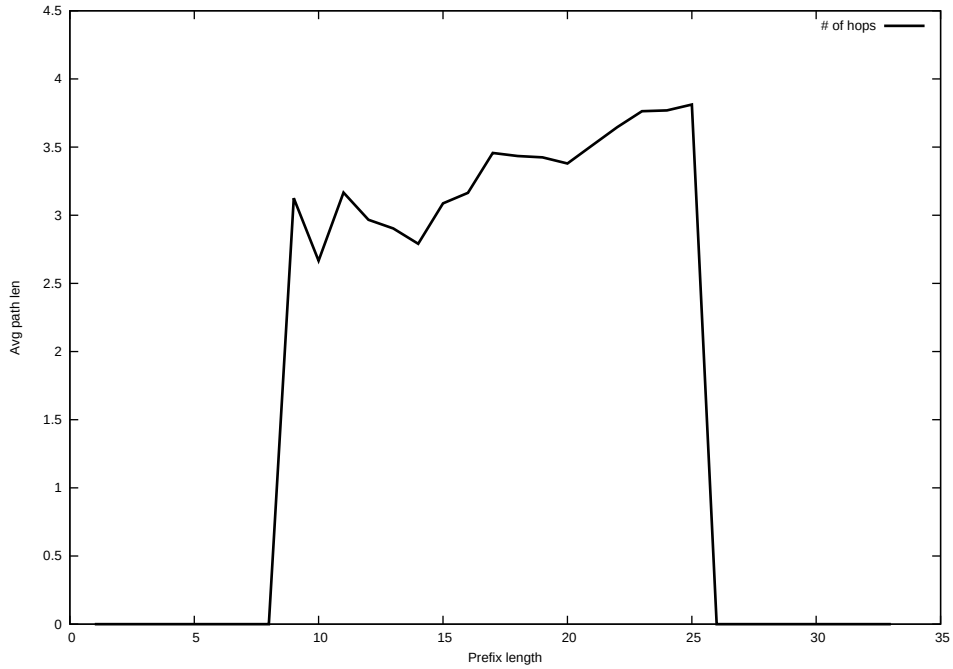
2014-05-26



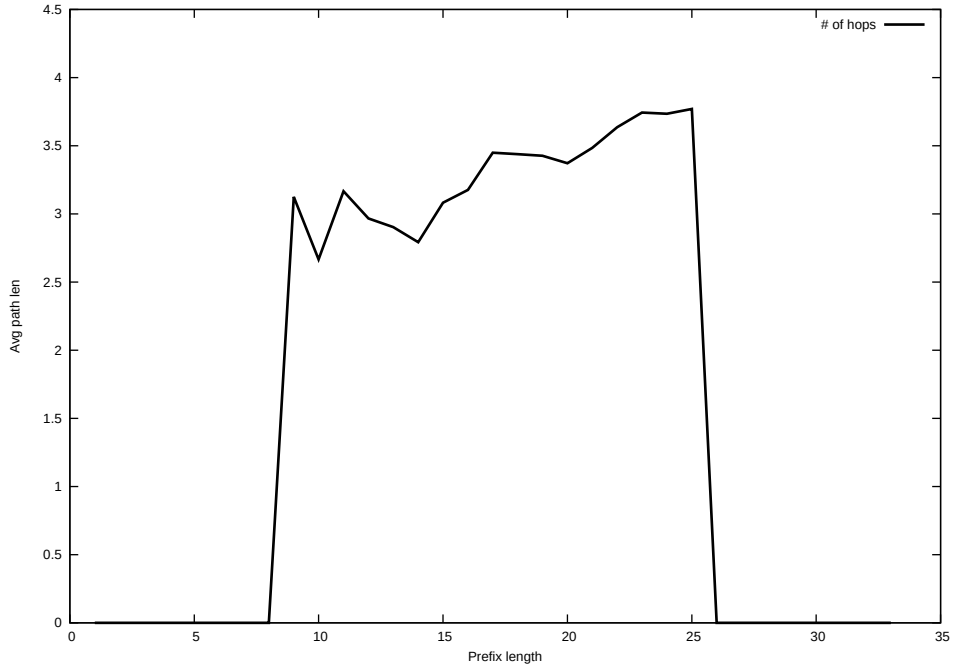
2014-05-27



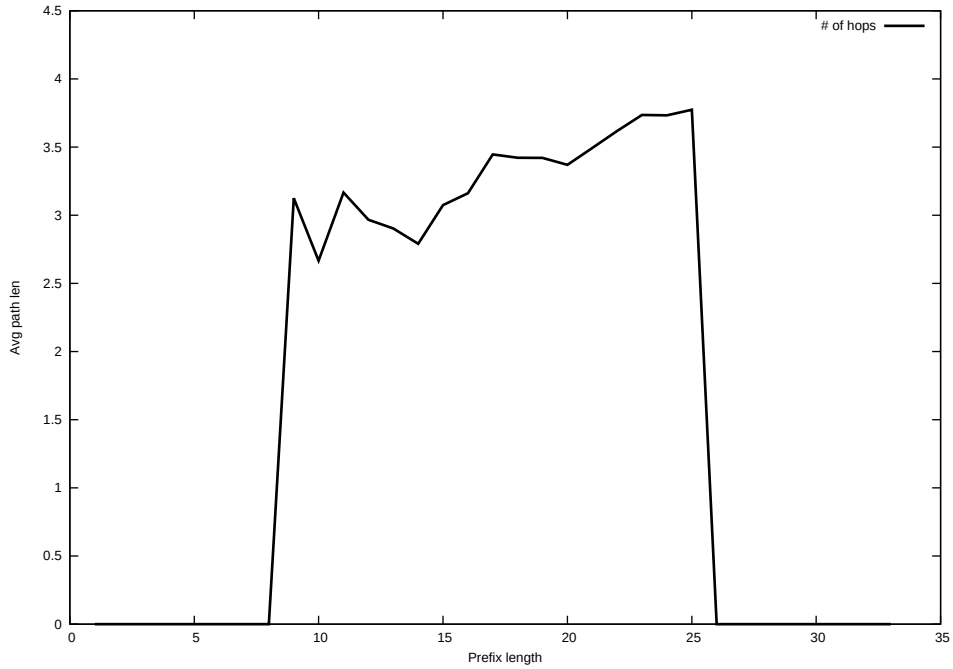
2014-05-28



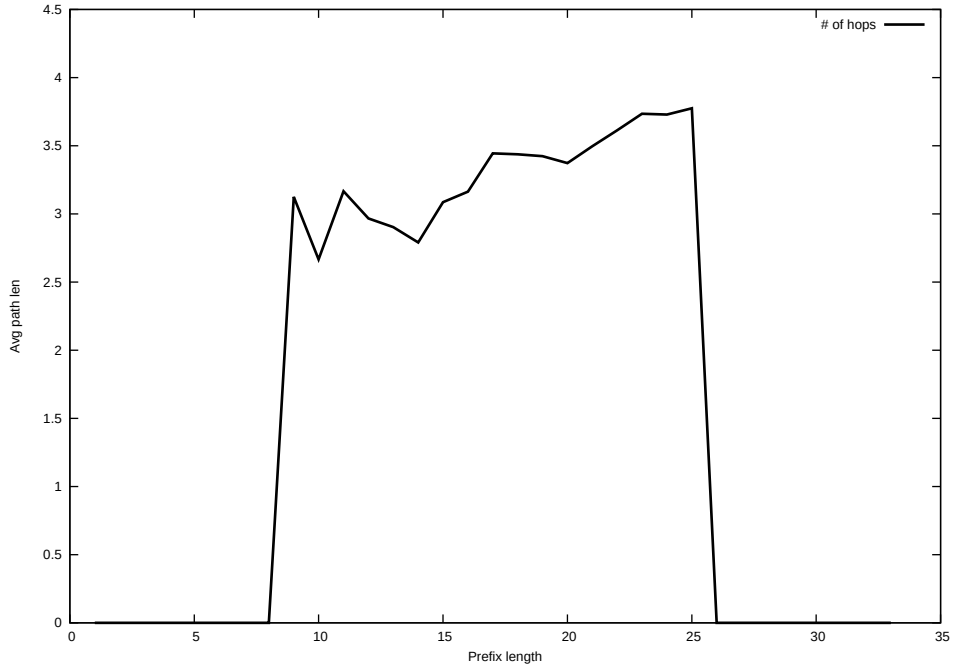
2014-05-29



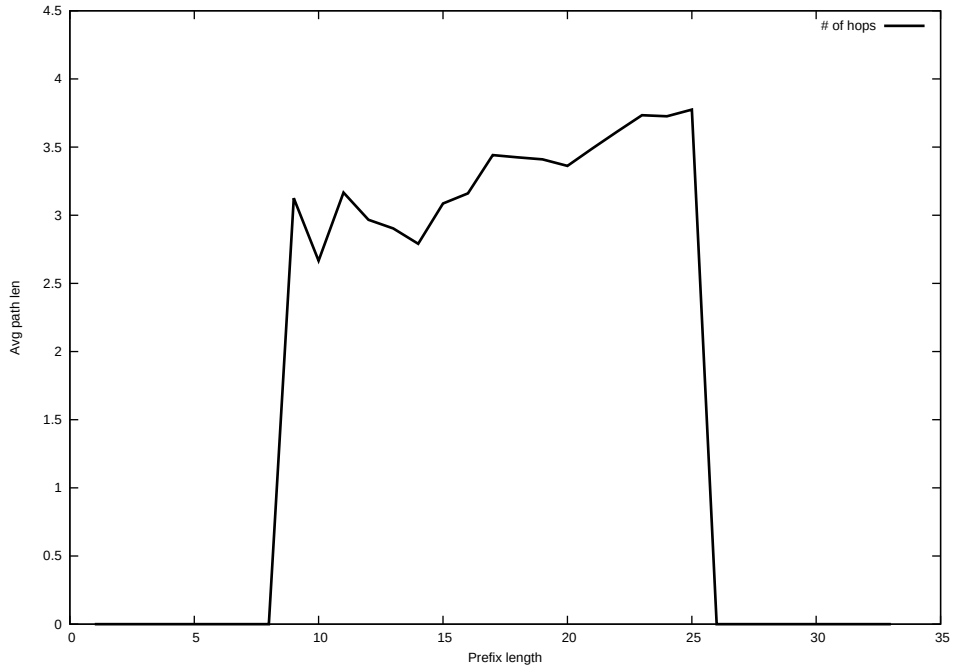
2014-05-30



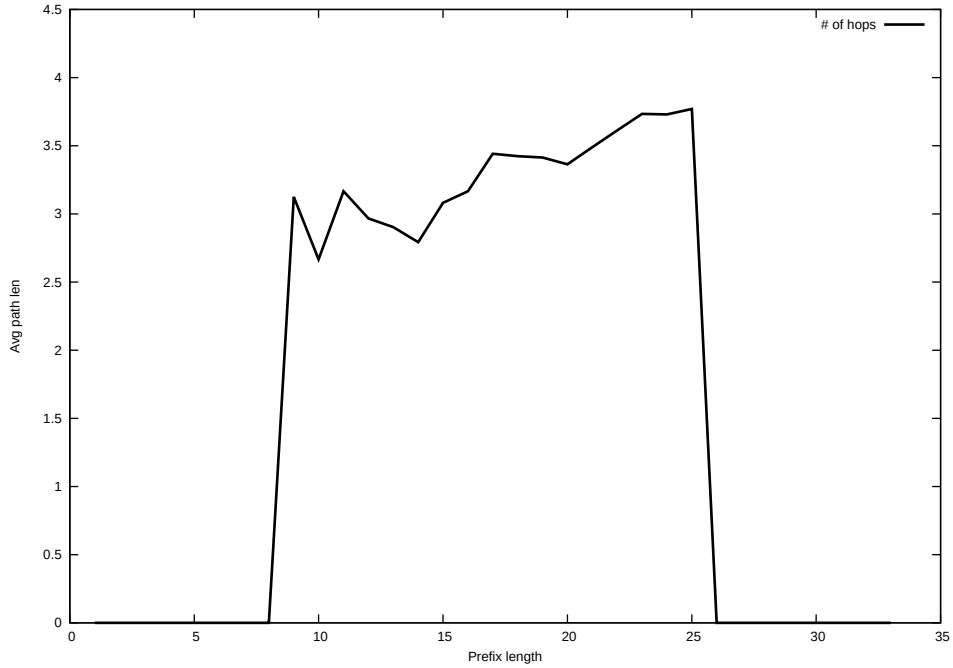
2014-05-31



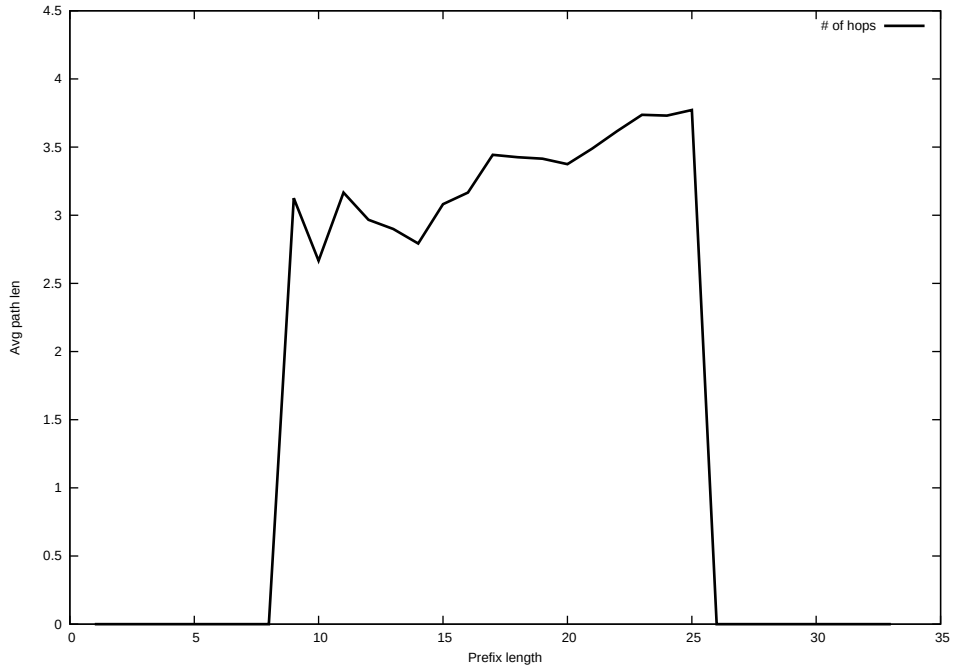
2014-06-01



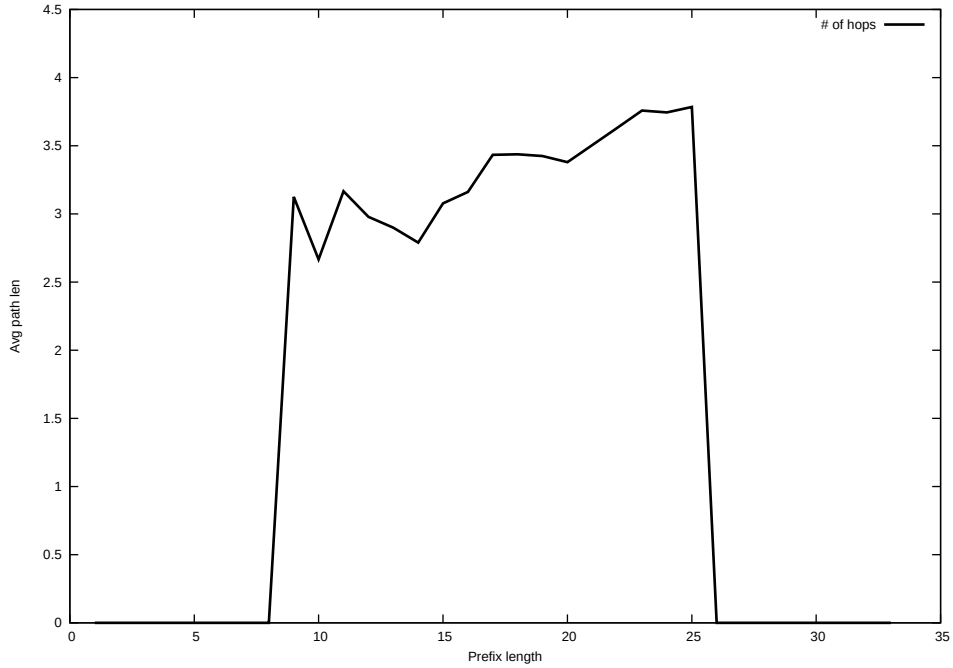
2014-06-02



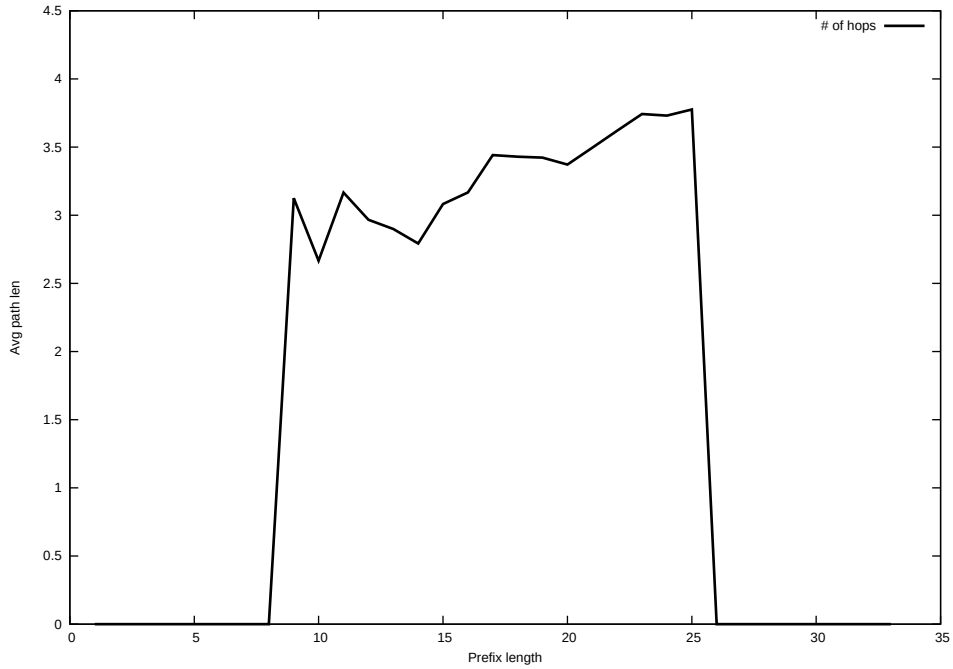
2014-06-03



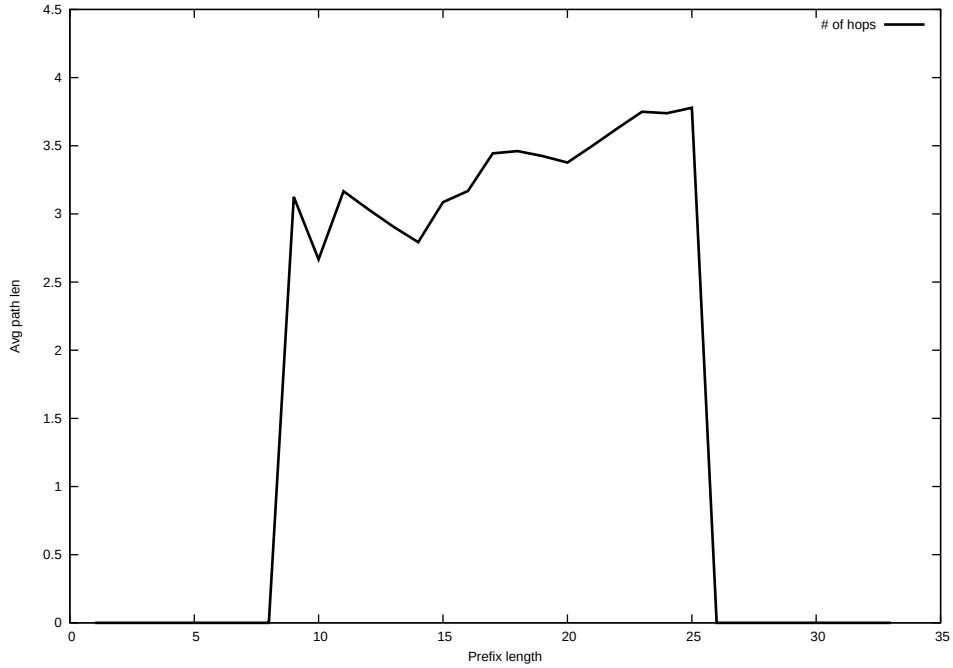
2014-06-04



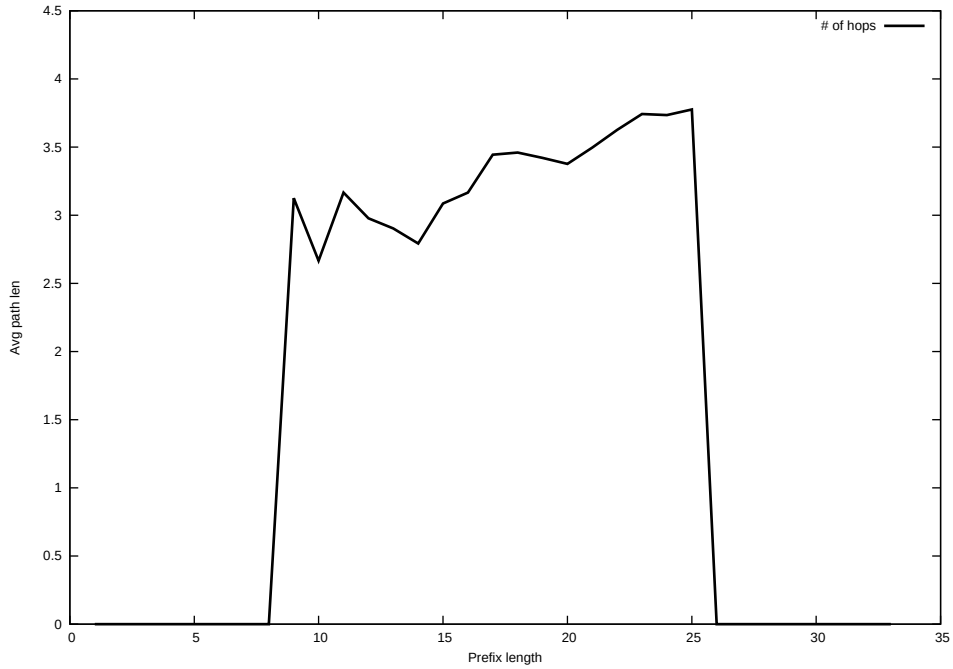
2014-06-05



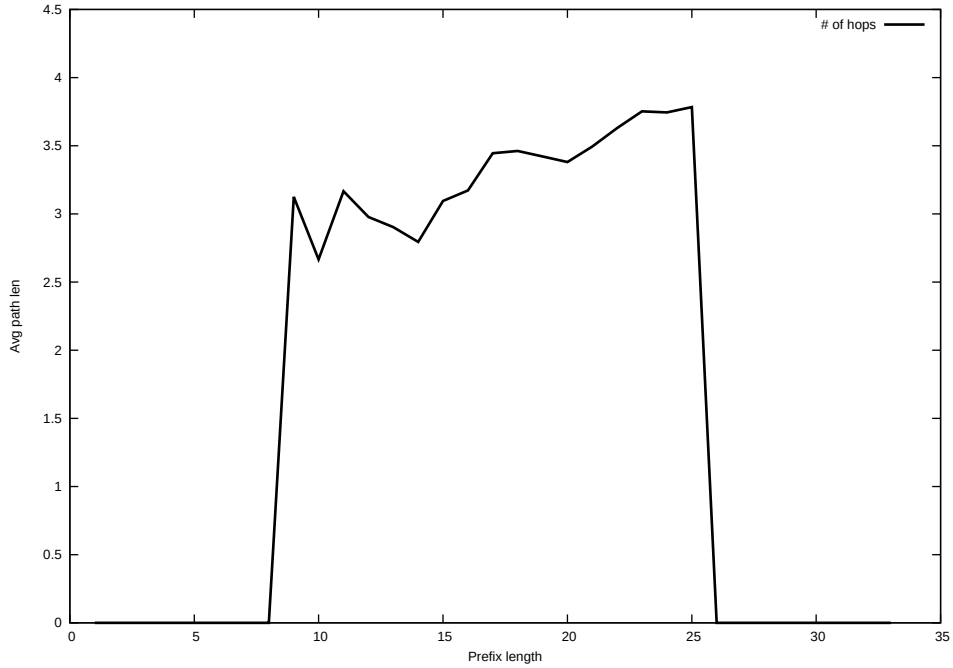
2014-06-06



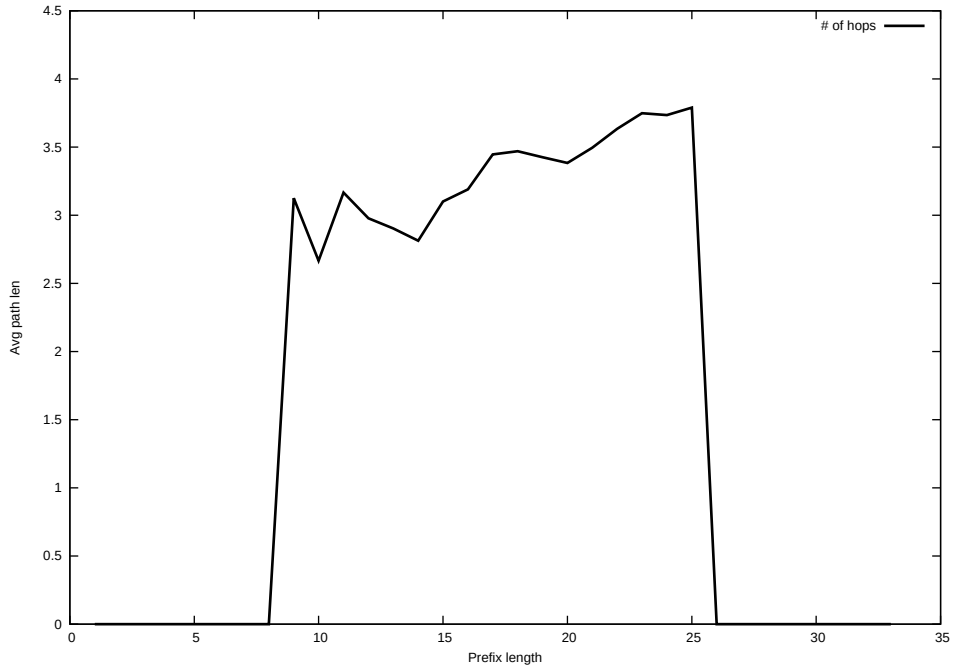
2014-06-07



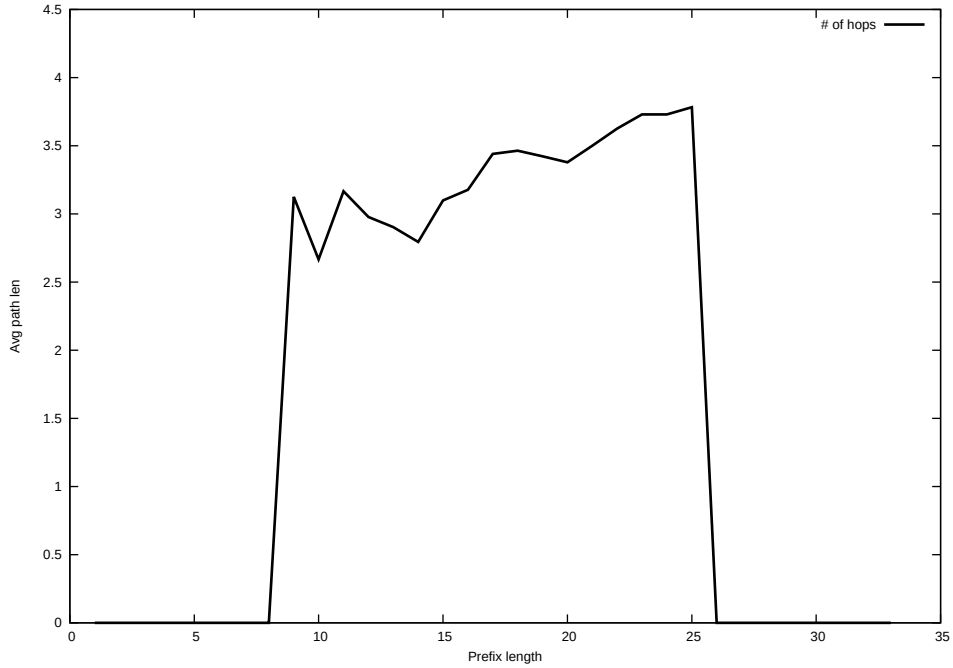
2014-06-08



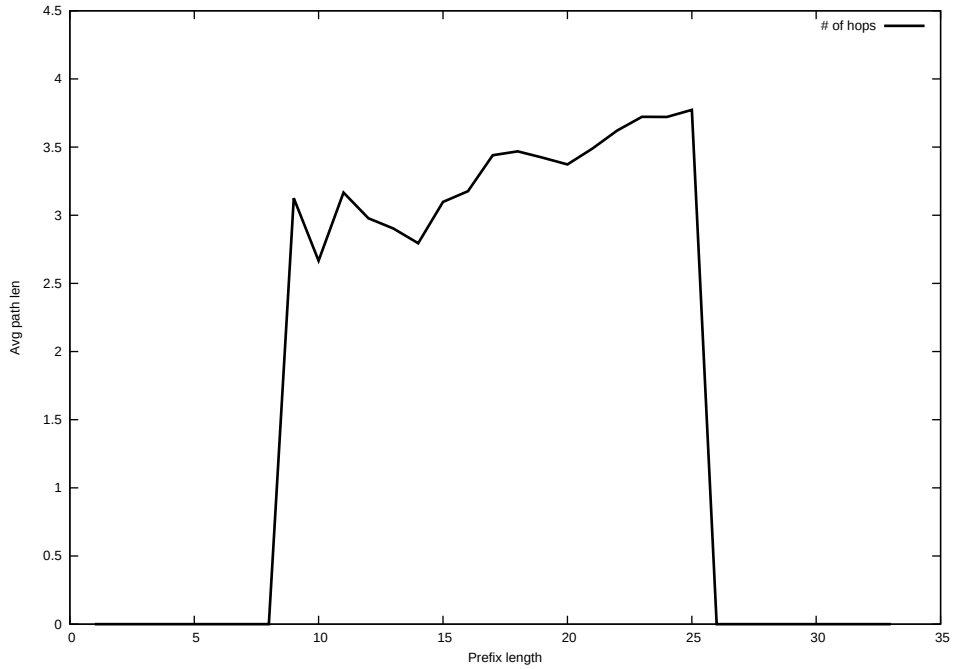
2014-06-09



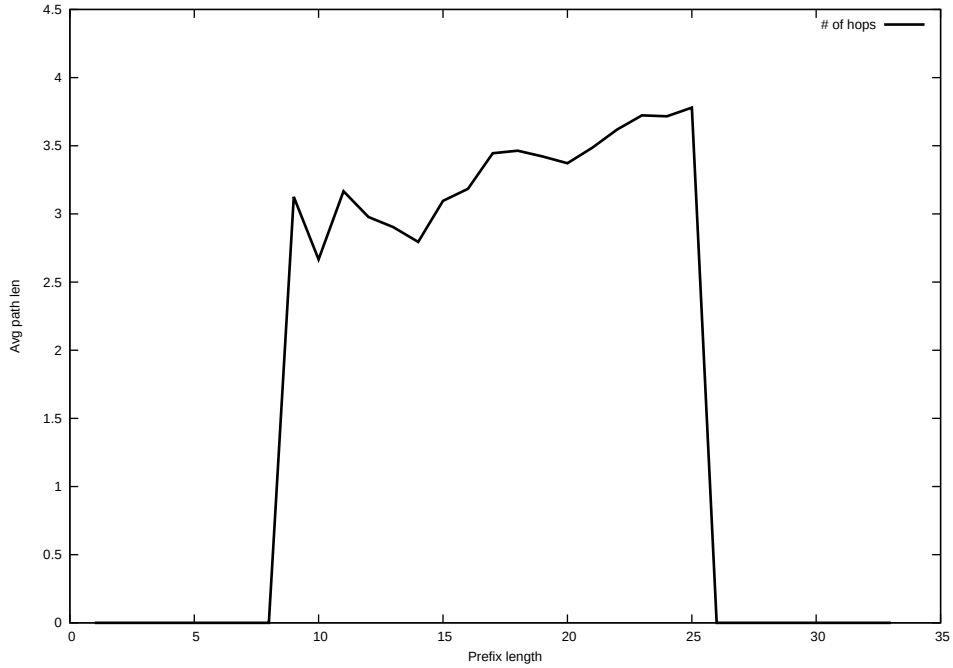
2014-06-10



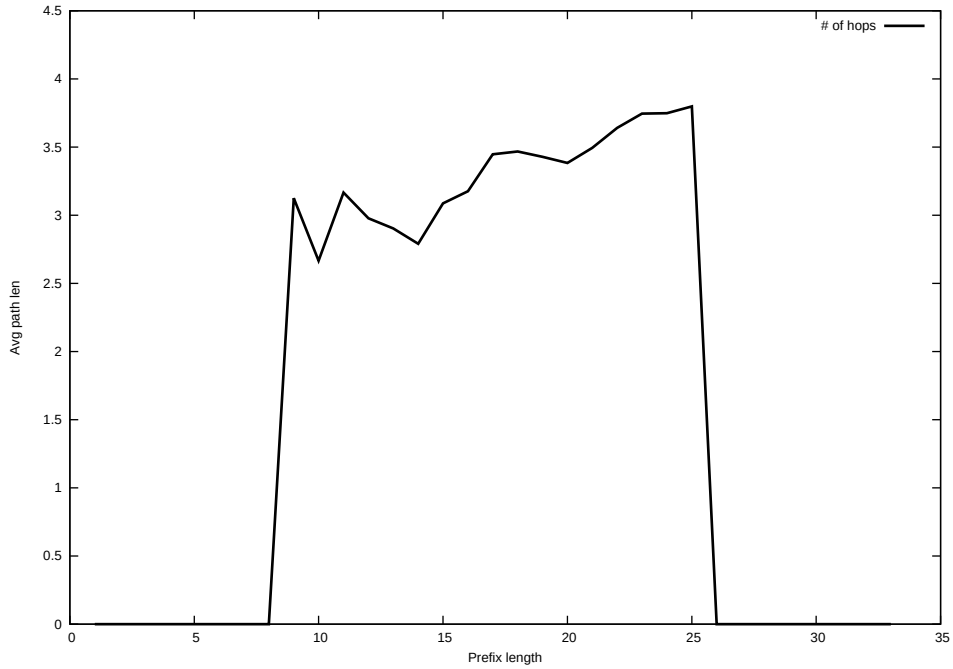
2014-06-11



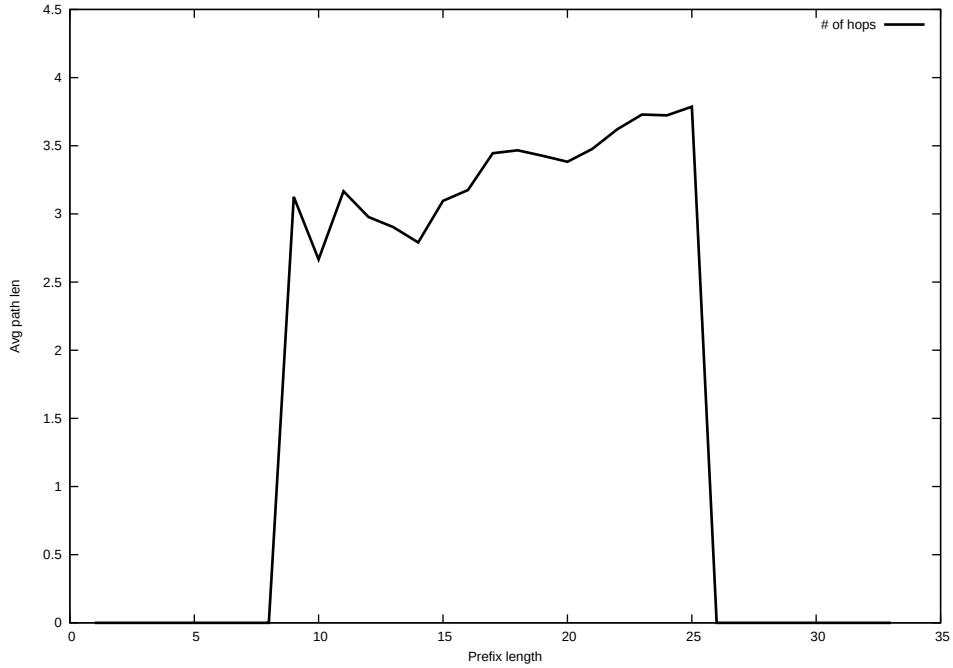
2014-06-12



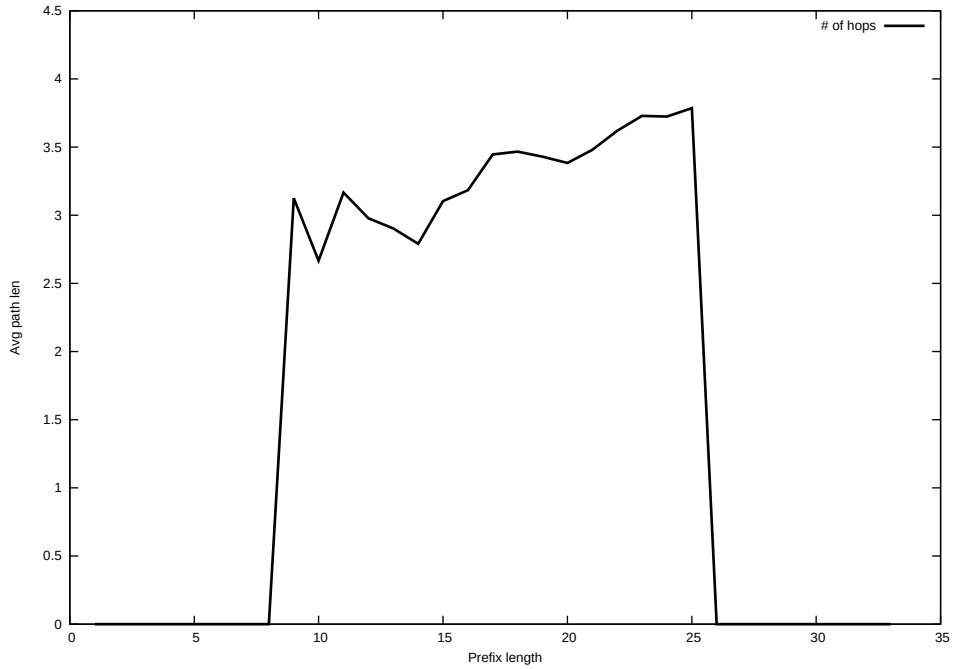
2014-06-13



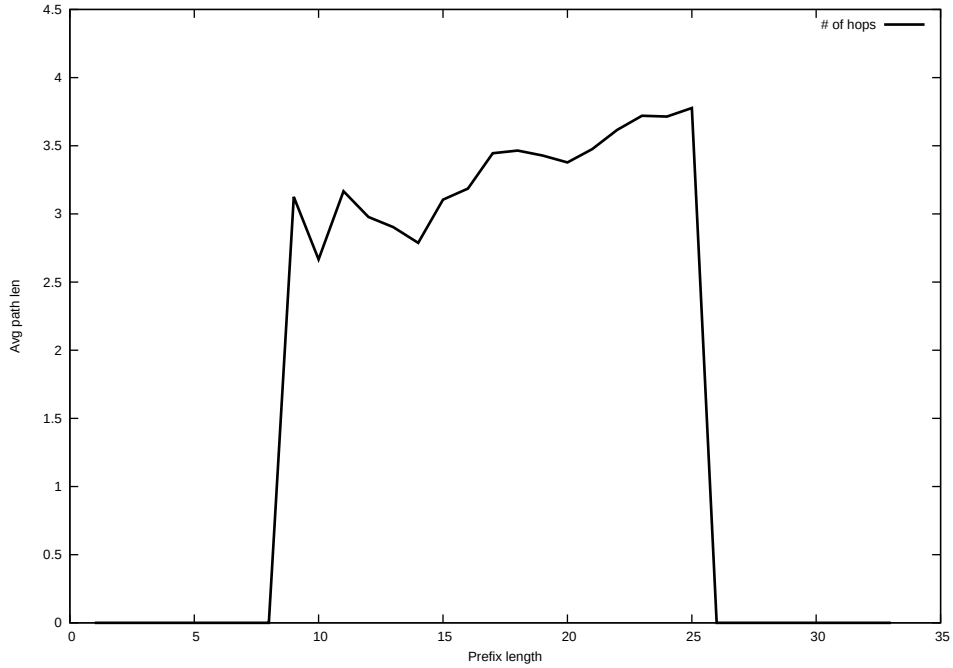
2014-06-14



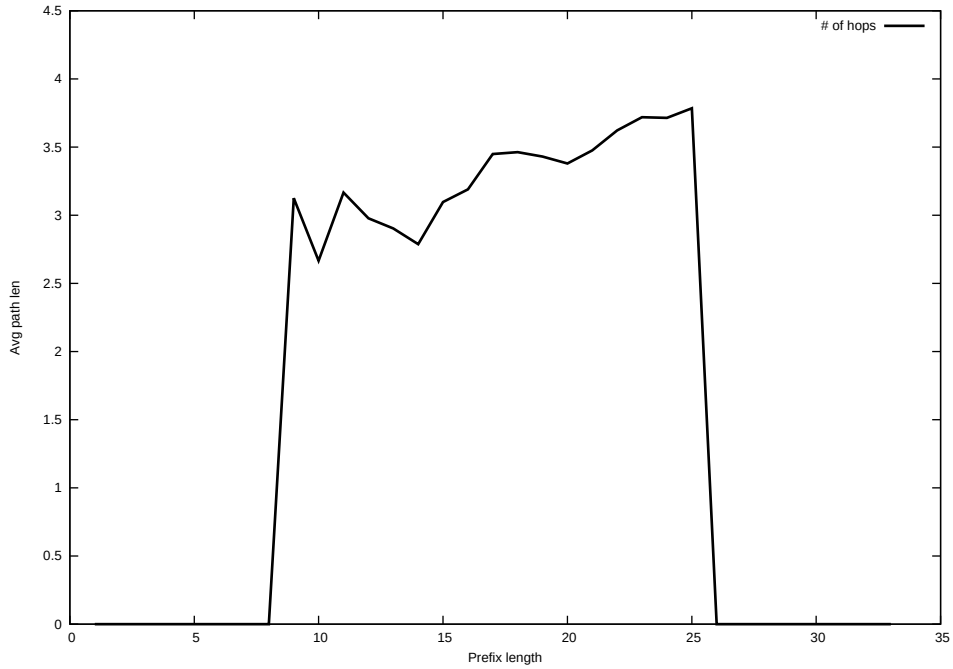
2014-06-15



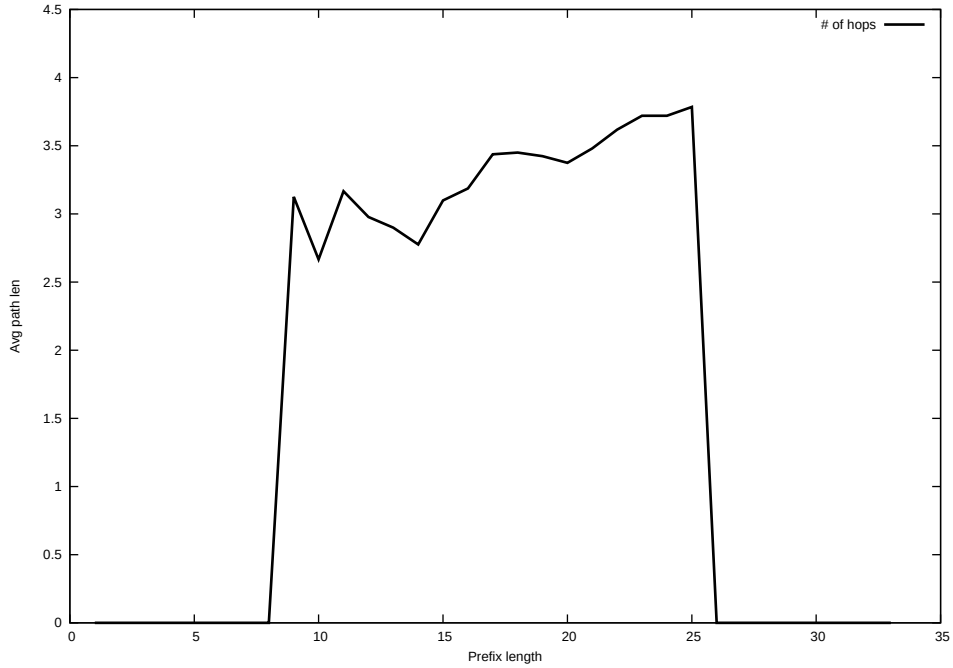
2014-06-16



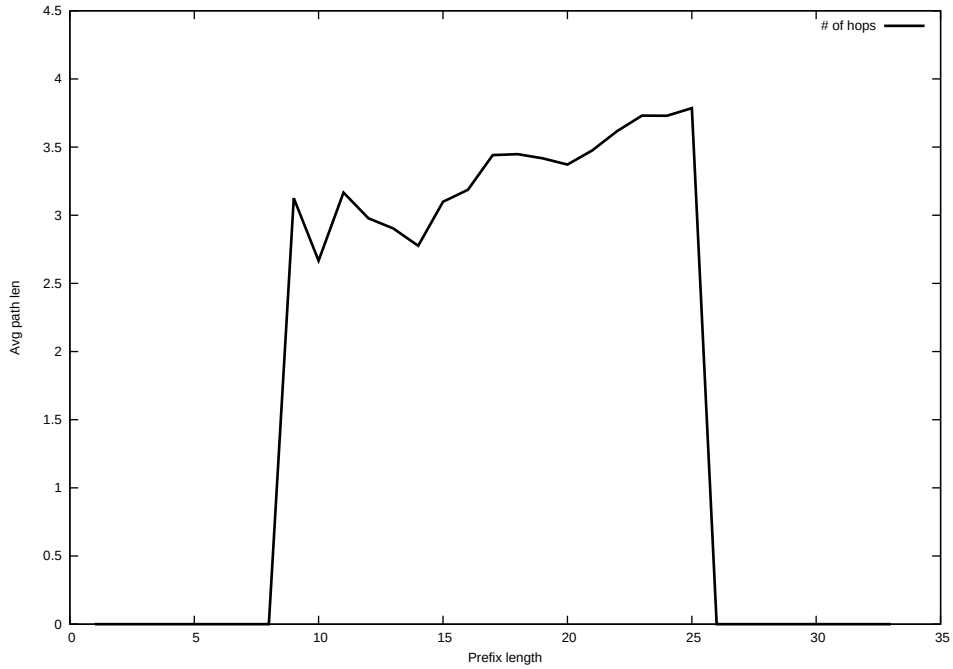
2014-06-17



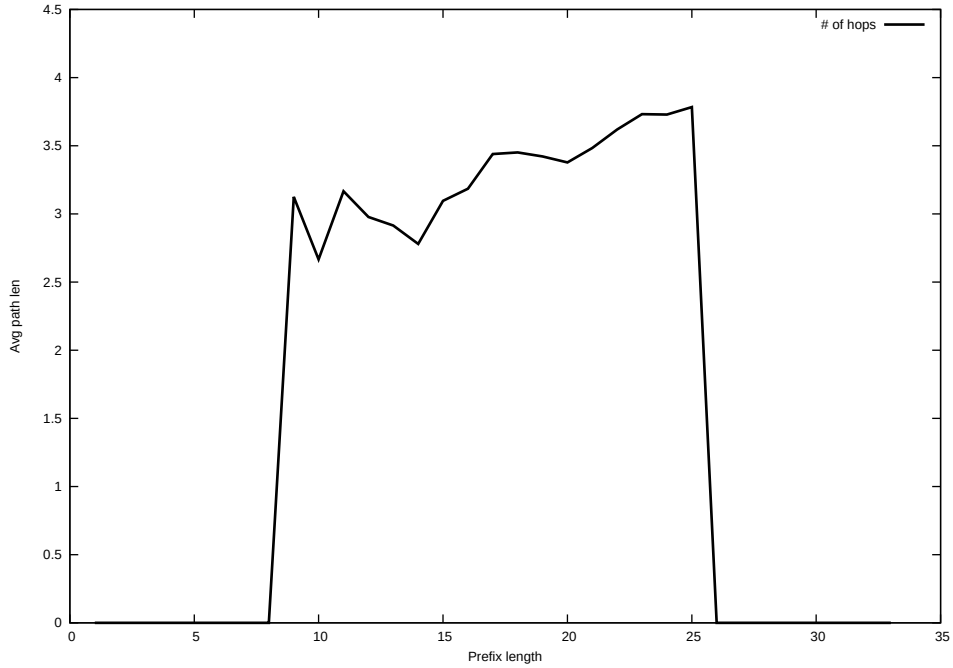
2014-06-18



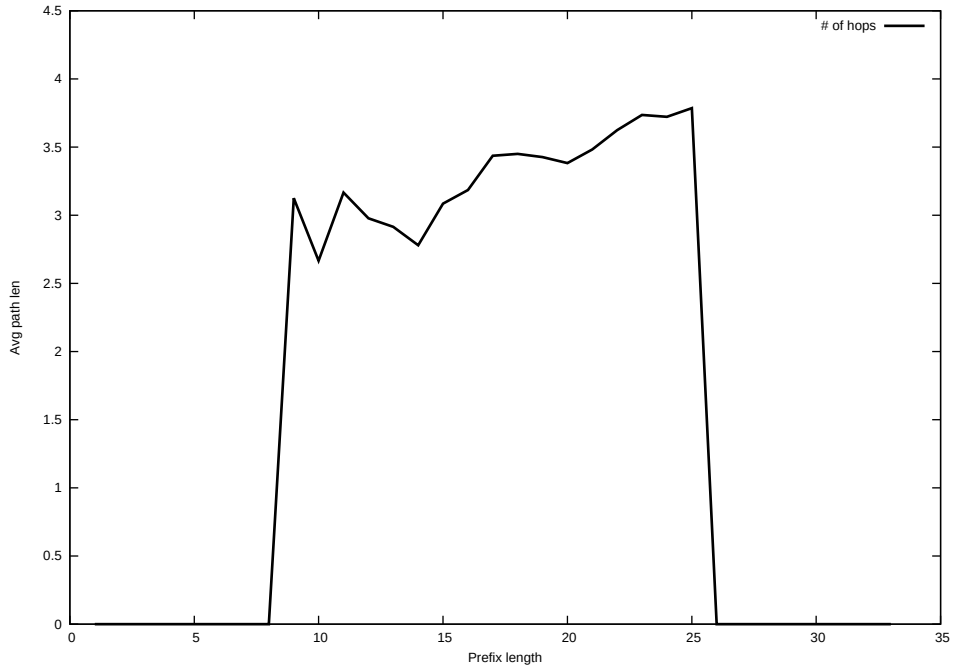
2014-06-19



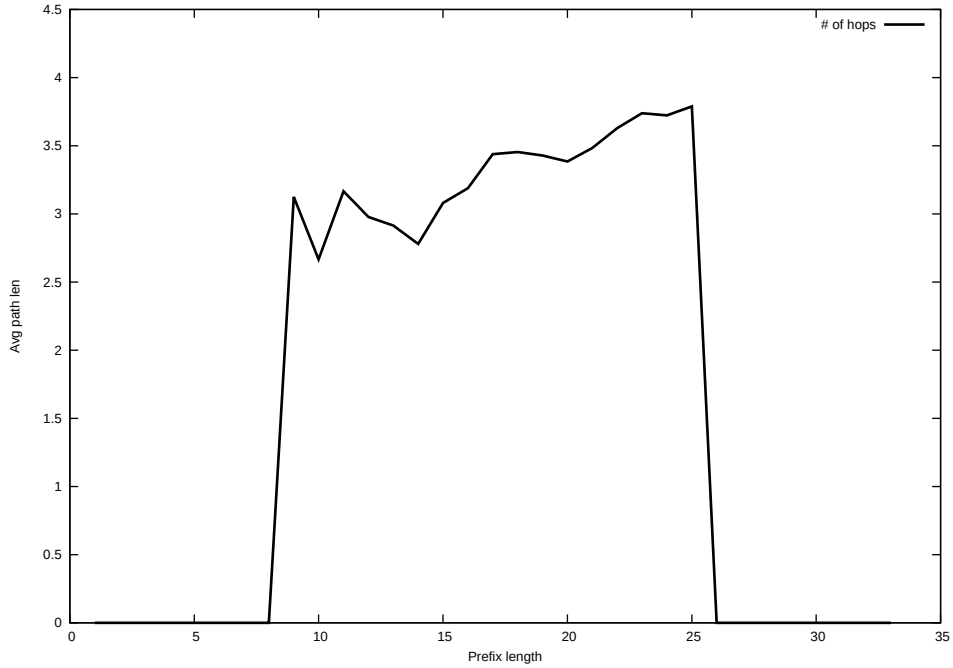
2014-06-20



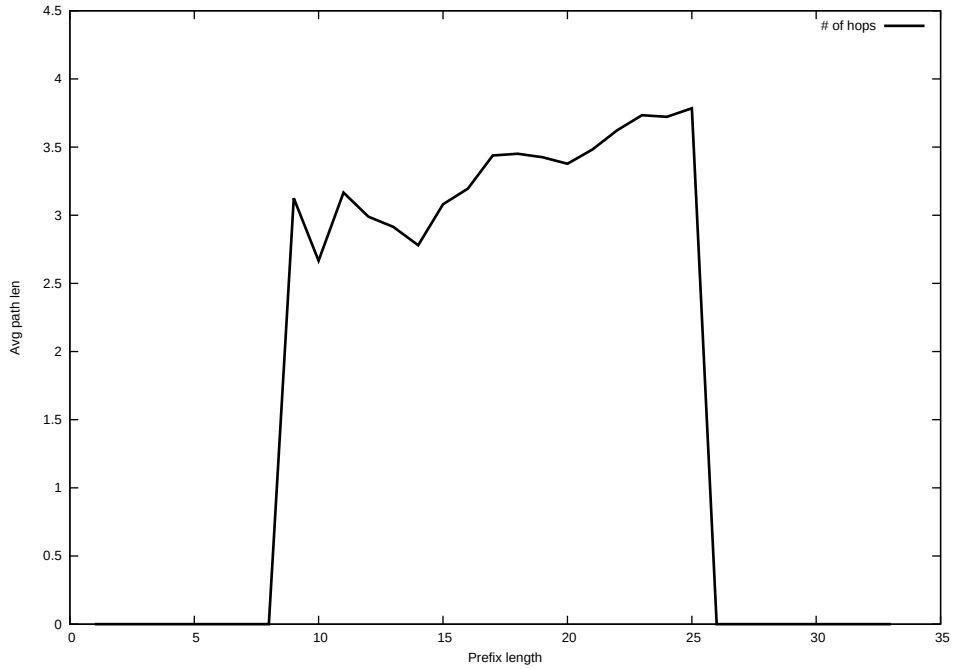
2014-06-21



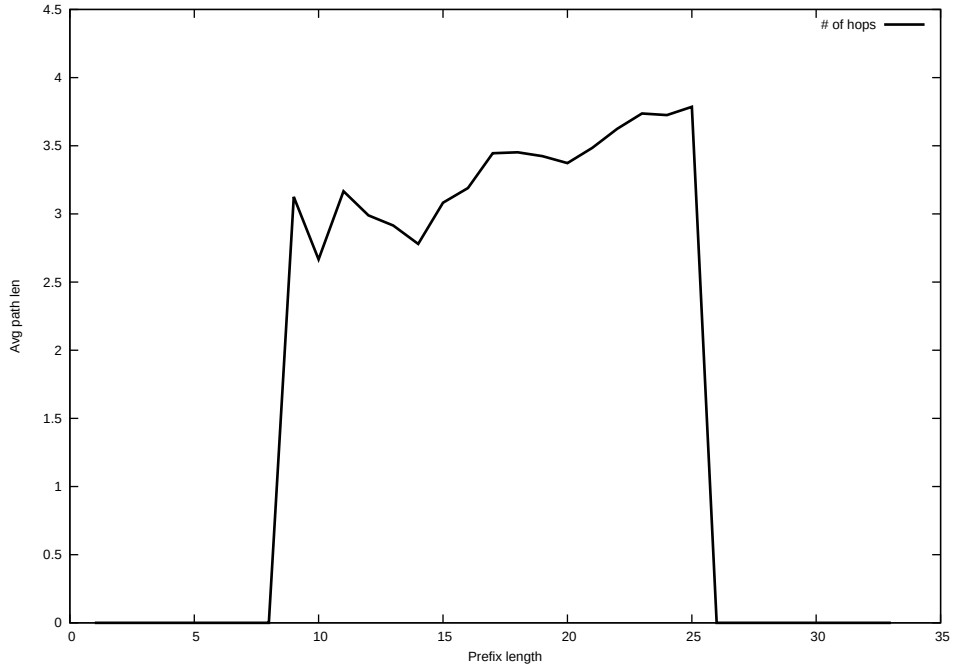
2014-06-22



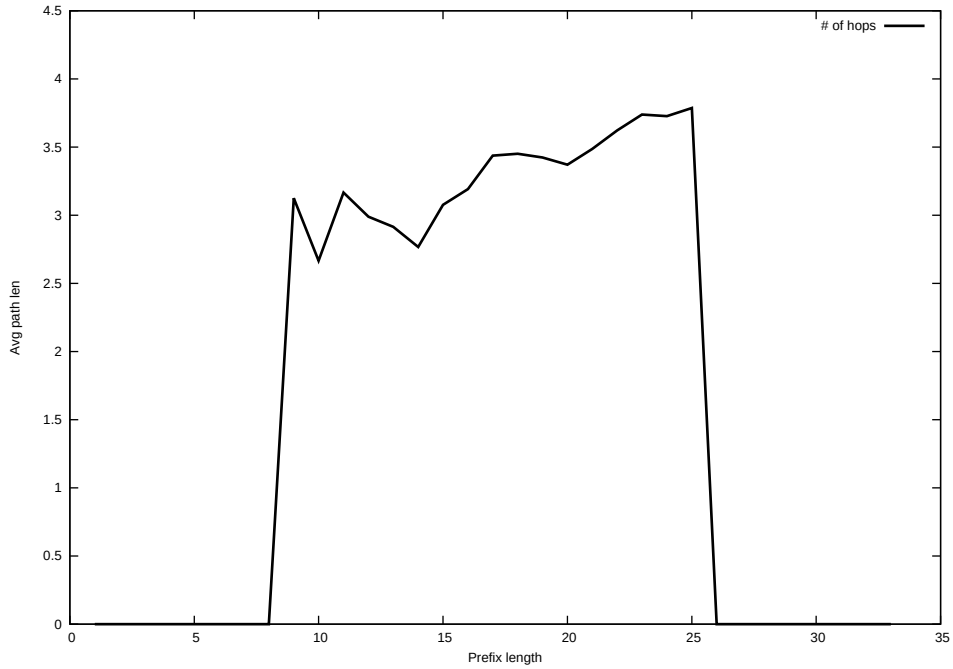
2014-06-23



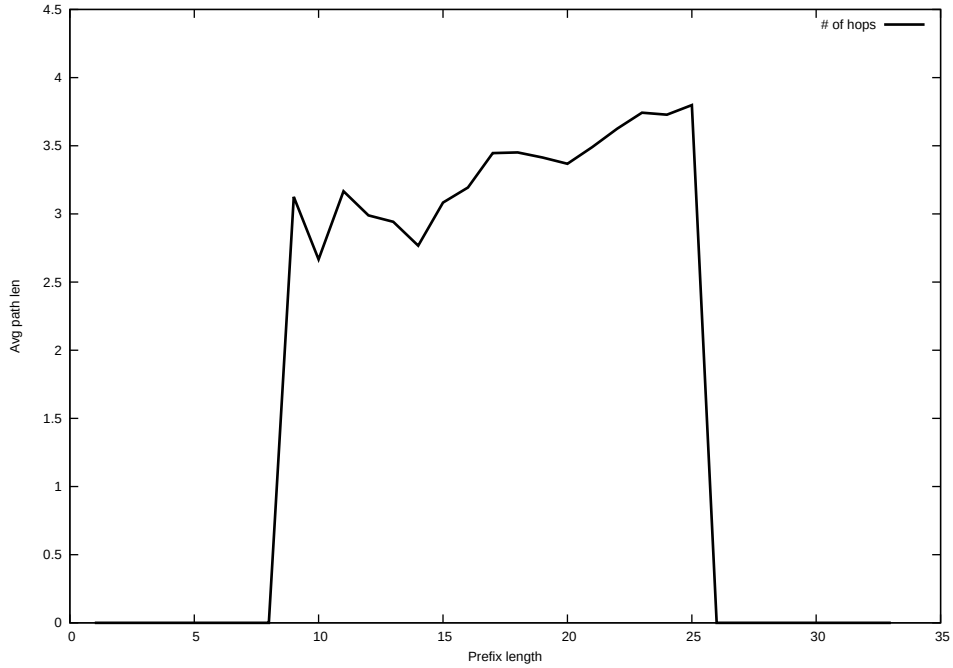
2014-06-24



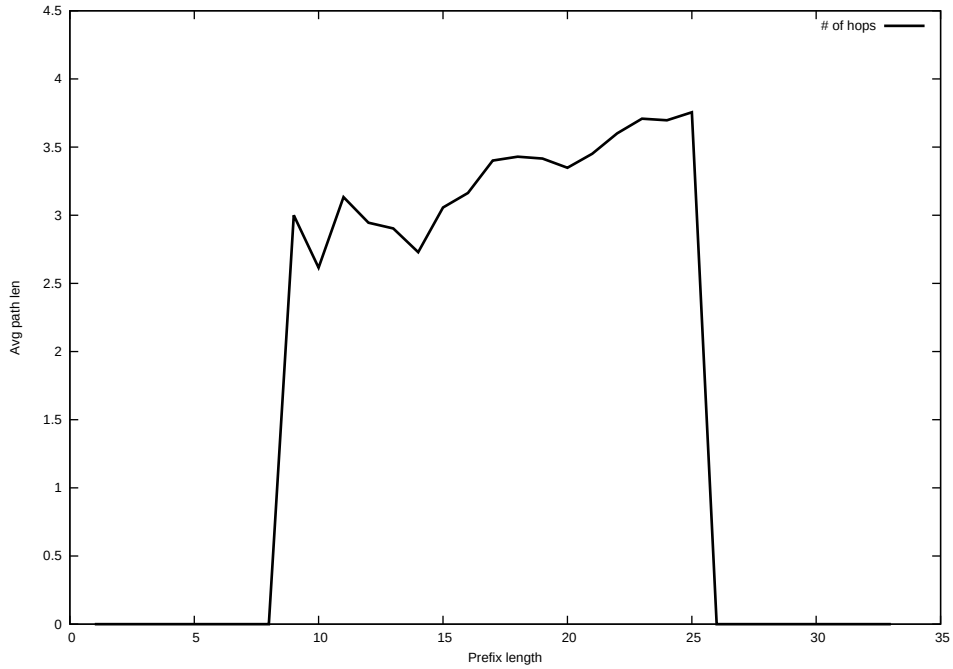
2014-06-25



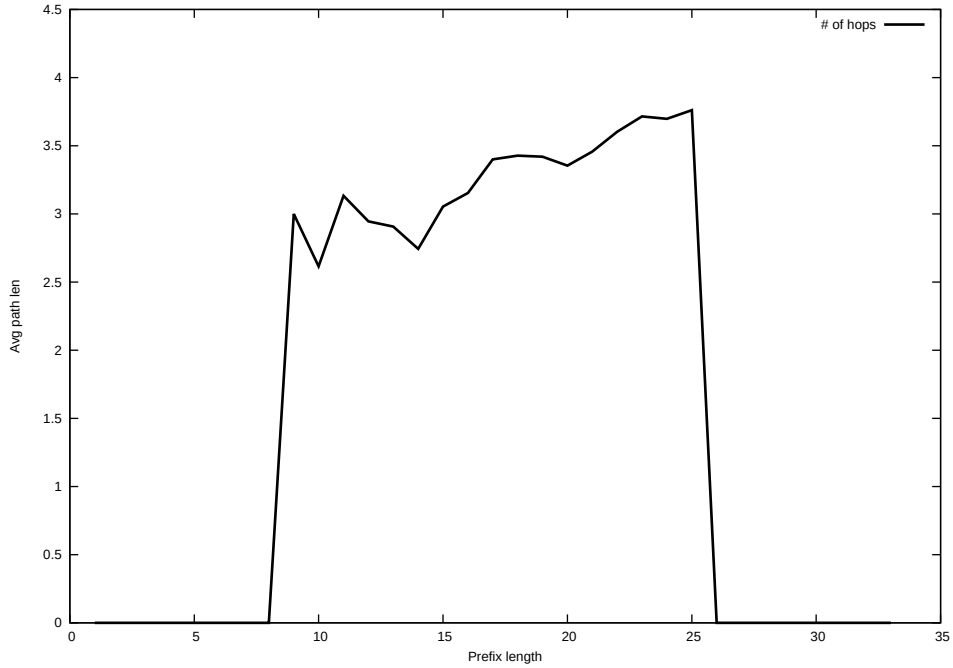
2014-06-26



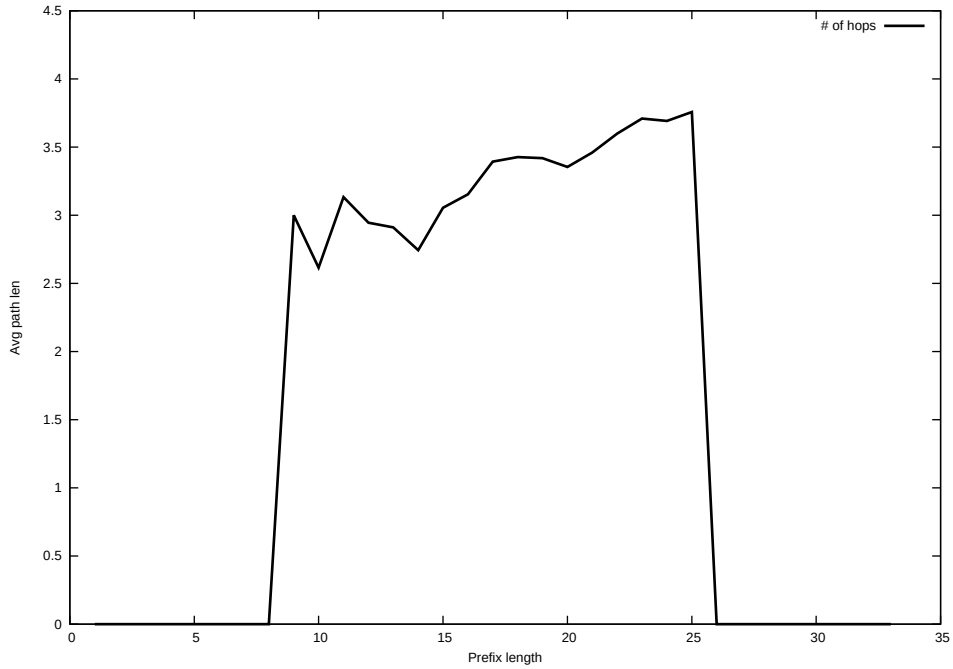
2014-06-27



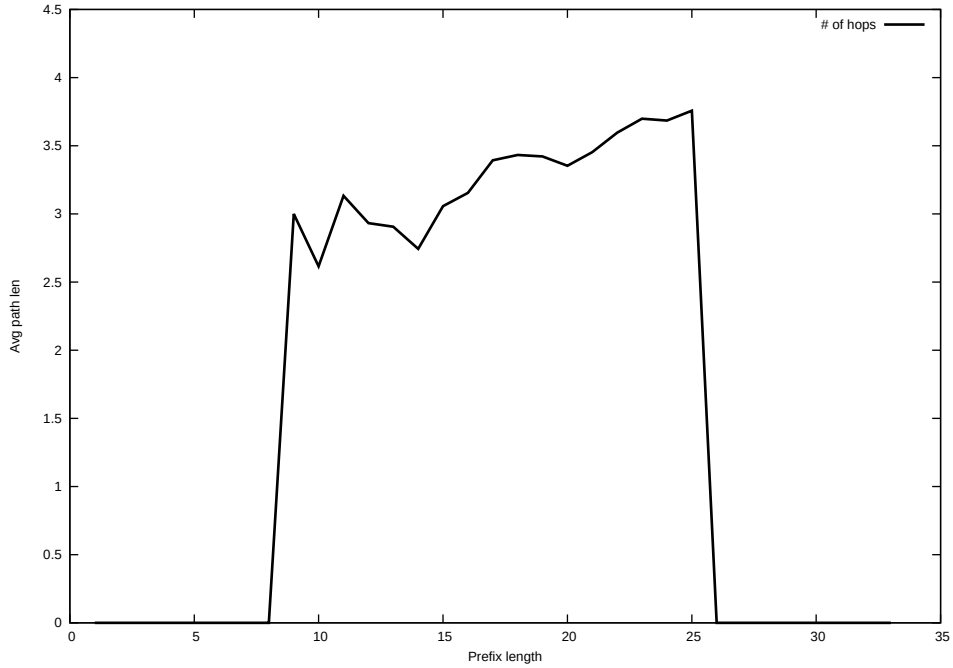
2014-06-28



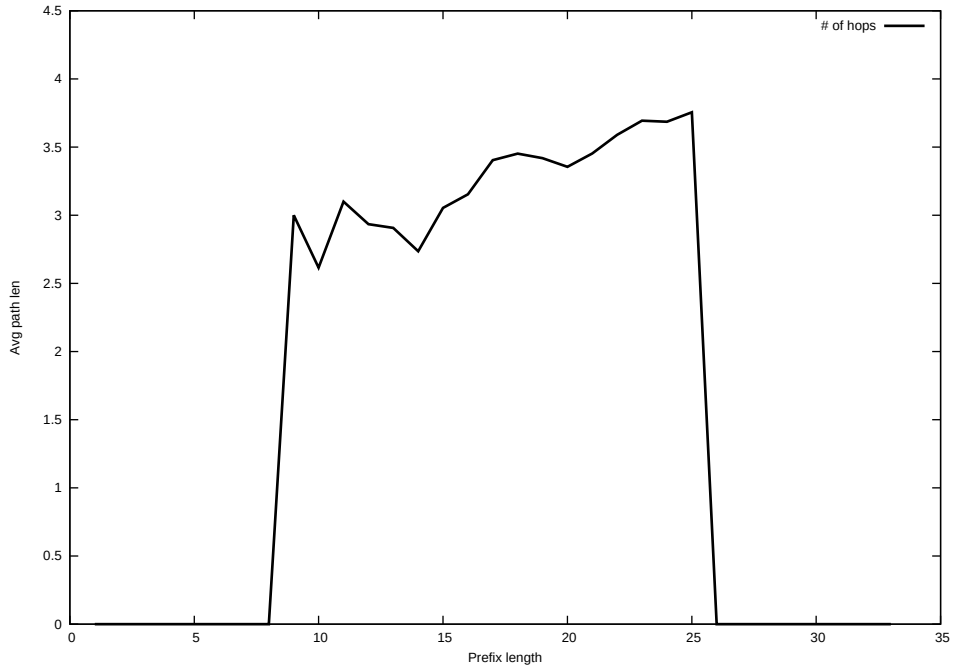
2014-06-29



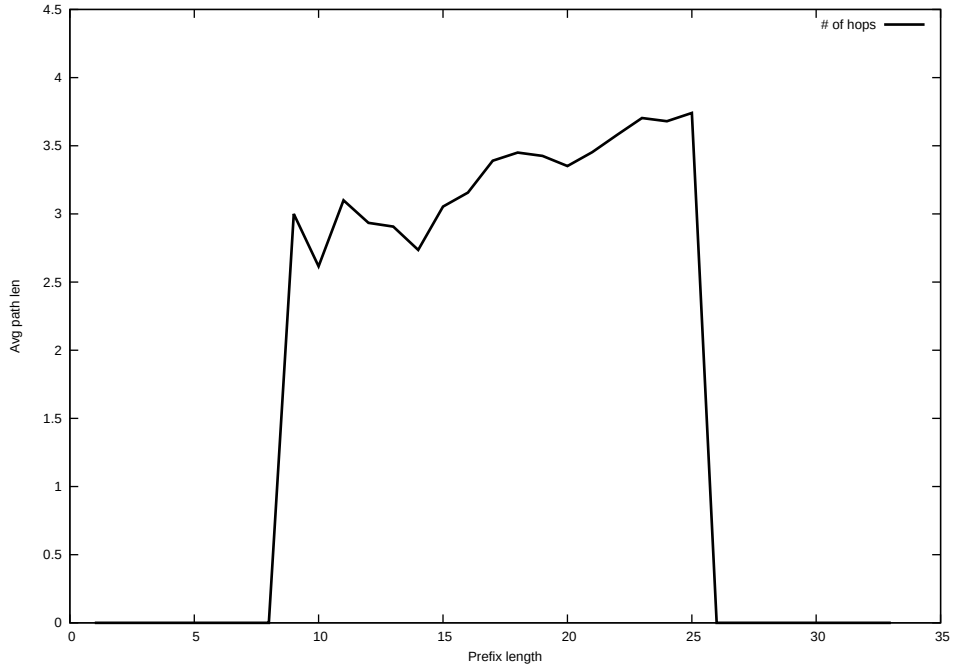
2014-06-30



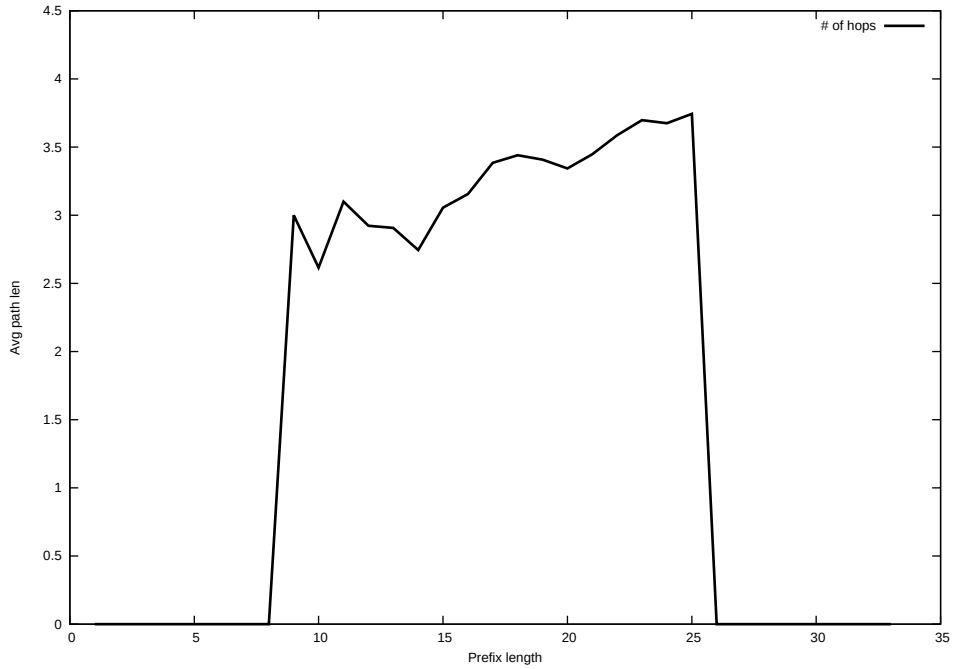
2014-07-01



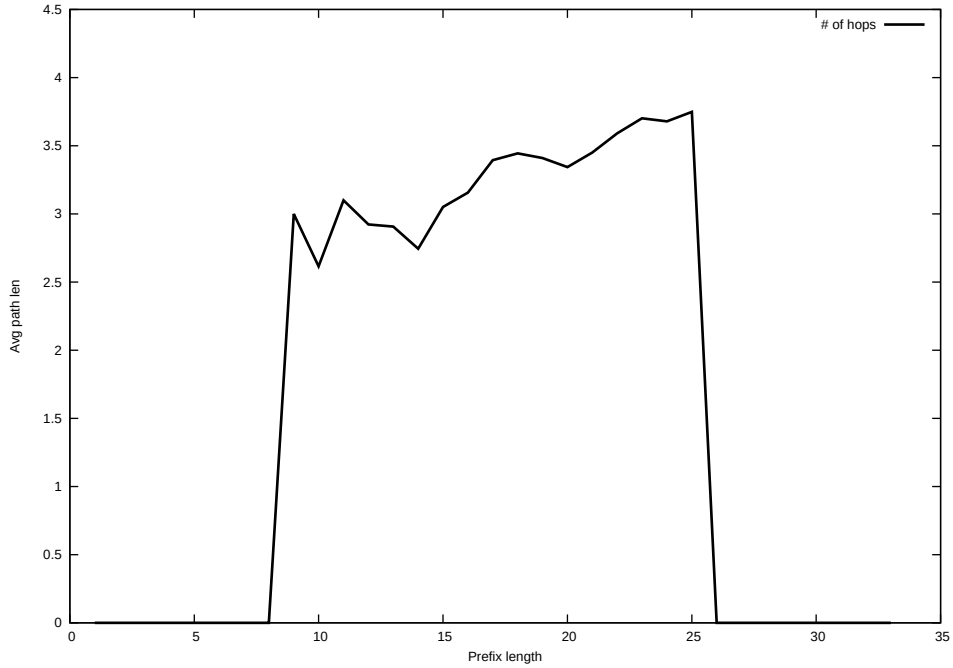
2014-07-02



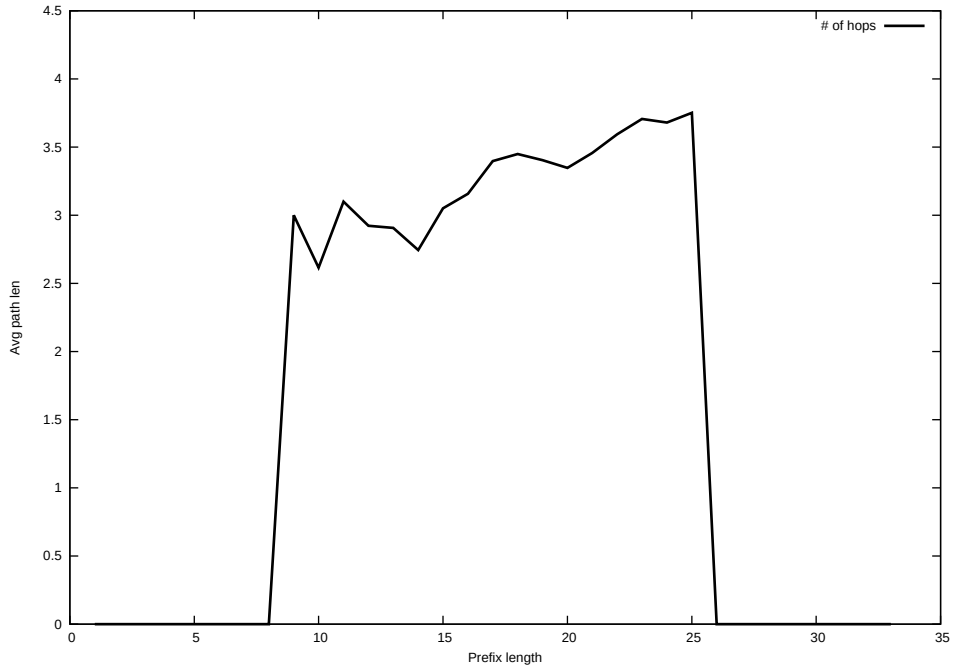
2014-07-03



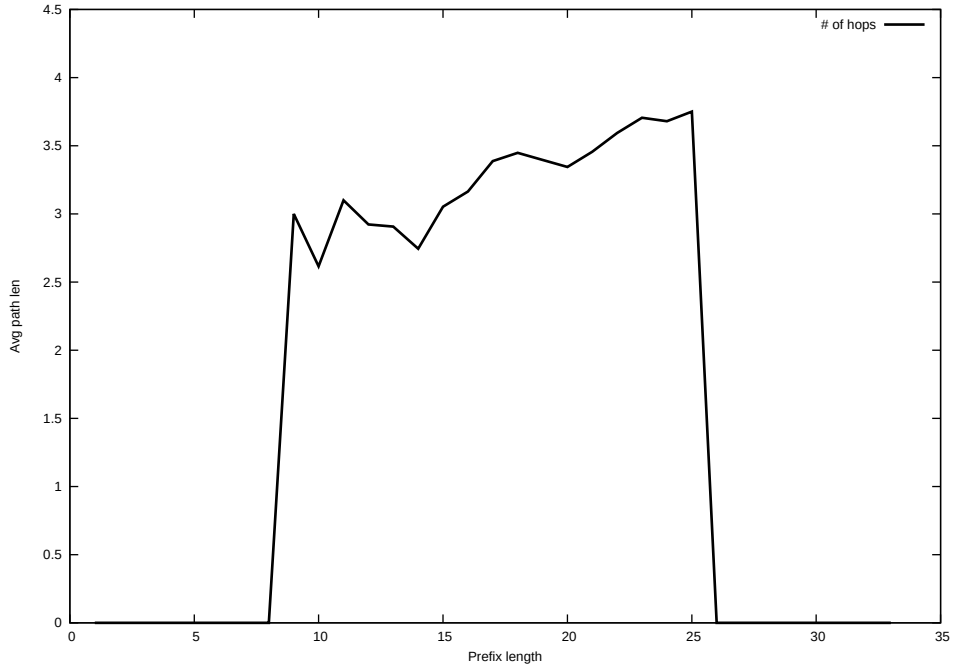
2014-07-04



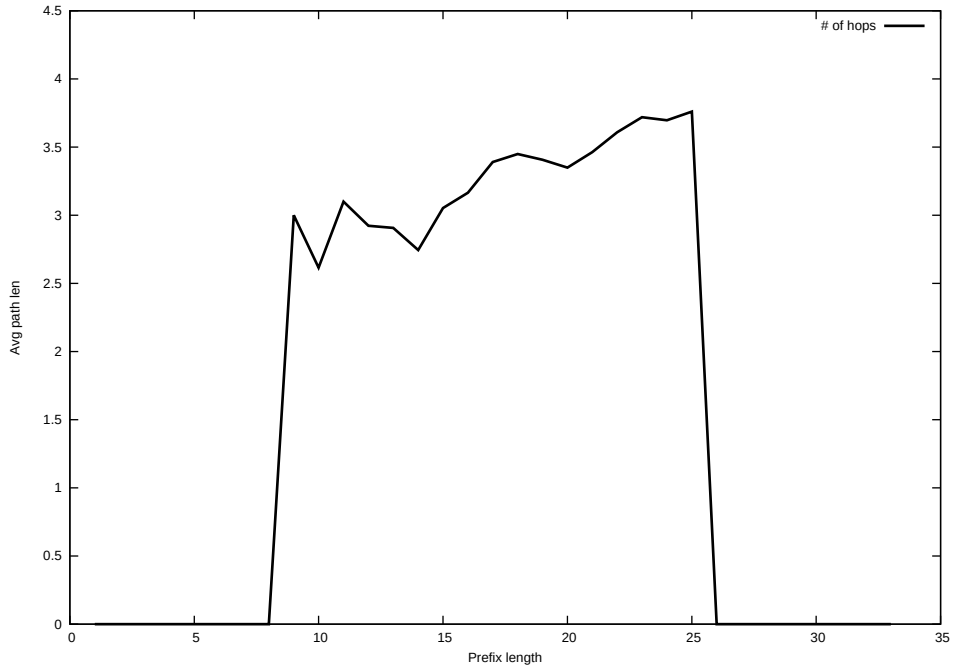
2014-07-05



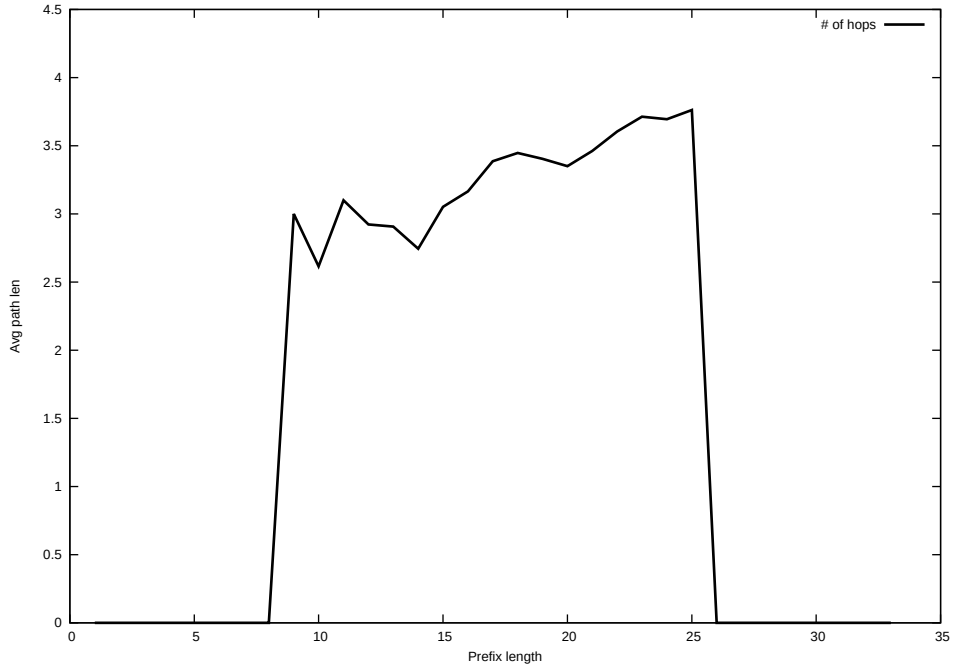
2014-07-06



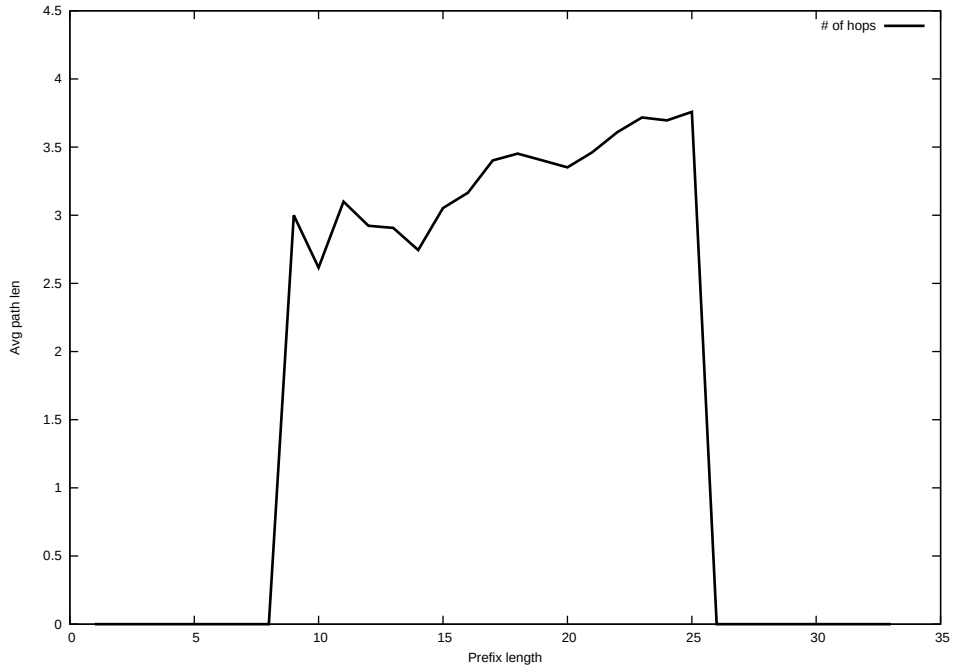
2014-07-07



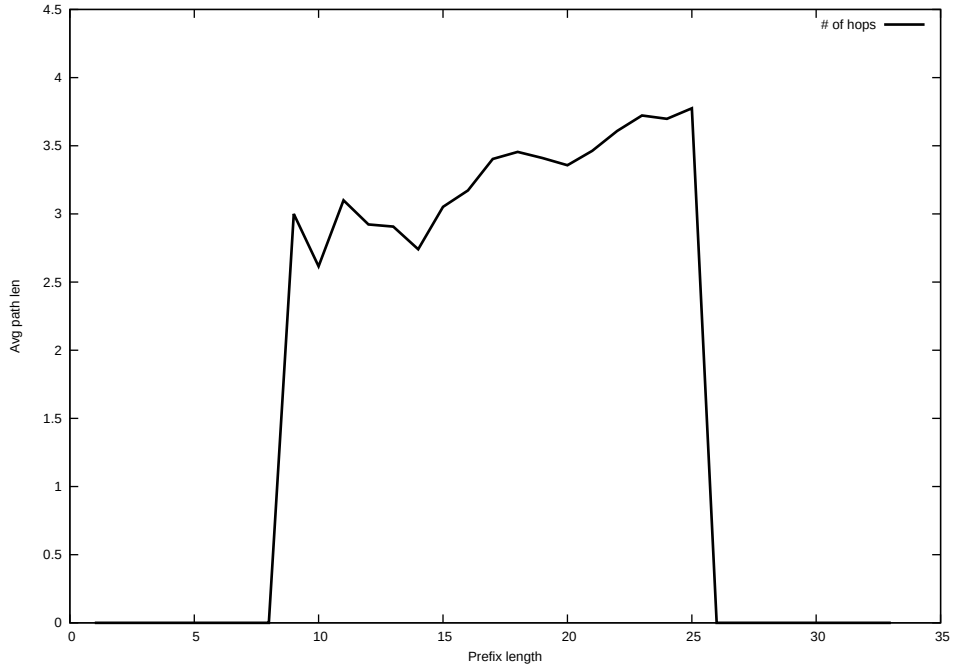
2014-07-08



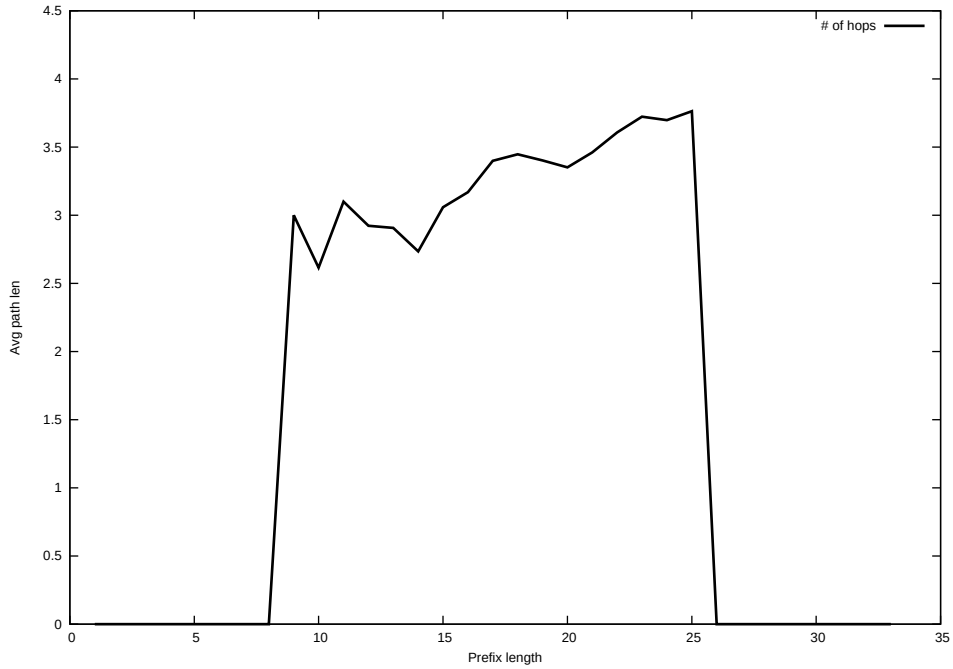
2014-07-09



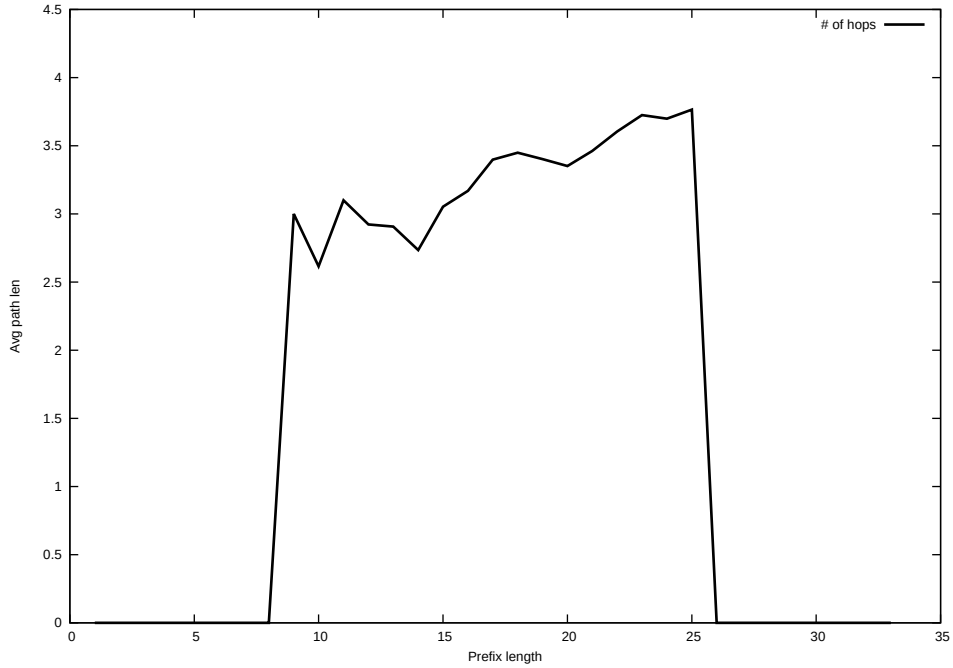
2014-07-10



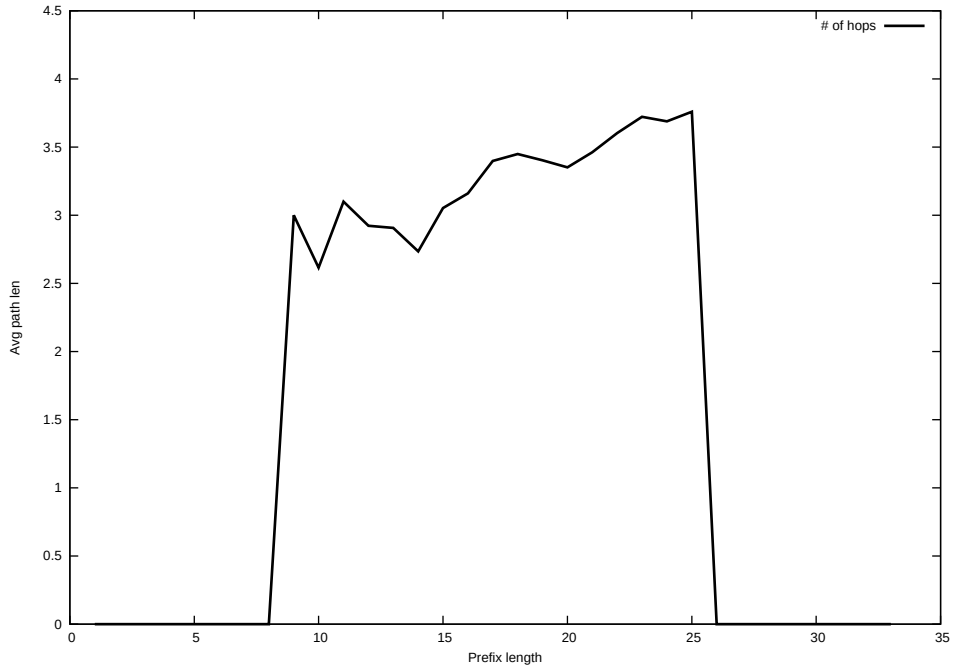
2014-07-11



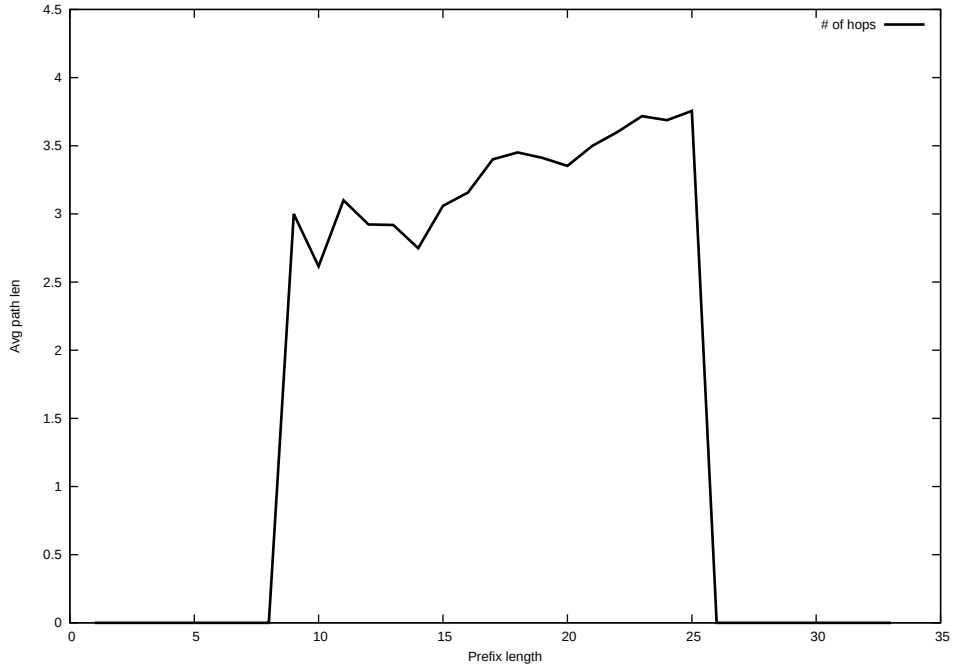
2014-07-12



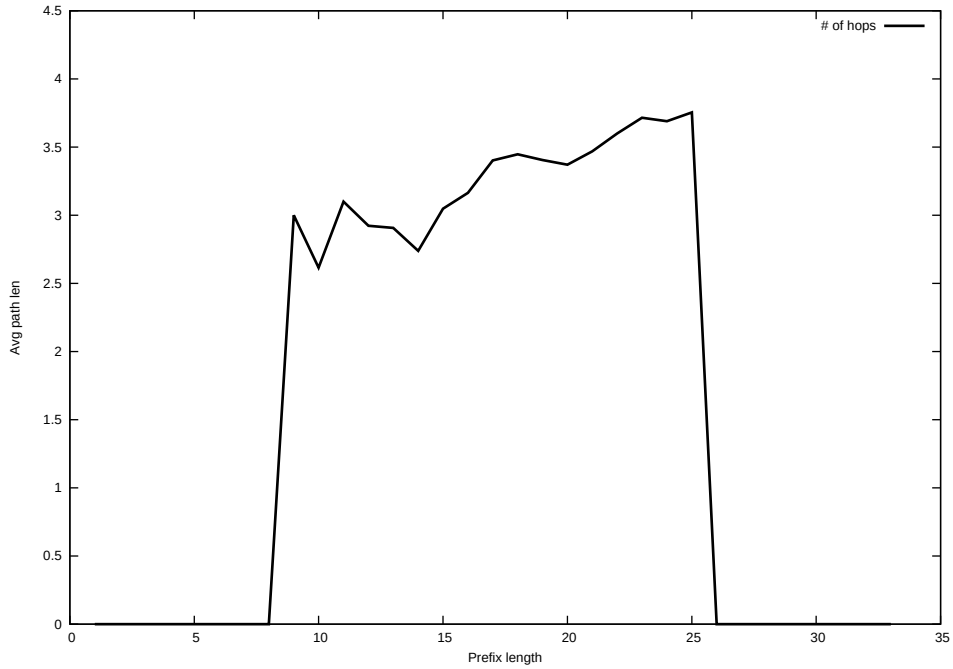
2014-07-13



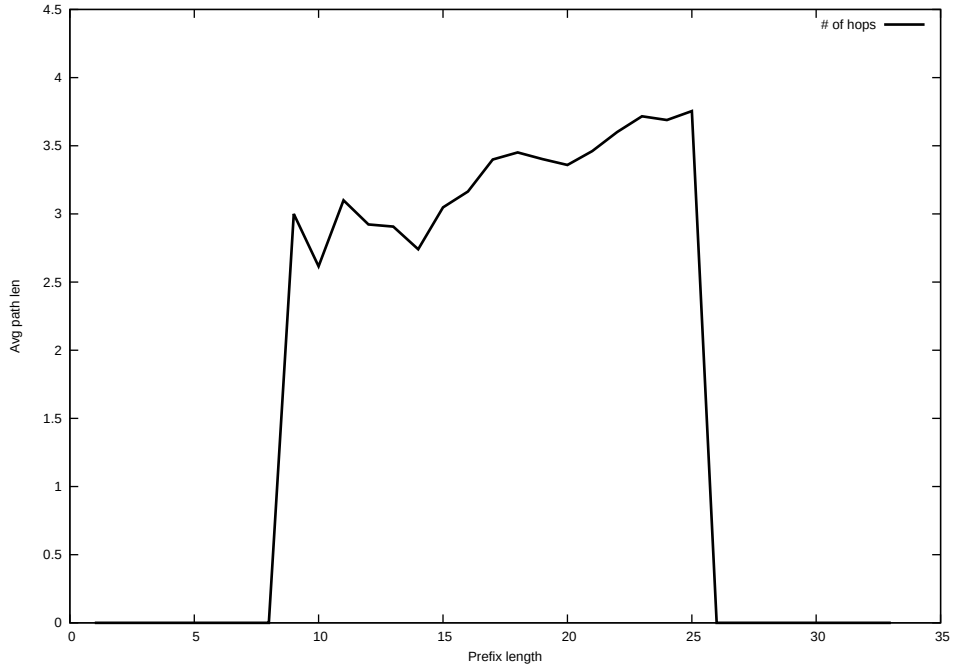
2014-07-14



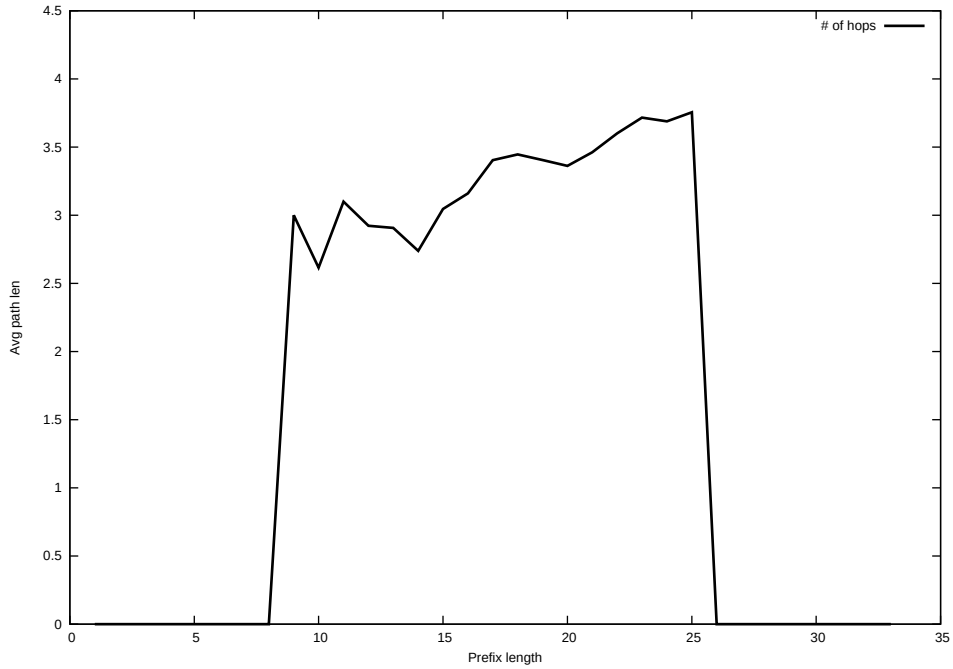
2014-07-15



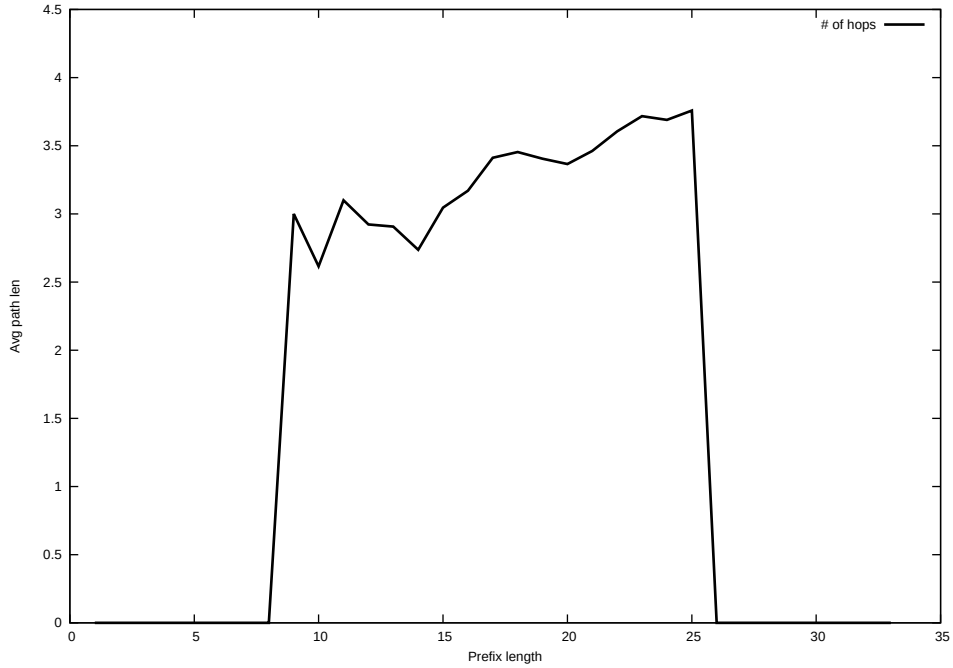
2014-07-16



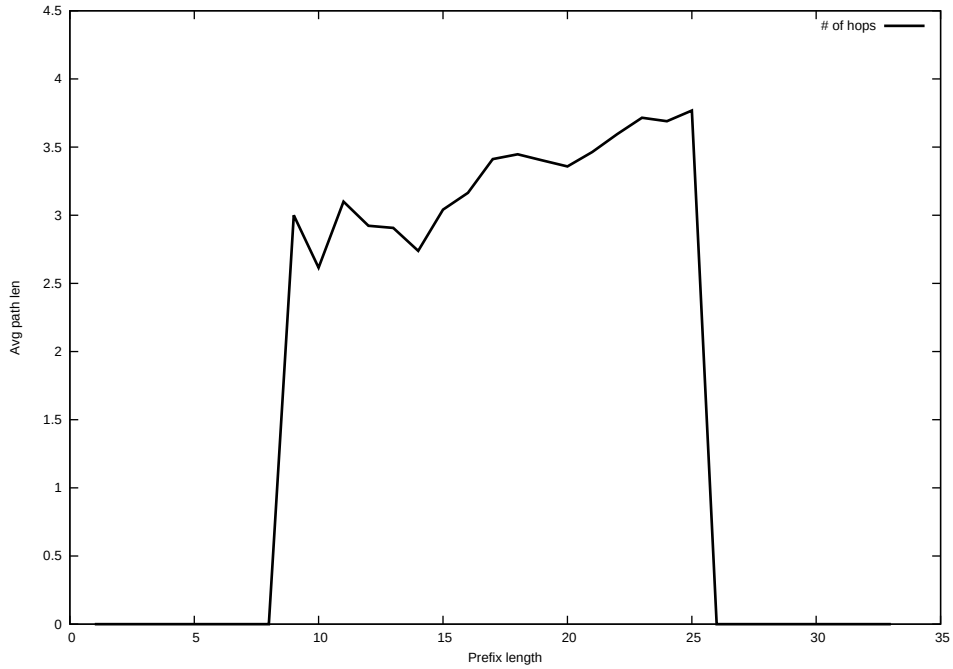
2014-07-17



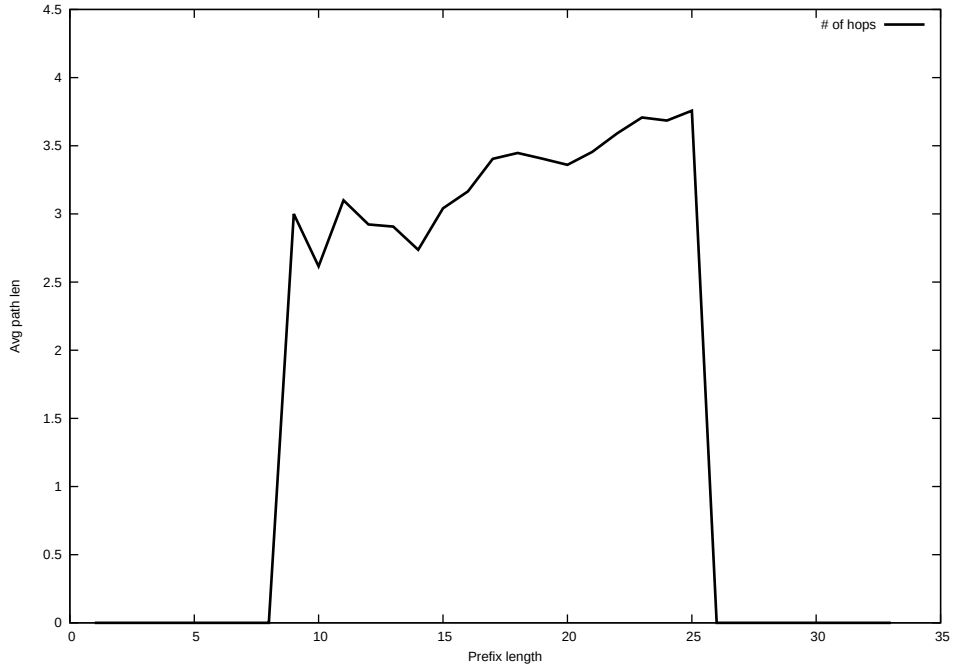
2014-07-18



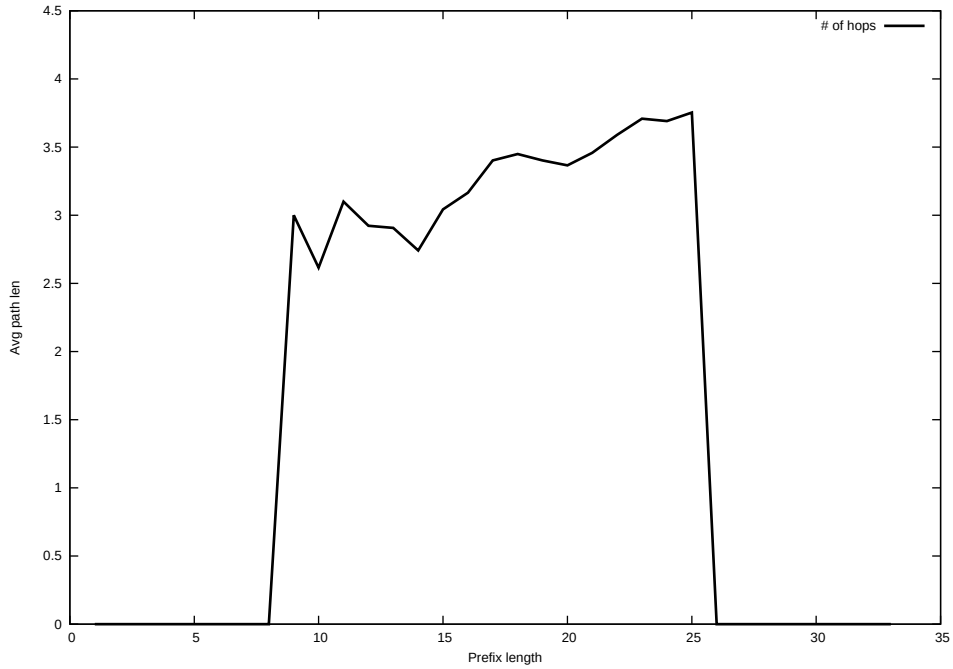
2014-07-19



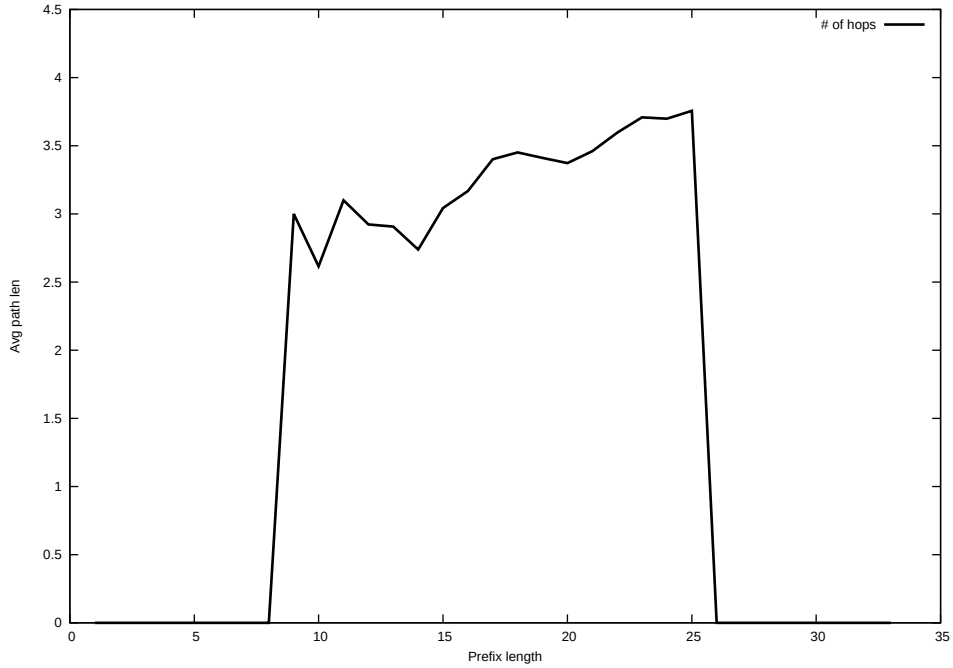
2014-07-20



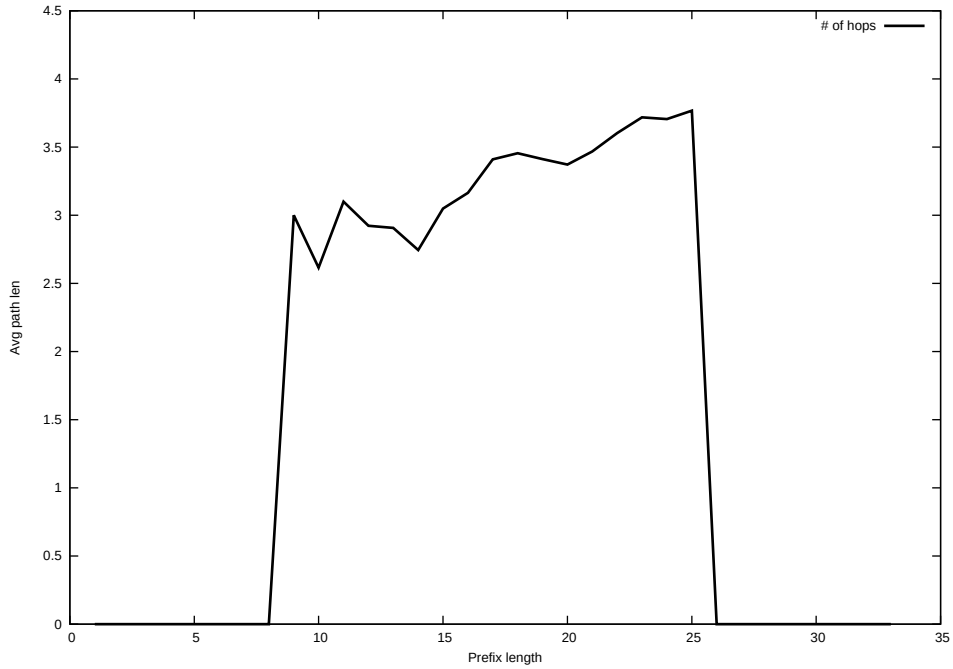
2014-07-21



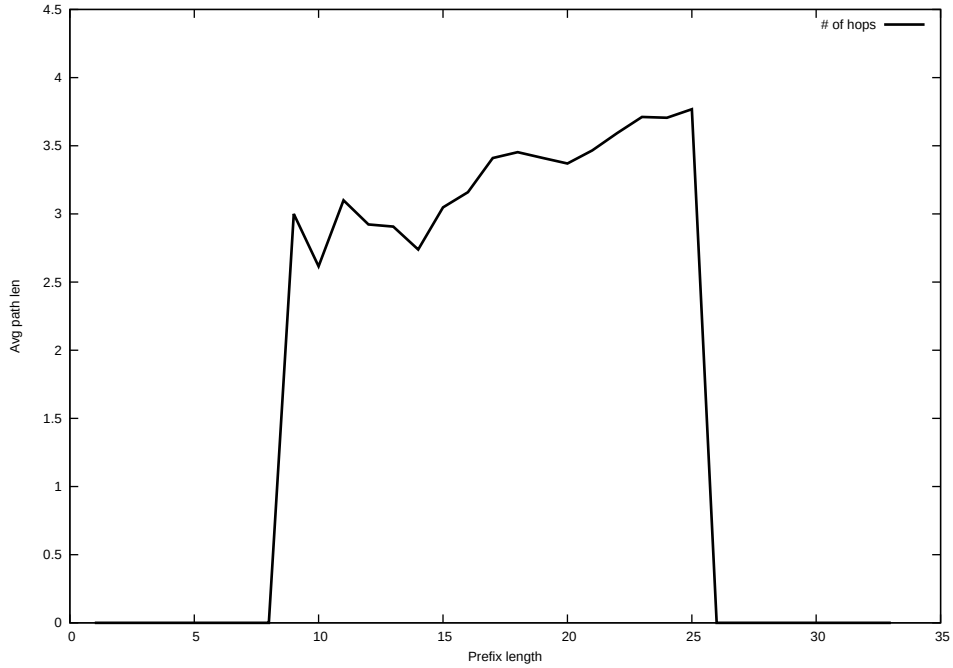
2014-07-22



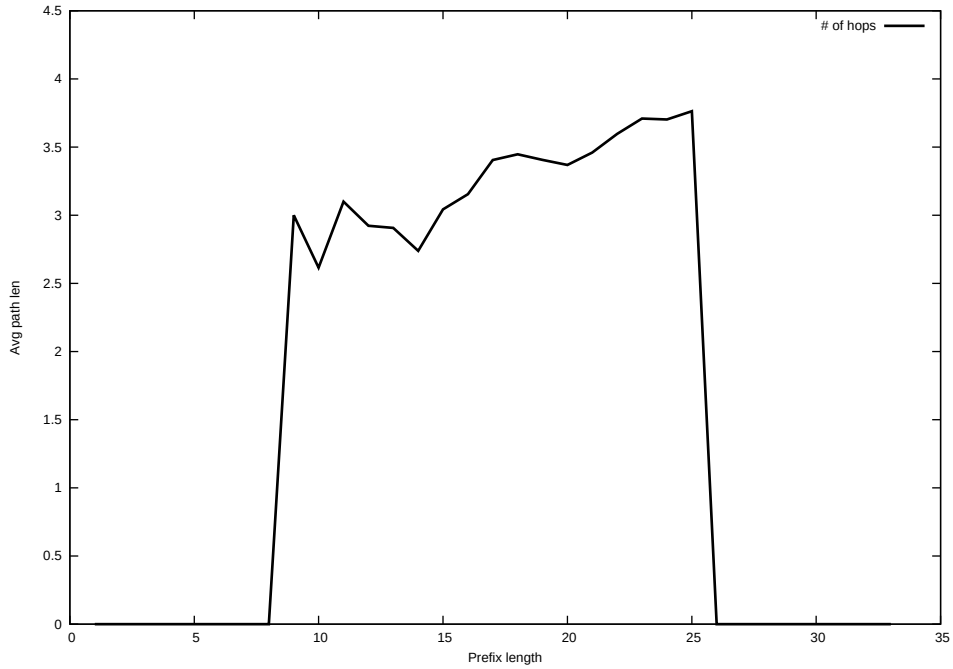
2014-07-23



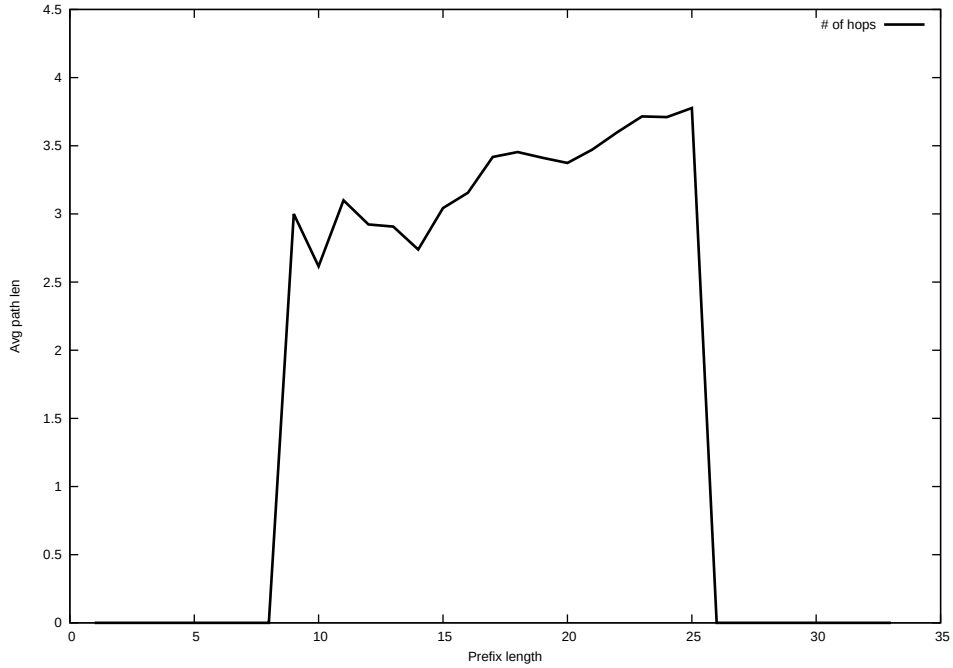
2014-07-24



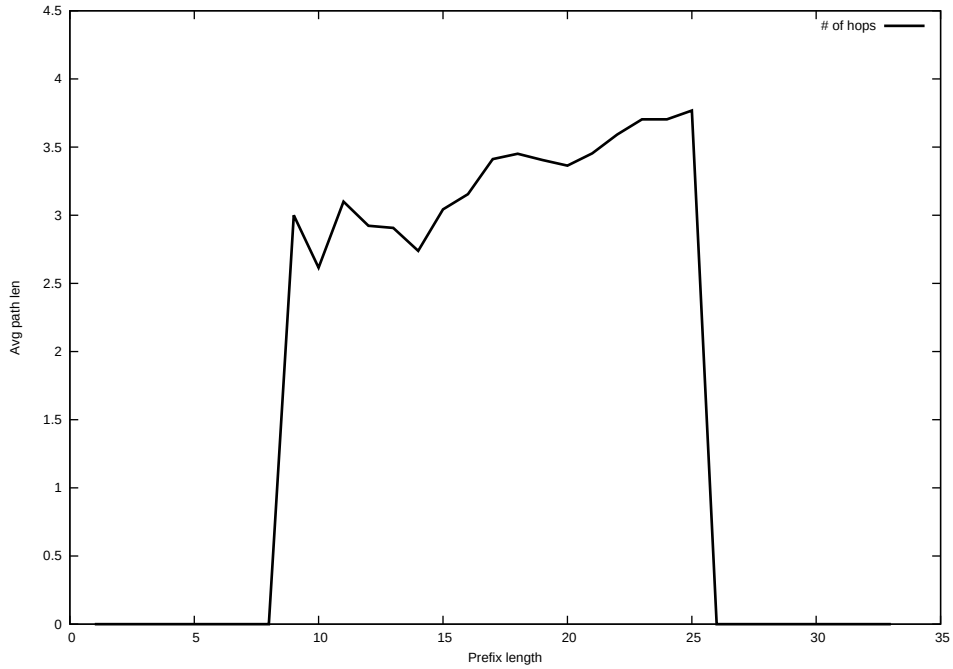
2014-07-25



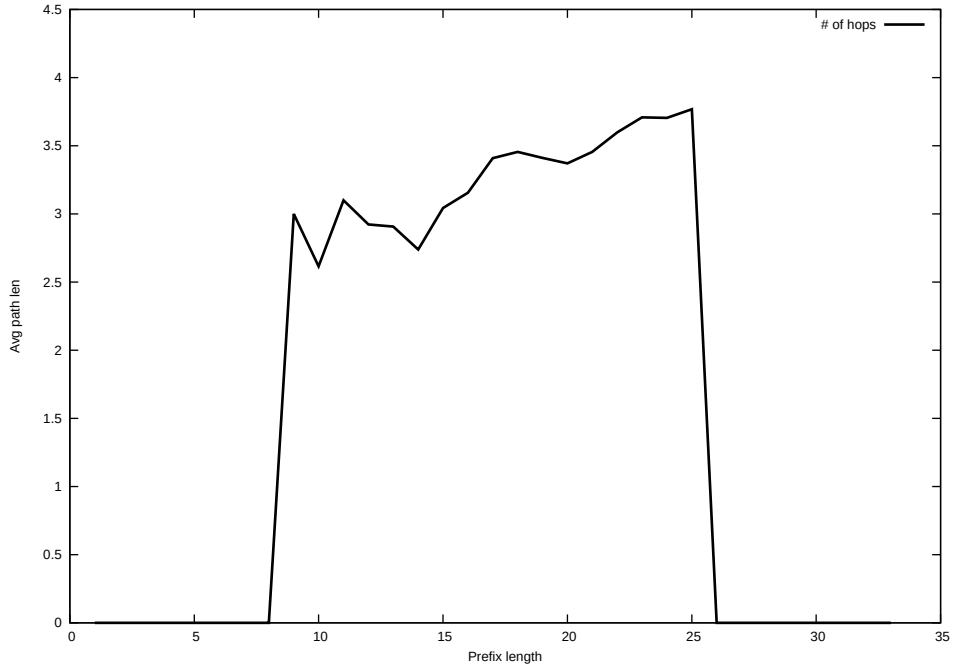
2014-07-26



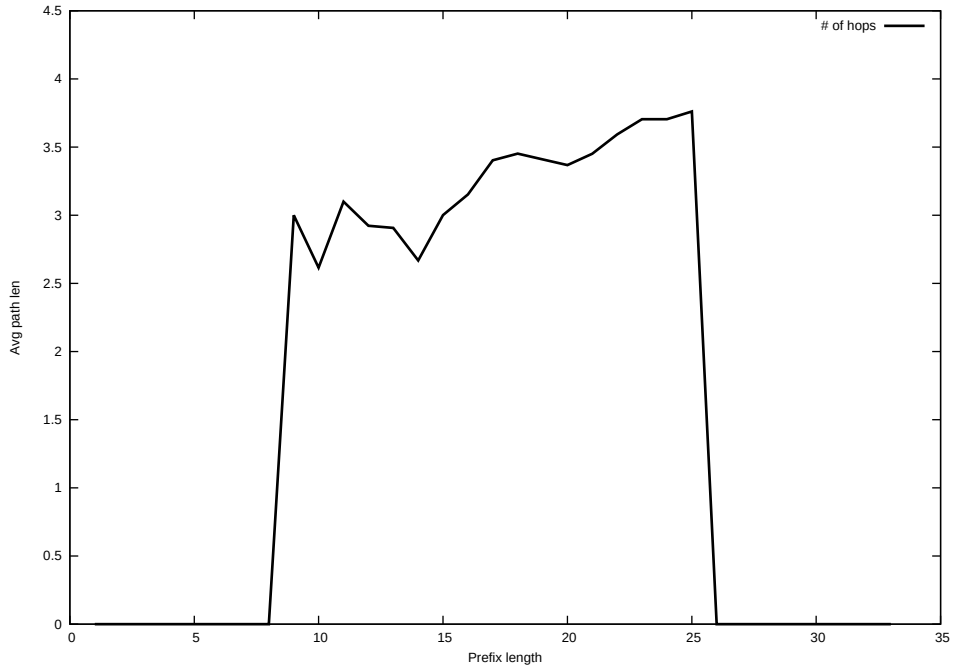
2014-07-27



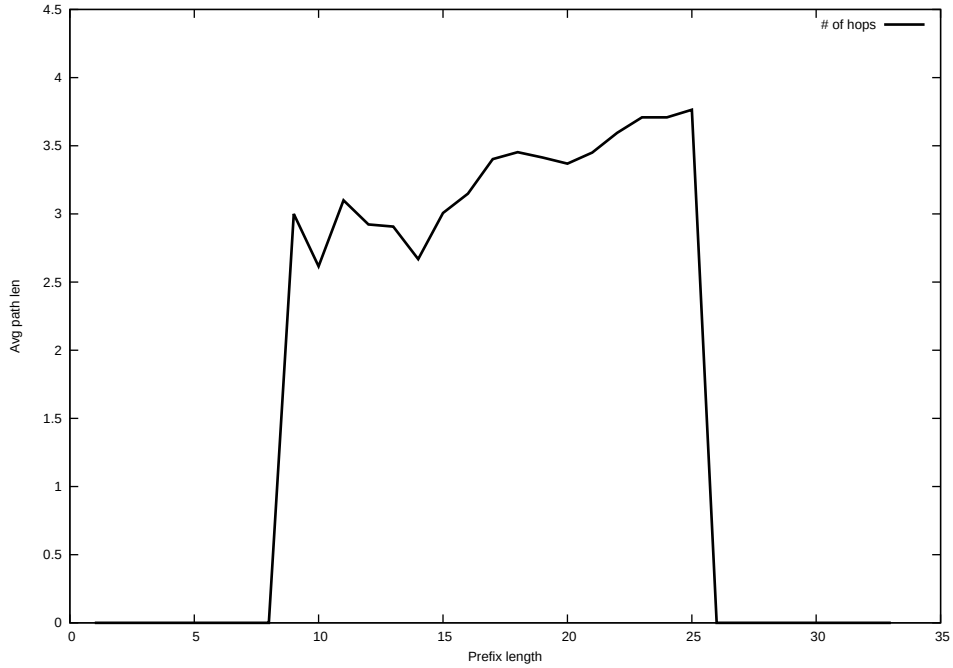
2014-07-28



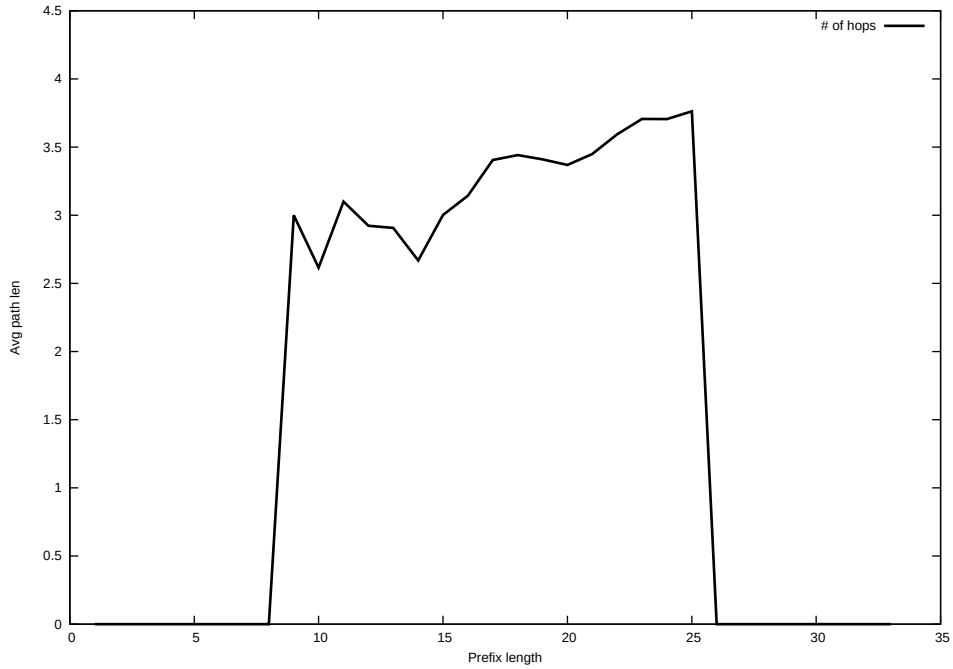
2014-07-29



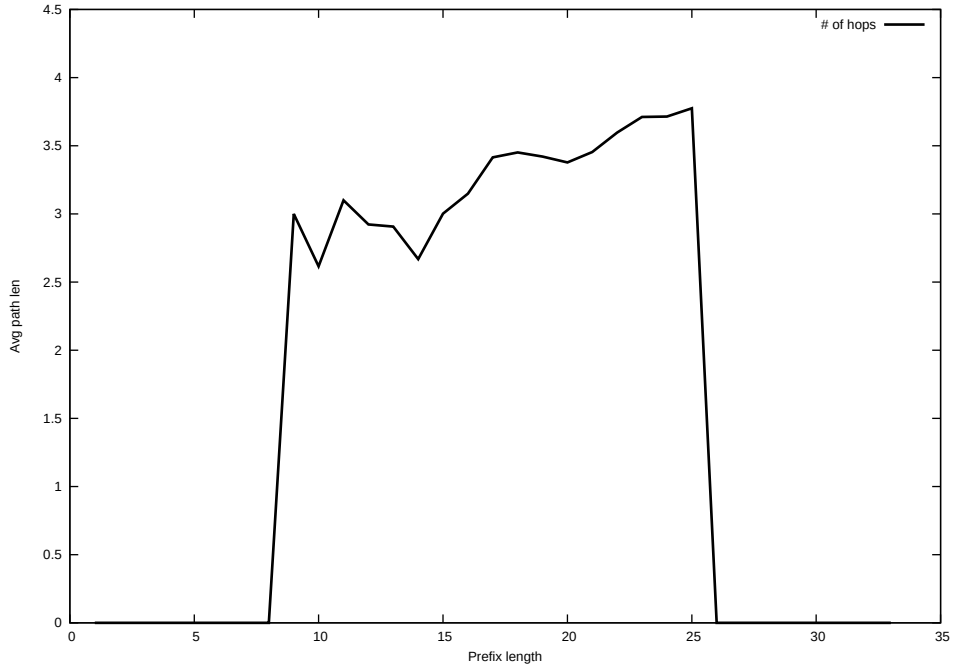
2014-07-30



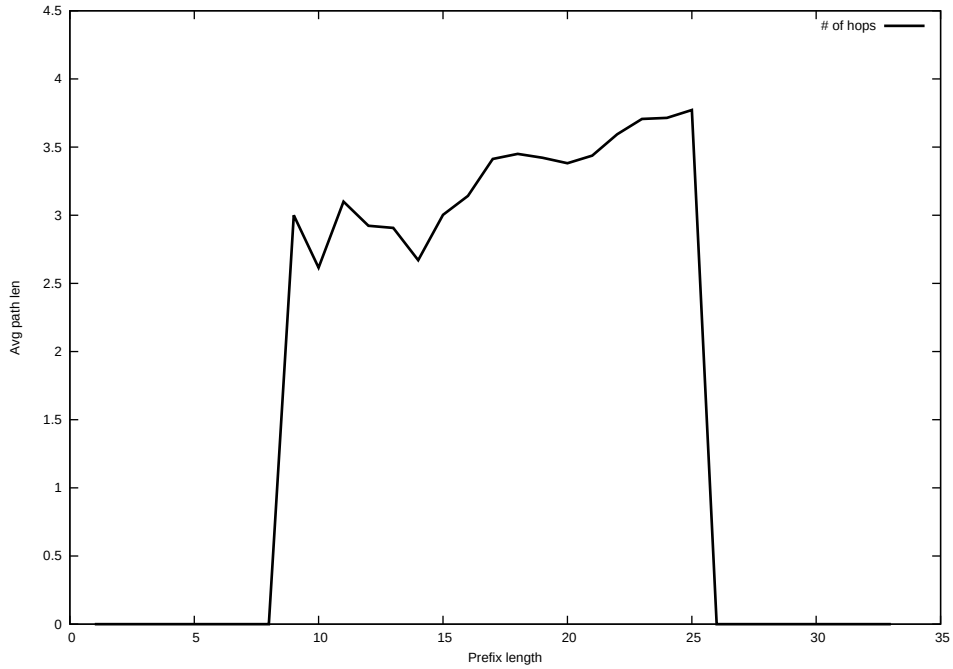
2014-07-31



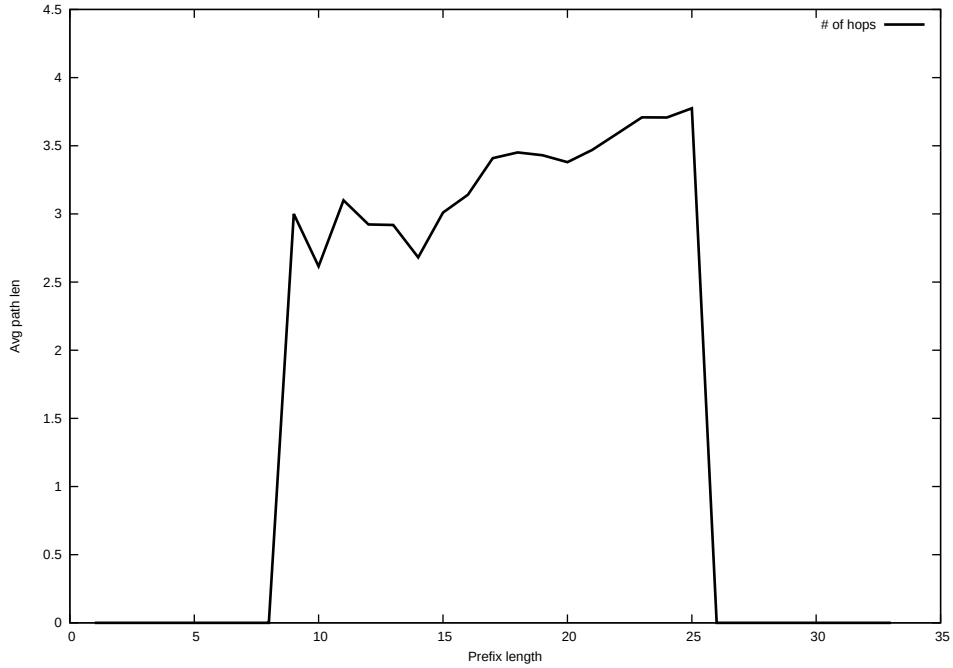
2014-08-01



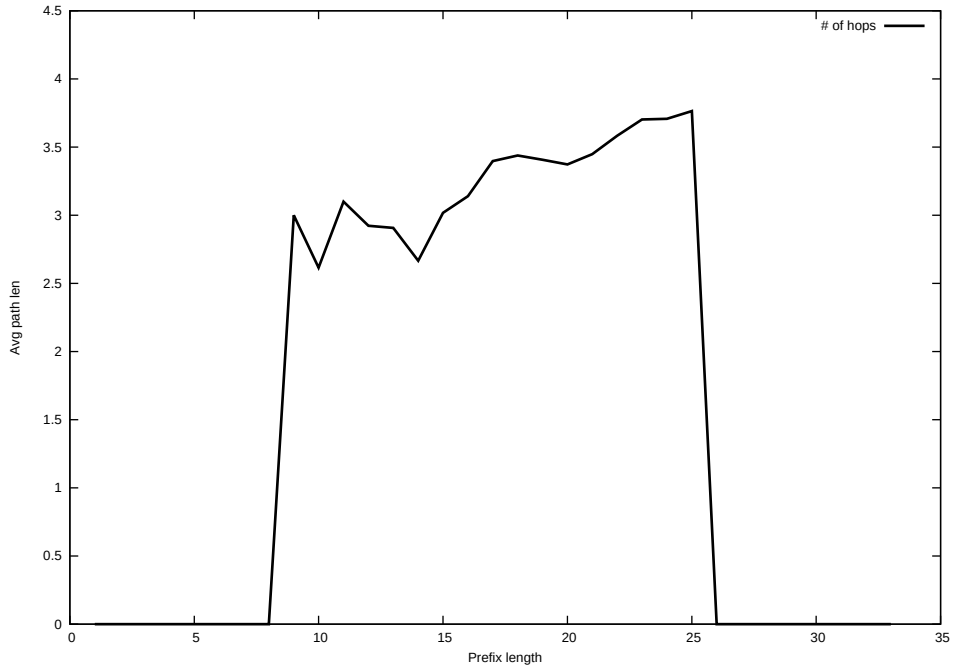
2014-08-02



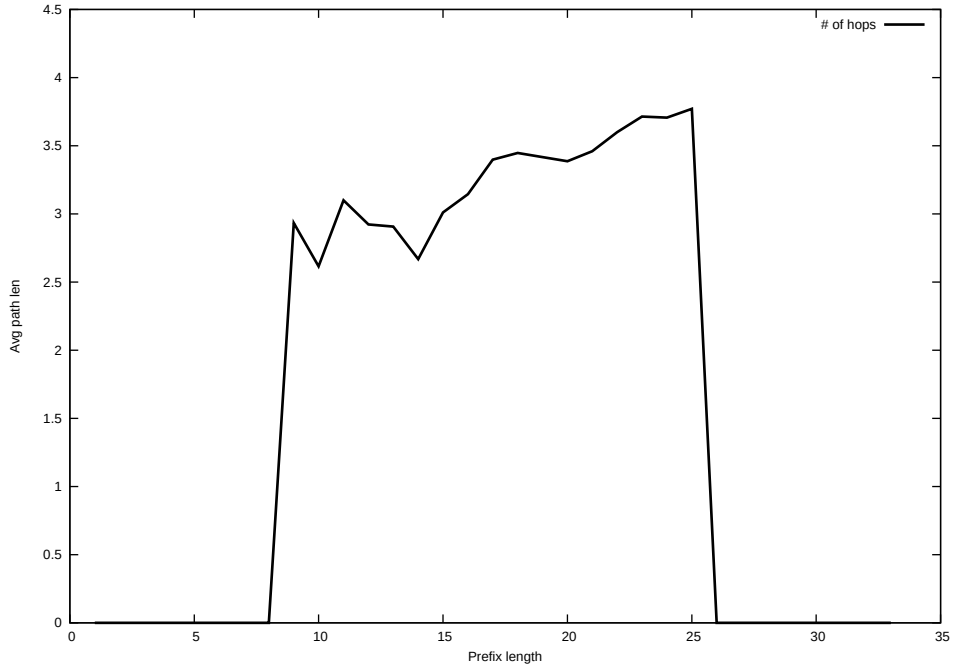
2014-08-03



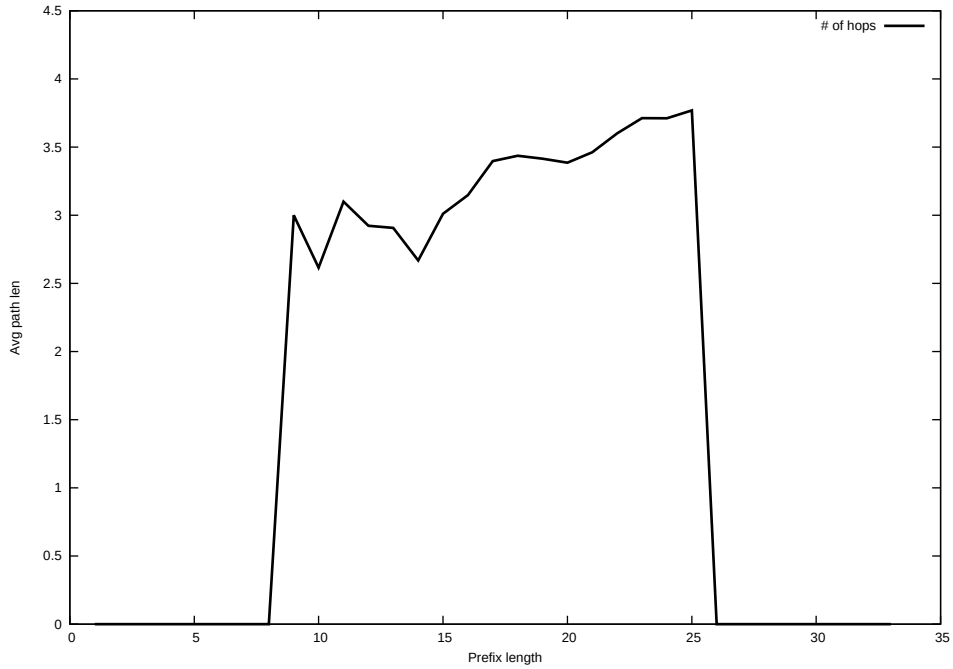
2014-08-04



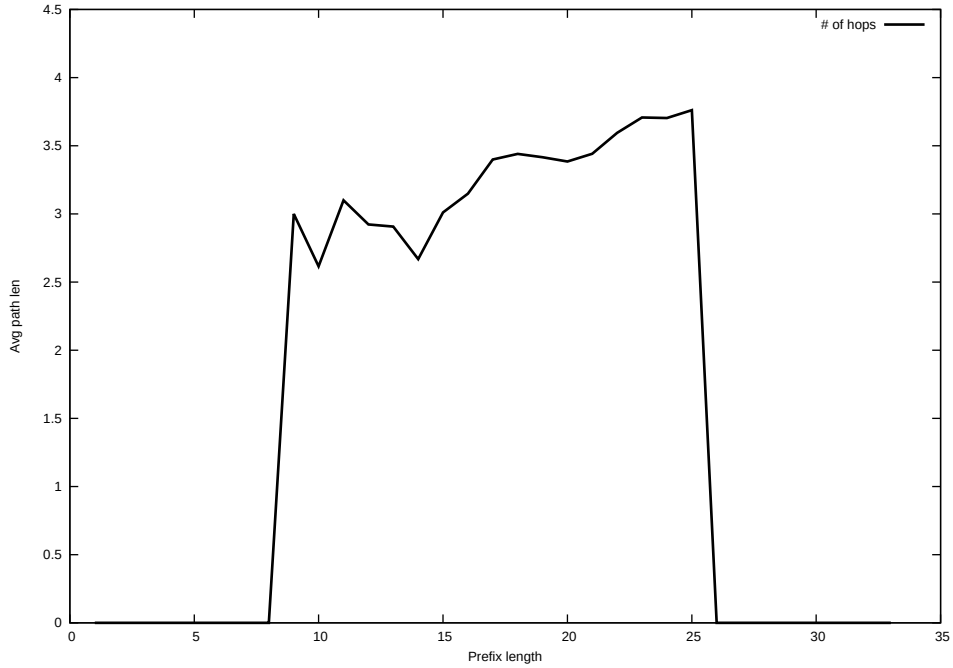
2014-08-05



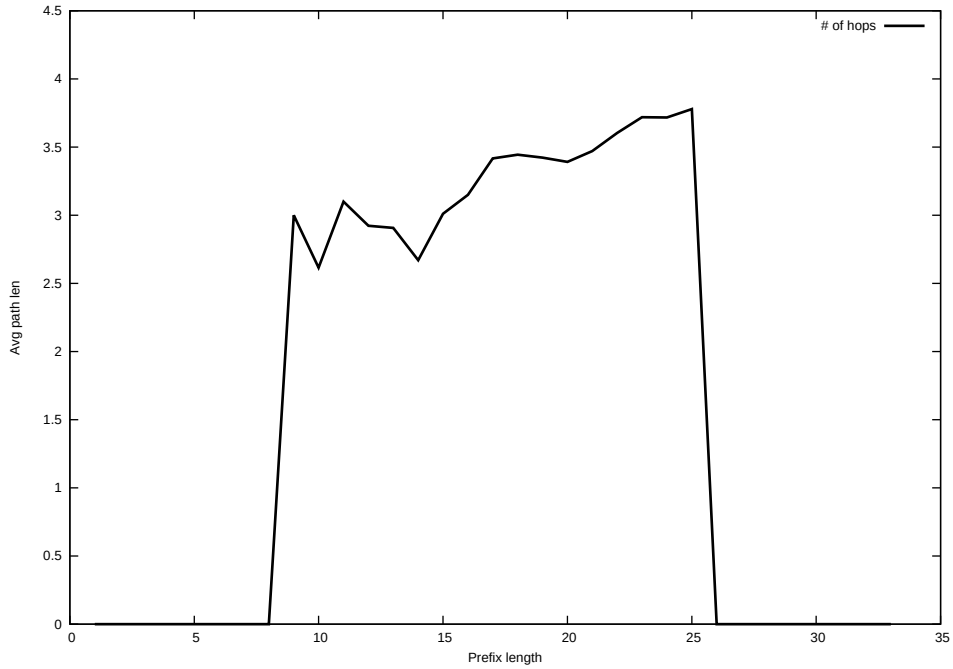
2014-08-06



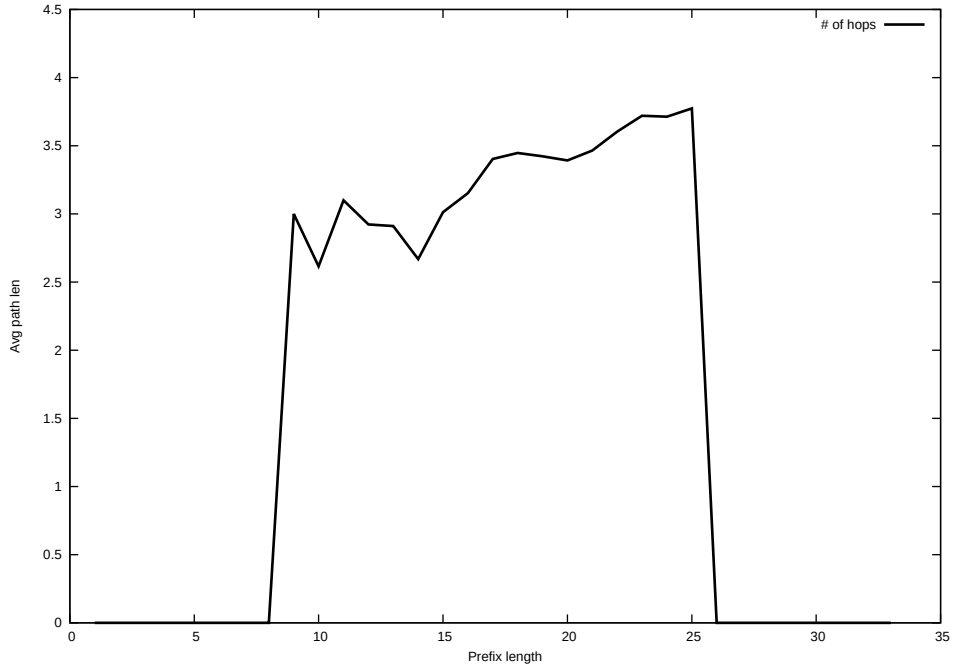
2014-08-07



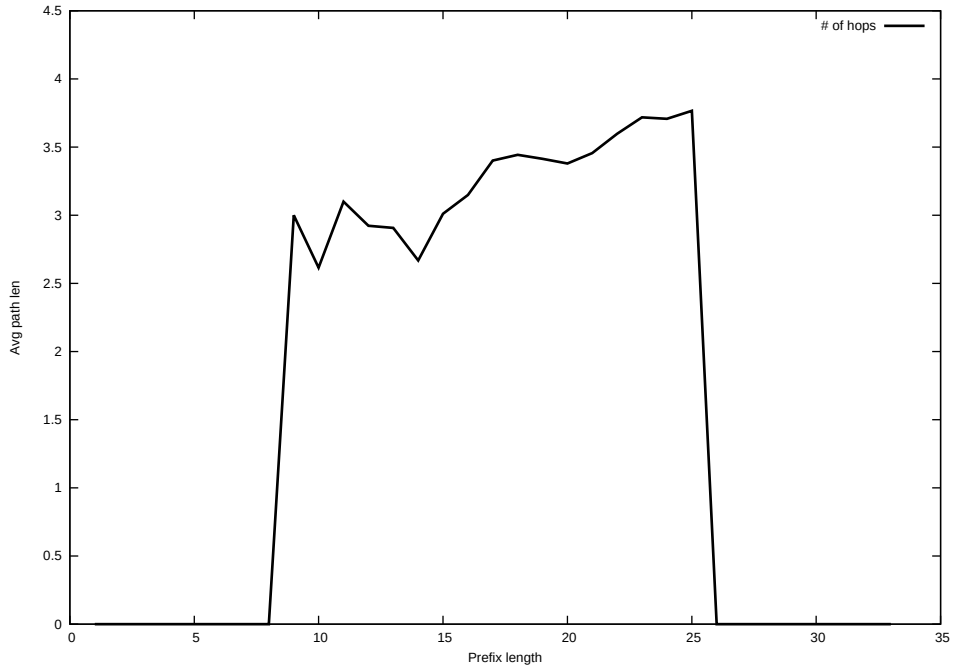
2014-08-08



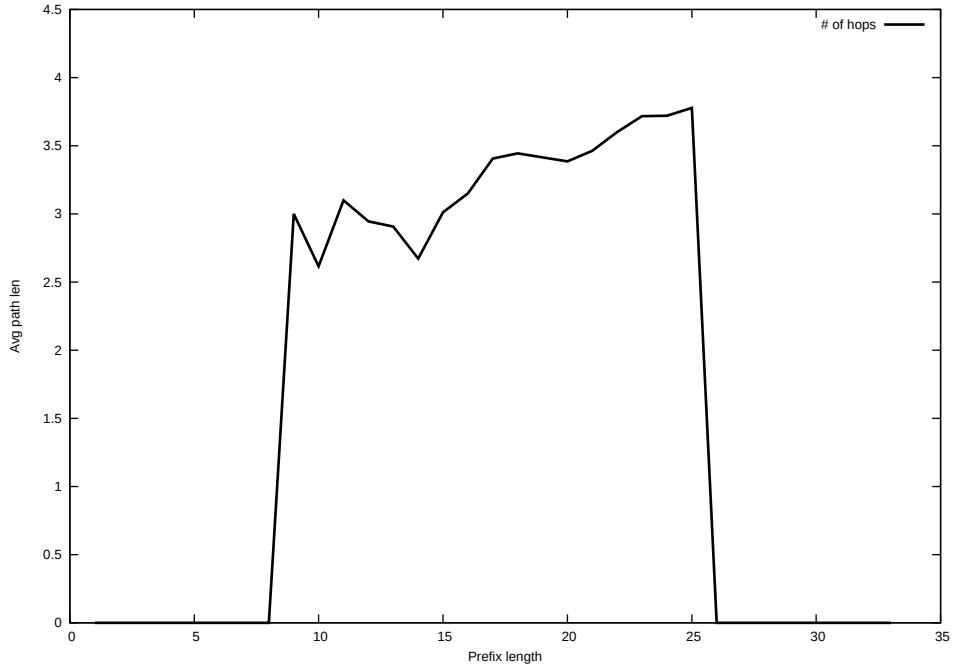
2014-08-09



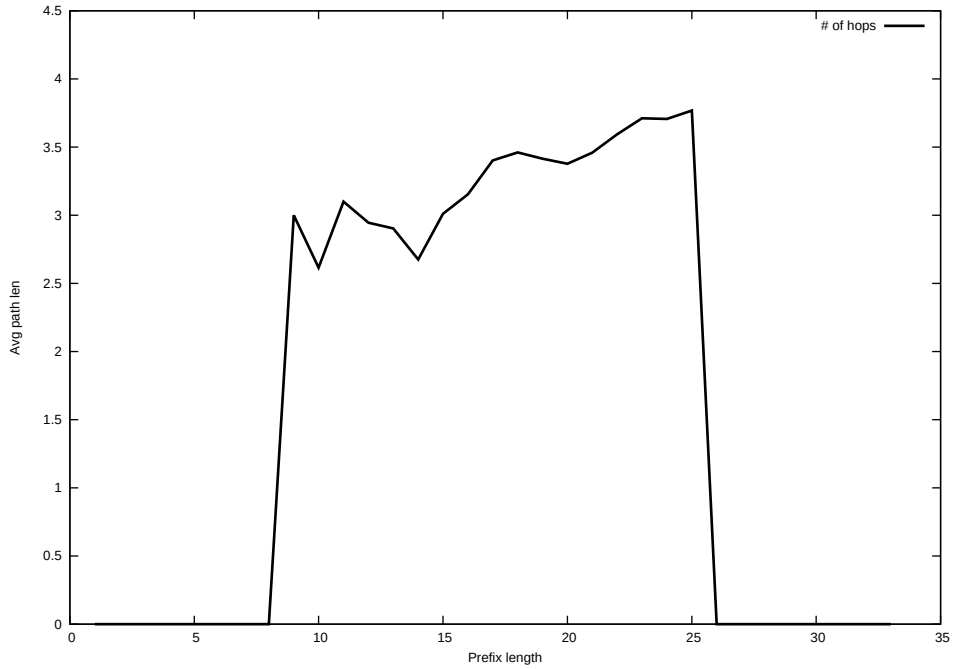
2014-08-10



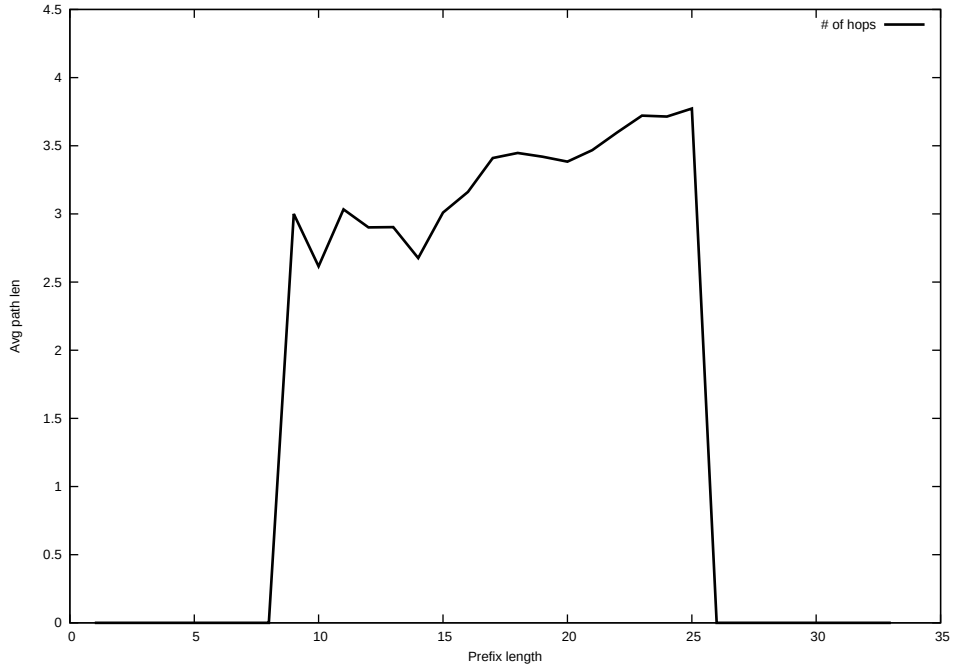
2014-08-11



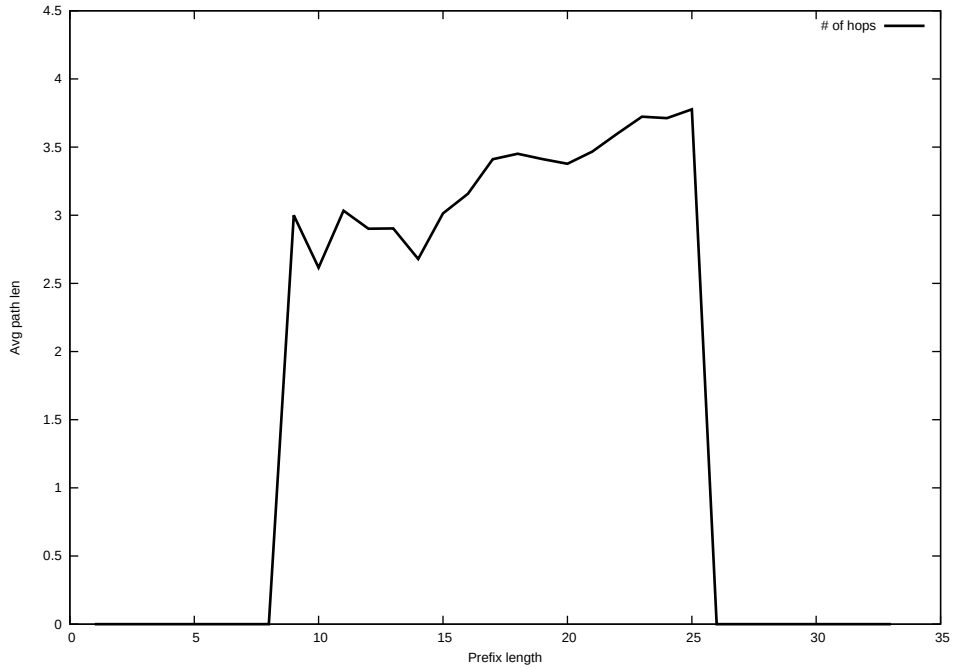
2014-08-12



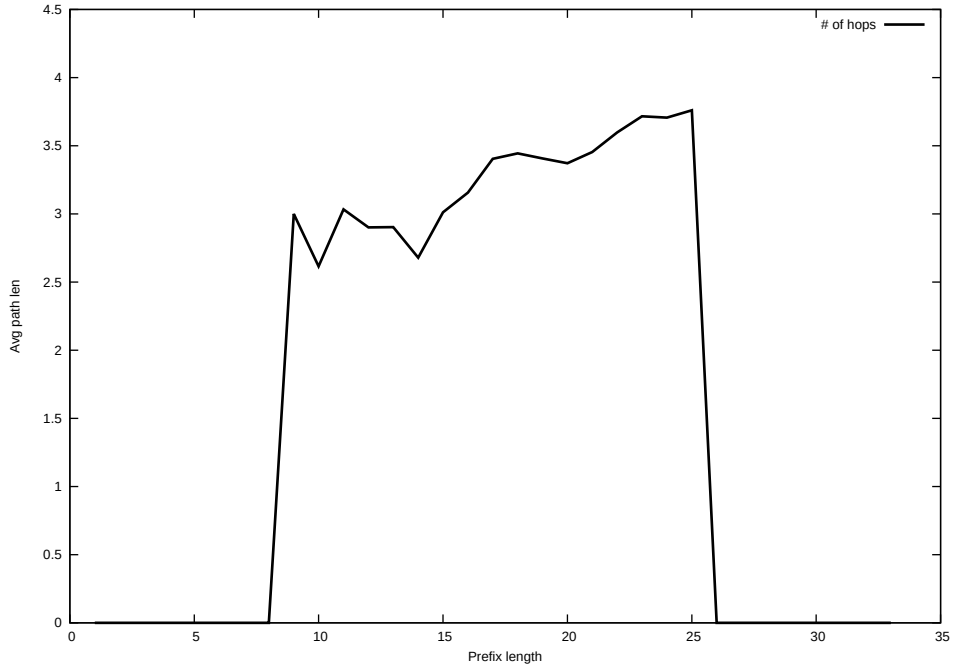
2014-08-13



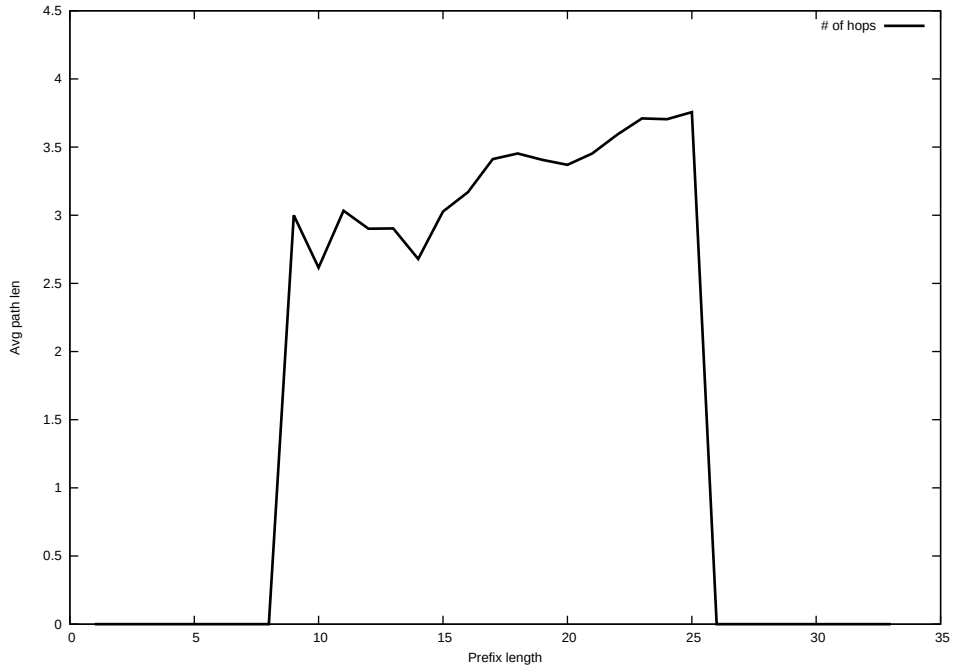
2014-08-14



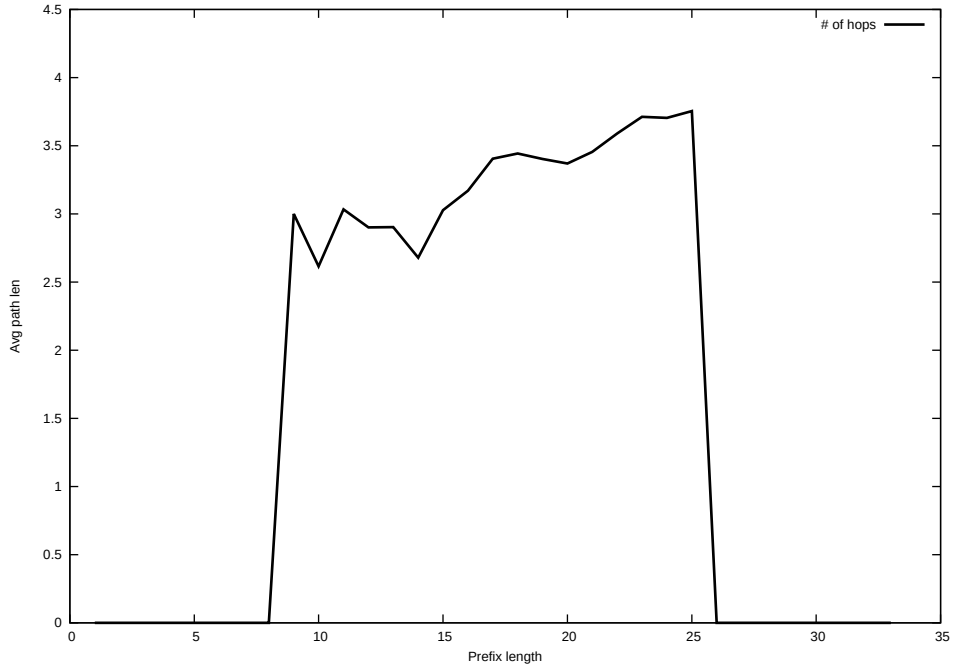
2014-08-15



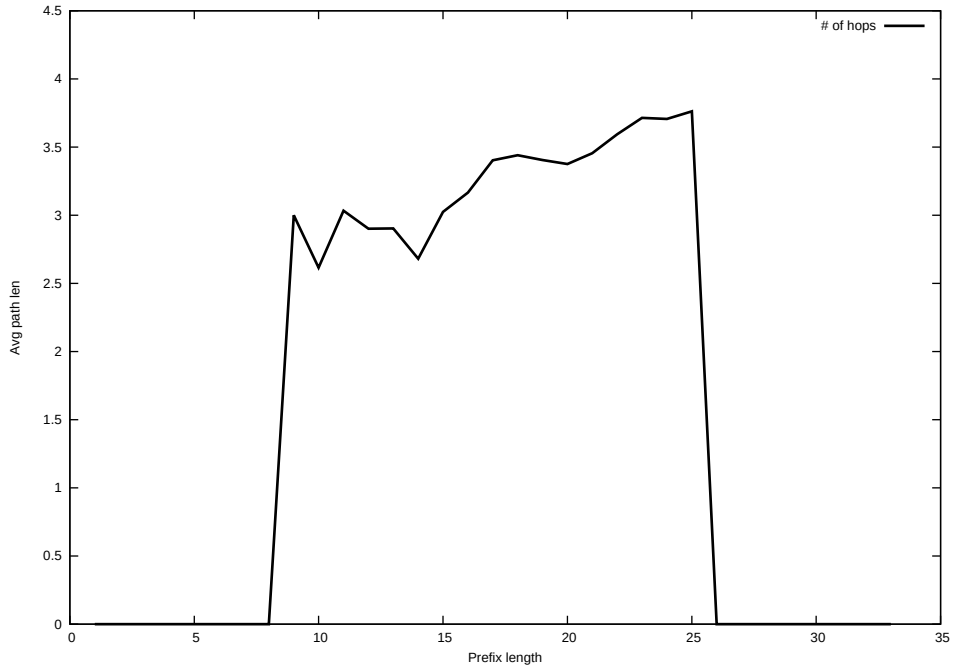
2014-08-16



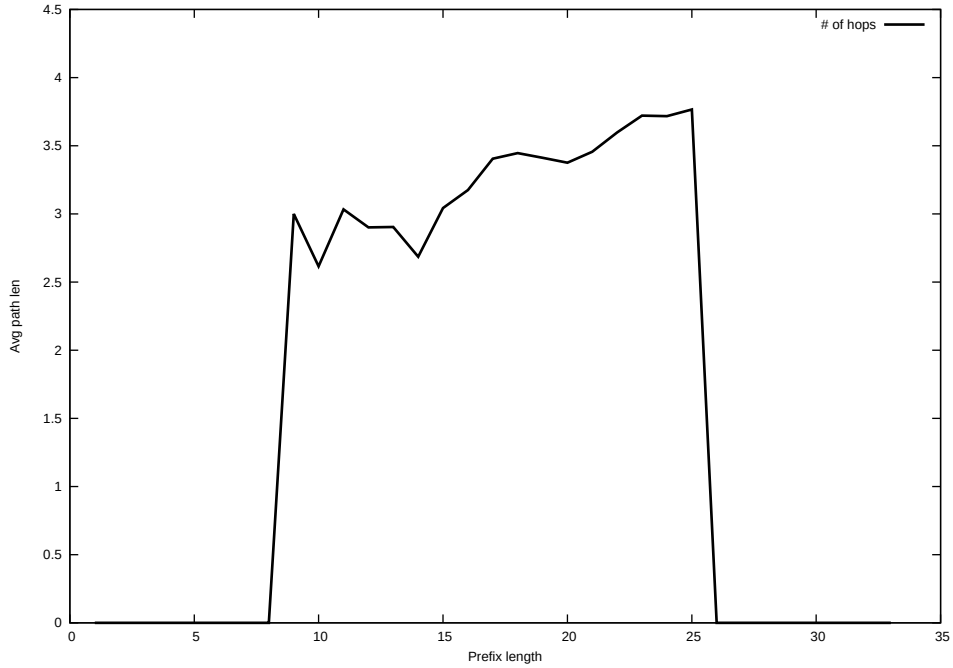
2014-08-17



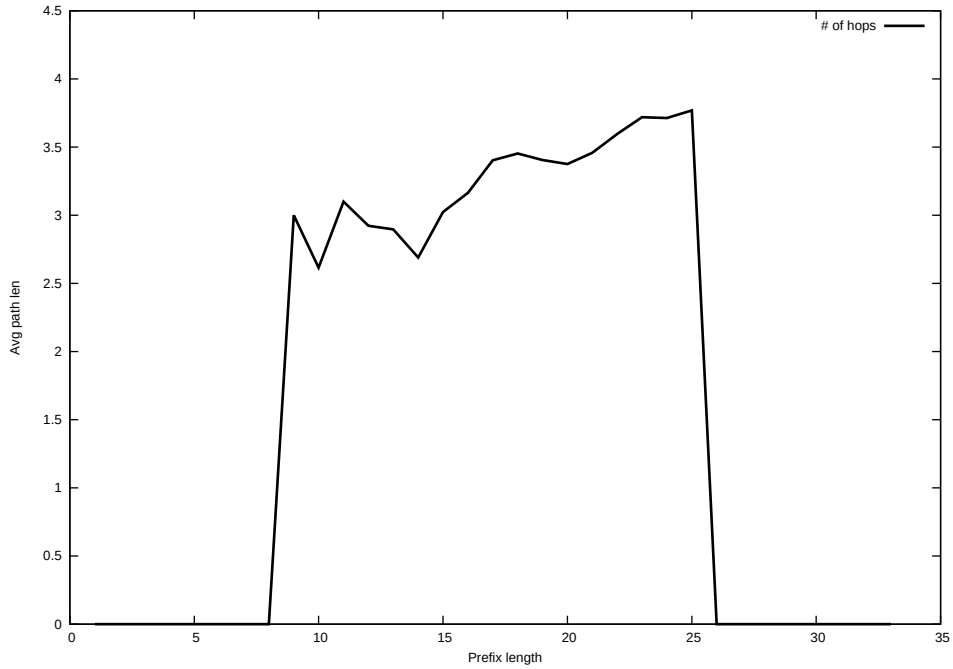
2014-08-18



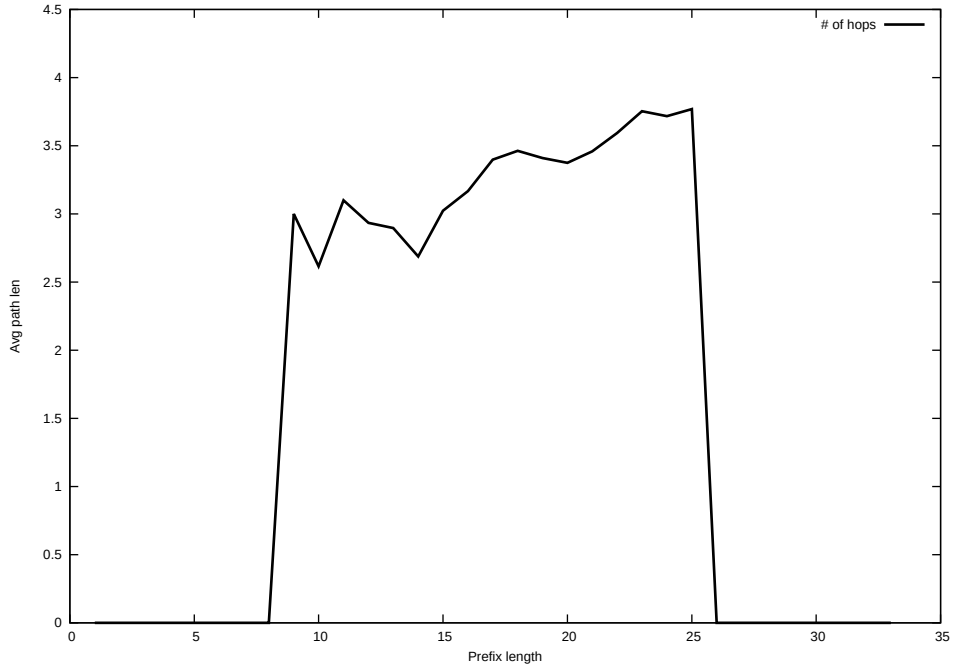
2014-08-19



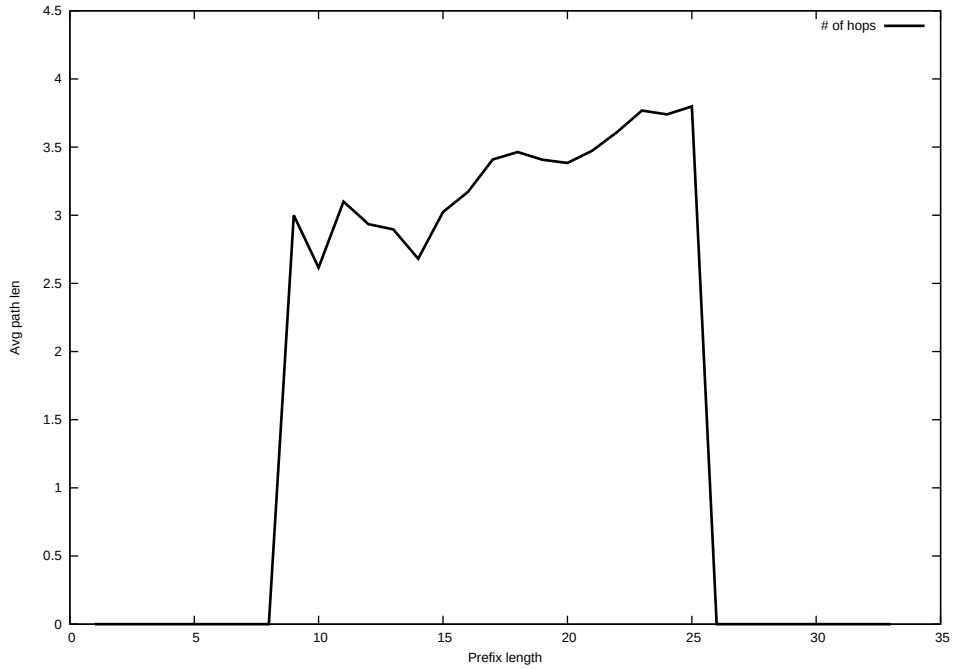
2014-08-20



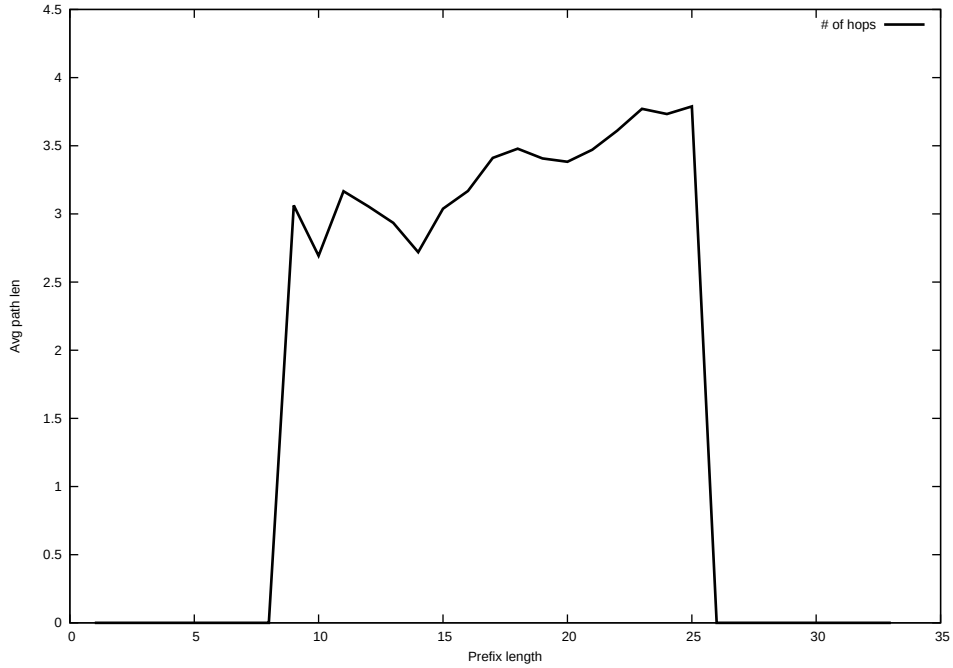
2014-08-21



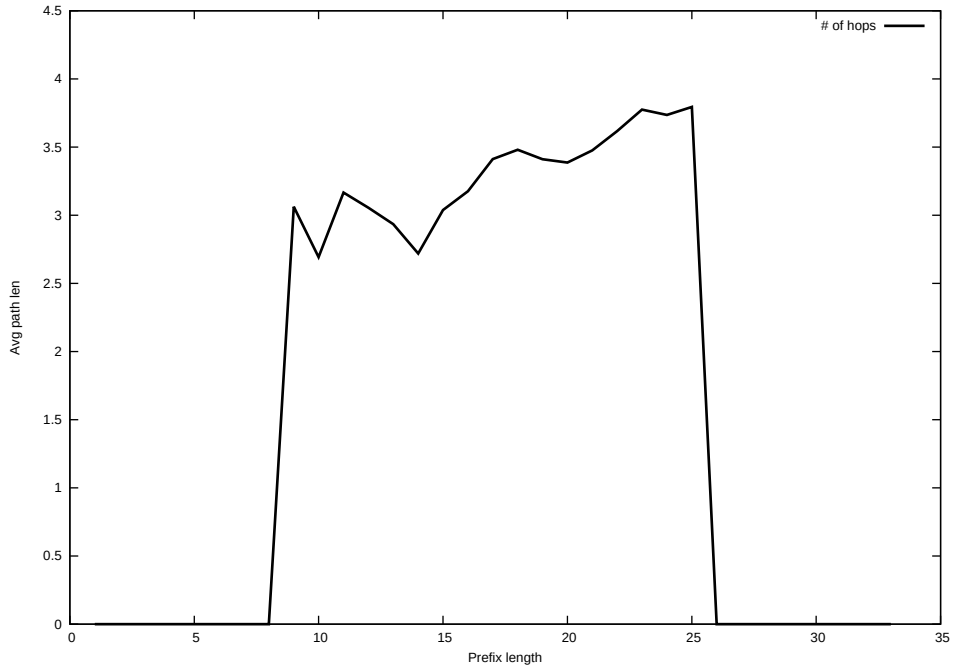
2014-08-22



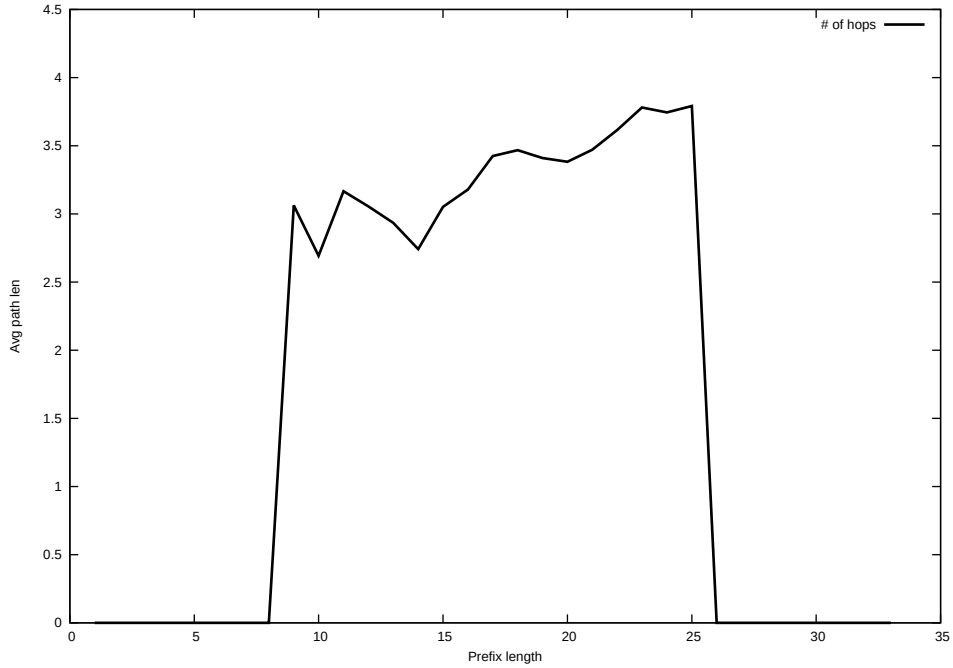
2014-08-23



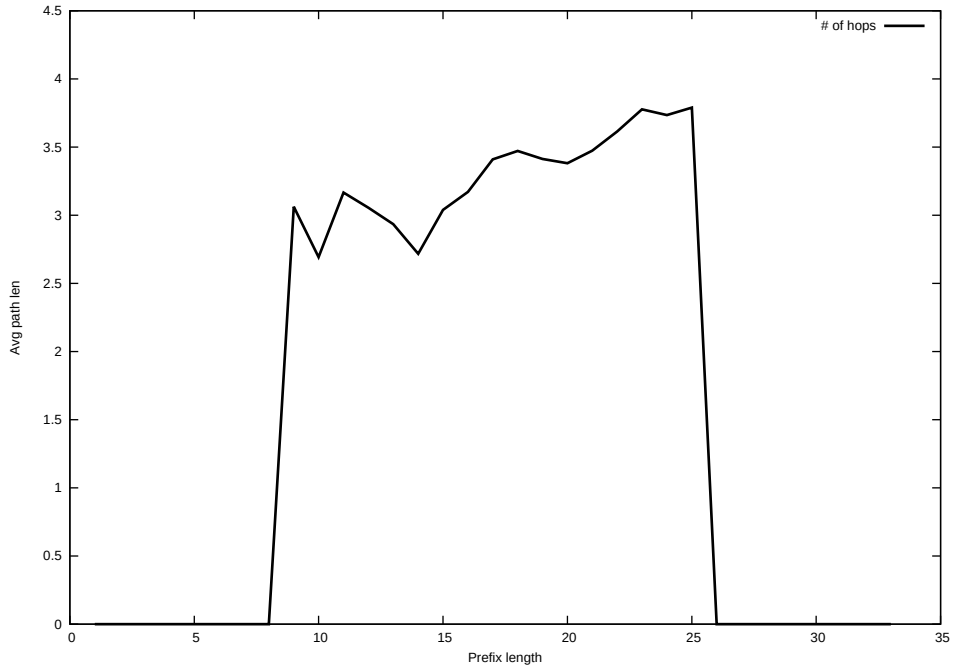
2014-08-24



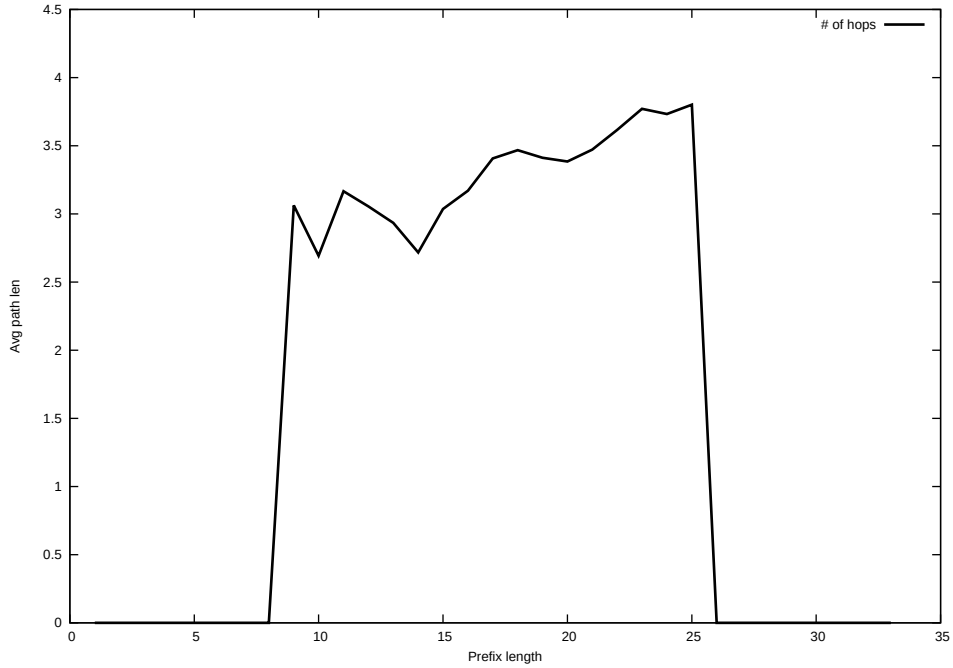
2014-08-25



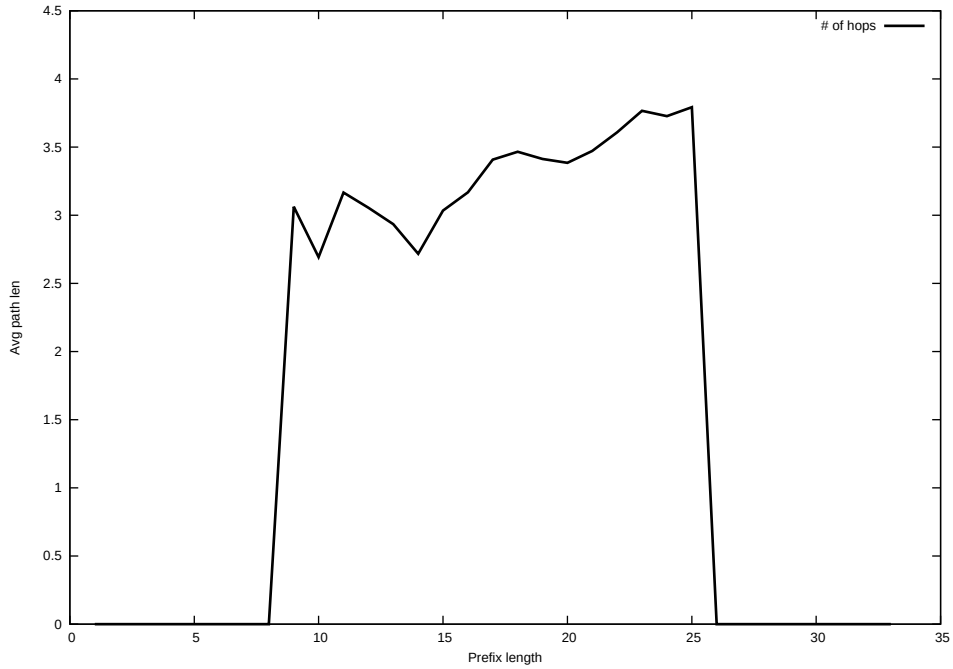
2014-08-26



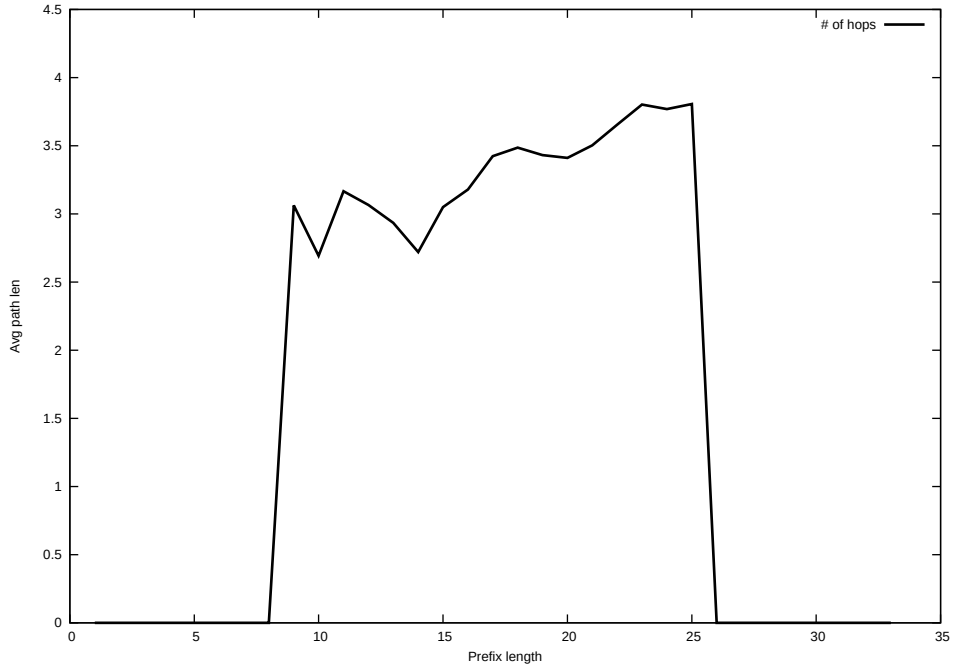
2014-08-27



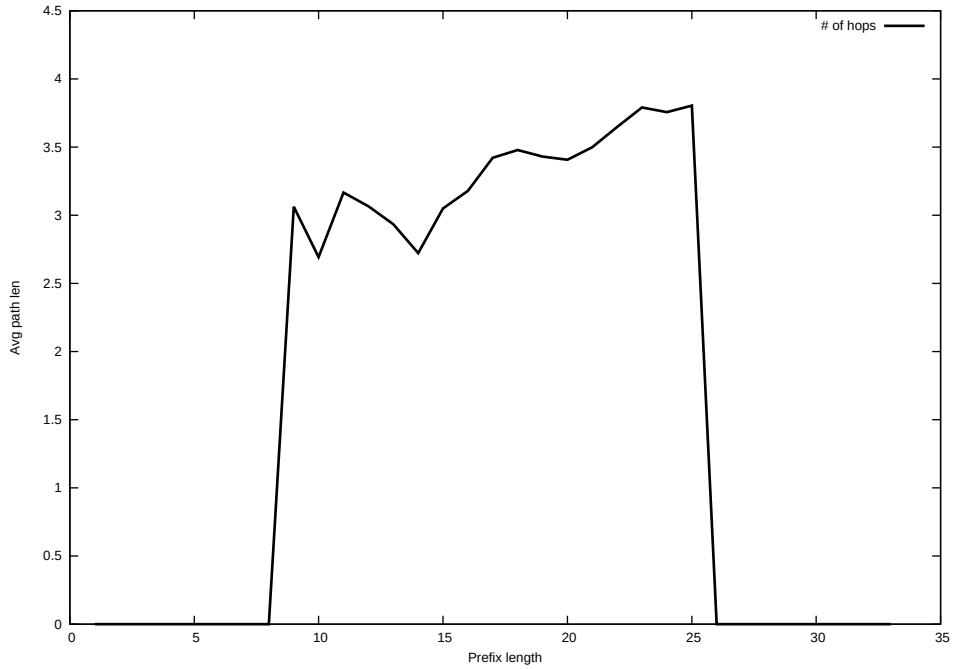
2014-08-28



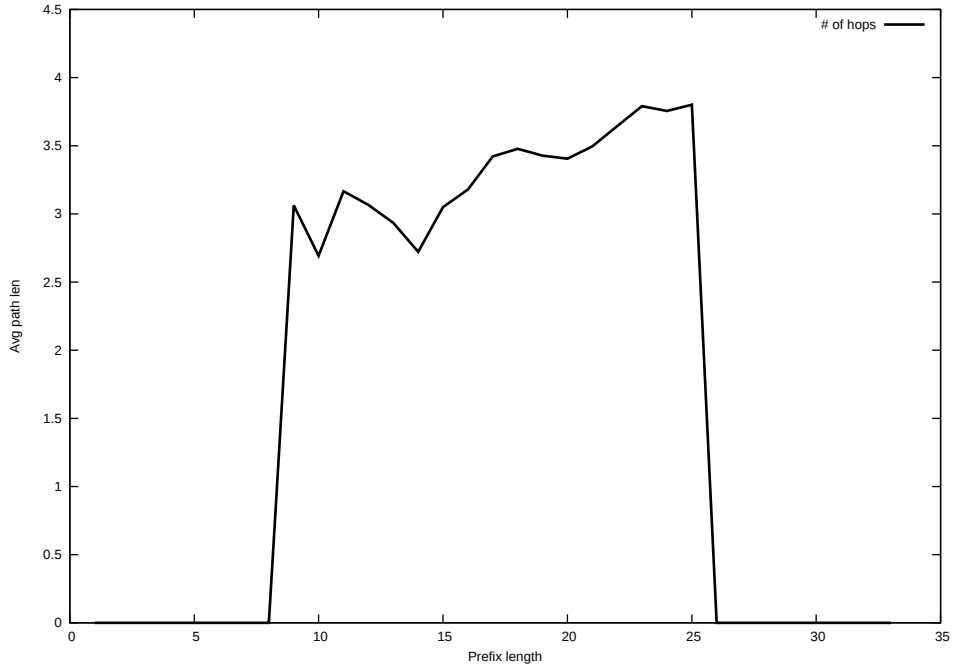
2014-08-29



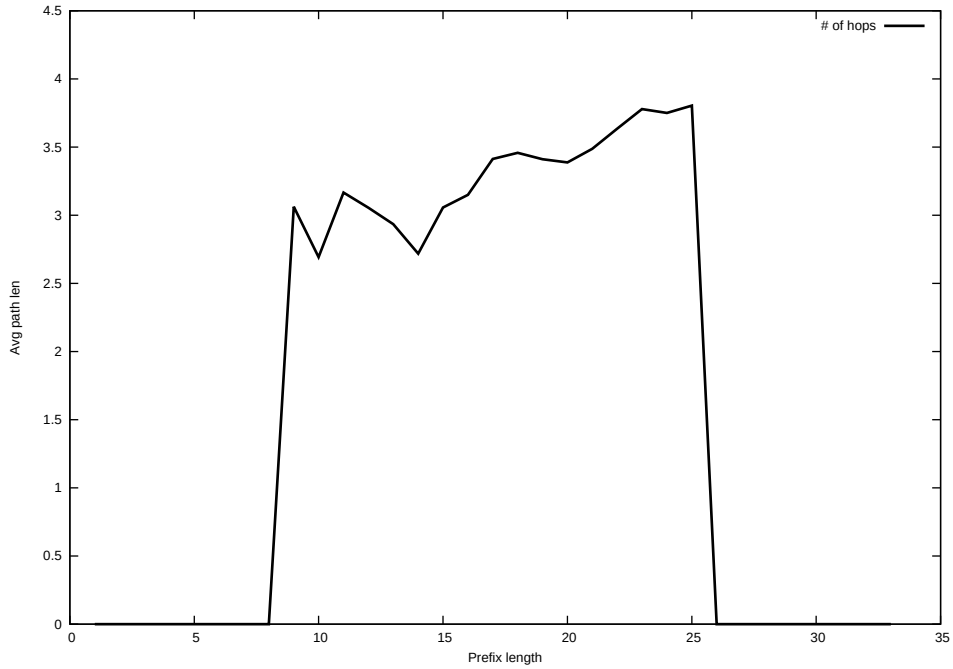
2014-08-30



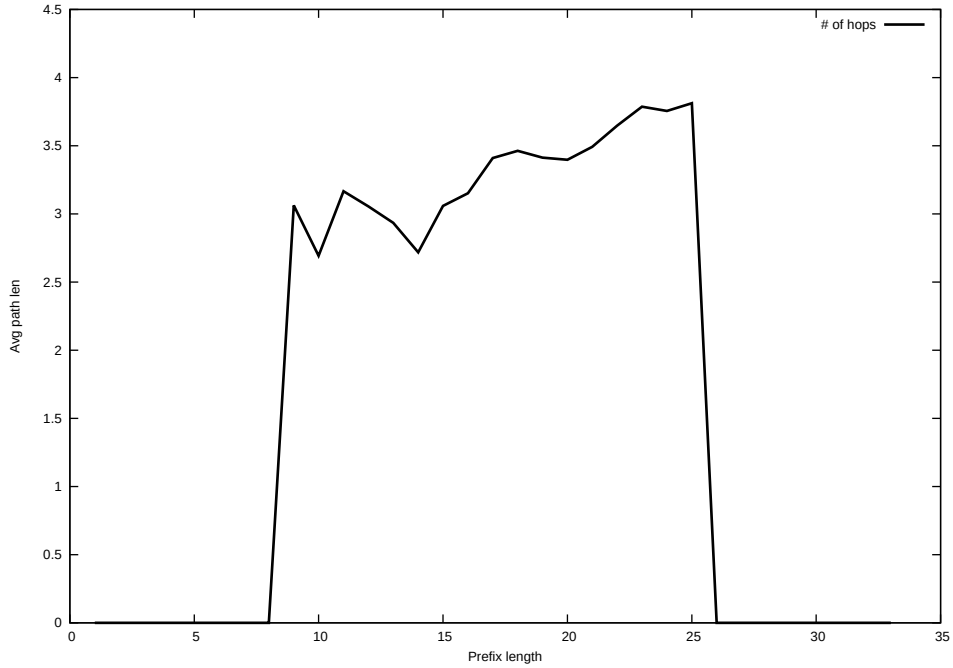
2014-08-31



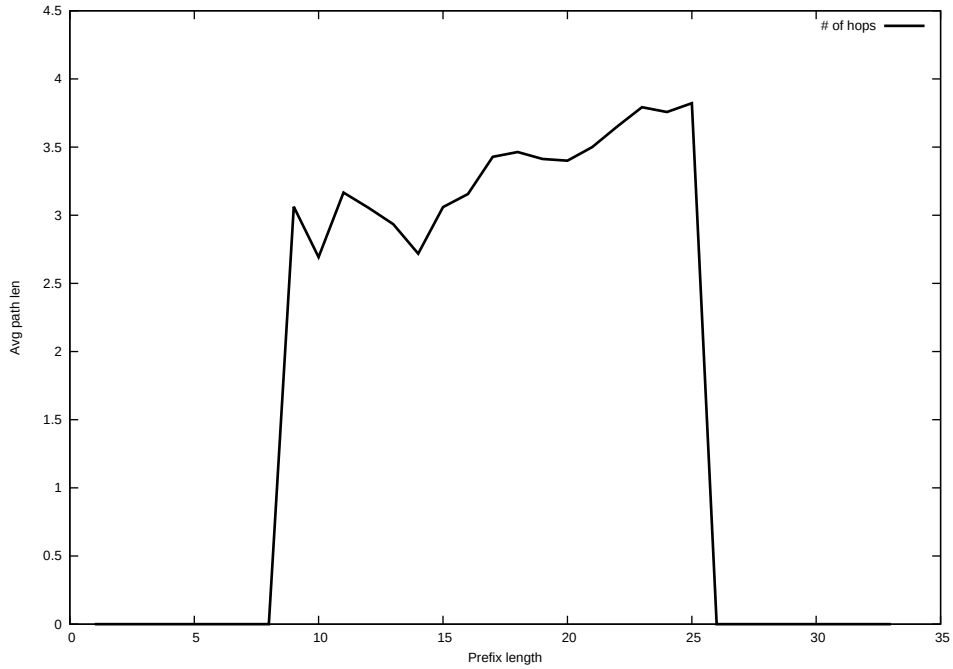
2014-09-01



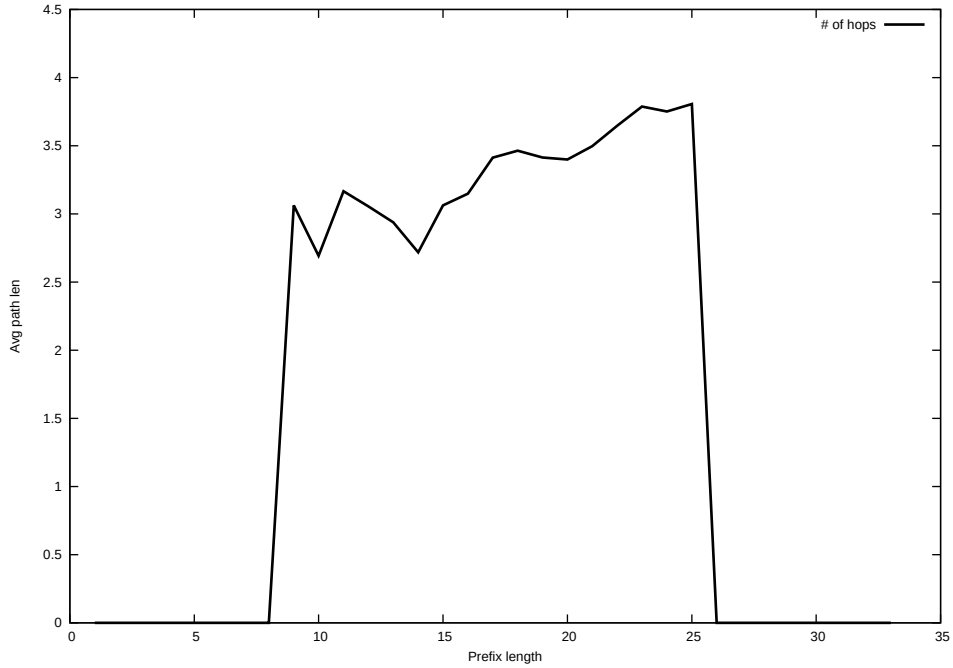
2014-09-02



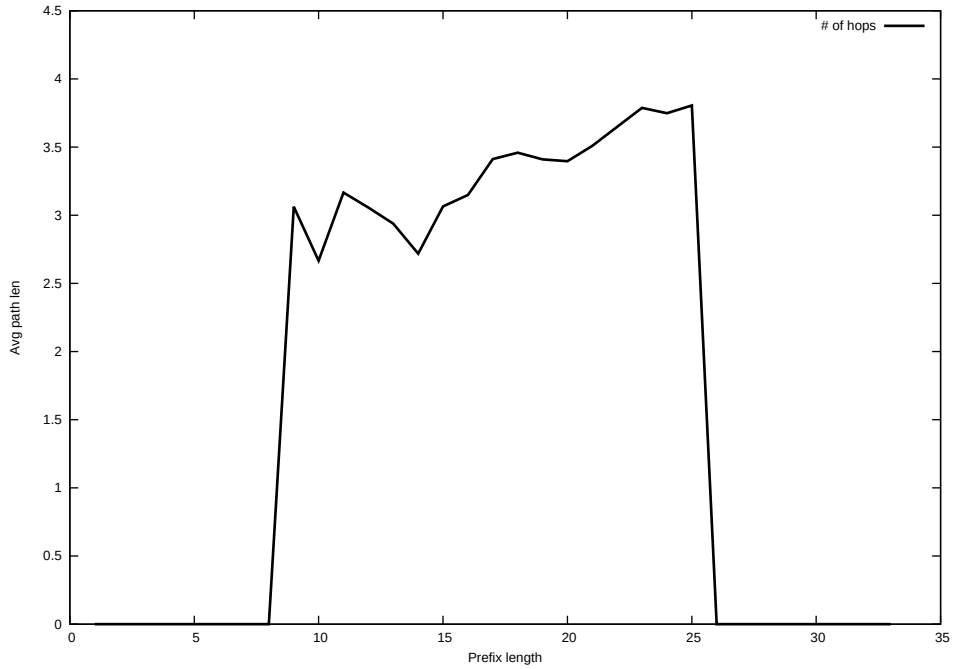
2014-09-03



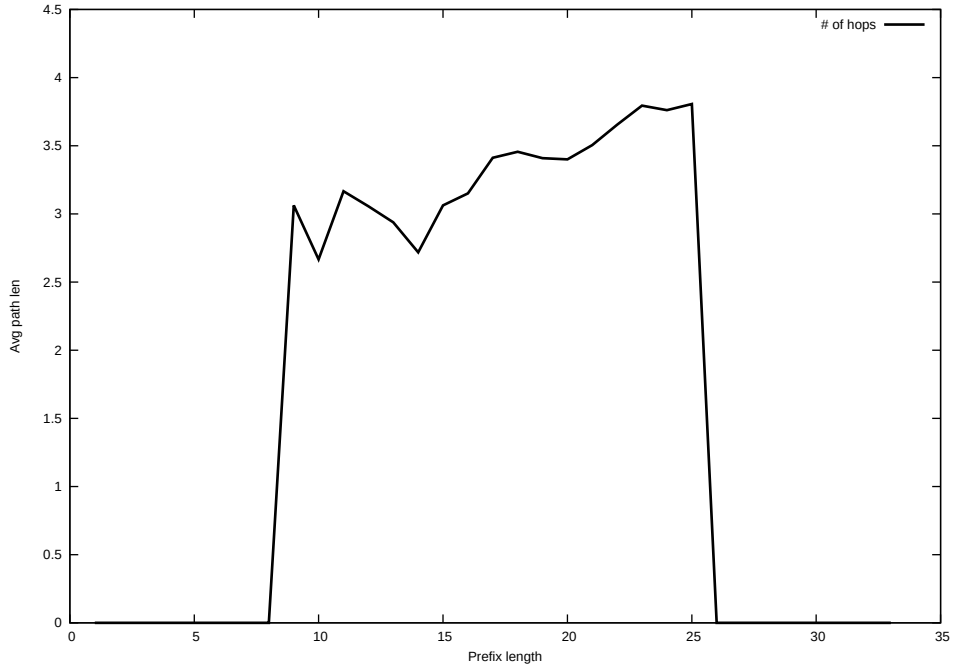
2014-09-04



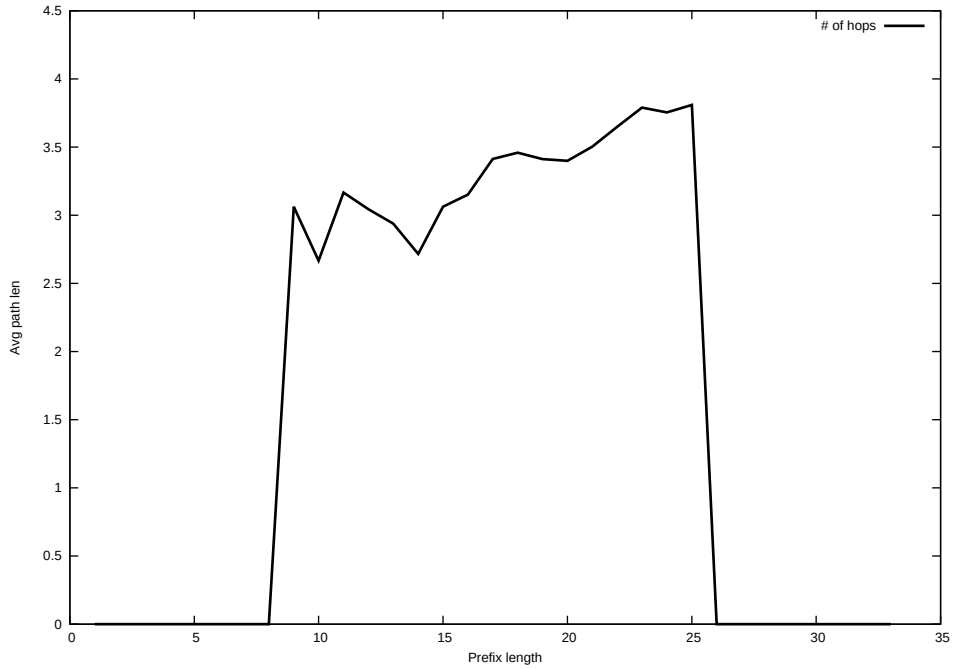
2014-09-05



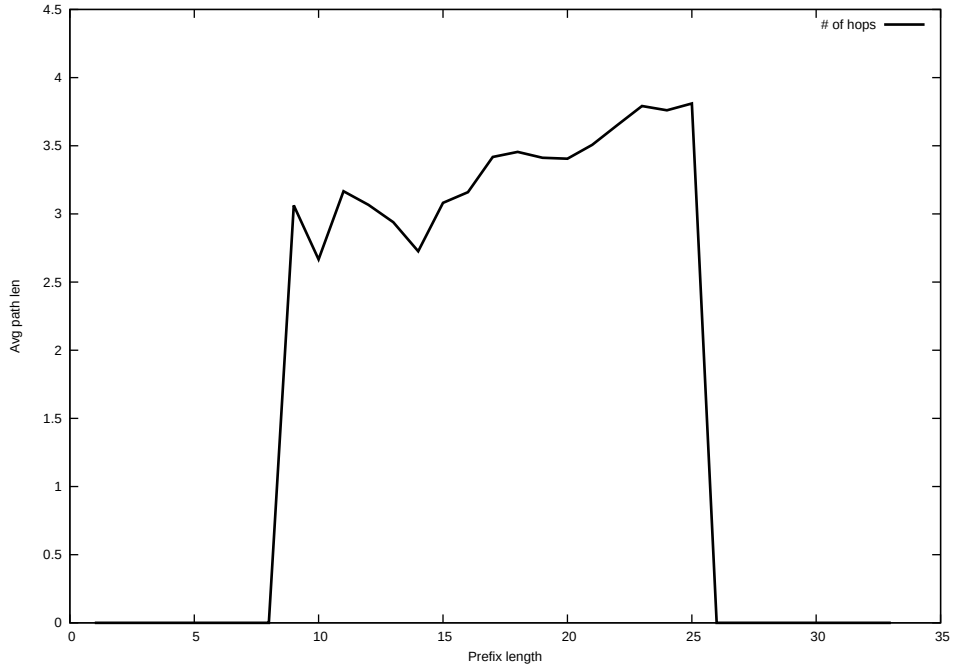
2014-09-06



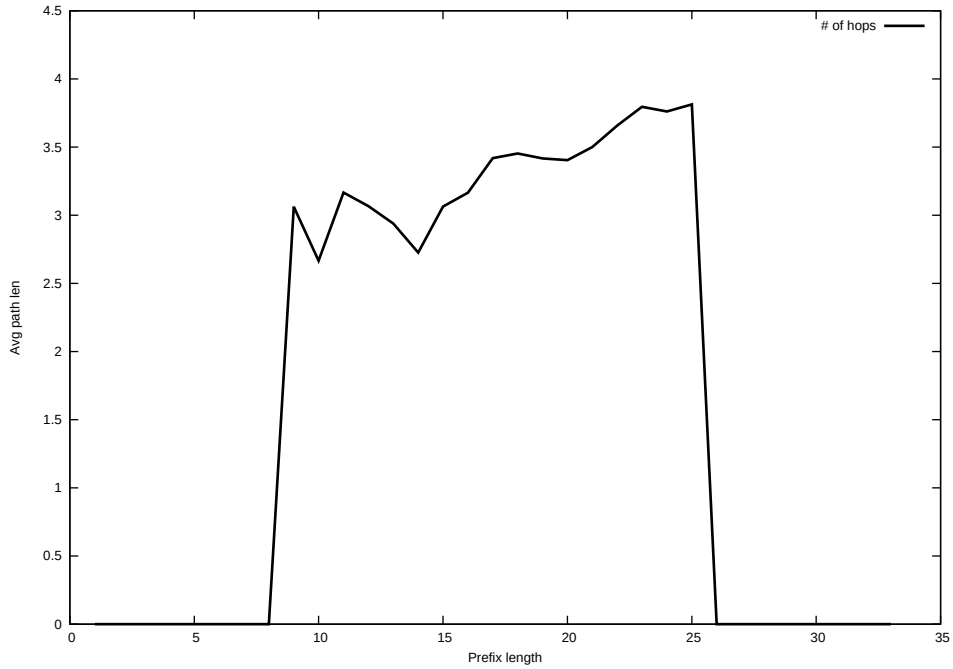
2014-09-07



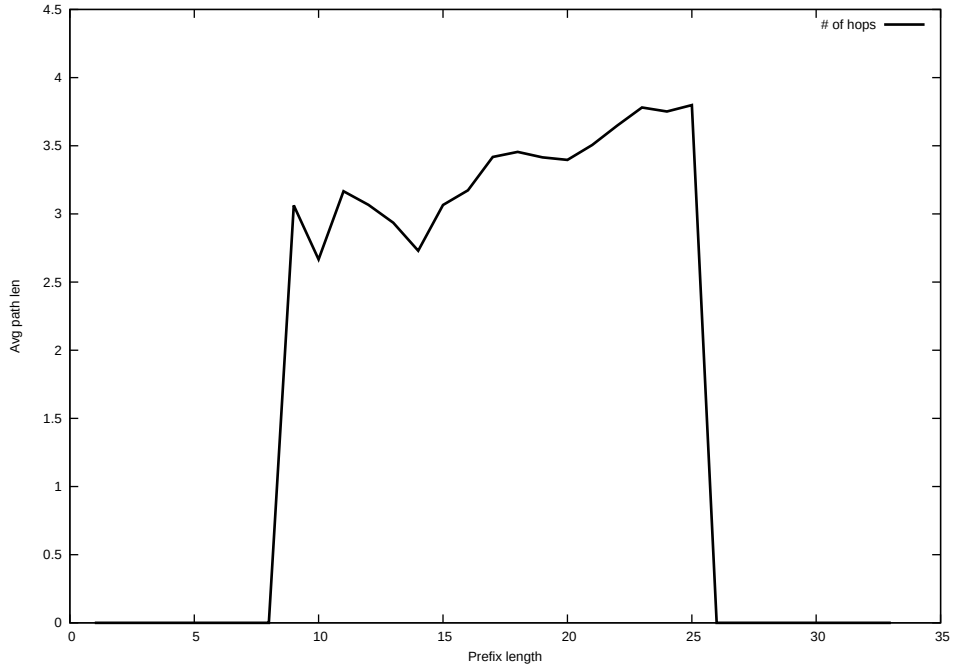
2014-09-08



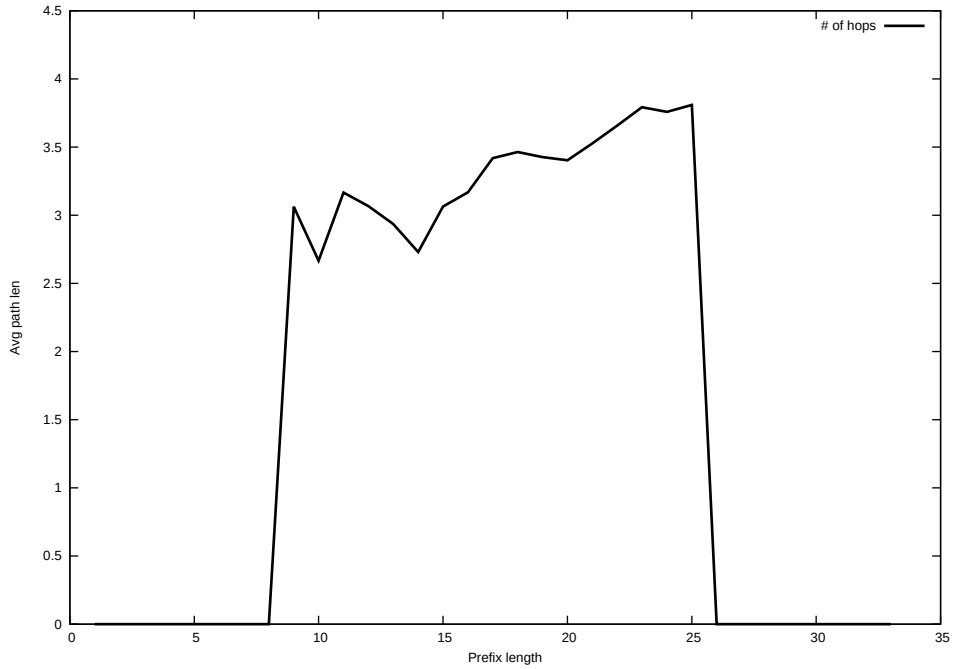
2014-09-09



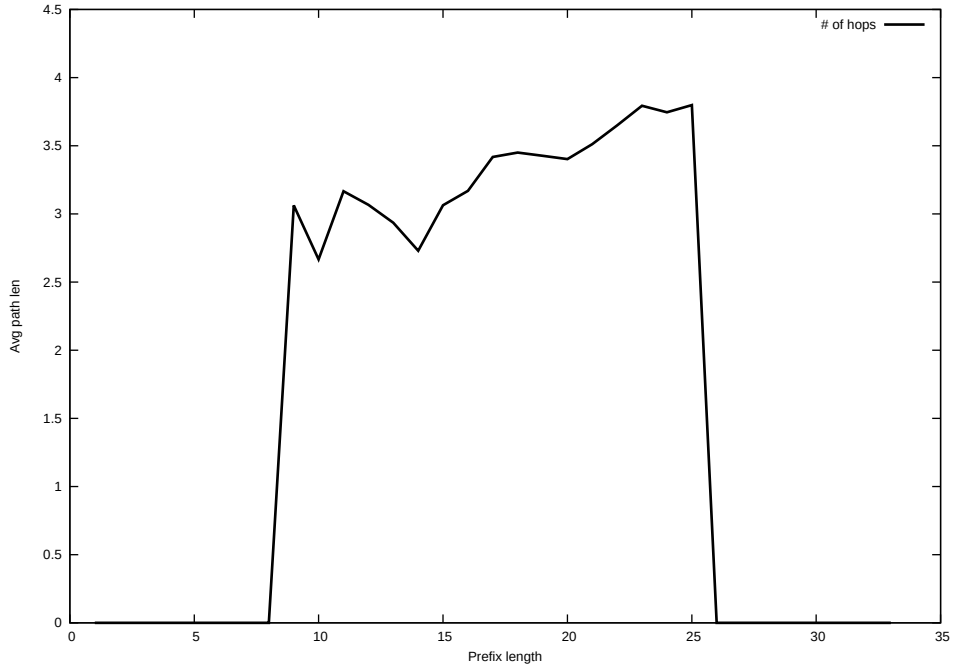
2014-09-10



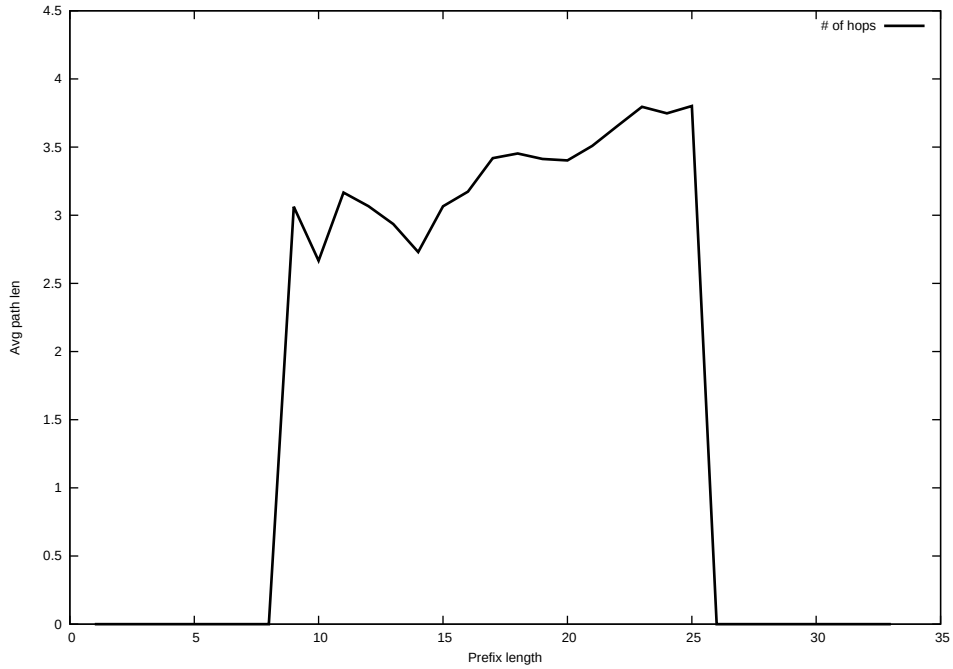
2014-09-11



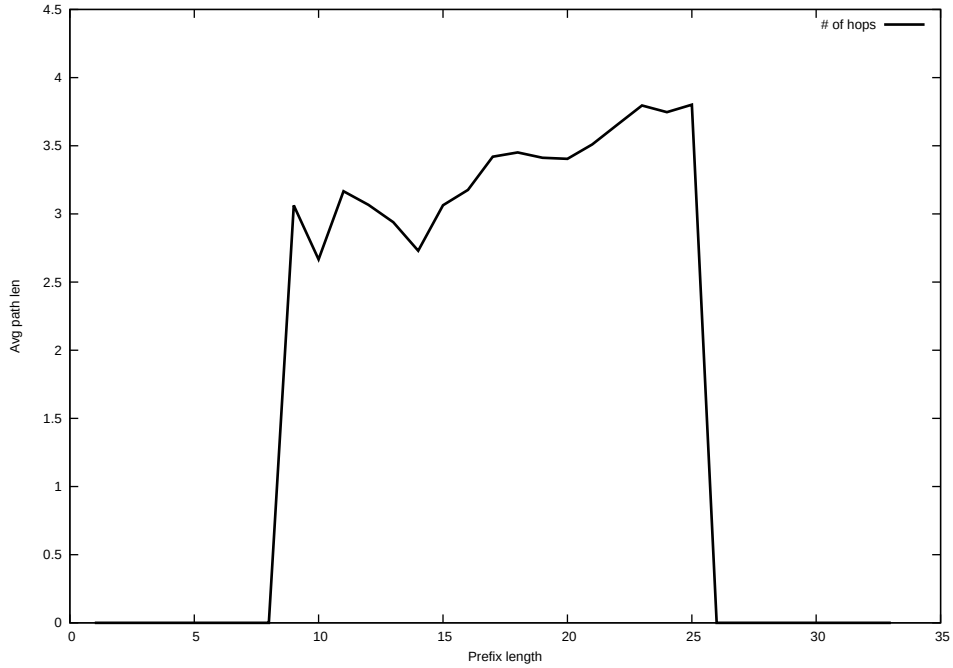
2014-09-12



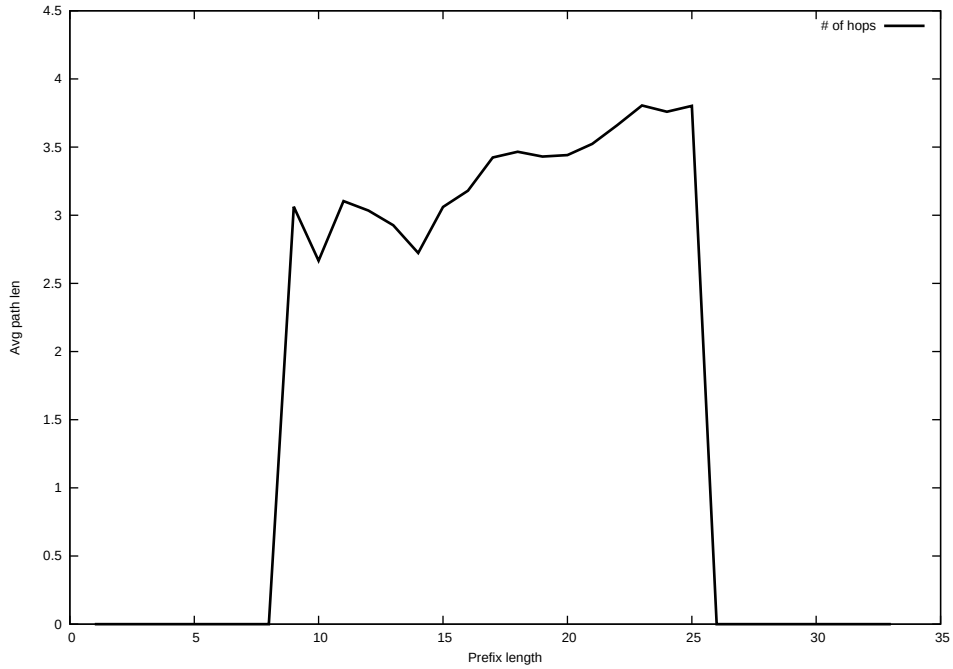
2014-09-13



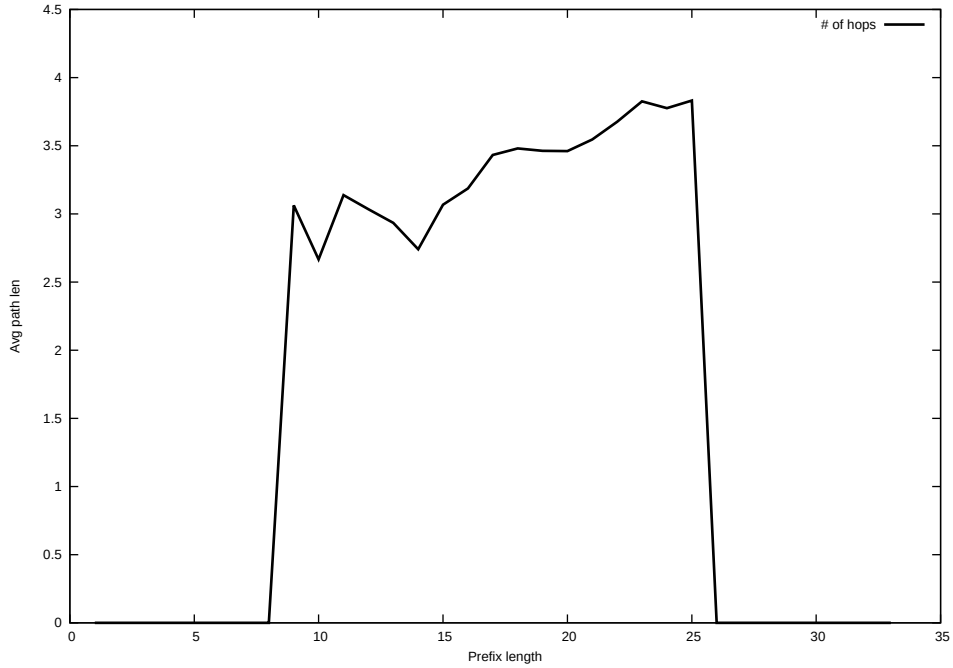
2014-09-14



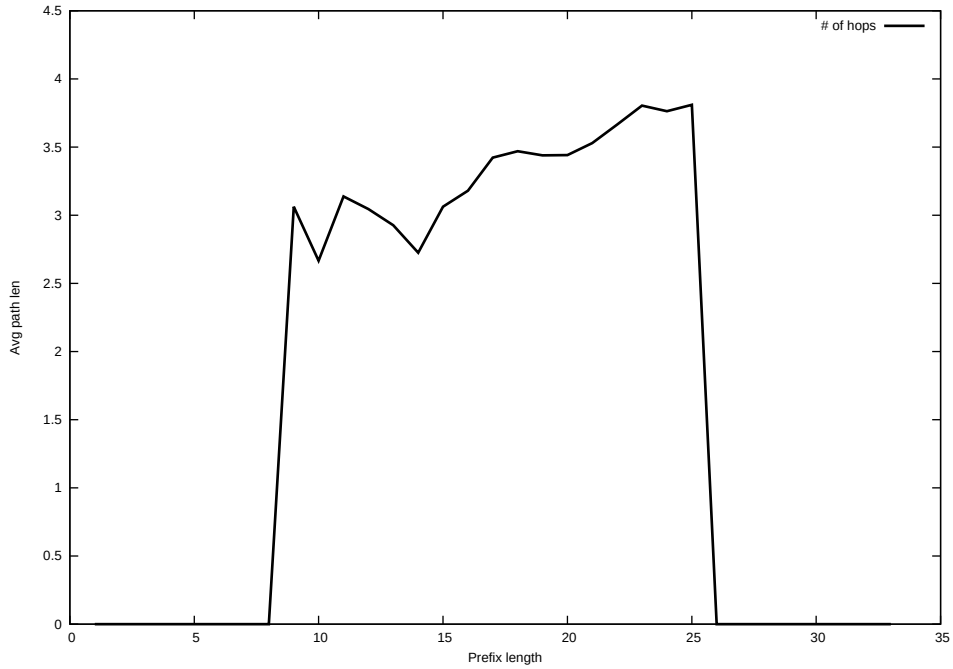
2014-09-15



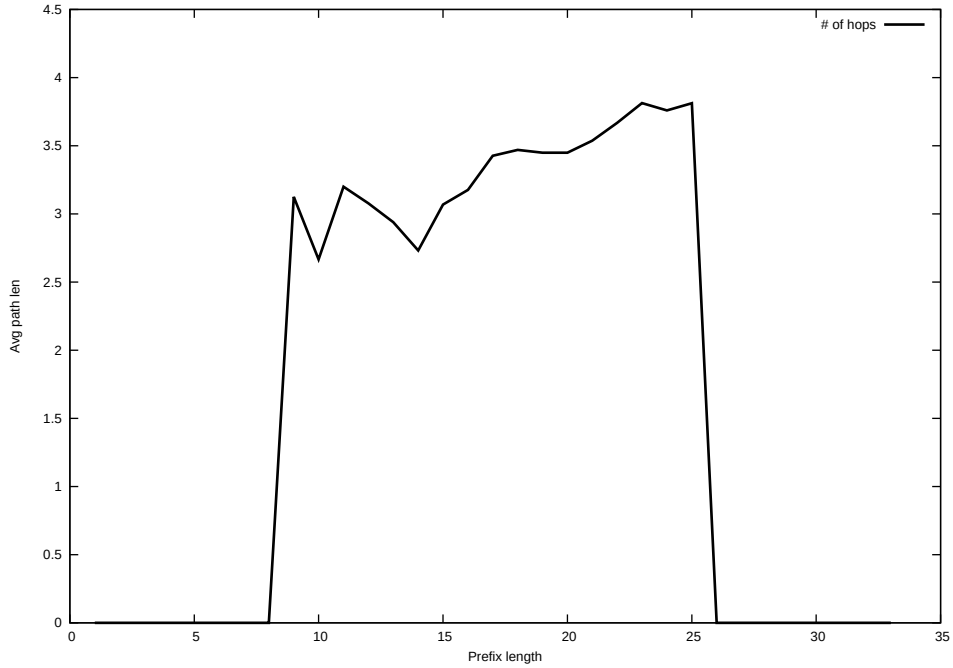
2014-09-16



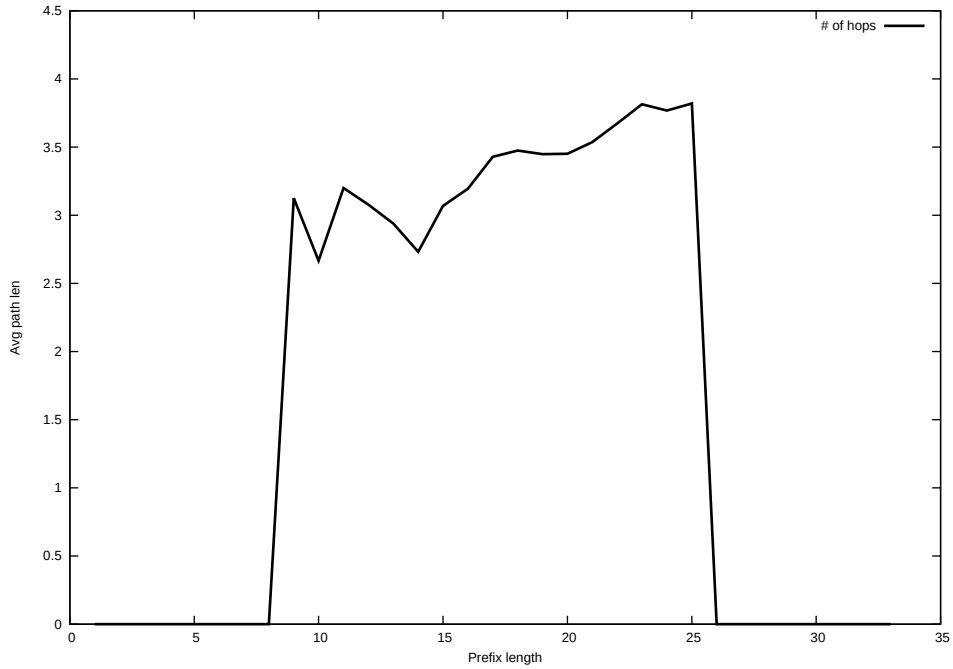
2014-09-17



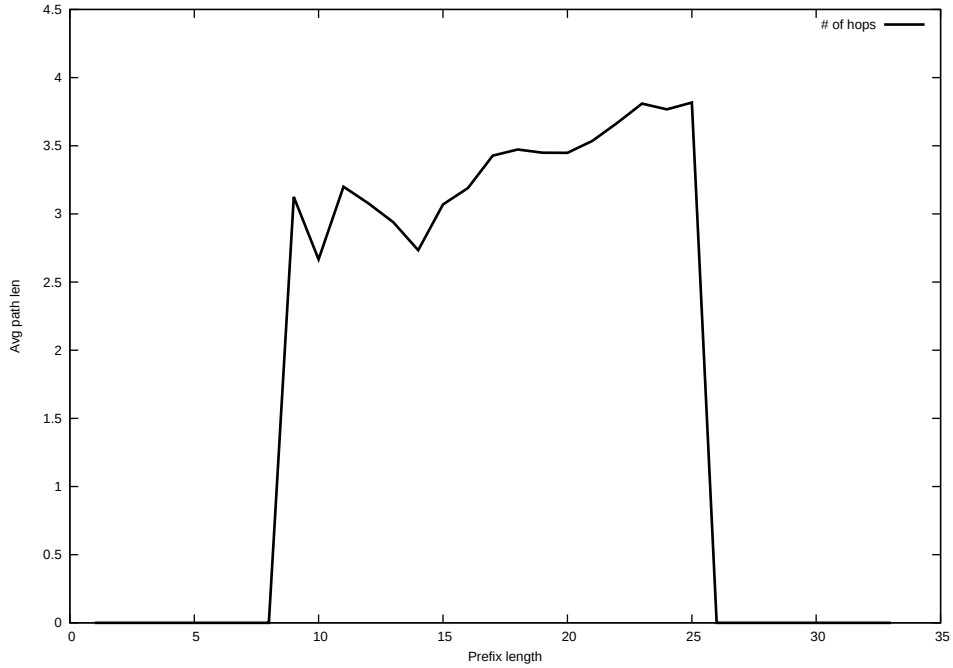
2014-09-18



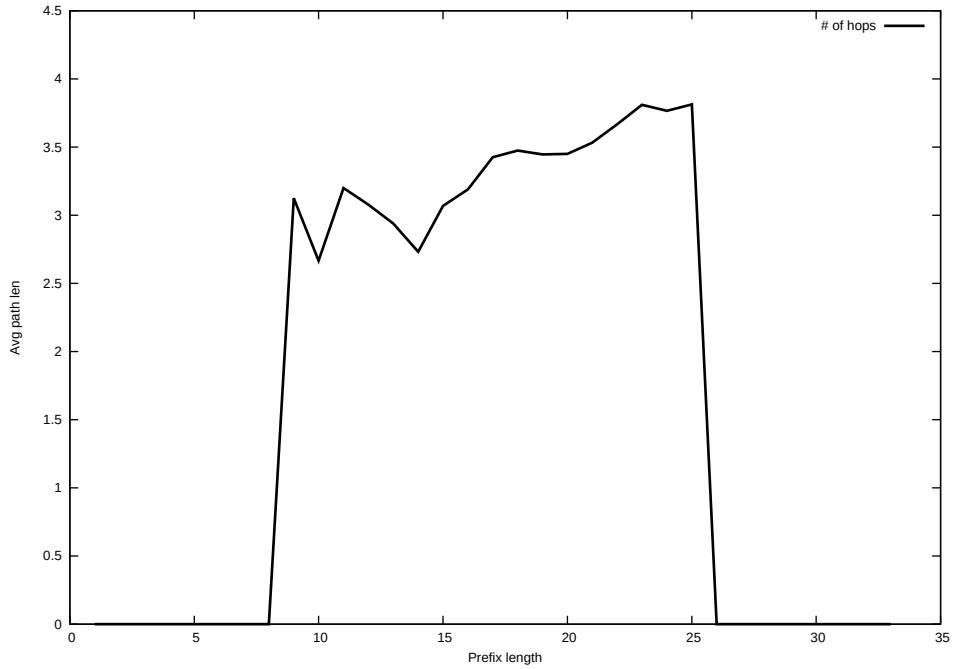
2014-09-19



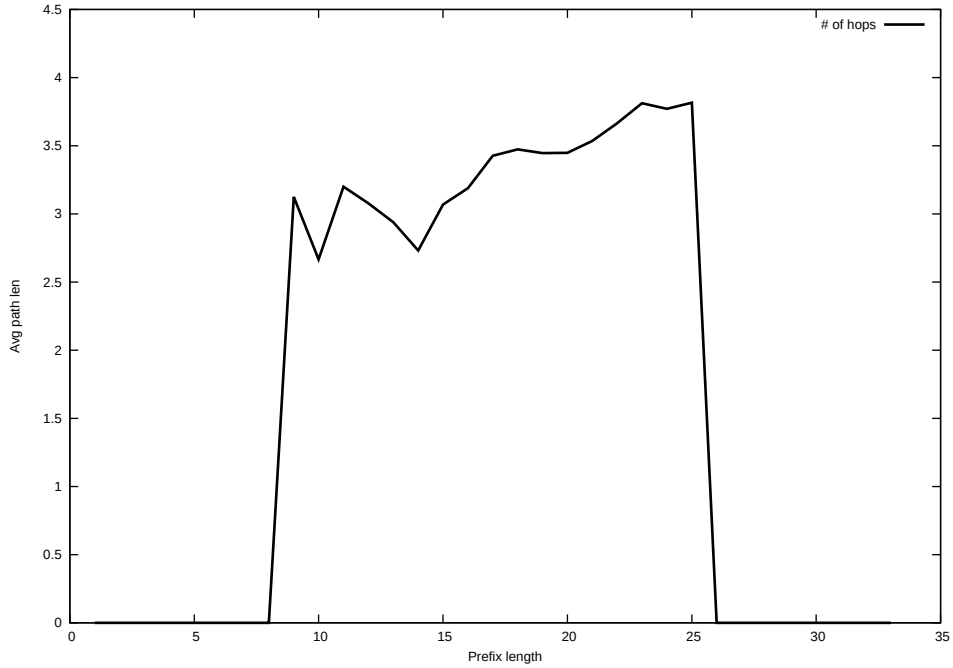
2014-09-20



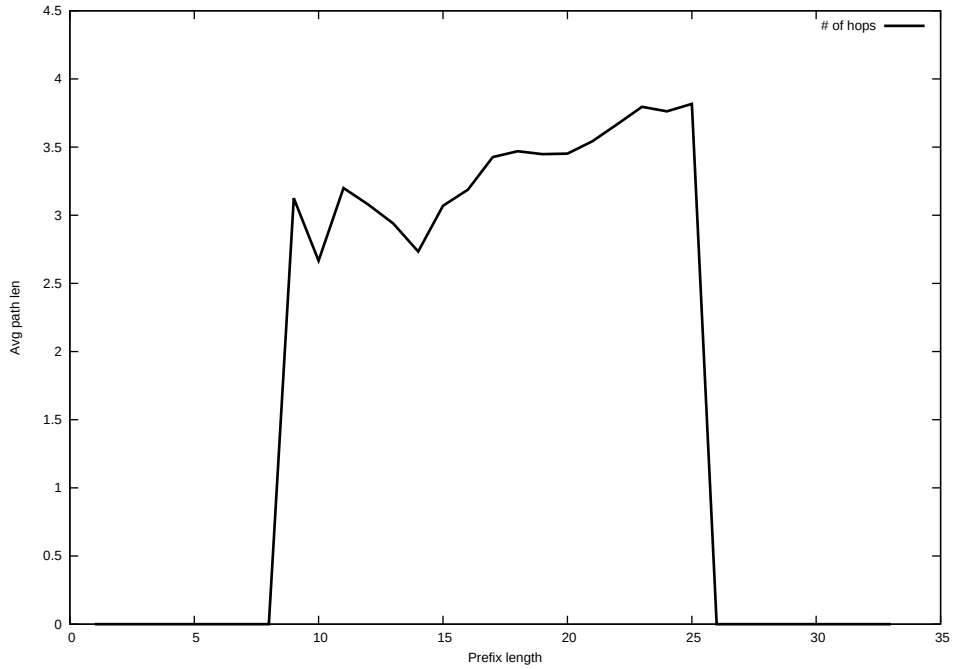
2014-09-21



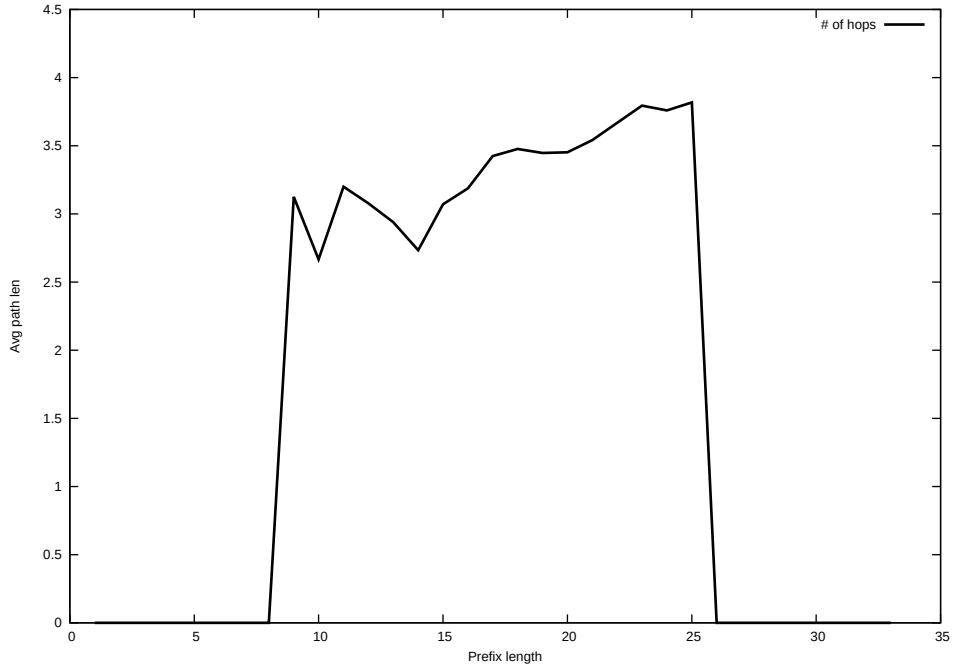
2014-09-22



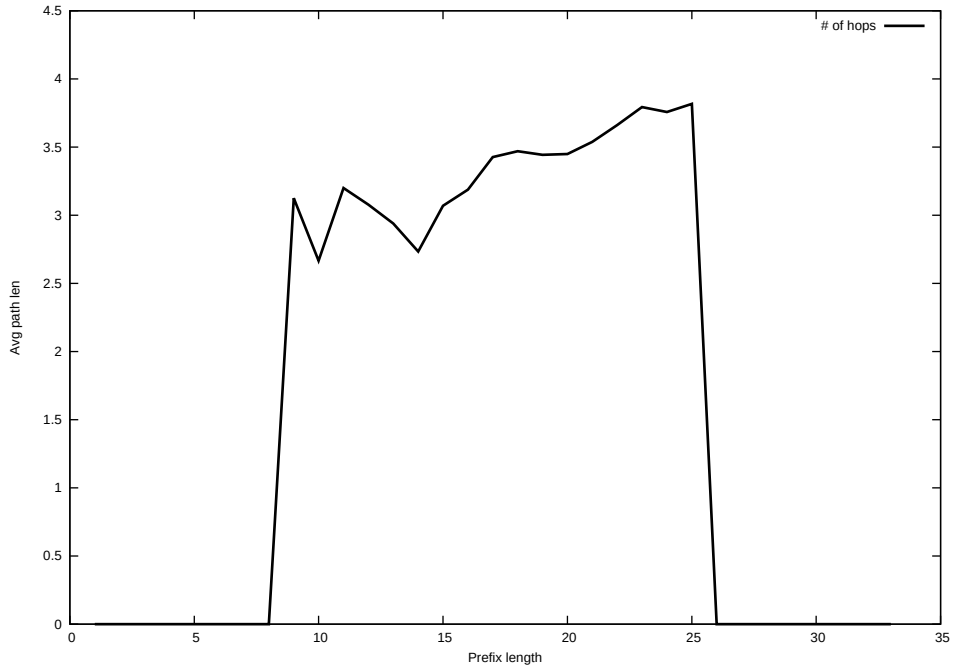
2014-09-23



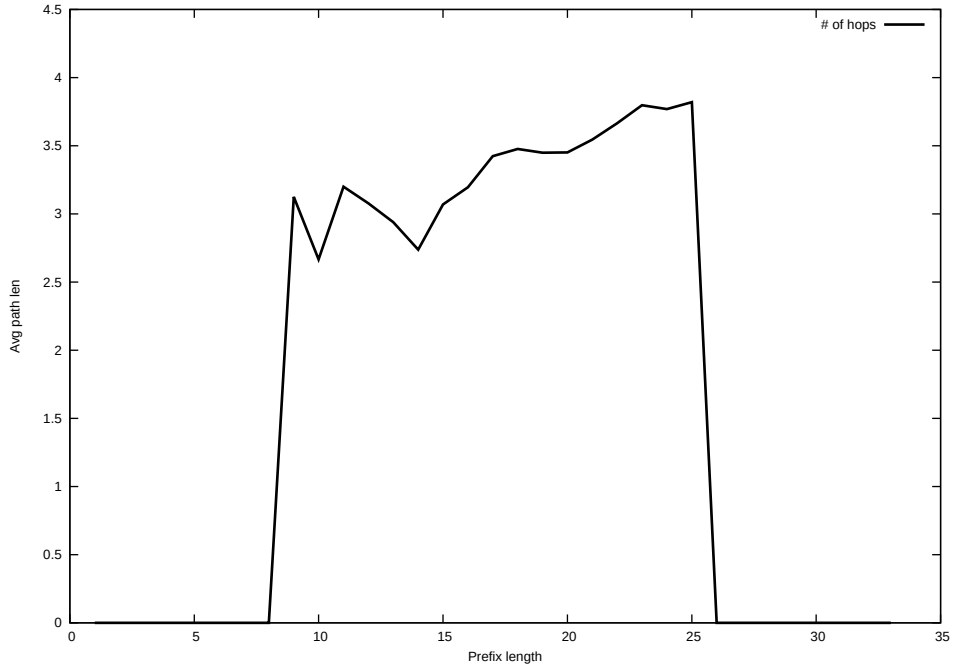
2014-09-24



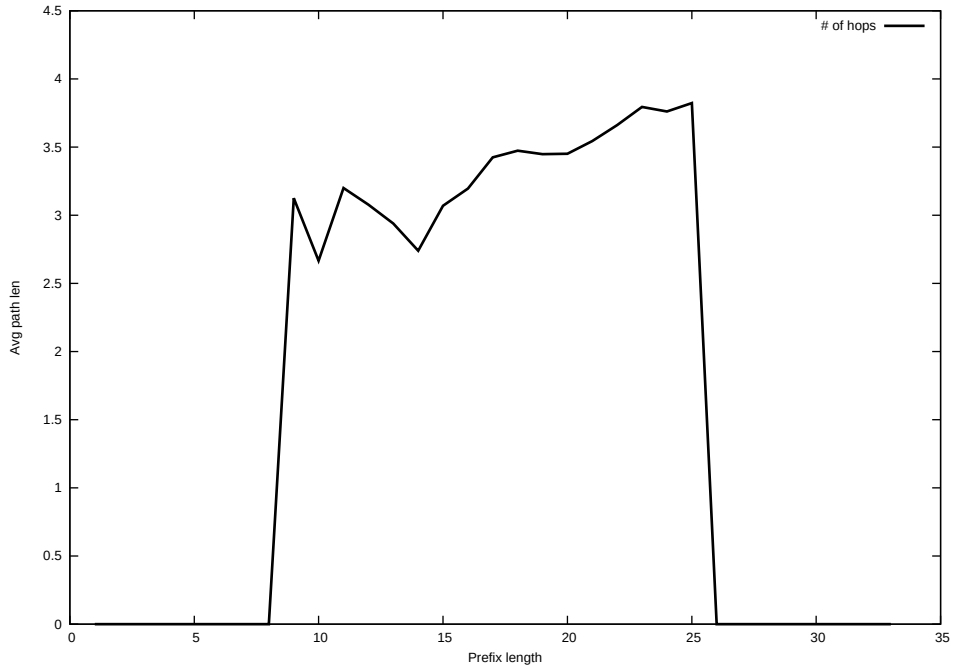
2014-09-25



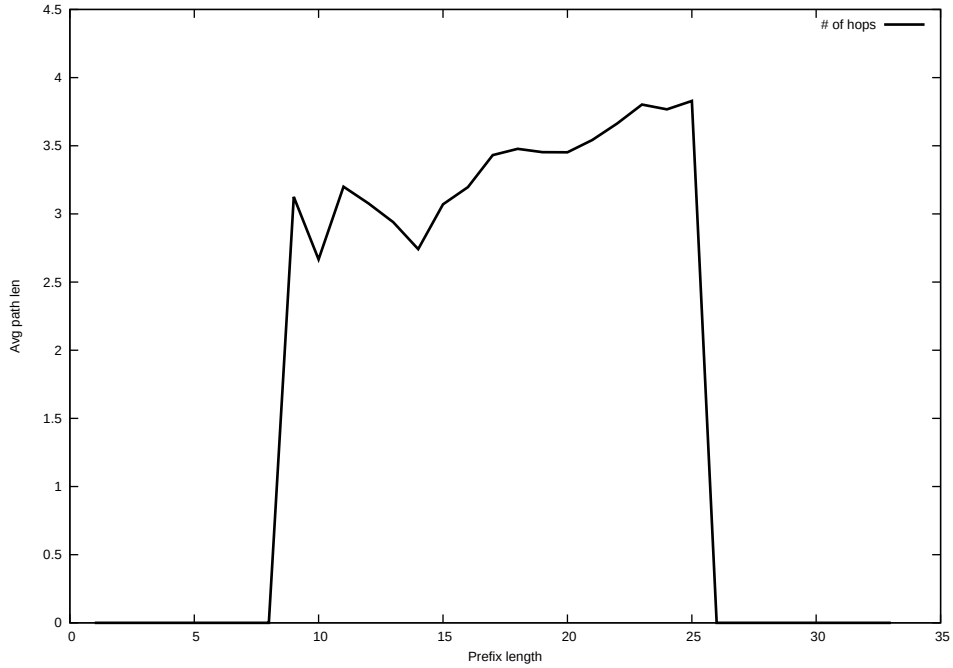
2014-09-26



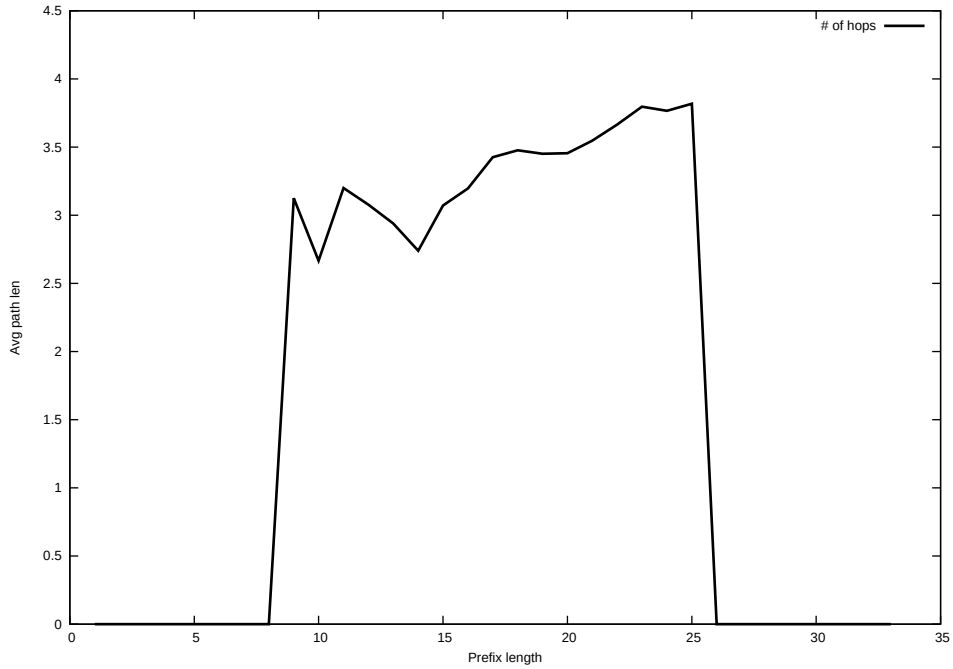
2014-09-27



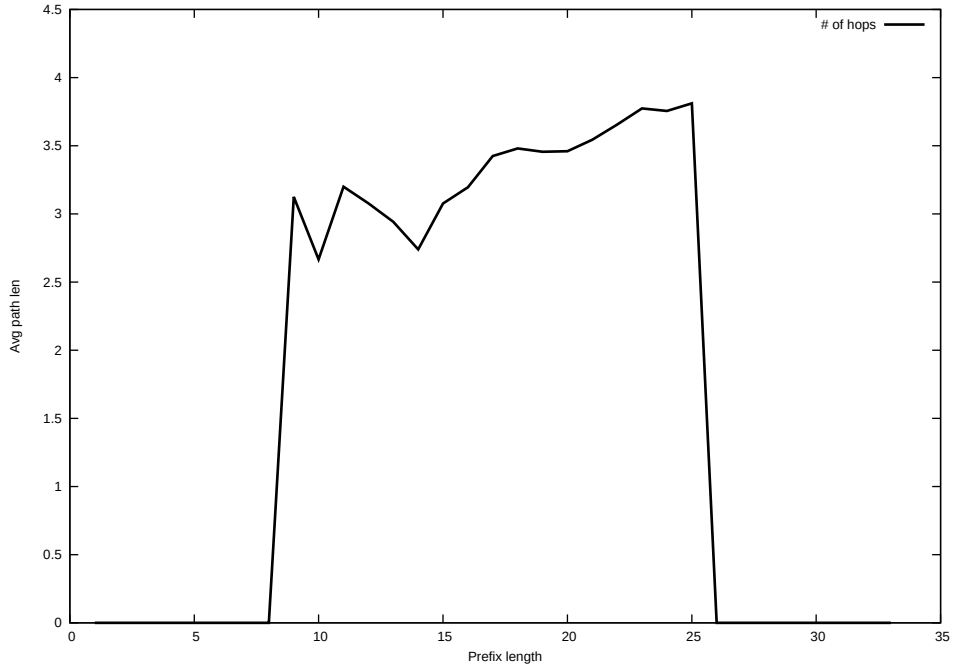
2014-09-28



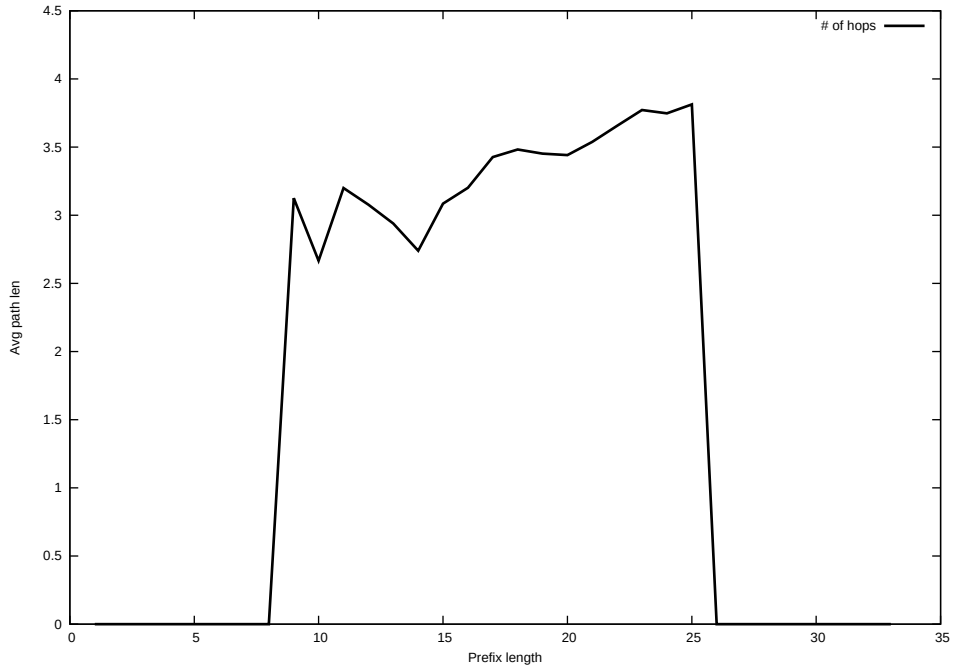
2014-09-29



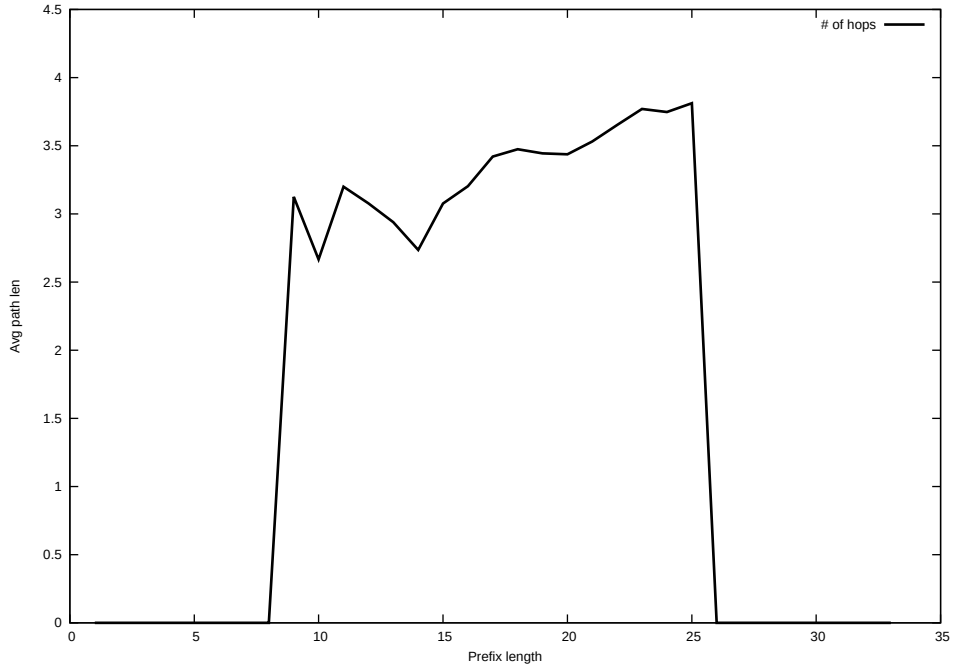
2014-09-30



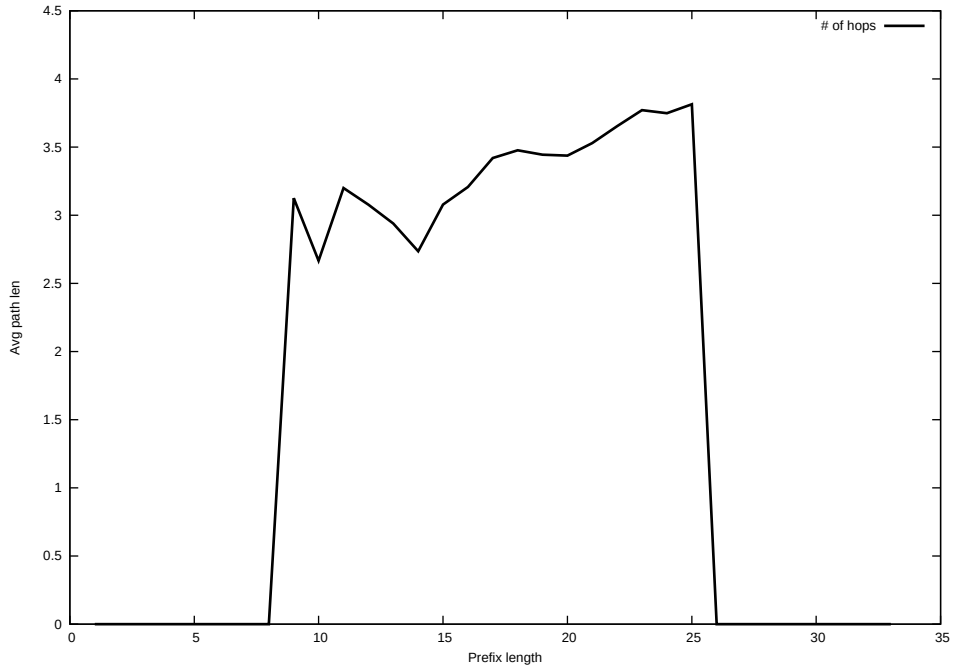
2014-10-01



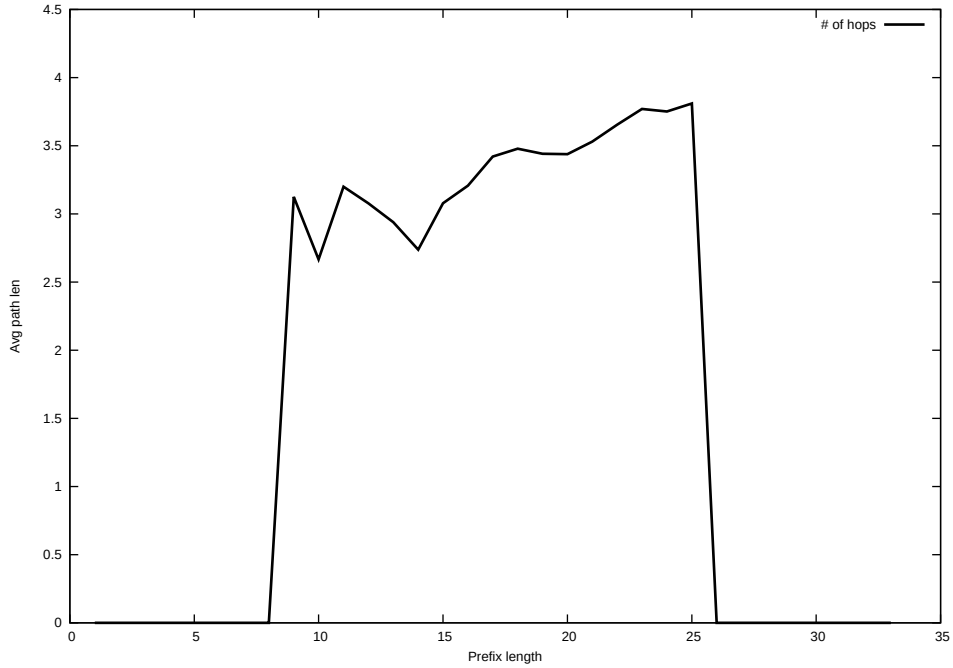
2014-10-02



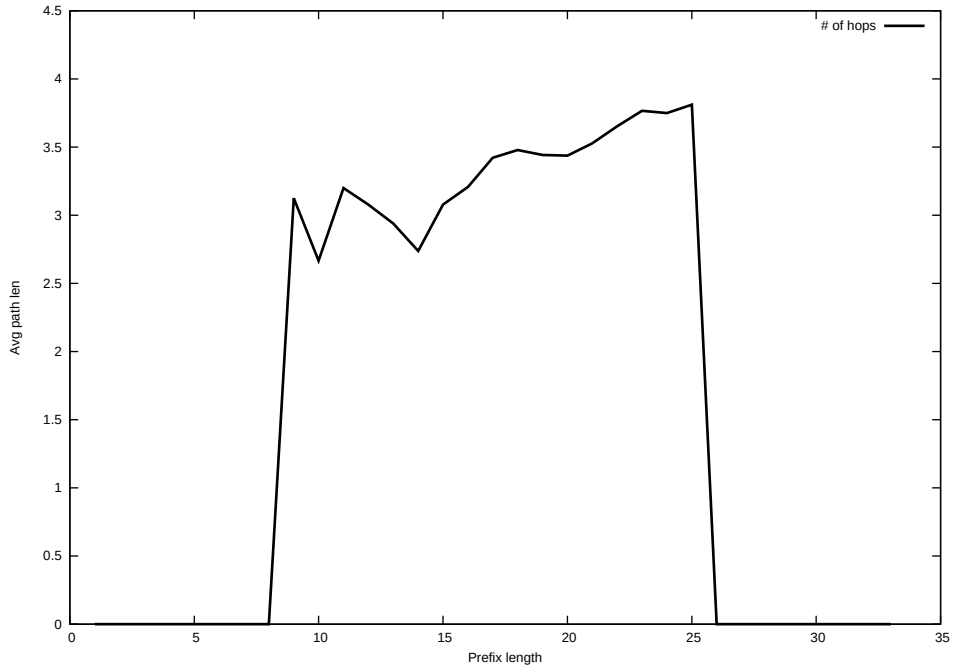
2014-10-03



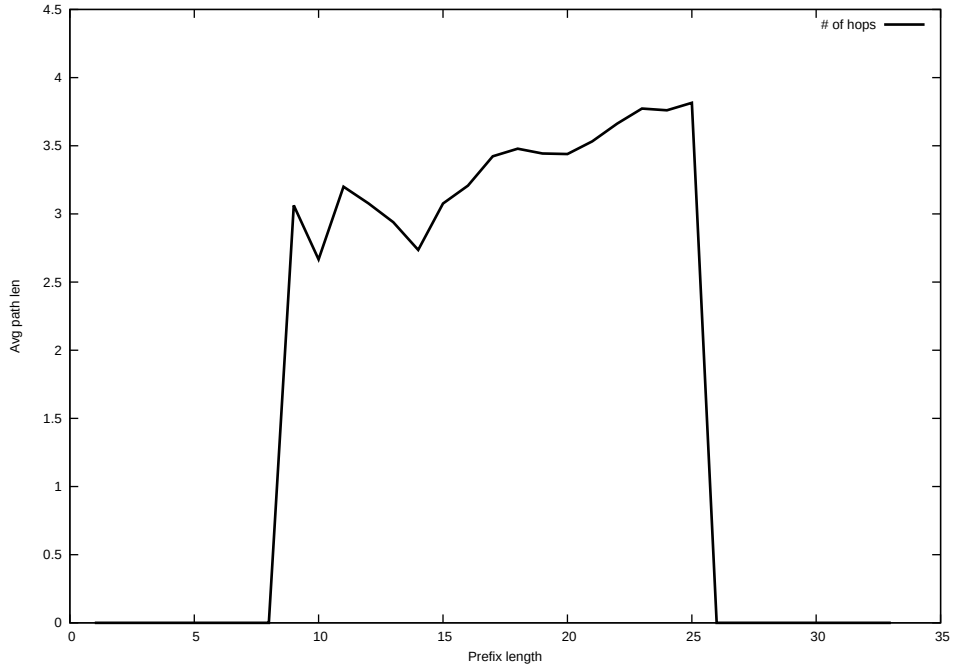
2014-10-04



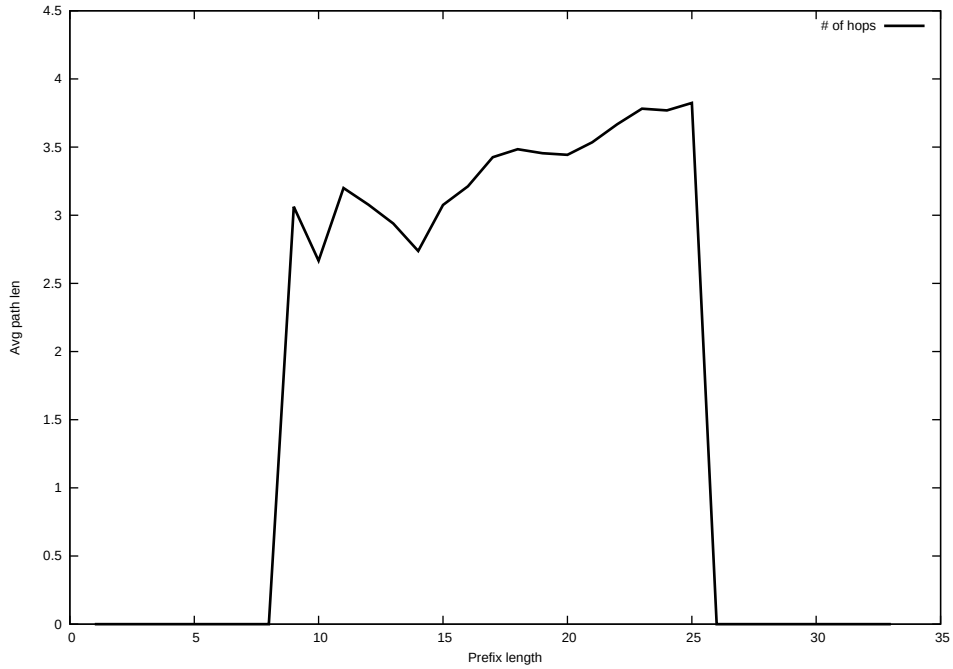
2014-10-05



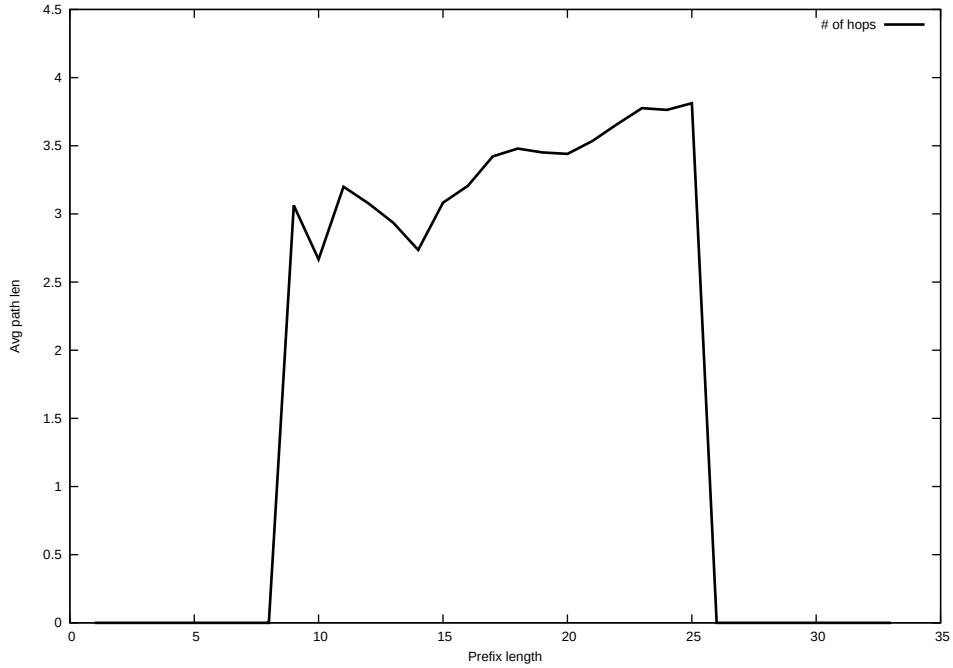
2014-10-06



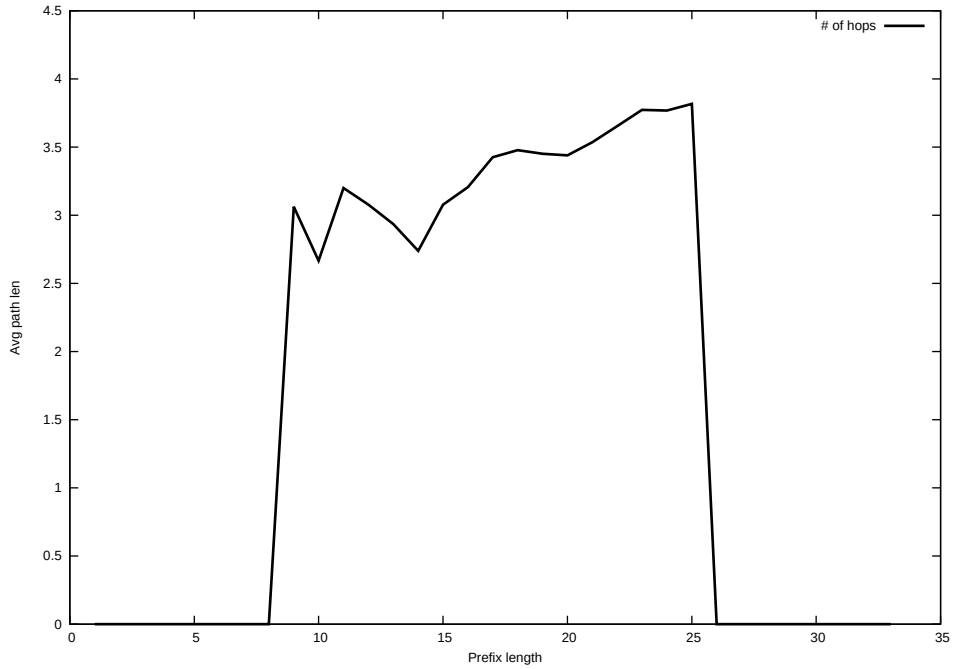
2014-10-07



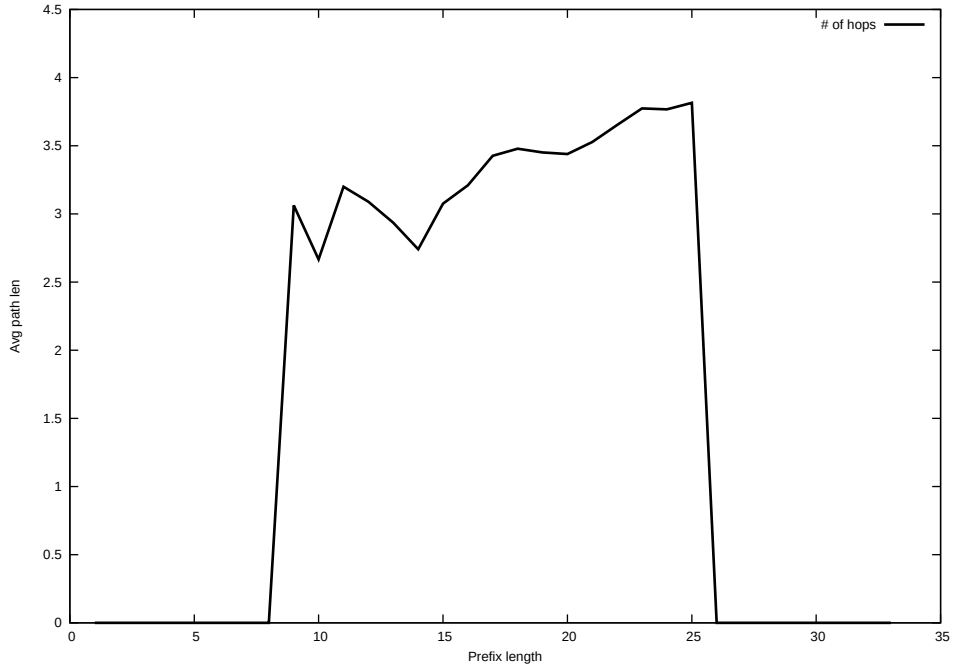
2014-10-08



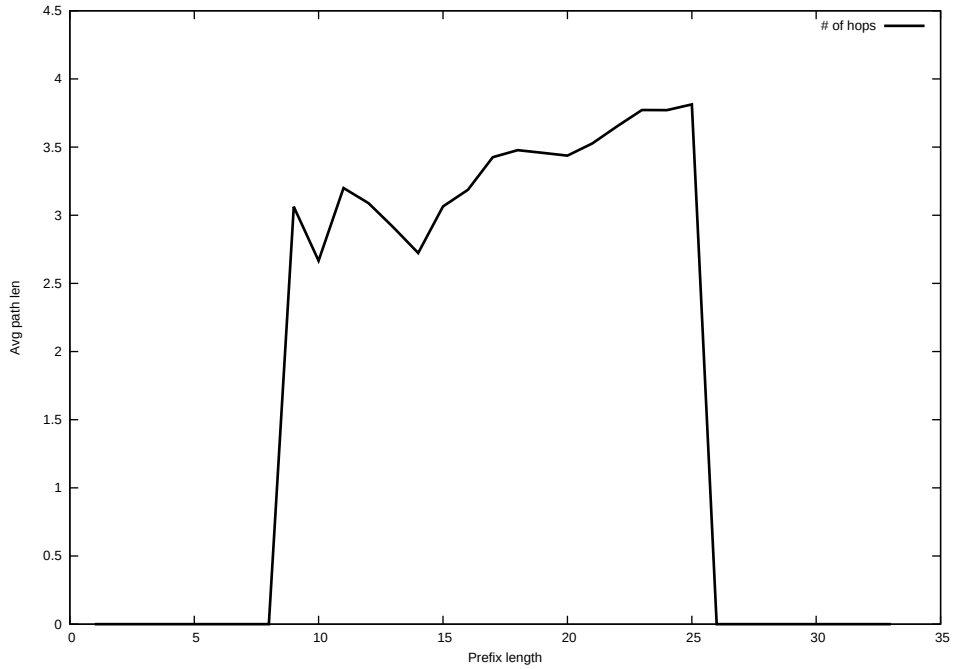
2014-10-09



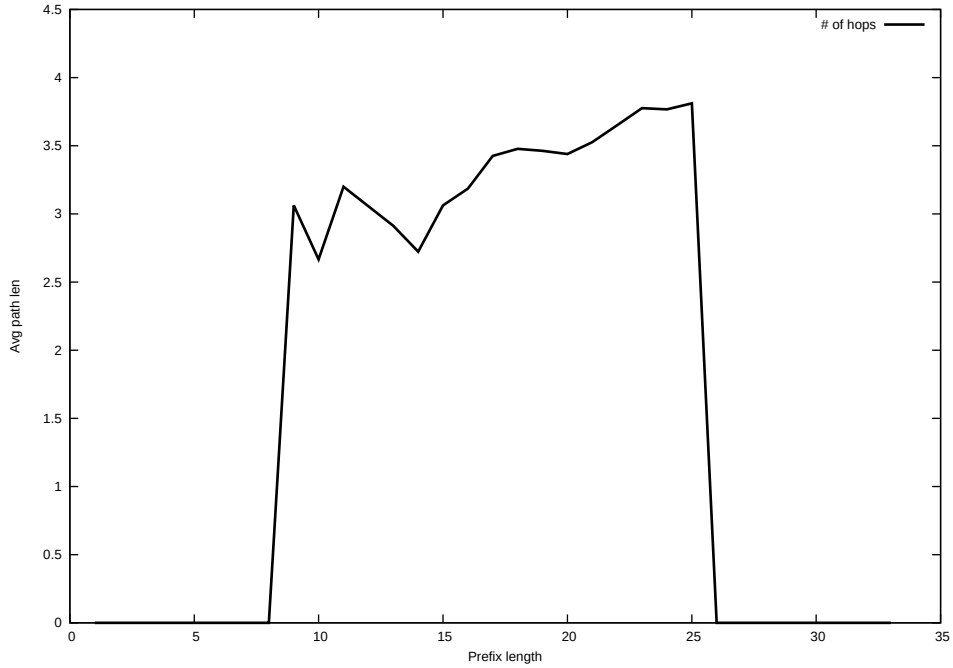
2014-10-10



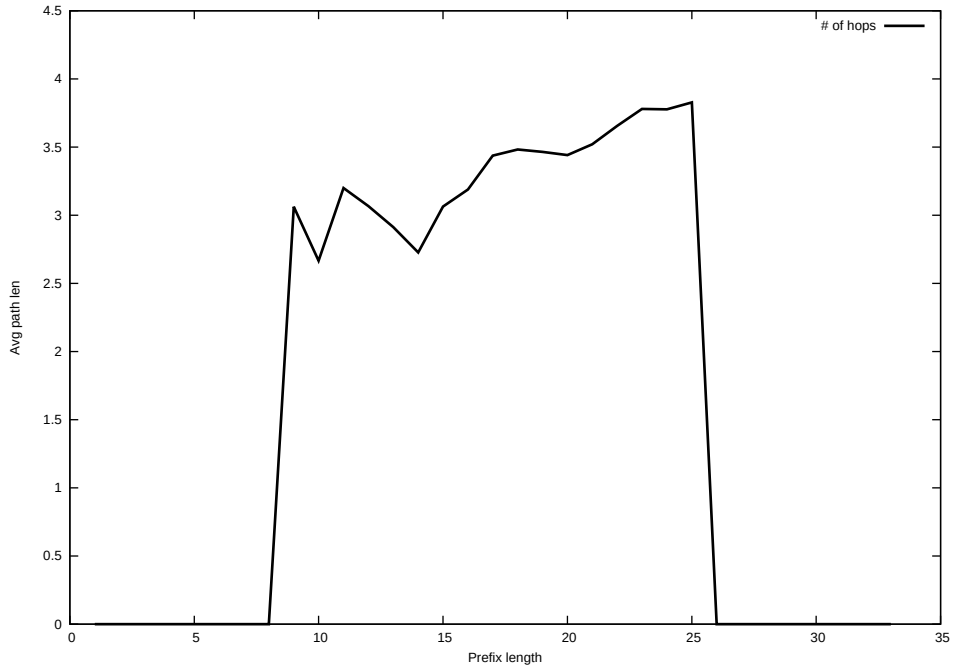
2014-10-11



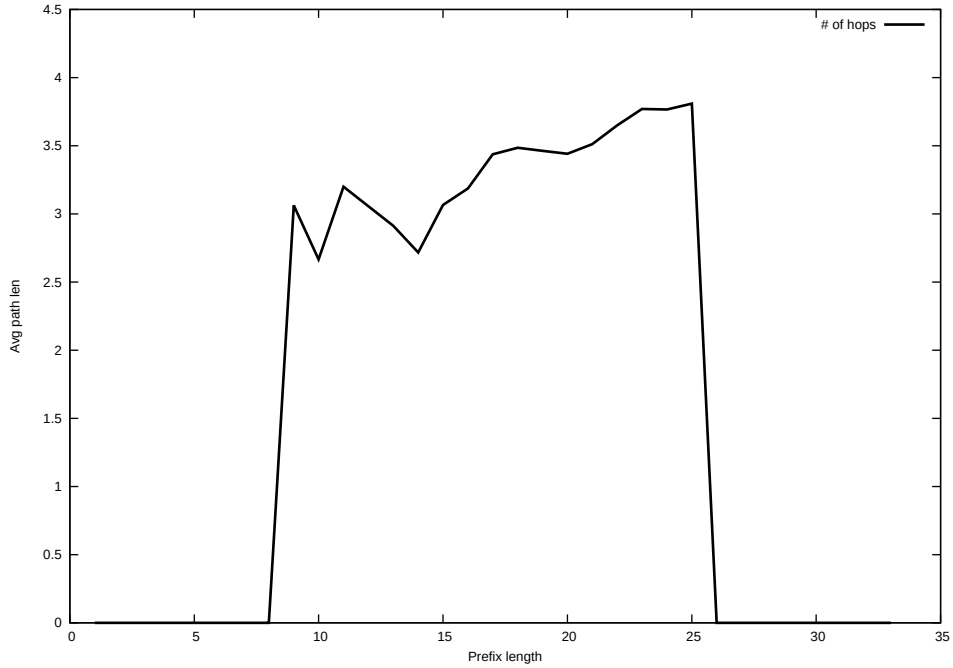
2014-10-12



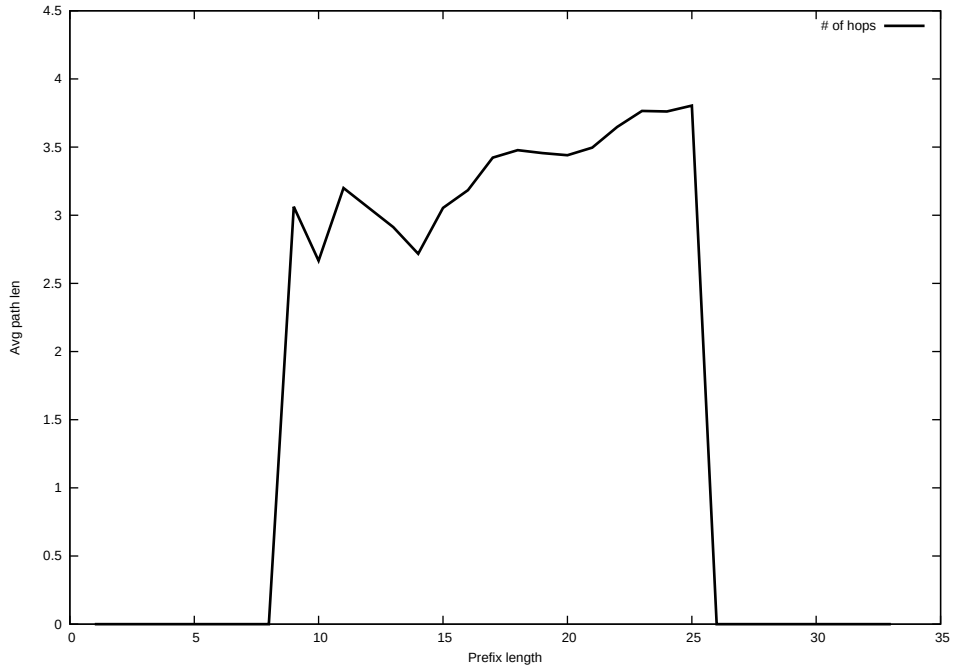
2014-10-13



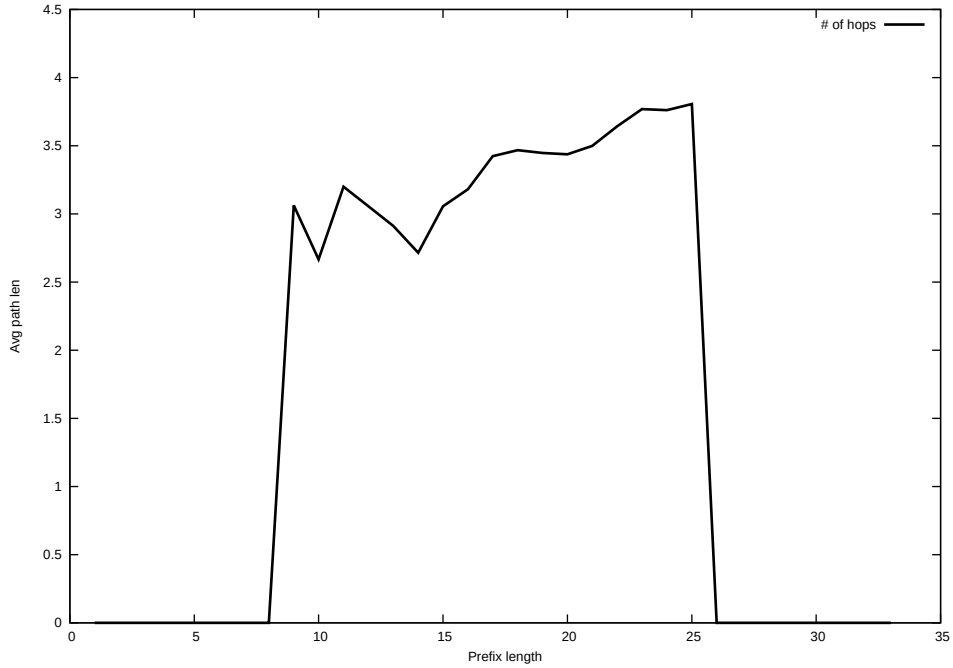
2014-10-14



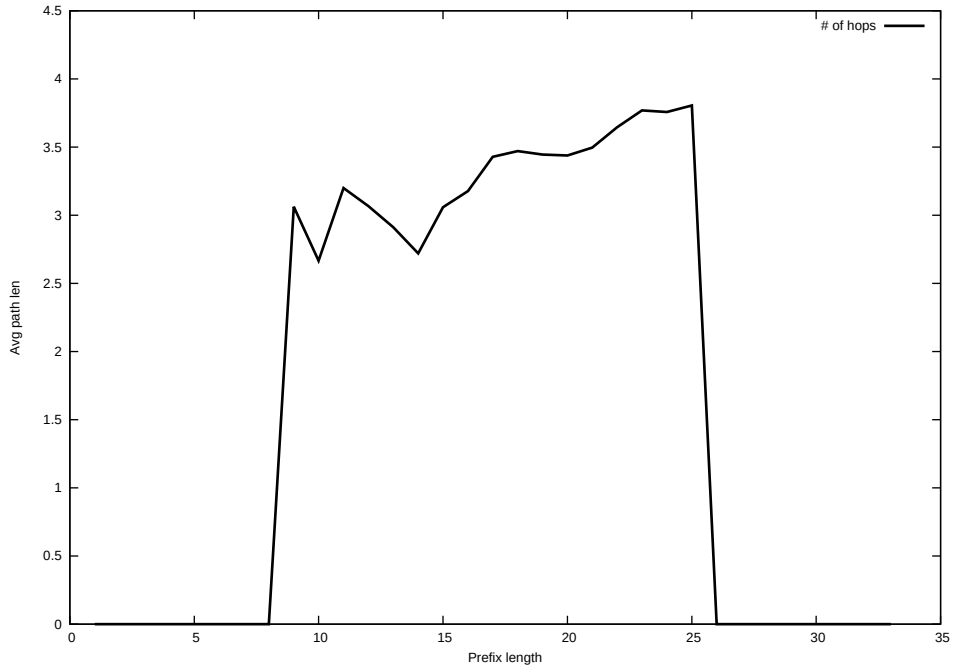
2014-10-15



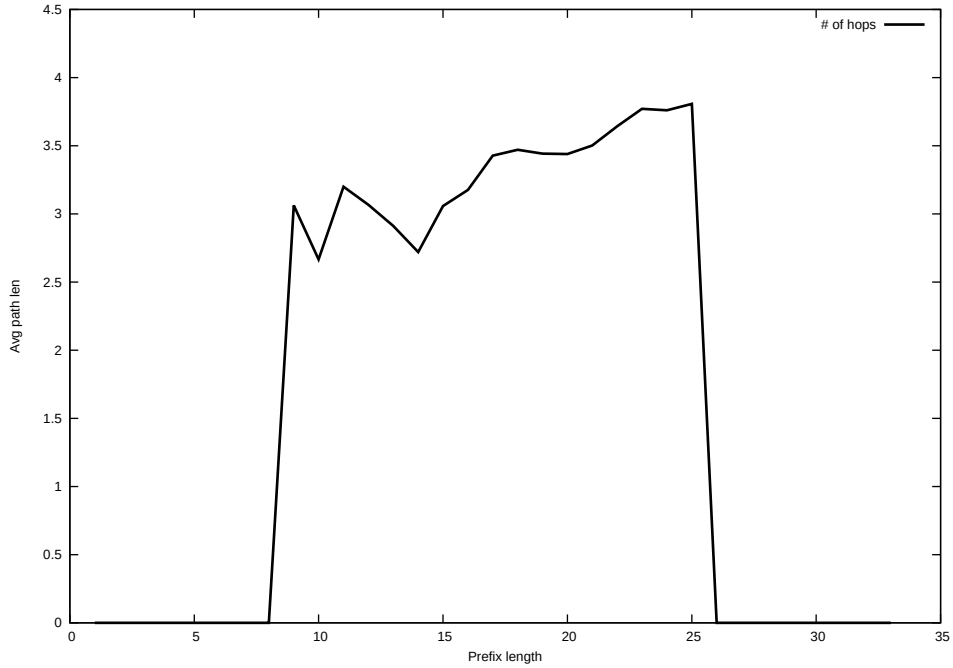
2014-10-16



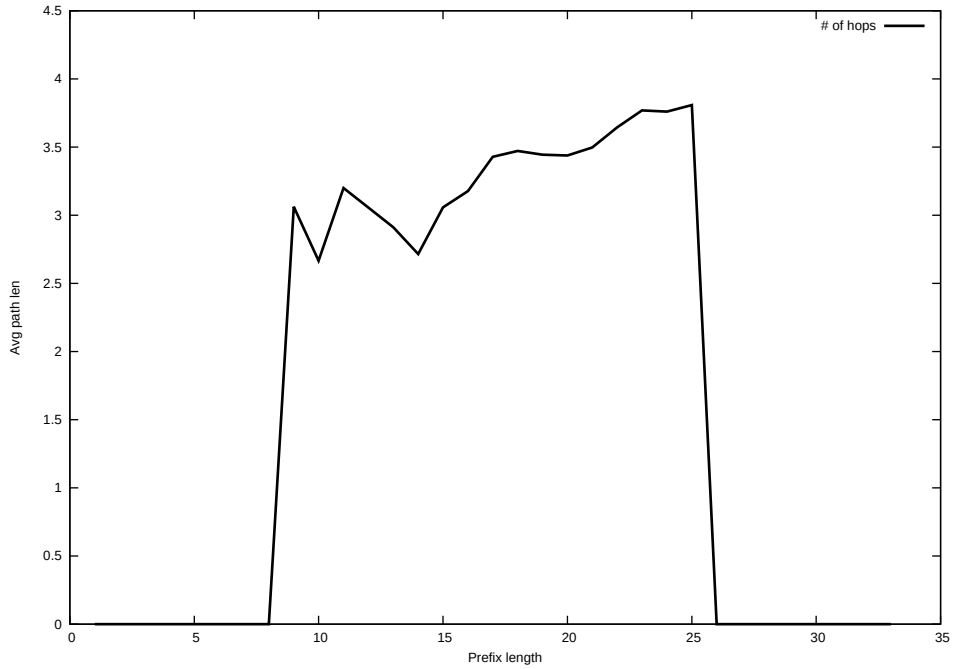
2014-10-17



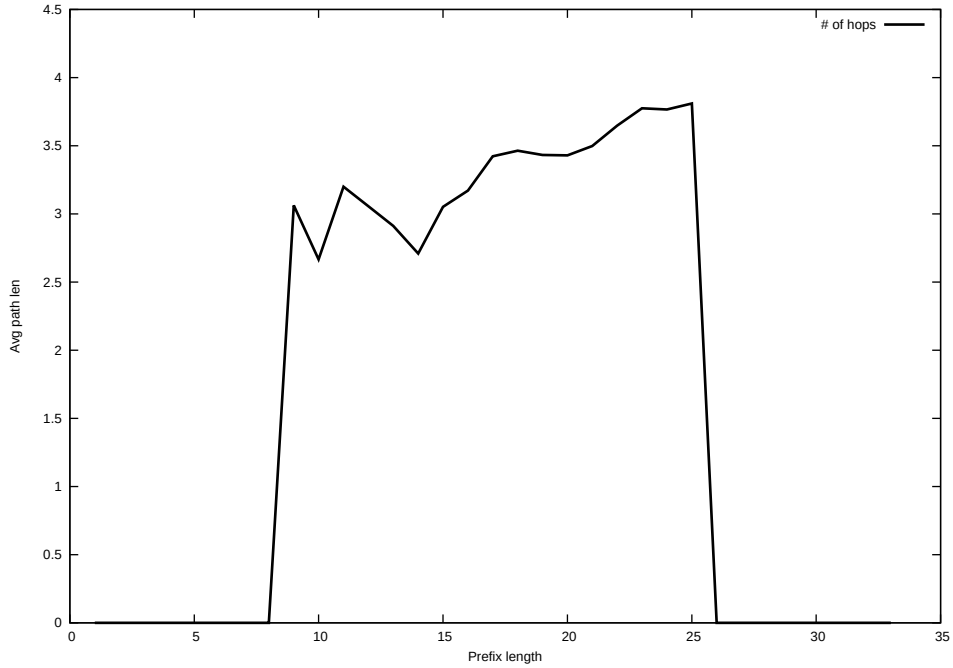
2014-10-18



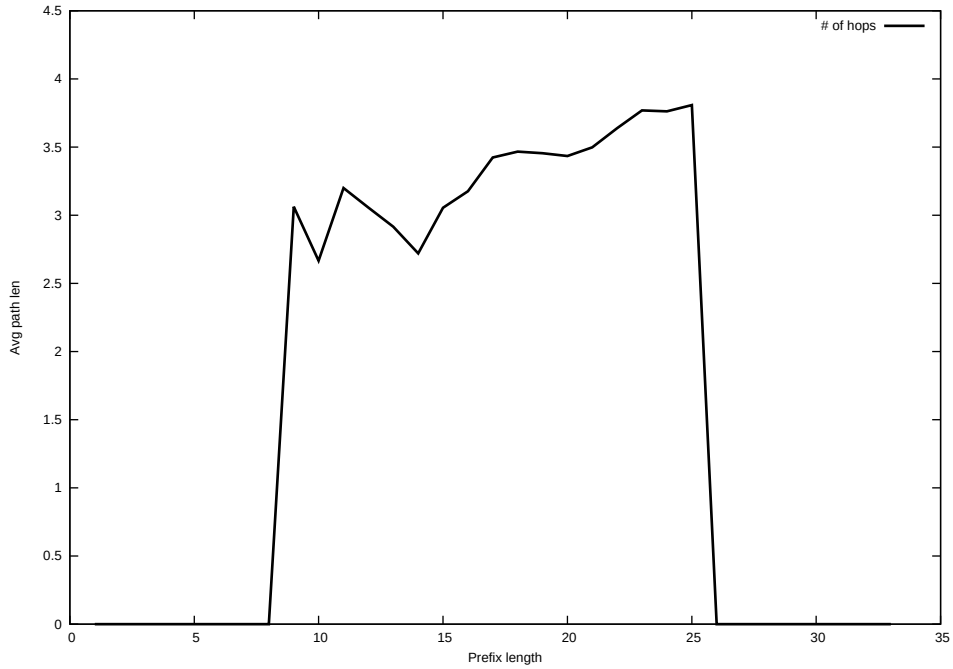
2014-10-19



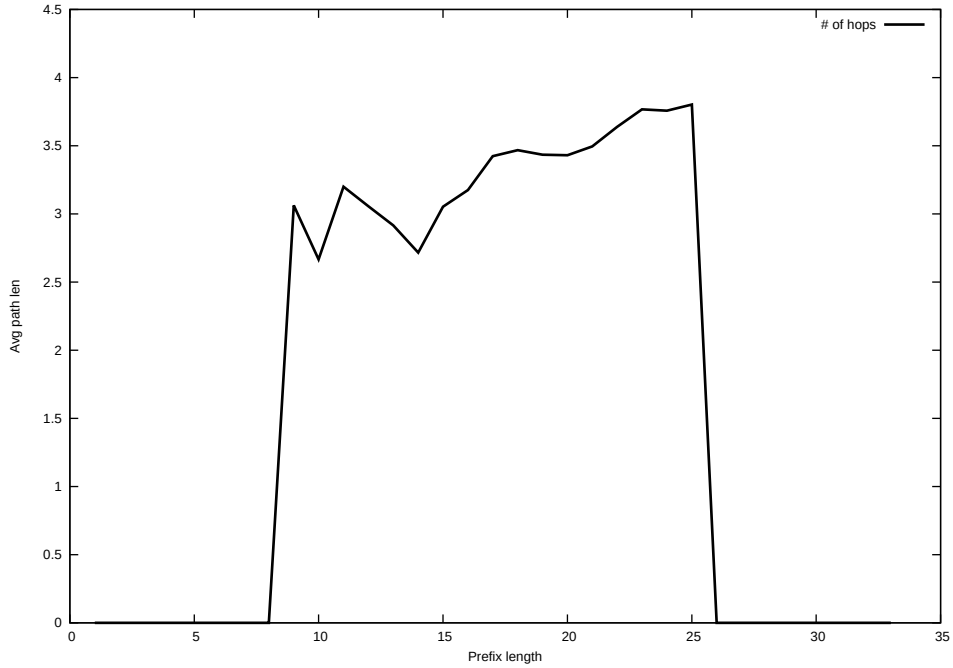
2014-10-20



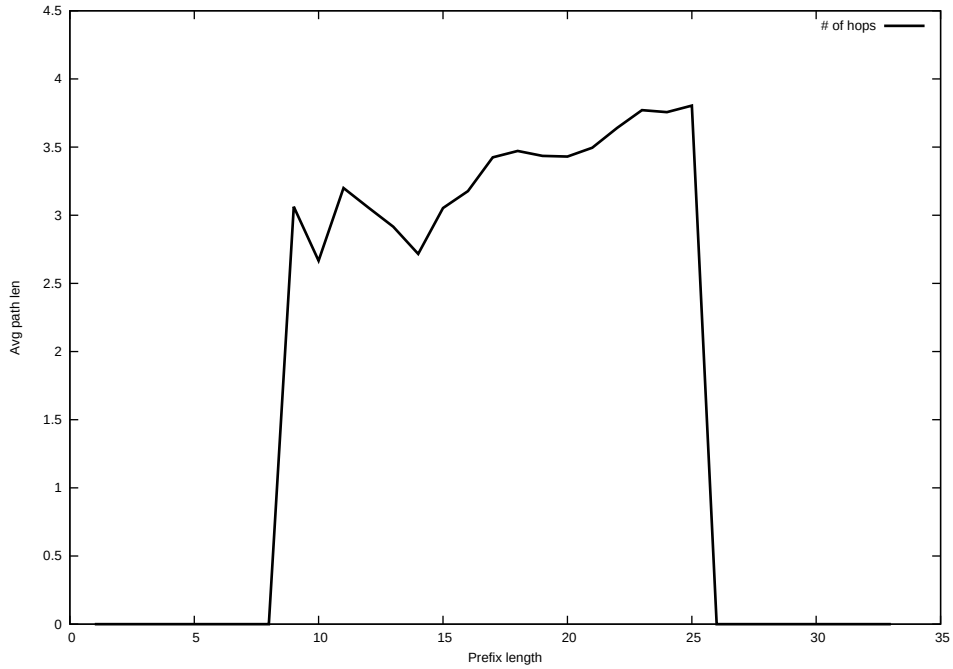
2014-10-21



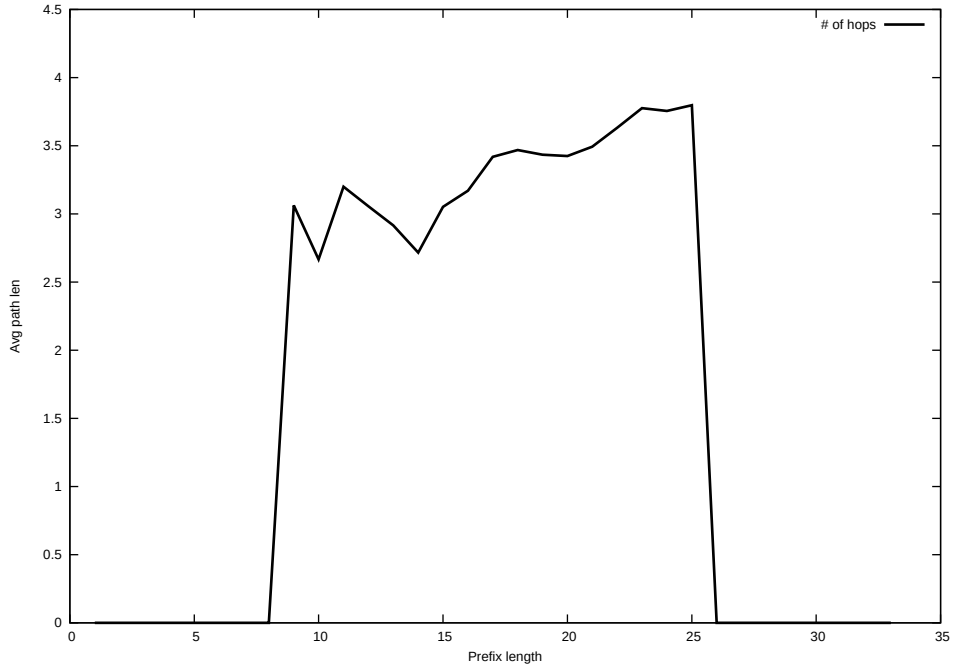
2014-10-22



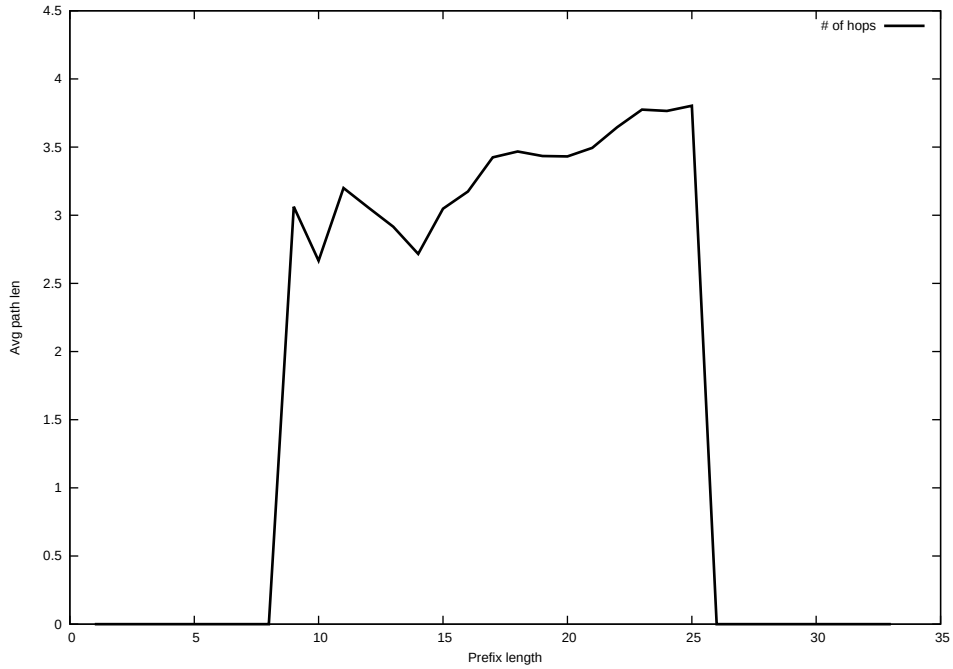
2014-10-23



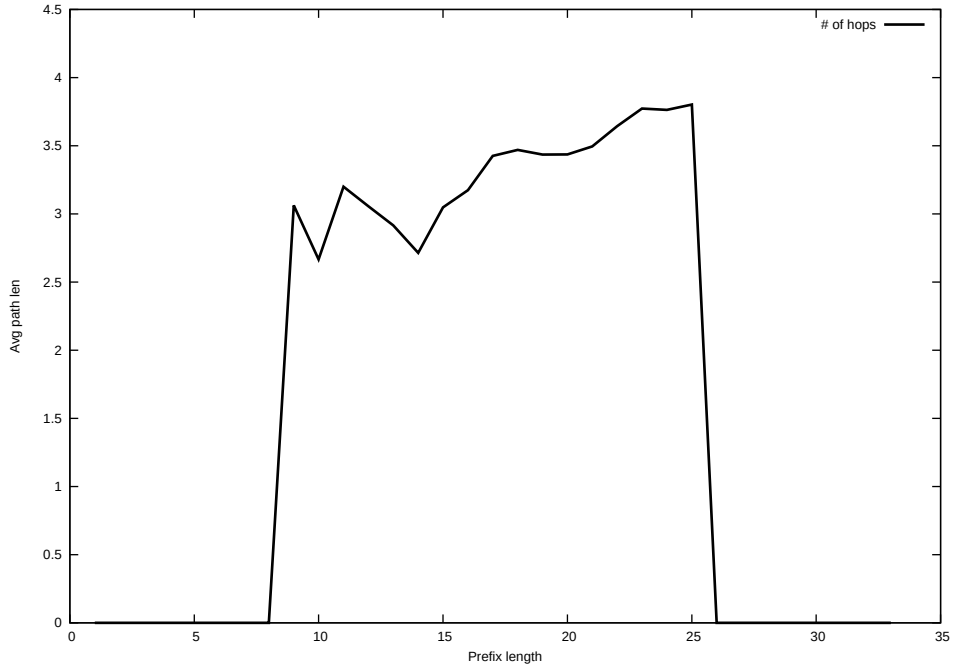
2014-10-24



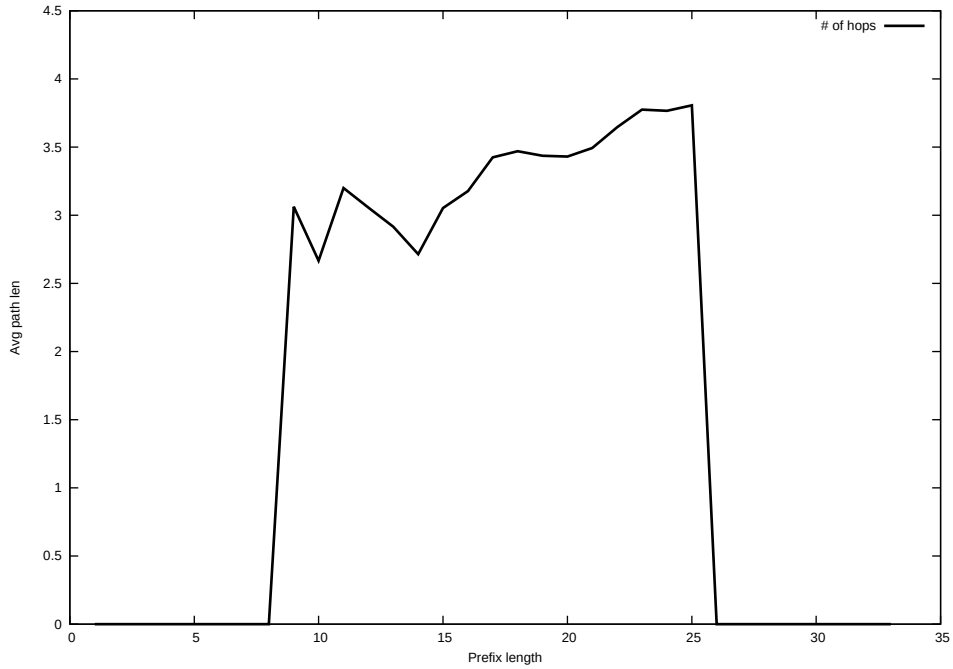
2014-10-25



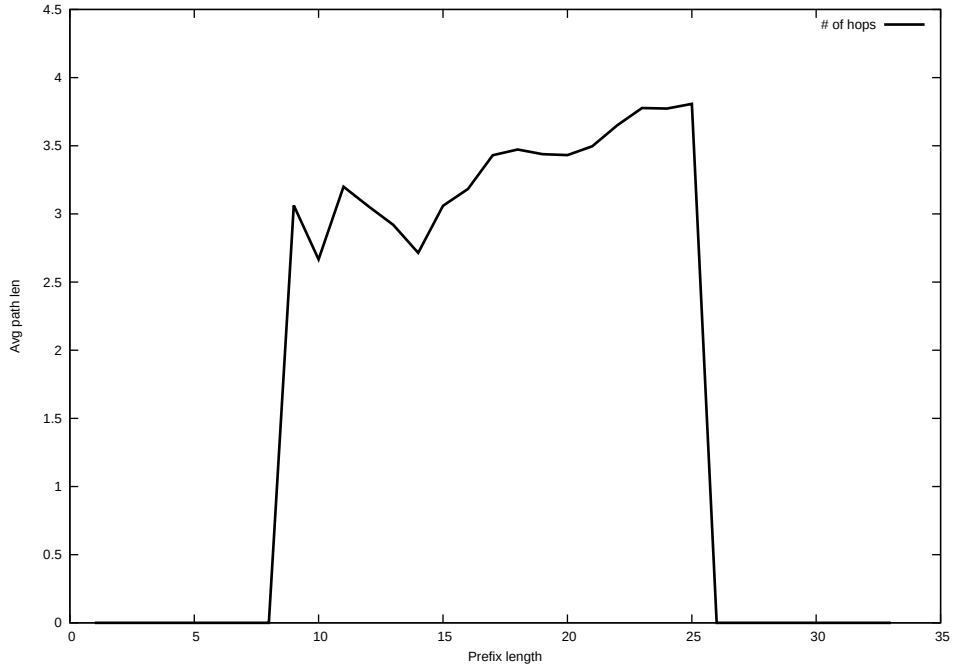
2014-10-26



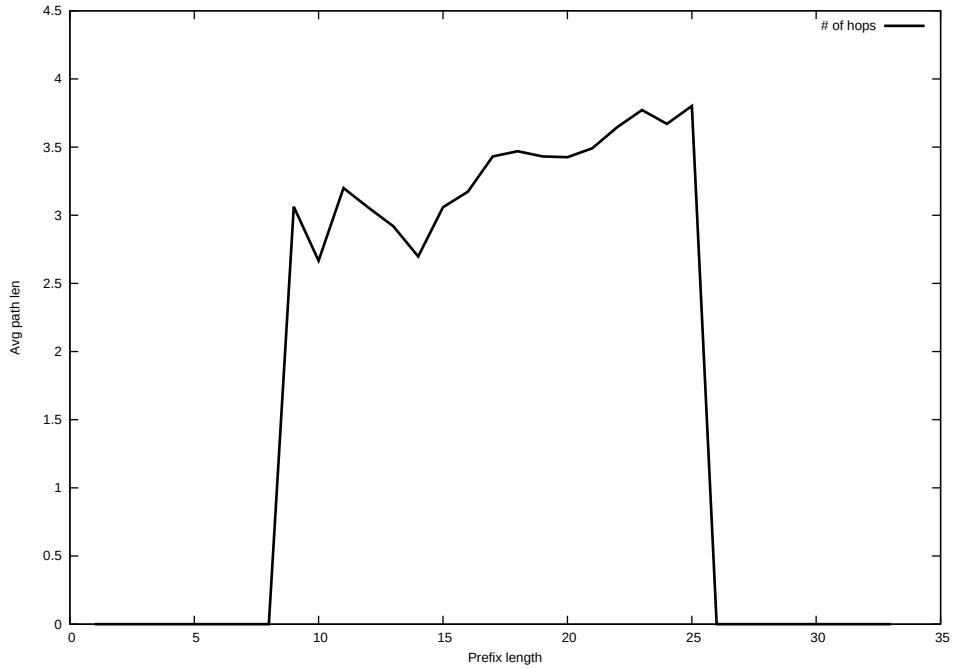
2014-10-27



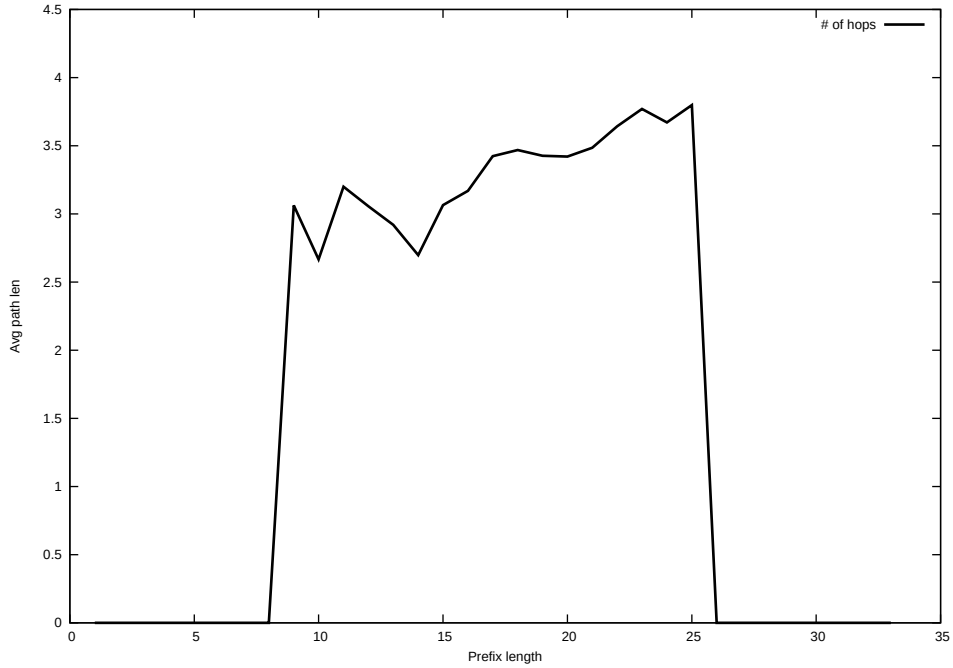
2014-10-28



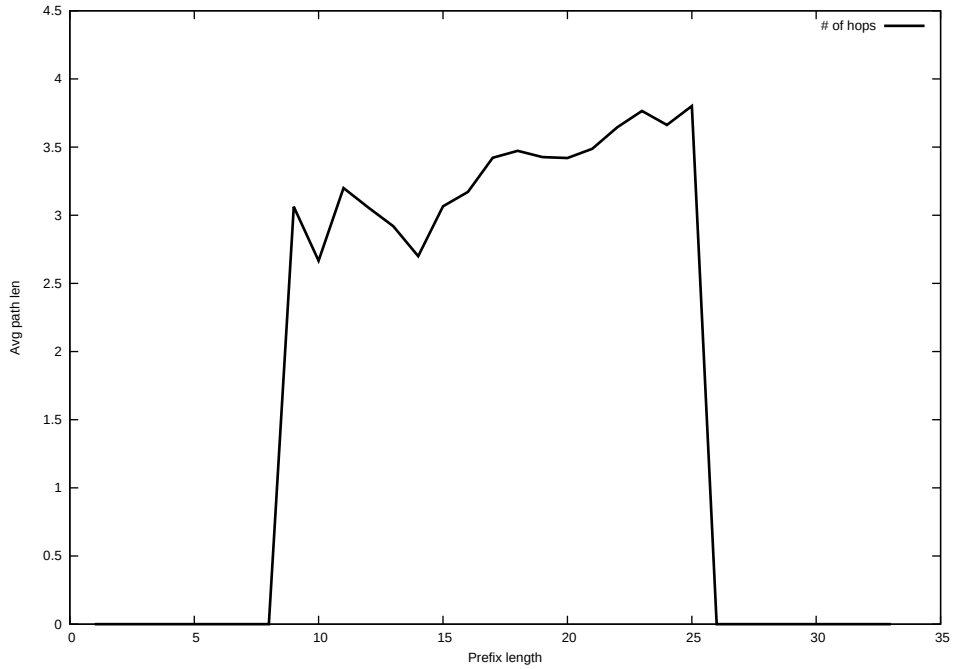
2014-10-29



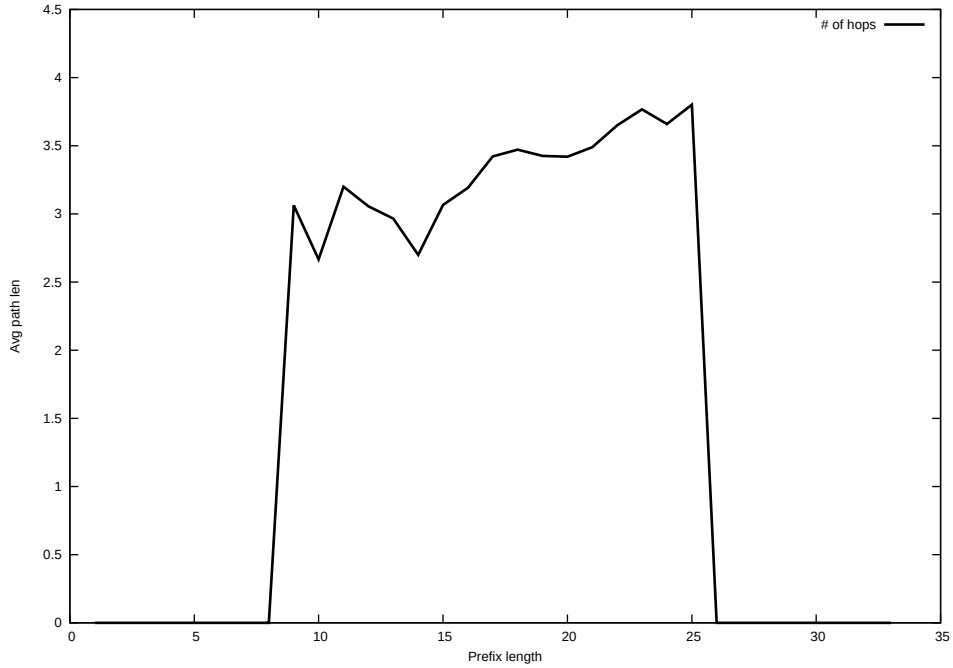
2014-10-30



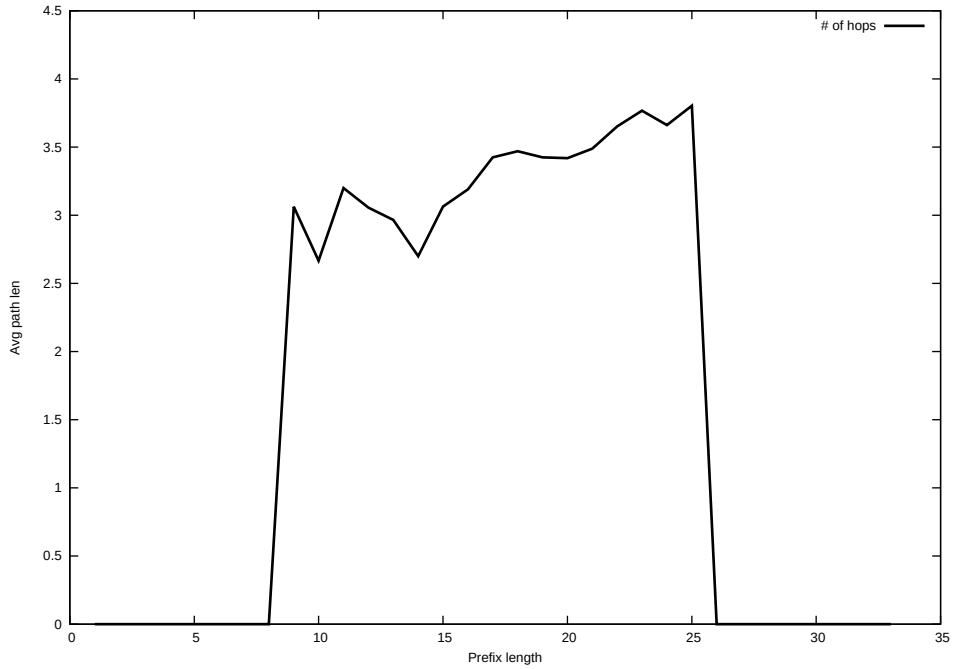
2014-10-31



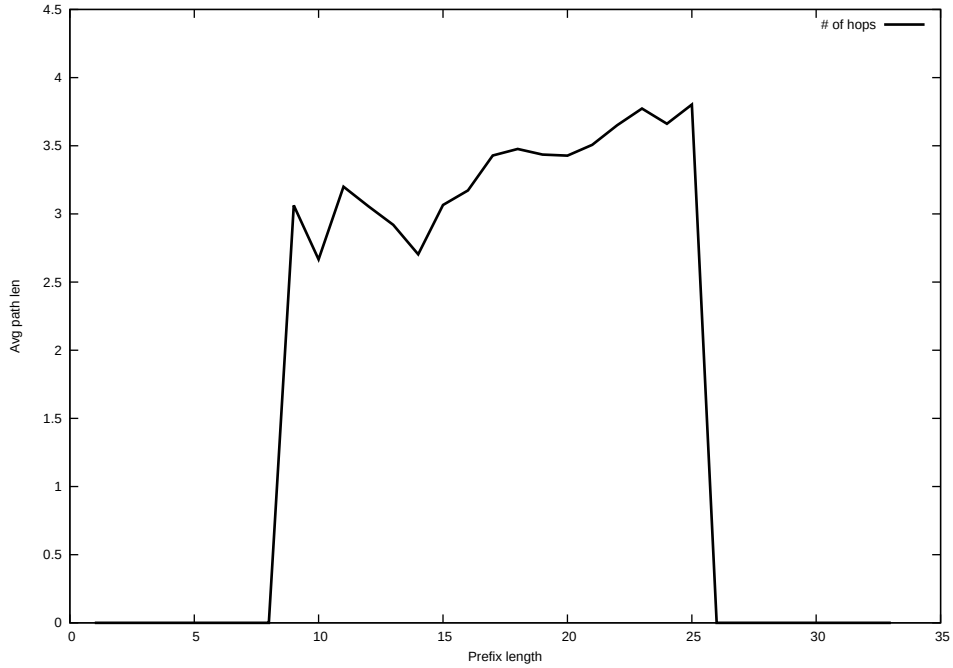
2014-11-02



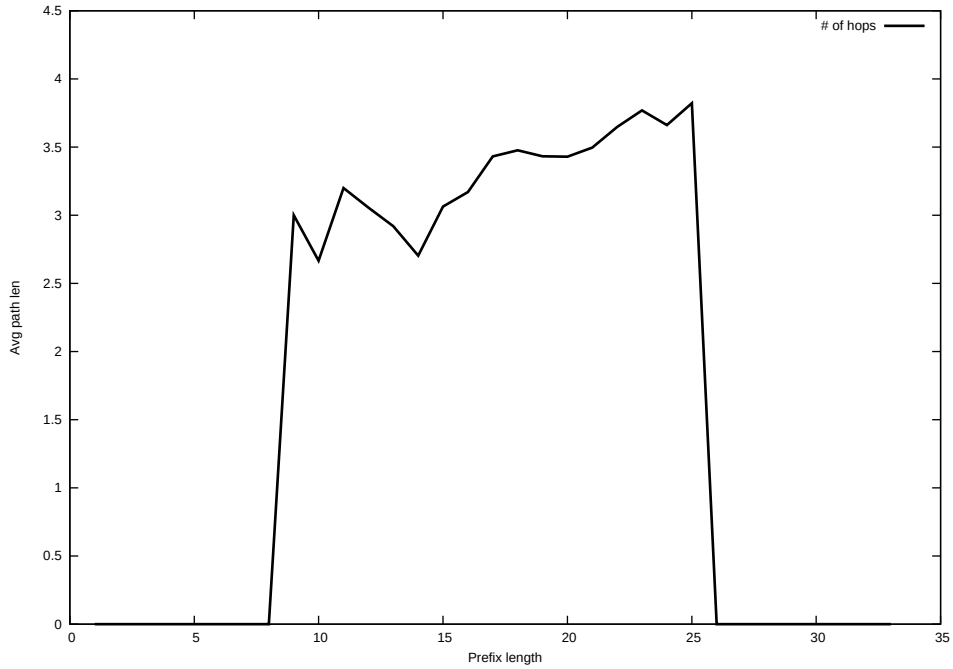
2014-11-03



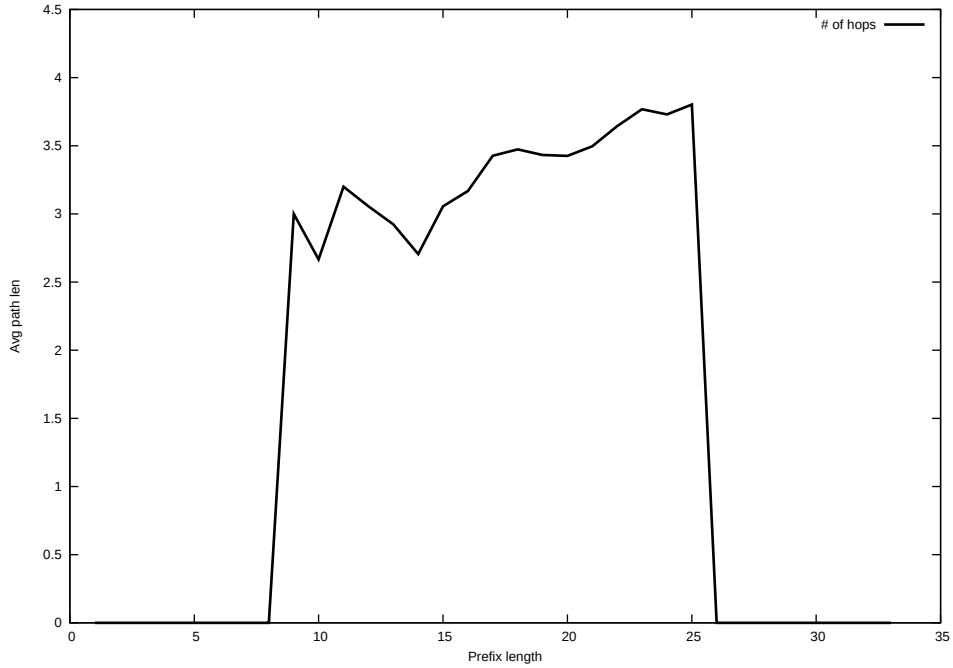
2014-11-04



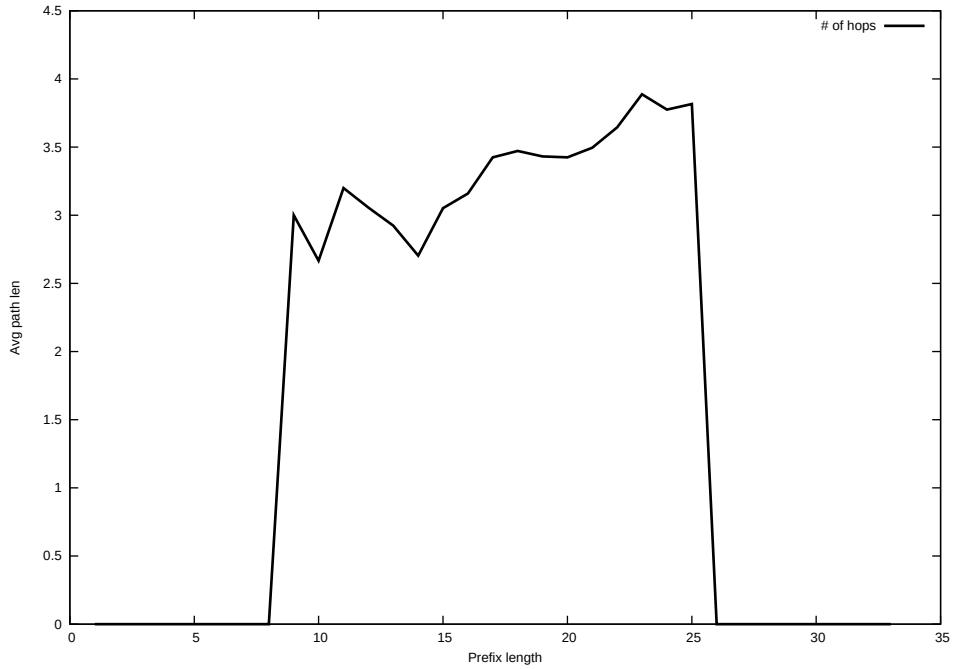
2014-11-05



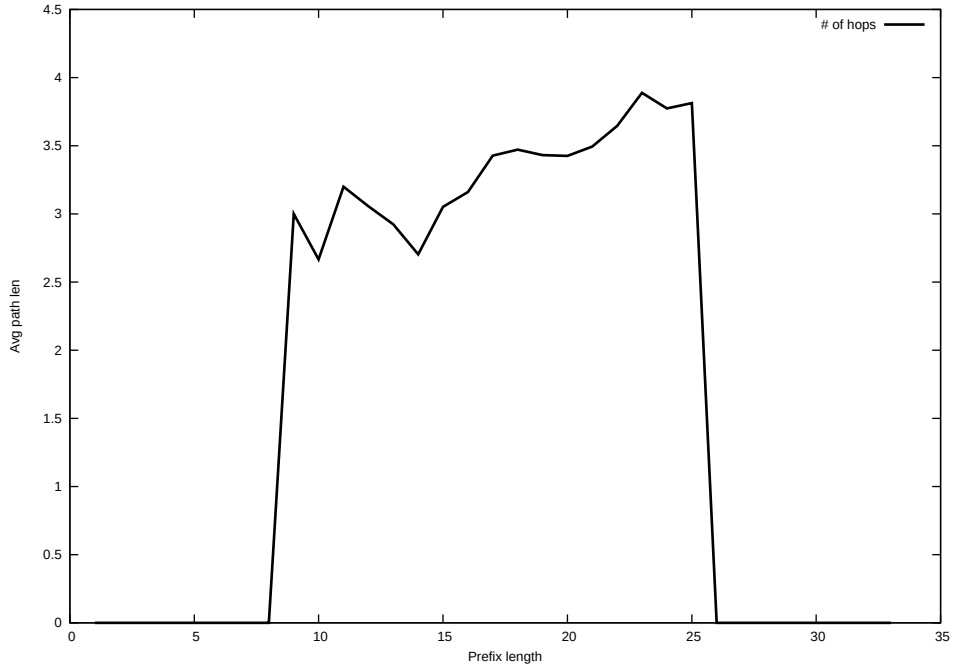
2014-11-06



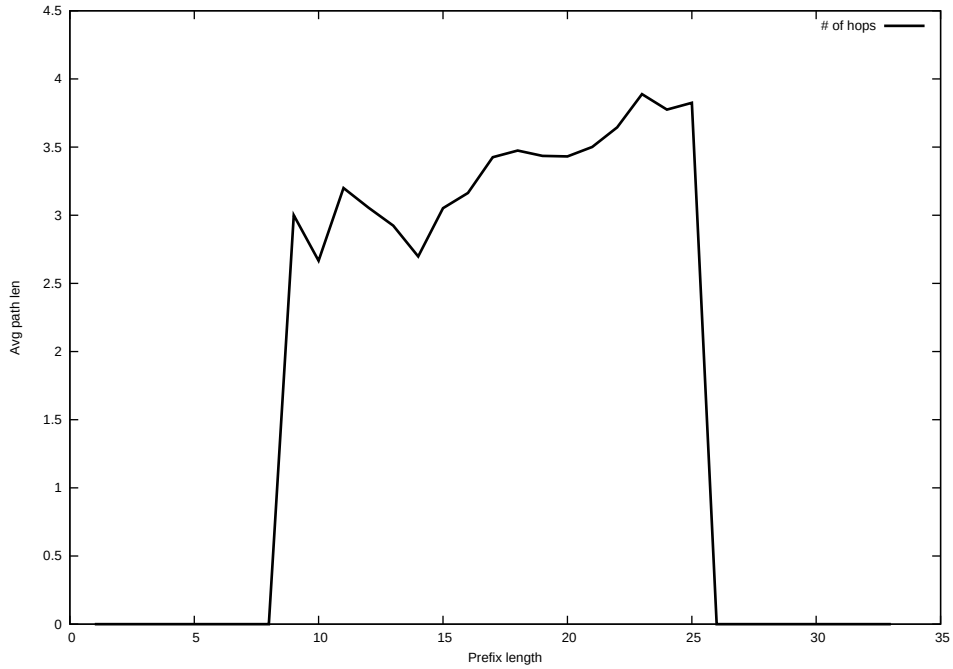
2014-11-07



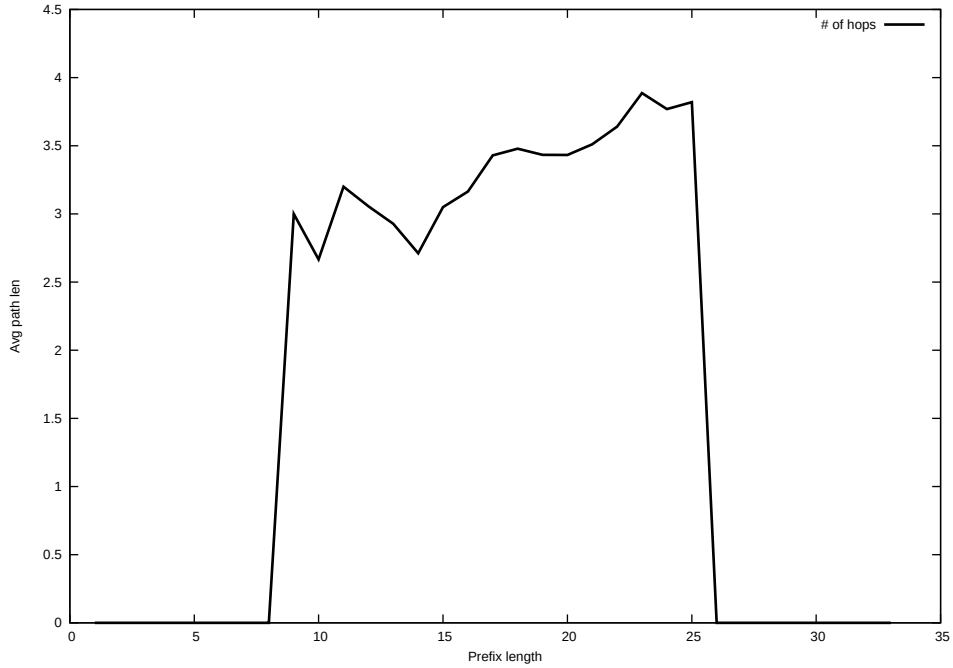
2014-11-08



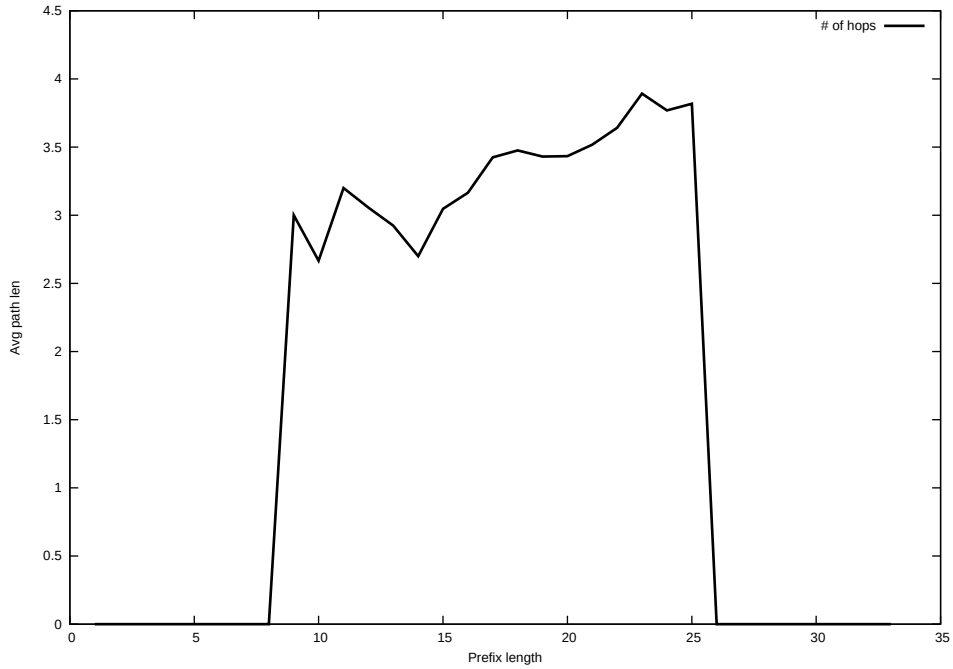
2014-11-09



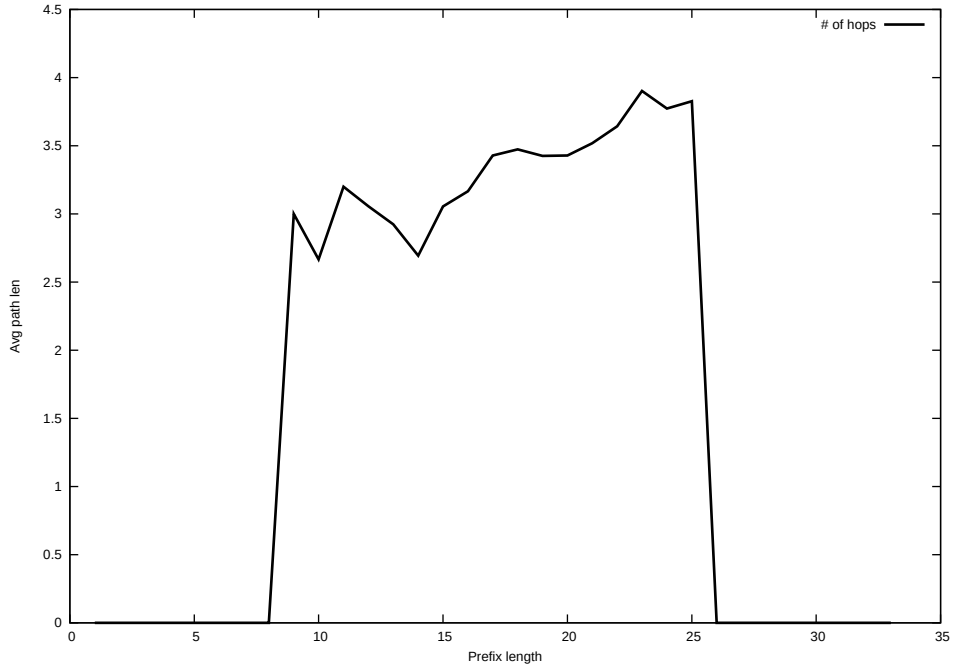
2014-11-10



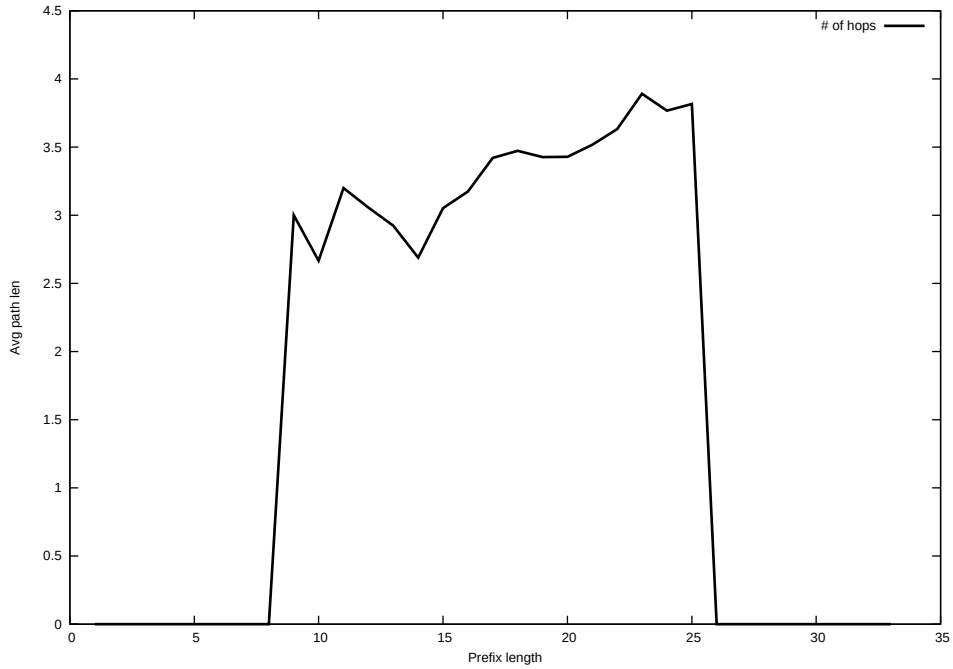
2014-11-11



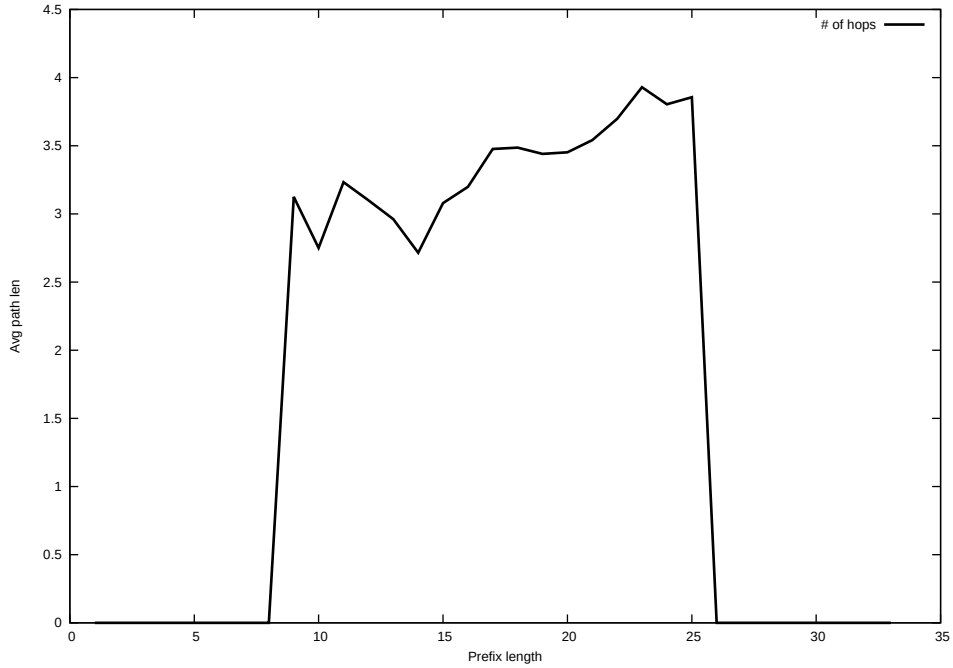
2014-11-12



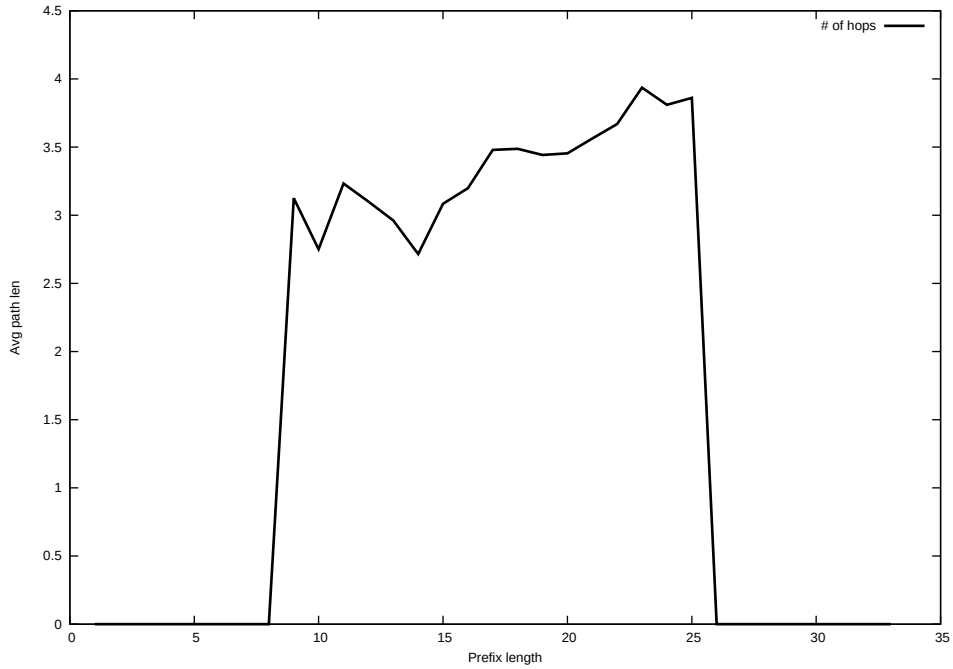
2014-11-13



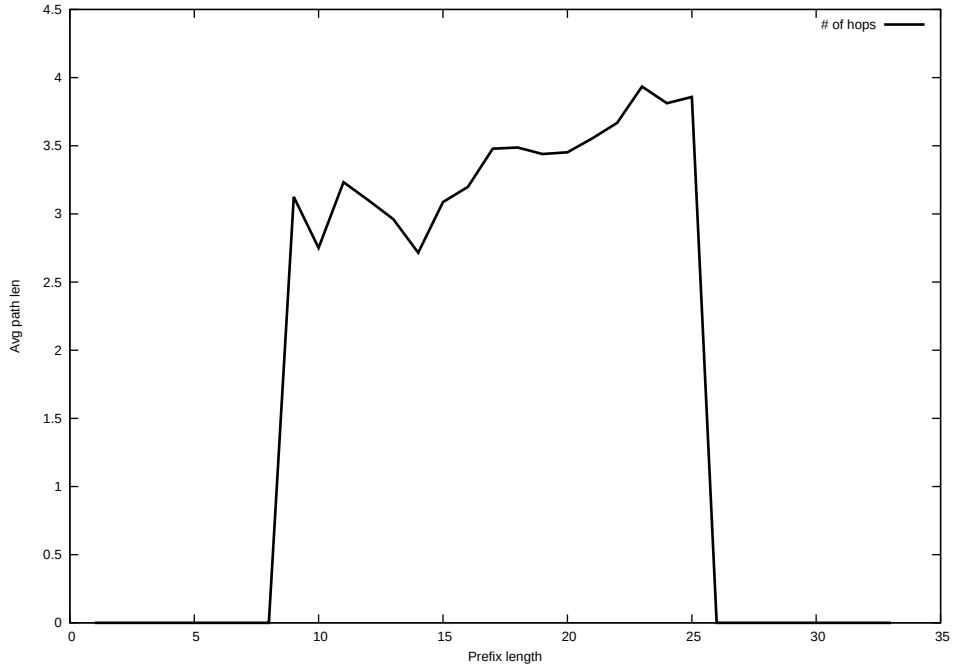
2014-11-14



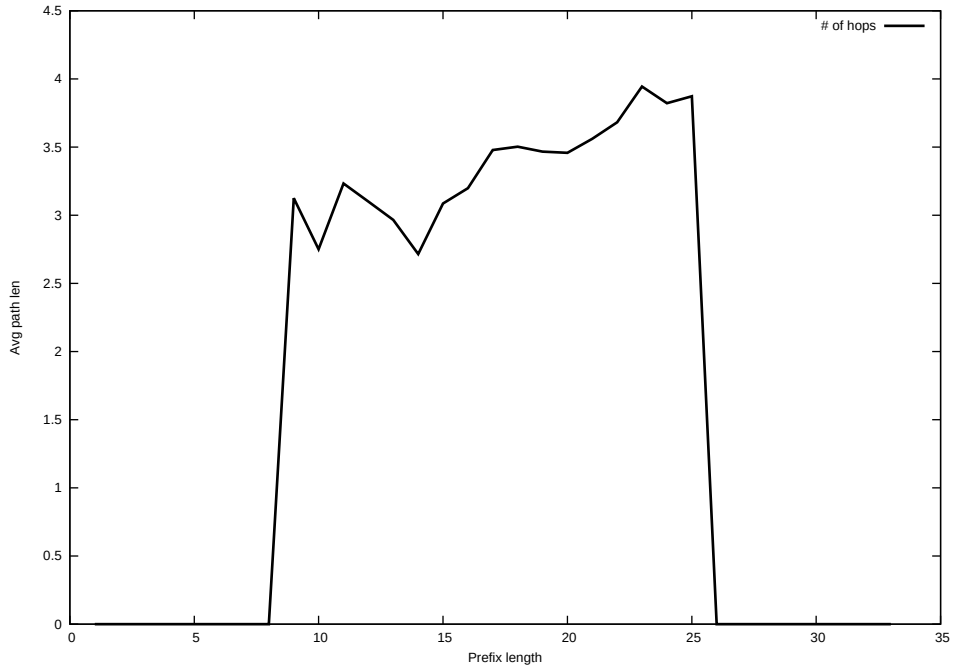
2014-11-15



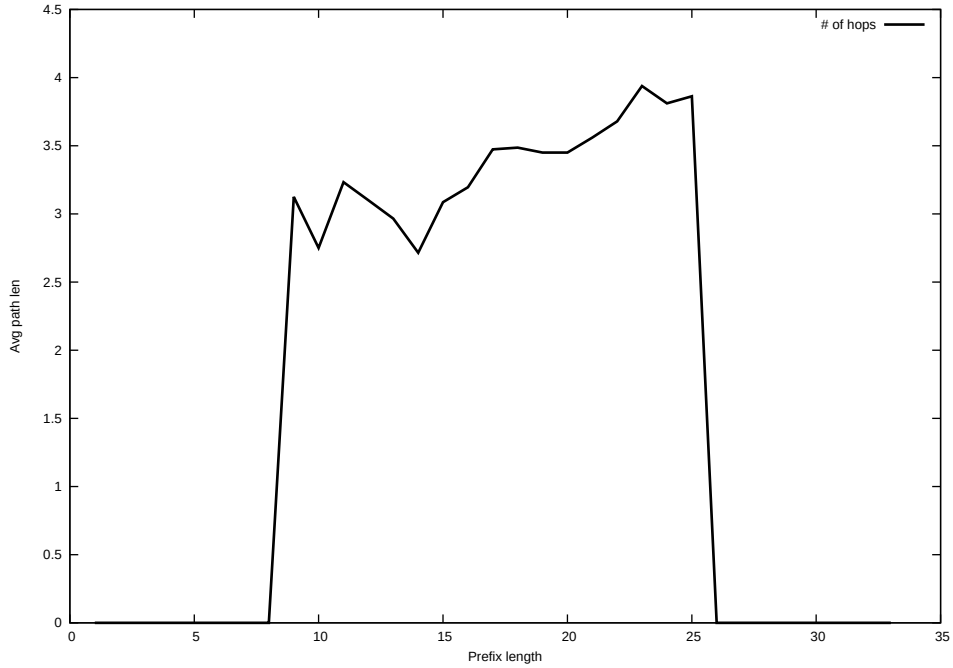
2014-11-16



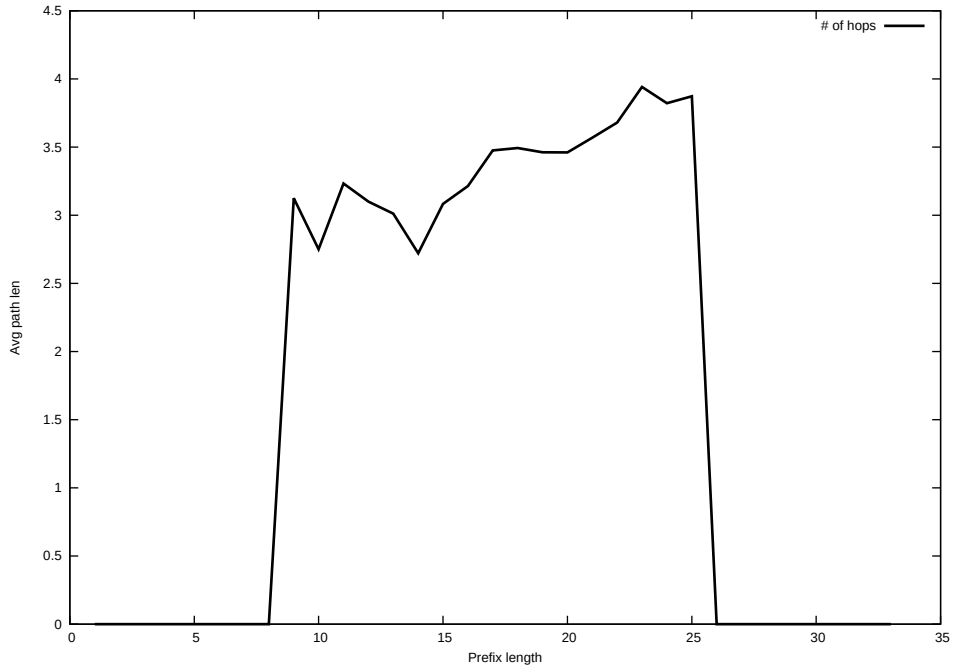
2014-11-17



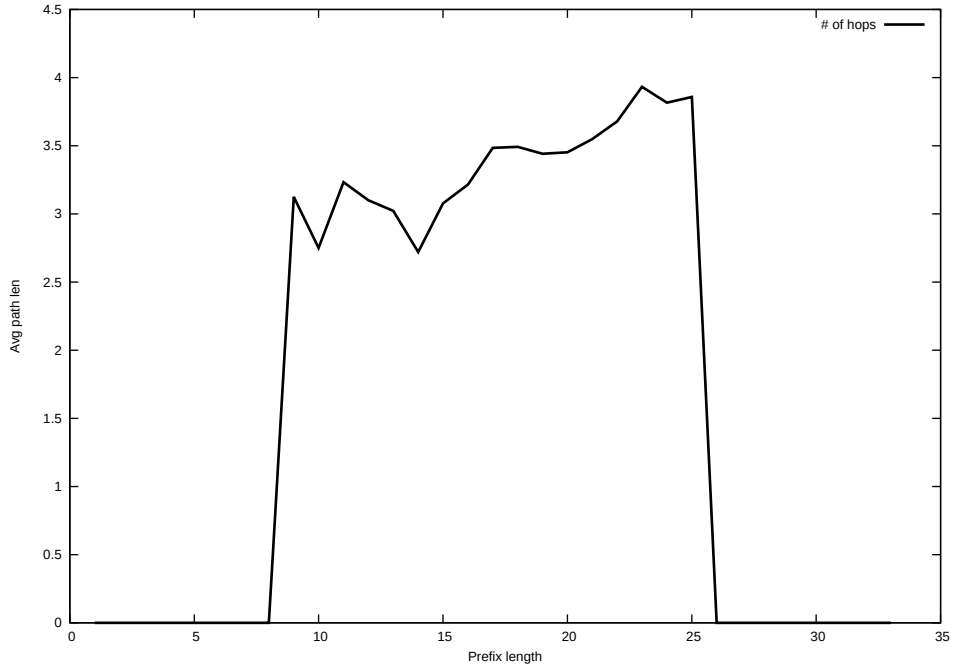
2014-11-18



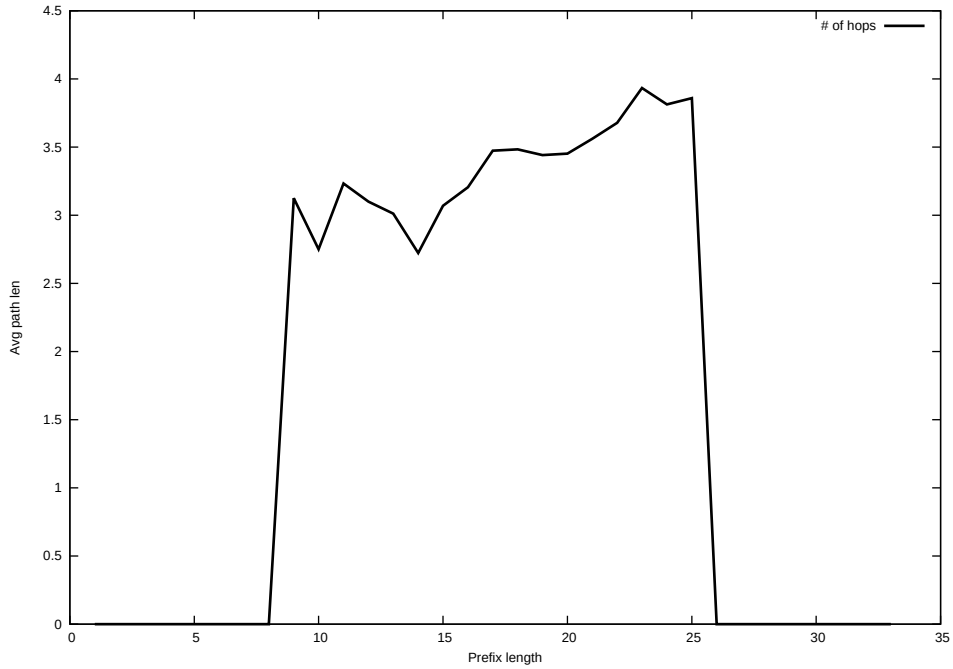
2014-11-19



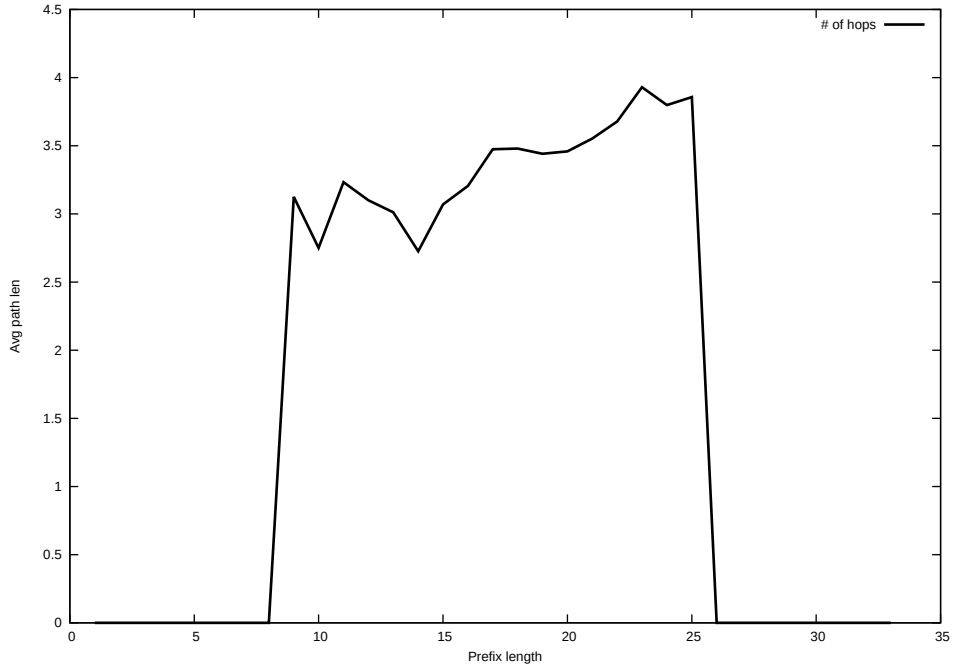
2014-11-20



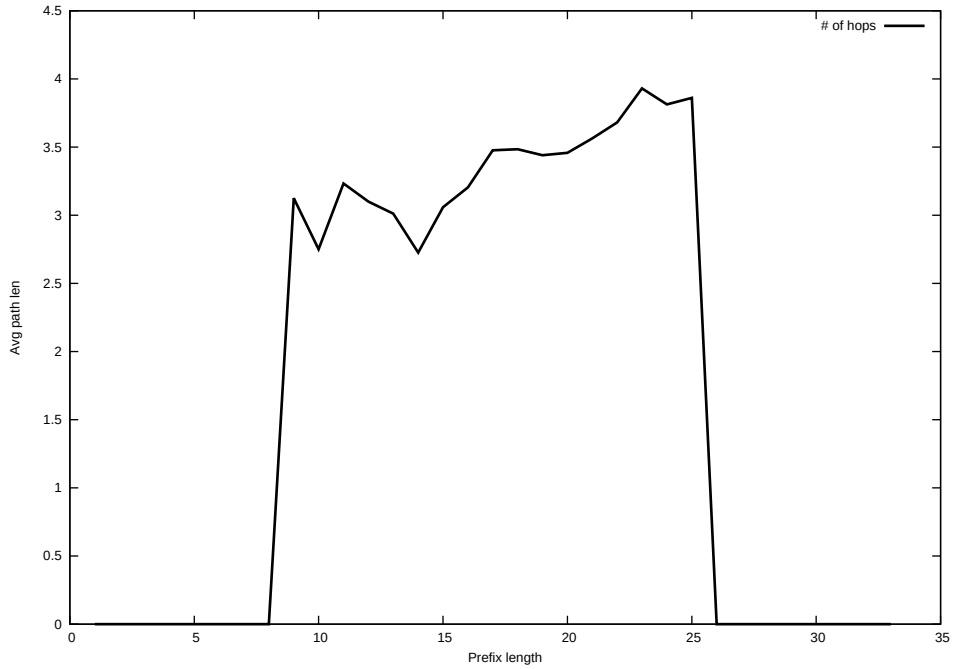
2014-11-21



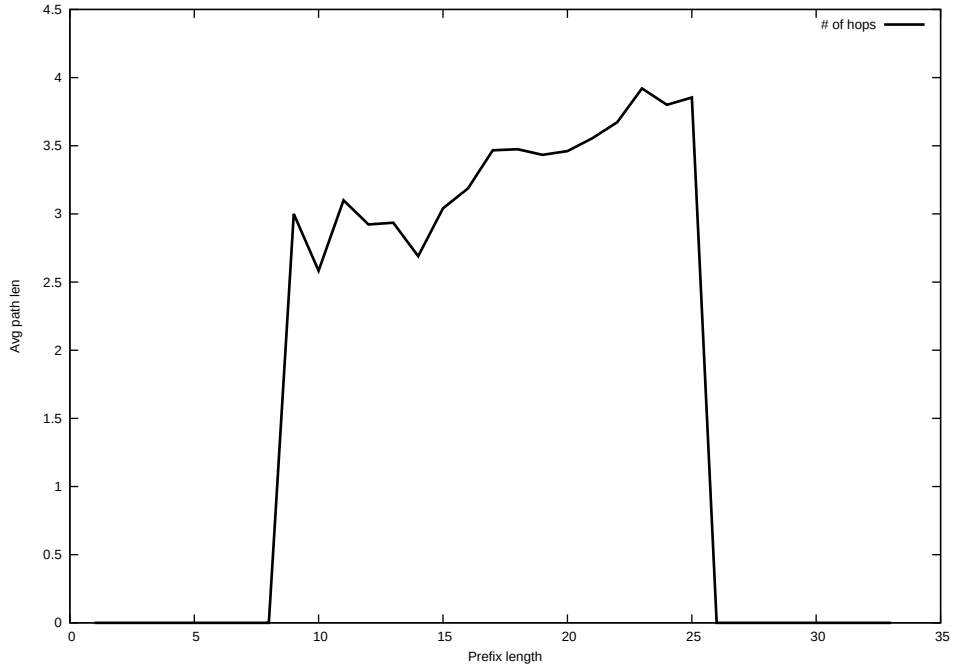
2014-11-22



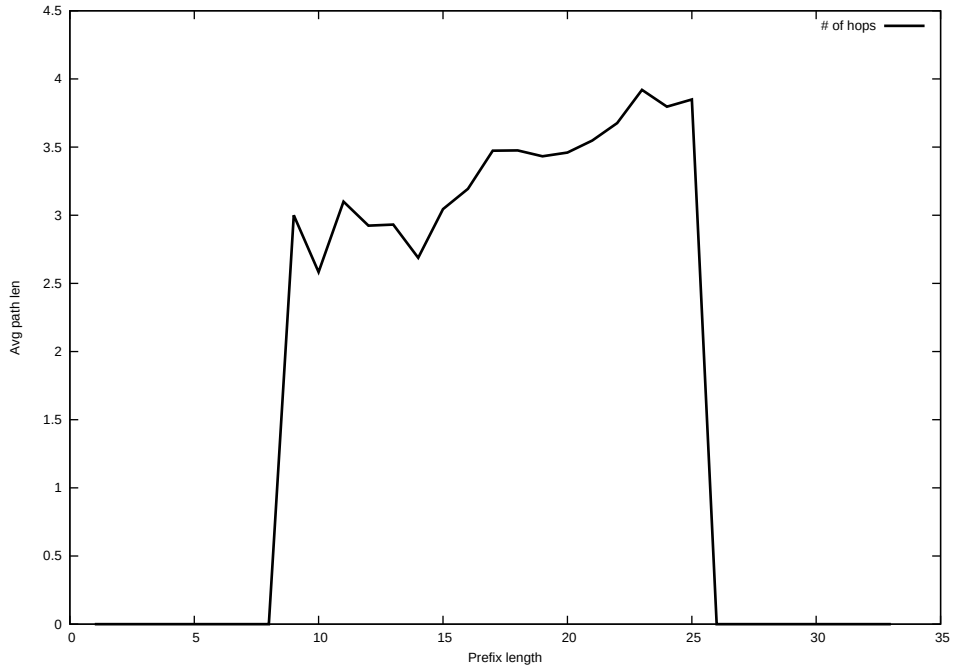
2014-11-23



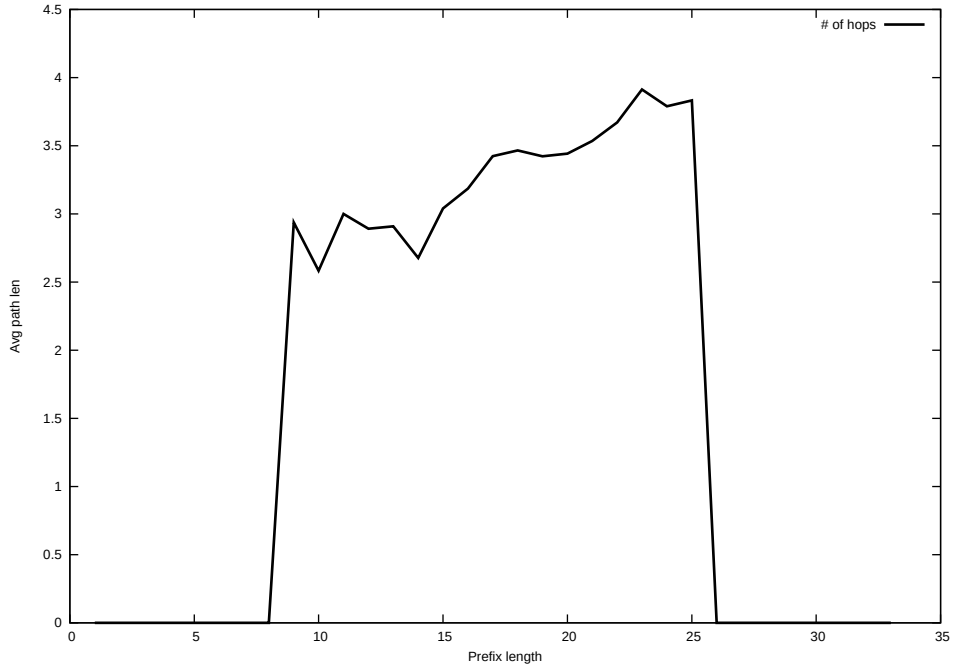
2014-11-24



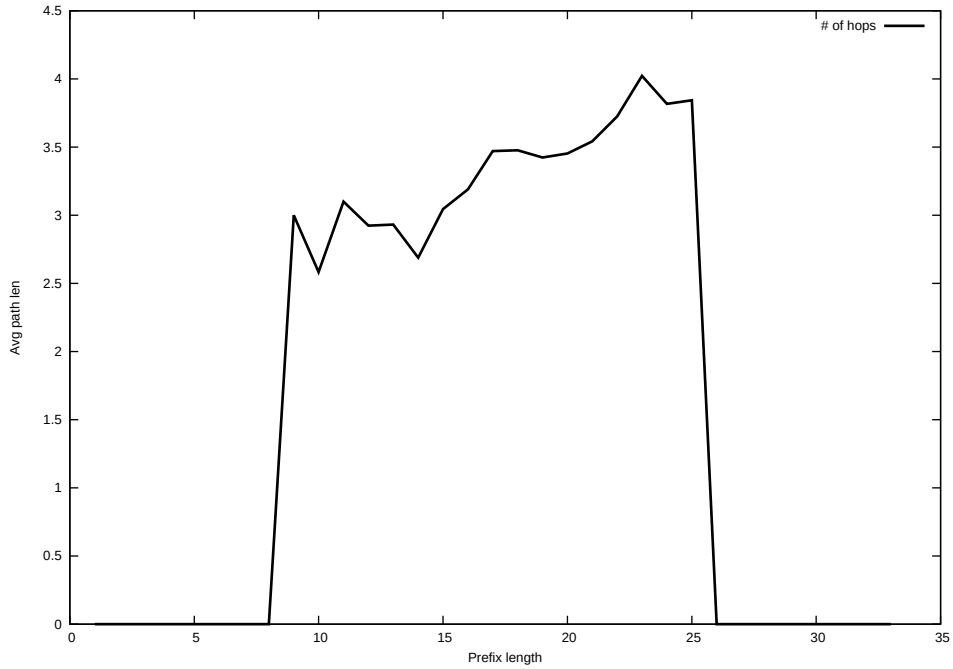
2014-11-25



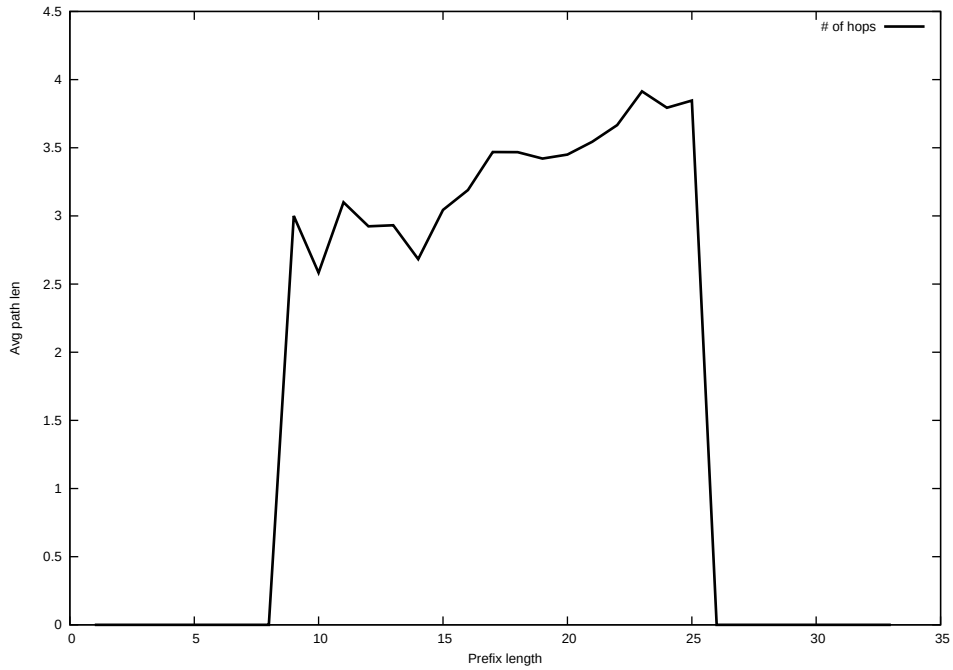
2014-11-26



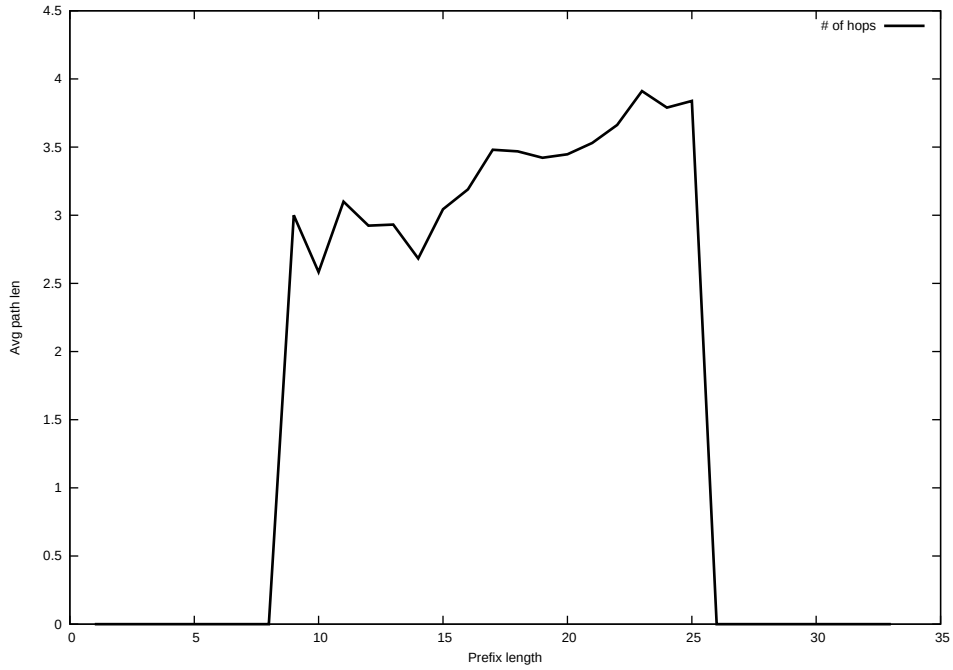
2014-11-27



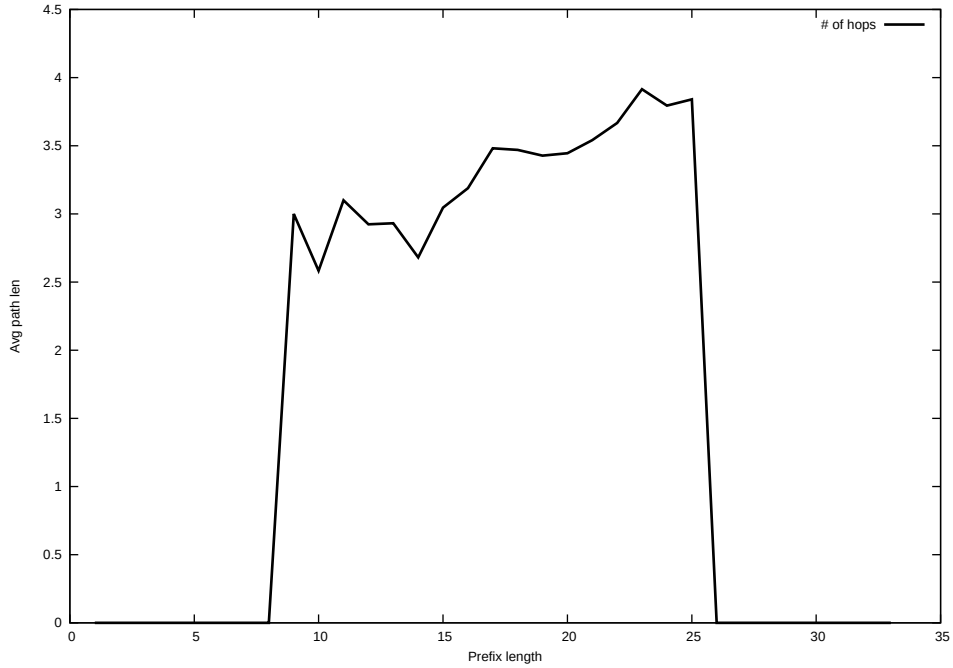
2014-11-28



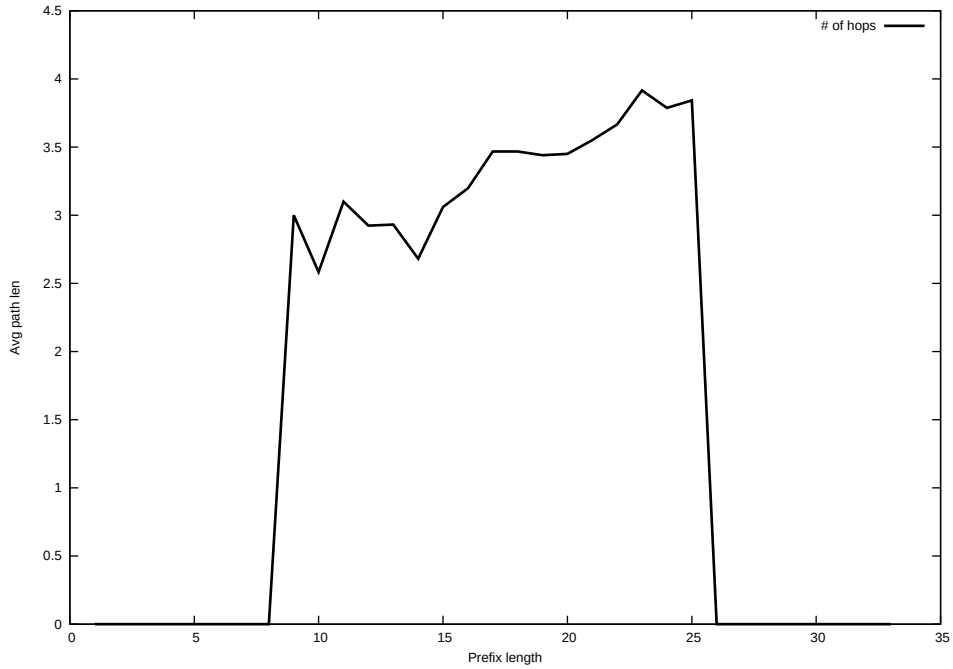
2014-11-29



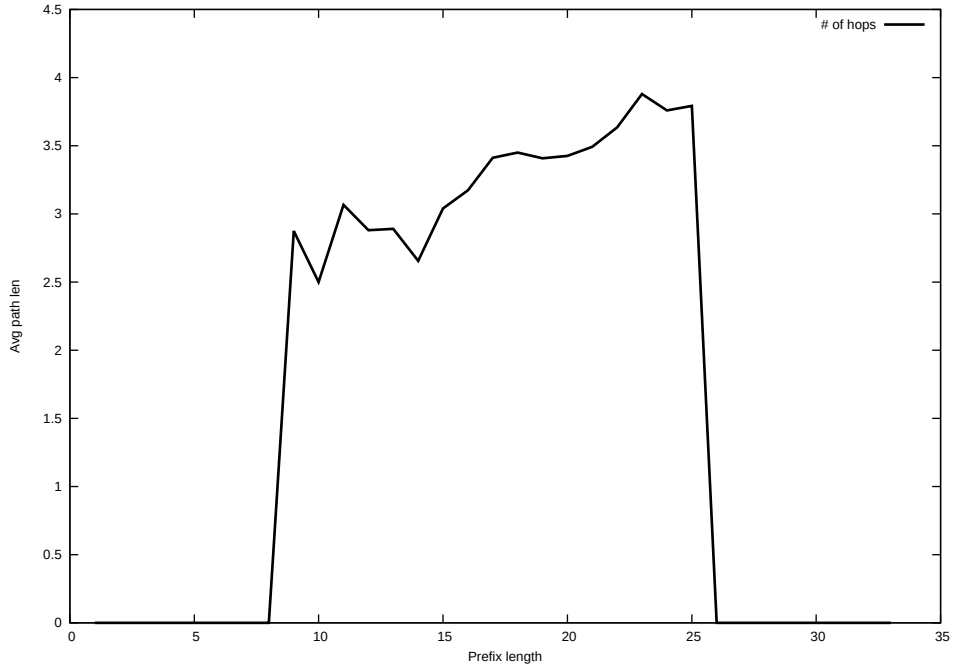
2014-11-30



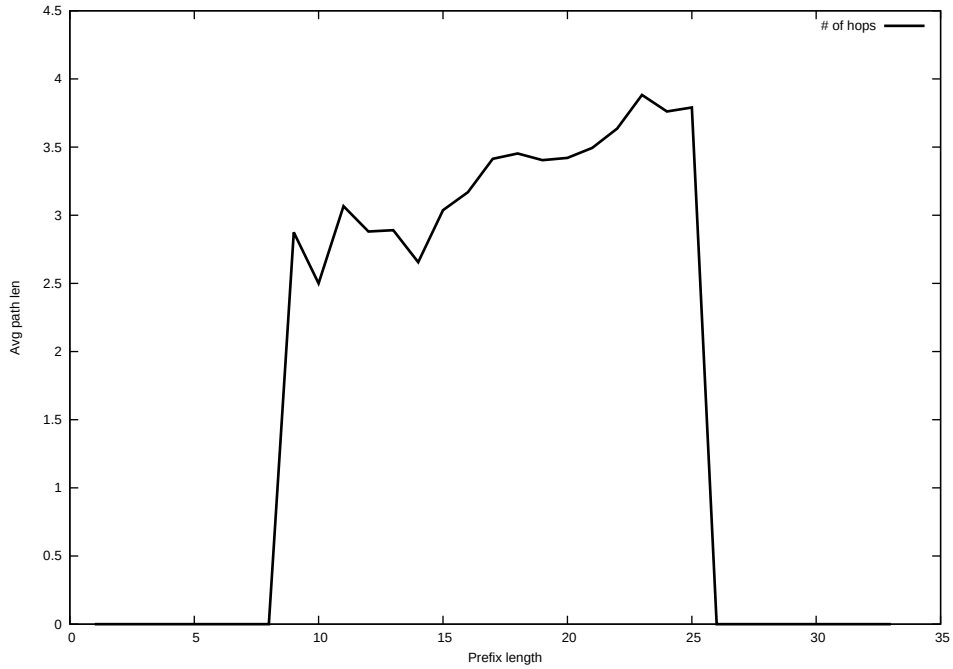
2014-12-01



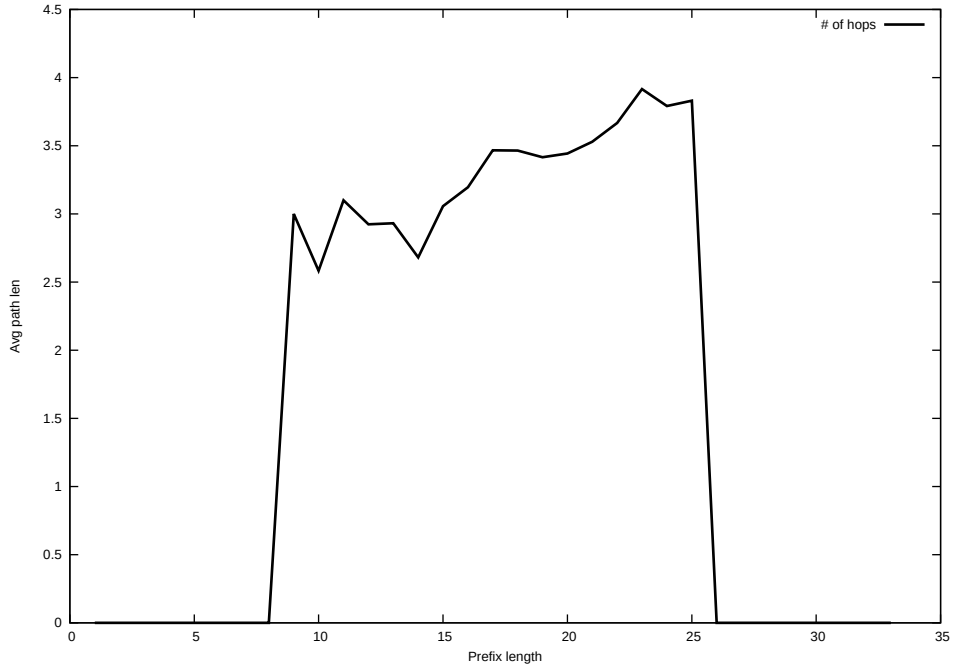
2014-12-02



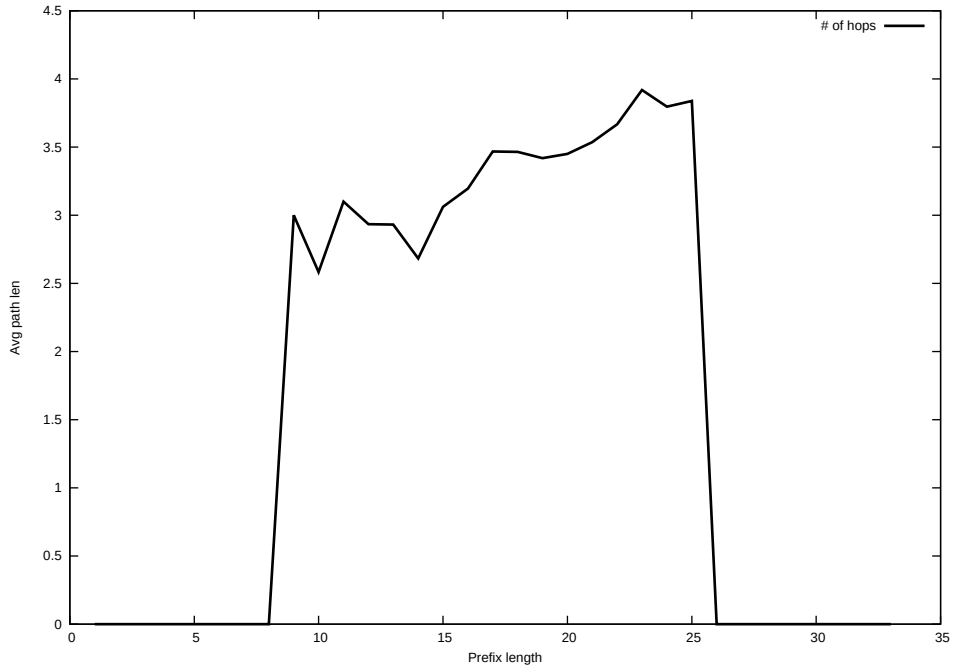
2014-12-03



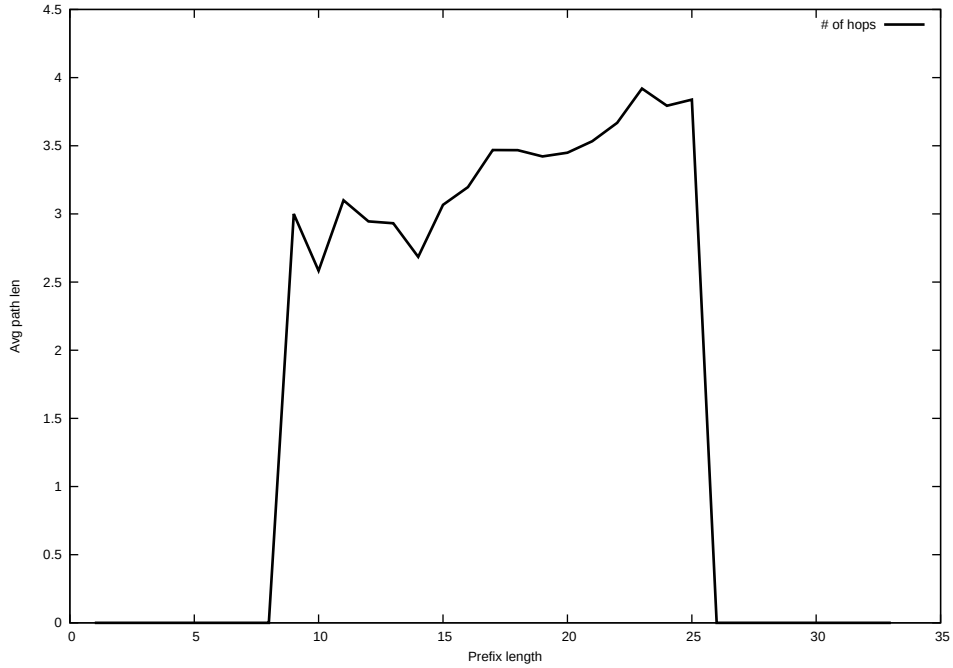
2014-12-04



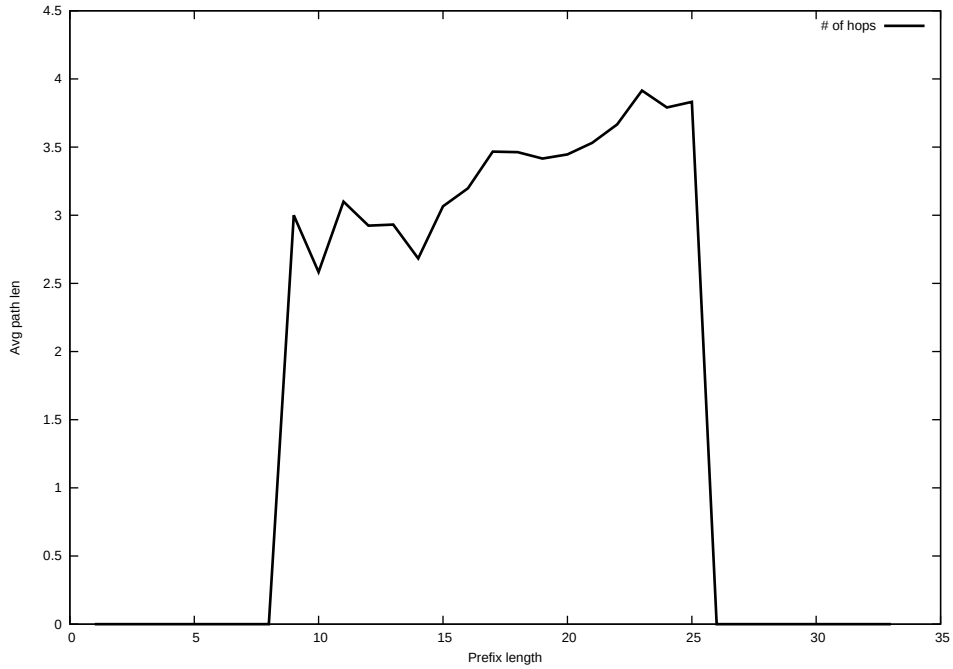
2014-12-05



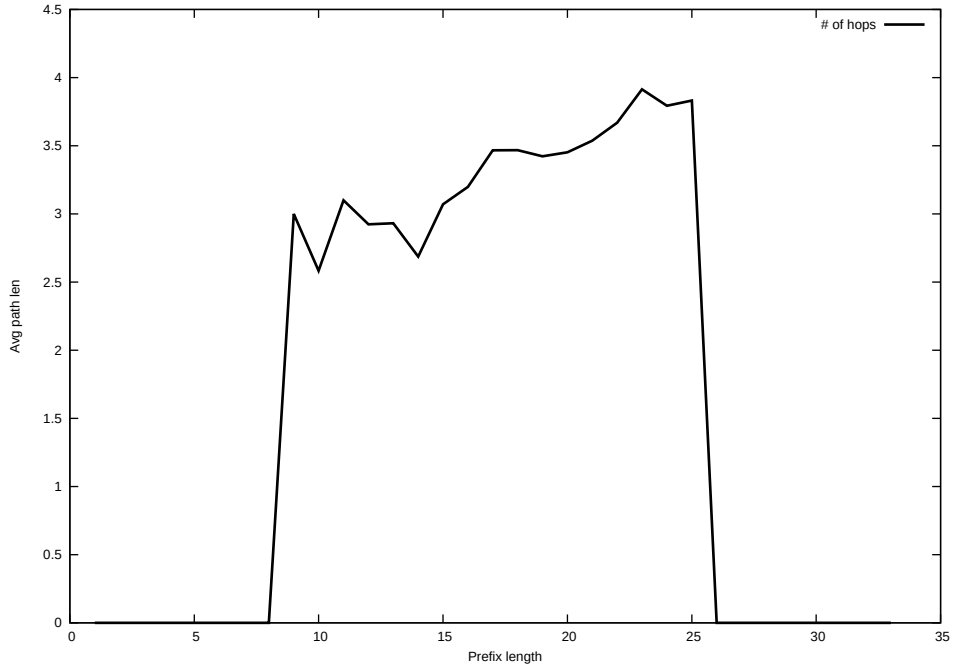
2014-12-06



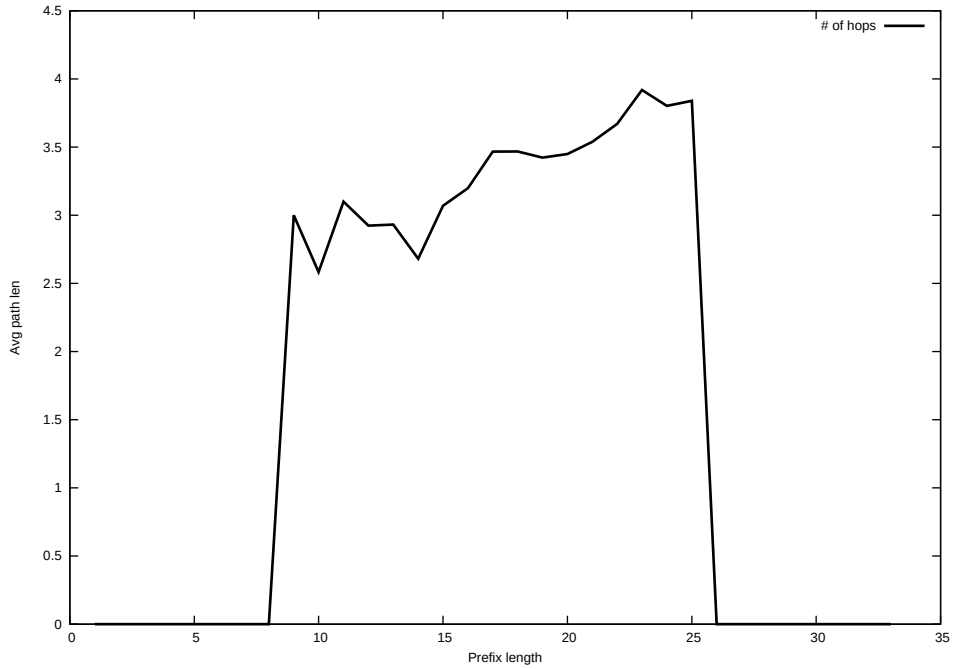
2014-12-07



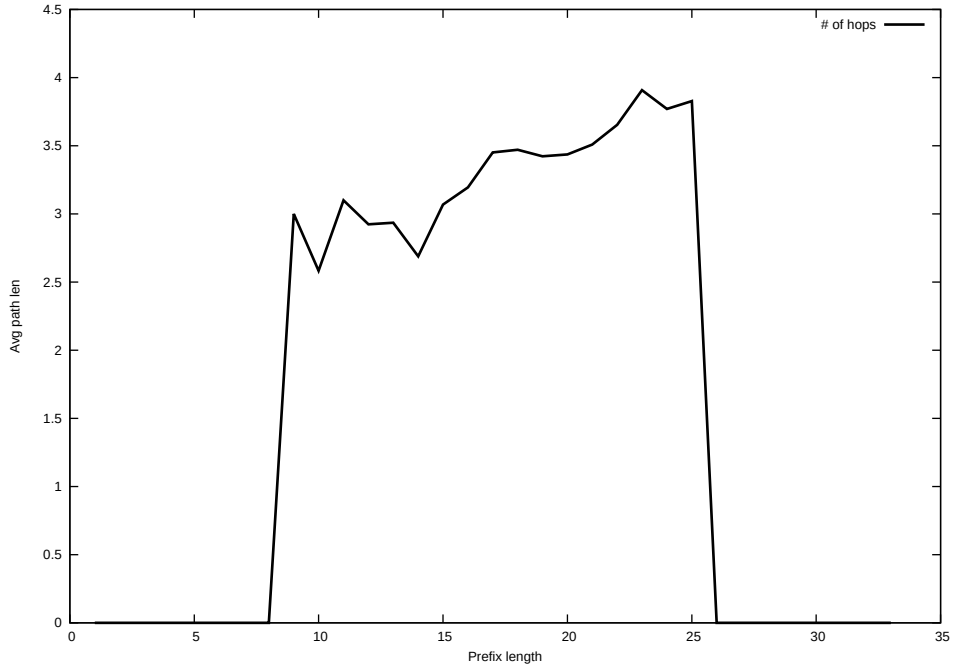
2014-12-08



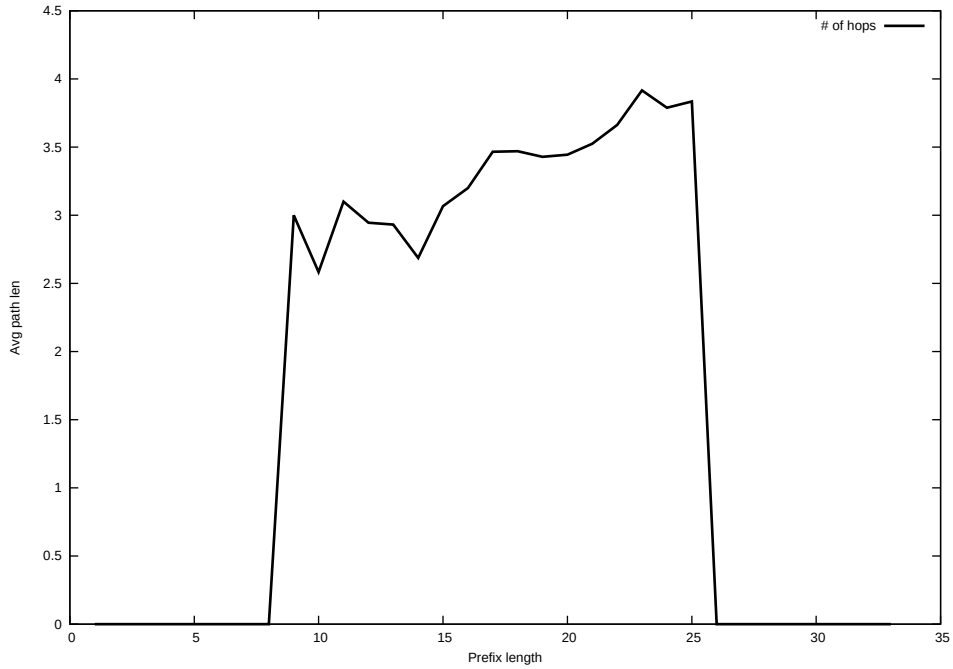
2014-12-09



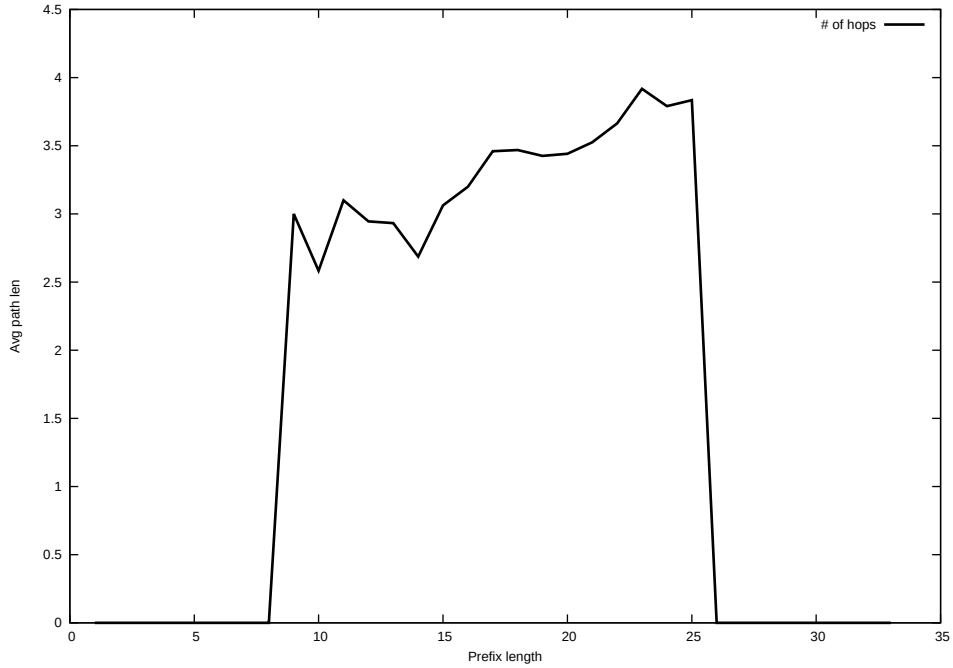
2014-12-10



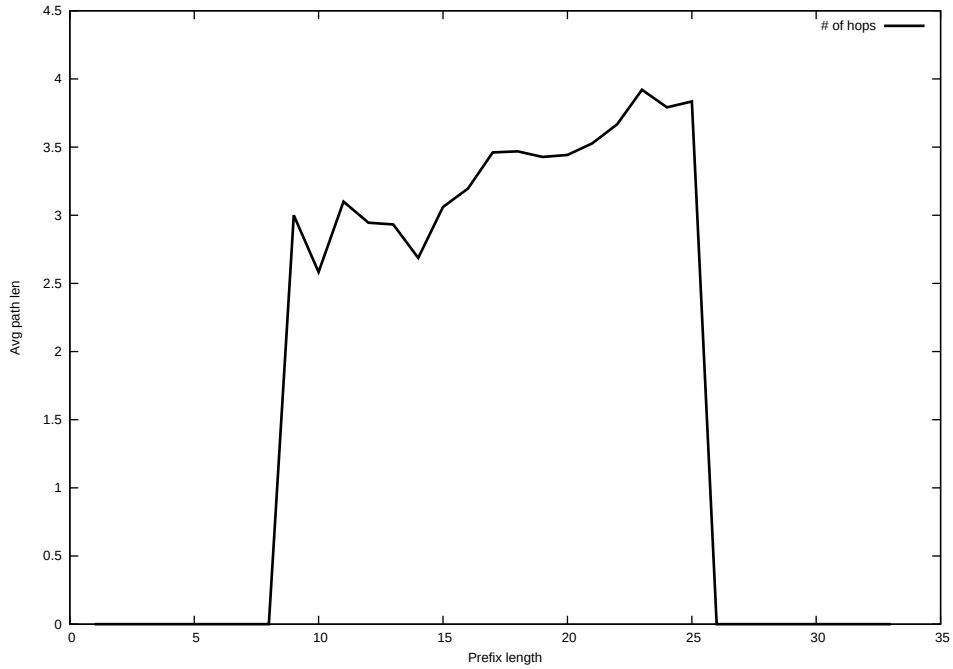
2014-12-11



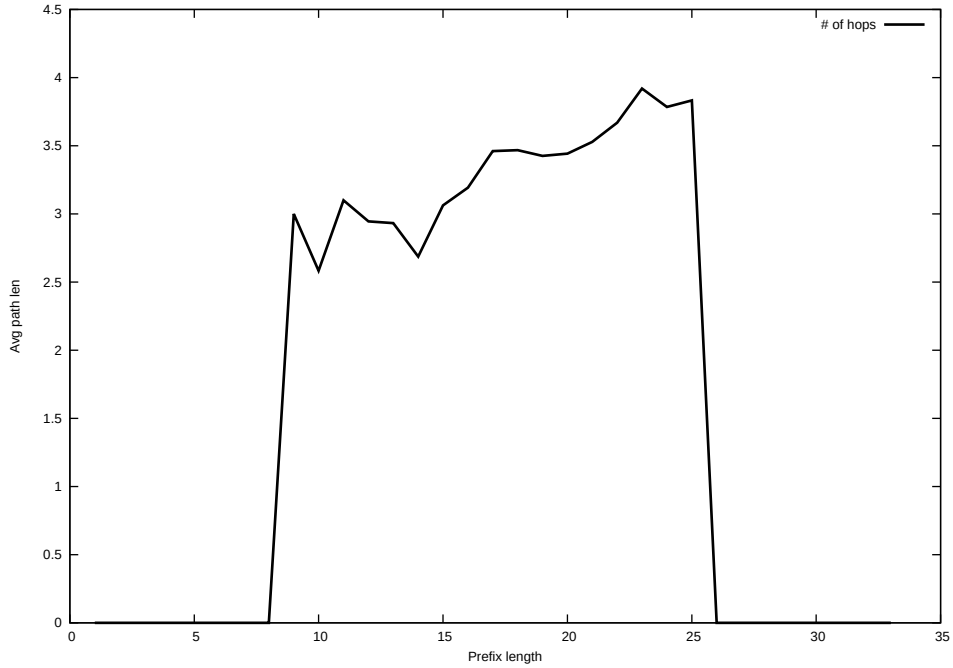
2014-12-12



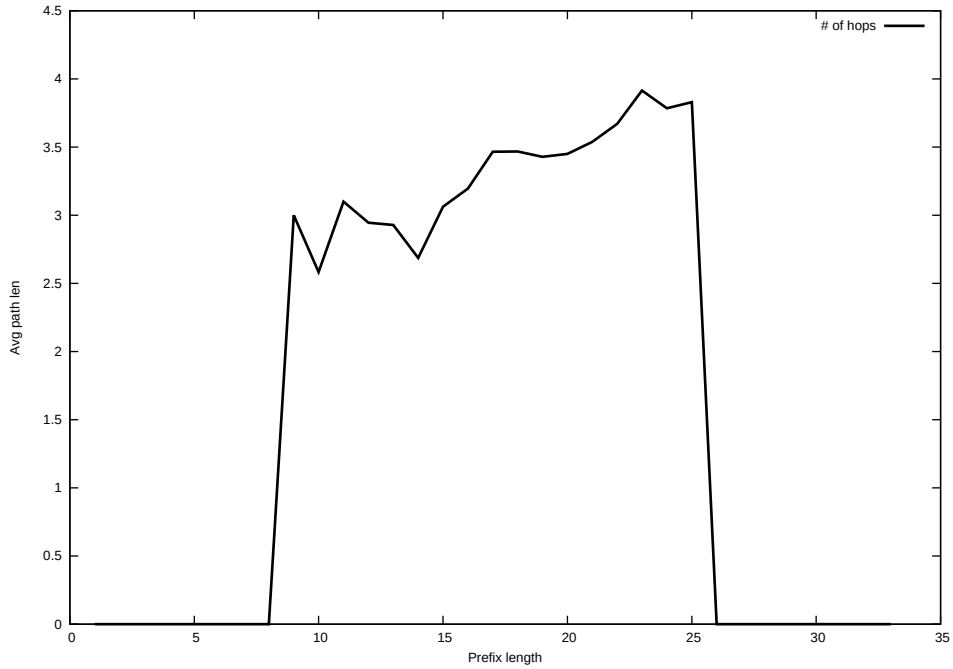
2014-12-13



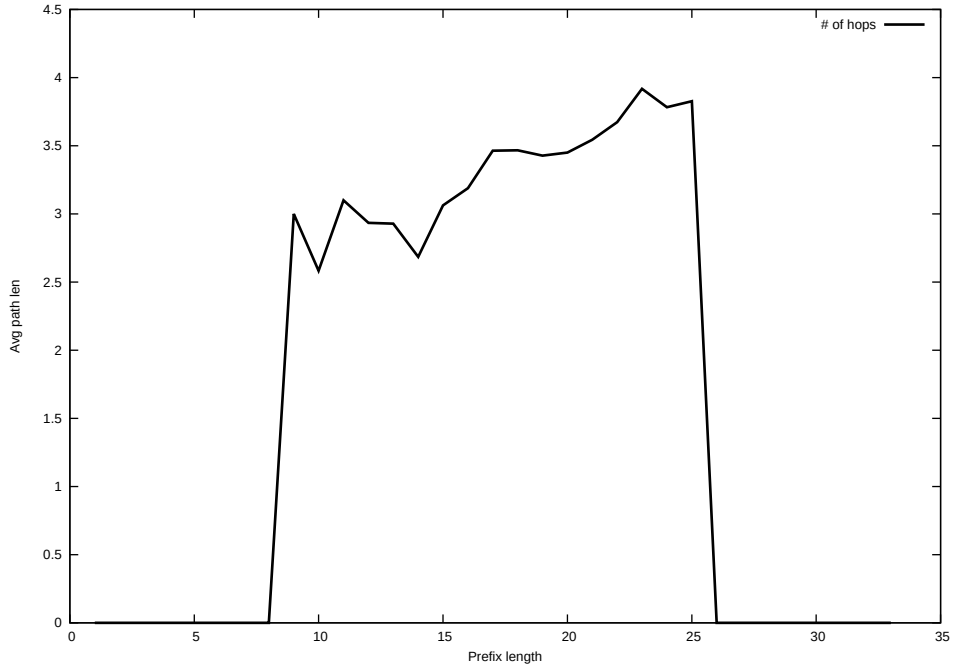
2014-12-14



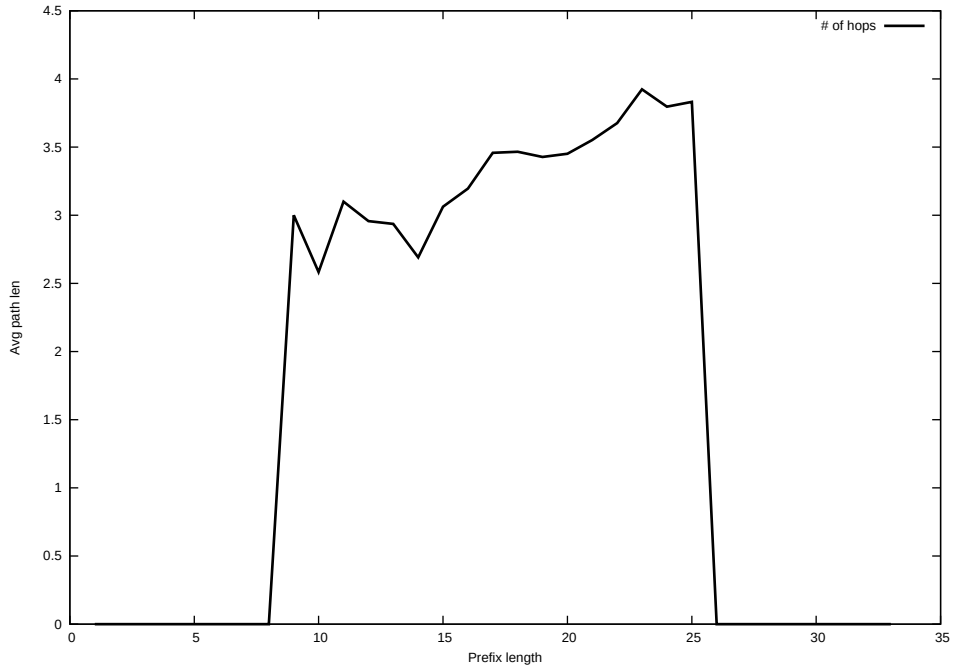
2014-12-15



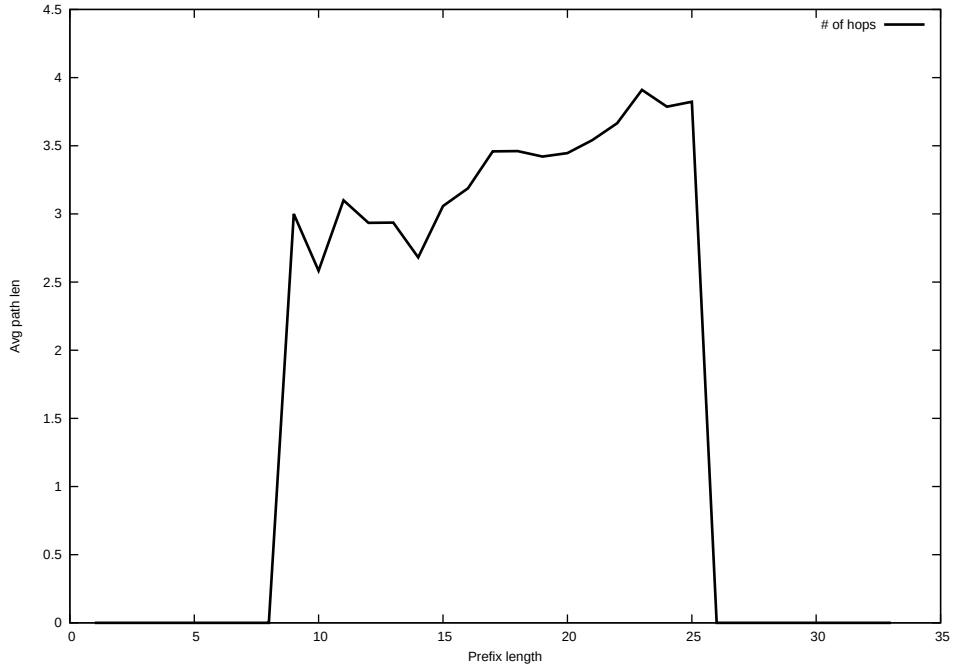
2014-12-16



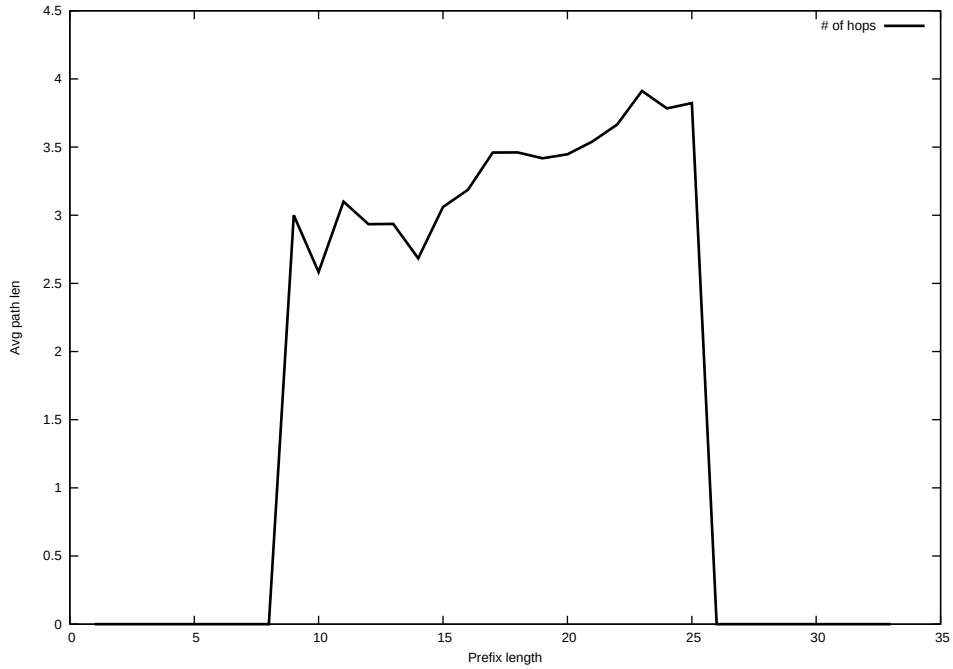
2014-12-17



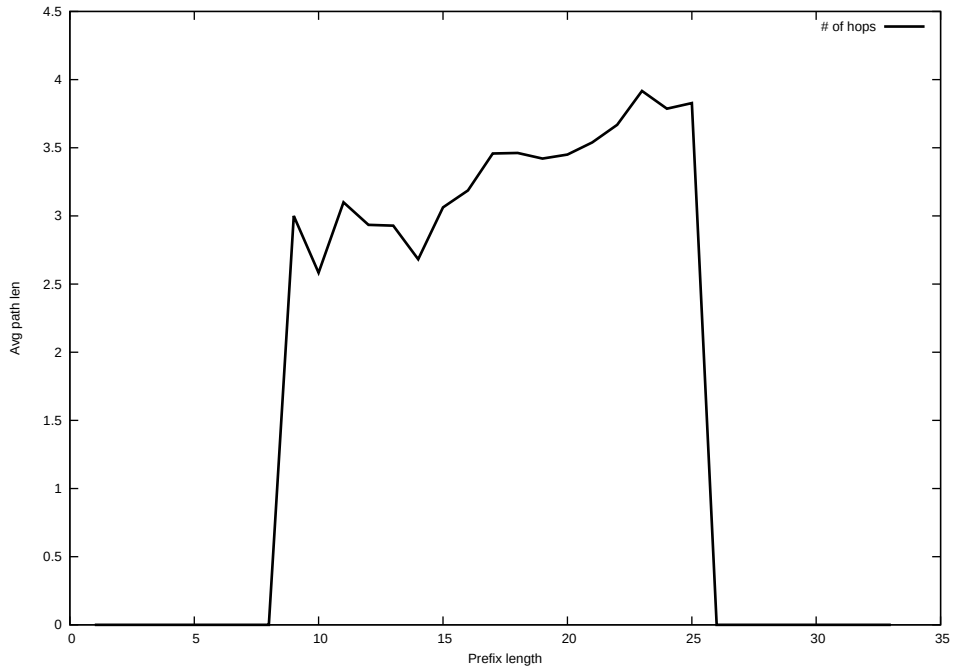
2014-12-18



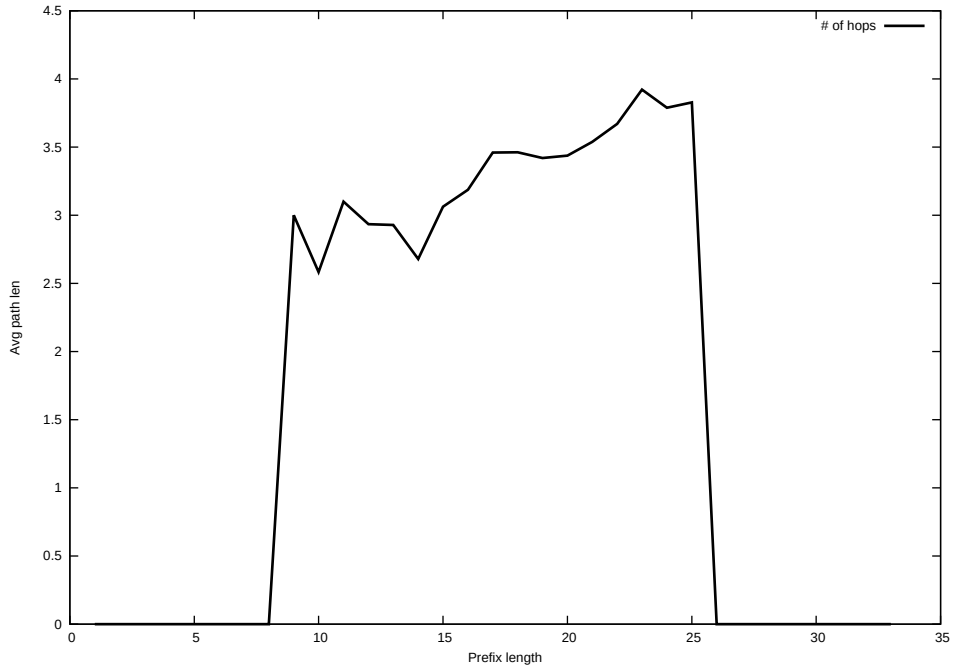
2014-12-19



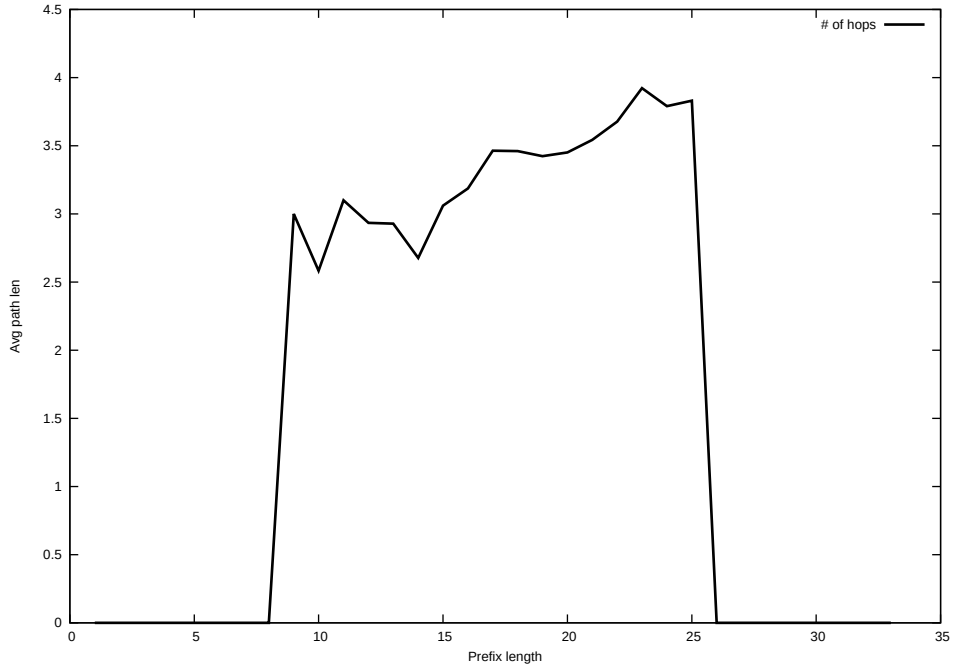
2014-12-20



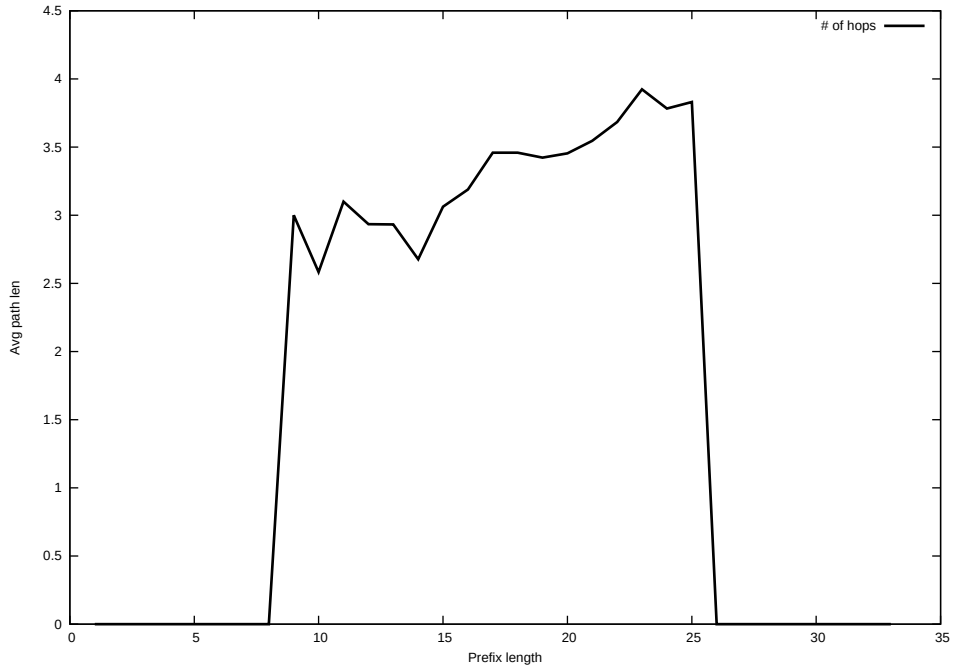
2014-12-21



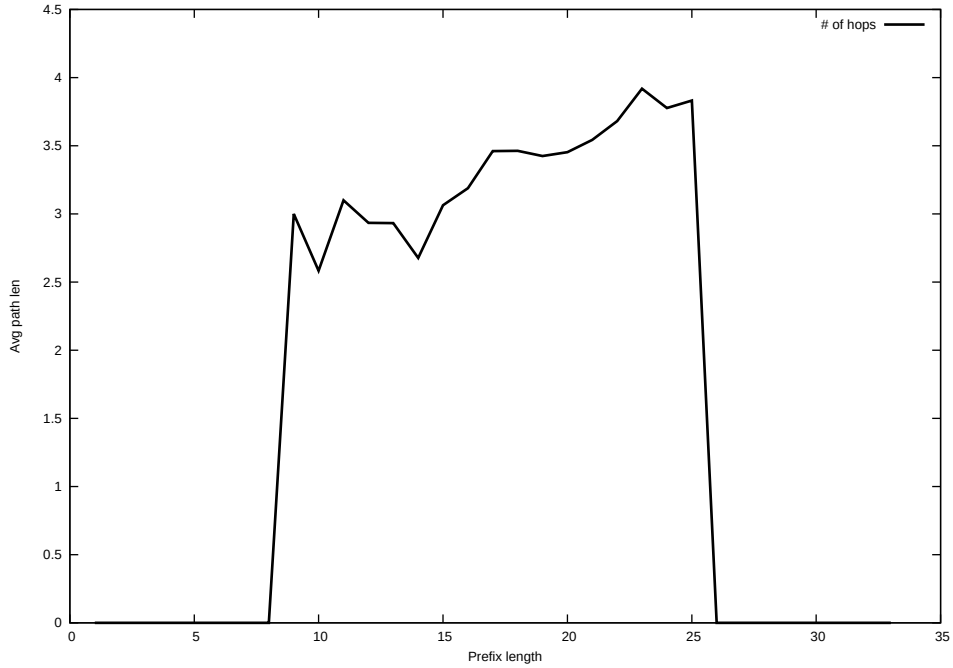
2014-12-22



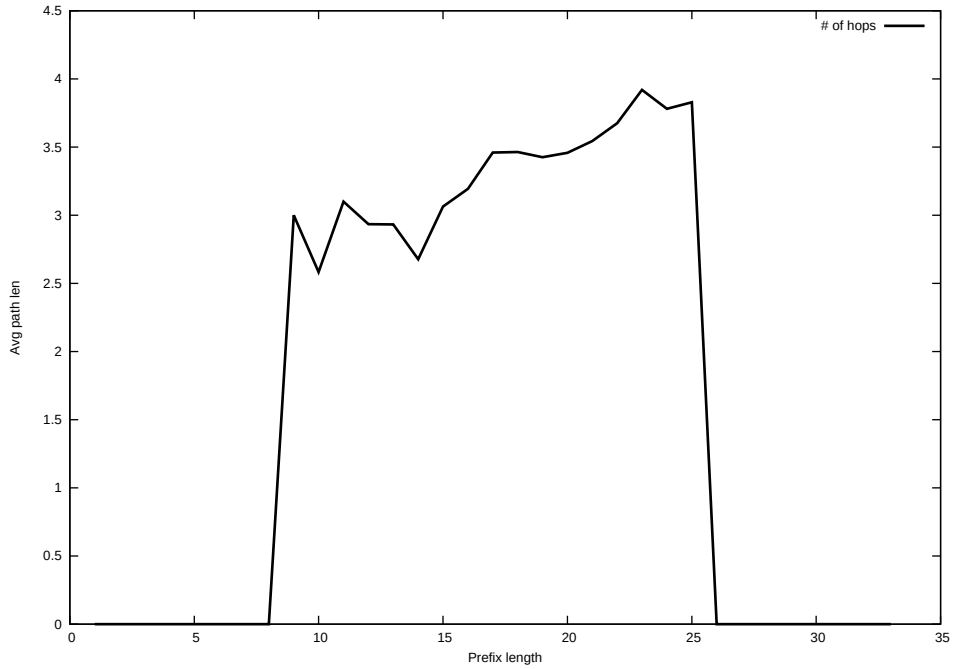
2014-12-23



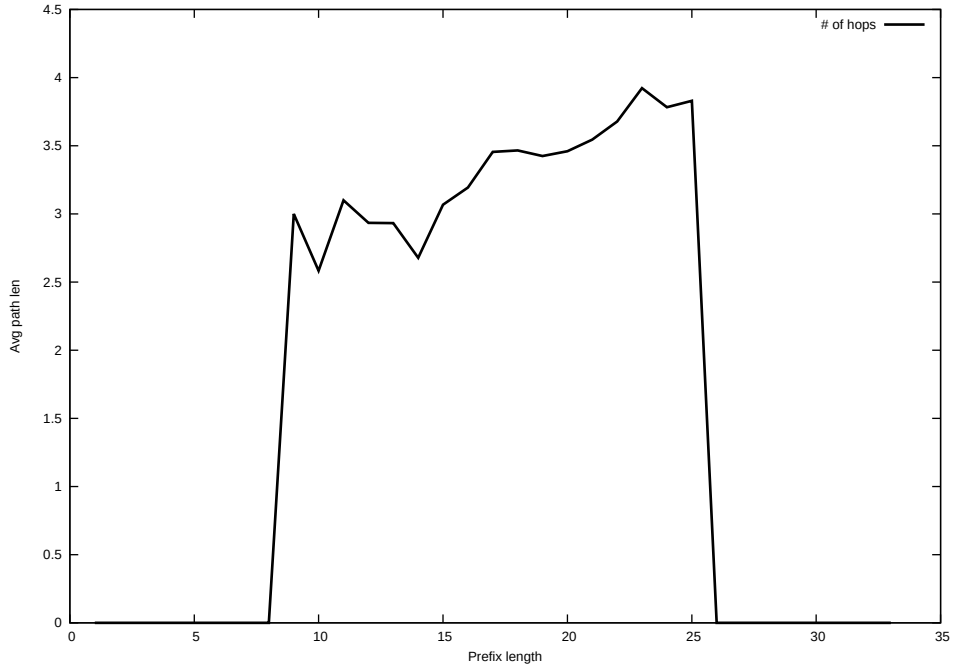
2014-12-24



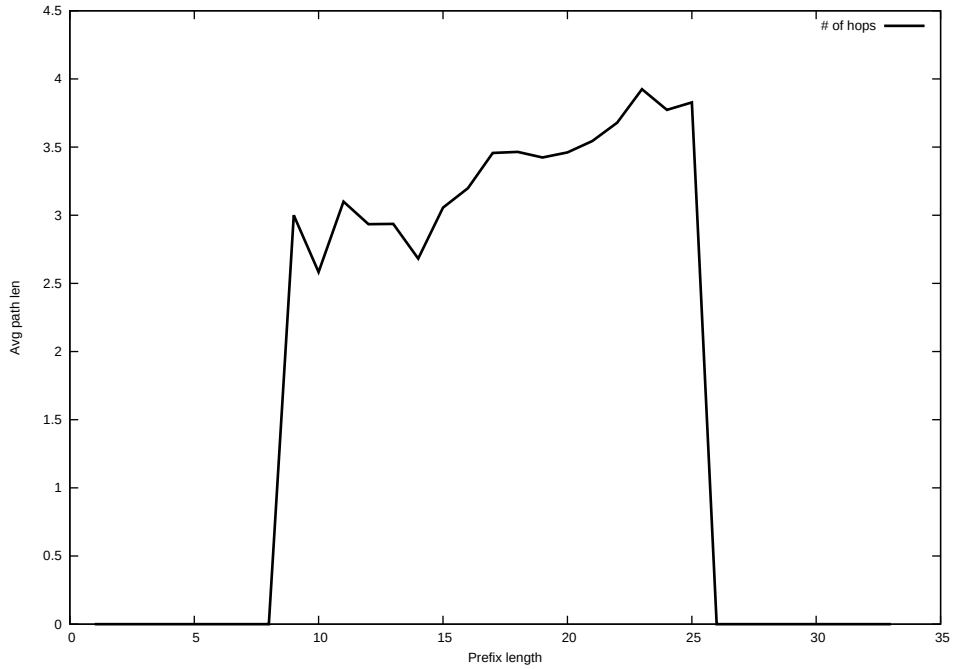
2014-12-25



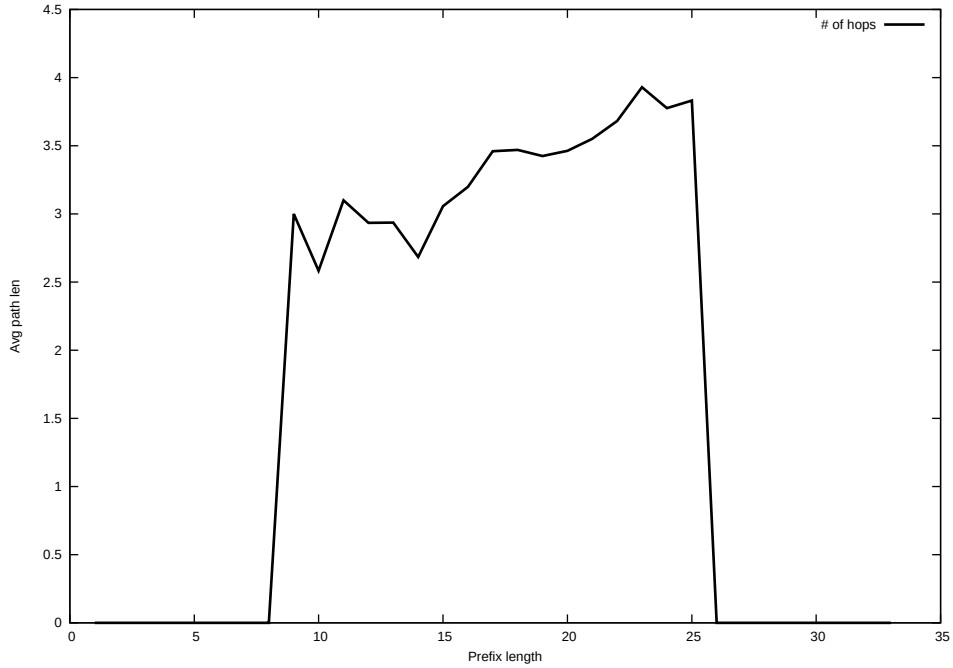
2014-12-26



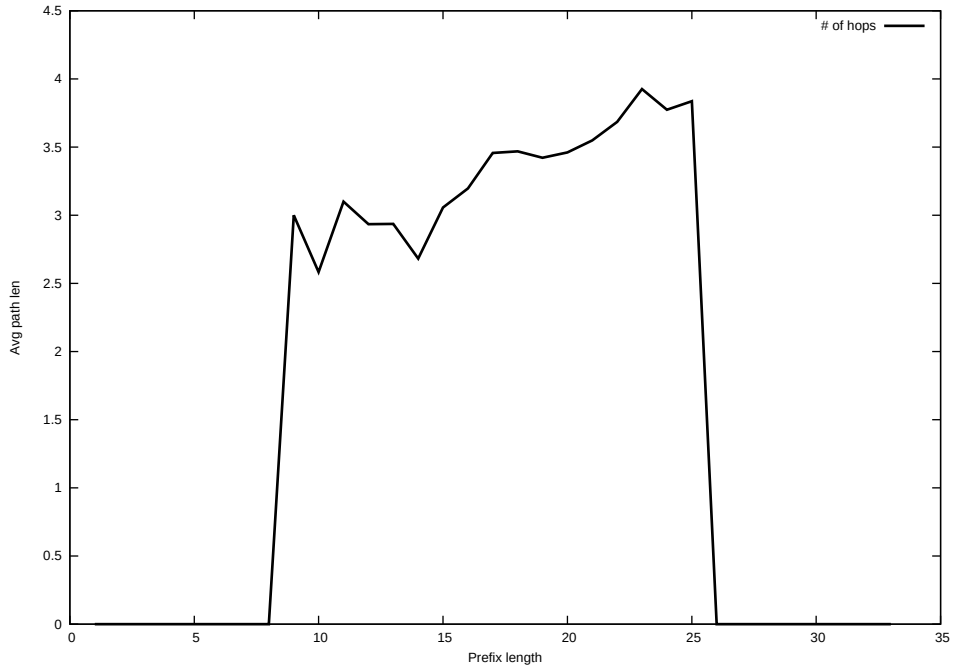
2014-12-27



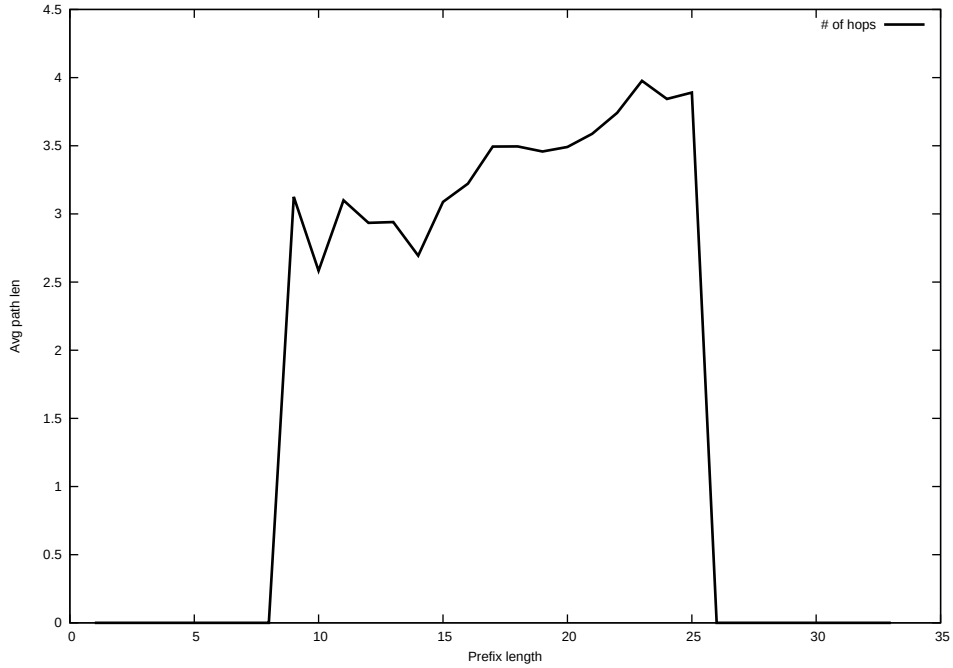
2014-12-28



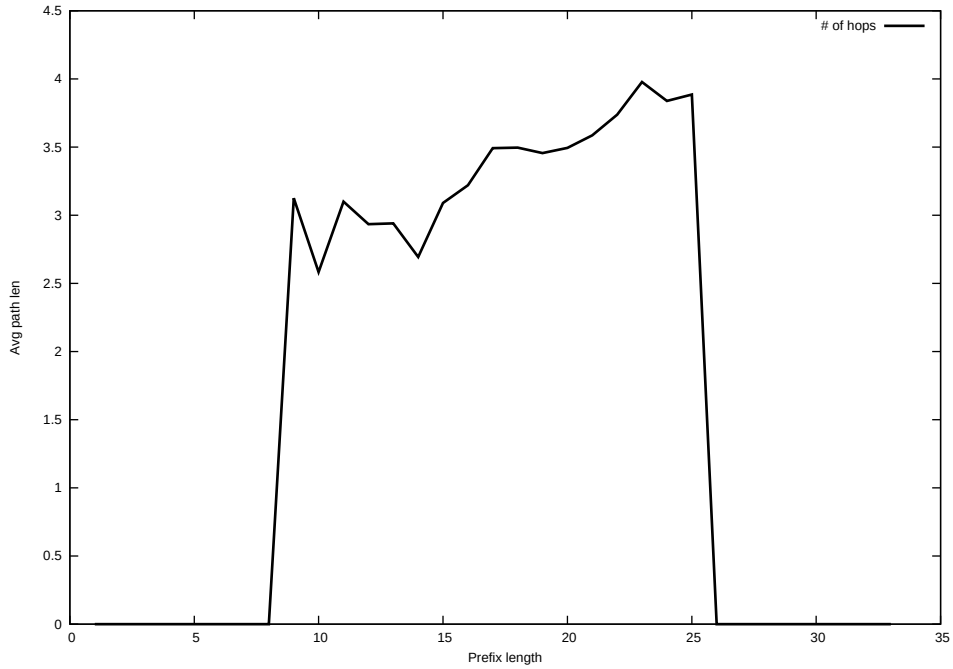
2014-12-29



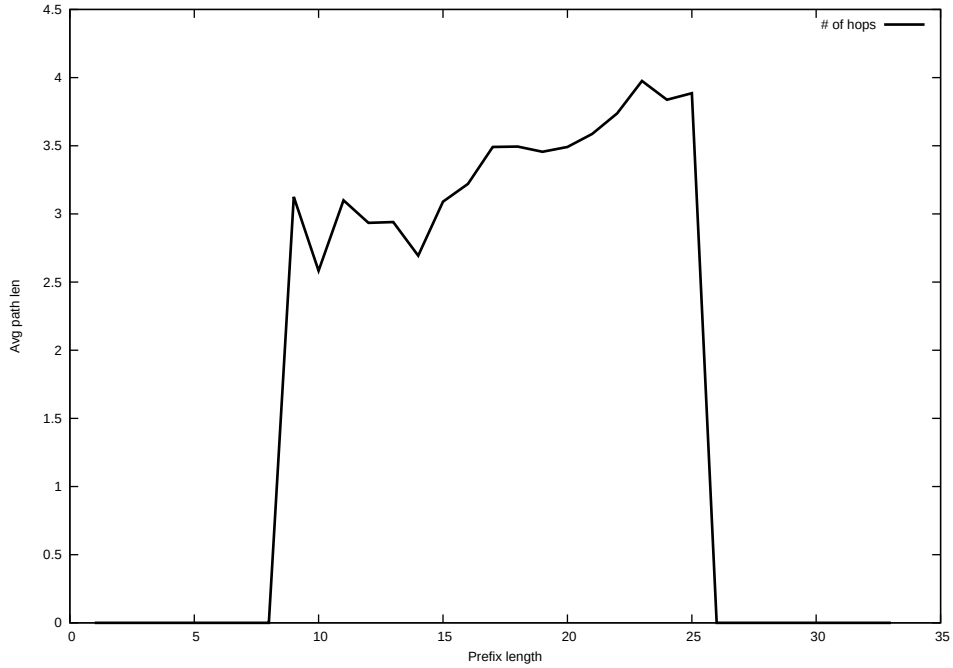
2014-12-30



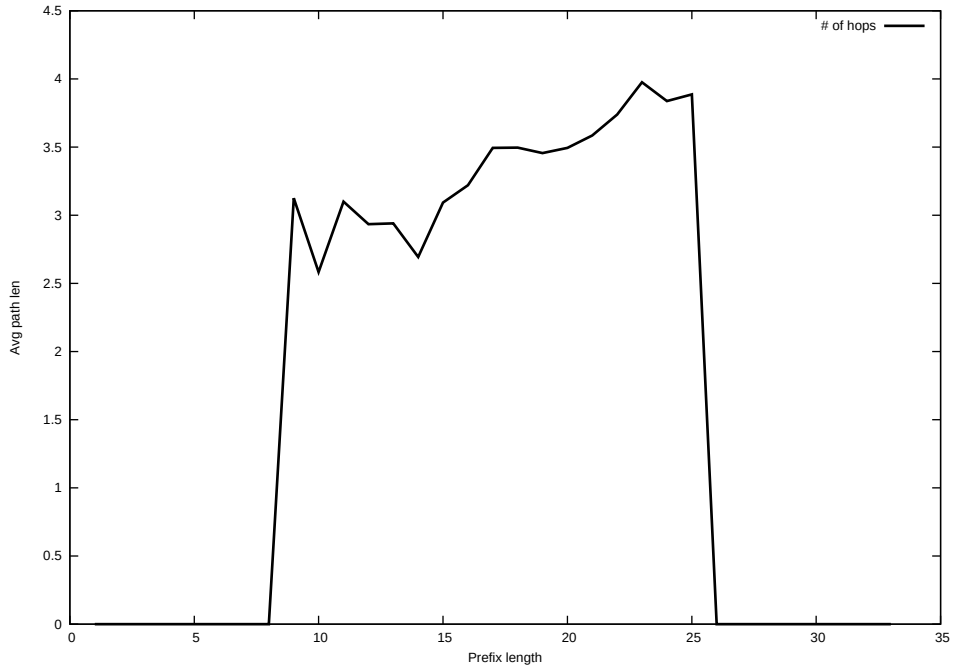
2014-12-31



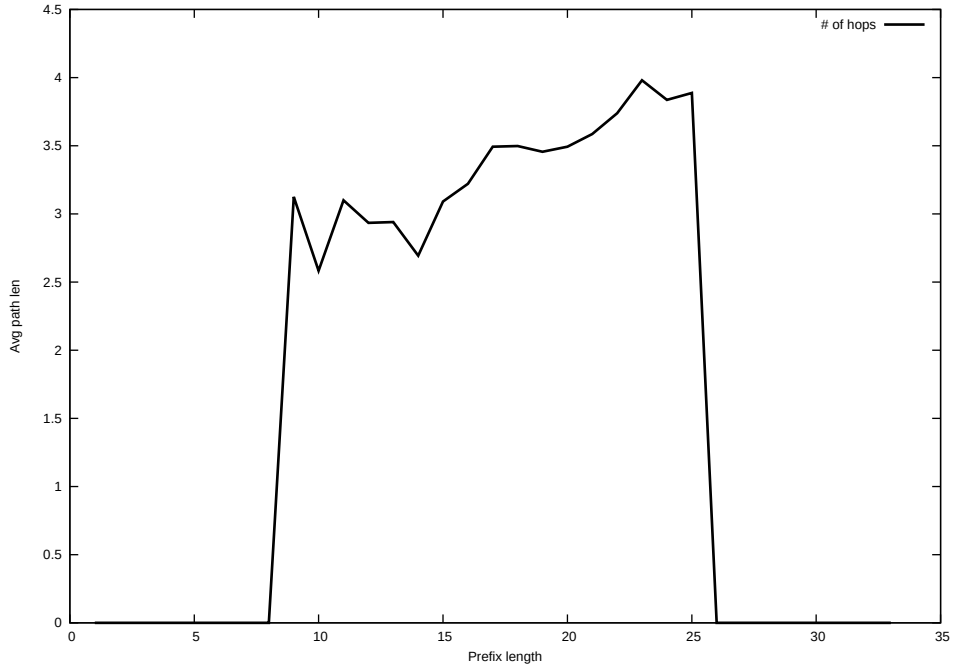
2015-01-01



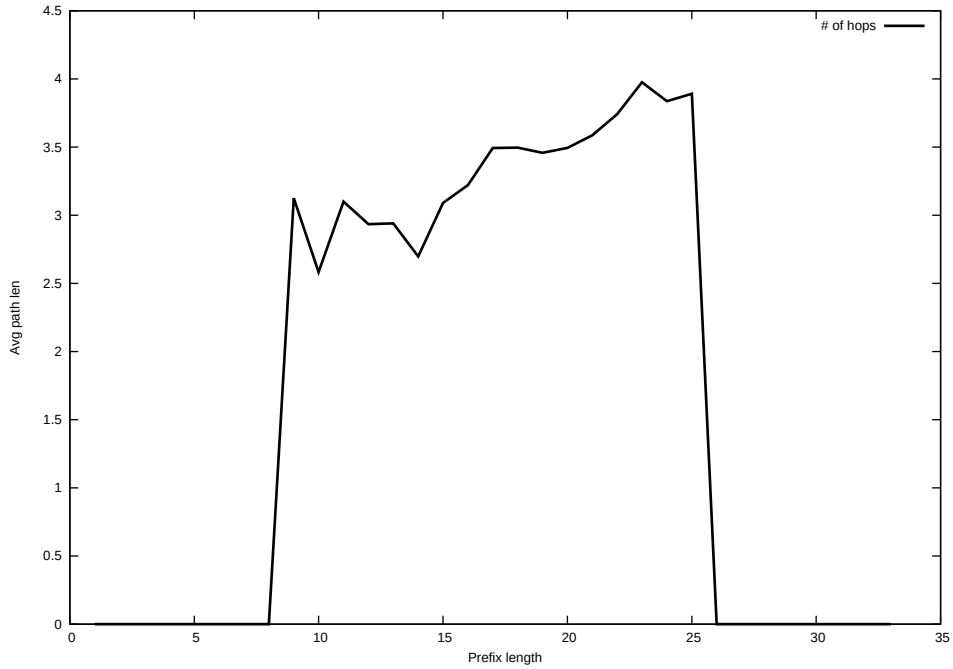
2015-01-02



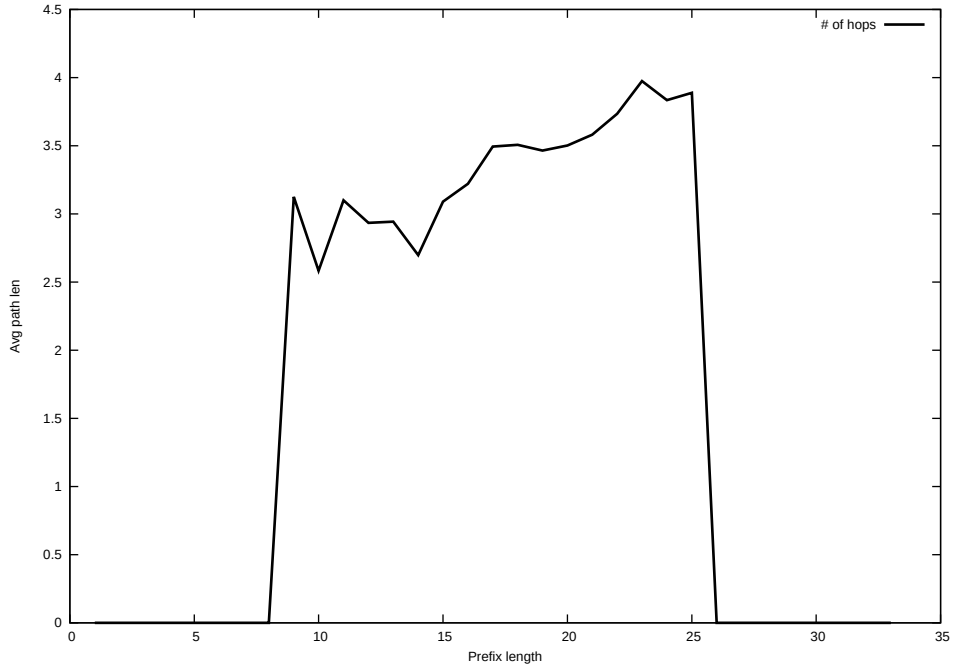
2015-01-03



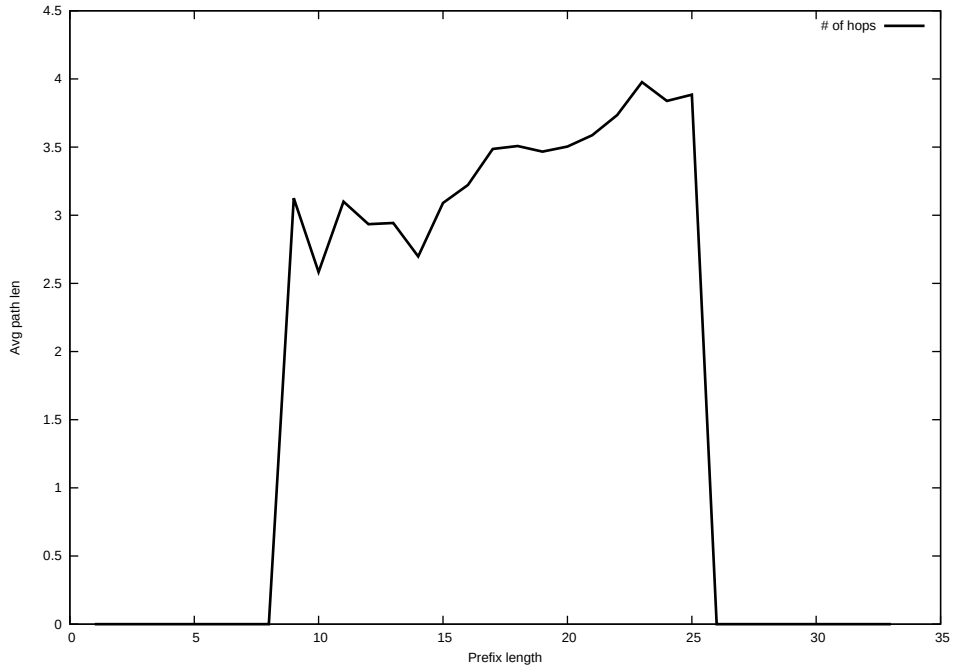
2015-01-04



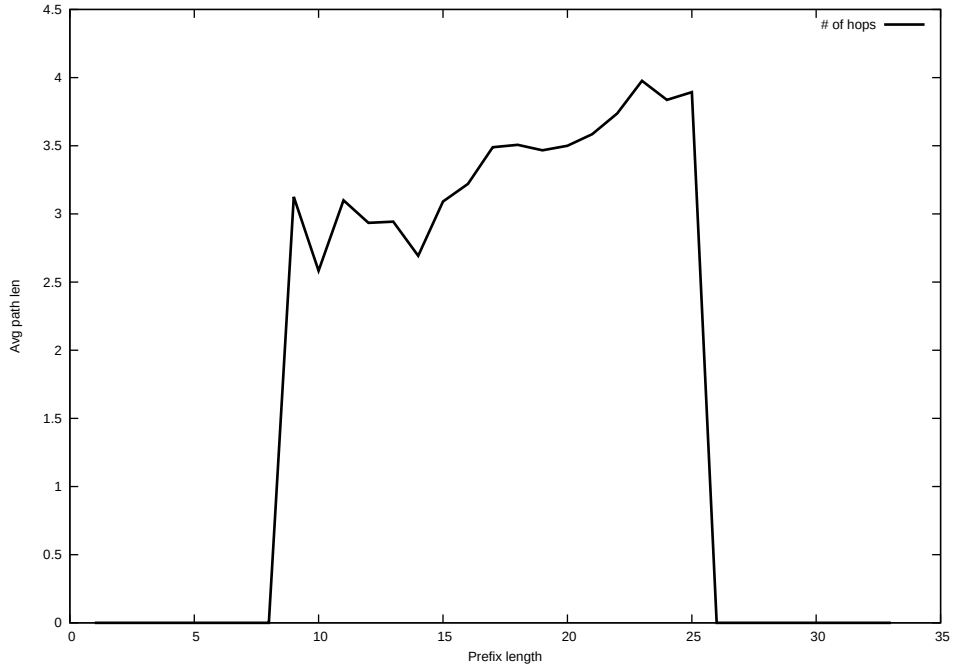
2015-01-05



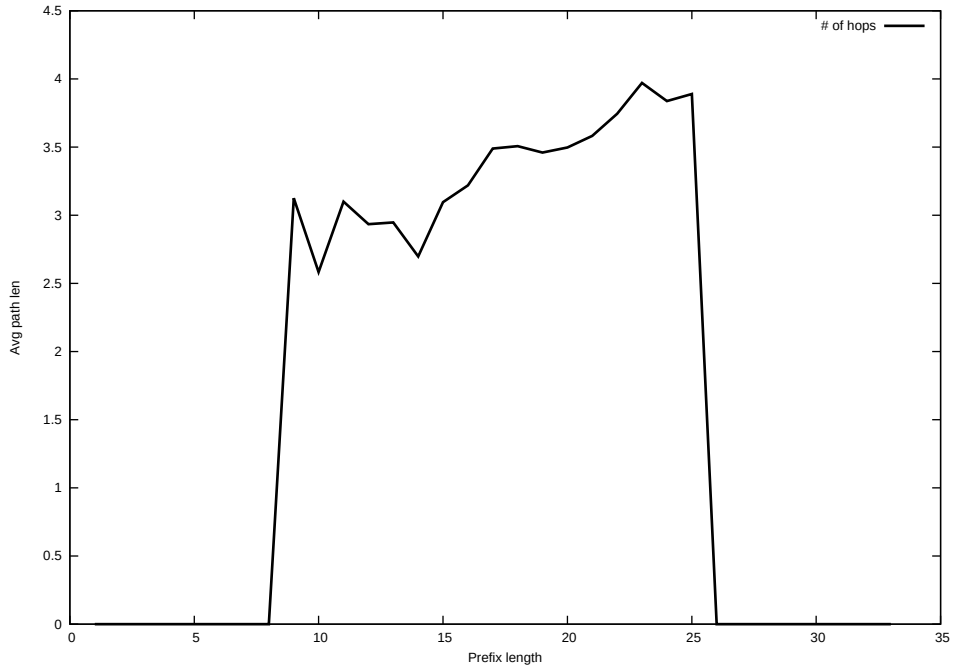
2015-01-06



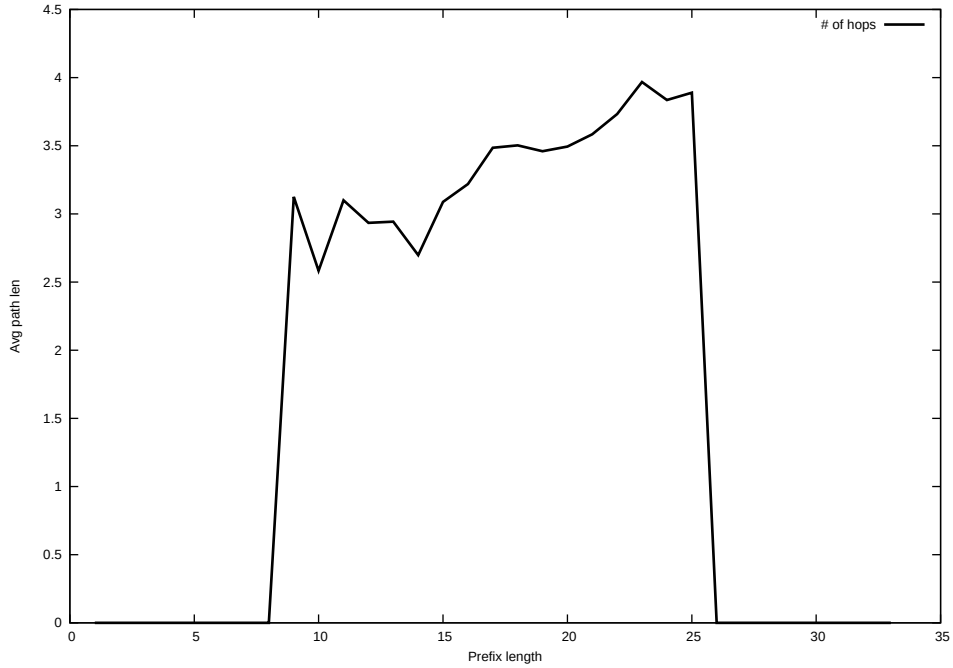
2015-01-07



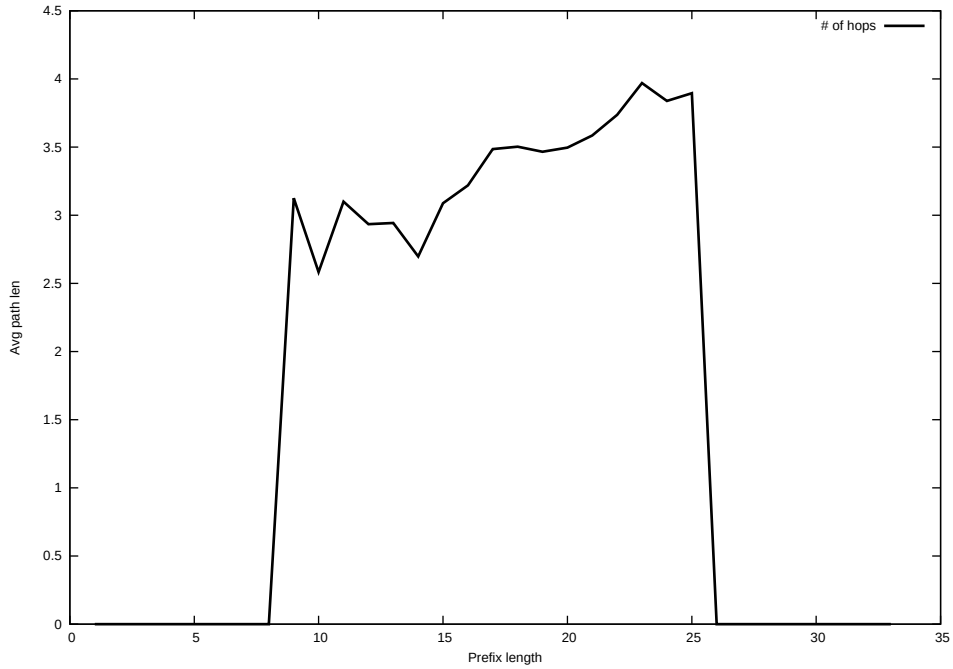
2015-01-08



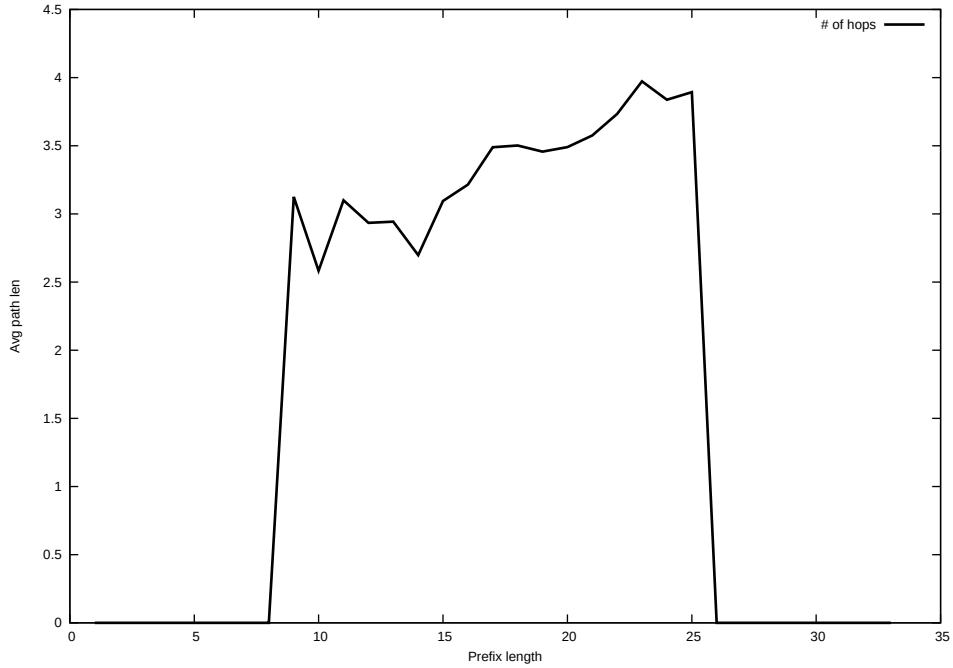
2015-01-09



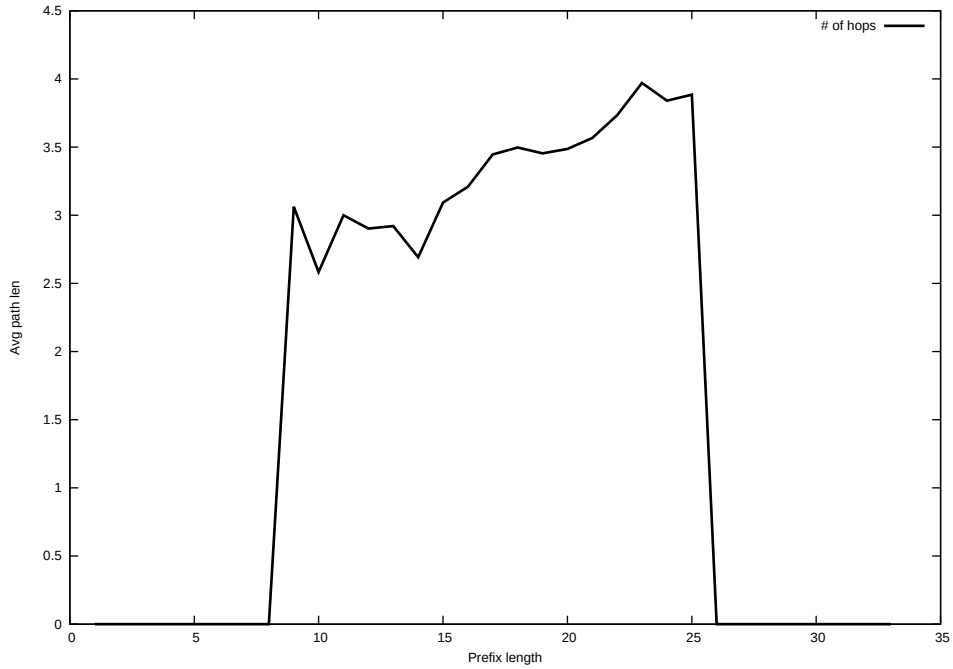
2015-01-10



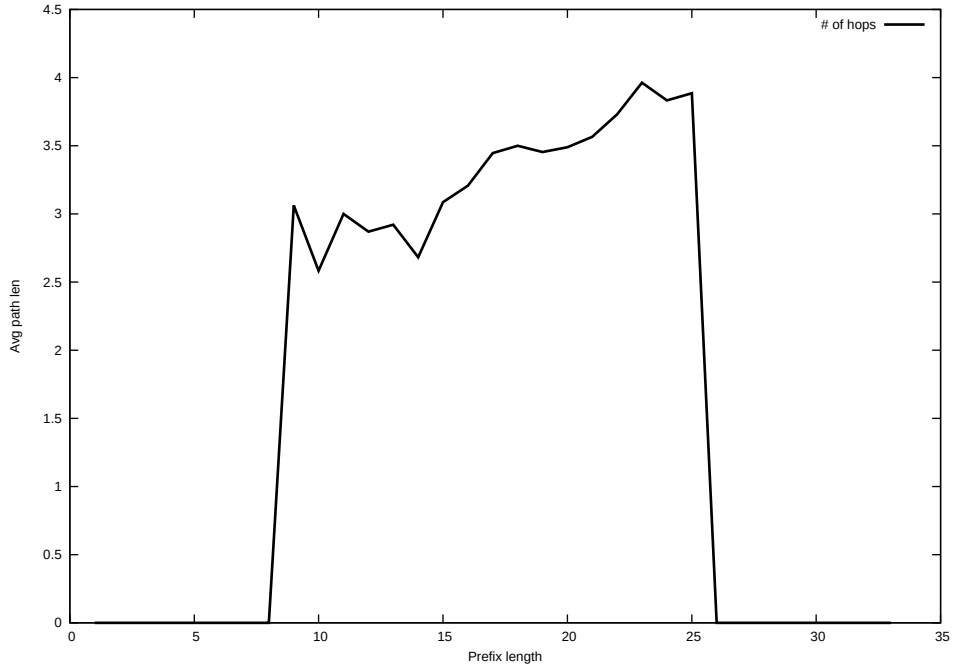
2015-01-11



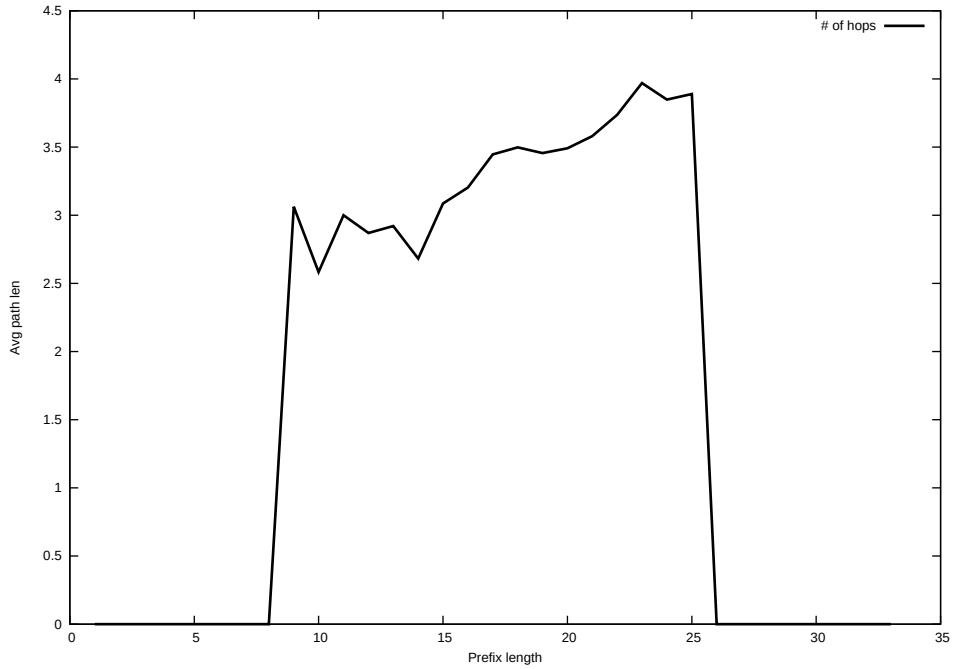
2015-01-12



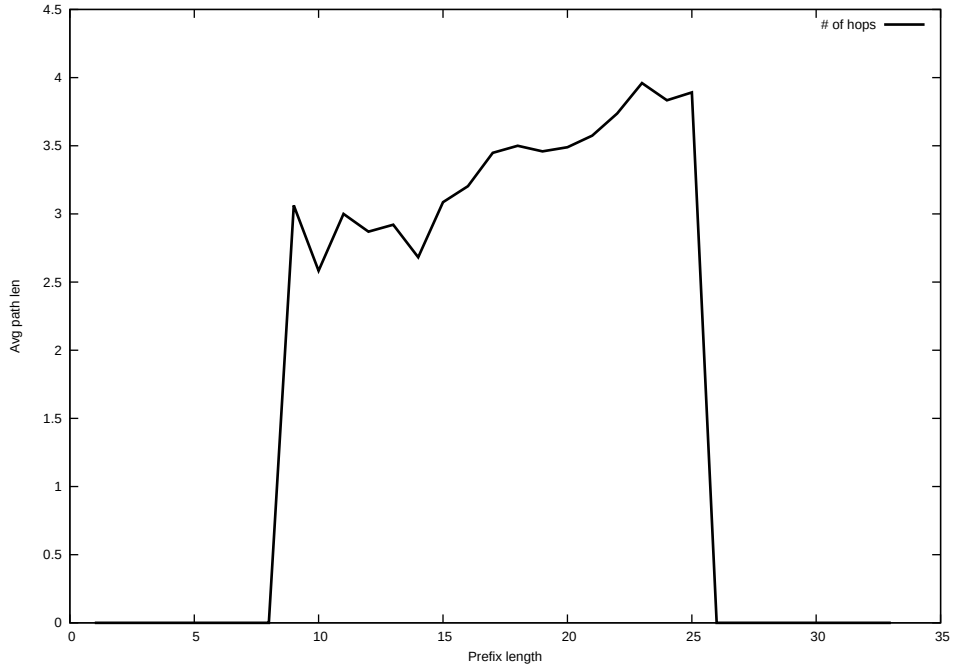
2015-01-13



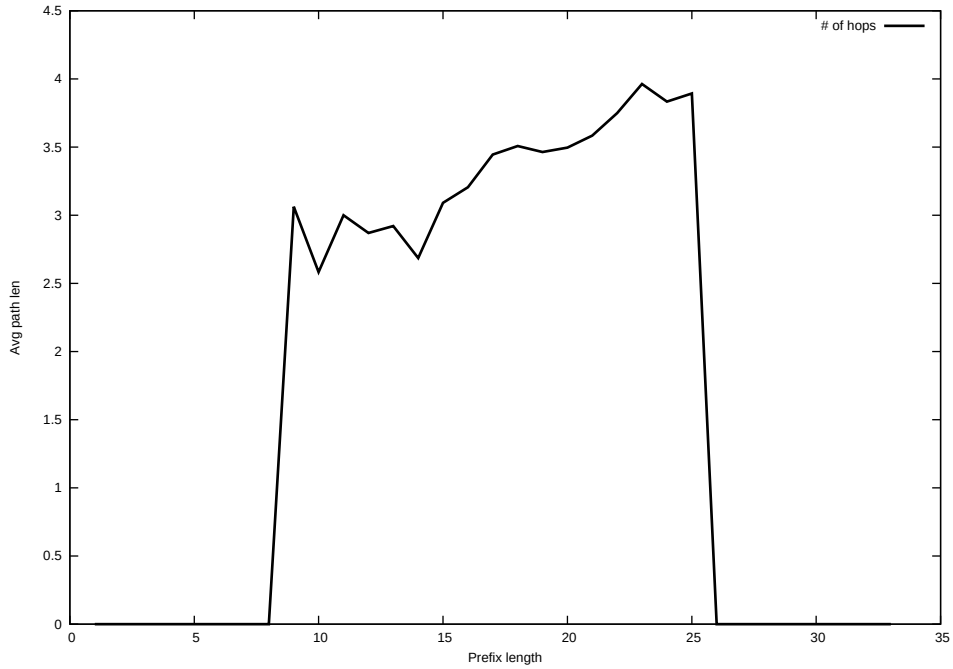
2015-01-14



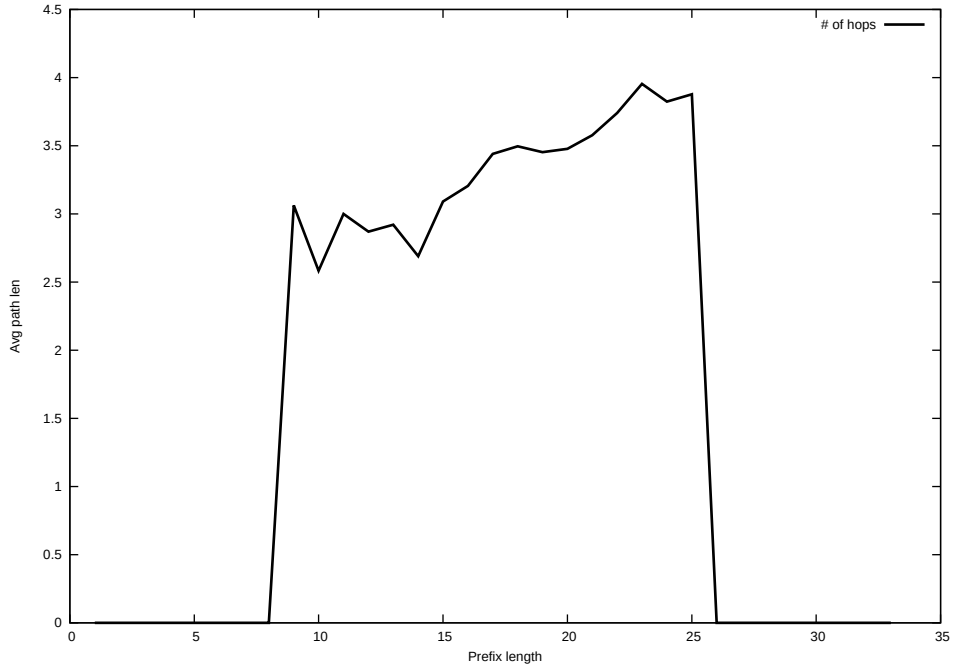
2015-01-15



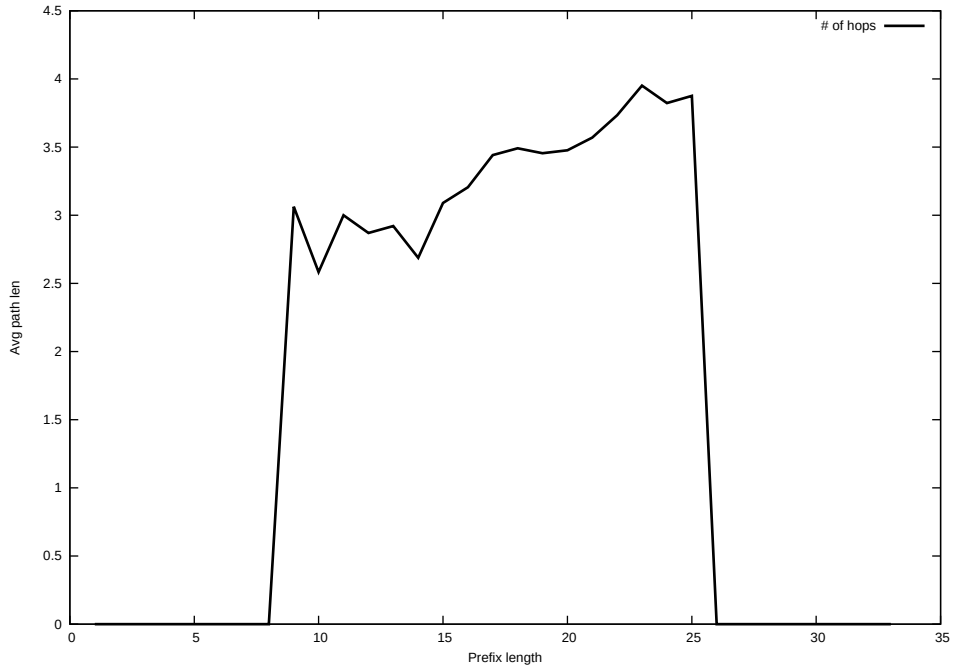
2015-01-16



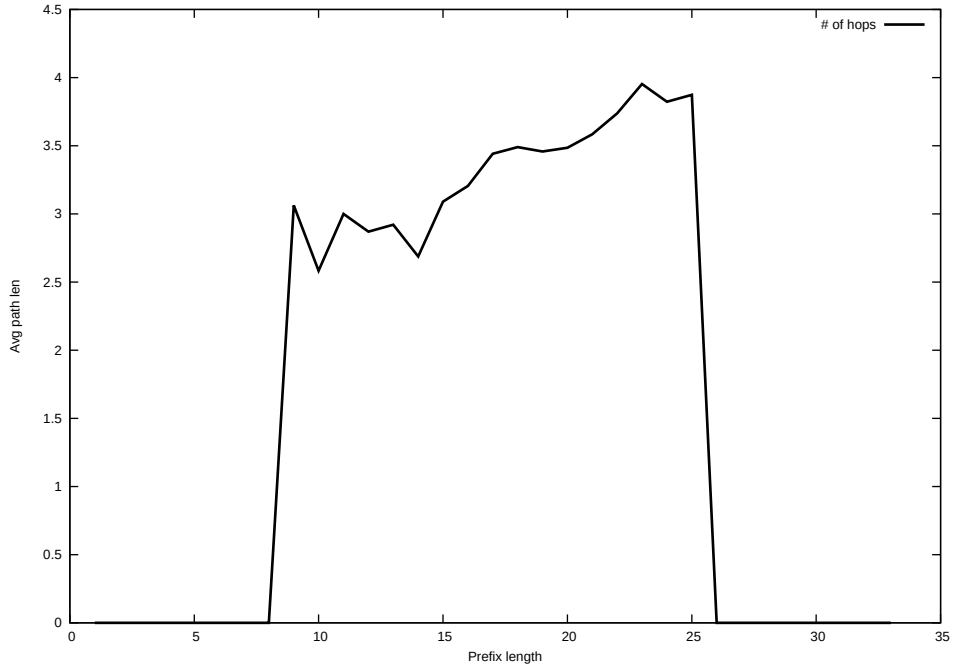
2015-01-17



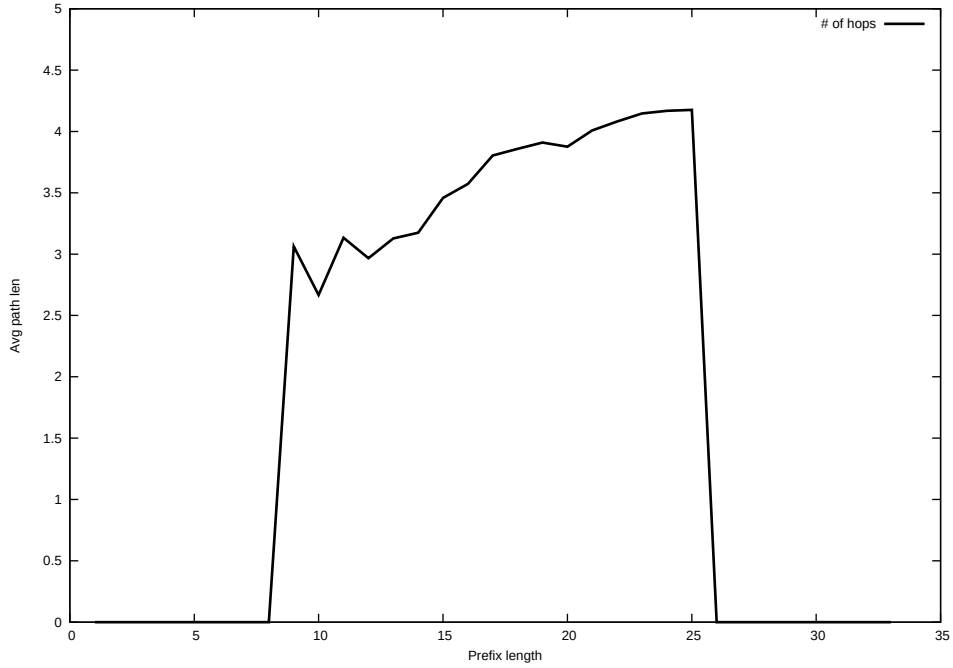
2015-01-18



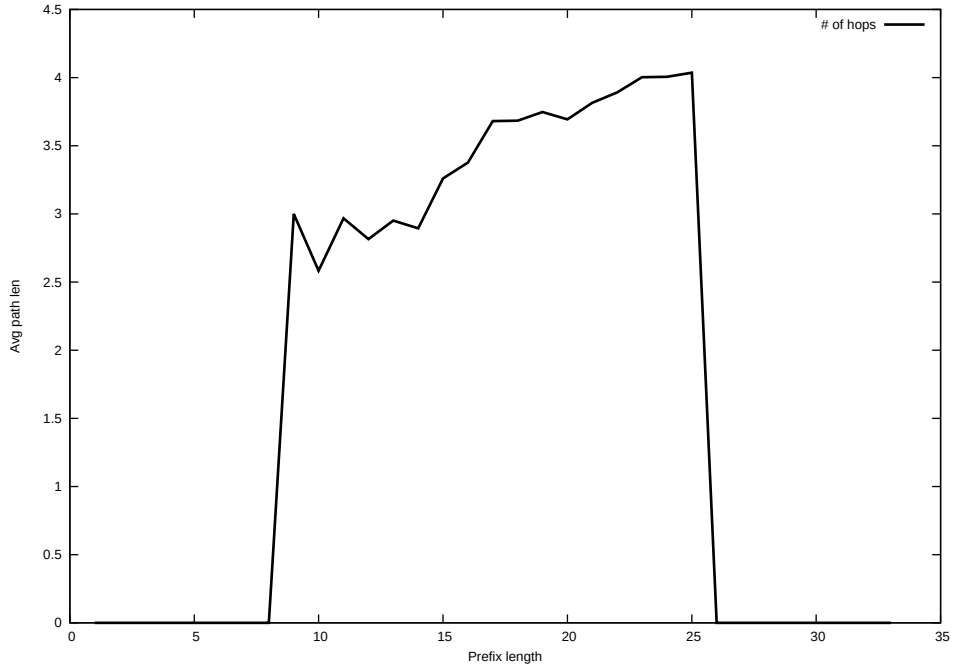
2015-01-19



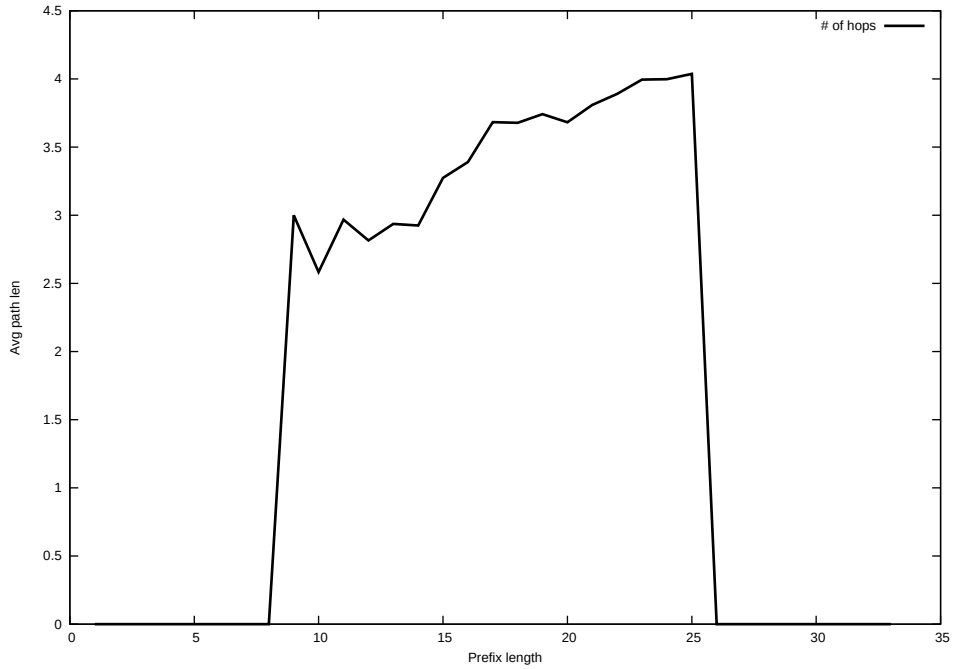
2015-01-20



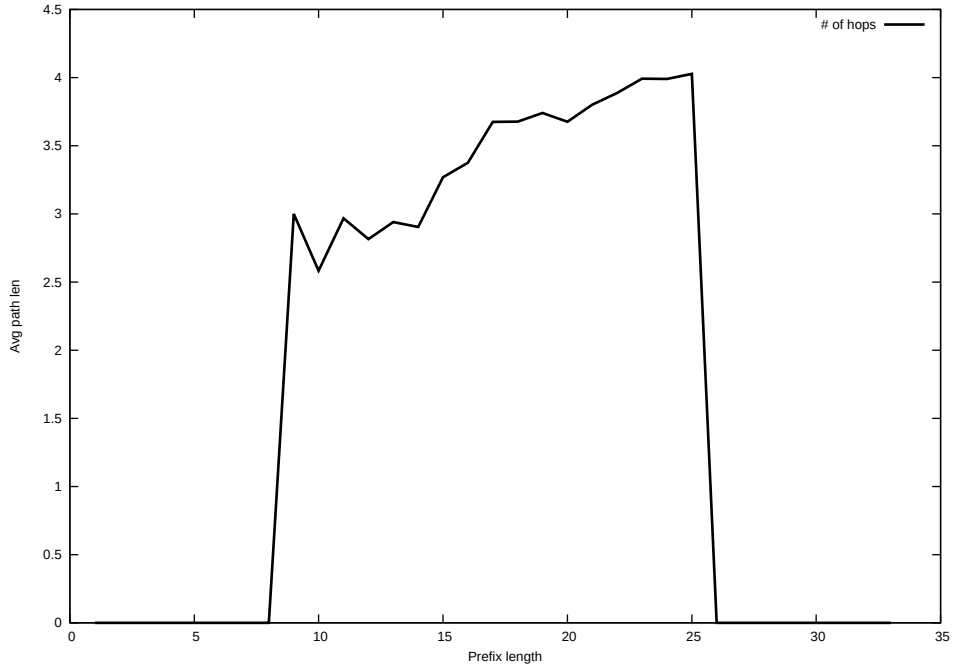
2015-01-21



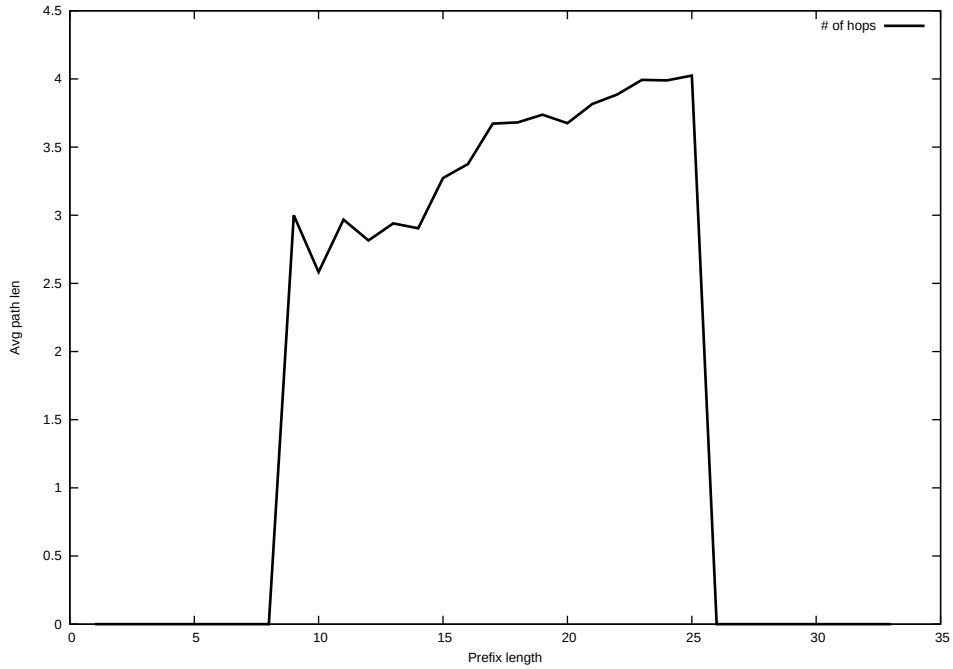
2015-01-22



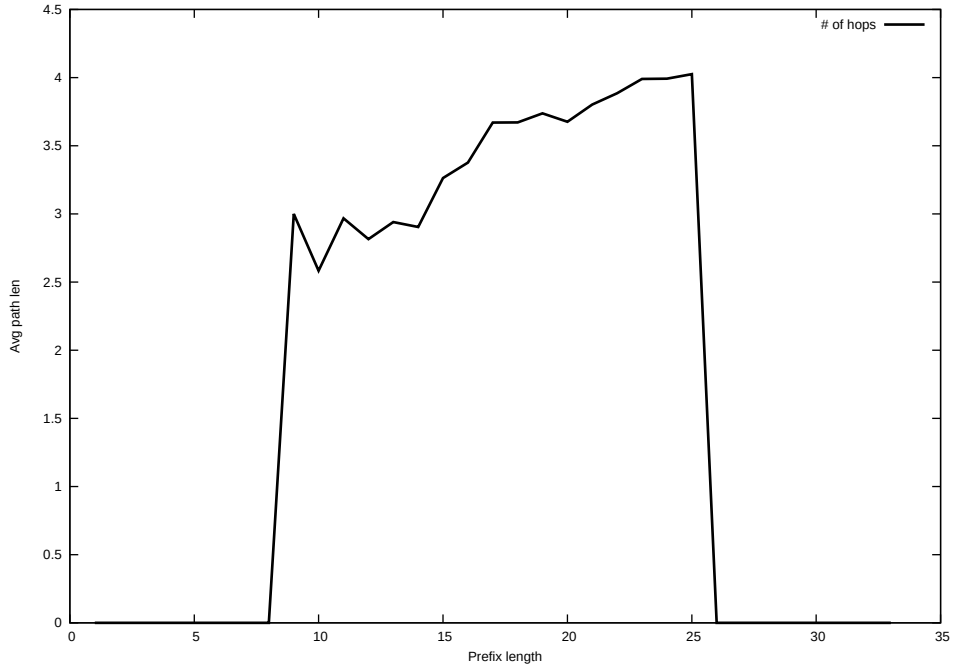
2015-01-23



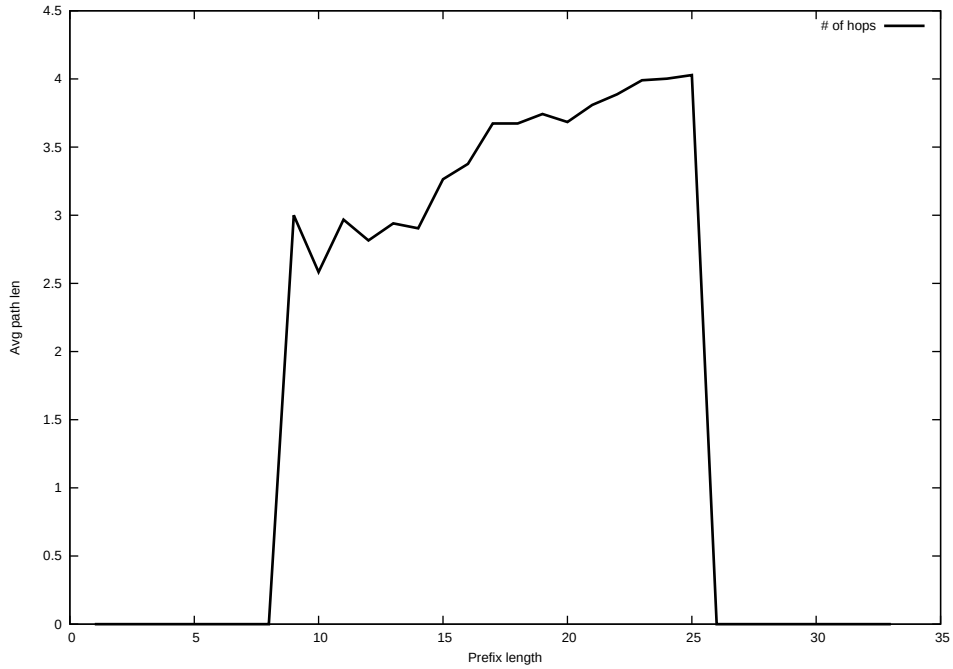
2015-01-24



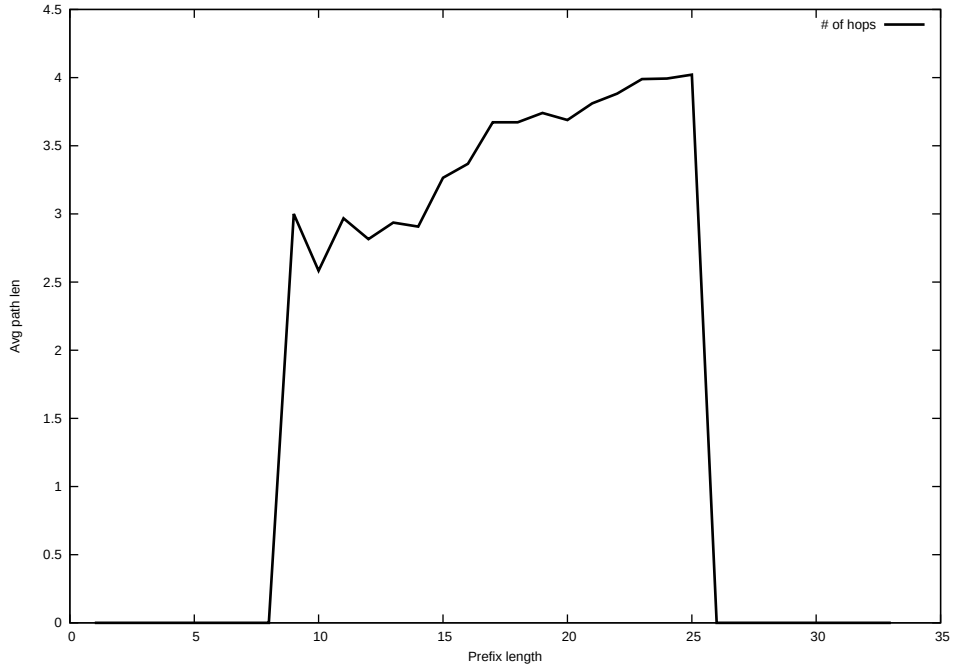
2015-01-25



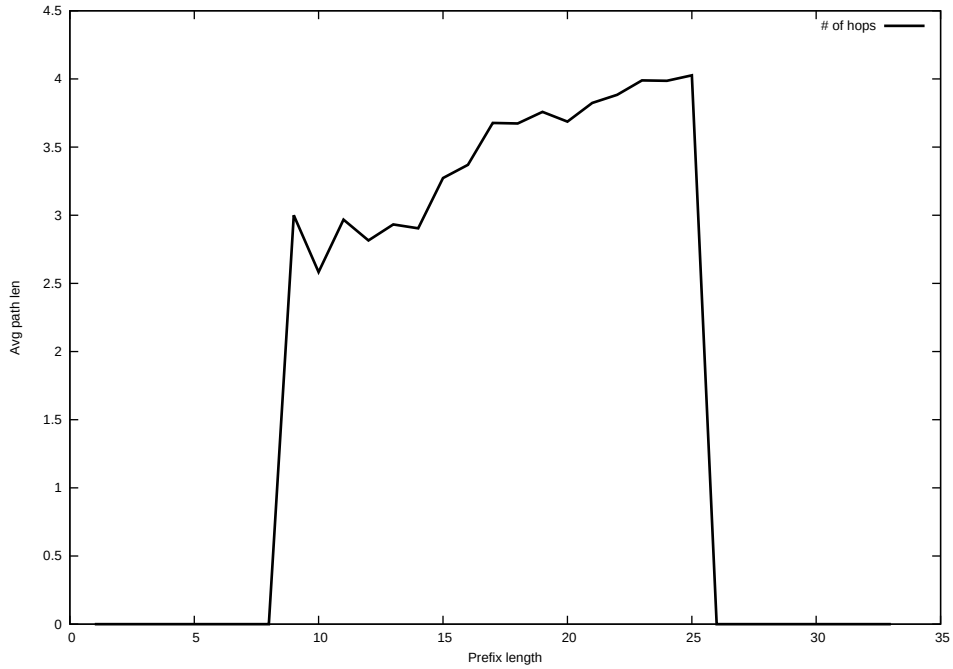
2015-01-26



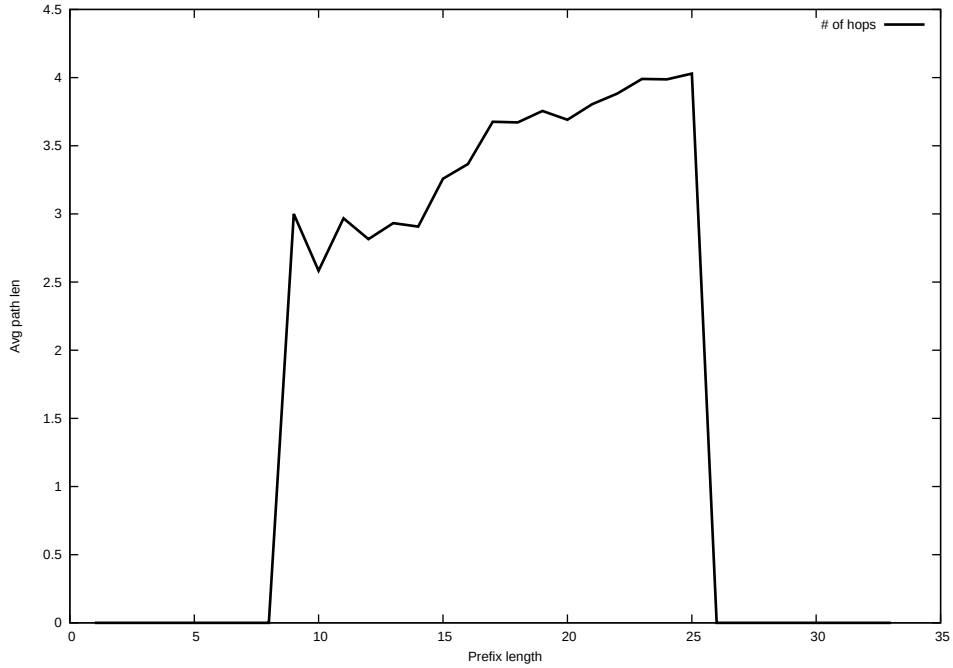
2015-01-27



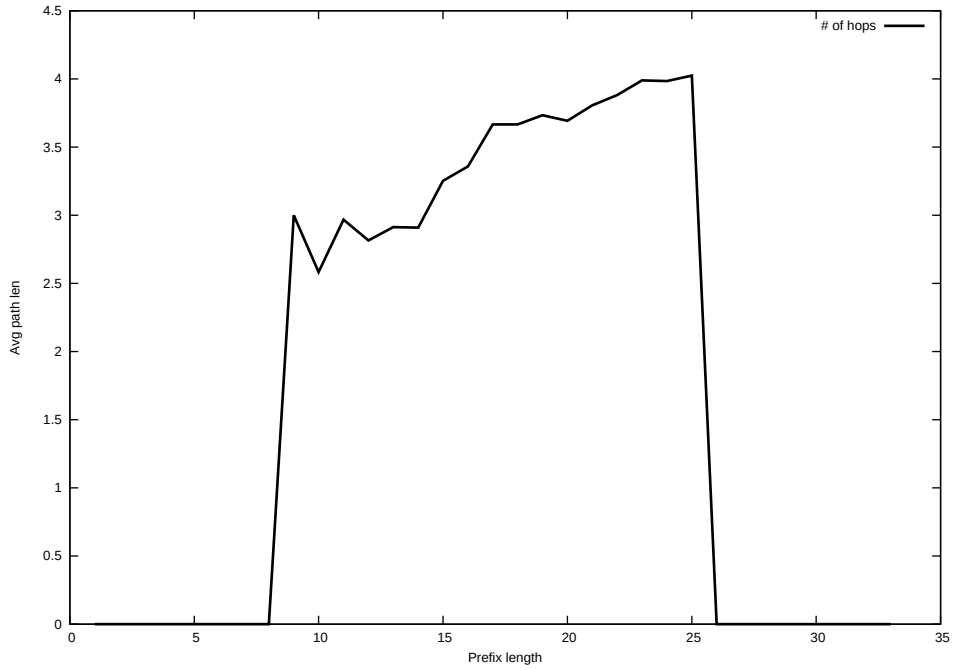
2015-01-28



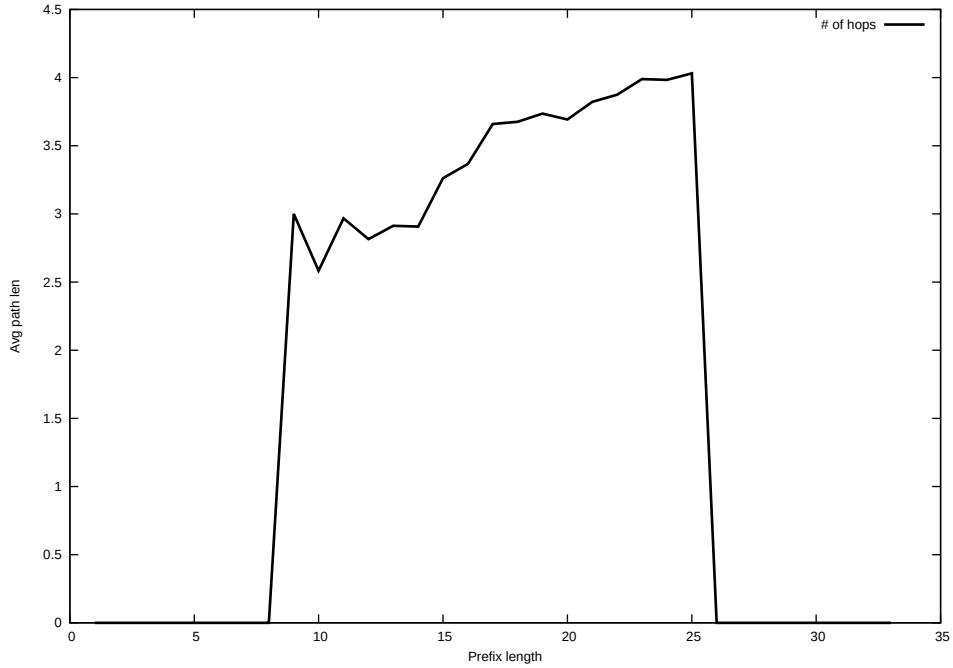
2015-01-29



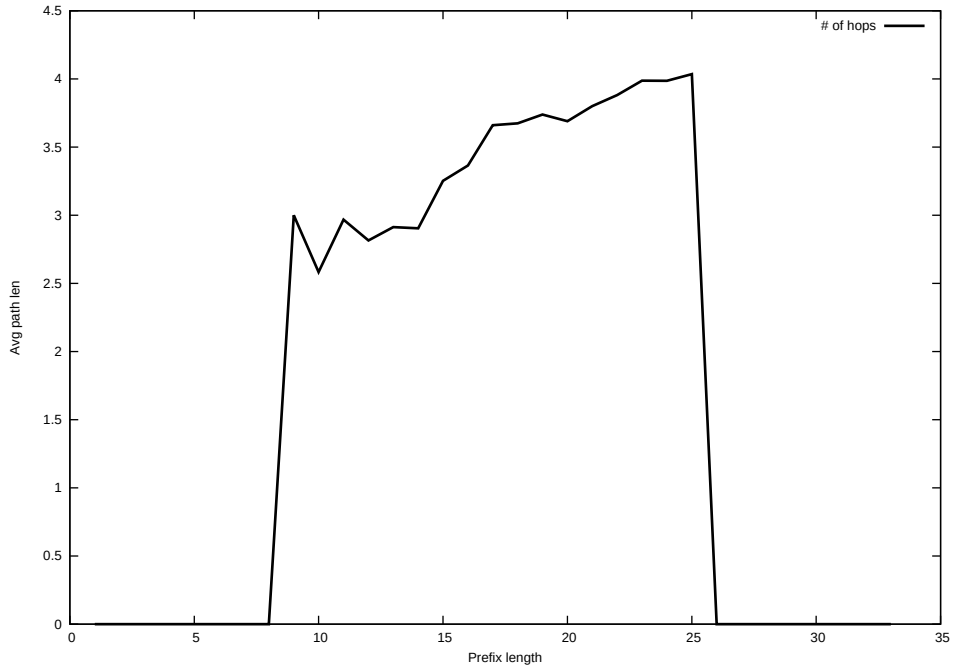
2015-01-30



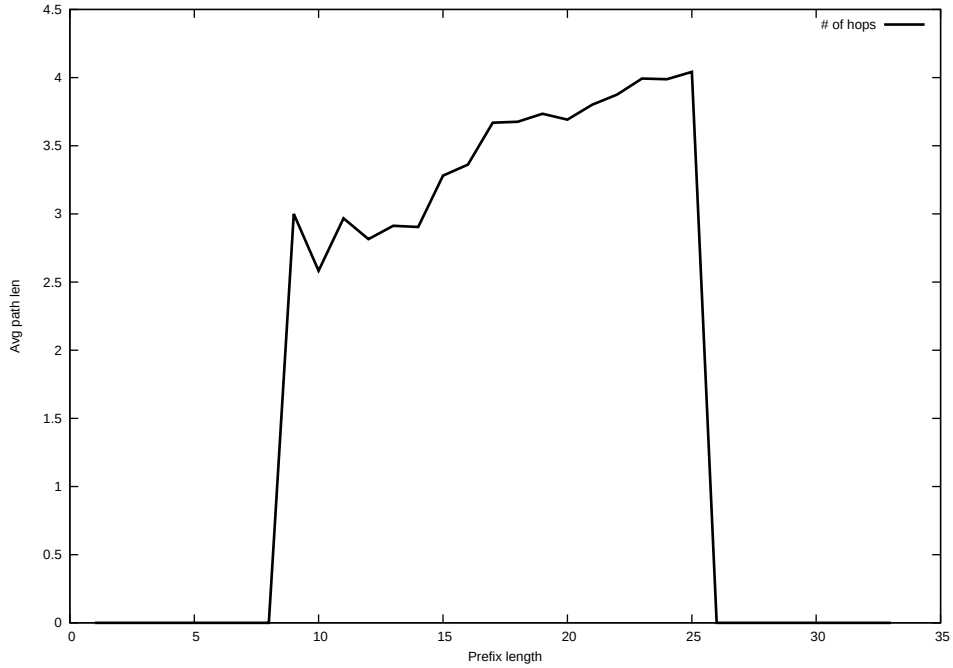
2015-01-31



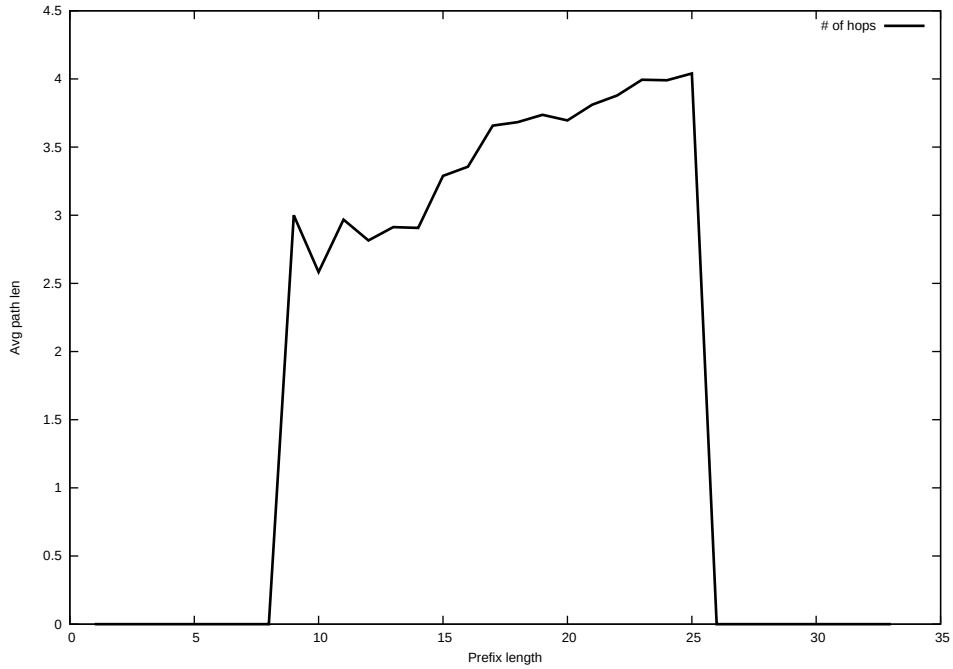
2015-02-01



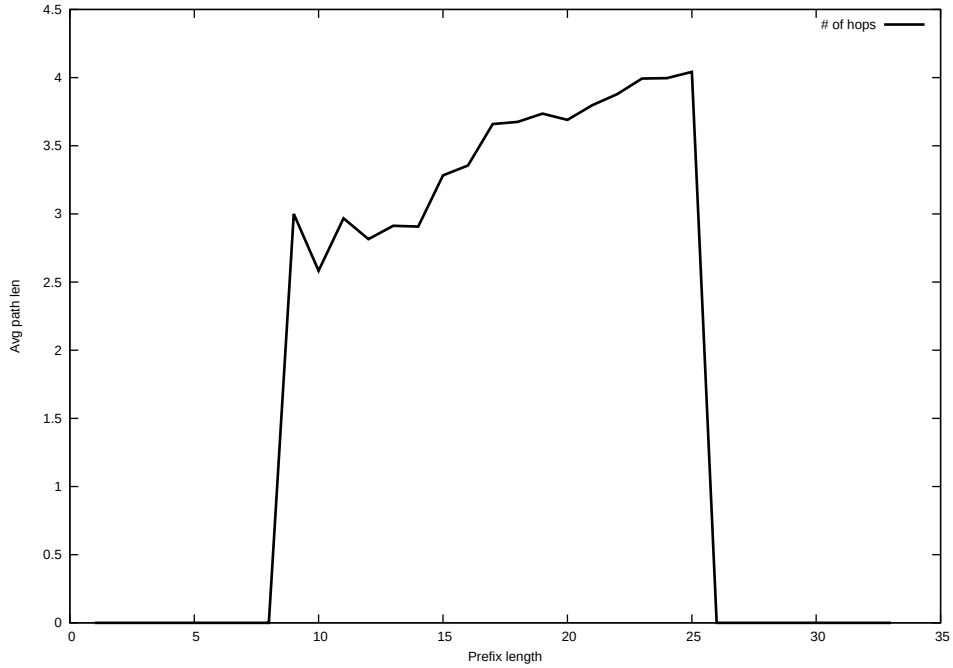
2015-02-02



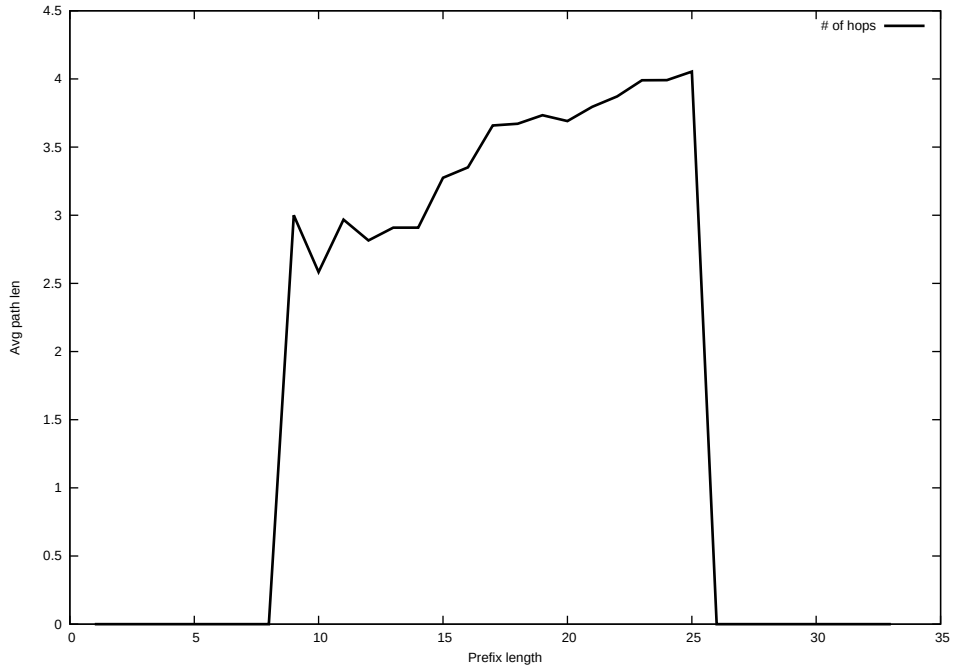
2015-02-03



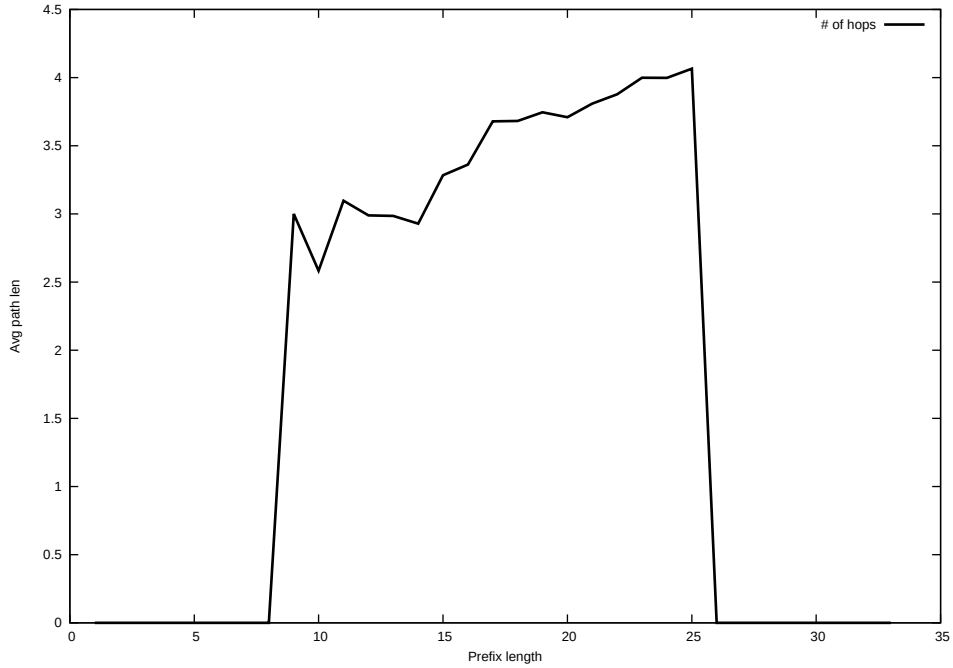
2015-02-04



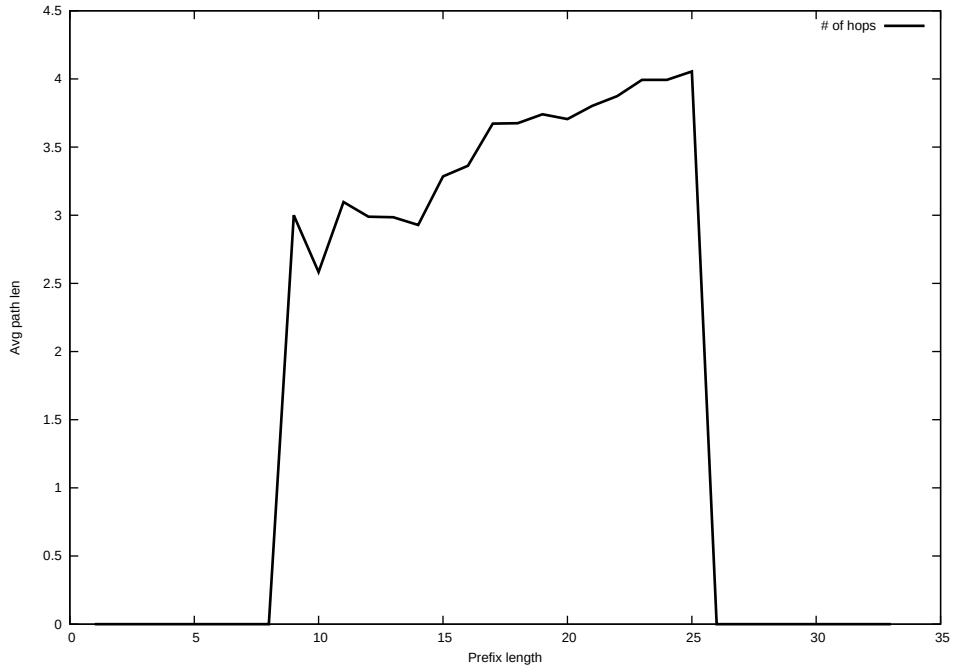
2015-02-05



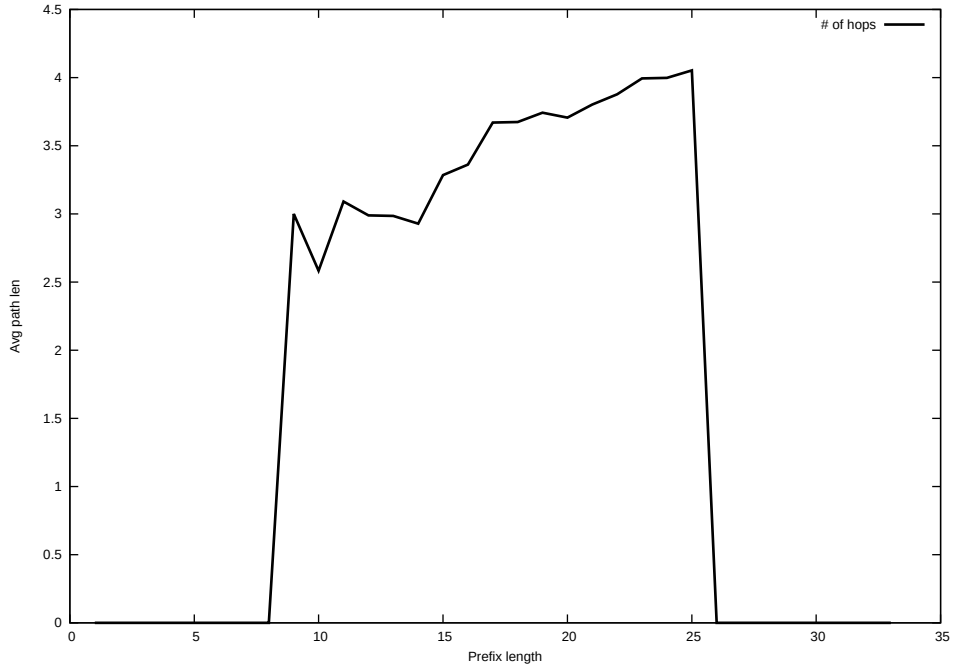
2015-02-06



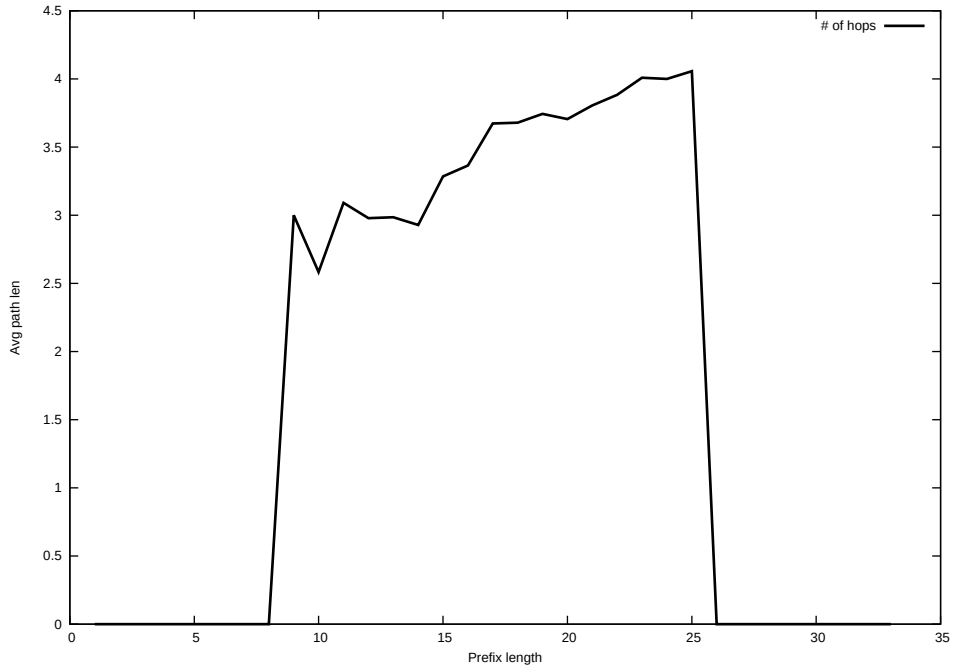
2015-02-07



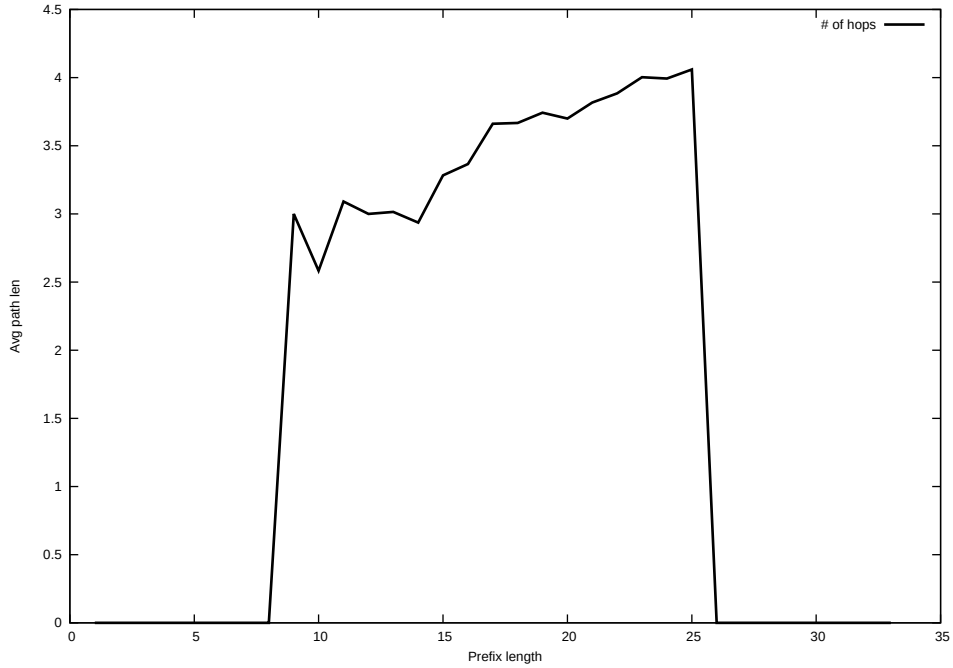
2015-02-08



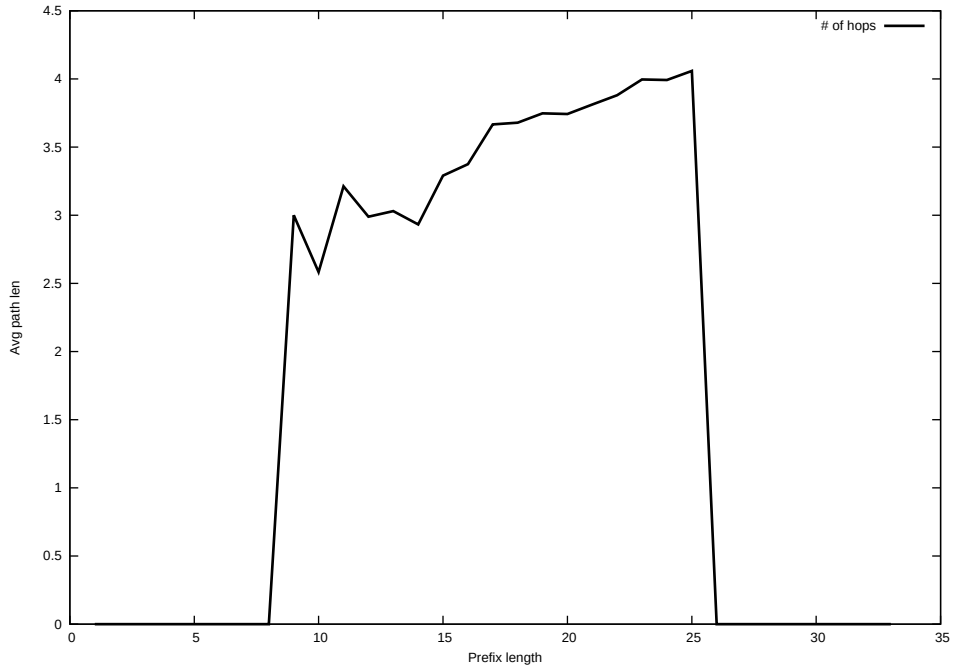
2015-02-09



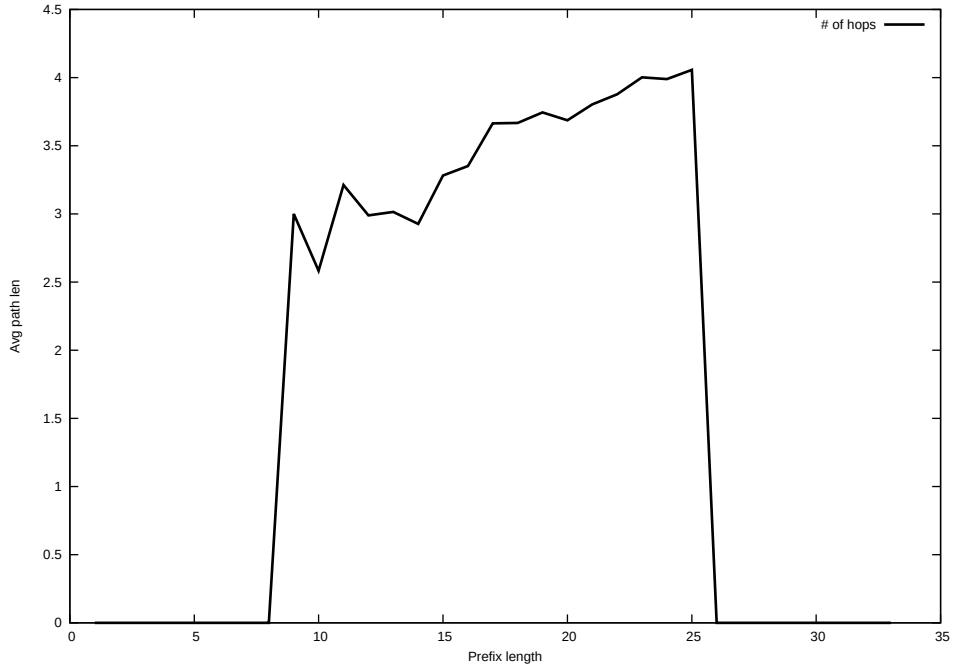
2015-02-10



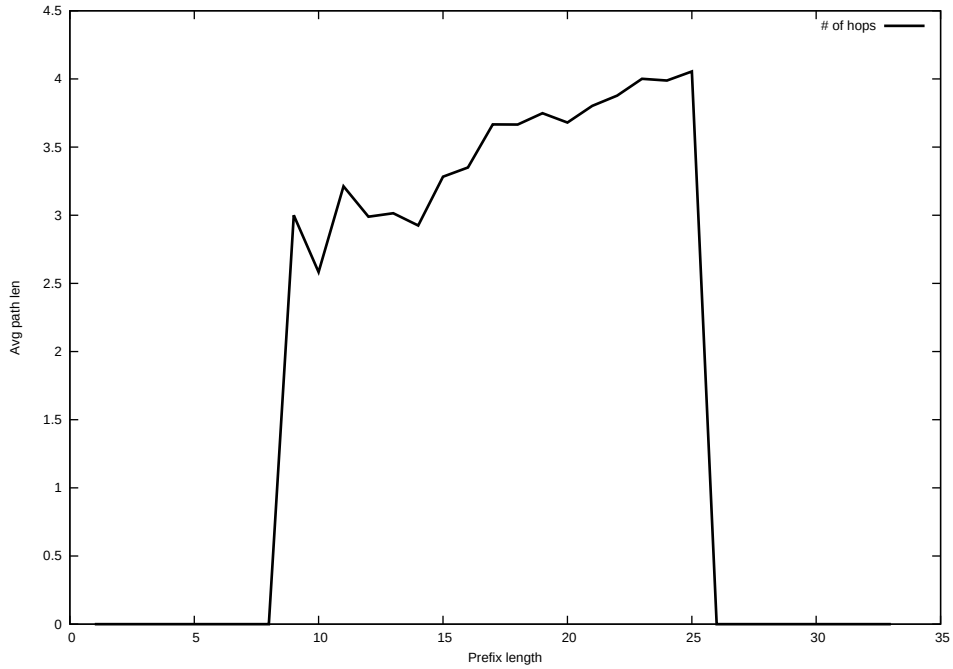
2015-02-11



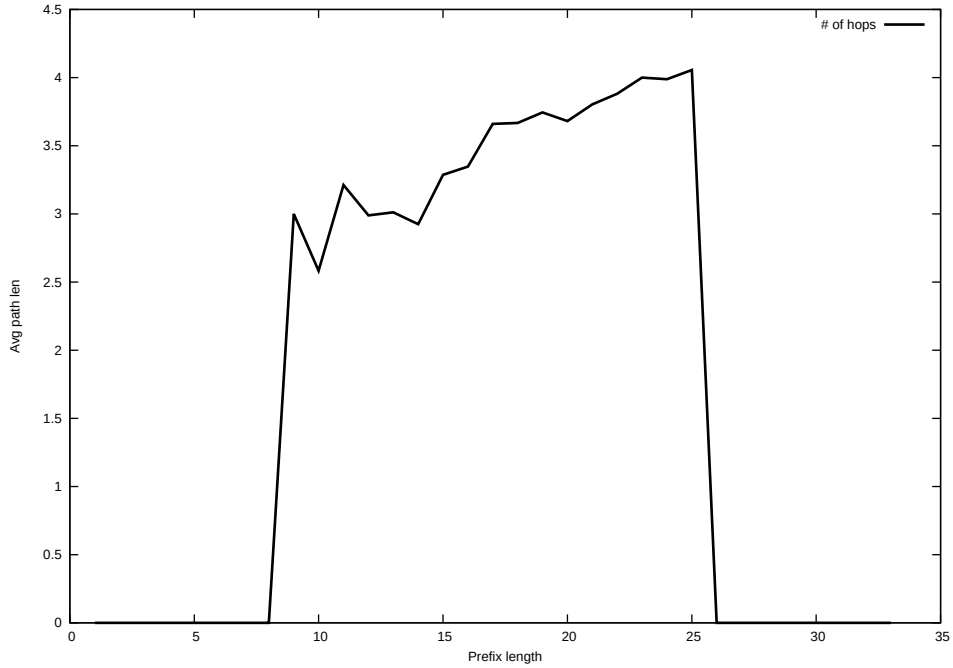
2015-02-12



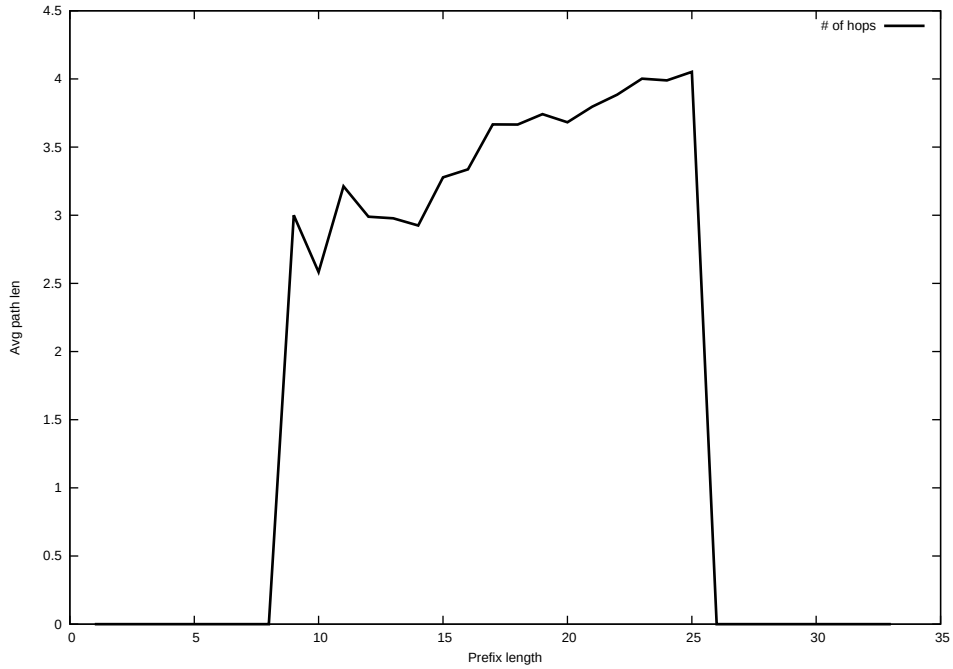
2015-02-13



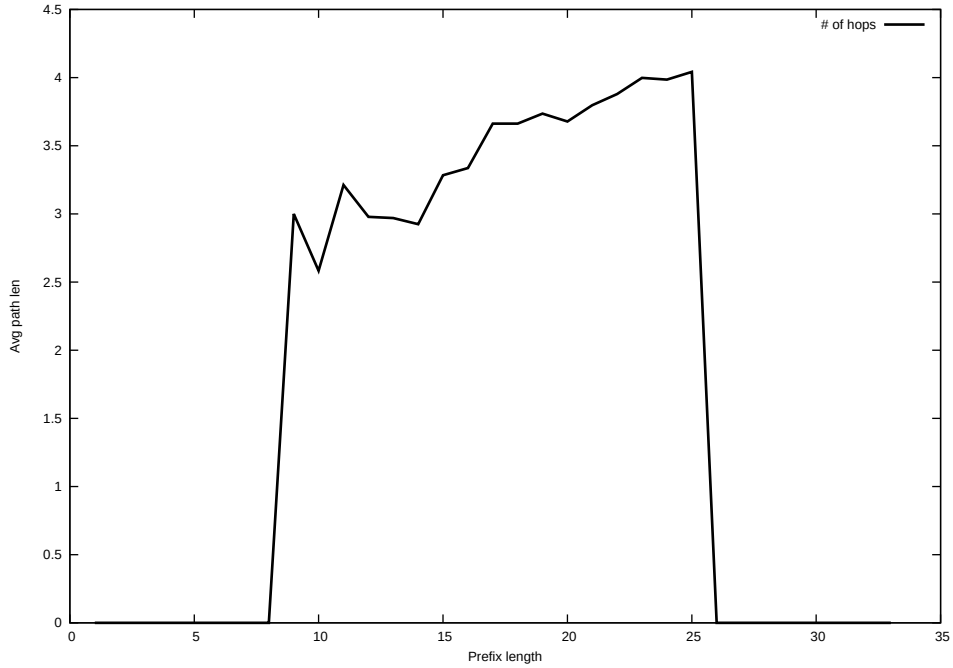
2015-02-14



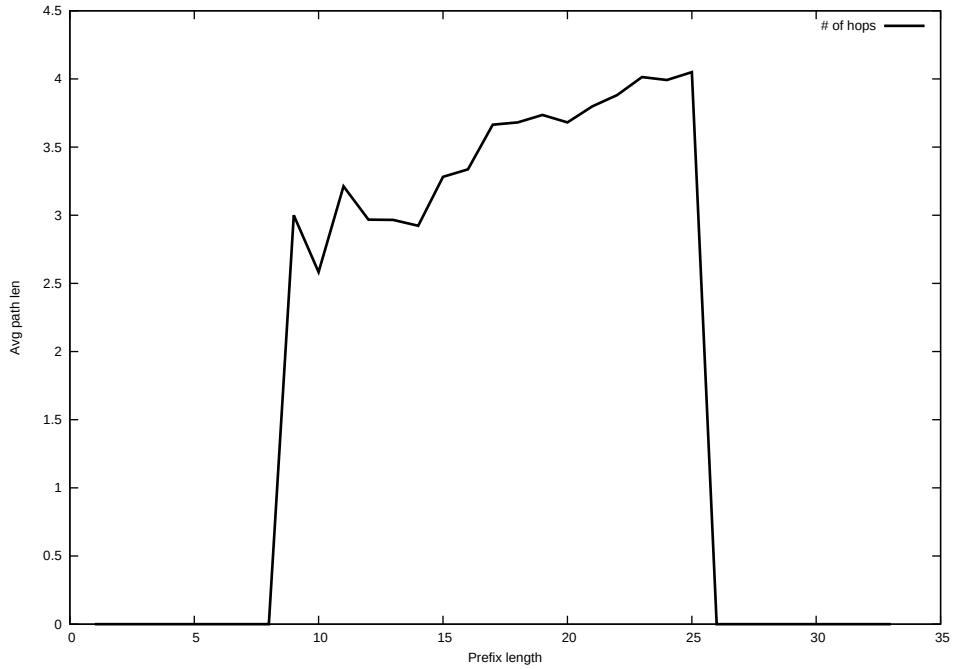
2015-02-15



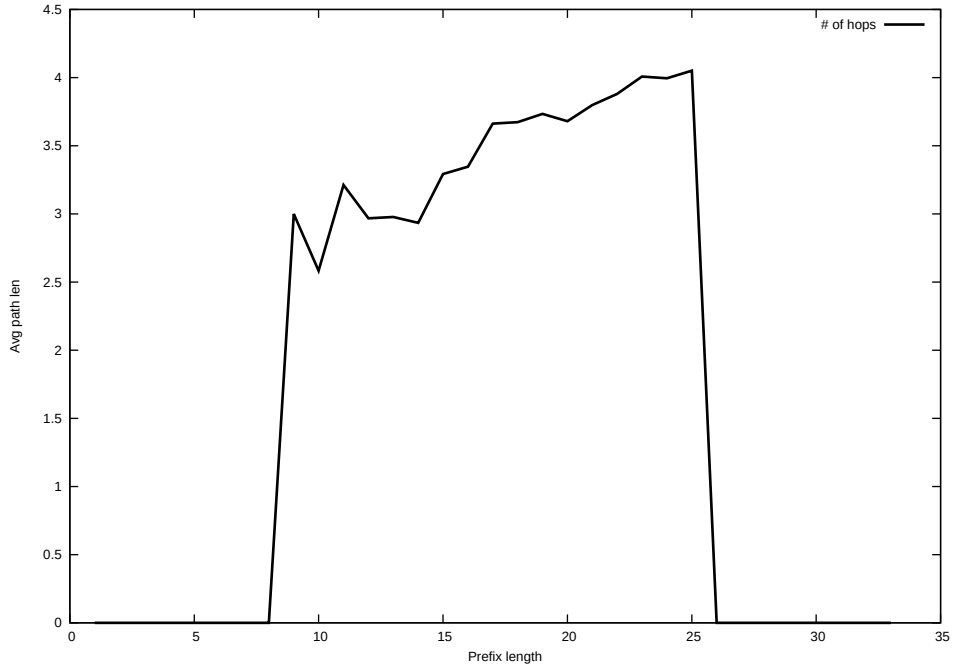
2015-02-16



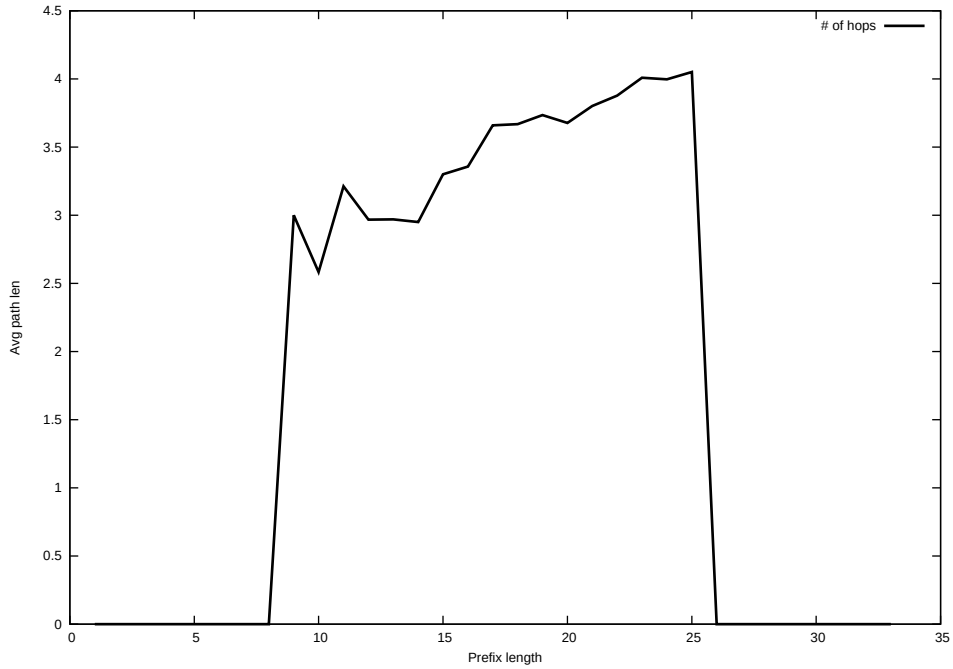
2015-02-17



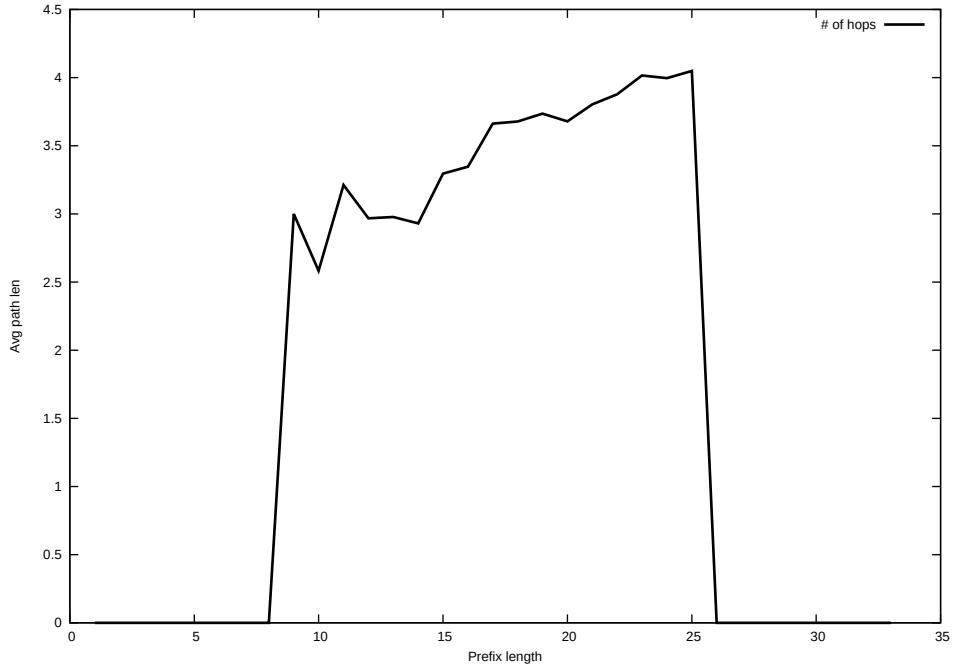
2015-02-18



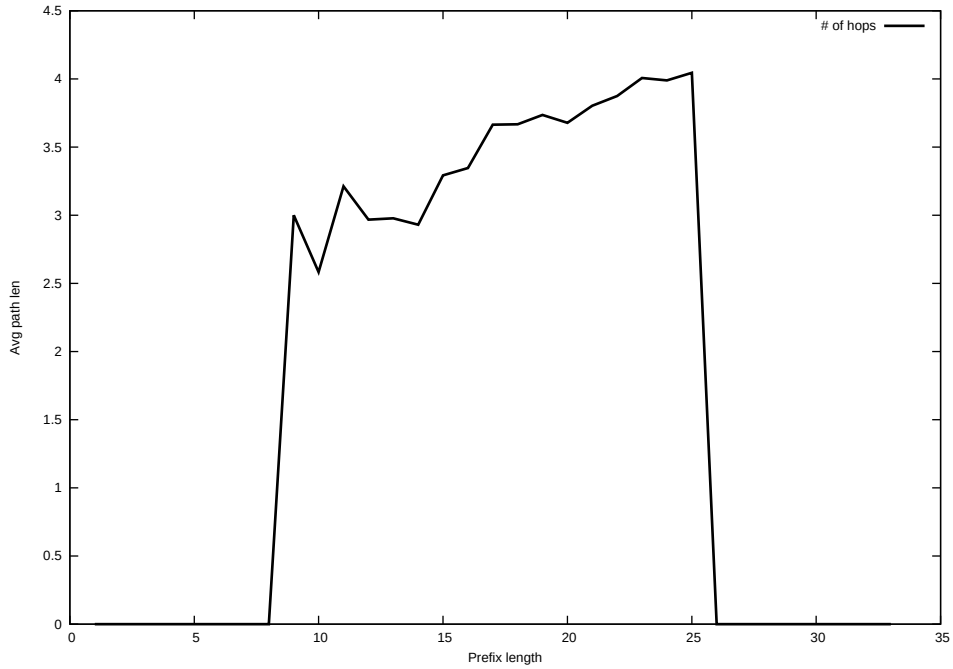
2015-02-19



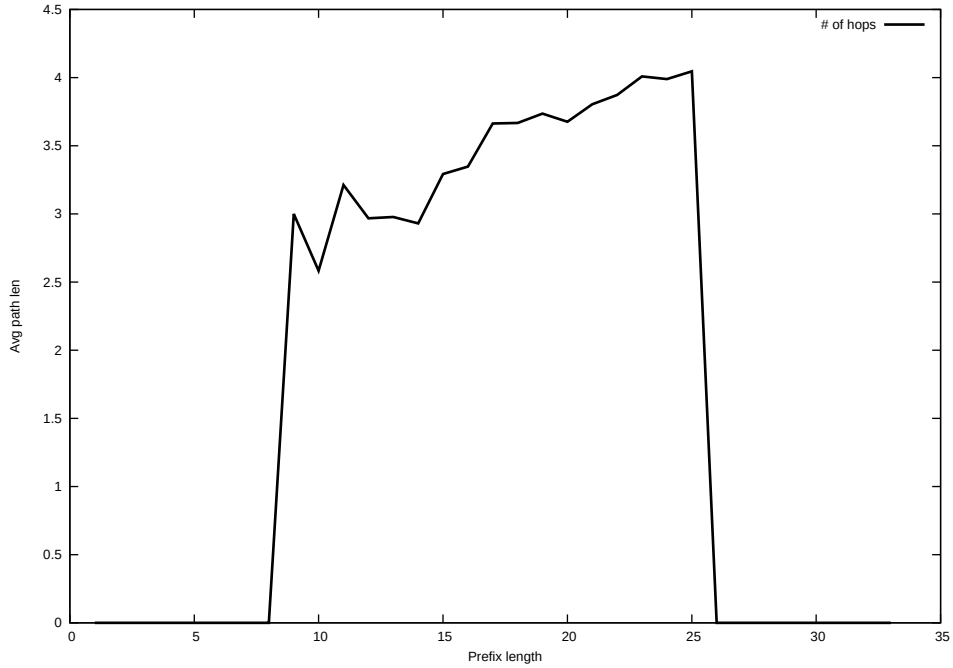
2015-02-20



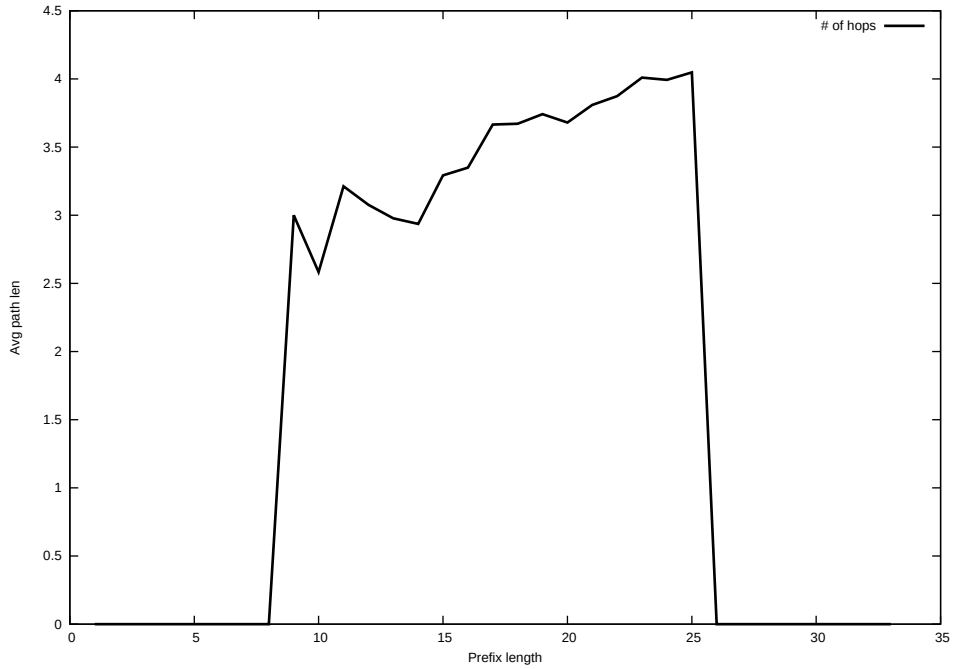
2015-02-21



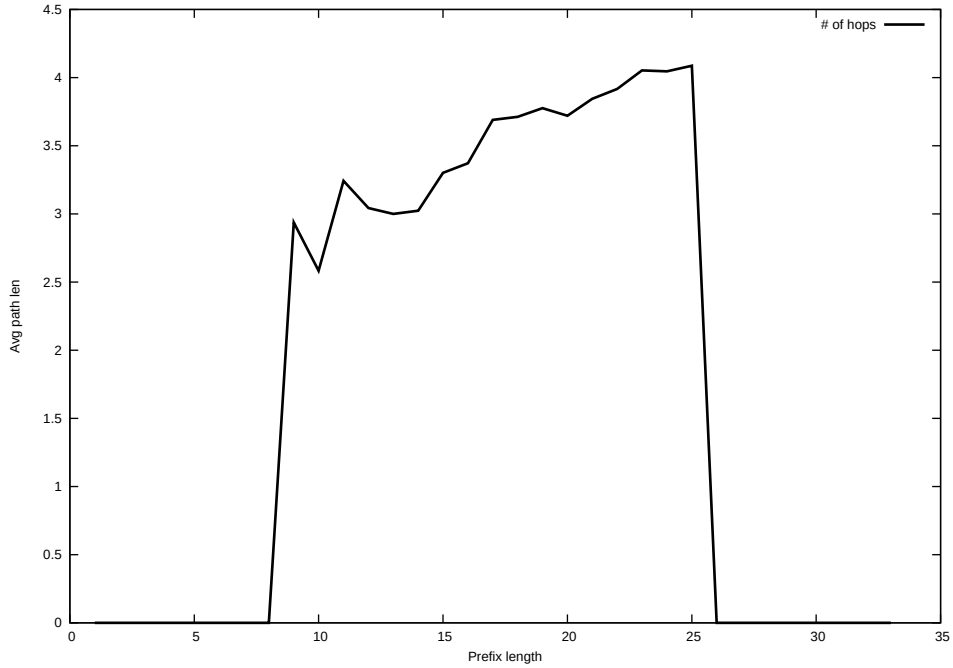
2015-02-22



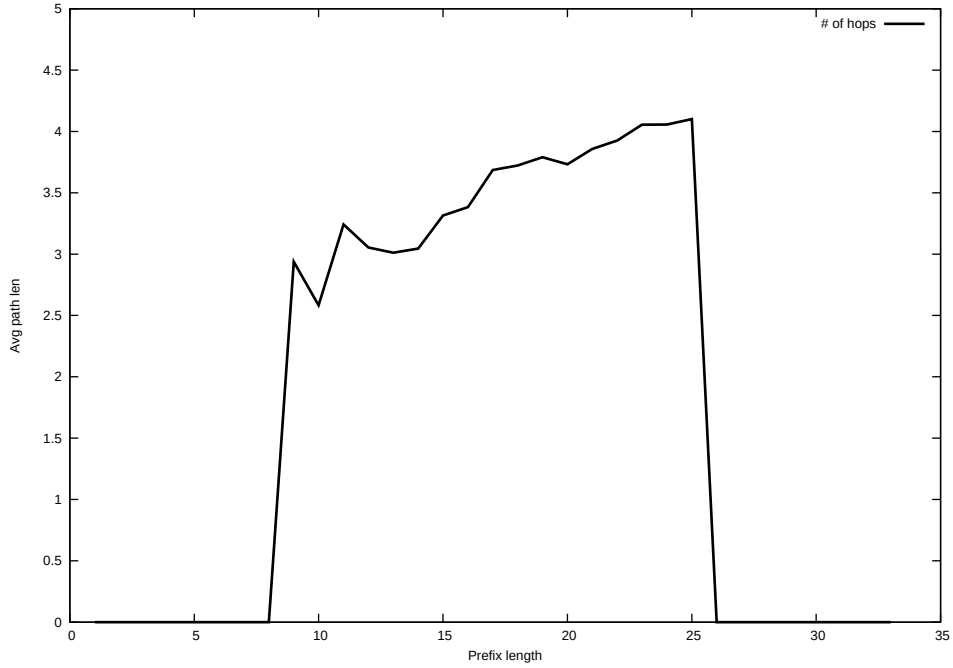
2015-02-23



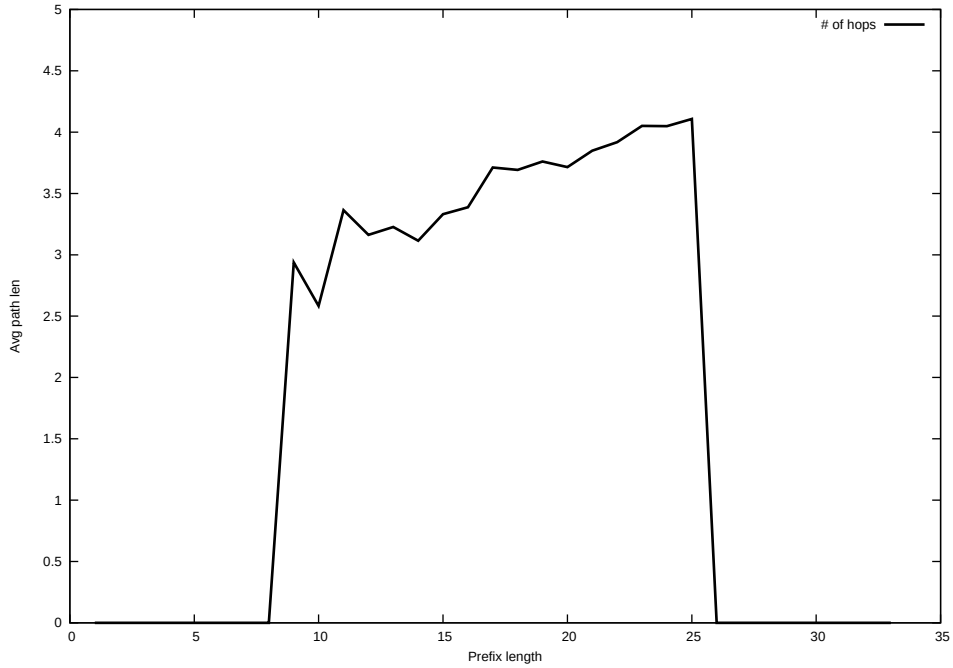
2015-02-24



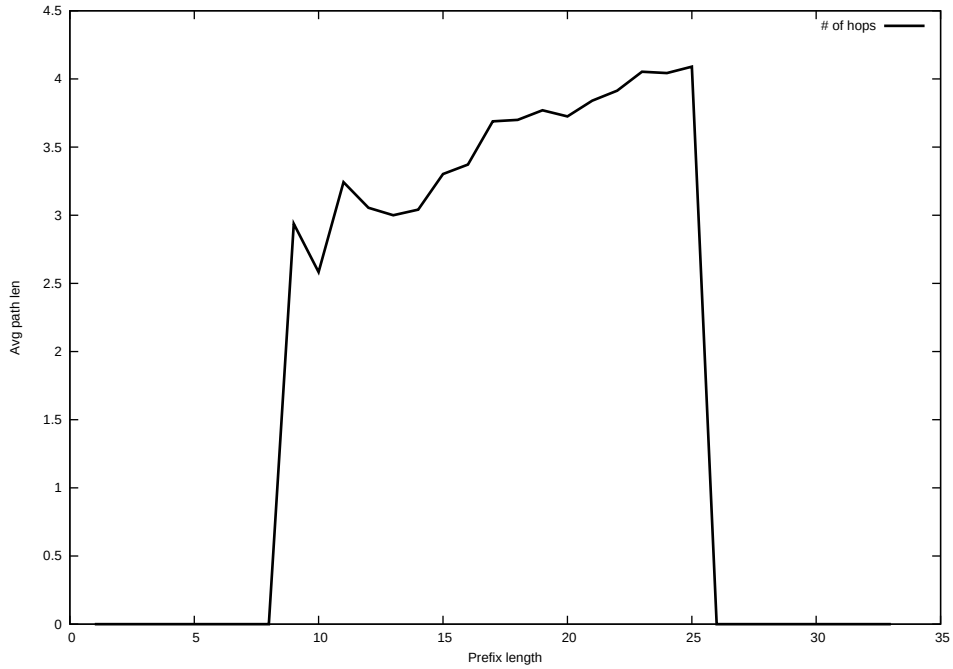
2015-02-25



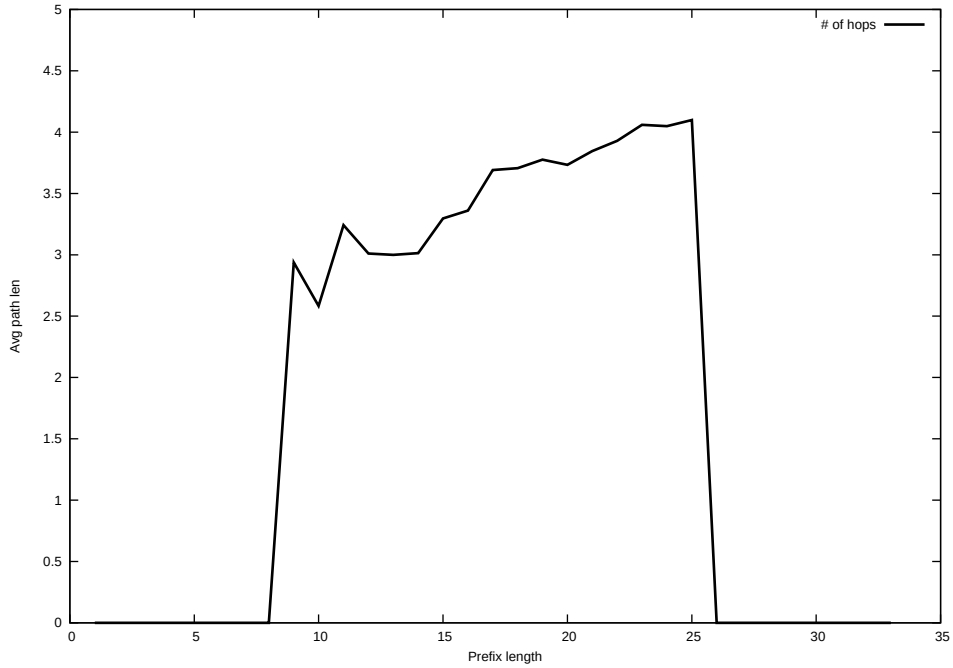
2015-02-26



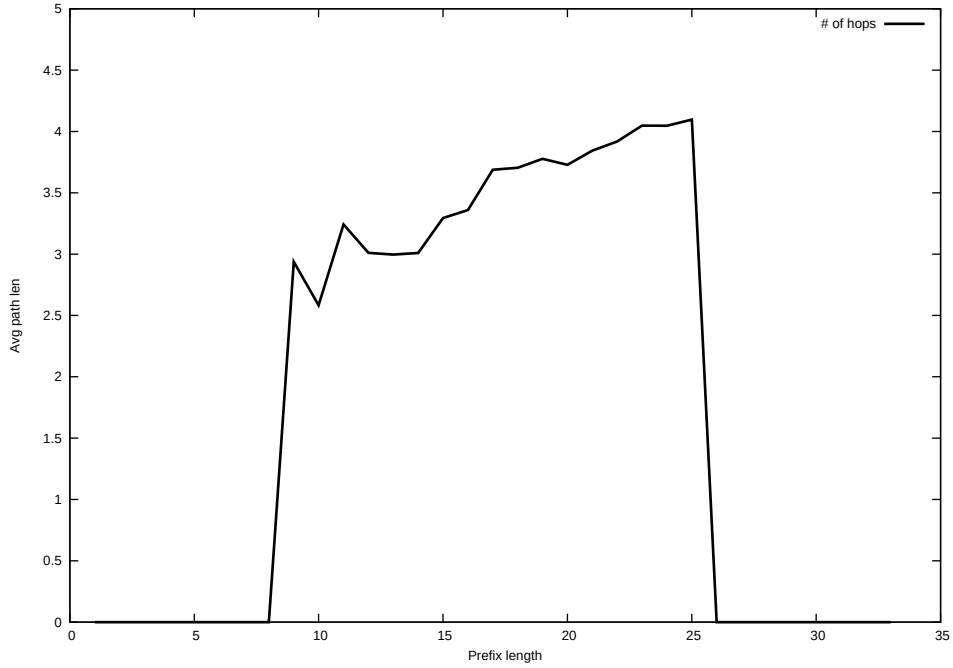
2015-02-27



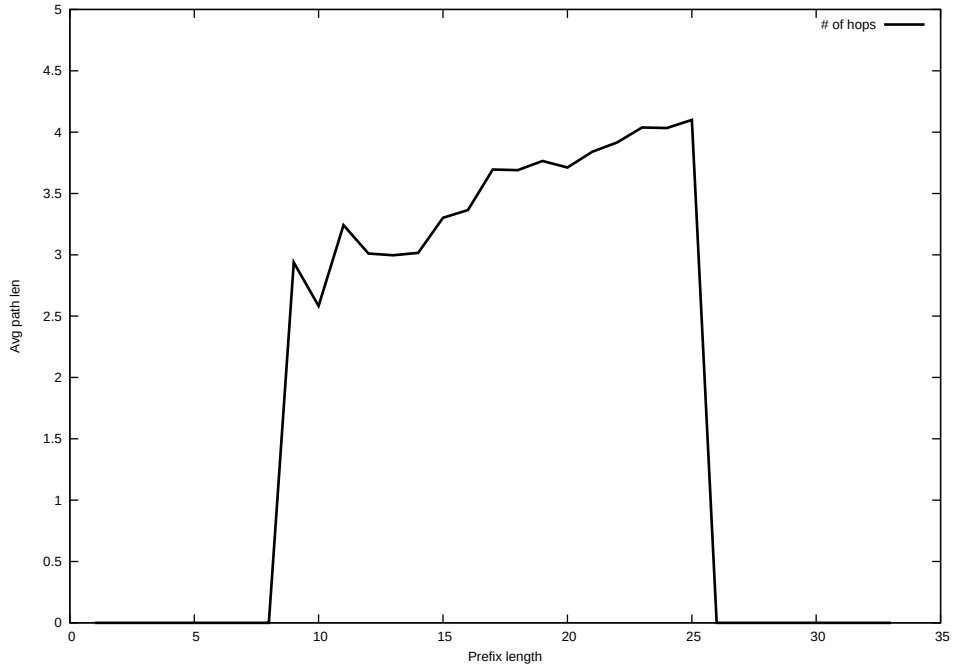
2015-02-28



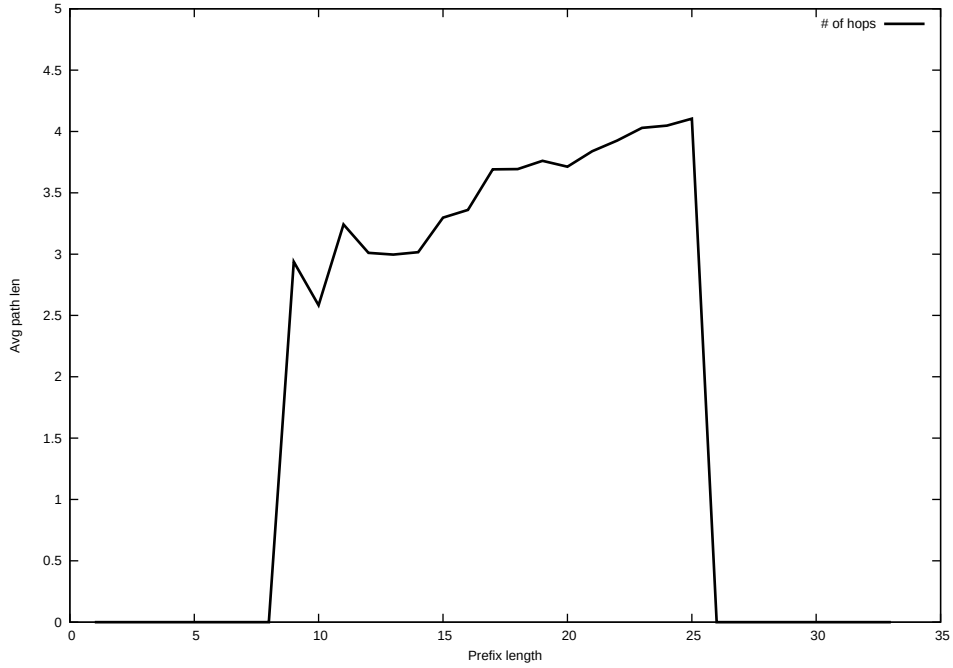
2015-03-01



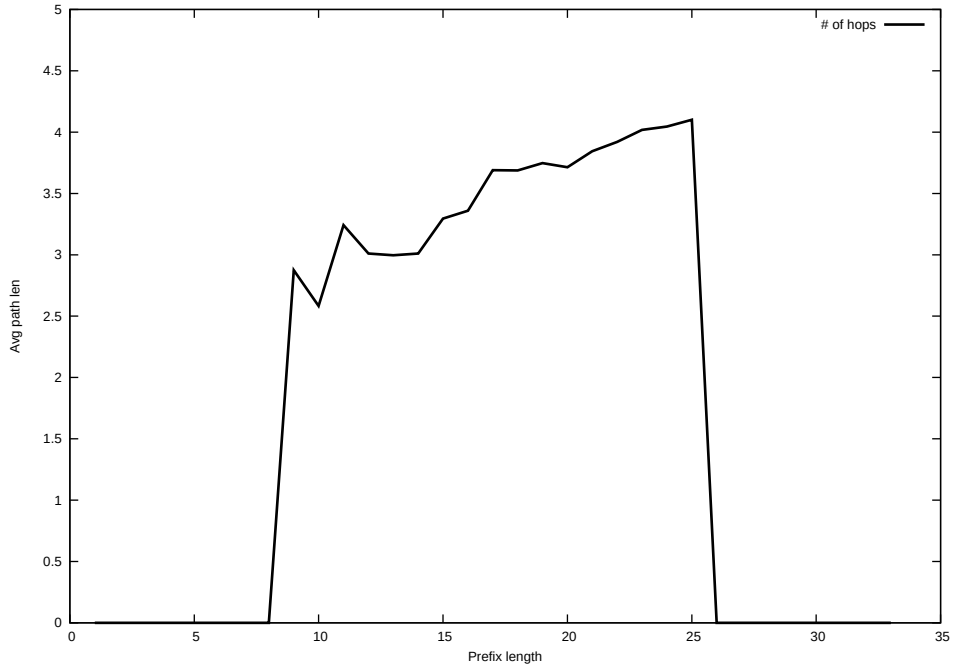
2015-03-02



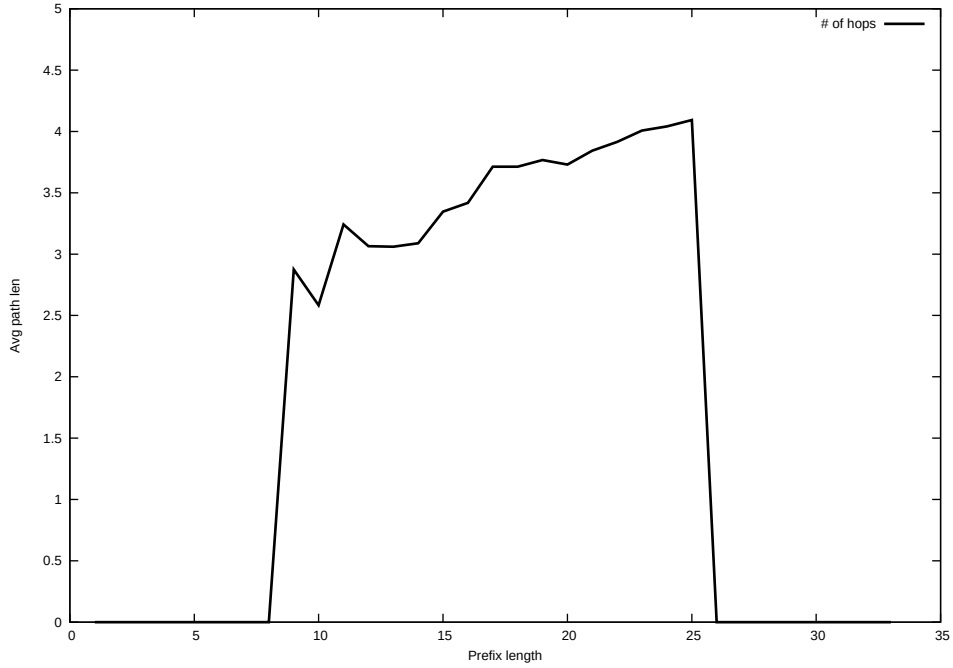
2015-03-03



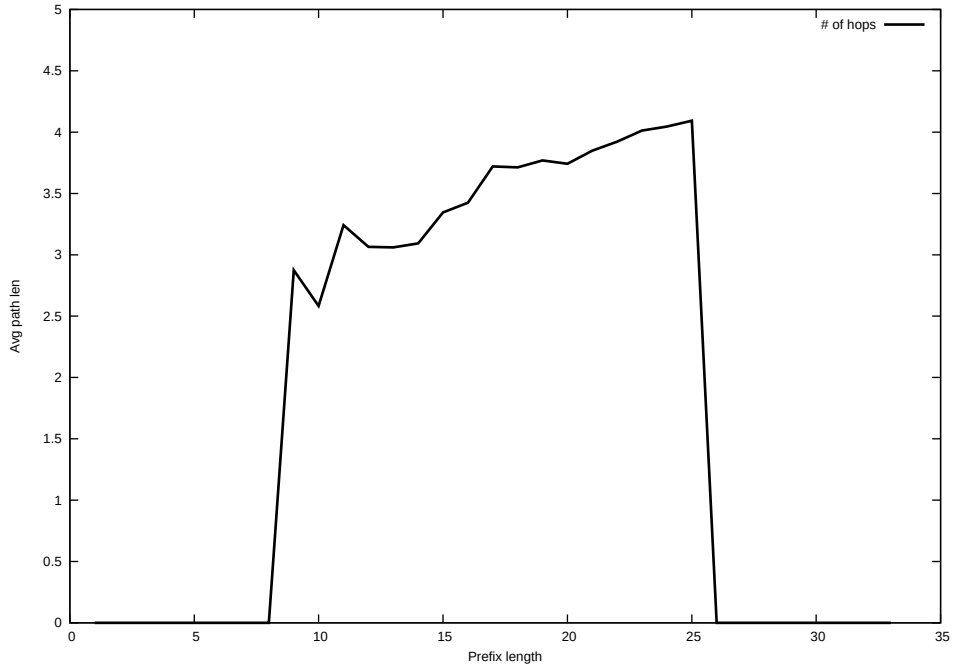
2015-03-04



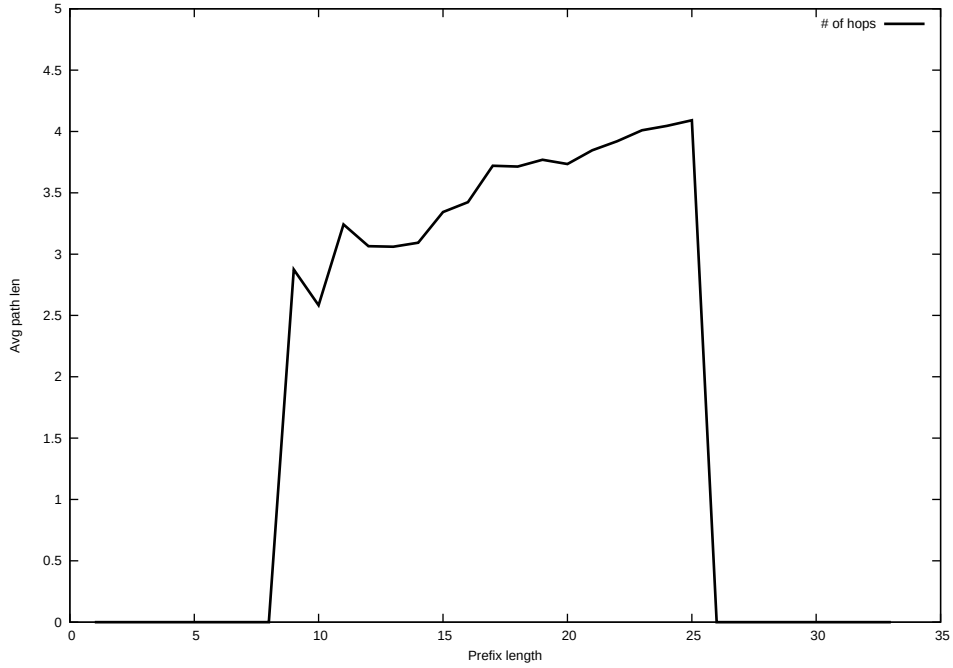
2015-03-05



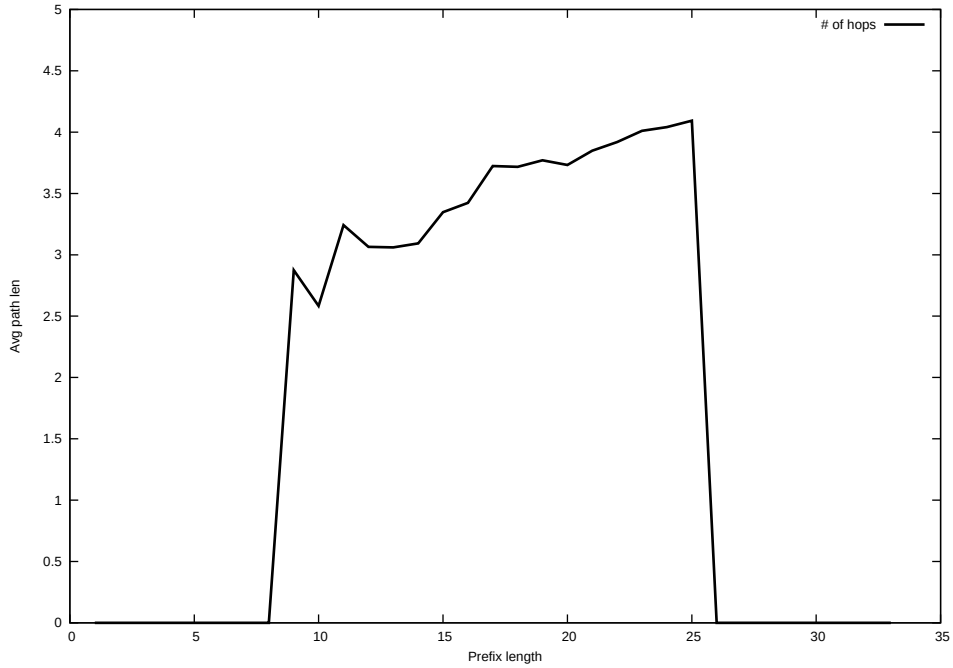
2015-03-06



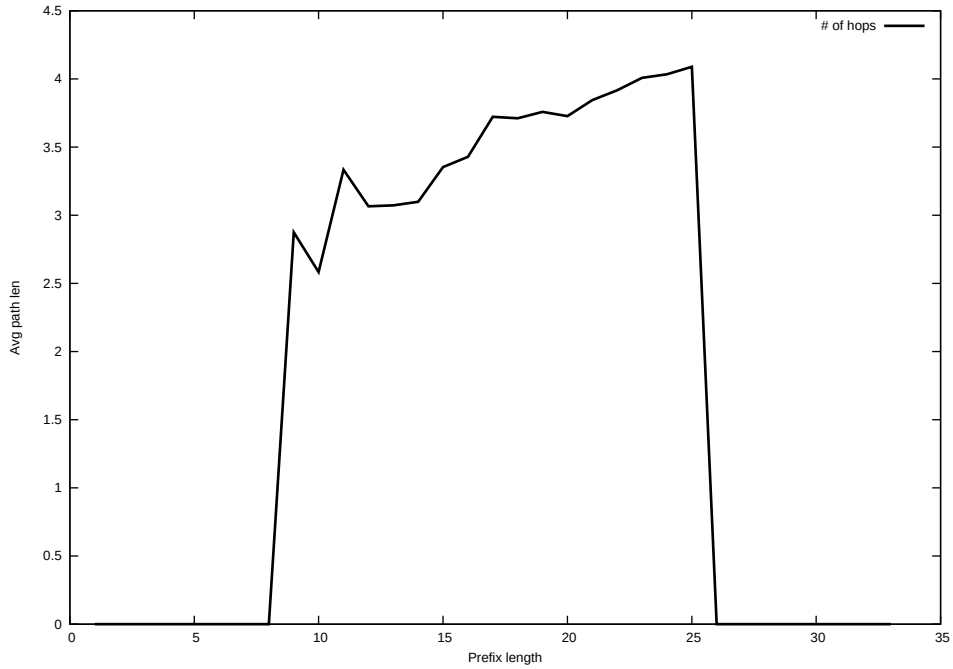
2015-03-07



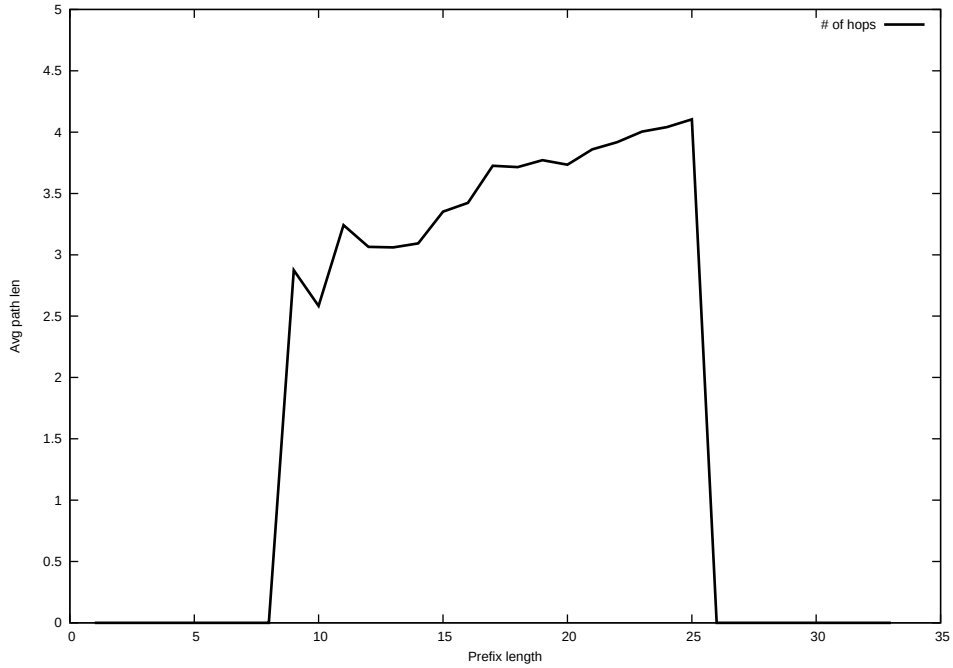
2015-03-08



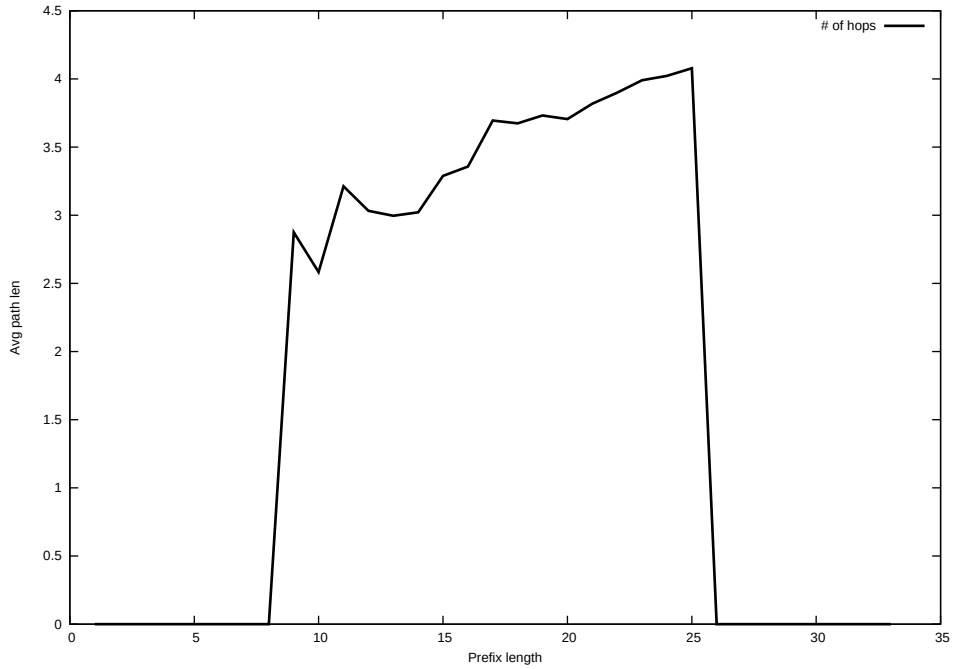
2015-03-09



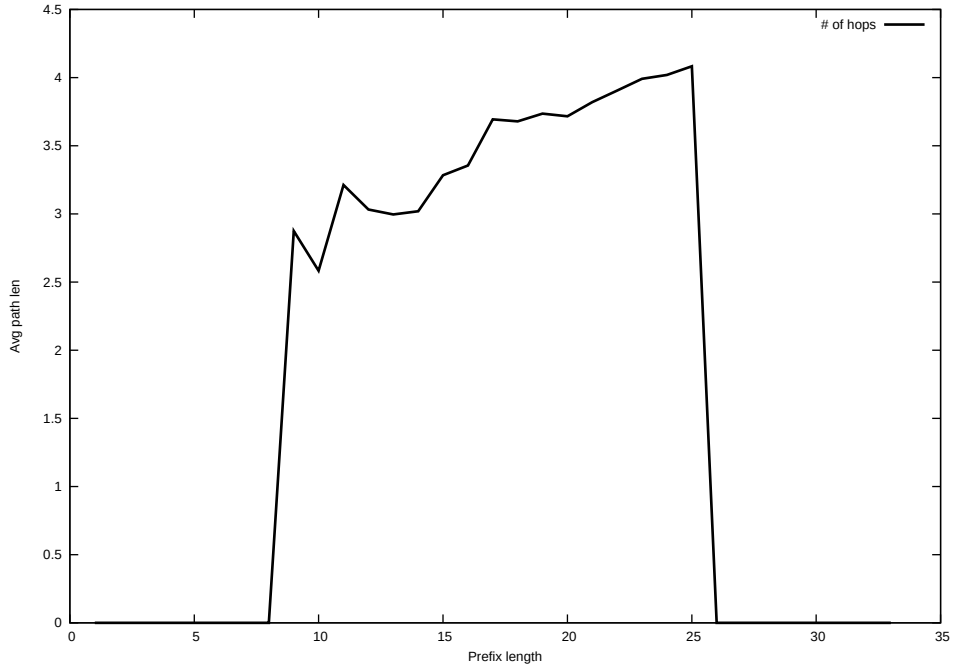
2015-03-10



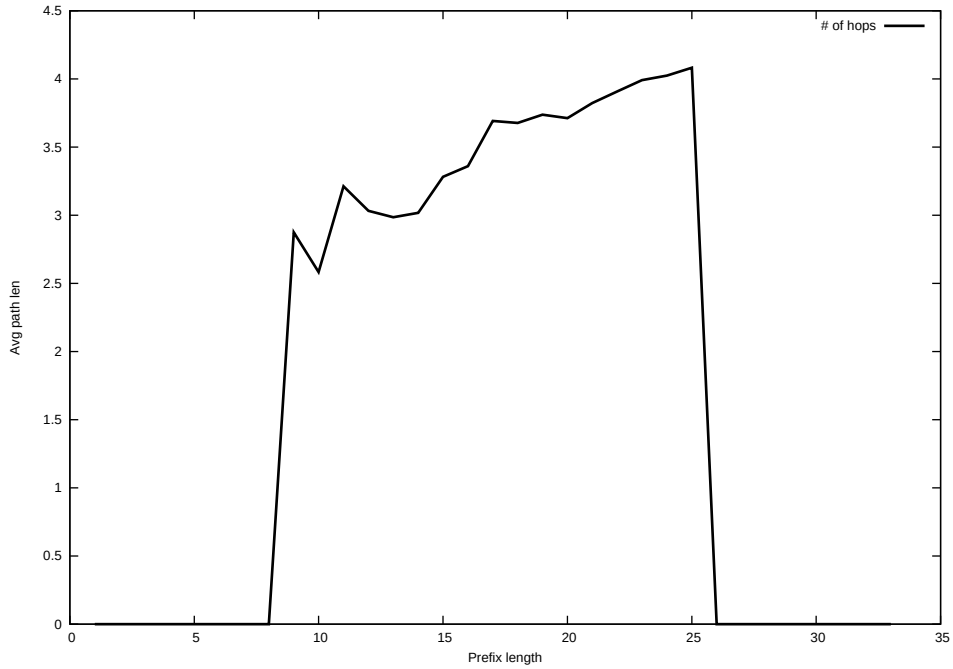
2015-03-11



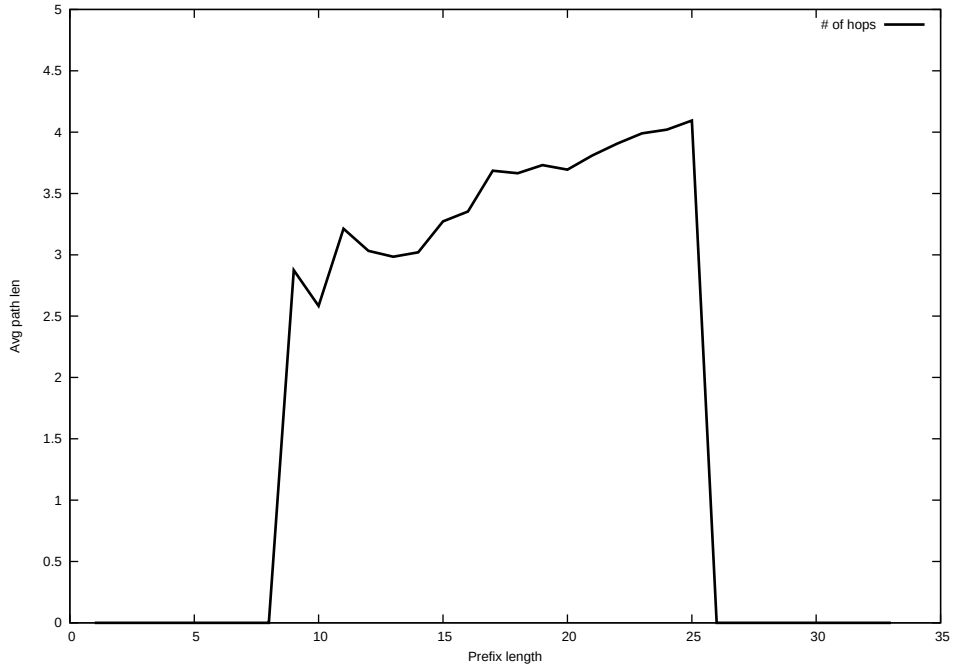
2015-03-12



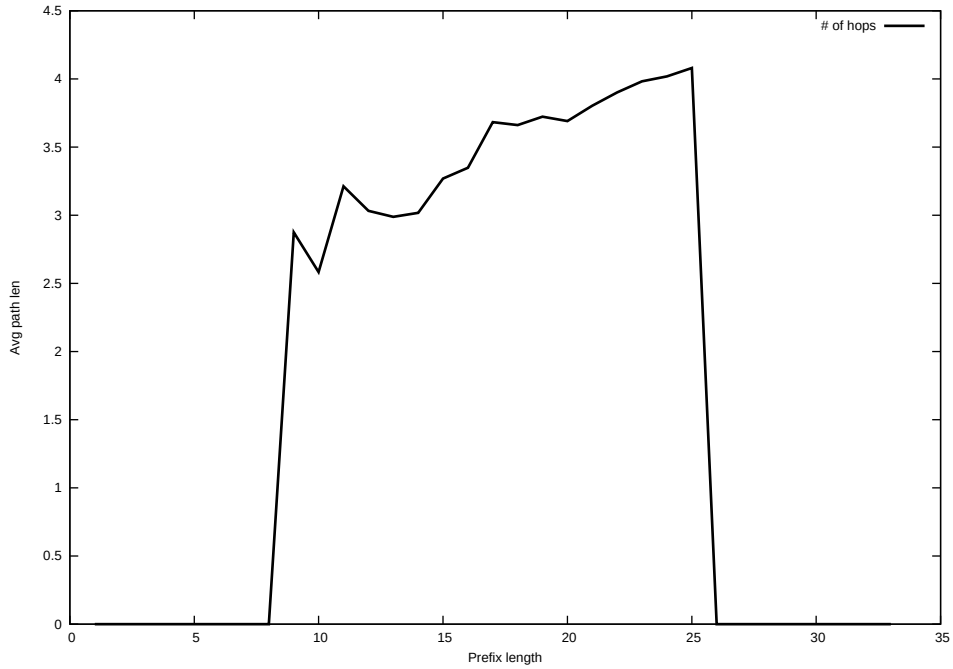
2015-03-13



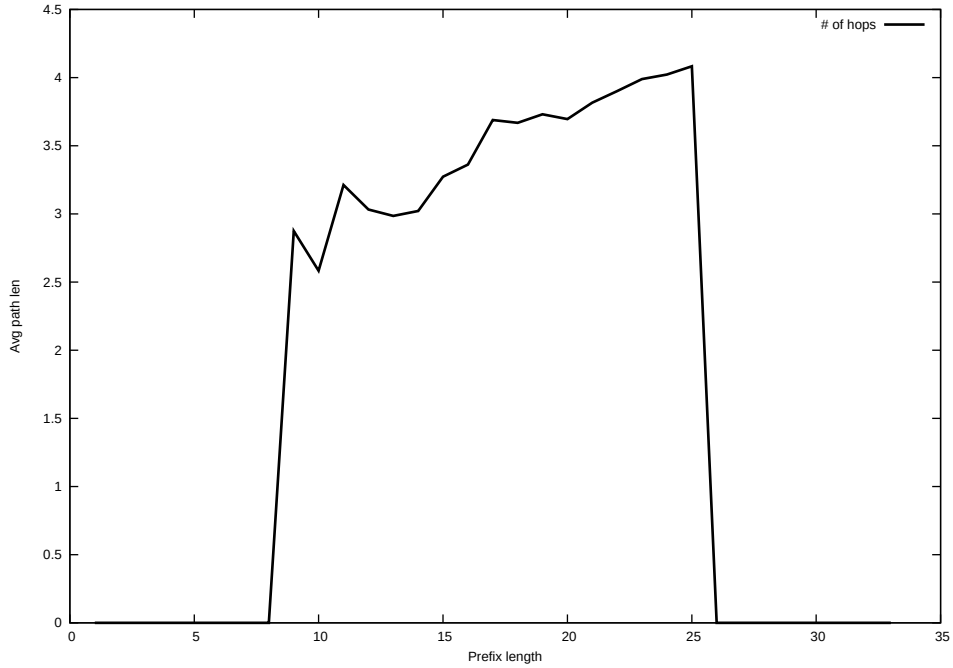
2015-03-14



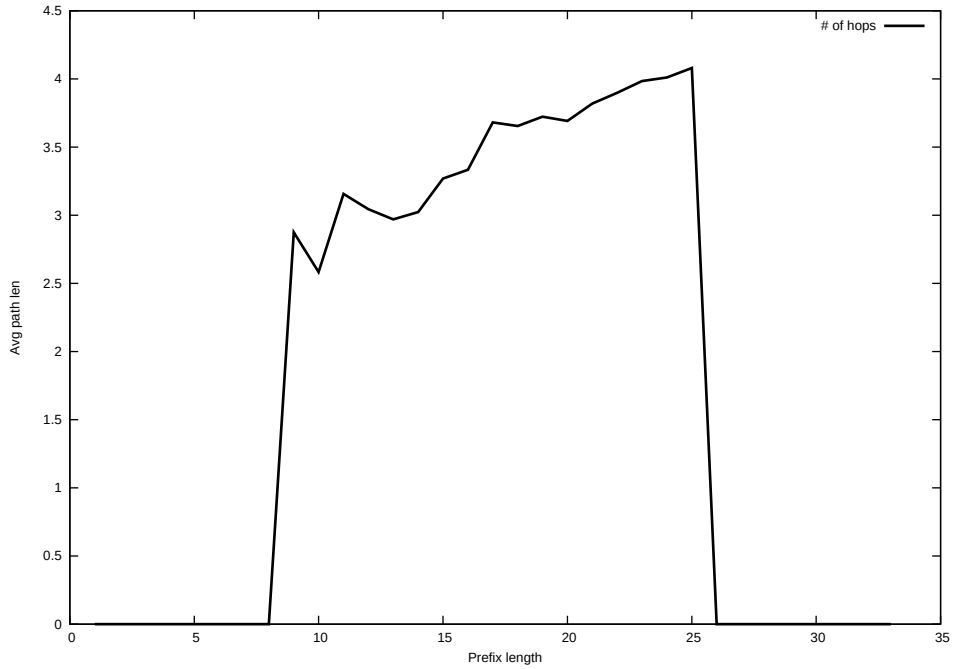
2015-03-15



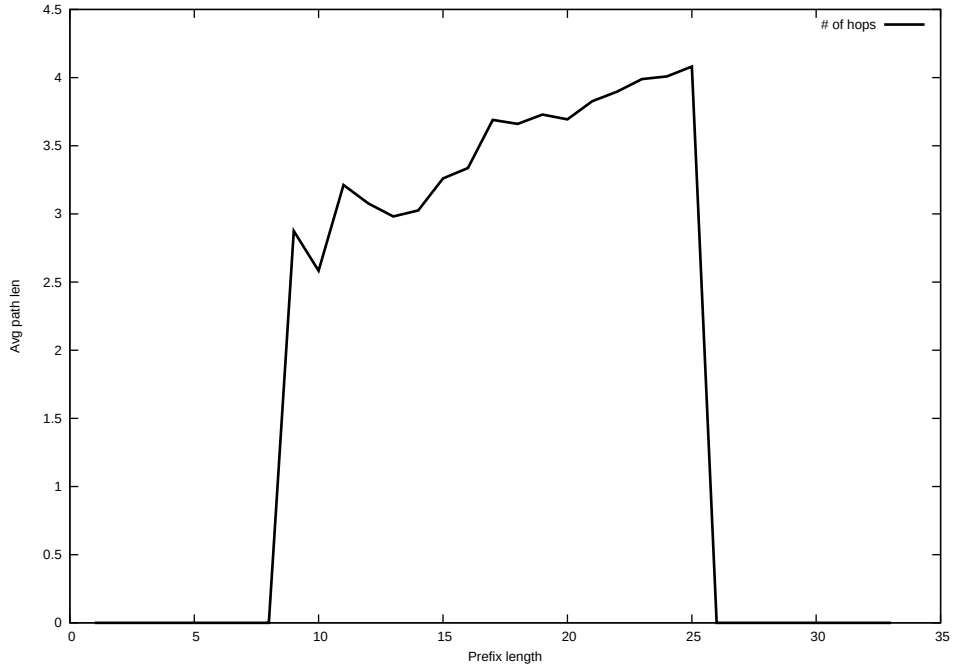
2015-03-16



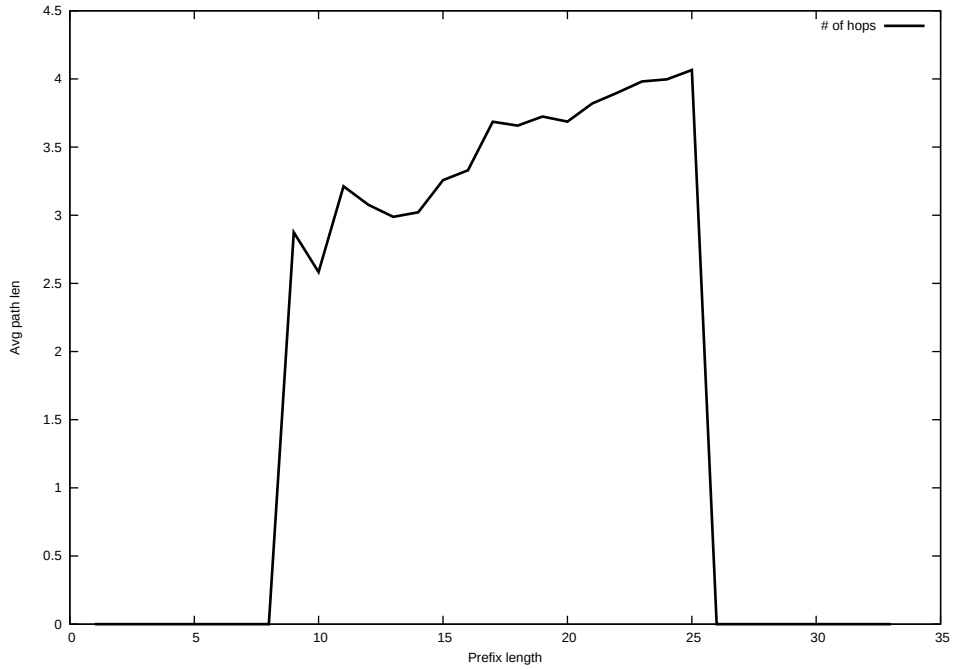
2015-03-17



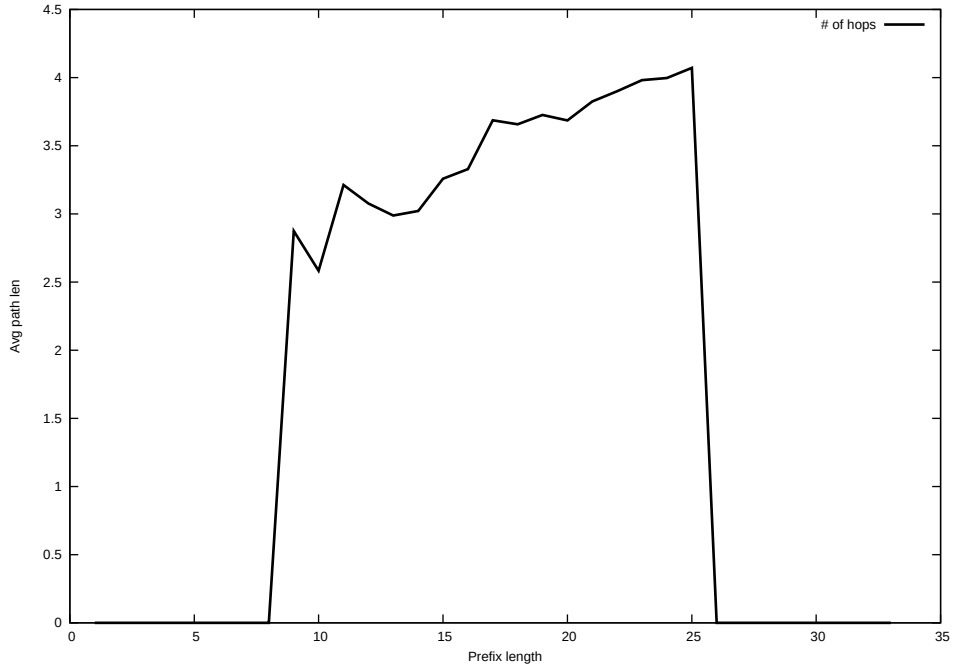
2015-03-18



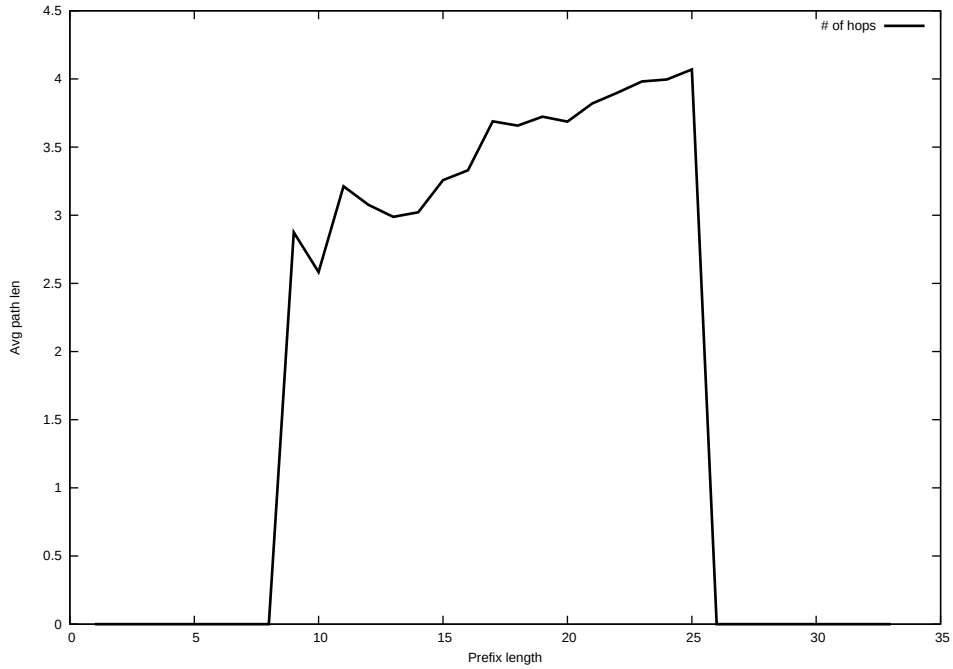
2015-03-19



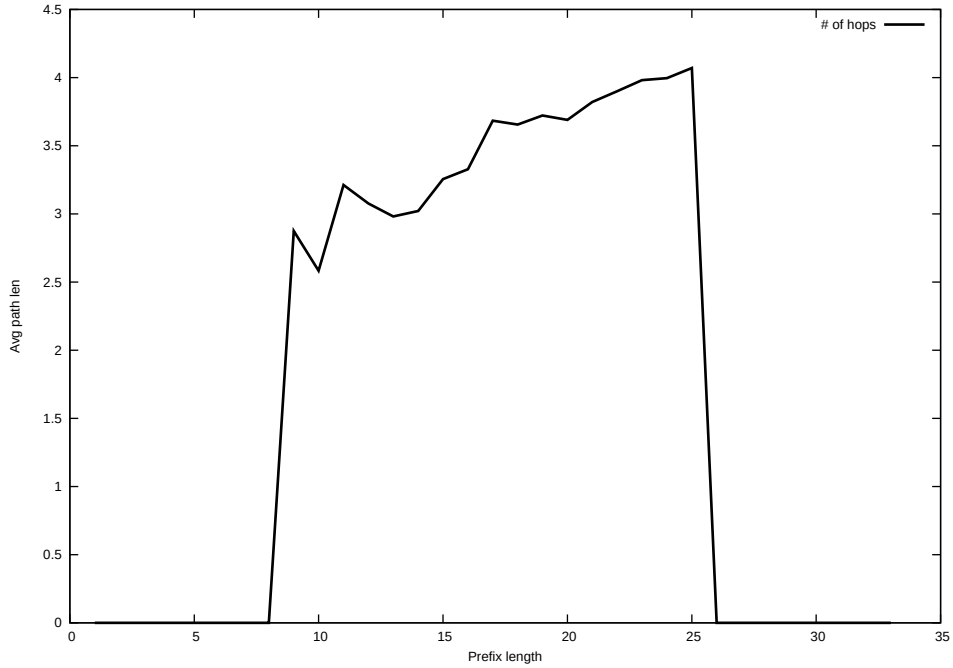
2015-03-20



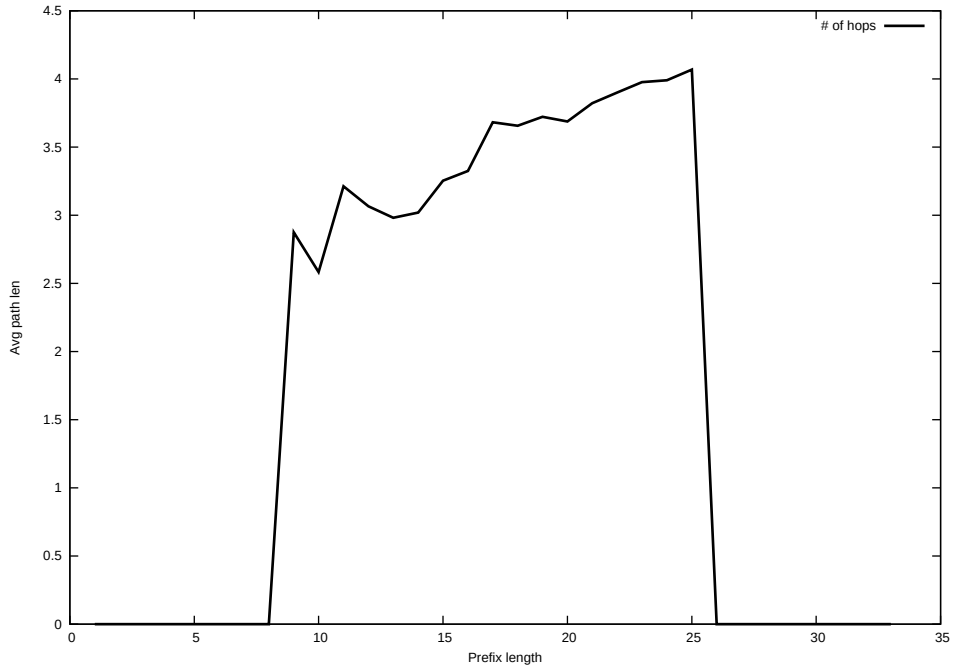
2015-03-21



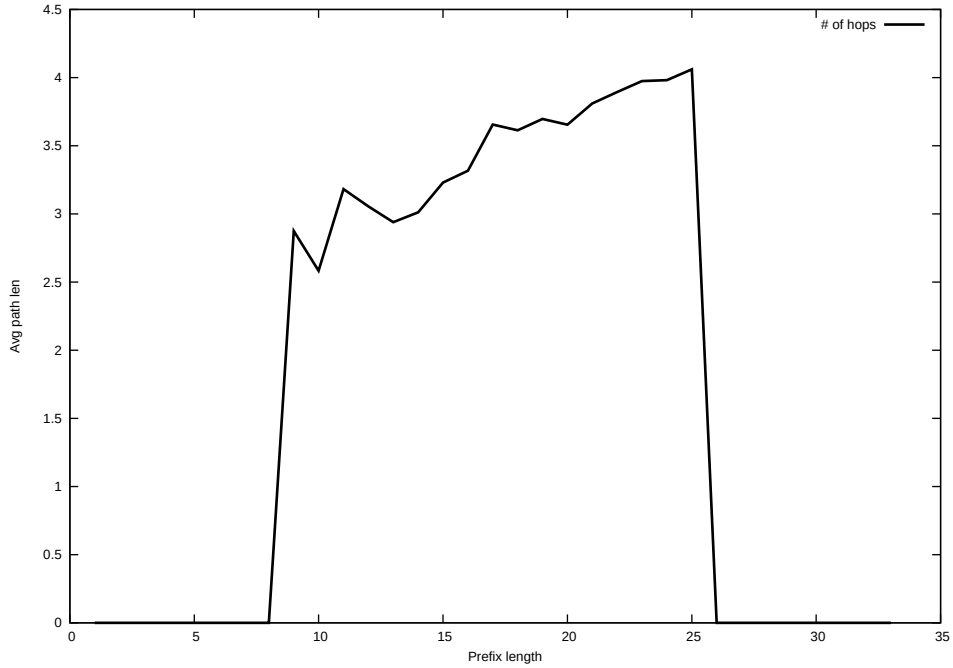
2015-03-22



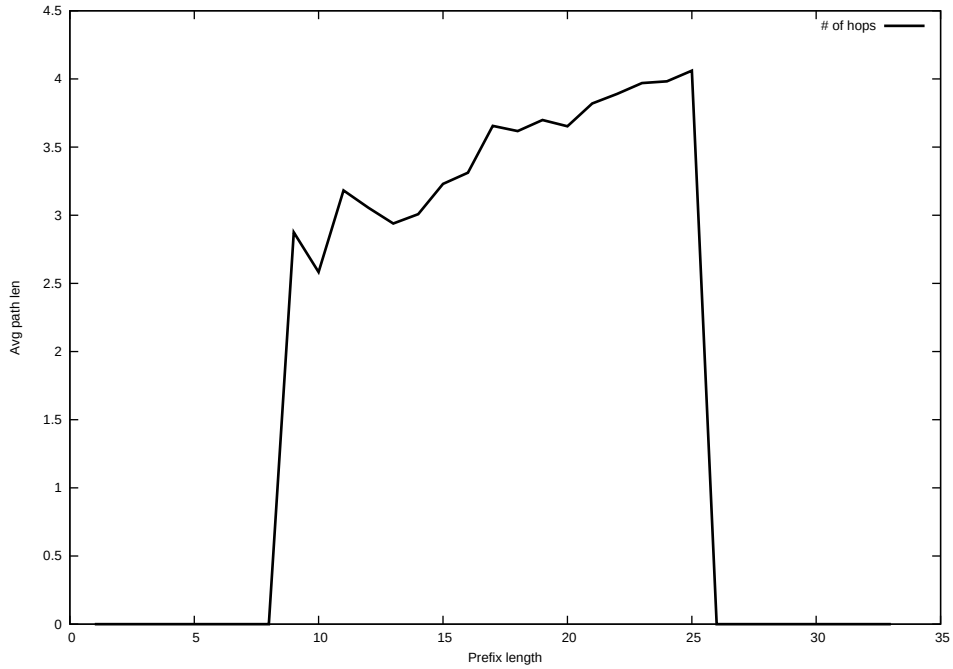
2015-03-23



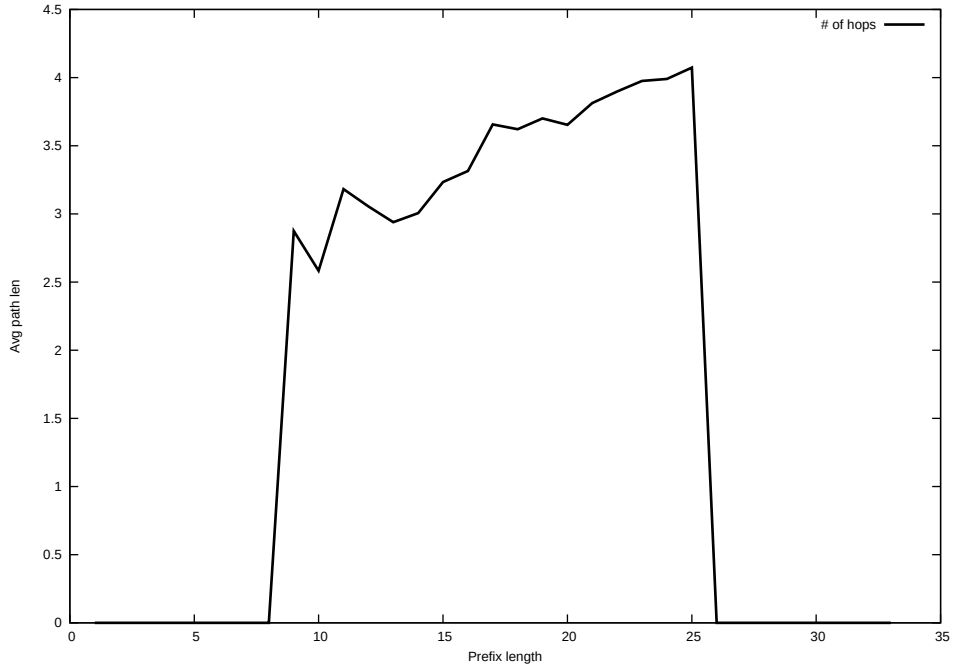
2015-03-24



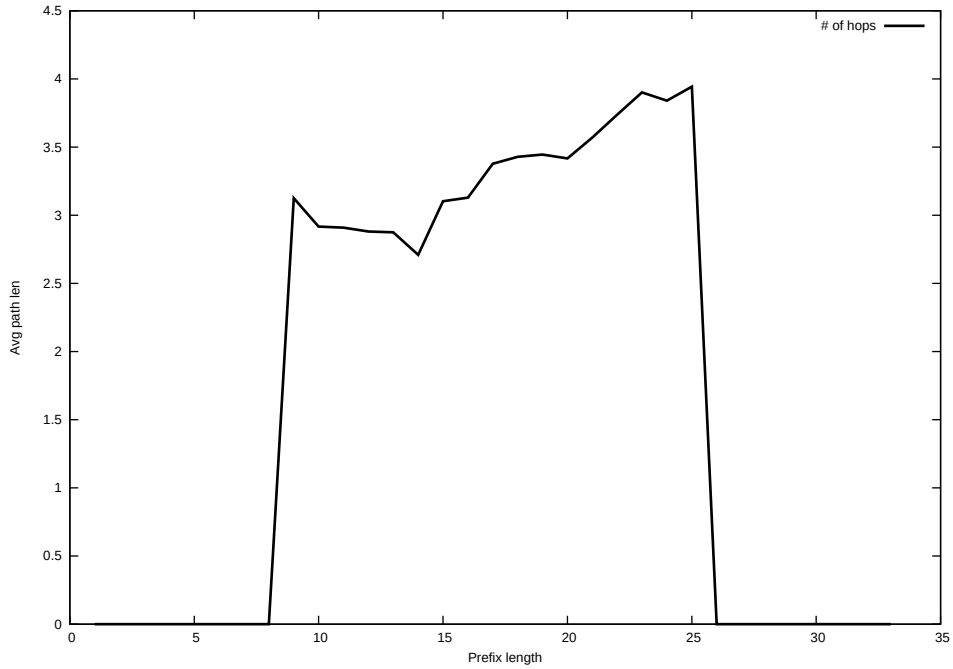
2015-03-25



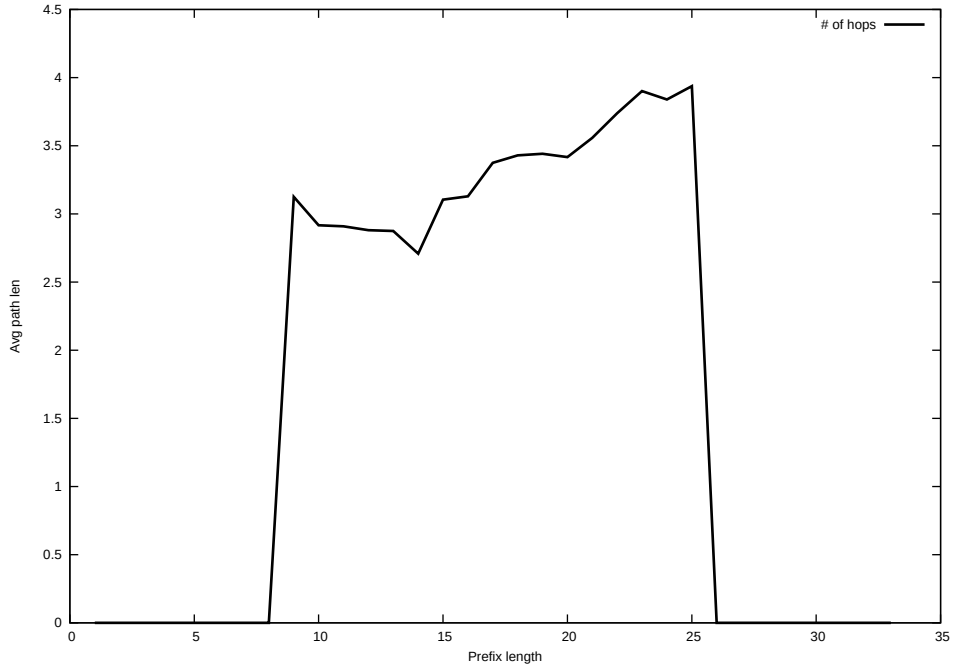
2015-03-26



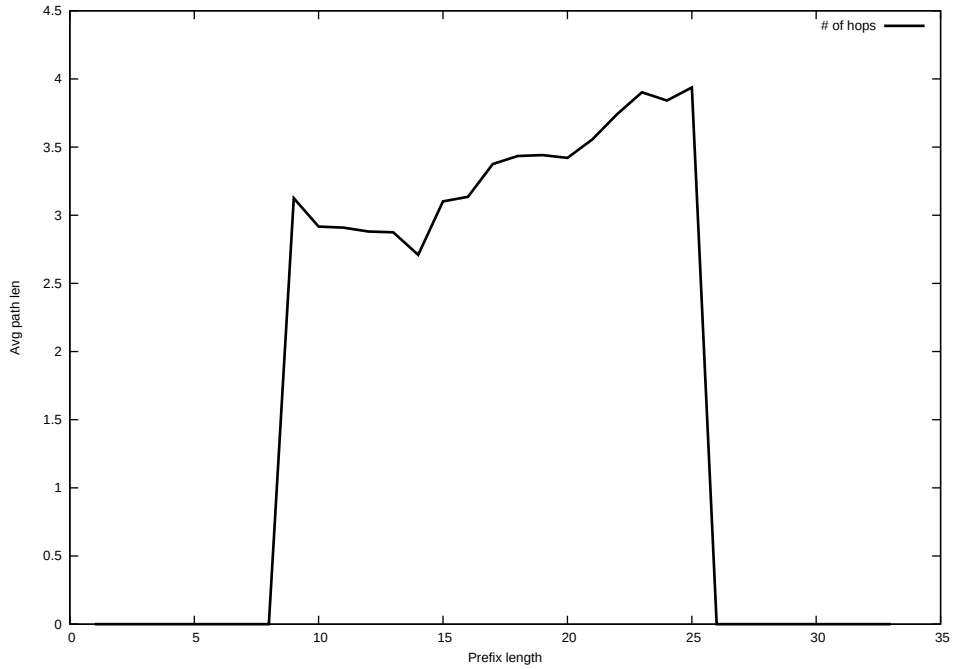
2015-03-27



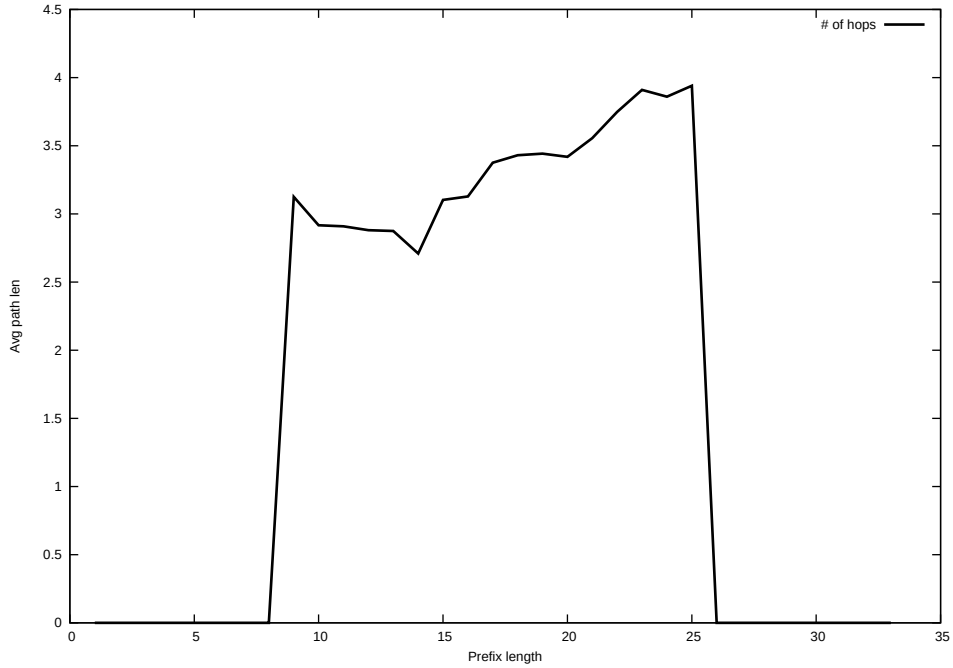
2015-03-28



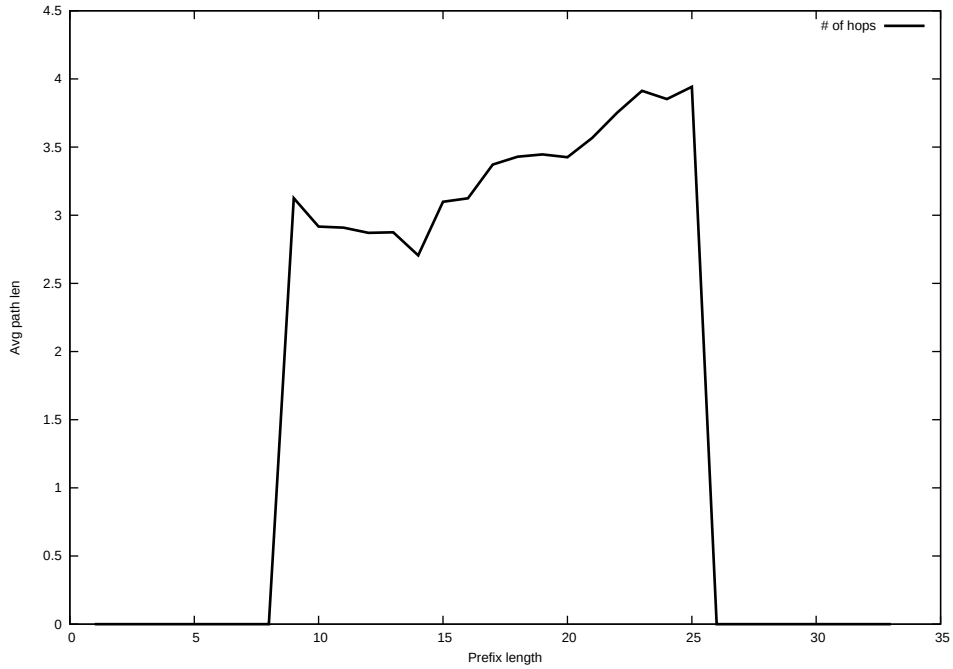
2015-03-29



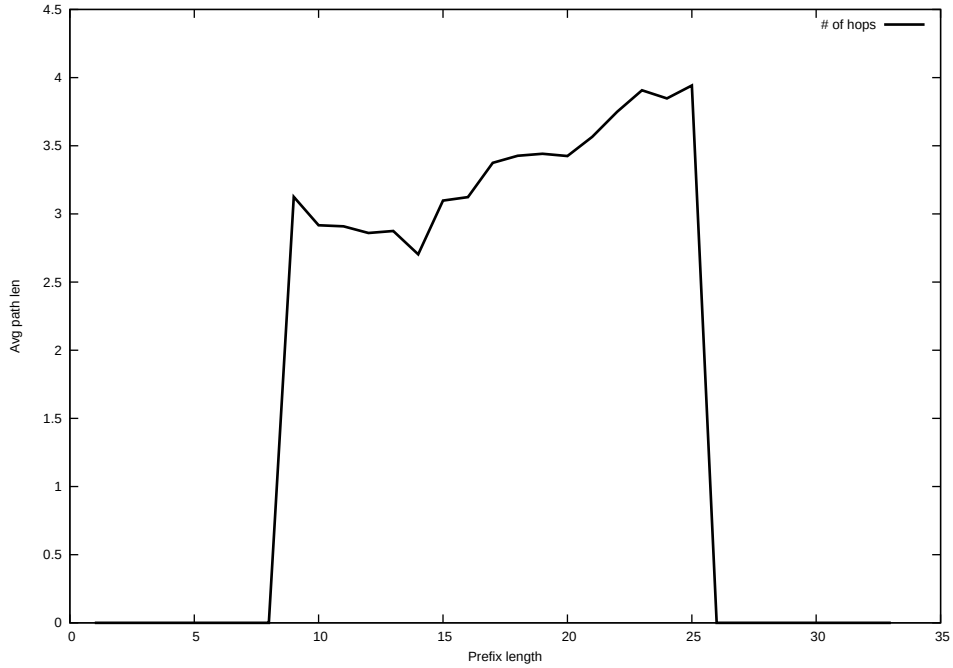
2015-03-30



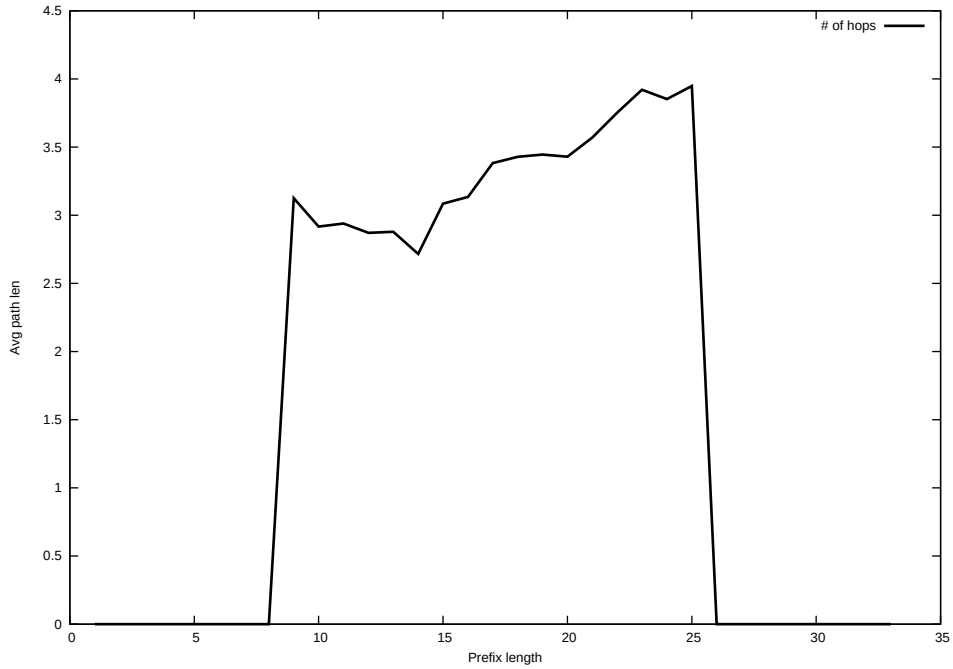
2015-03-31



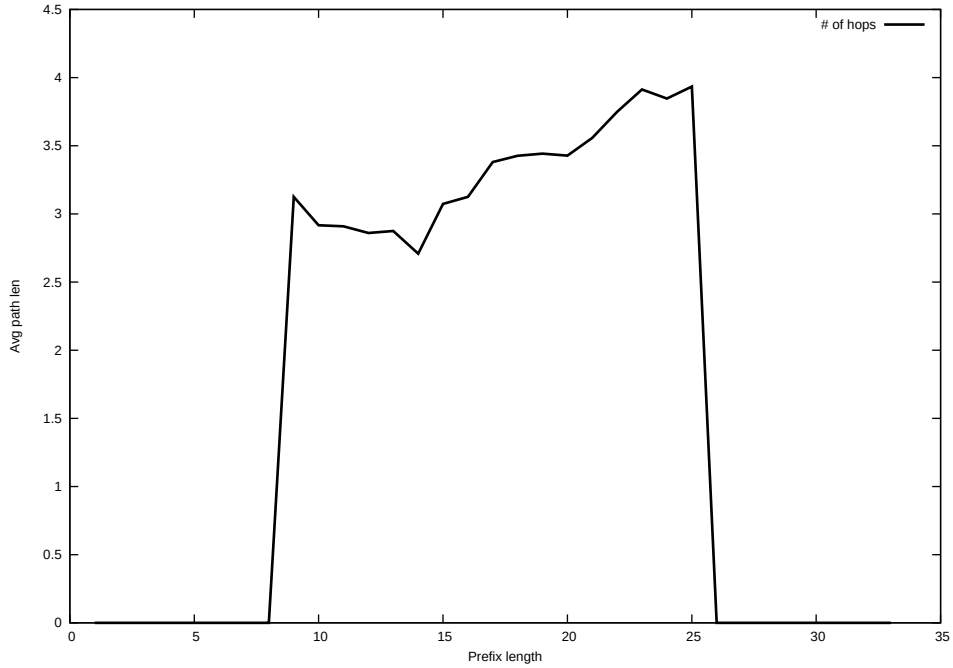
2015-04-01



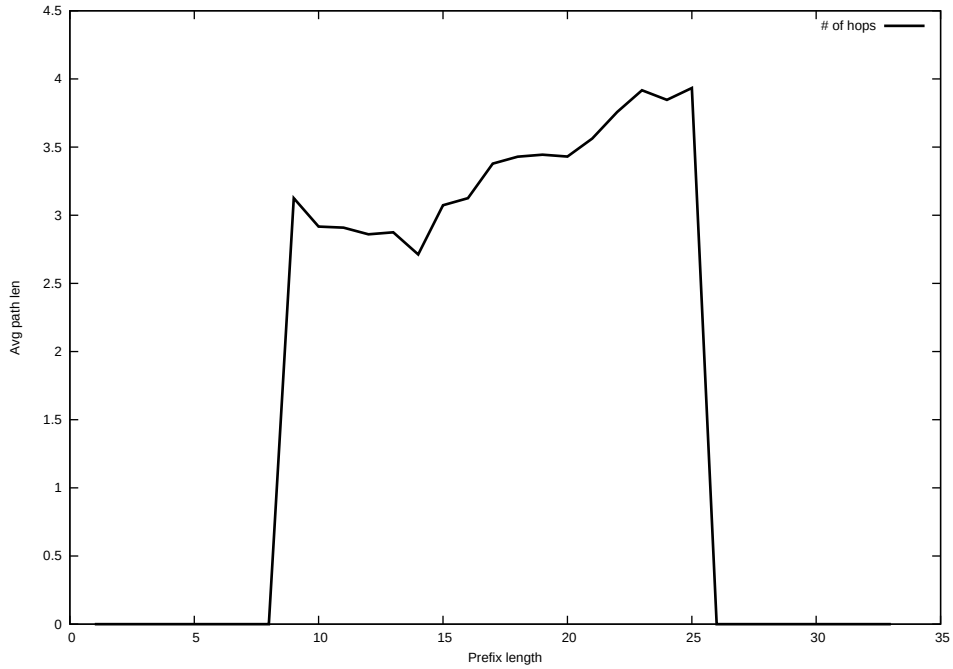
2015-04-02



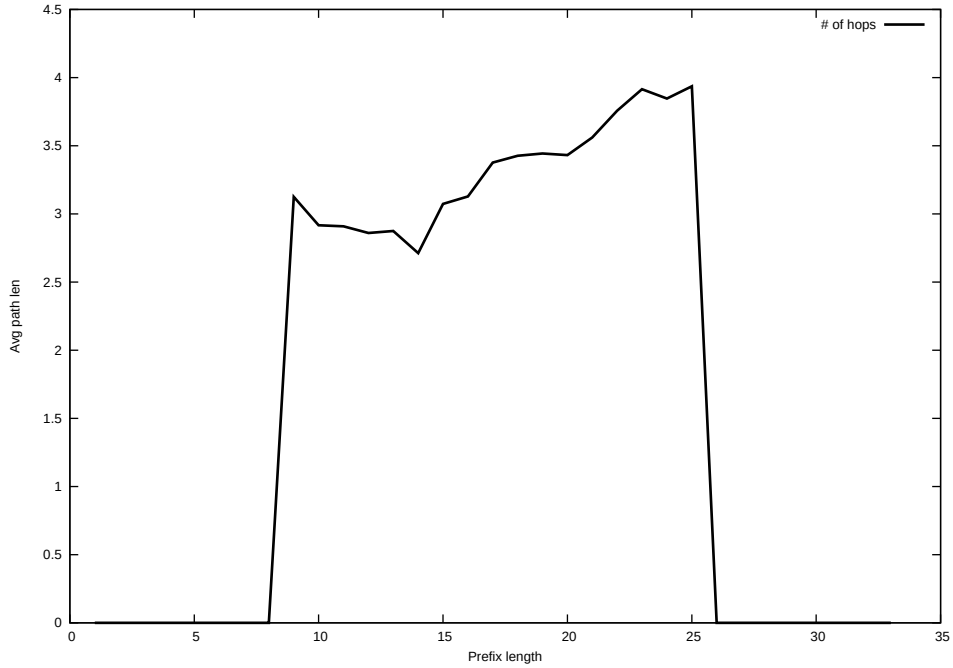
2015-04-03



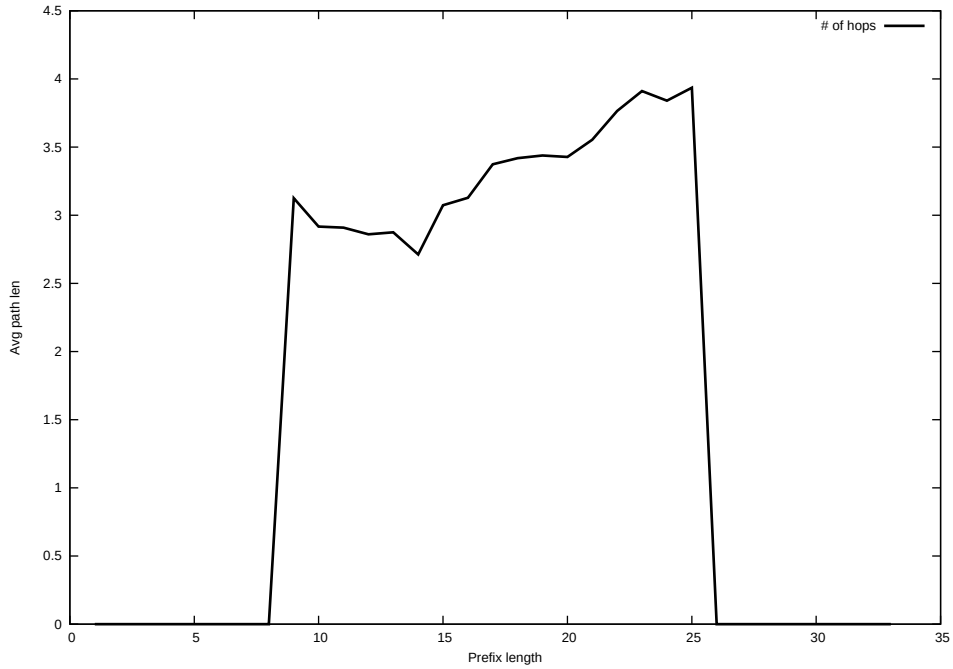
2015-04-04



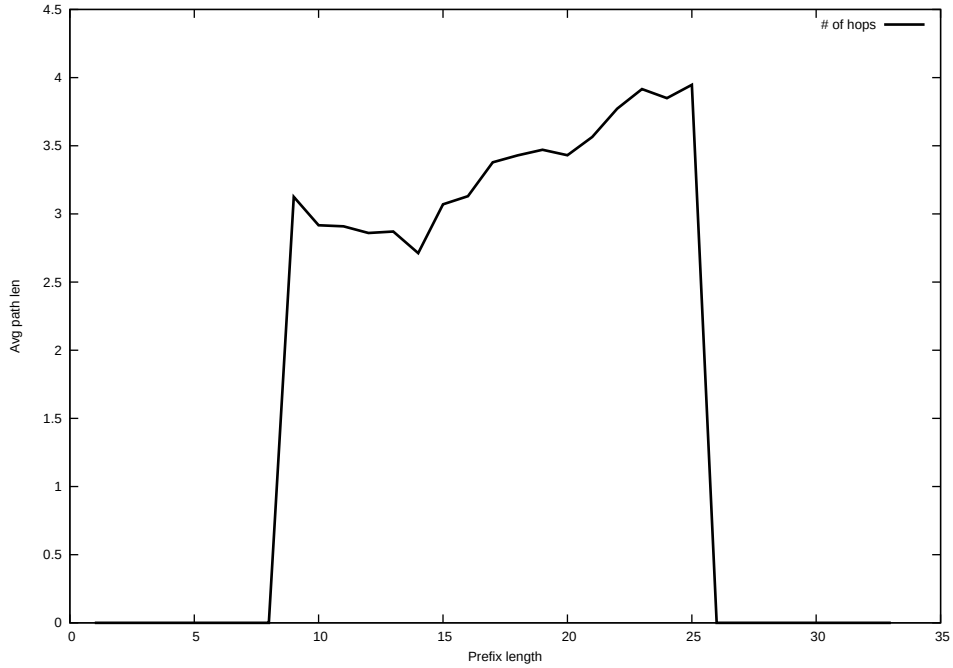
2015-04-05



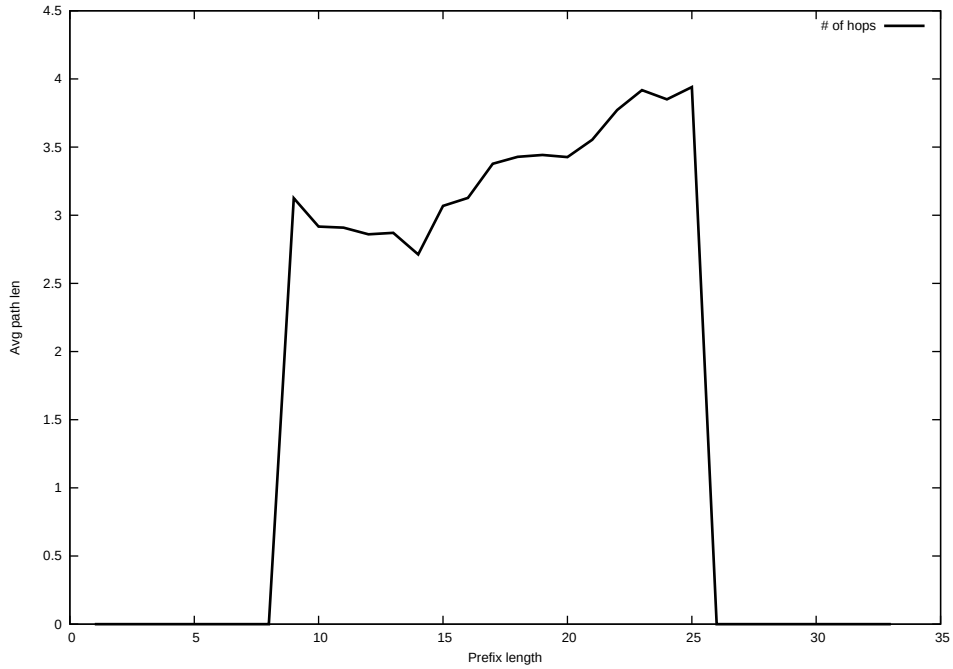
2015-04-06



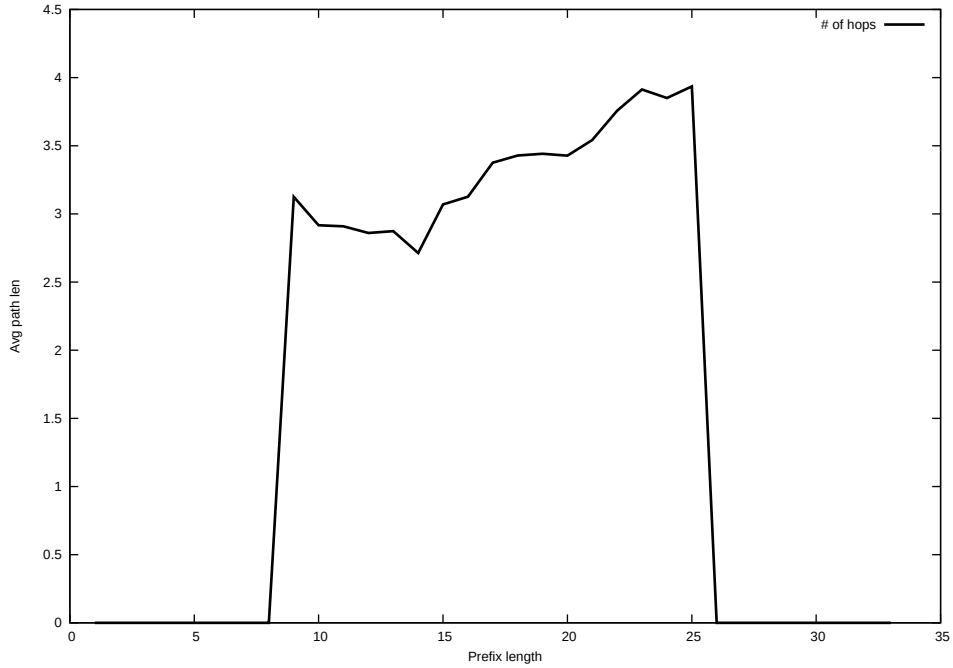
2015-04-07



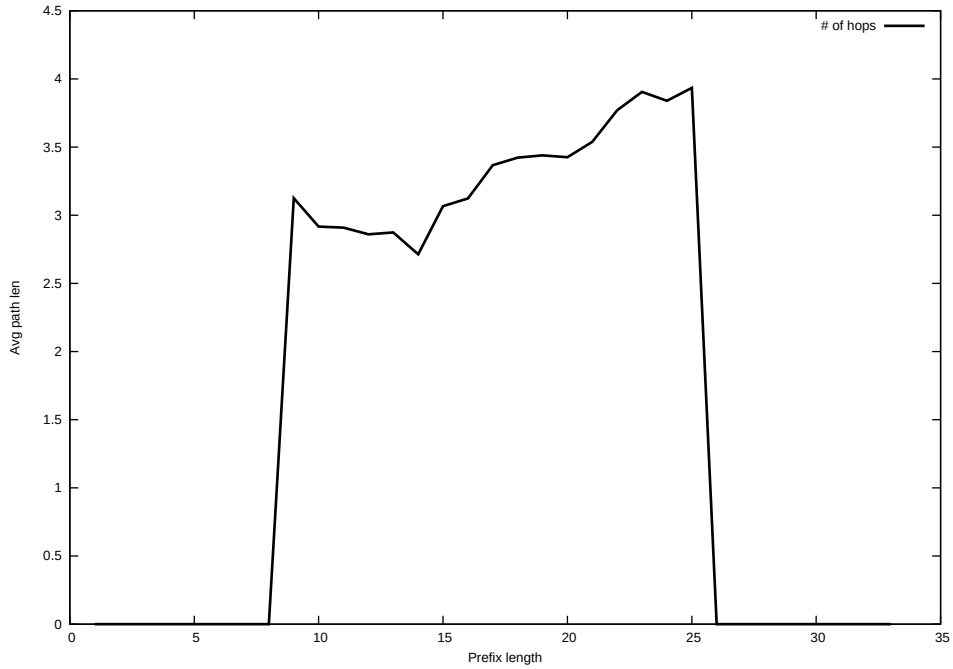
2015-04-08



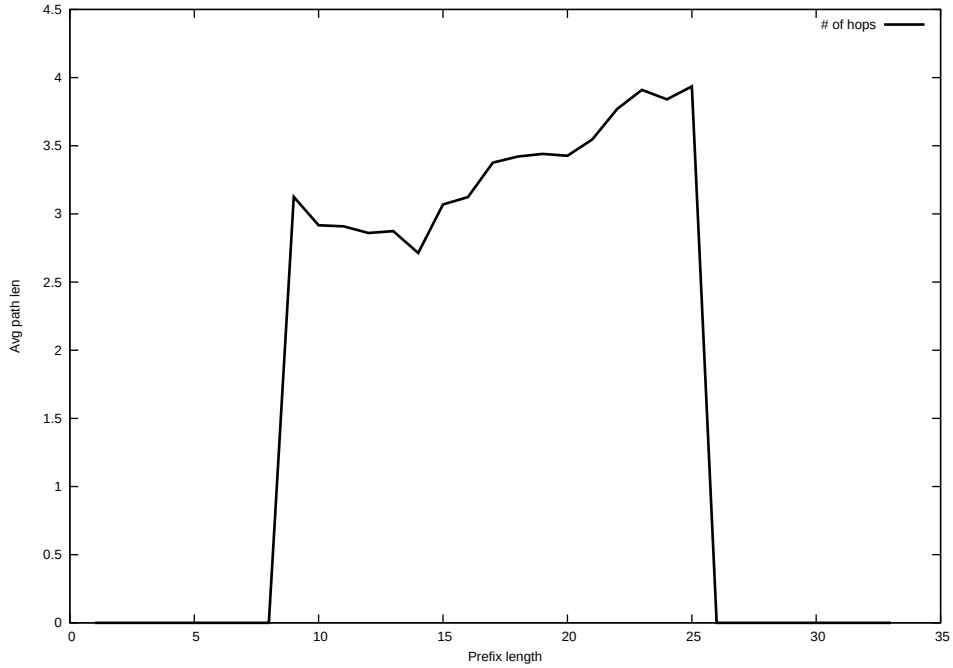
2015-04-09



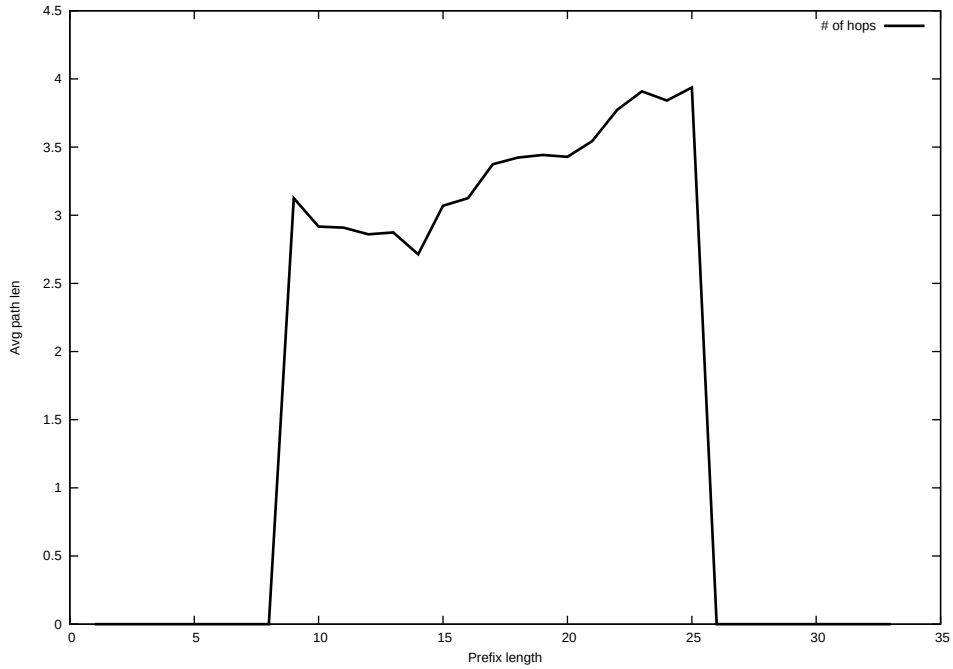
2015-04-10



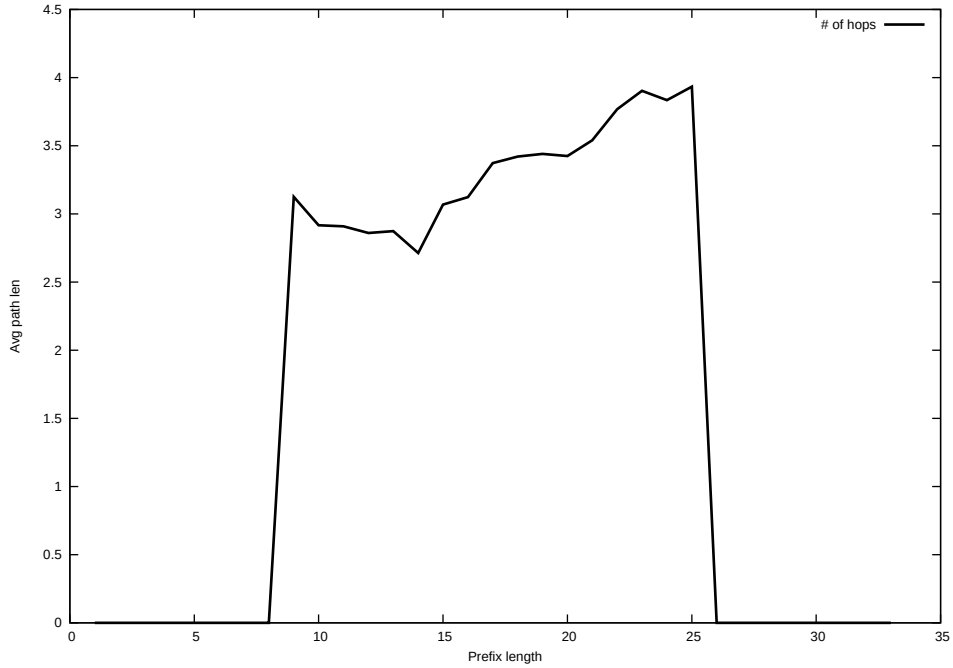
2015-04-11



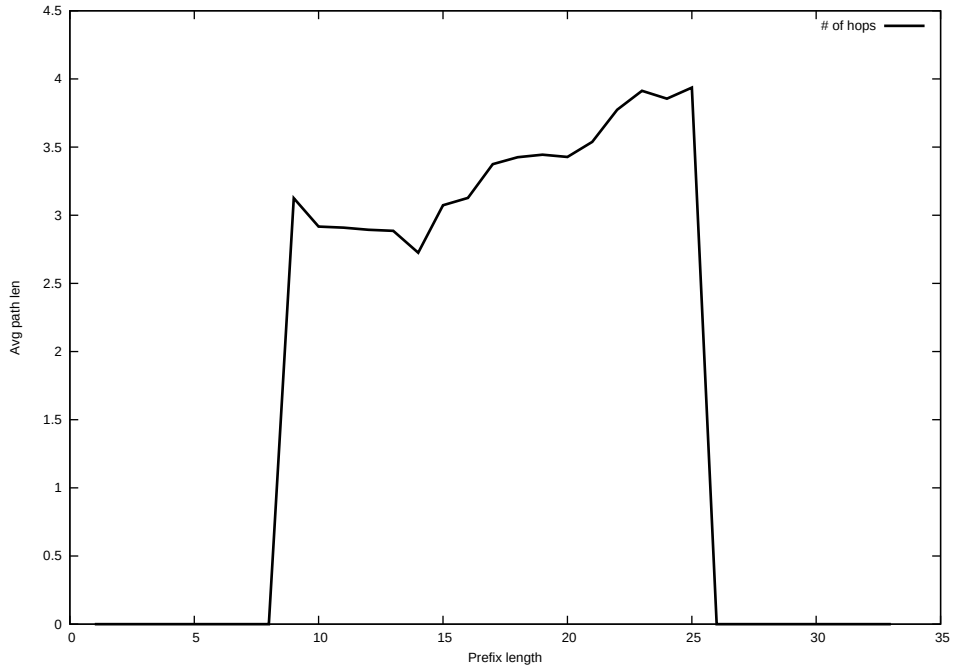
2015-04-12



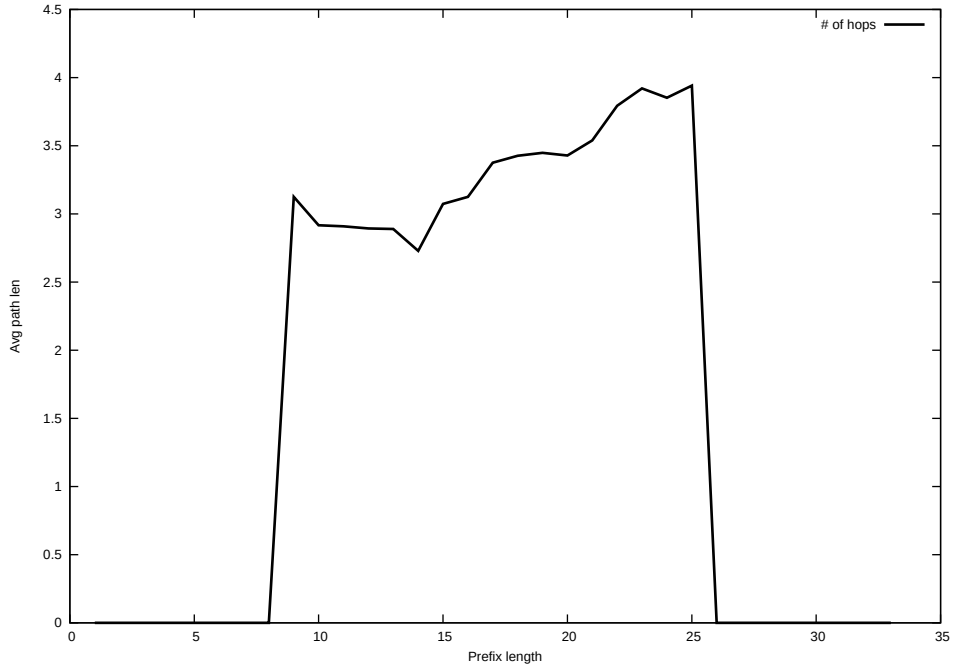
2015-04-13



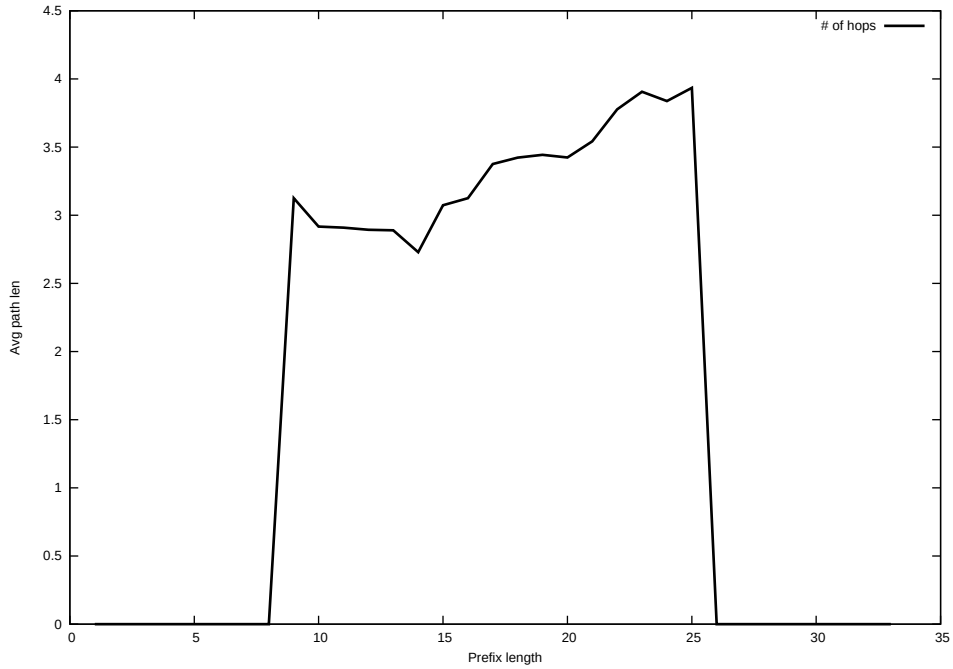
2015-04-14



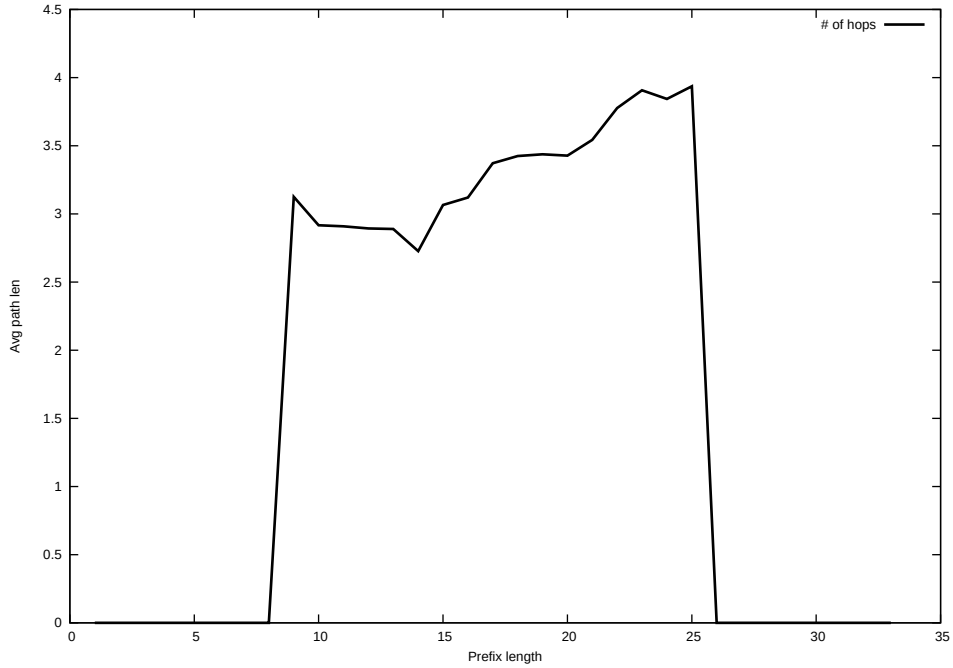
2015-04-15



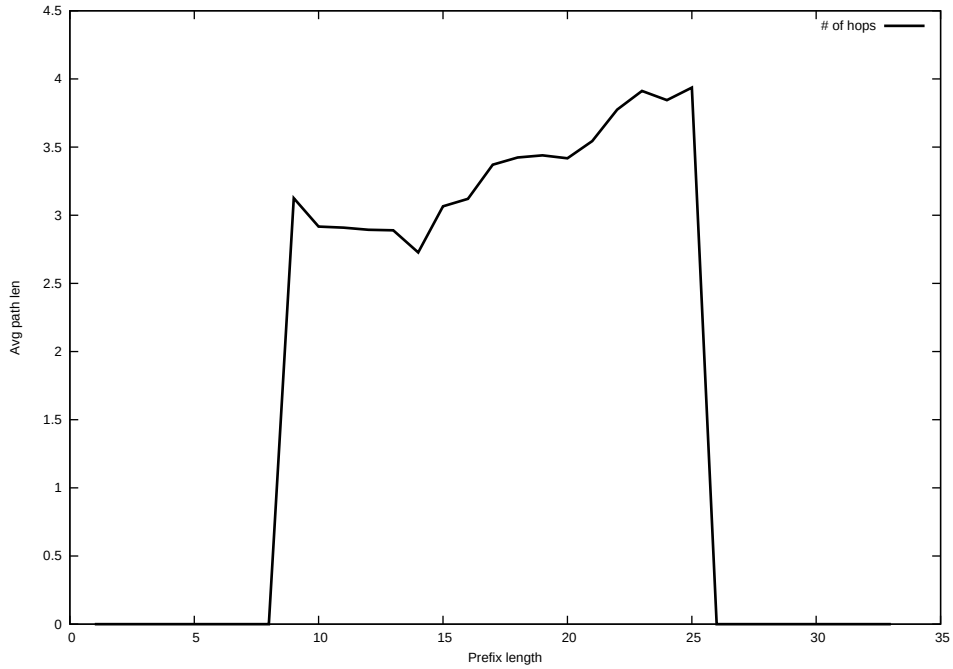
2015-04-16



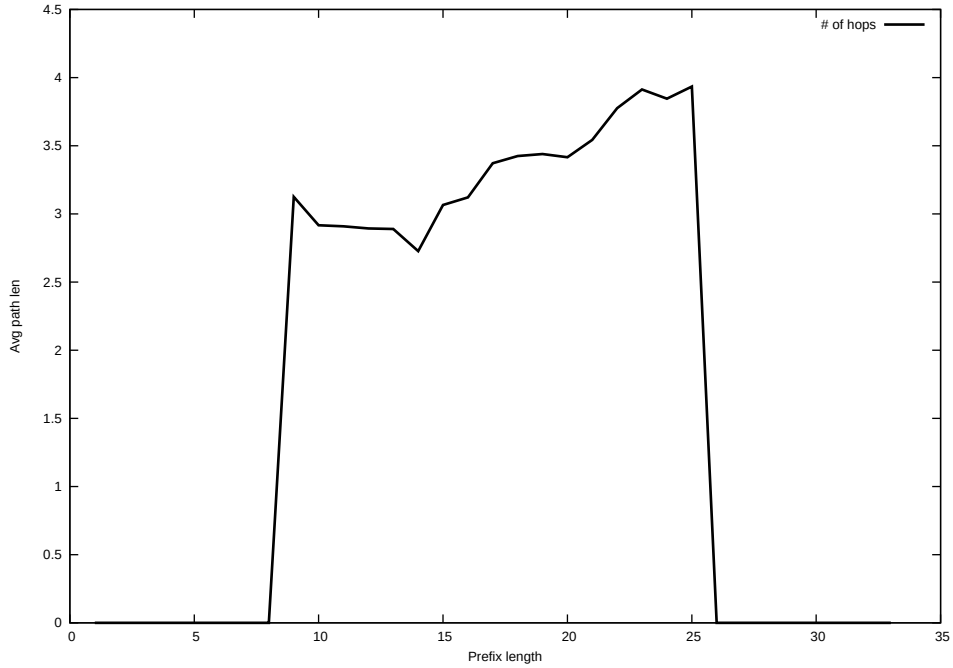
2015-04-17



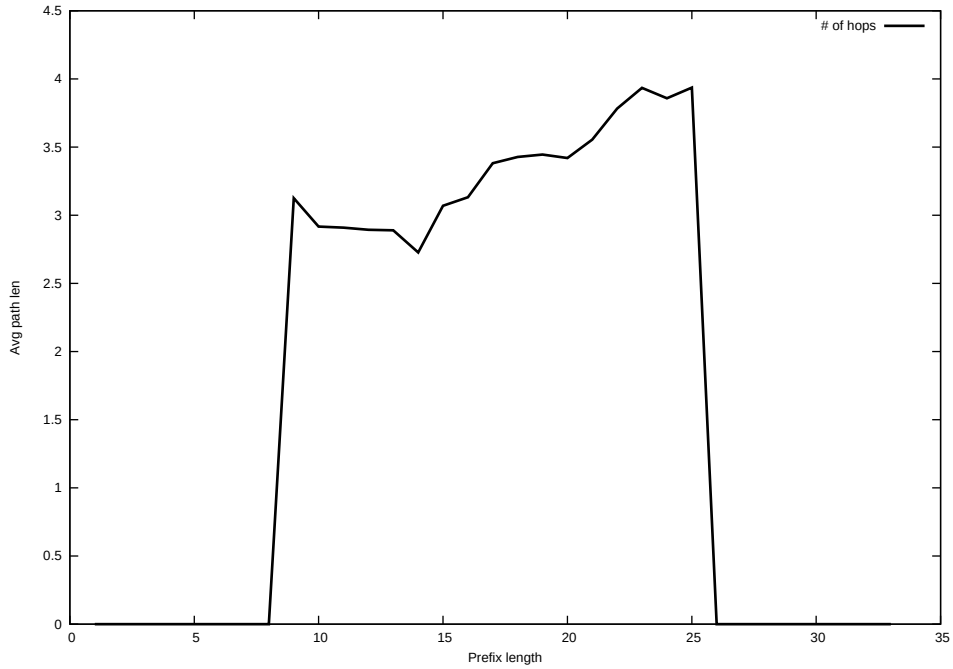
2015-04-18



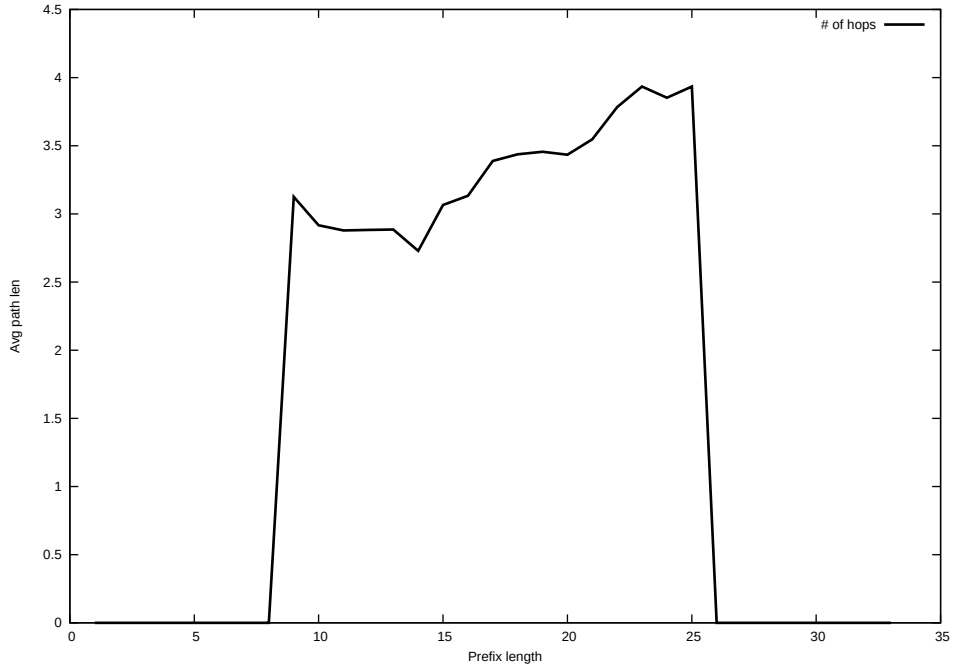
2015-04-19



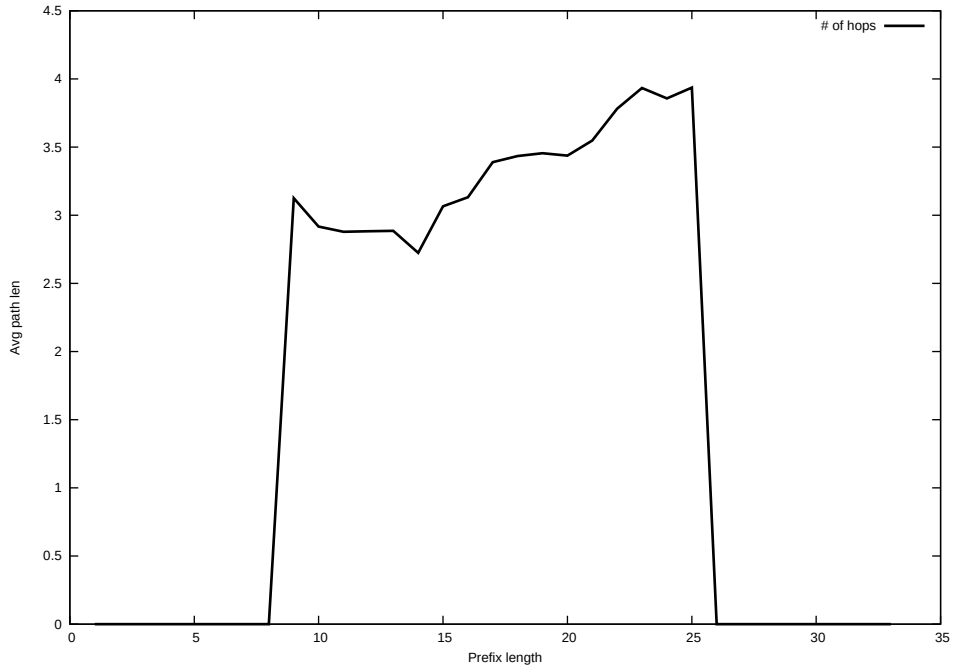
2015-04-20



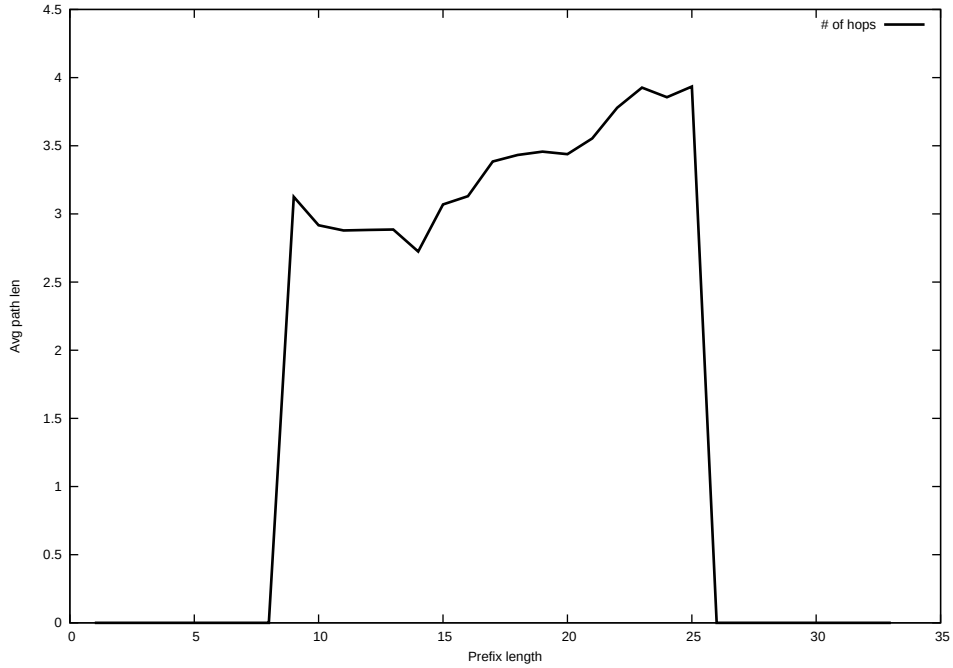
2015-04-21



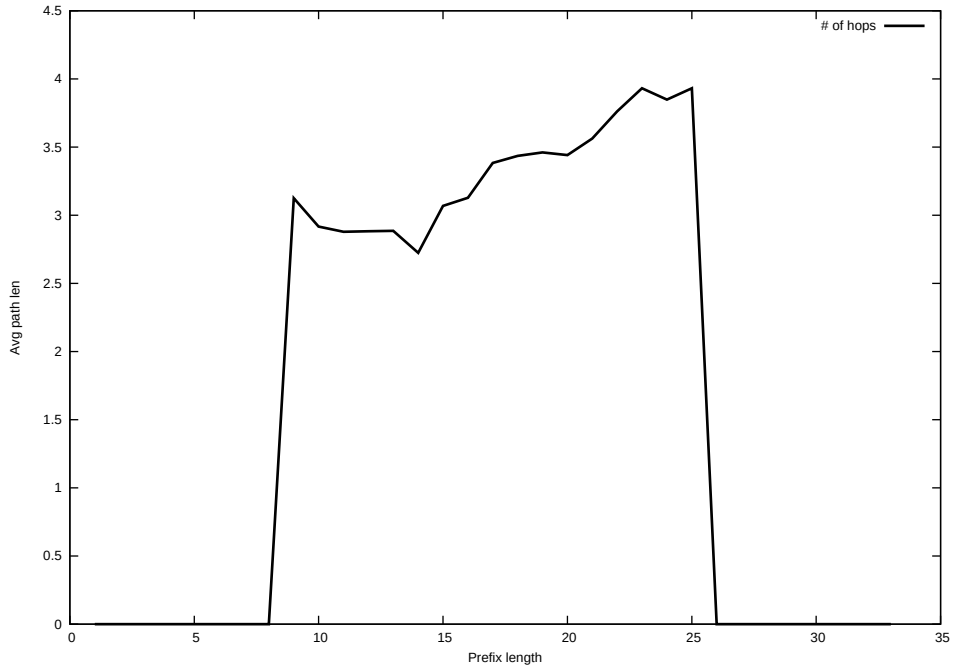
2015-04-22



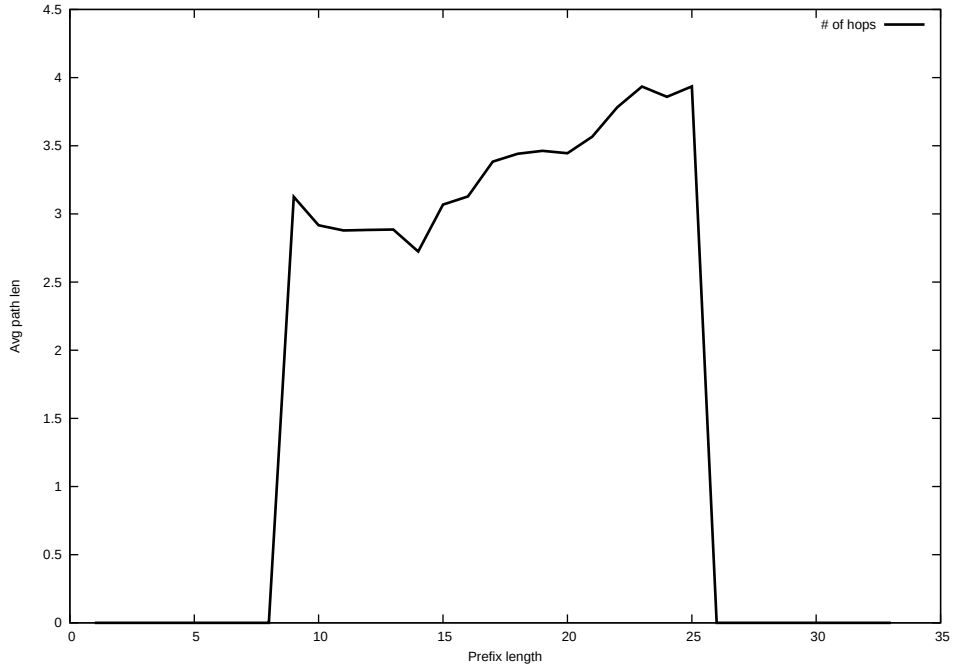
2015-04-23



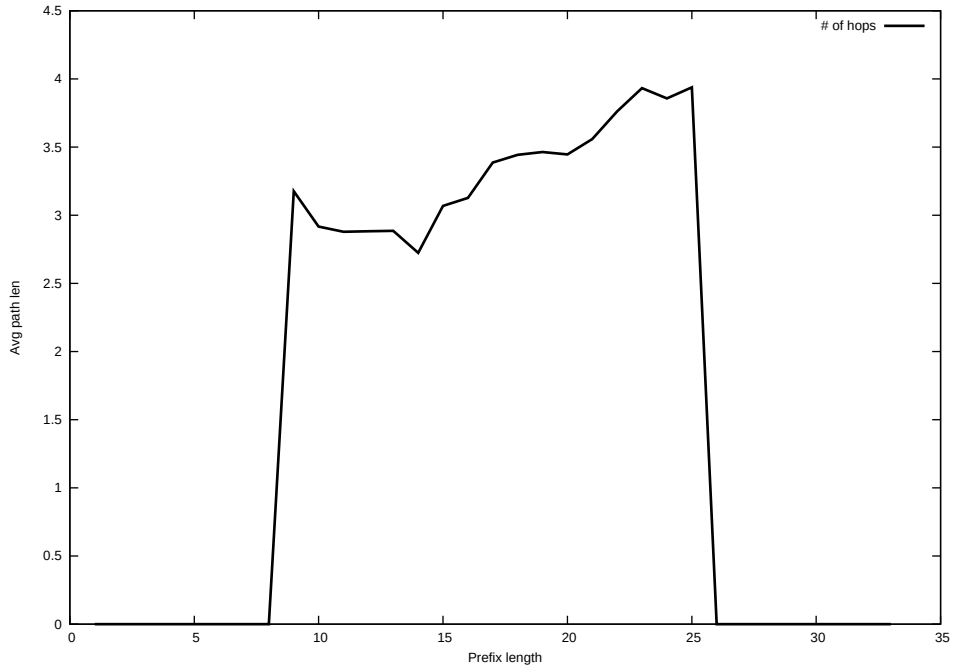
2015-04-24



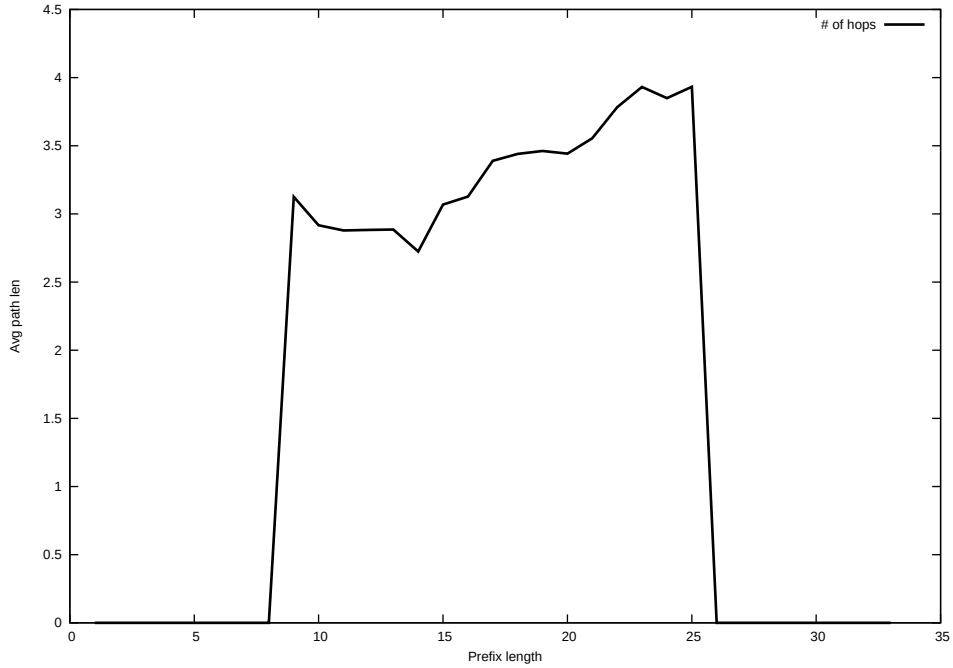
2015-04-25



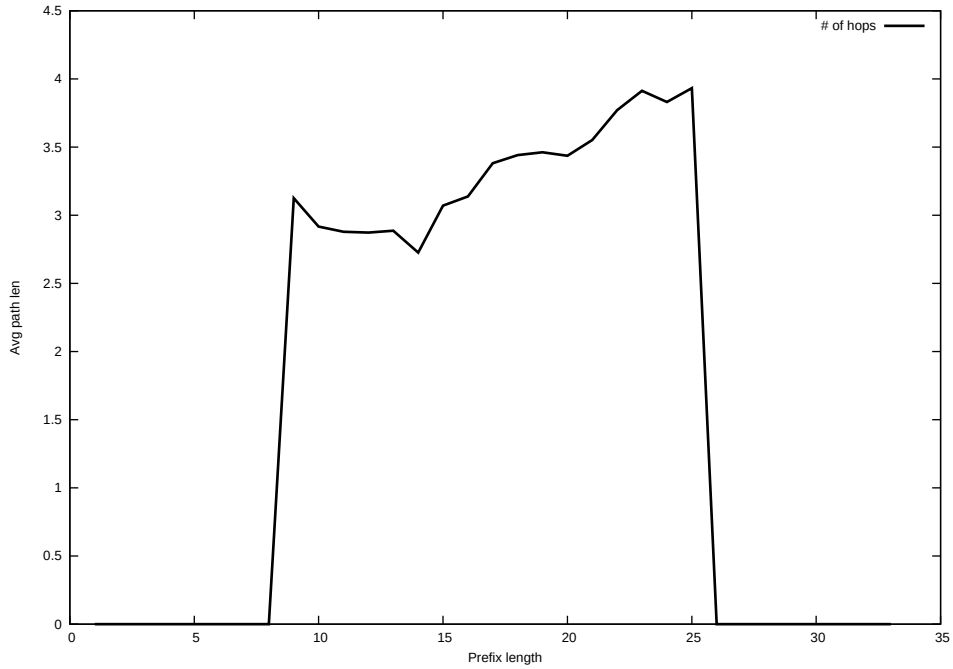
2015-04-26



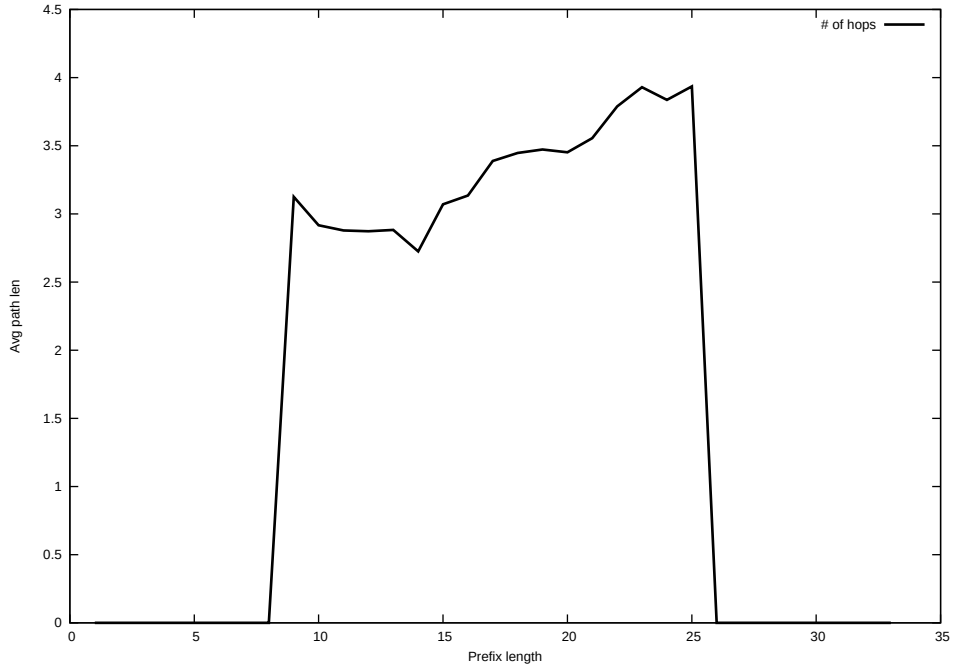
2015-04-27



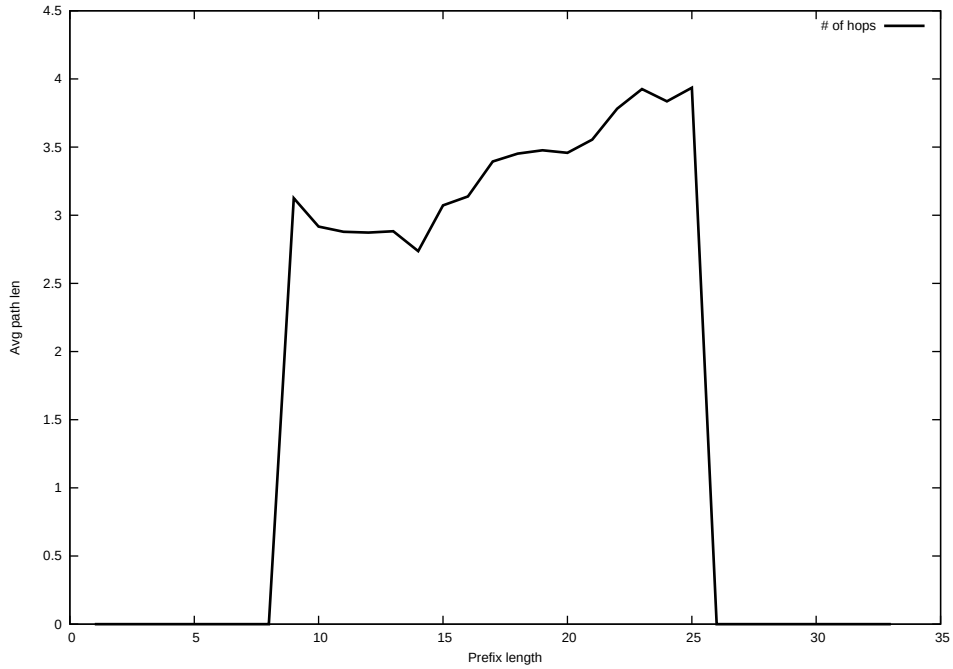
2015-04-28



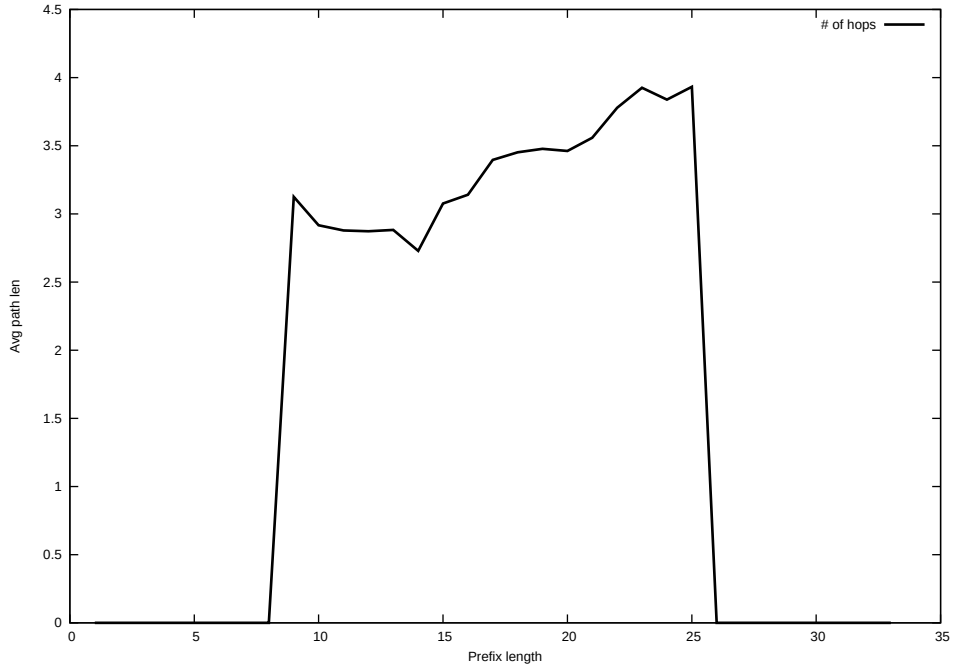
2015-04-29



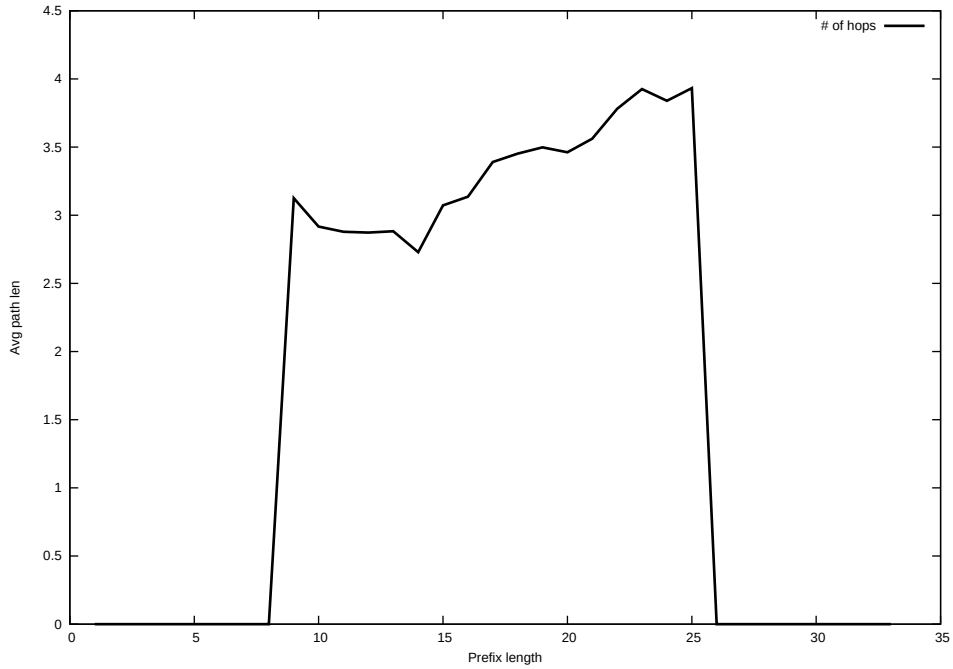
2015-04-30



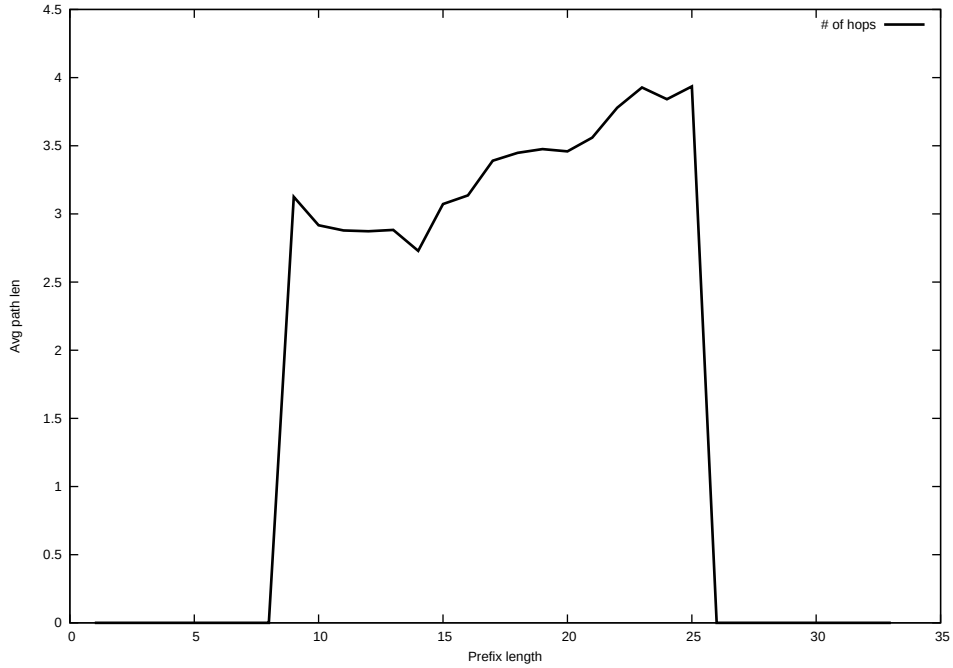
2015-05-01



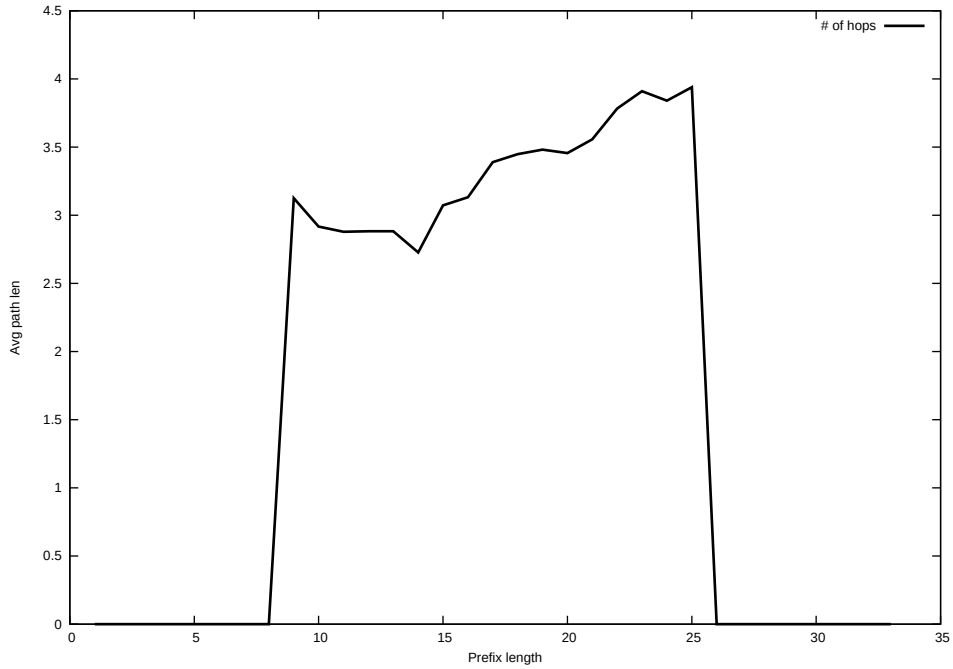
2015-05-02



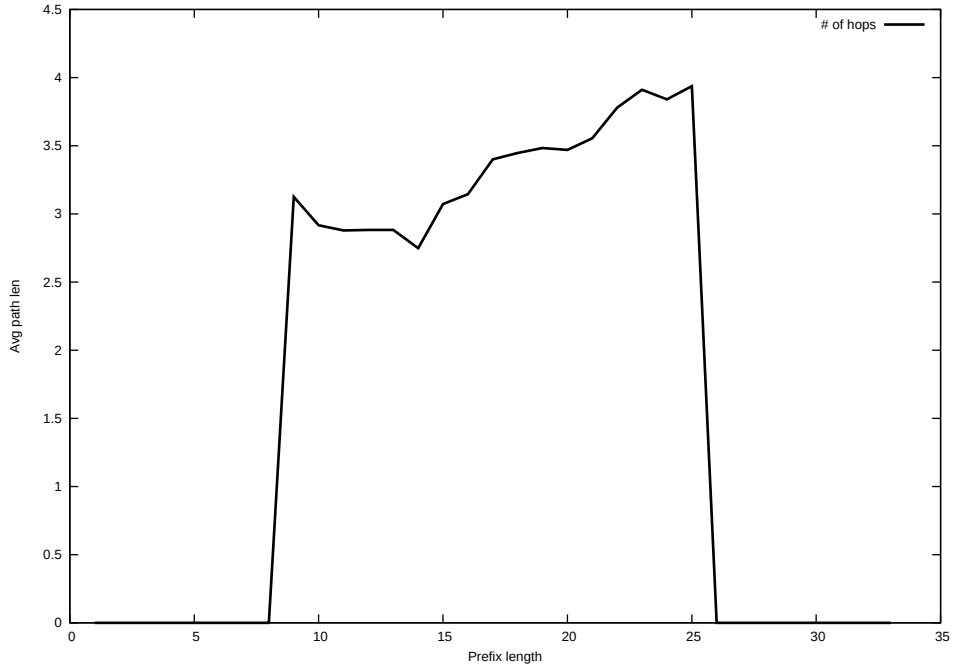
2015-05-03



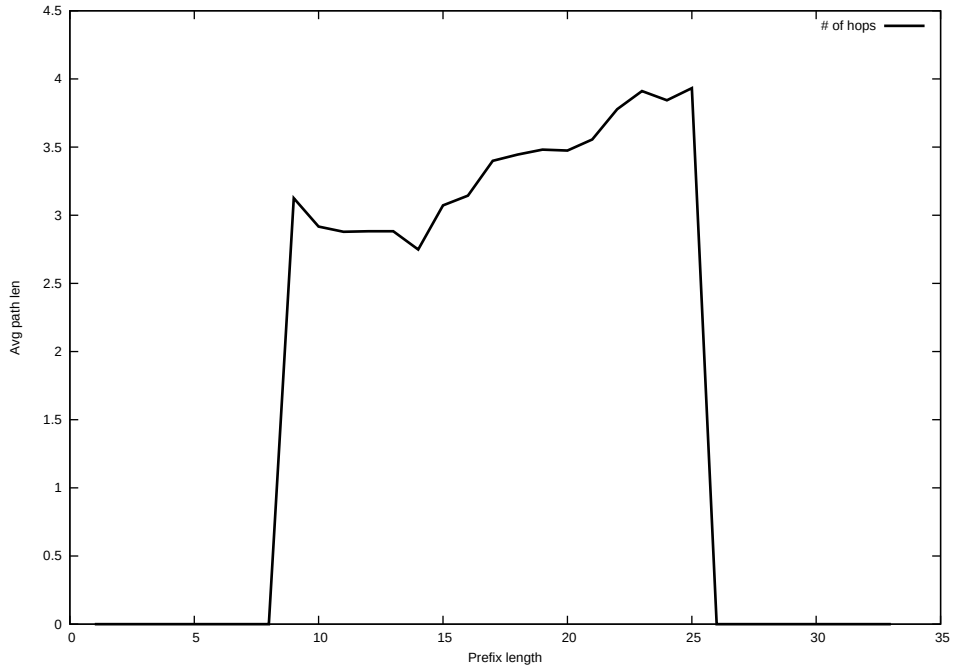
2015-05-04



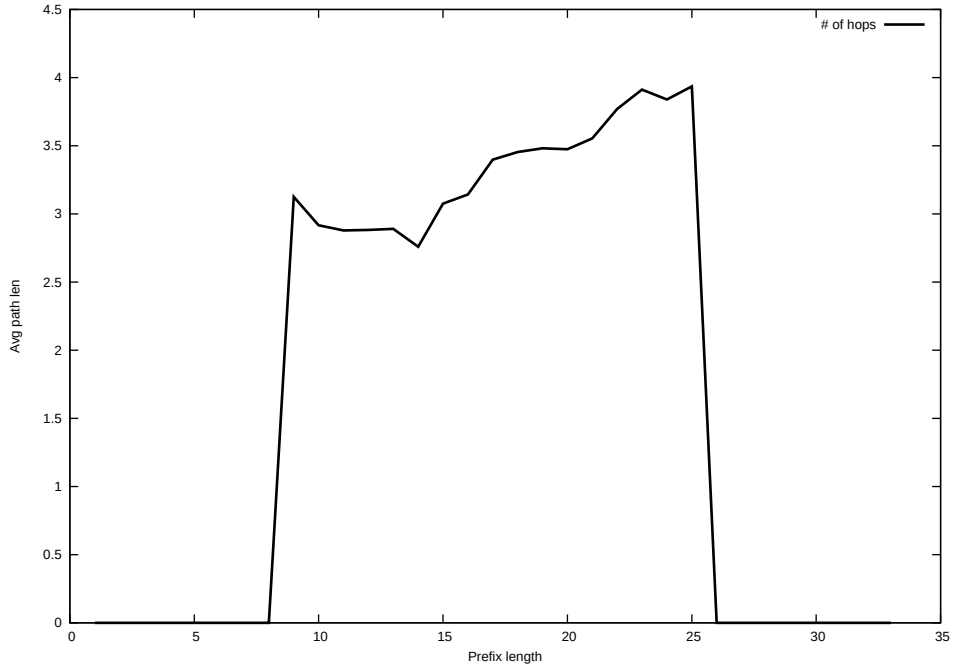
2015-05-05



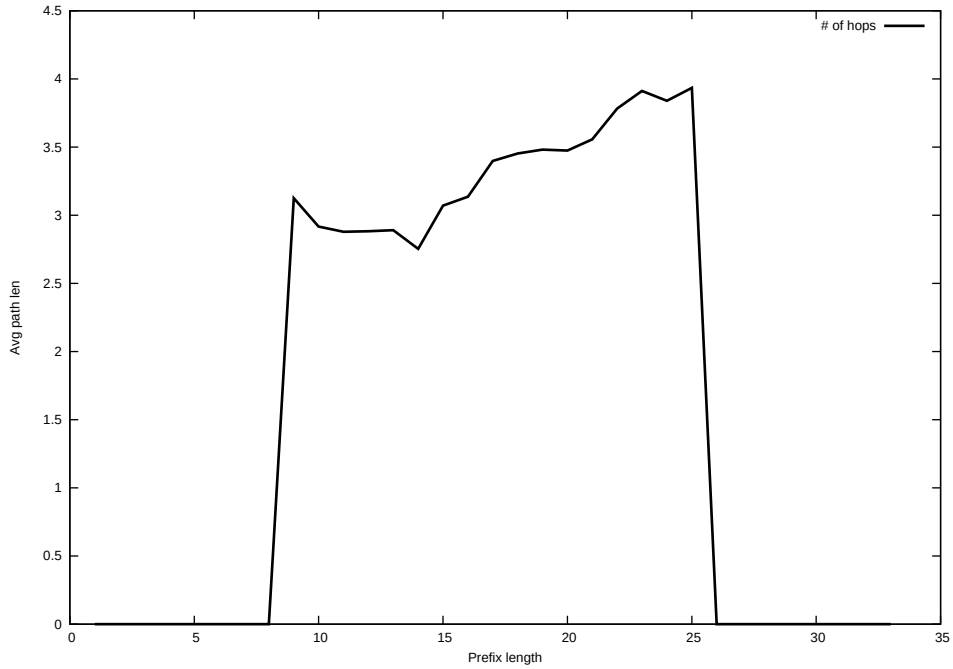
2015-05-06



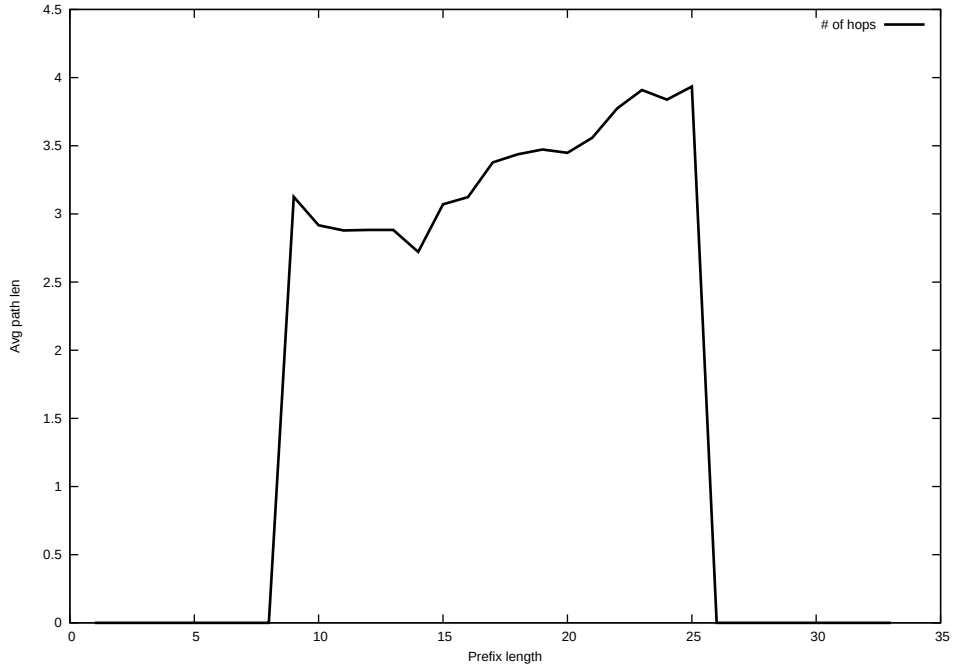
2015-05-07



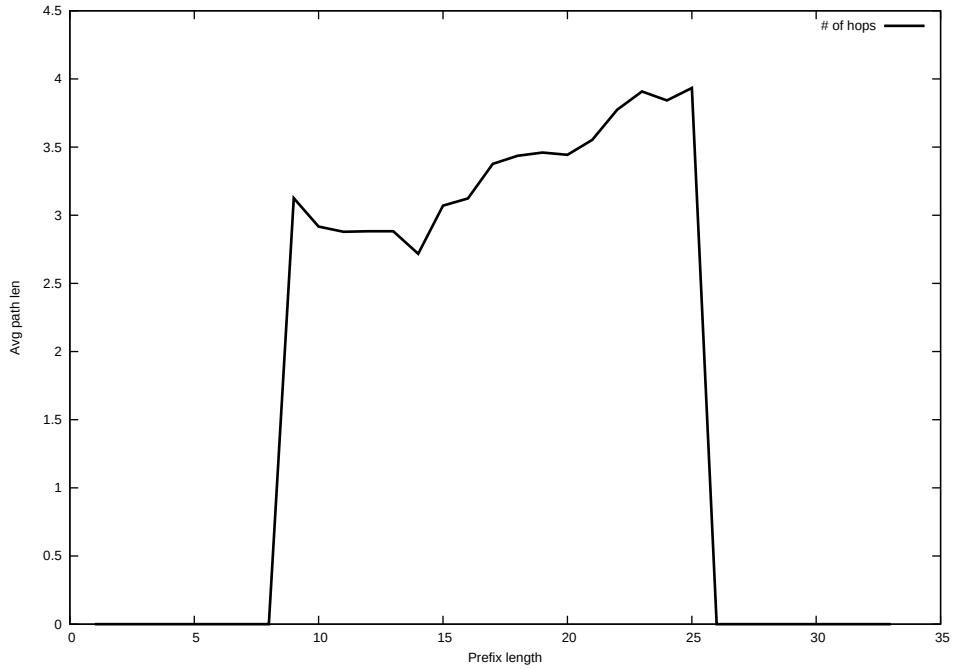
2015-05-08



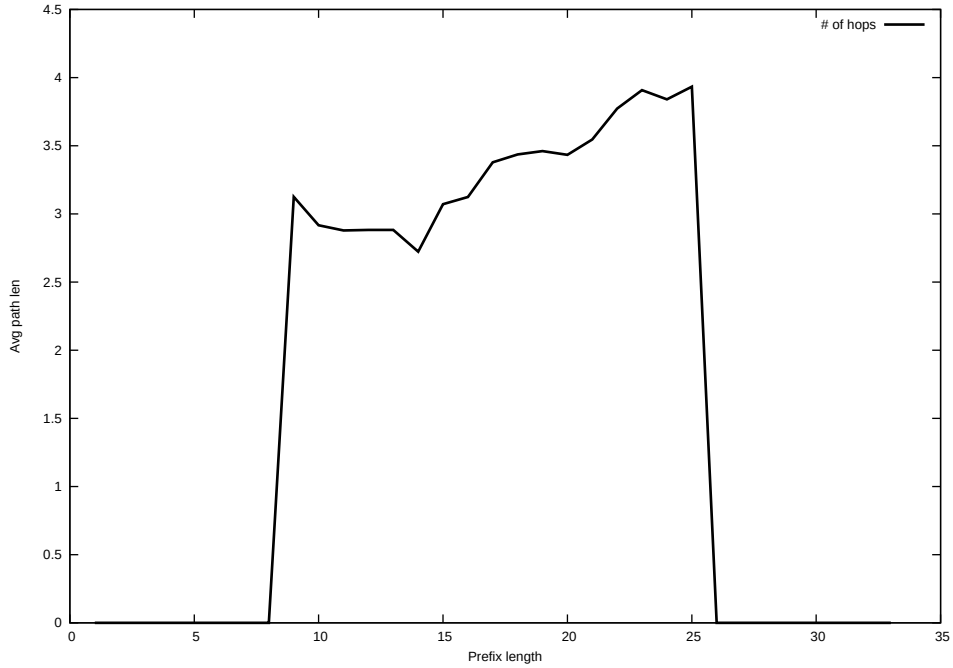
2015-05-09



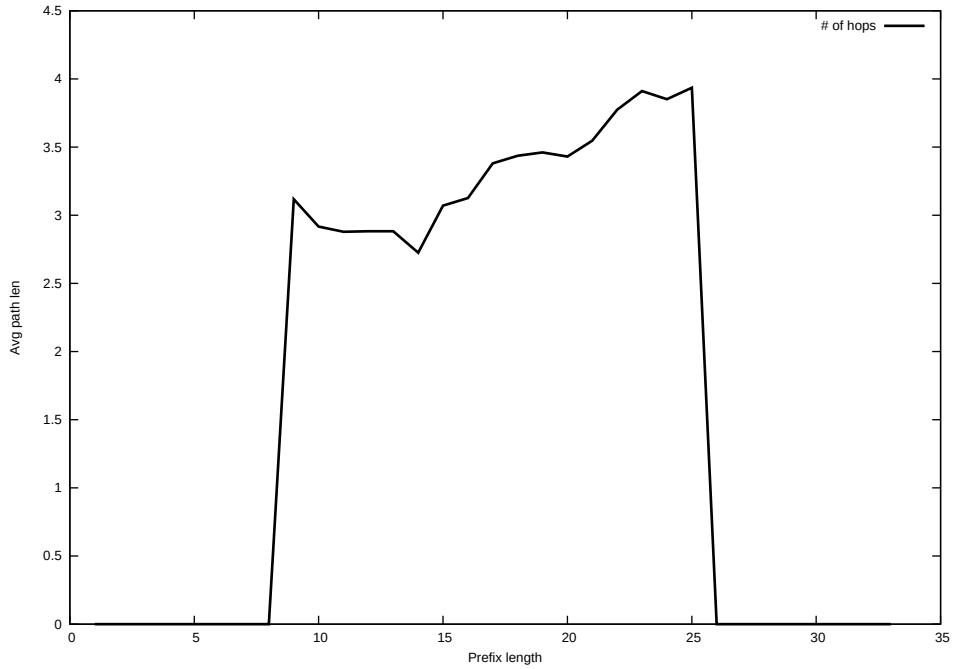
2015-05-10



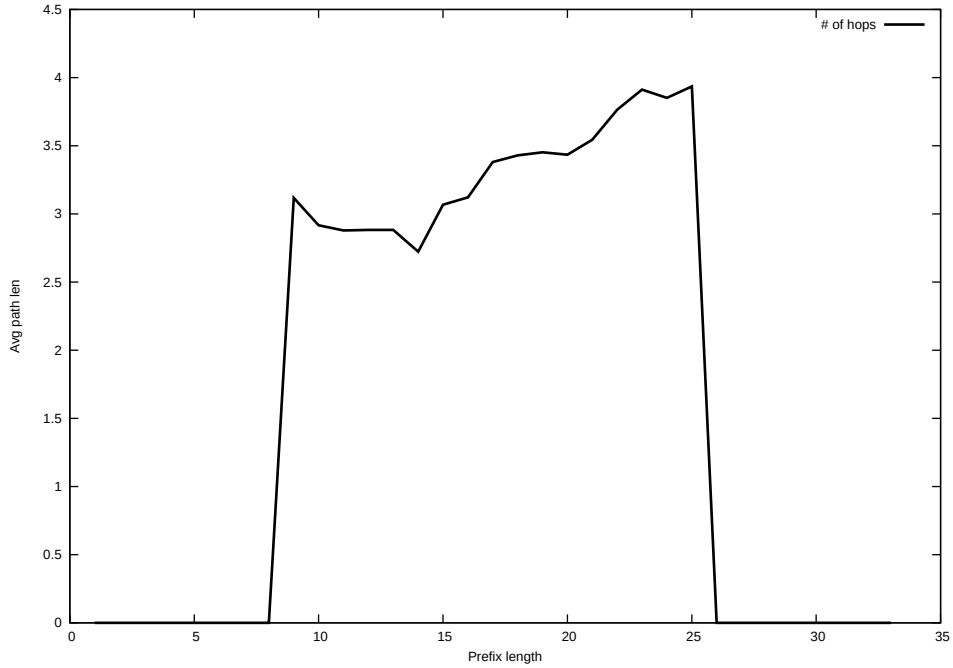
2015-05-11



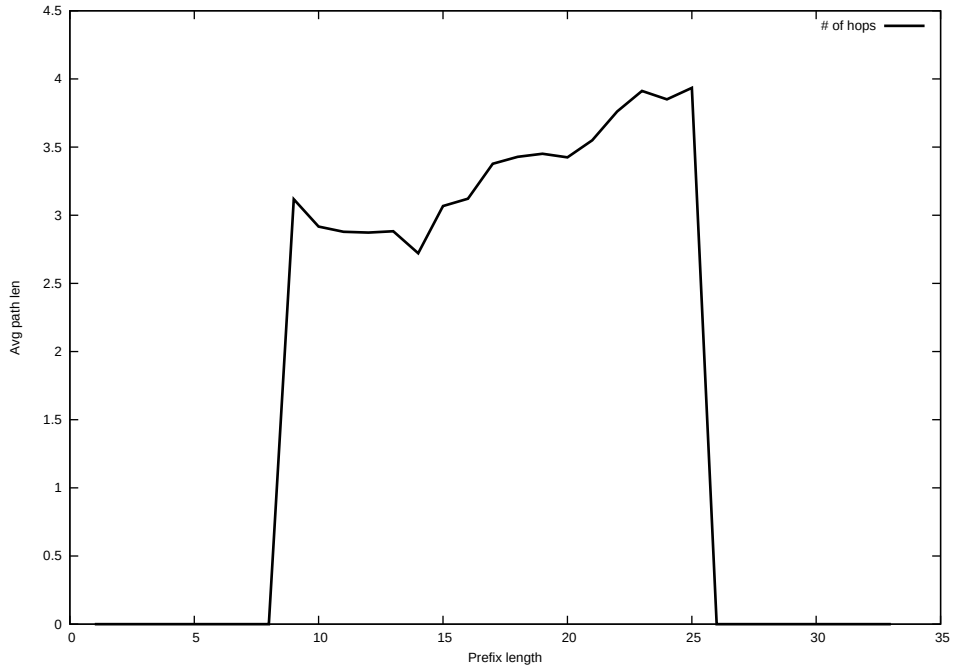
2015-05-12



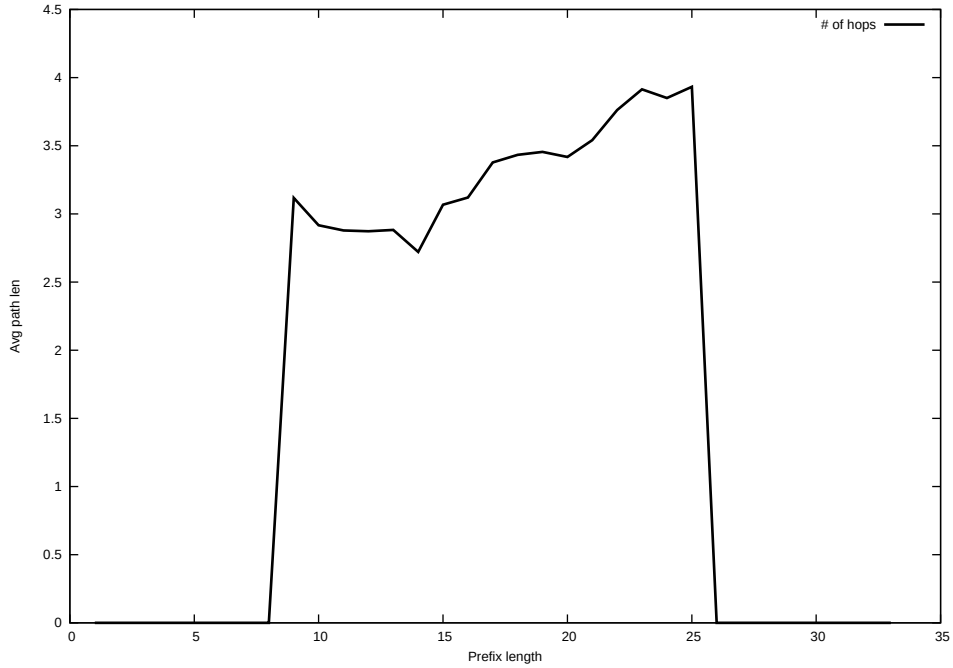
2015-05-13



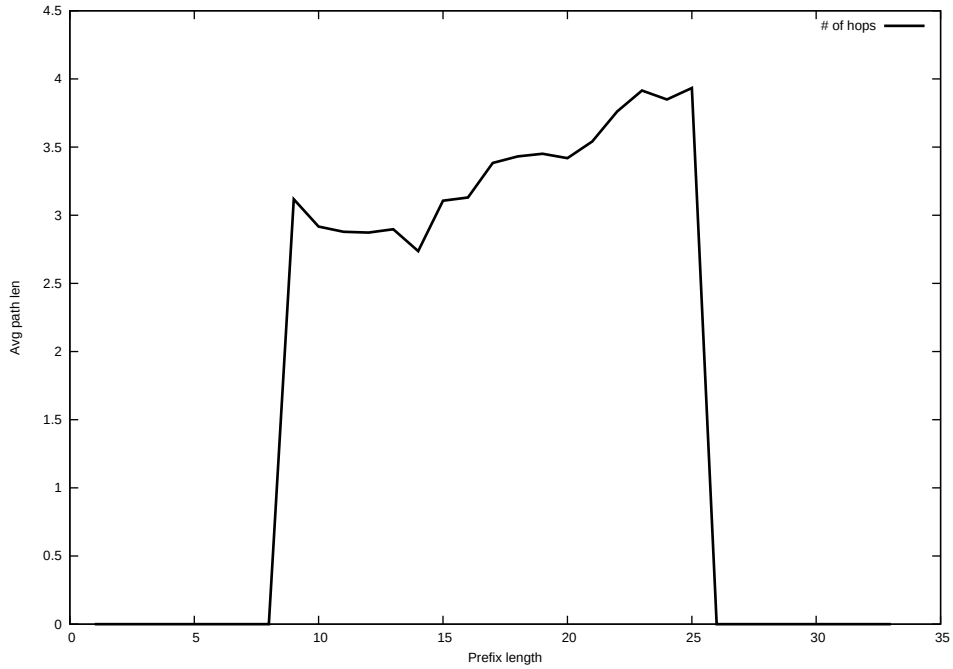
2015-05-14



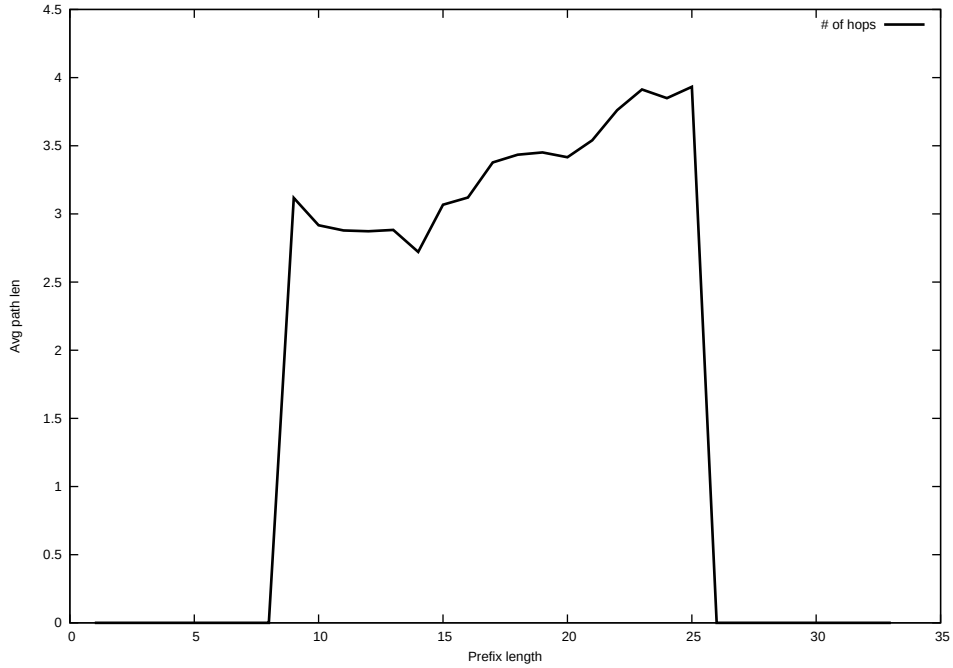
2015-05-15



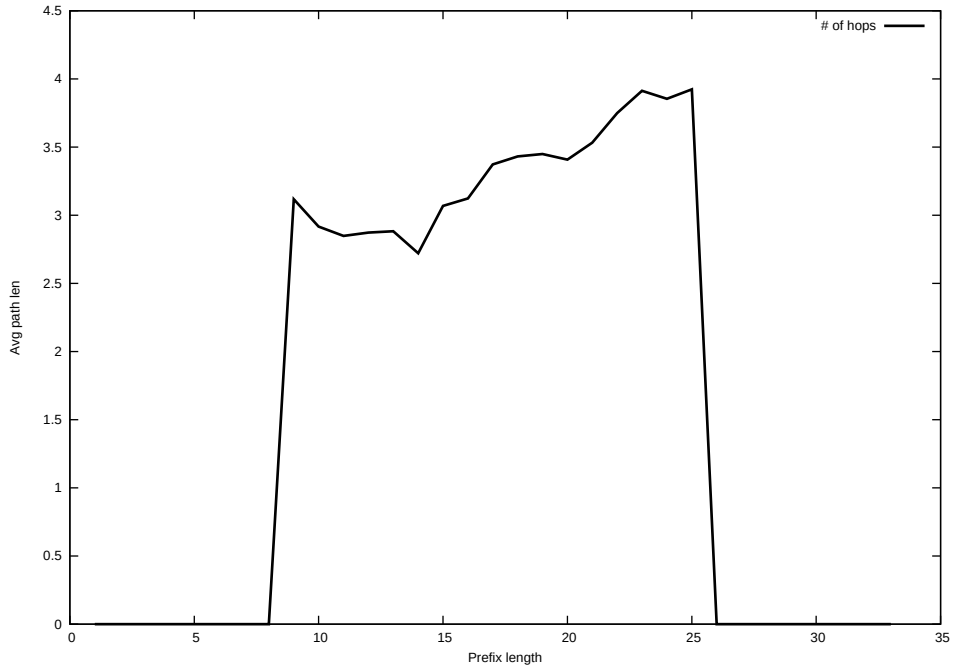
2015-05-16



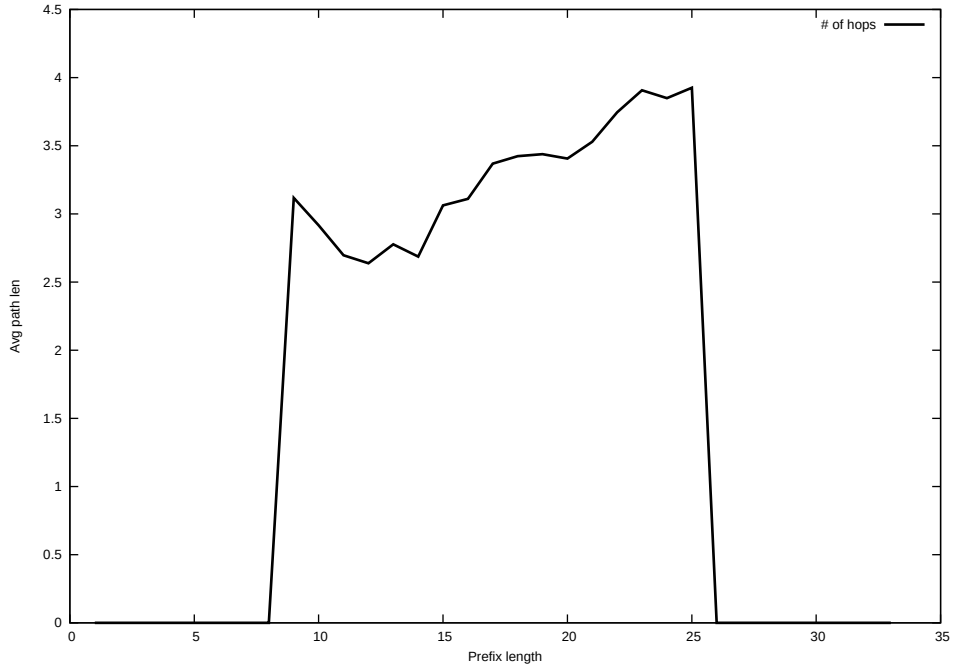
2015-05-17



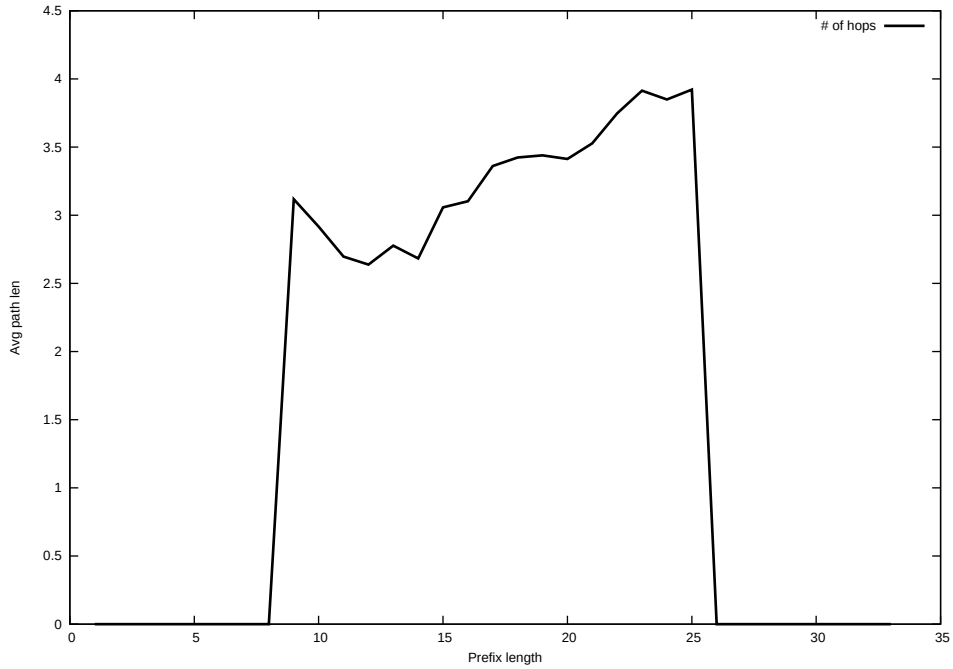
2015-05-18



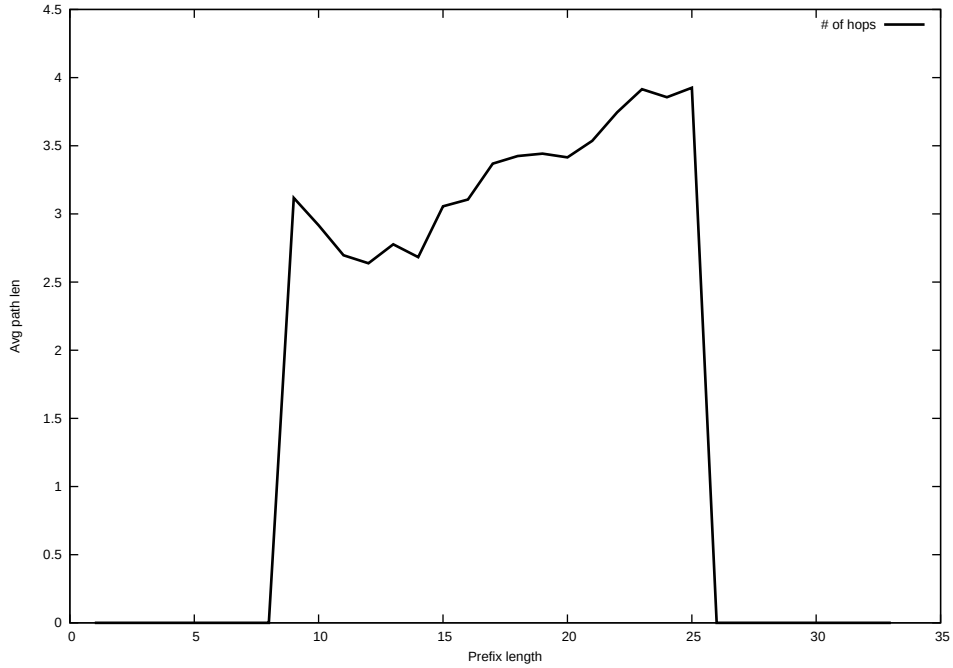
2015-05-19



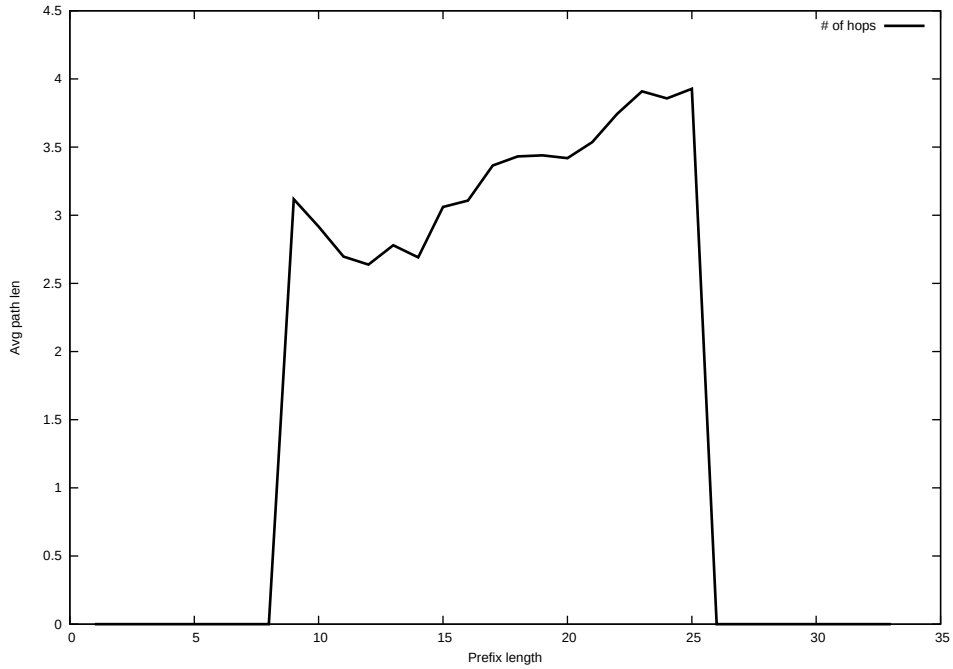
2015-05-20



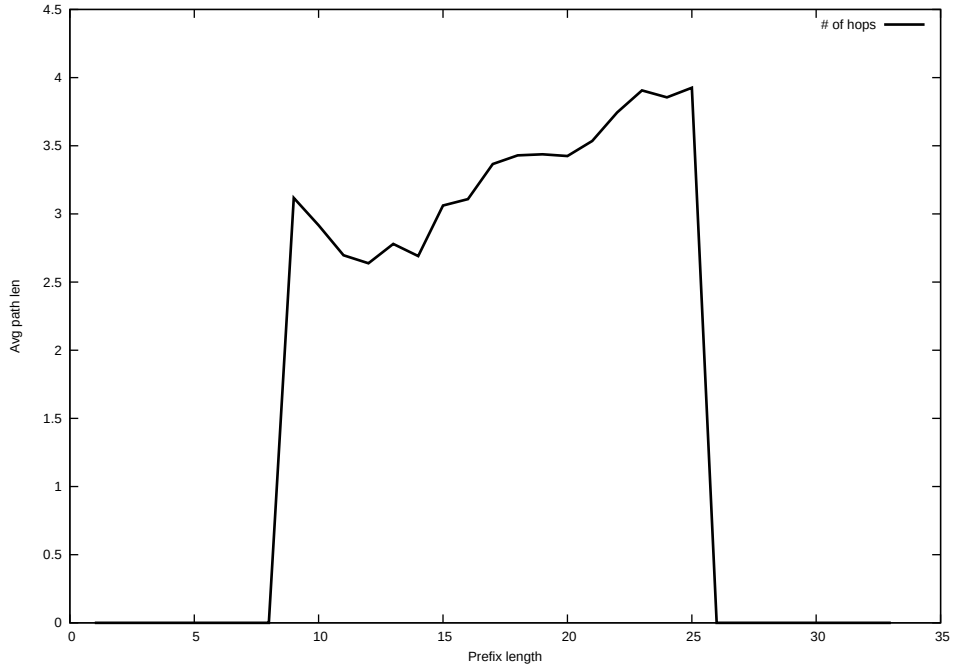
2015-05-21



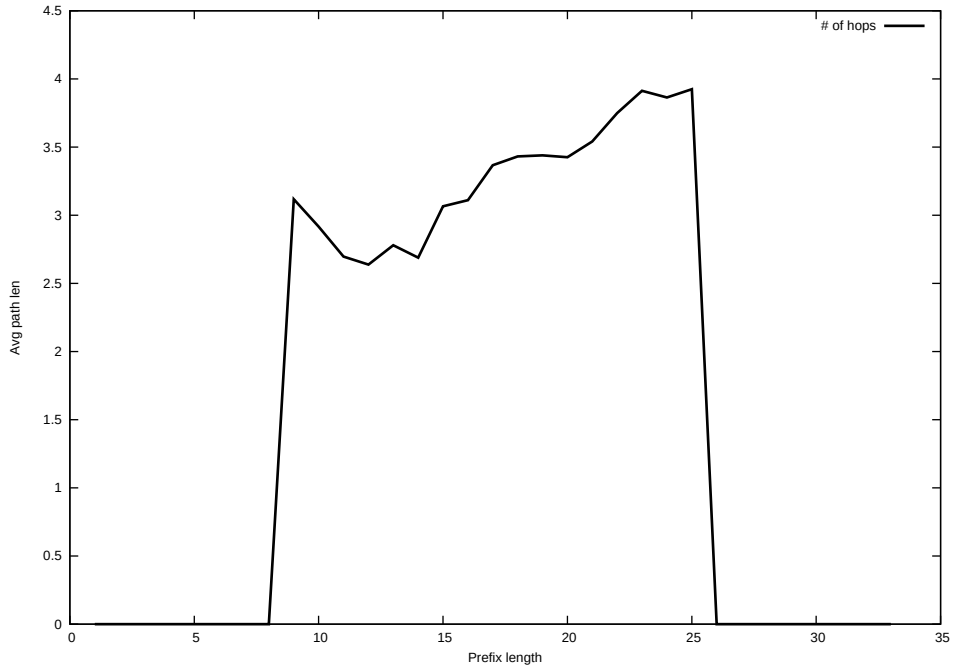
2015-05-22



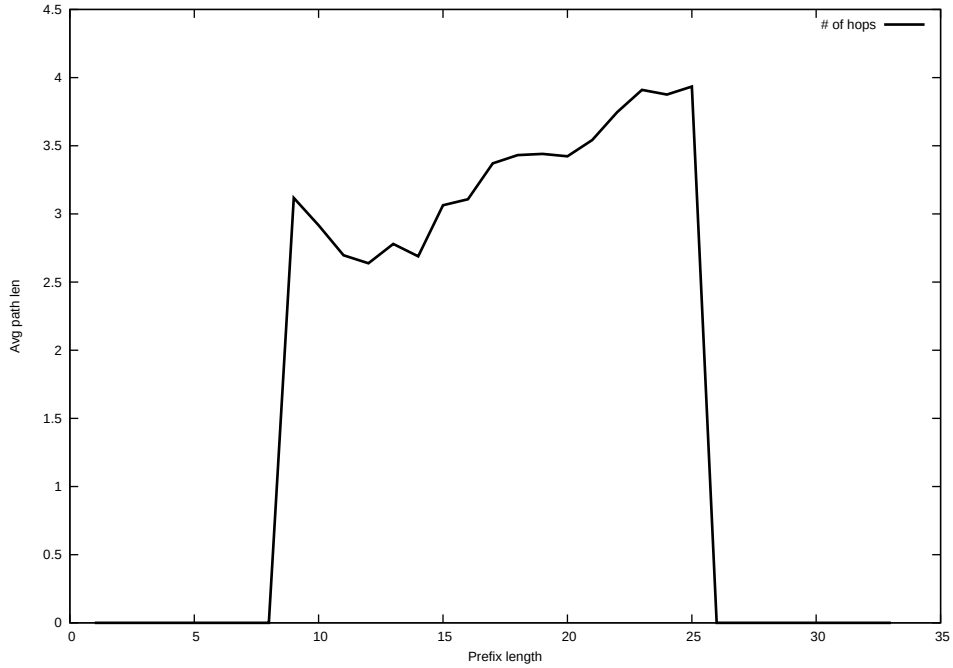
2015-05-23



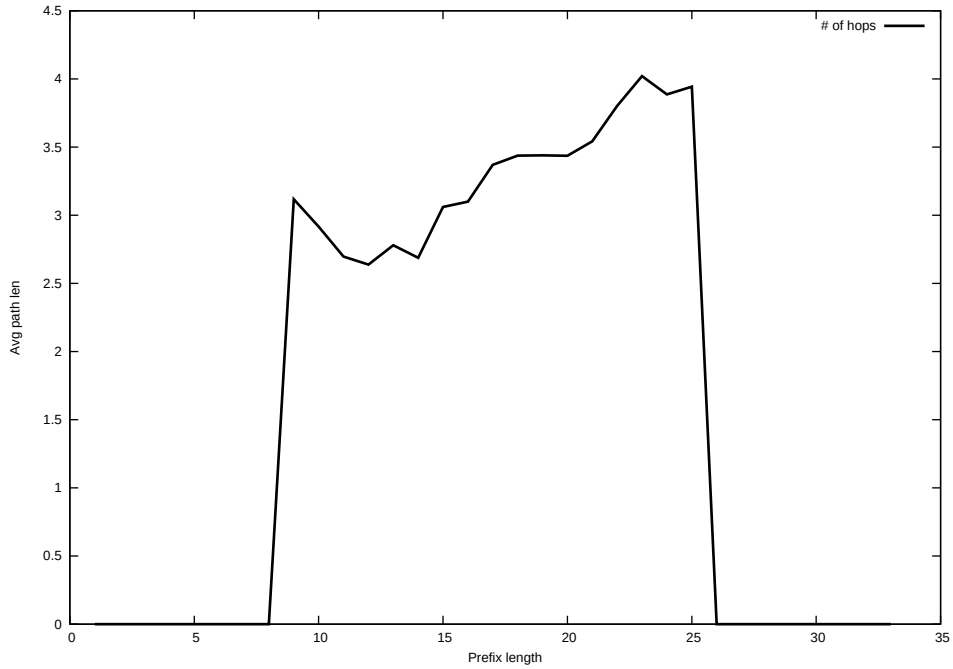
2015-05-24



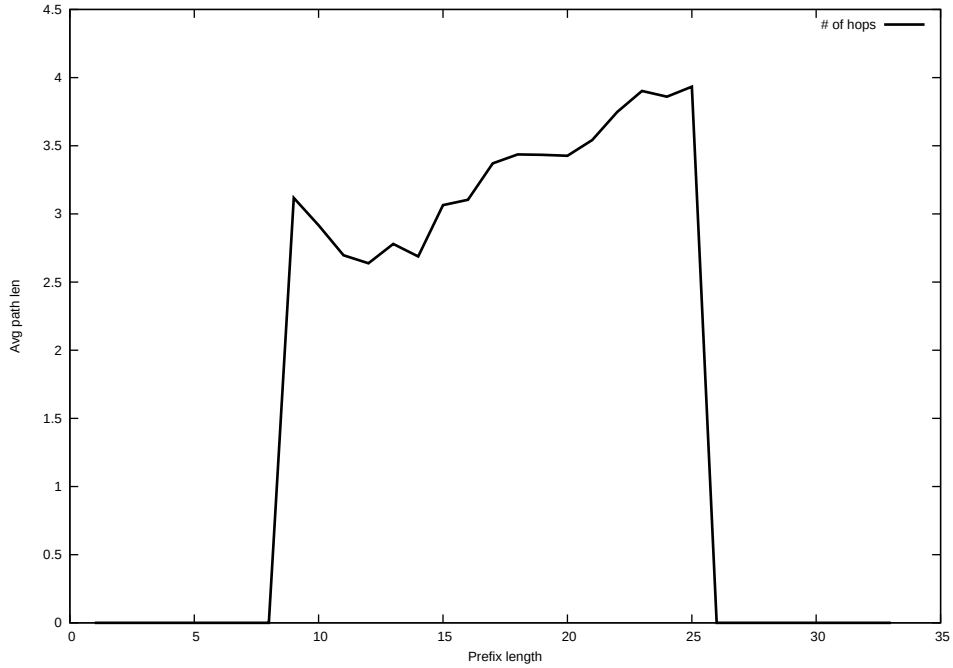
2015-05-25



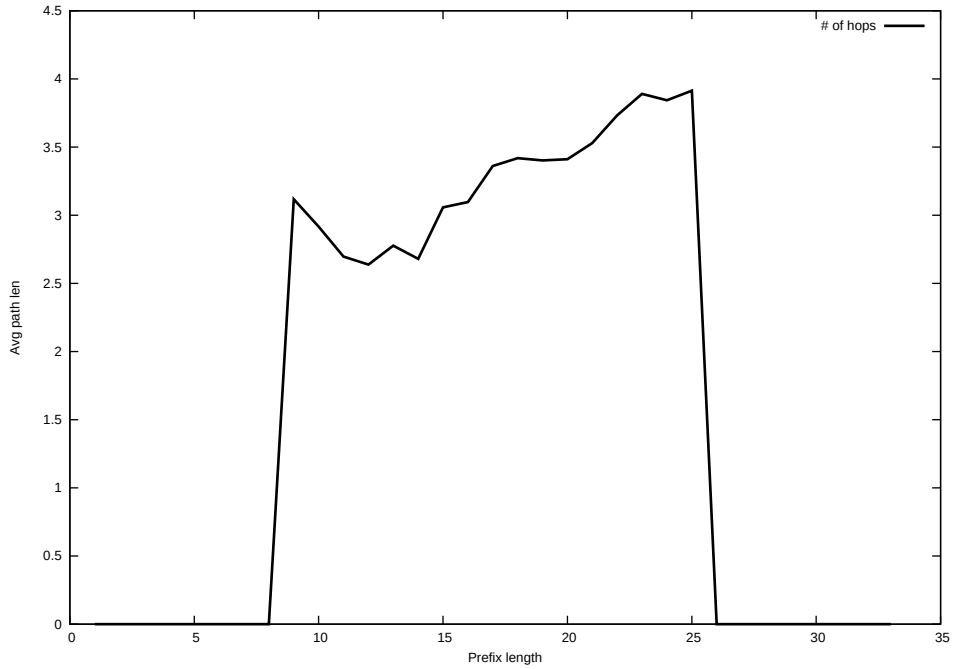
2015-05-26



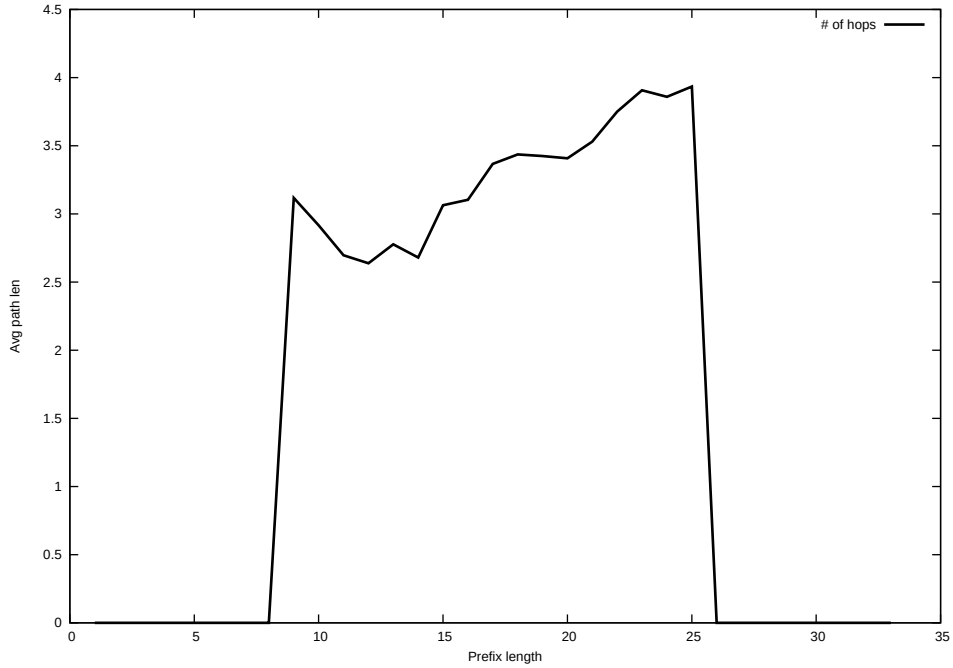
2015-05-27



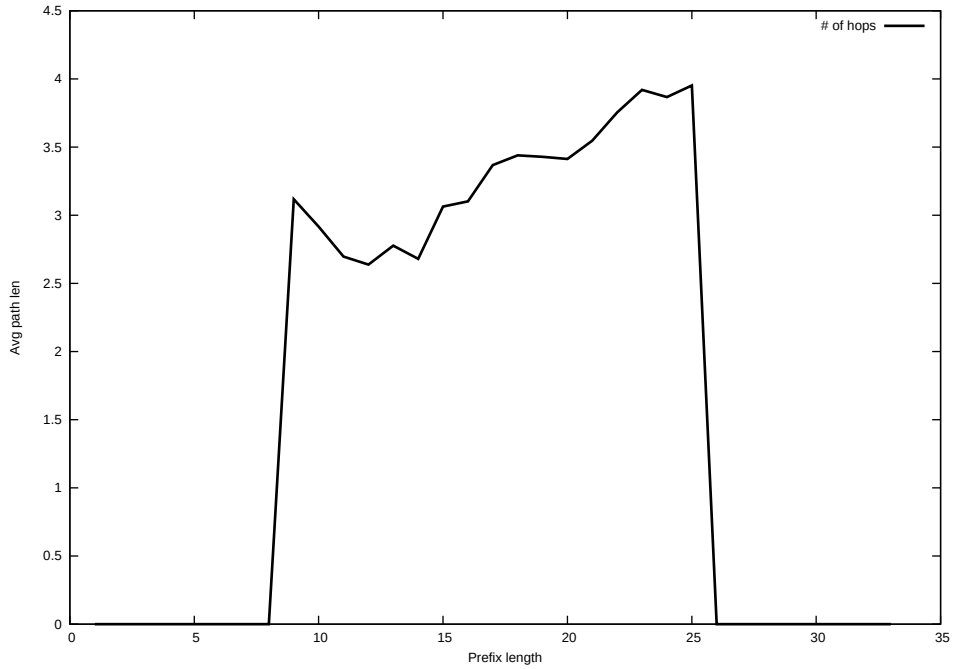
2015-05-28



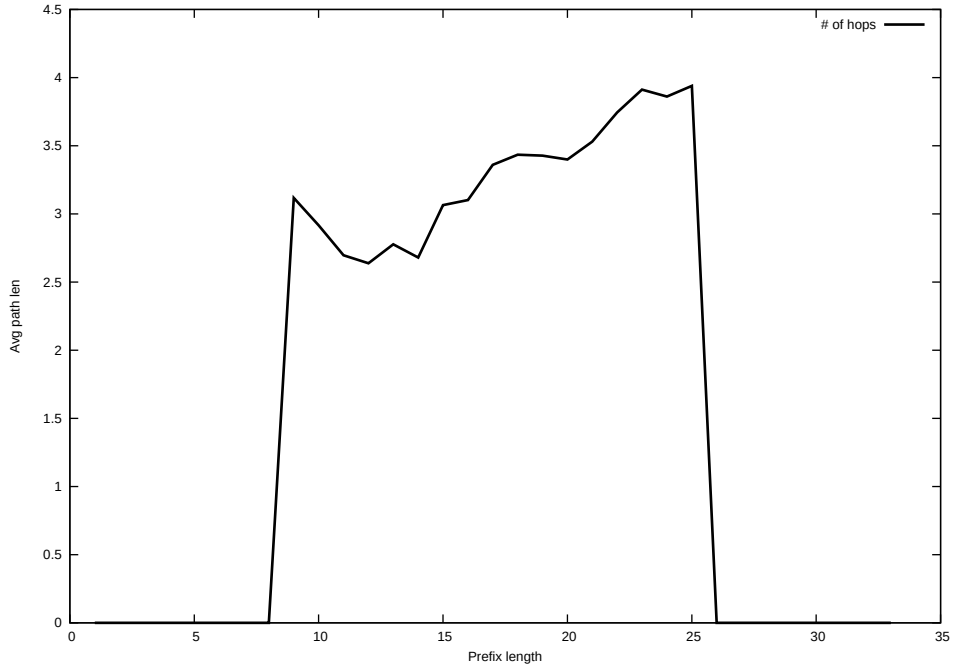
2015-05-29



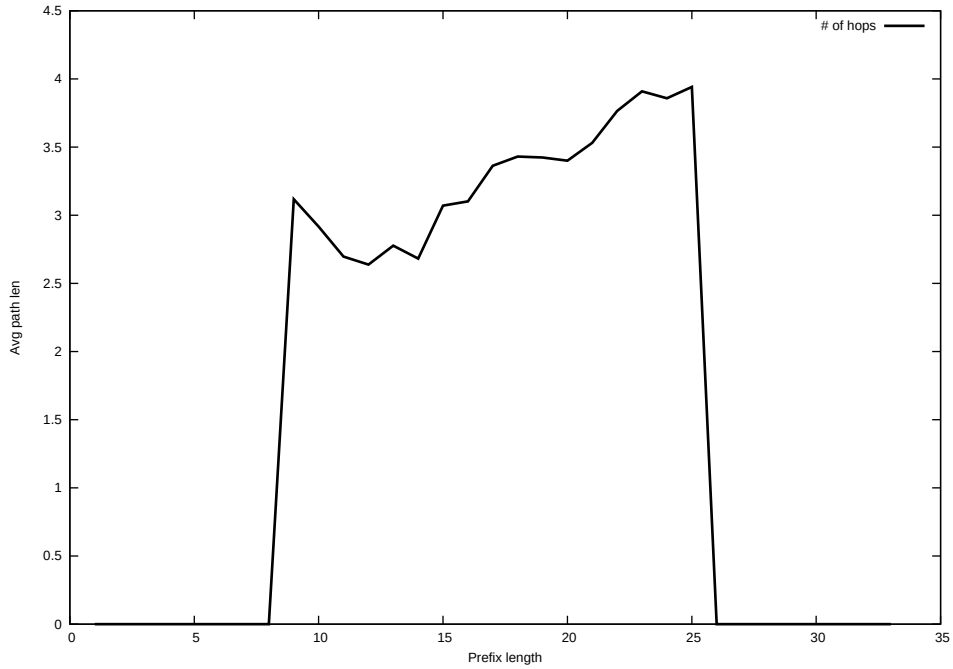
2015-05-30



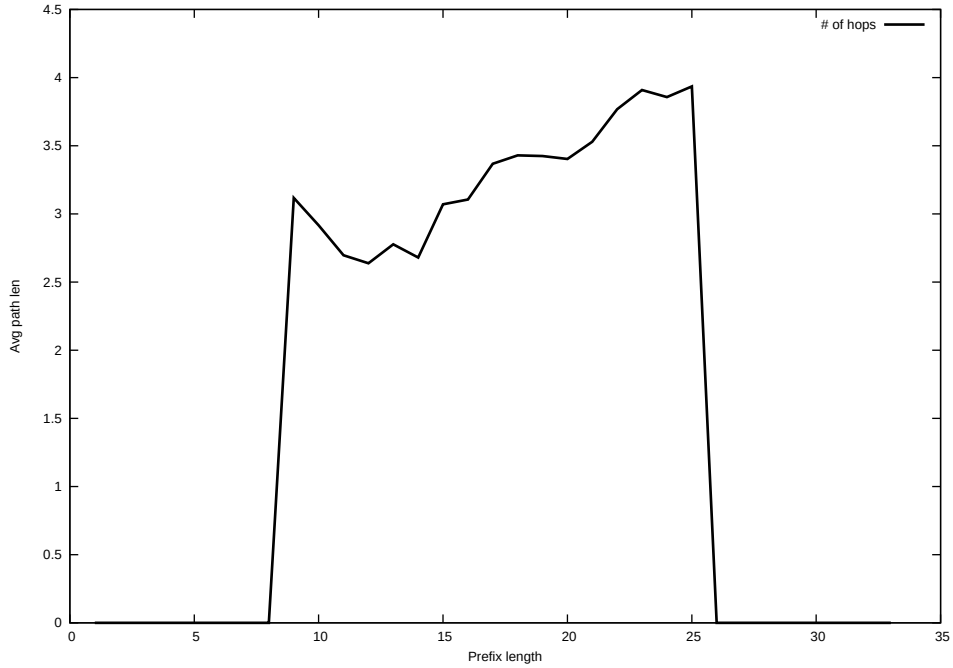
2015-05-31



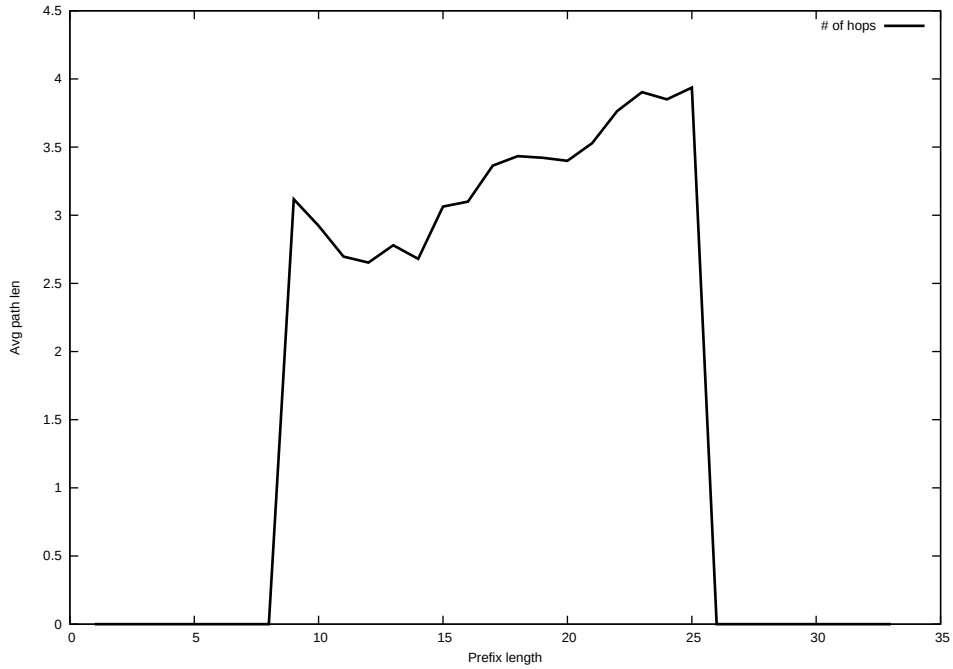
2015-06-01



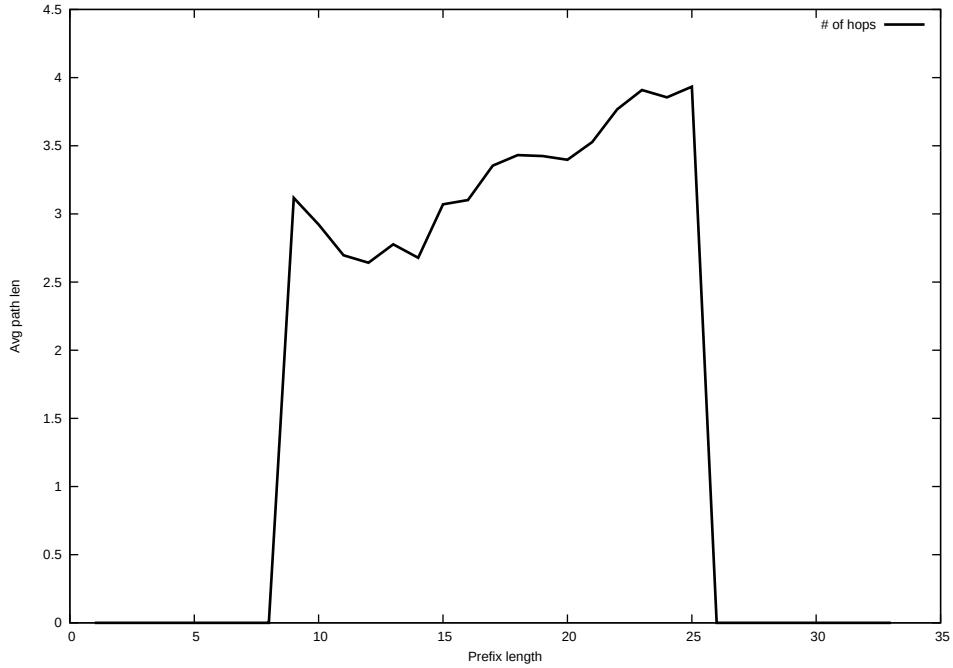
2015-06-02



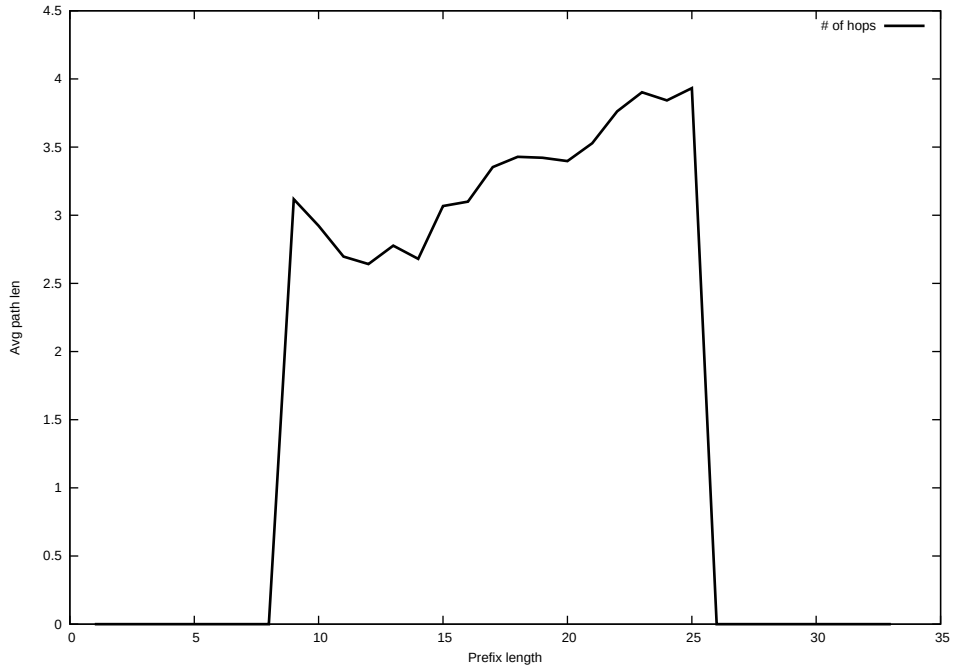
2015-06-03



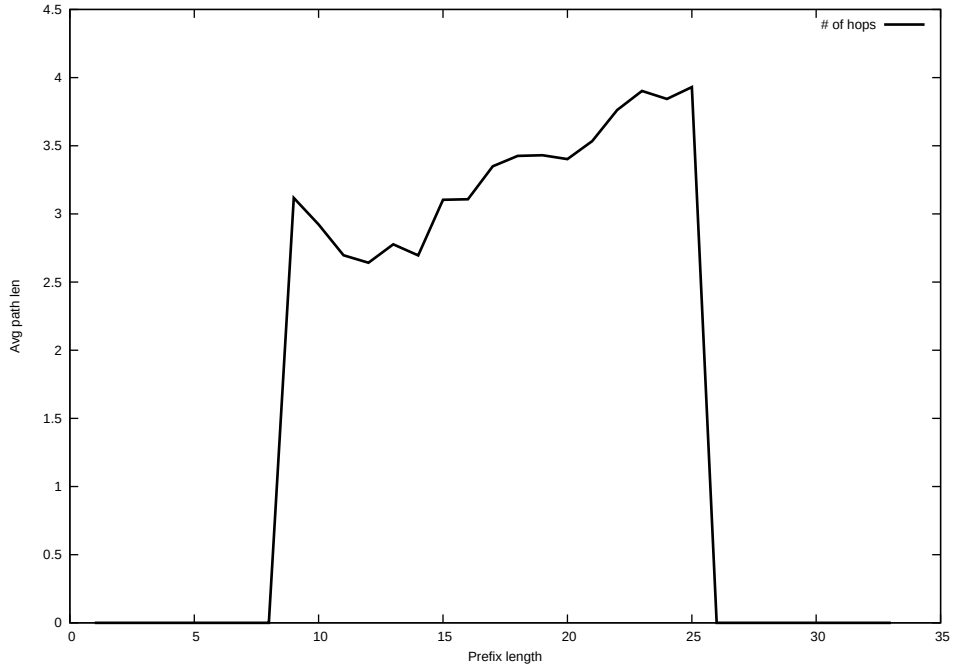
2015-06-04



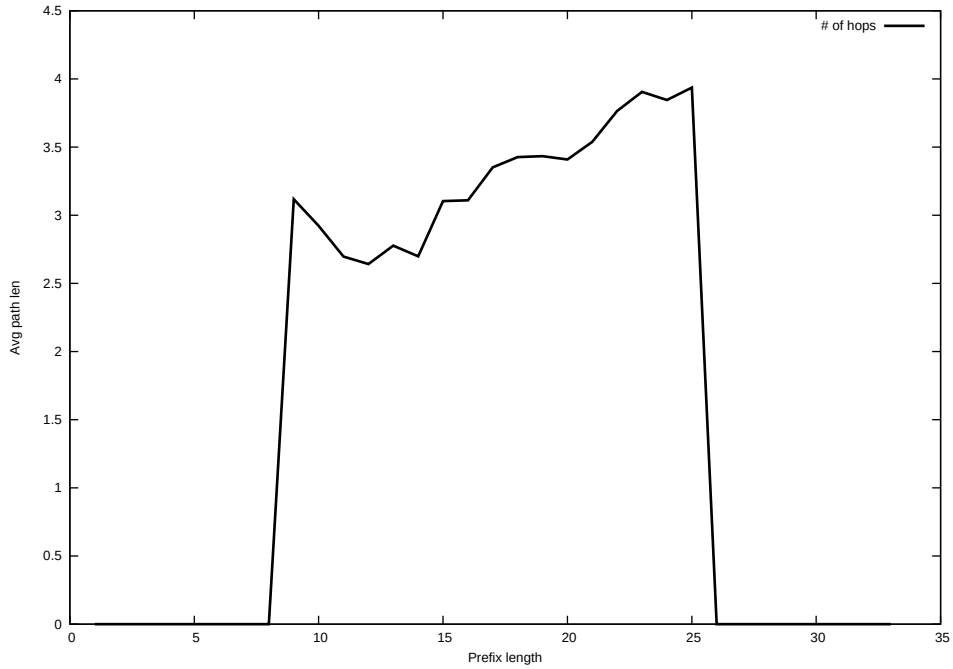
2015-06-05



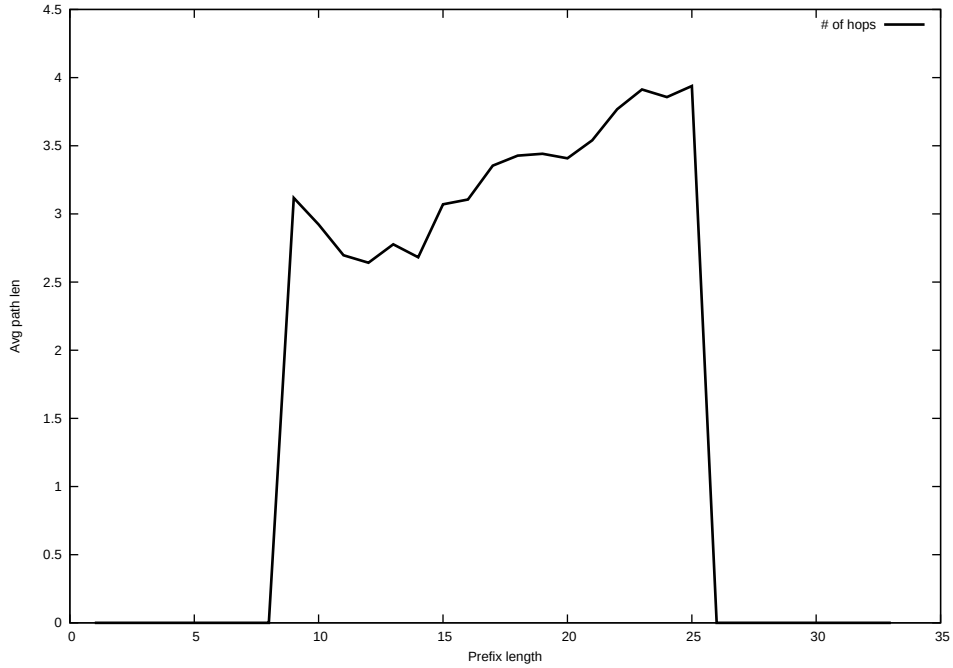
2015-06-06



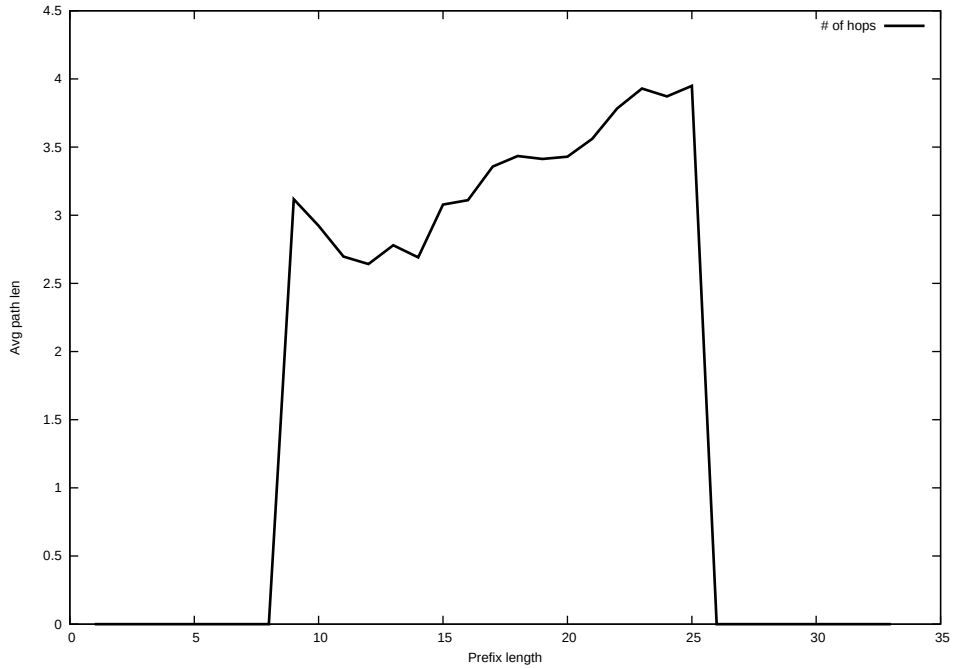
2015-06-07



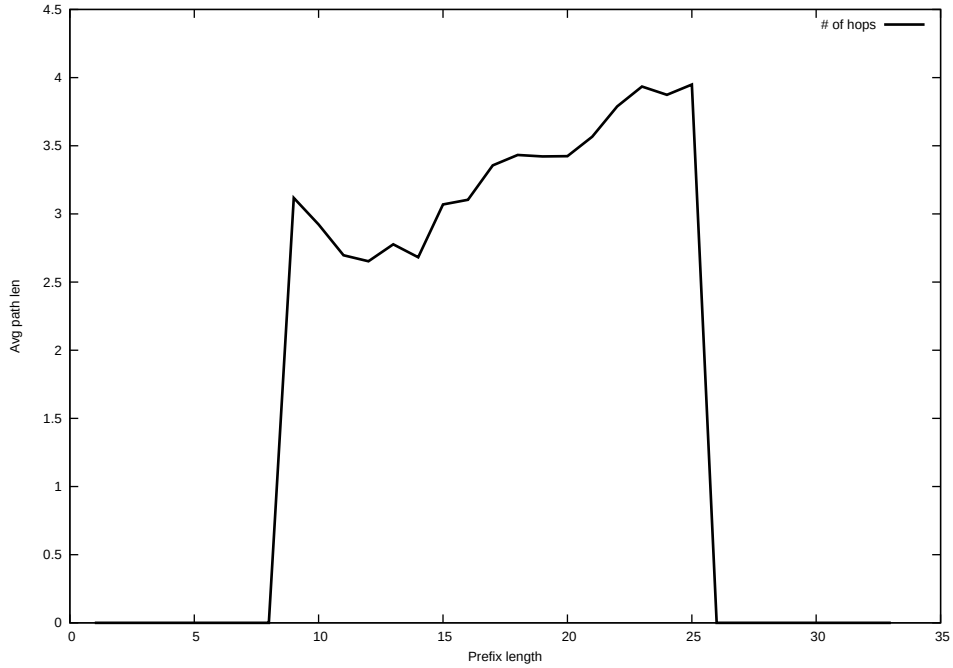
2015-06-08



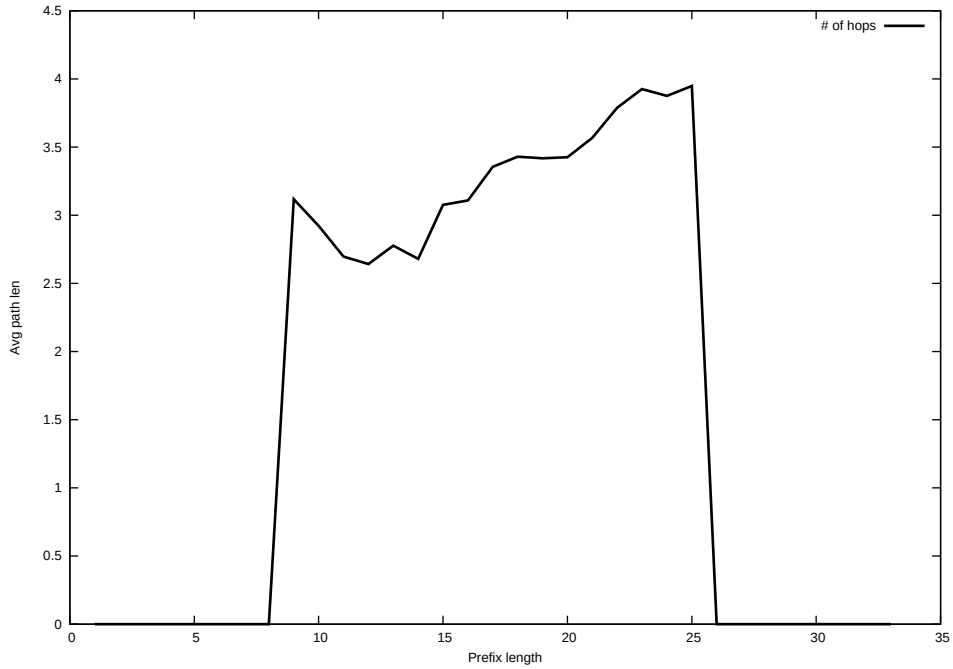
2015-06-09



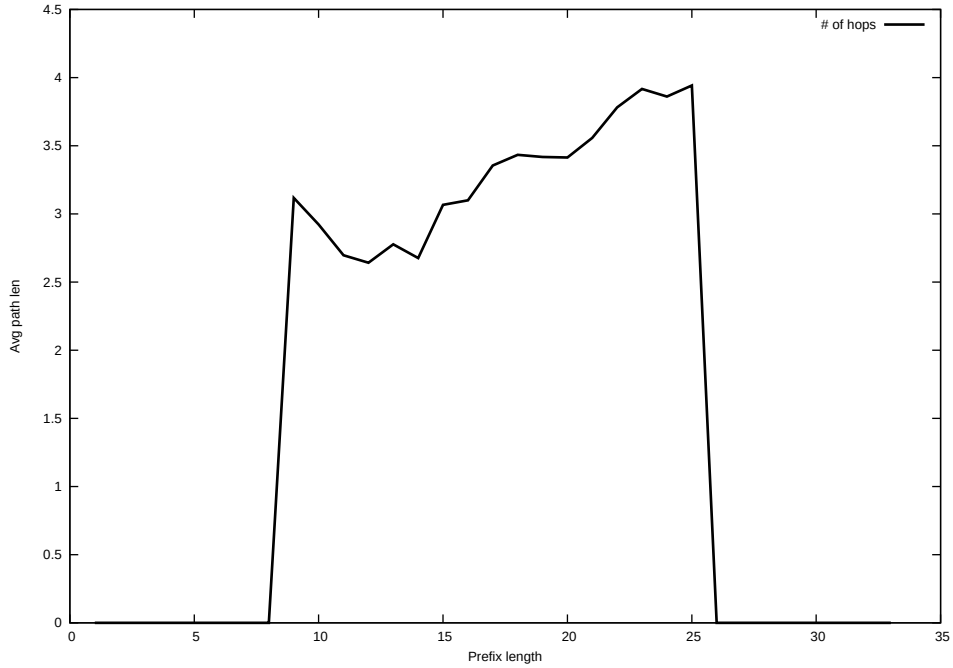
2015-06-10



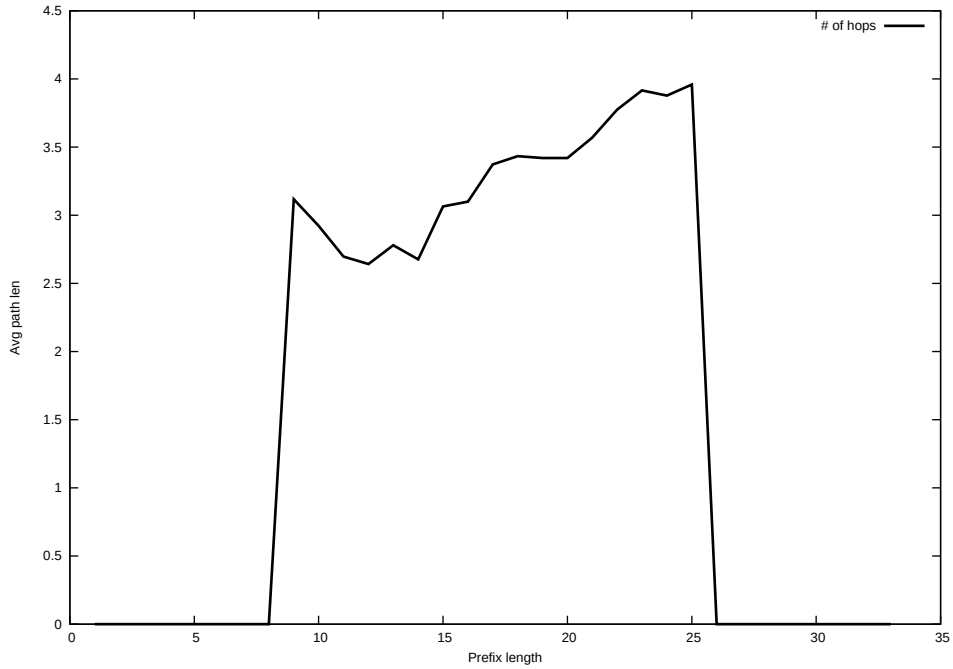
2015-06-11



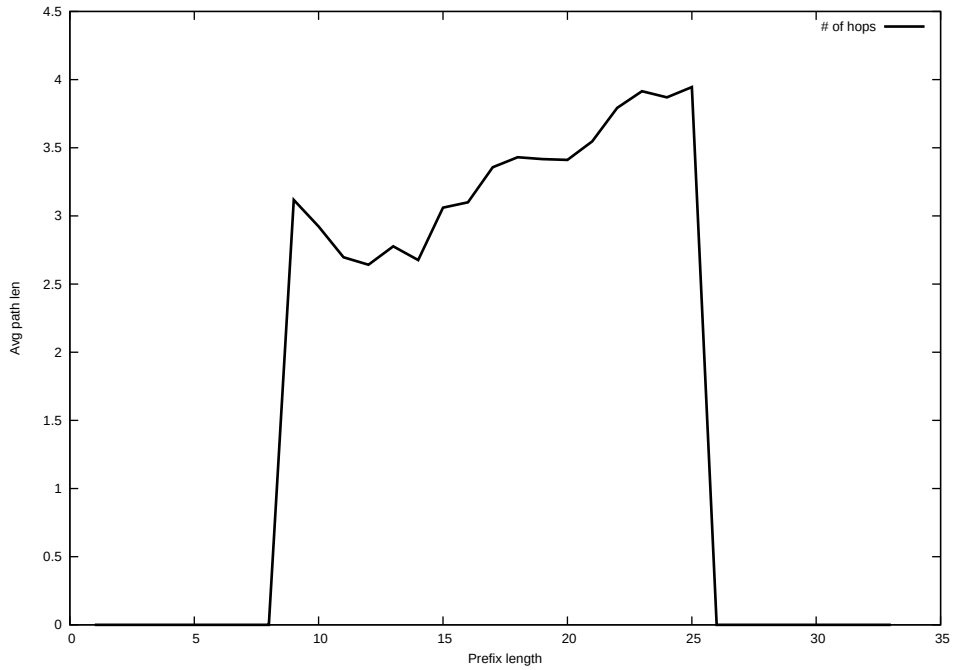
2015-06-12



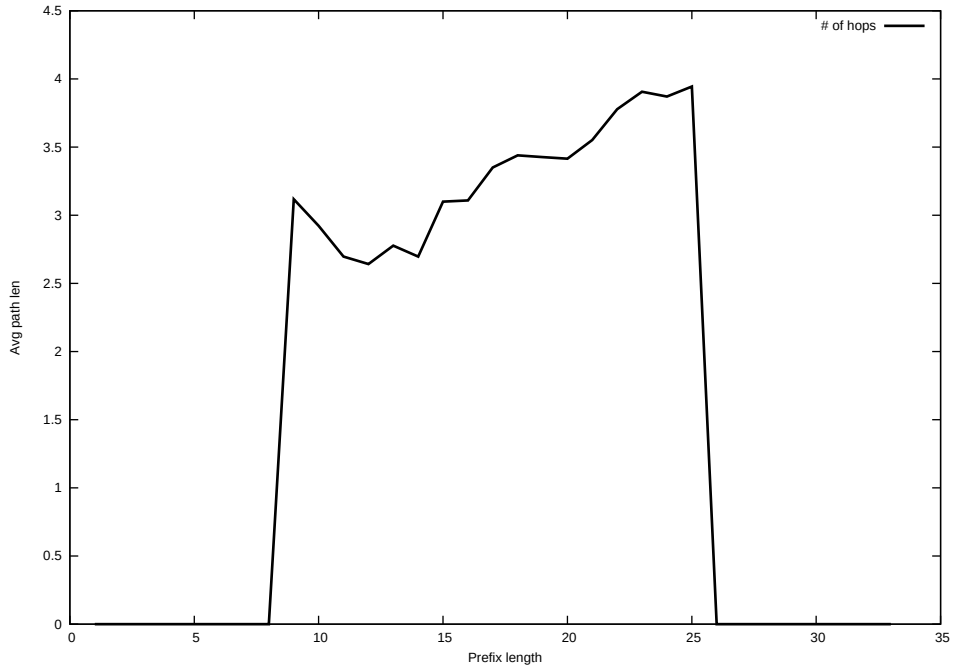
2015-06-13



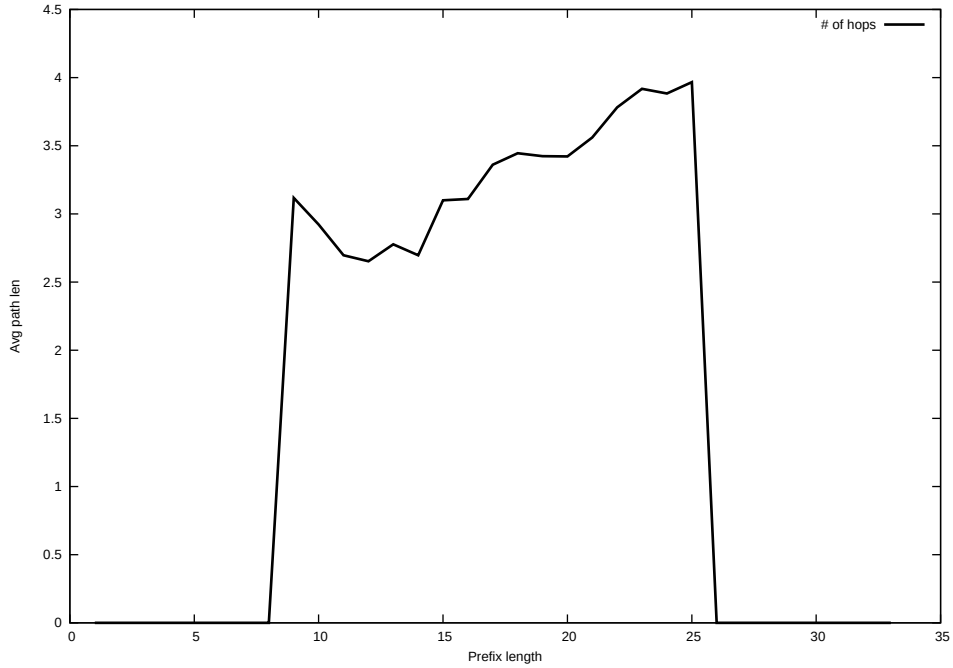
2015-06-14



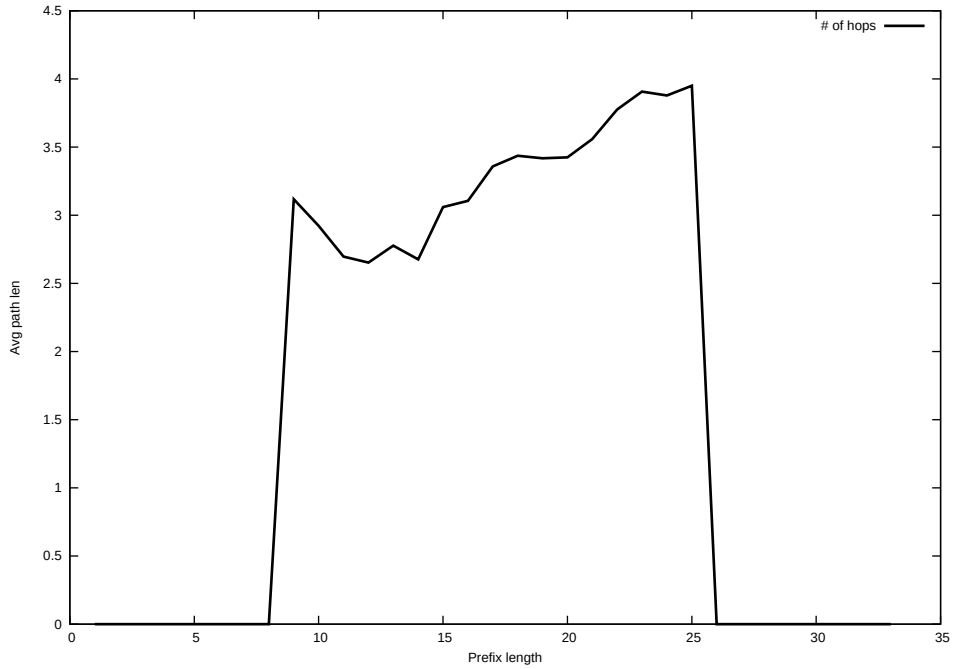
2015-06-15



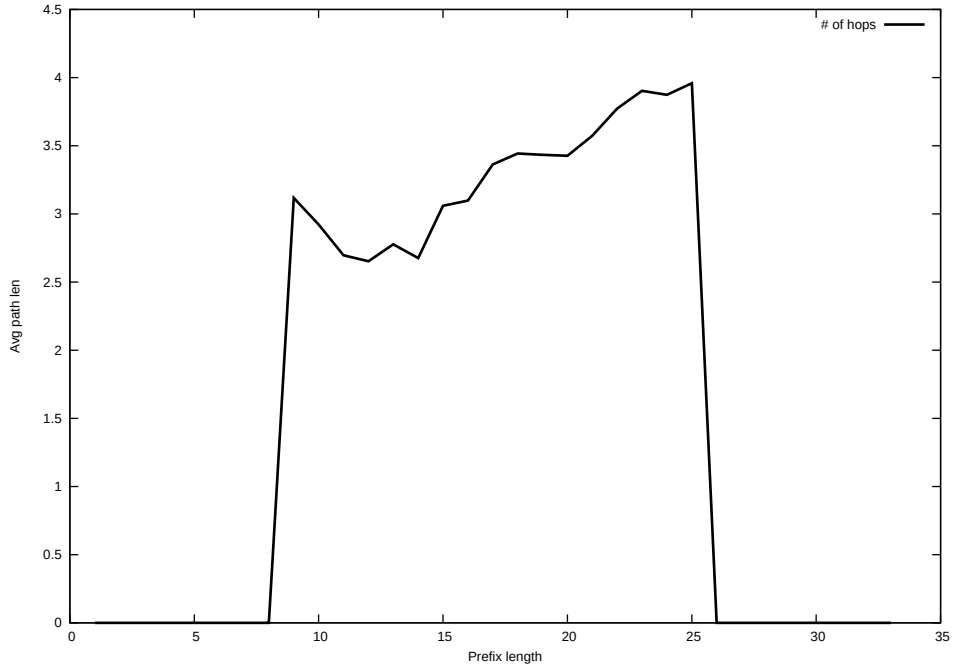
2015-06-16



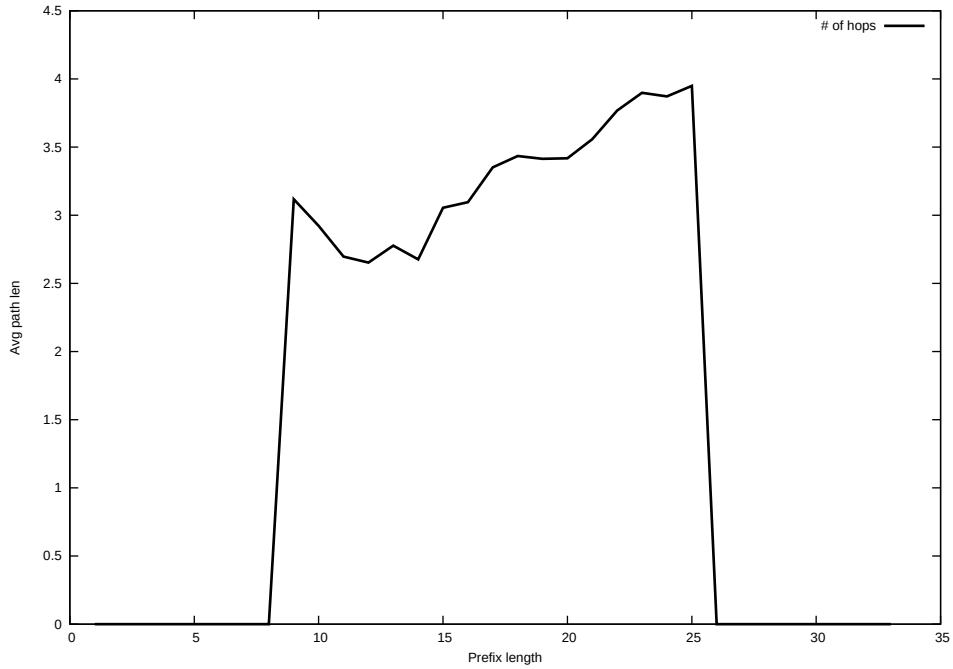
2015-06-17



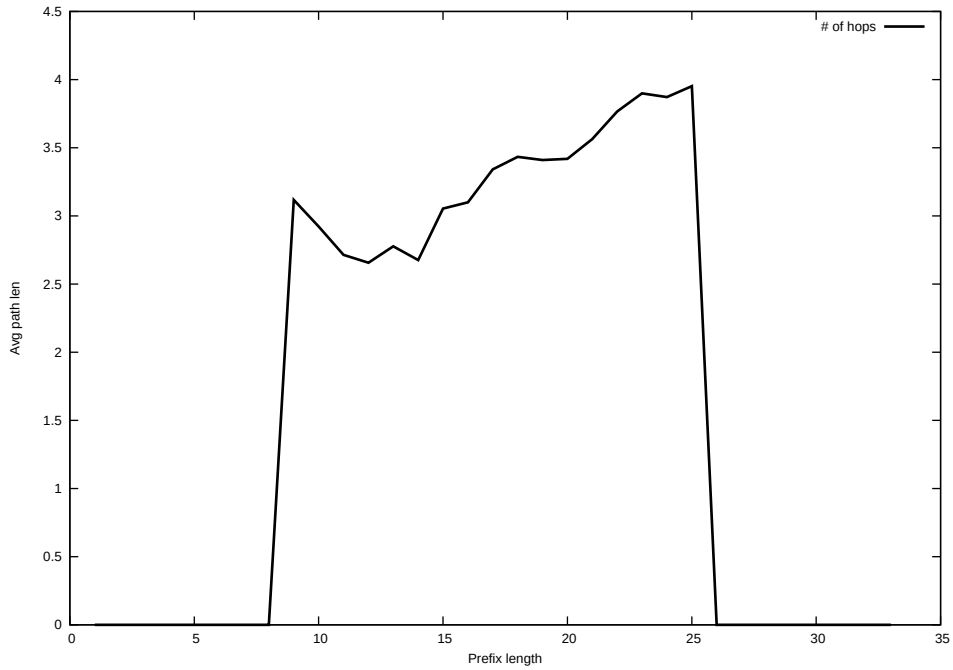
2015-06-18



2015-06-19

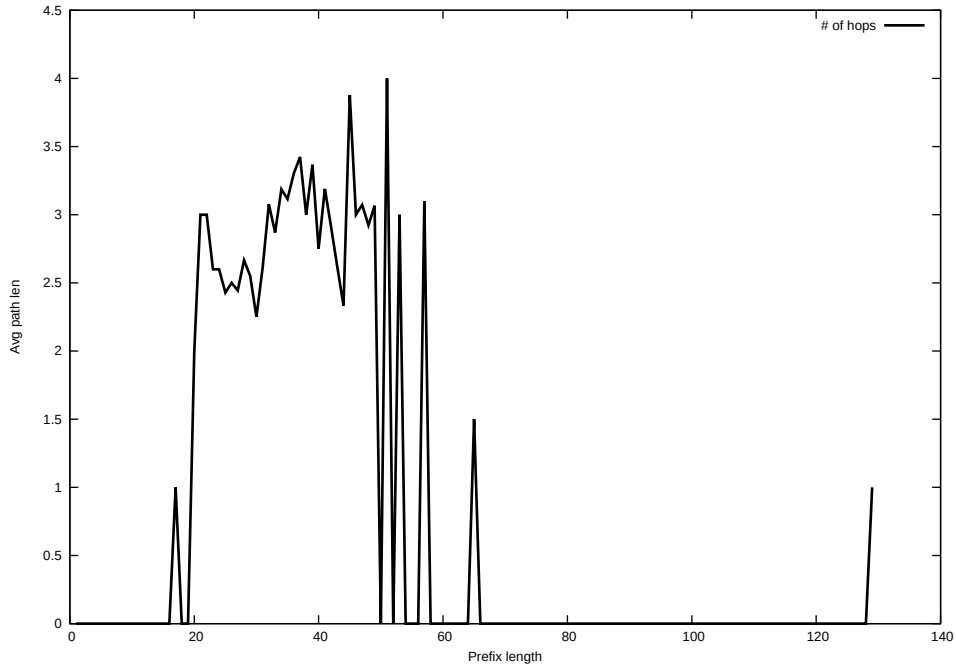


2015-06-20

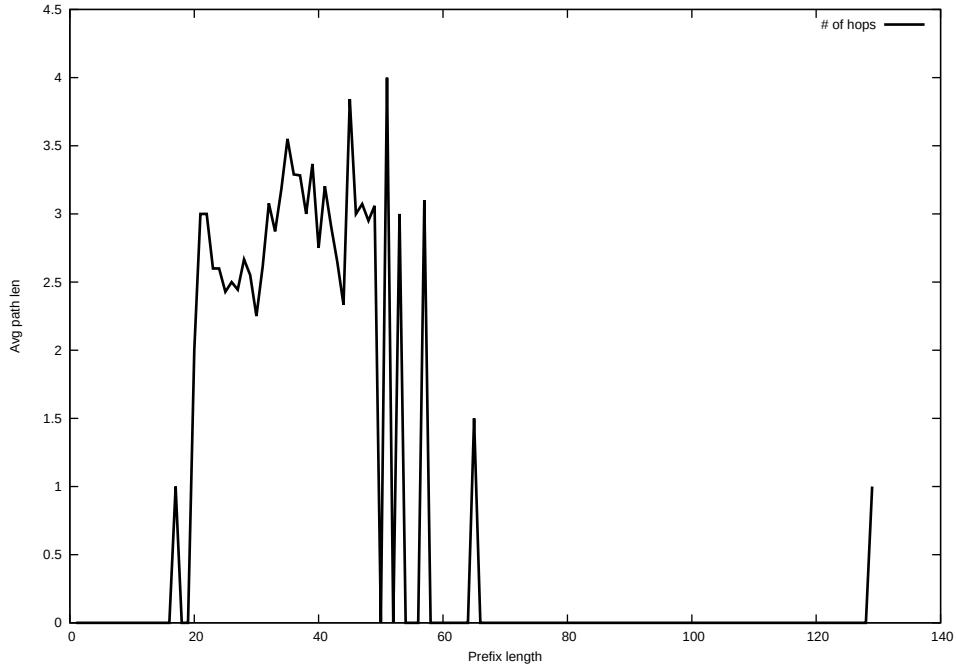


2015-06-21

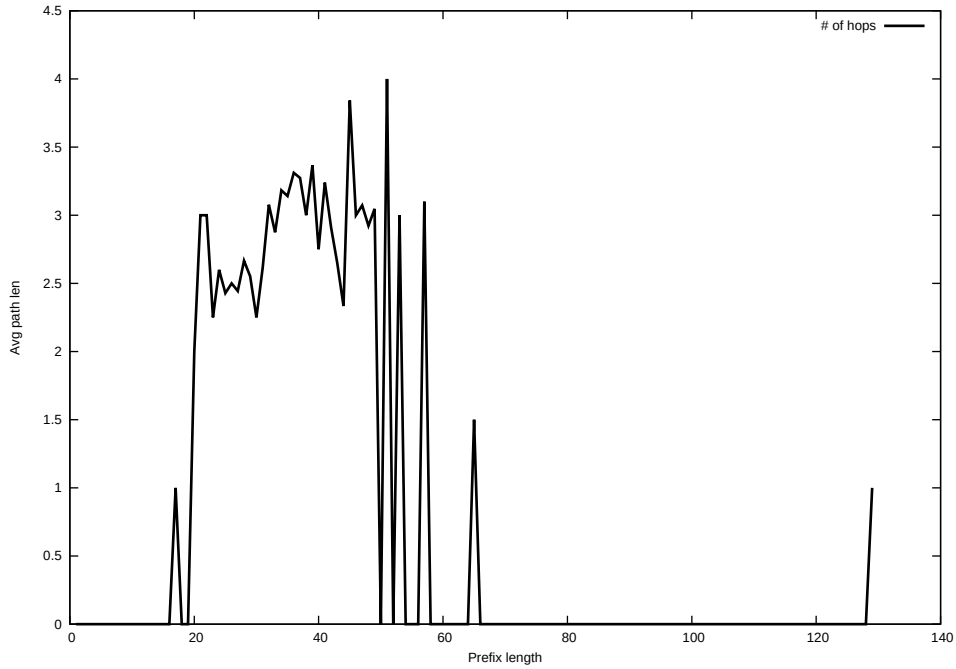
A.7 Daily IPv6 BGP avg path length by prefix length



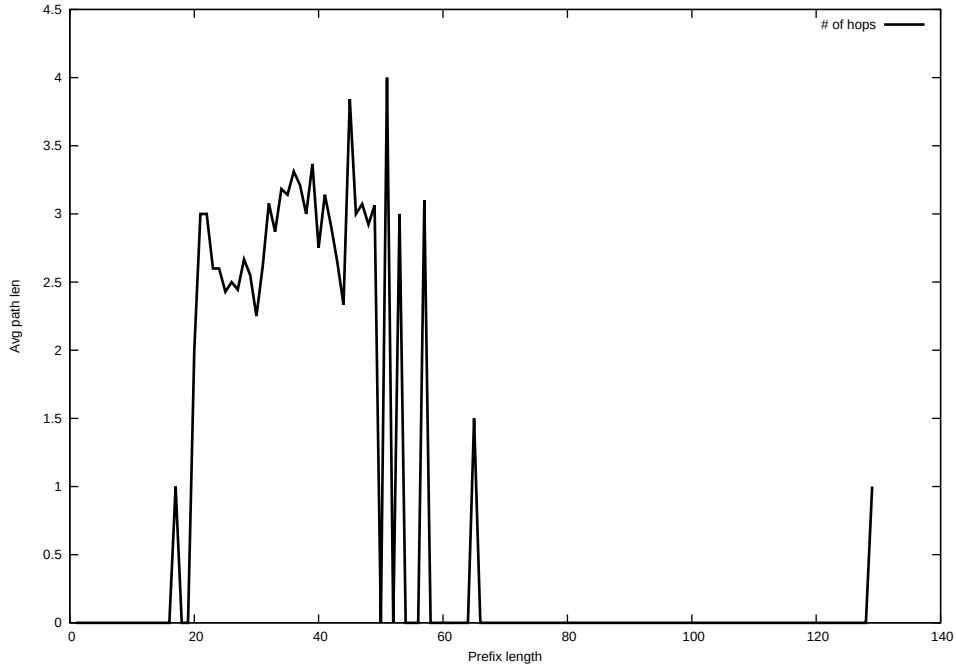
2012-03-22



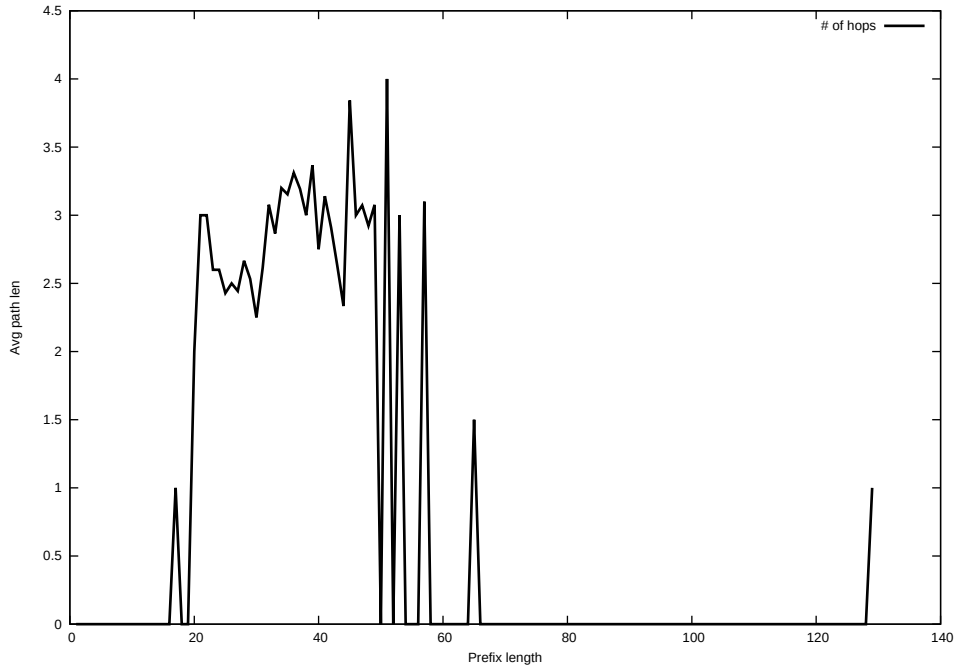
2012-03-23



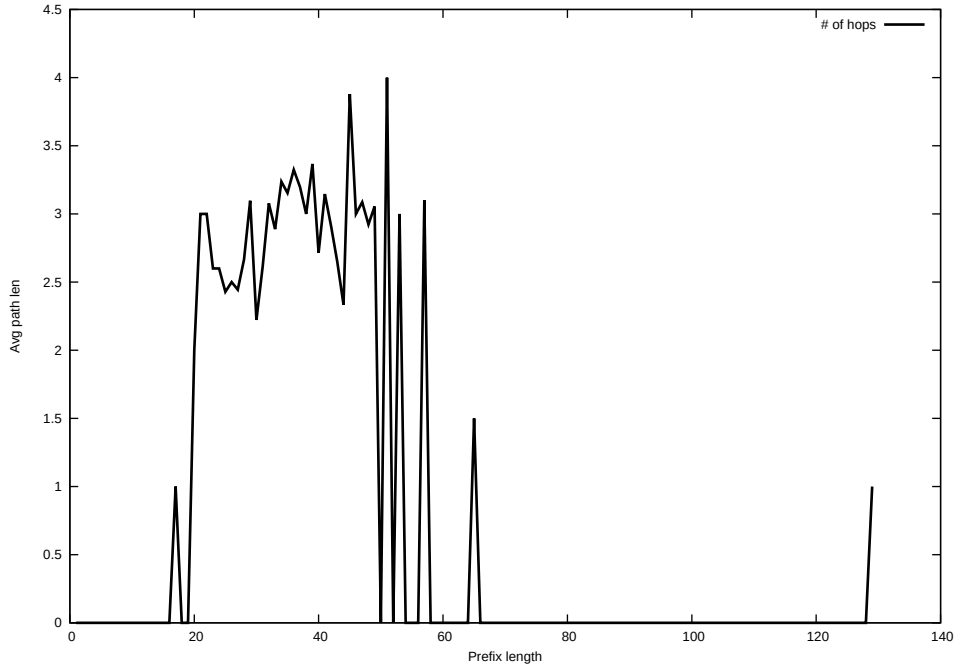
2012-03-24



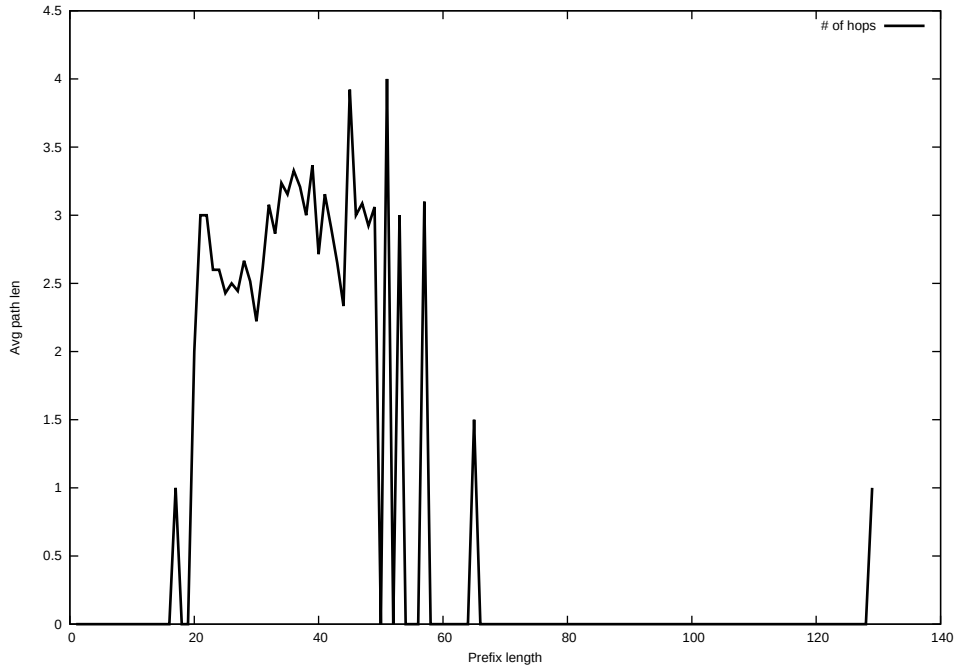
2012-03-25



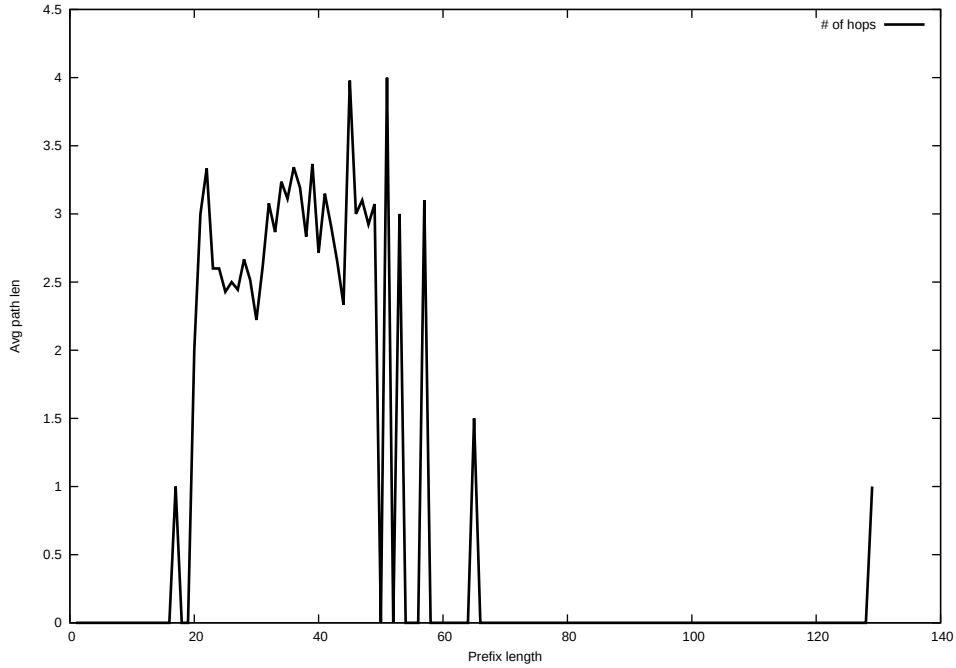
2012-03-26



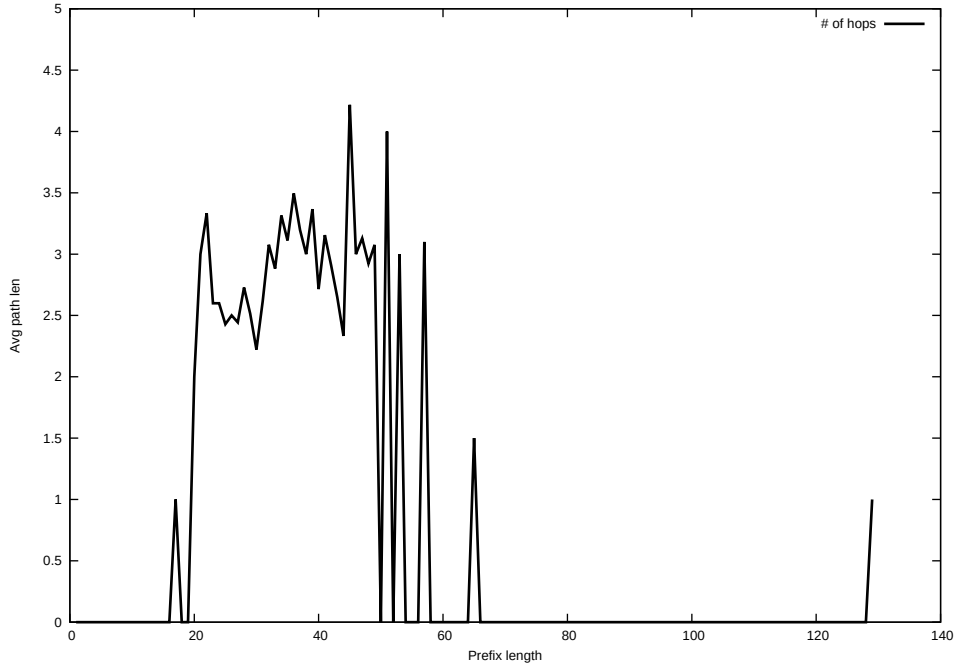
2012-03-27



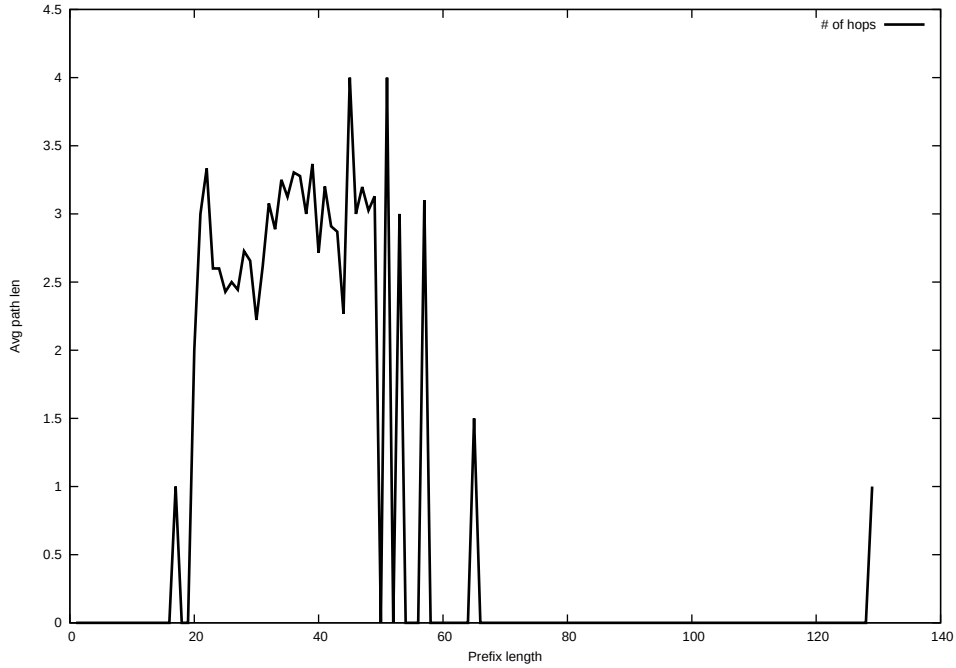
2012-03-28



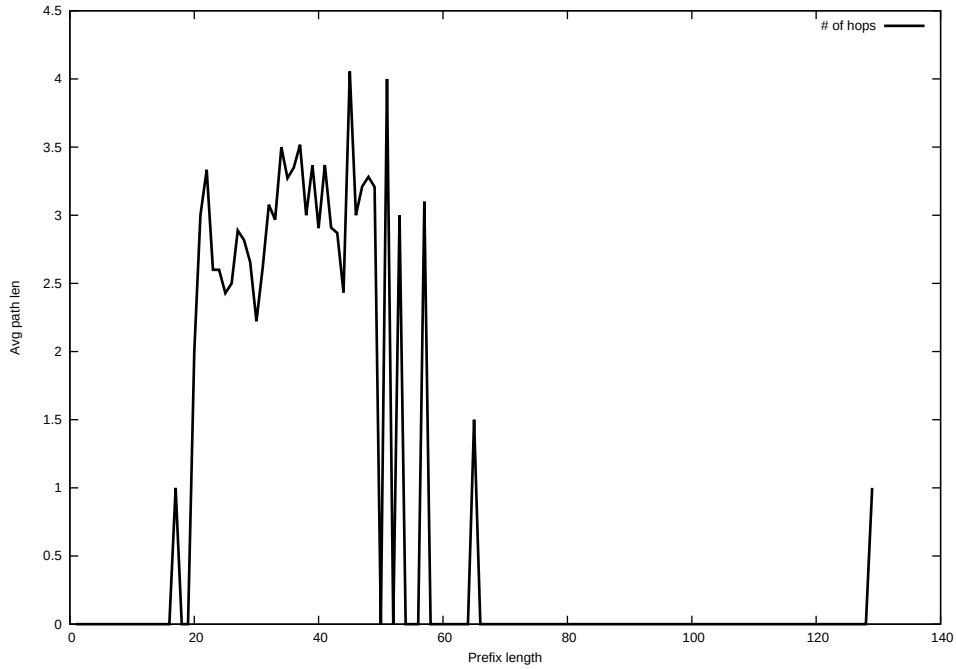
2012-03-29



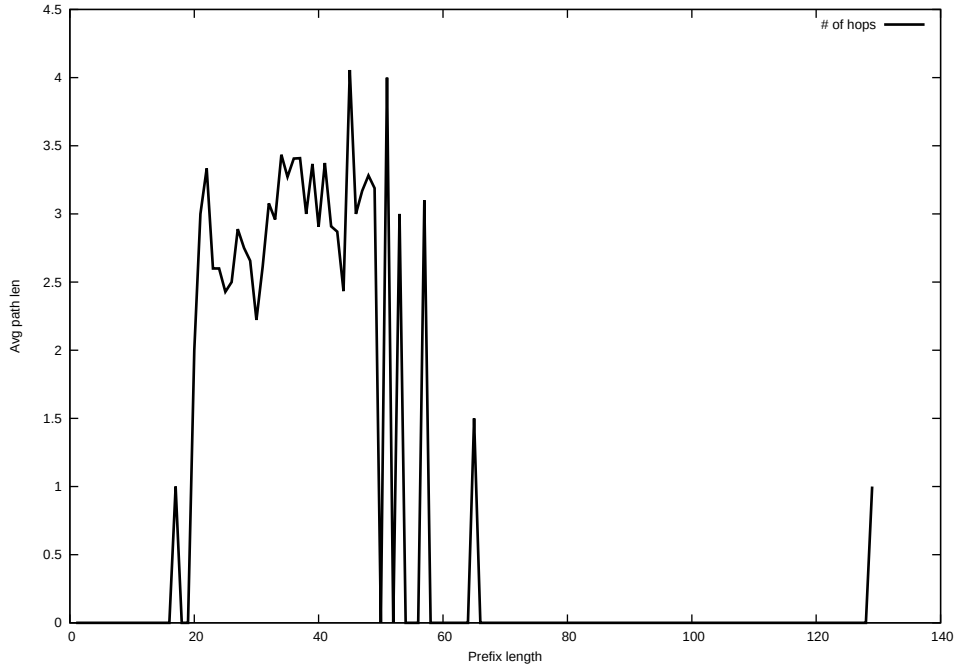
2012-03-30



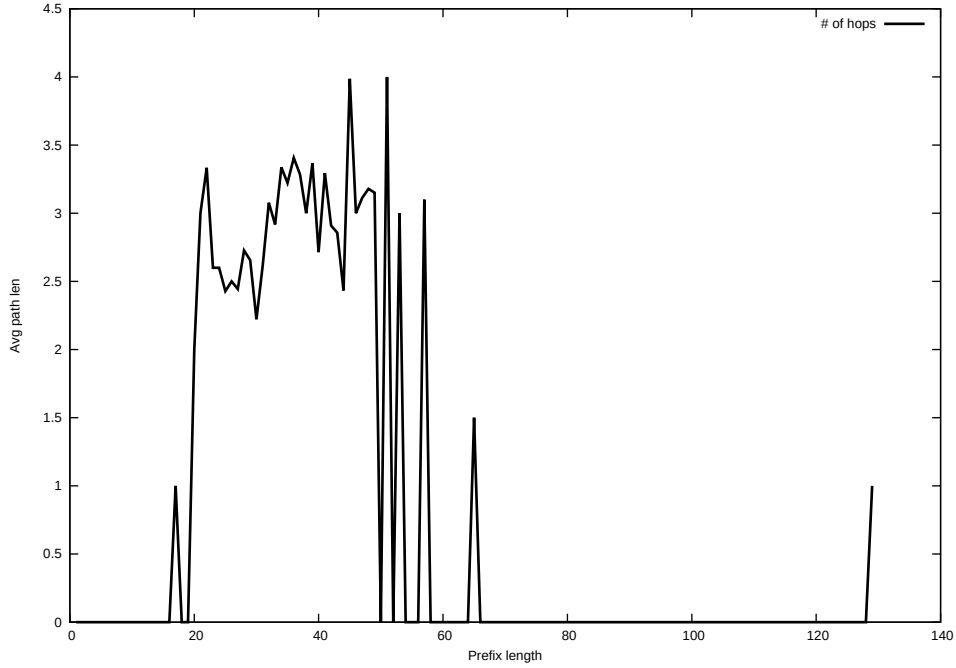
2012-03-31



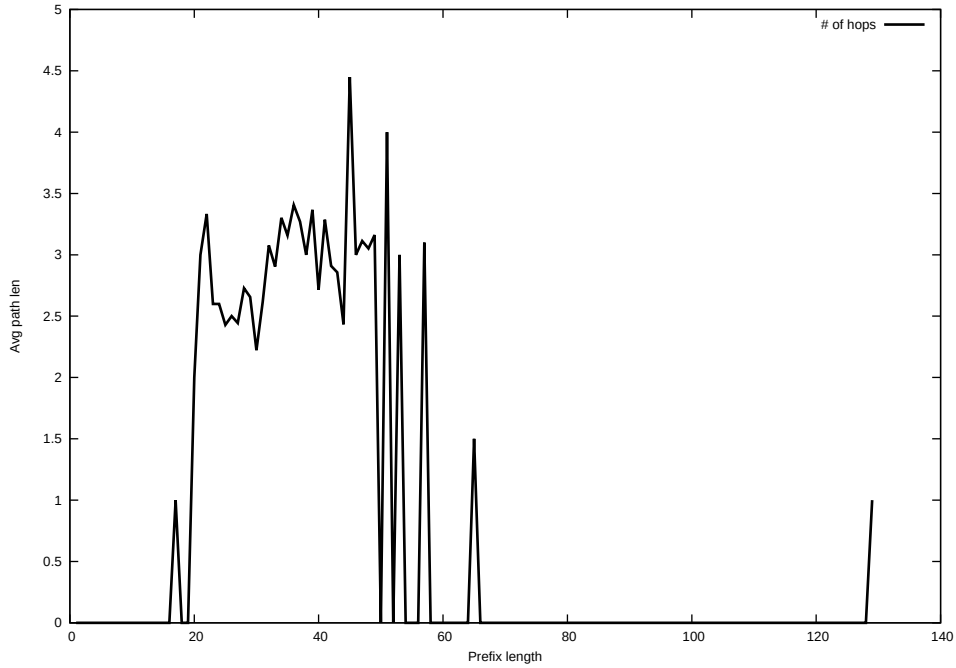
2012-04-01



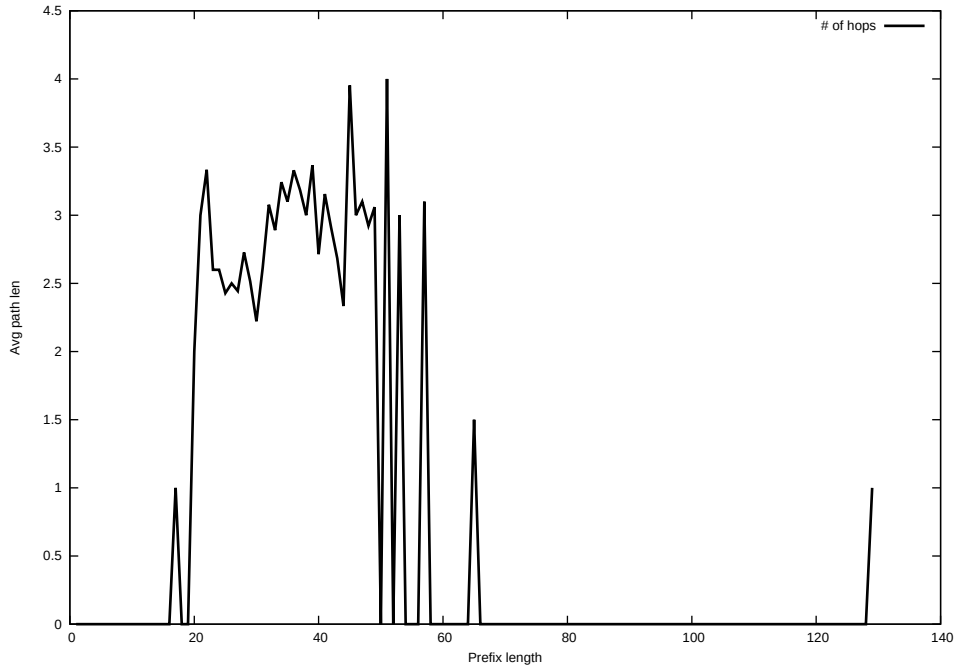
2012-04-02



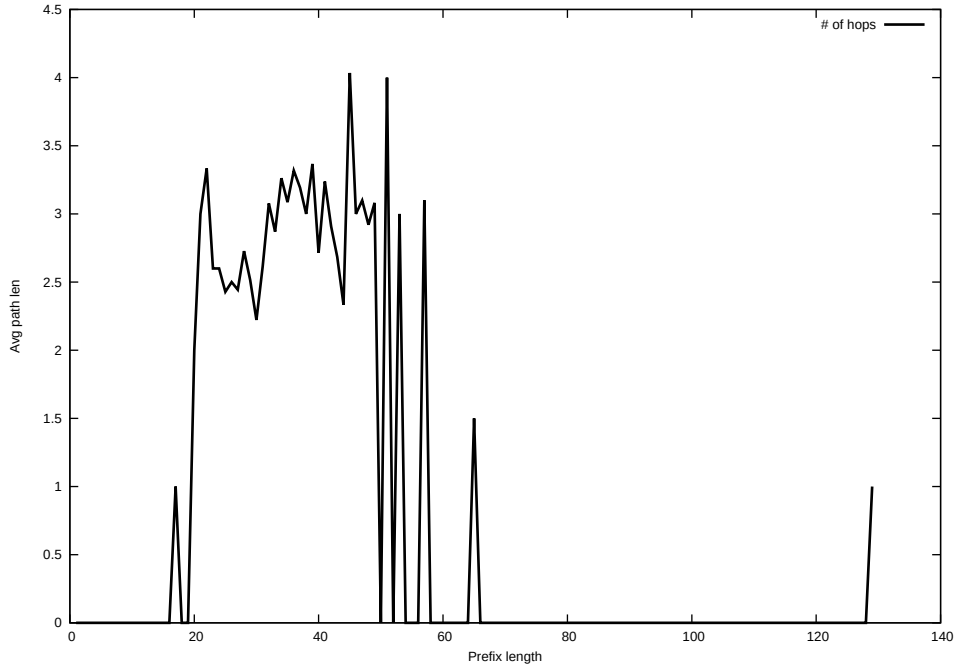
2012-04-03



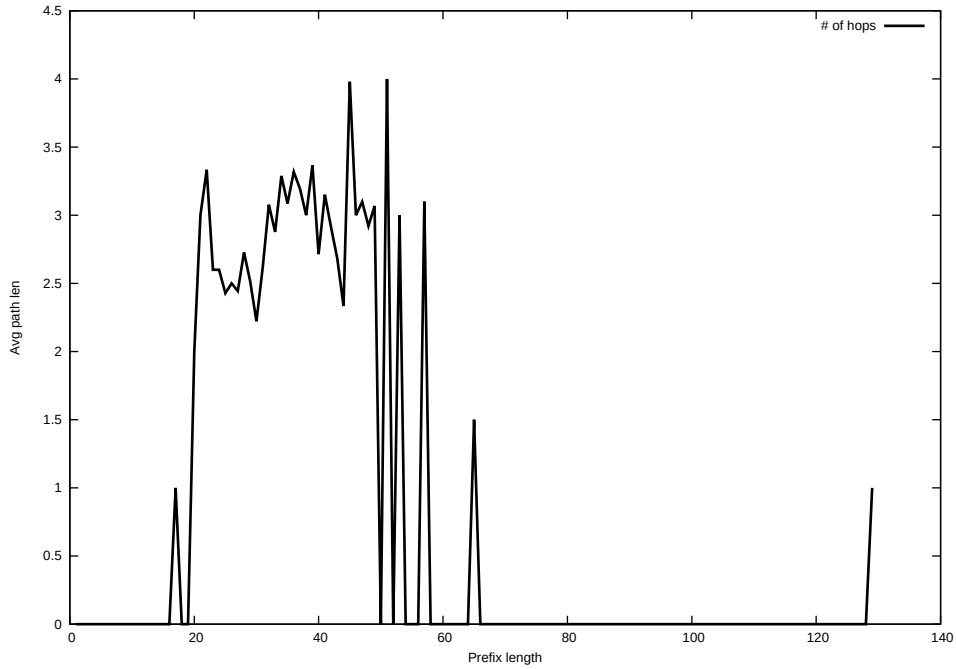
2012-04-04



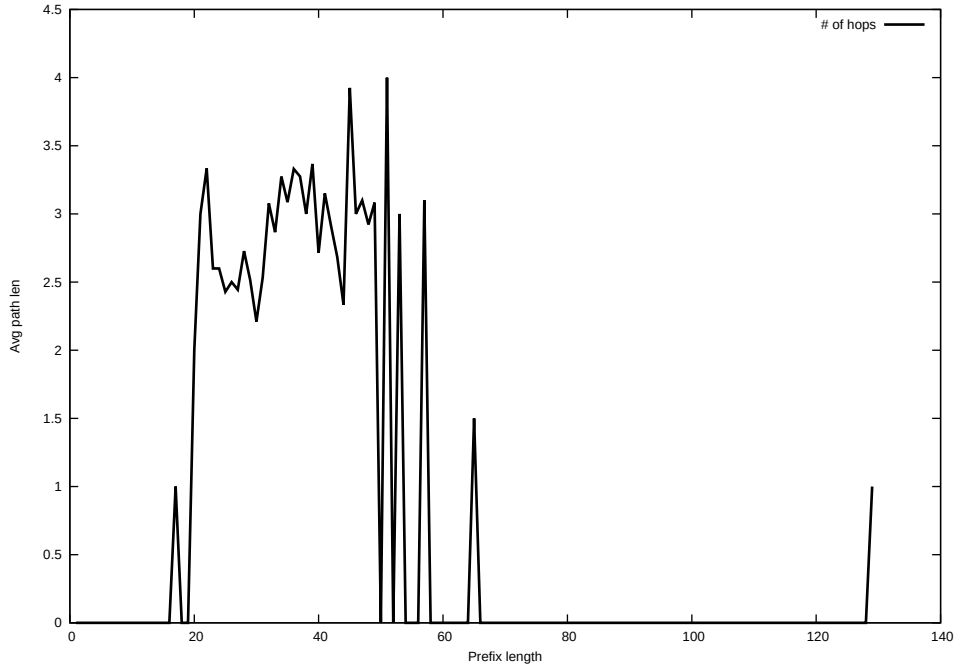
2012-04-05



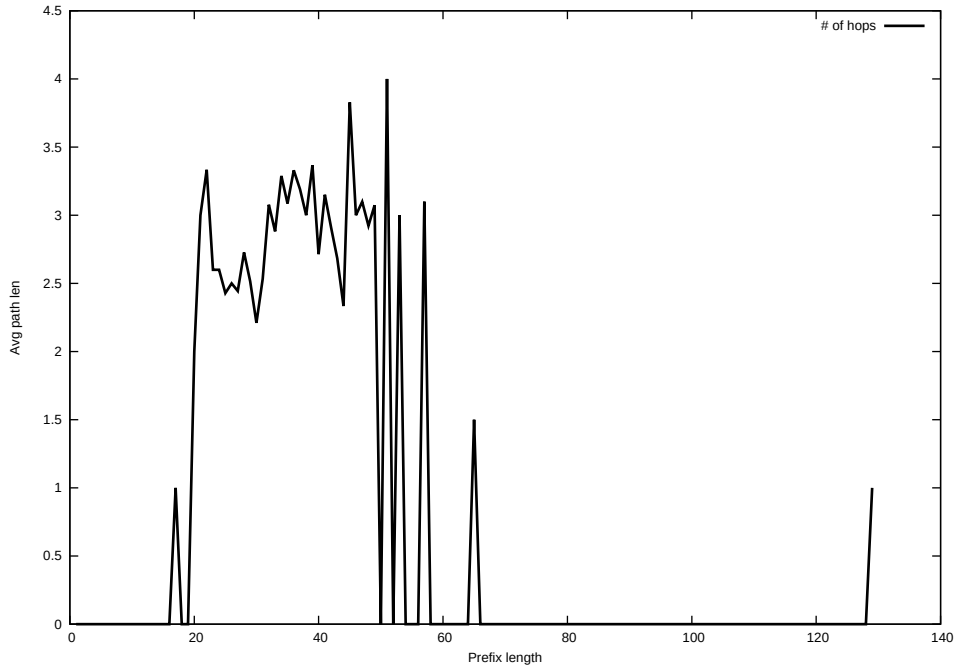
2012-04-06



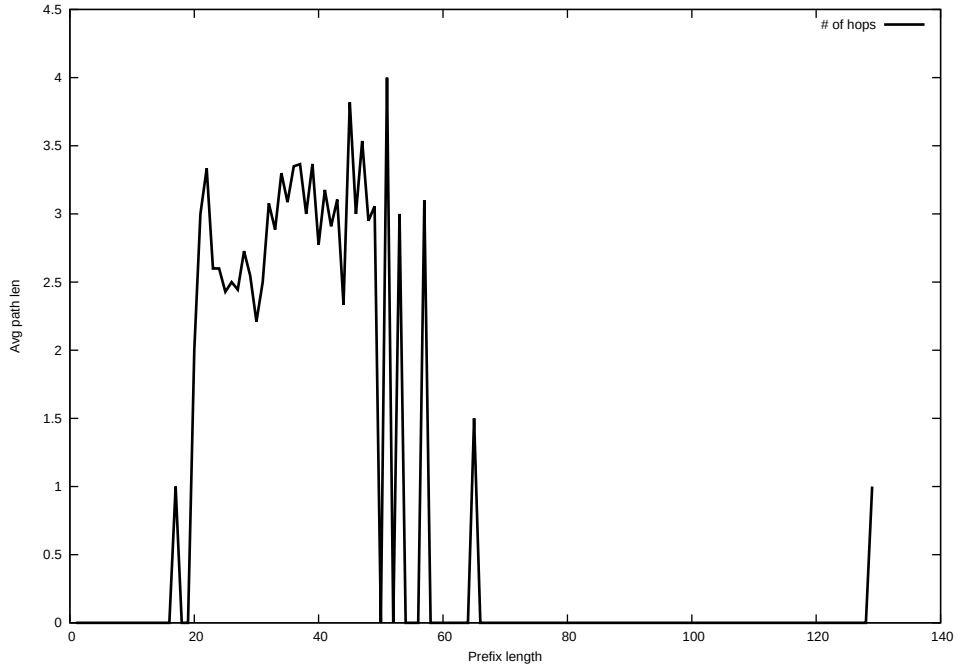
2012-04-07



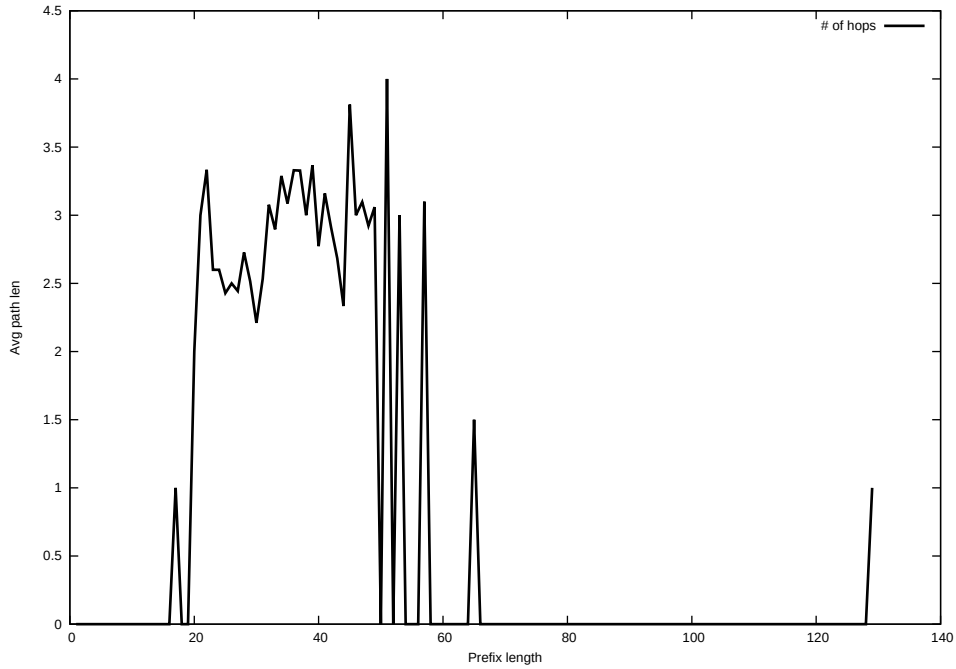
2012-04-08



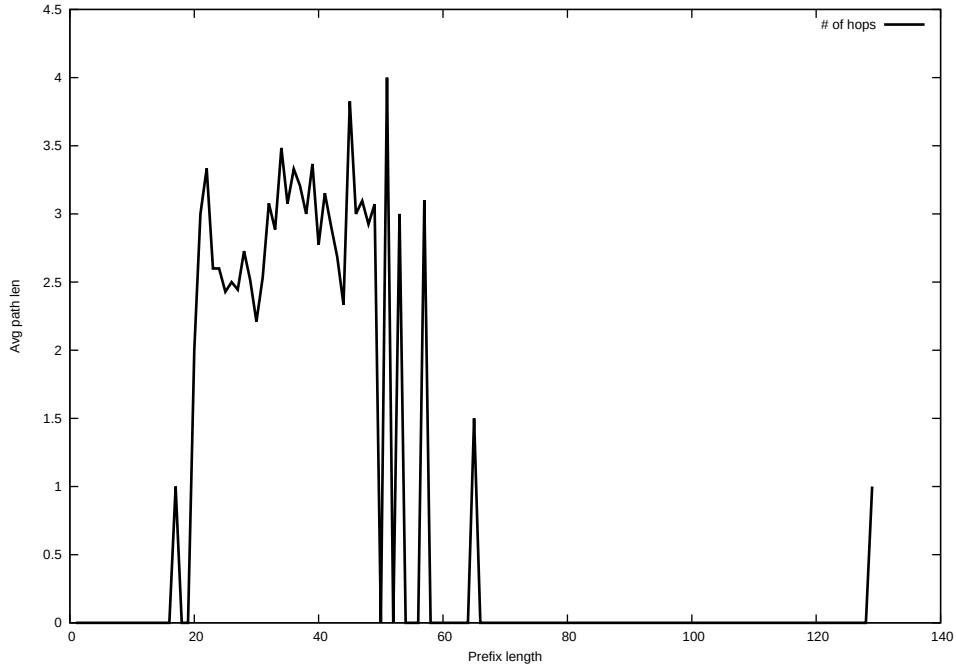
2012-04-09



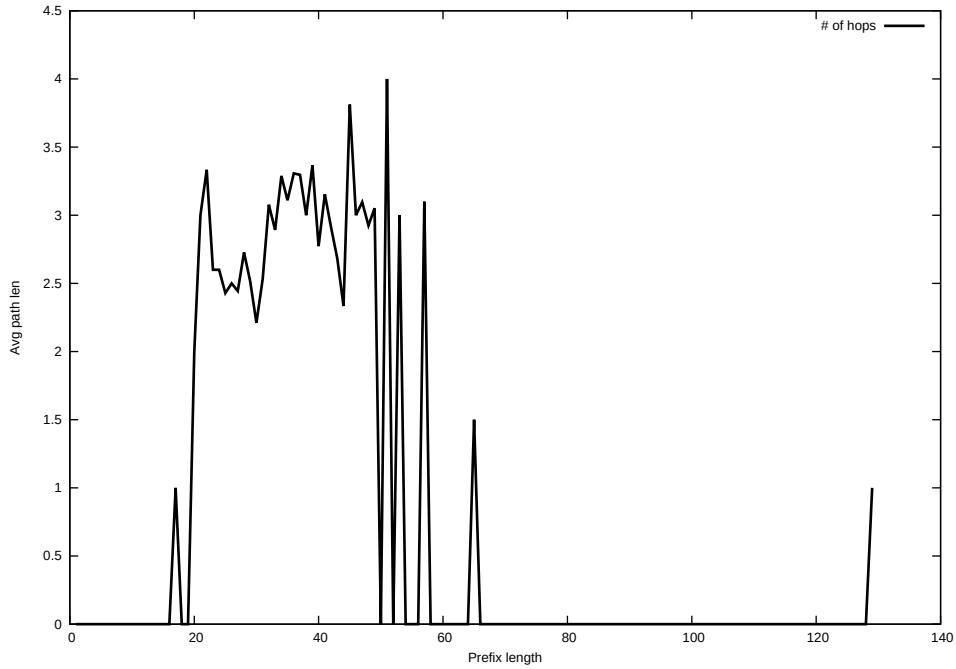
2012-04-10



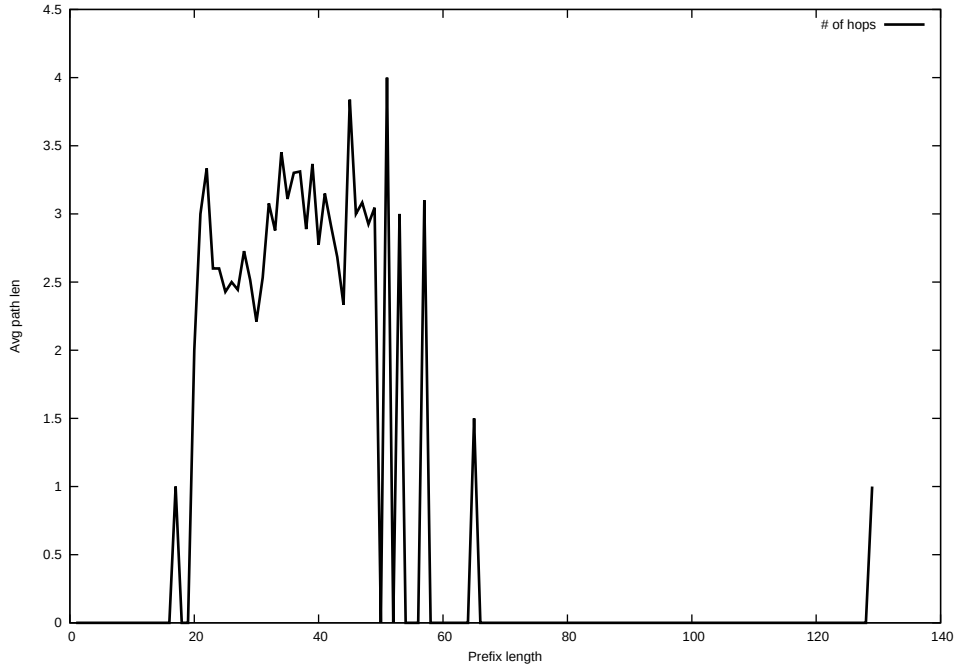
2012-04-11



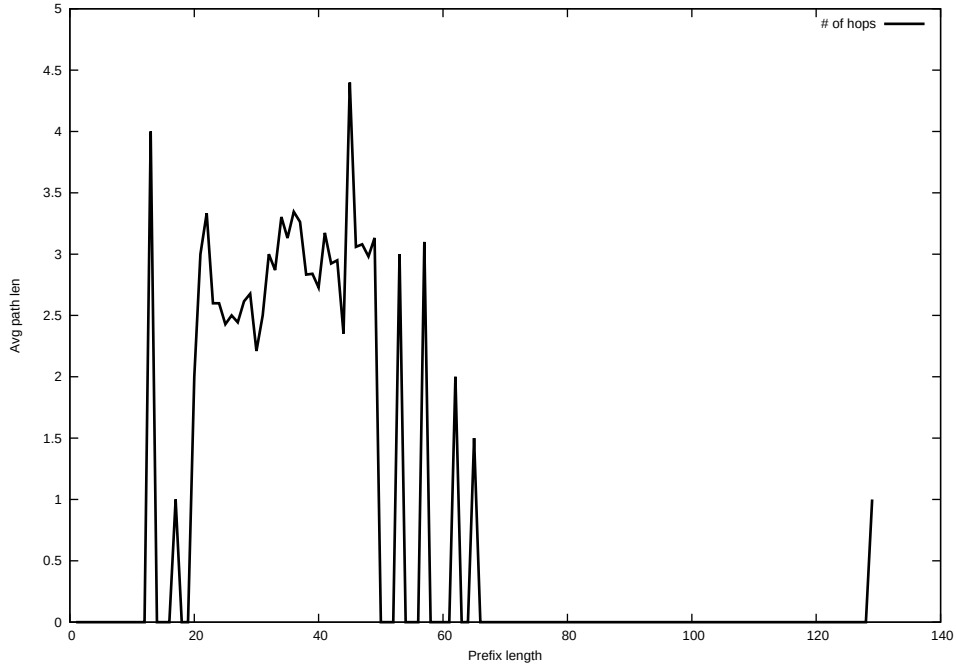
2012-04-12



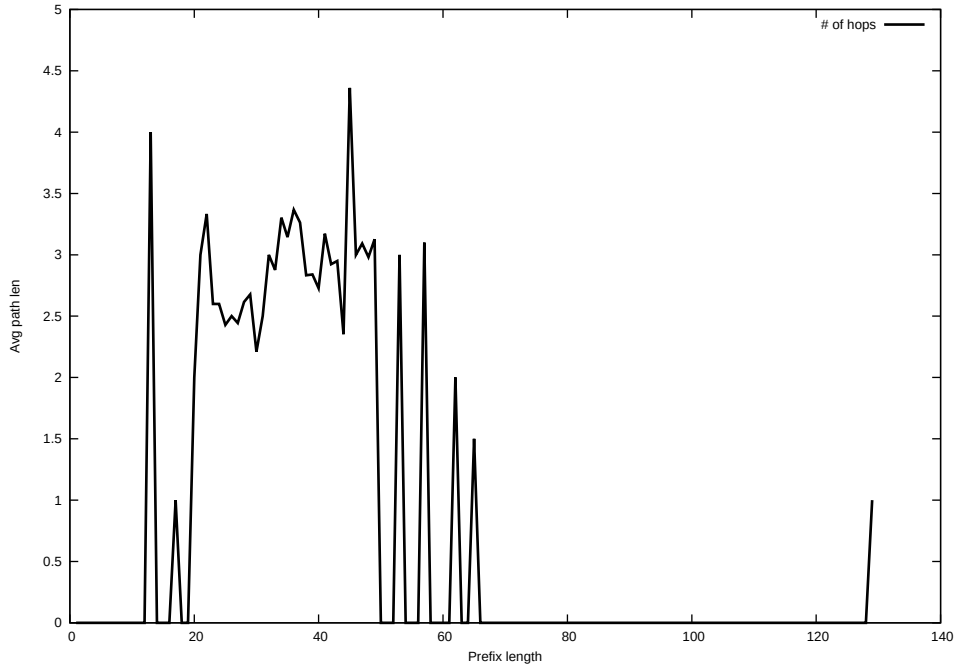
2012-04-13



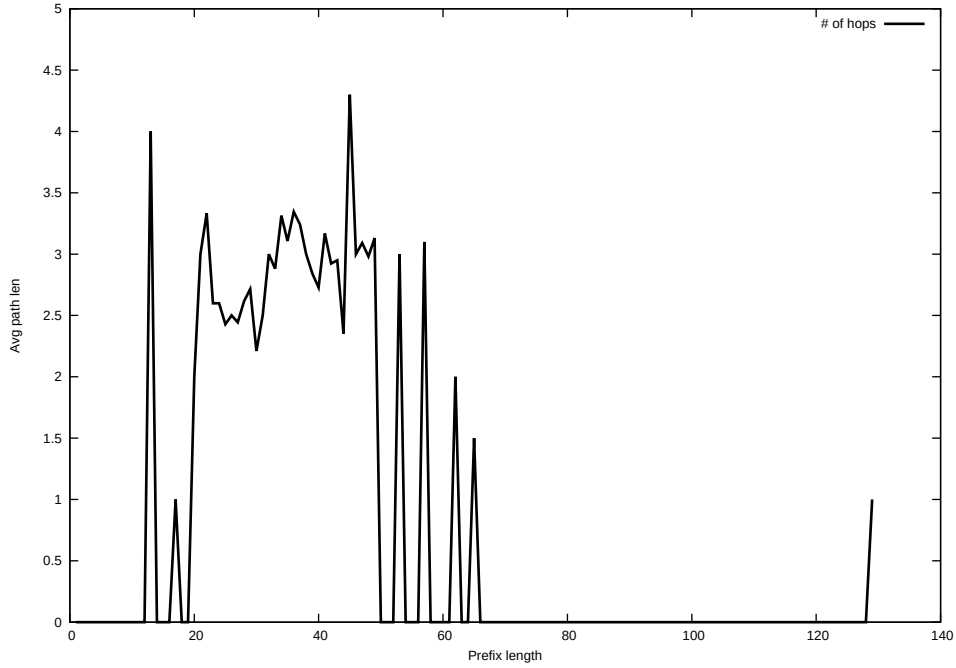
2012-04-14



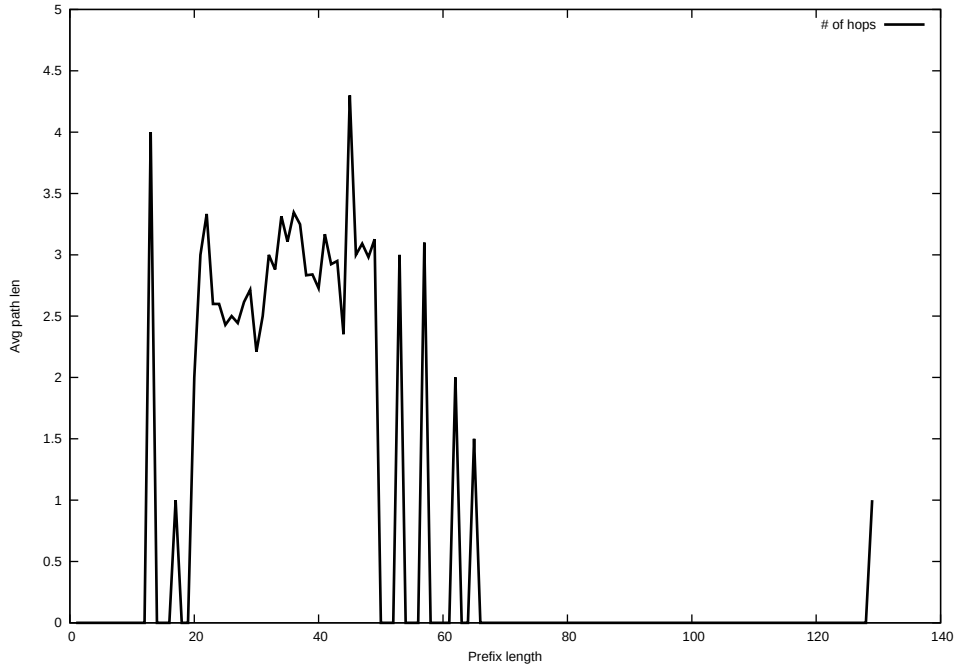
2012-05-16



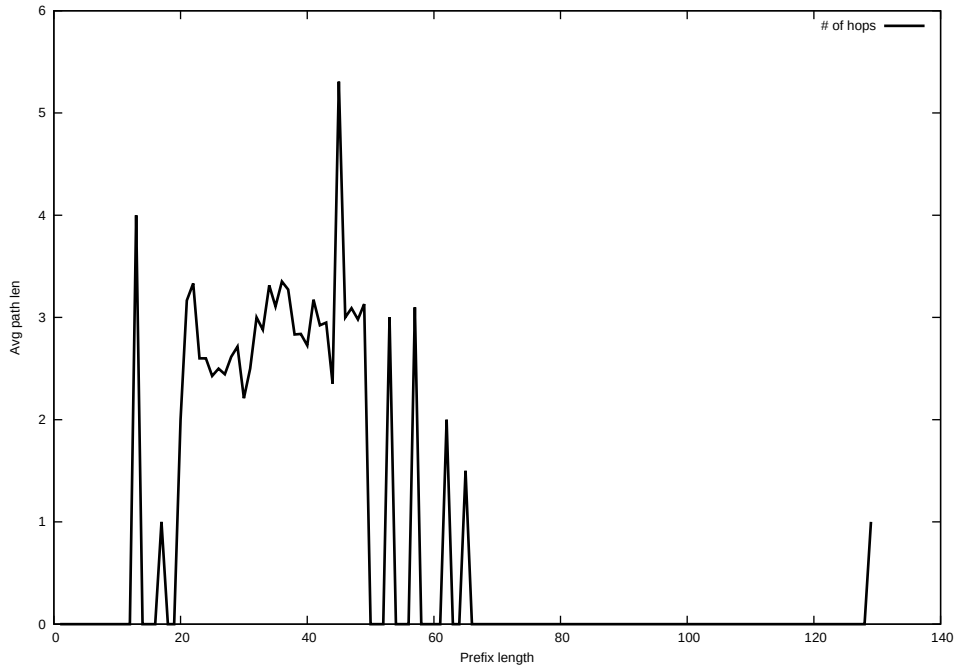
2012-05-17



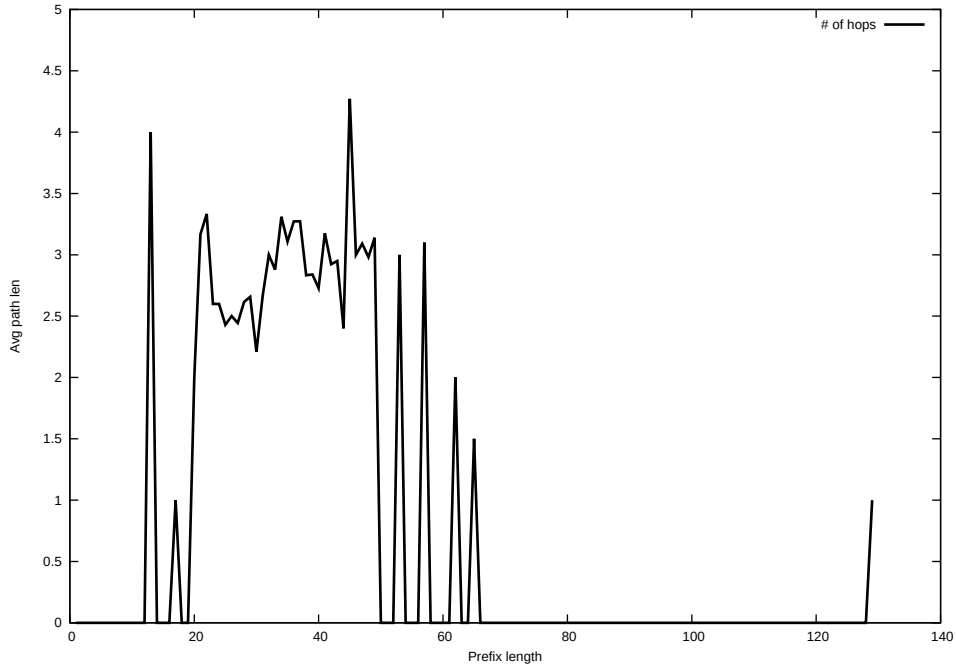
2012-05-19



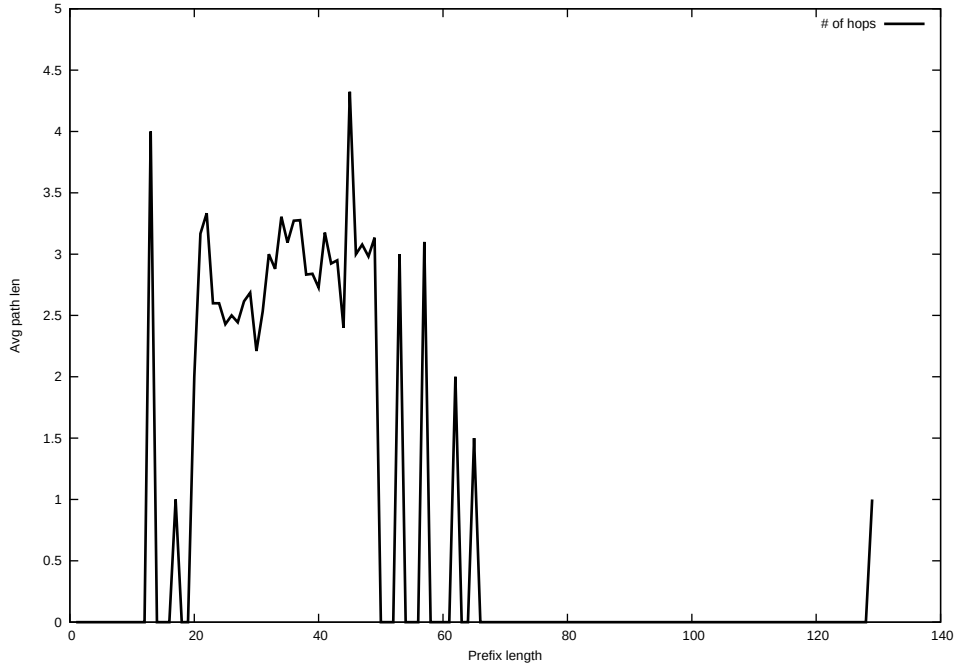
2012-05-20



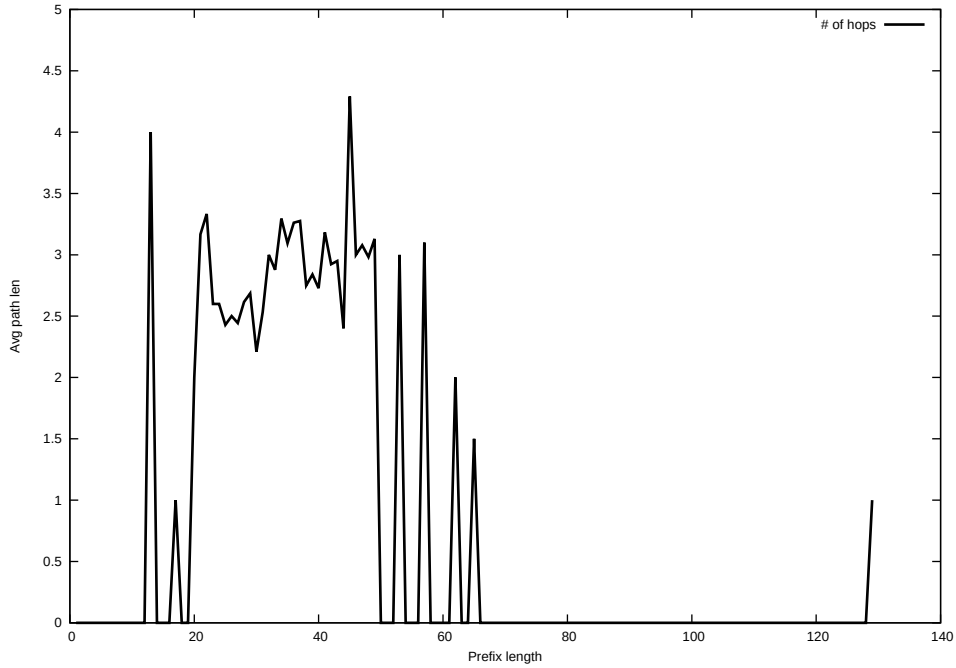
2012-05-21



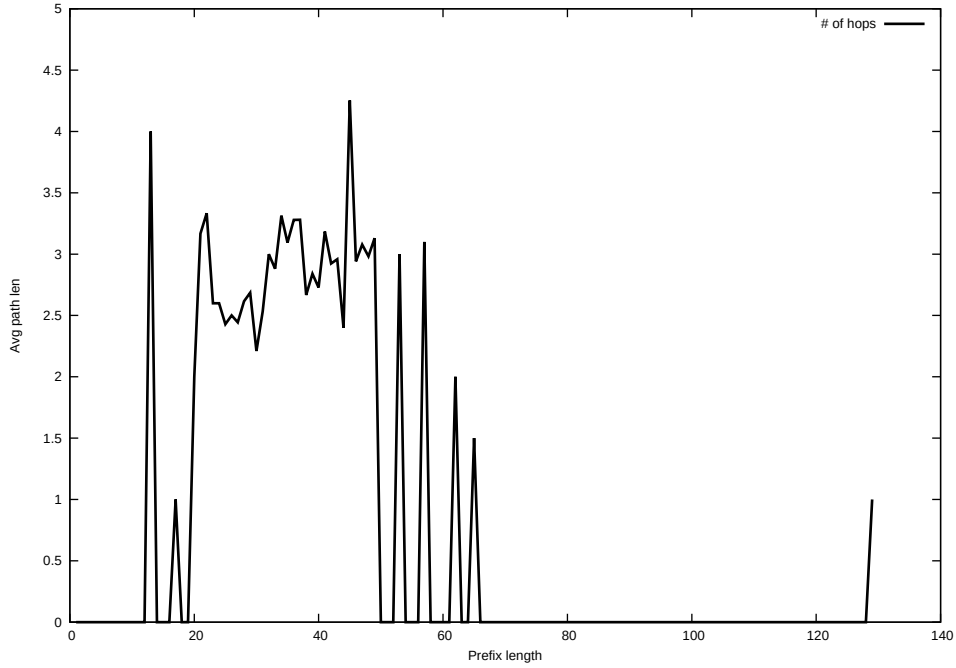
2012-05-22



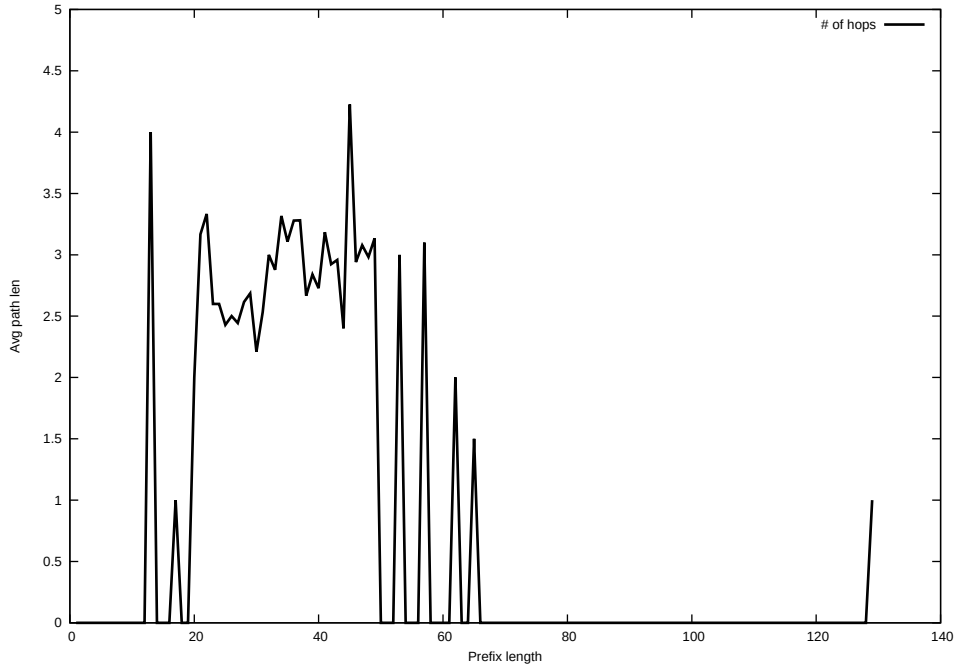
2012-05-23



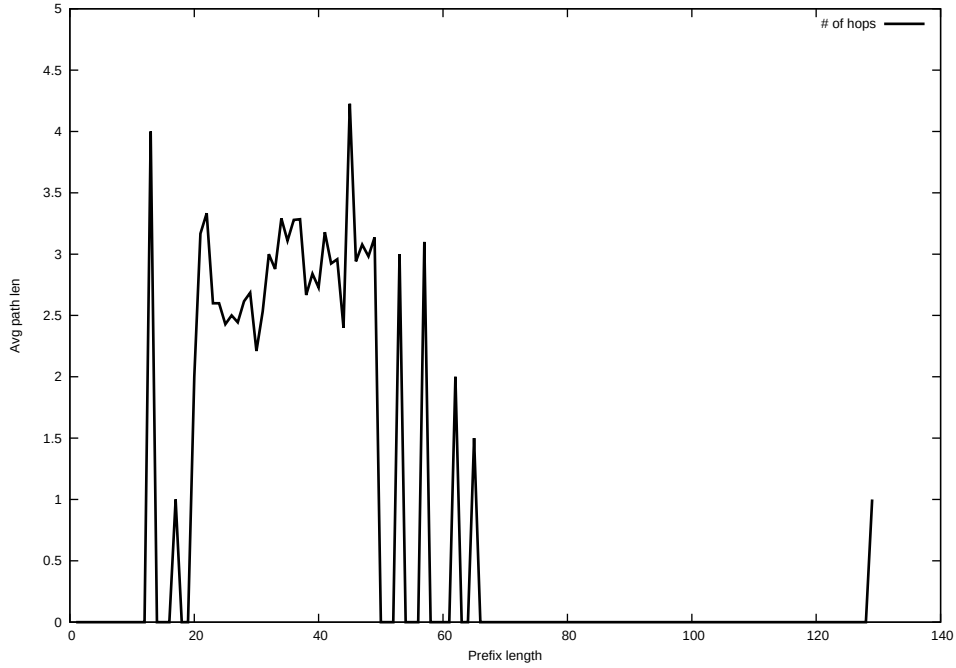
2012-05-24



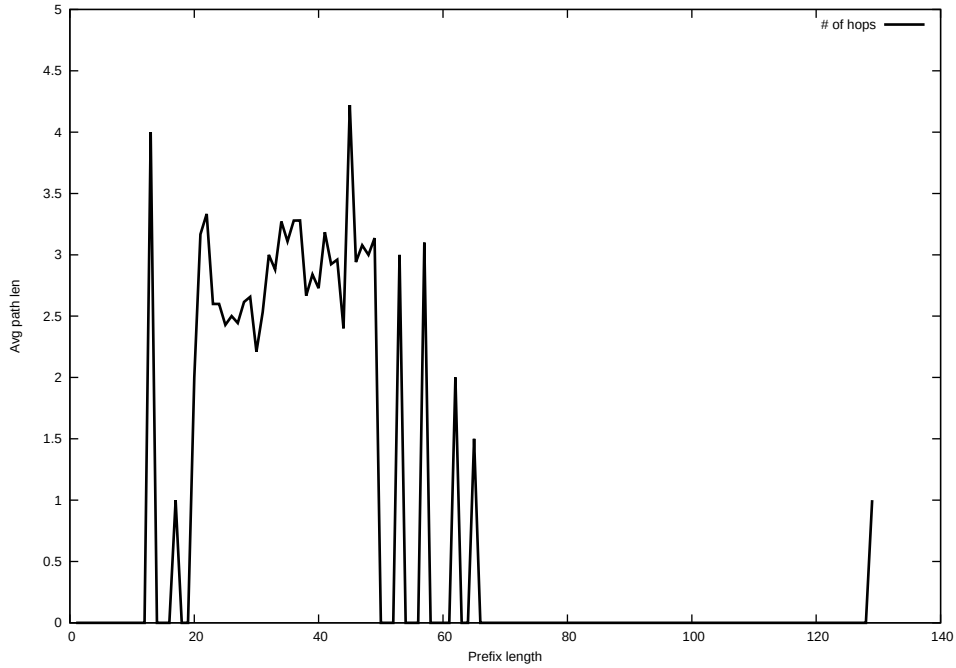
2012-05-25



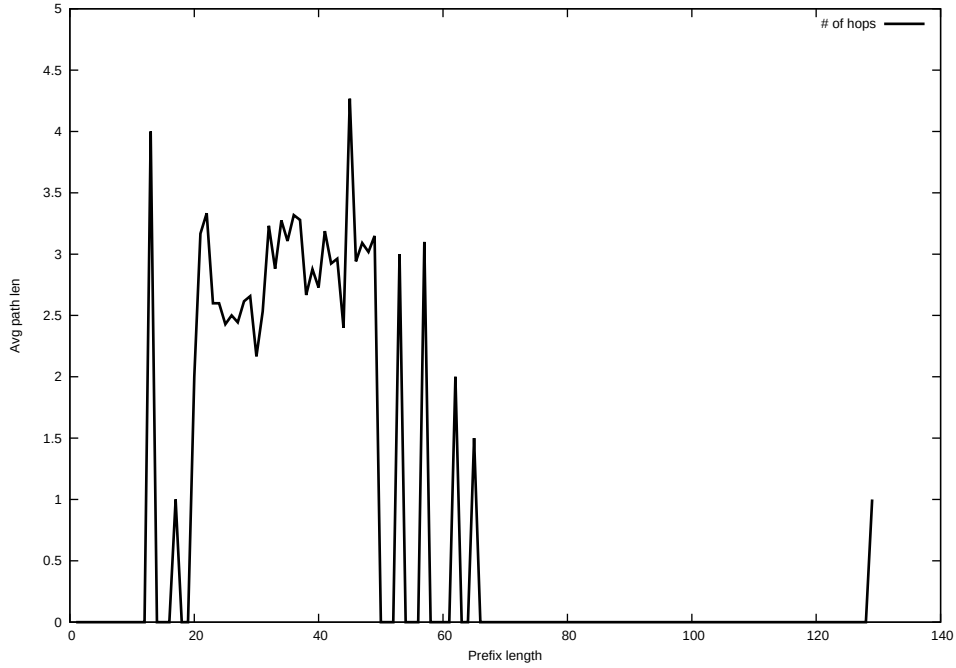
2012-05-26



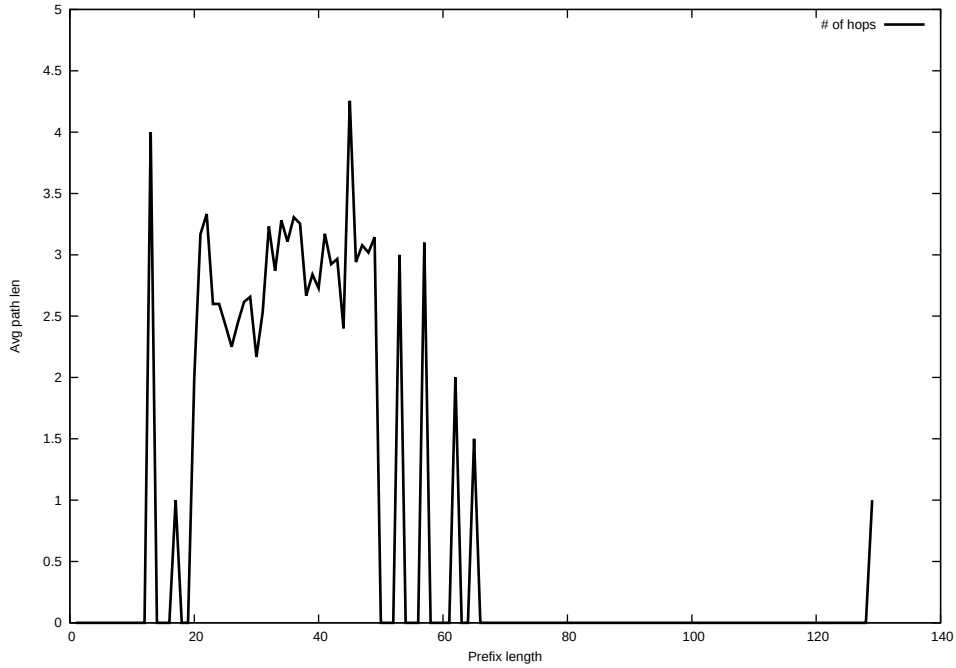
2012-05-27



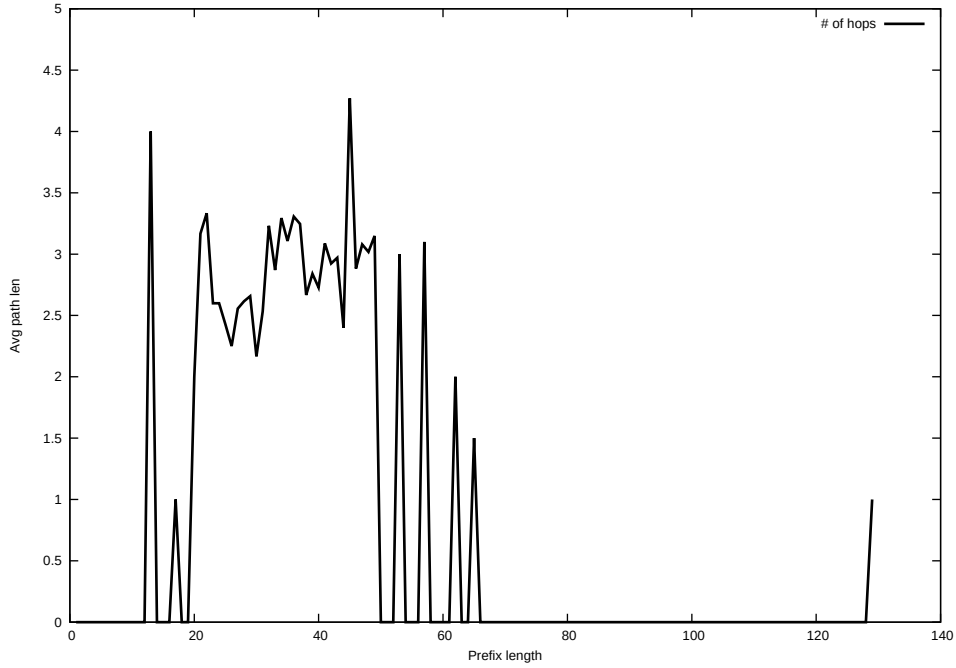
2012-05-28



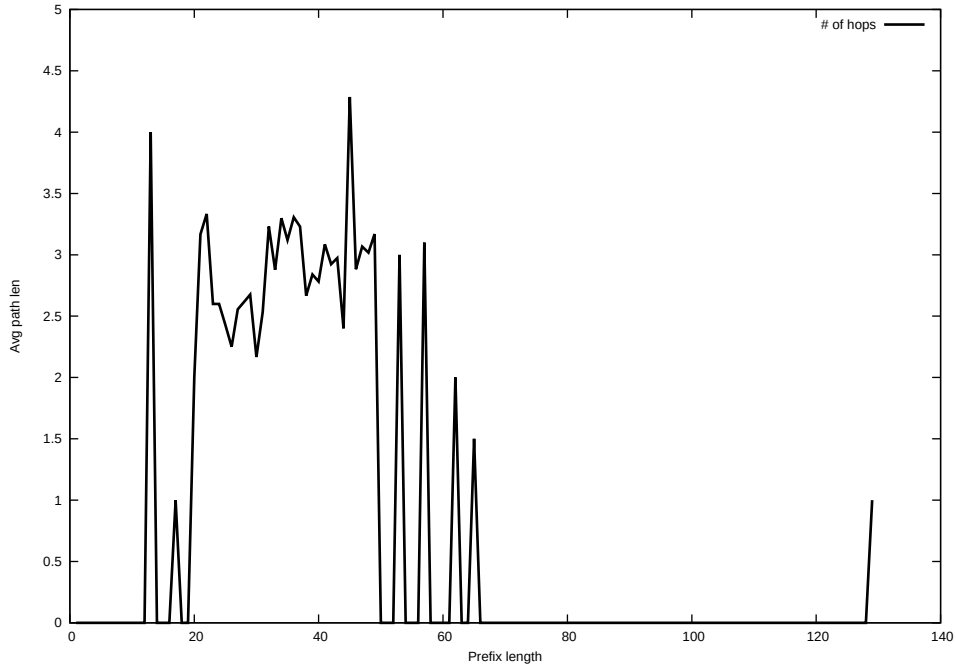
2012-05-29



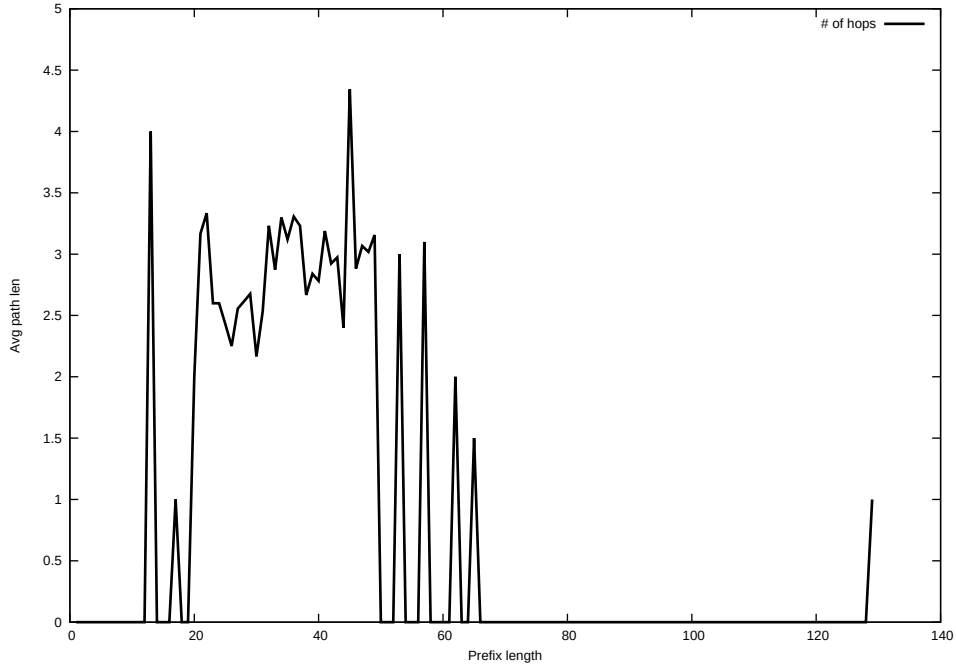
2012-05-30



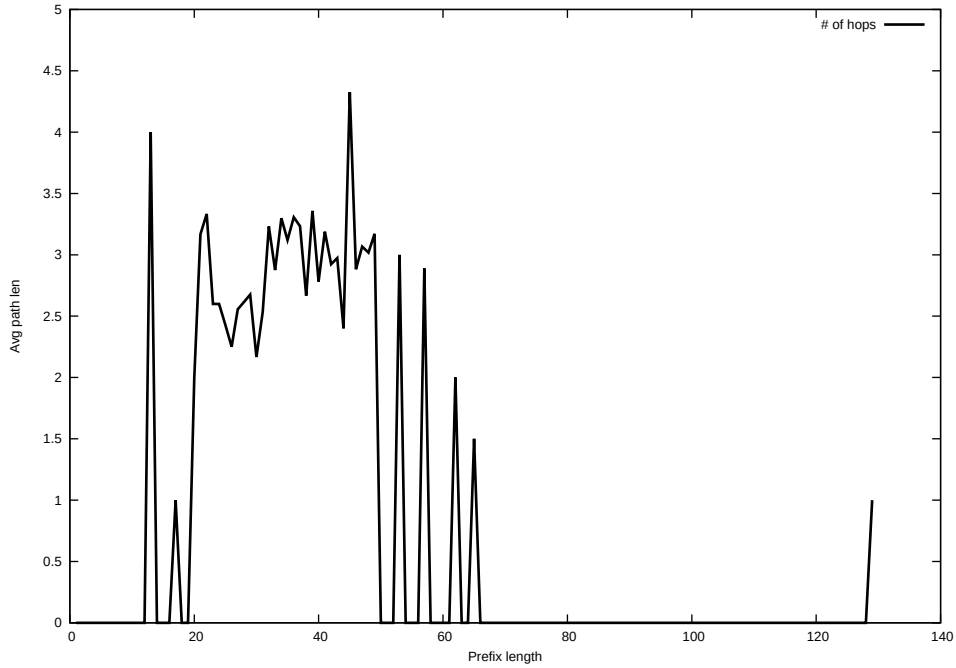
2012-05-31



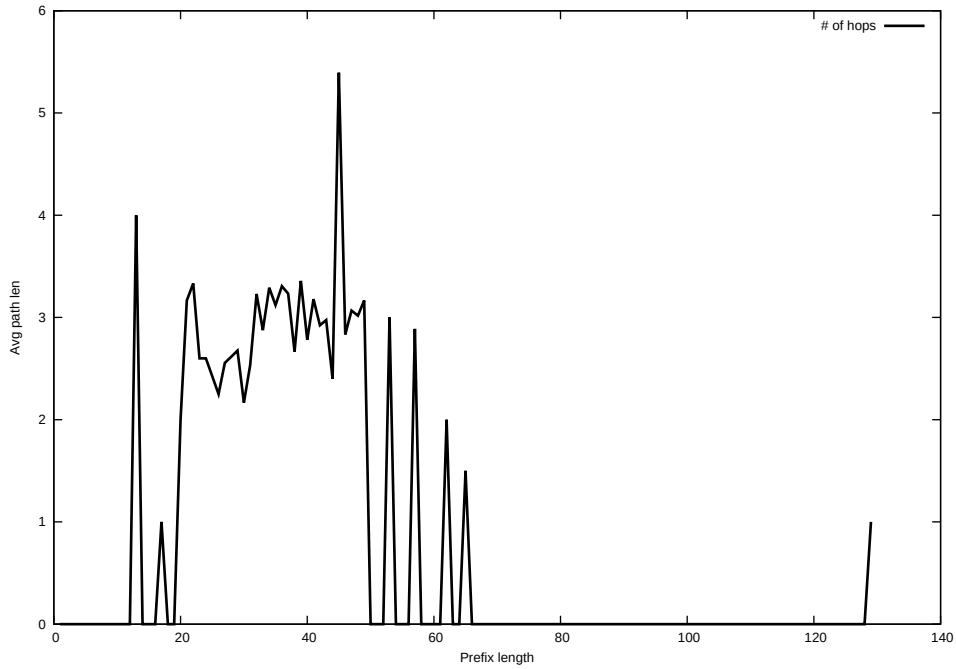
2012-06-01



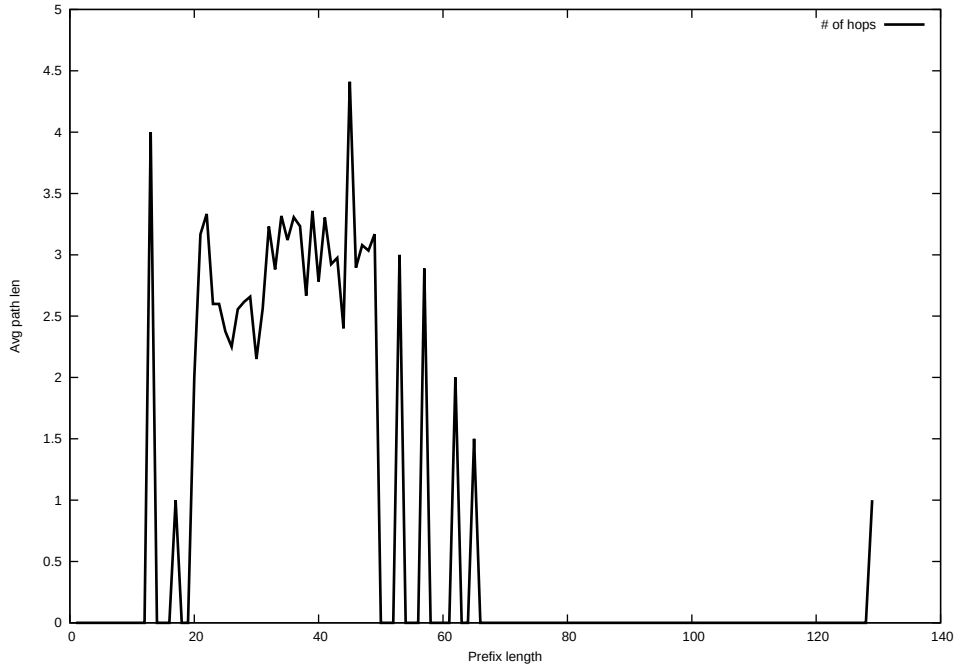
2012-06-02



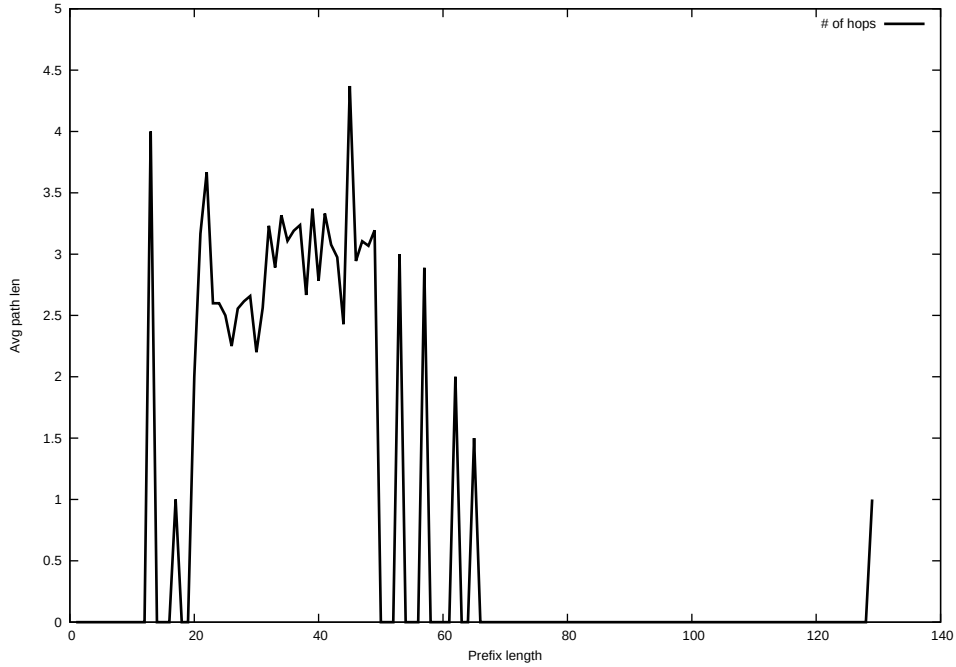
2012-06-03



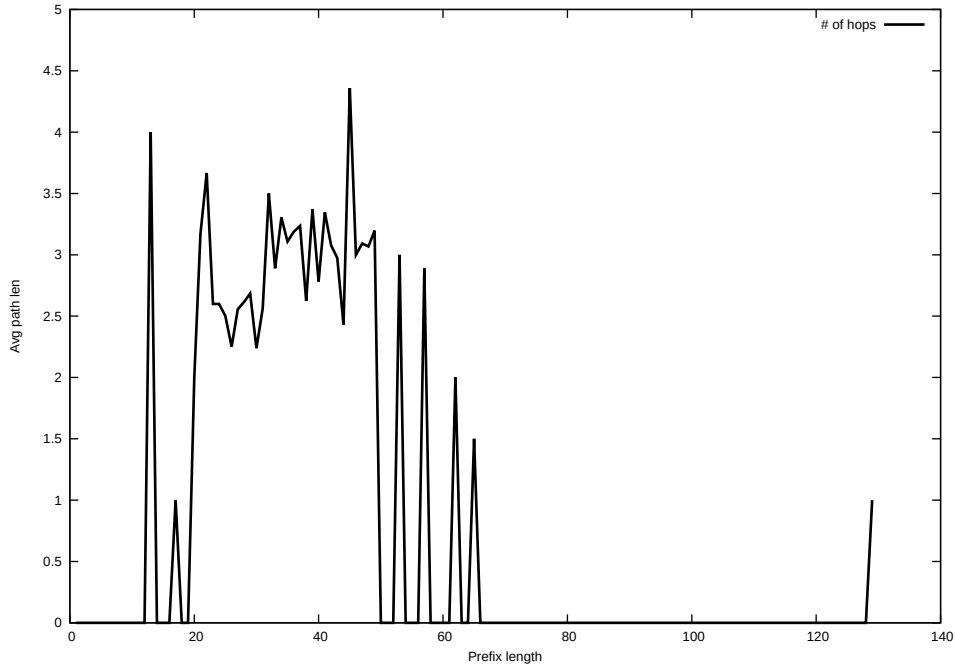
2012-06-04



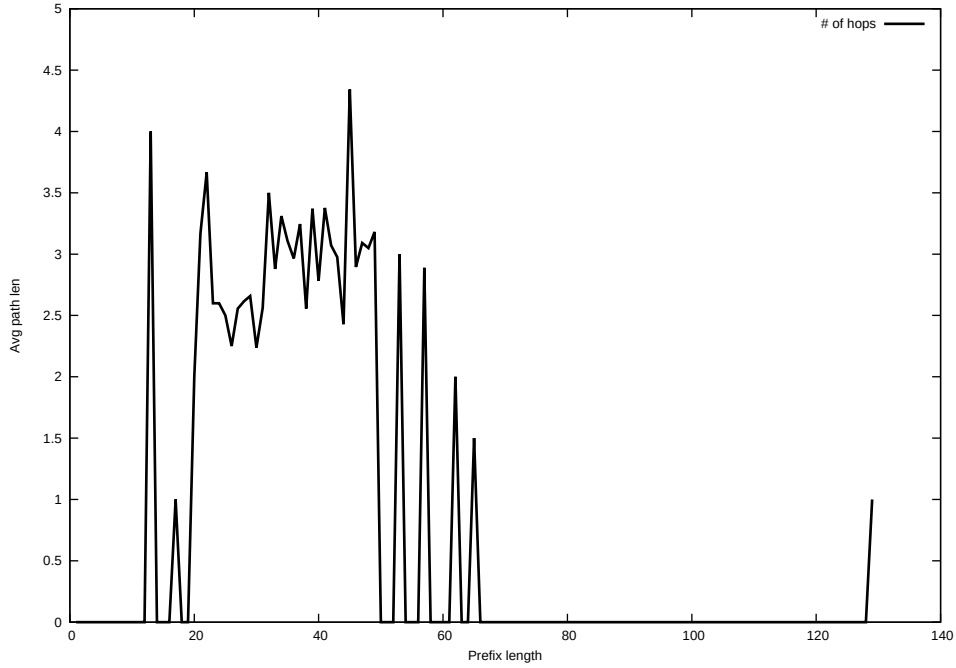
2012-06-05



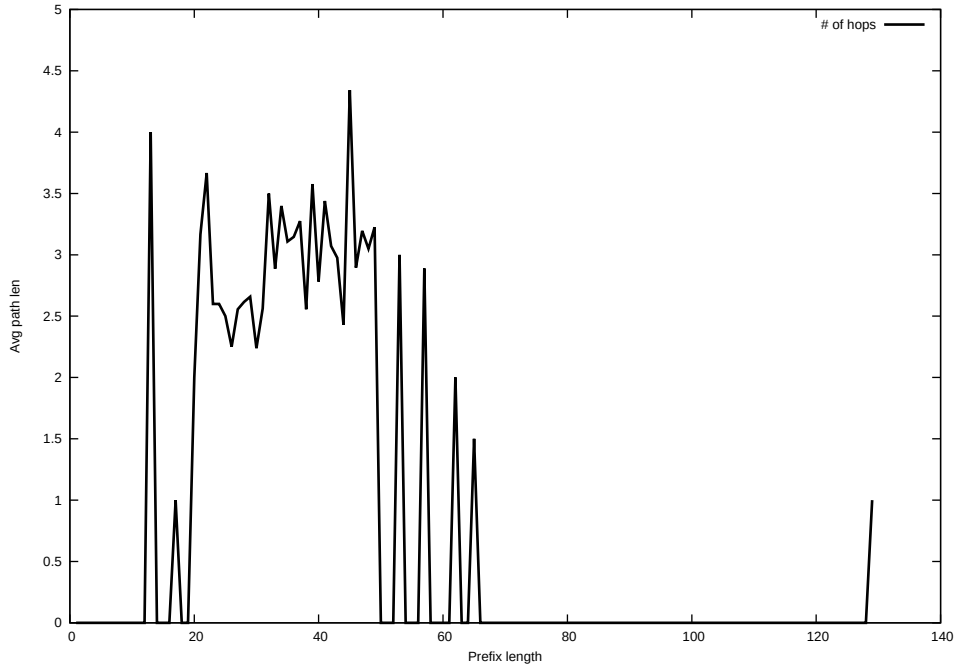
2012-06-06



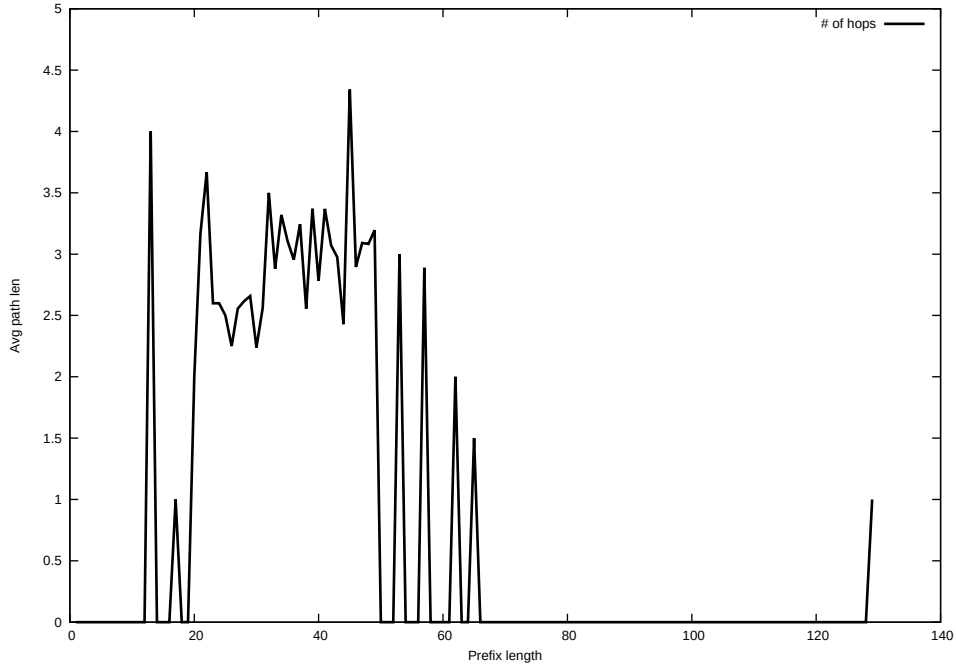
2012-06-07



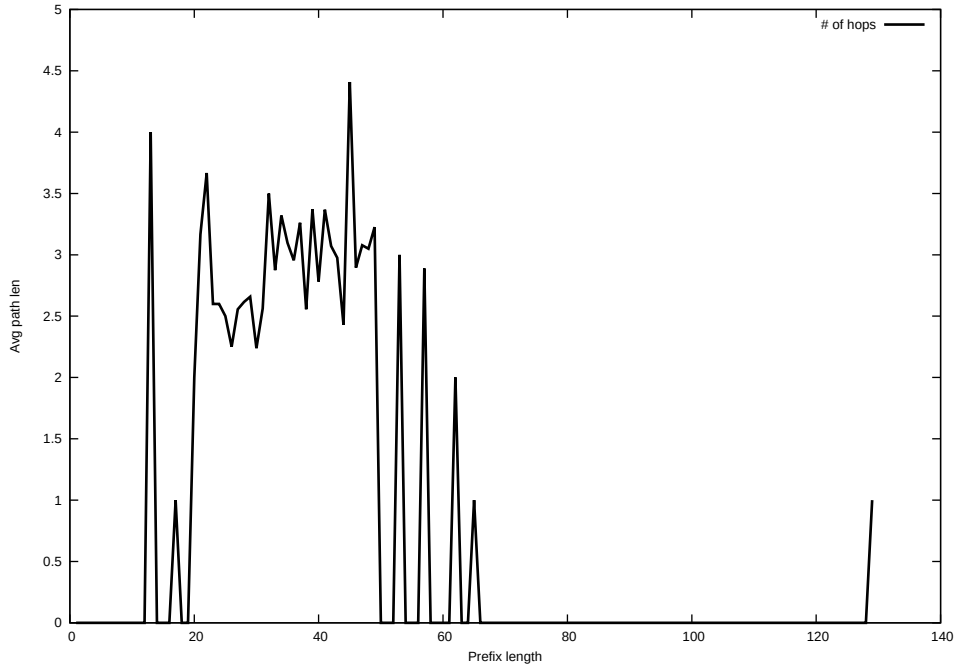
2012-06-08



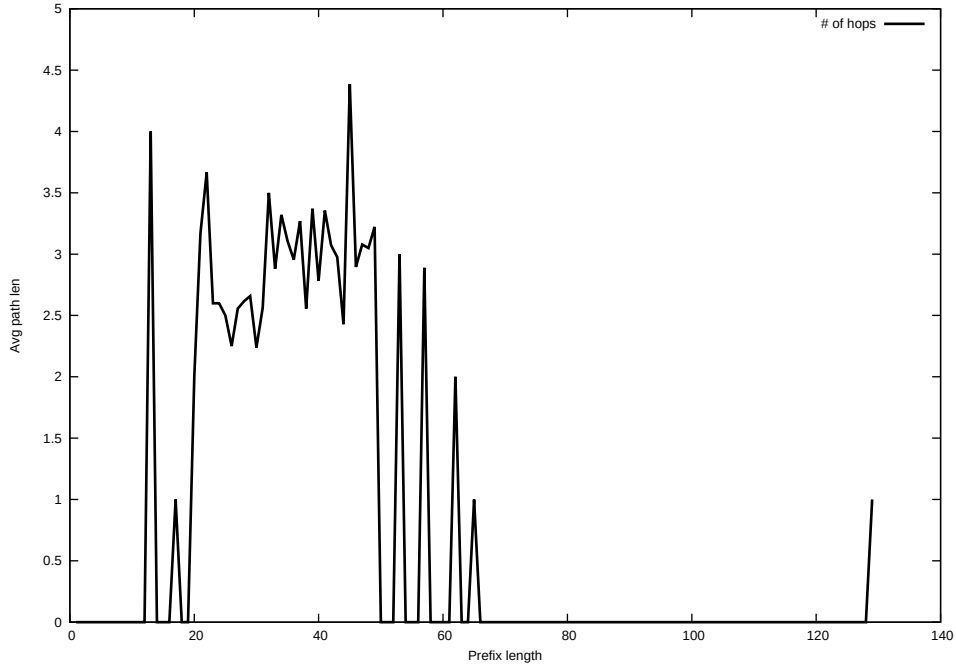
2012-06-10



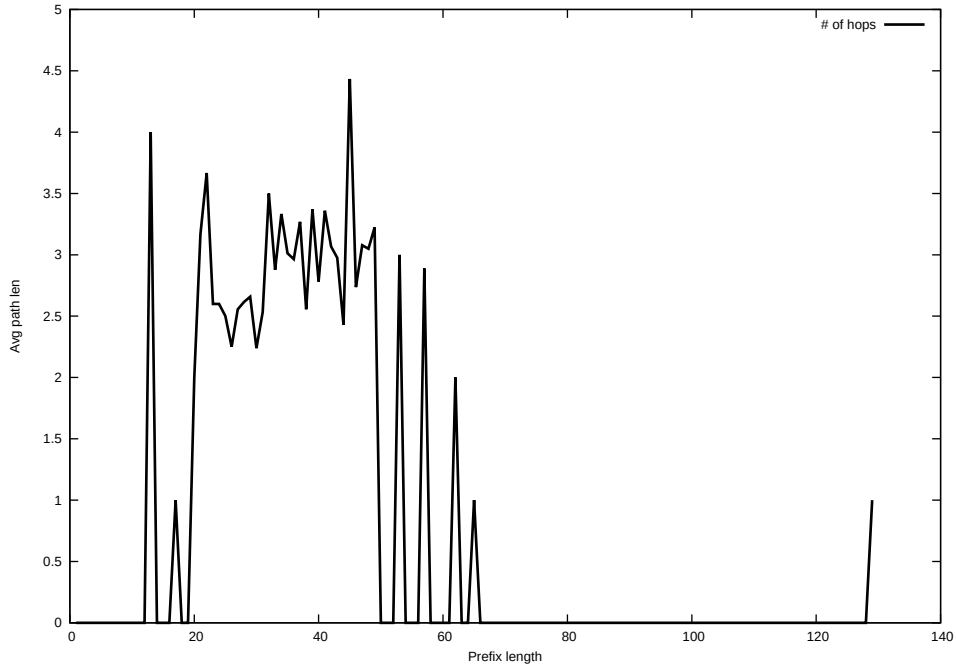
2012-06-11



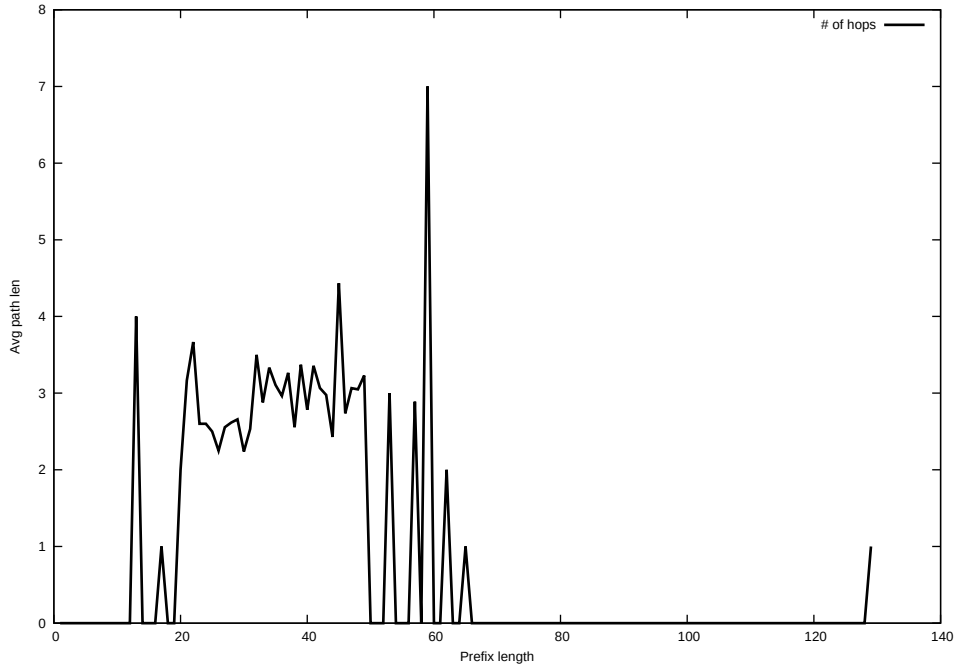
2012-06-12



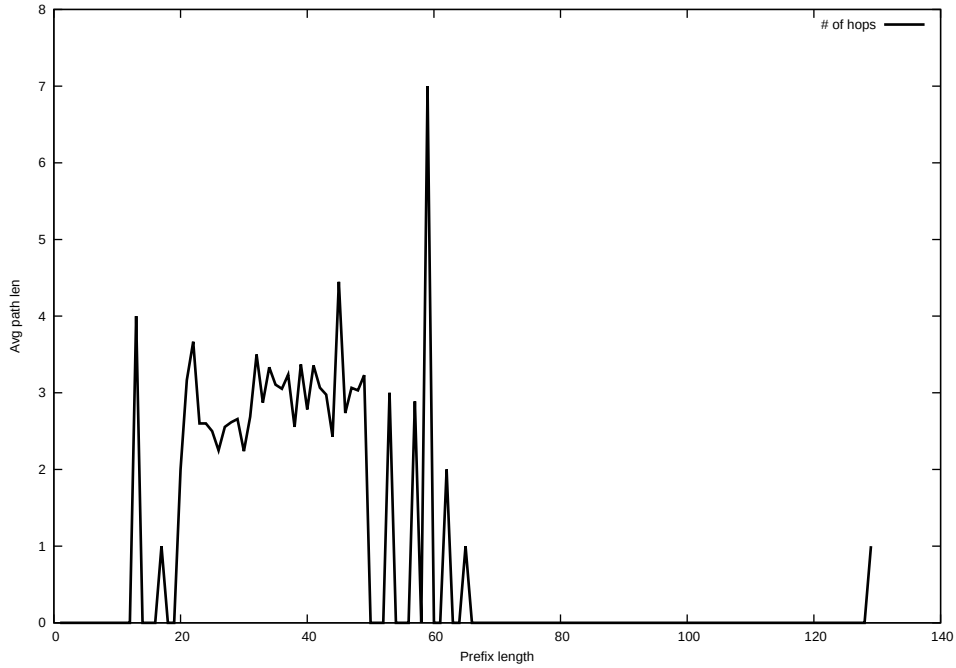
2012-06-13



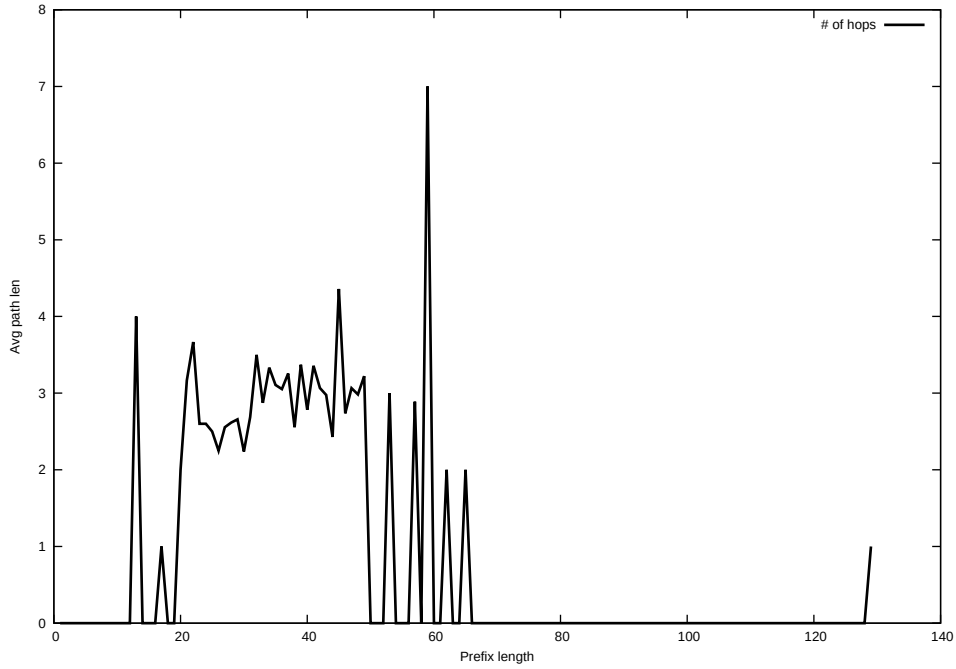
2012-06-14



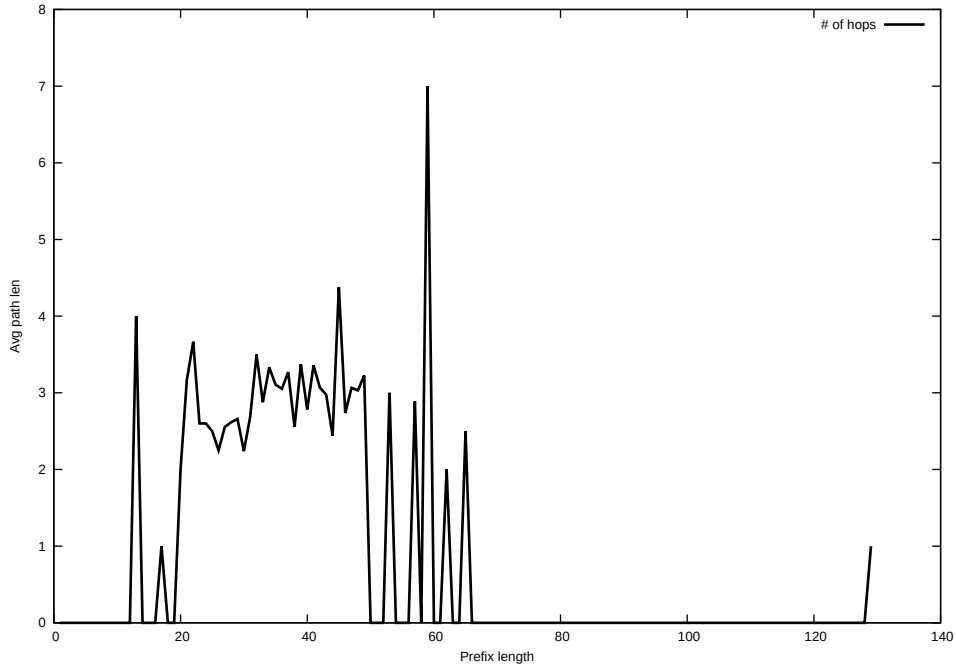
2012-06-15



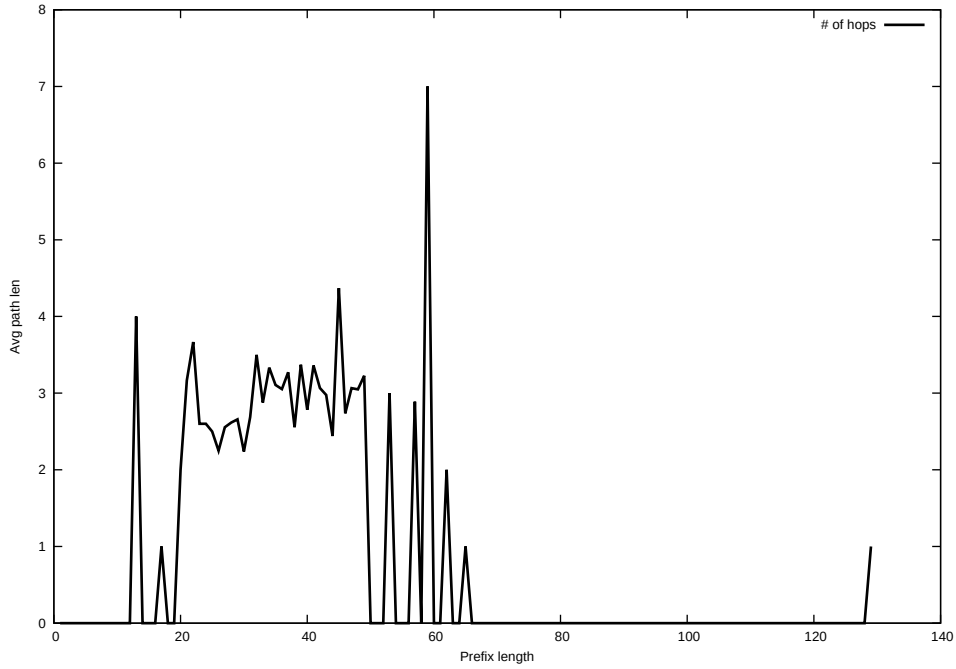
2012-06-16



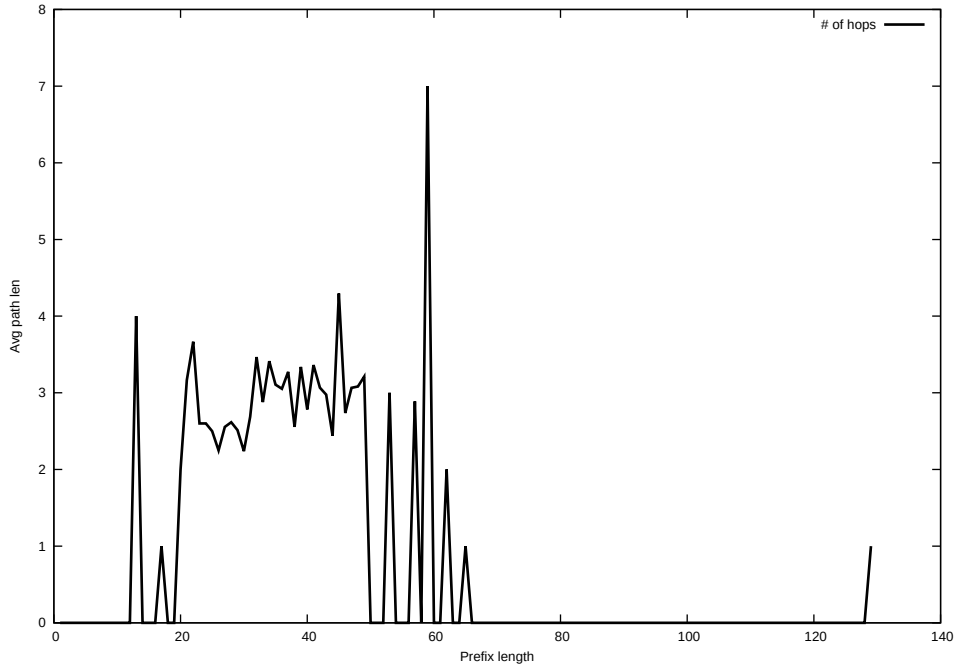
2012-06-17



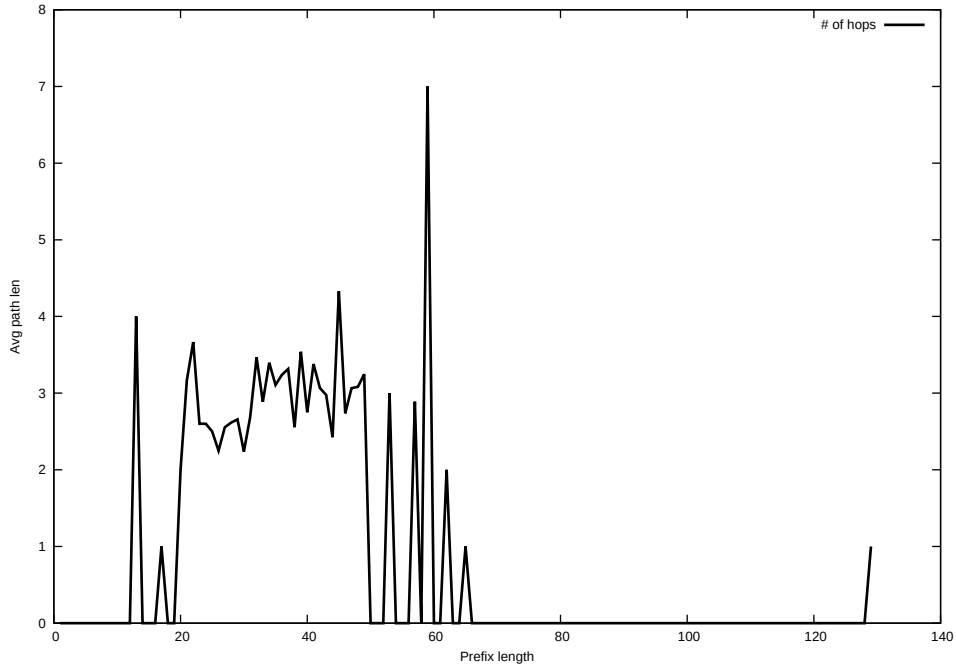
2012-06-18



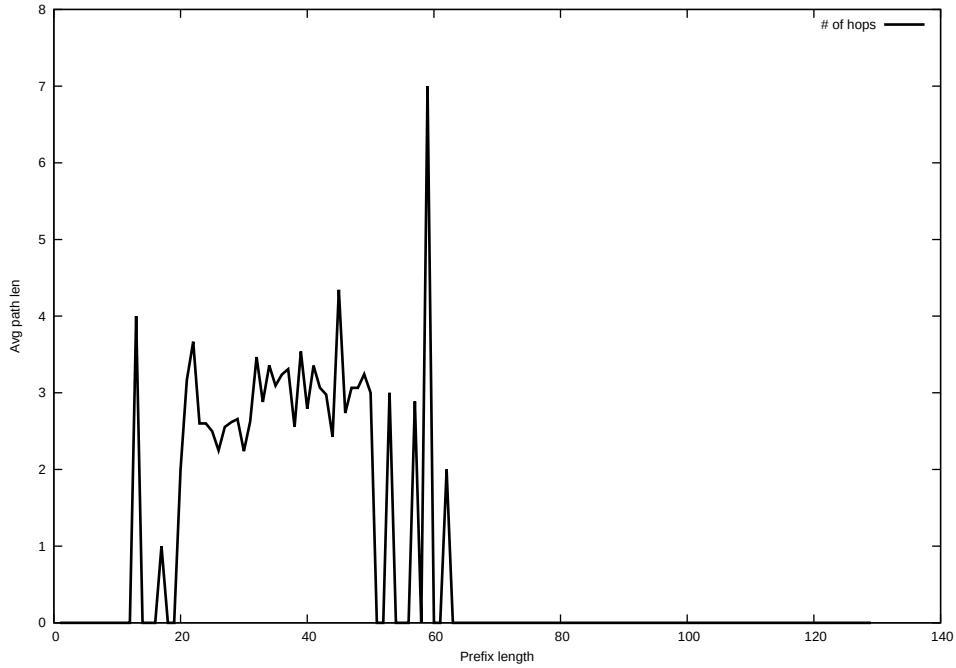
2012-06-19



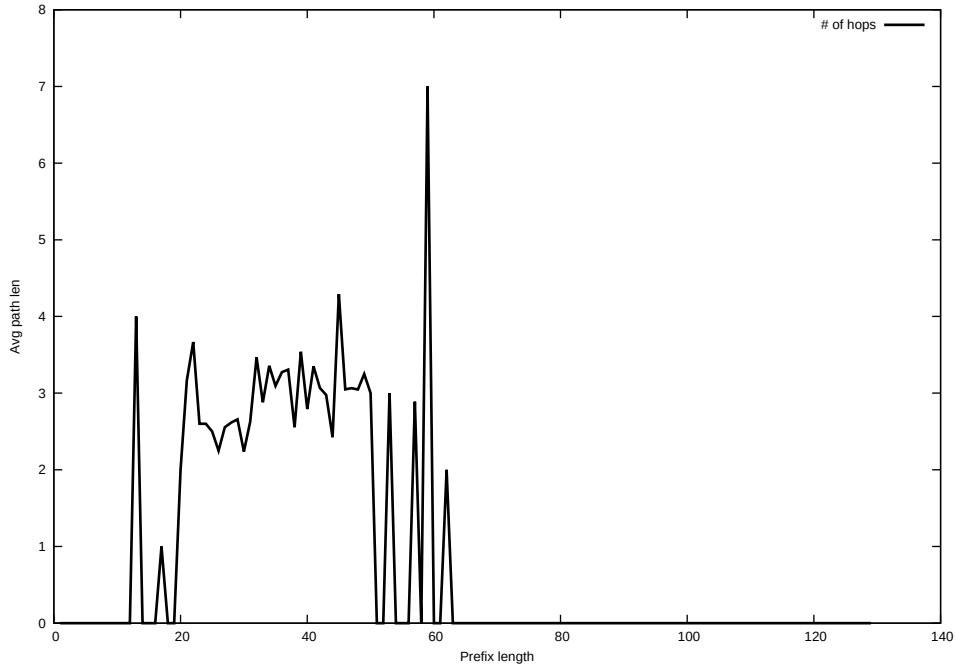
2012-06-20



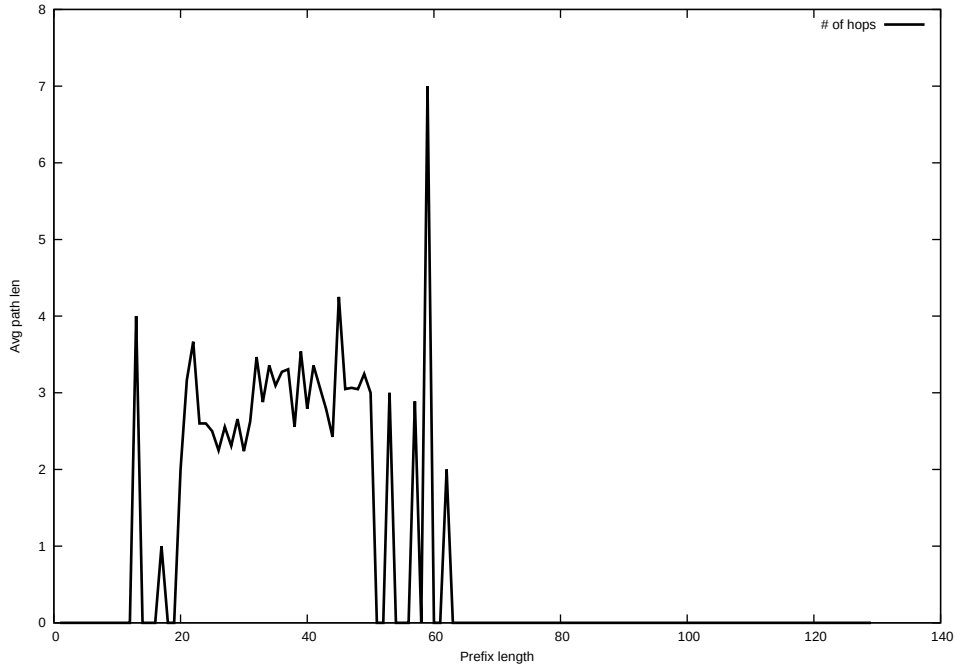
2012-06-21



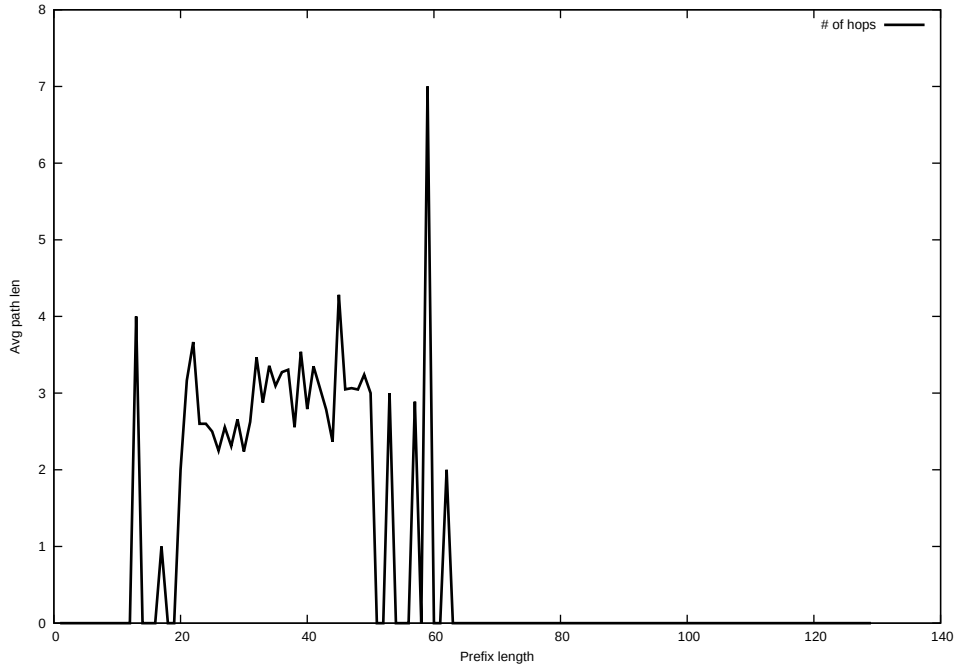
2012-06-22



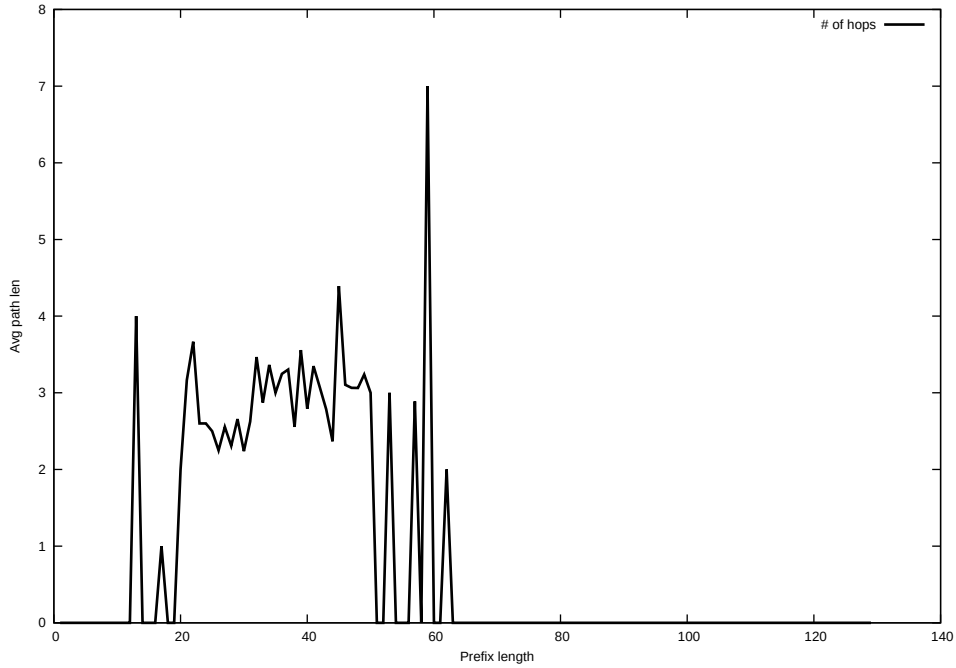
2012-06-23



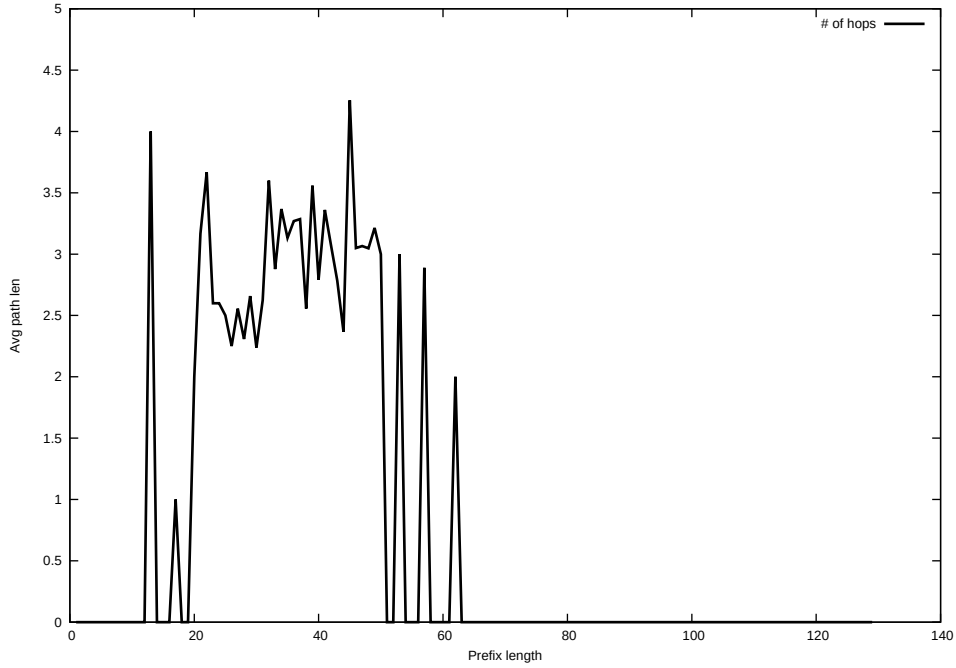
2012-06-24



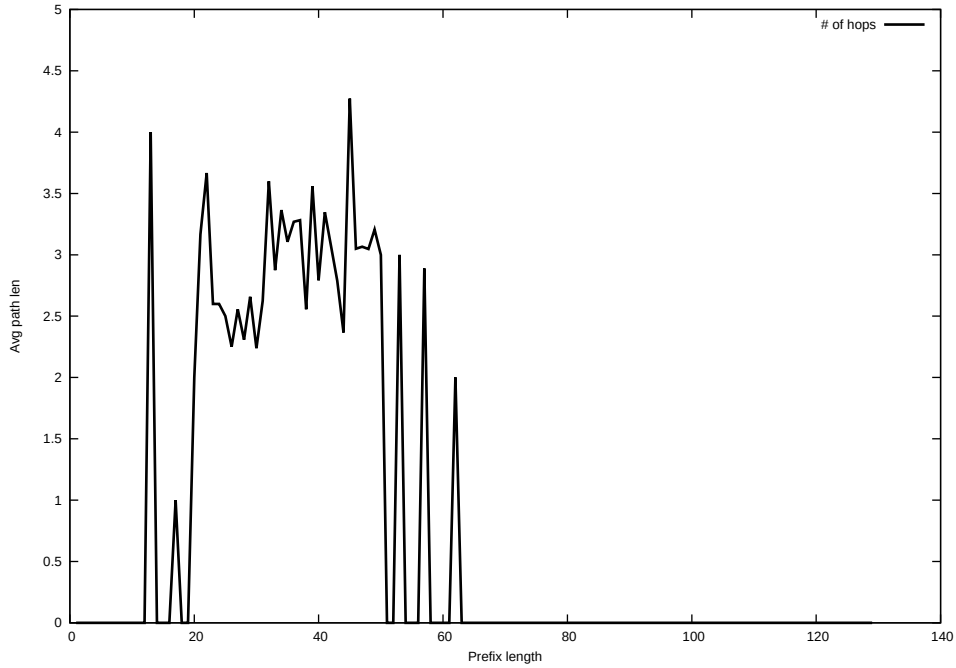
2012-06-25



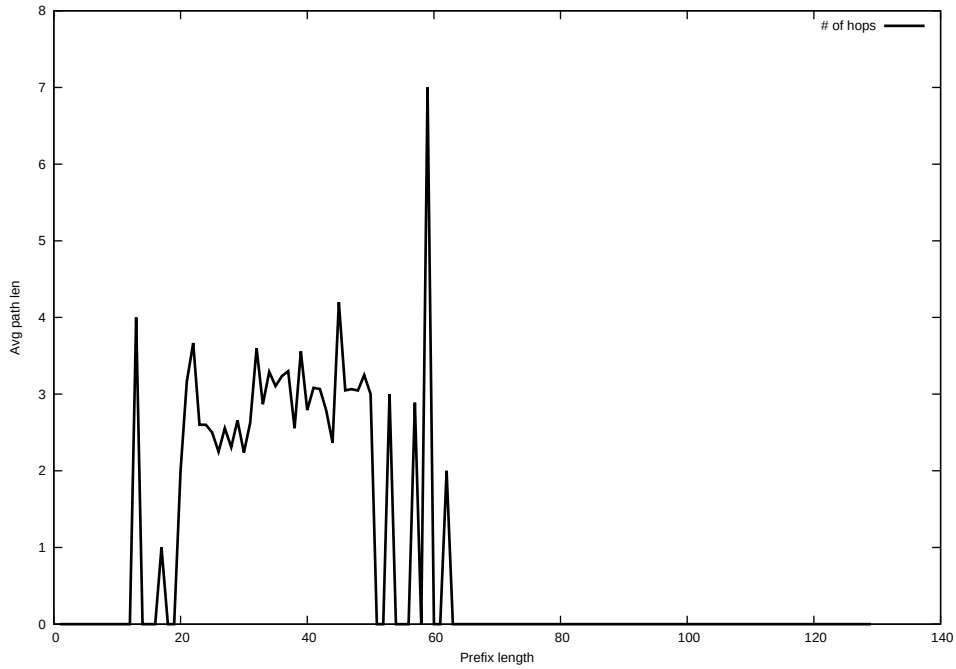
2012-06-26



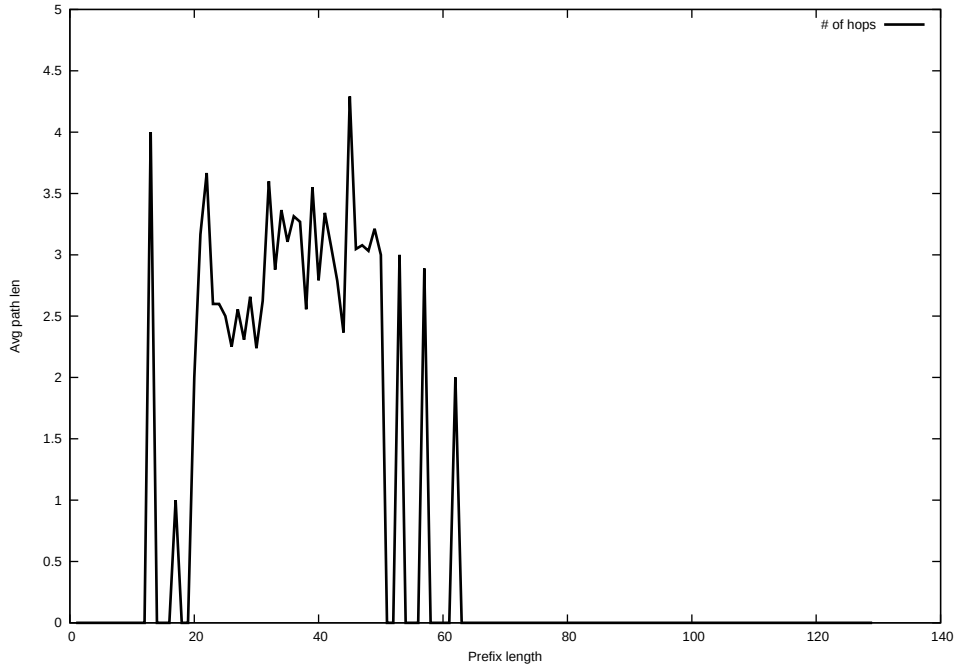
2012-06-27



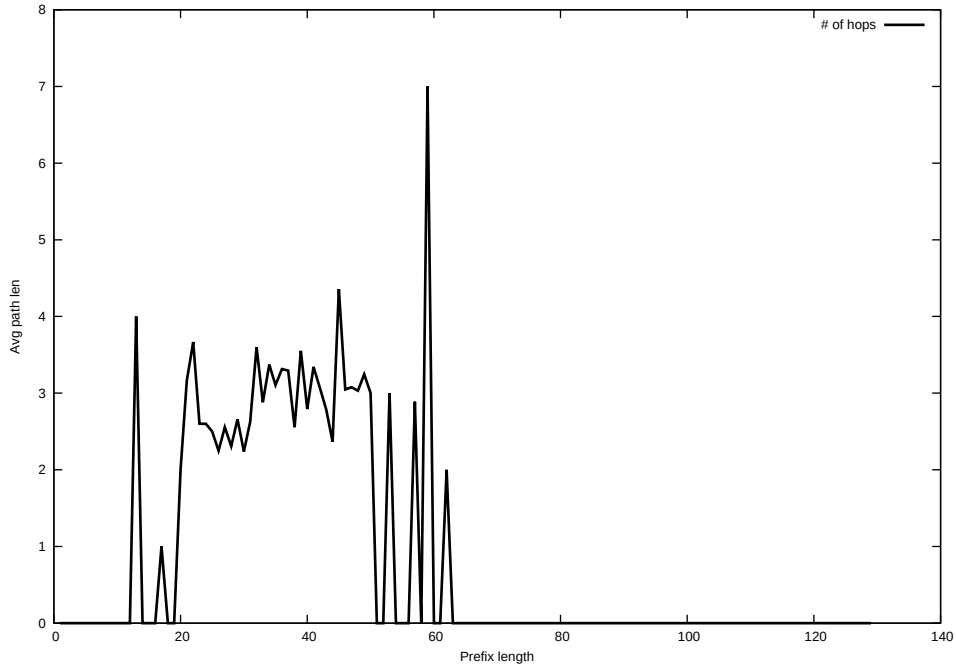
2012-06-28



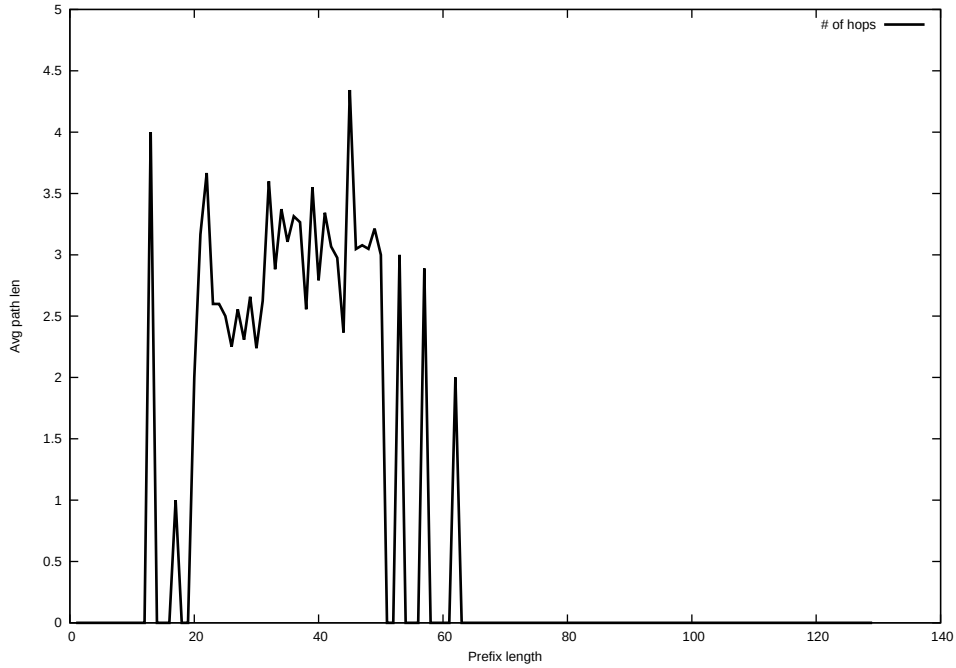
2012-06-29



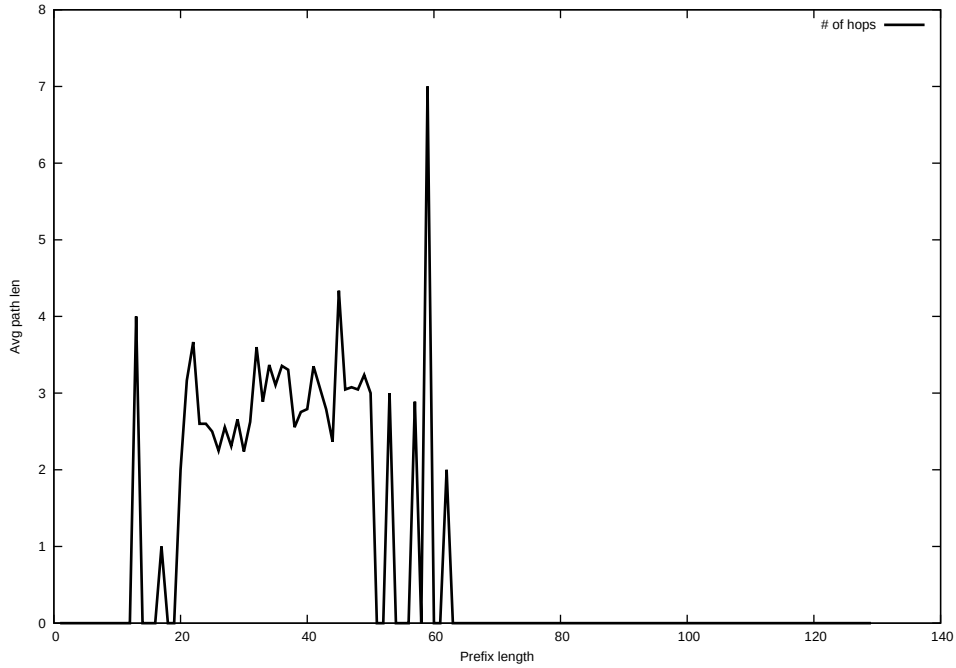
2012-06-30



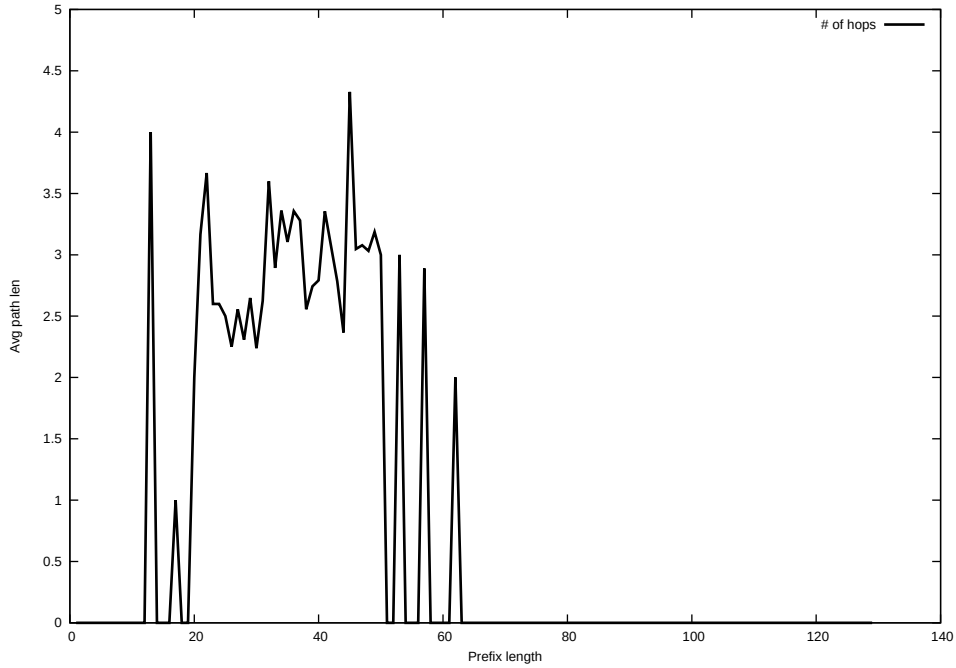
2012-07-01



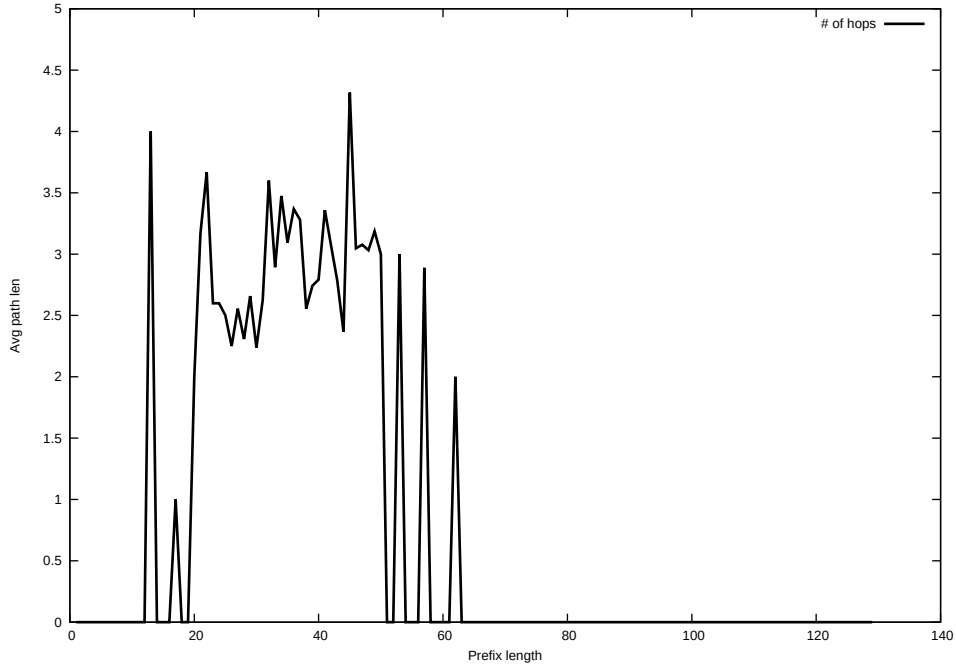
2012-07-02



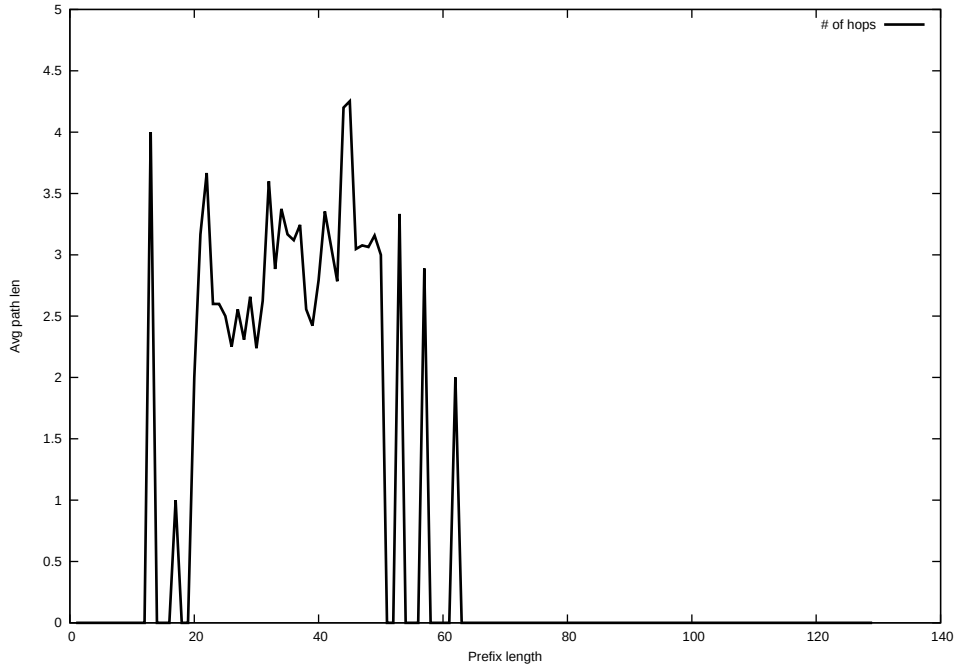
2012-07-03



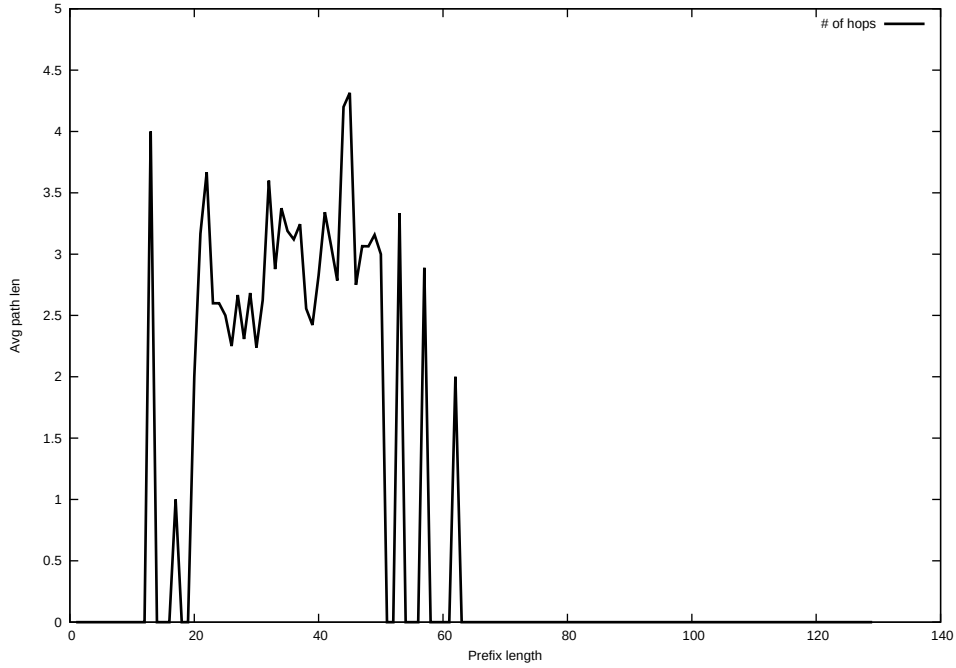
2012-07-04



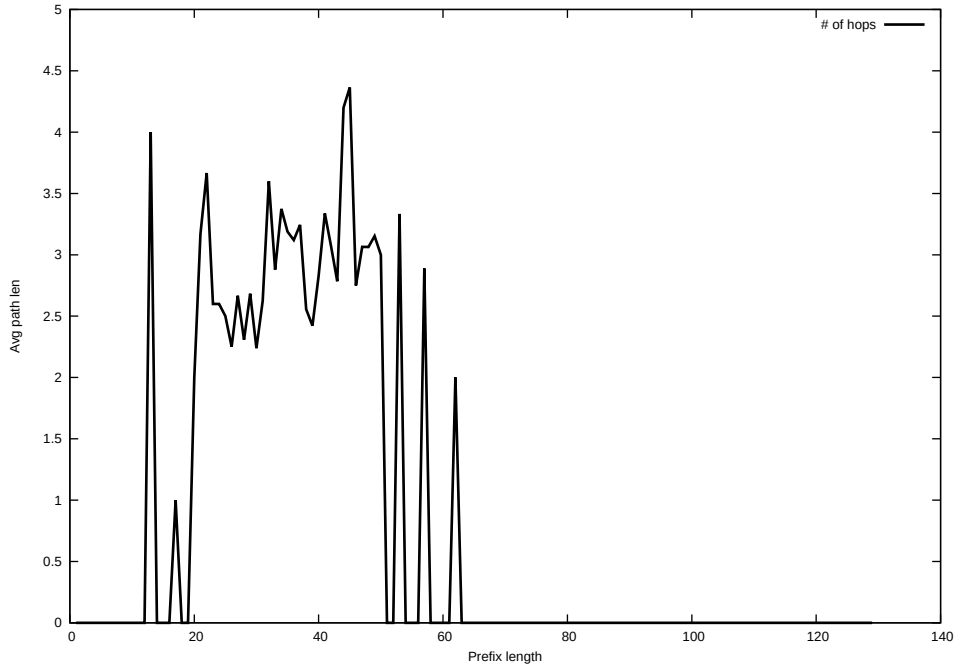
2012-07-05



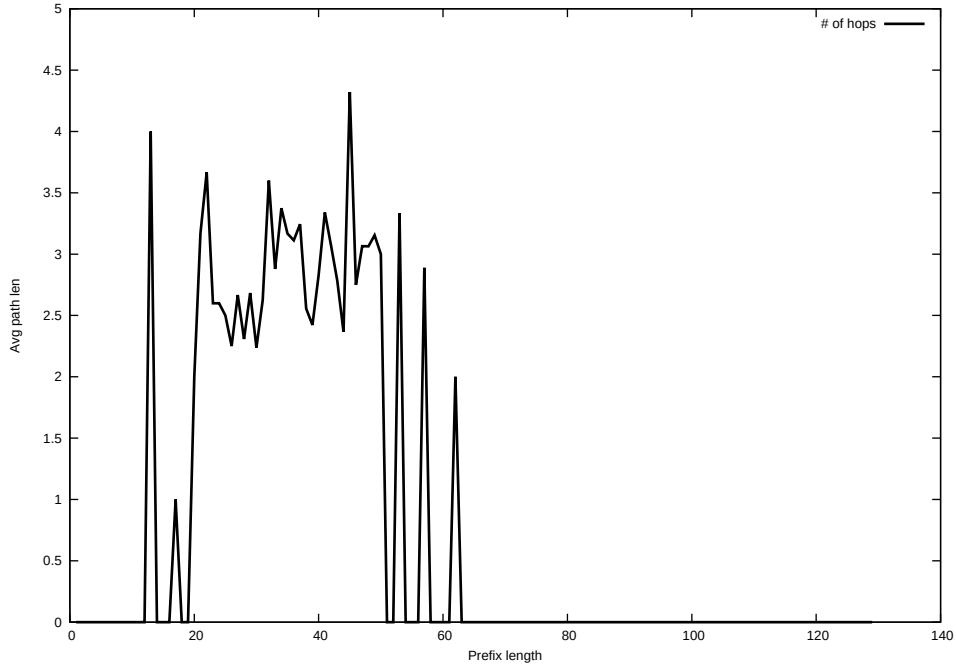
2012-07-06



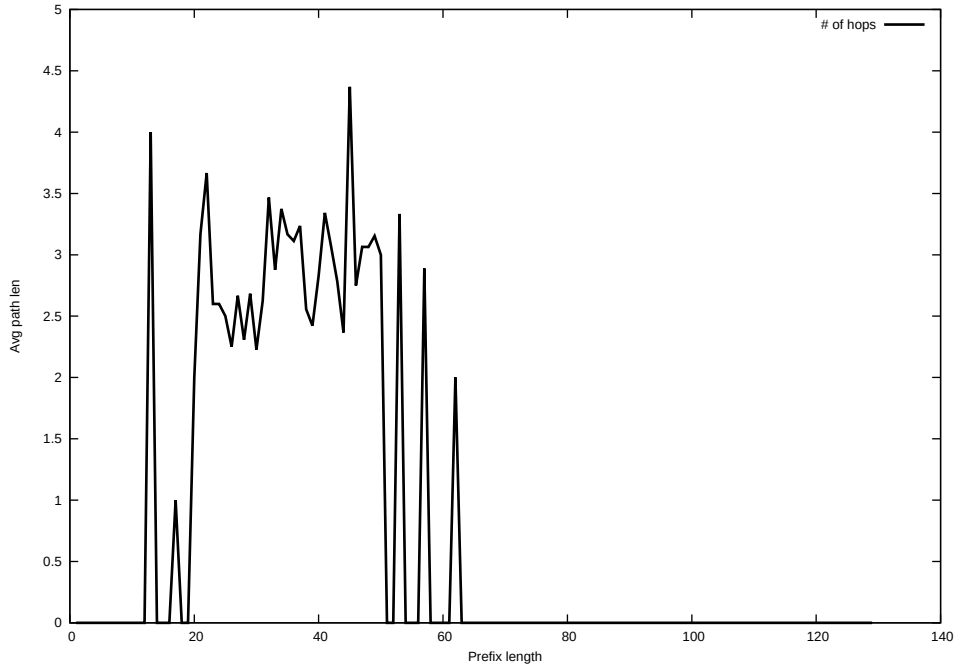
2012-07-07



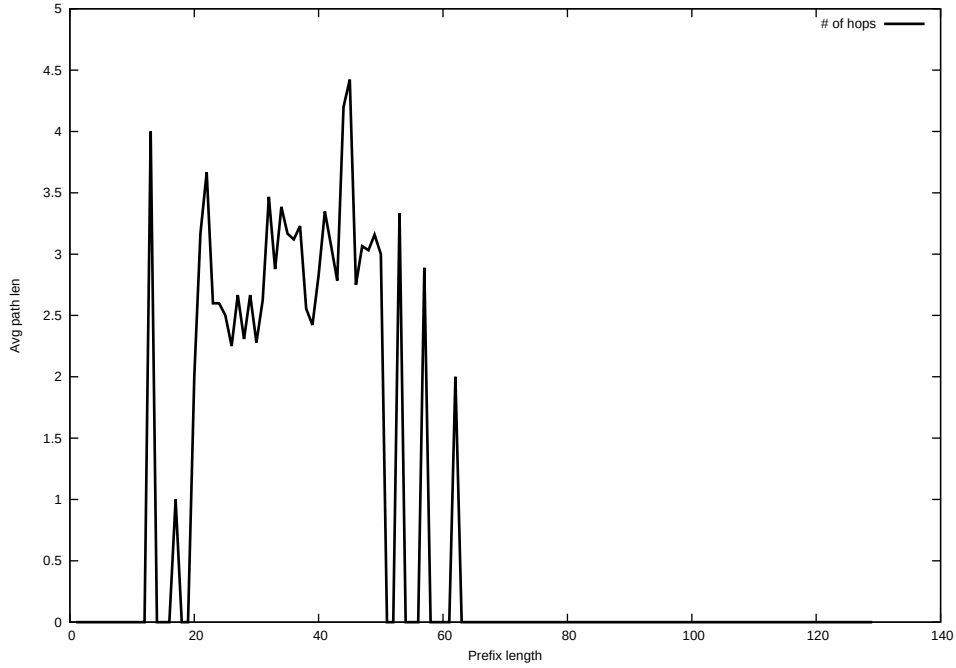
2012-07-08



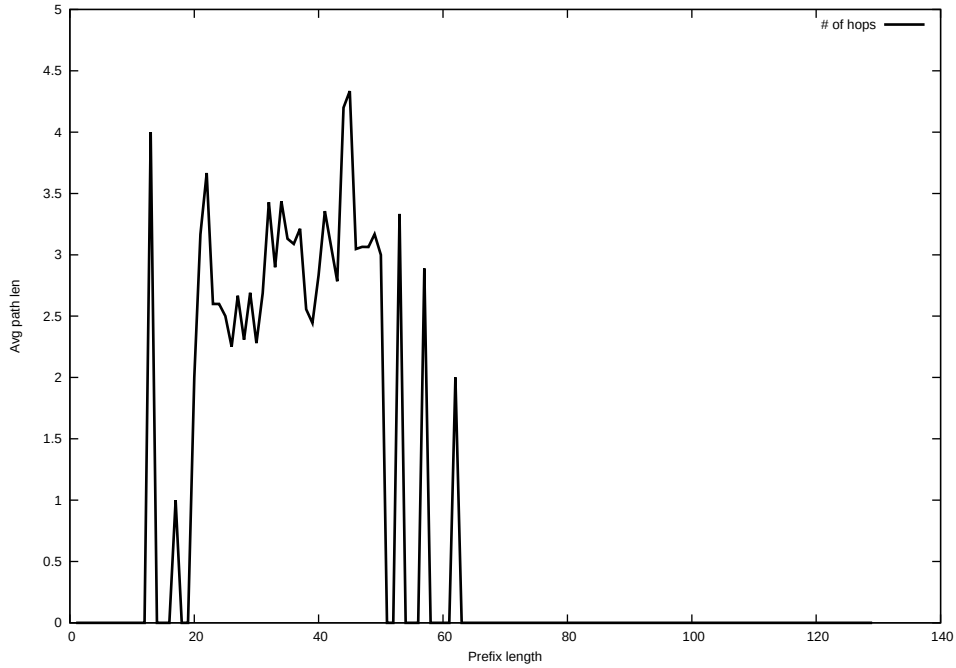
2012-07-09



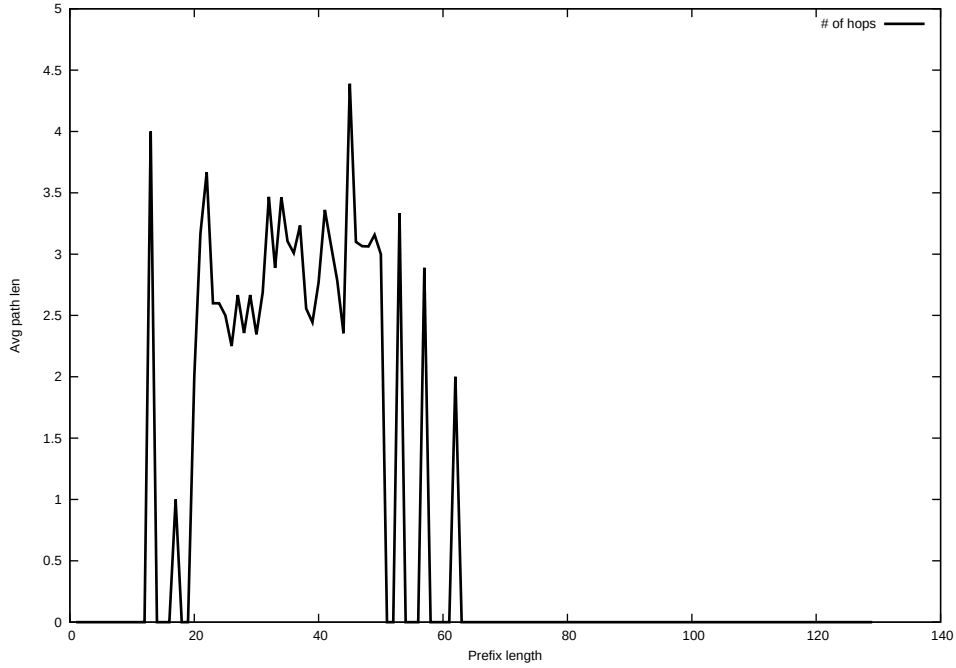
2012-07-10



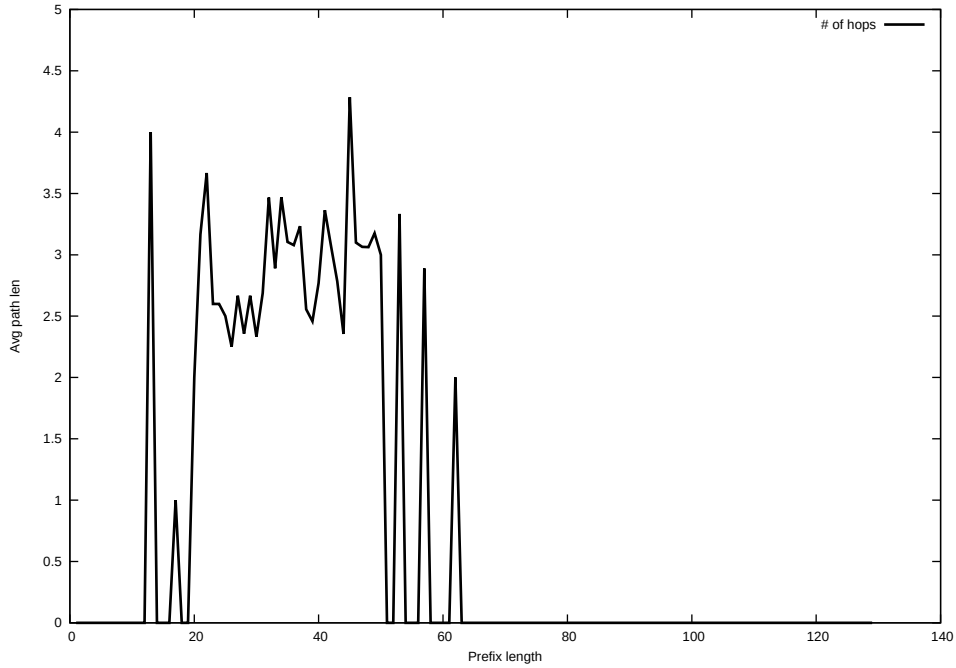
2012-07-11



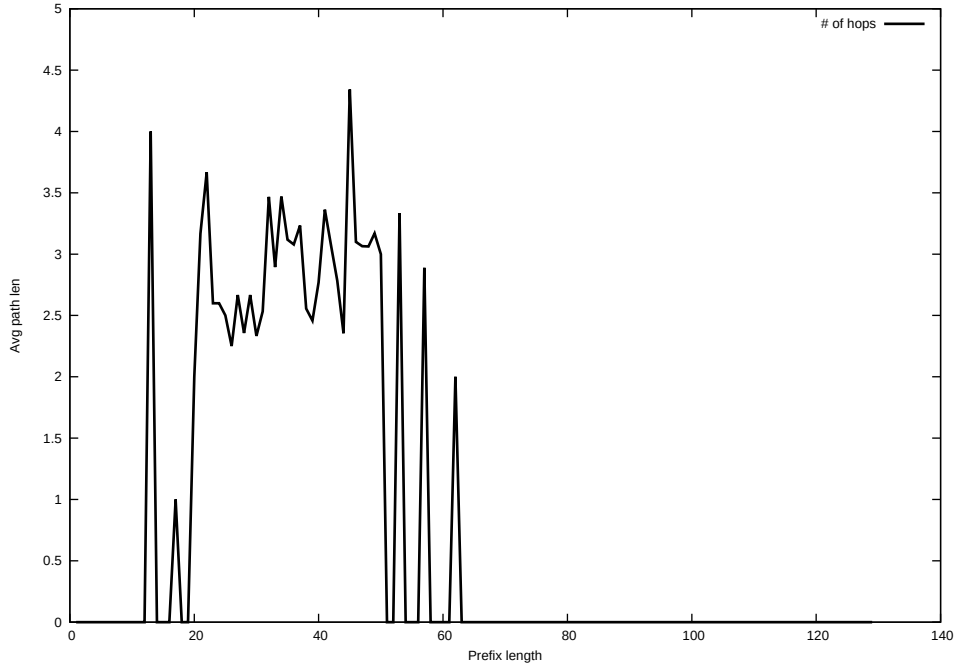
2012-07-12



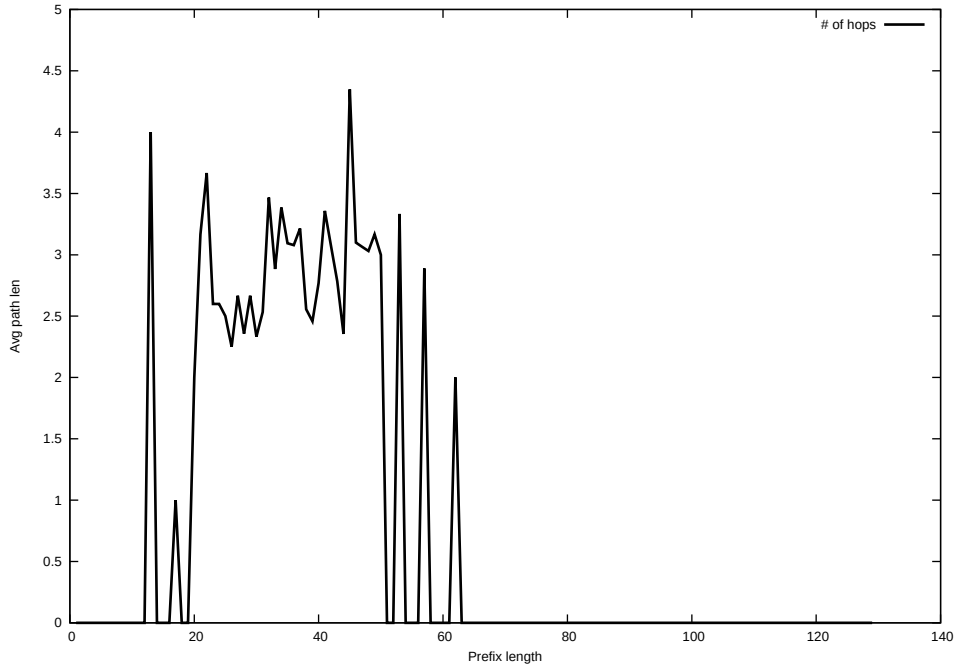
2012-07-13



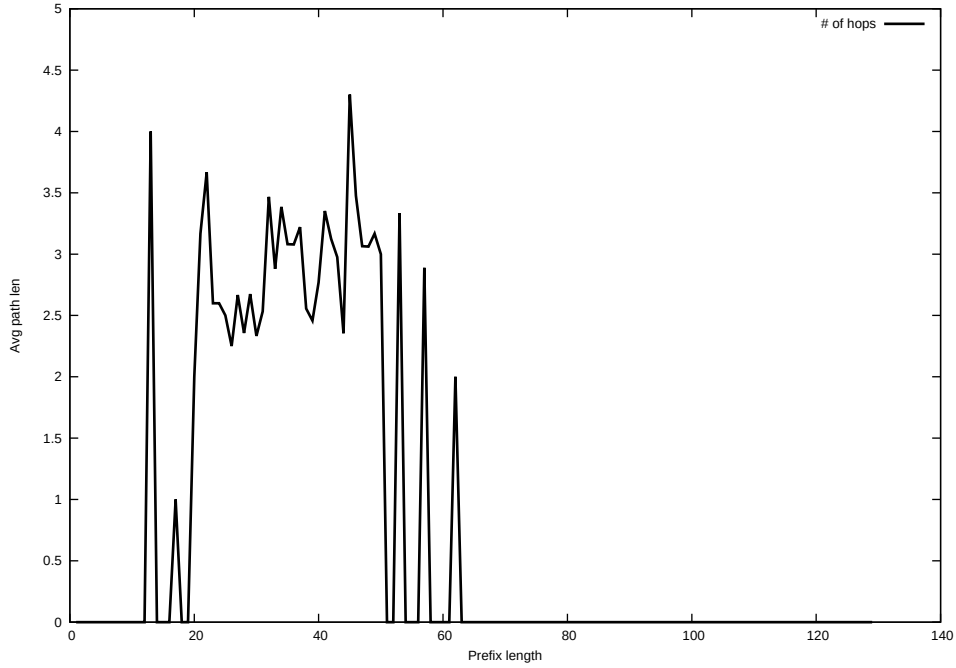
2012-07-14



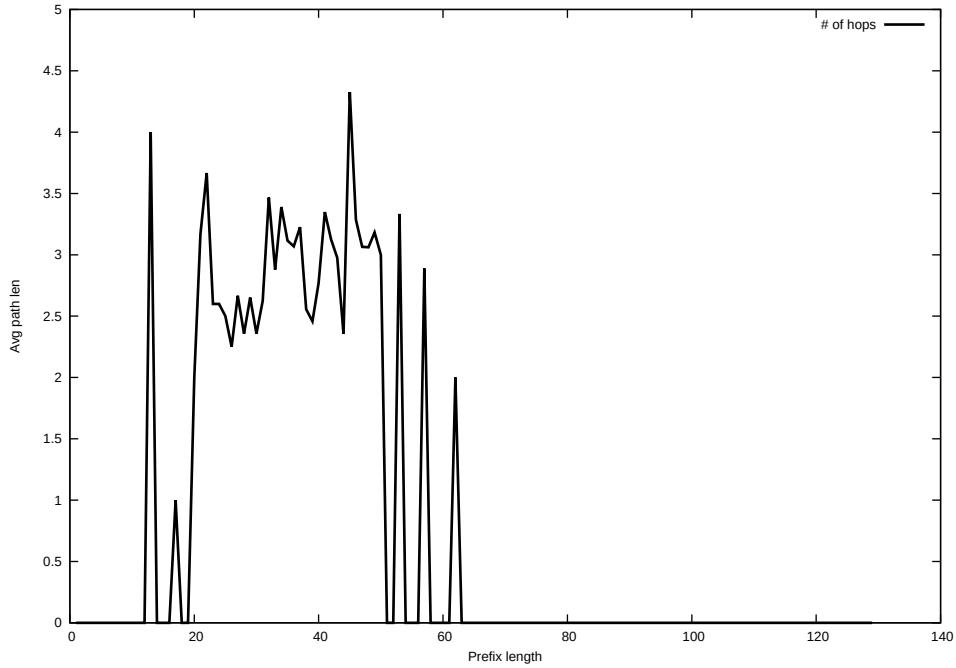
2012-07-15



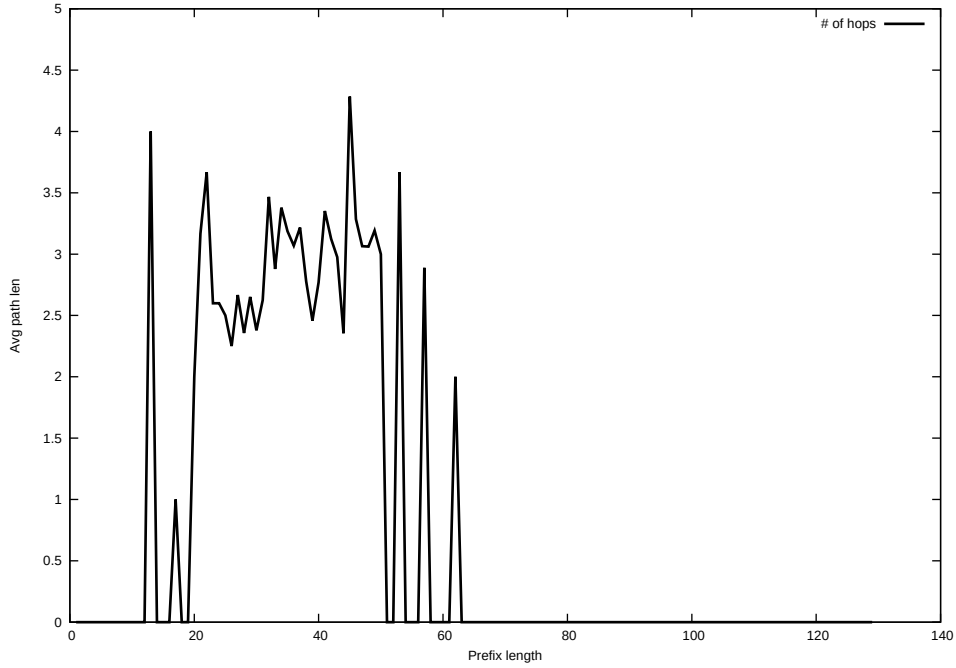
2012-07-16



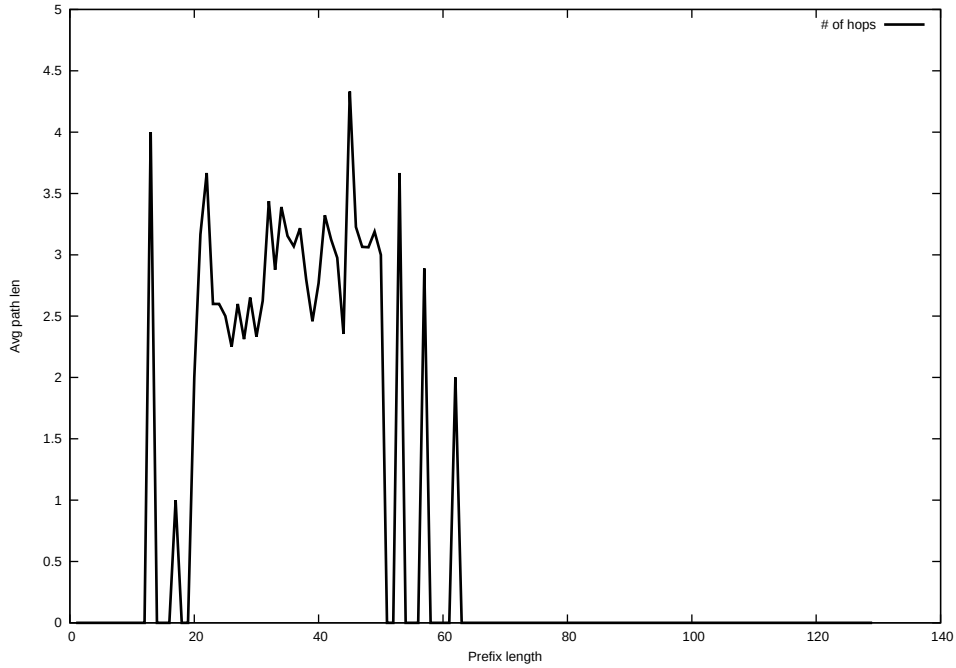
2012-07-17



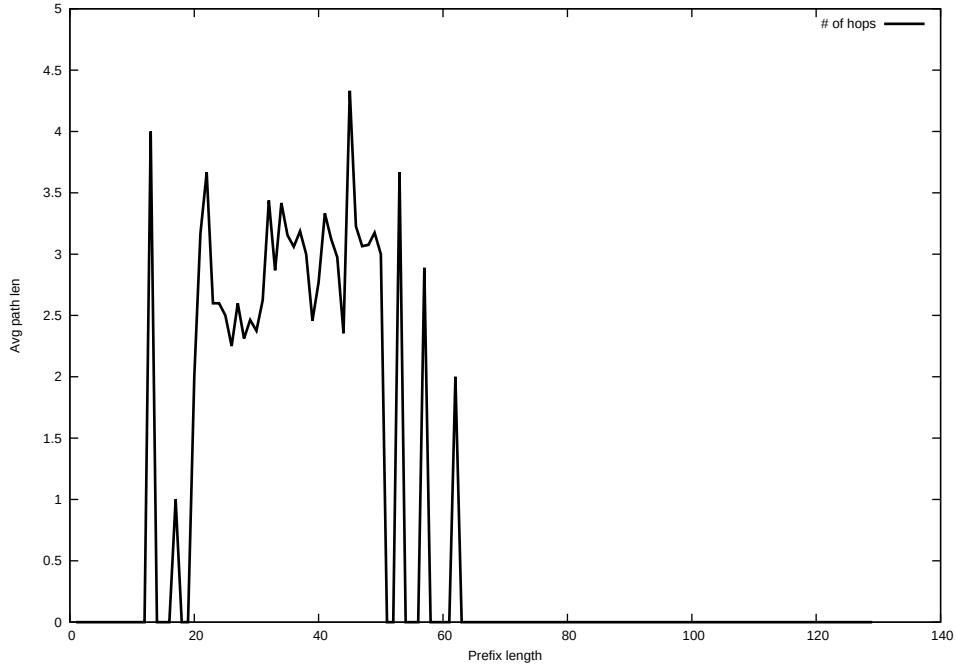
2012-07-18



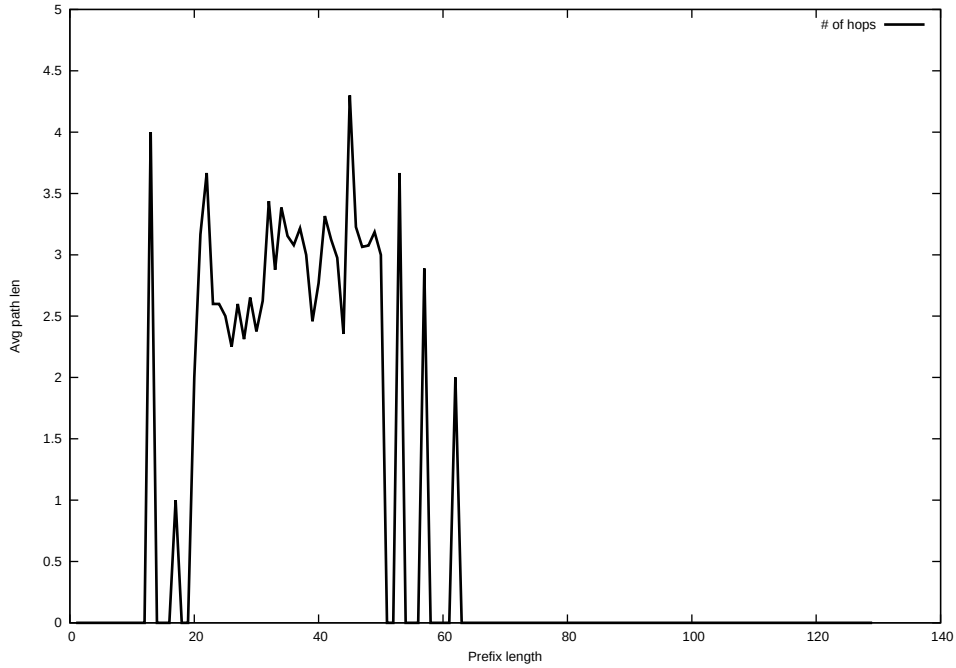
2012-07-19



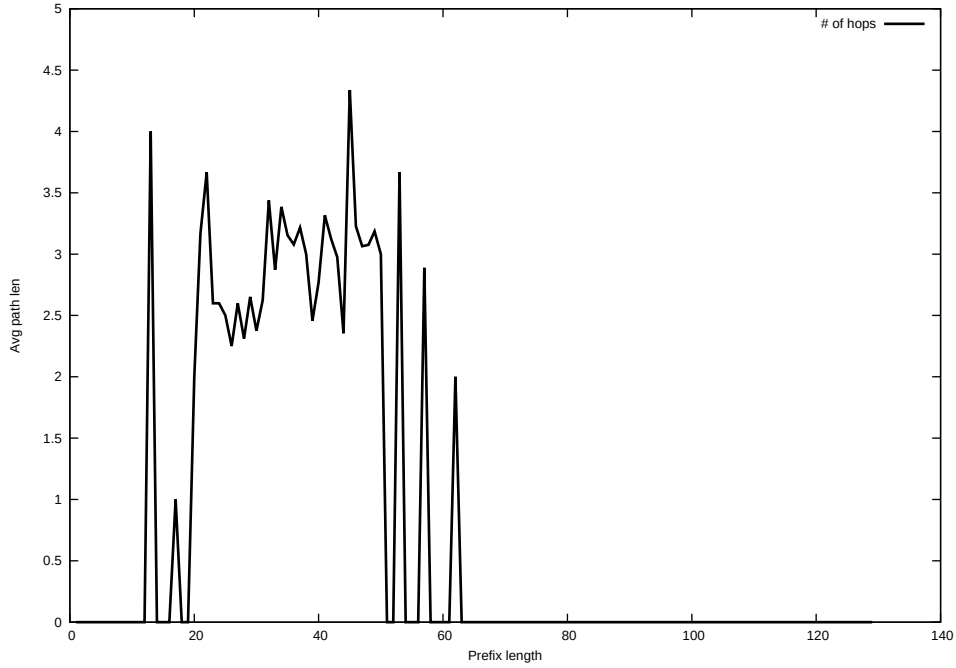
2012-07-20



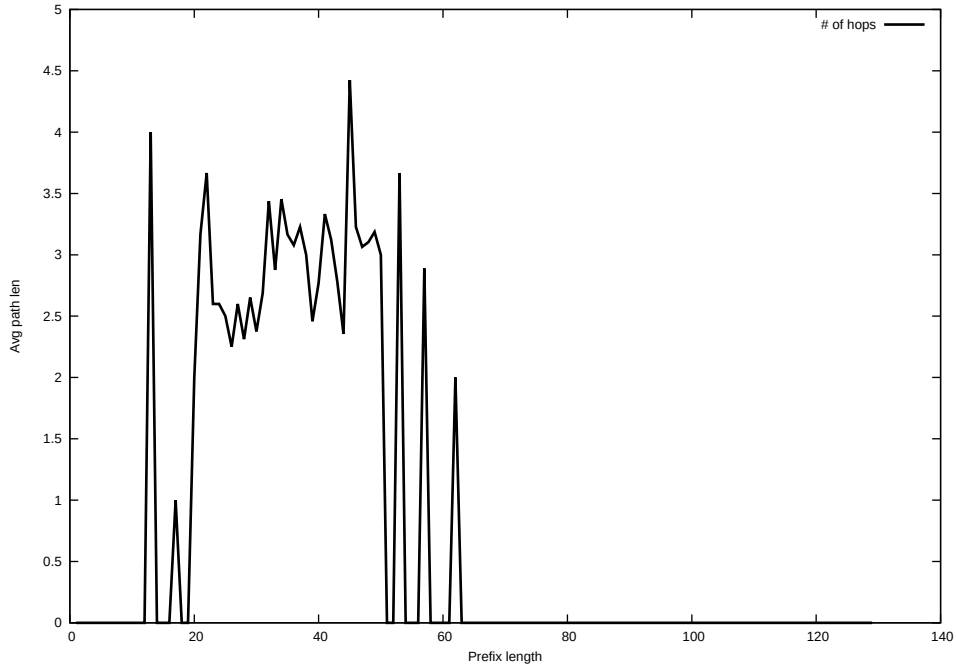
2012-07-21



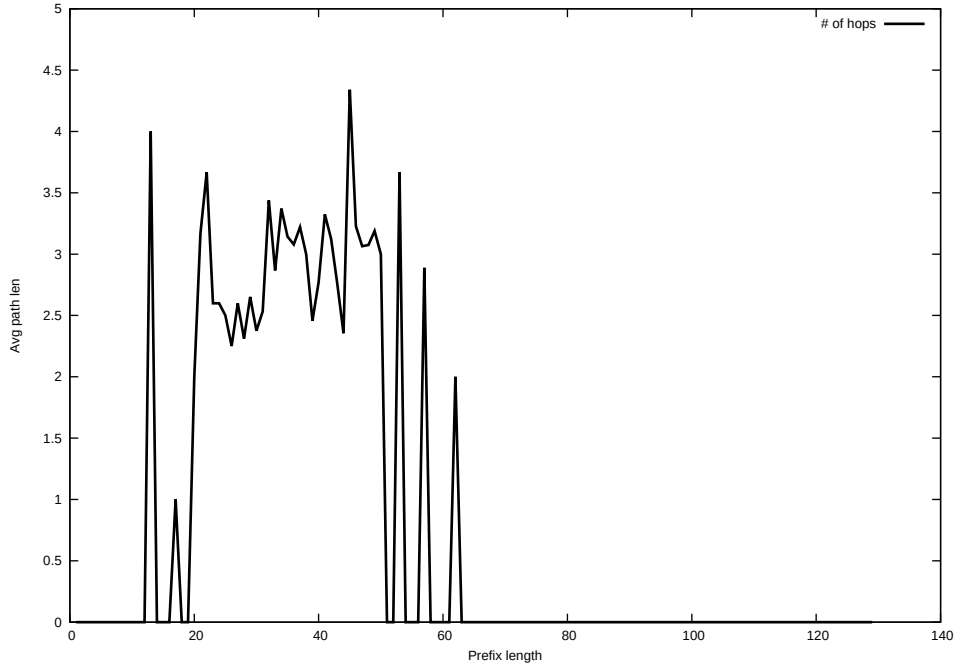
2012-07-22



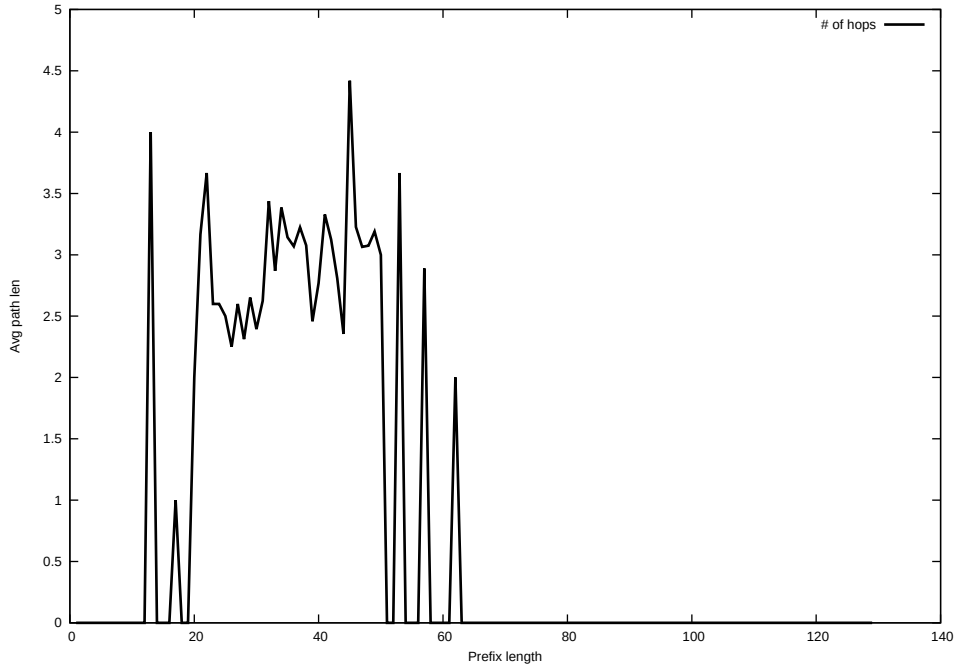
2012-07-23



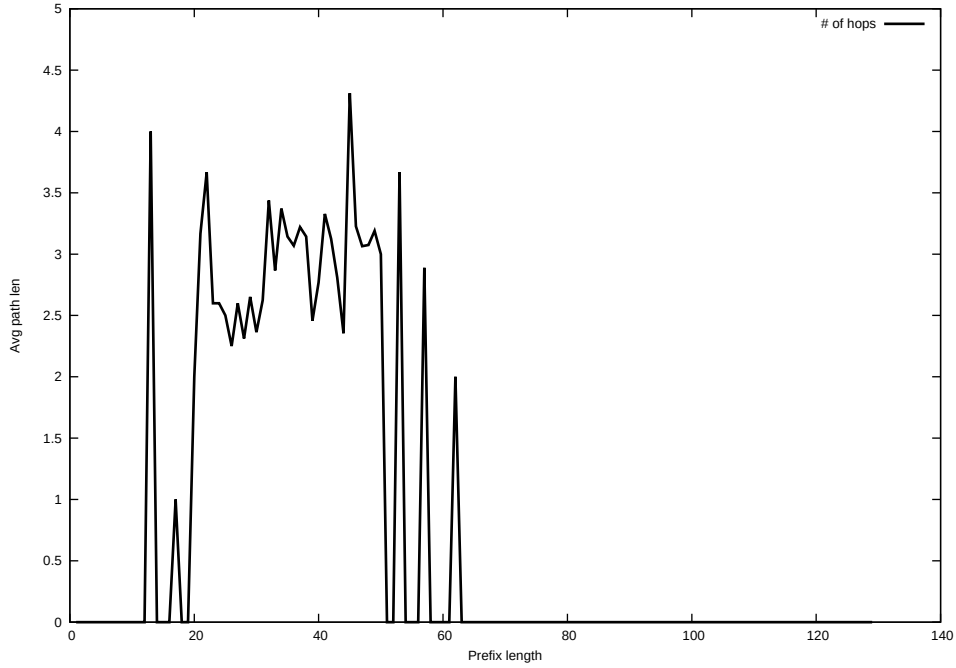
2012-07-24



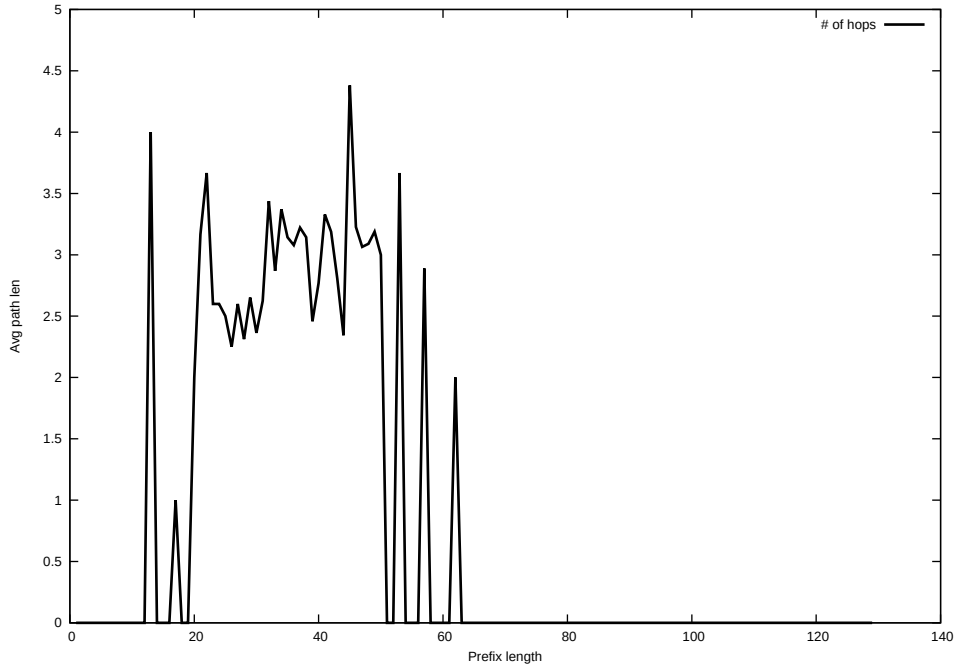
2012-07-25



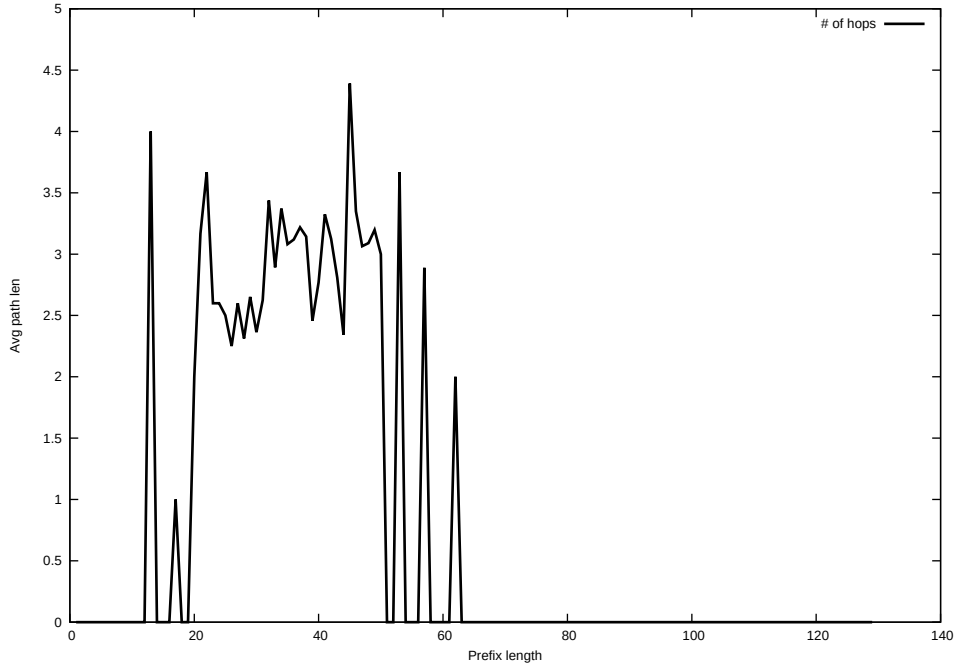
2012-07-26



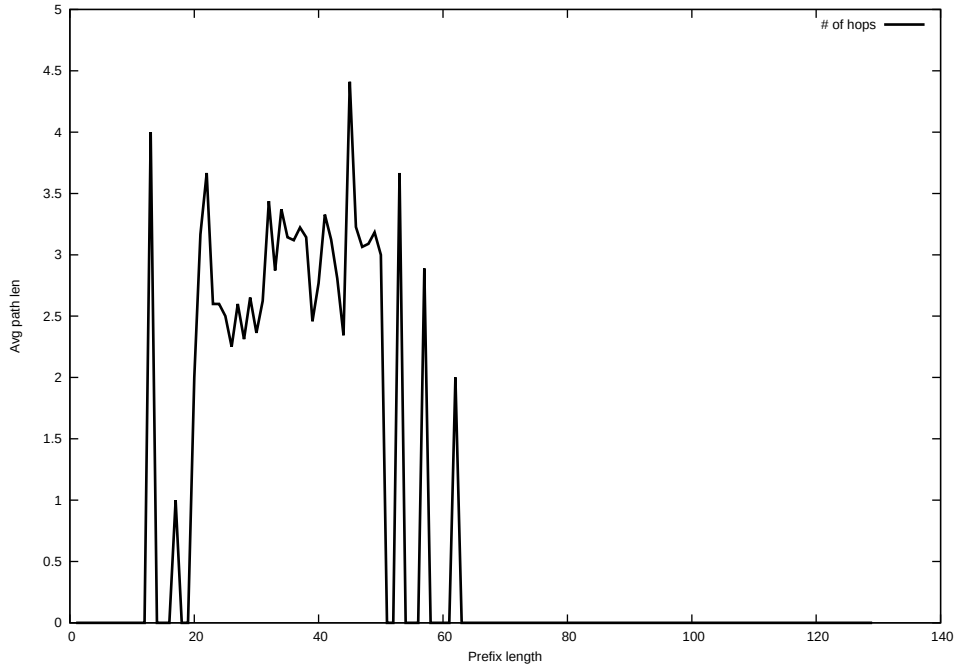
2012-07-27



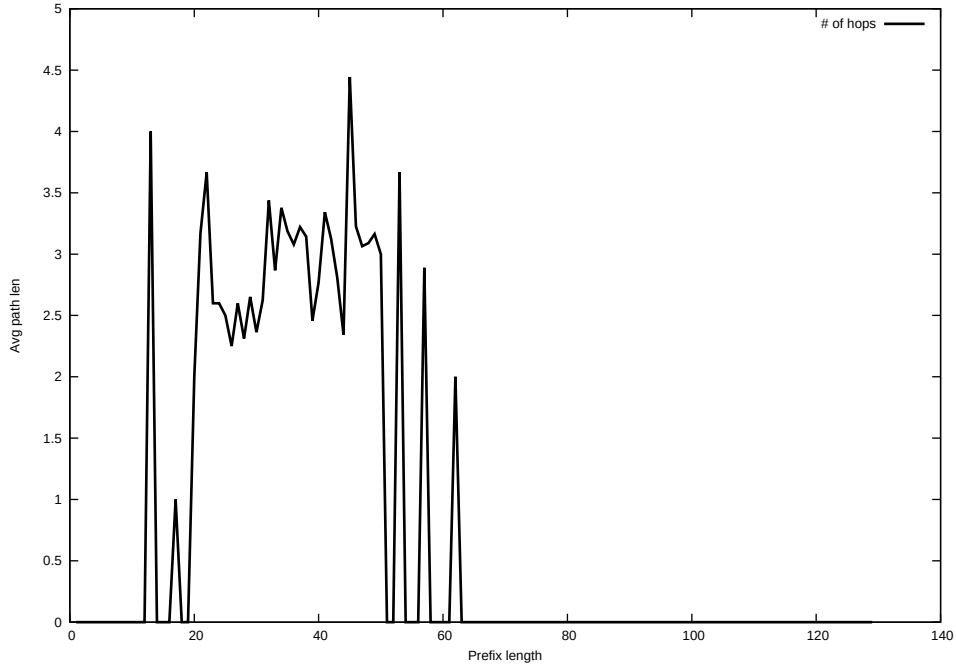
2012-07-28



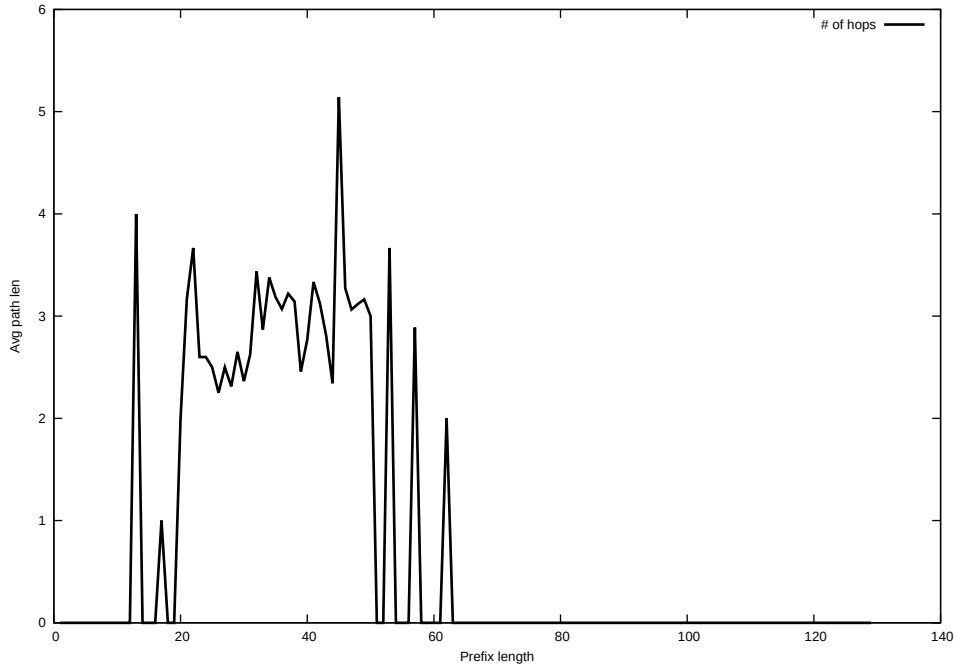
2012-07-29



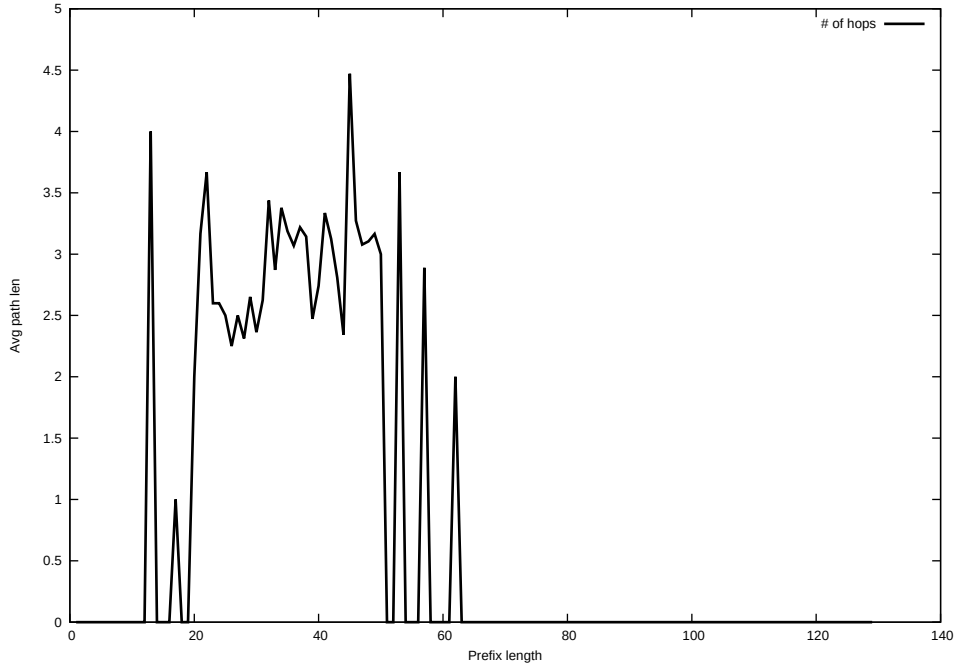
2012-07-30



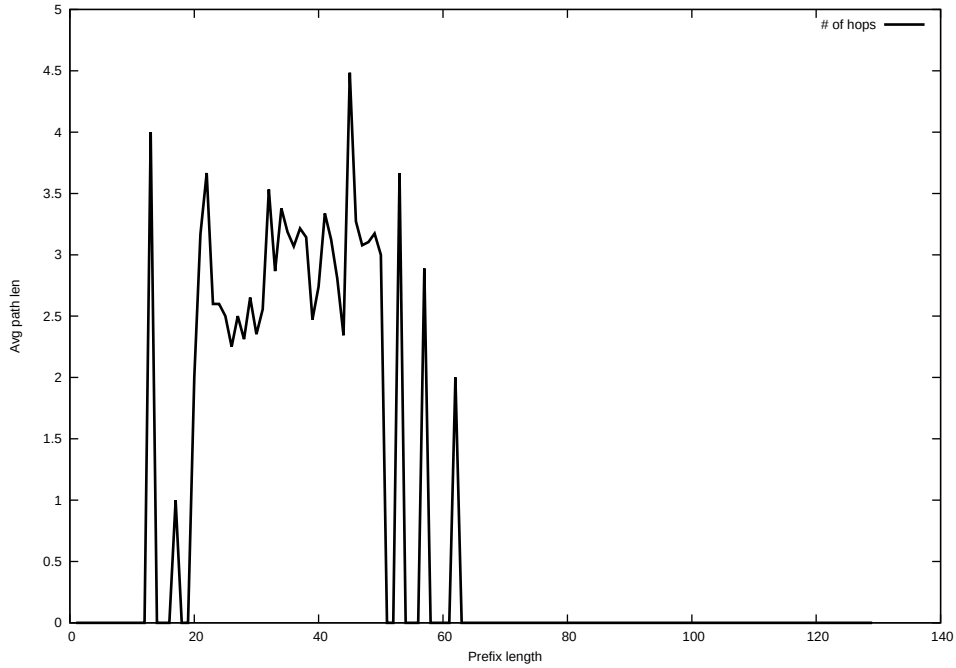
2012-07-31



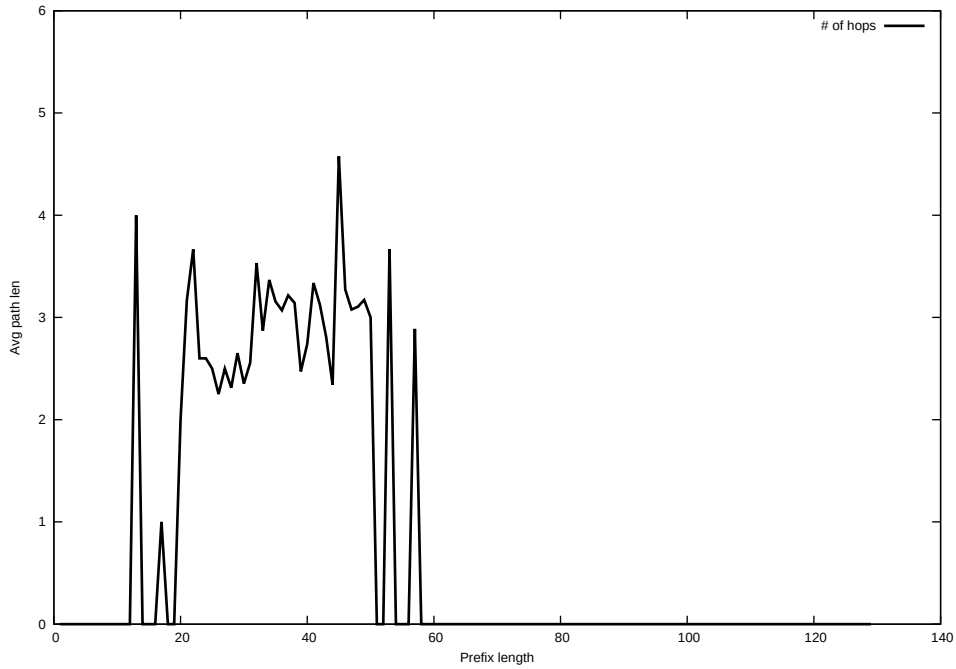
2012-08-01



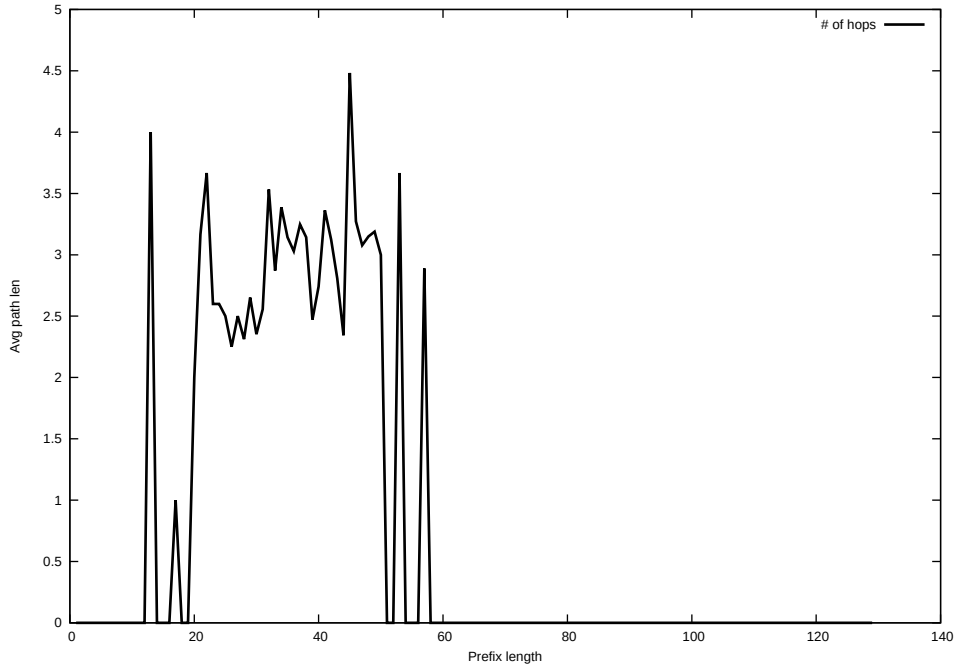
2012-08-02



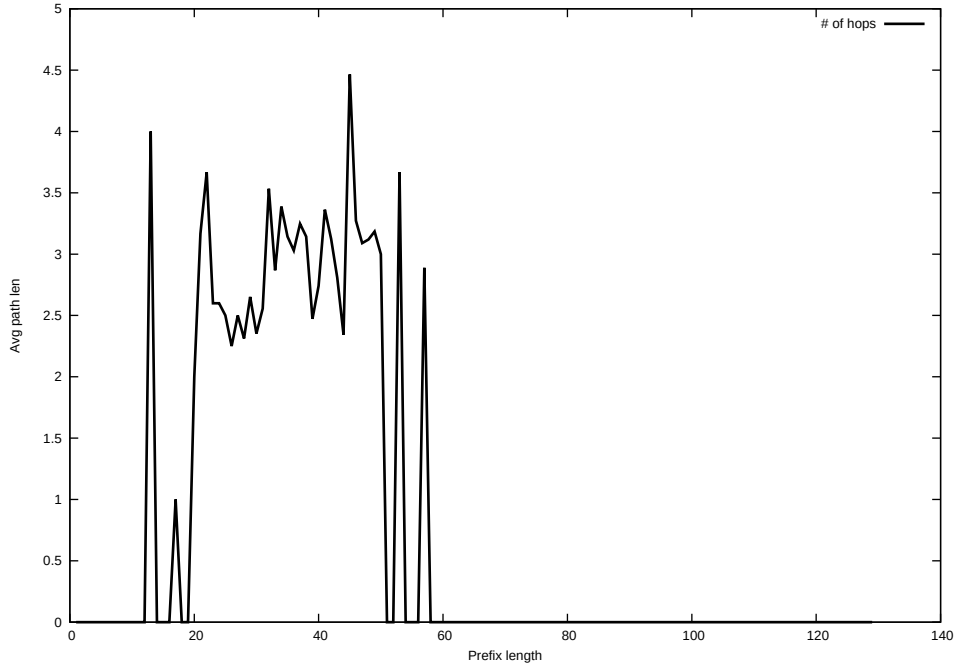
2012-08-03



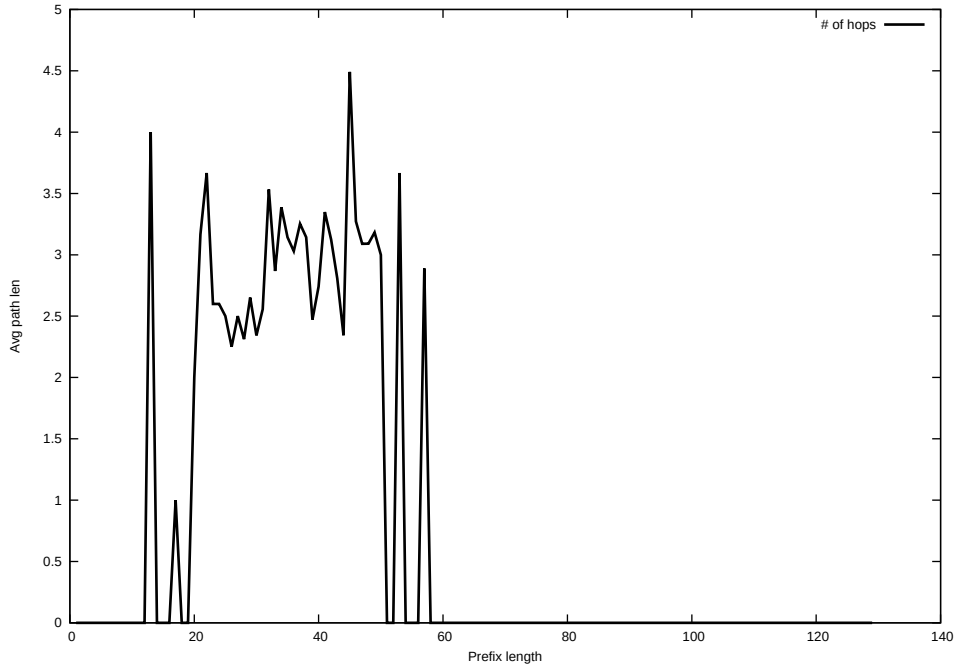
2012-08-04



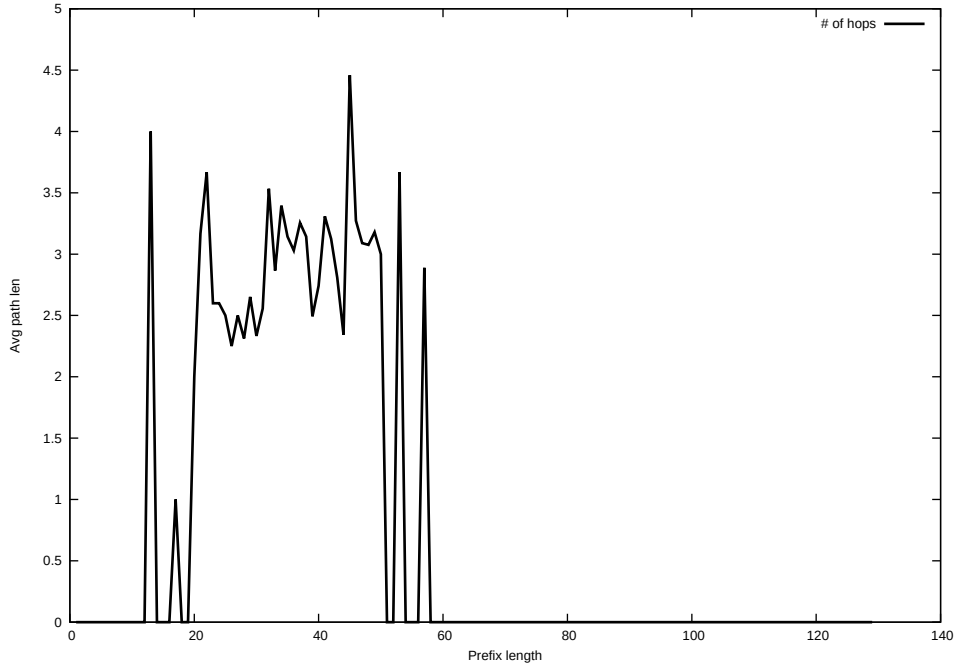
2012-08-05



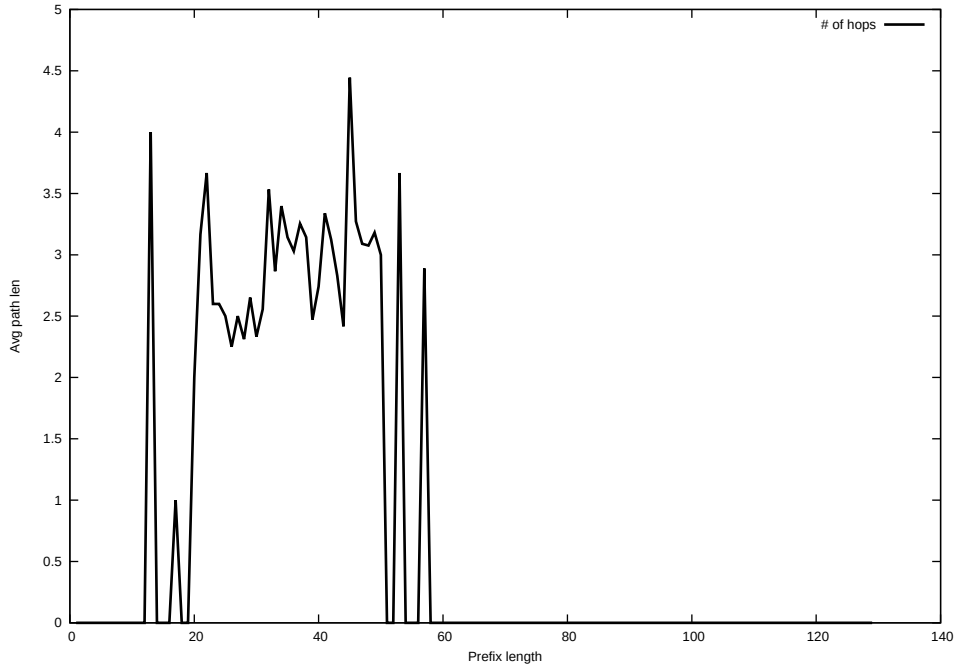
2012-08-06



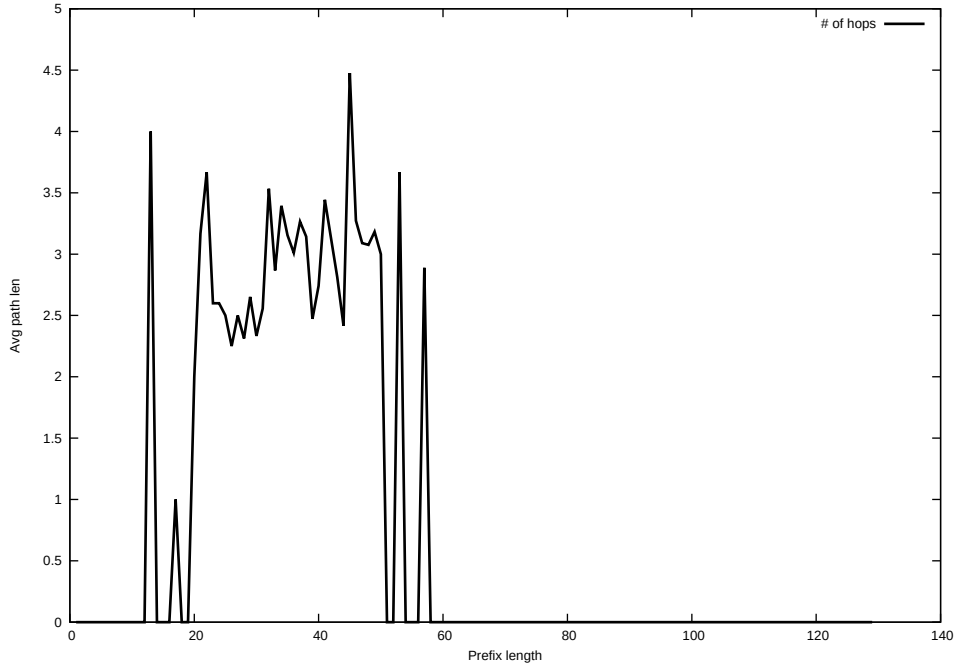
2012-08-07



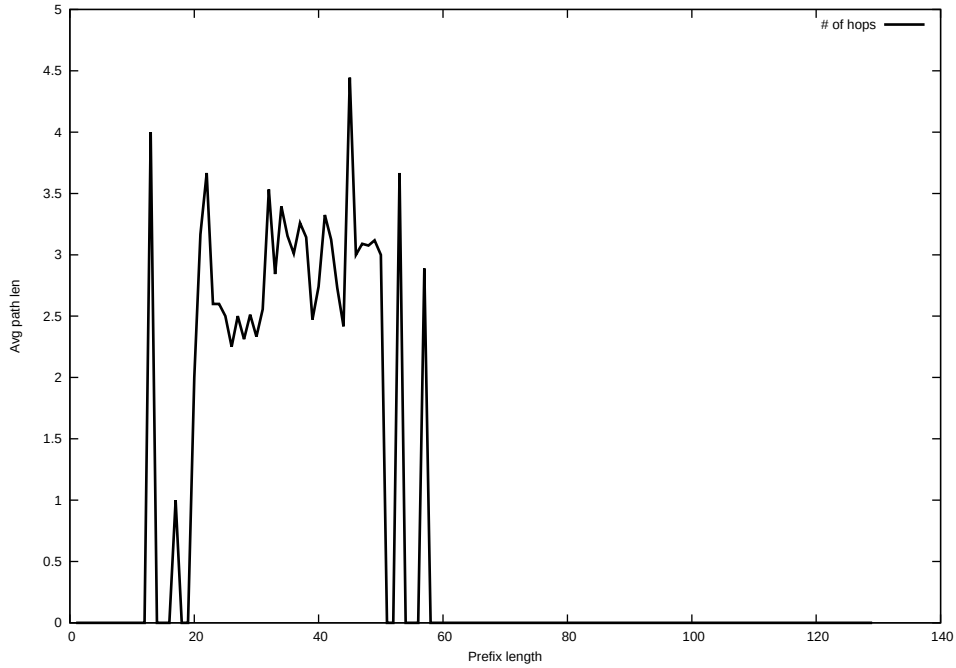
2012-08-08



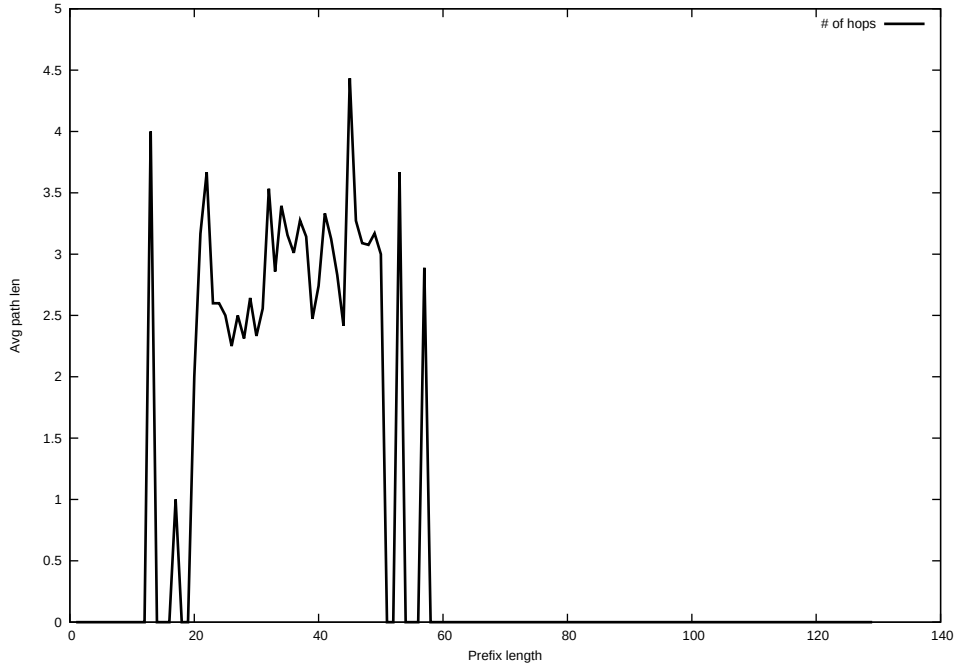
2012-08-09



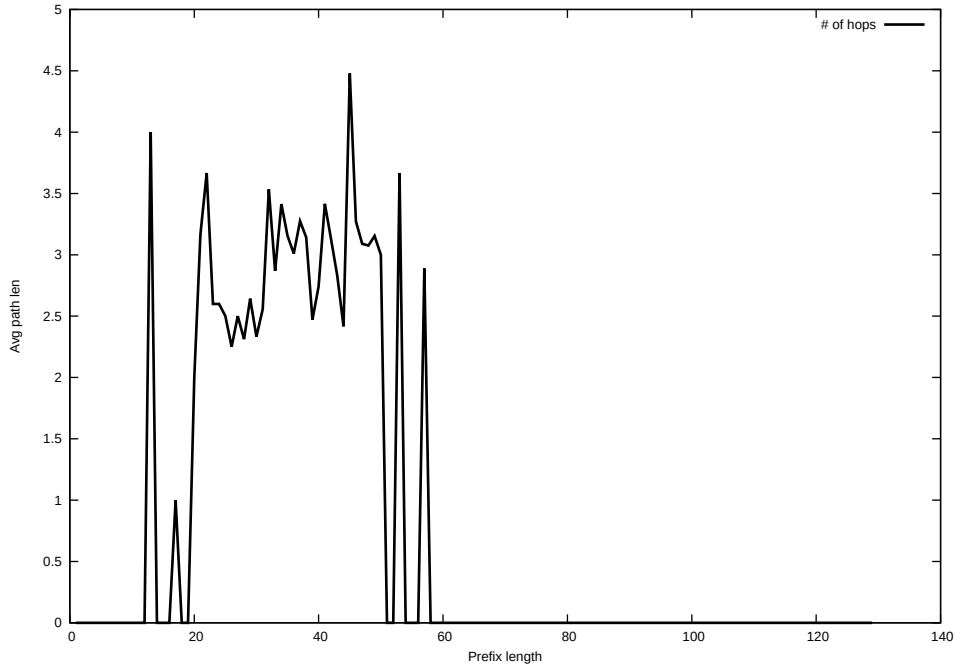
2012-08-10



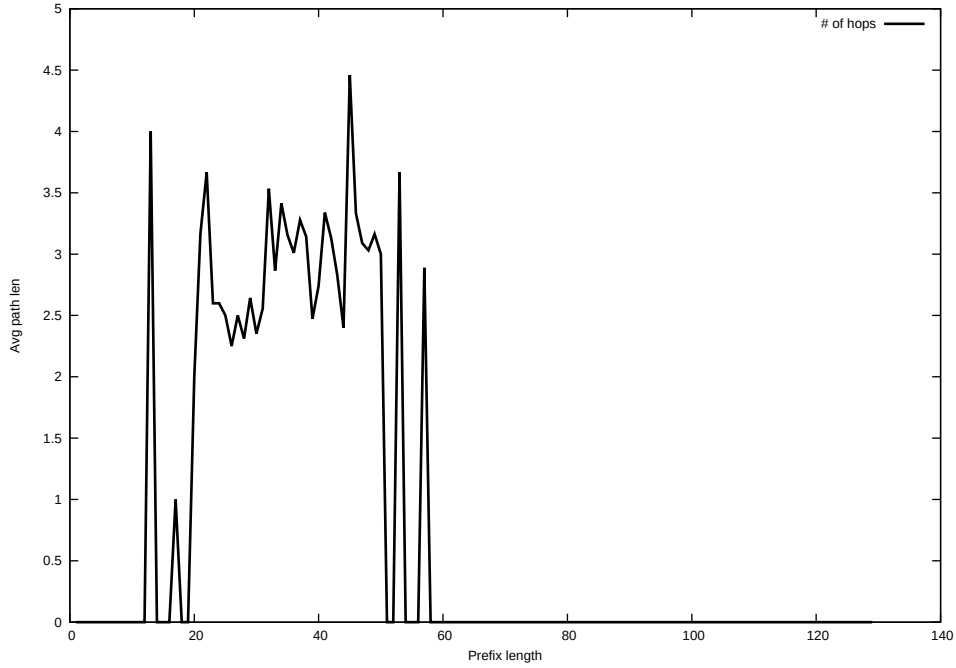
2012-08-11



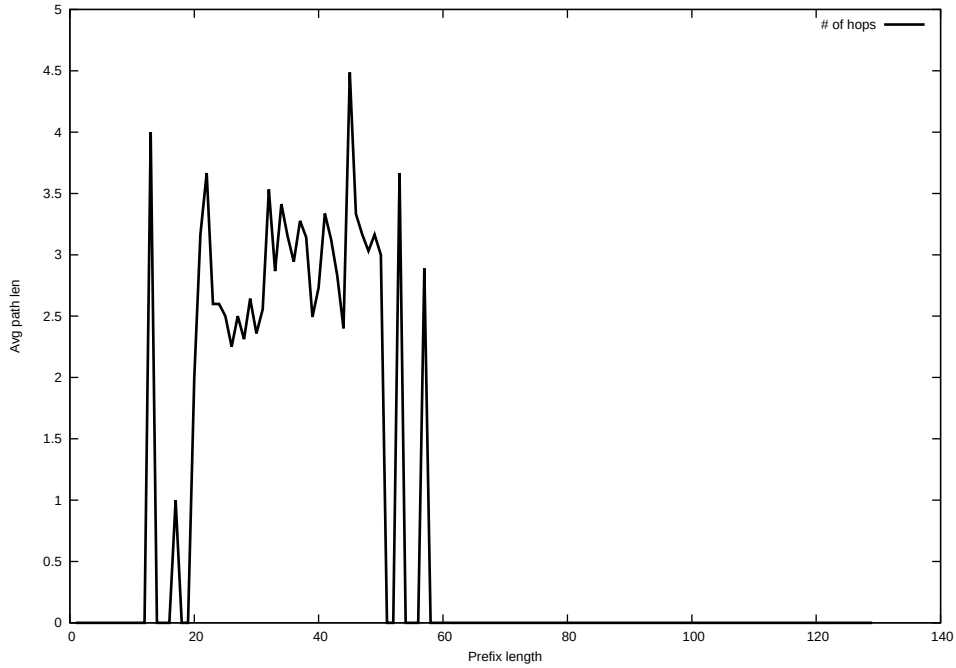
2012-08-12



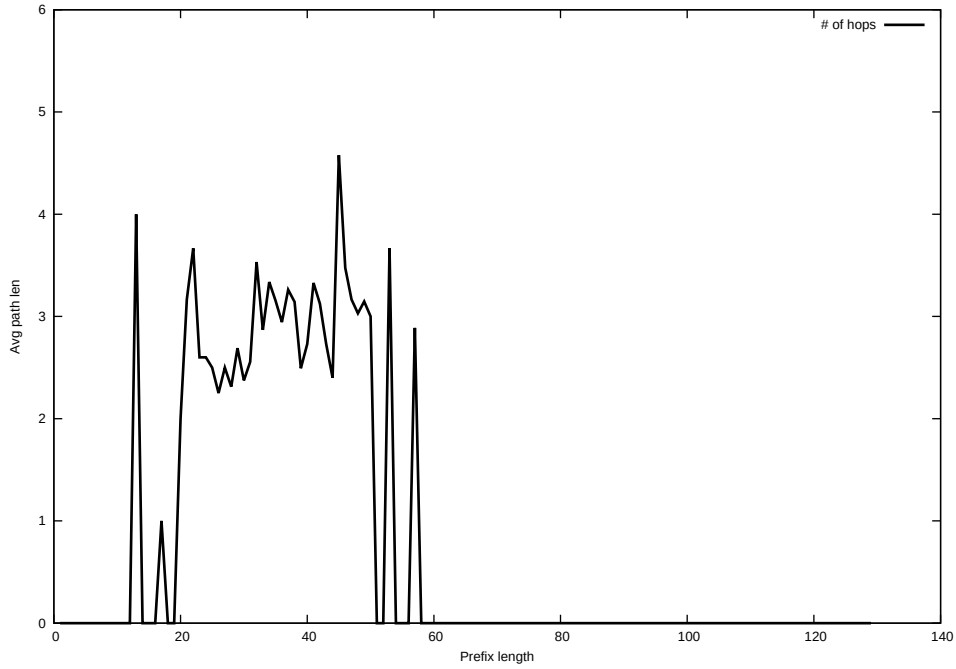
2012-08-13



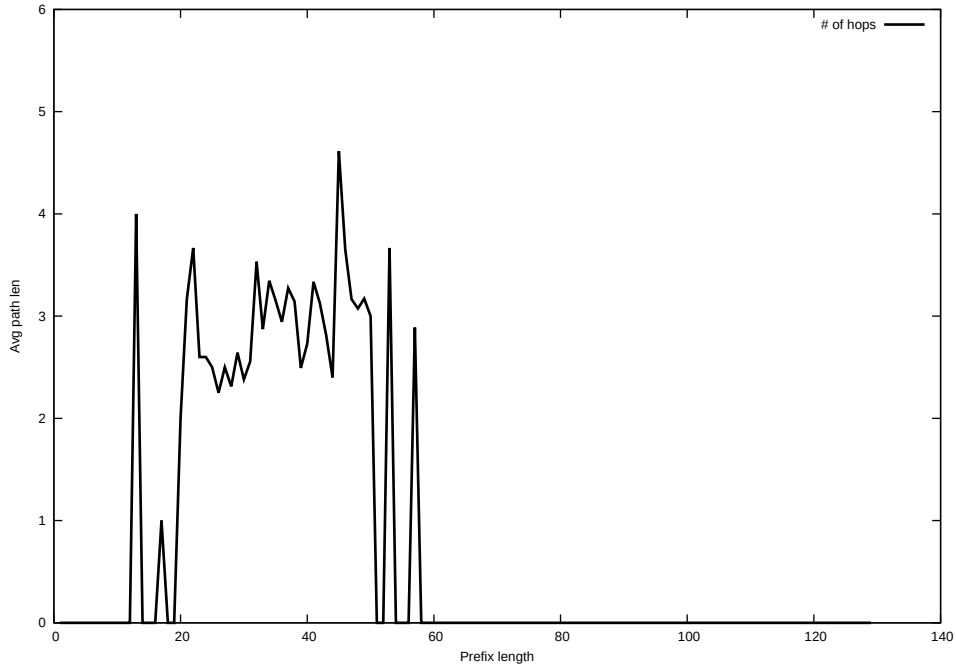
2012-08-14



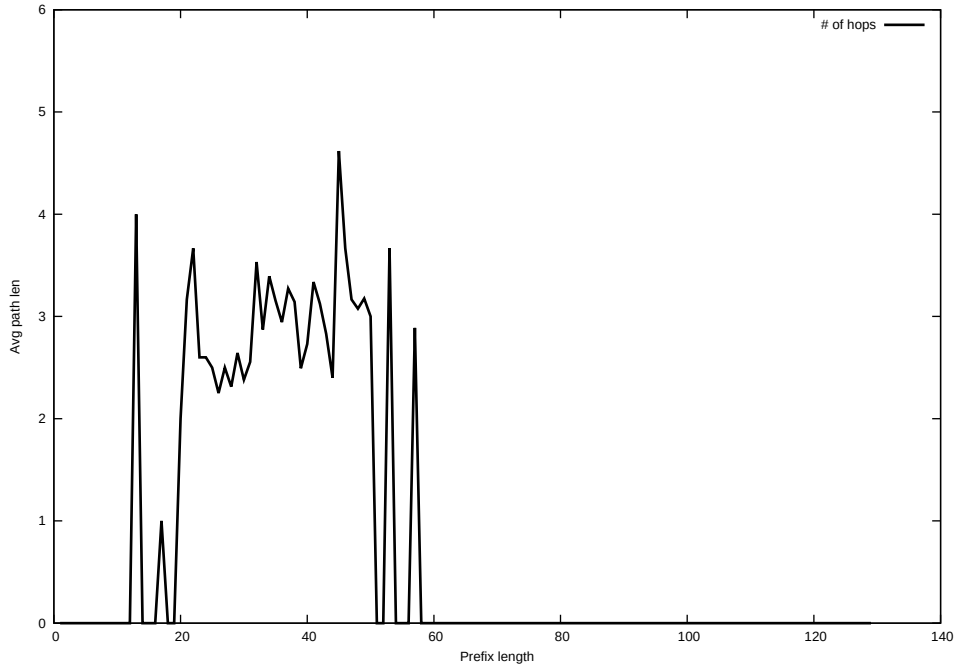
2012-08-15



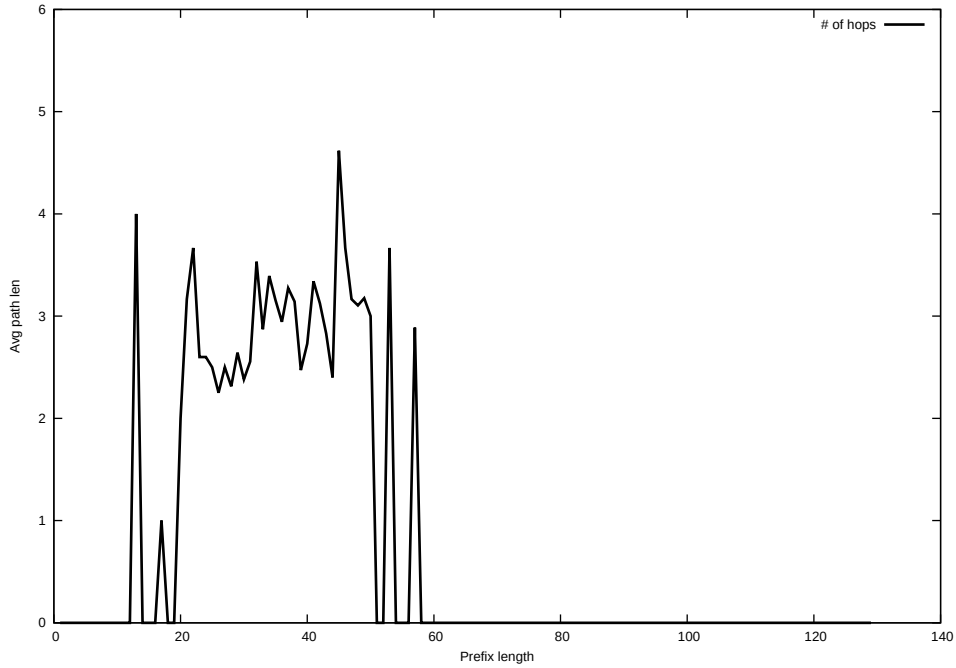
2012-08-16



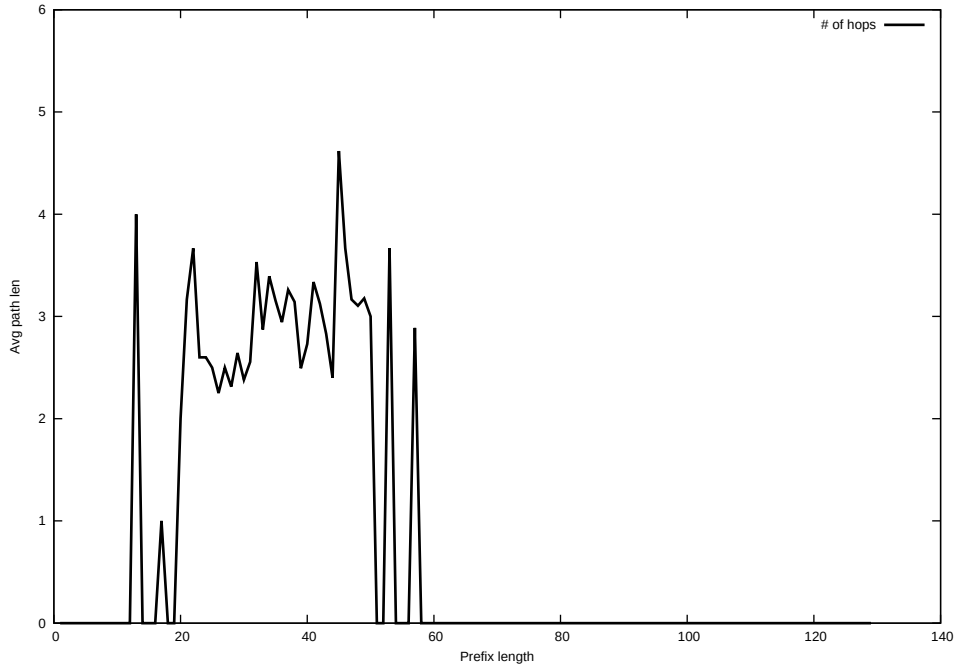
2012-08-17



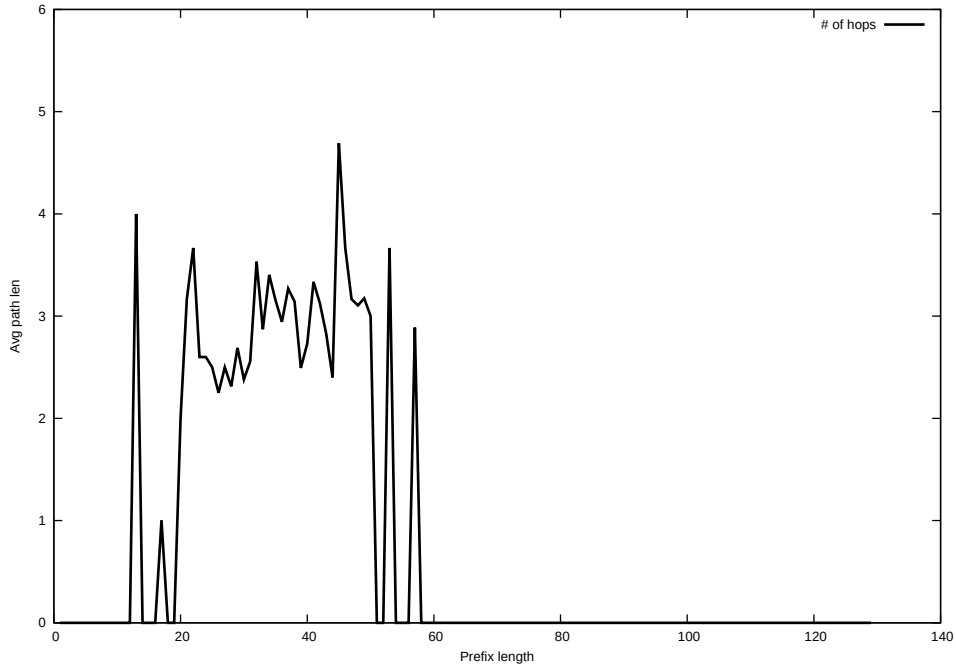
2012-08-18



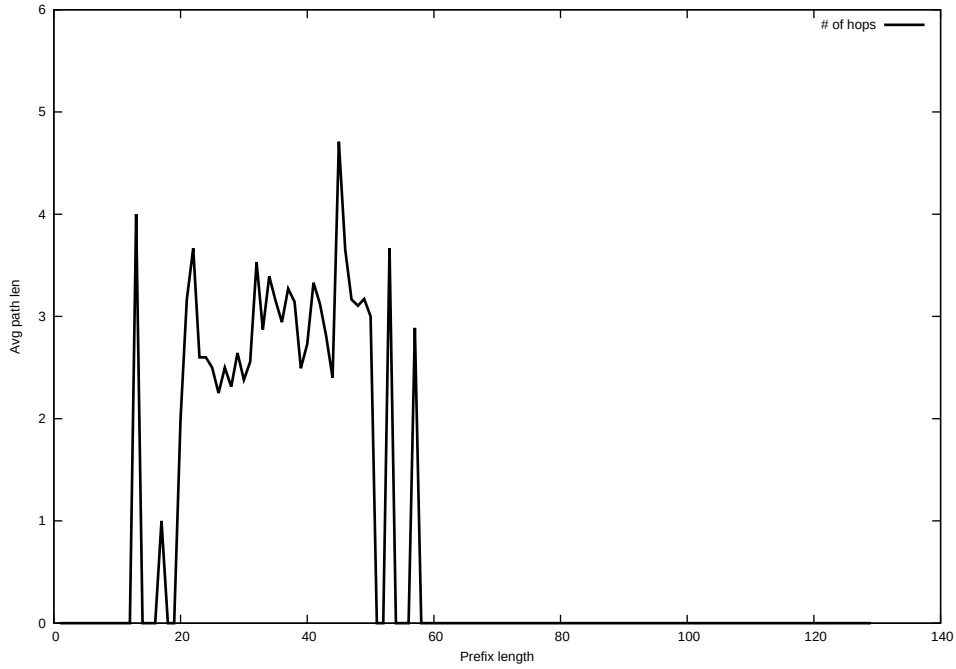
2012-08-19



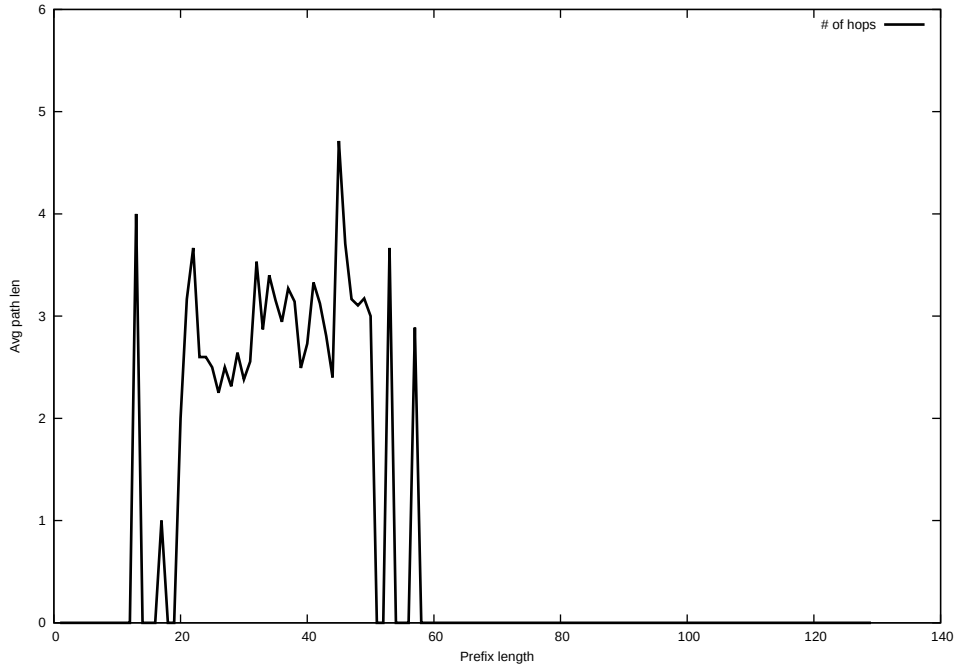
2012-08-20



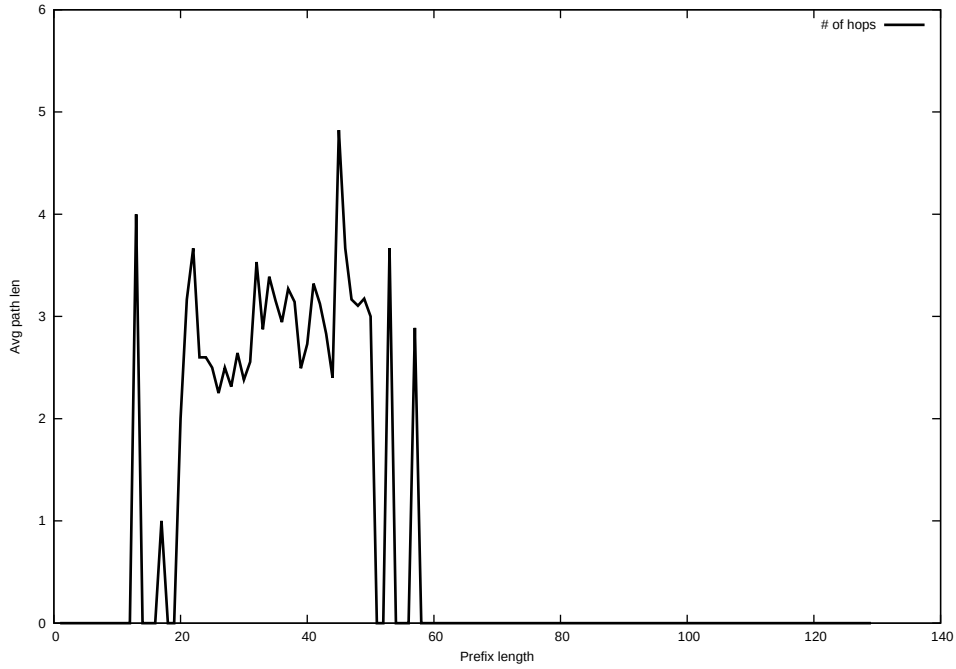
2012-08-21



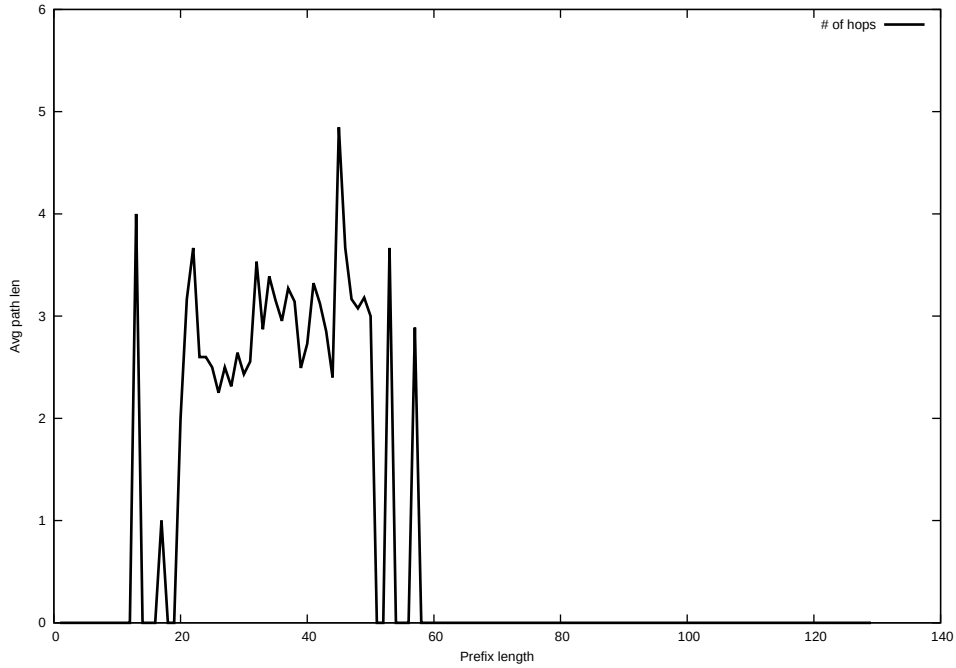
2012-08-22



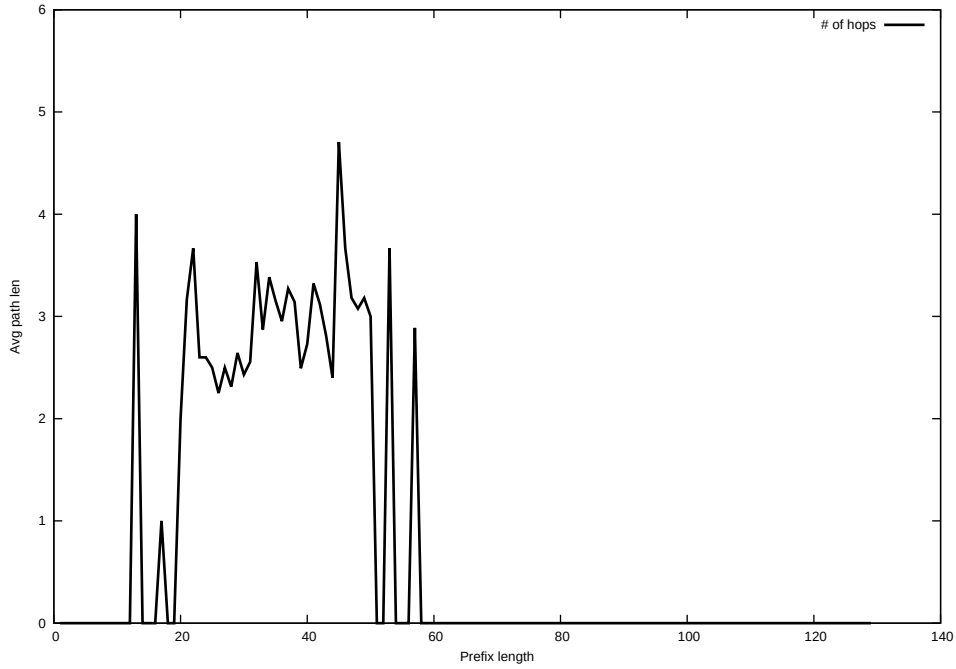
2012-08-23



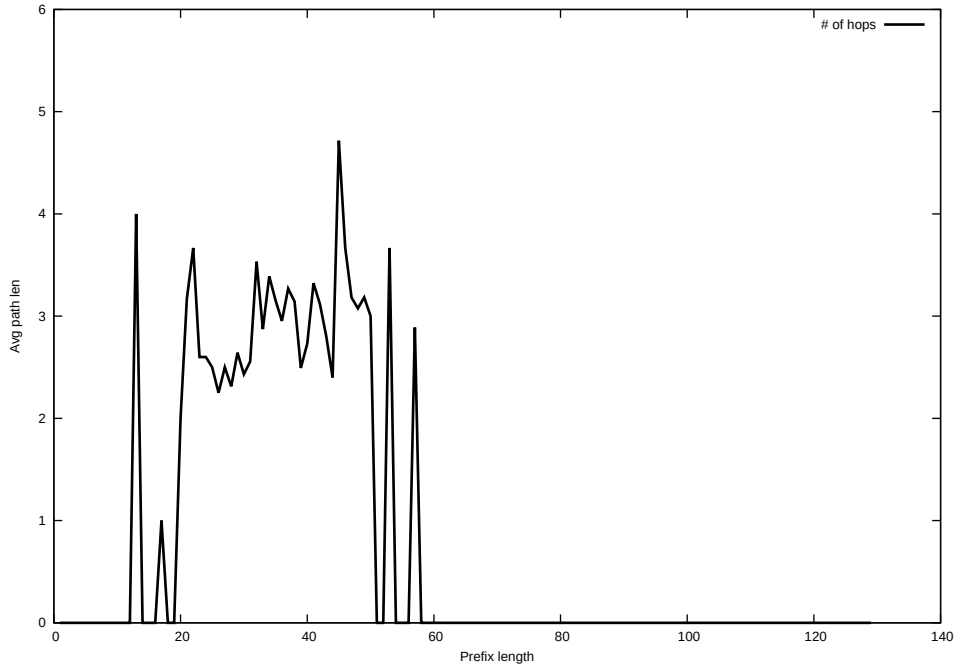
2012-08-24



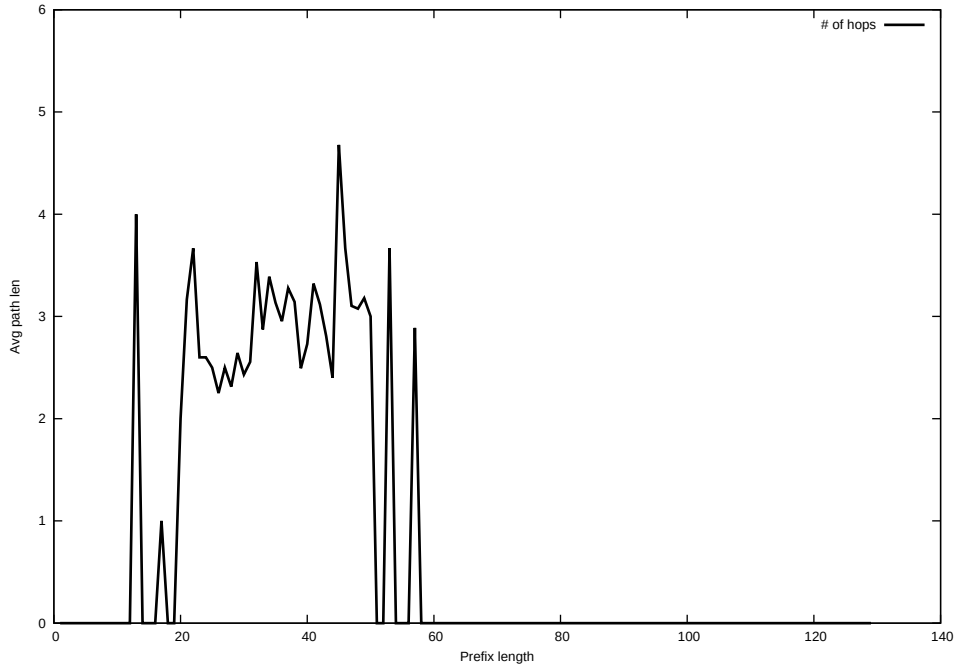
2012-08-25



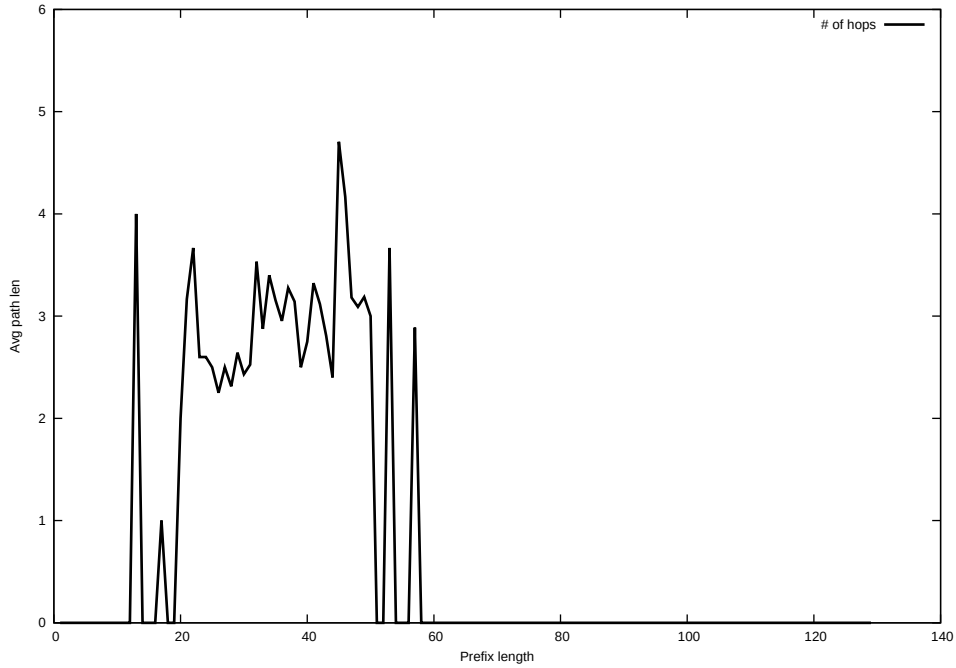
2012-08-26



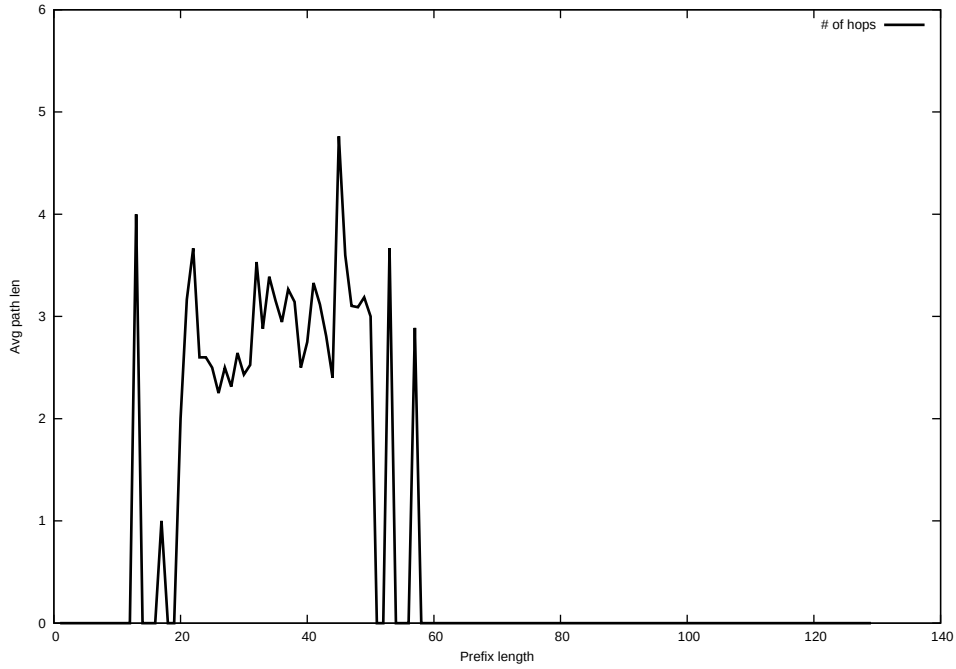
2012-08-27



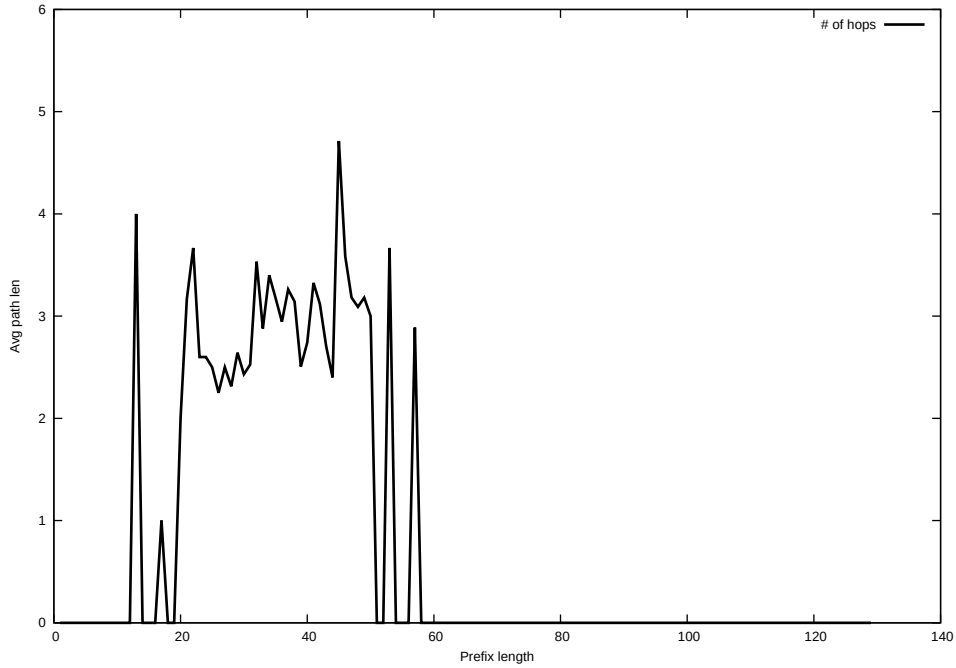
2012-08-28



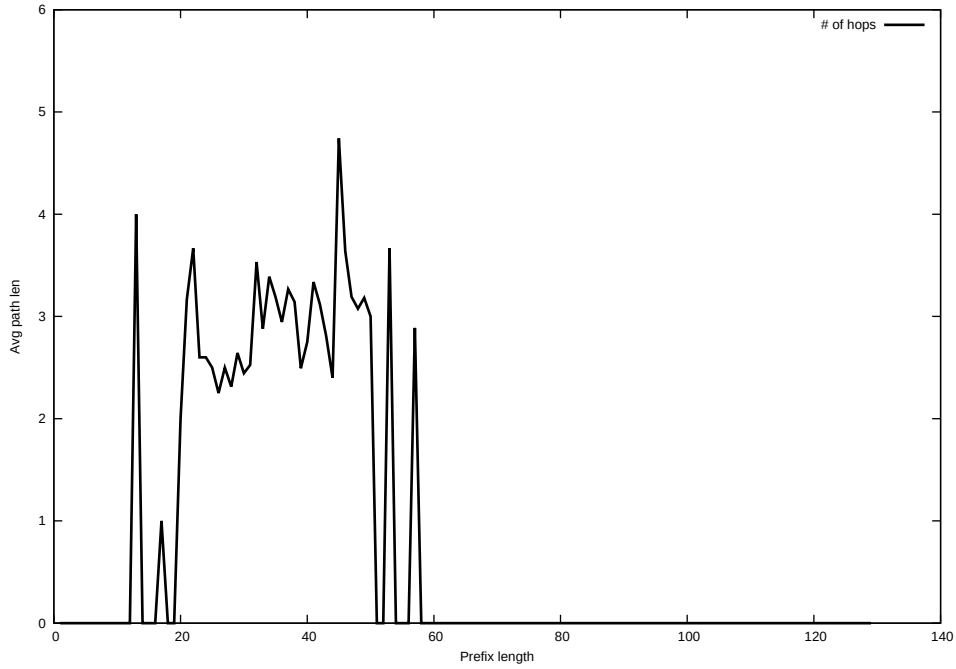
2012-08-29



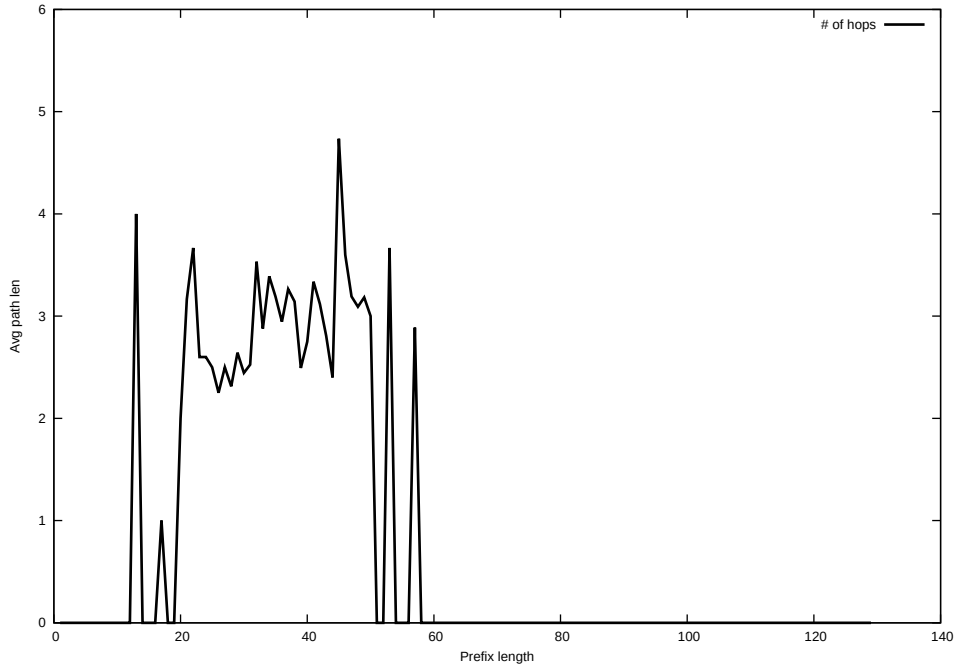
2012-08-30



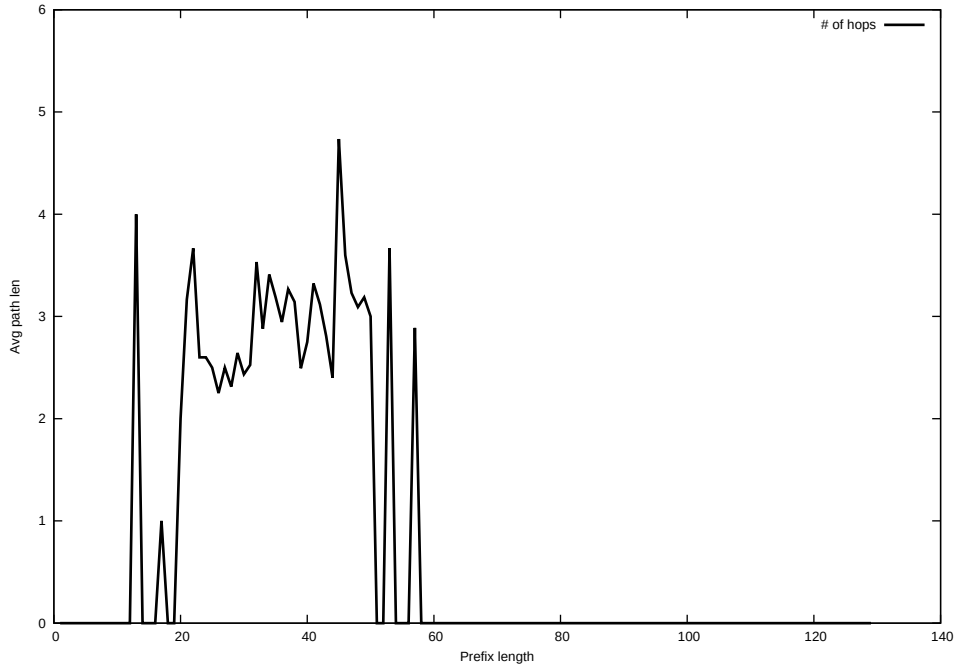
2012-08-31



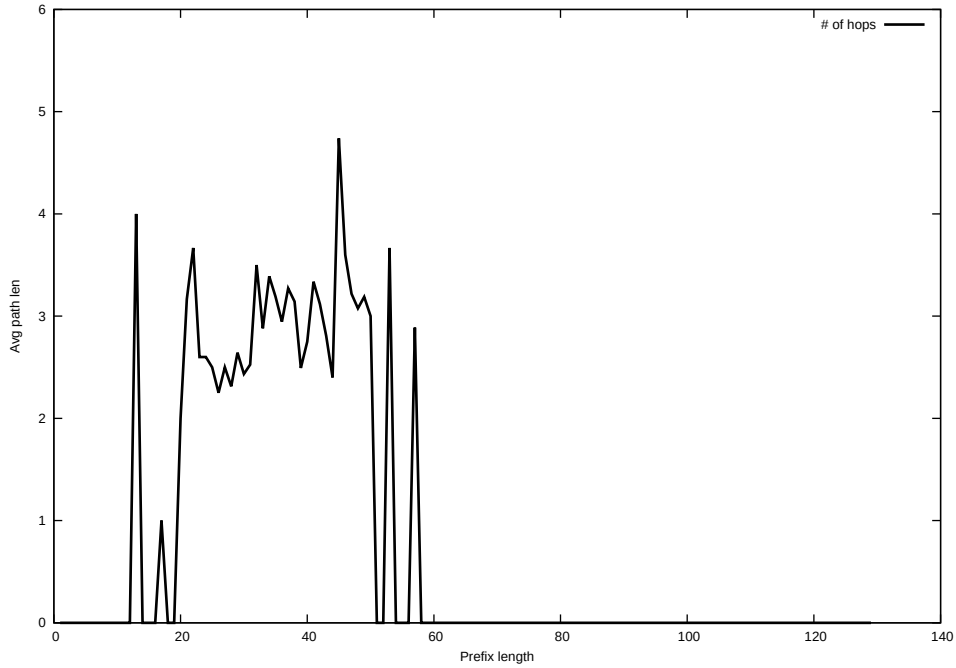
2012-09-01



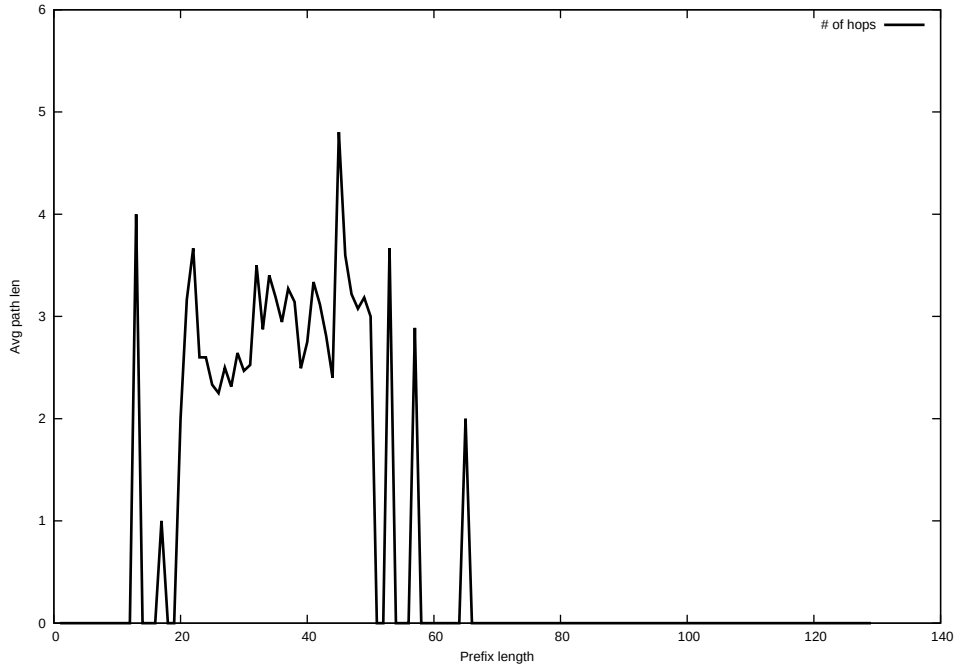
2012-09-02



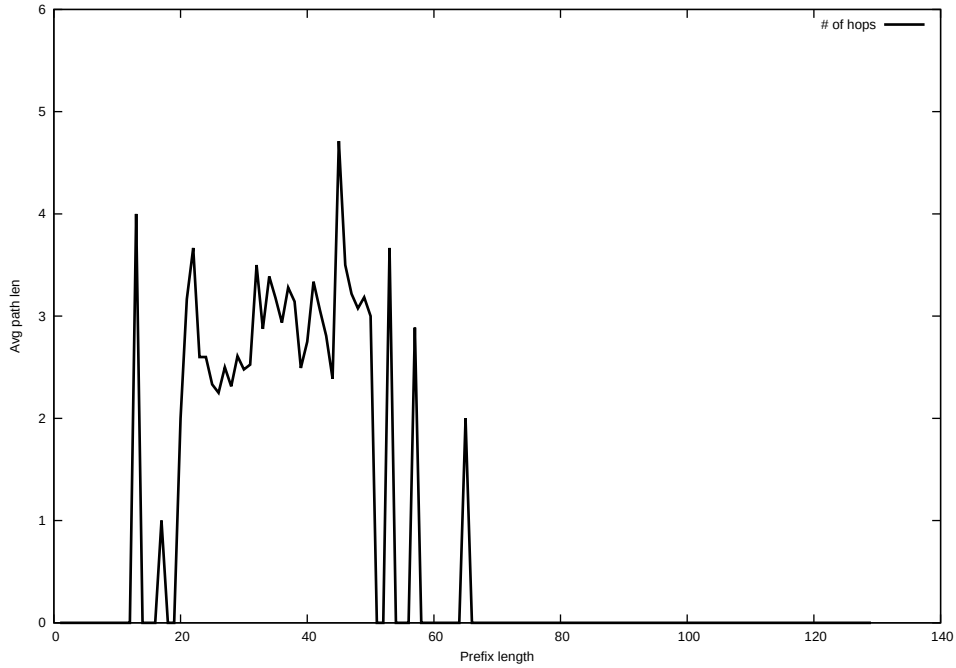
2012-09-03



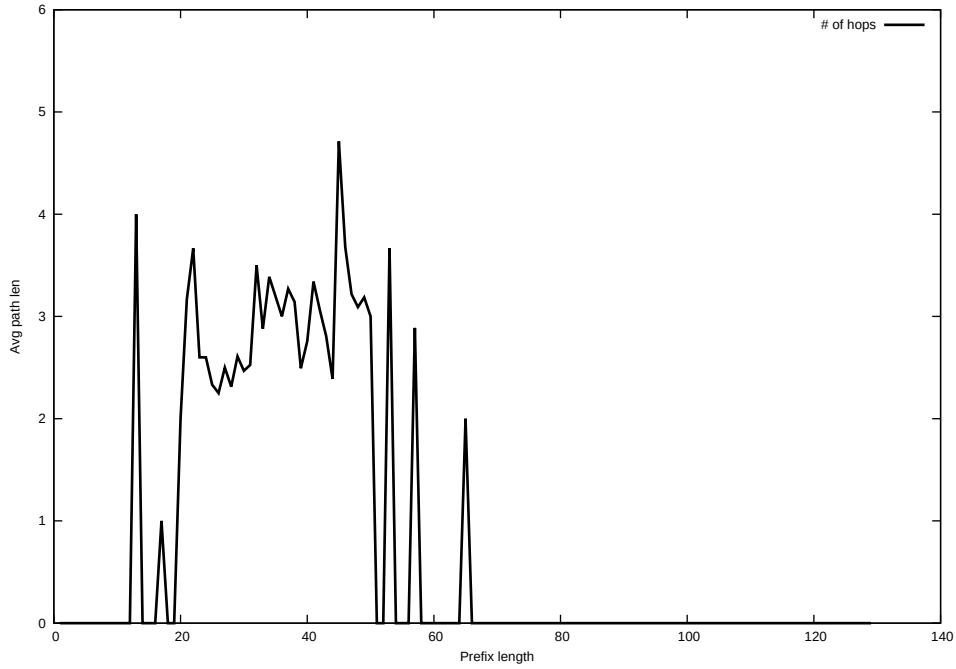
2012-09-04



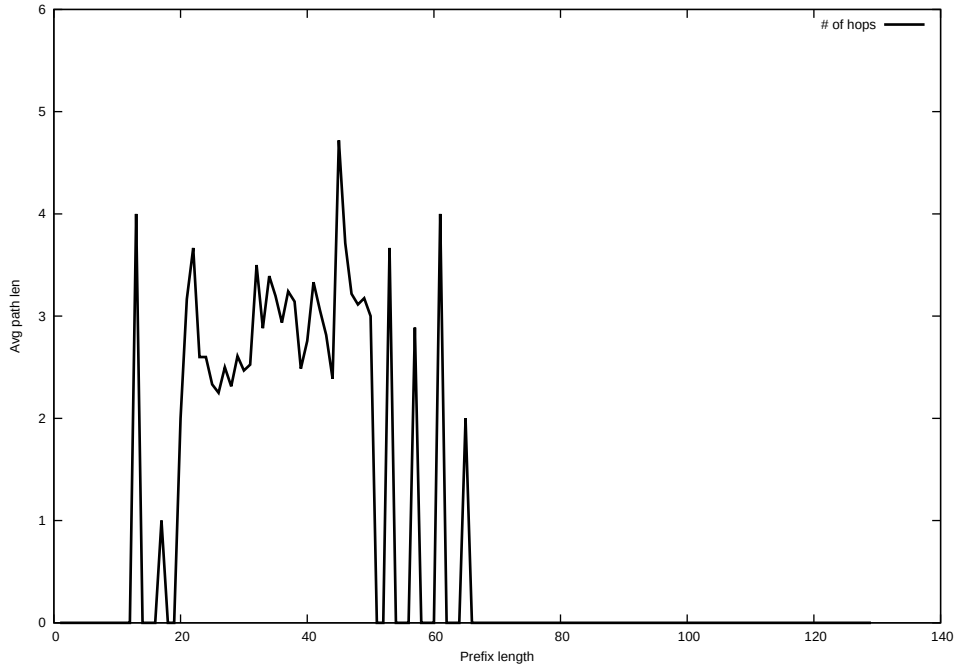
2012-09-05



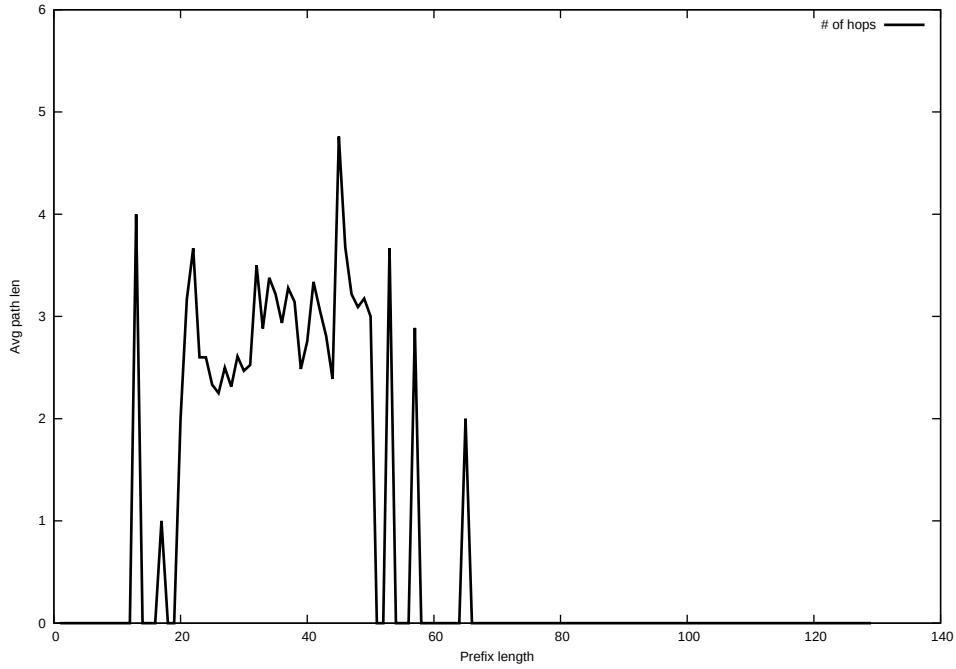
2012-09-06



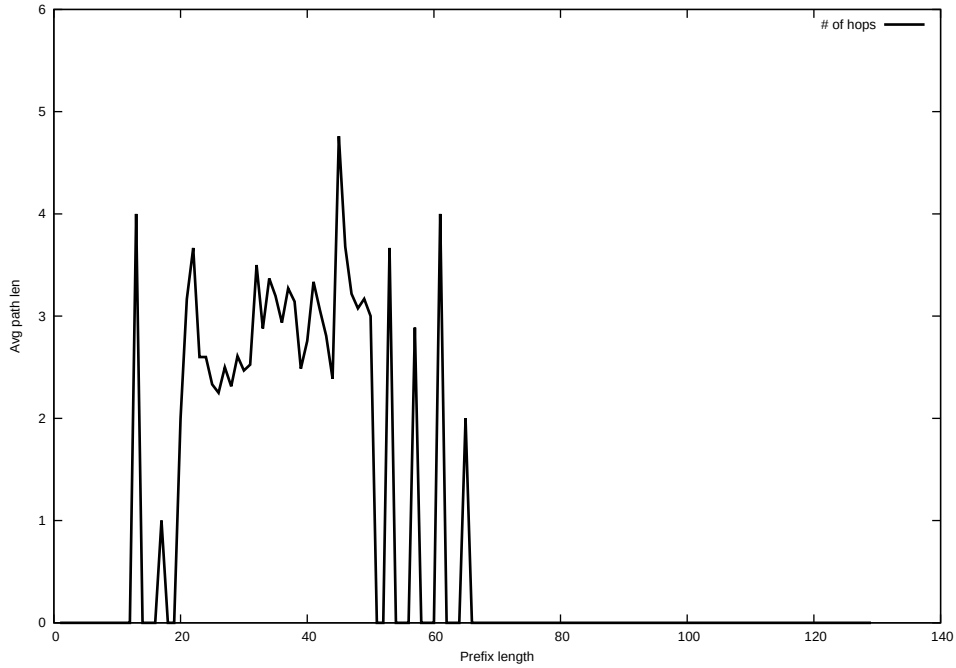
2012-09-07



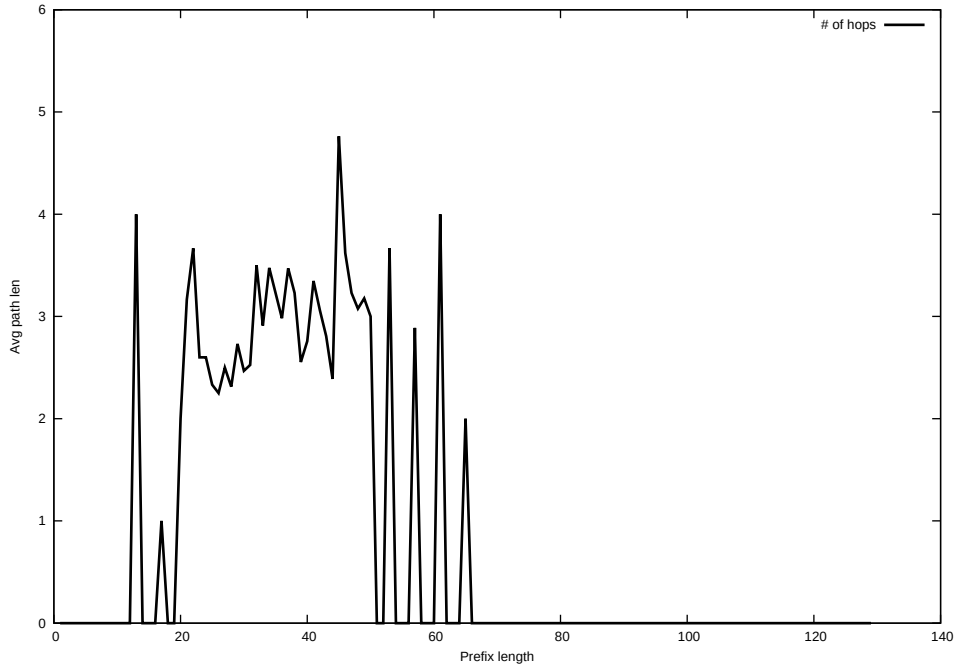
2012-09-08



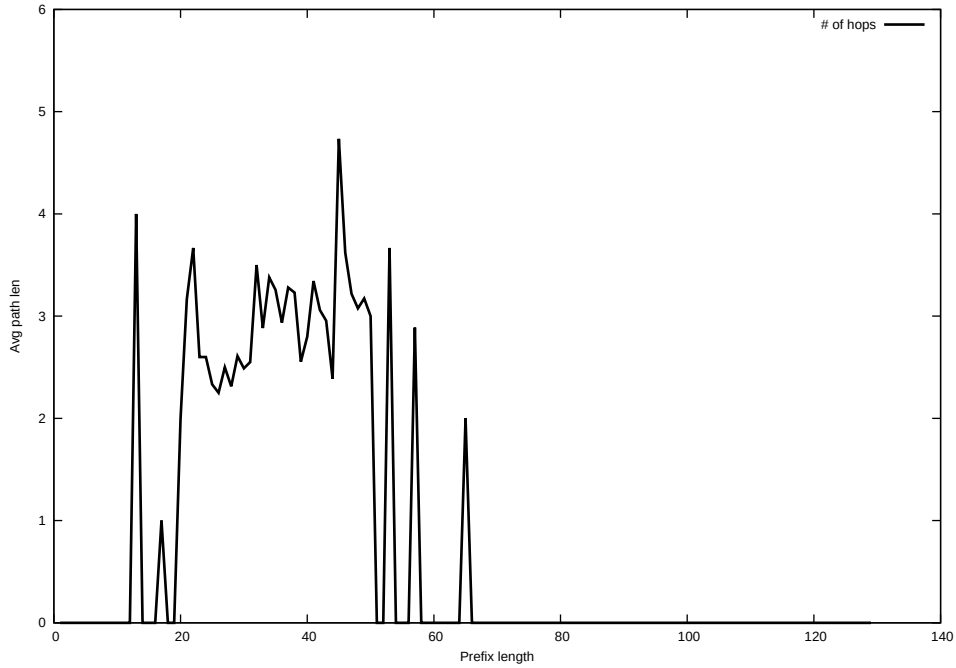
2012-09-09



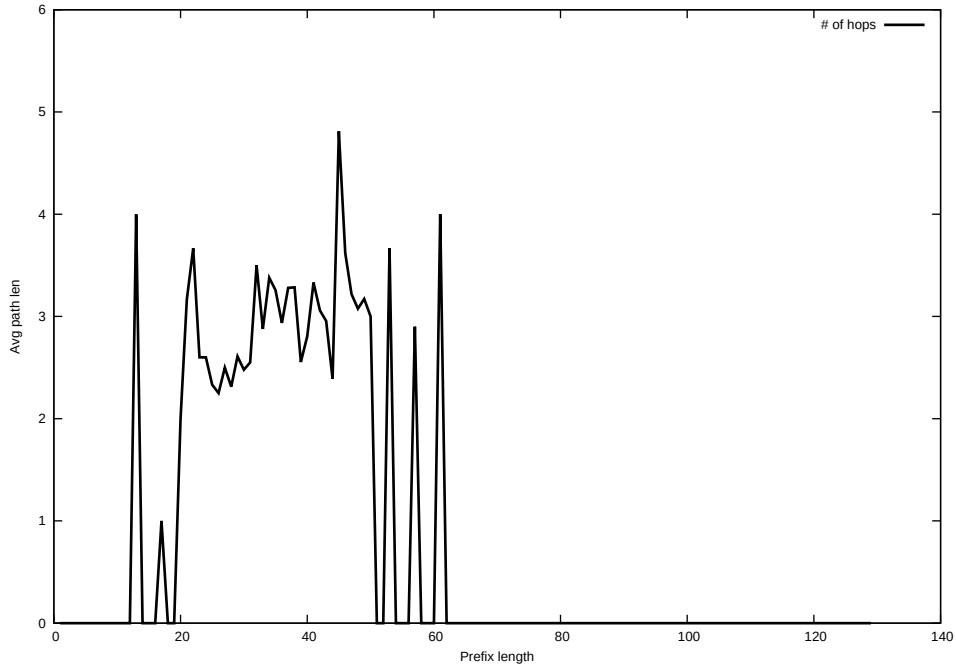
2012-09-10



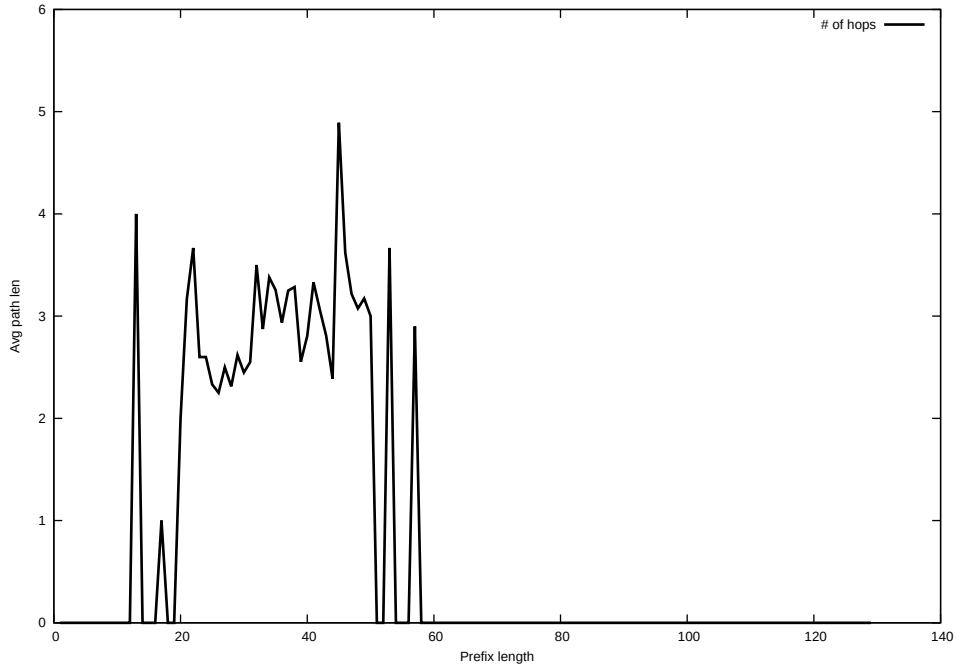
2012-09-11



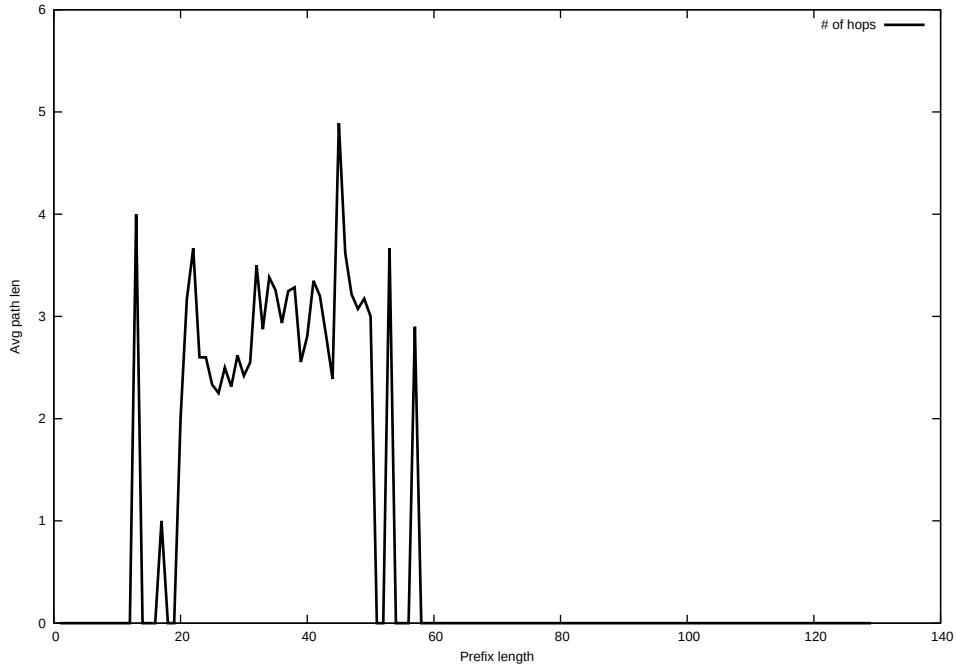
2012-09-12



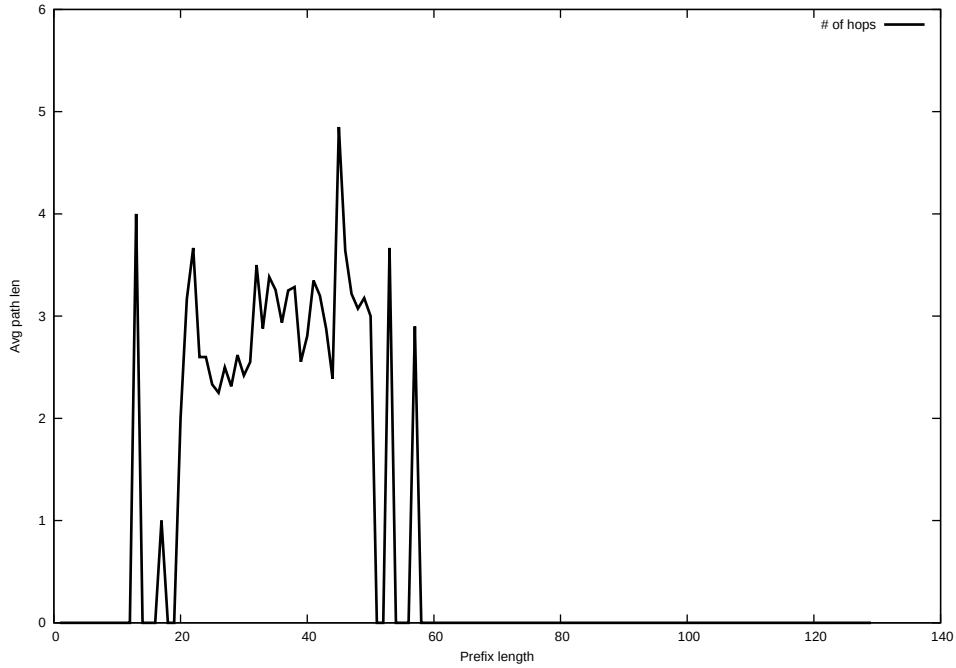
2012-09-13



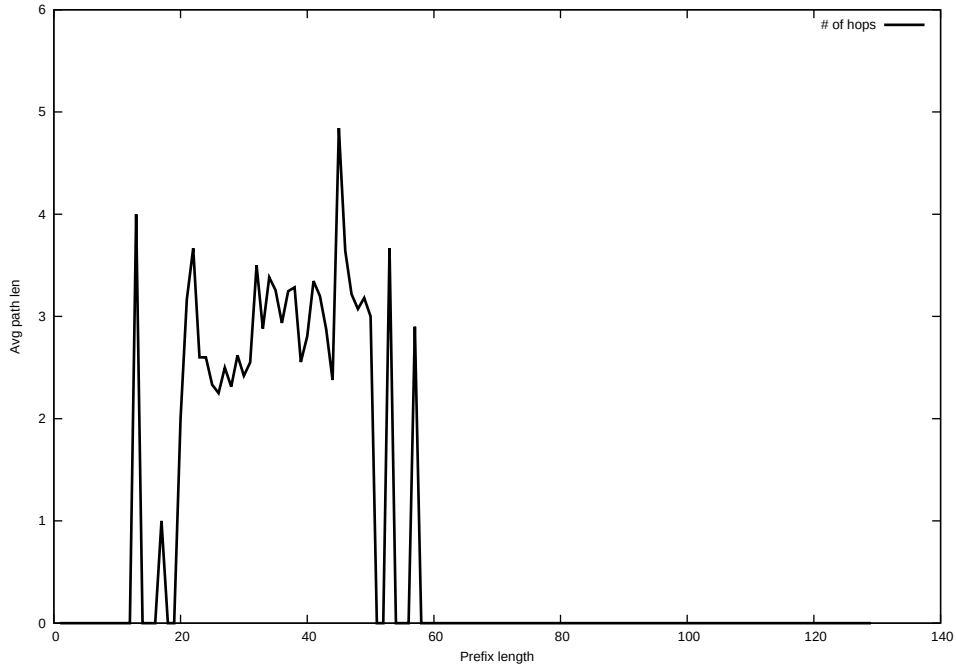
2012-09-14



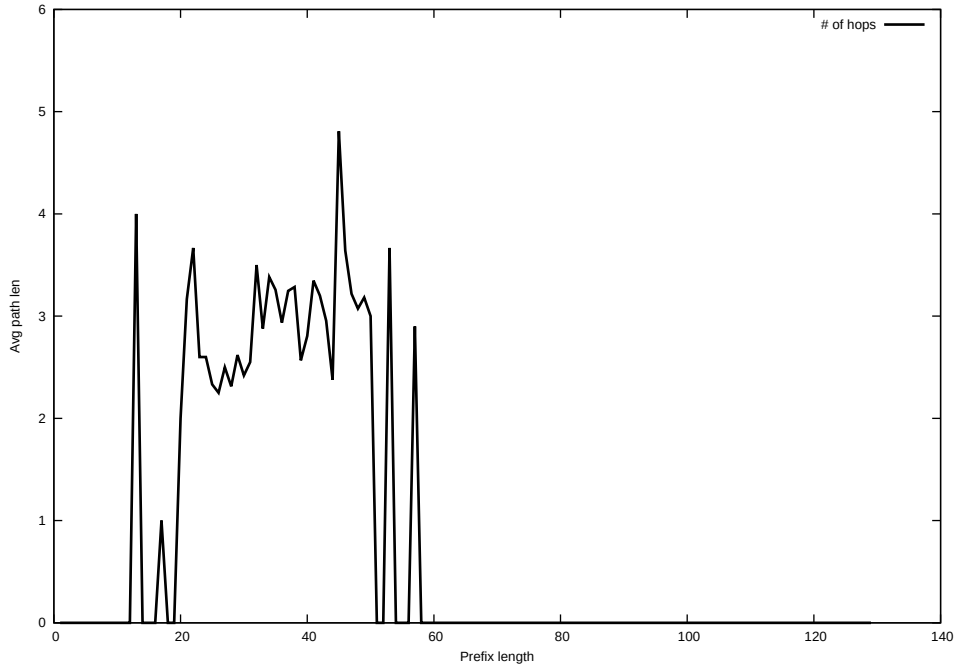
2012-09-15



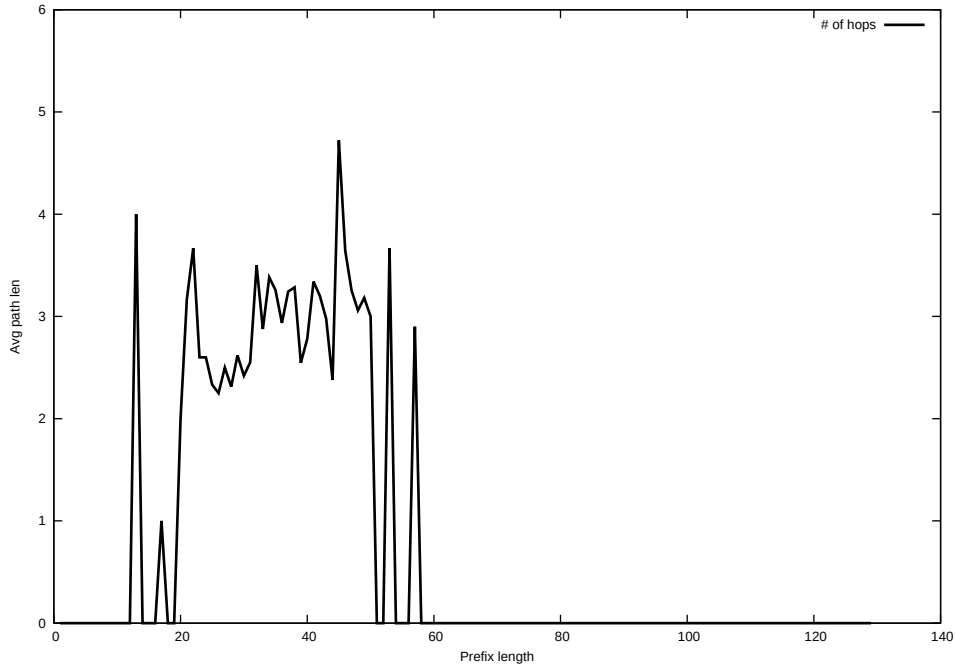
2012-09-16



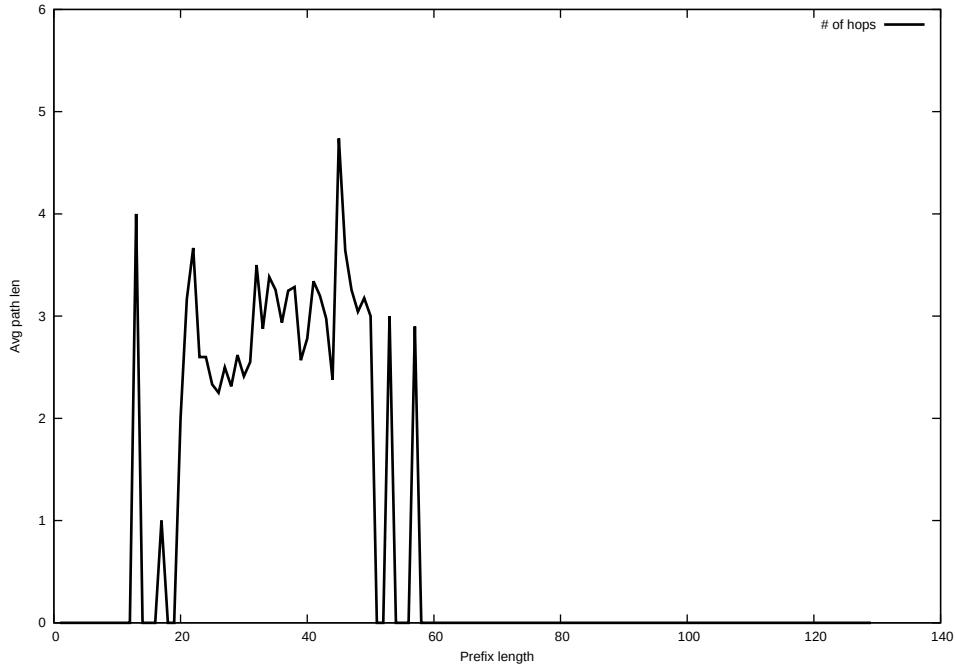
2012-09-17



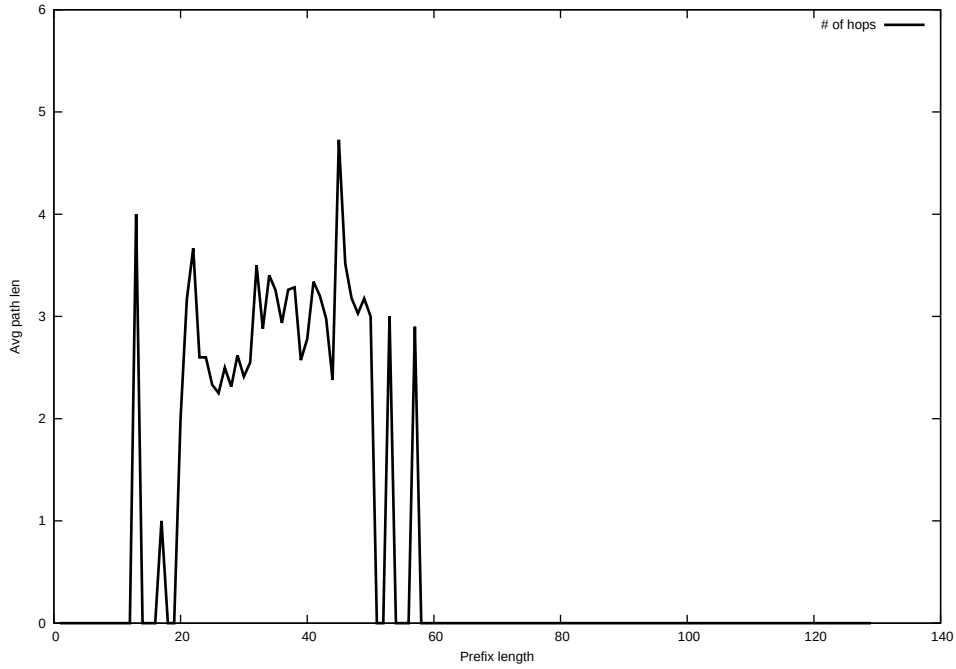
2012-09-18



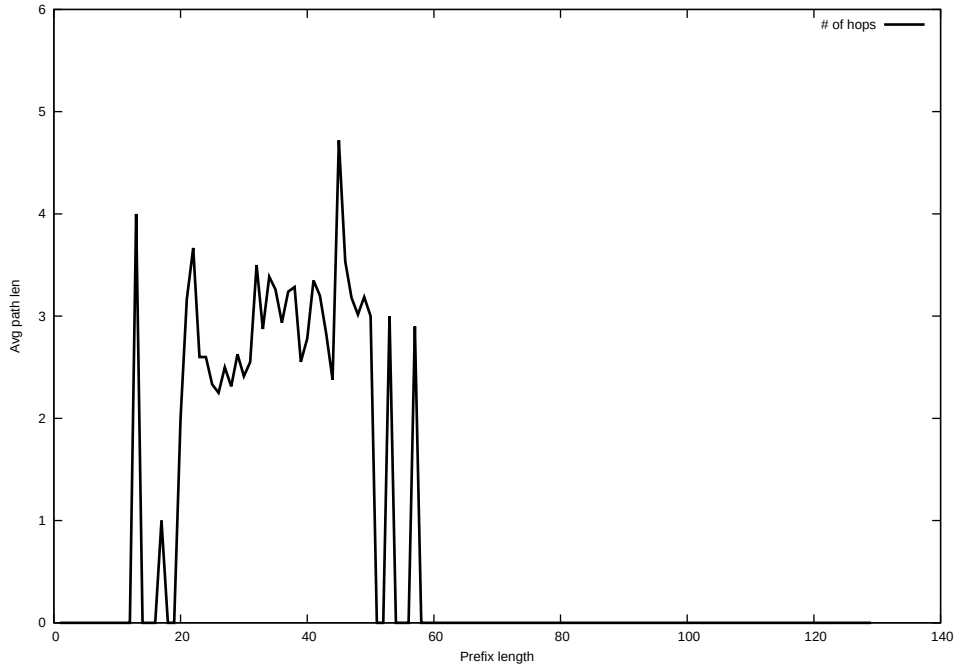
2012-09-19



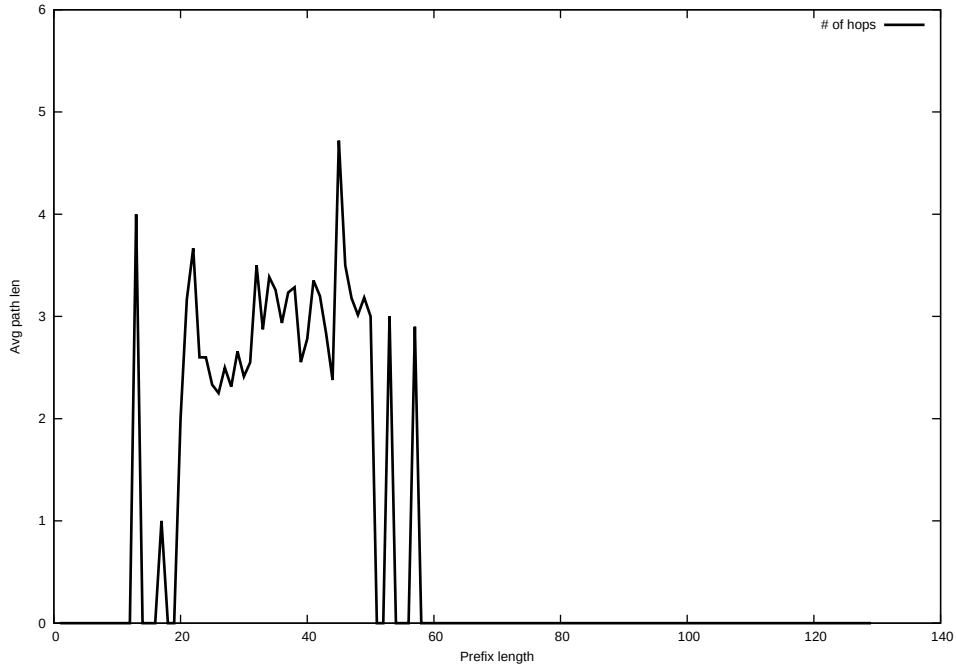
2012-09-20



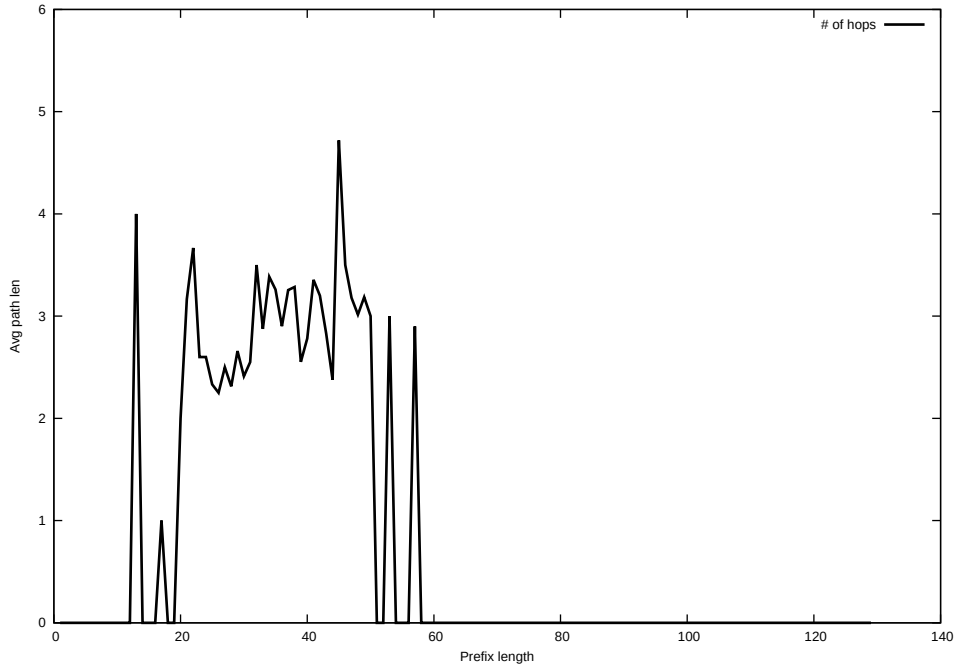
2012-09-21



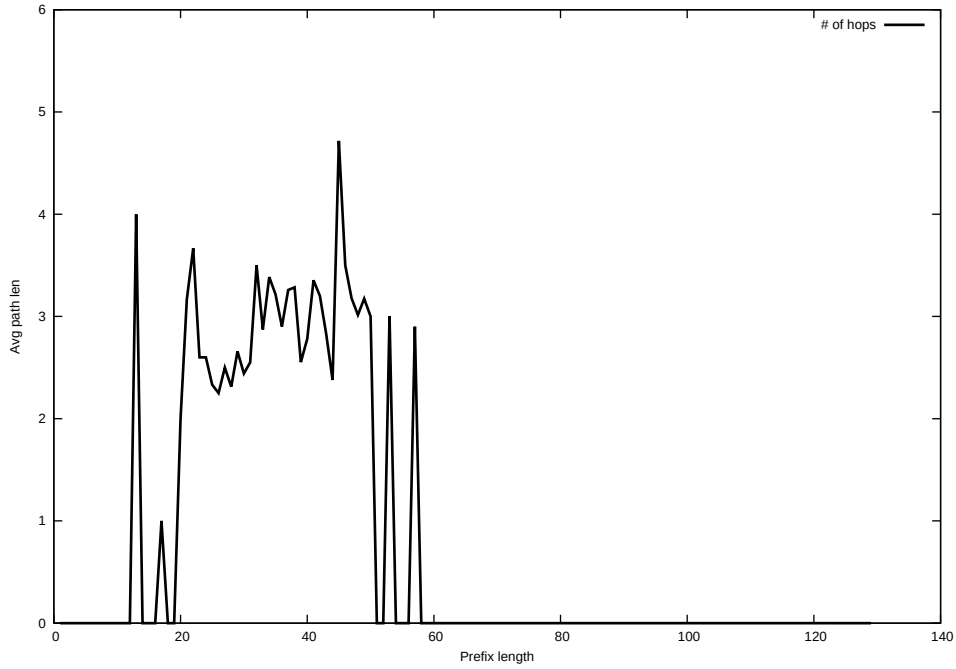
2012-09-22



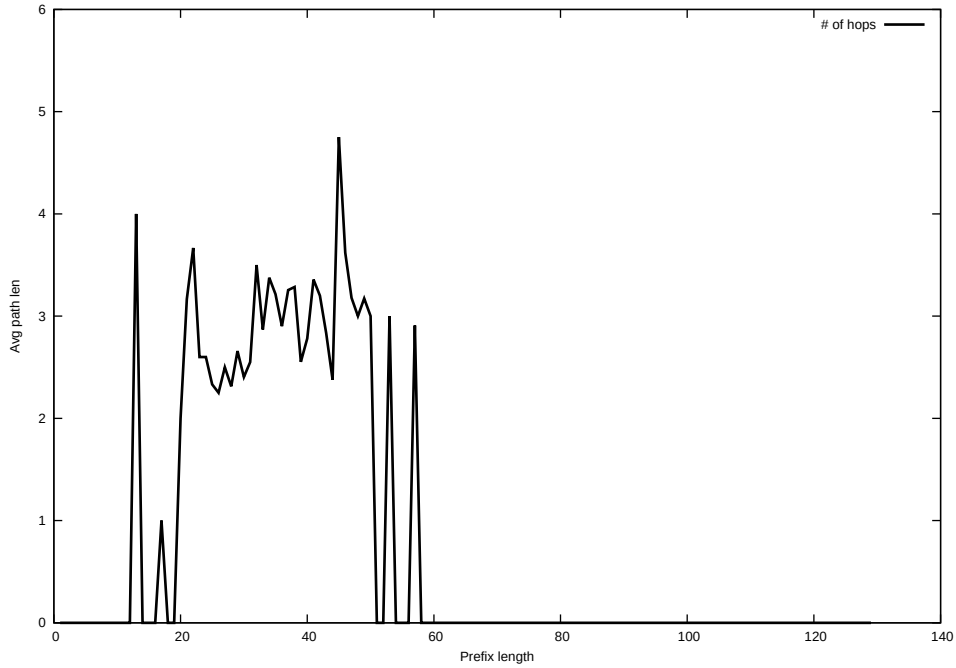
2012-09-23



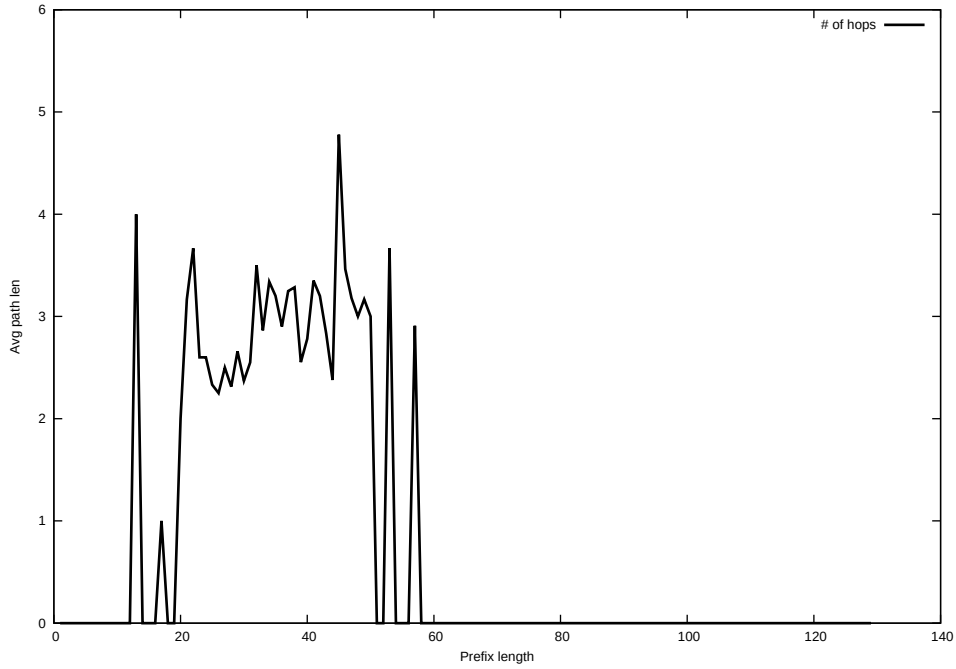
2012-09-24



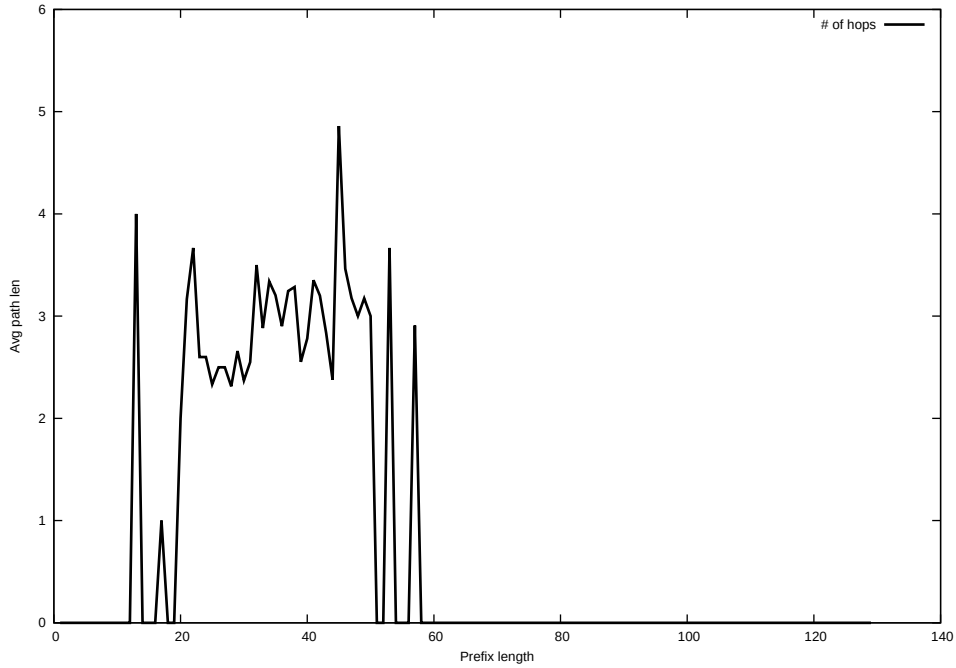
2012-09-25



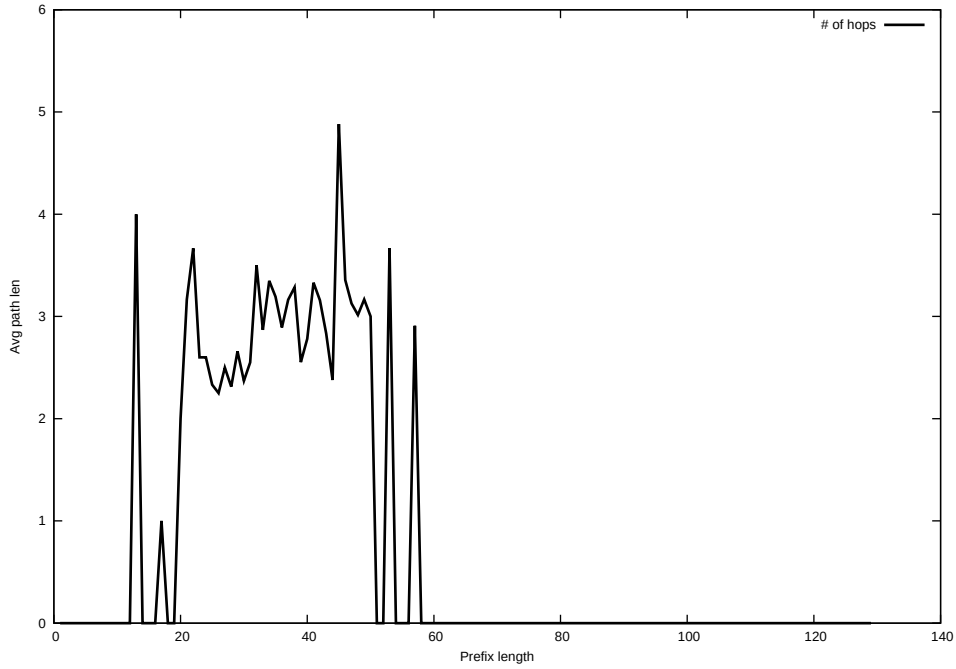
2012-09-26



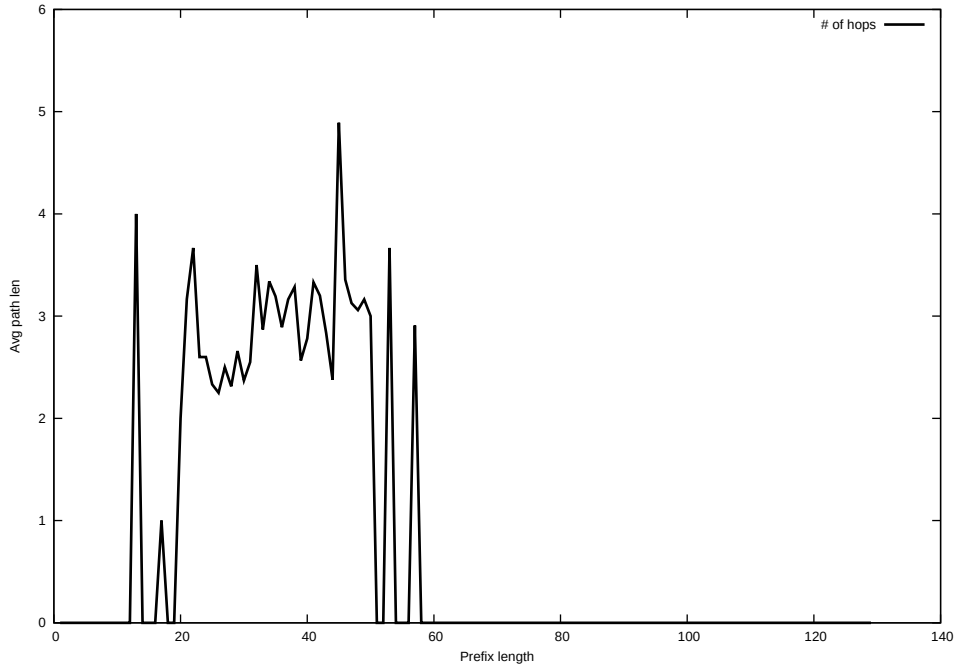
2012-09-27



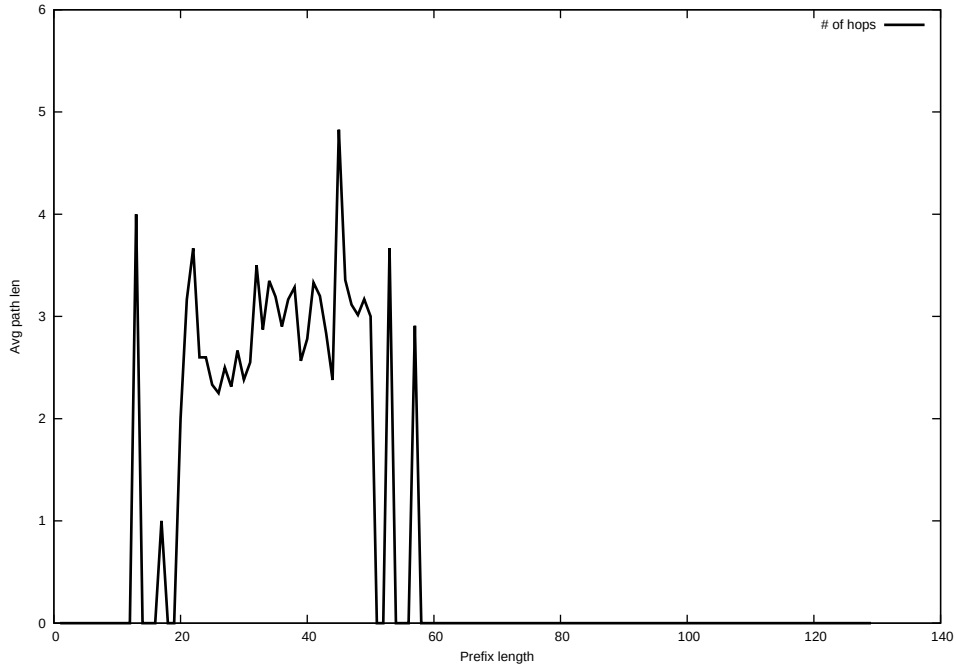
2012-09-28



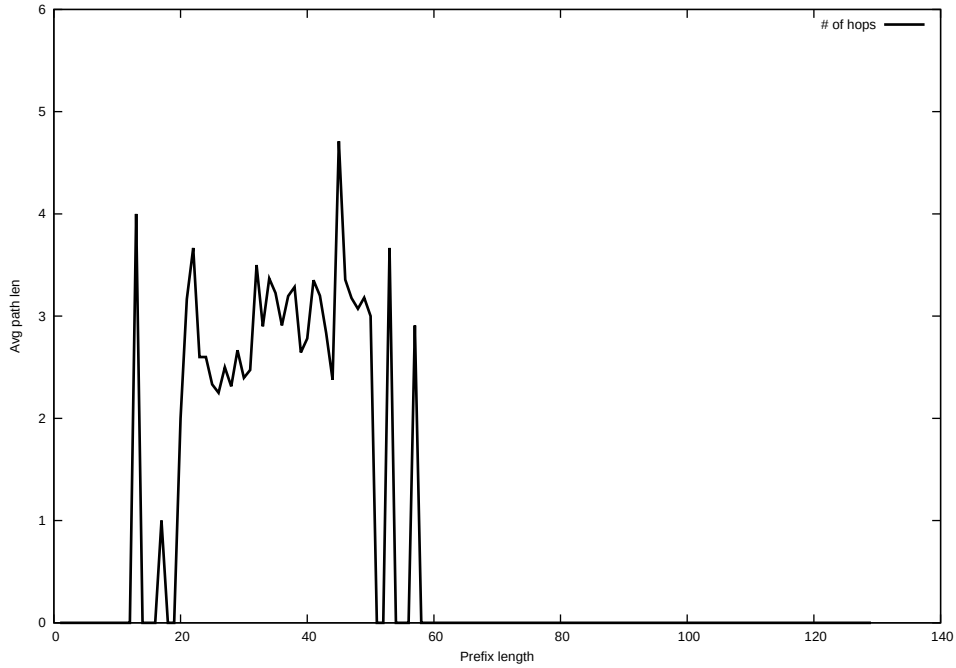
2012-09-29



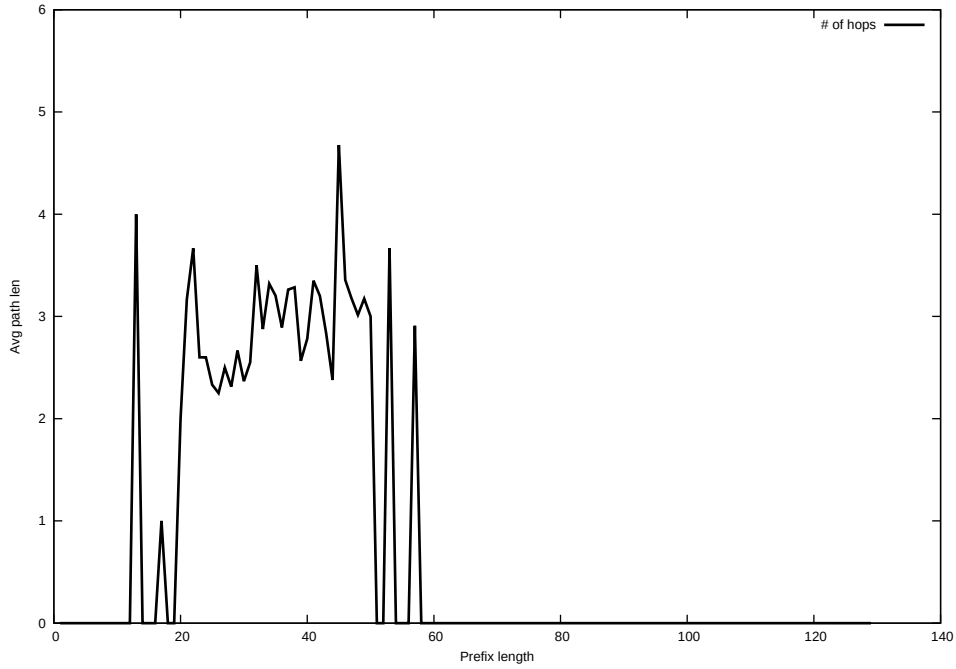
2012-09-30



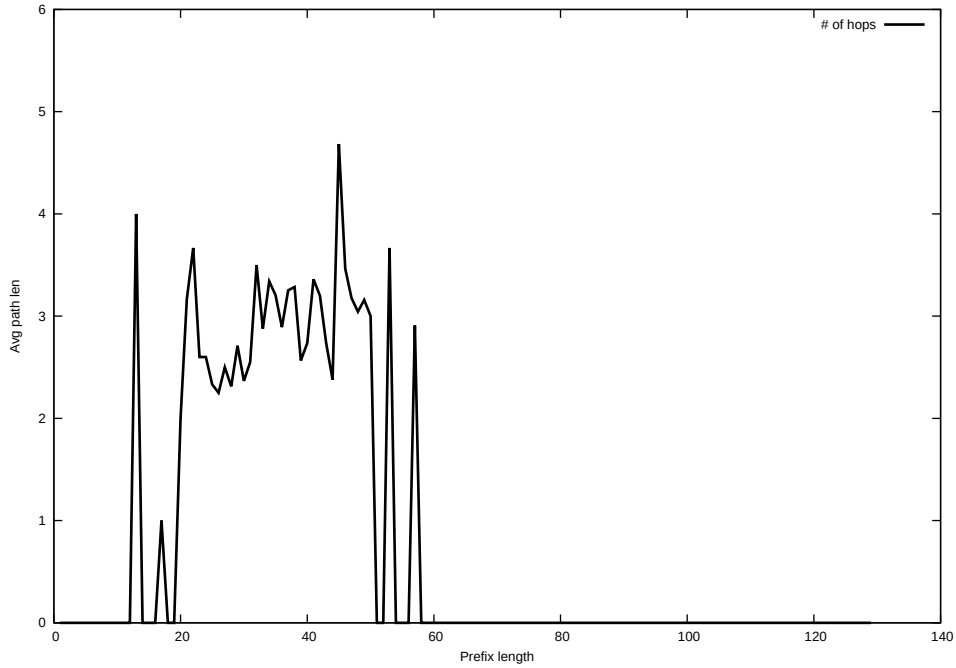
2012-10-01



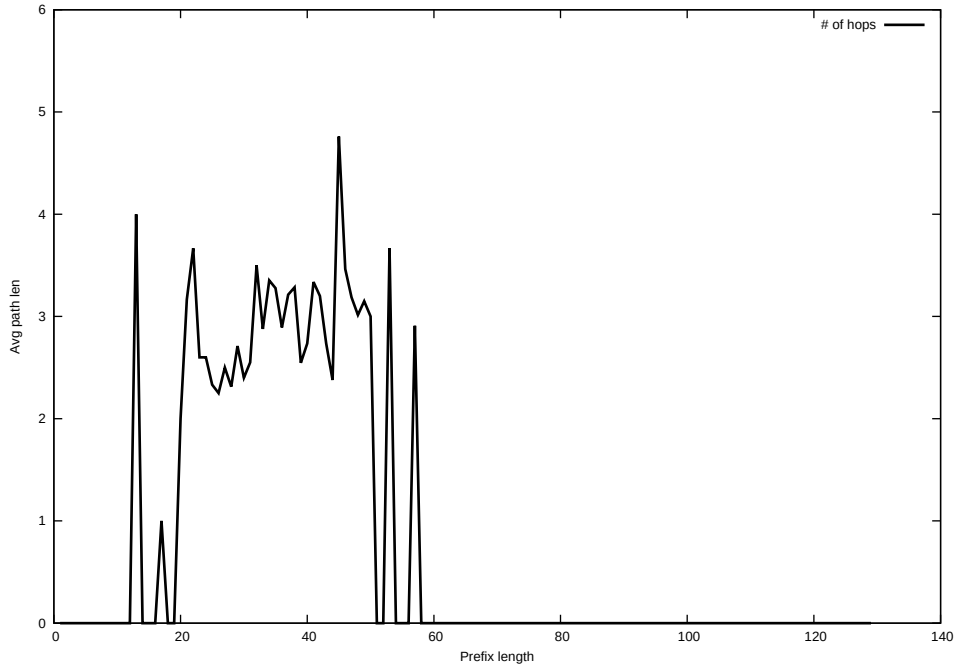
2012-10-02



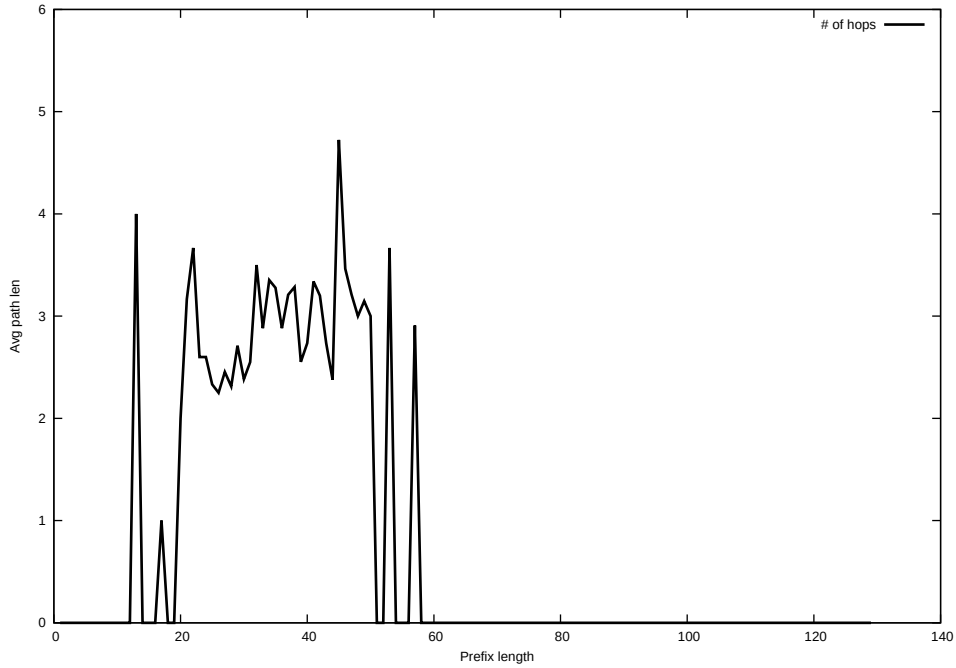
2012-10-03



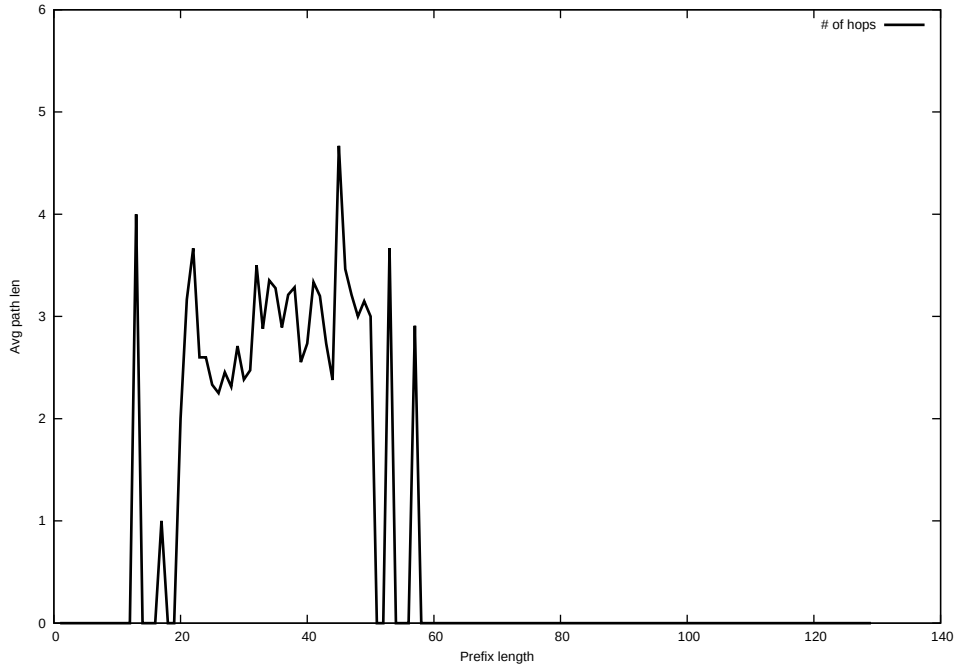
2012-10-04



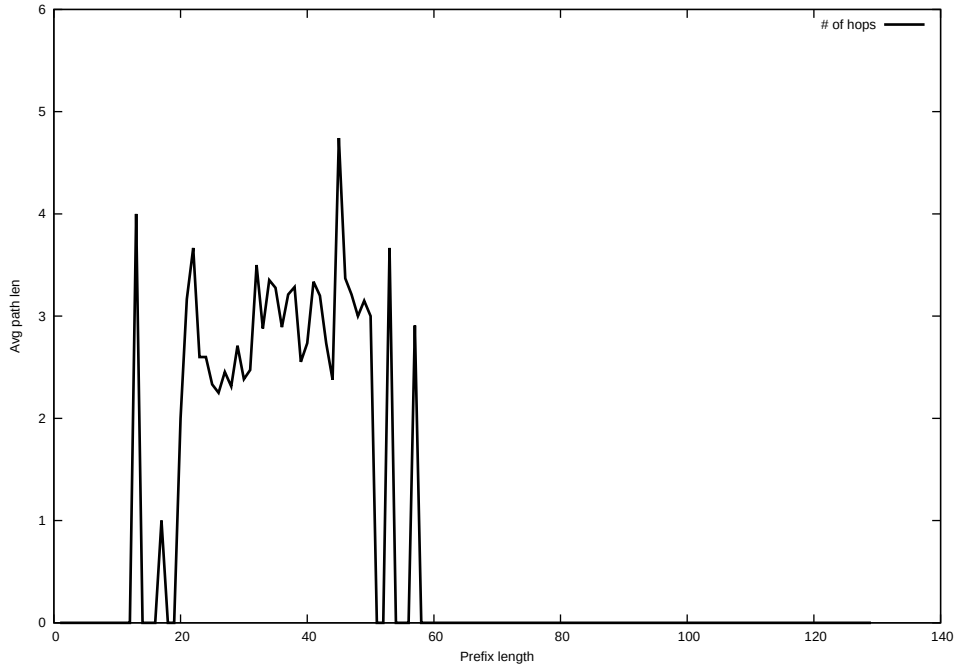
2012-10-05



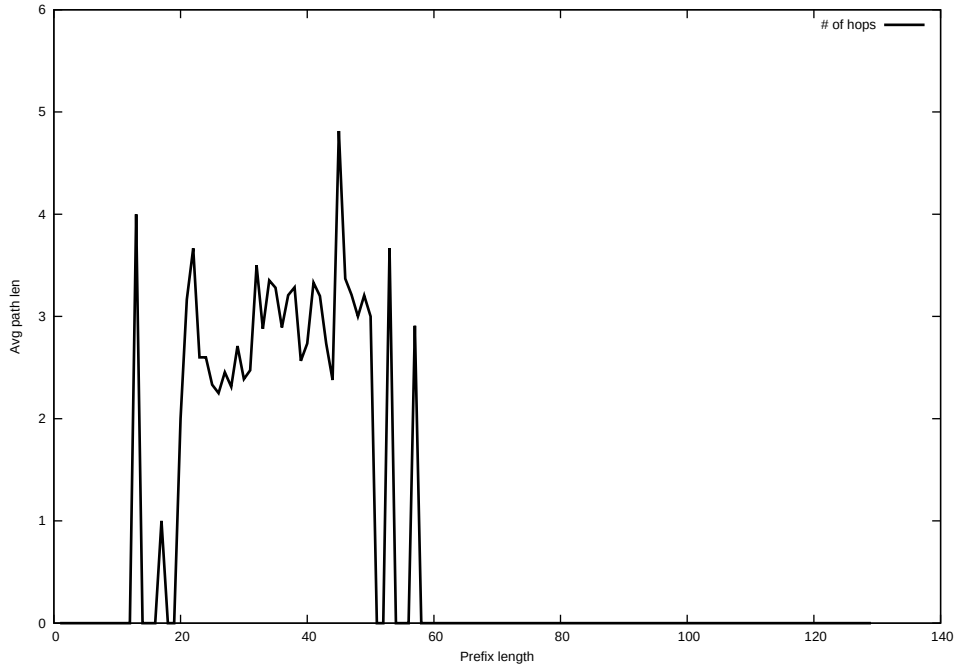
2012-10-06



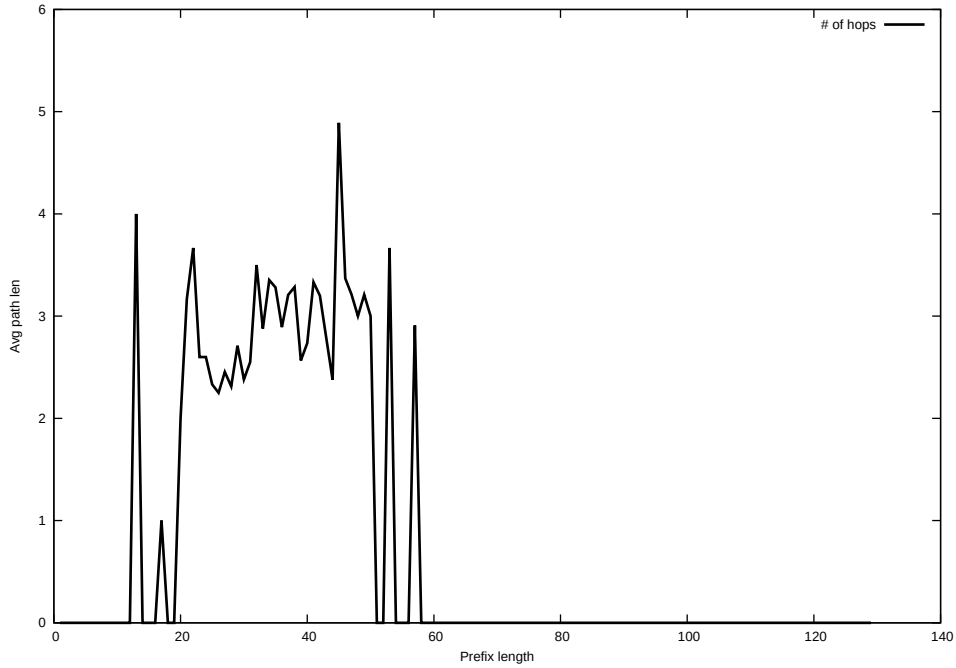
2012-10-07



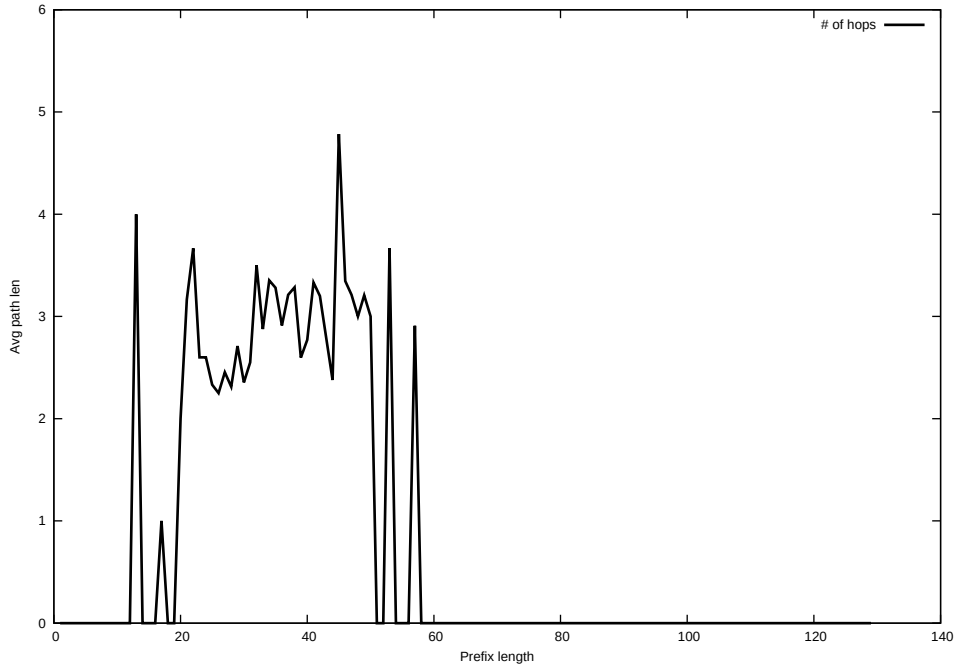
2012-10-08



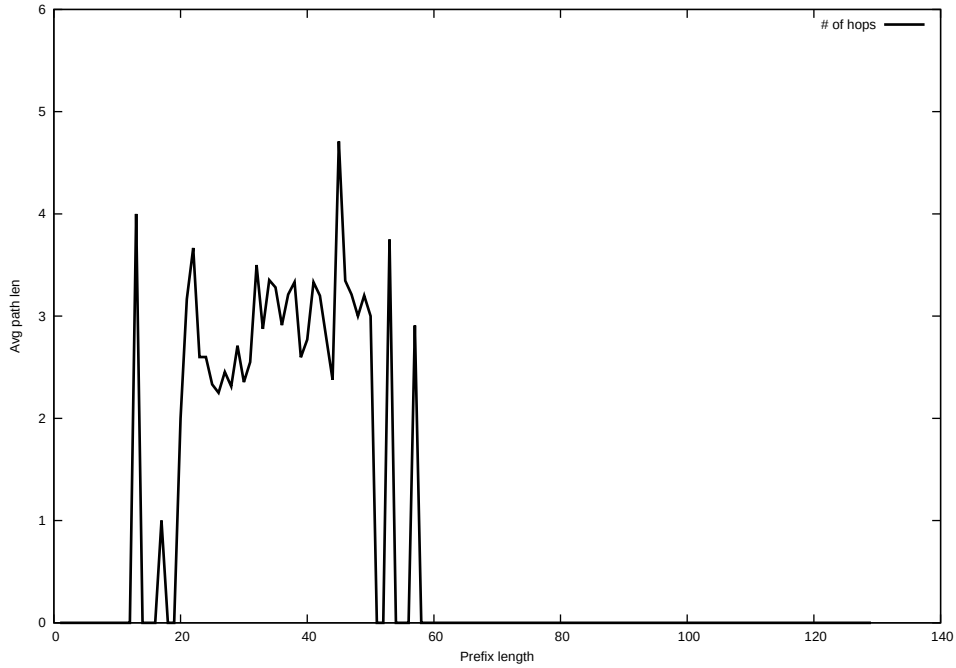
2012-10-09



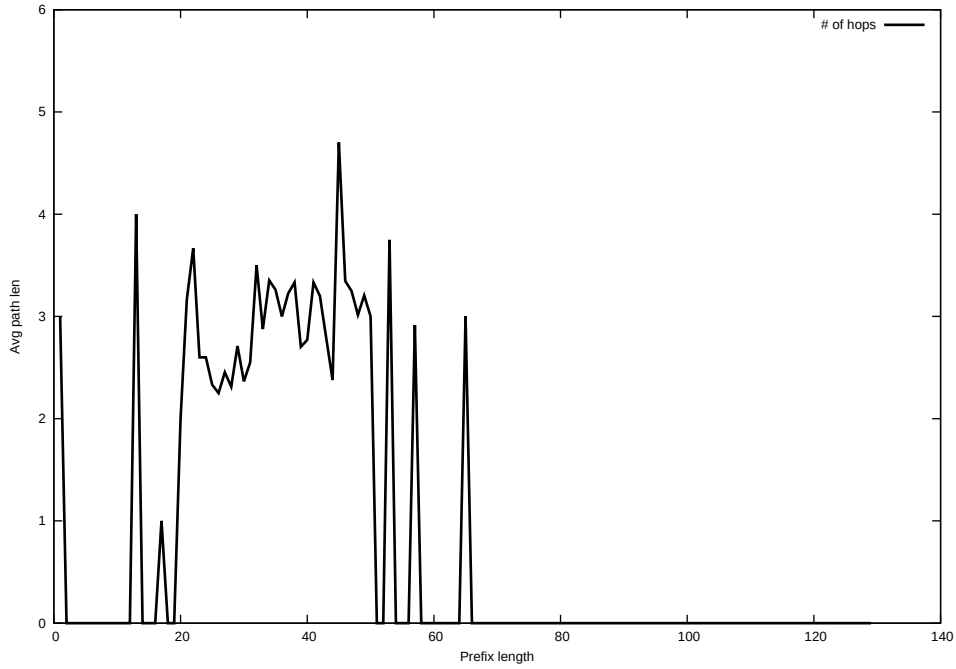
2012-10-10



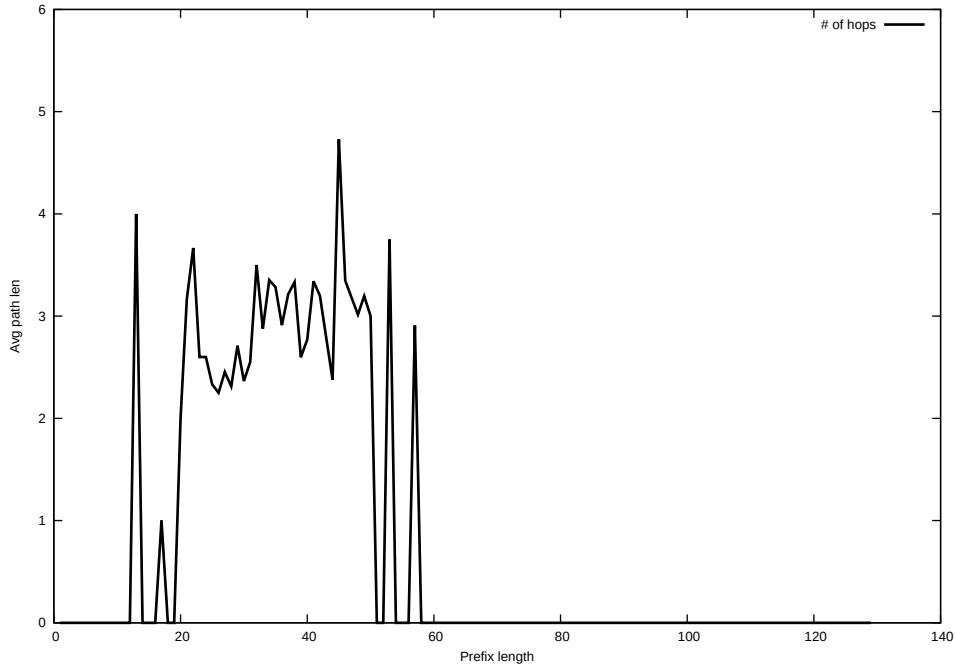
2012-10-11



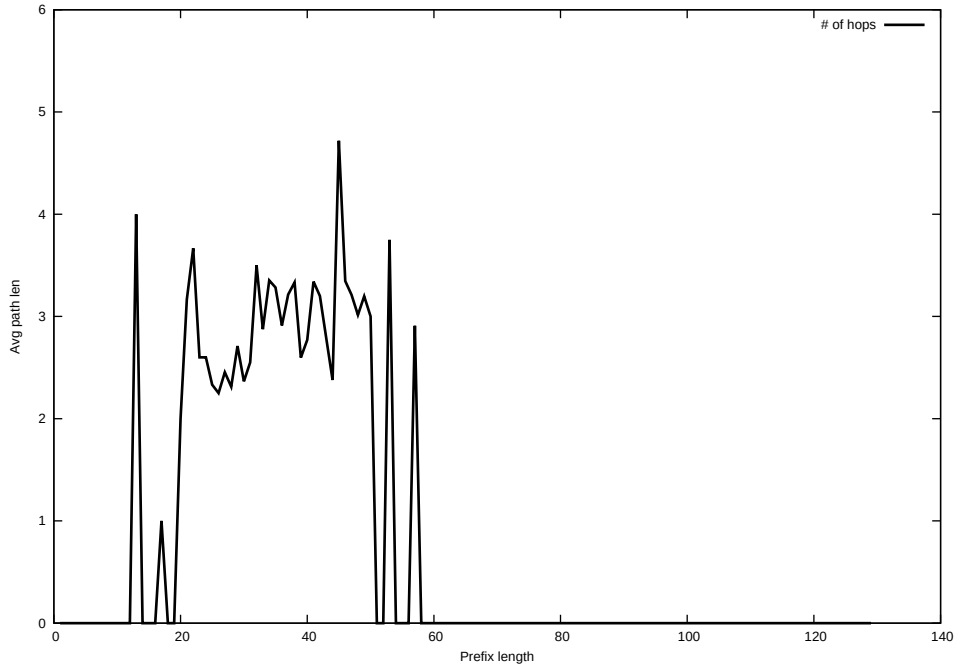
2012-10-12



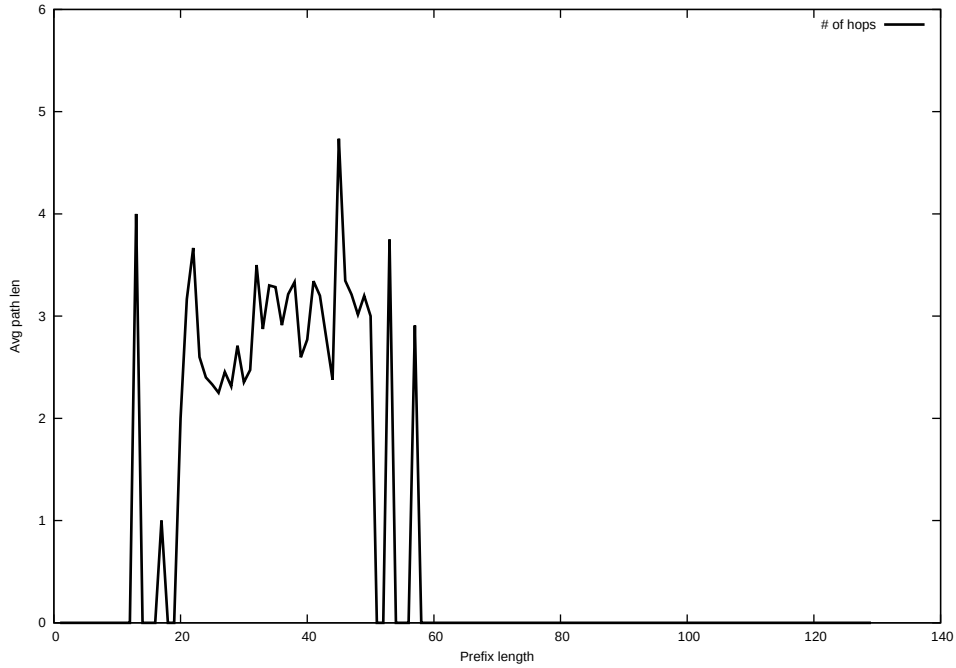
2012-10-13



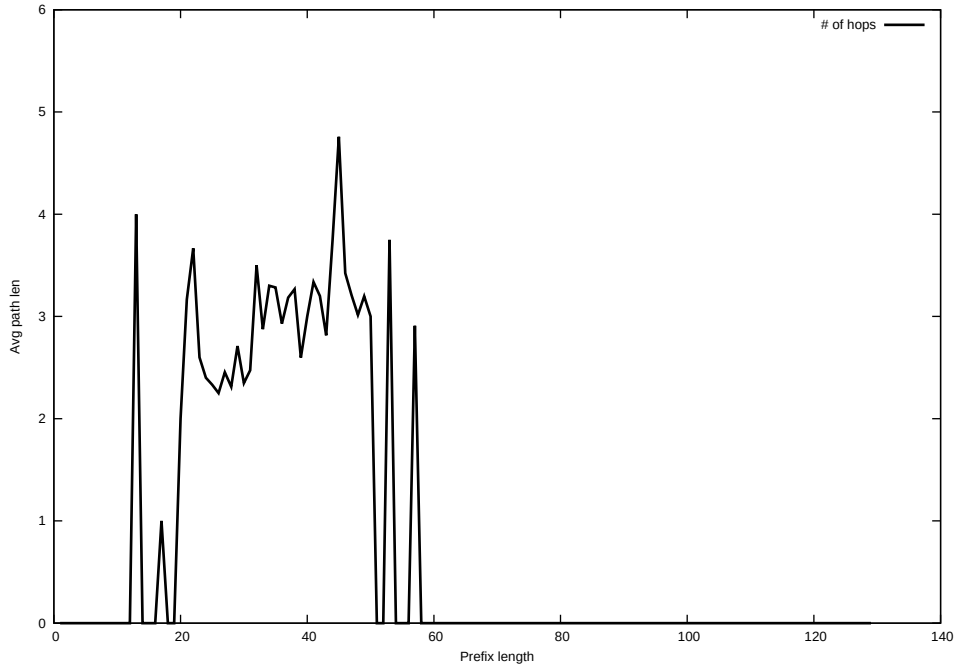
2012-10-14



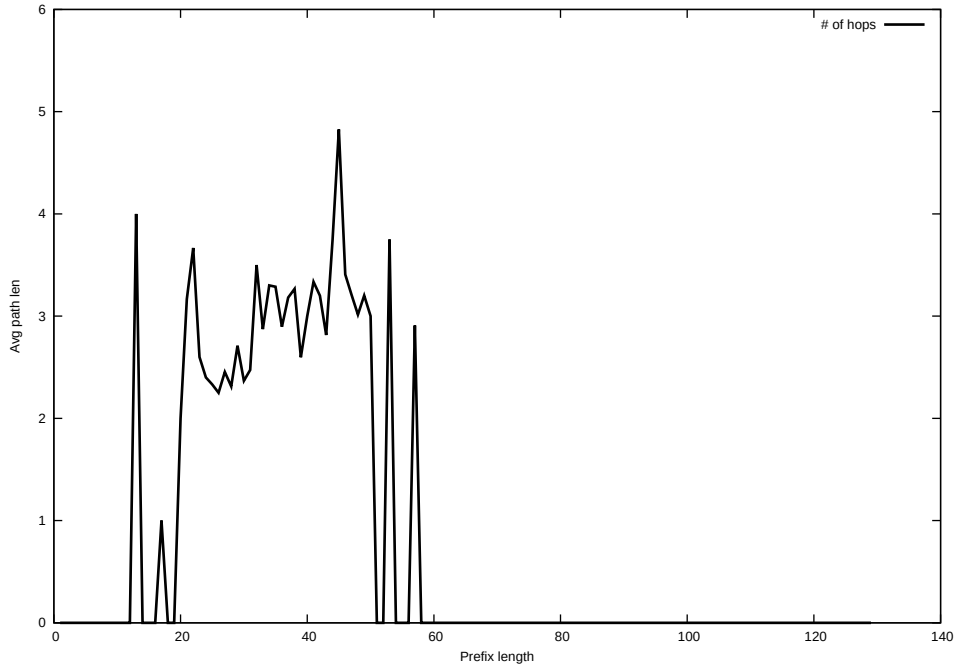
2012-10-15



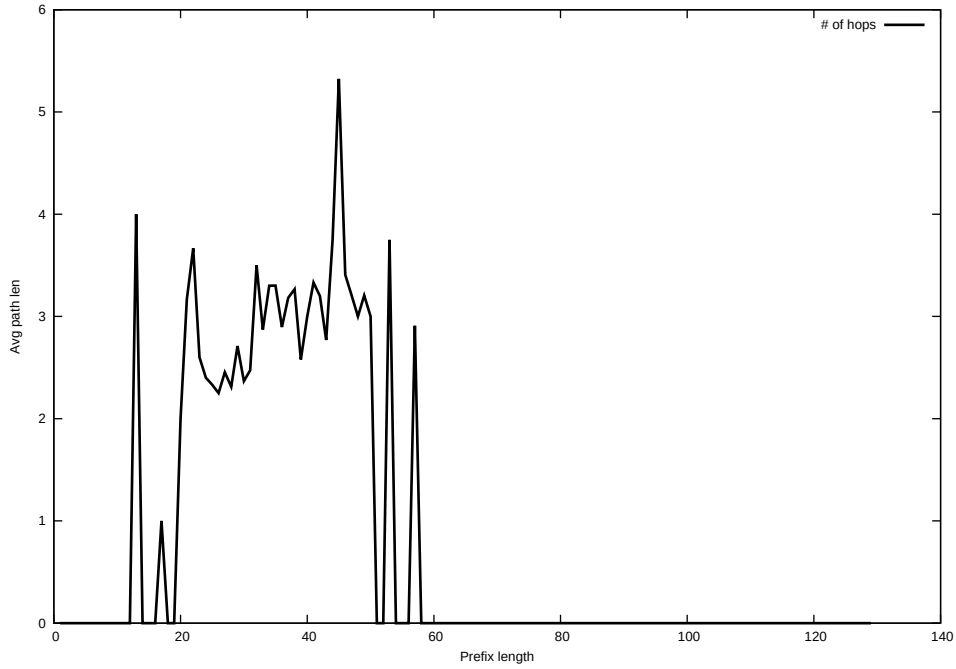
2012-10-16



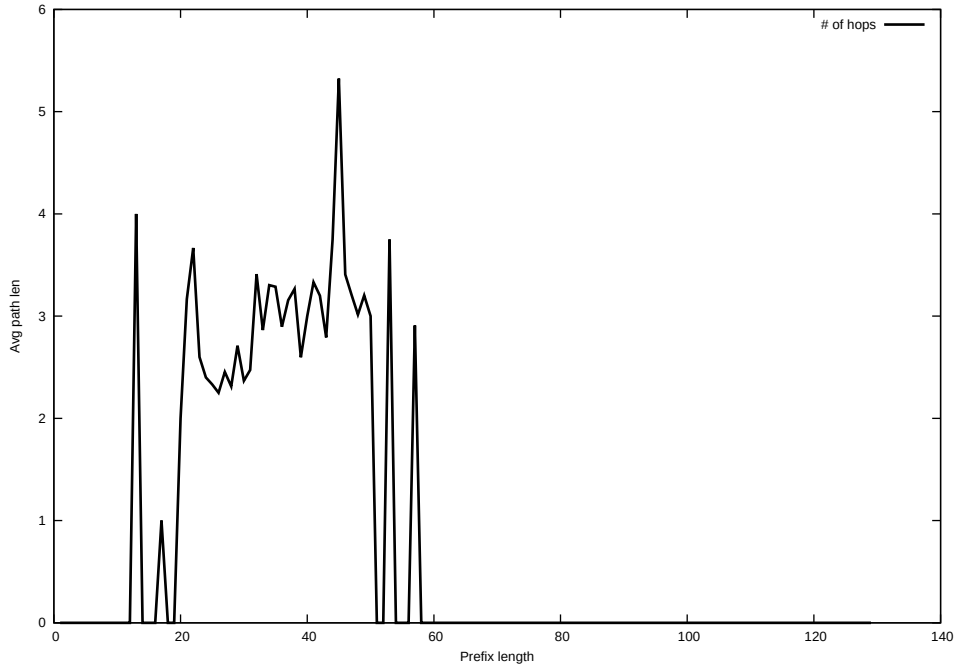
2012-10-18



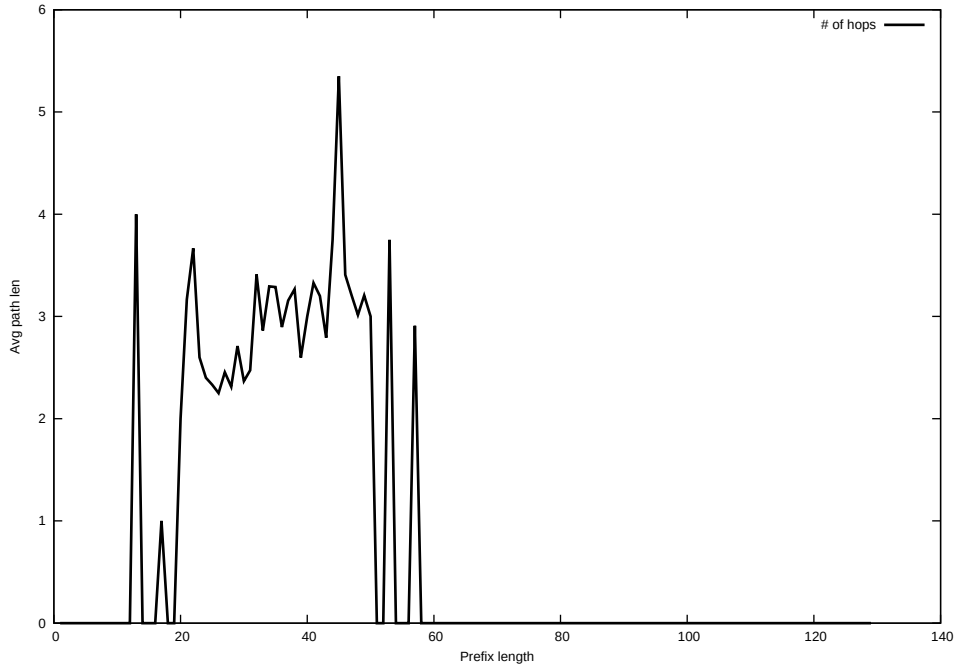
2012-10-19



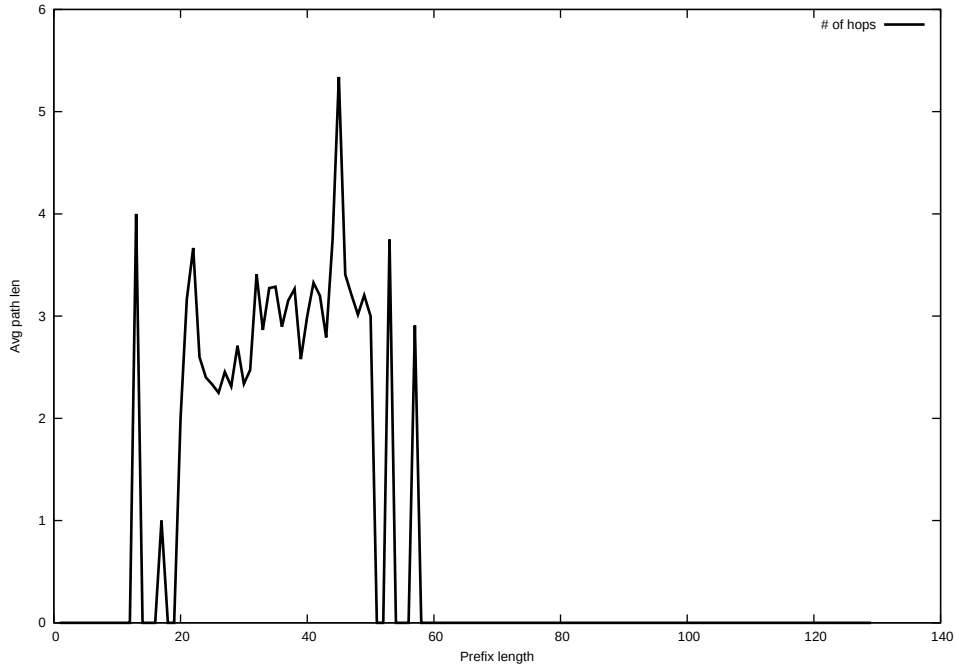
2012-10-20



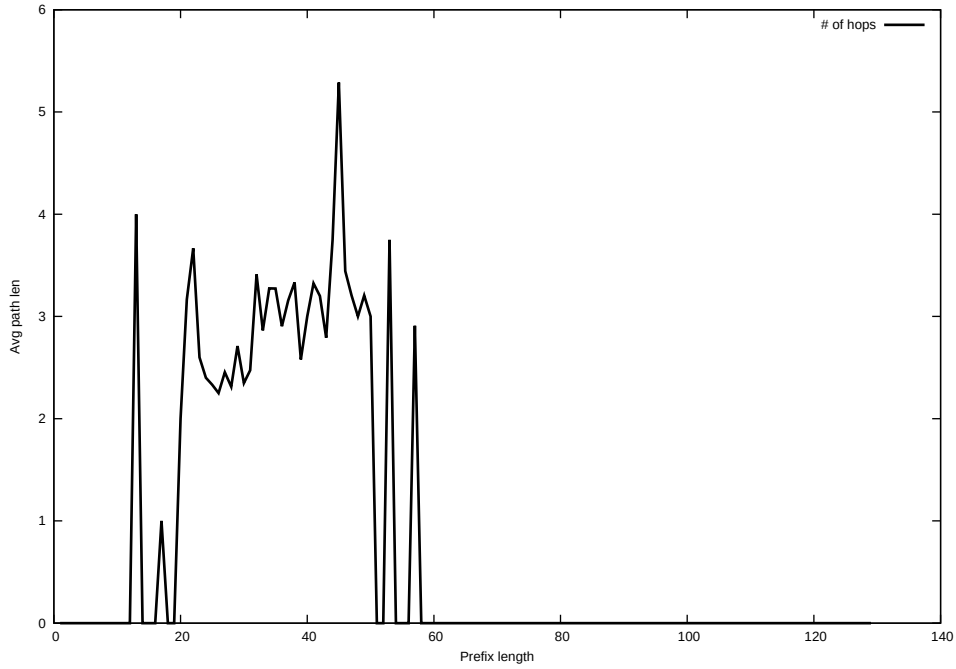
2012-10-21



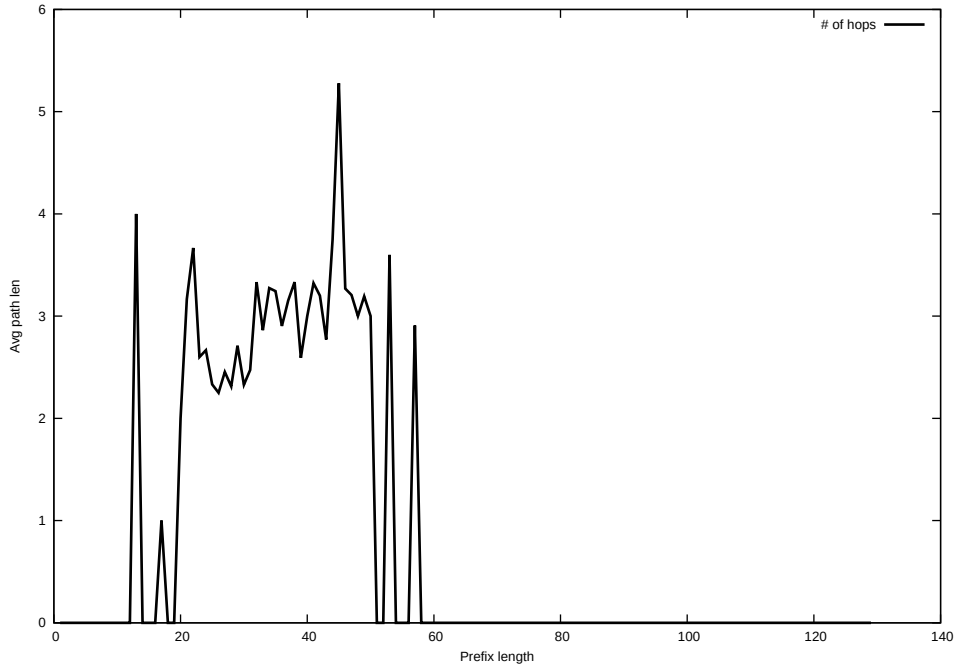
2012-10-22



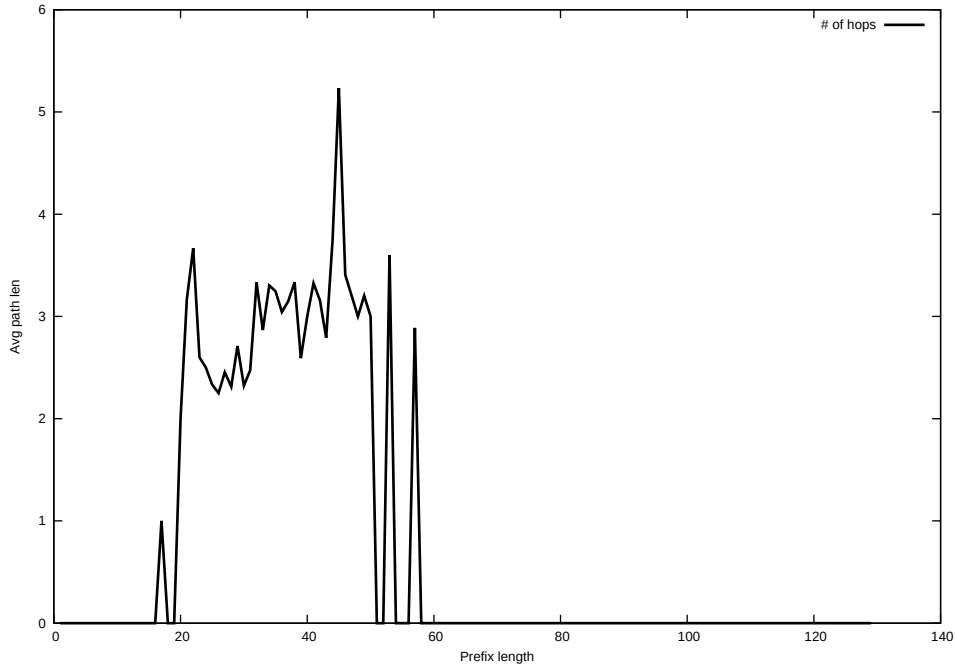
2012-10-23



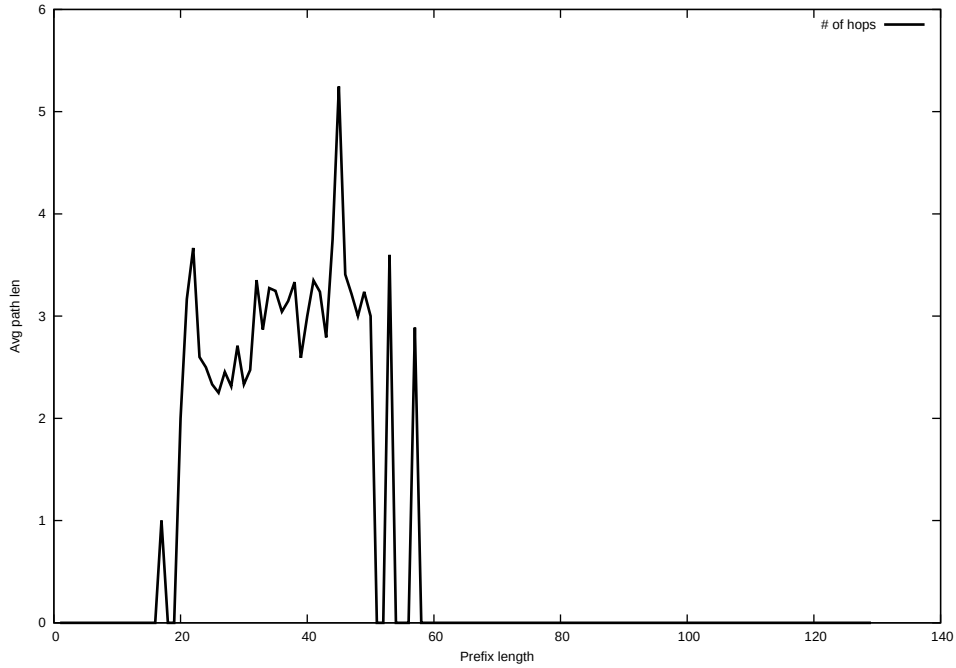
2012-10-24



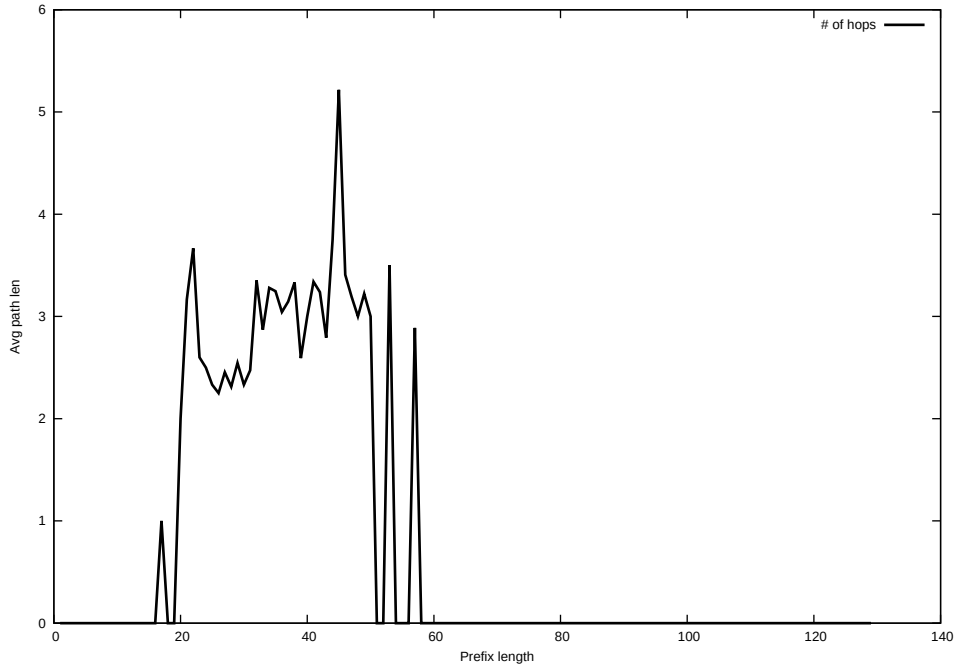
2012-10-25



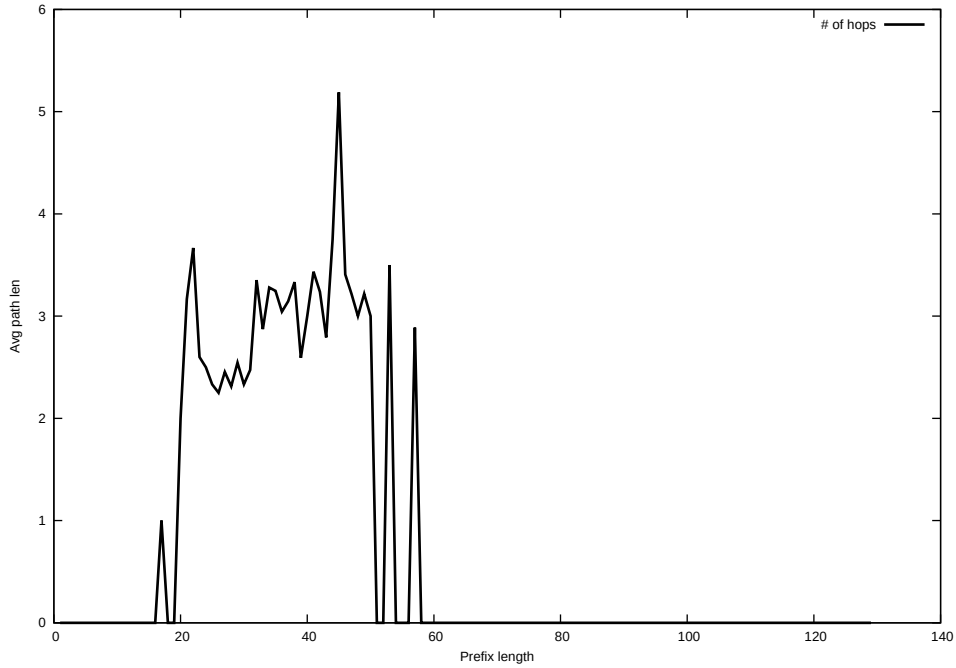
2012-10-26



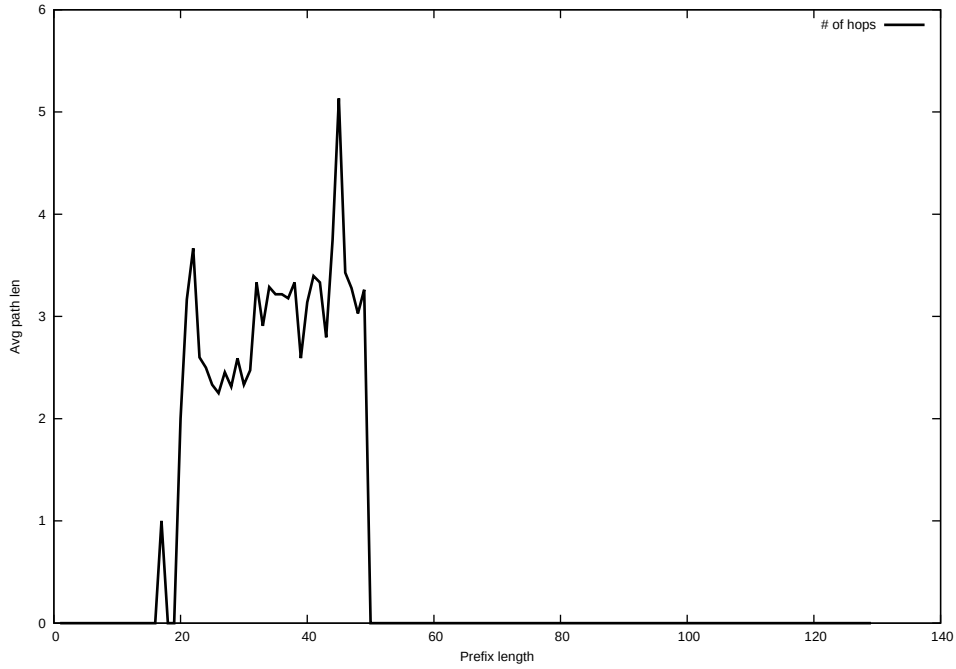
2012-10-27



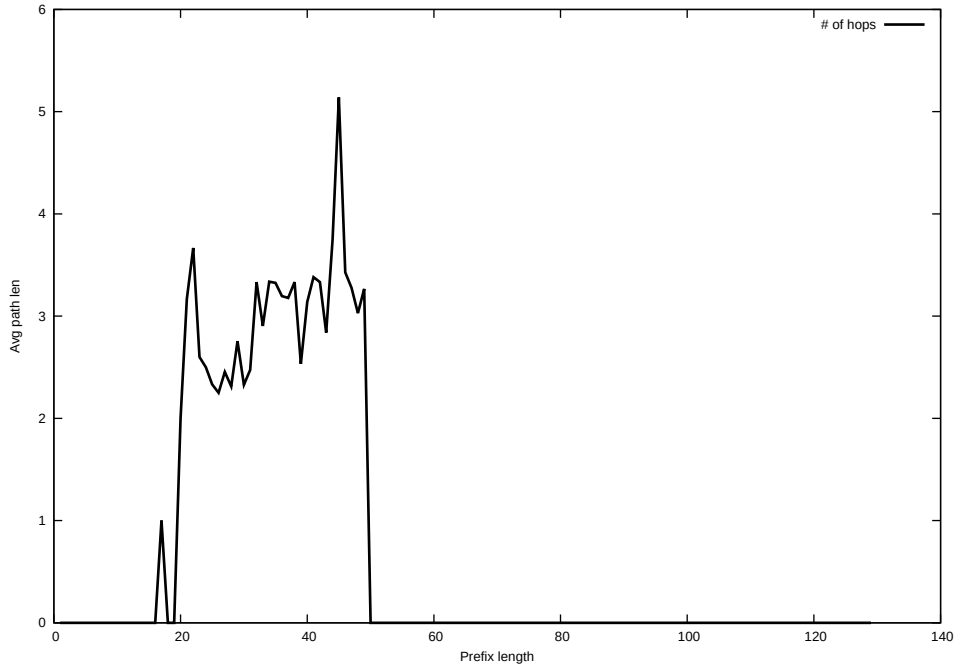
2012-10-28



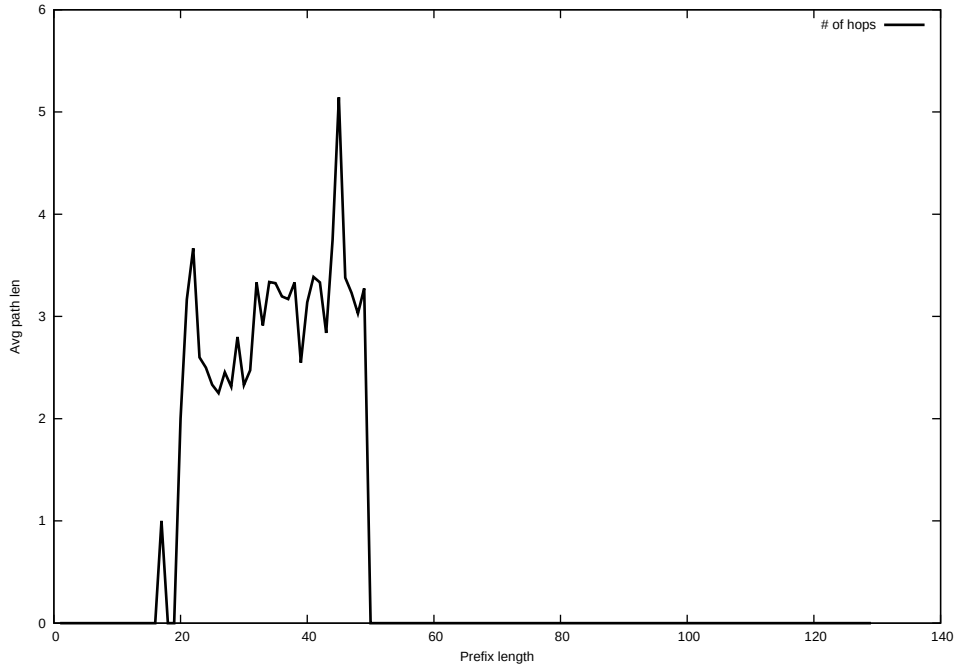
2012-10-29



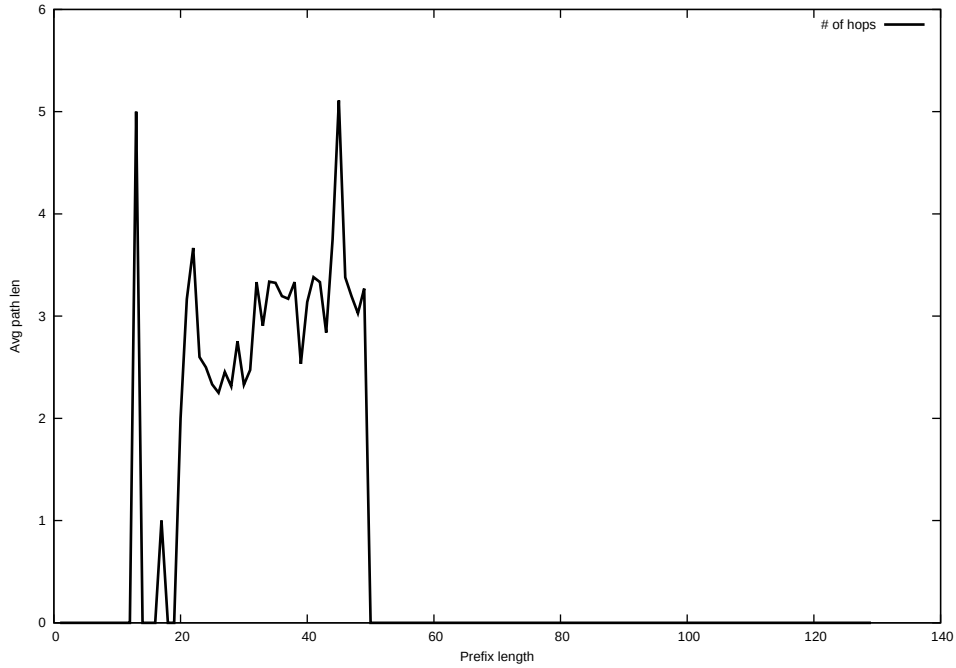
2012-10-30



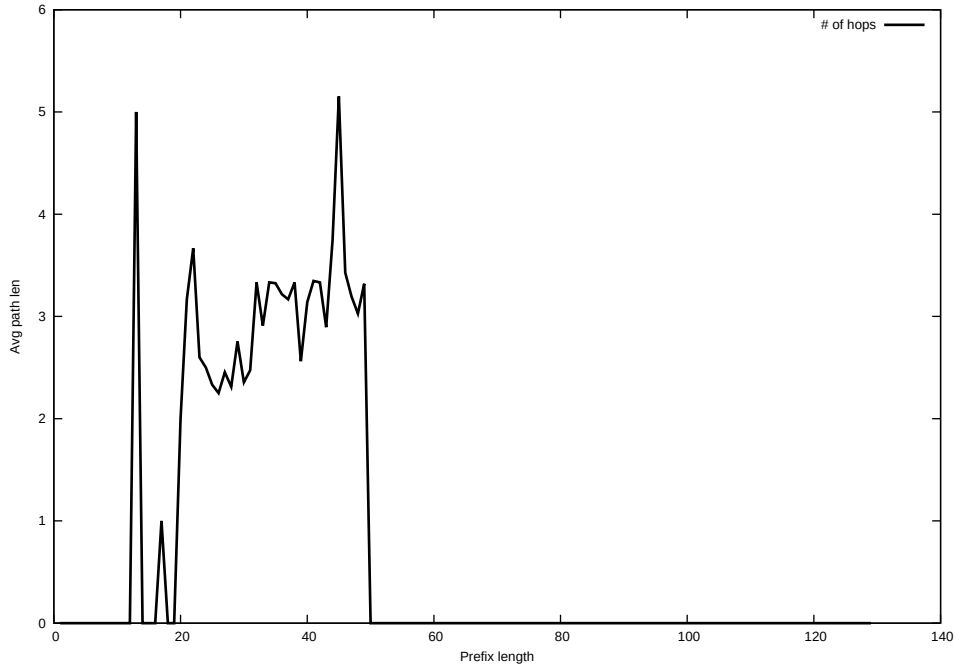
2012-10-31



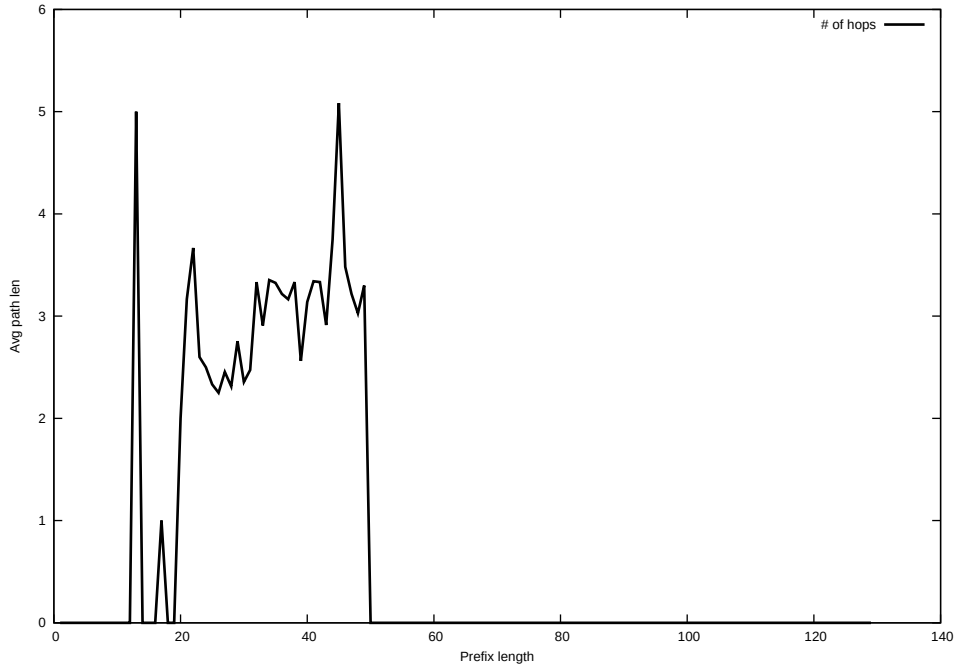
2012-11-01



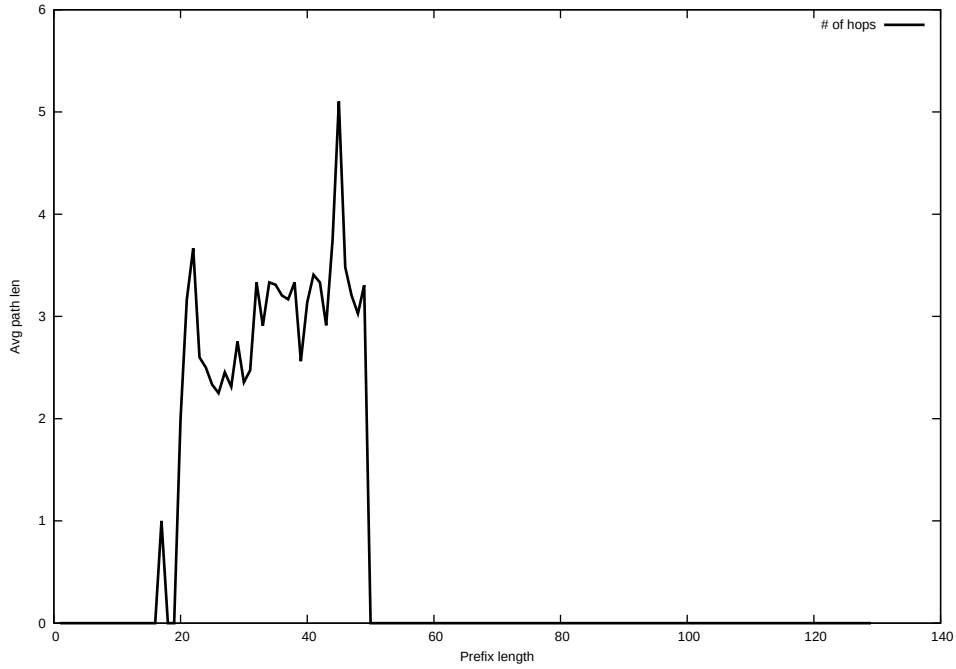
2012-11-02



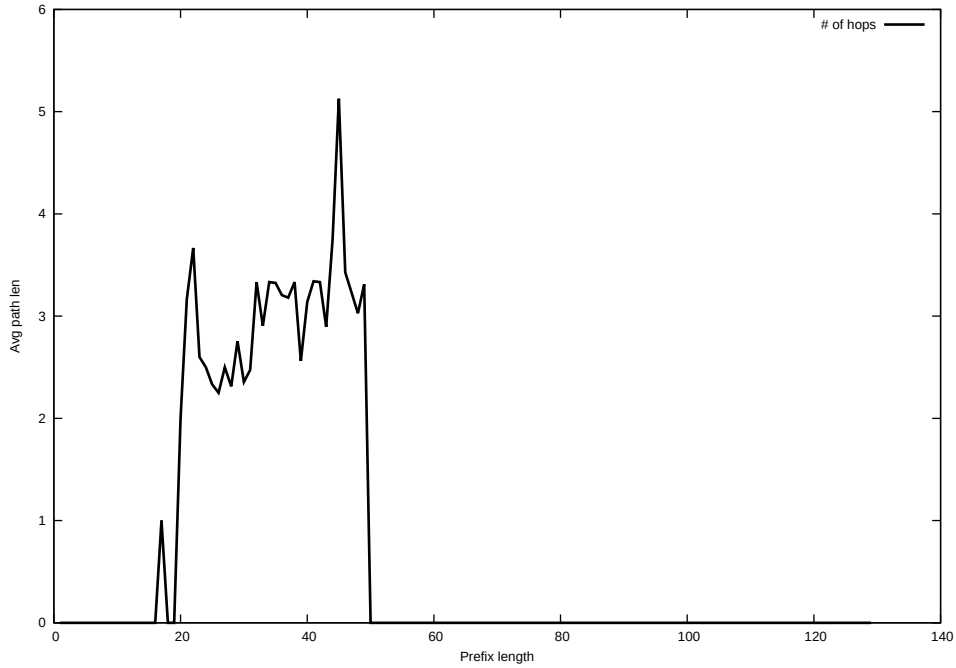
2012-11-03



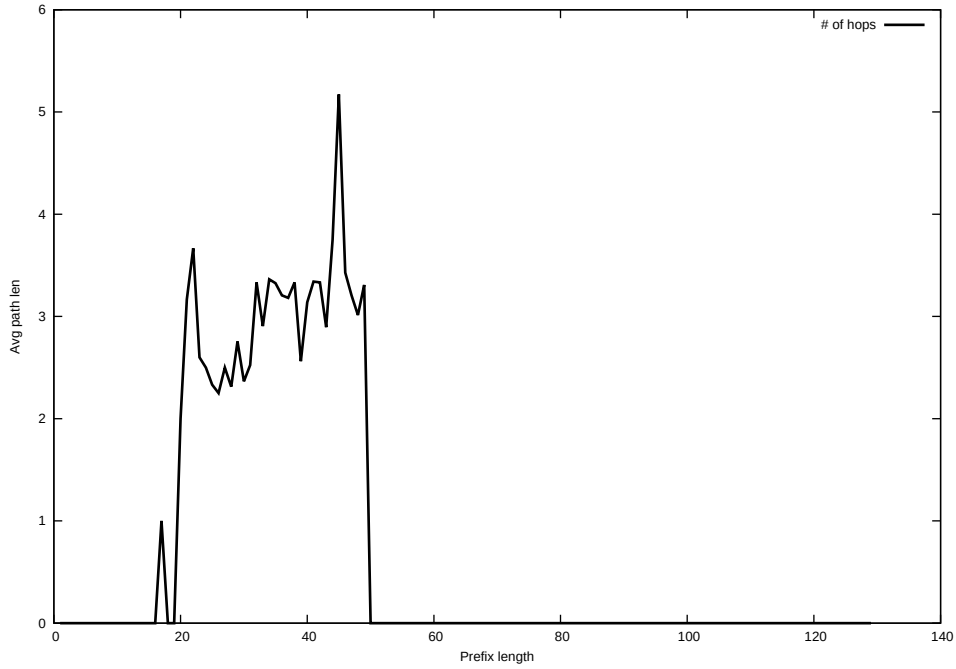
2012-11-04



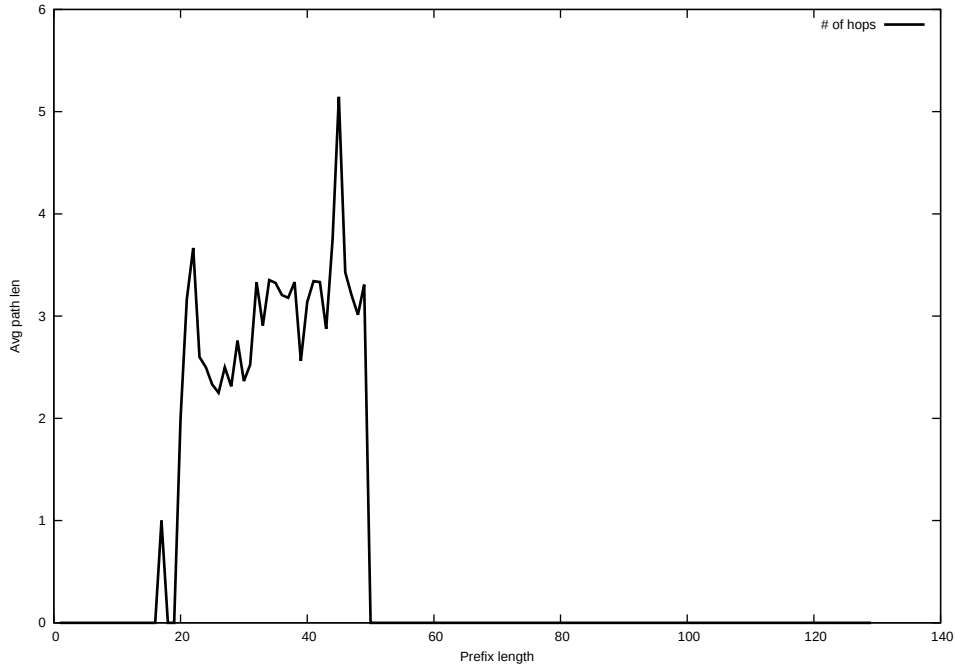
2012-11-05



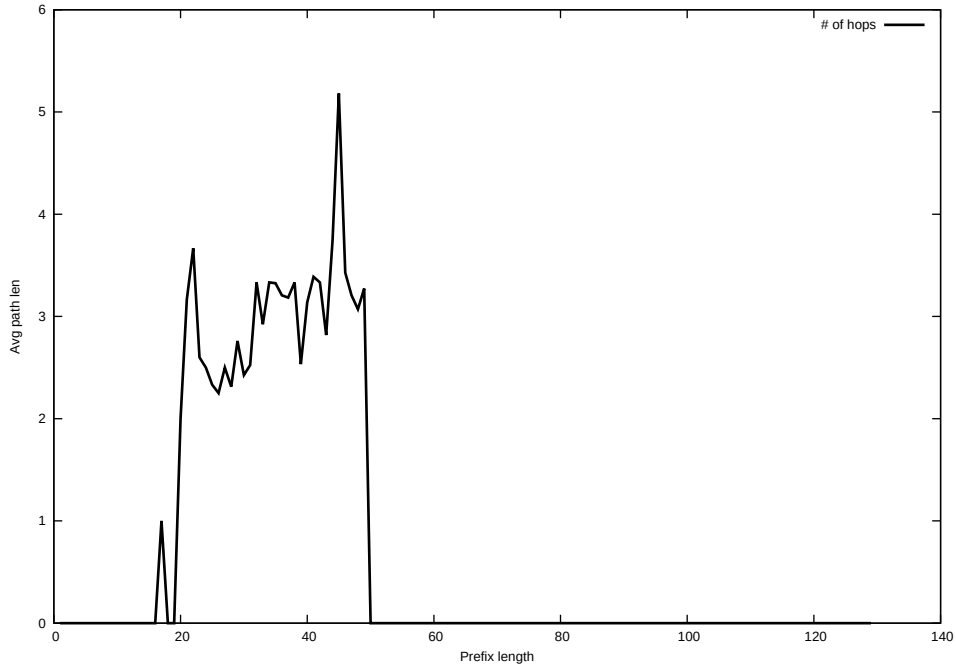
2012-11-06



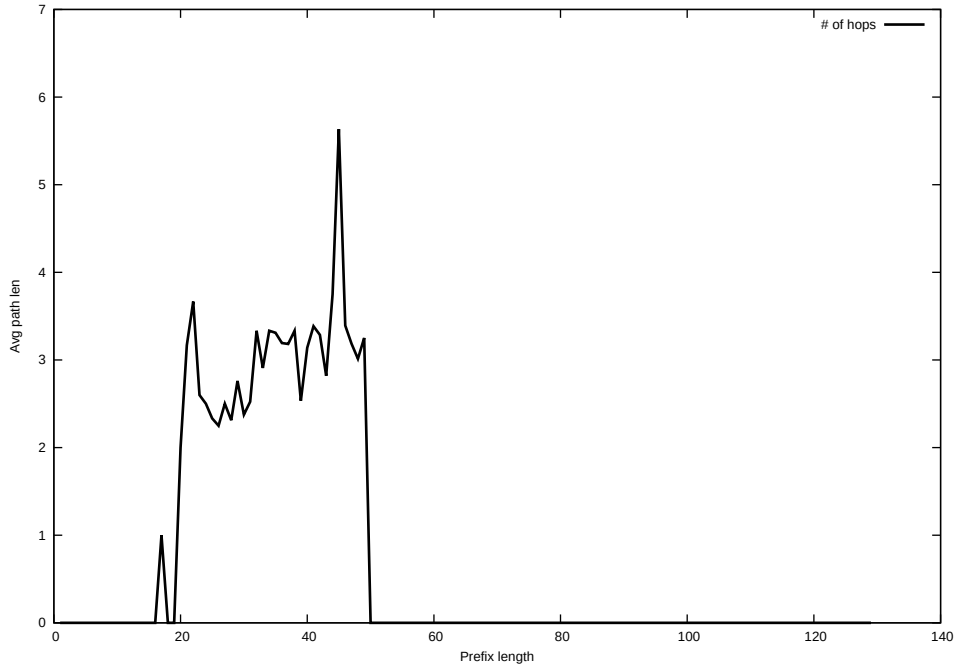
2012-11-07



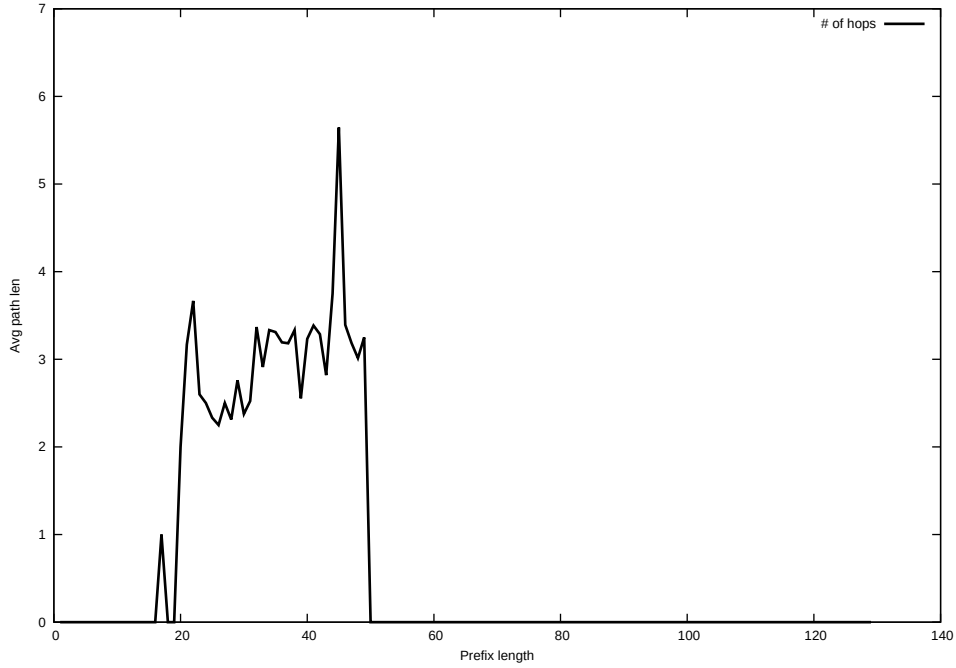
2012-11-08



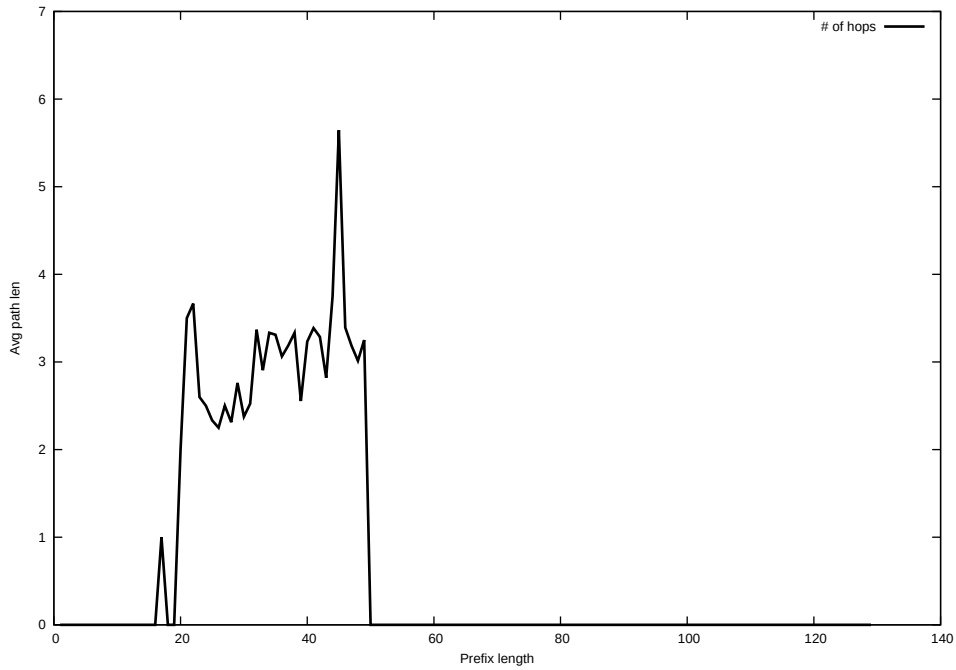
2012-11-09



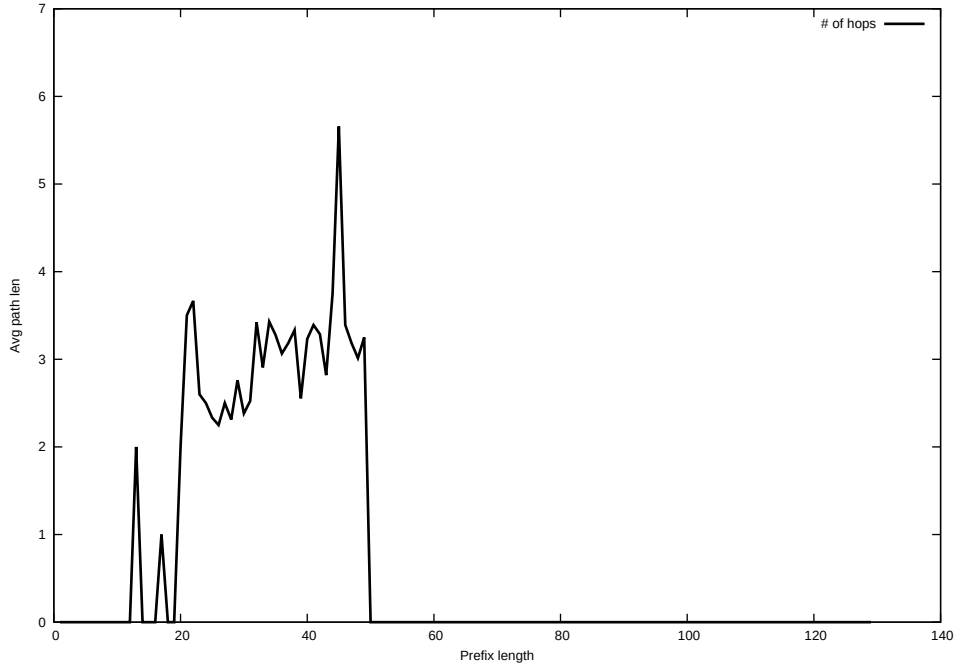
2012-11-10



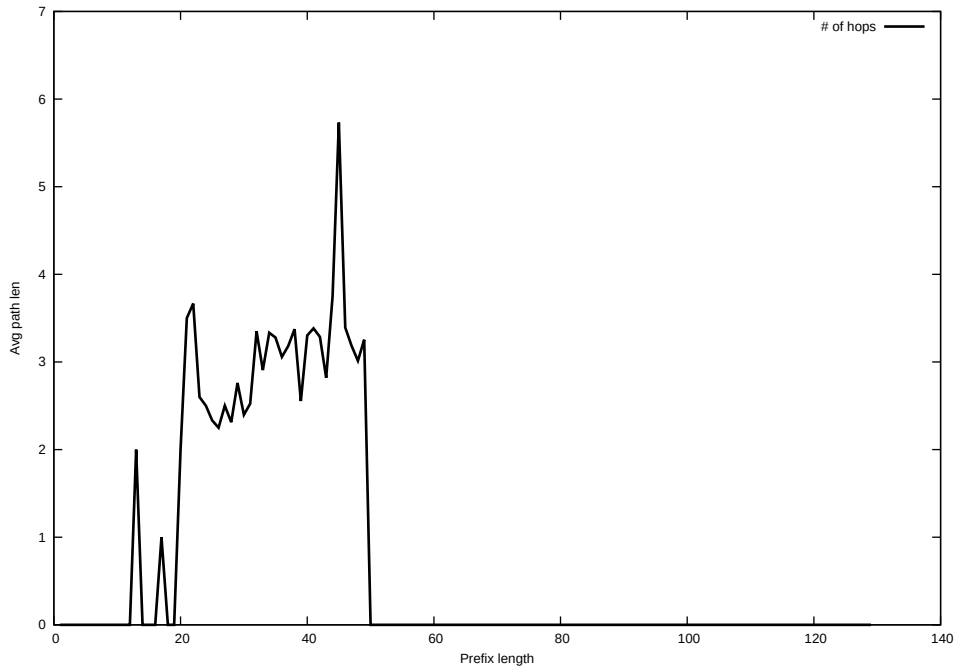
2012-11-11



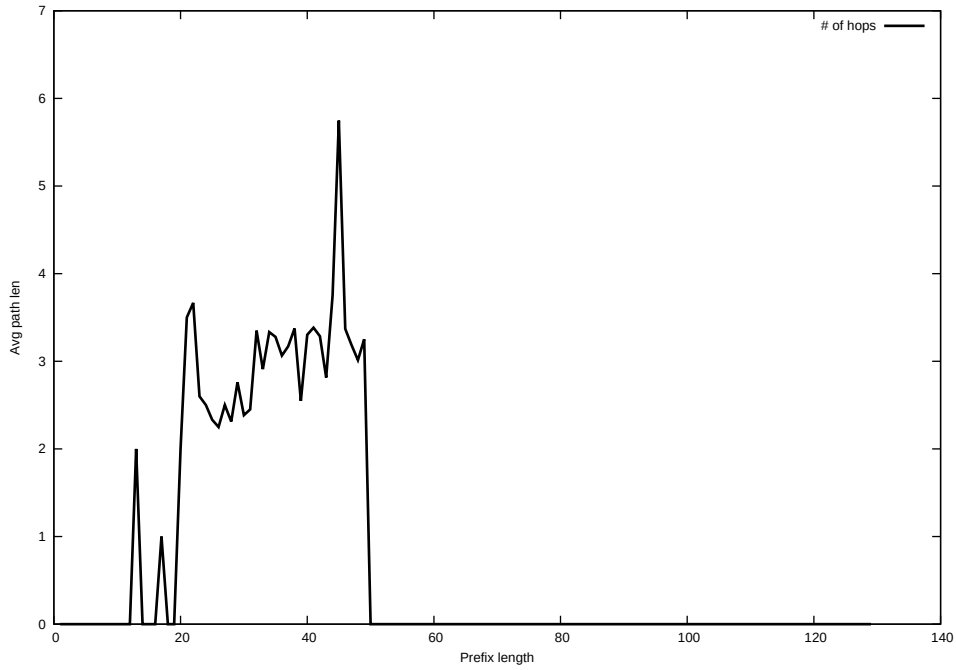
2012-11-12



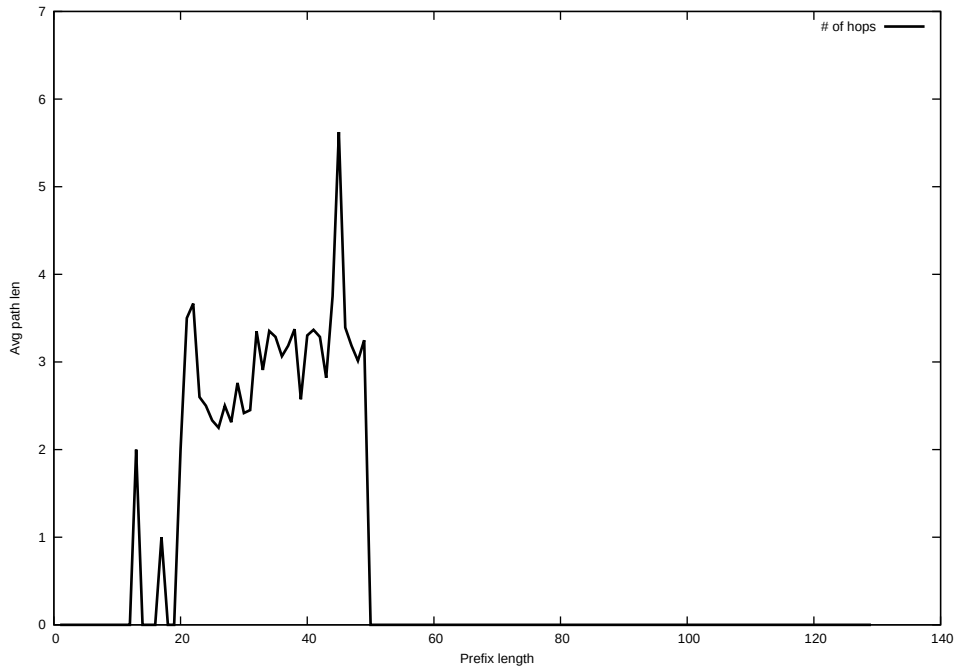
2012-11-13



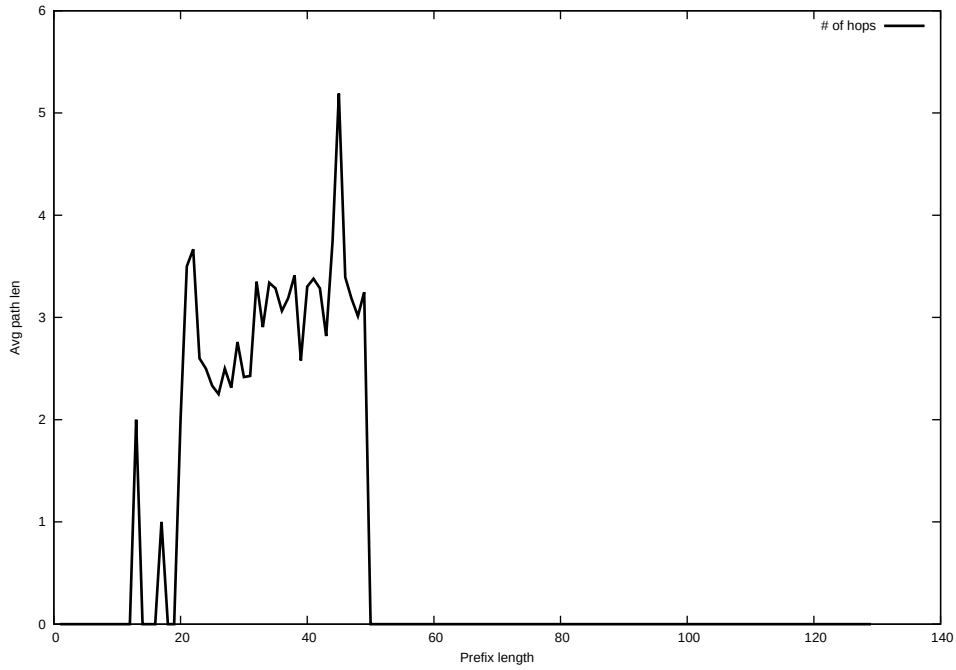
2012-11-14



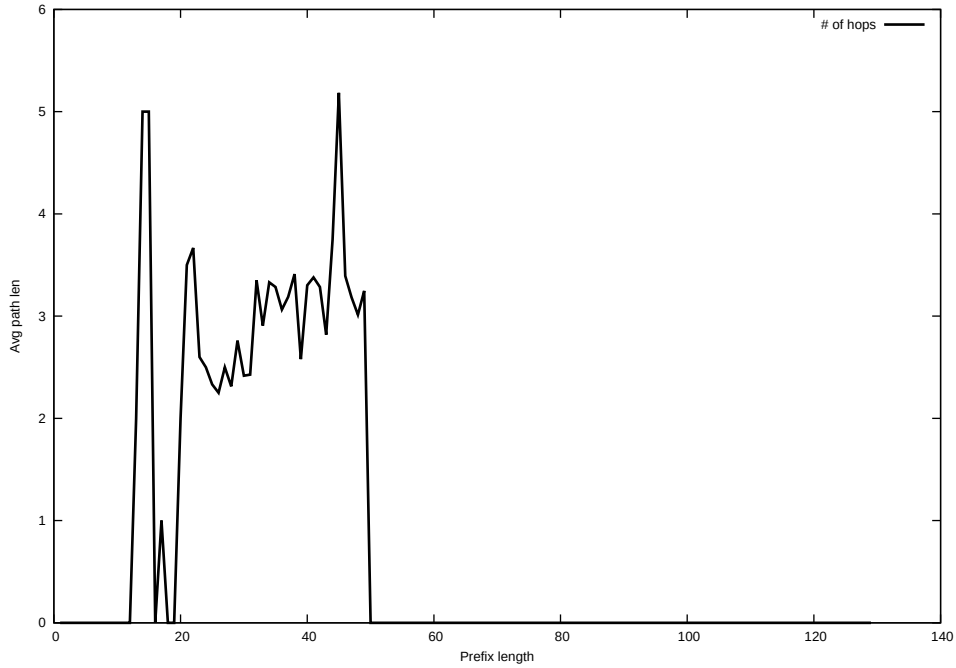
2012-11-15



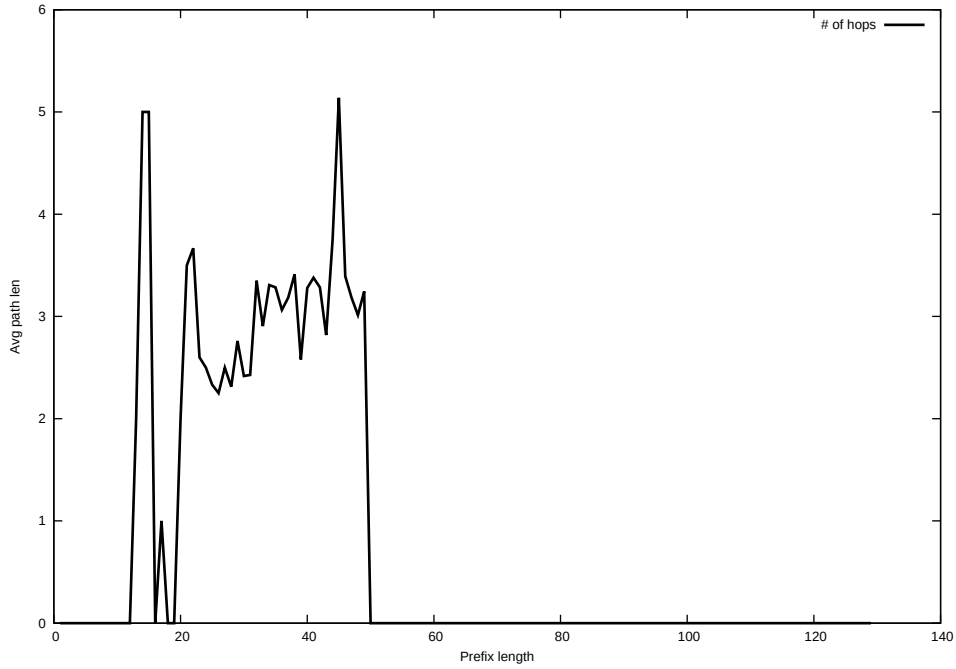
2012-11-16



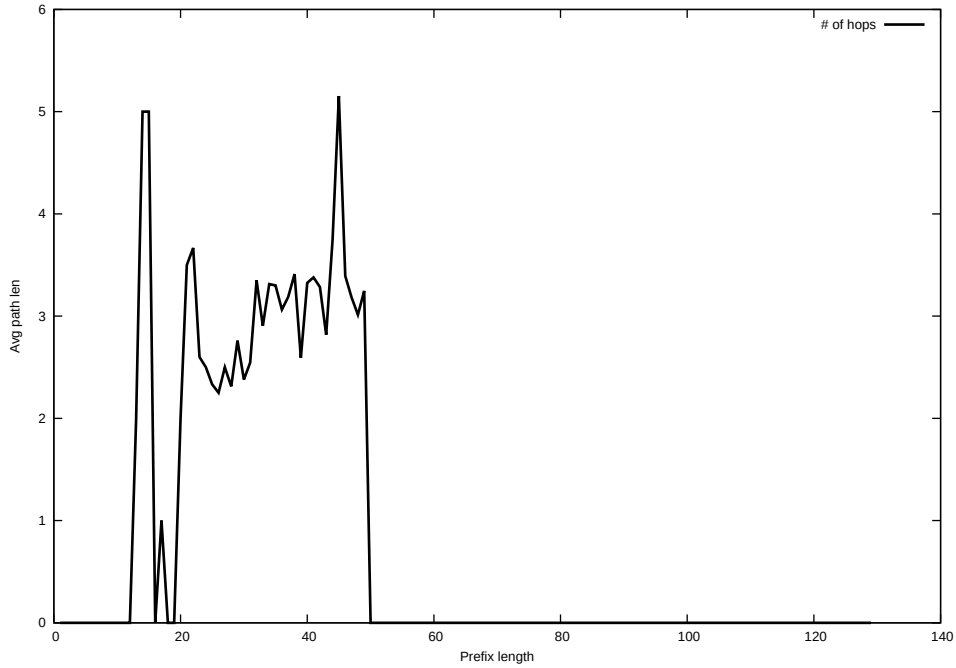
2012-11-17



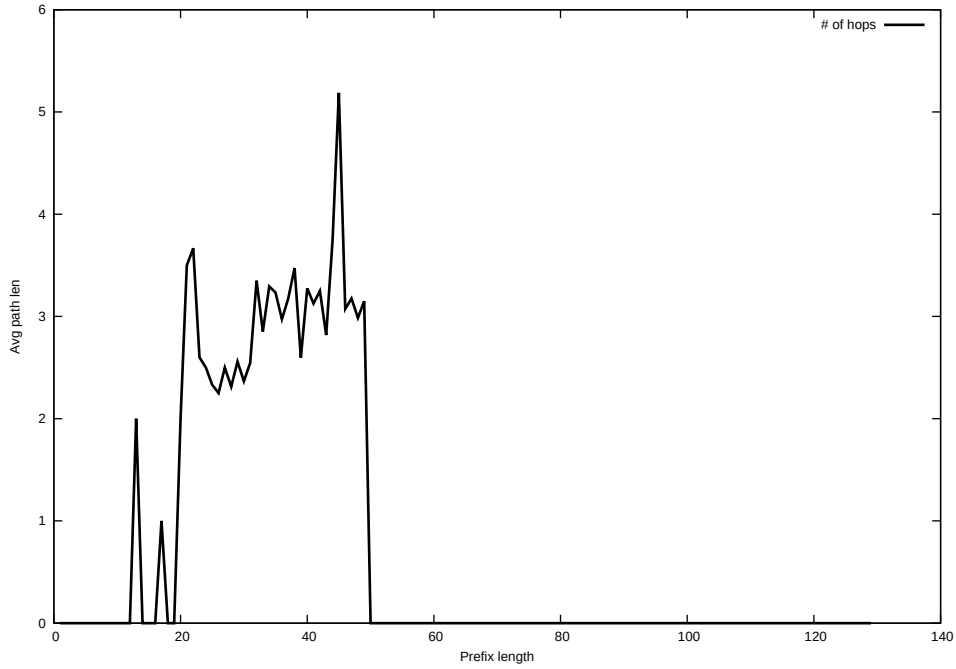
2012-11-18



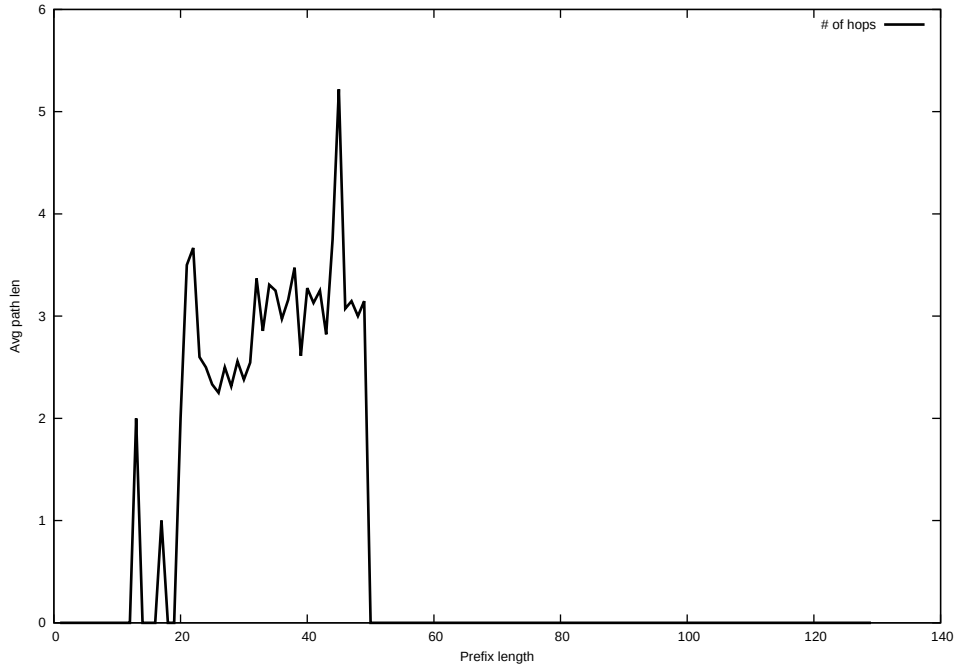
2012-11-19



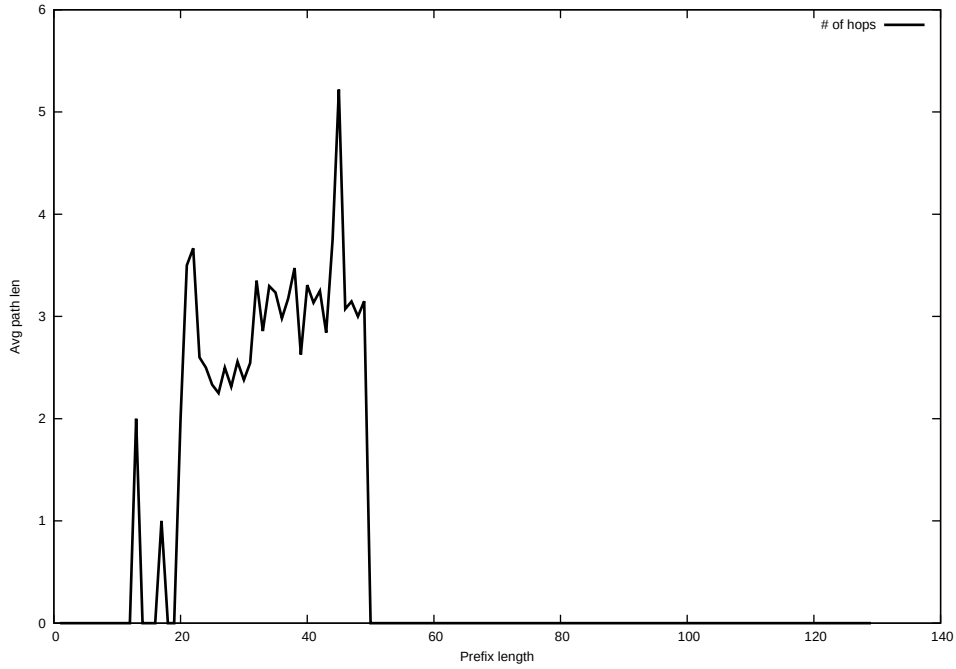
2012-11-20



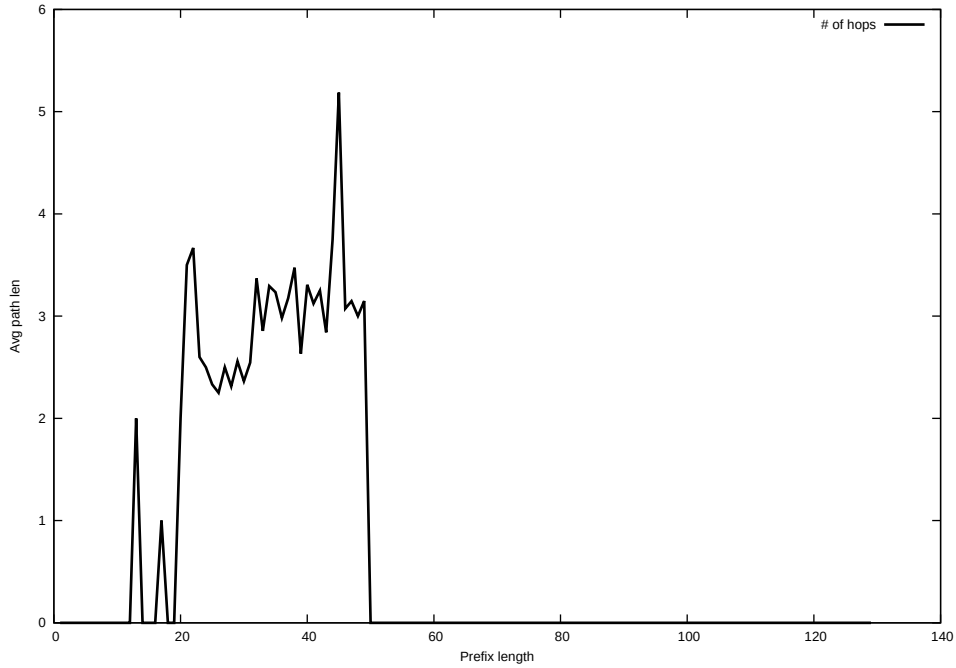
2012-11-21



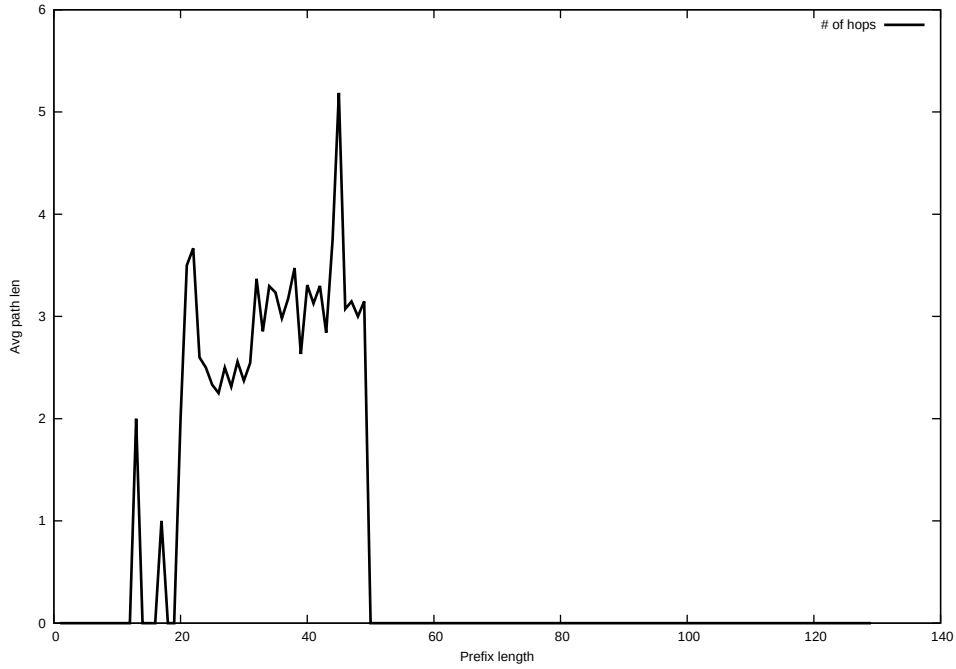
2012-11-22



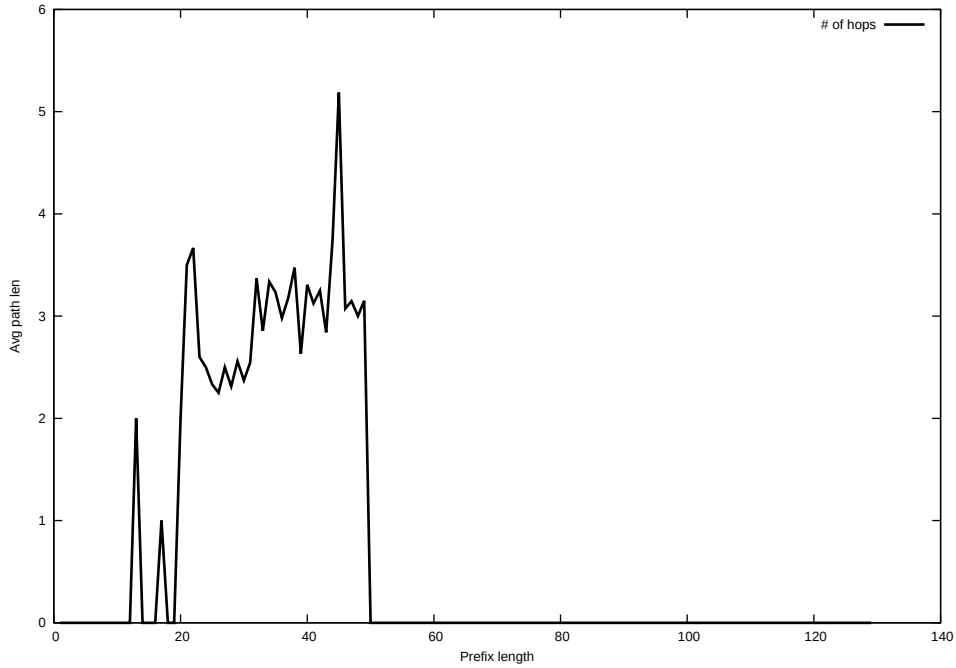
2012-11-23



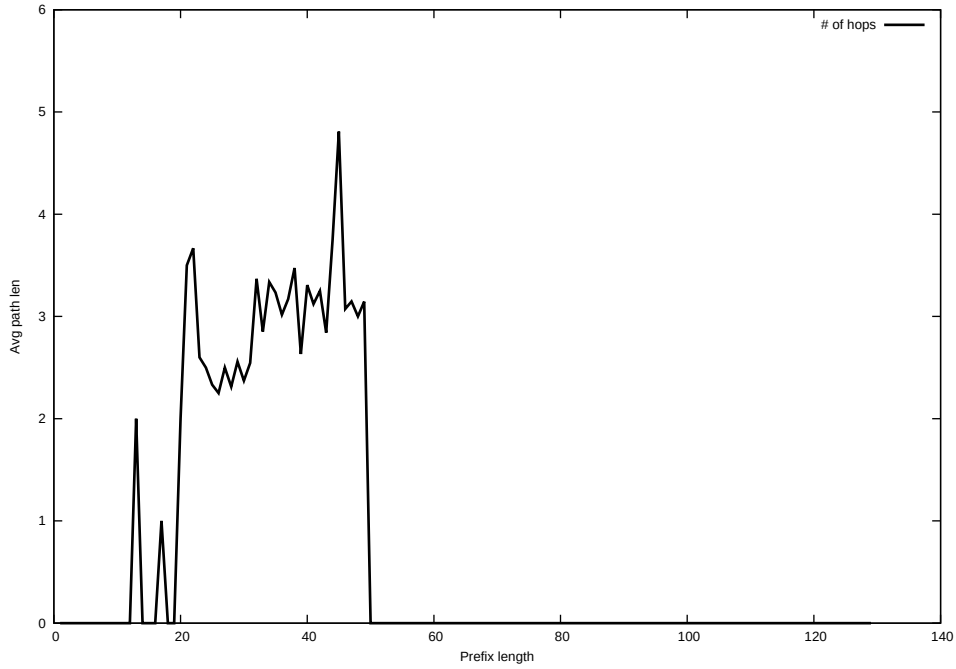
2012-11-24



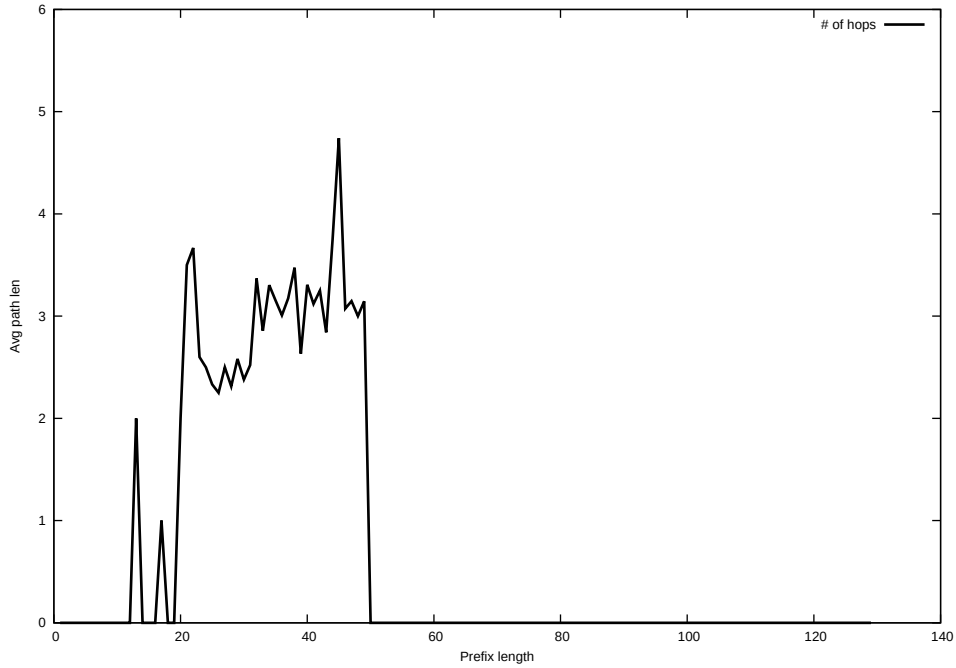
2012-11-25



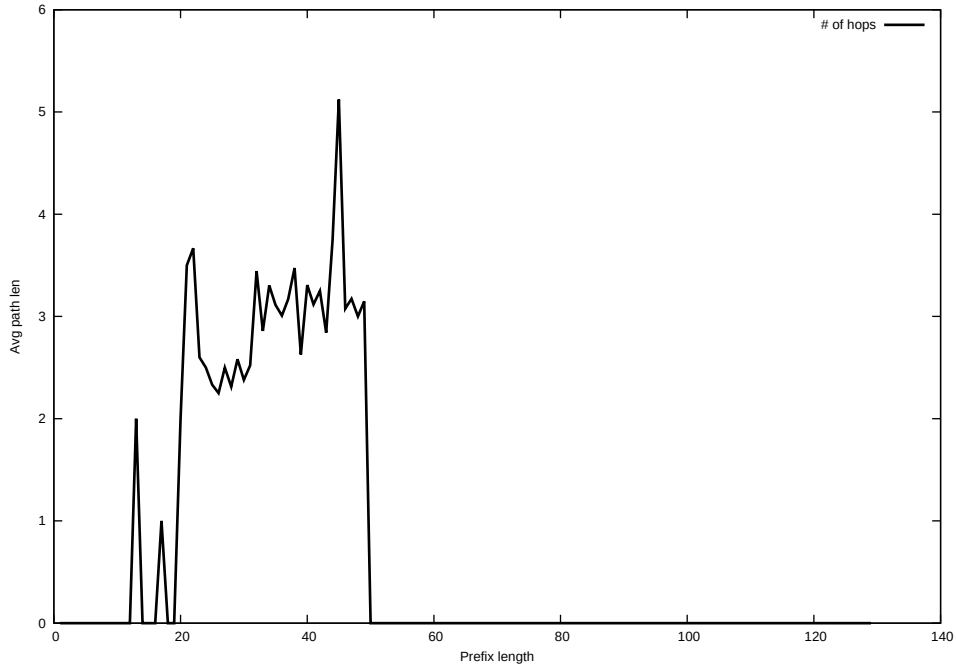
2012-11-26



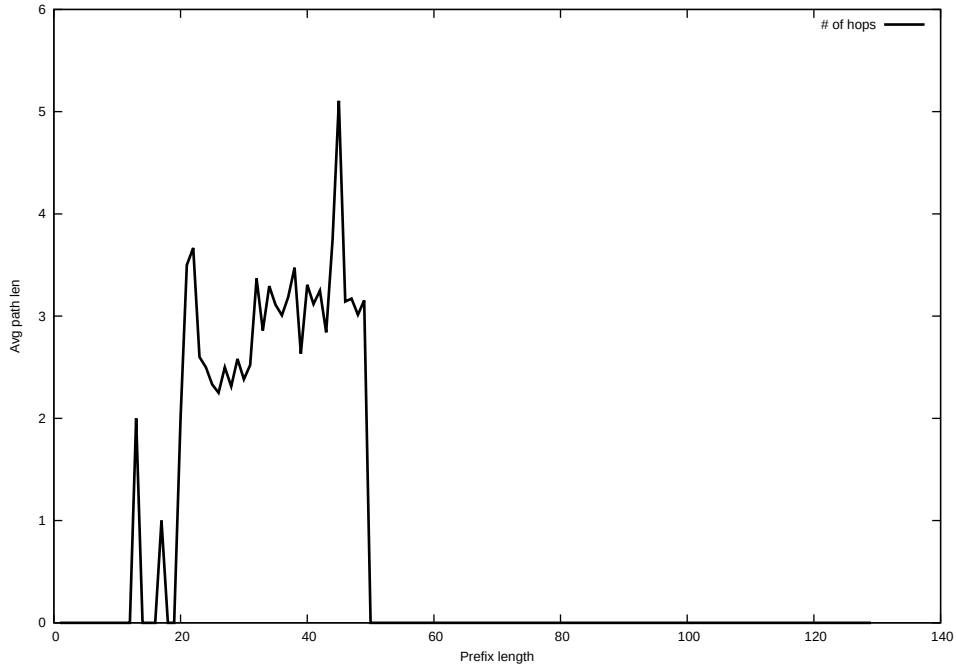
2012-11-27



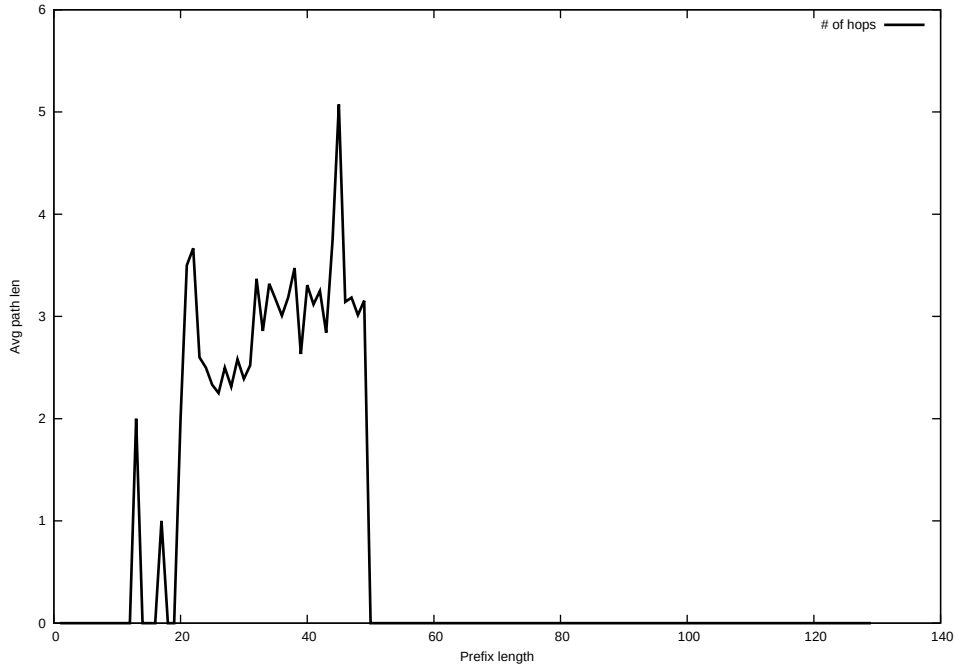
2012-11-28



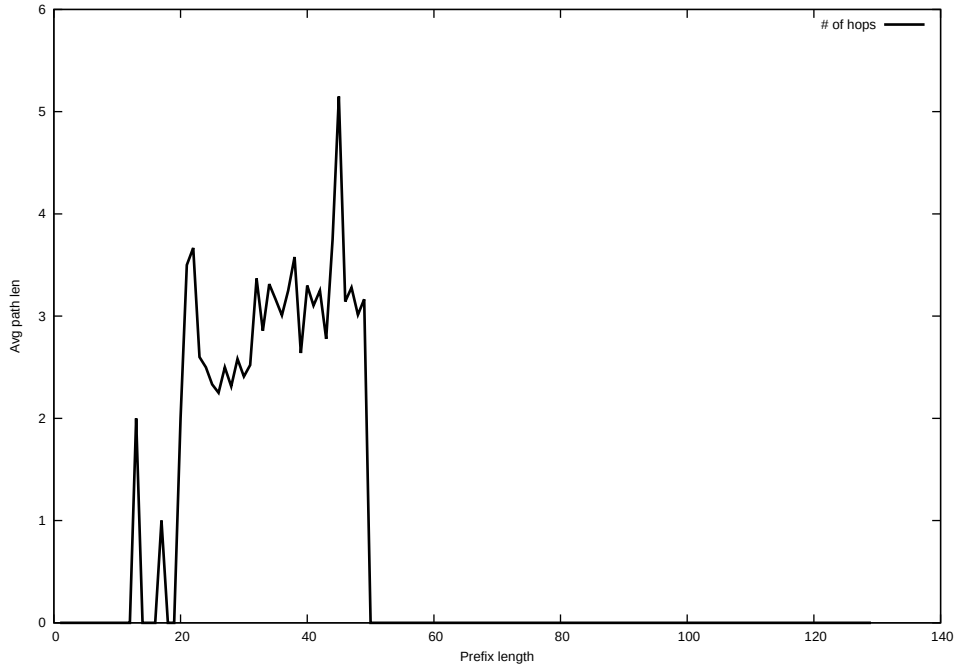
2012-11-29



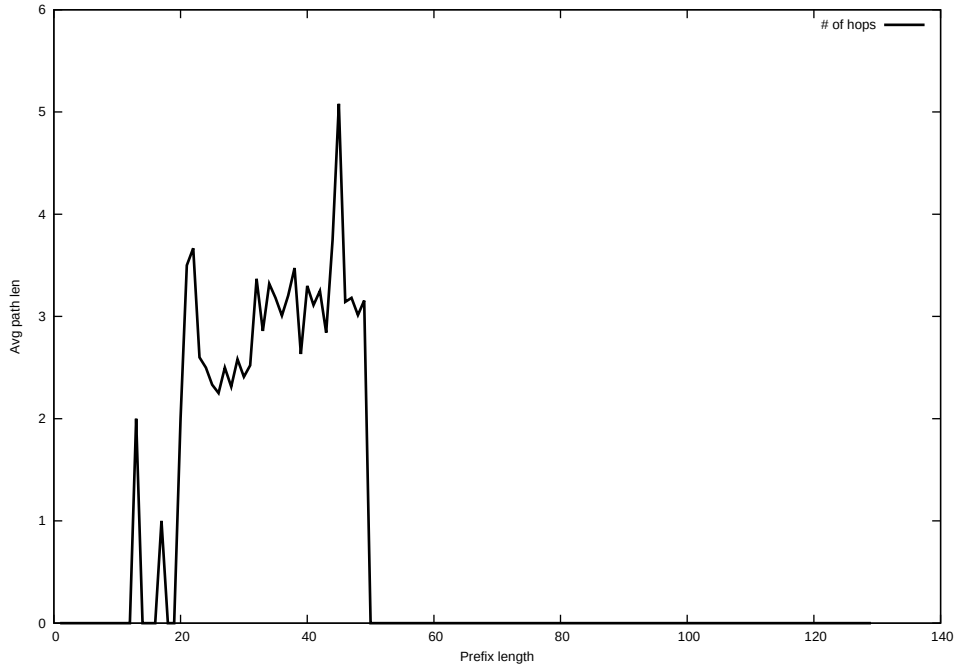
2012-11-30



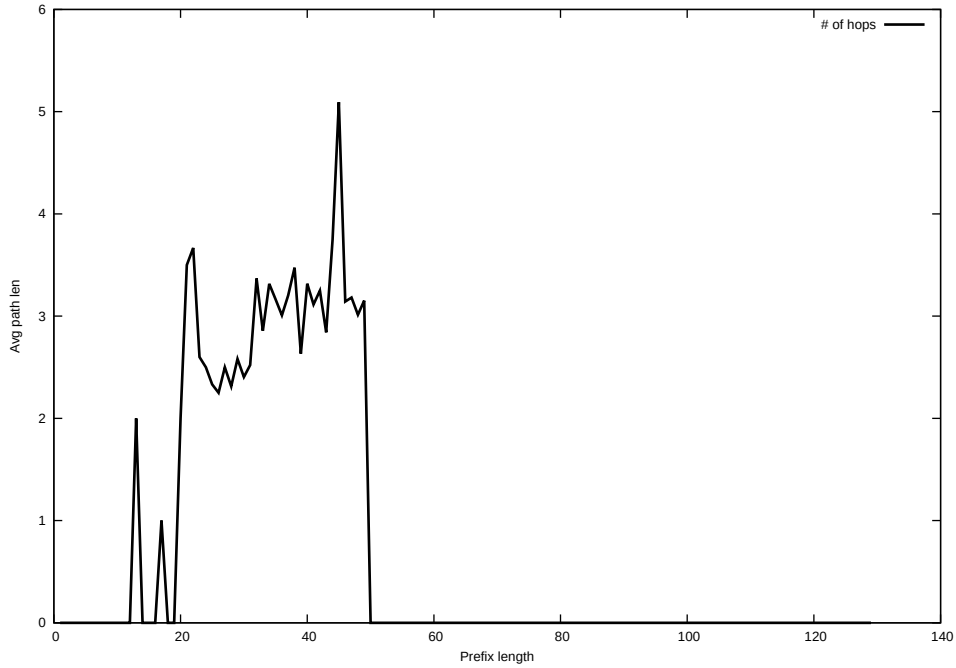
2012-12-01



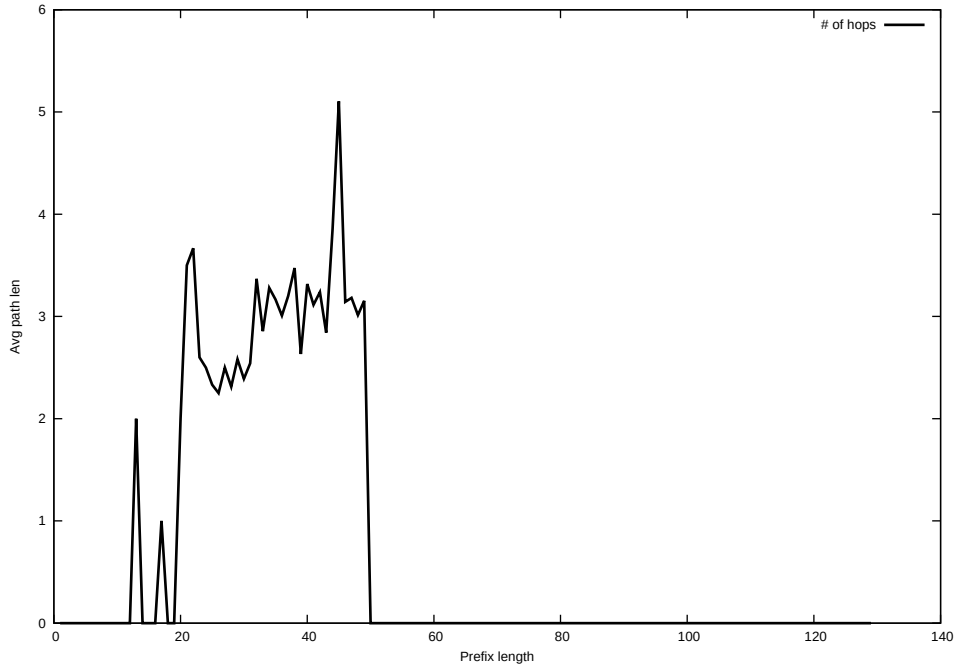
2012-12-02



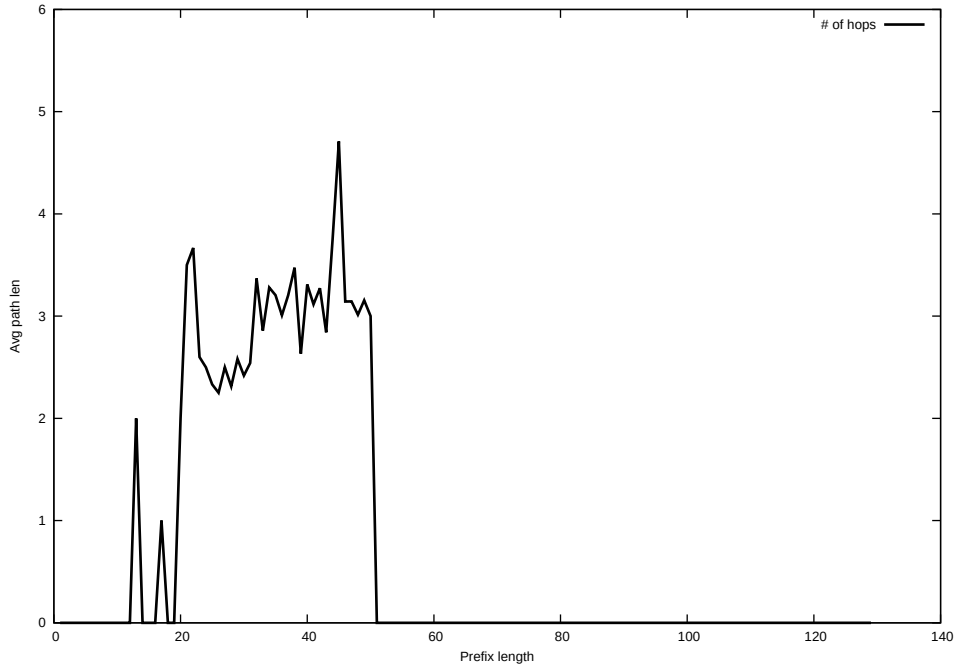
2012-12-03



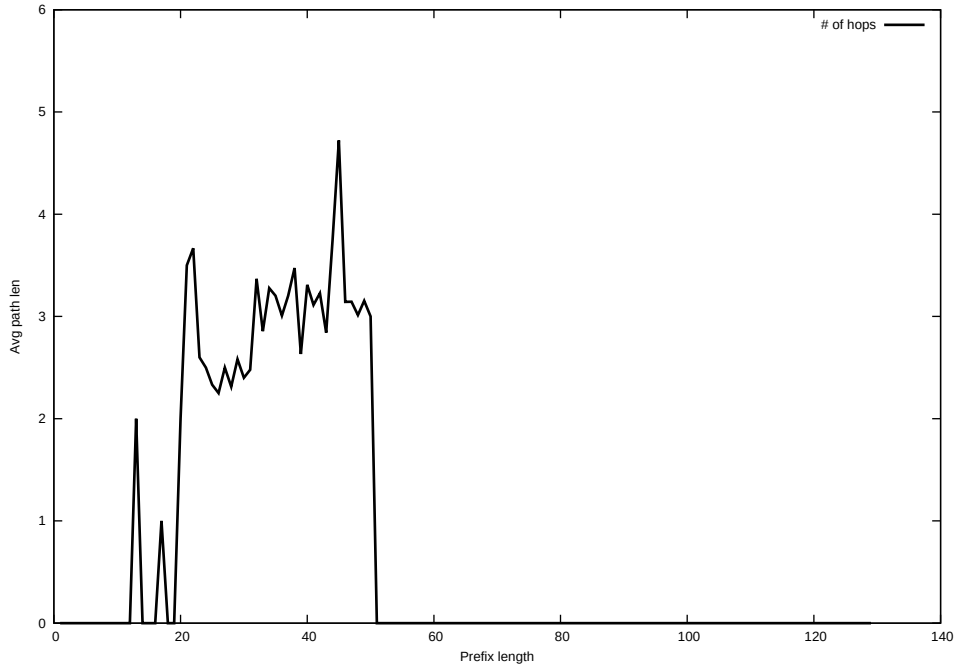
2012-12-04



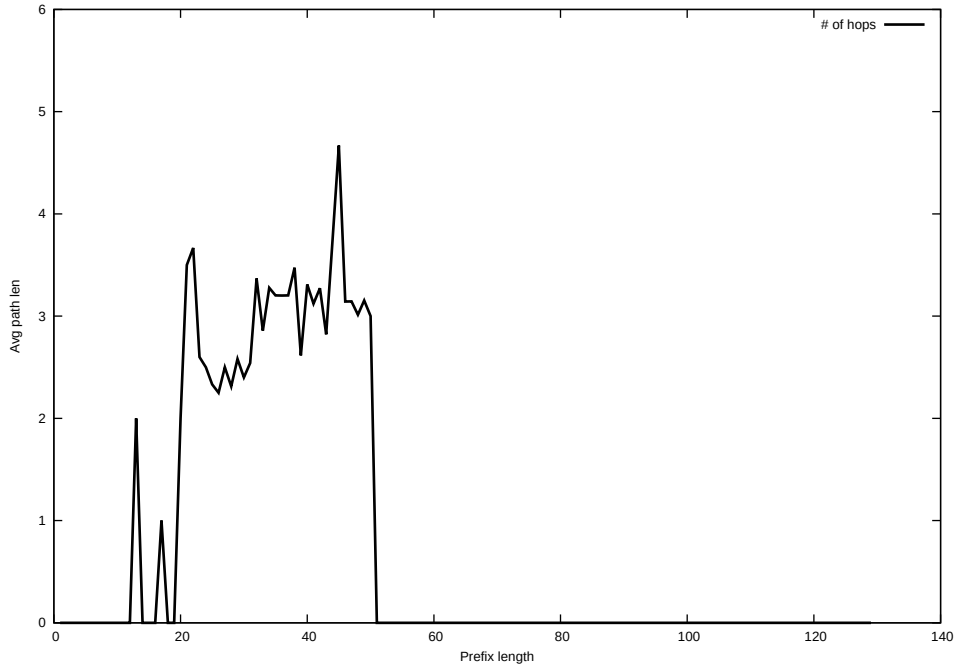
2012-12-05



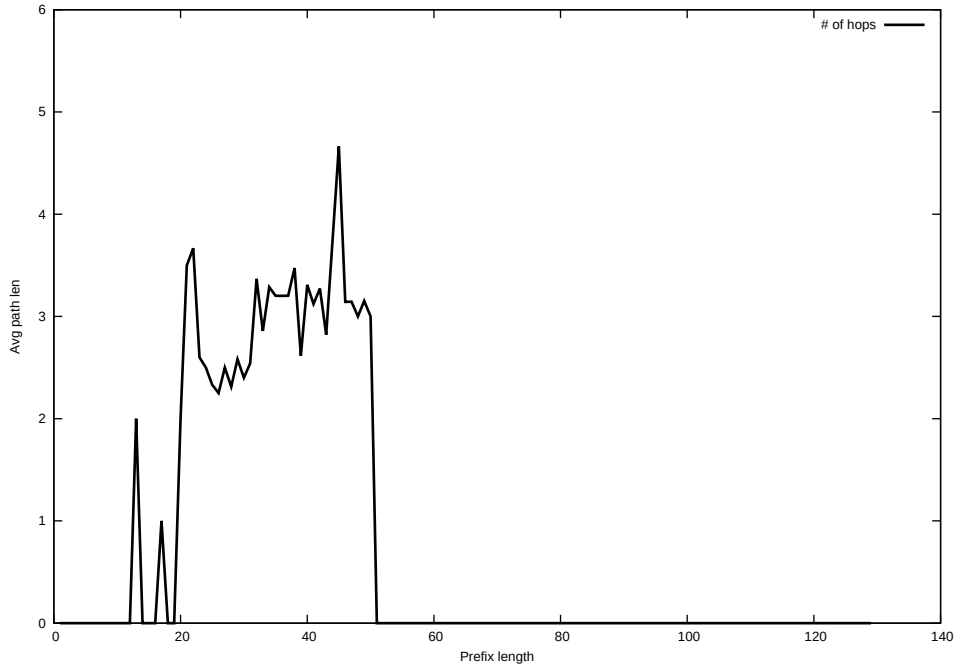
2012-12-06



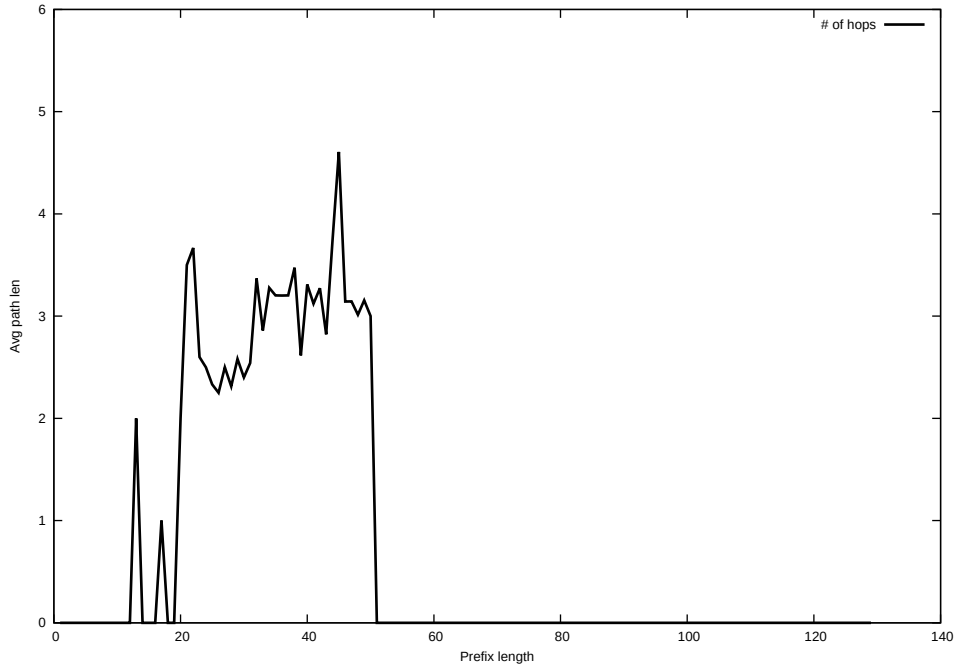
2012-12-07



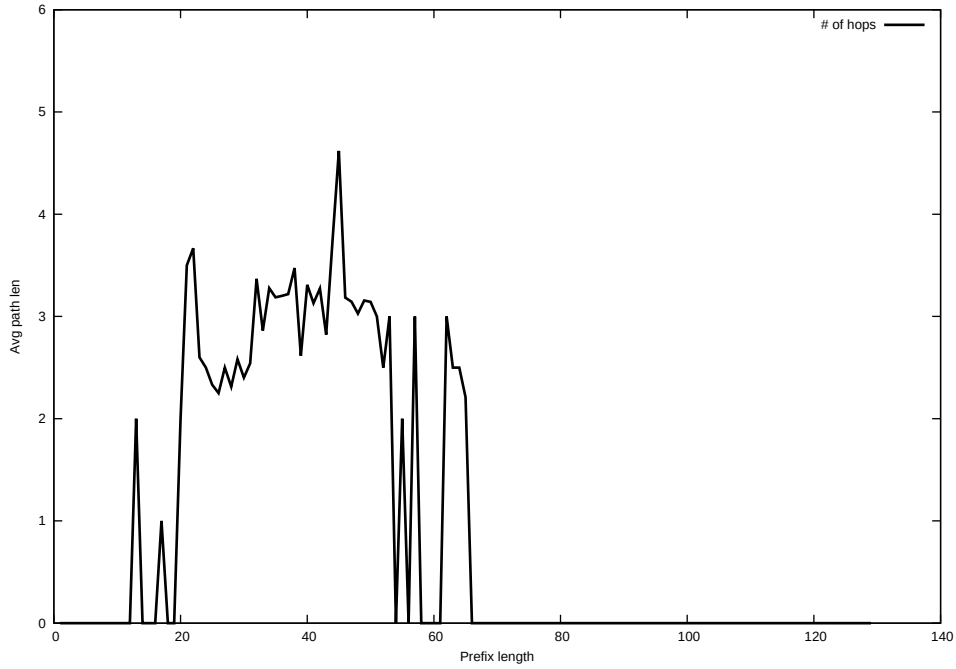
2012-12-08



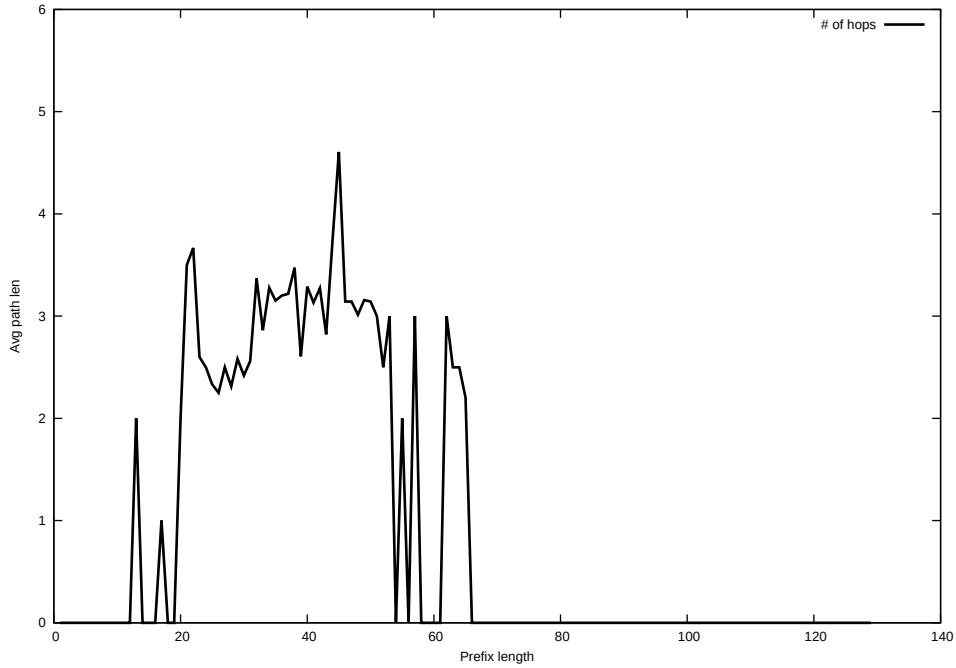
2012-12-09



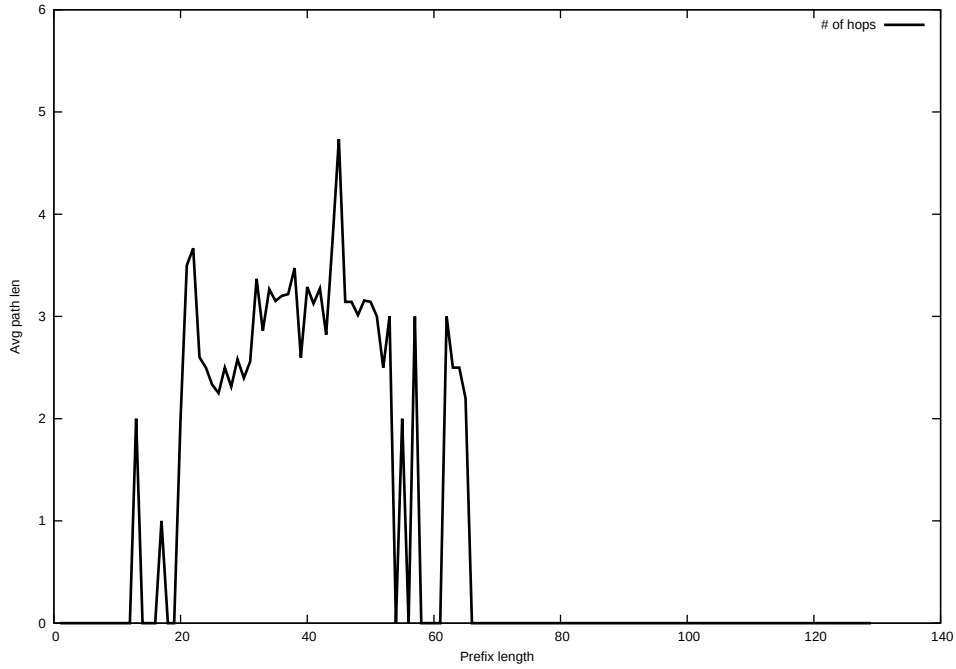
2012-12-10



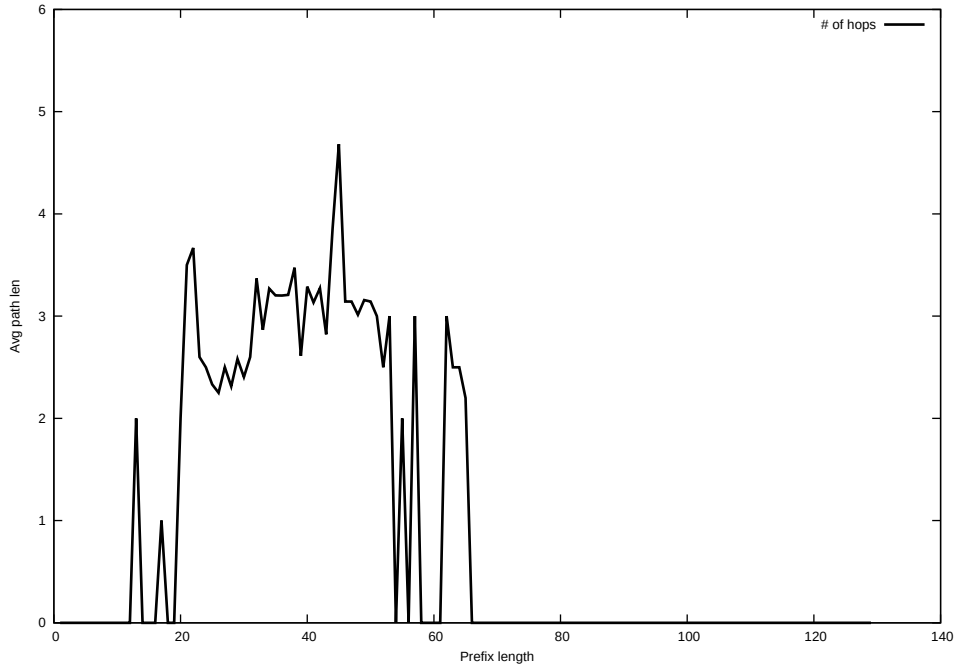
2012-12-11



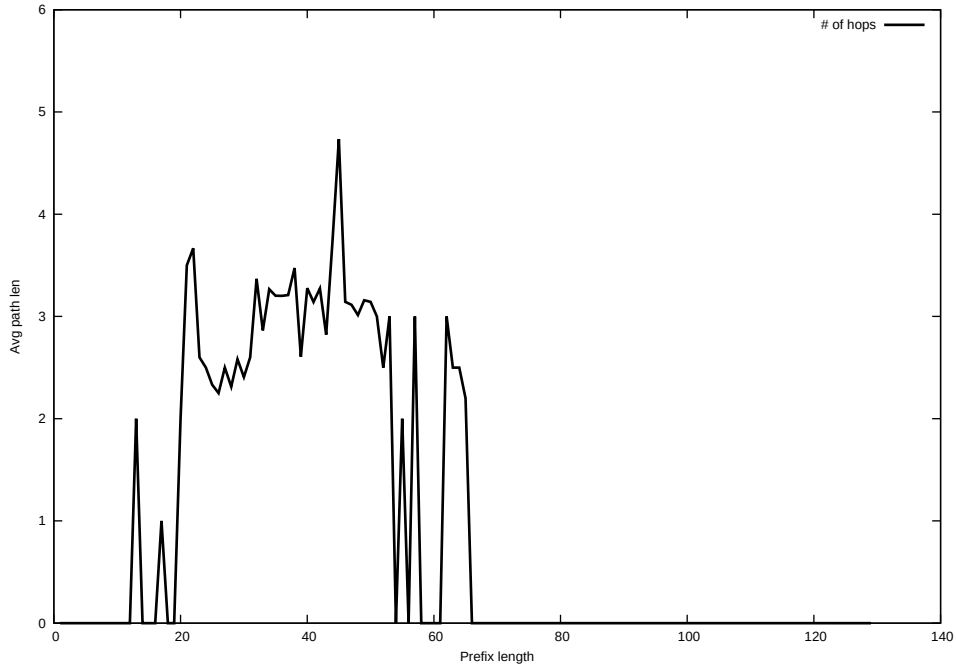
2012-12-12



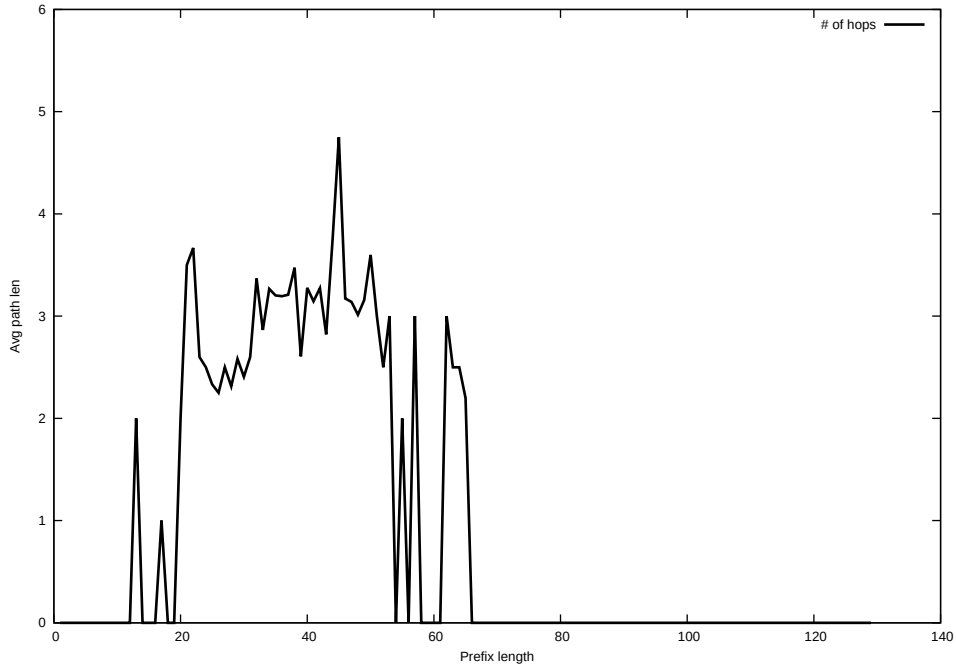
2012-12-13



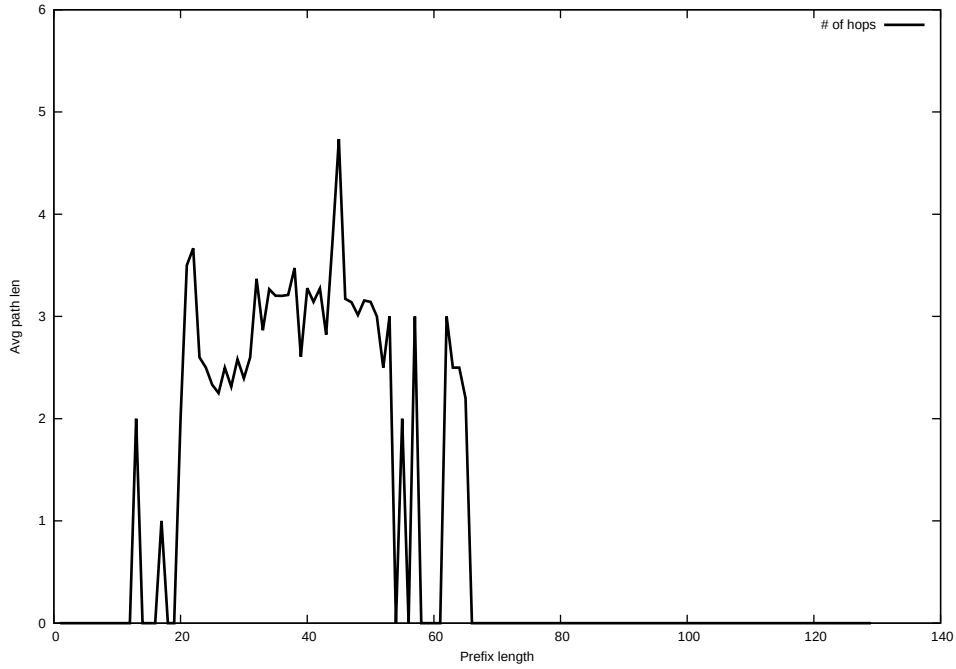
2012-12-14



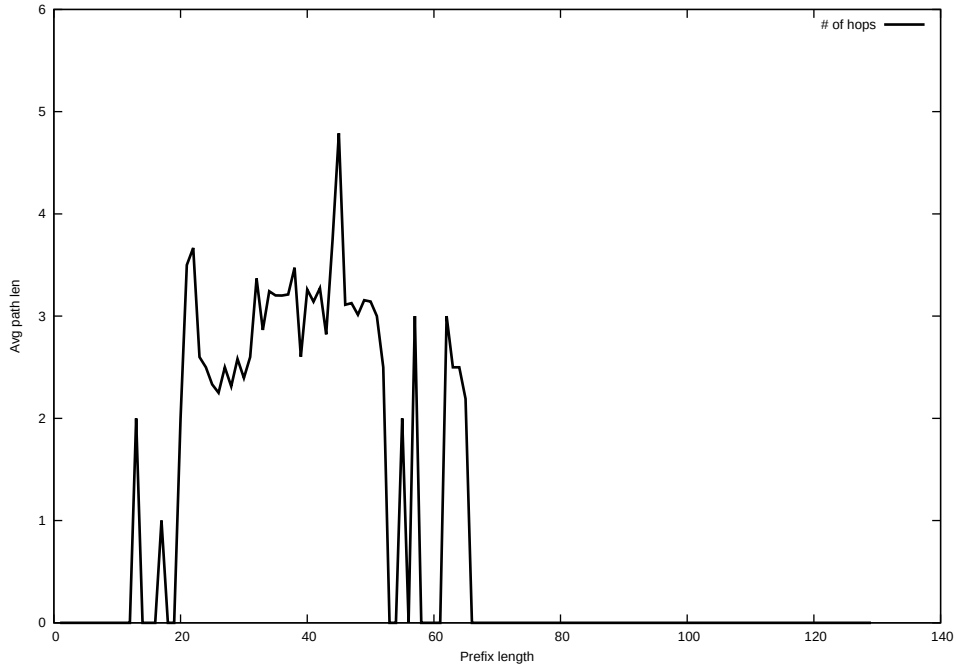
2012-12-15



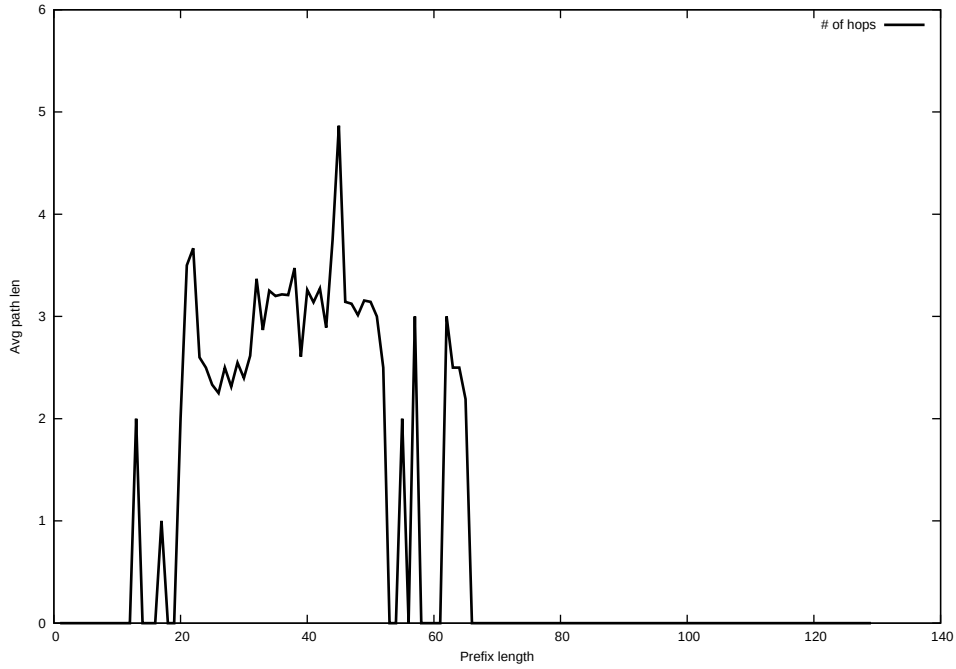
2012-12-16



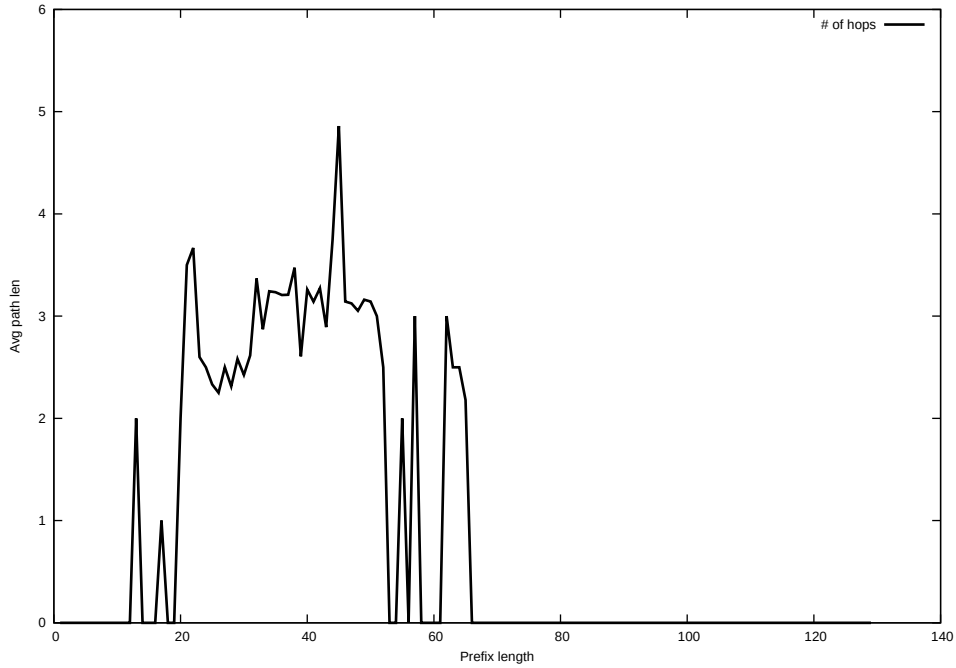
2012-12-17



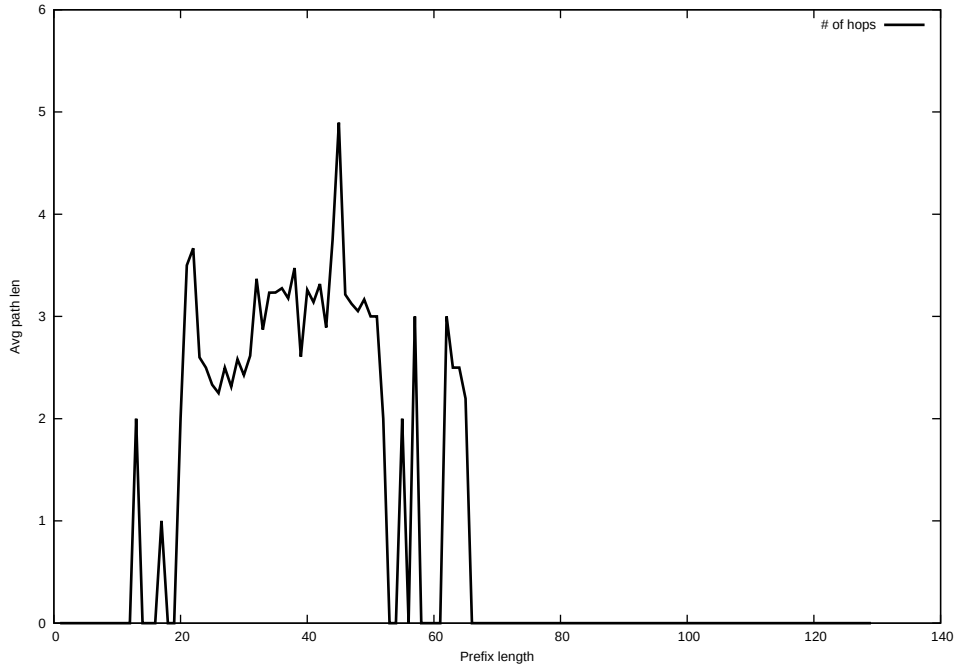
2012-12-18



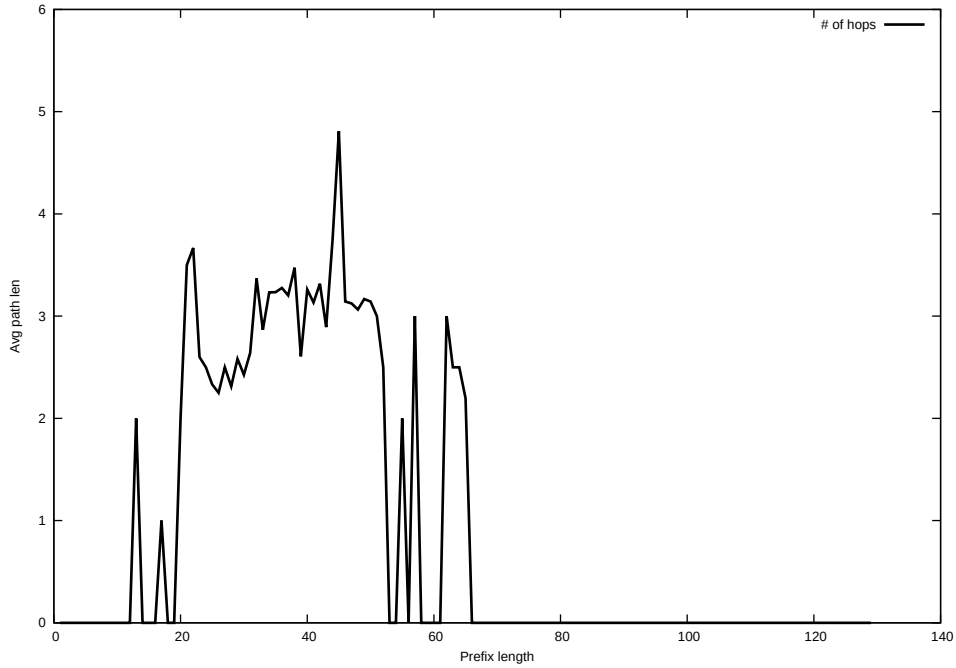
2012-12-19



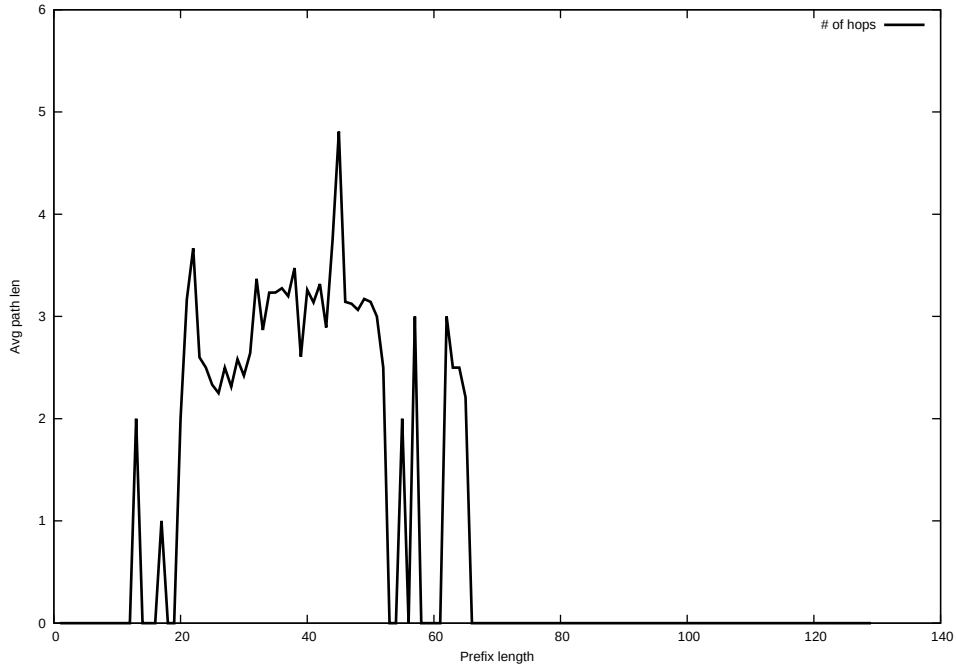
2012-12-20



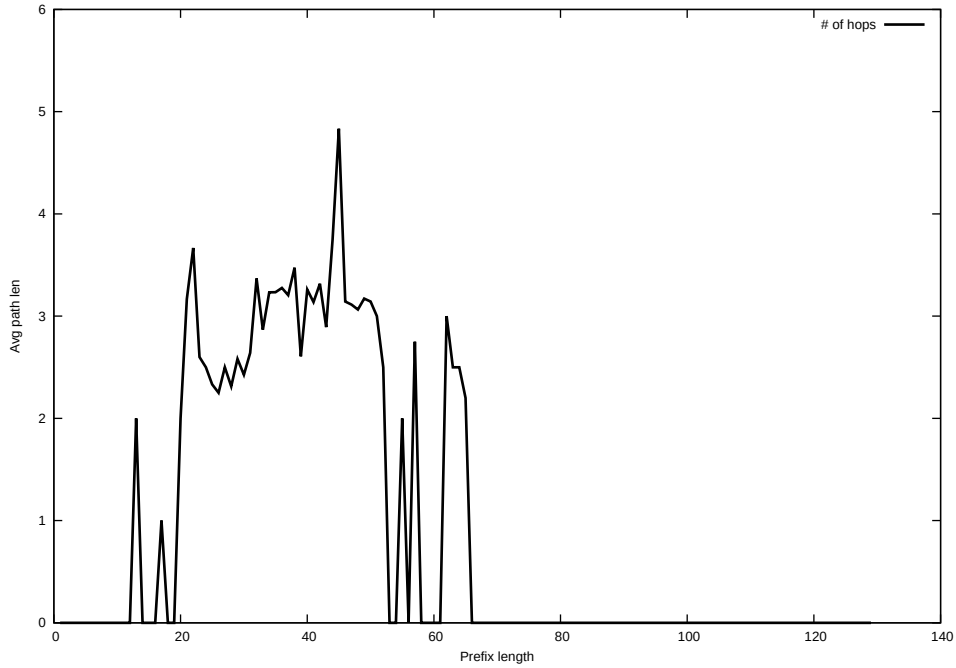
2012-12-21



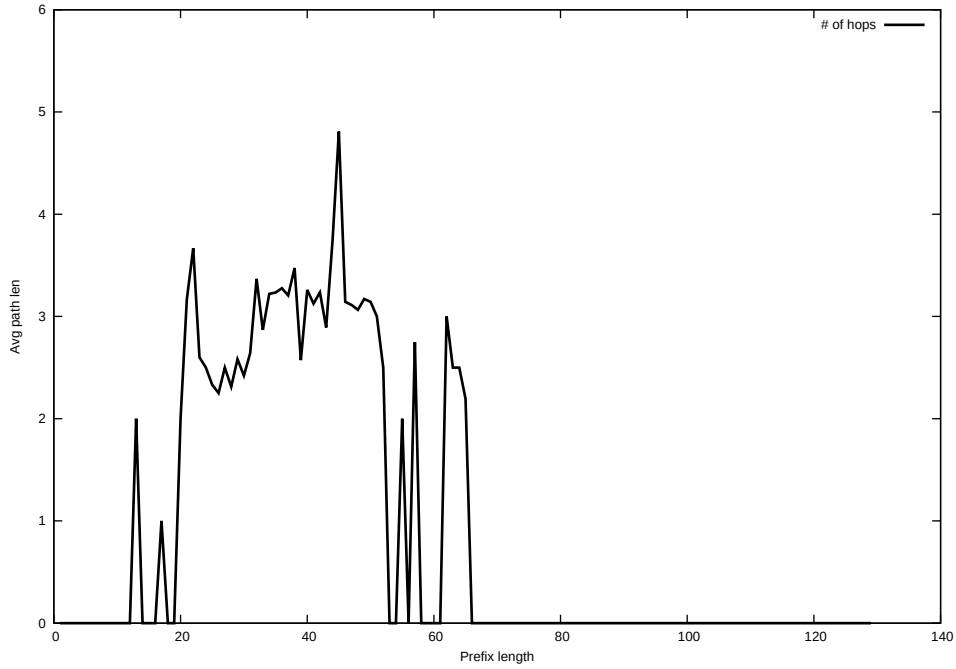
2012-12-22



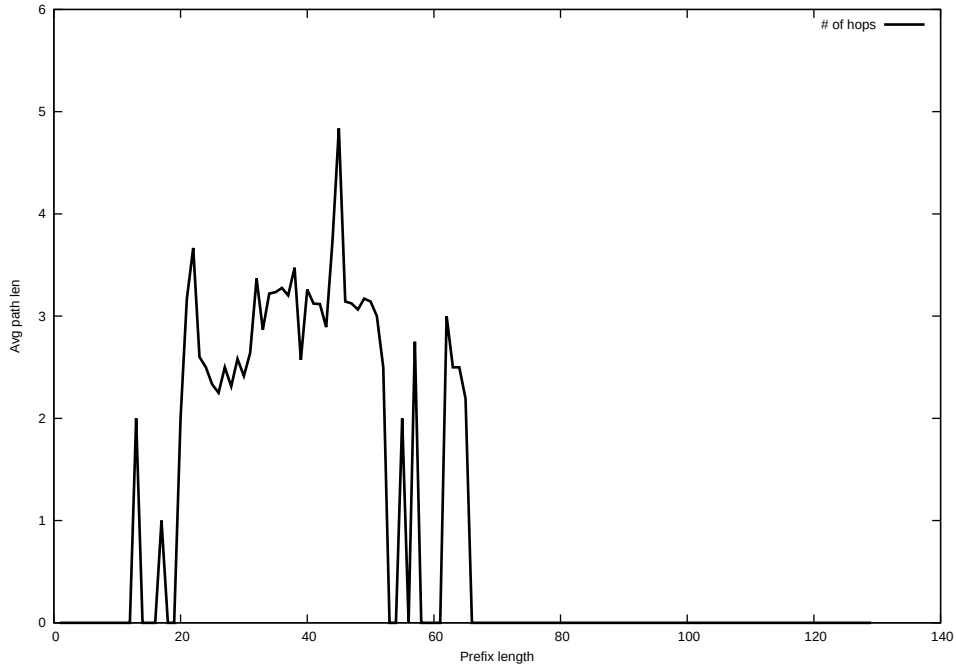
2012-12-23



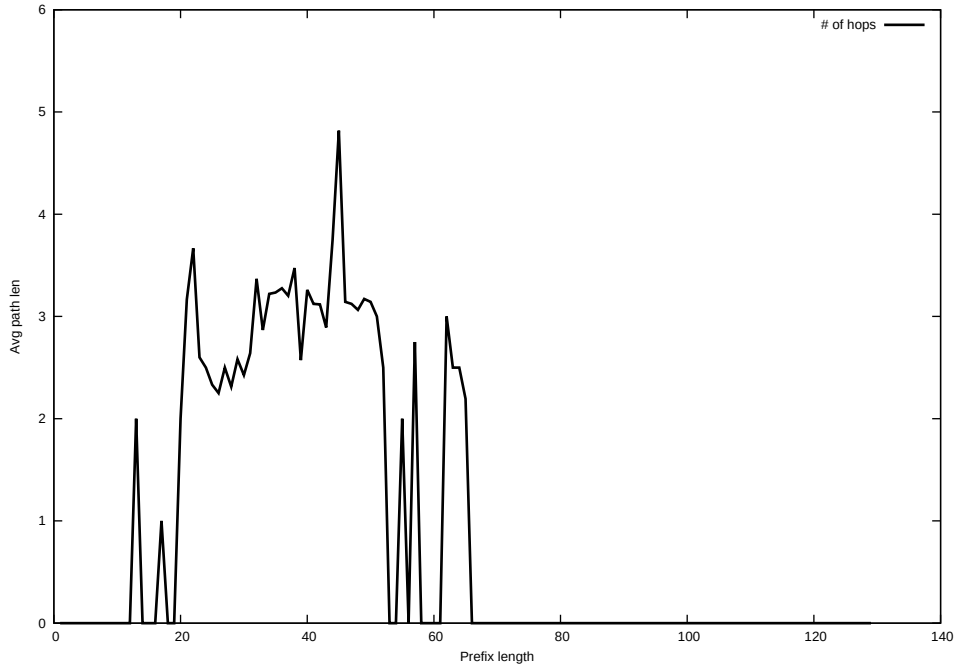
2012-12-24



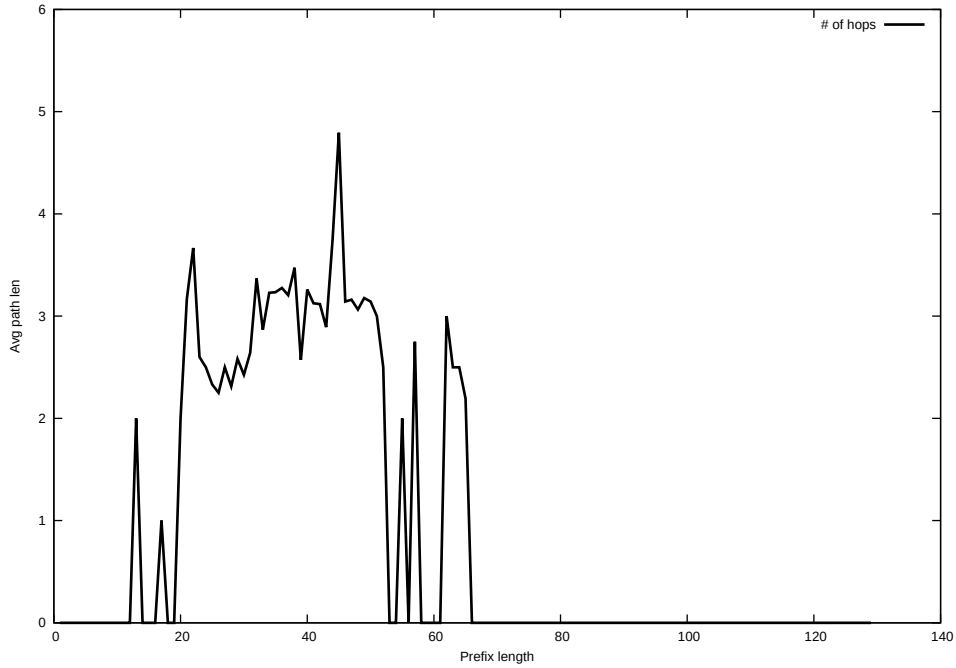
2012-12-25



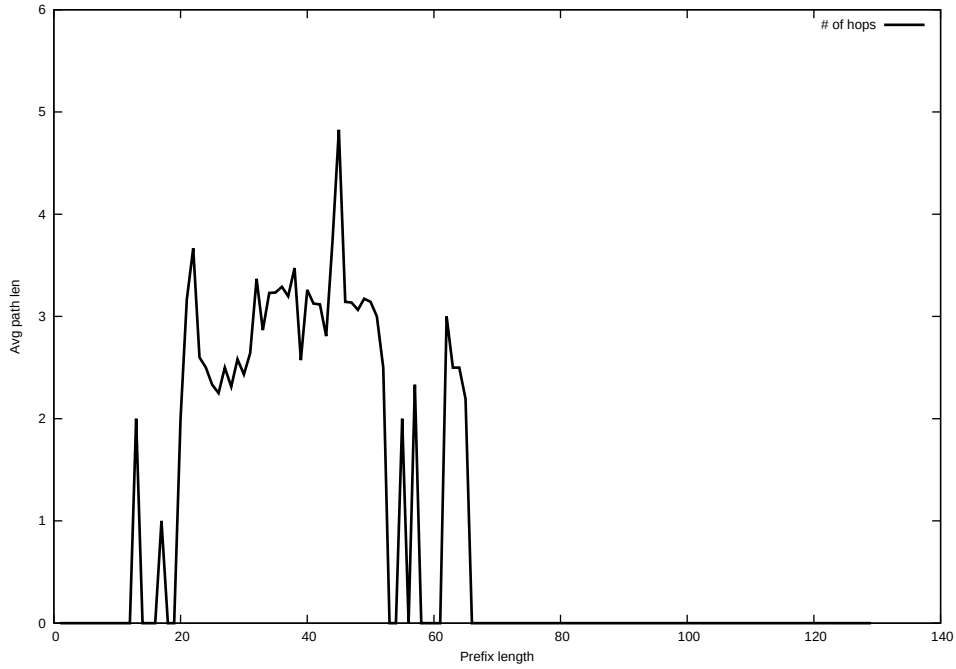
2012-12-26



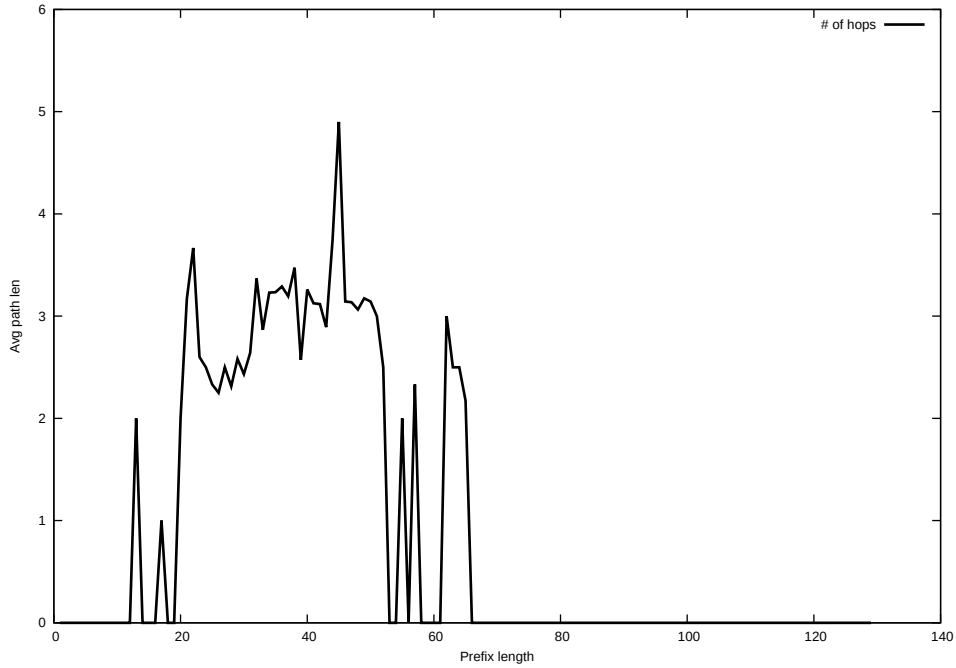
2012-12-27



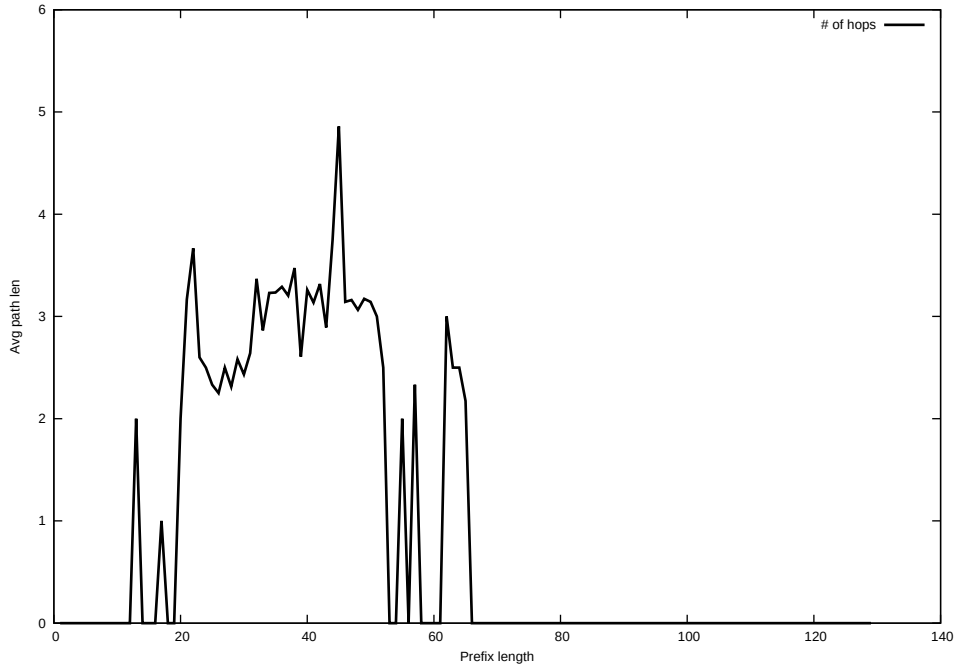
2012-12-28



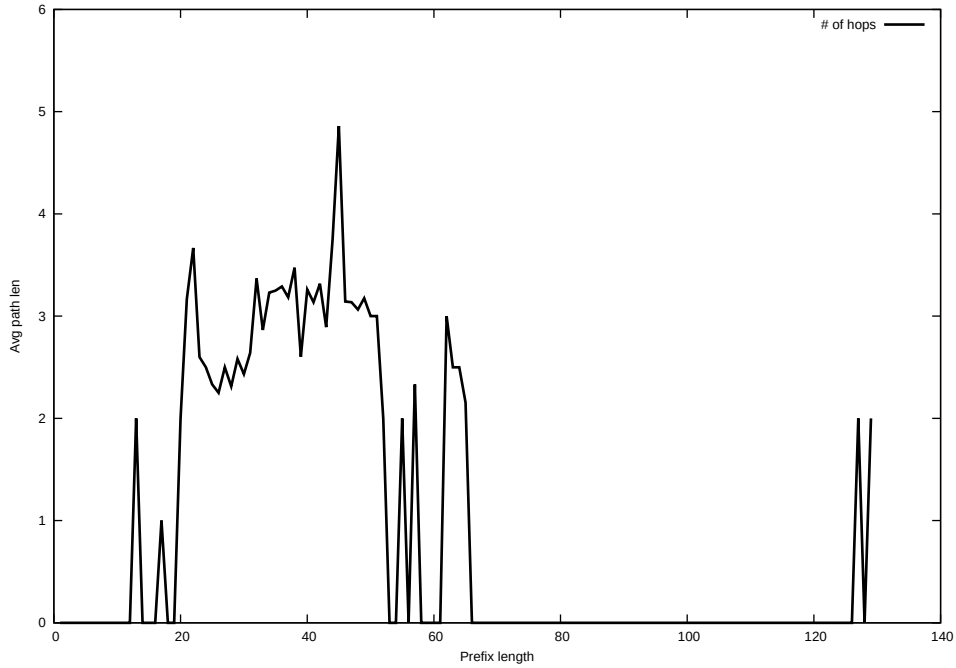
2012-12-29



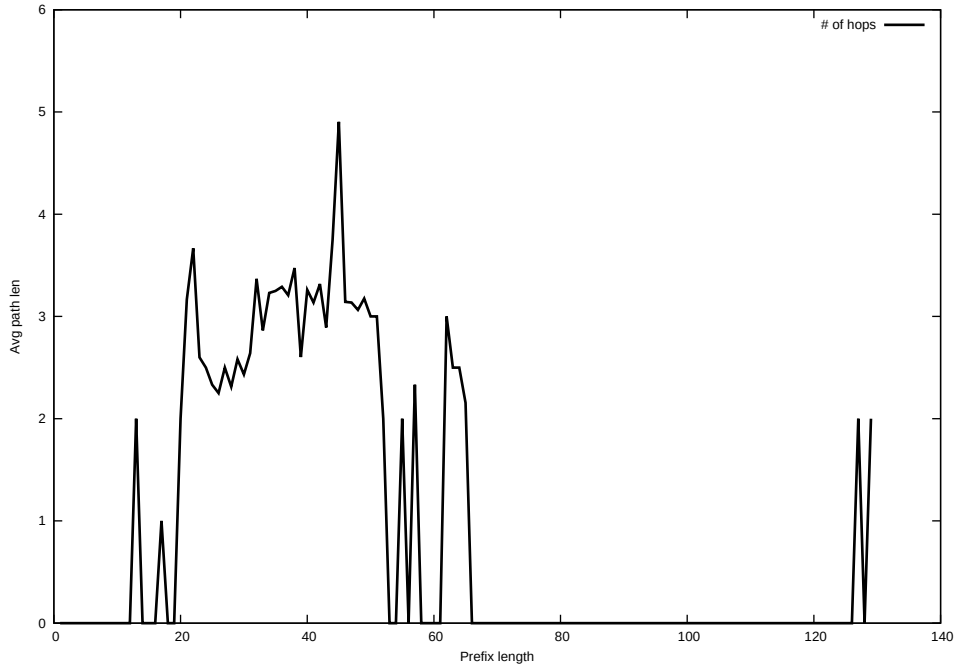
2012-12-30



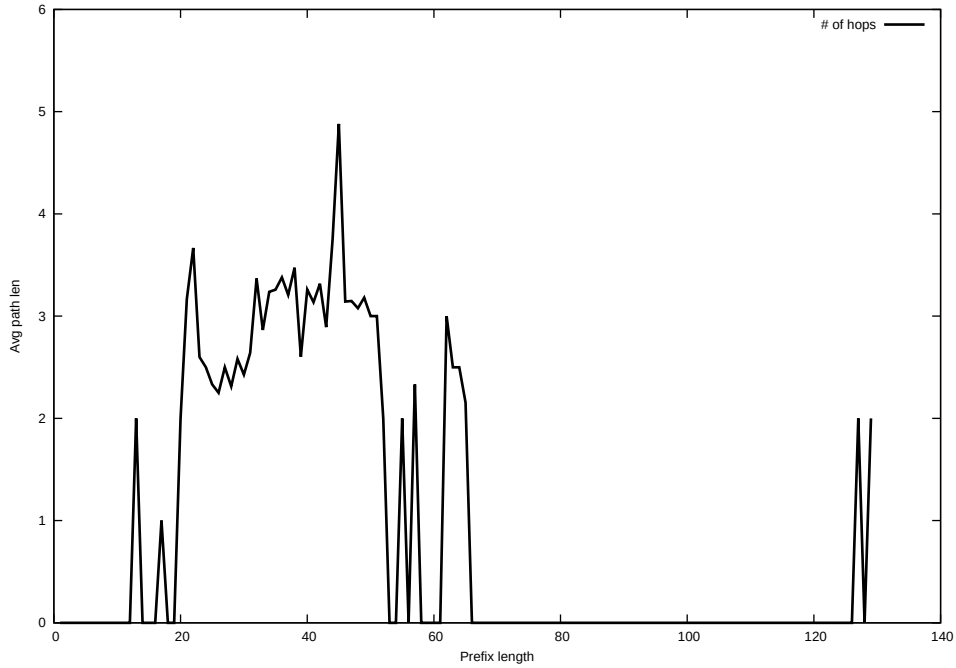
2012-12-31



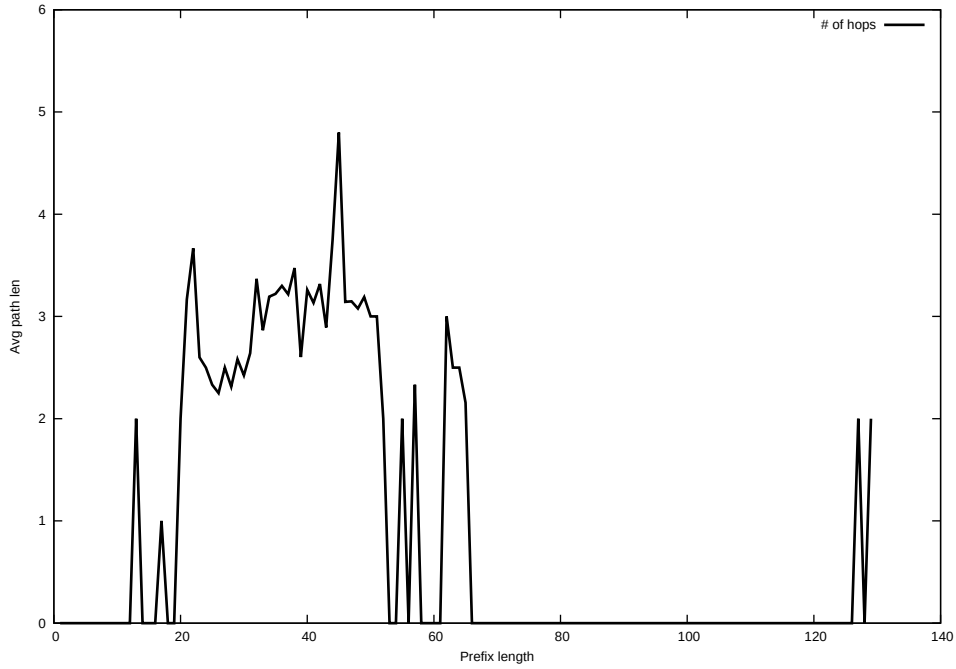
2013-01-01



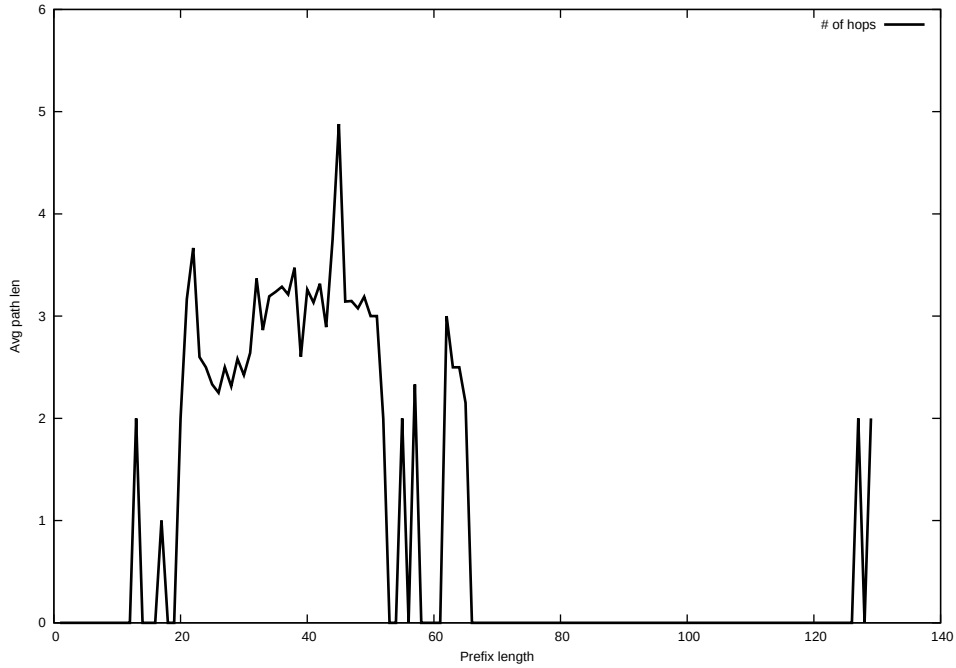
2013-01-02



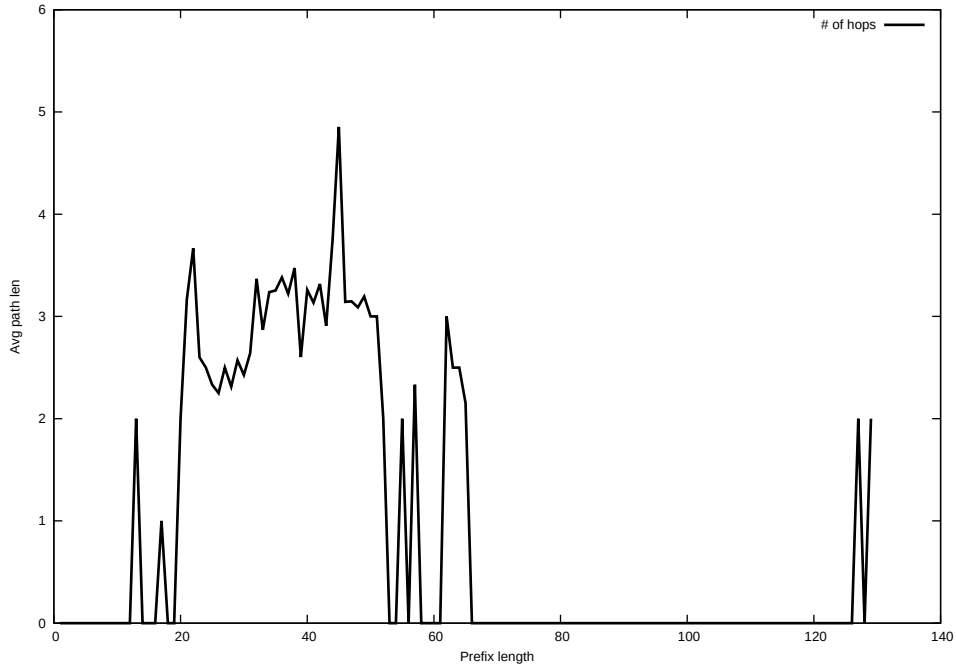
2013-01-03



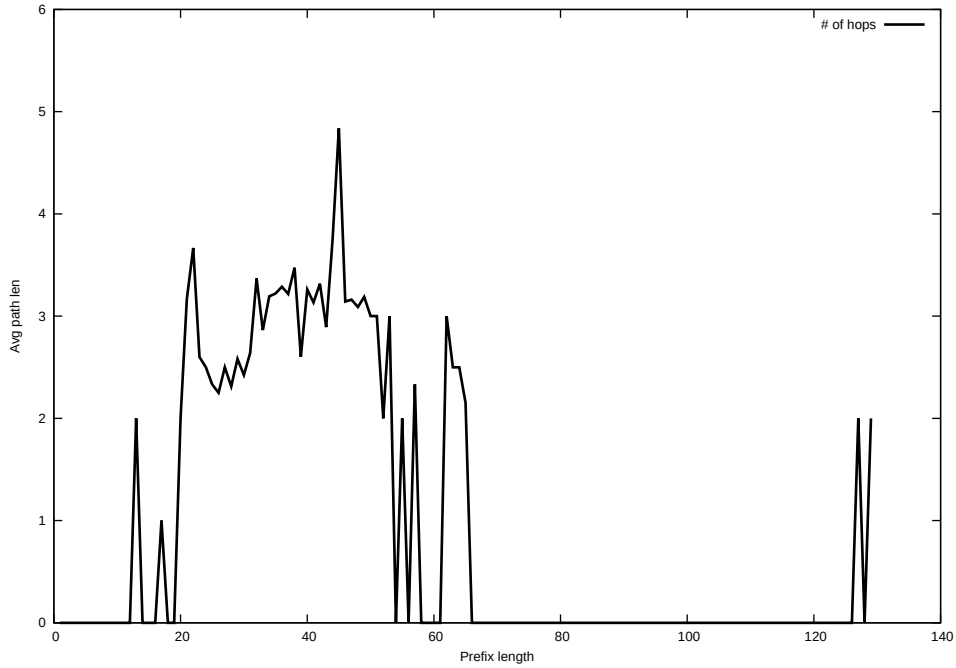
2013-01-04



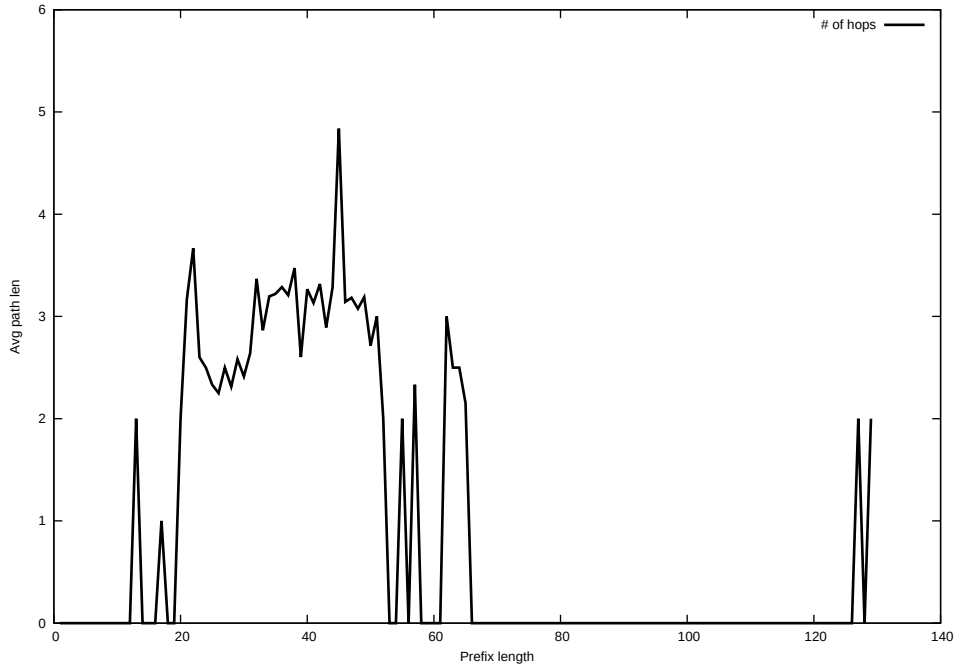
2013-01-05



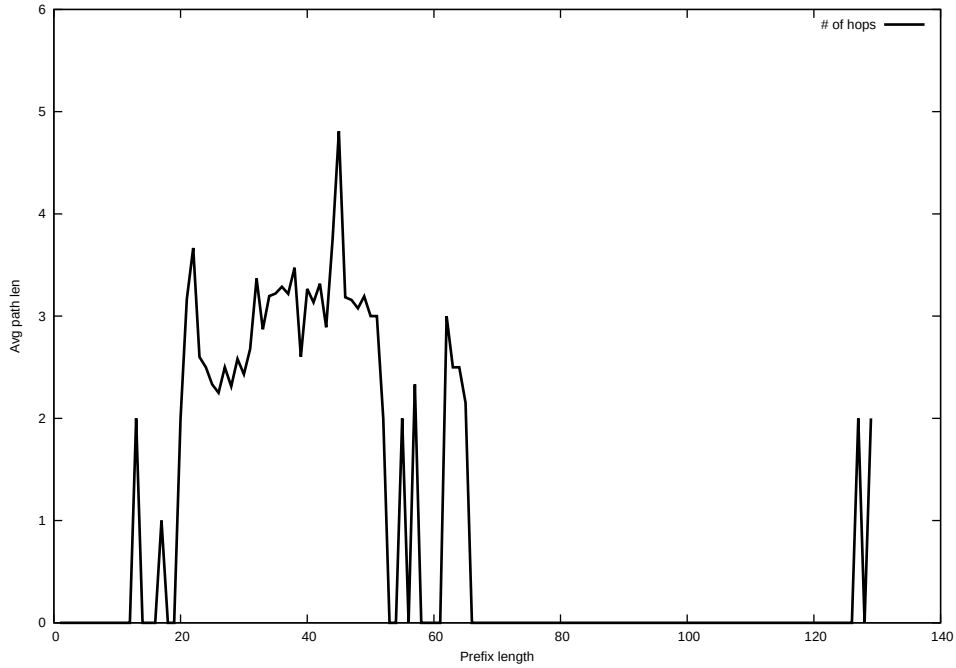
2013-01-06



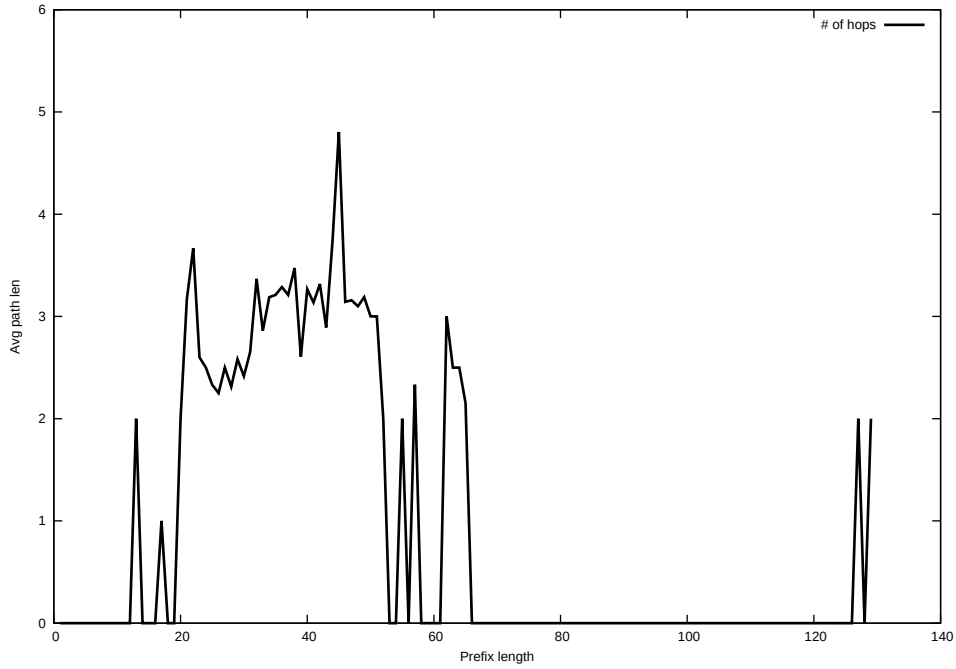
2013-01-07



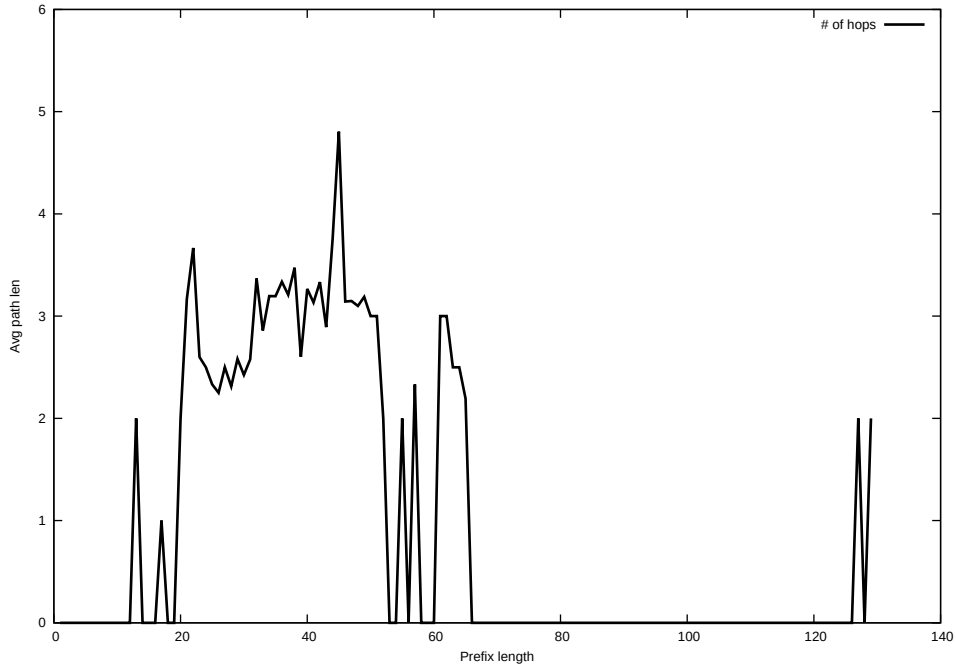
2013-01-08



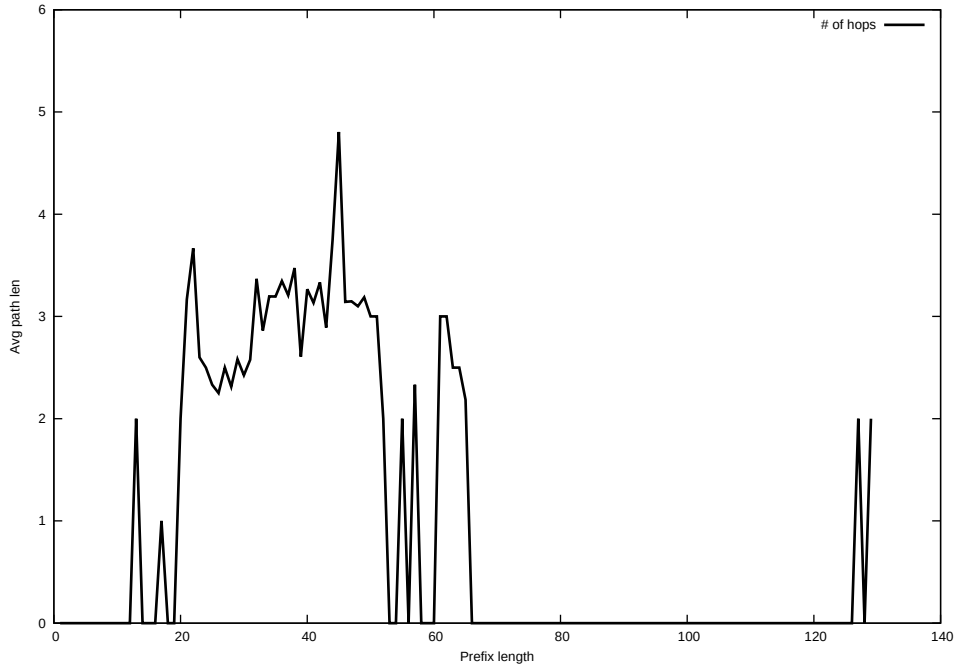
2013-01-09



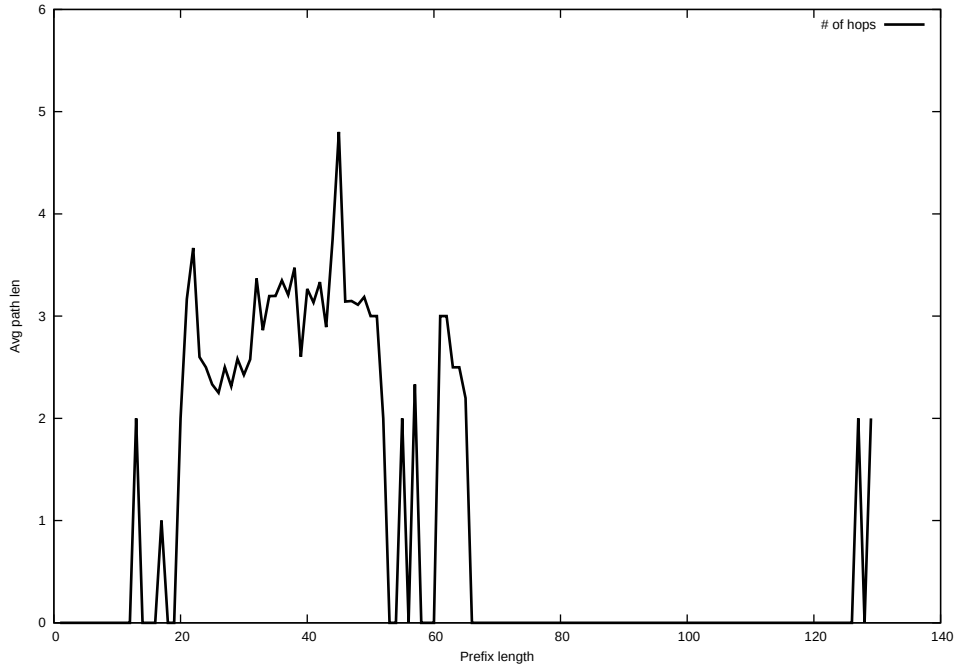
2013-01-10



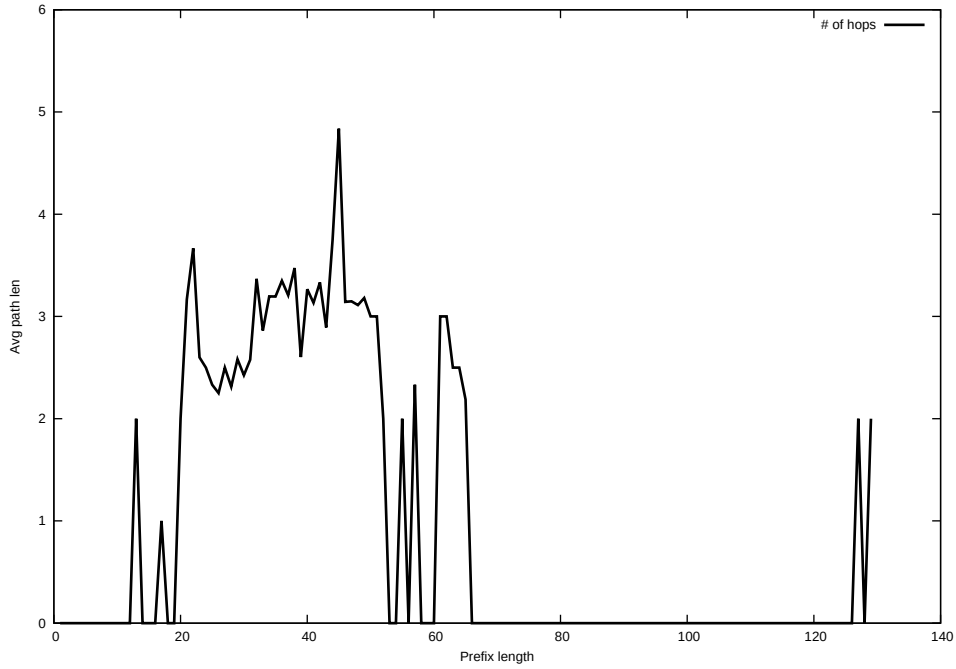
2013-01-11



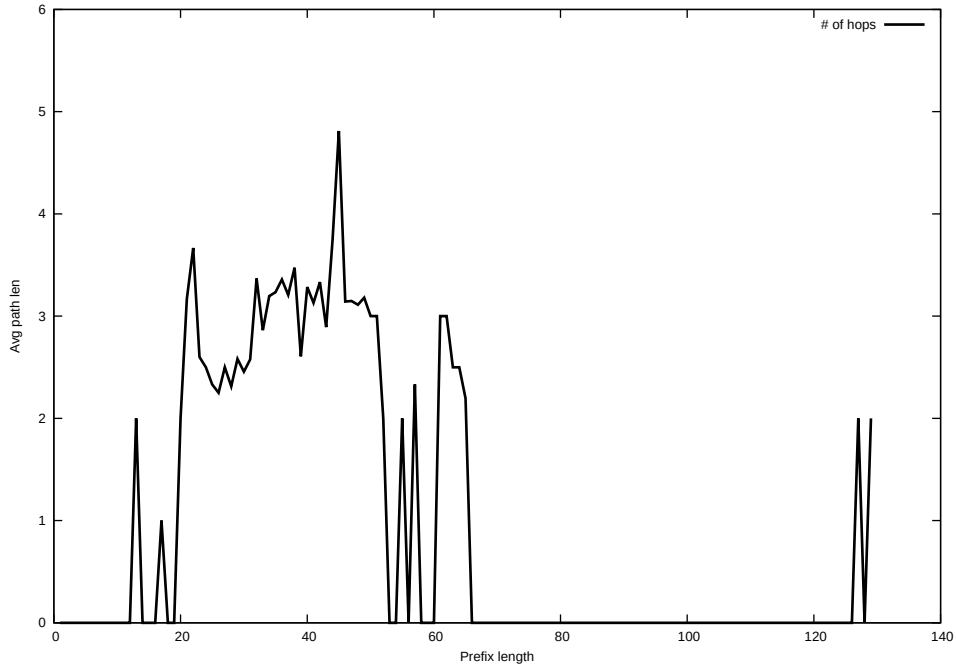
2013-01-12



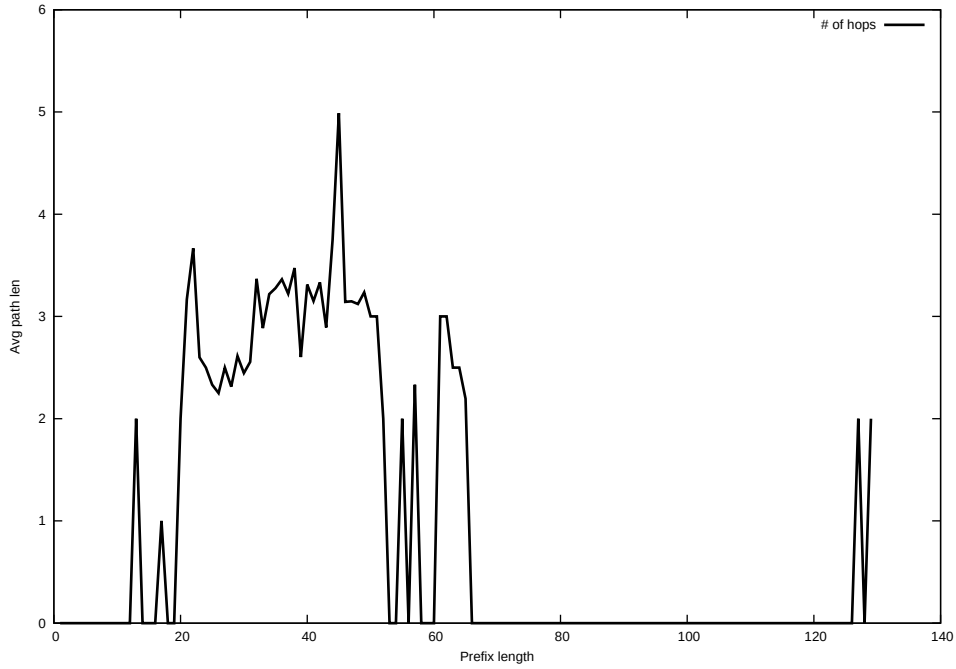
2013-01-13



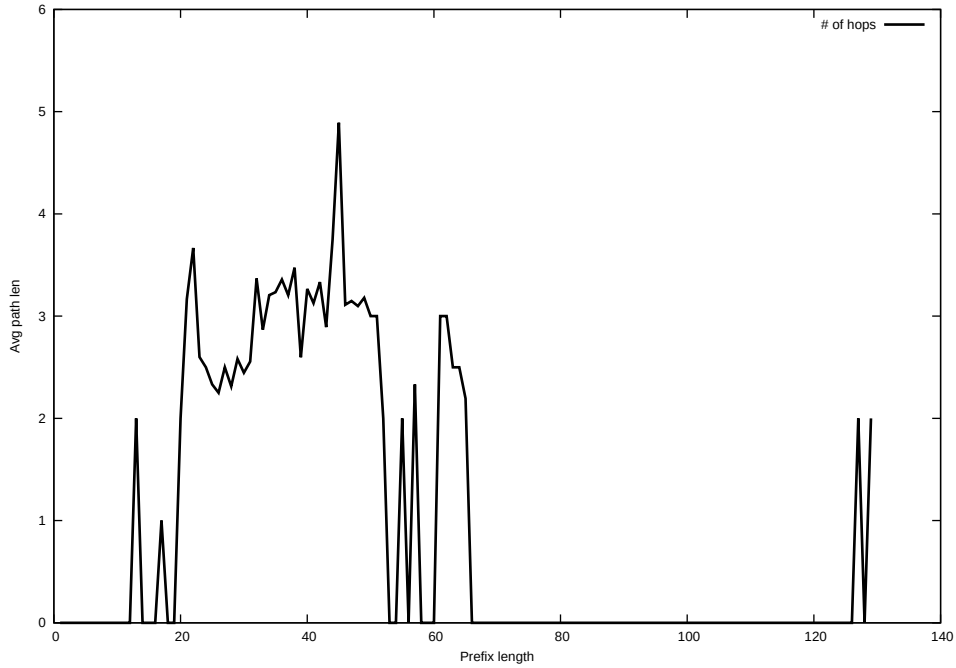
2013-01-14



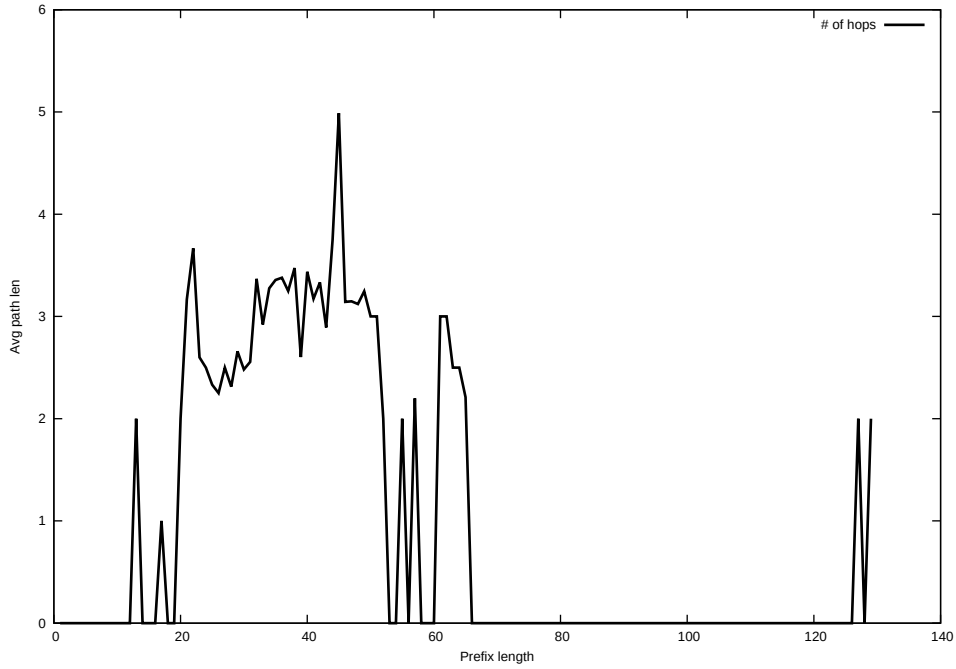
2013-01-15



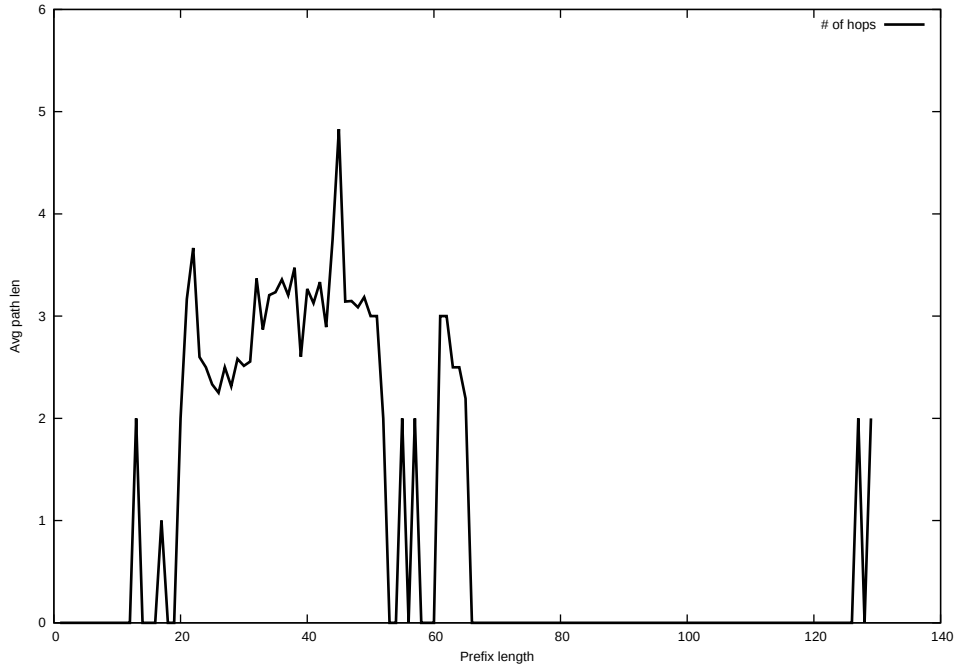
2013-01-16



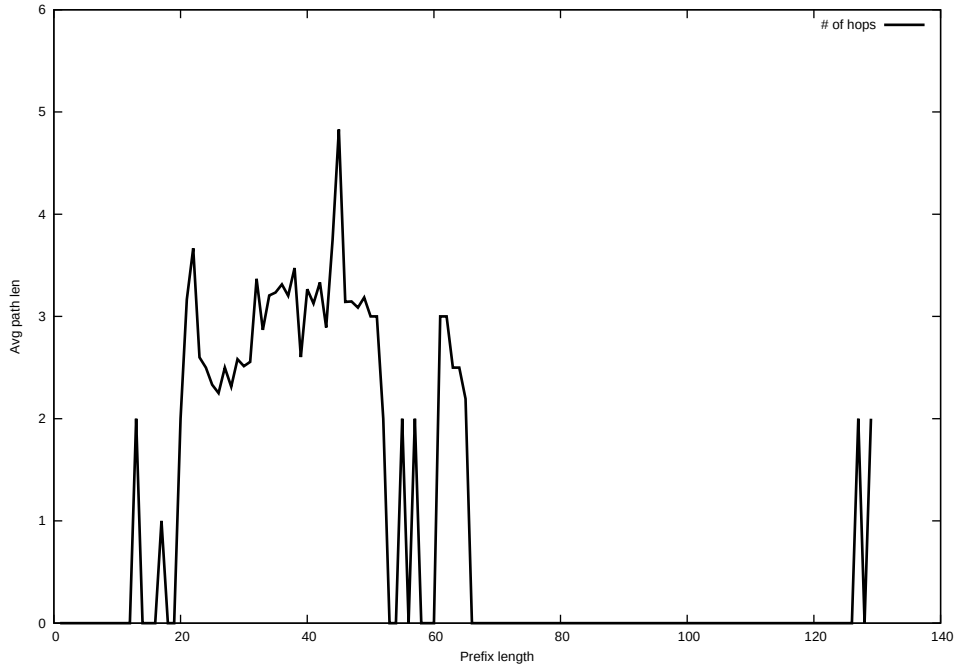
2013-01-17



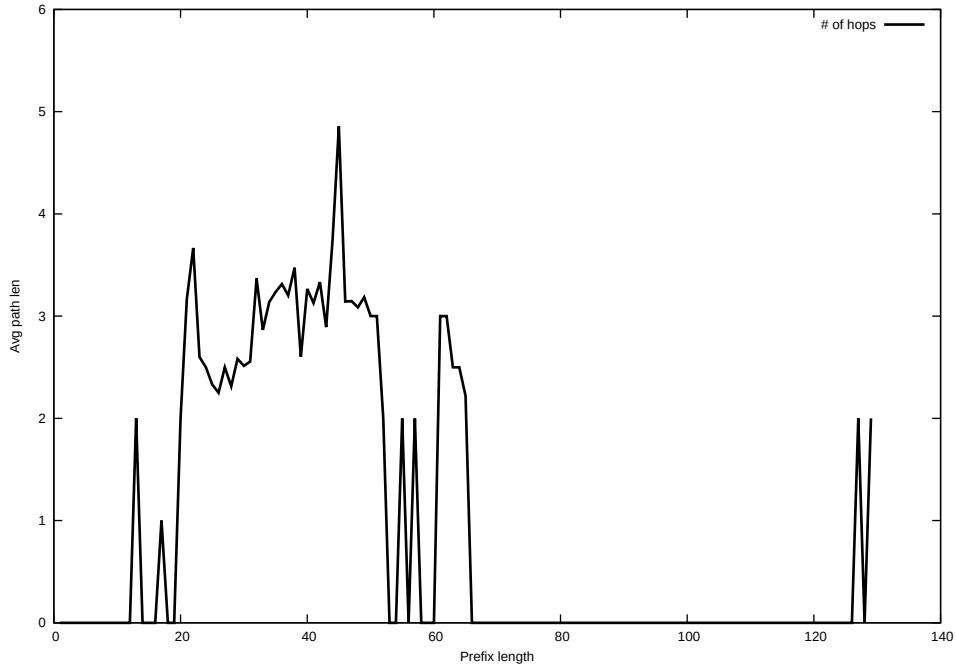
2013-01-18



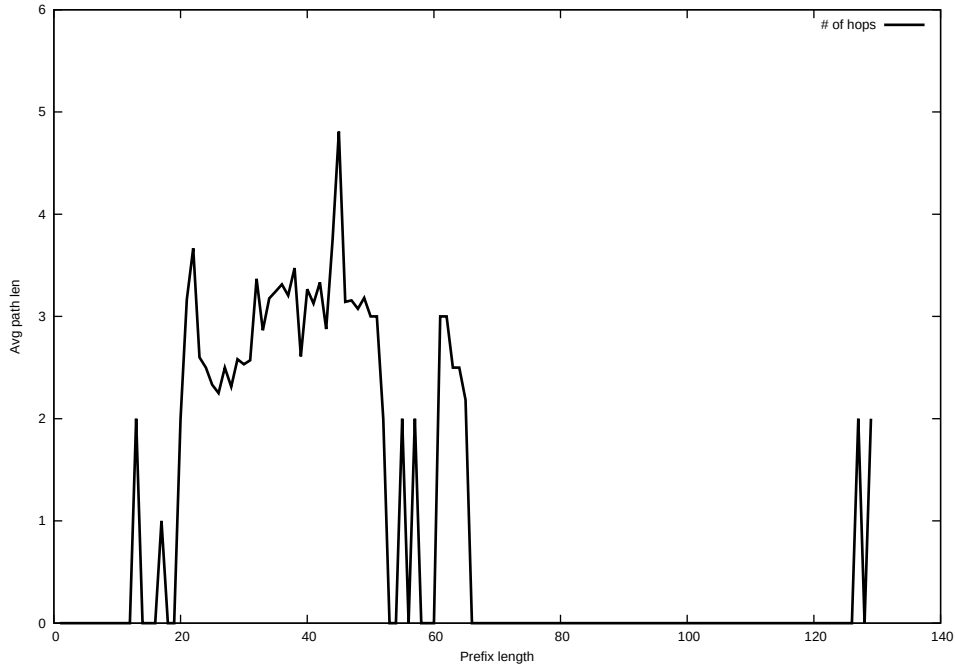
2013-01-19



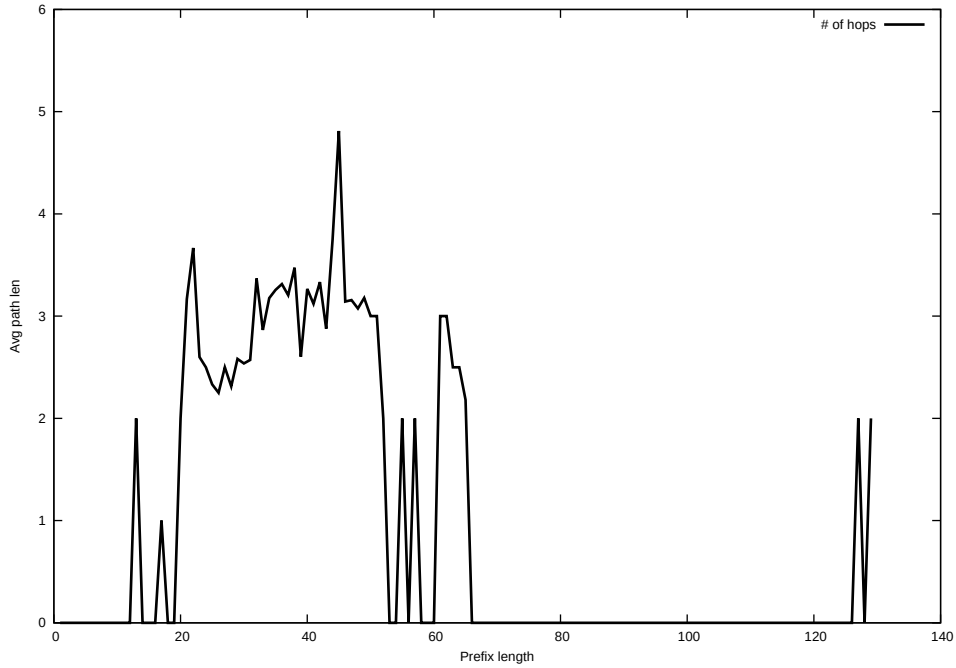
2013-01-20



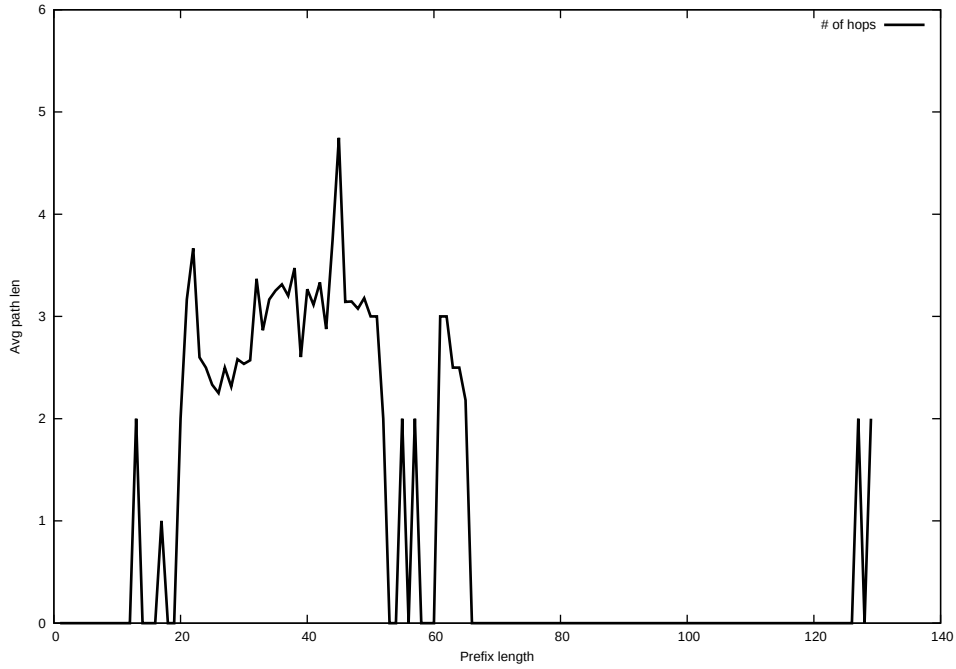
2013-01-21



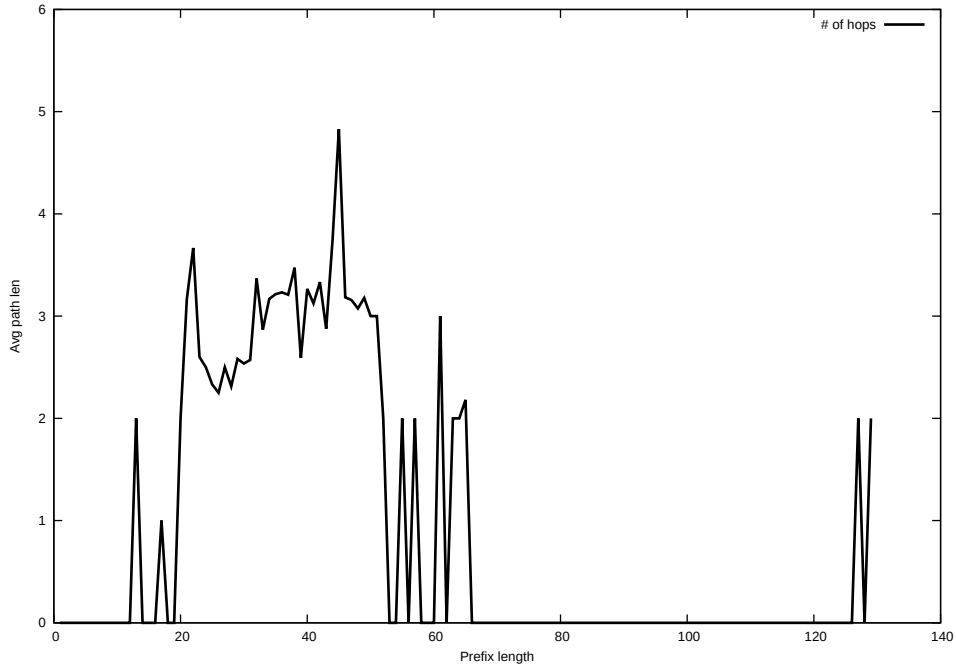
2013-01-22



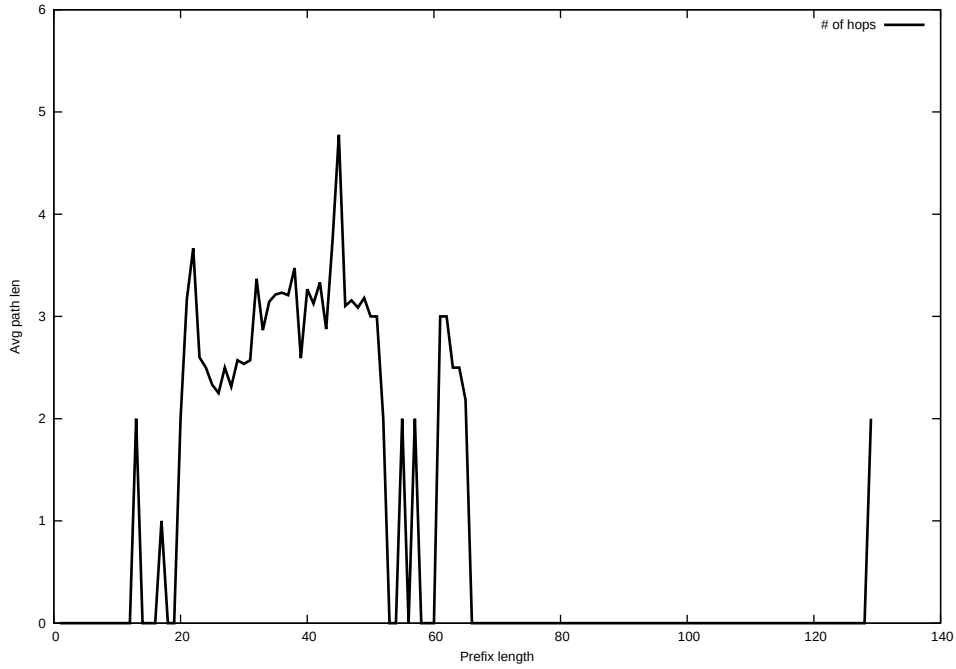
2013-01-23



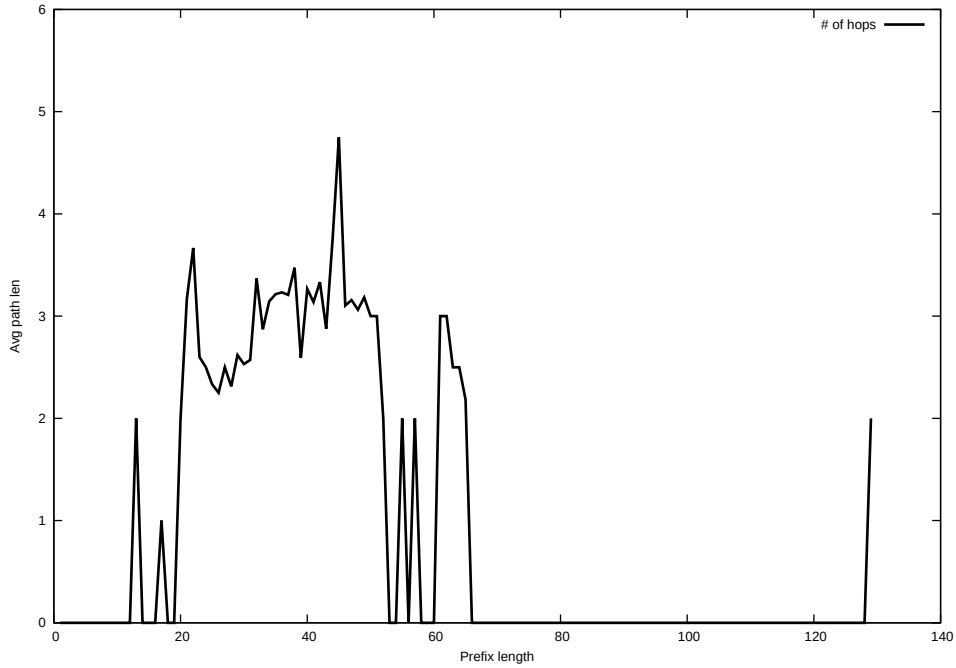
2013-01-24



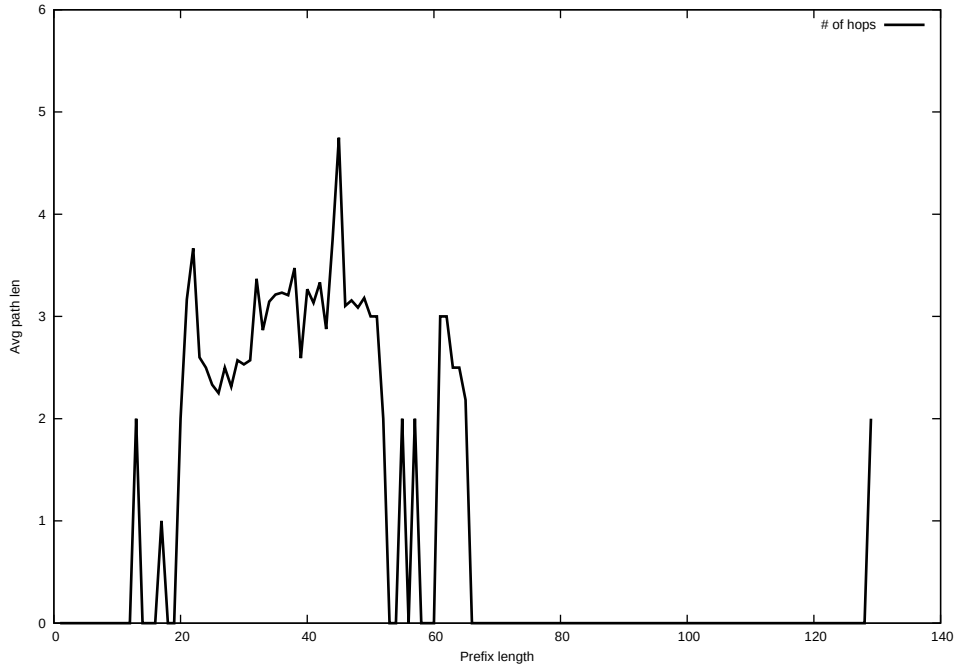
2013-01-25



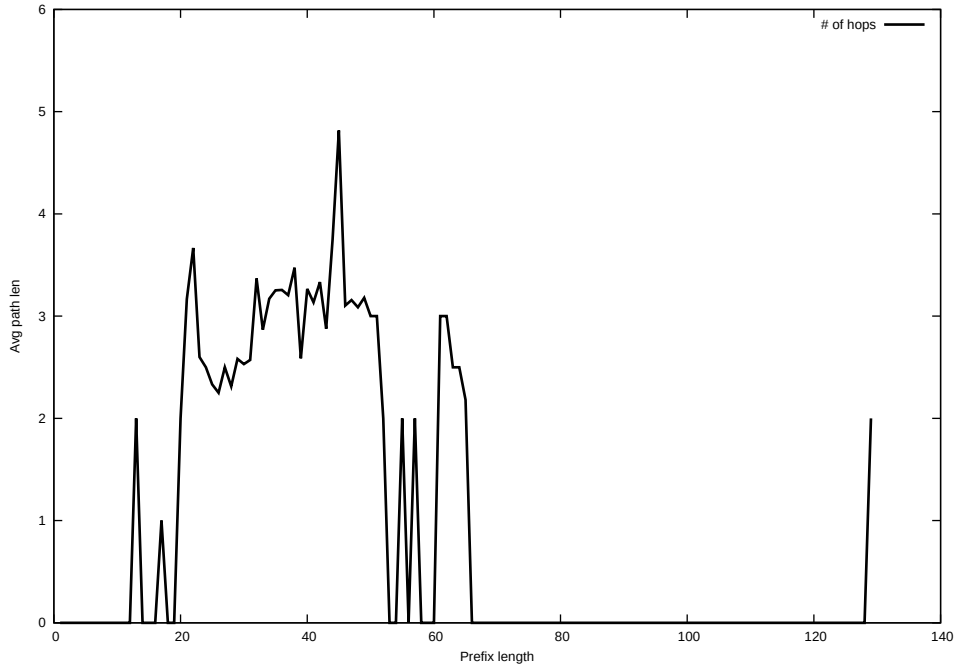
2013-01-26



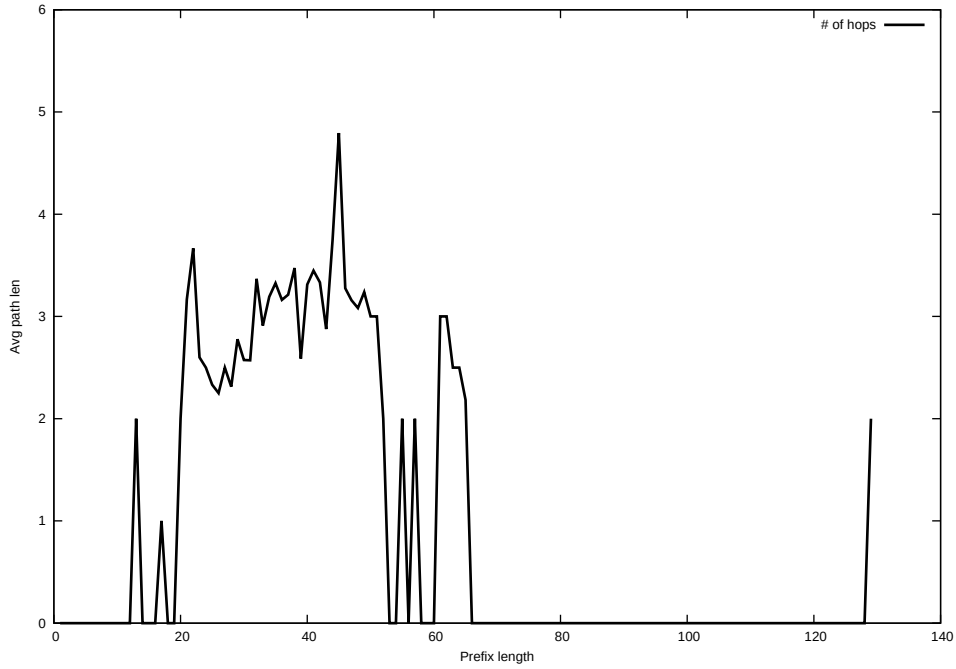
2013-01-27



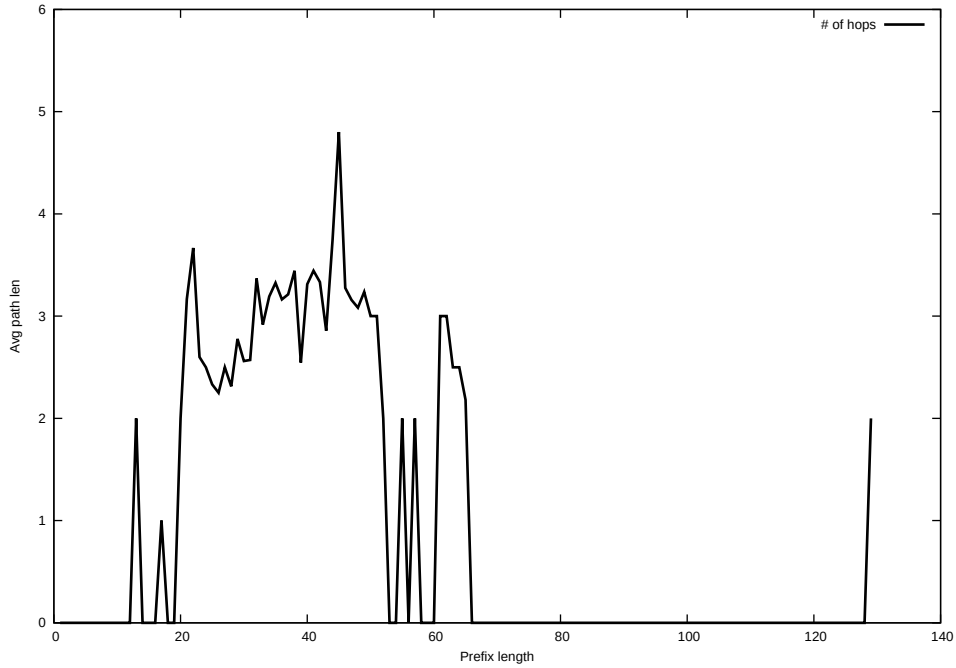
2013-01-28



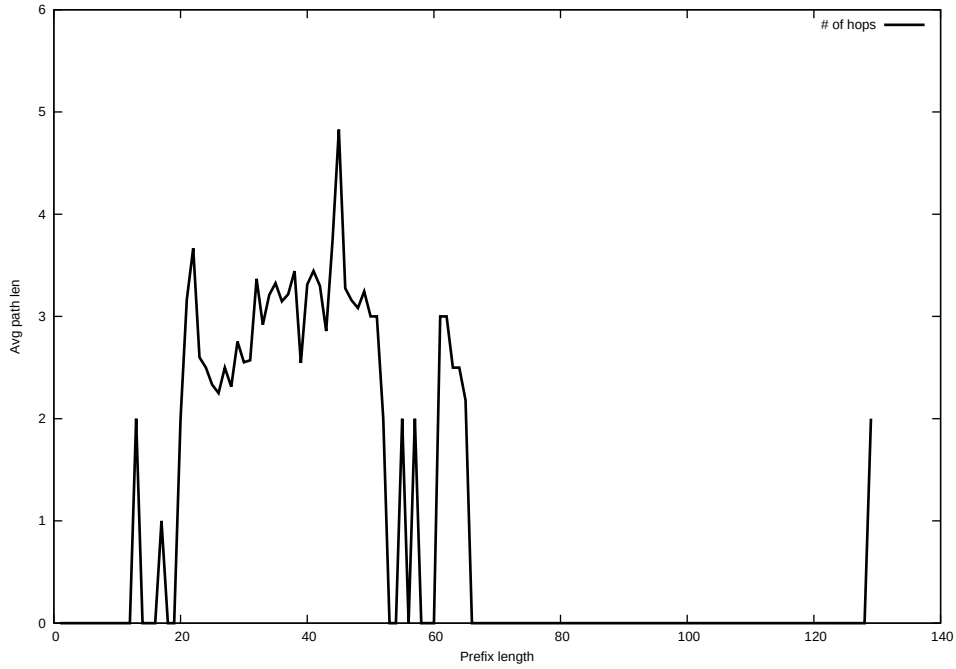
2013-01-29



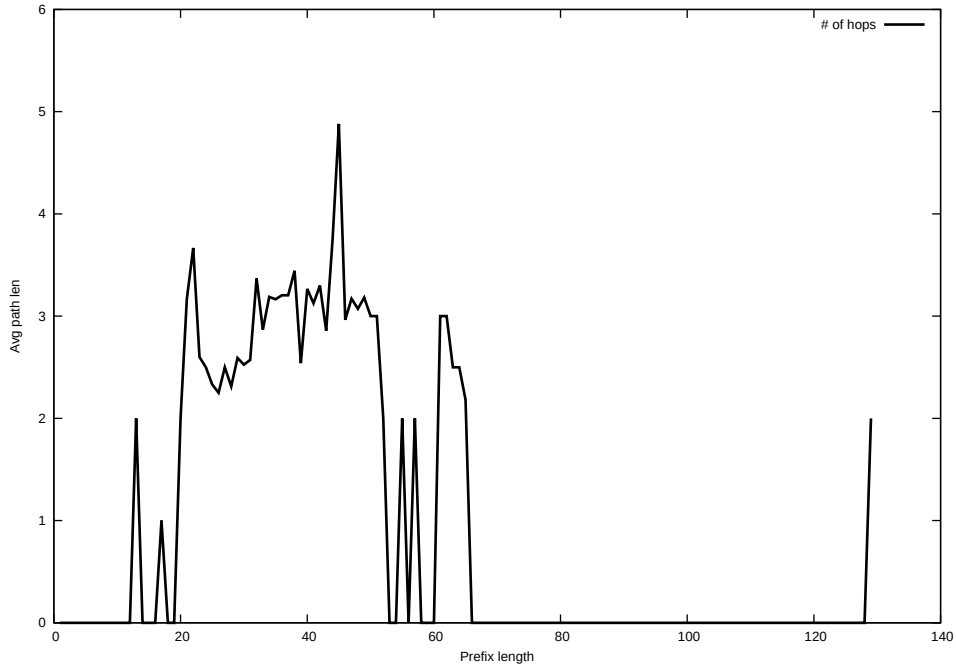
2013-01-30



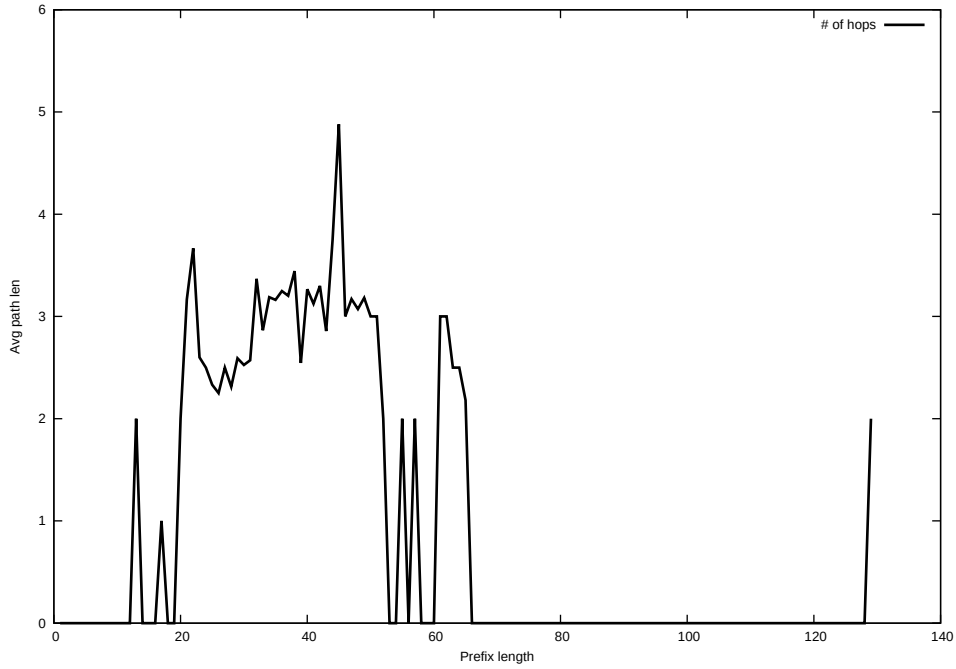
2013-01-31



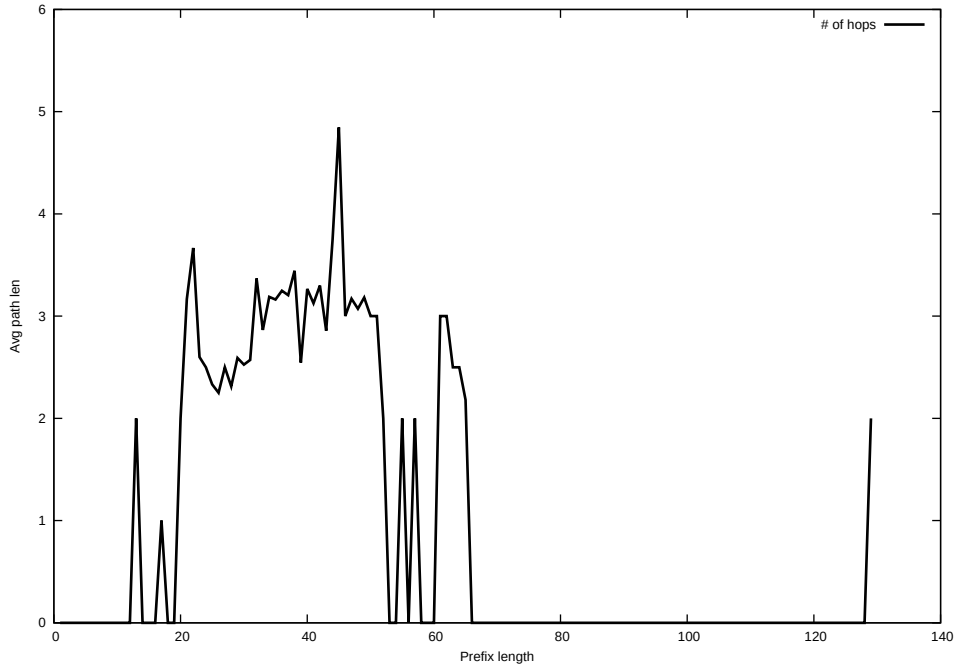
2013-02-01



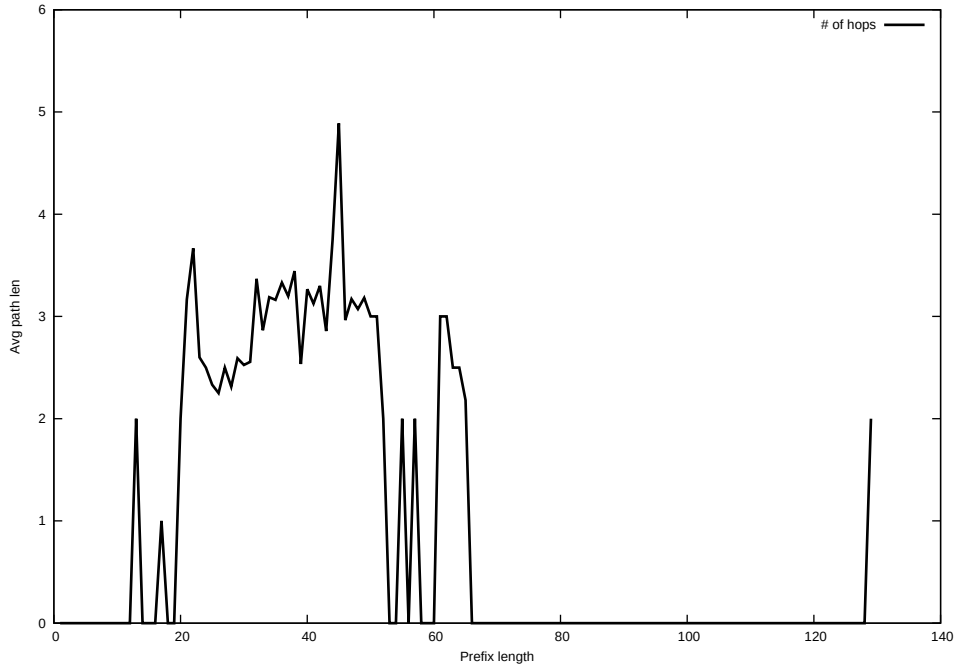
2013-02-02



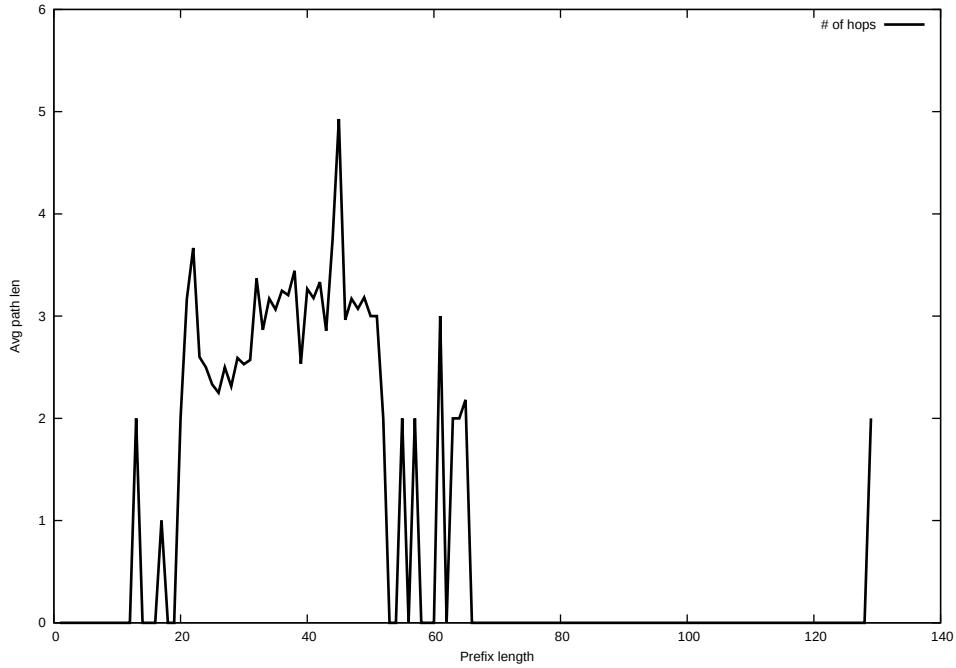
2013-02-03



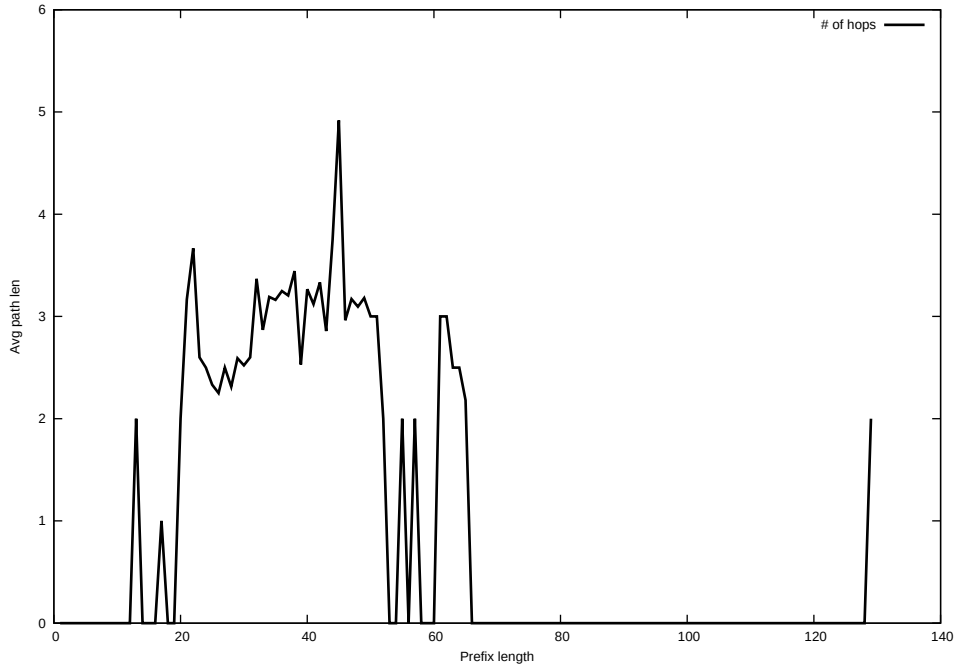
2013-02-04



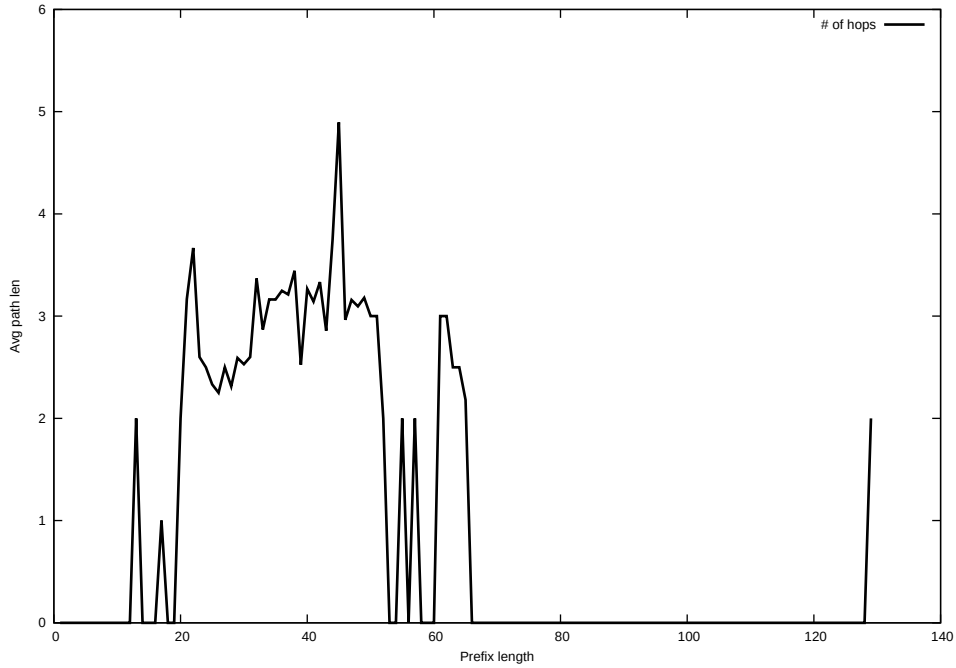
2013-02-05



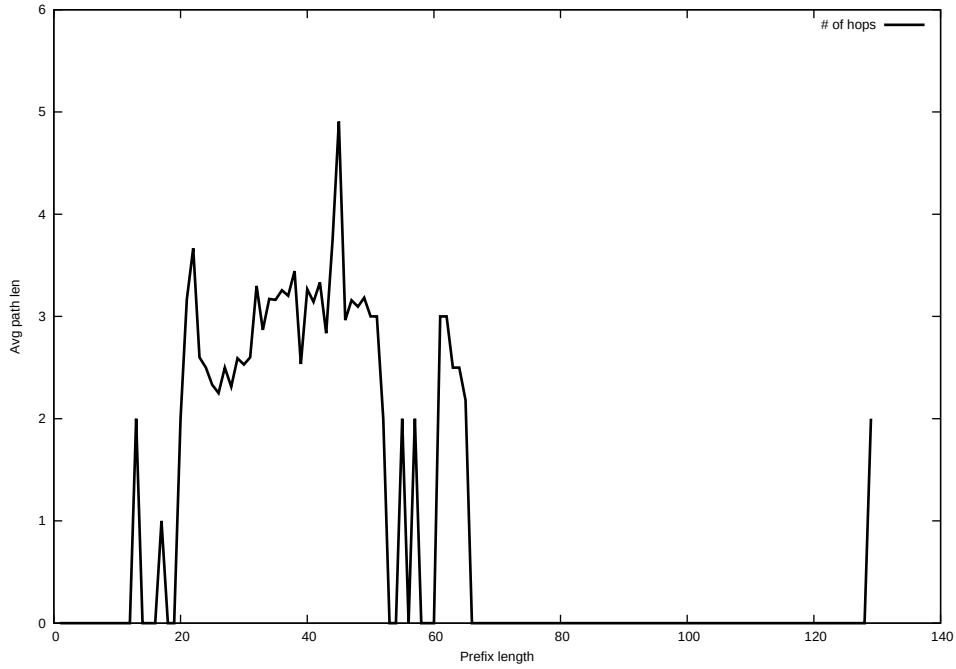
2013-02-06



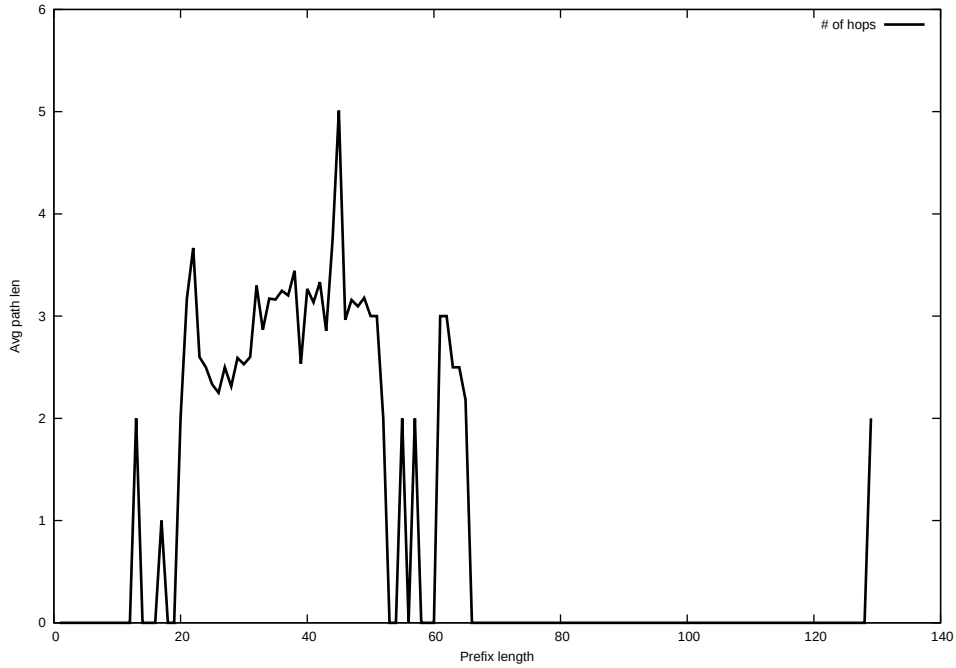
2013-02-07



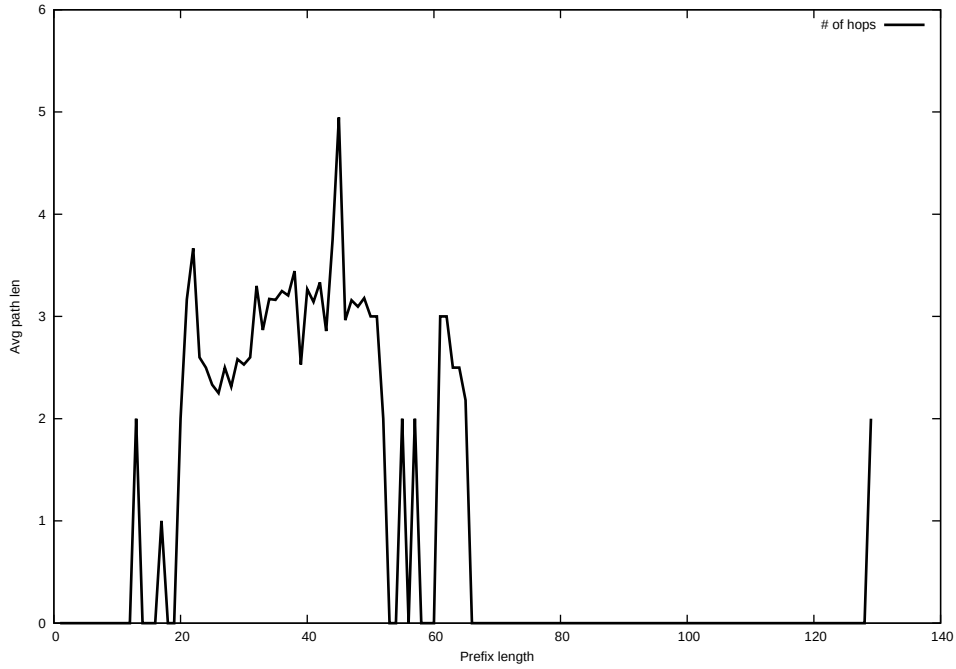
2013-02-08



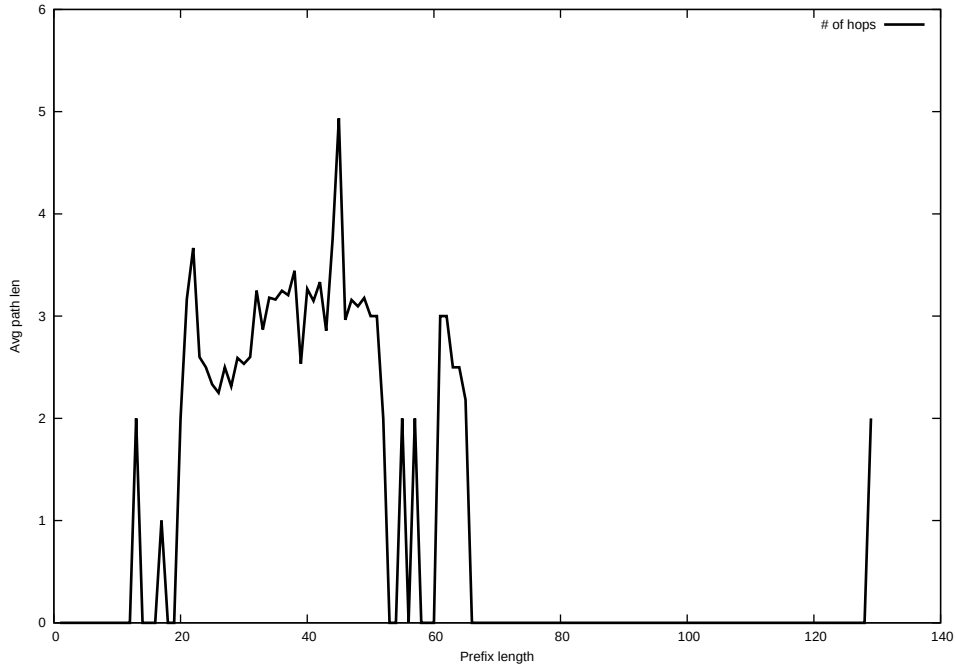
2013-02-09



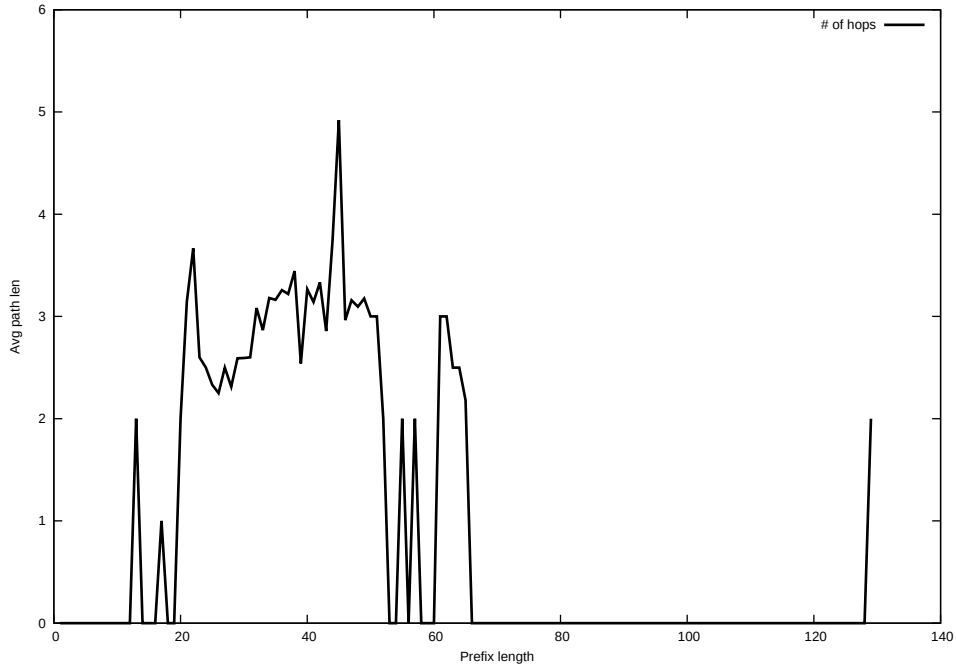
2013-02-10



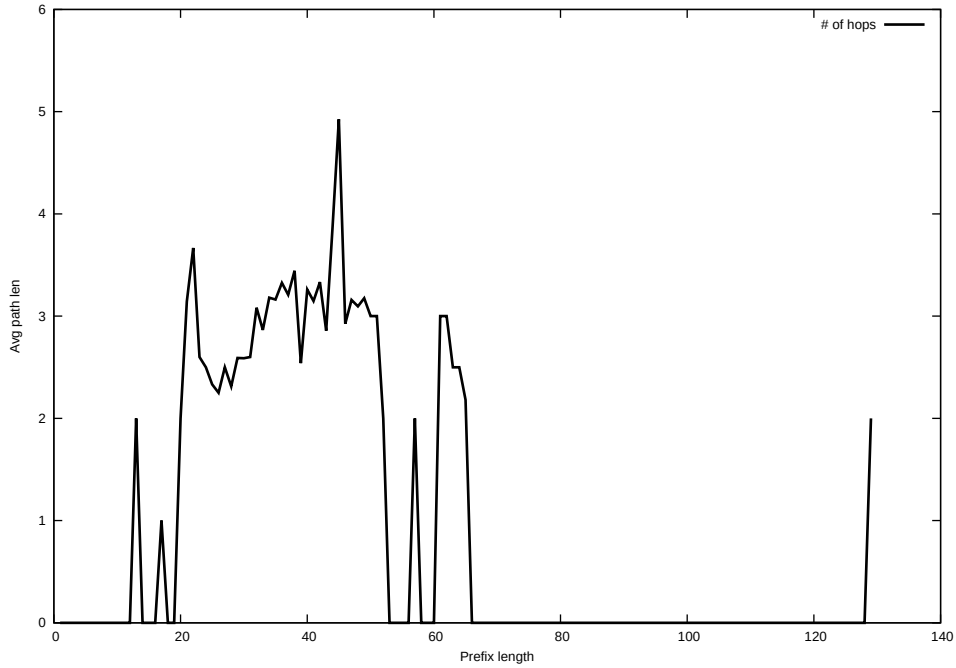
2013-02-11



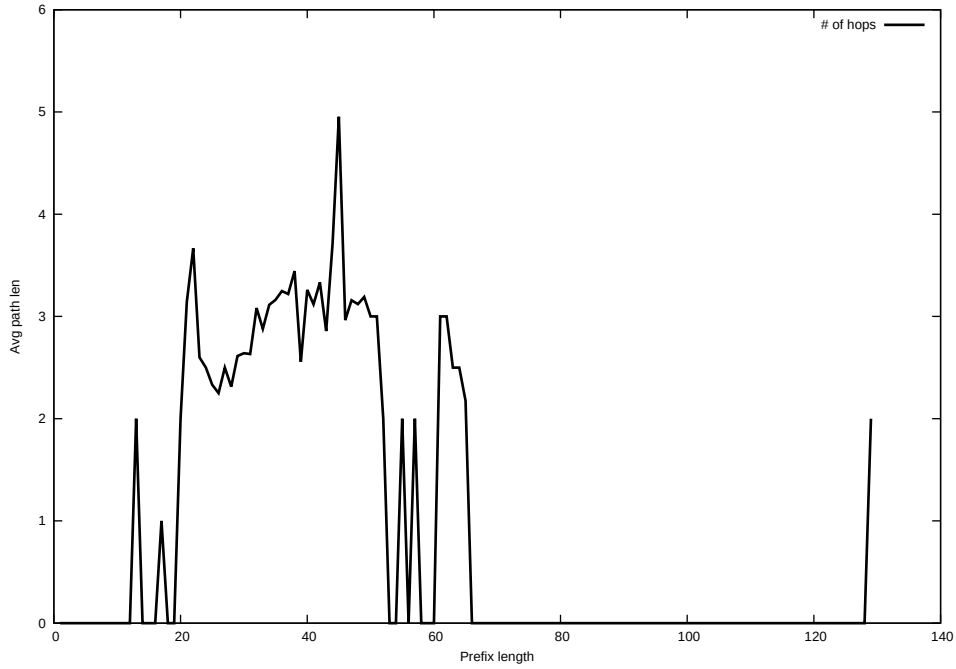
2013-02-12



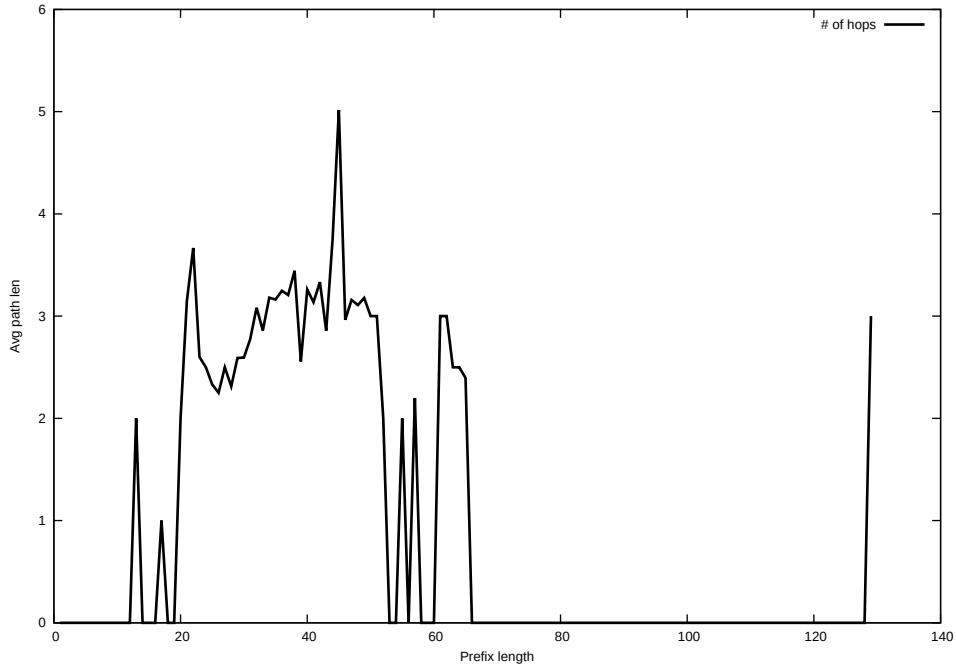
2013-02-13



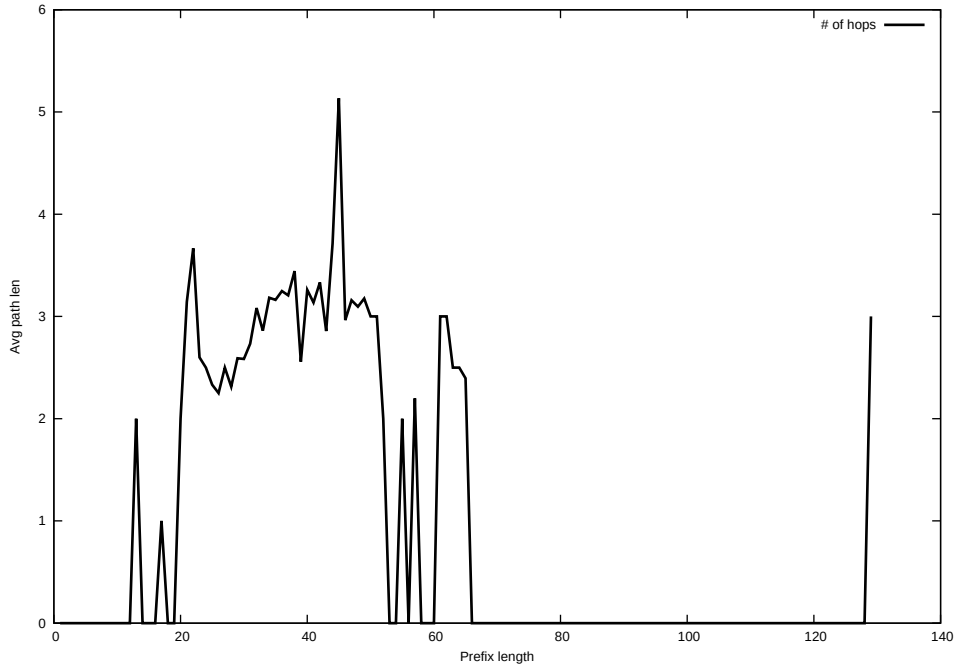
2013-02-14



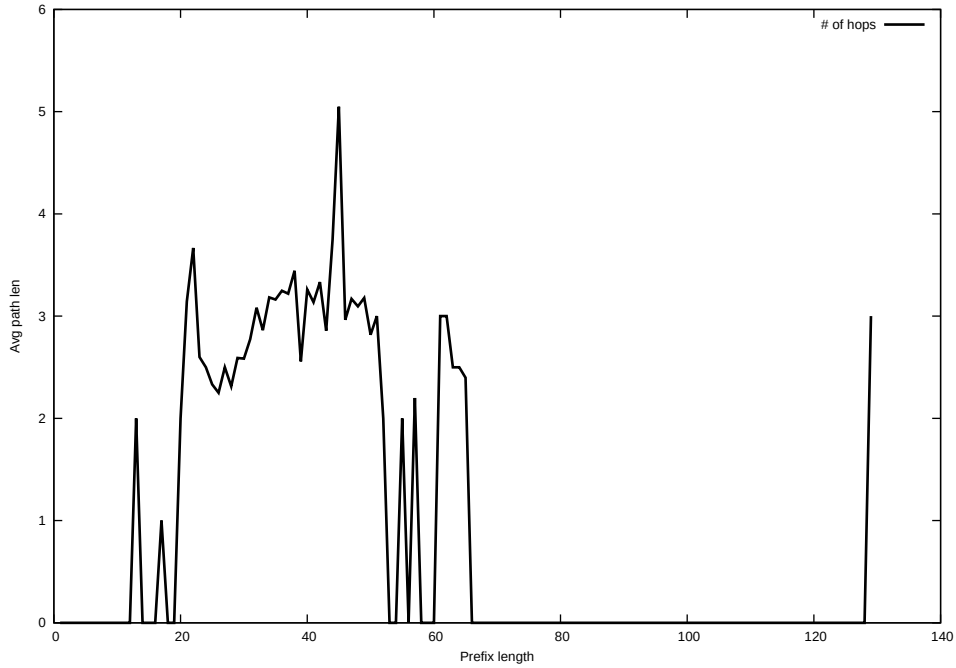
2013-02-15



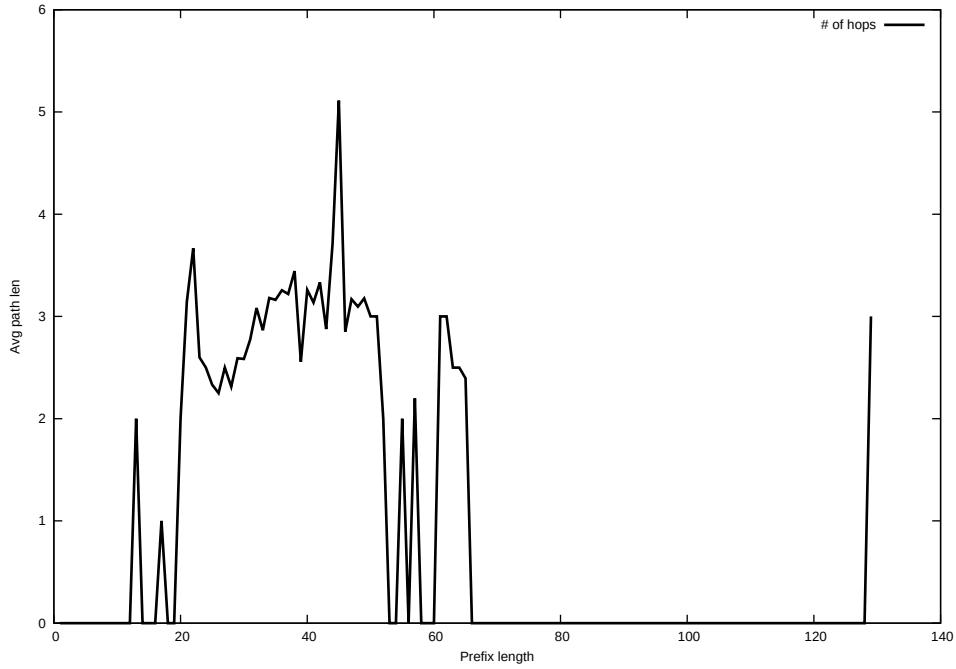
2013-02-16



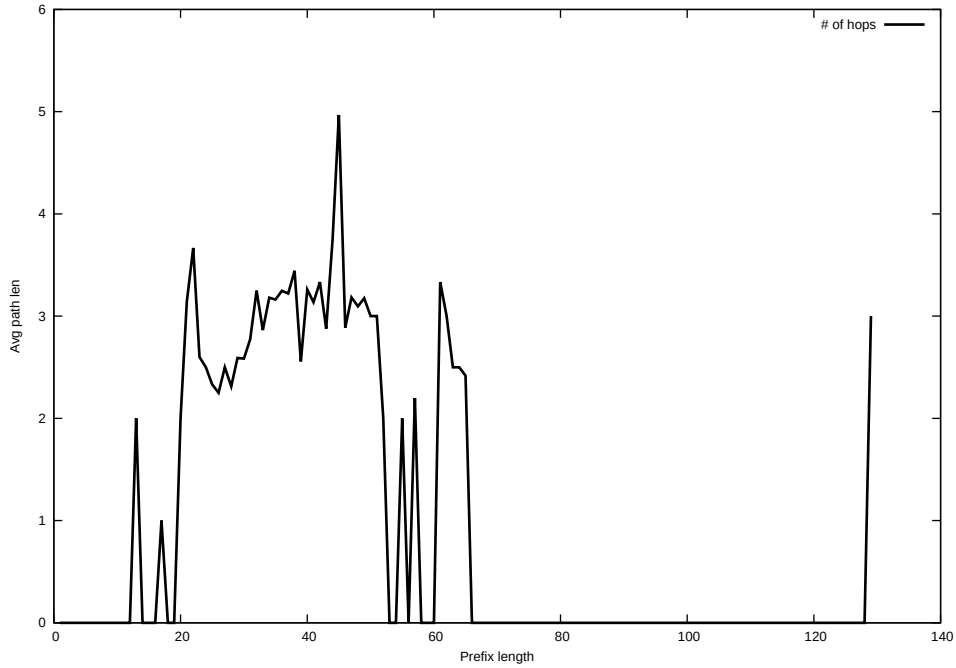
2013-02-17



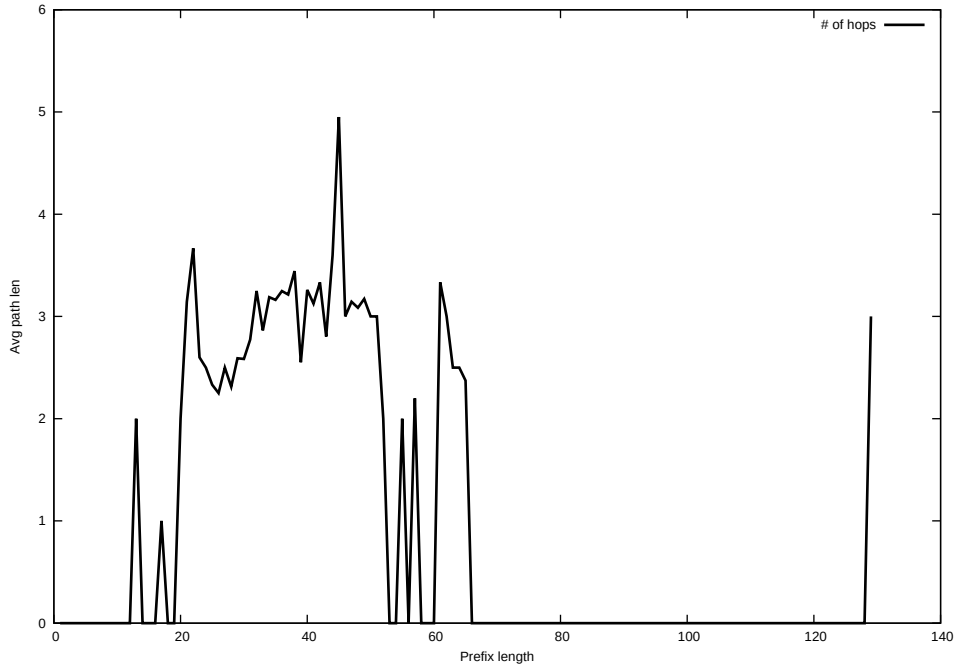
2013-02-18



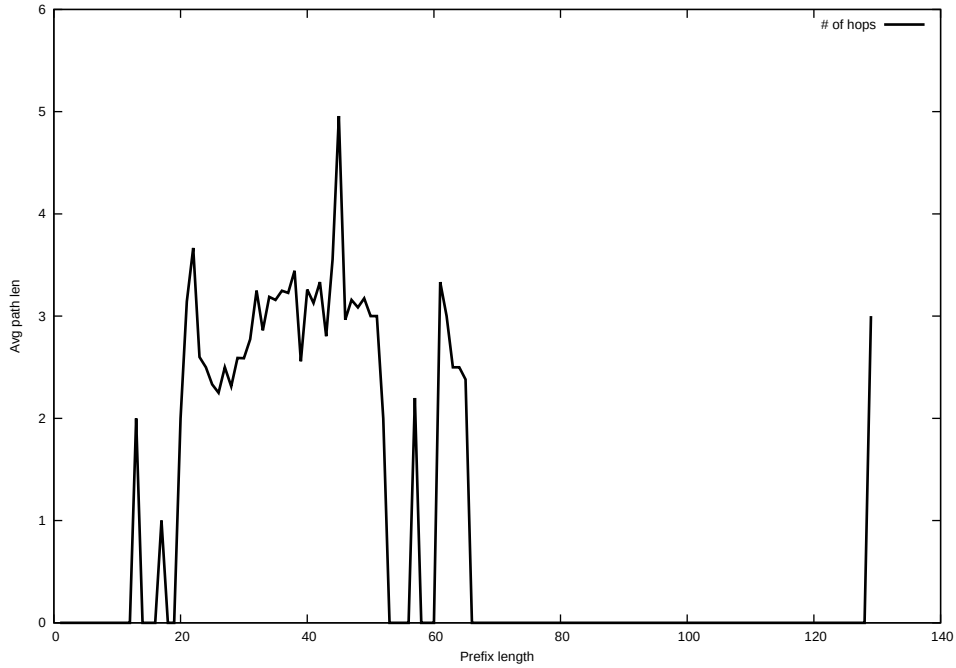
2013-02-19



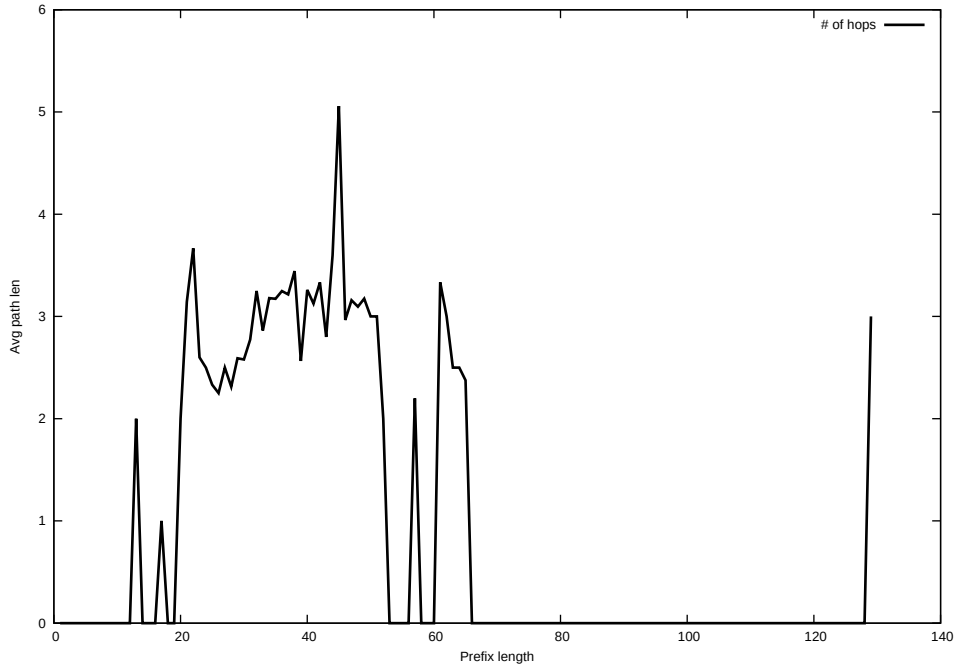
2013-02-20



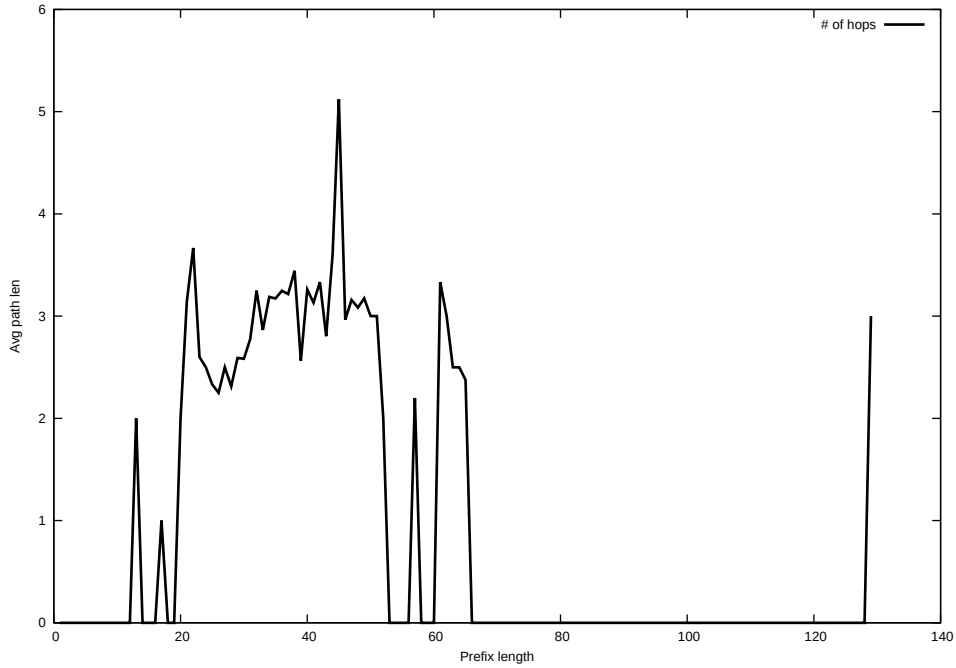
2013-02-21



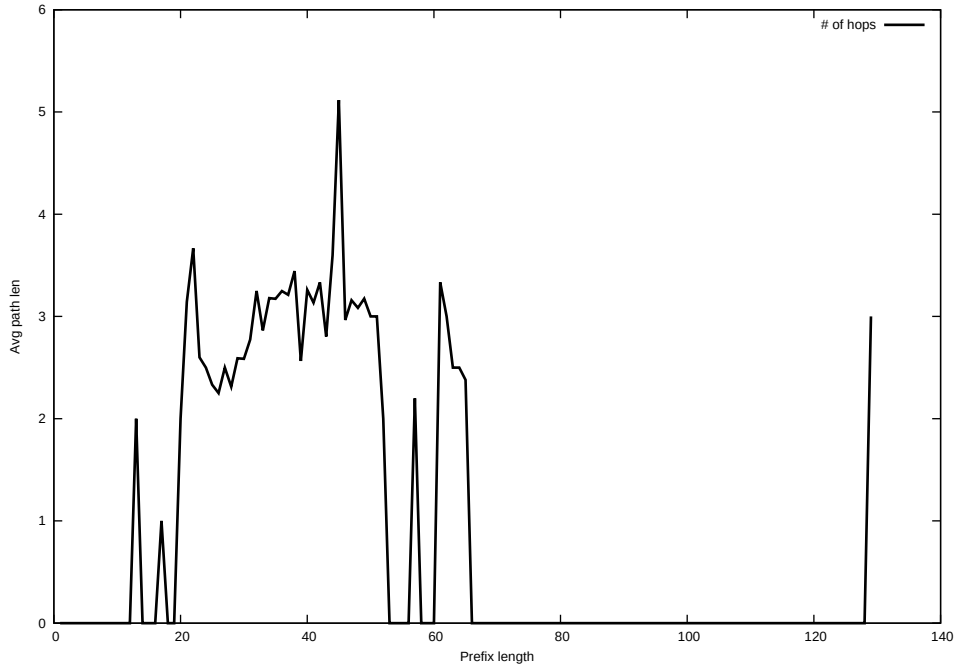
2013-02-22



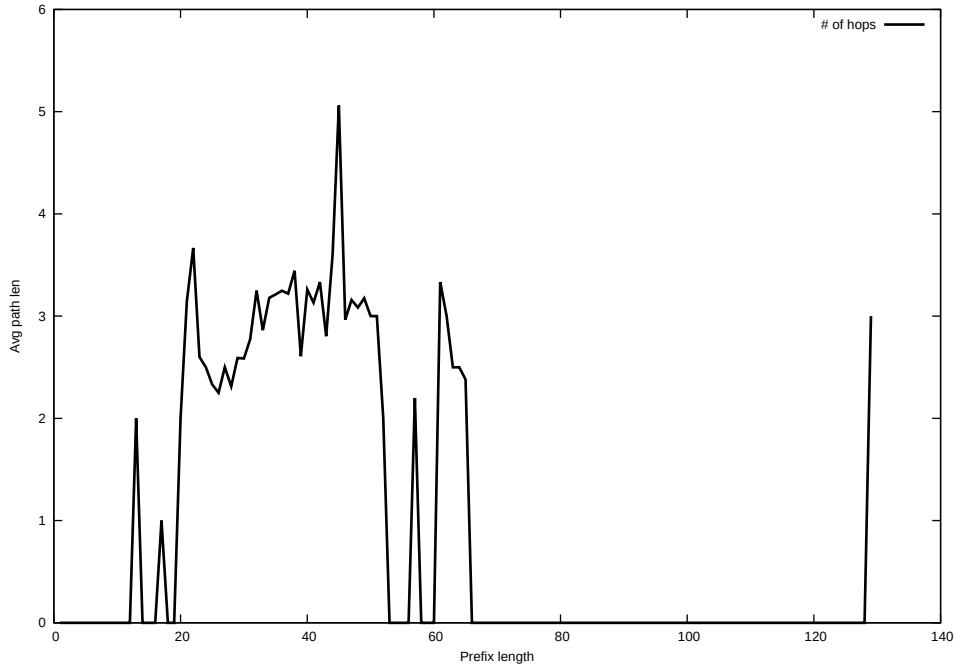
2013-02-23



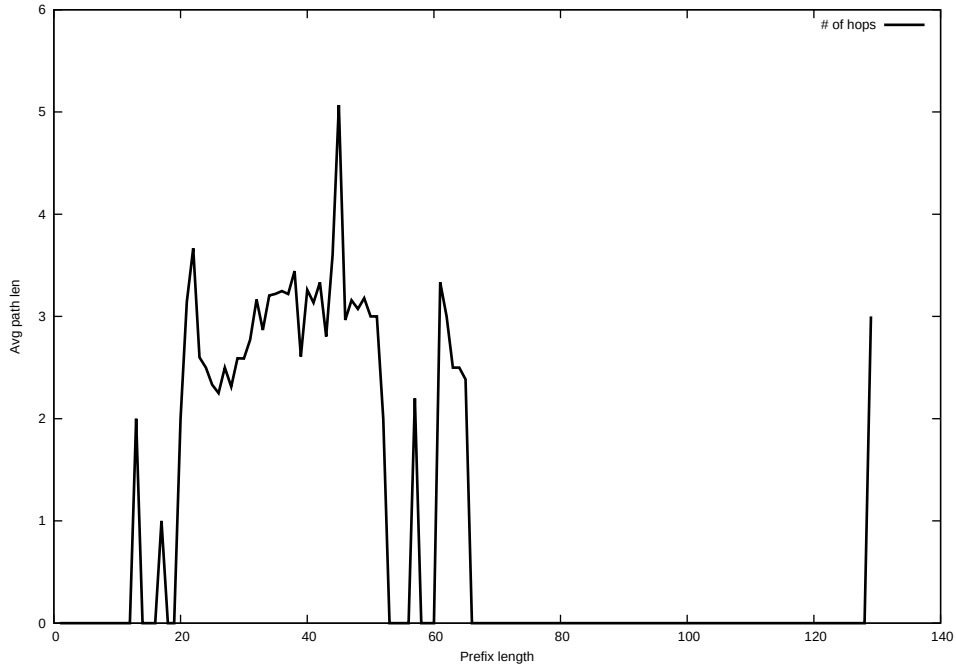
2013-02-24



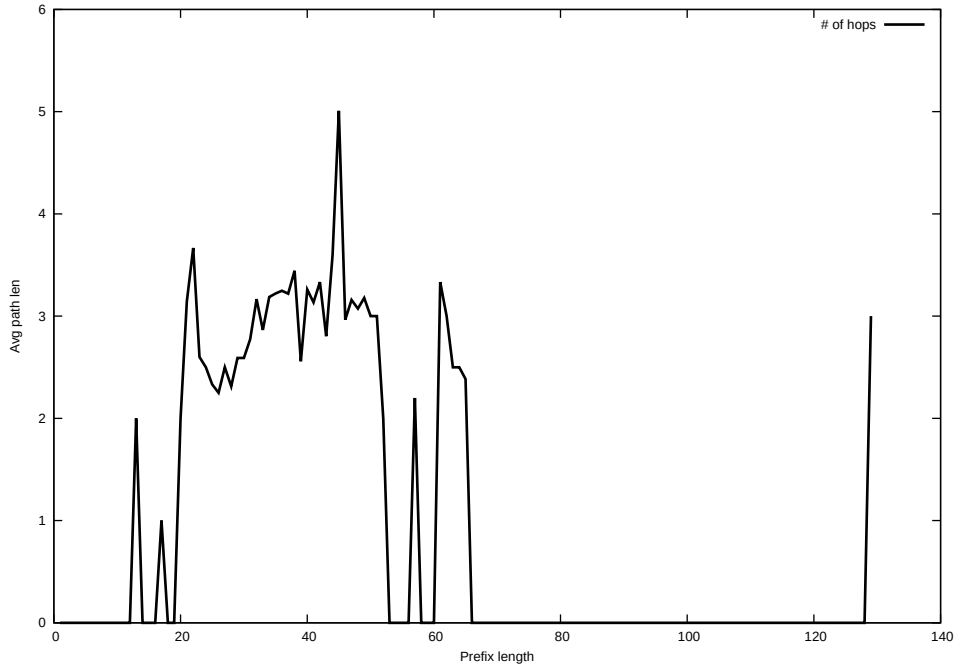
2013-02-25



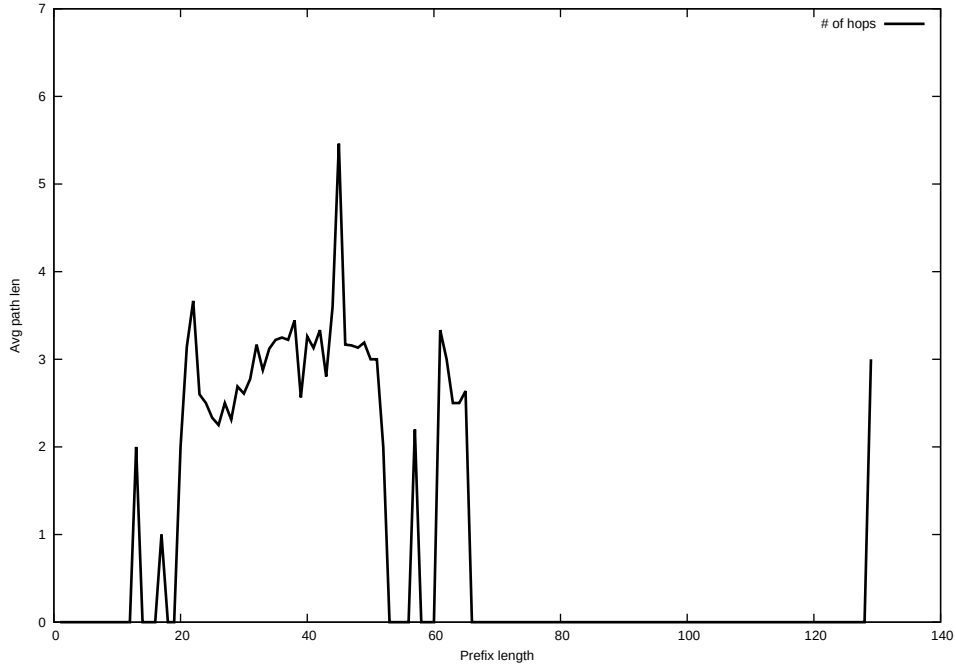
2013-02-26



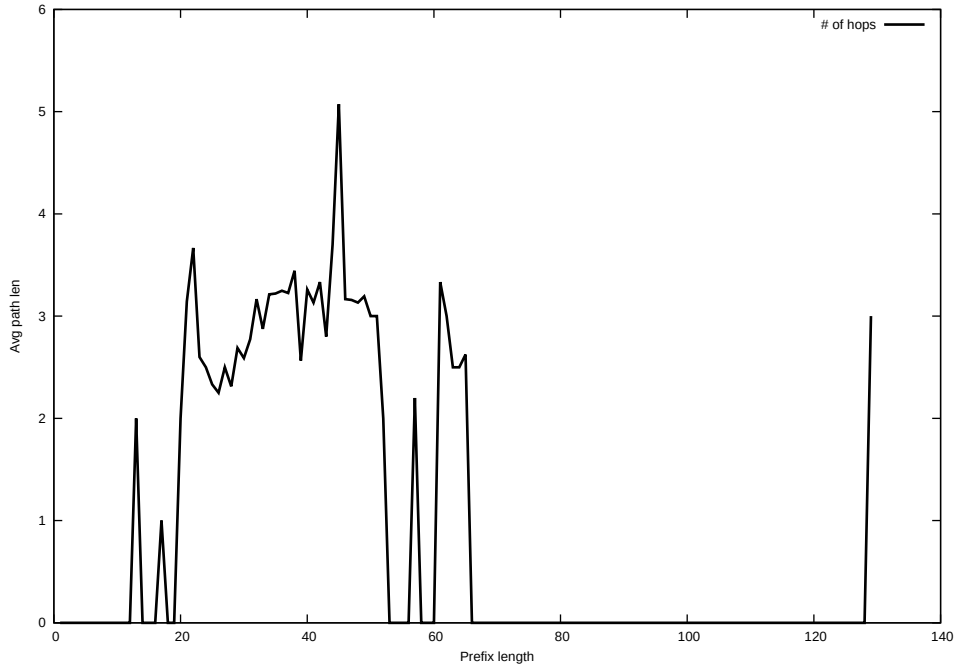
2013-02-27



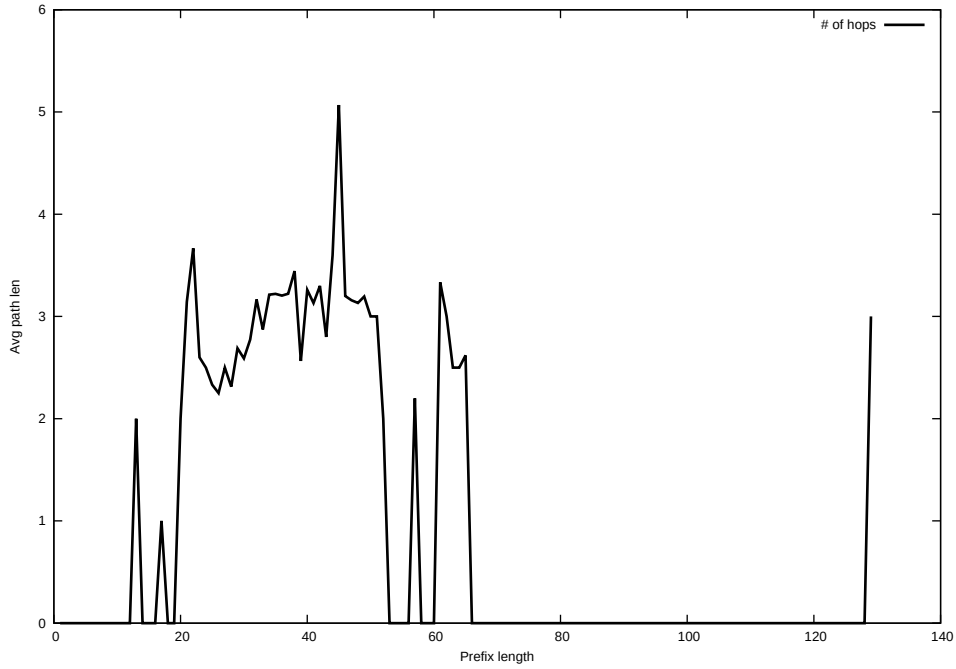
2013-02-28



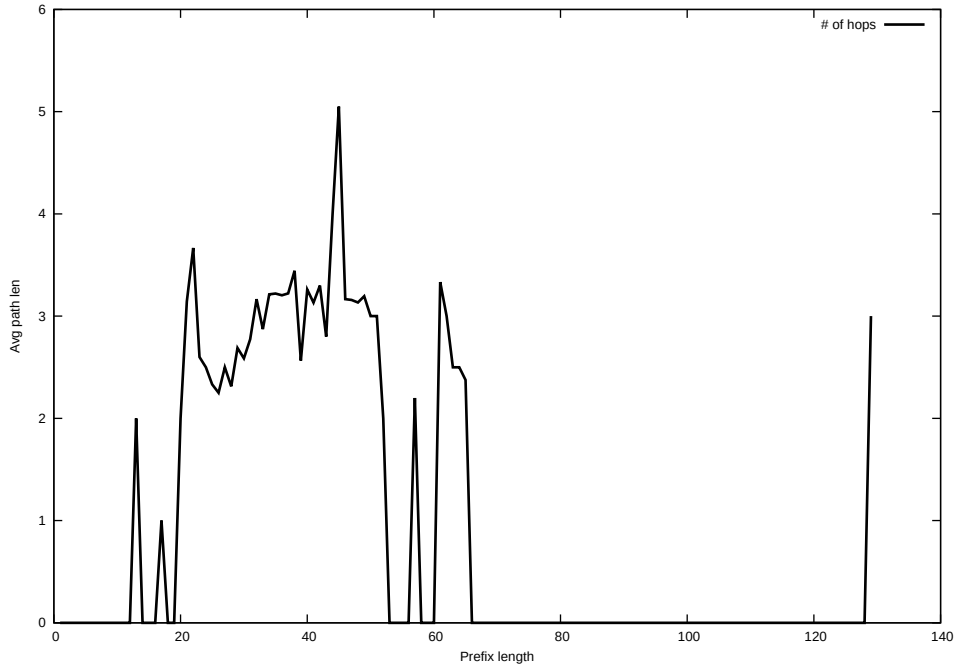
2013-03-01



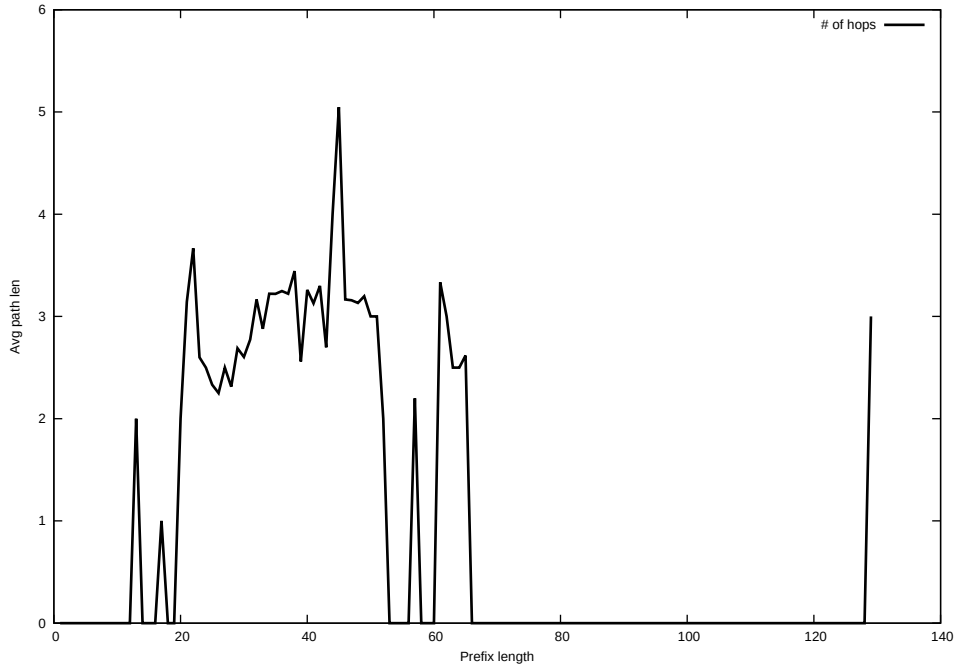
2013-03-02



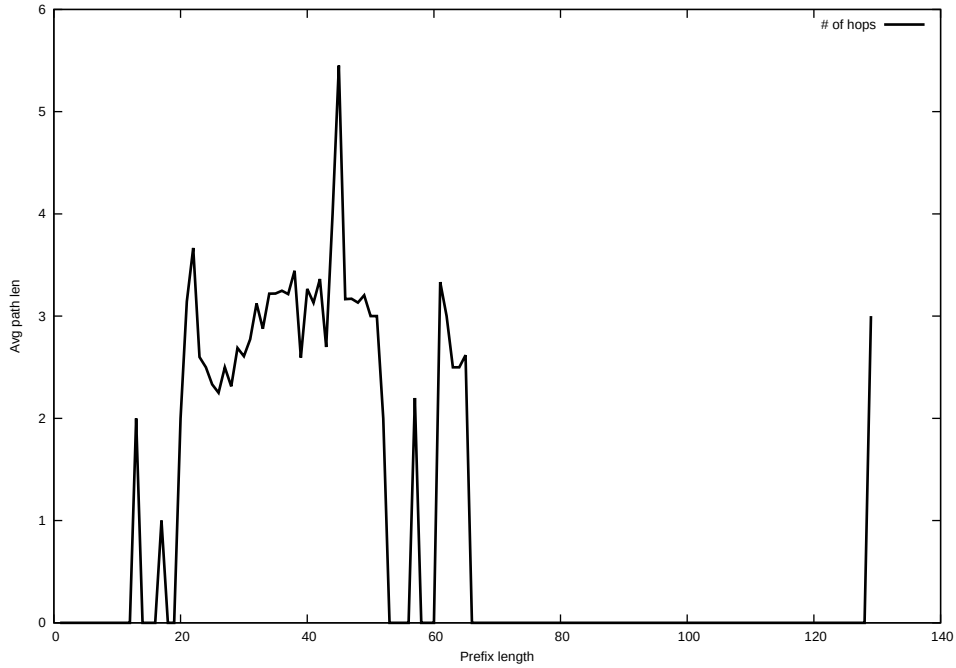
2013-03-03



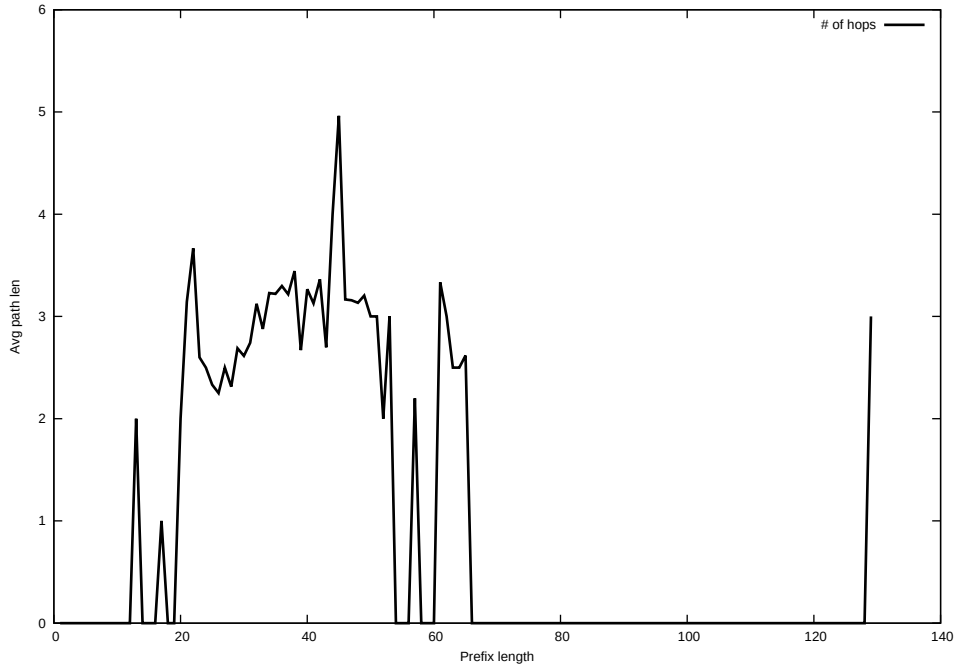
2013-03-04



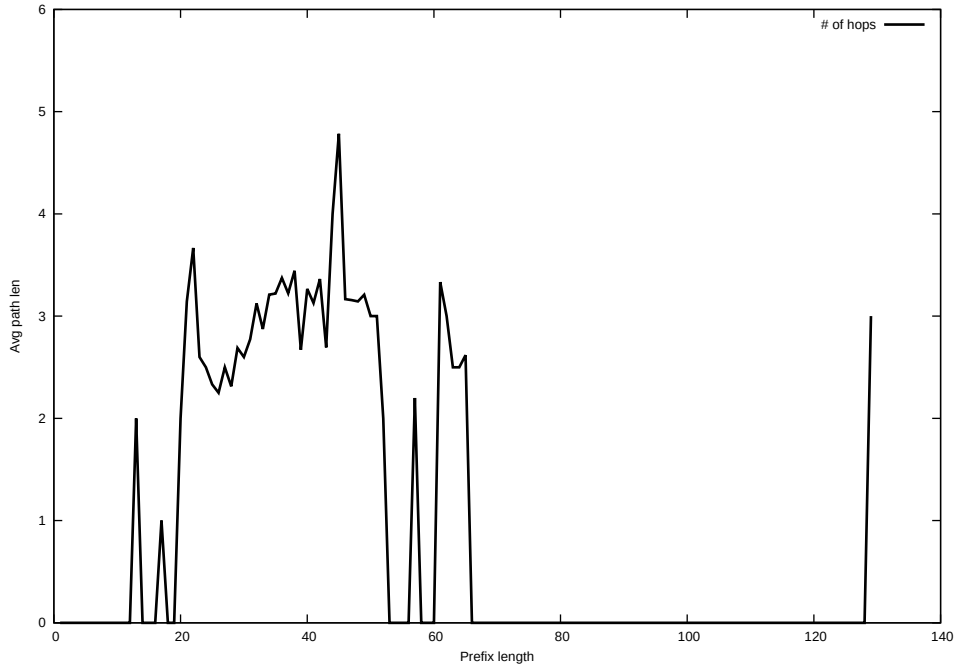
2013-03-05



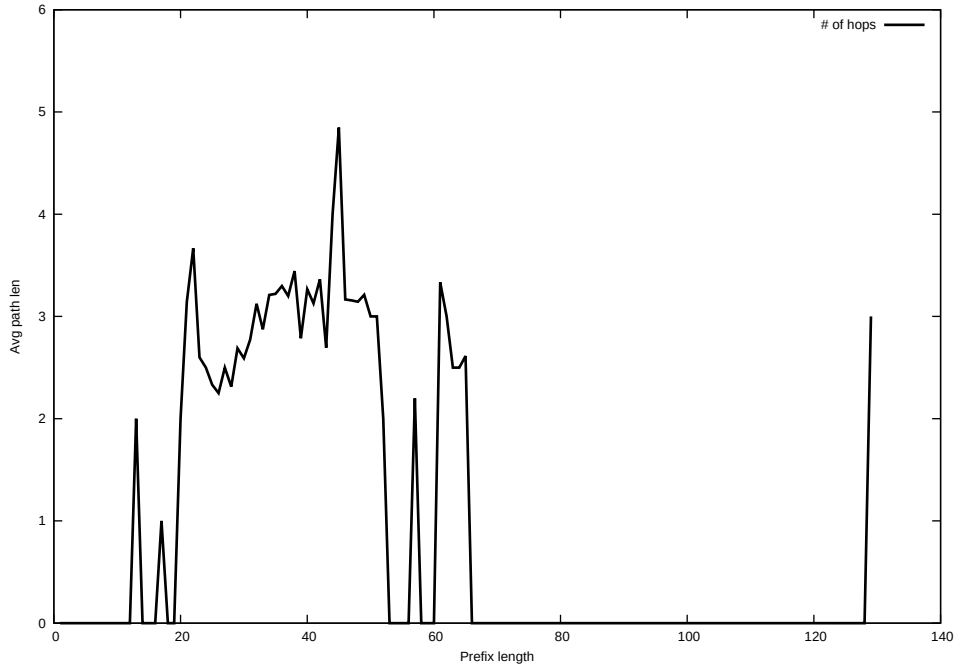
2013-03-06



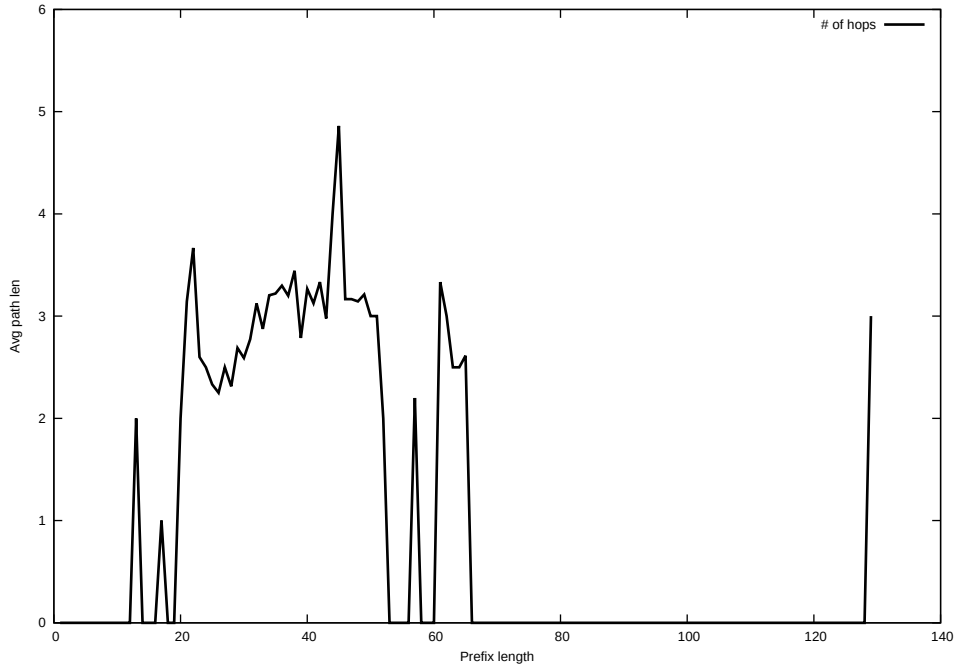
2013-03-07



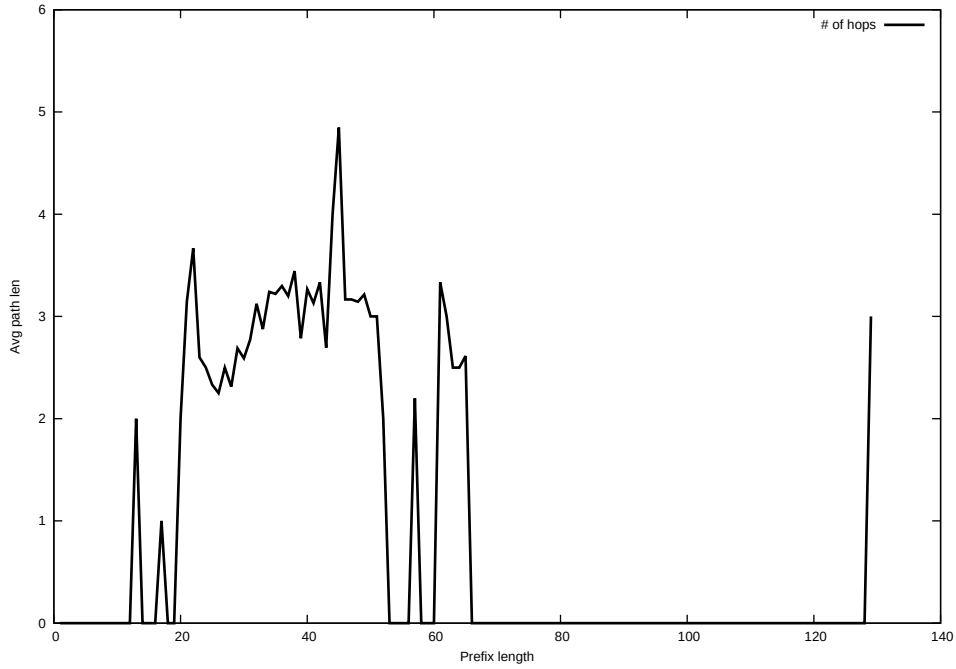
2013-03-08



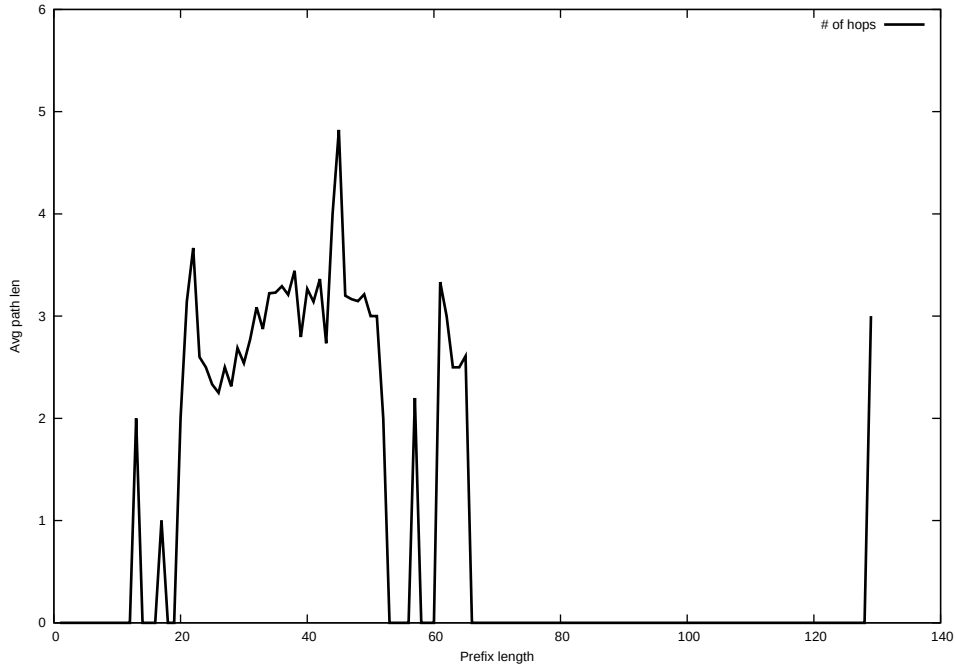
2013-03-09



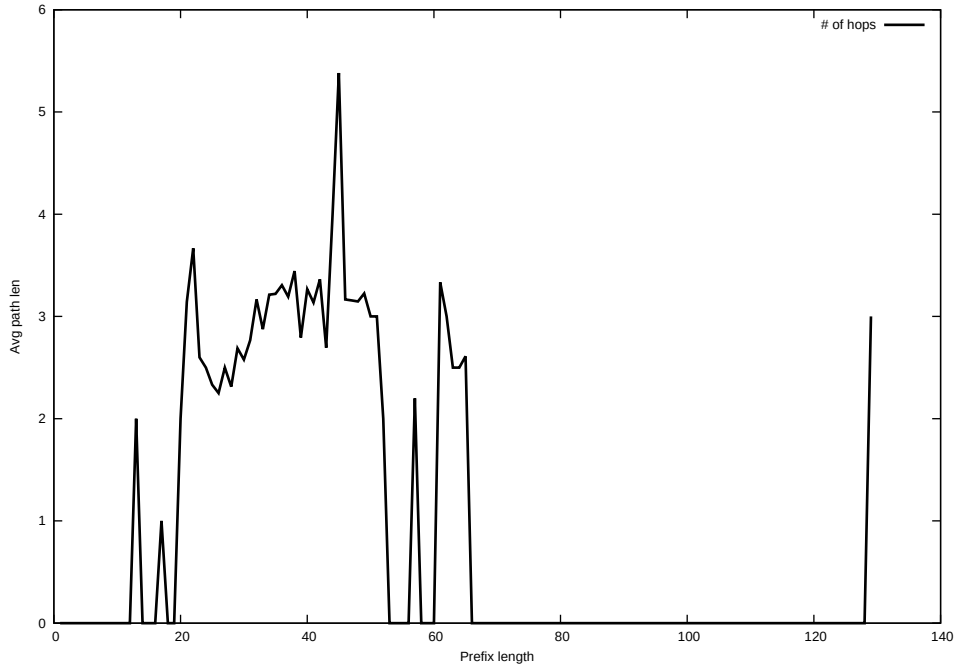
2013-03-10



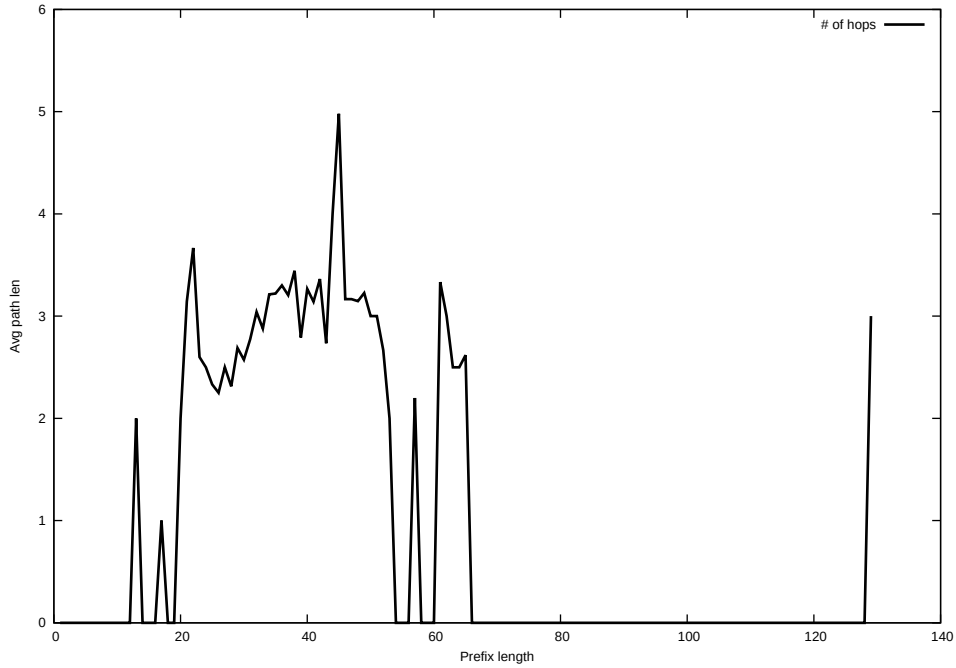
2013-03-11



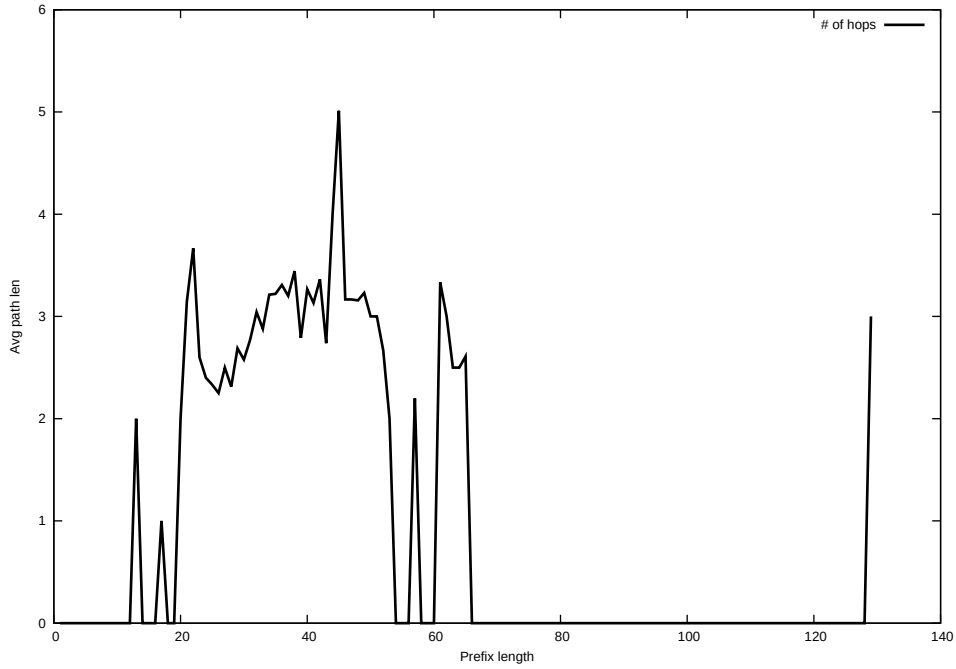
2013-03-12



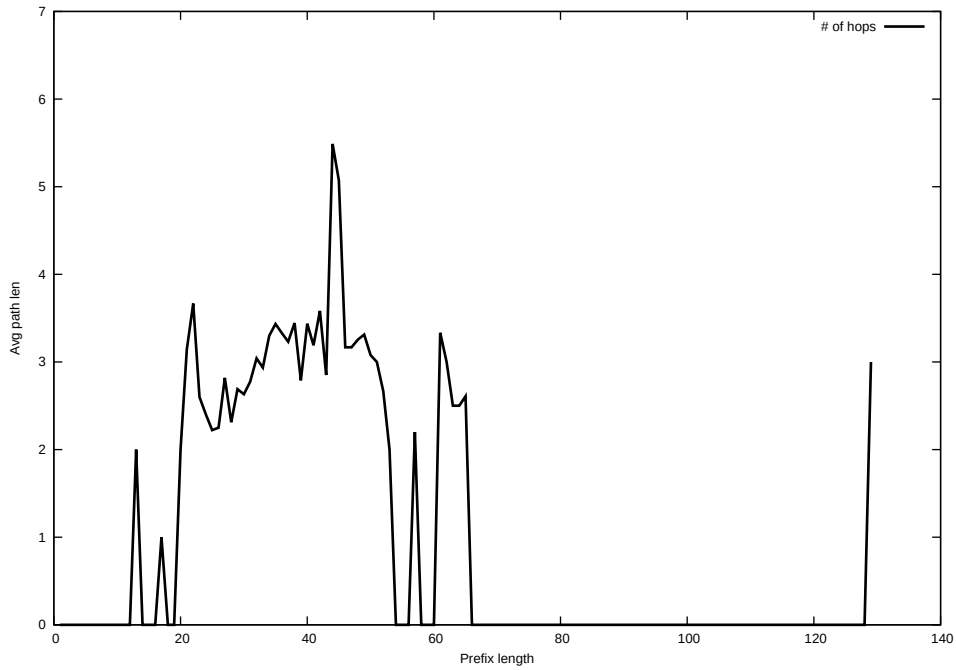
2013-03-13



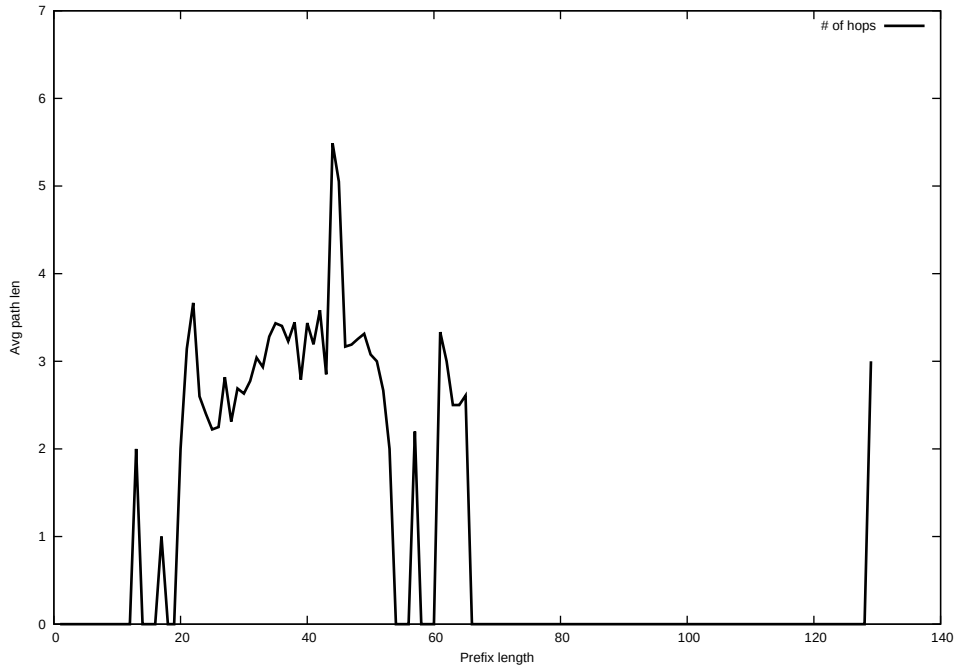
2013-03-14



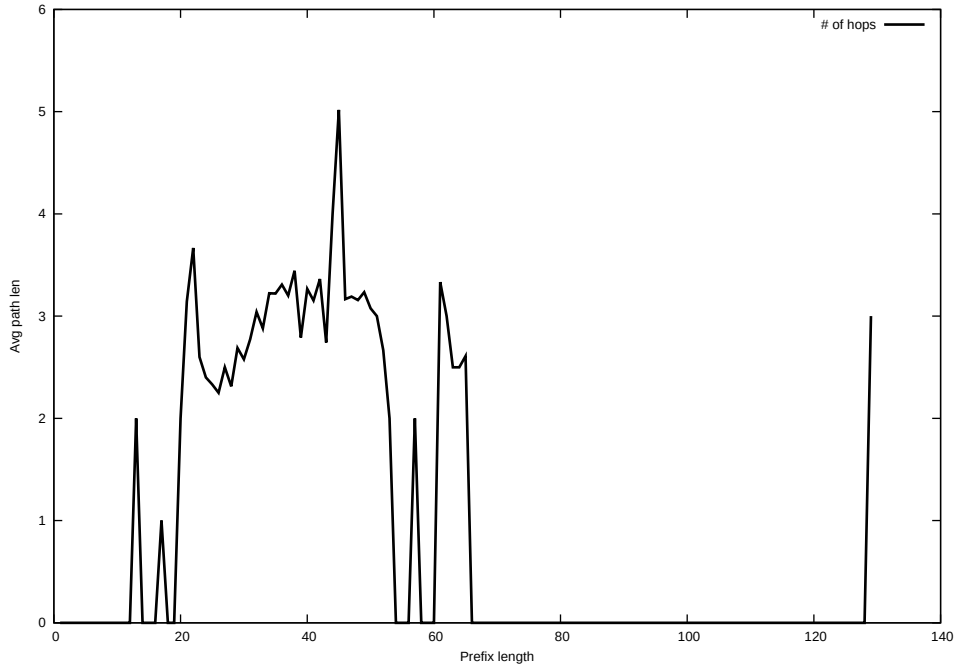
2013-03-15



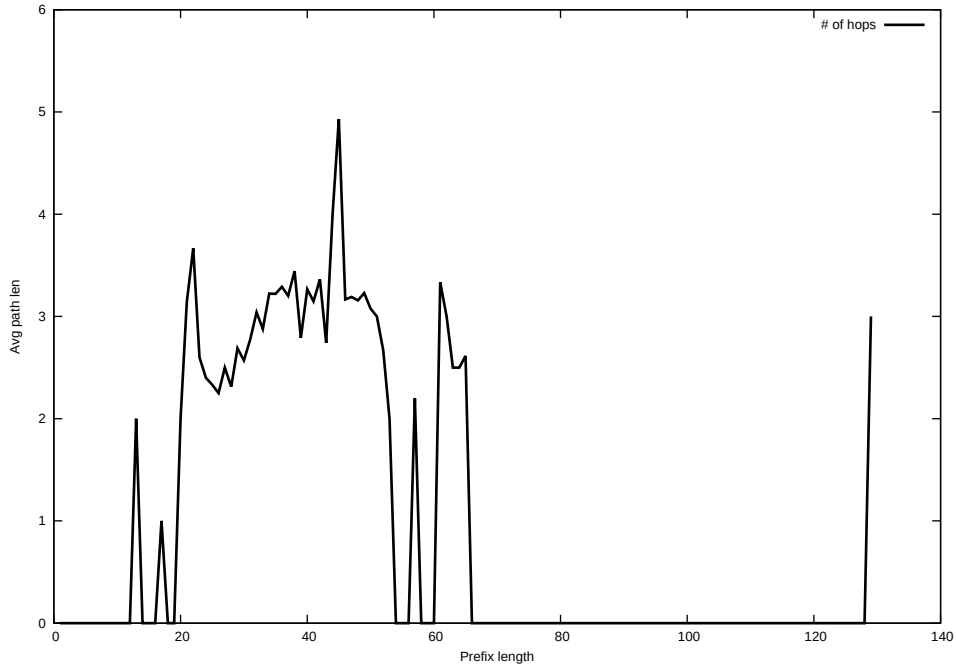
2013-03-16



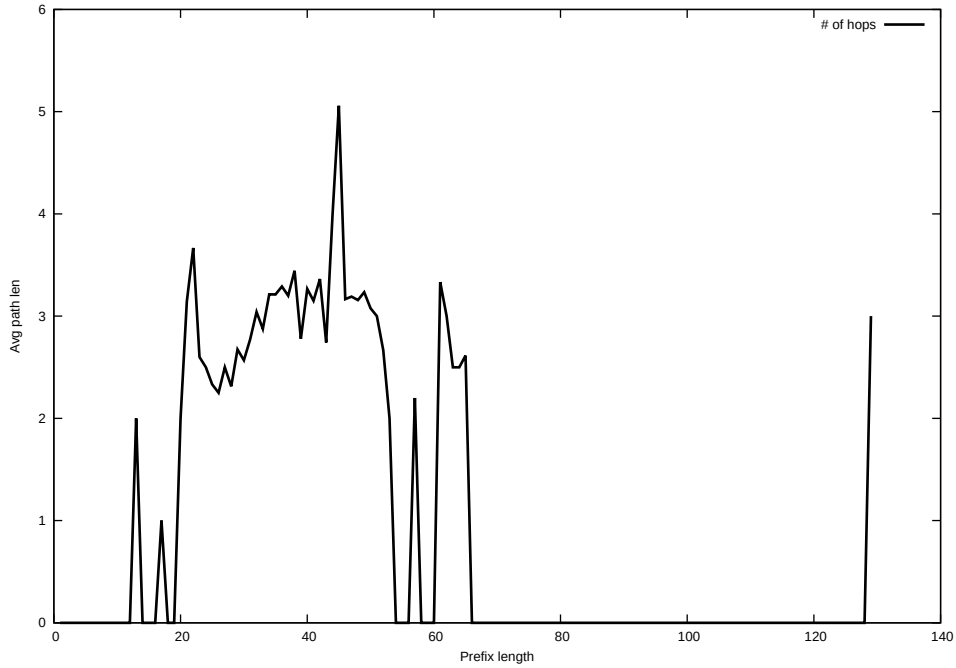
2013-03-17



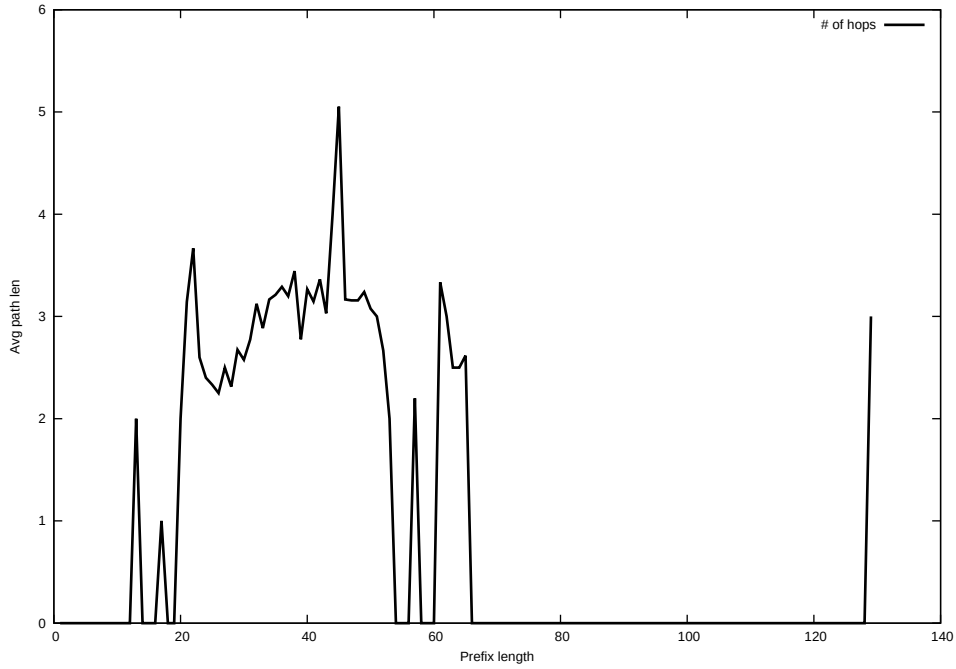
2013-03-18



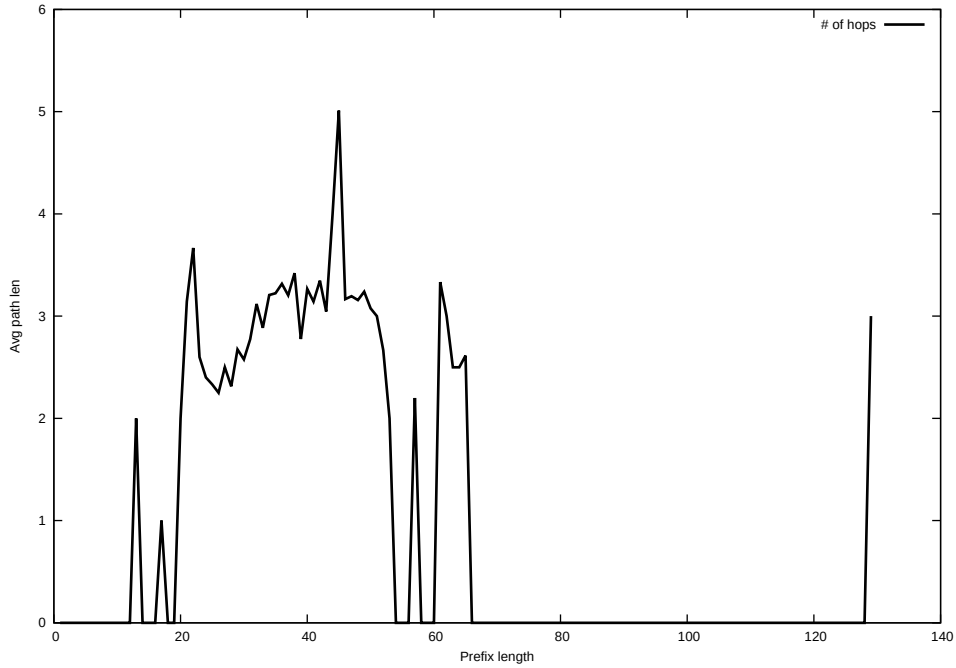
2013-03-19



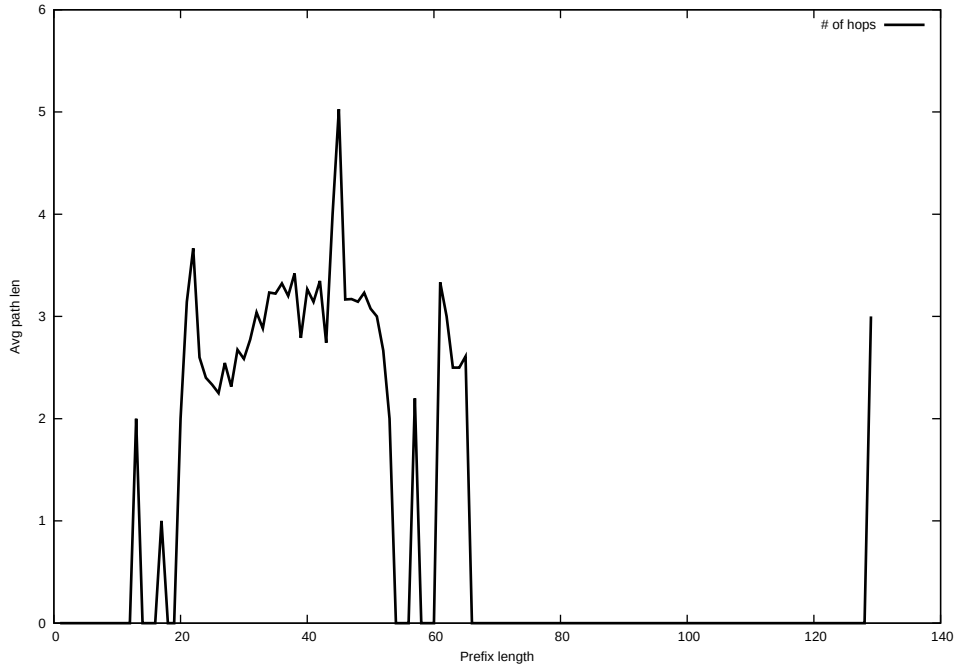
2013-03-20



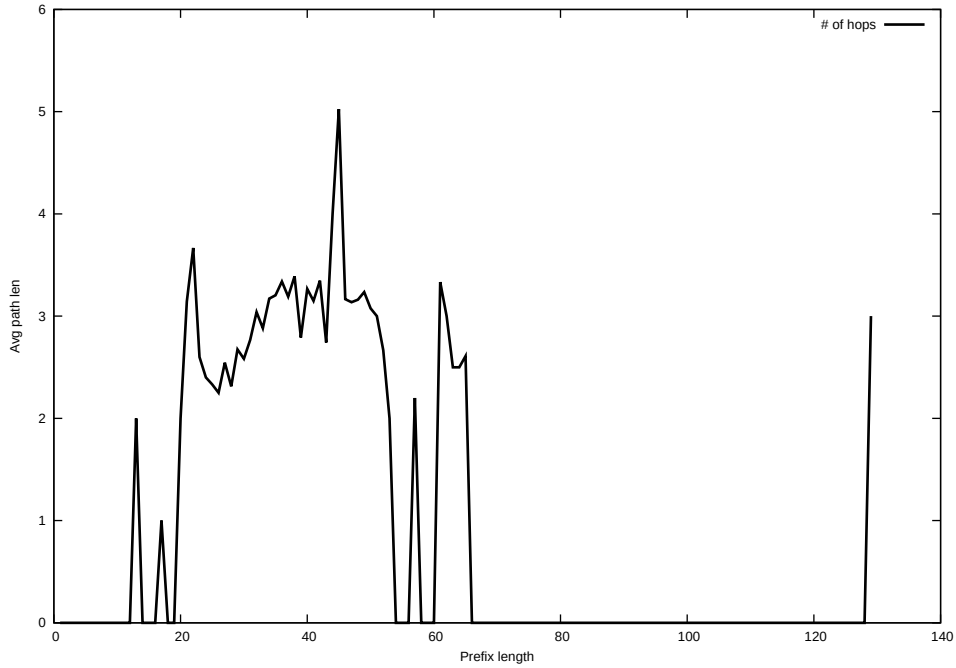
2013-03-21



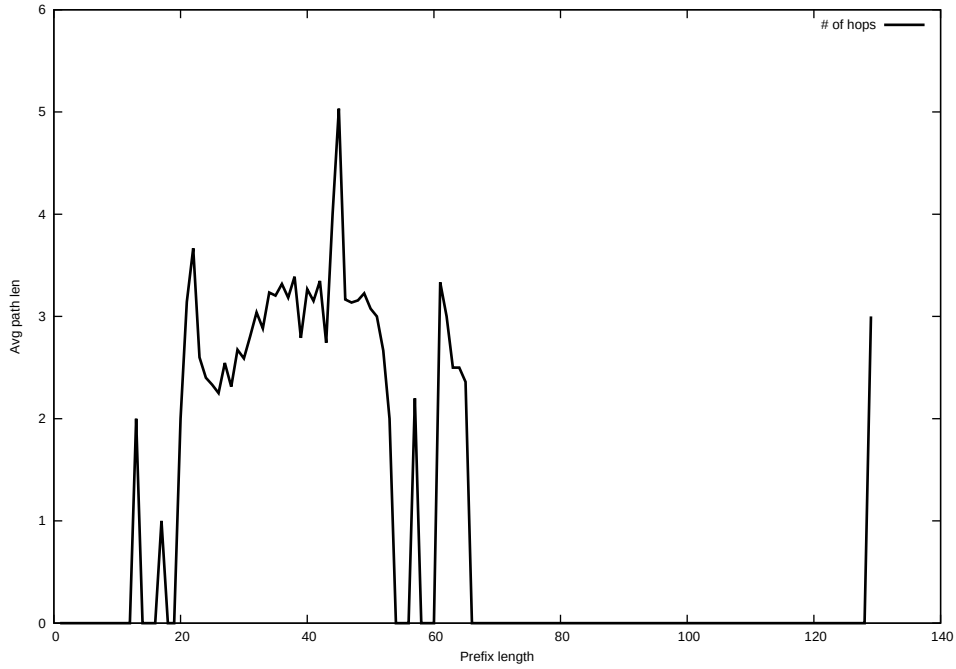
2013-03-22



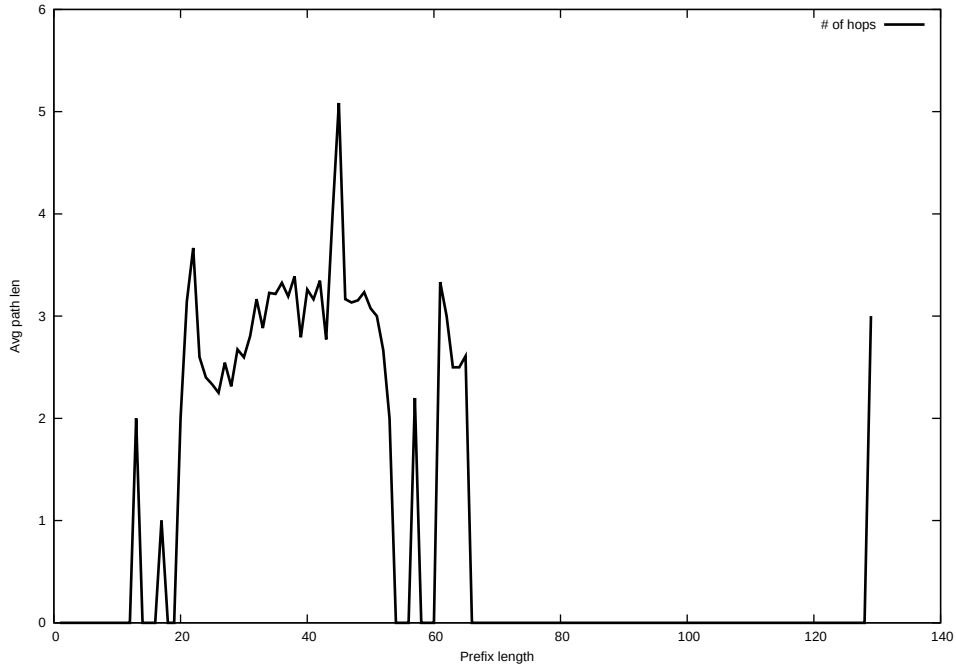
2013-03-23



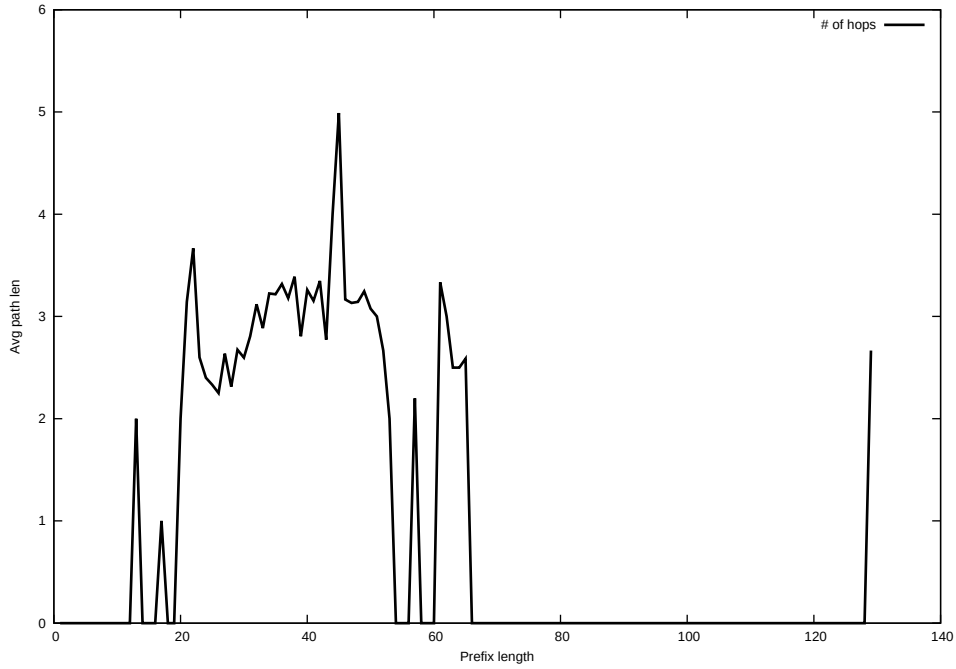
2013-03-24



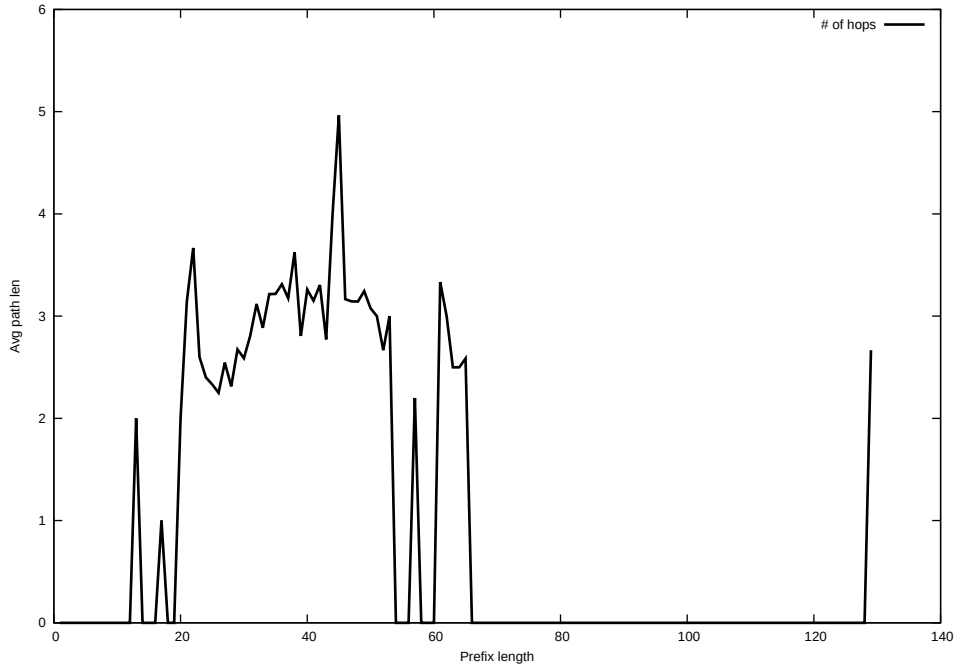
2013-03-25



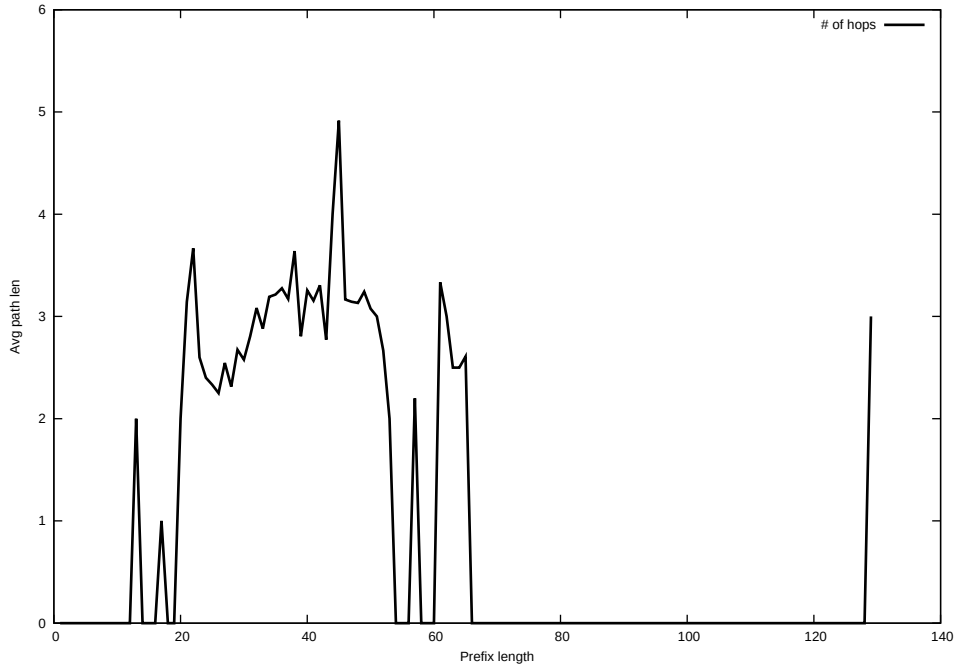
2013-03-26



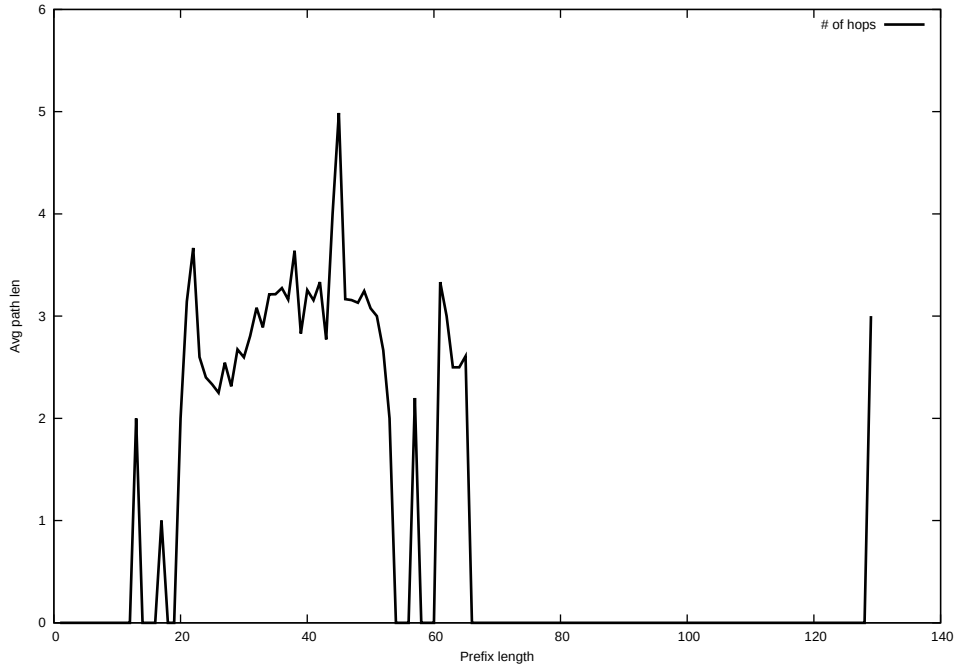
2013-03-27



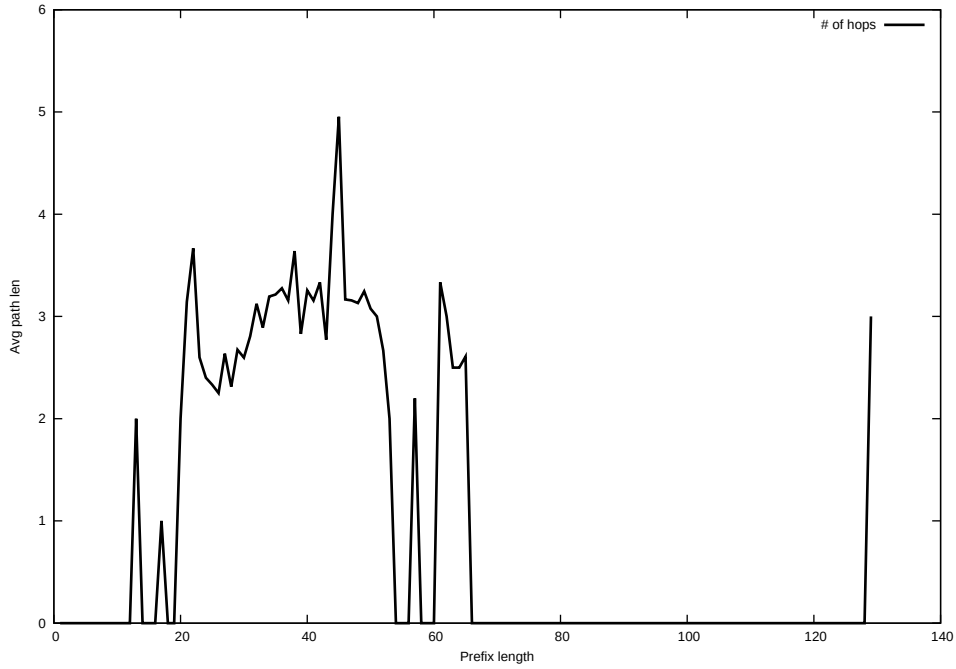
2013-03-28



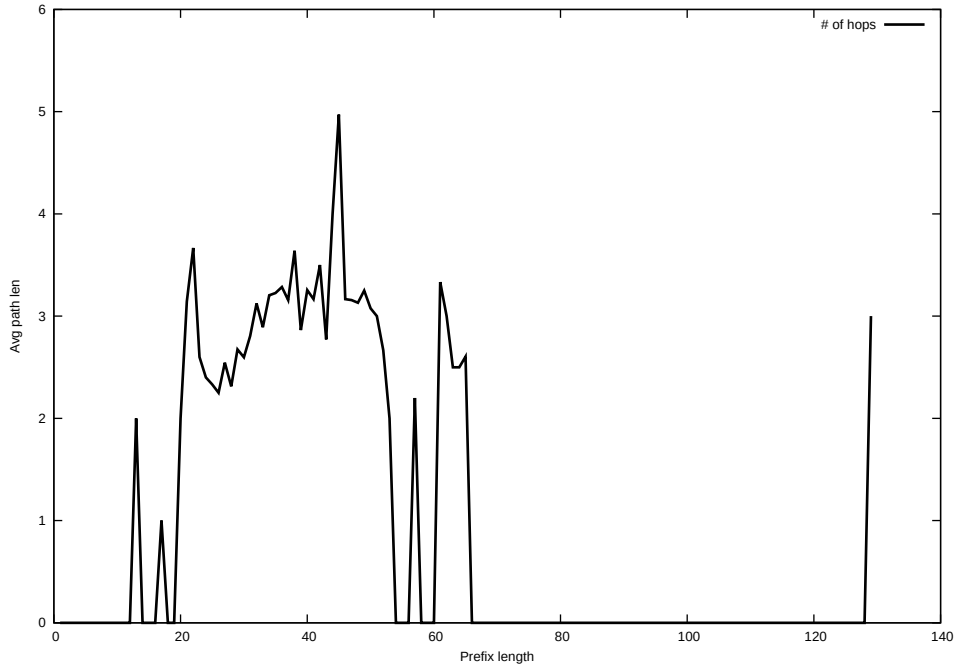
2013-03-29



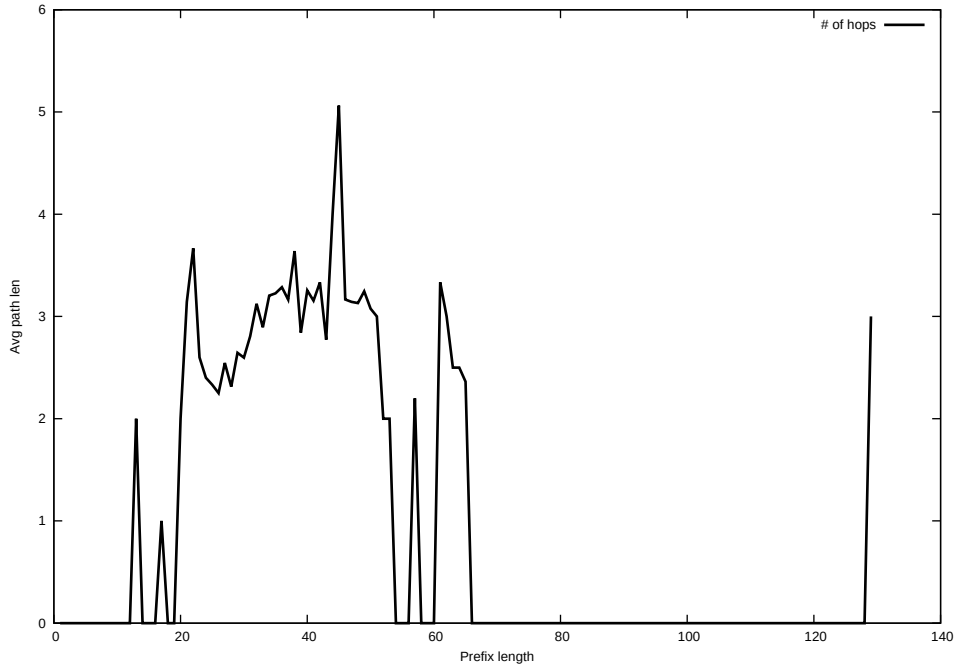
2013-03-30



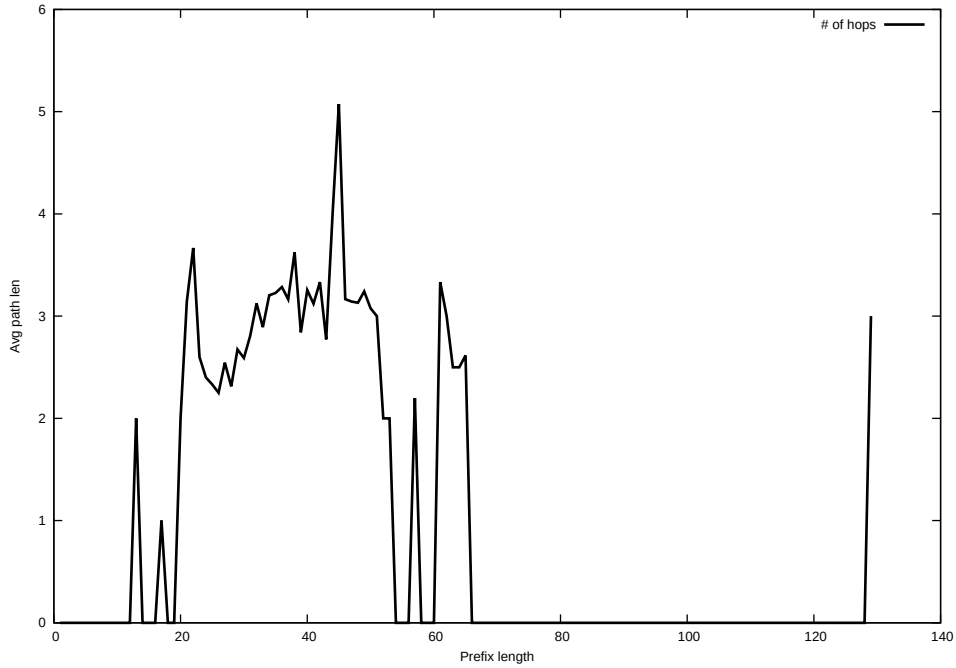
2013-03-31



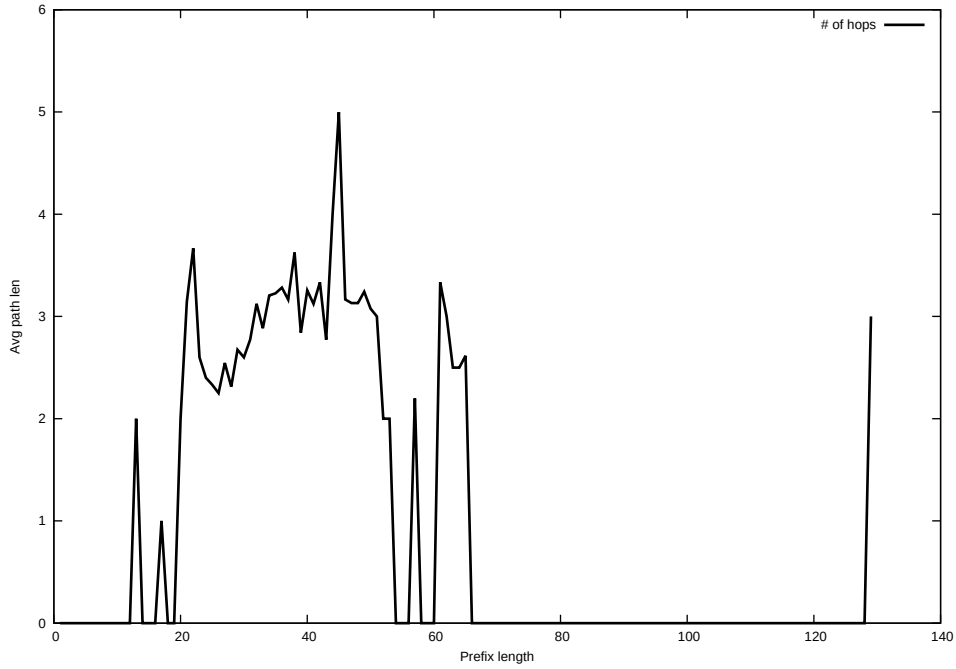
2013-04-01



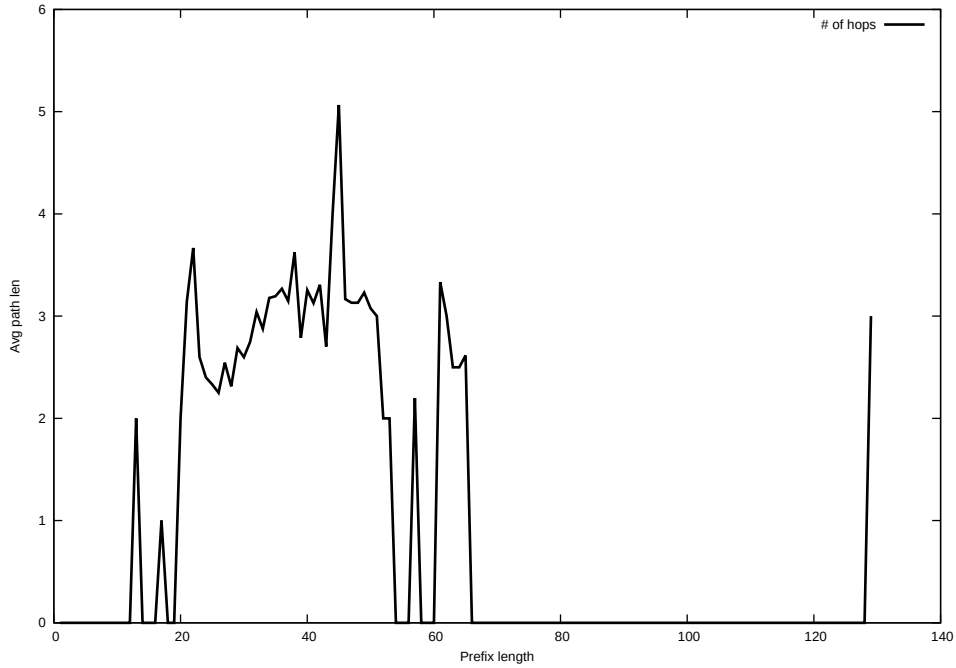
2013-04-02



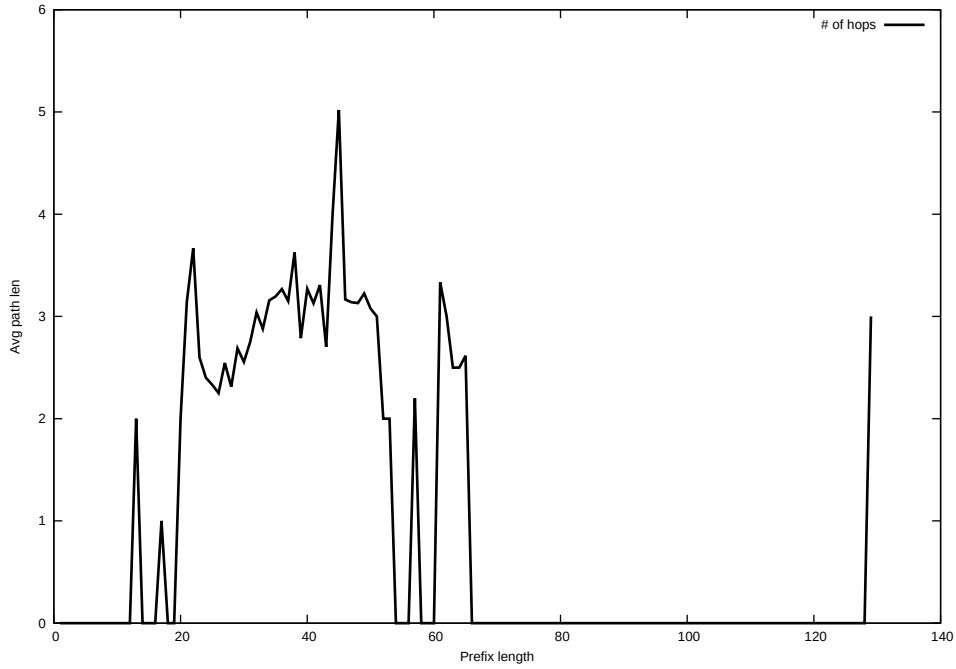
2013-04-03



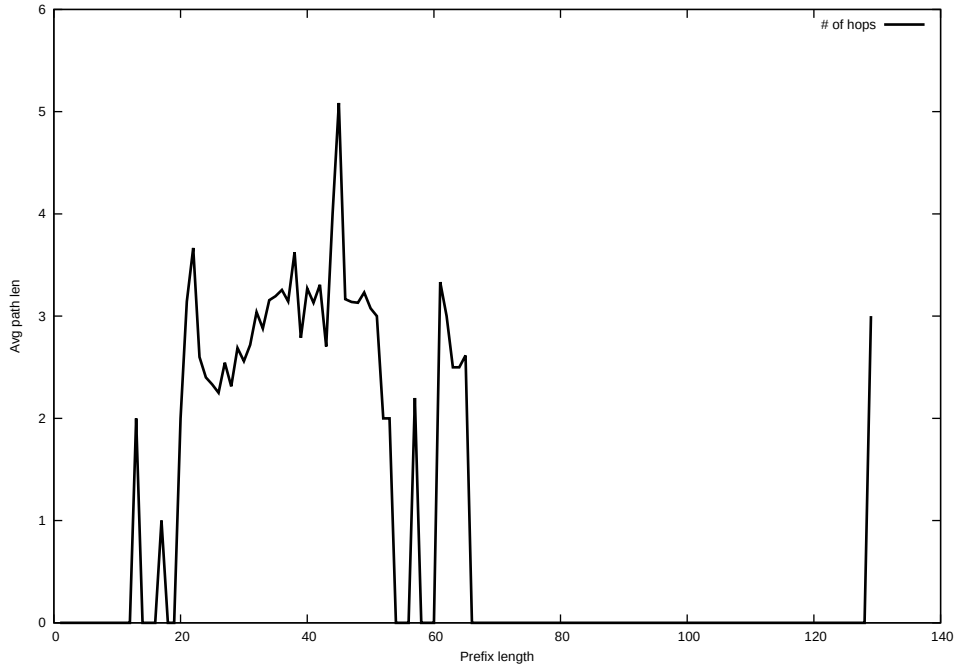
2013-04-04



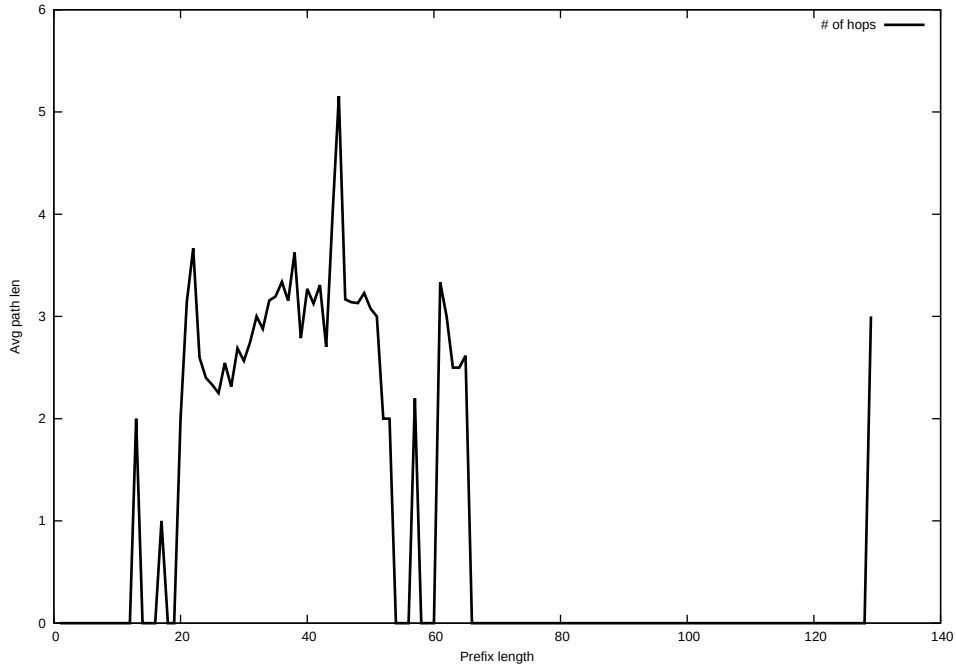
2013-04-05



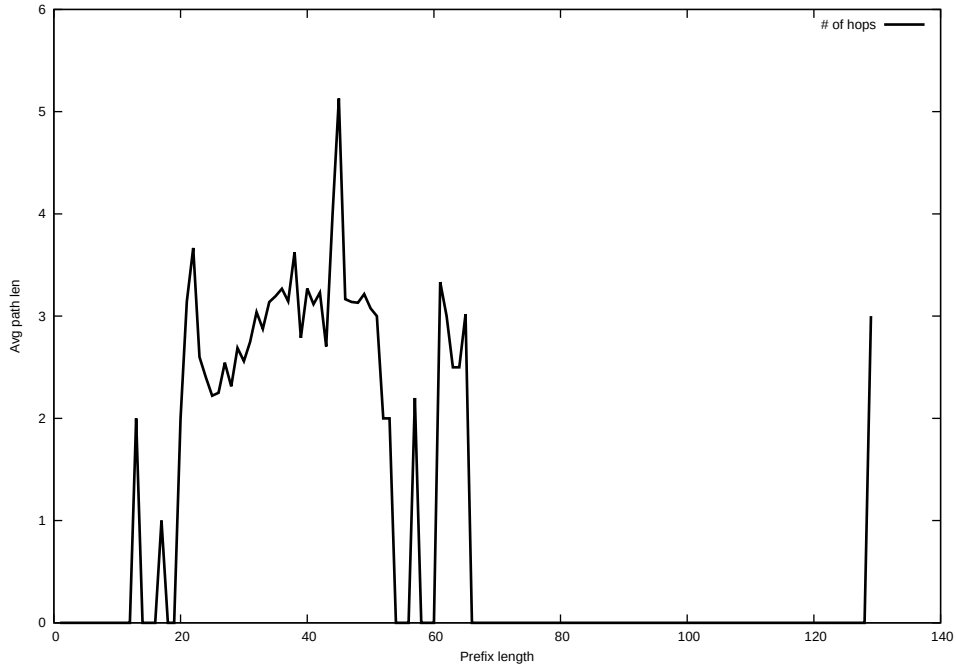
2013-04-06



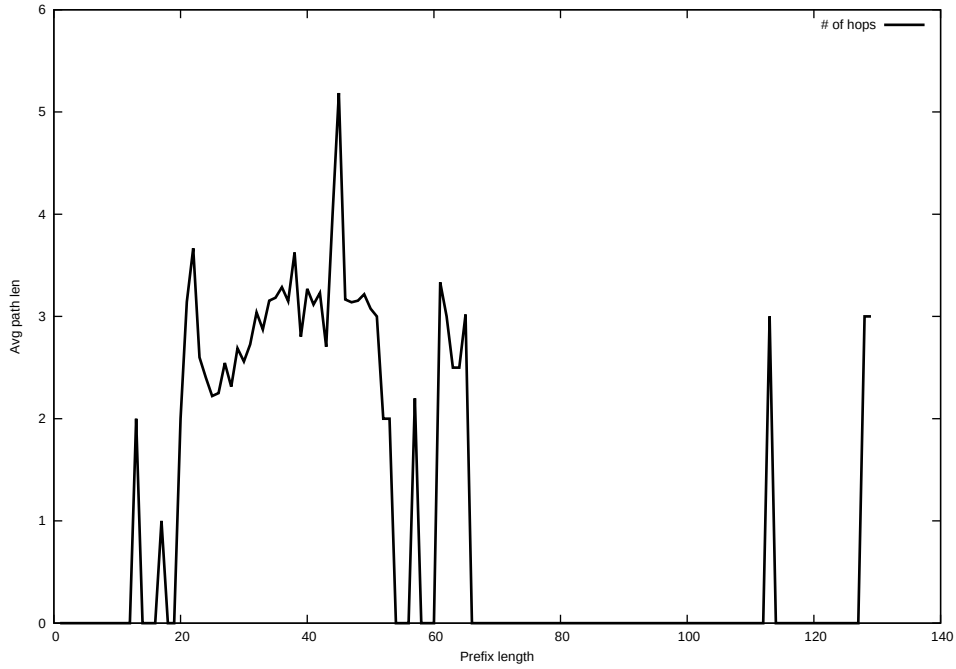
2013-04-07



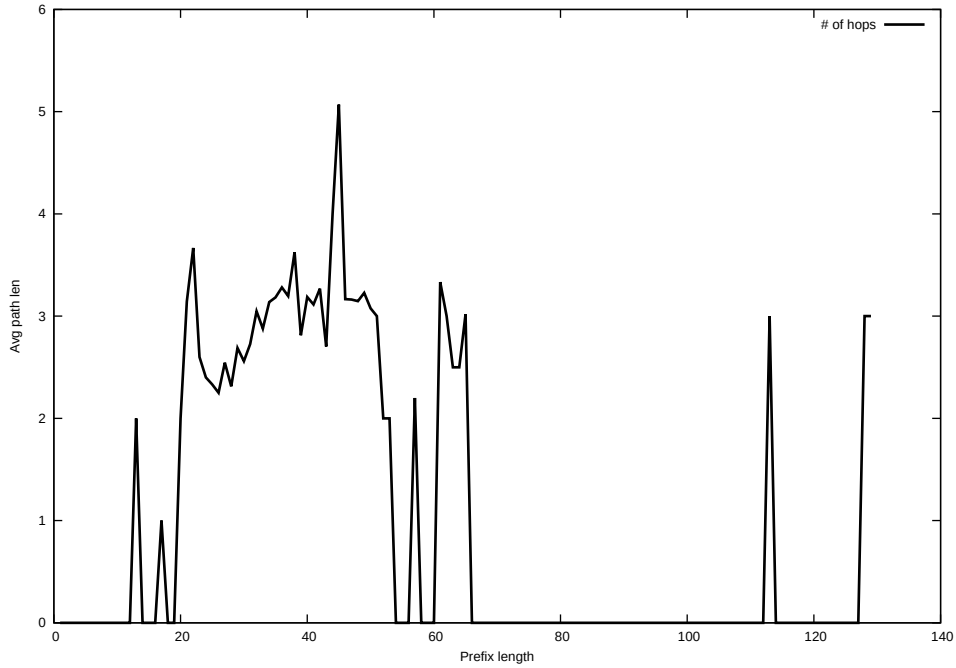
2013-04-08



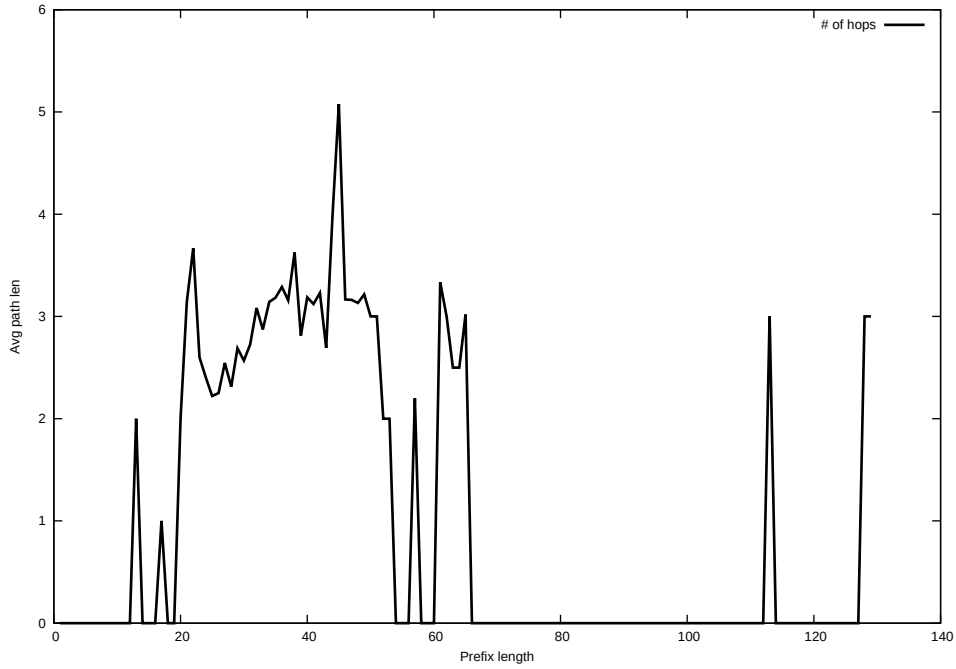
2013-04-09



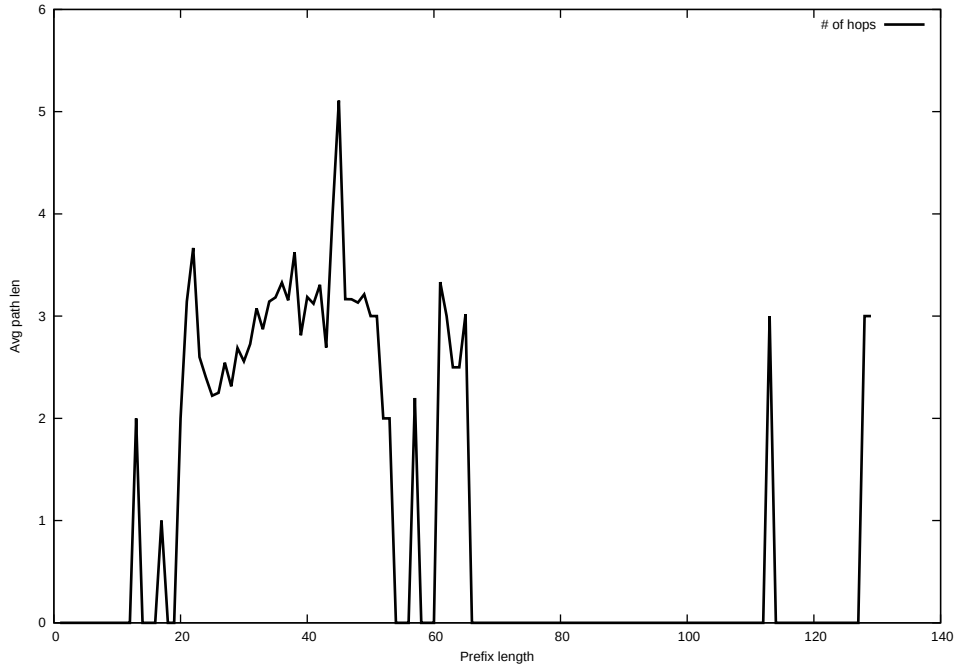
2013-04-10



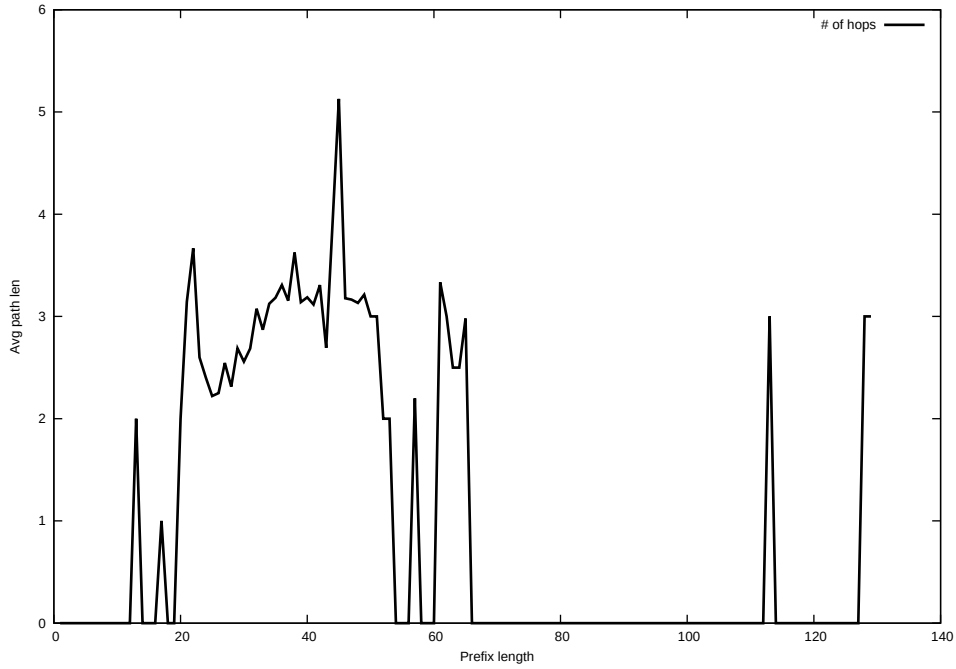
2013-04-11



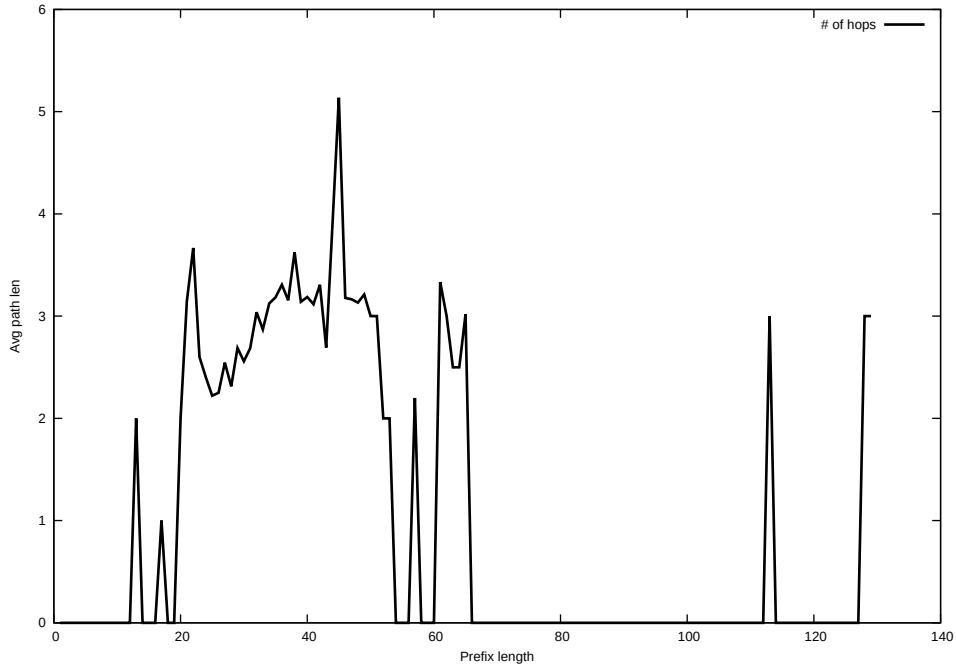
2013-04-12



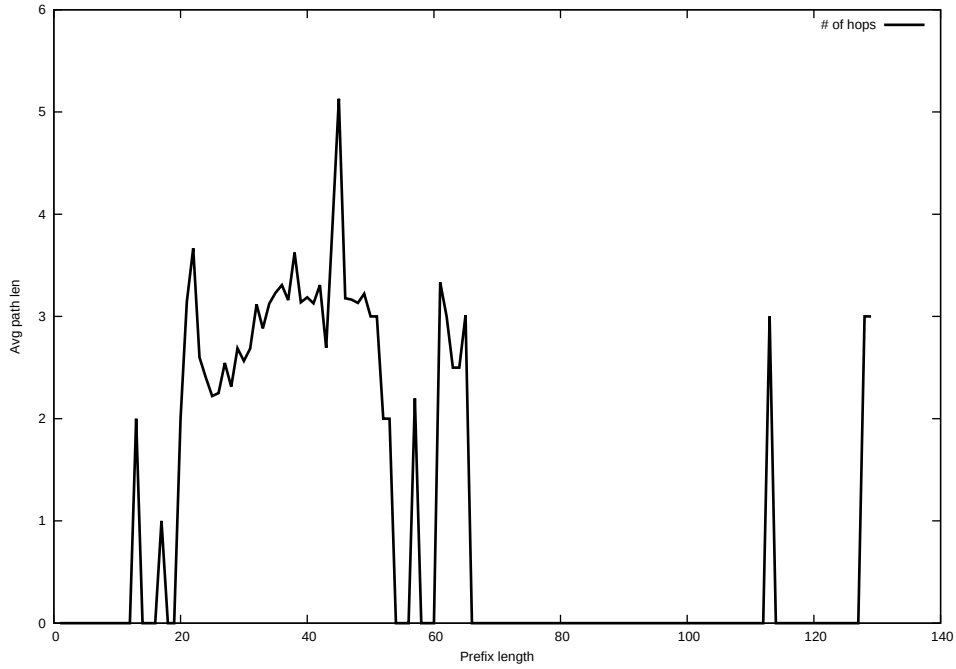
2013-04-13



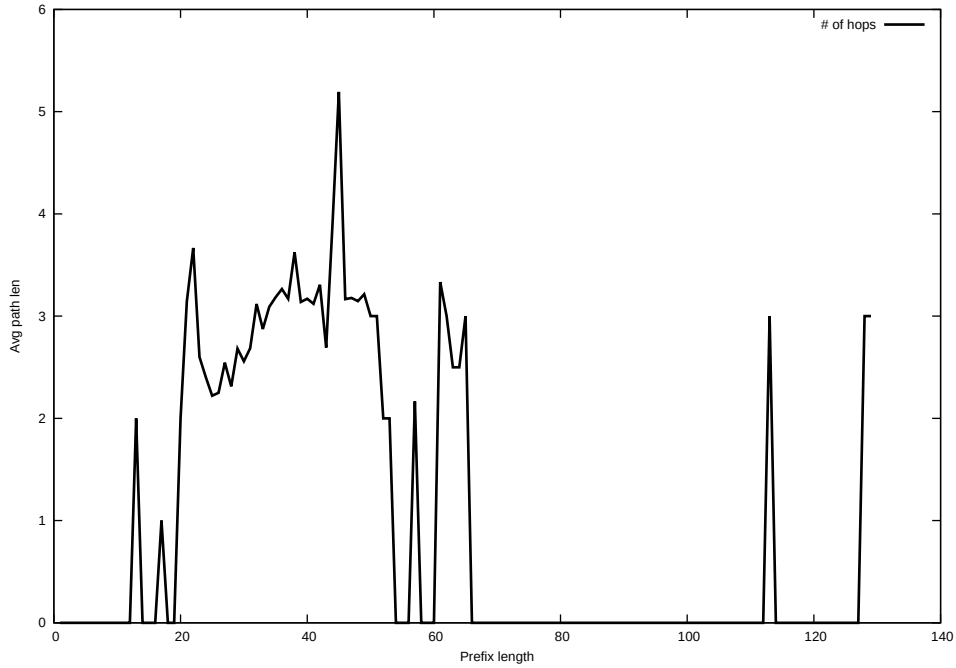
2013-04-14



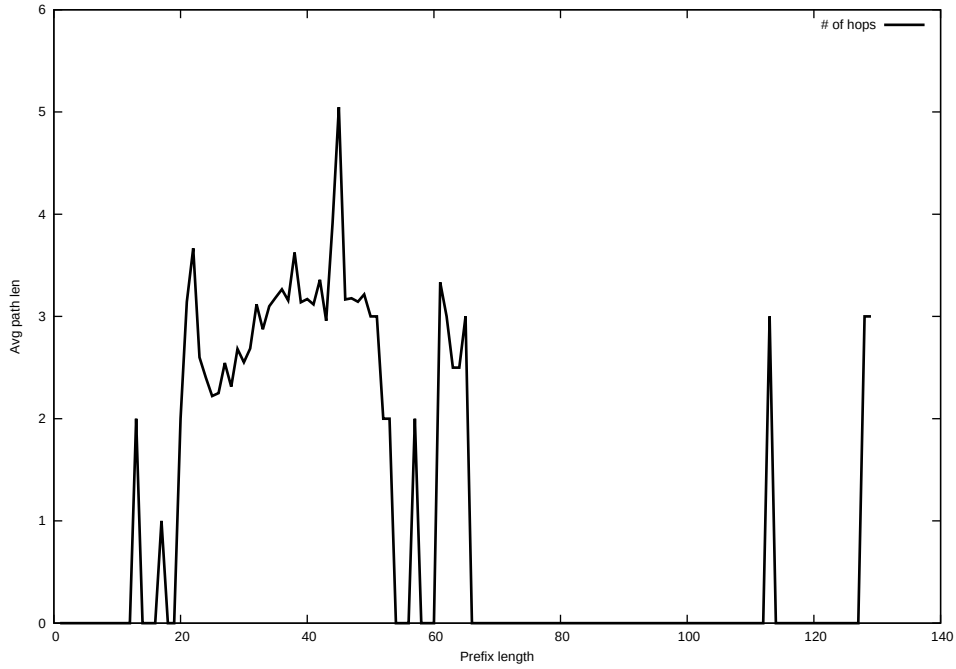
2013-04-15



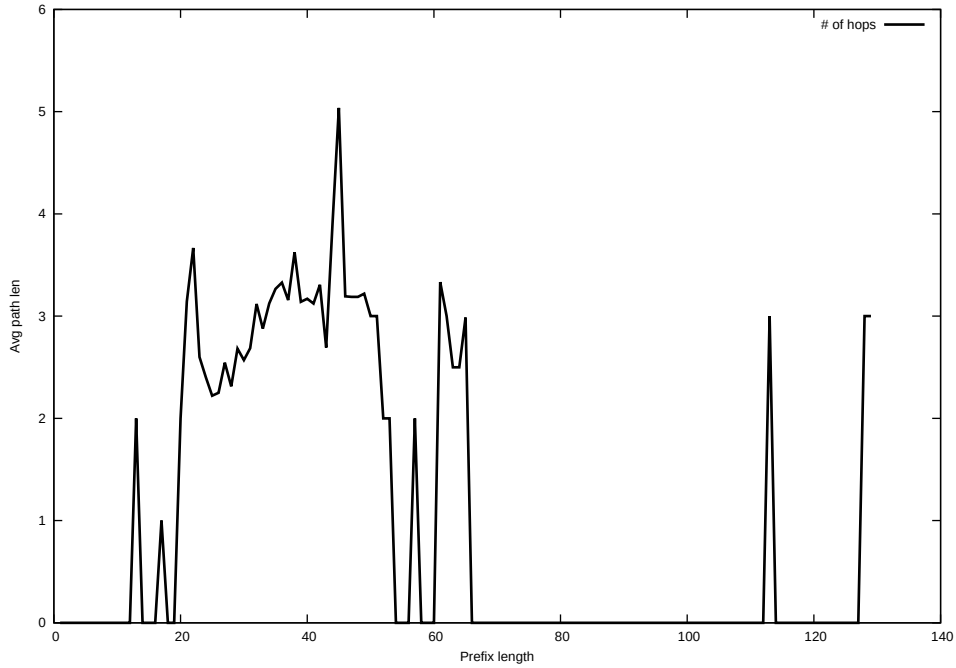
2013-04-16



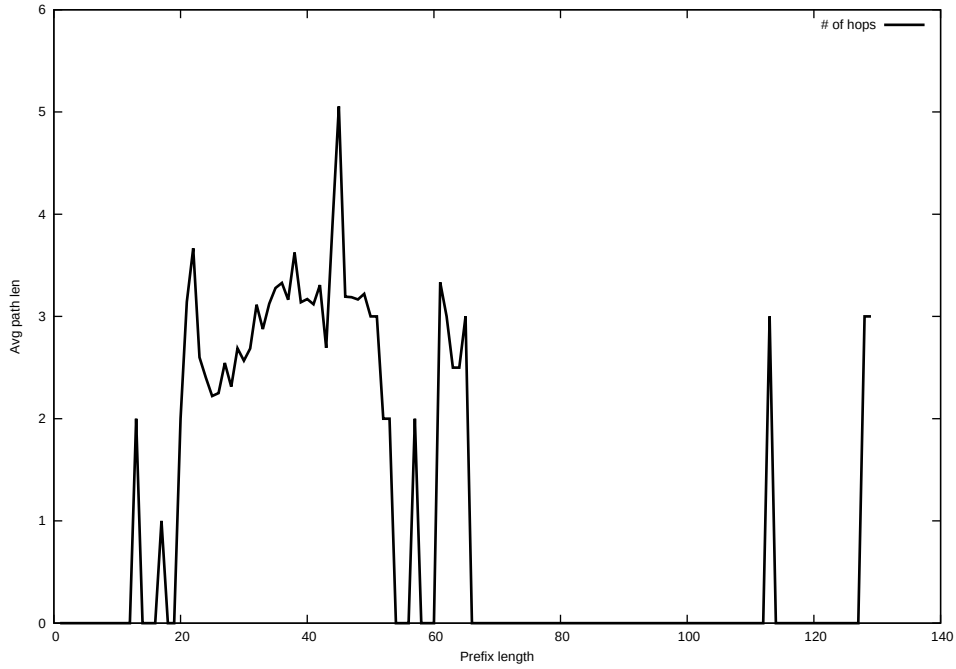
2013-04-17



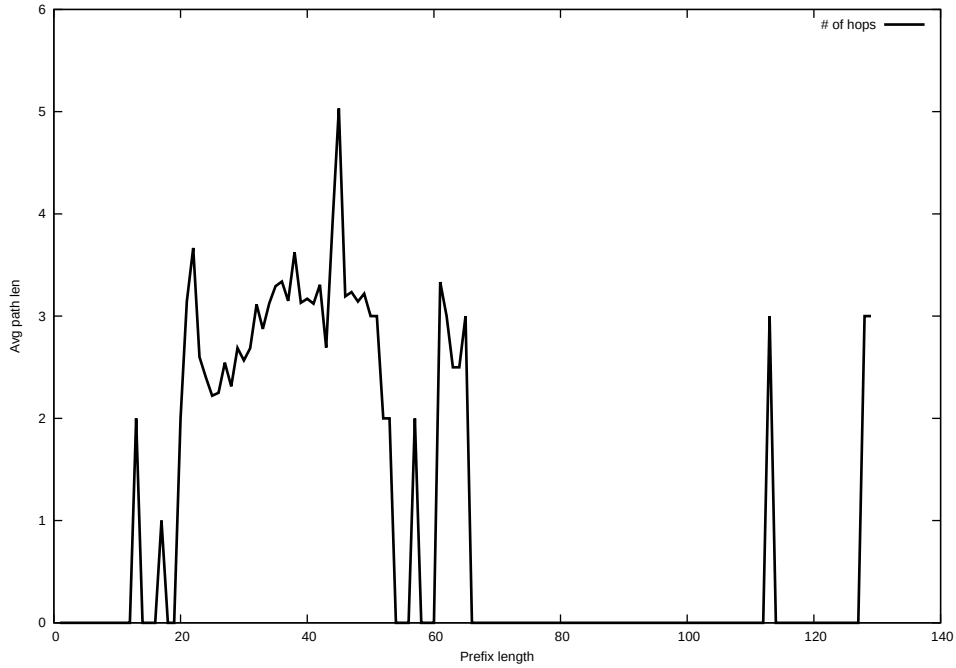
2013-04-18



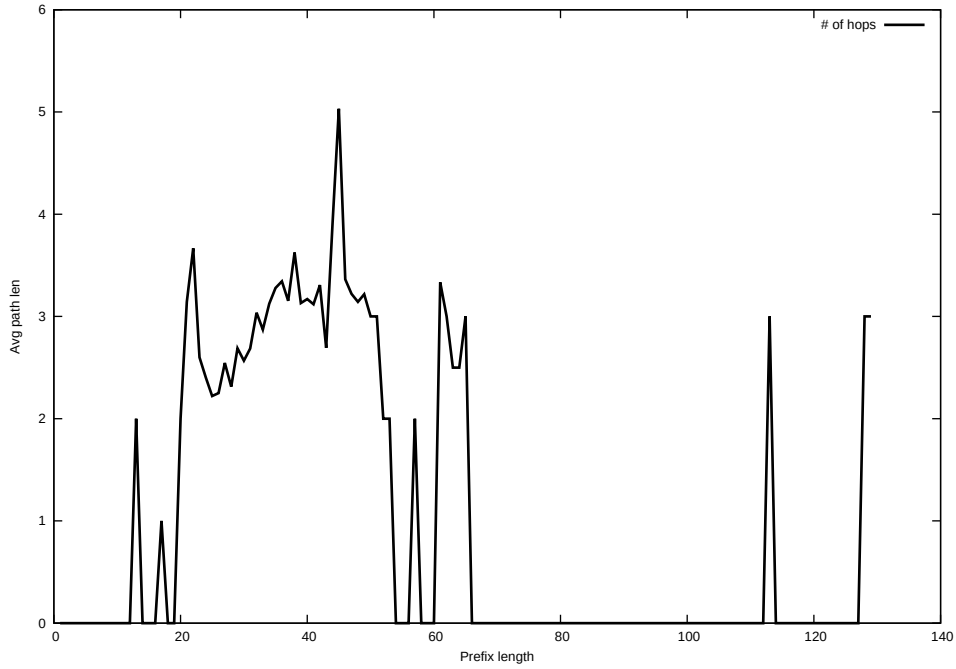
2013-04-19



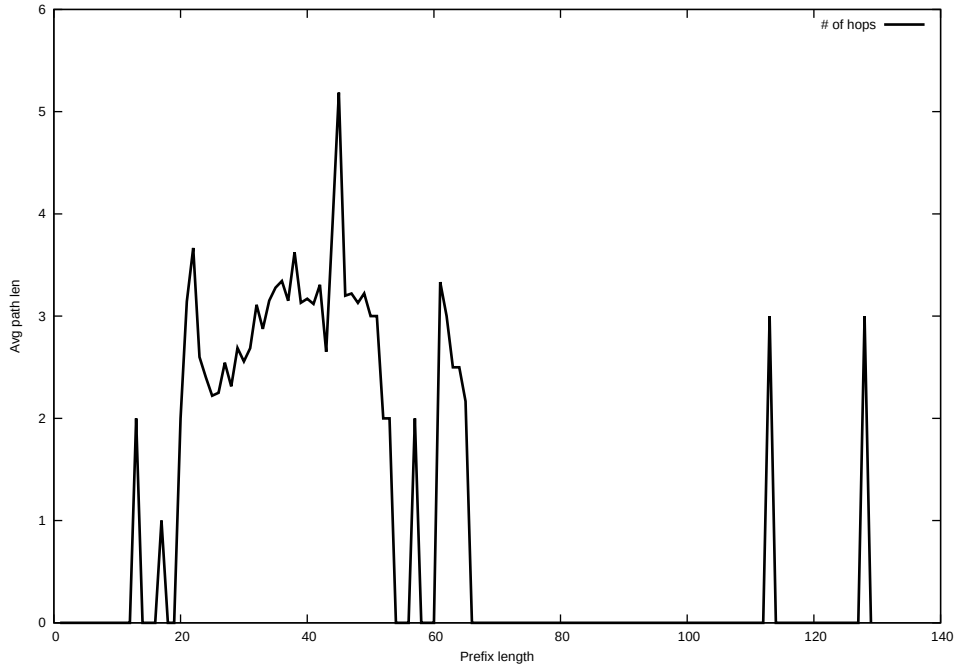
2013-04-20



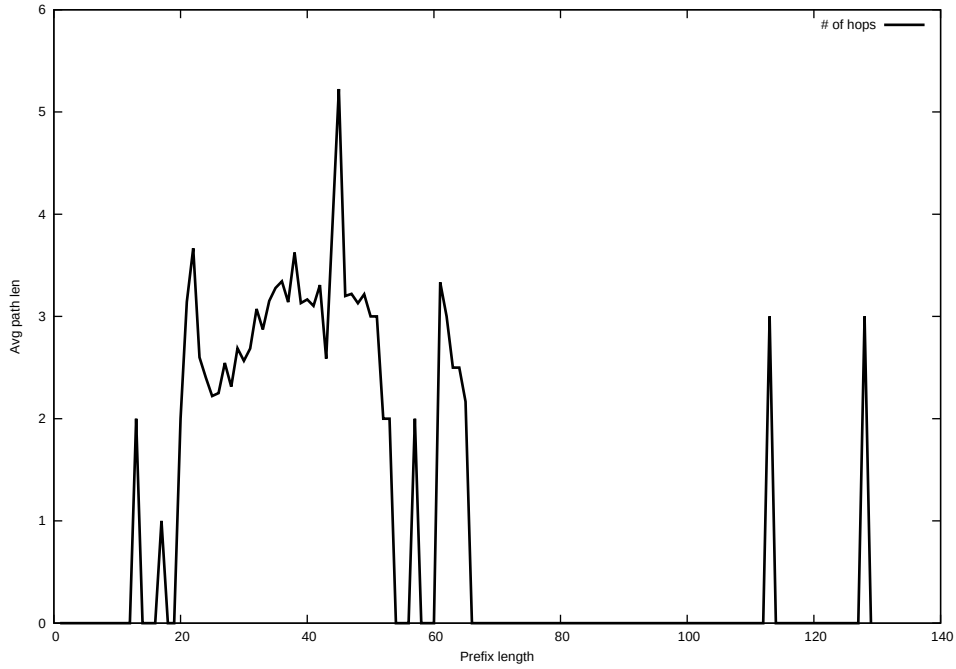
2013-04-21



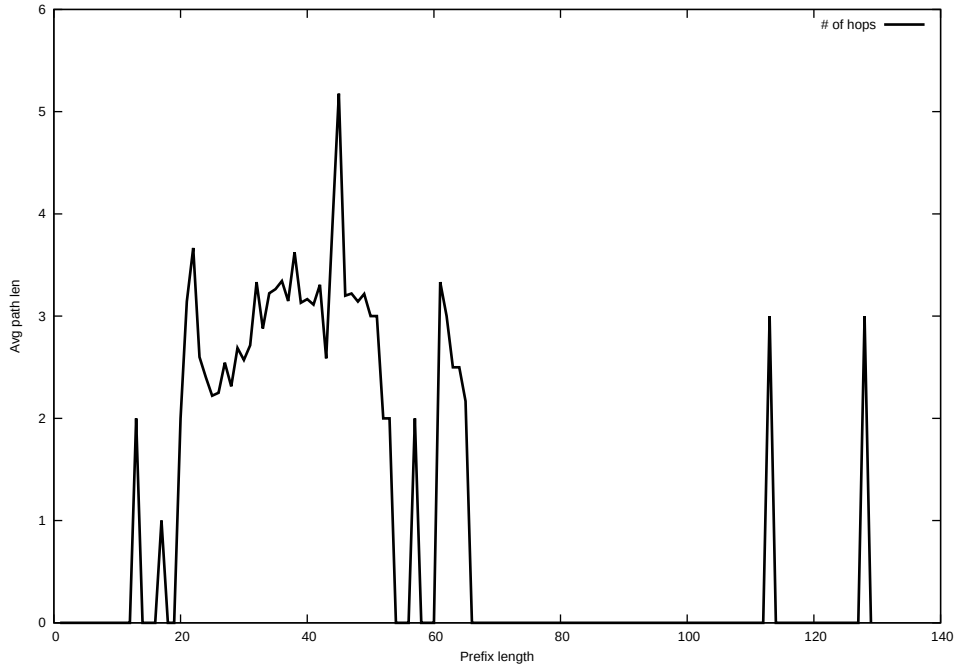
2013-04-22



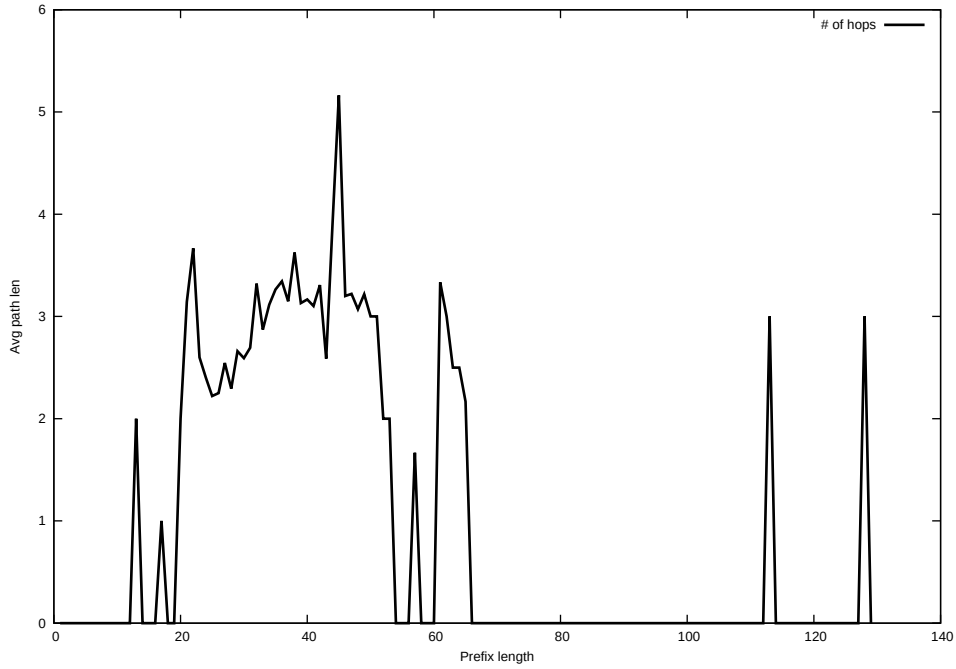
2013-04-23



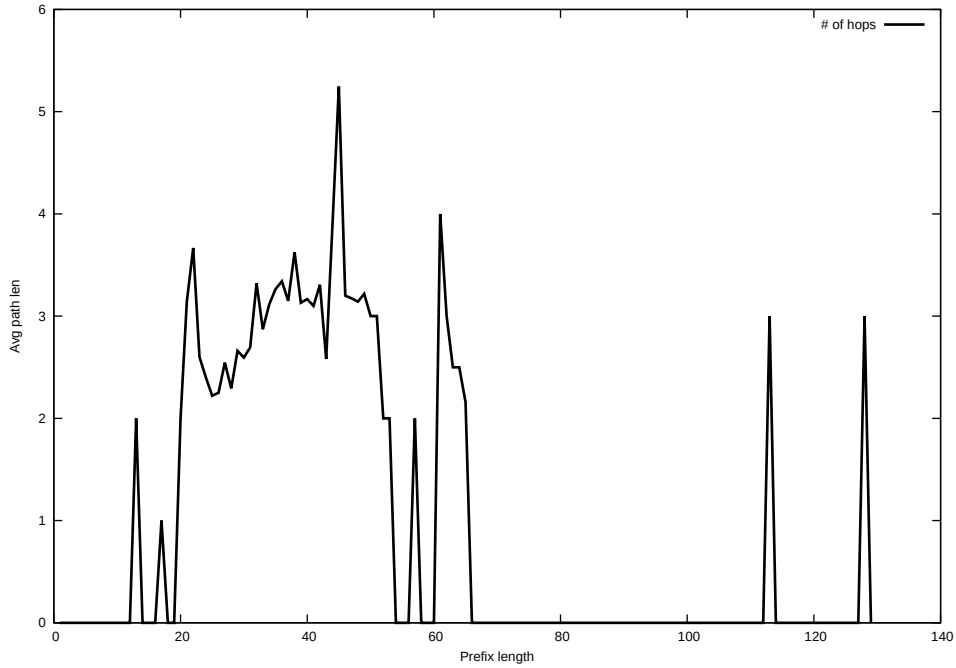
2013-04-24



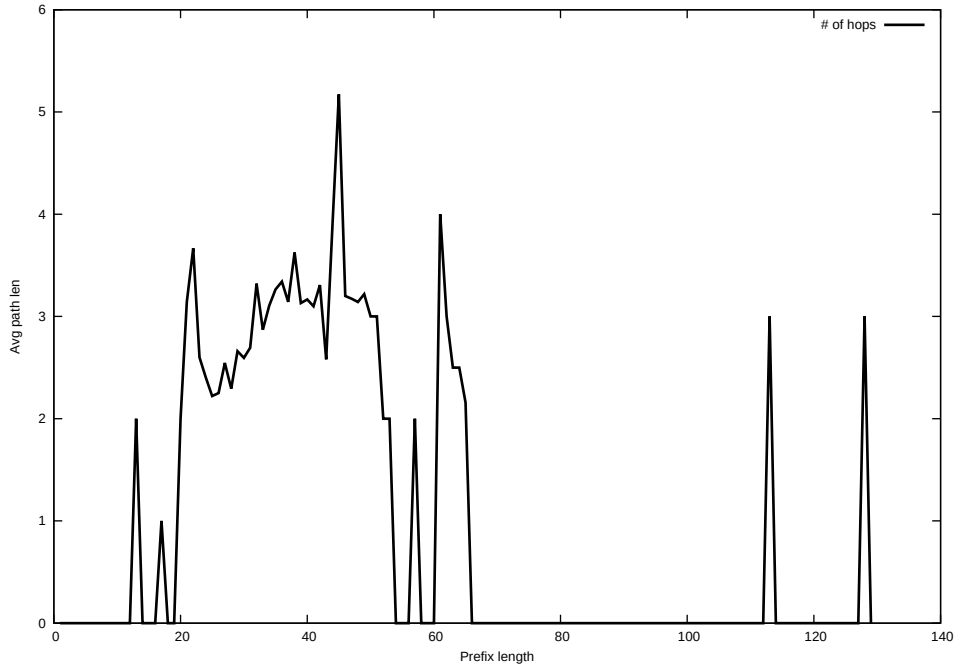
2013-04-25



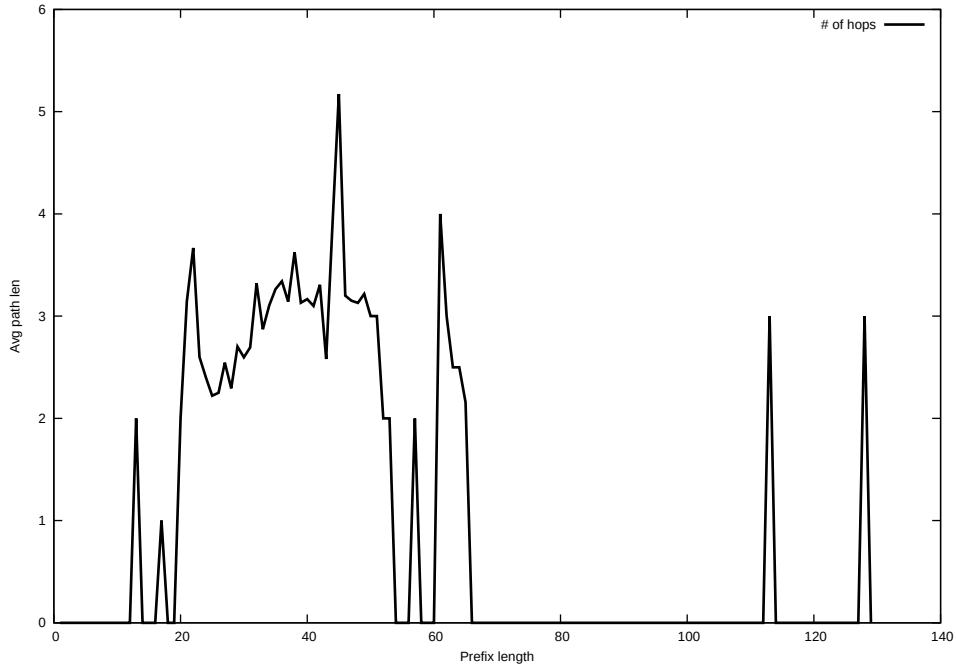
2013-04-26



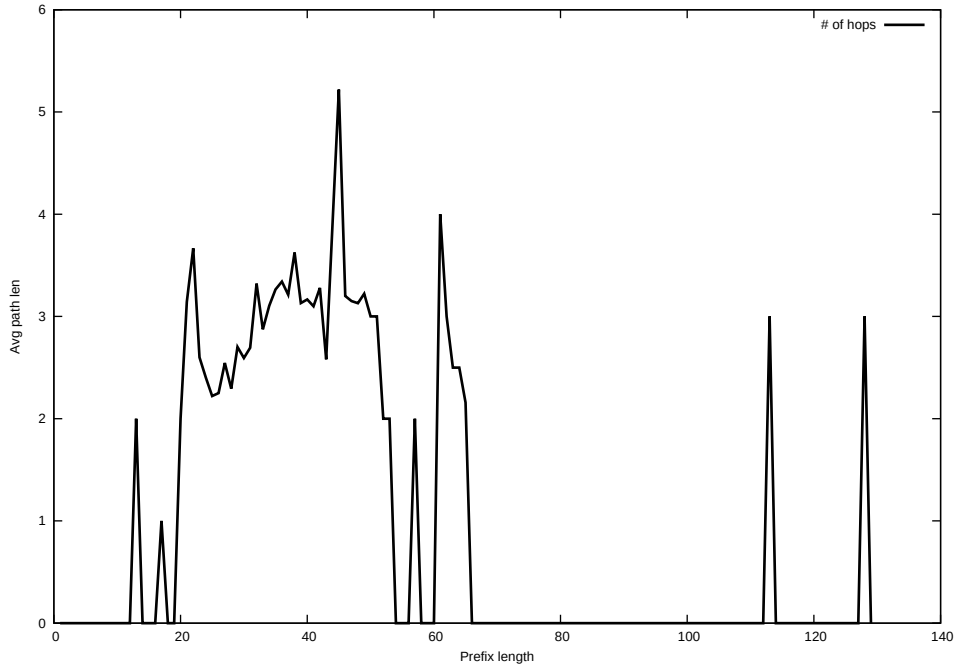
2013-04-27



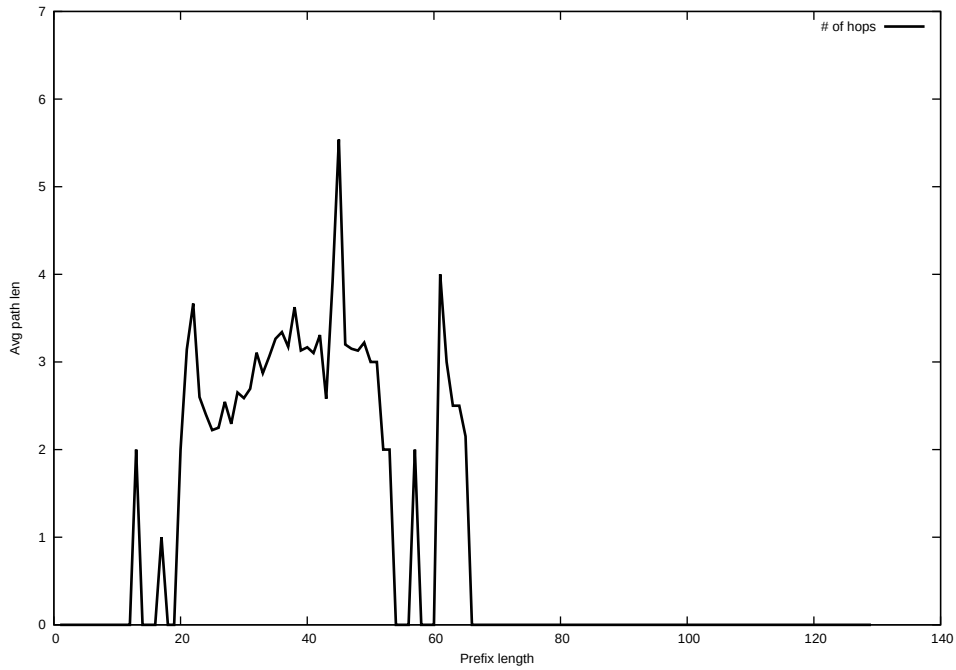
2013-04-28



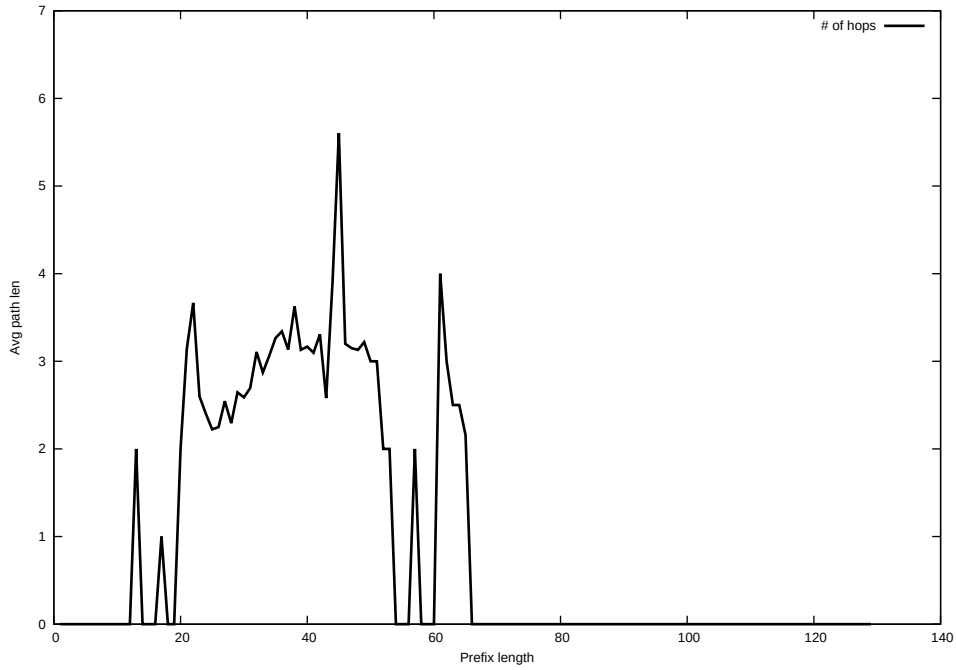
2013-04-29



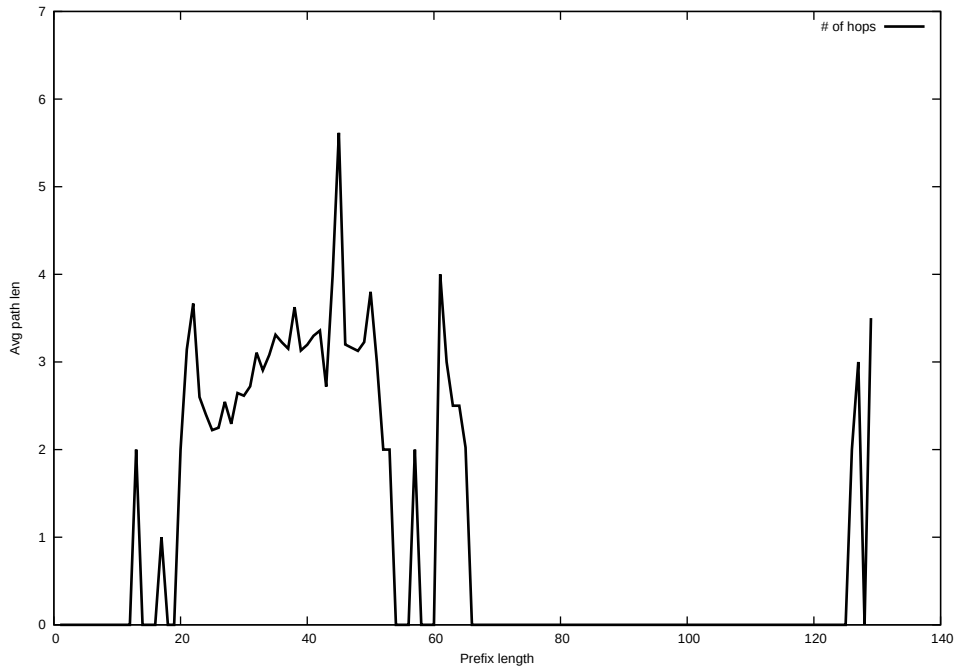
2013-04-30



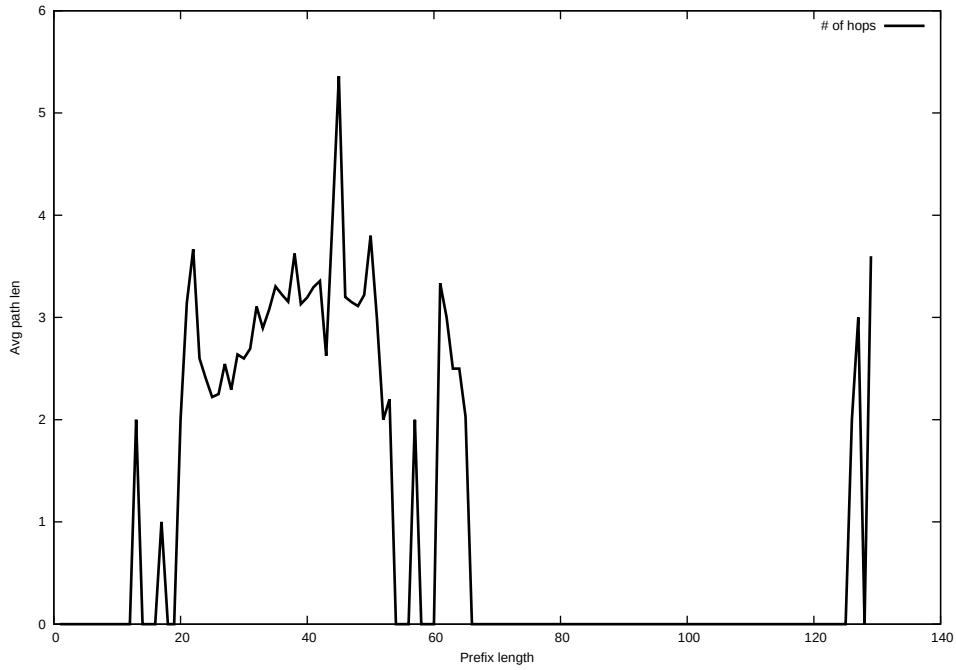
2013-05-01



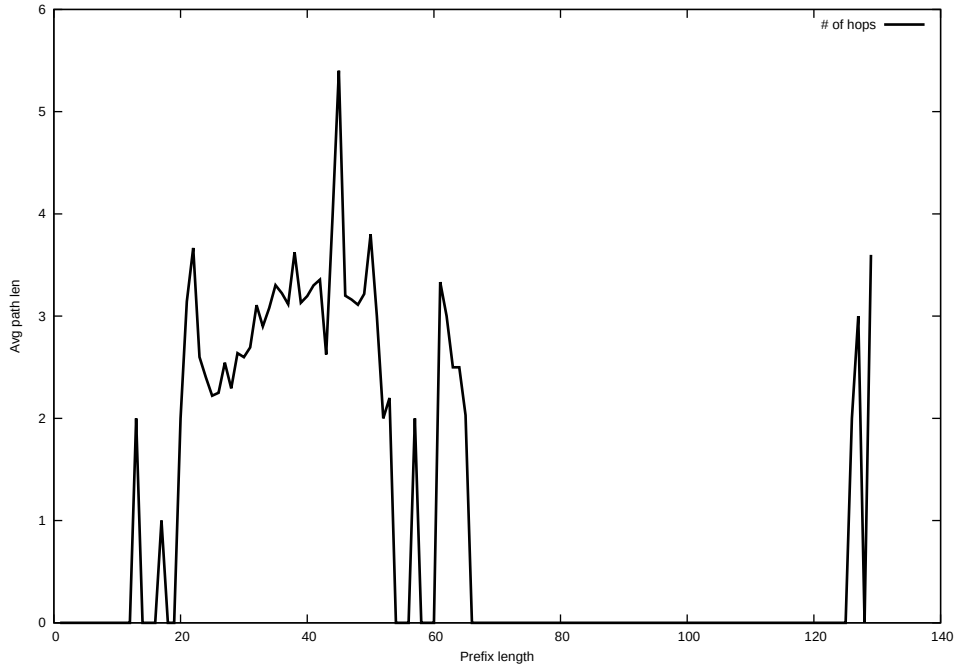
2013-05-02



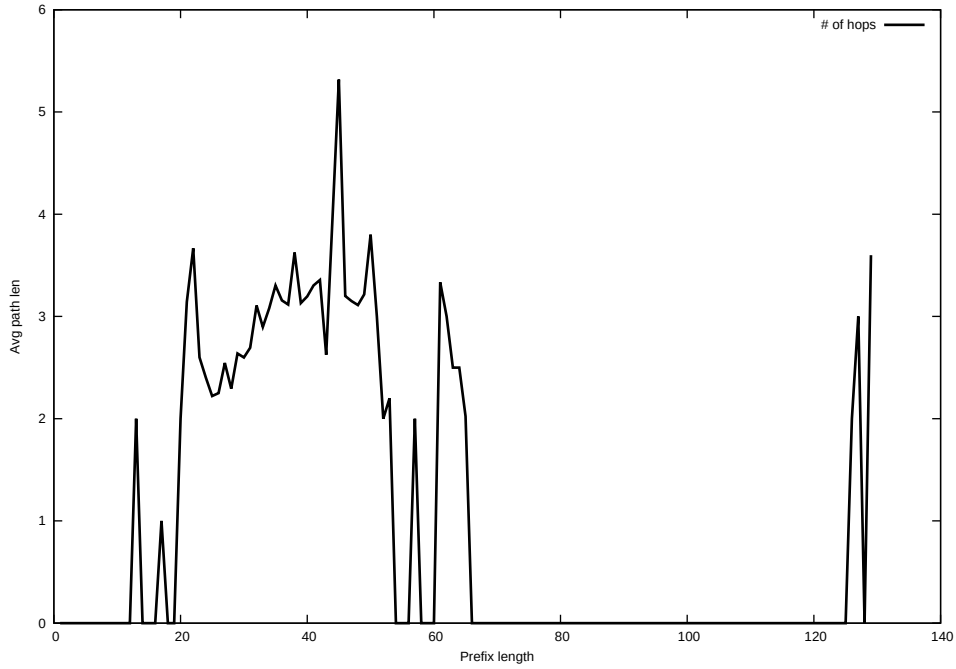
2013-05-03



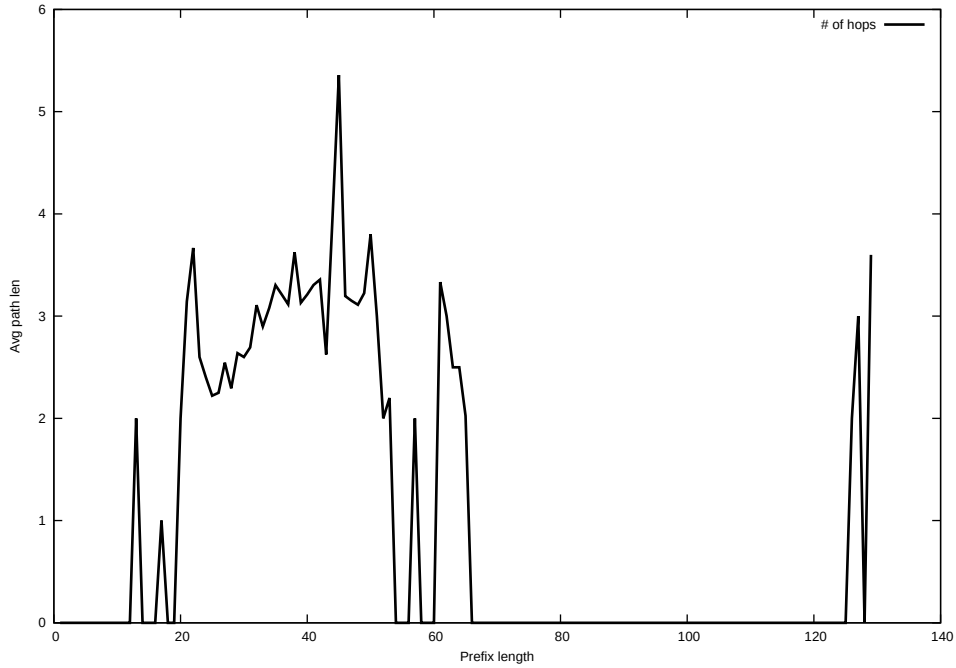
2013-05-04



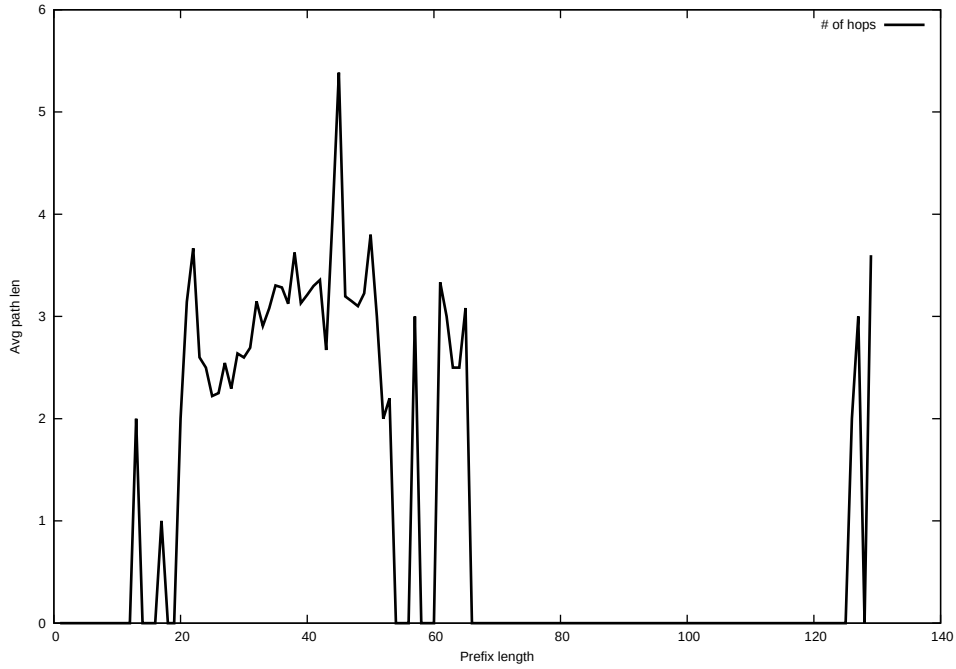
2013-05-05



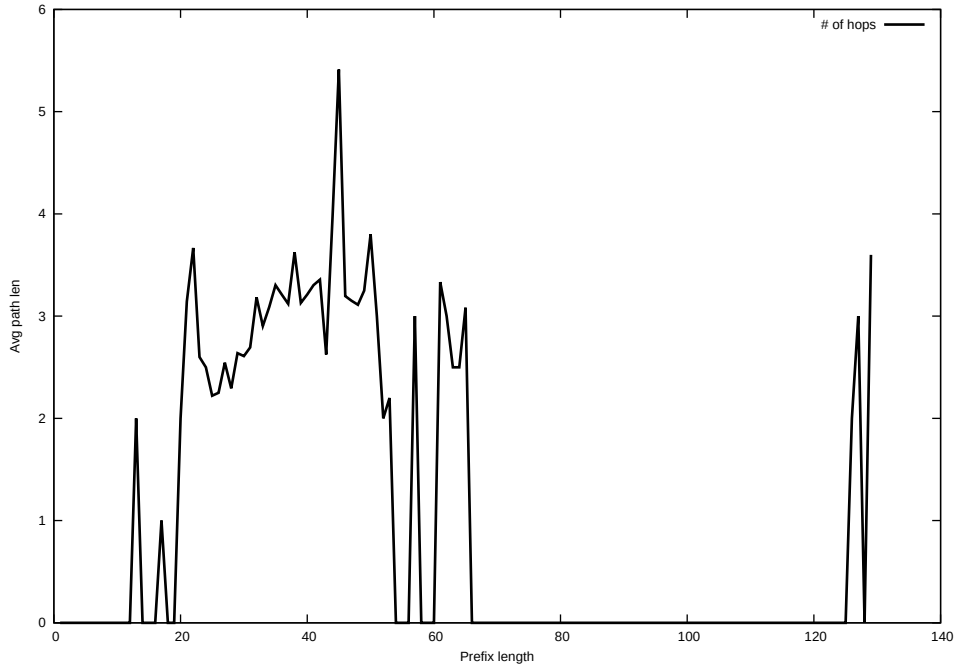
2013-05-06



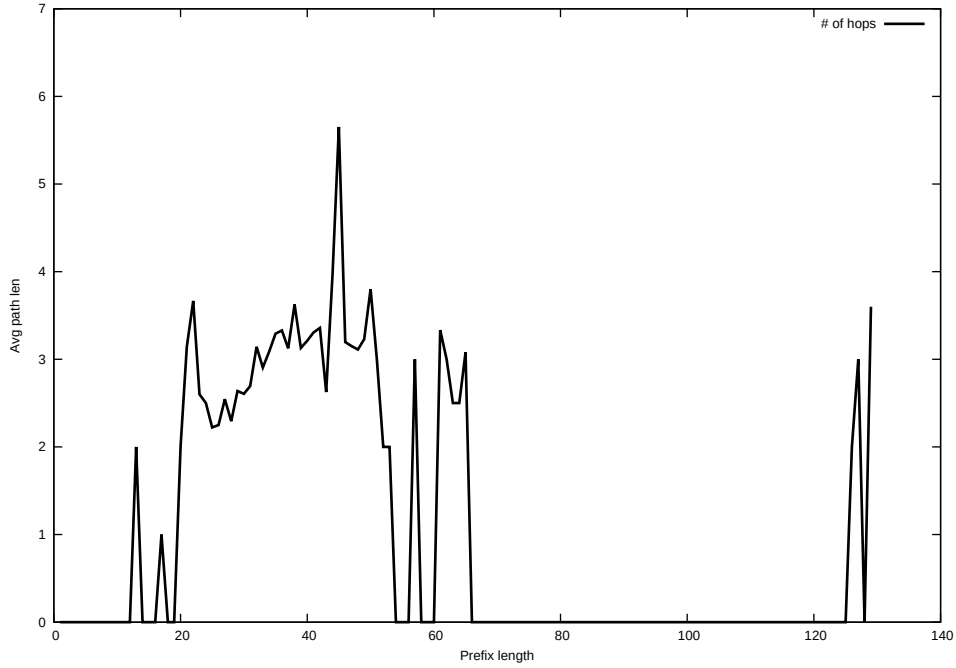
2013-05-07



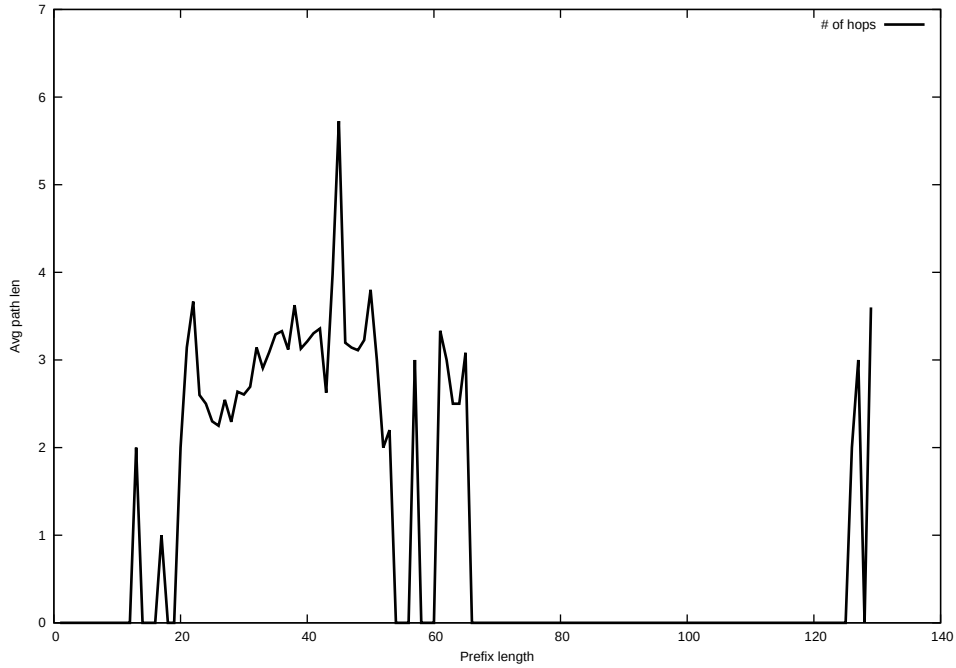
2013-05-08



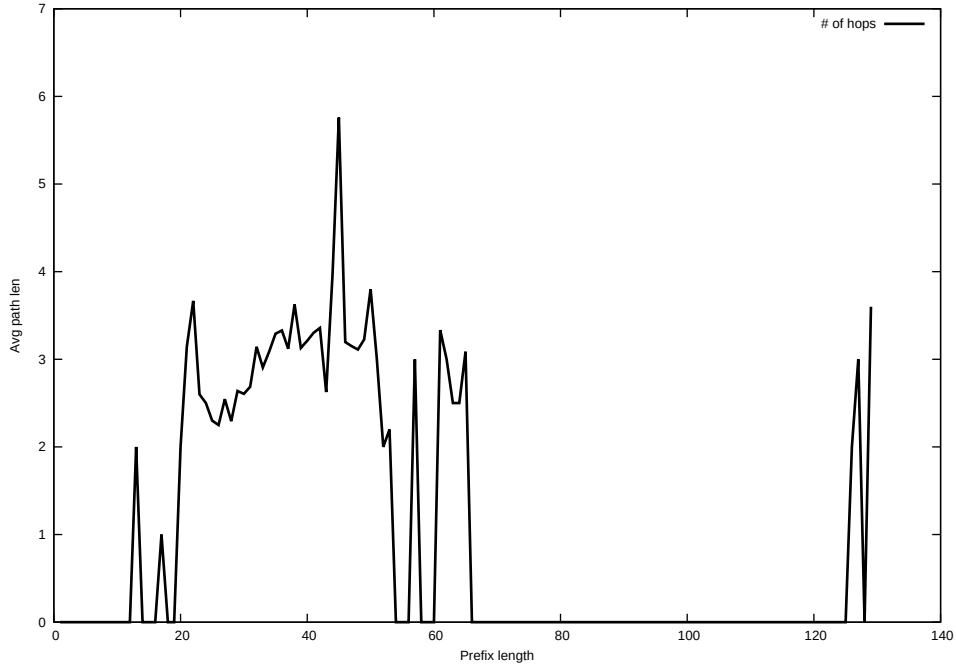
2013-05-09



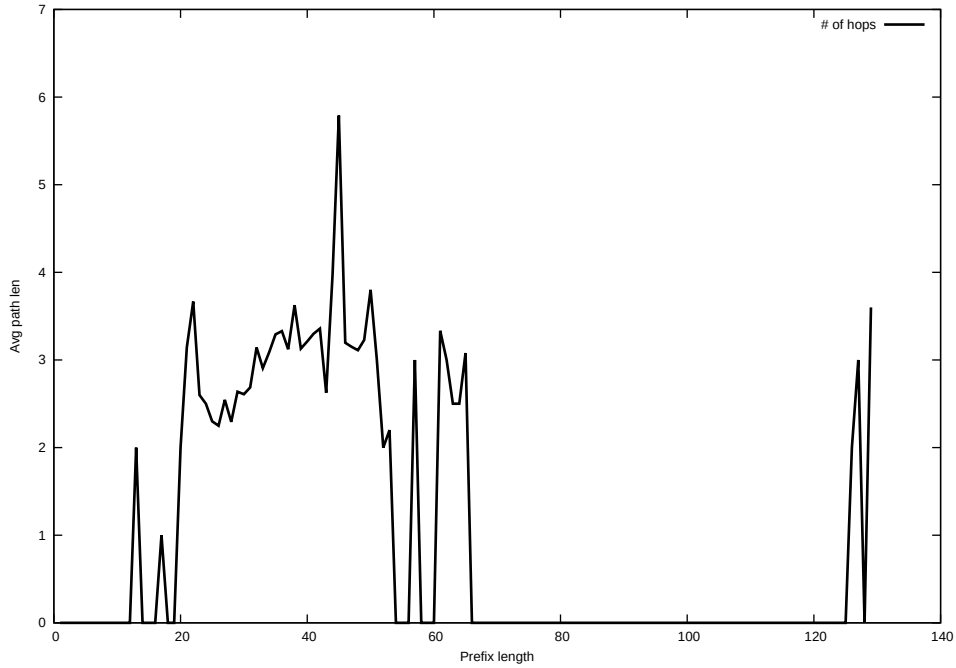
2013-05-10



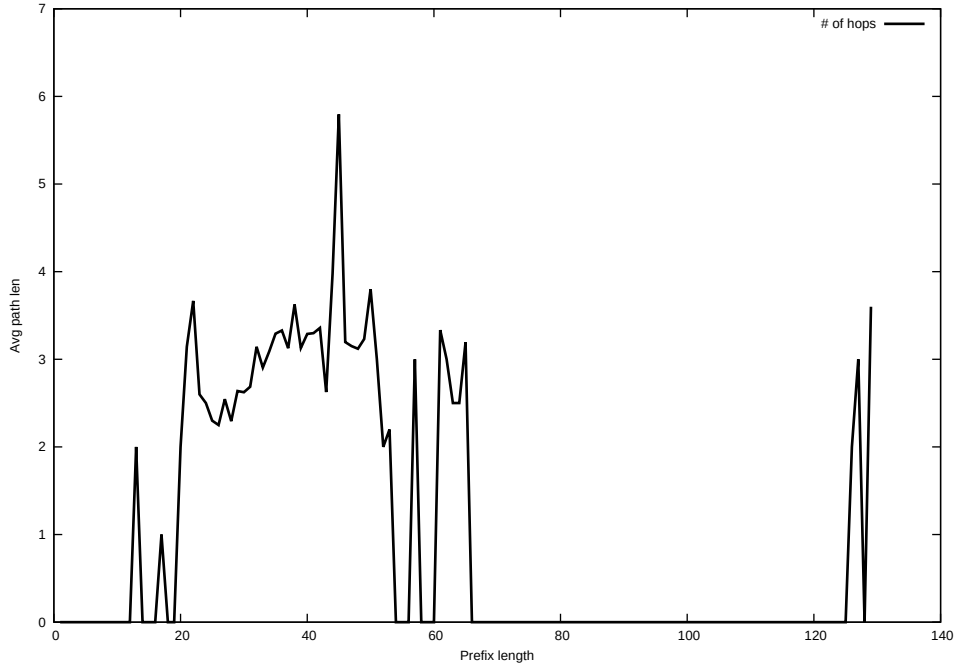
2013-05-11



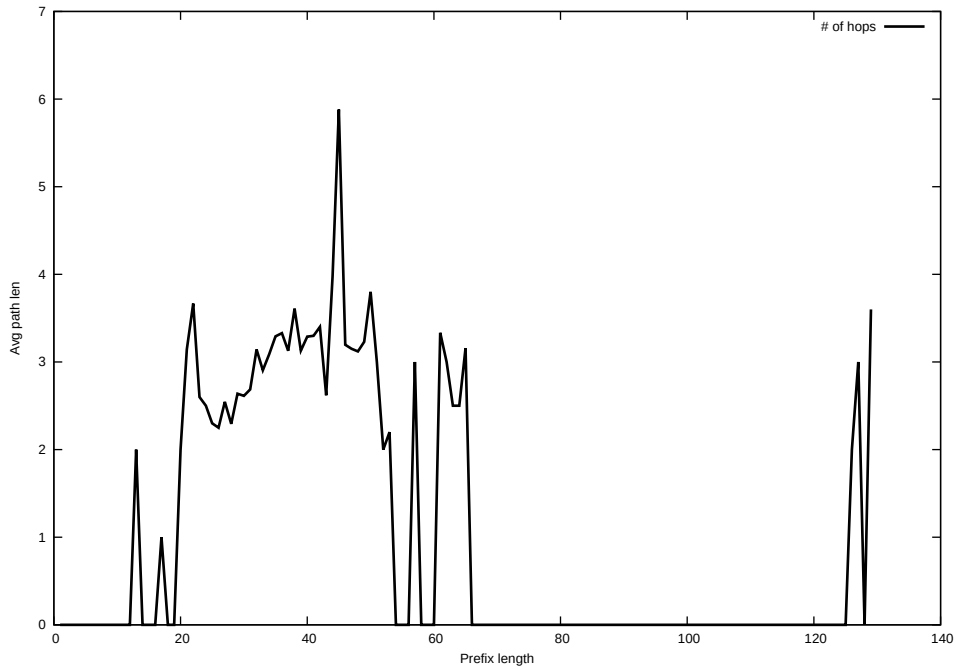
2013-05-12



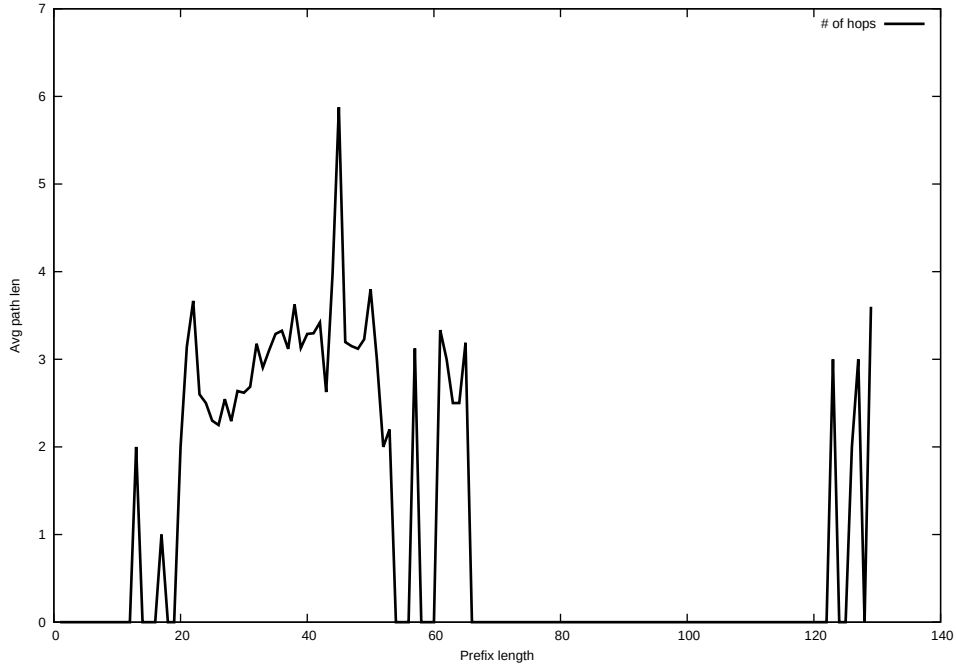
2013-05-13



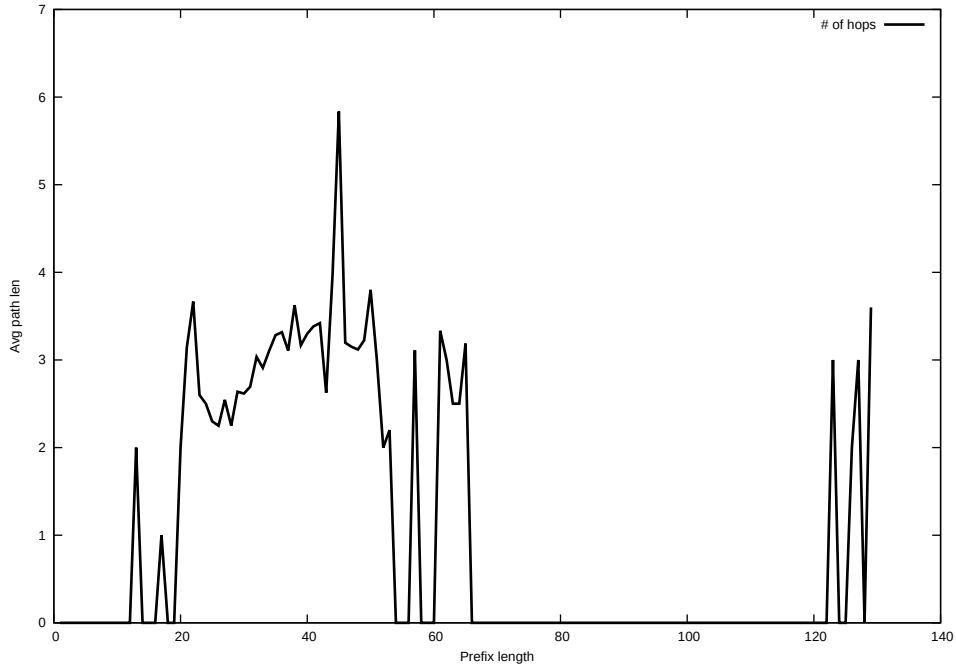
2013-05-14



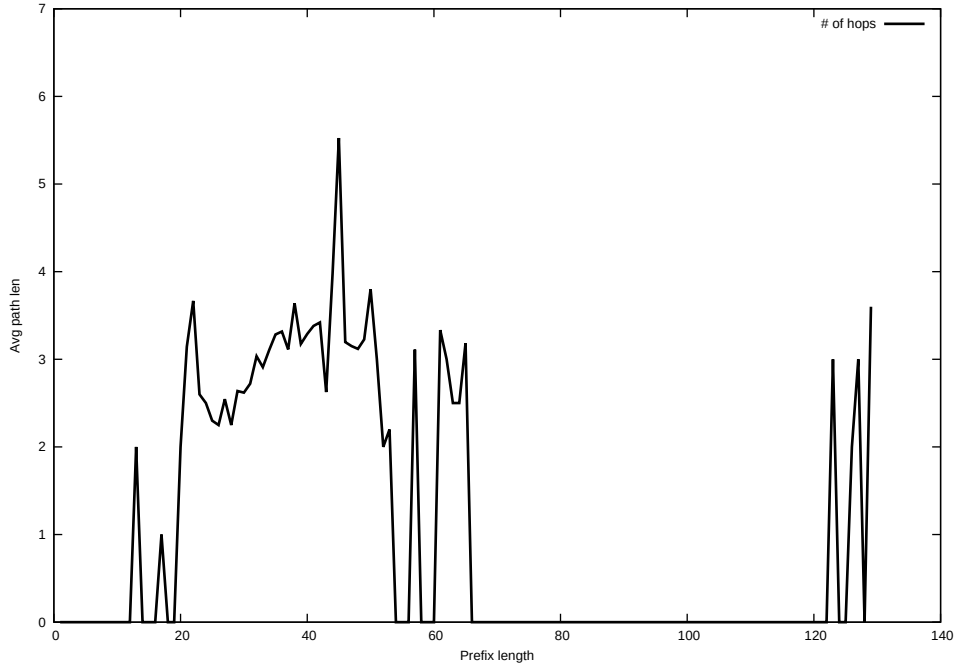
2013-05-15



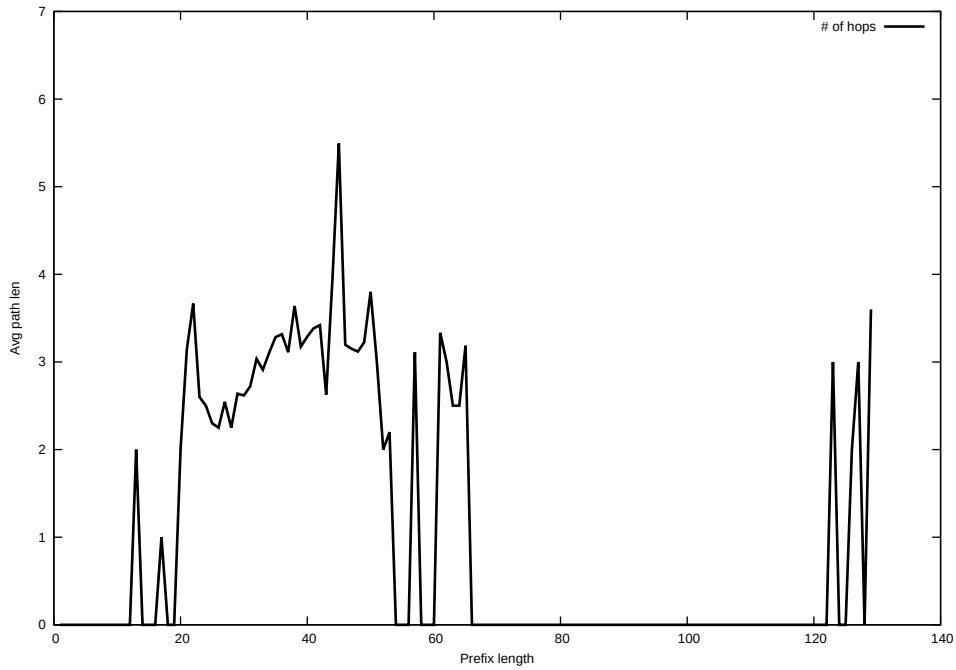
2013-05-16



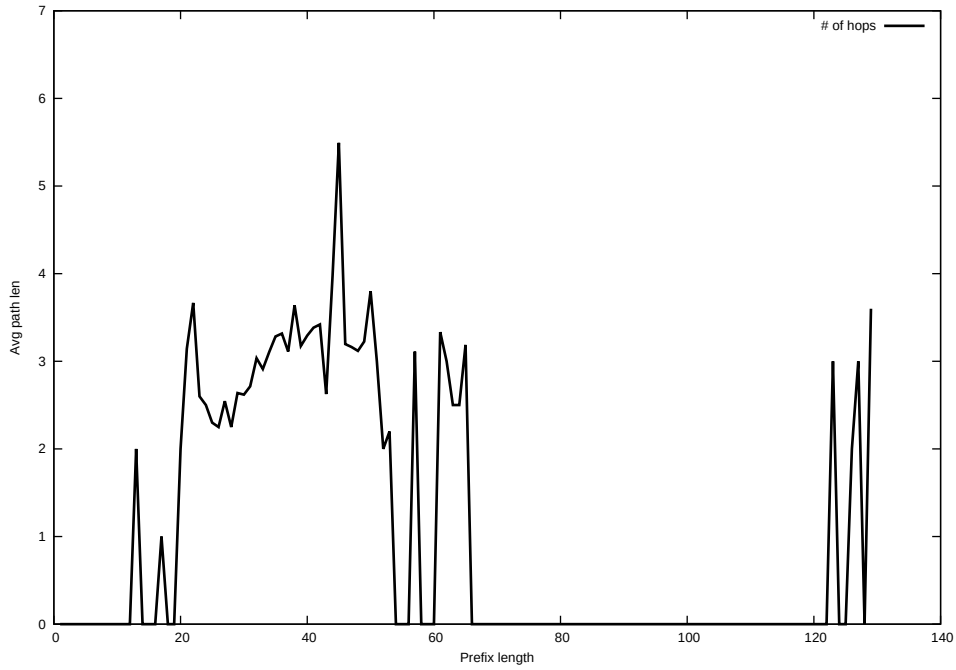
2013-05-17



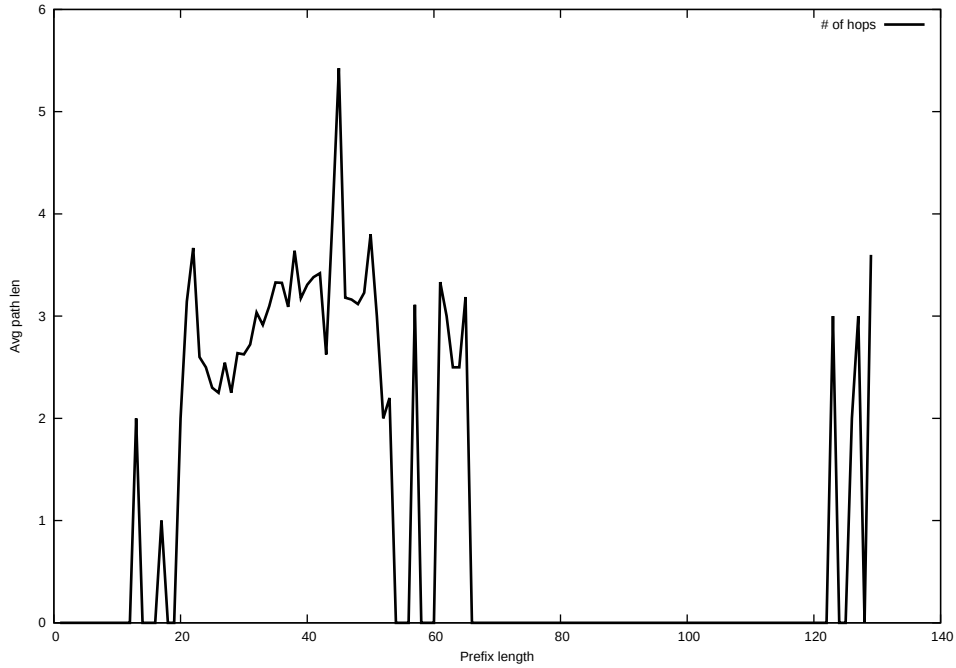
2013-05-18



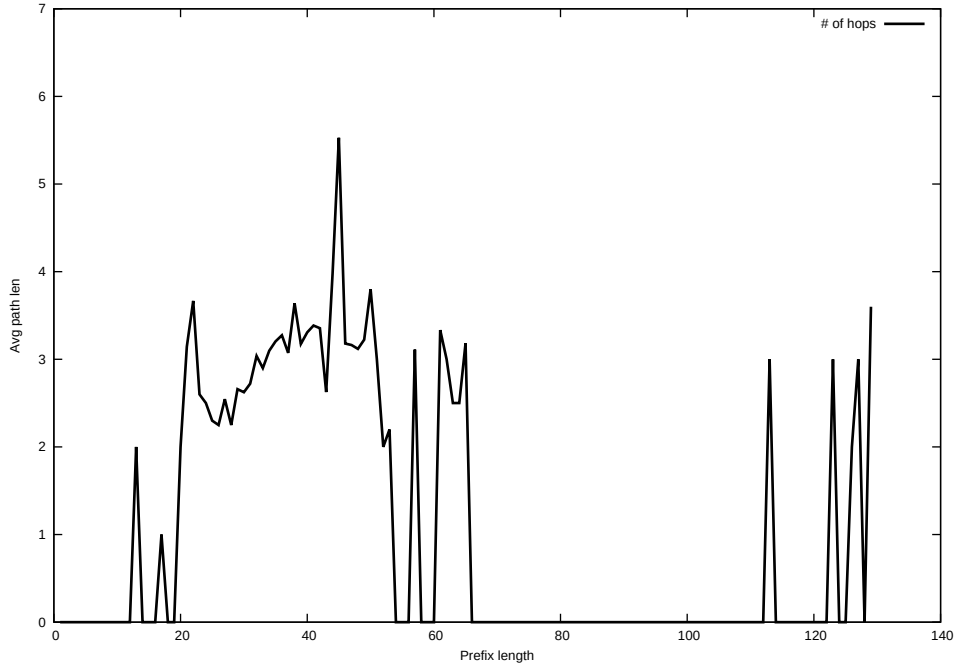
2013-05-19



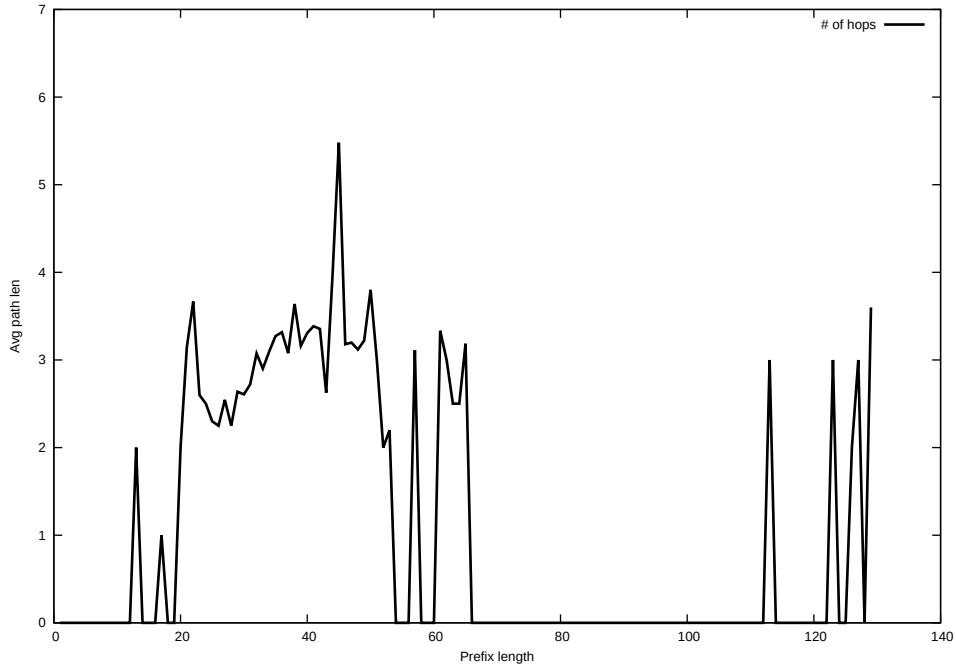
2013-05-20



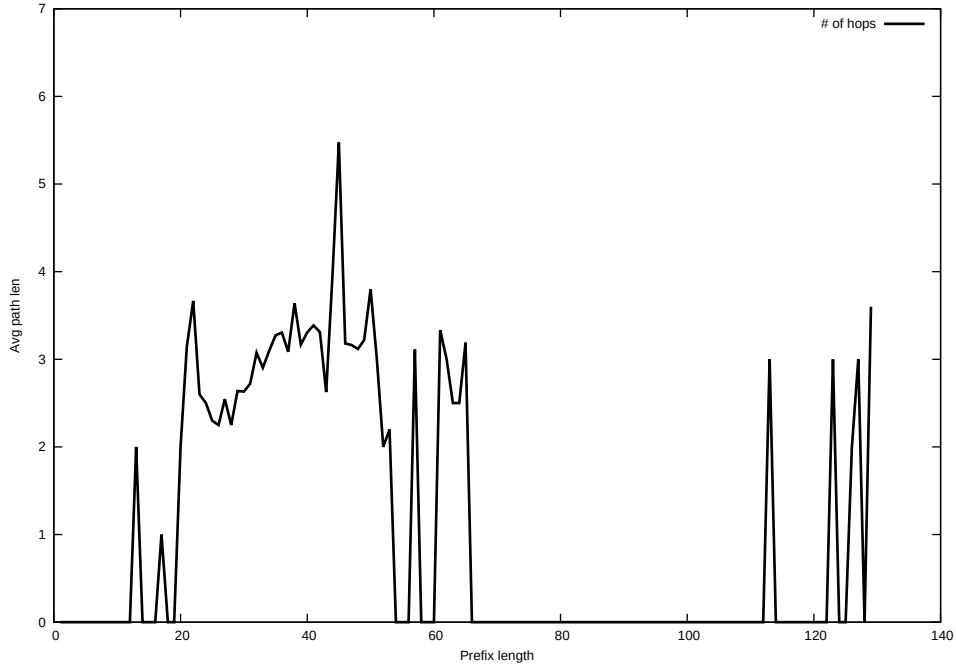
2013-05-21



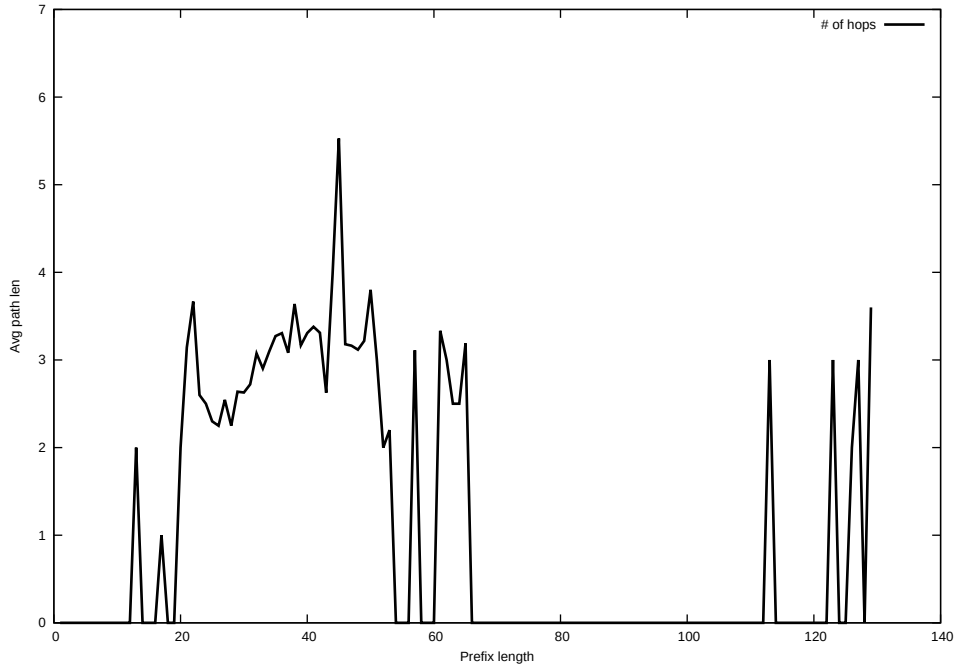
2013-05-22



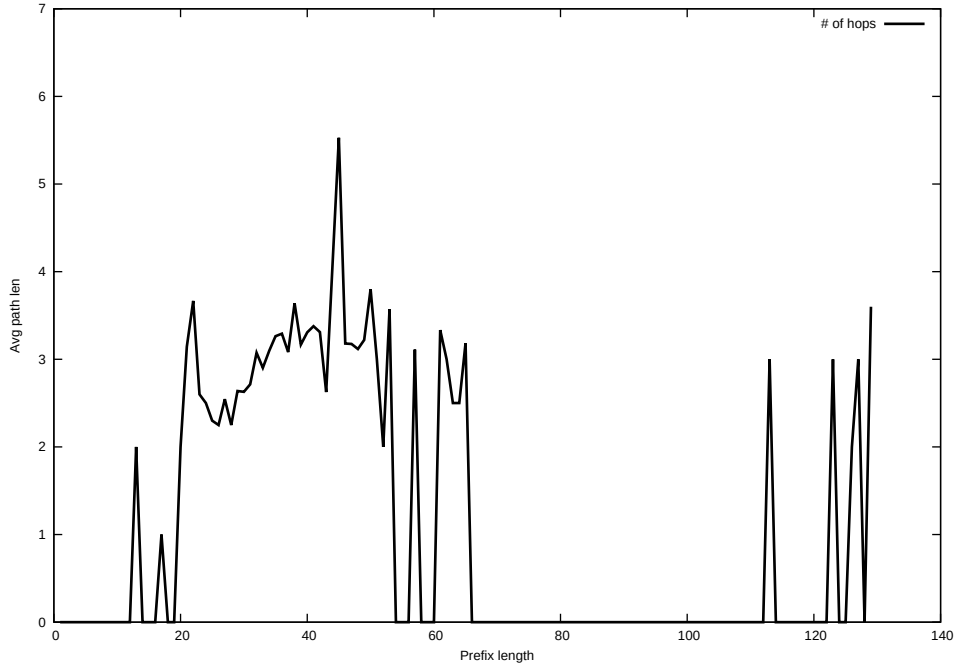
2013-05-23



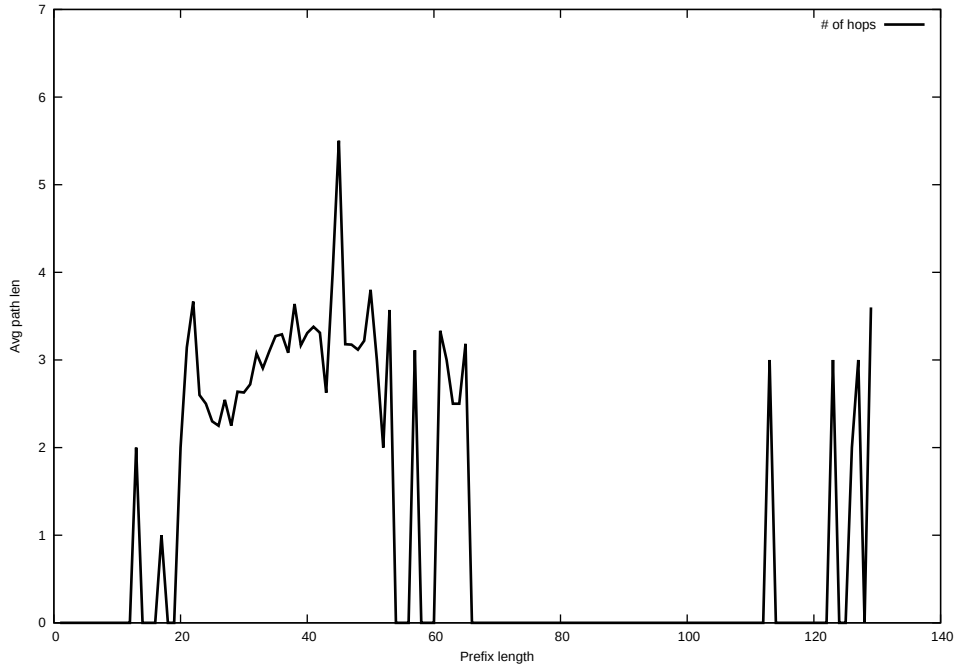
2013-05-24



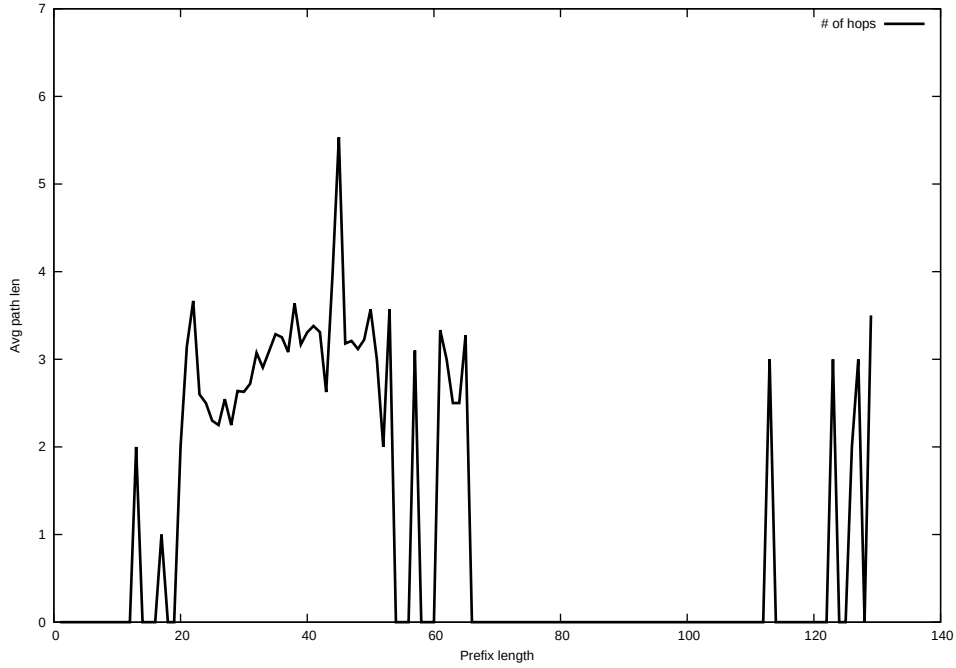
2013-05-25



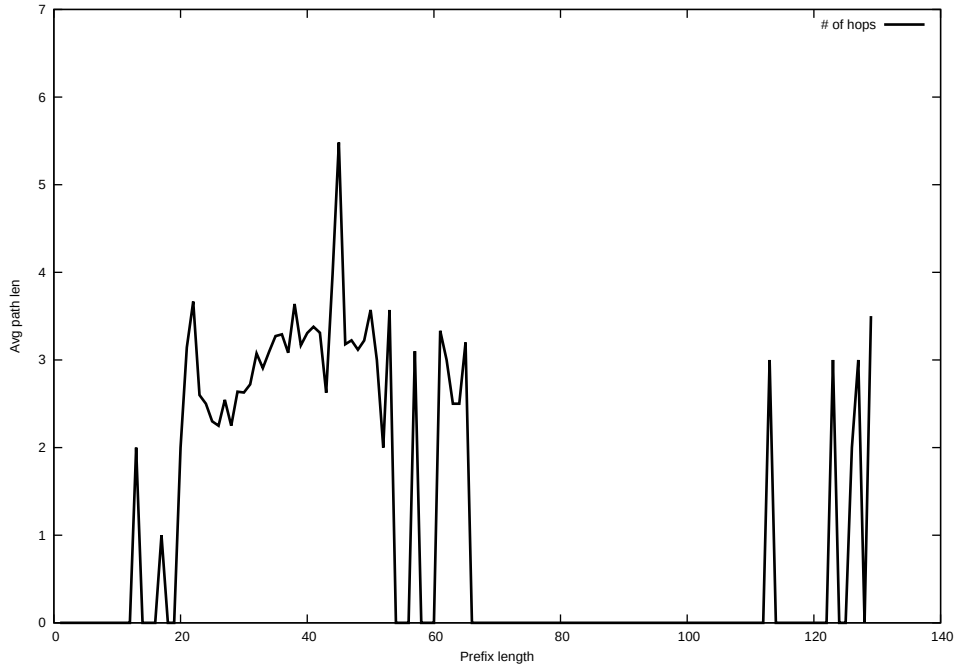
2013-05-26



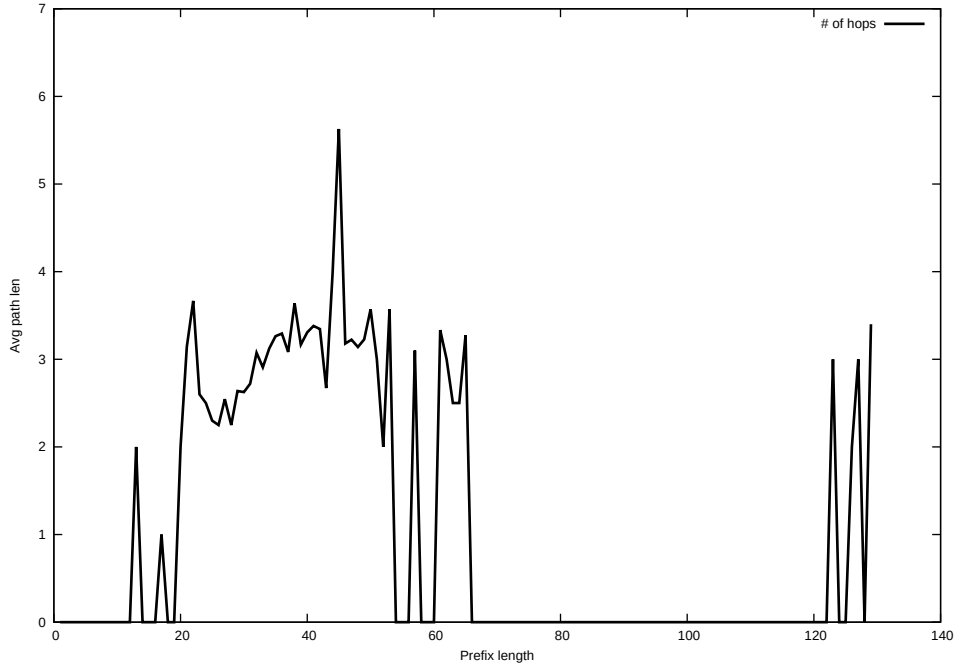
2013-05-27



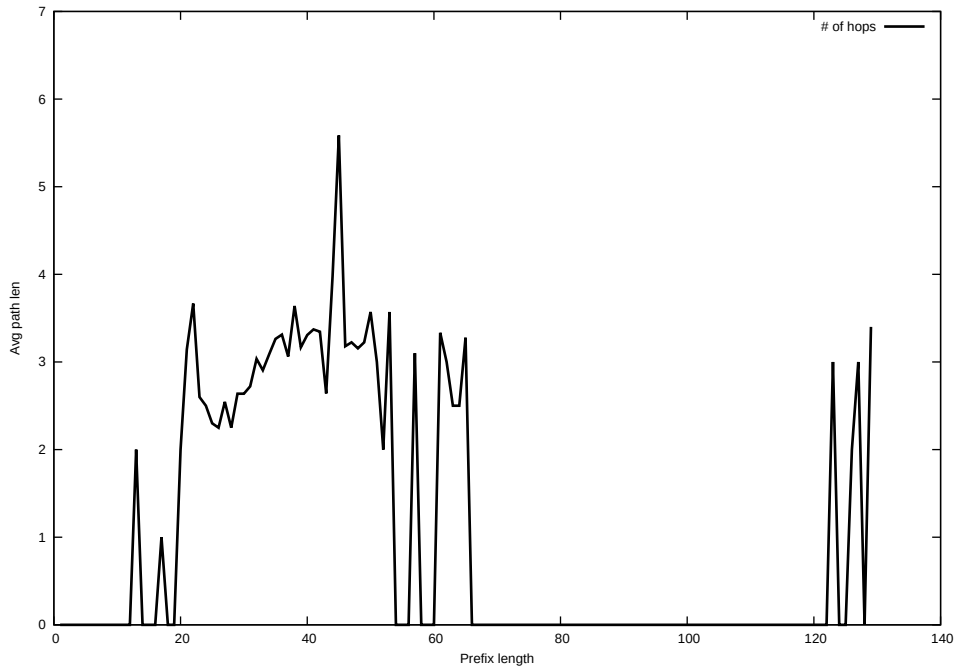
2013-05-28



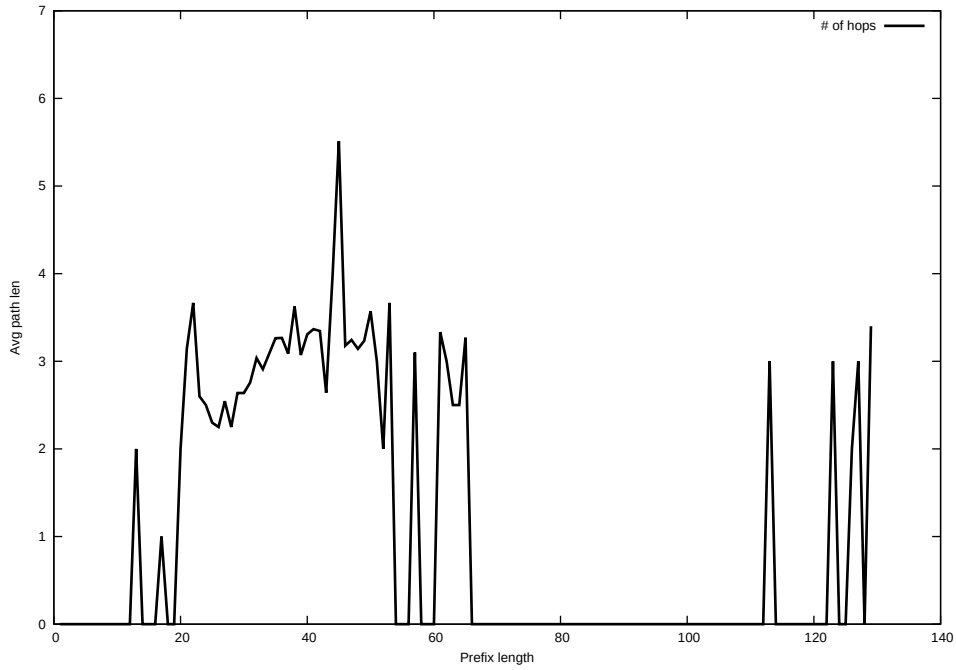
2013-05-29



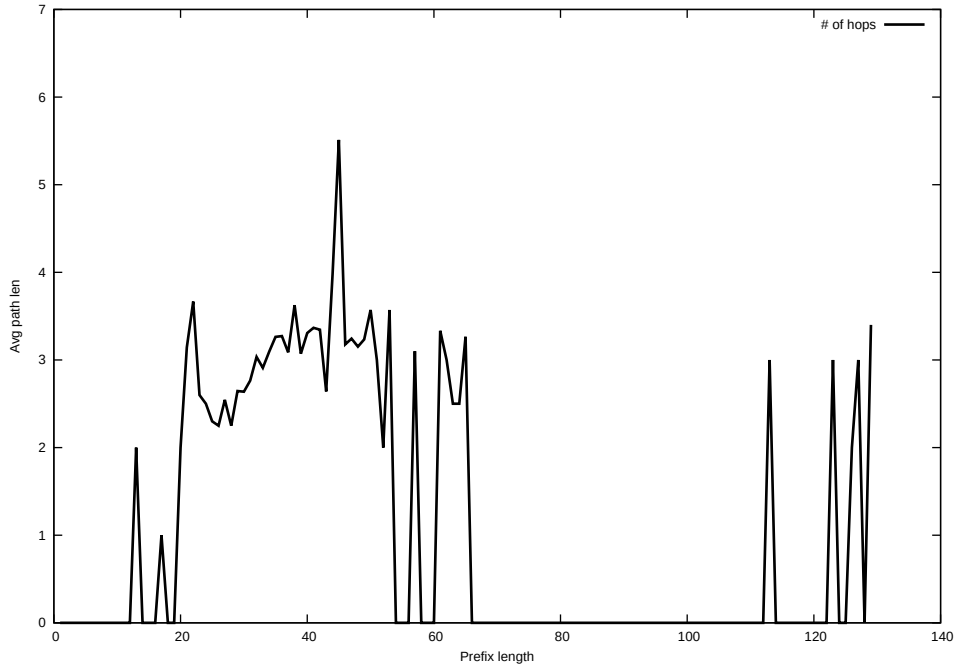
2013-05-30



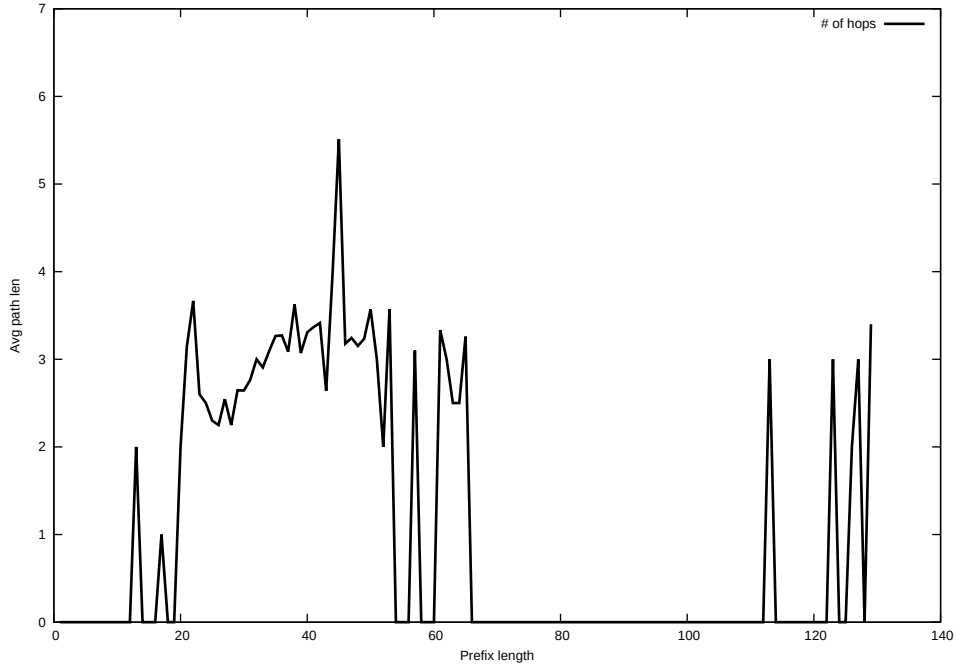
2013-05-31



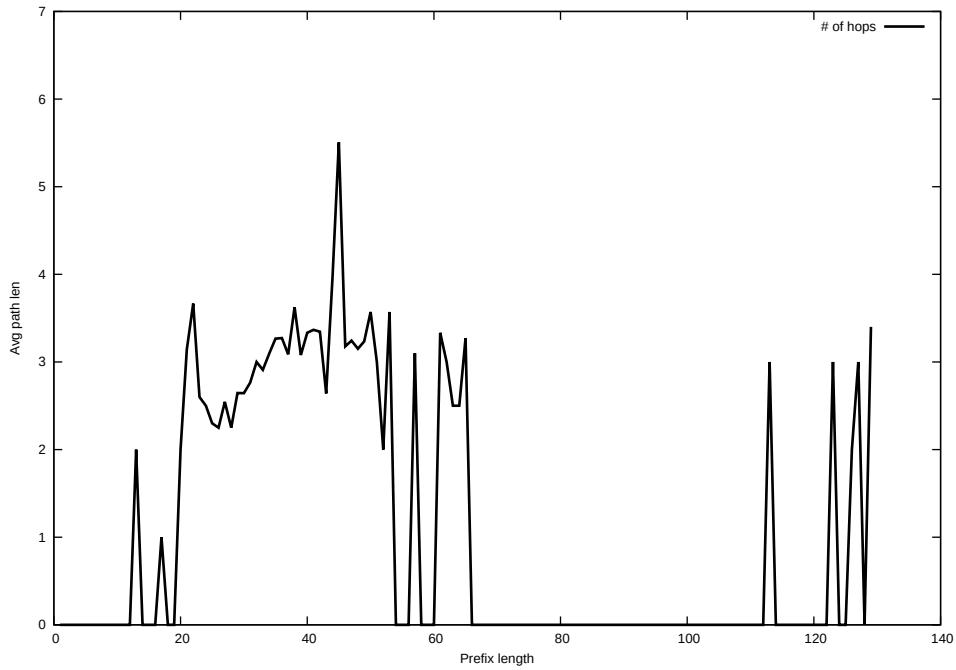
2013-06-01



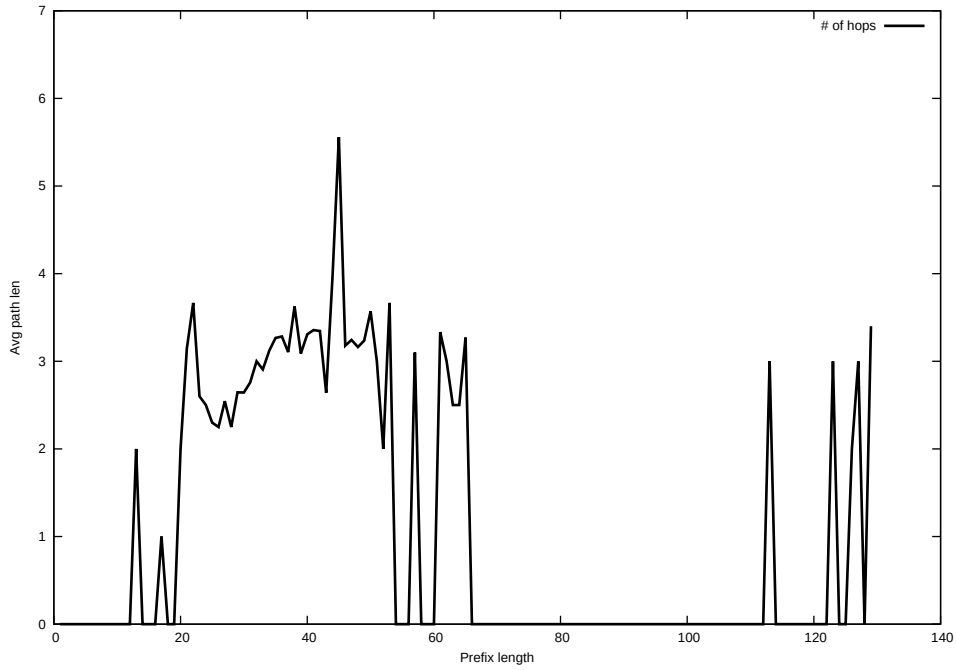
2013-06-02



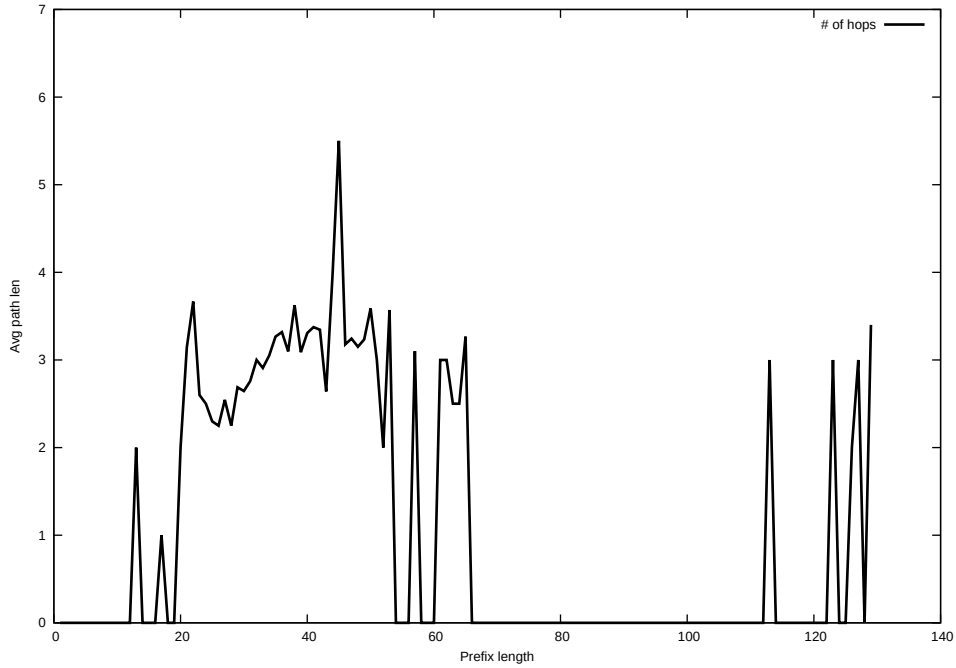
2013-06-03



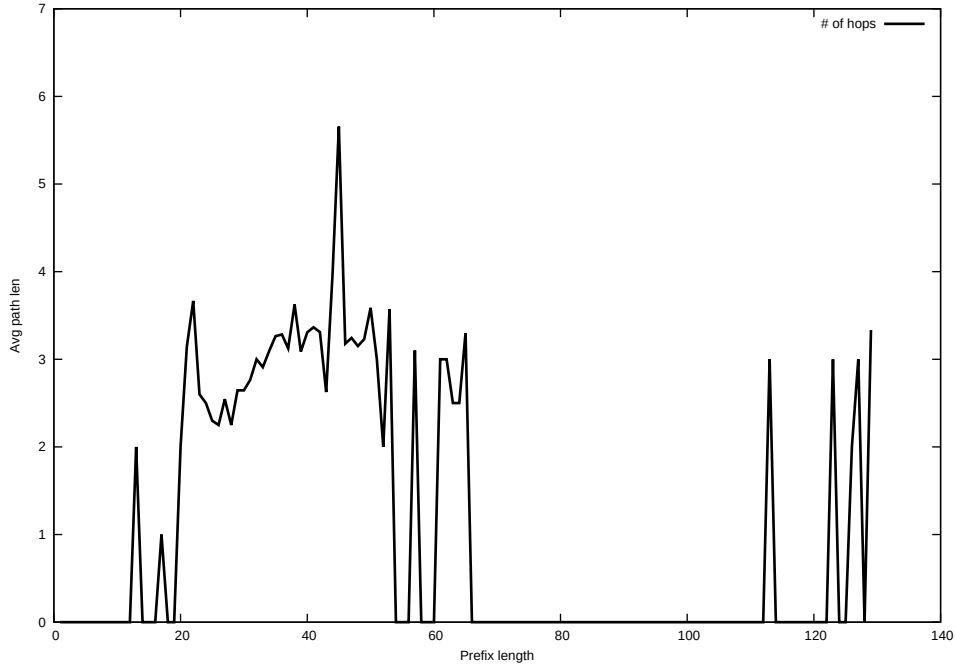
2013-06-04



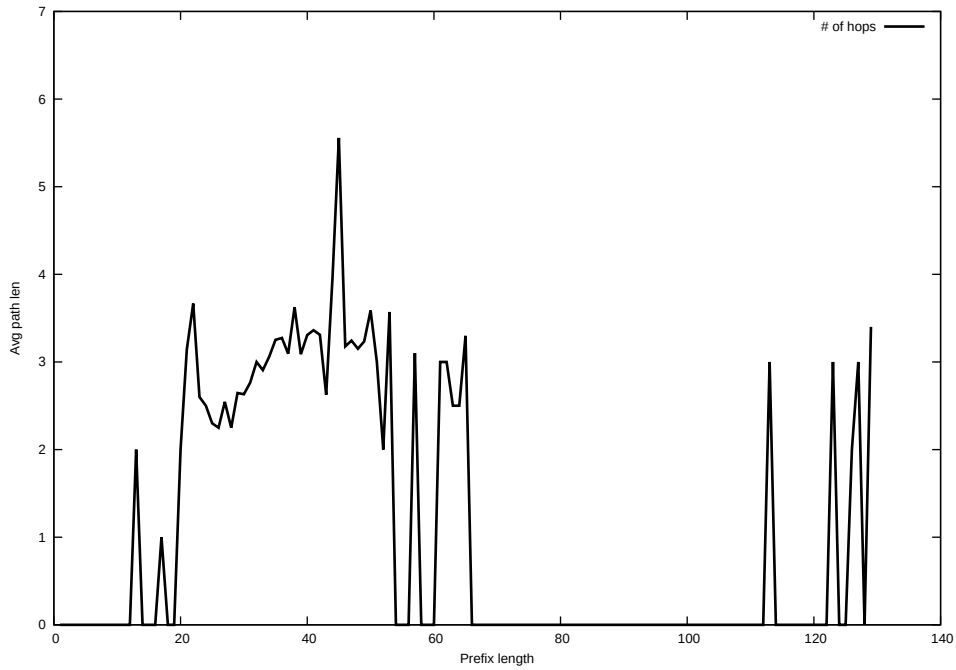
2013-06-05



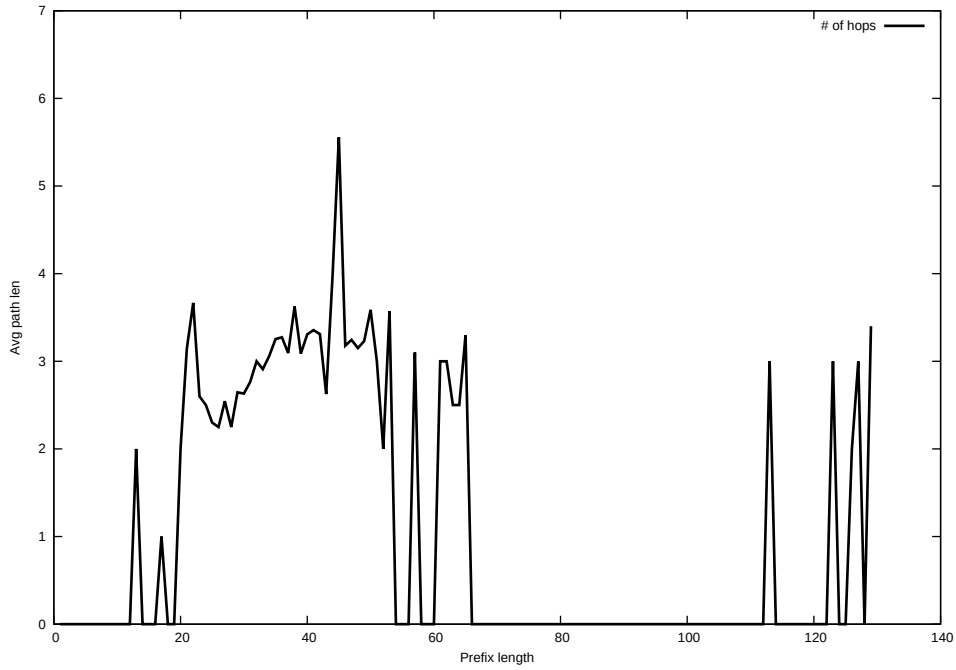
2013-06-06



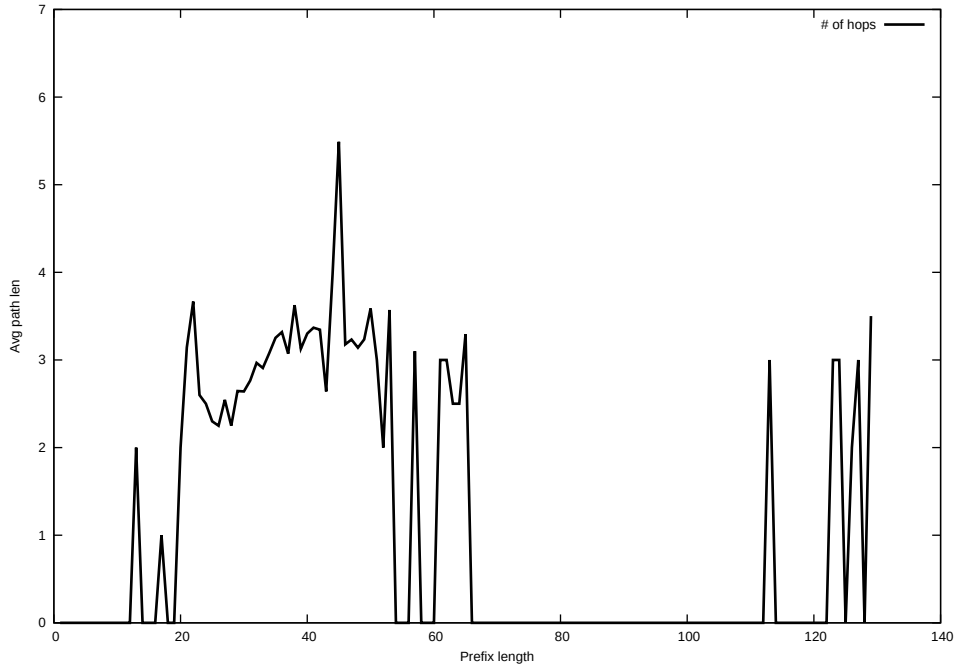
2013-06-07



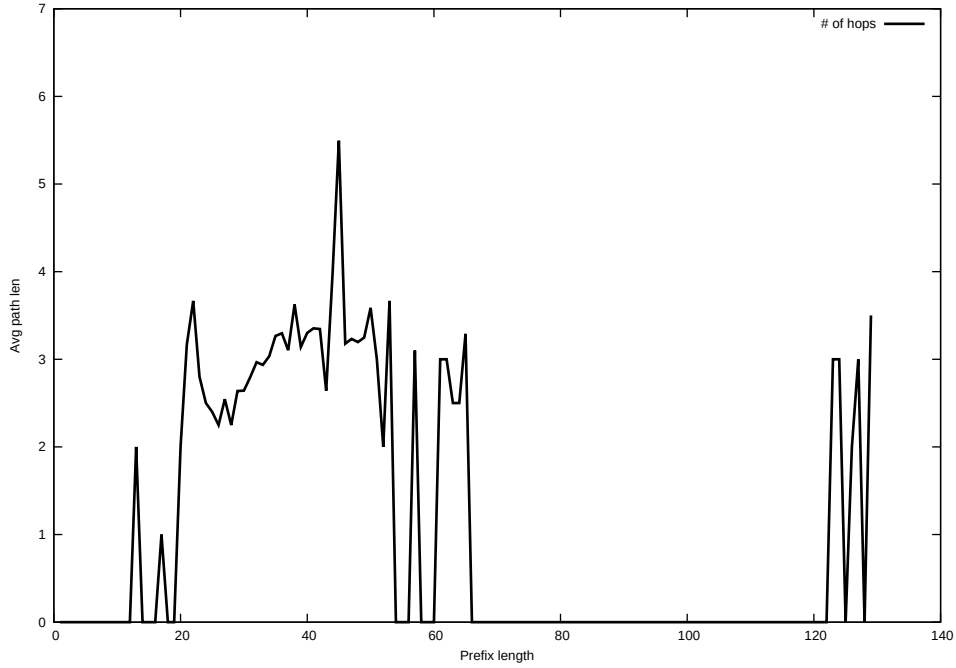
2013-06-09



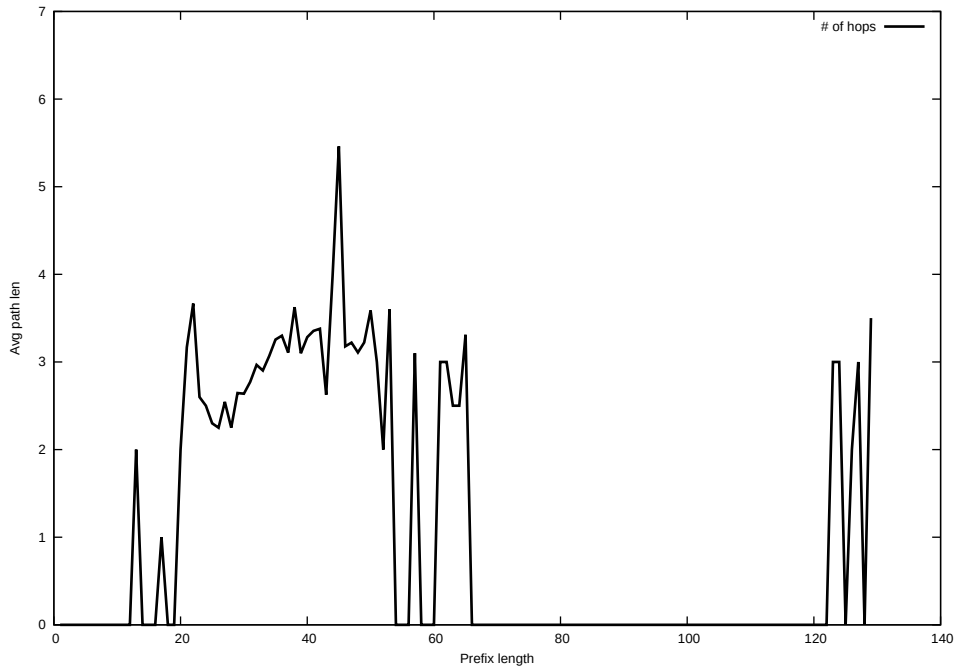
2013-06-10



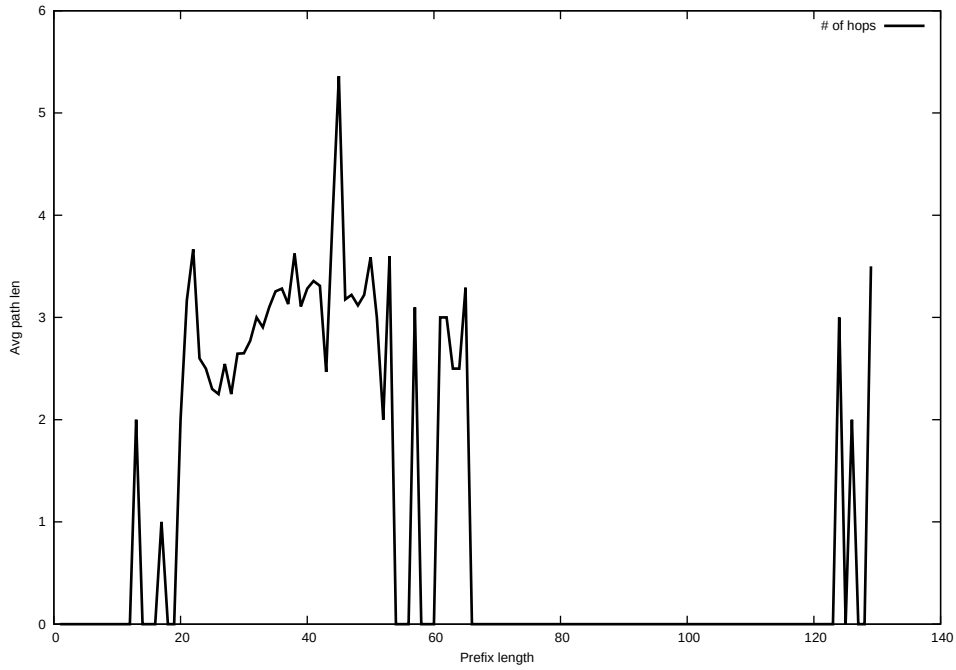
2013-06-12



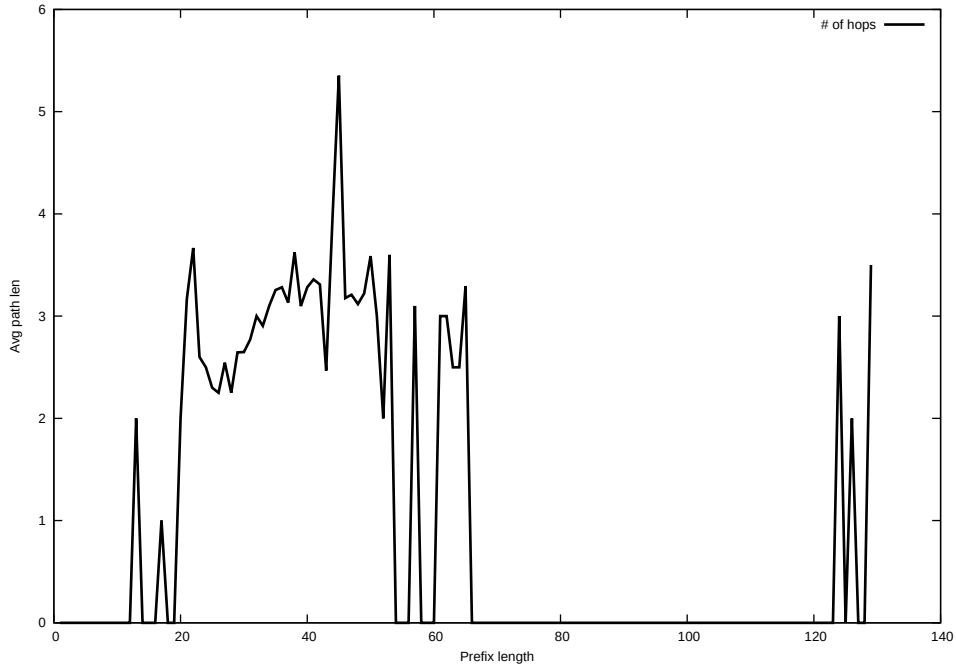
2013-06-13



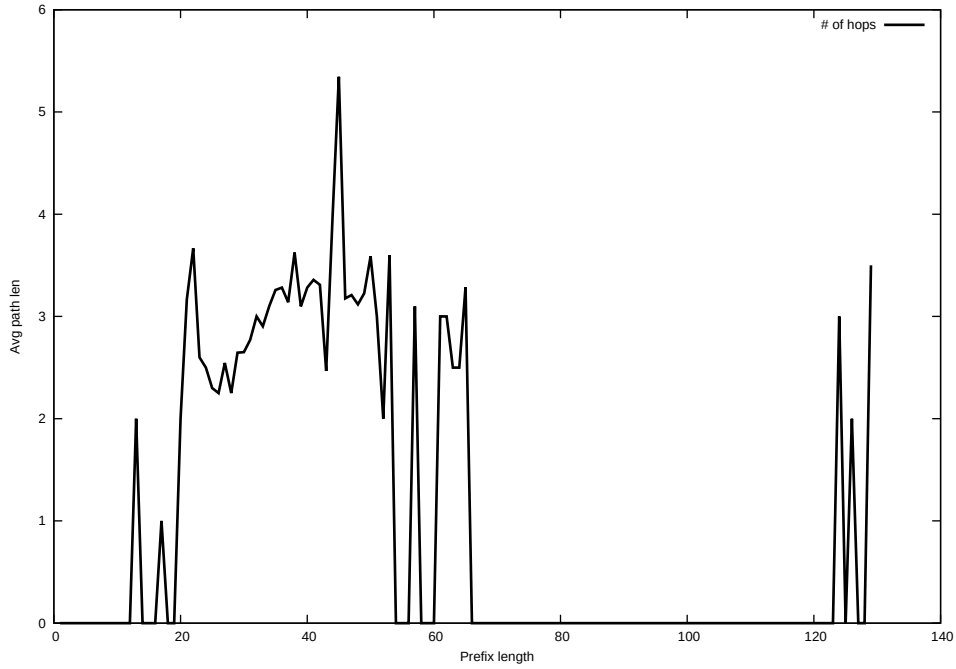
2013-06-14



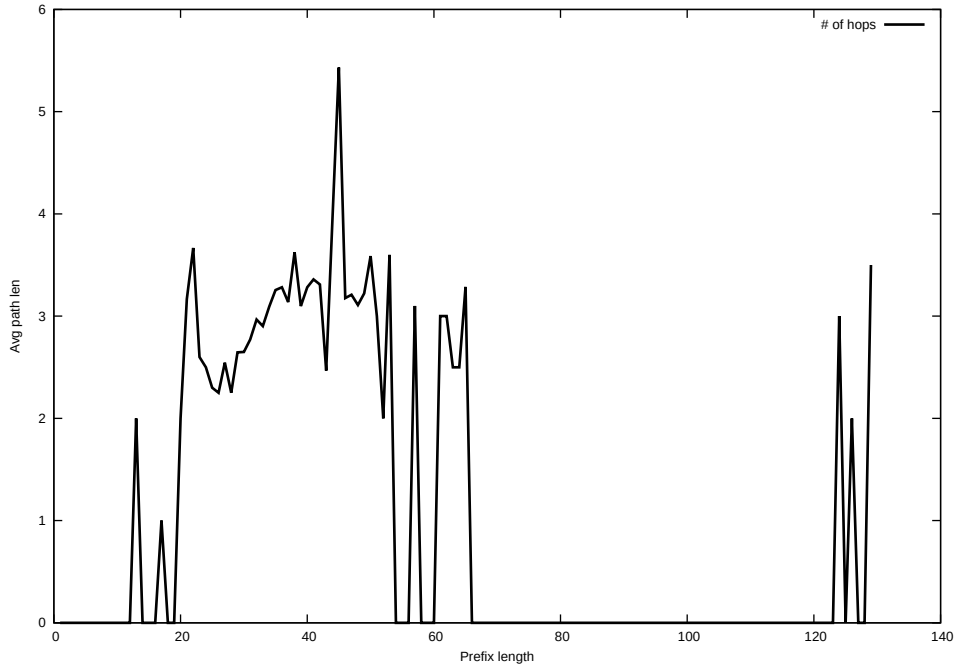
2013-06-15



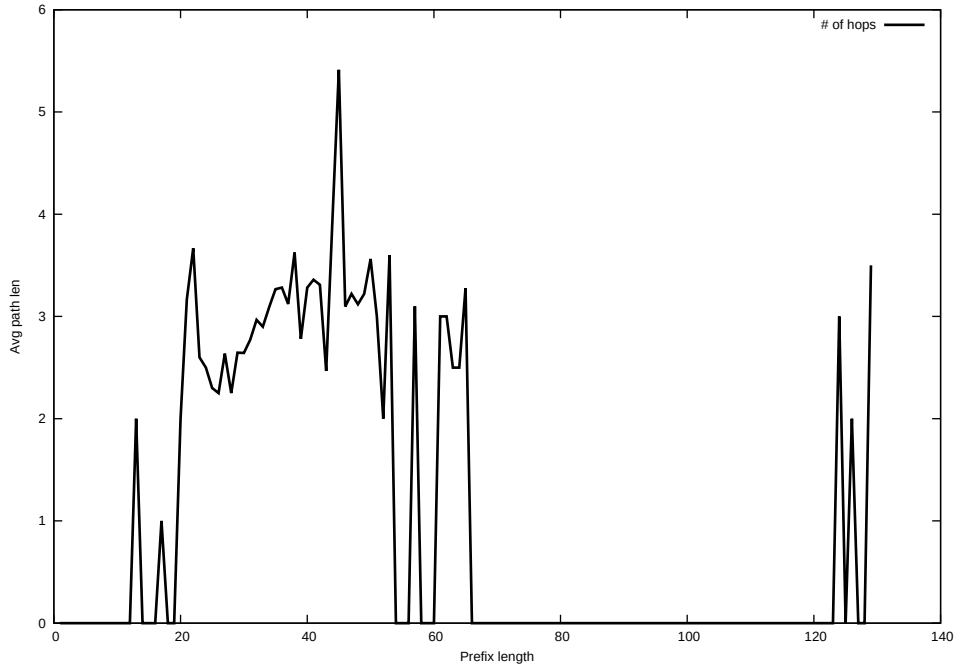
2013-06-16



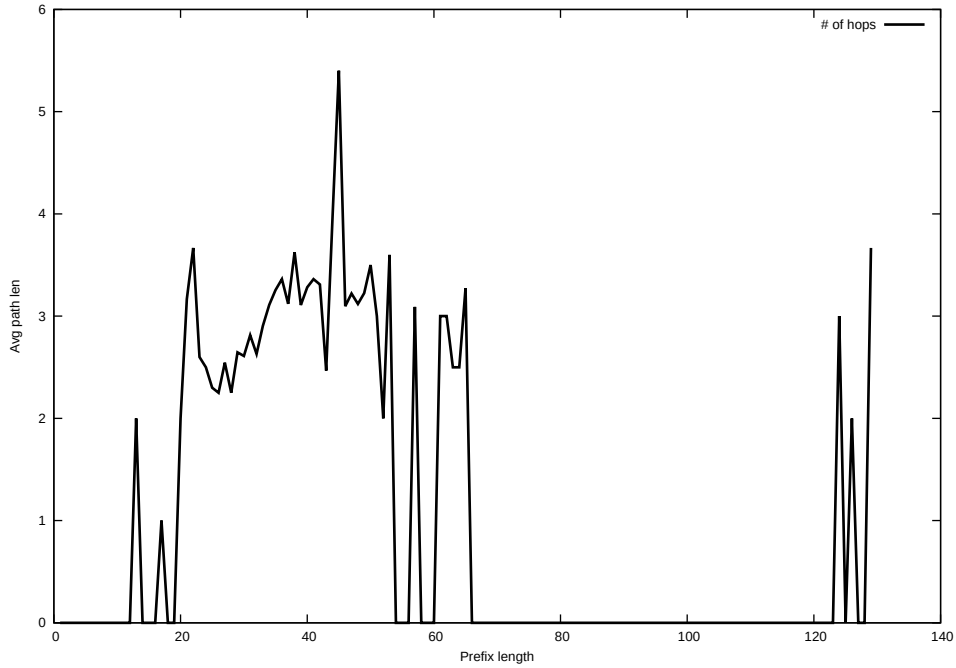
2013-06-17



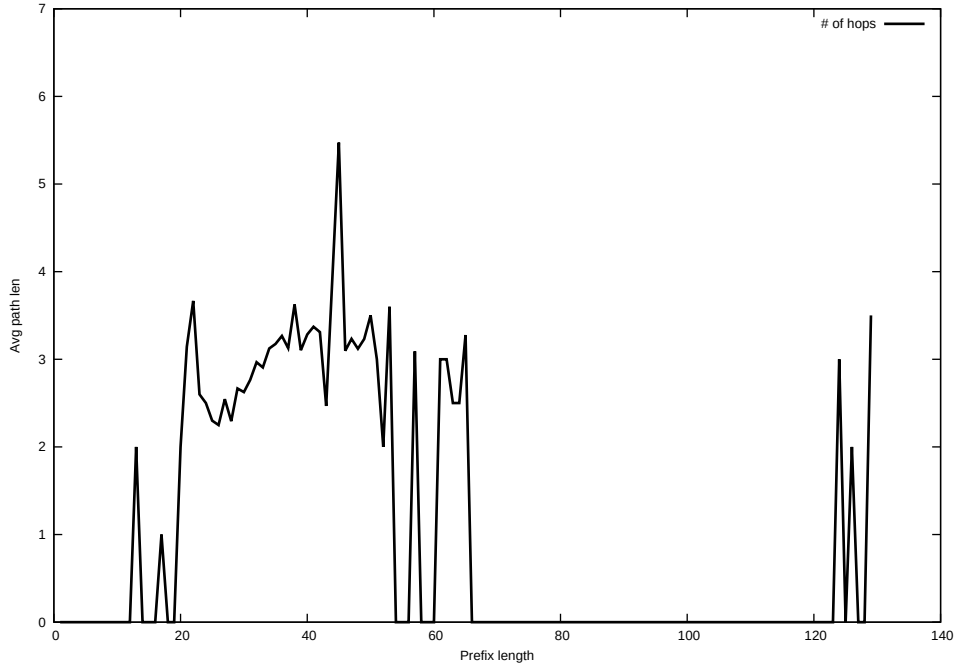
2013-06-18



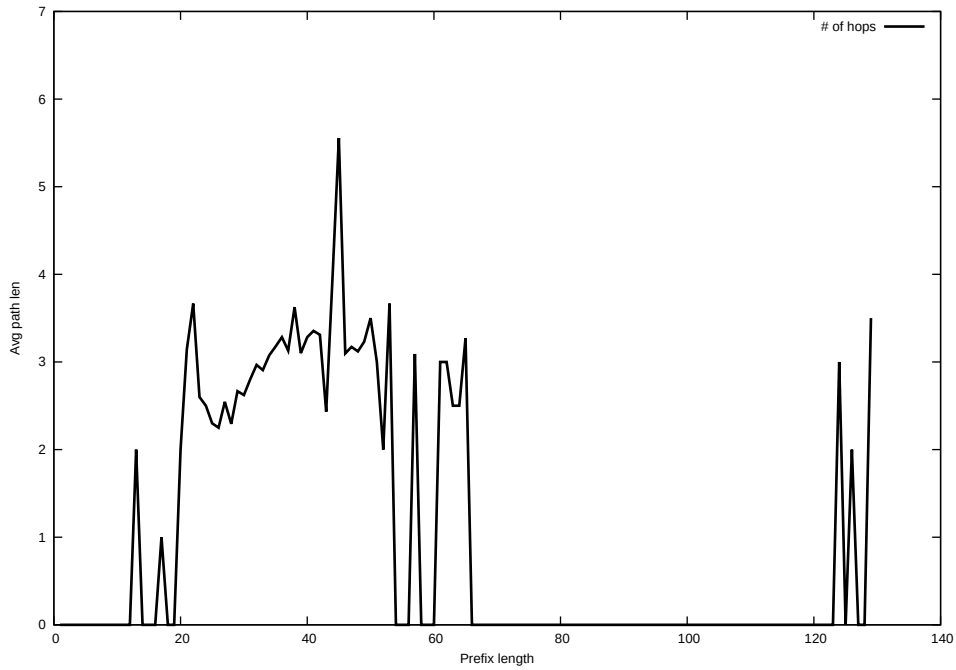
2013-06-19



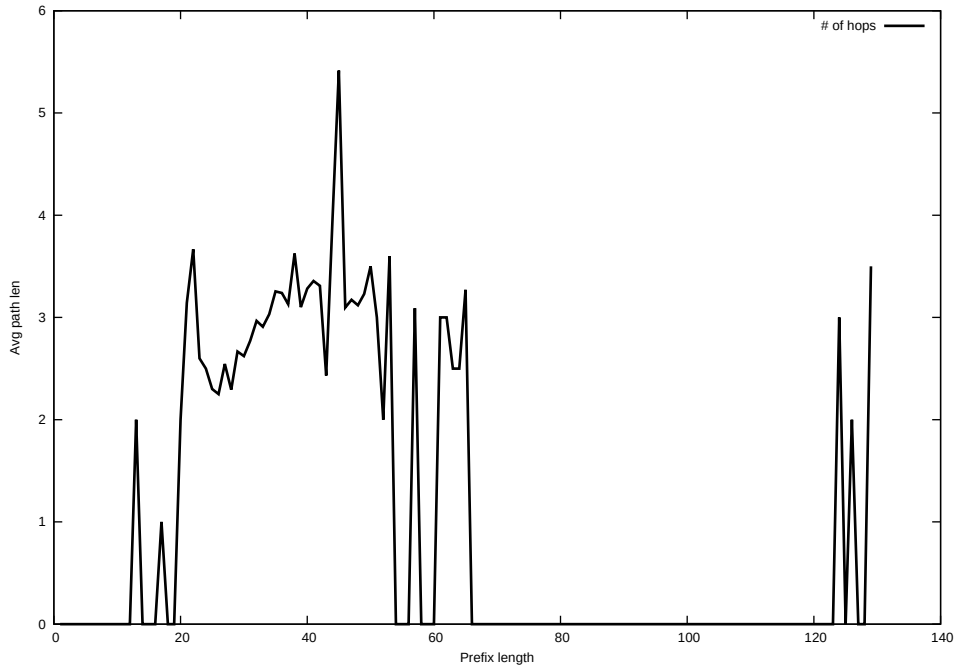
2013-06-20



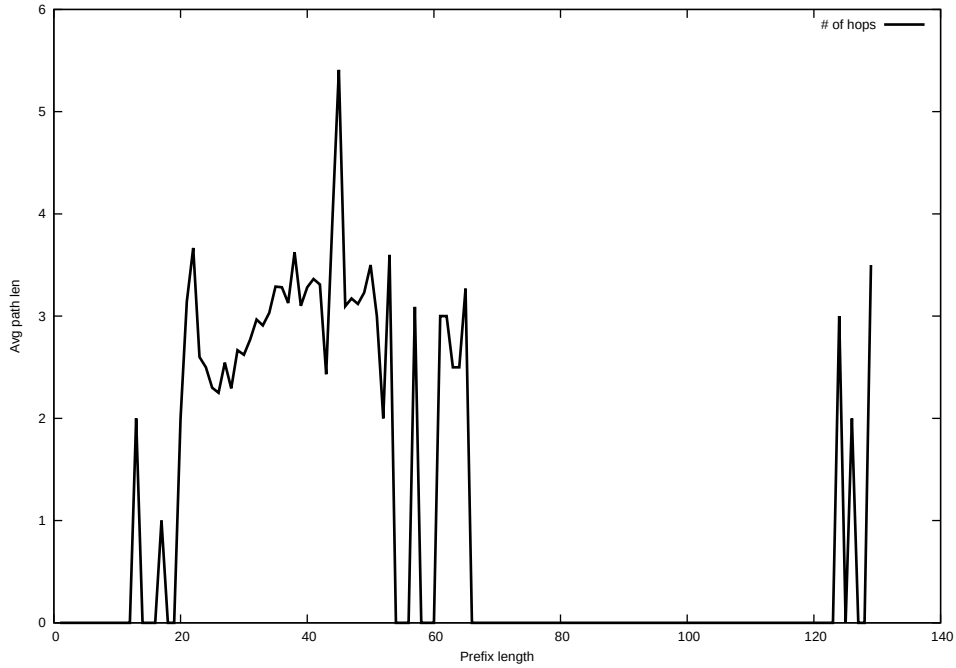
2013-06-21



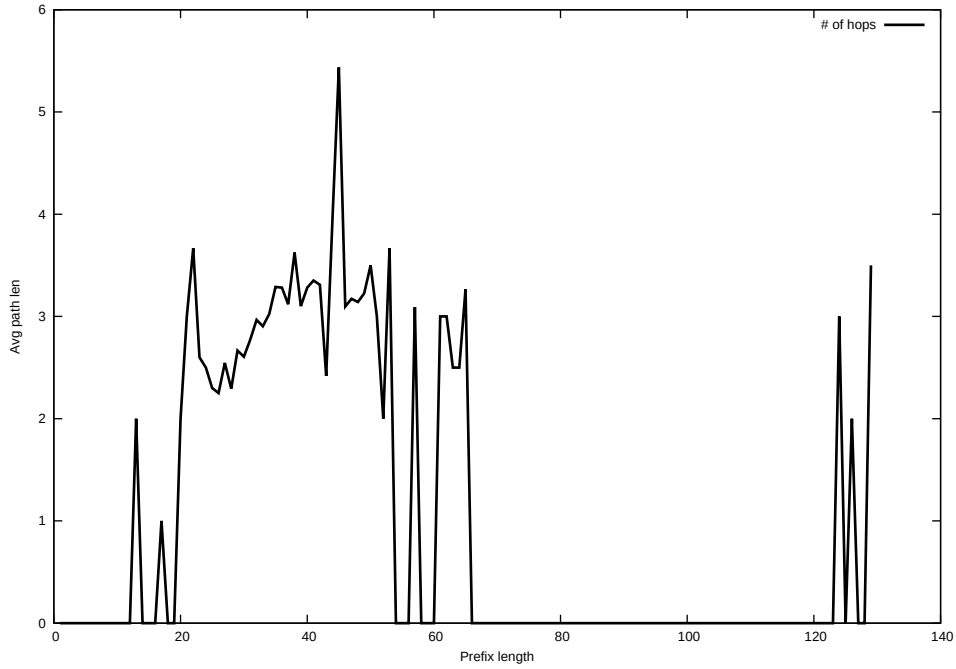
2013-06-22



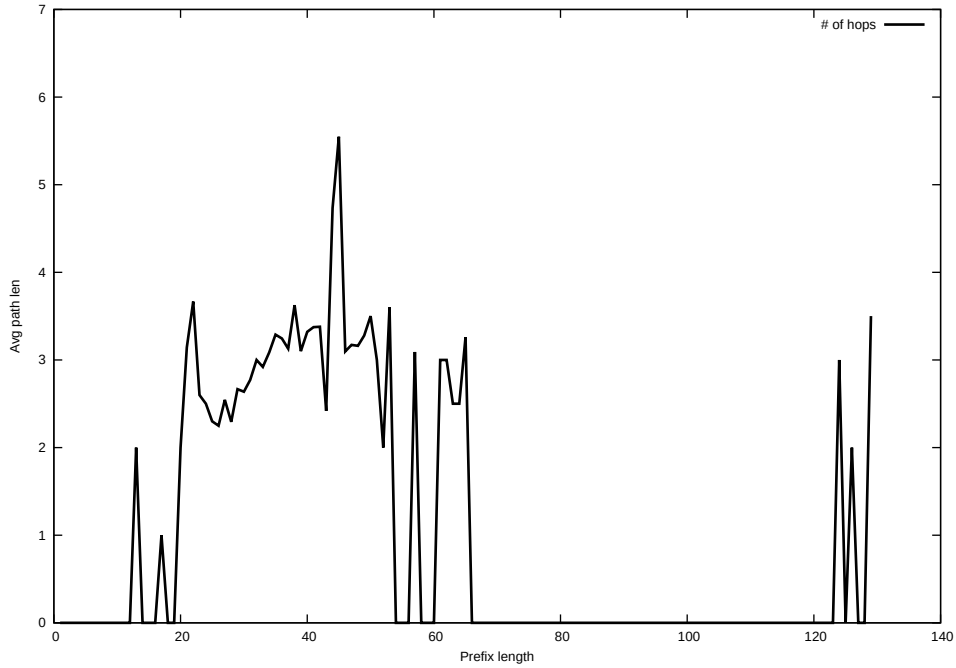
2013-06-23



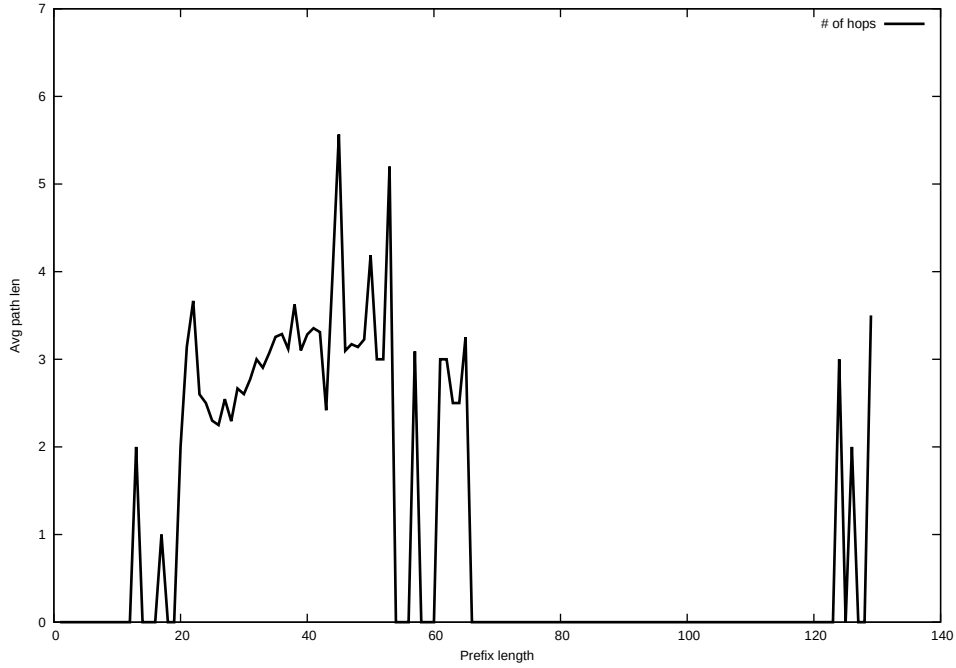
2013-06-24



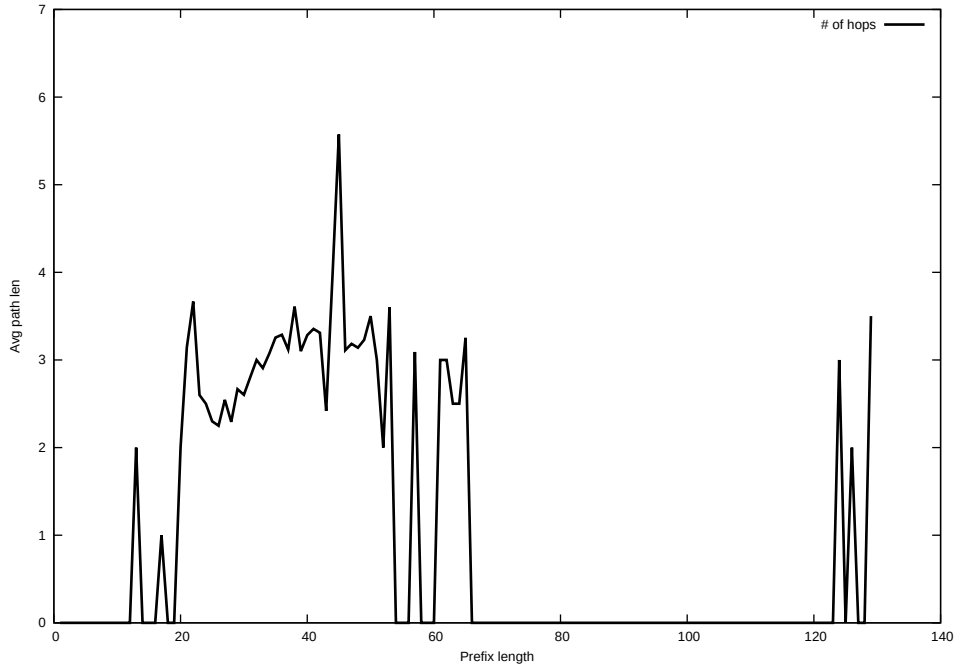
2013-06-25



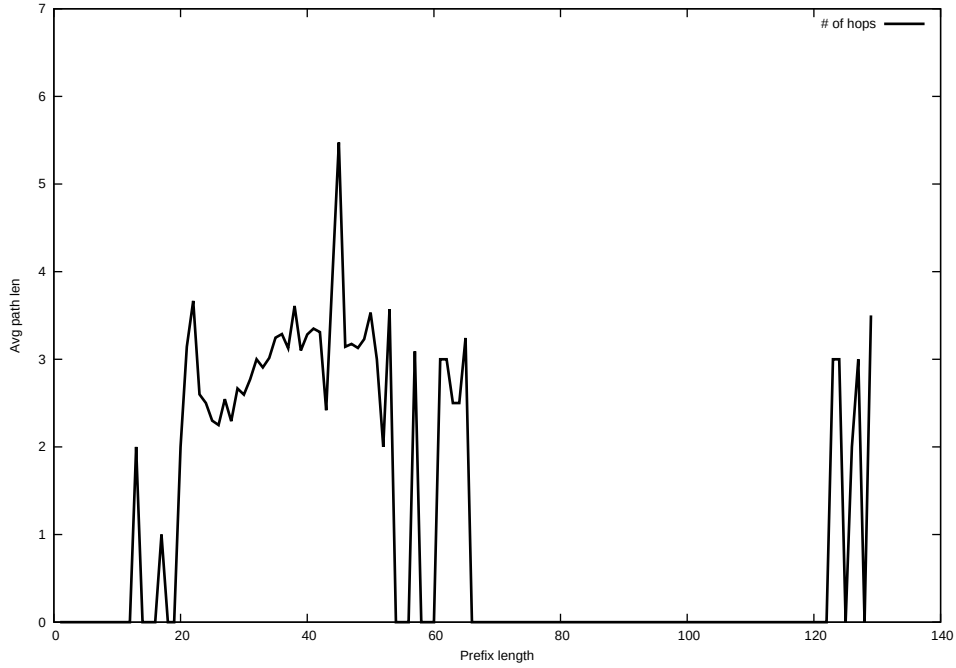
2013-06-26



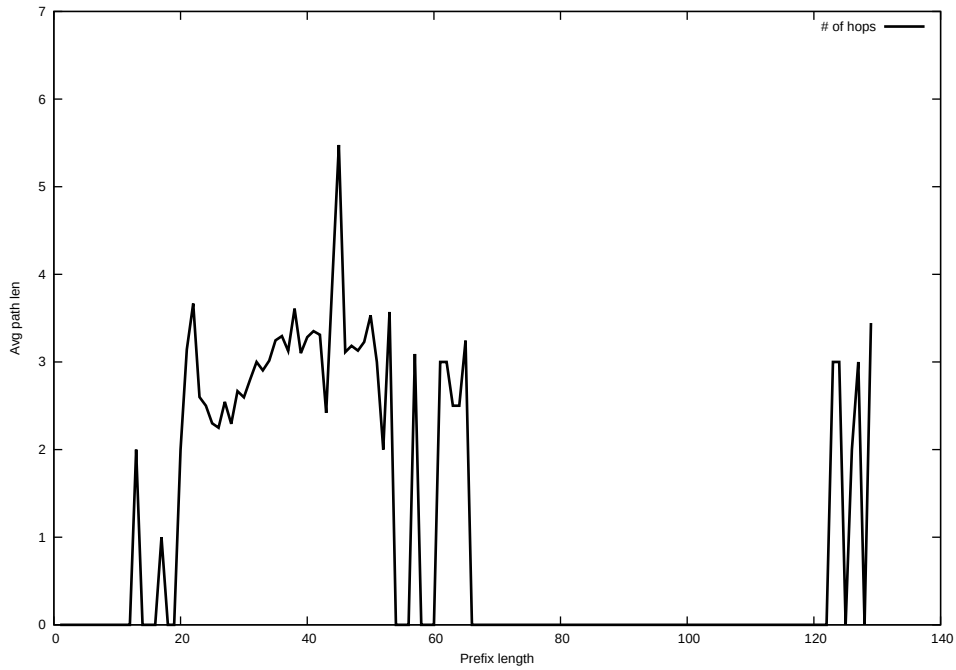
2013-06-27



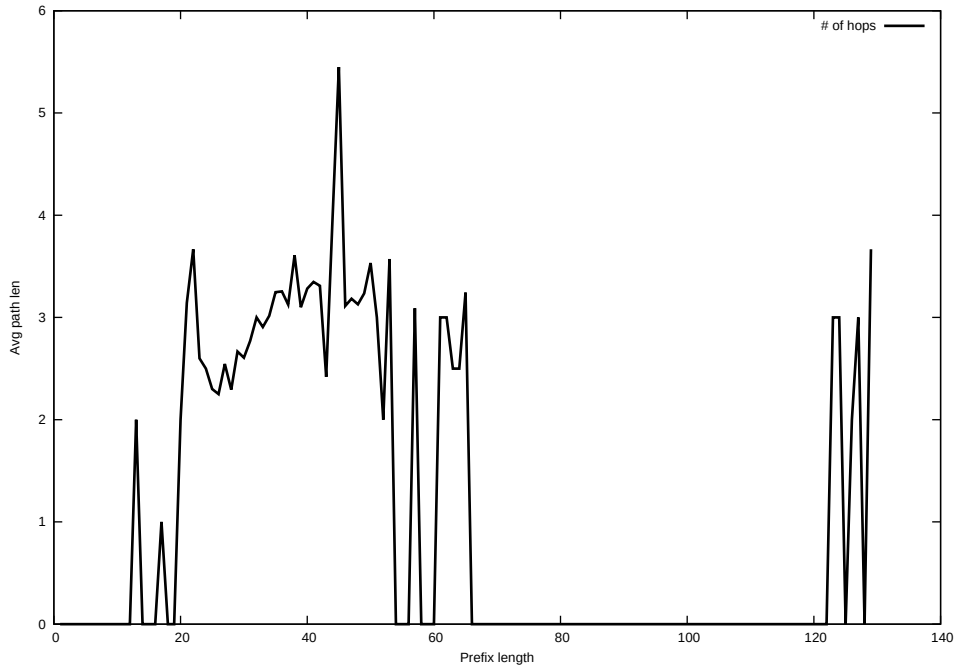
2013-06-28



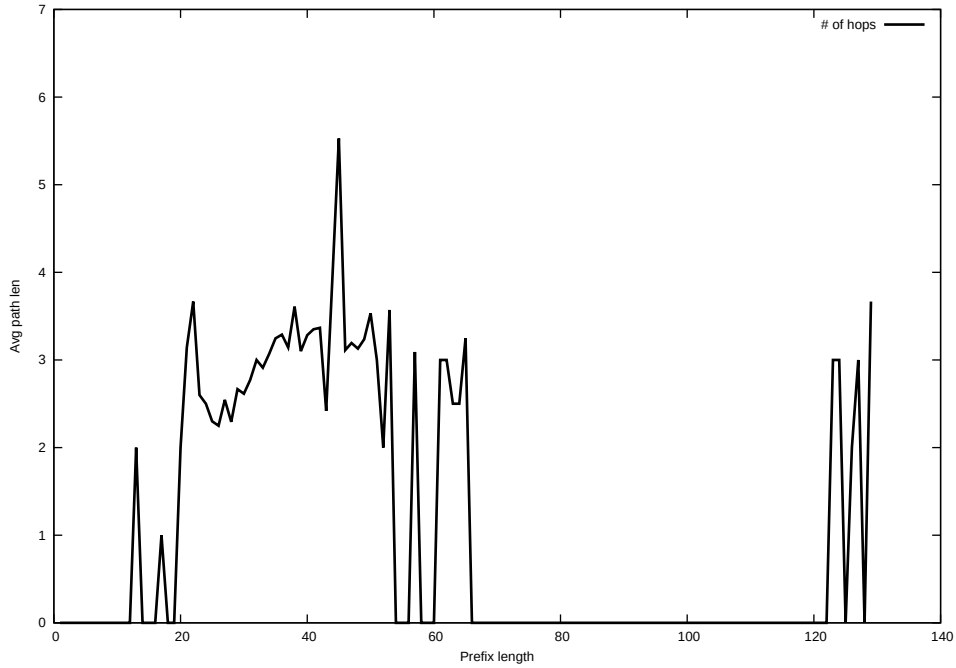
2013-06-29



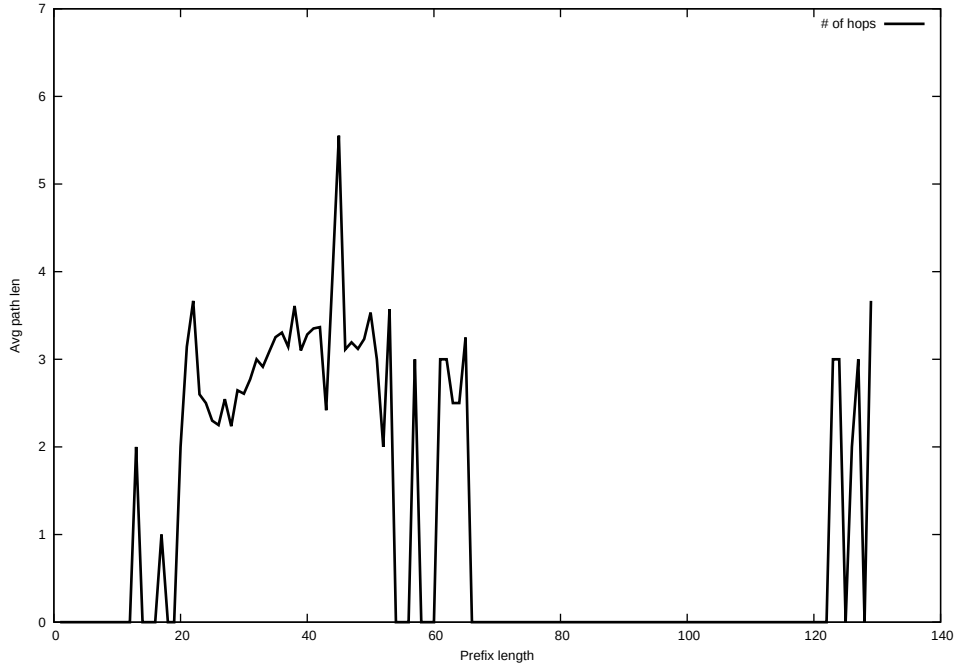
2013-06-30



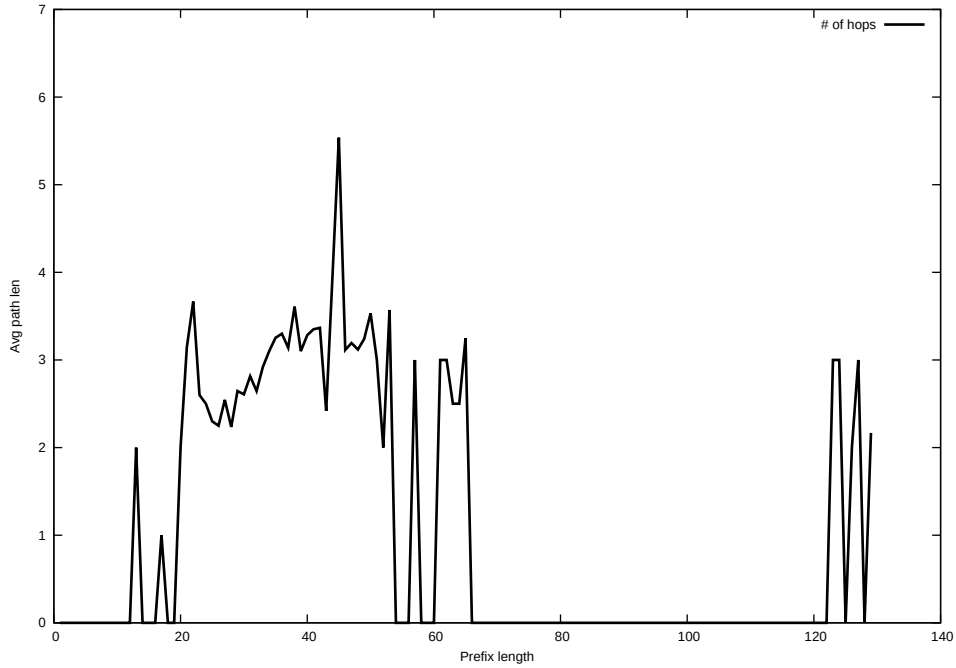
2013-07-01



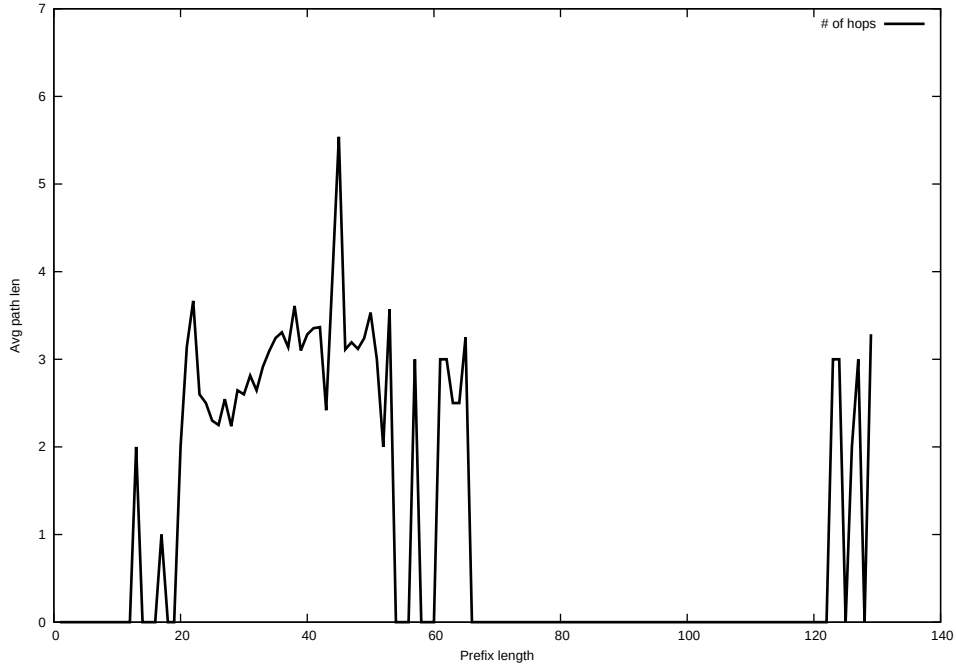
2013-07-02



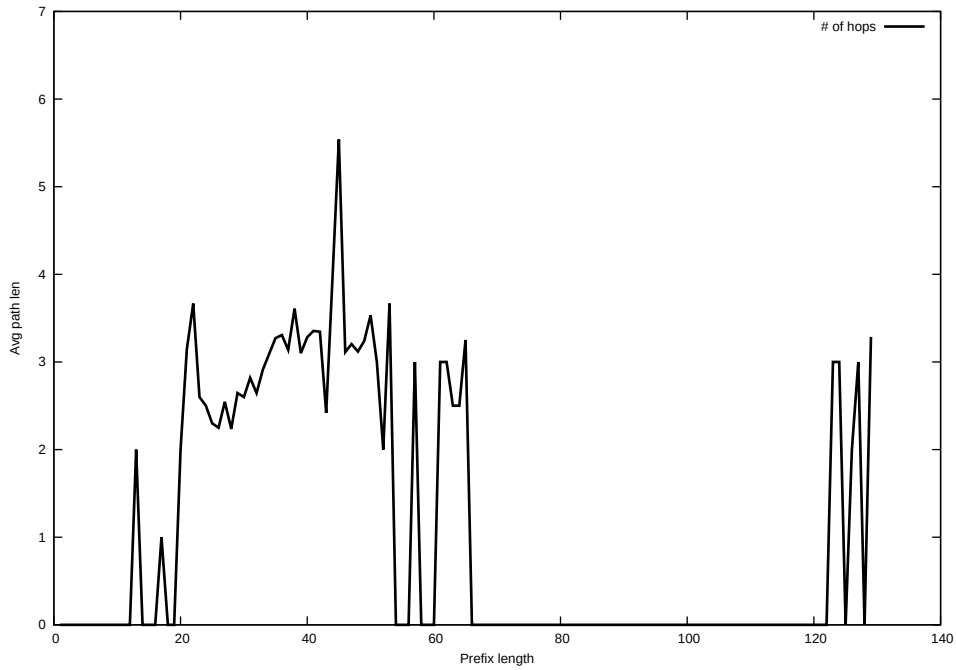
2013-07-03



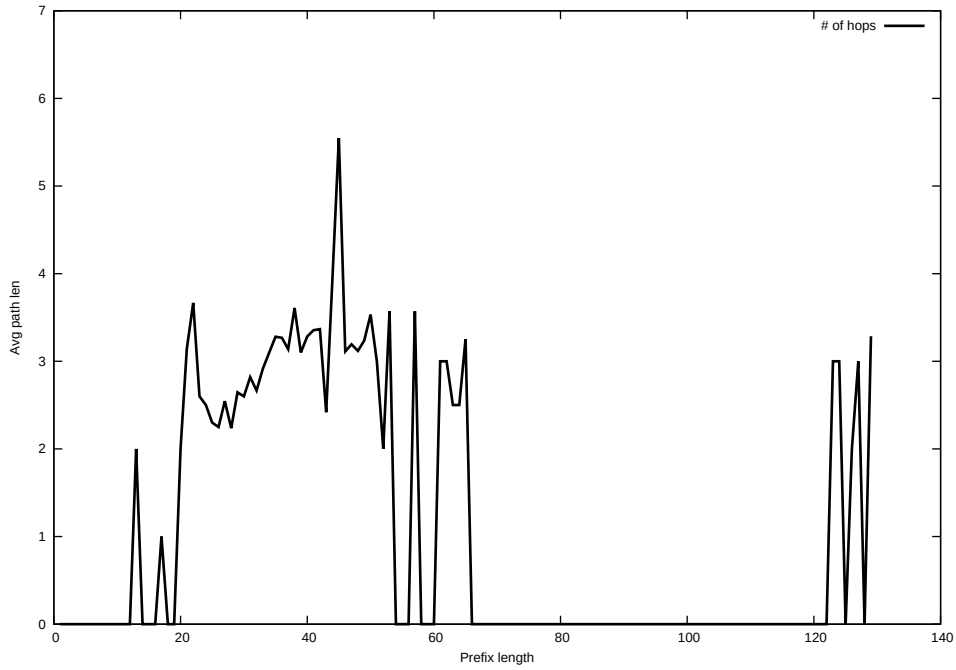
2013-07-04



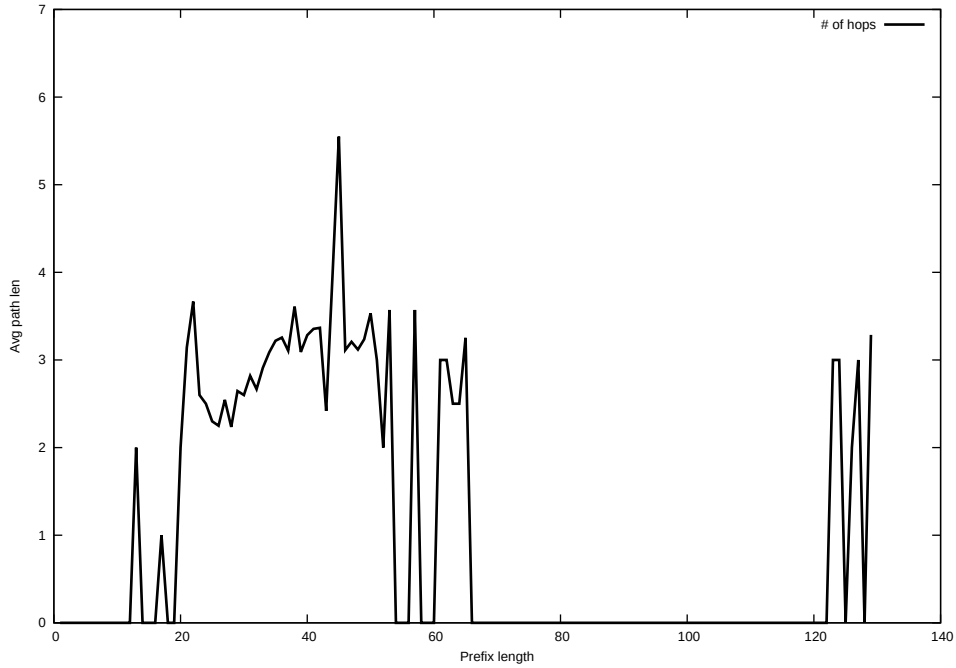
2013-07-05



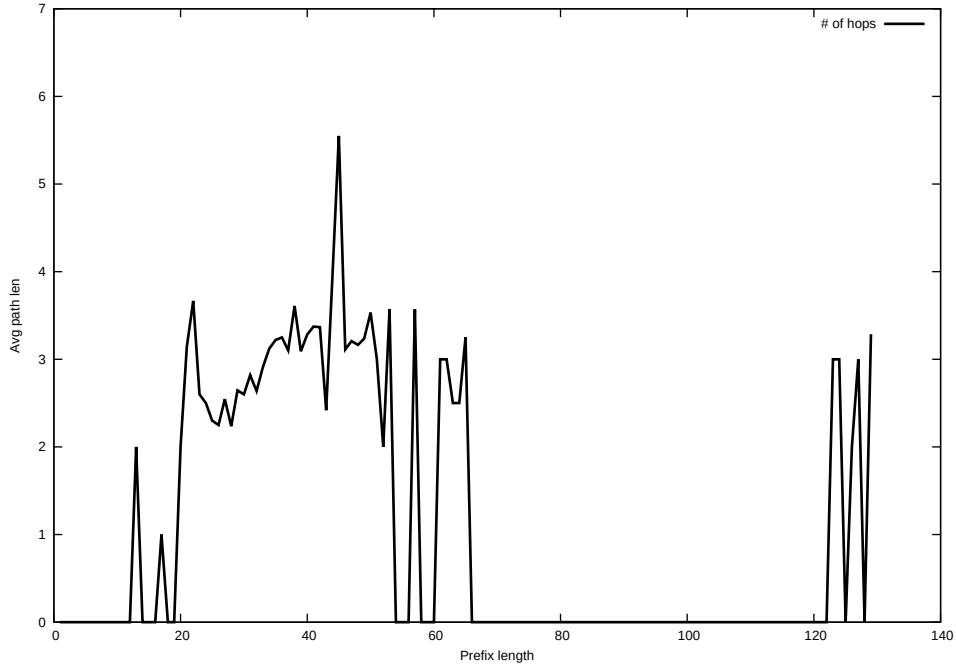
2013-07-06



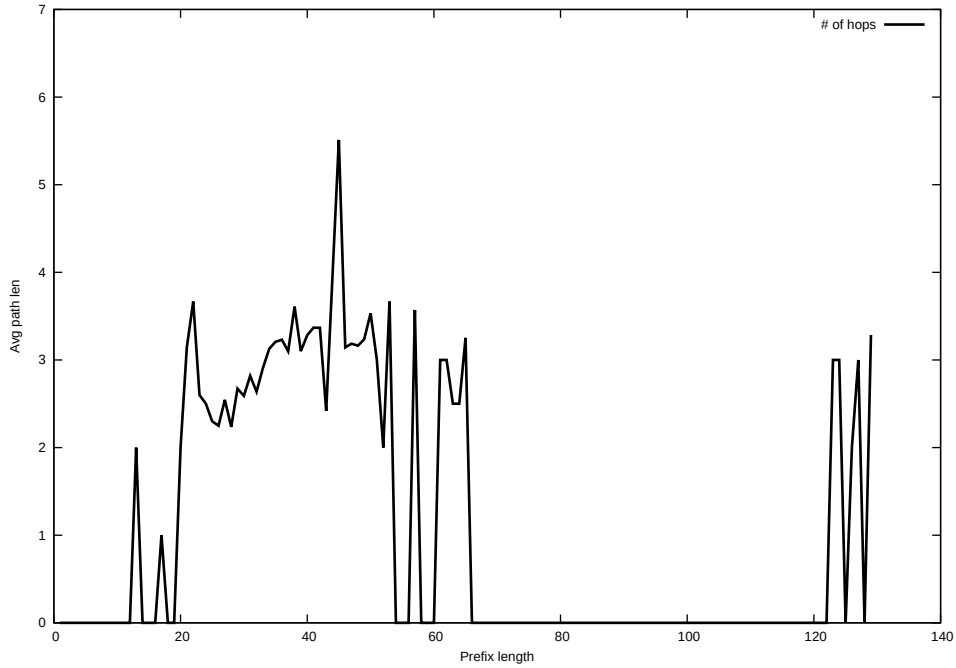
2013-07-07



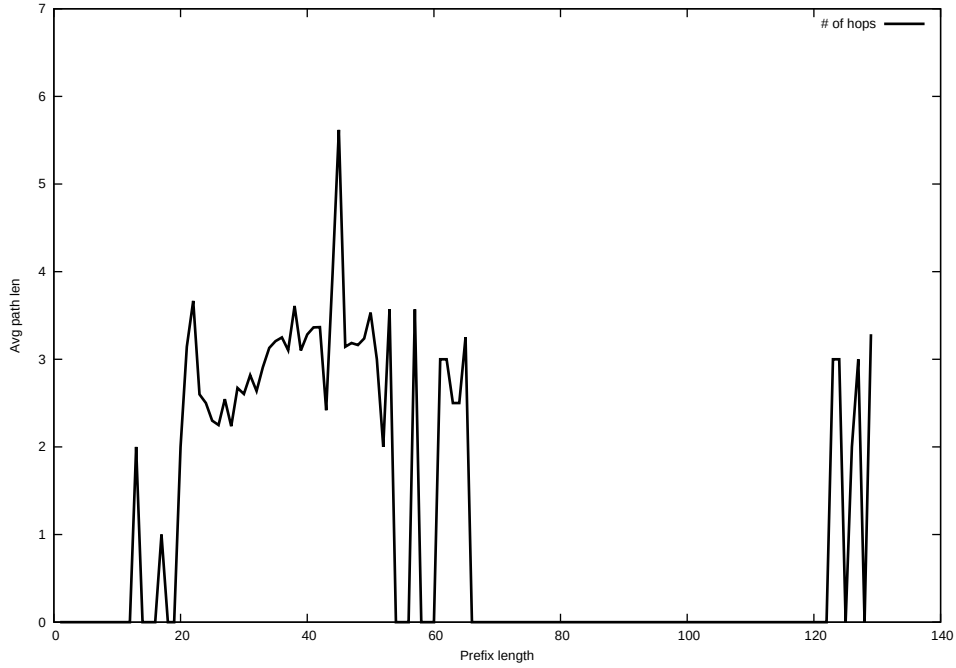
2013-07-08



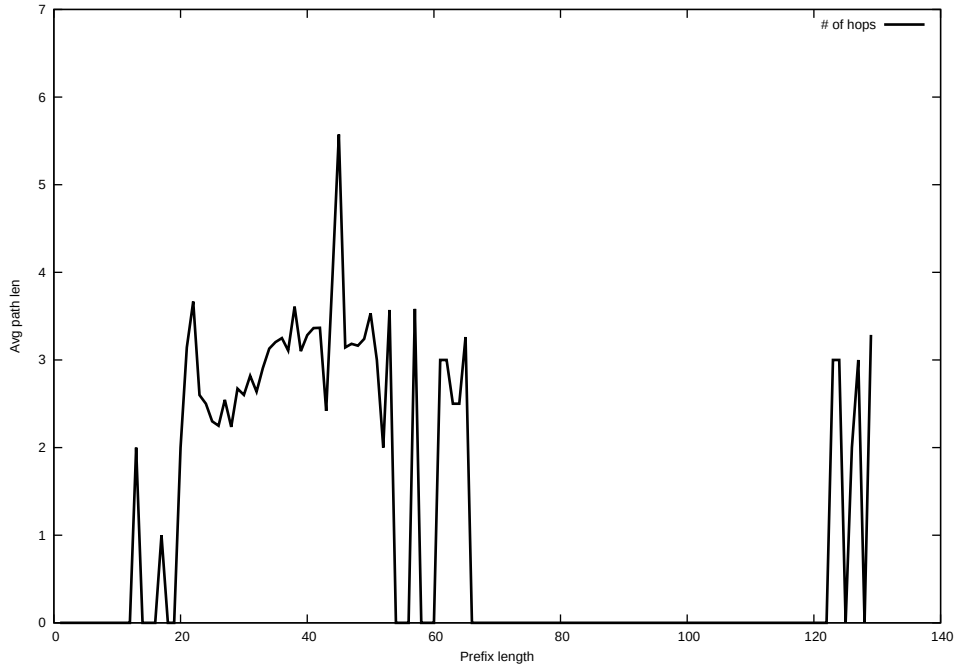
2013-07-09



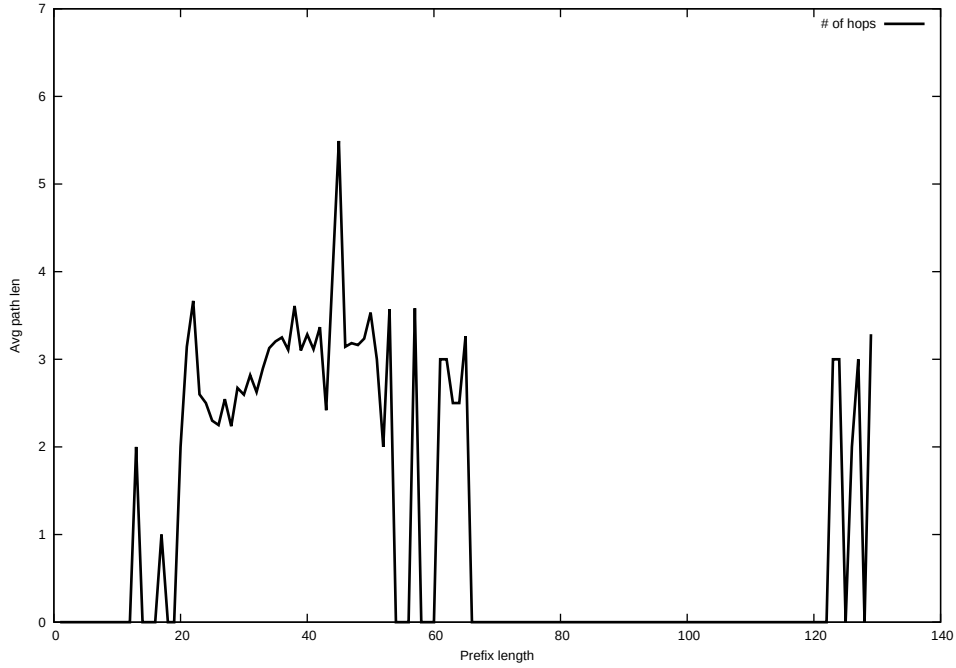
2013-07-10



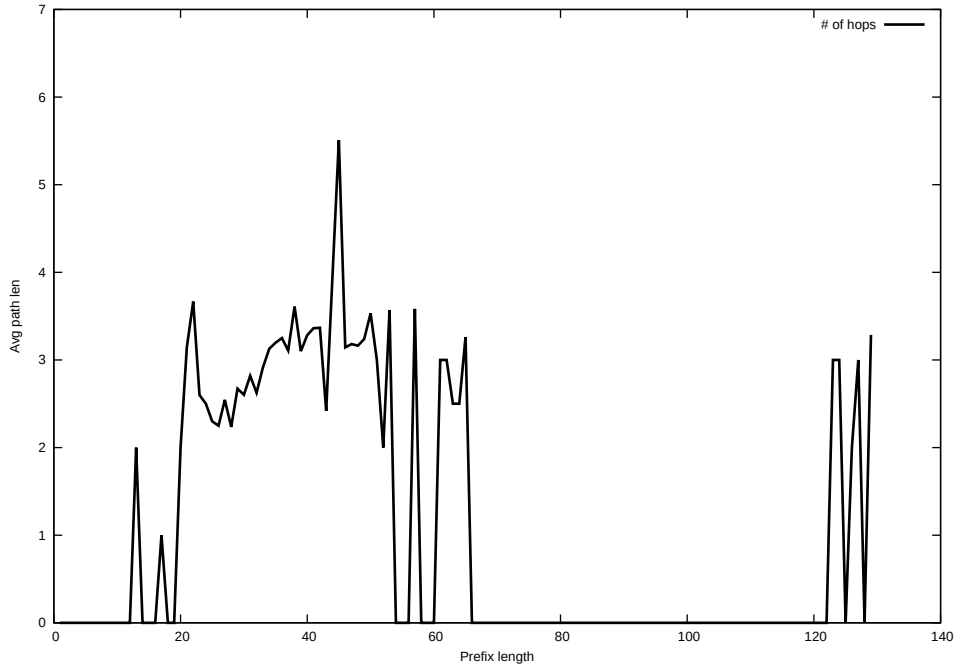
2013-07-11



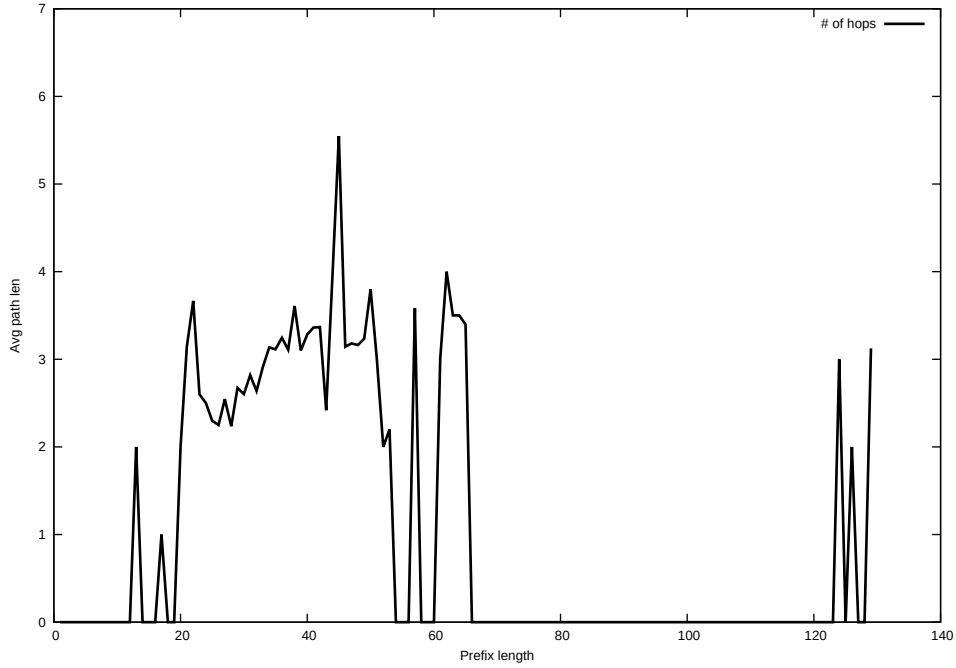
2013-07-12



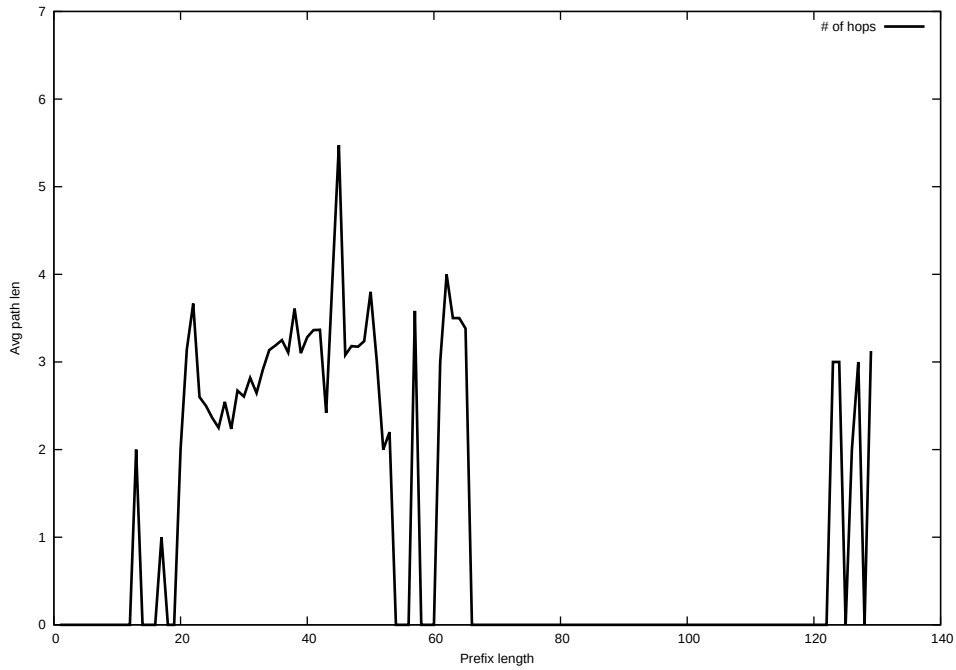
2013-07-13



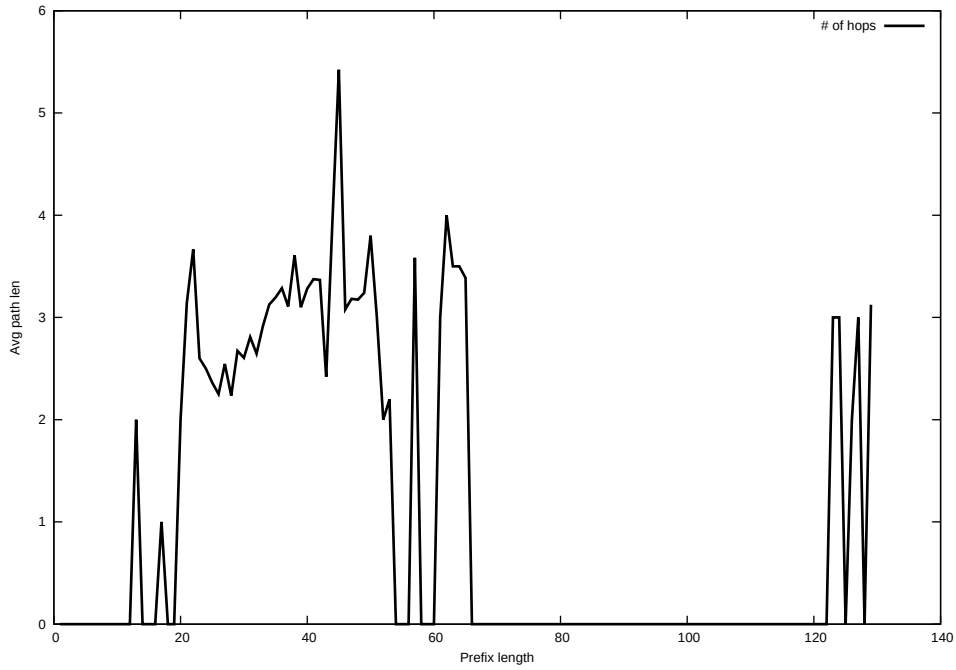
2013-07-14



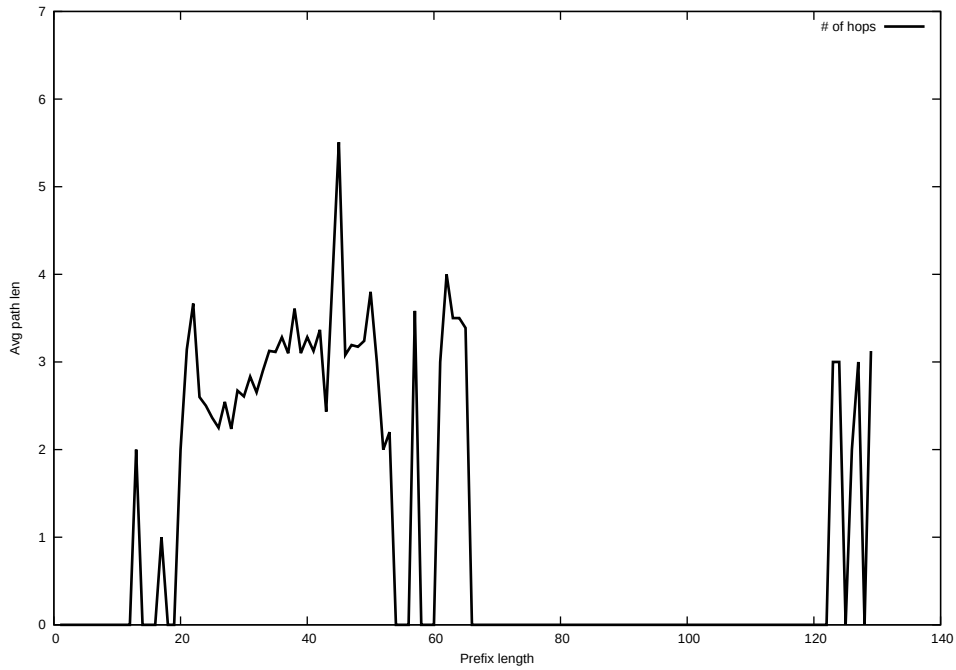
2013-07-15



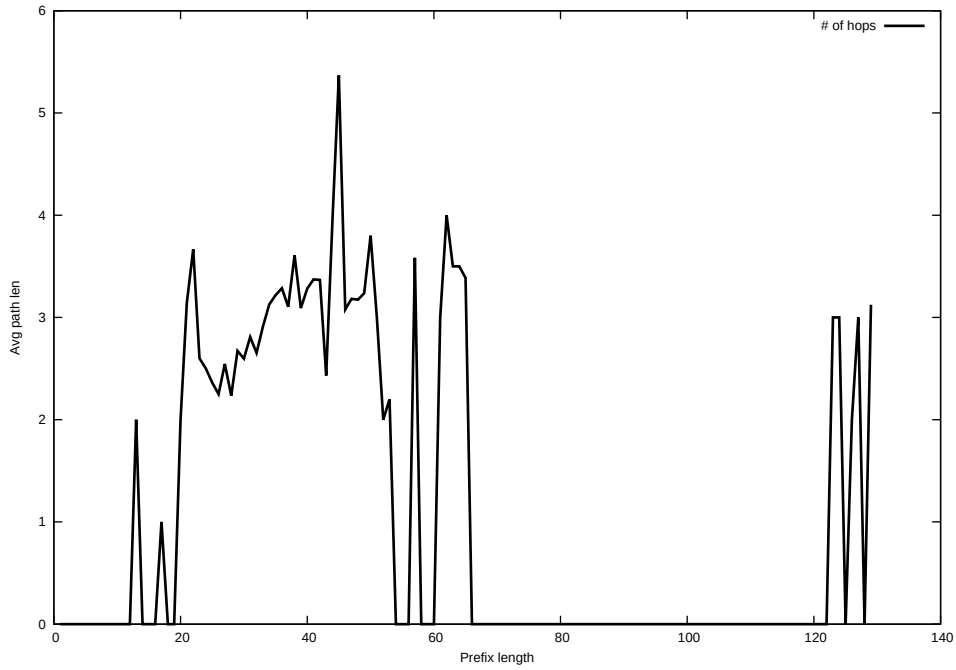
2013-07-16



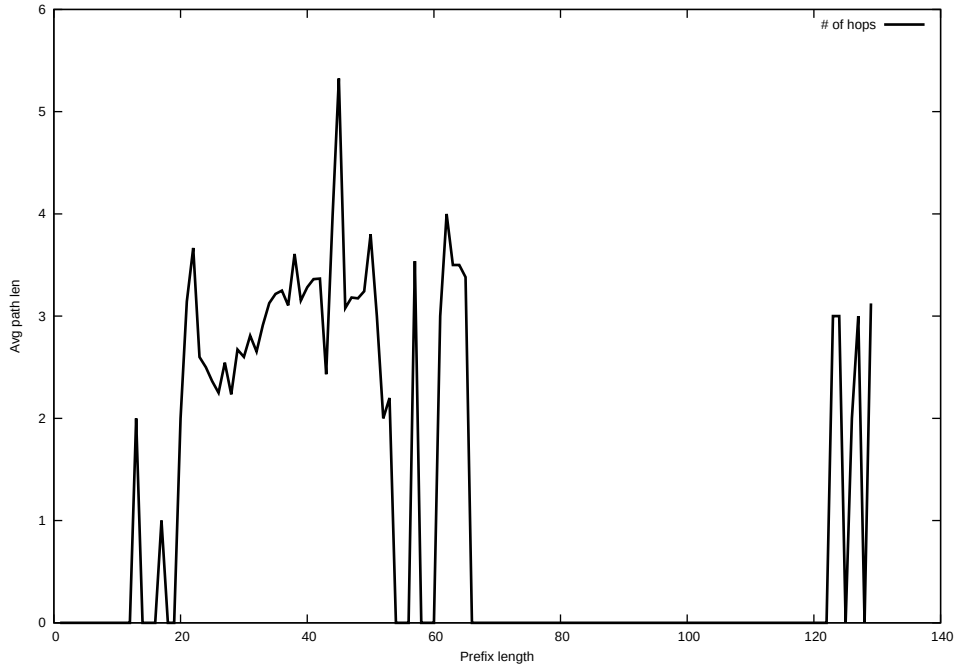
2013-07-17



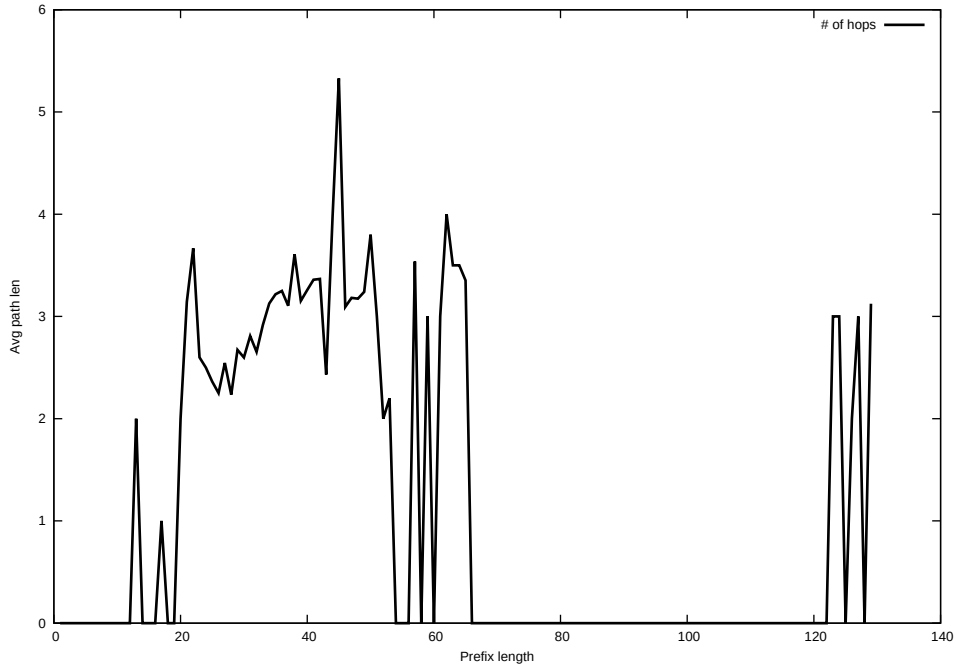
2013-07-18



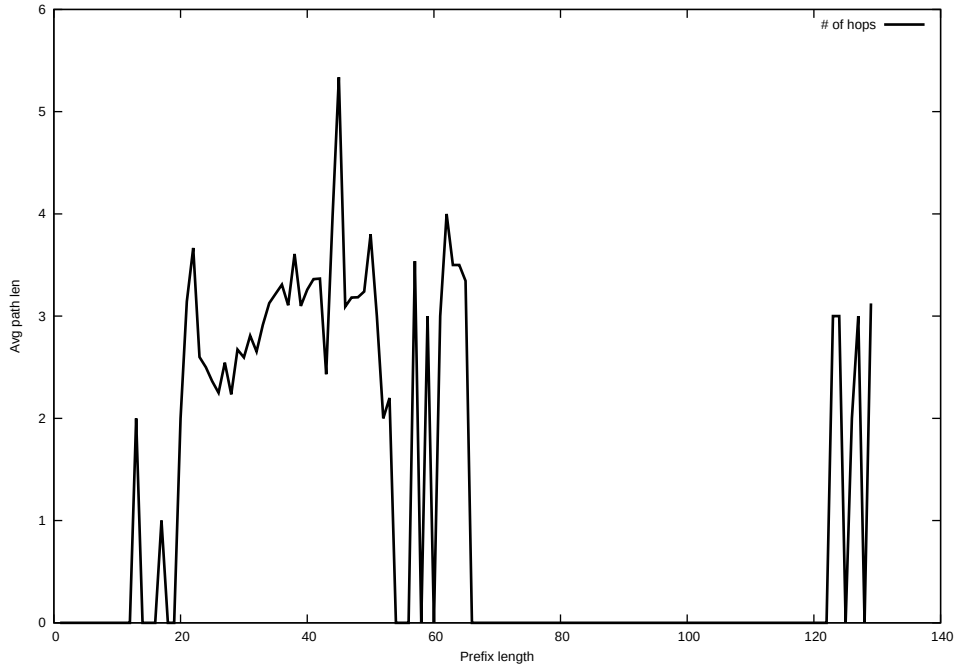
2013-07-19



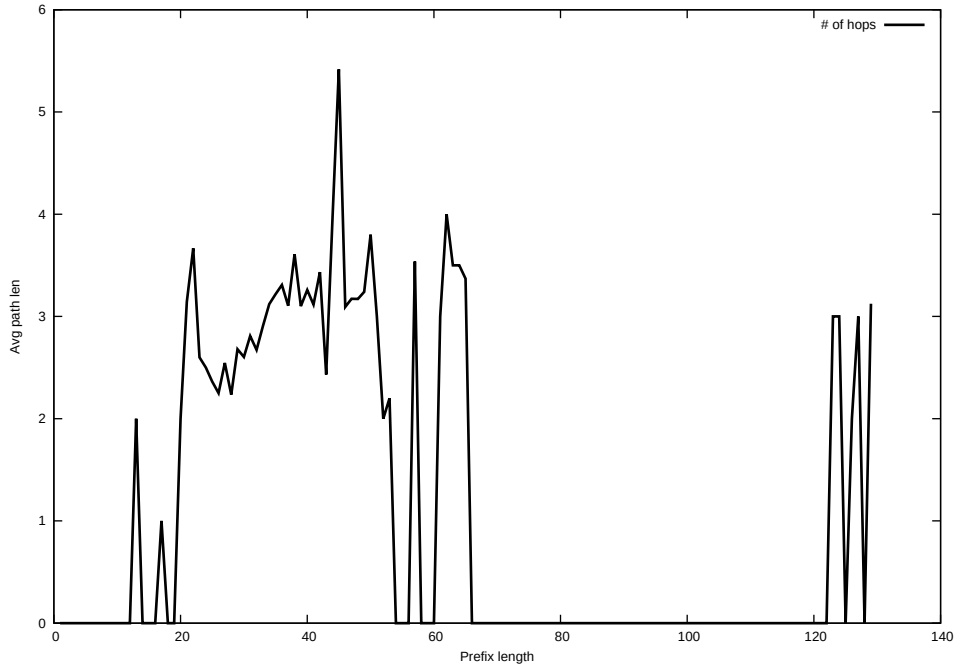
2013-07-20



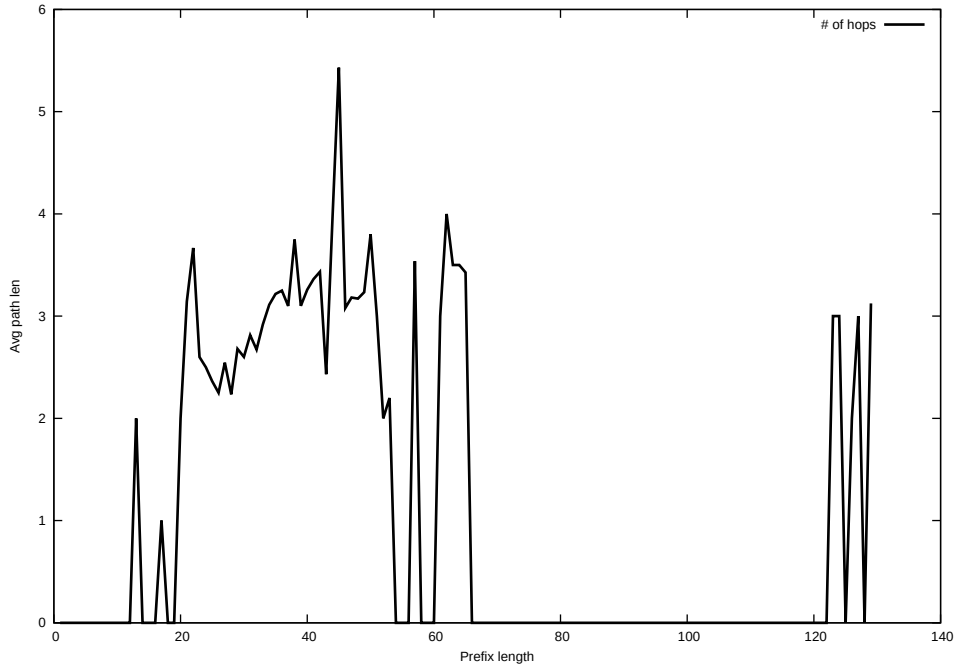
2013-07-21



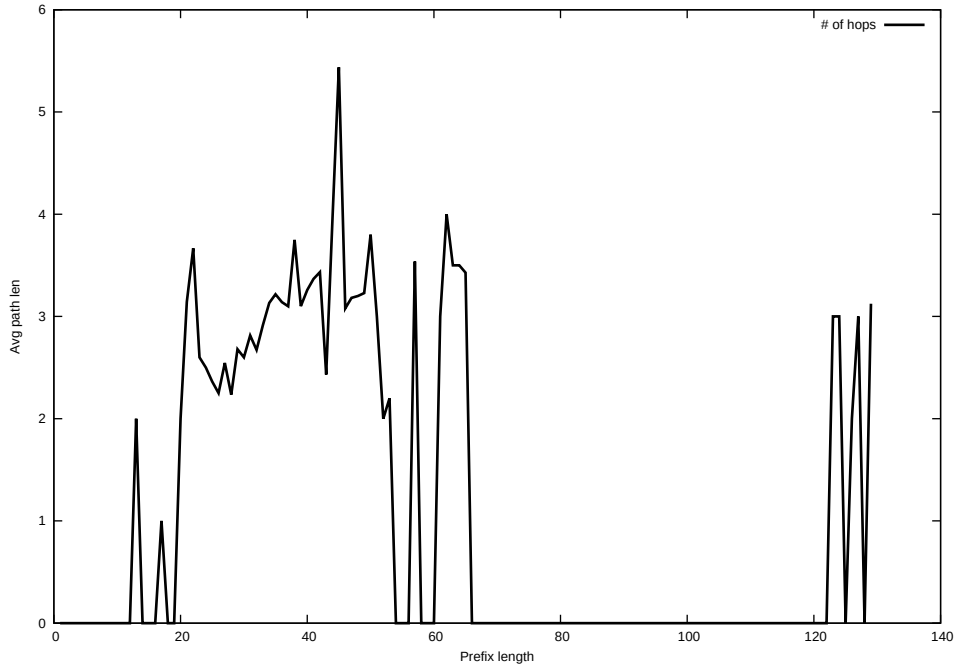
2013-07-22



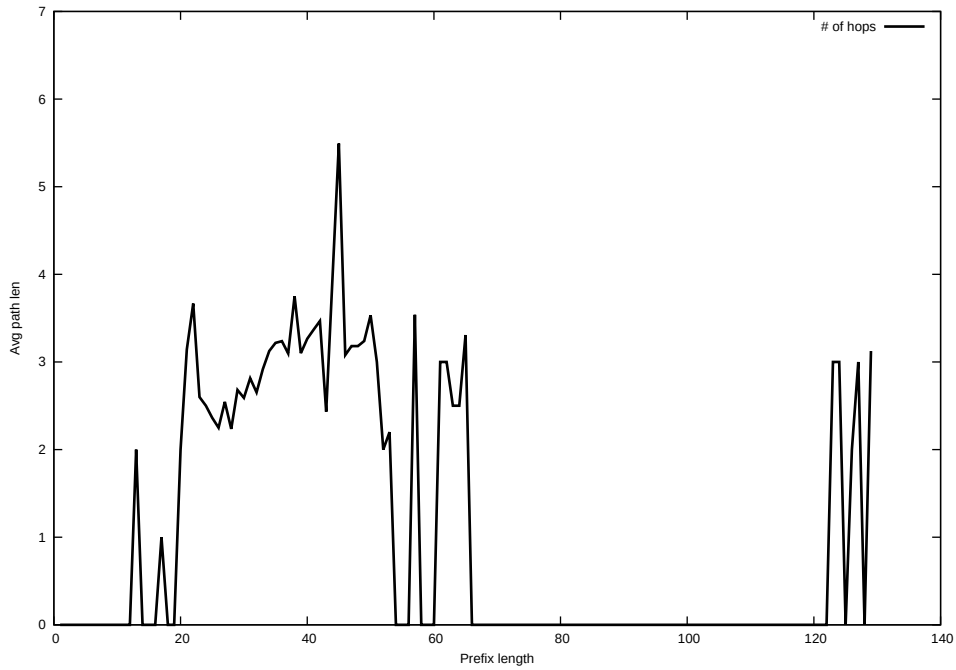
2013-07-23



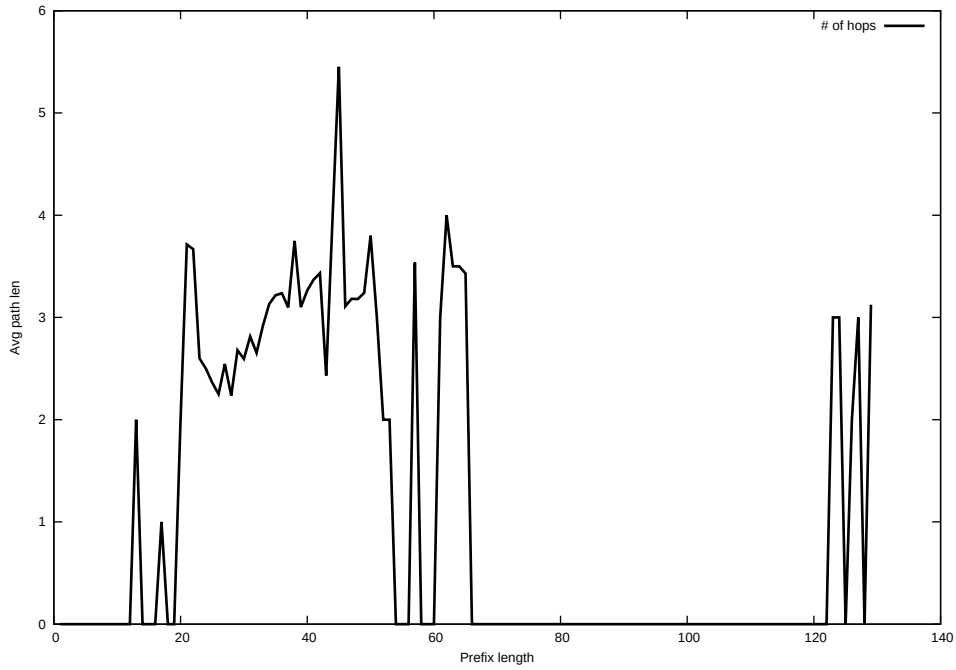
2013-07-24



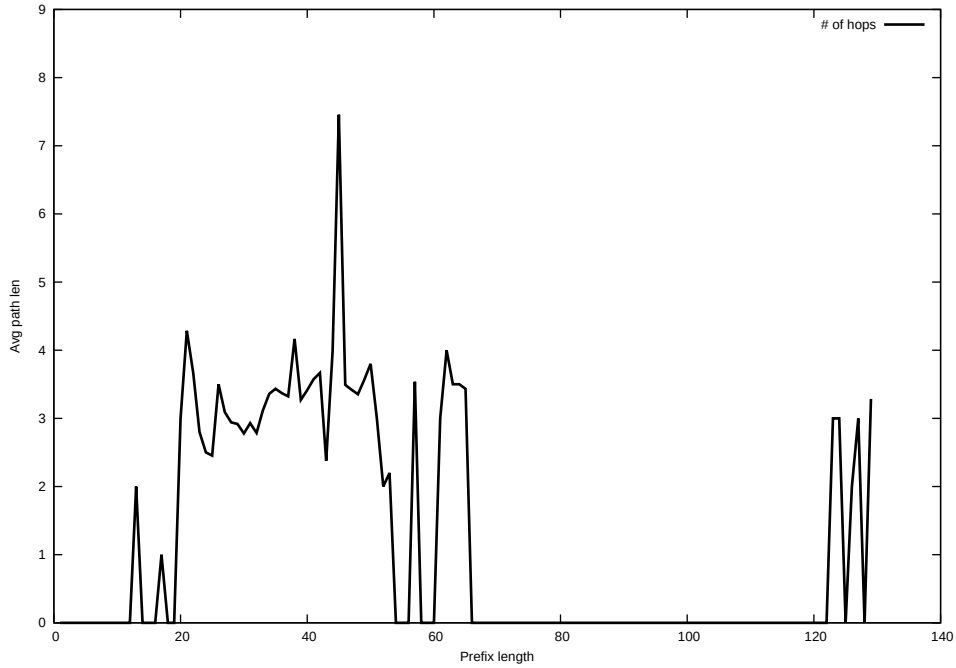
2013-07-25



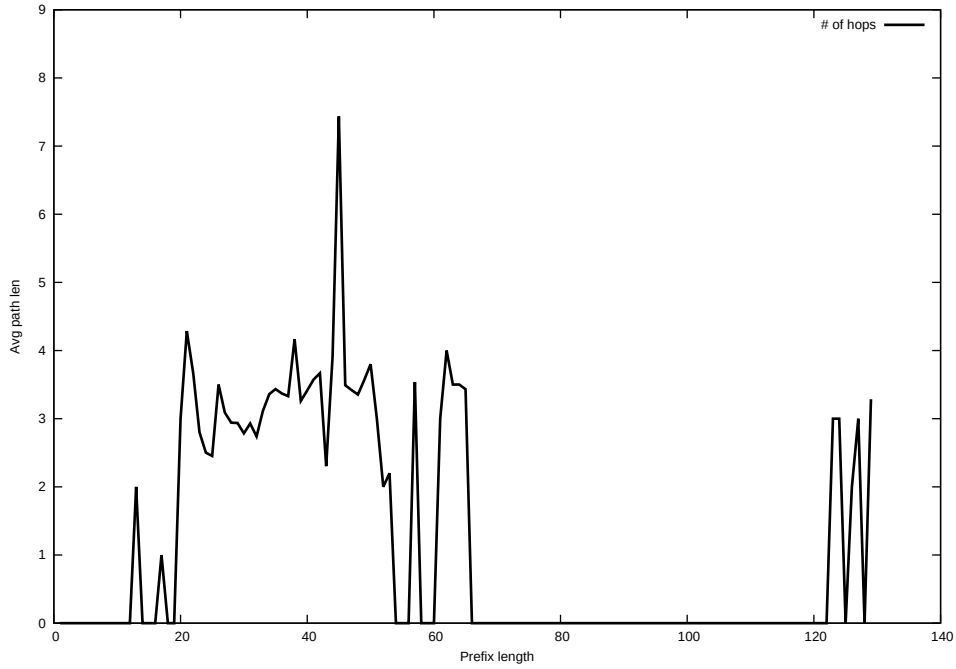
2013-07-26



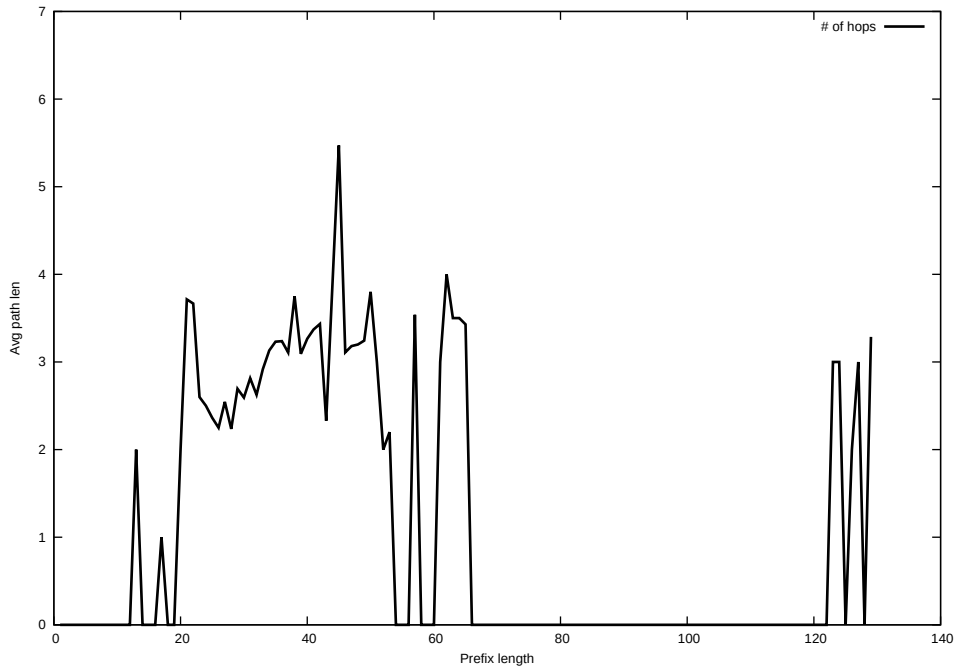
2013-07-27



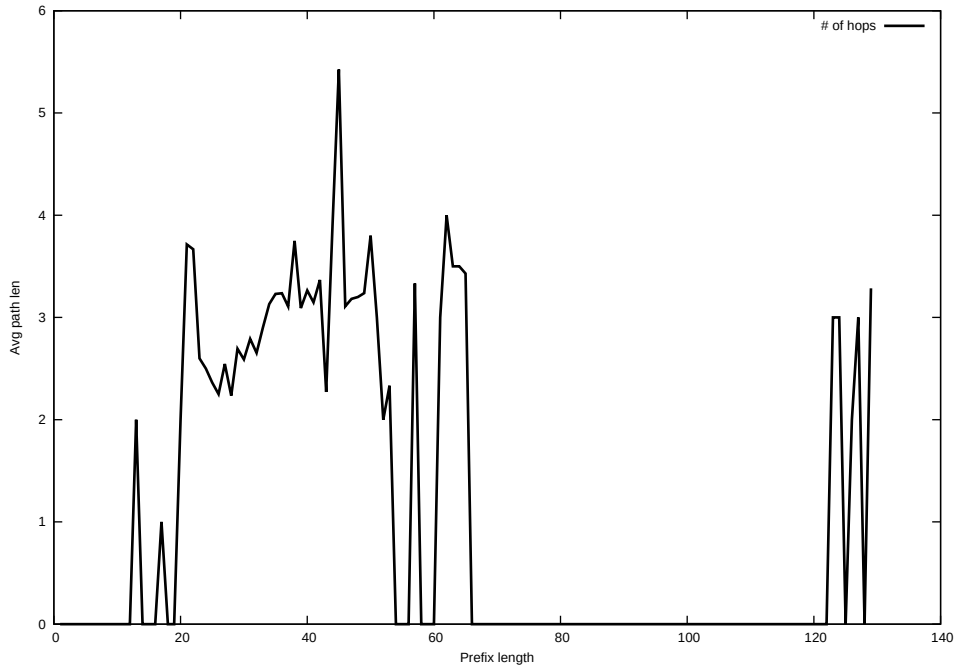
2013-07-28



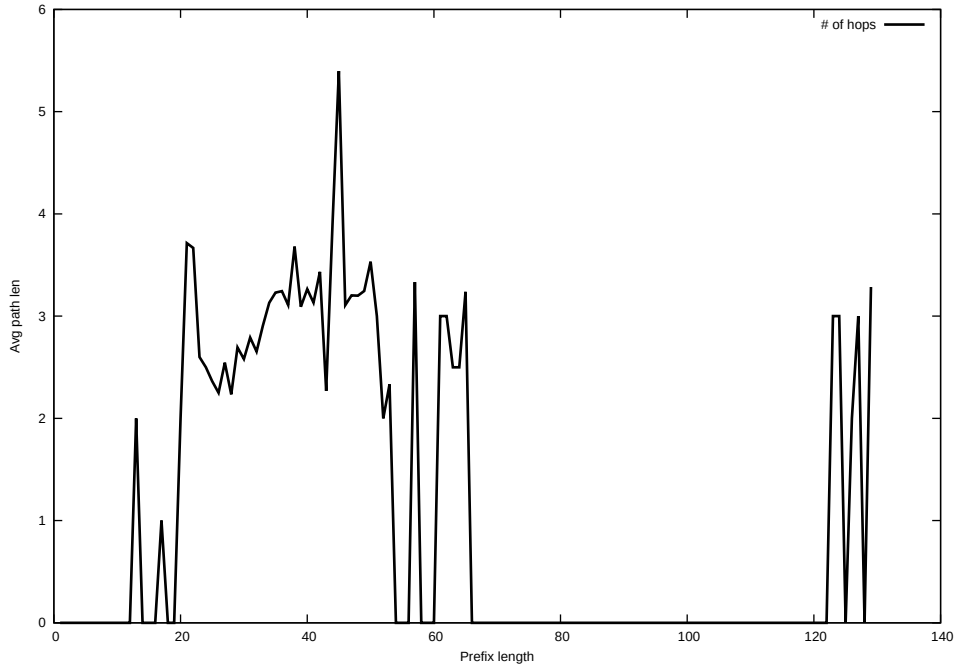
2013-07-29



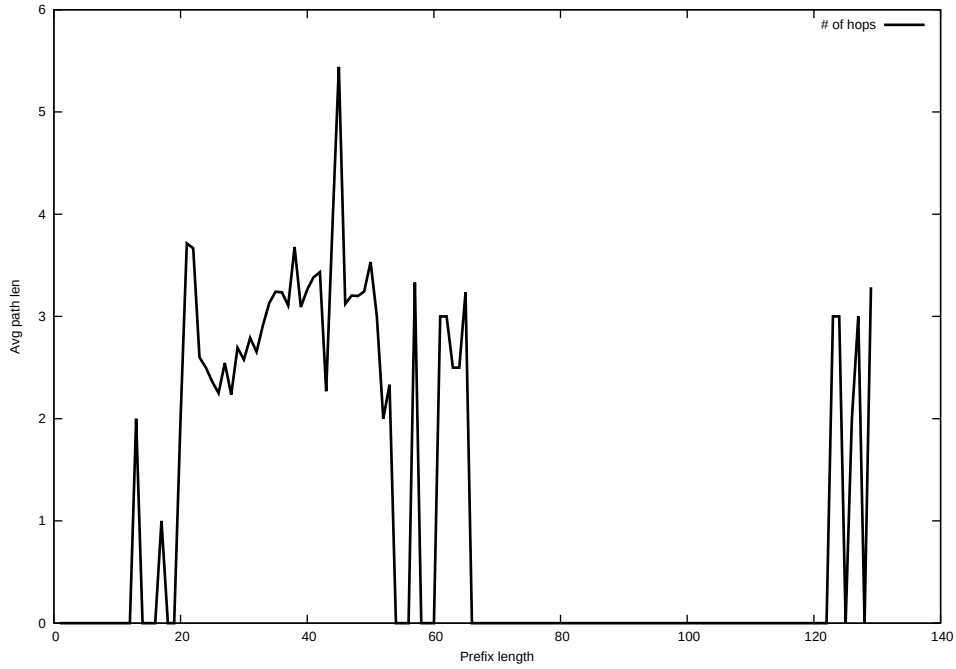
2013-07-30



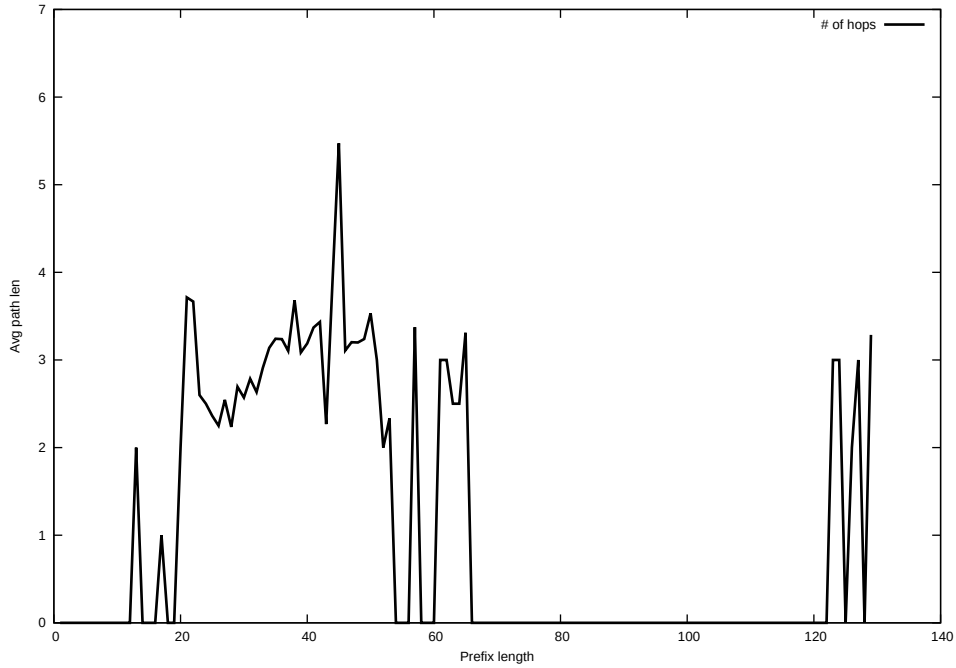
2013-07-31



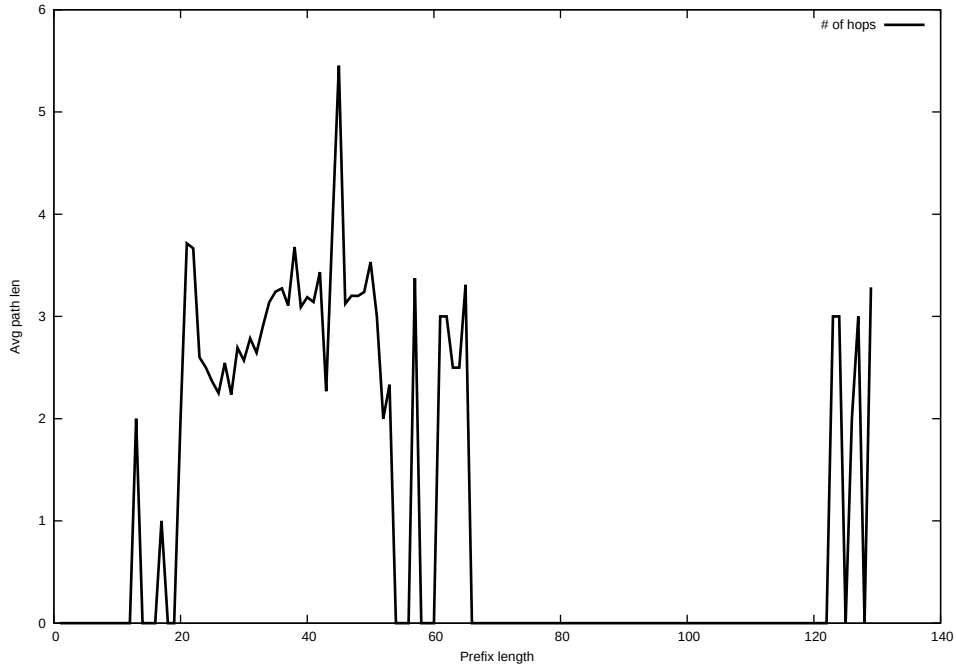
2013-08-01



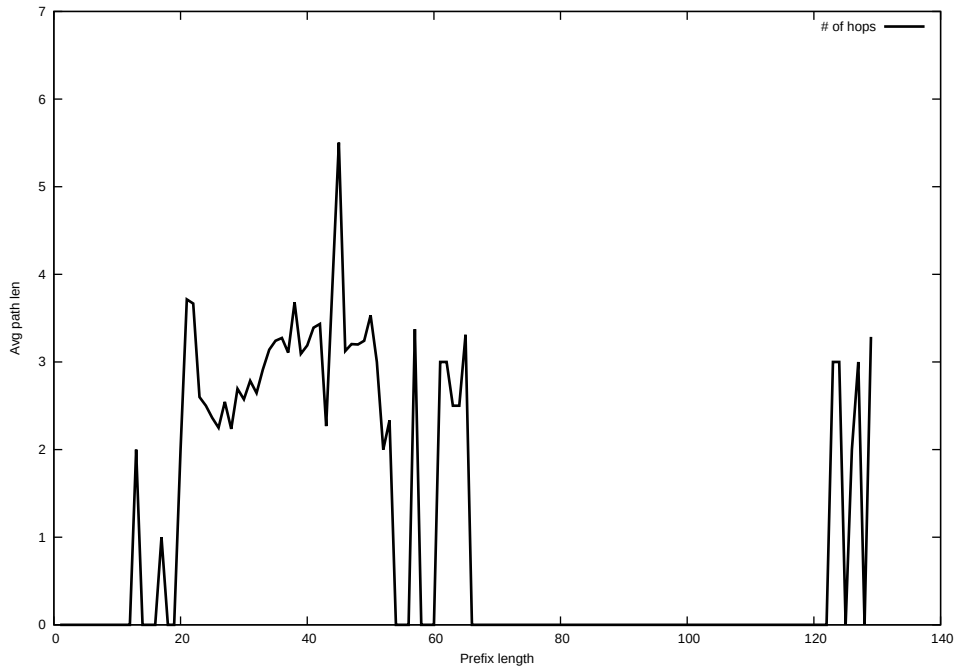
2013-08-02



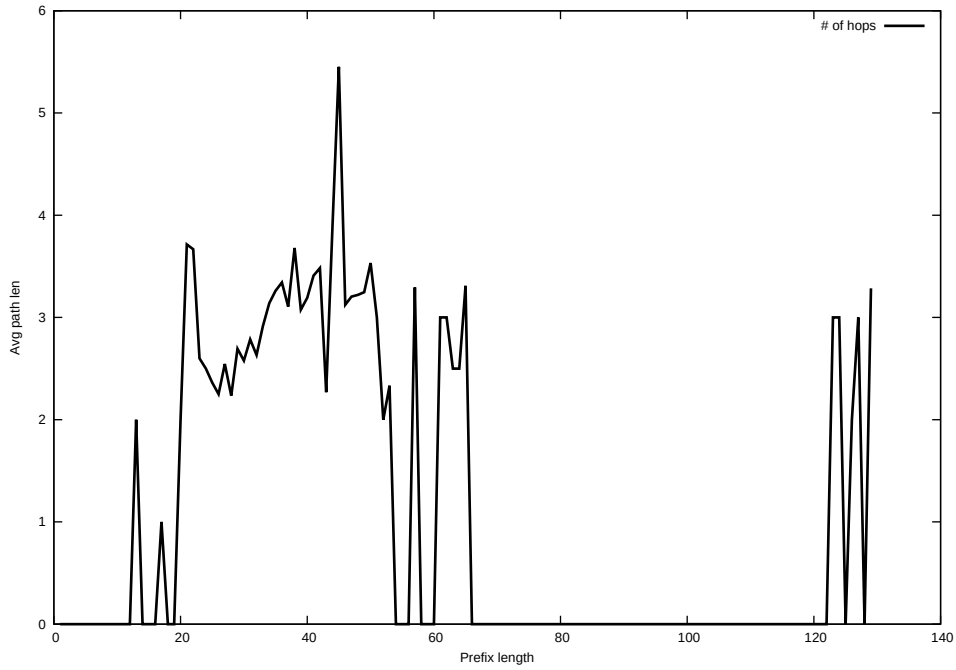
2013-08-03



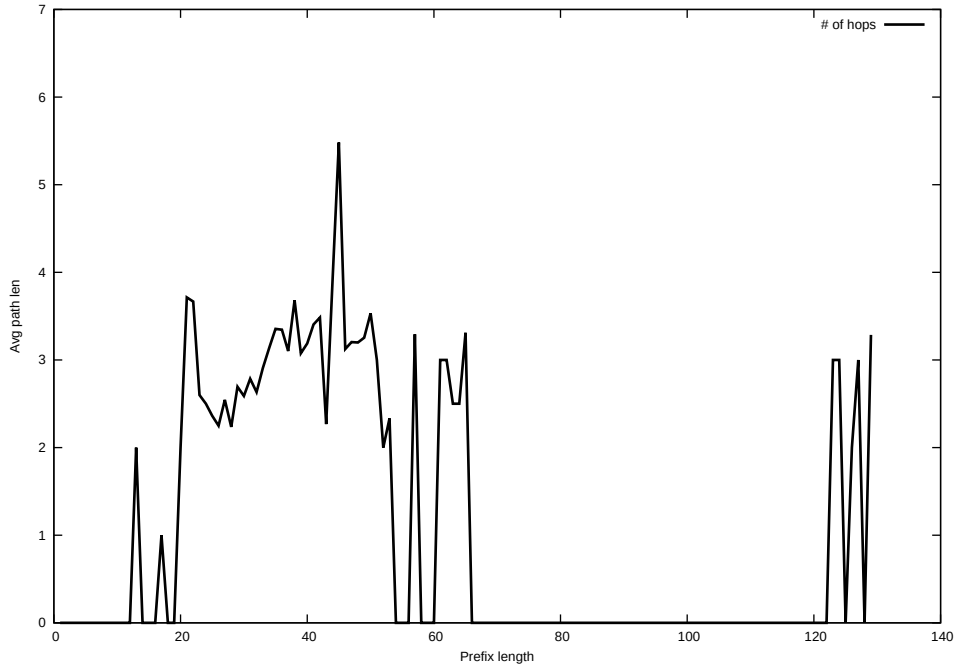
2013-08-04



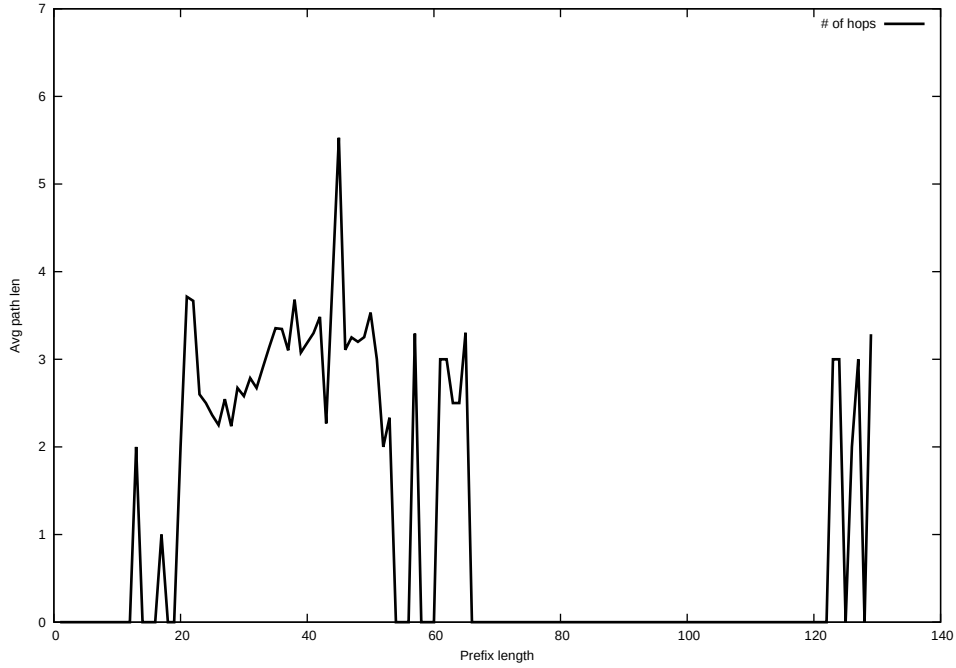
2013-08-05



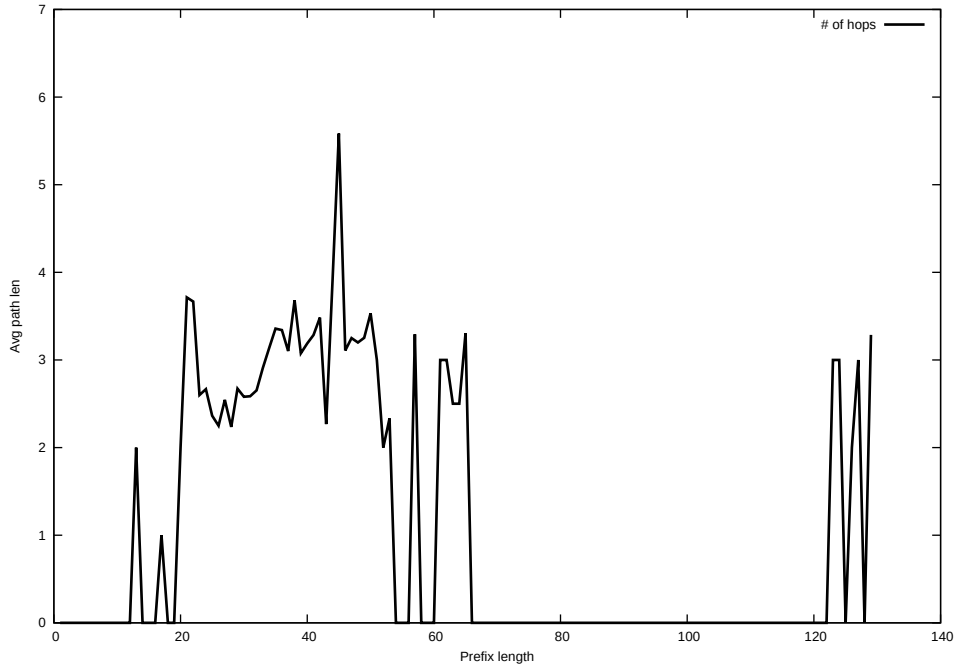
2013-08-06



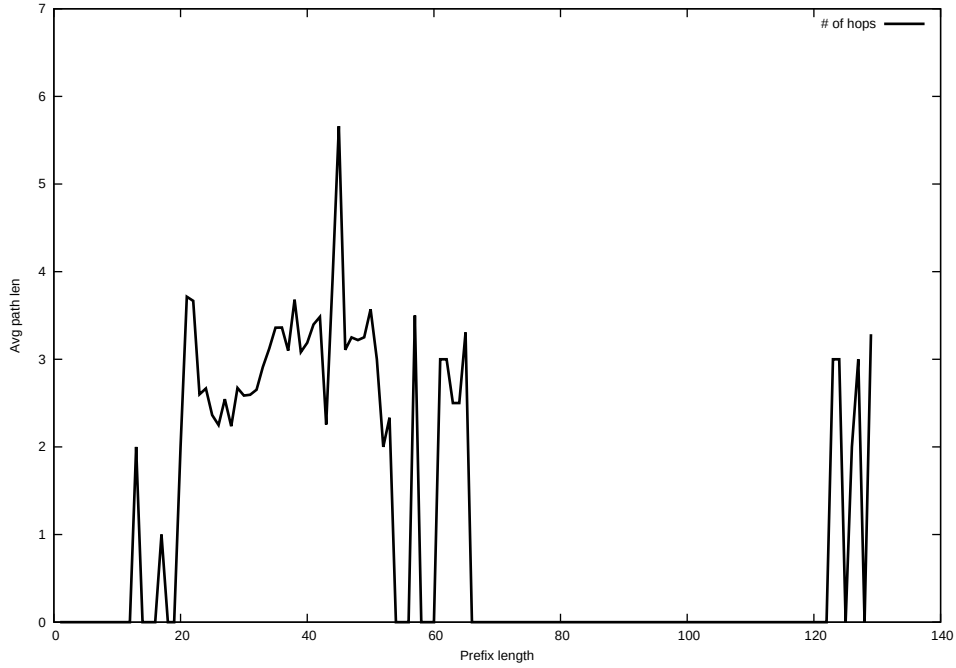
2013-08-07



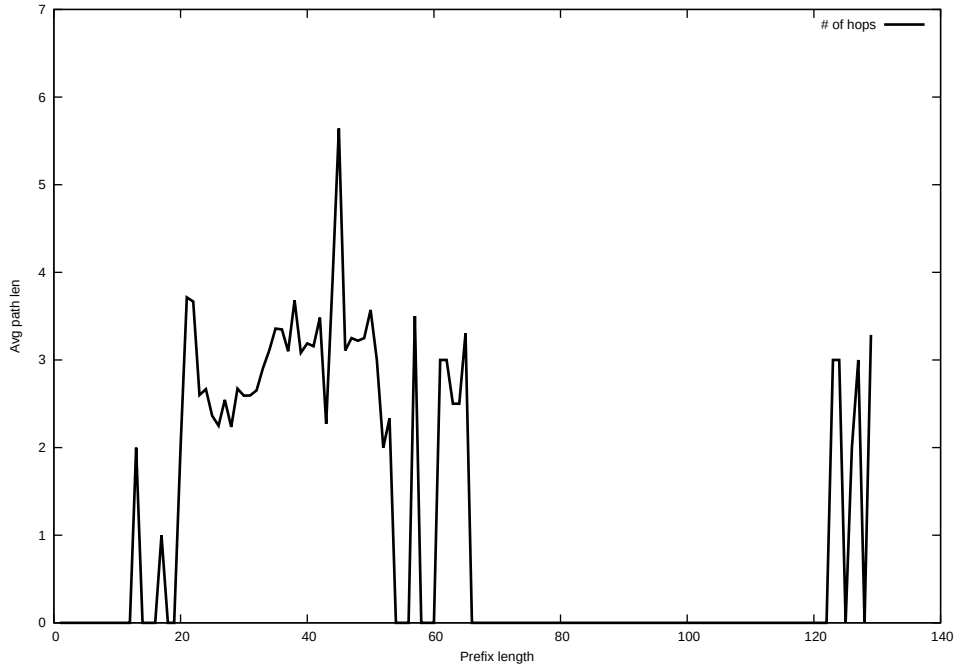
2013-08-08



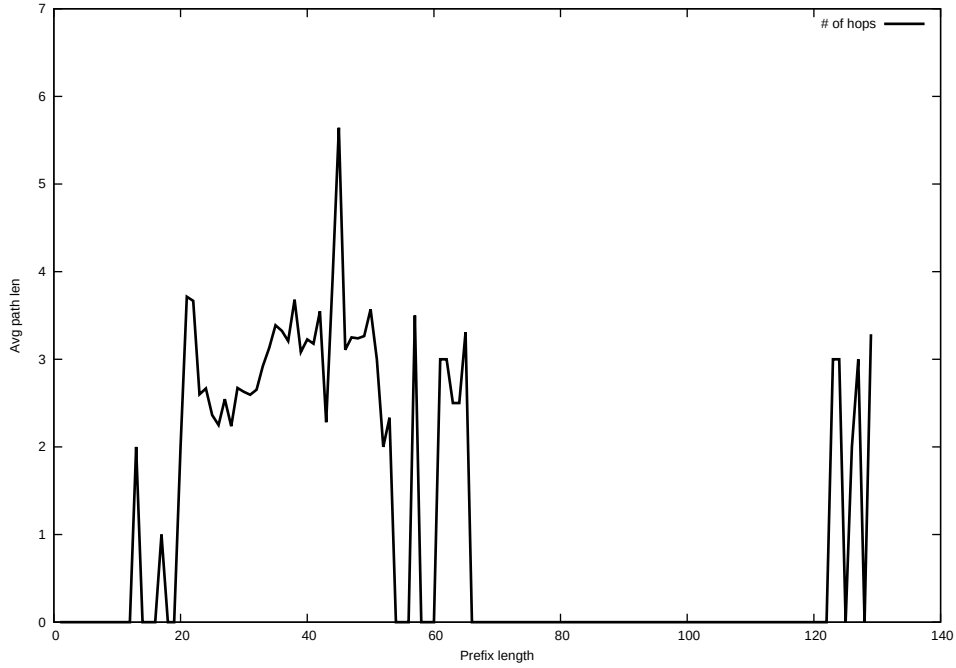
2013-08-09



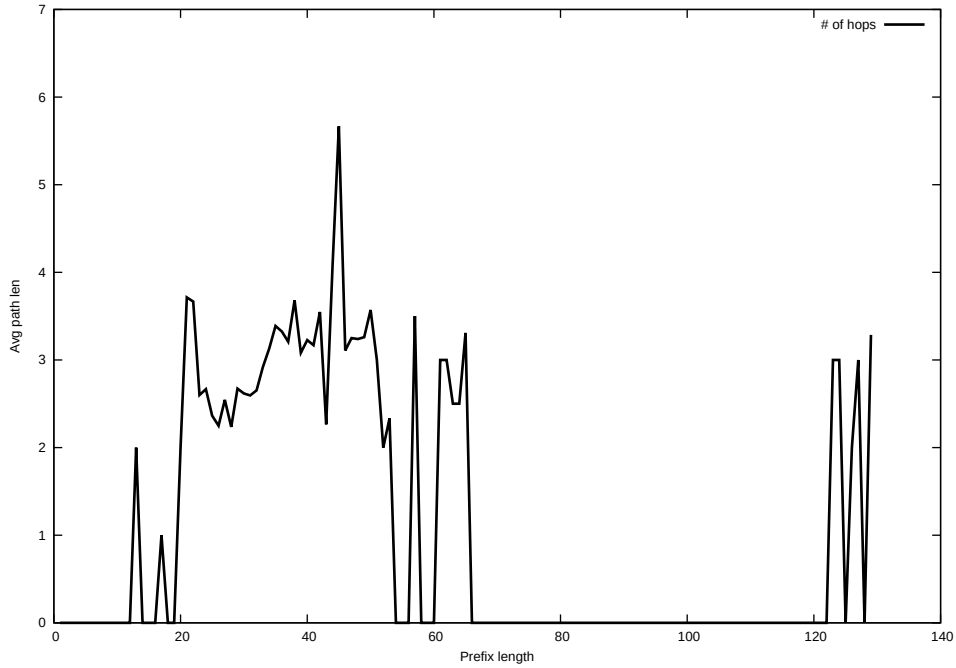
2013-08-10



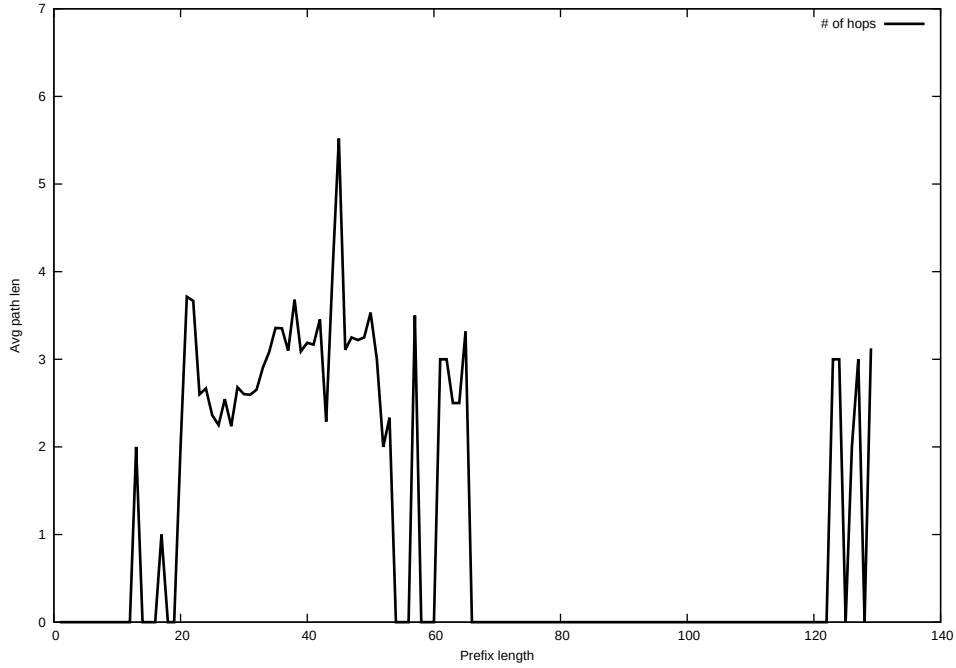
2013-08-11



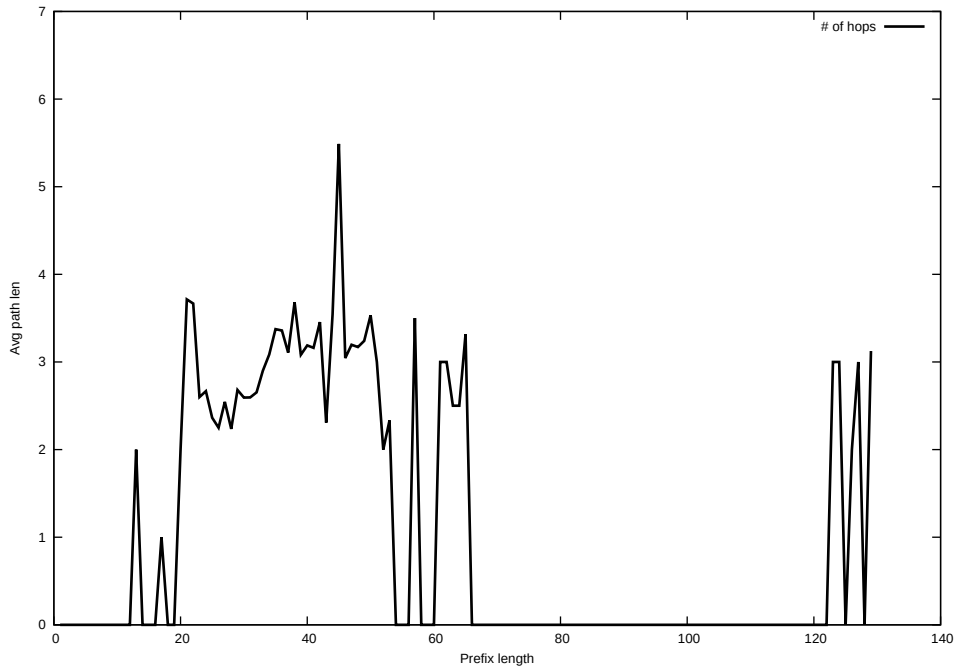
2013-08-12



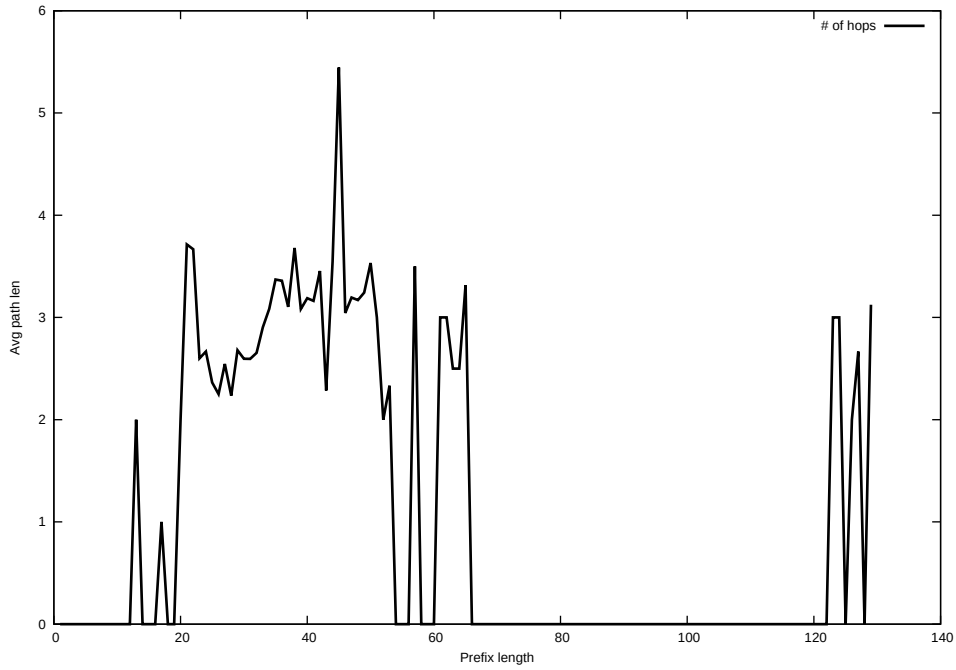
2013-08-13



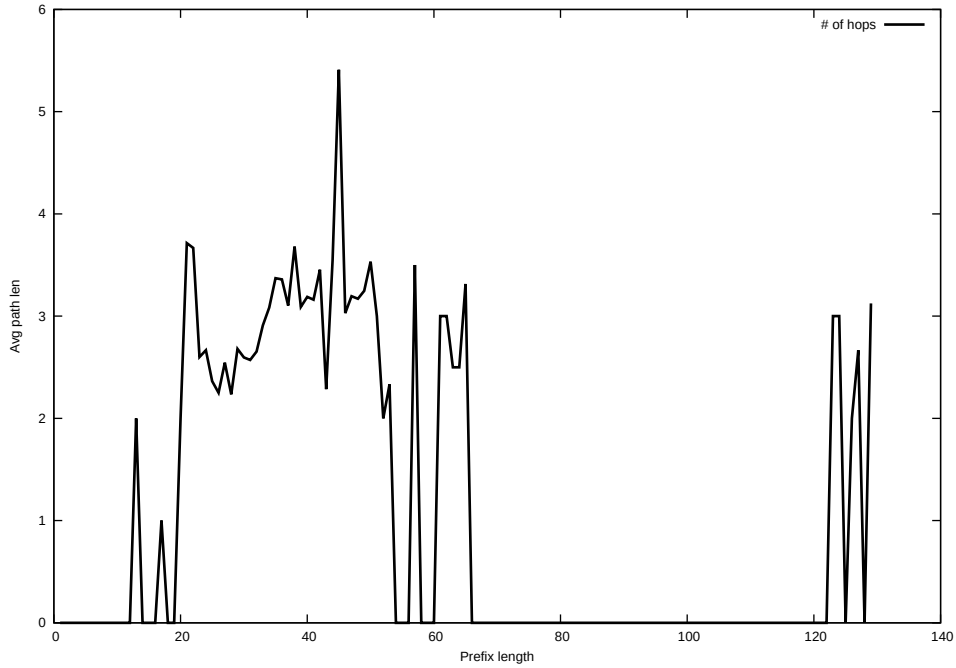
2013-08-15



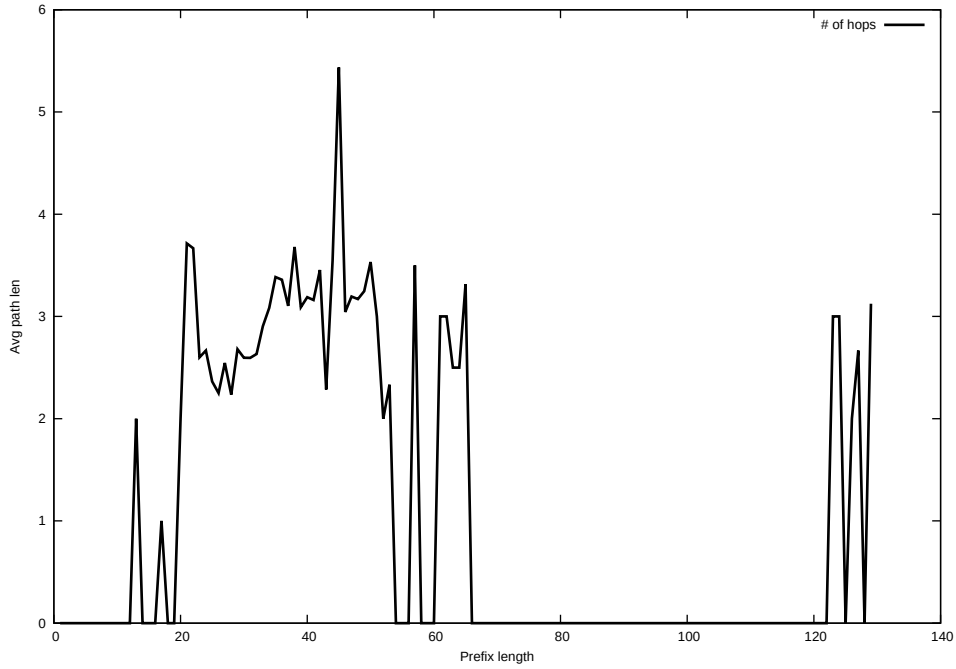
2013-08-16



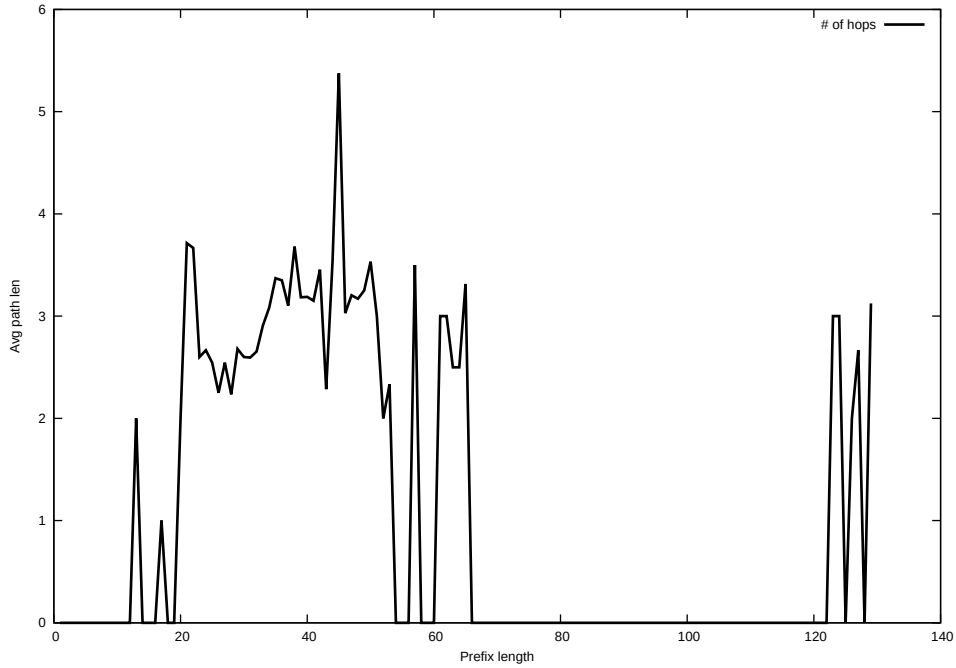
2013-08-17



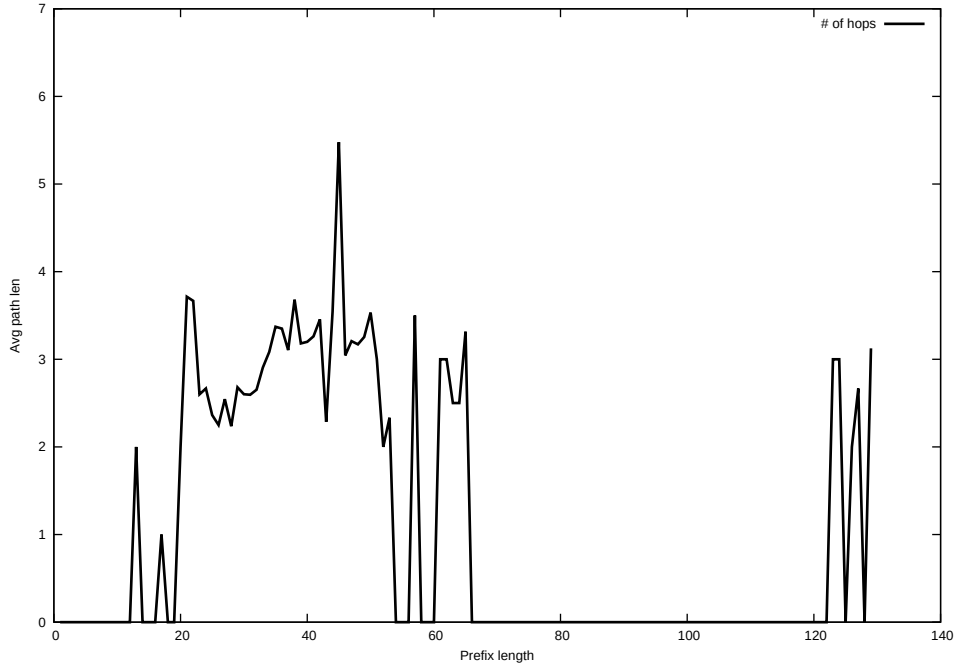
2013-08-18



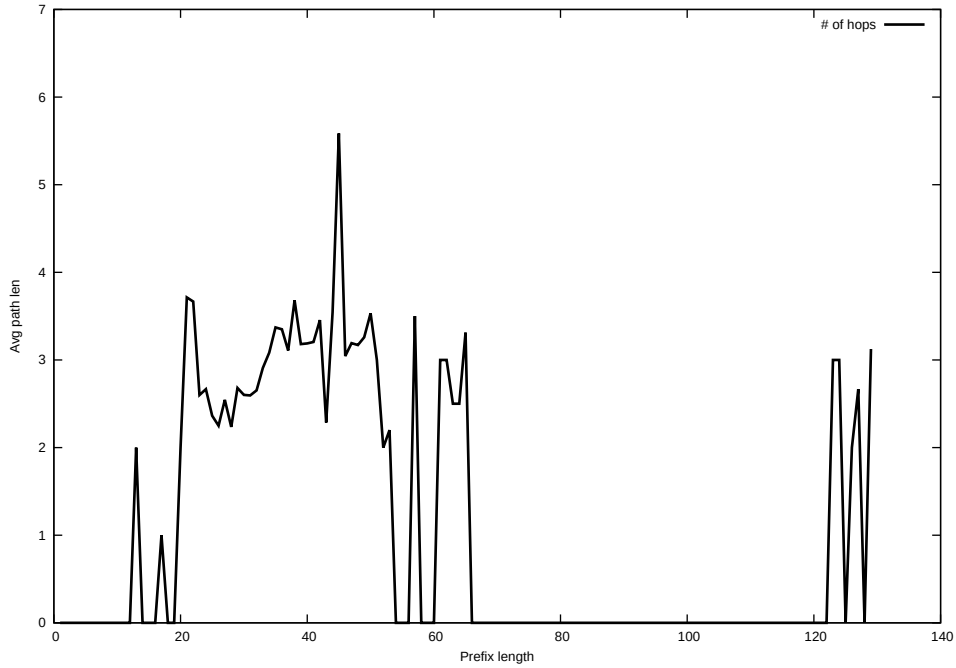
2013-08-19



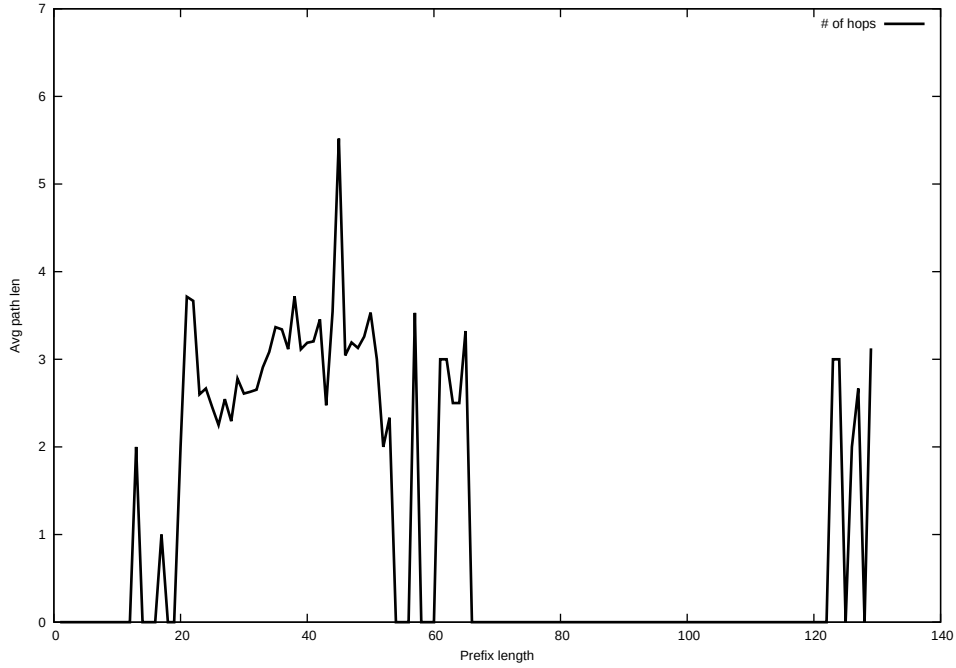
2013-08-20



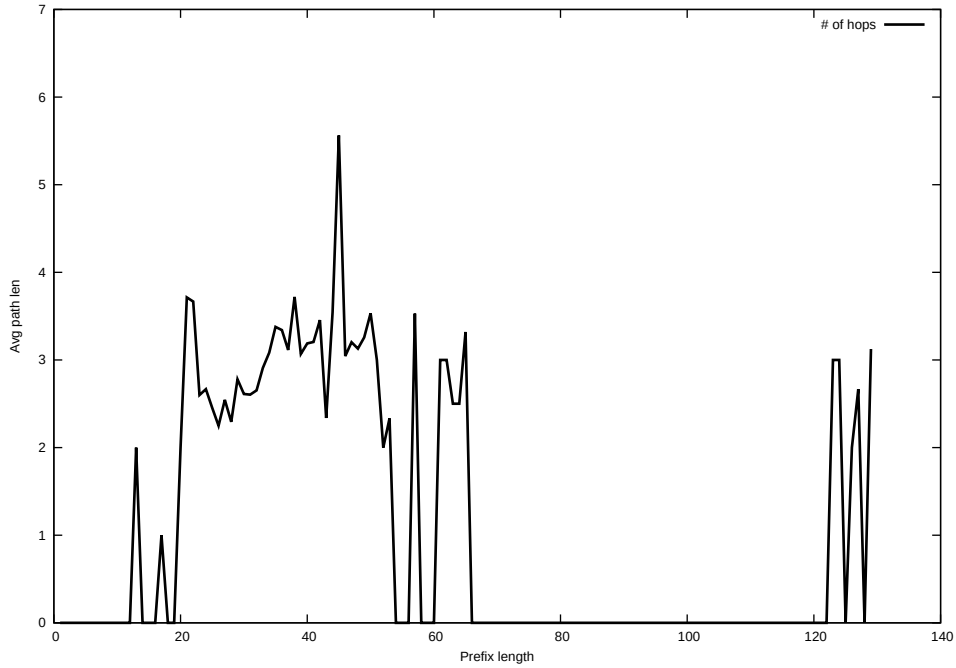
2013-08-21



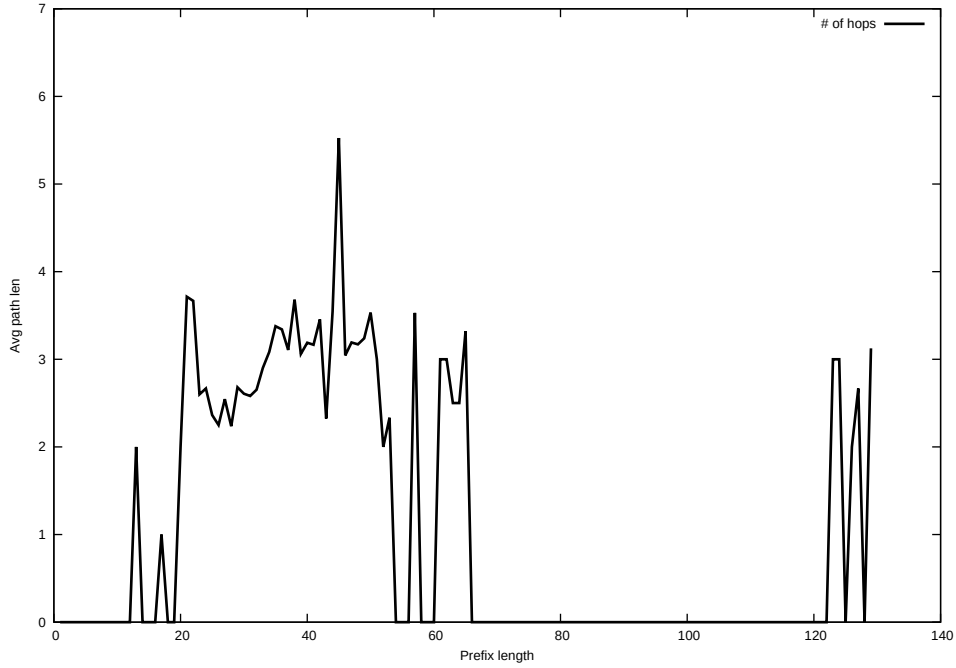
2013-08-22



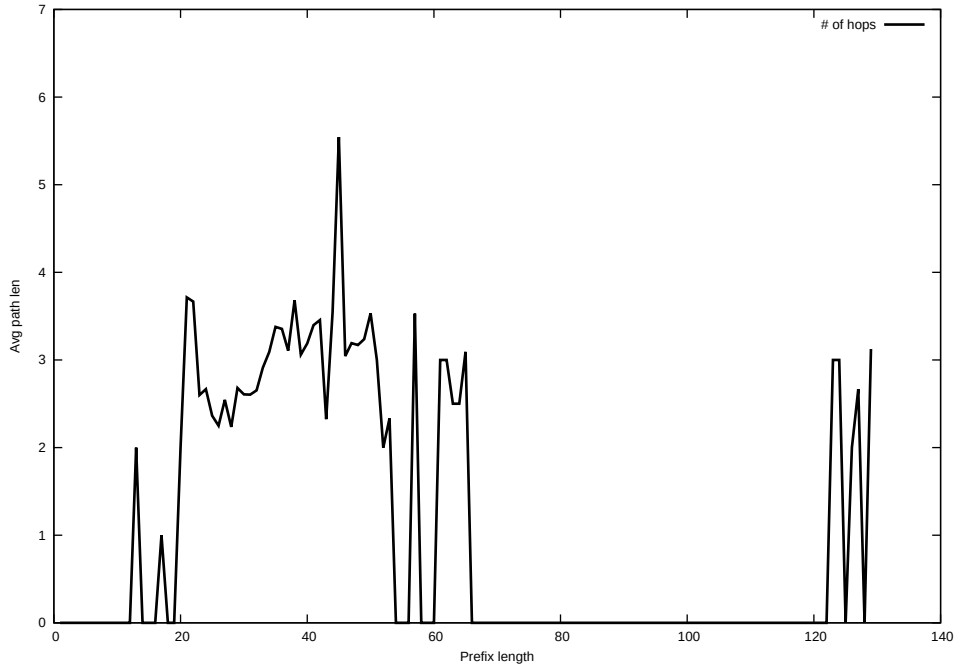
2013-08-23



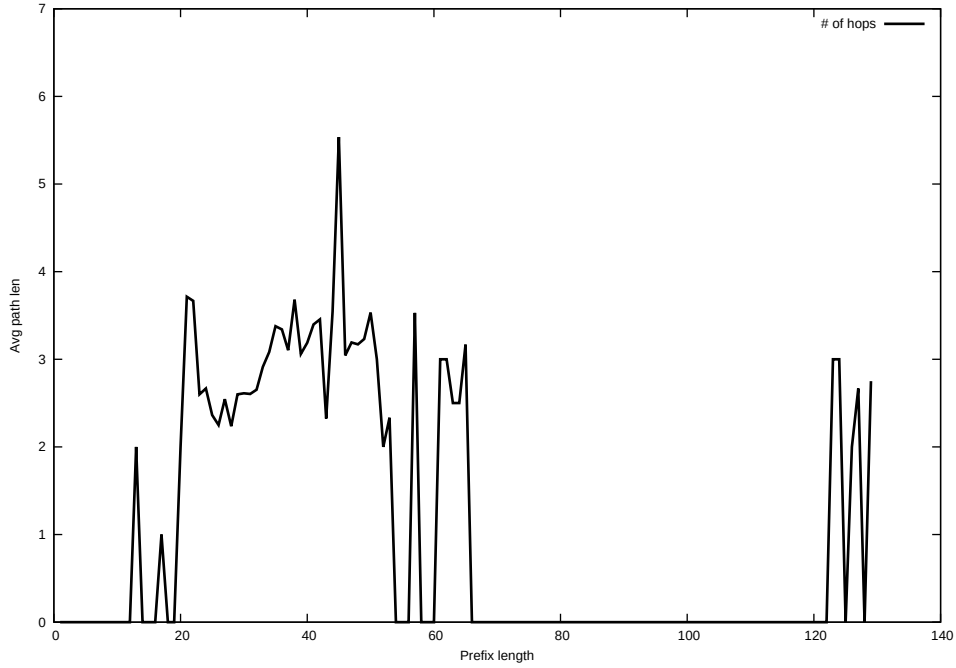
2013-08-24



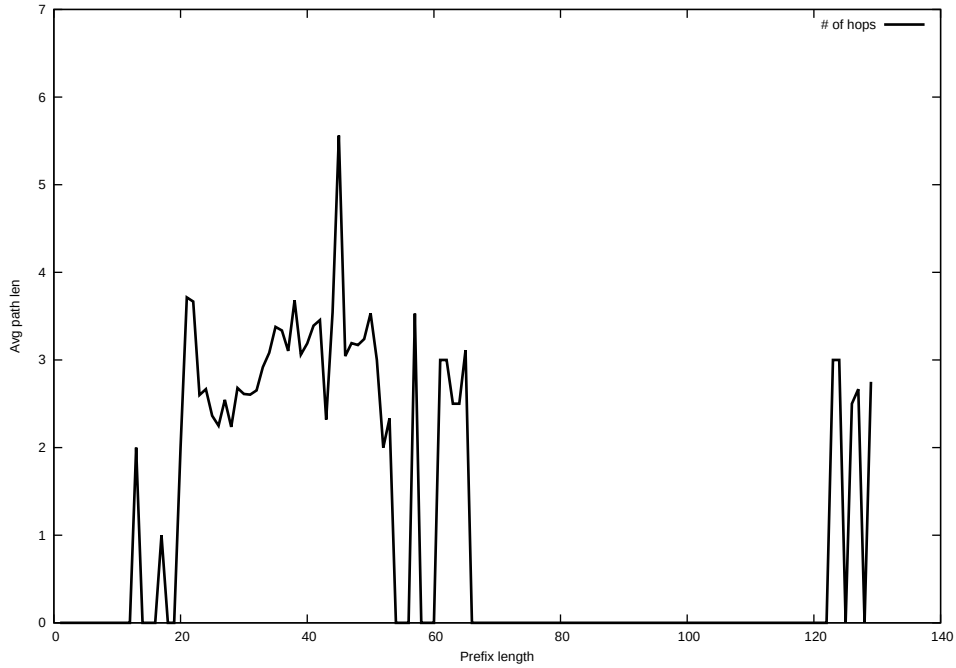
2013-08-25



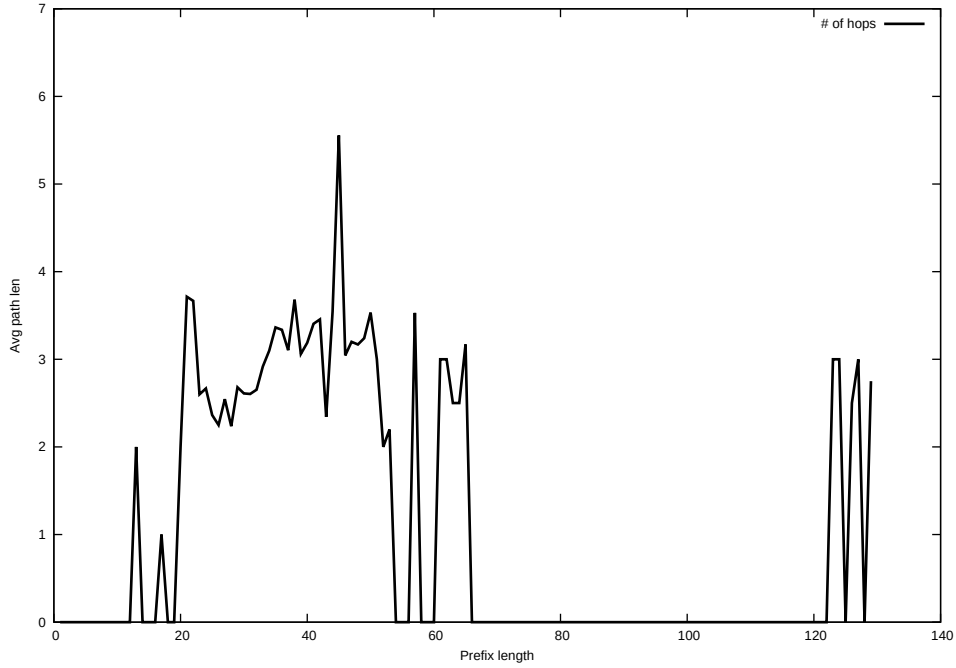
2013-08-26



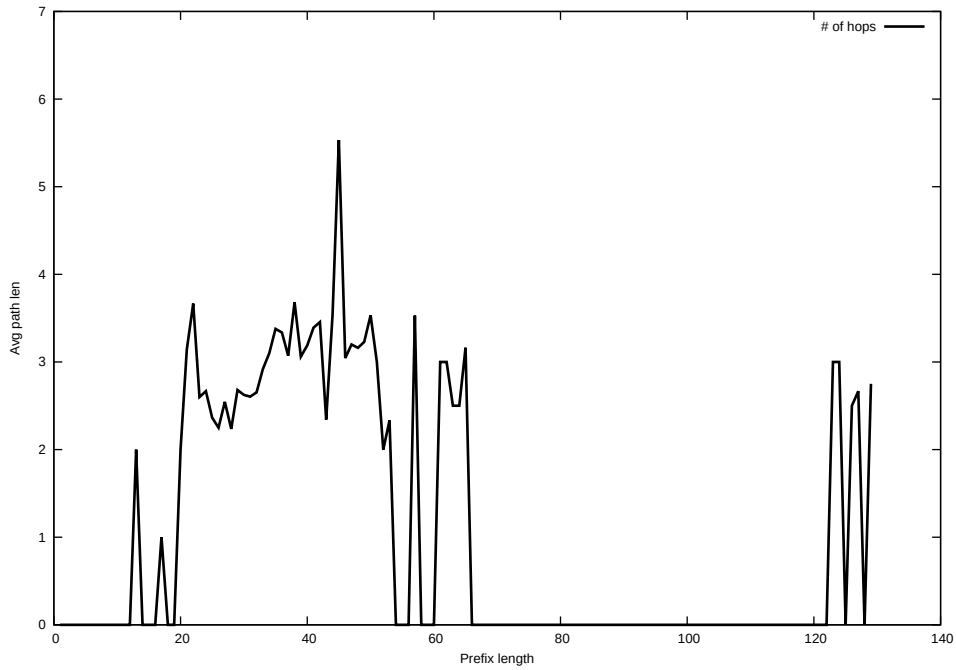
2013-08-27



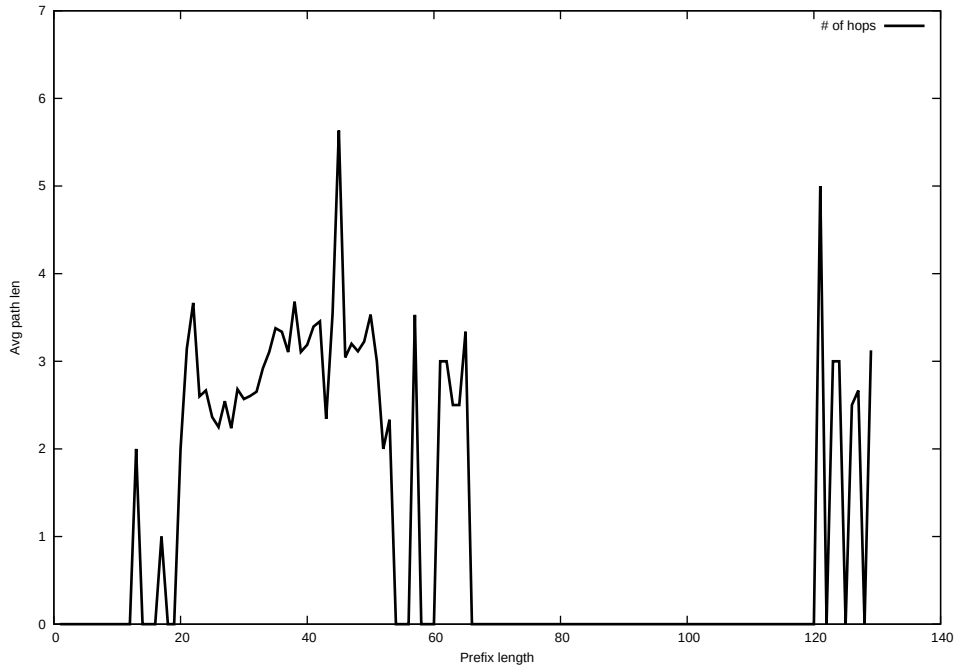
2013-08-28



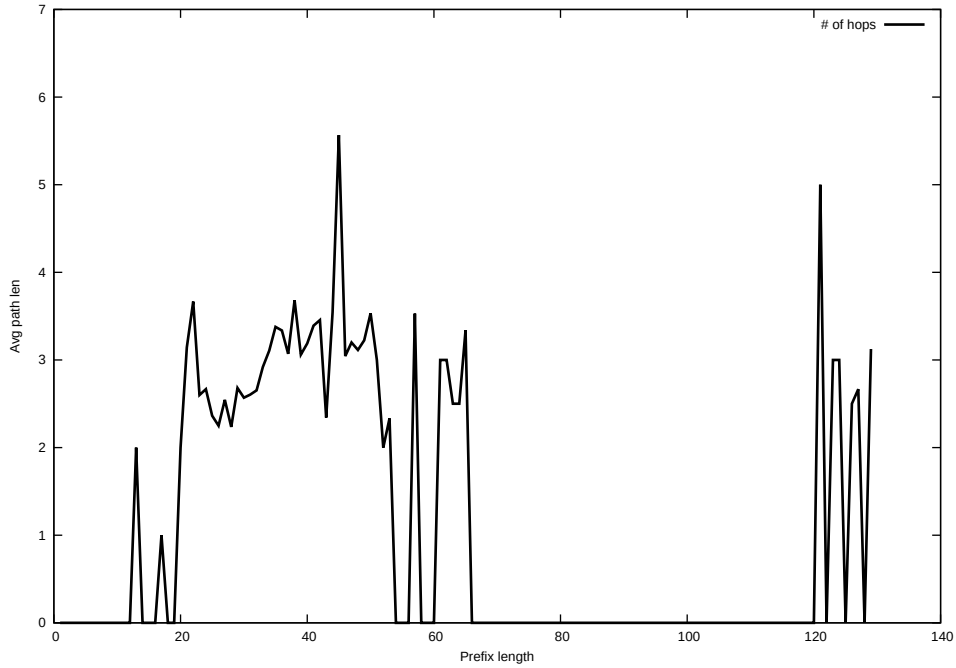
2013-08-29



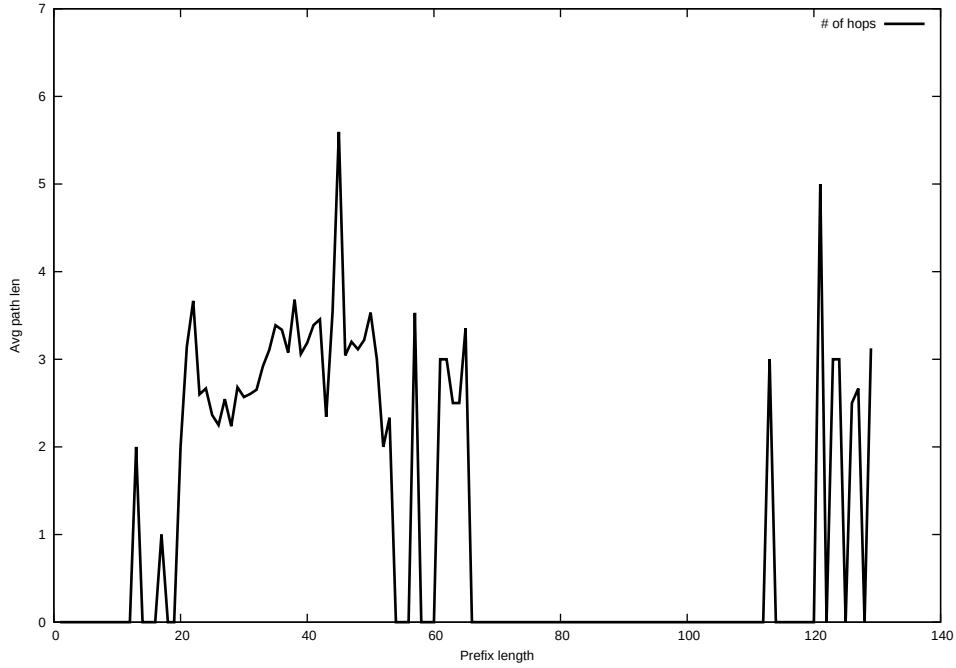
2013-08-30



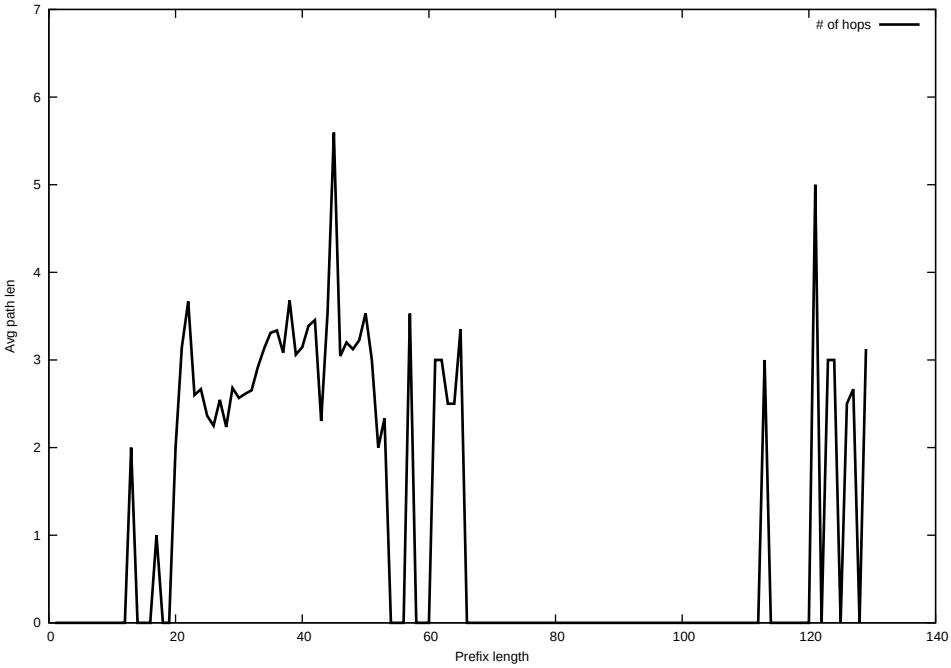
2013-08-31



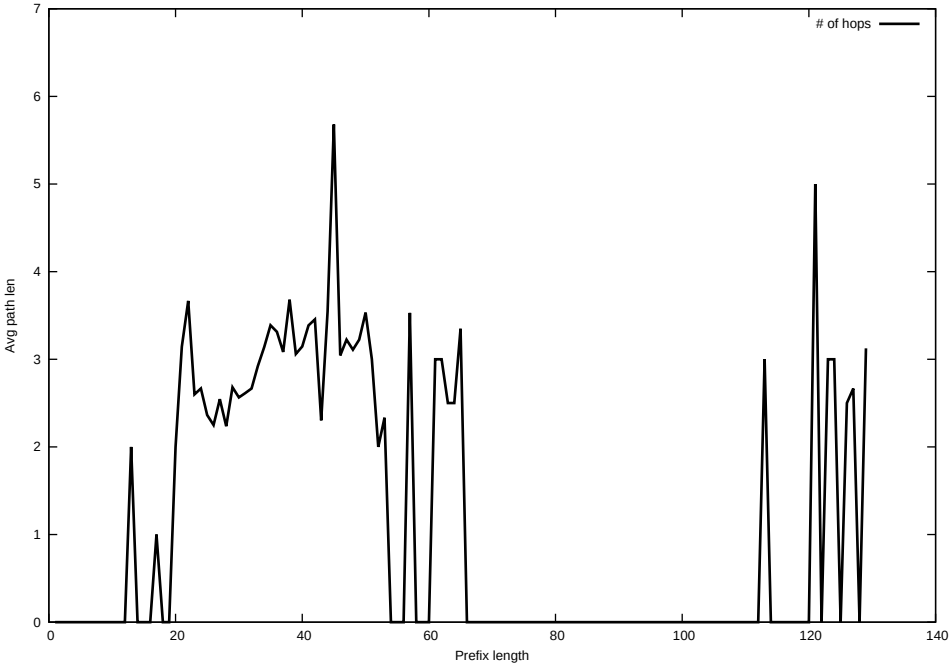
2013-09-01



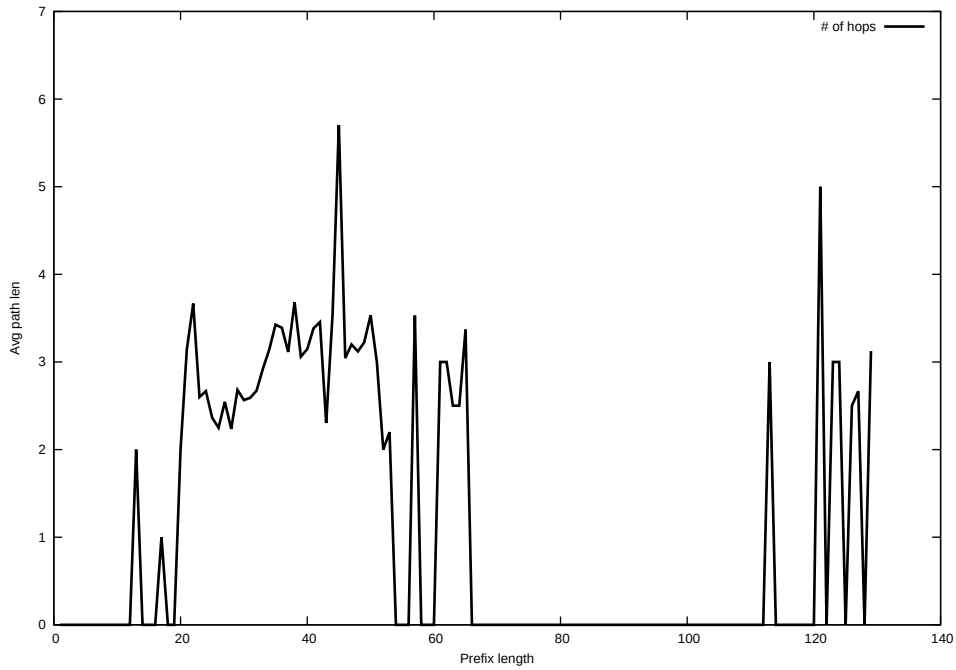
2013-09-02



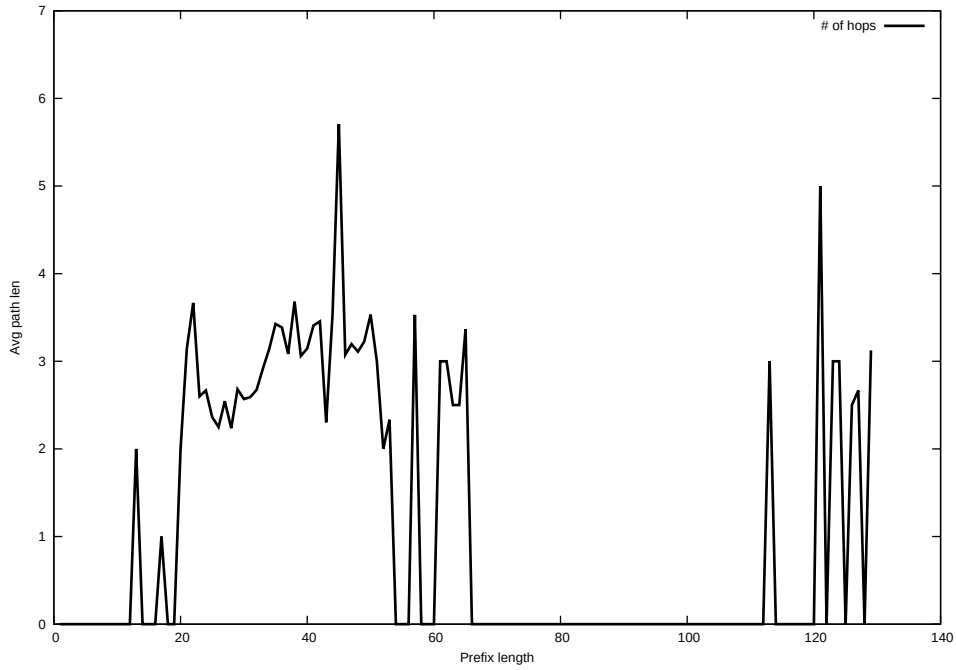
2013-09-03



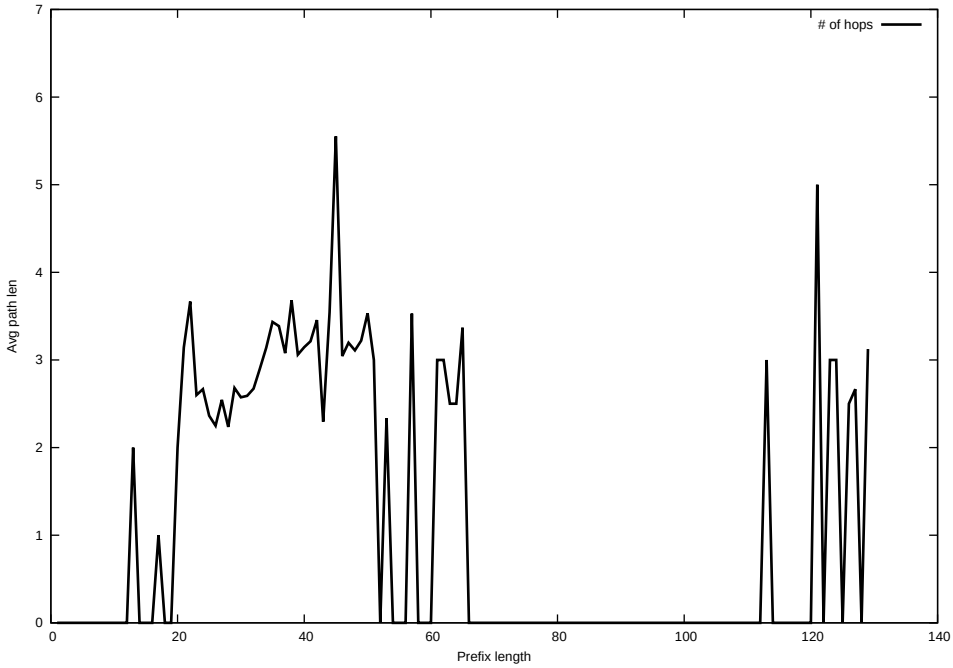
2013-09-04



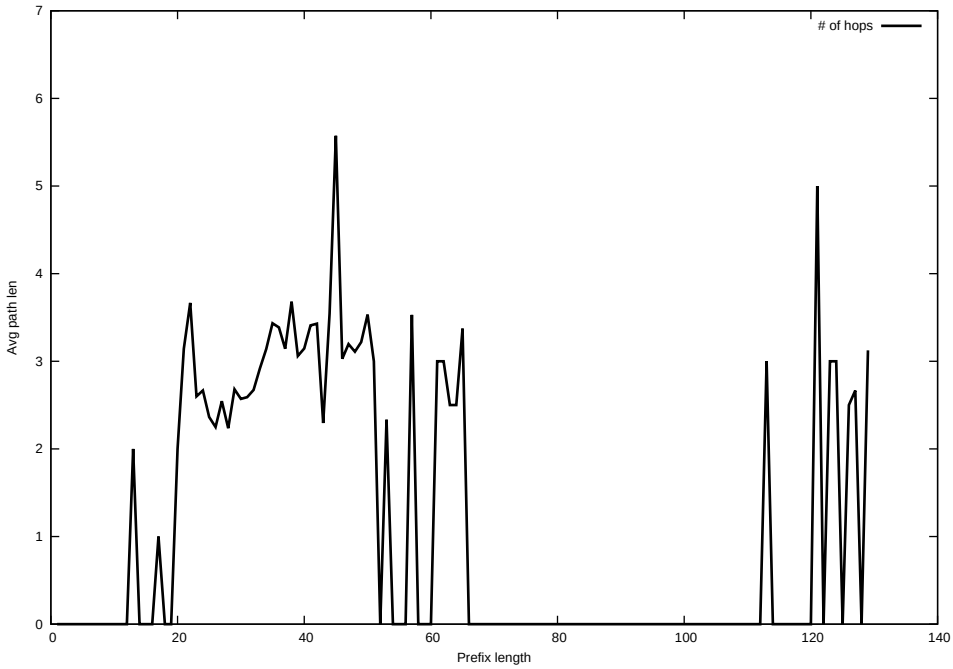
2013-09-05



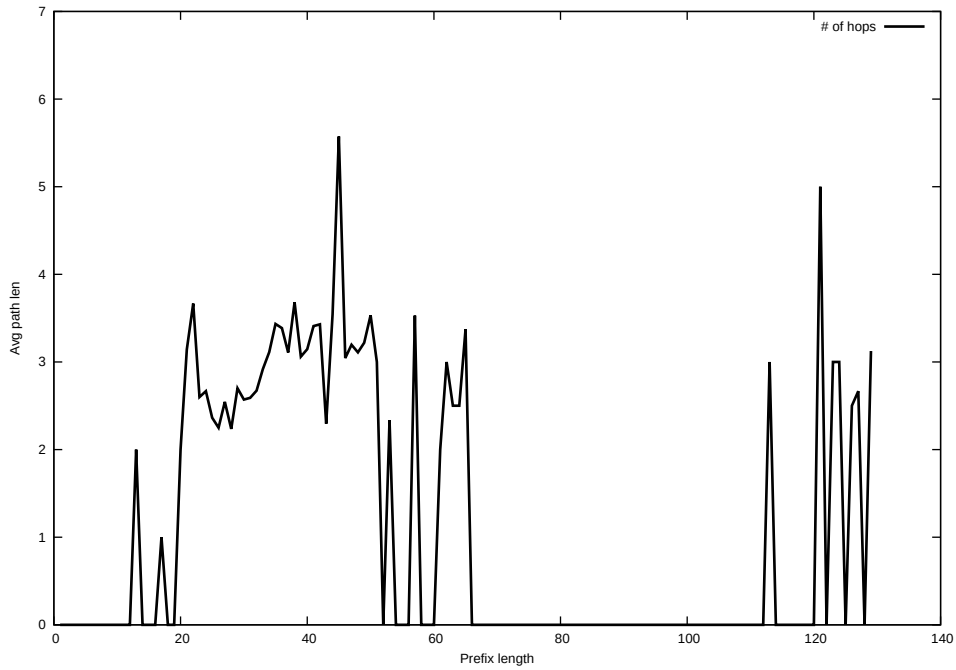
2013-09-06



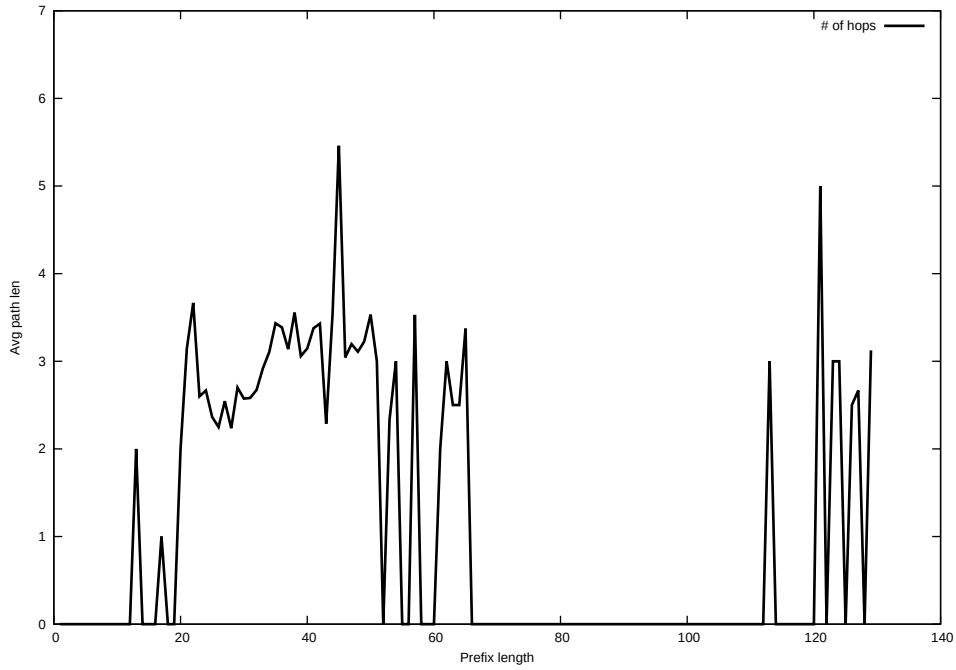
2013-09-07



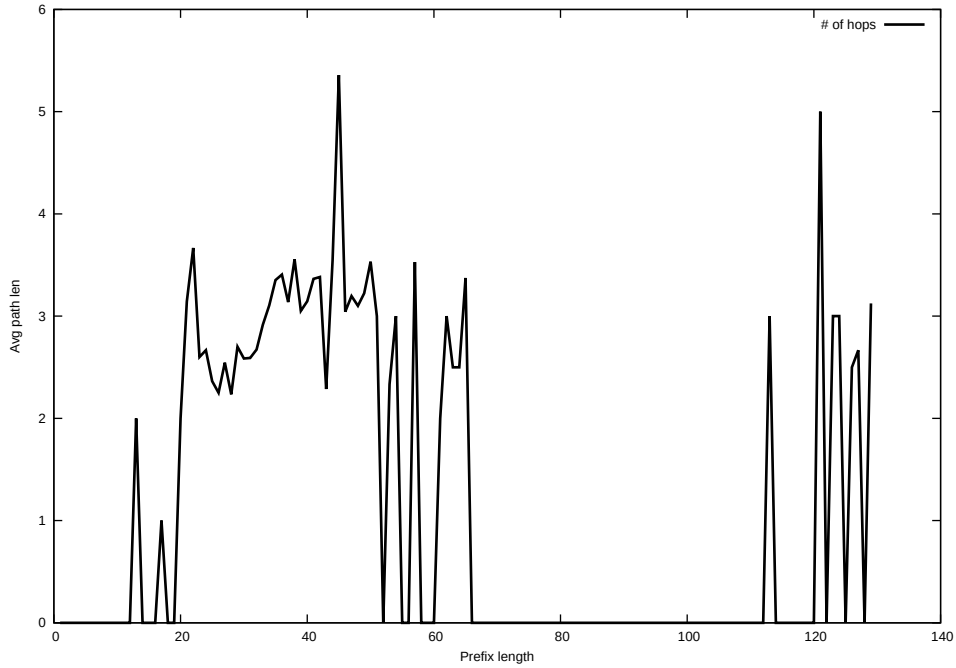
2013-09-08



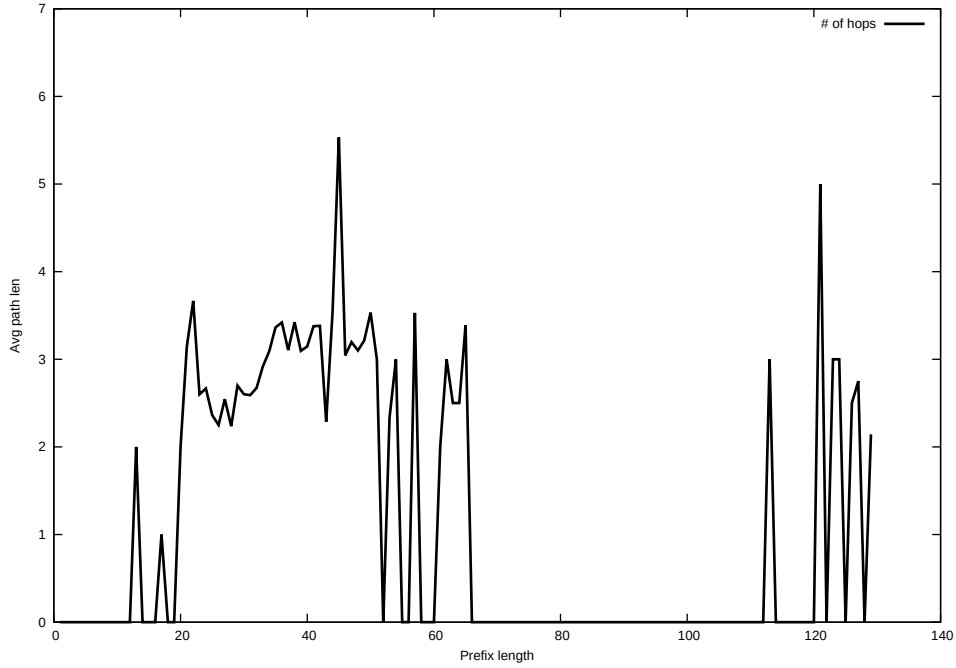
2013-09-09



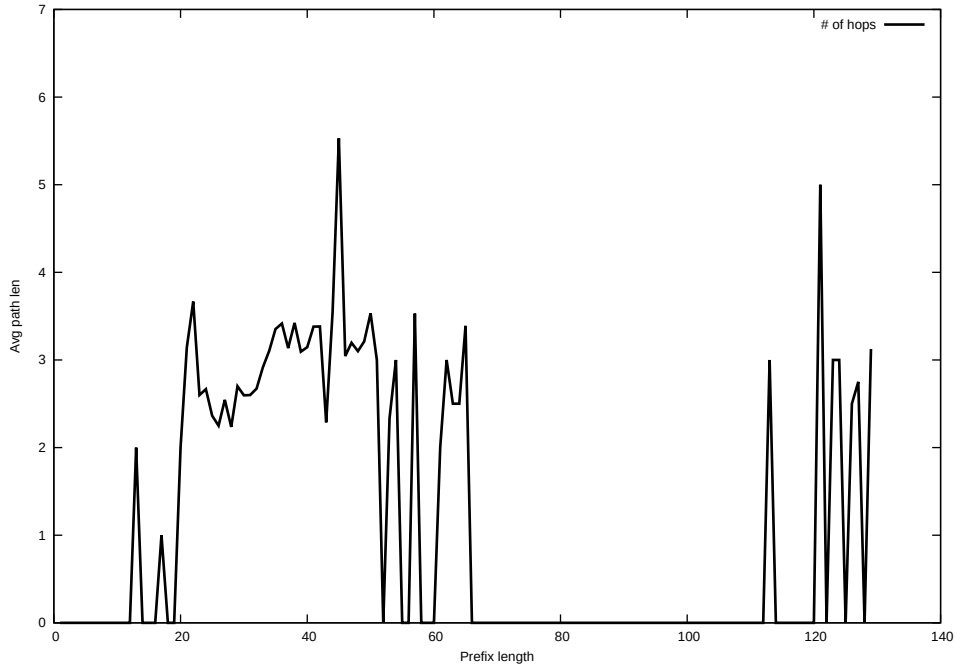
2013-09-10



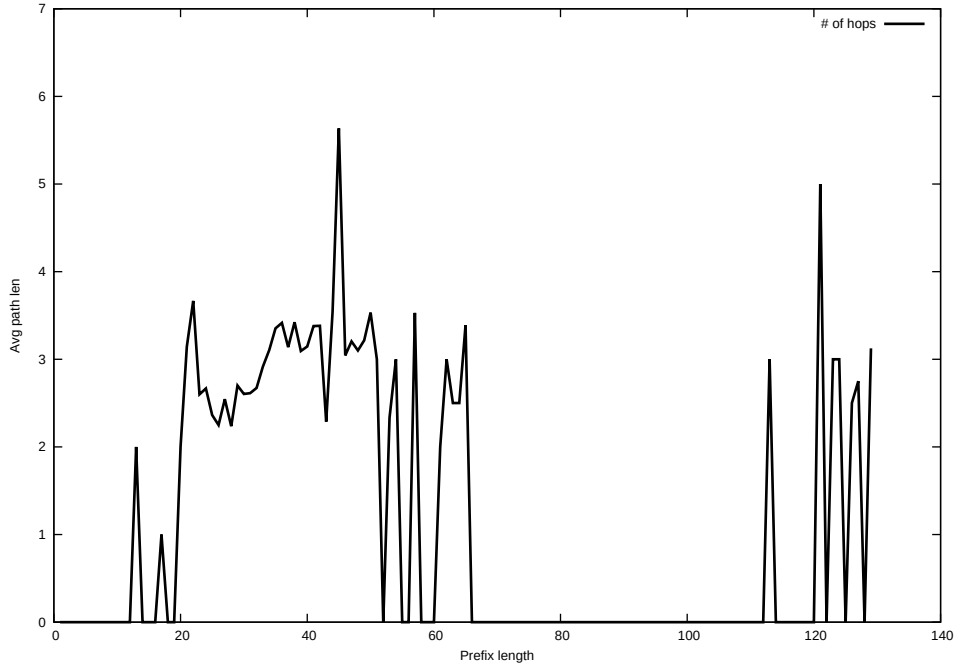
2013-09-11



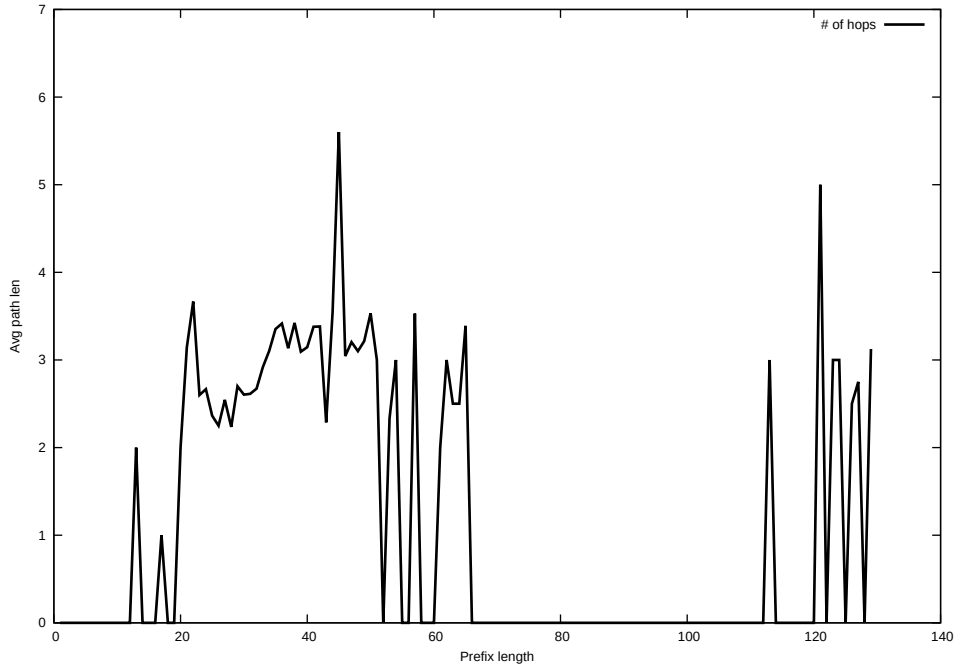
2013-09-12



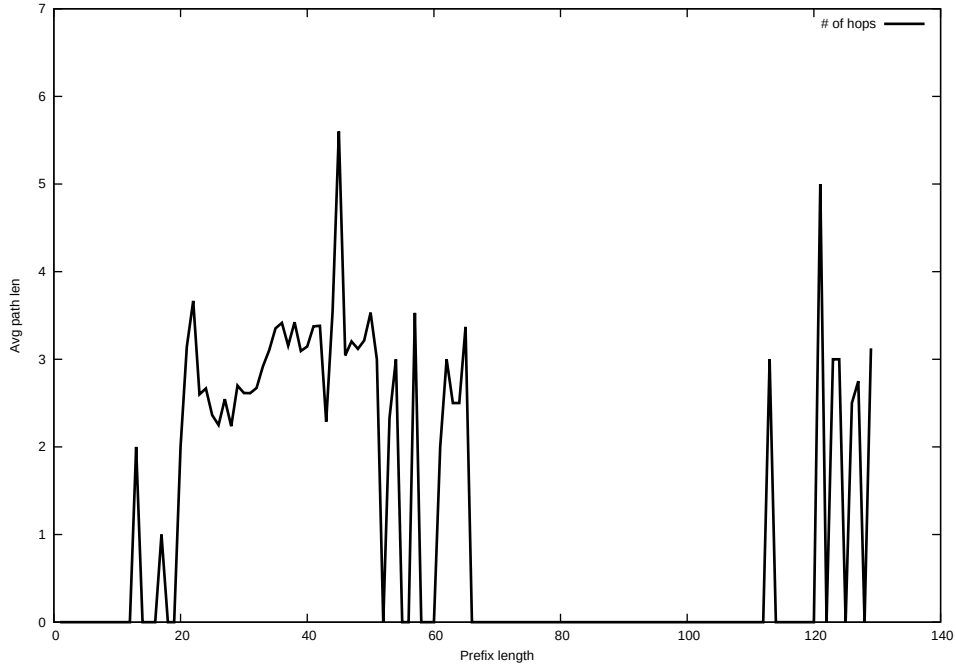
2013-09-13



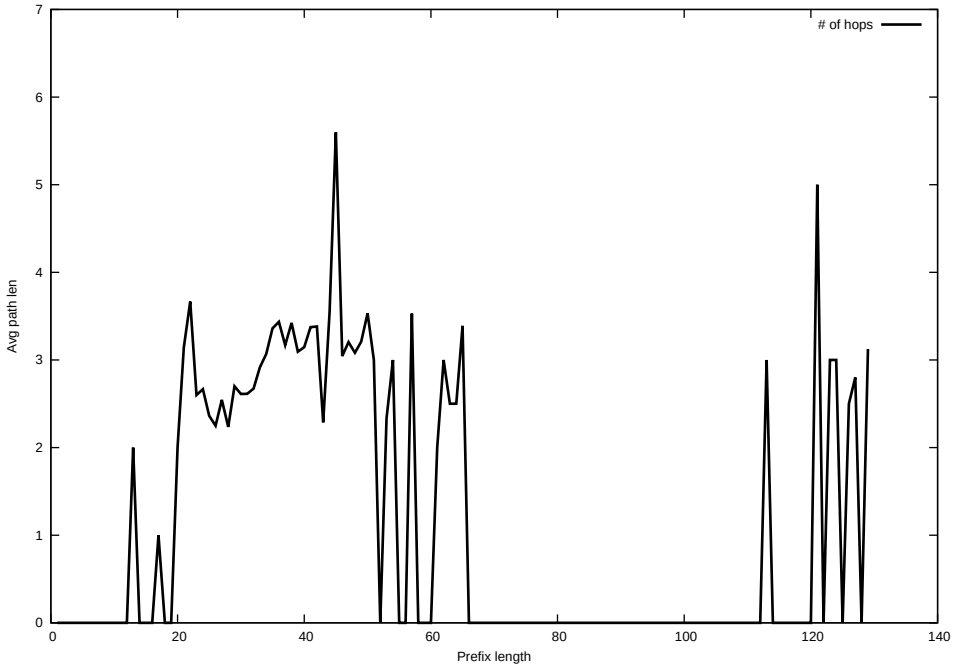
2013-09-14



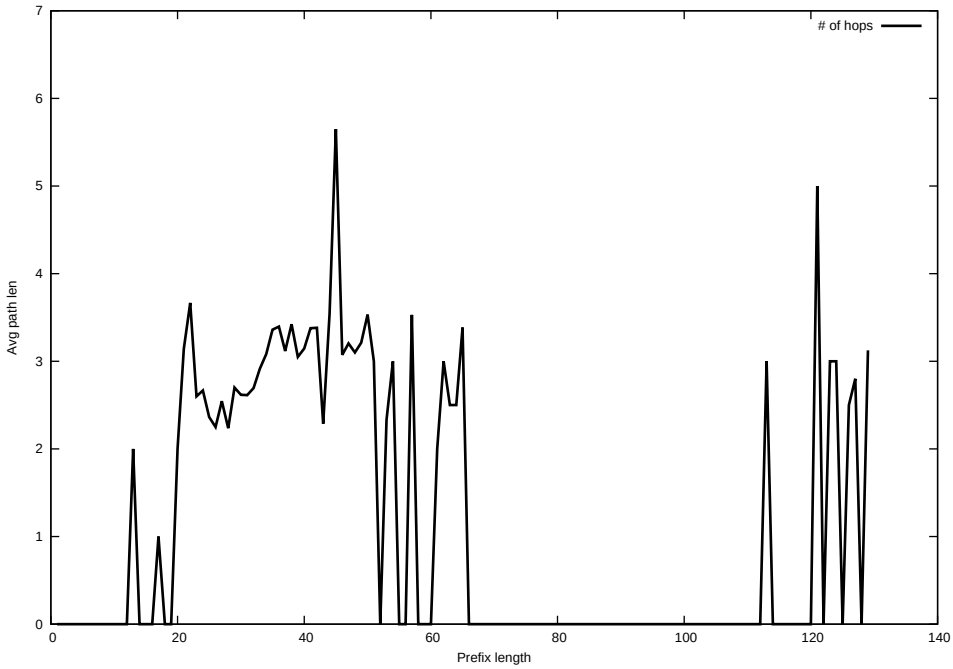
2013-09-15



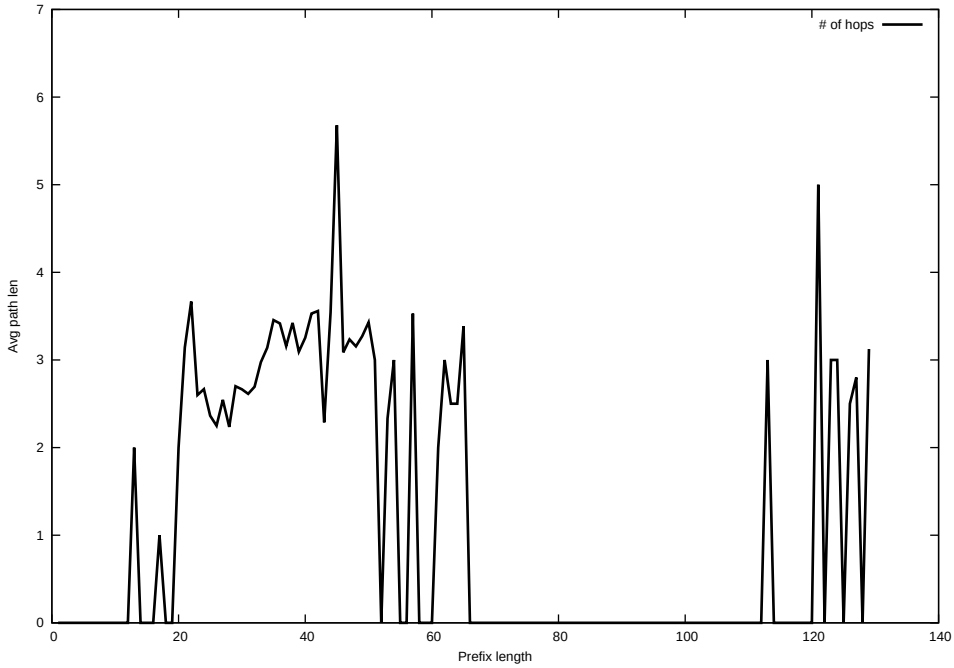
2013-09-16



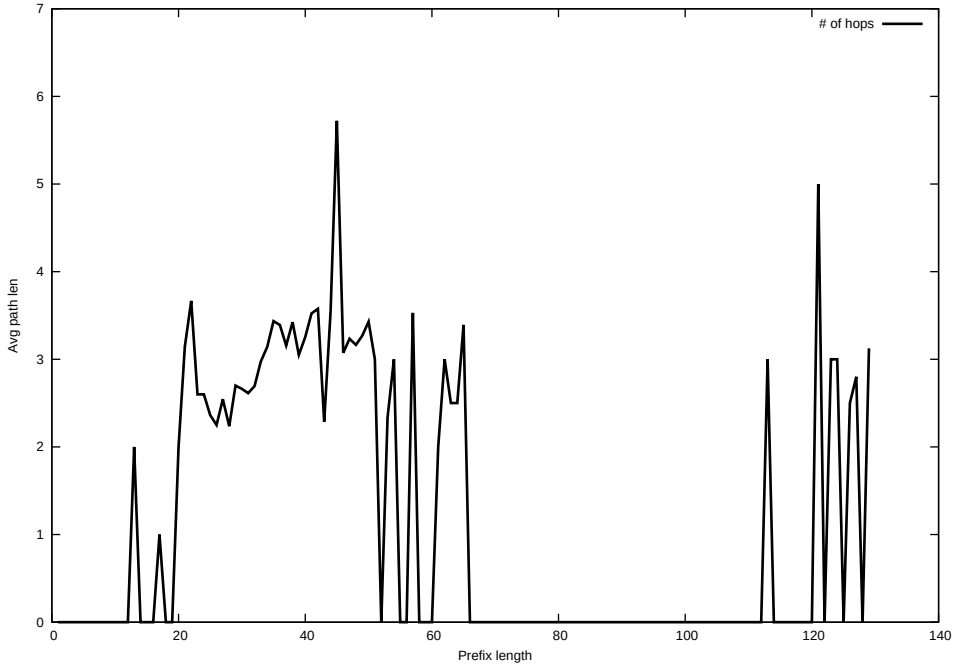
2013-09-17



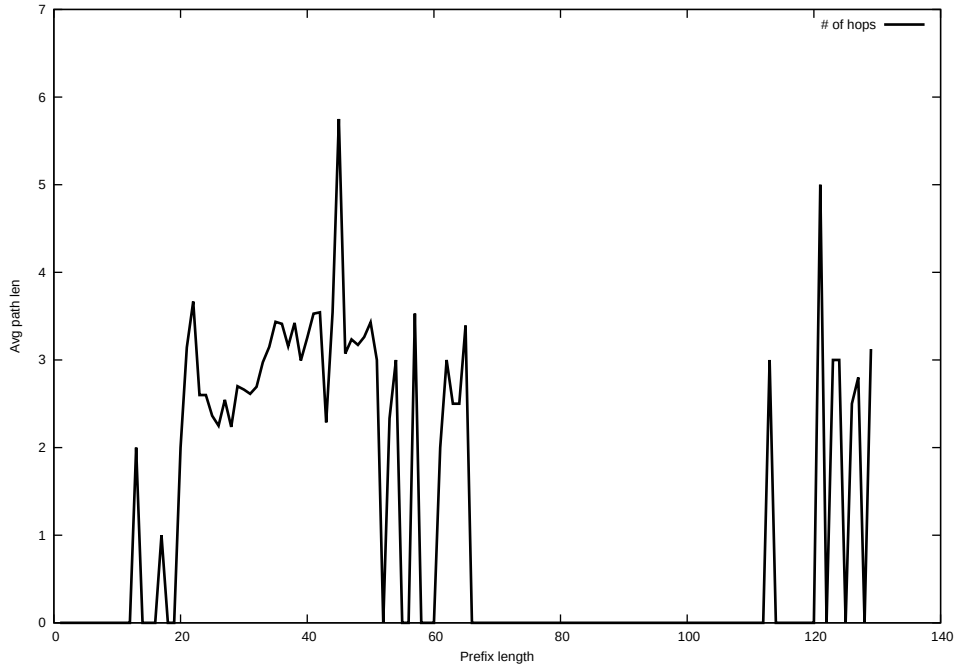
2013-09-18



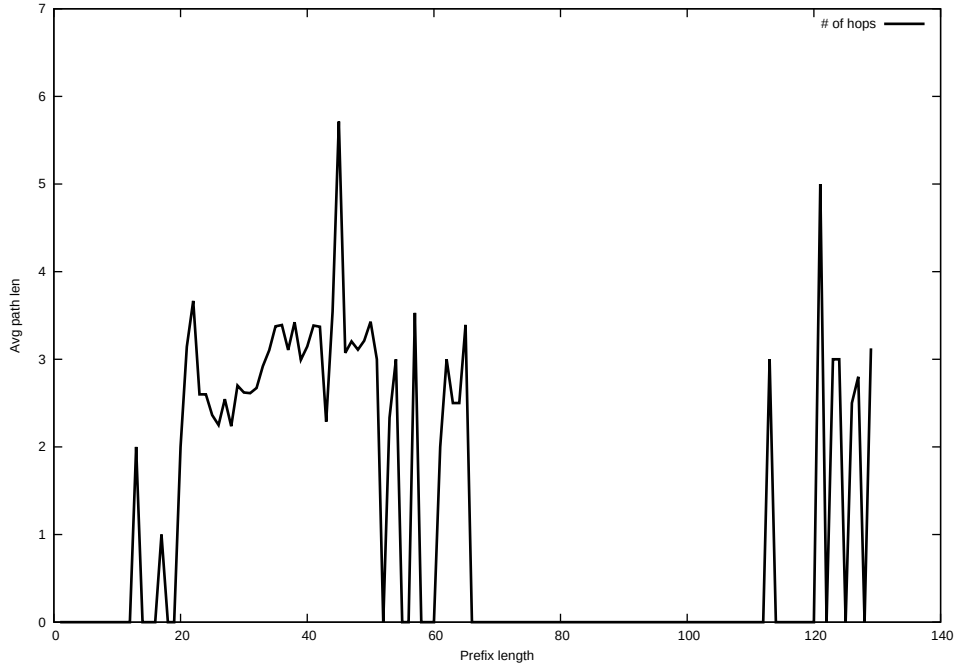
2013-09-19



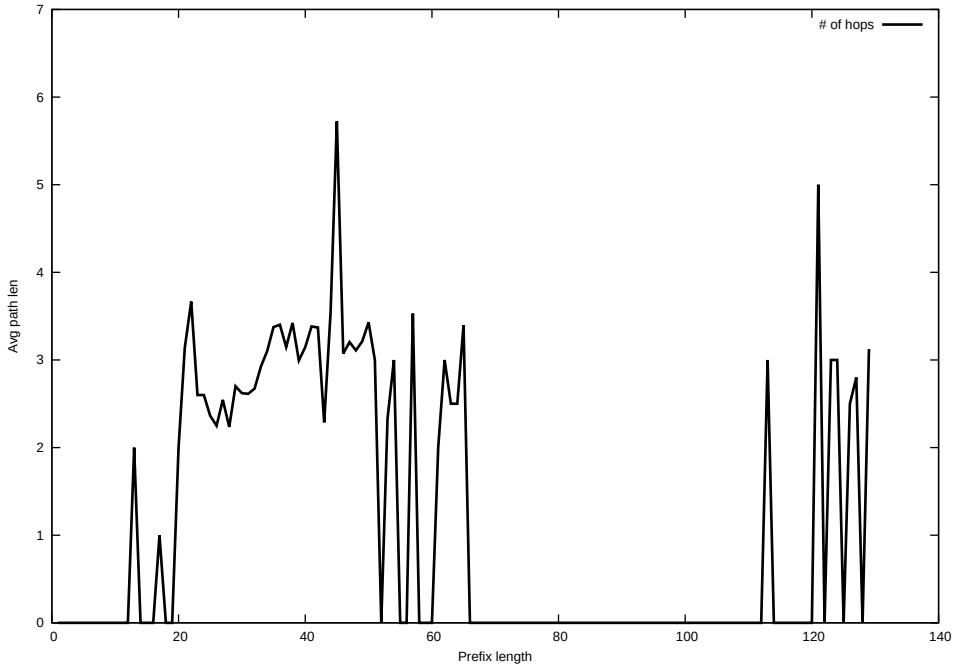
2013-09-20



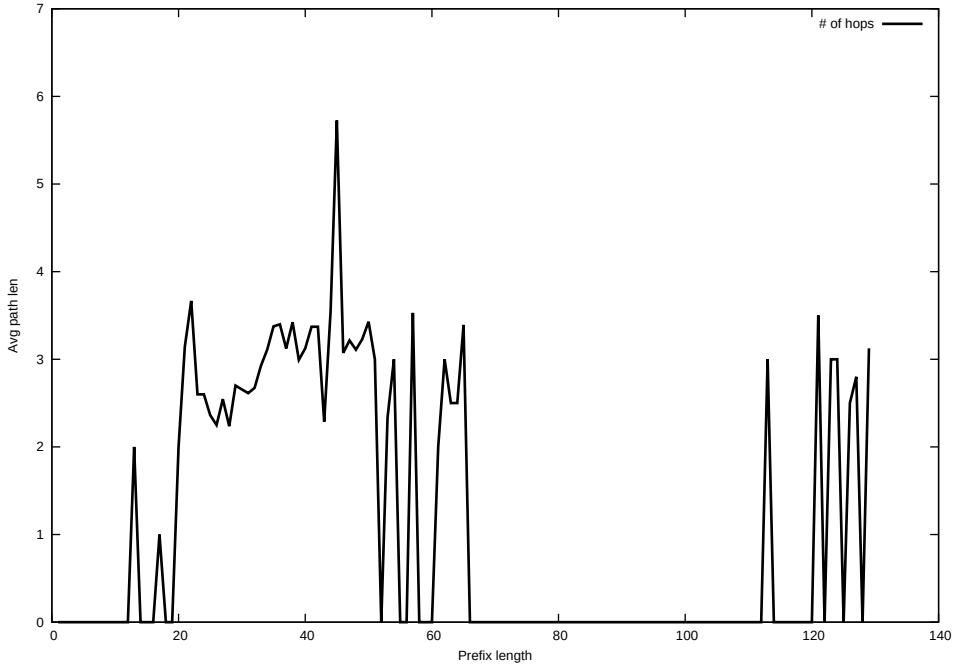
2013-09-21



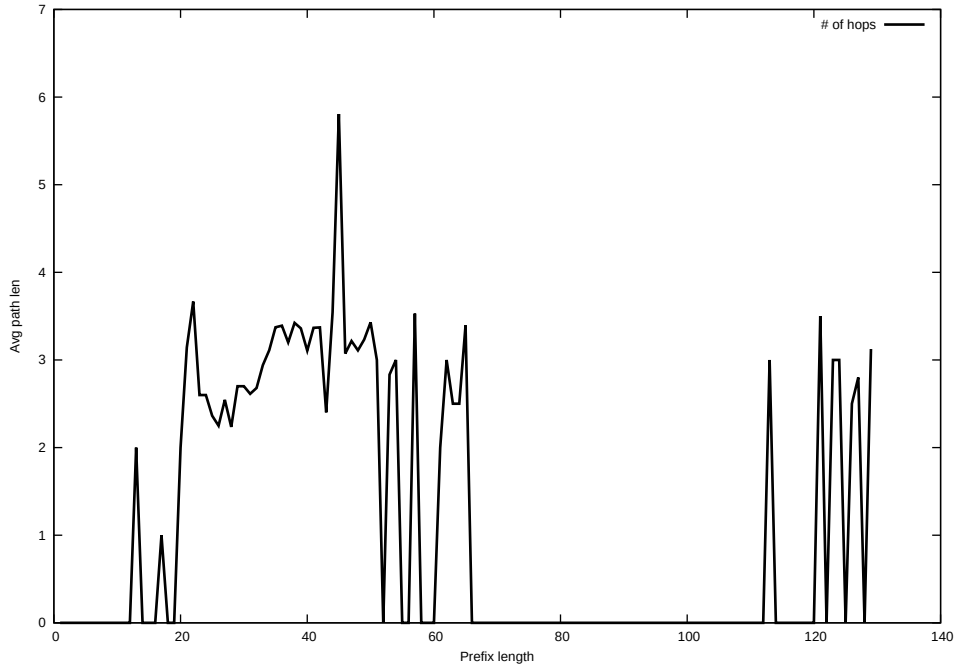
2013-09-22



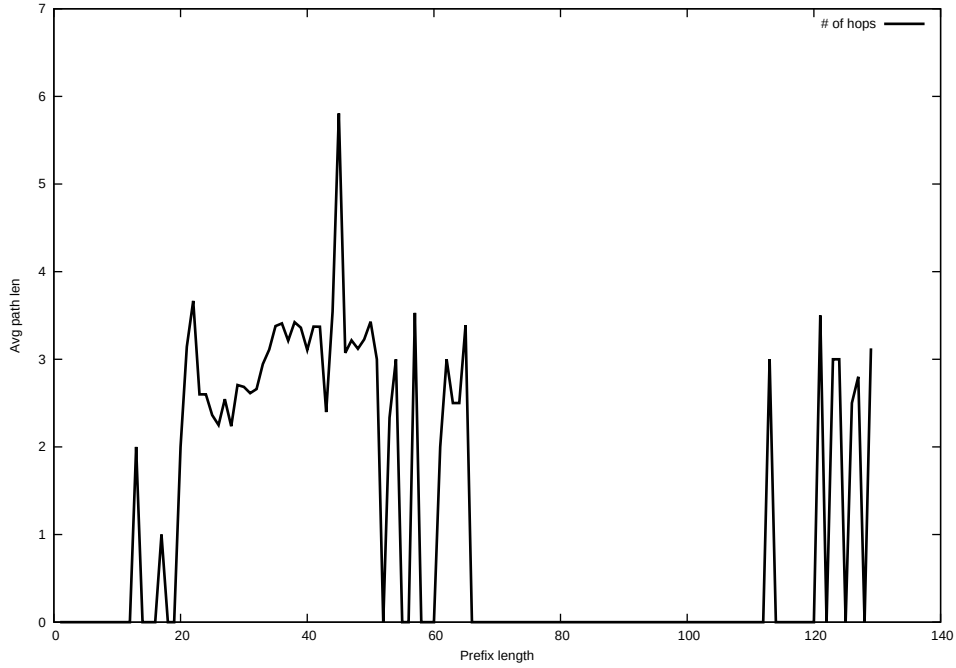
2013-09-23



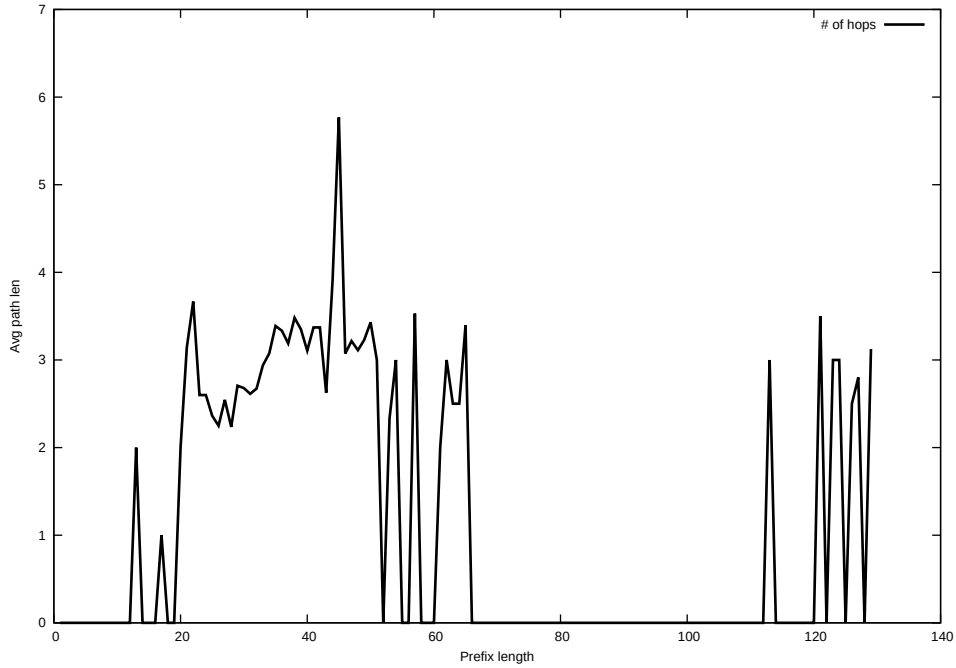
2013-09-24



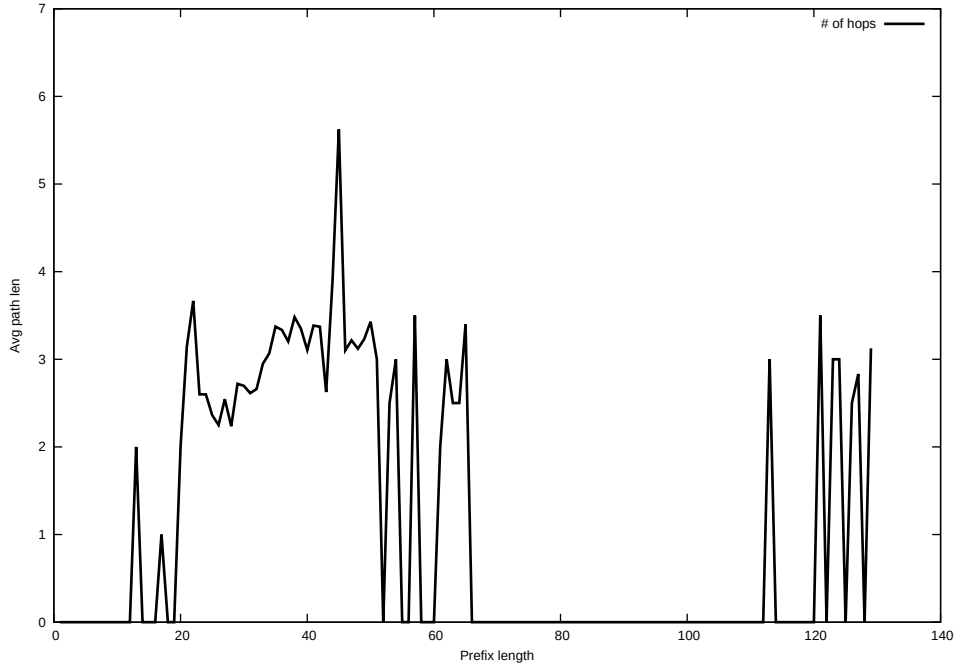
2013-09-25



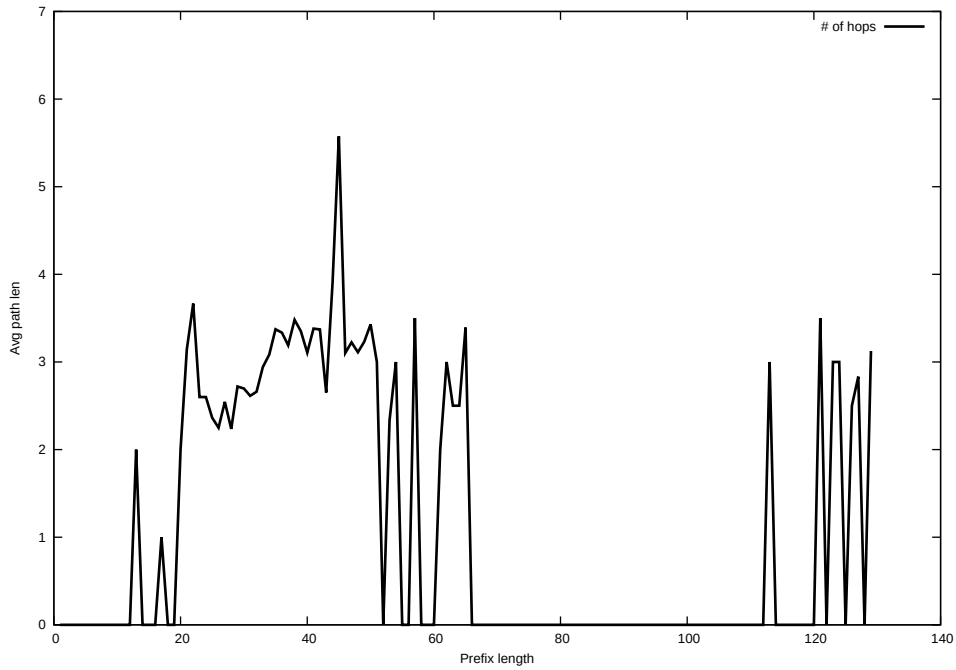
2013-09-26



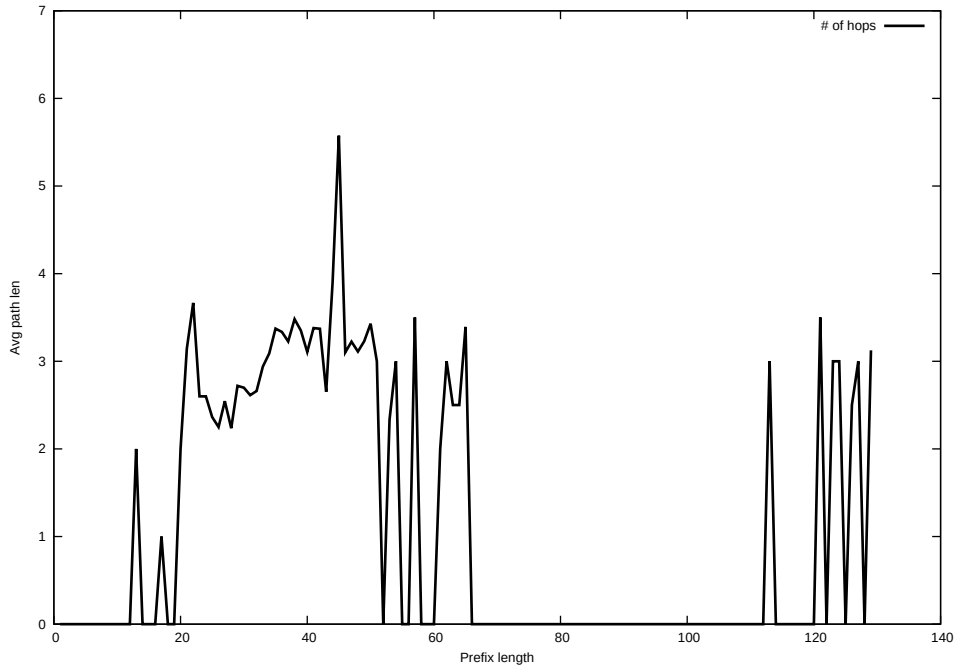
2013-09-27



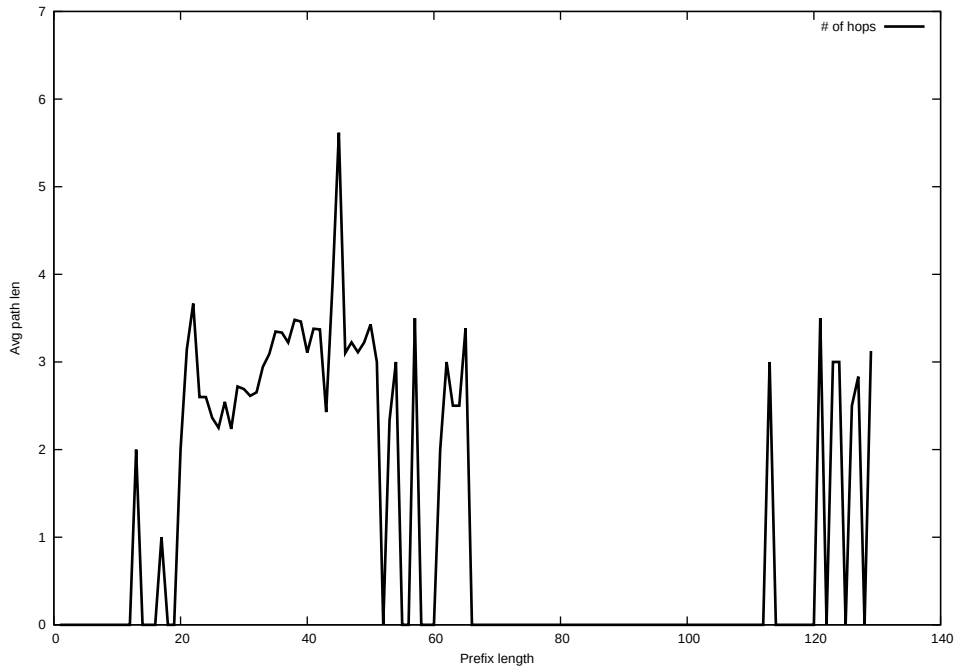
2013-09-28



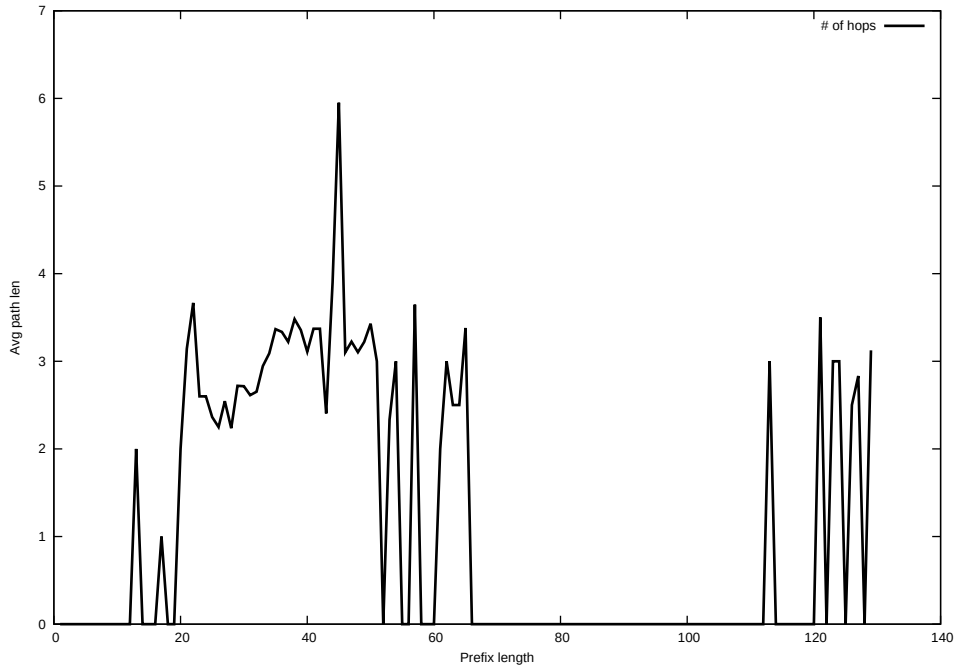
2013-09-29



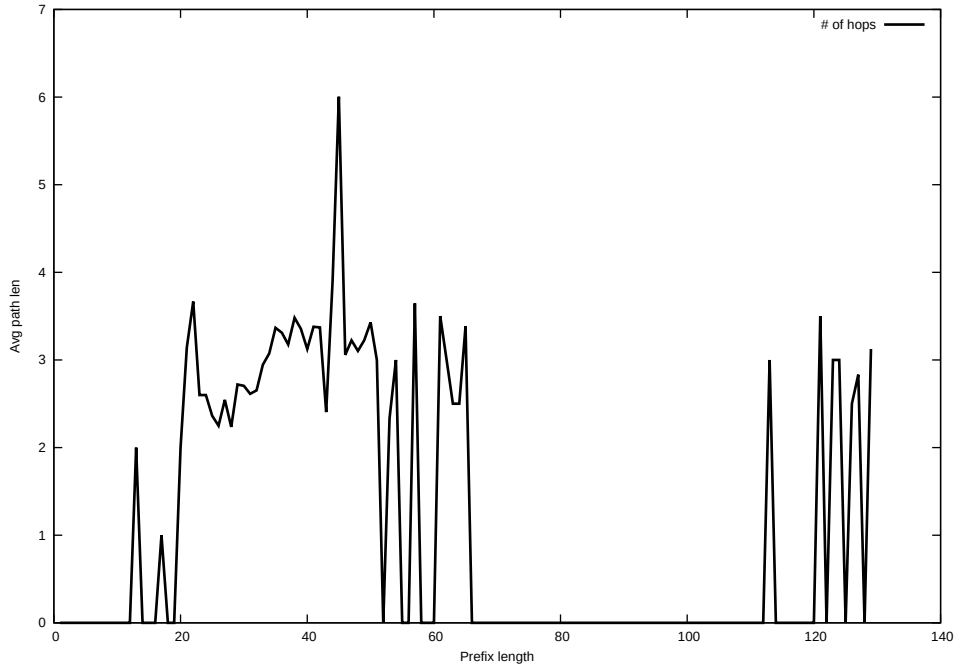
2013-09-30



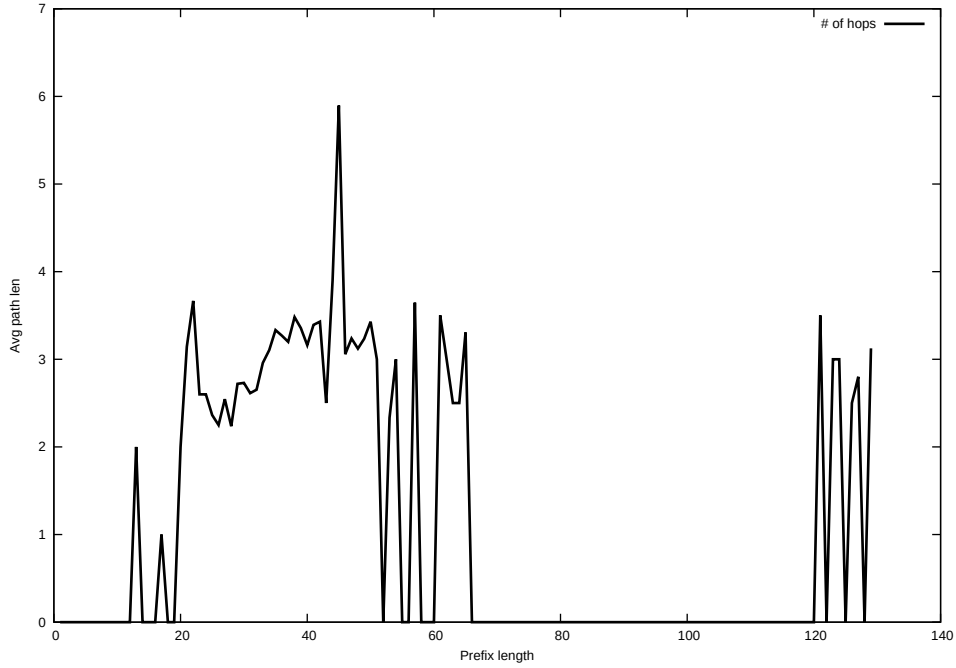
2013-10-01



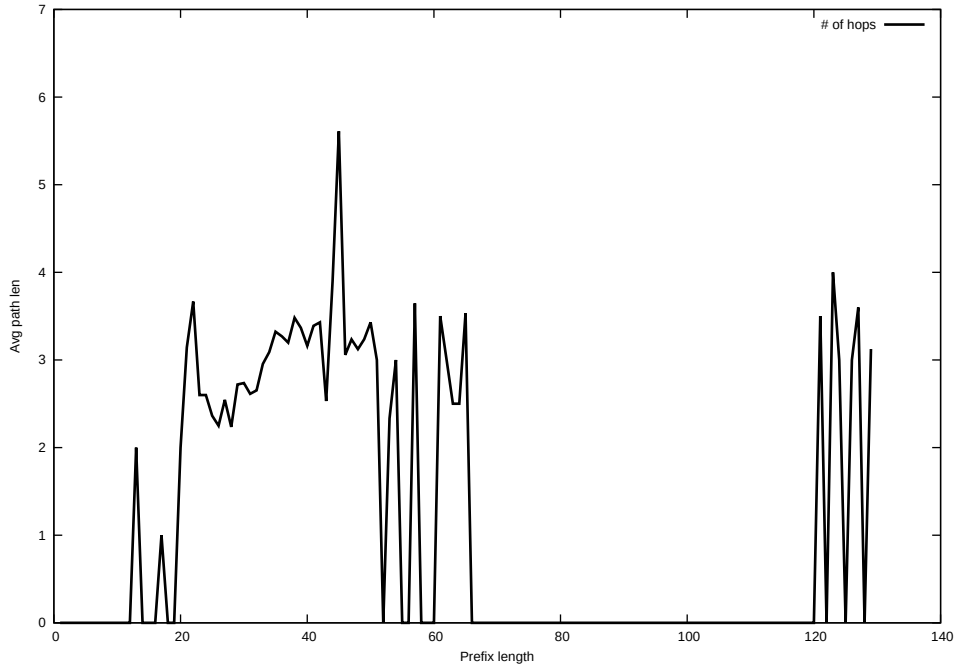
2013-10-02



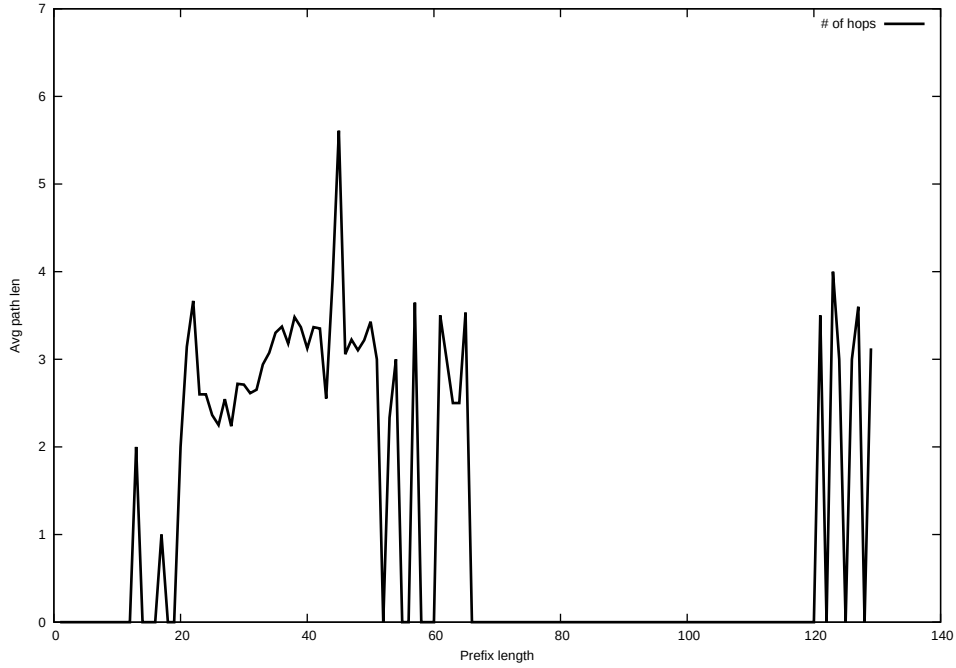
2013-10-03



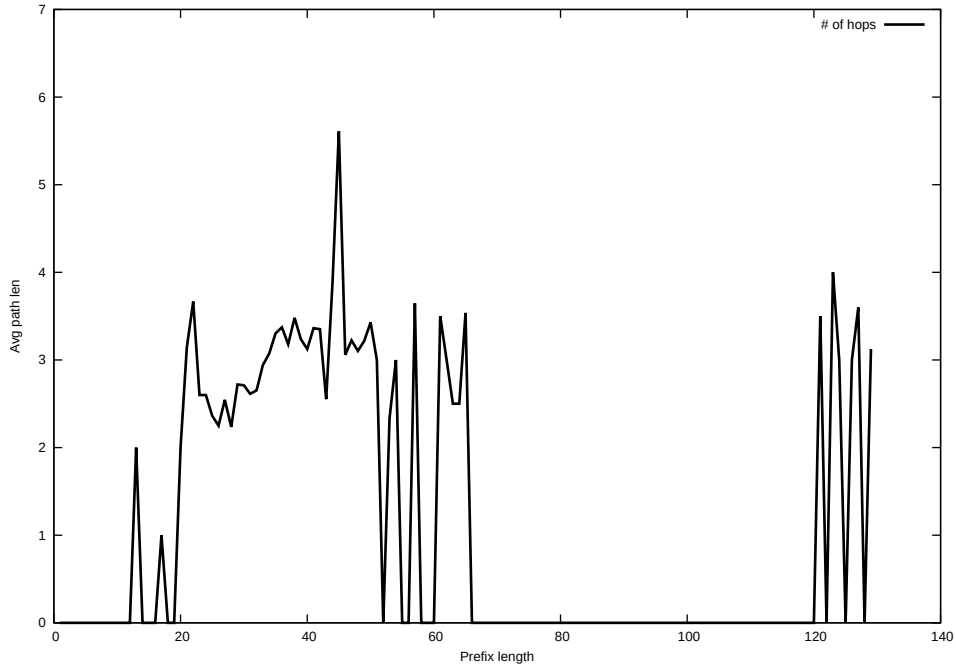
2013-10-04



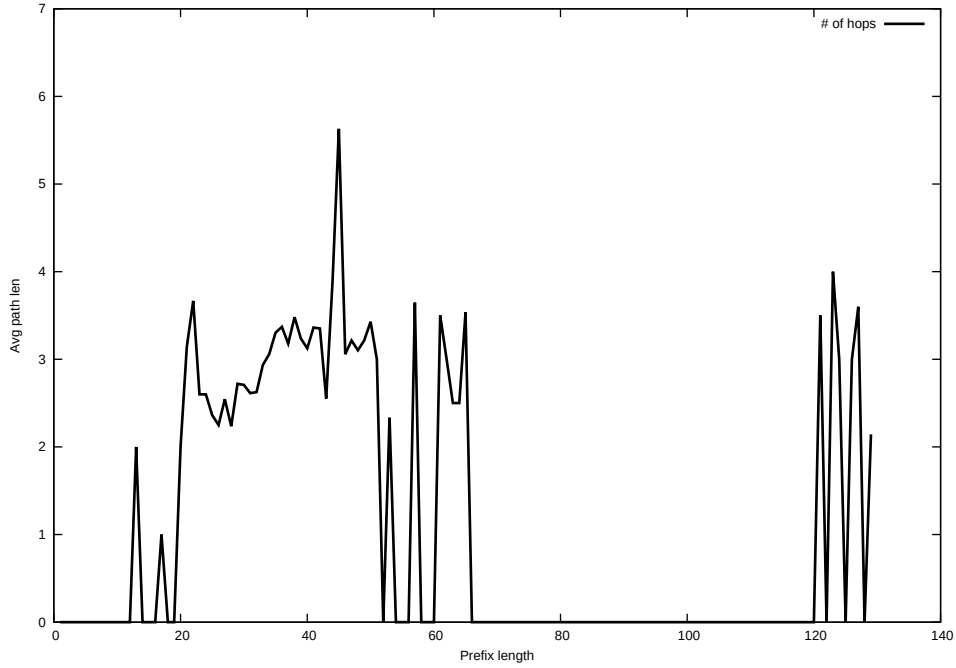
2013-10-05



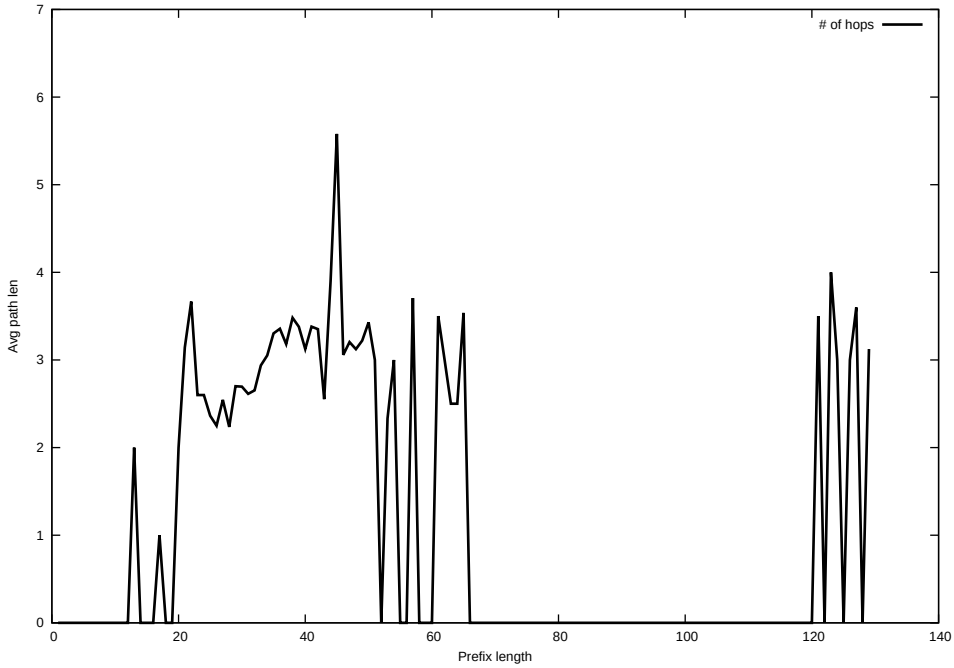
2013-10-06



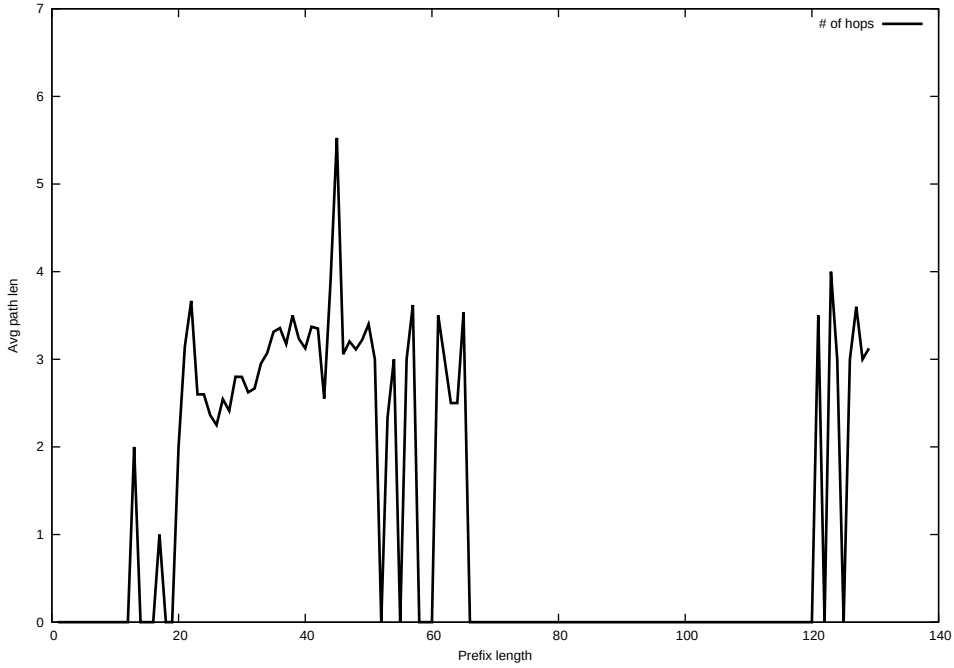
2013-10-07



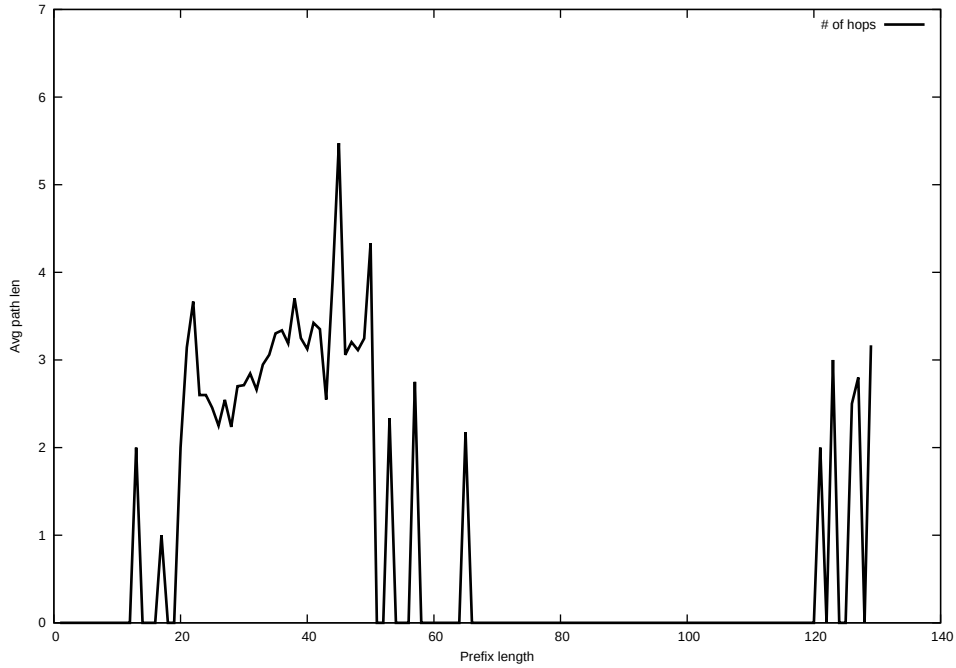
2013-10-08



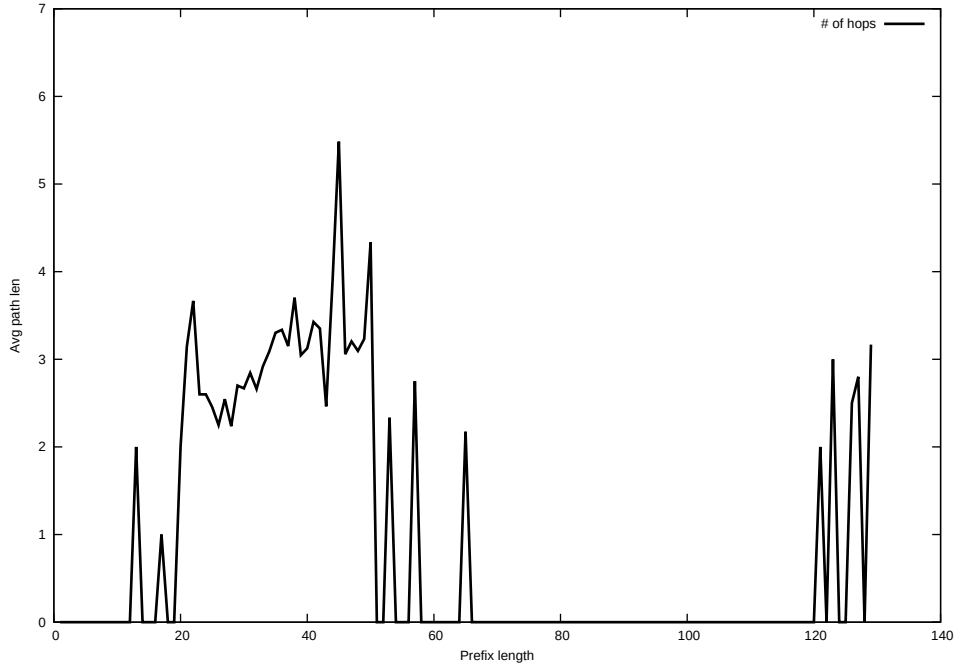
2013-10-09



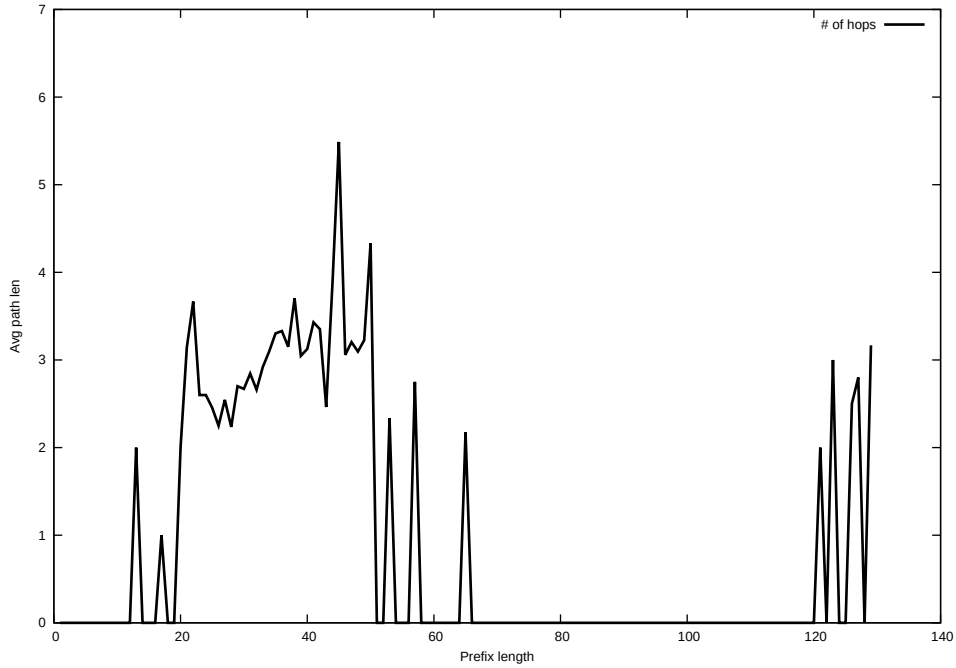
2013-10-10



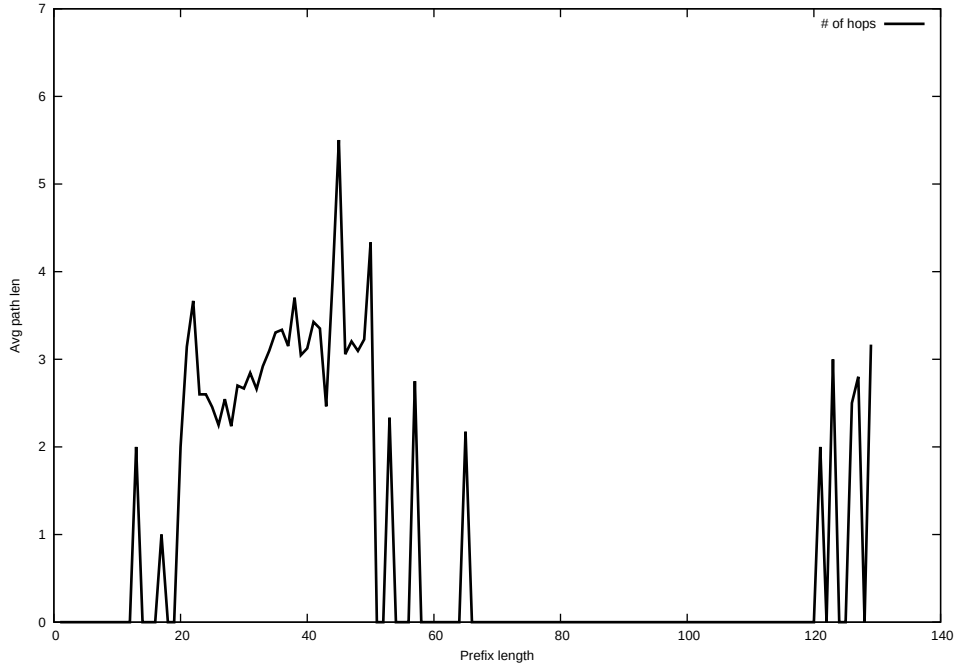
2013-10-11



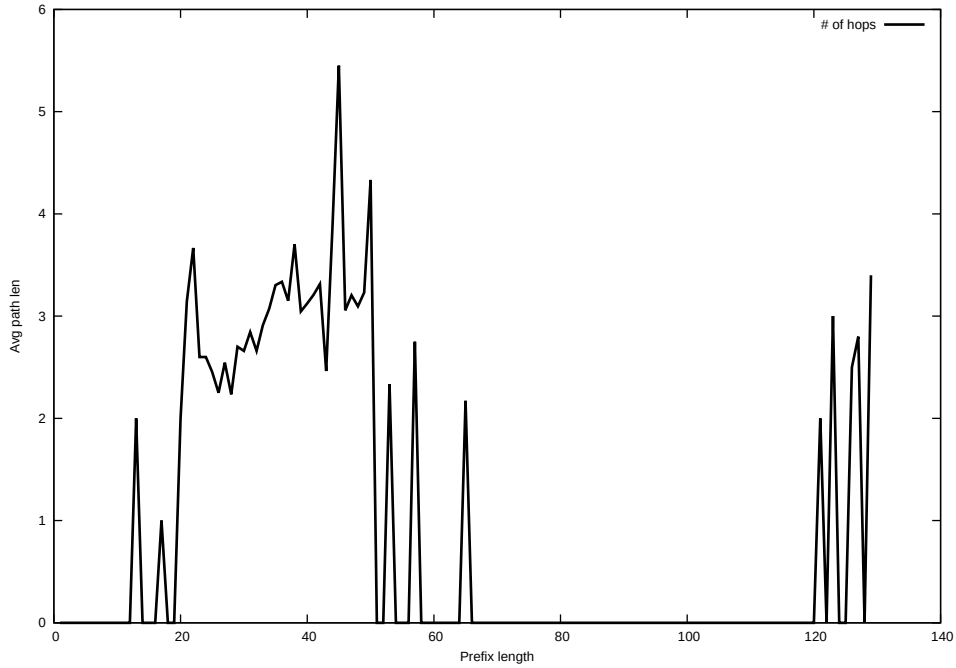
2013-10-12



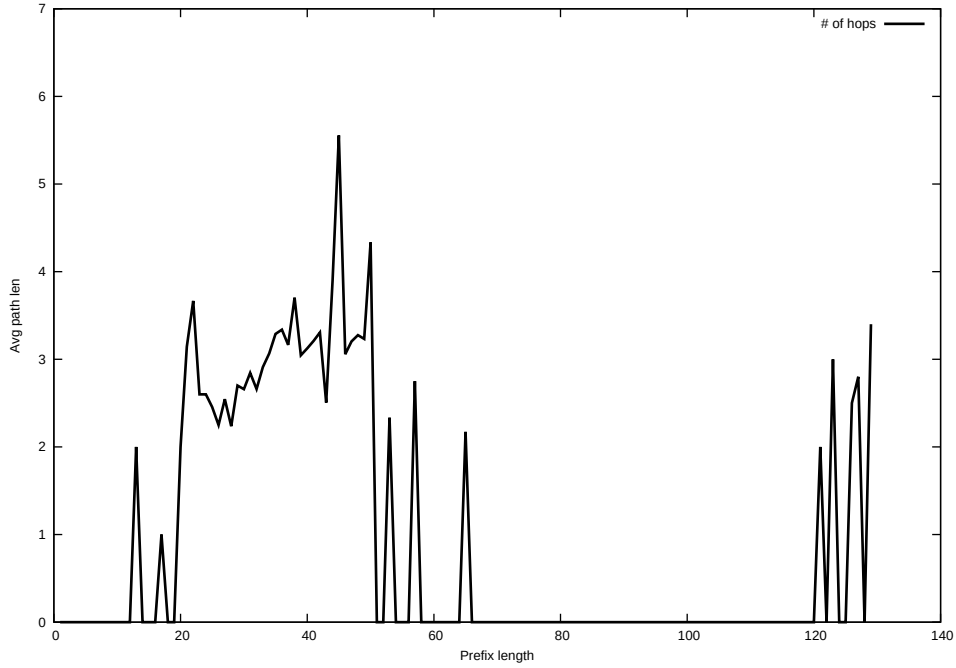
2013-10-13



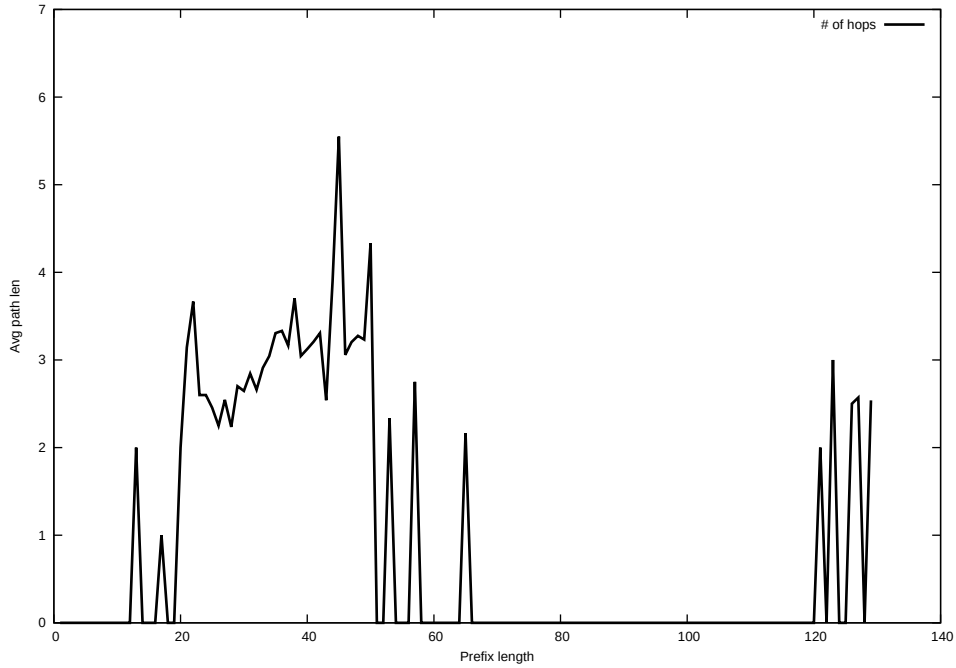
2013-10-14



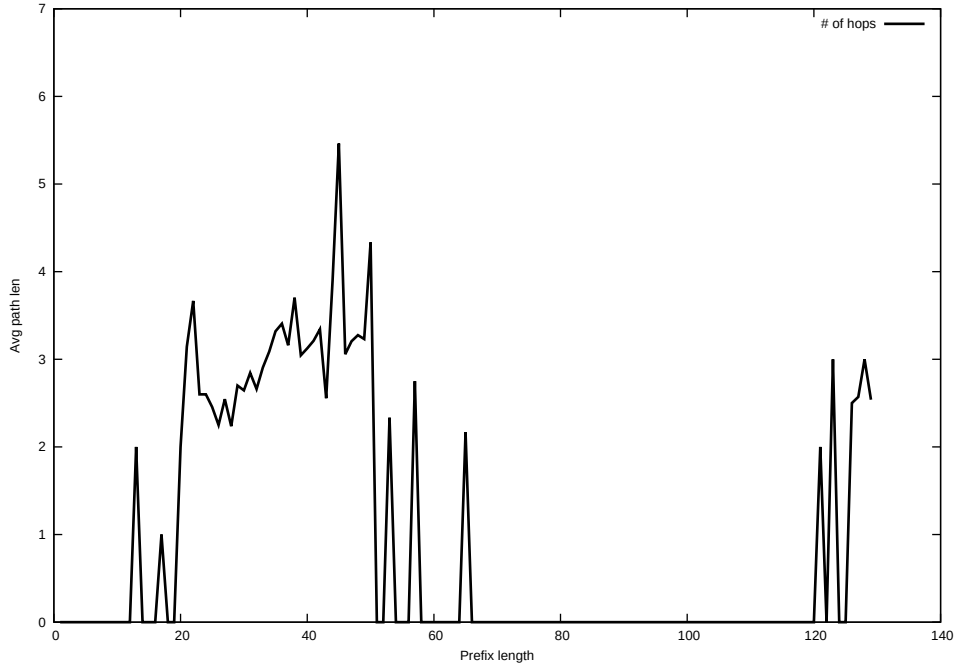
2013-10-15



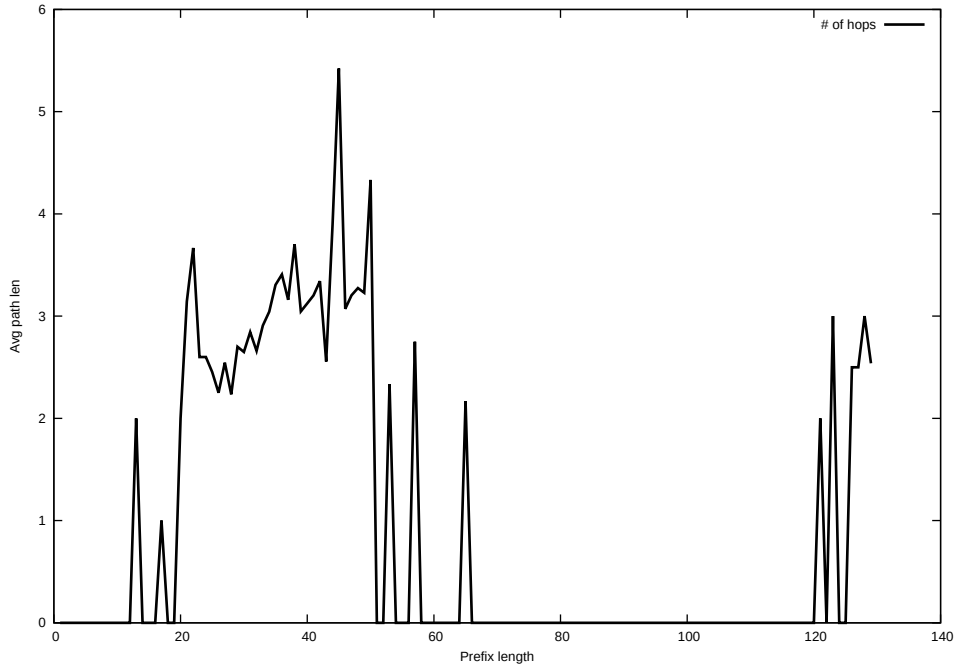
2013-10-16



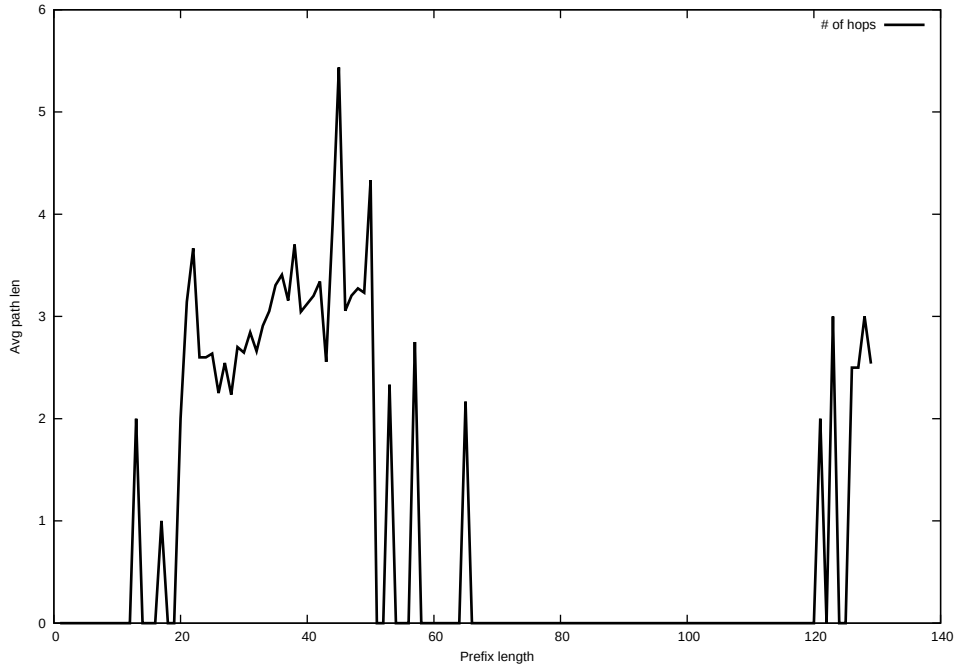
2013-10-17



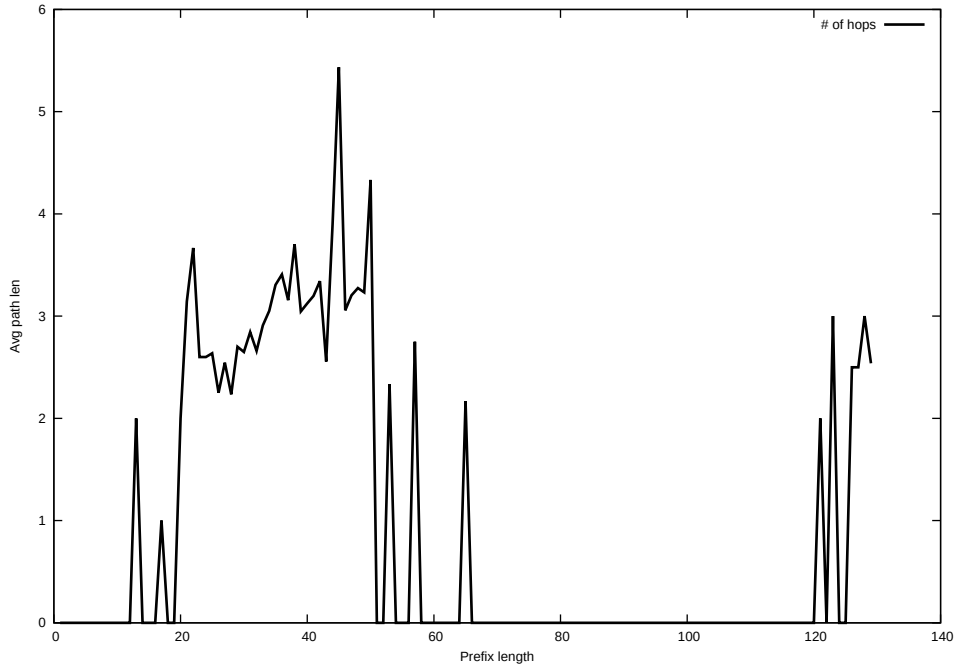
2013-10-18



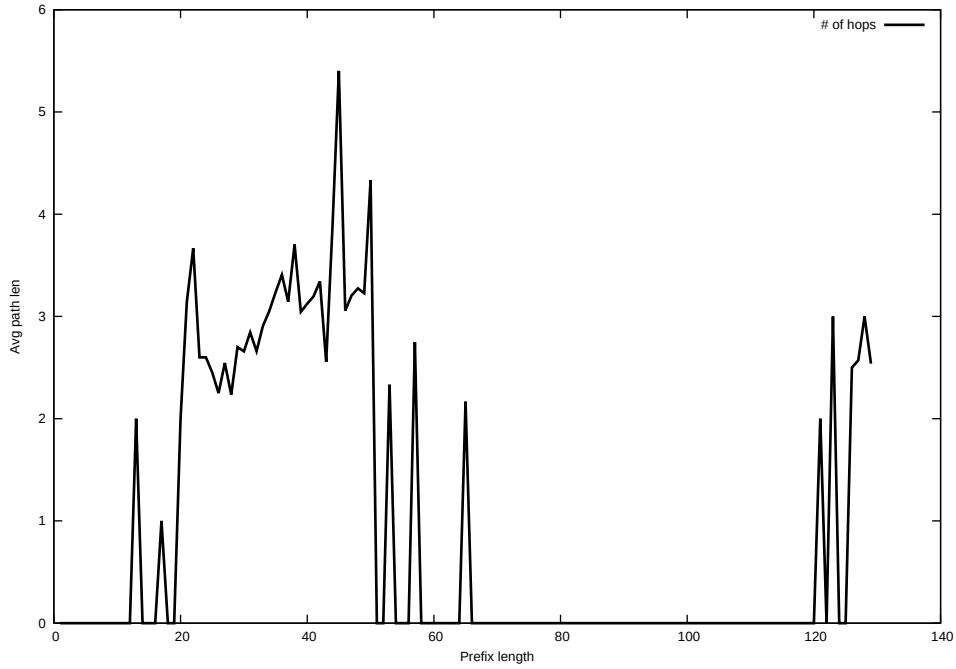
2013-10-19



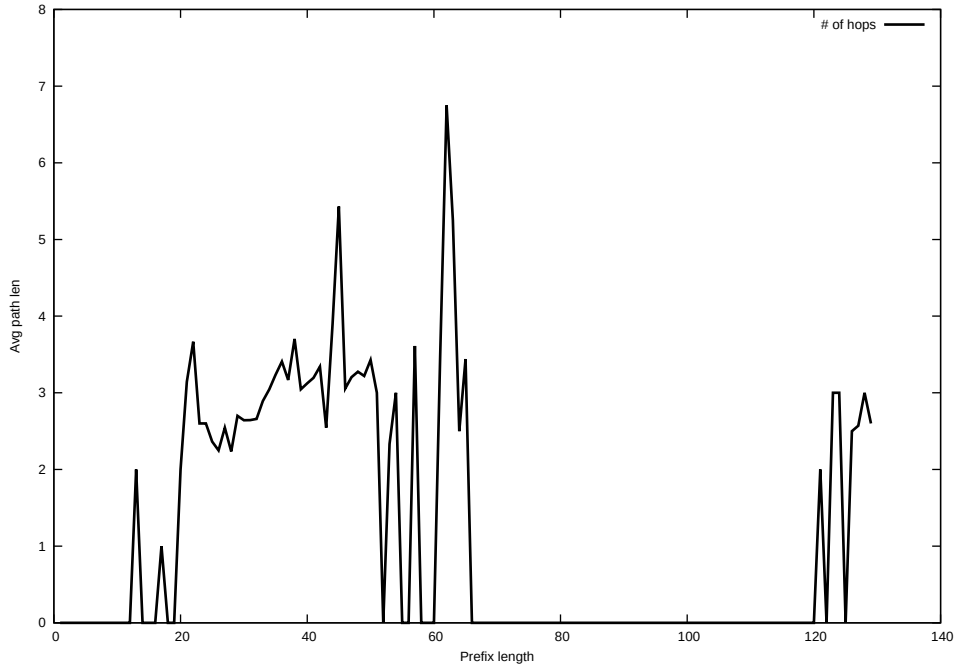
2013-10-20



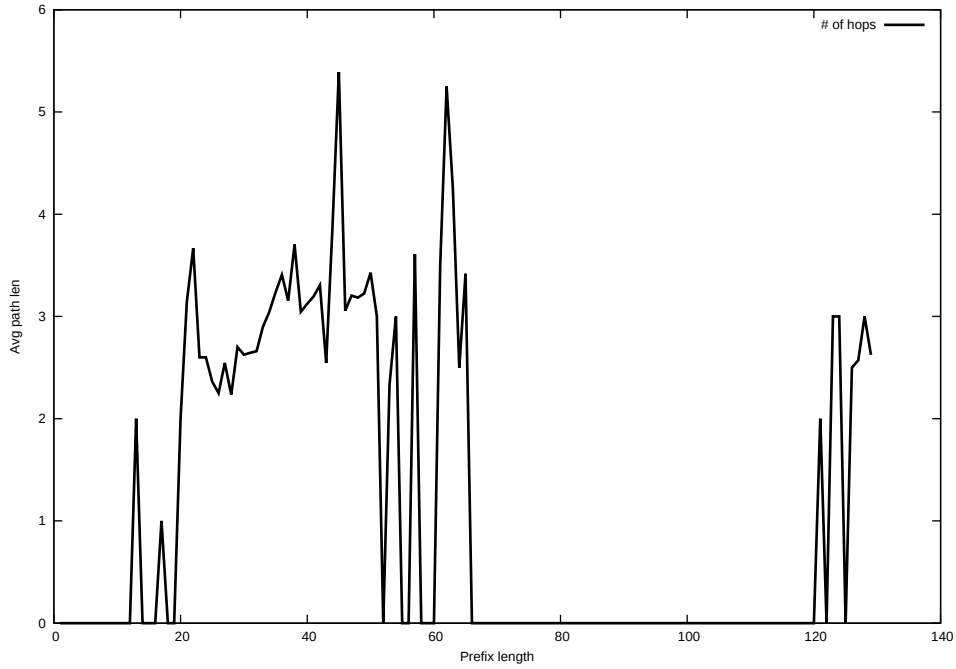
2013-10-21



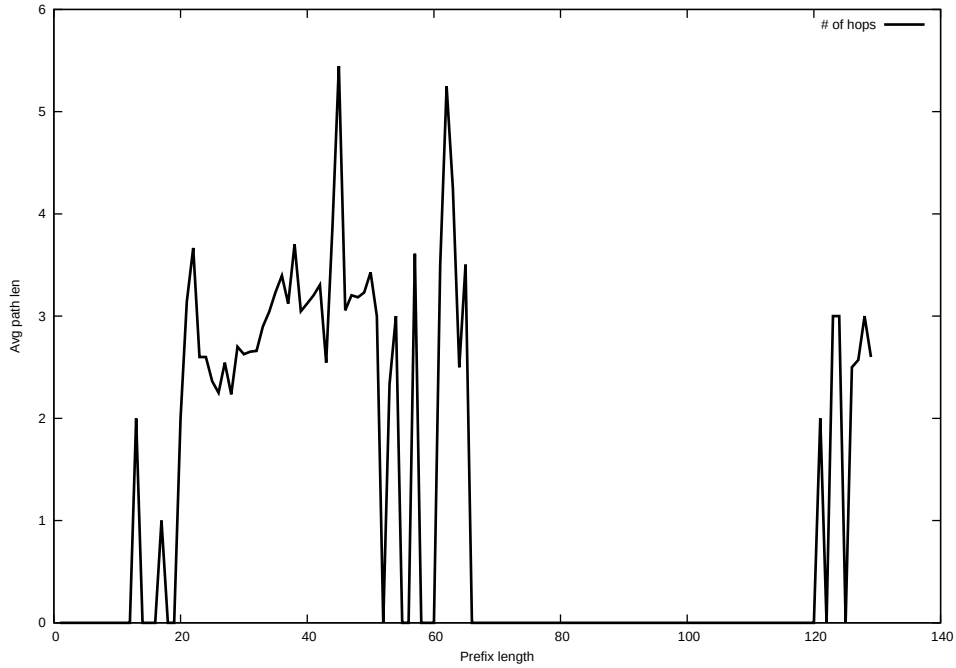
2013-10-22



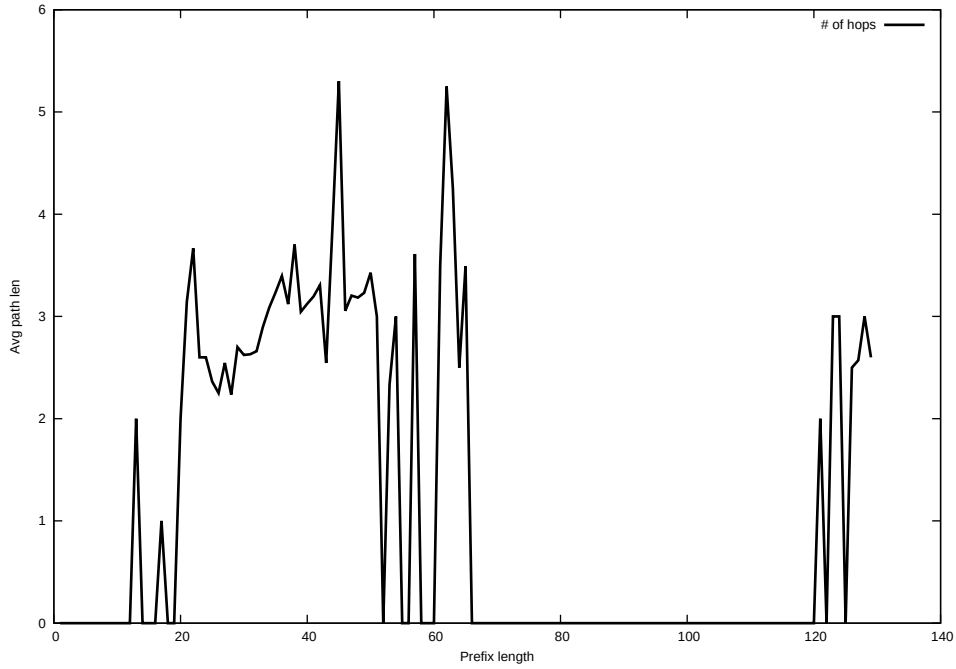
2013-10-23



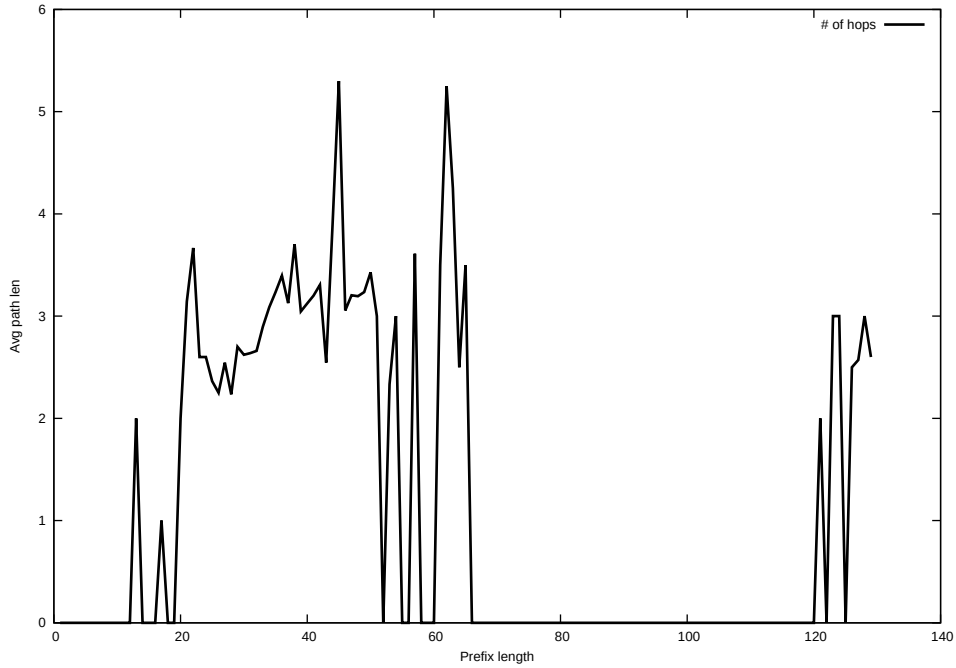
2013-10-24



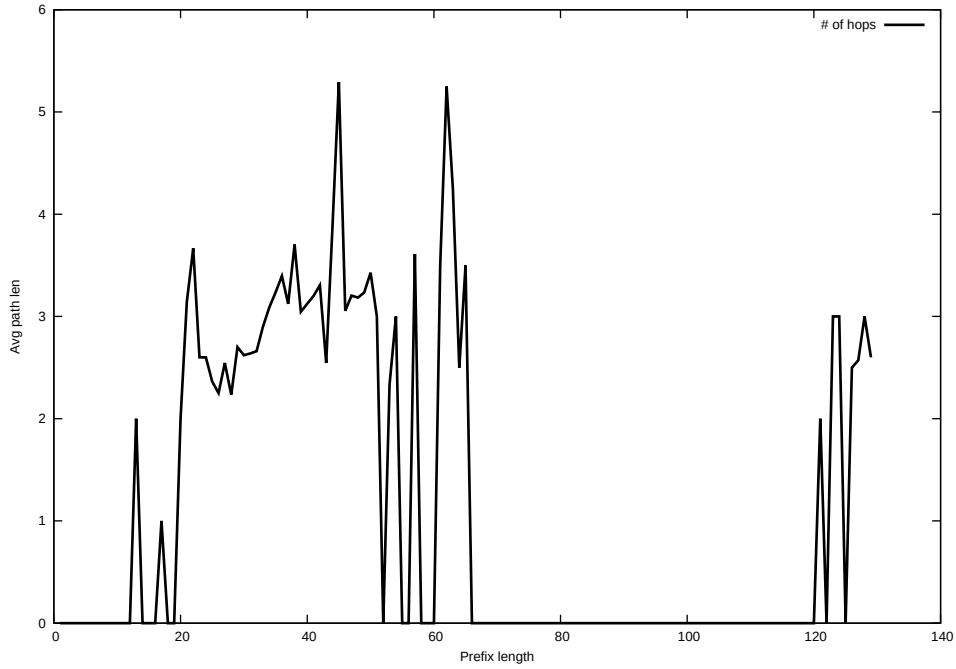
2013-10-25



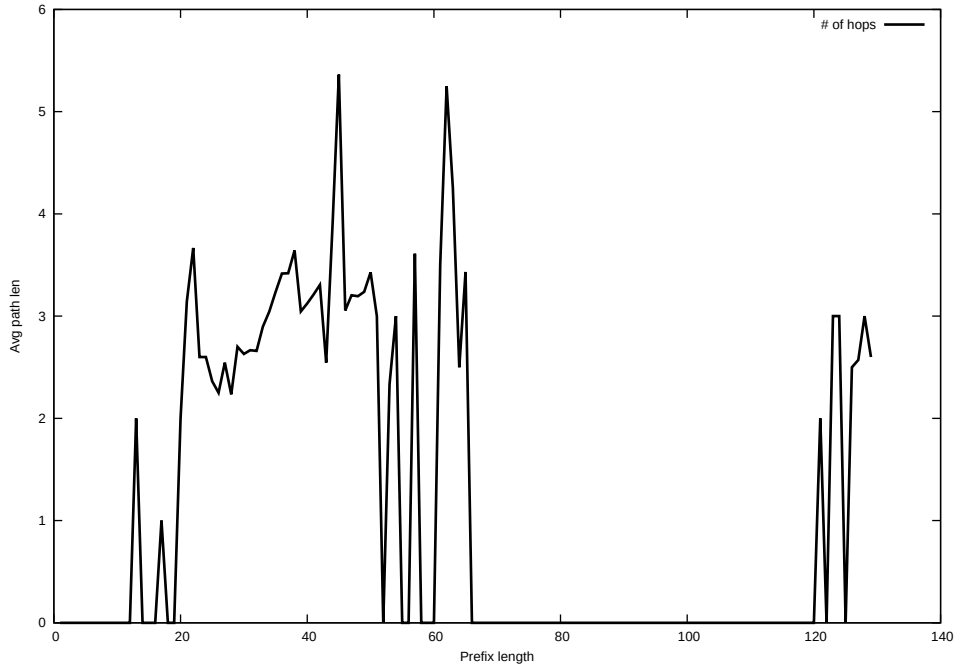
2013-10-26



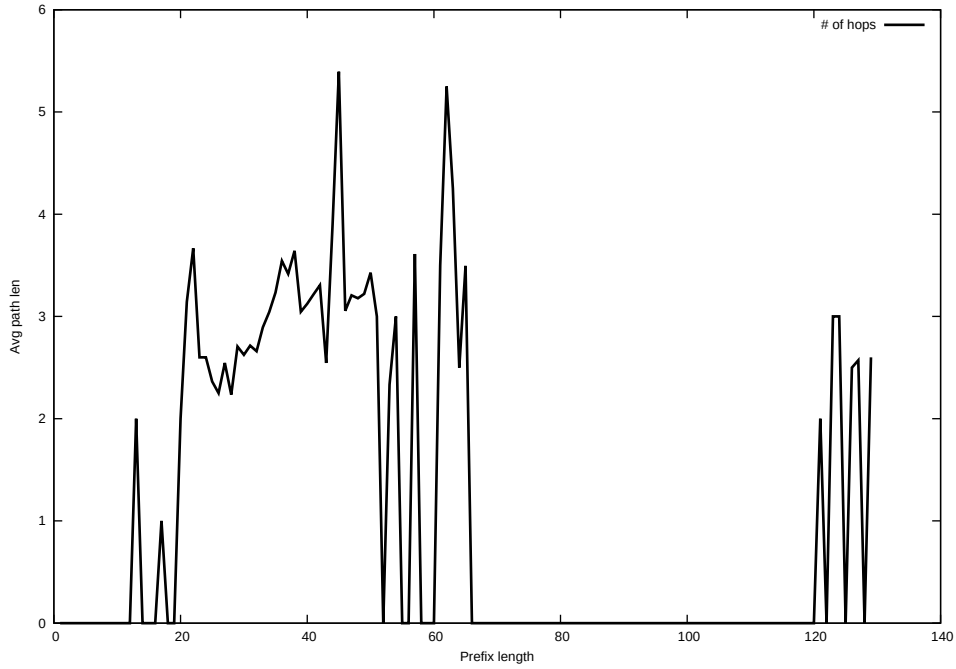
2013-10-27



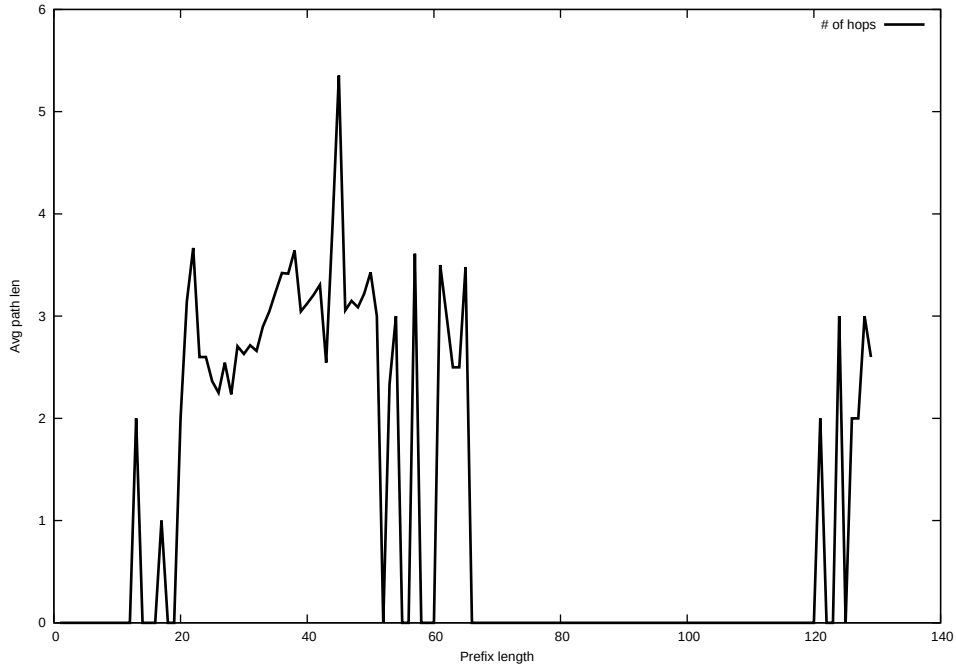
2013-10-28



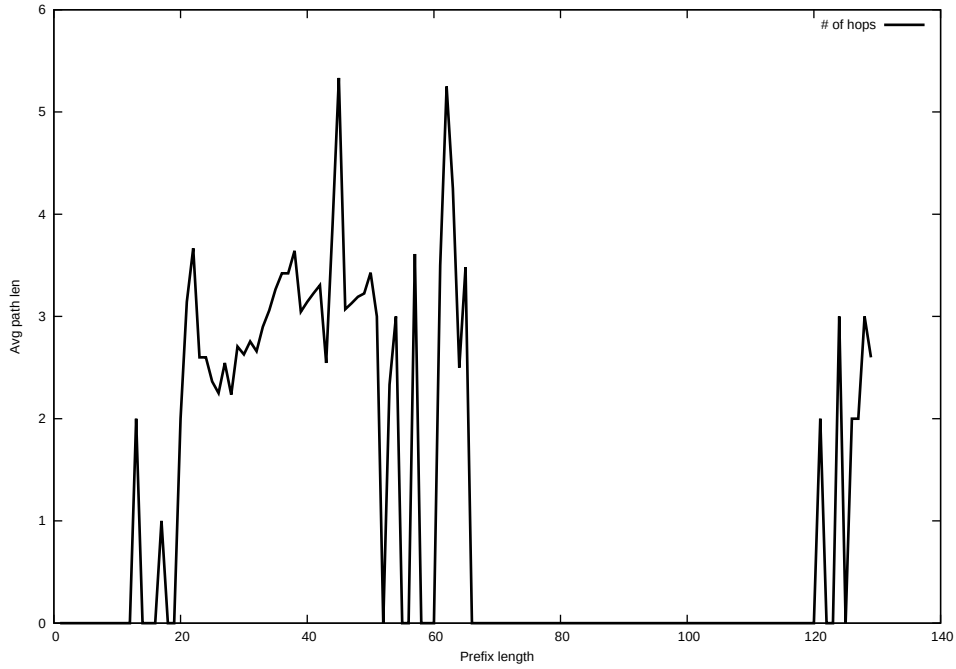
2013-10-29



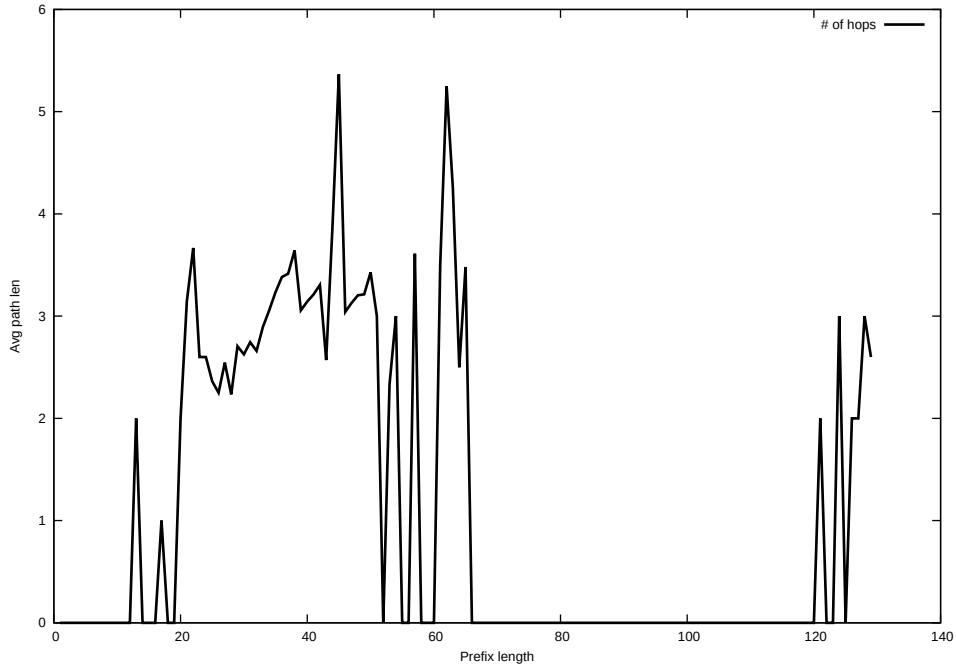
2013-10-30



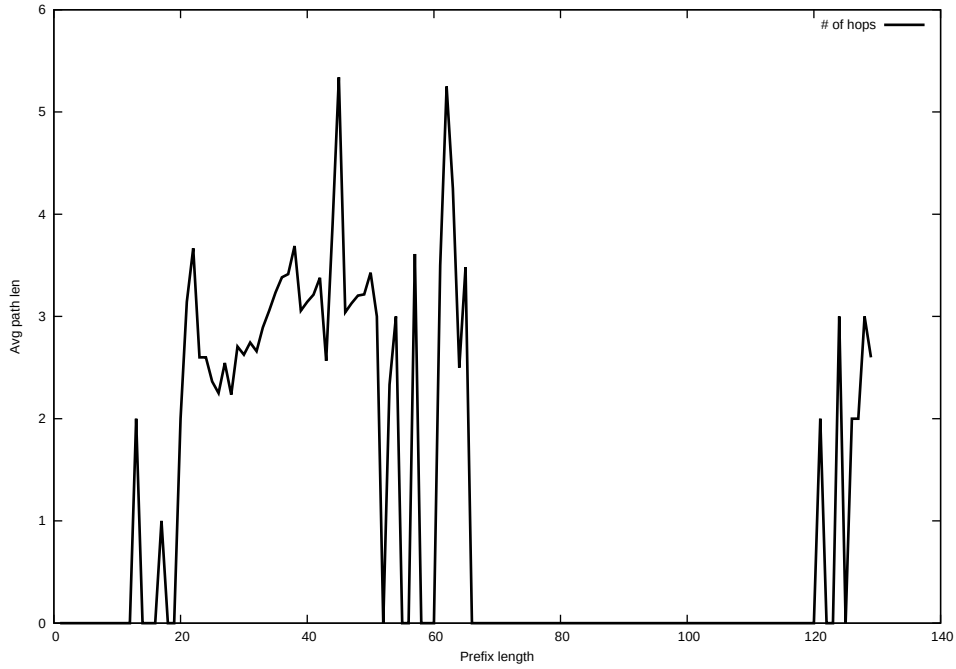
2013-10-31



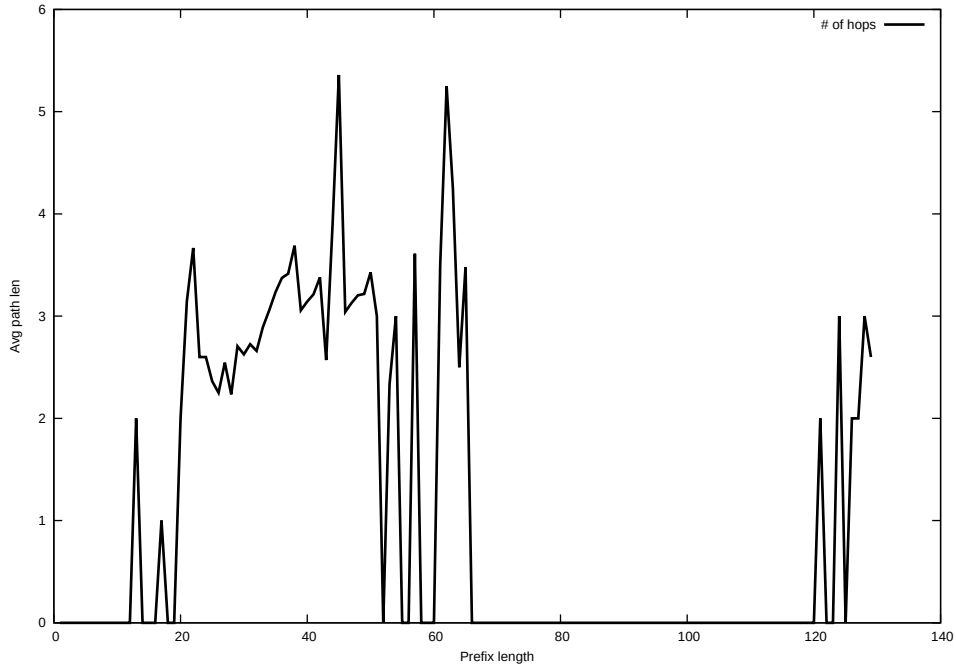
2013-11-01



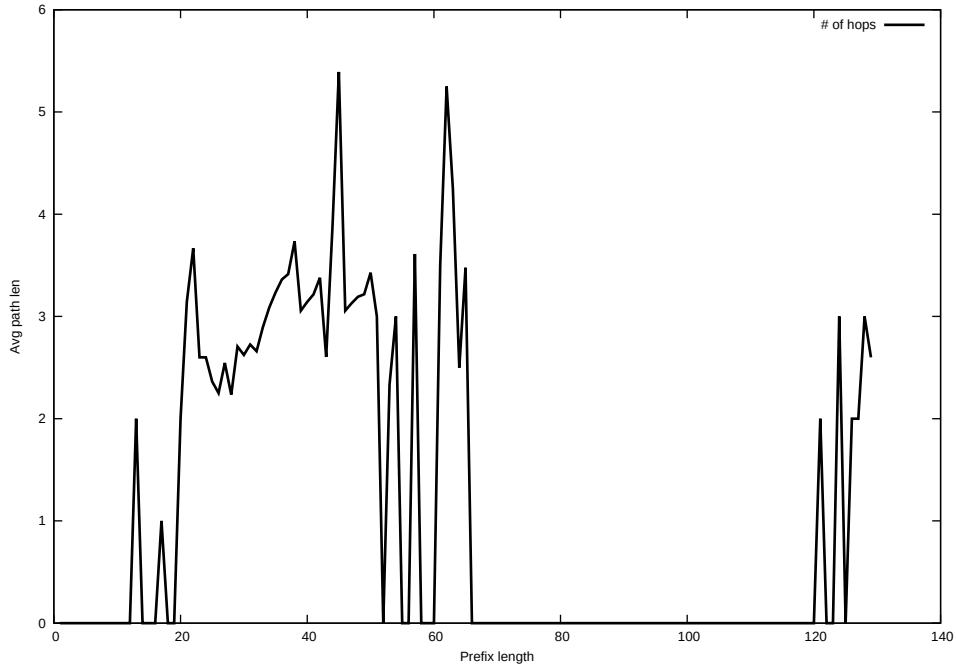
2013-11-02



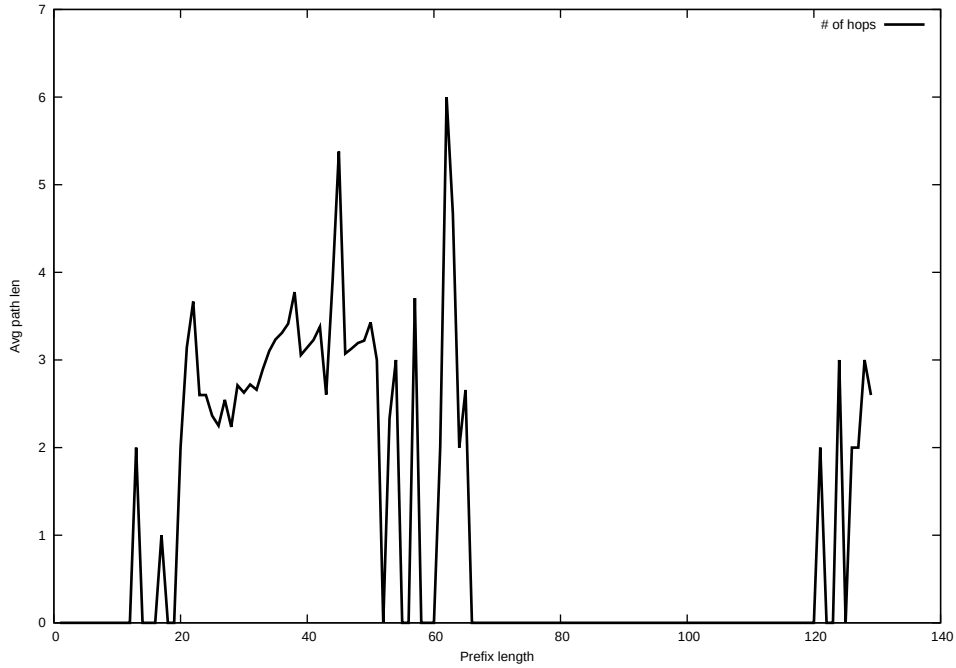
2013-11-03



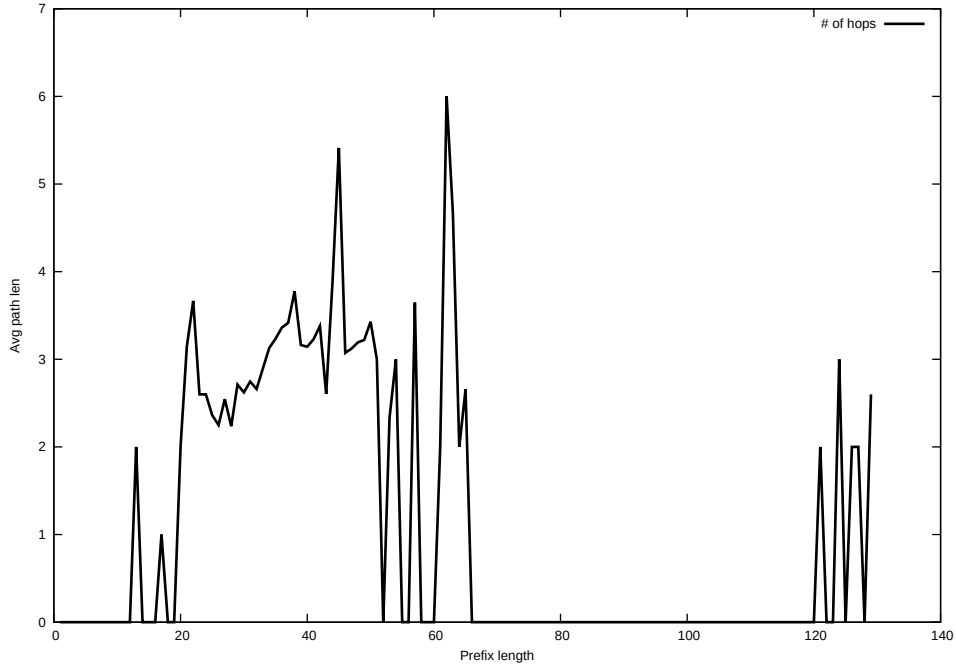
2013-11-04



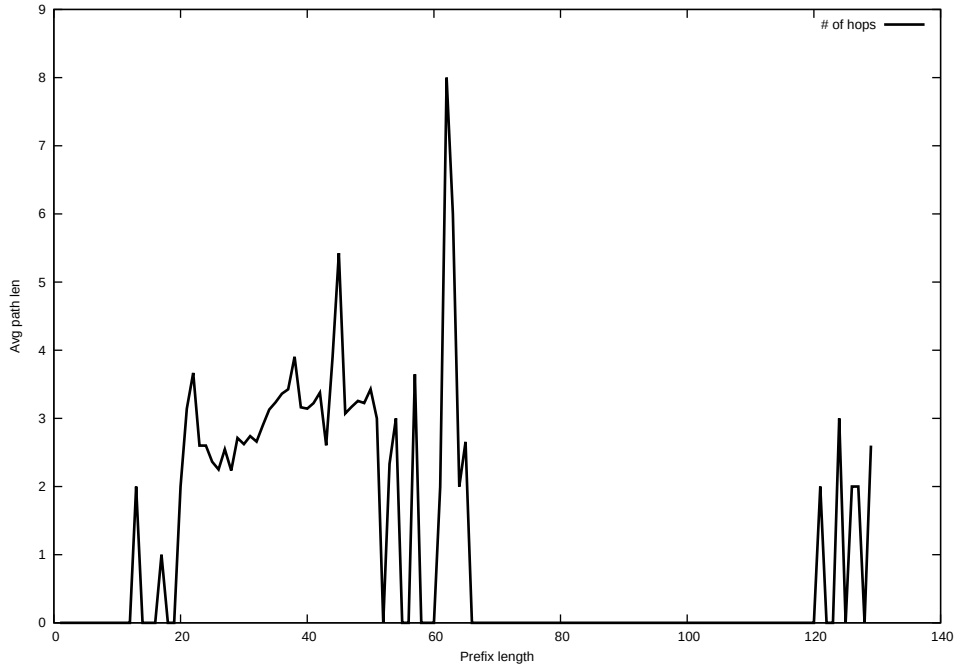
2013-11-05



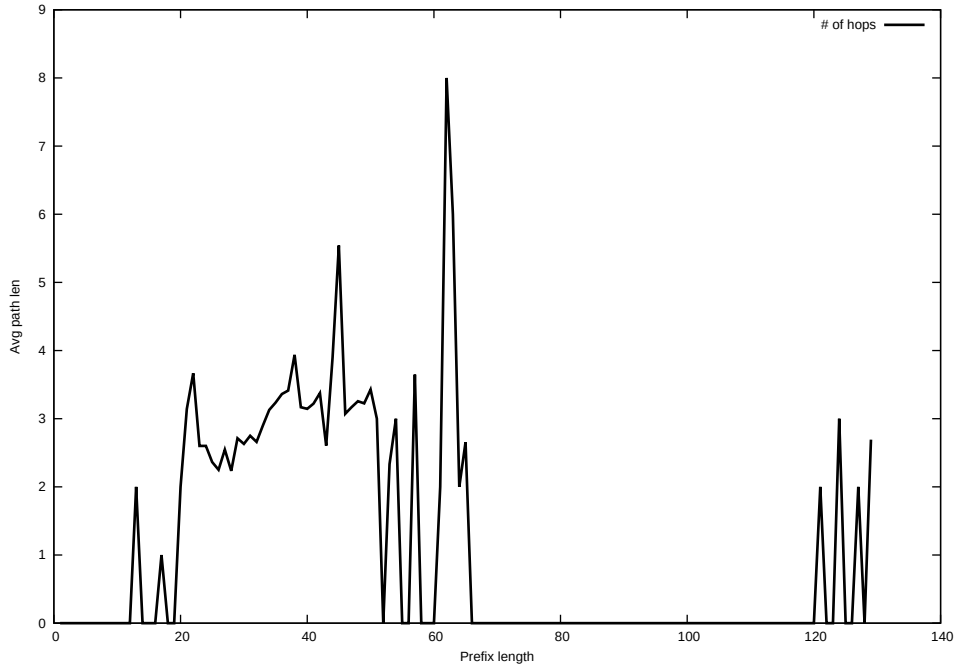
2013-11-06



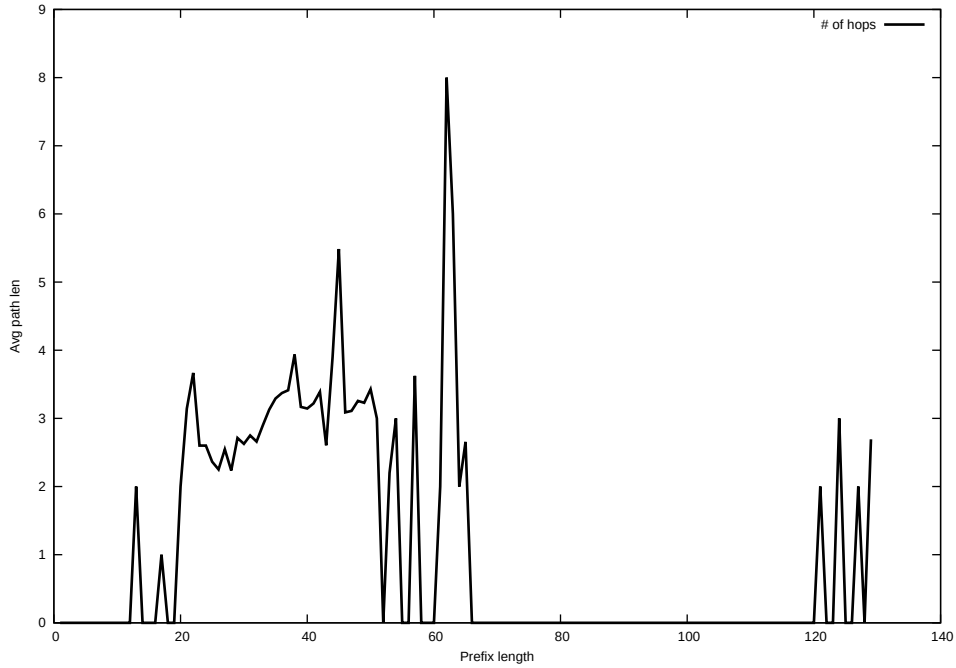
2013-11-07



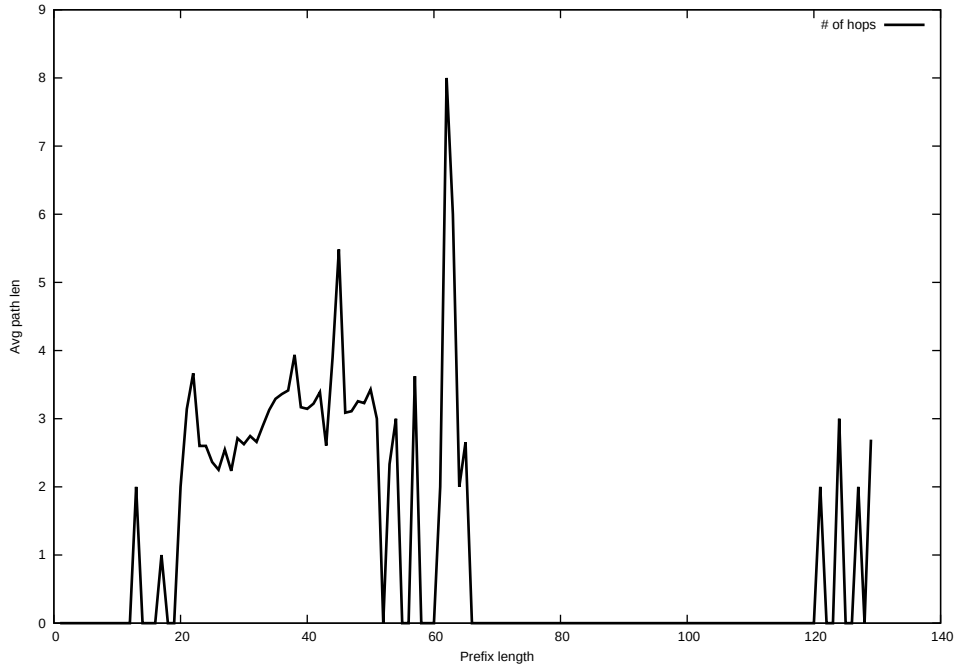
2013-11-08



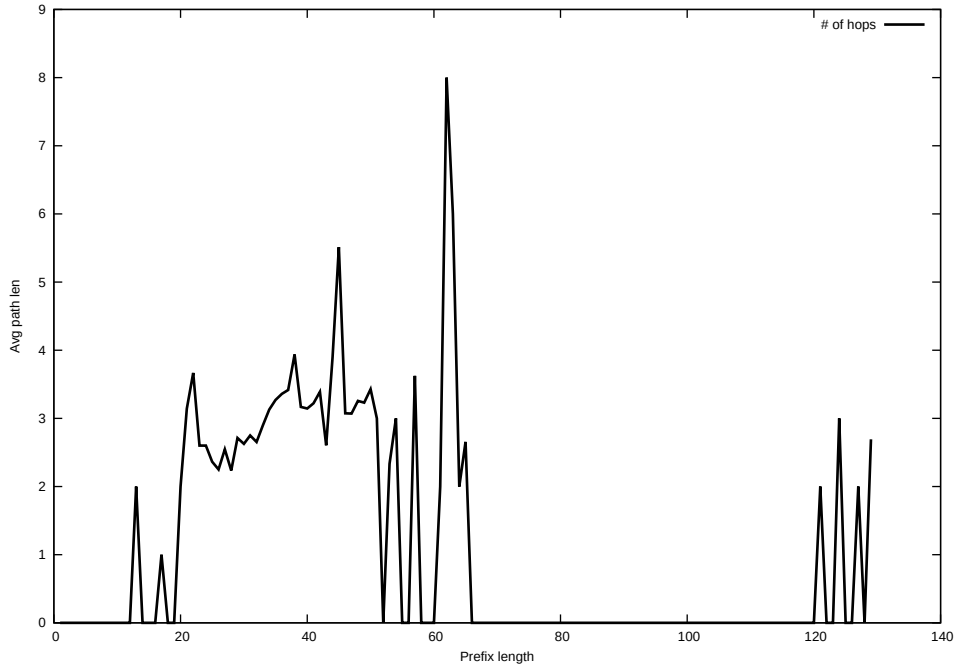
2013-11-09



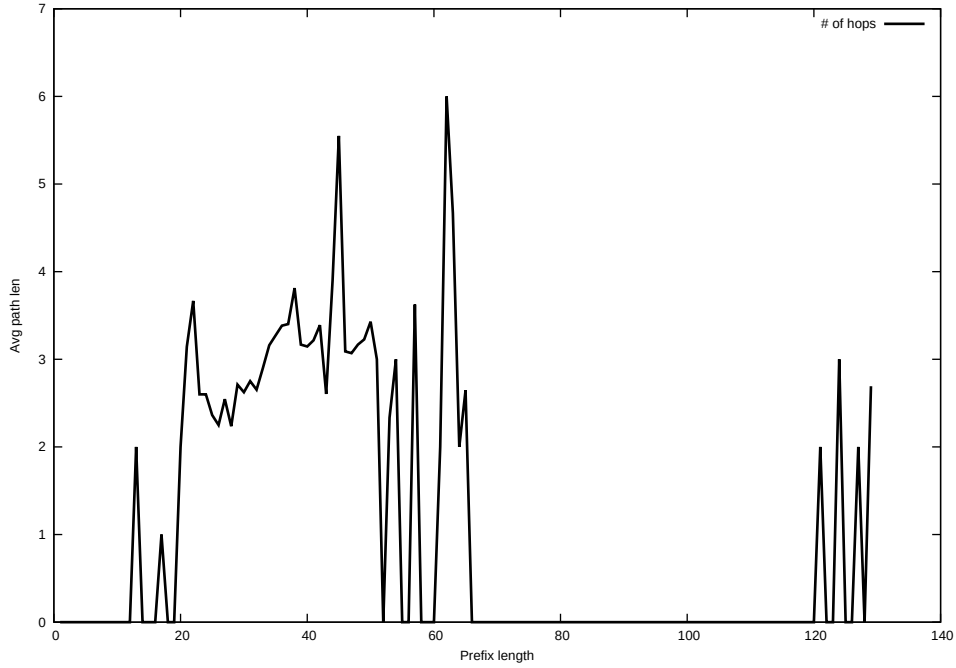
2013-11-10



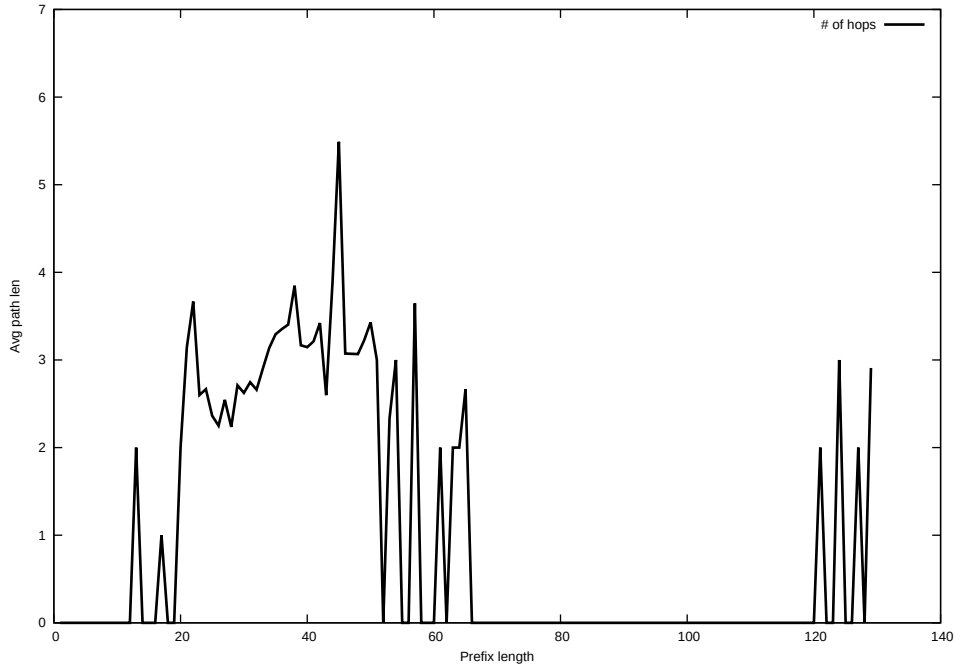
2013-11-11



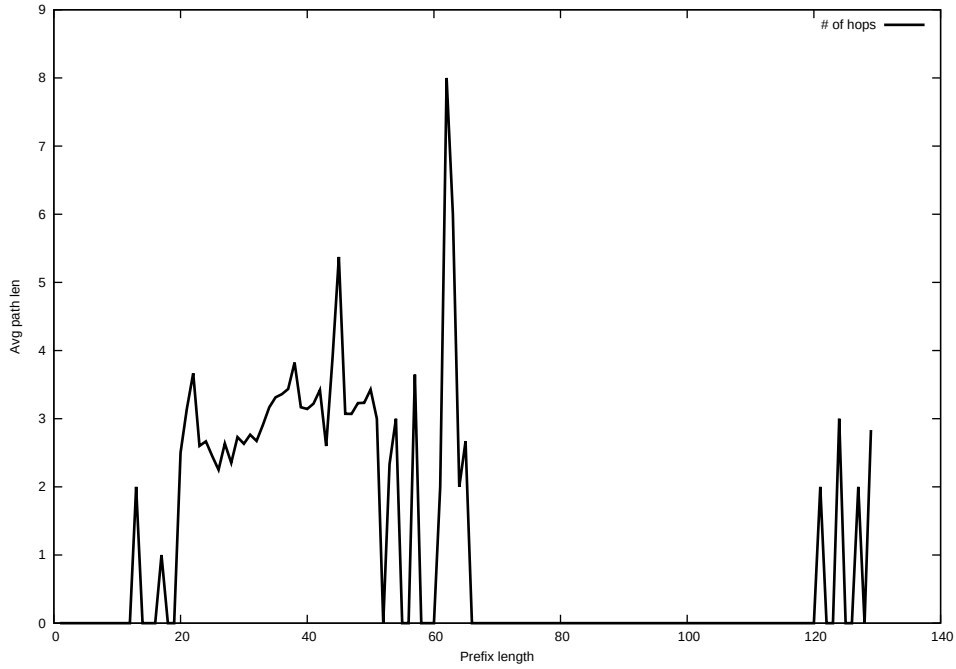
2013-11-12



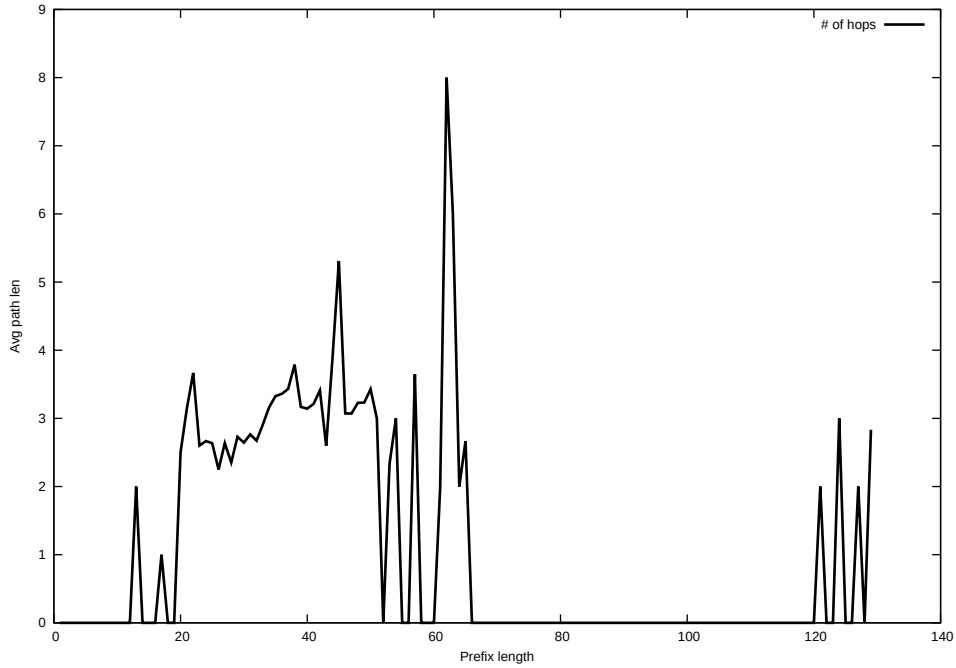
2013-11-13



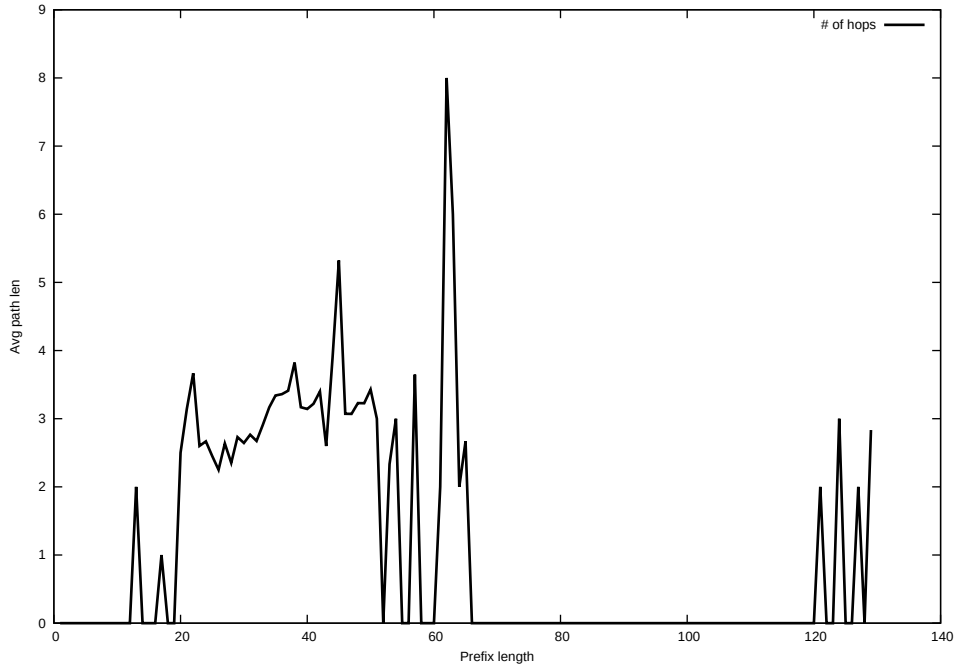
2013-11-14



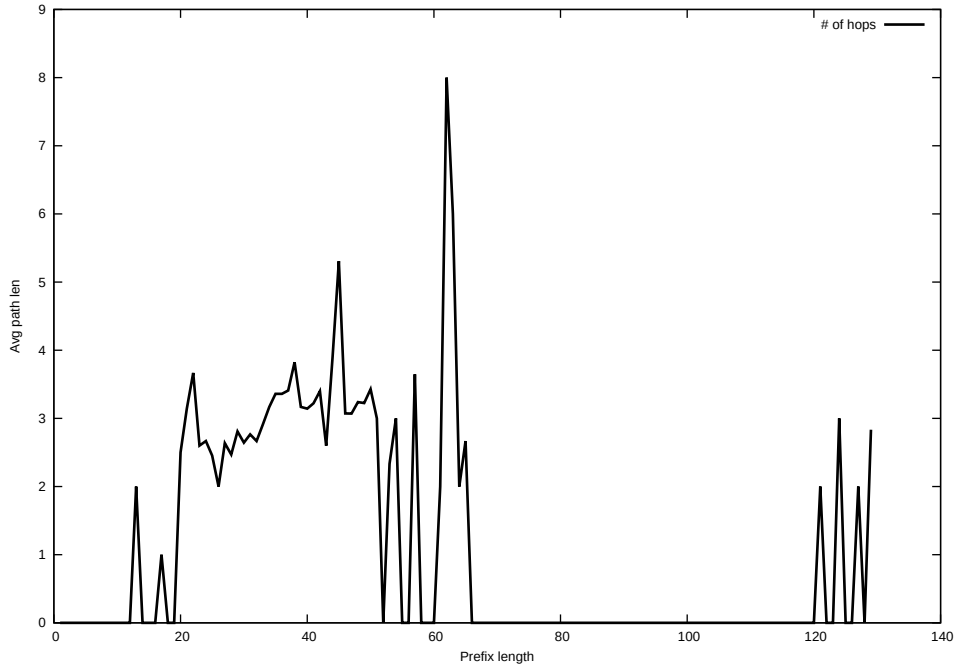
2013-11-15



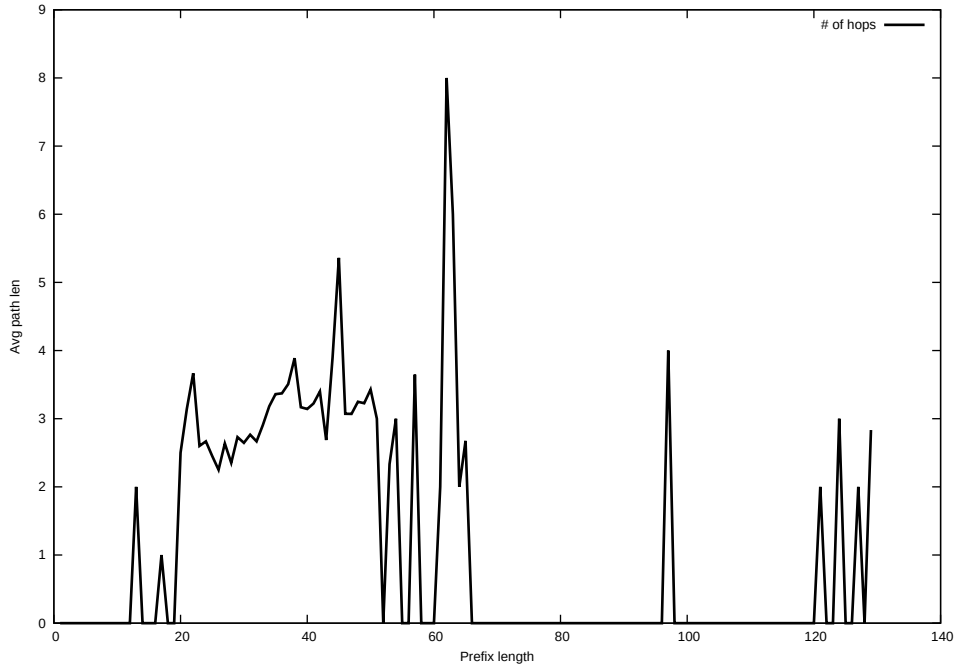
2013-11-16



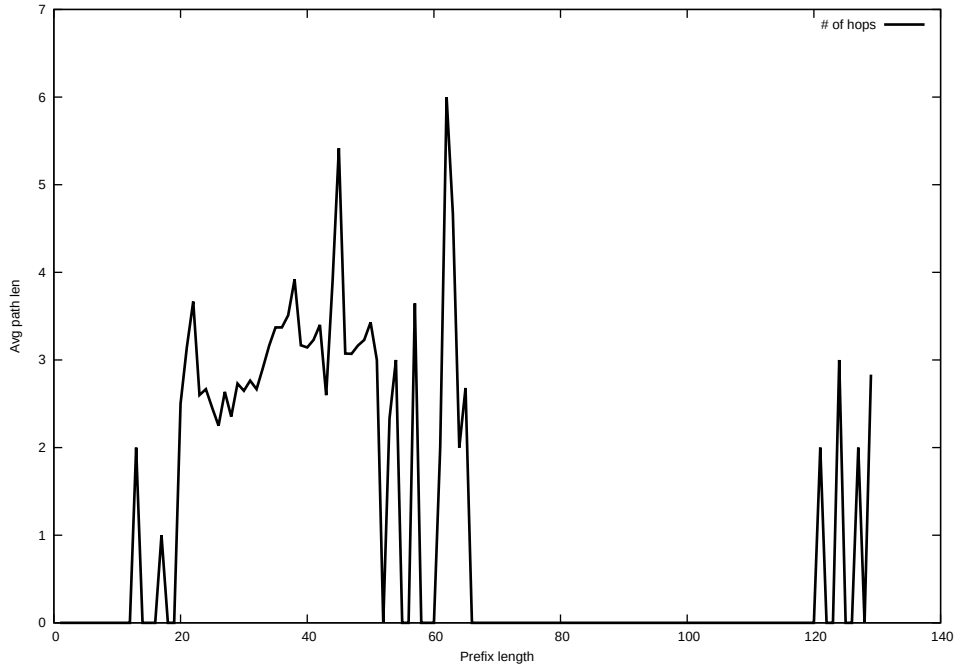
2013-11-17



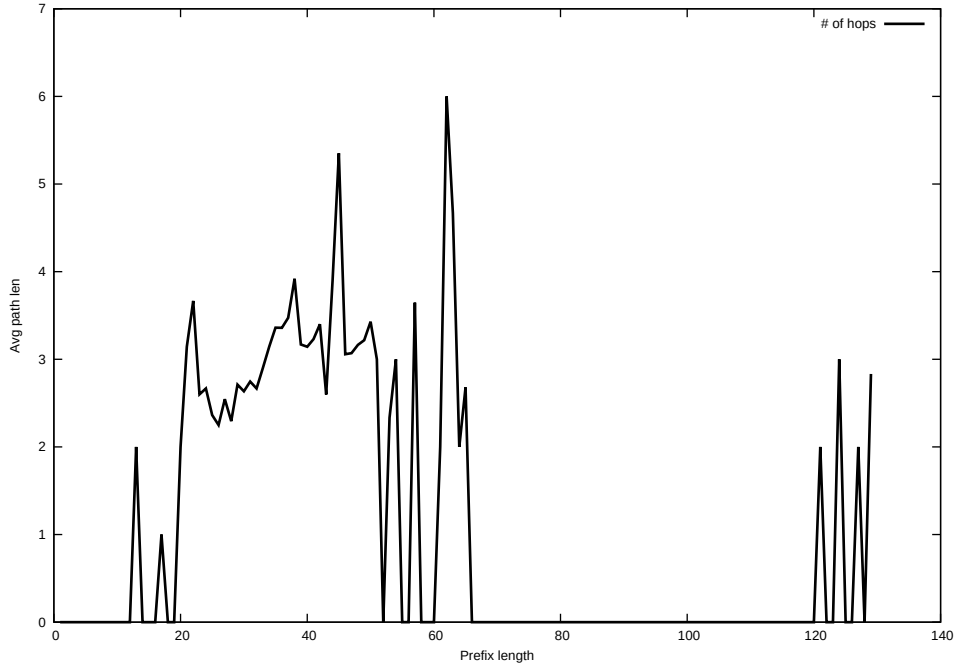
2013-11-18



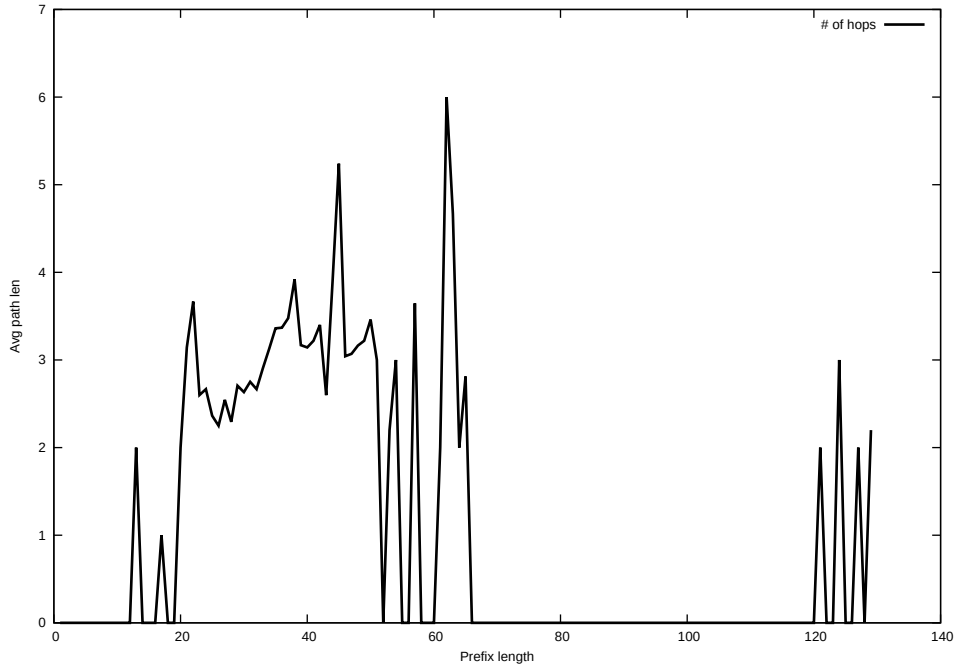
2013-11-19



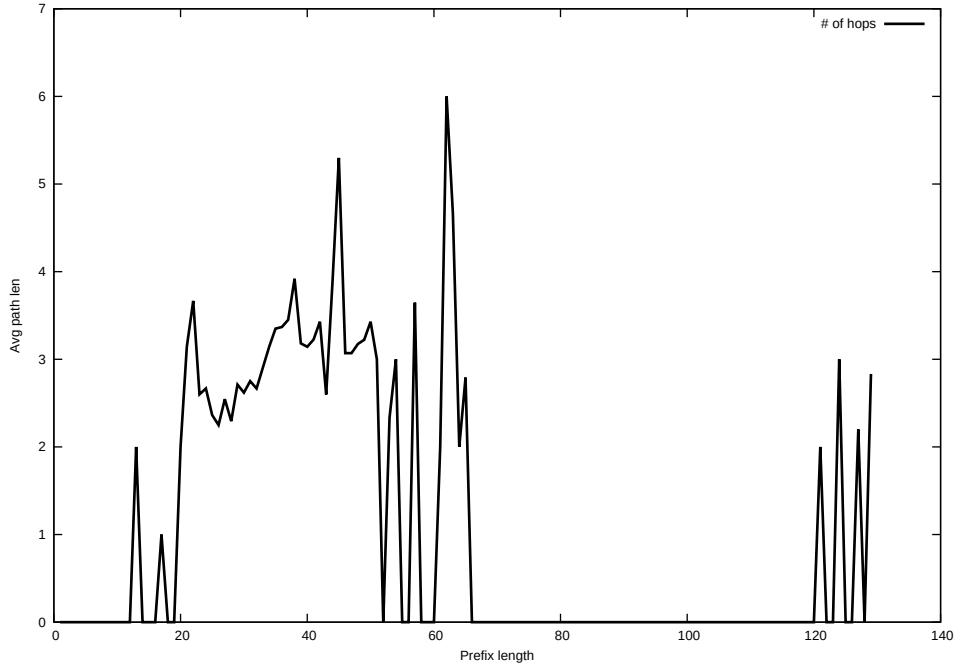
2013-11-20



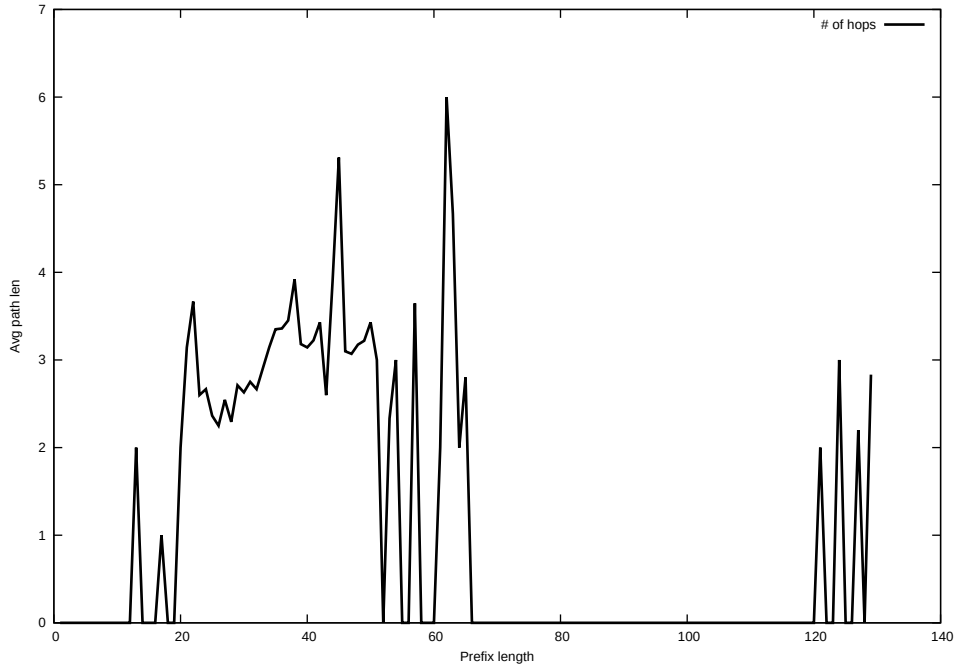
2013-11-21



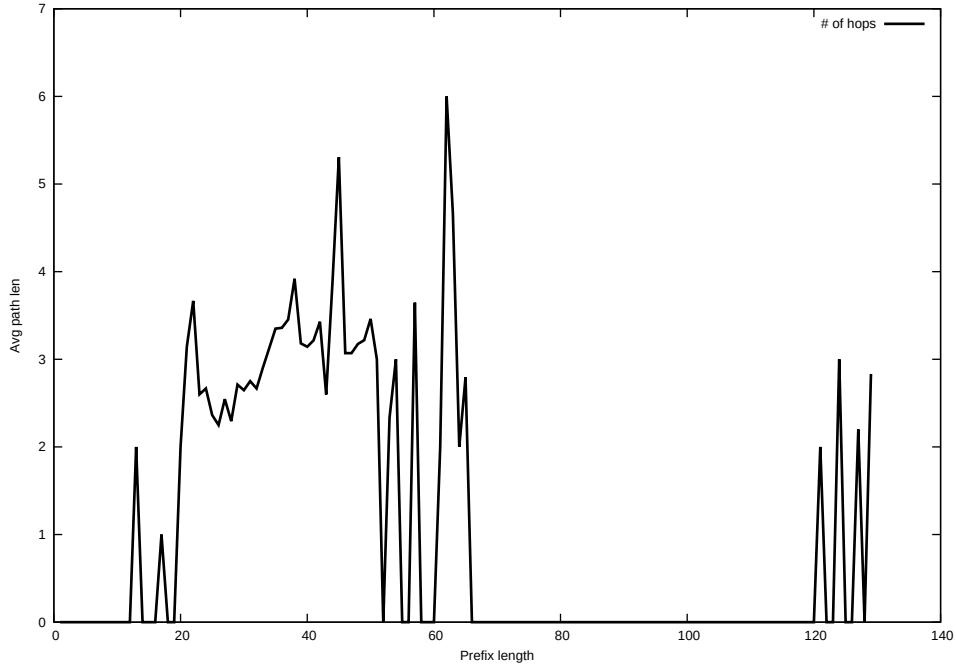
2013-11-22



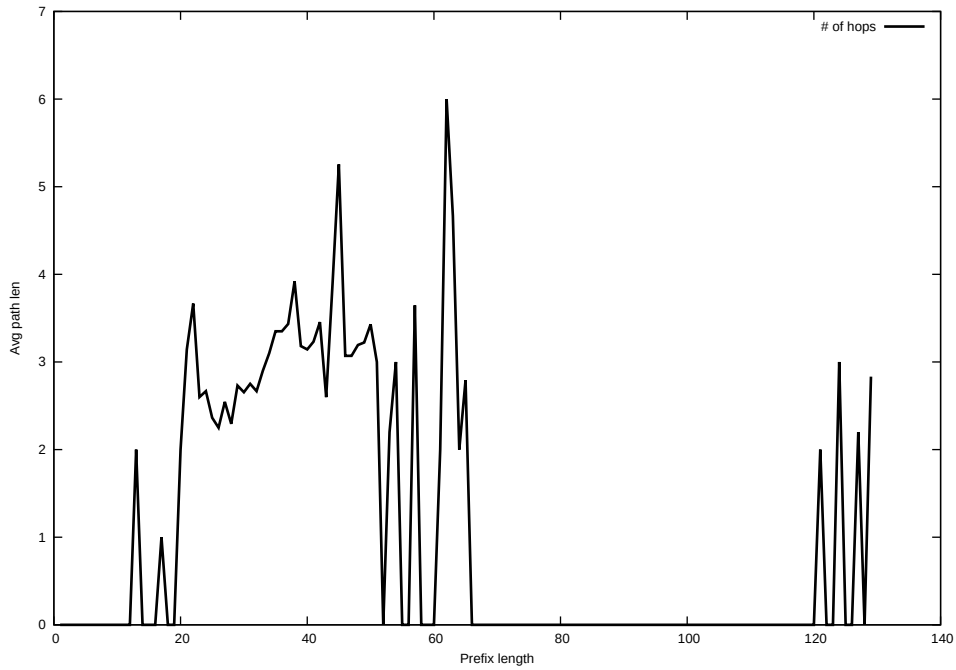
2013-11-23



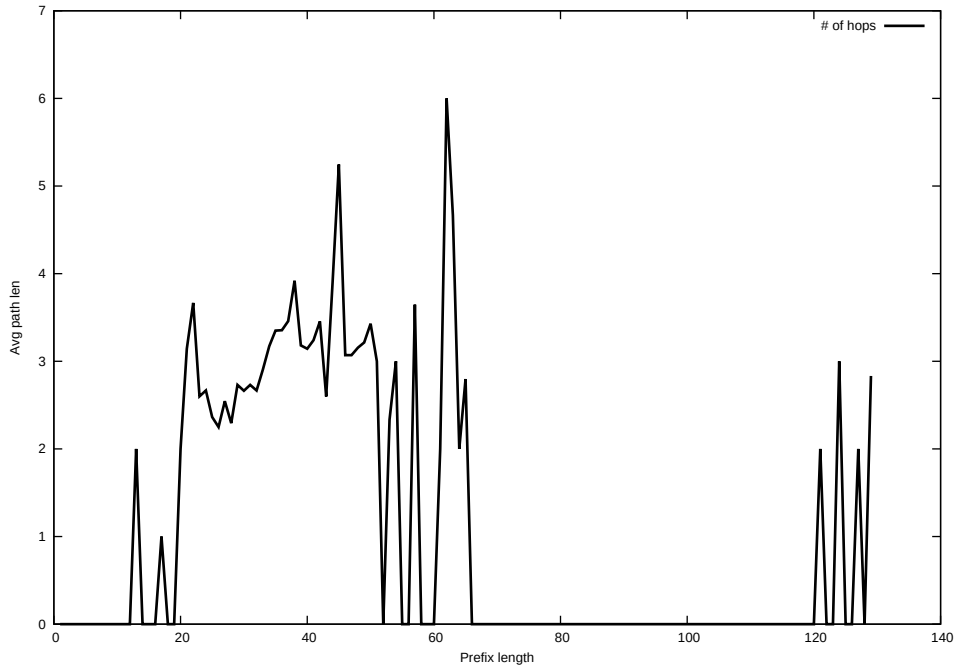
2013-11-24



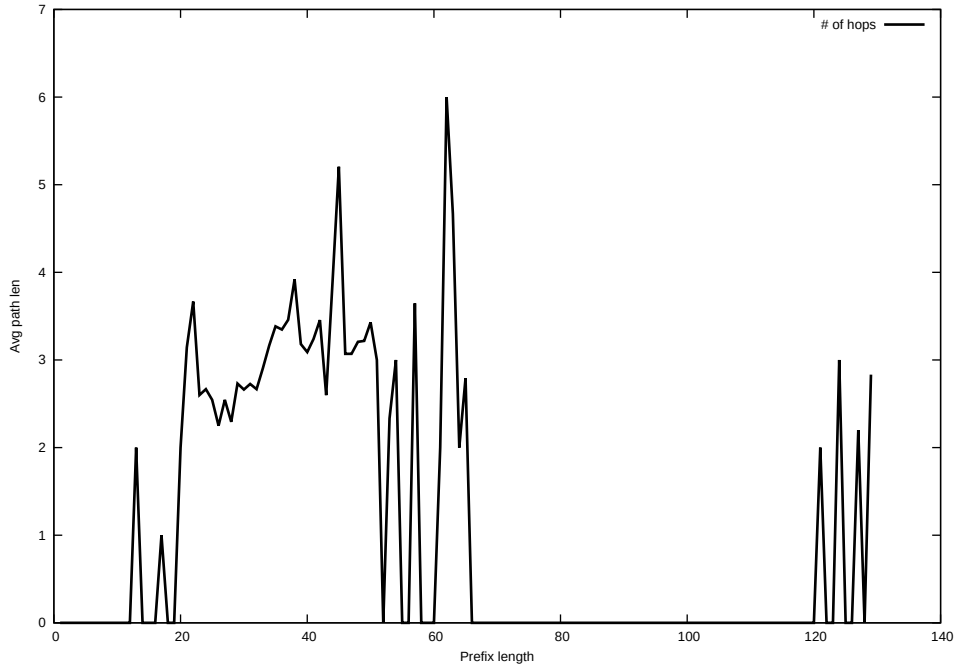
2013-11-25



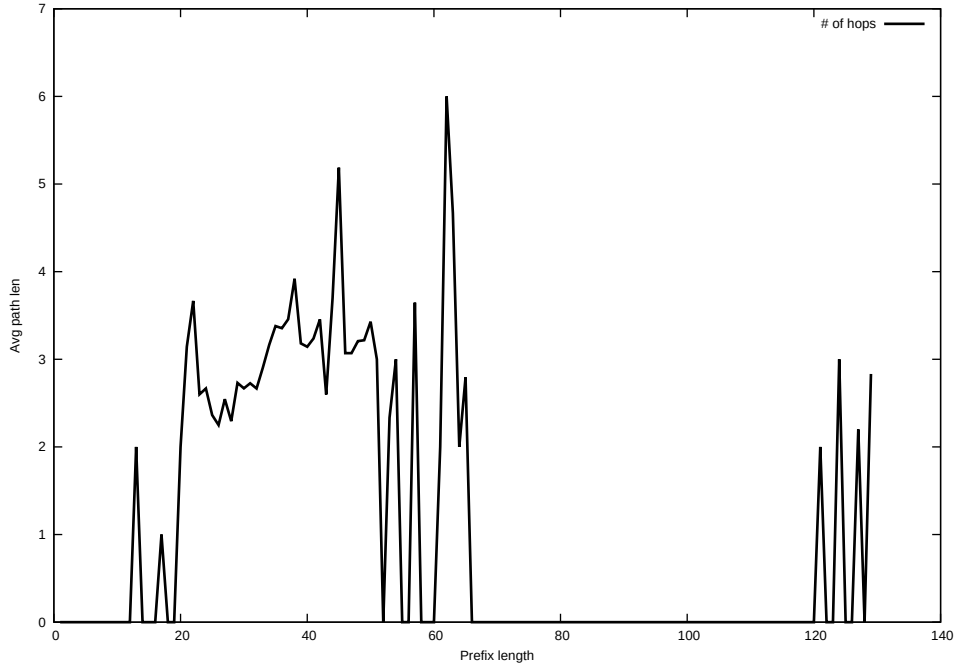
2013-11-26



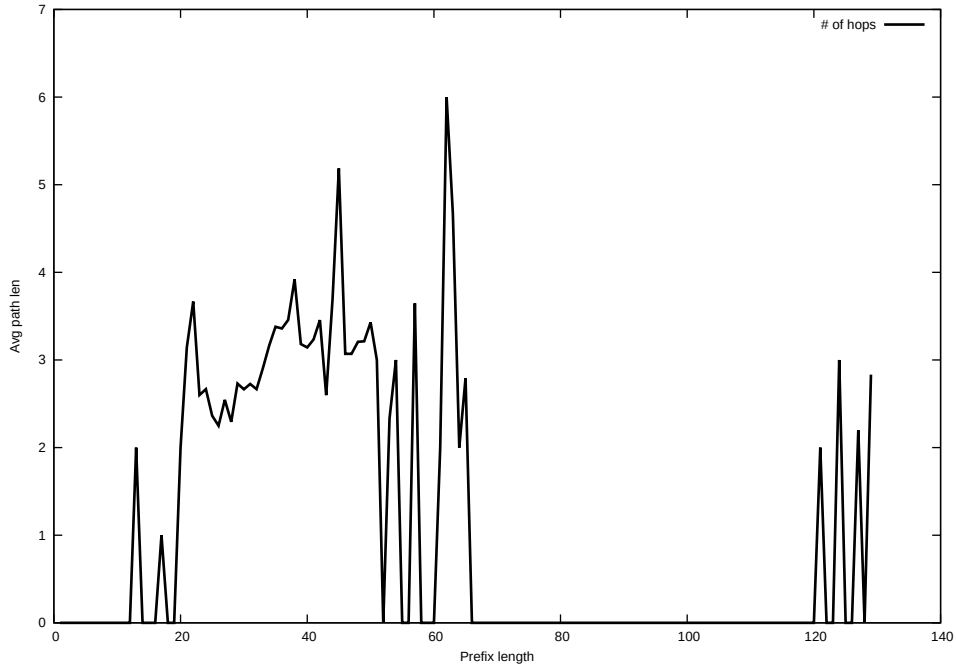
2013-11-27



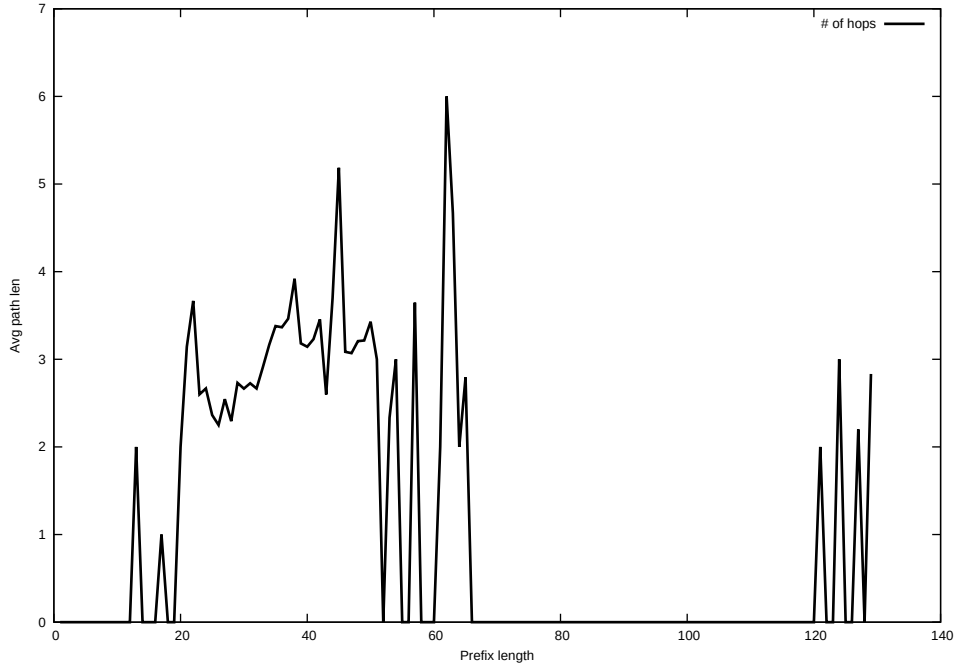
2013-11-28



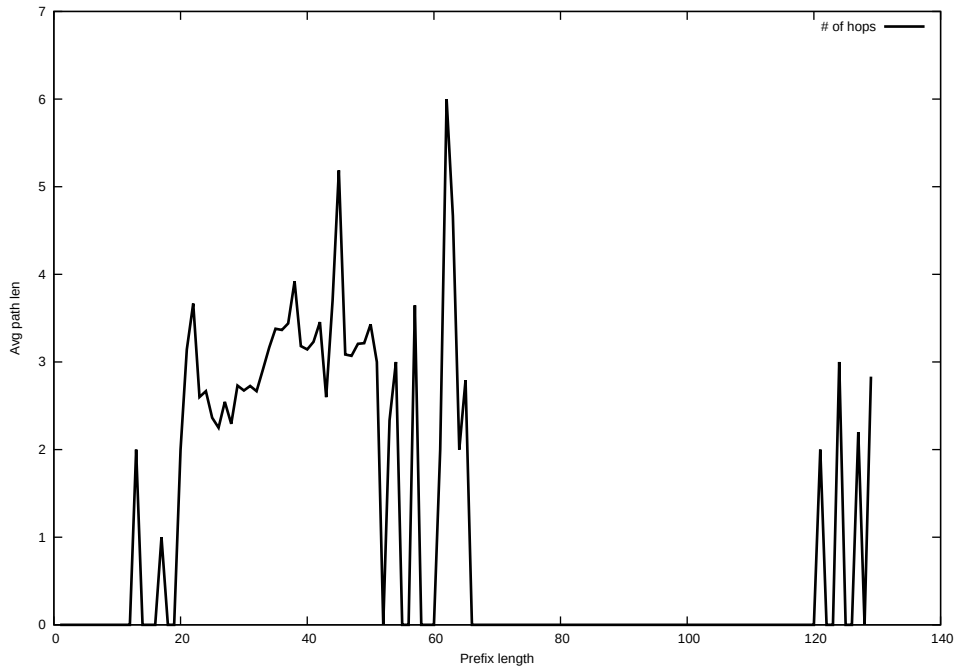
2013-11-29



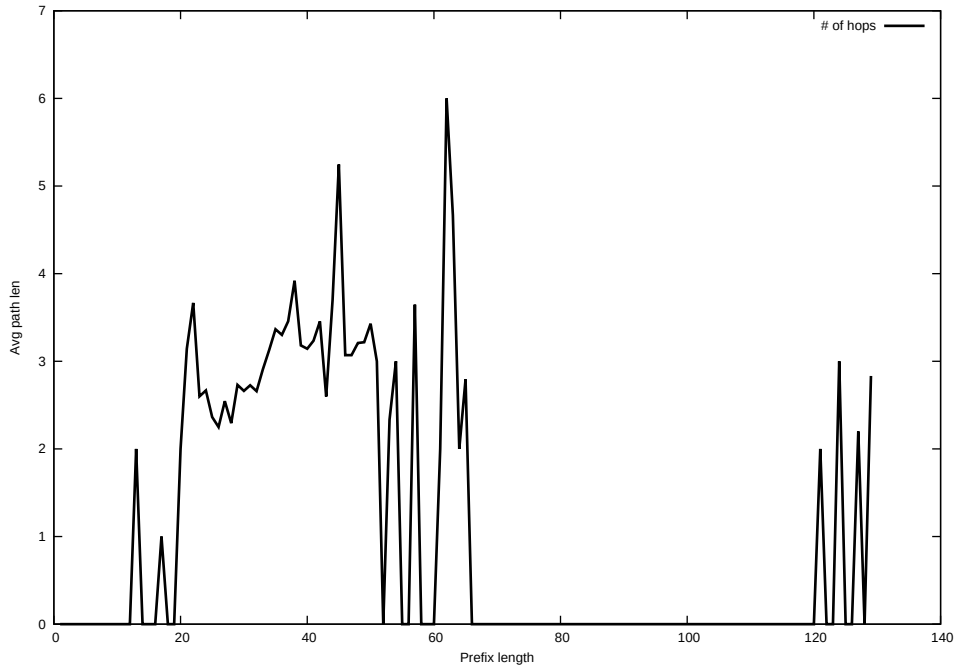
2013-11-30



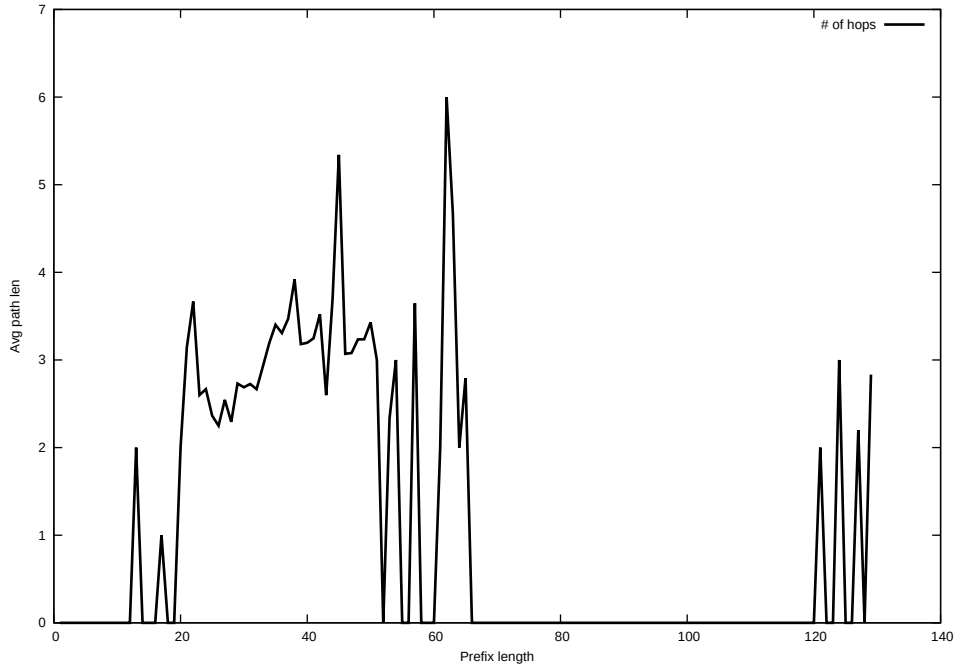
2013-12-01



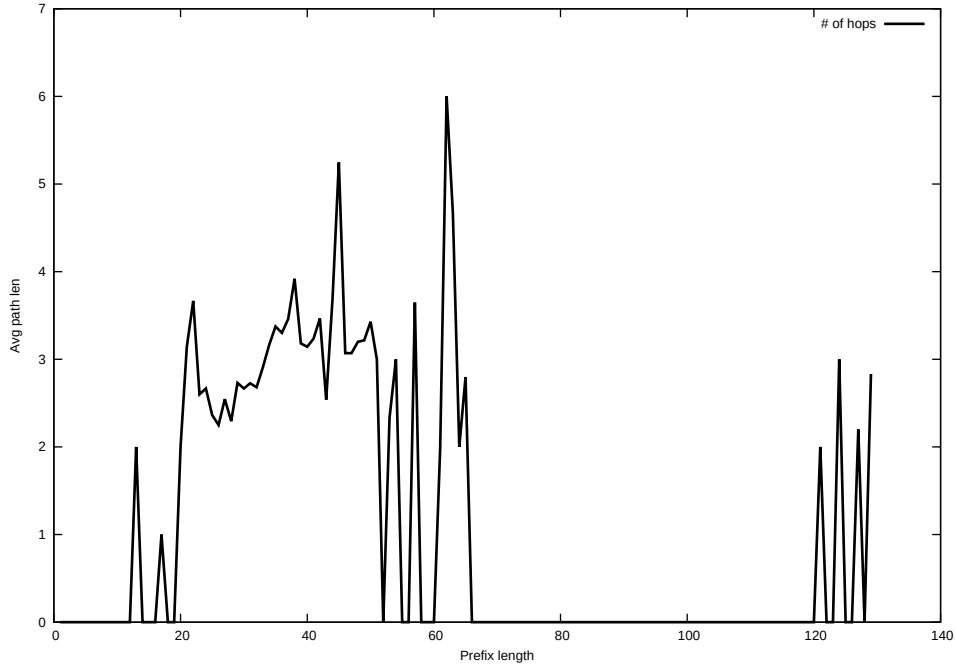
2013-12-02



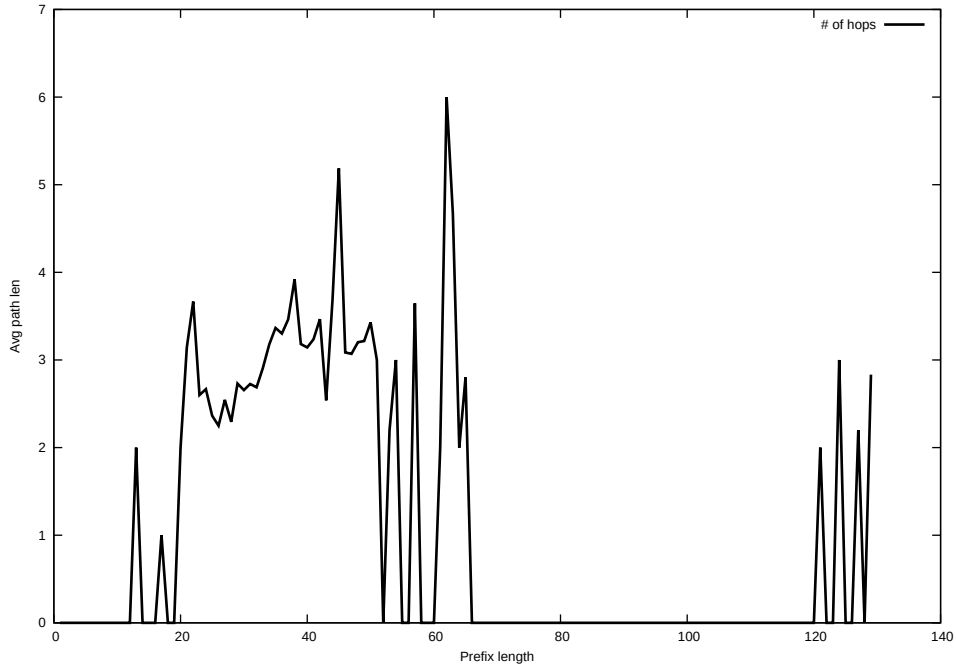
2013-12-03



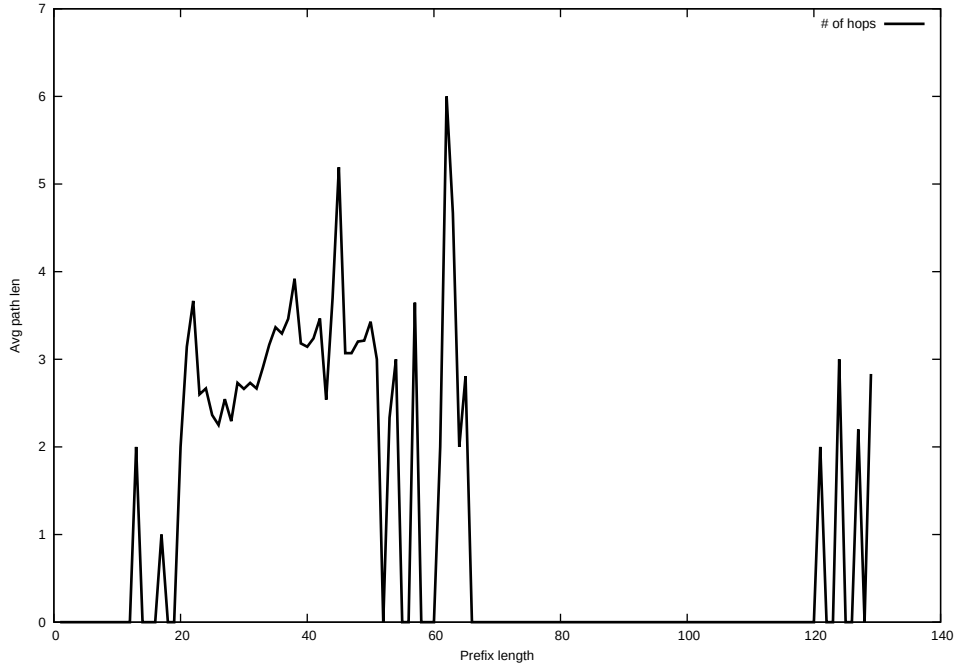
2013-12-04



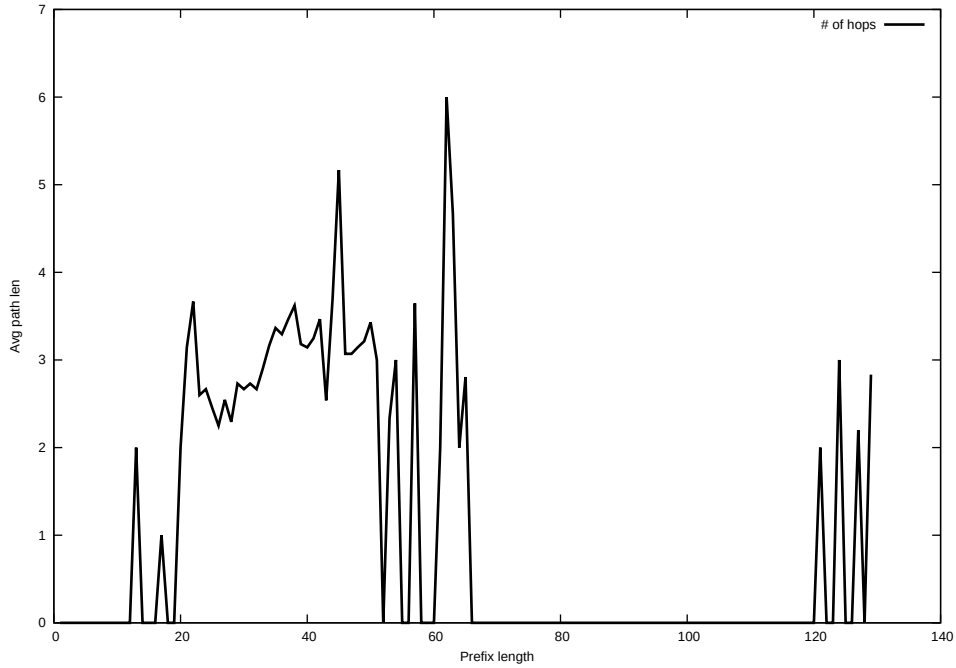
2013-12-05



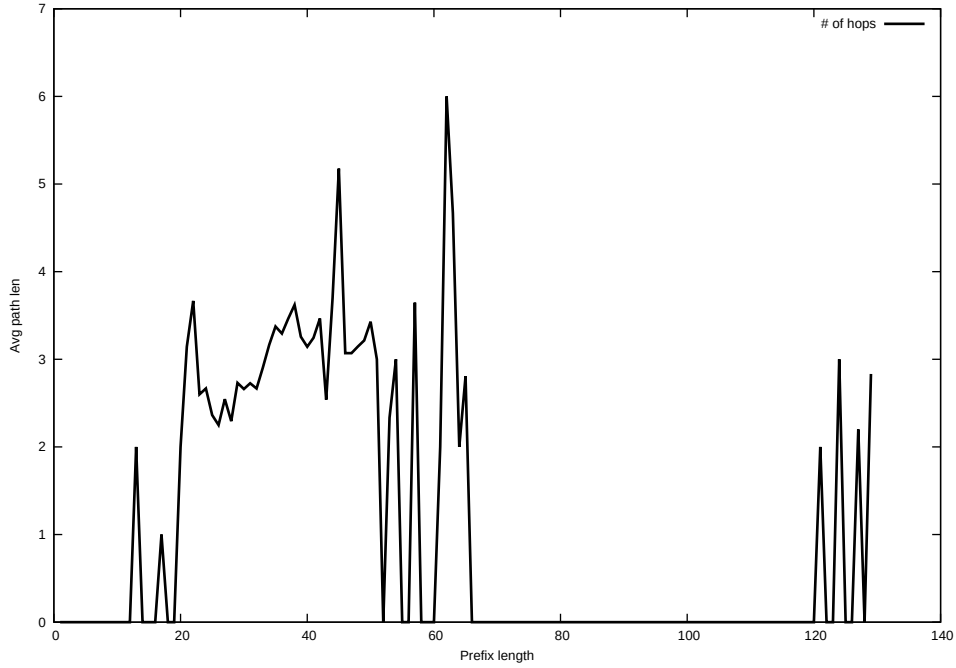
2013-12-06



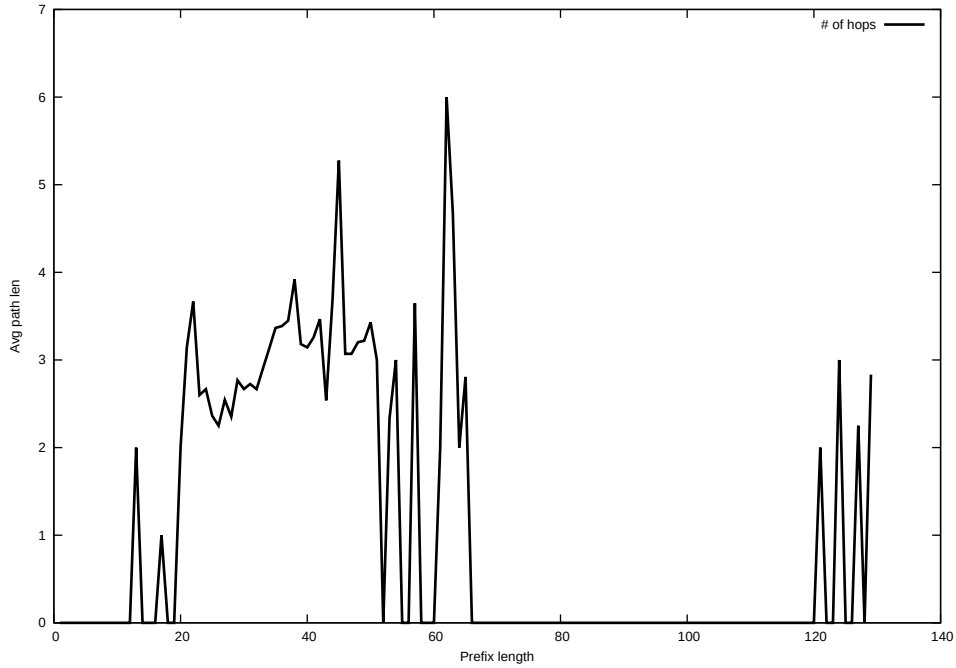
2013-12-07



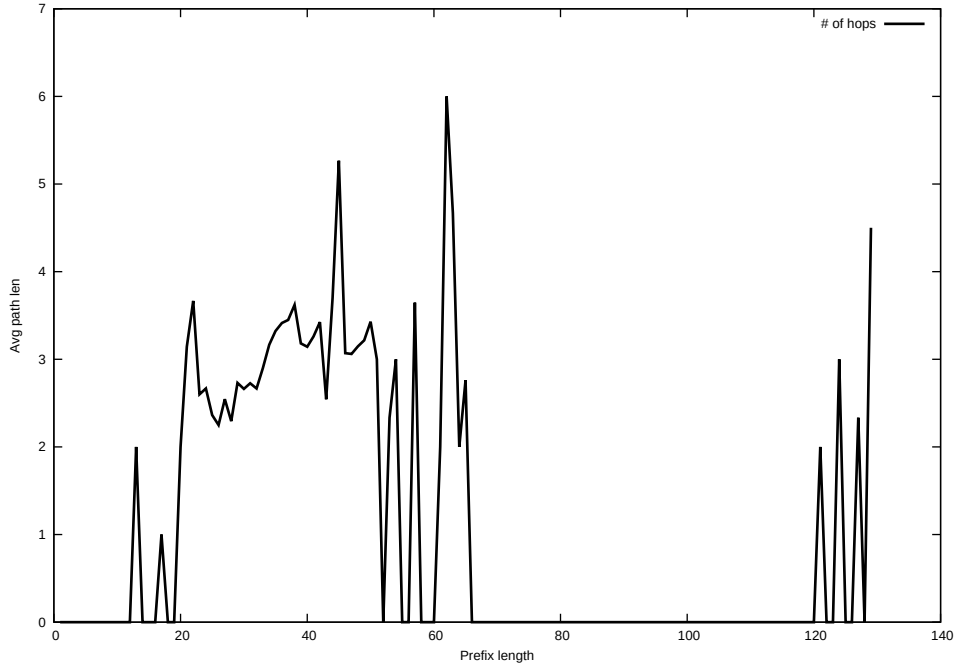
2013-12-08



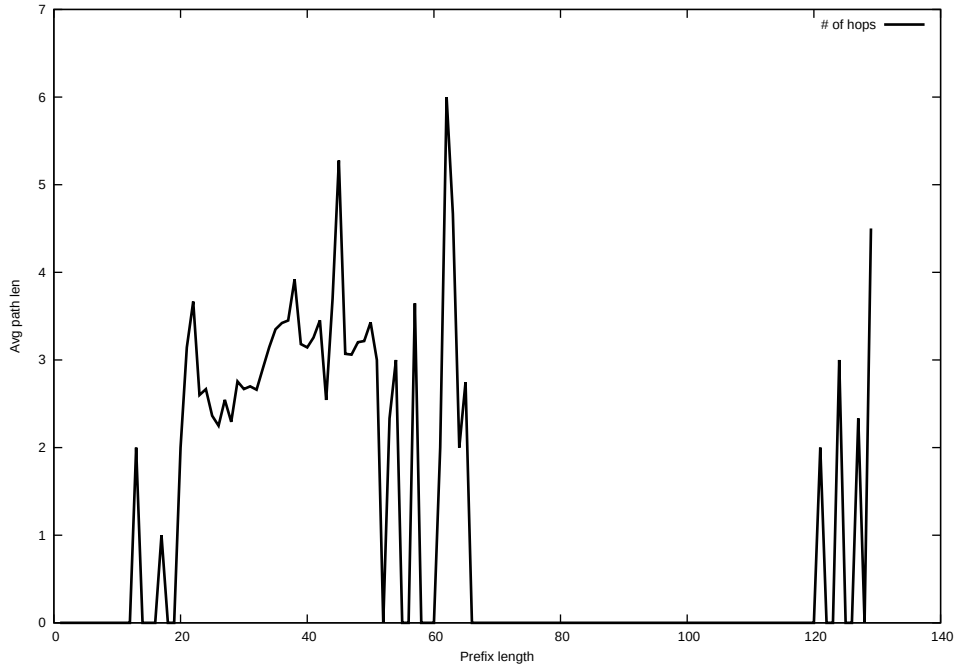
2013-12-09



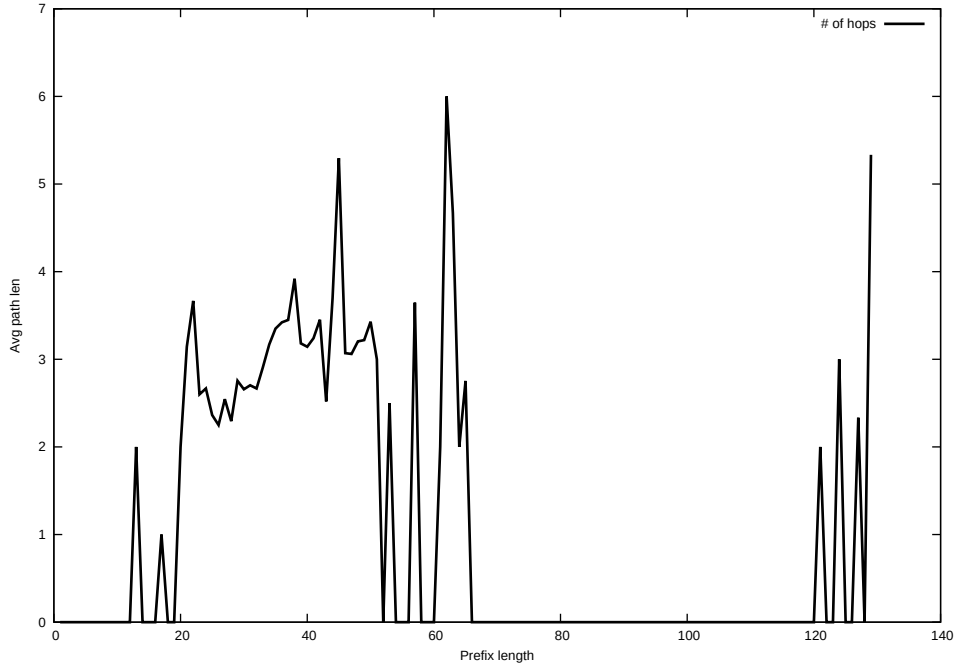
2013-12-10



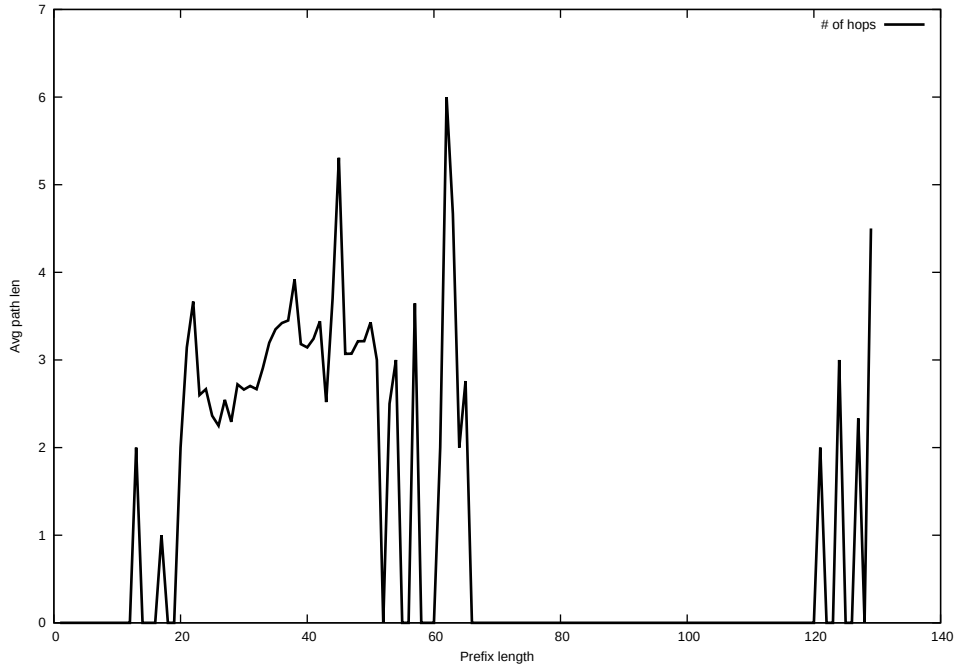
2013-12-11



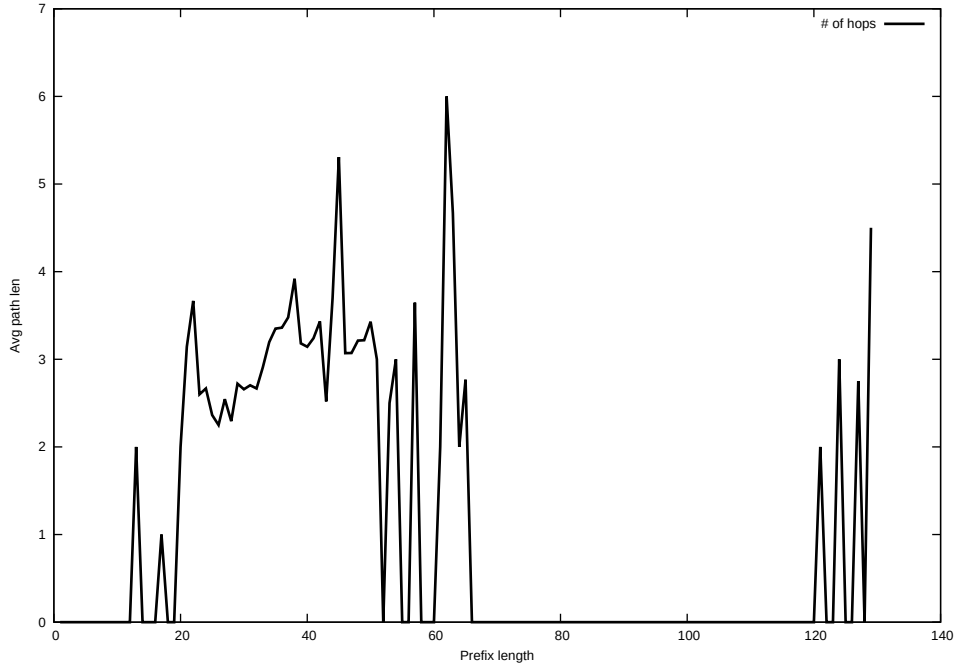
2013-12-12



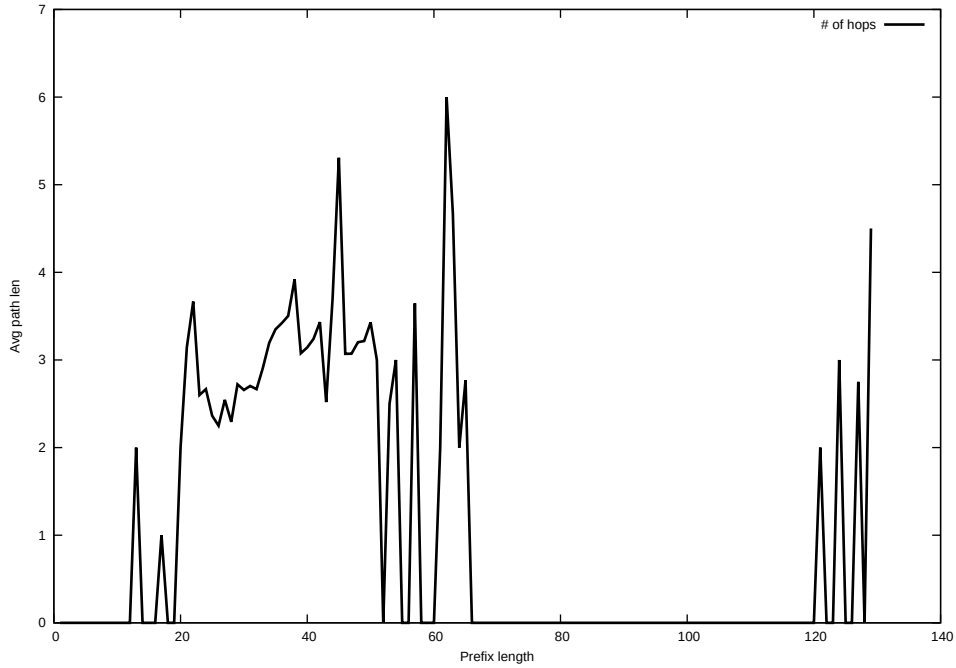
2013-12-13



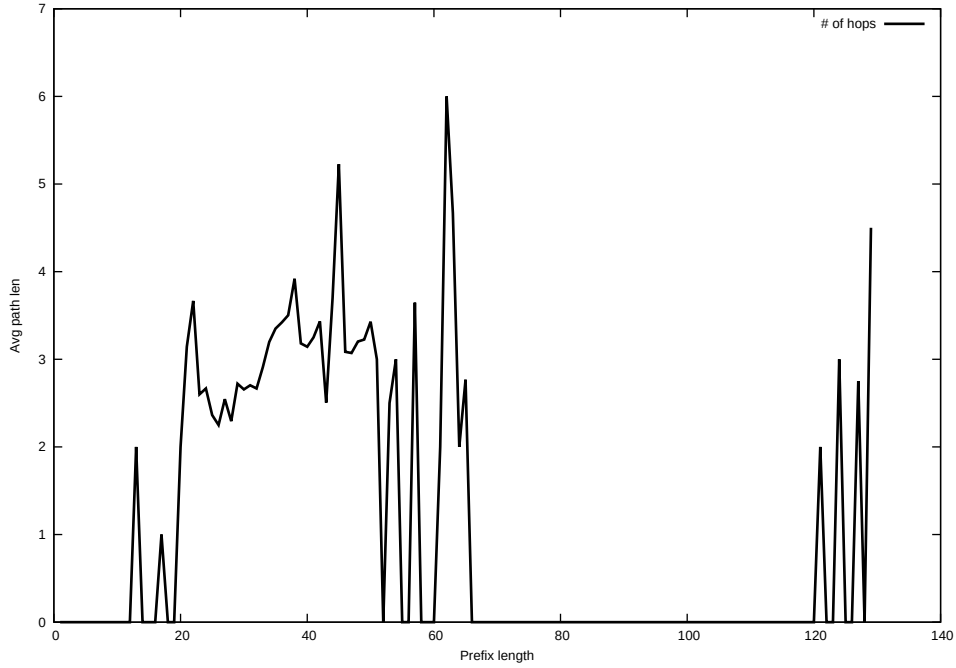
2013-12-14



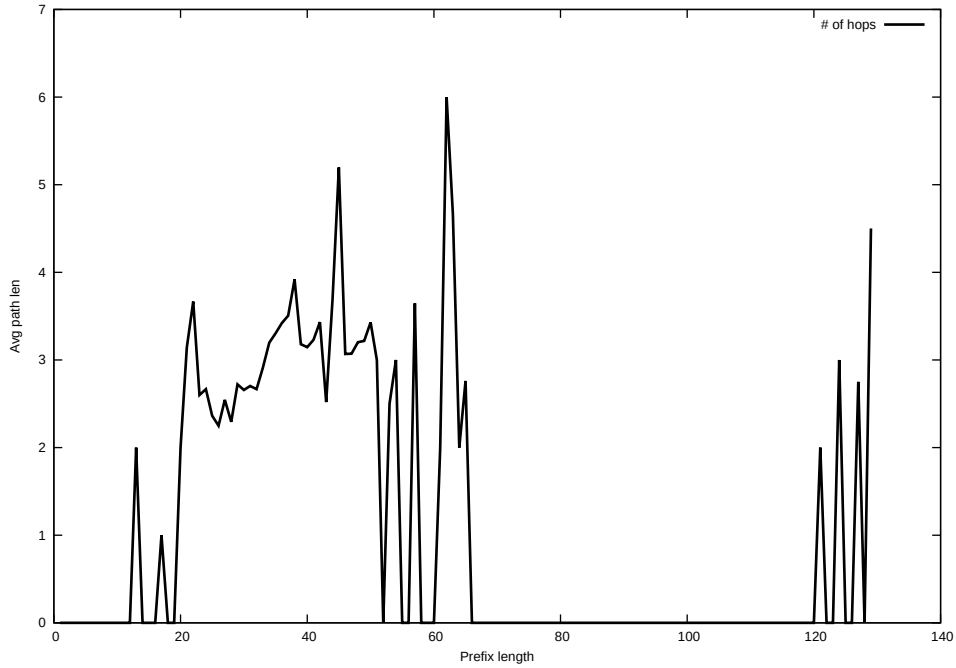
2013-12-15



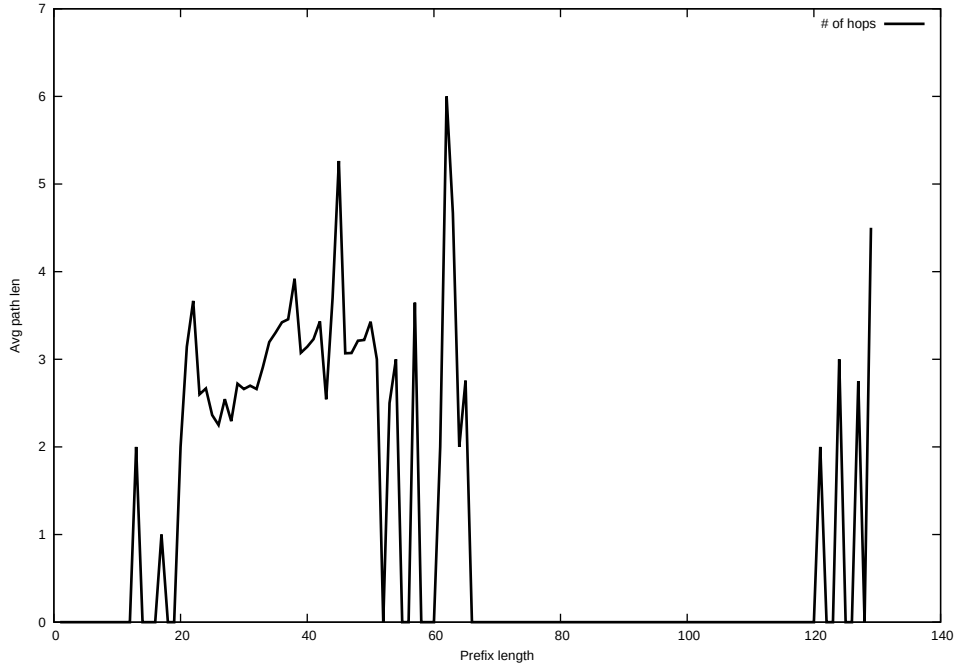
2013-12-16



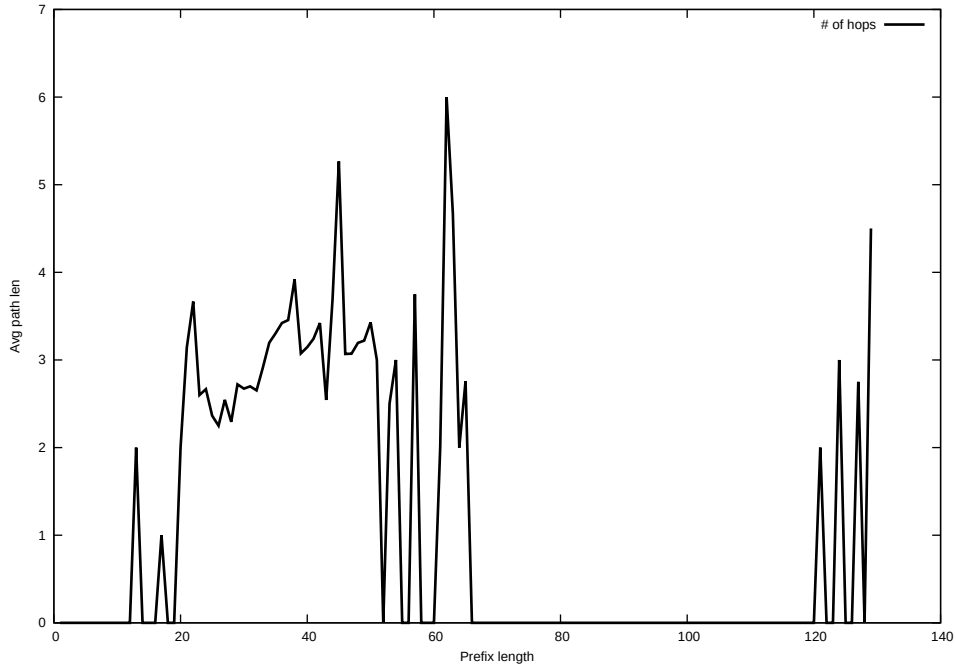
2013-12-17



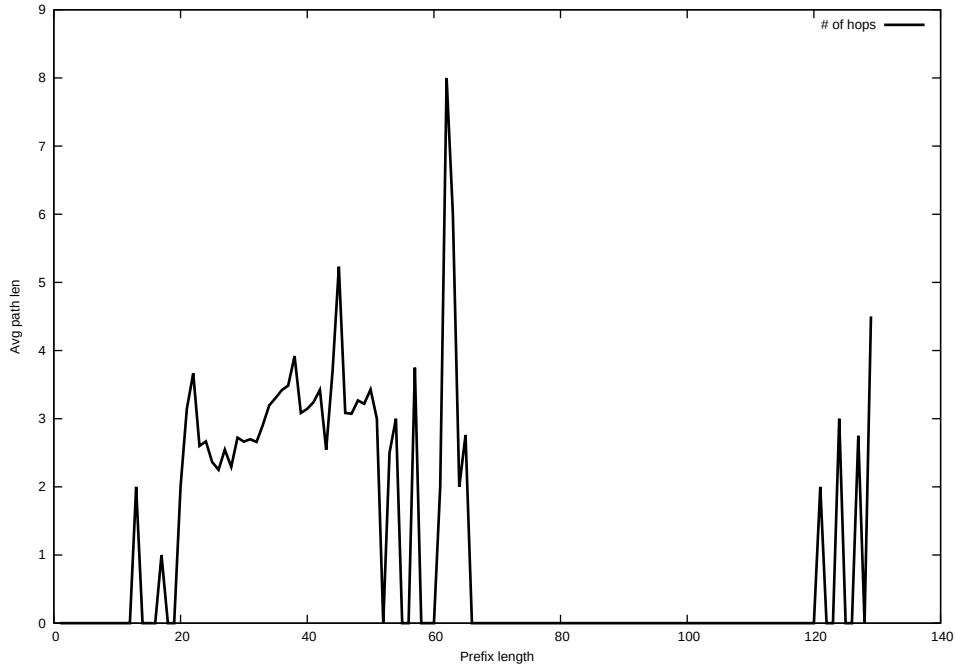
2013-12-18



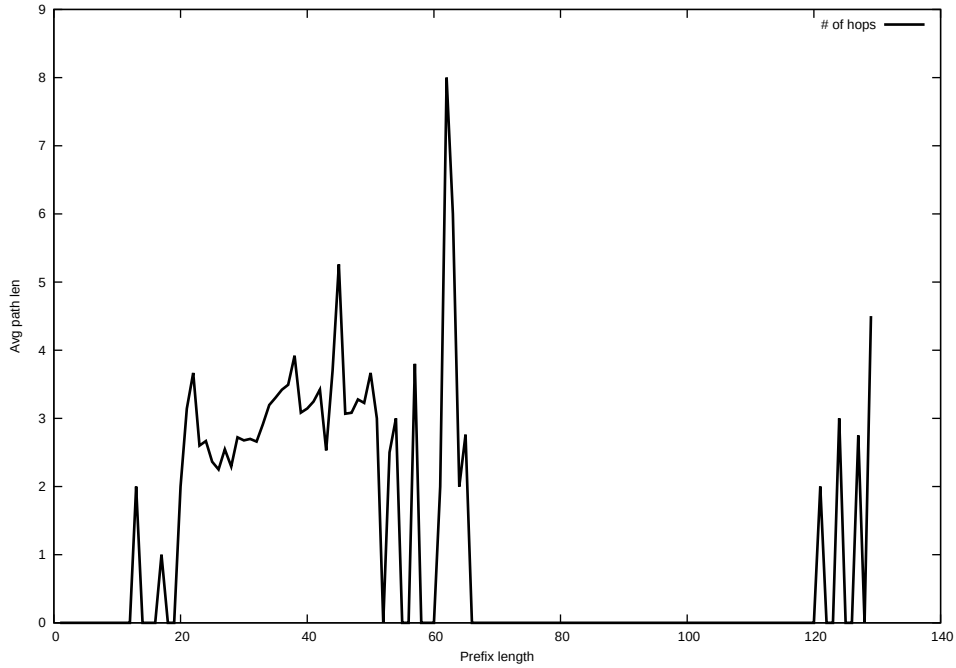
2013-12-19



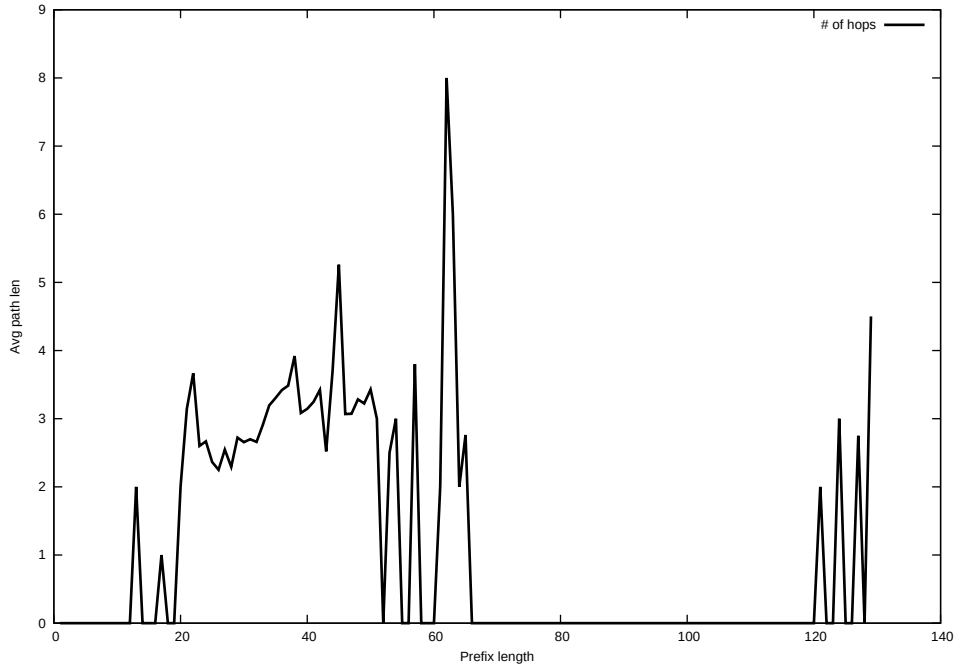
2013-12-20



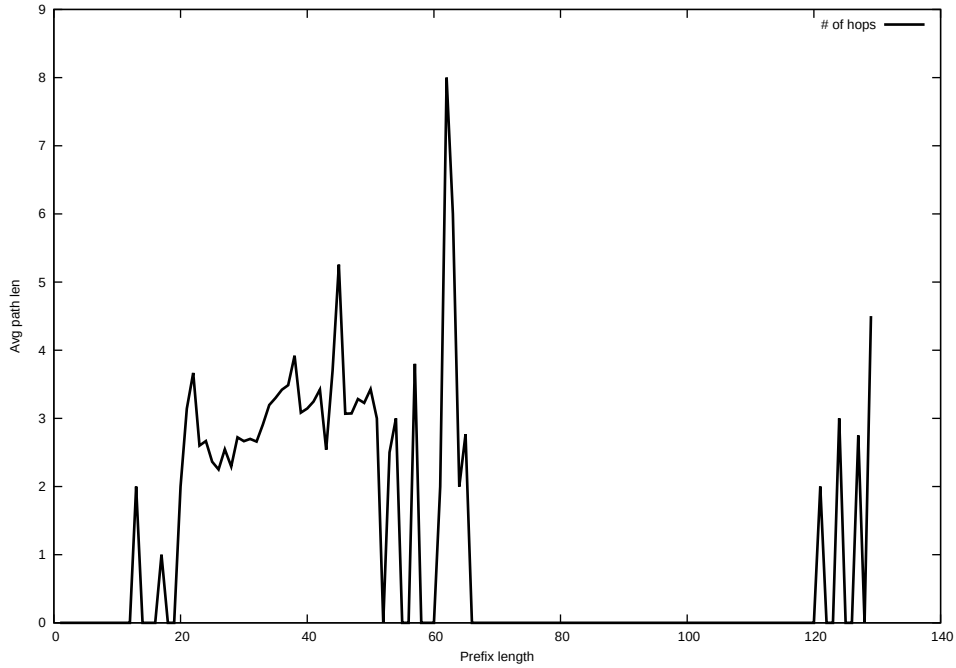
2013-12-21



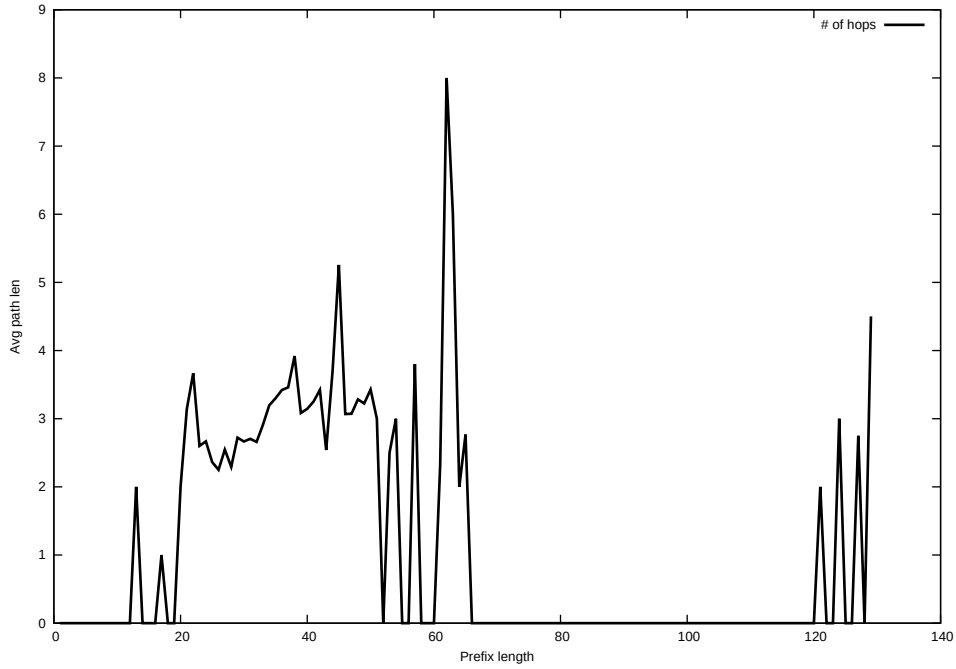
2013-12-22



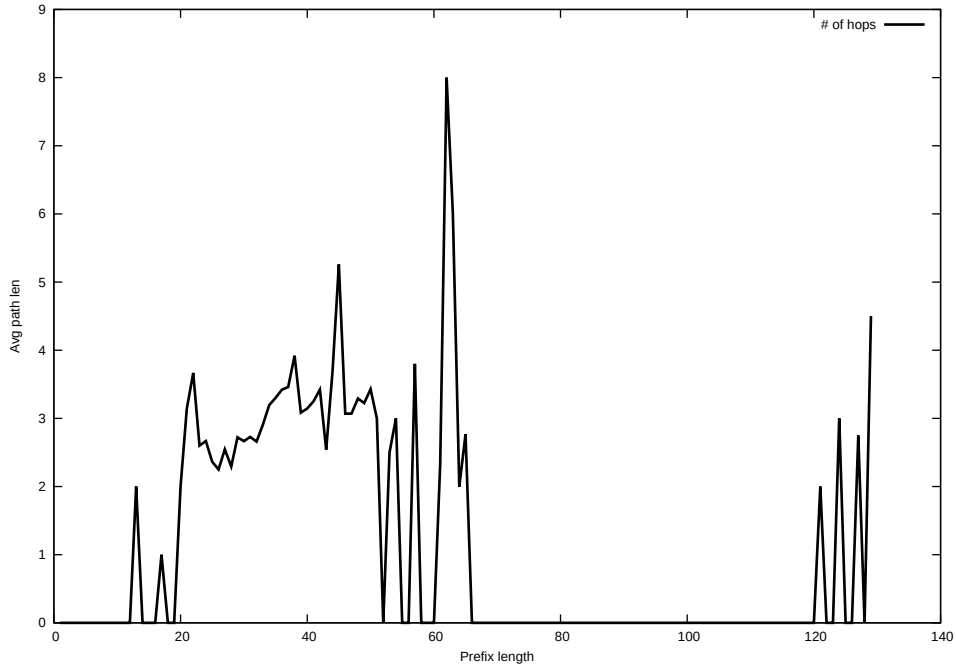
2013-12-23



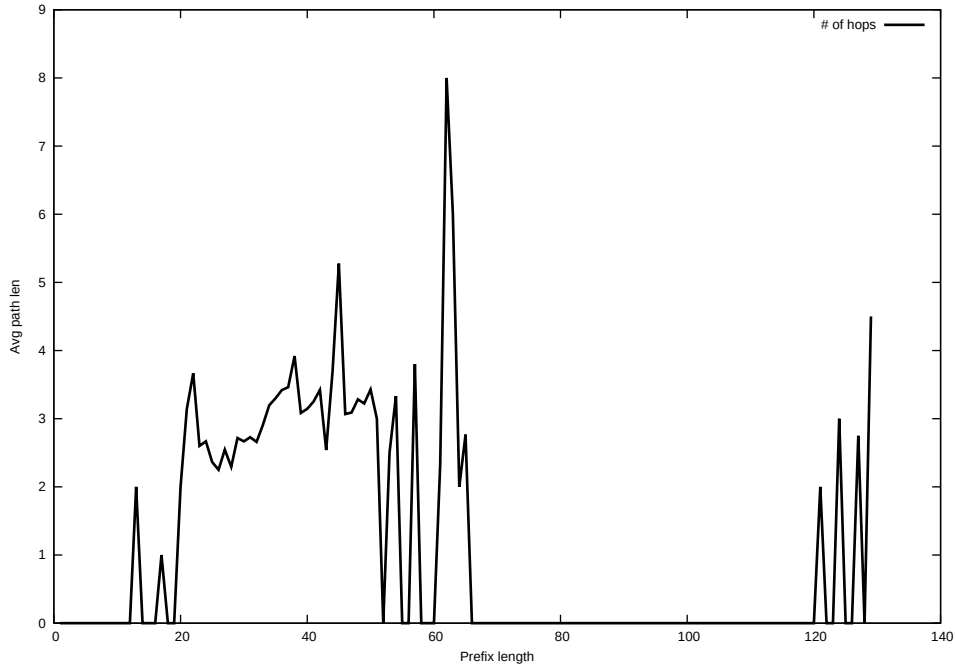
2013-12-24



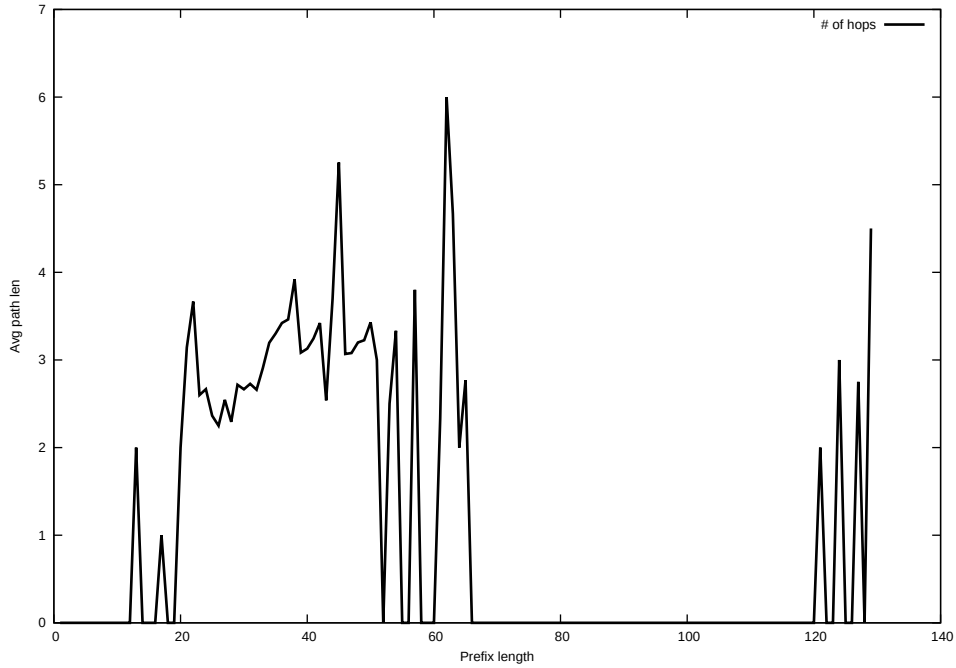
2013-12-25



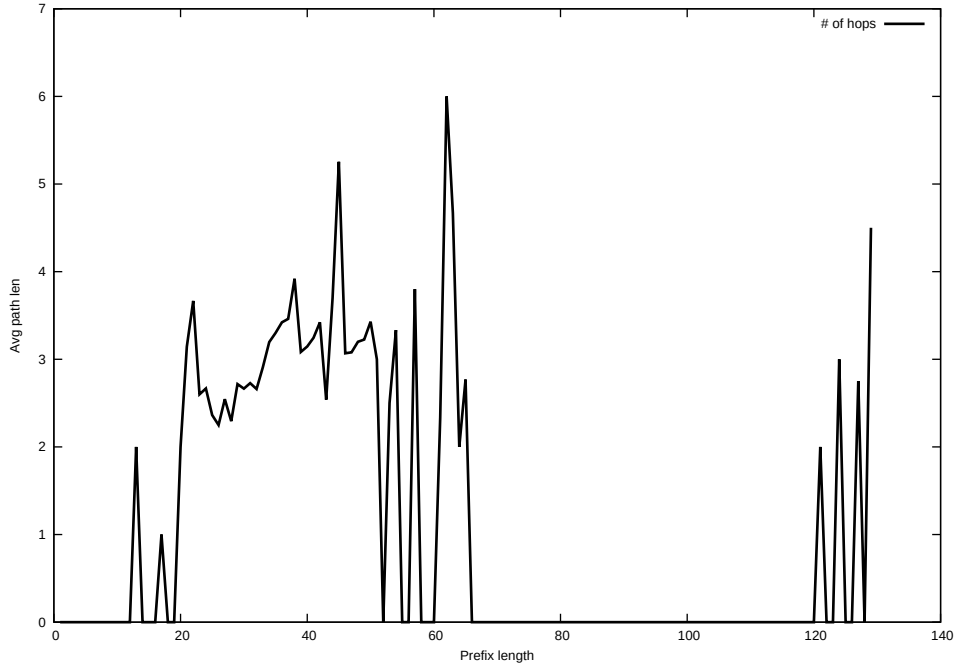
2013-12-26



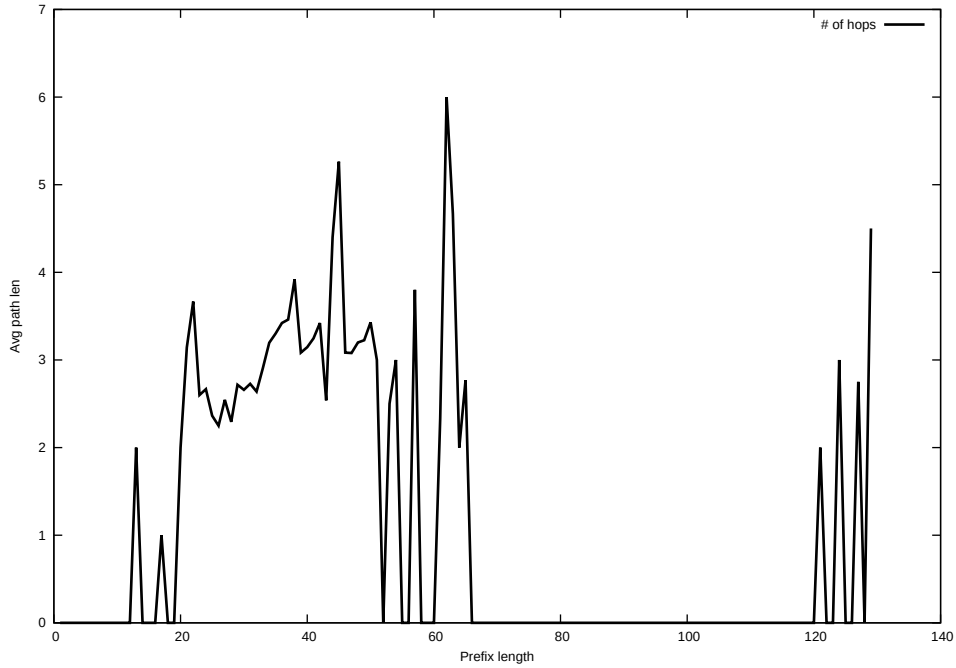
2013-12-27



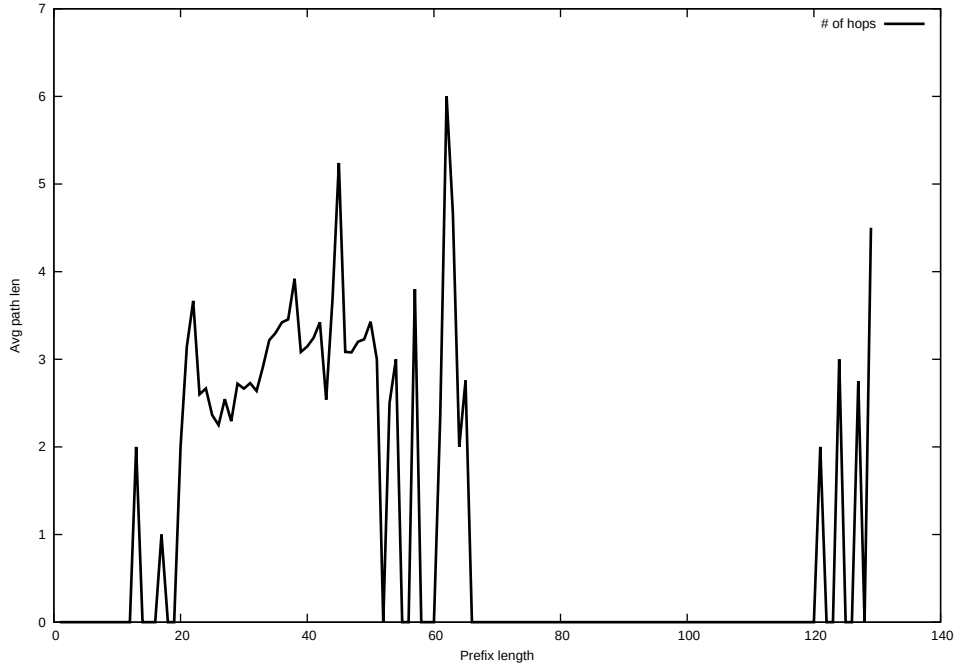
2013-12-28



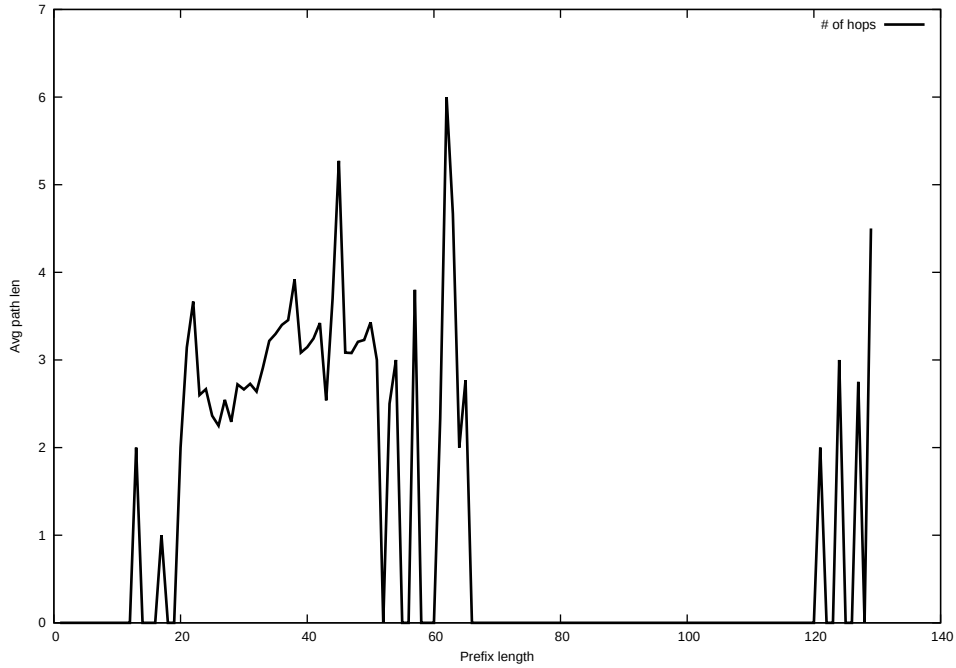
2013-12-29



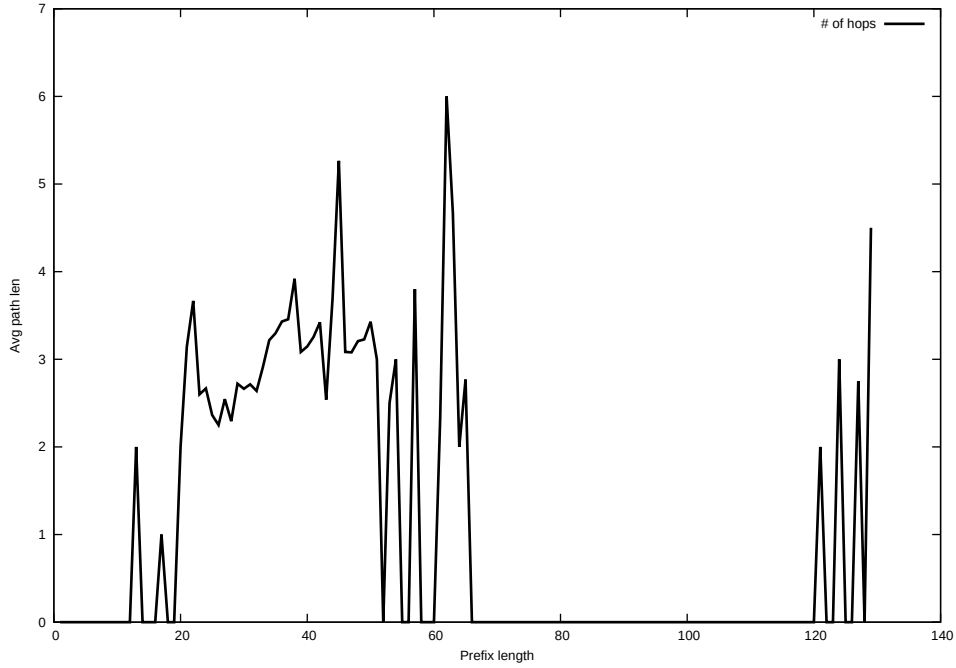
2013-12-30



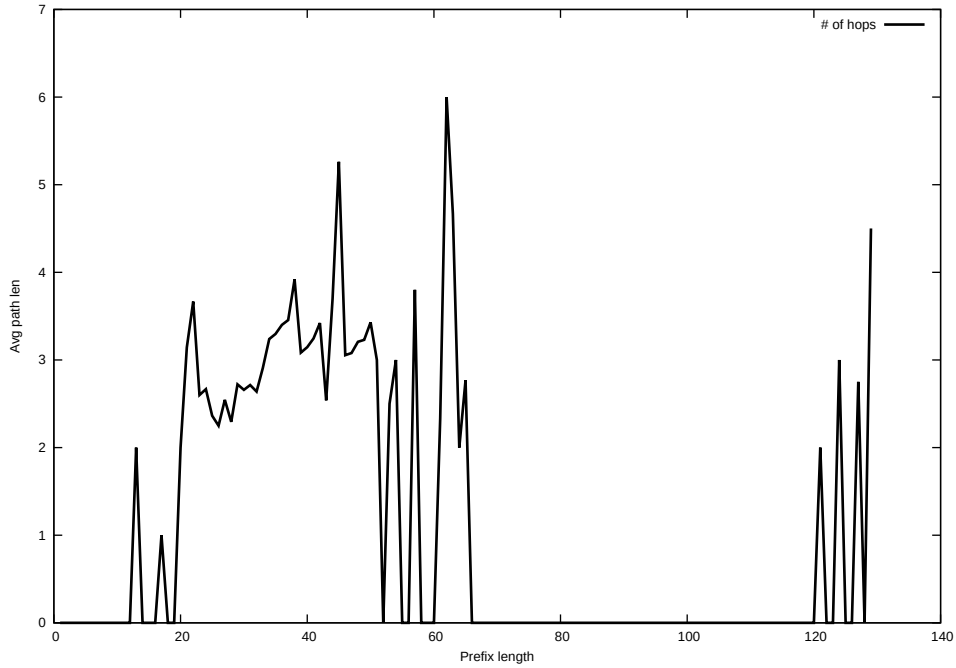
2013-12-31



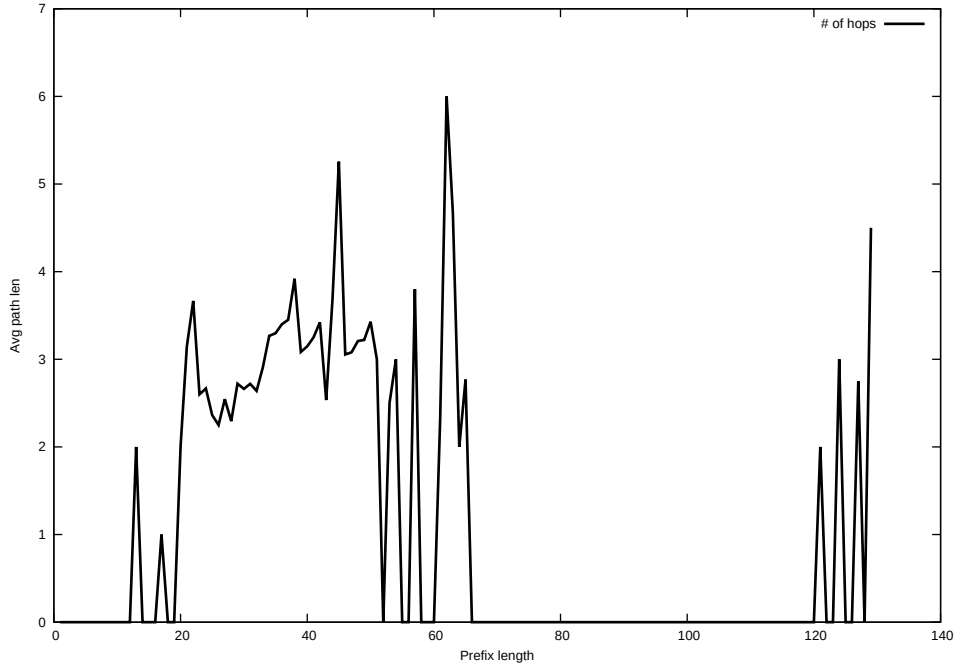
2014-01-01



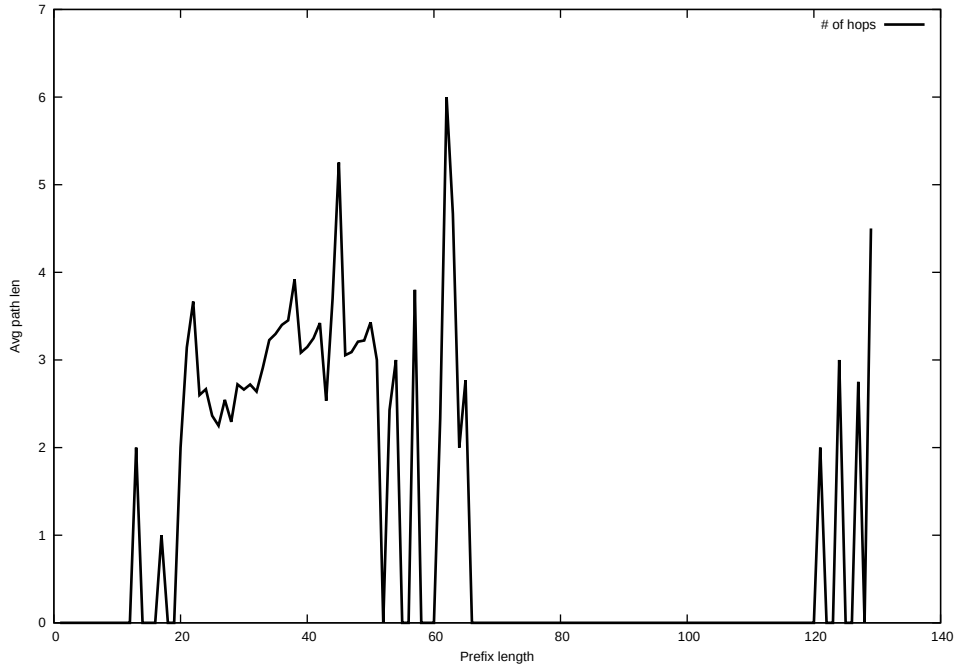
2014-01-02



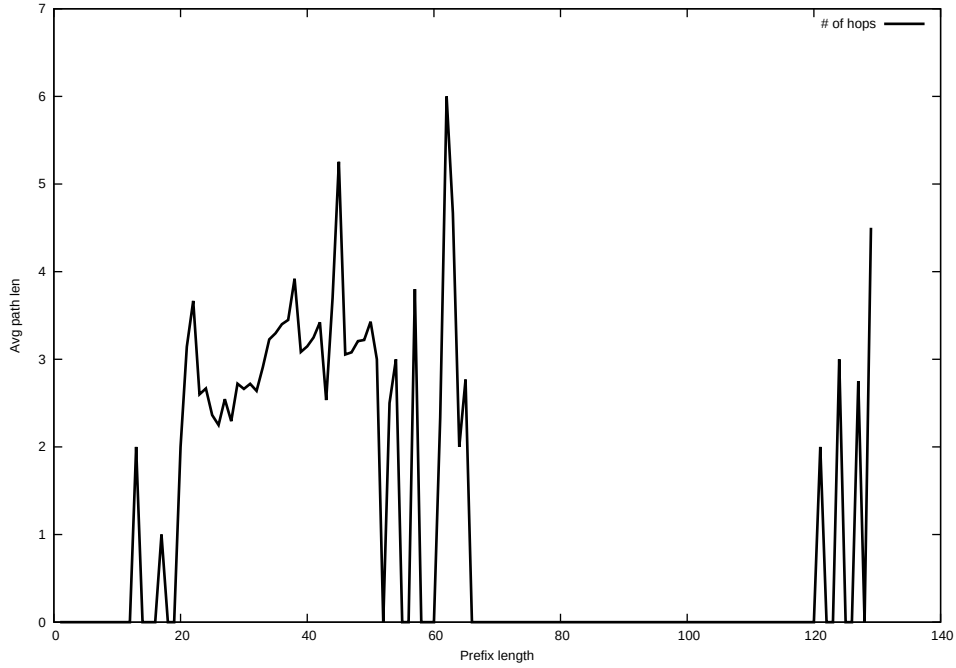
2014-01-03



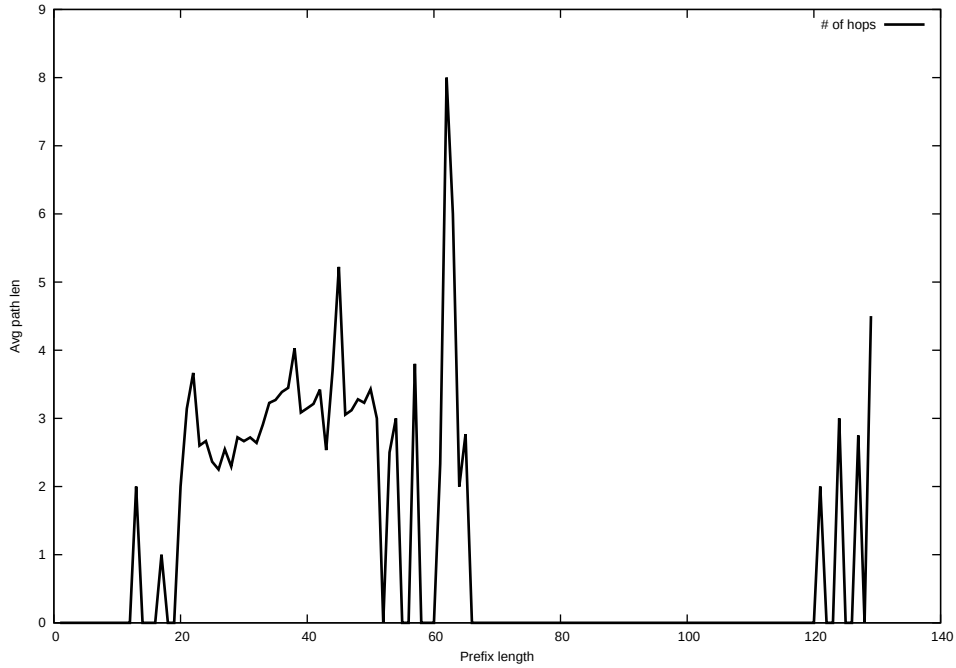
2014-01-04



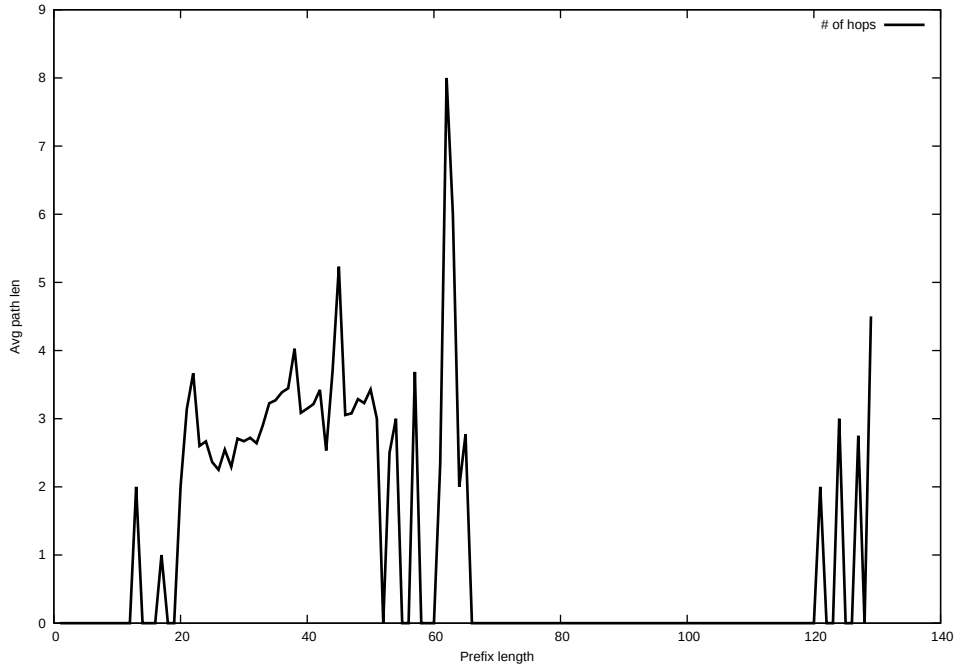
2014-01-05



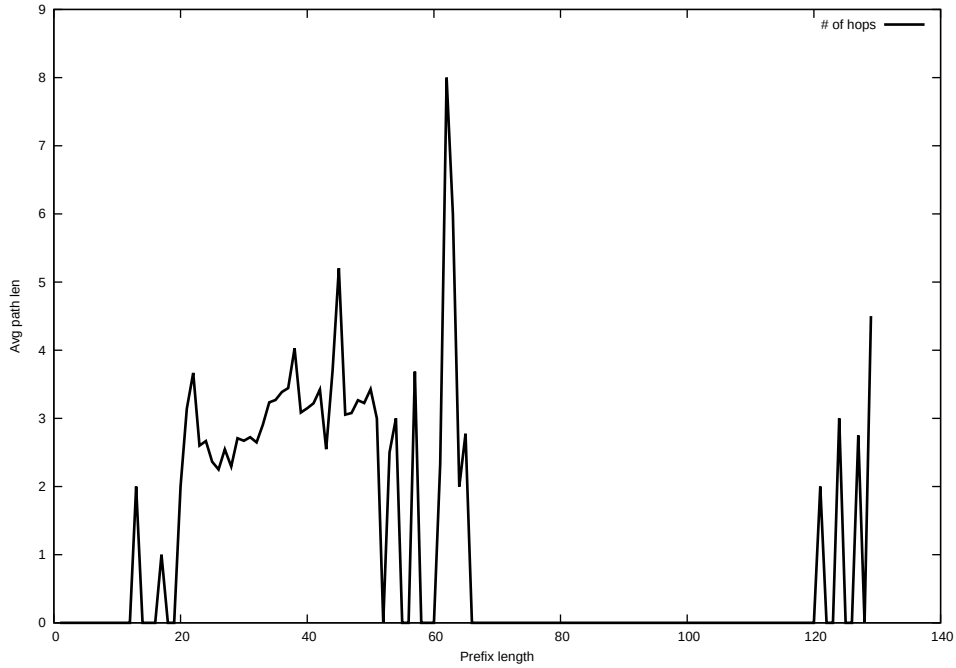
2014-01-06



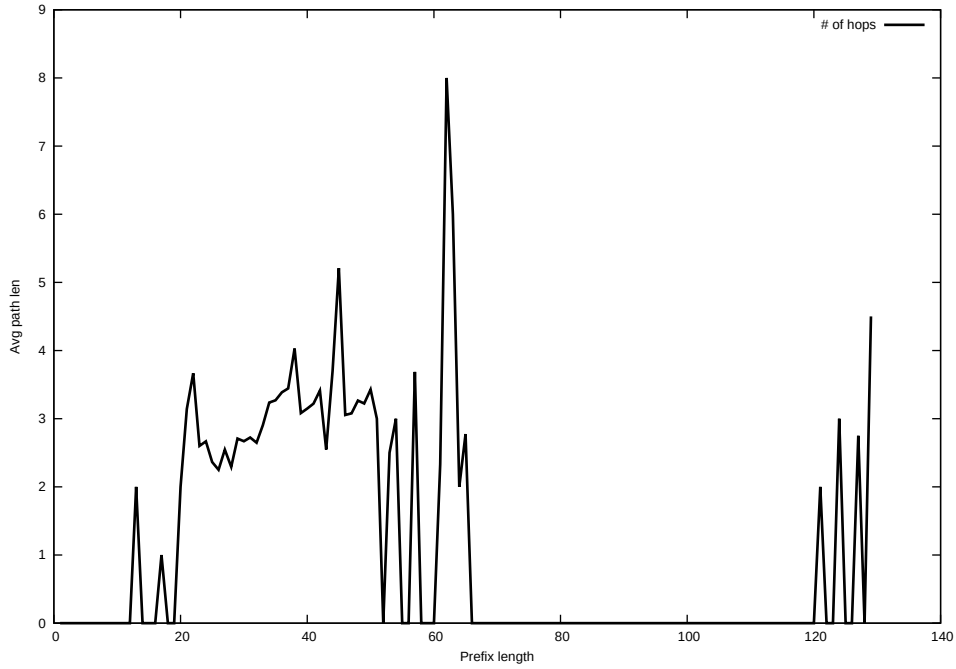
2014-01-07



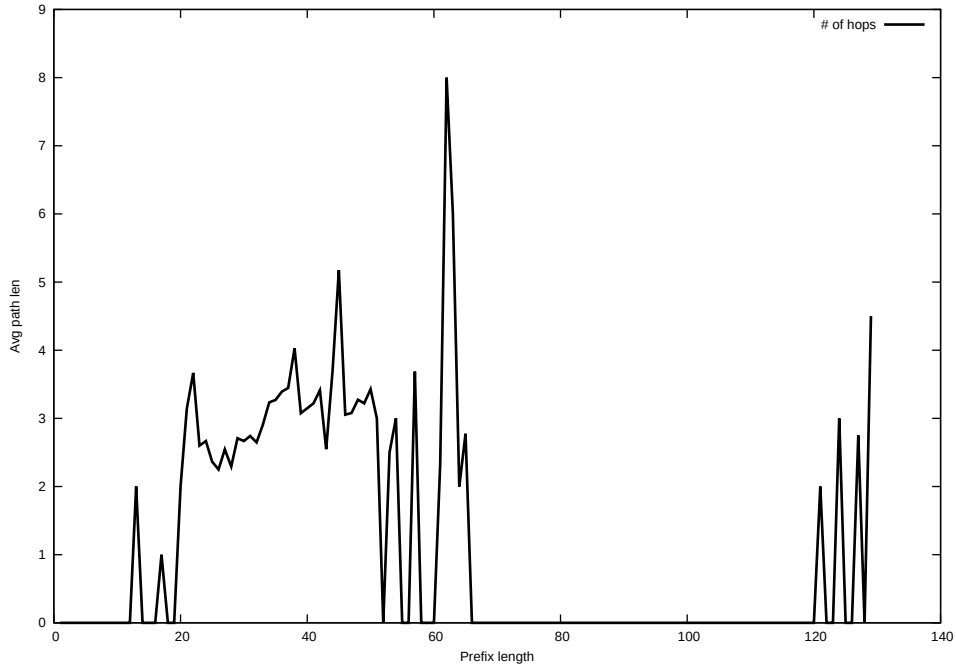
2014-01-08



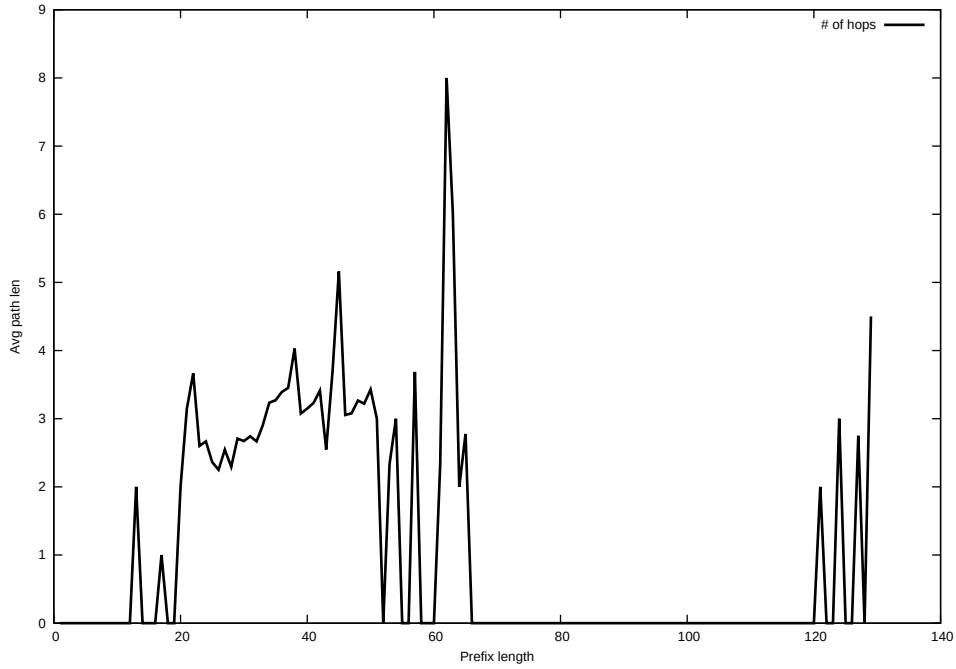
2014-01-09



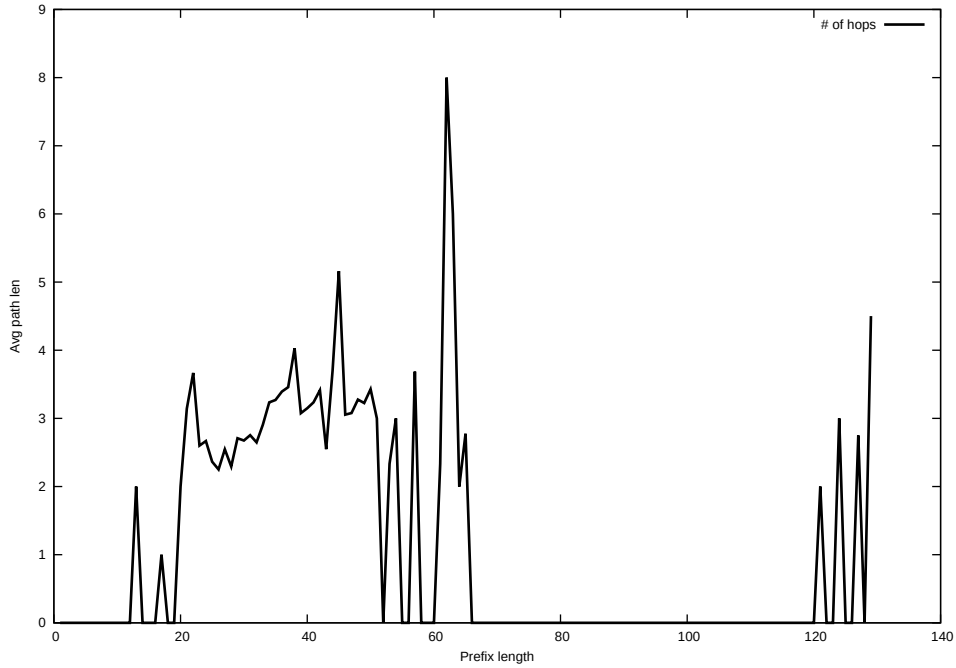
2014-01-10



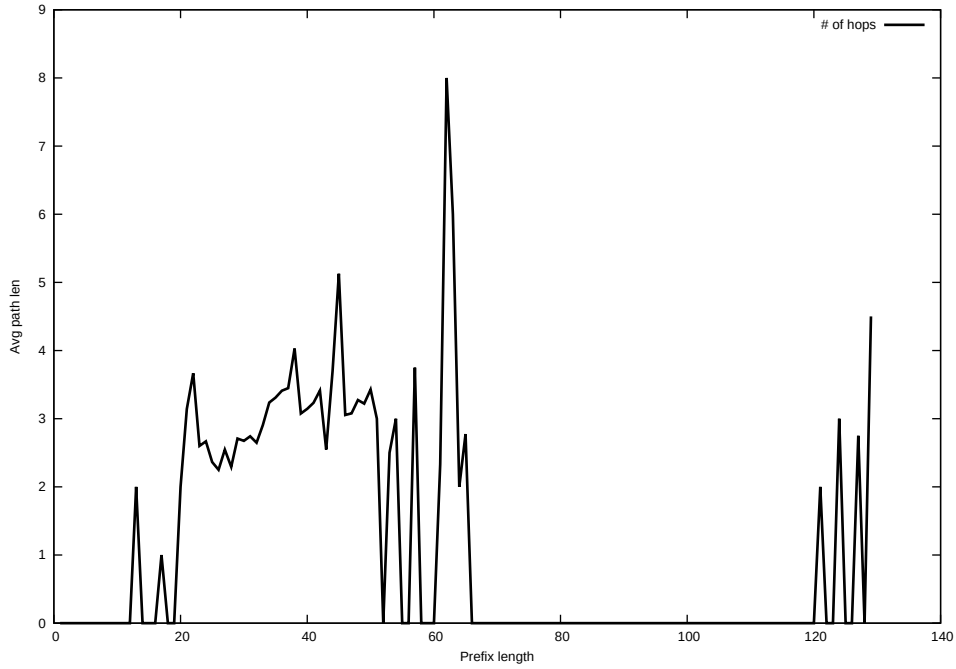
2014-01-11



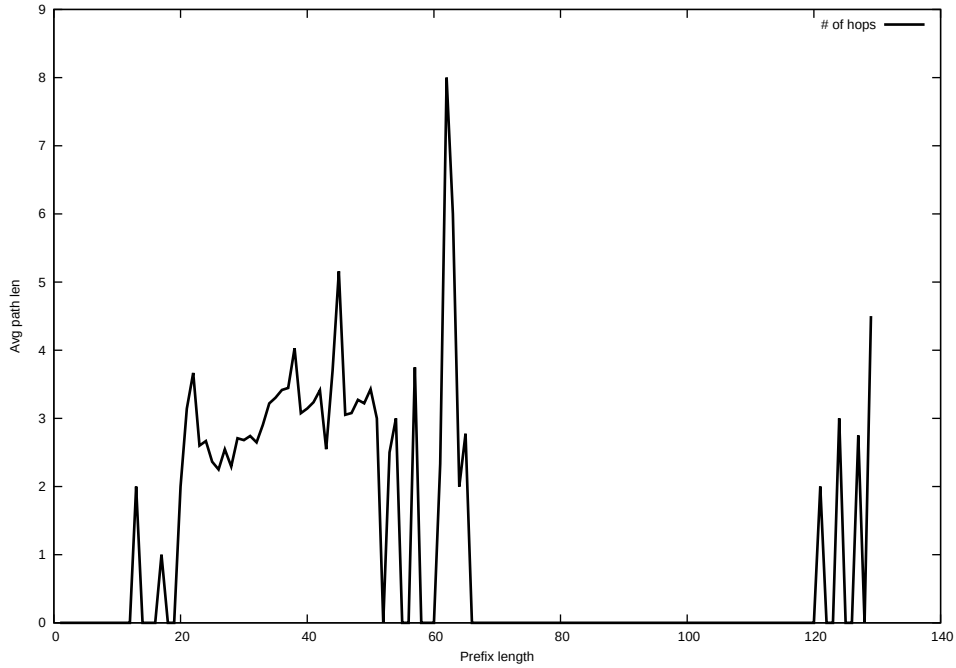
2014-01-12



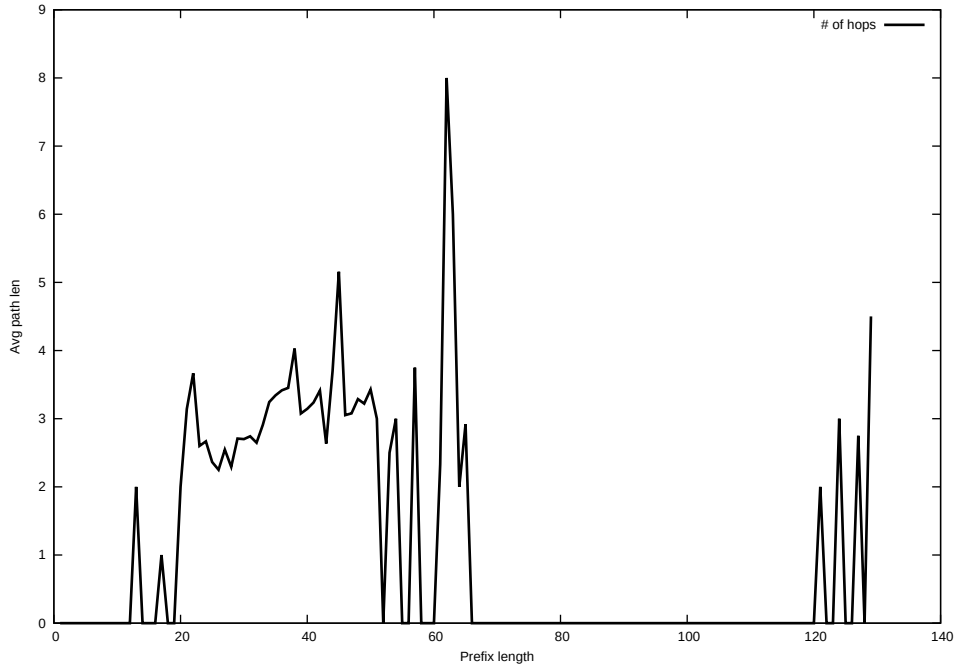
2014-01-13



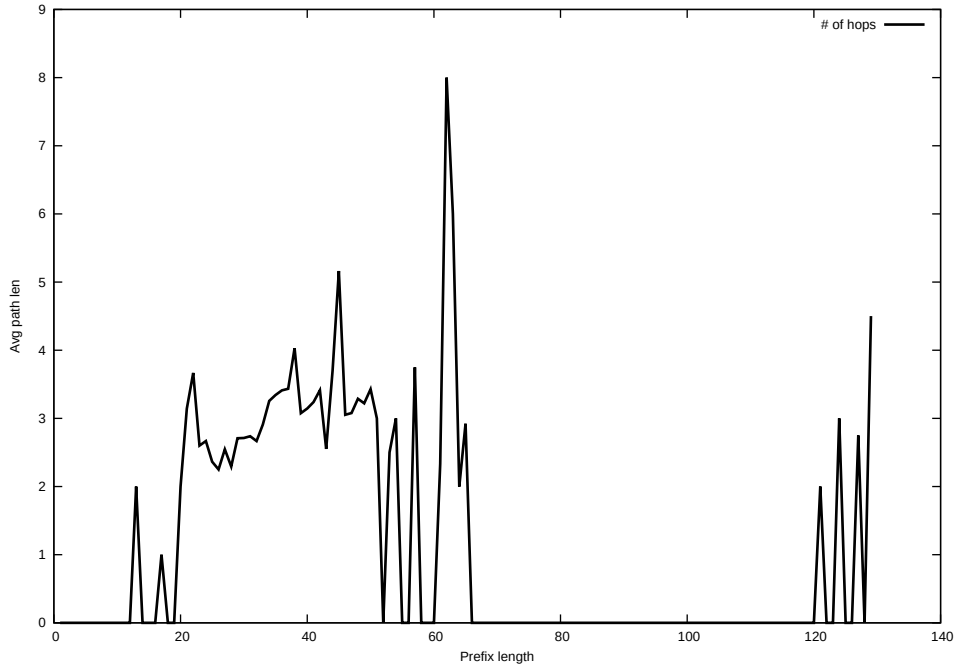
2014-01-14



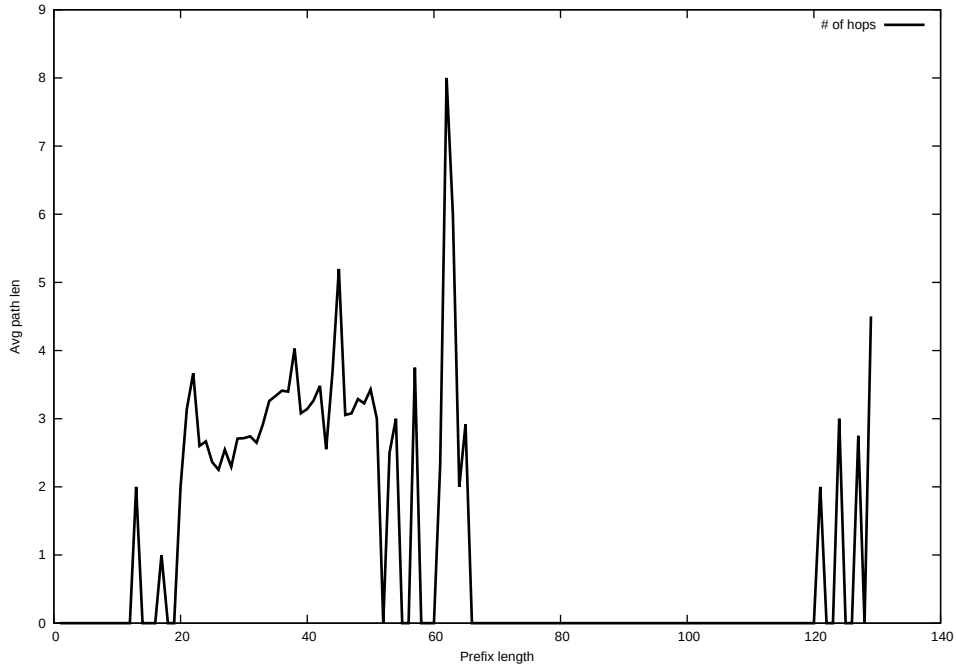
2014-01-15



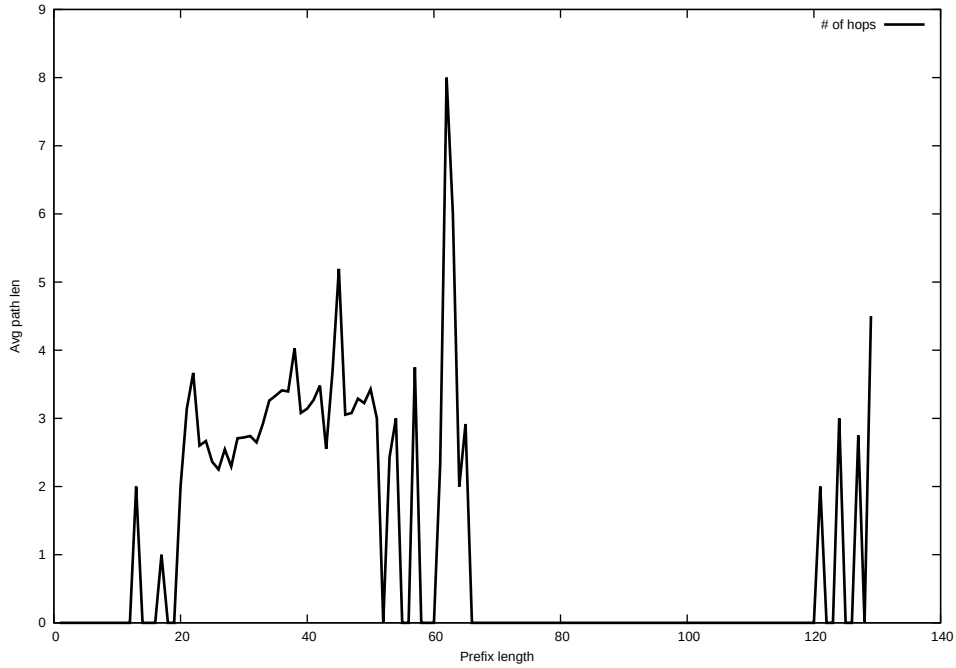
2014-01-16



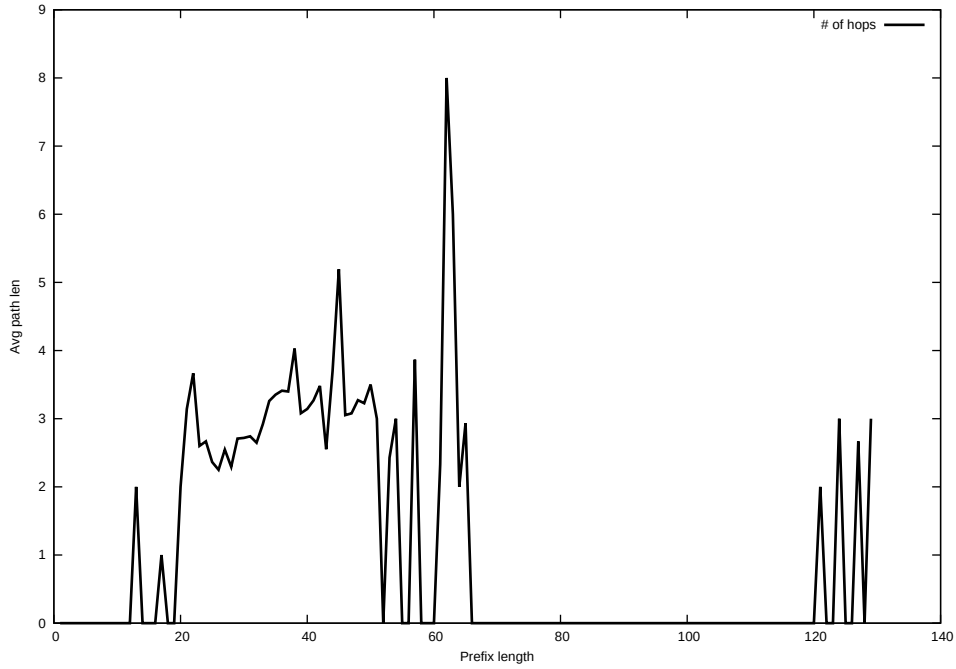
2014-01-17



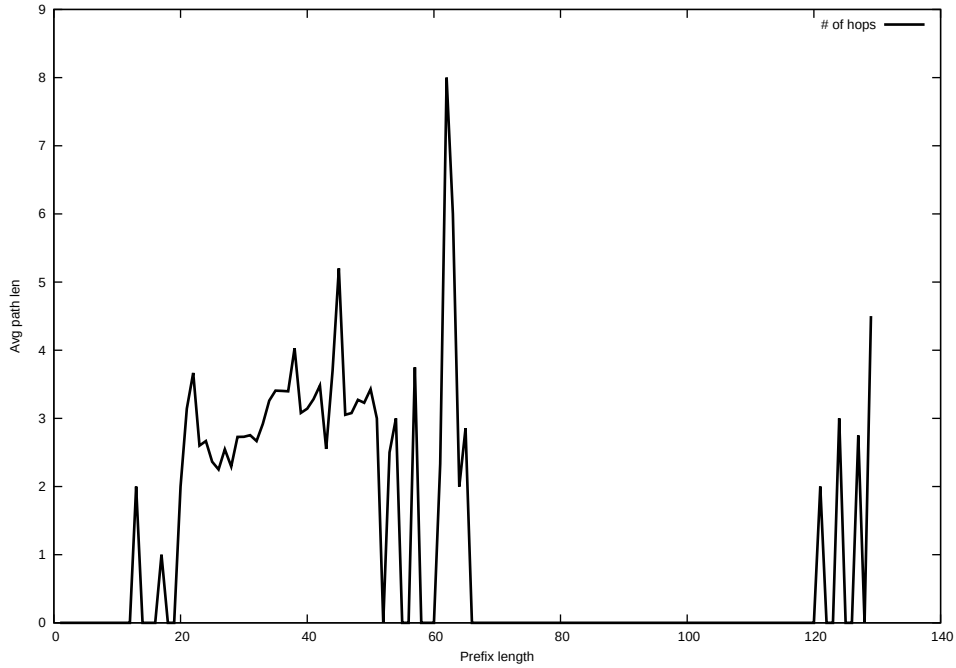
2014-01-18



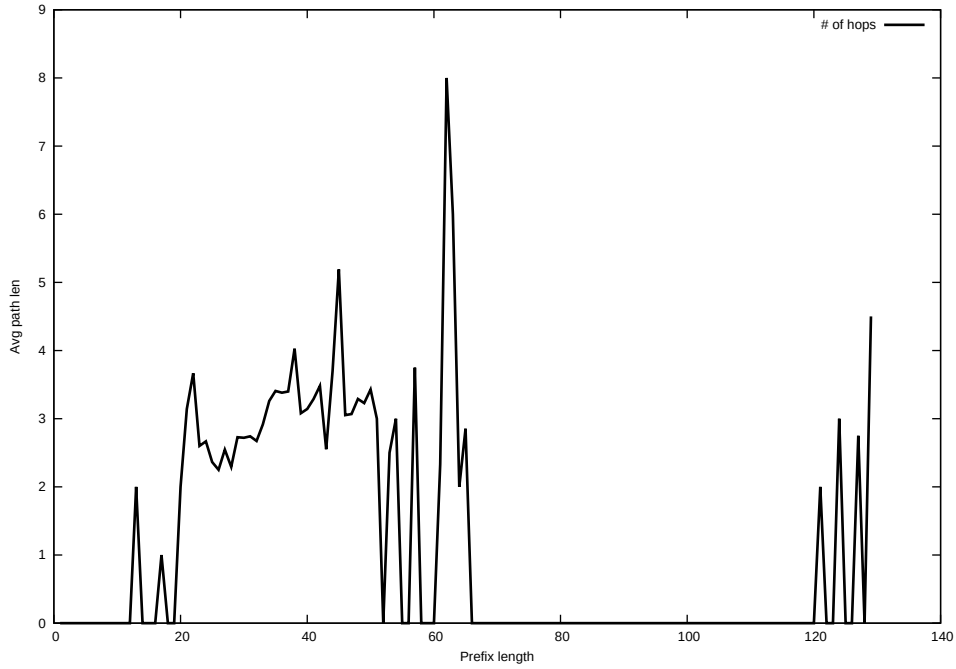
2014-01-19



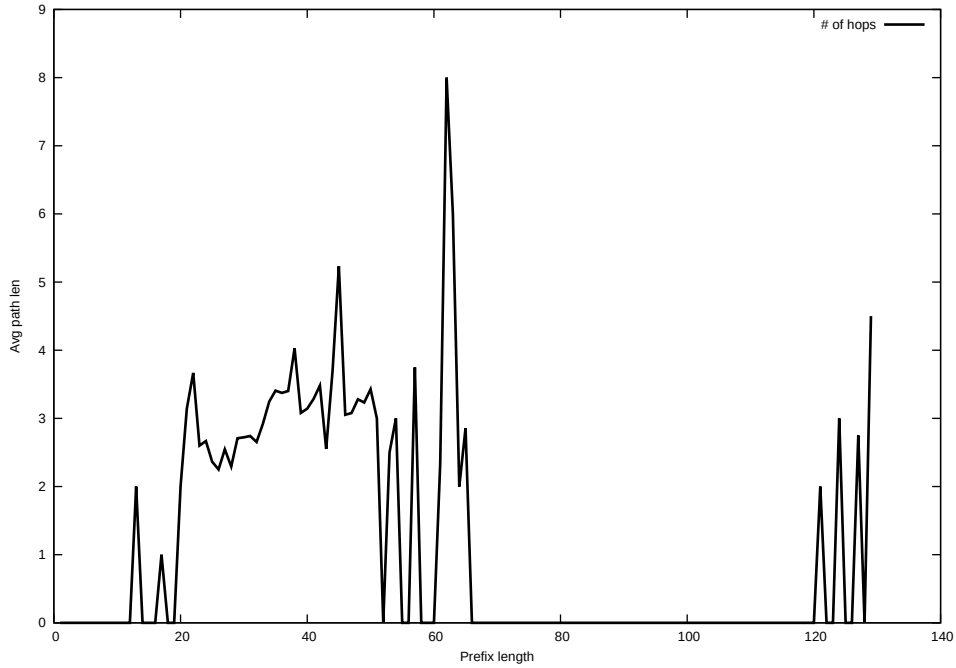
2014-01-20



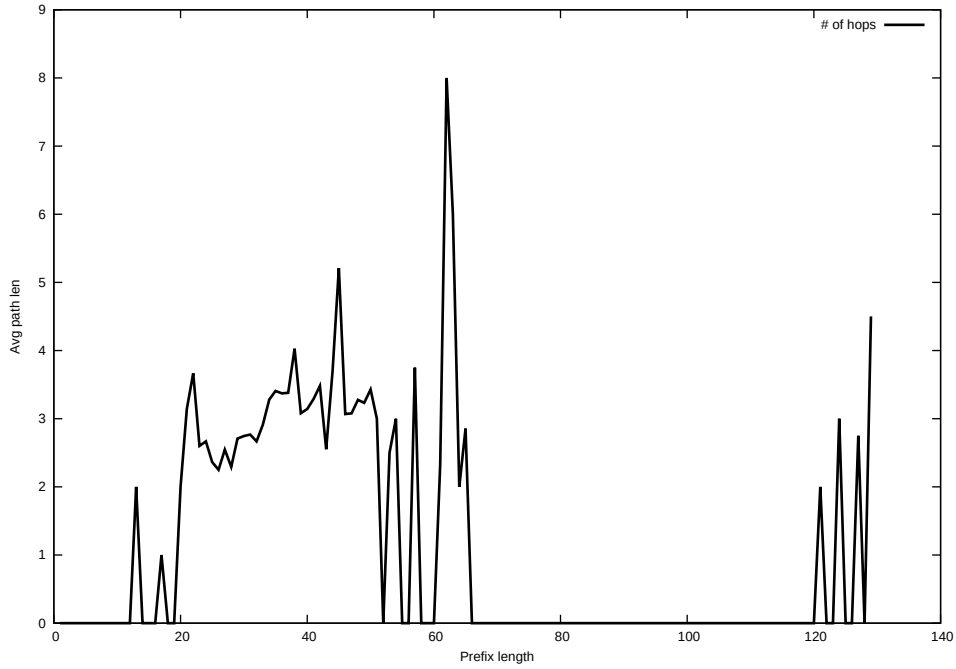
2014-01-21



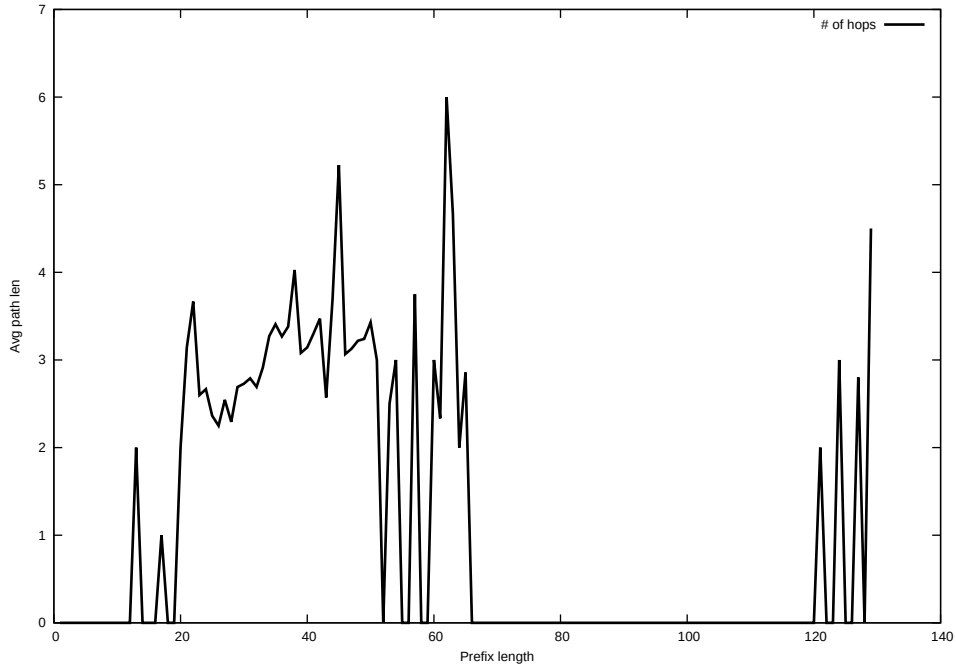
2014-01-22



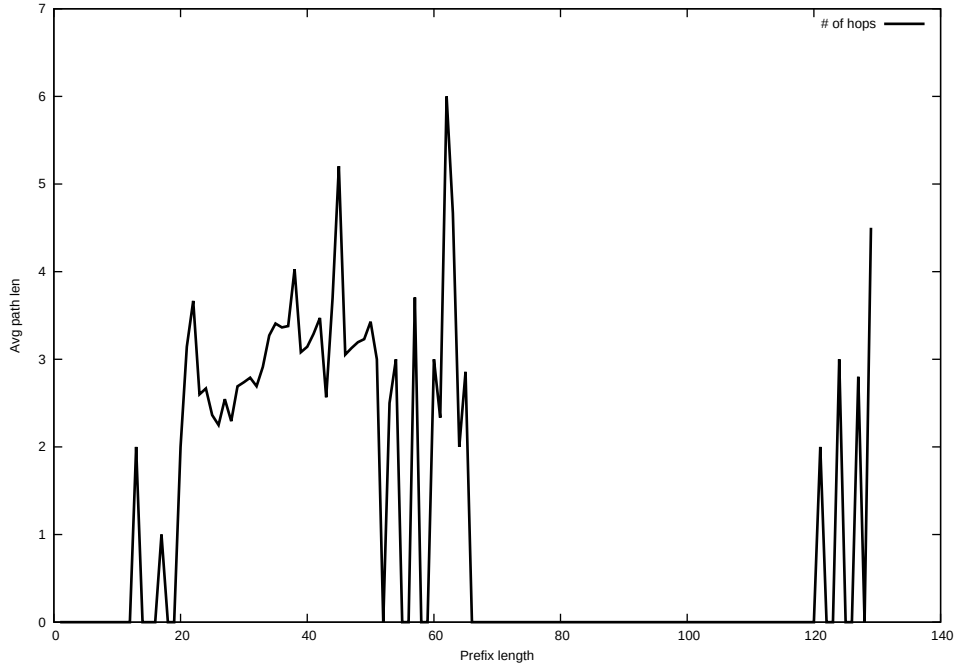
2014-01-23



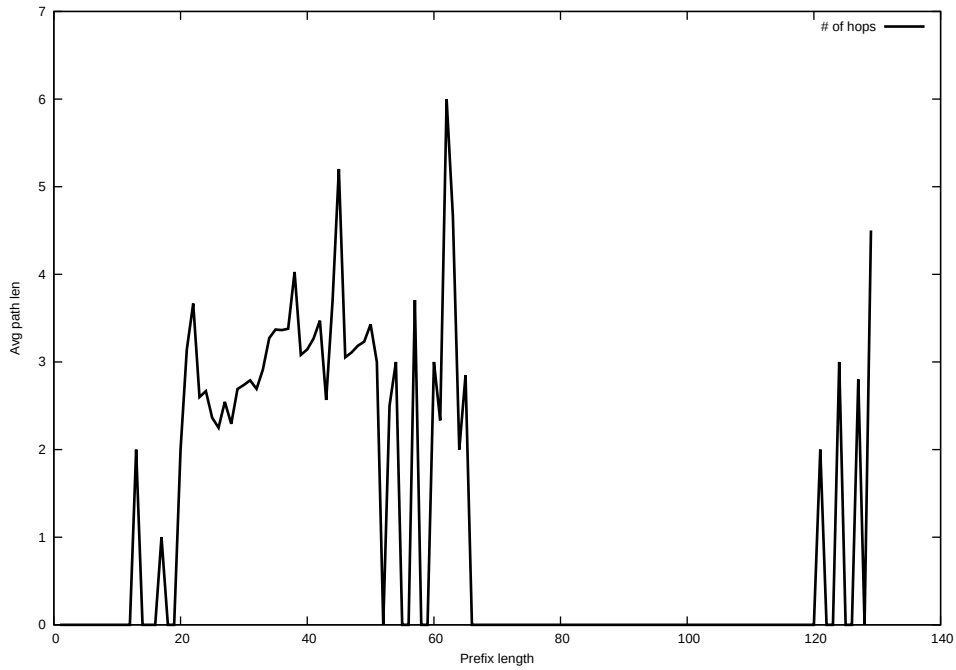
2014-01-24



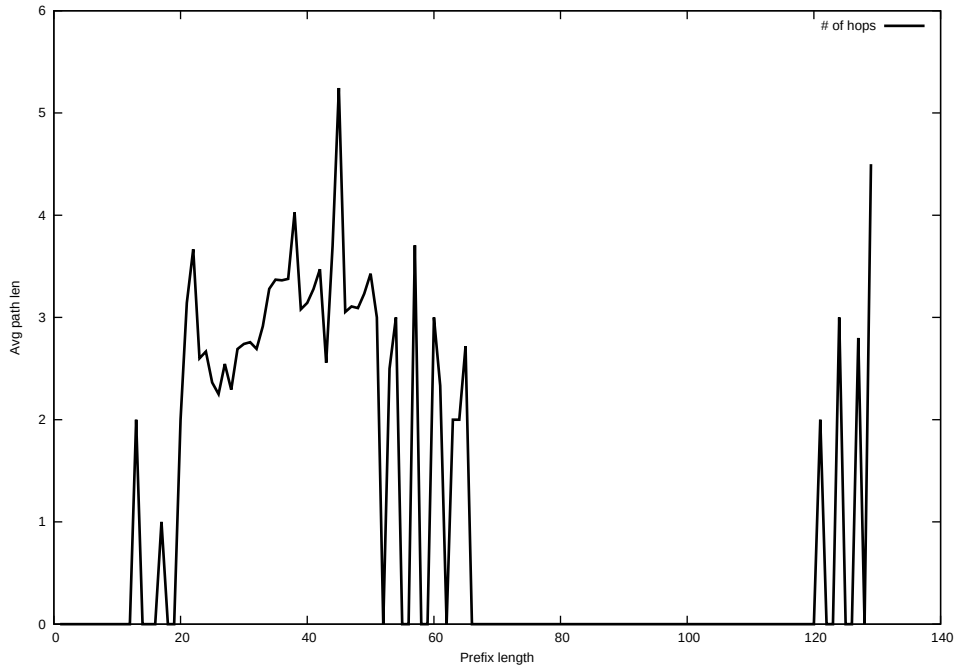
2014-01-25



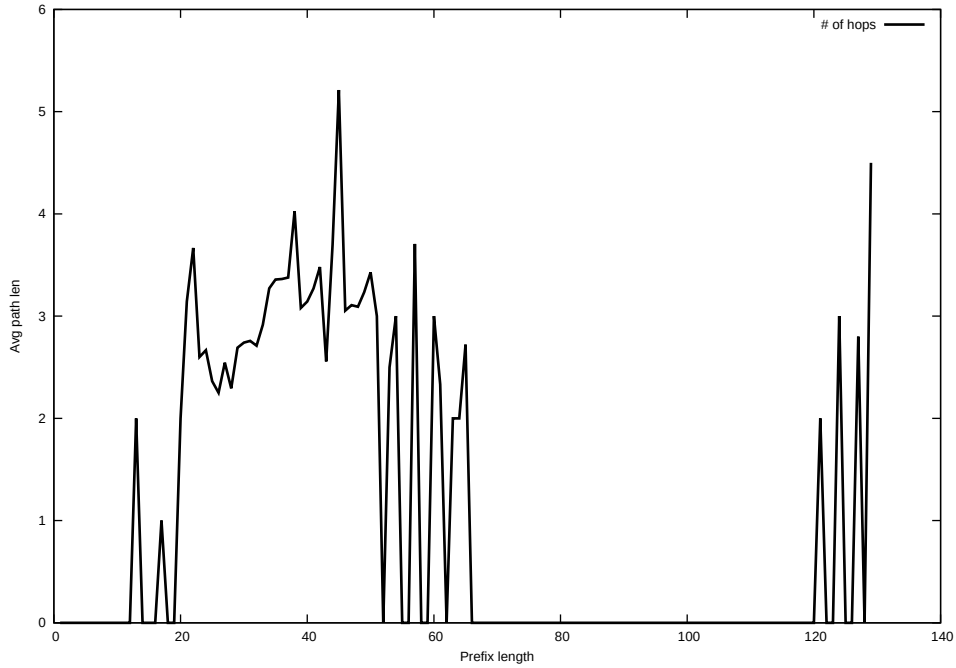
2014-01-26



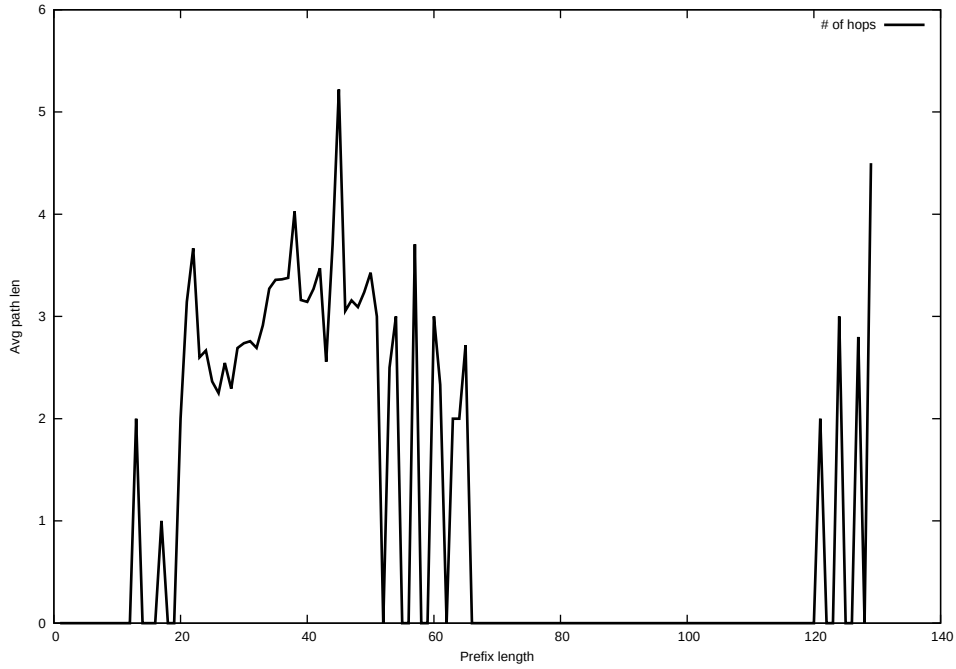
2014-01-27



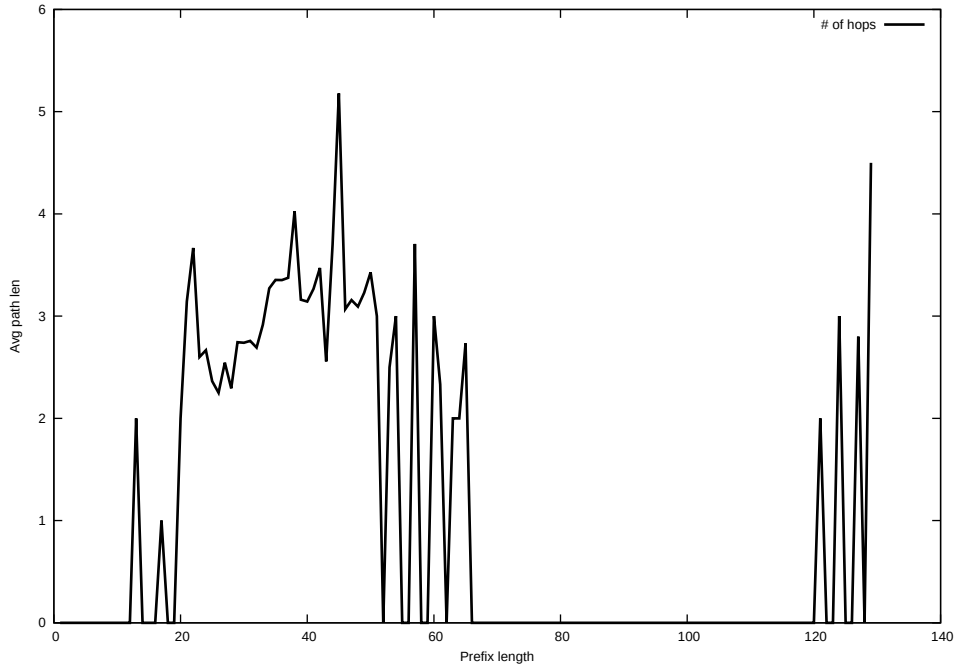
2014-01-28



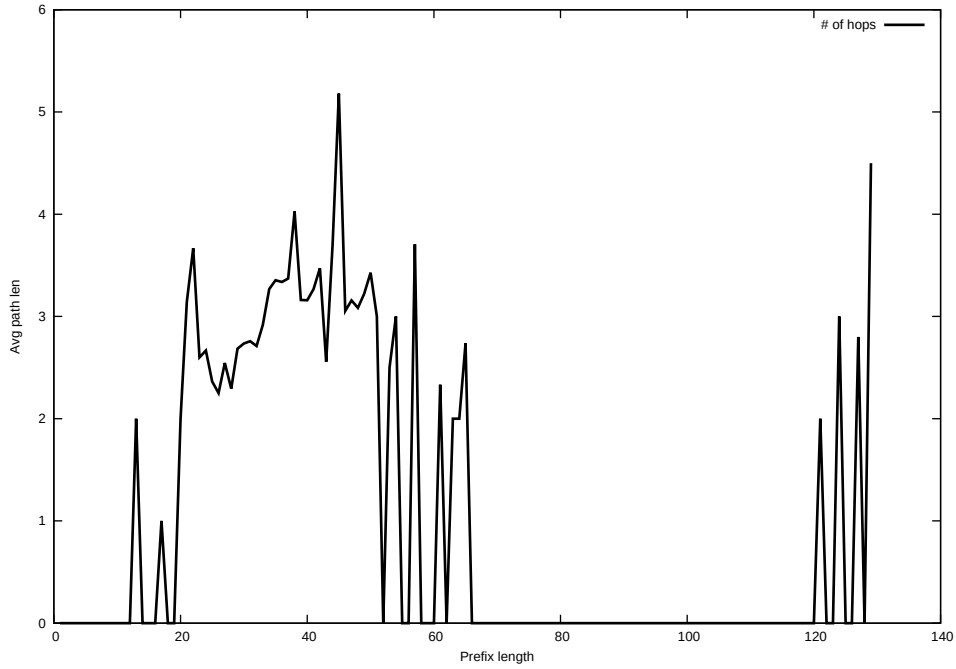
2014-01-29



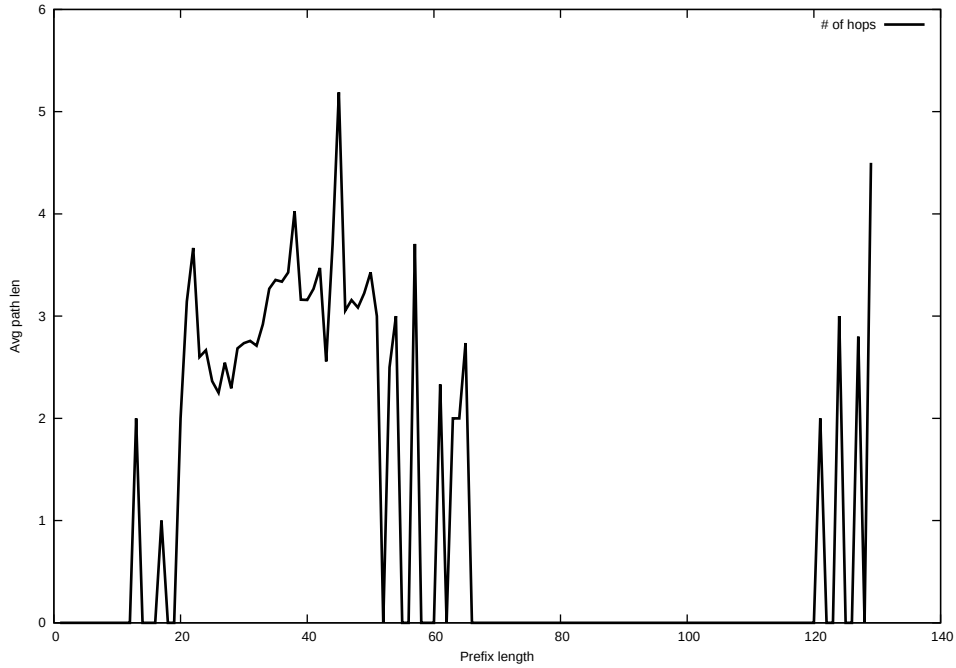
2014-01-30



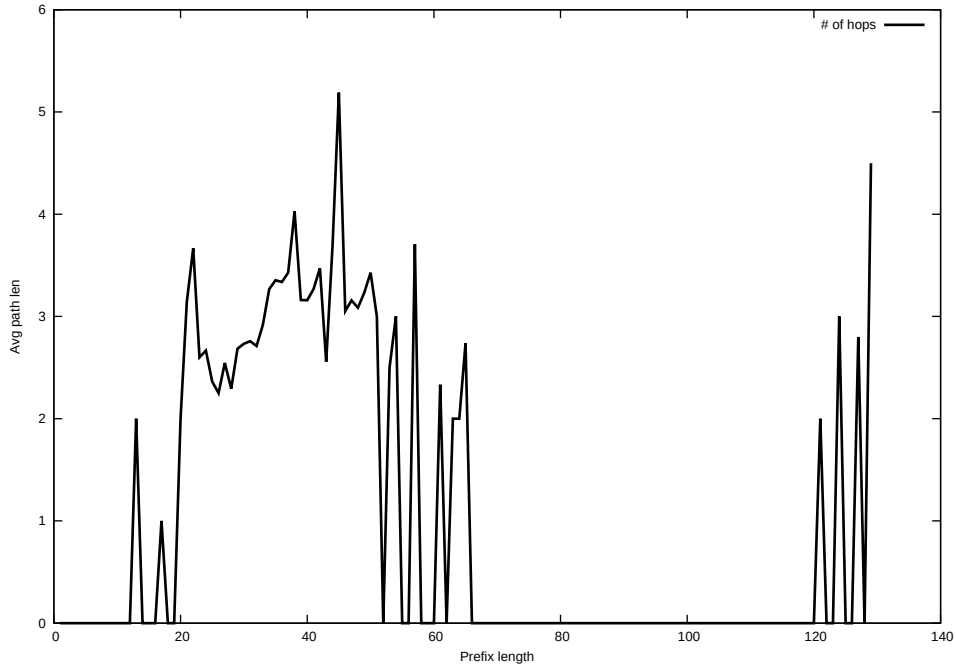
2014-01-31



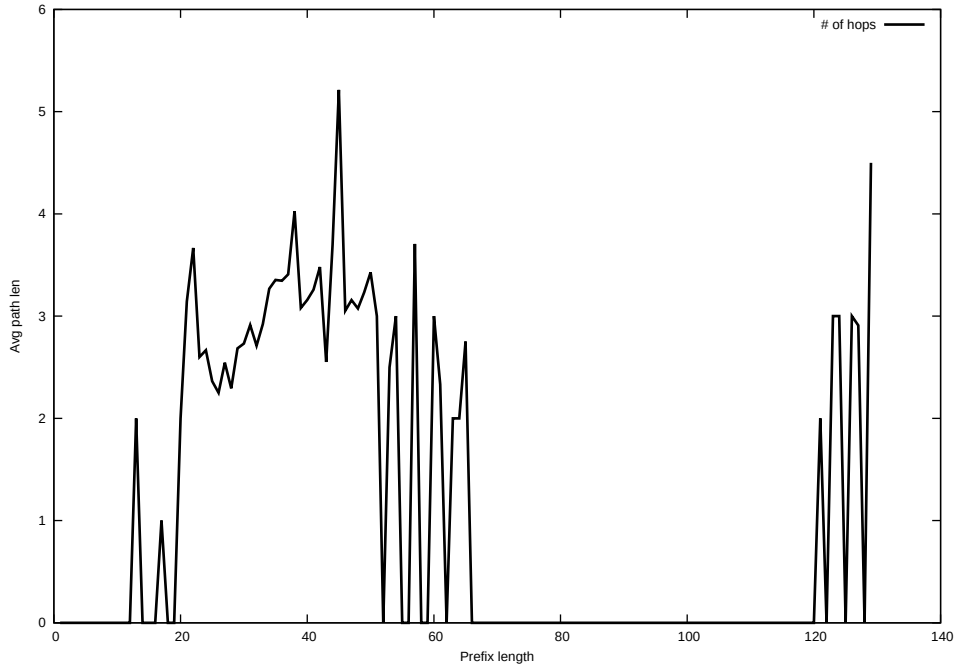
2014-02-01



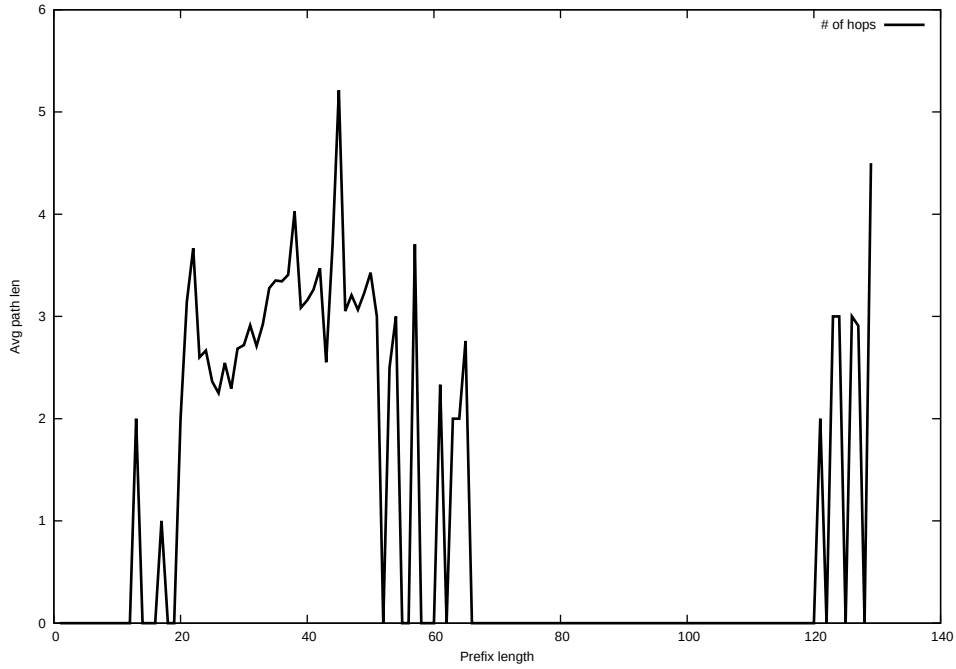
2014-02-02



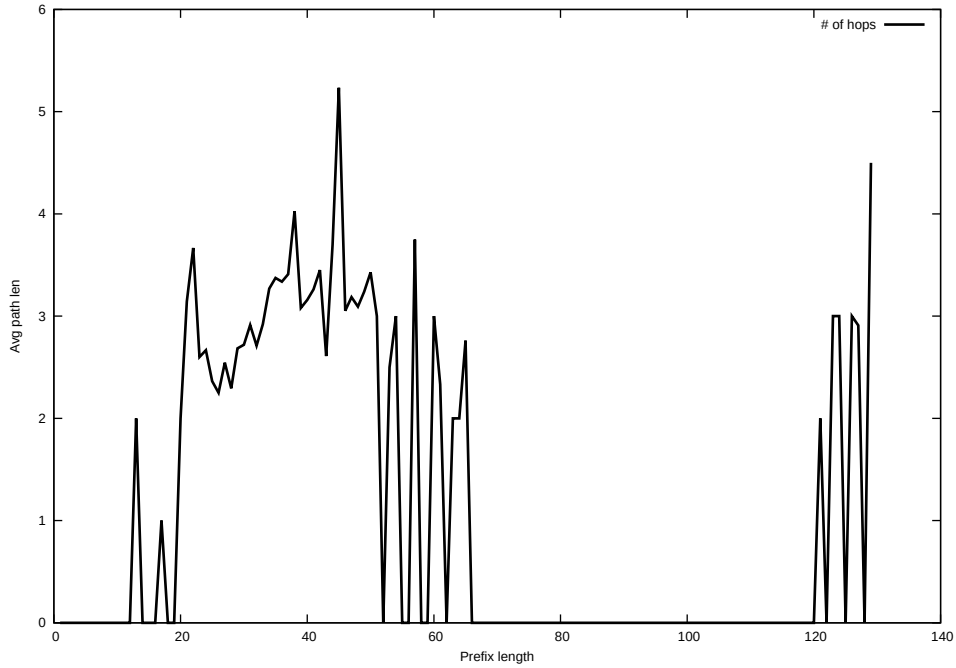
2014-02-03



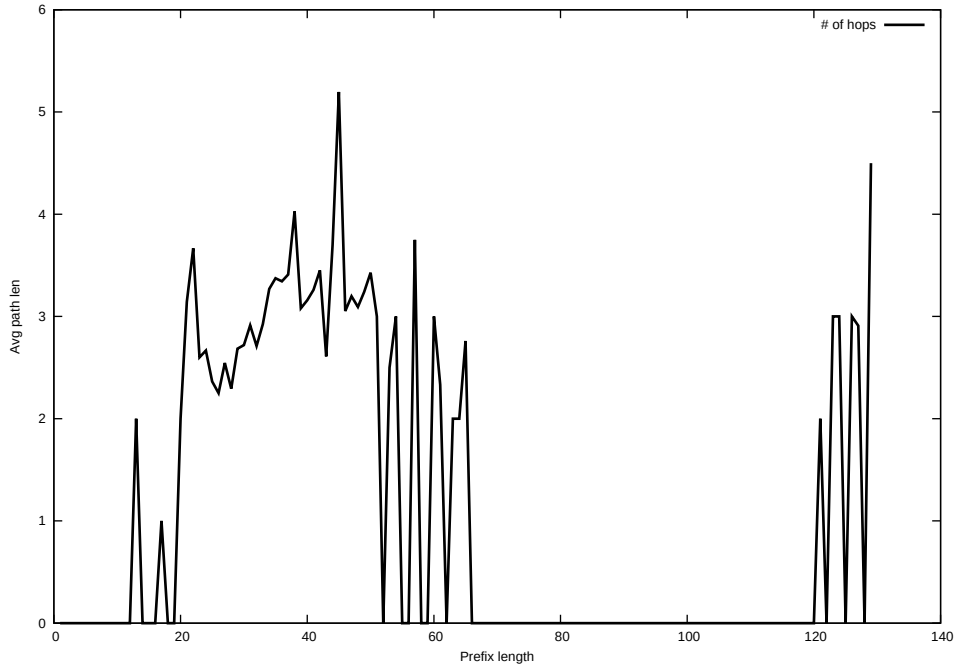
2014-02-04



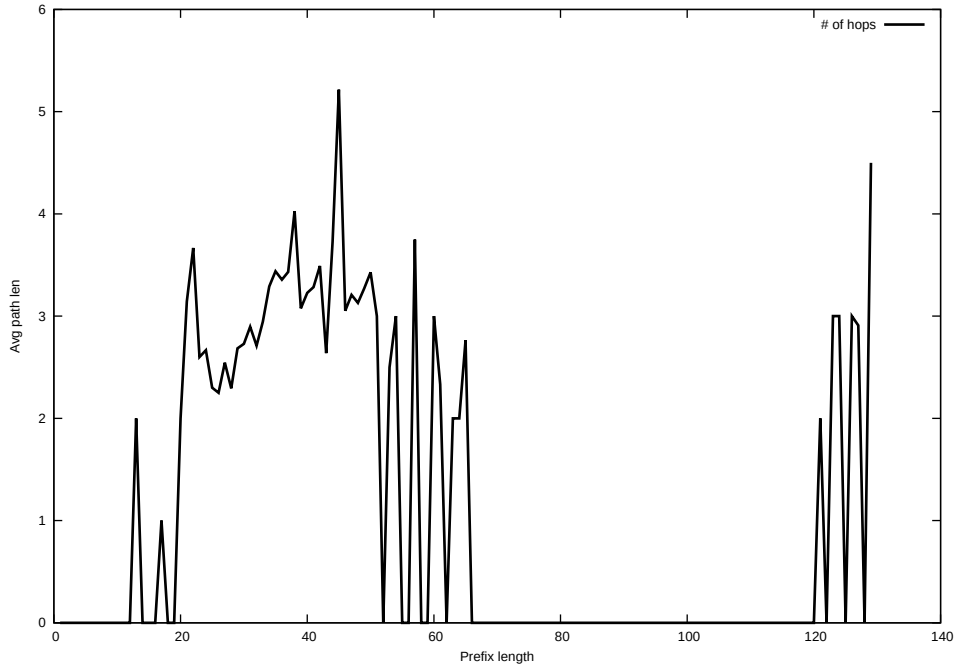
2014-02-05



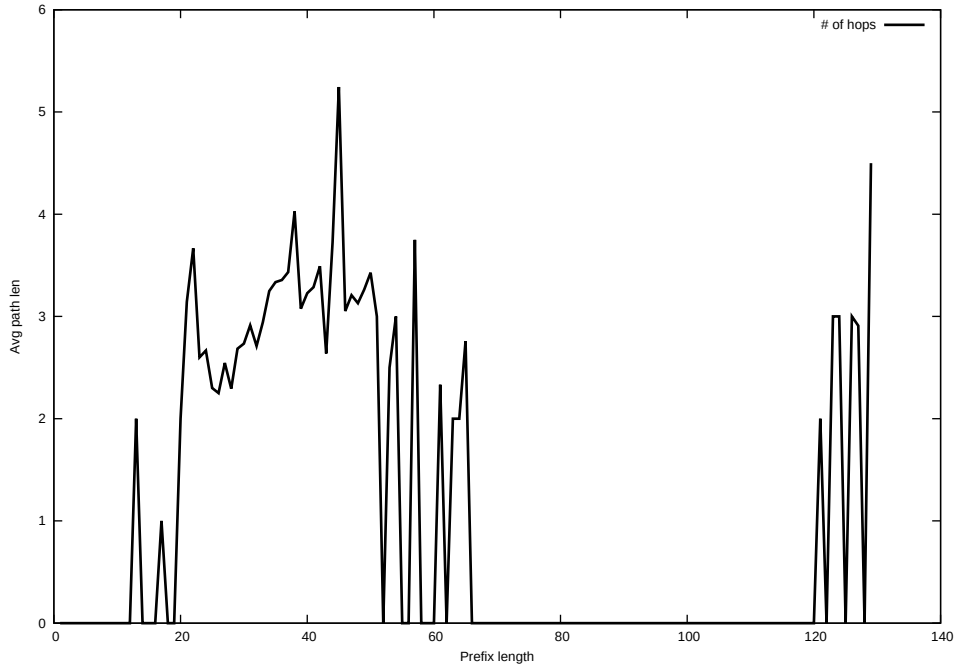
2014-02-06



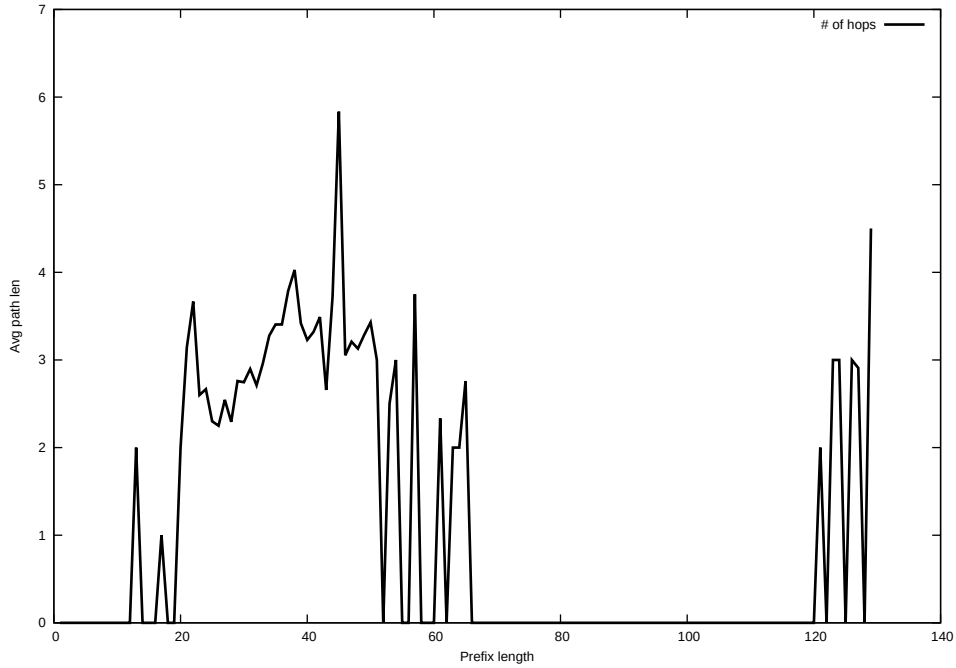
2014-02-07



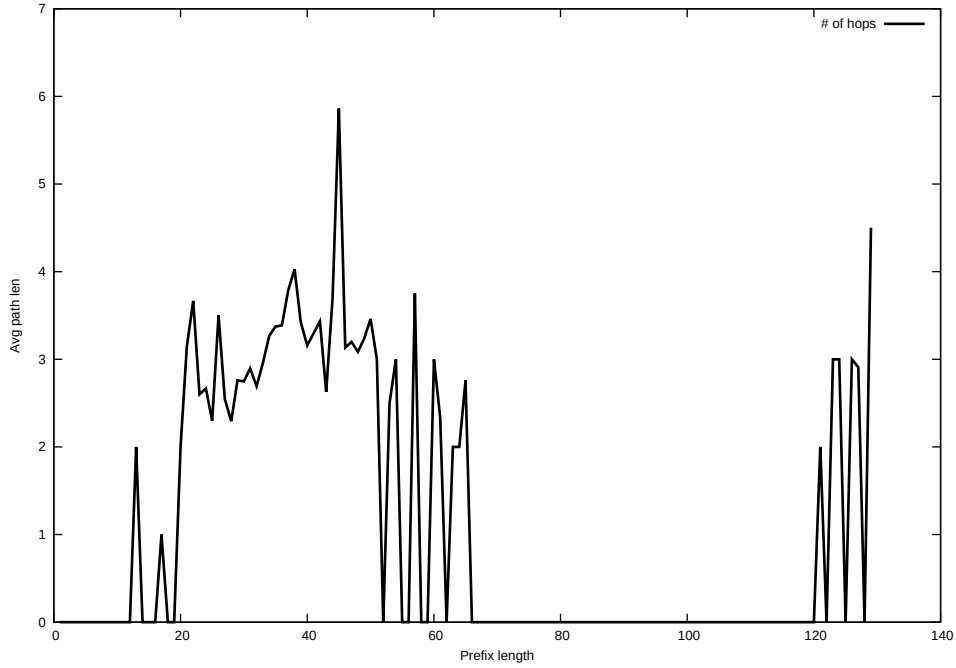
2014-02-08



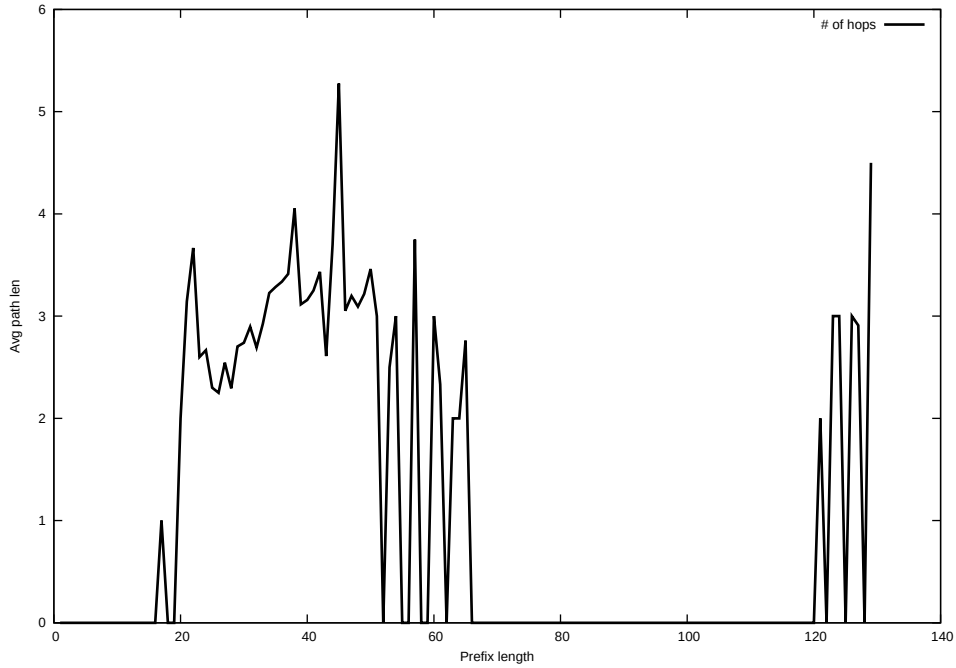
2014-02-09



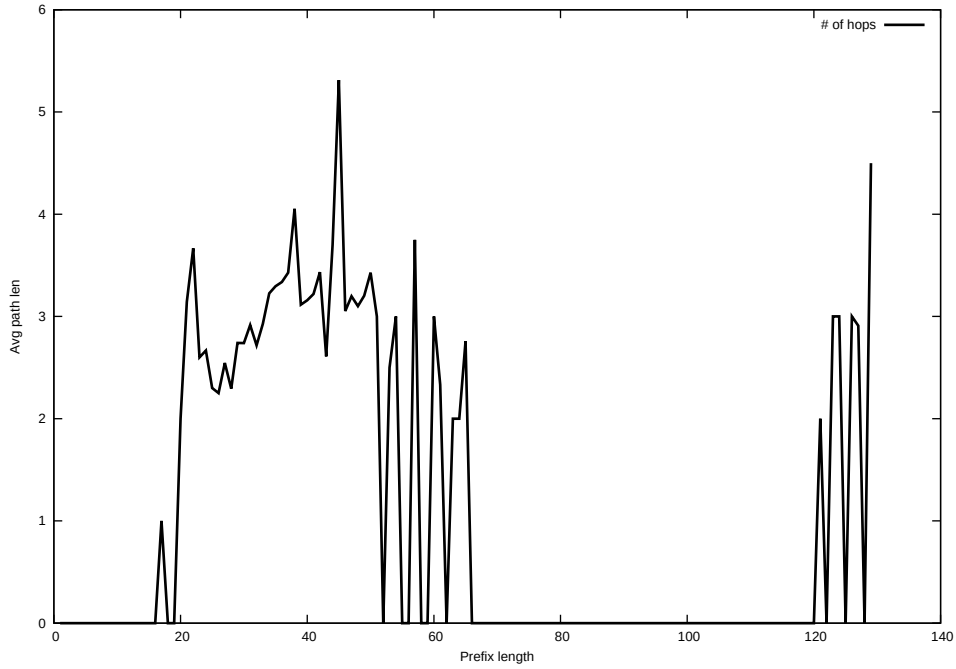
2014-02-10



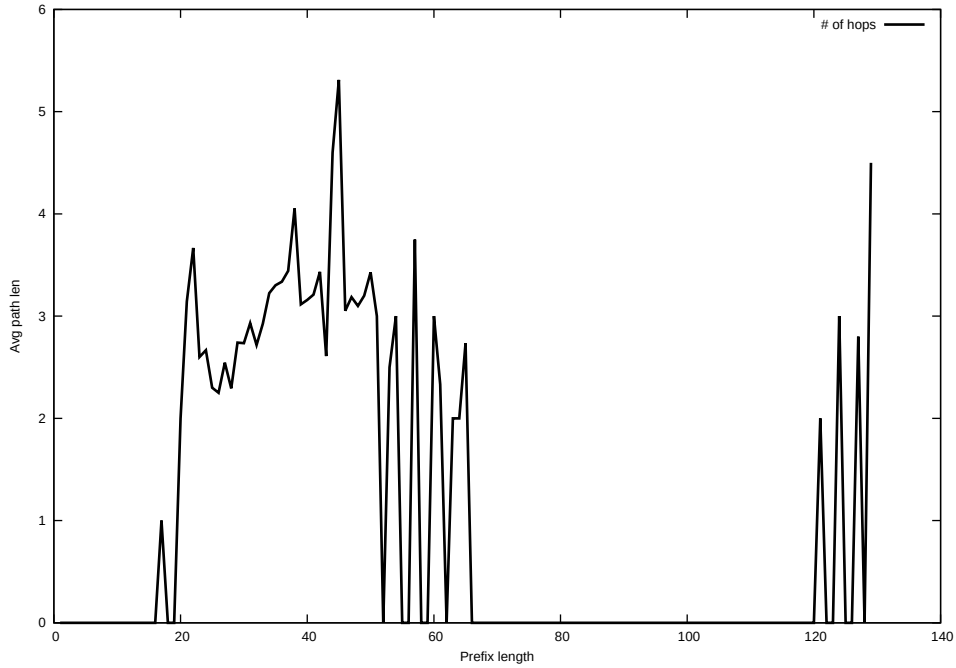
2014-02-11



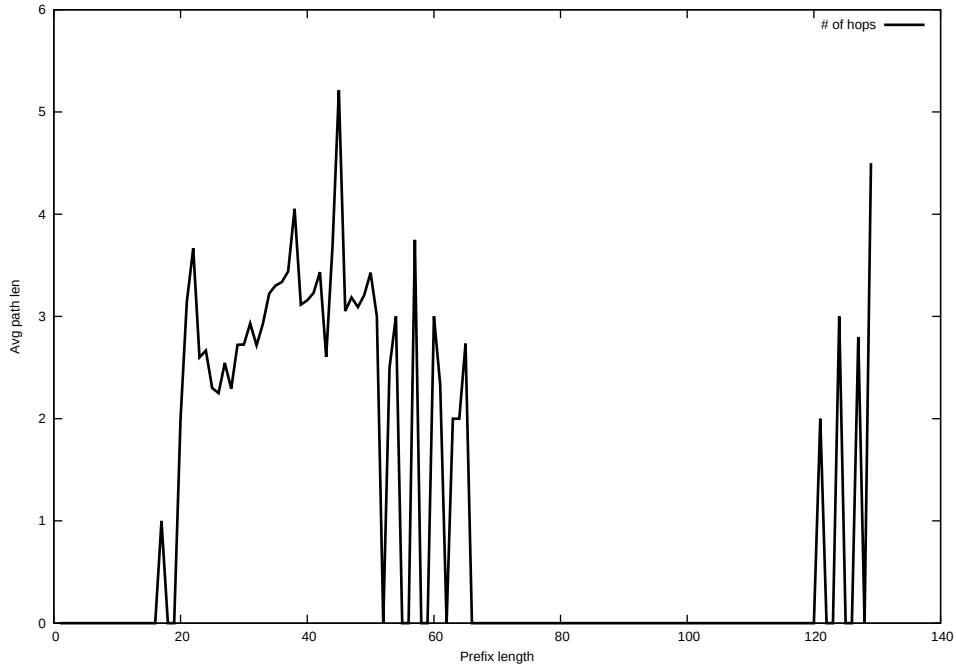
2014-02-12



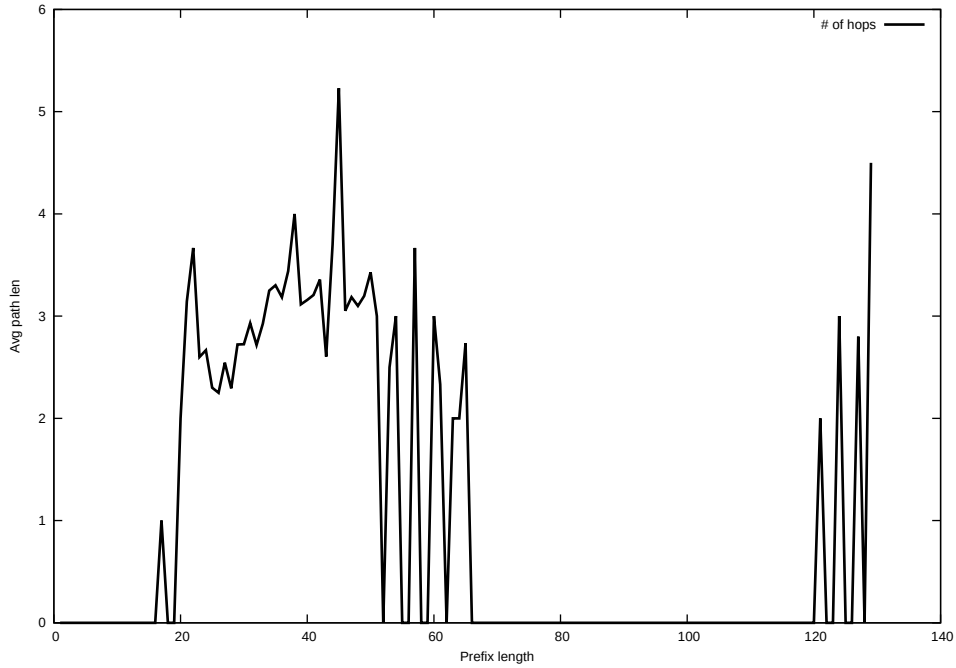
2014-02-13



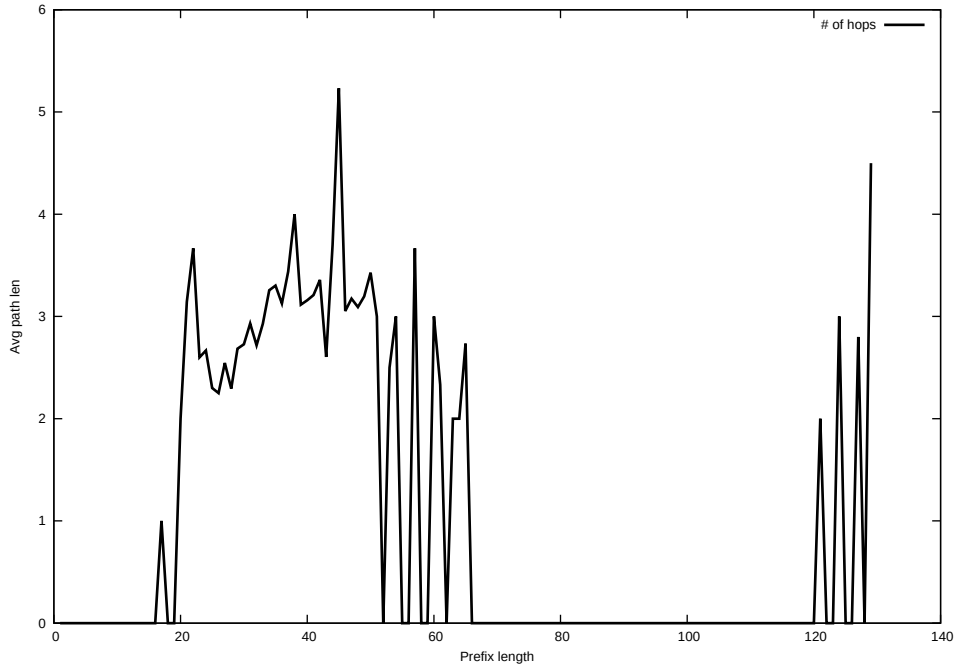
2014-02-14



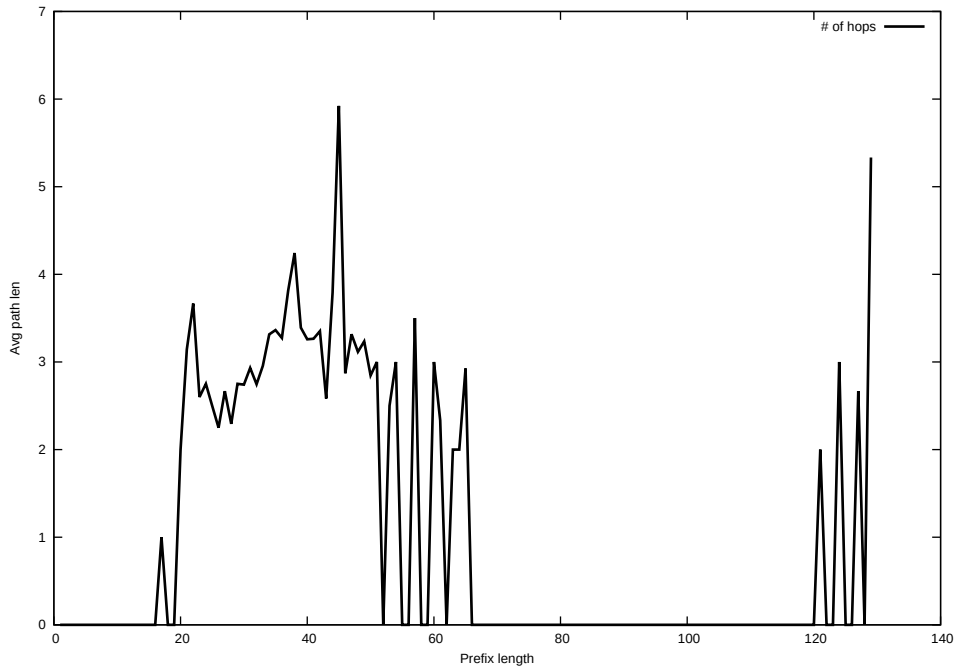
2014-02-15



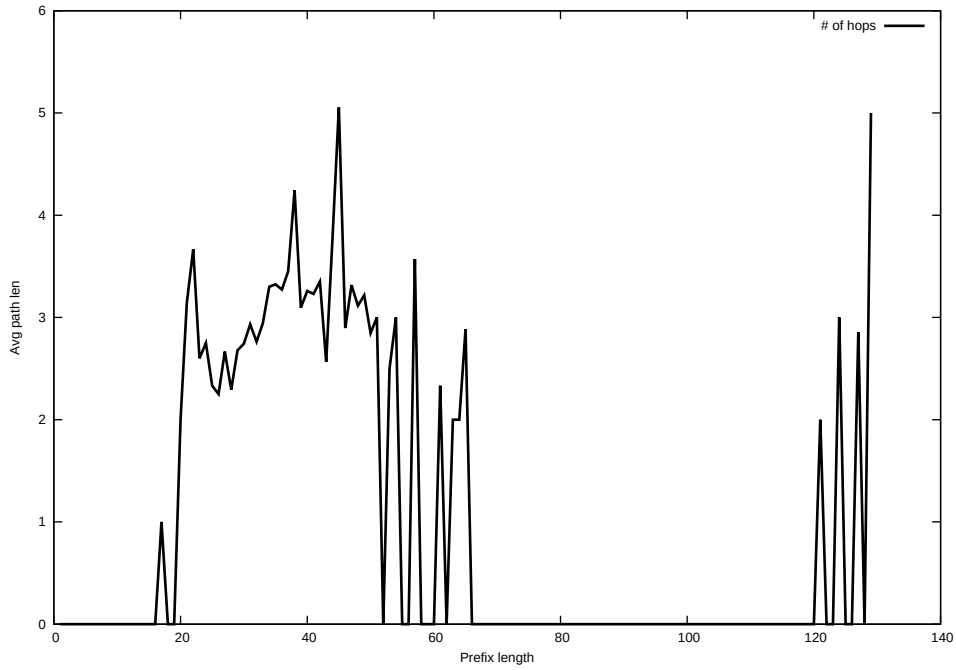
2014-02-16



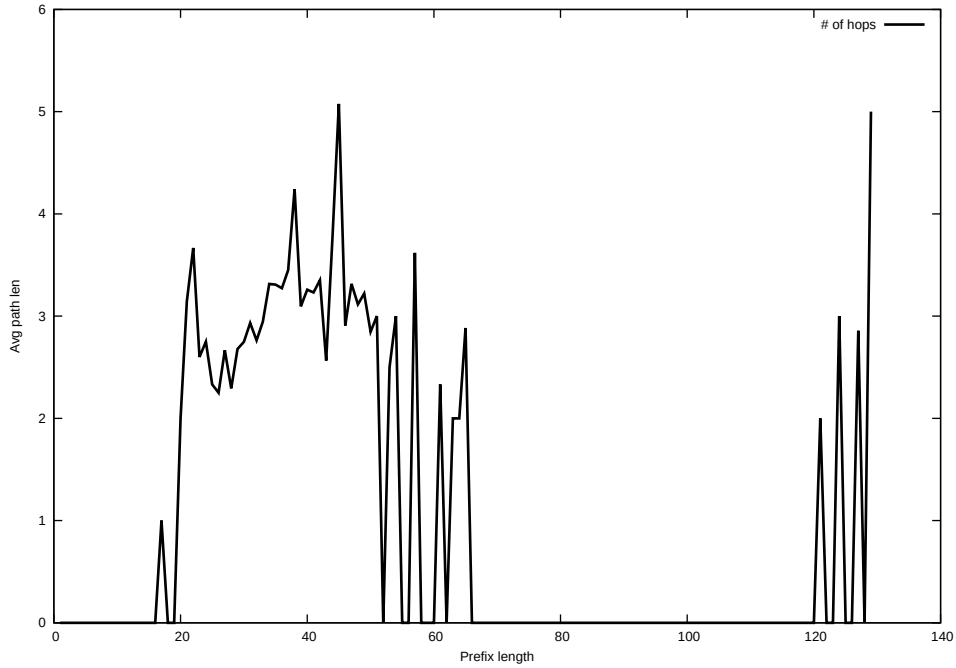
2014-02-17



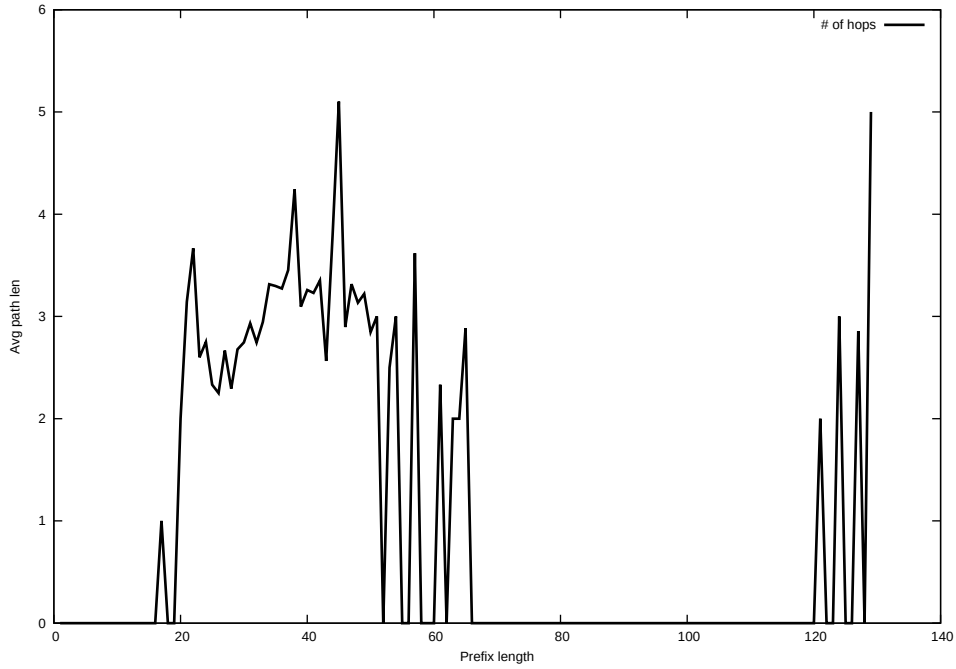
2014-03-20



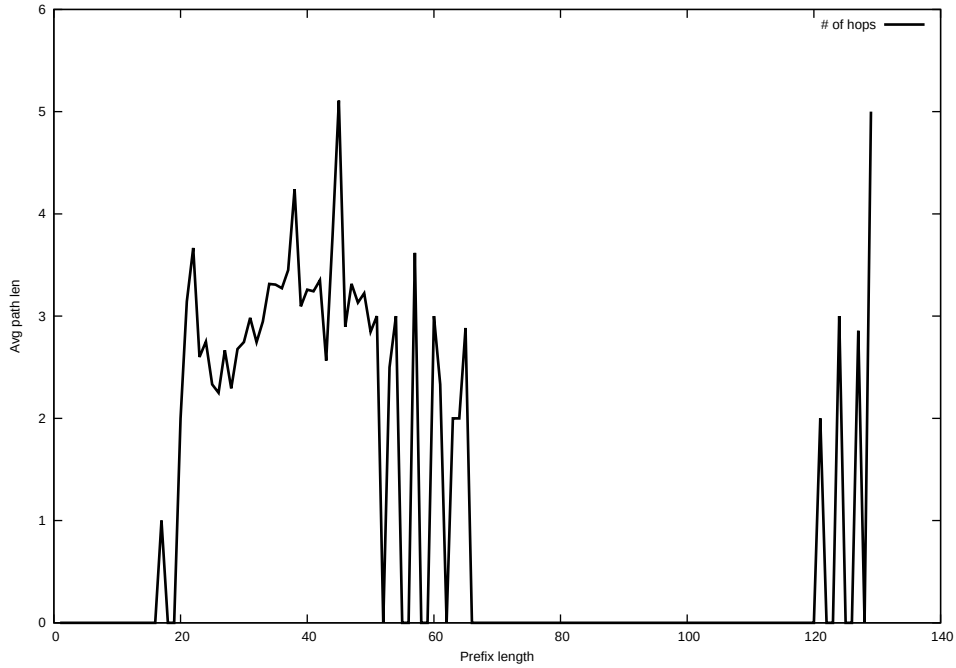
2014-03-21



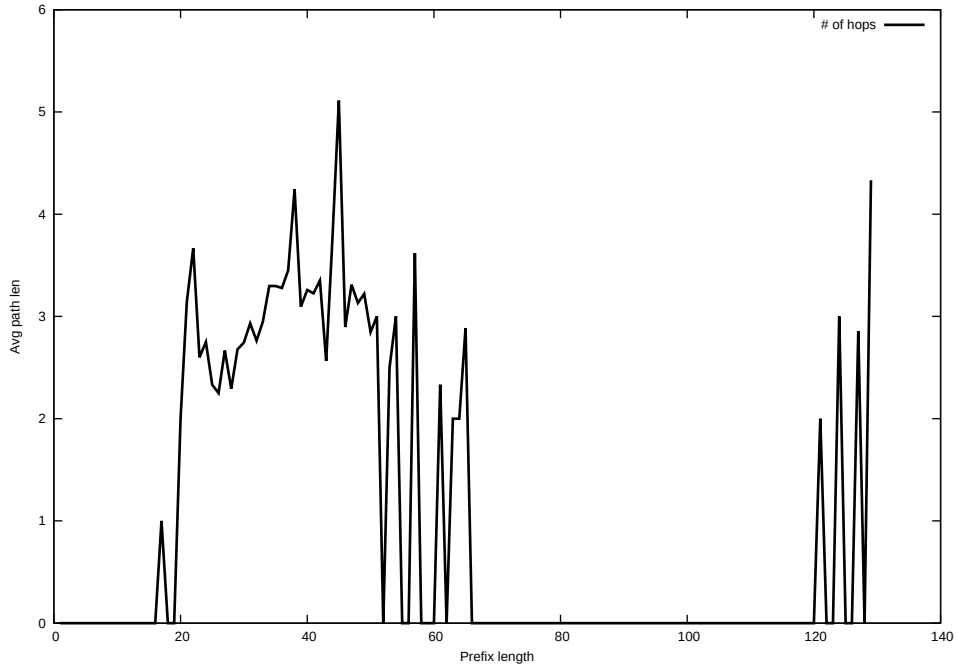
2014-03-22



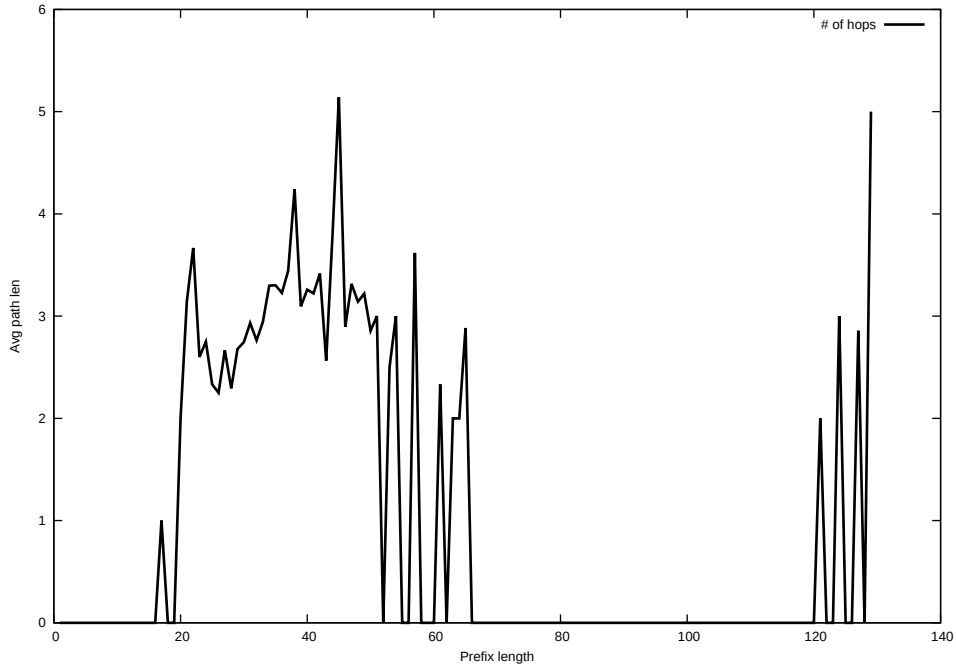
2014-03-23



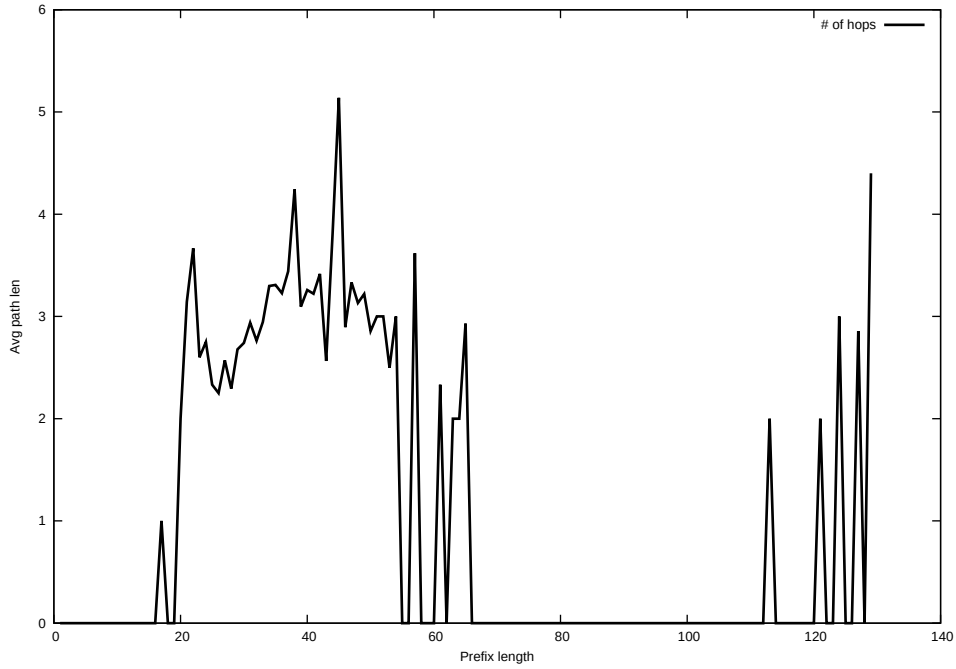
2014-03-24



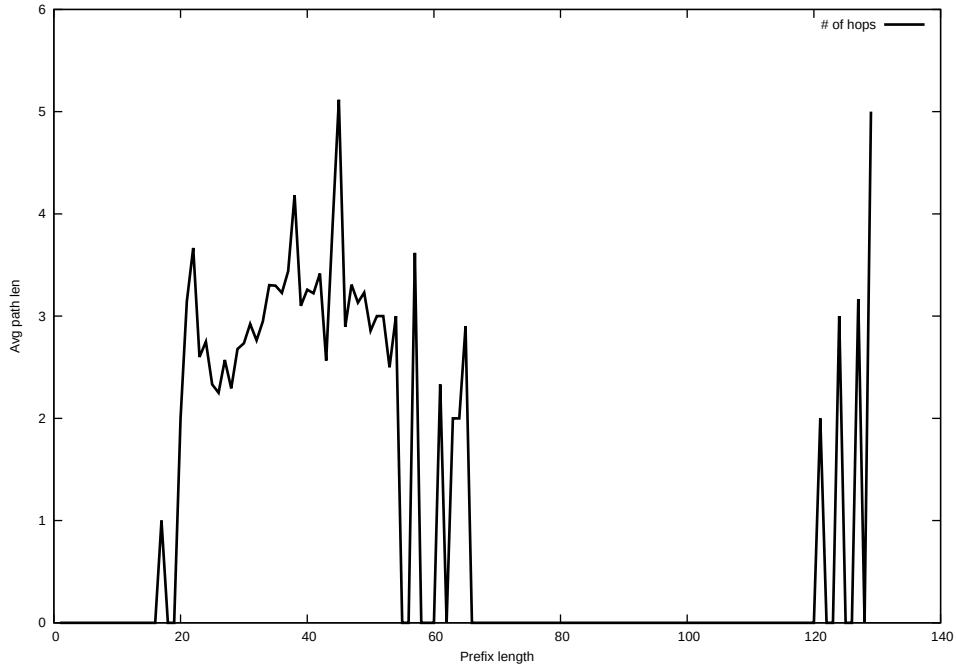
2014-03-25



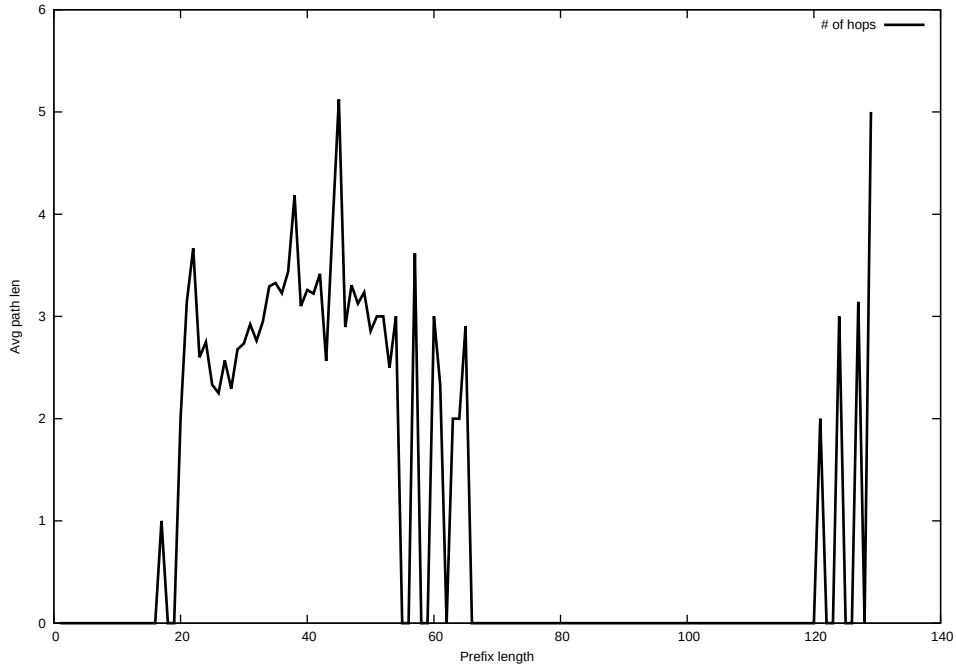
2014-03-26



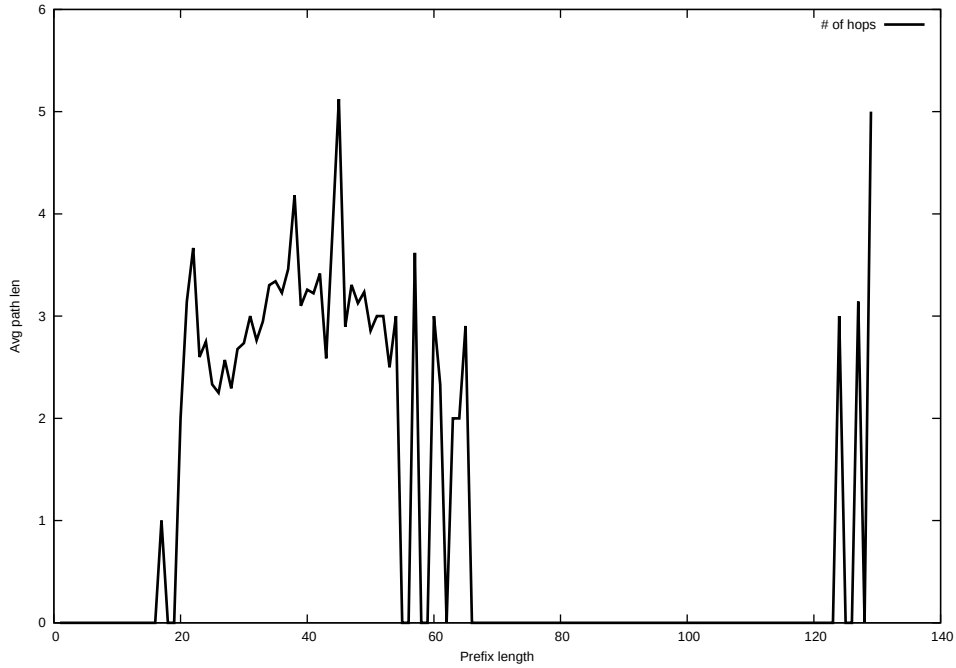
2014-03-27



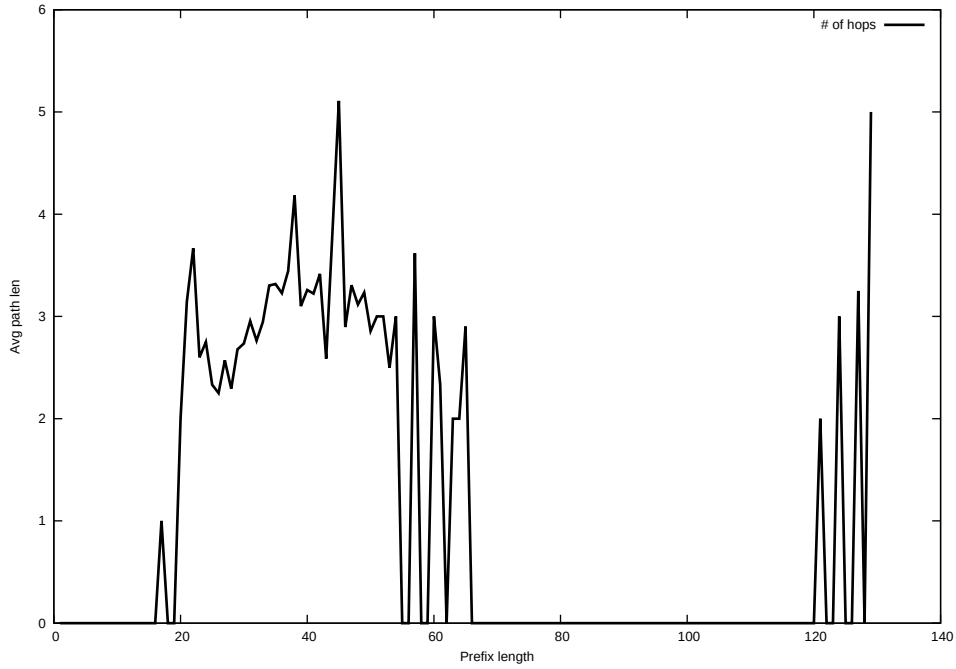
2014-03-28



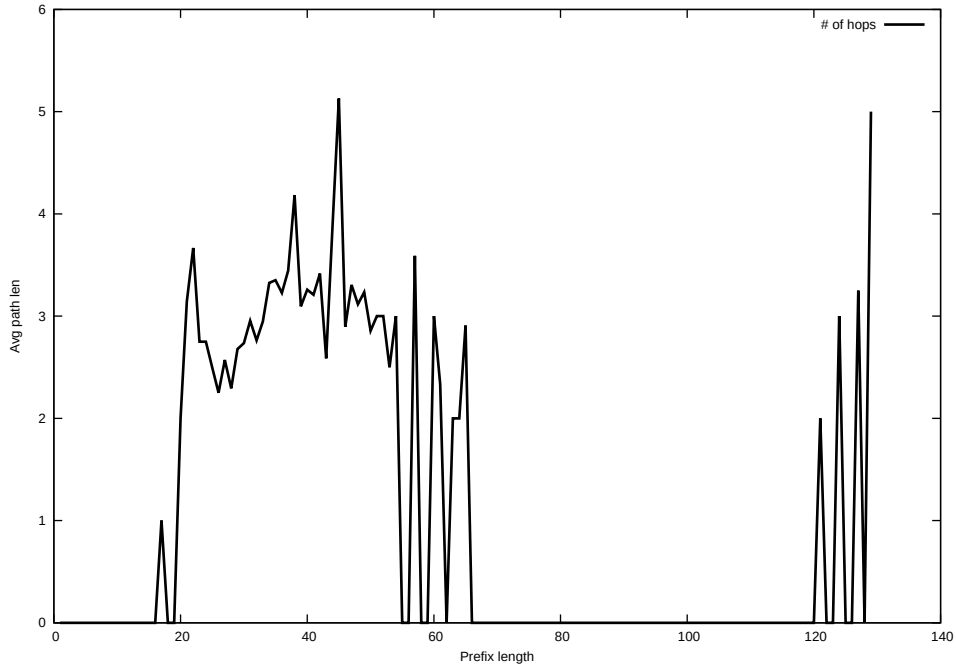
2014-03-29



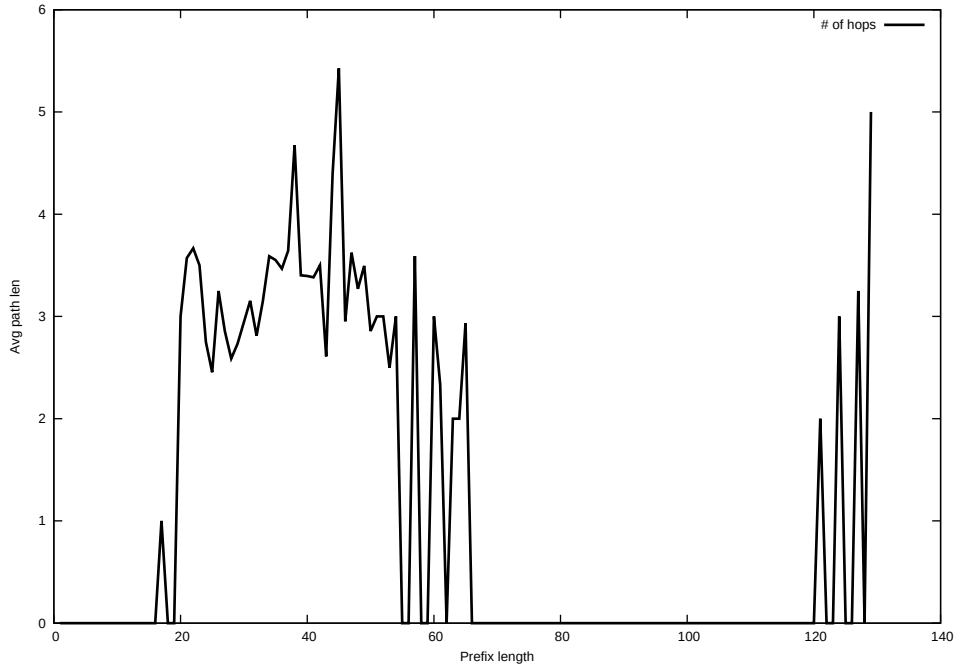
2014-03-30



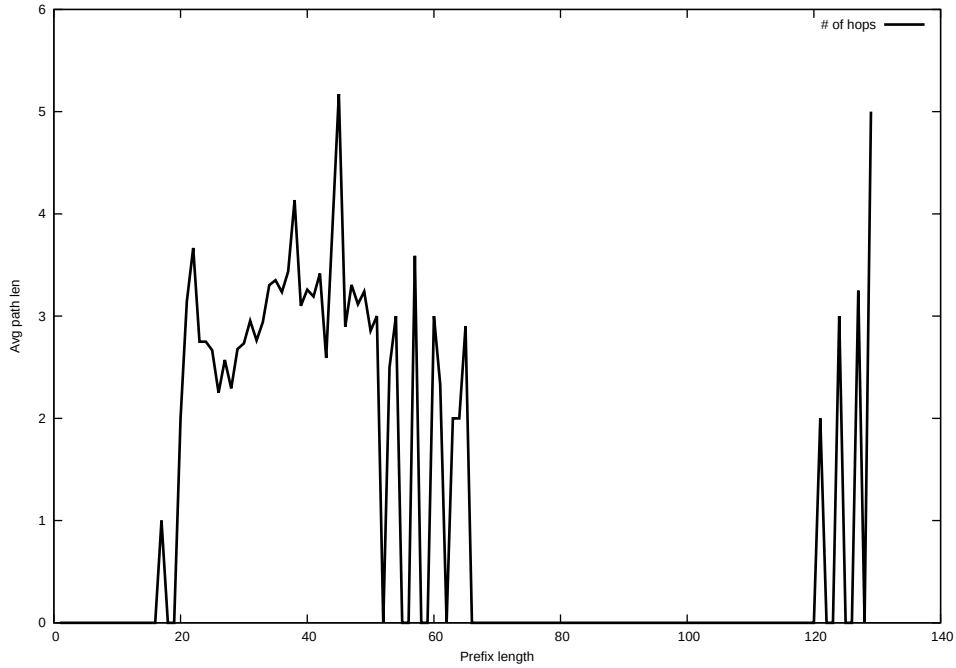
2014-03-31



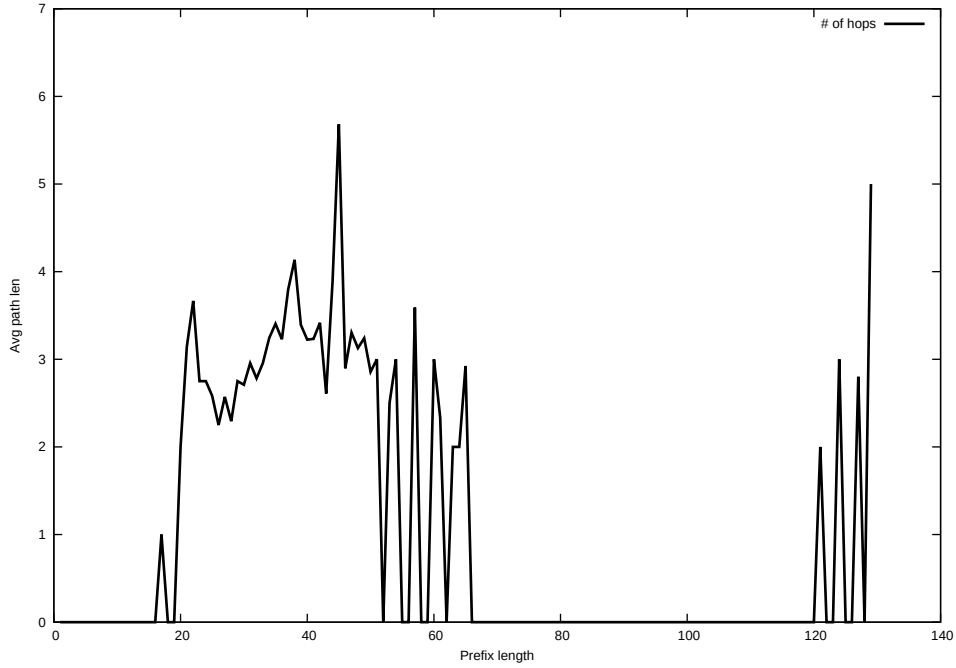
2014-04-01



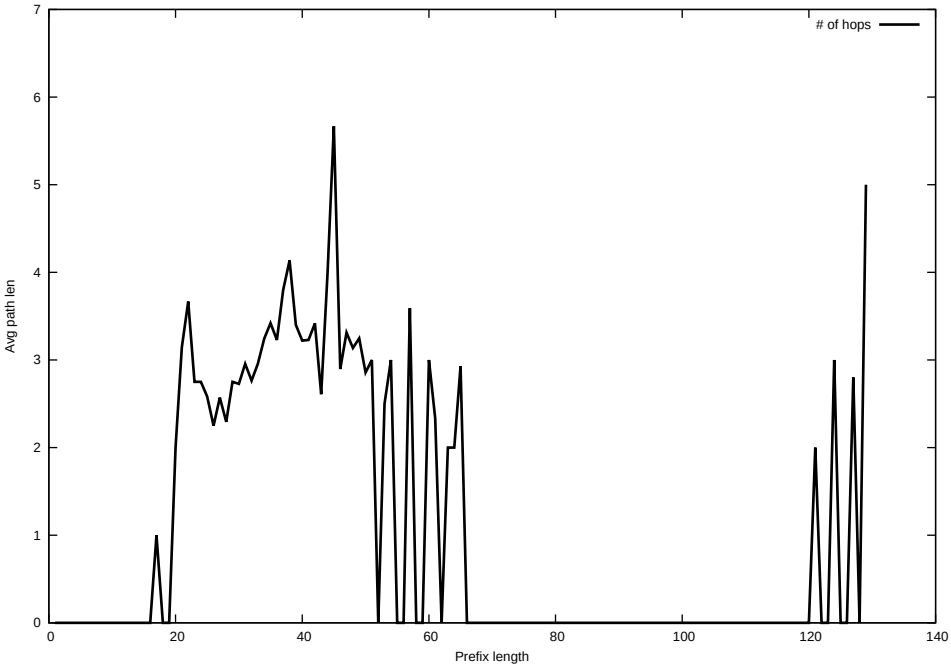
2014-04-02



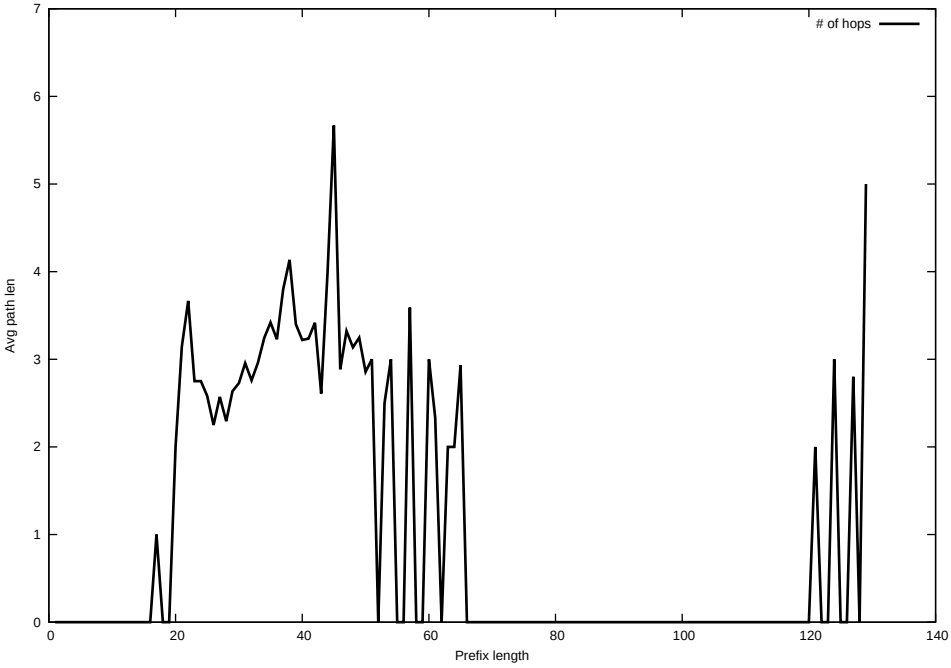
2014-04-03



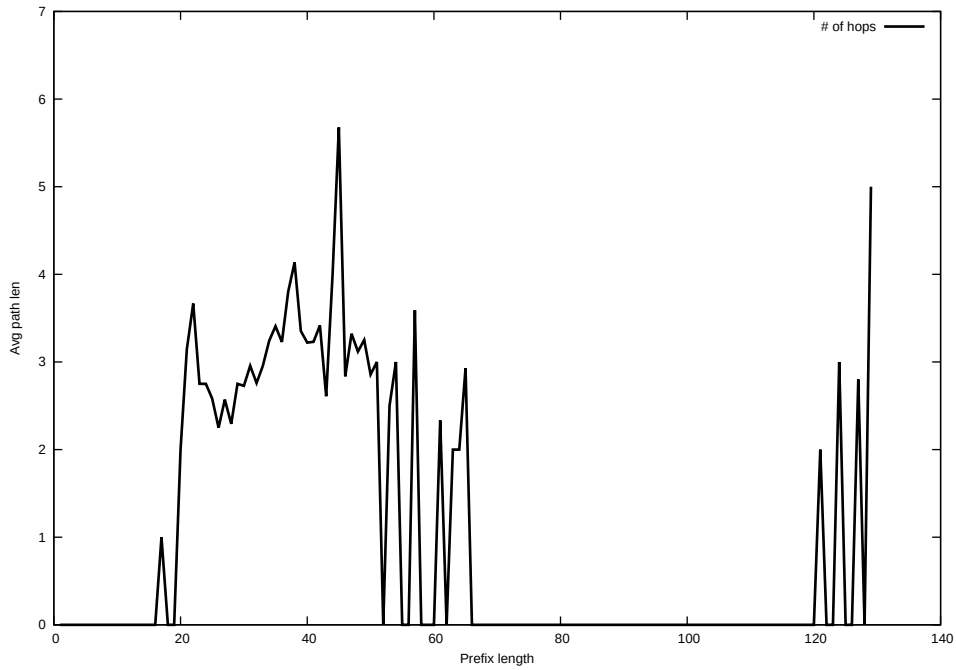
2014-04-04



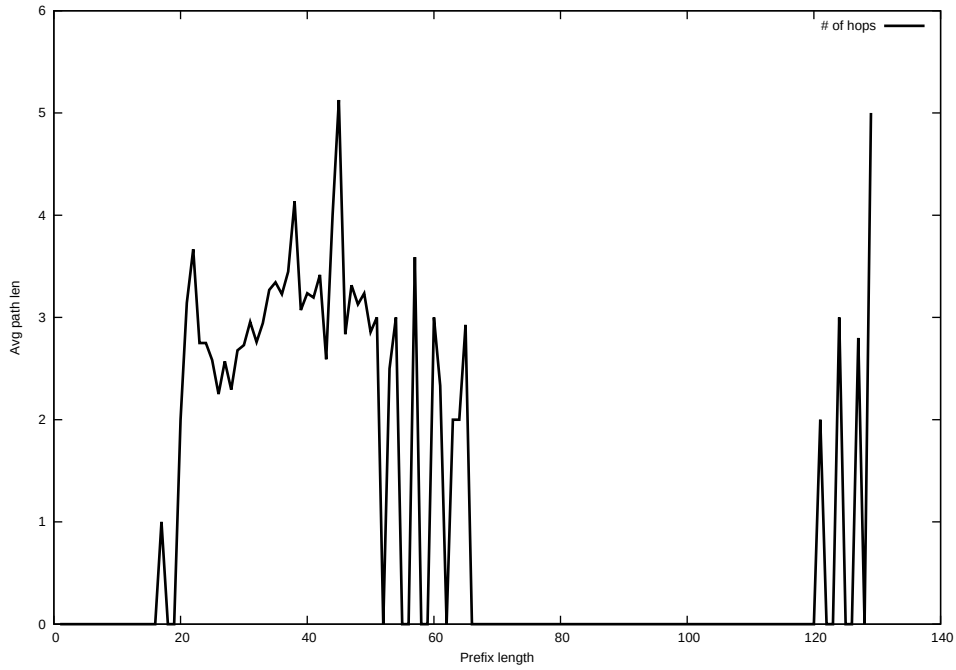
2014-04-05



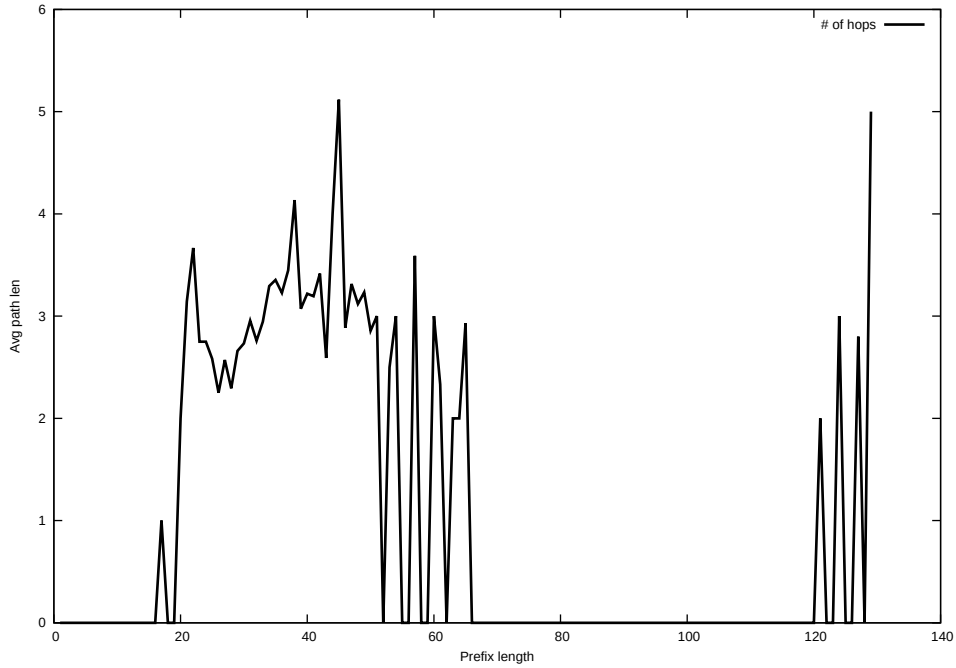
2014-04-06



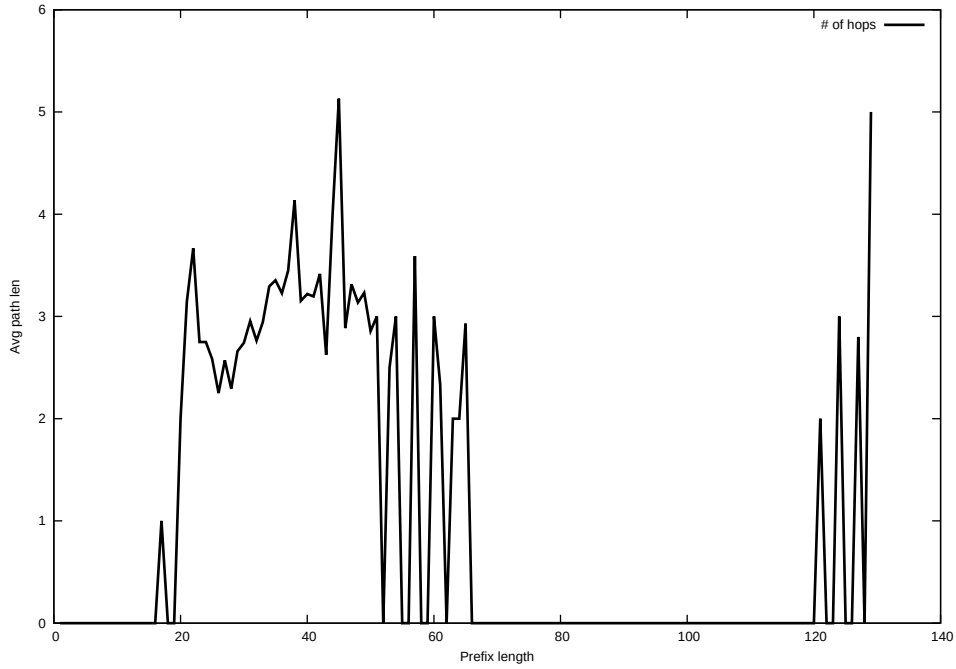
2014-04-07



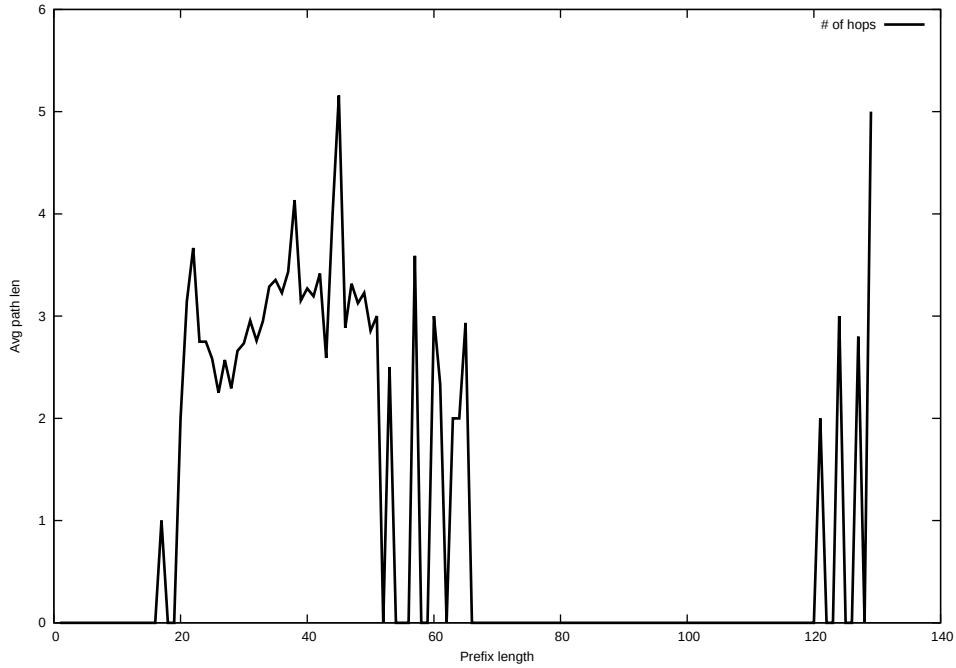
2014-04-08



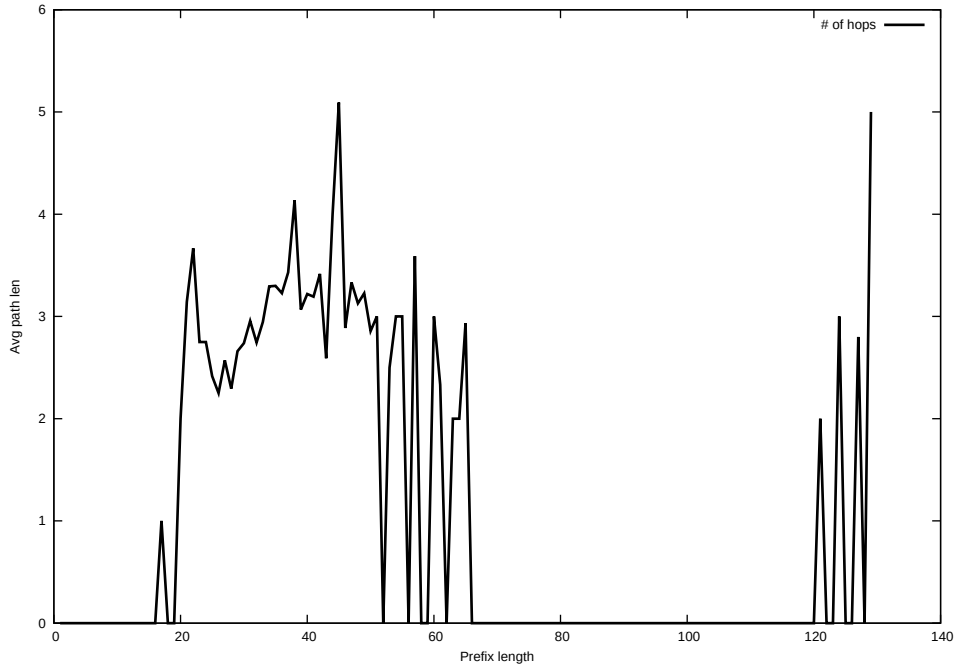
2014-04-09



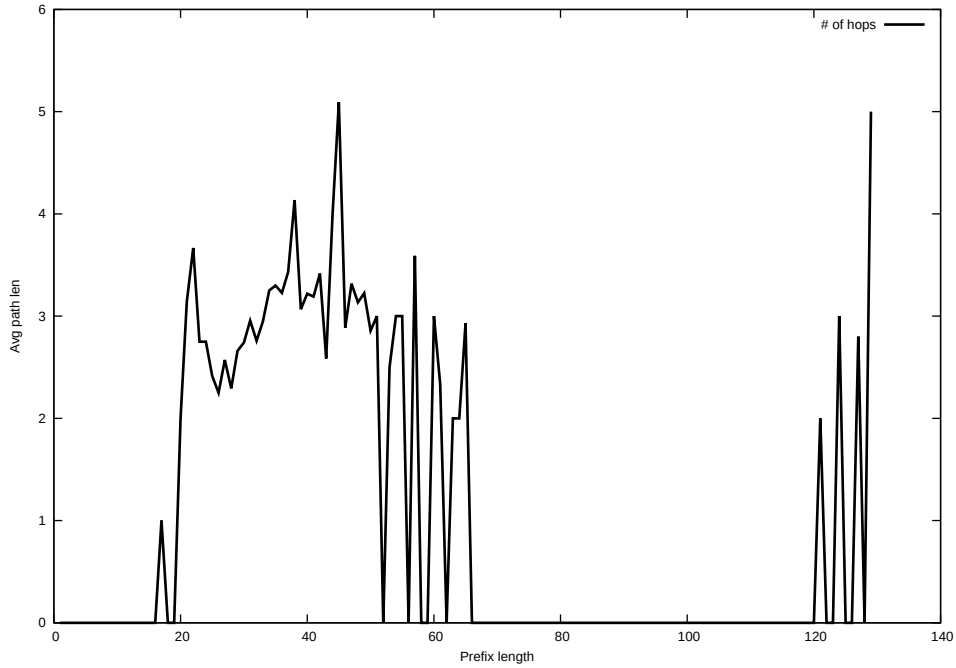
2014-04-10



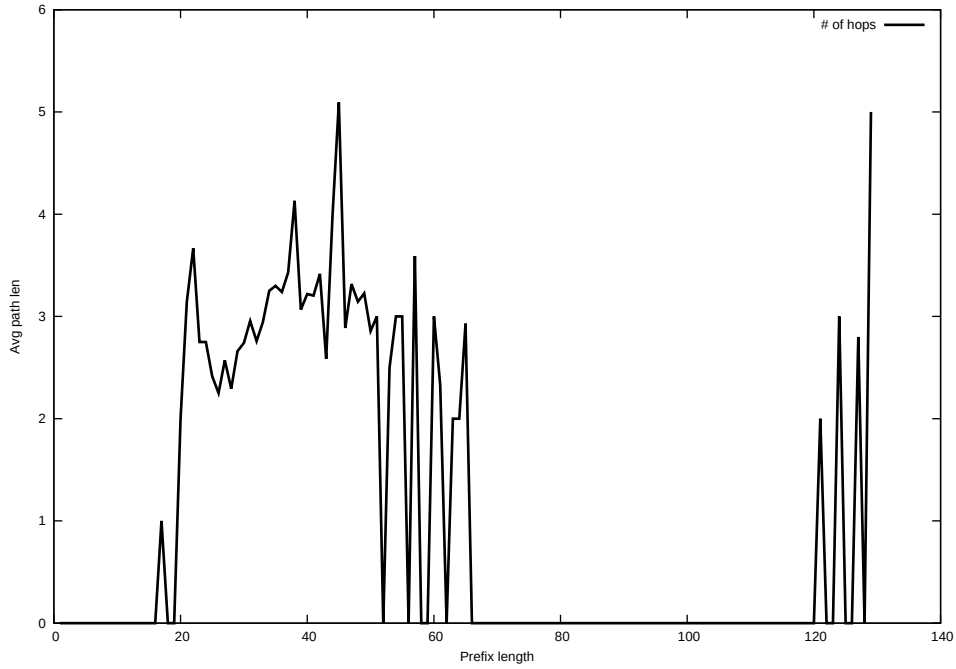
2014-04-11



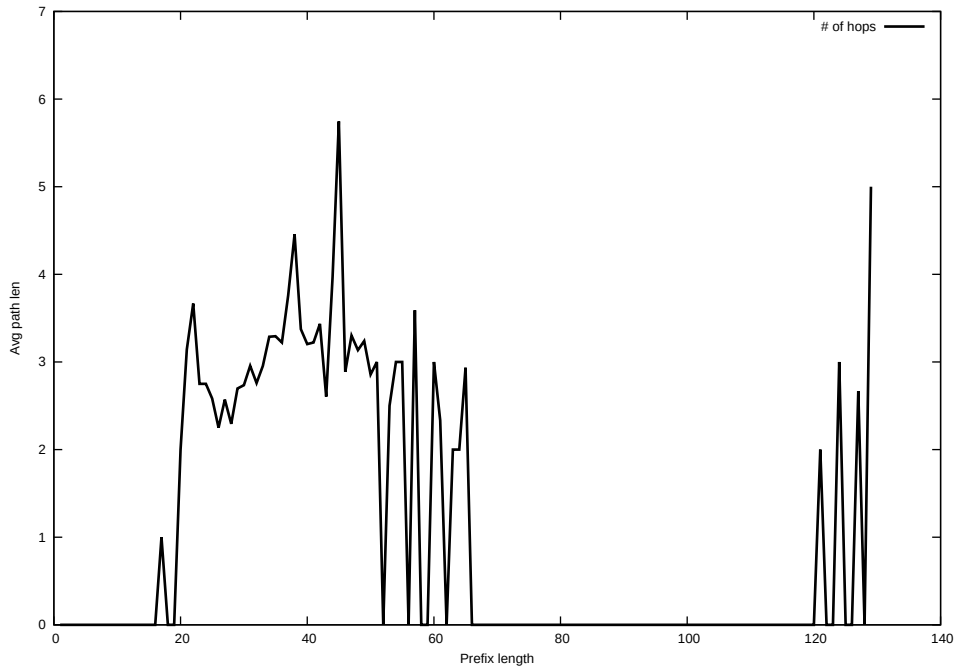
2014-04-12



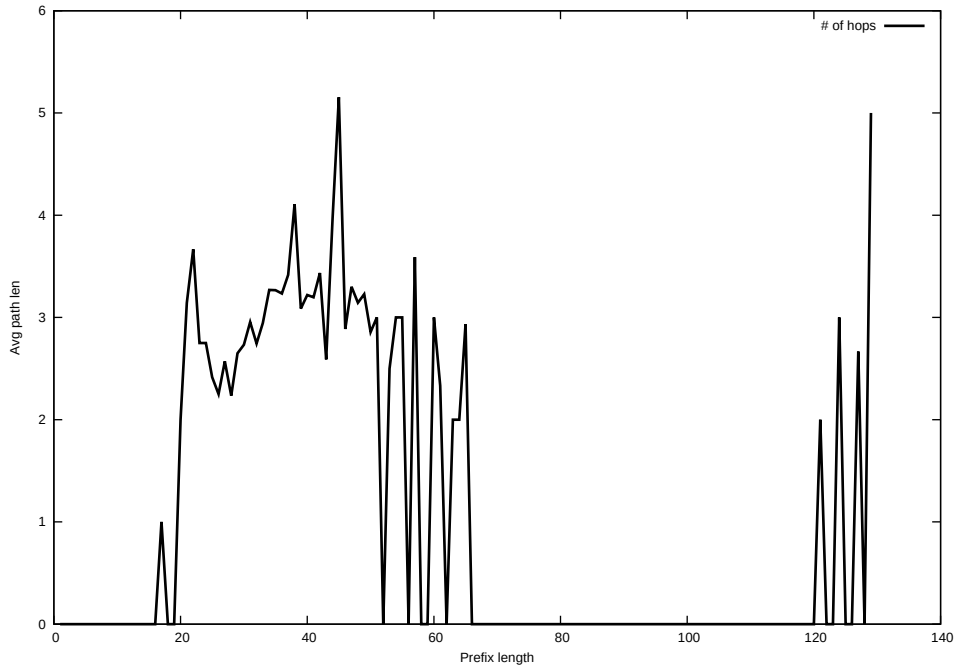
2014-04-13



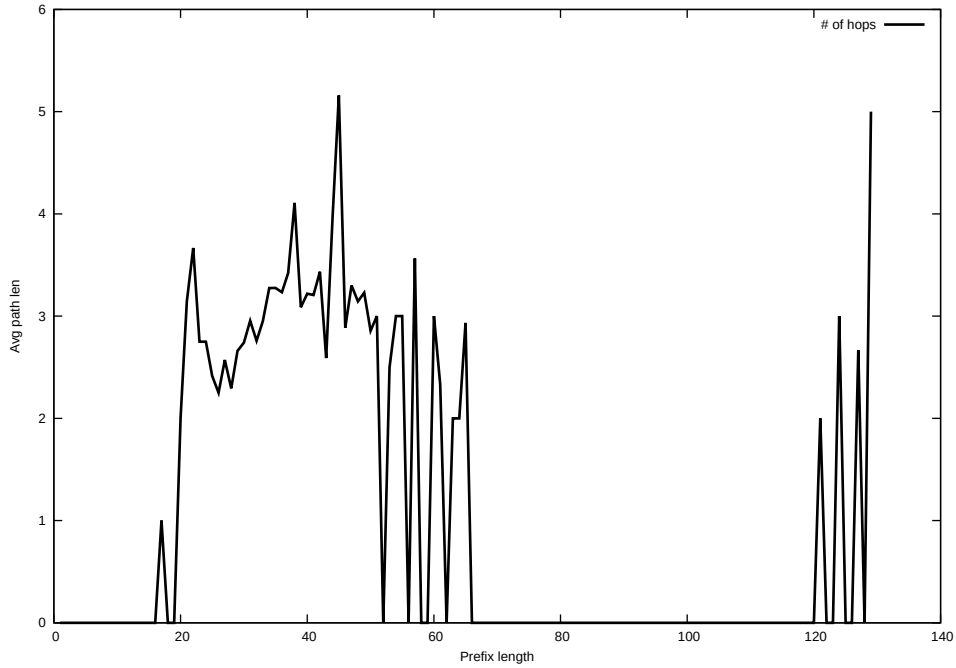
2014-04-14



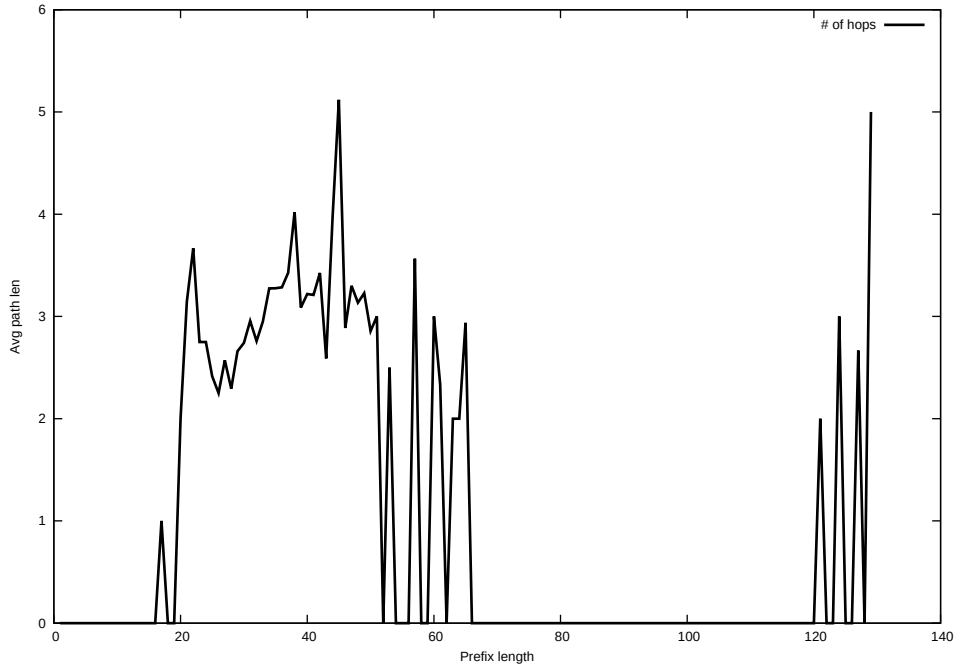
2014-04-15



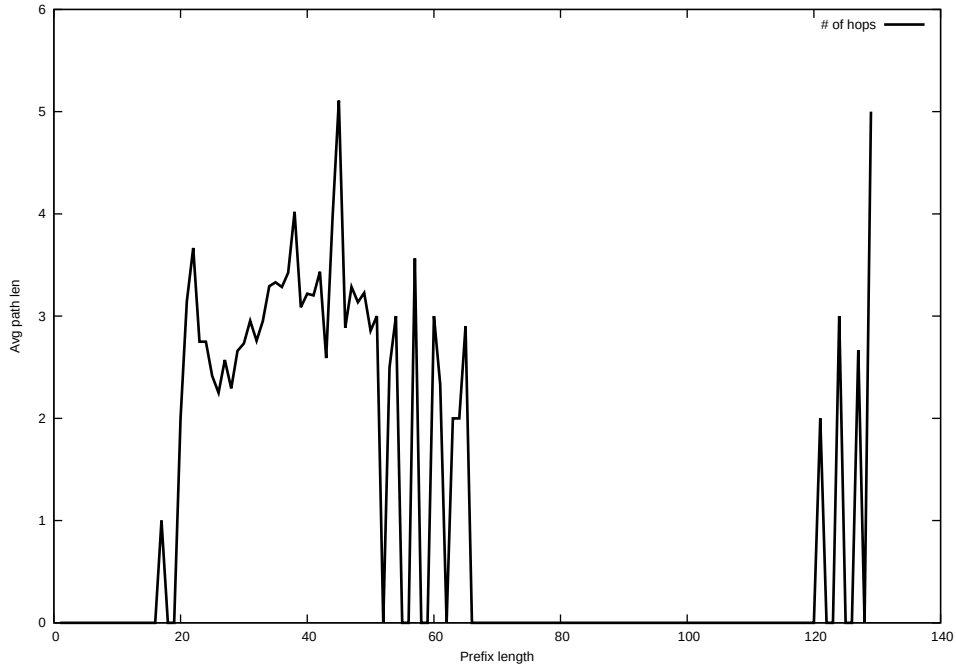
2014-04-16



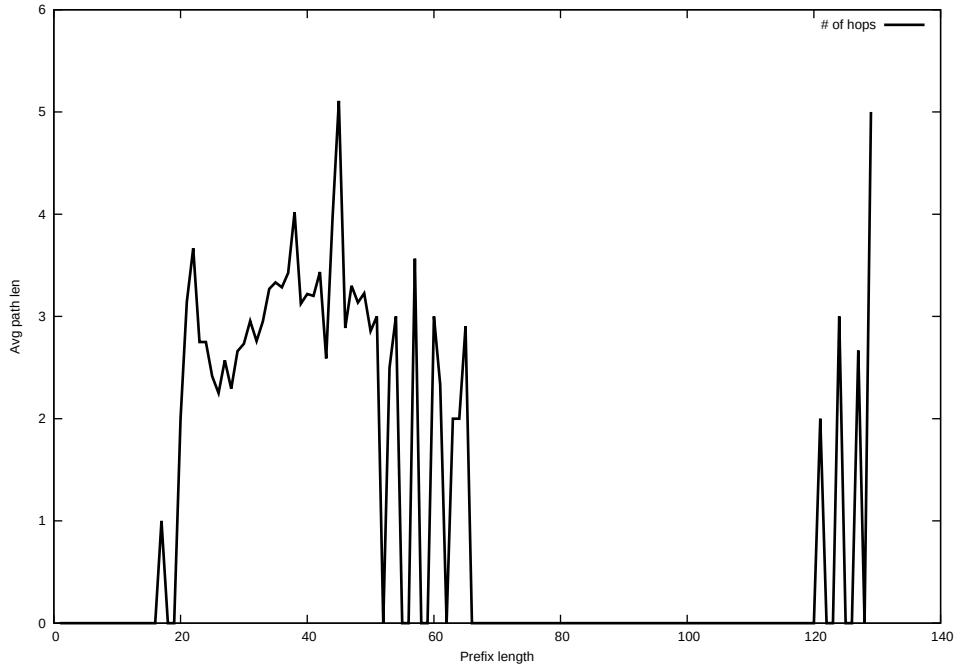
2014-04-17



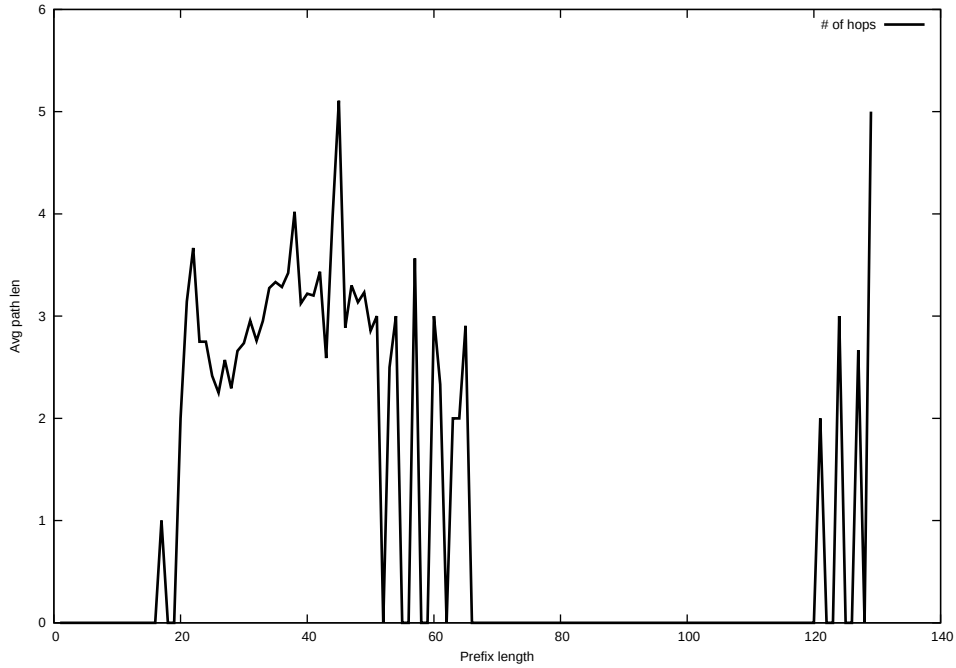
2014-04-18



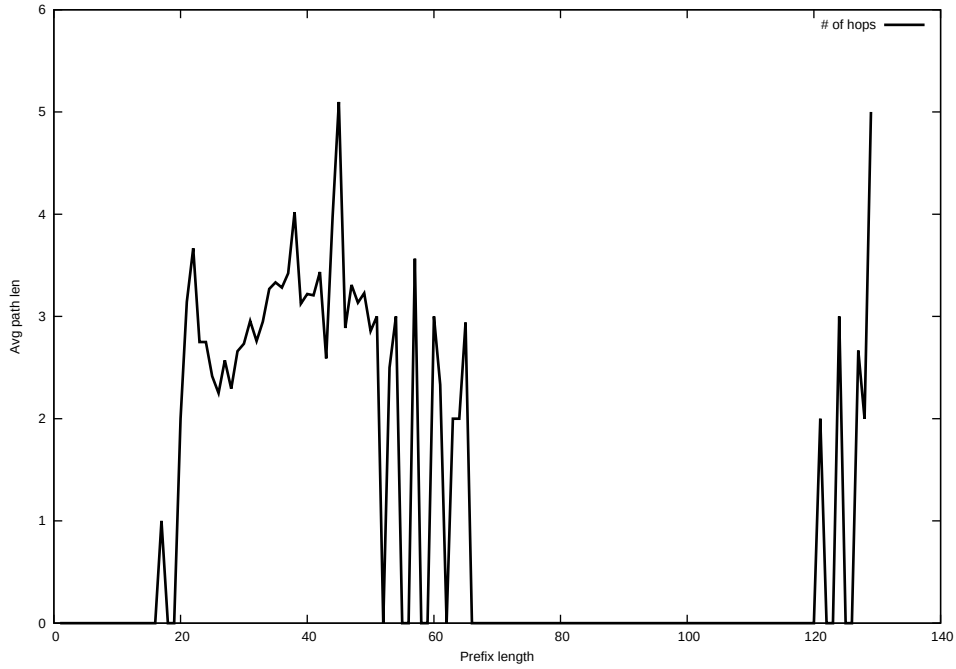
2014-04-19



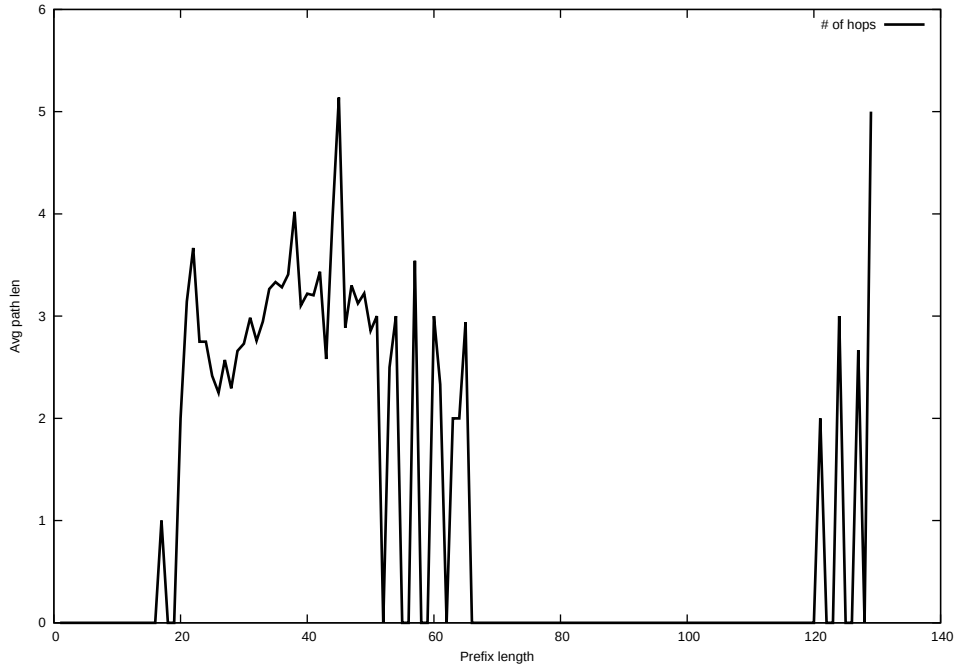
2014-04-20



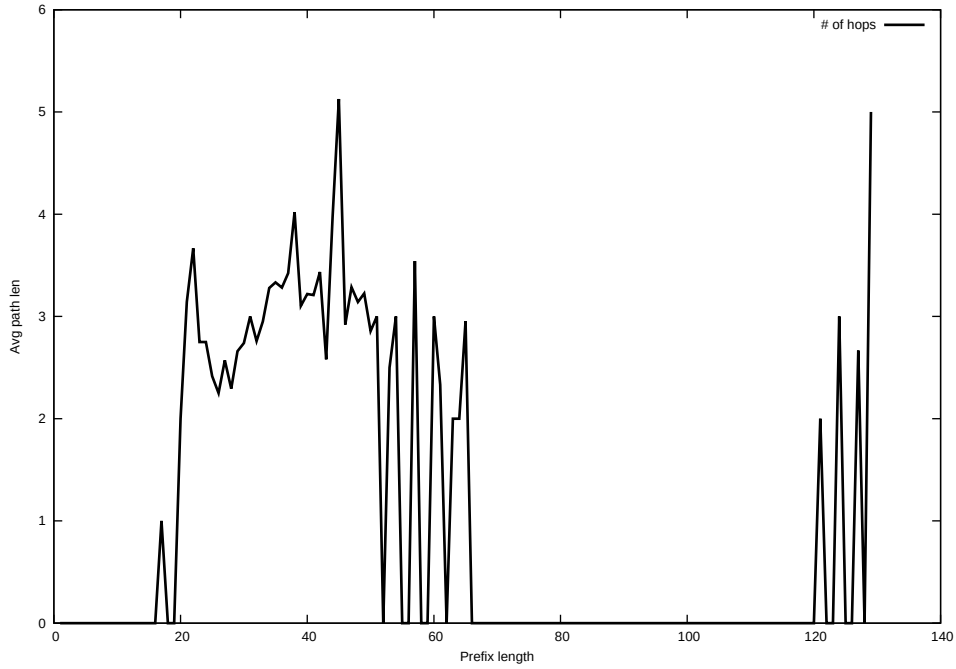
2014-04-21



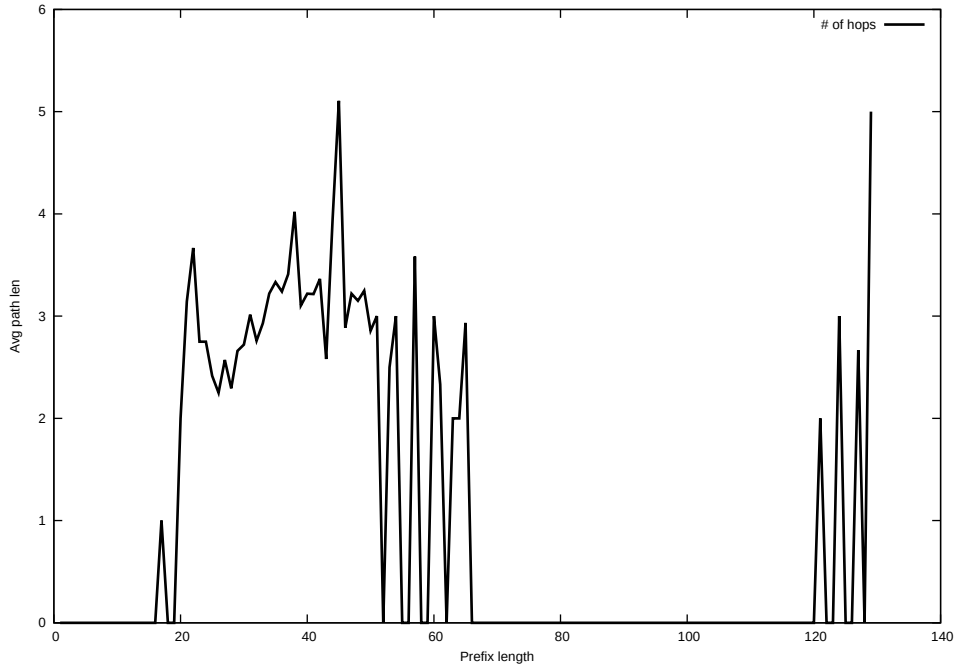
2014-04-22



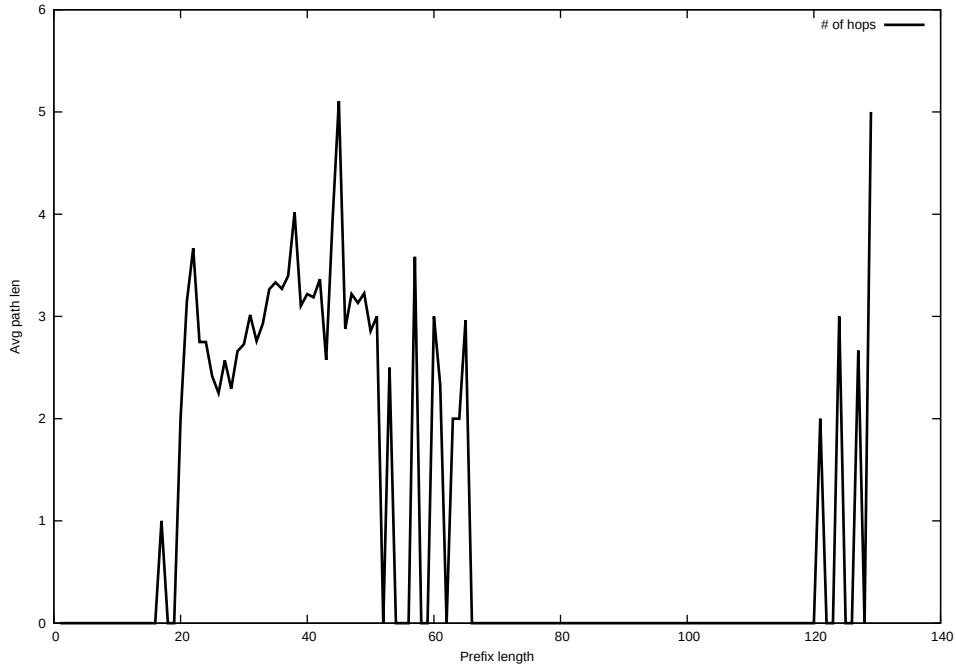
2014-04-23



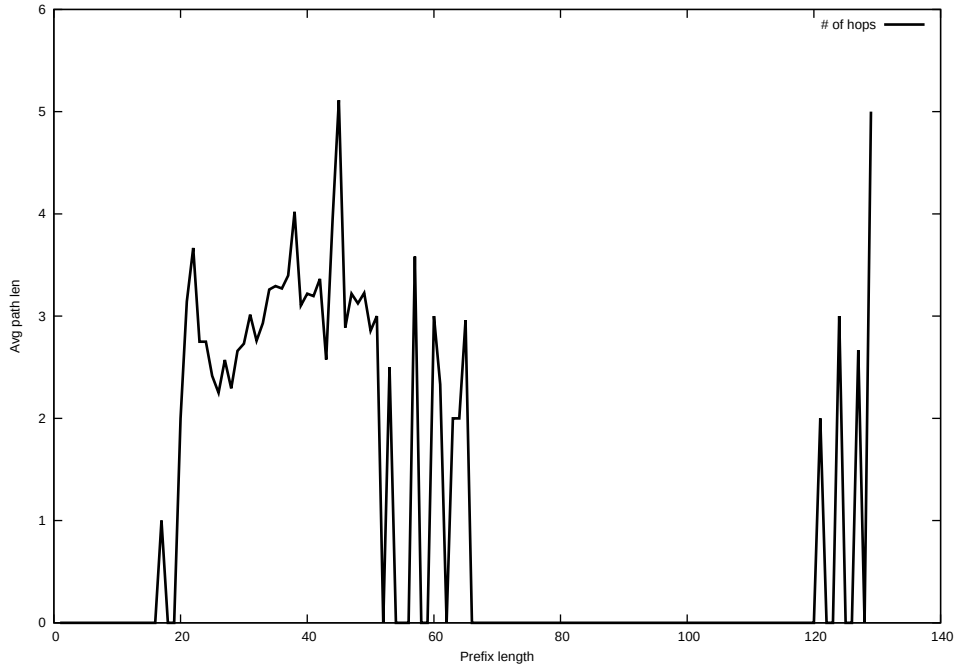
2014-04-24



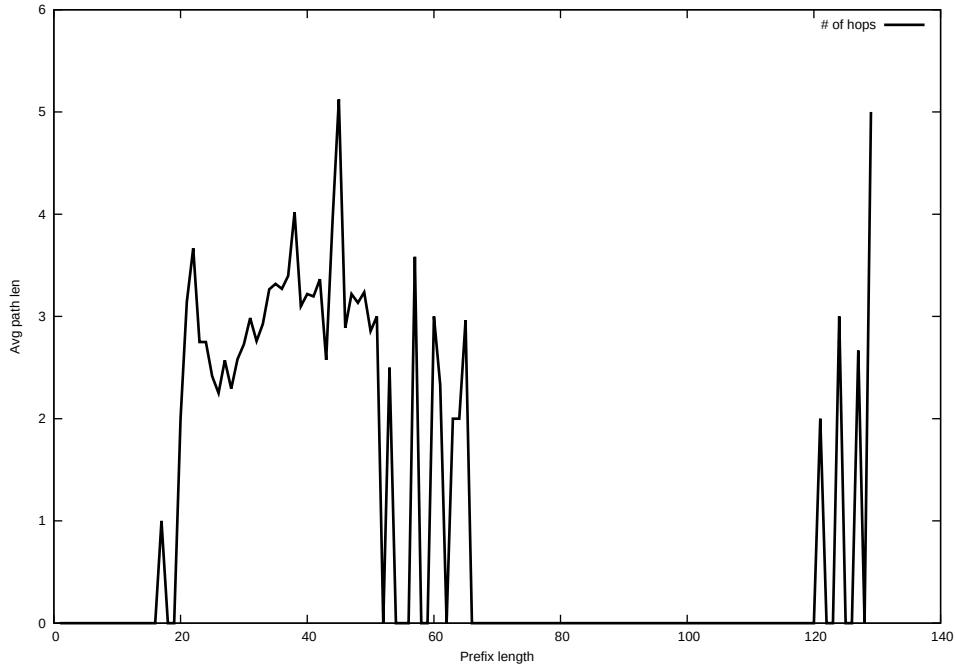
2014-04-25



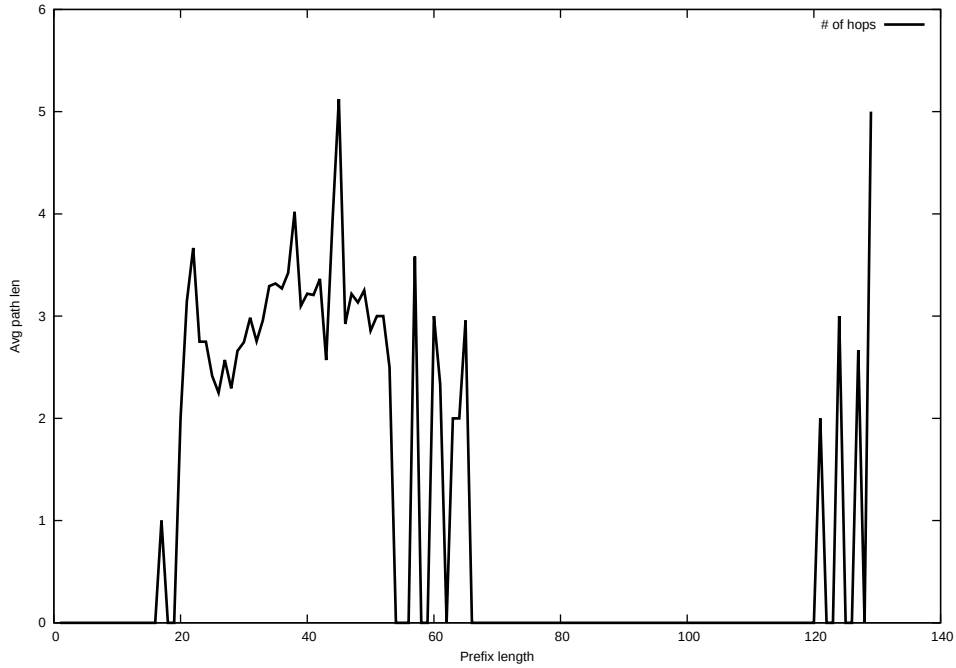
2014-04-26



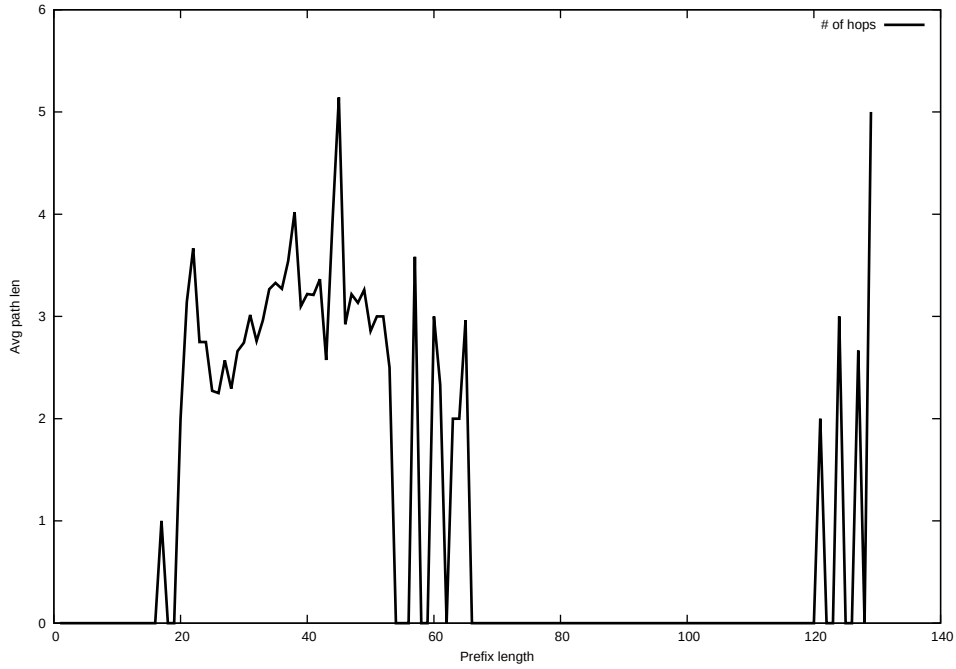
2014-04-27



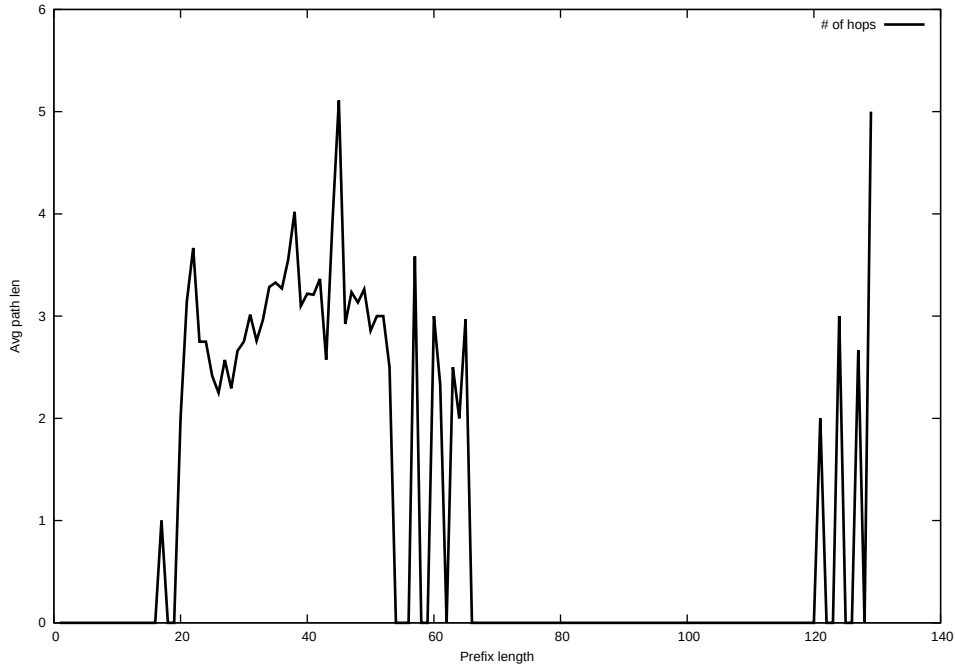
2014-04-28



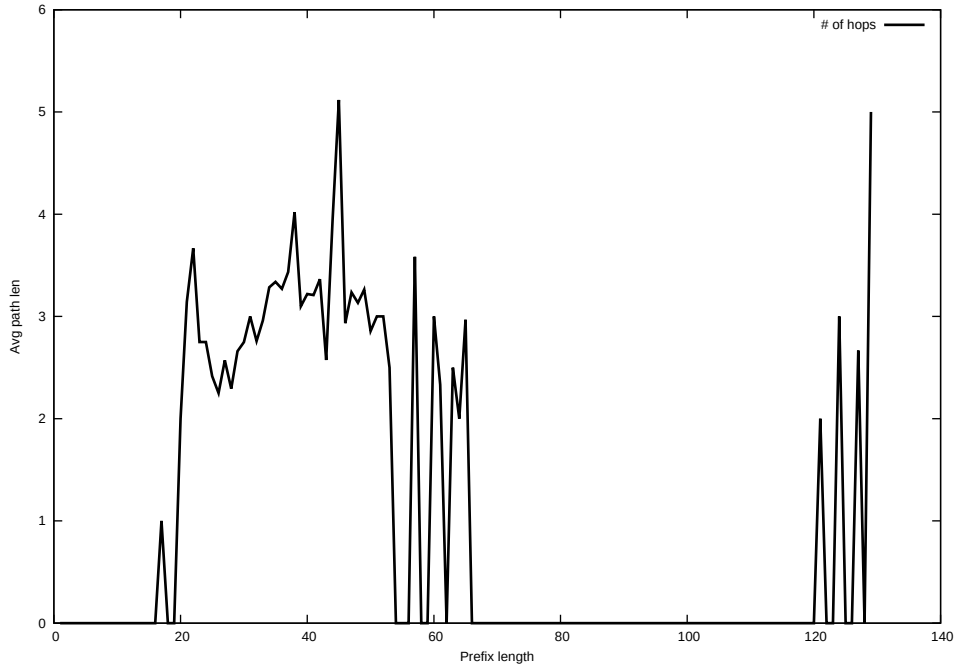
2014-04-29



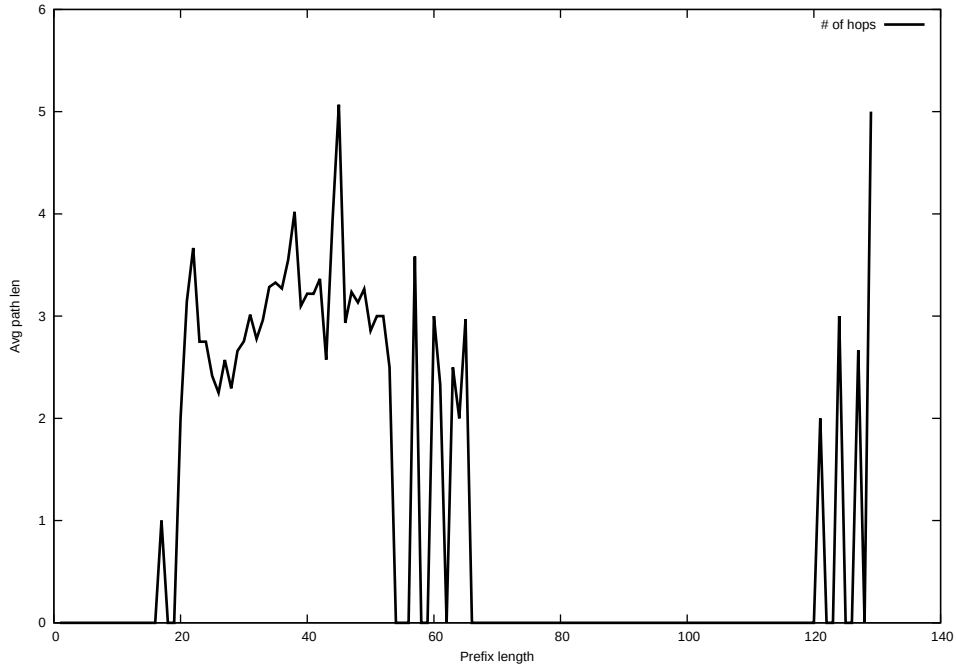
2014-04-30



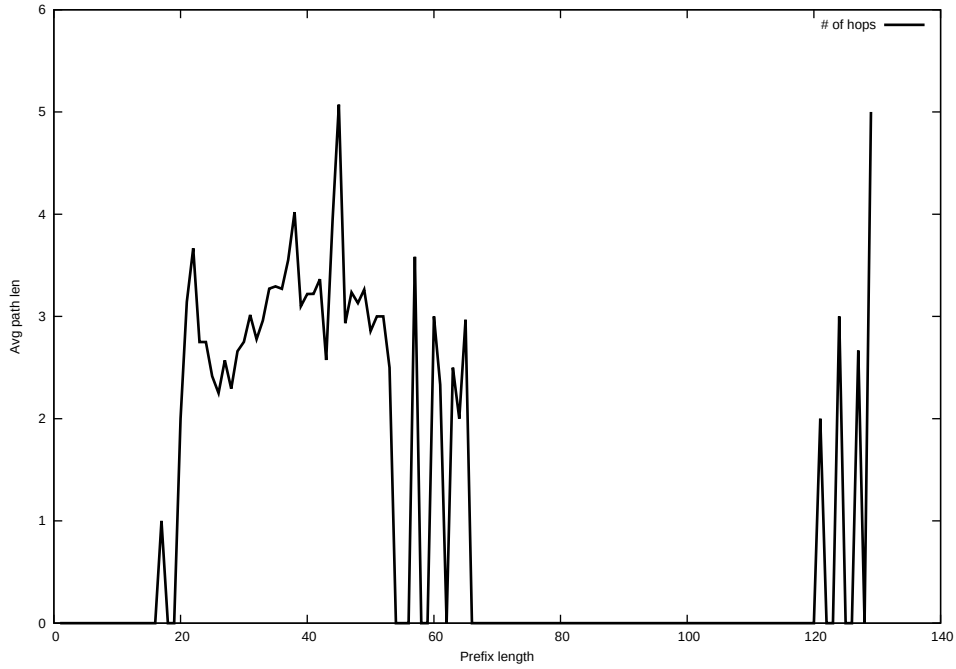
2014-05-01



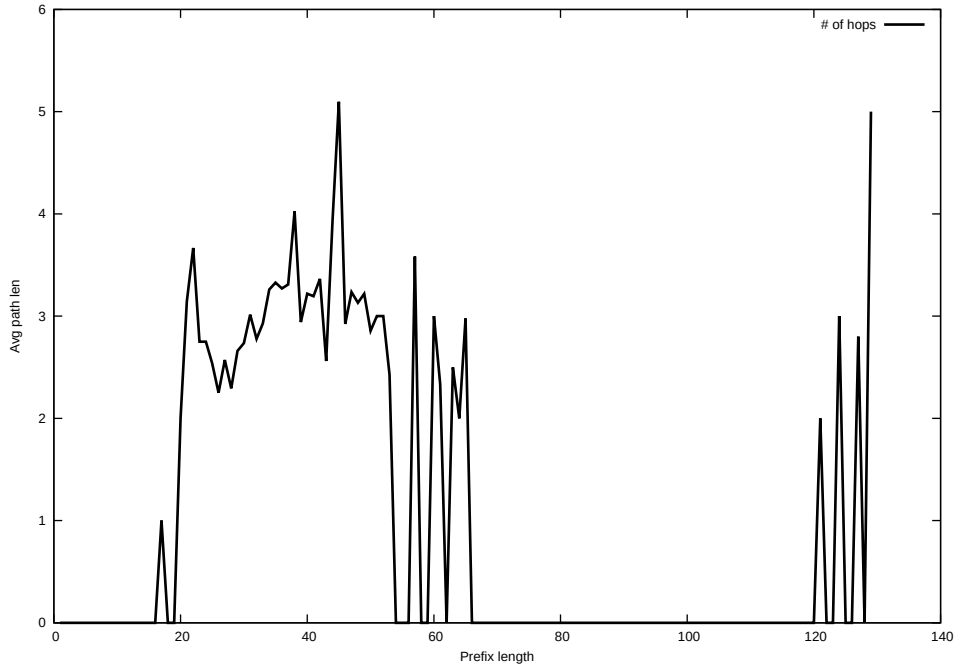
2014-05-02



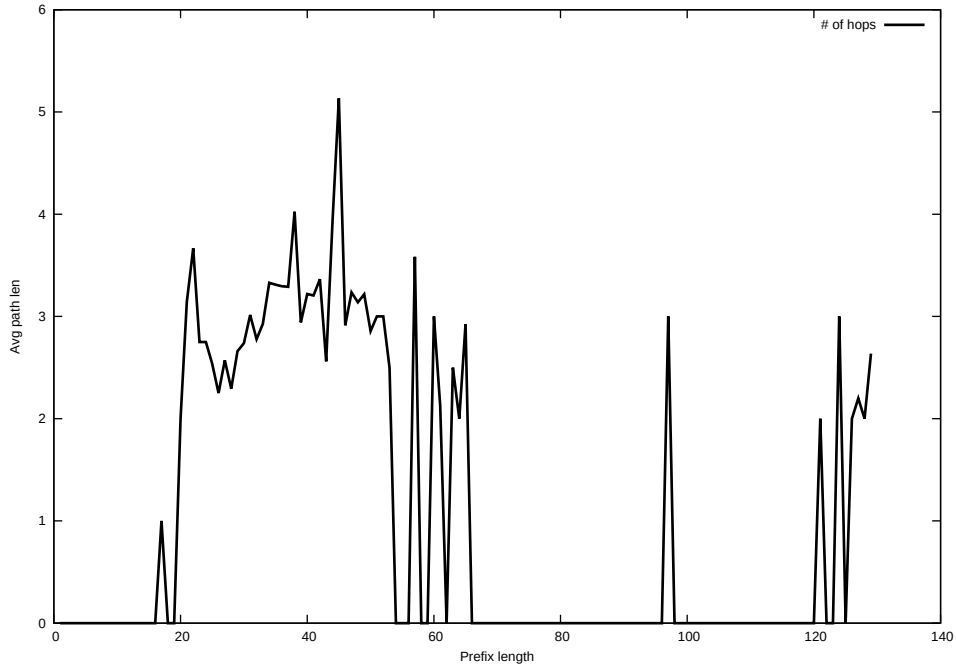
2014-05-03



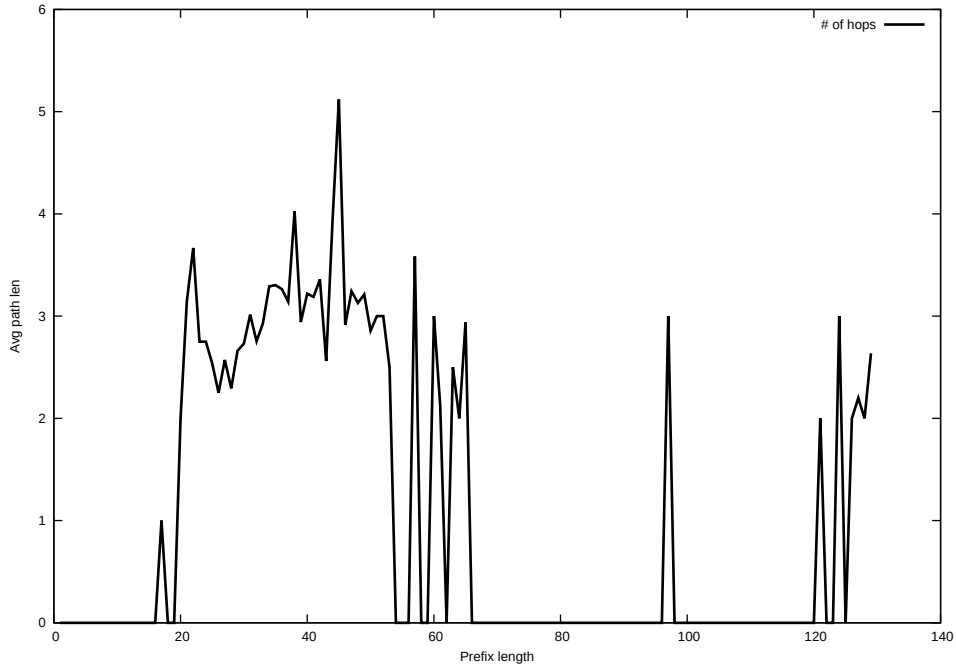
2014-05-04



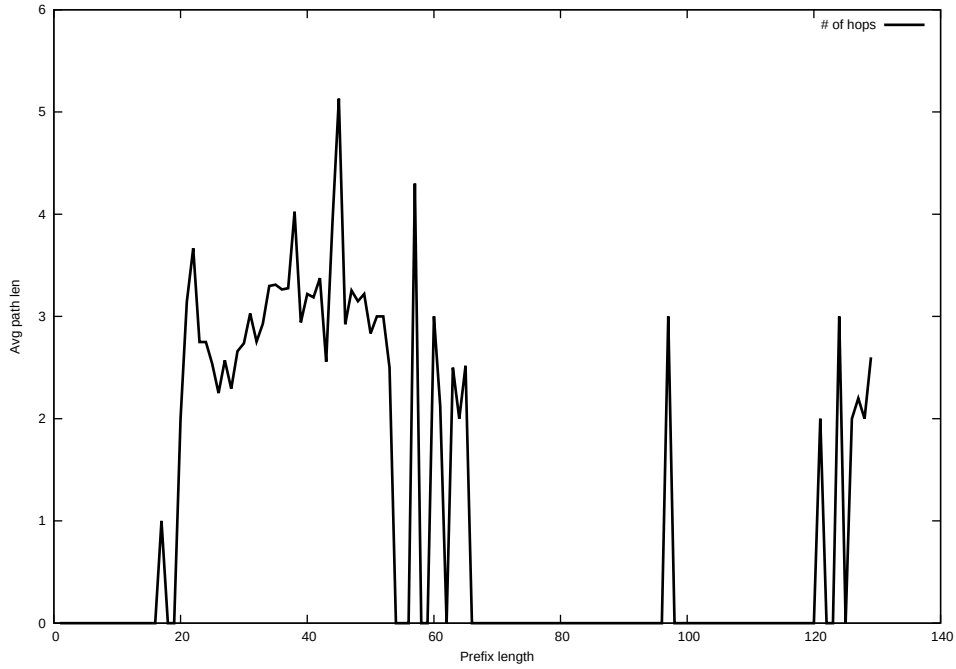
2014-05-05



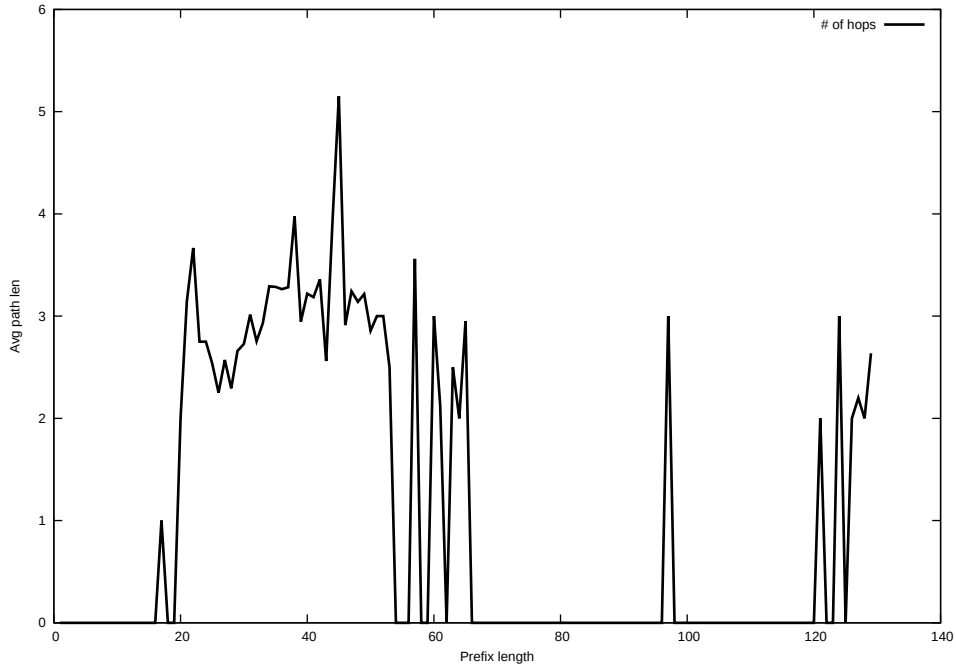
2014-05-06



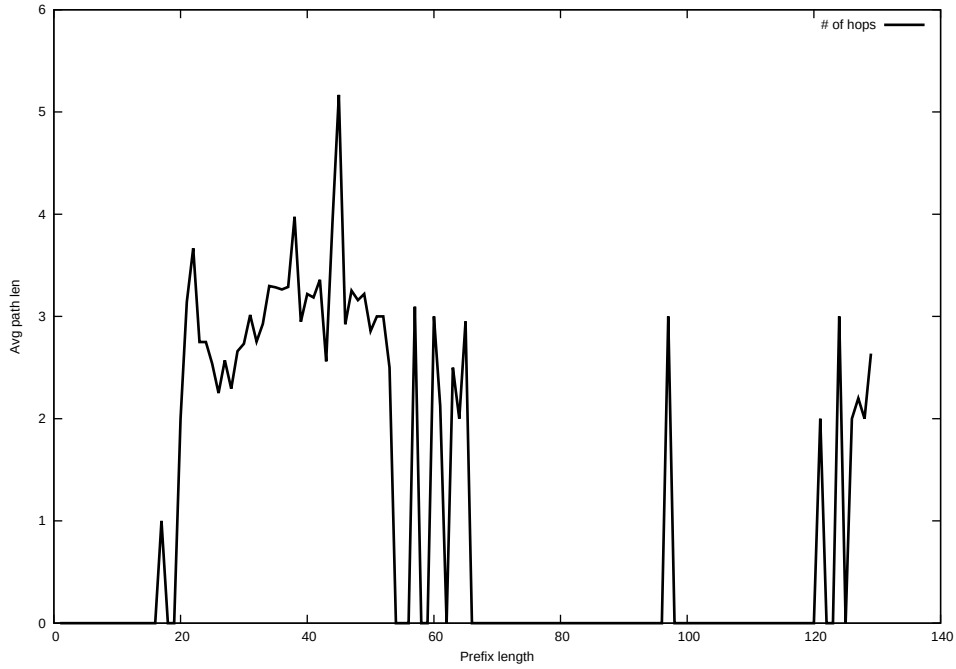
2014-05-07



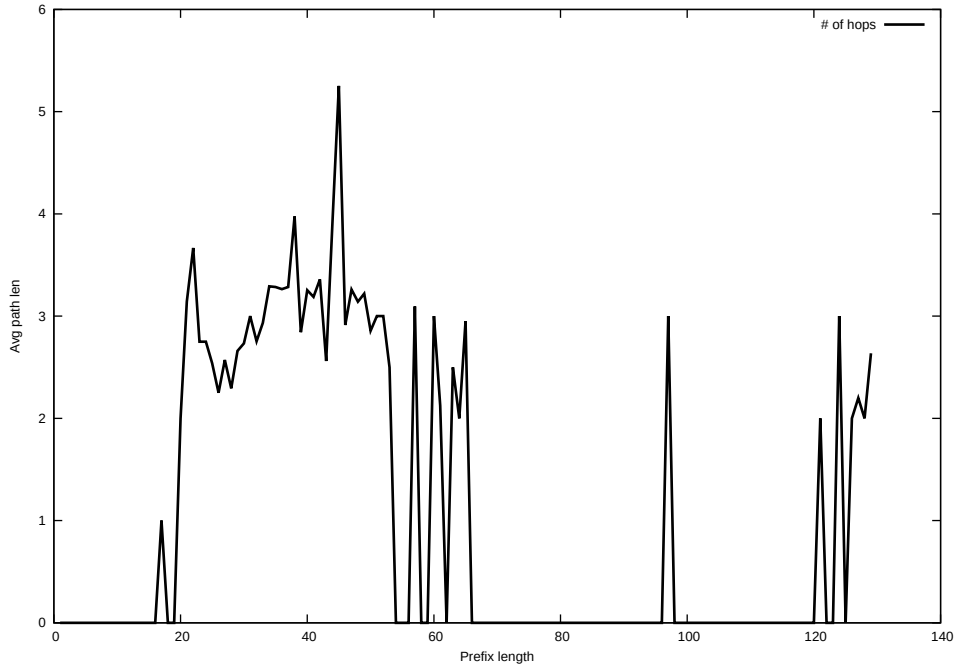
2014-05-08



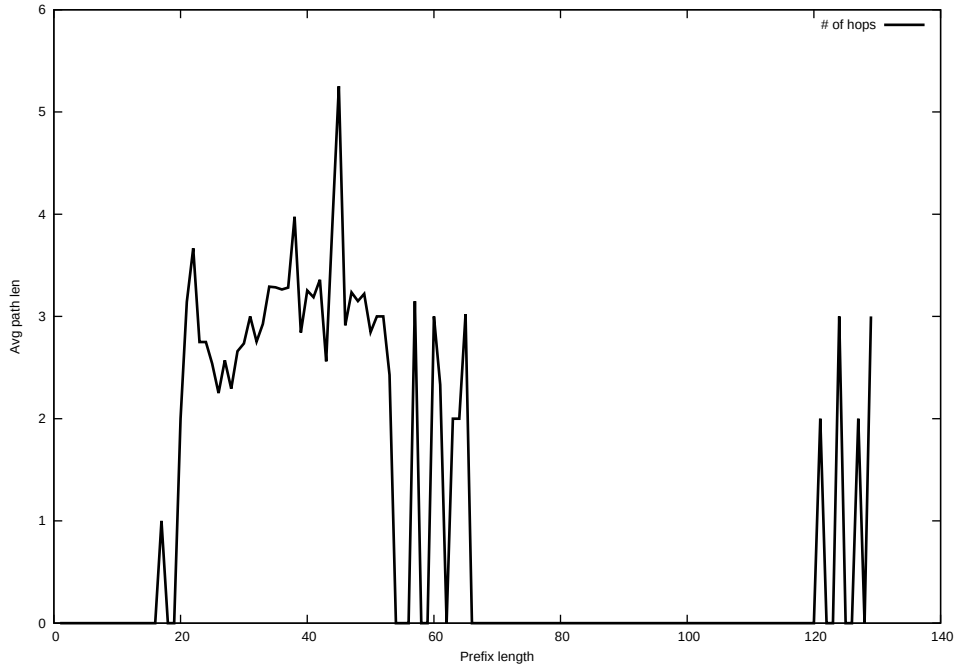
2014-05-09



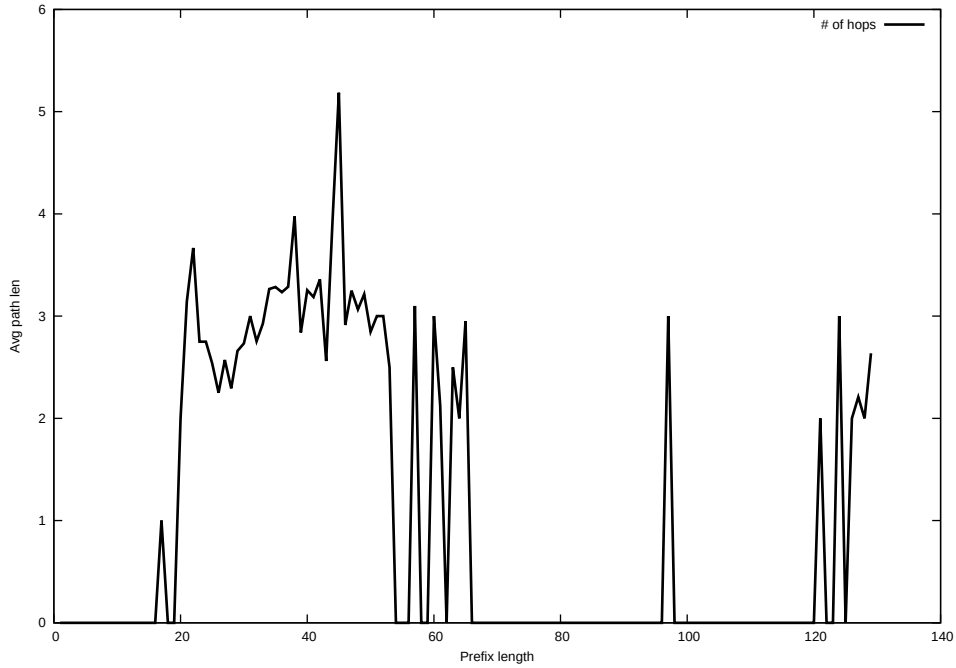
2014-05-10



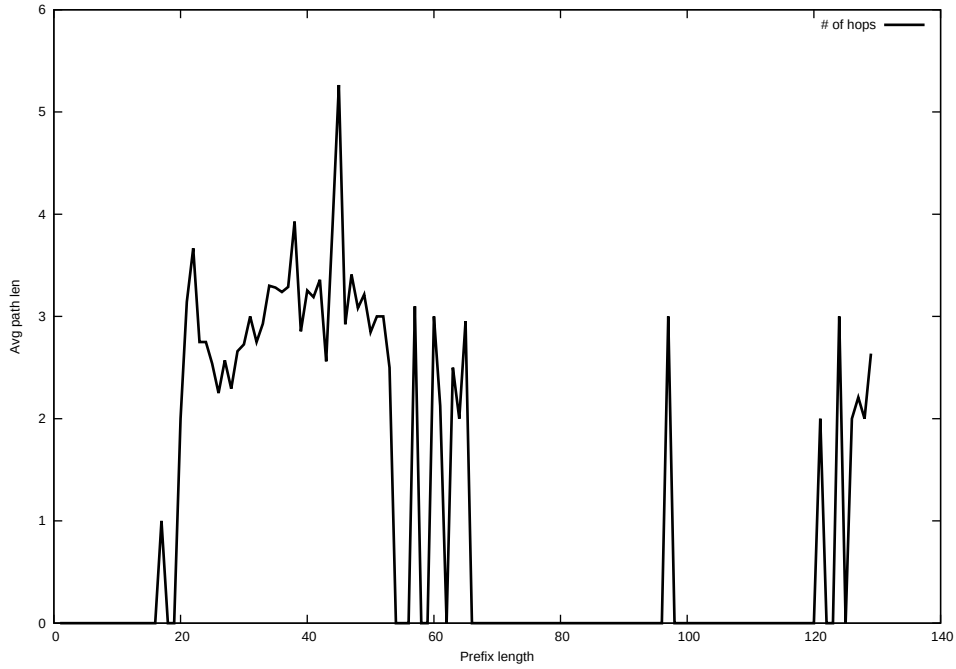
2014-05-11



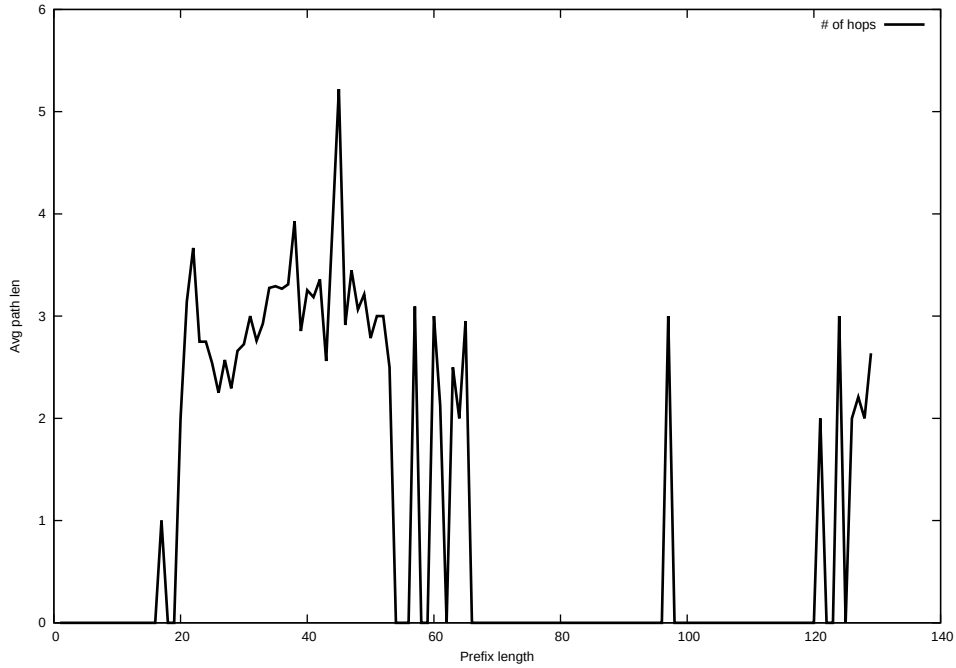
2014-05-12



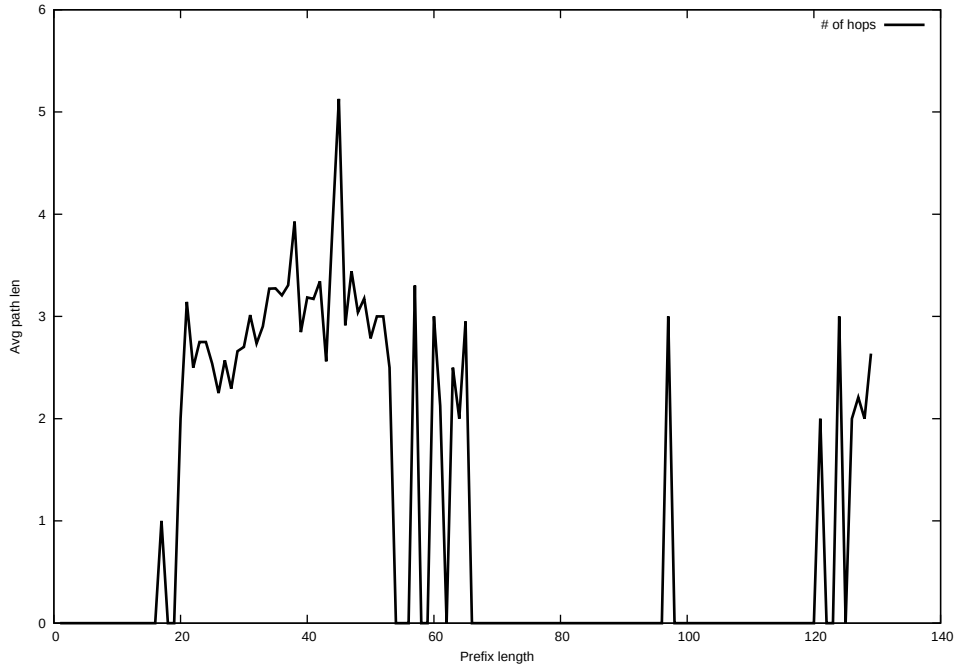
2014-05-13



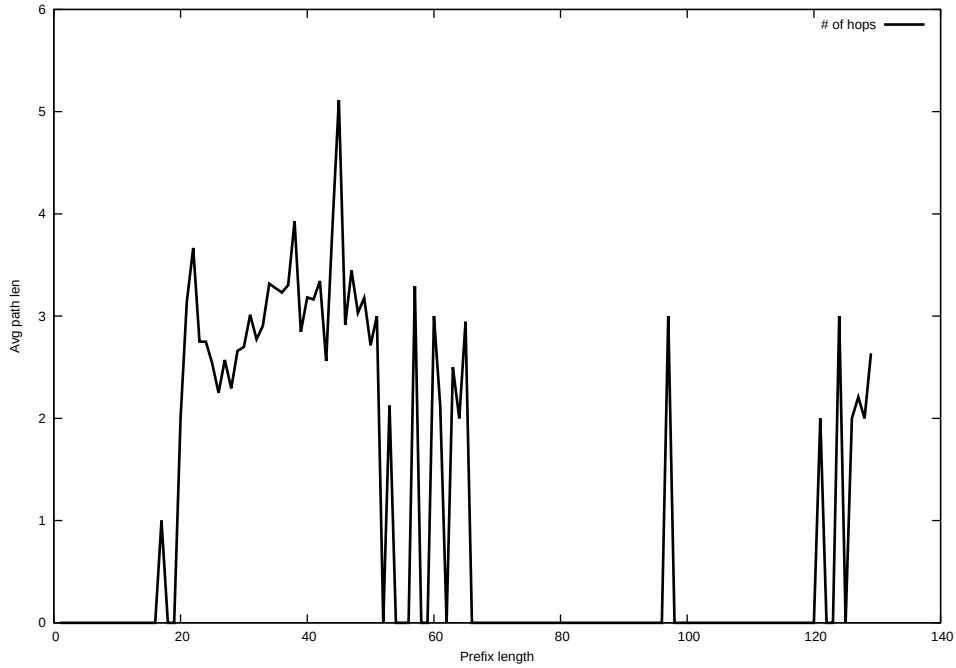
2014-05-14



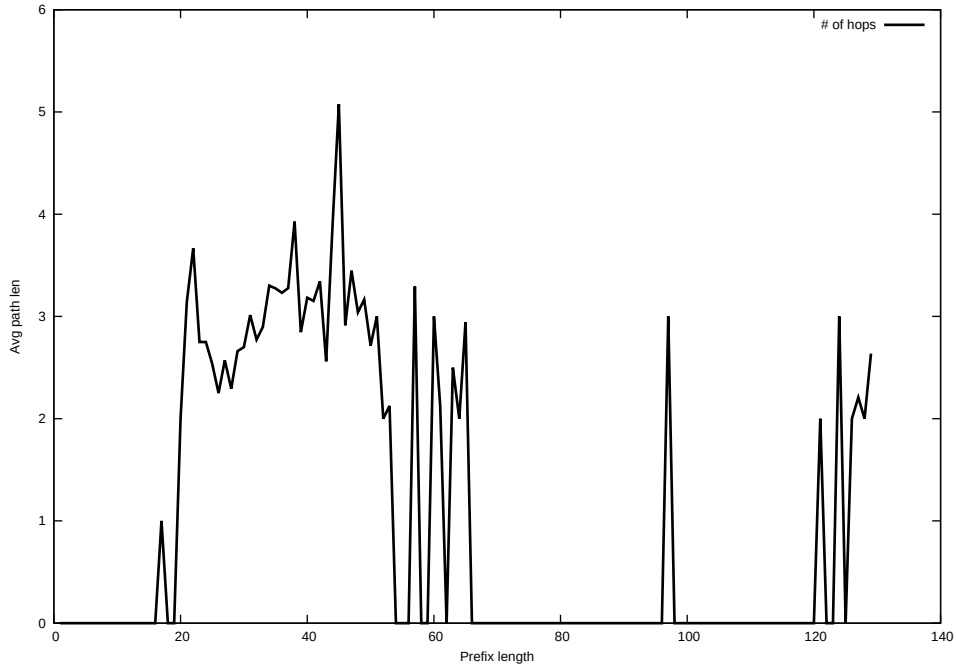
2014-05-15



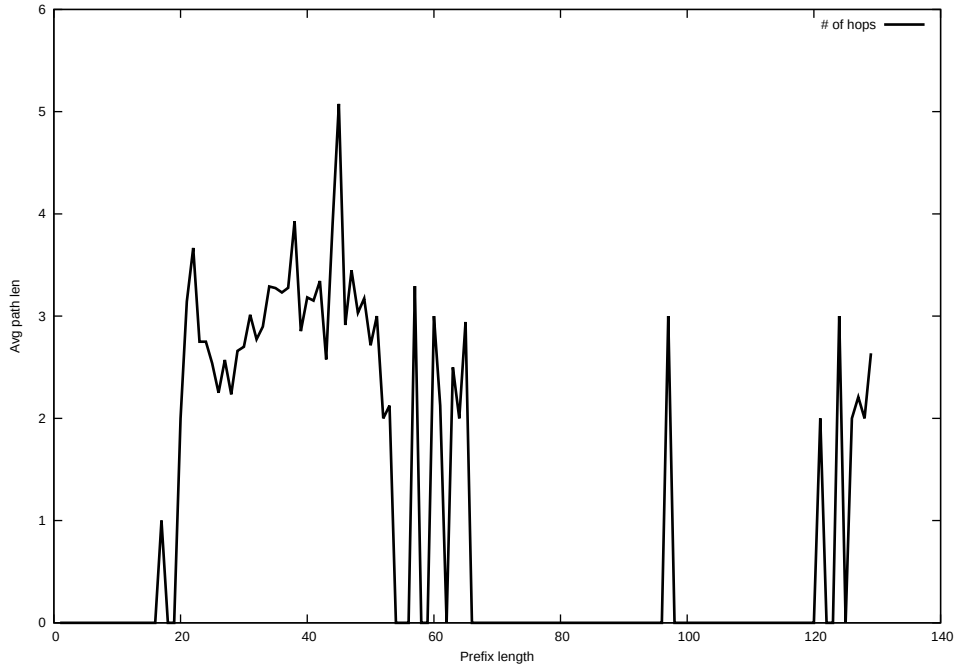
2014-05-16



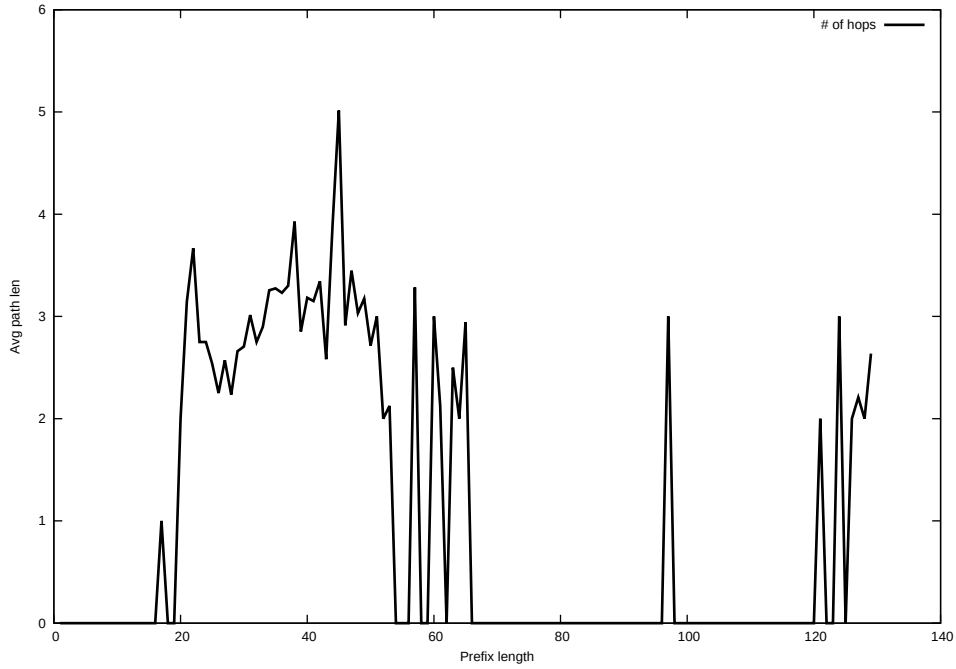
2014-05-17



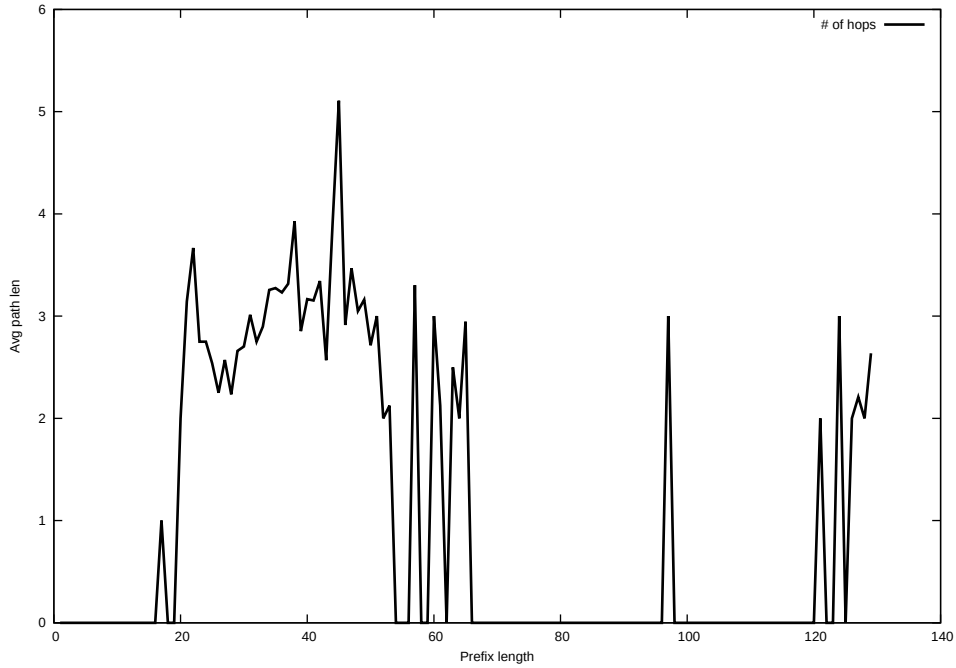
2014-05-18



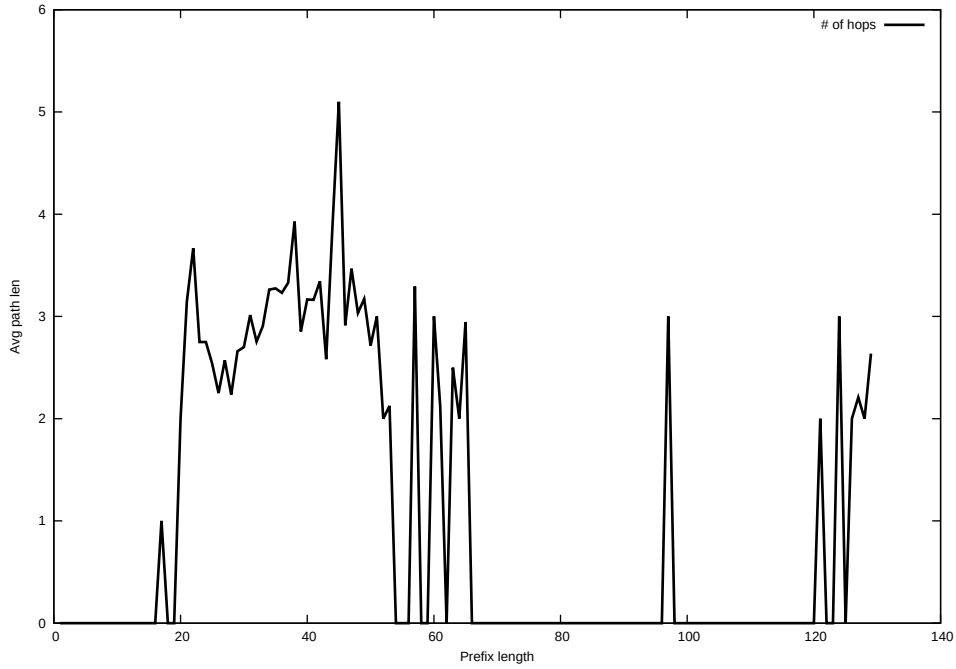
2014-05-19



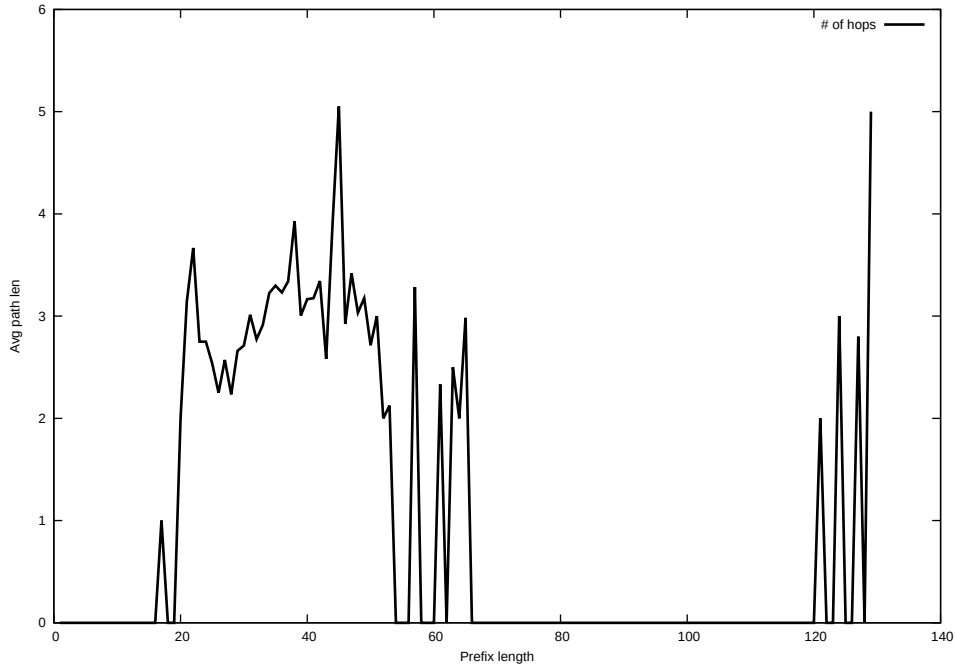
2014-05-20



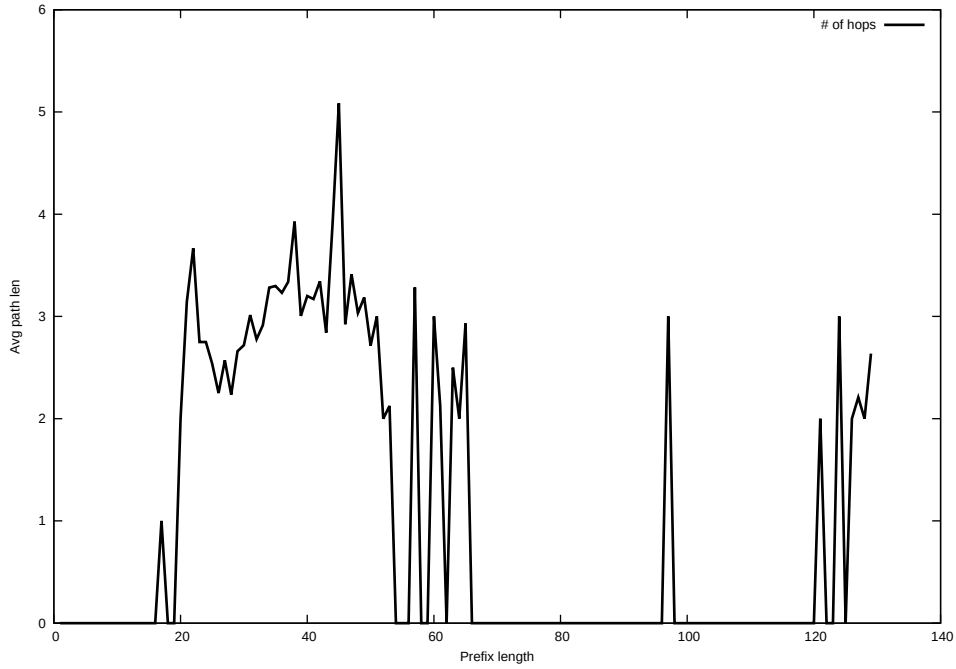
2014-05-21



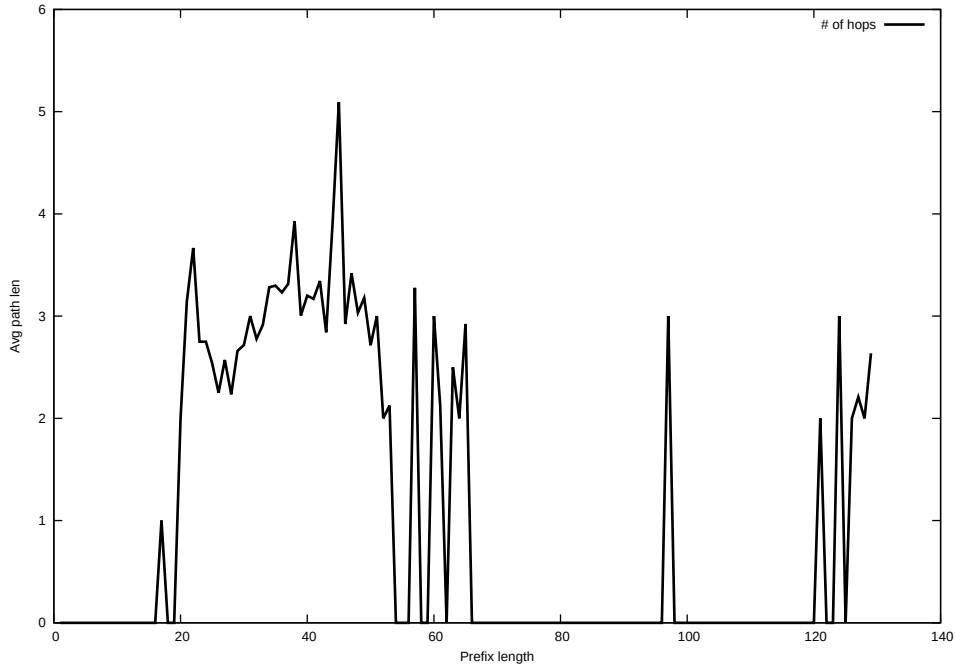
2014-05-22



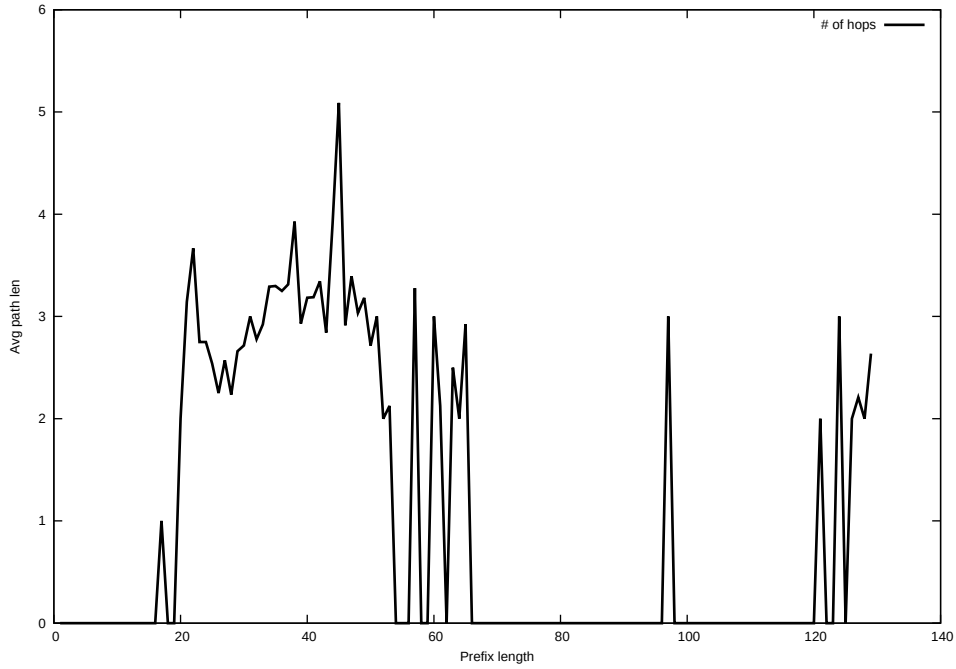
2014-05-23



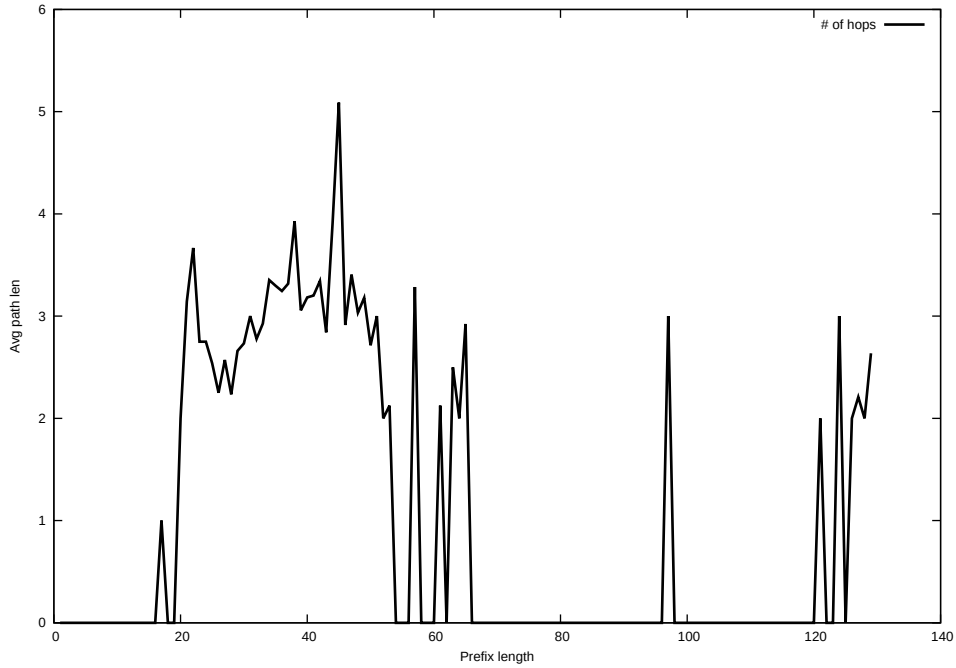
2014-05-24



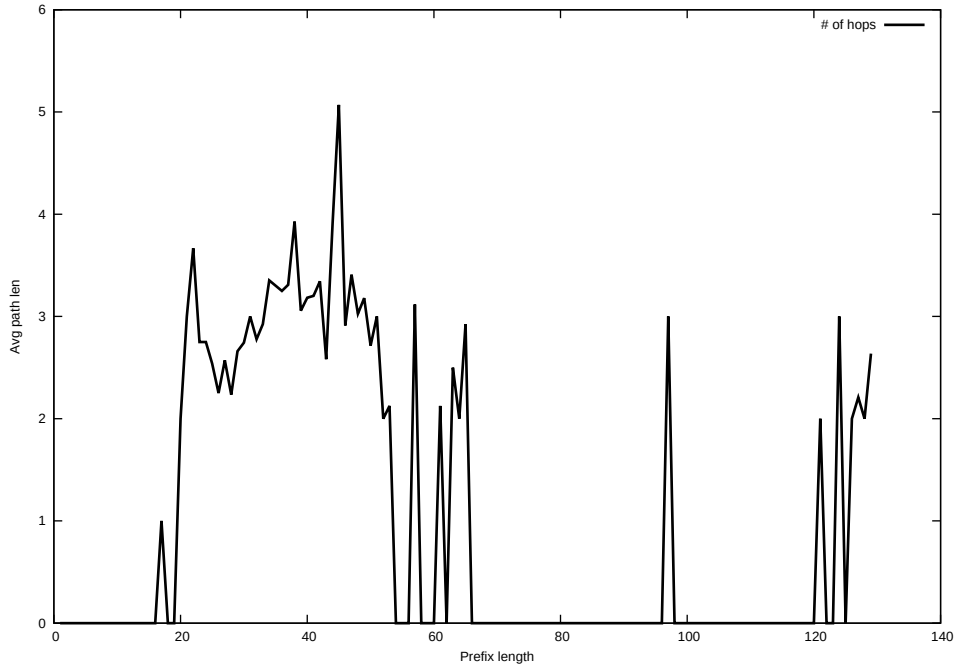
2014-05-25



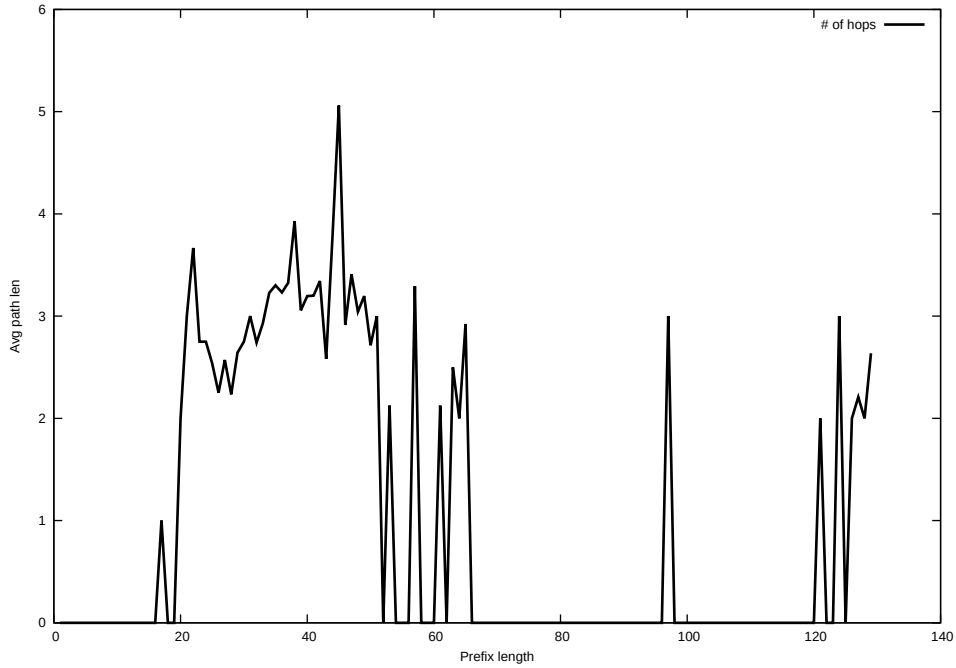
2014-05-26



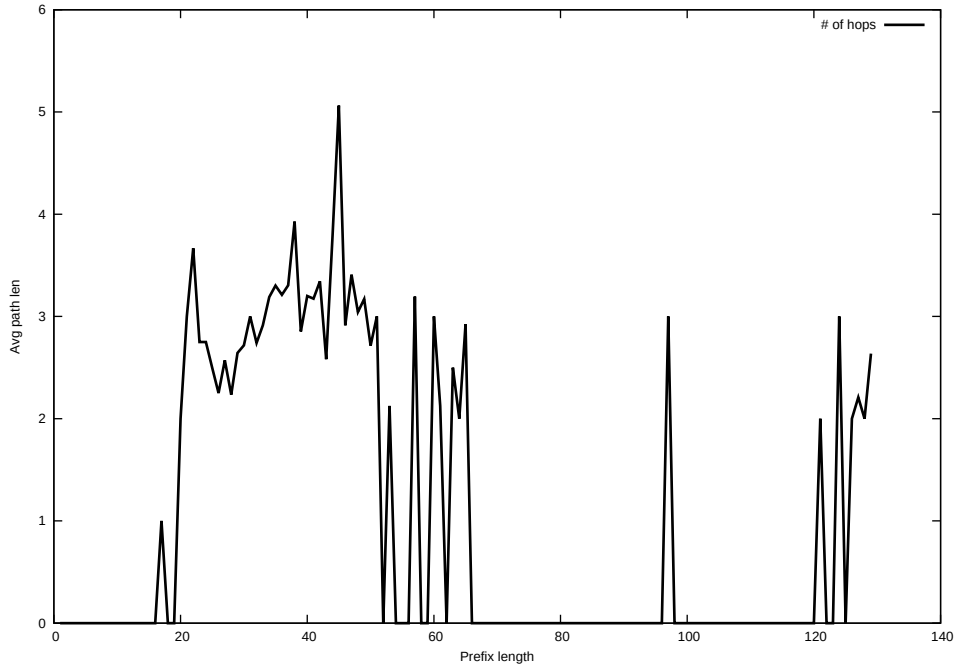
2014-05-27



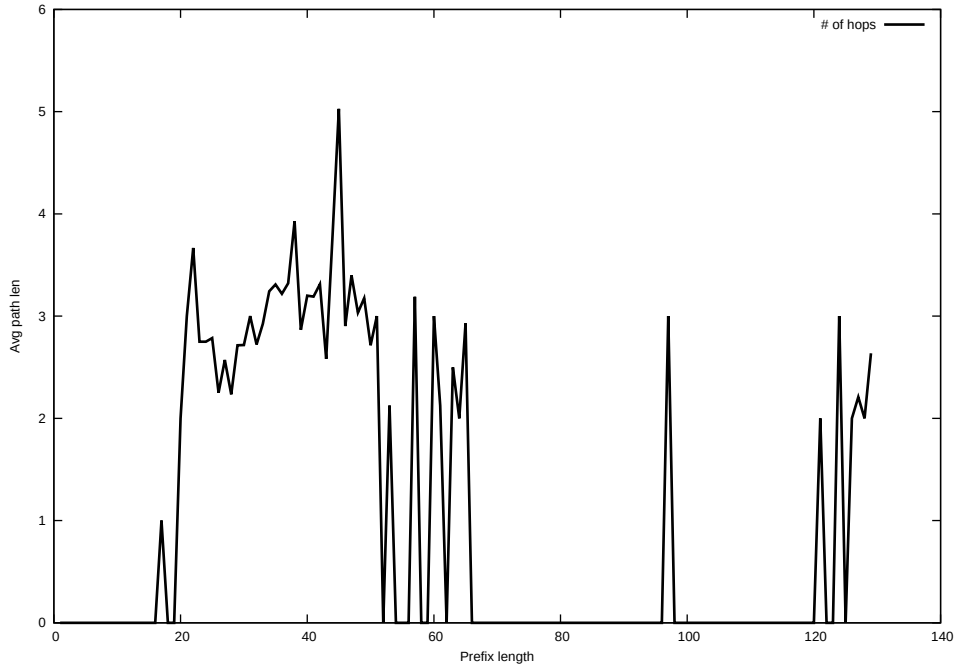
2014-05-28



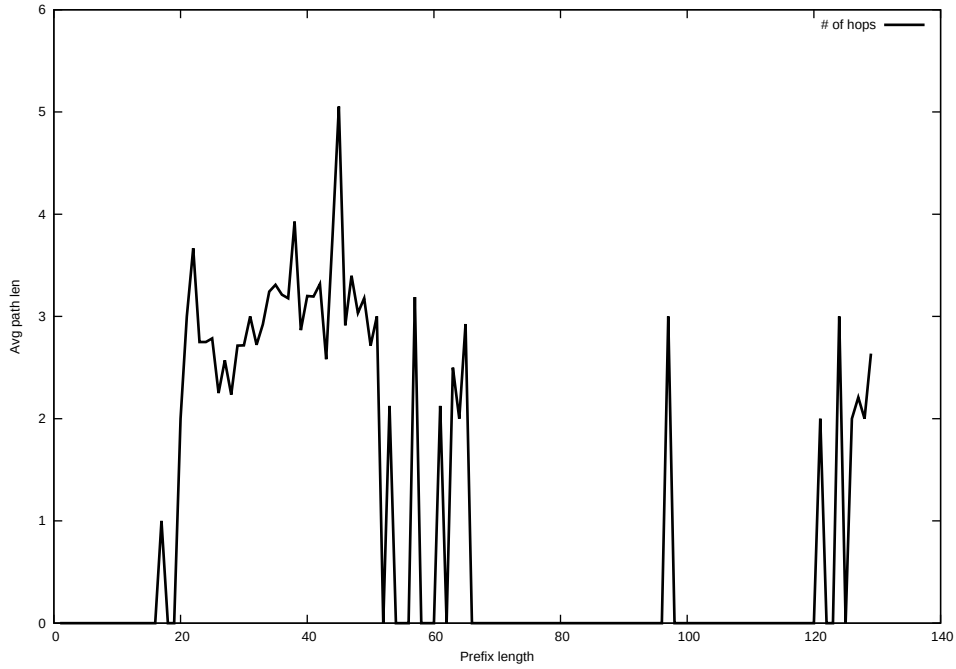
2014-05-29



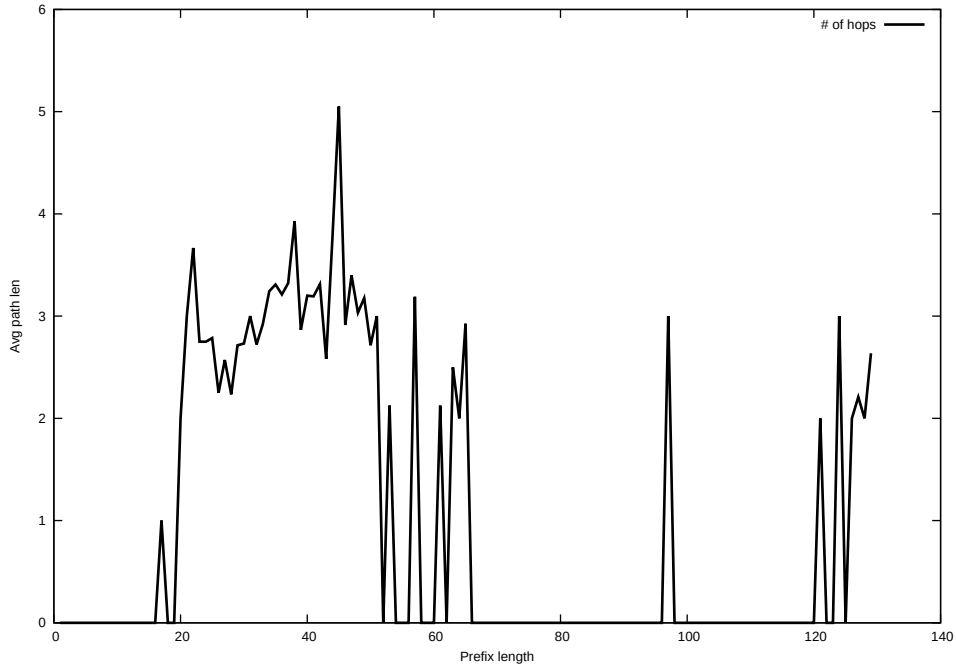
2014-05-30



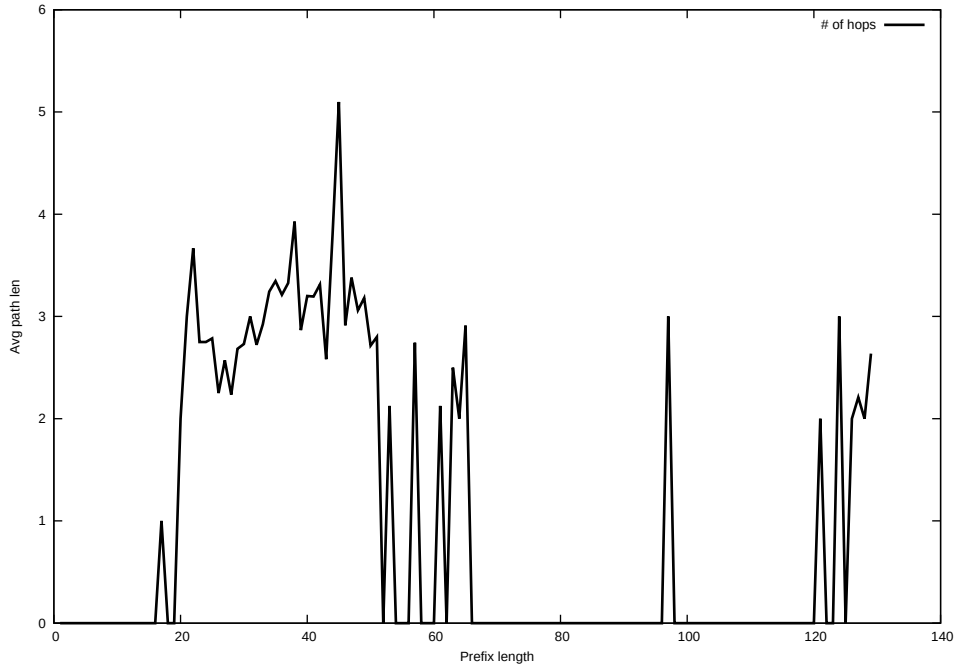
2014-05-31



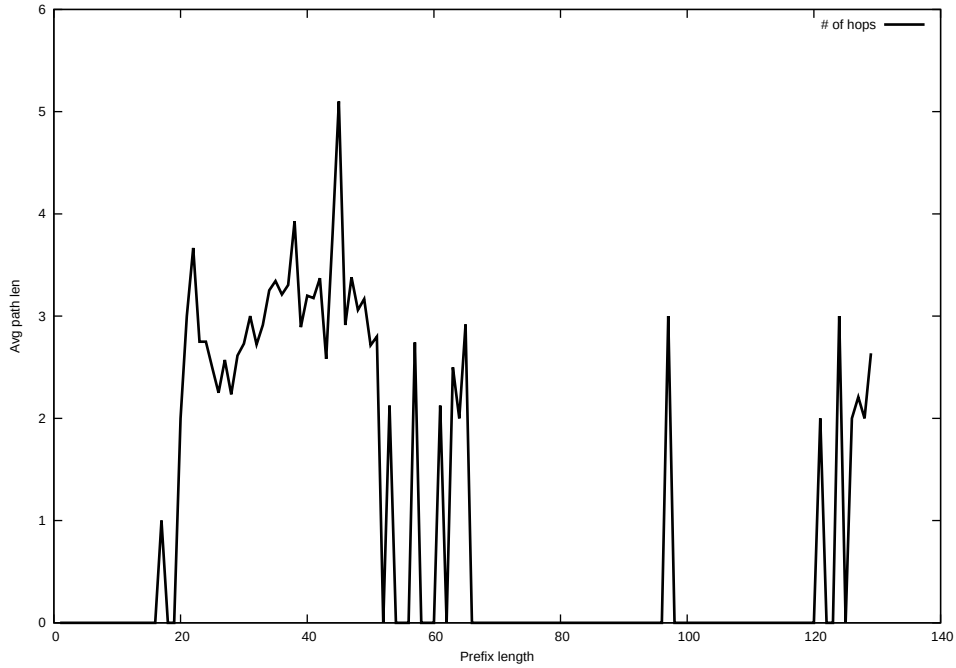
2014-06-01



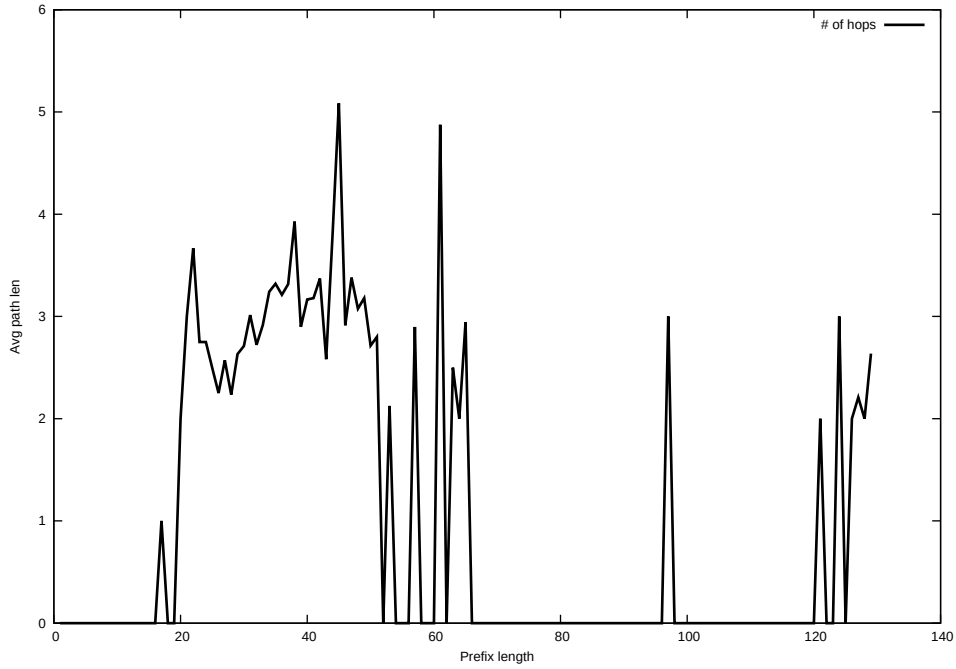
2014-06-02



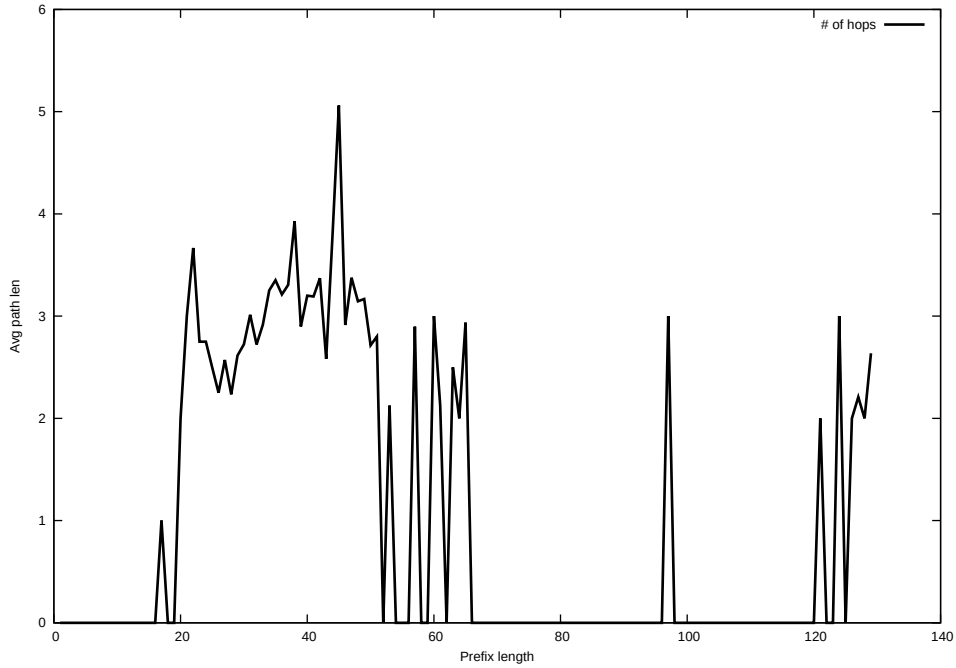
2014-06-03



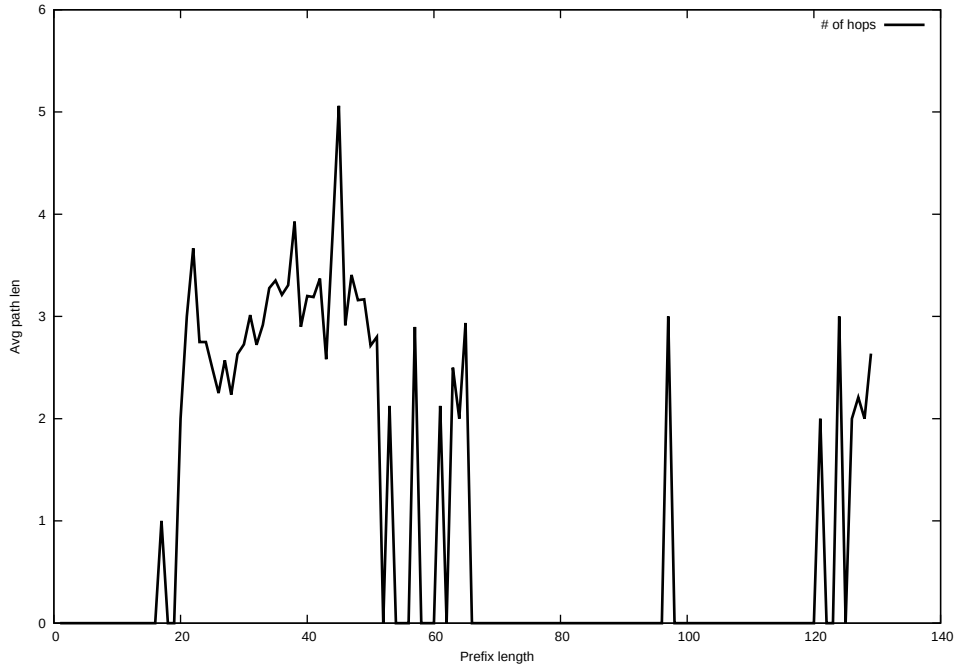
2014-06-04



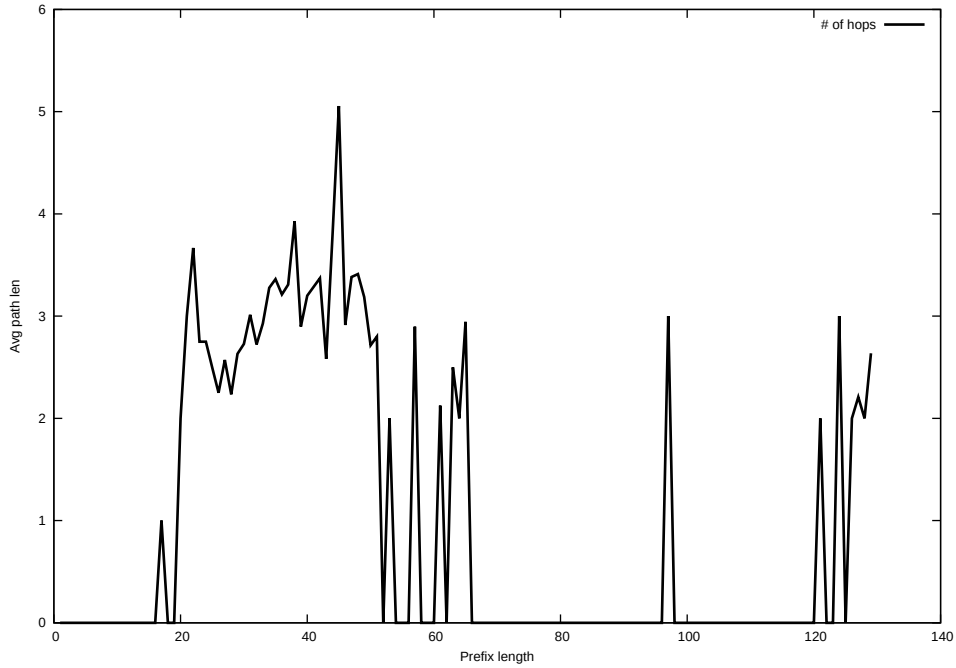
2014-06-05



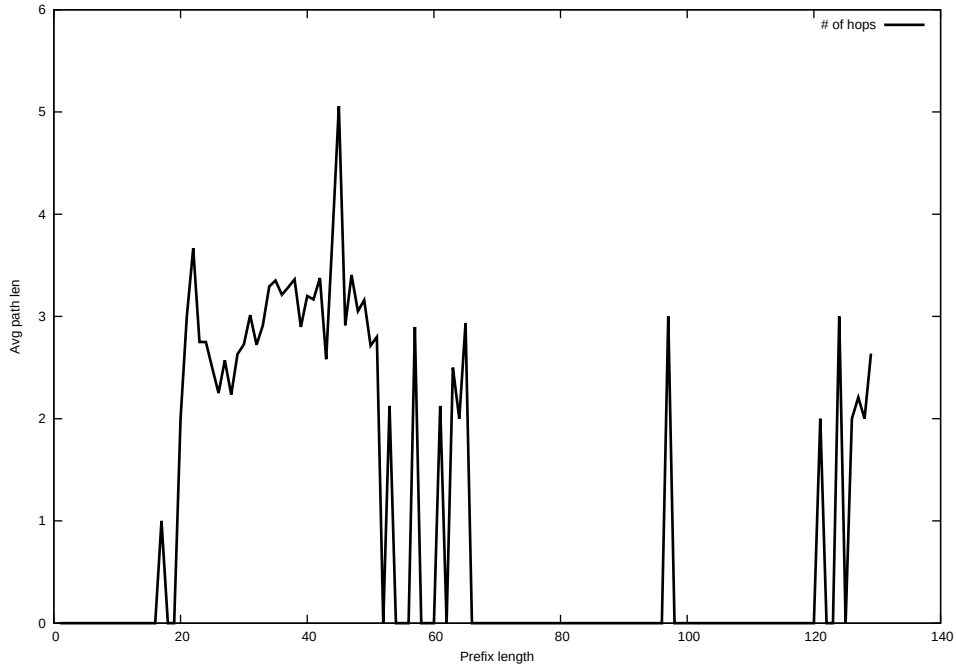
2014-06-06



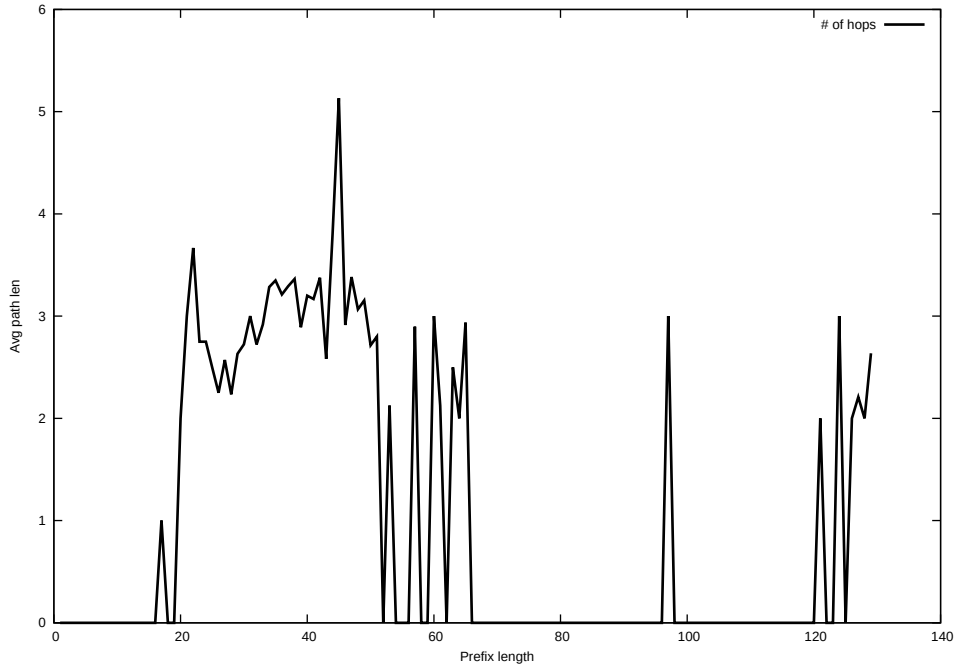
2014-06-07



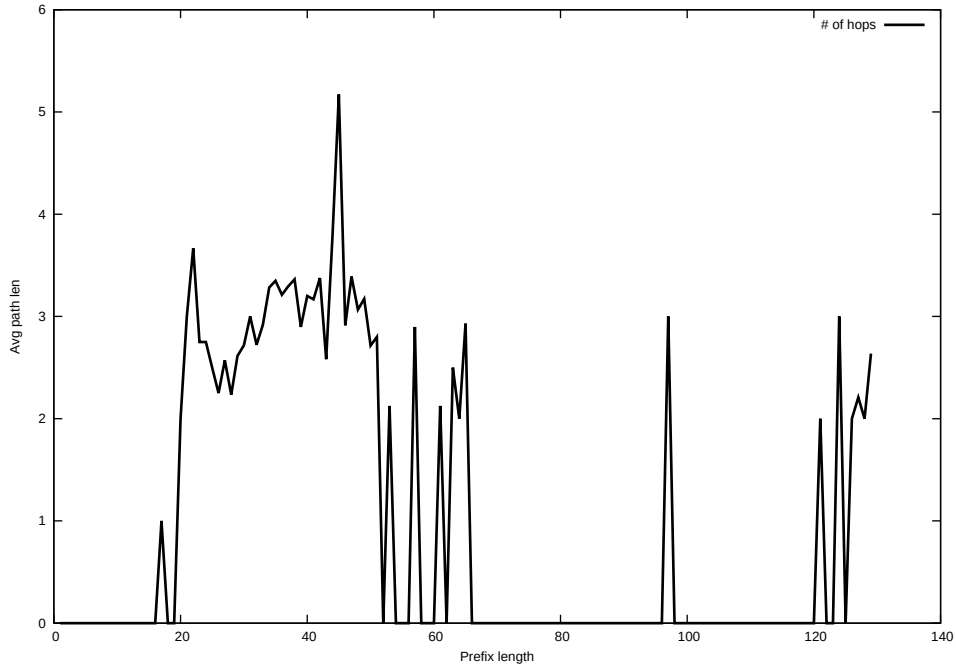
2014-06-08



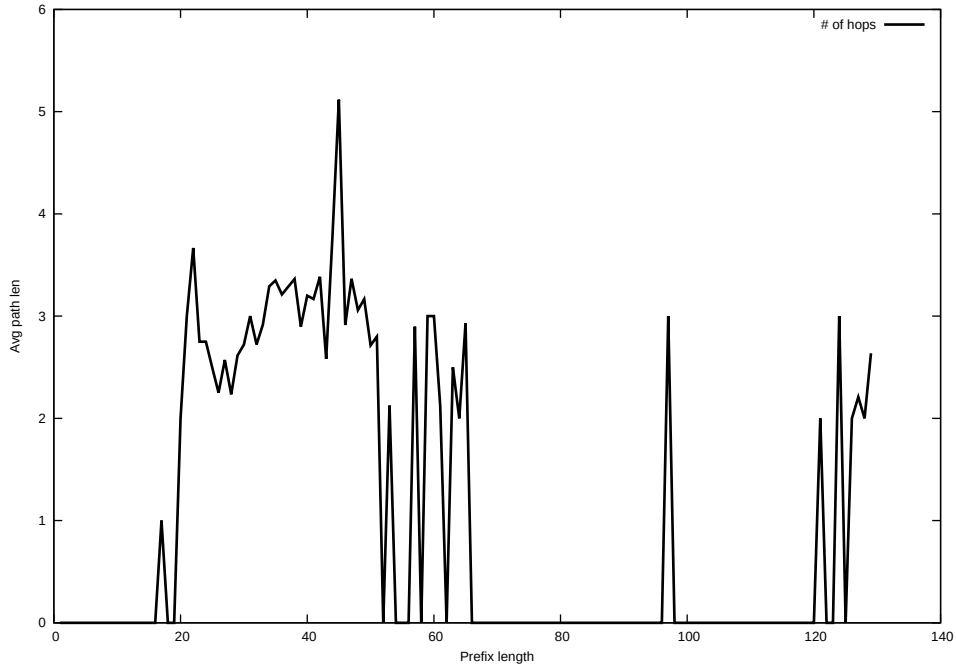
2014-06-09



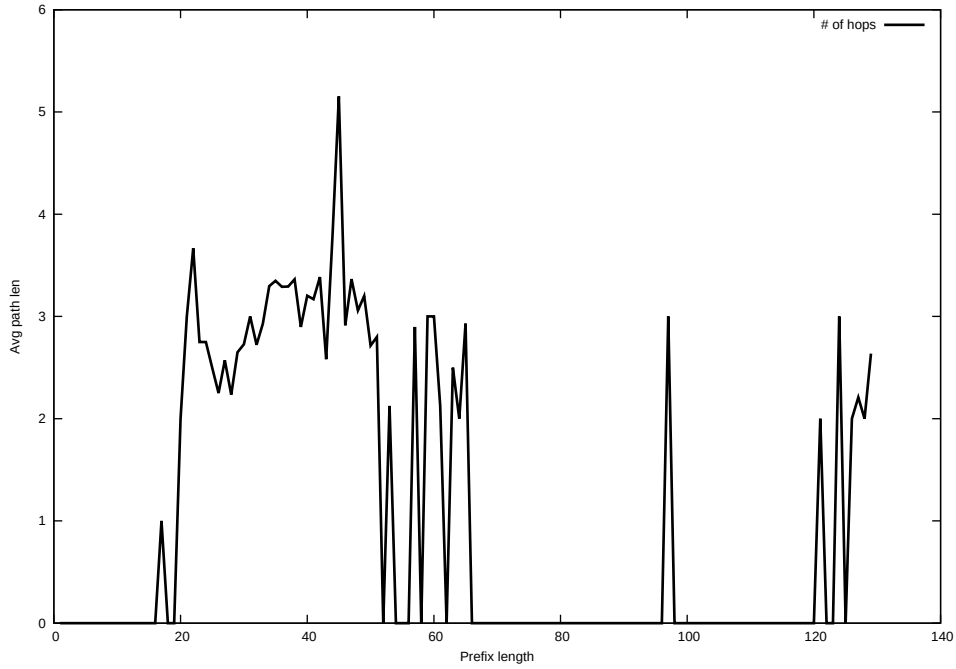
2014-06-10



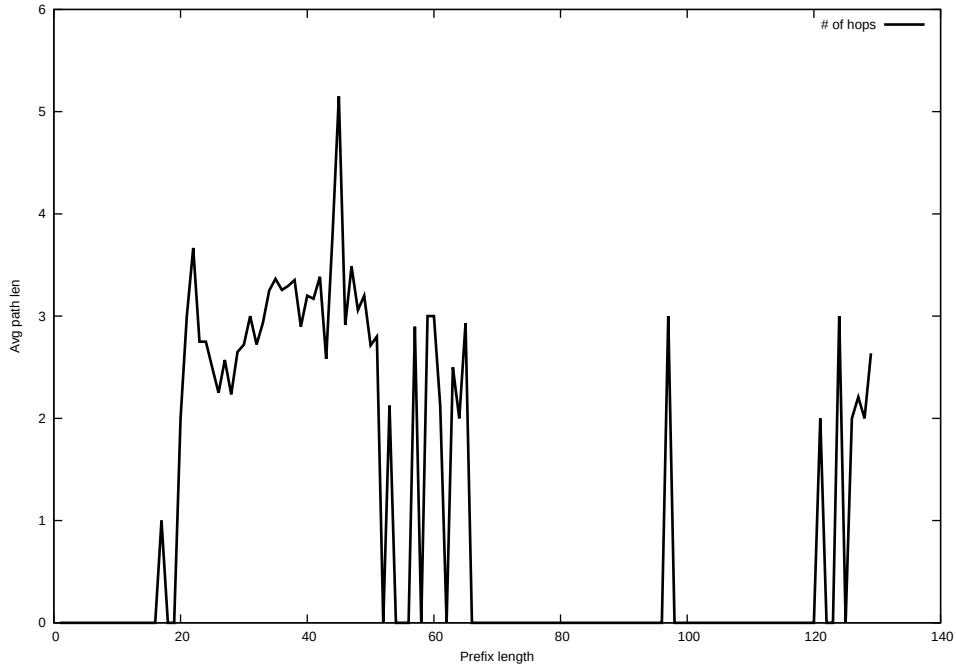
2014-06-11



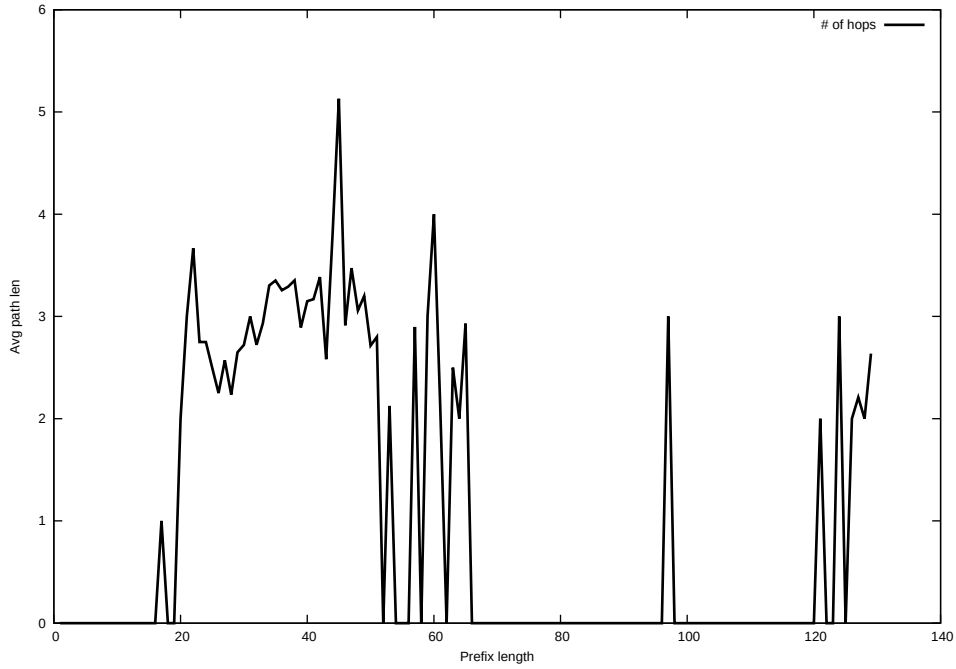
2014-06-12



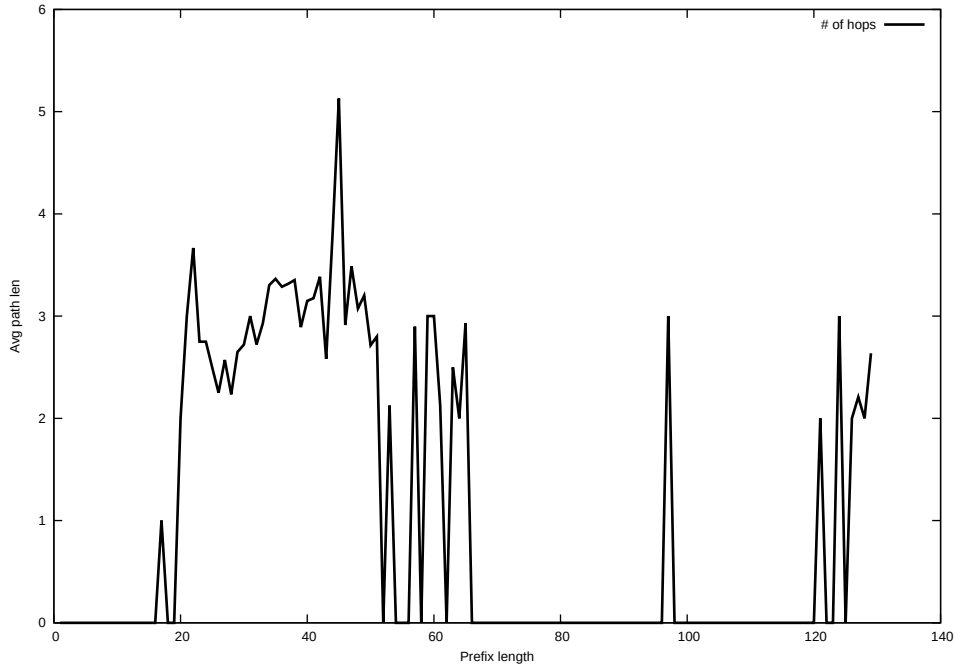
2014-06-13



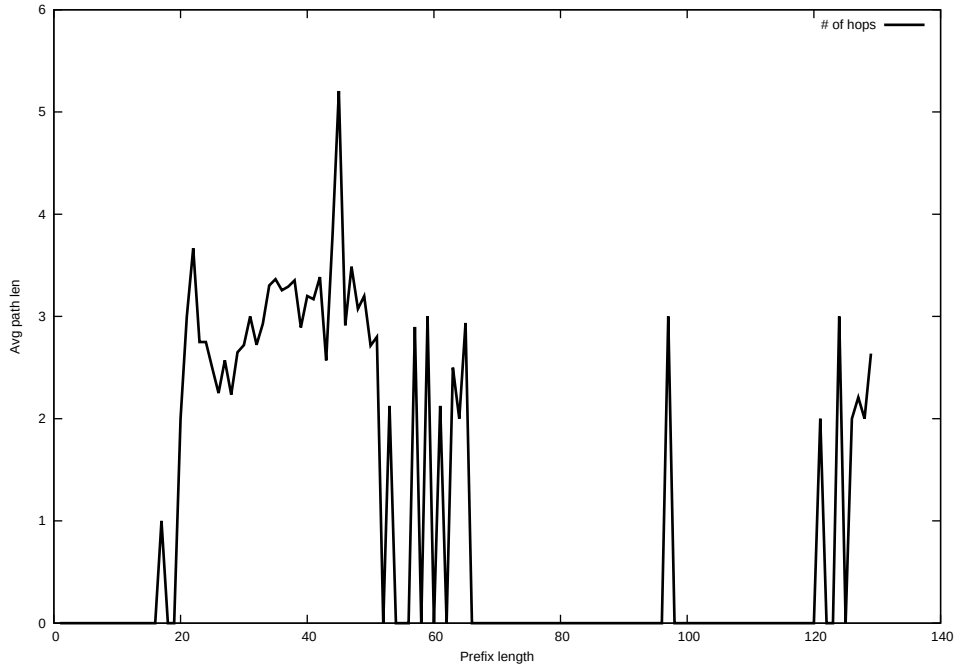
2014-06-14



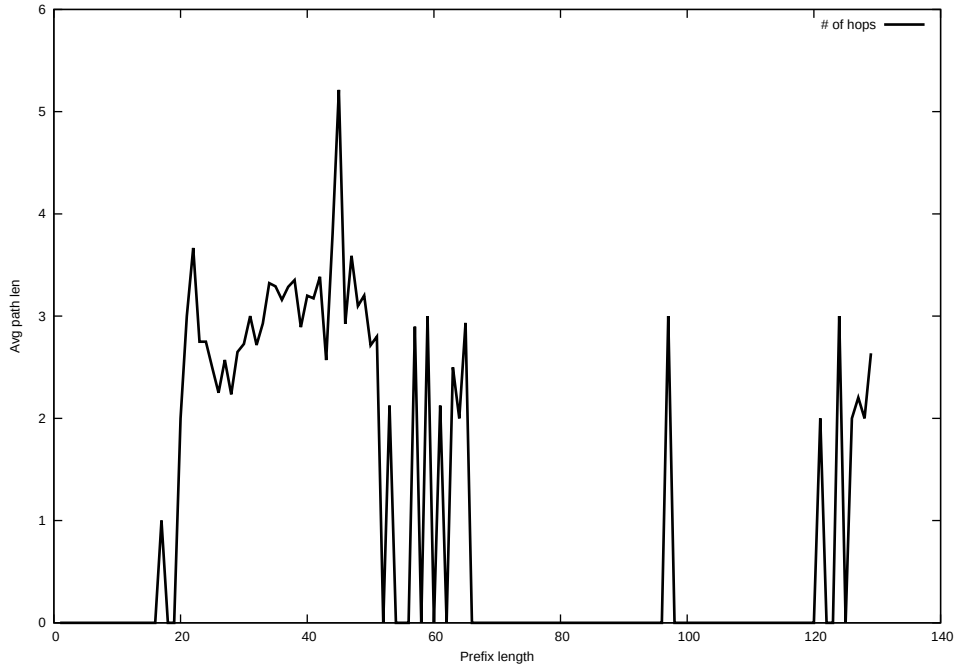
2014-06-15



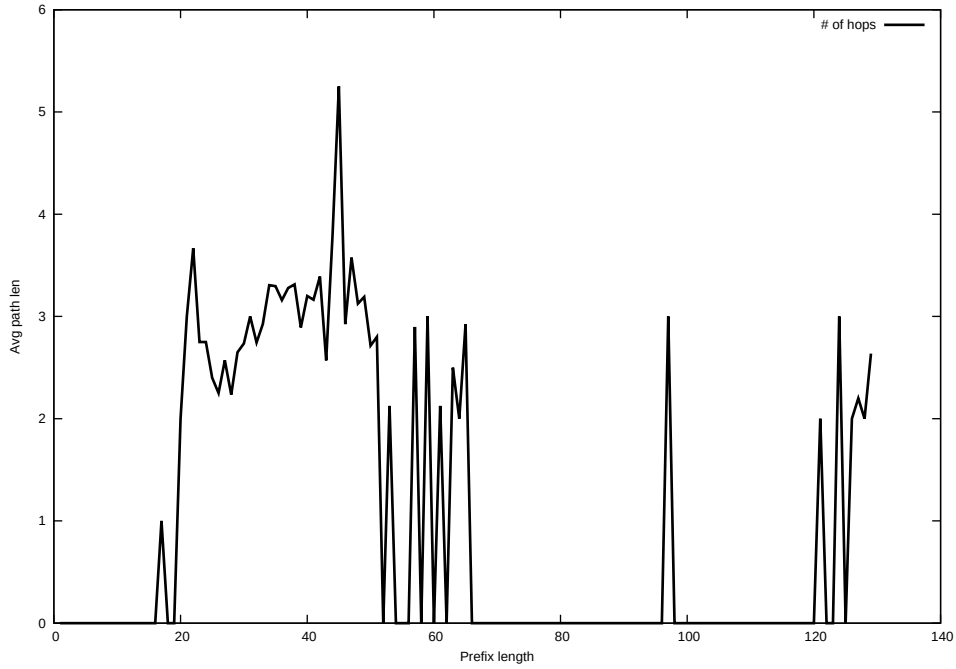
2014-06-16



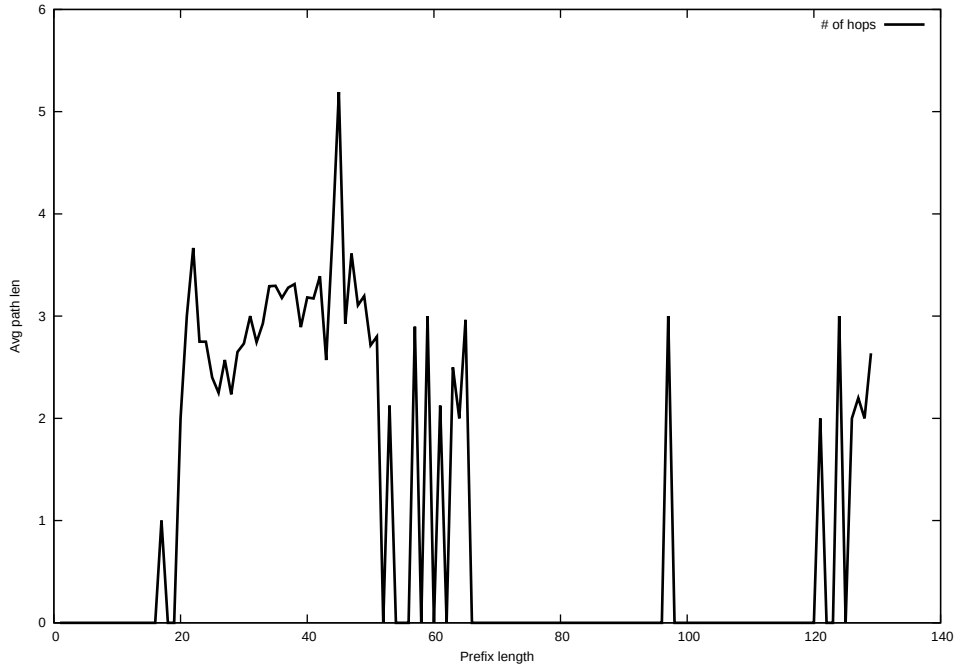
2014-06-17



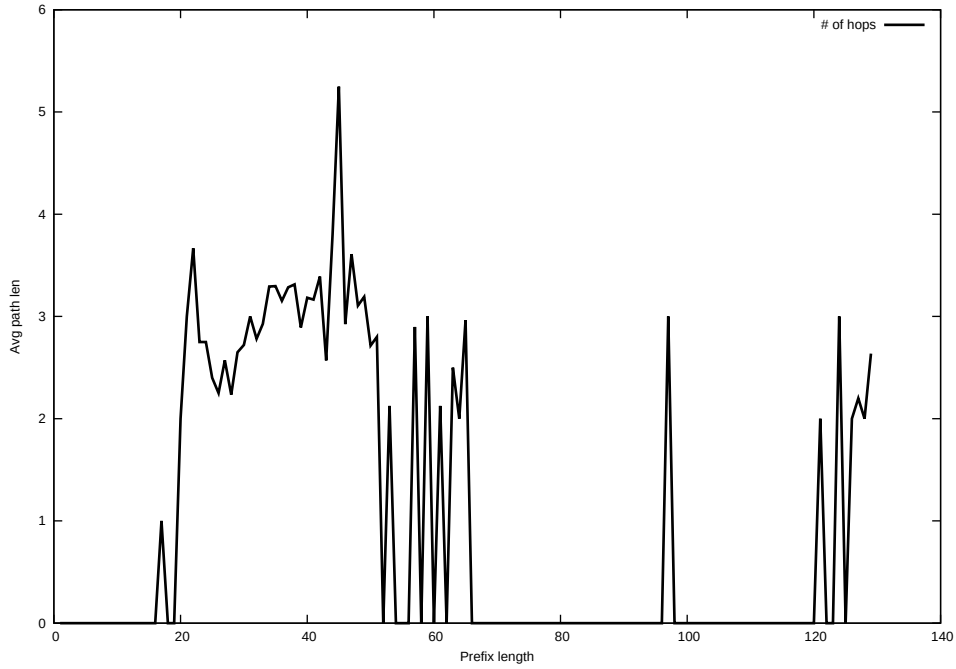
2014-06-18



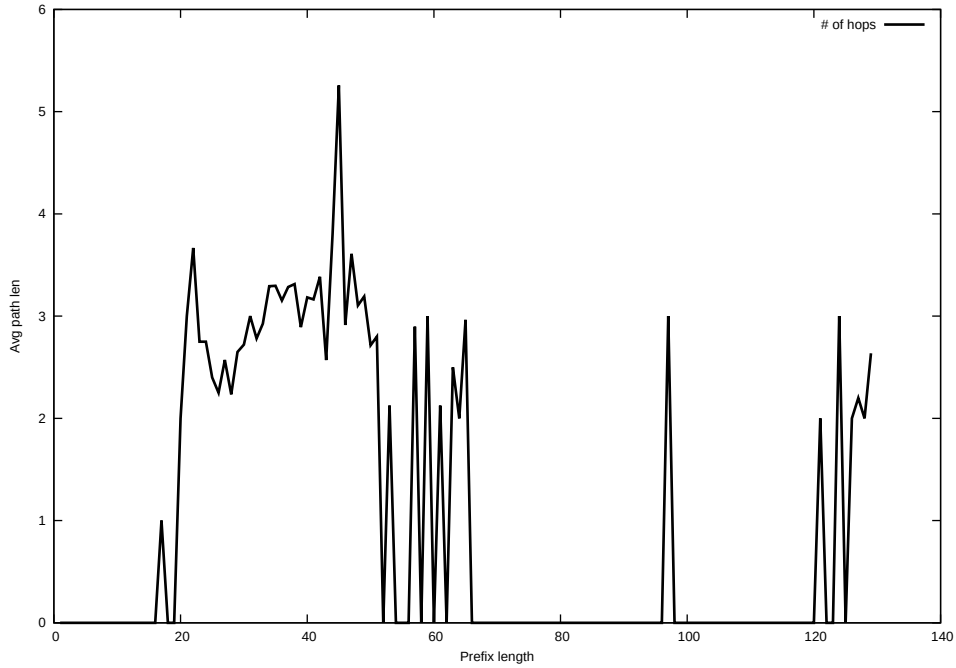
2014-06-19



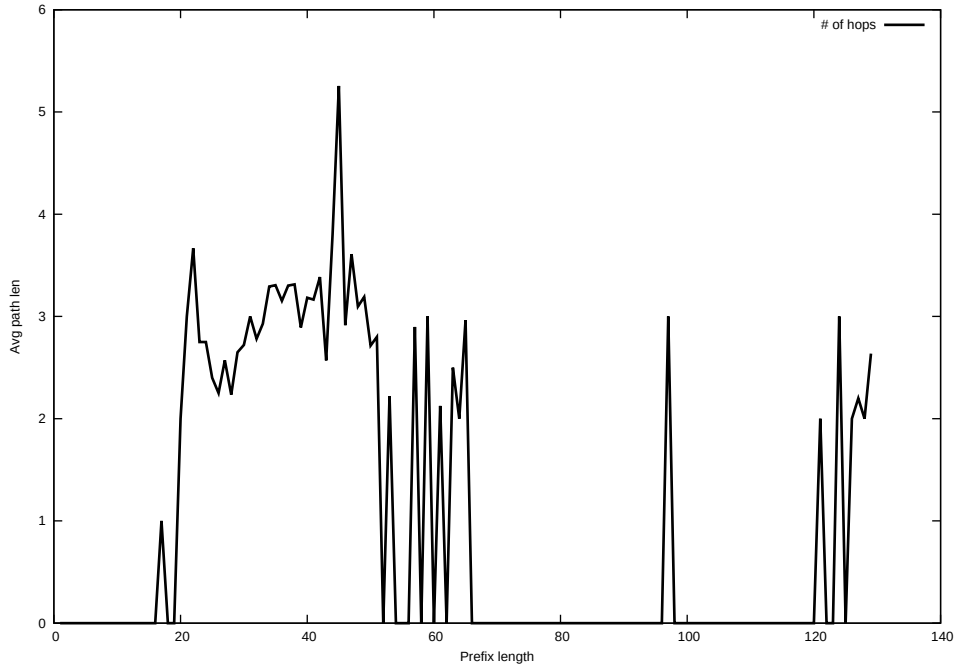
2014-06-20



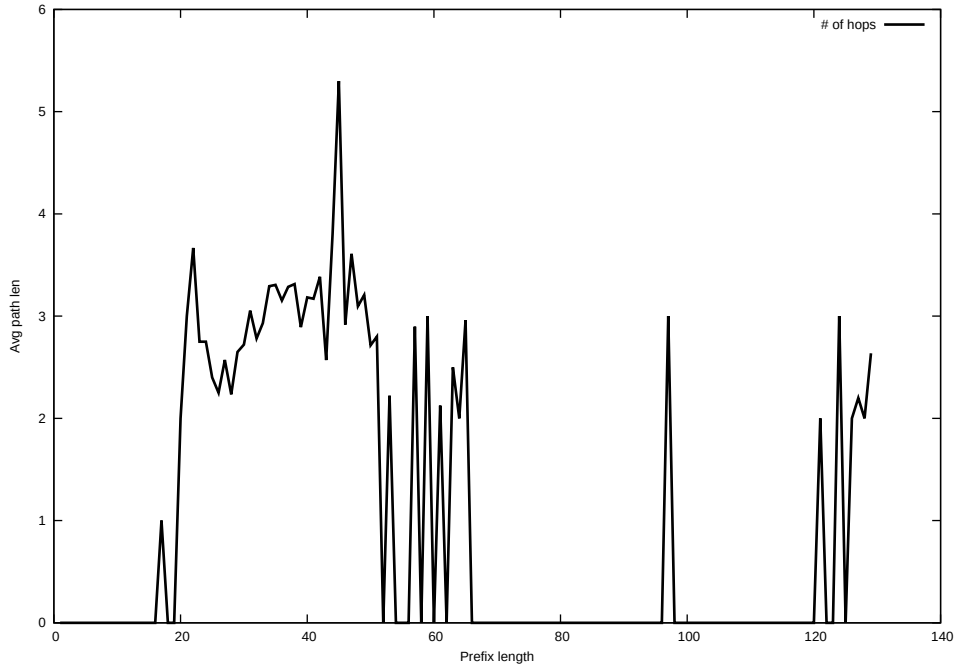
2014-06-21



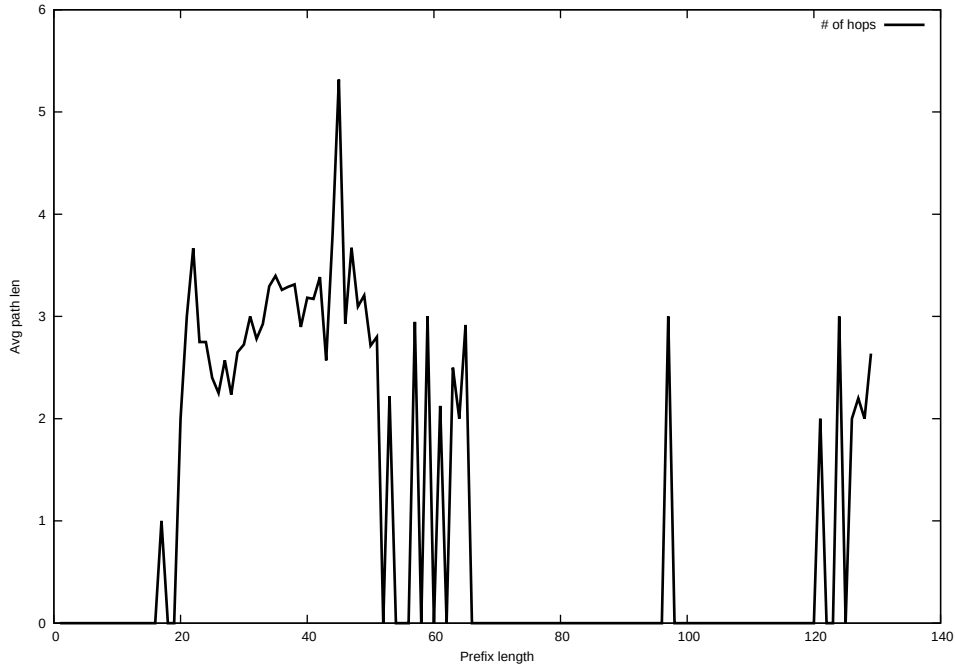
2014-06-22



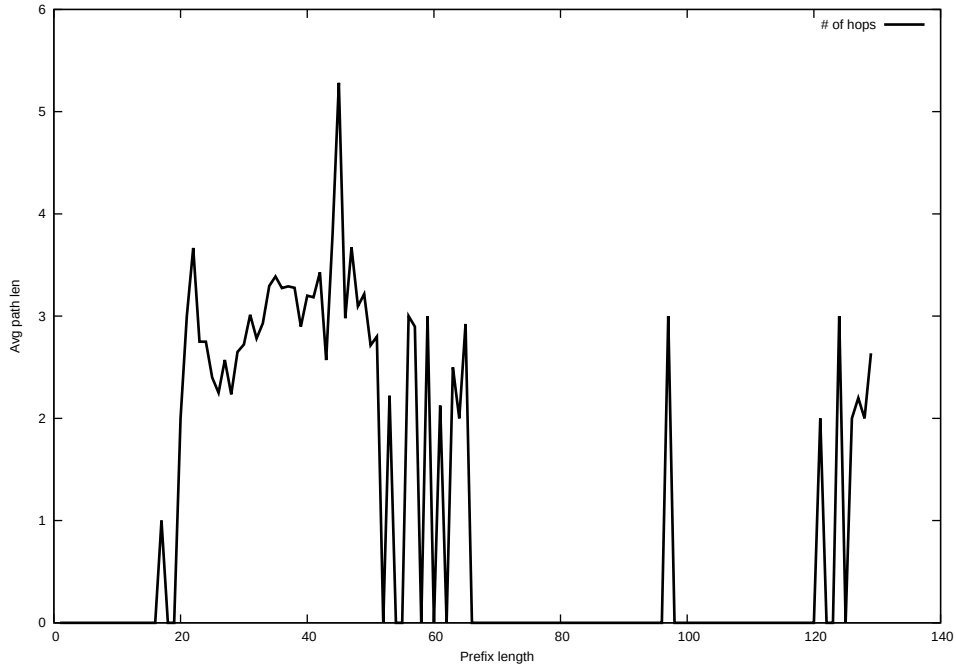
2014-06-23



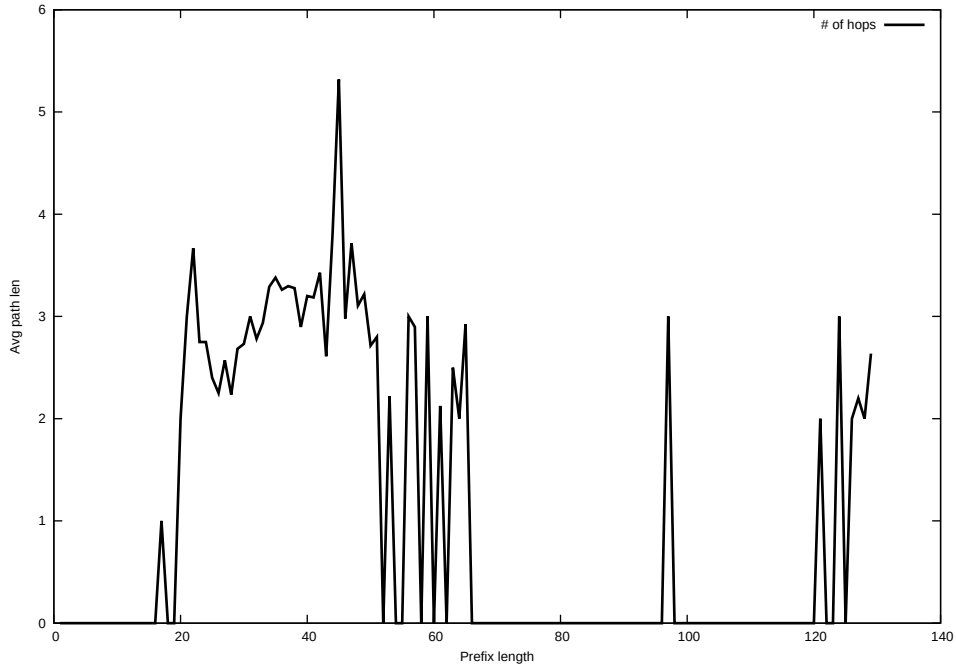
2014-06-24



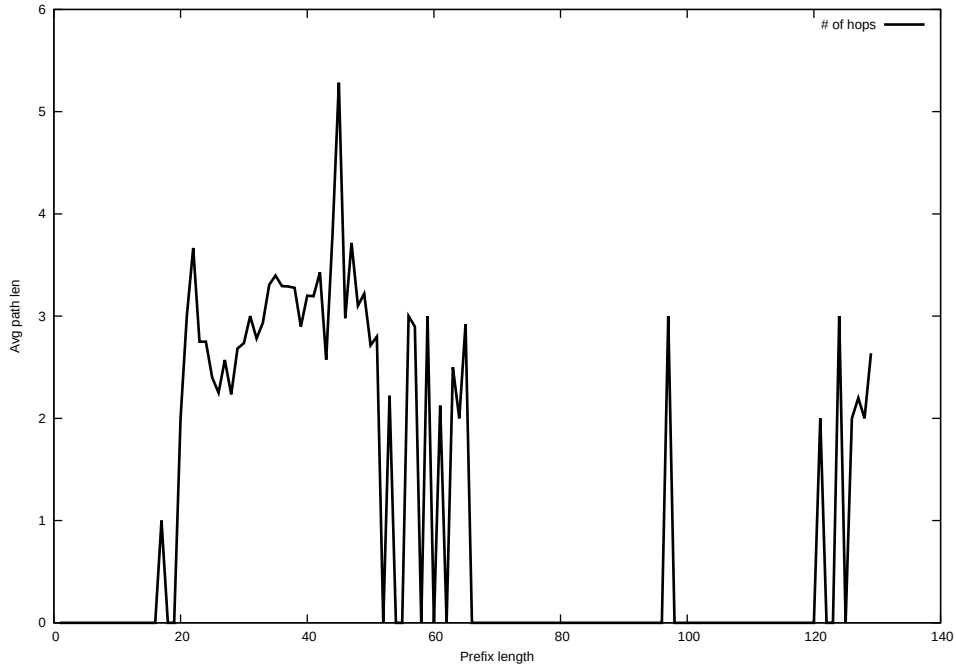
2014-06-25



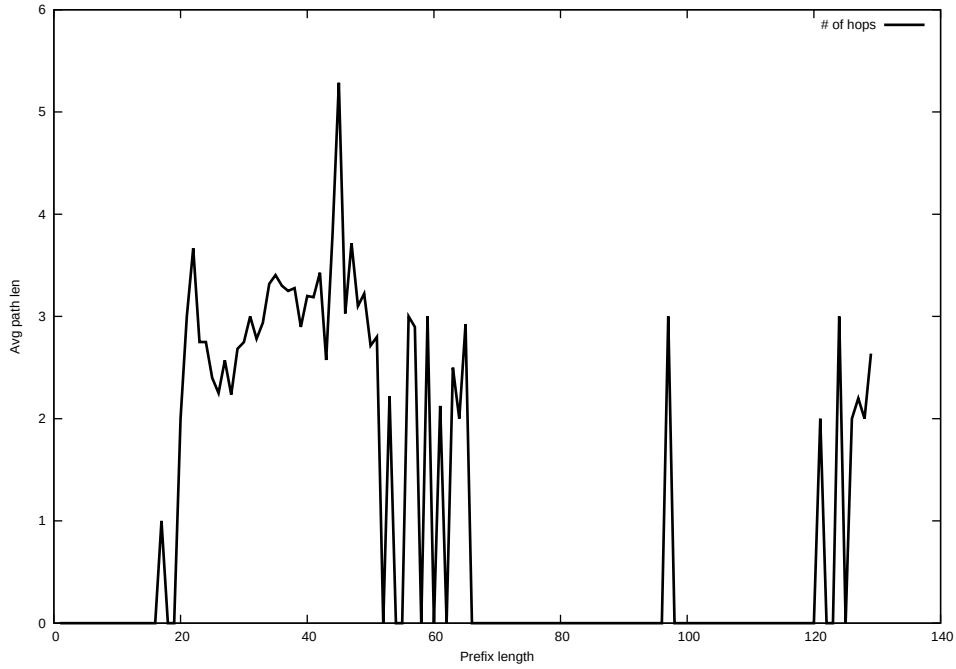
2014-06-26



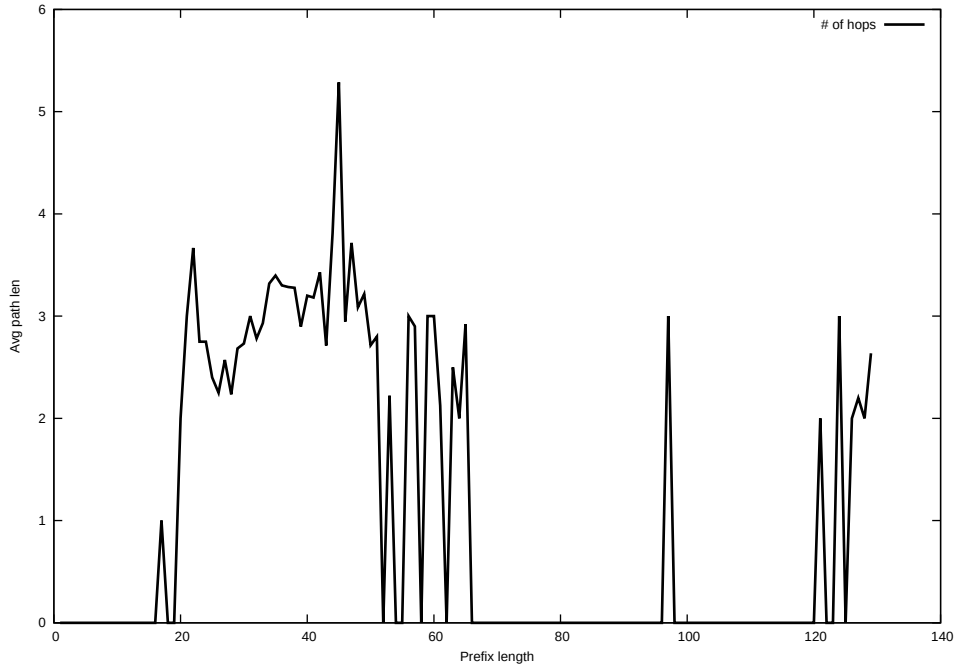
2014-06-27



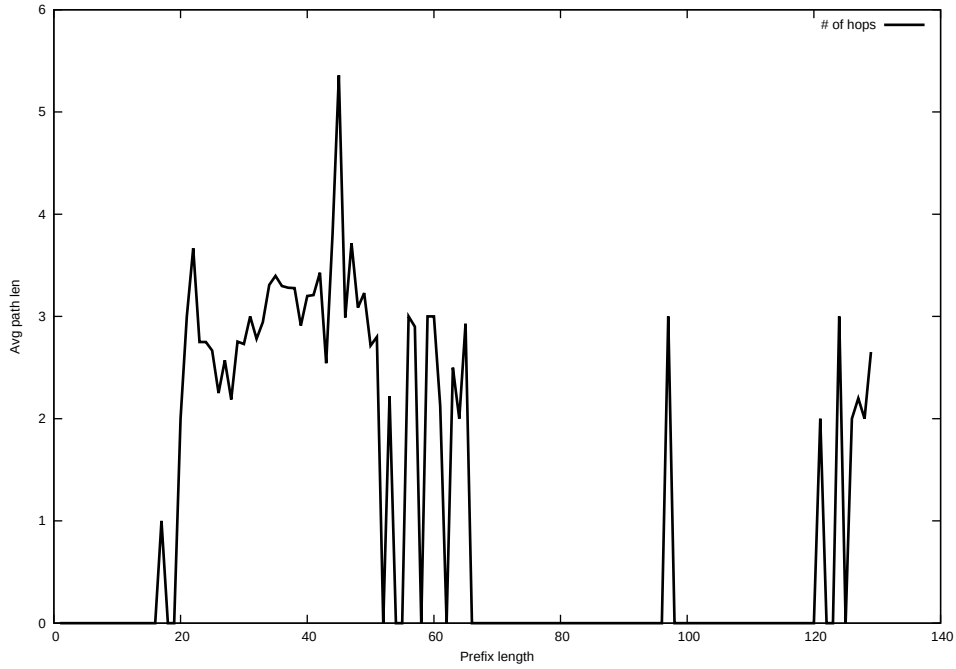
2014-06-28



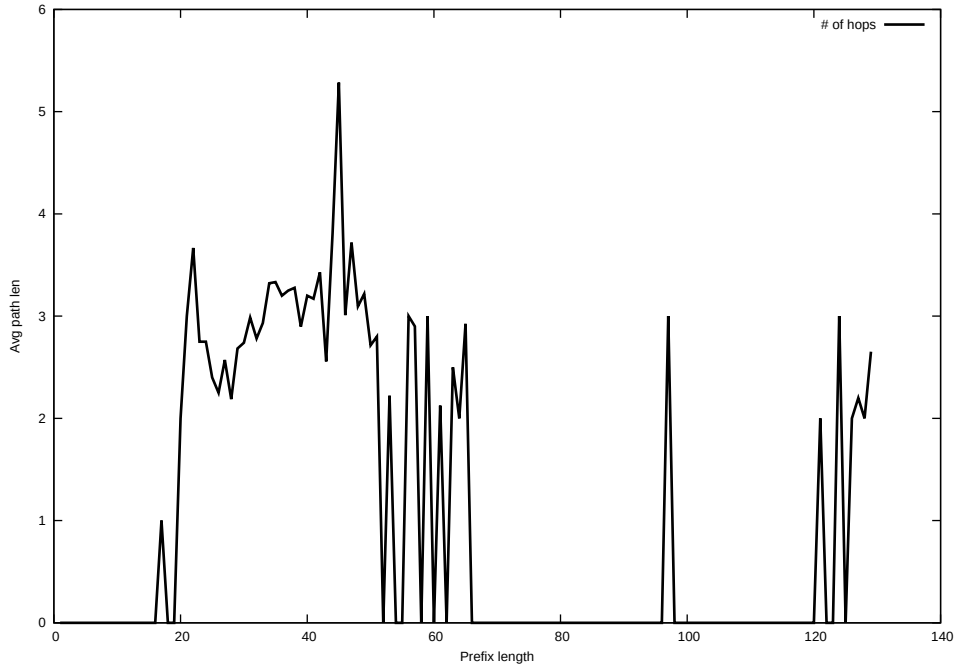
2014-06-29



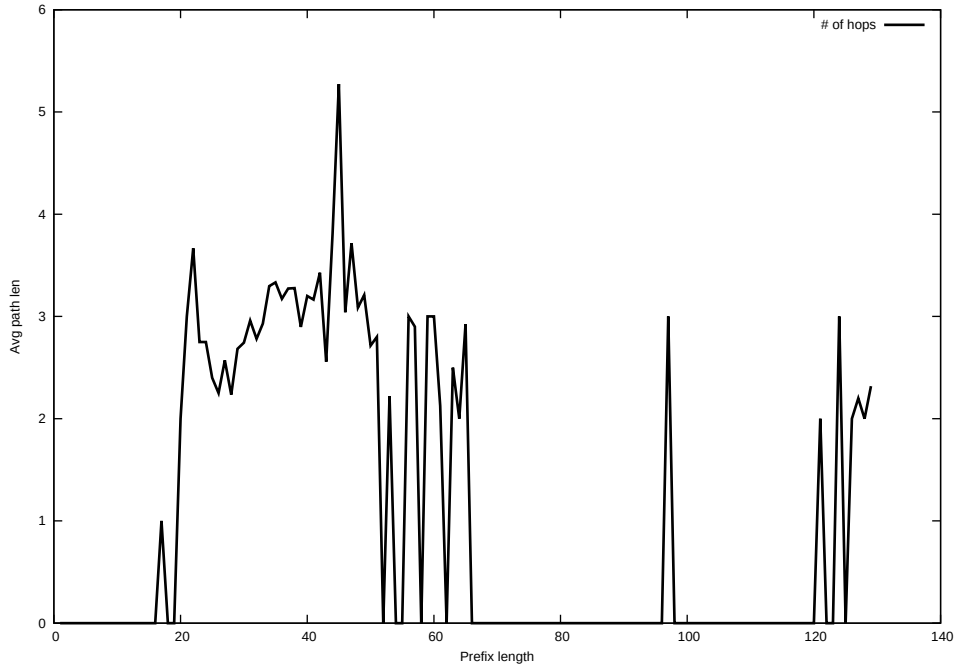
2014-06-30



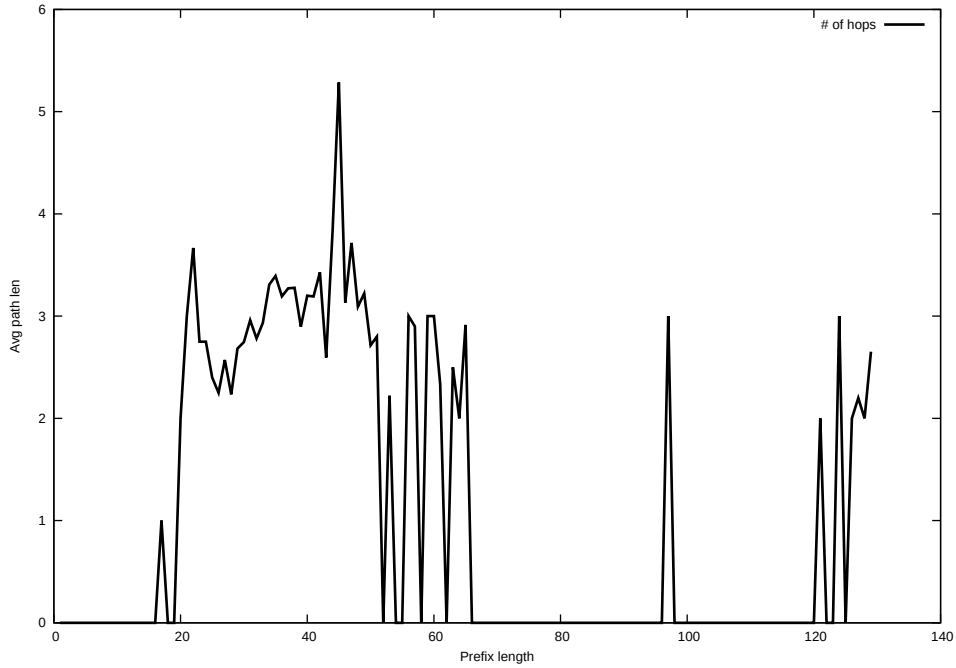
2014-07-01



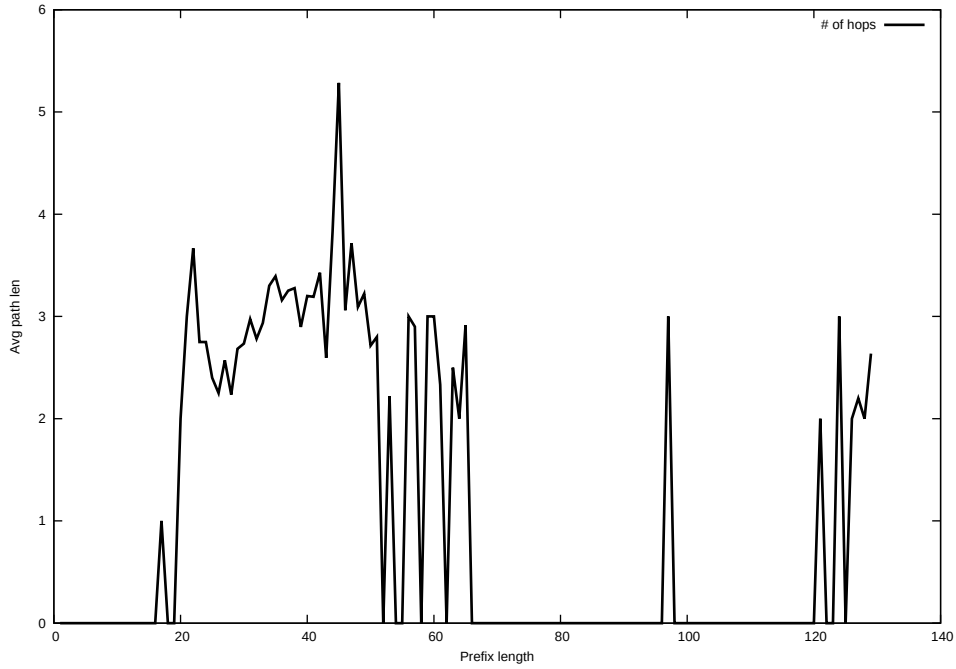
2014-07-02



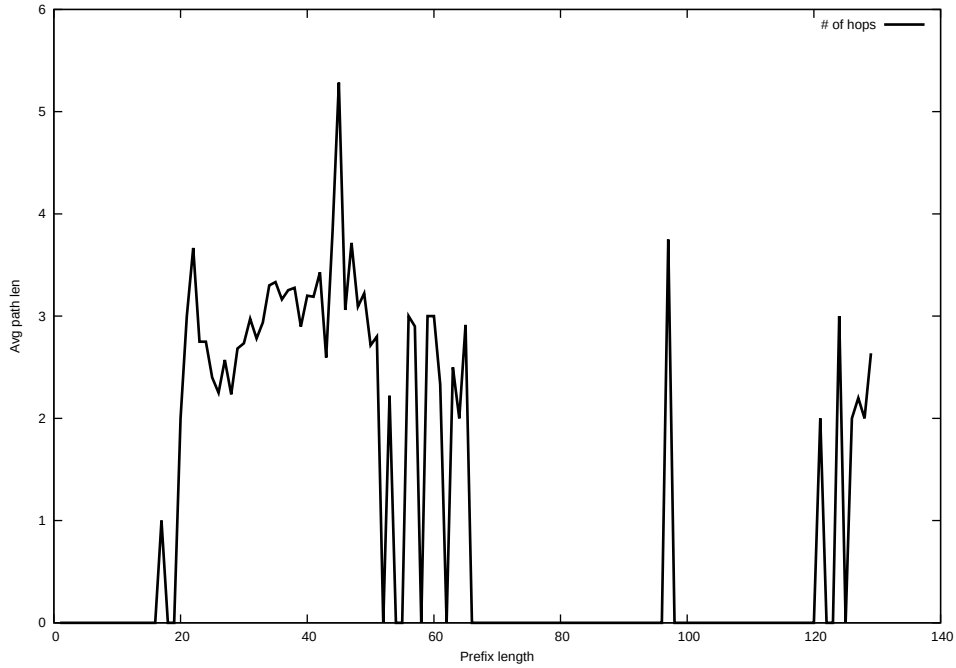
2014-07-03



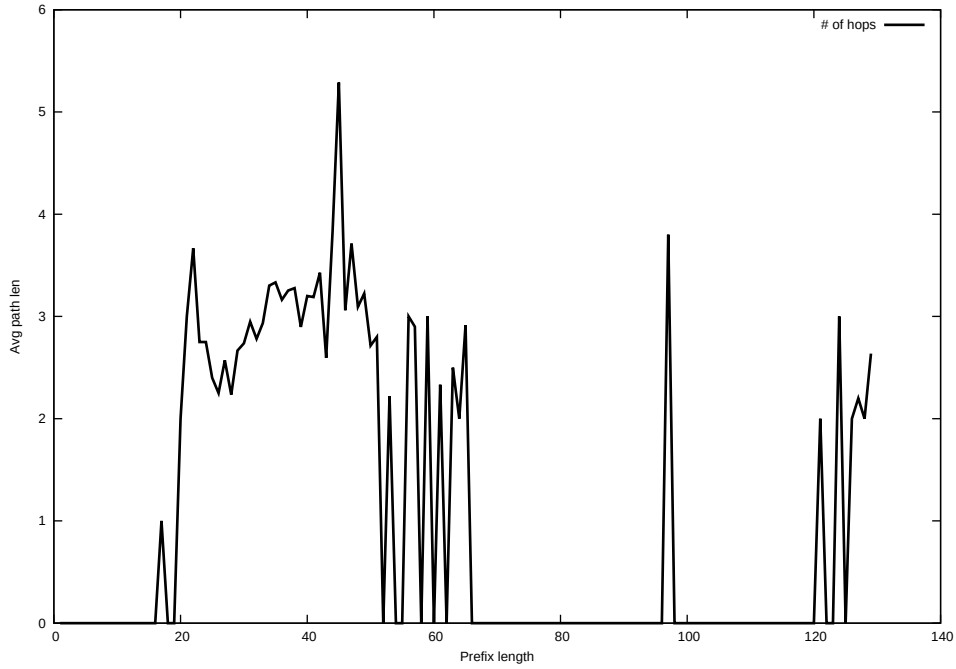
2014-07-04



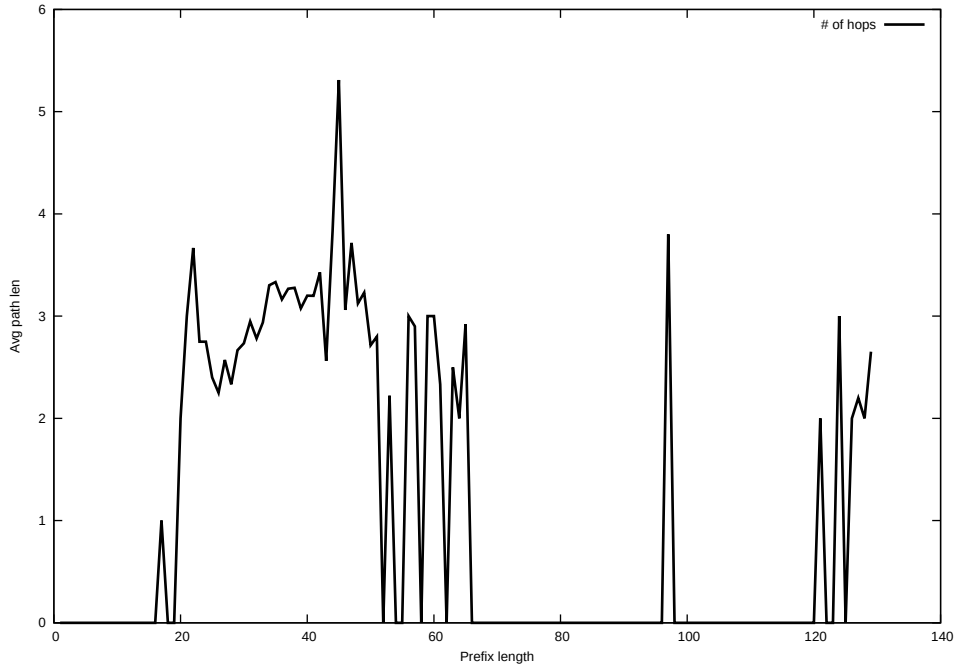
2014-07-05



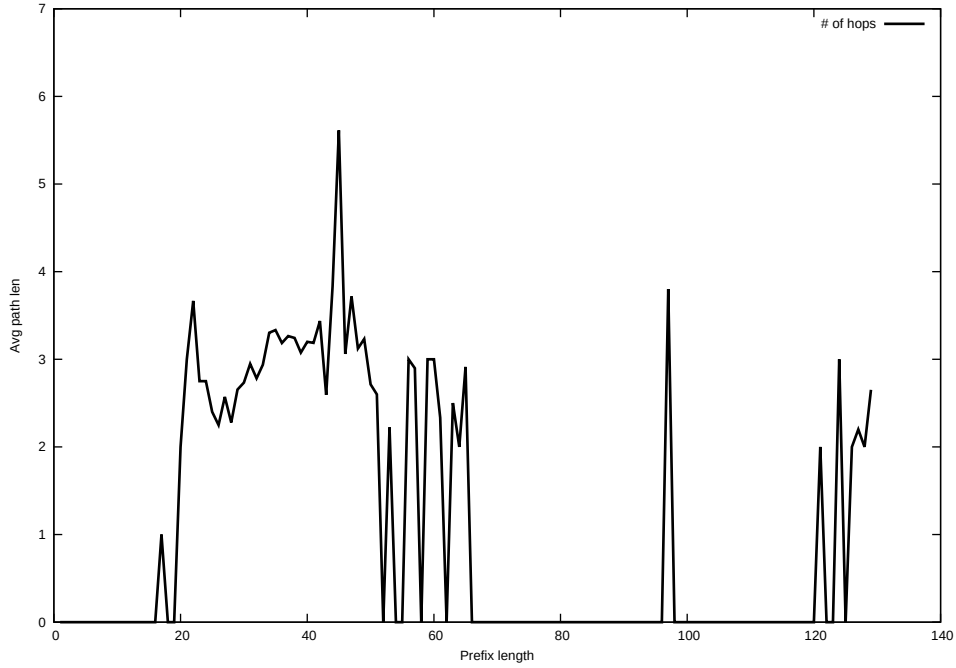
2014-07-06



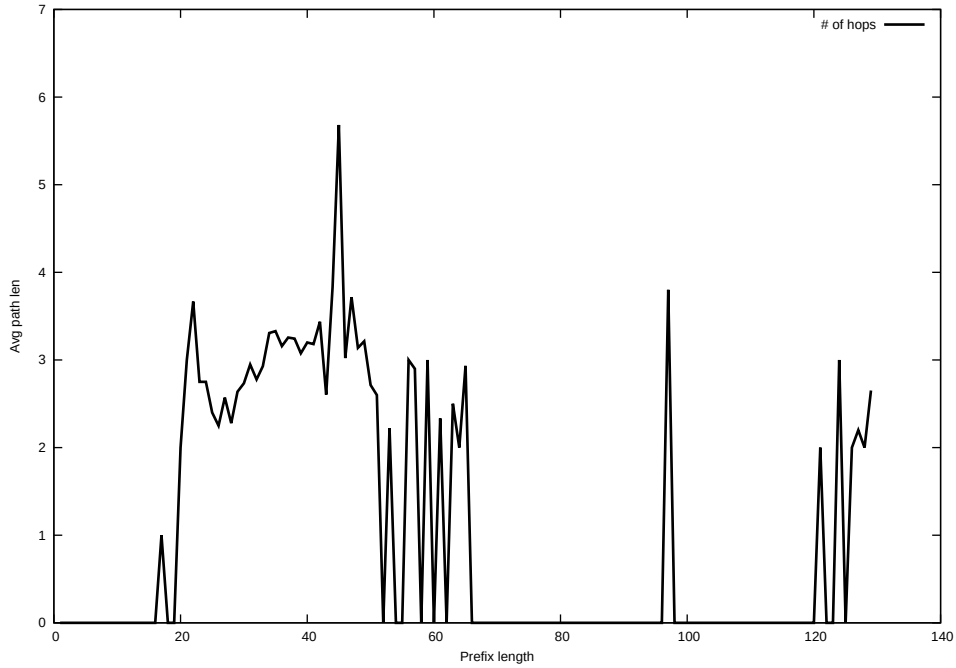
2014-07-07



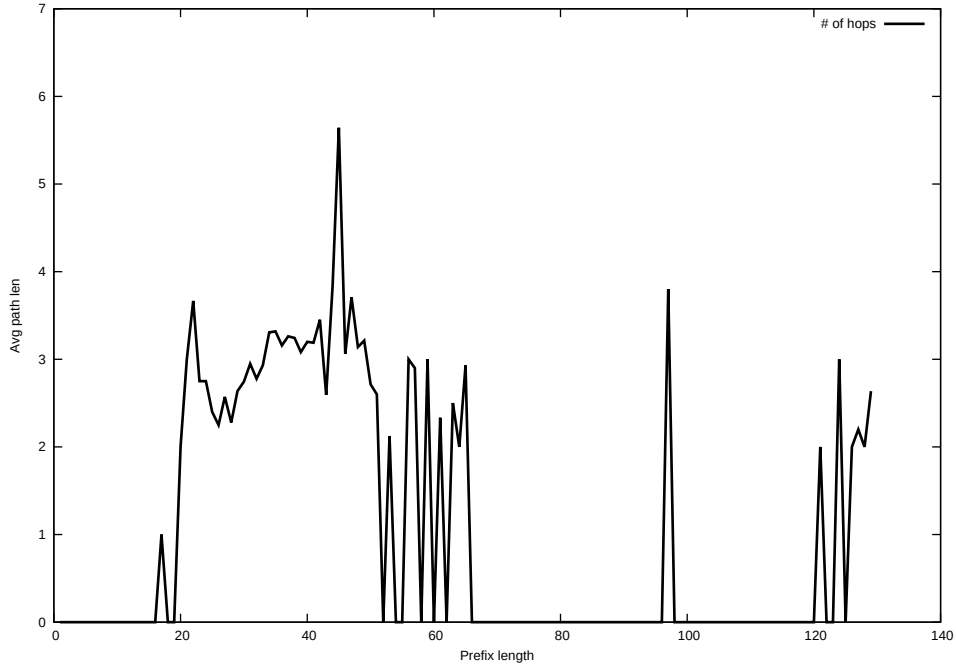
2014-07-08



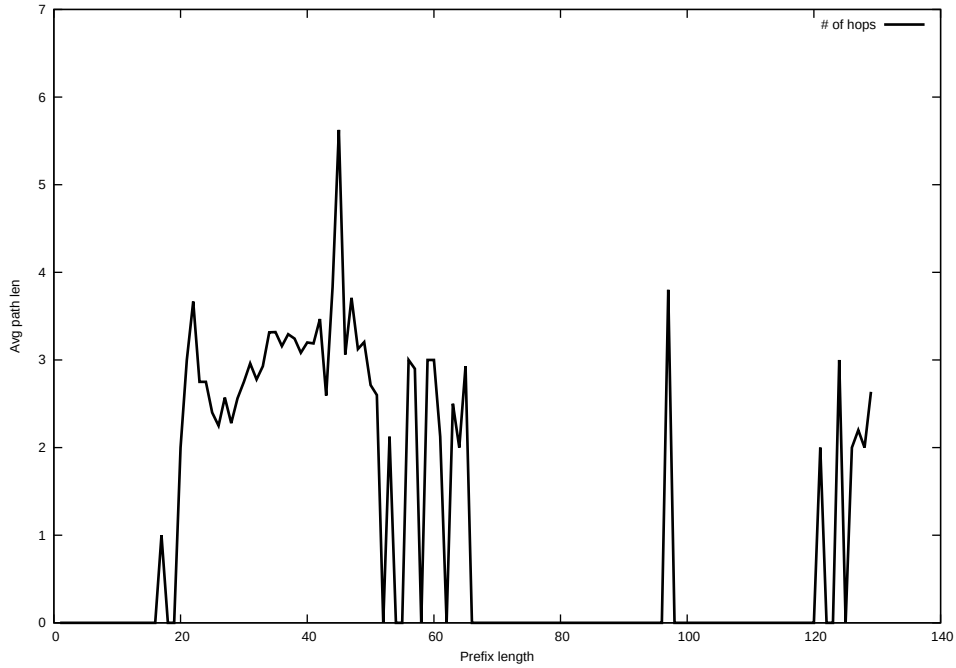
2014-07-09



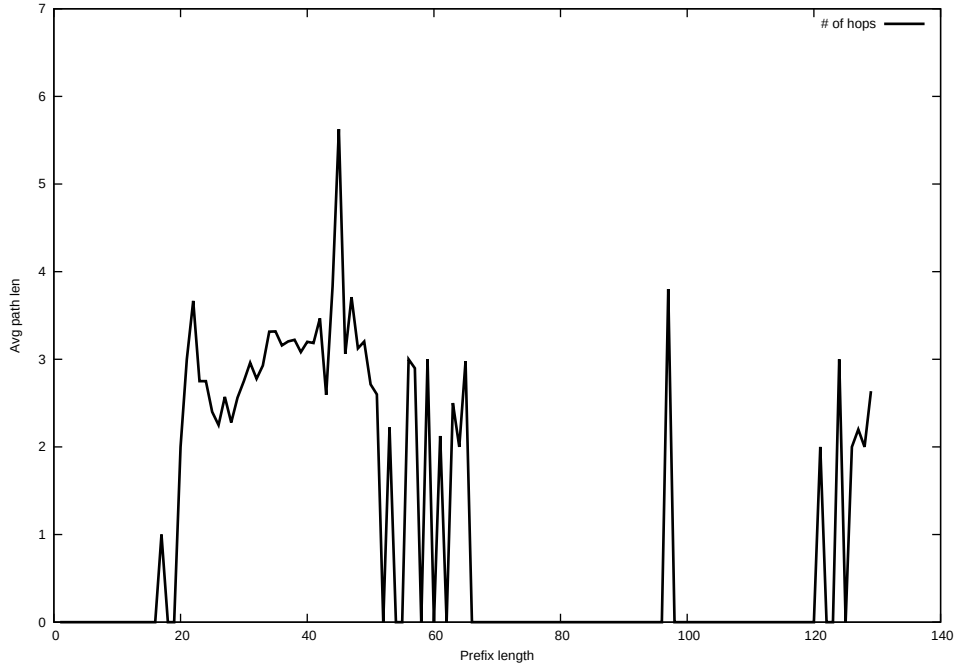
2014-07-10



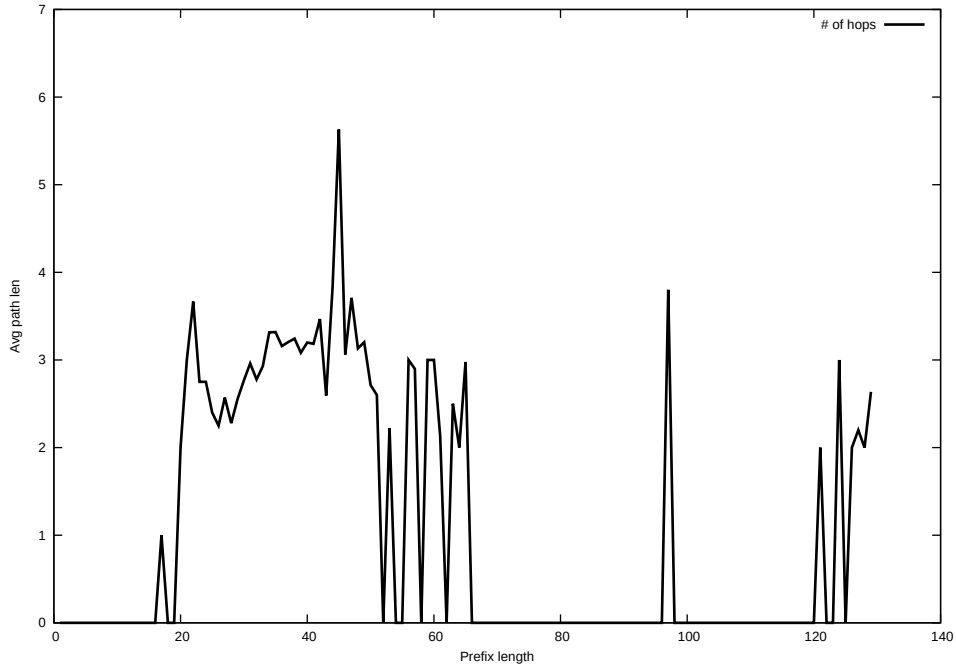
2014-07-11



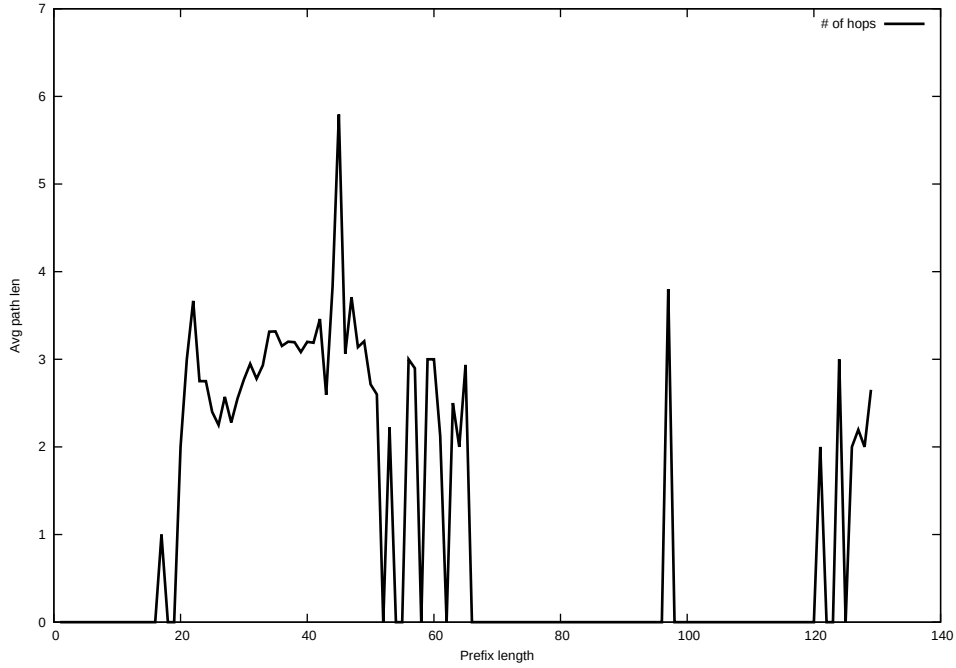
2014-07-12



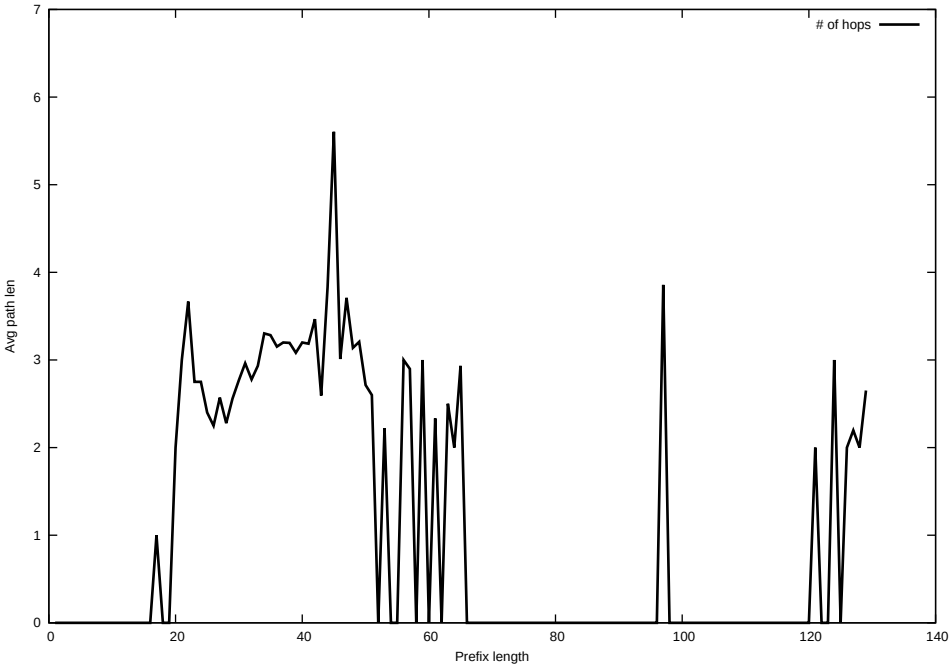
2014-07-13



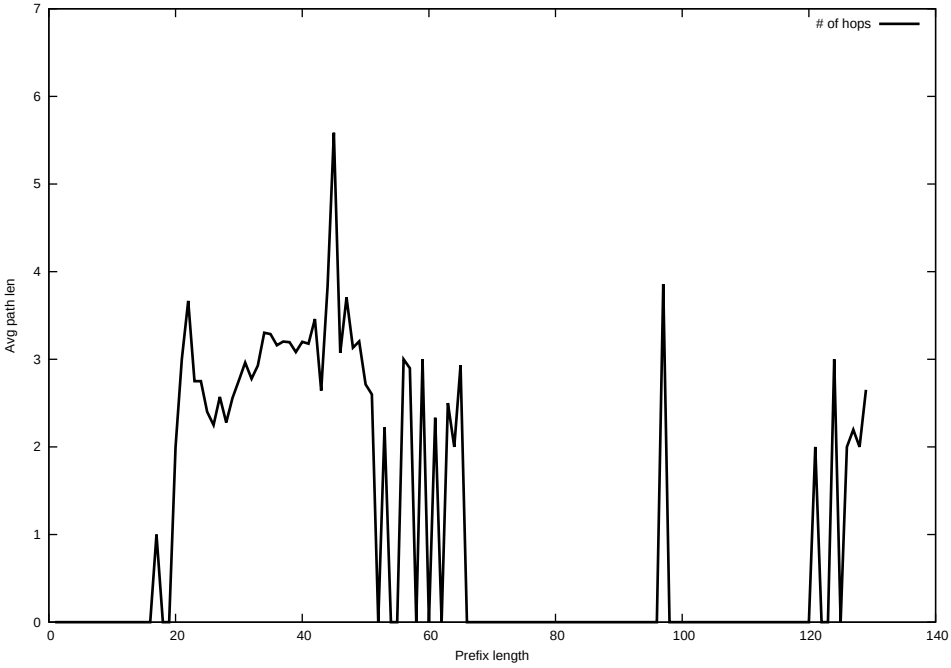
2014-07-14



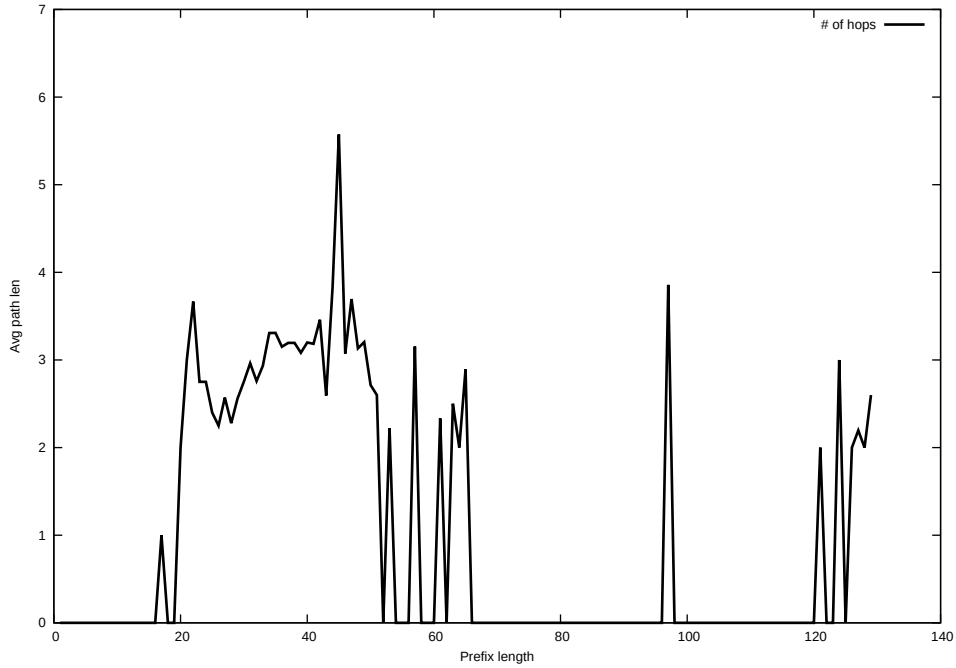
2014-07-15



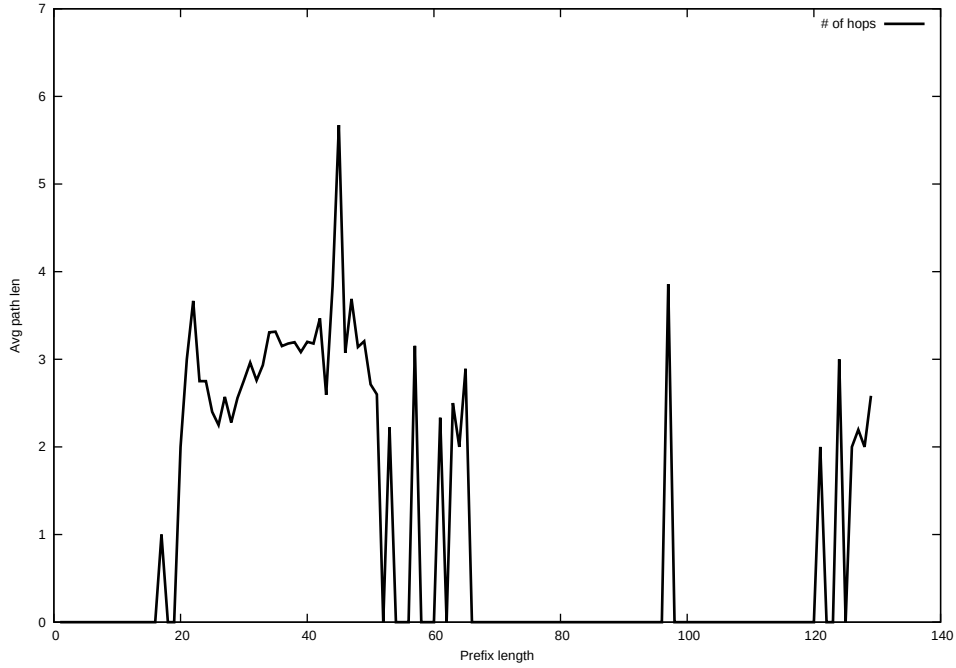
2014-07-16



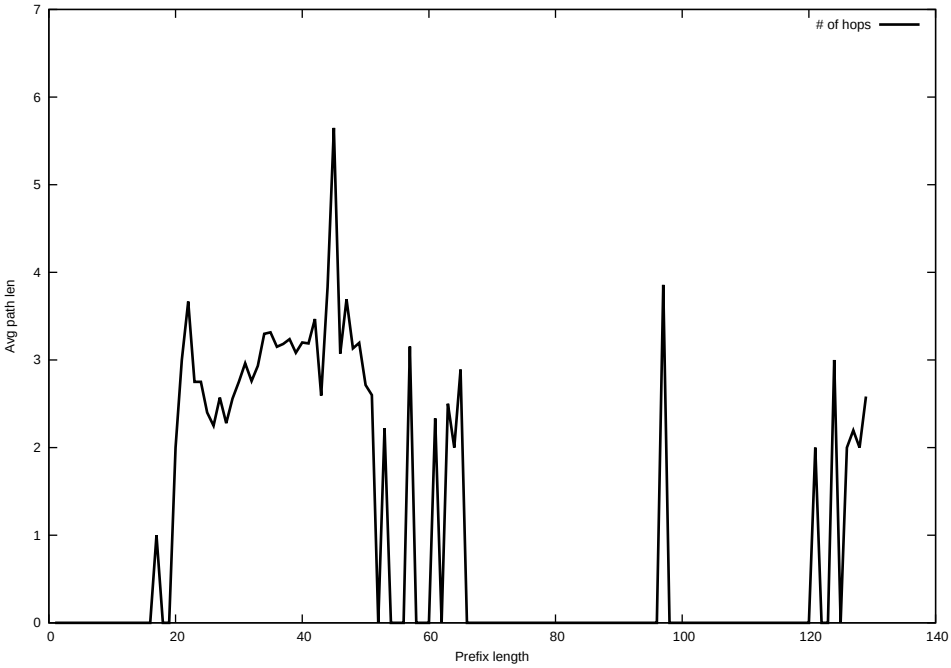
2014-07-17



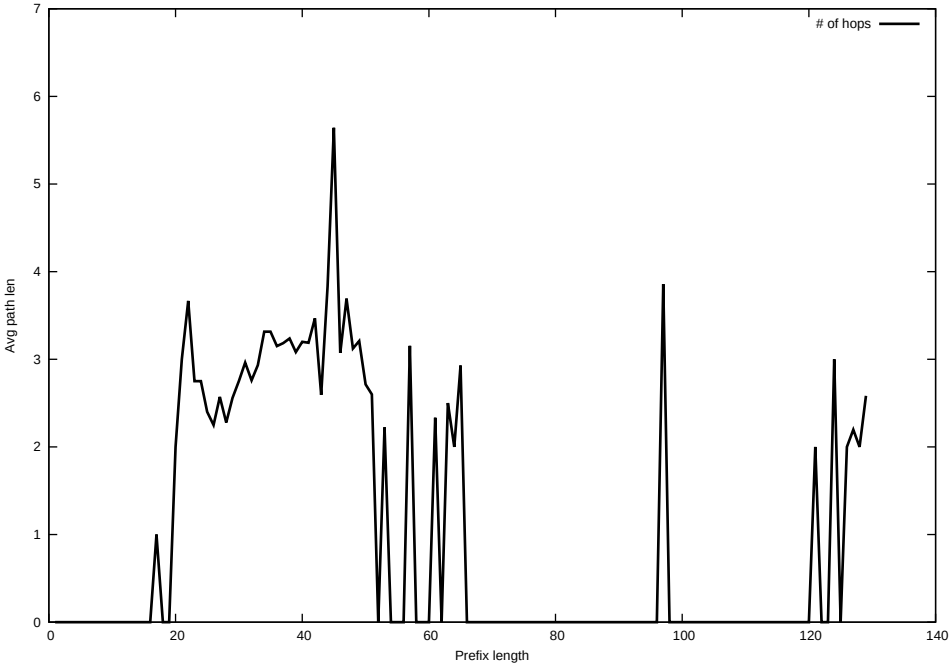
2014-07-18



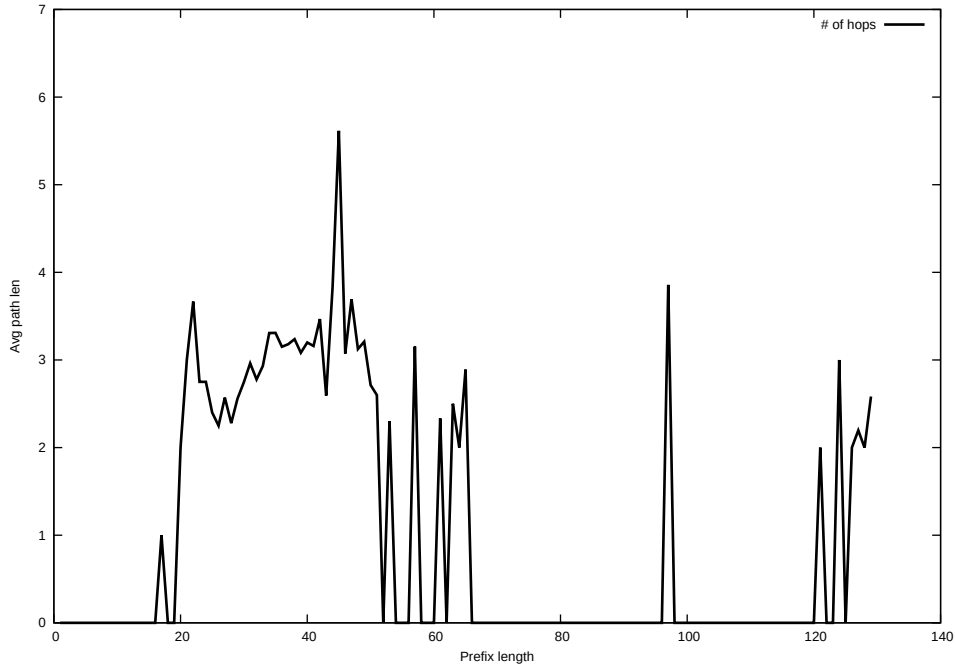
2014-07-19



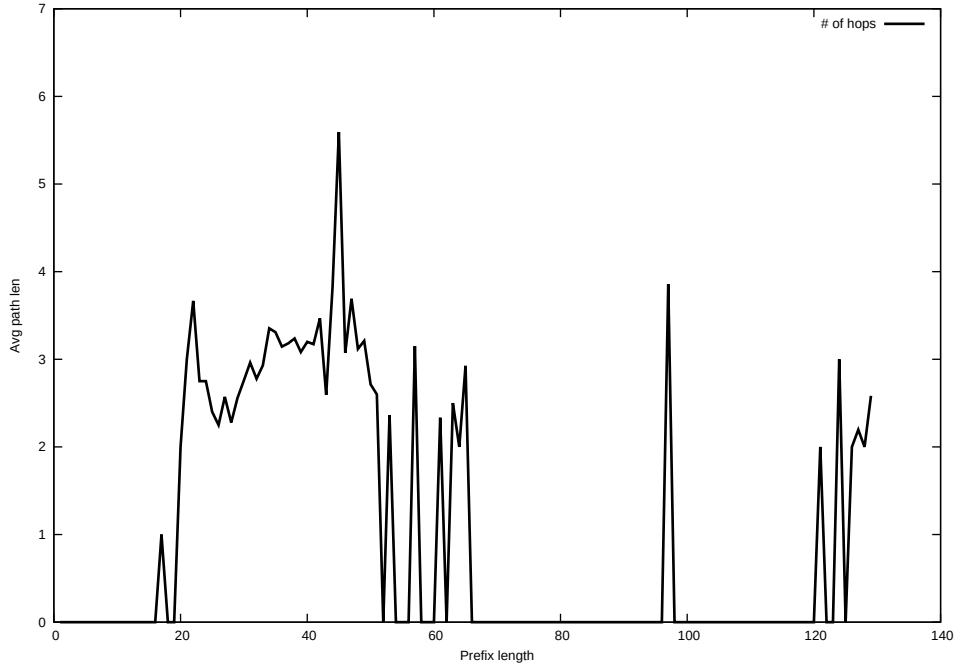
2014-07-20



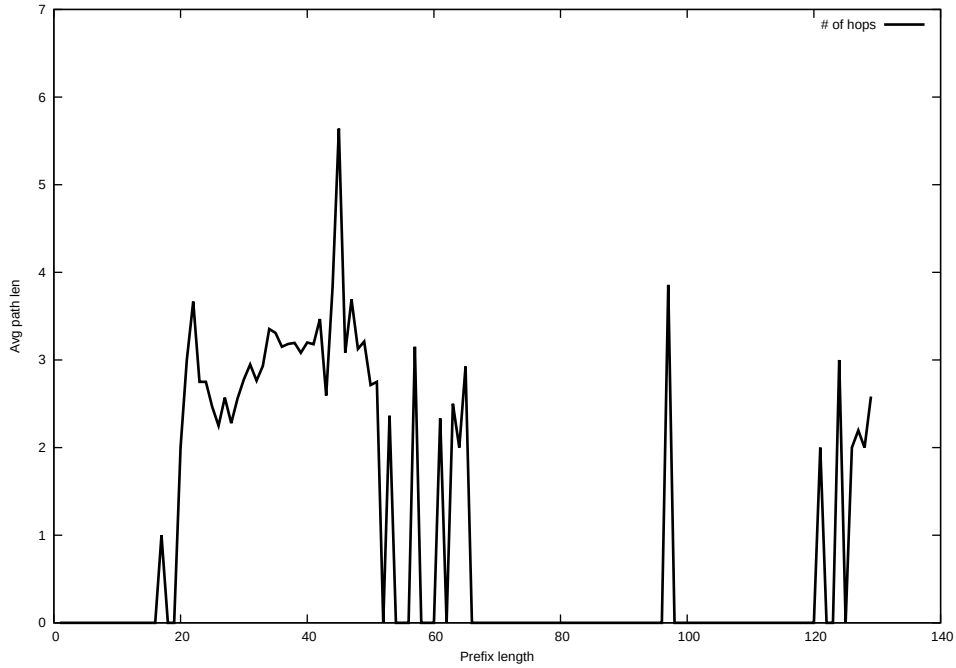
2014-07-21



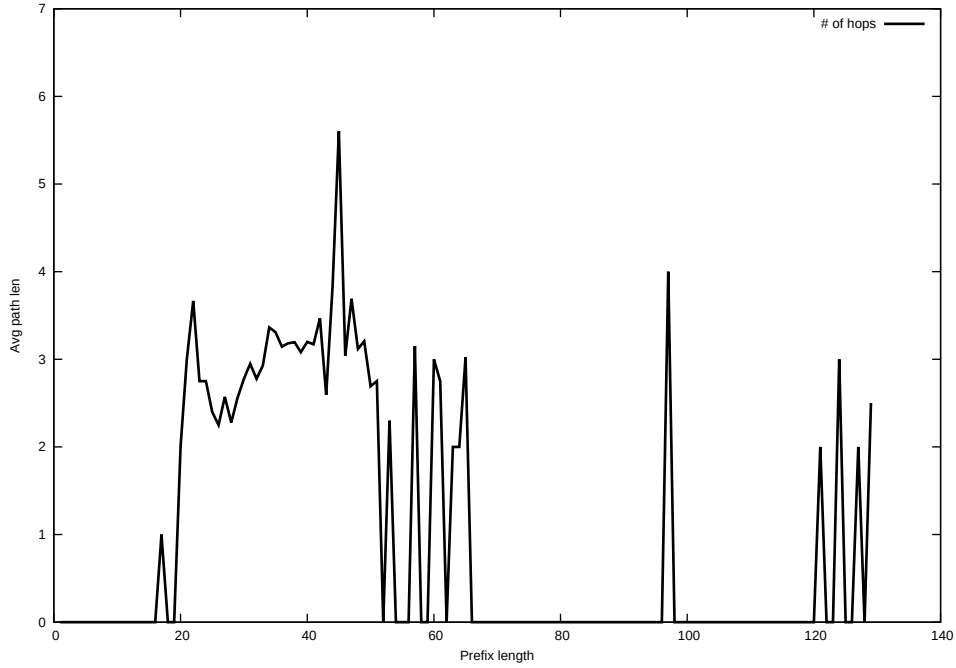
2014-07-22



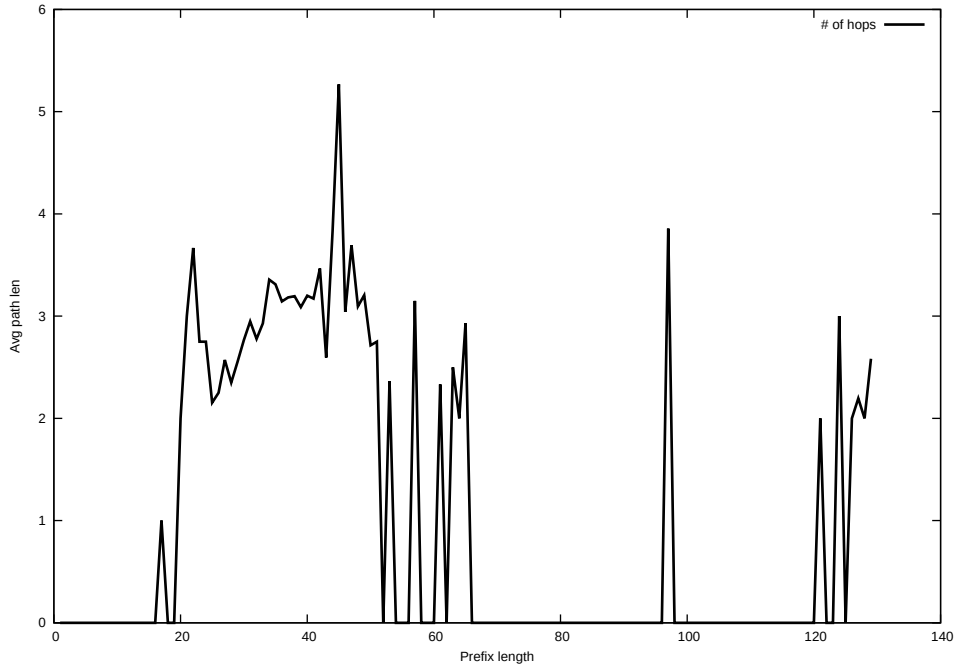
2014-07-23



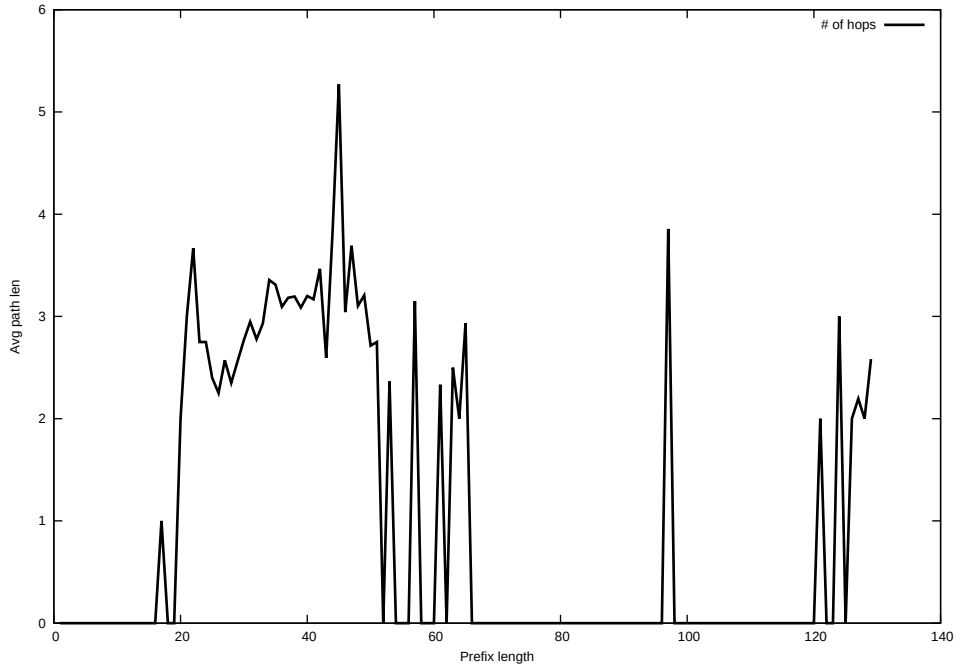
2014-07-24



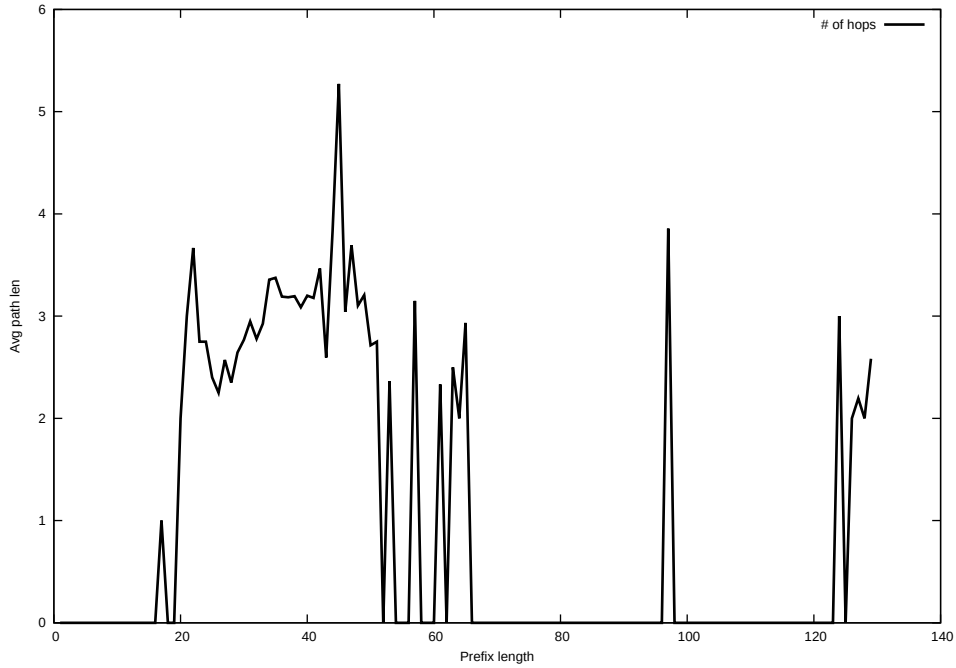
2014-07-25



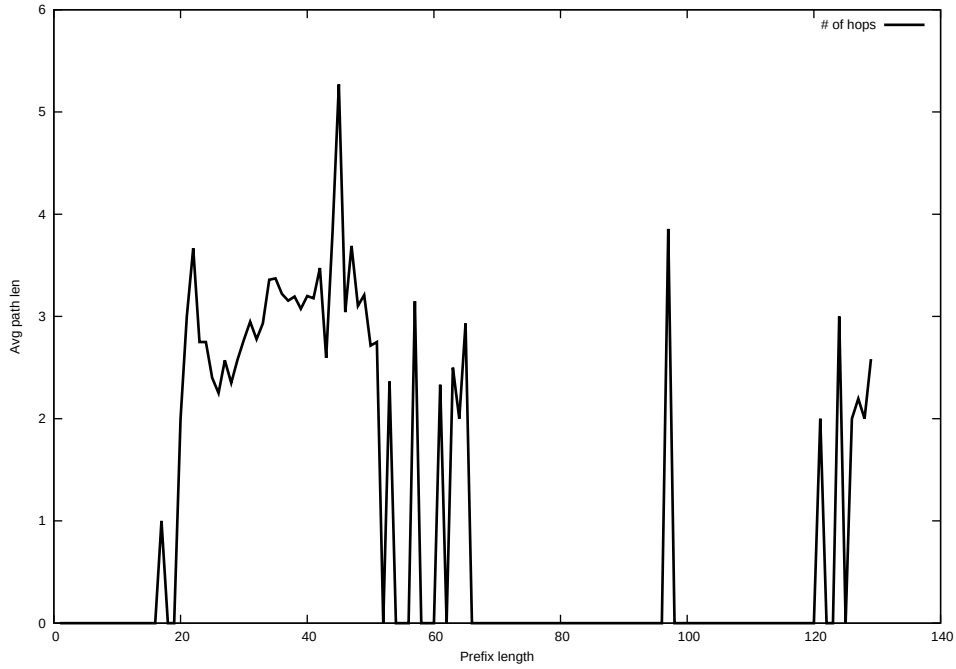
2014-07-26



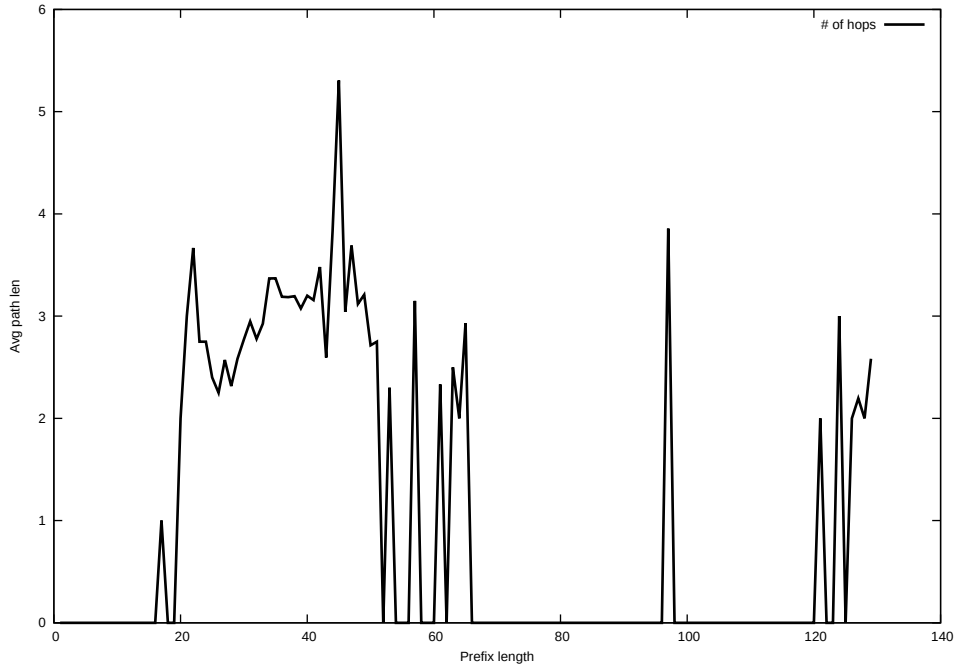
2014-07-27



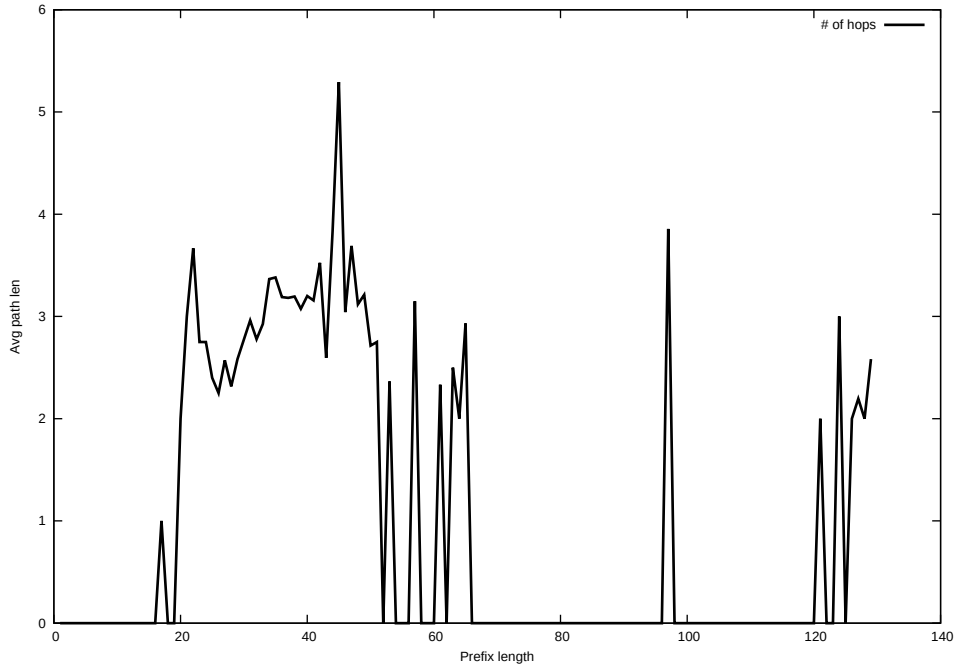
2014-07-28



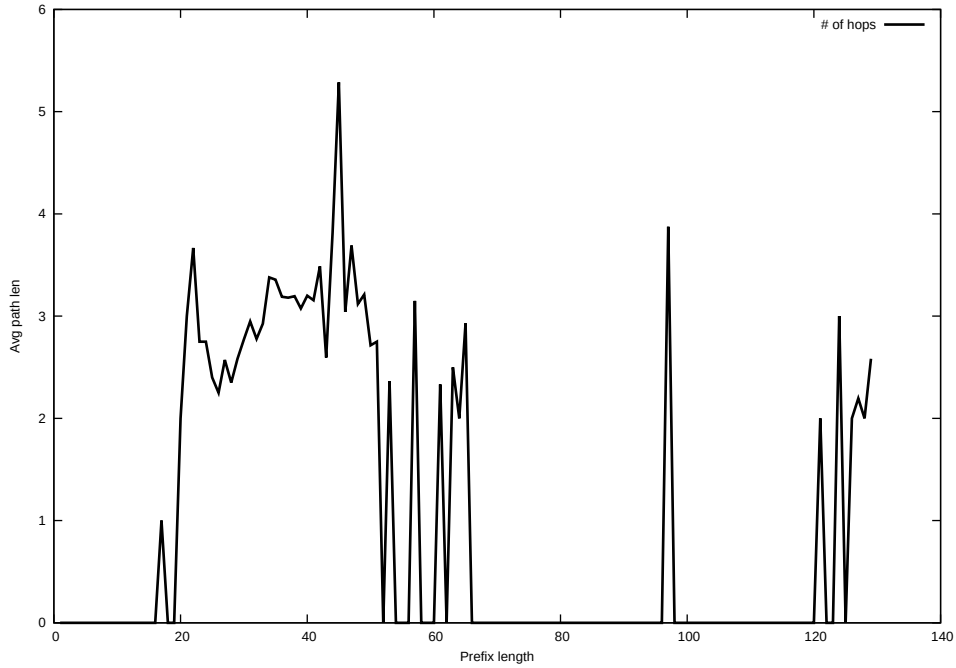
2014-07-29



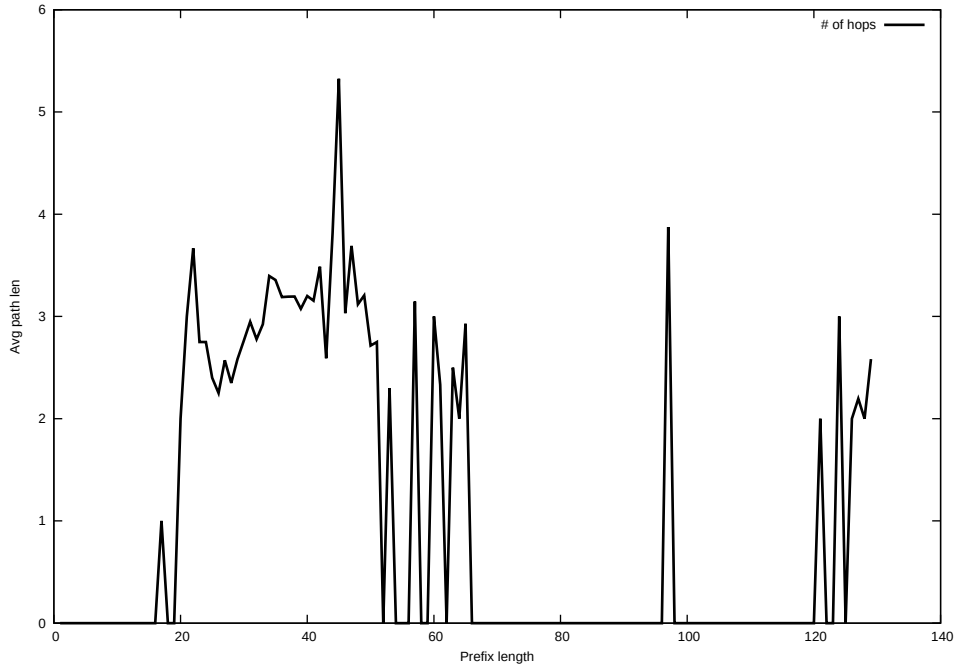
2014-07-30



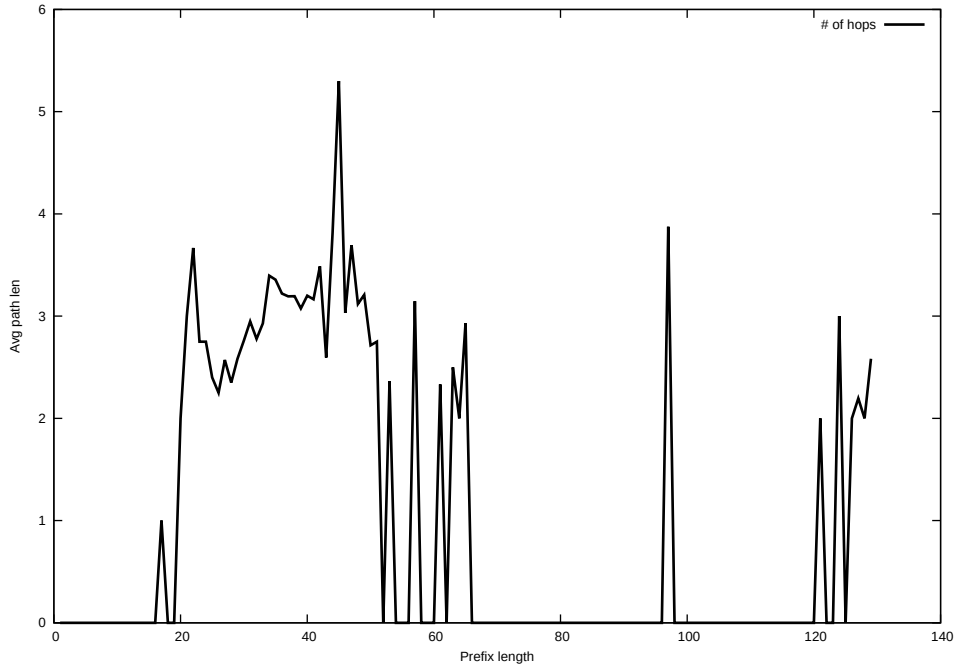
2014-07-31



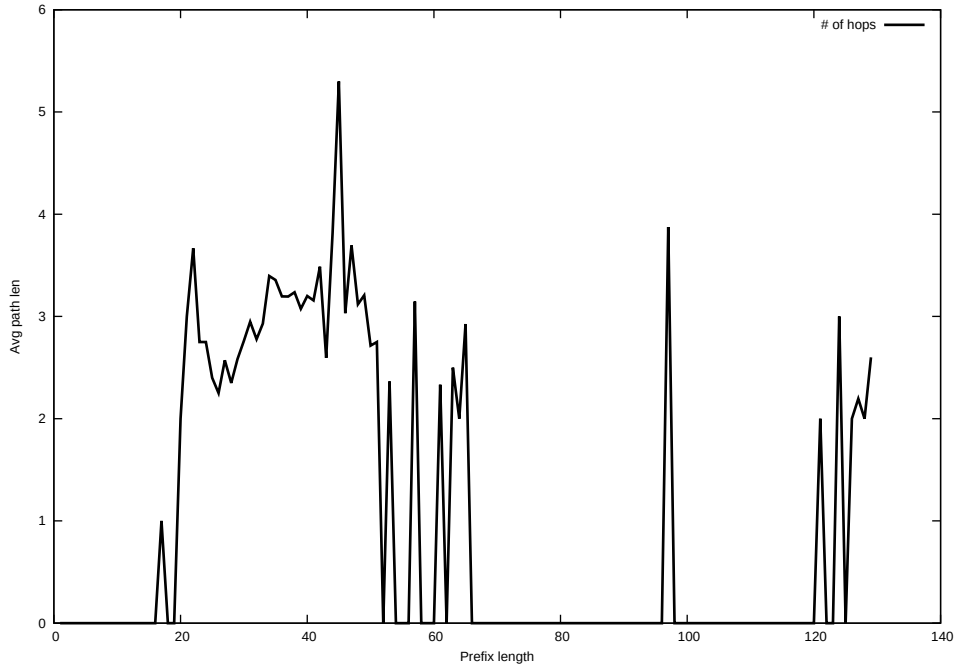
2014-08-01



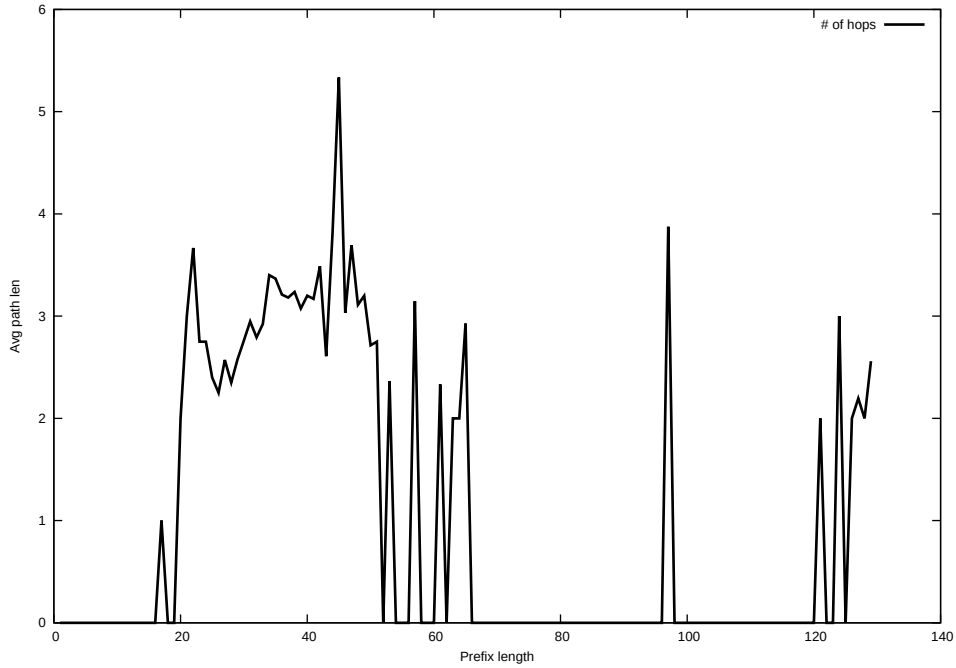
2014-08-02



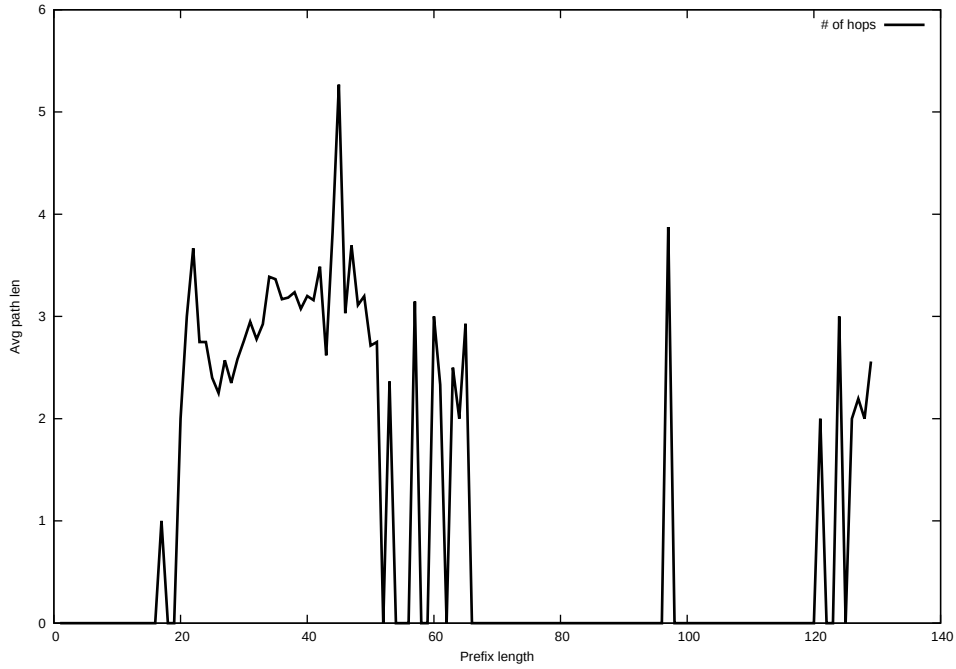
2014-08-03



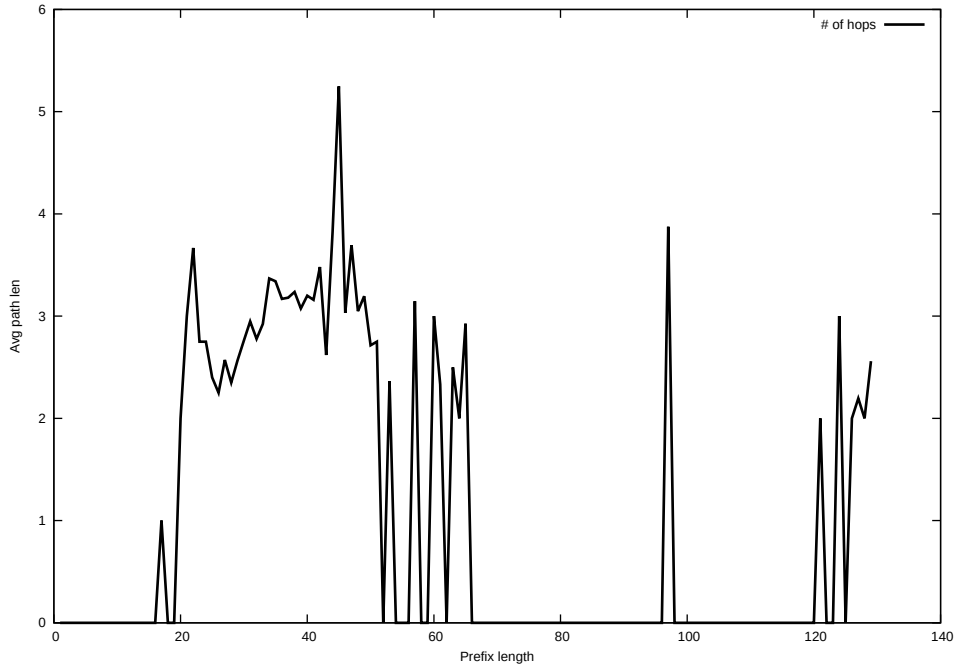
2014-08-04



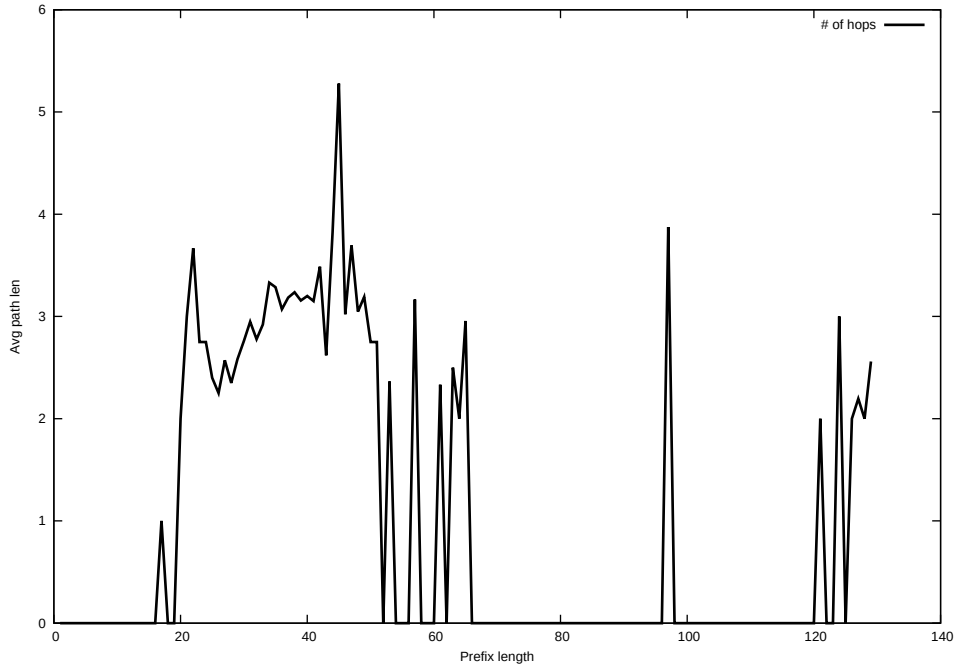
2014-08-05



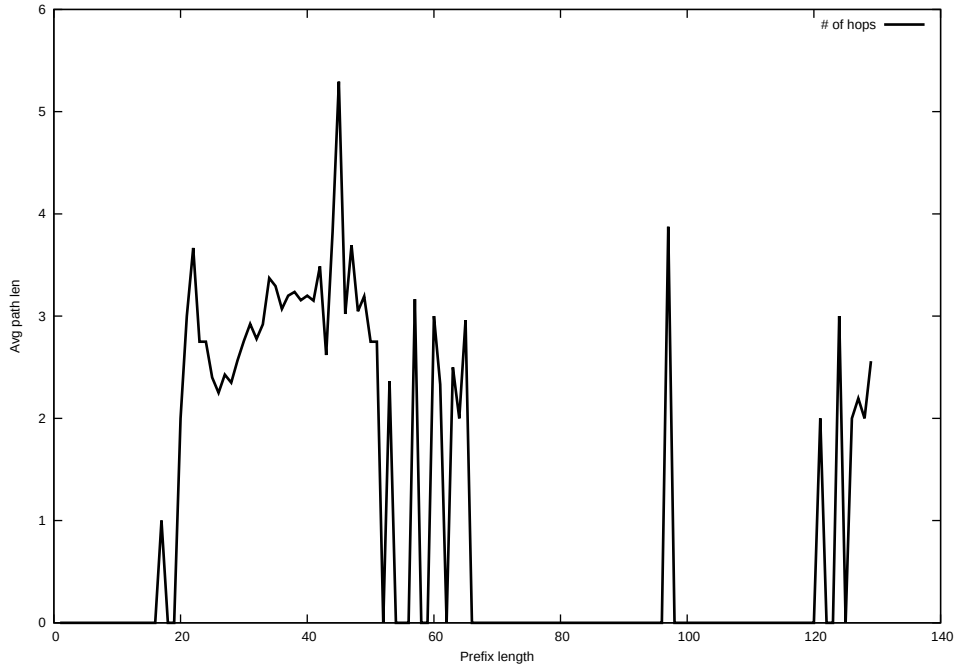
2014-08-06



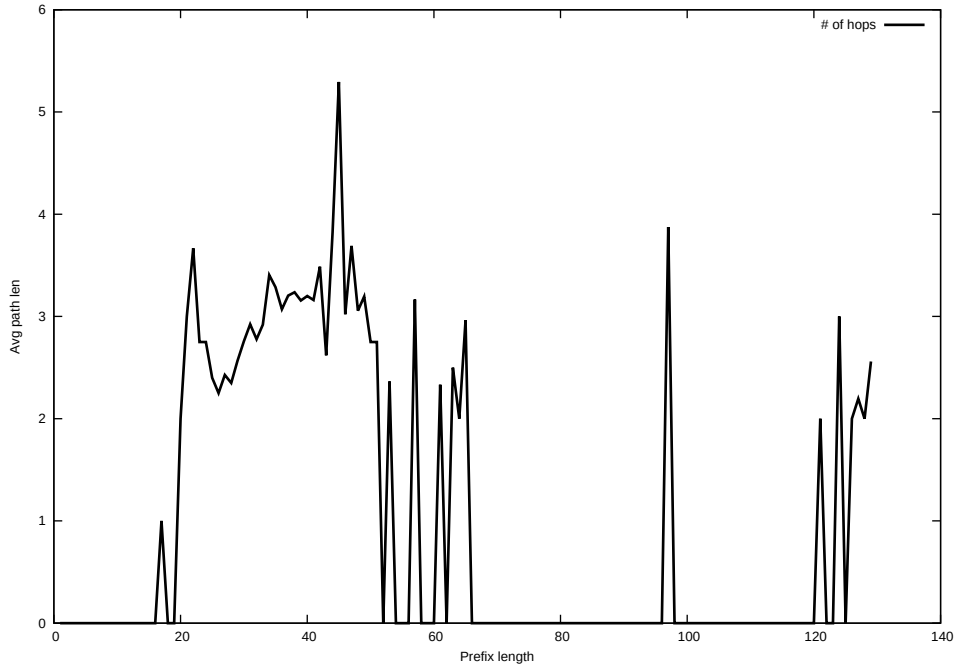
2014-08-07



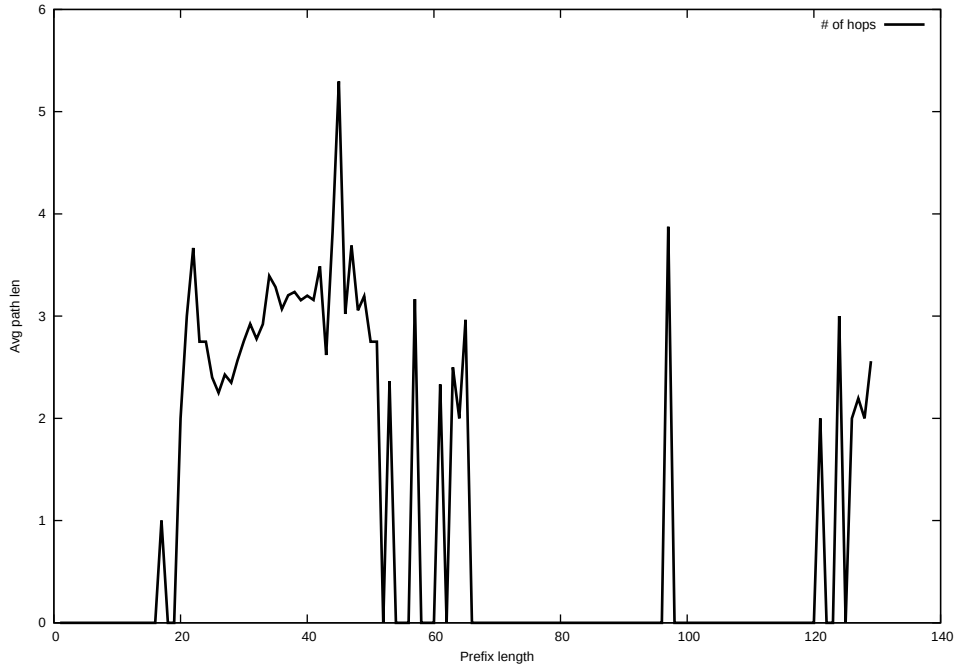
2014-08-08



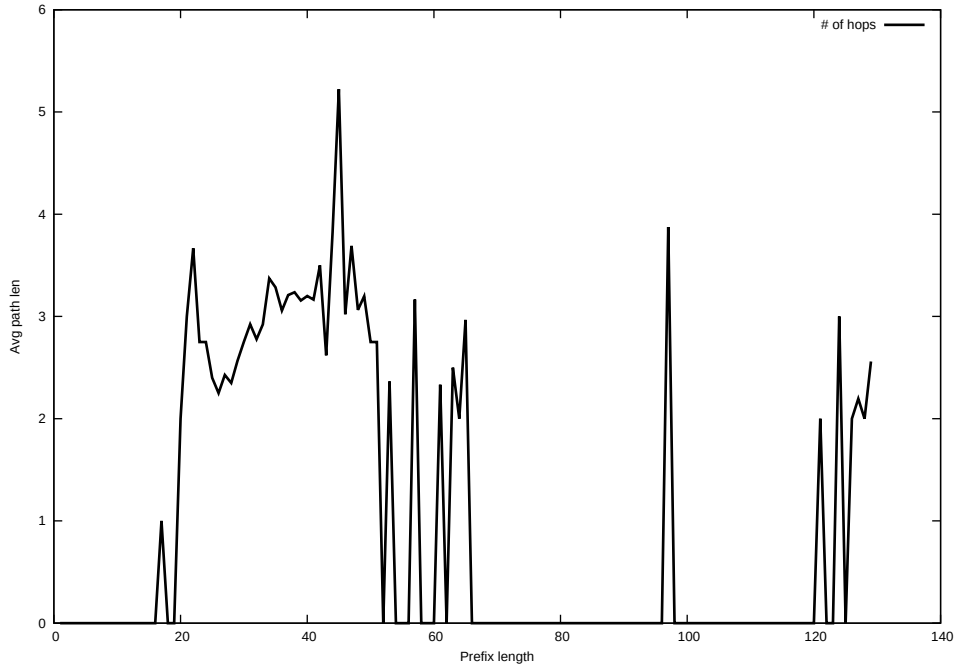
2014-08-09



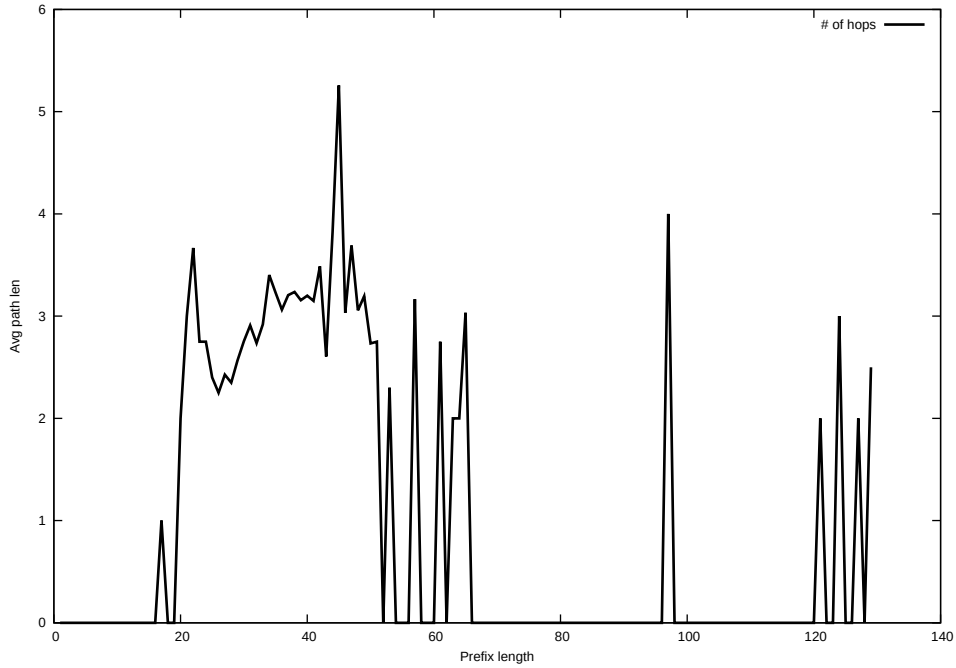
2014-08-10



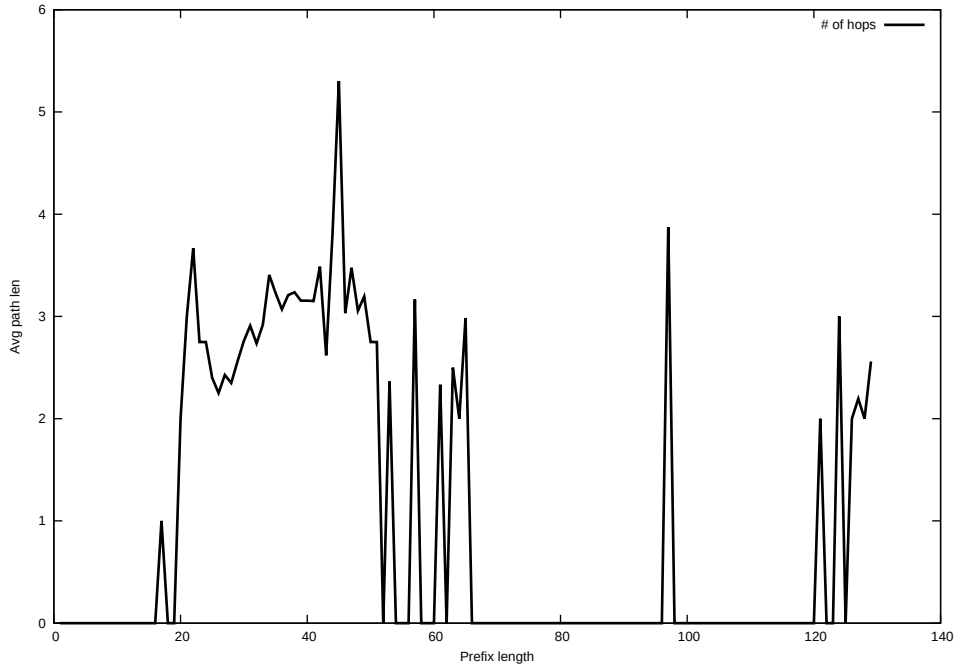
2014-08-11



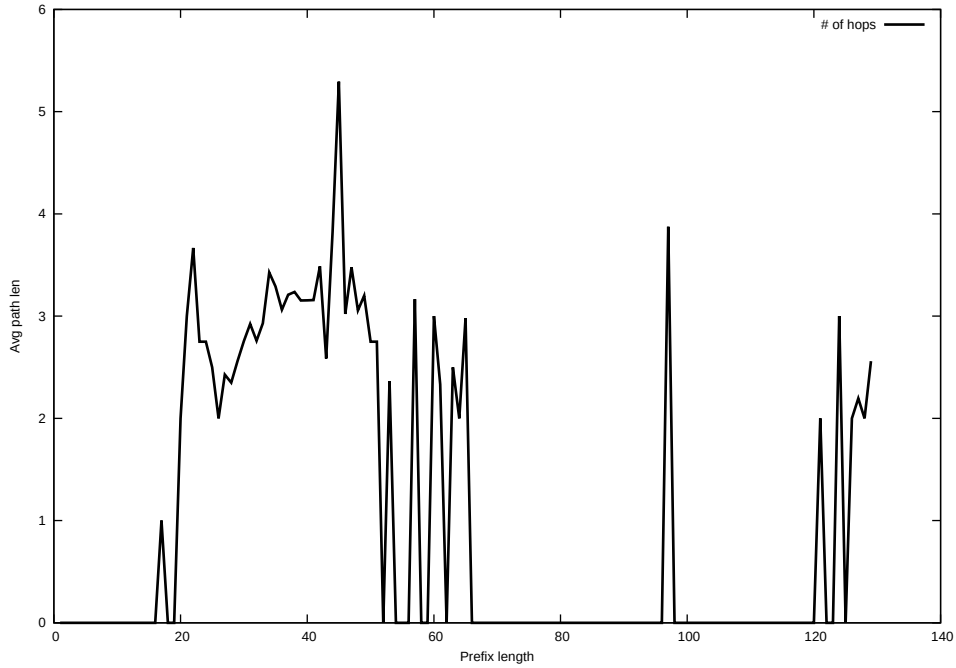
2014-08-12



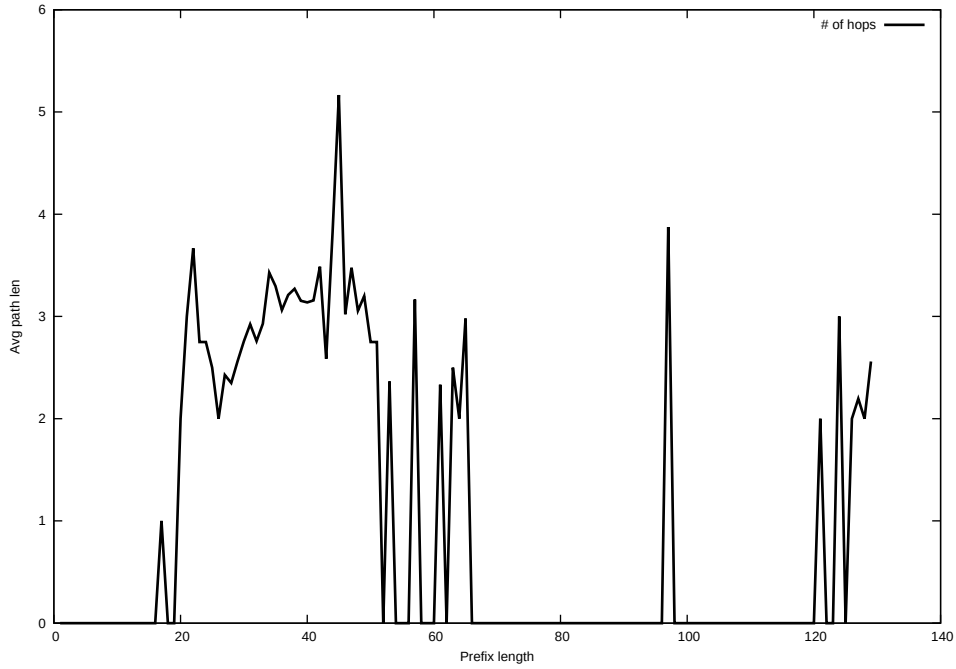
2014-08-13



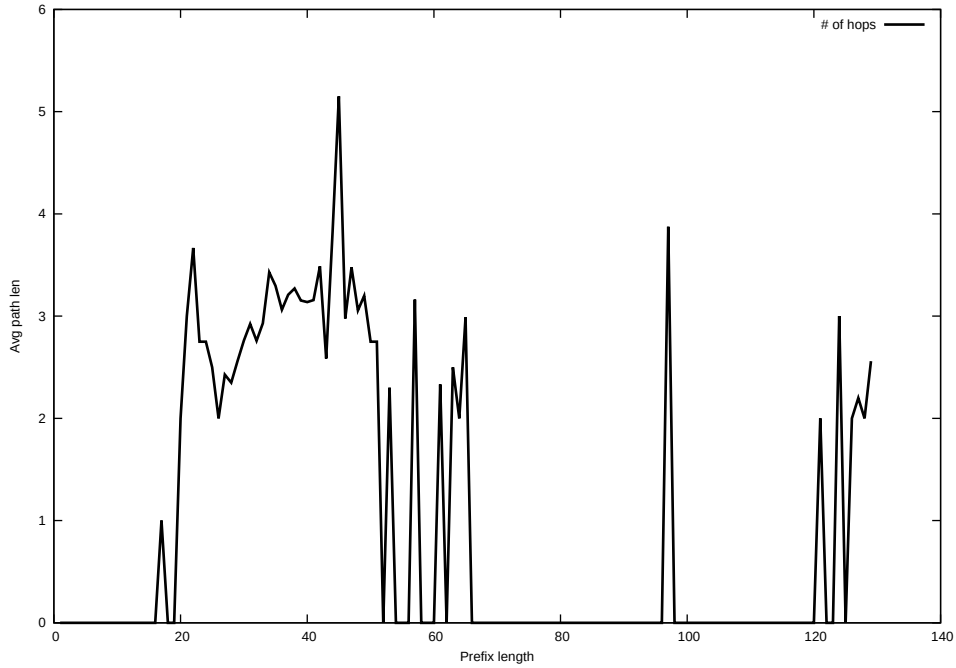
2014-08-14



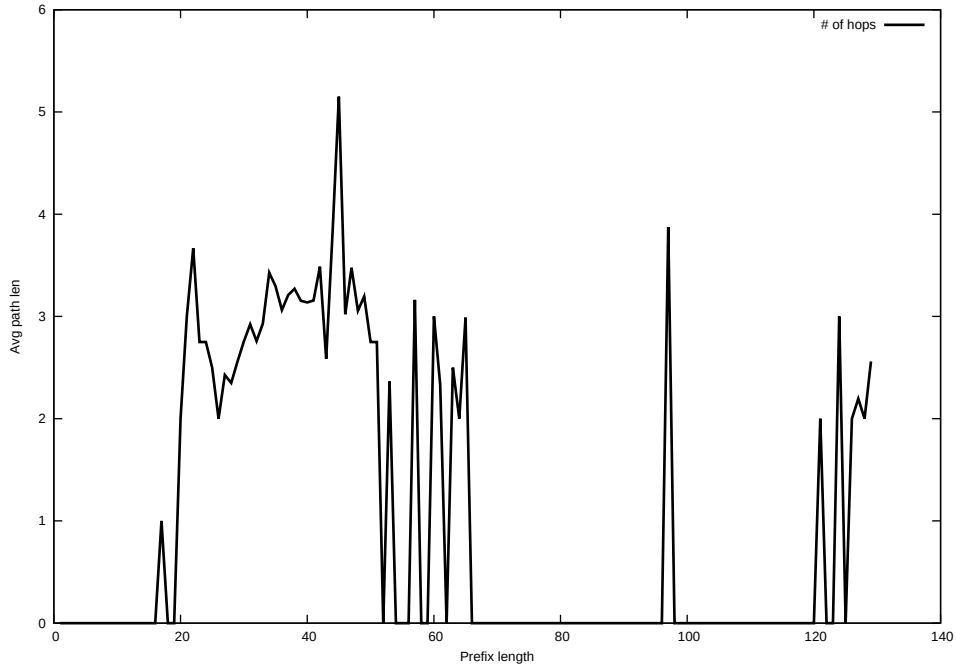
2014-08-15



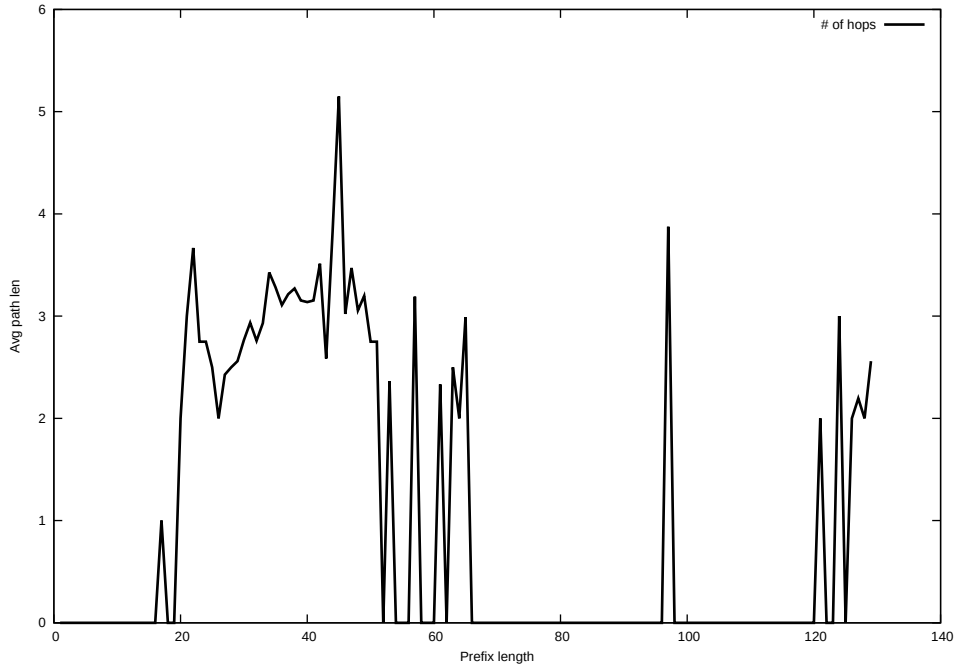
2014-08-16



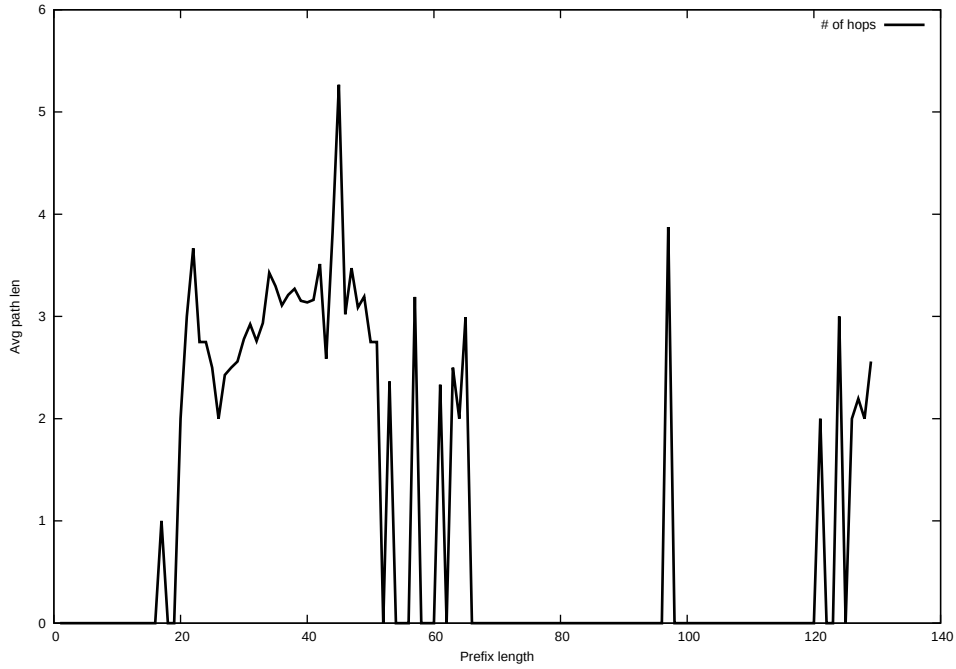
2014-08-17



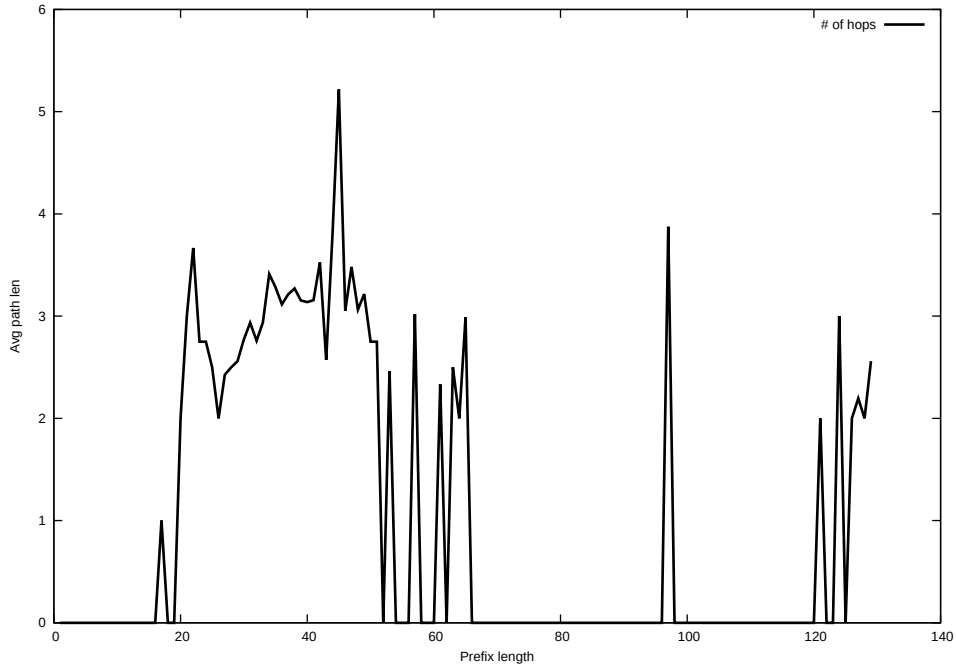
2014-08-18



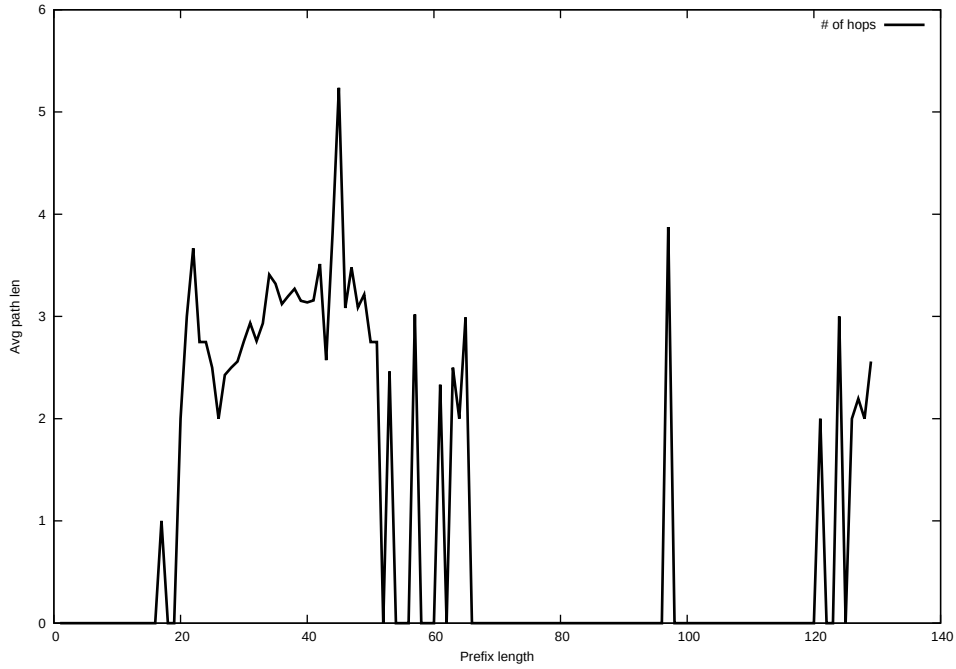
2014-08-19



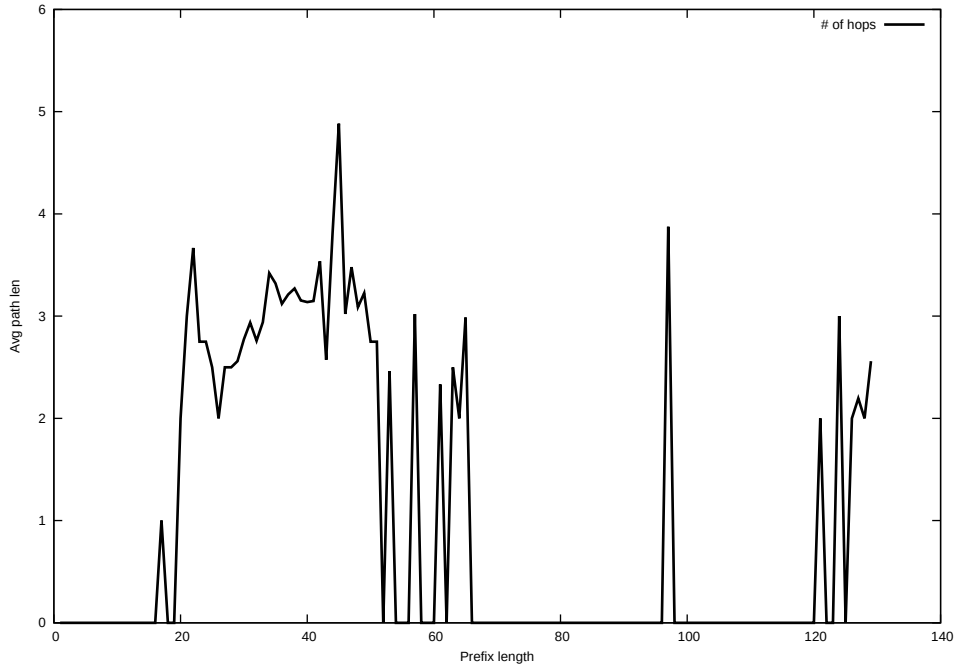
2014-08-20



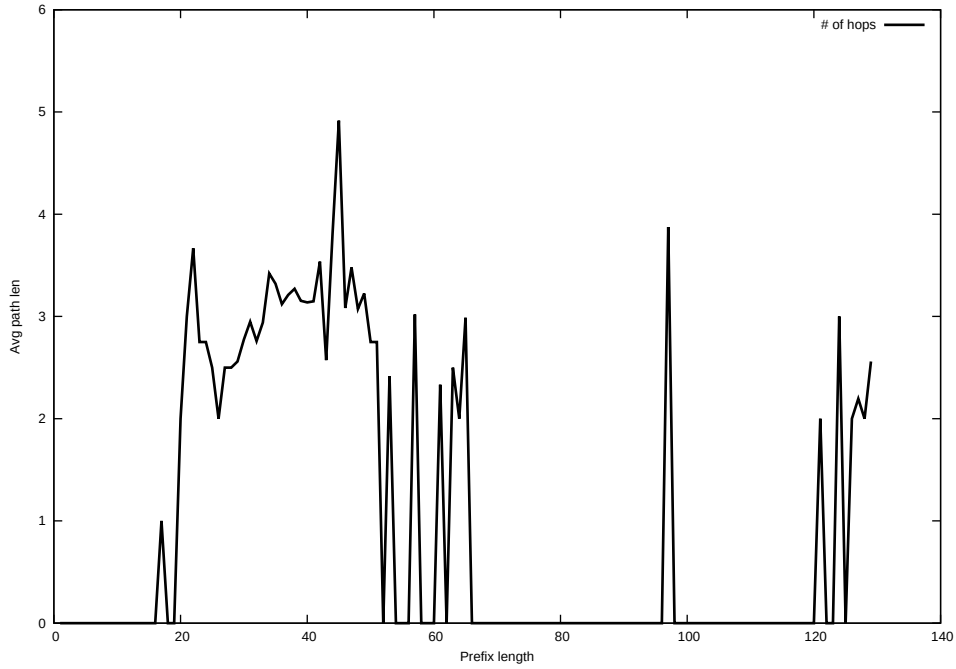
2014-08-21



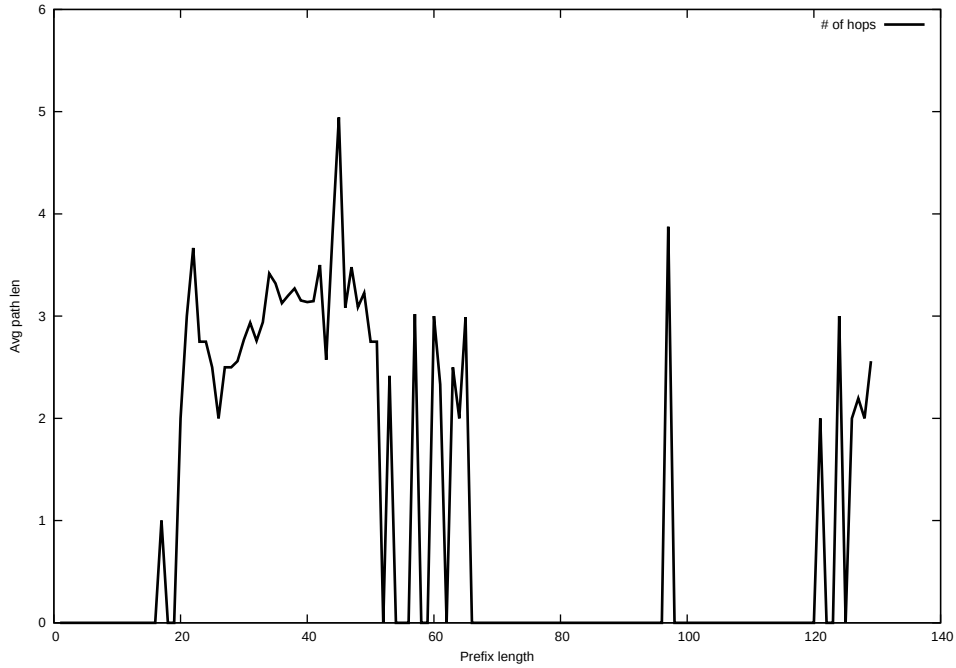
2014-08-22



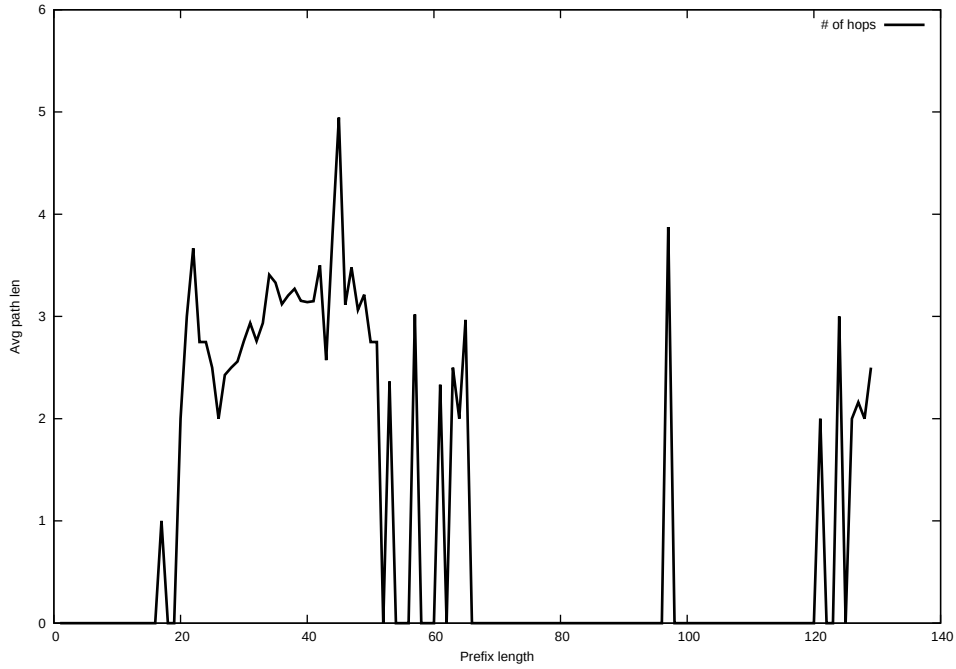
2014-08-23



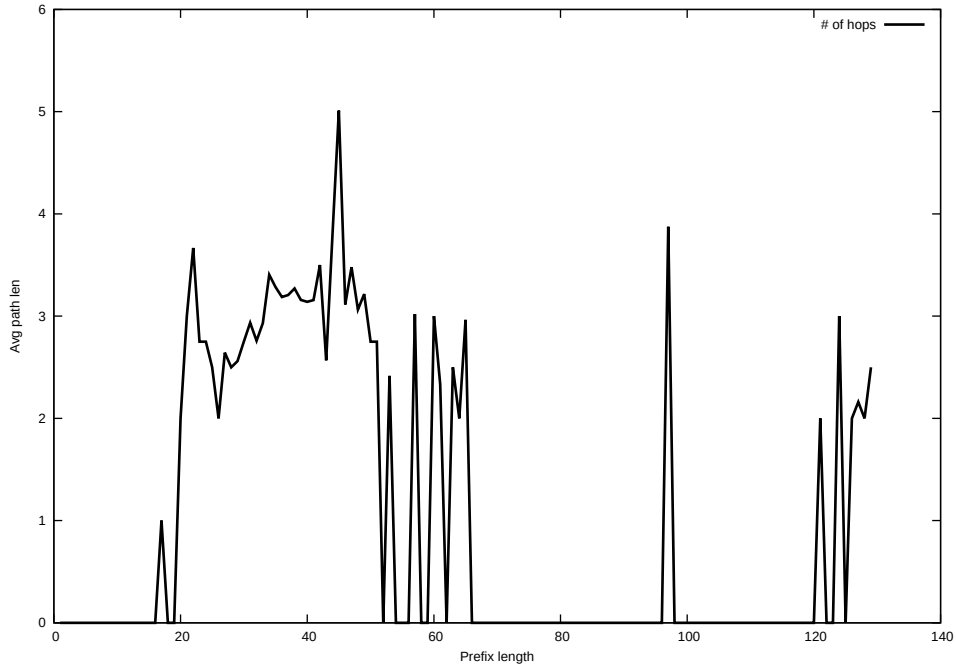
2014-08-24



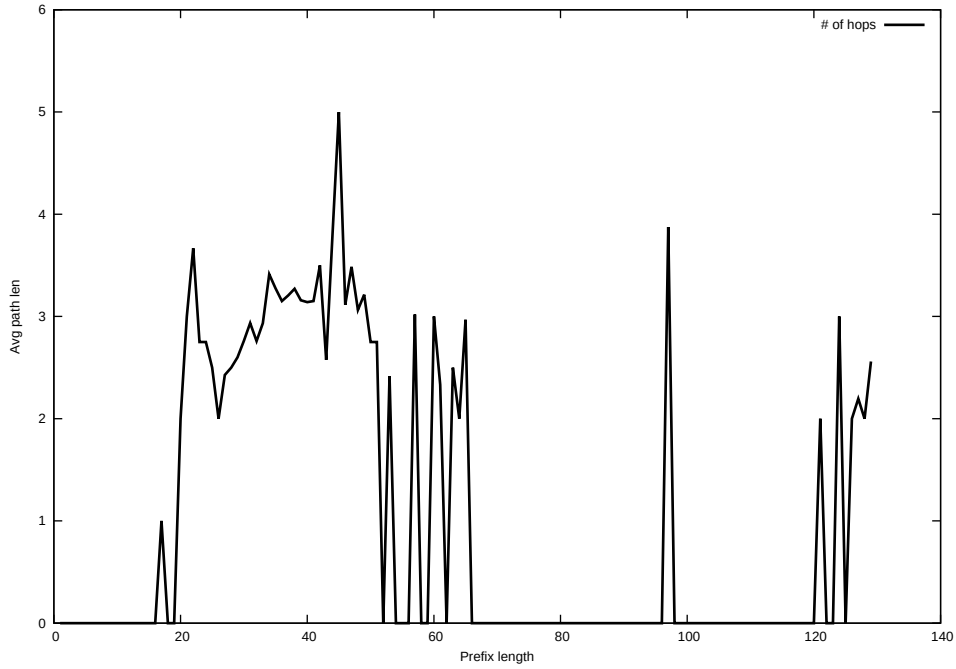
2014-08-25



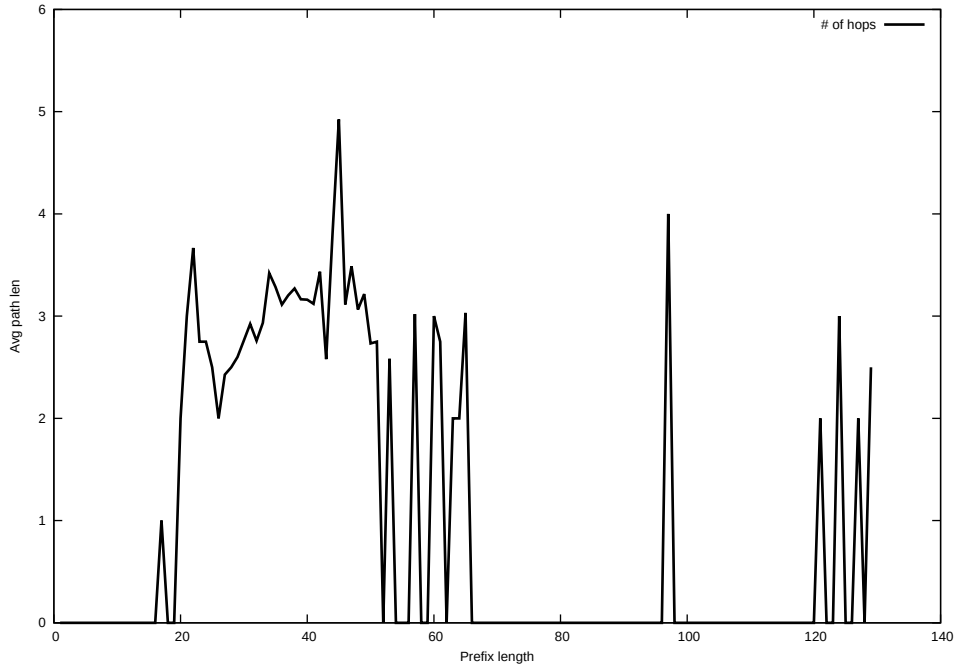
2014-08-26



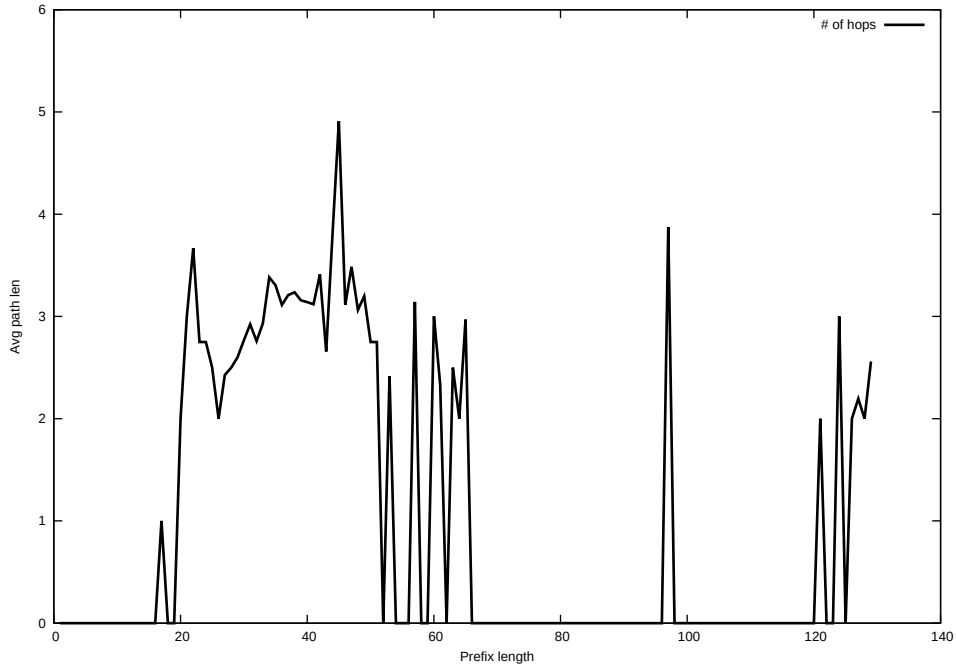
2014-08-27



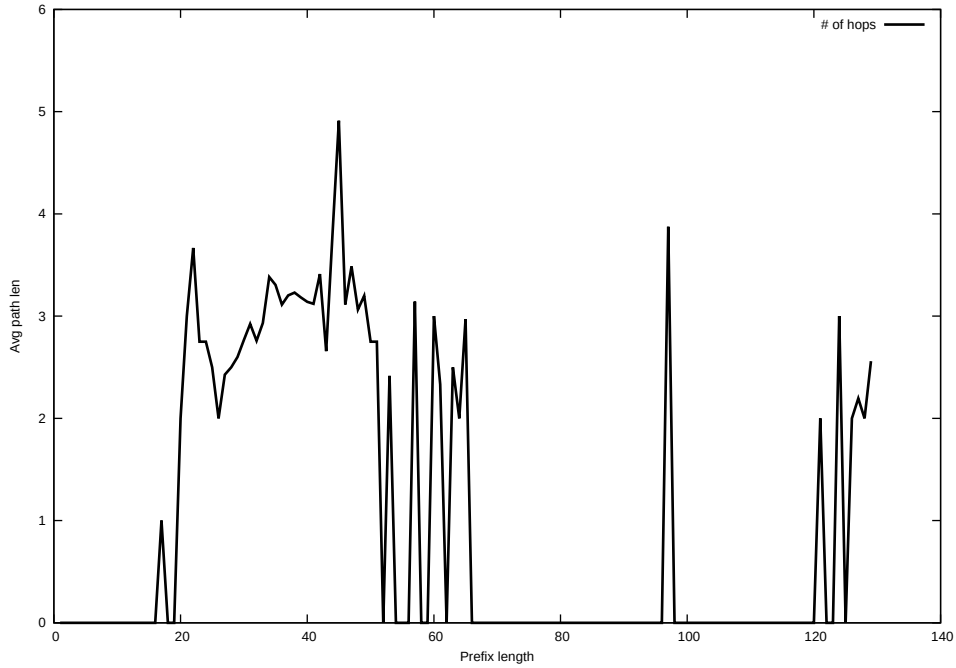
2014-08-28



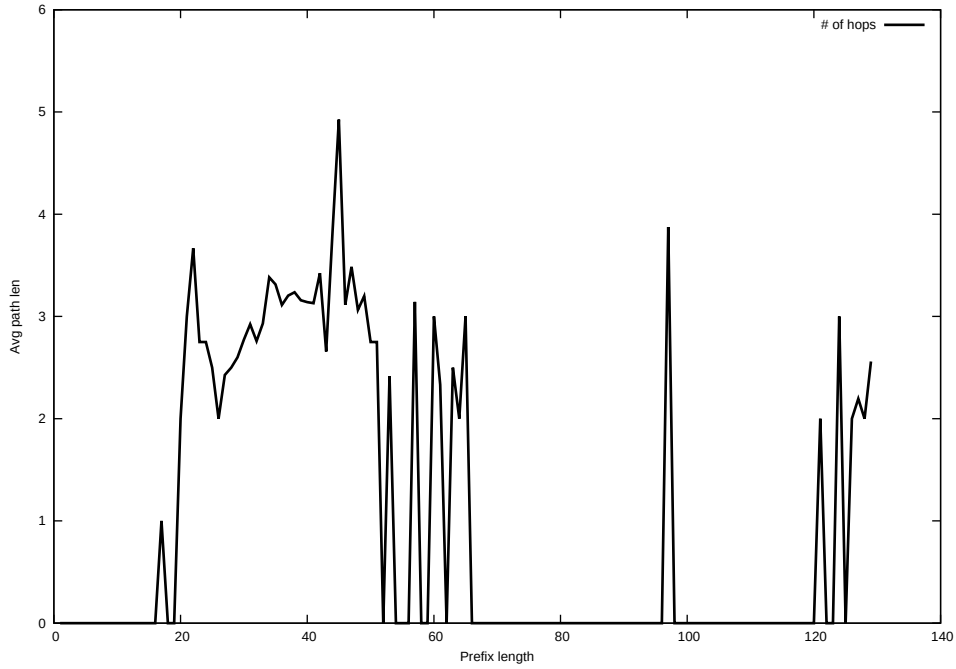
2014-08-29



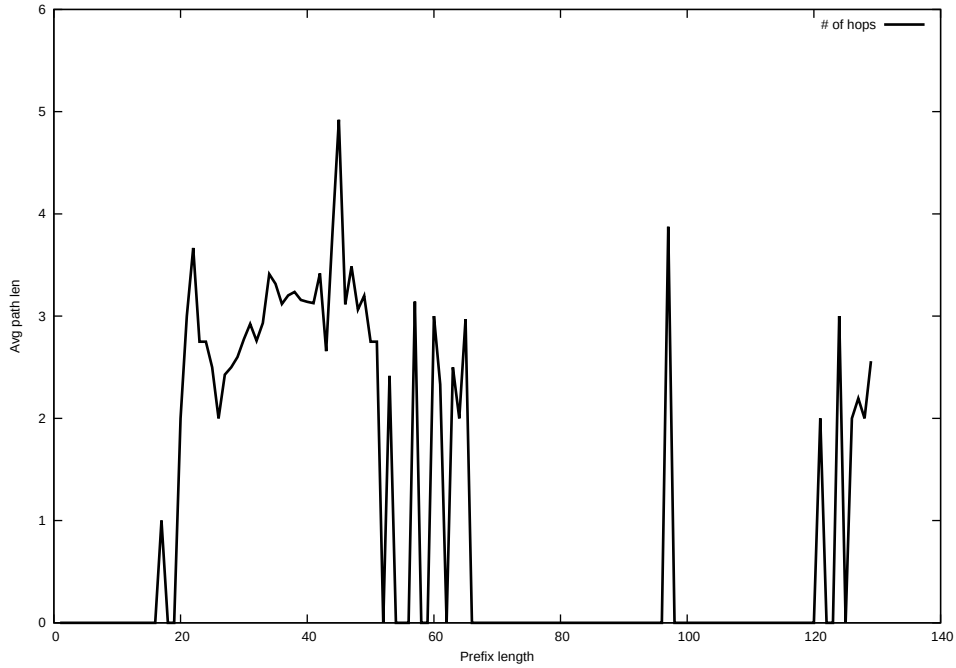
2014-08-30



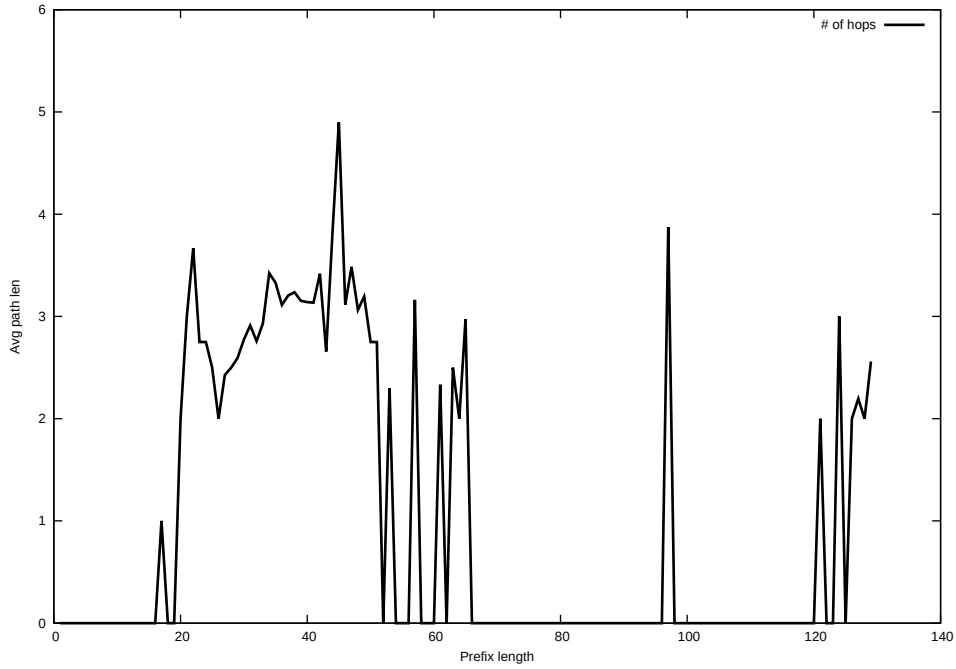
2014-08-31



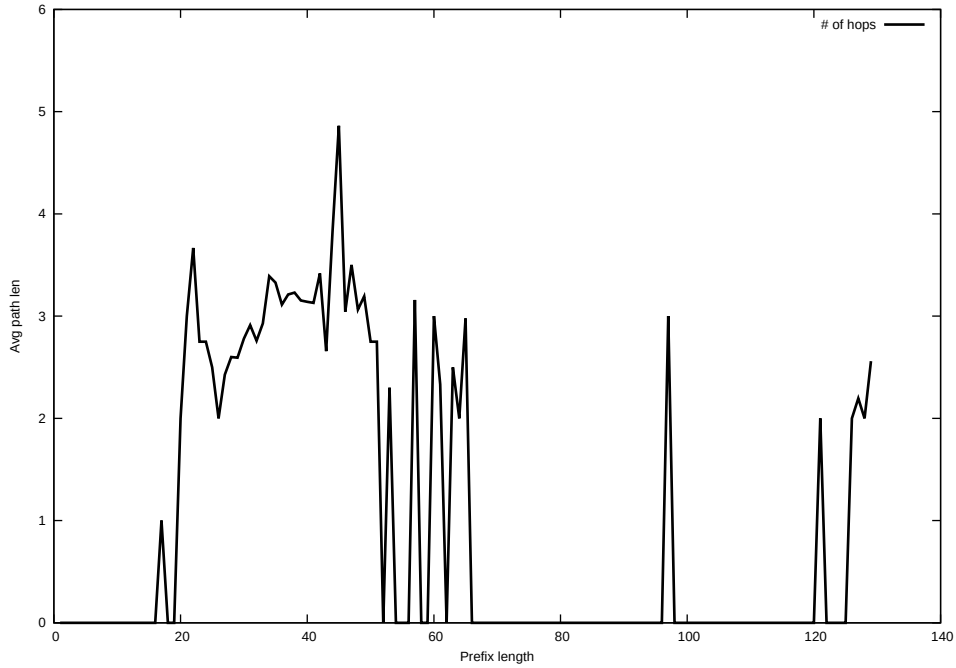
2014-09-01



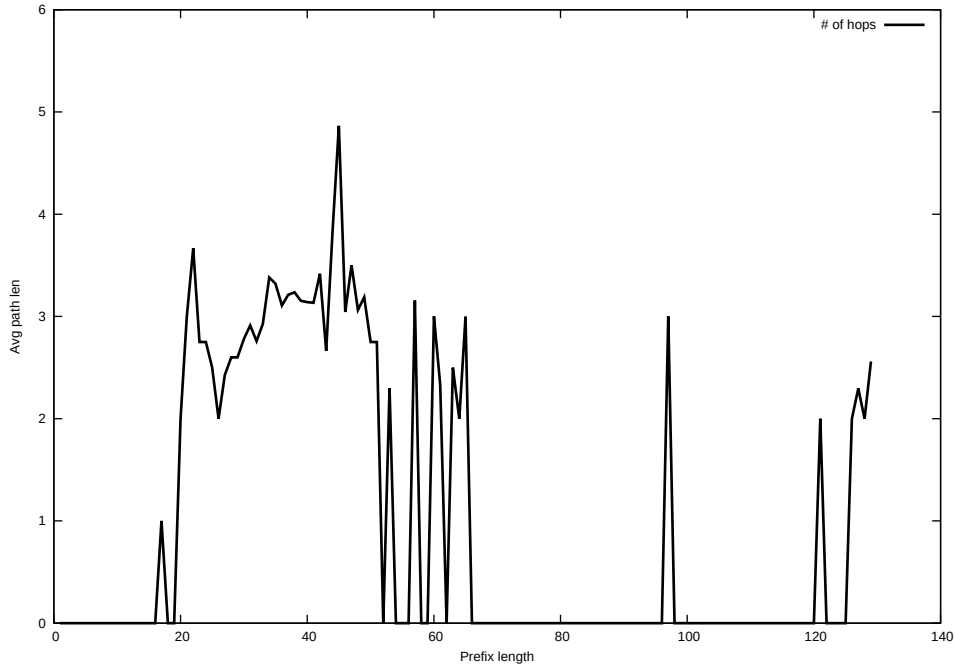
2014-09-02



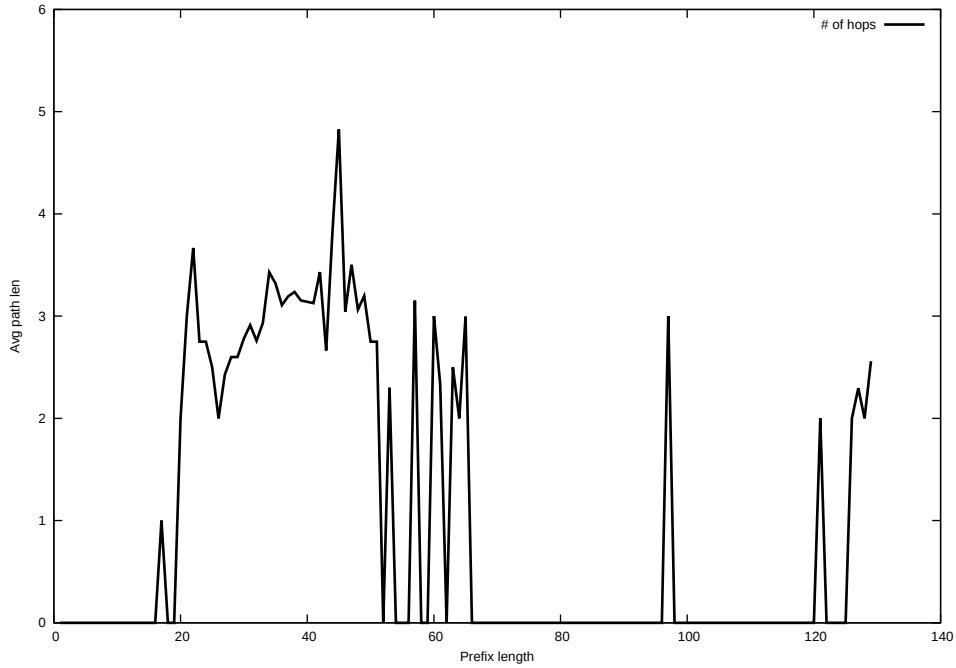
2014-09-03



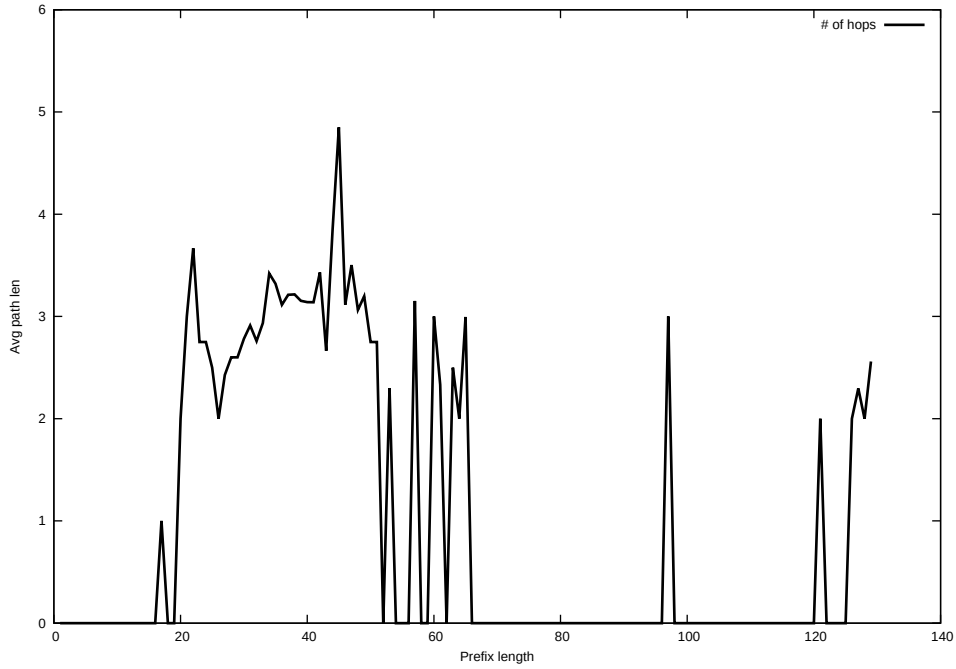
2014-09-04



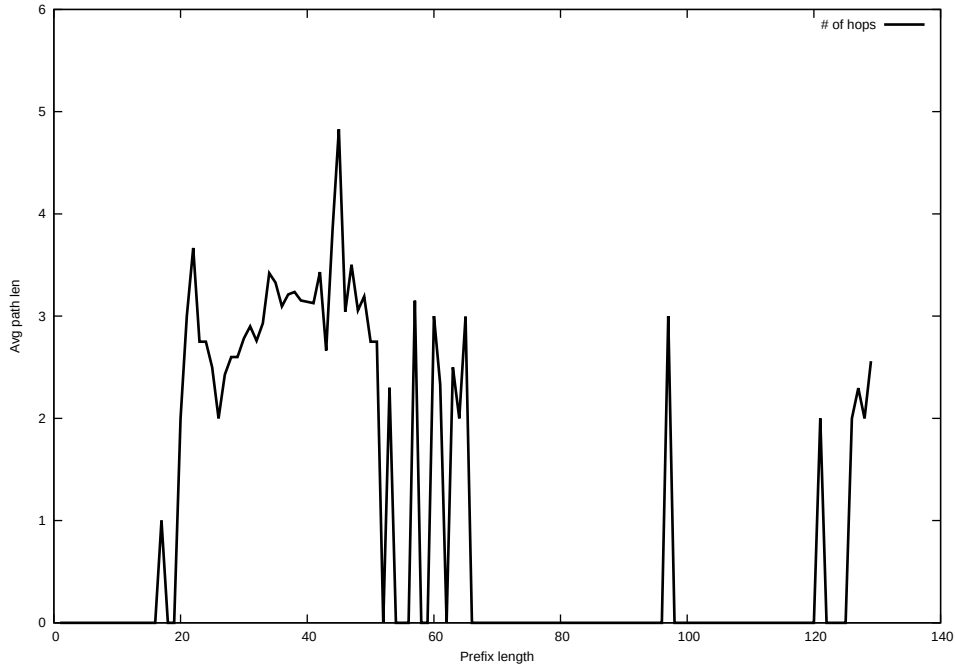
2014-09-05



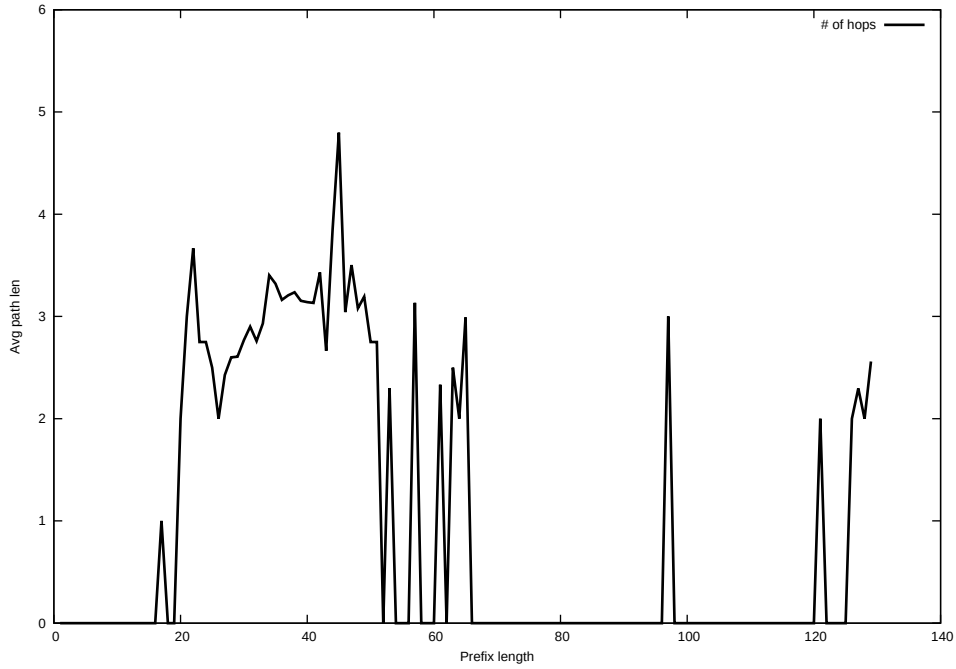
2014-09-06



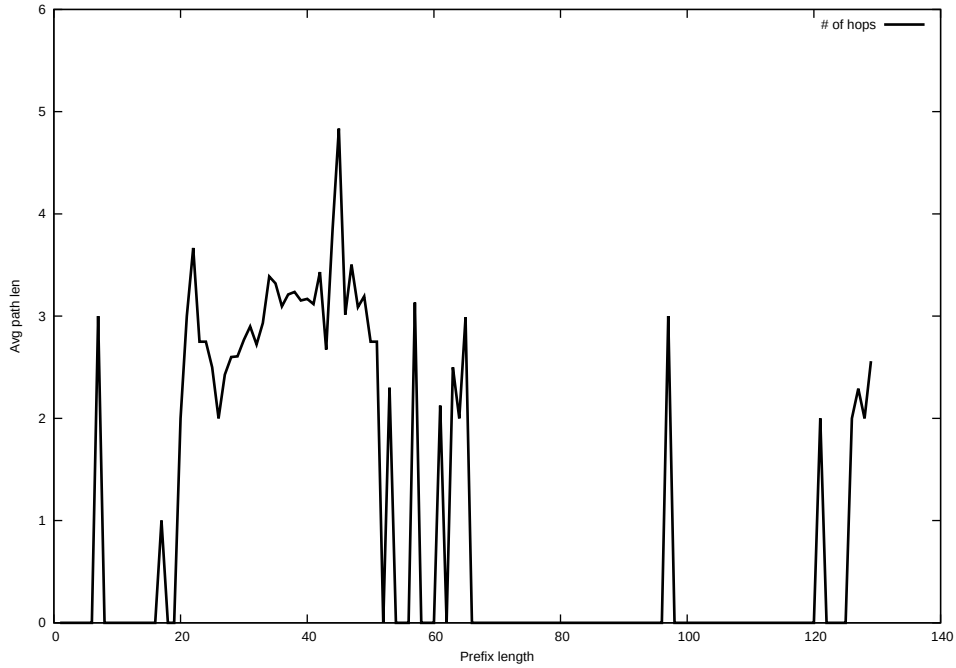
2014-09-07



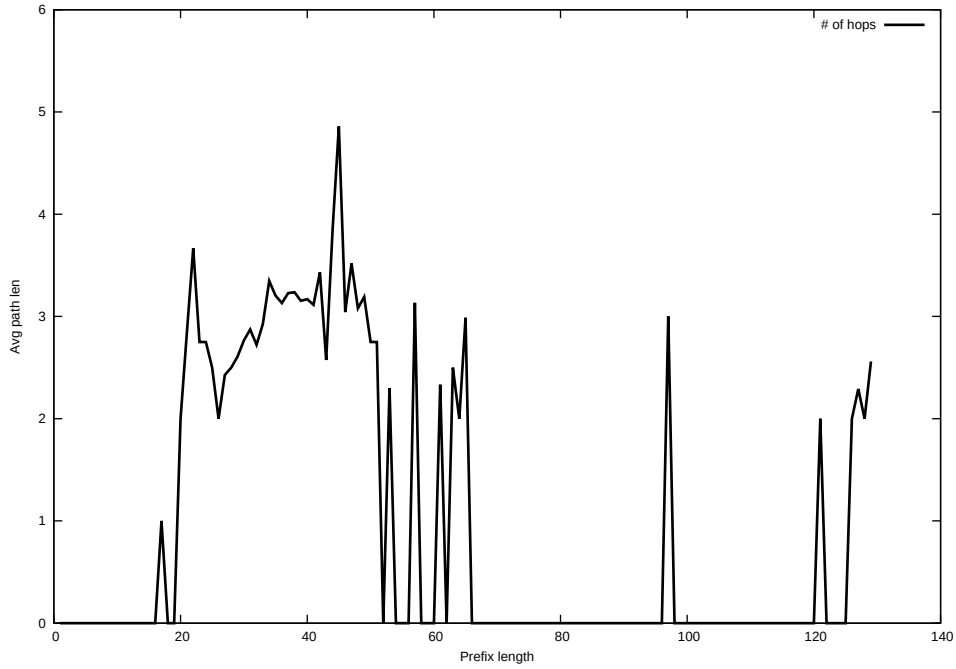
2014-09-08



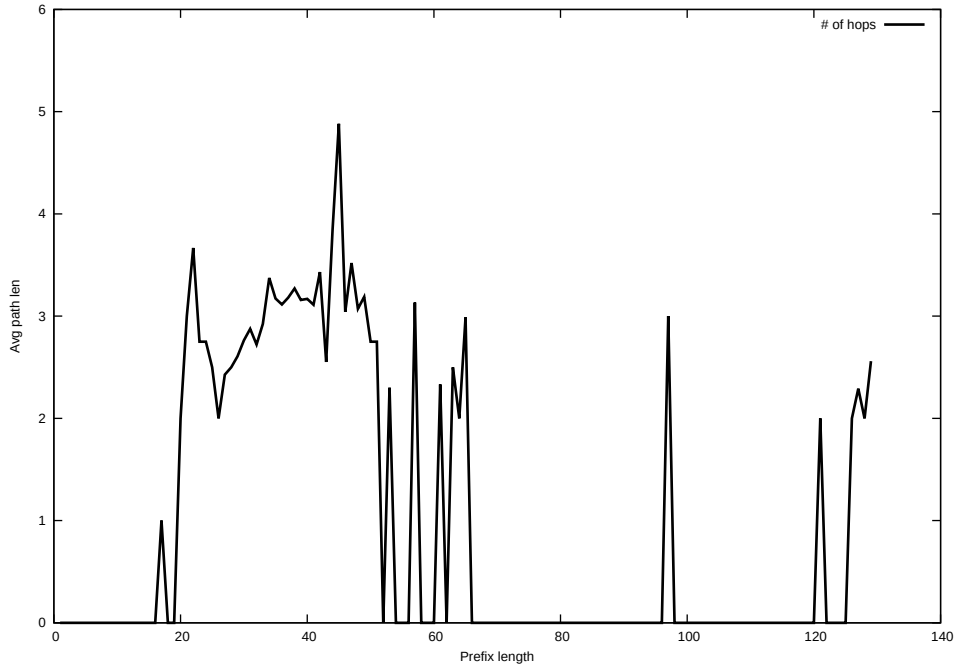
2014-09-09



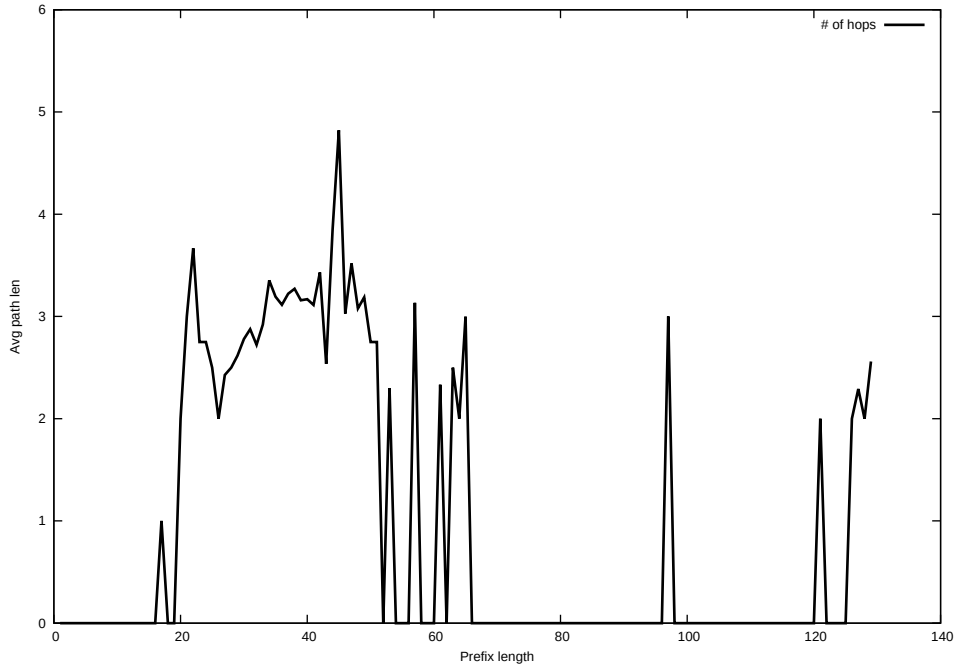
2014-09-10



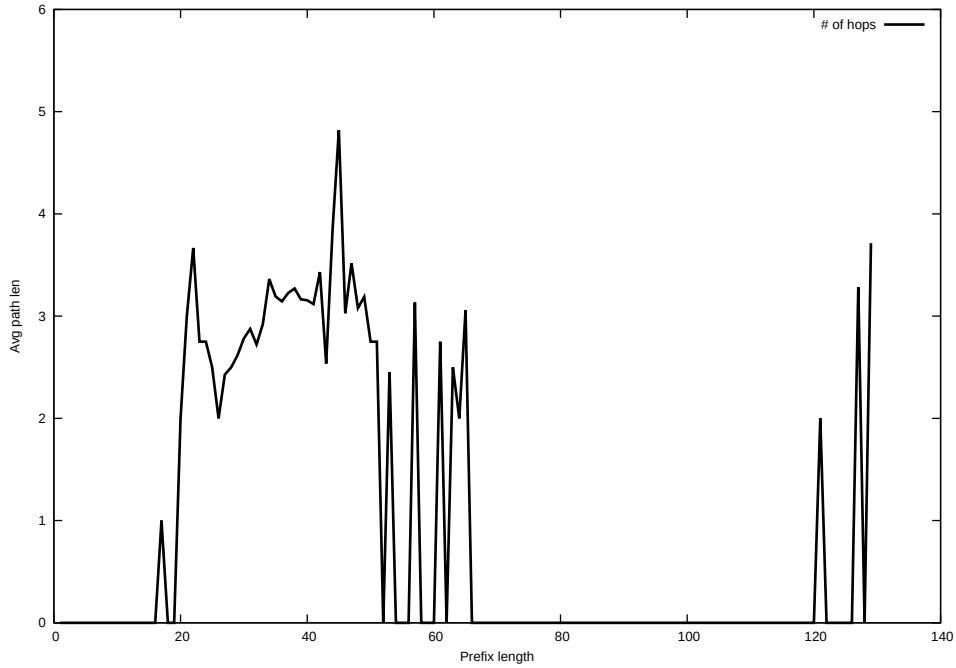
2014-09-11



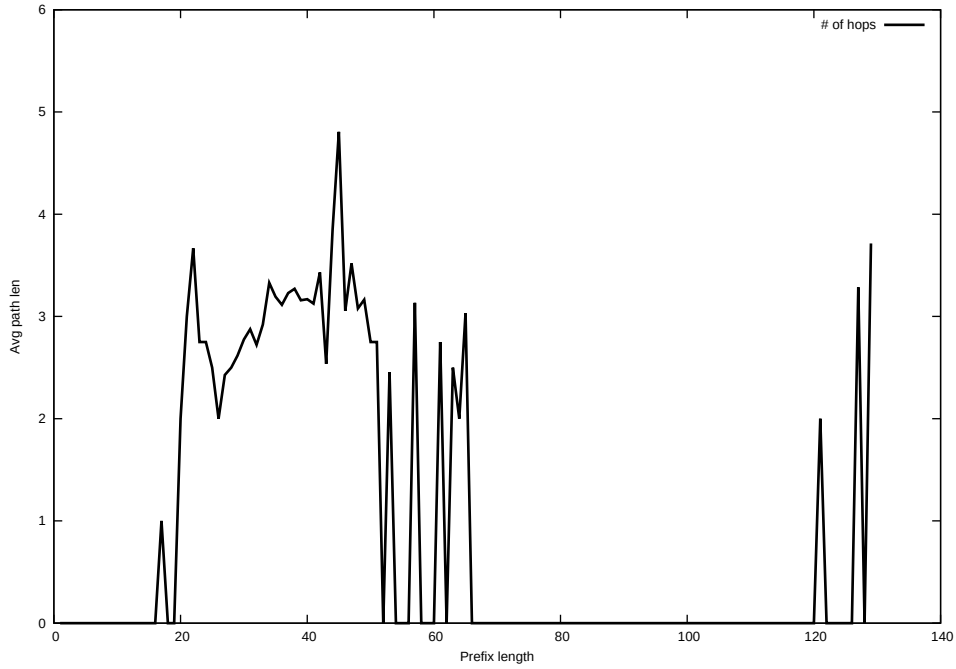
2014-09-12



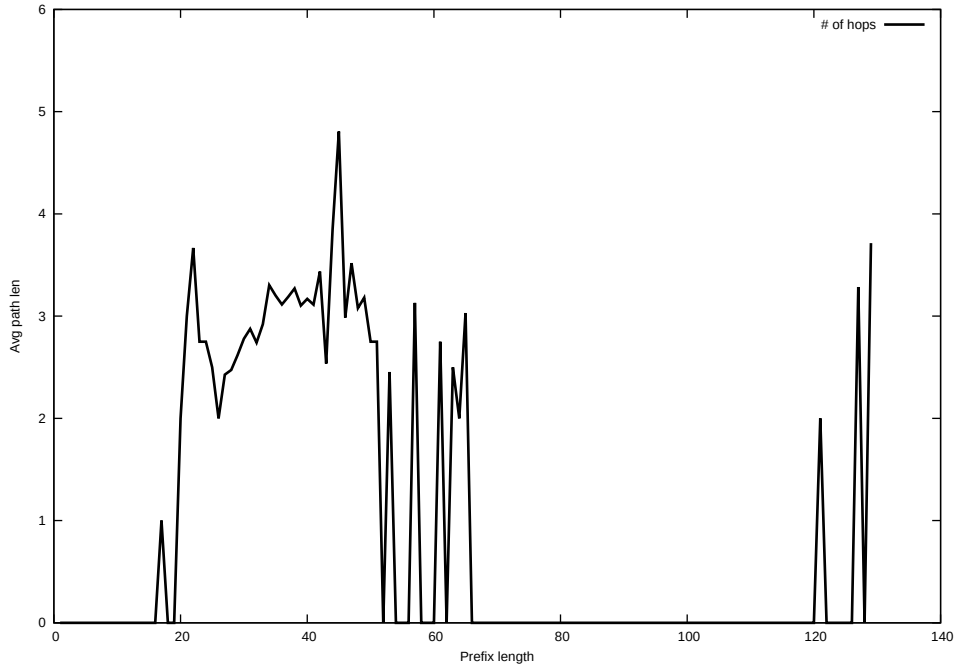
2014-09-13



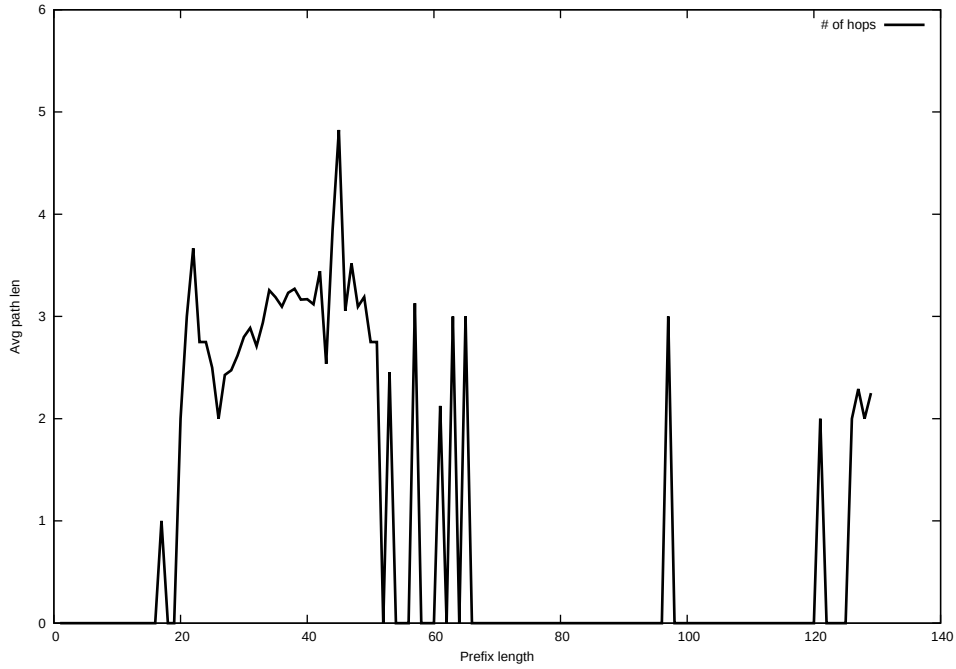
2014-09-14



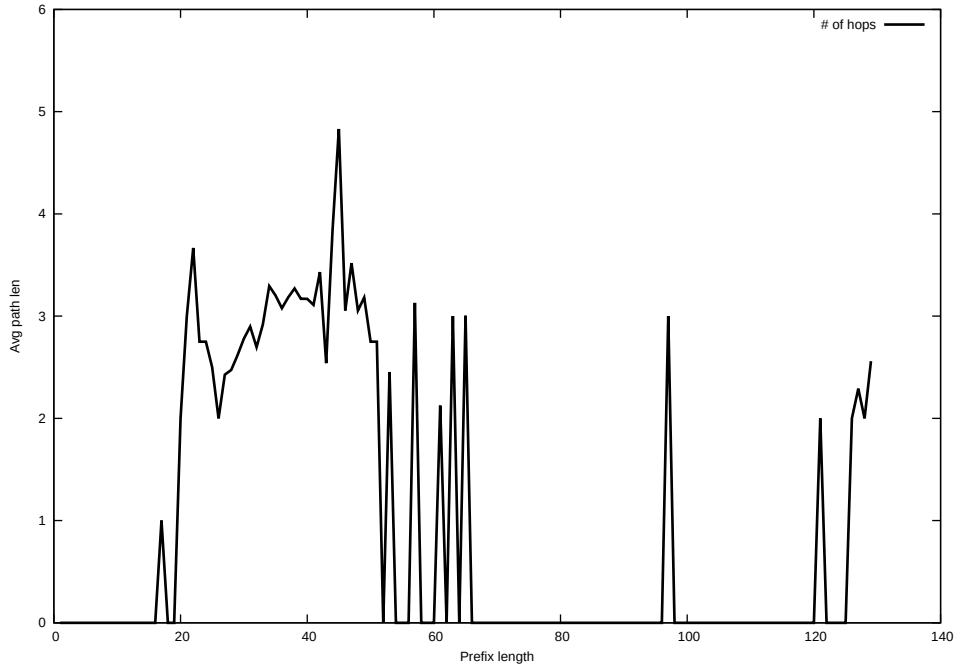
2014-09-15



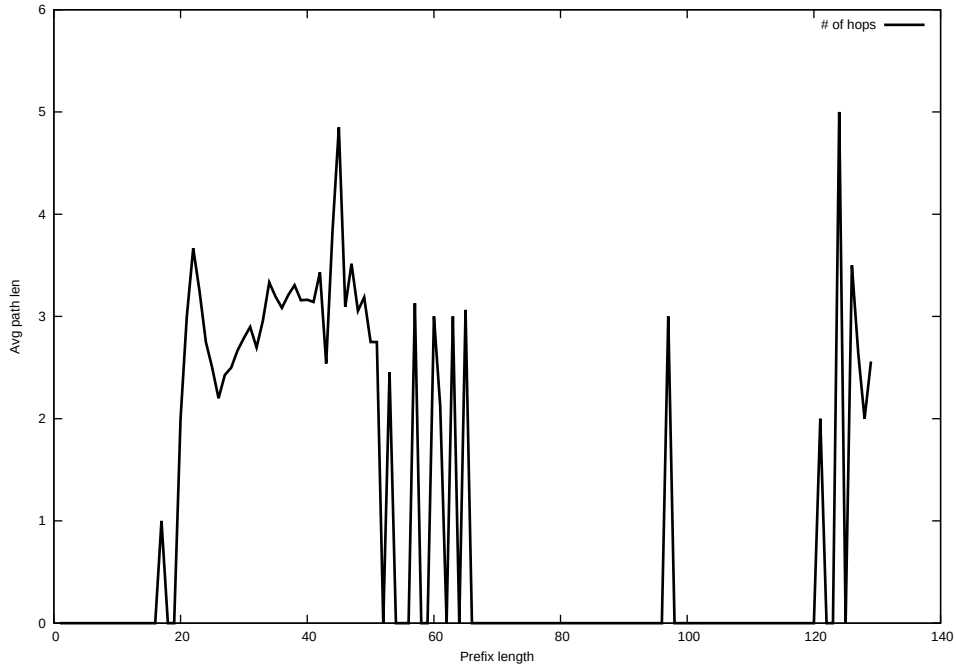
2014-09-16



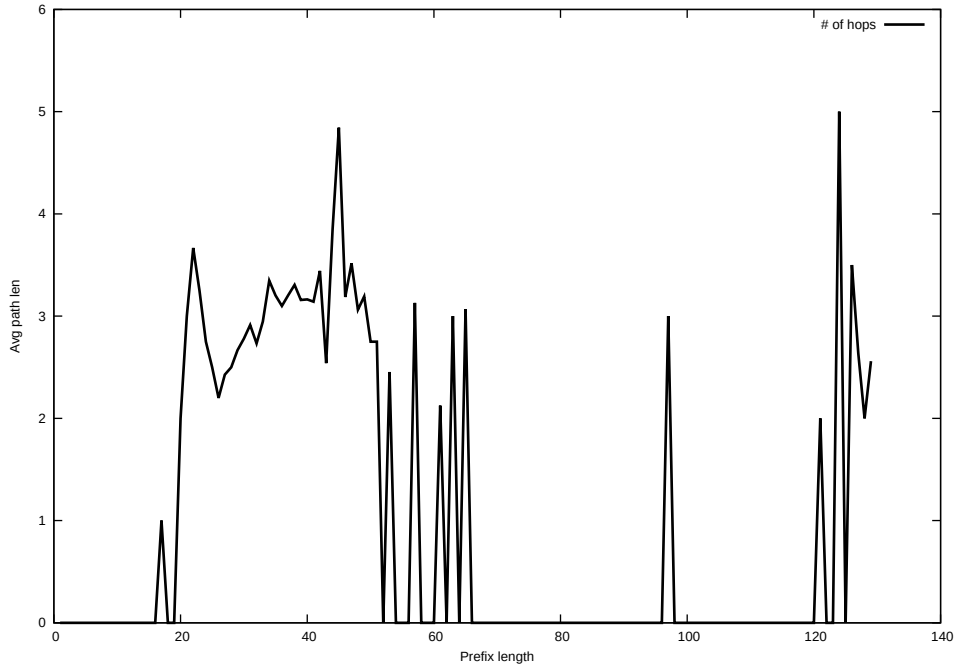
2014-09-17



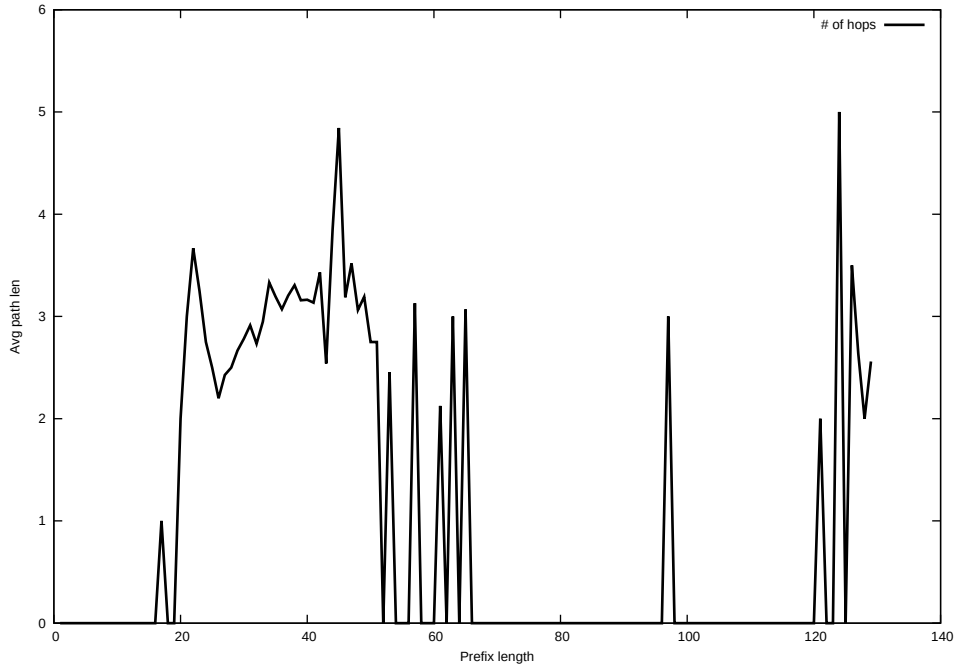
2014-09-18



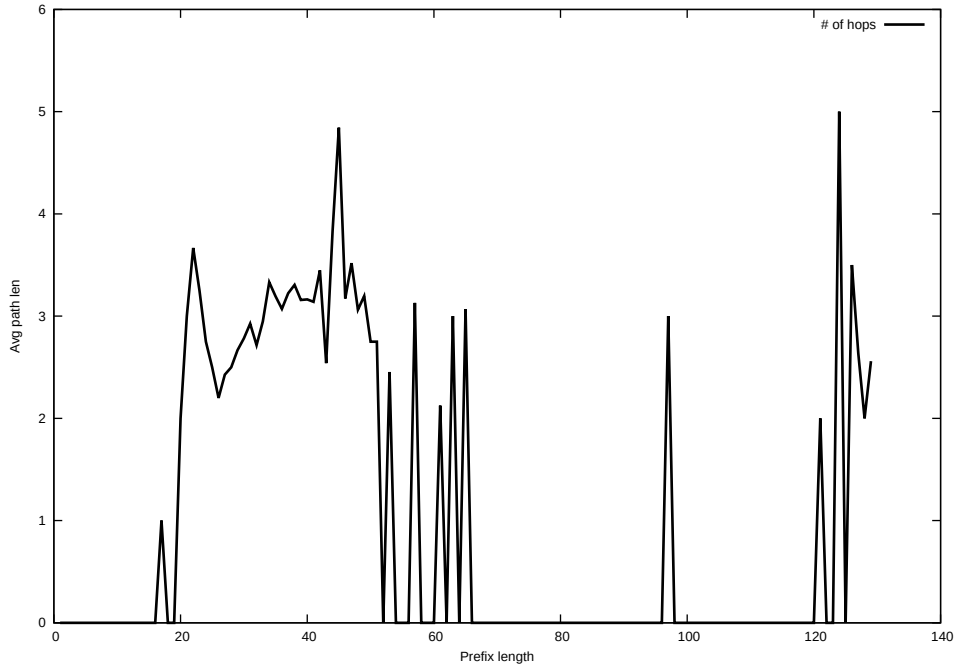
2014-09-19



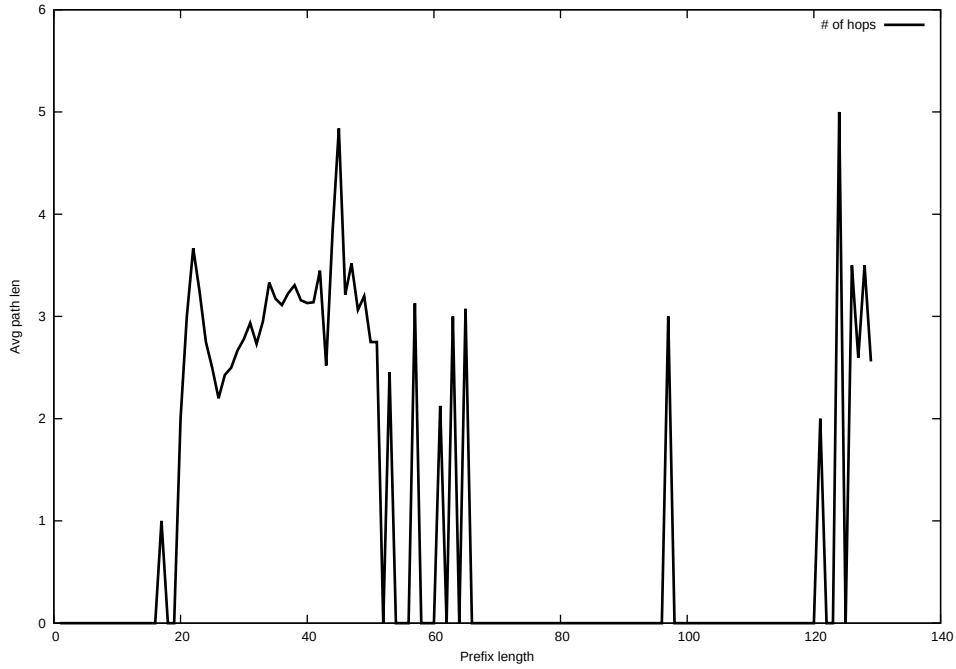
2014-09-20



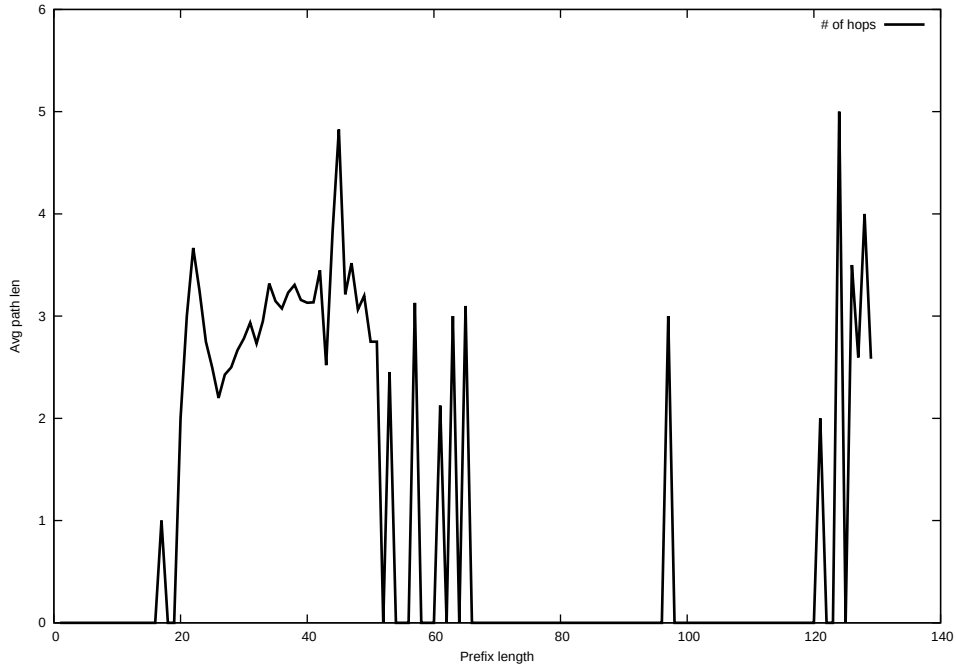
2014-09-21



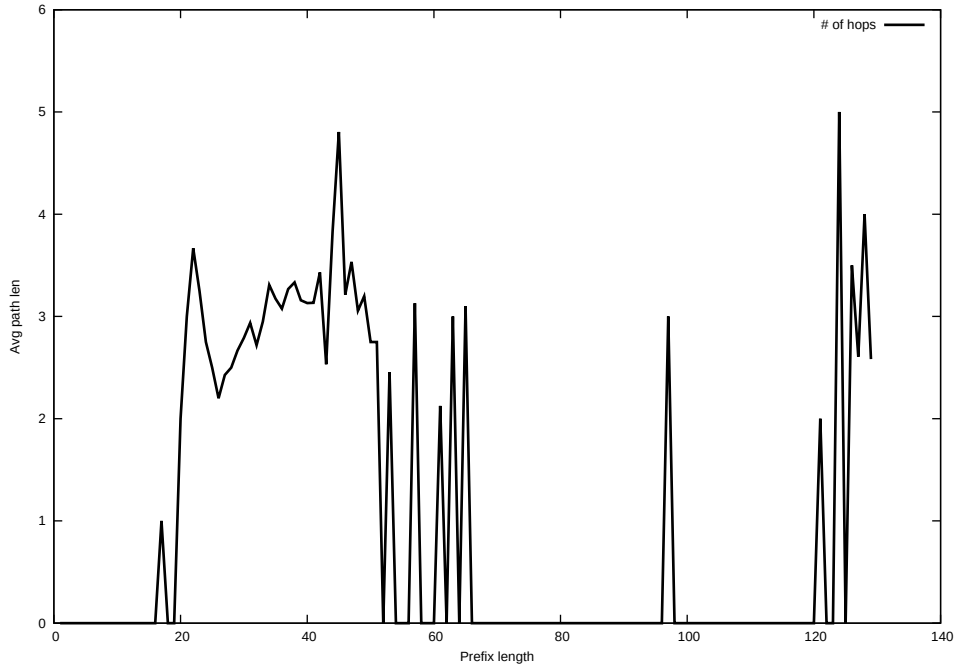
2014-09-22



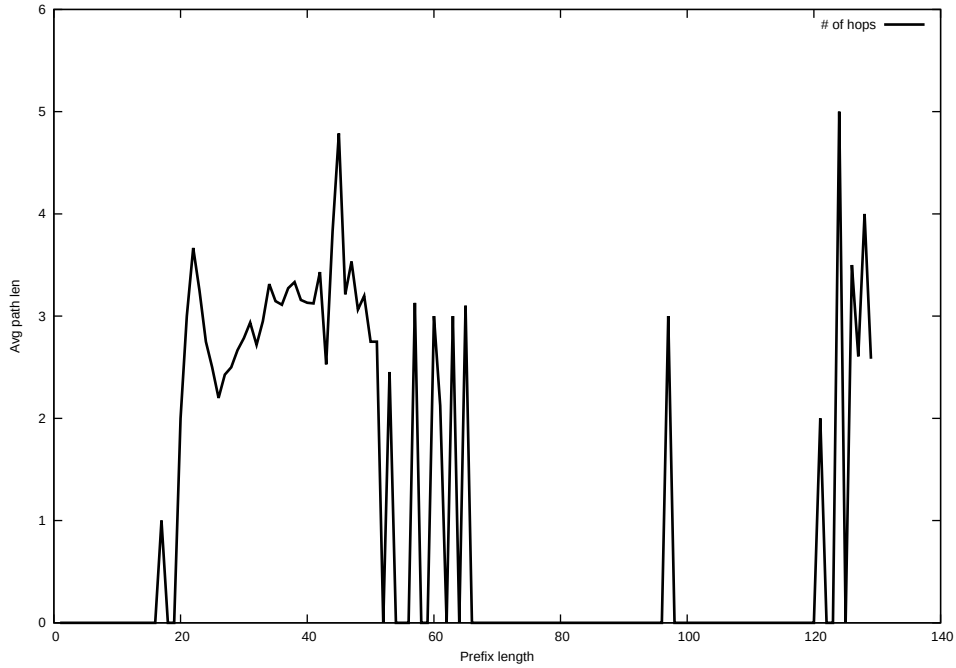
2014-09-23



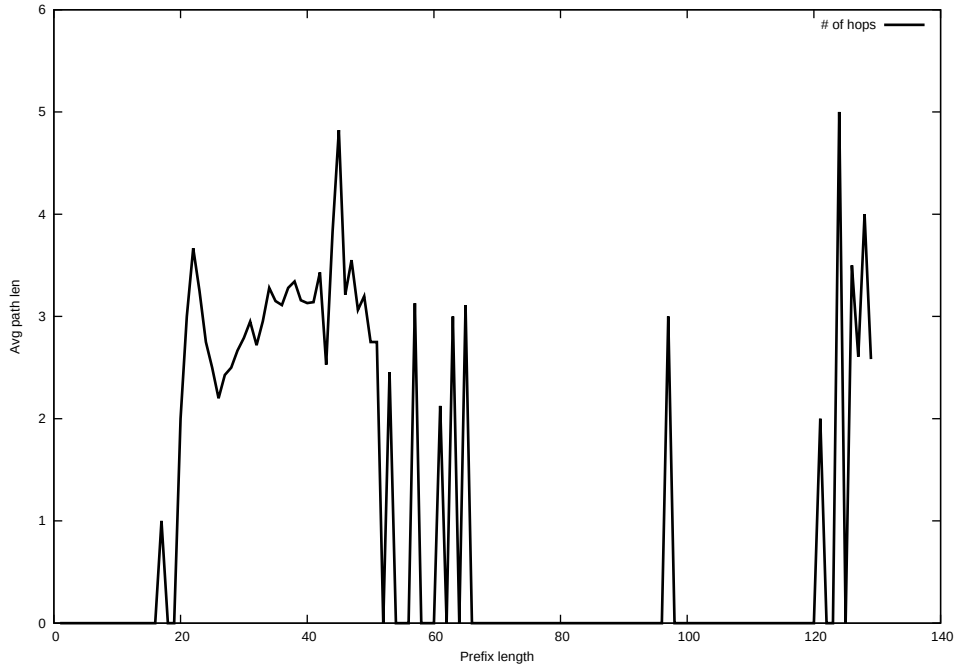
2014-09-24



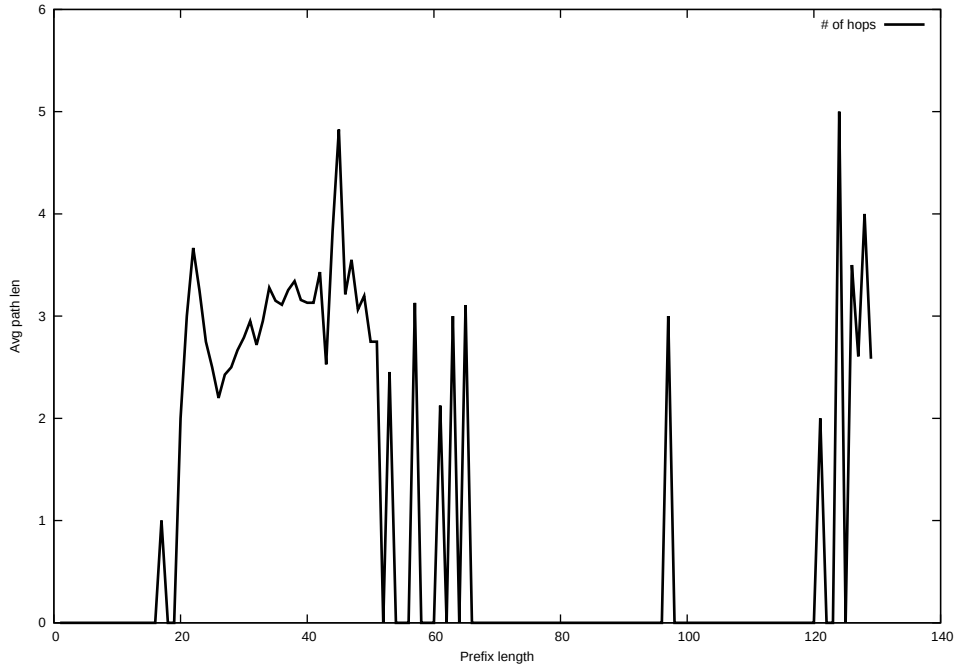
2014-09-25



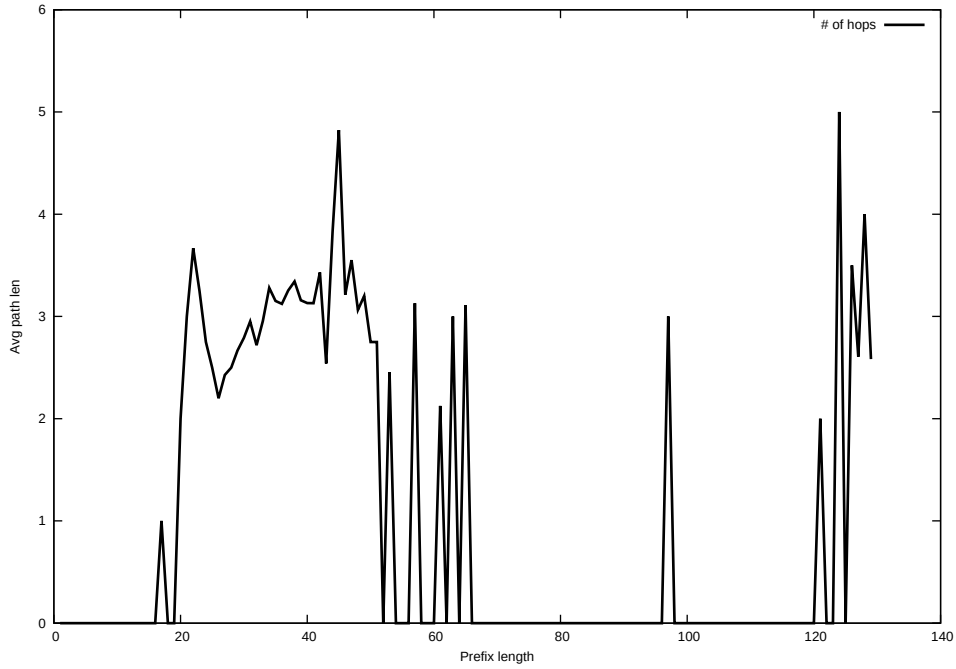
2014-09-26



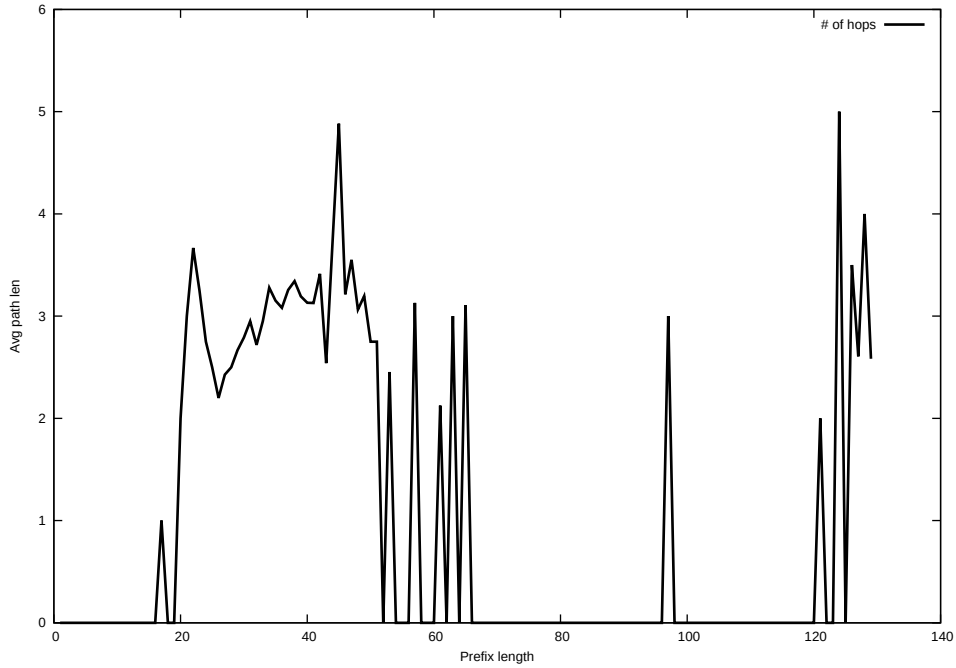
2014-09-27



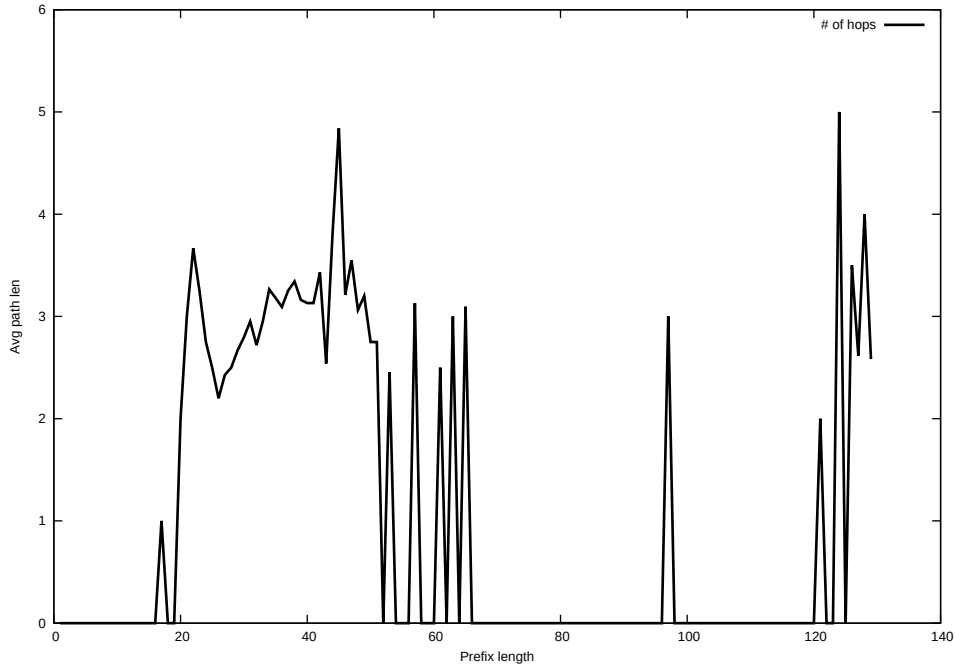
2014-09-28



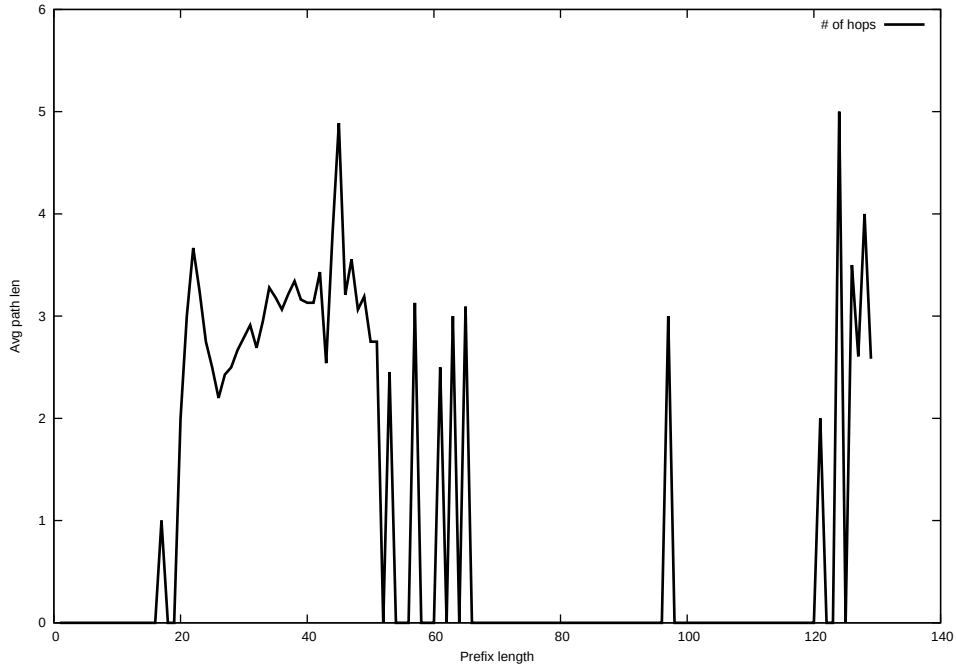
2014-09-29



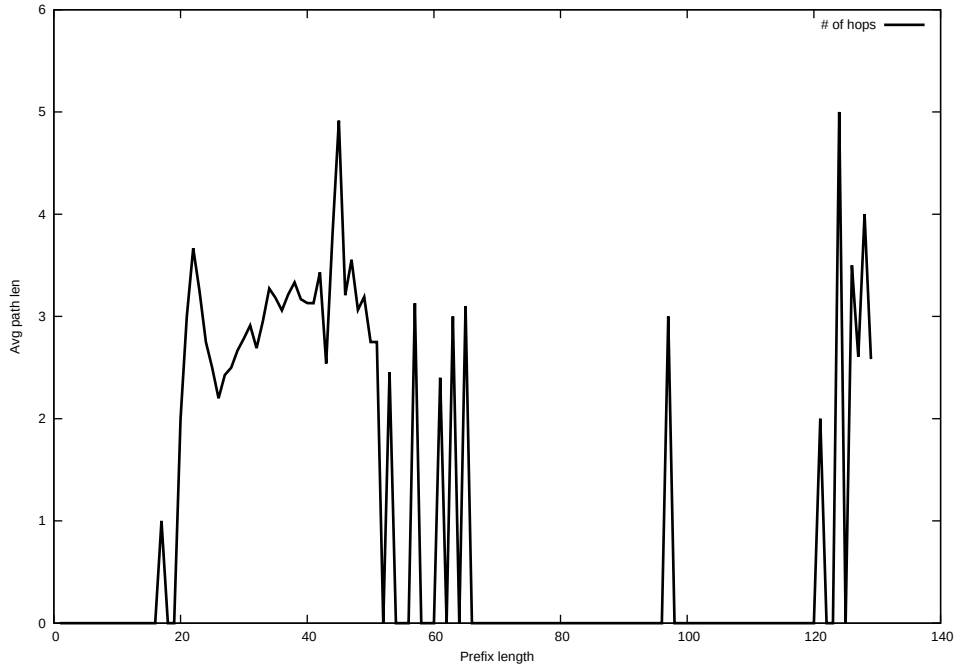
2014-09-30



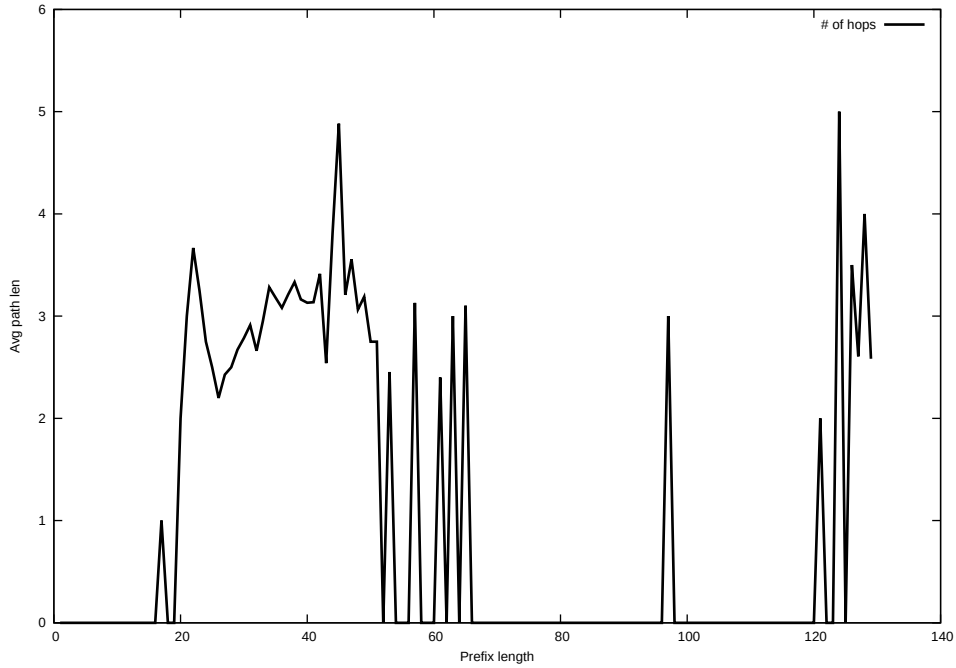
2014-10-01



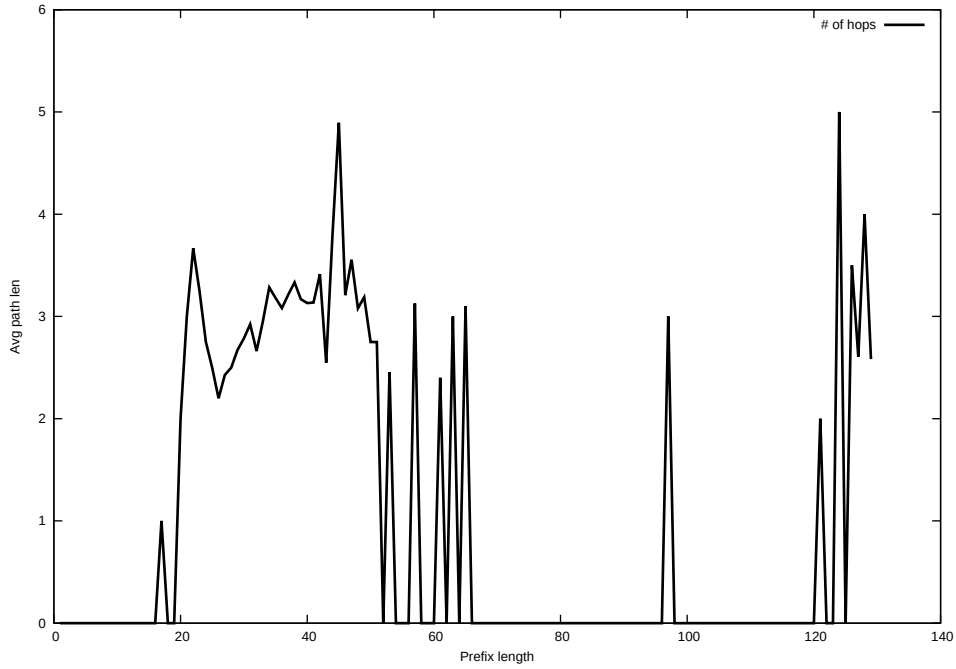
2014-10-02



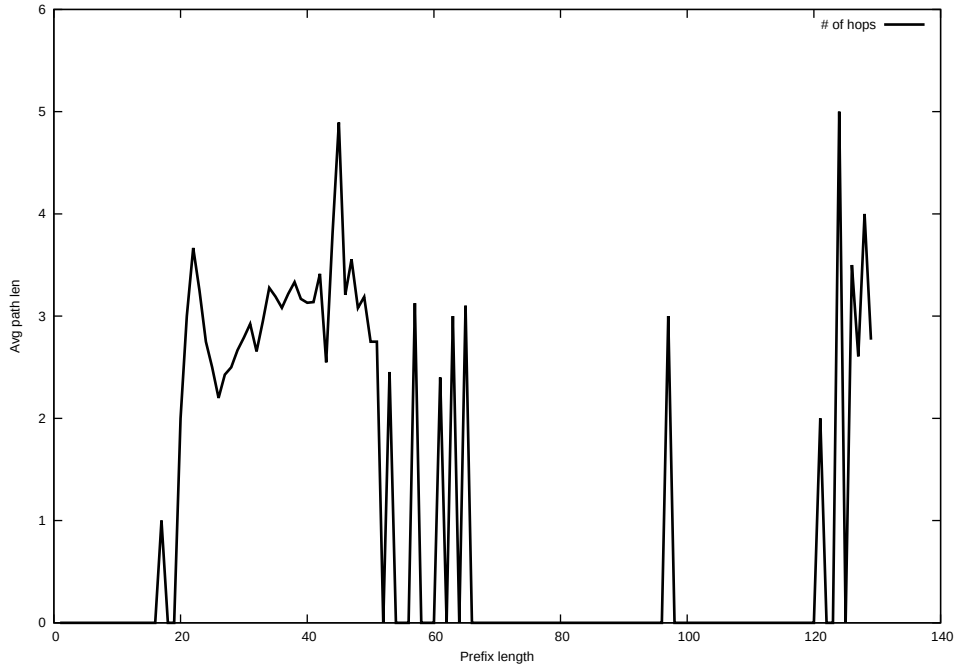
2014-10-03



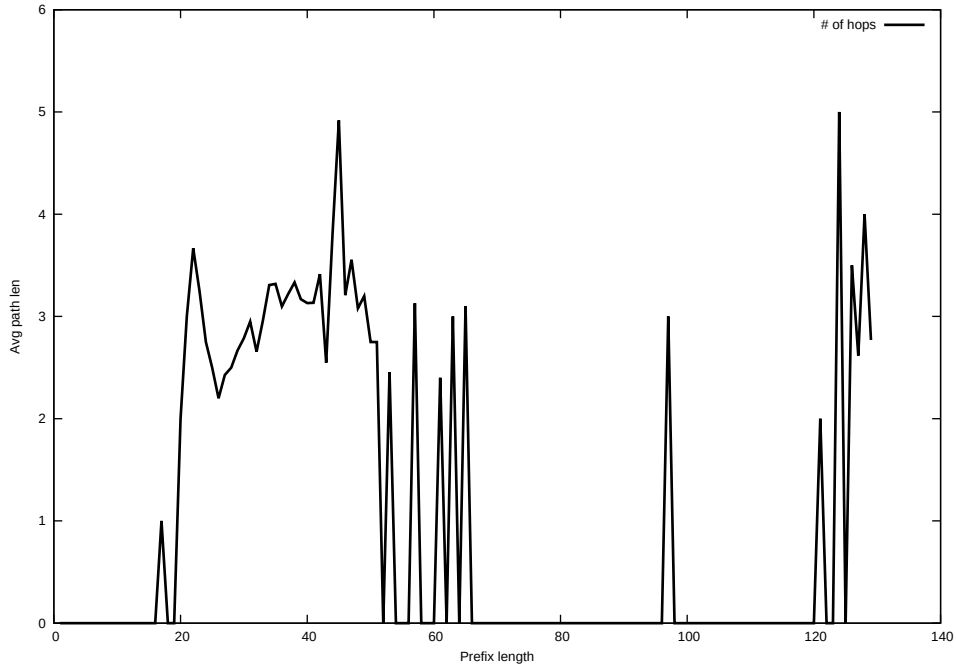
2014-10-04



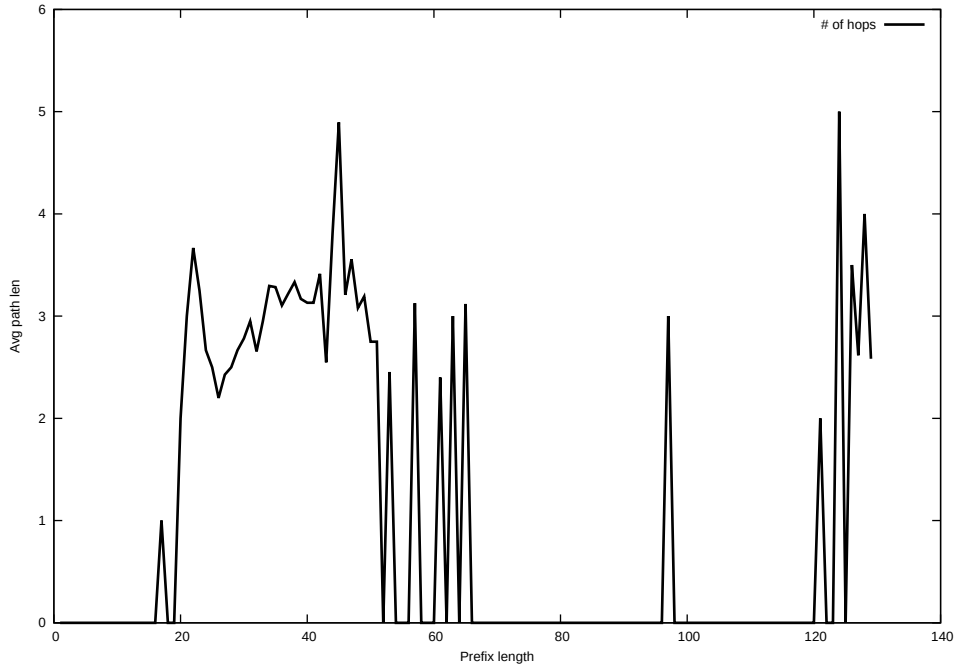
2014-10-05



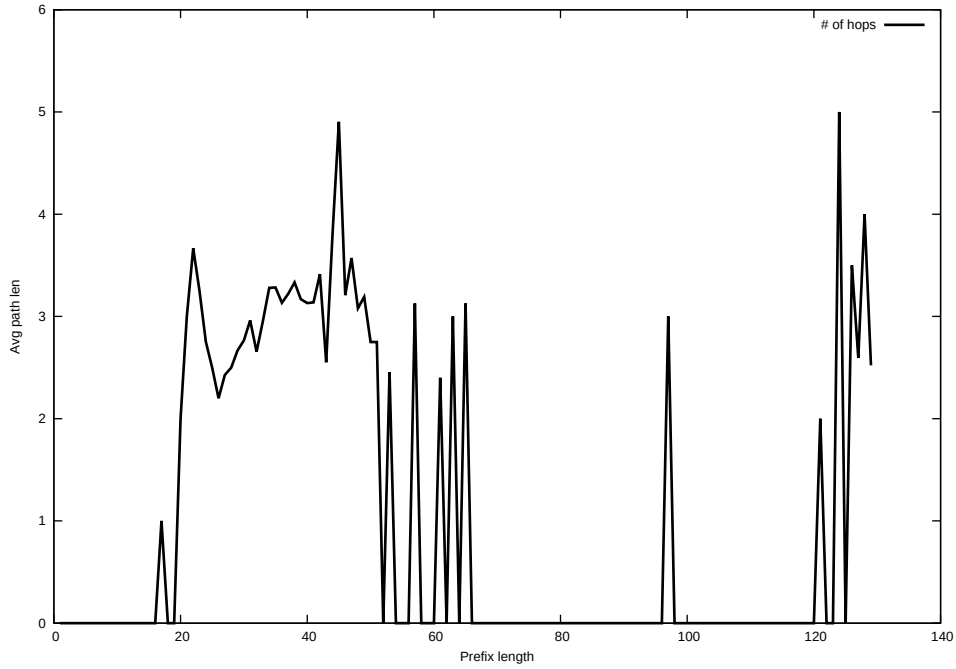
2014-10-06



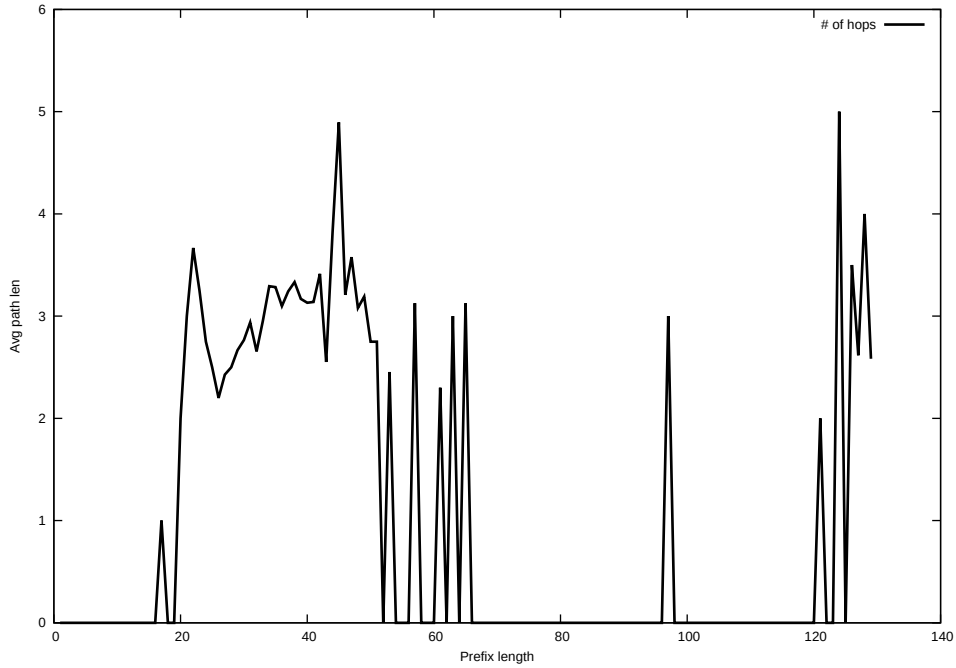
2014-10-07



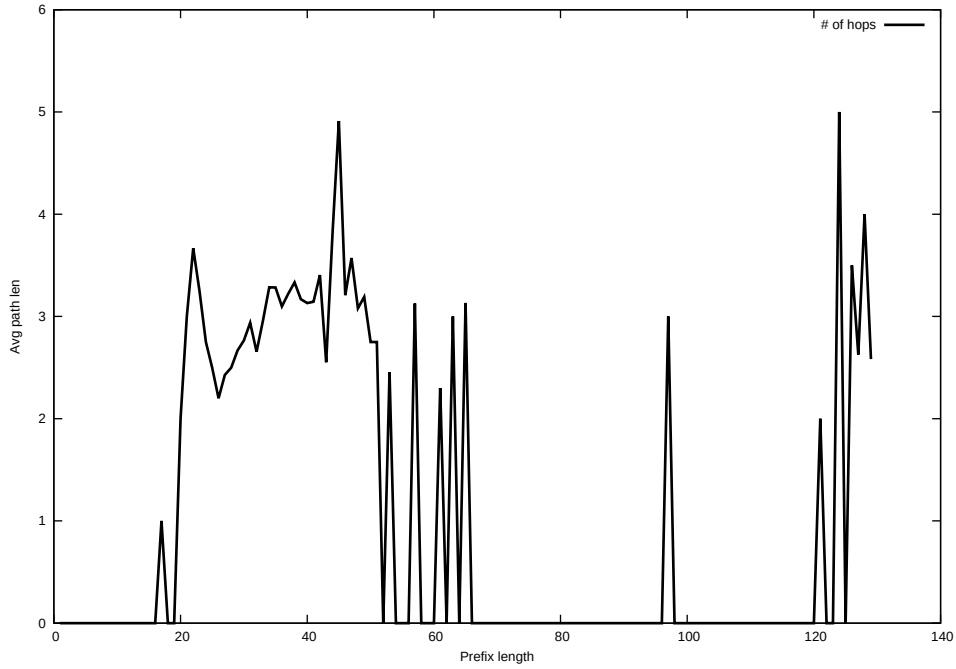
2014-10-08



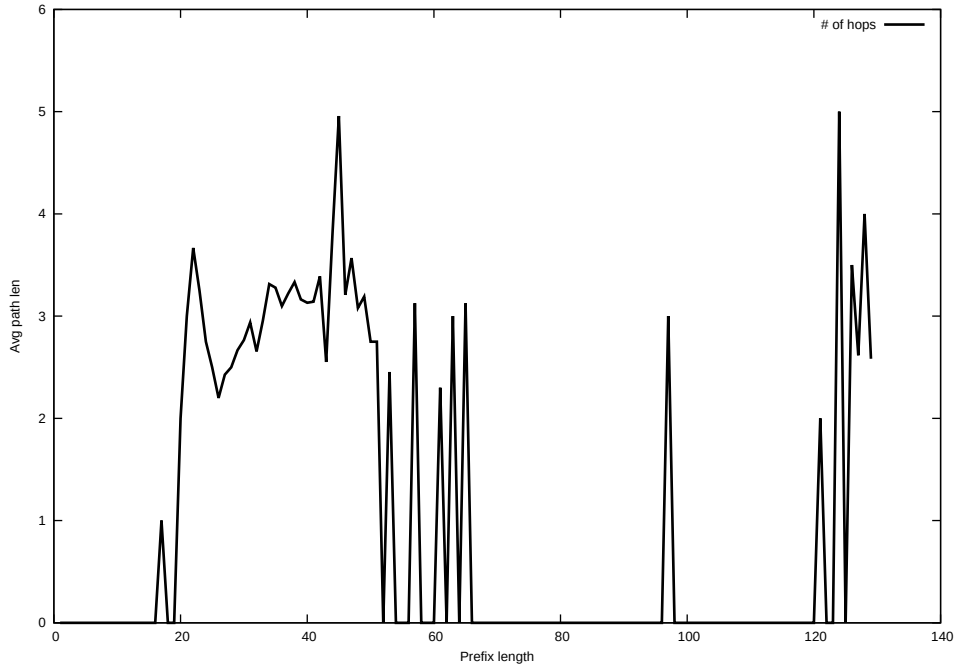
2014-10-09



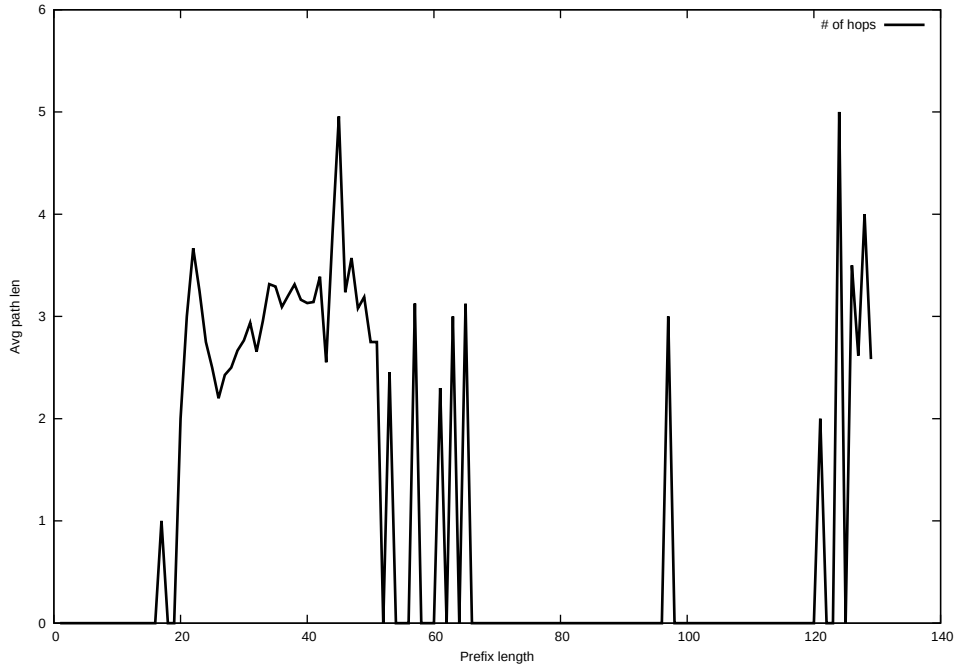
2014-10-10



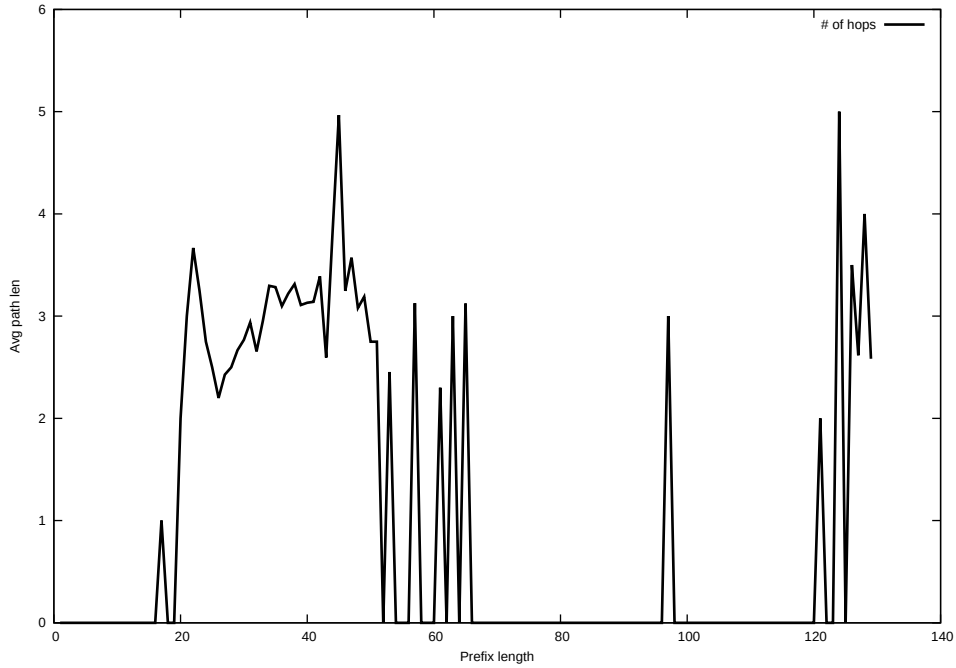
2014-10-11



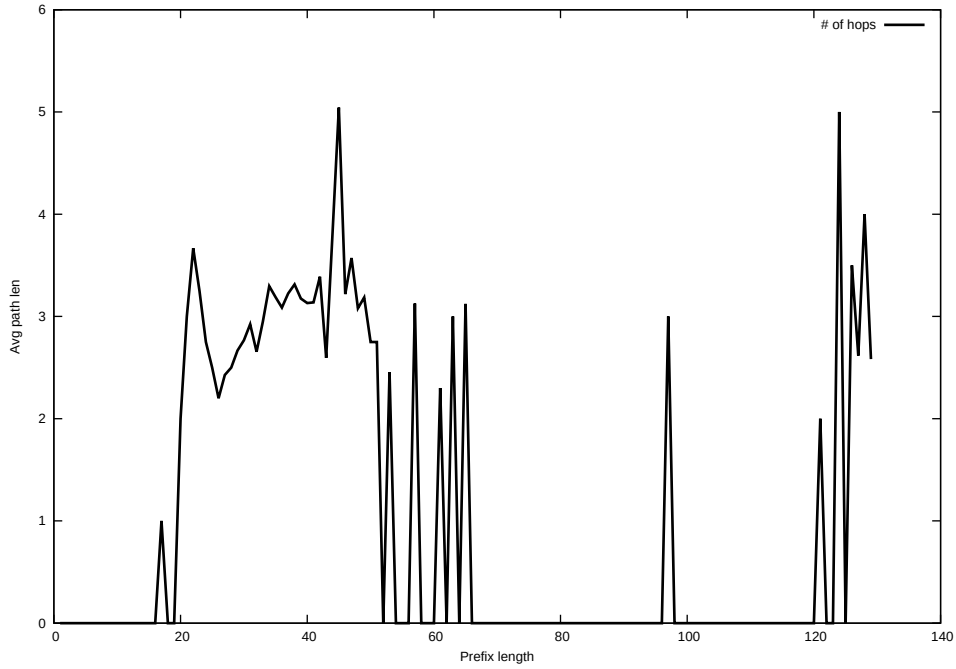
2014-10-12



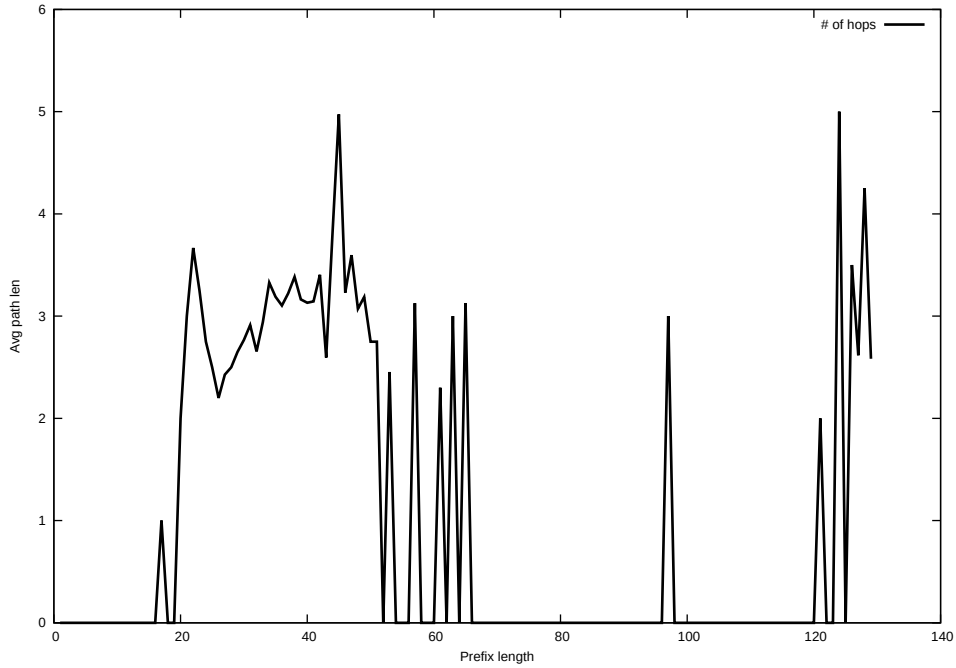
2014-10-13



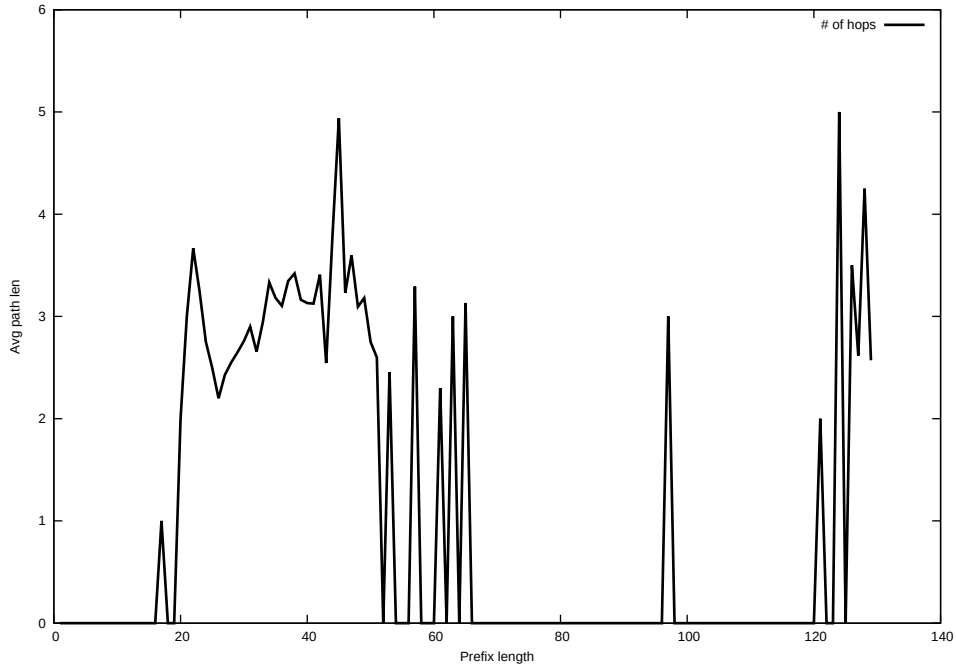
2014-10-14



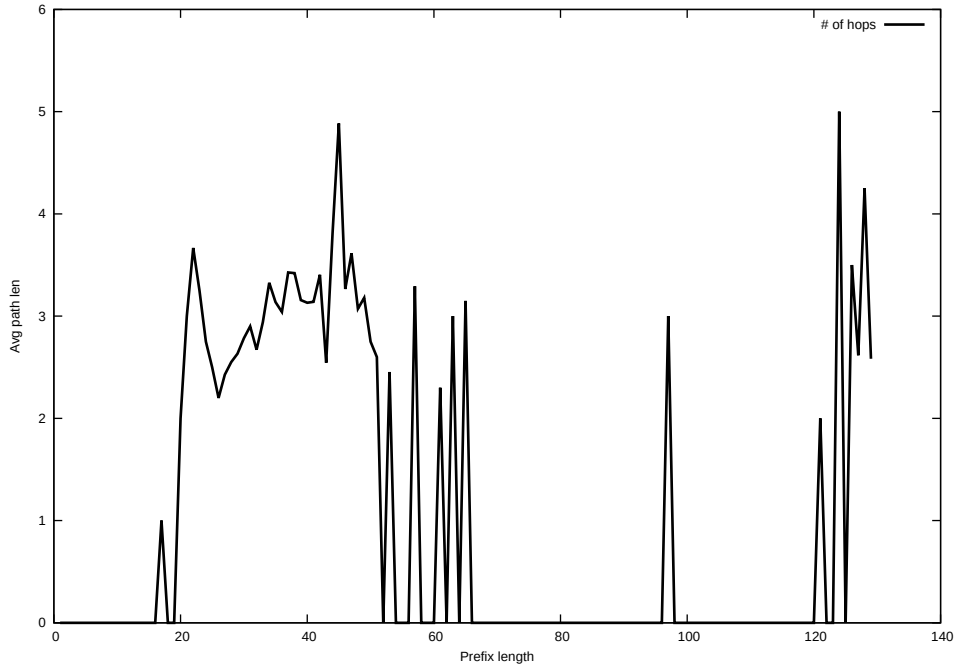
2014-10-15



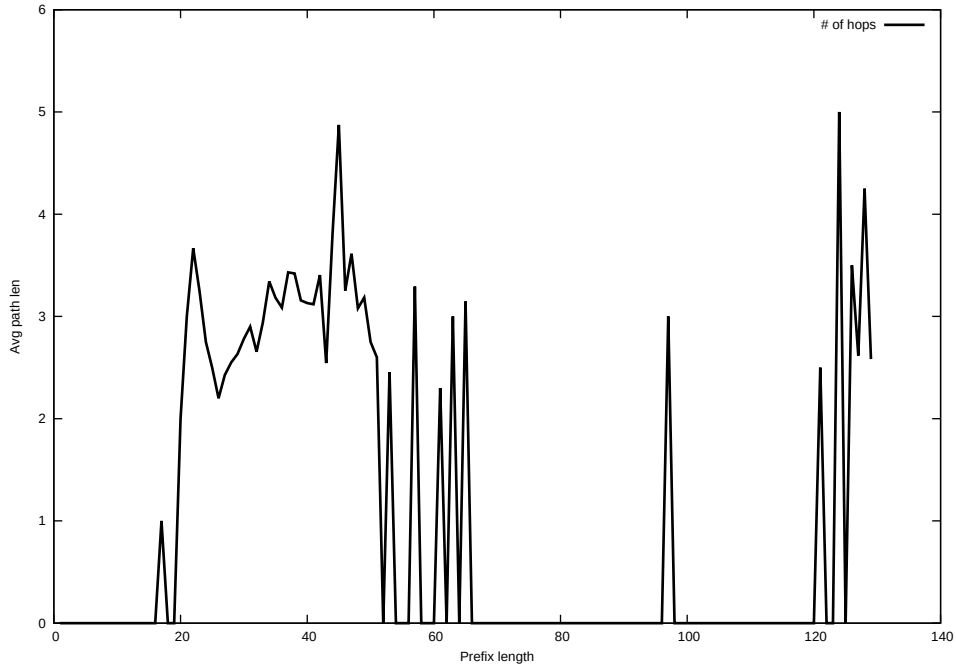
2014-10-16



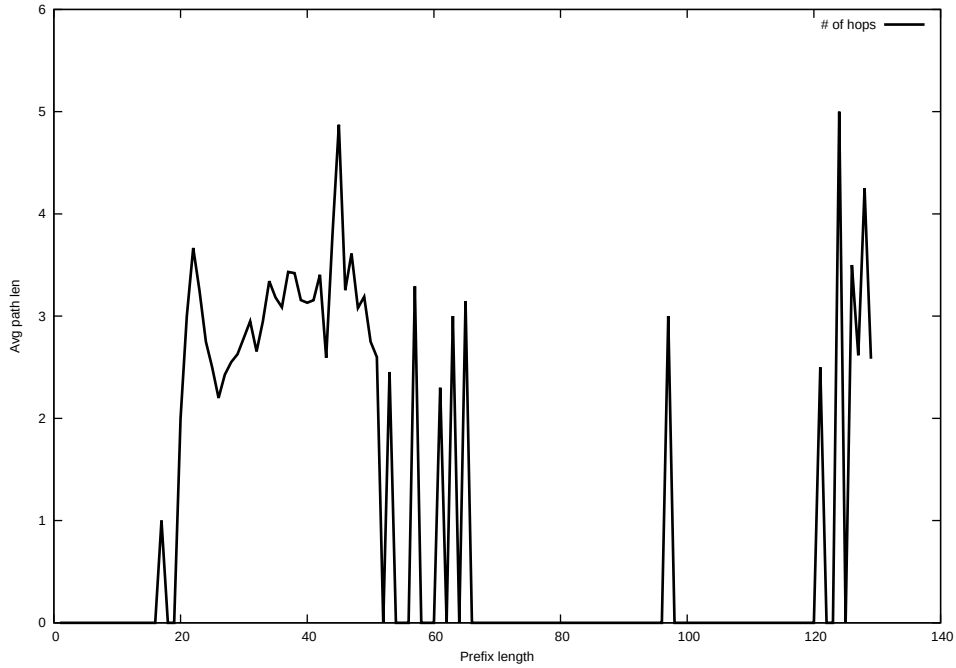
2014-10-17



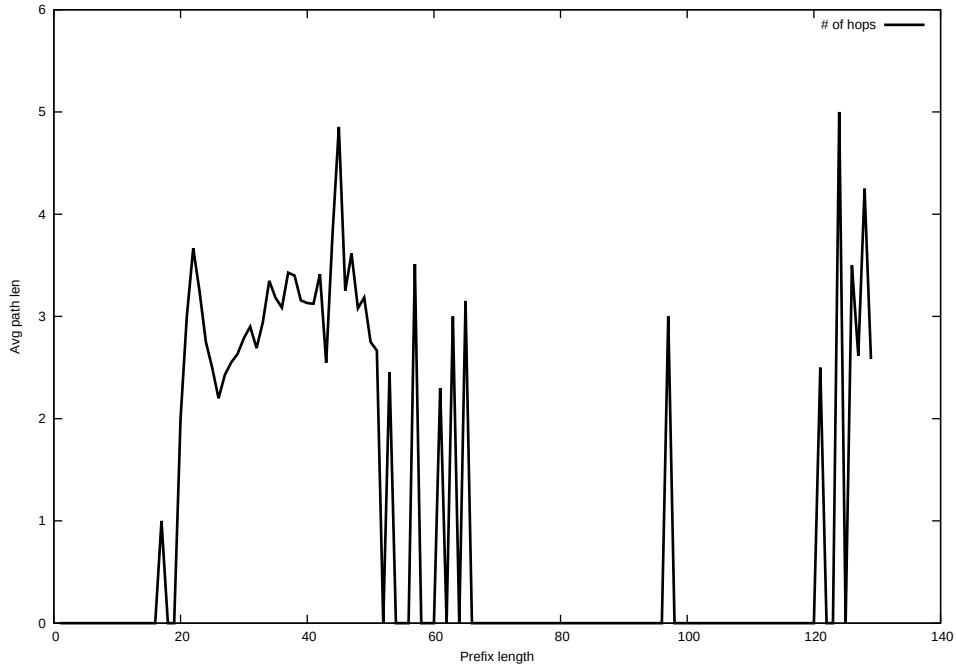
2014-10-18



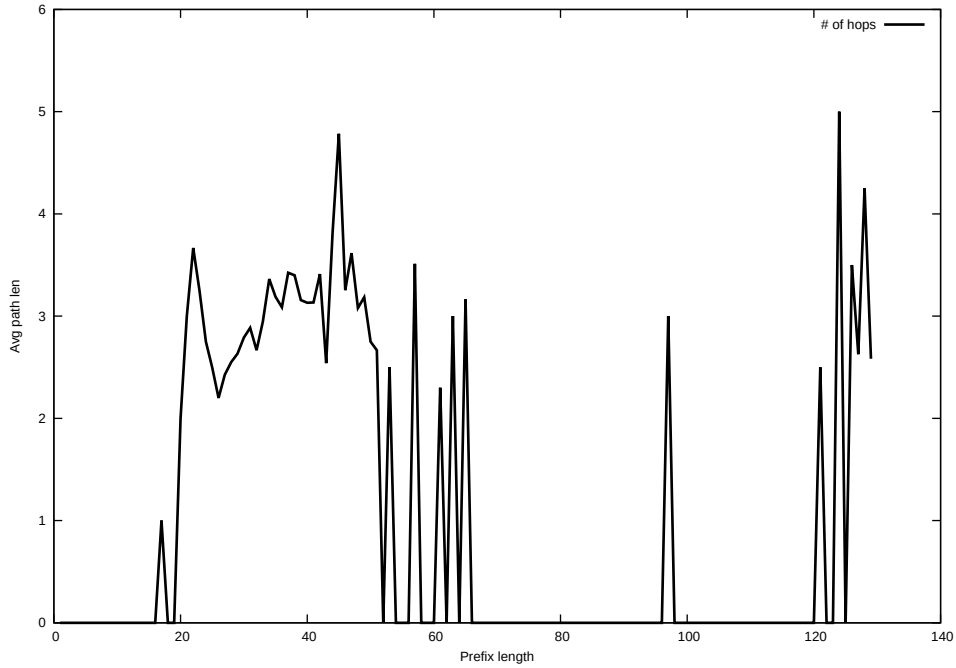
2014-10-19



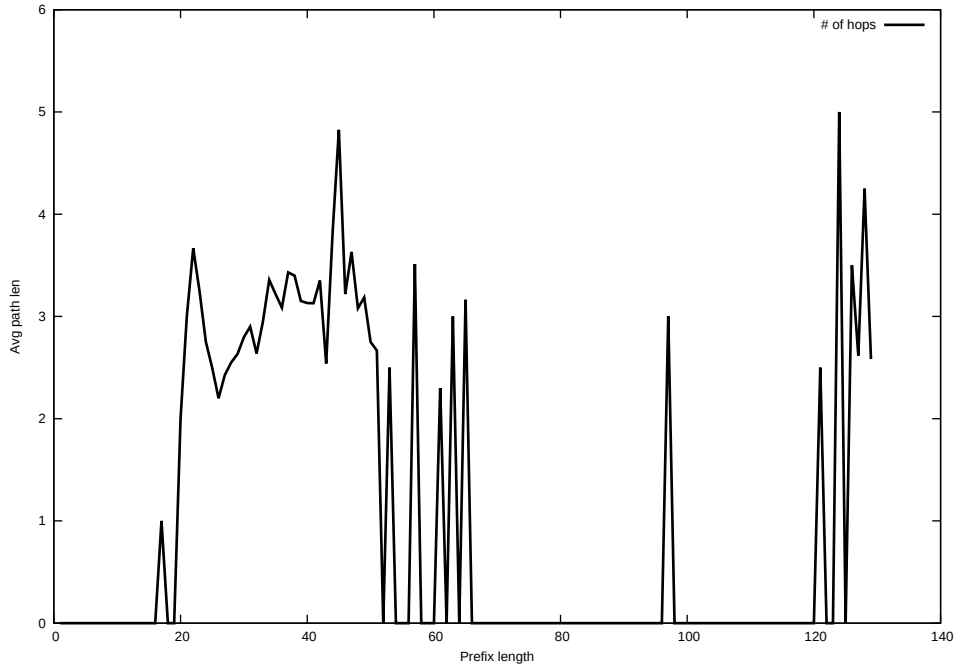
2014-10-20



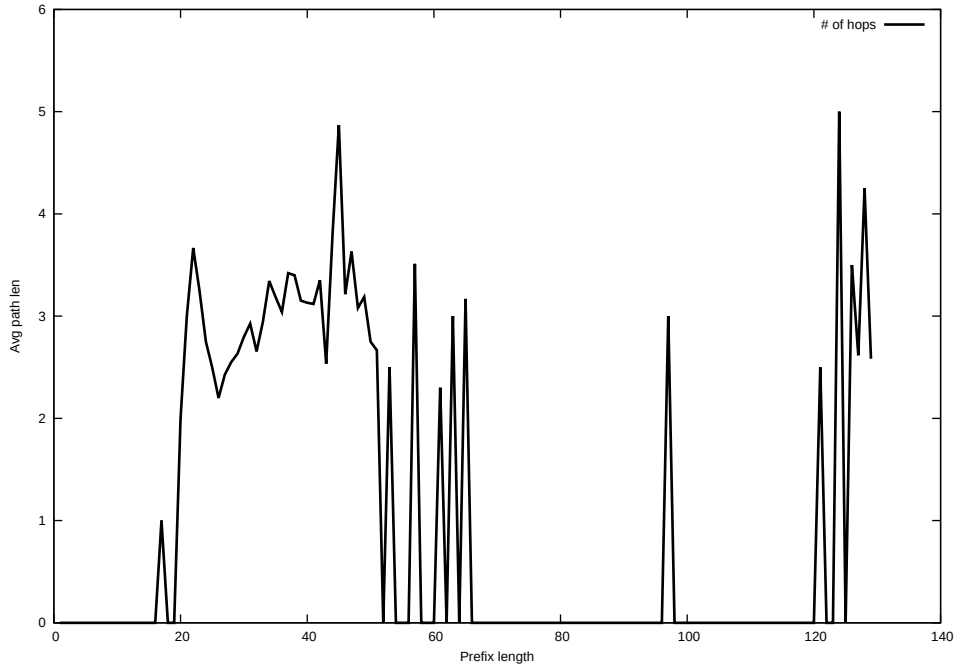
2014-10-21



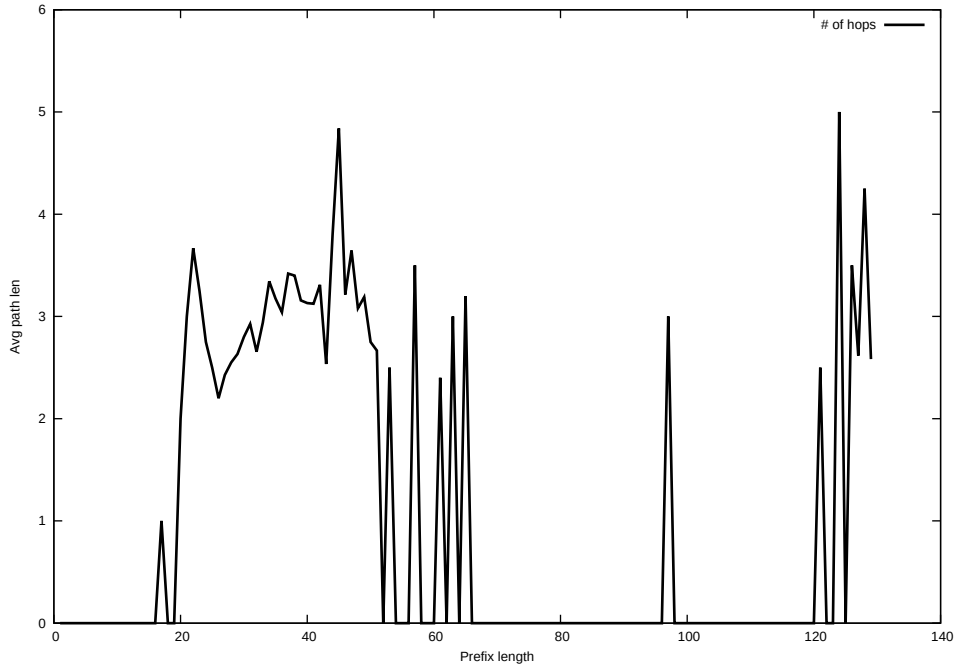
2014-10-22



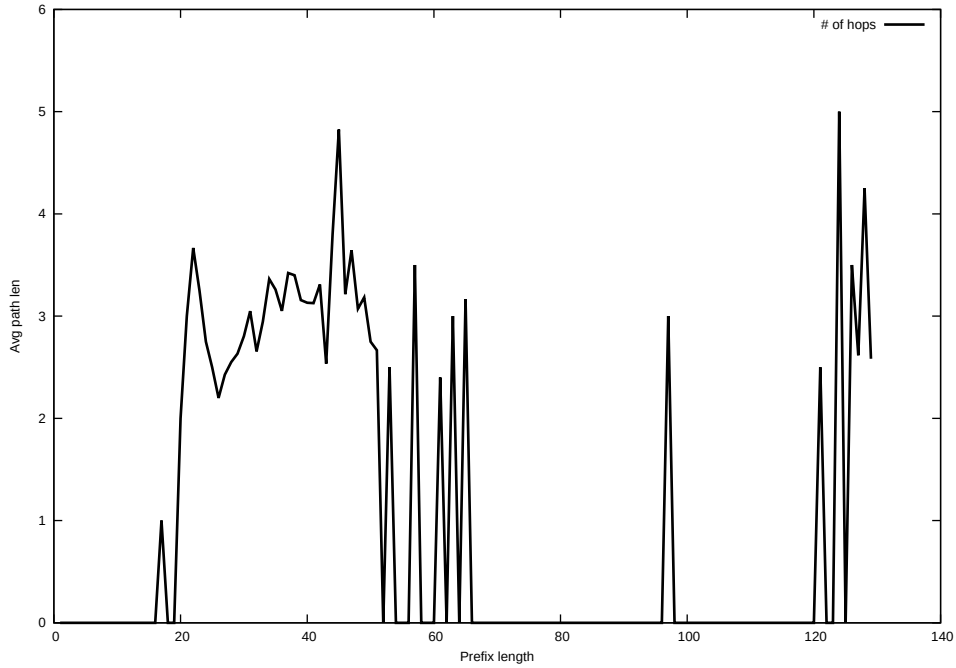
2014-10-23



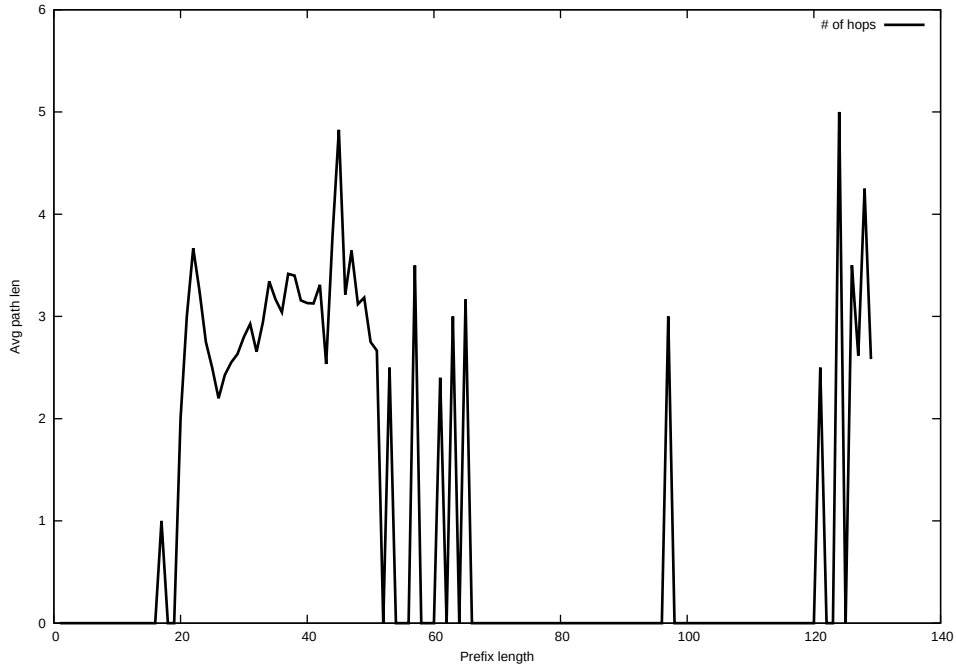
2014-10-24



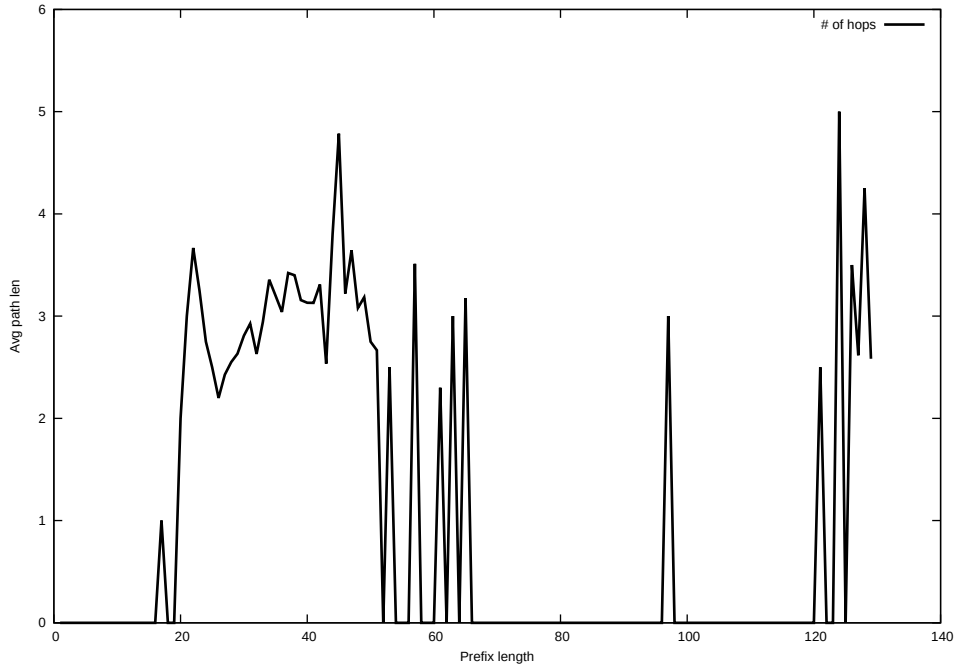
2014-10-25



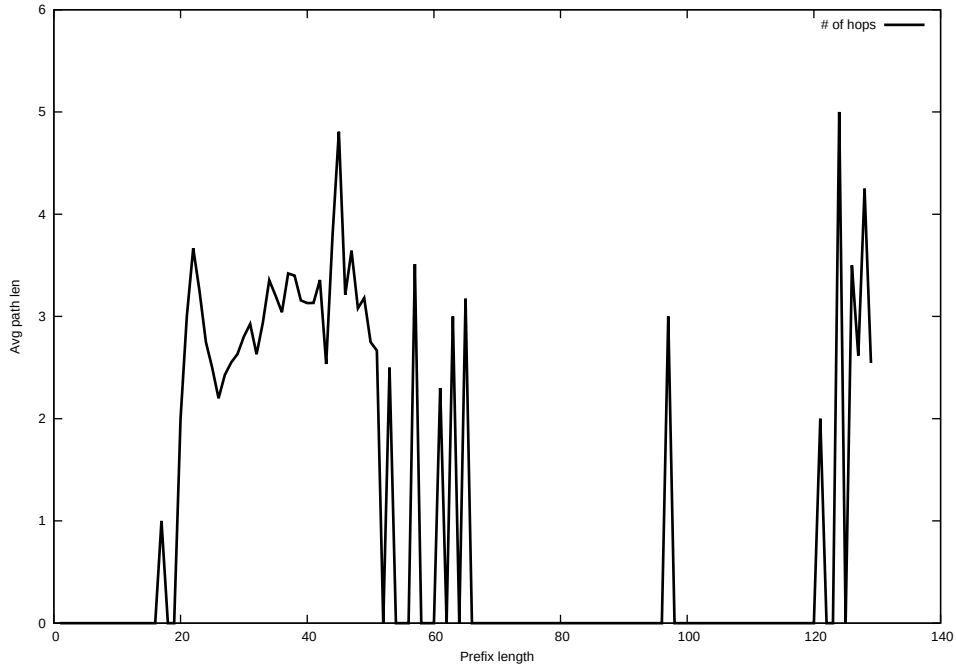
2014-10-26



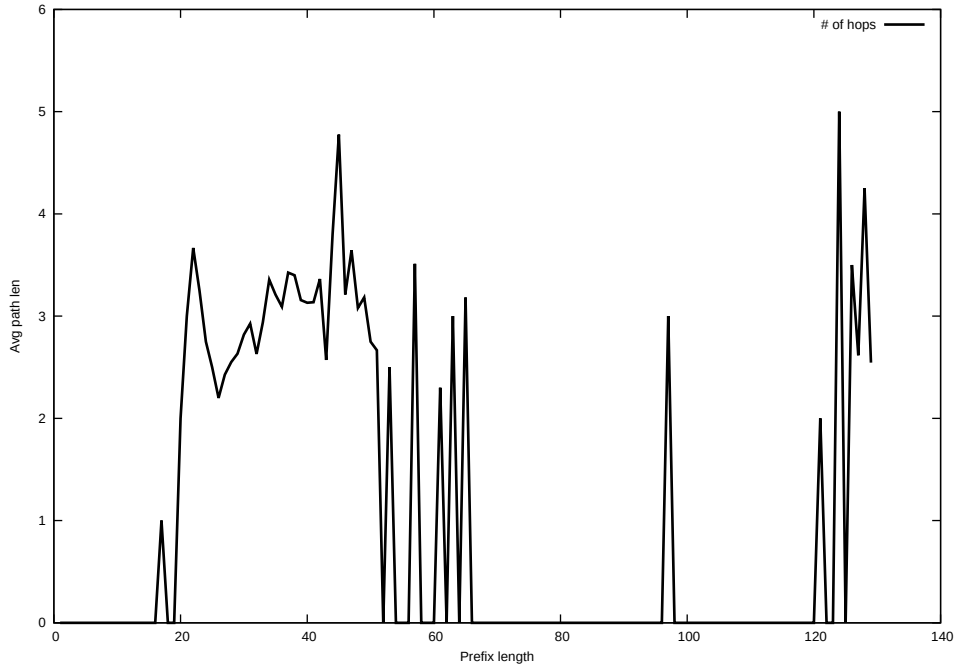
2014-10-27



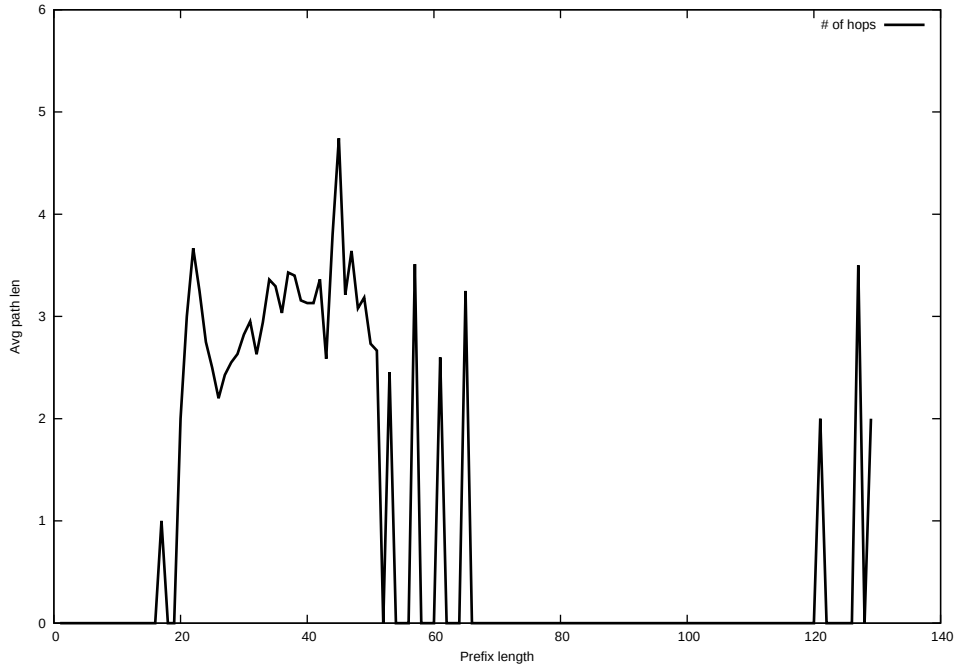
2014-10-28



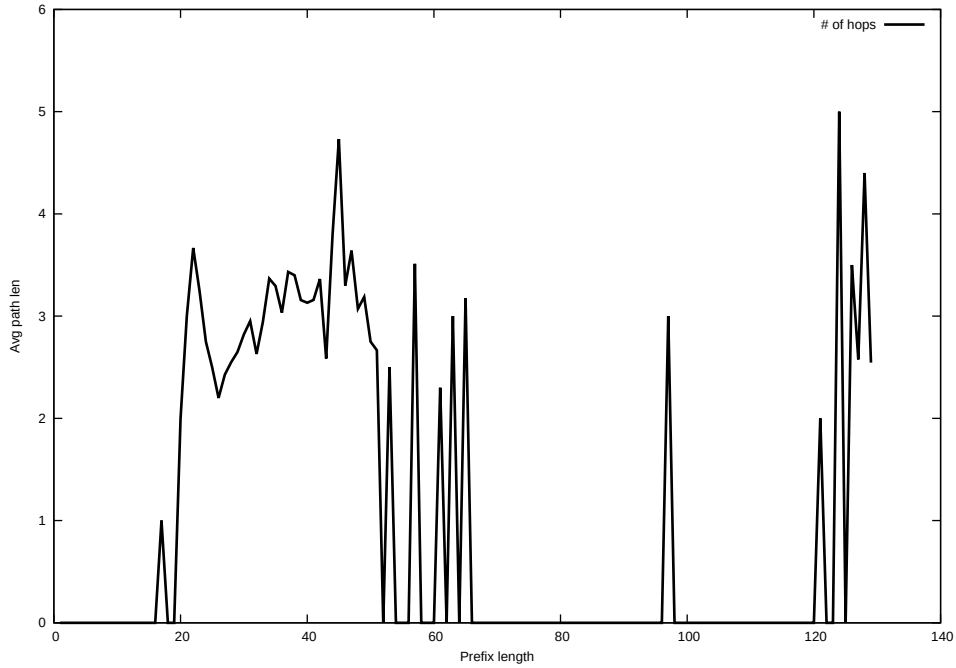
2014-10-29



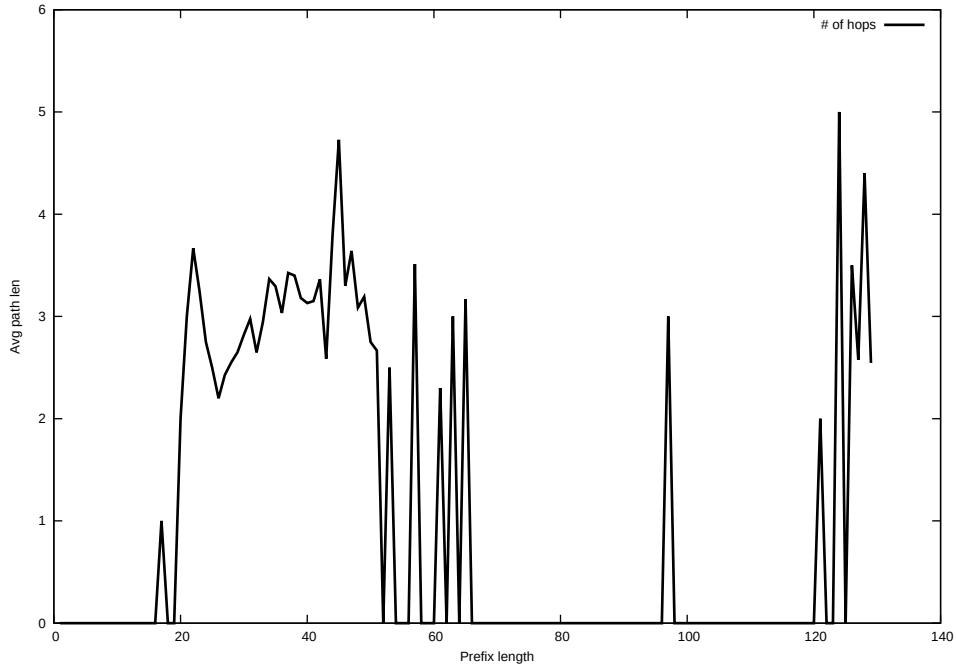
2014-10-30



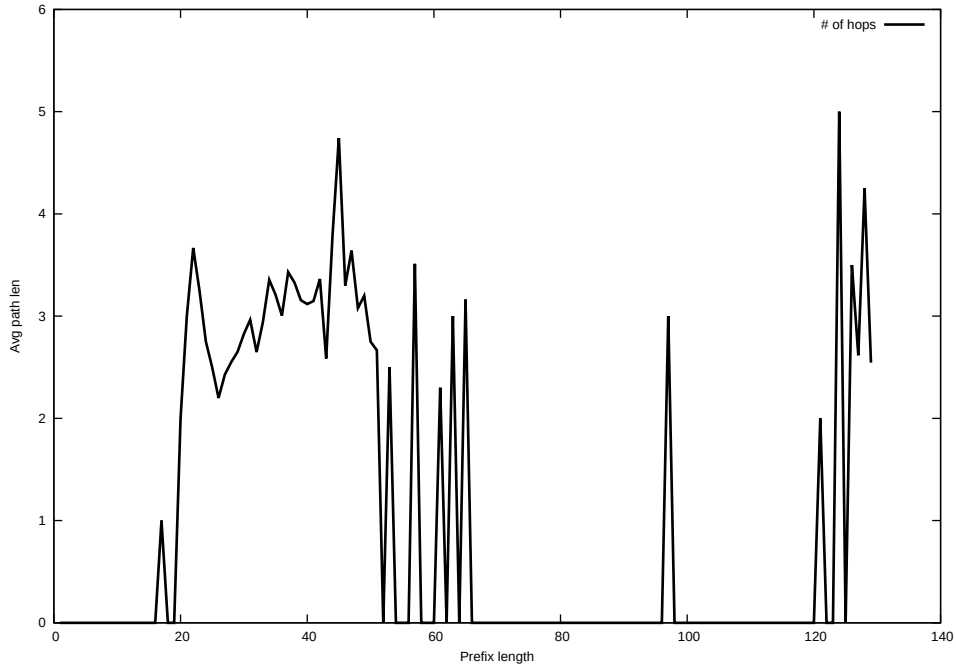
2014-10-31



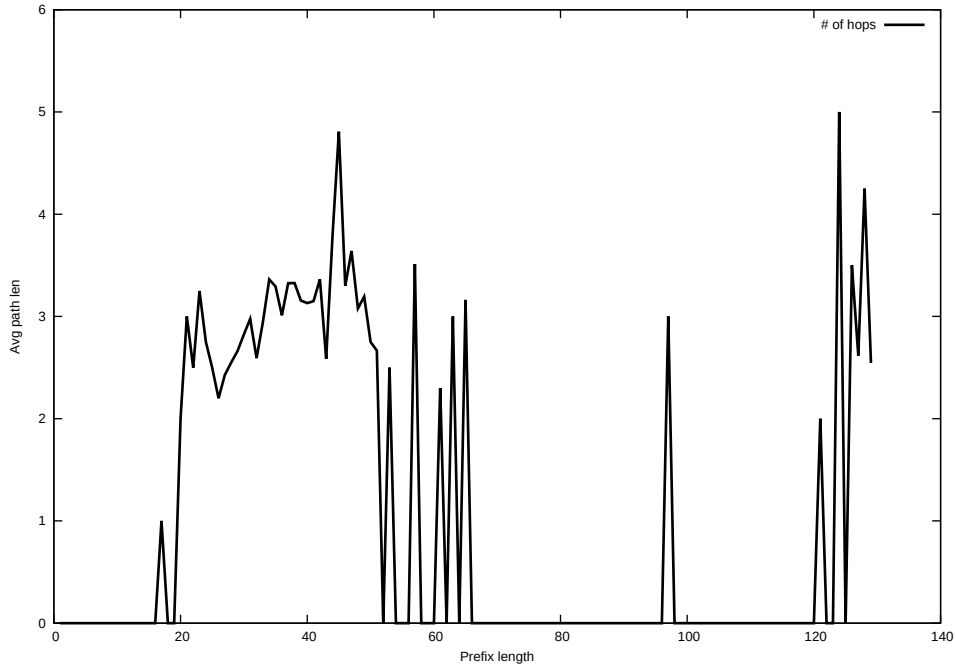
2014-11-02



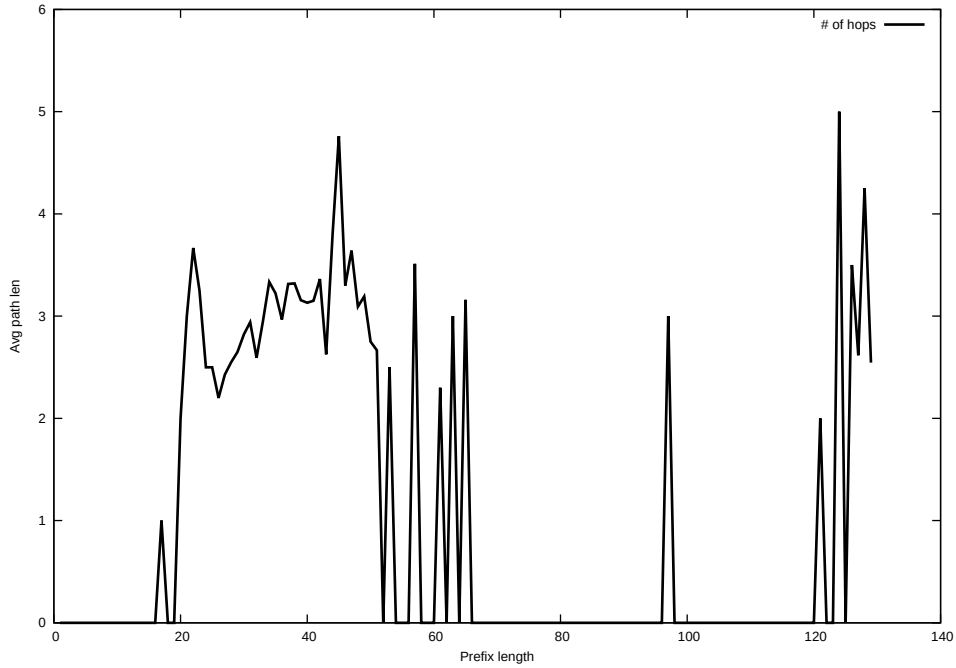
2014-11-03



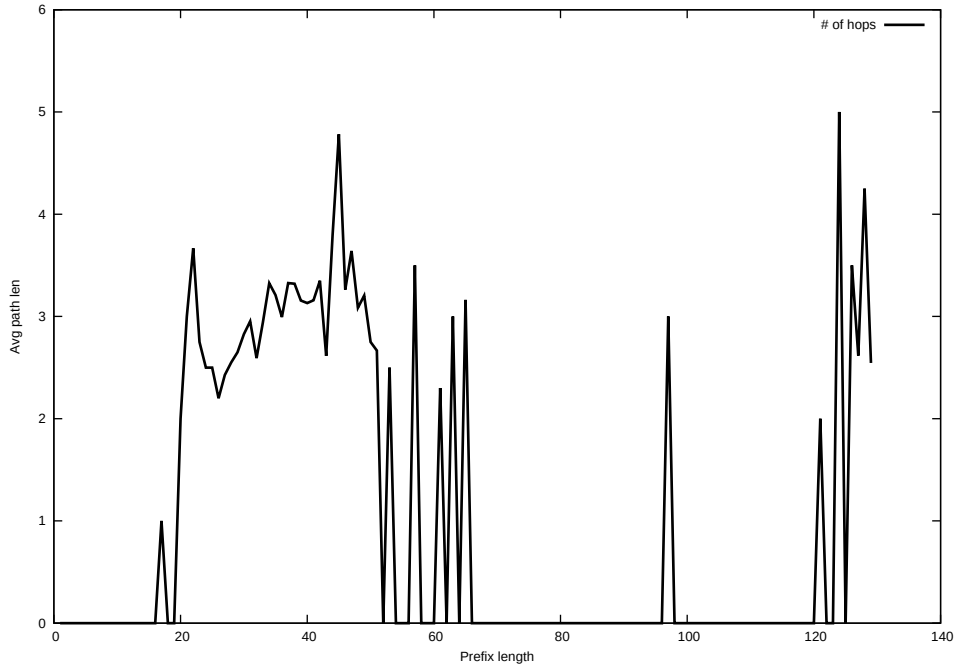
2014-11-04



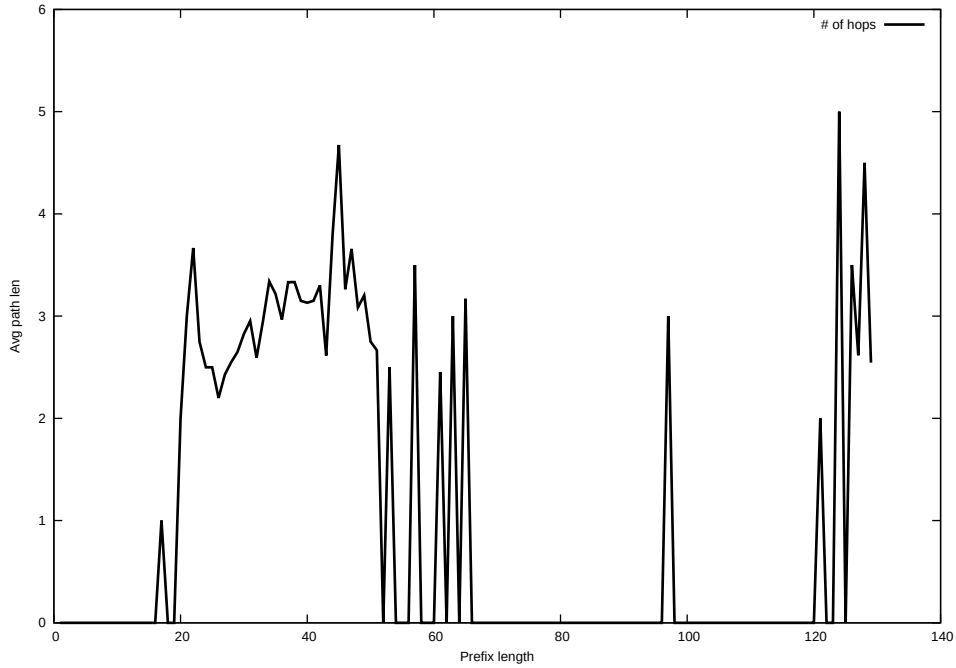
2014-11-05



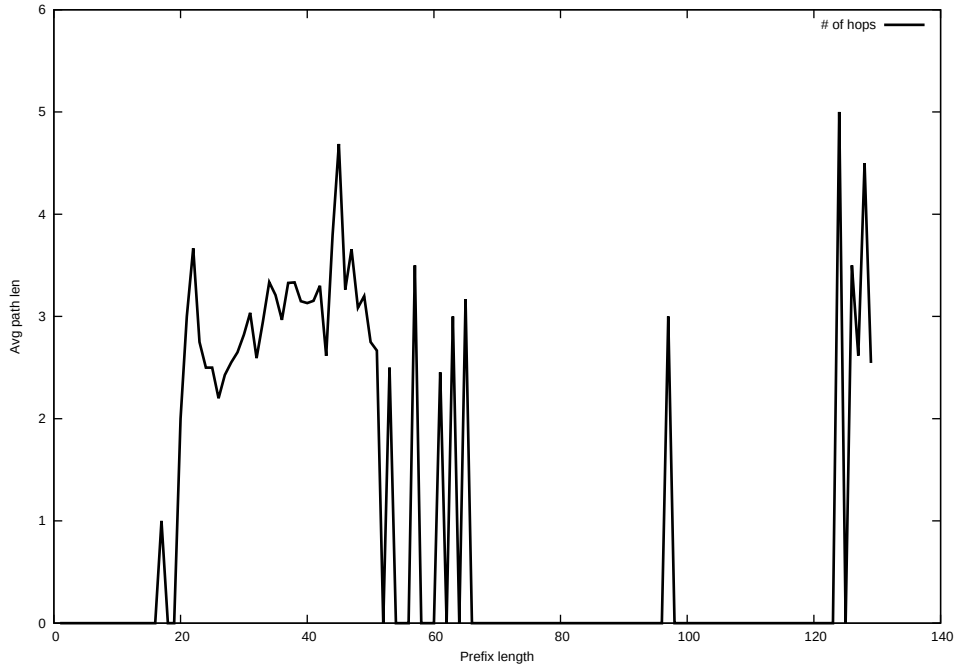
2014-11-06



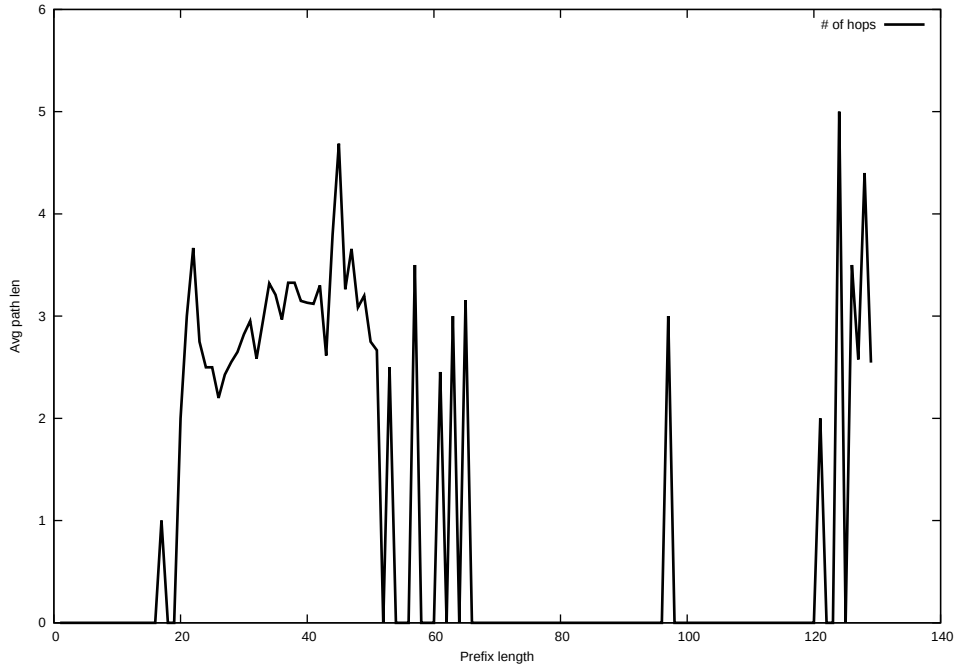
2014-11-07



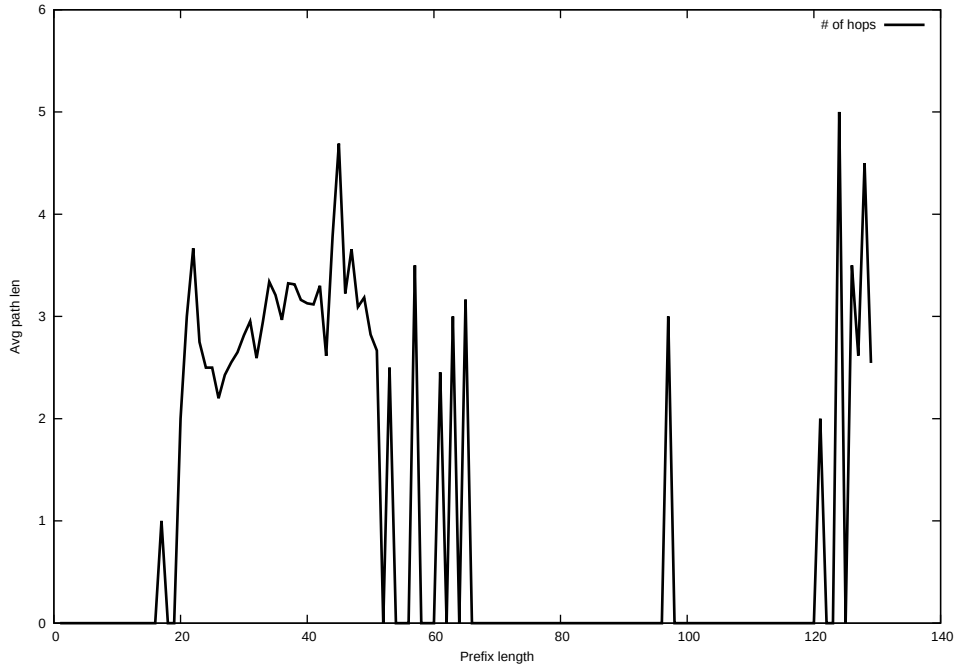
2014-11-08



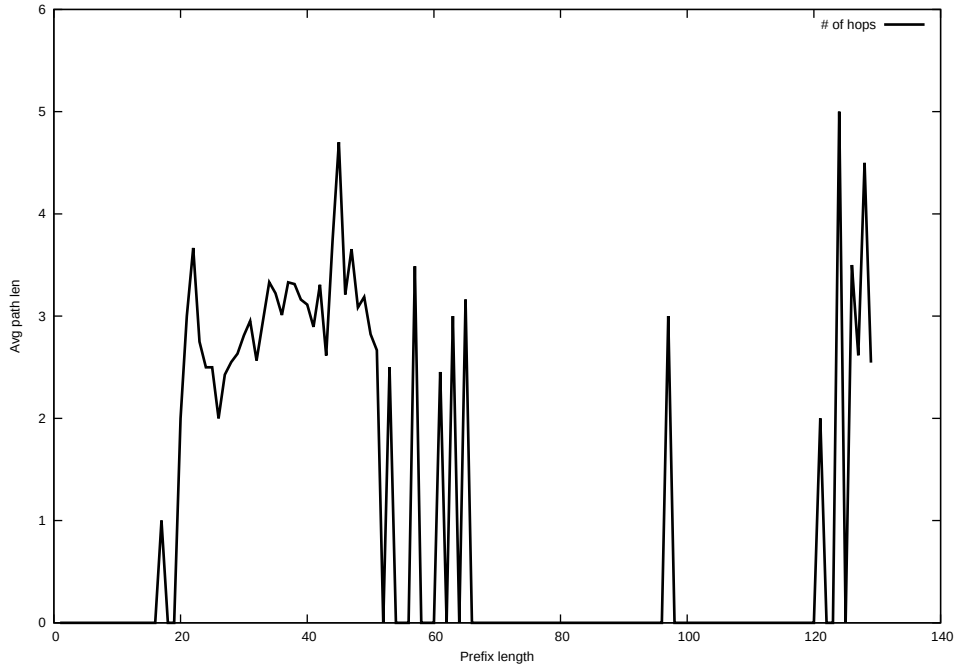
2014-11-09



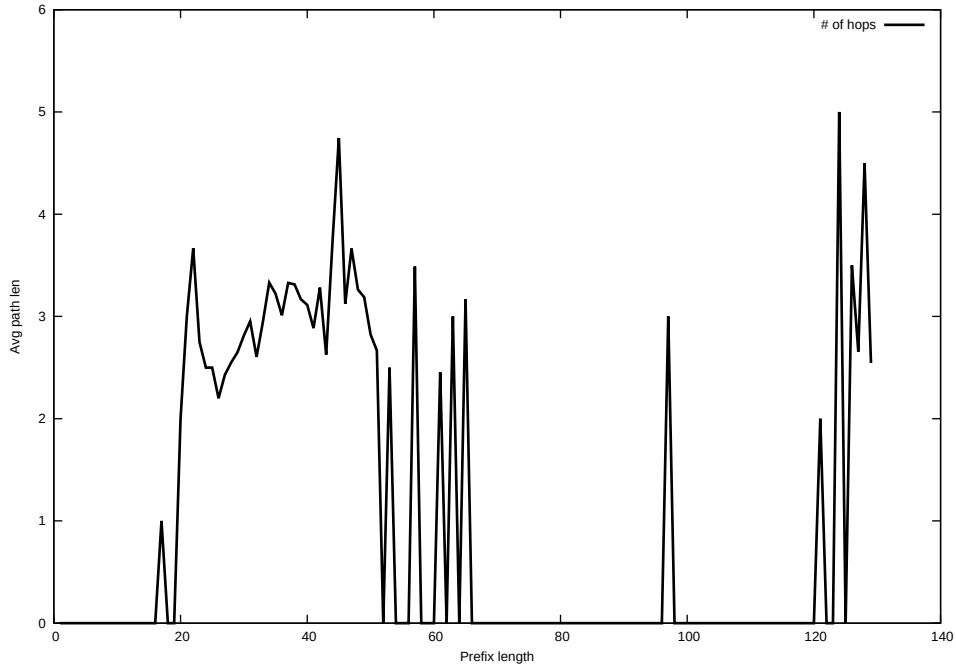
2014-11-10



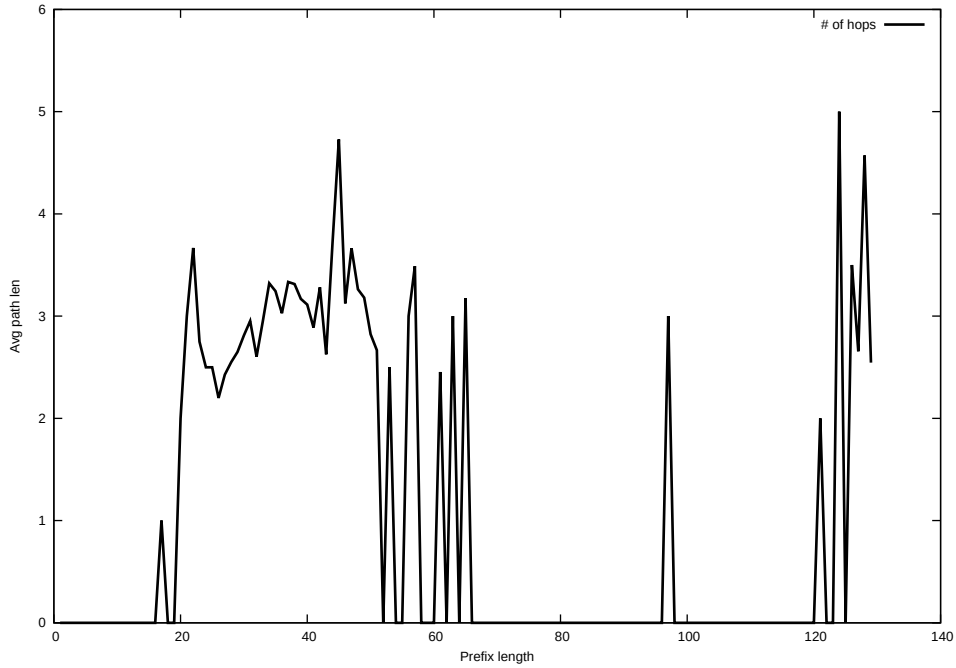
2014-11-11



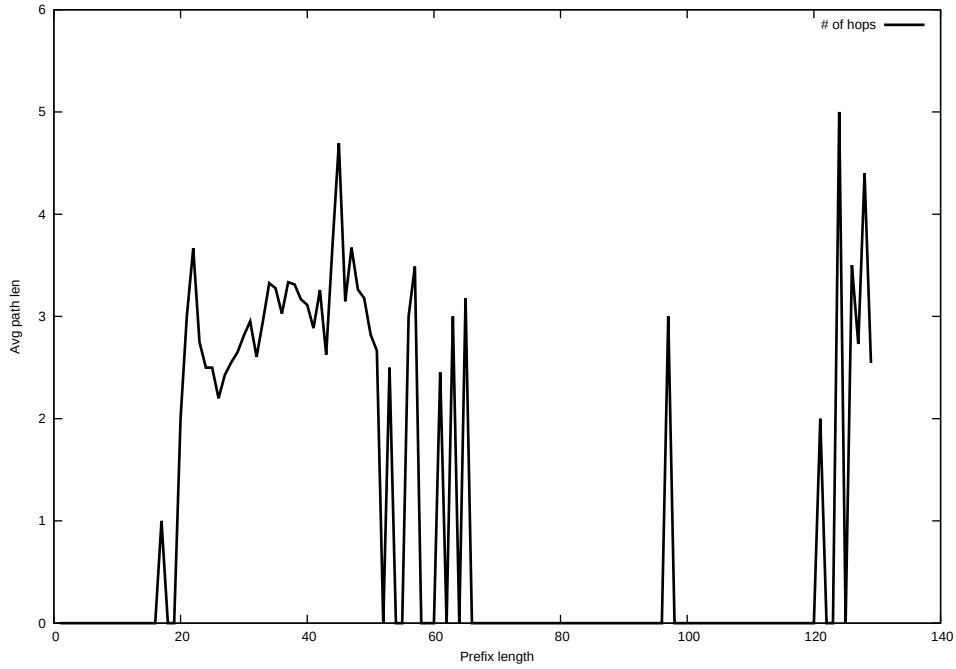
2014-11-12



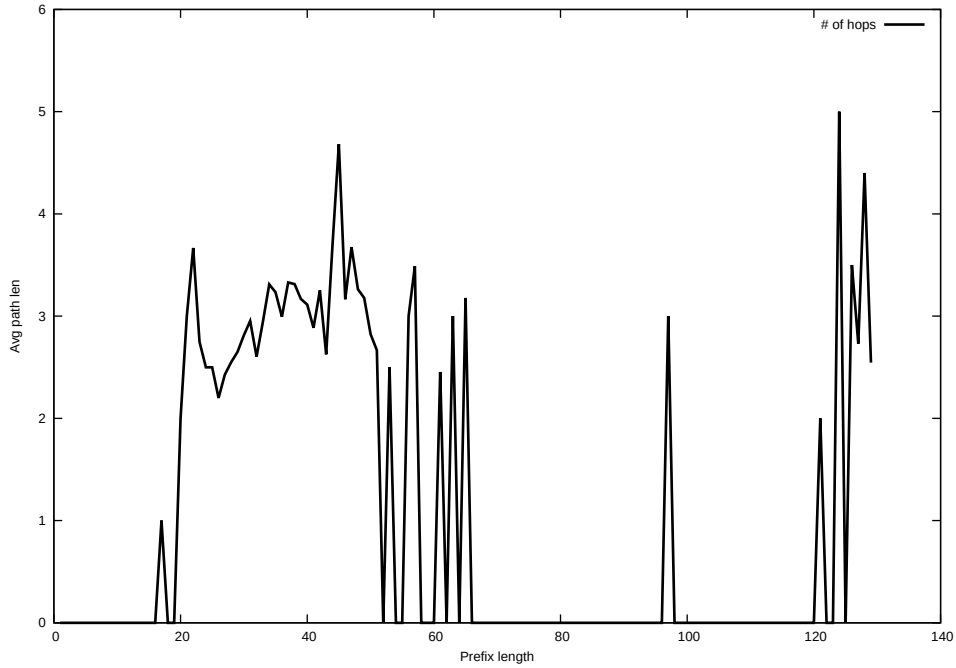
2014-11-13



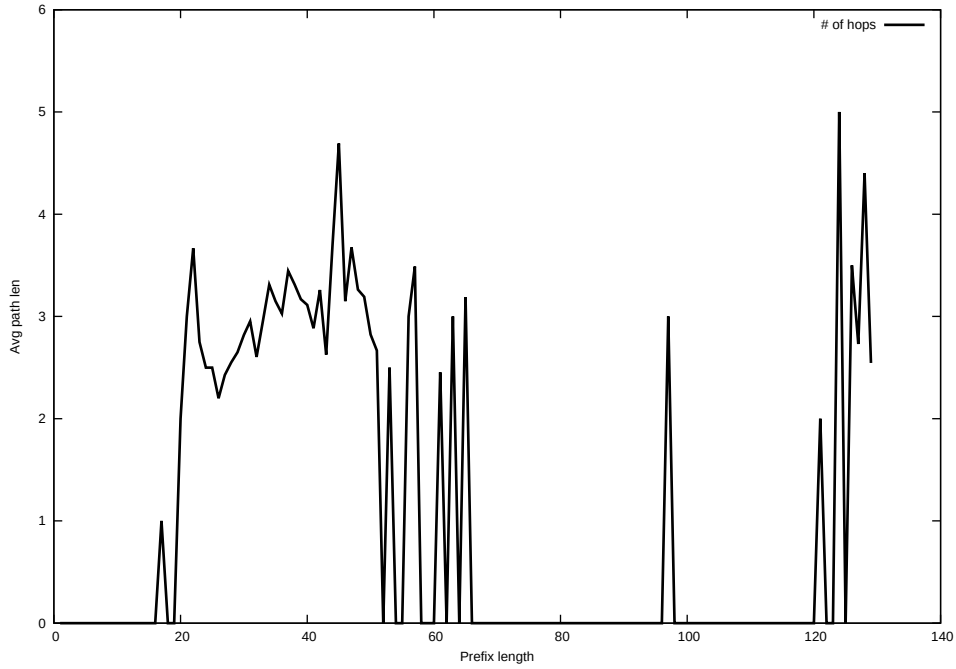
2014-11-14



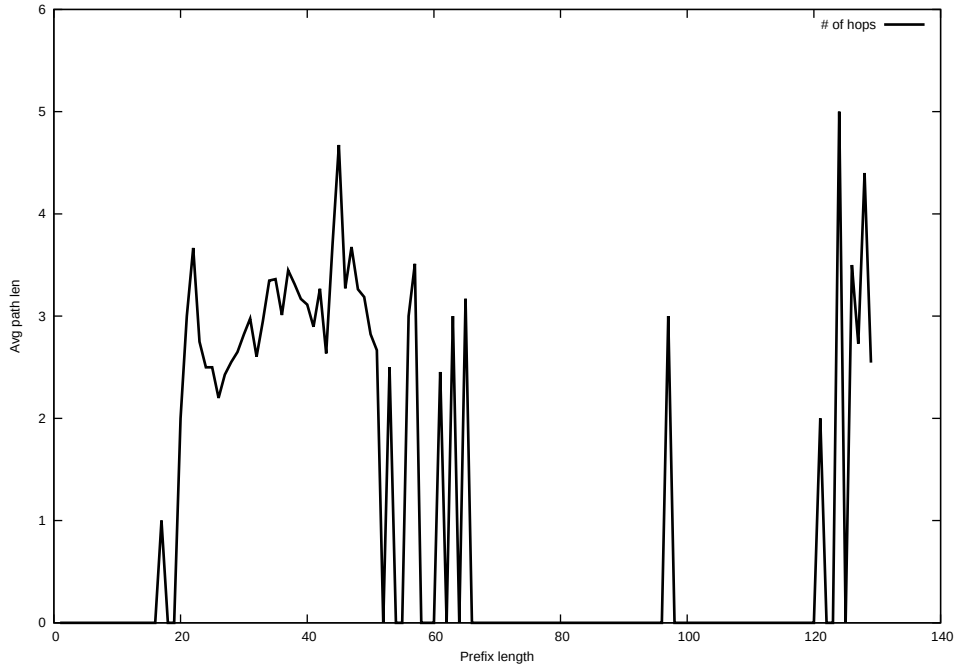
2014-11-15



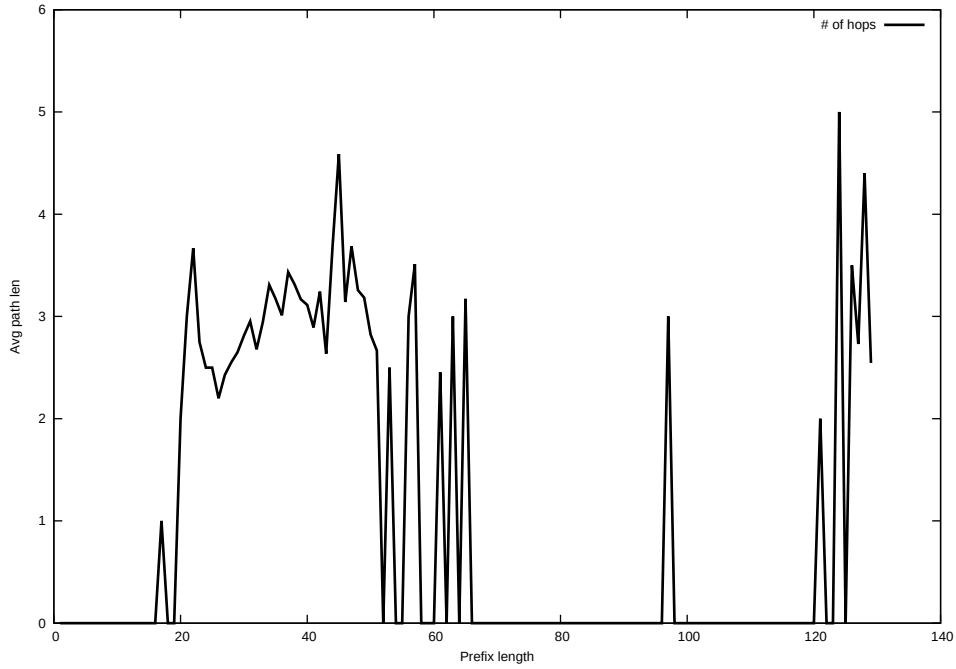
2014-11-16



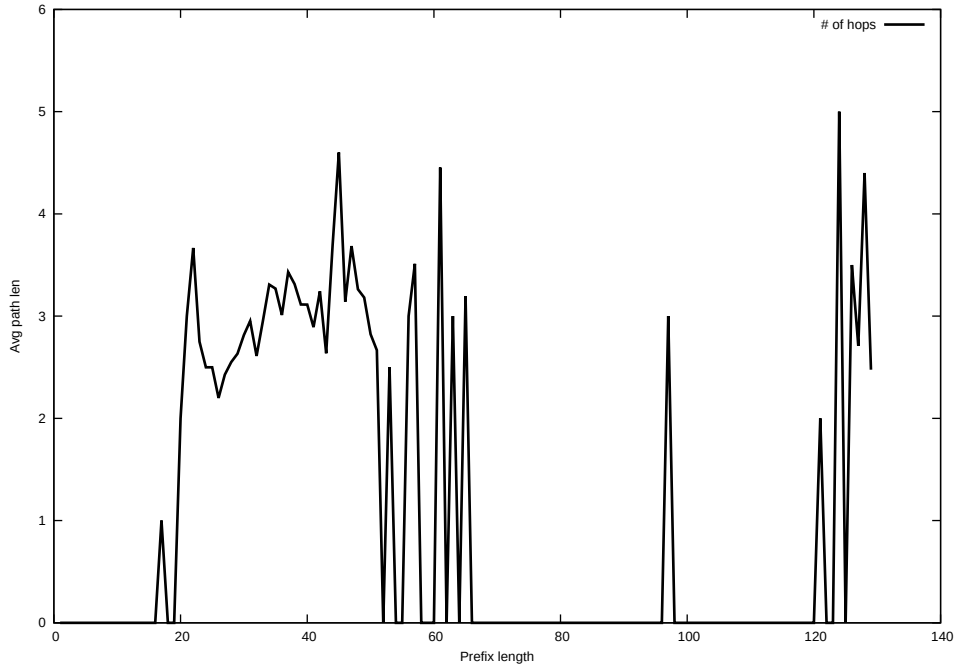
2014-11-17



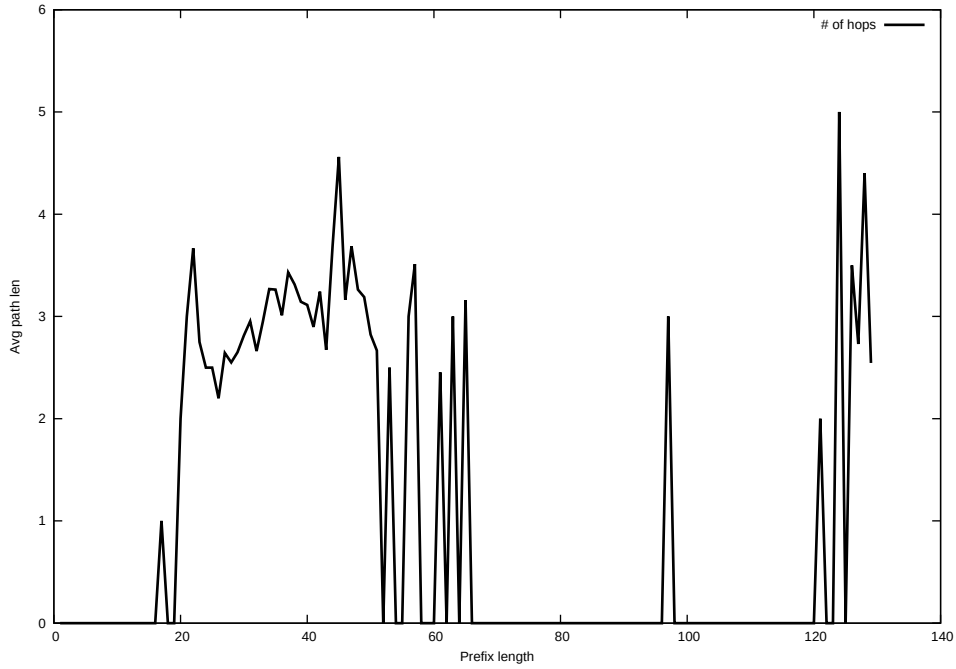
2014-11-18



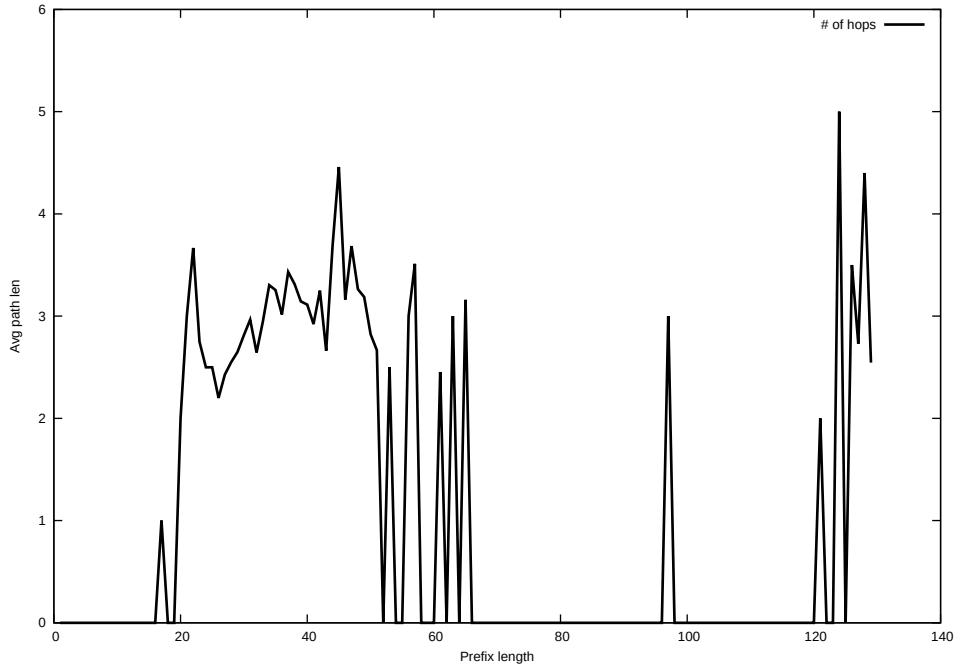
2014-11-19



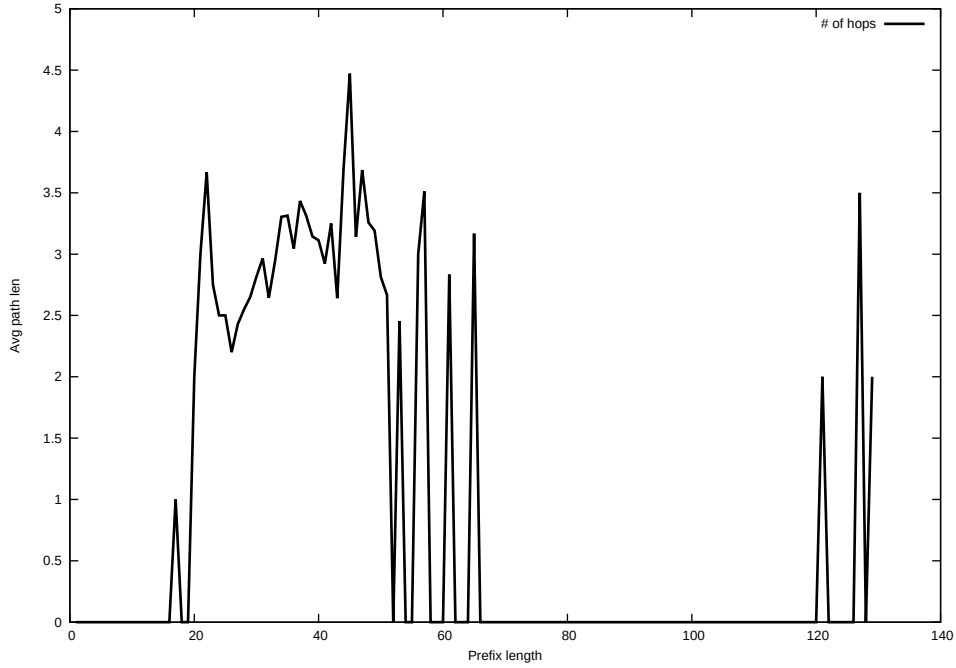
2014-11-20



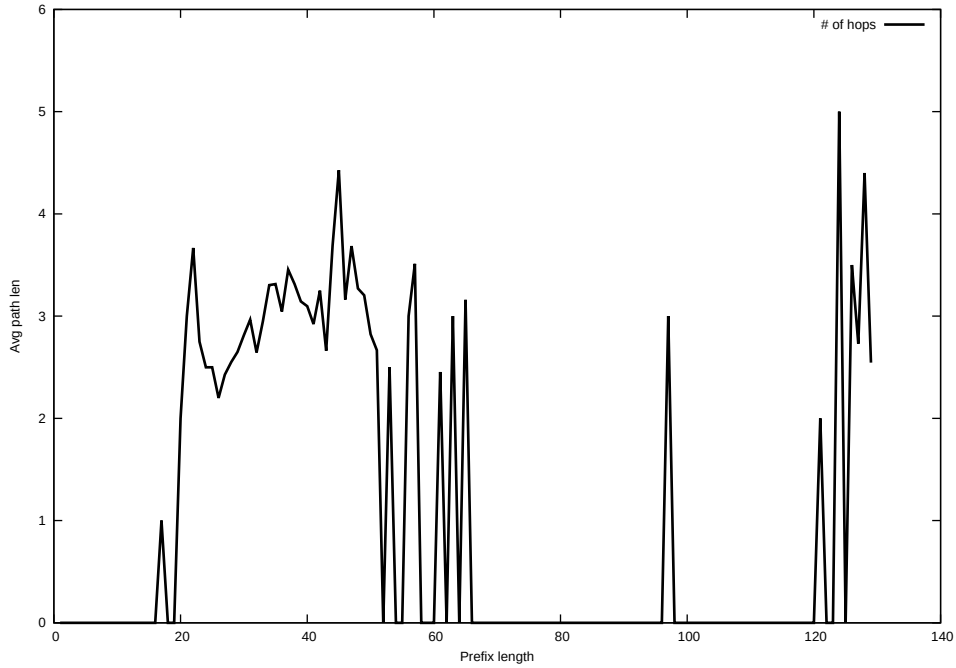
2014-11-21



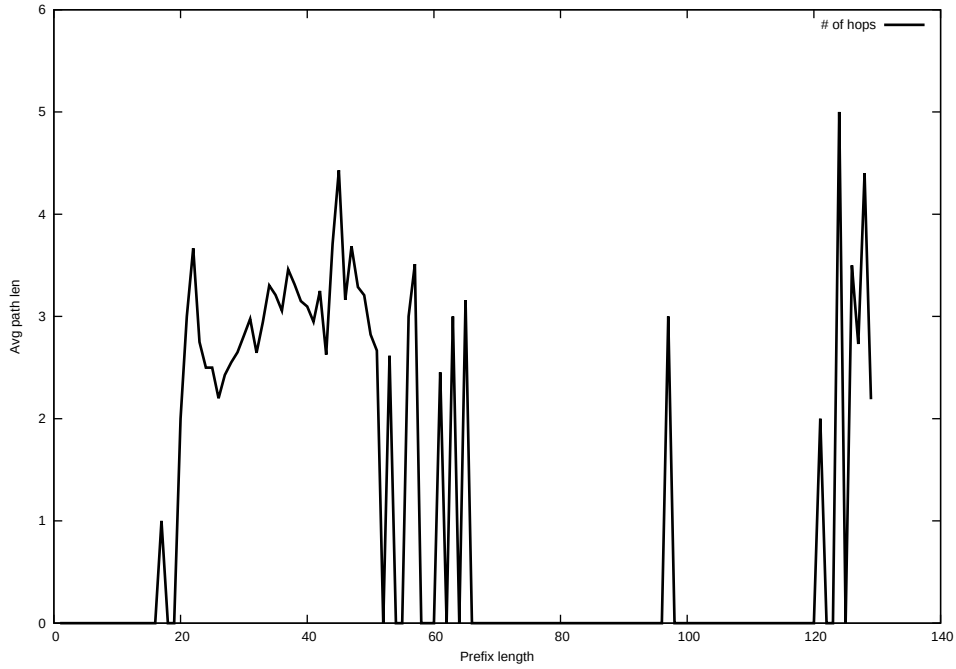
2014-11-22



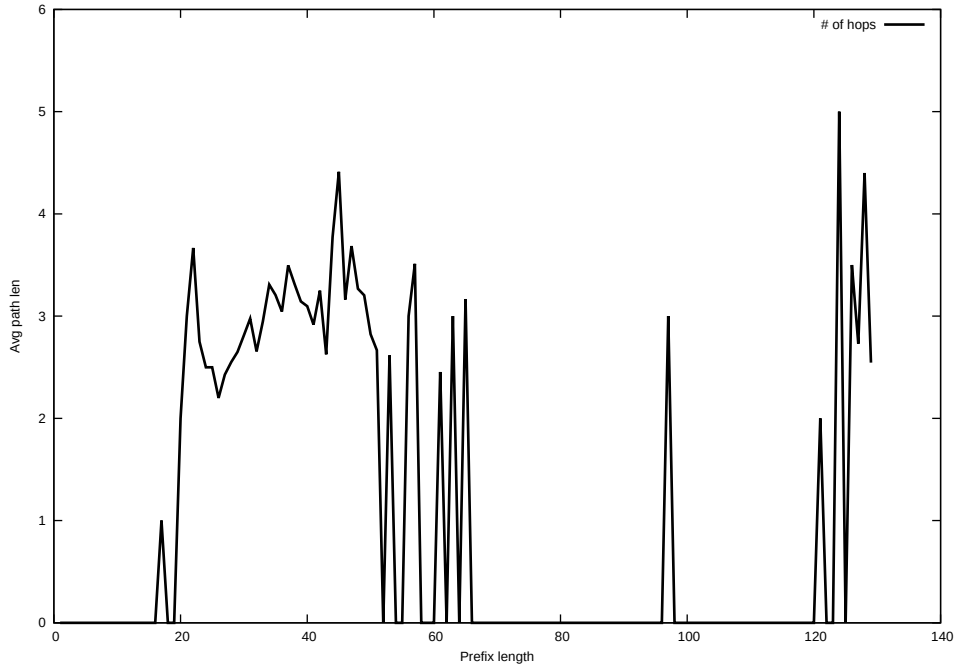
2014-11-23



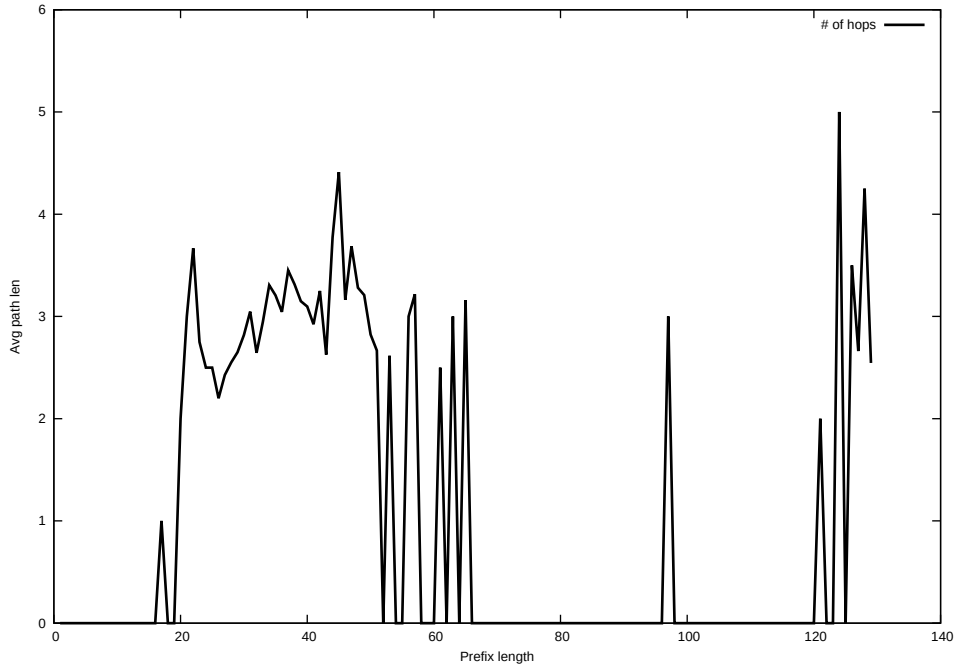
2014-11-24



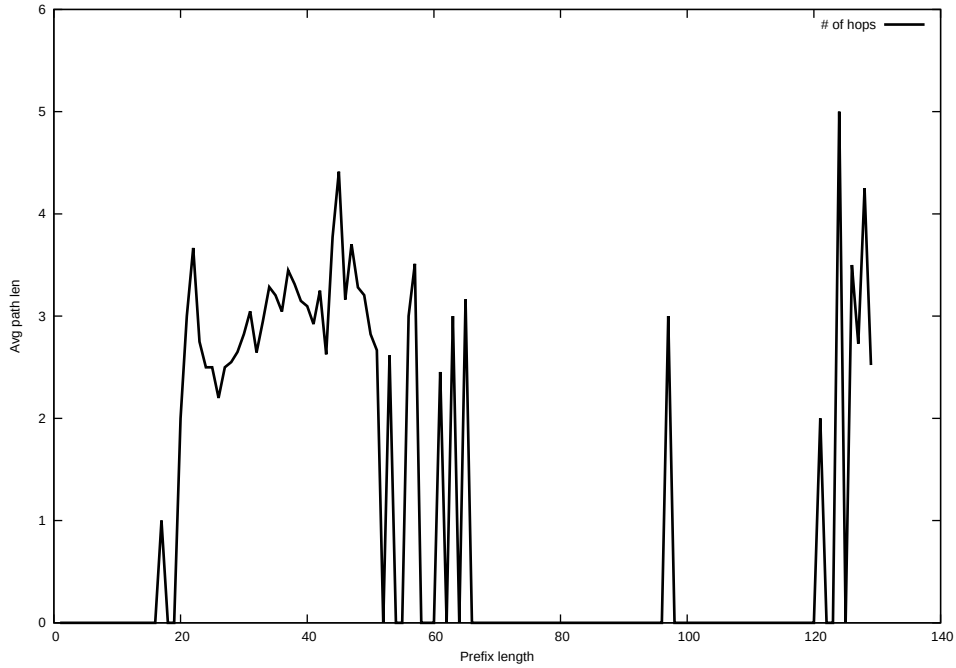
2014-11-25



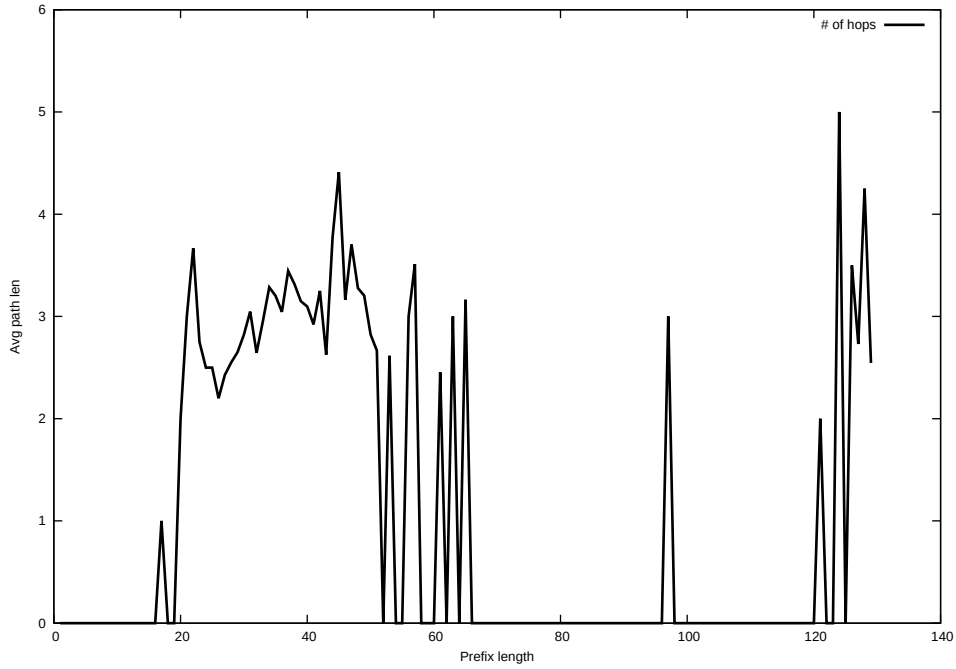
2014-11-26



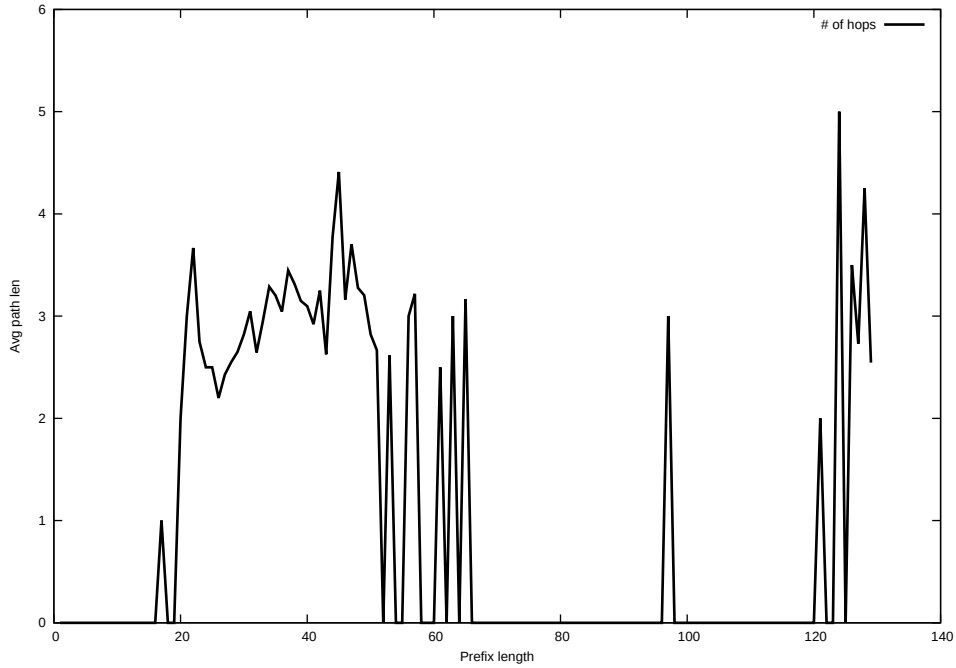
2014-11-27



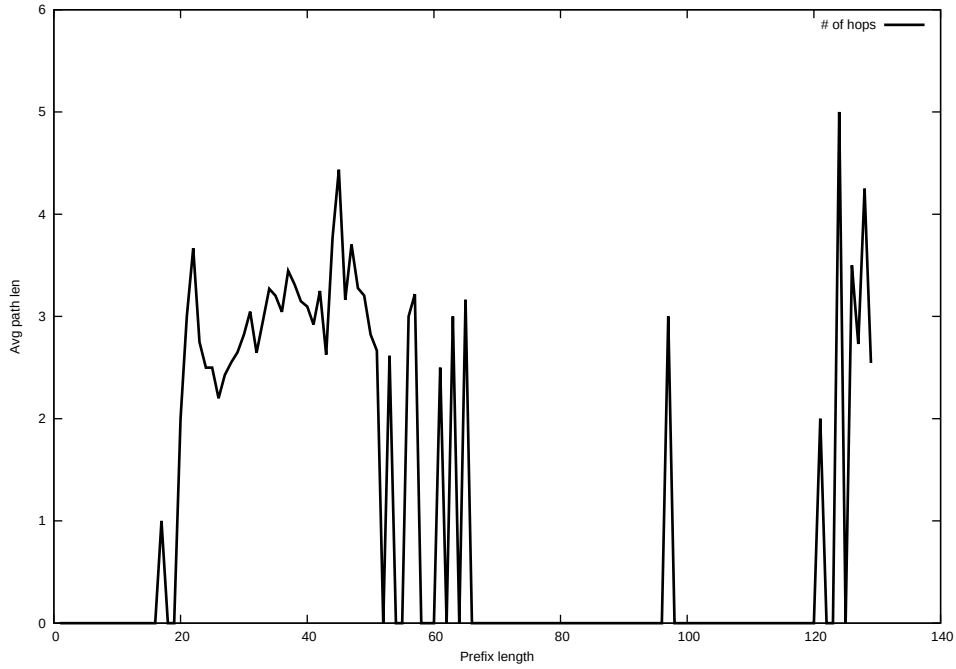
2014-11-28



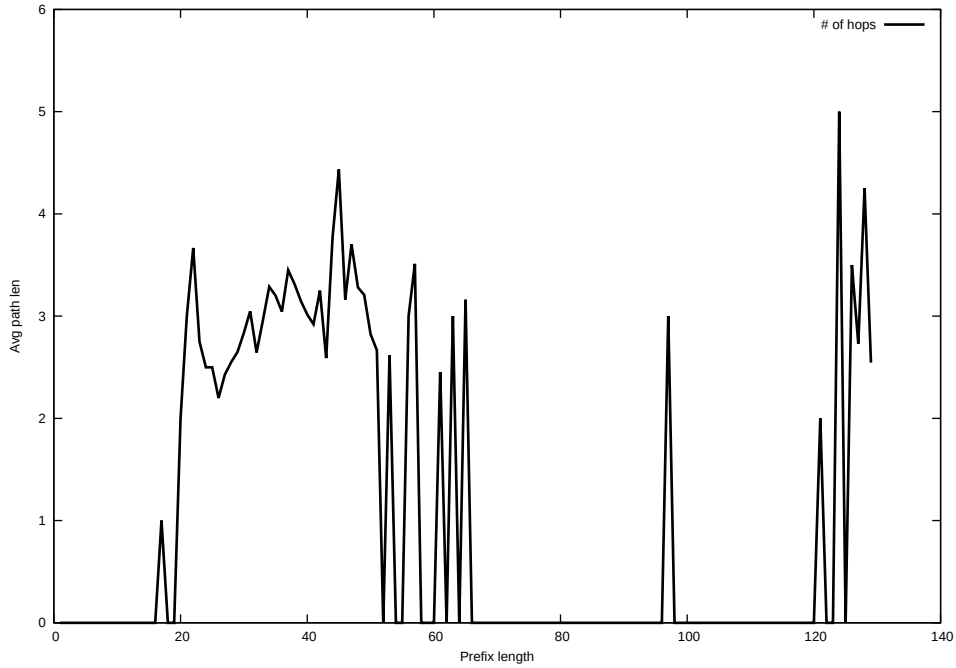
2014-11-29



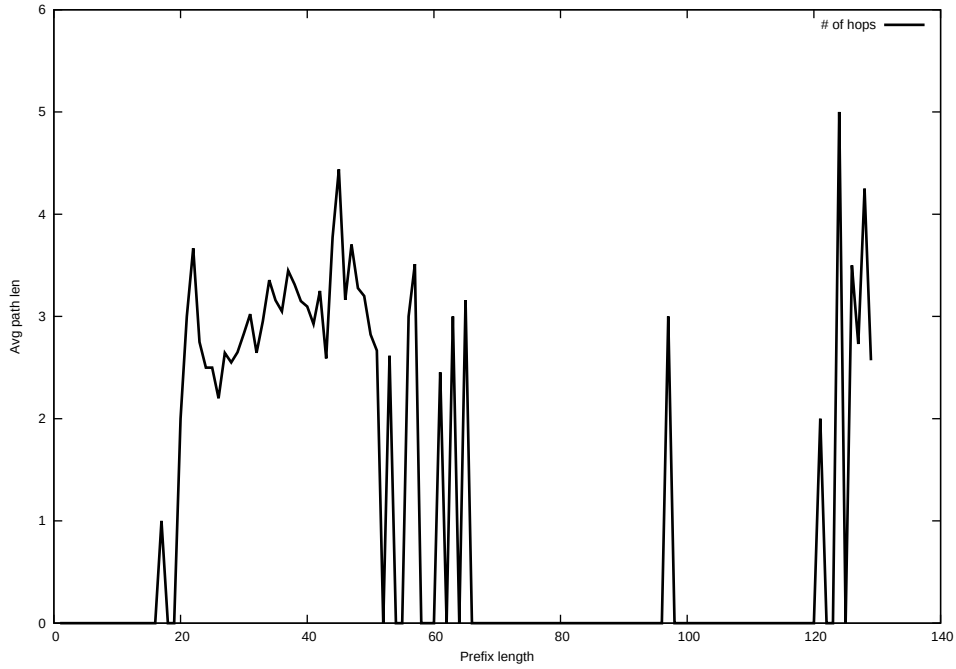
2014-11-30



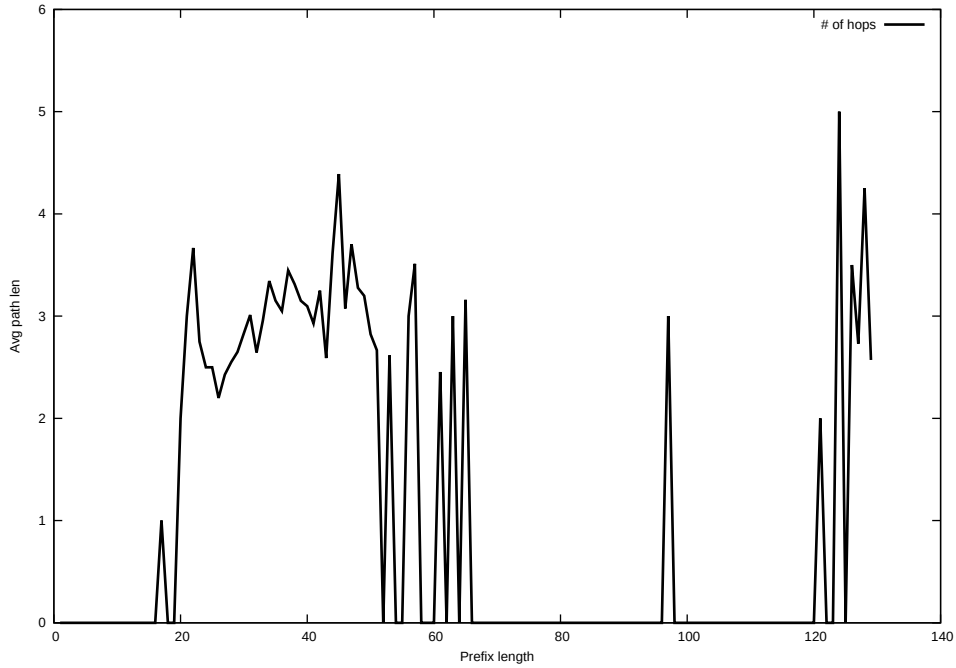
2014-12-01



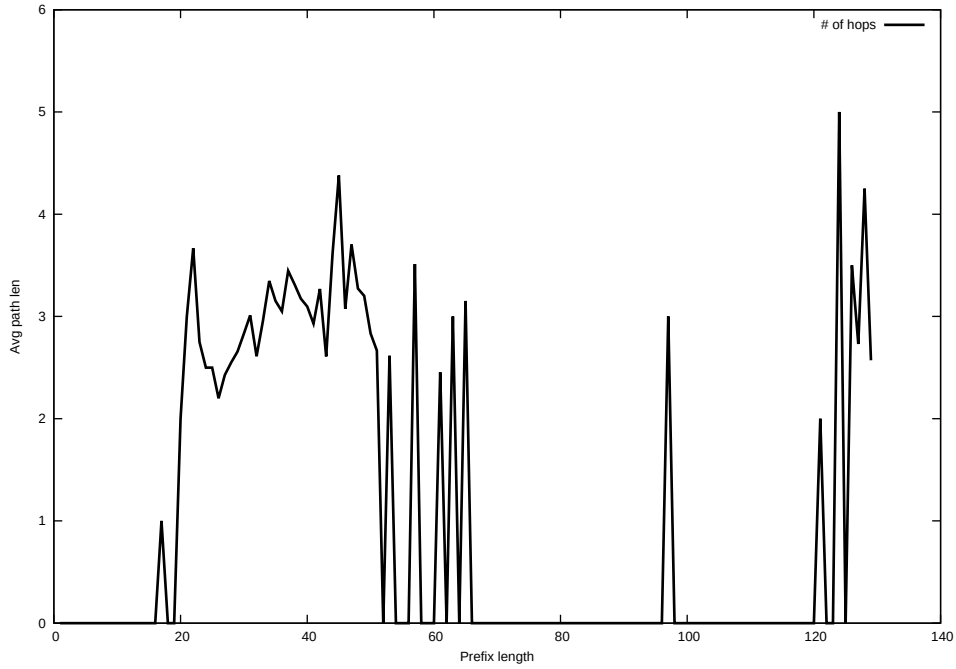
2014-12-02



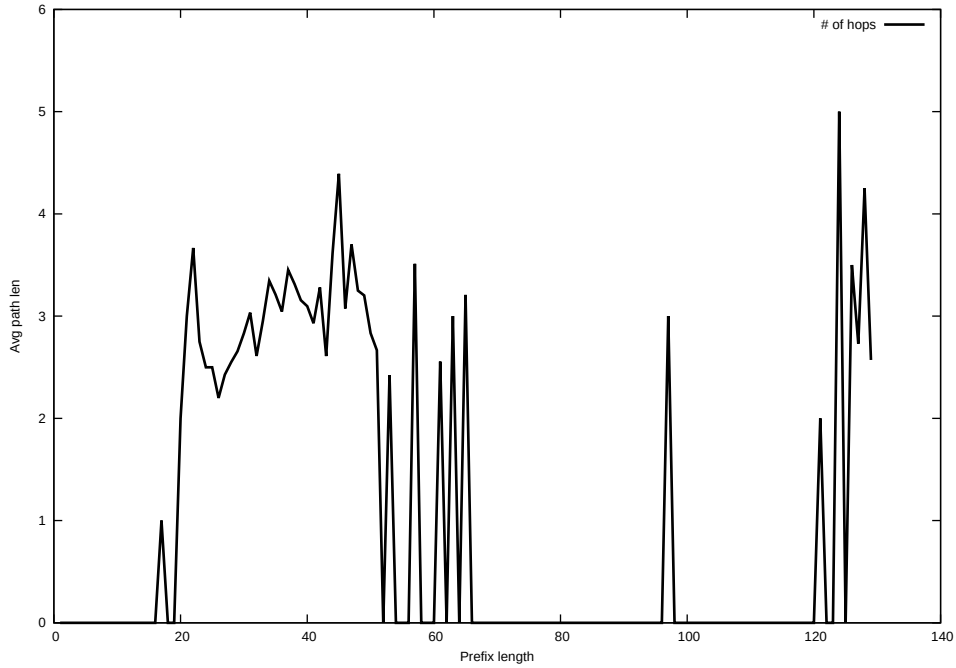
2014-12-03



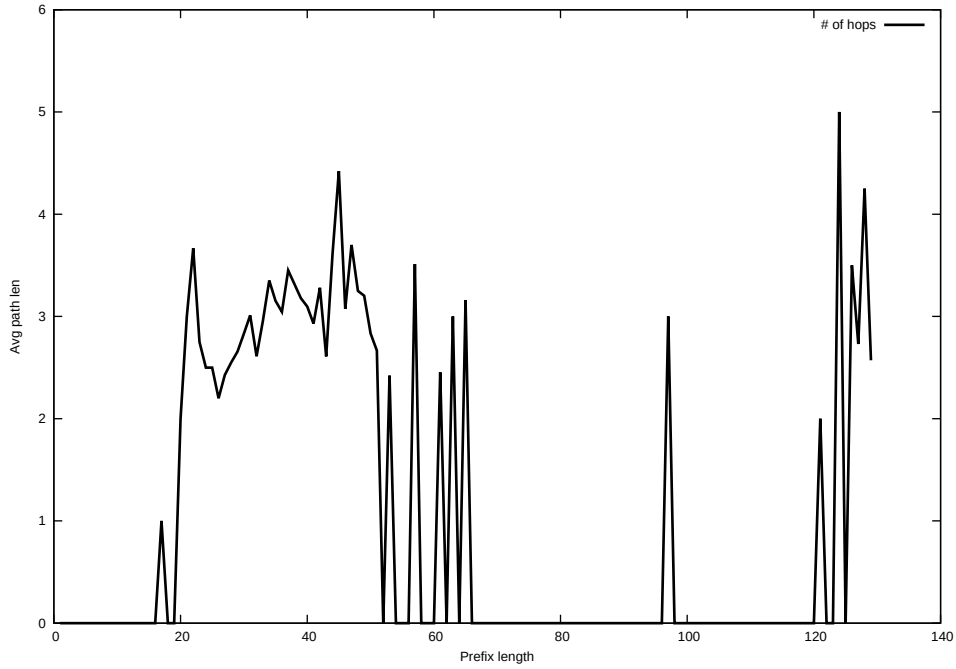
2014-12-04



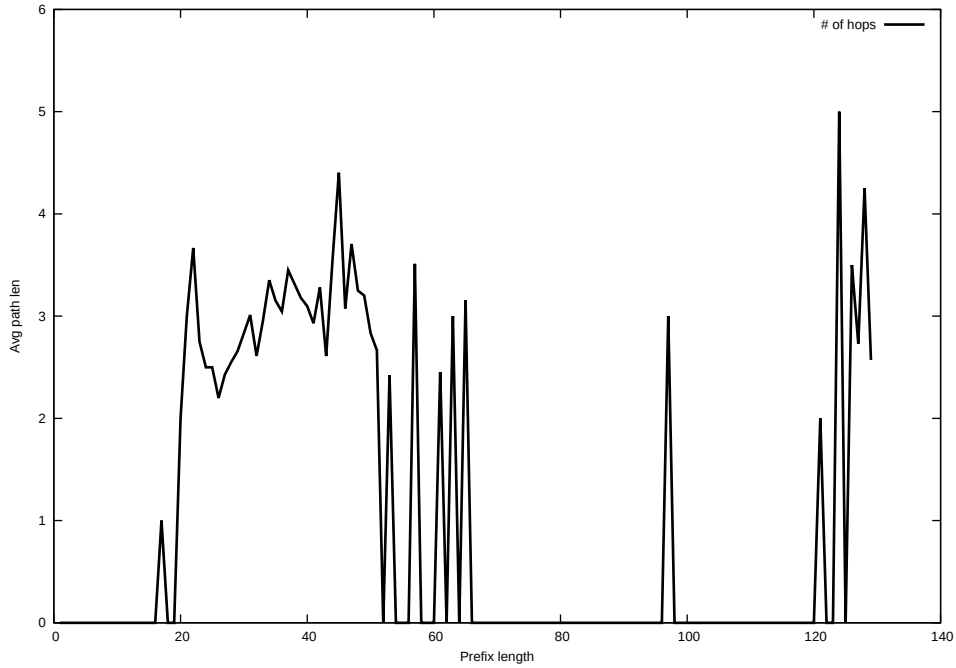
2014-12-05



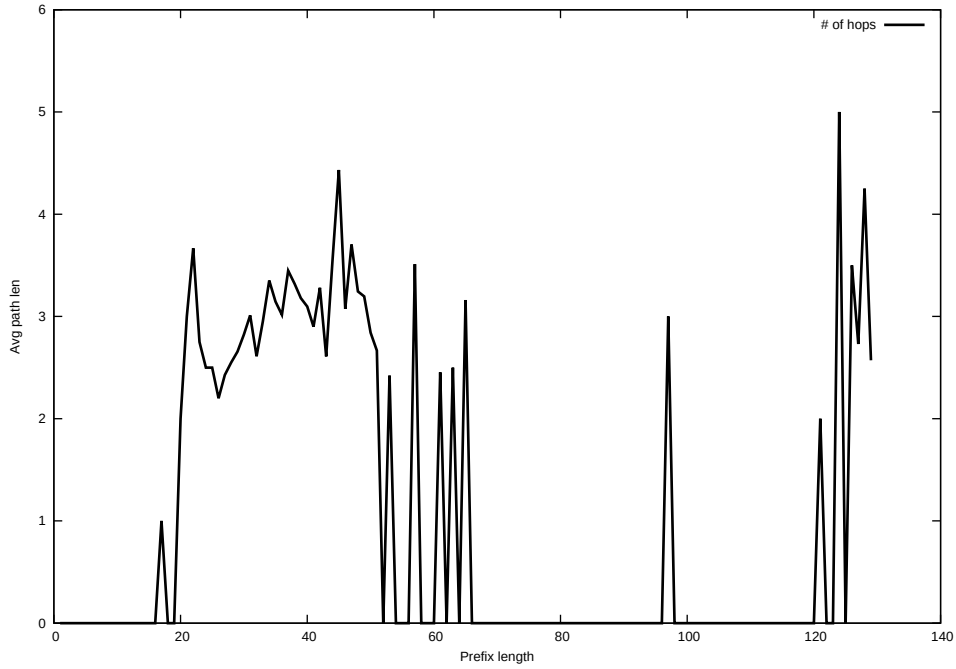
2014-12-06



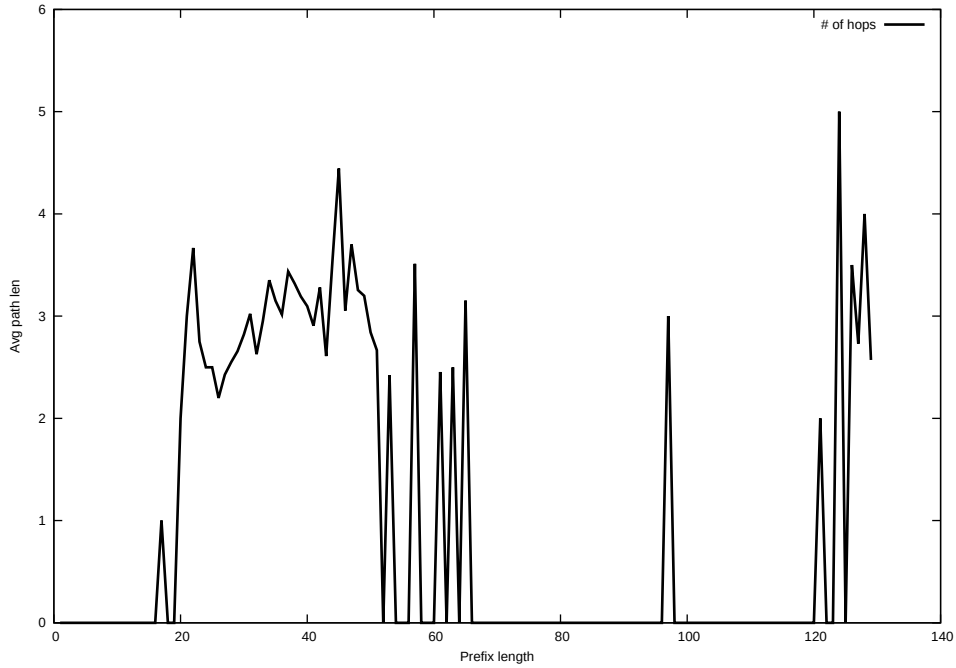
2014-12-07



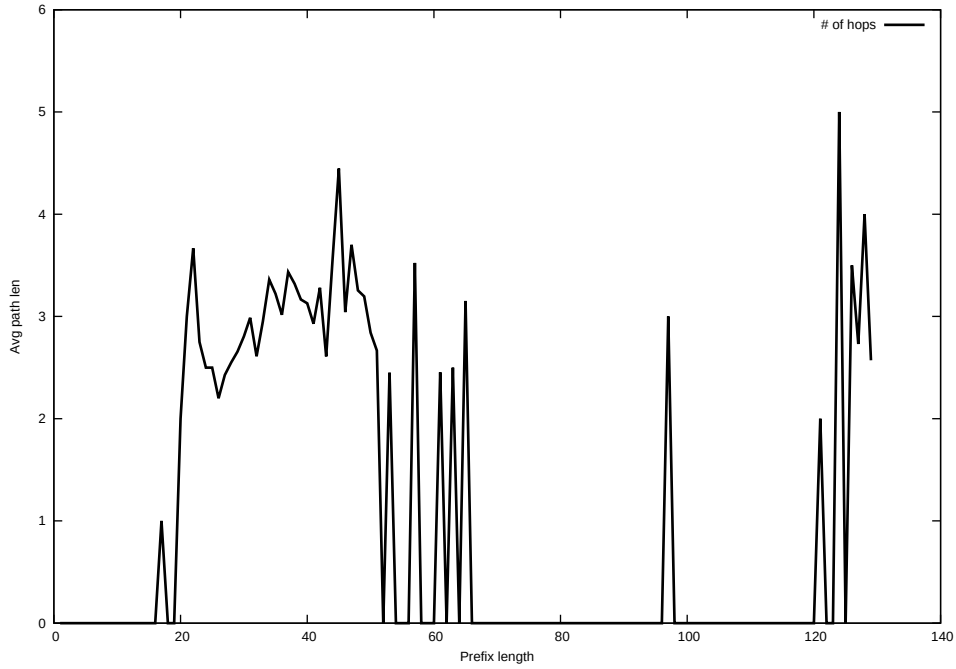
2014-12-08



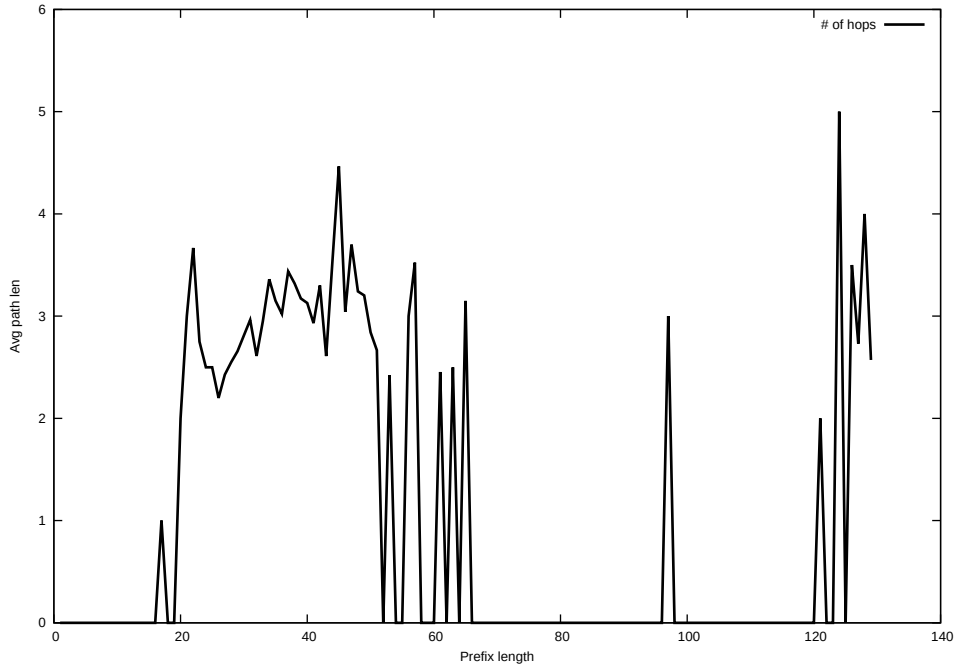
2014-12-09



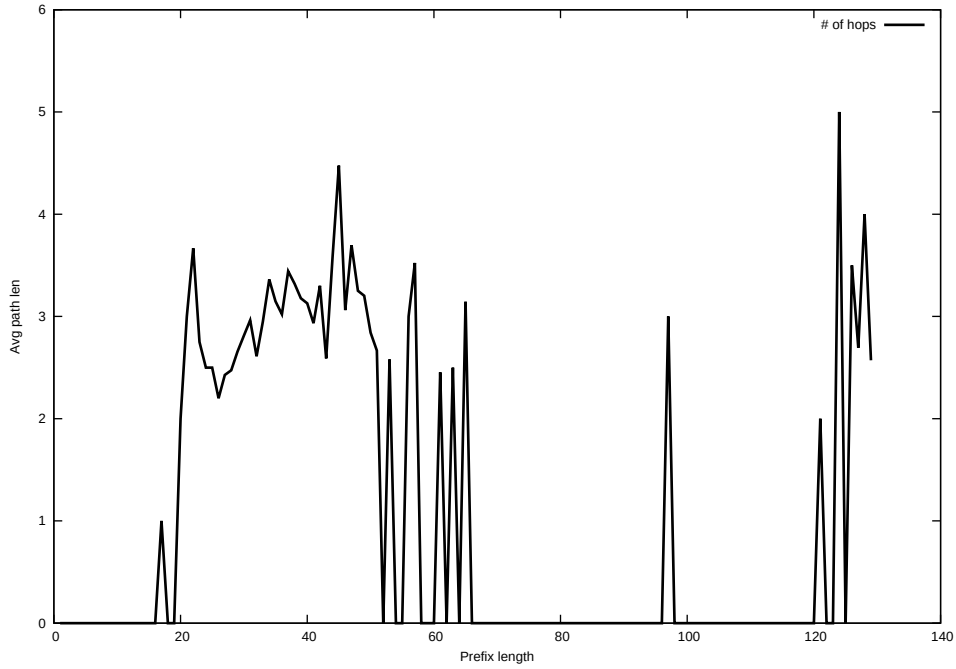
2014-12-10



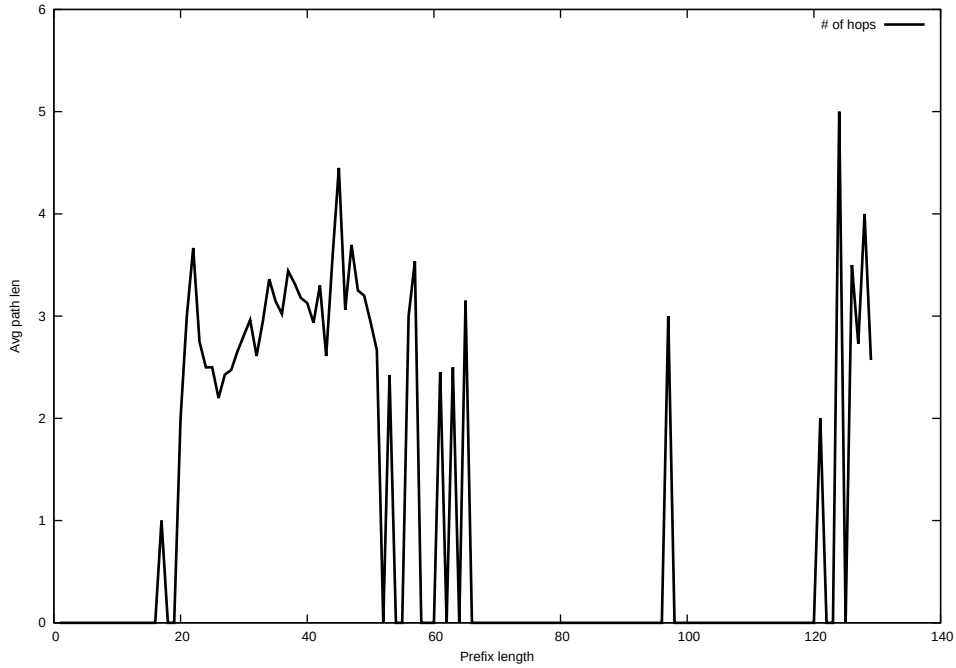
2014-12-11



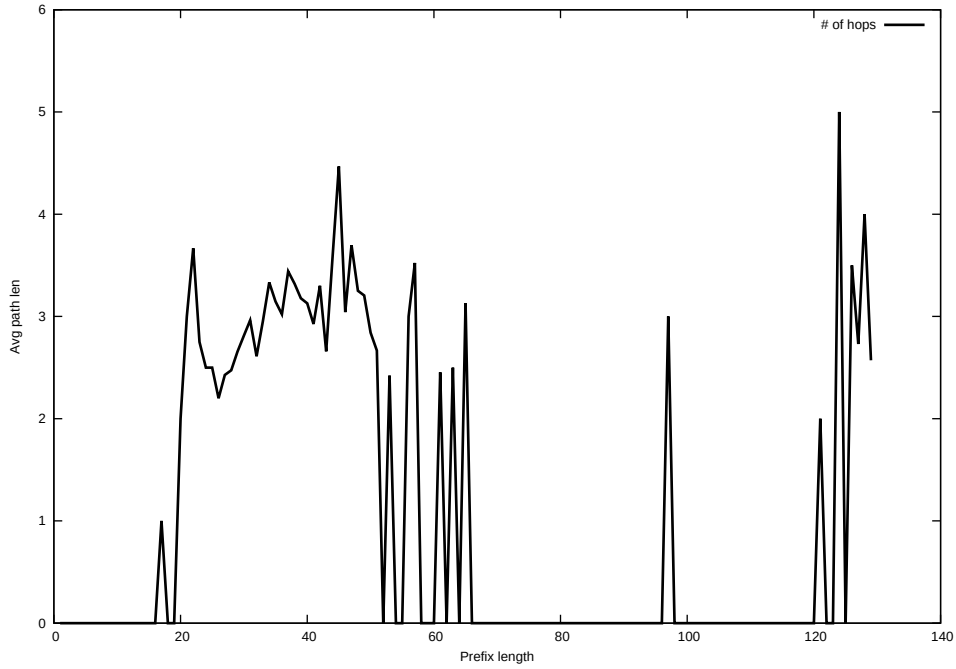
2014-12-12



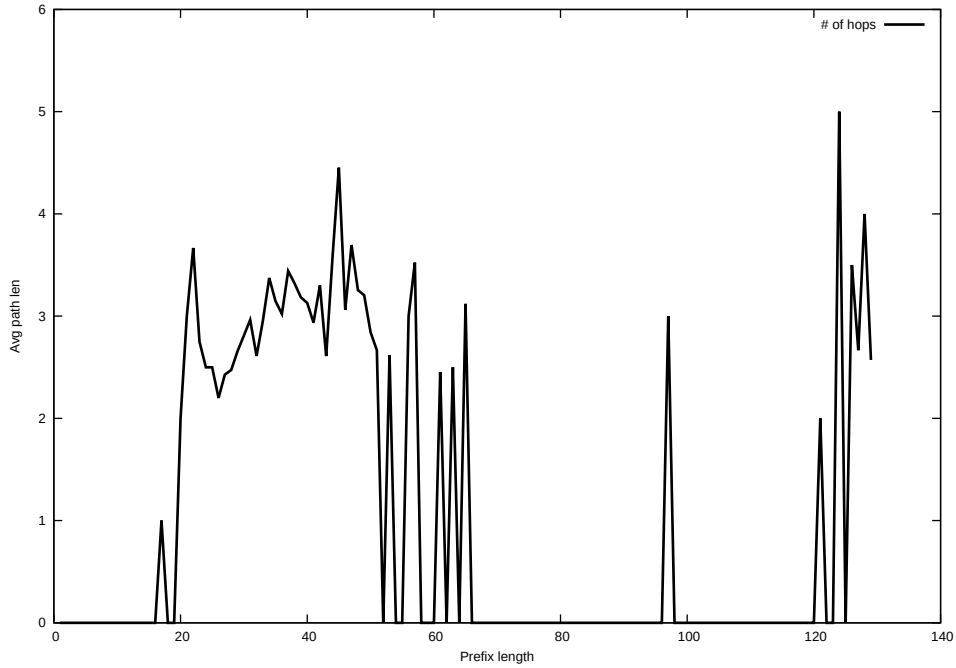
2014-12-13



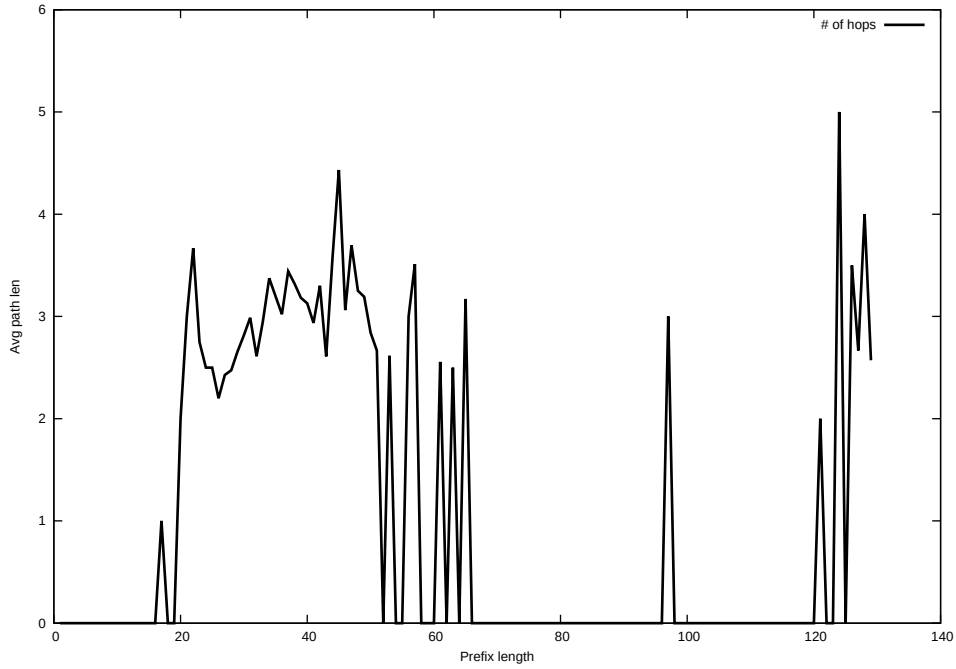
2014-12-14



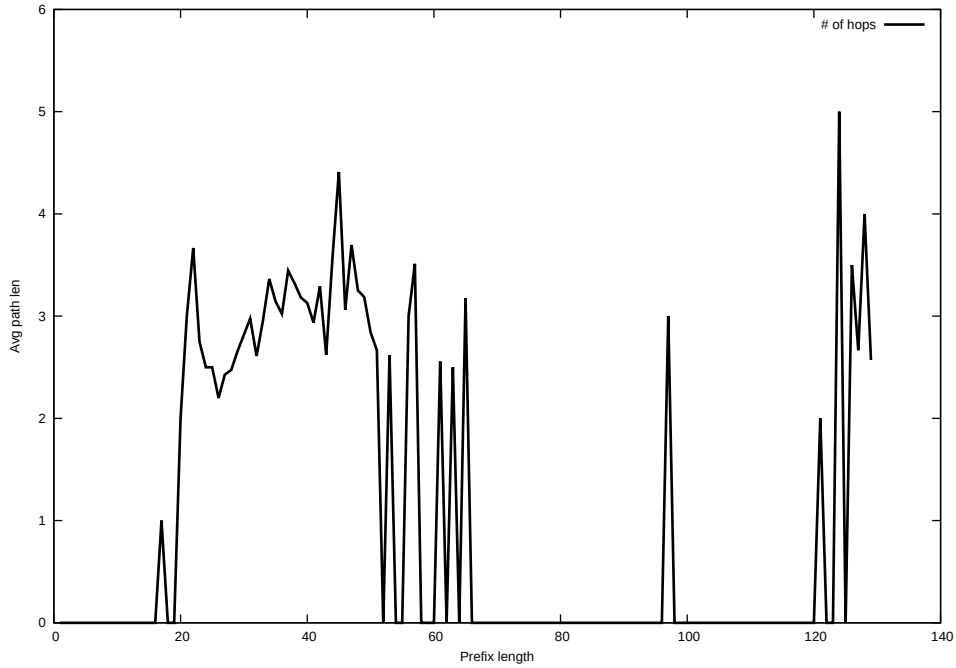
2014-12-15



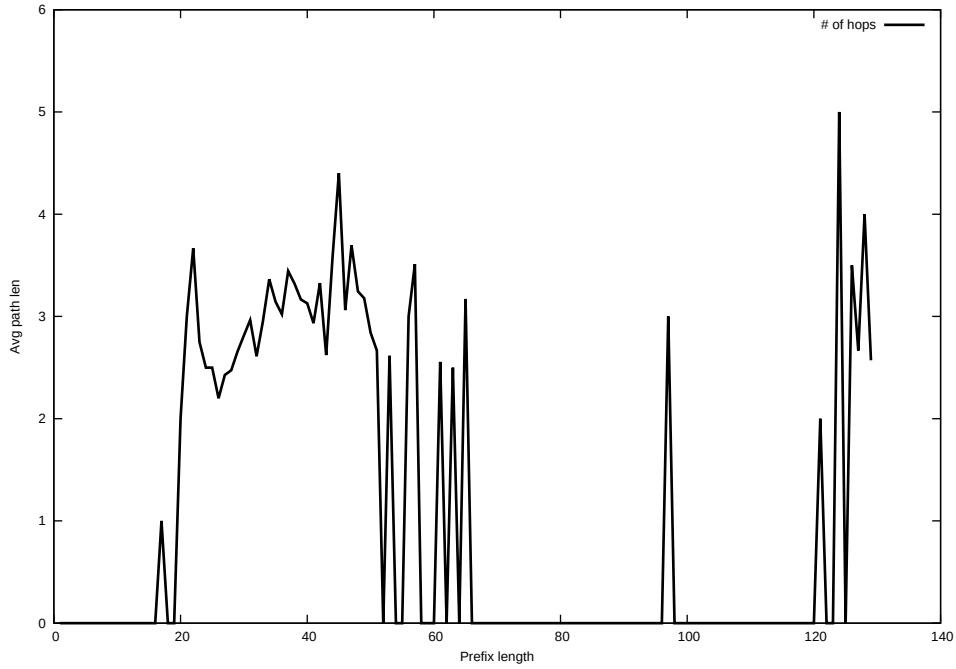
2014-12-16



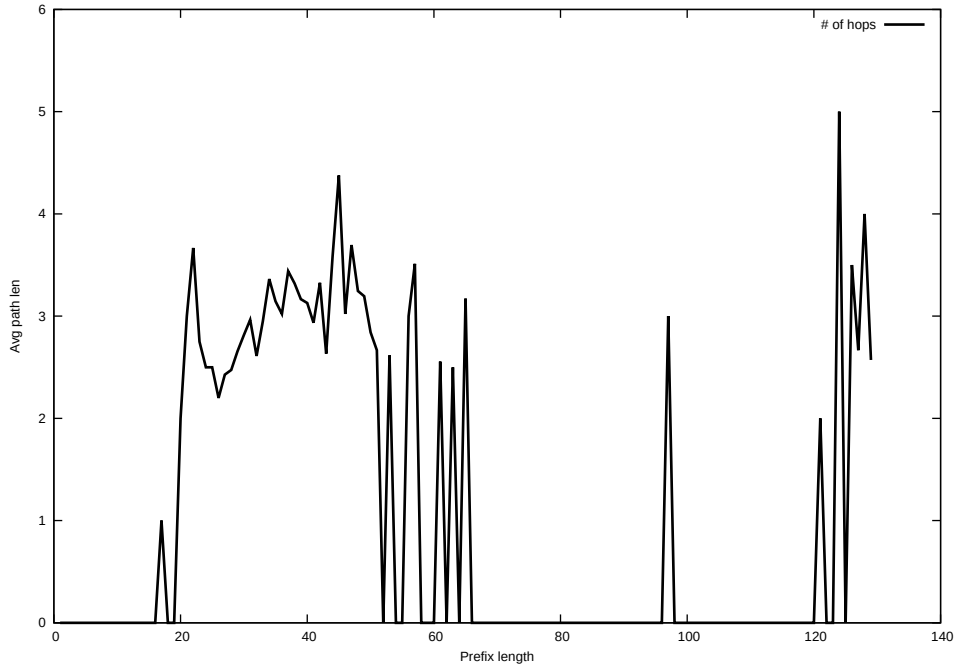
2014-12-17



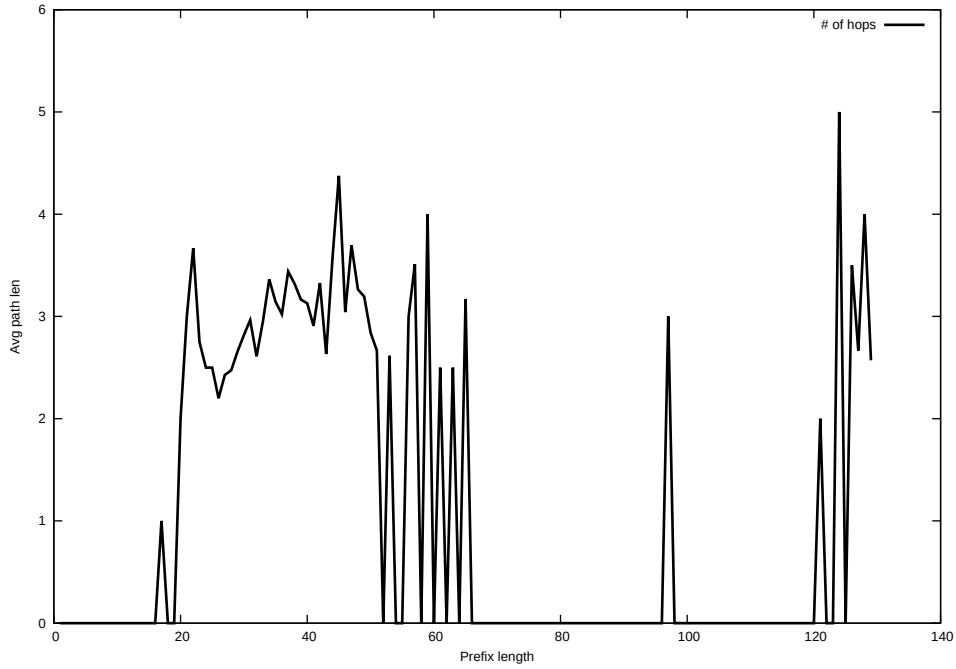
2014-12-18



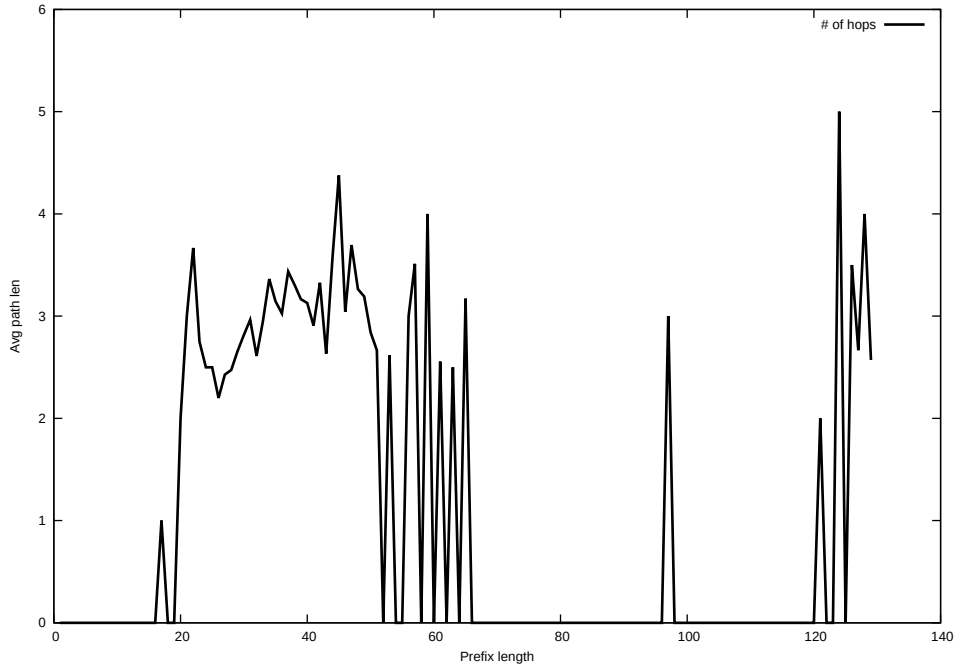
2014-12-19



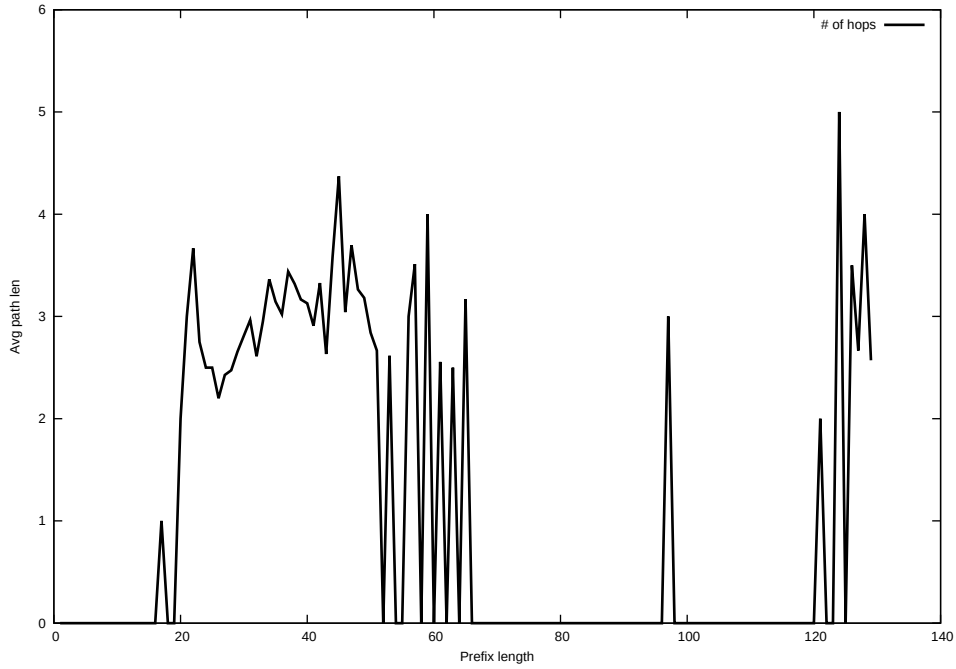
2014-12-20



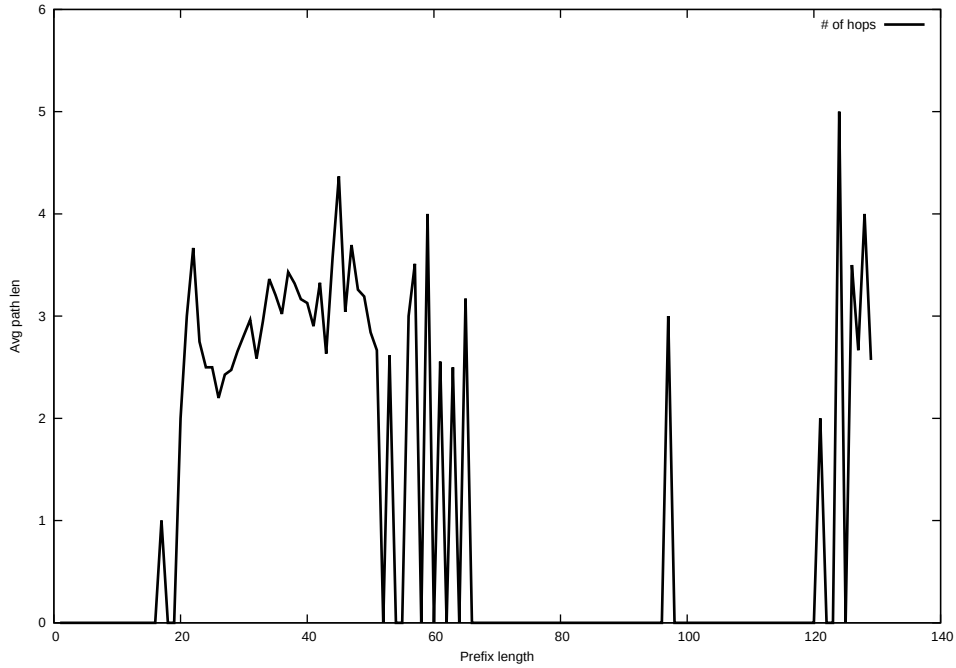
2014-12-21



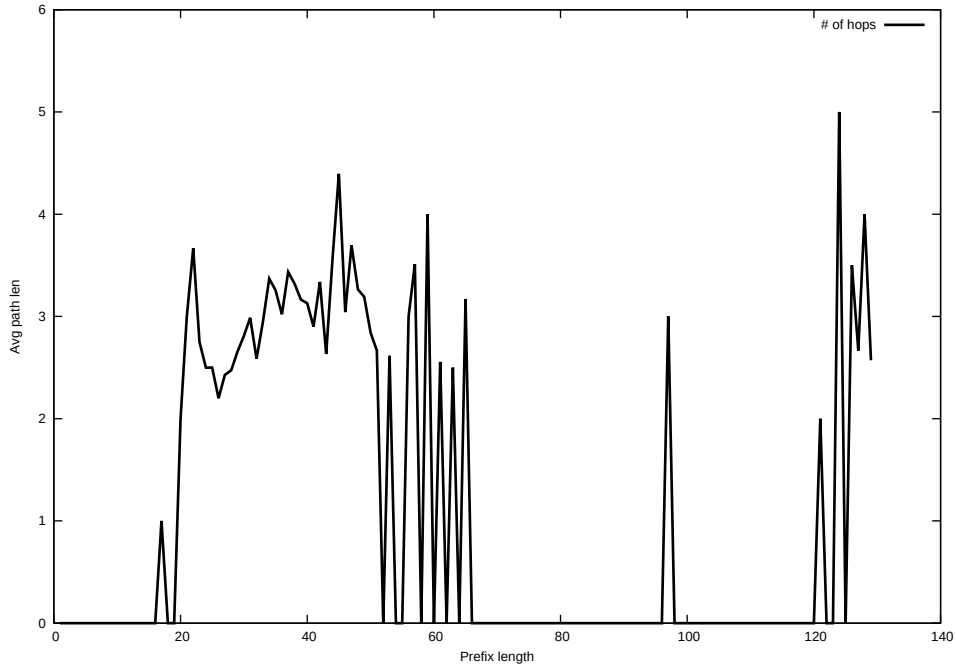
2014-12-22



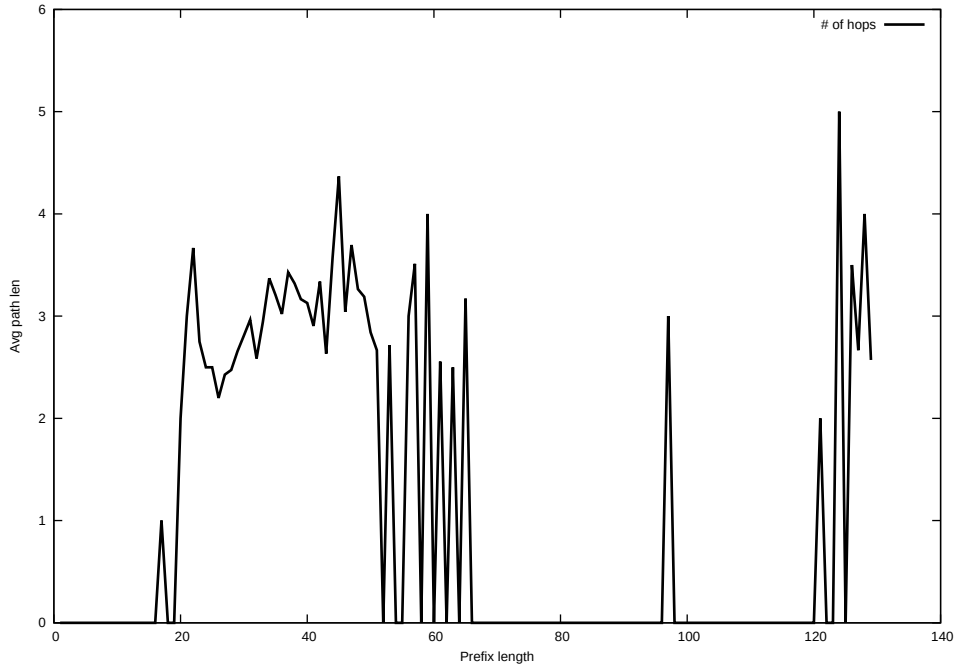
2014-12-23



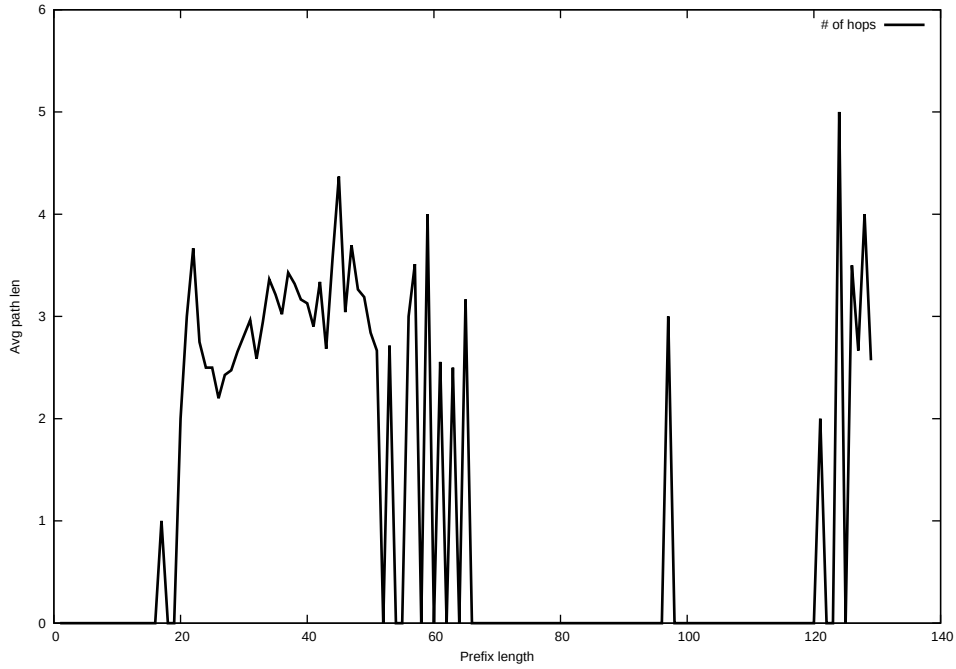
2014-12-24



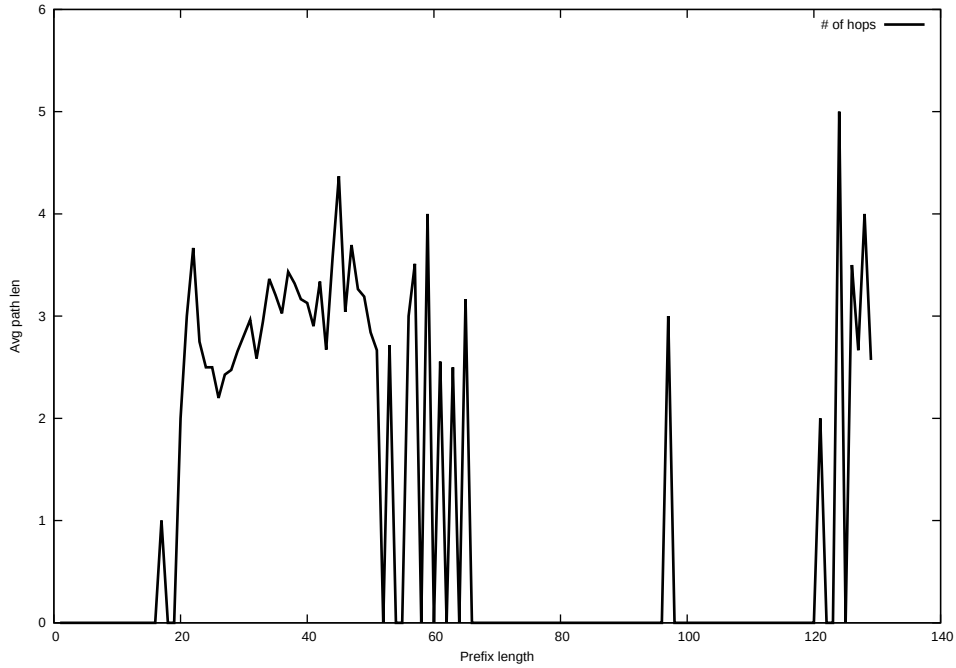
2014-12-25



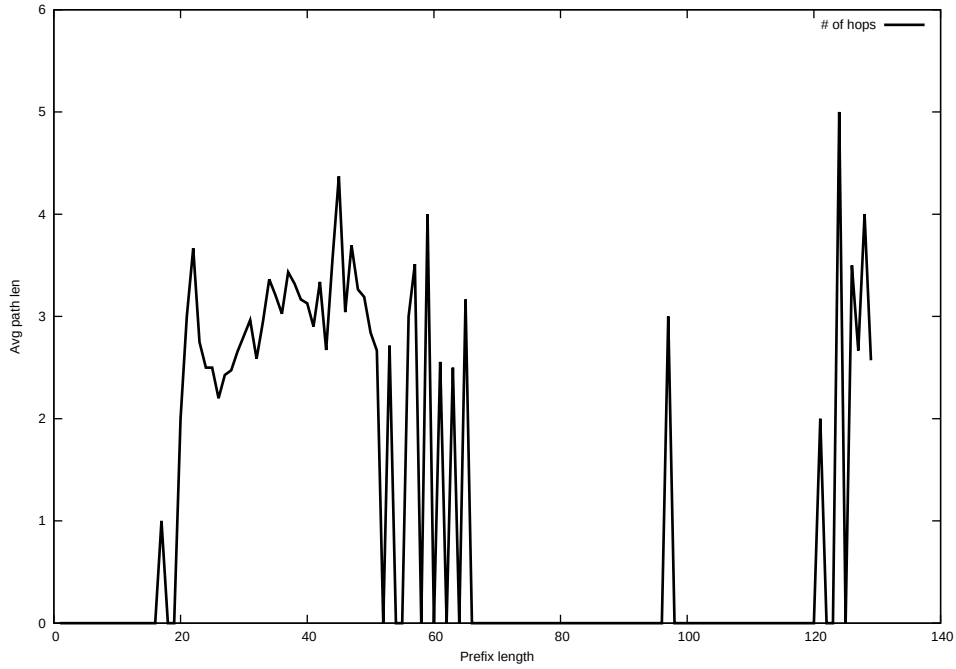
2014-12-26



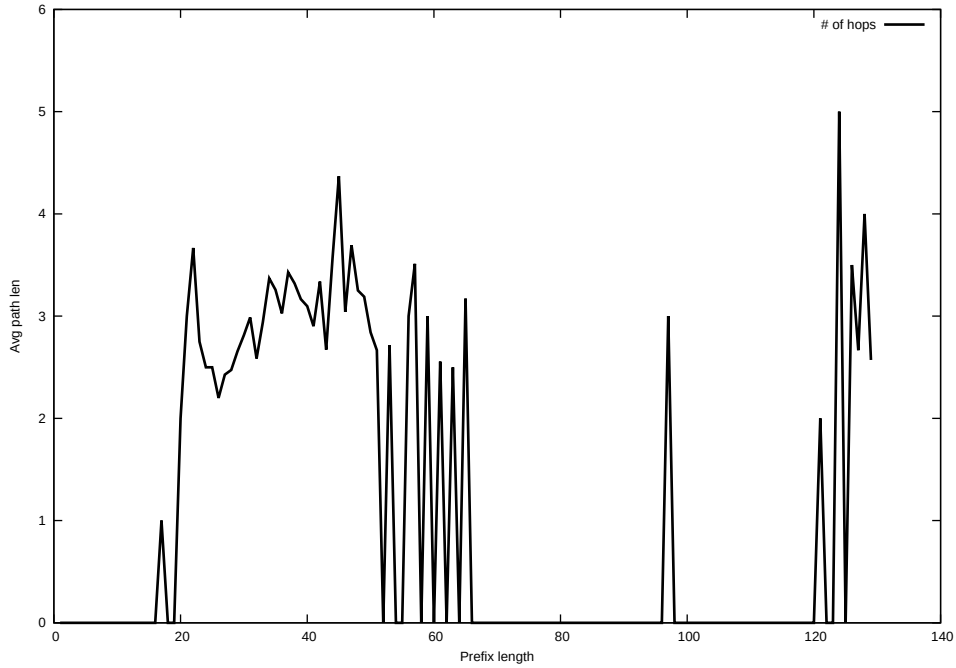
2014-12-27



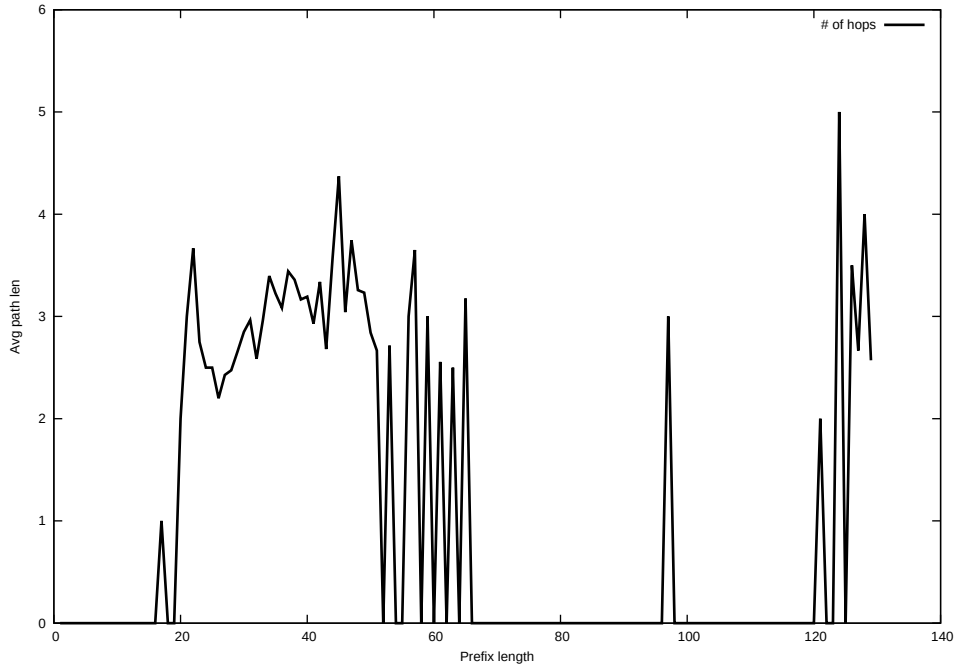
2014-12-28



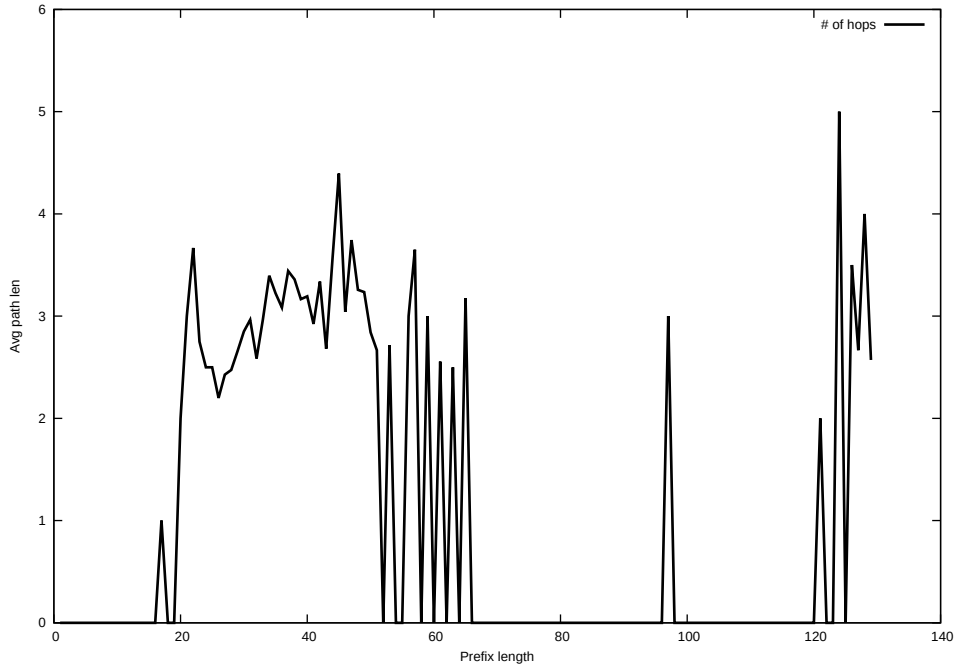
2014-12-29



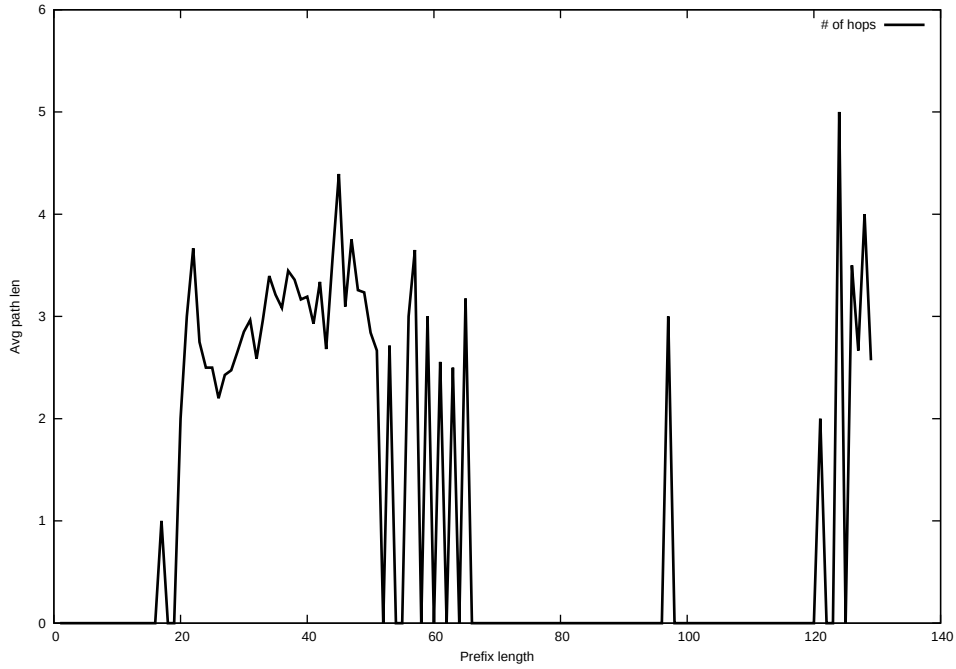
2014-12-30



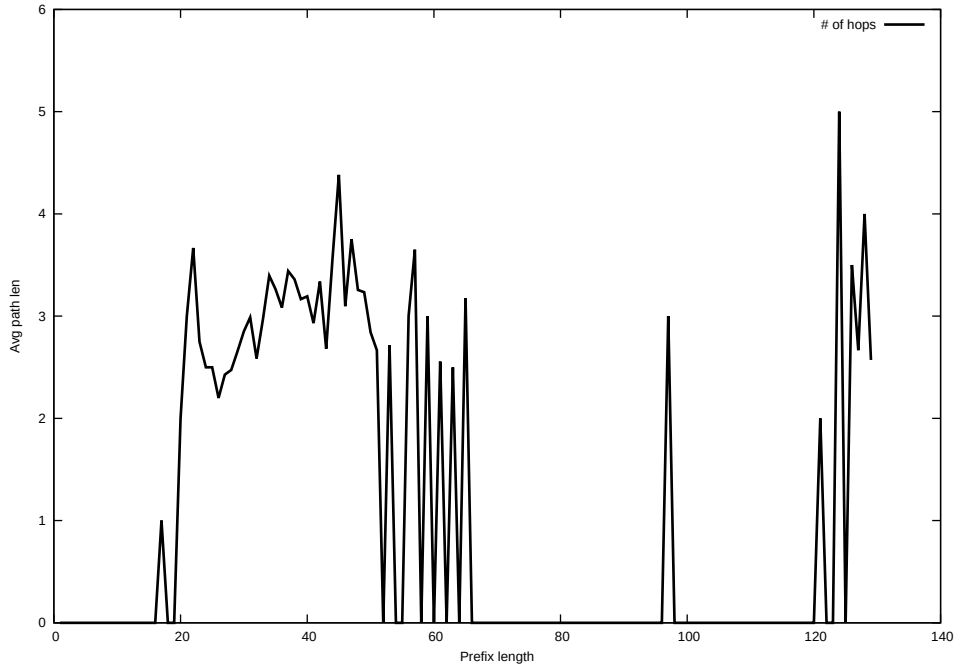
2014-12-31



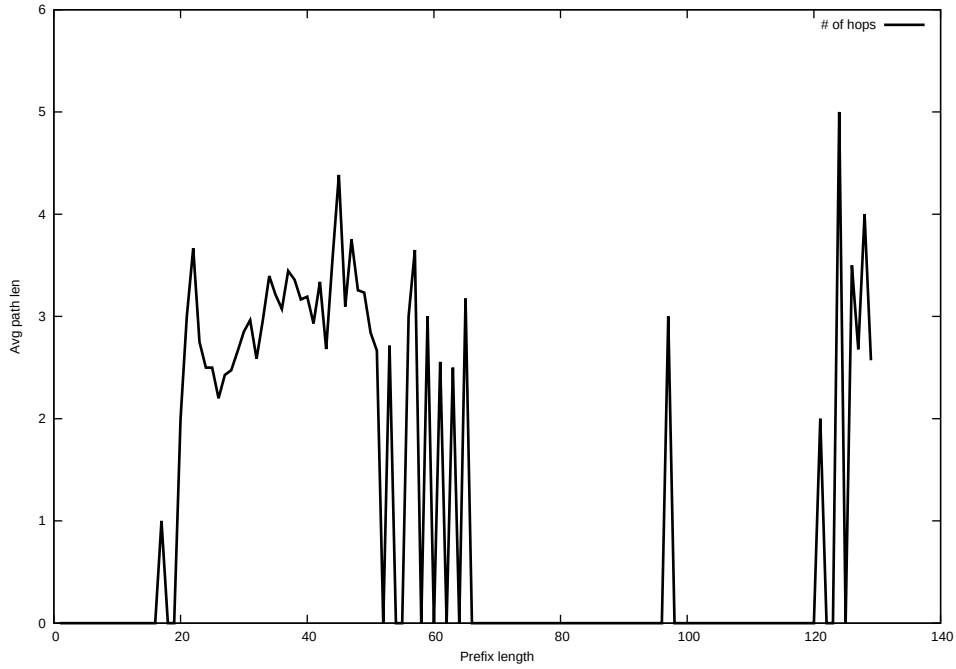
2015-01-01



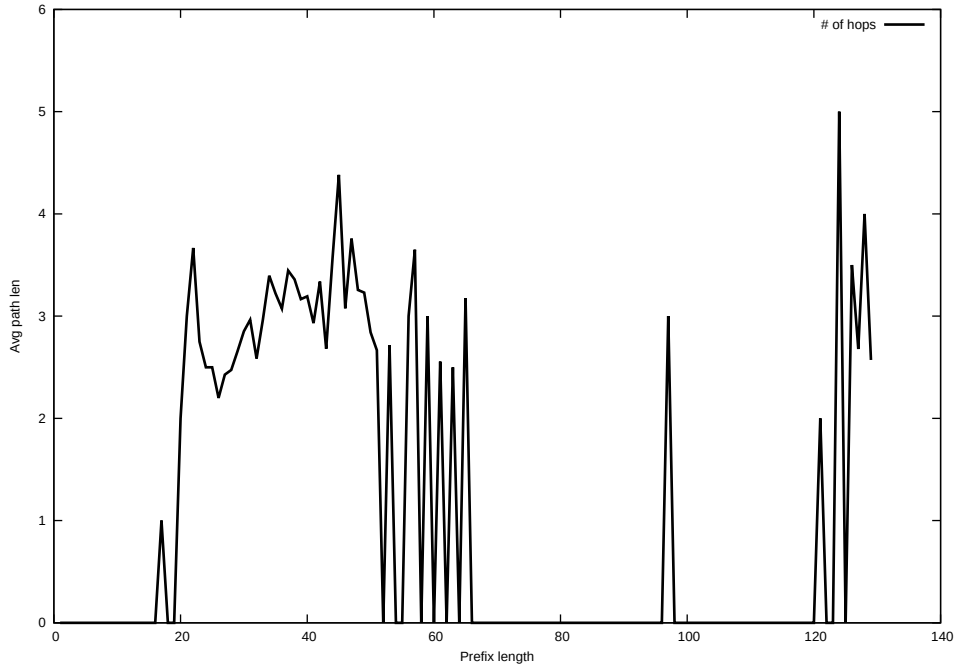
2015-01-02



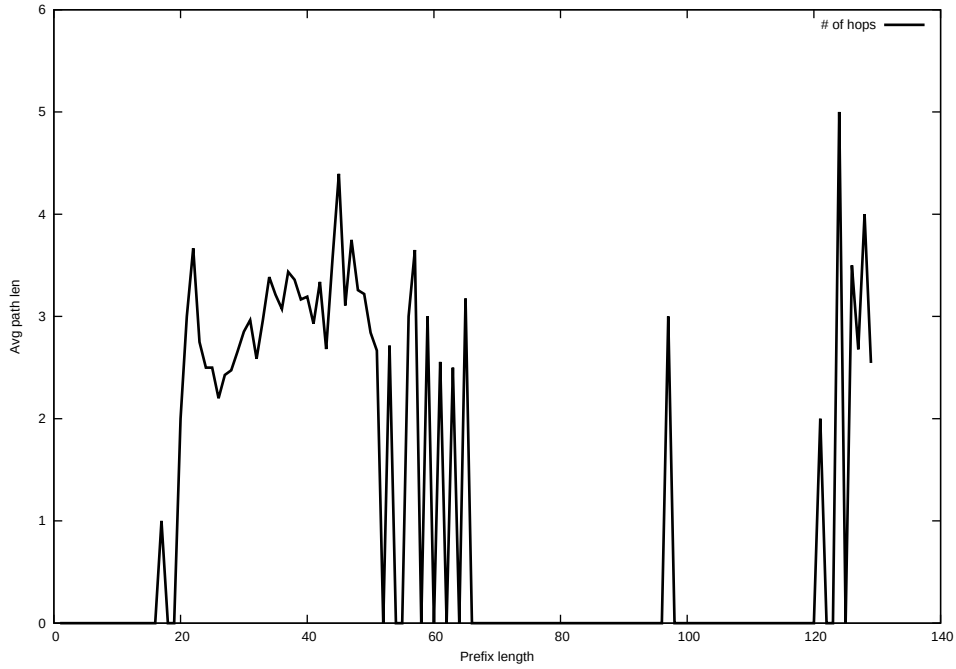
2015-01-03



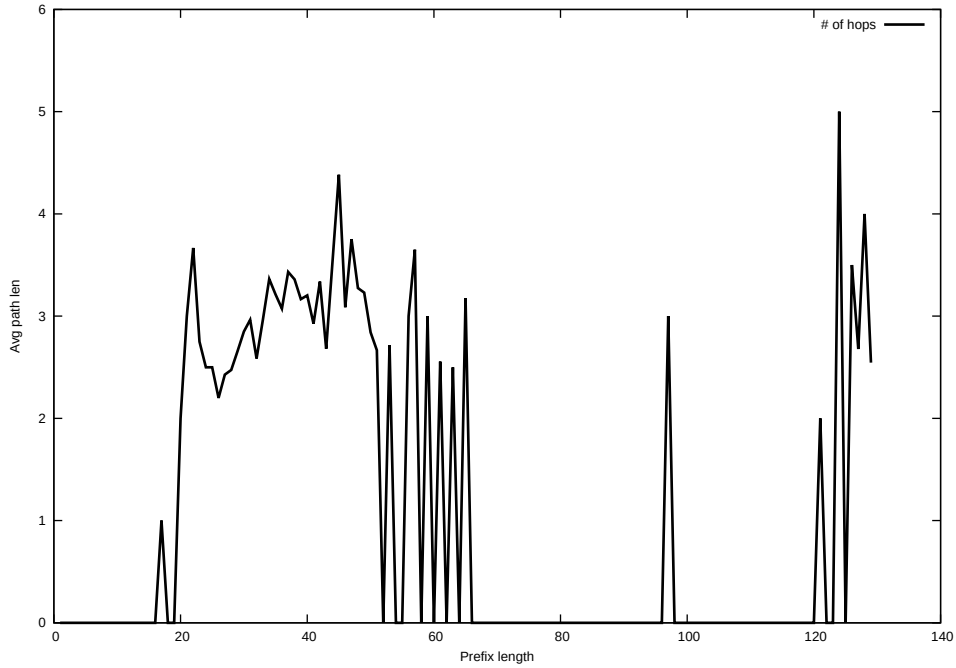
2015-01-04



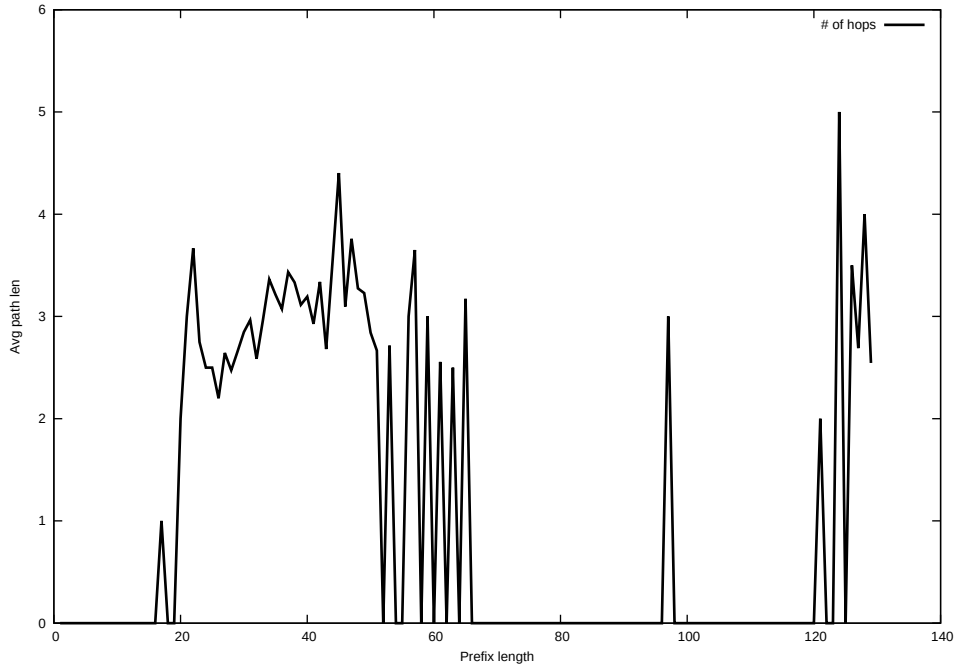
2015-01-05



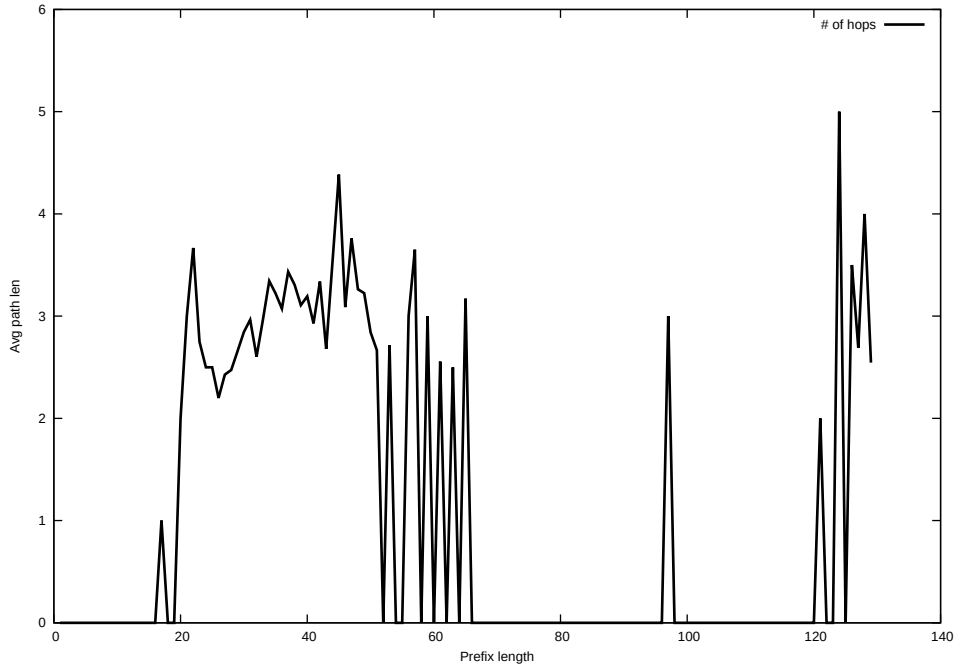
2015-01-06



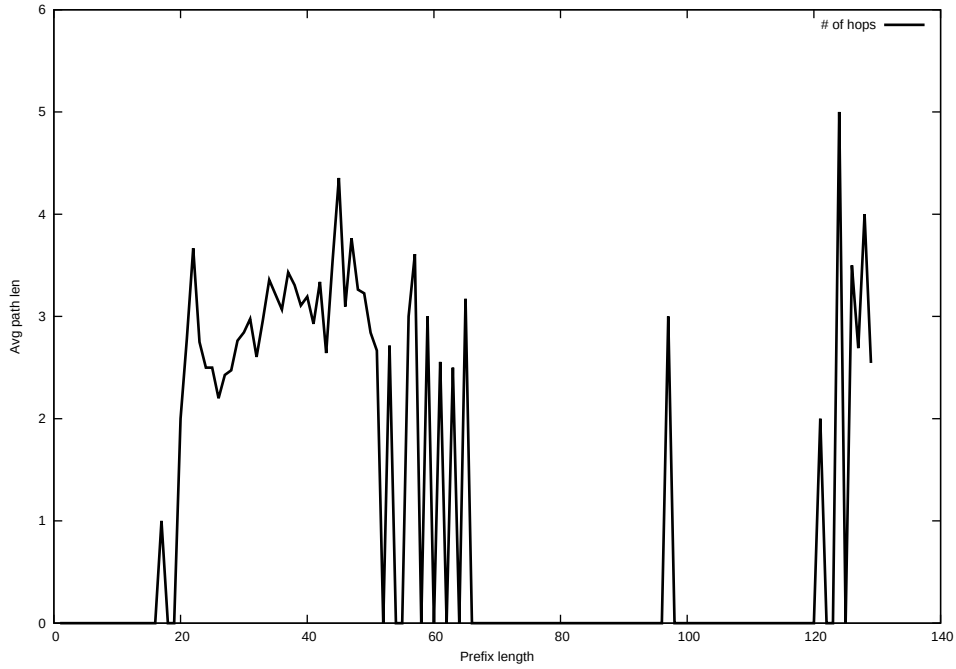
2015-01-07



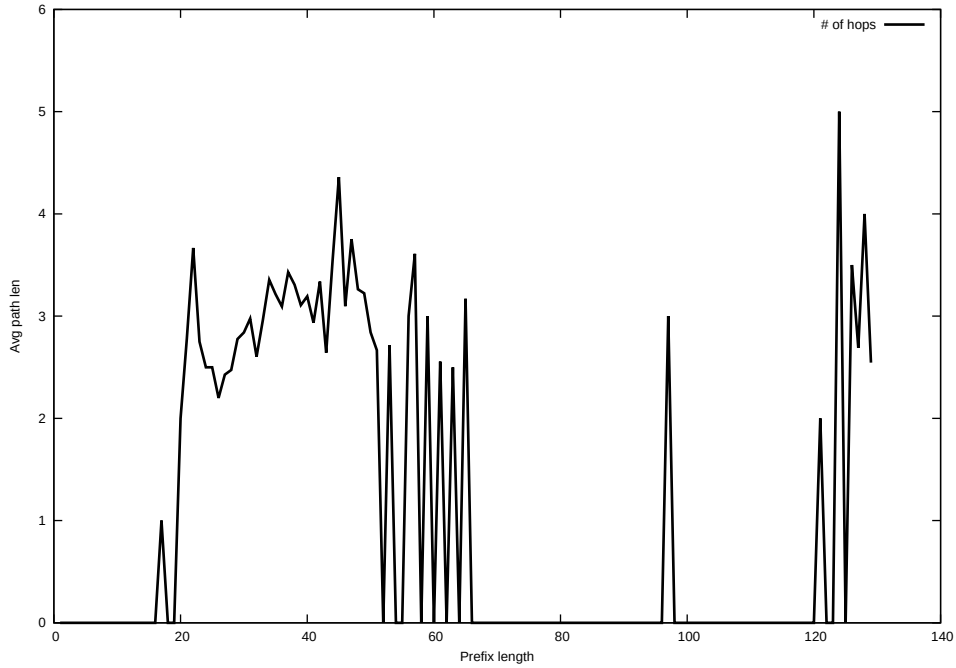
2015-01-08



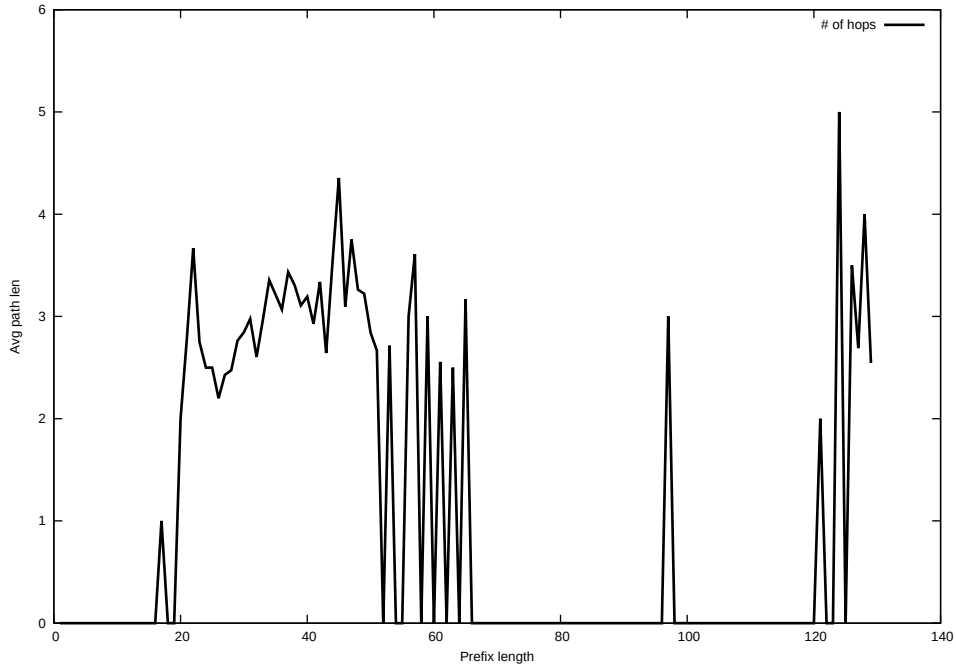
2015-01-09



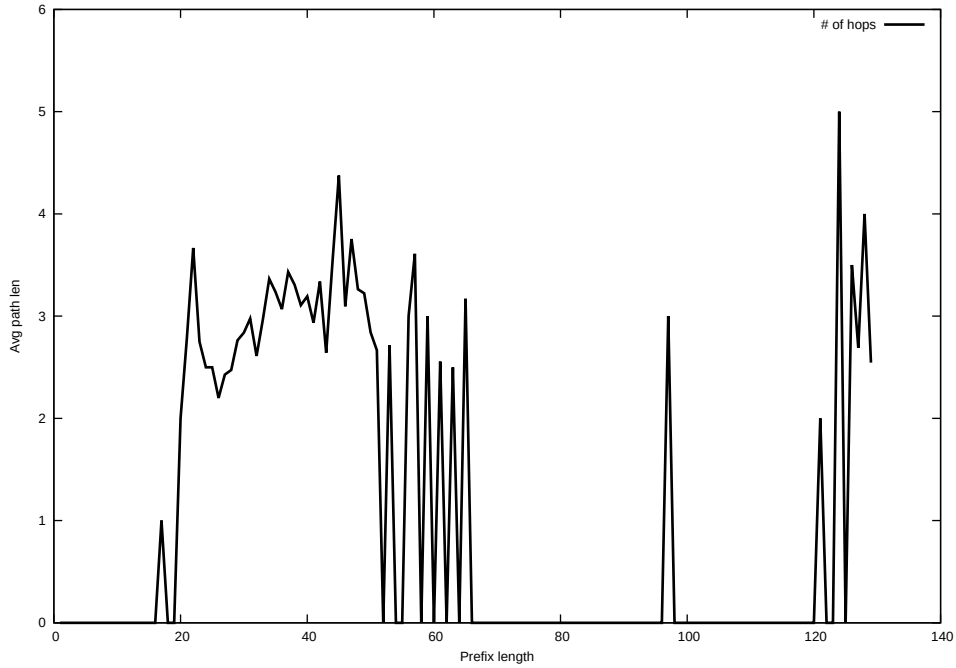
2015-01-10



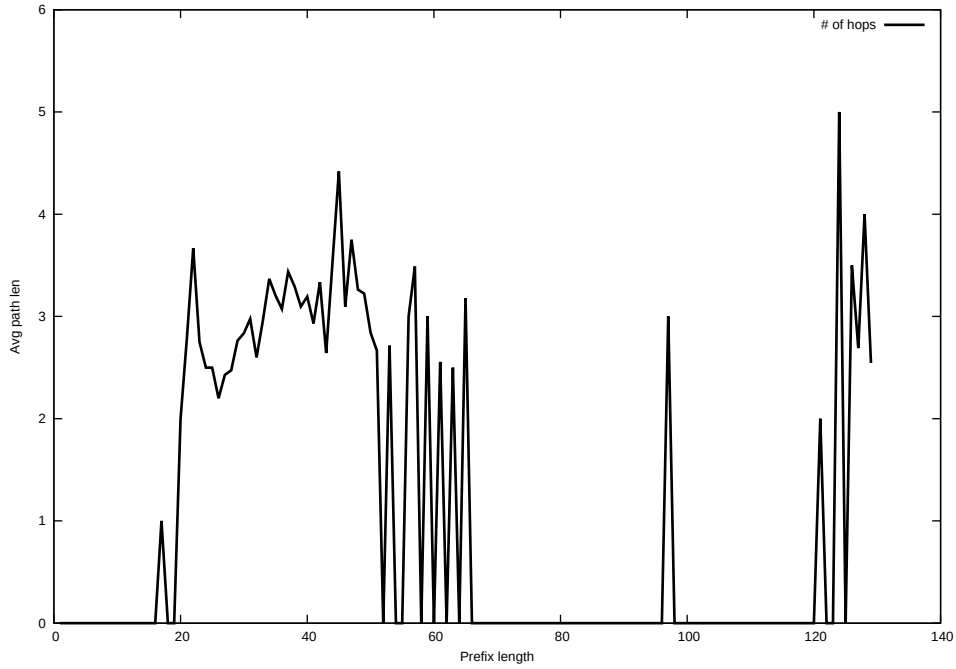
2015-01-11



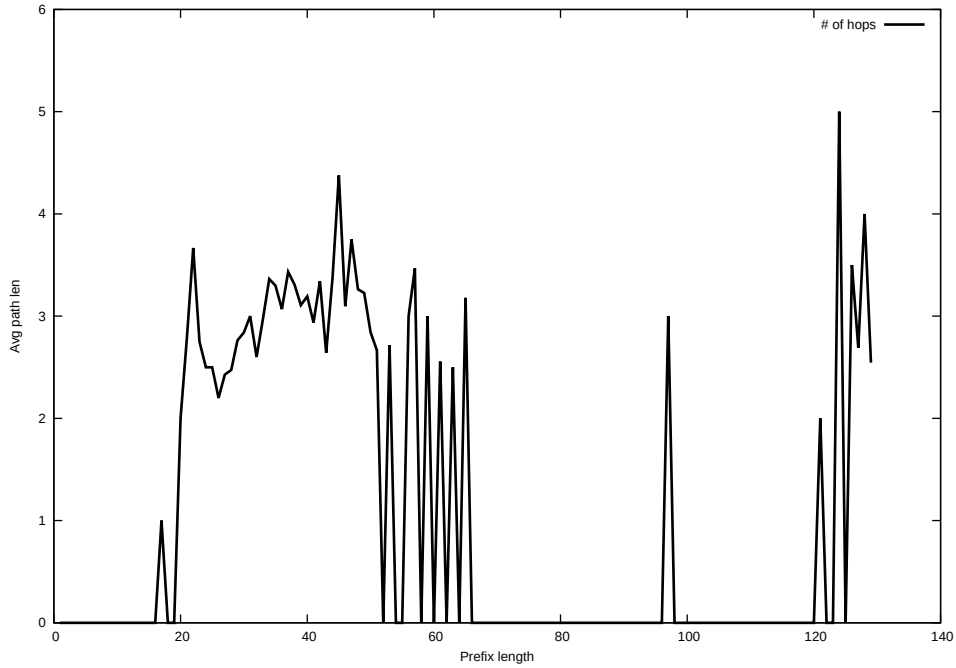
2015-01-12



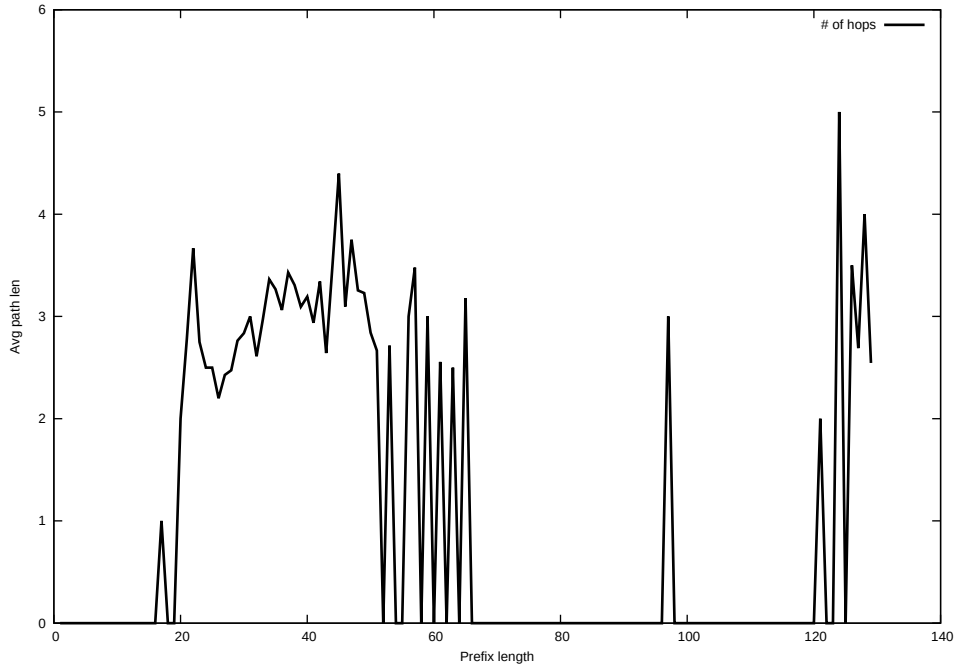
2015-01-13



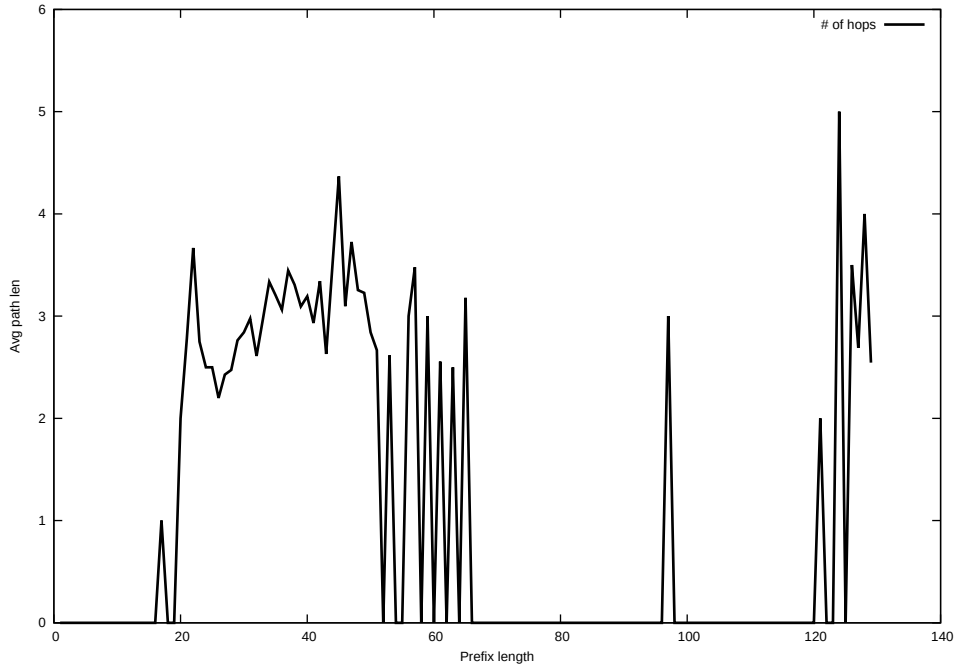
2015-01-14



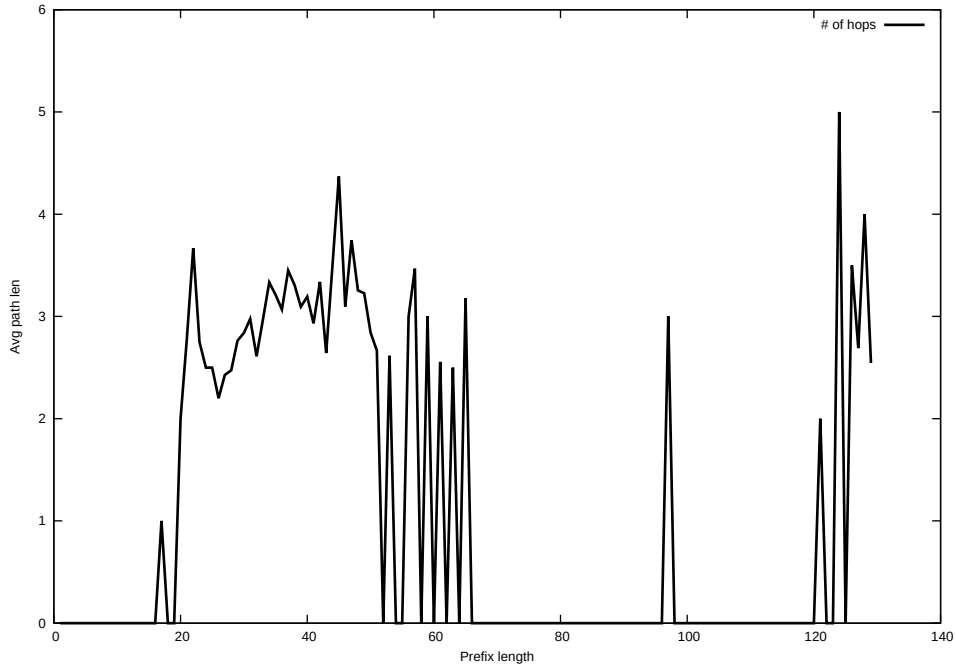
2015-01-15



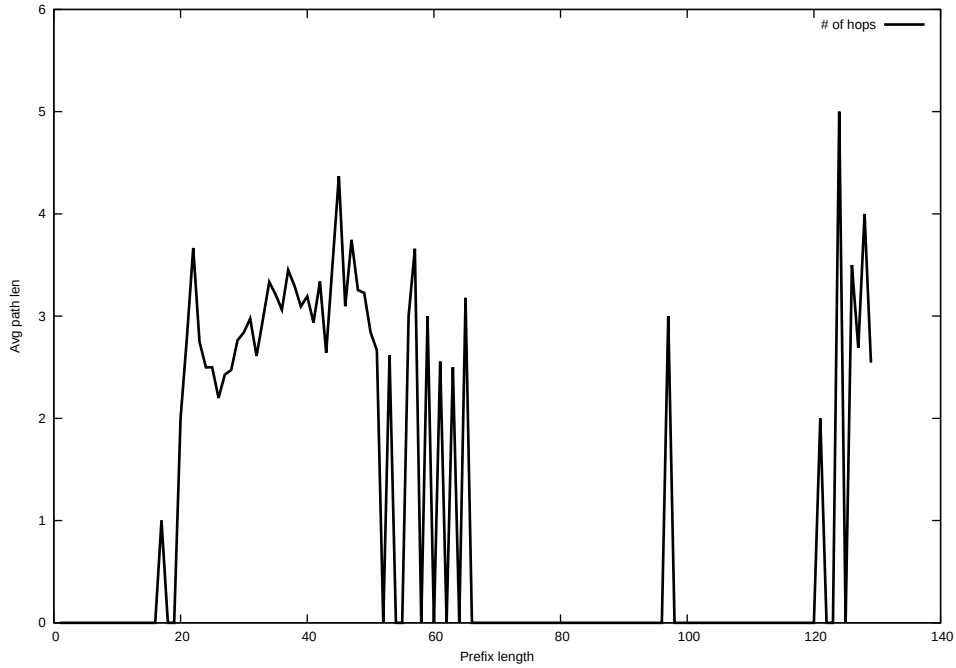
2015-01-16



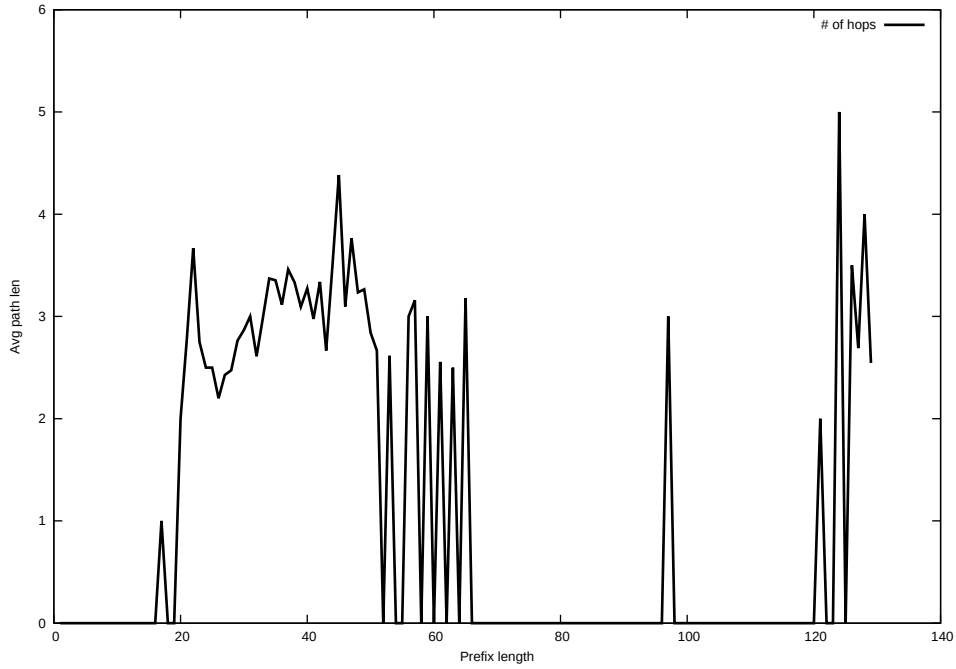
2015-01-17



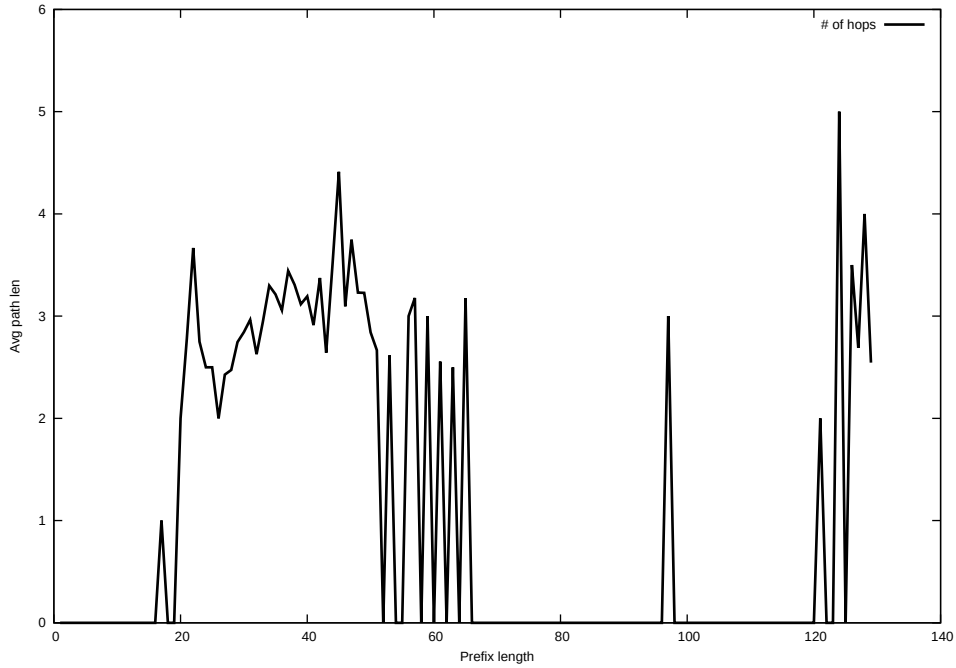
2015-01-18



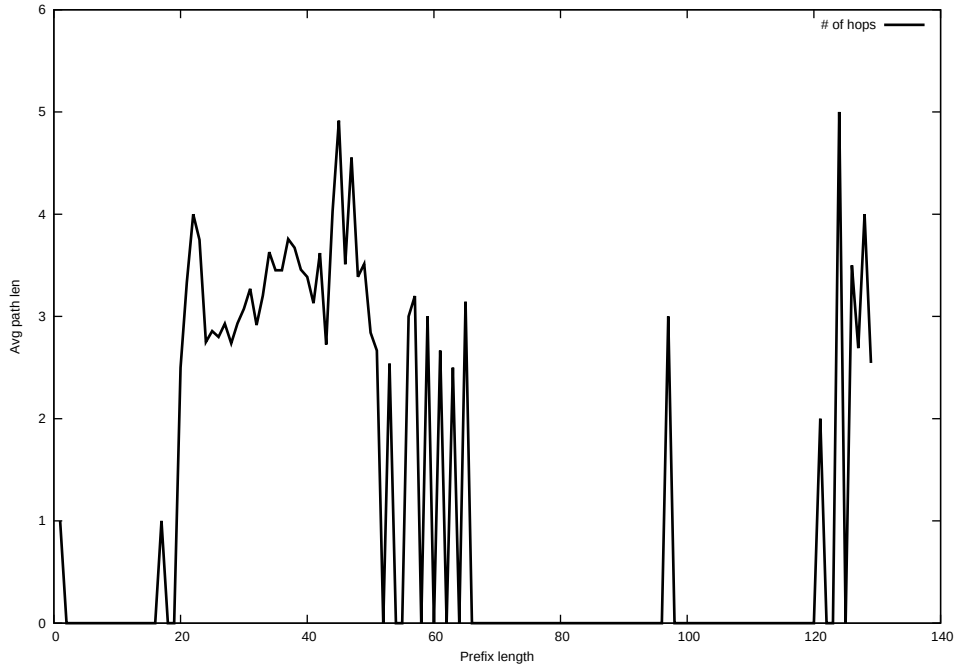
2015-01-19



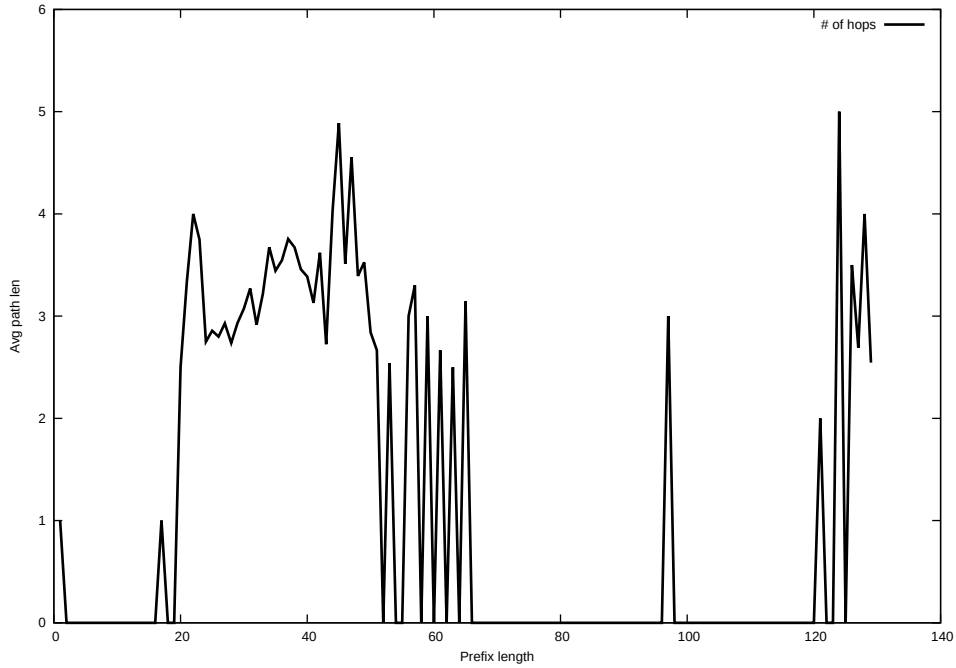
2015-01-20



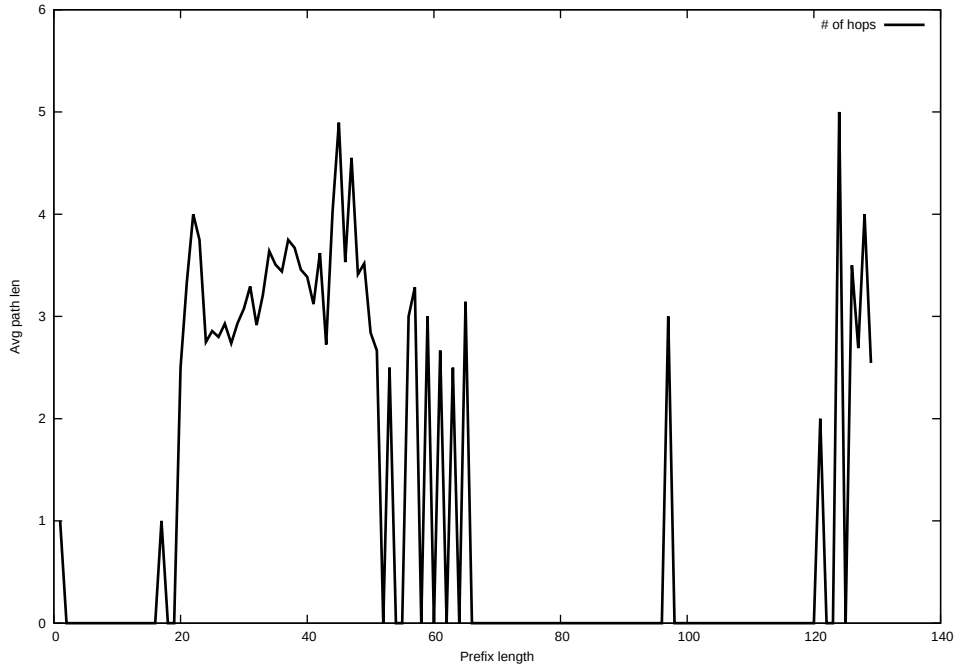
2015-01-21



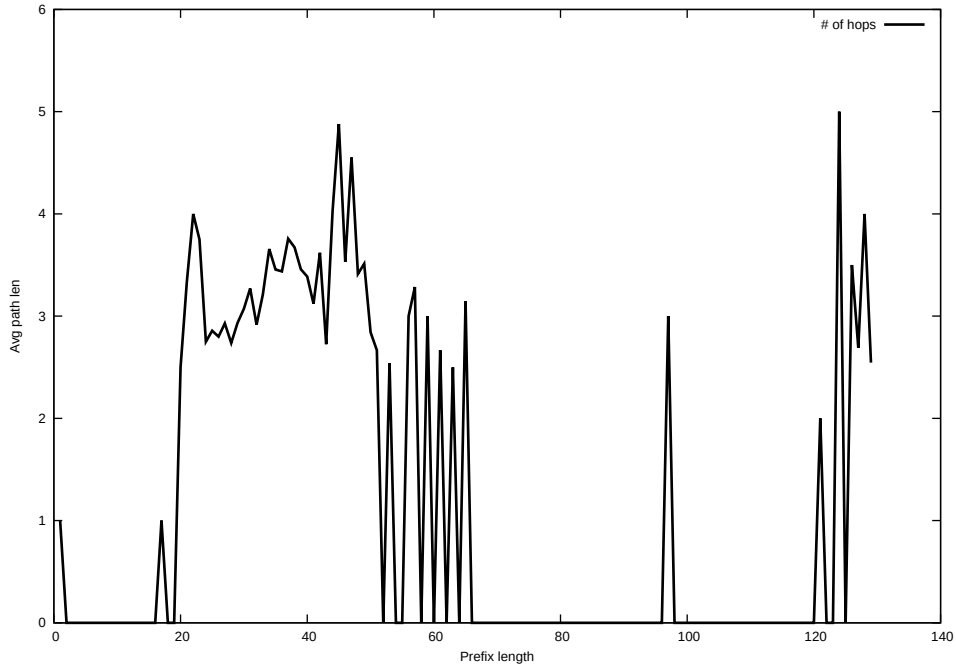
2015-01-22



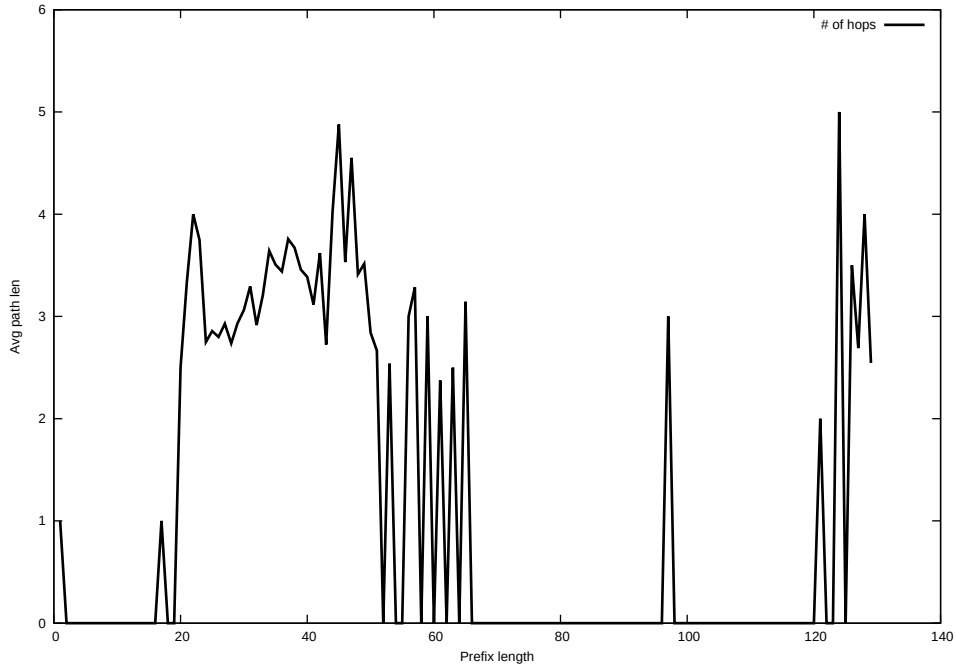
2015-01-23



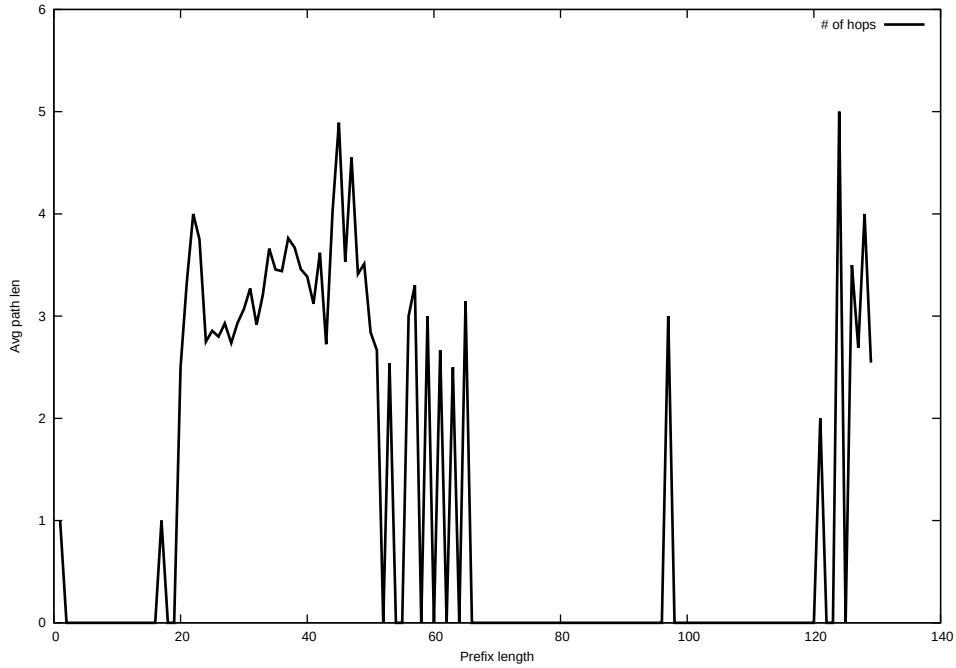
2015-01-24



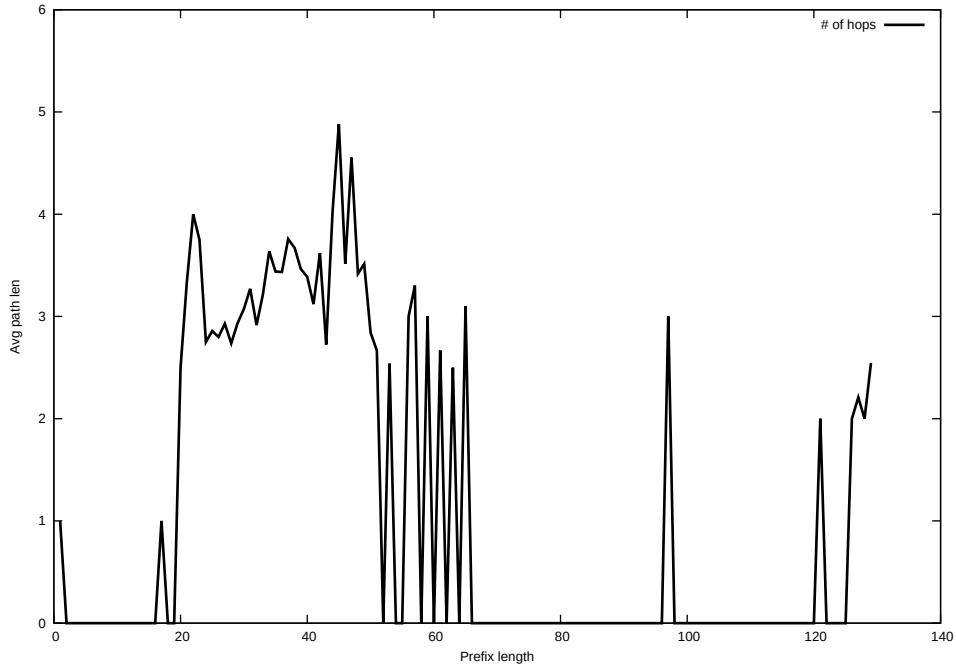
2015-01-25



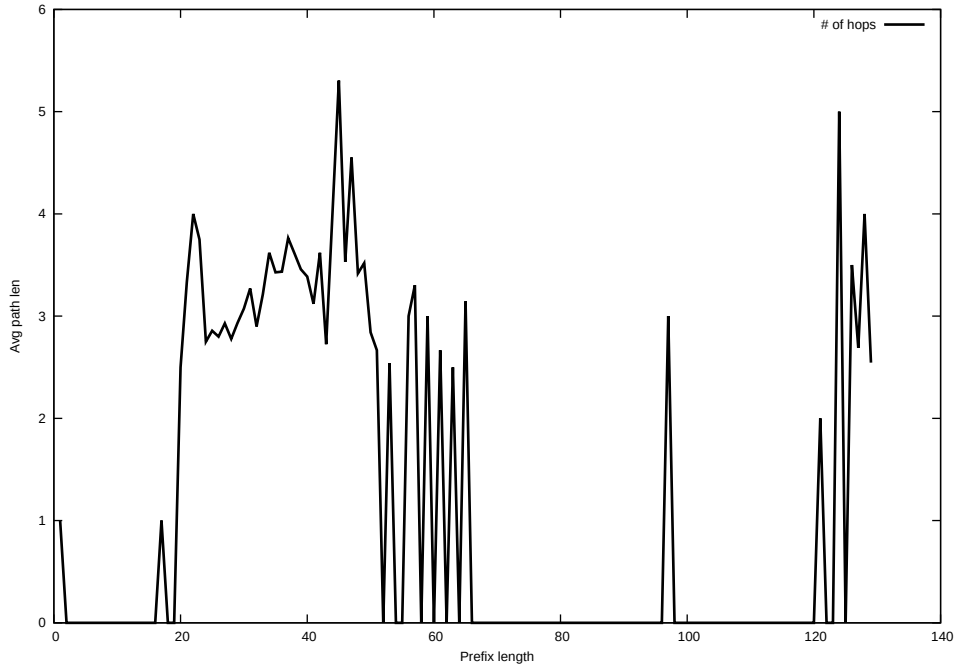
2015-01-26



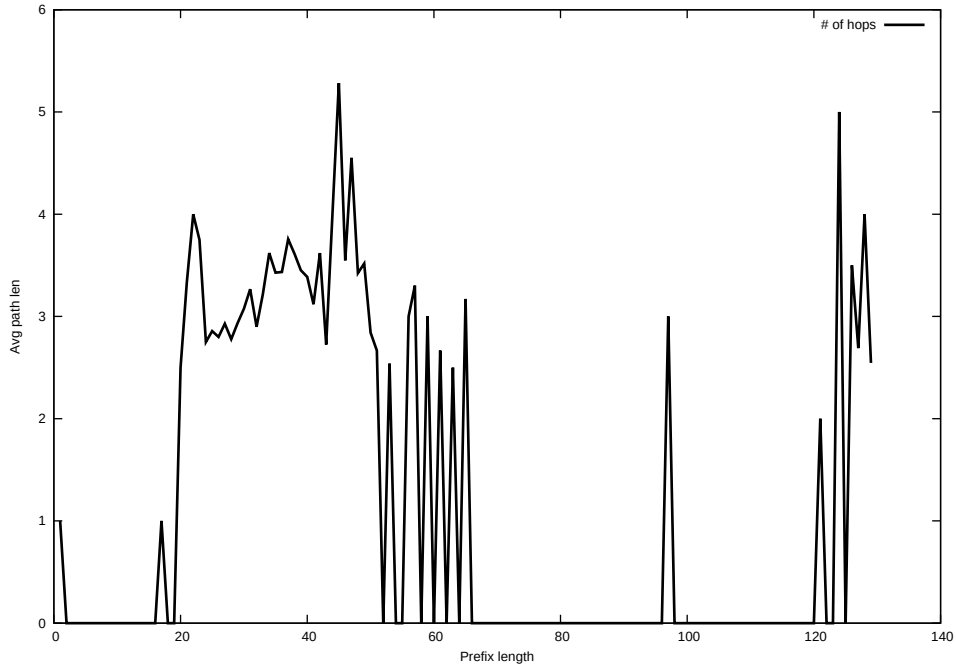
2015-01-27



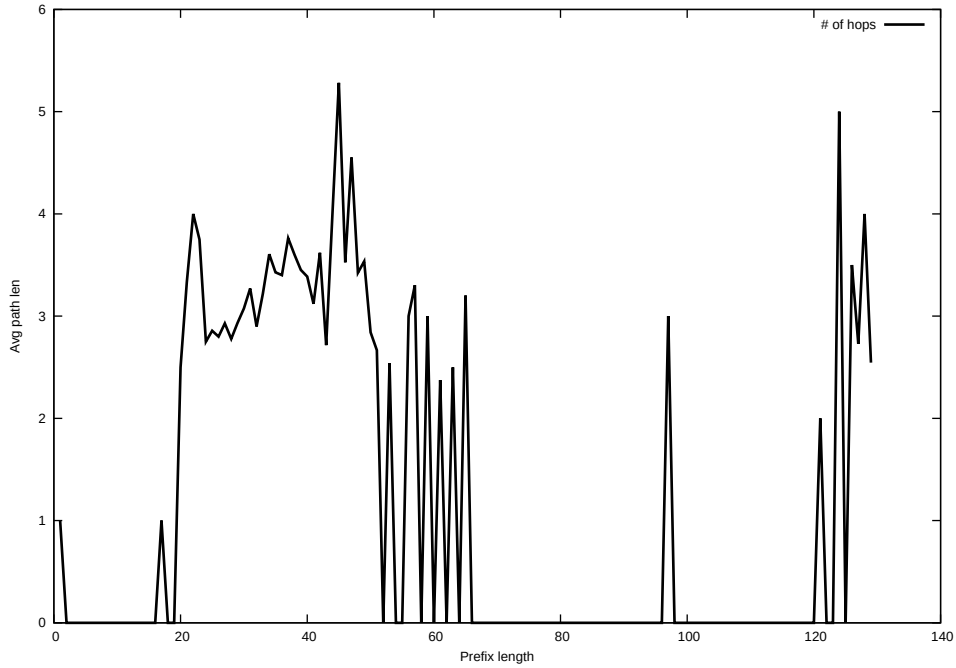
2015-01-28



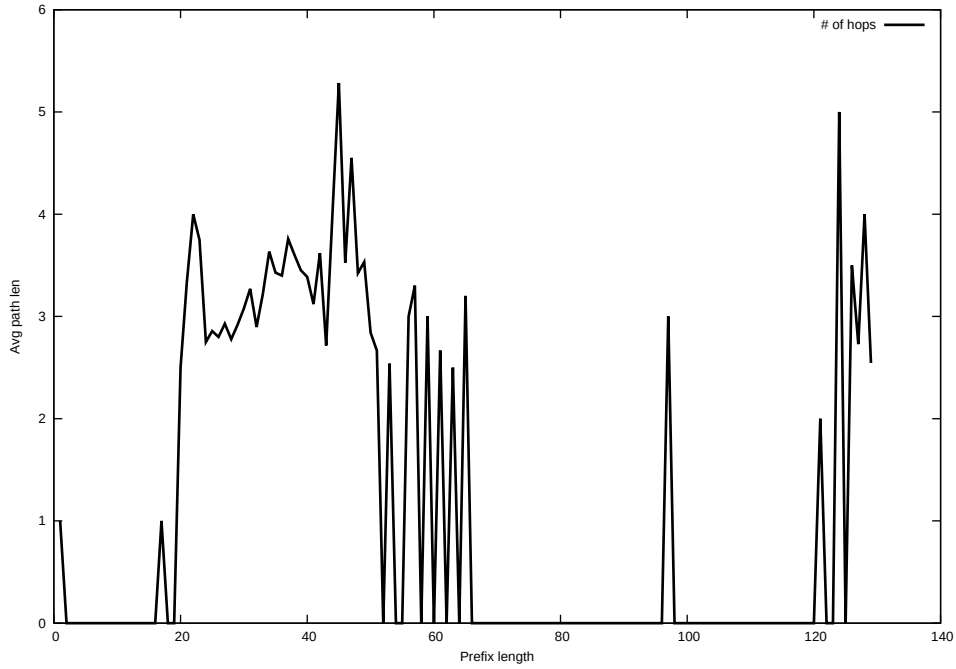
2015-01-29



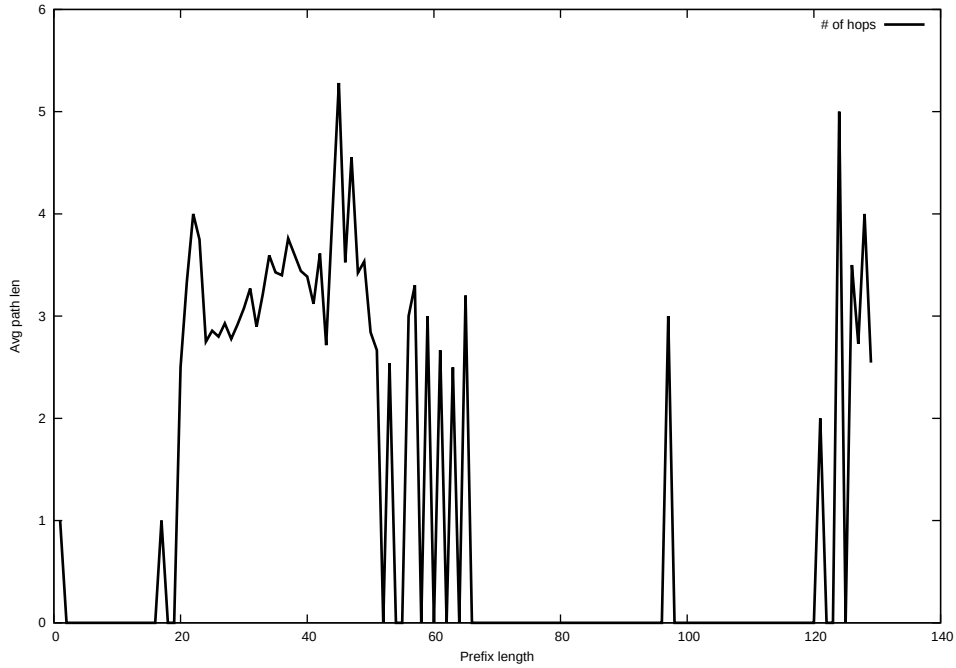
2015-01-30



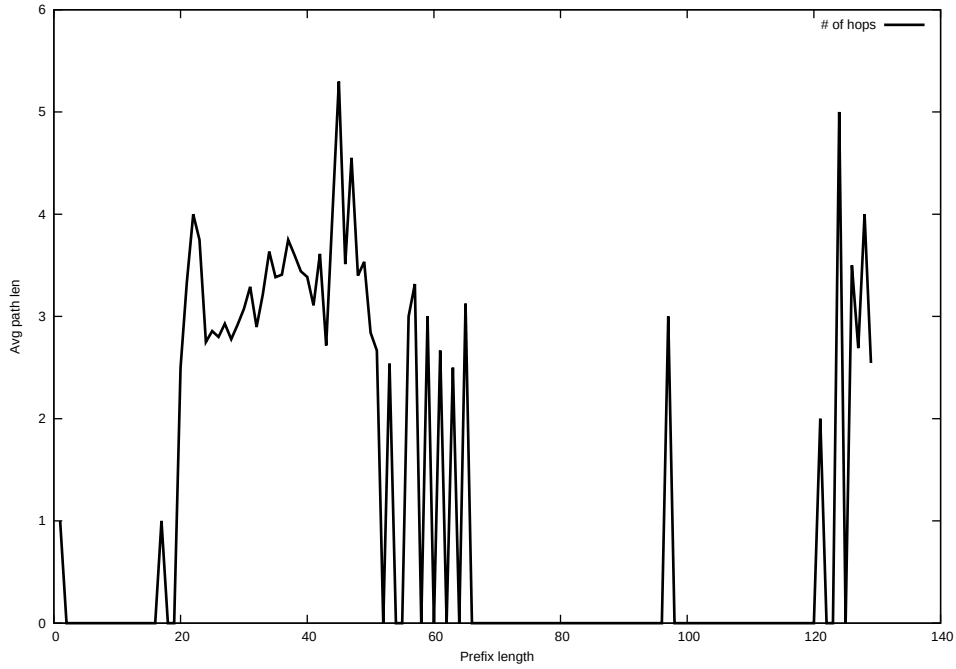
2015-01-31



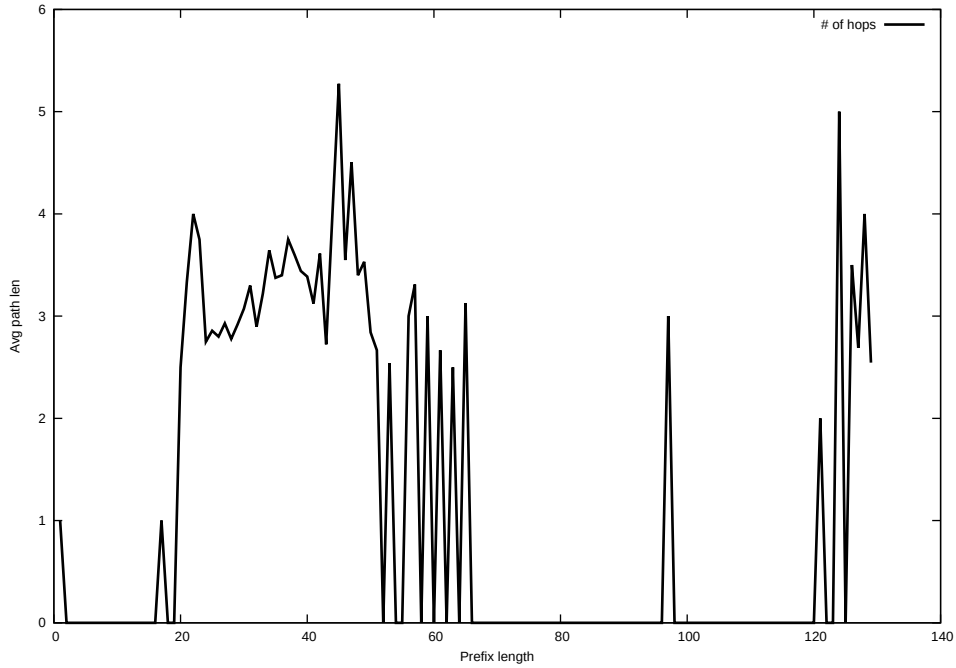
2015-02-01



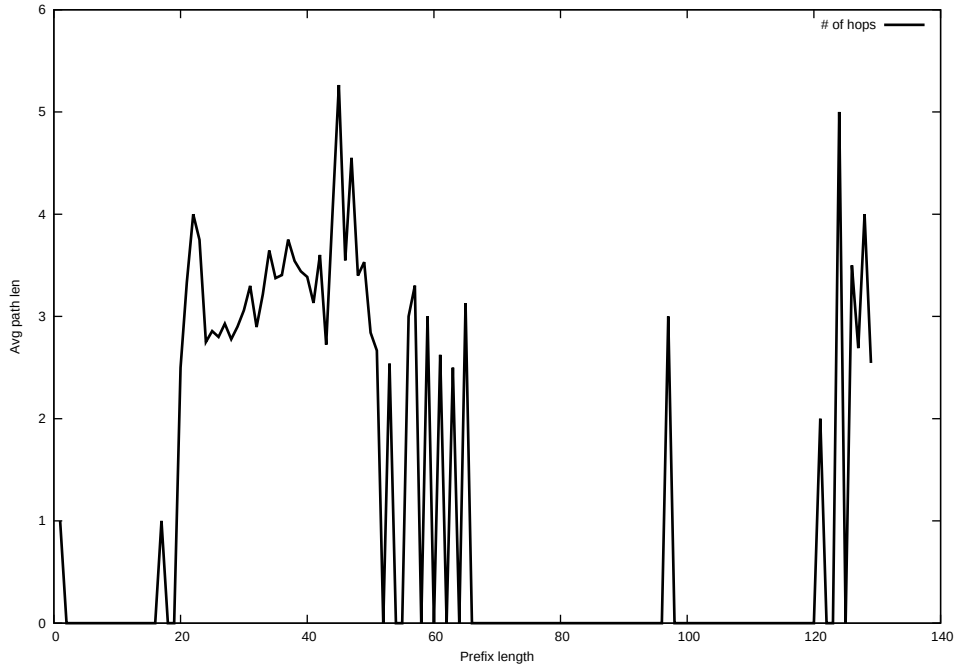
2015-02-02



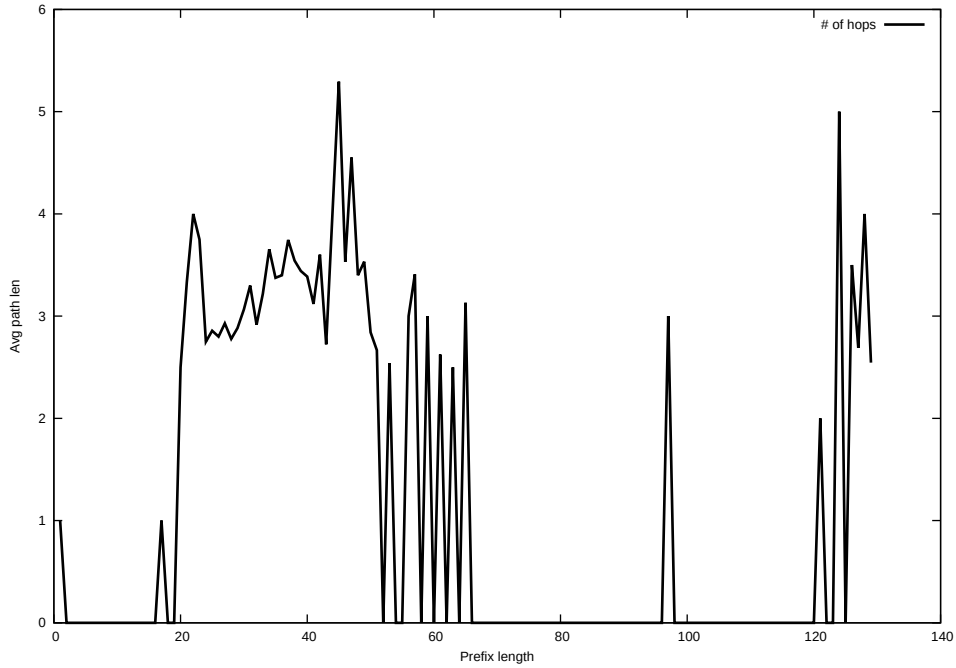
2015-02-03



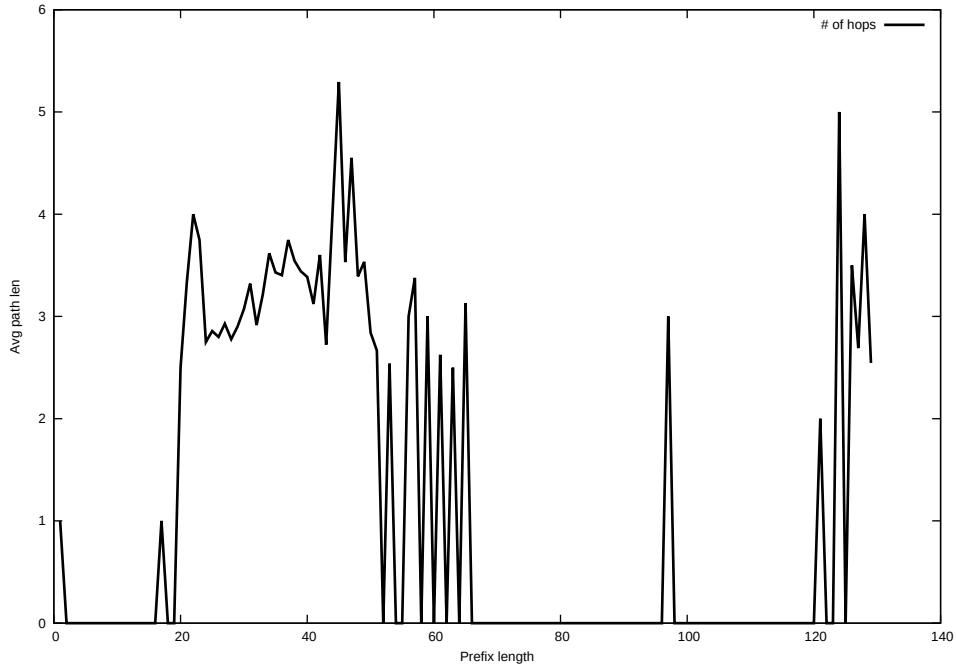
2015-02-04



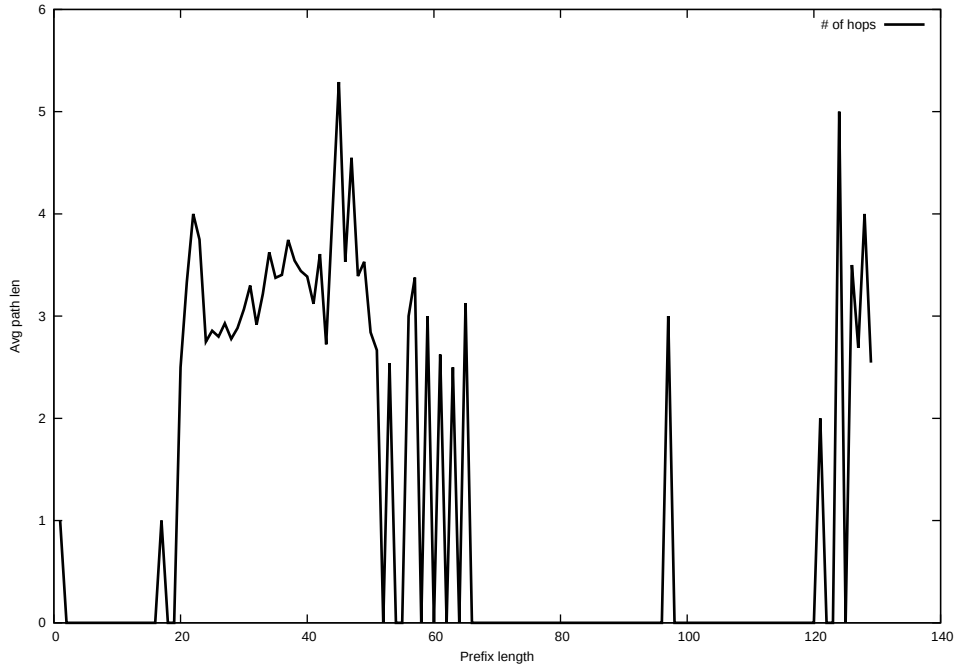
2015-02-05



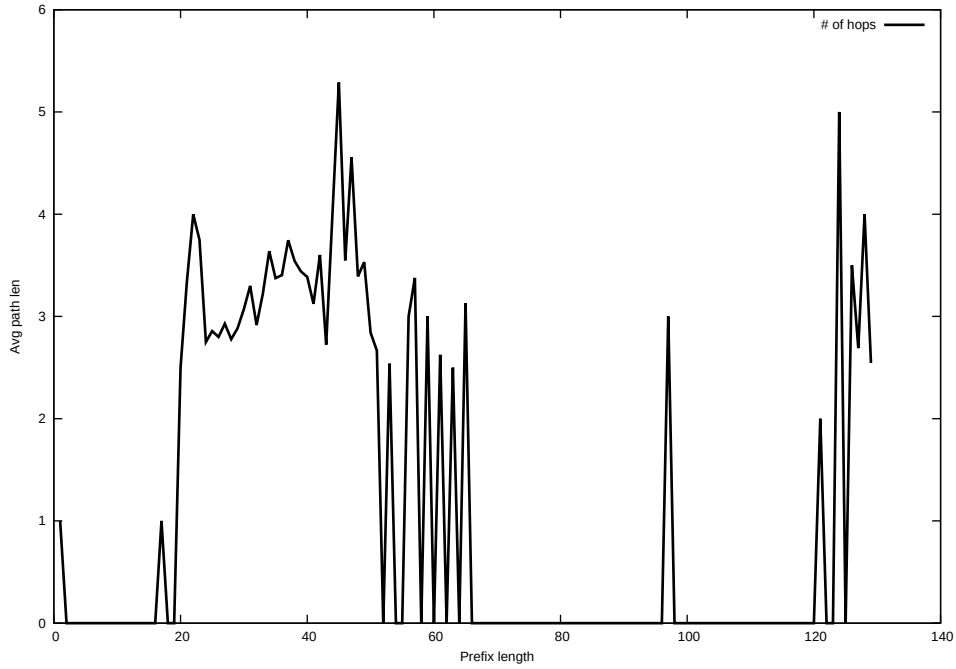
2015-02-06



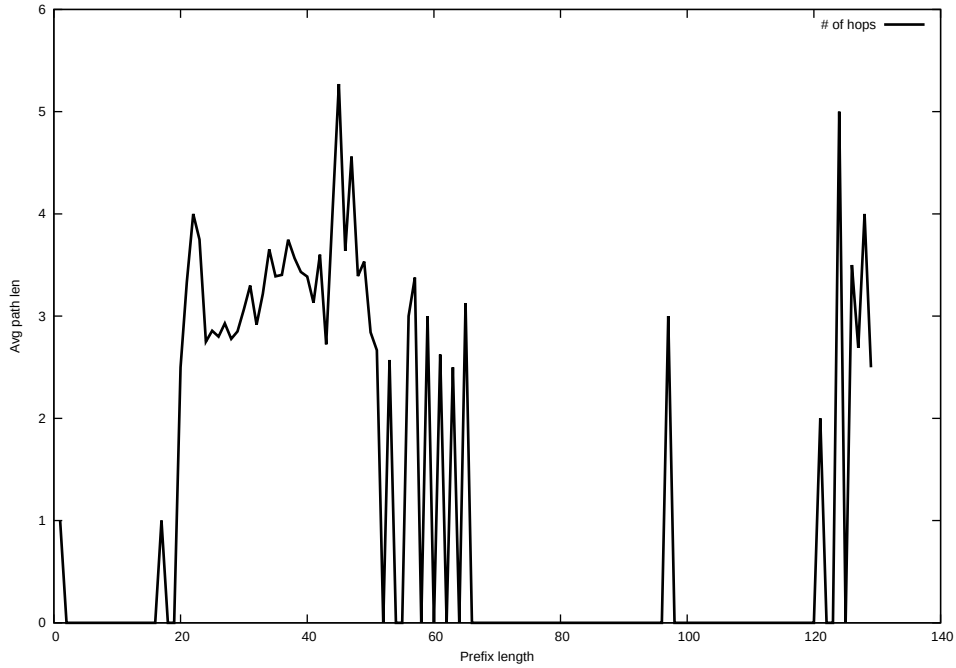
2015-02-07



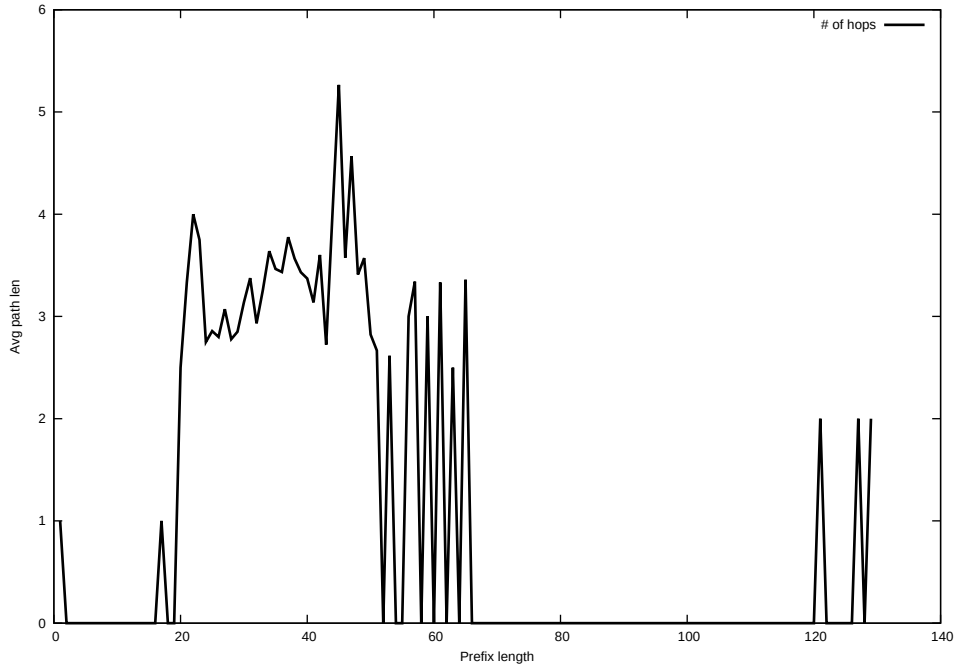
2015-02-08



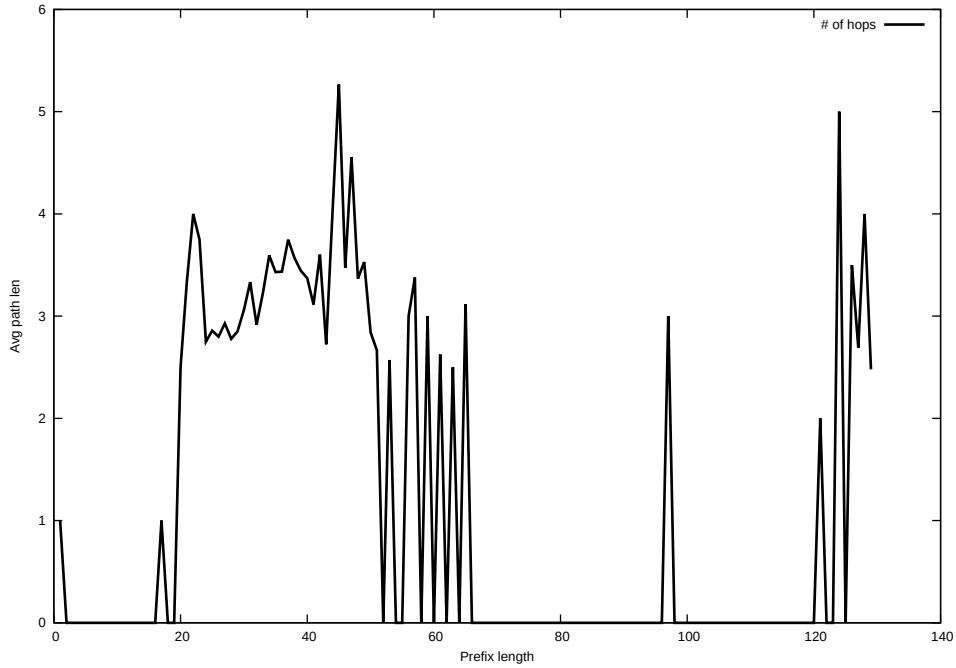
2015-02-09



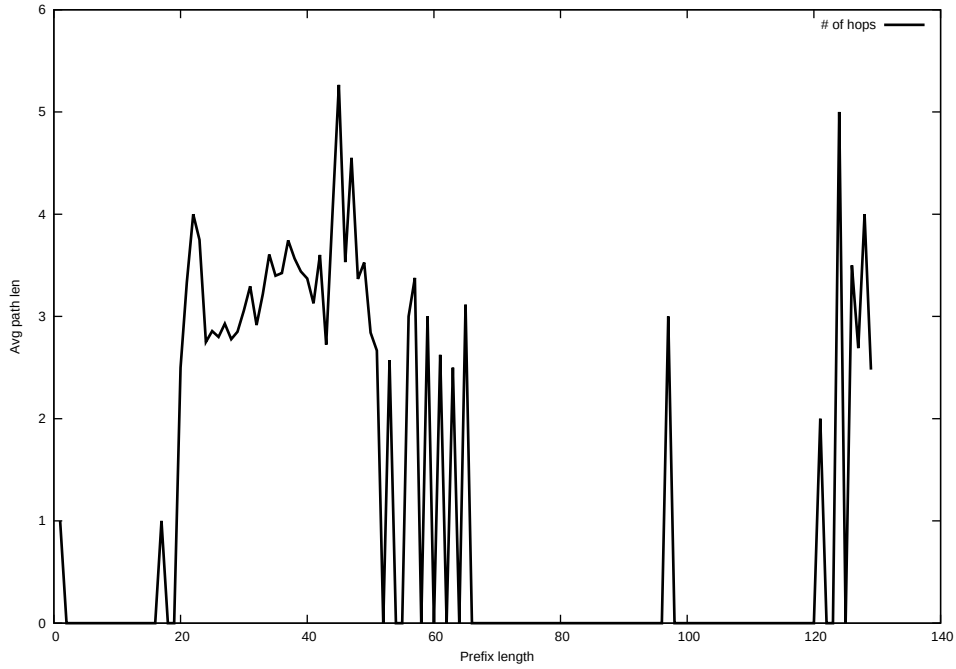
2015-02-10



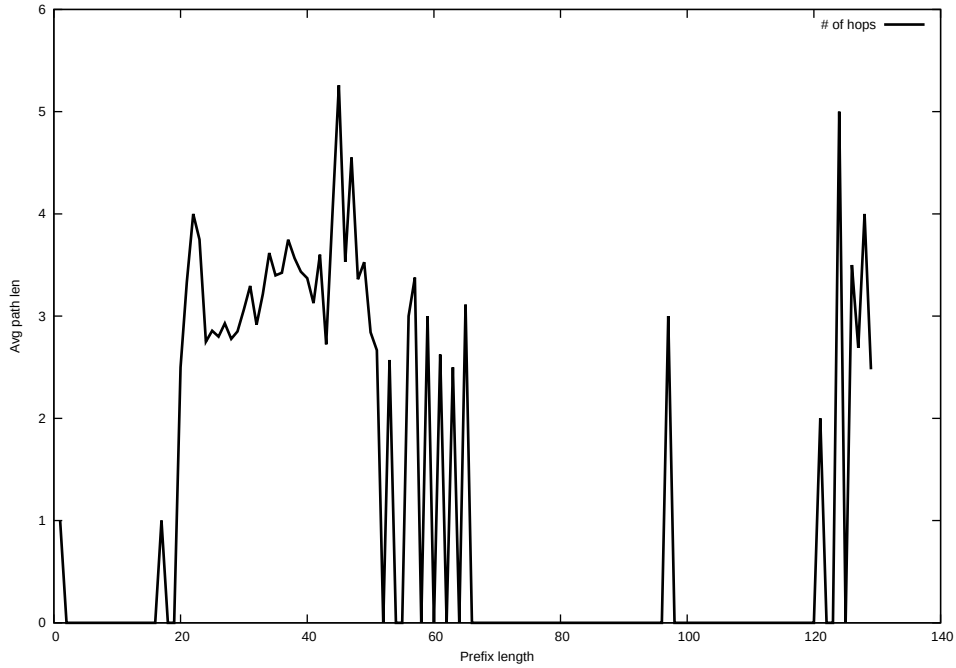
2015-02-11



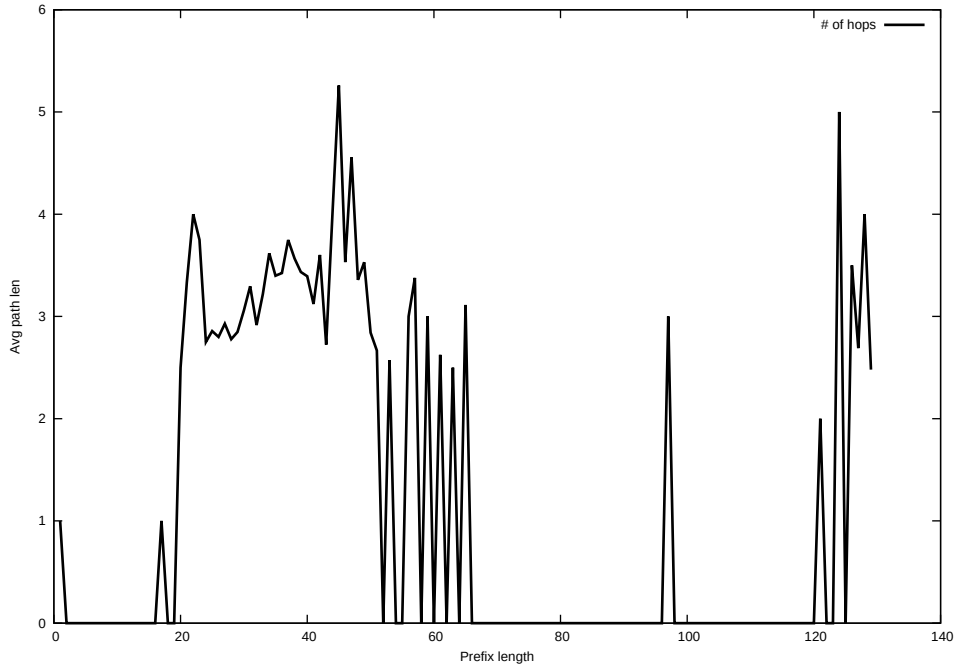
2015-02-12



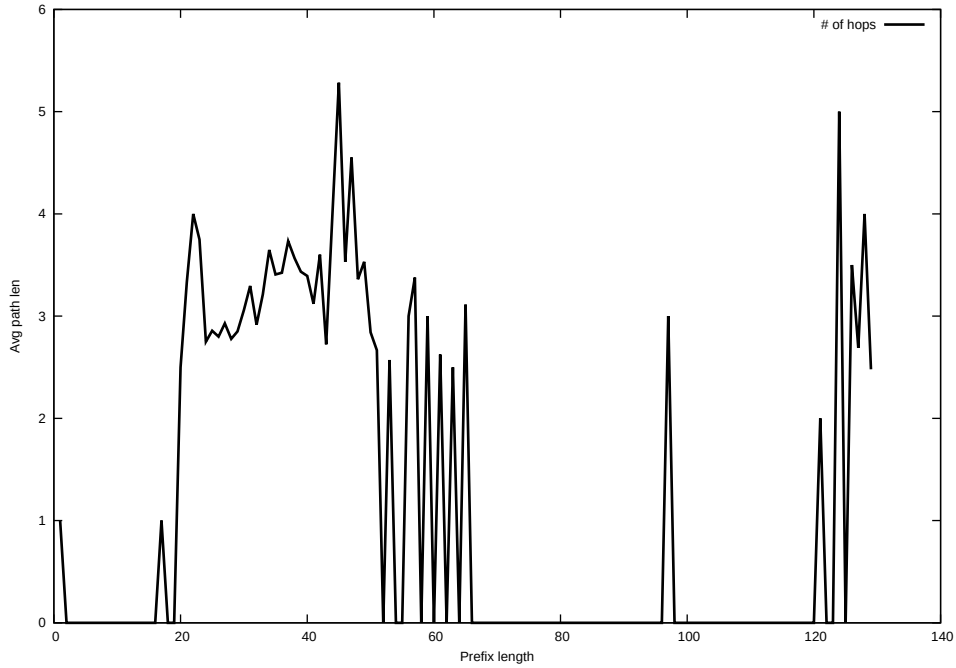
2015-02-13



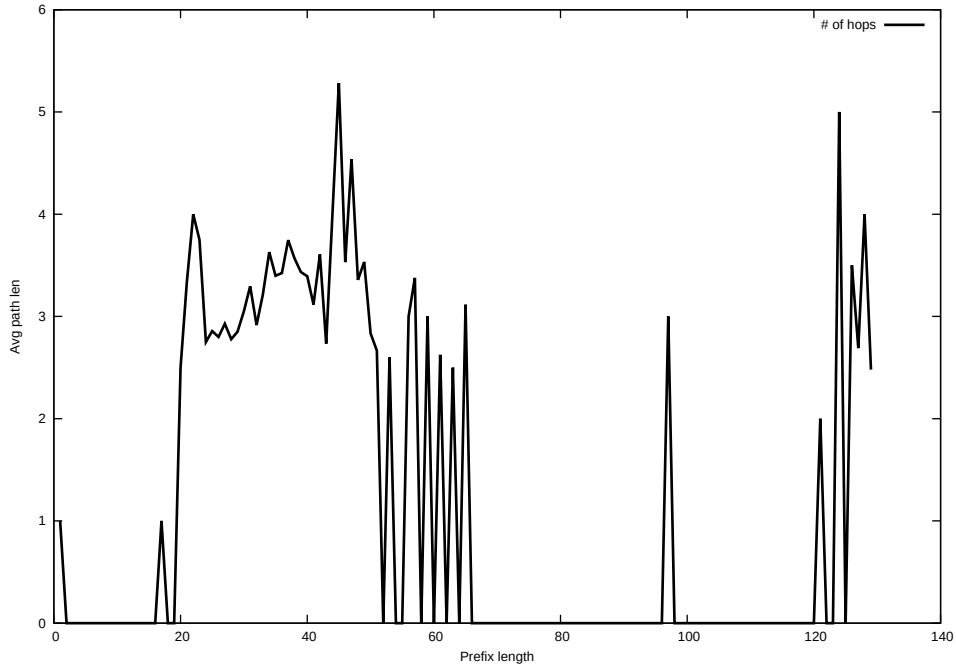
2015-02-14



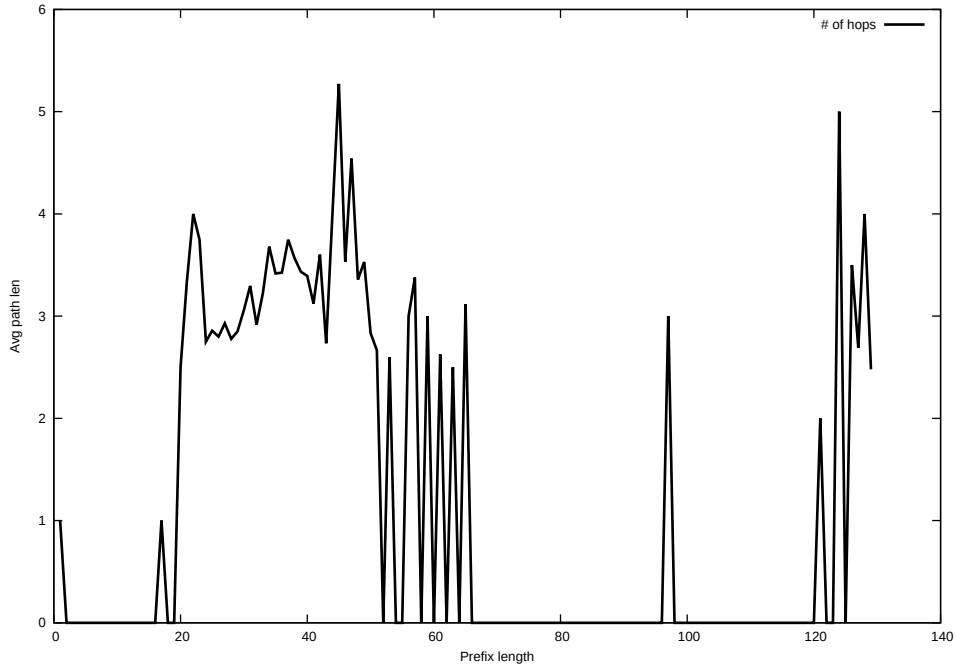
2015-02-15



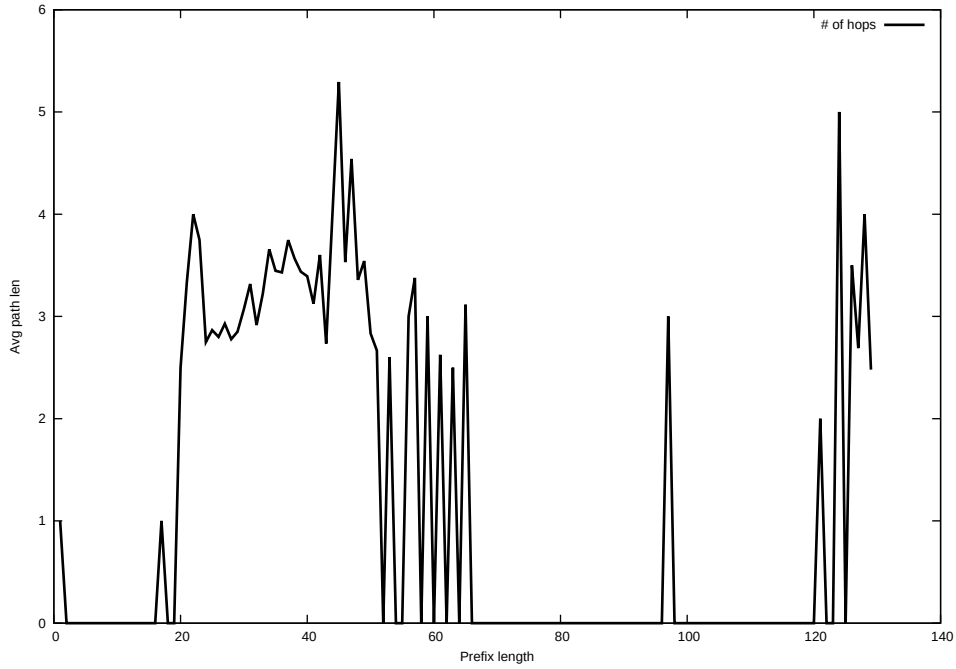
2015-02-16



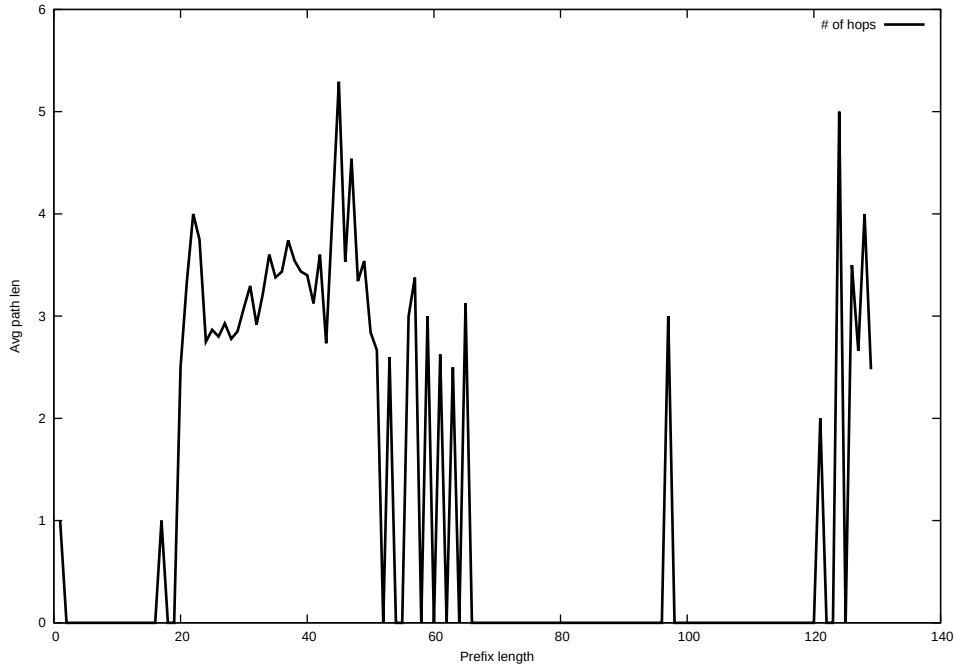
2015-02-17



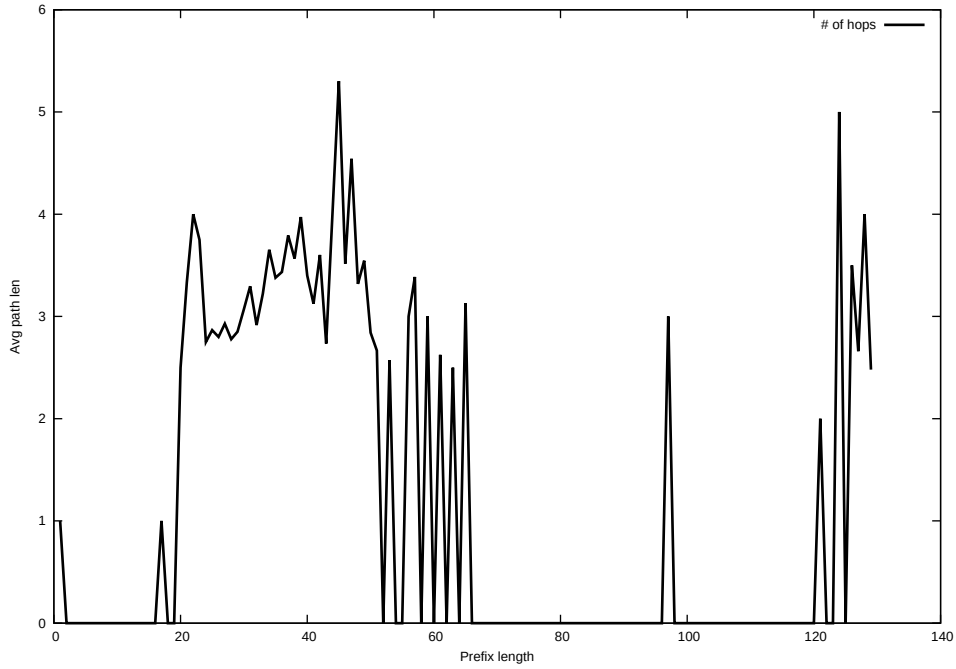
2015-02-18



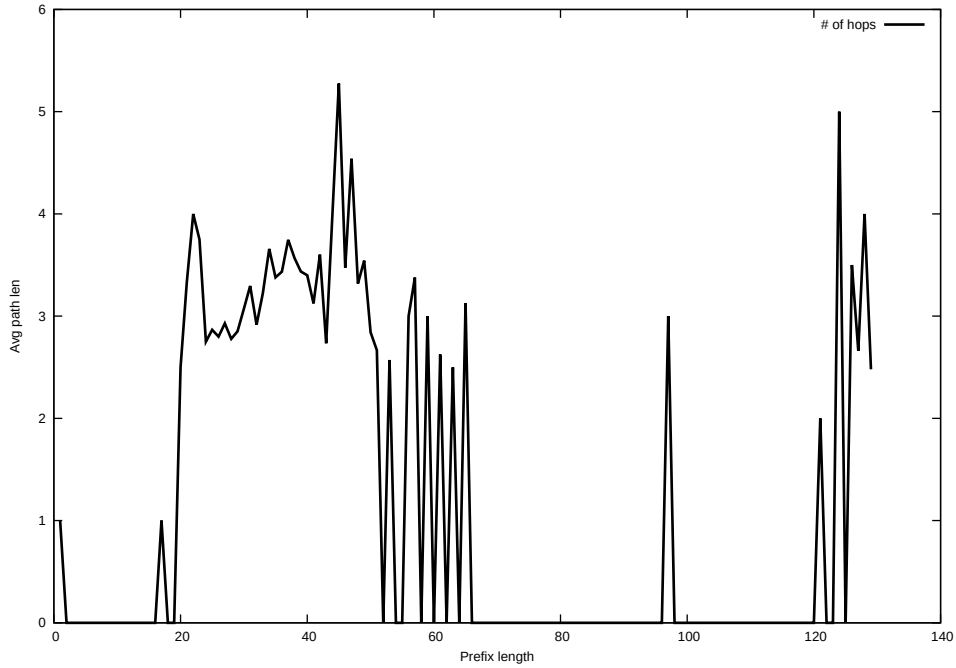
2015-02-19



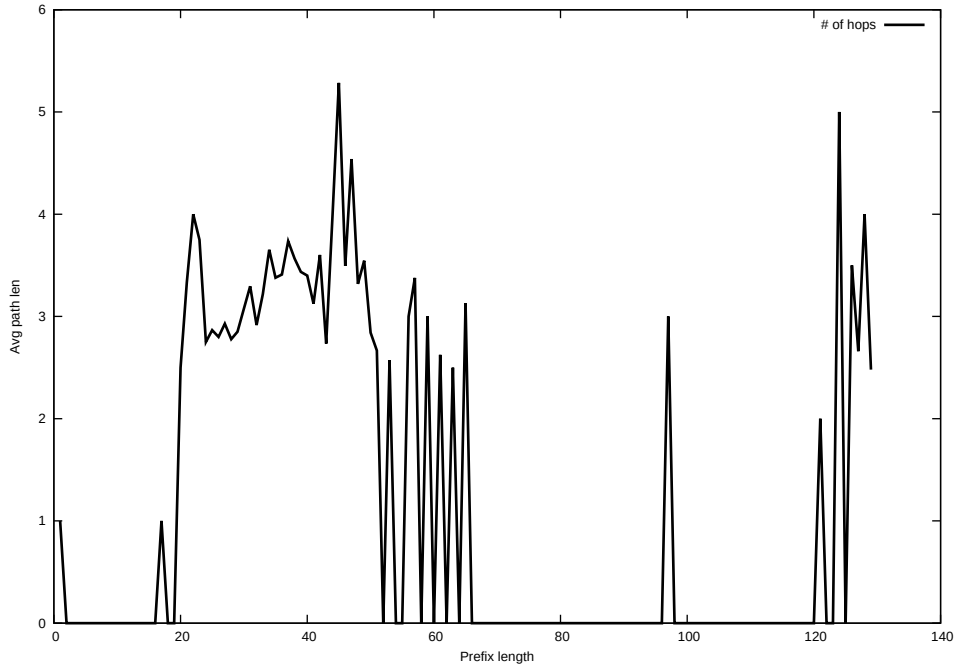
2015-02-20



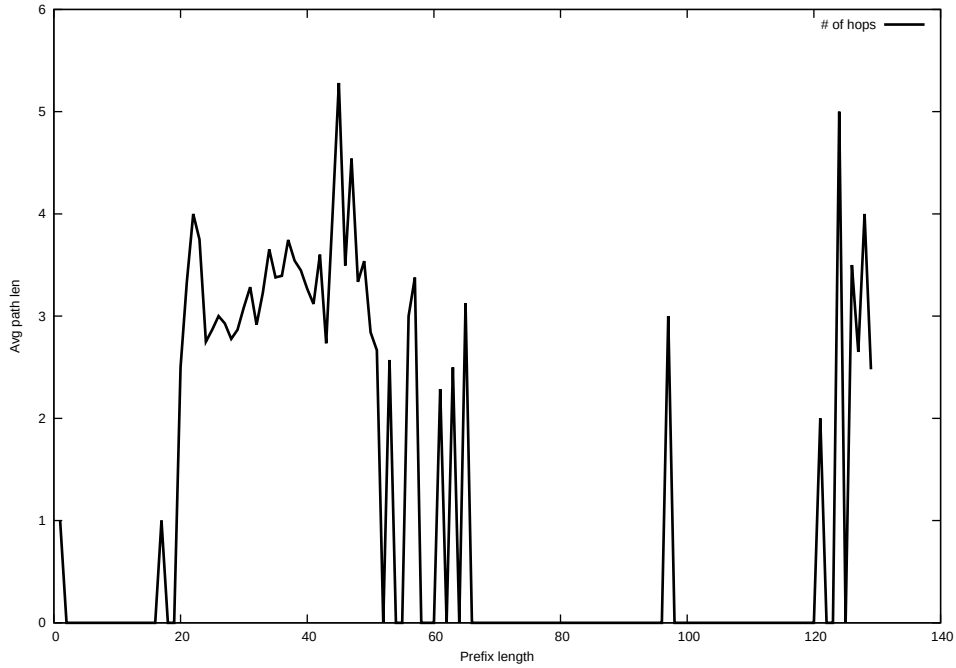
2015-02-21



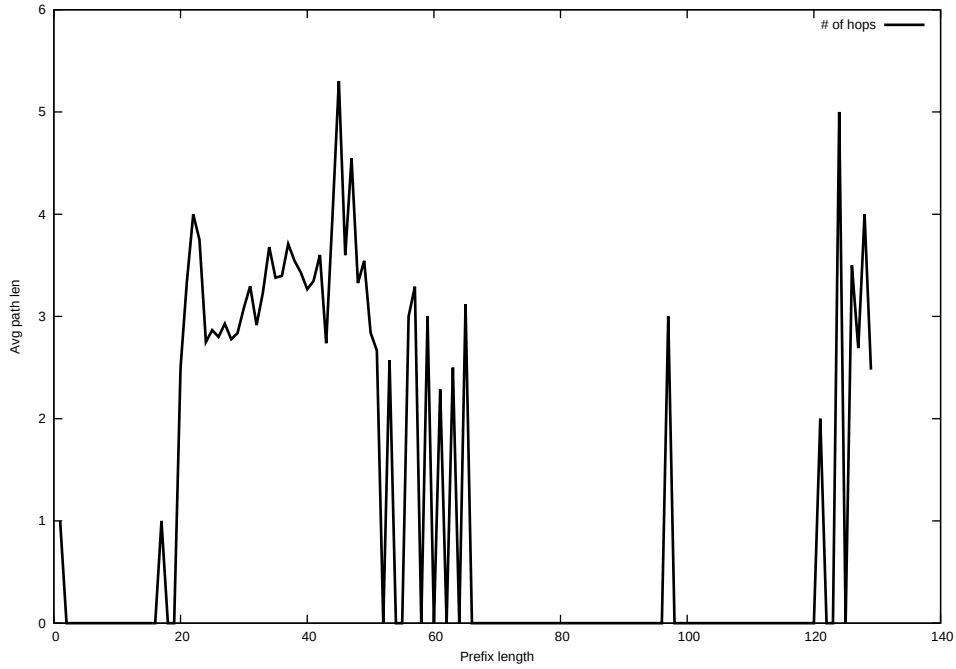
2015-02-22



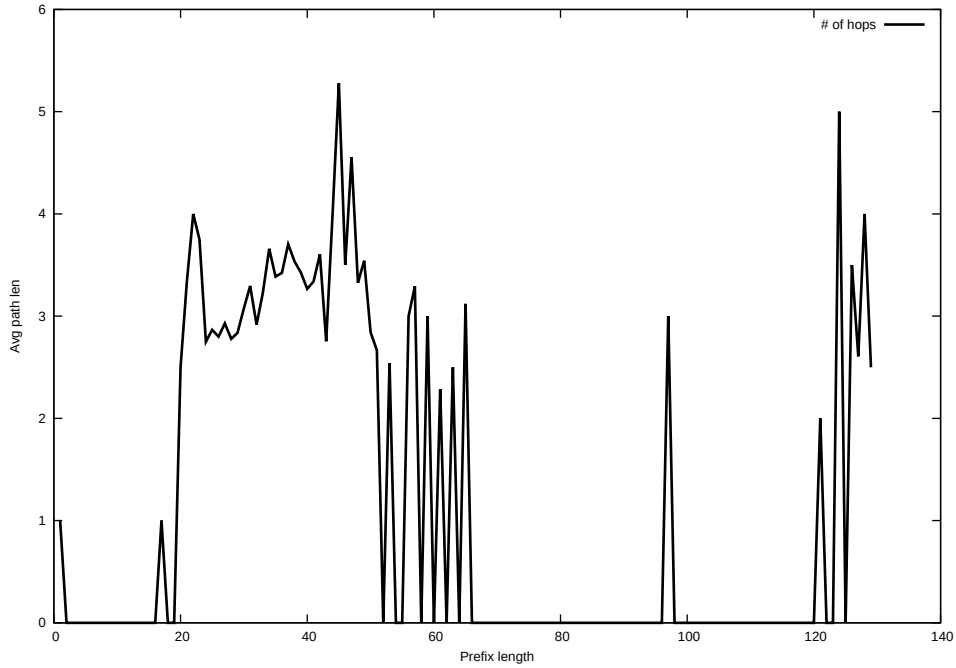
2015-02-23



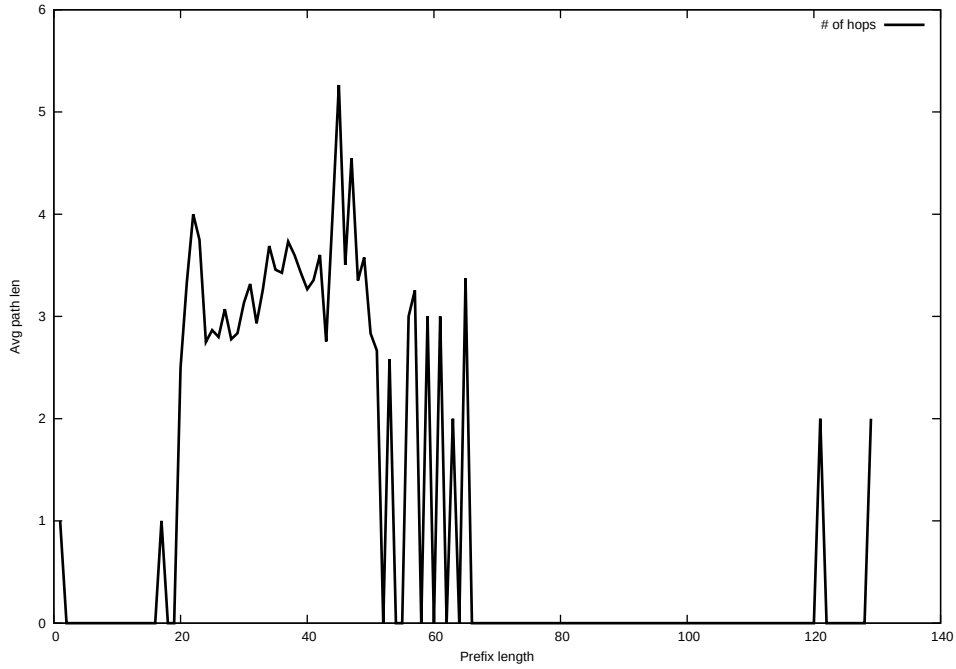
2015-02-24



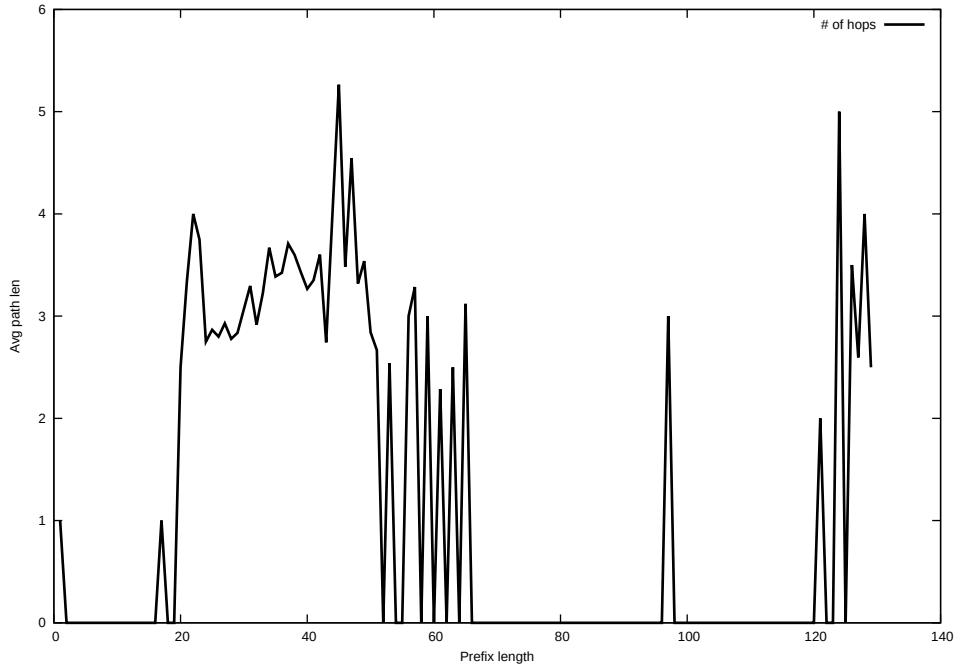
2015-02-25



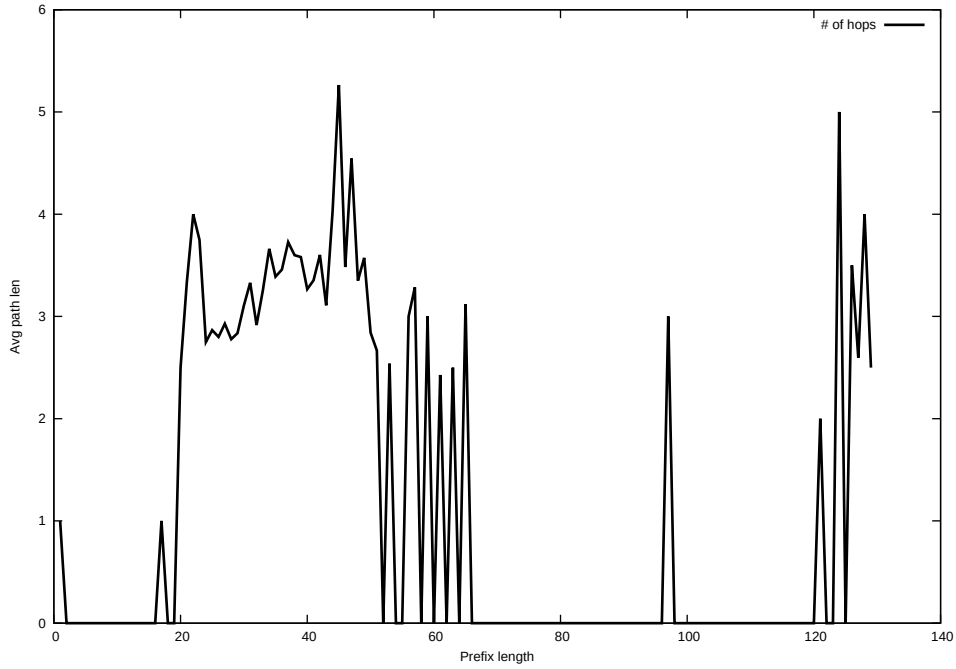
2015-02-26



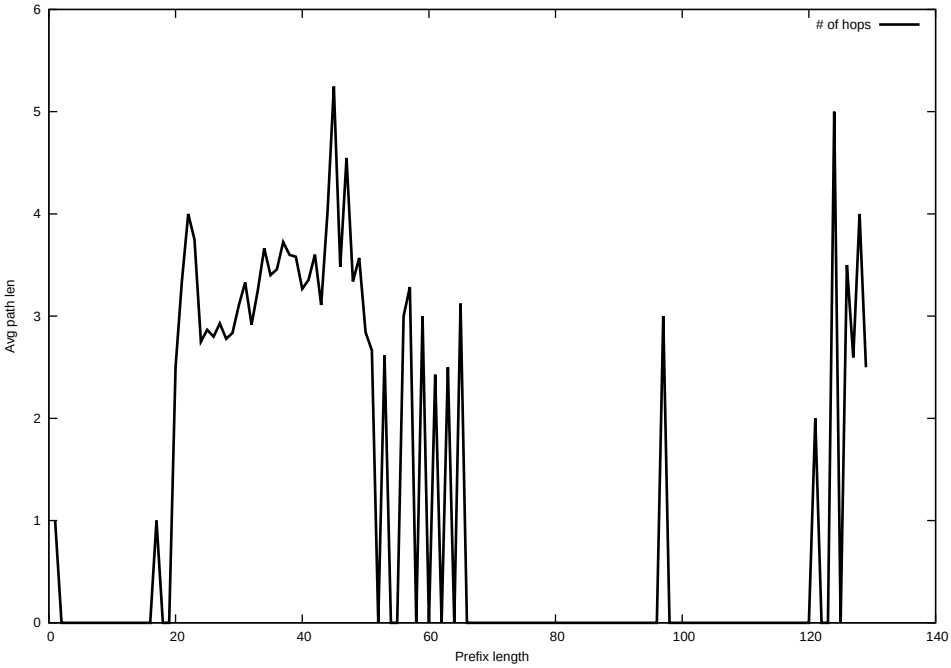
2015-02-27



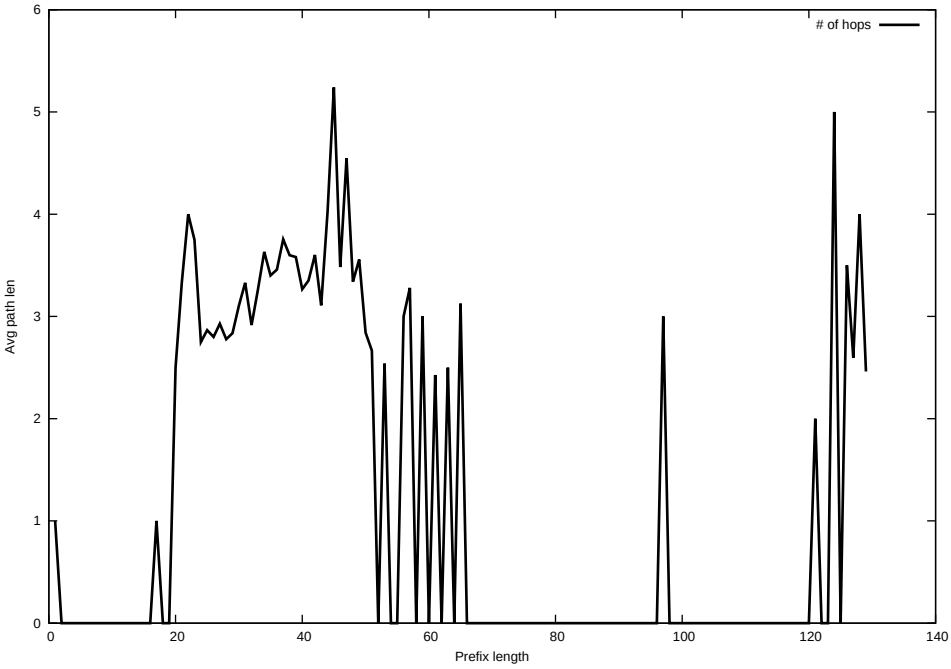
2015-02-28



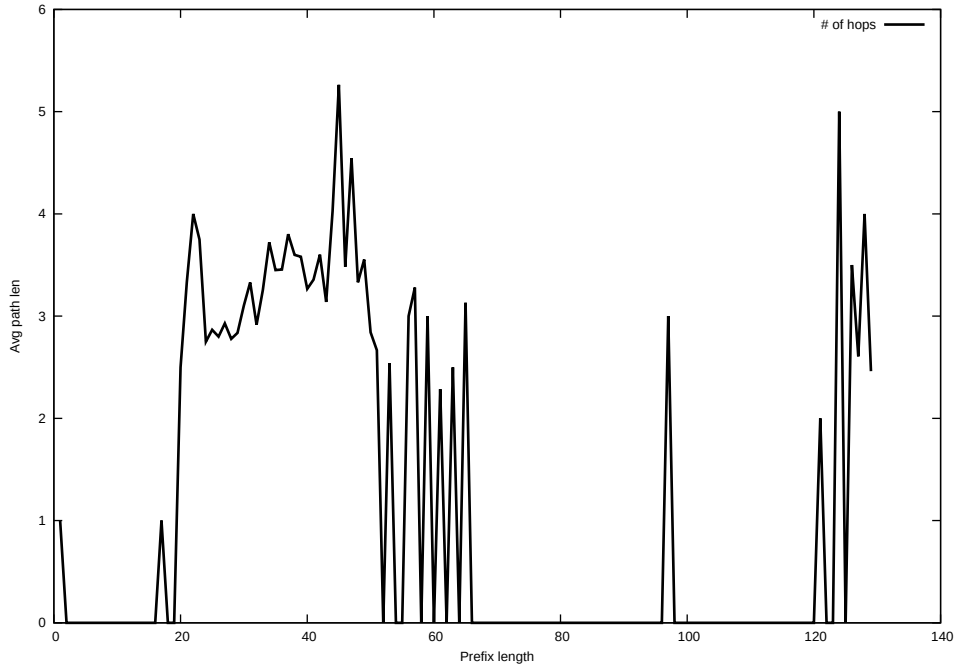
2015-03-01



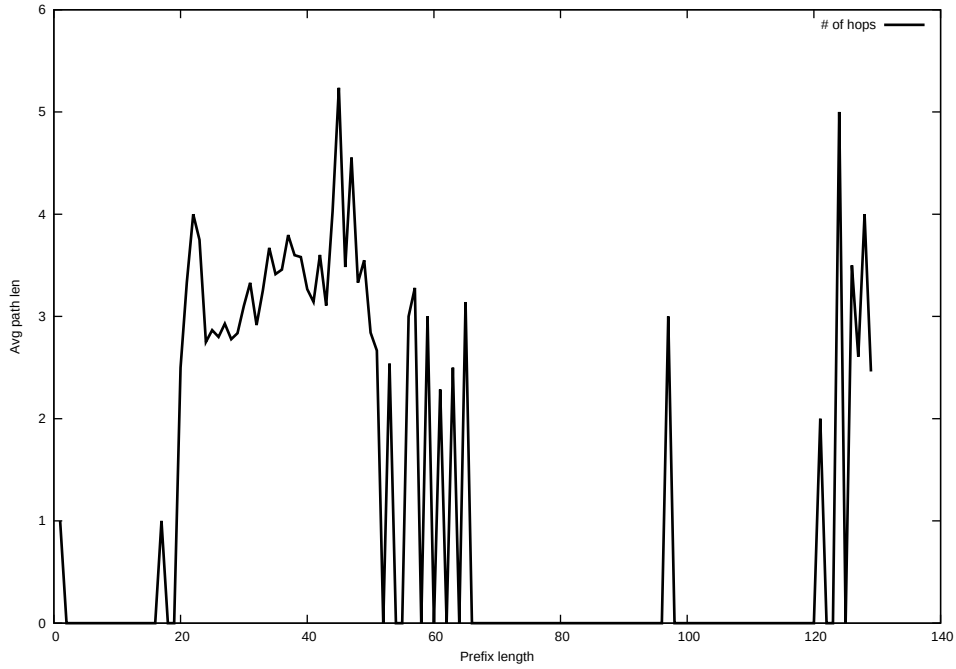
2015-03-02



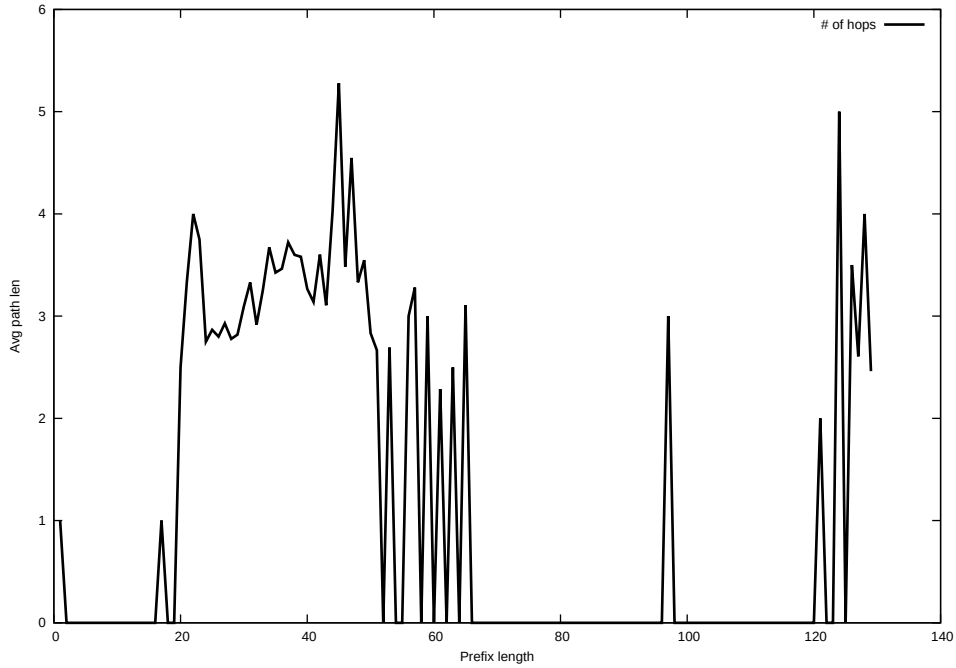
2015-03-03



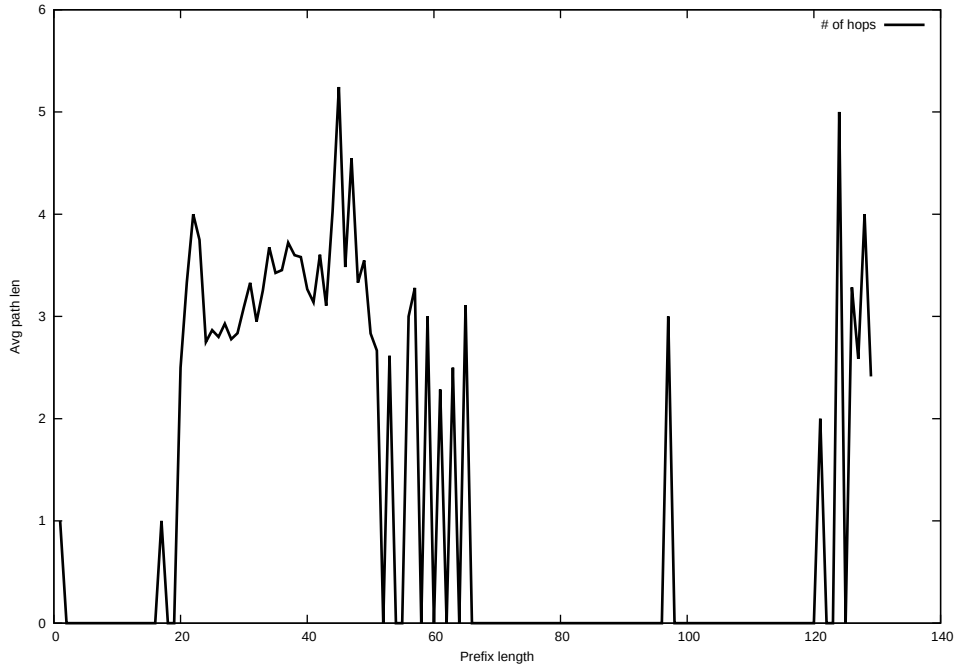
2015-03-04



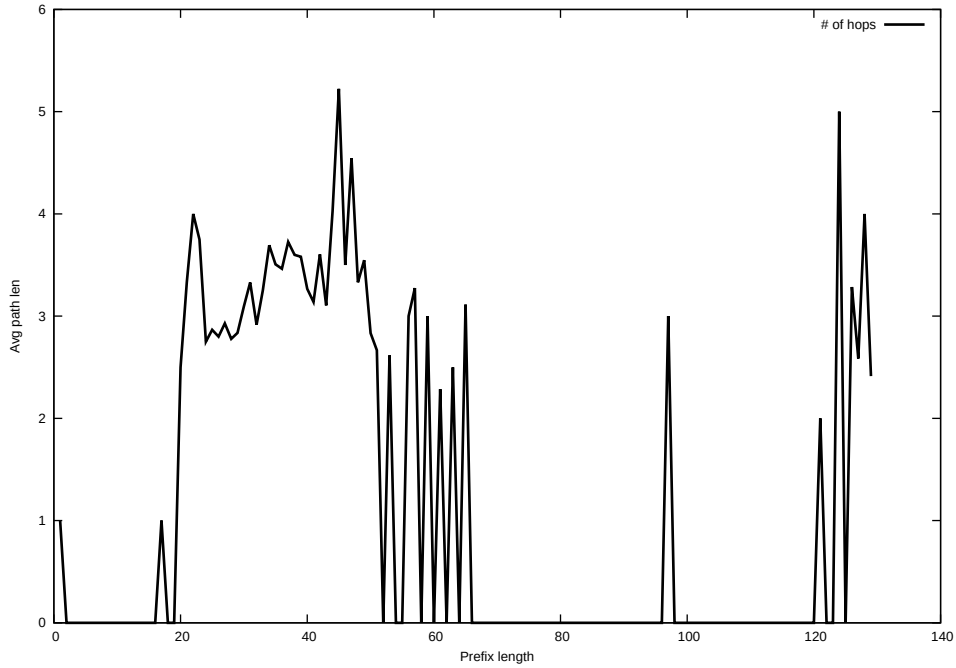
2015-03-05



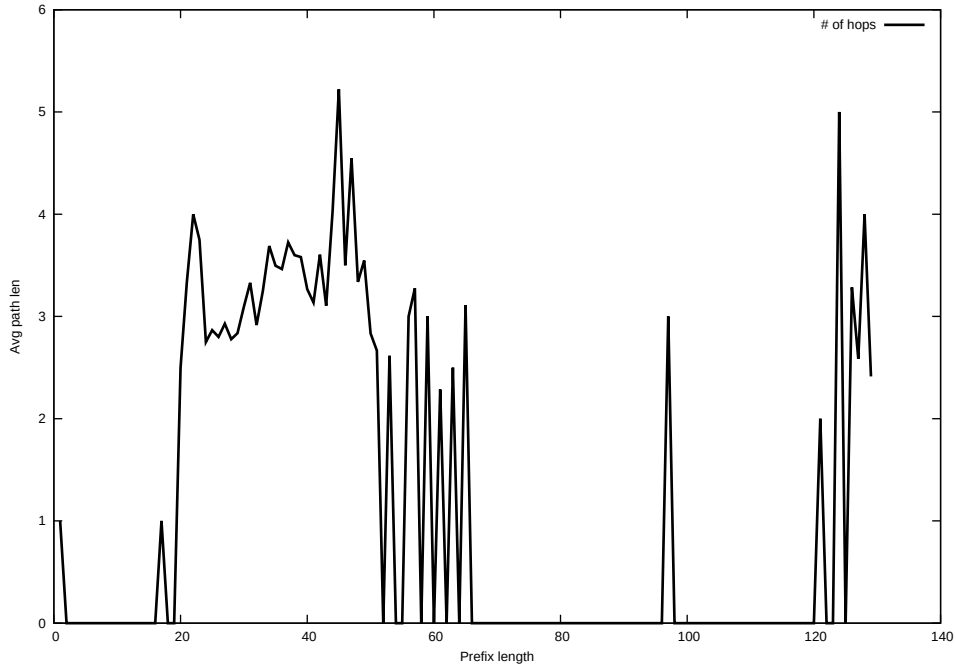
2015-03-06



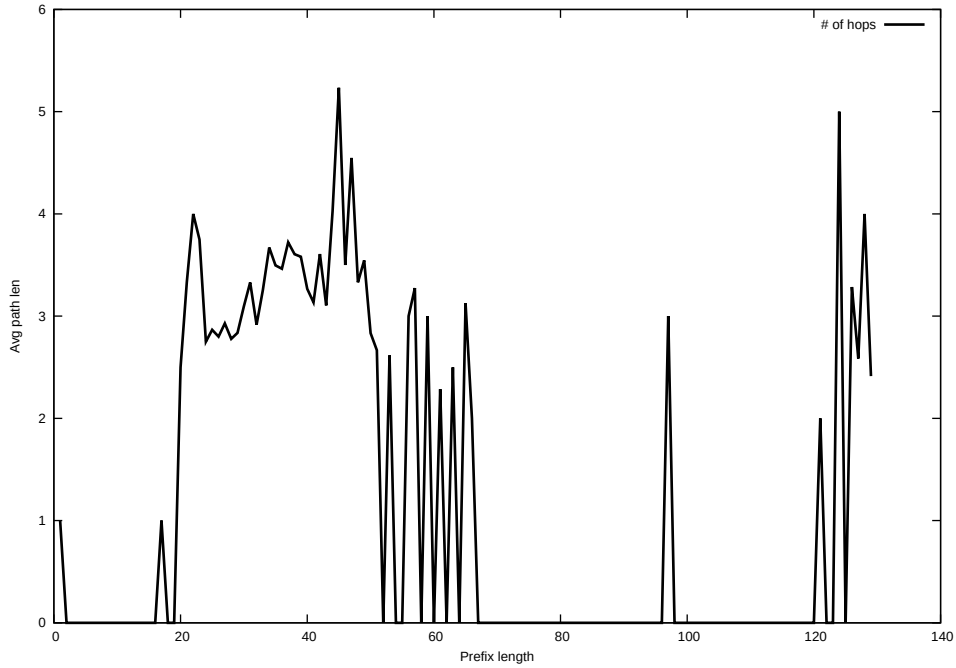
2015-03-07



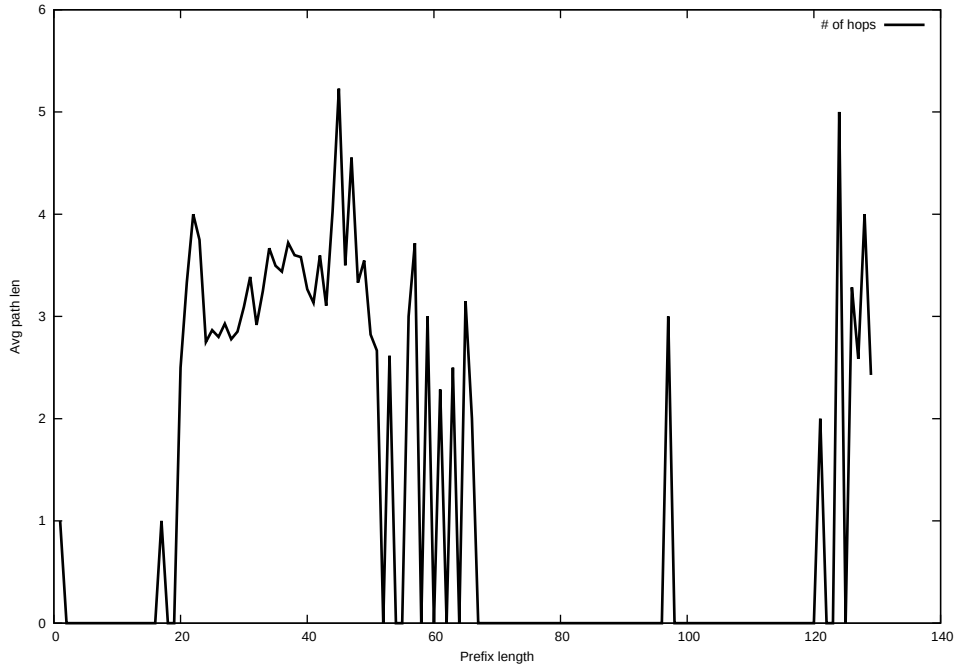
2015-03-08



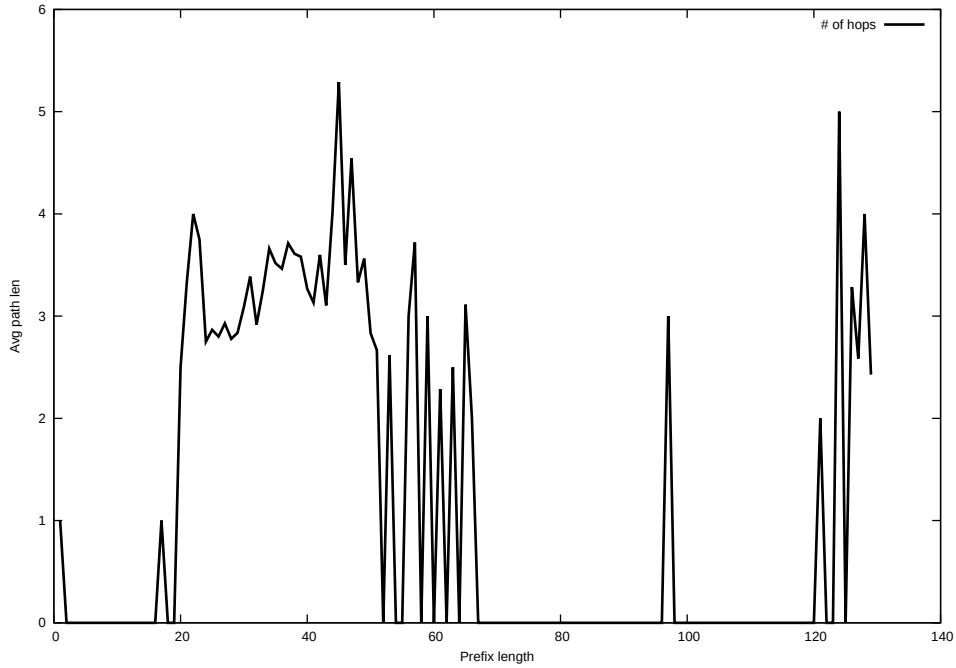
2015-03-09



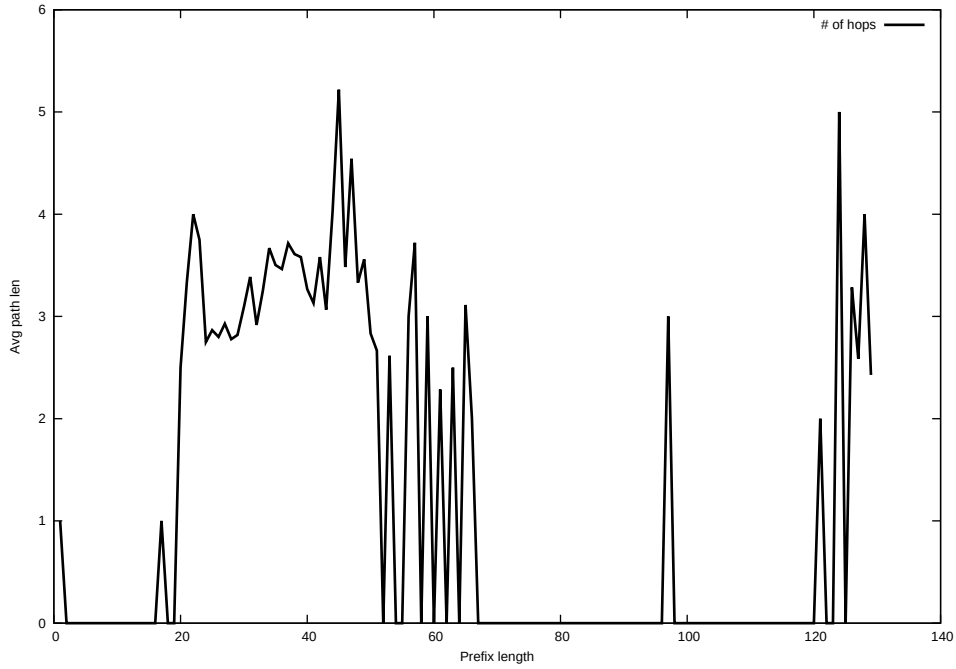
2015-03-10



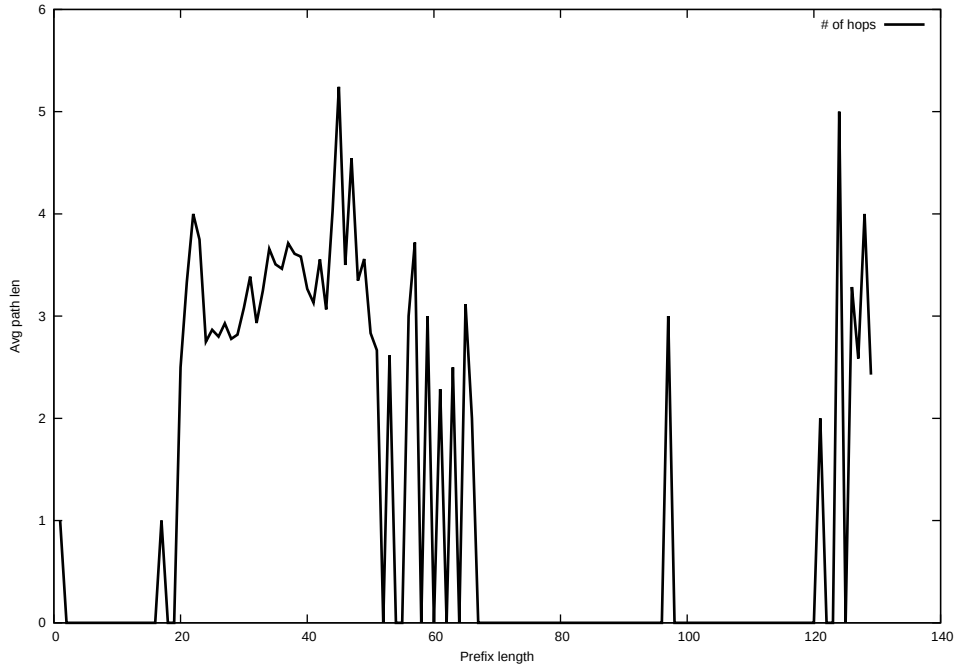
2015-03-11



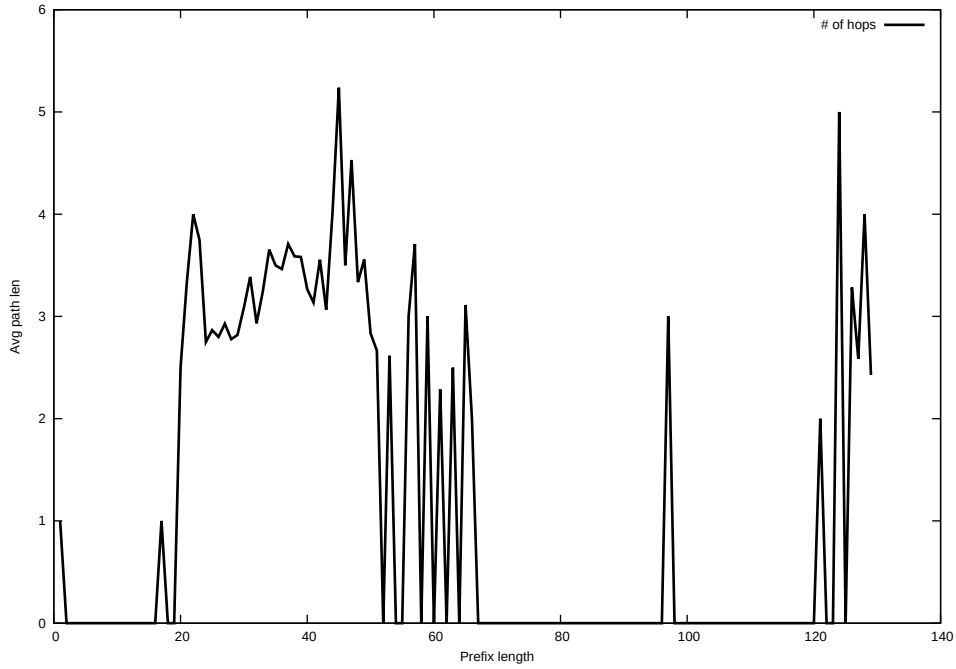
2015-03-12



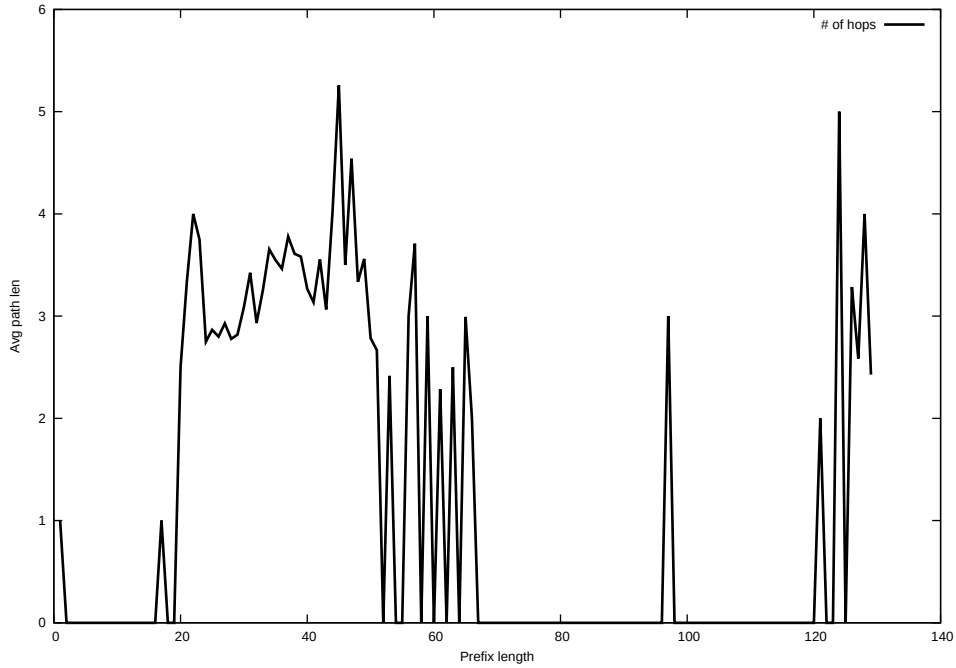
2015-03-13



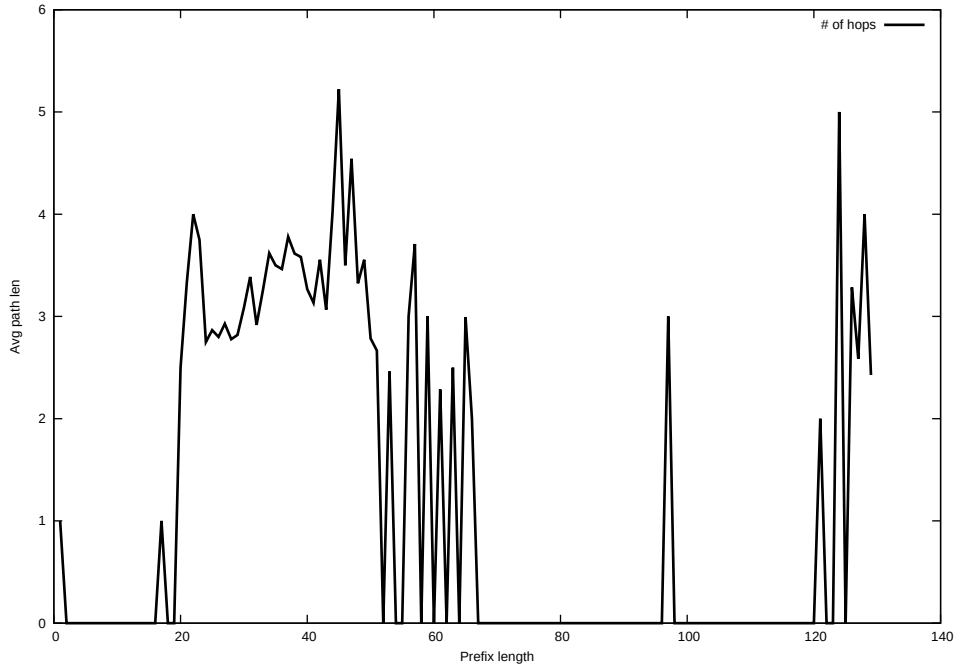
2015-03-14



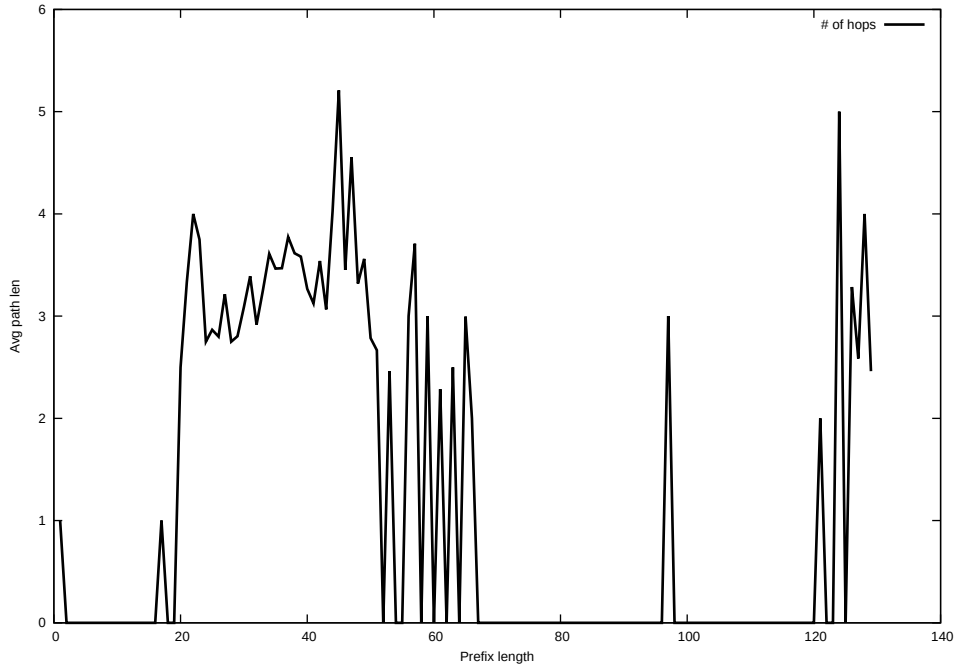
2015-03-15



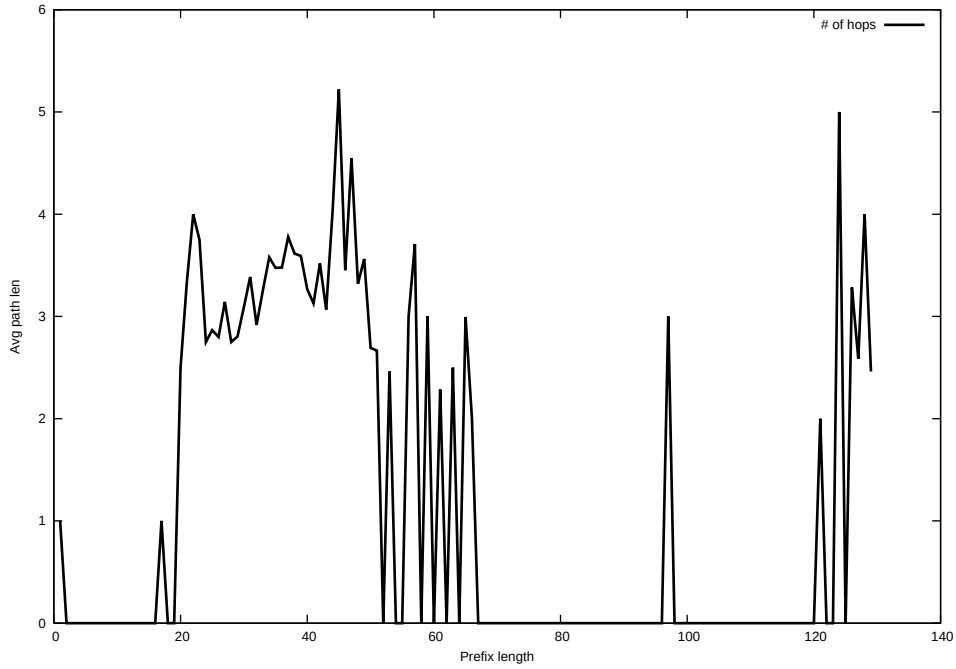
2015-03-16



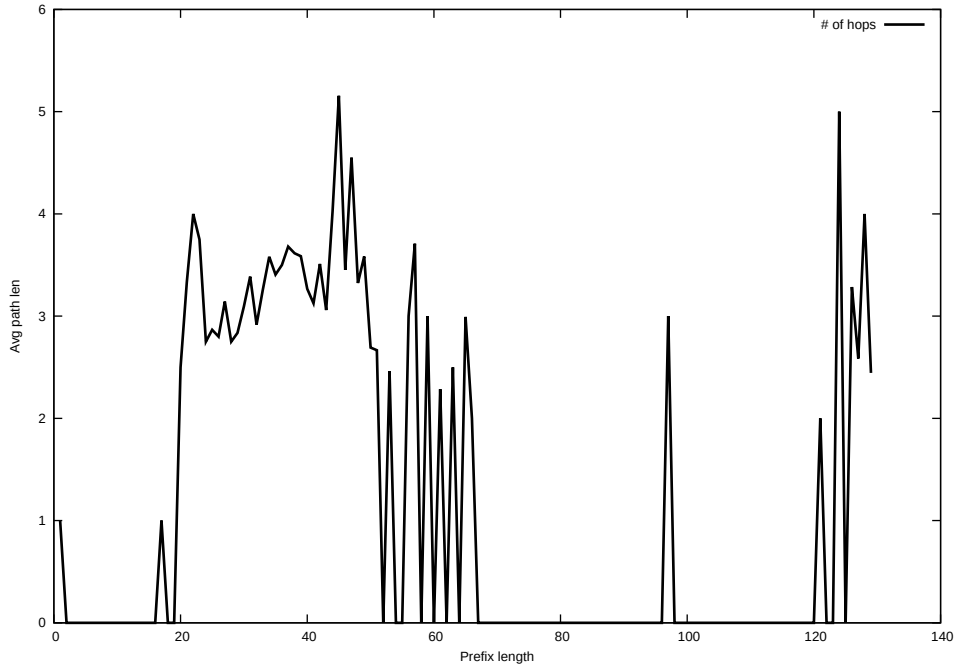
2015-03-17



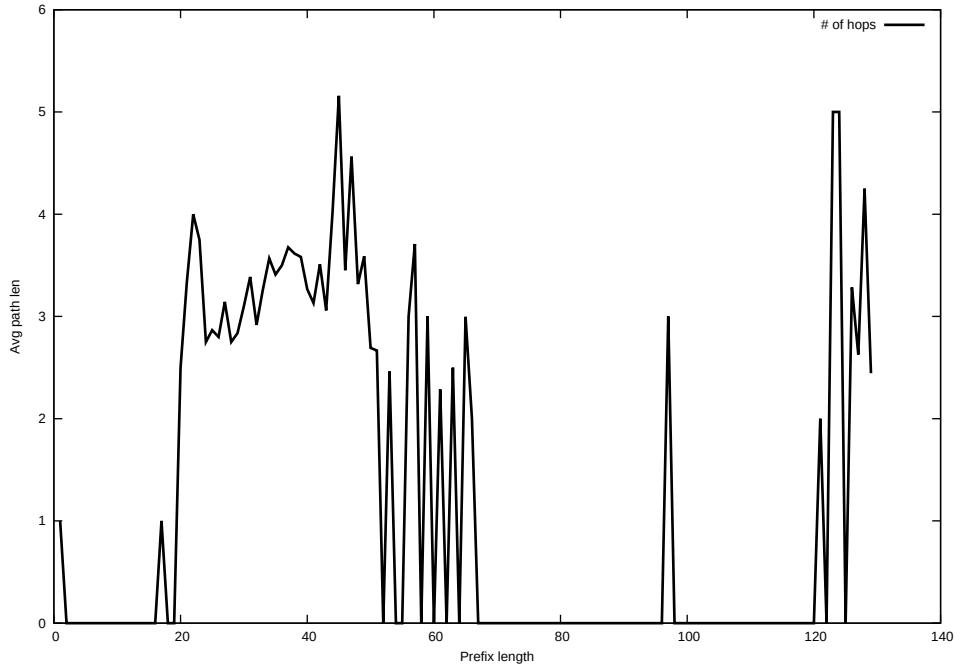
2015-03-18



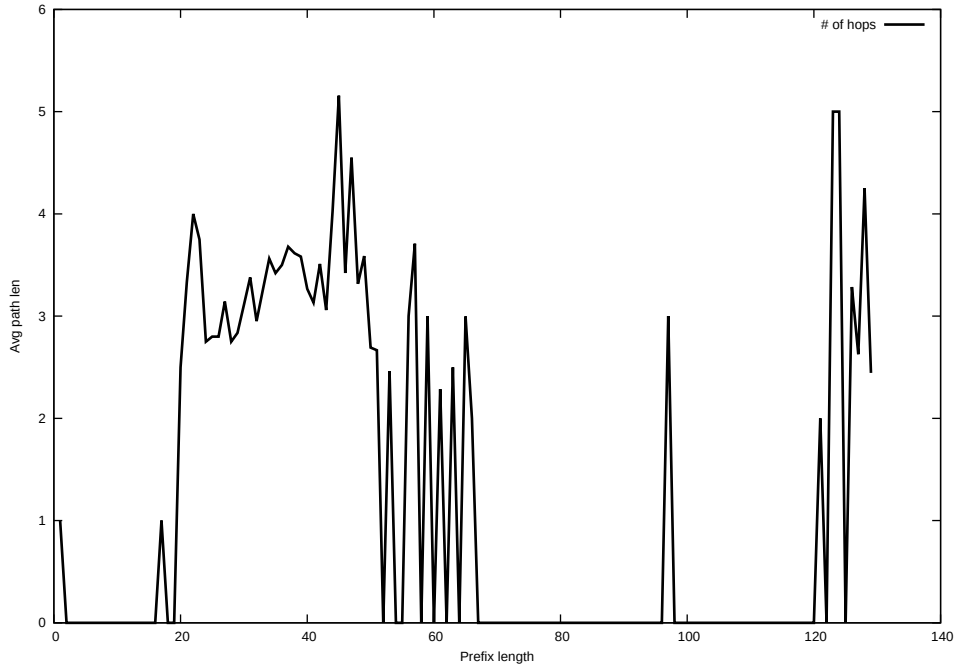
2015-03-19



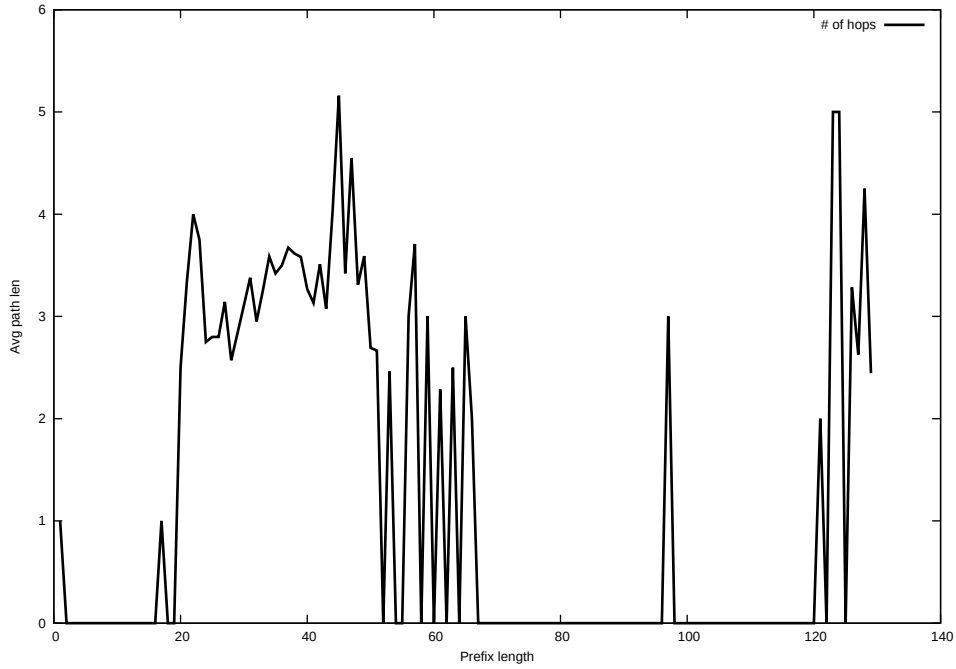
2015-03-20



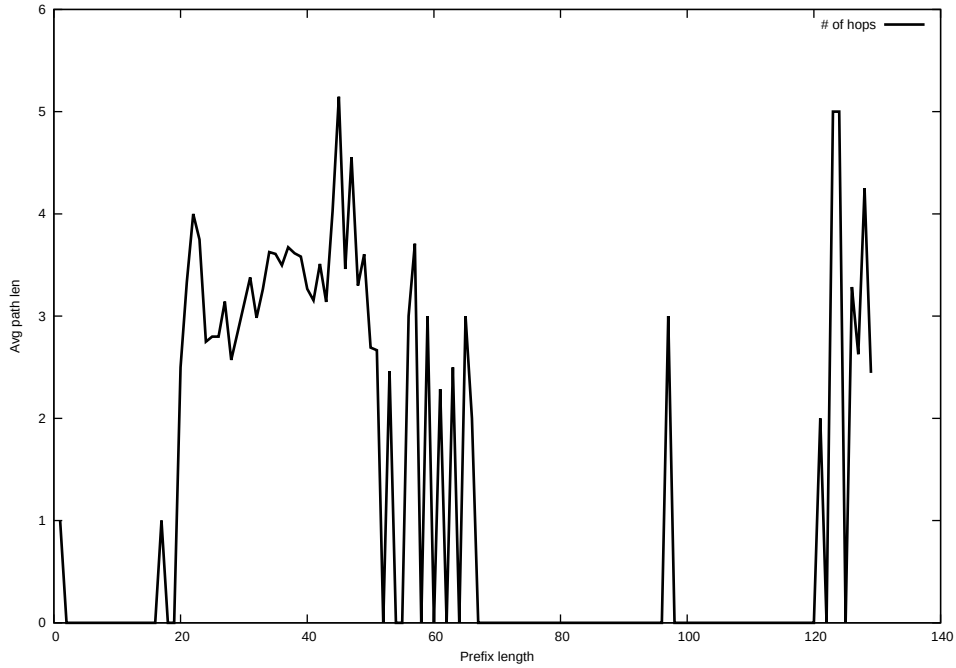
2015-03-21



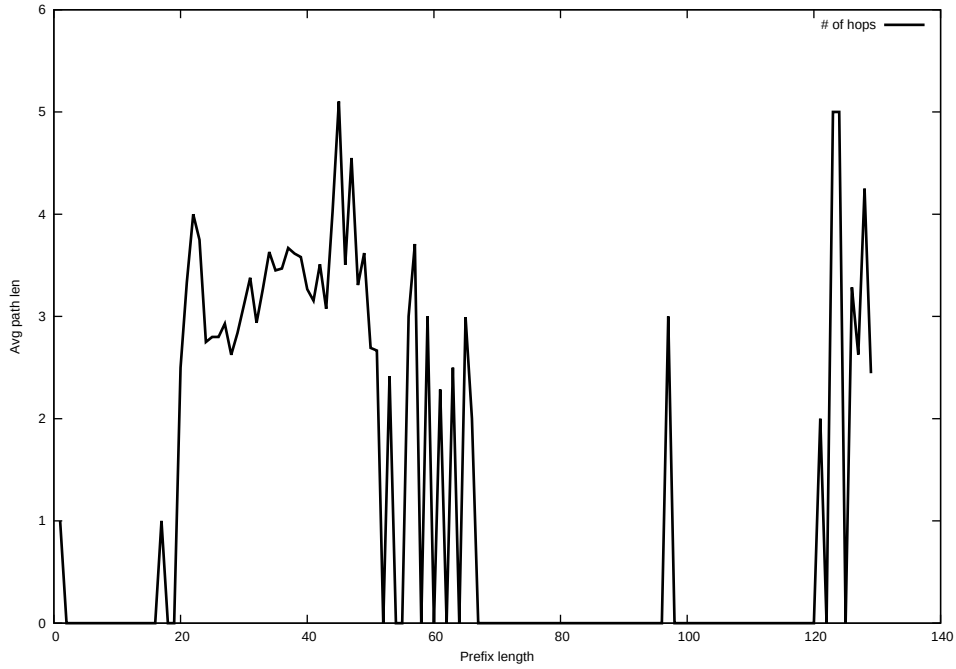
2015-03-22



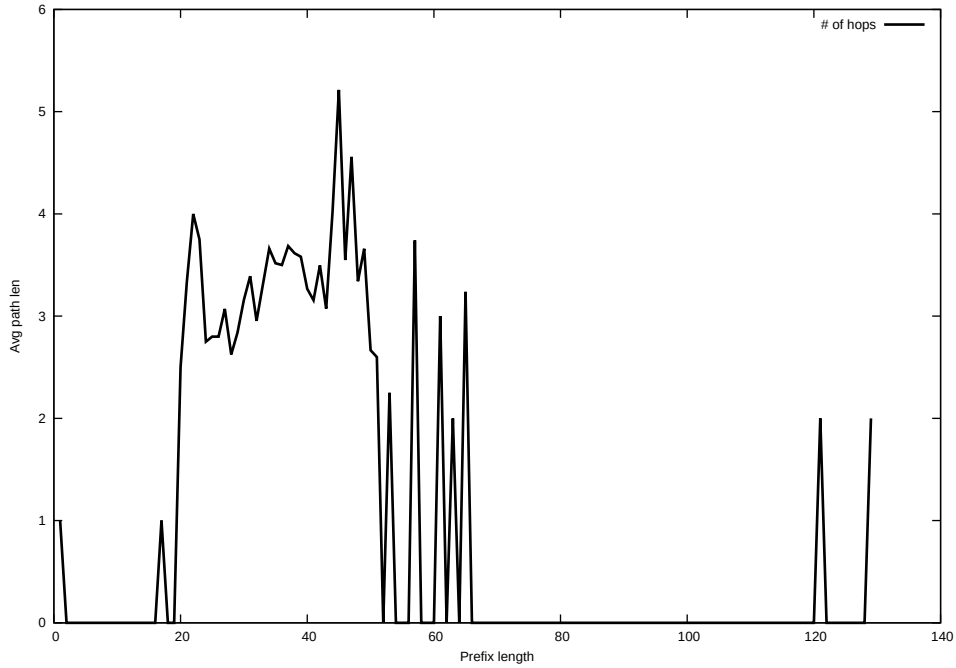
2015-03-23



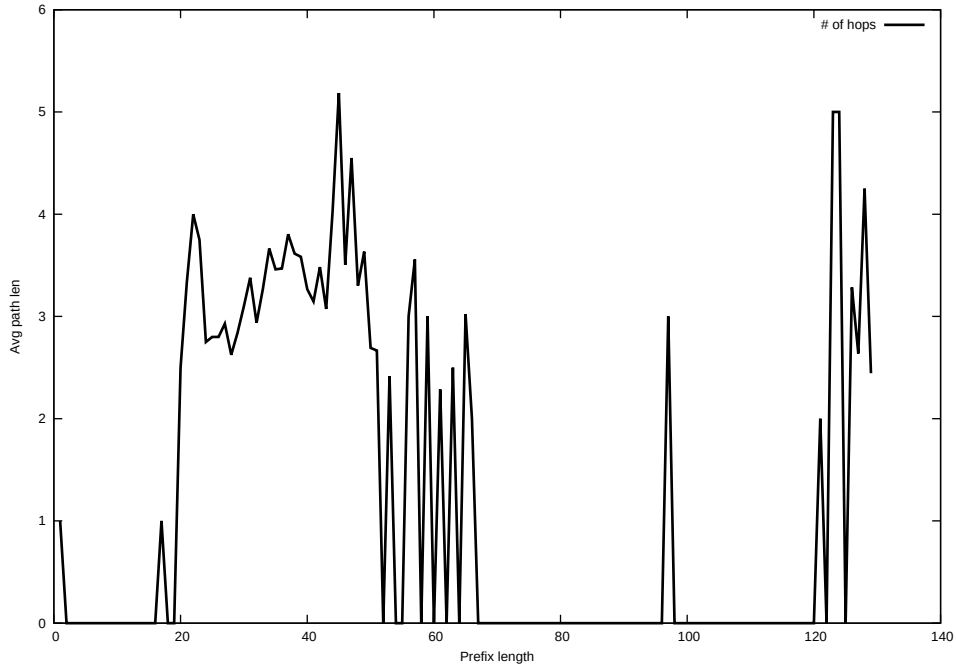
2015-03-24



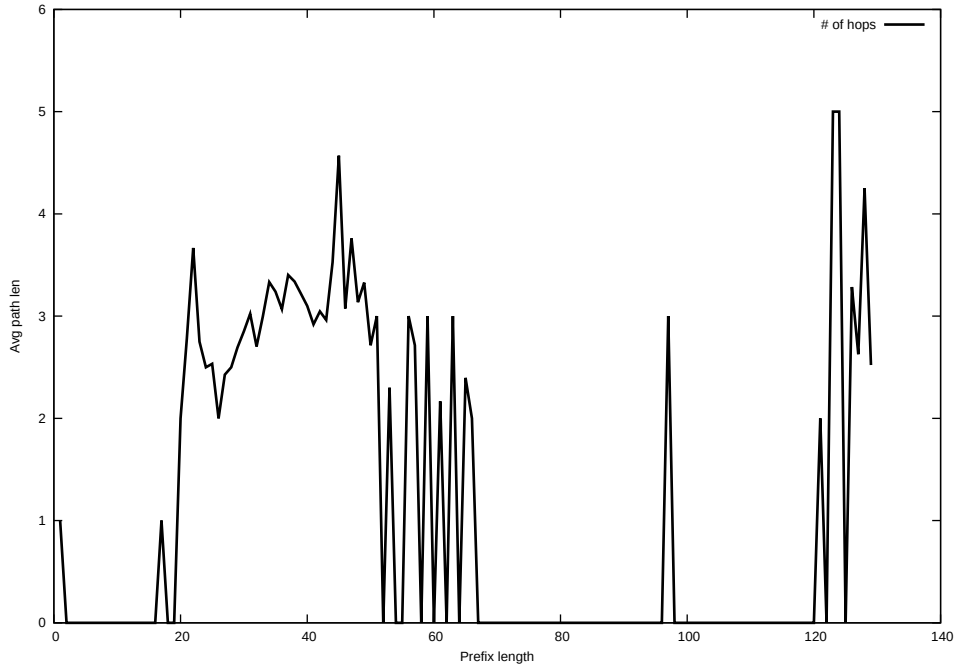
2015-03-25



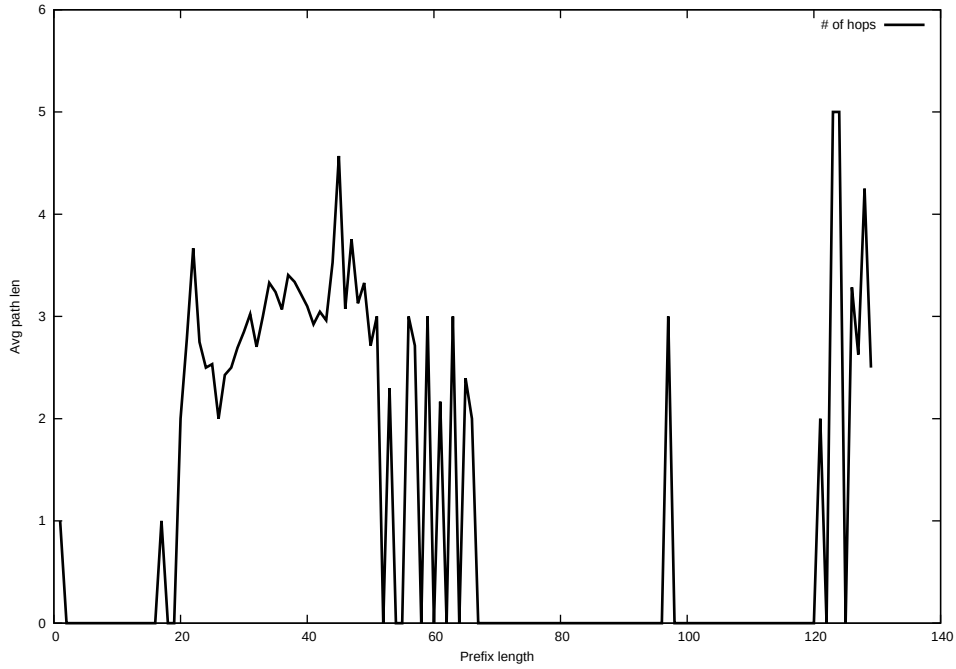
2015-03-26



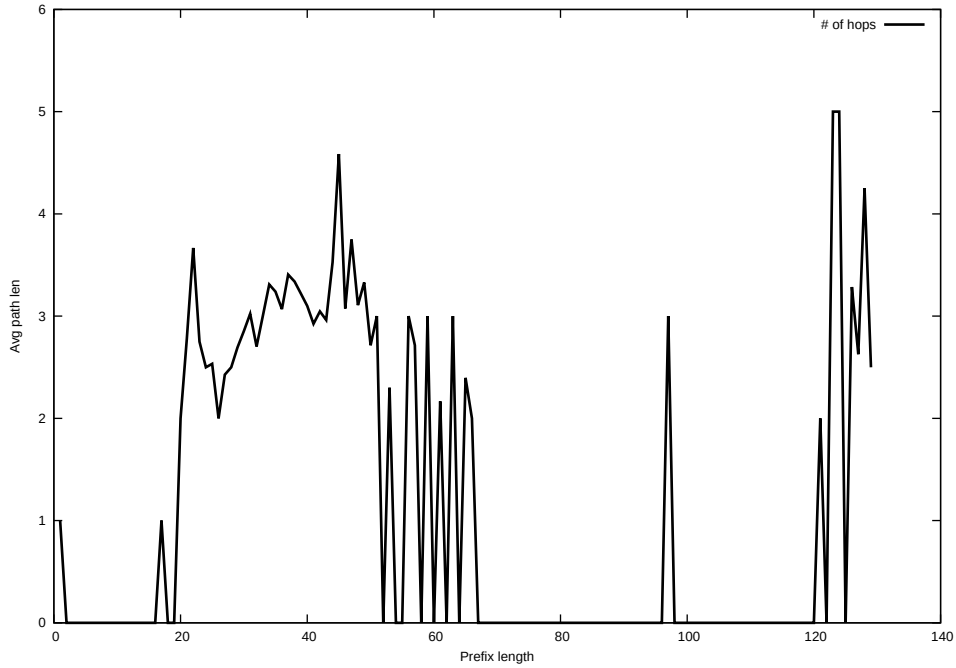
2015-03-27



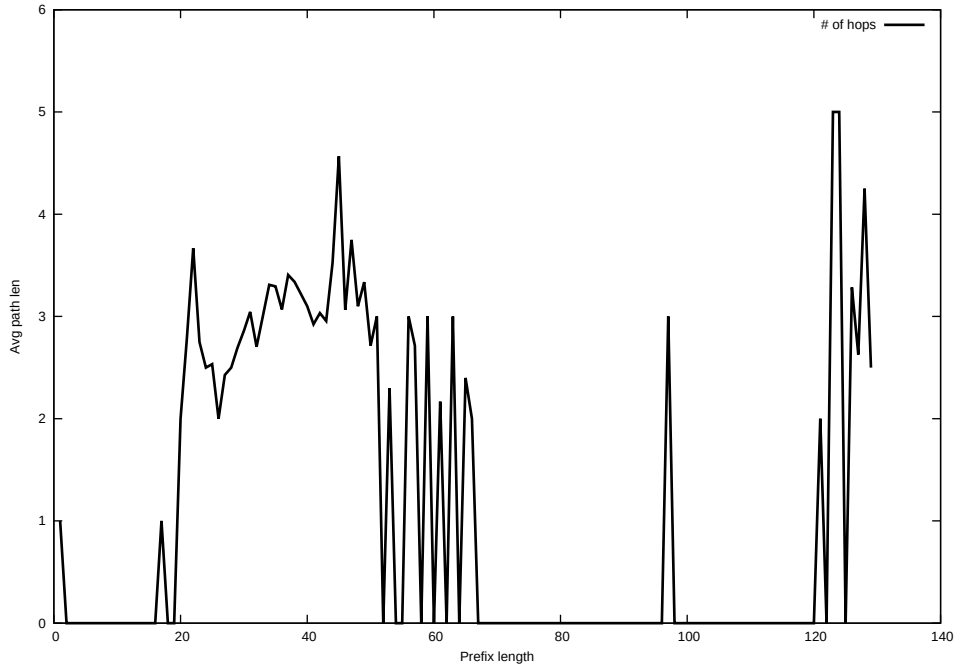
2015-03-28



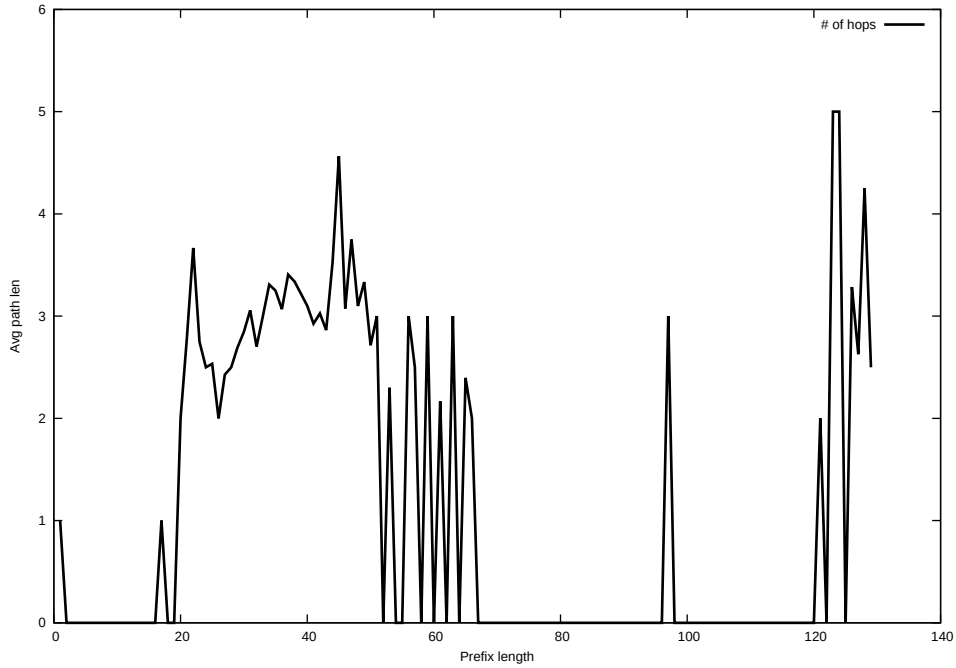
2015-03-29



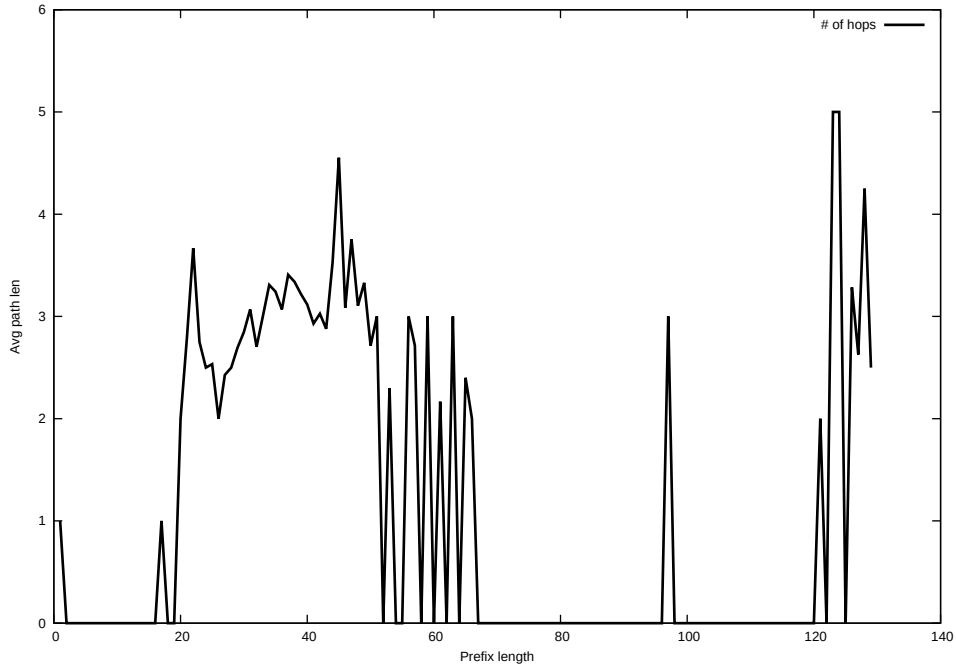
2015-03-30



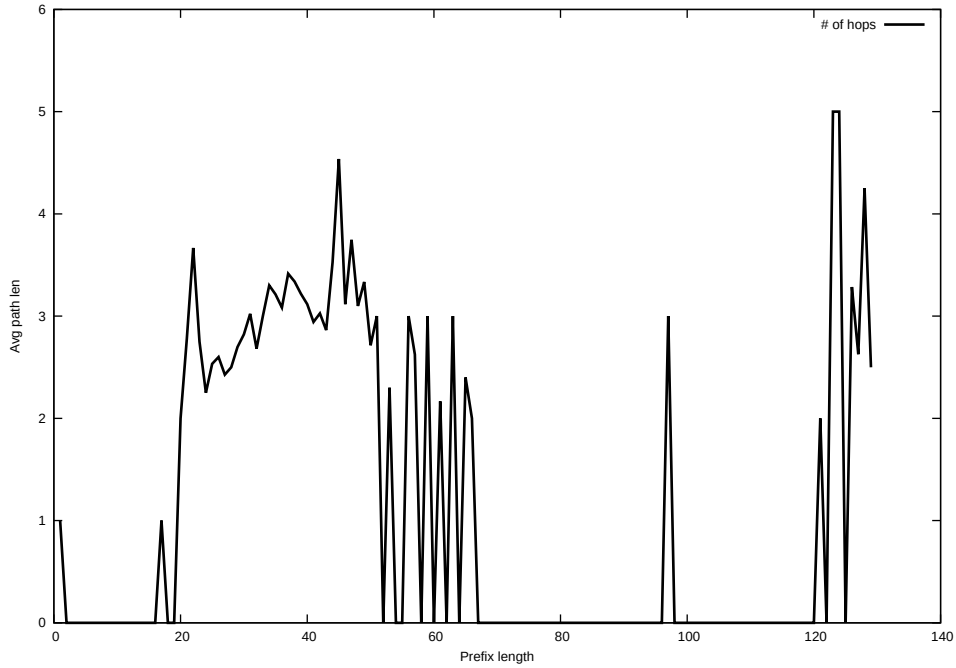
2015-03-31



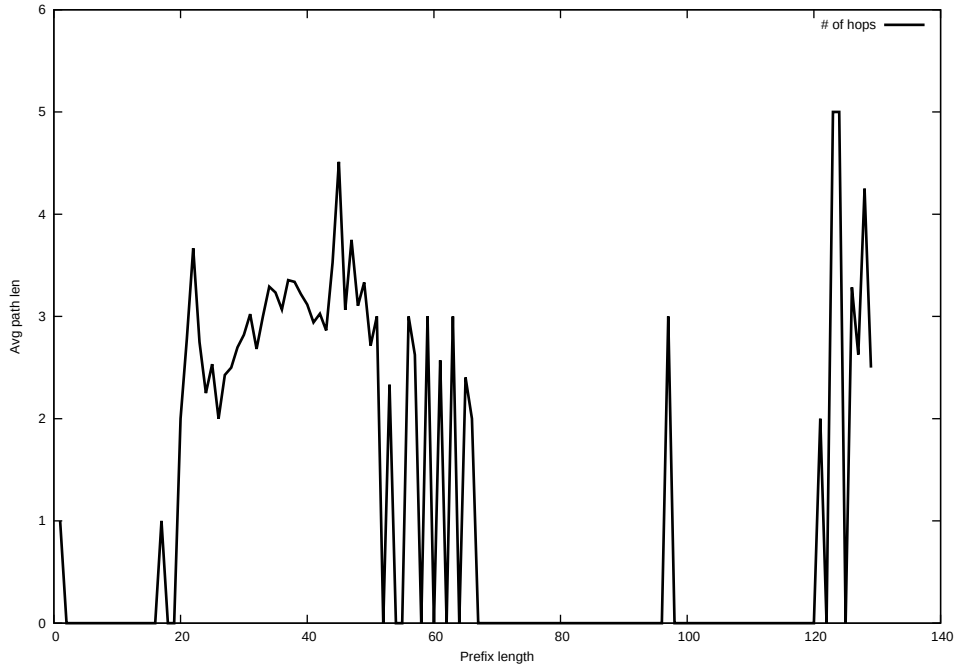
2015-04-01



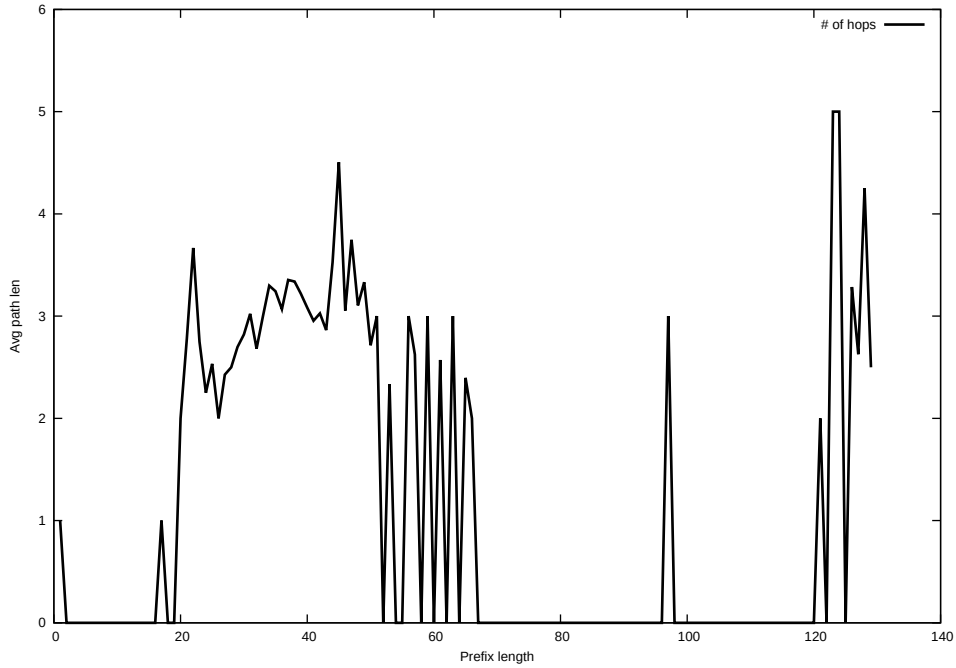
2015-04-02



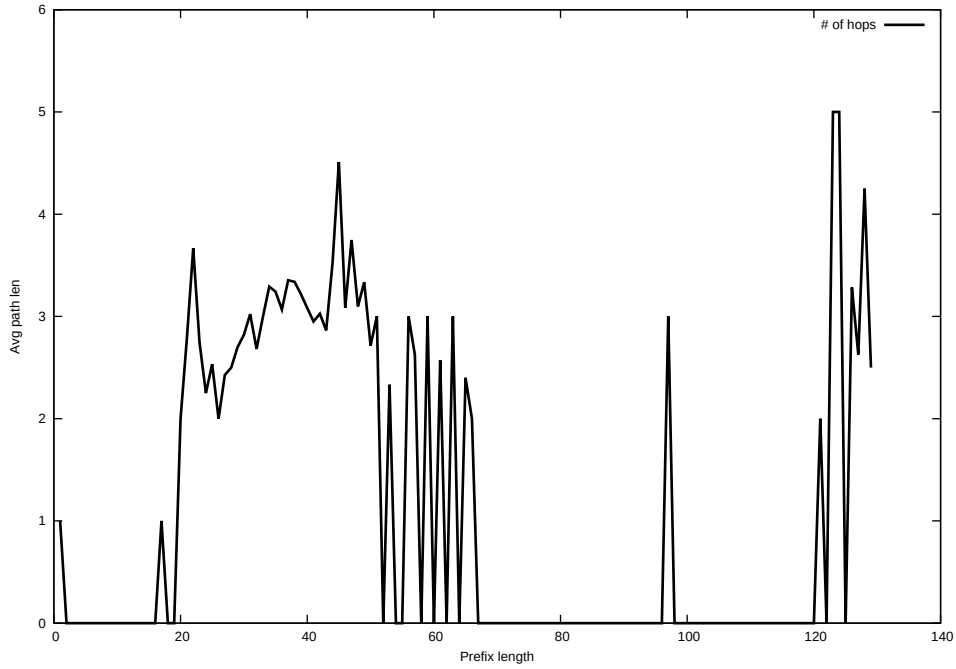
2015-04-03



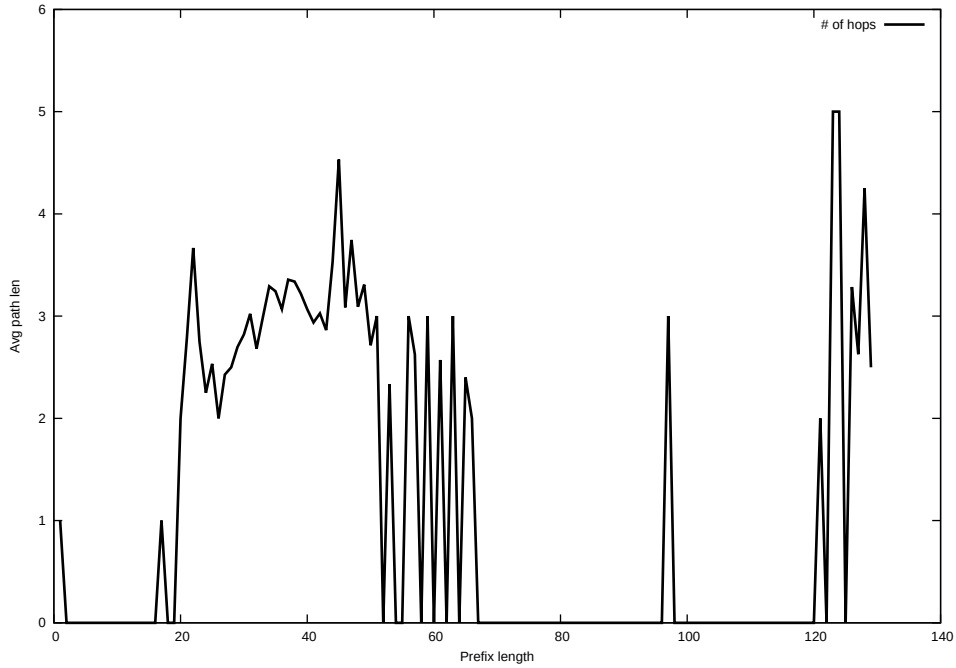
2015-04-04



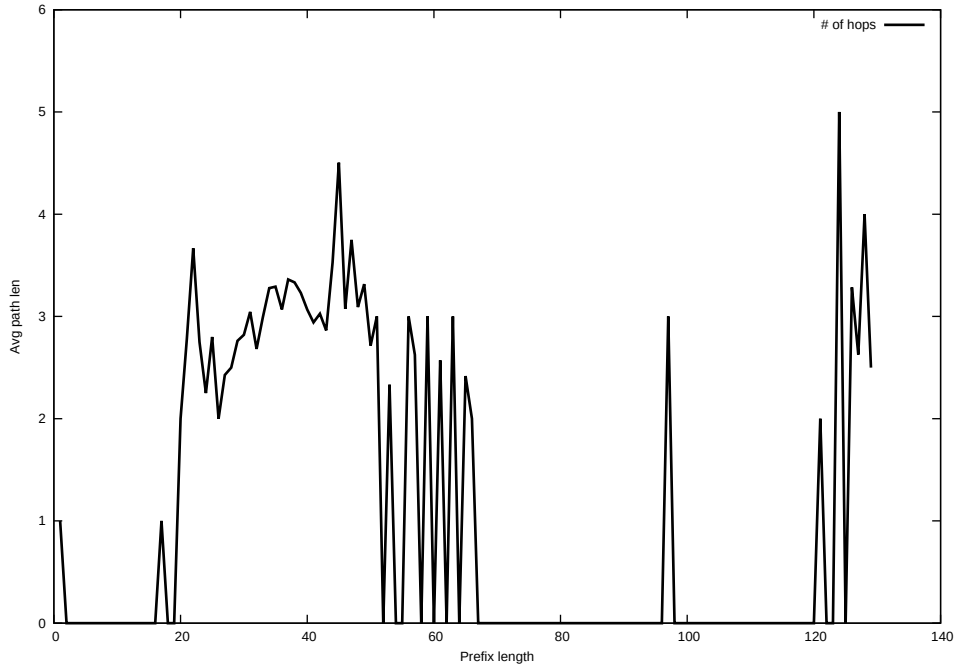
2015-04-05



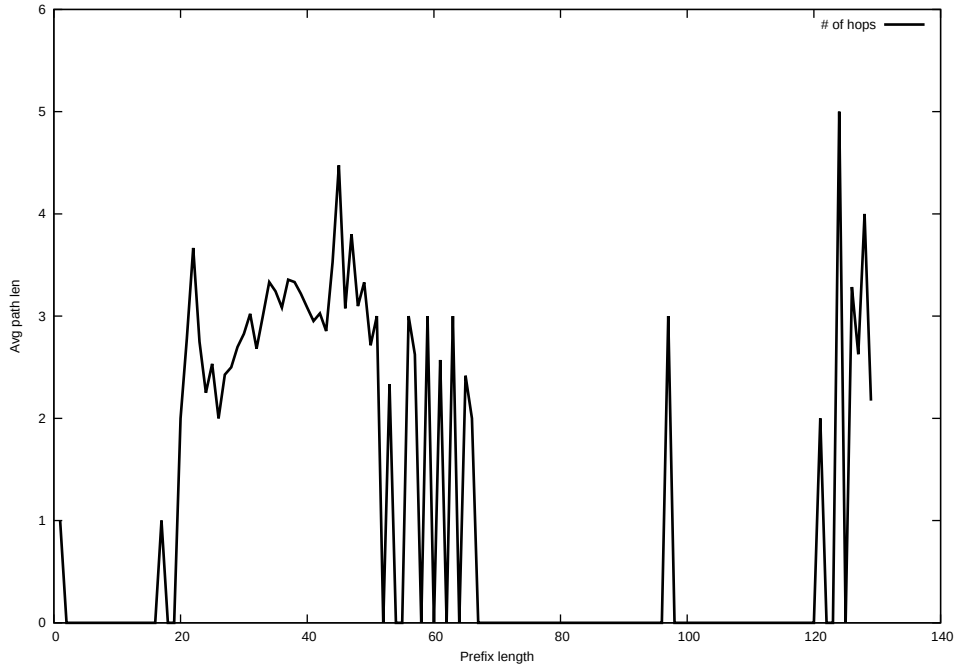
2015-04-06



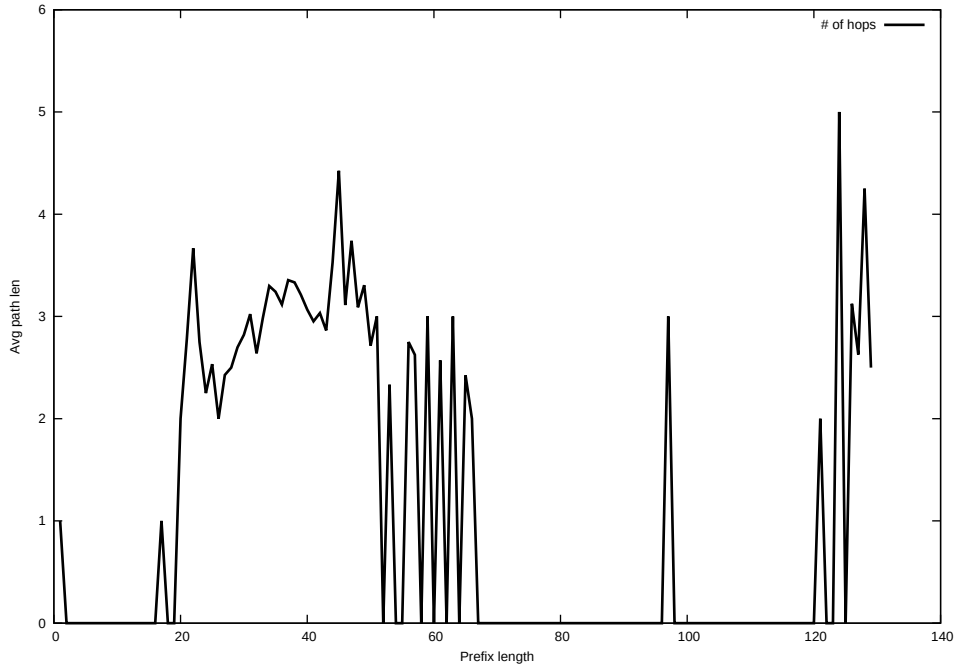
2015-04-07



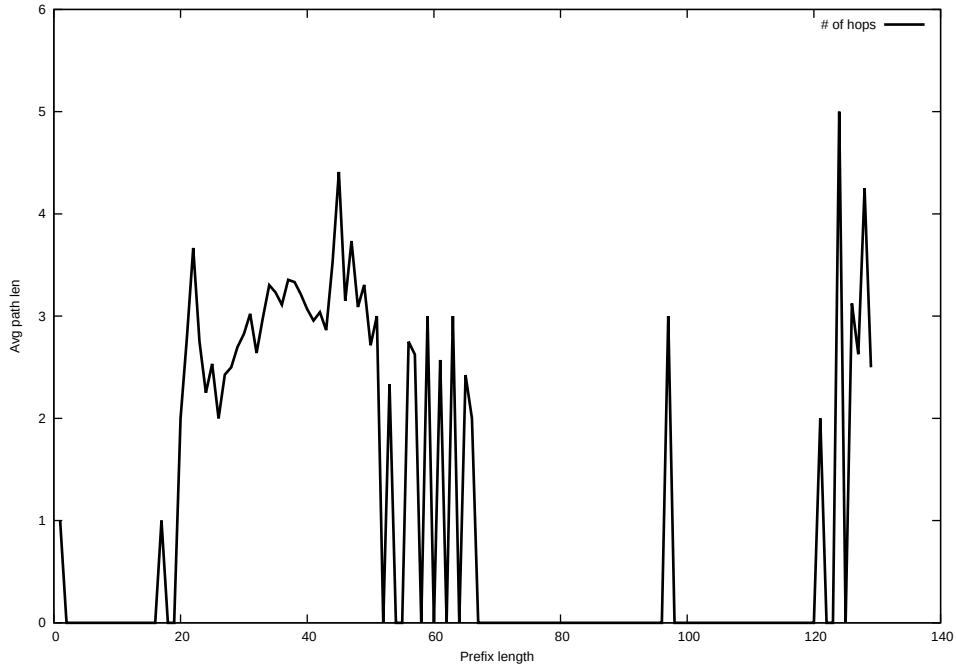
2015-04-08



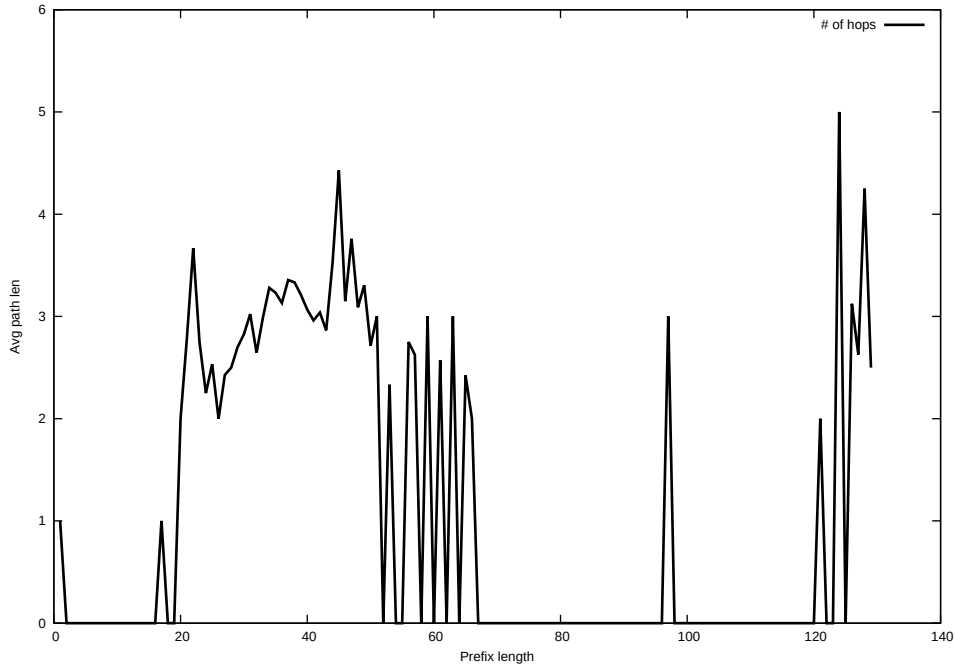
2015-04-09



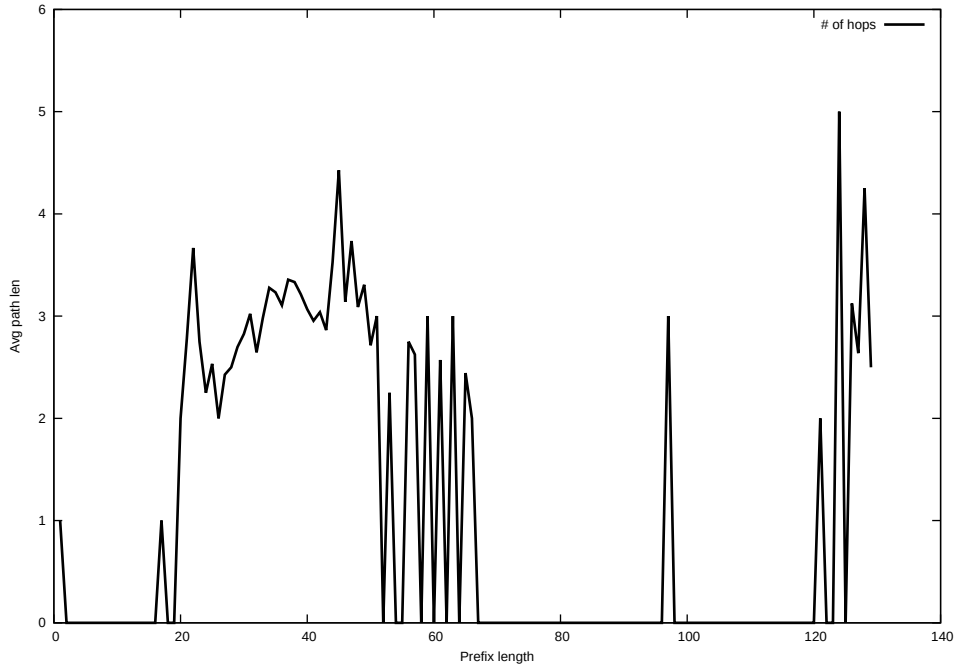
2015-04-10



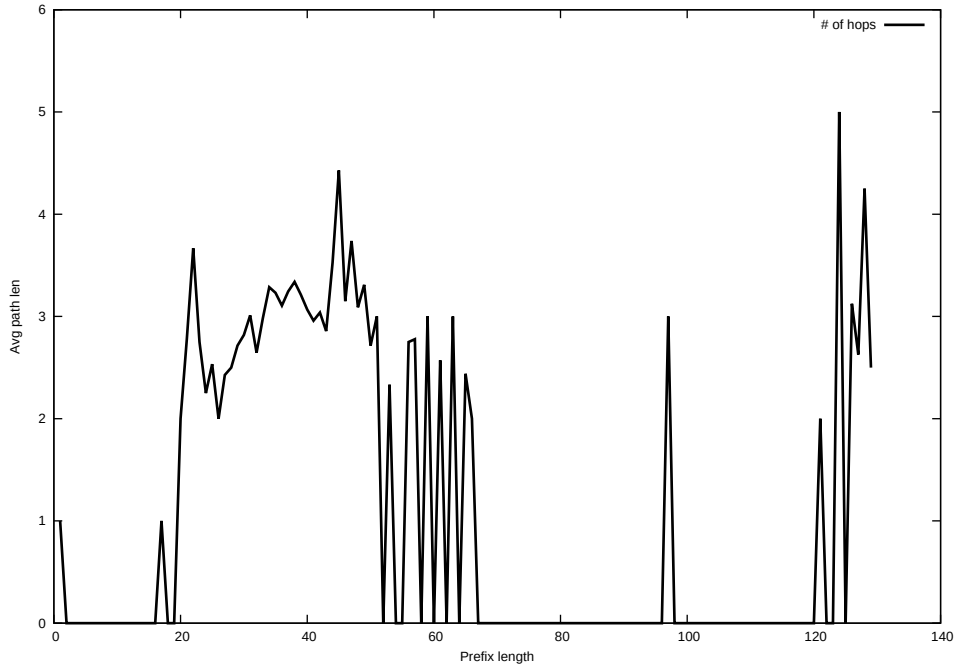
2015-04-11



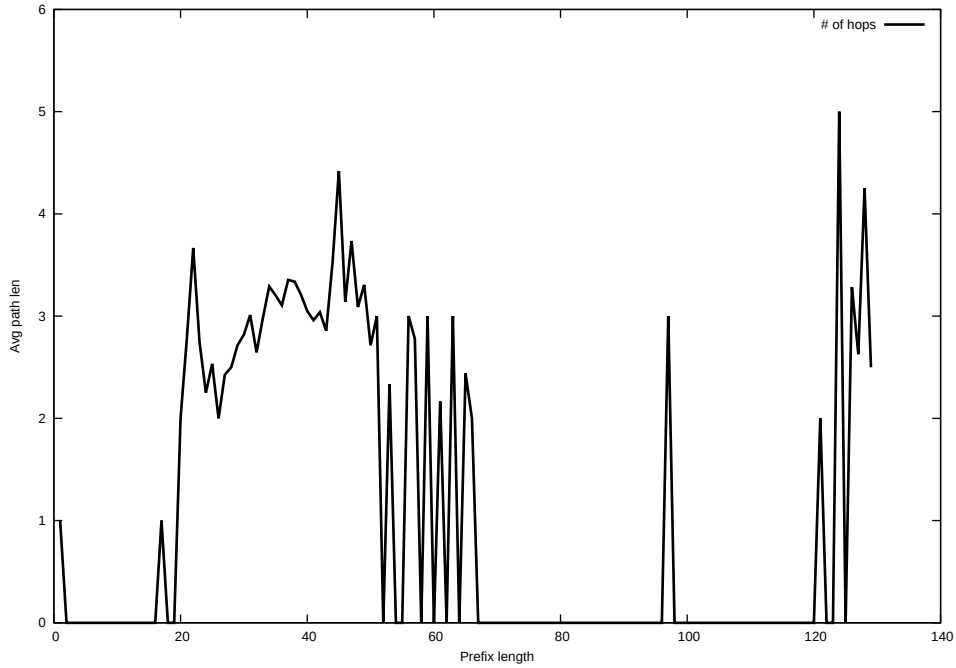
2015-04-12



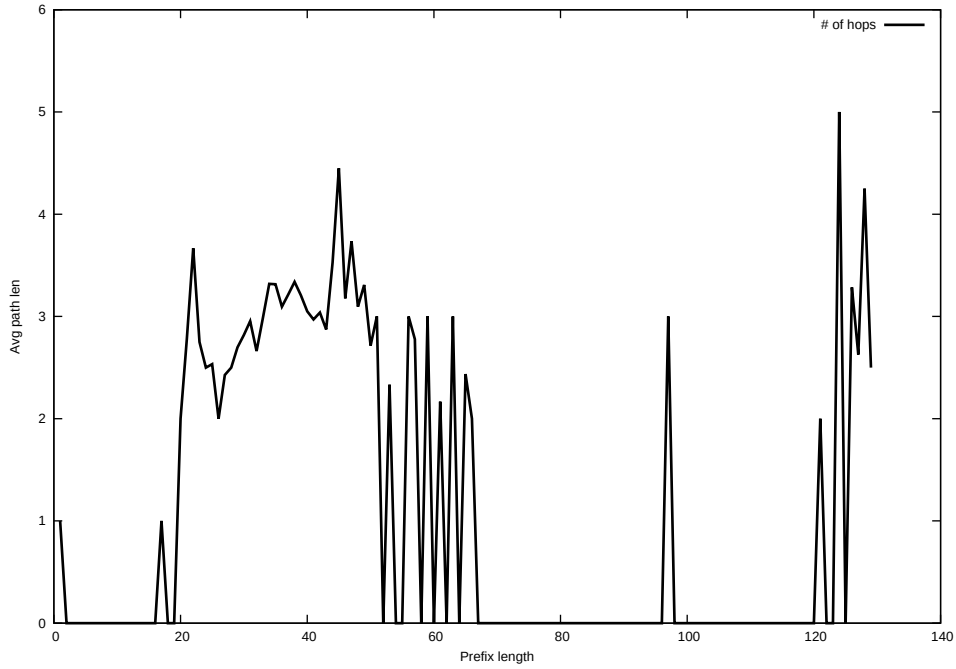
2015-04-13



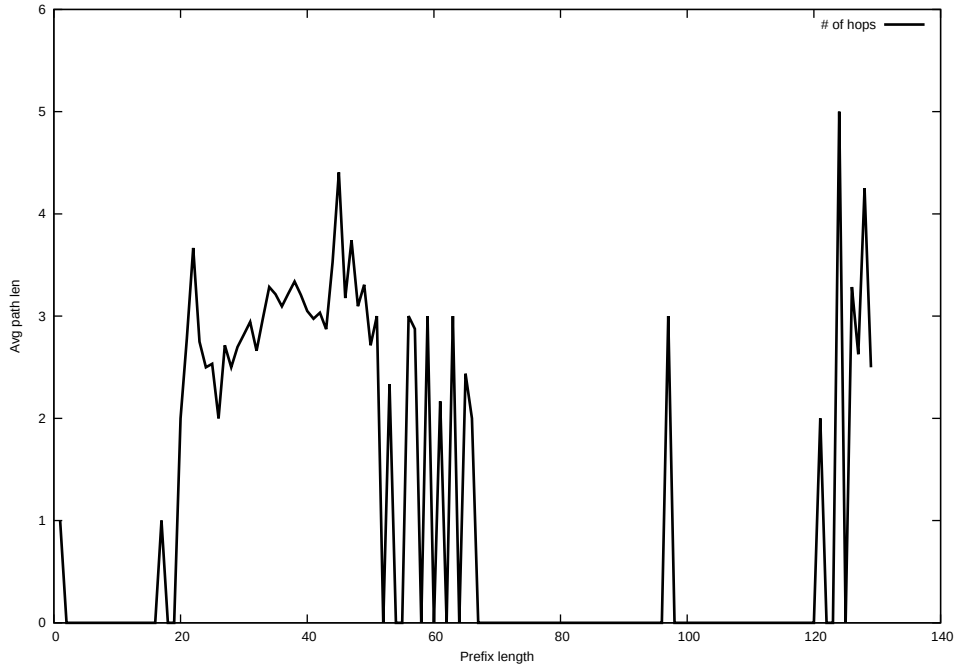
2015-04-14



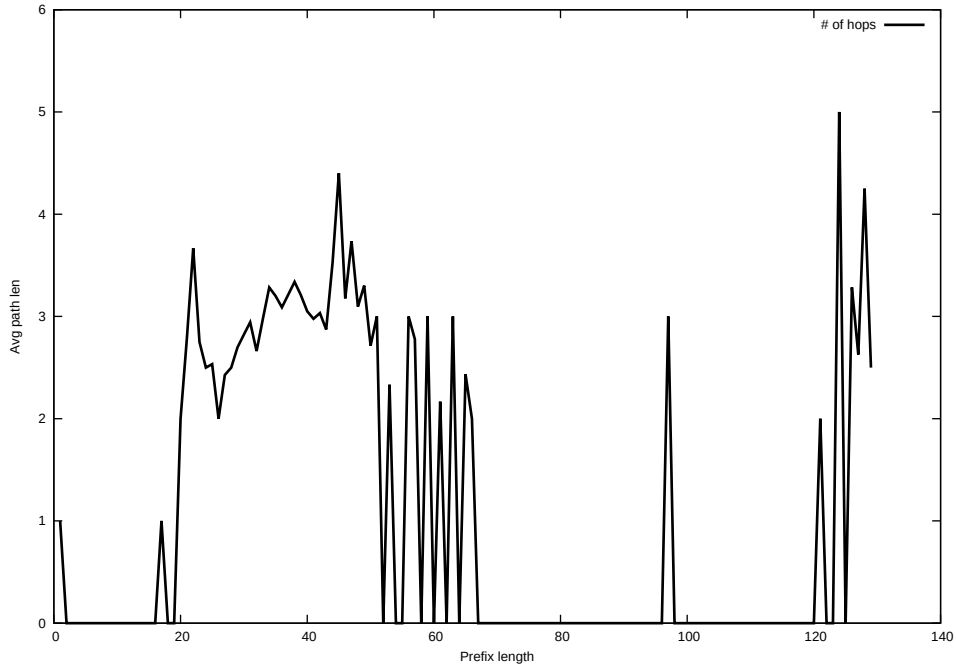
2015-04-15



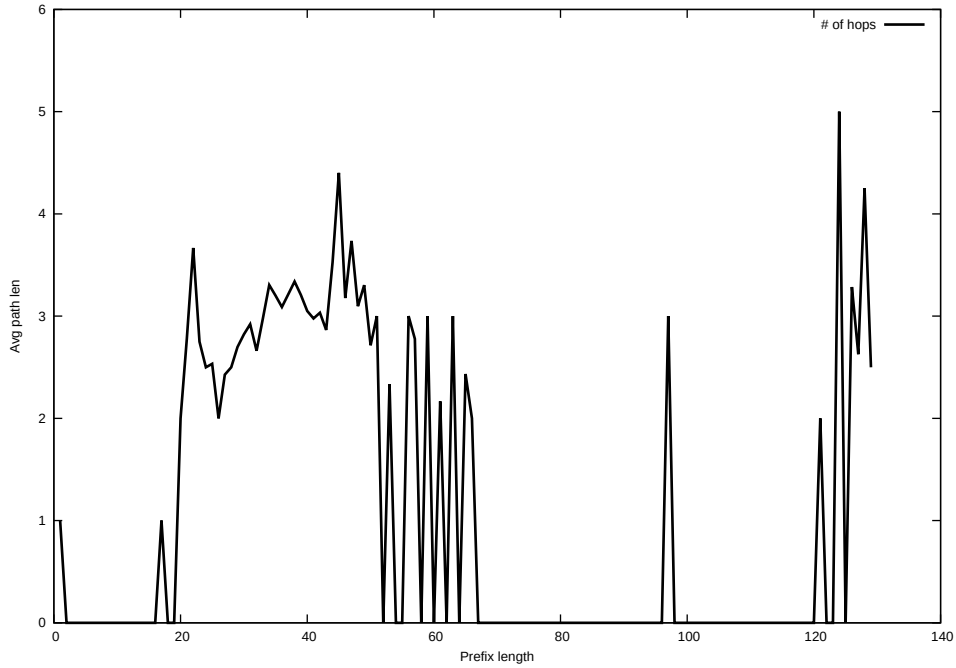
2015-04-16



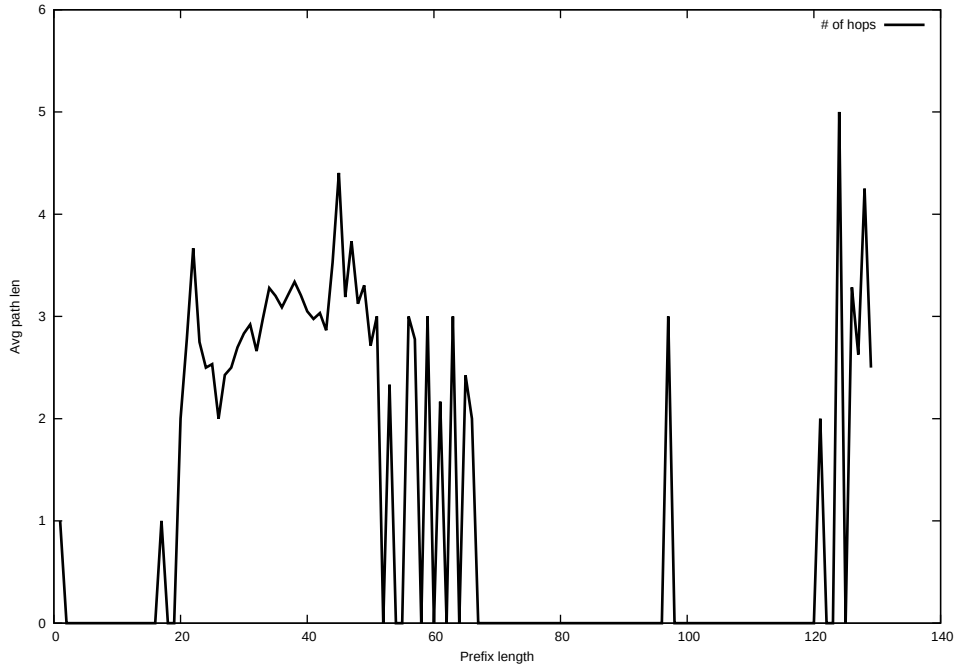
2015-04-17



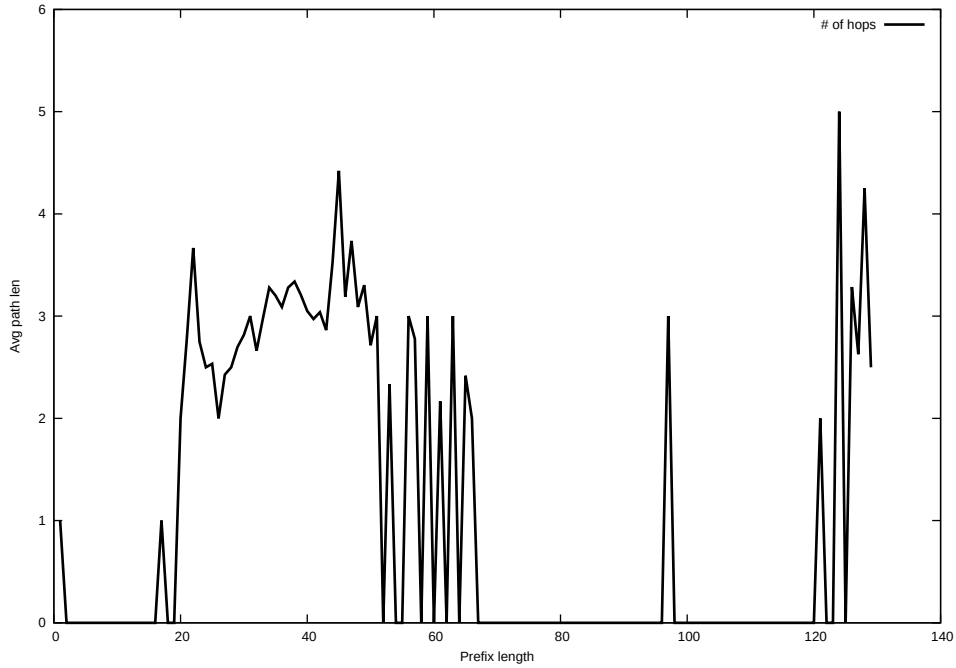
2015-04-18



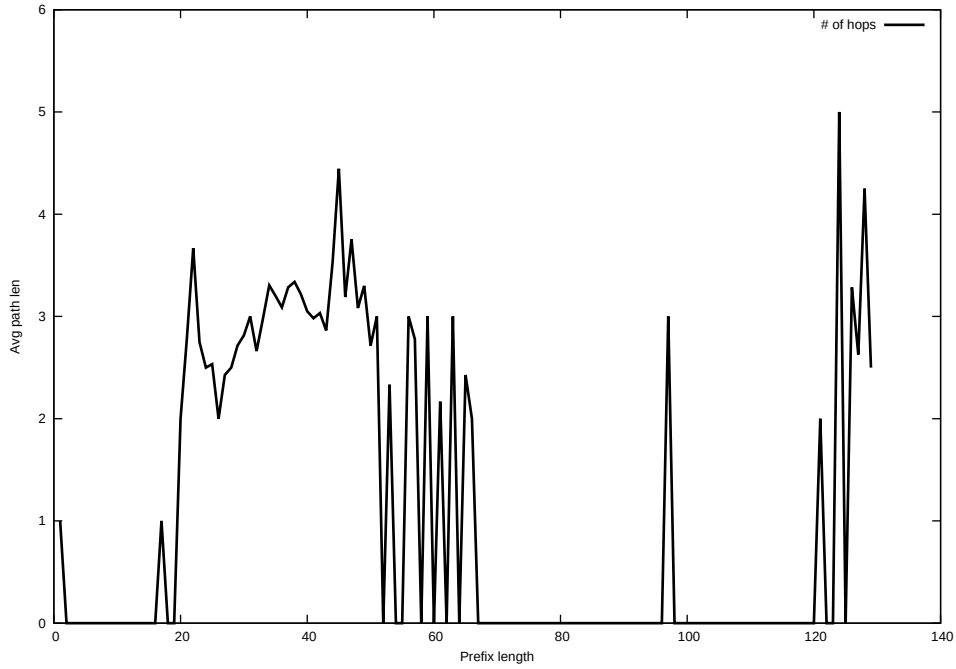
2015-04-19



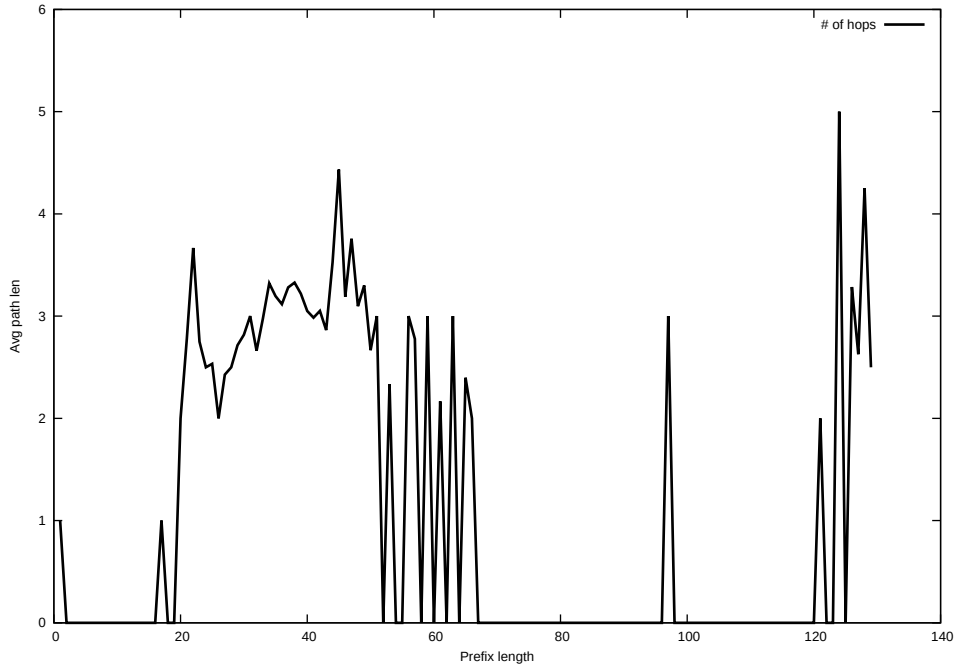
2015-04-20



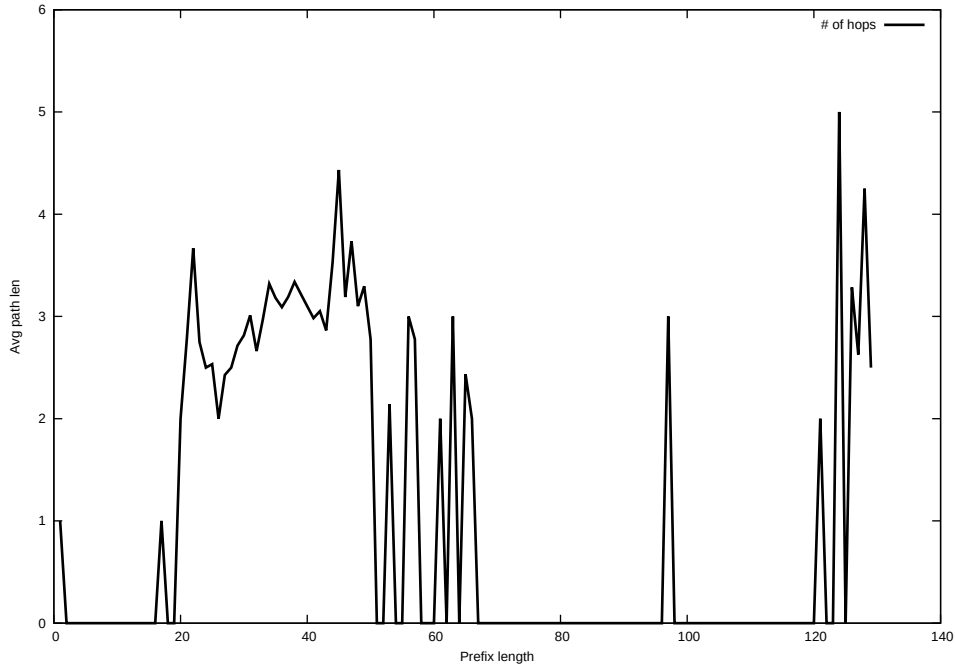
2015-04-21



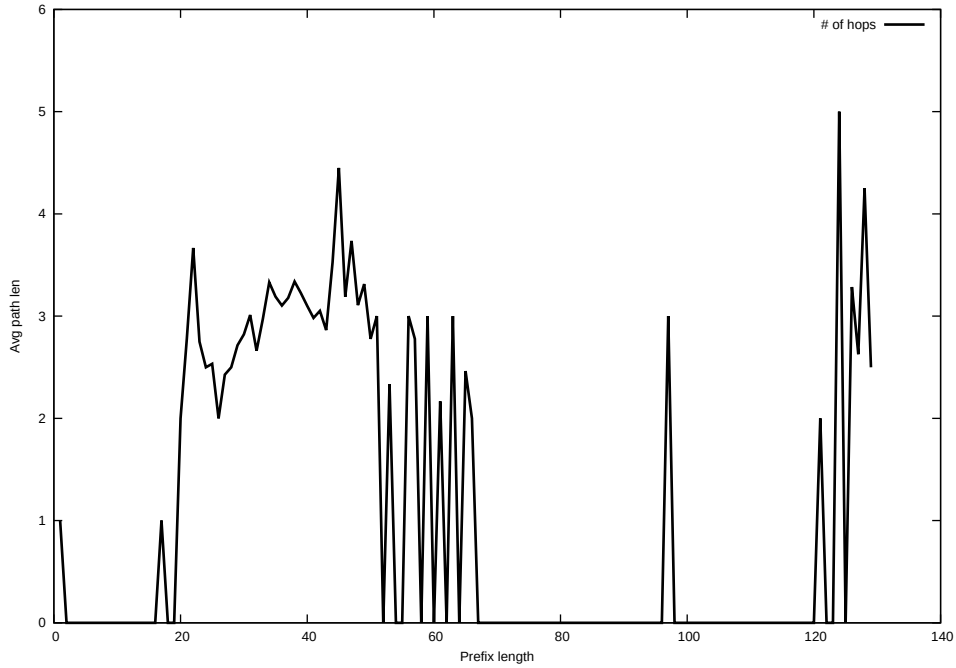
2015-04-22



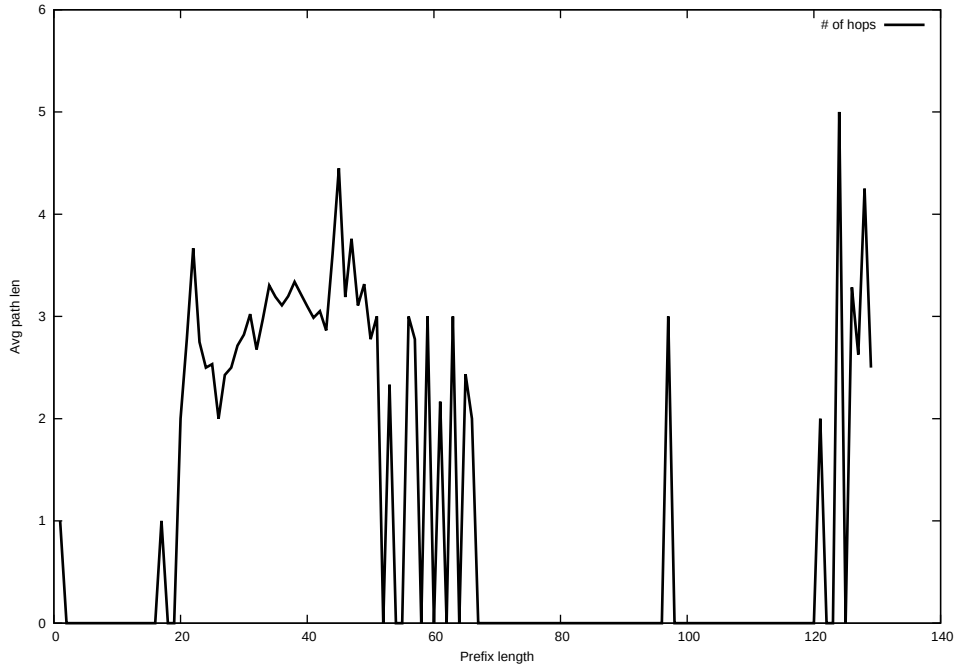
2015-04-23



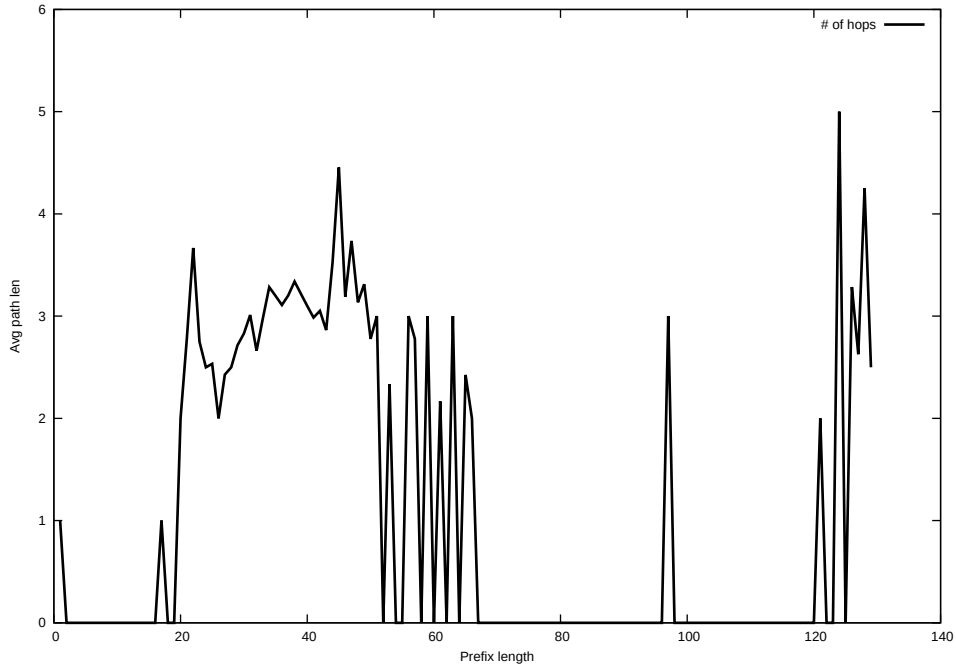
2015-04-24



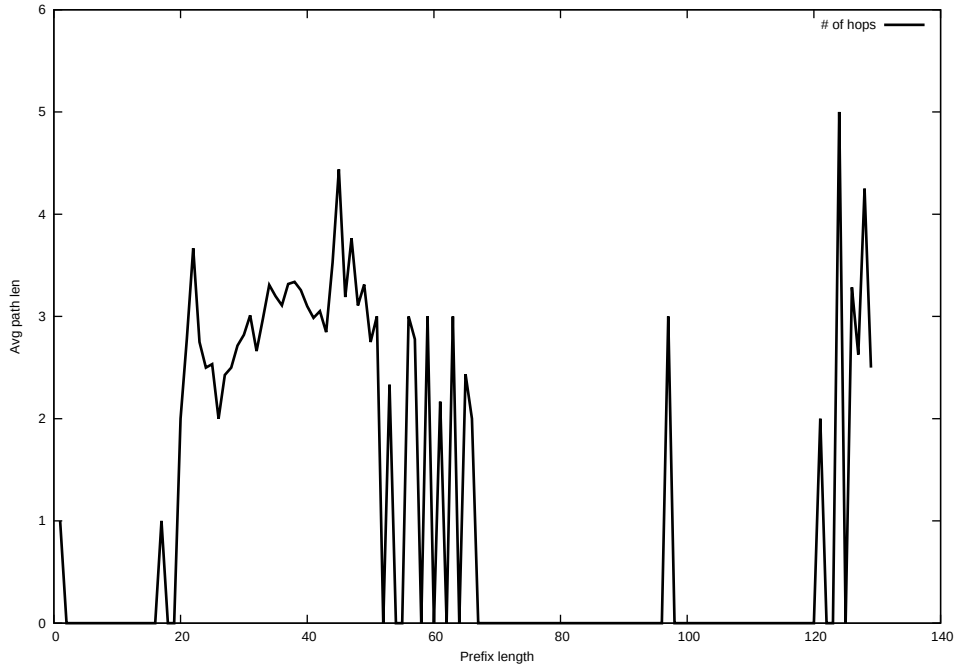
2015-04-25



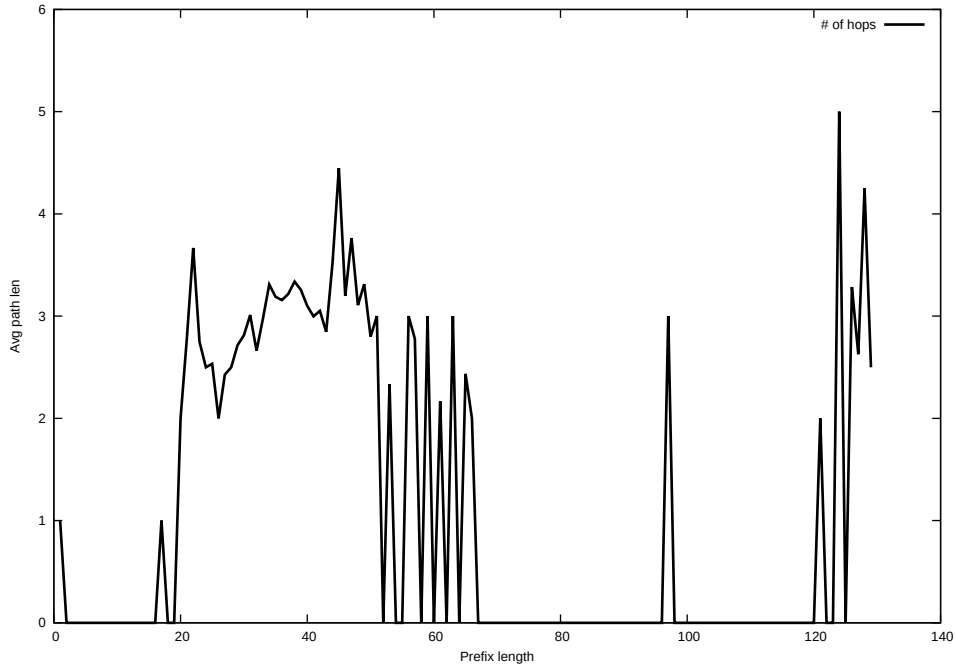
2015-04-26



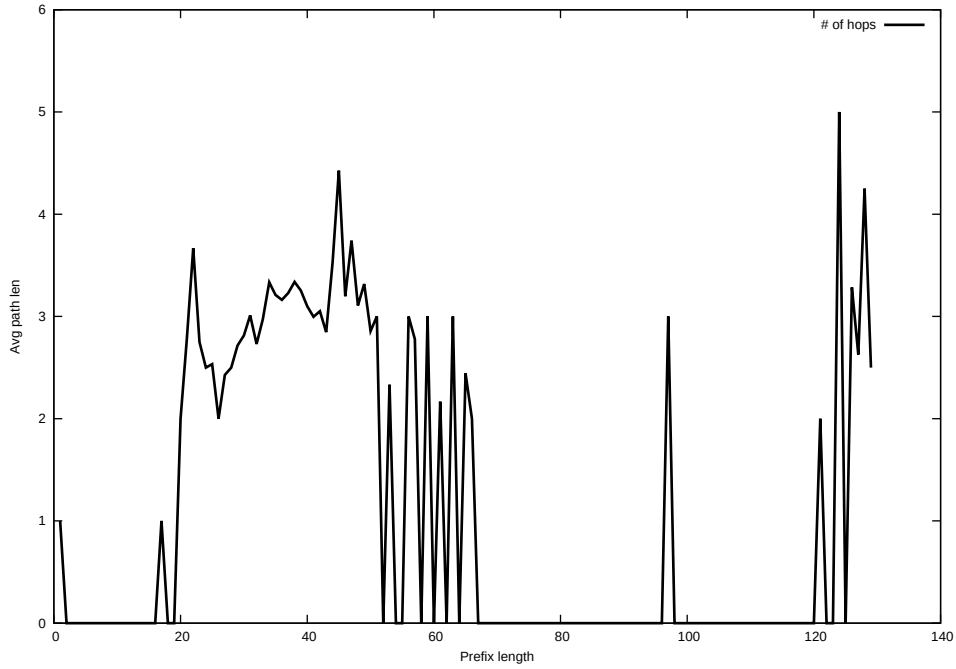
2015-04-27



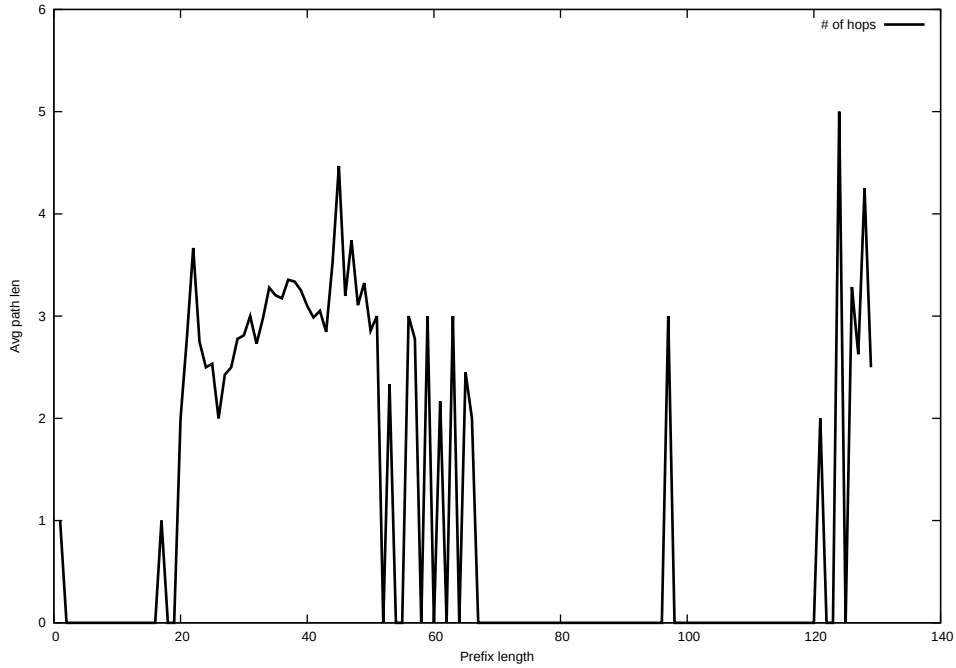
2015-04-28



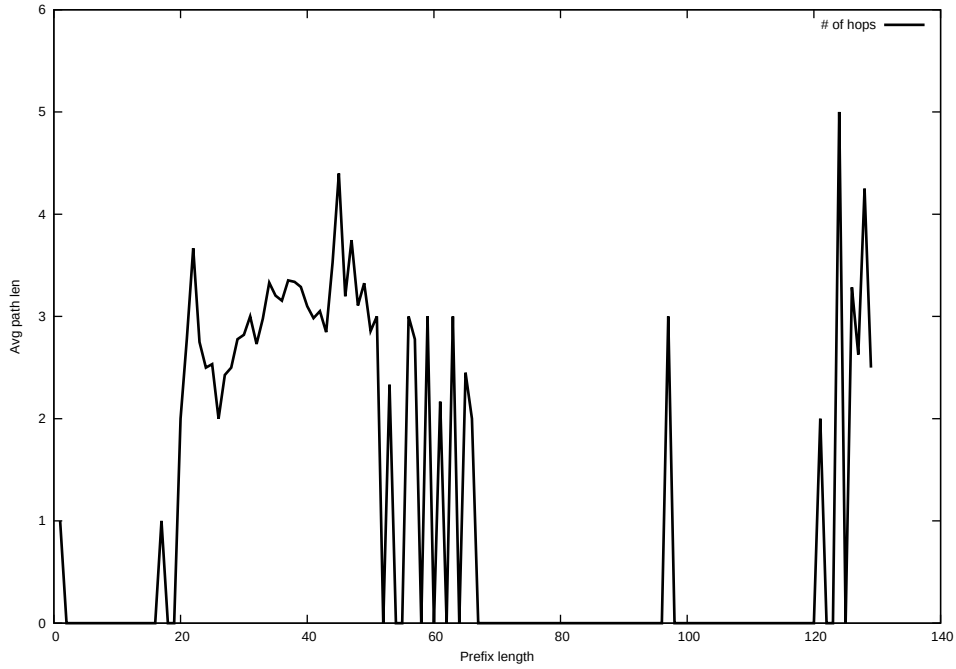
2015-04-29



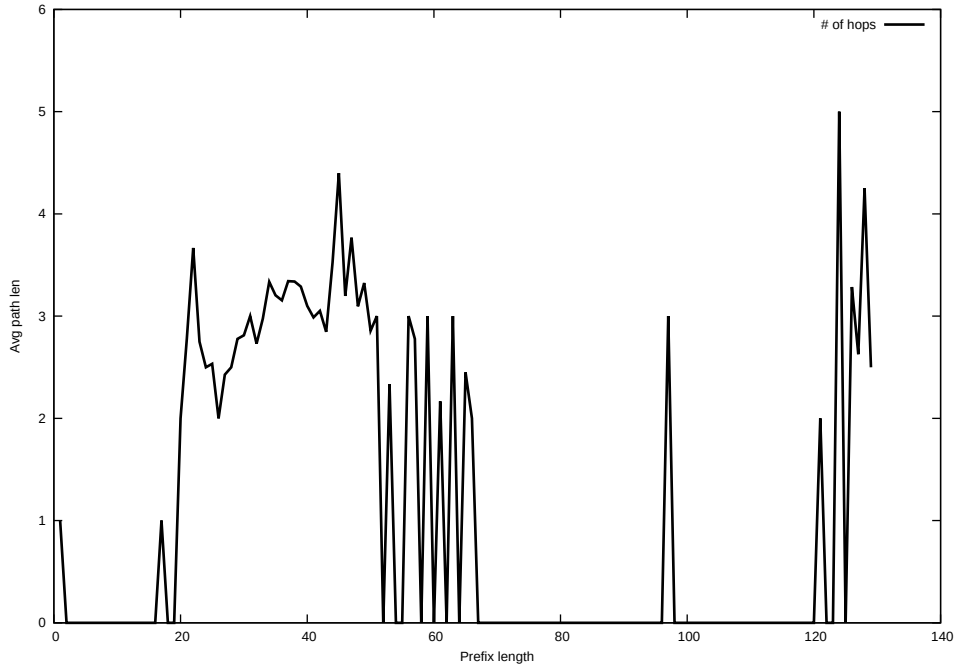
2015-04-30



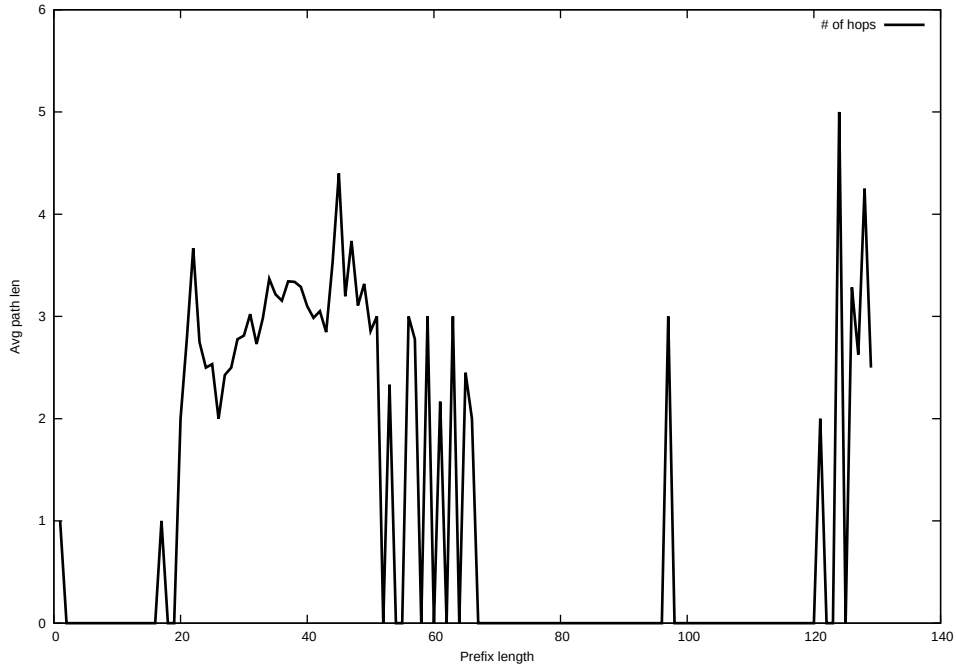
2015-05-01



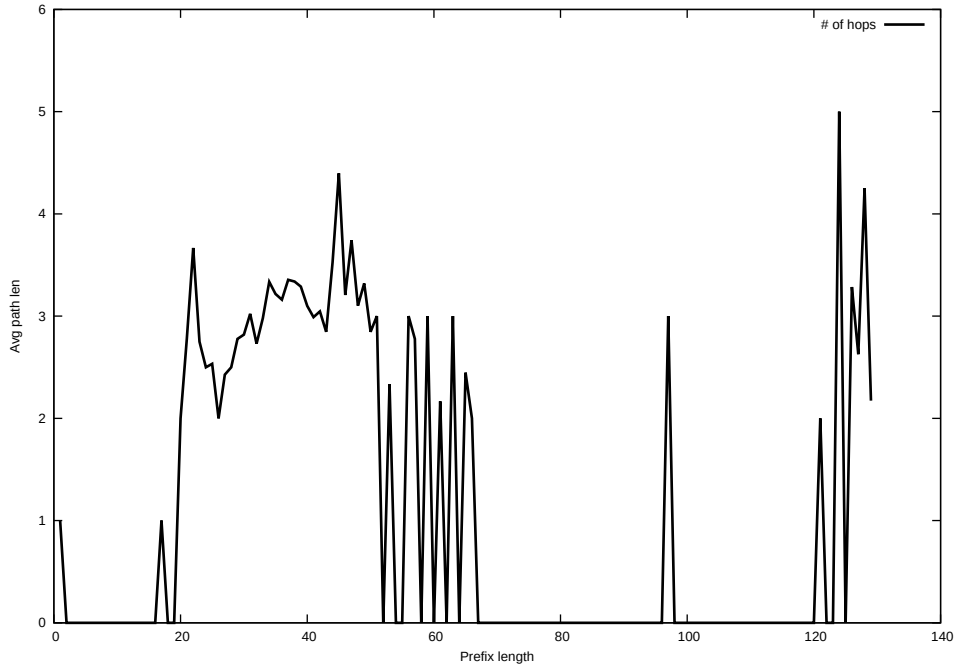
2015-05-02



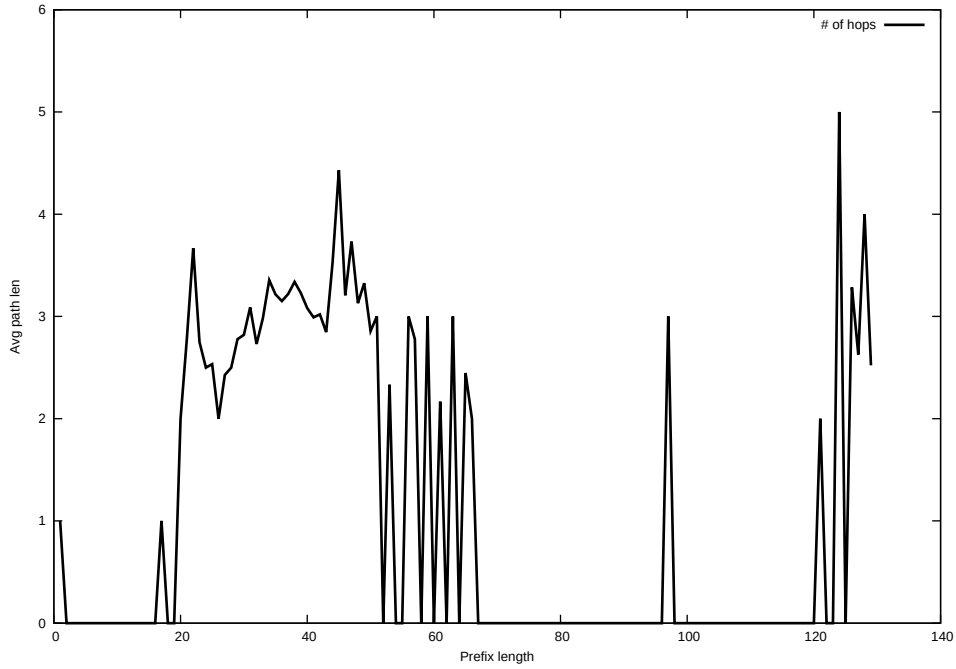
2015-05-03



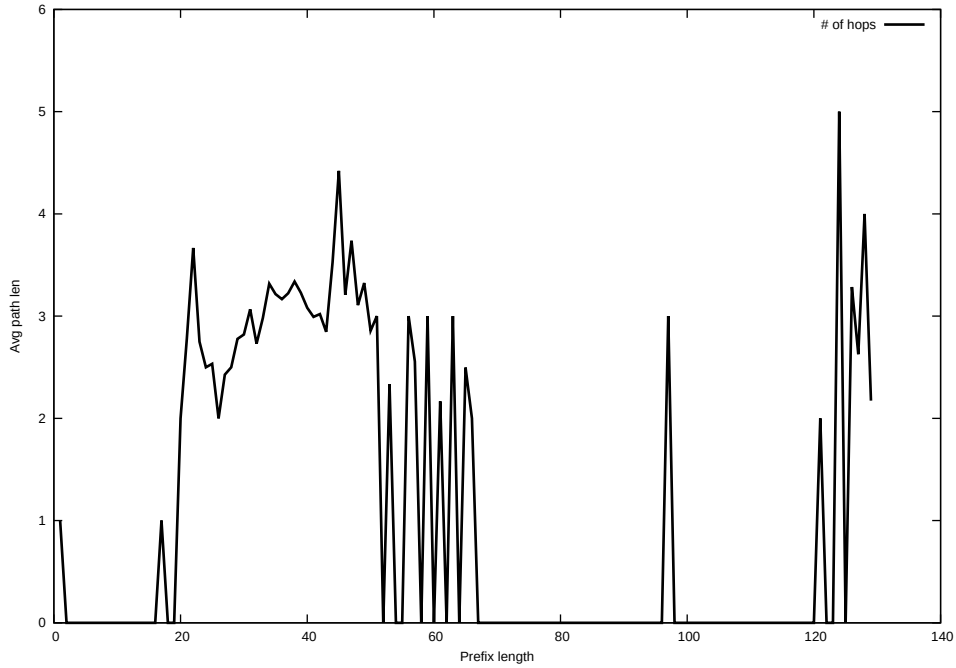
2015-05-04



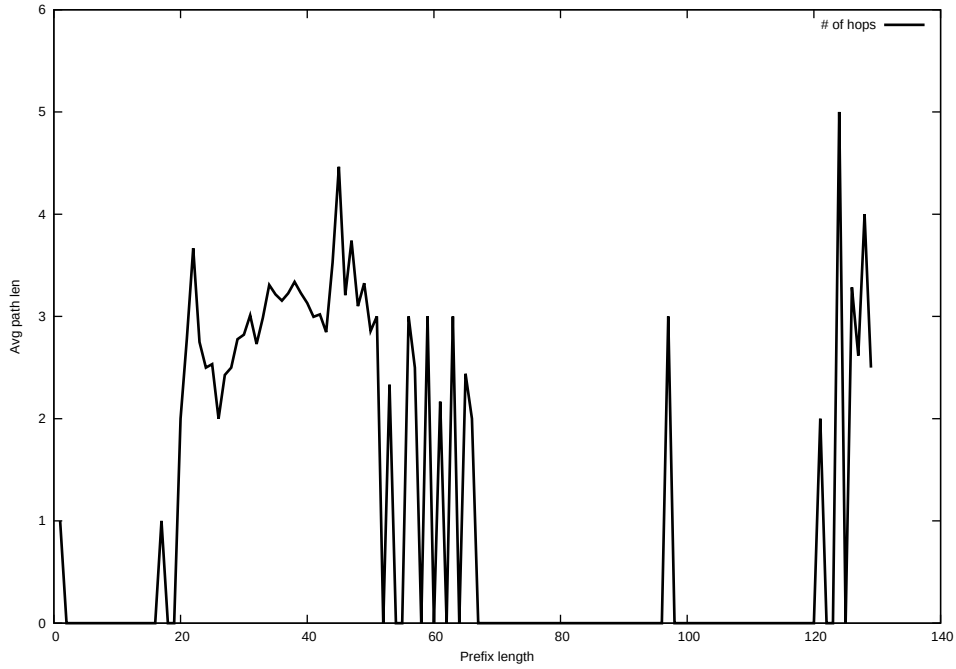
2015-05-05



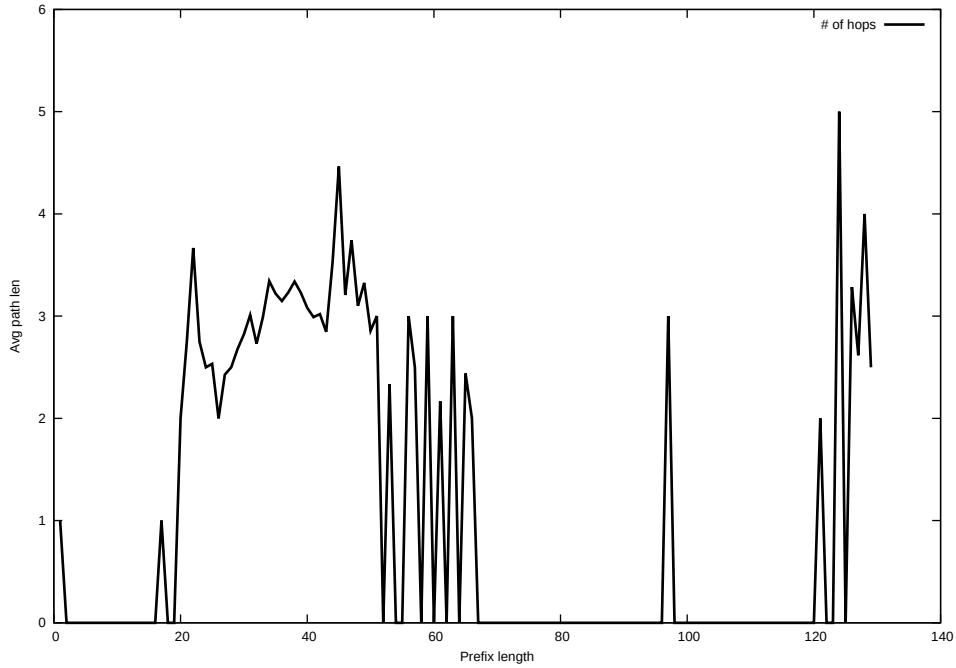
2015-05-06



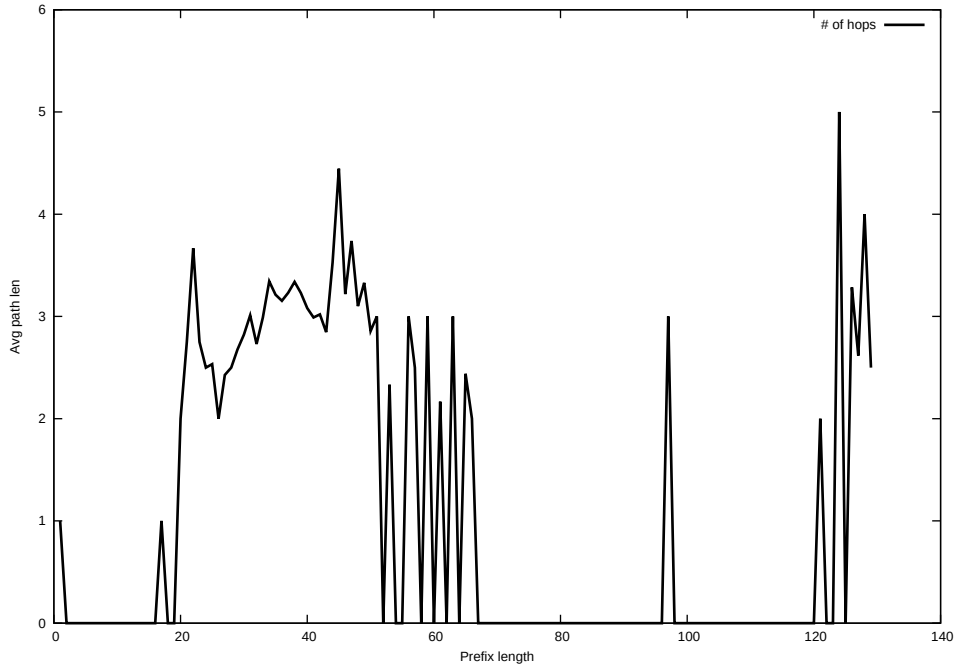
2015-05-07



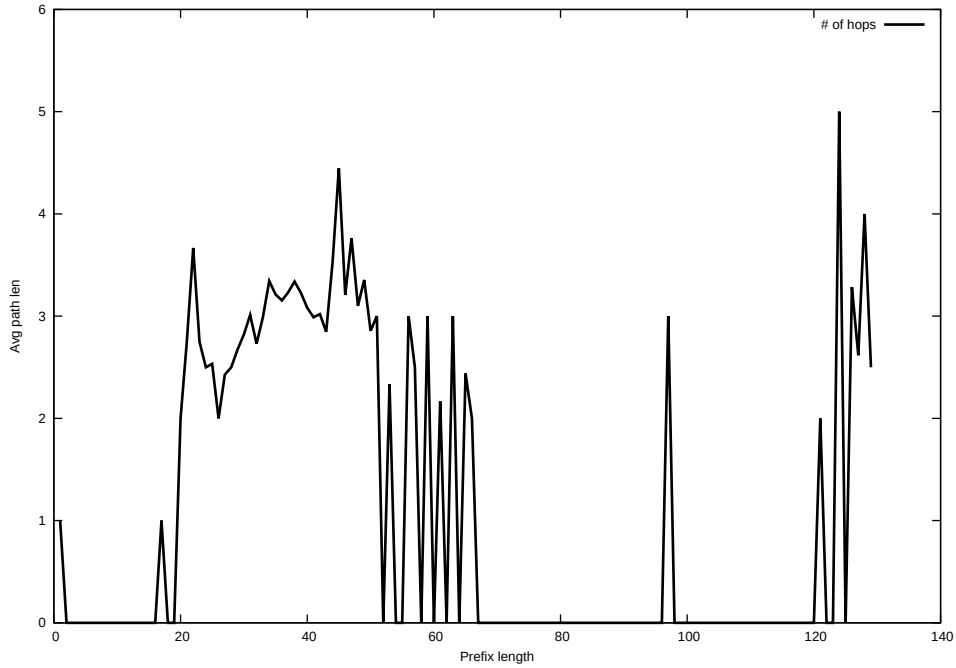
2015-05-08



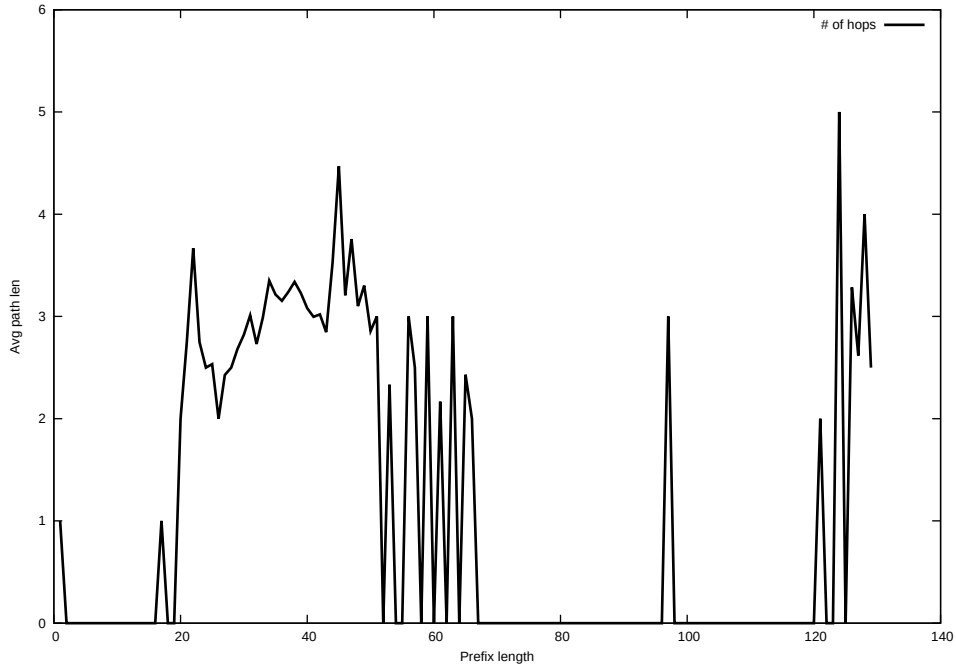
2015-05-09



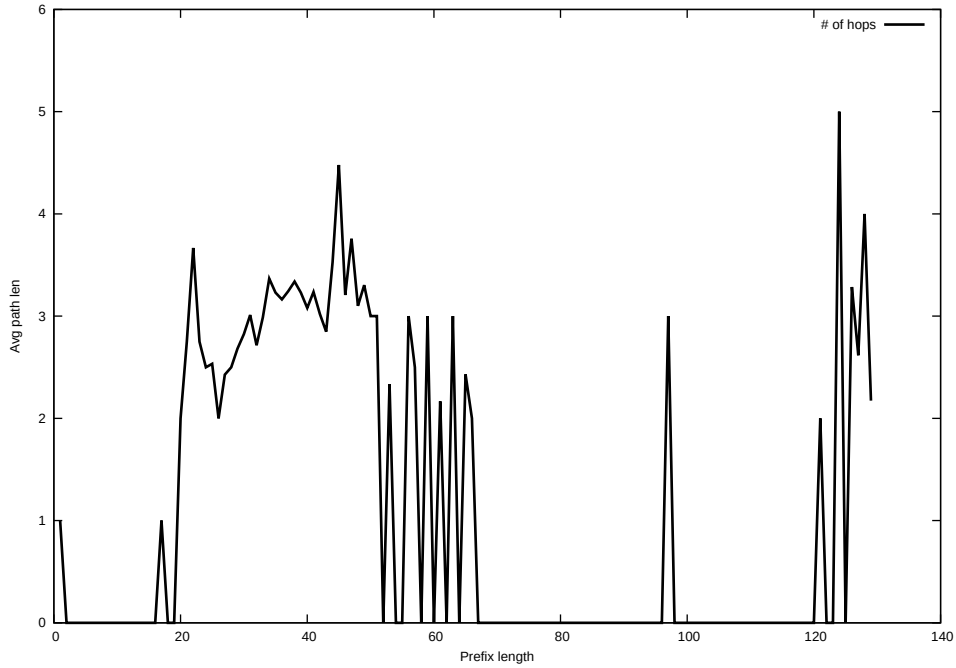
2015-05-10



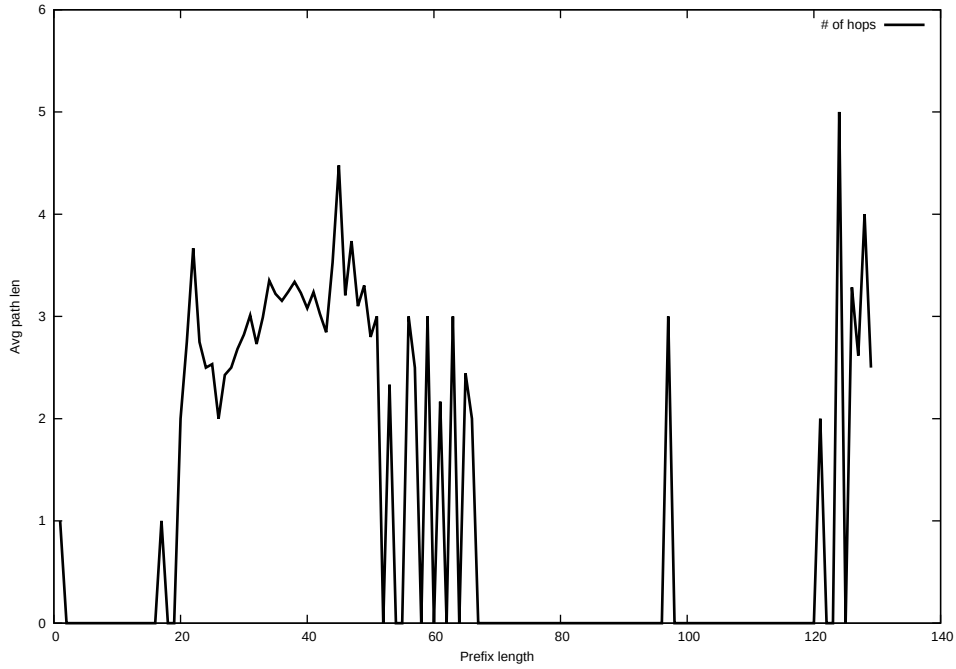
2015-05-11



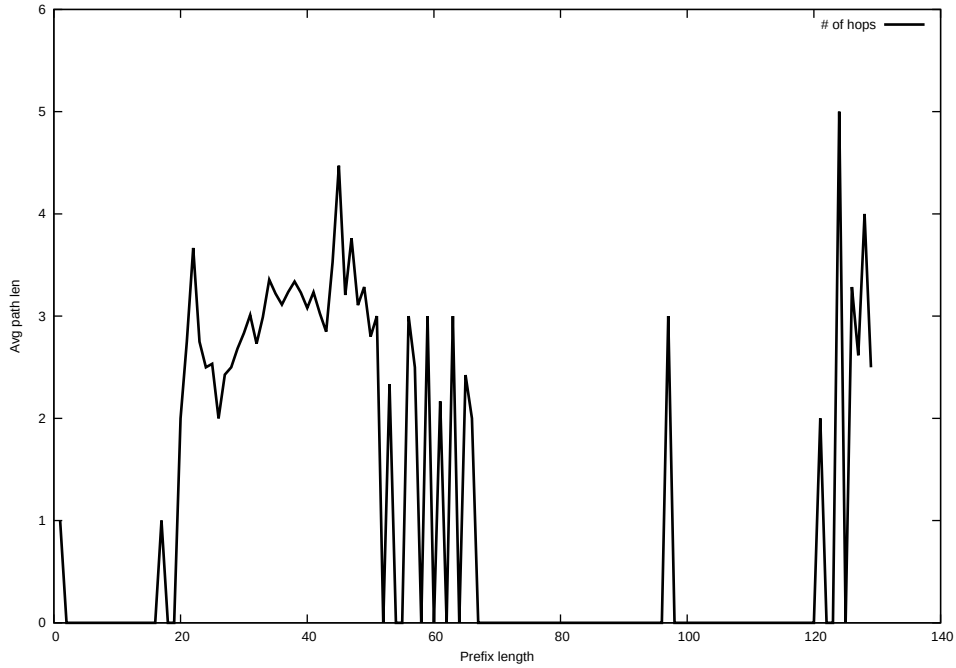
2015-05-12



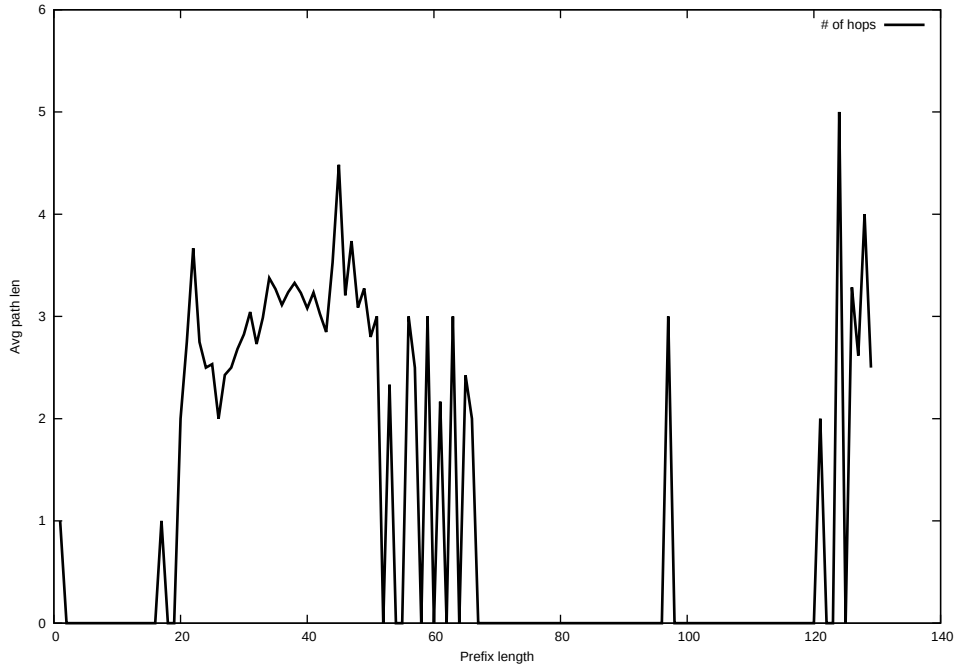
2015-05-13



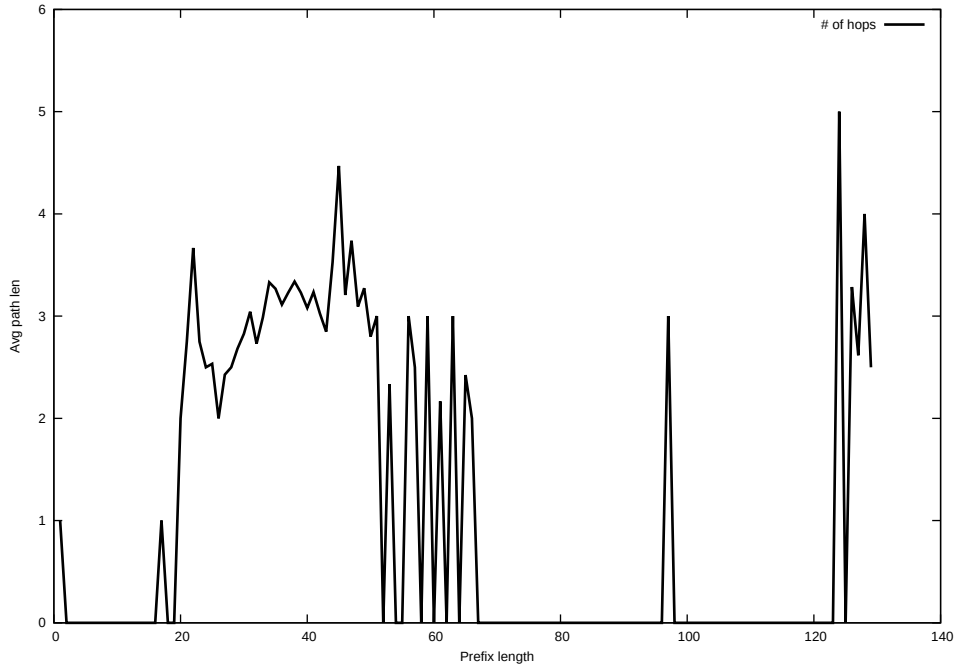
2015-05-14



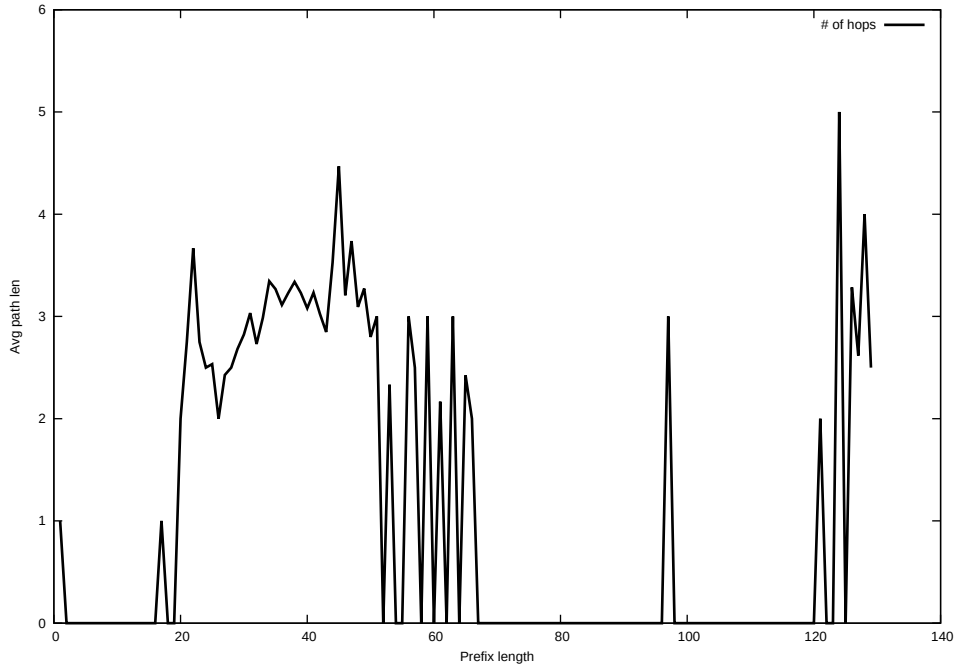
2015-05-15



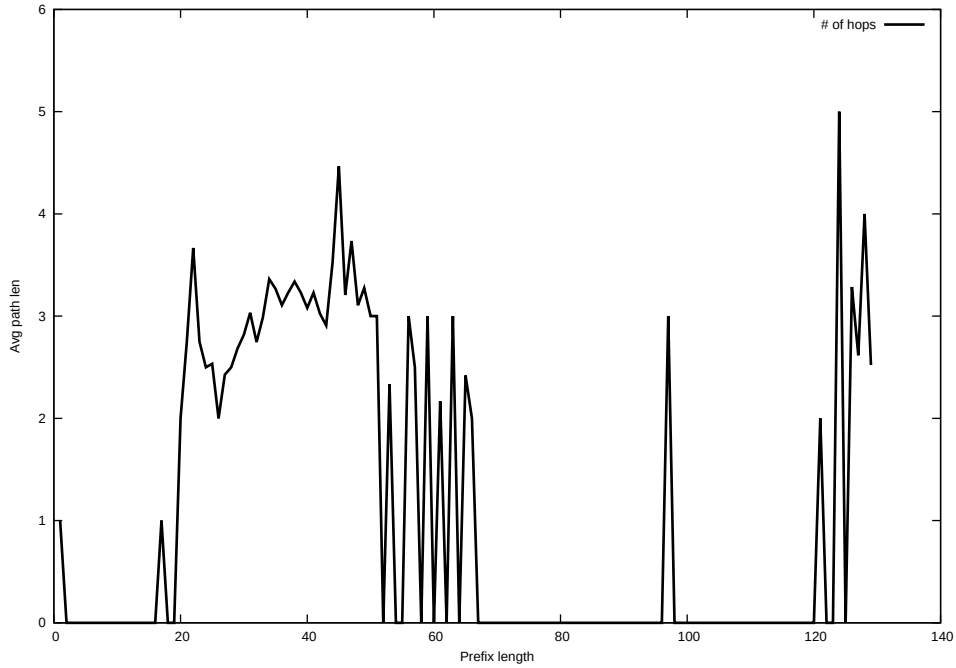
2015-05-16



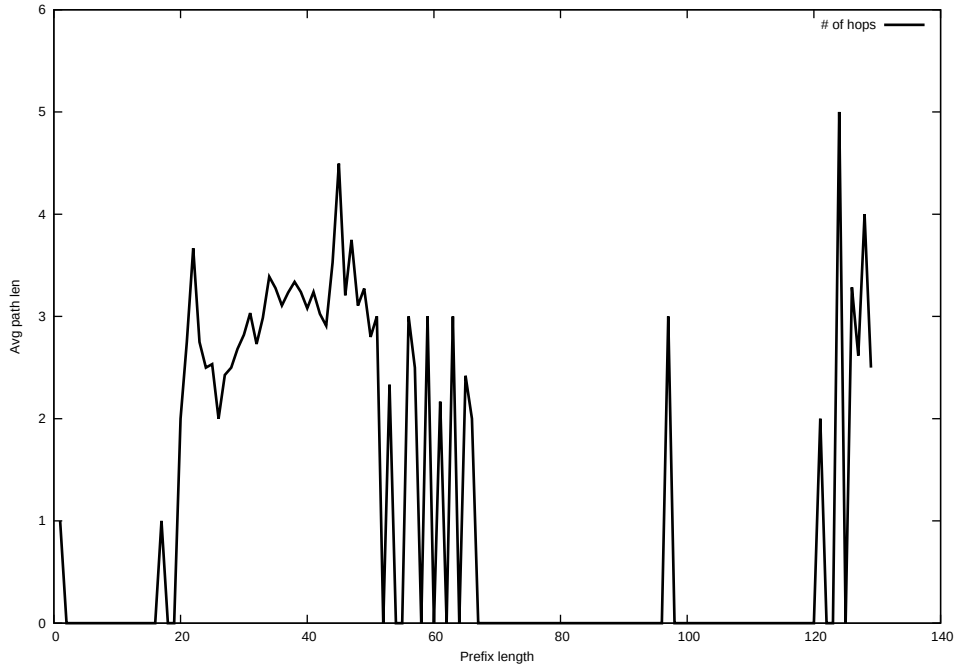
2015-05-17



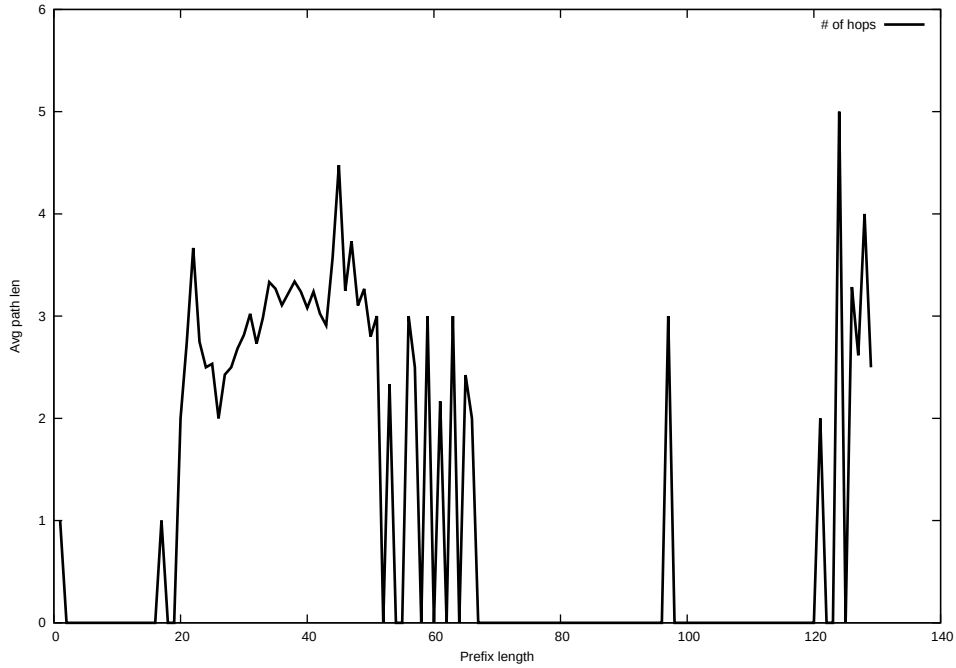
2015-05-18



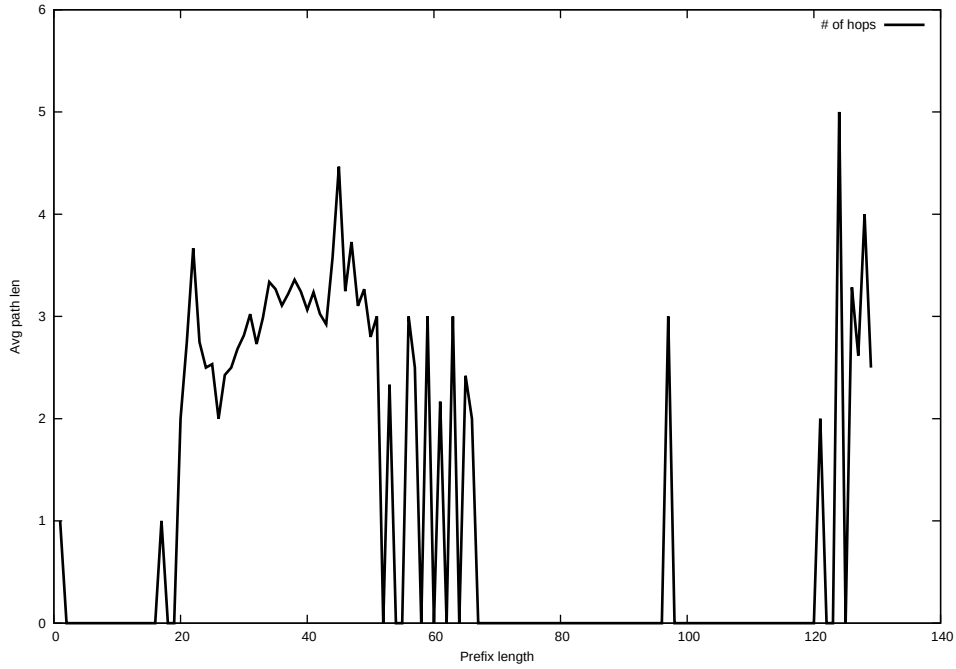
2015-05-19



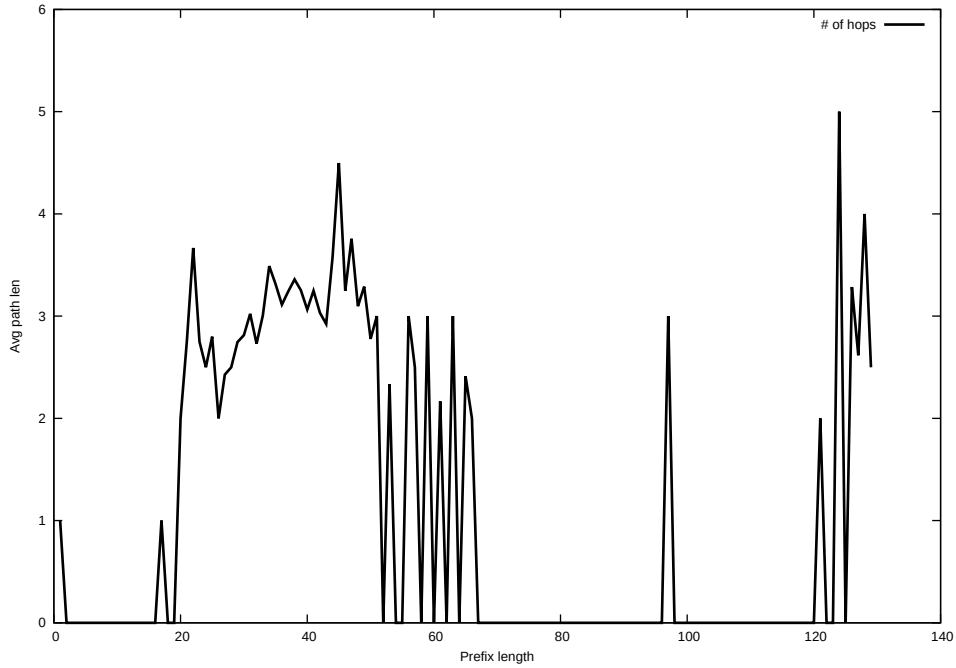
2015-05-20



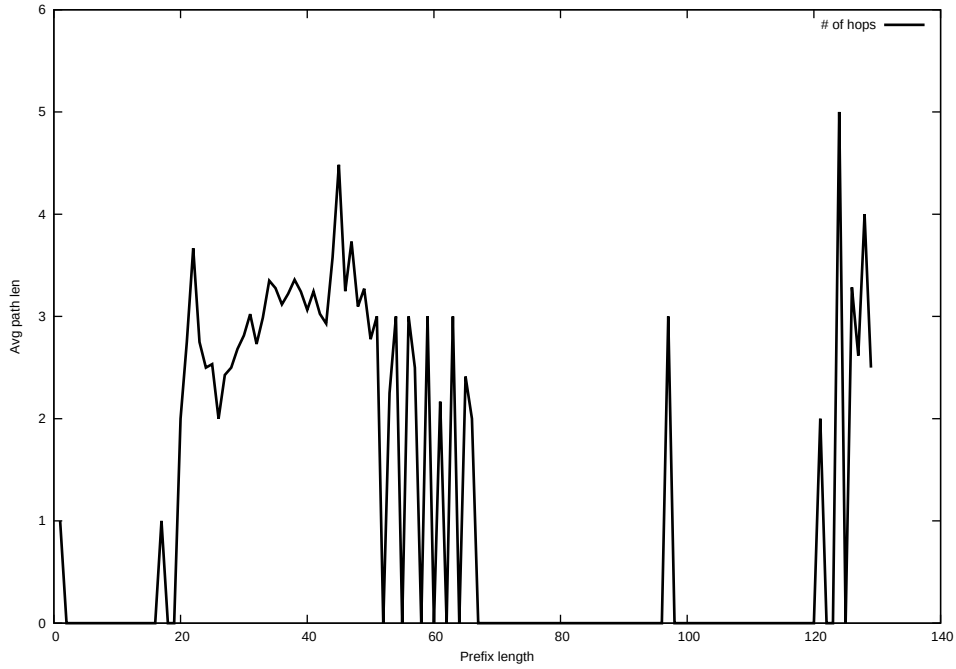
2015-05-21



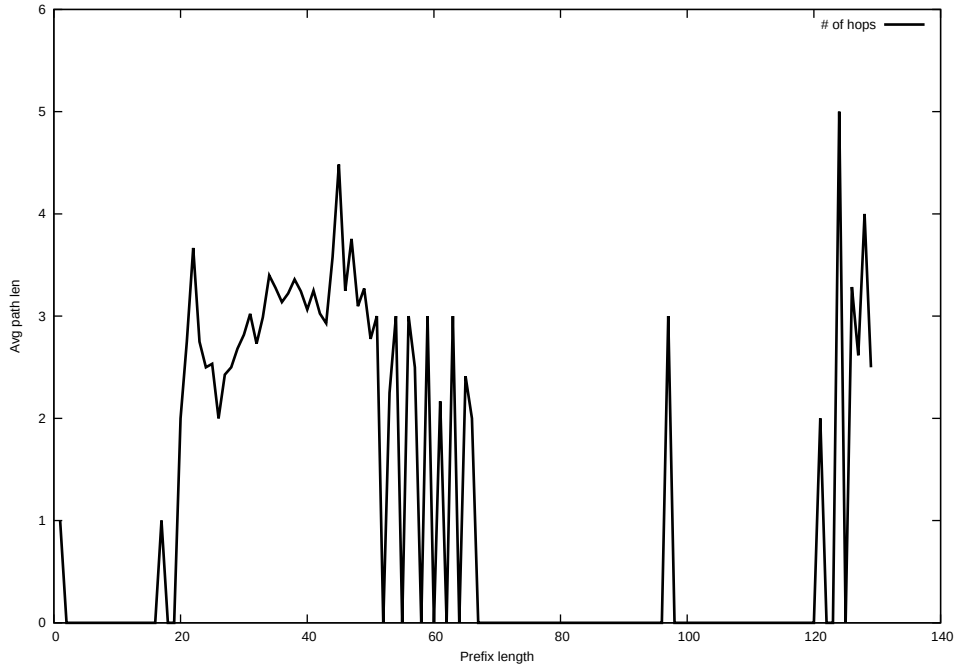
2015-05-22



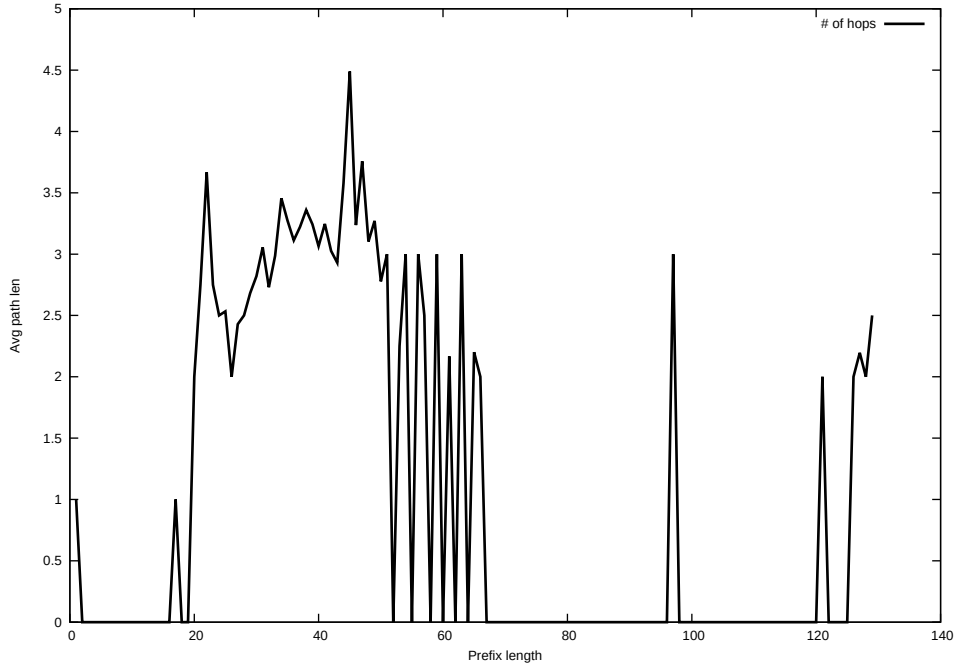
2015-05-23



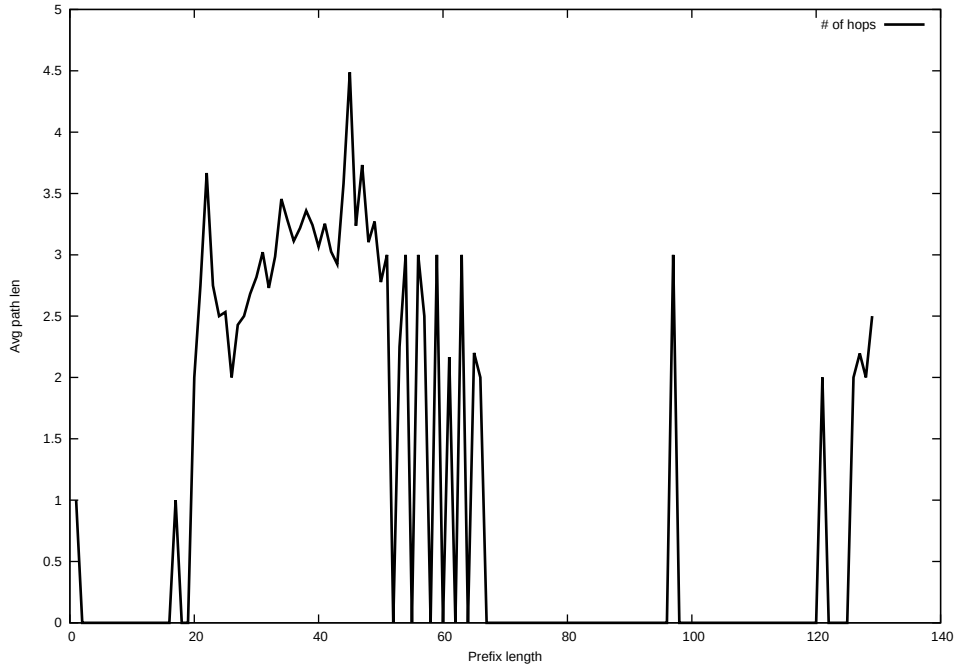
2015-05-24



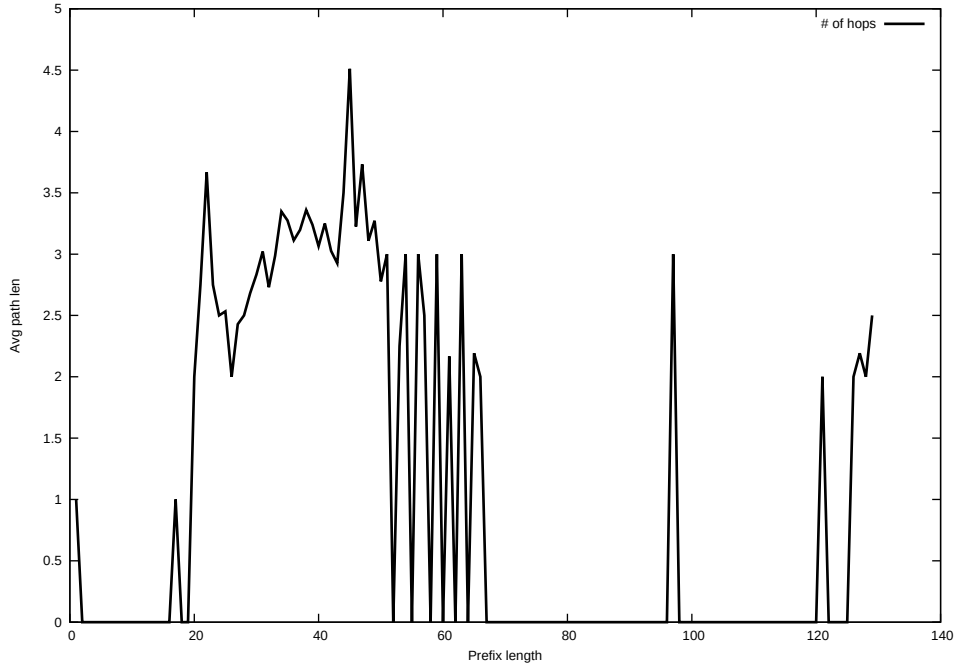
2015-05-25



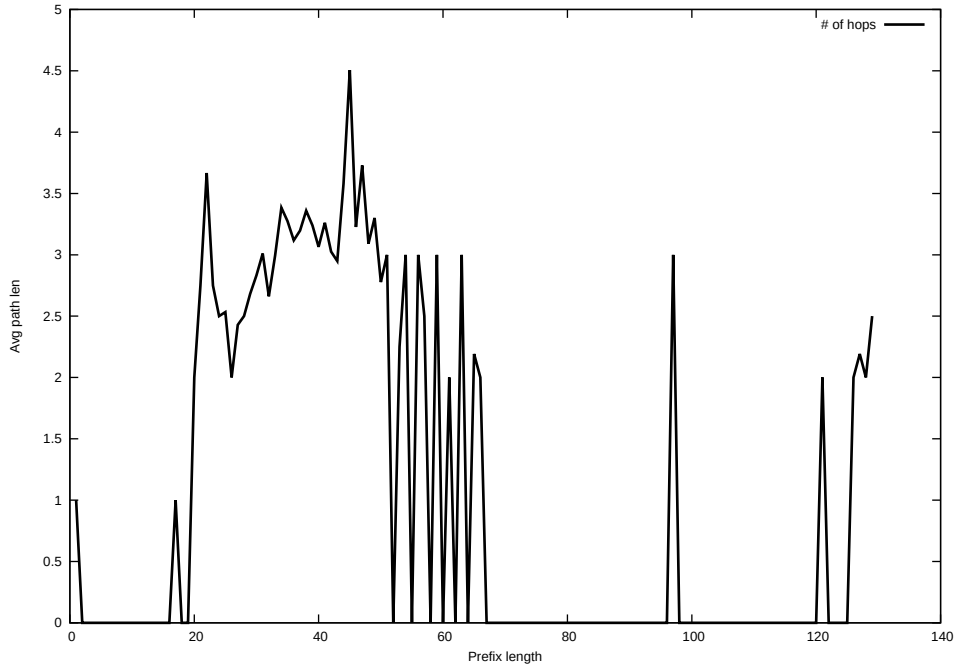
2015-05-26



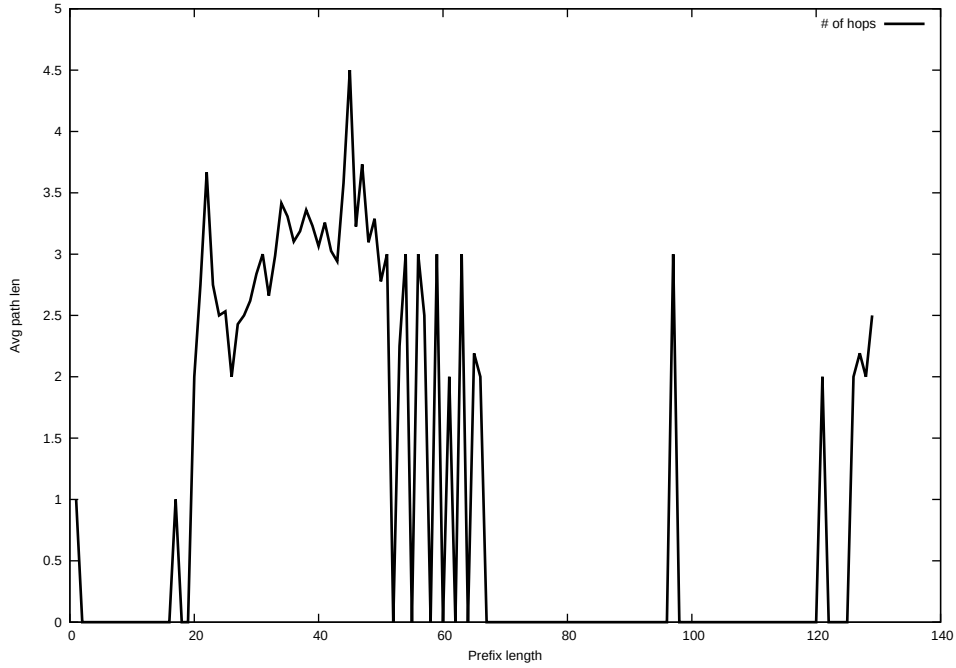
2015-05-27



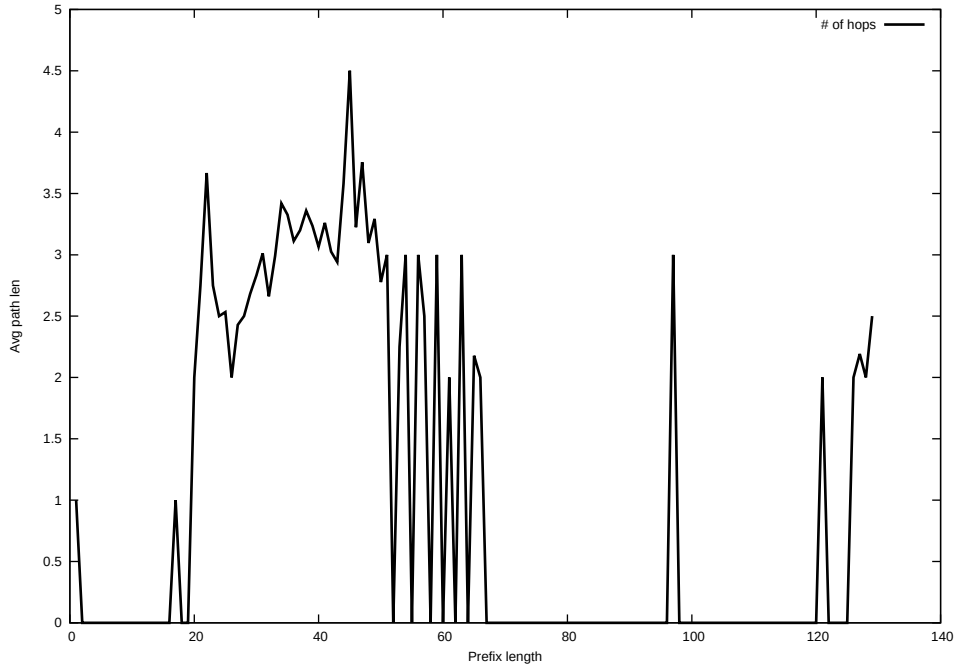
2015-05-28



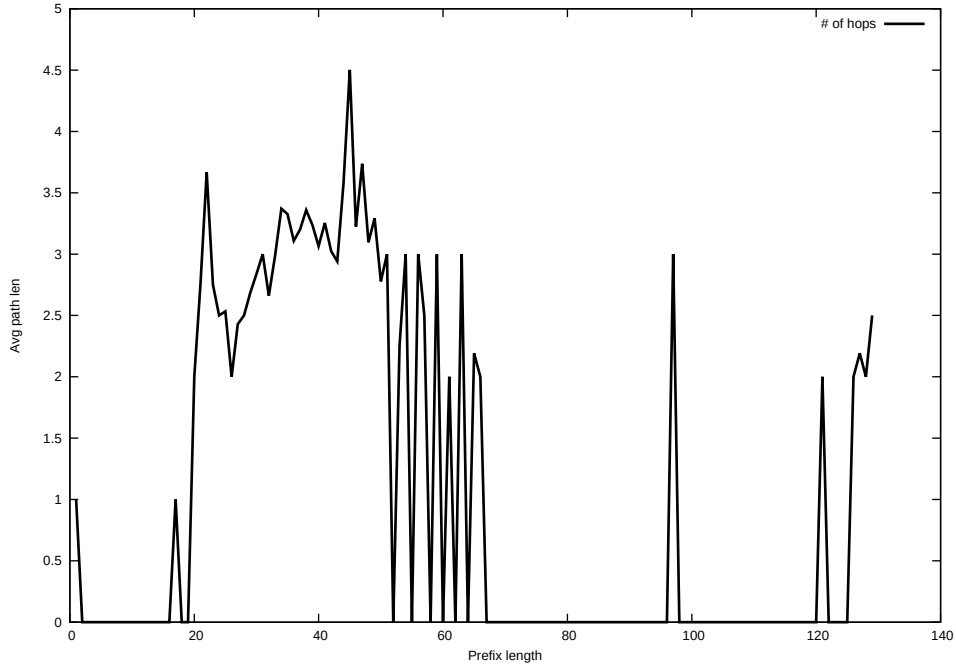
2015-05-29



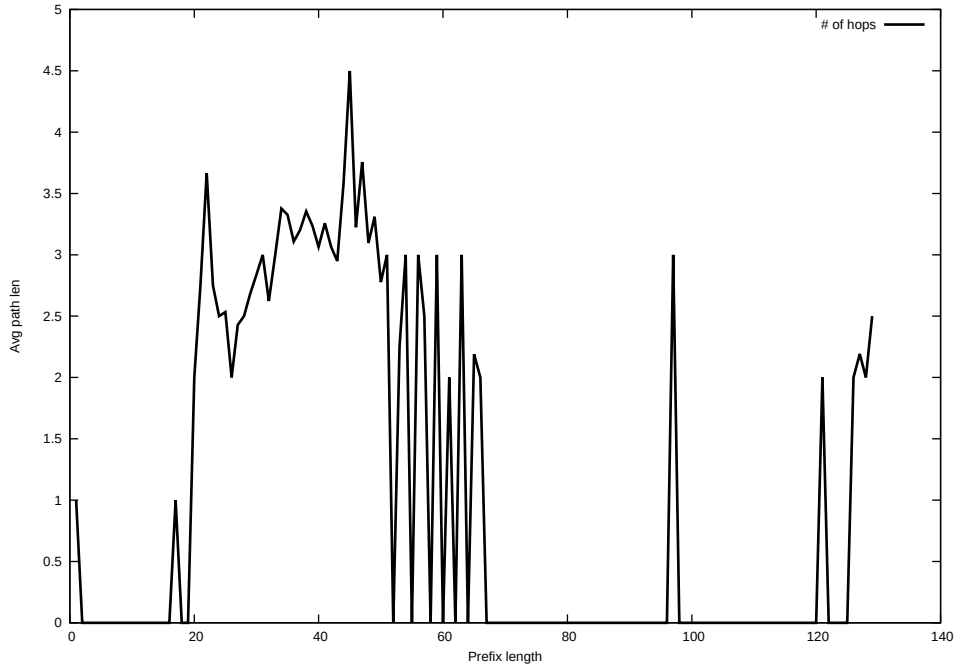
2015-05-30



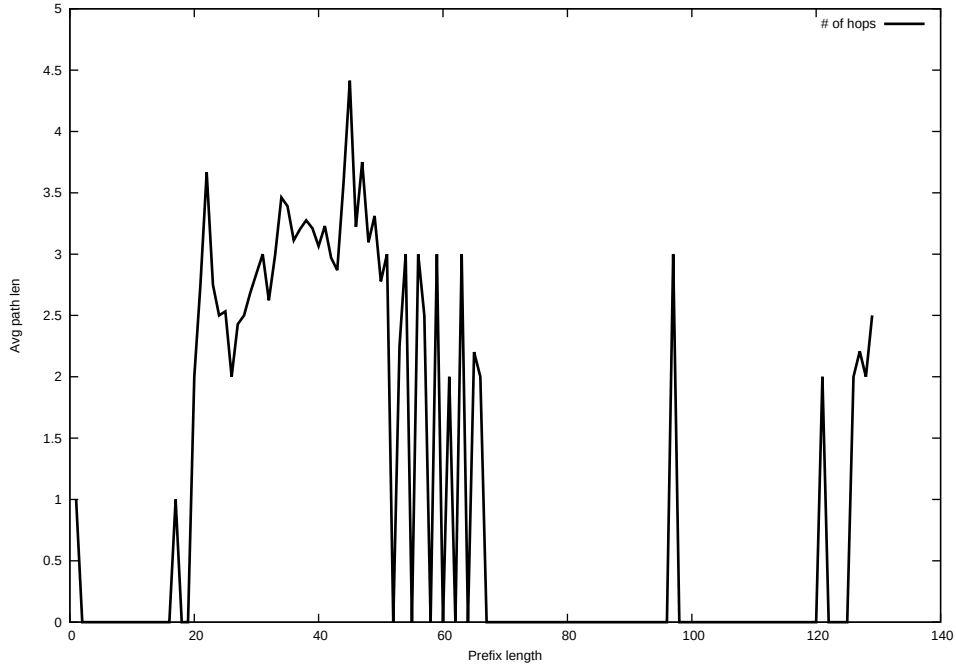
2015-05-31



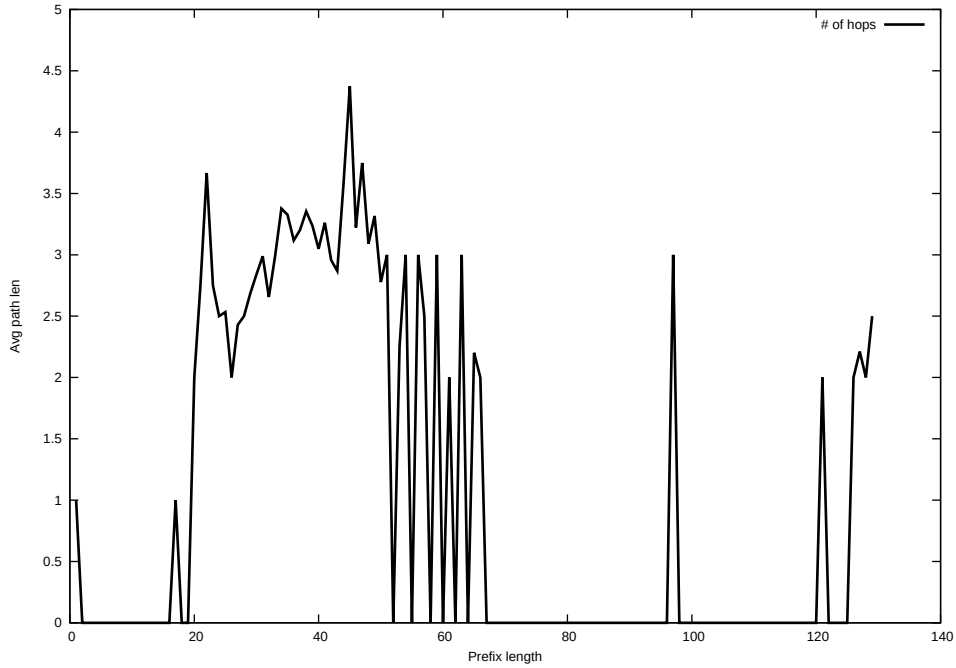
2015-06-01



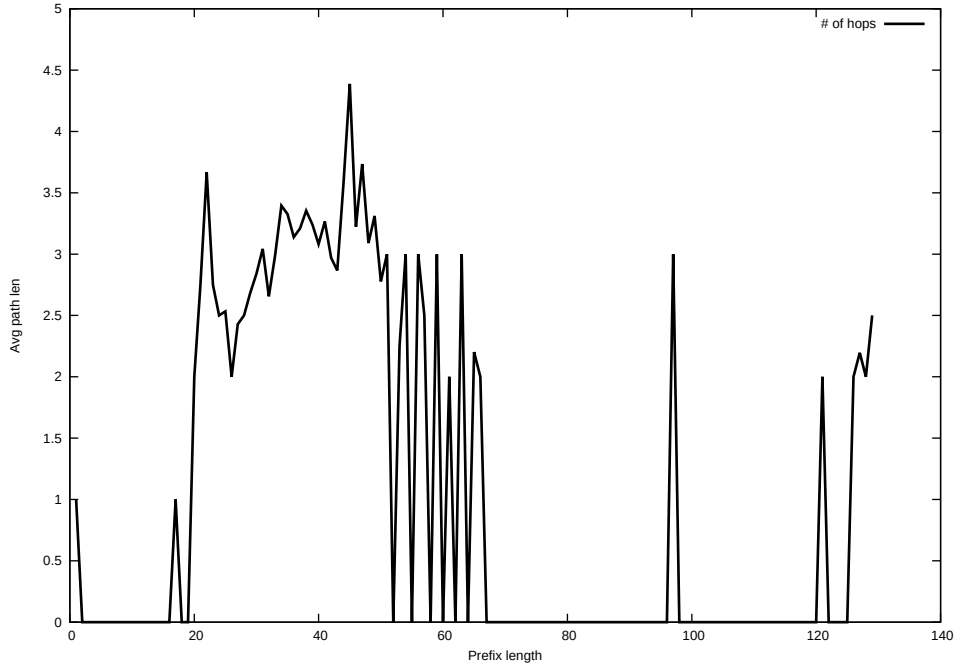
2015-06-02



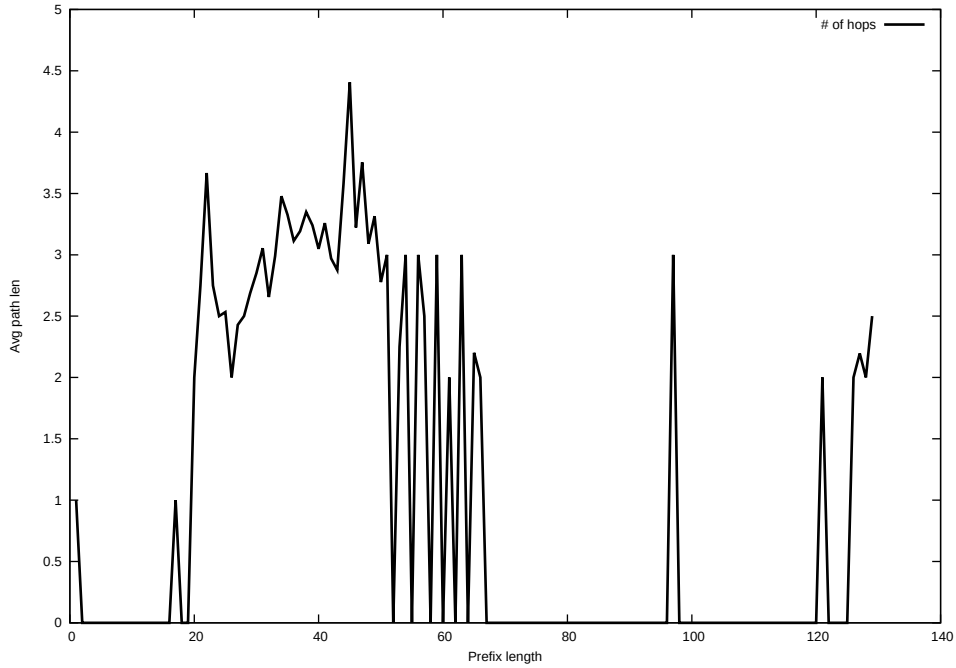
2015-06-03



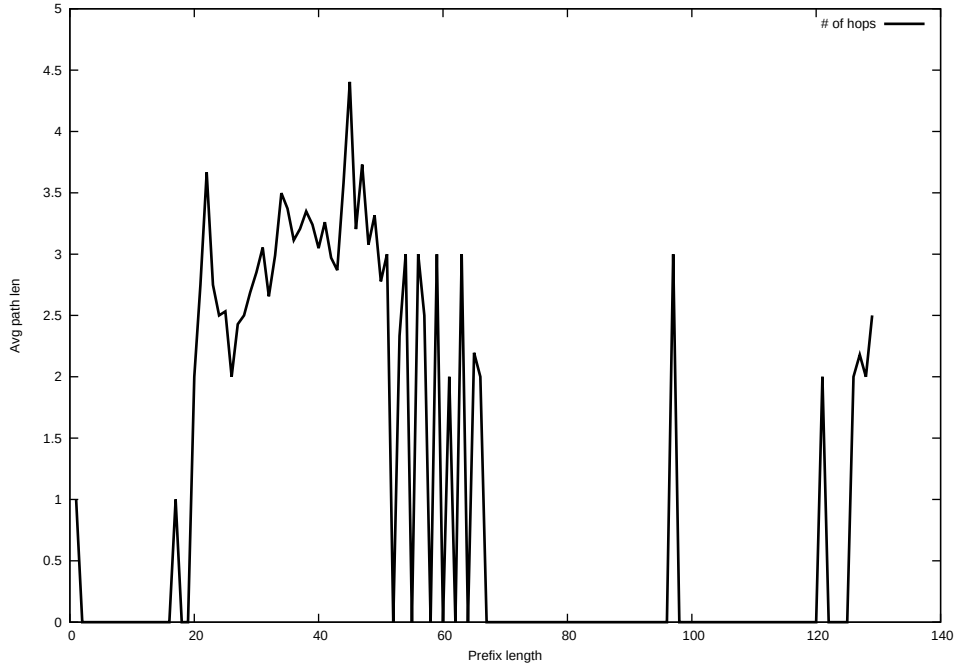
2015-06-04



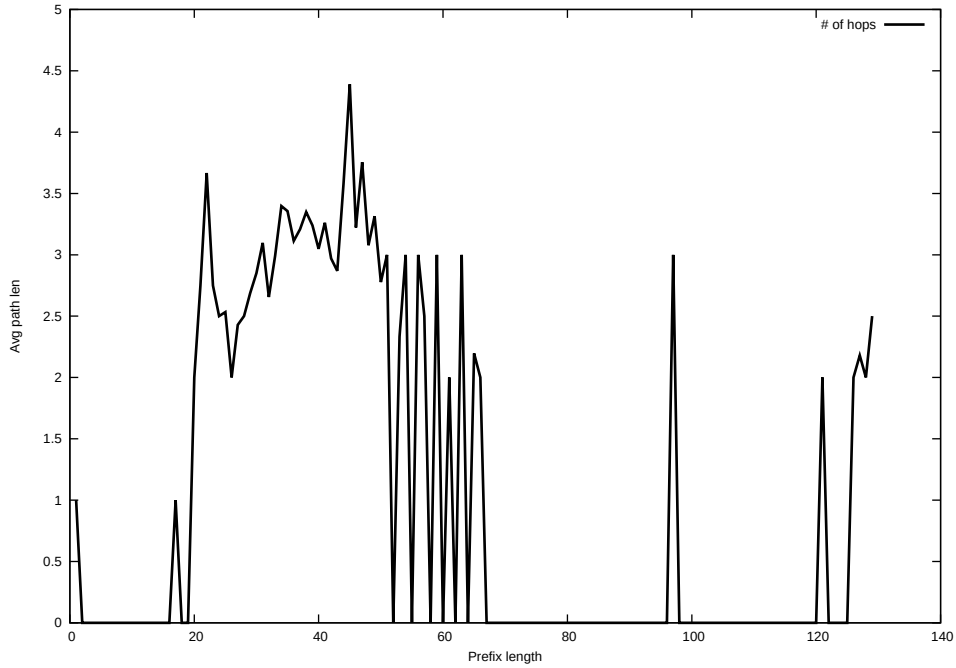
2015-06-05



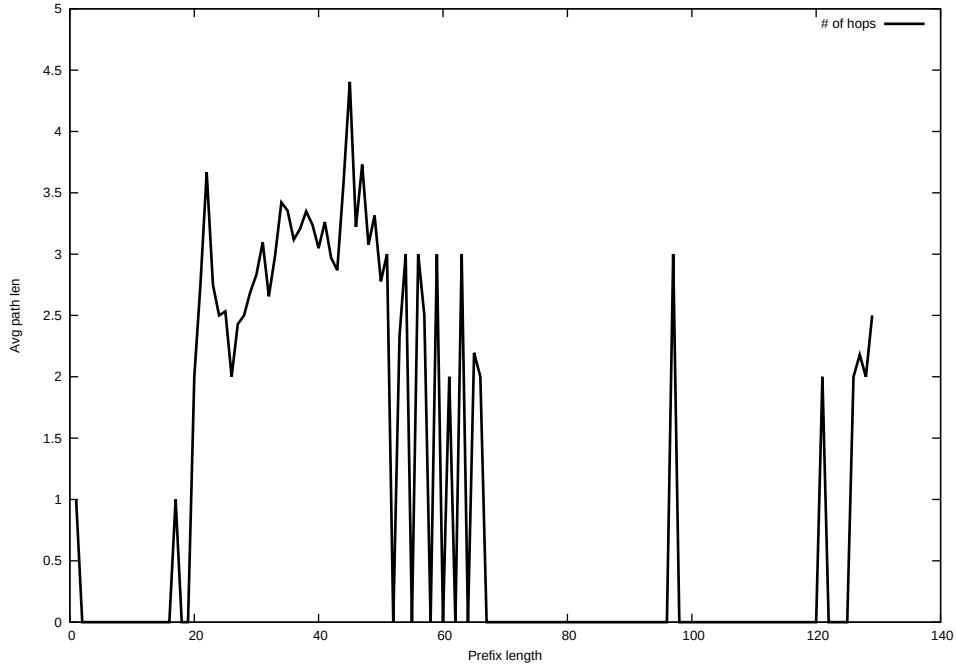
2015-06-06



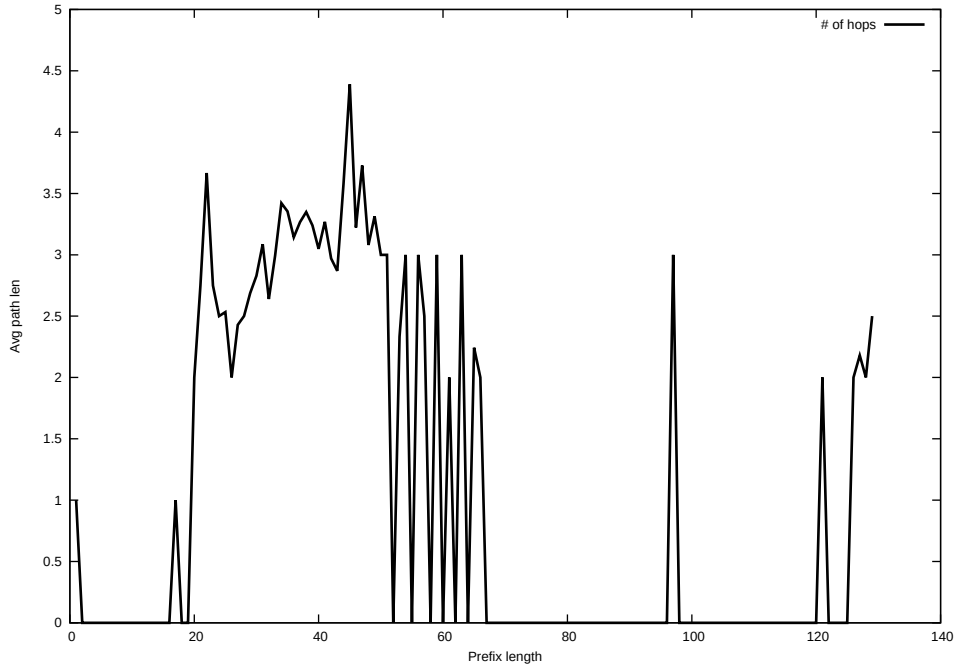
2015-06-07



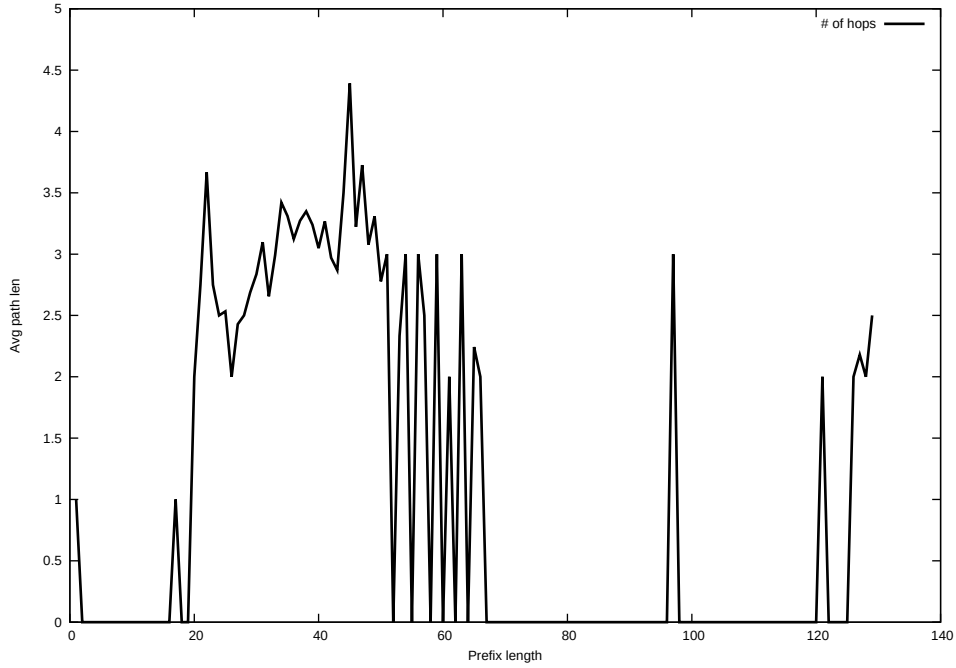
2015-06-08



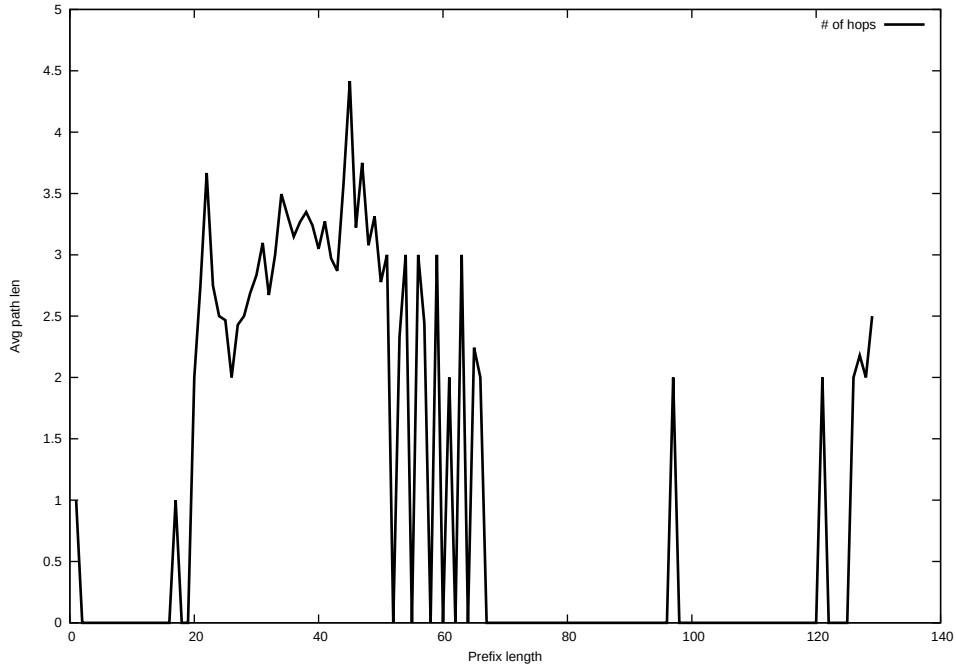
2015-06-09



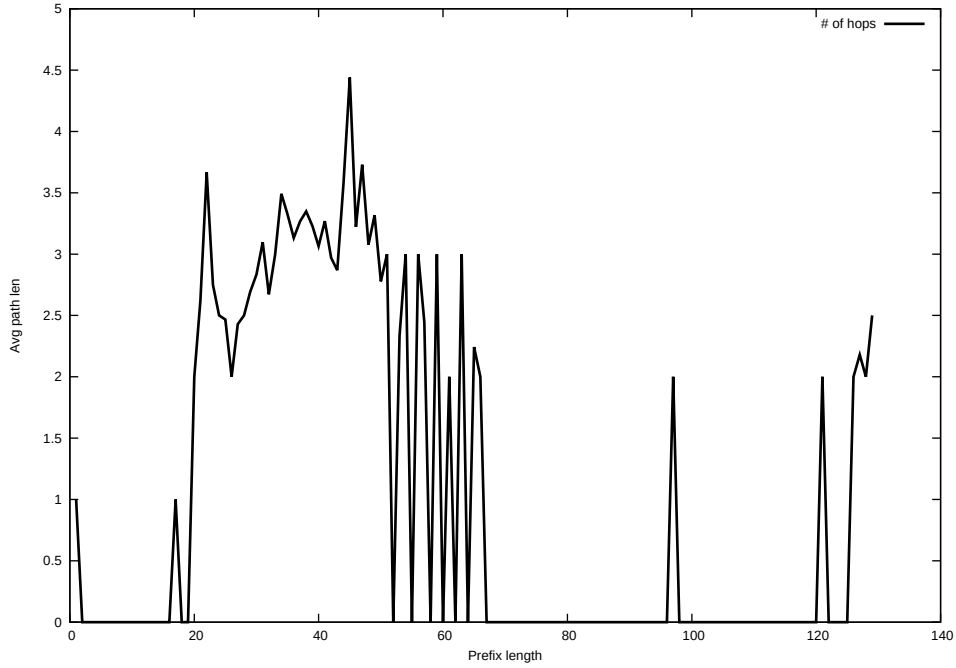
2015-06-10



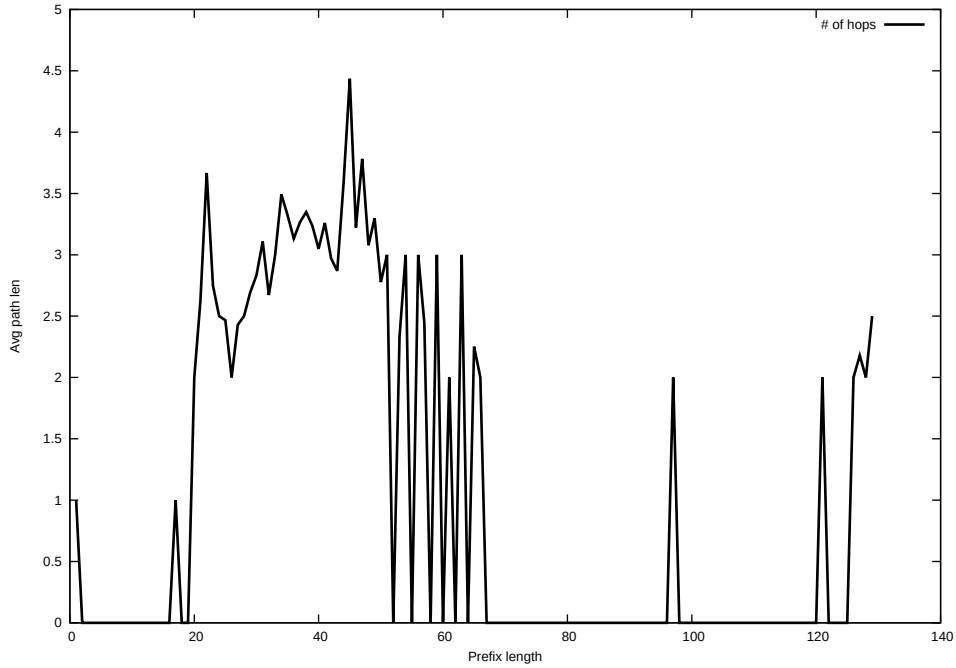
2015-06-11



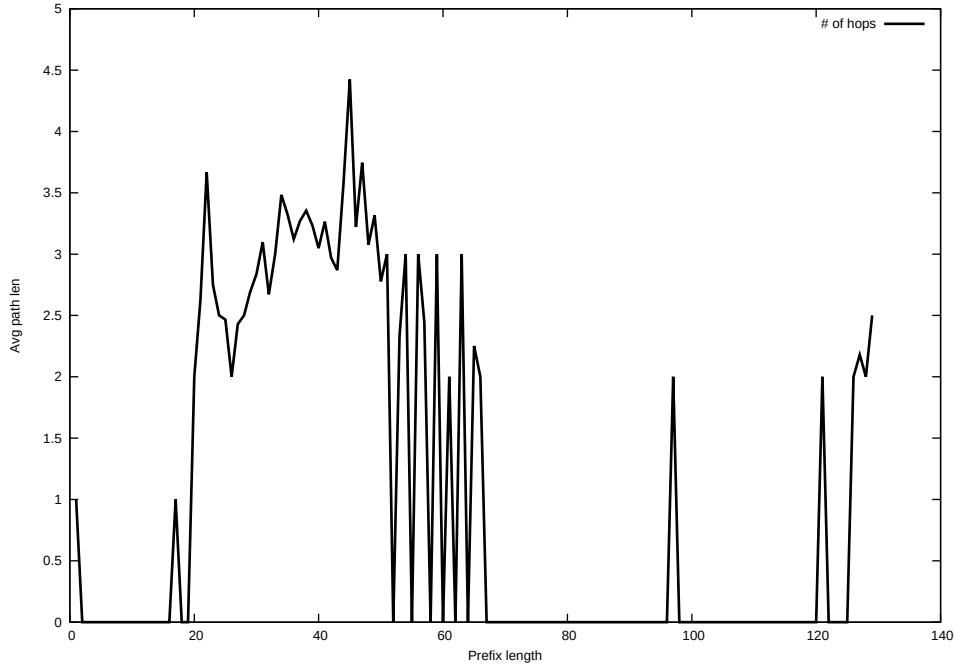
2015-06-12



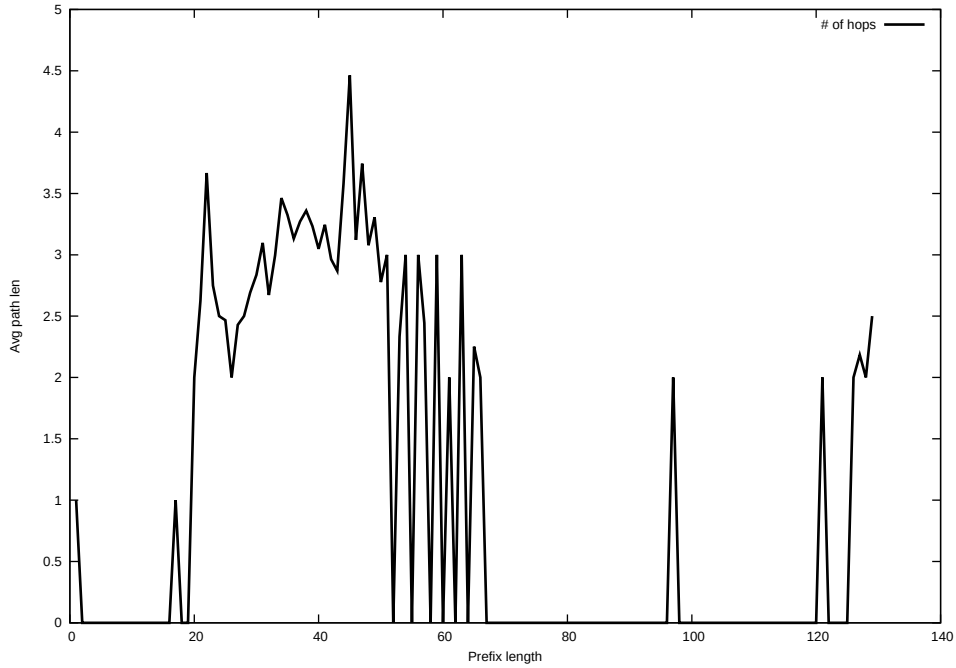
2015-06-13



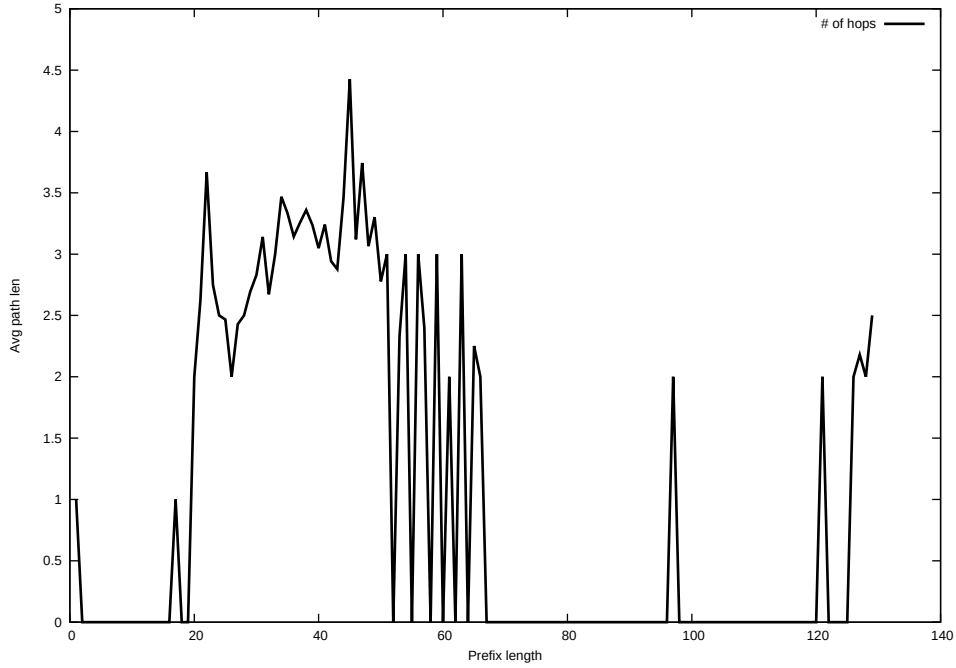
2015-06-14



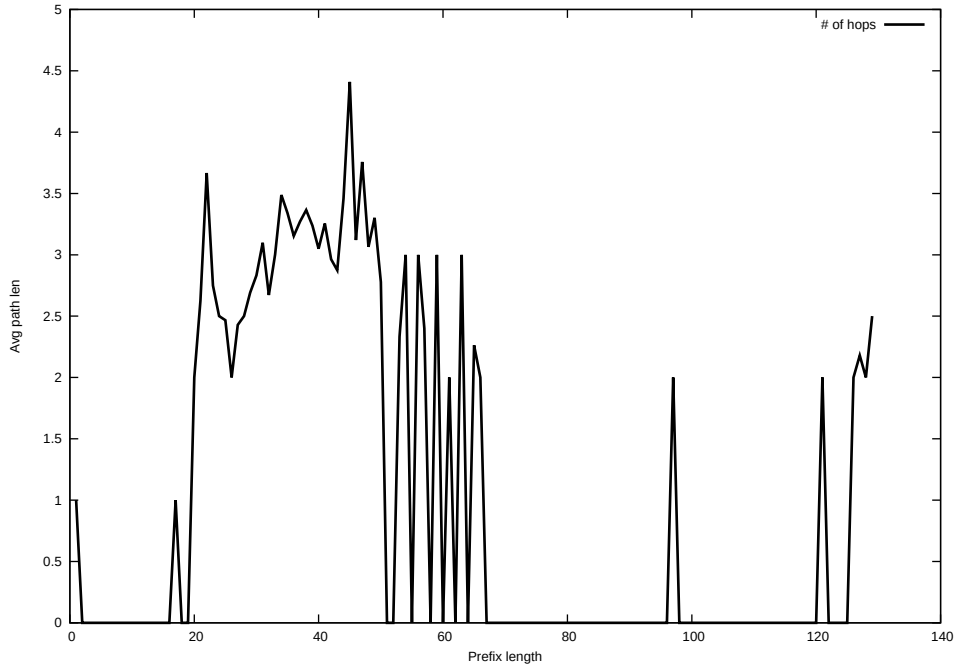
2015-06-15



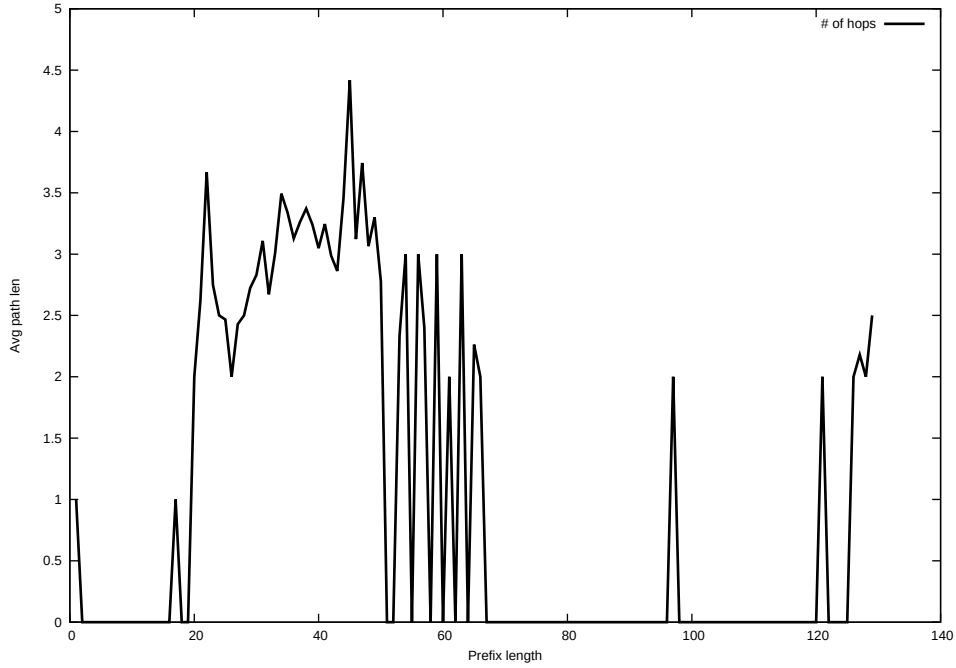
2015-06-16



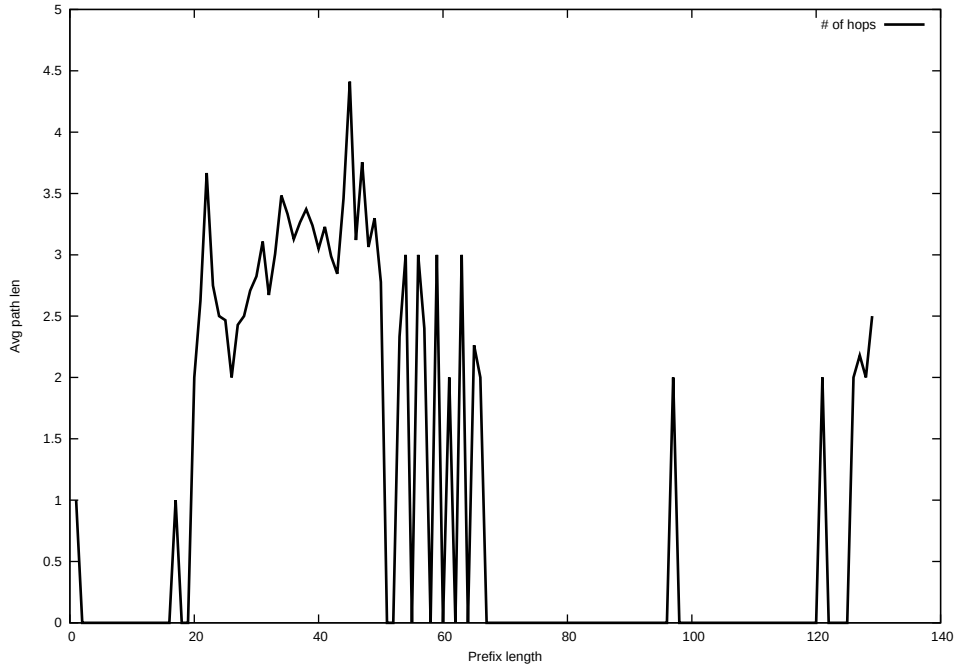
2015-06-17



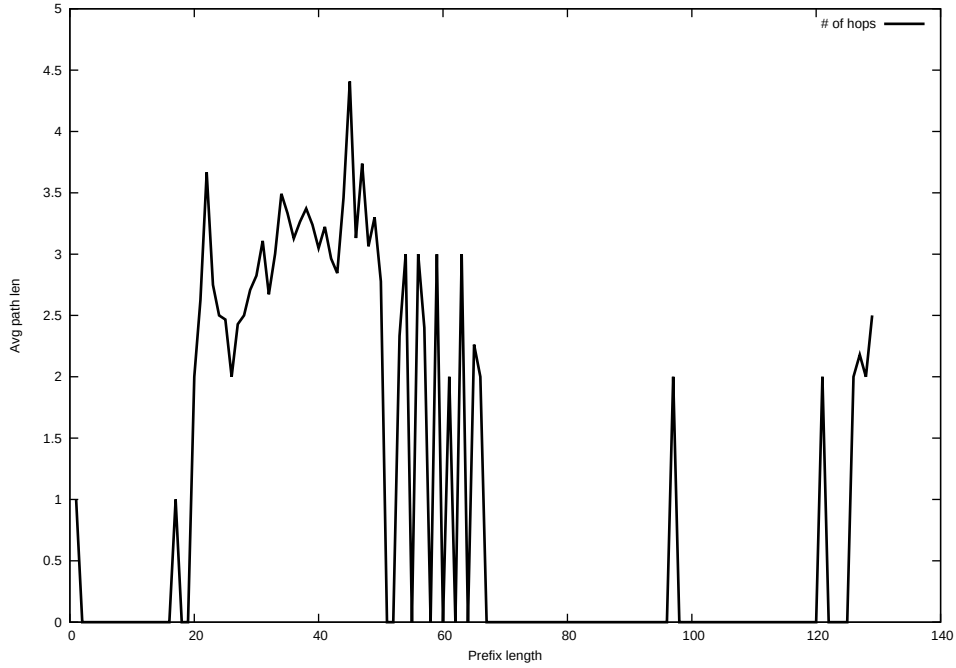
2015-06-18



2015-06-19



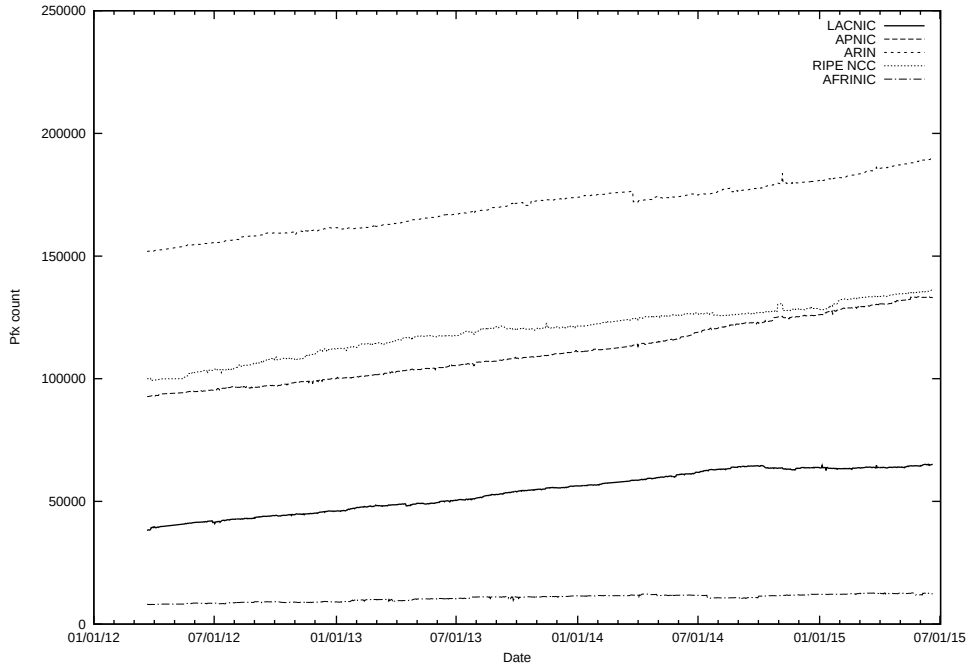
2015-06-20



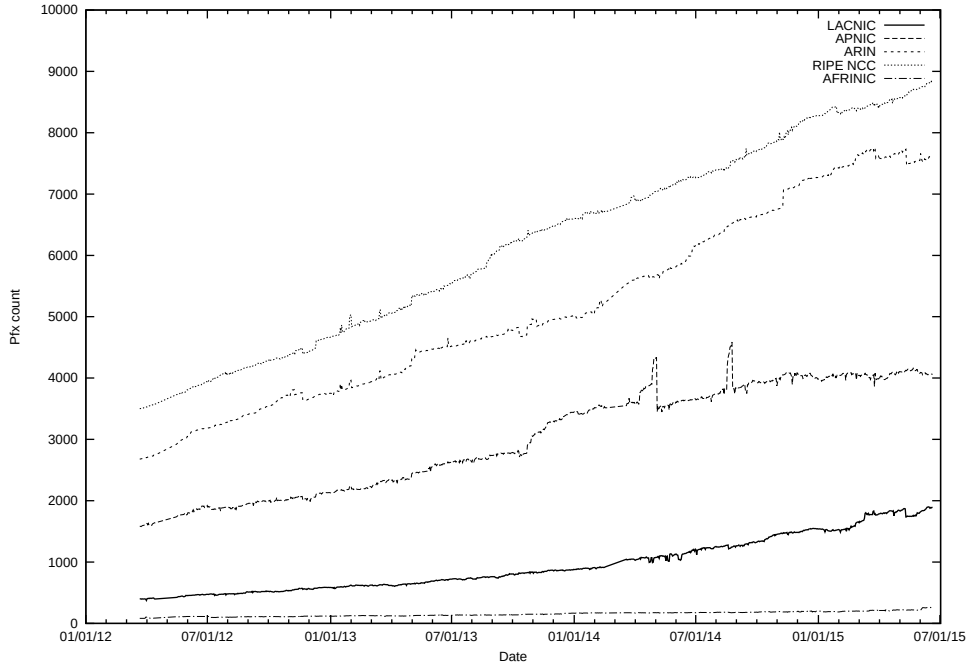
2015-06-21

A.8 RIR service regions share

A.8.1 Prefix counts per RIR

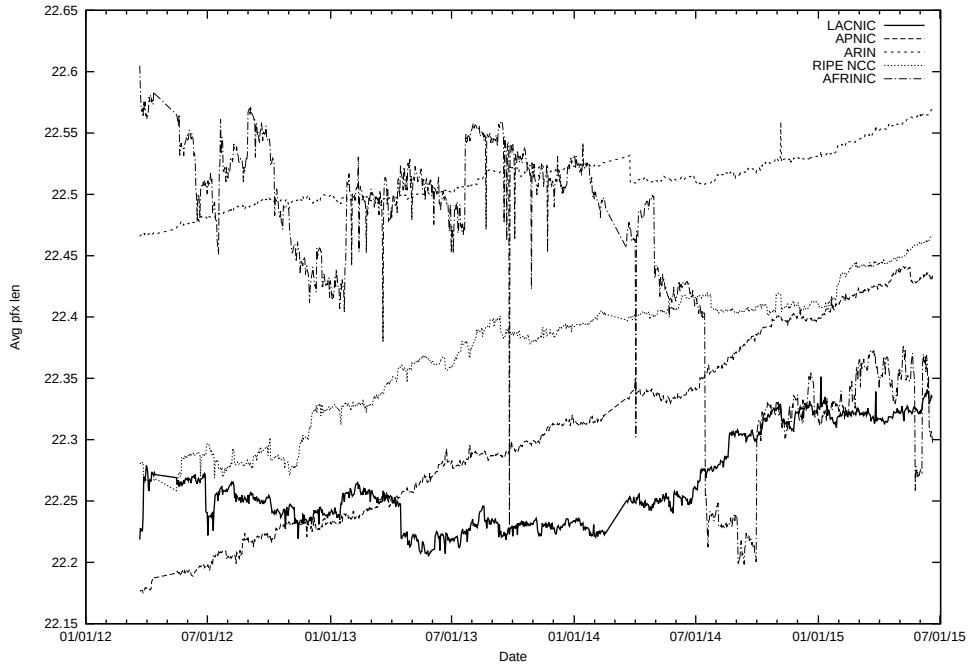


IPv4 BGP prefix count per RIR

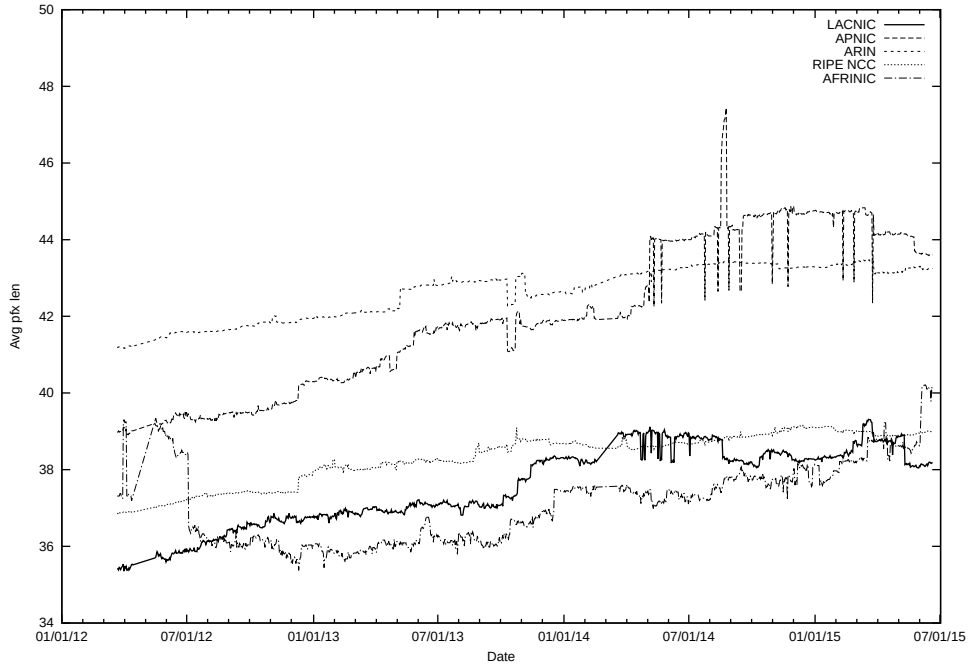


IPv6 BGP prefix count per RIR

A.8.2 Observed avgerage prefix length per RIR



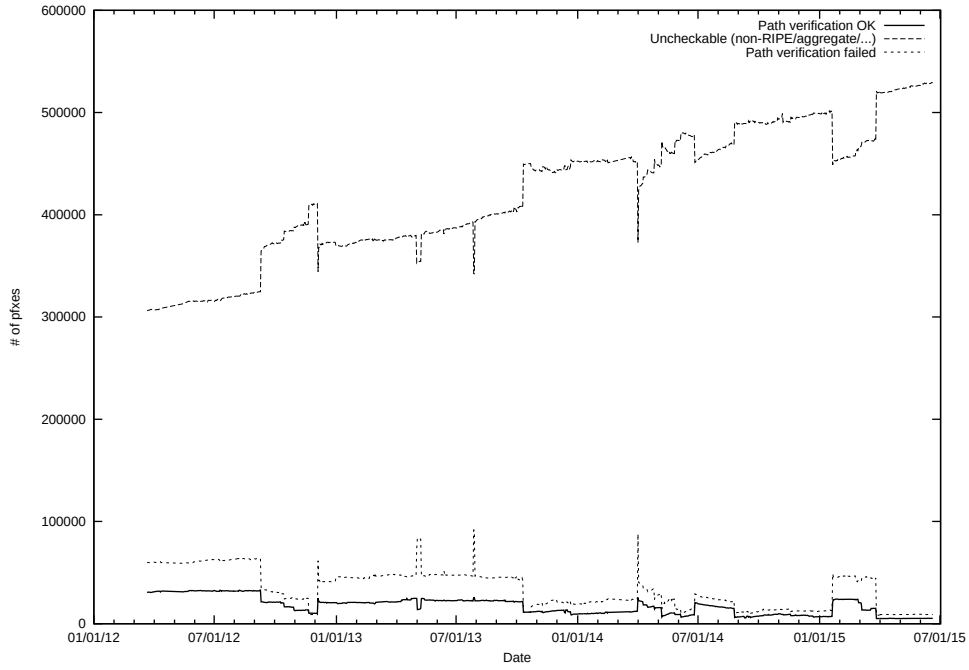
IPv4 BGP average prefix length per RIR



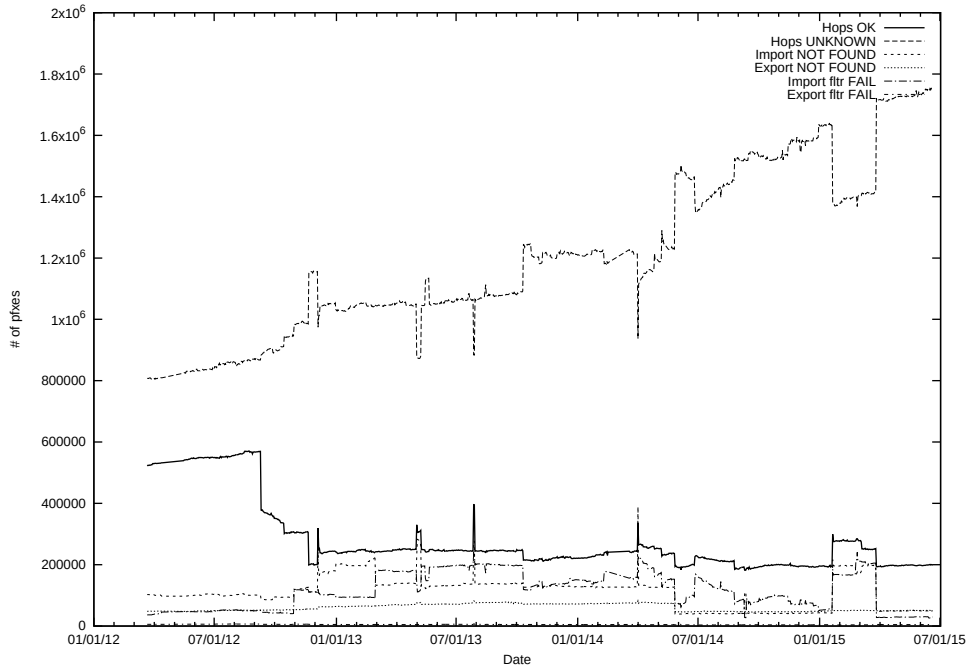
IPv6 BGP average prefix length per RIR

A.9 BGP path verification results

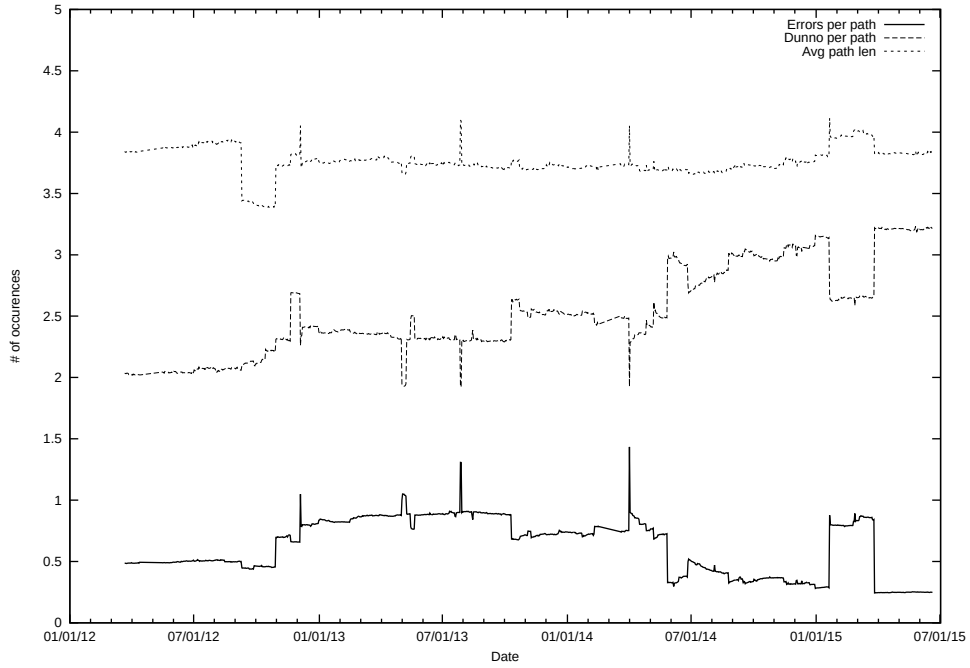
A.9.1 IPv4



Path verification results

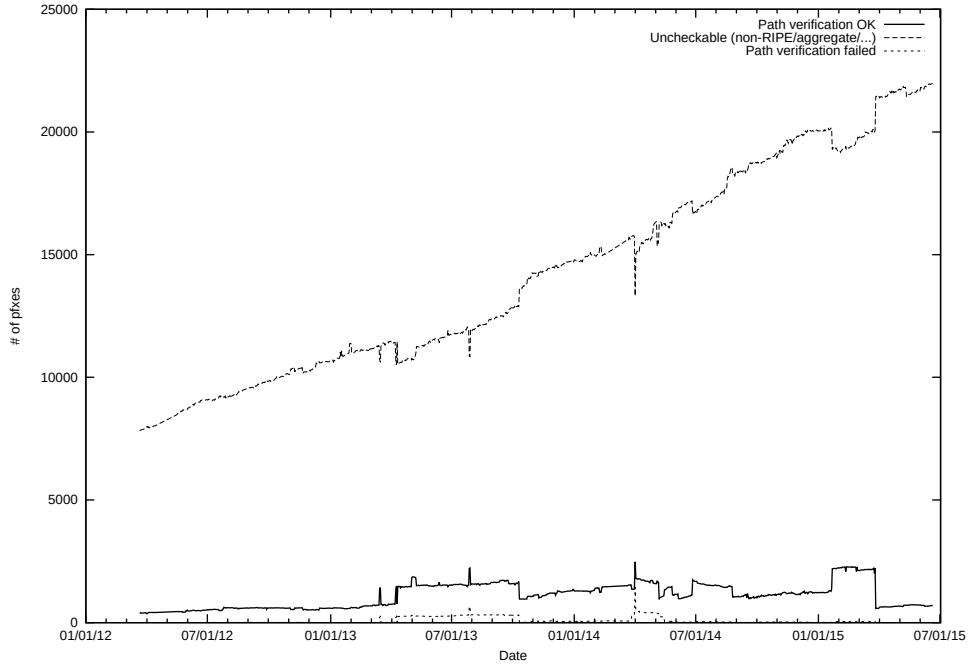


Path verification details

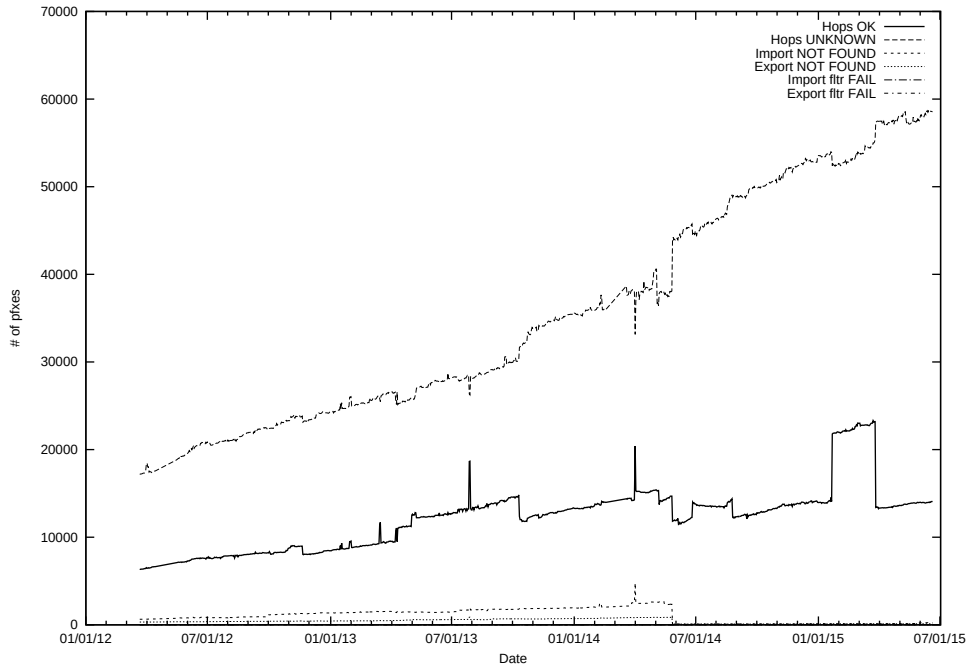


Path verification errors per path

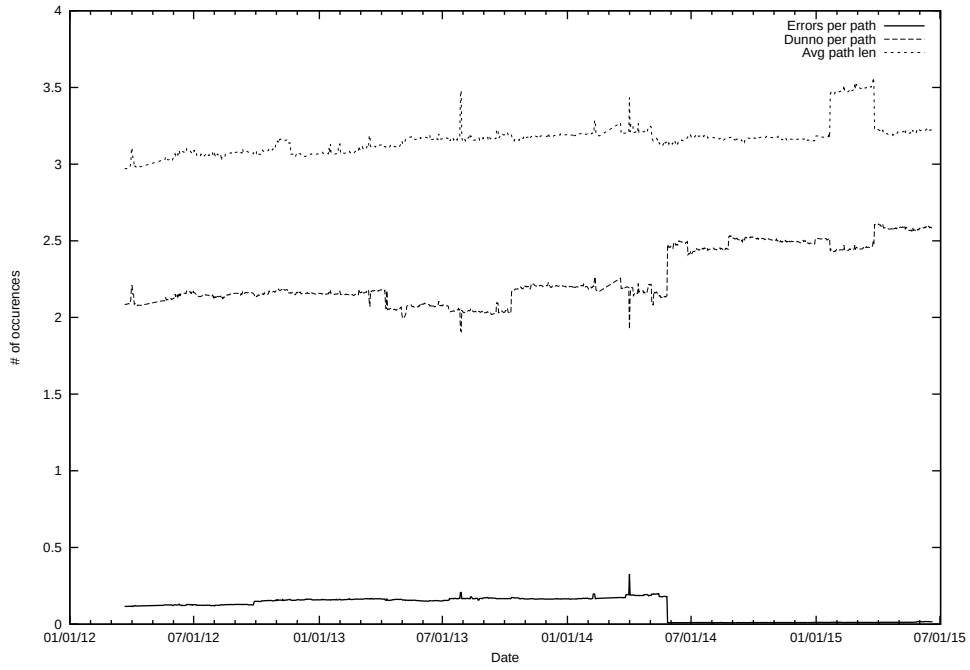
A.9.2 IPv6



Path verification results

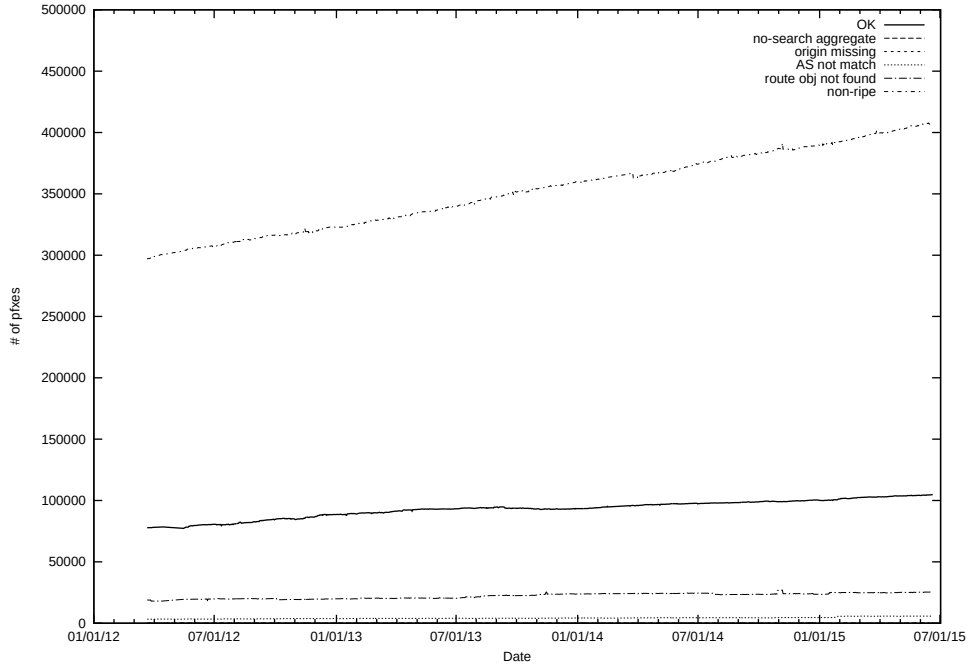


Path verification details

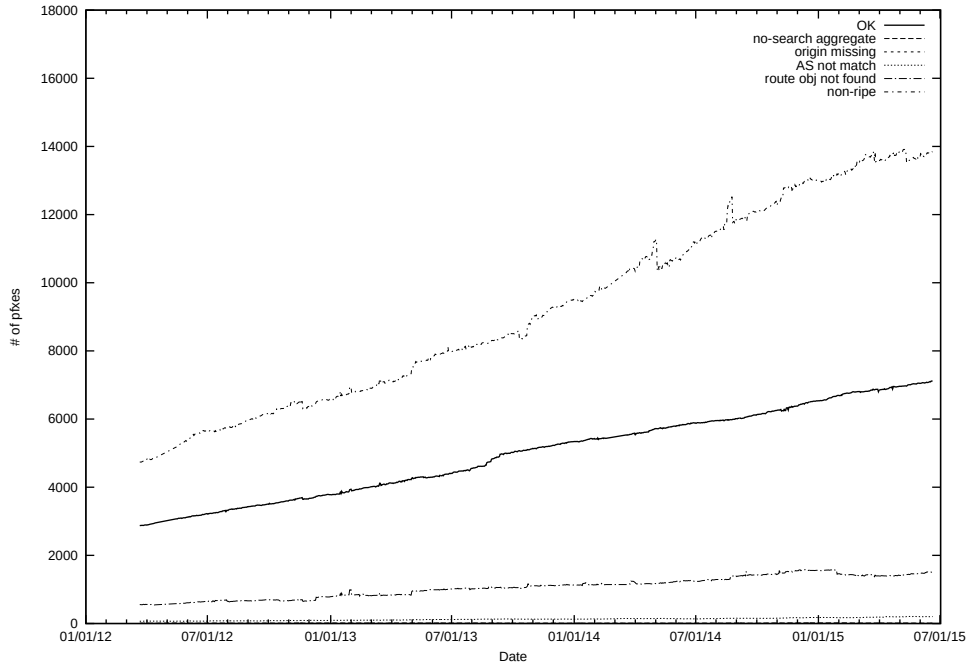


Path verification errors per path

A.10 BGP origin verification results

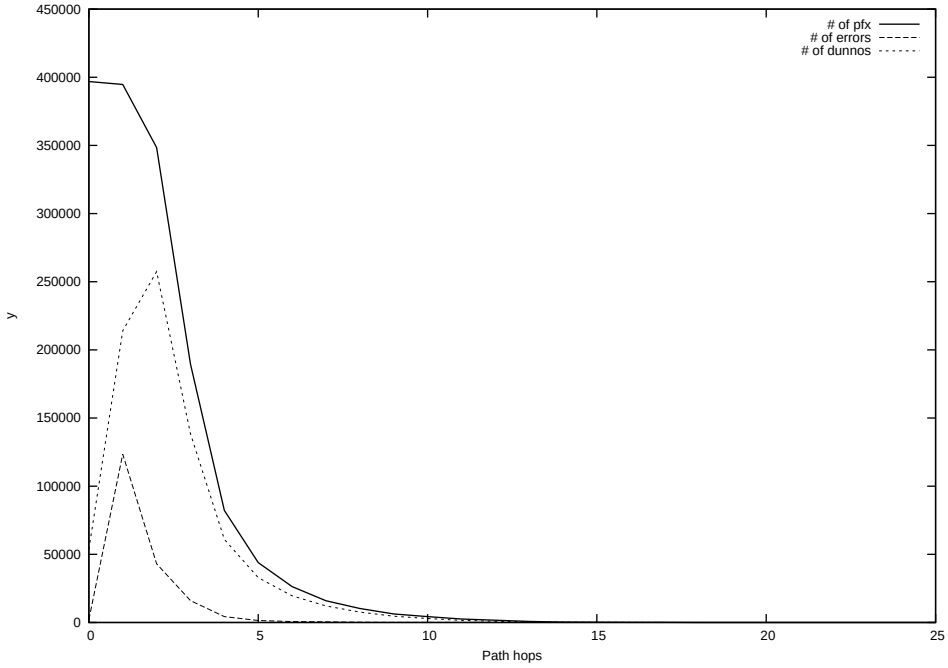


IPv4 BGP origin verification results

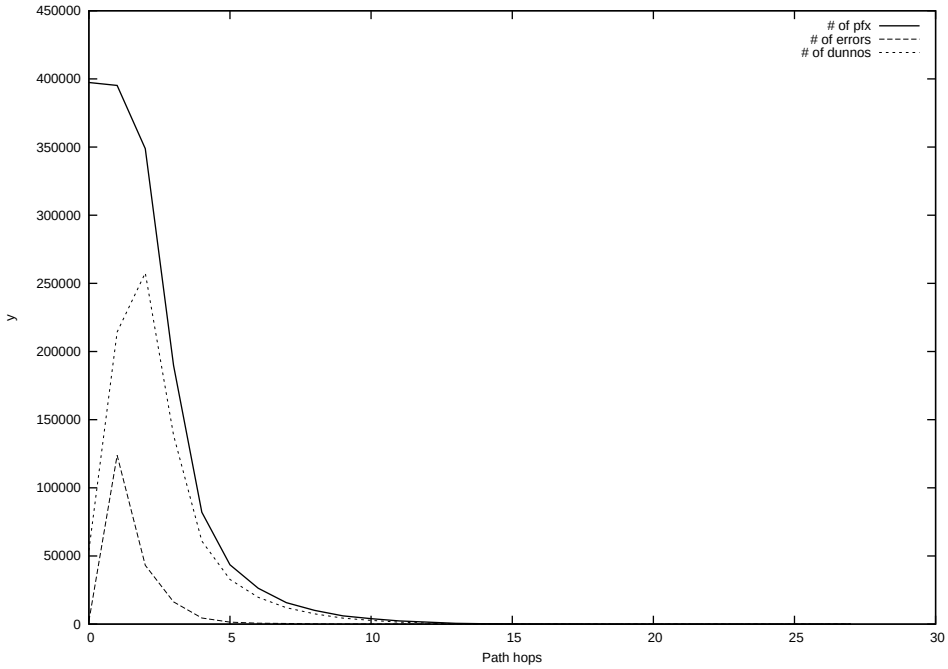


IPv6 BGP origin verification results

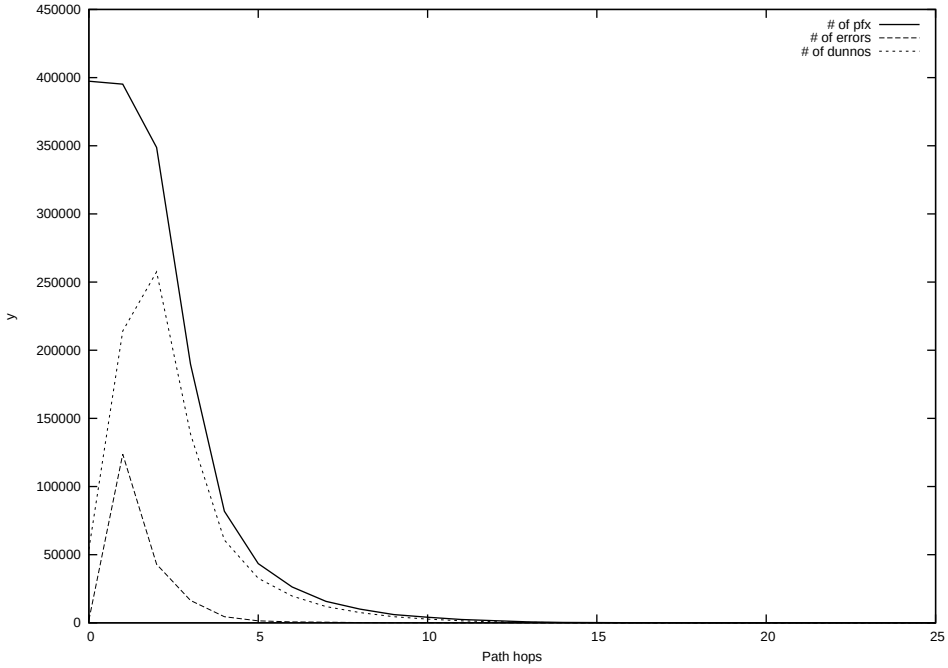
A.11 IPv4 BGP full paths matched against RIPE DB



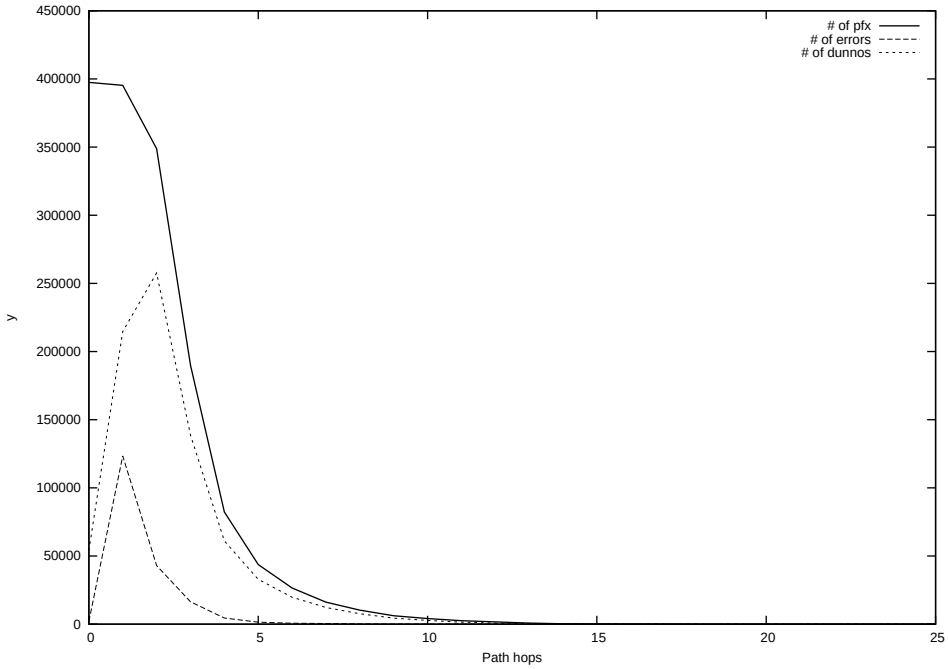
2012-03-22



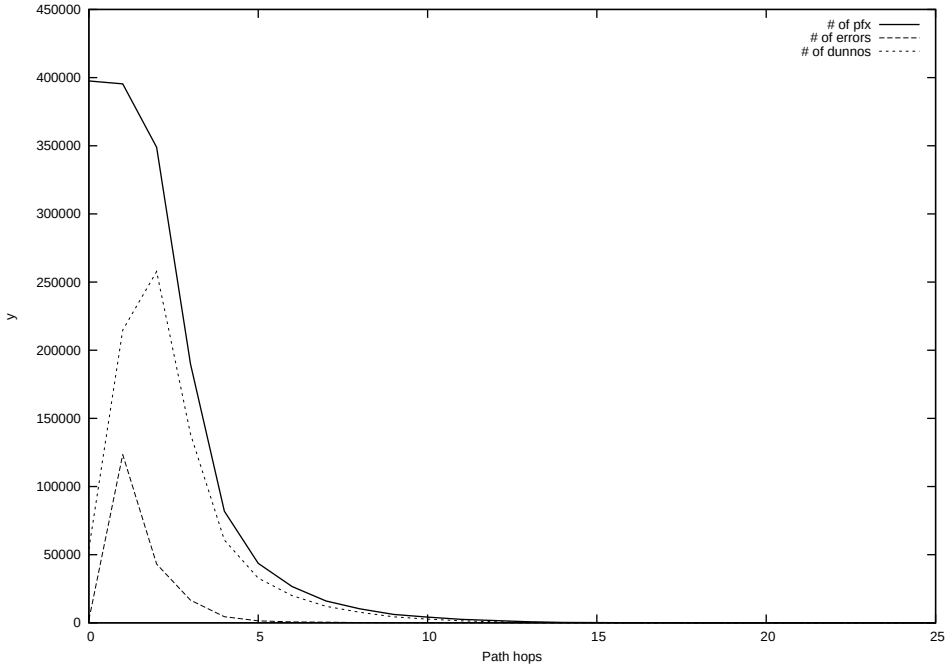
2012-03-23



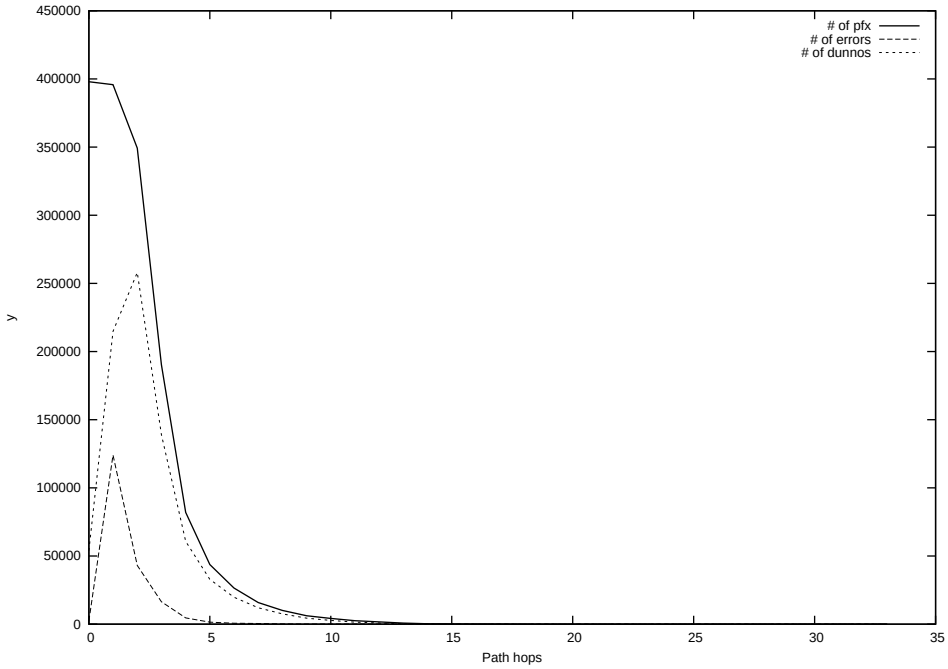
2012-03-24



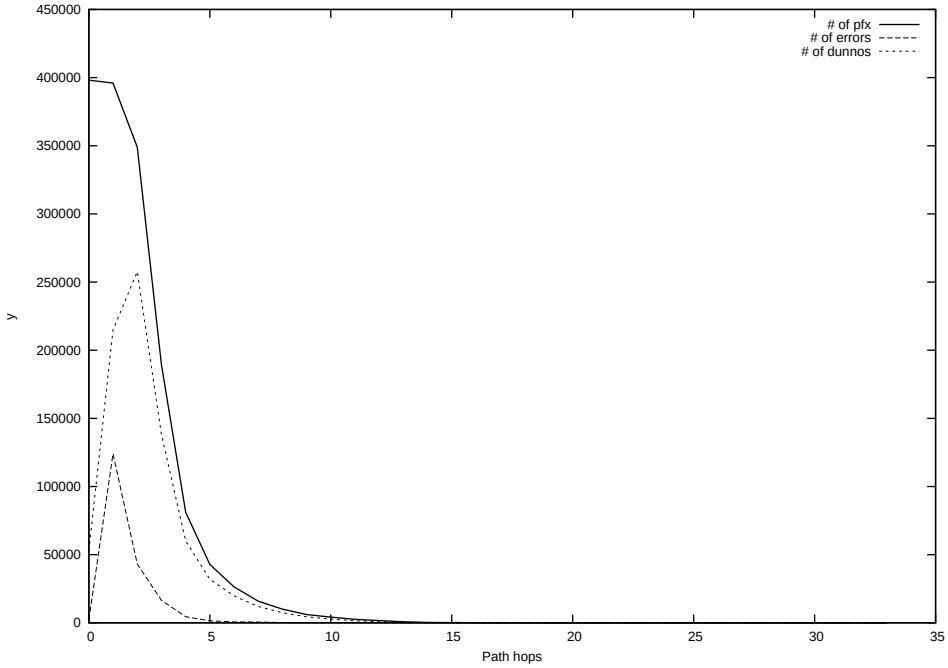
2012-03-25



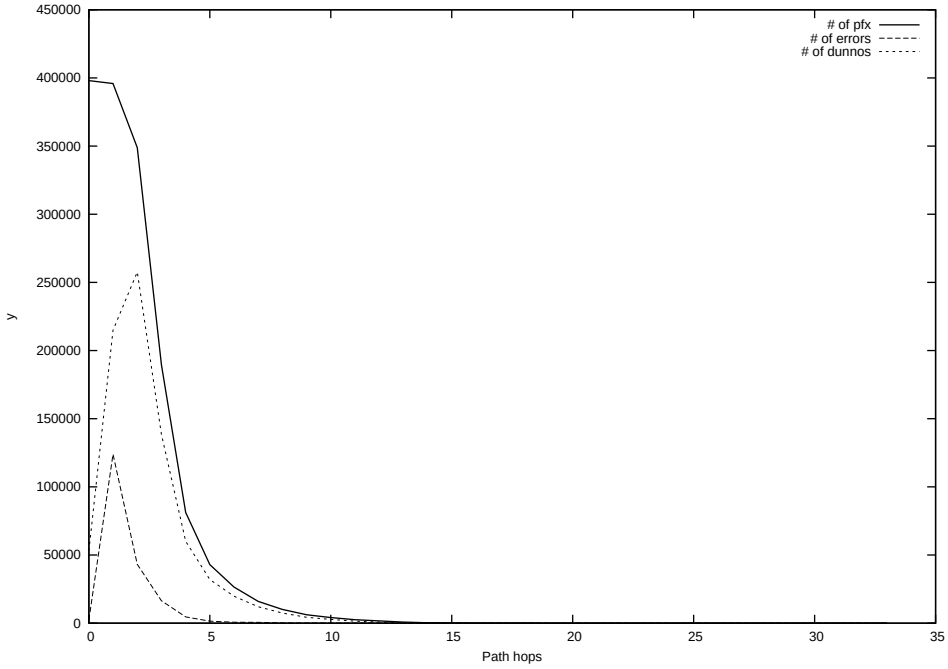
2012-03-26



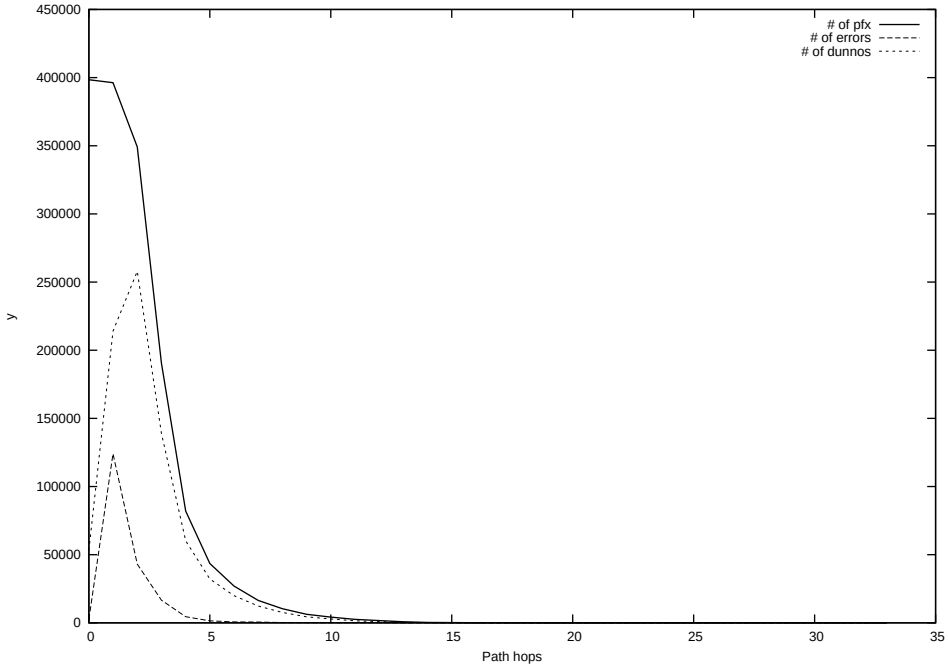
2012-03-27



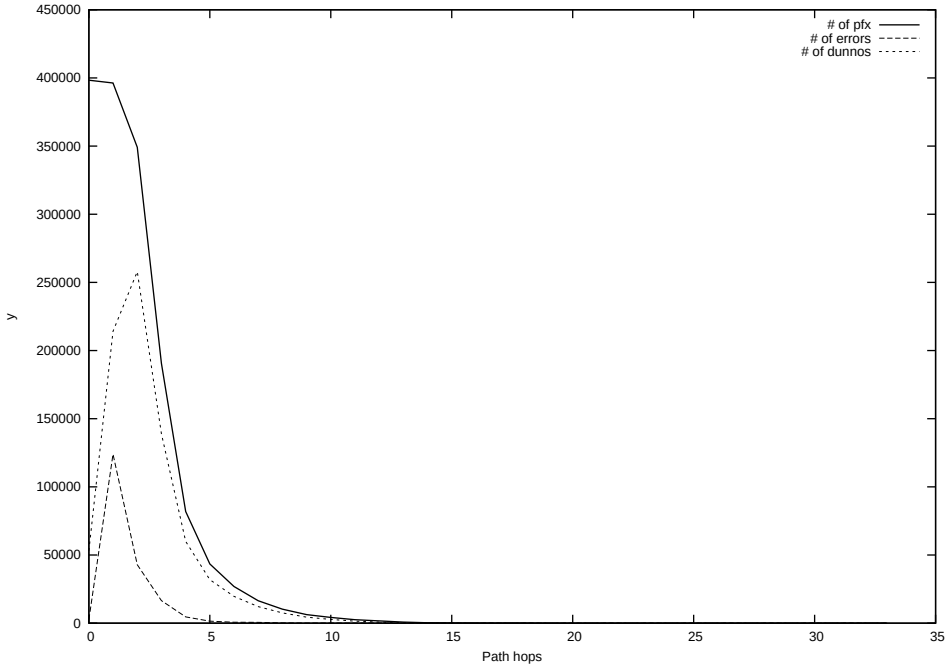
2012-03-28



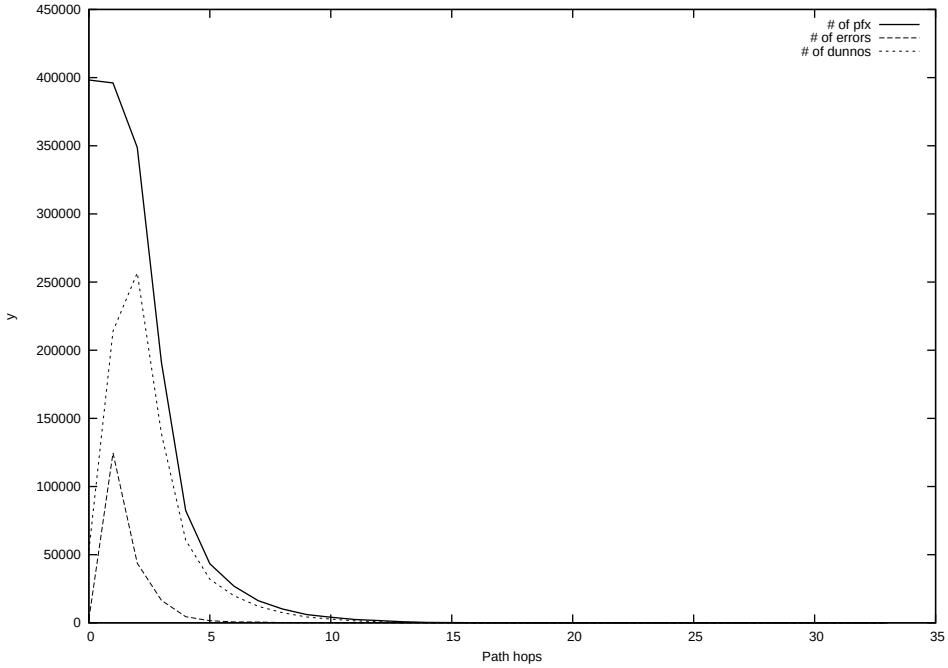
2012-03-29



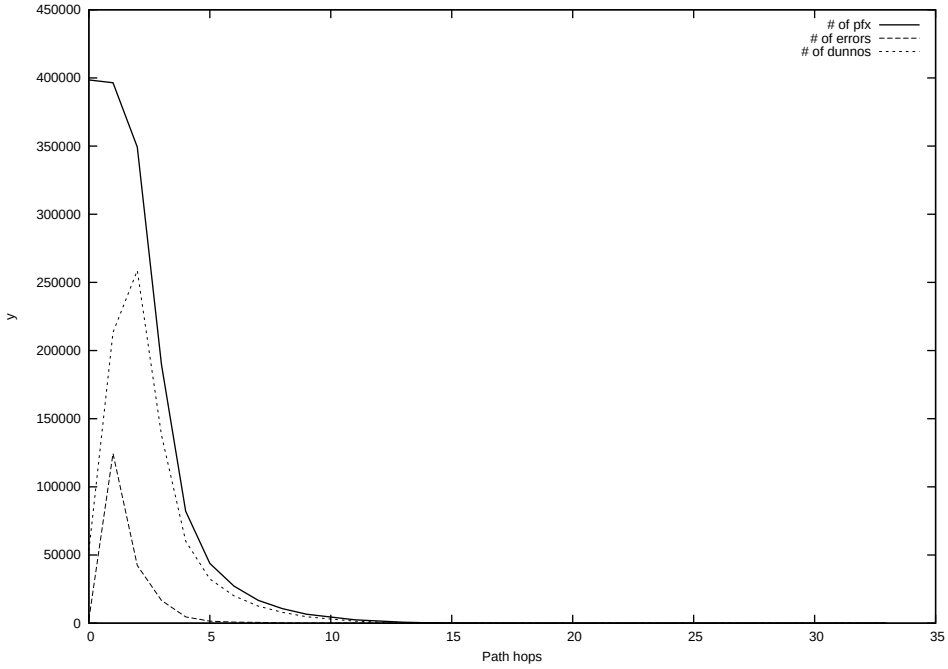
2012-03-30



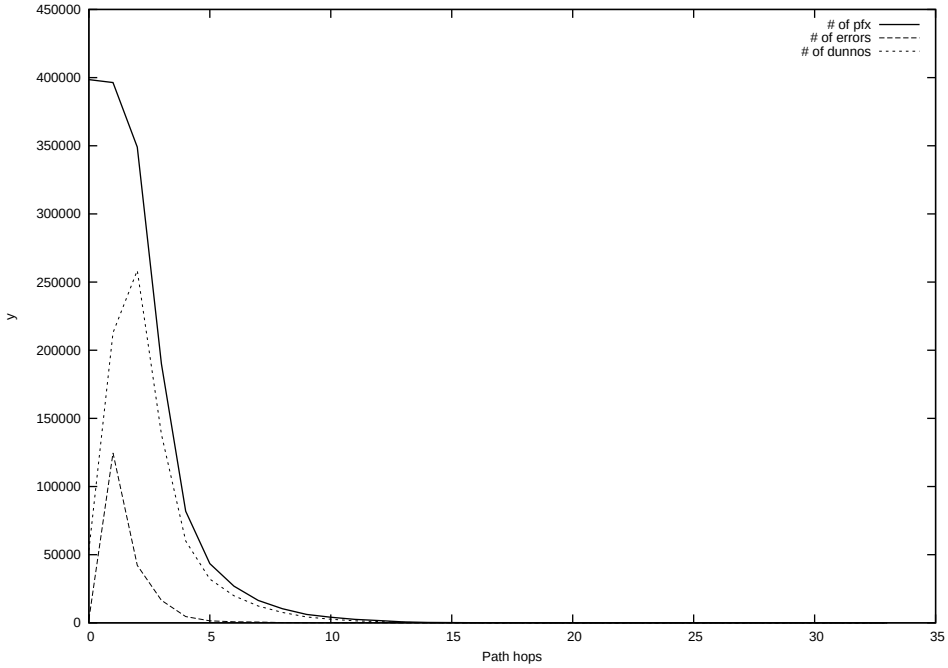
2012-03-31



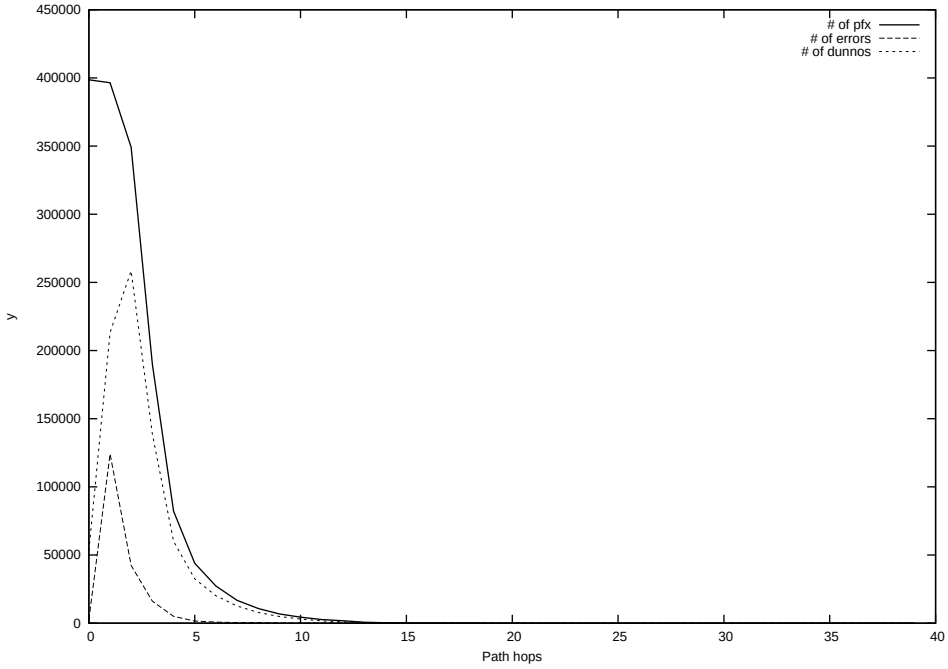
2012-04-01



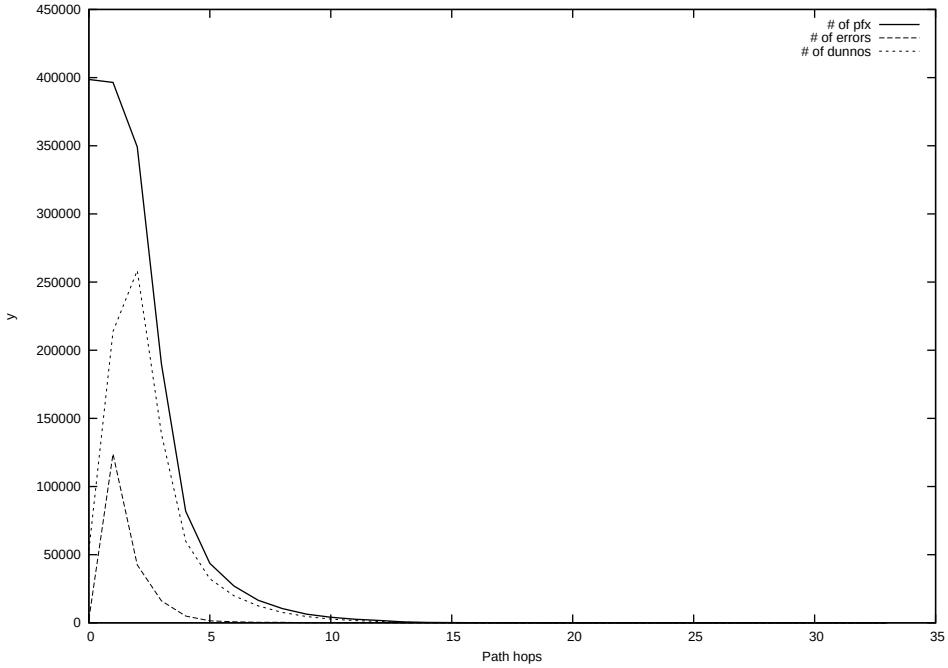
2012-04-02



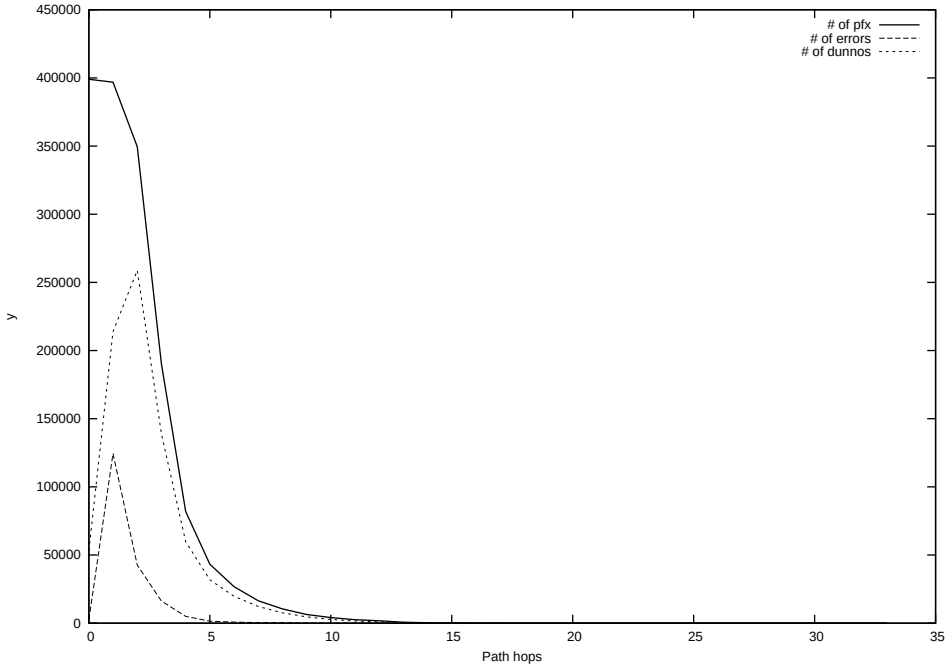
2012-04-03



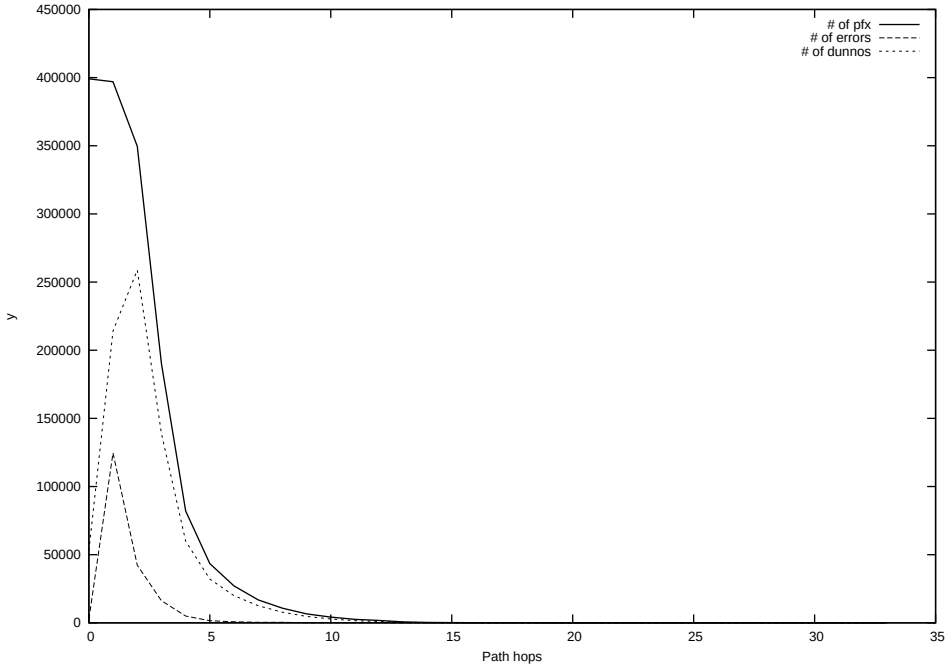
2012-04-04



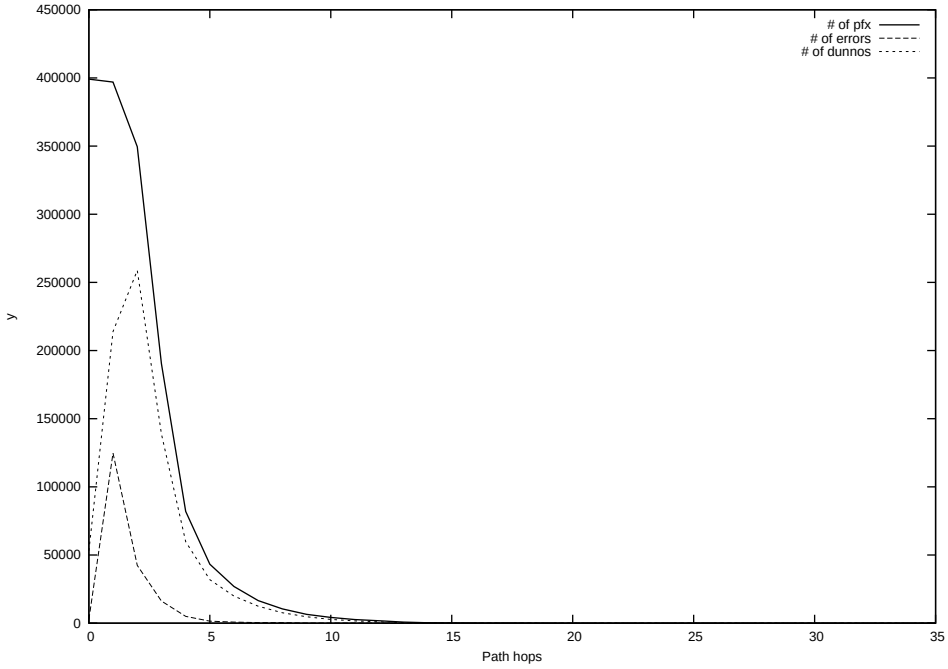
2012-04-05



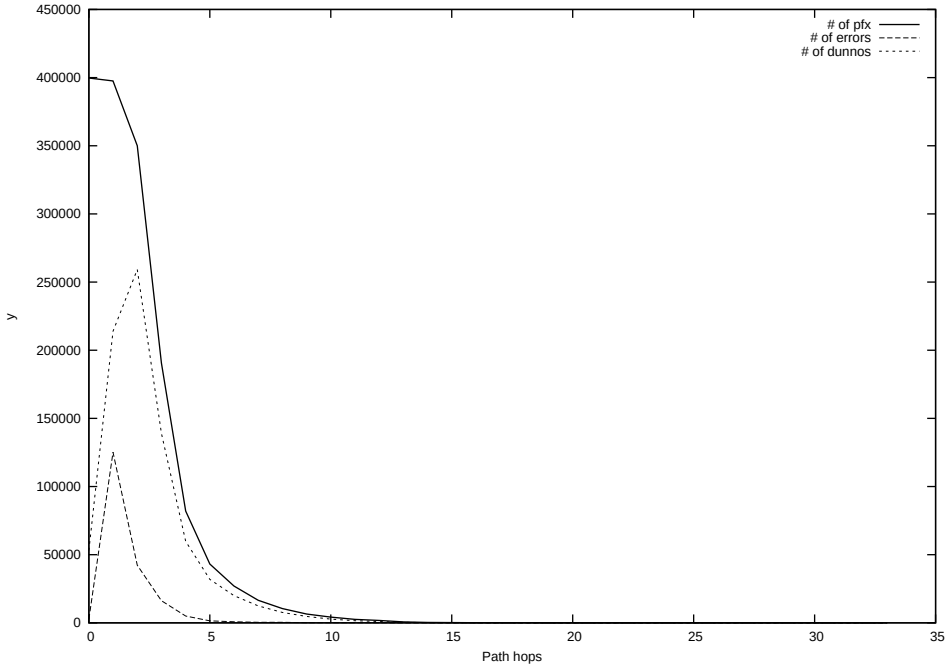
2012-04-06



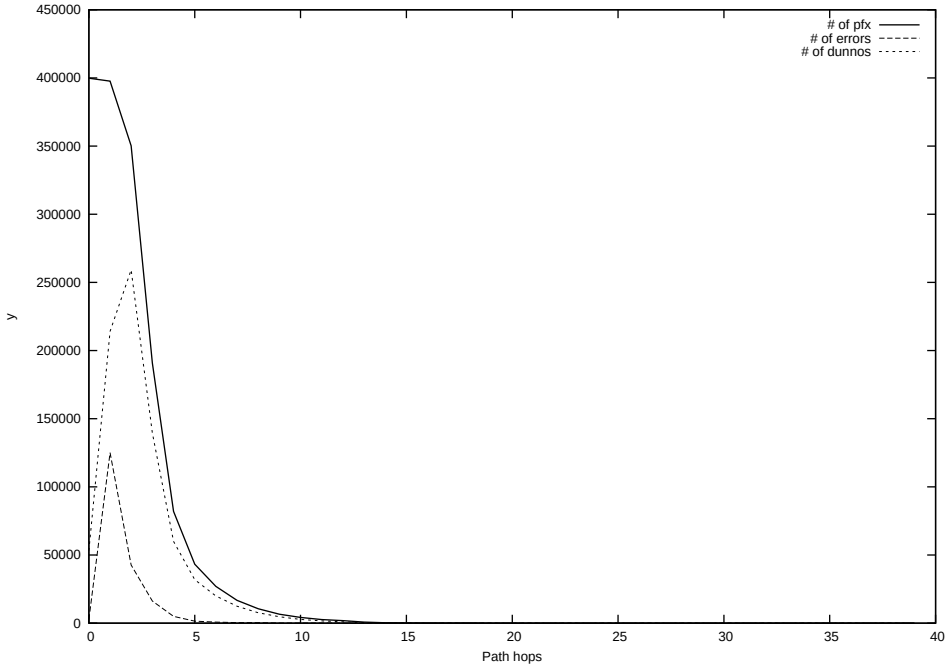
2012-04-07



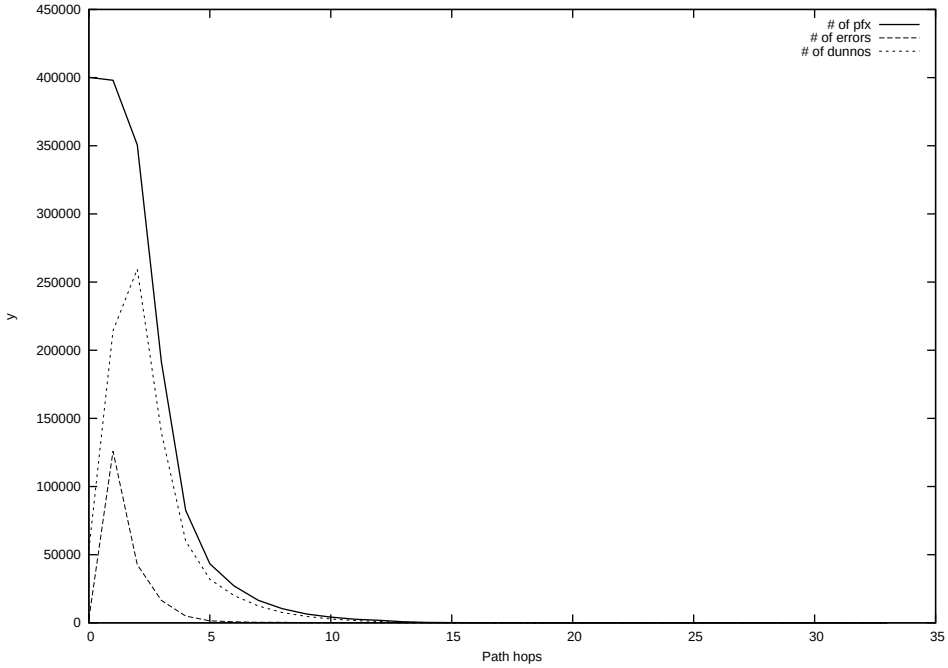
2012-04-08



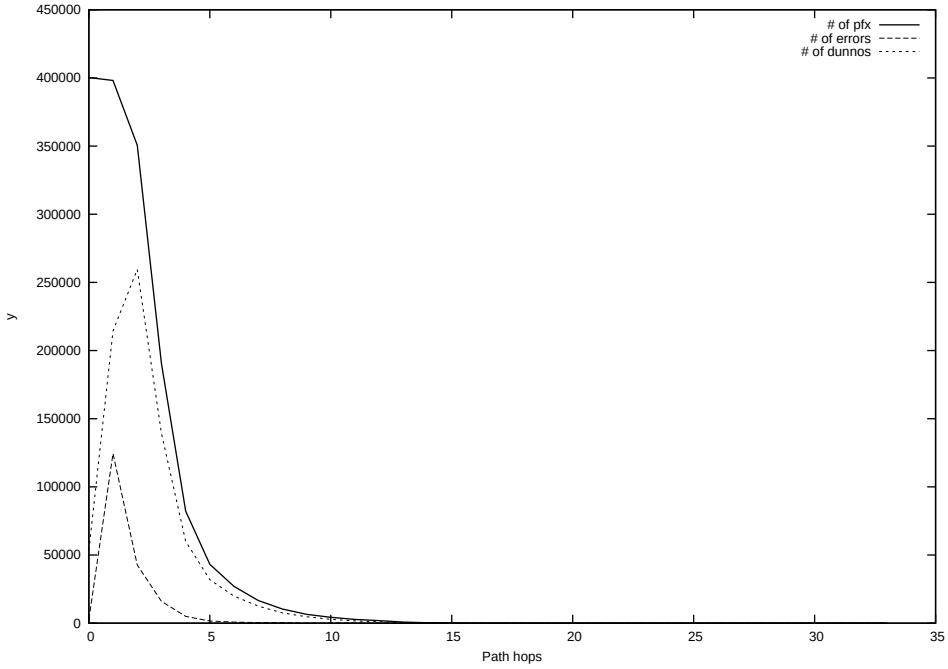
2012-04-09



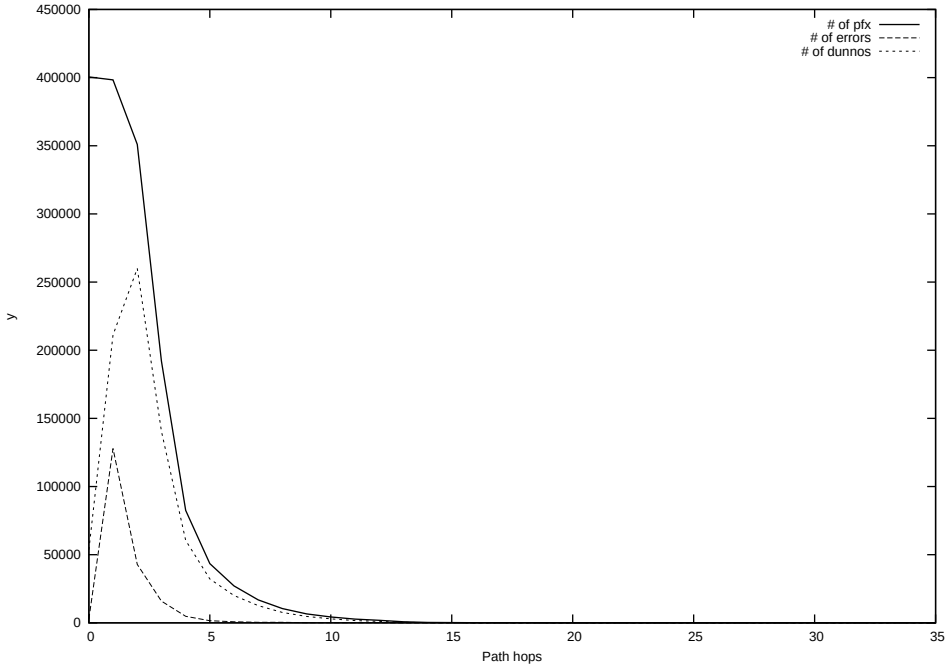
2012-04-10



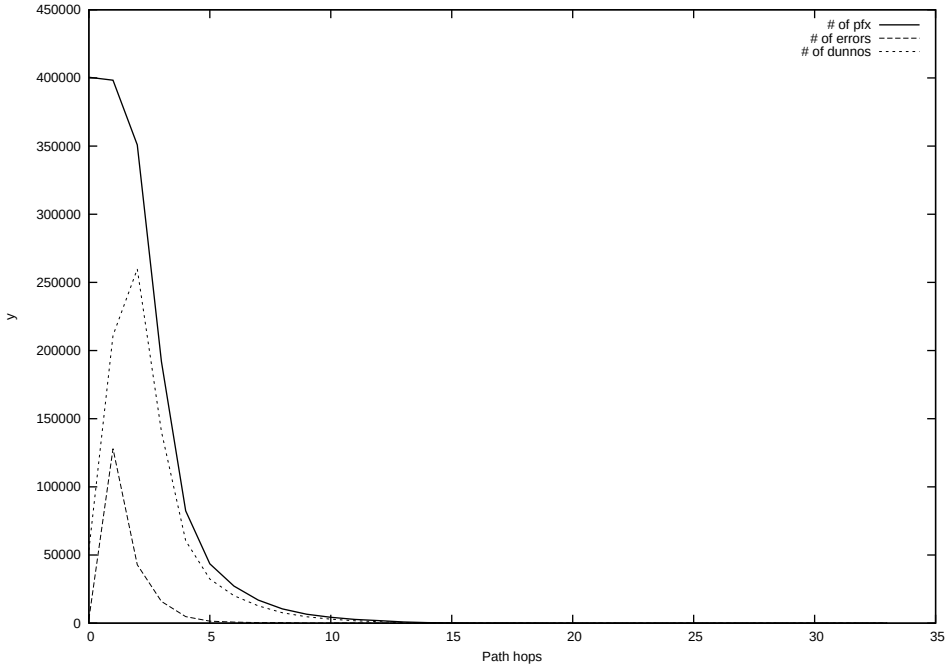
2012-04-11



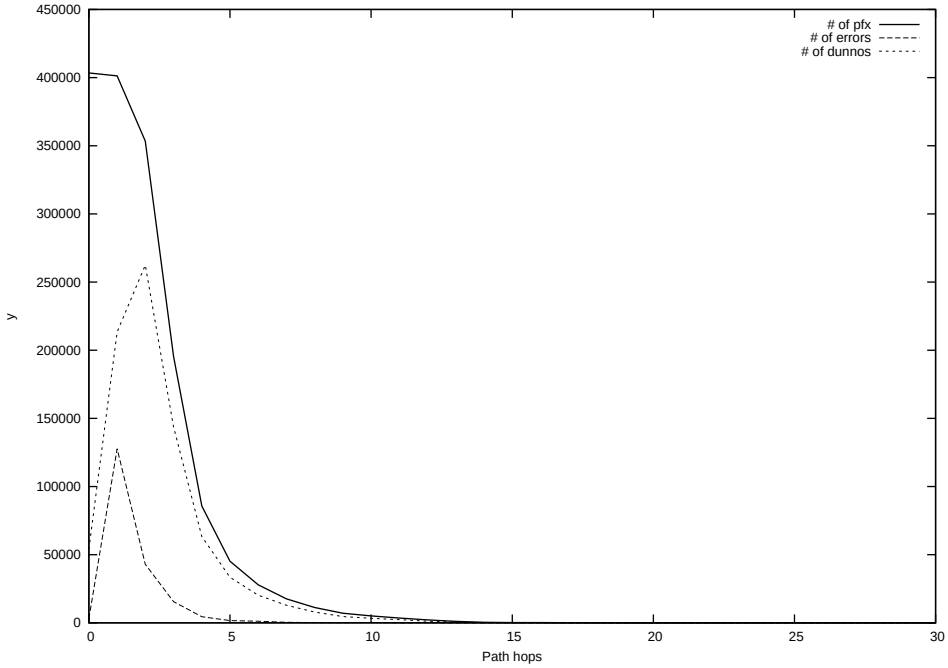
2012-04-12



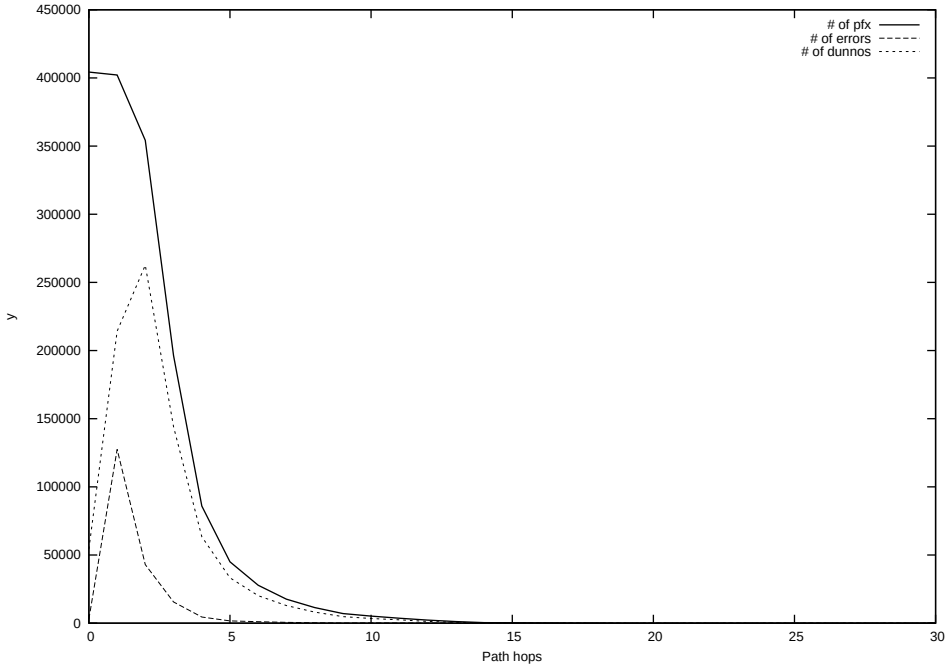
2012-04-13



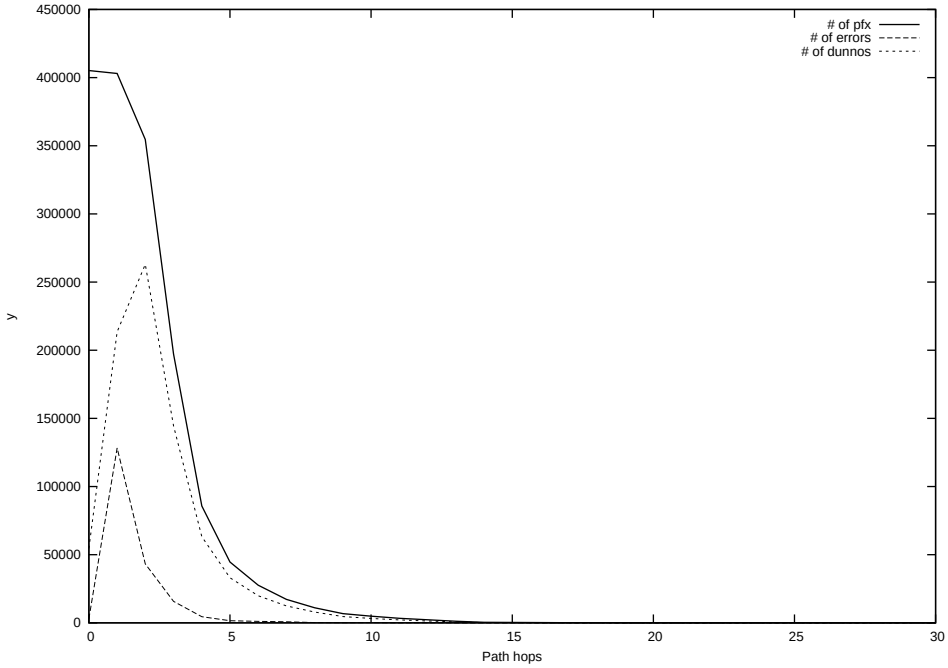
2012-04-14



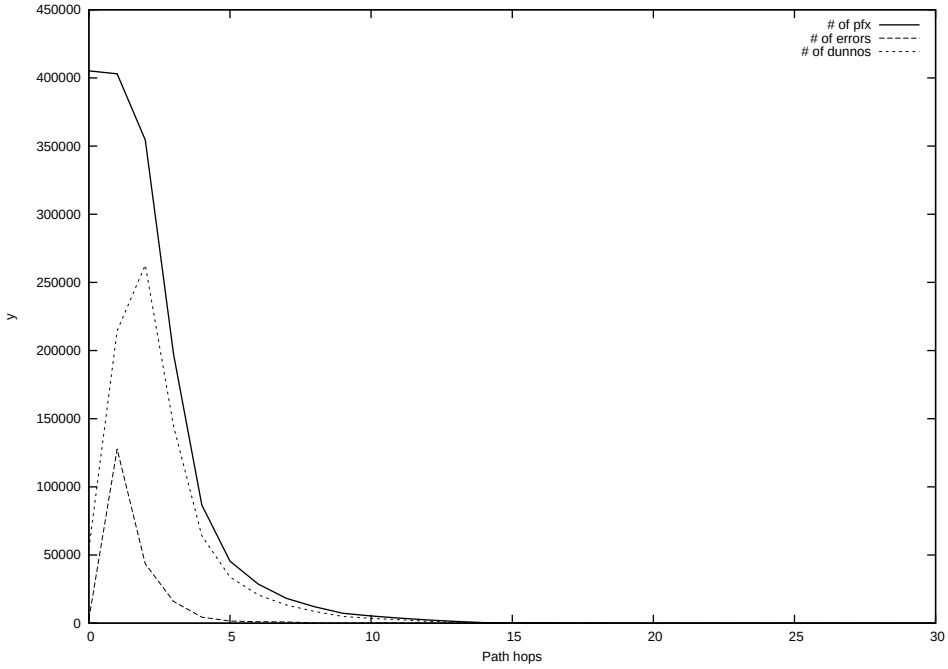
2012-05-16



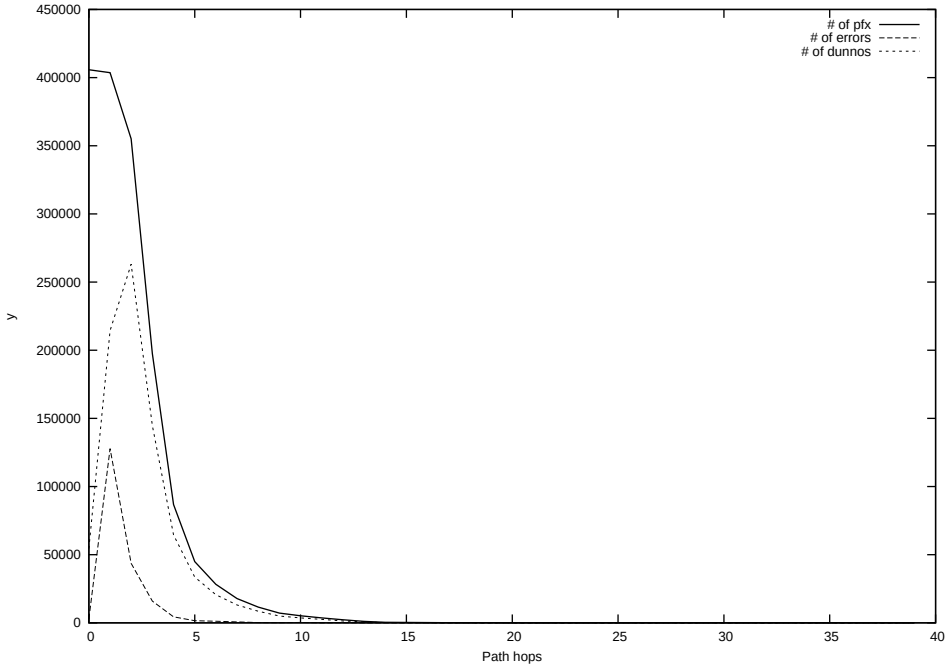
2012-05-17



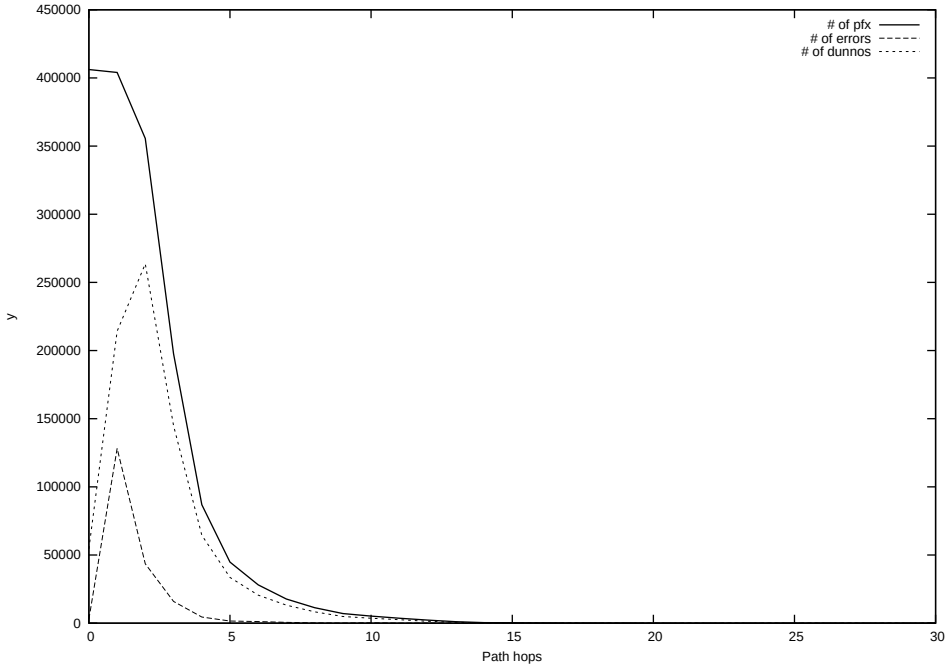
2012-05-19



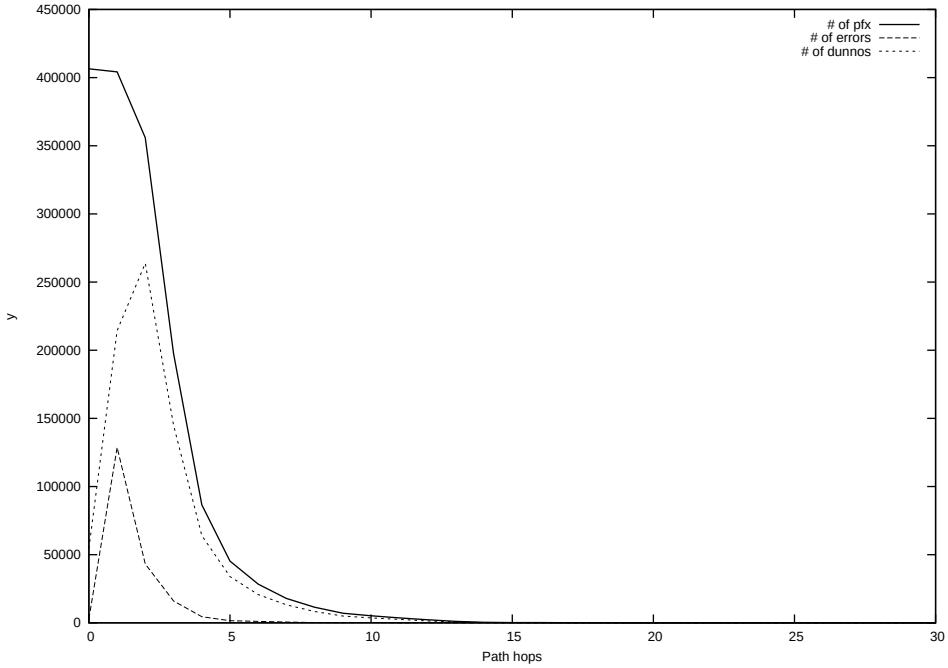
2012-05-20



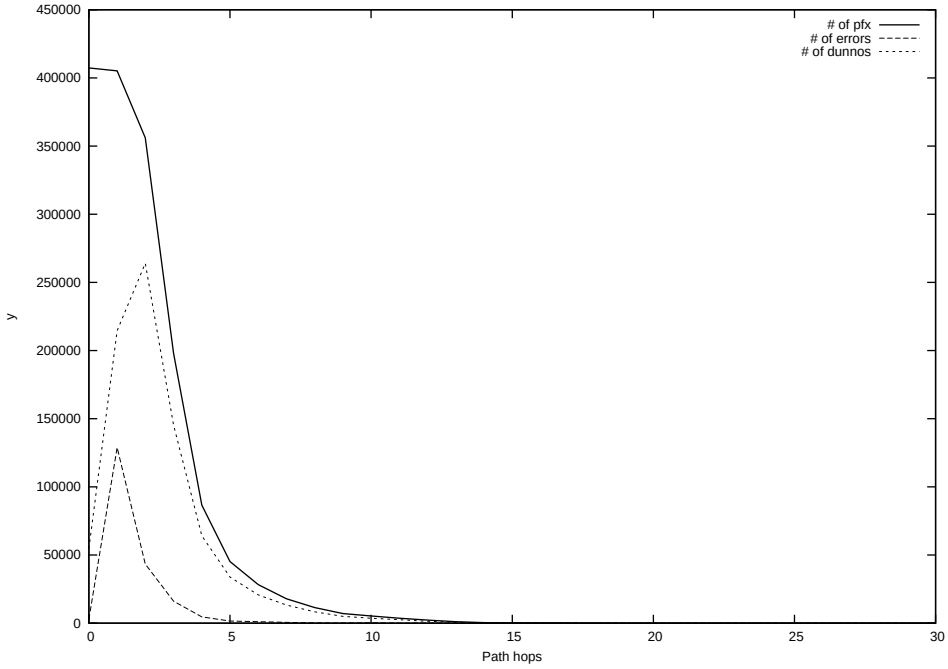
2012-05-21



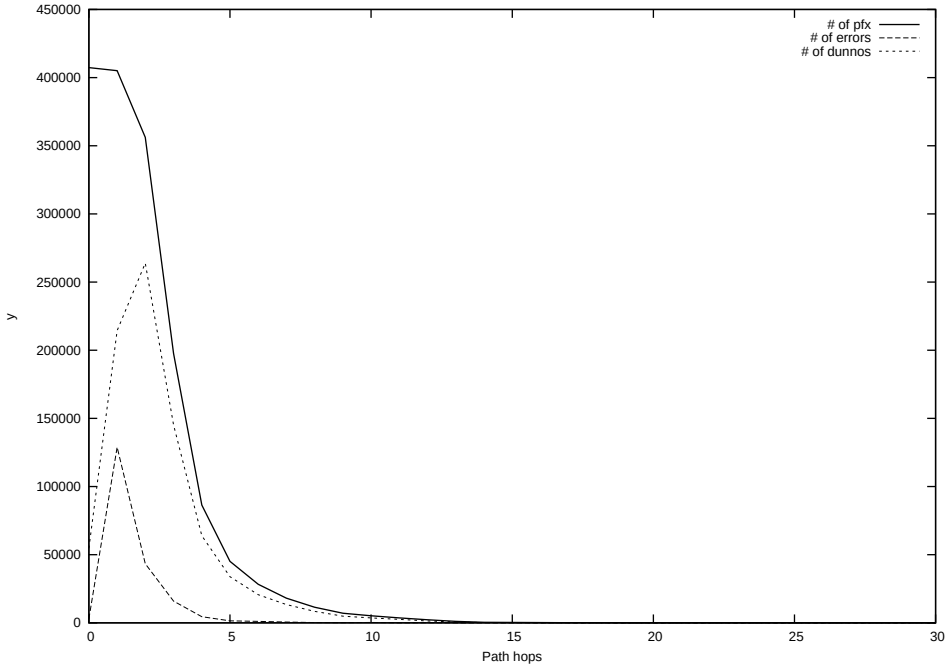
2012-05-22



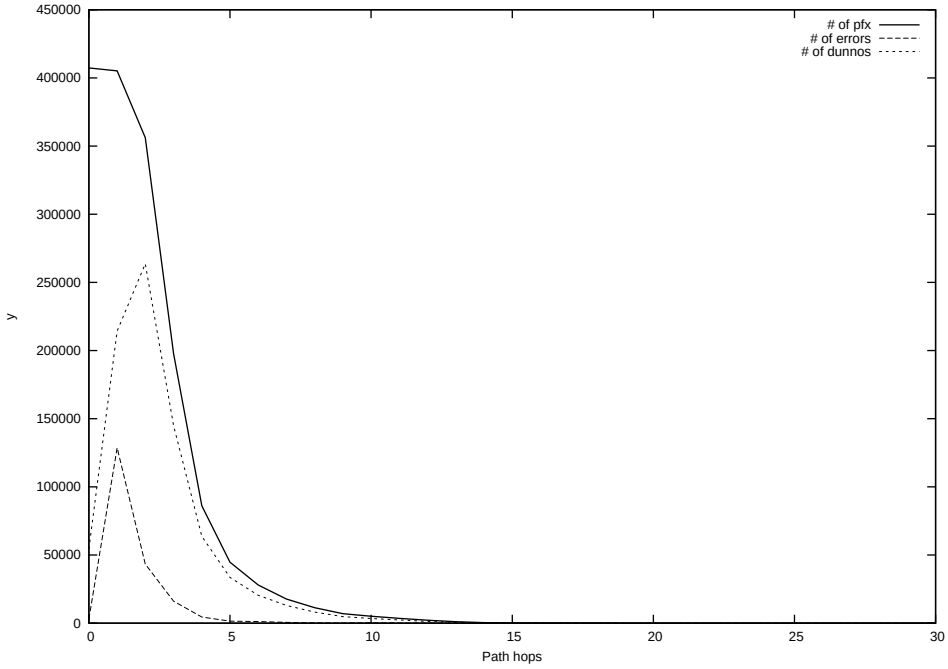
2012-05-23



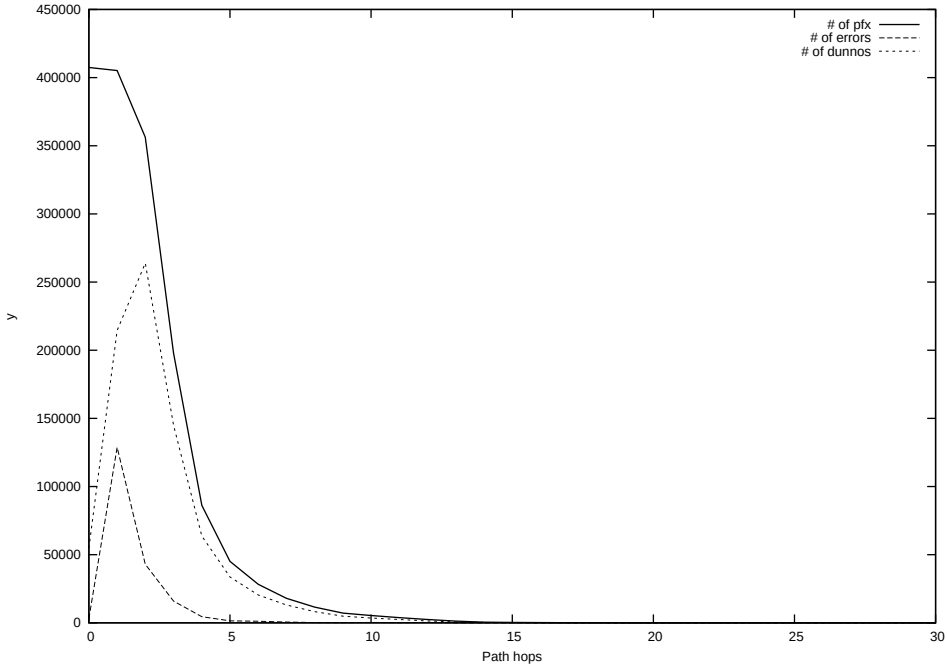
2012-05-24



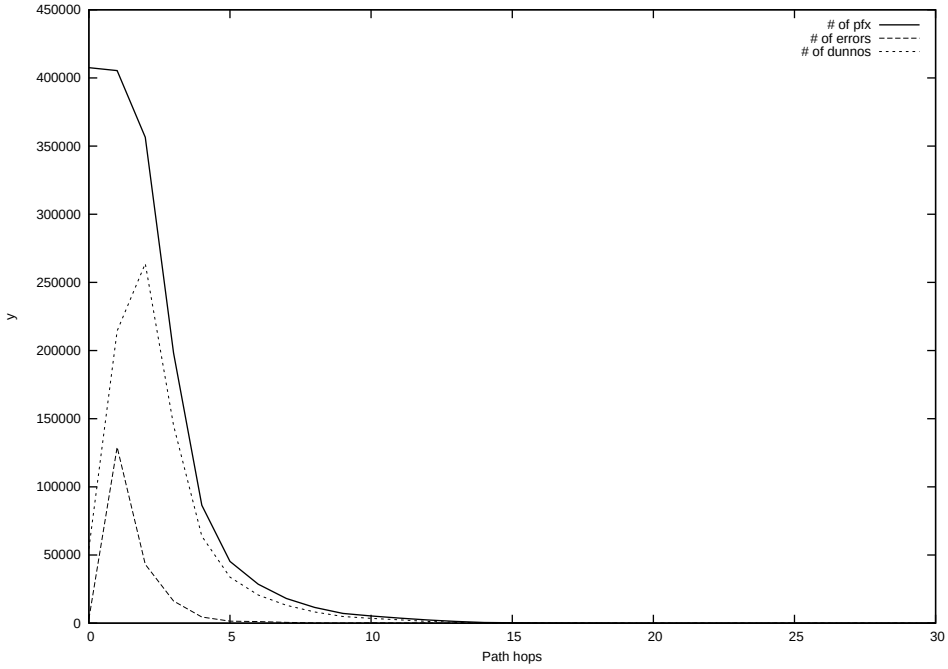
2012-05-25



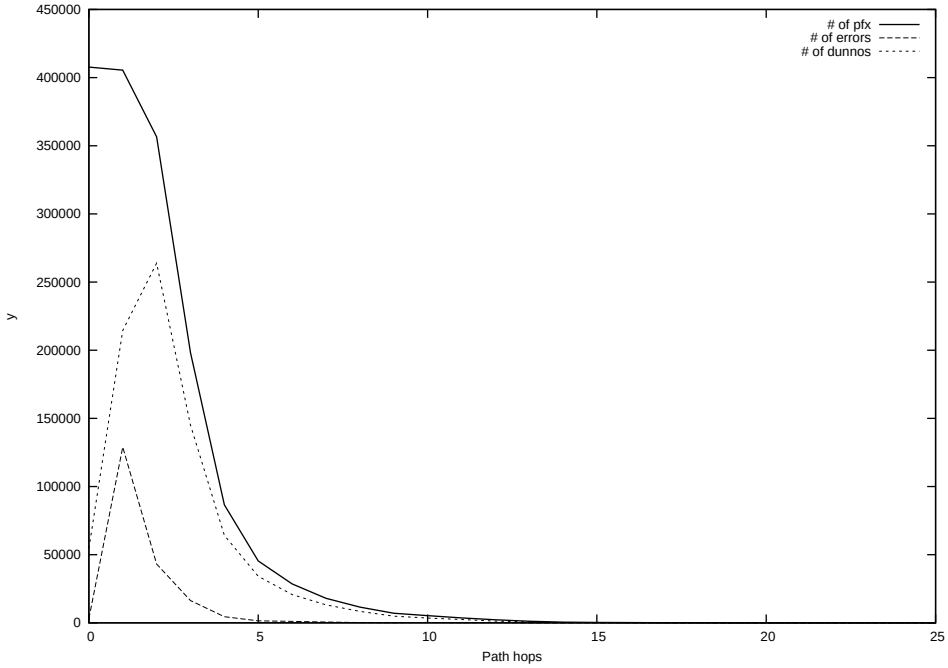
2012-05-26



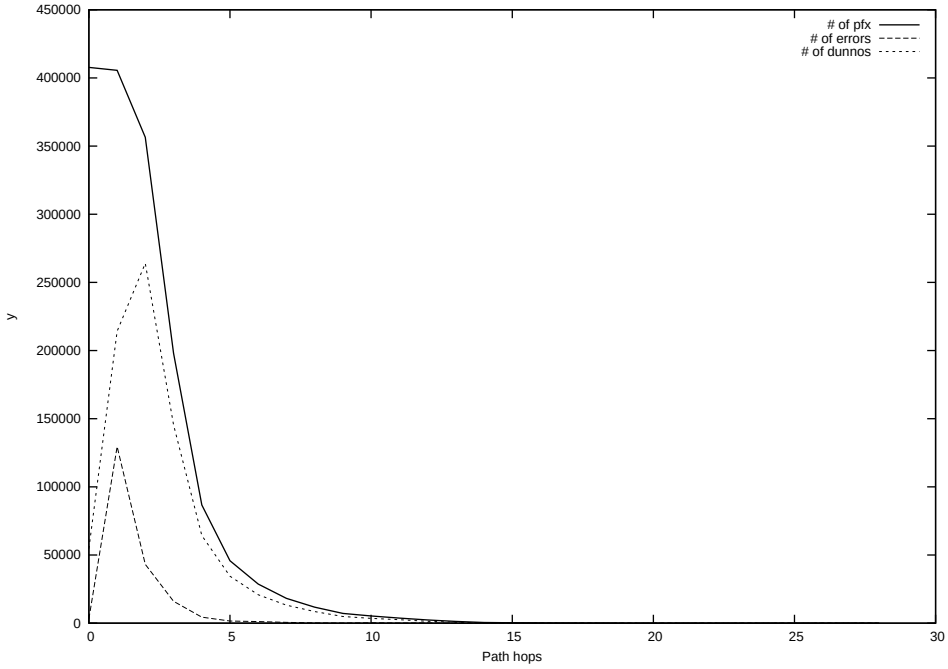
2012-05-27



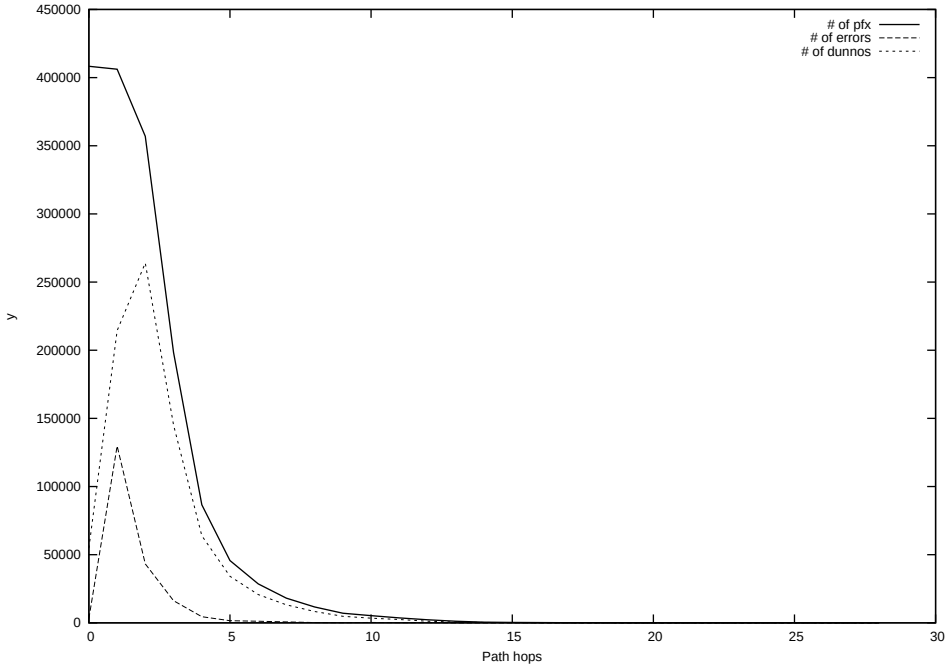
2012-05-28



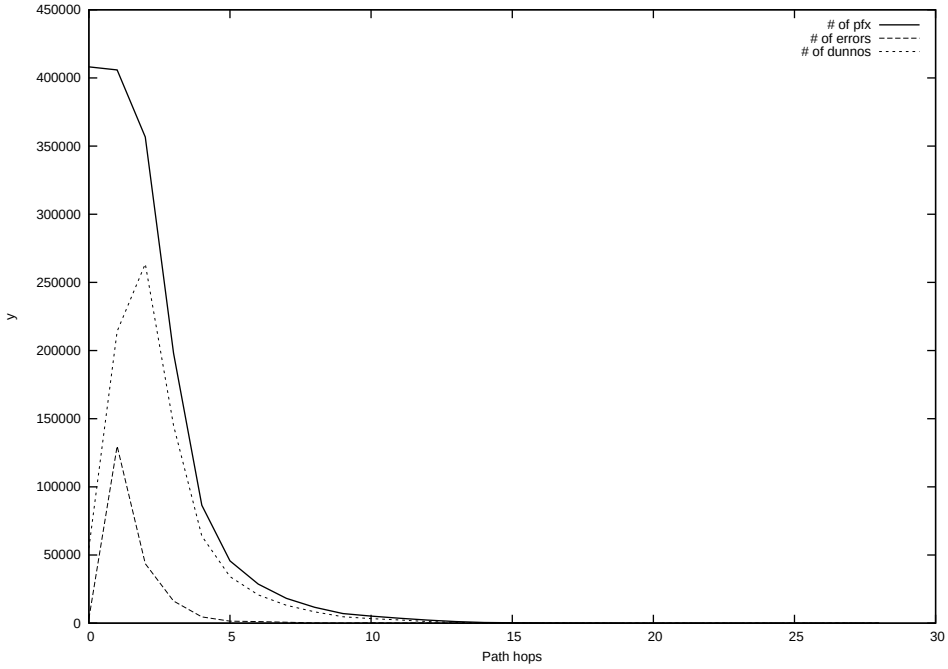
2012-05-29



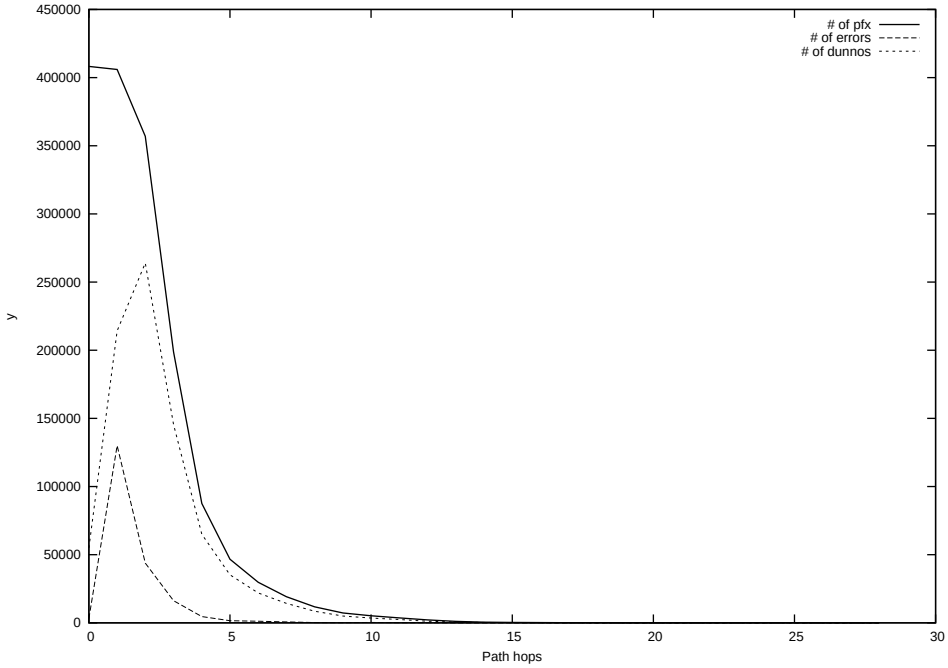
2012-05-30



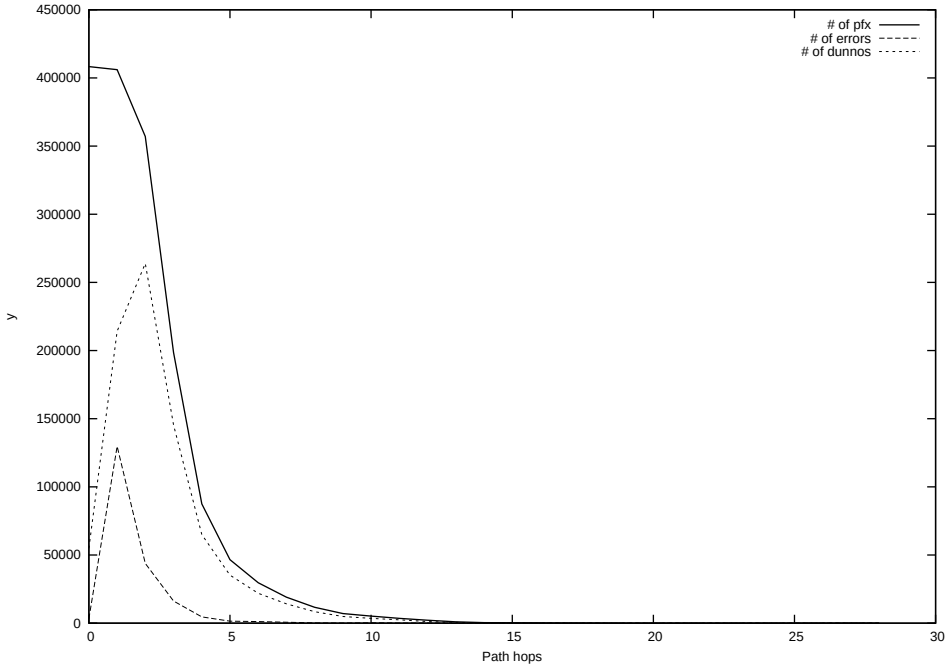
2012-05-31



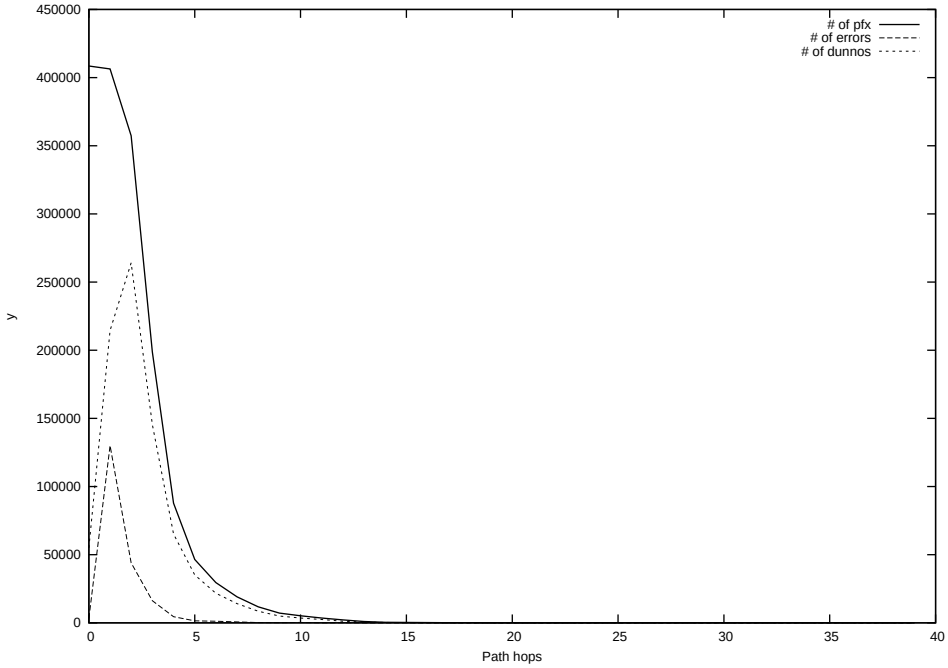
2012-06-01



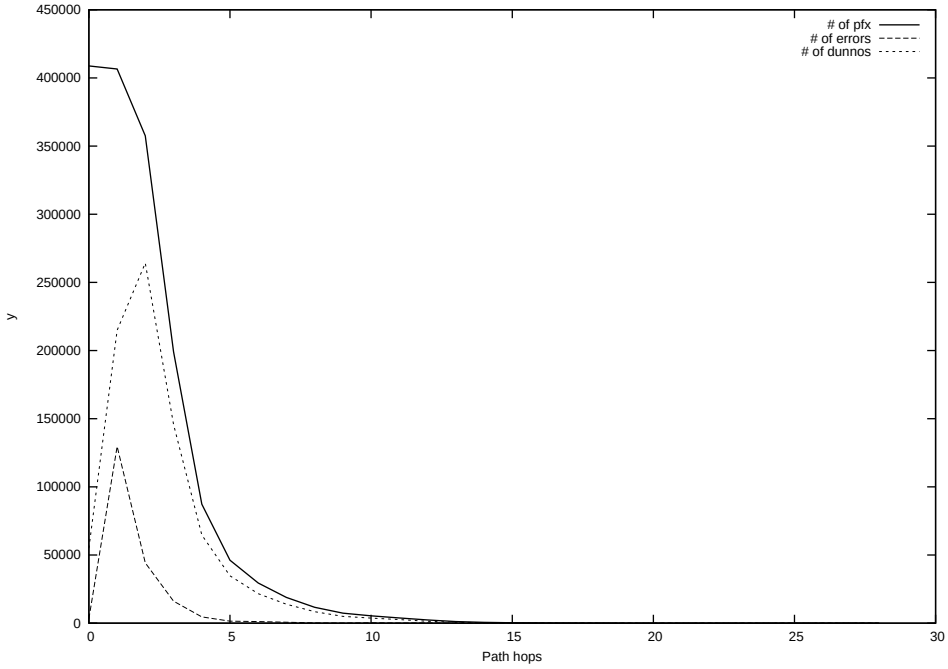
2012-06-02



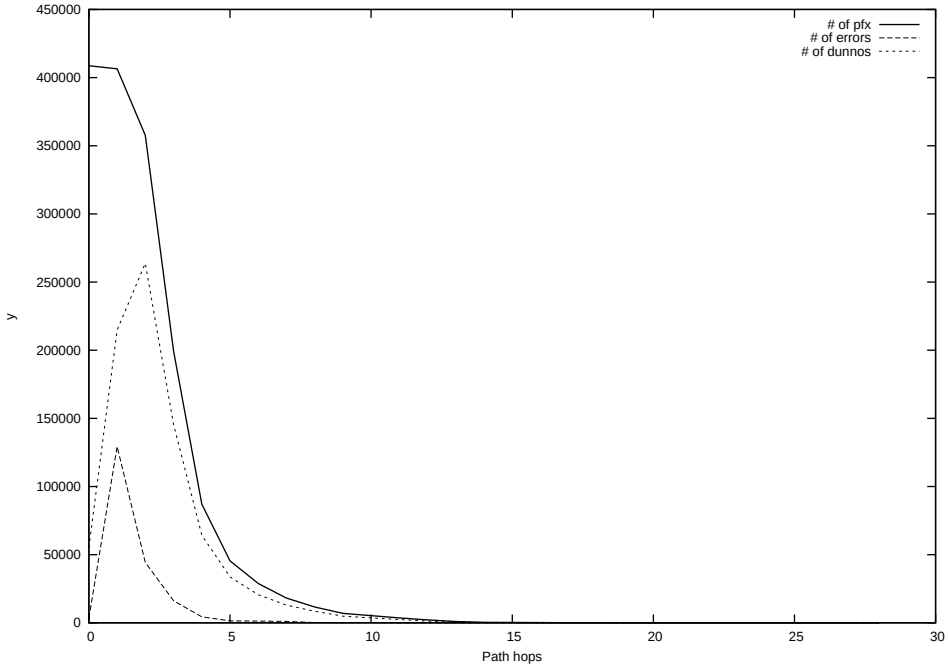
2012-06-03



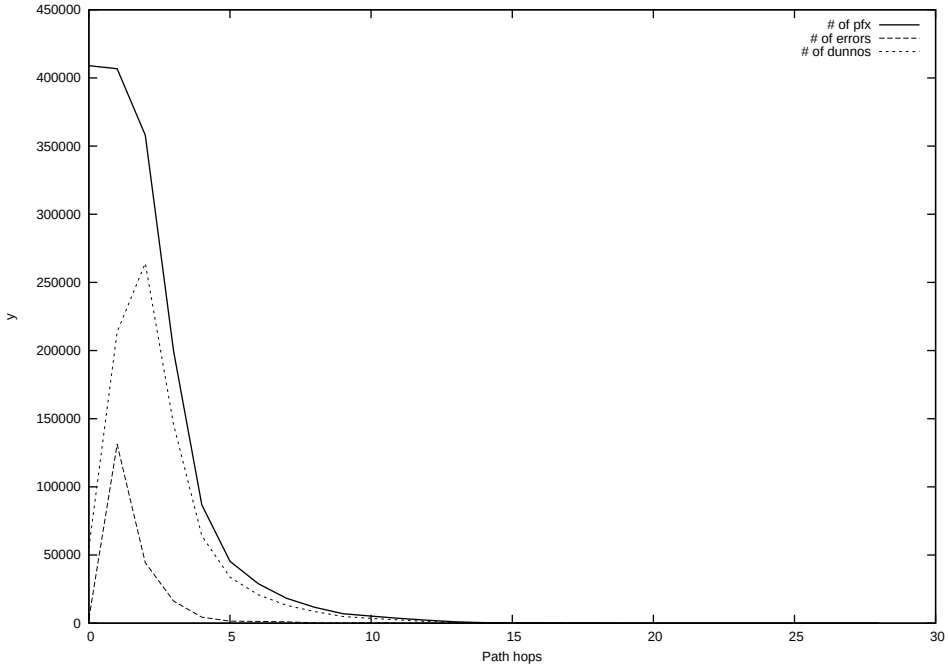
2012-06-04



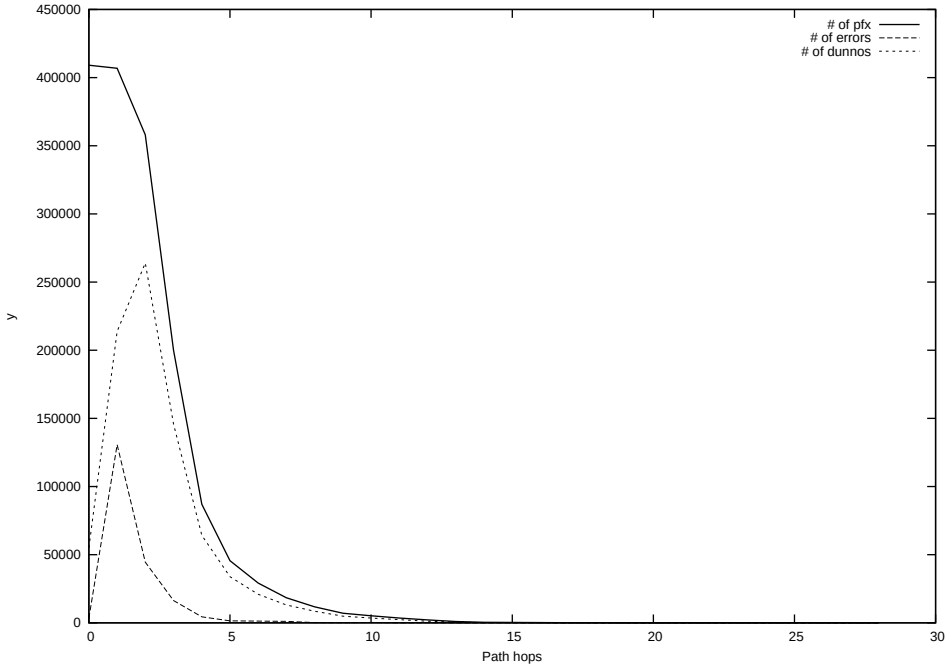
2012-06-05



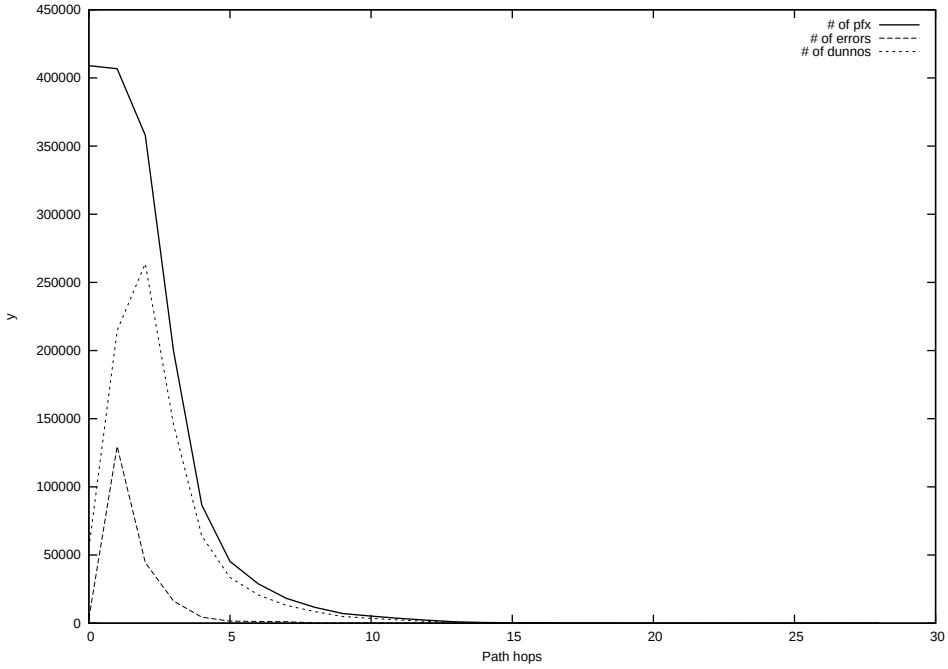
2012-06-06



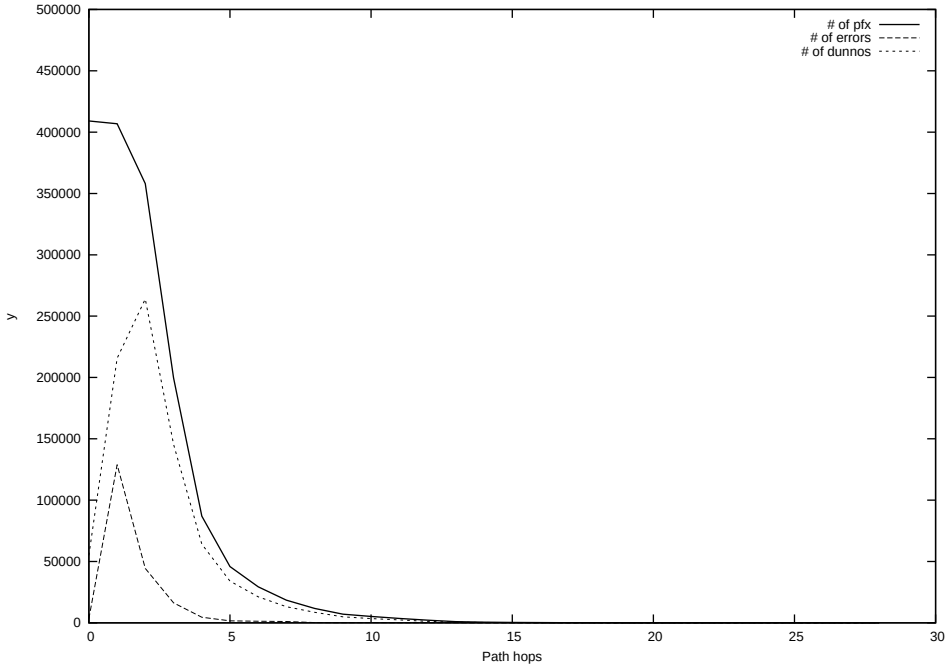
2012-06-07



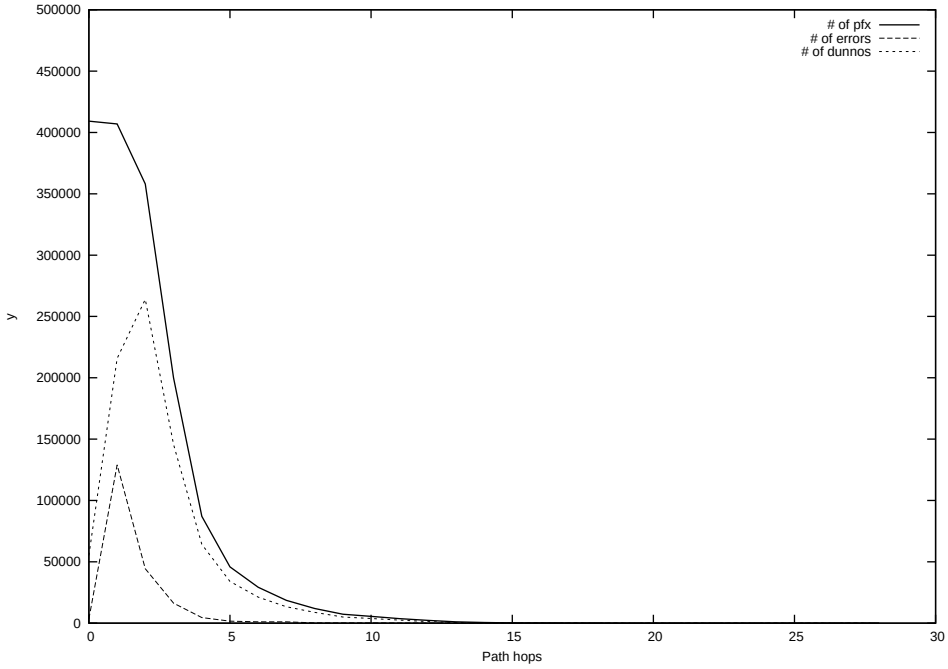
2012-06-08



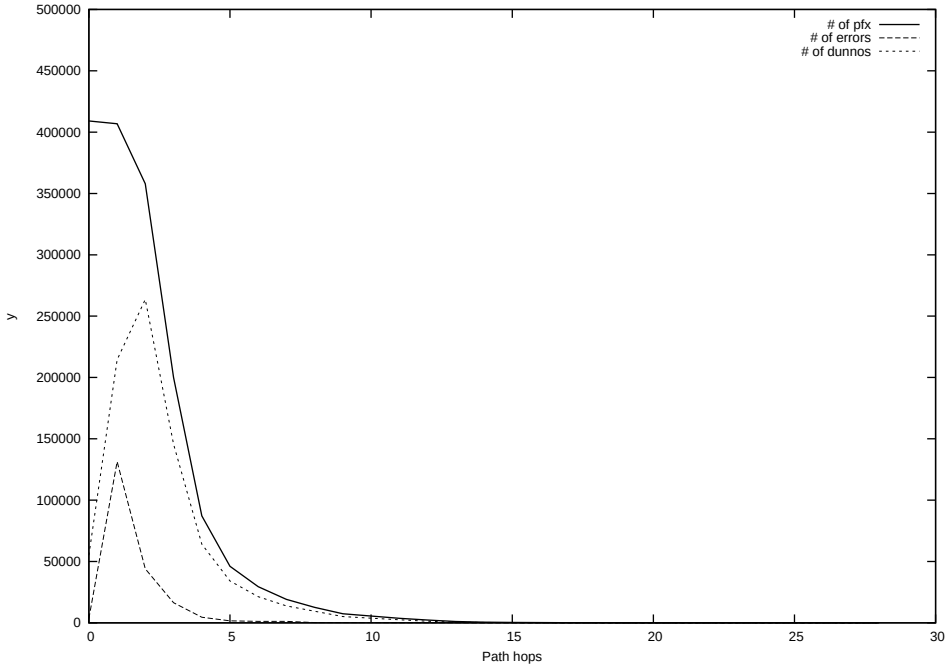
2012-06-10



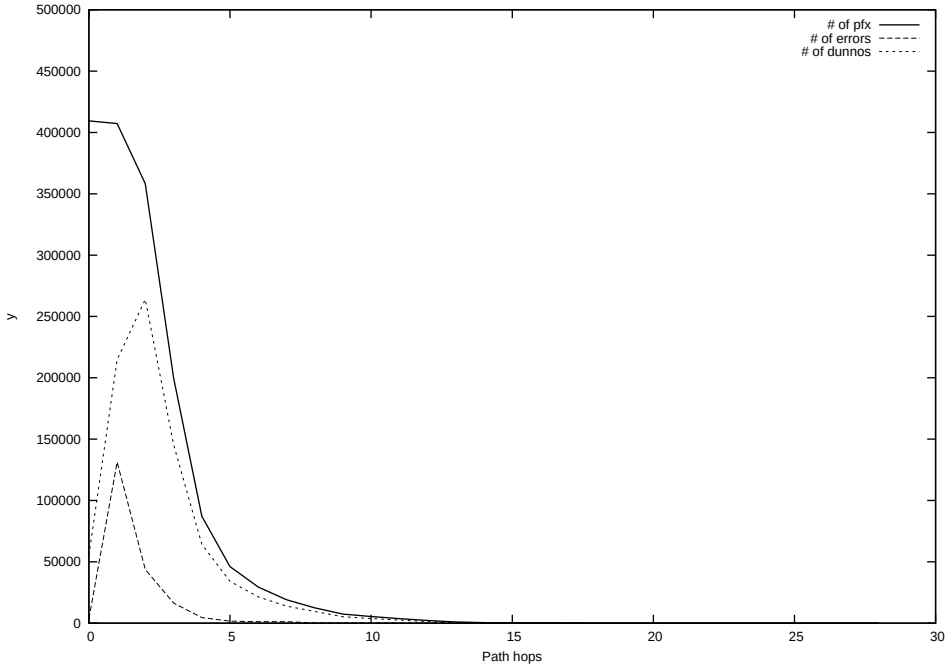
2012-06-11



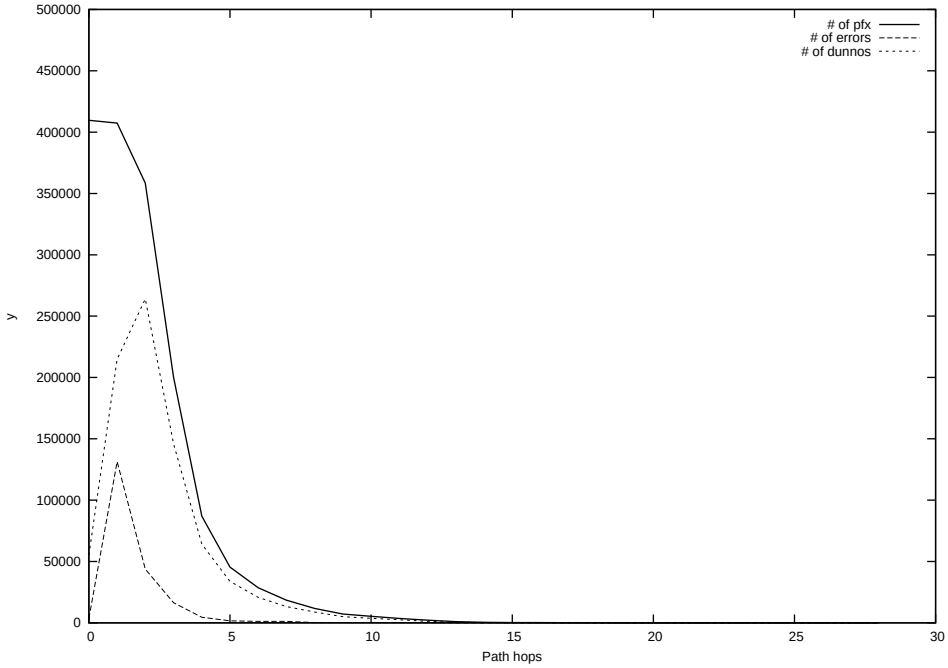
2012-06-12



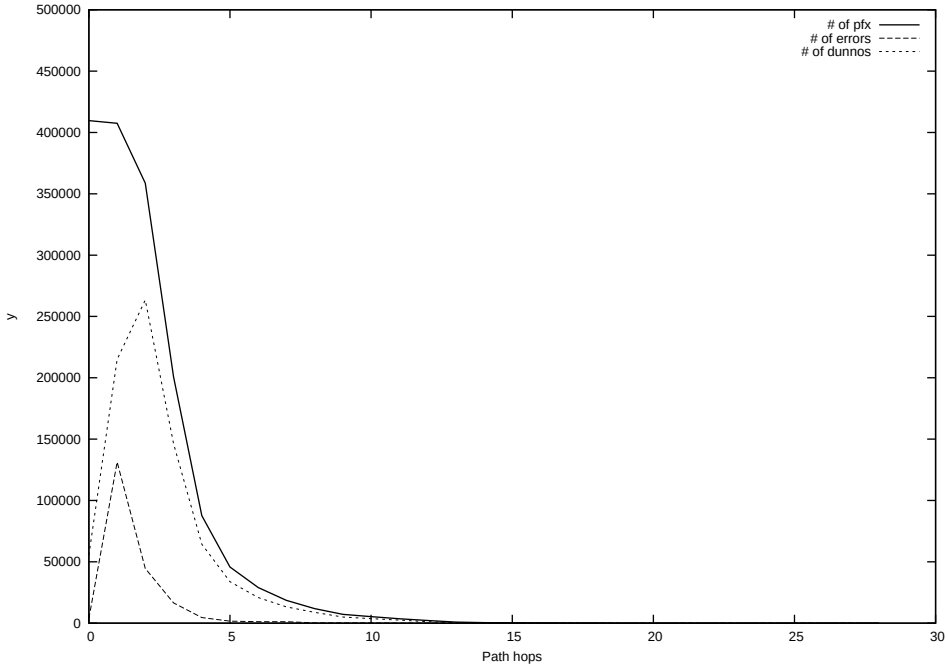
2012-06-13



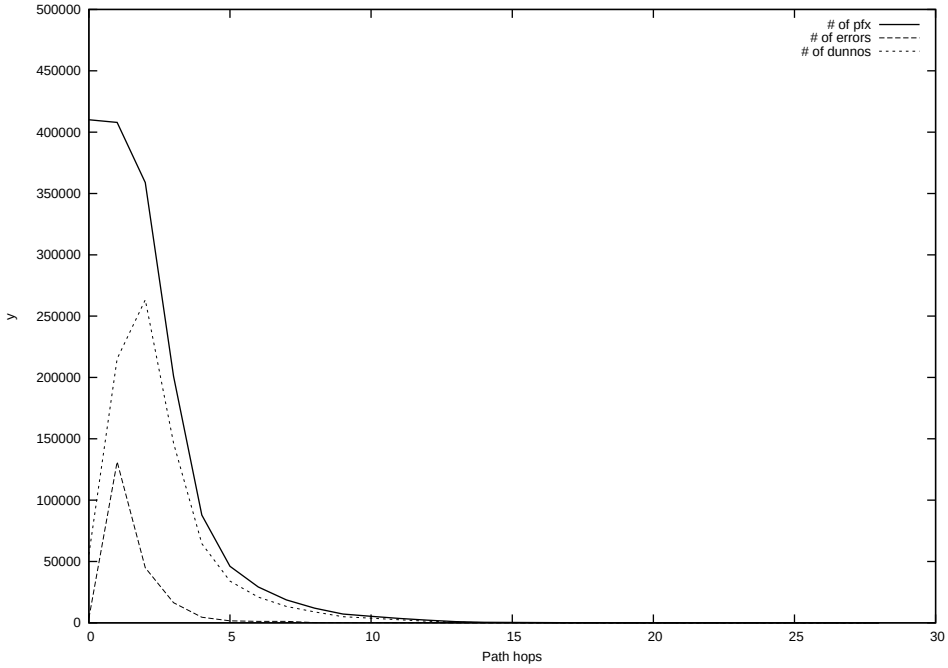
2012-06-14



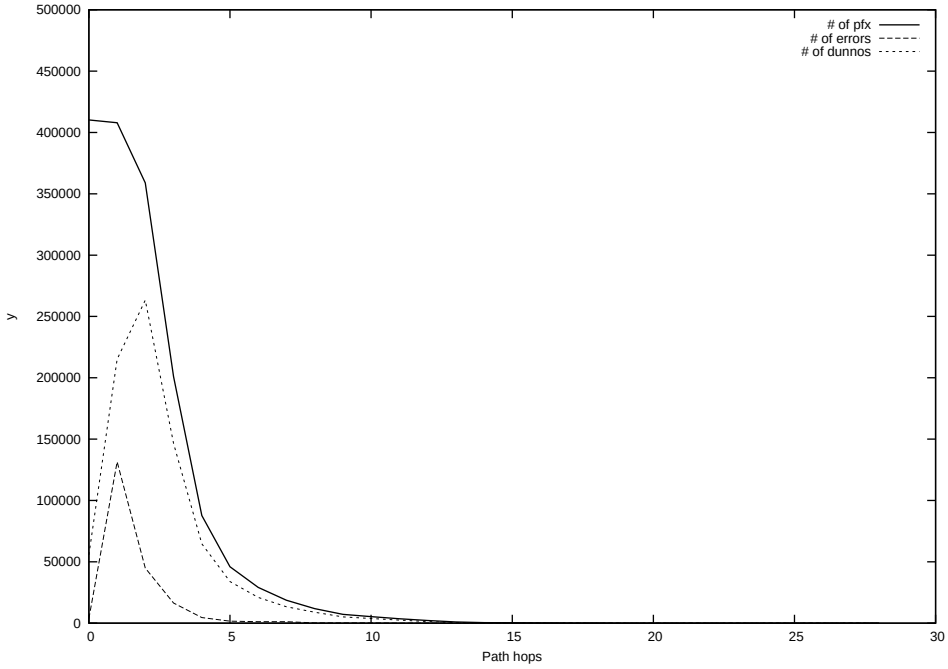
2012-06-15



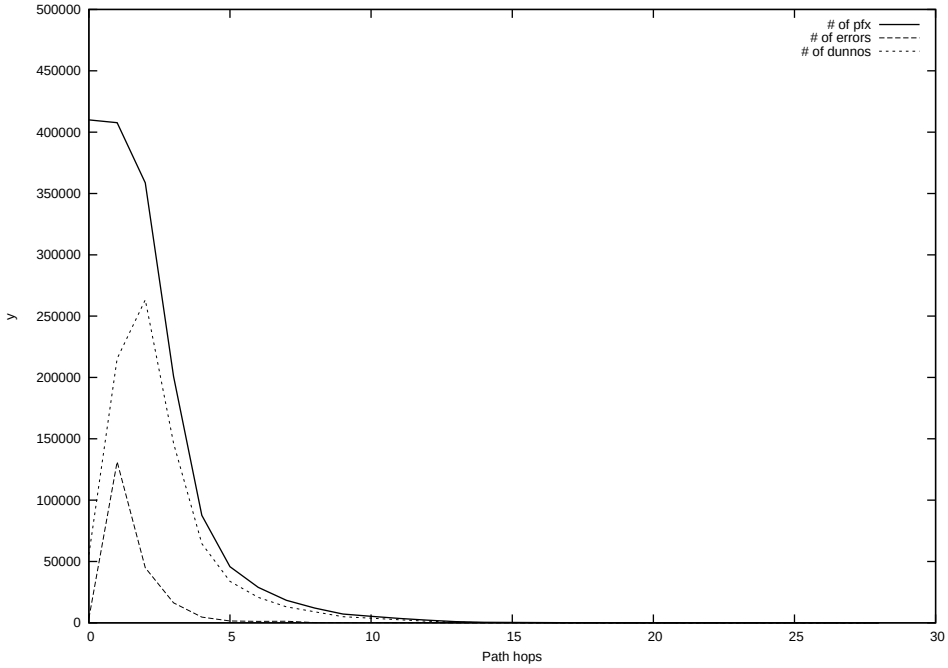
2012-06-16



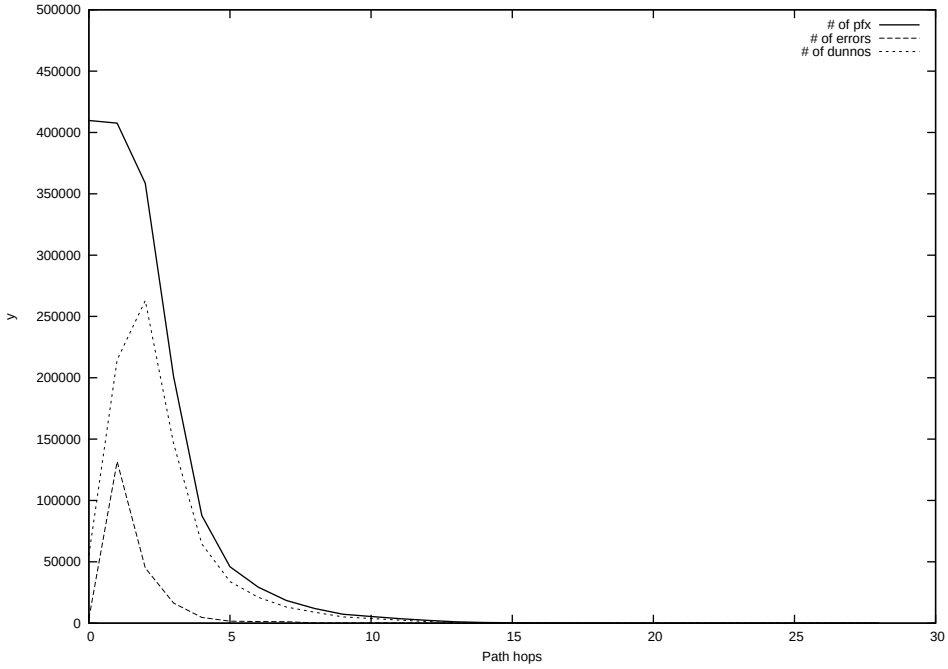
2012-06-17



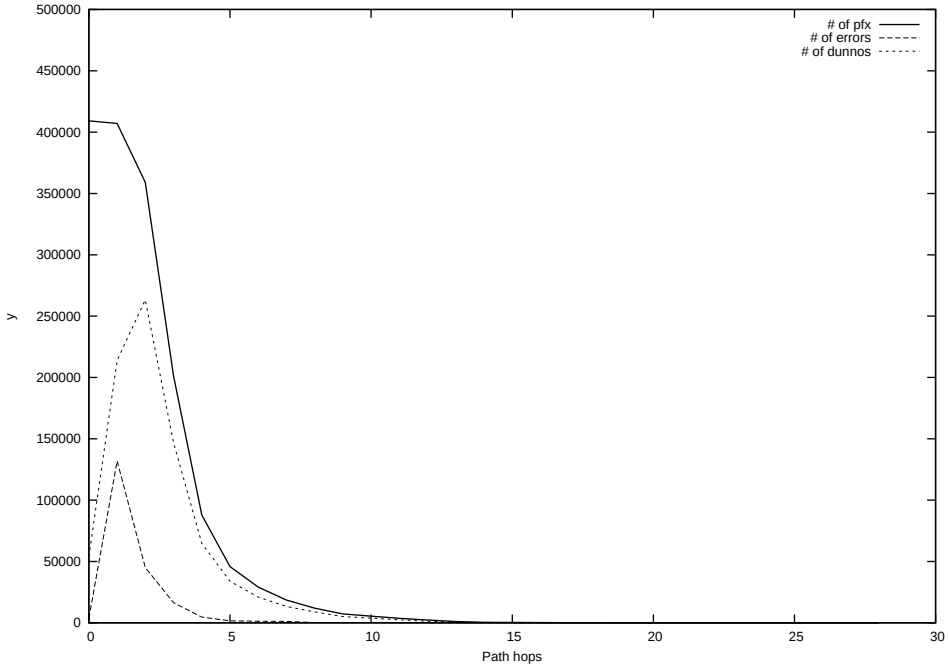
2012-06-18



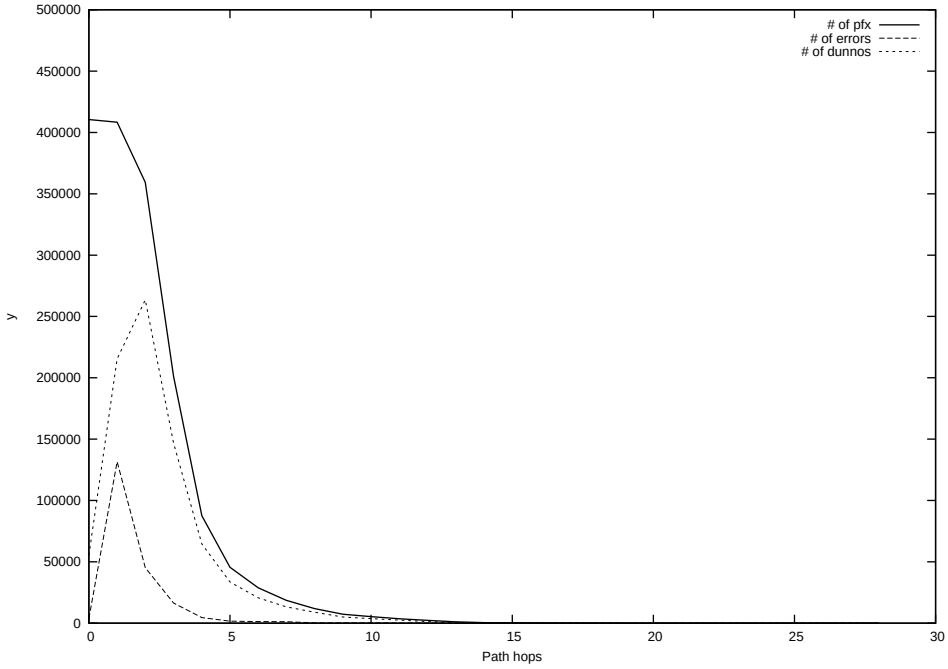
2012-06-19



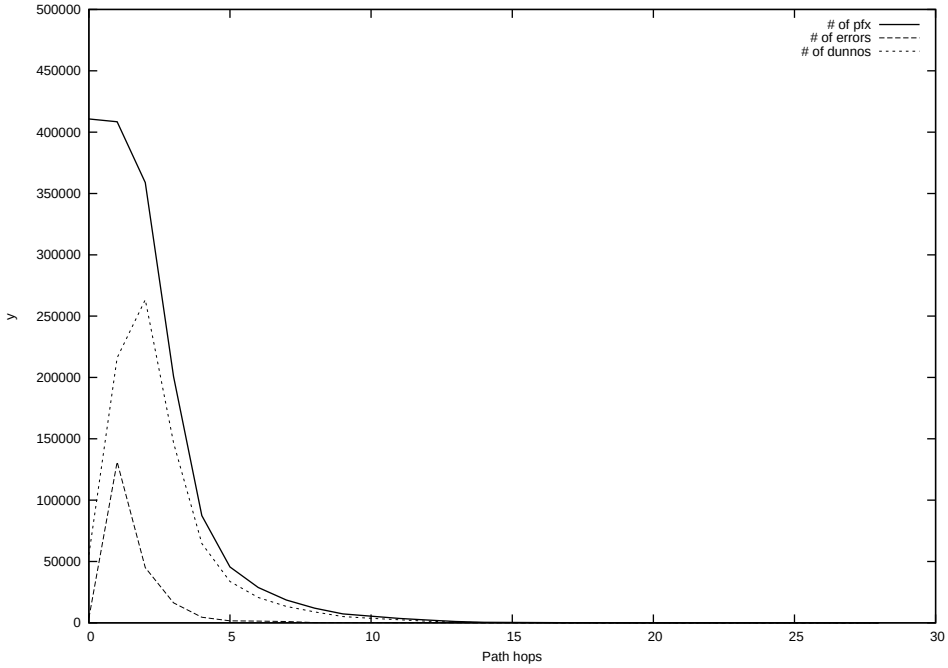
2012-06-20



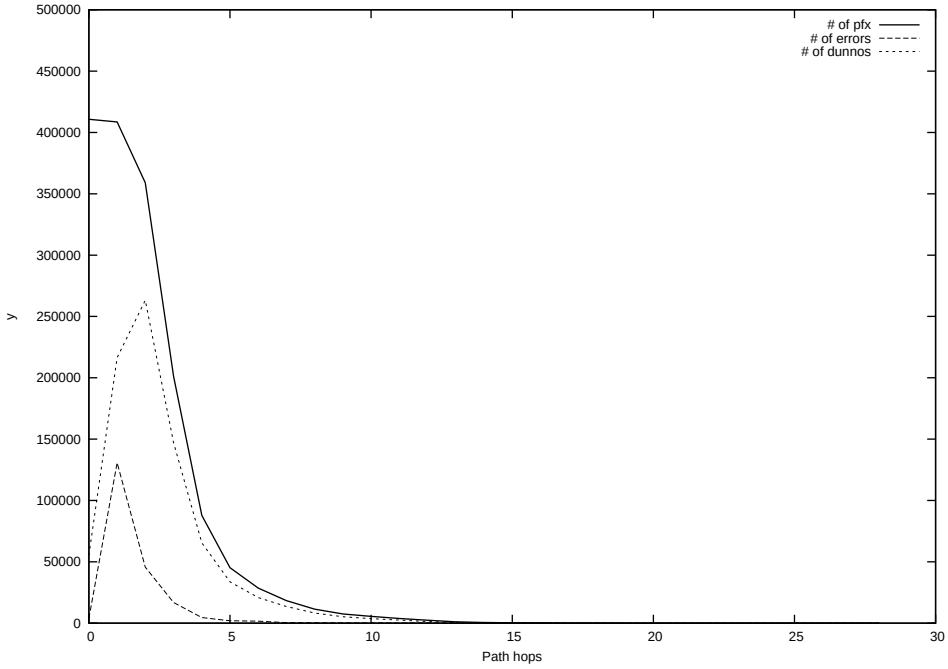
2012-06-21



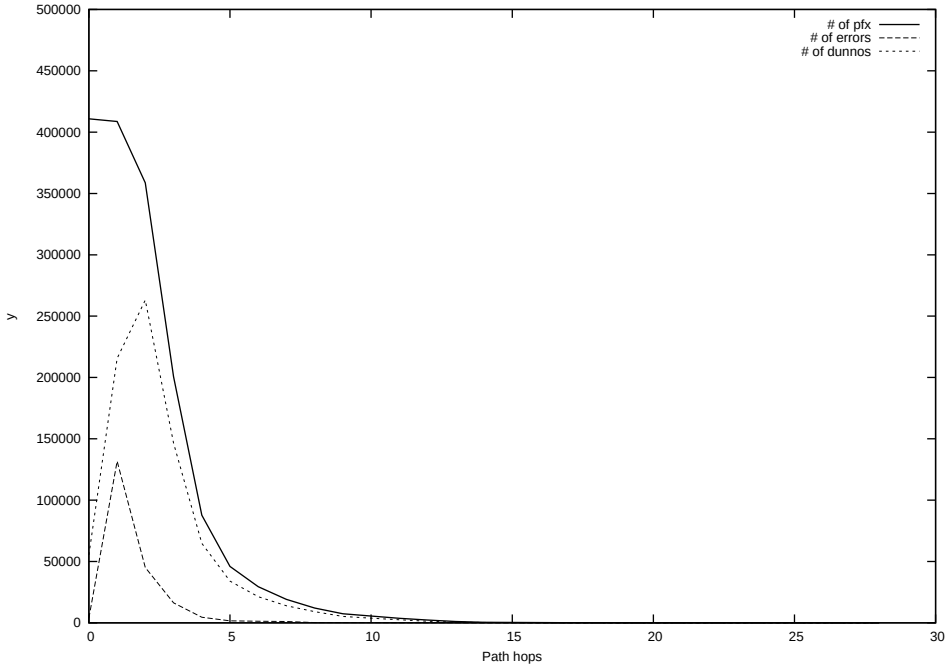
2012-06-22



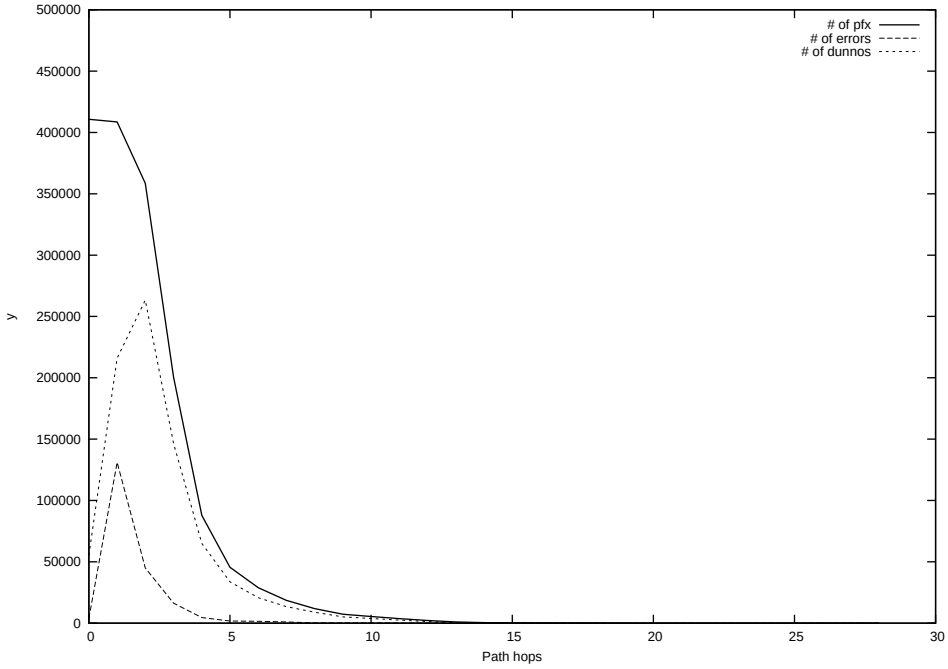
2012-06-23



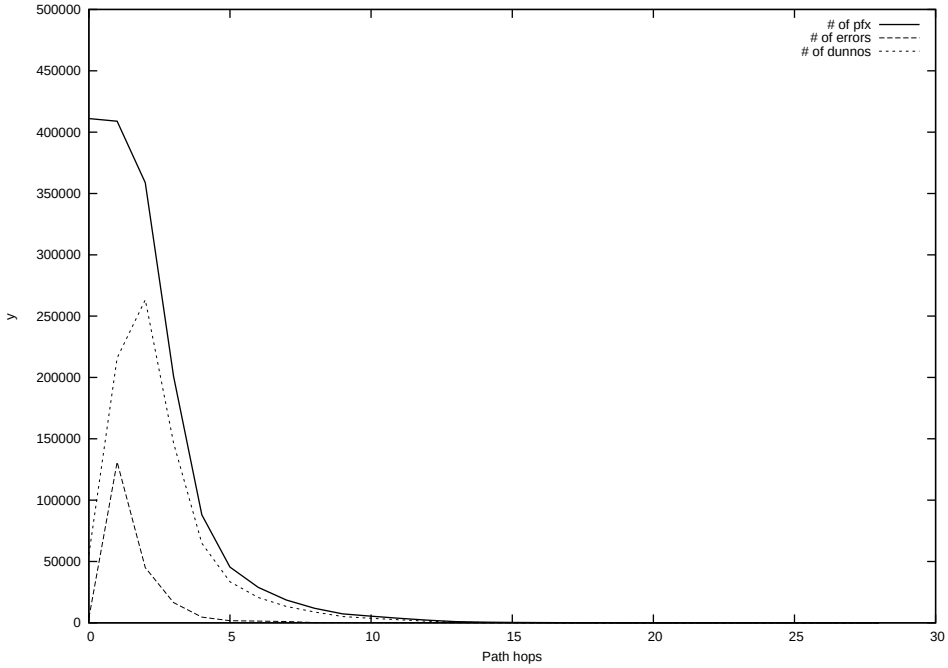
2012-06-24



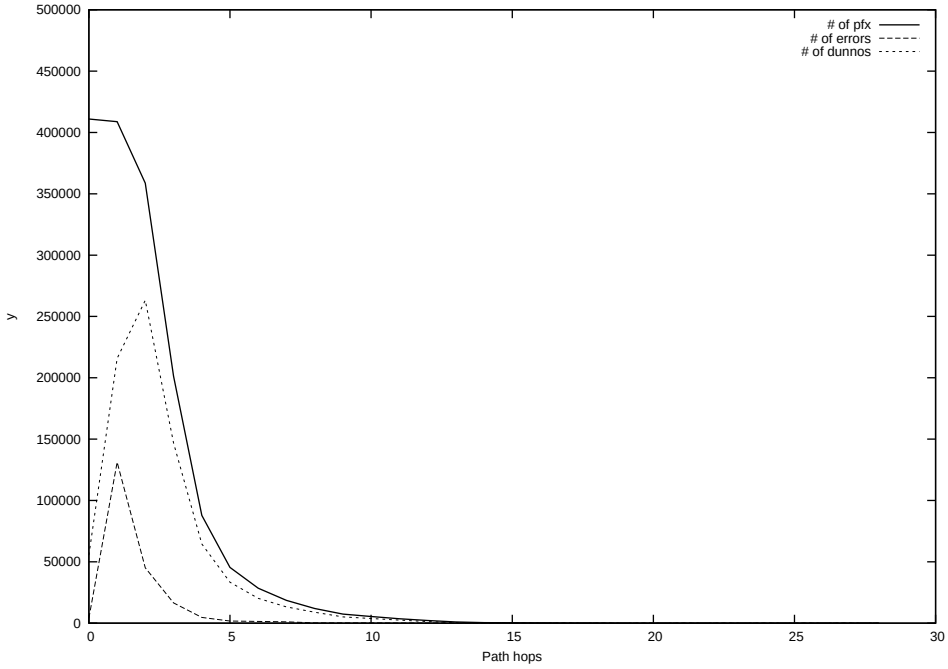
2012-06-25



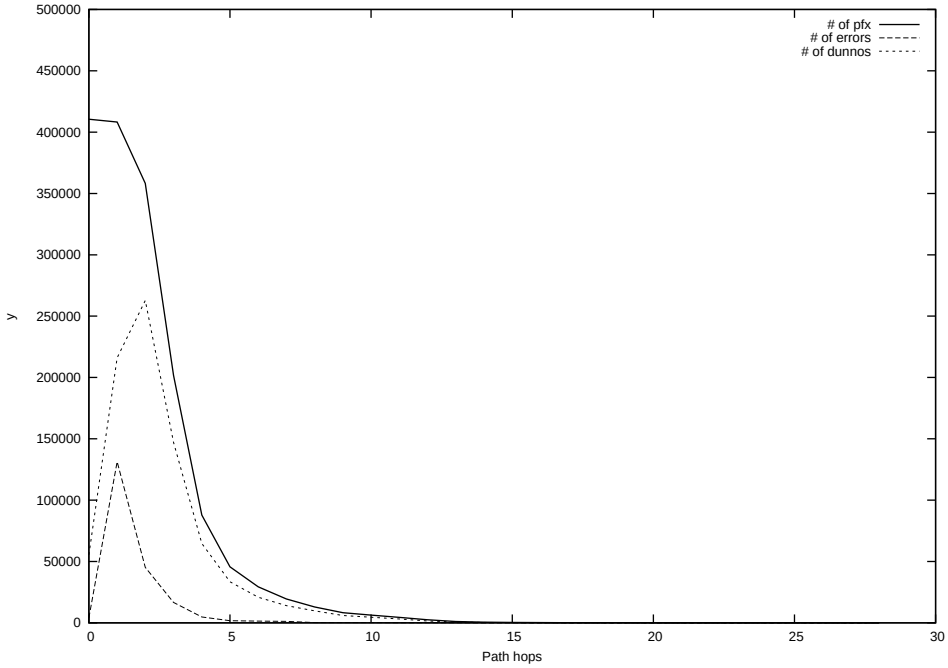
2012-06-26



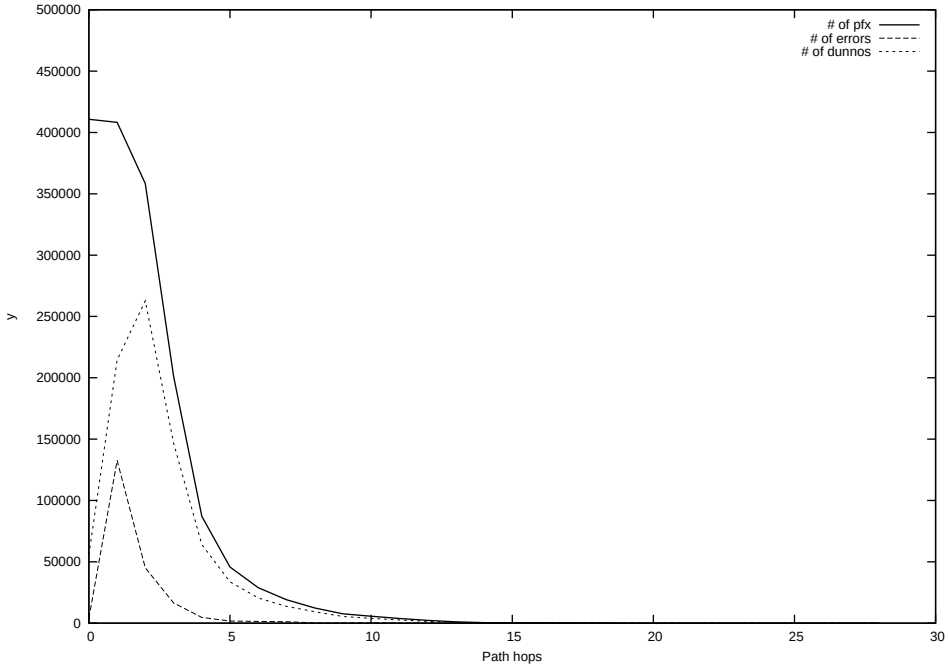
2012-06-27



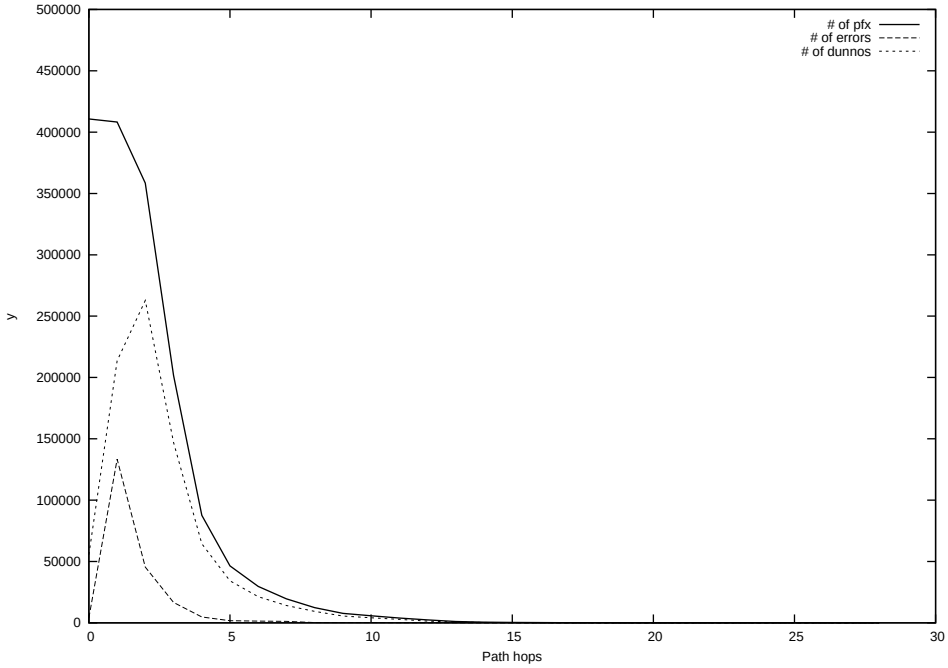
2012-06-28



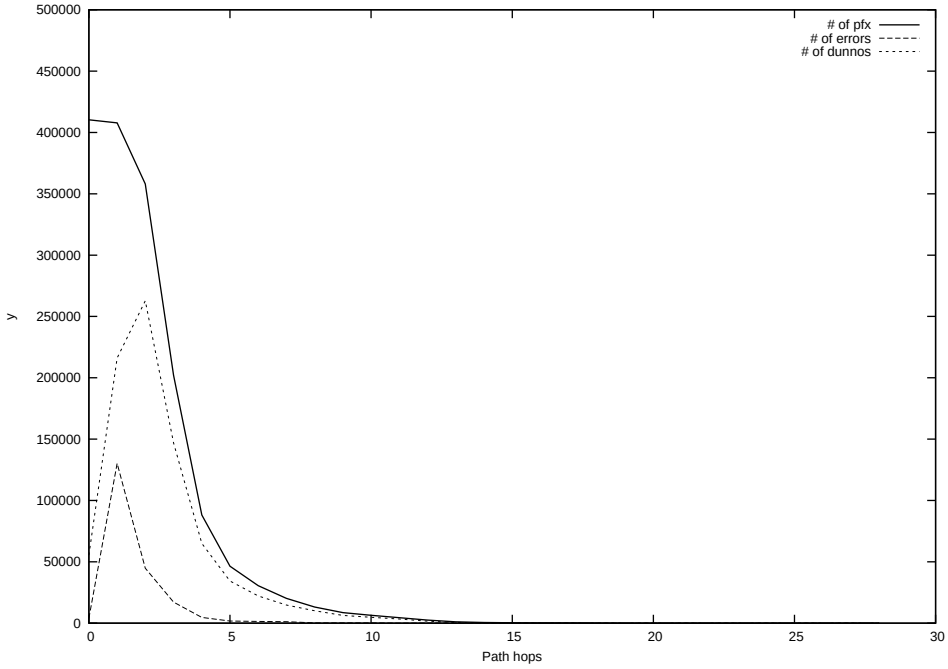
2012-06-29



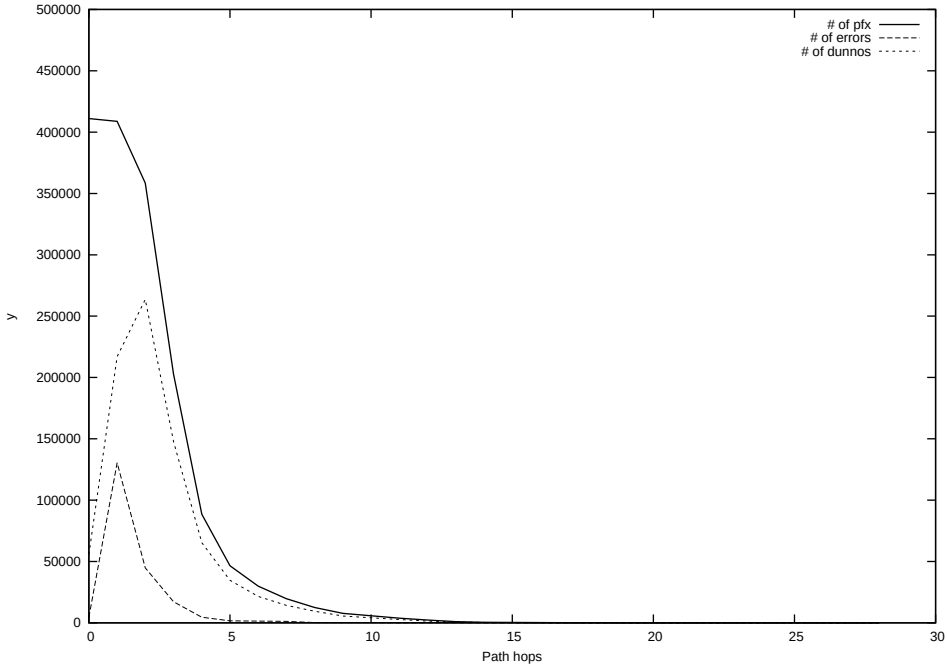
2012-06-30



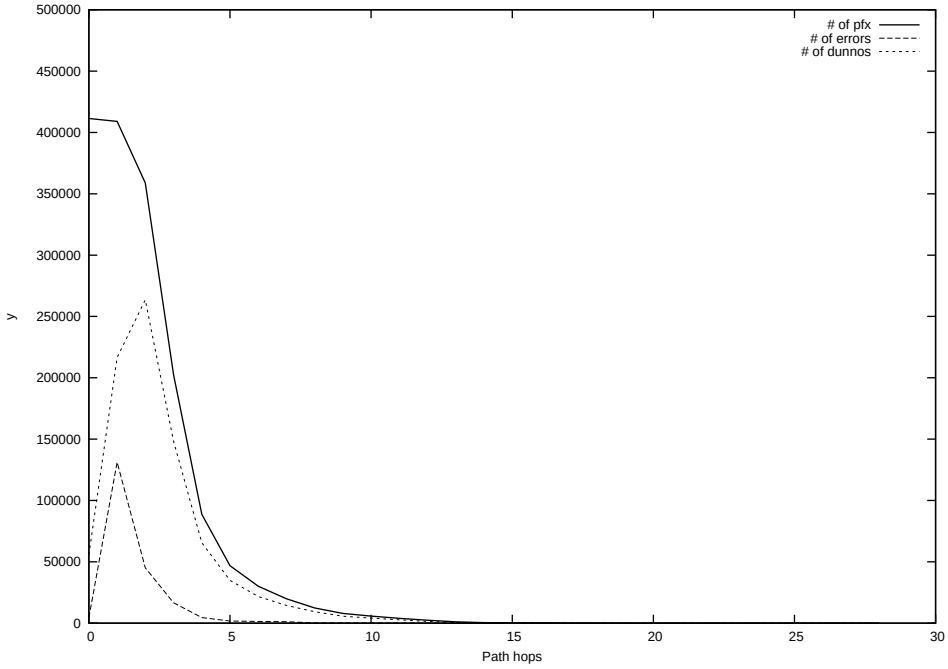
2012-07-01



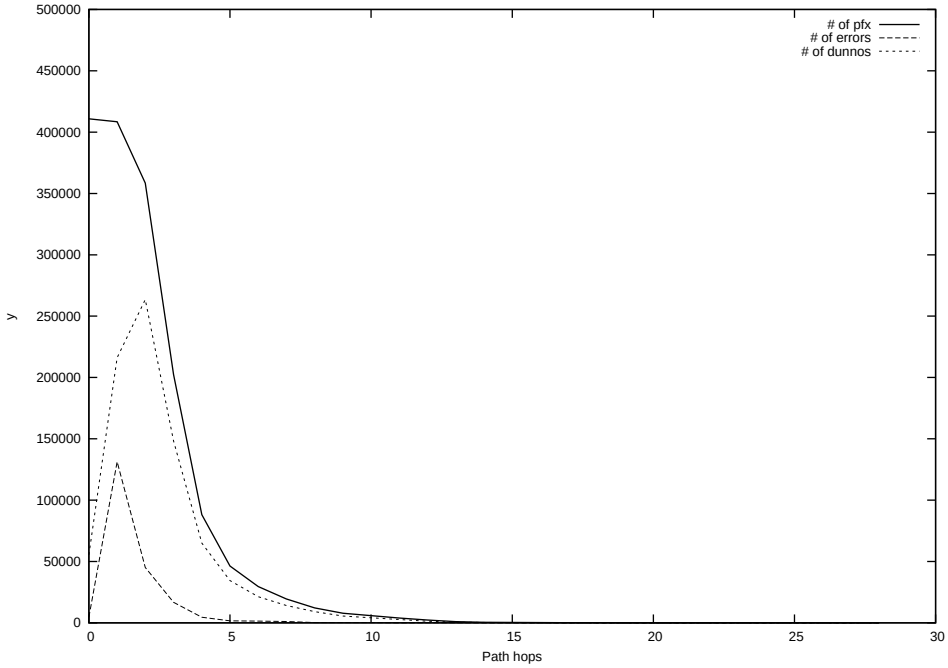
2012-07-02



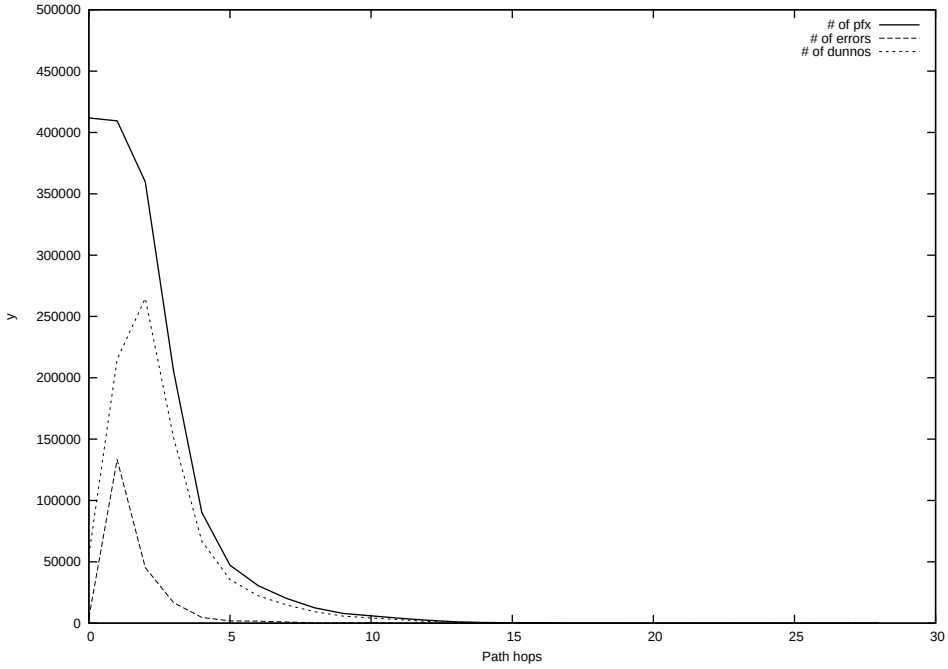
2012-07-03



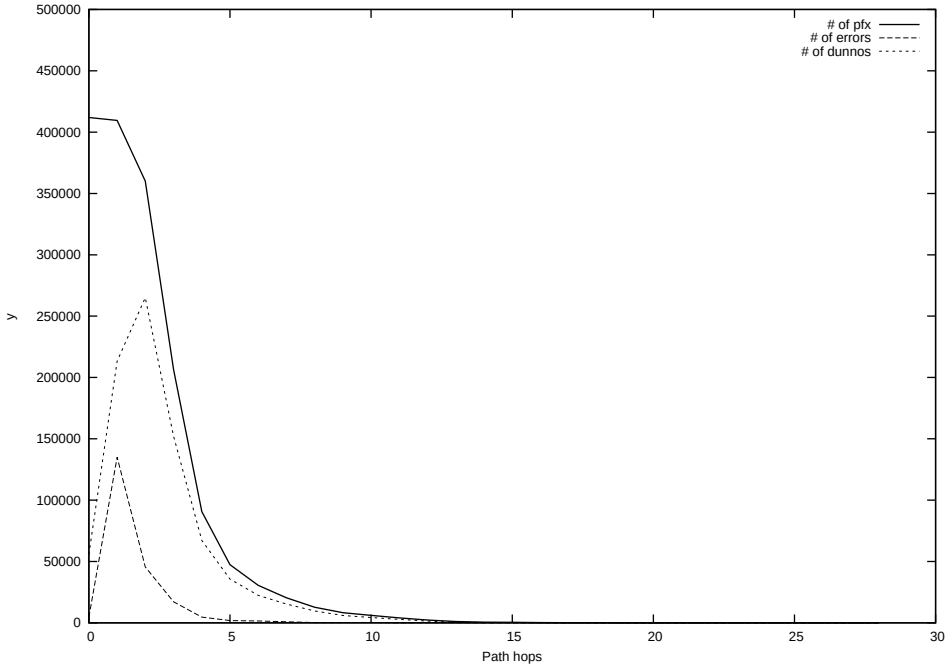
2012-07-04



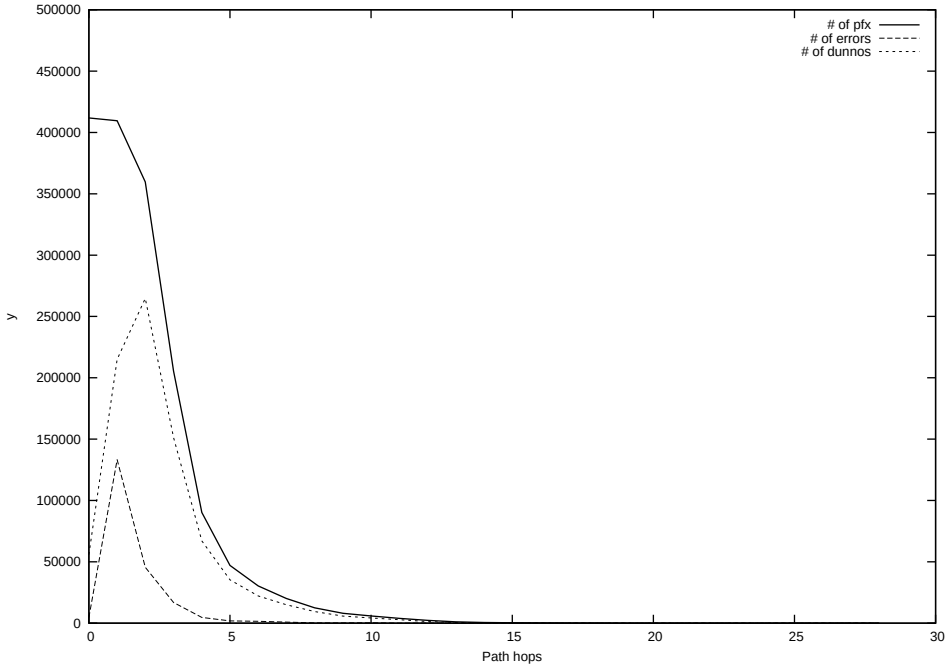
2012-07-05



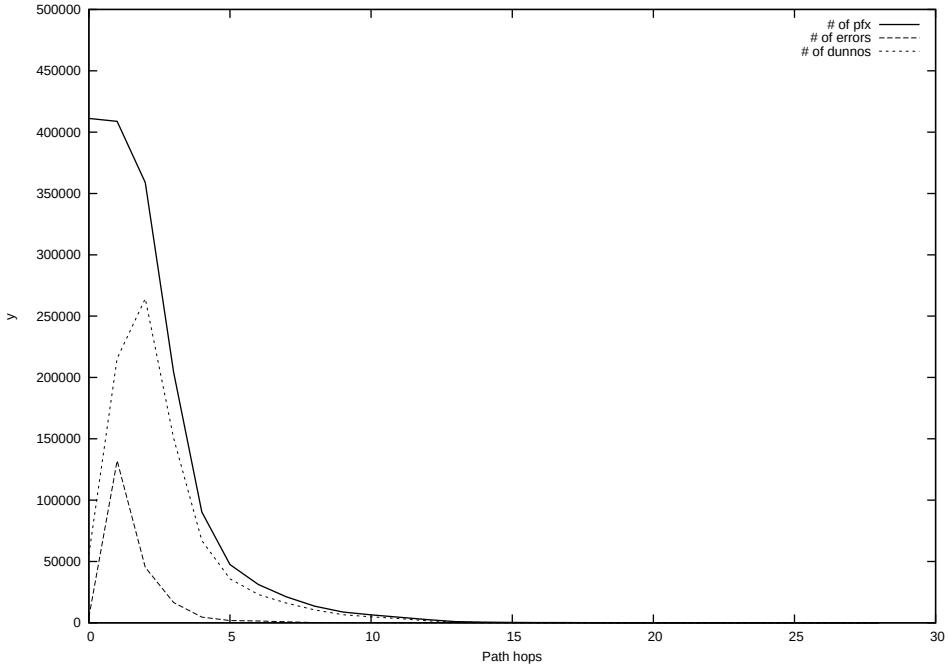
2012-07-06



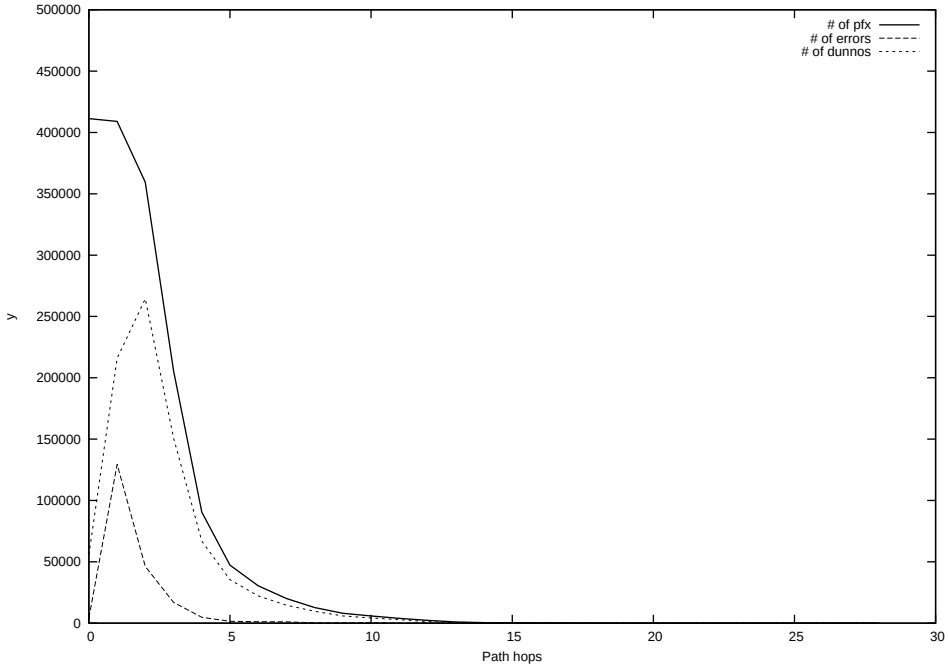
2012-07-07



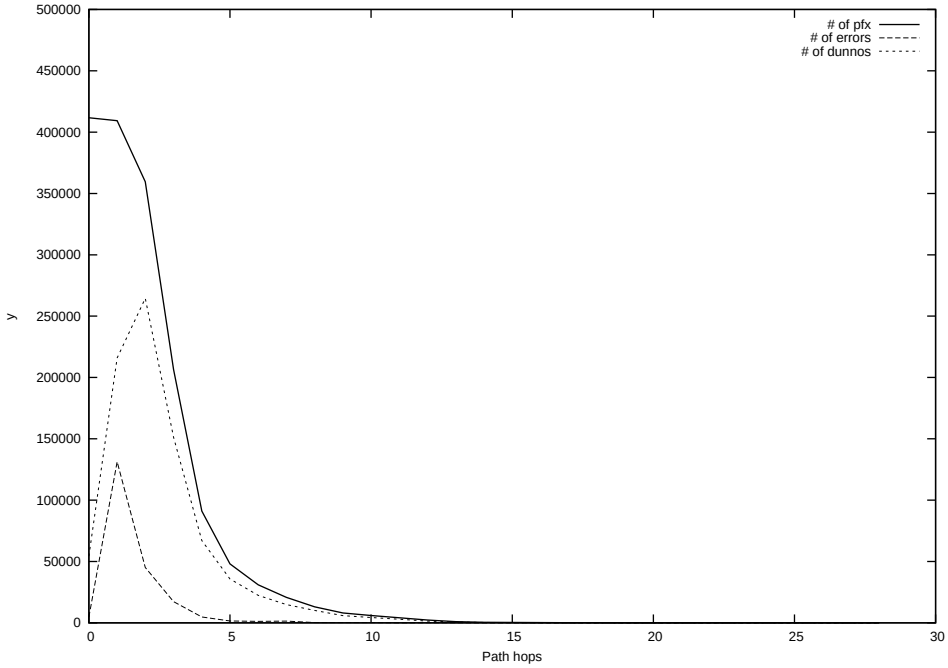
2012-07-08



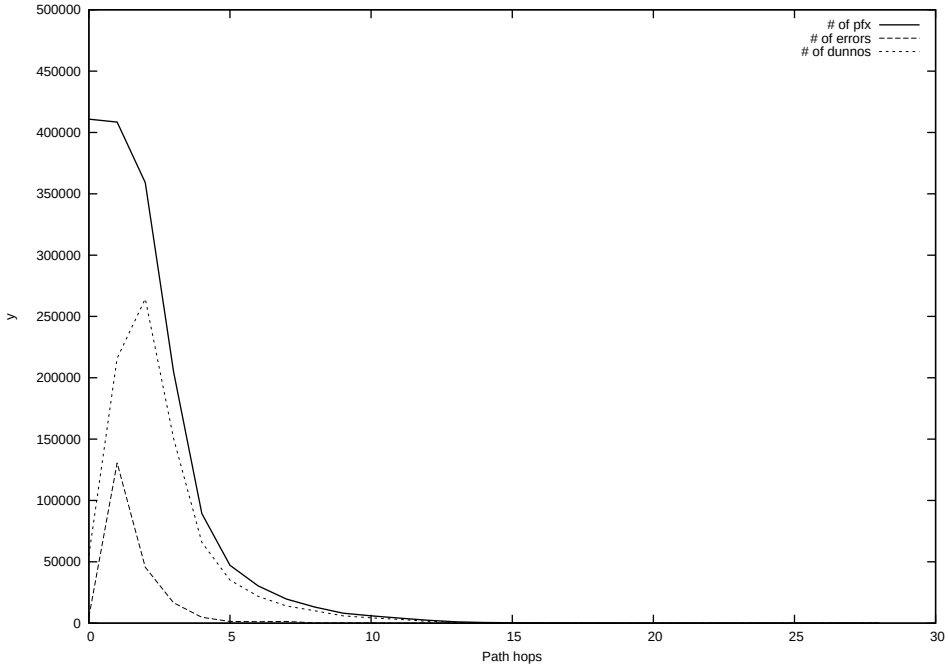
2012-07-09



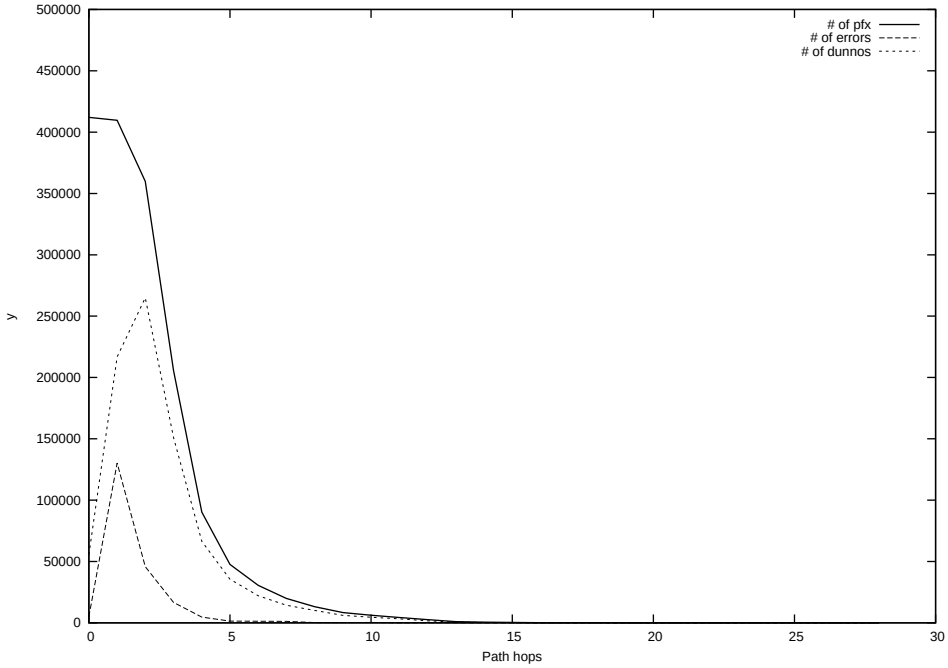
2012-07-10



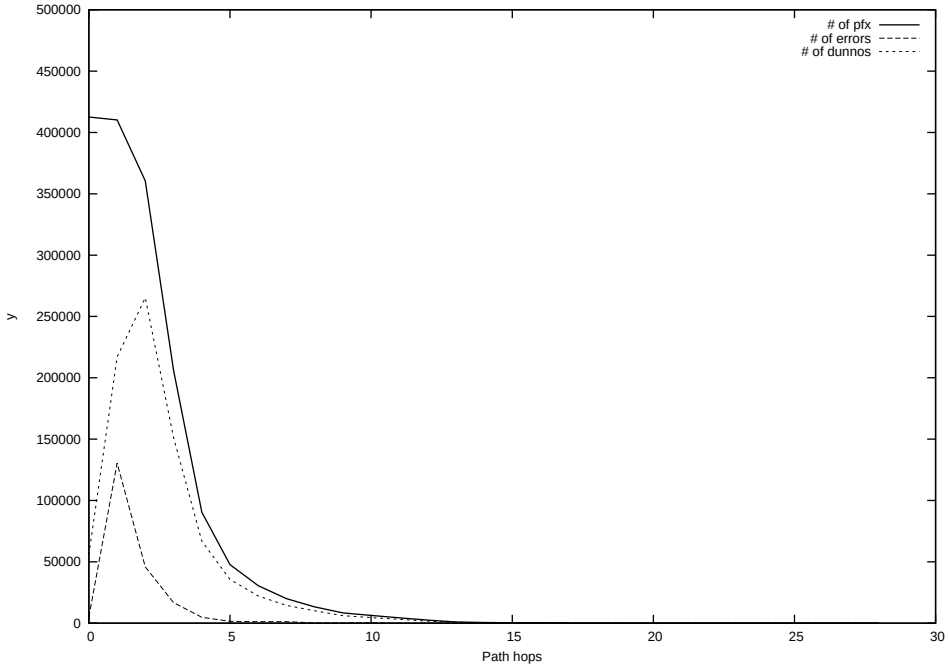
2012-07-11



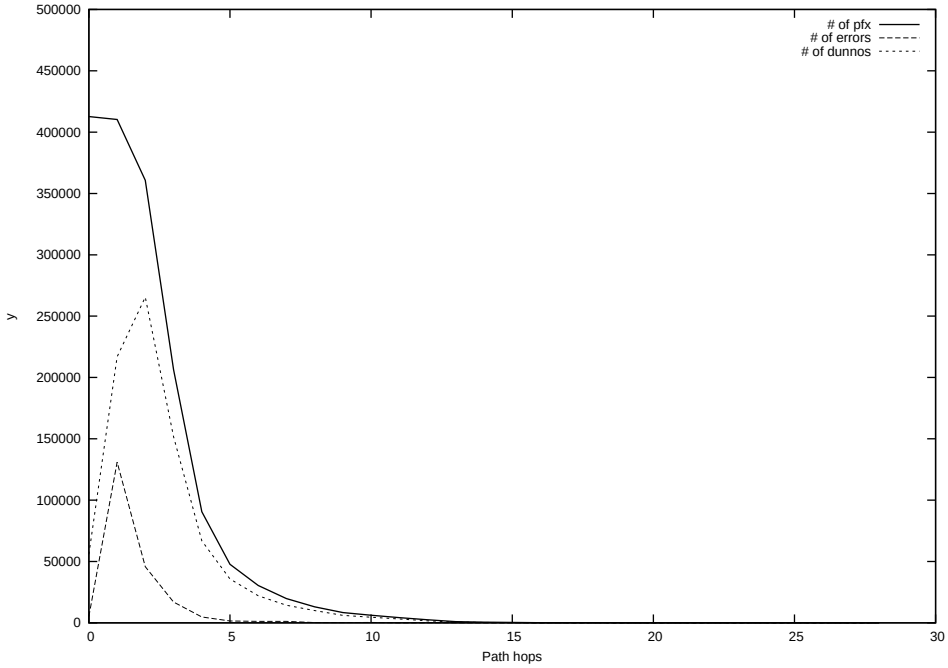
2012-07-12



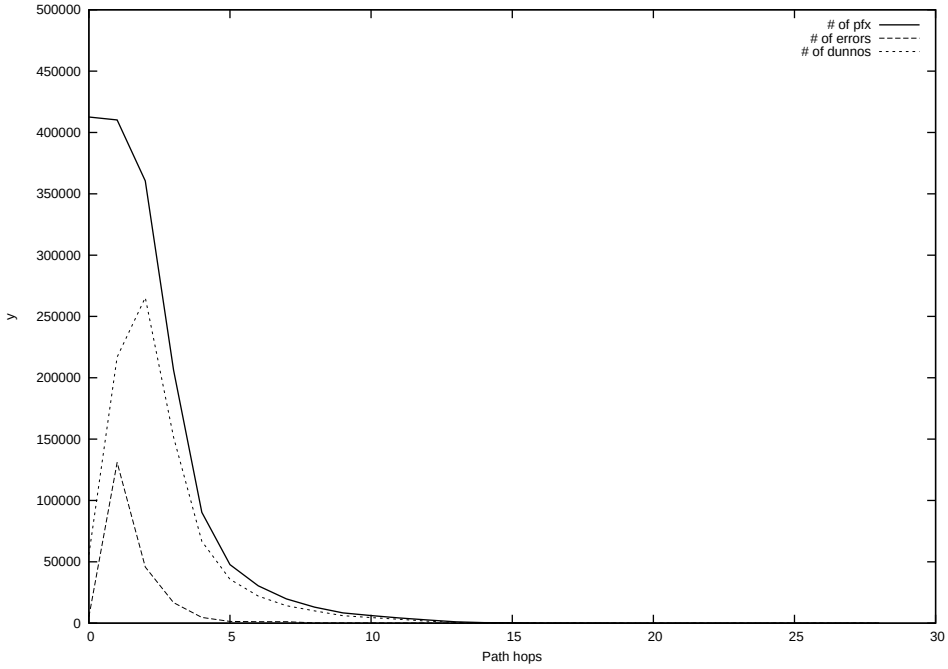
2012-07-13



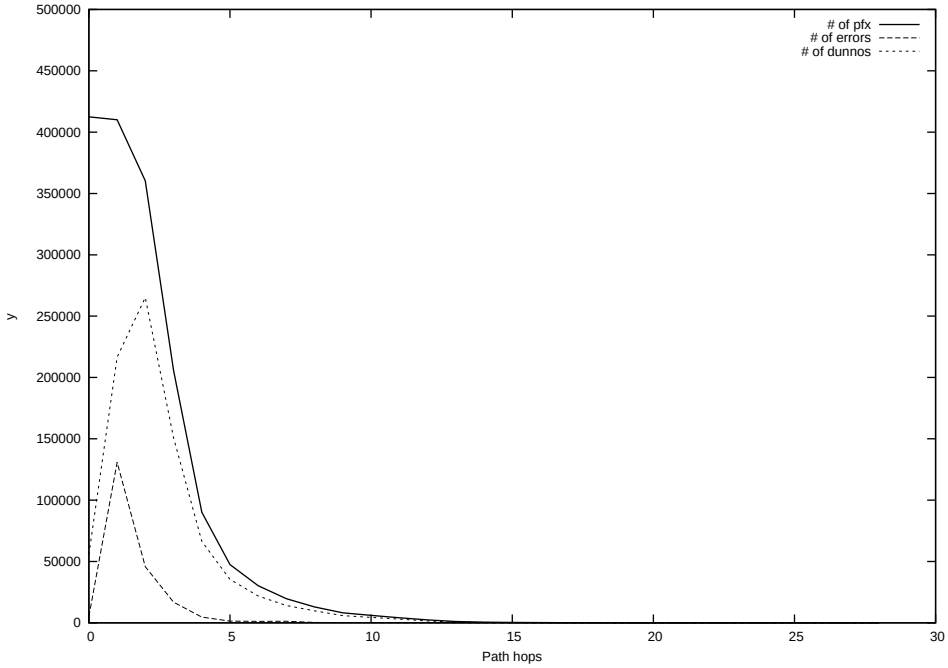
2012-07-14



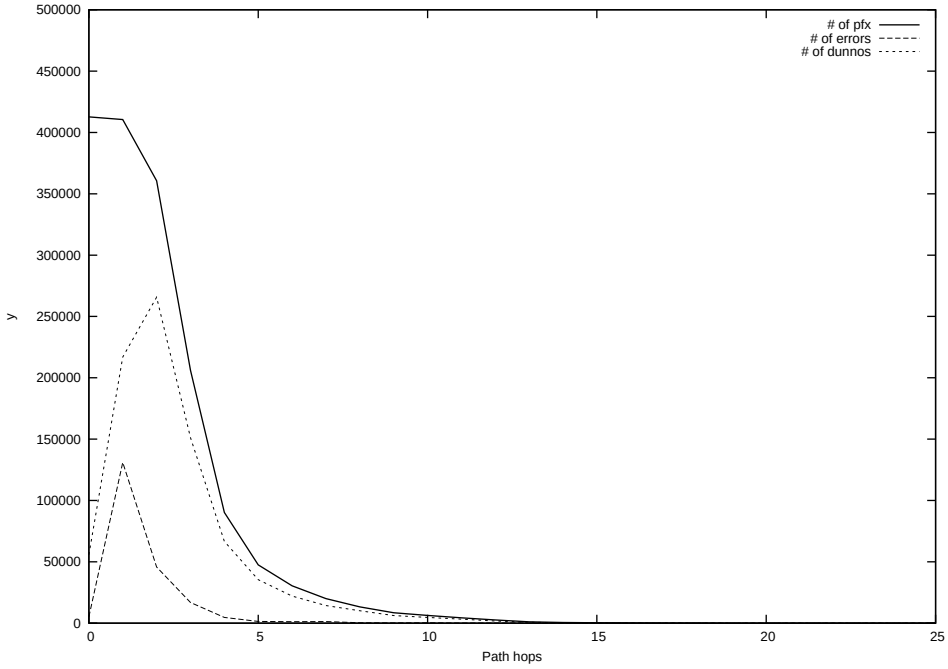
2012-07-15



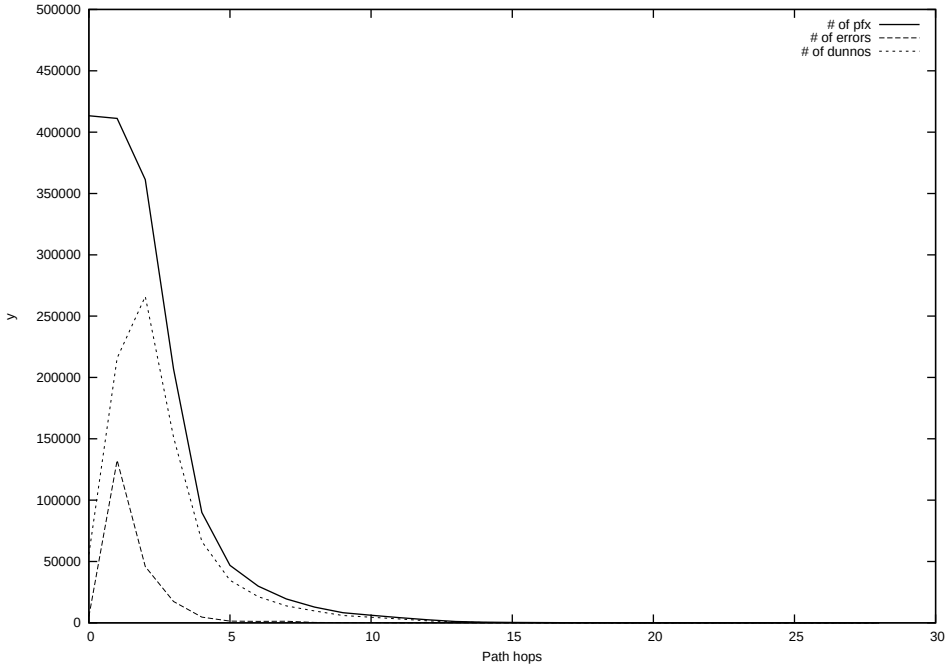
2012-07-16



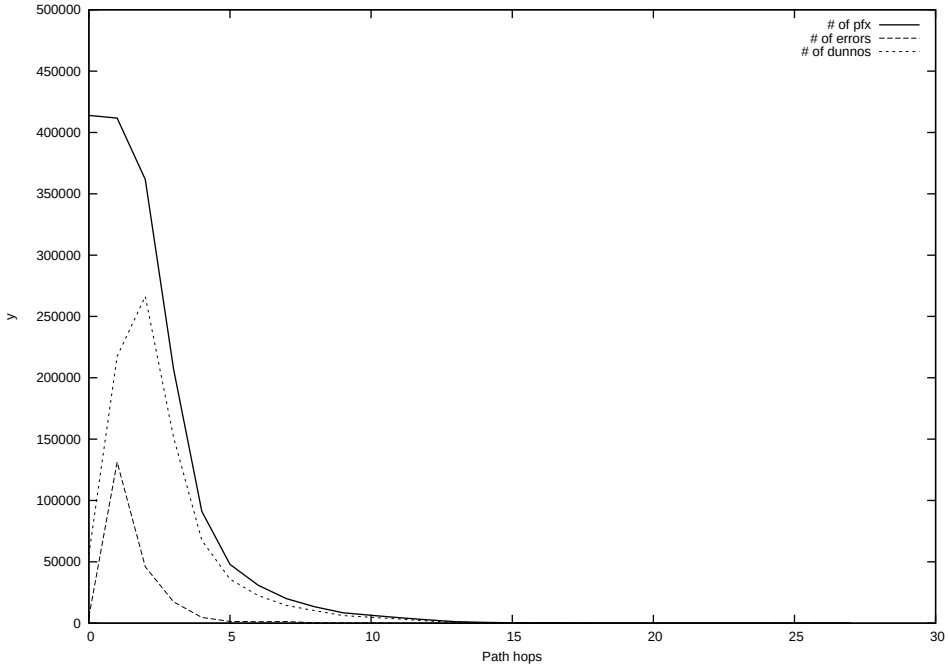
2012-07-17



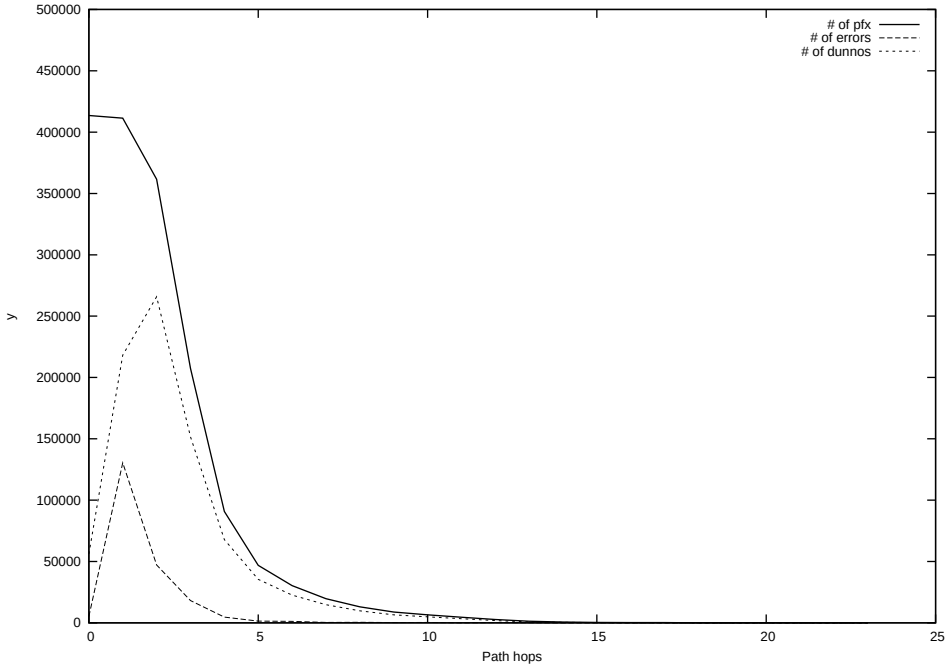
2012-07-18



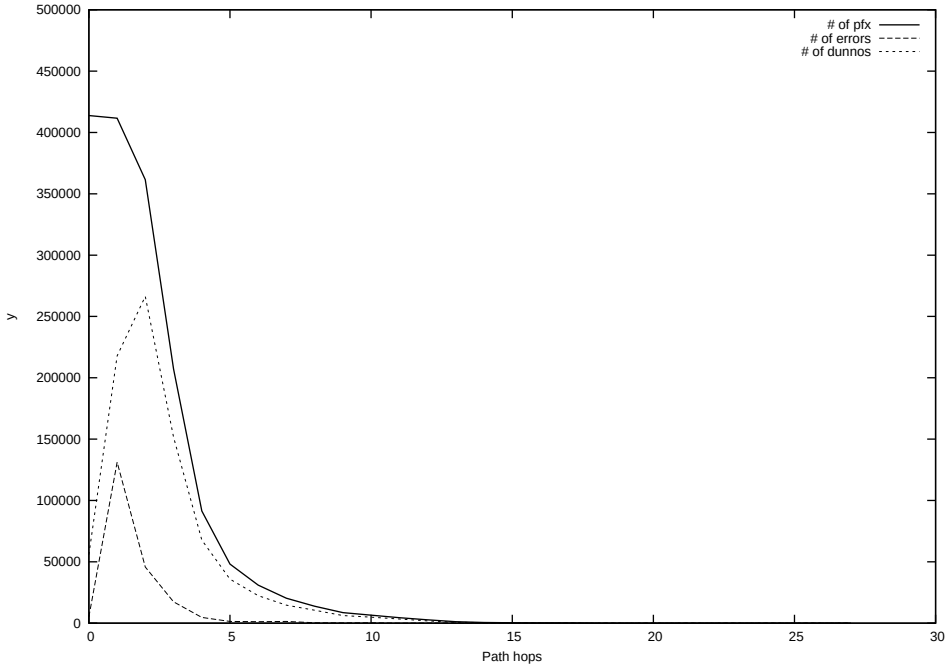
2012-07-19



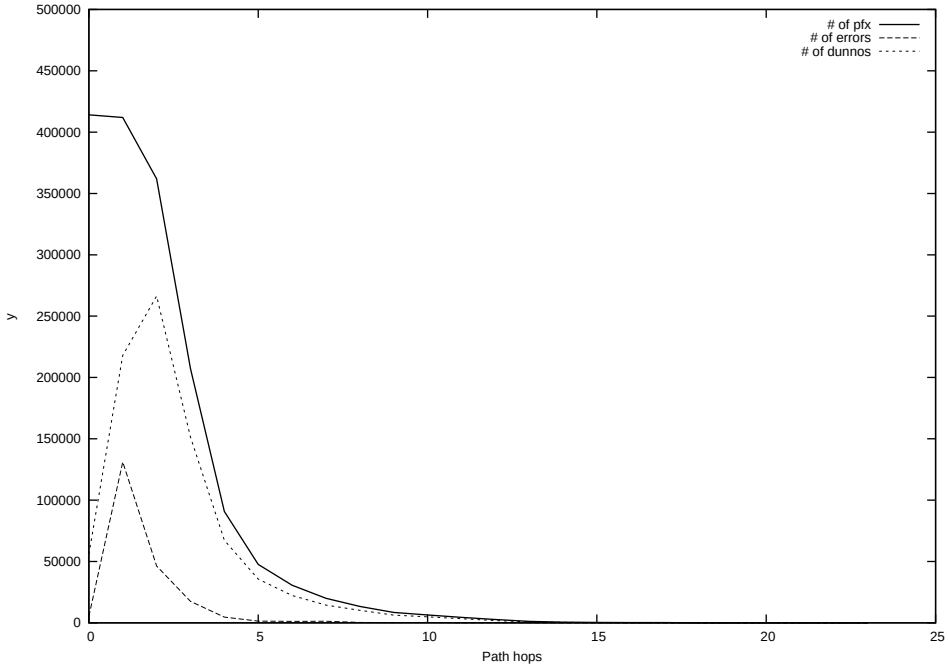
2012-07-20



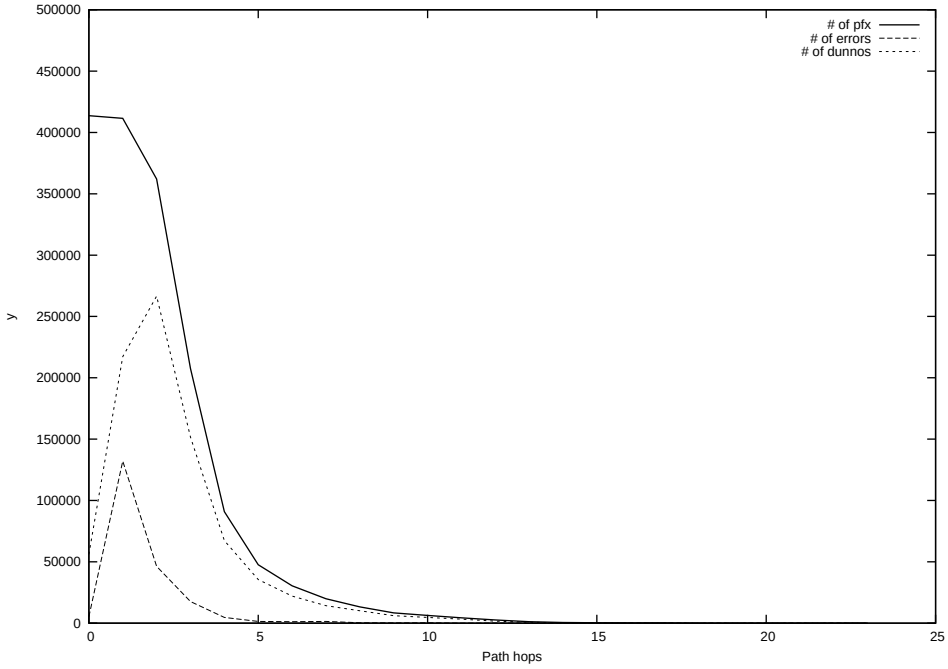
2012-07-21



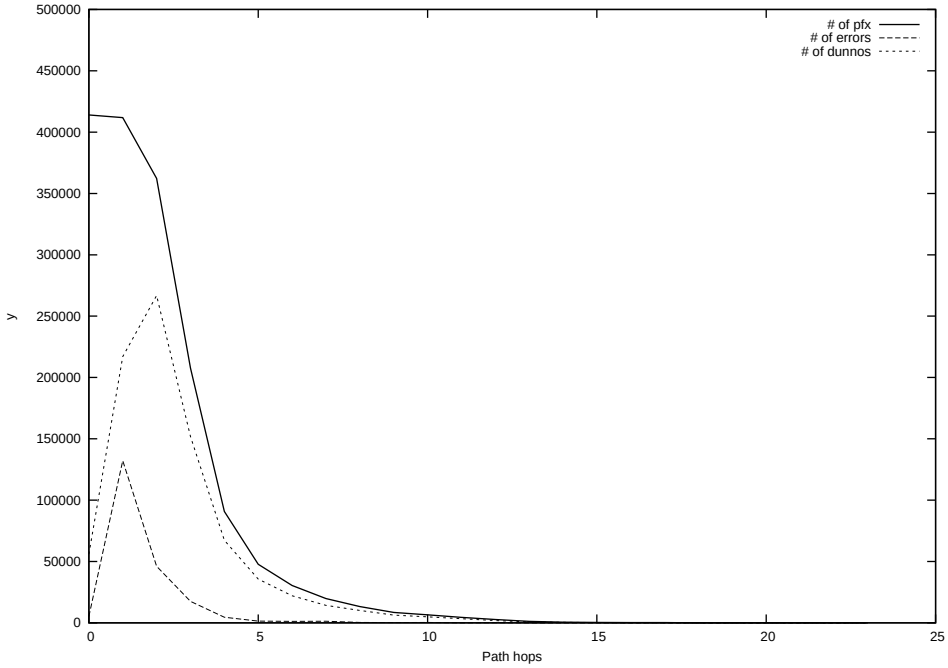
2012-07-22



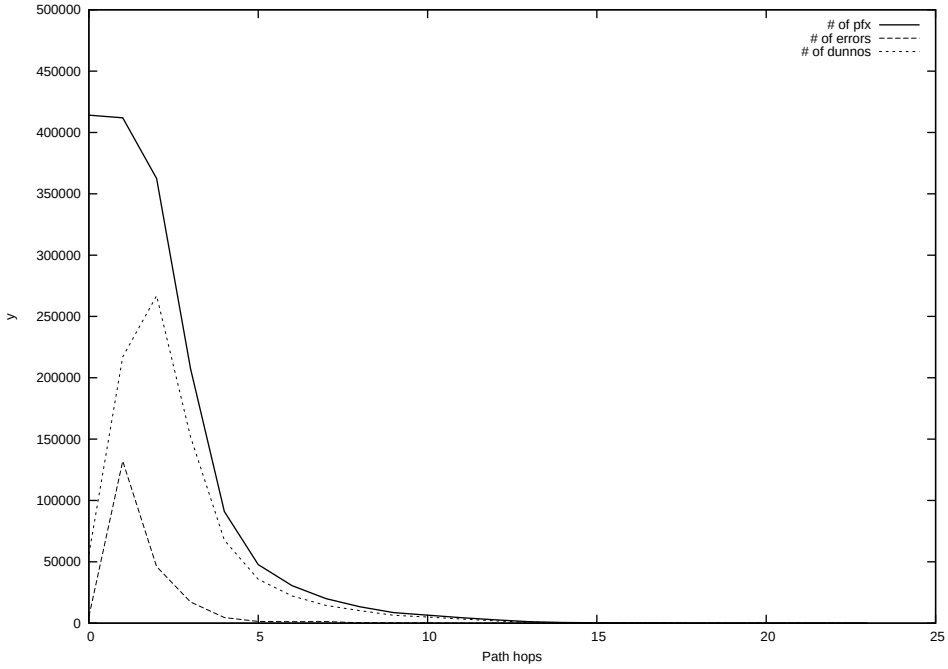
2012-07-23



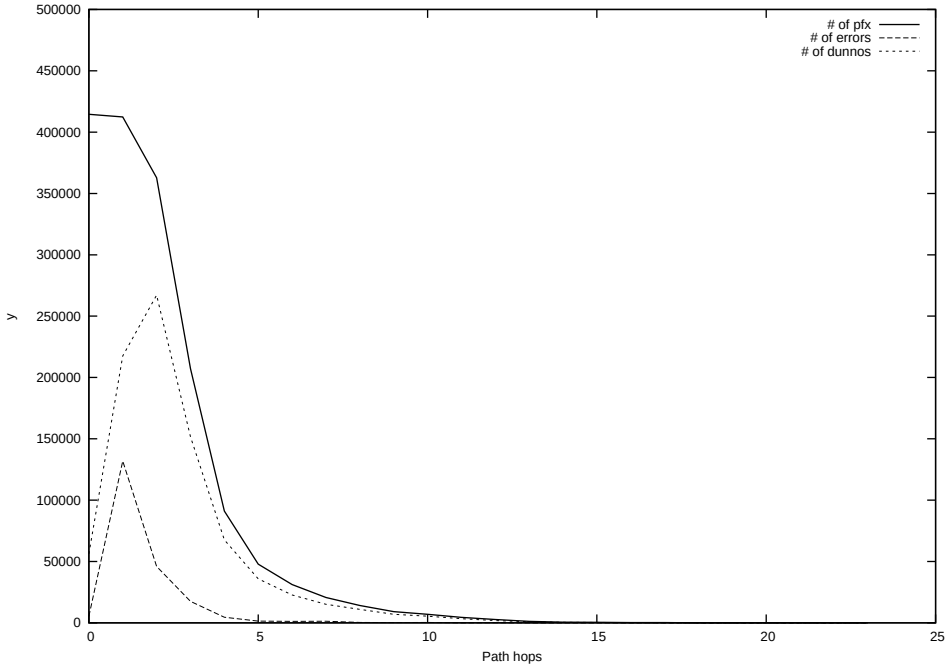
2012-07-24



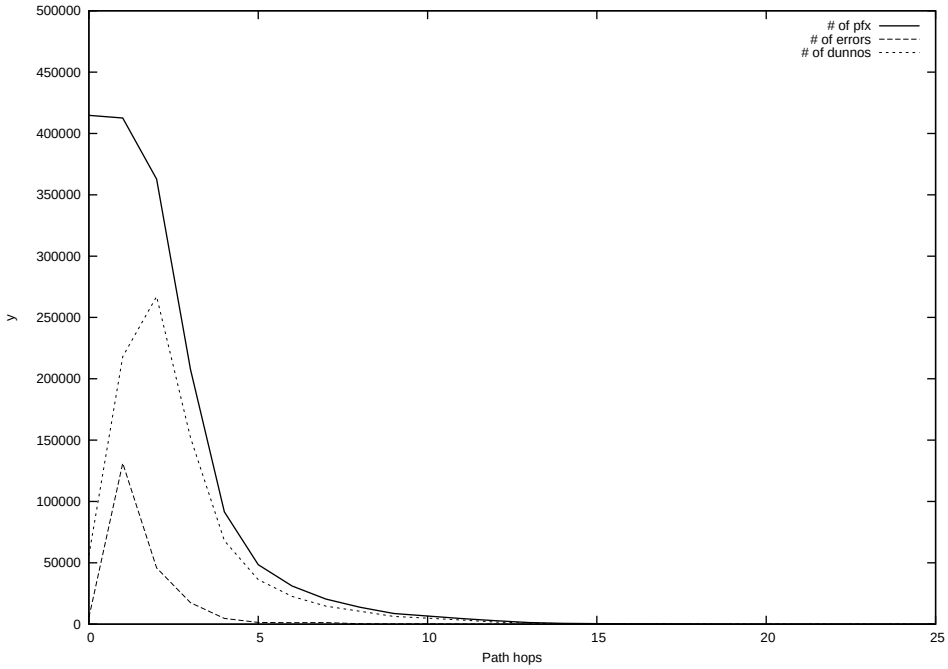
2012-07-25



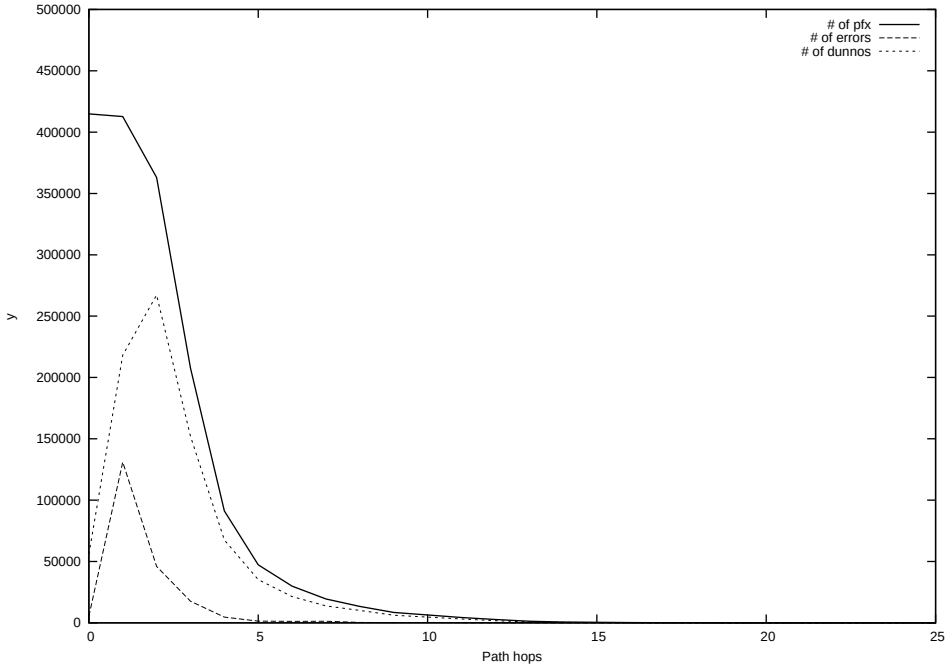
2012-07-26



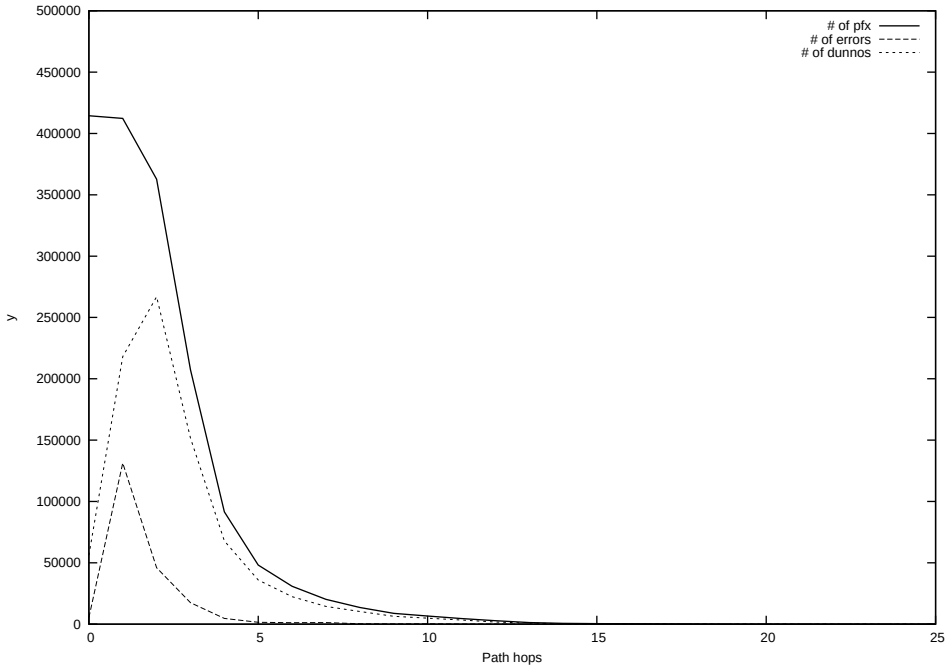
2012-07-27



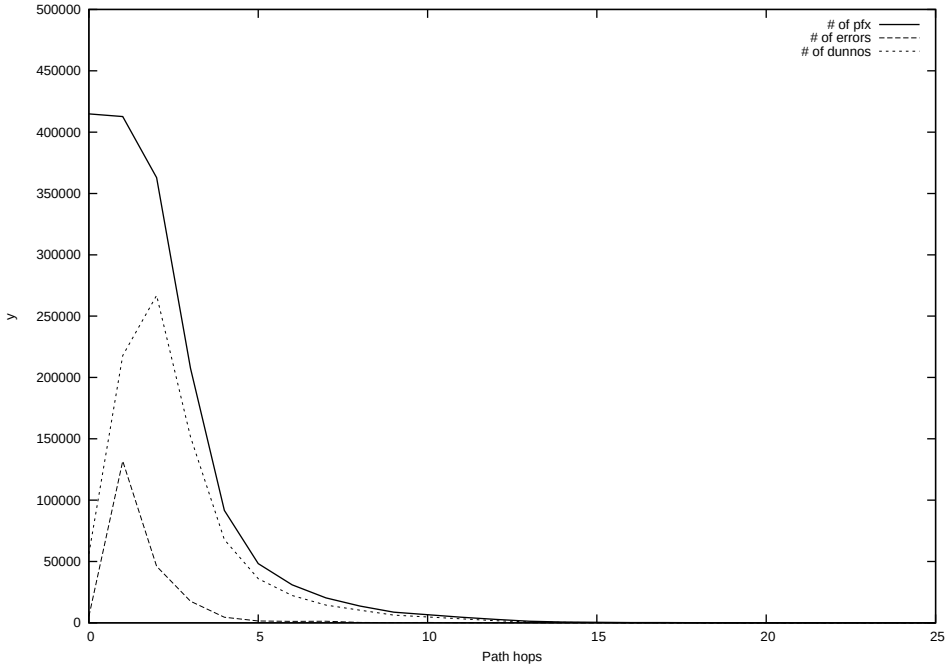
2012-07-28



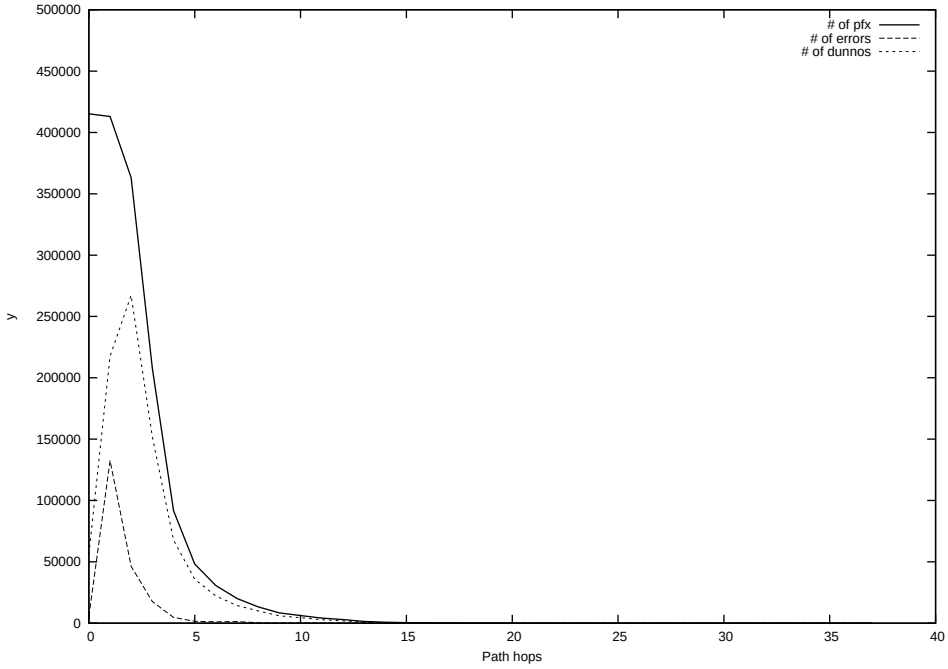
2012-07-29



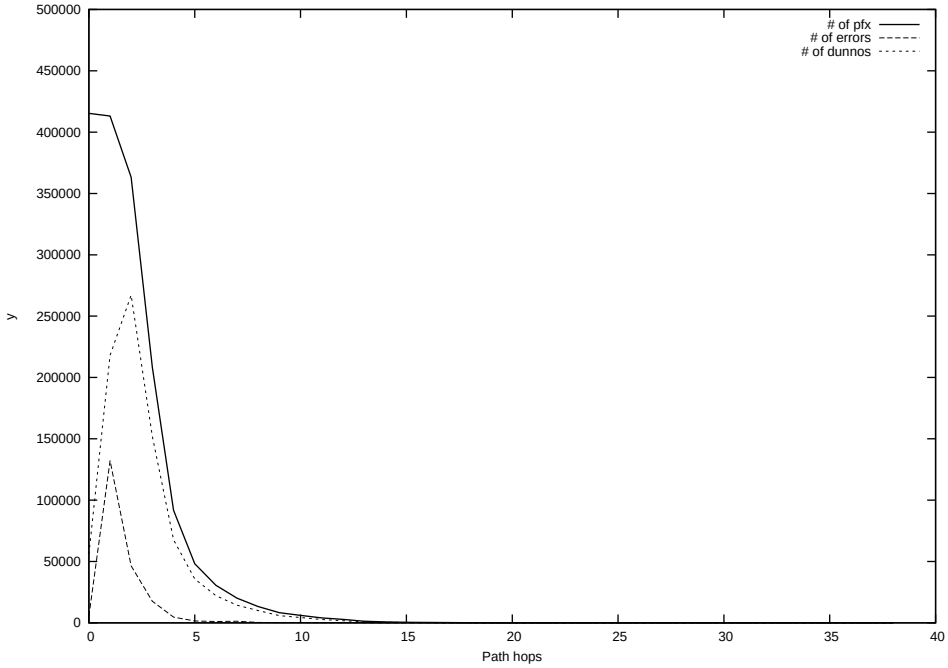
2012-07-30



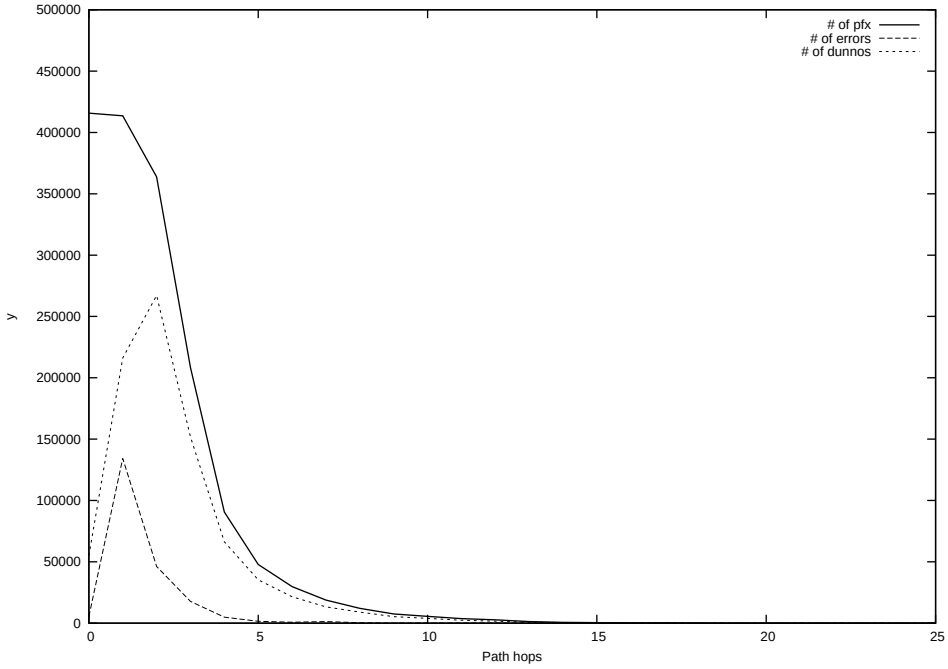
2012-07-31



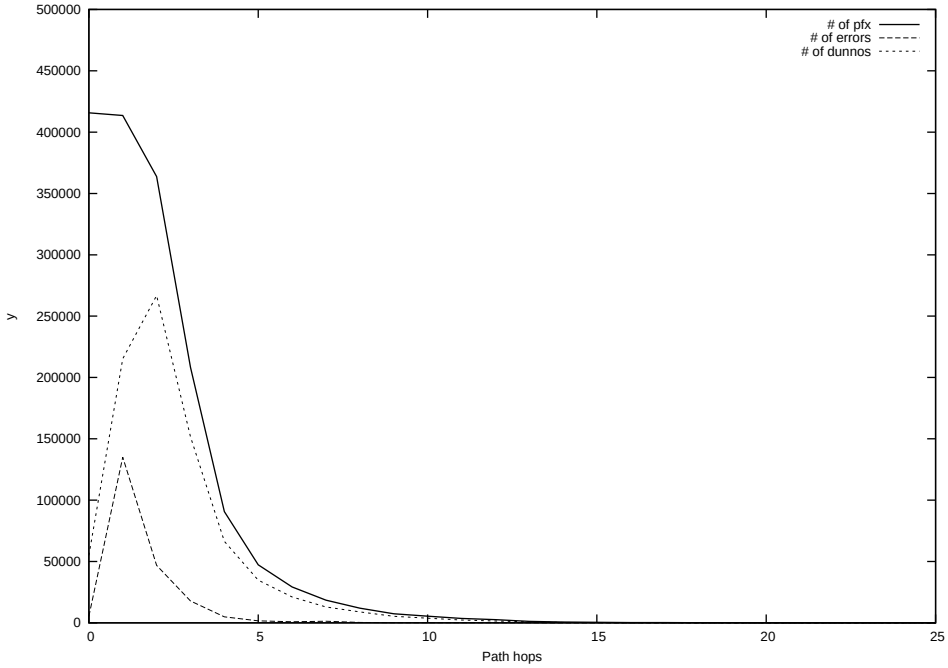
2012-08-01



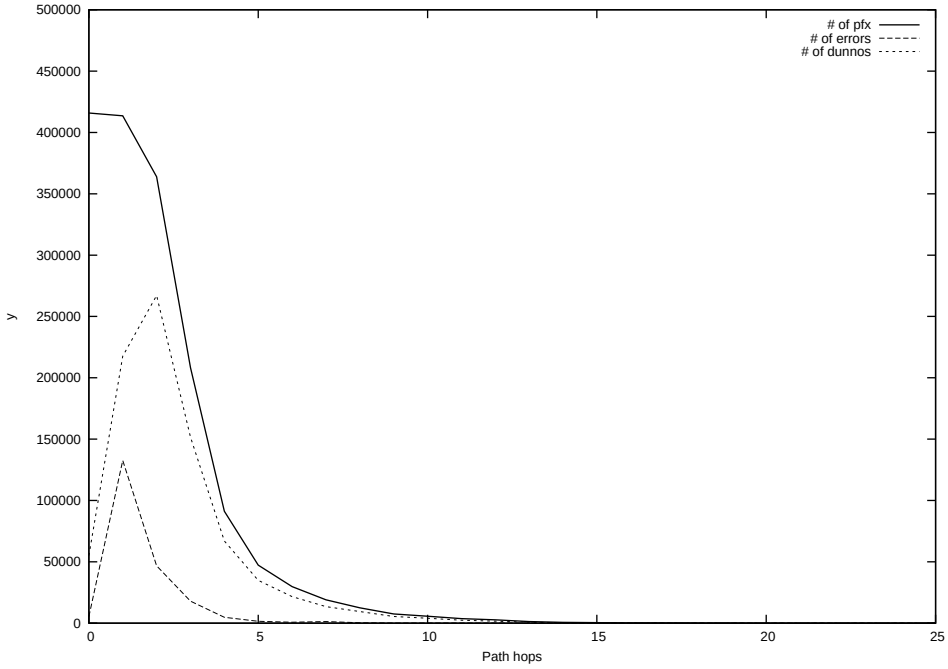
2012-08-02



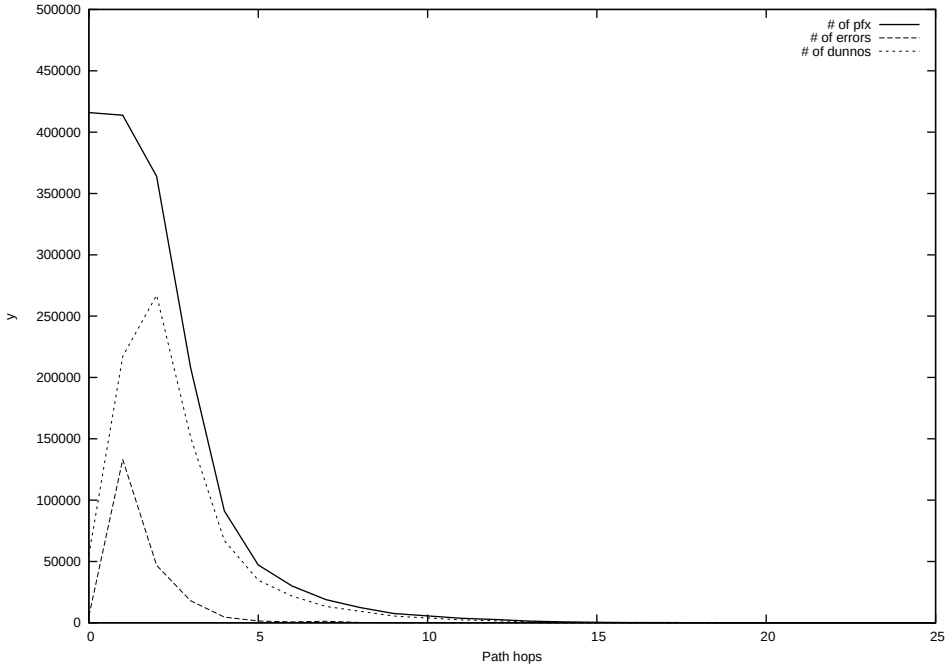
2012-08-03



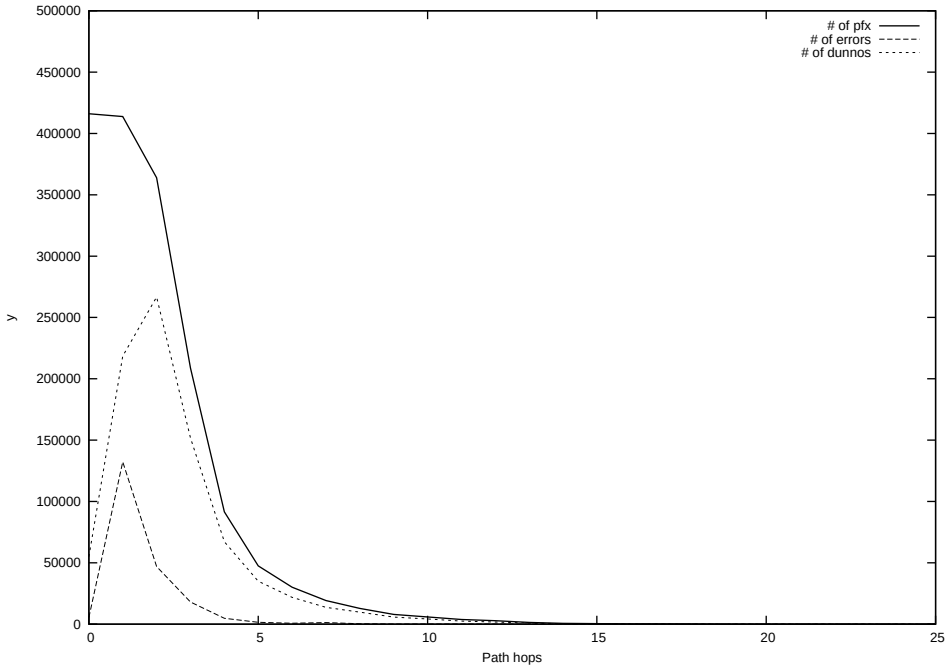
2012-08-04



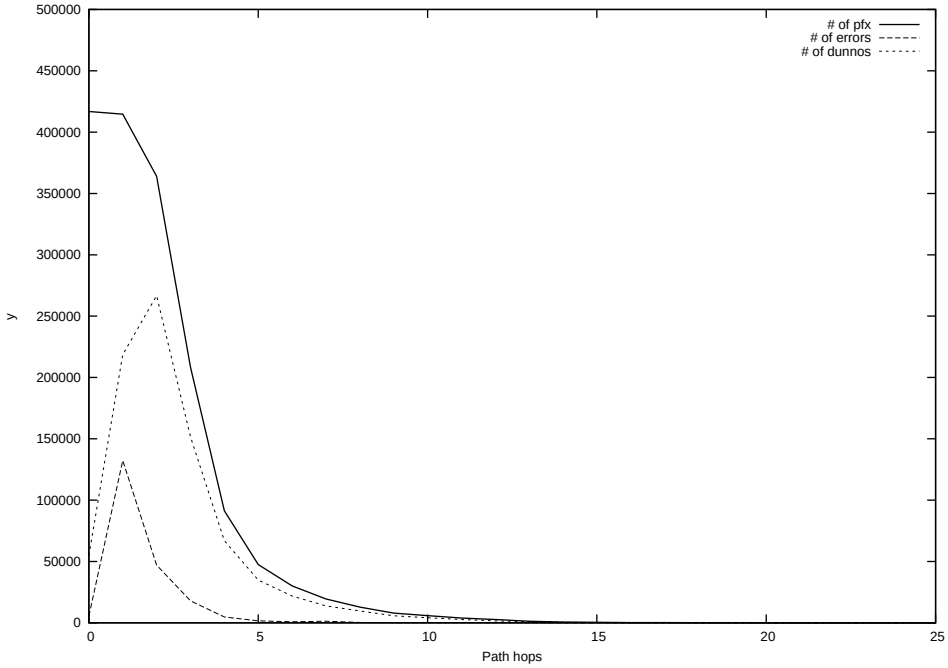
2012-08-05



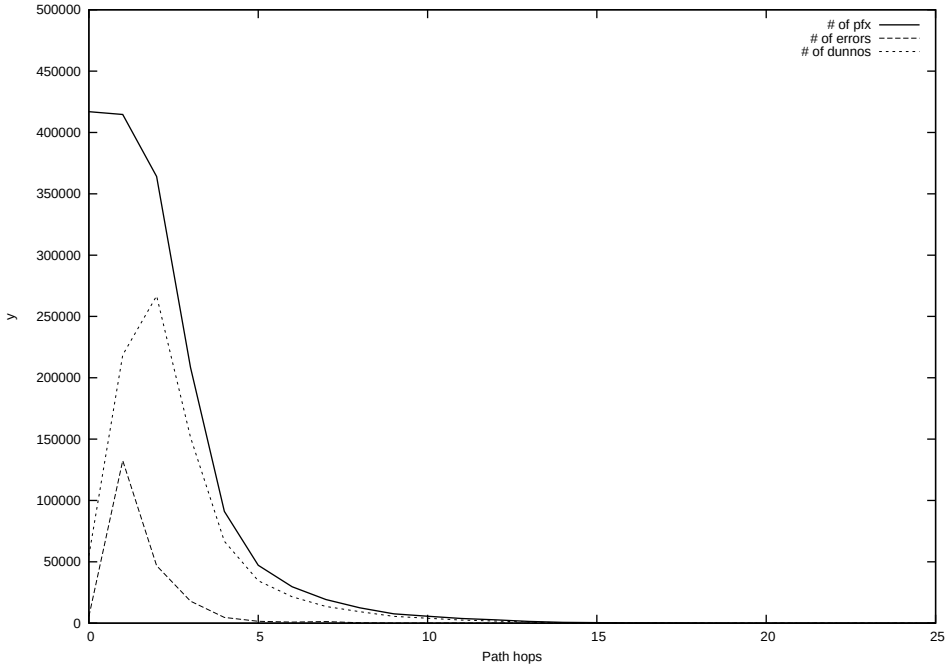
2012-08-06



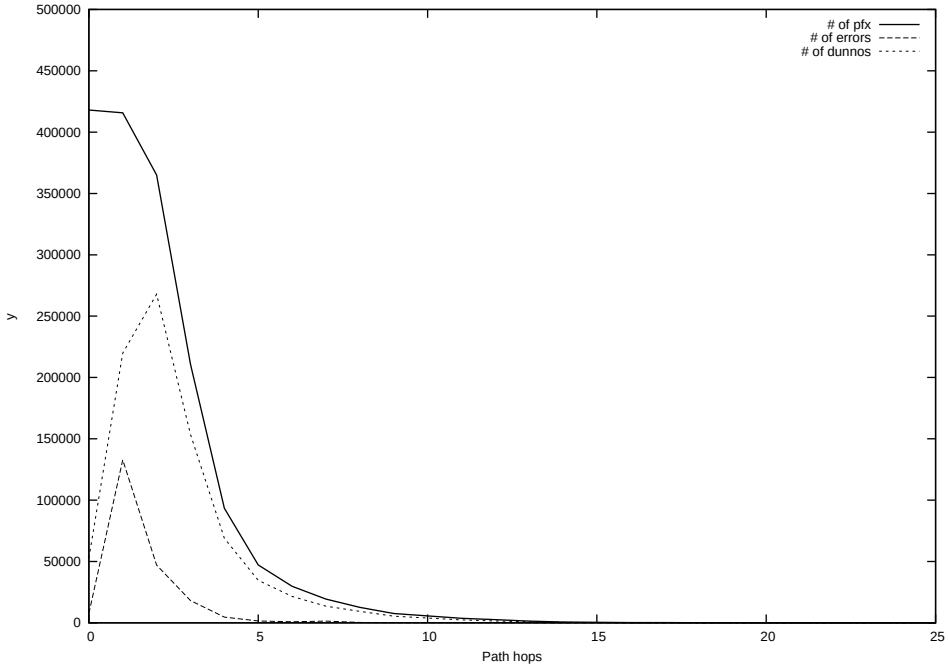
2012-08-07



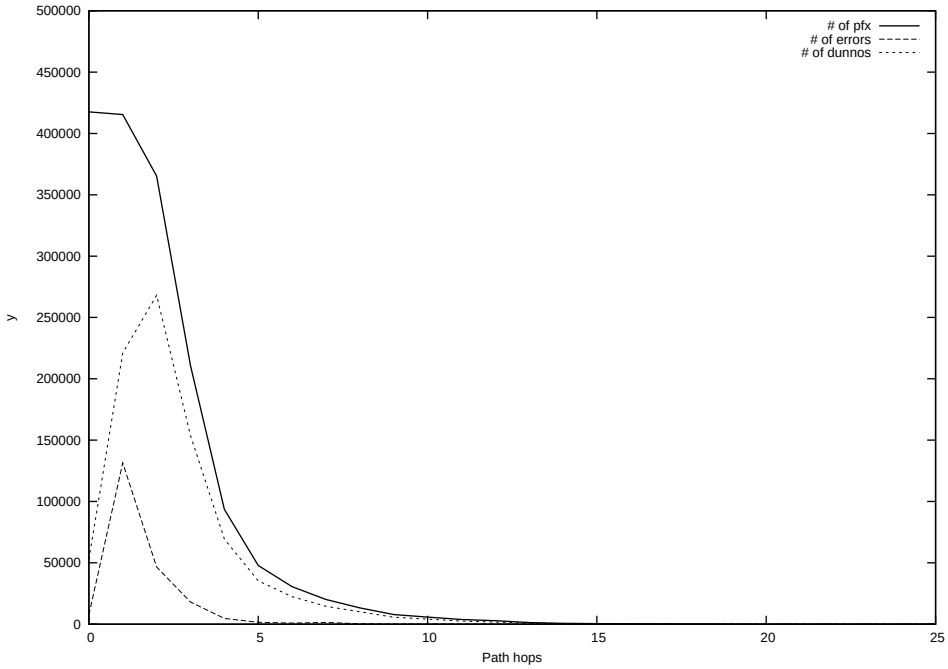
2012-08-08



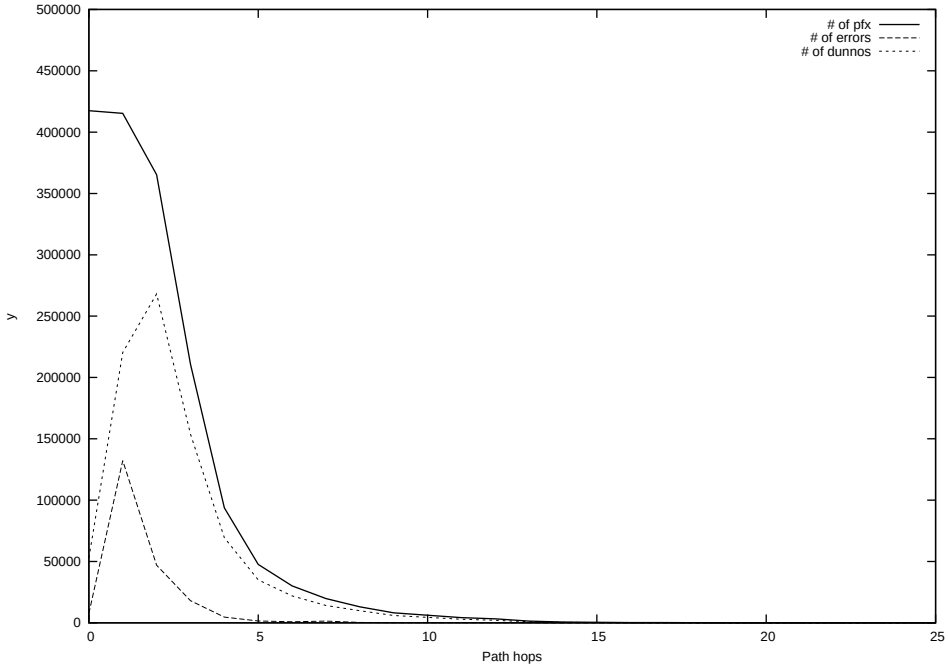
2012-08-09



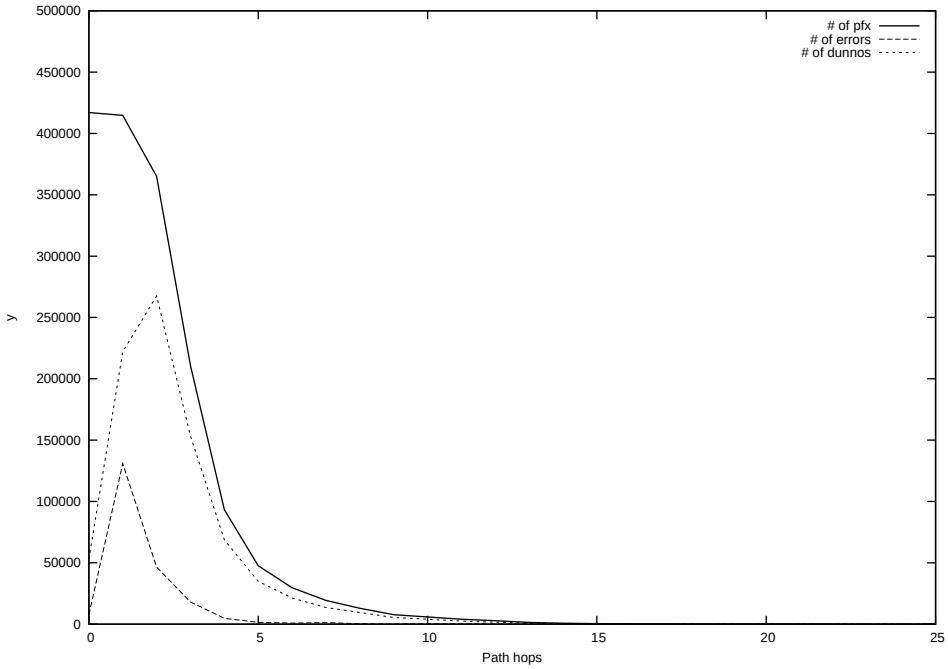
2012-08-10



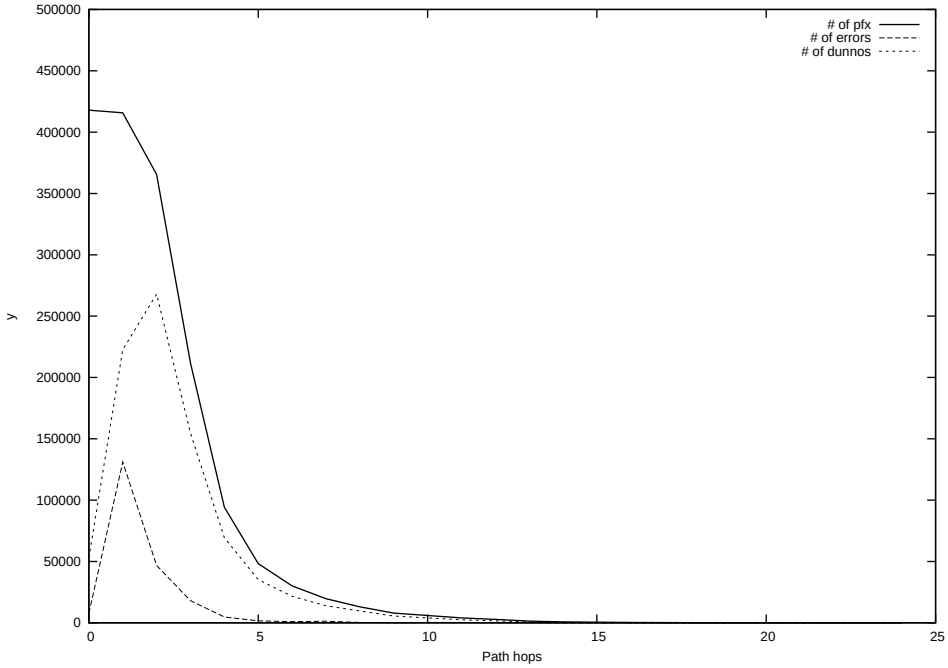
2012-08-11



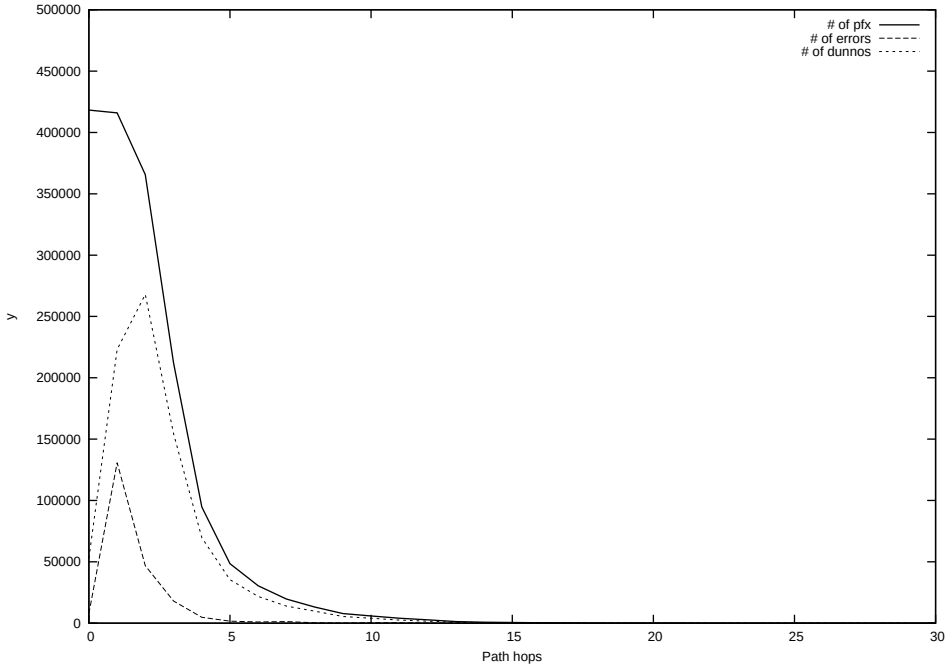
2012-08-12



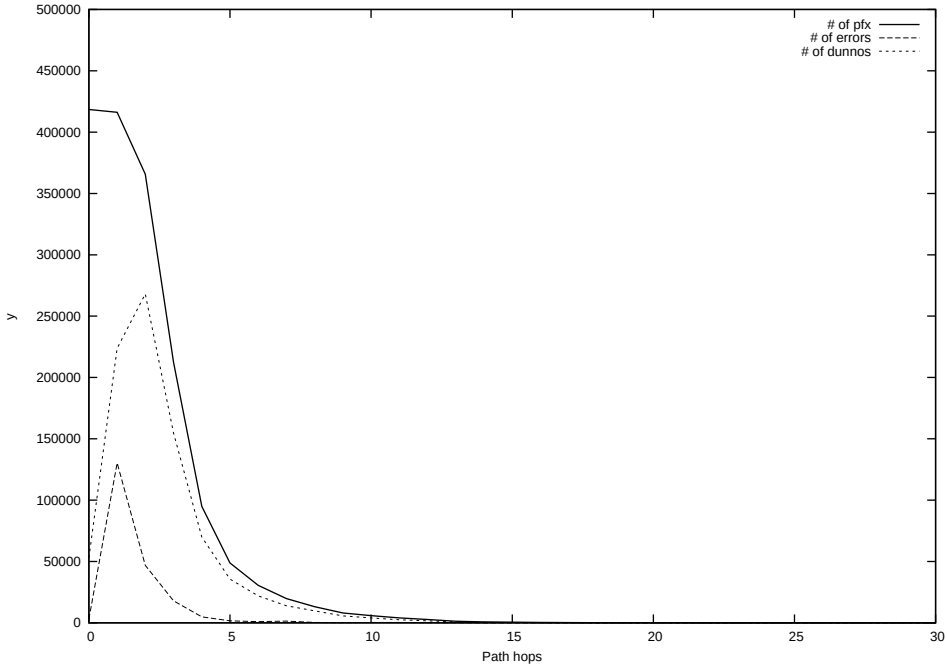
2012-08-13



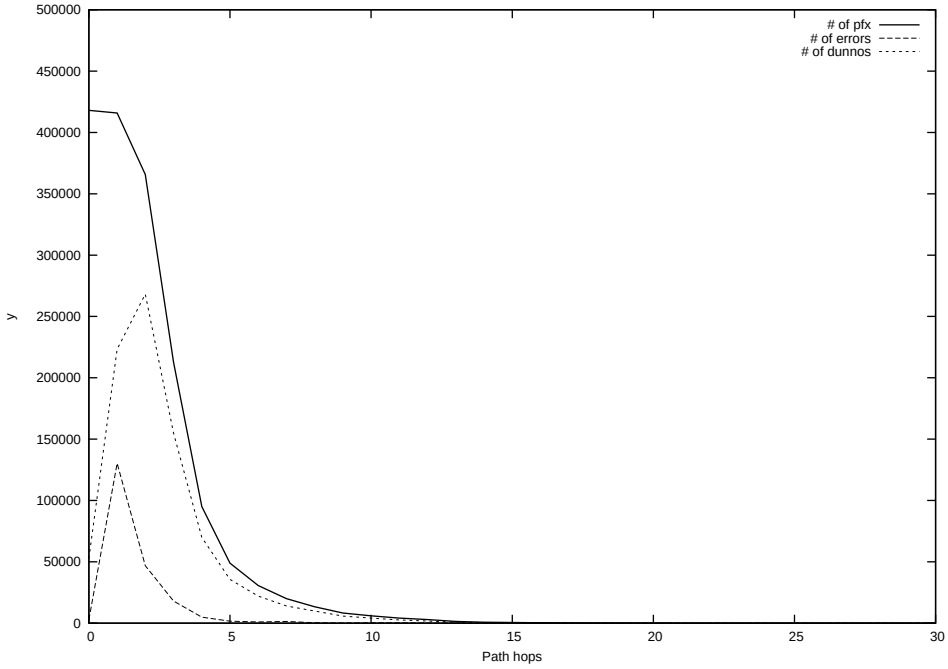
2012-08-14



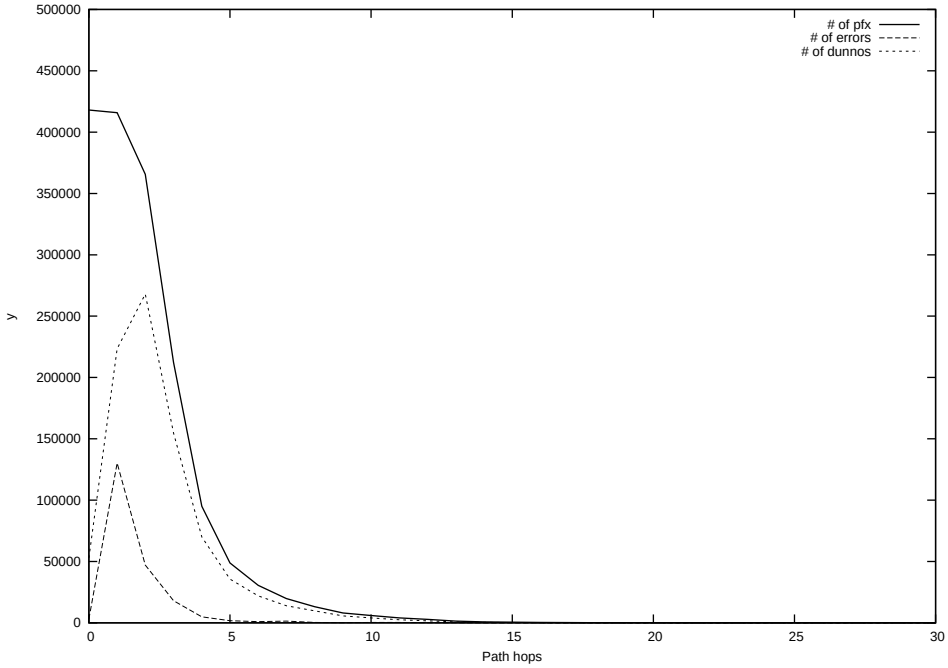
2012-08-15



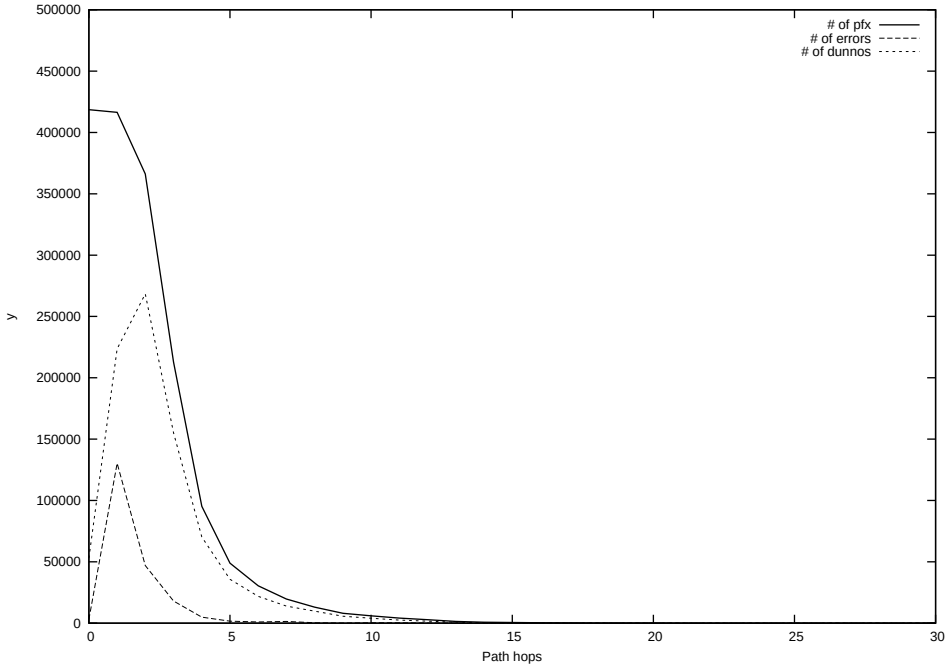
2012-08-16



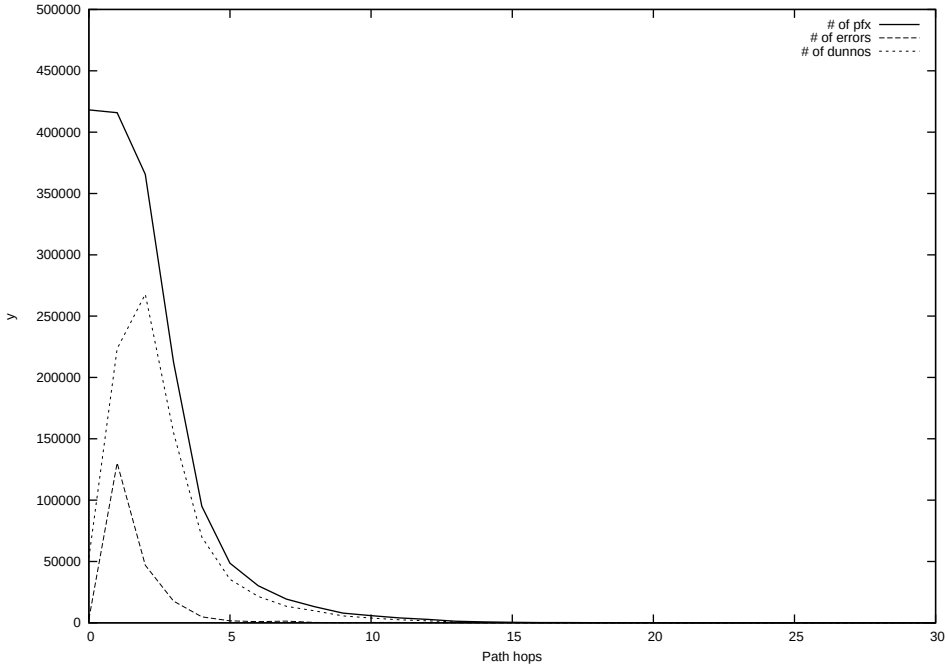
2012-08-17



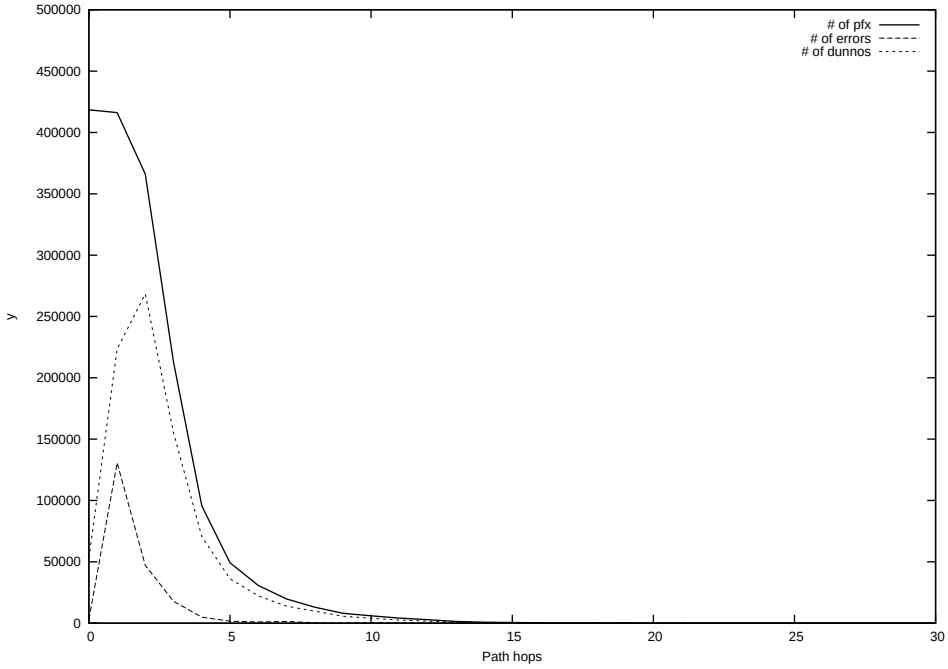
2012-08-18



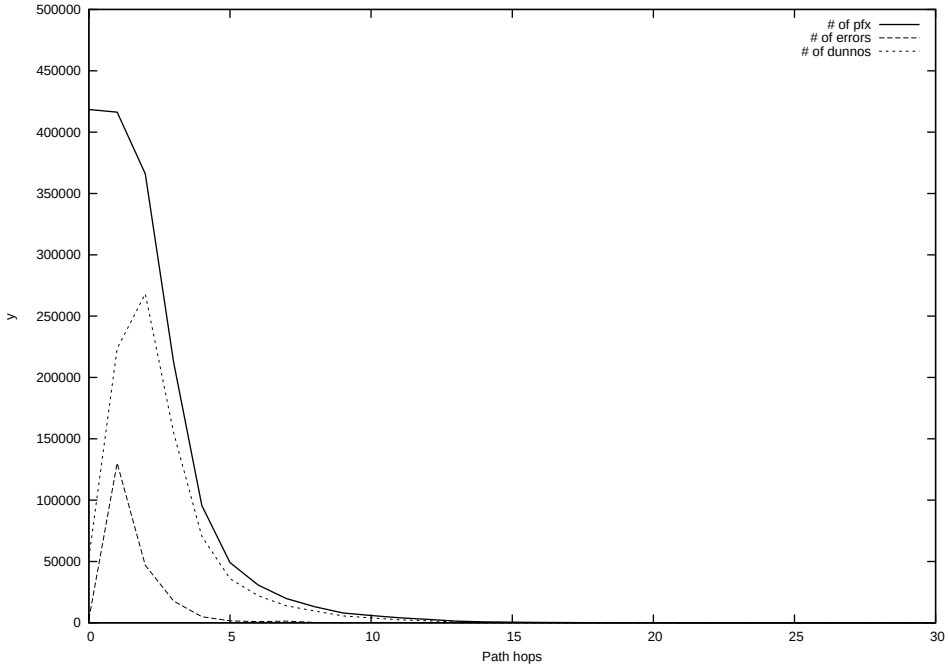
2012-08-19



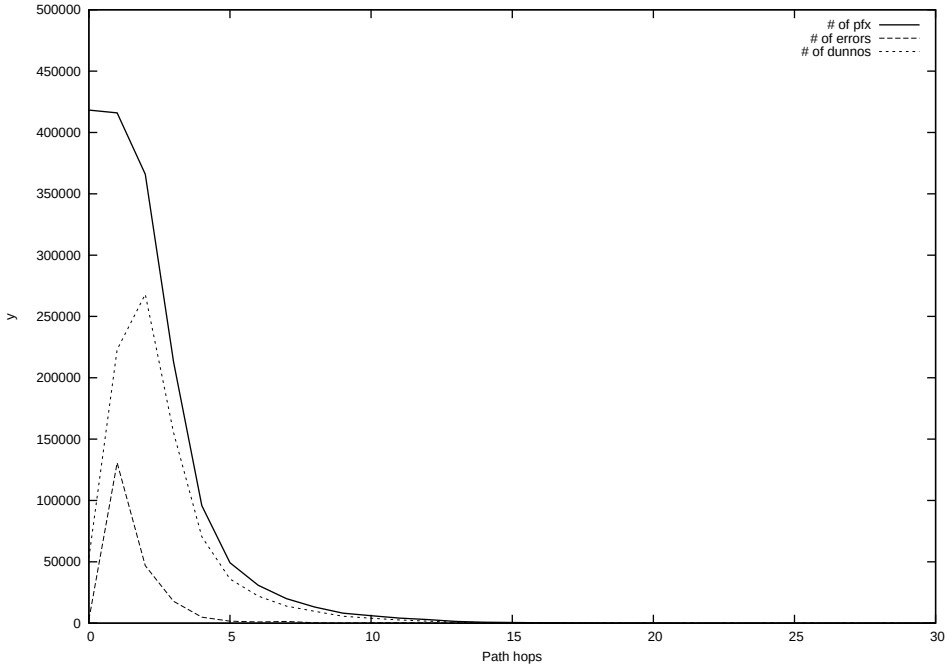
2012-08-20



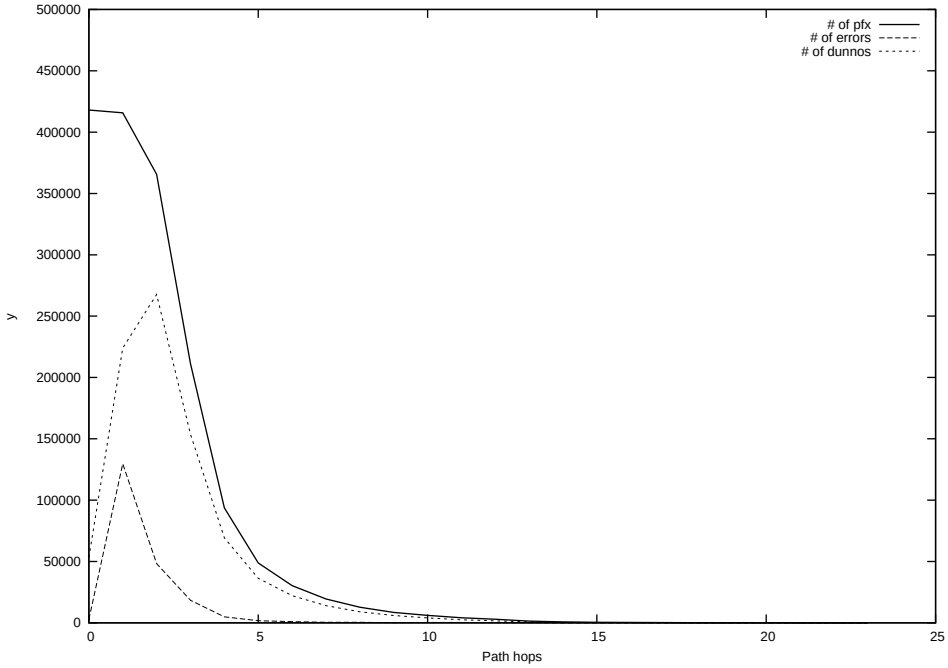
2012-08-21



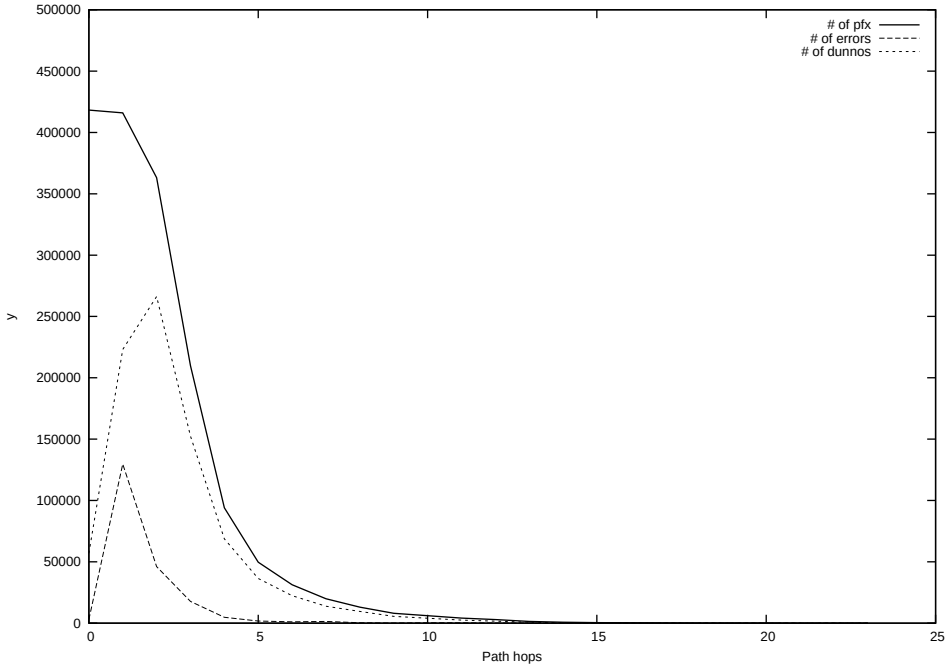
2012-08-22



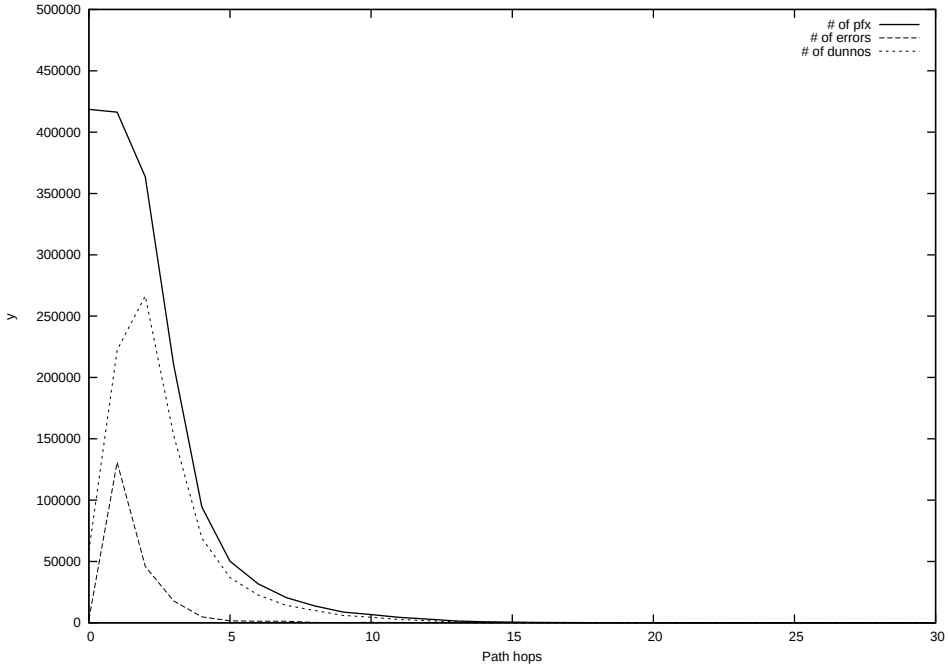
2012-08-23



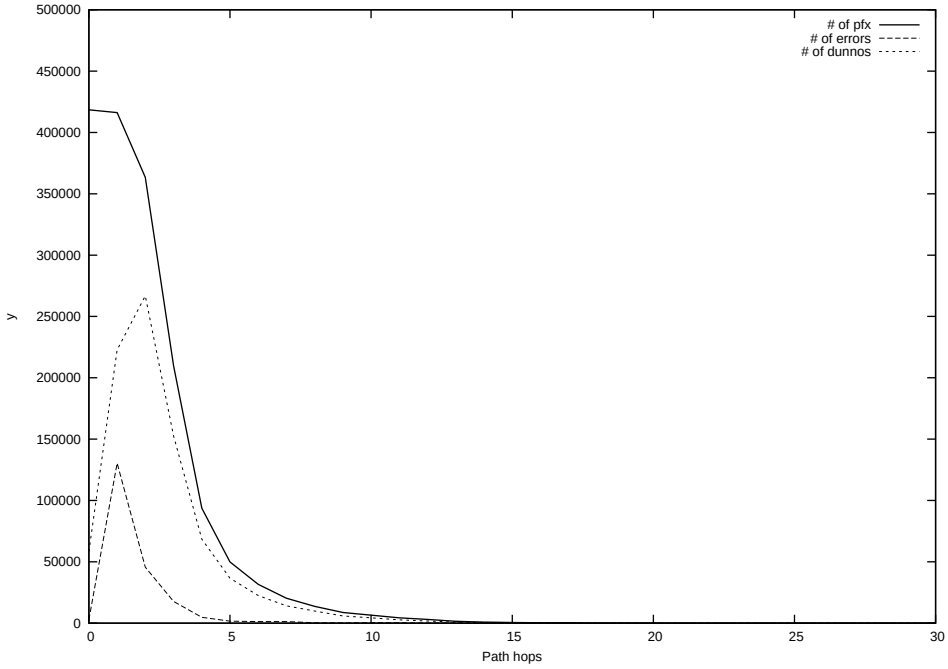
2012-08-24



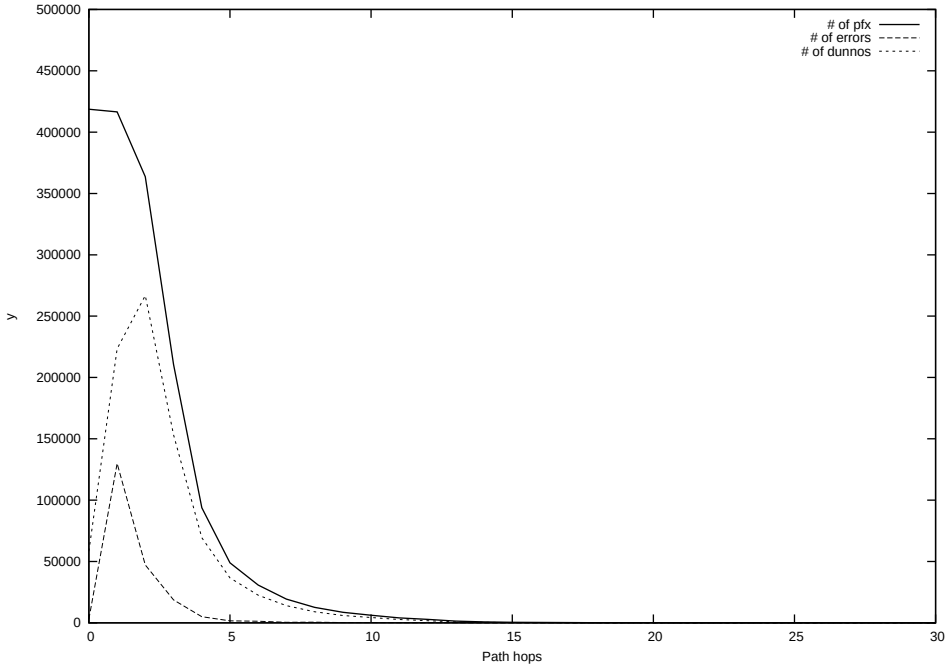
2012-08-25



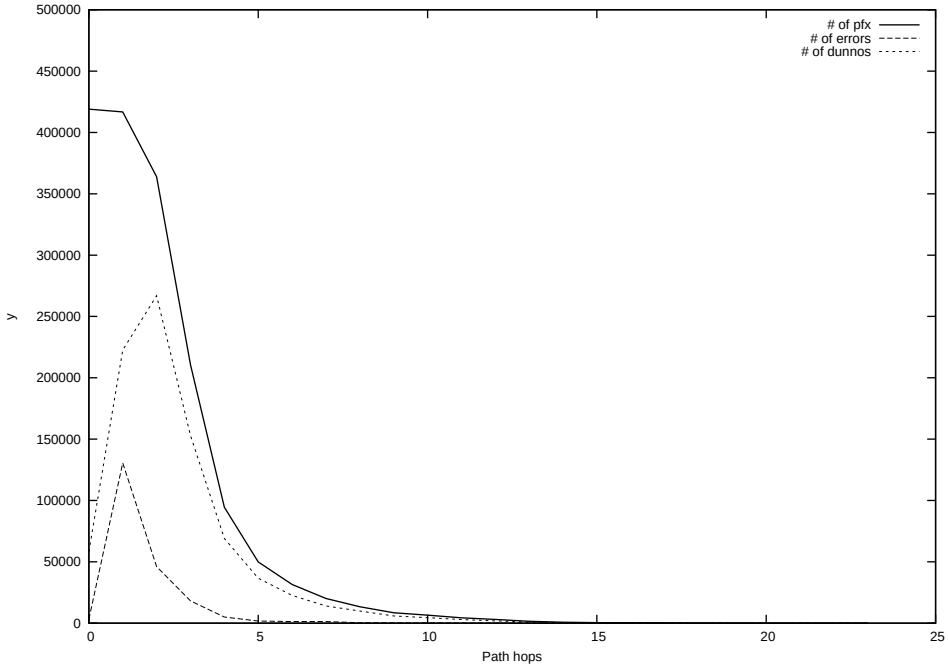
2012-08-26



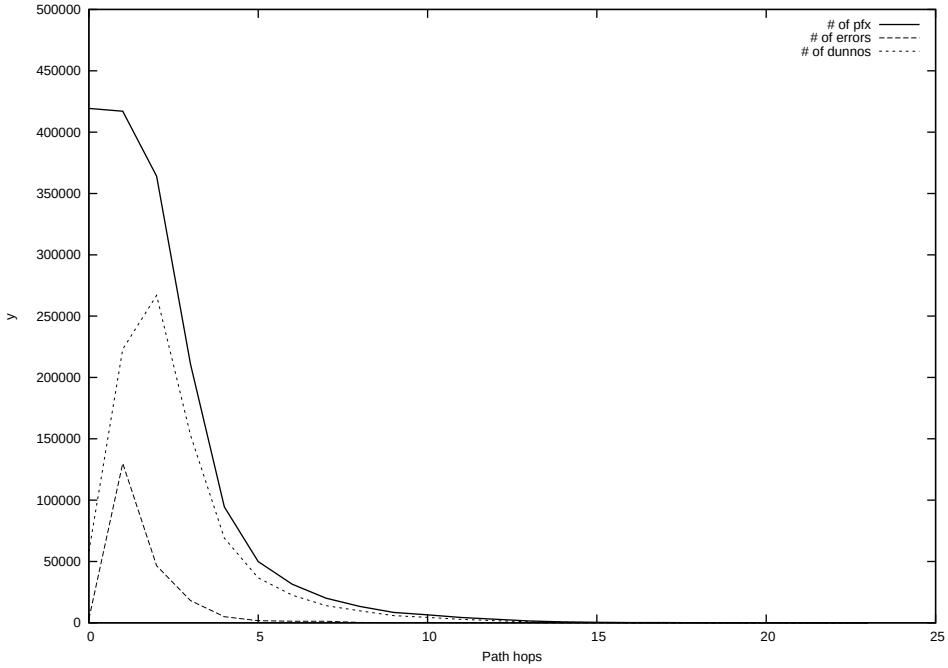
2012-08-27



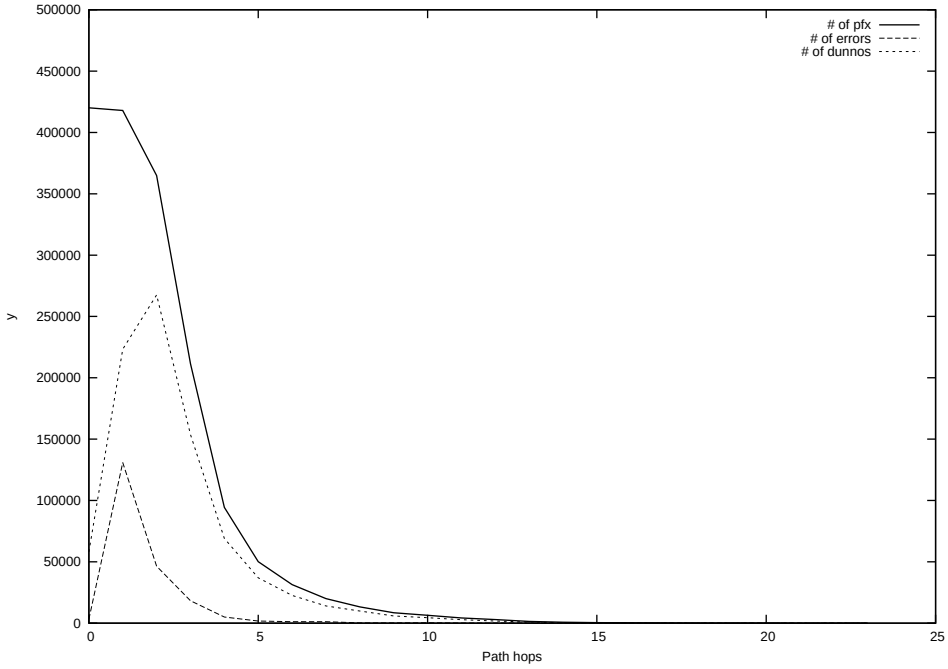
2012-08-28



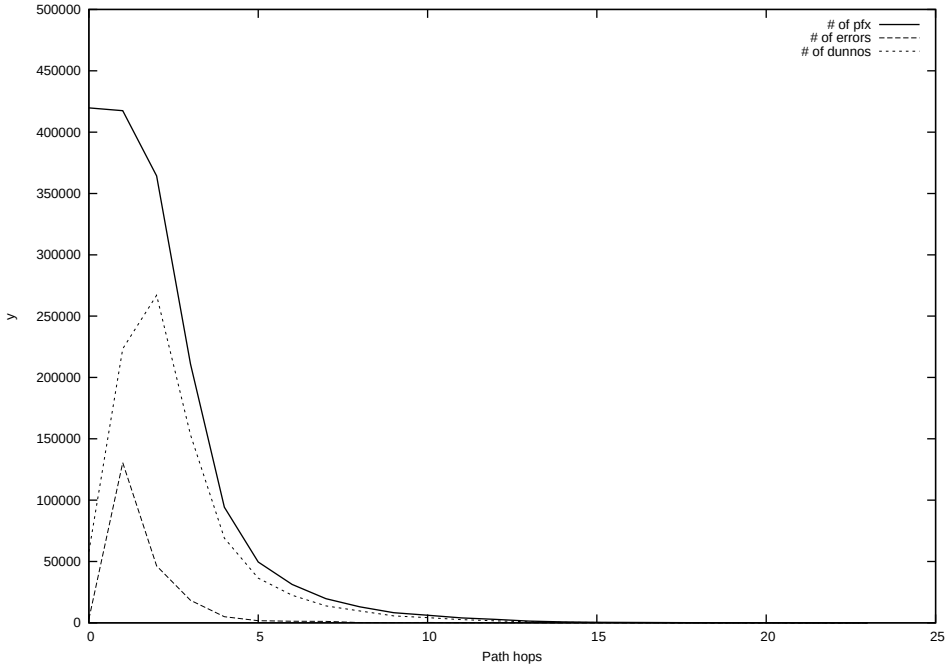
2012-08-29



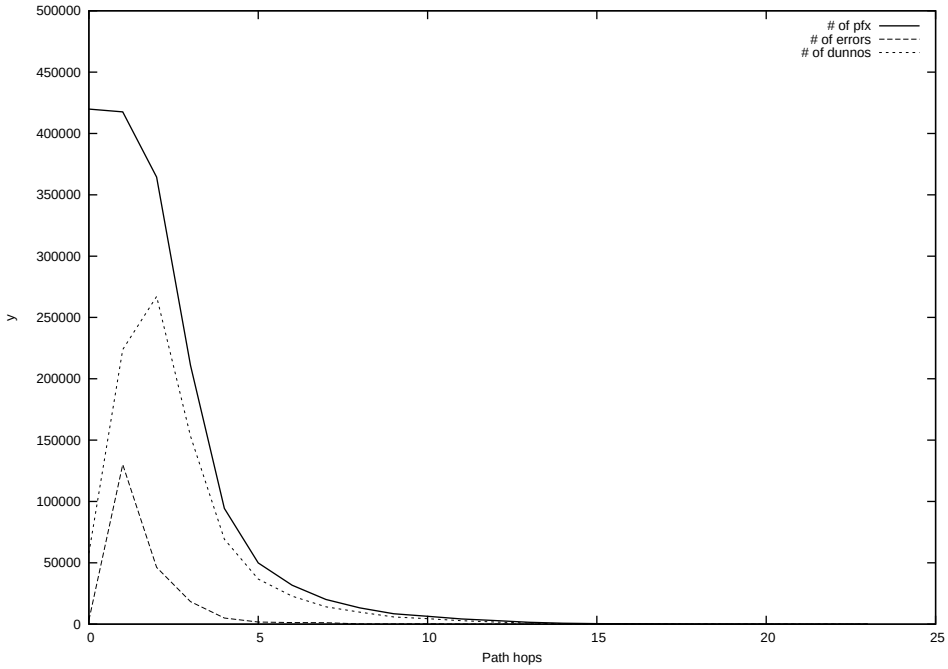
2012-08-30



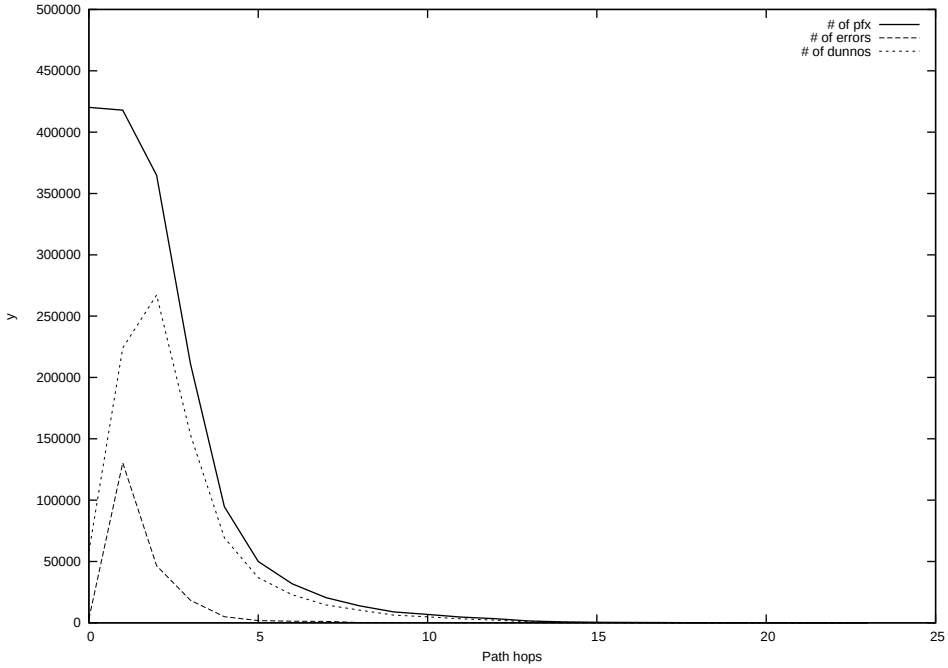
2012-08-31



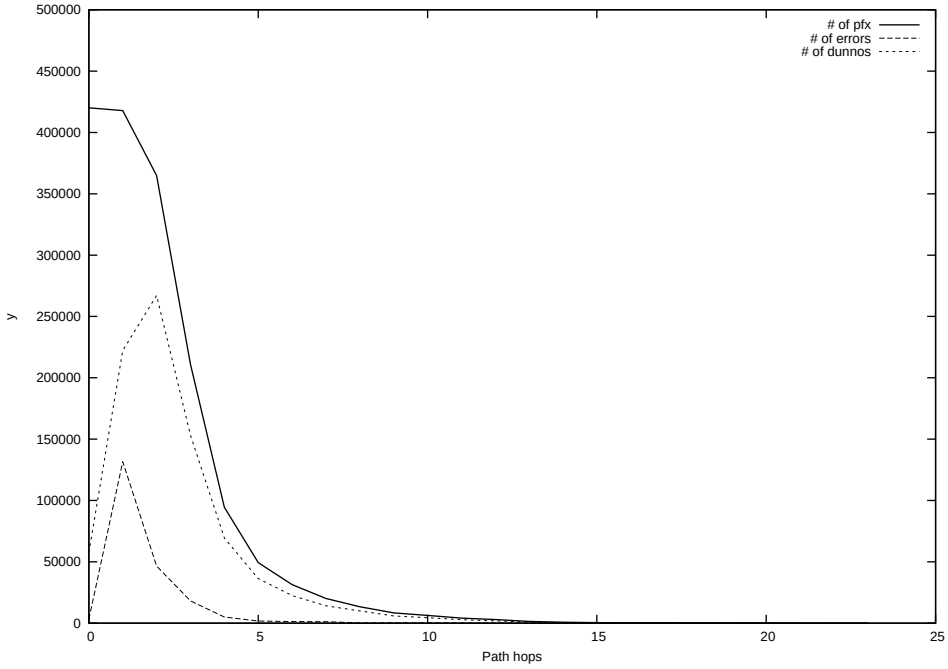
2012-09-01



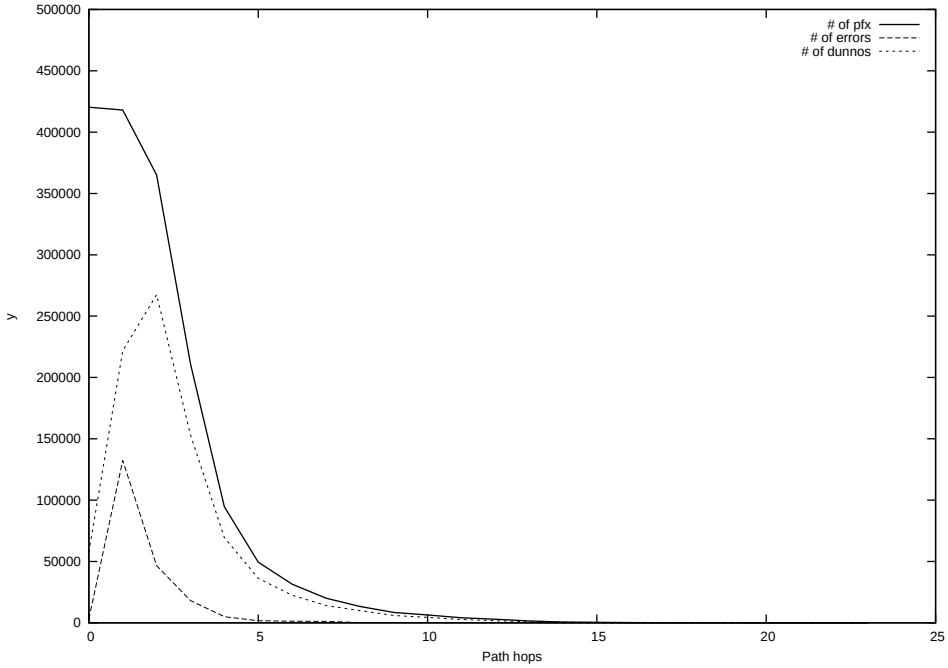
2012-09-02



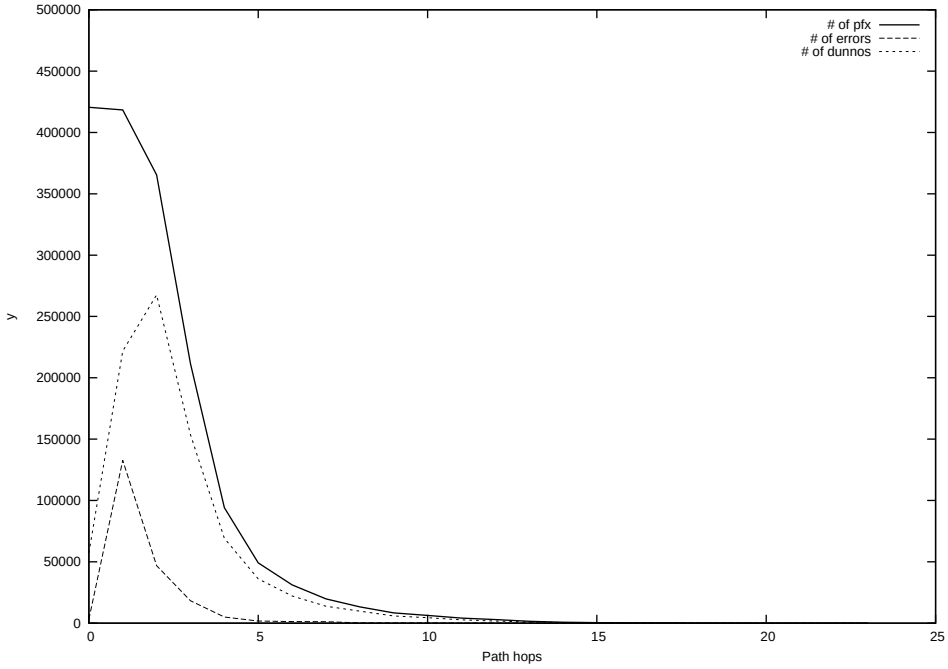
2012-09-03



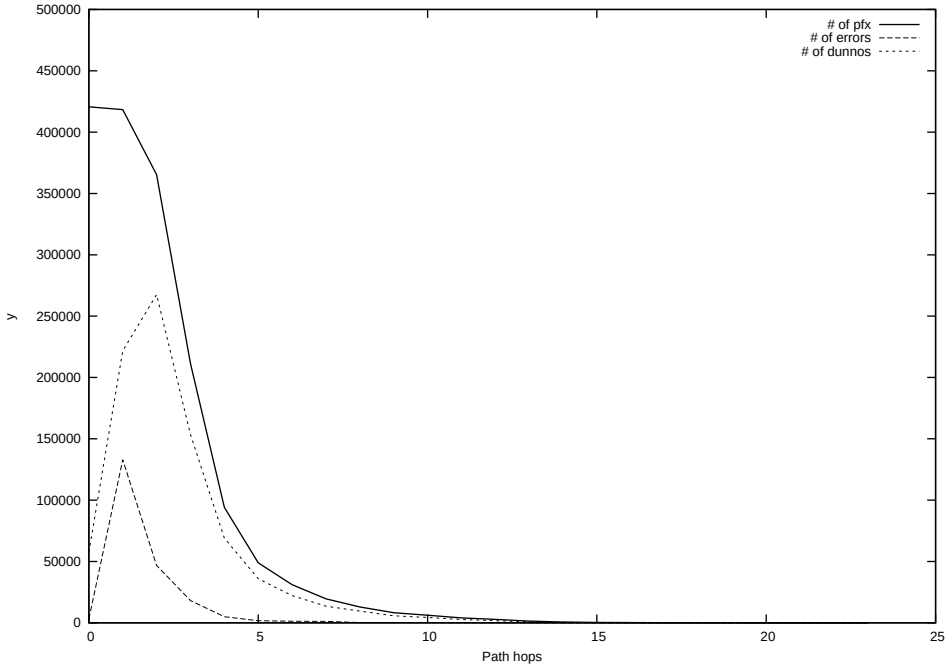
2012-09-04



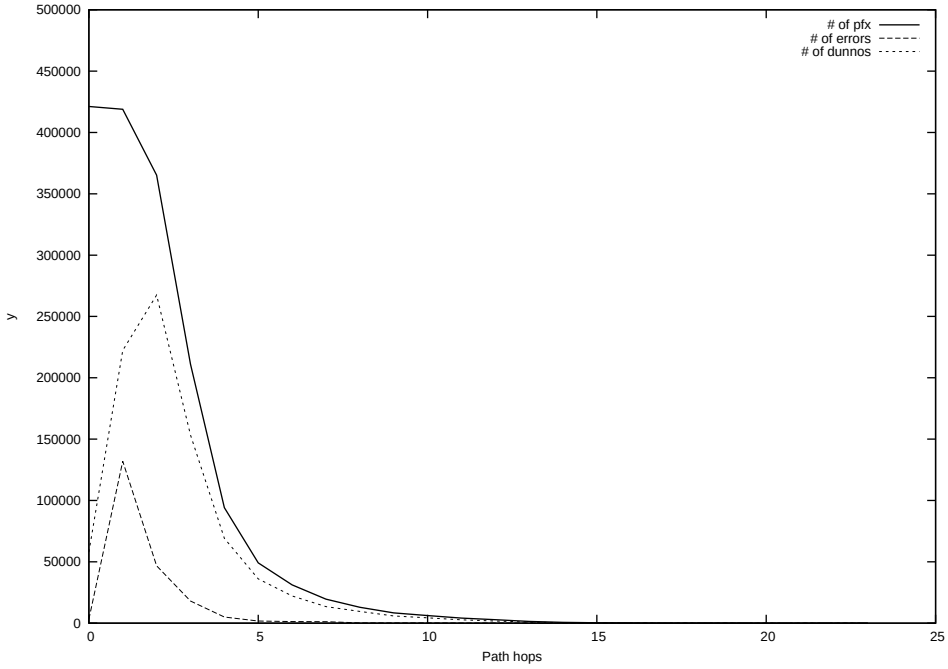
2012-09-05



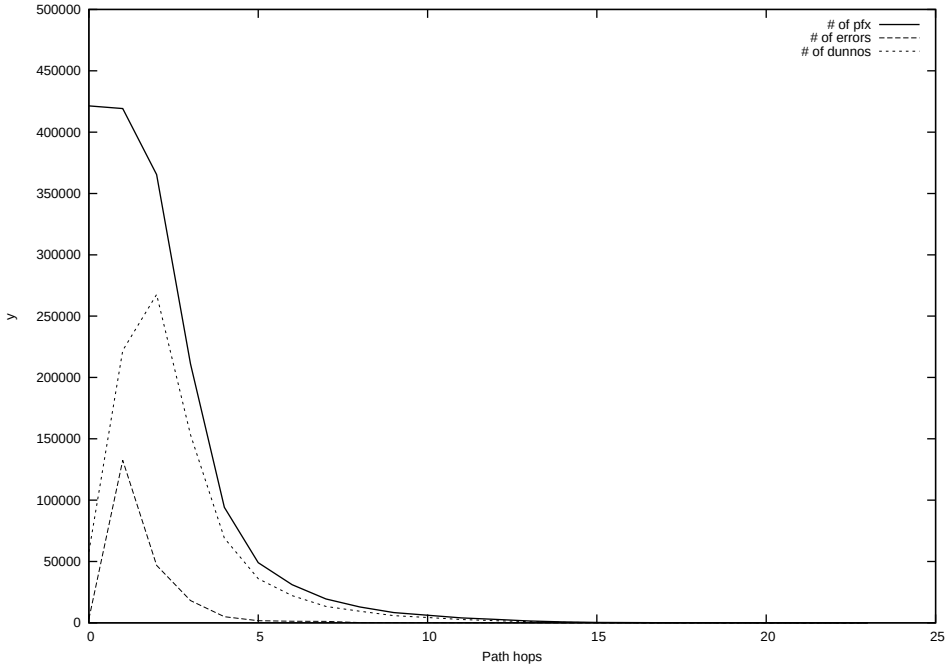
2012-09-06



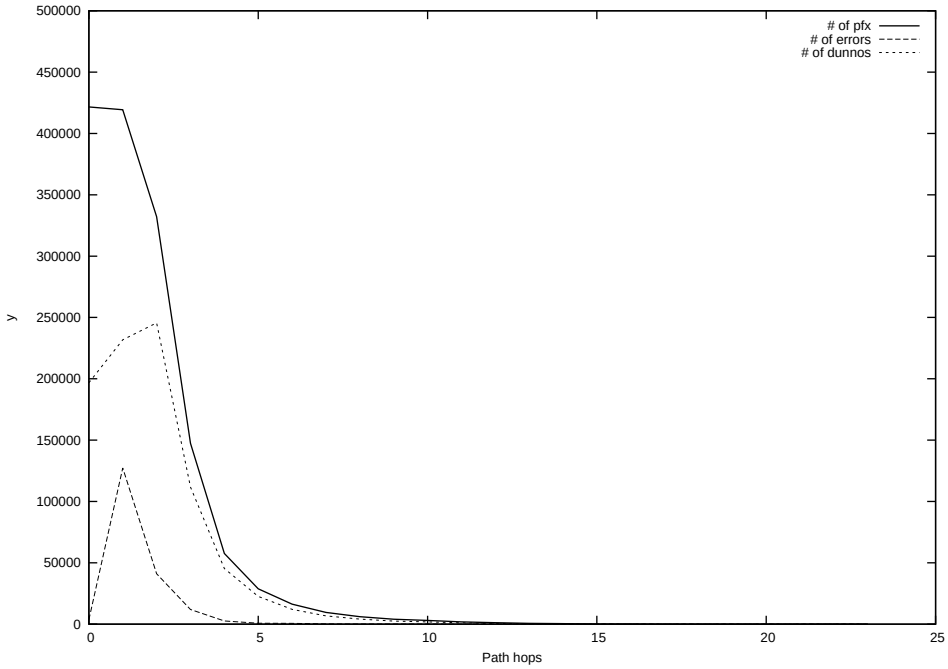
2012-09-07



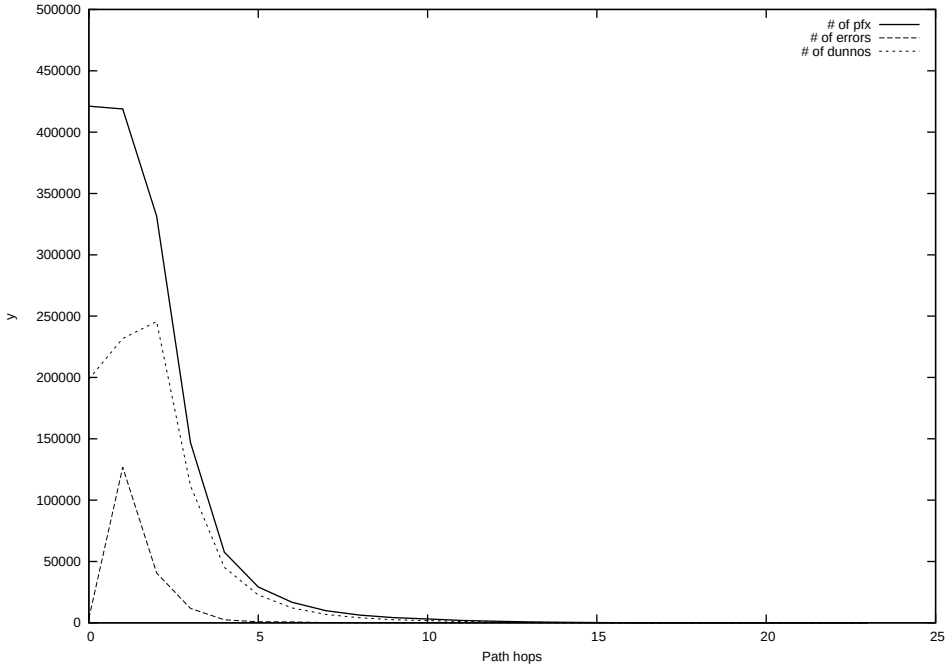
2012-09-08



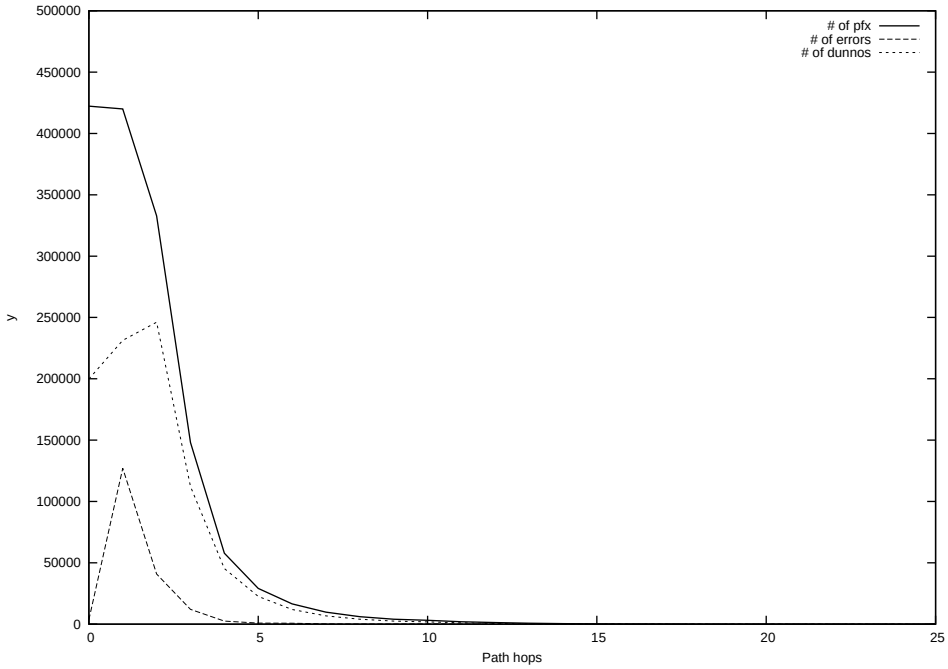
2012-09-09



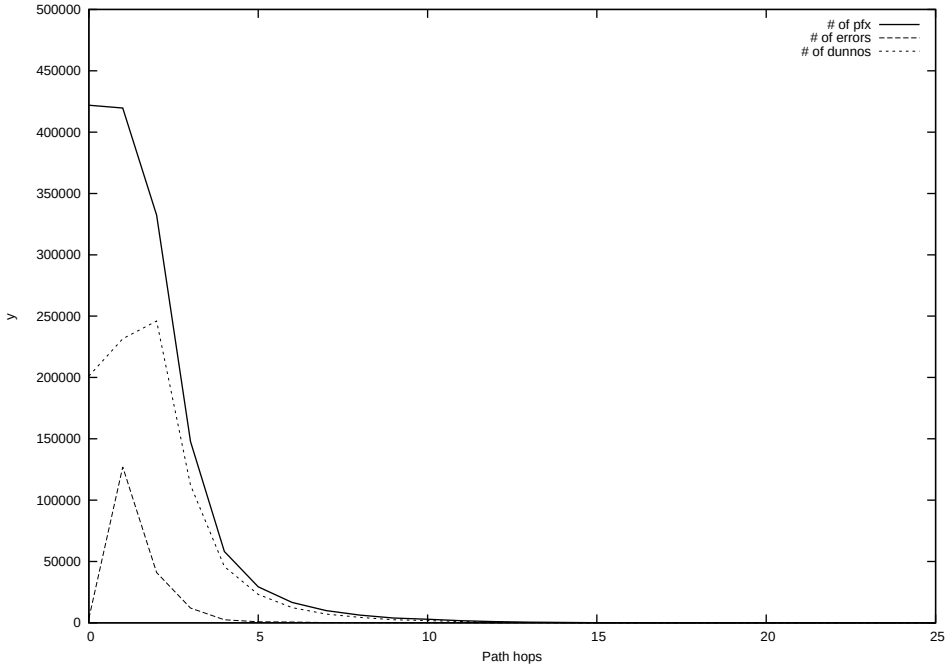
2012-09-10



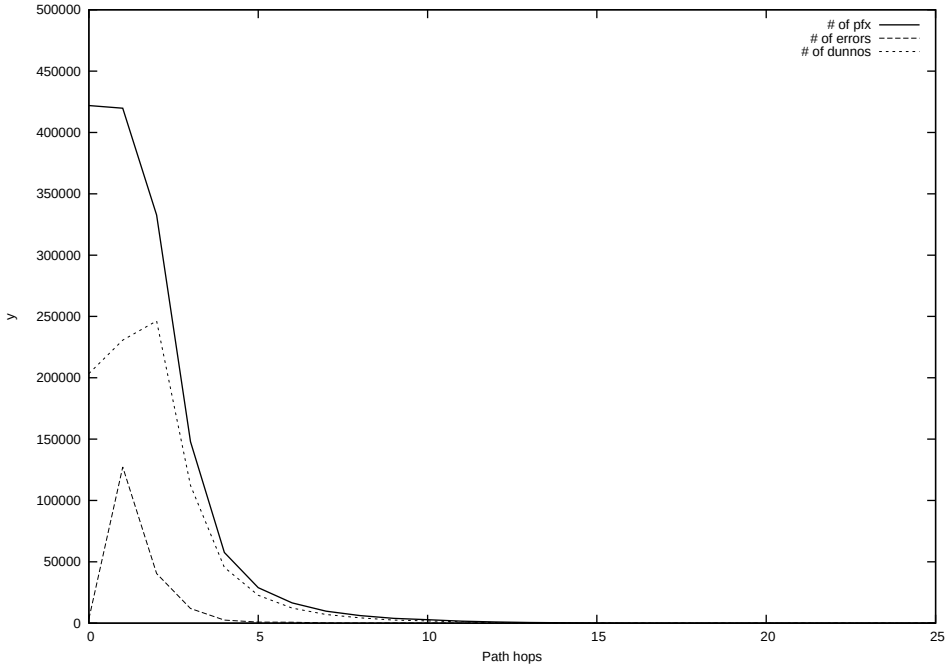
2012-09-11



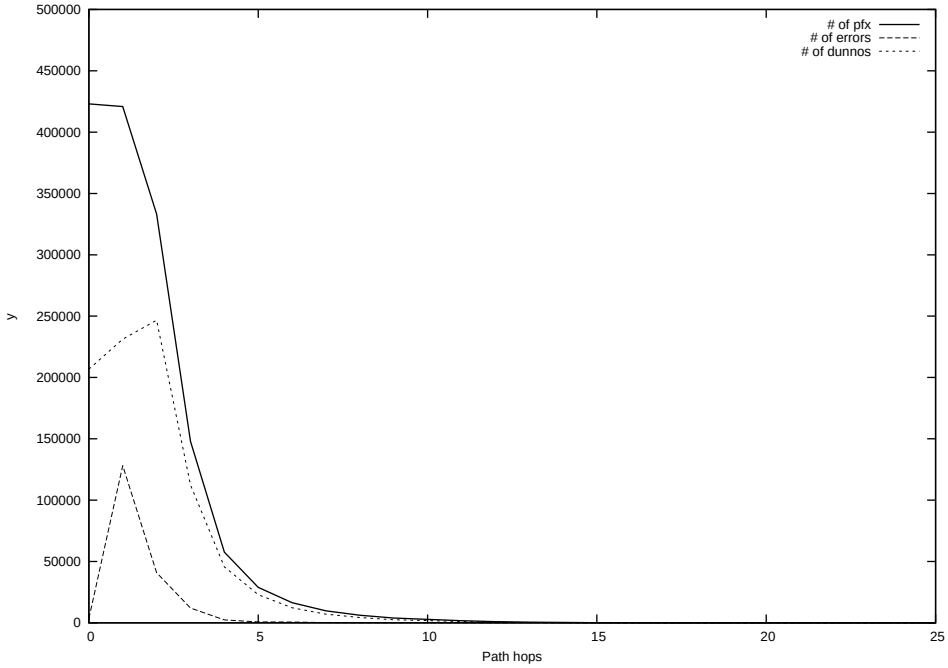
2012-09-12



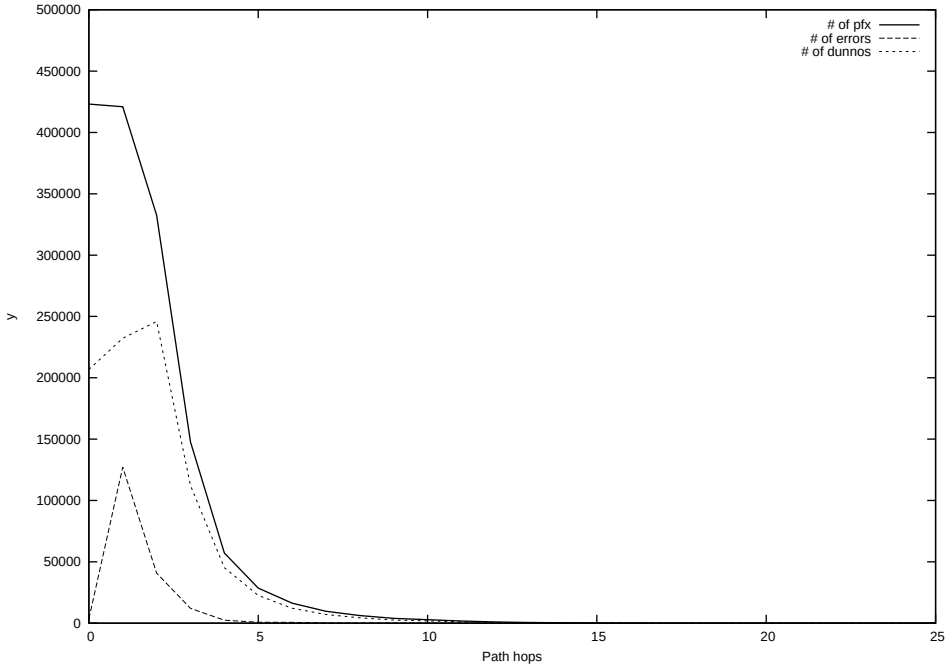
2012-09-13



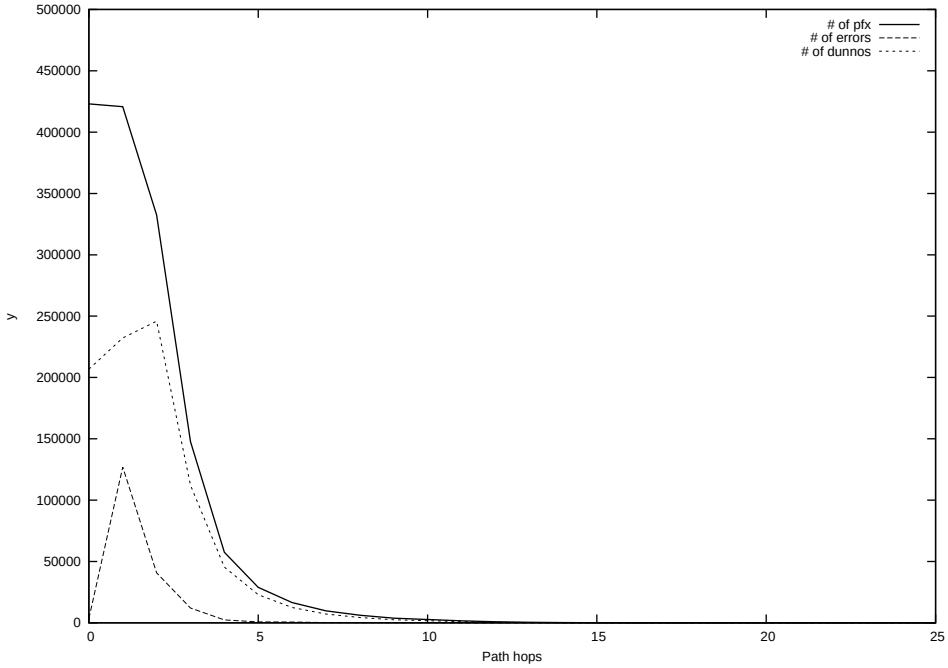
2012-09-14



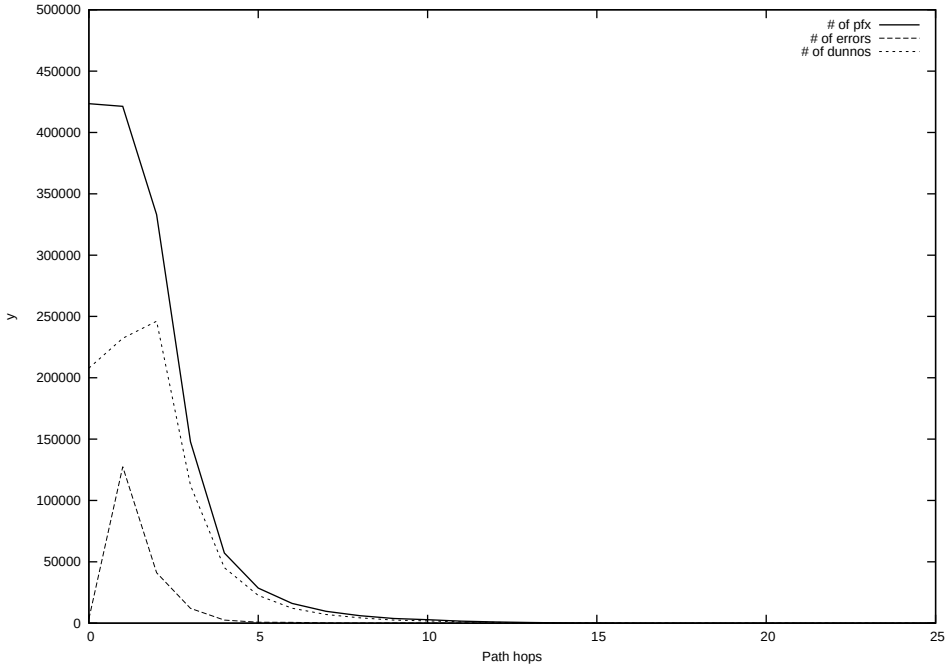
2012-09-15



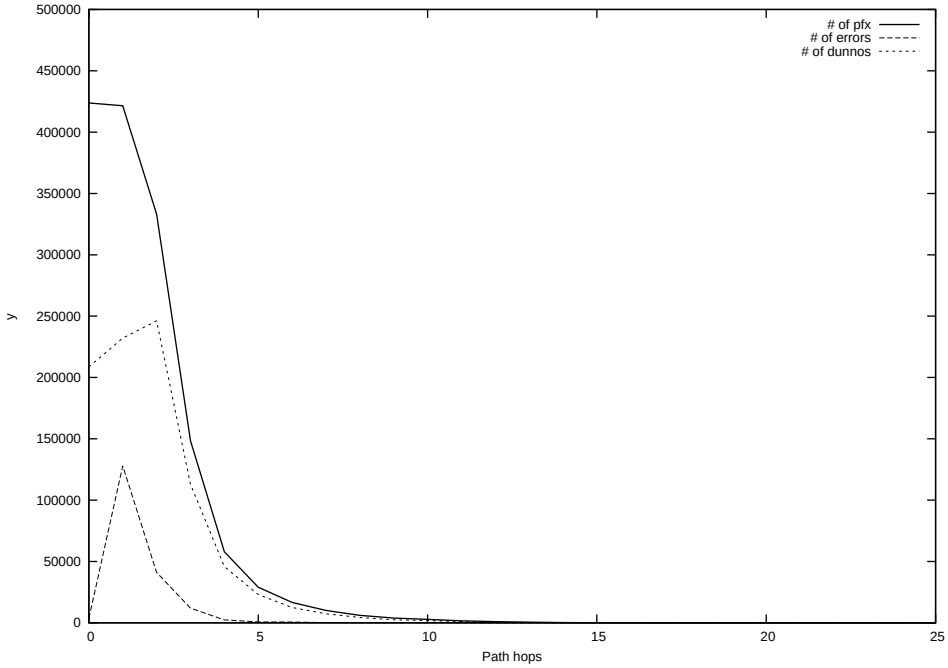
2012-09-16



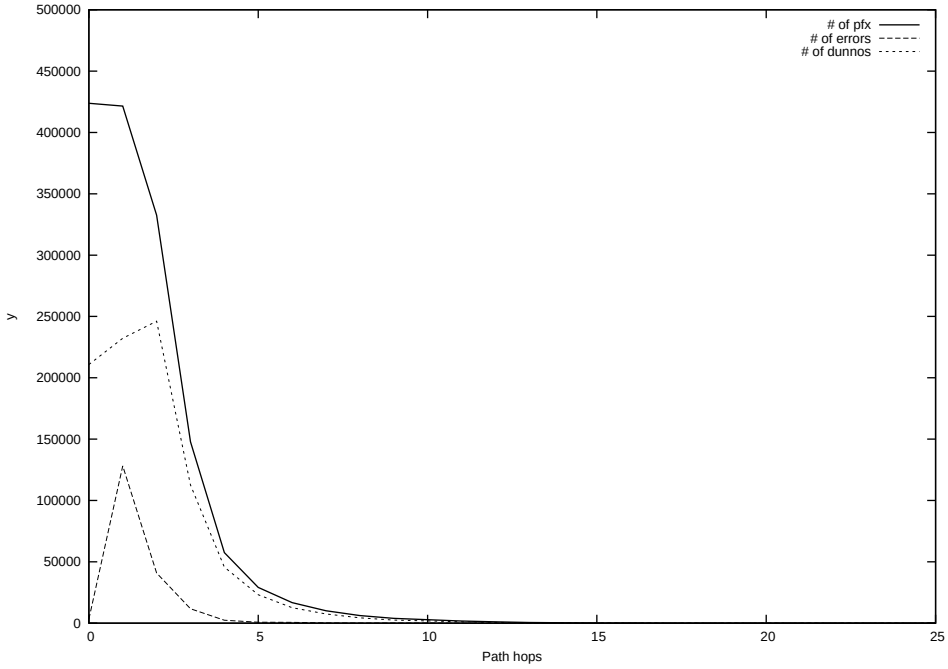
2012-09-17



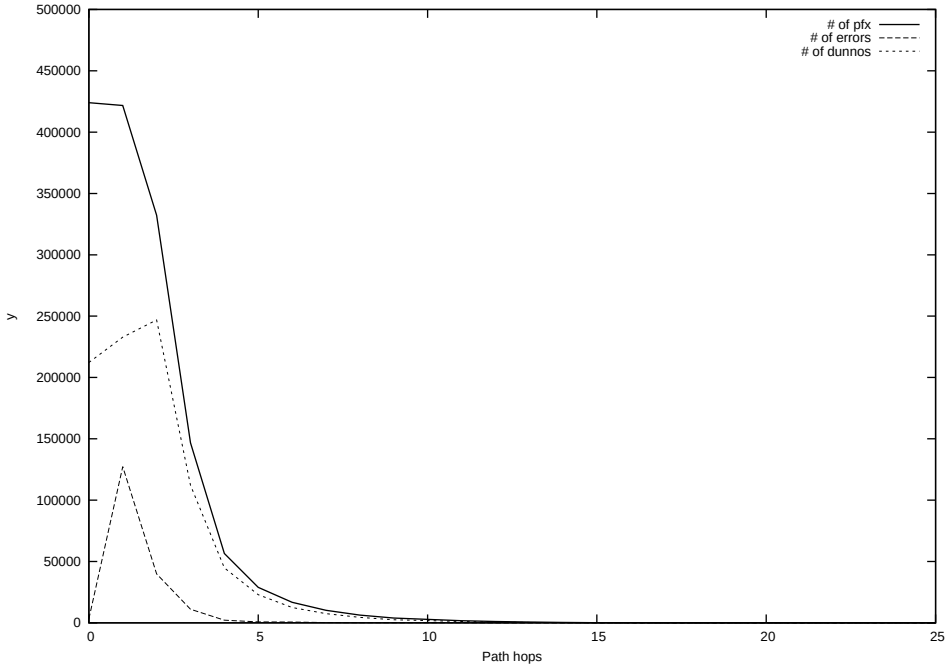
2012-09-18



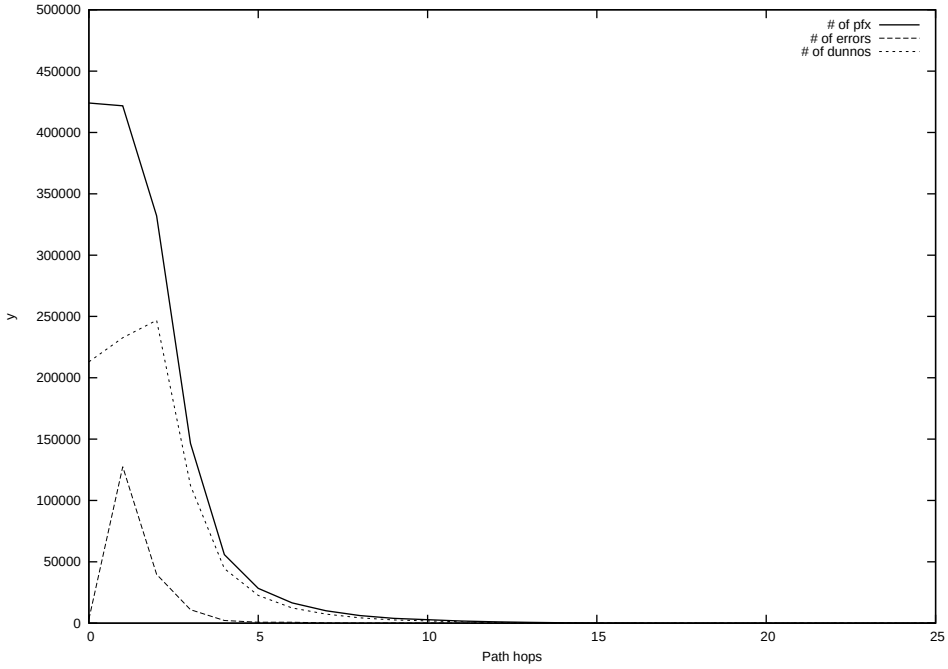
2012-09-19



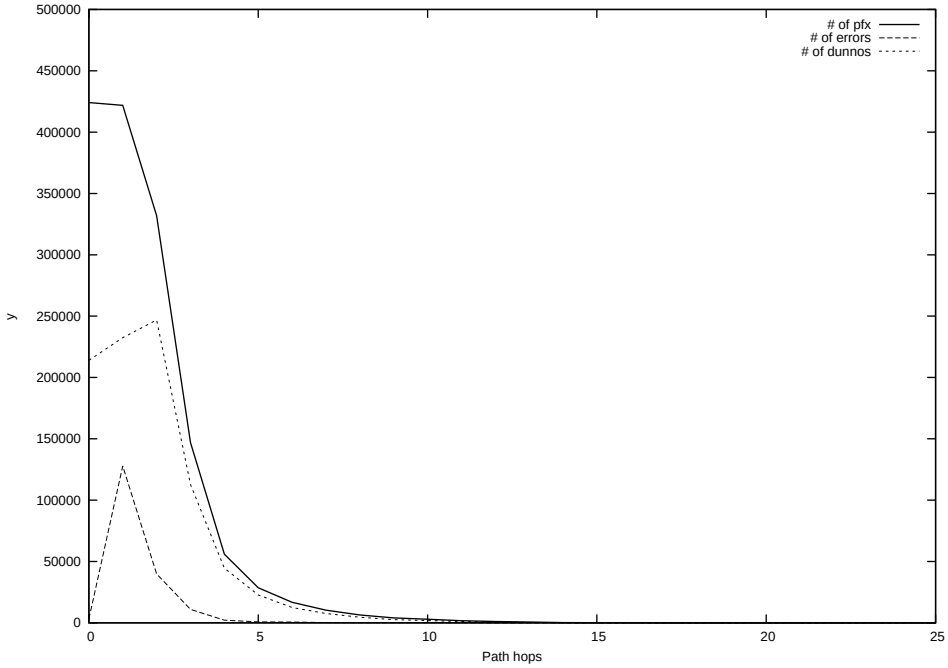
2012-09-20



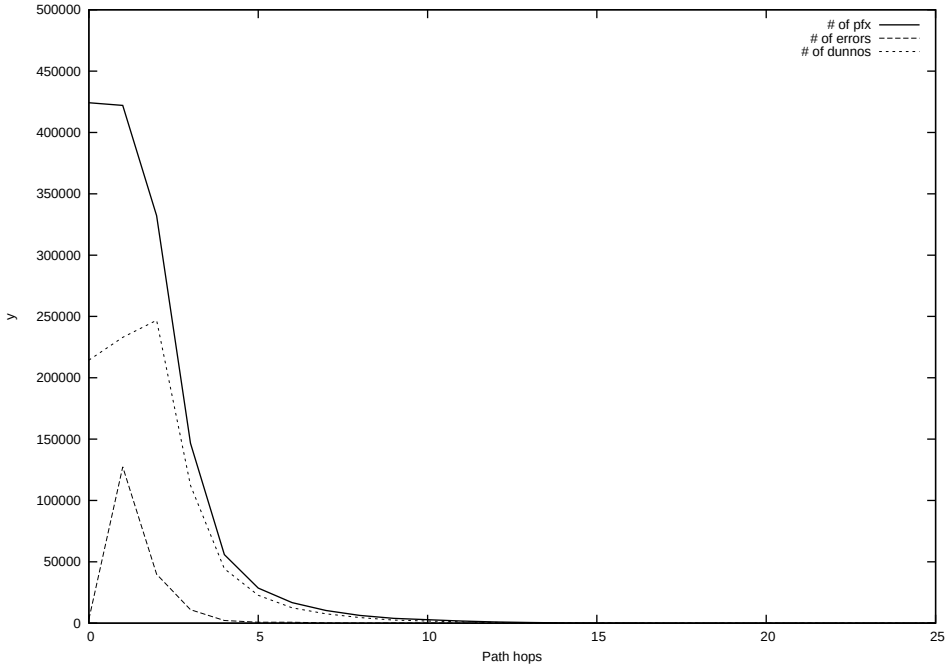
2012-09-21



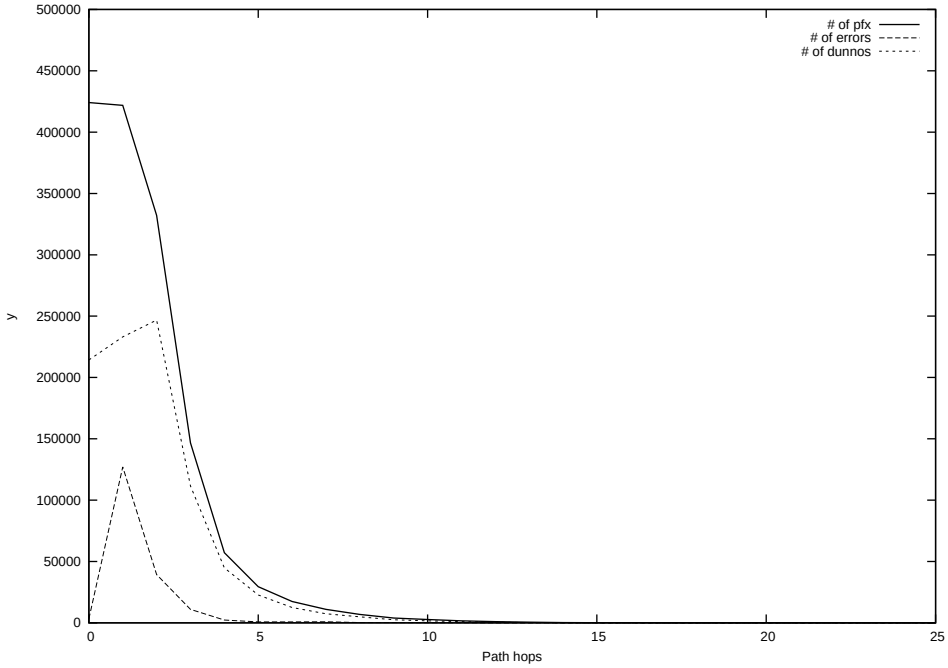
2012-09-22



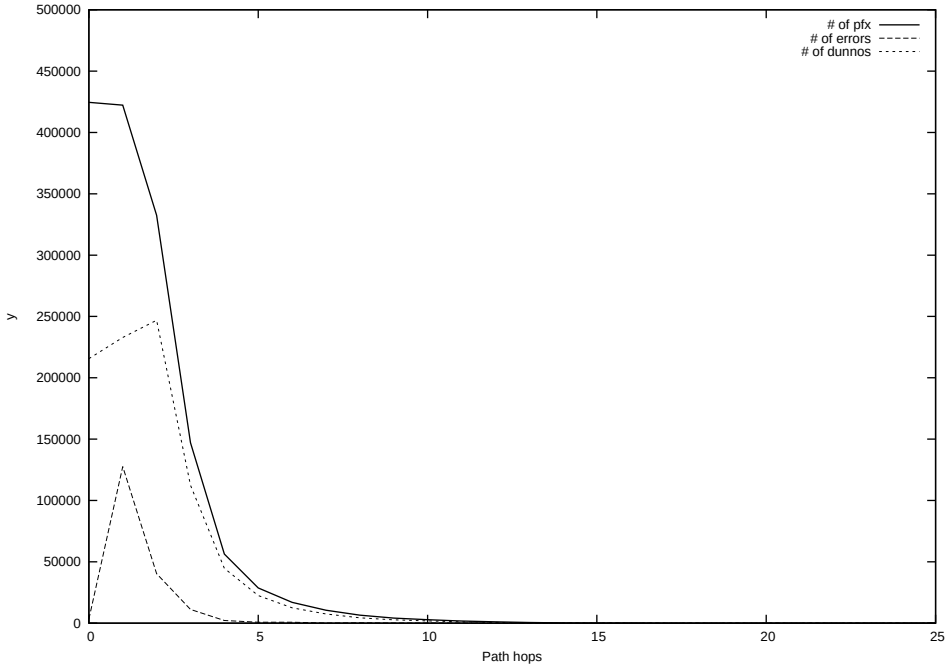
2012-09-23



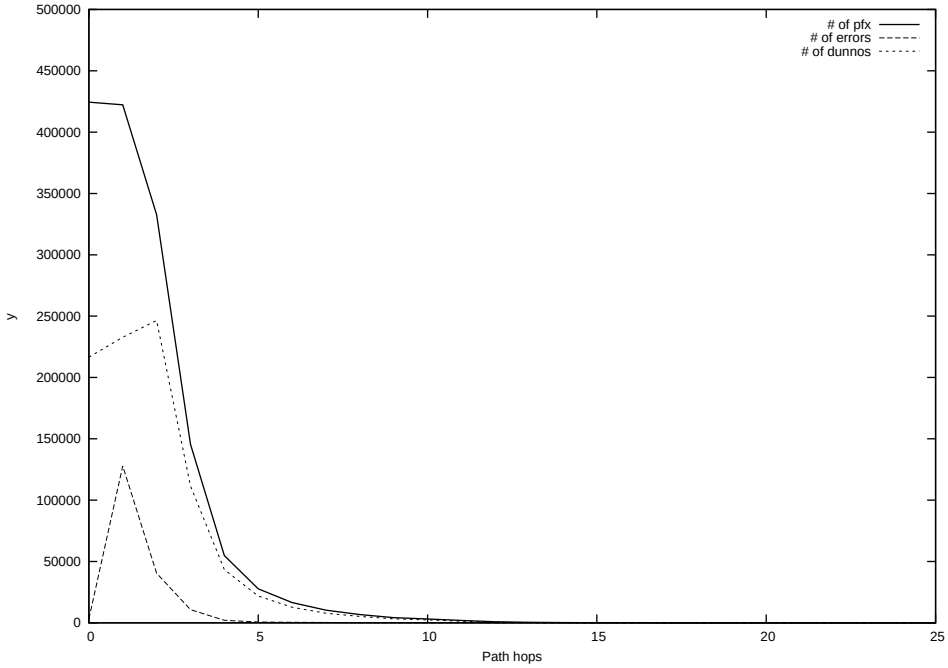
2012-09-24



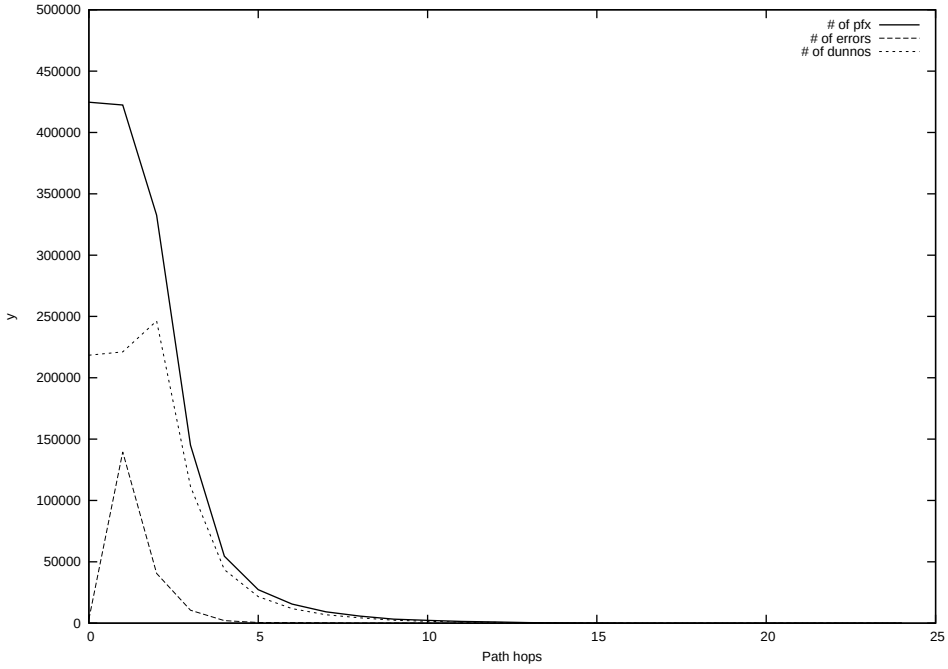
2012-09-25



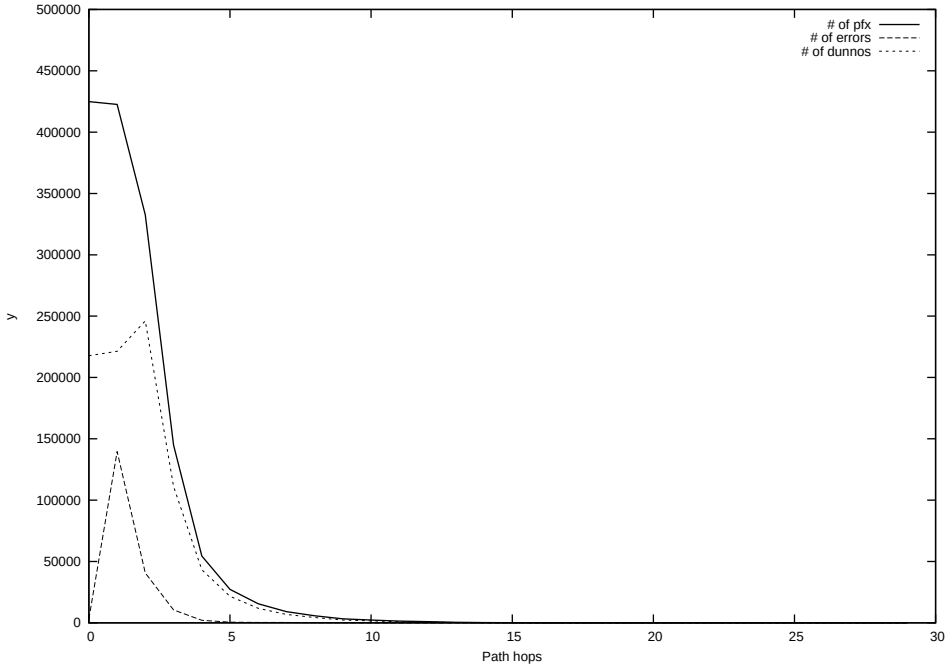
2012-09-26



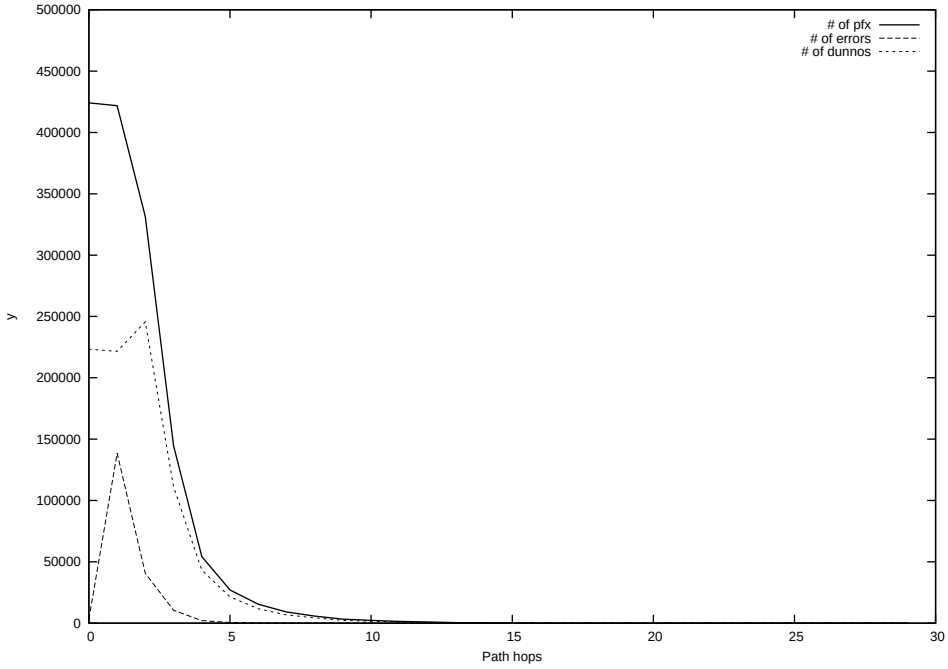
2012-09-27



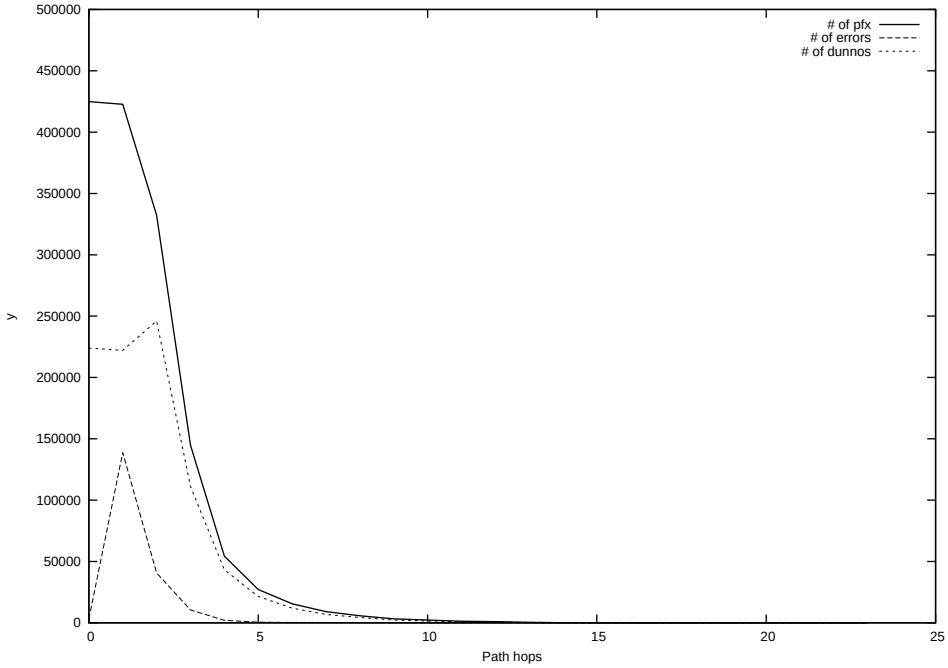
2012-09-28



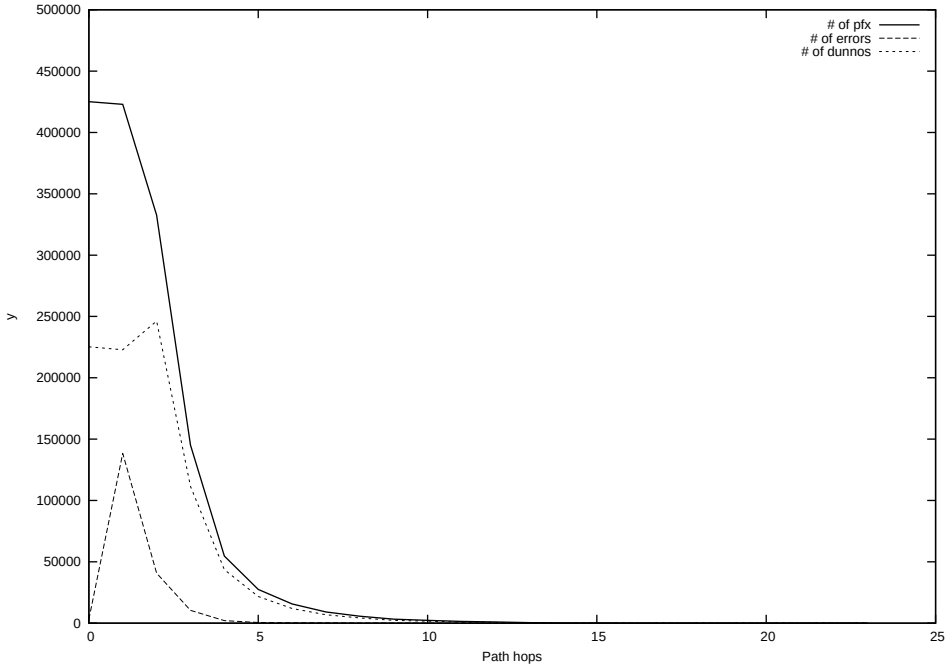
2012-09-29



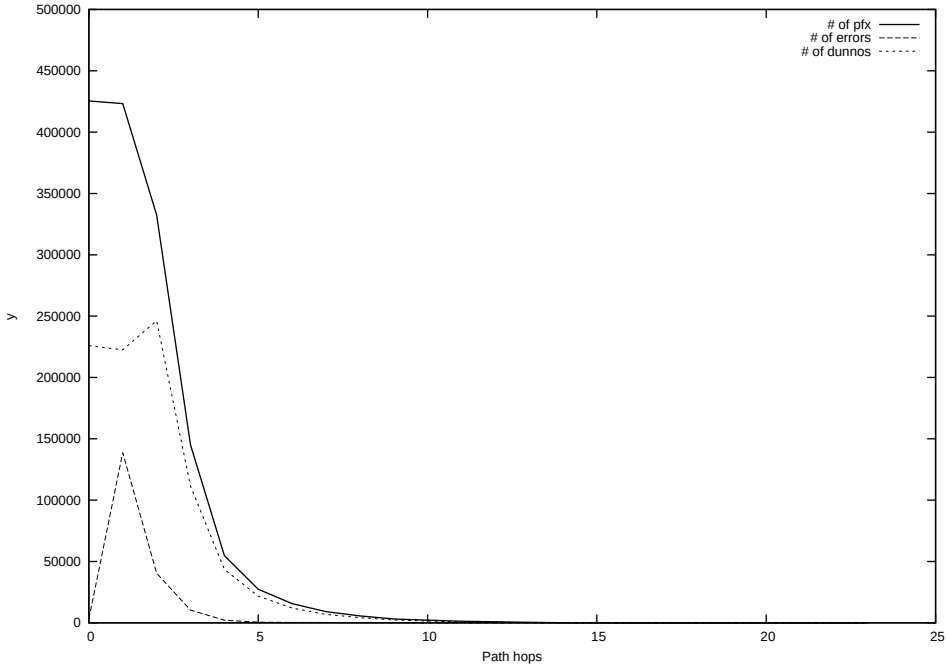
2012-09-30



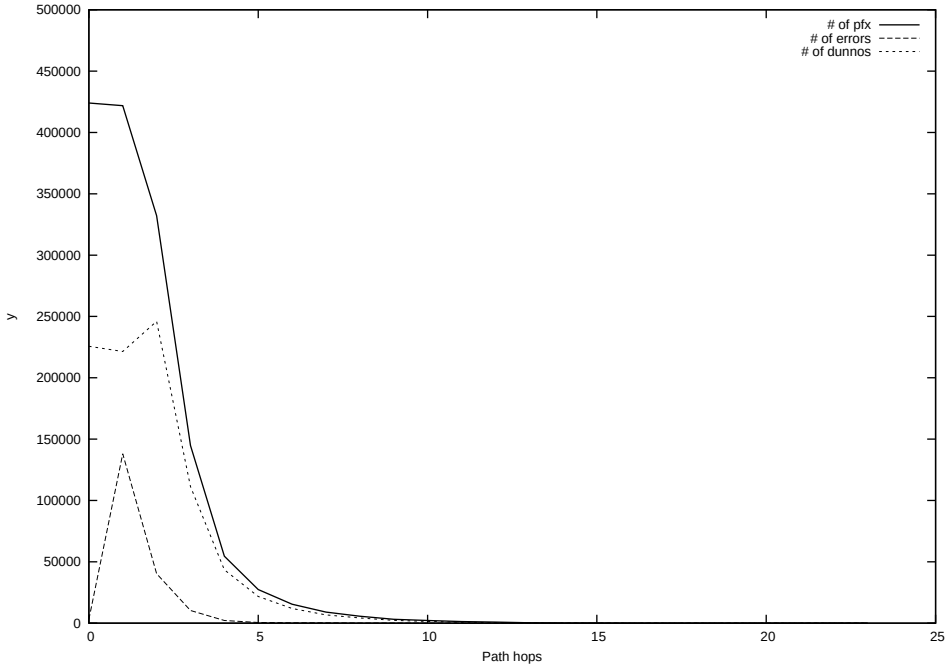
2012-10-01



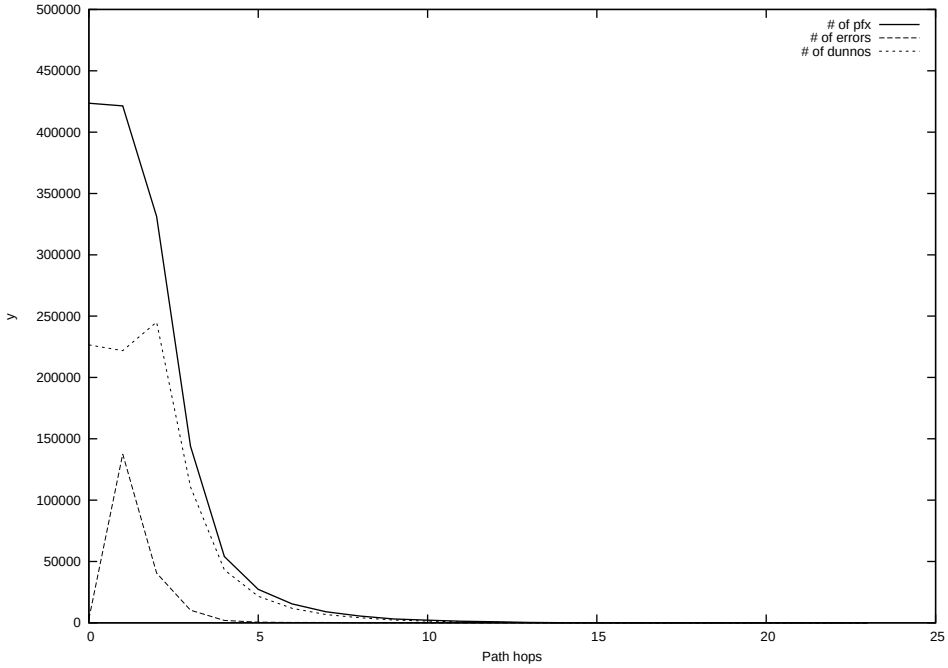
2012-10-02



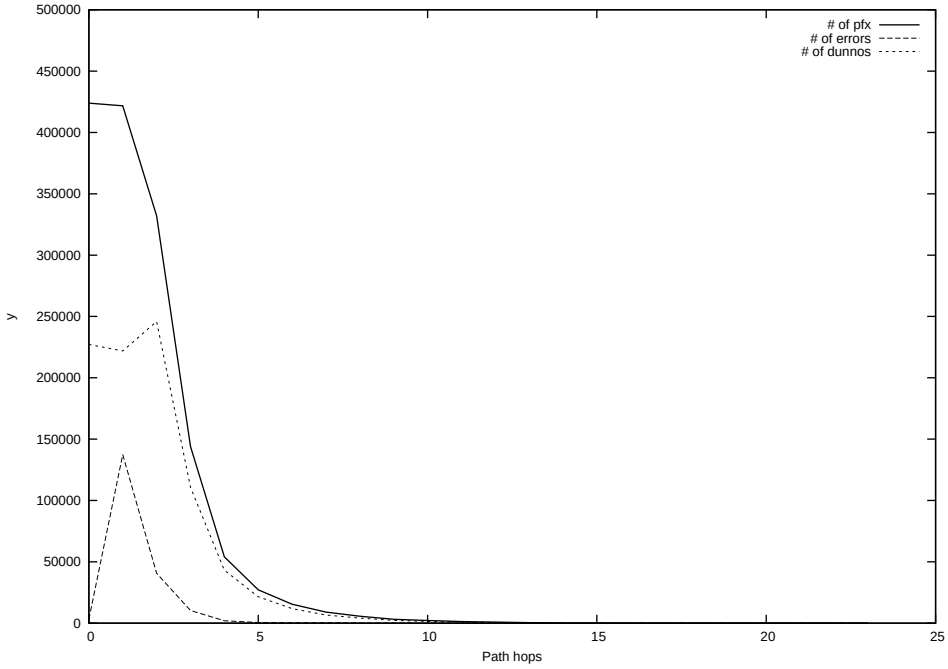
2012-10-03



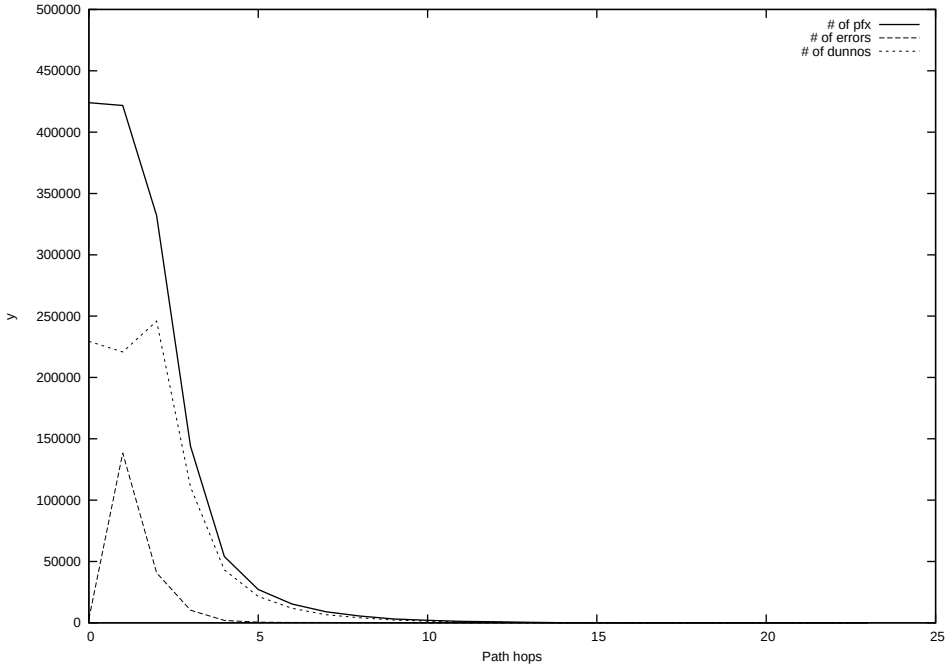
2012-10-04



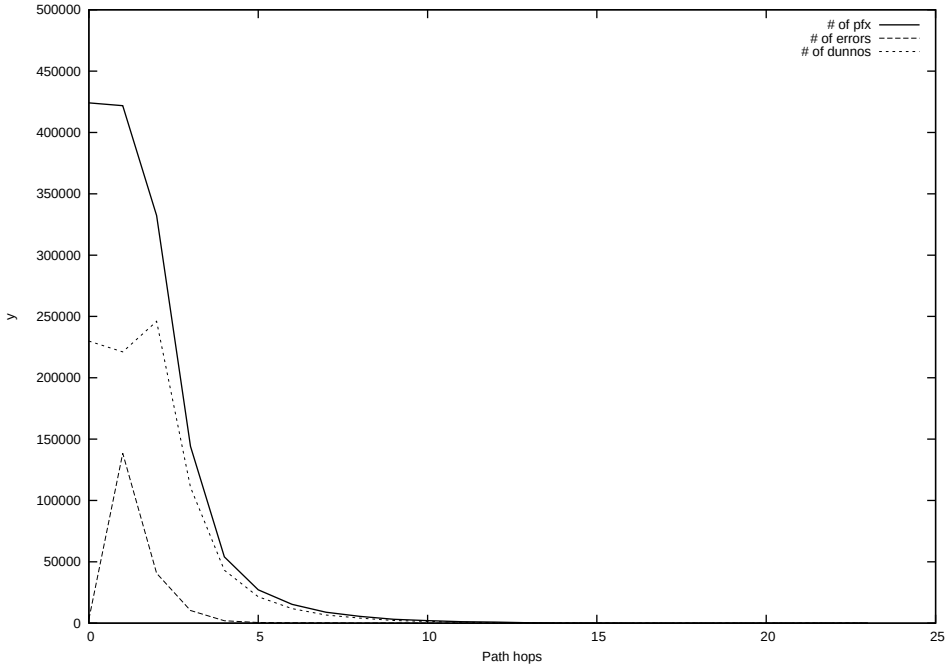
2012-10-05



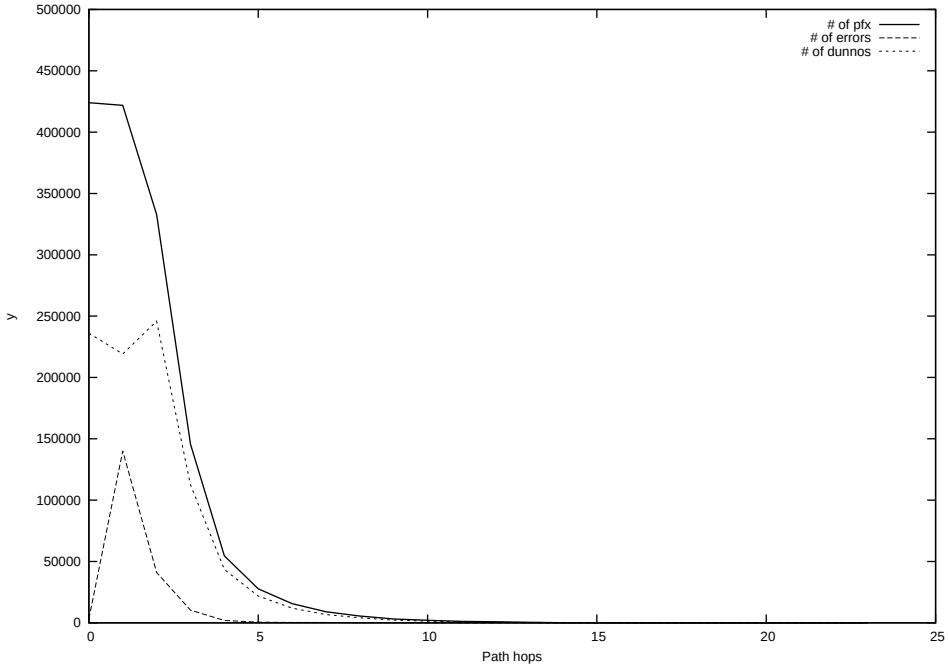
2012-10-06



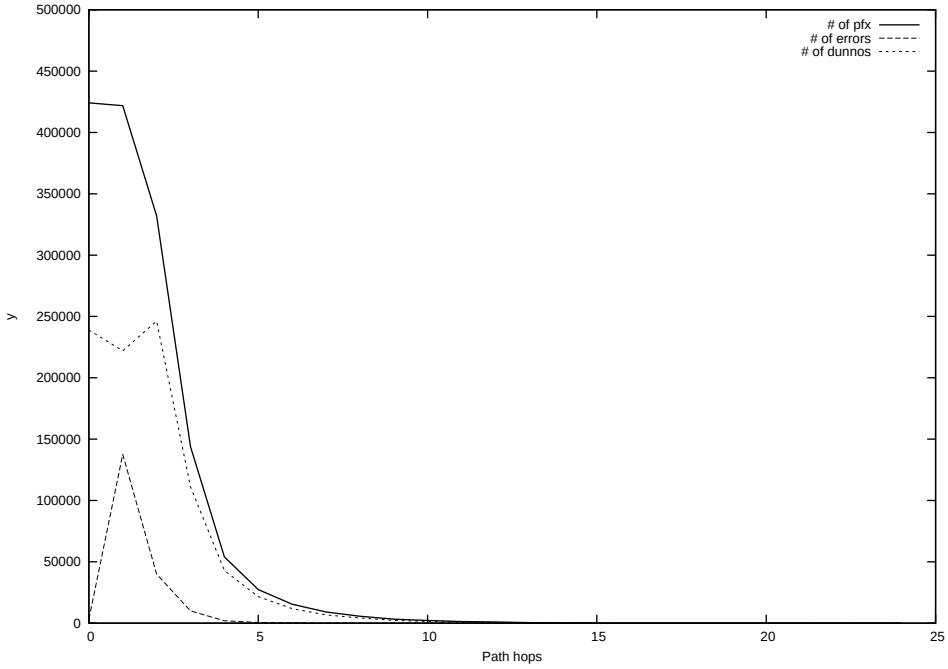
2012-10-07



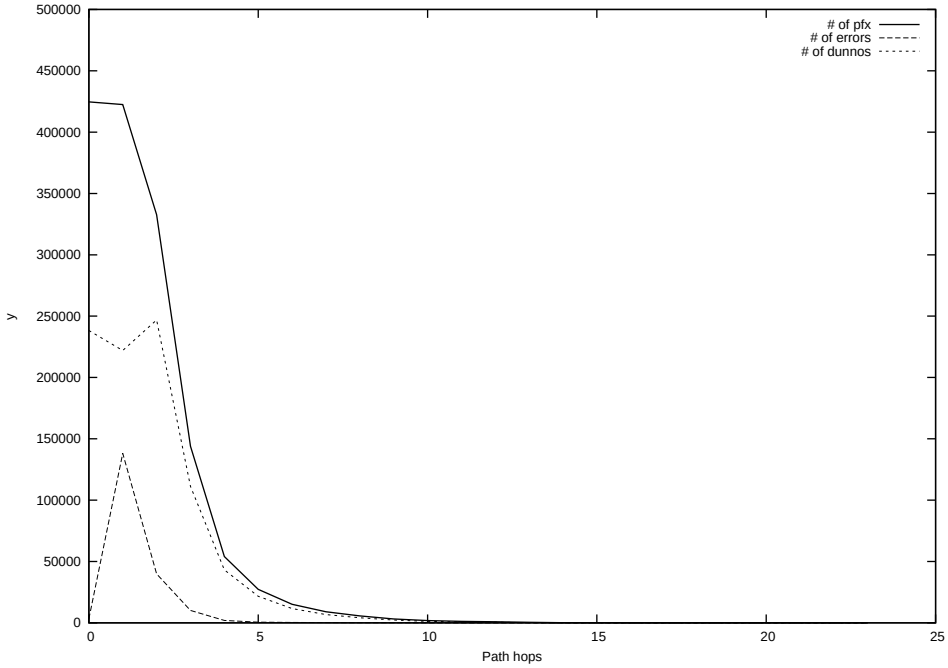
2012-10-08



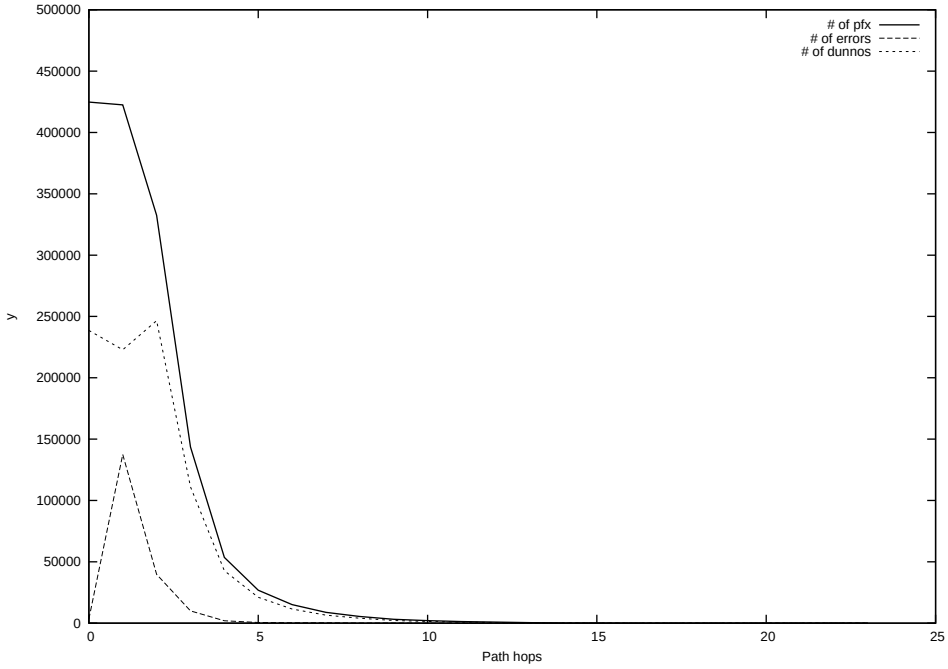
2012-10-09



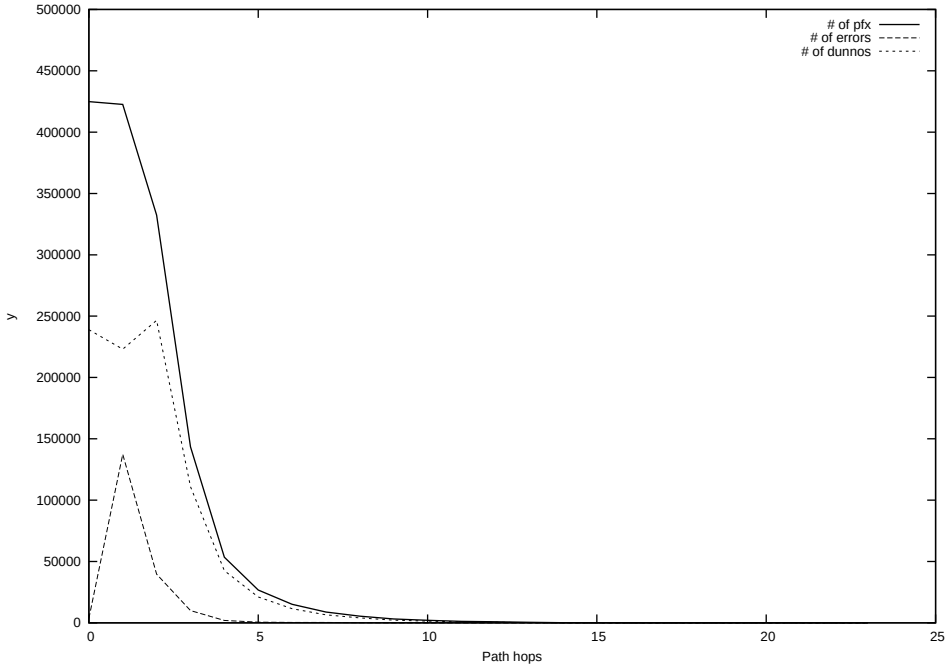
2012-10-10



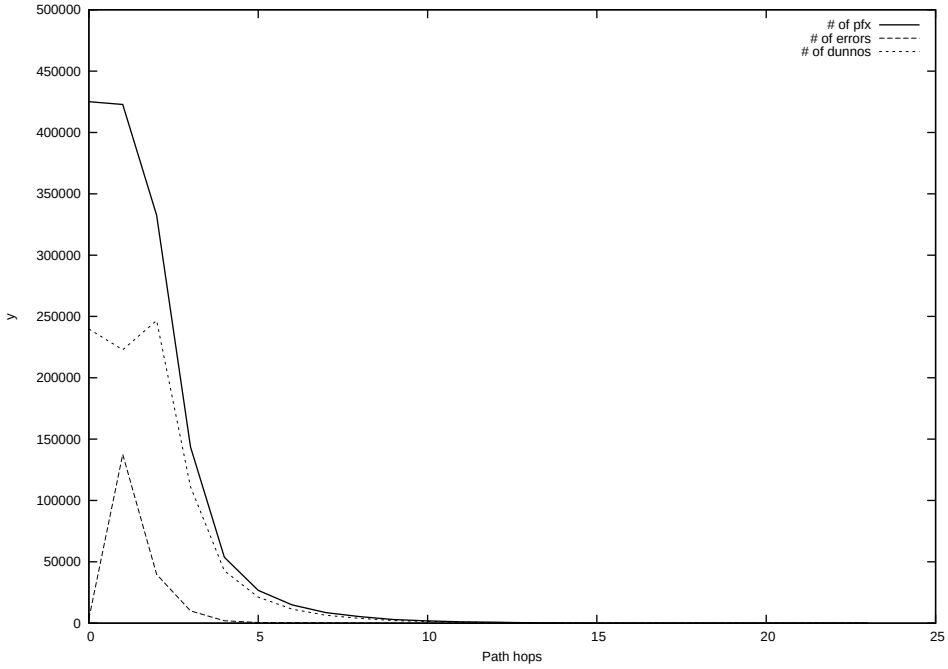
2012-10-11



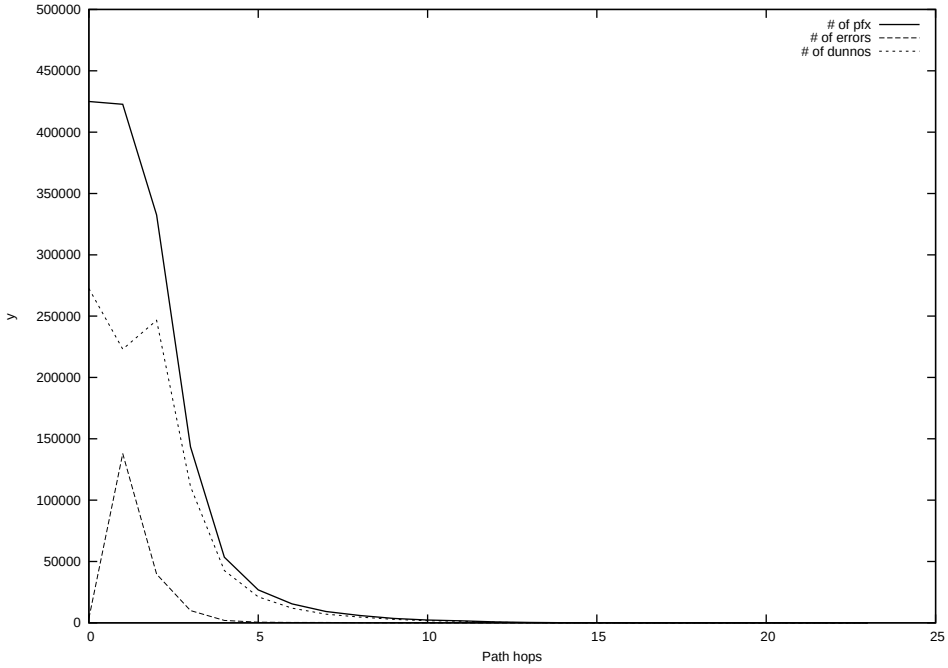
2012-10-12



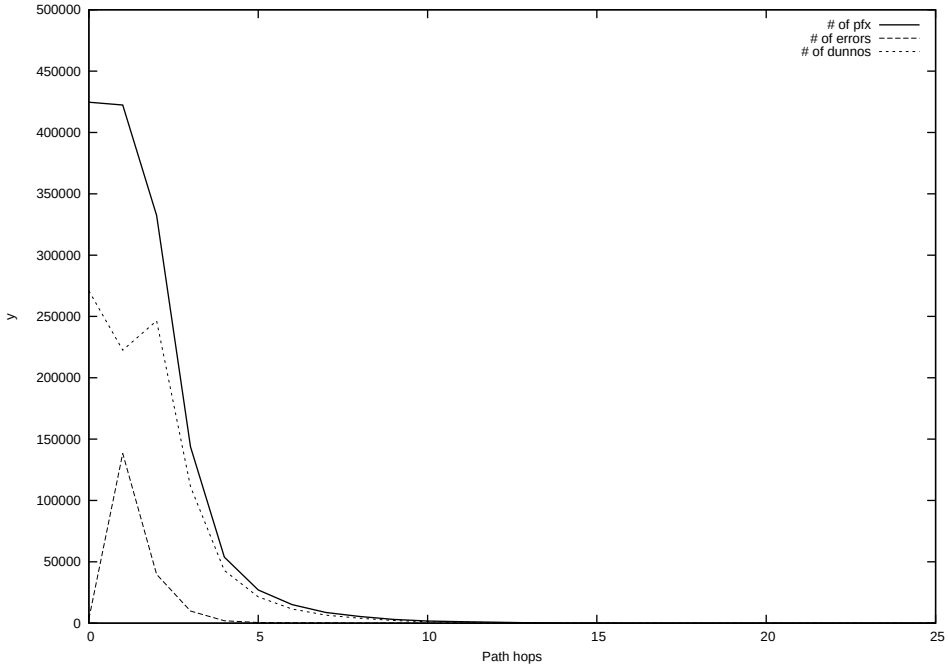
2012-10-13



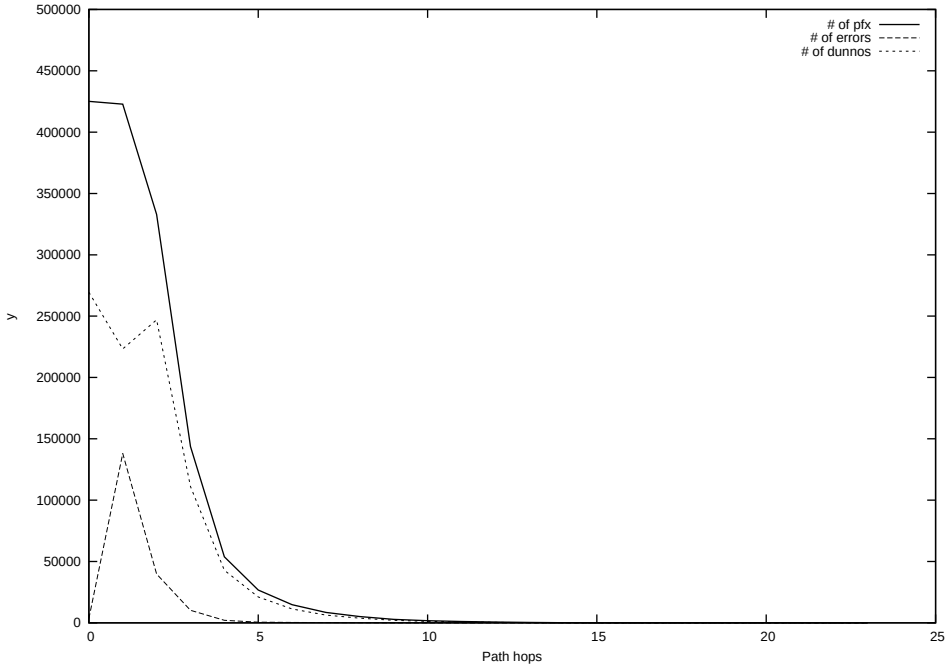
2012-10-14



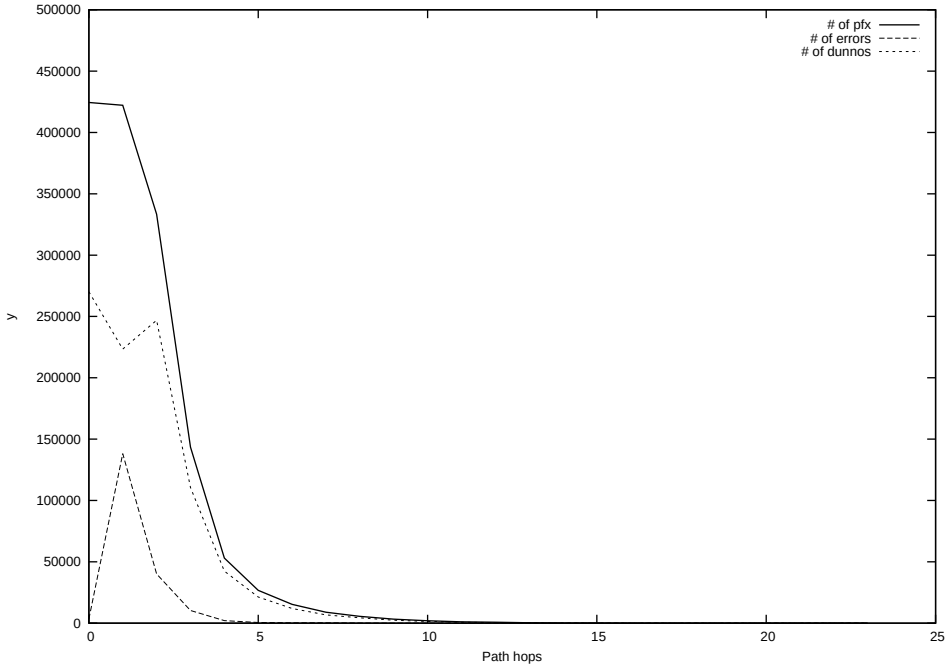
2012-10-15



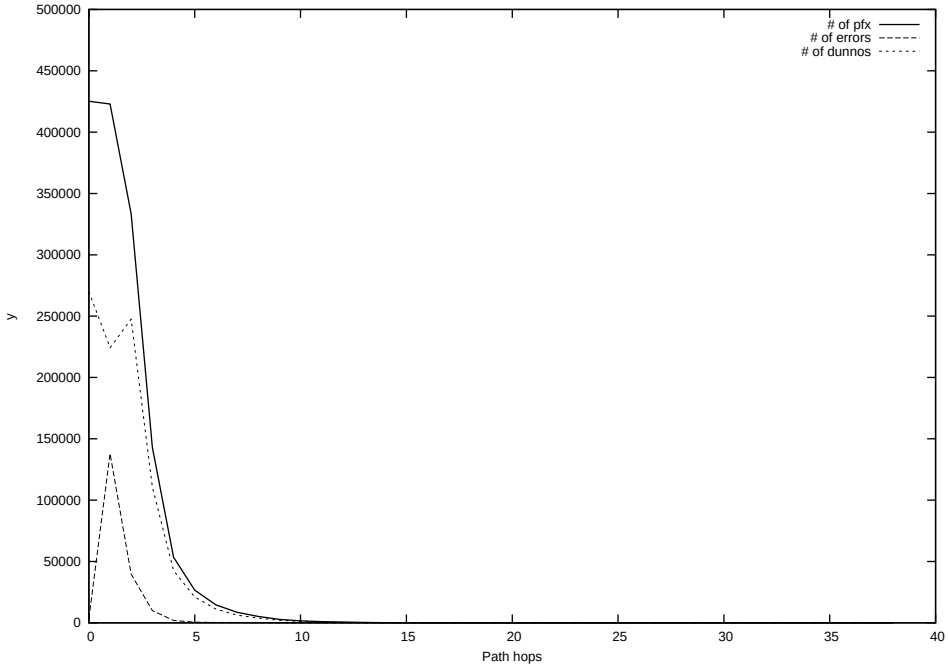
2012-10-16



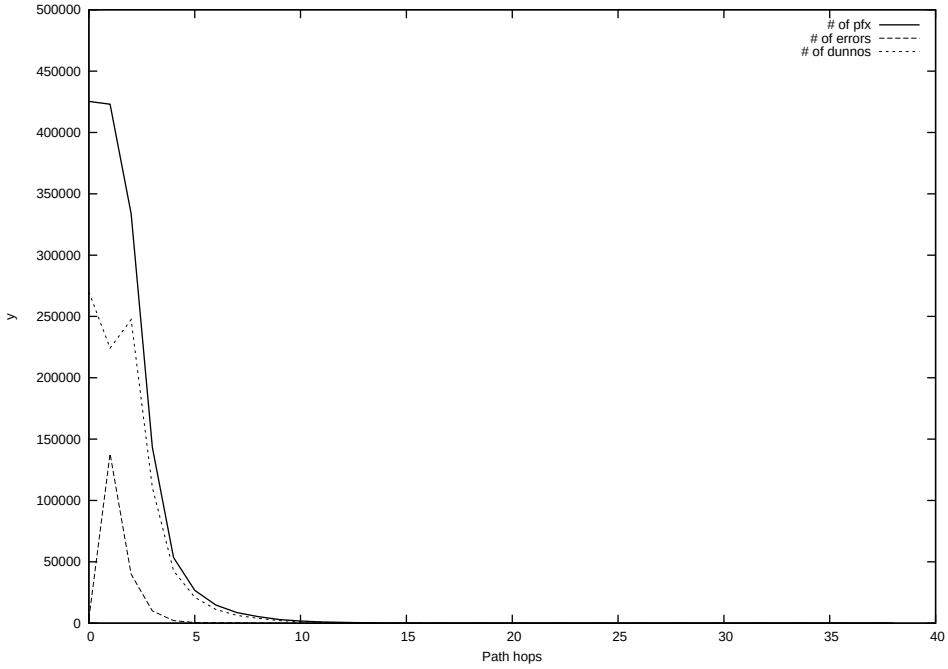
2012-10-18



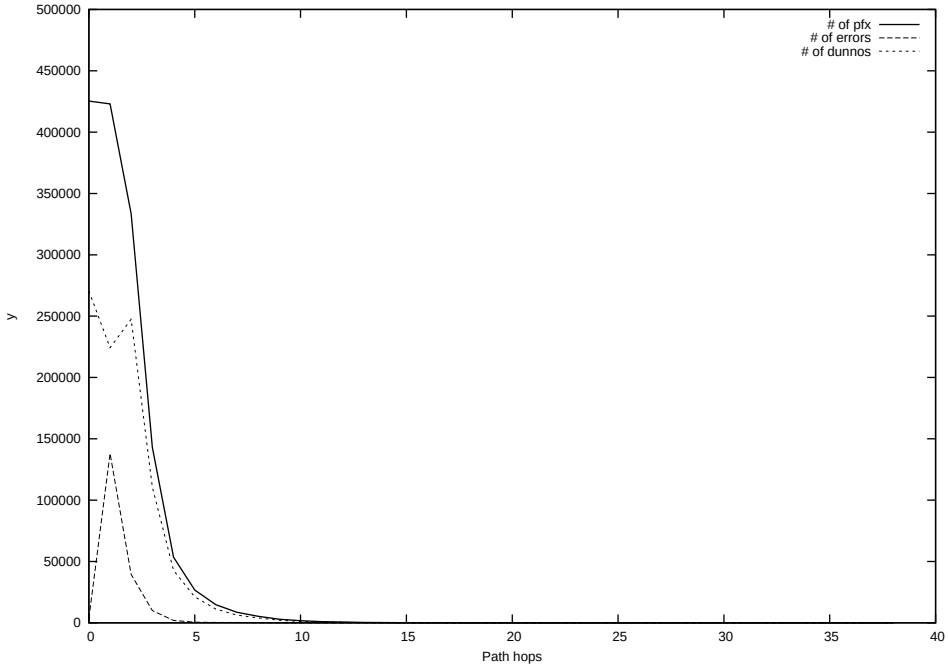
2012-10-19



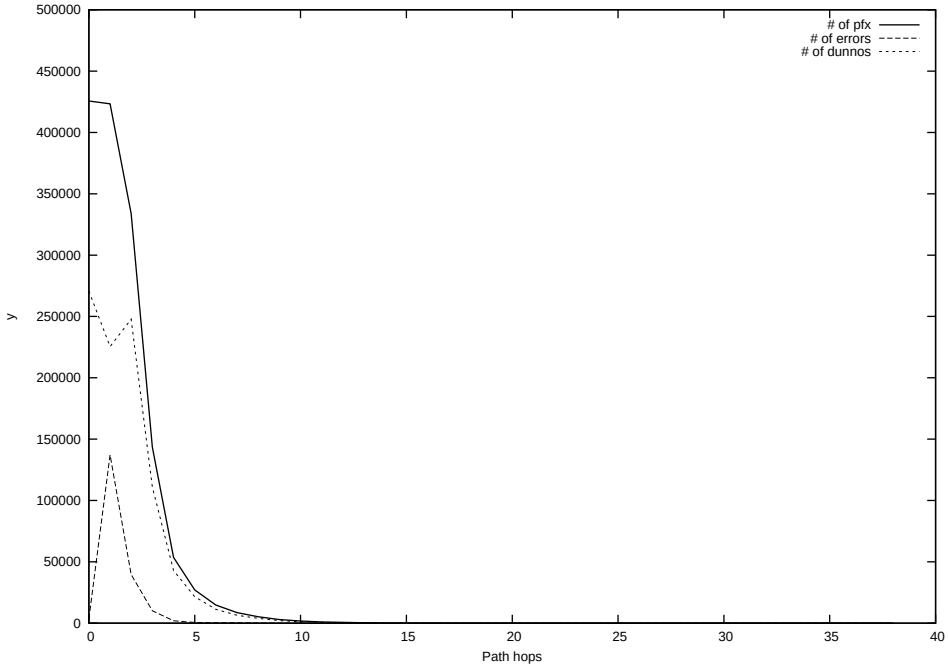
2012-10-20



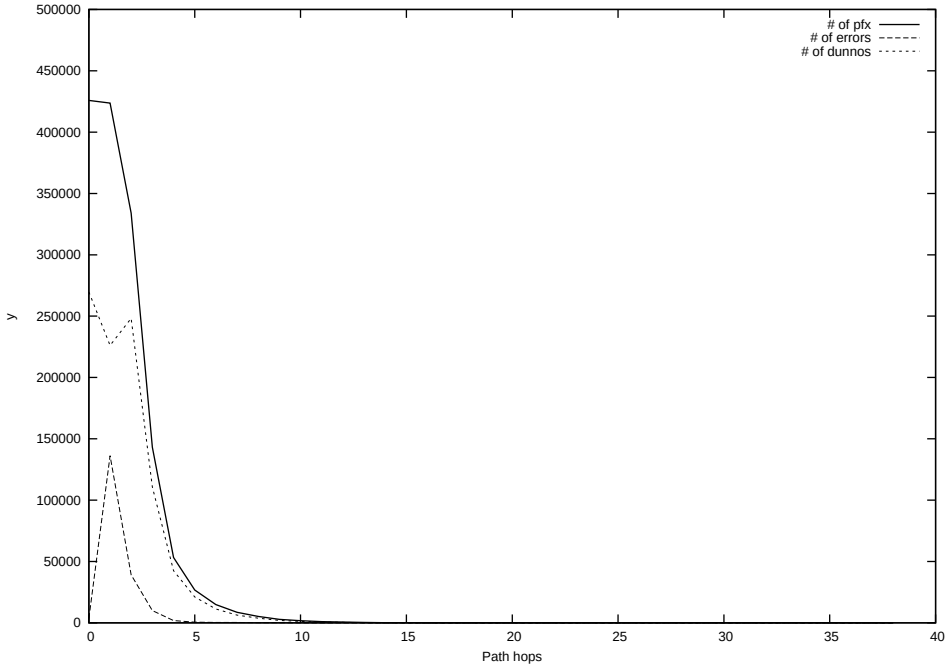
2012-10-21



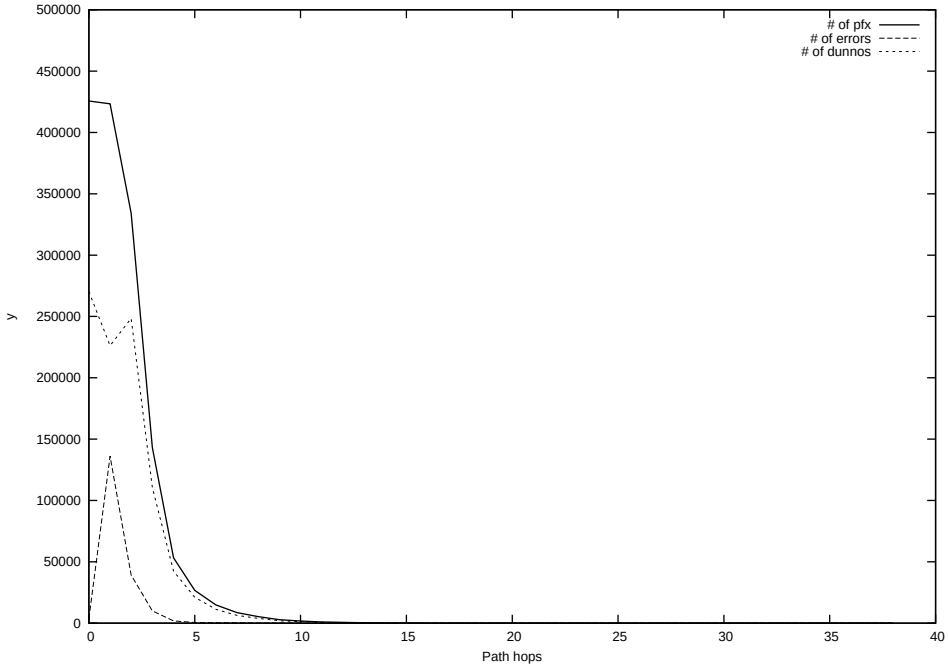
2012-10-22



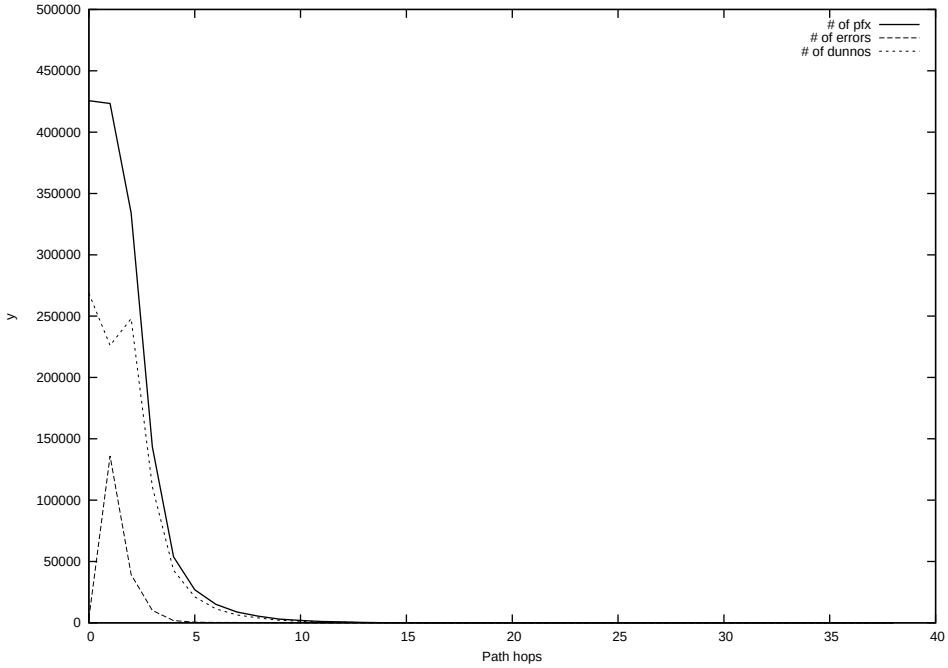
2012-10-23



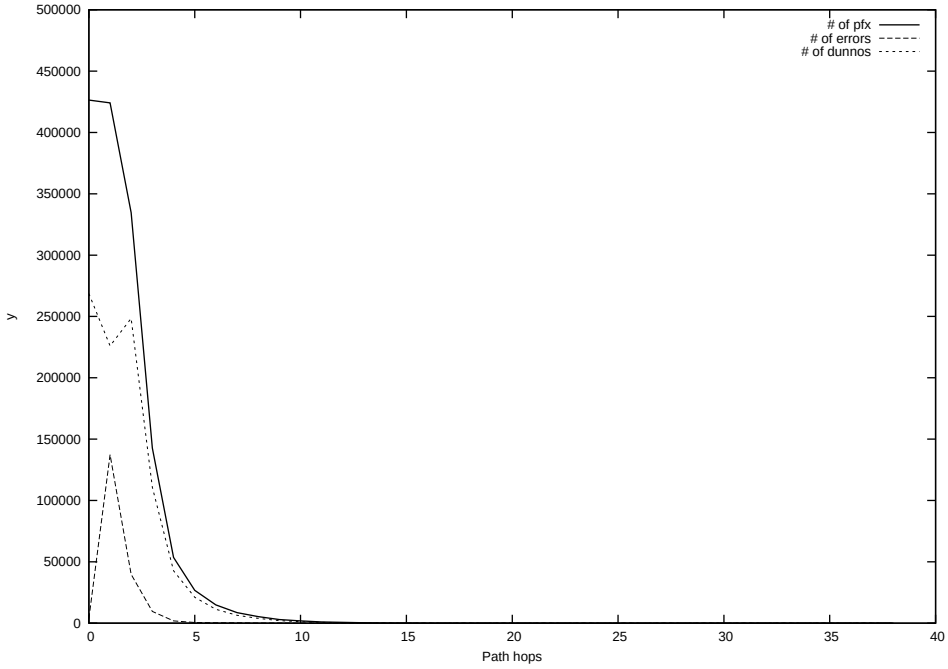
2012-10-24



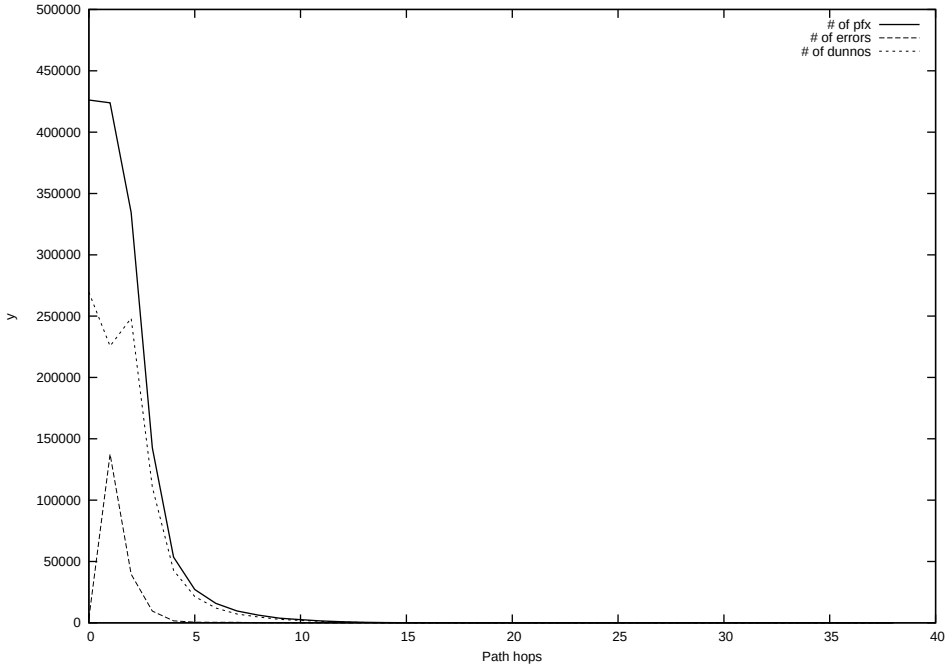
2012-10-25



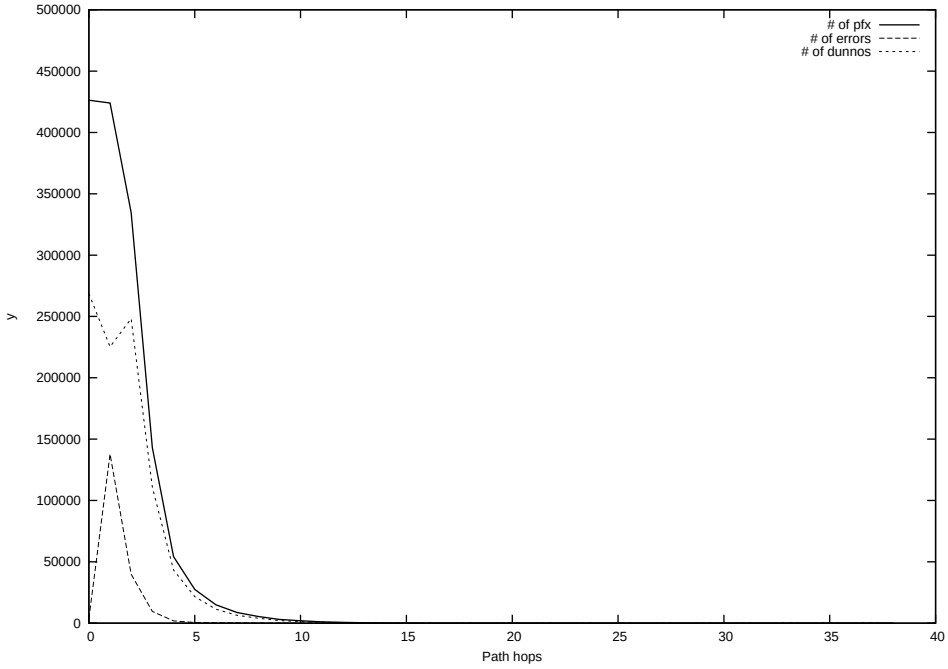
2012-10-26



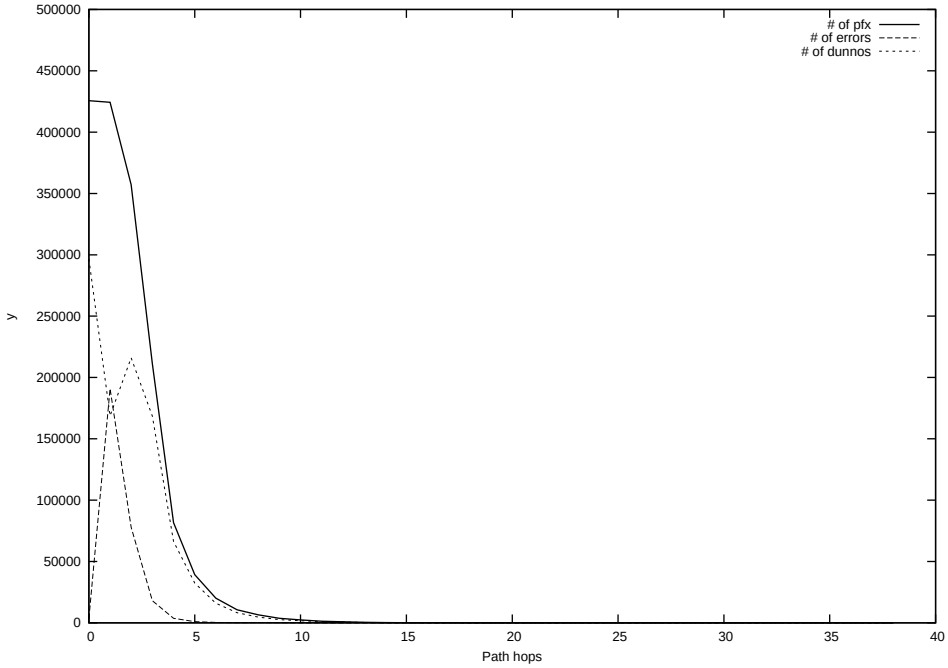
2012-10-27



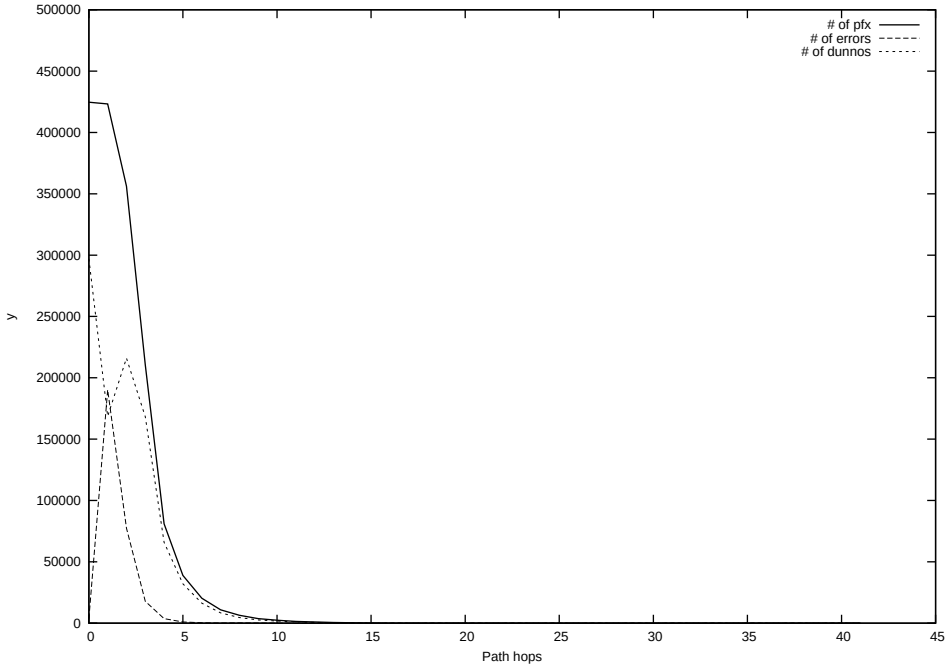
2012-10-28



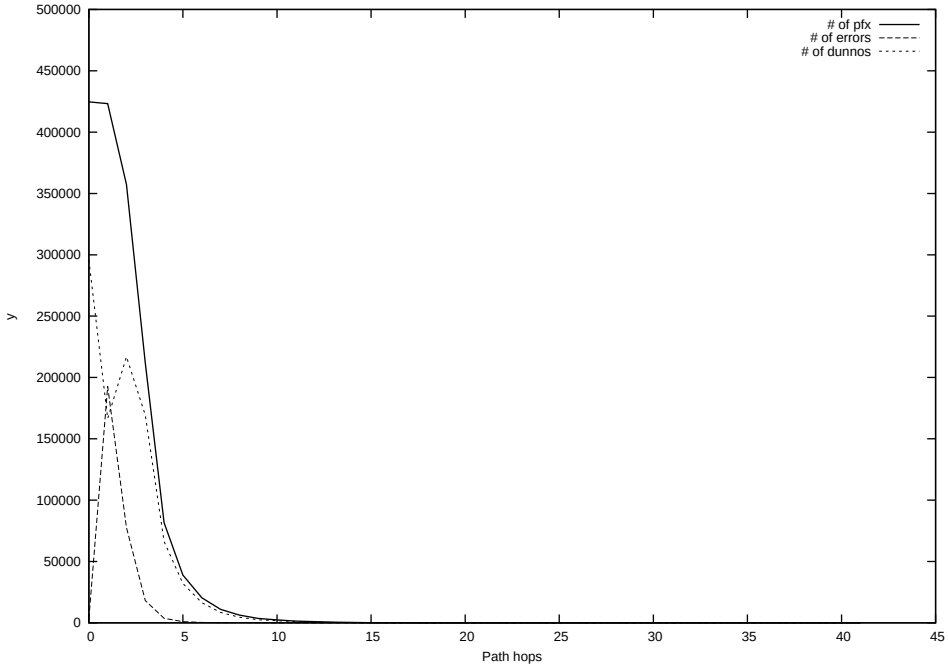
2012-10-29



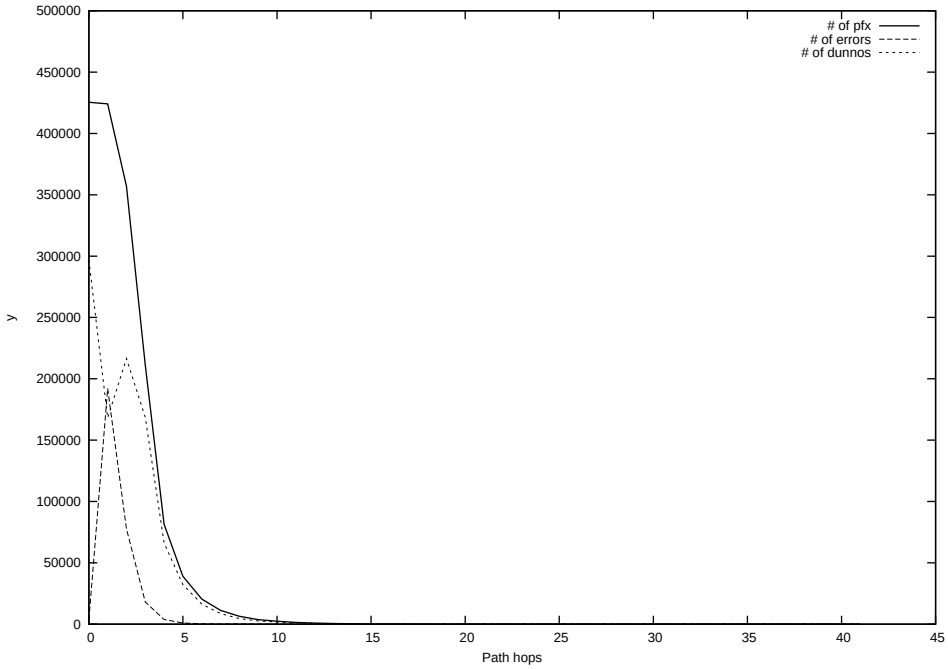
2012-10-30



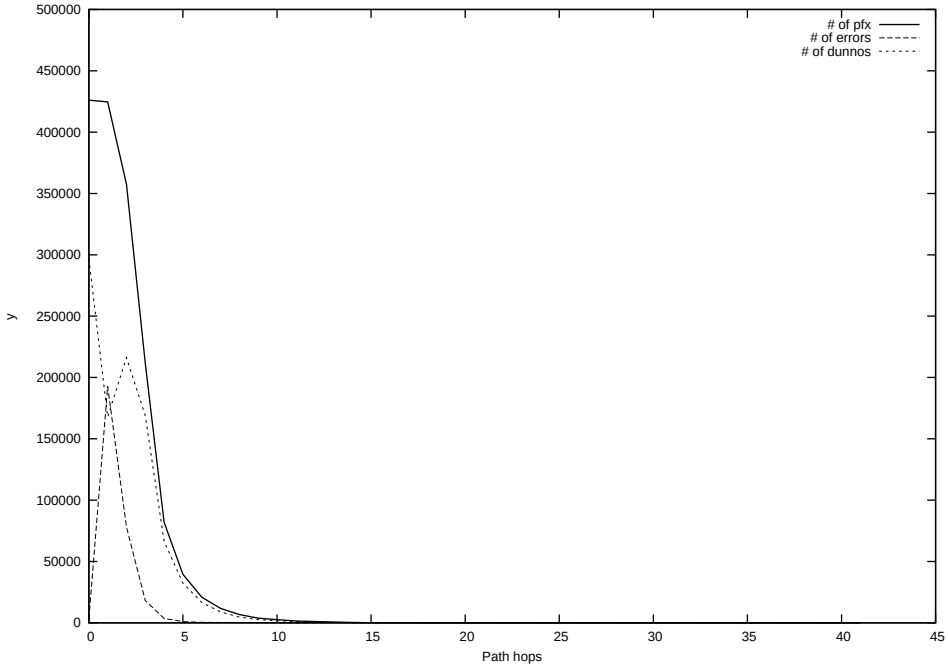
2012-10-31



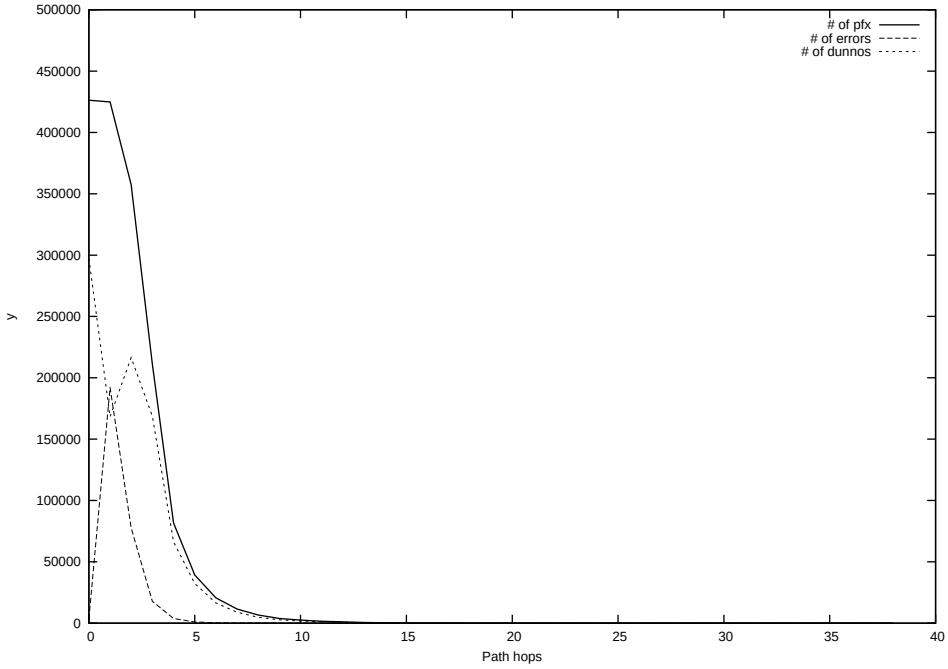
2012-11-01



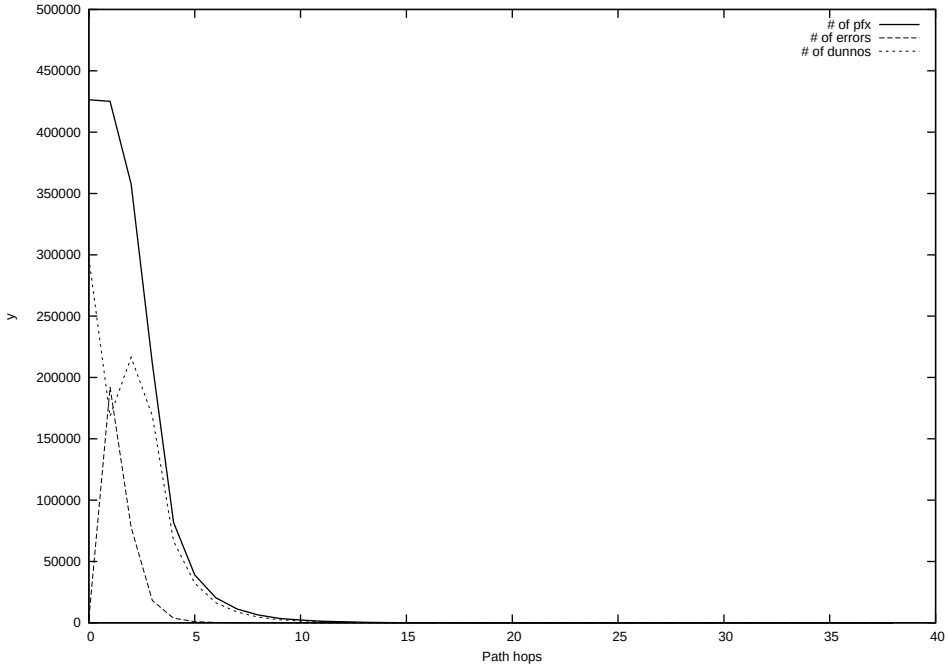
2012-11-02



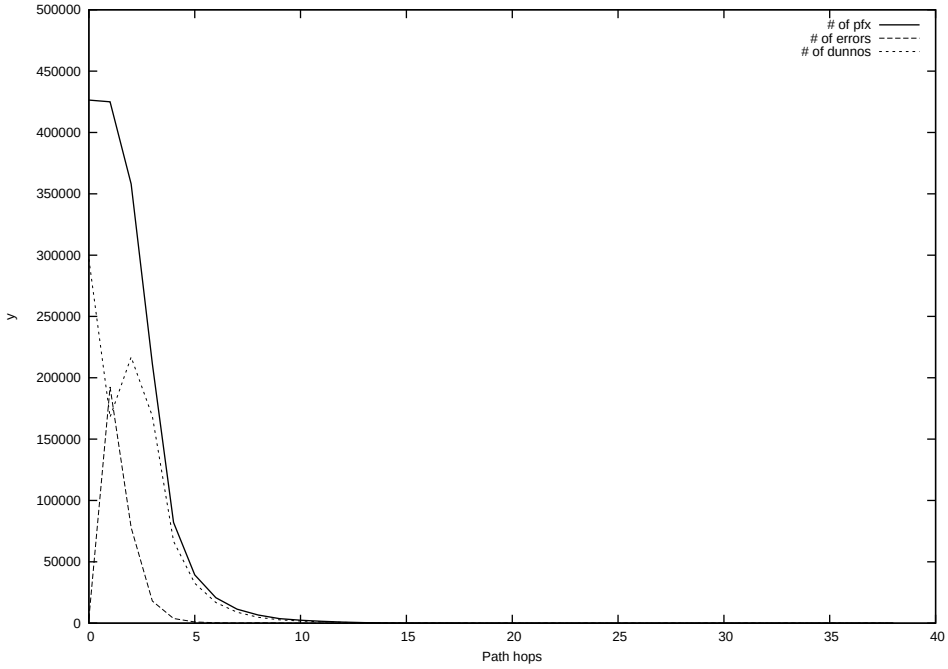
2012-11-03



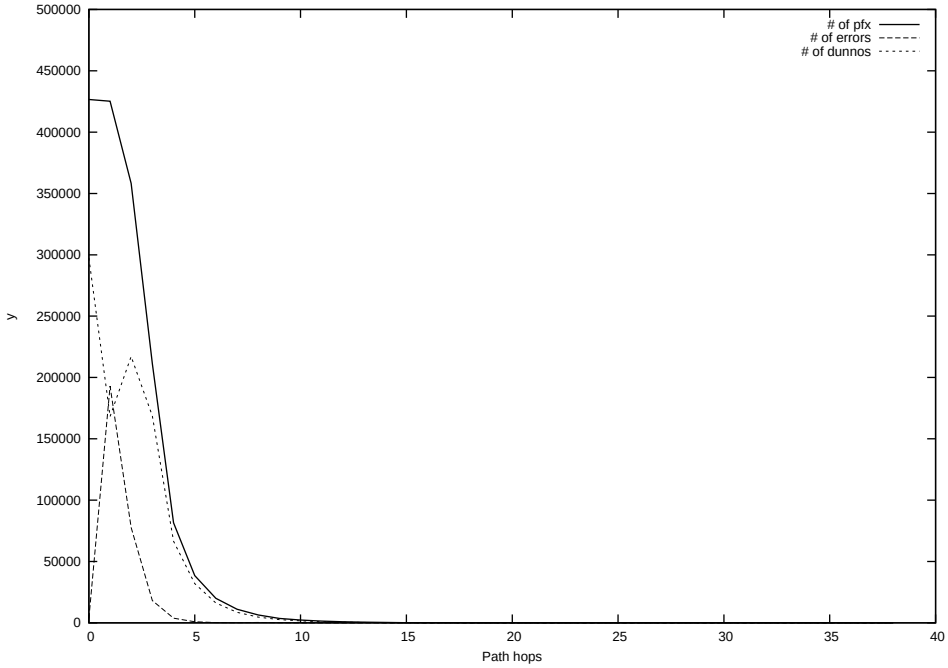
2012-11-04



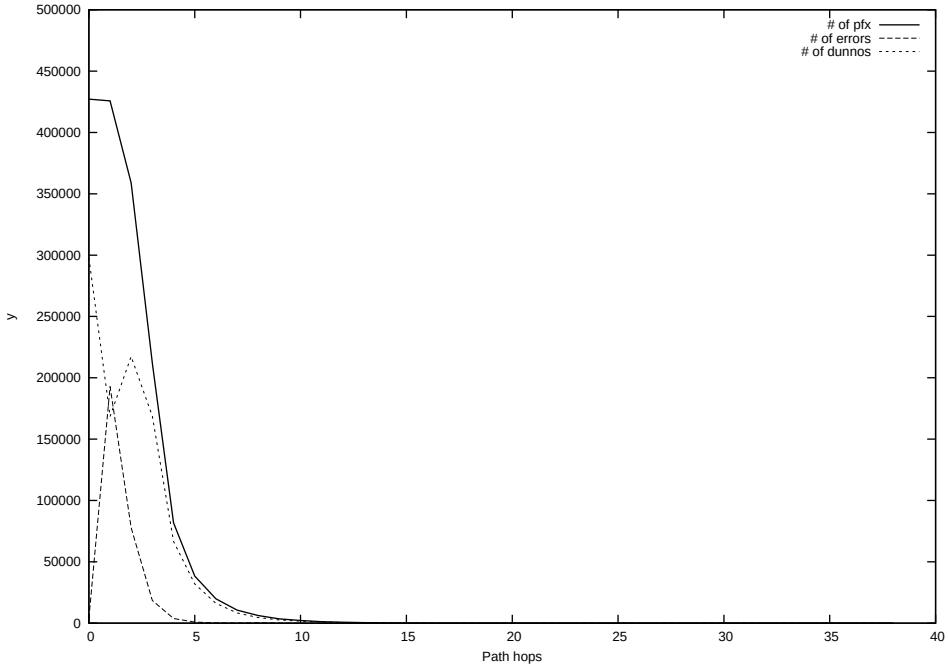
2012-11-05



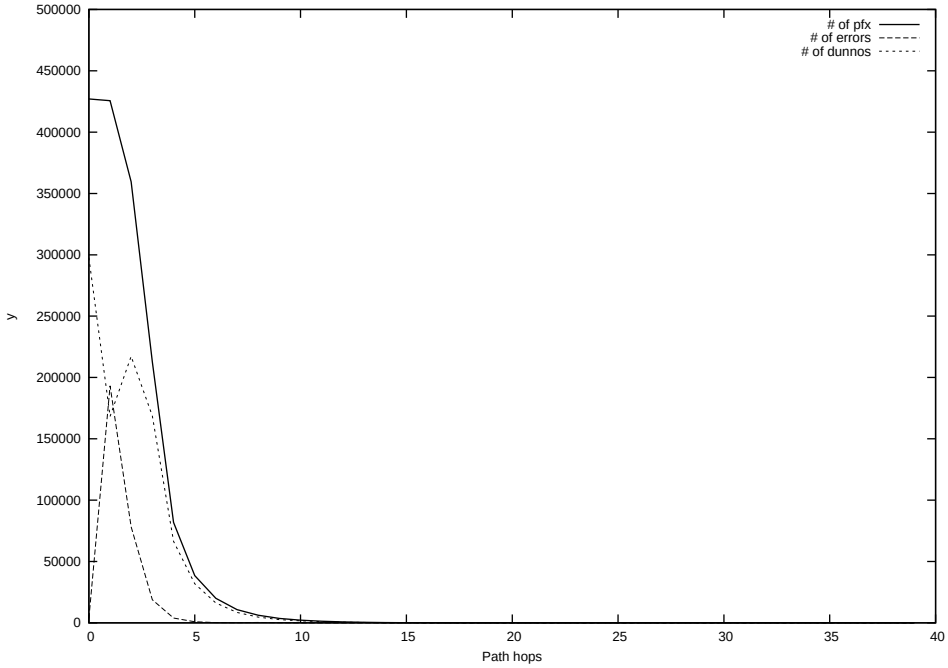
2012-11-06



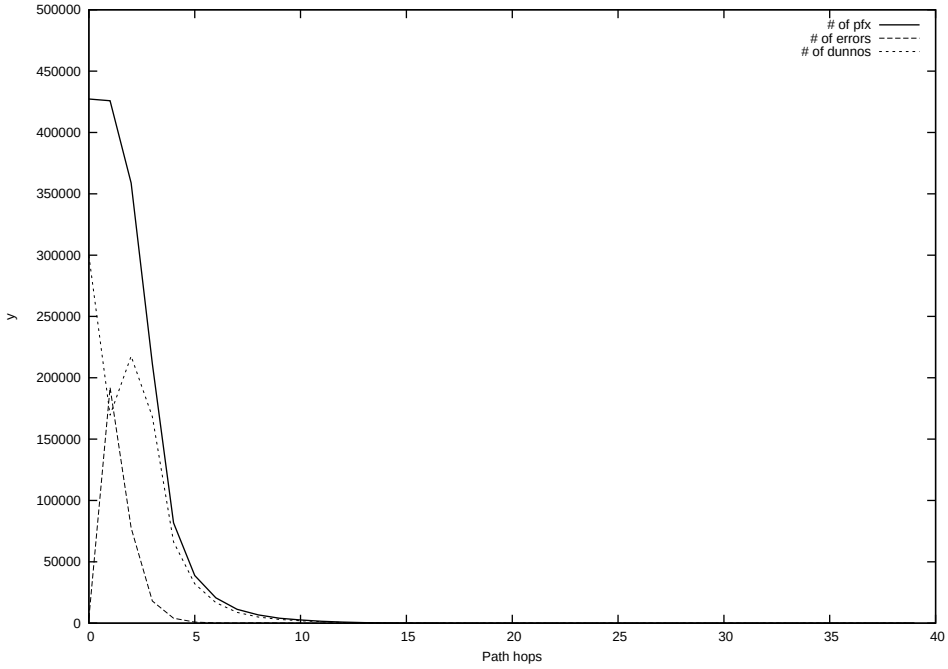
2012-11-07



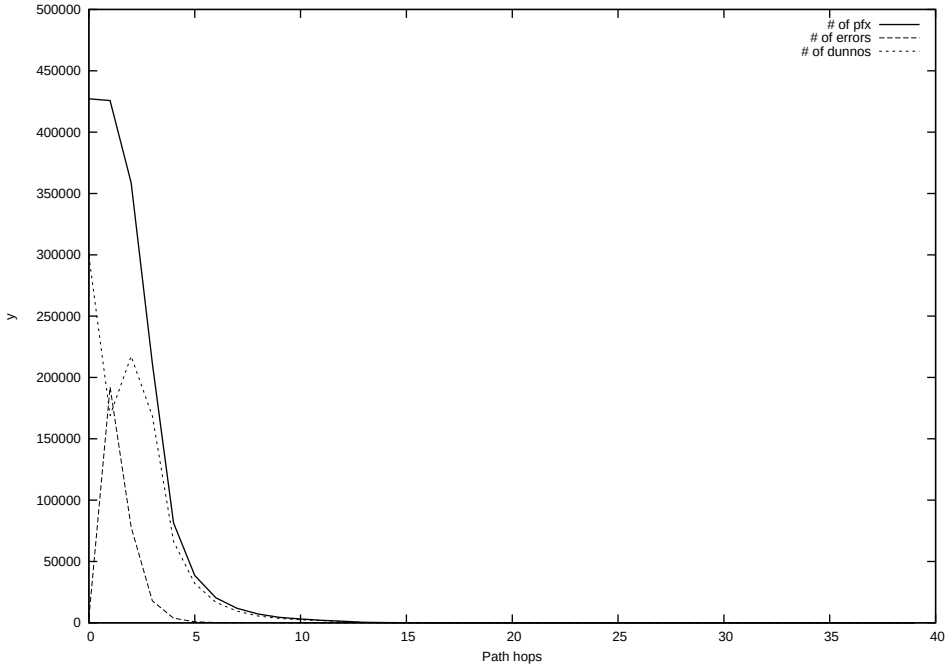
2012-11-08



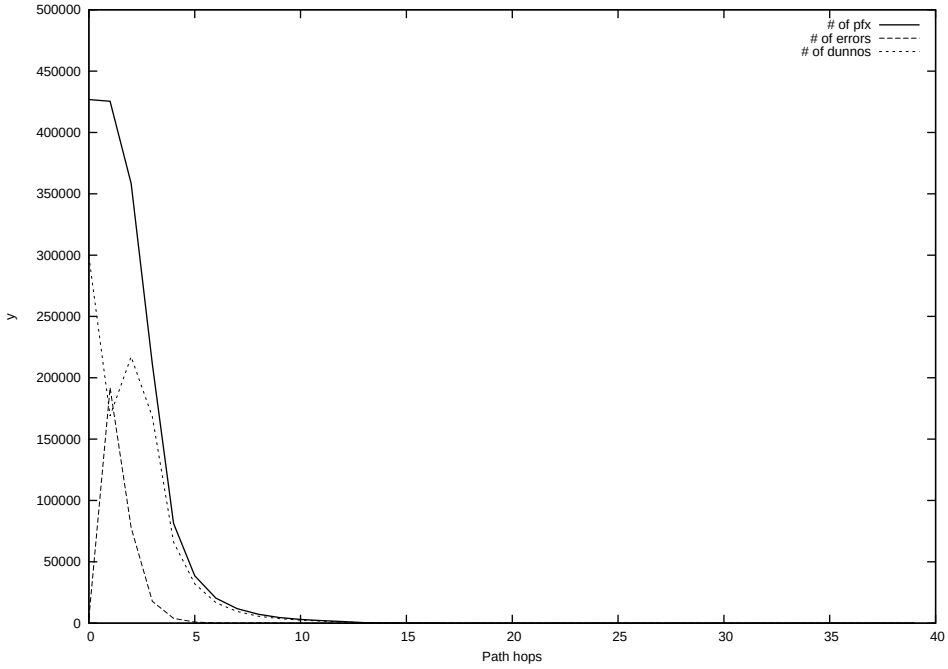
2012-11-09



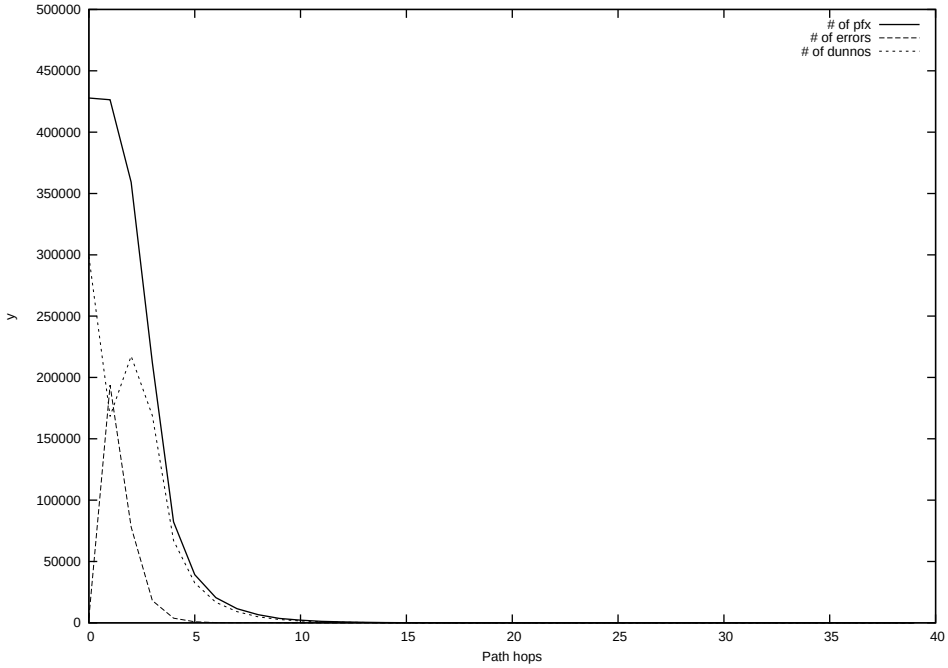
2012-11-10



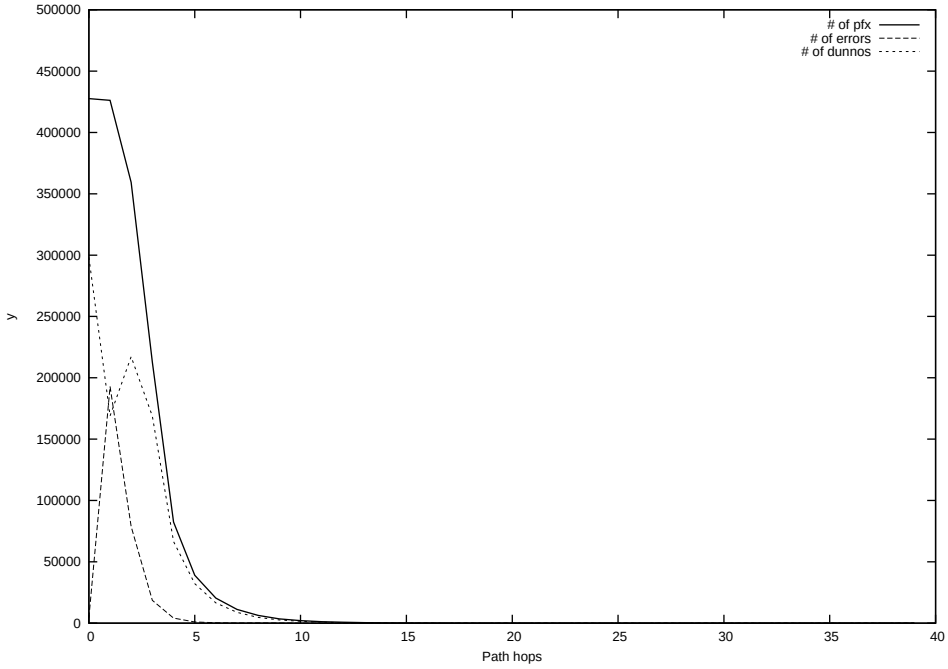
2012-11-11



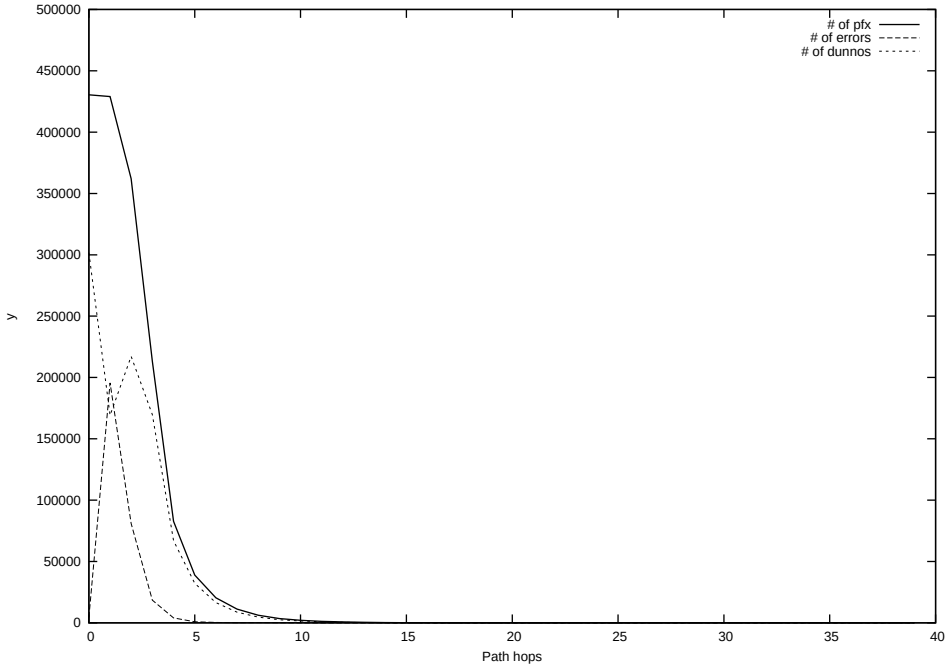
2012-11-12



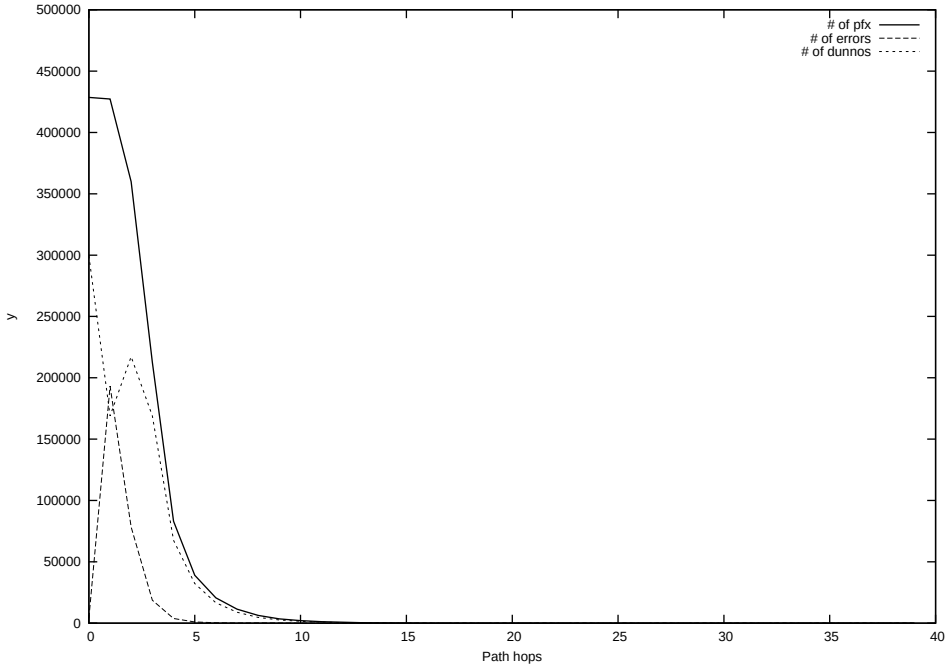
2012-11-13



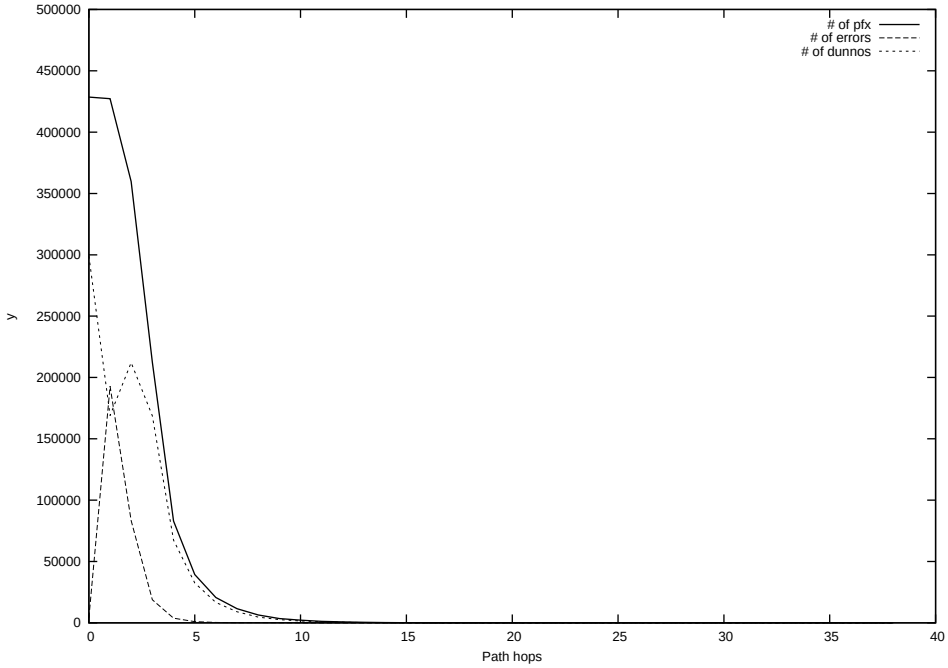
2012-11-14



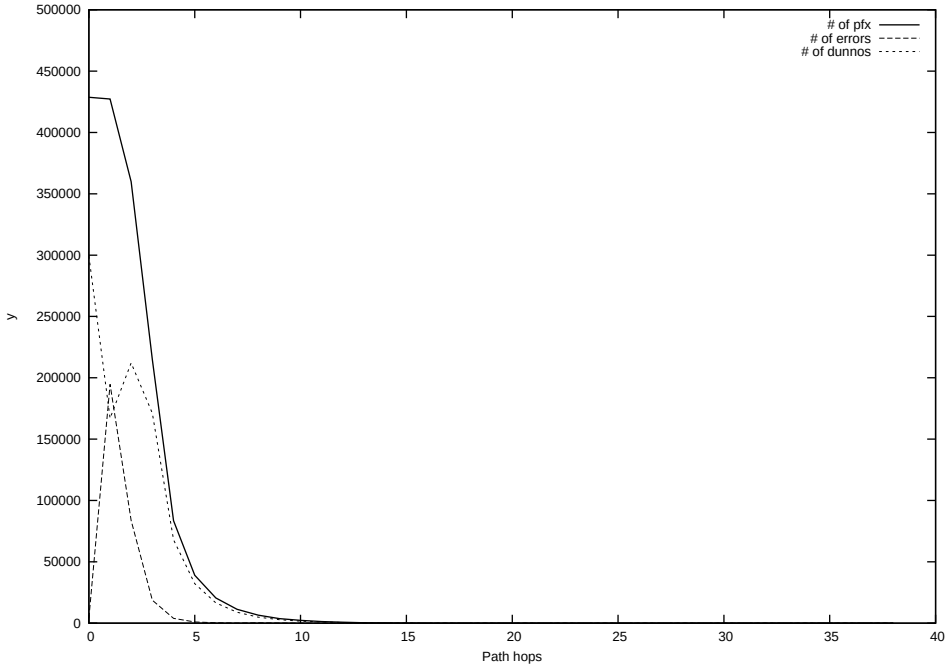
2012-11-15



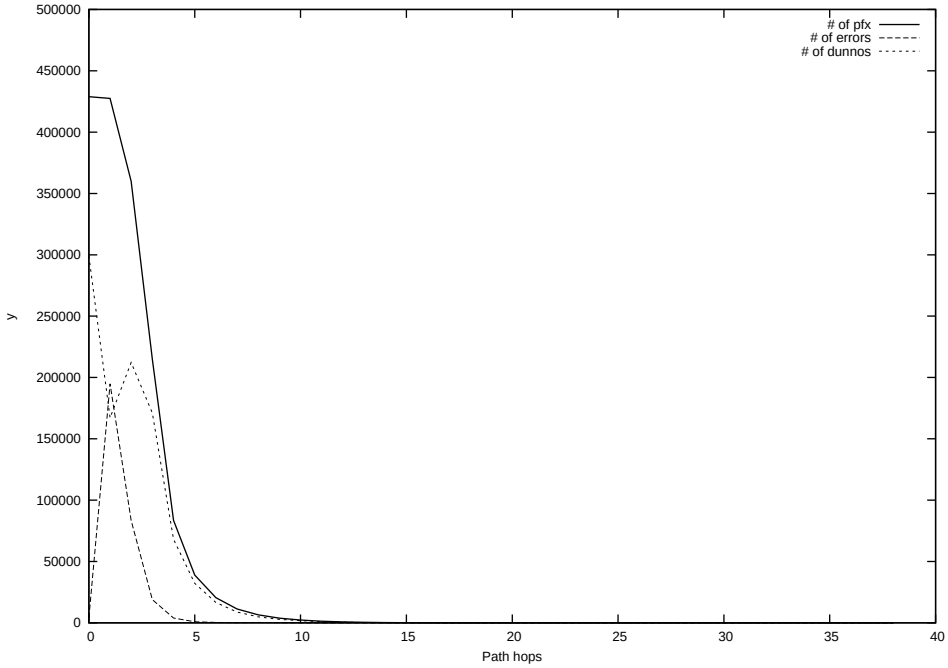
2012-11-16



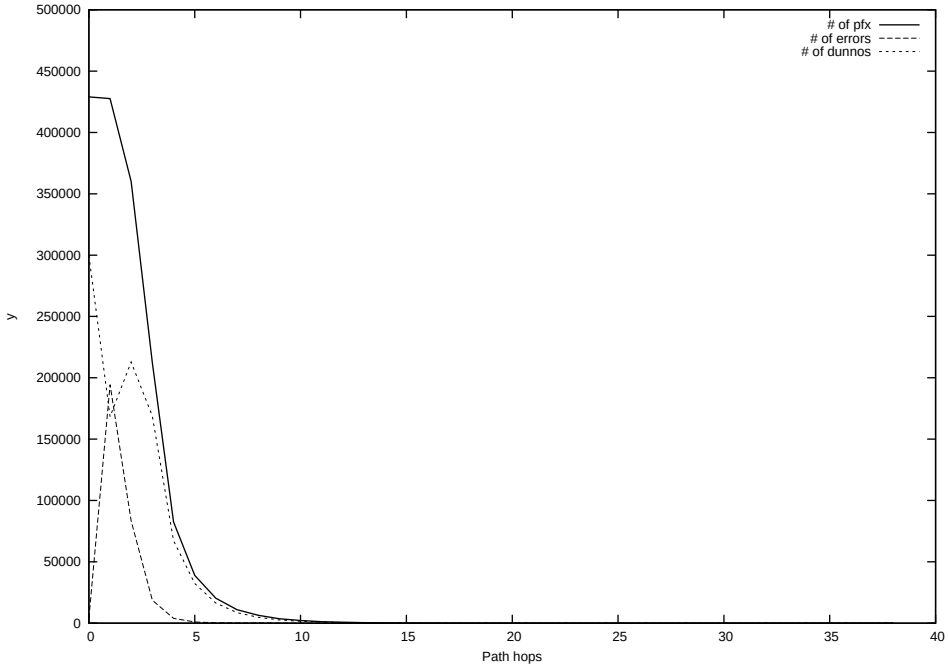
2012-11-17



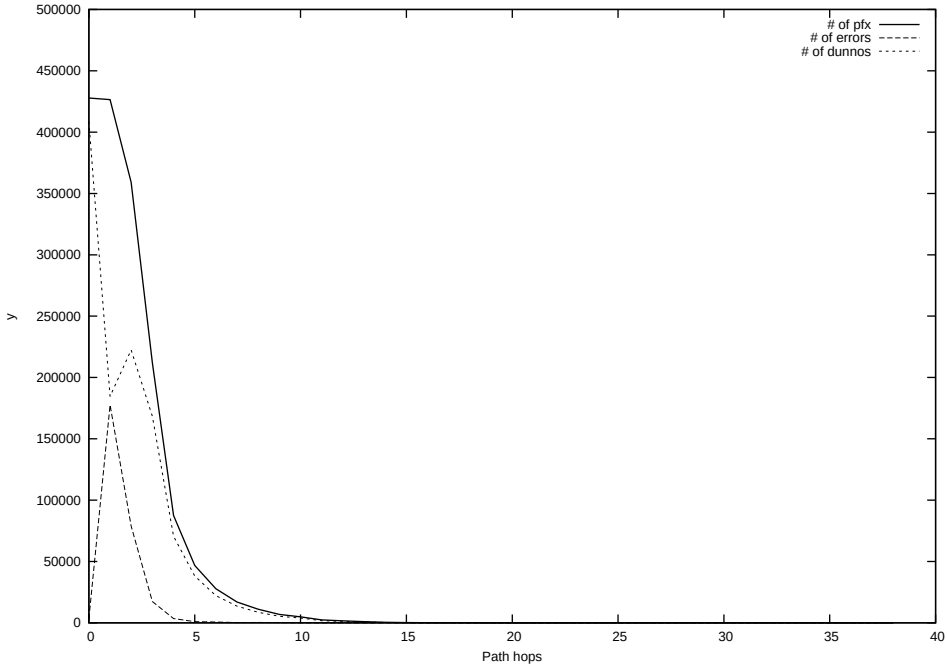
2012-11-18



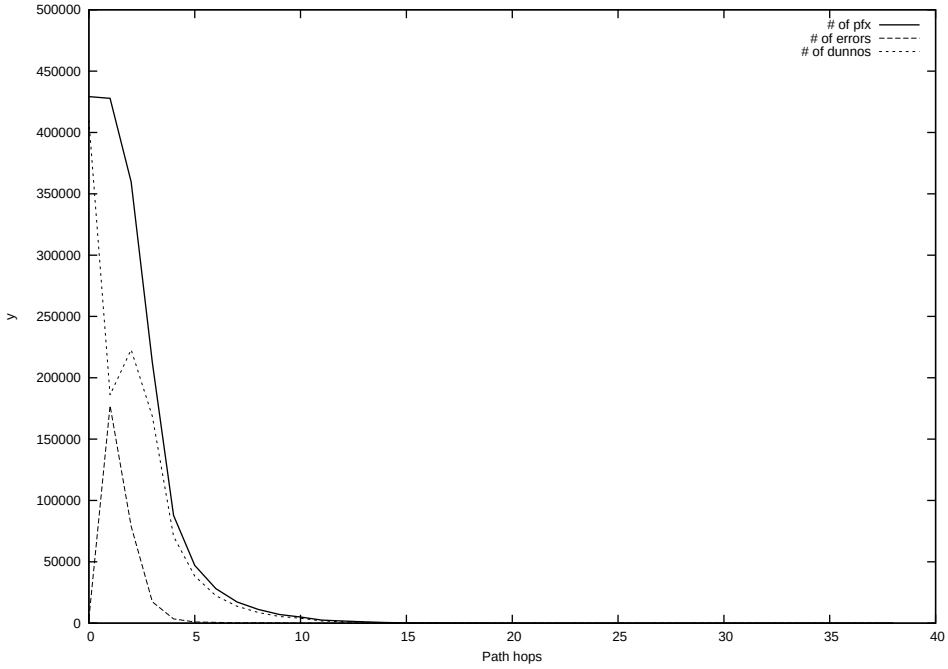
2012-11-19



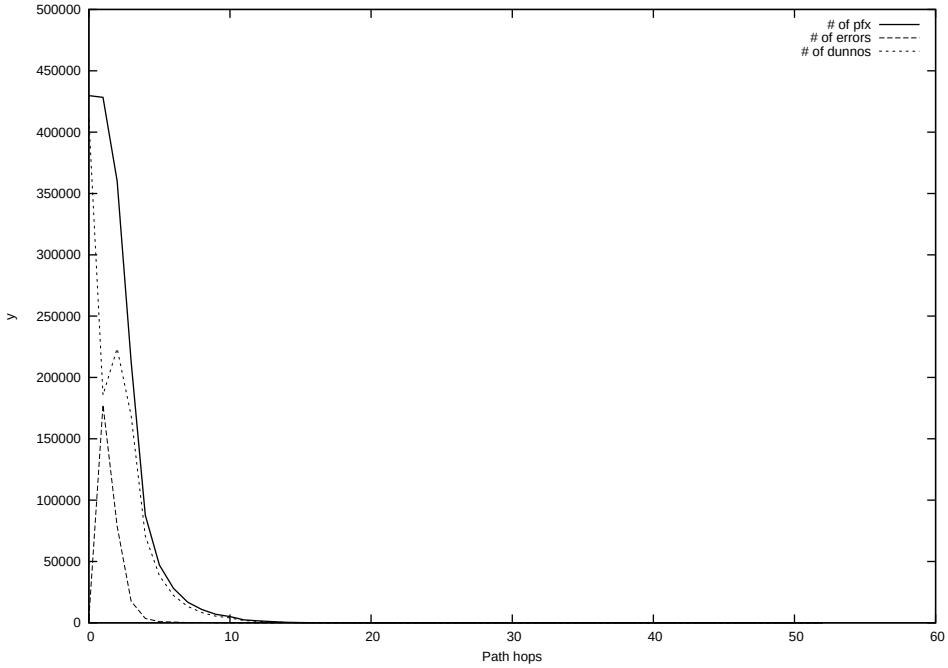
2012-11-20



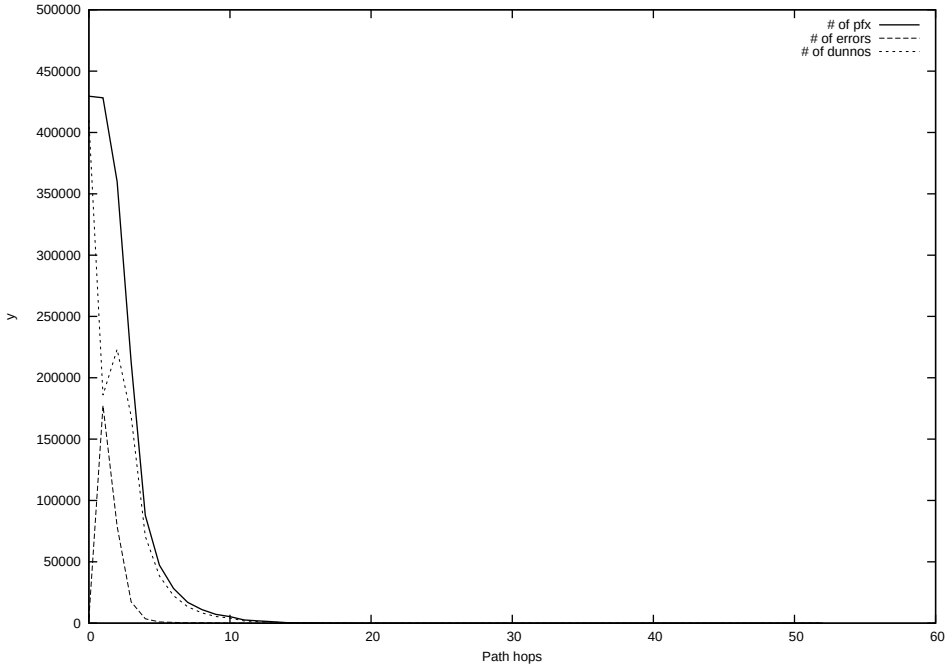
2012-11-21



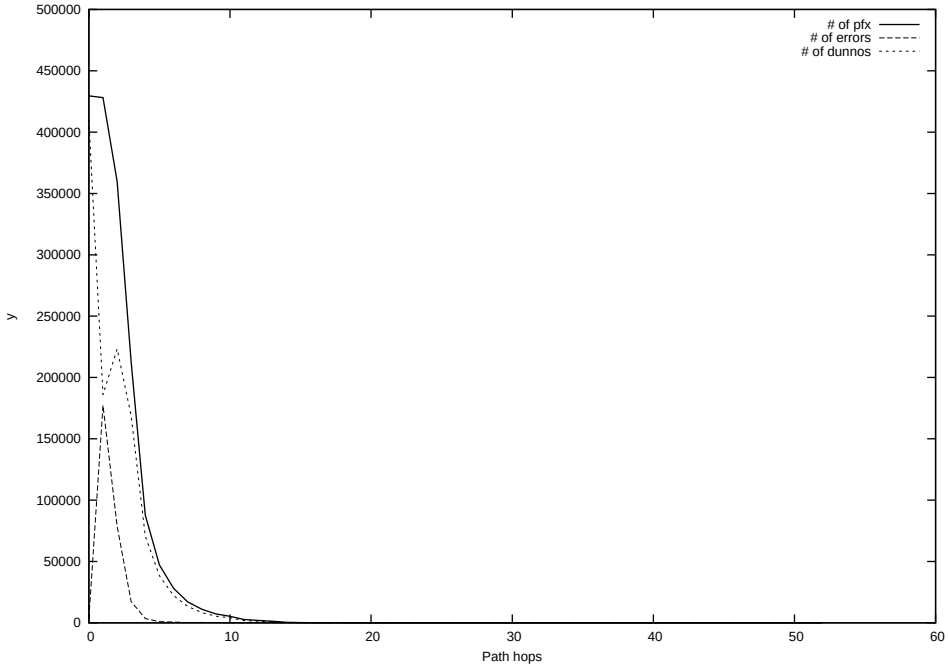
2012-11-22



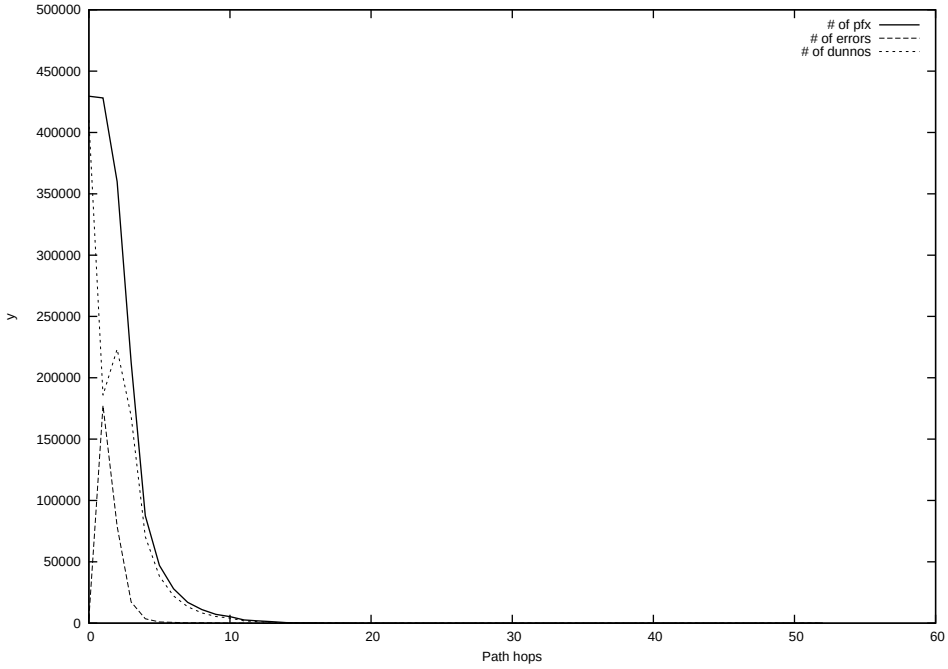
2012-11-23



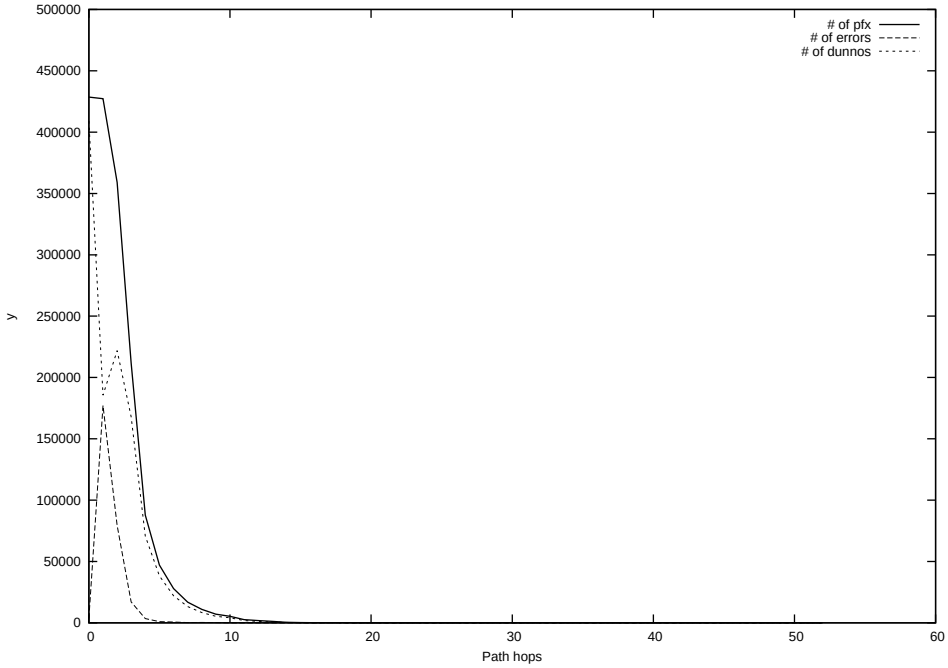
2012-11-24



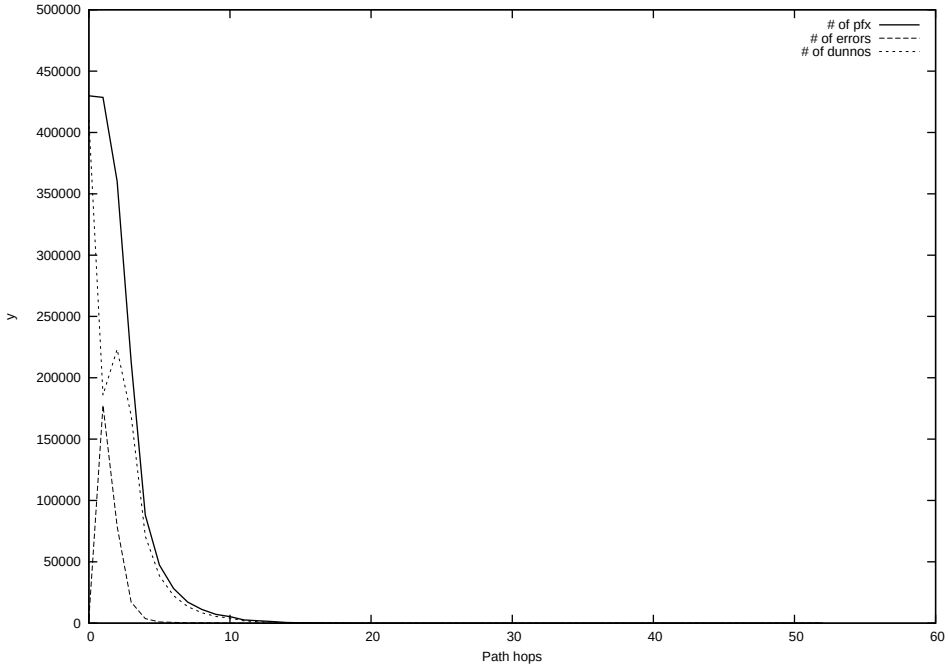
2012-11-25



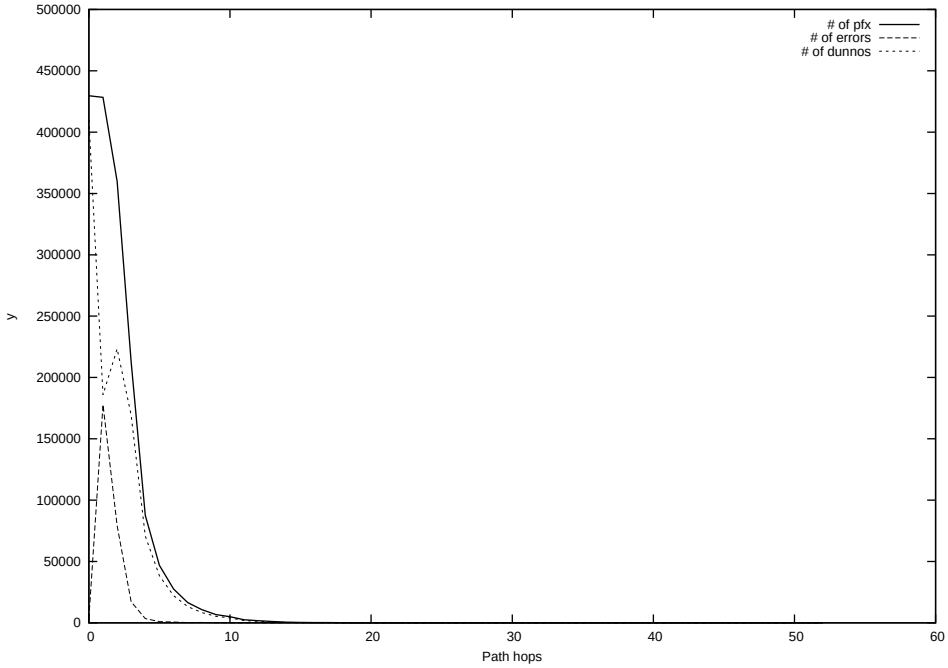
2012-11-26



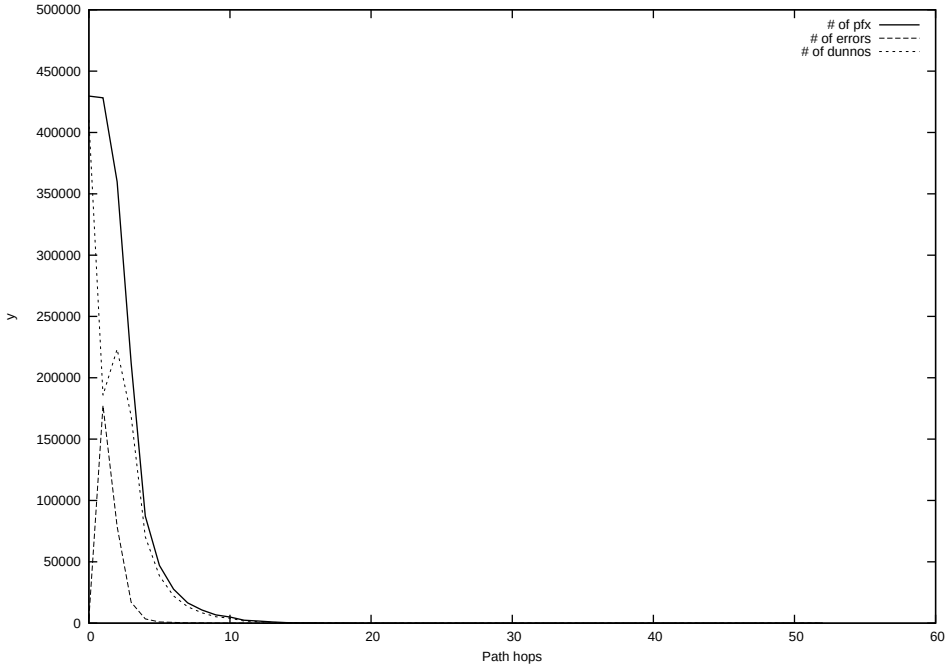
2012-11-27



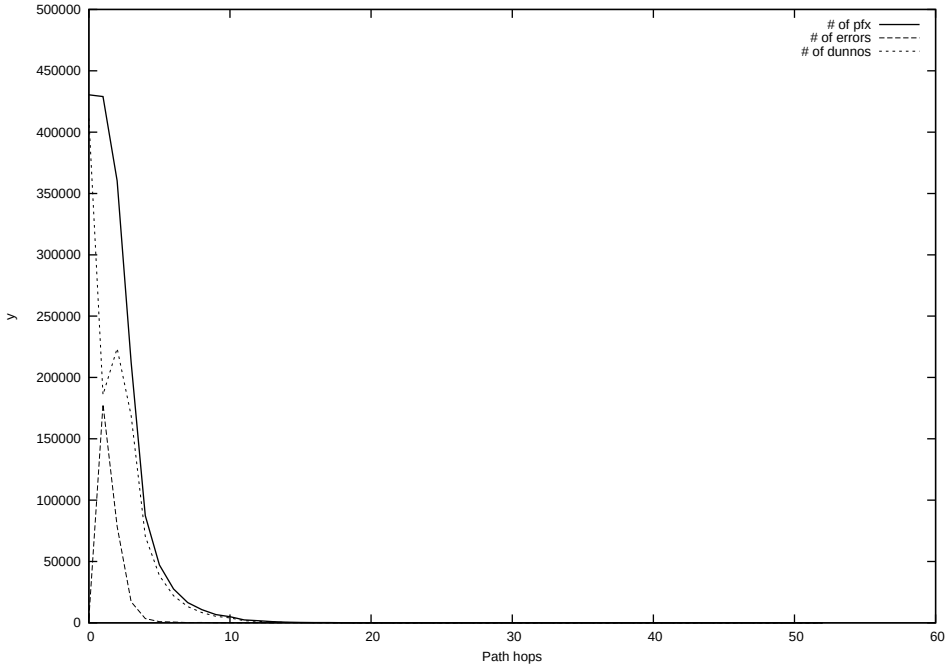
2012-11-28



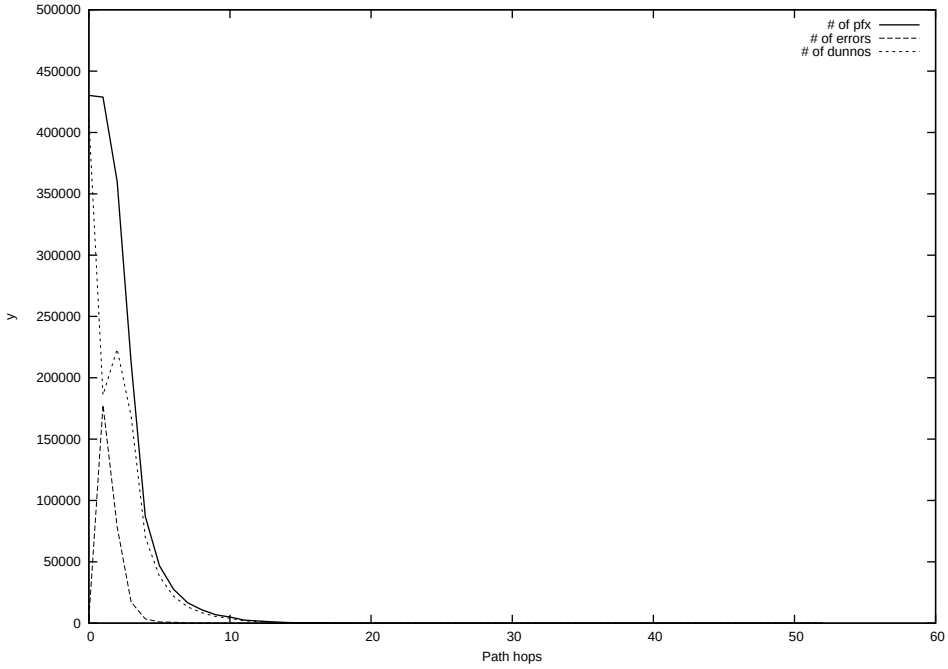
2012-11-29



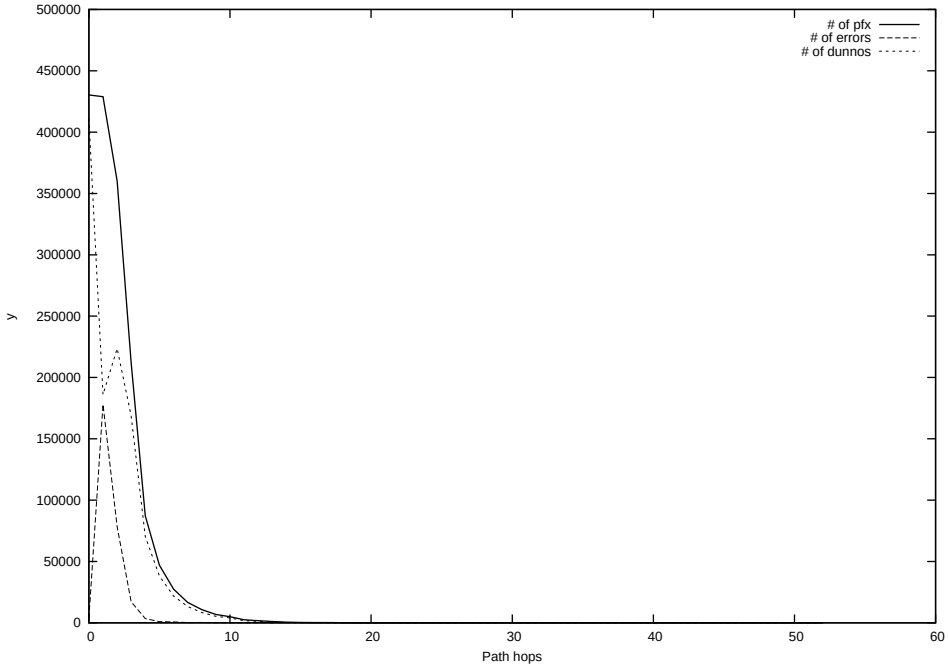
2012-11-30



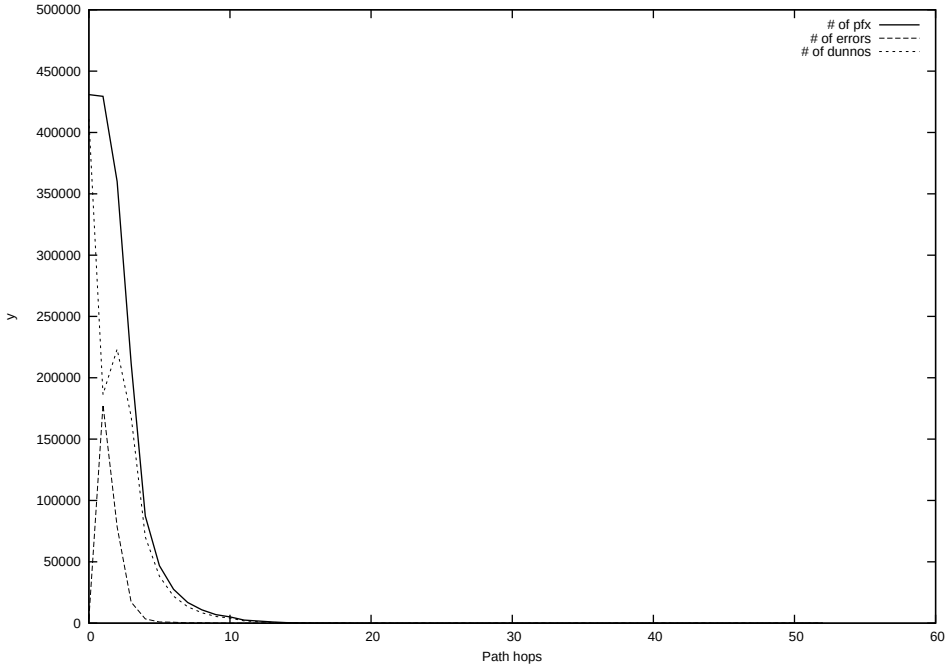
2012-12-01



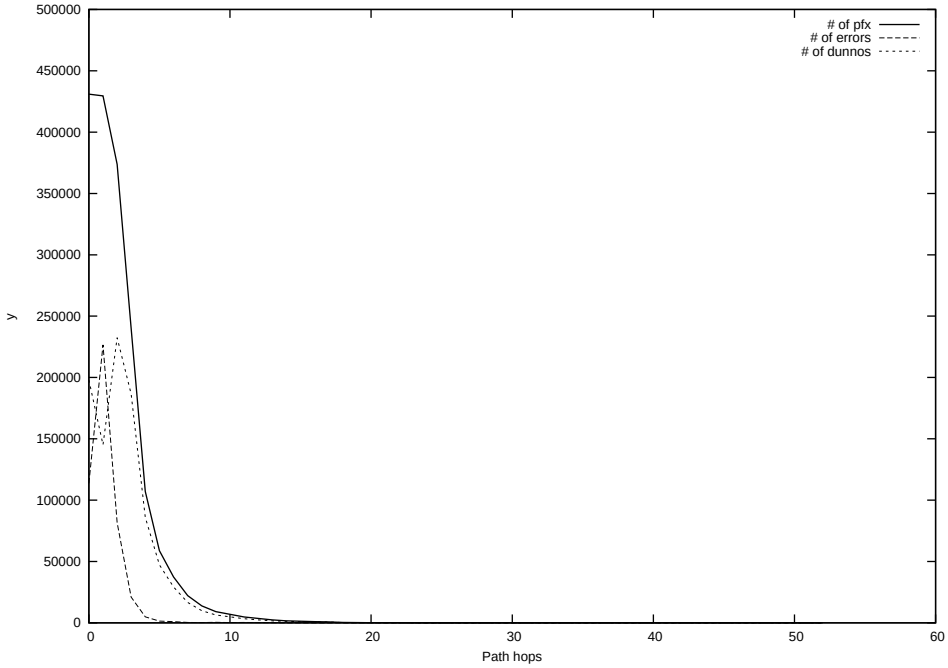
2012-12-02



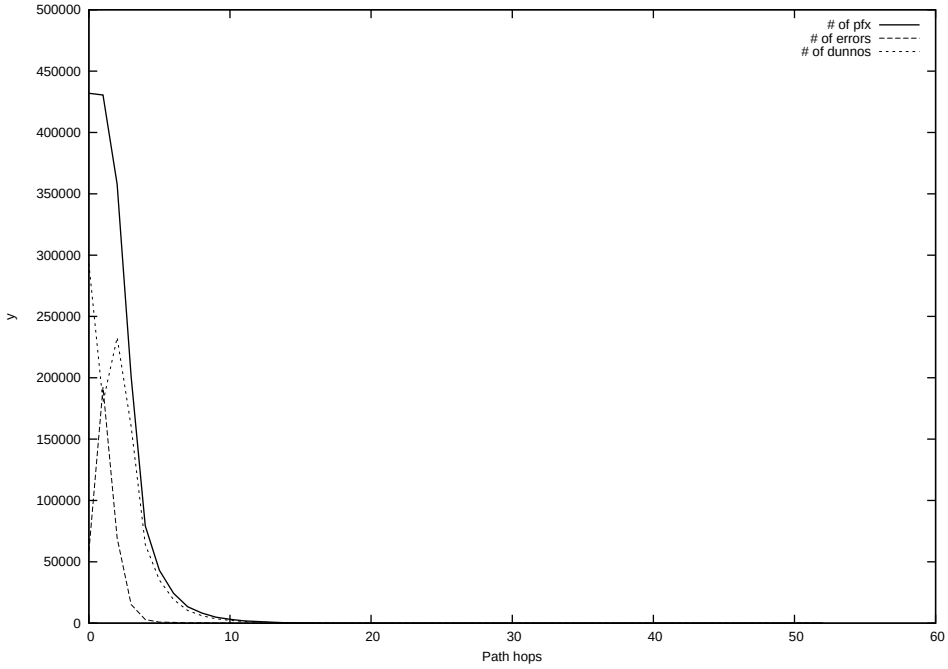
2012-12-03



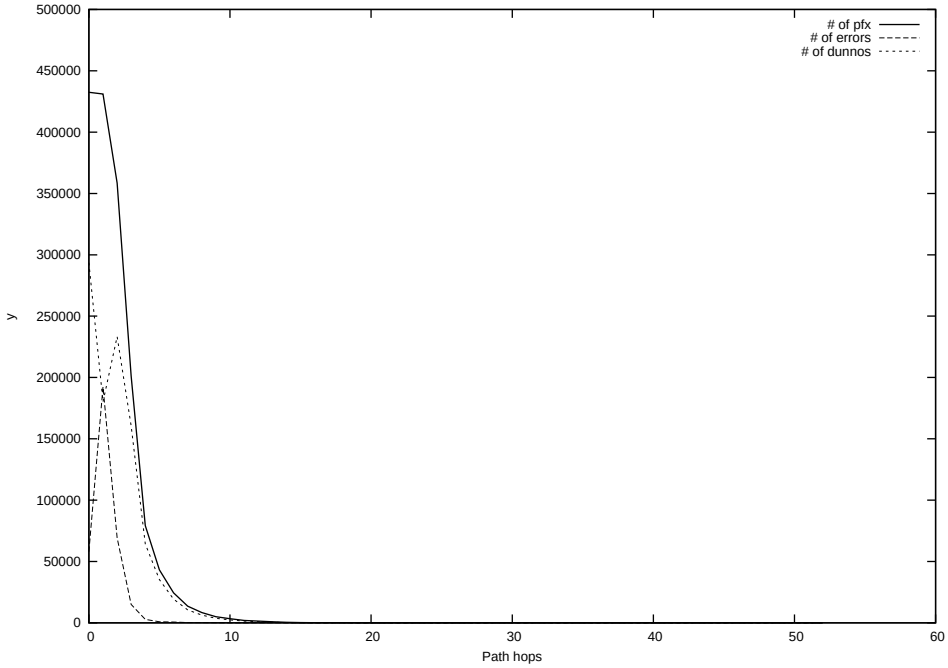
2012-12-04



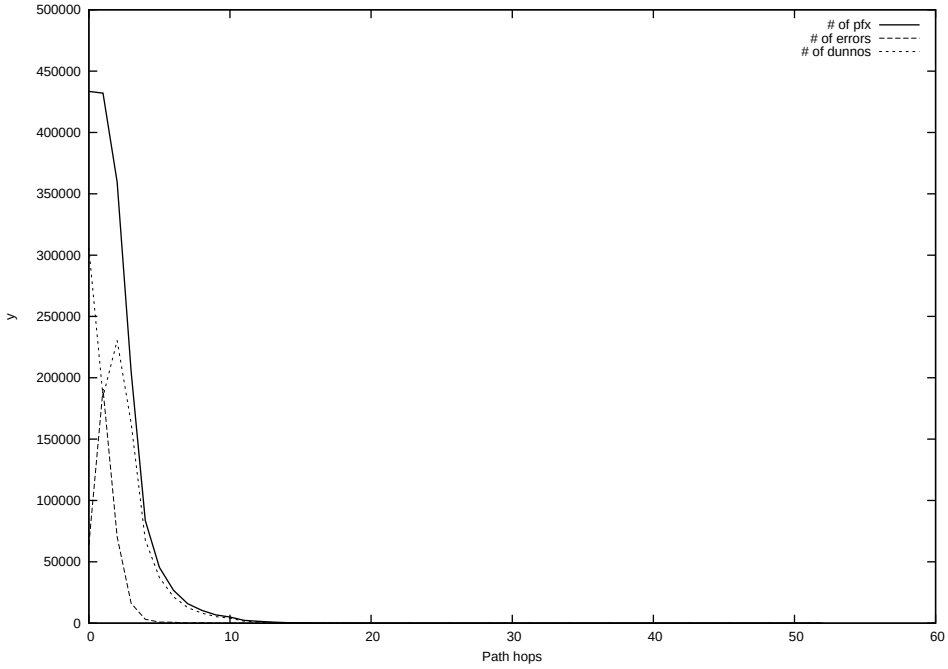
2012-12-05



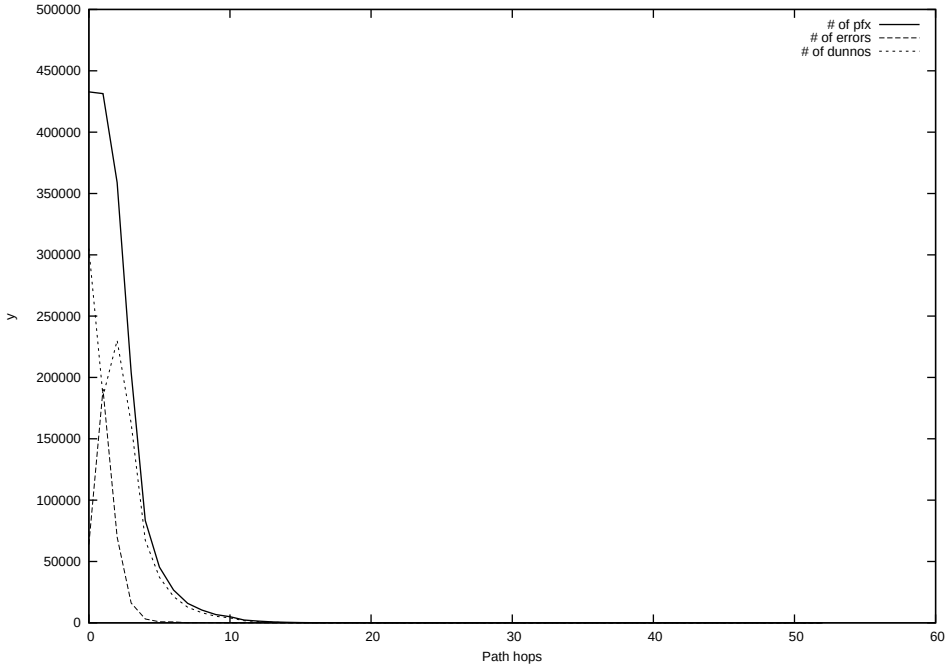
2012-12-06



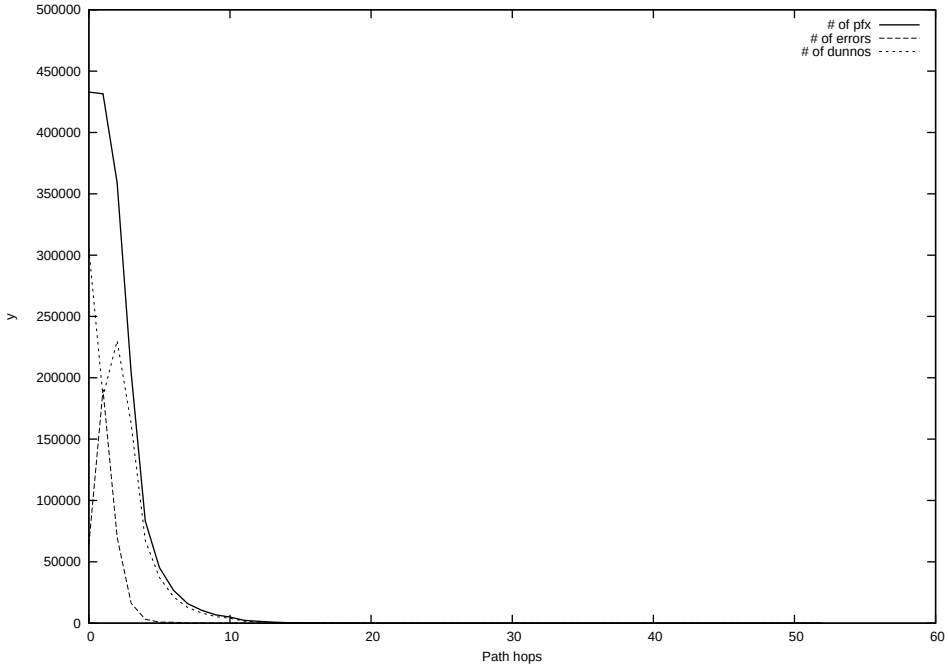
2012-12-07



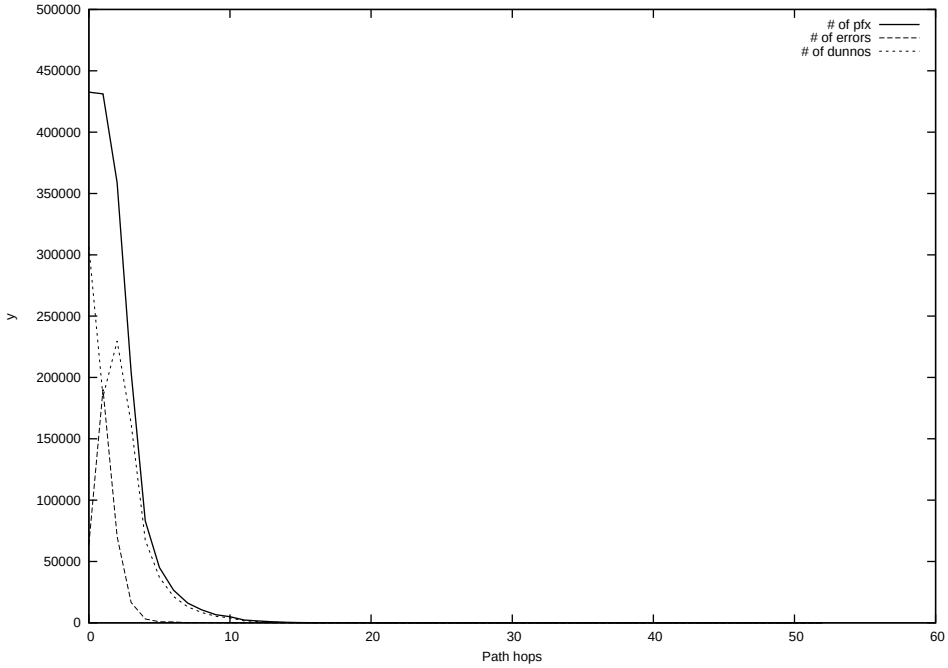
2012-12-08



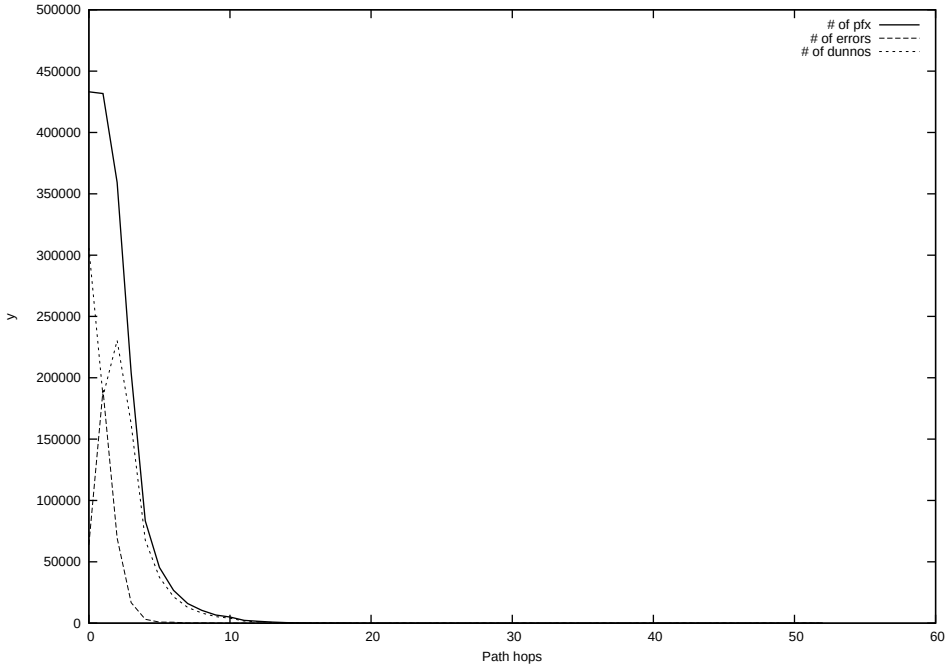
2012-12-09



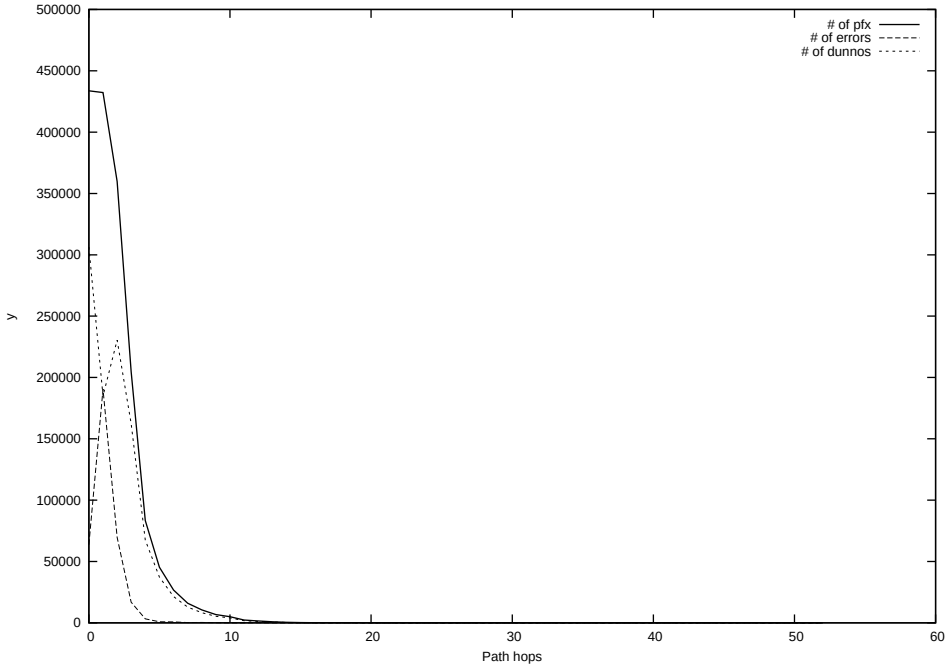
2012-12-10



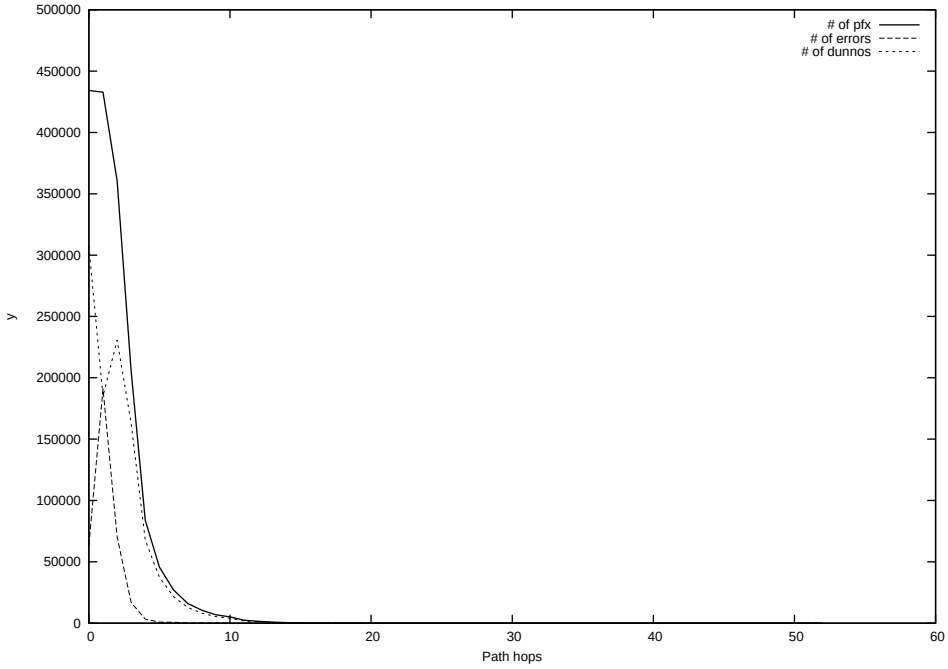
2012-12-11



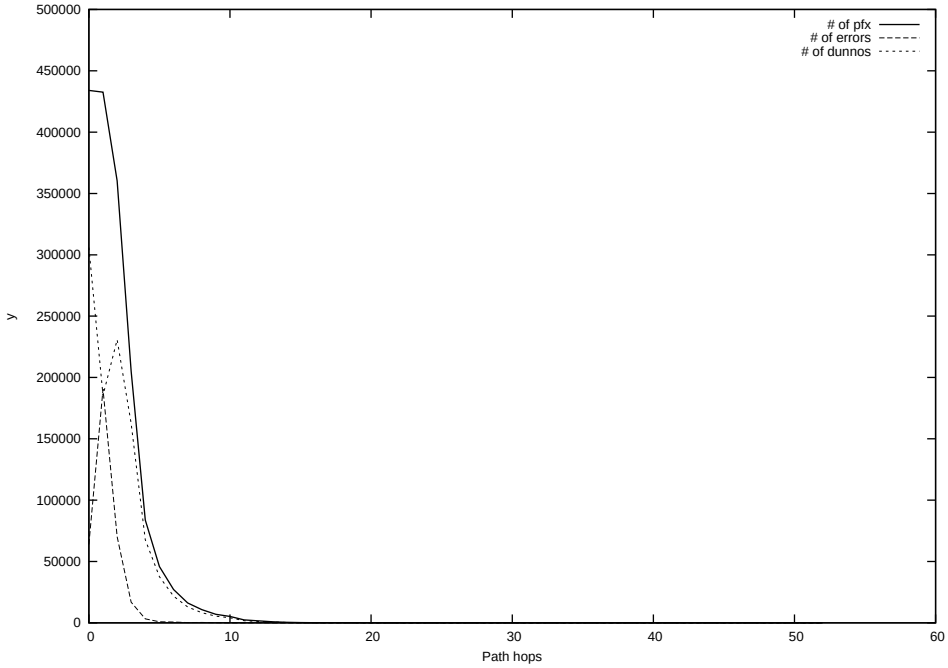
2012-12-12



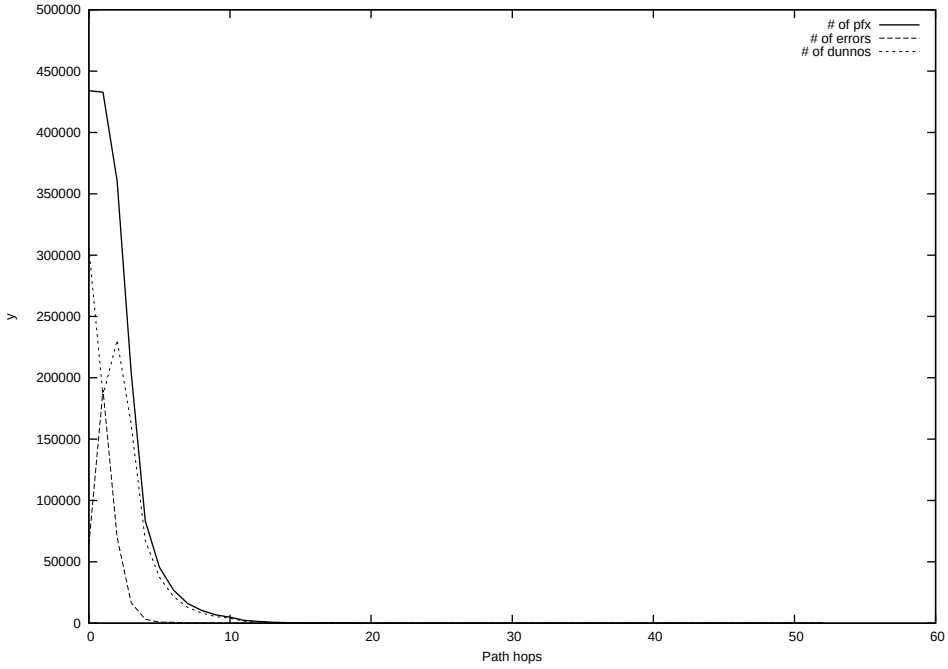
2012-12-13



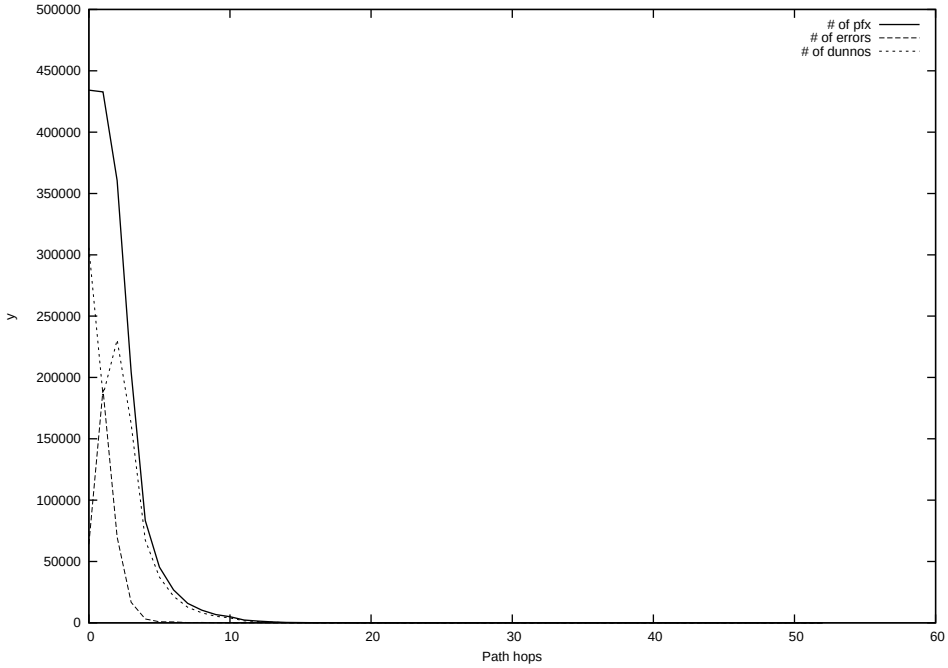
2012-12-14



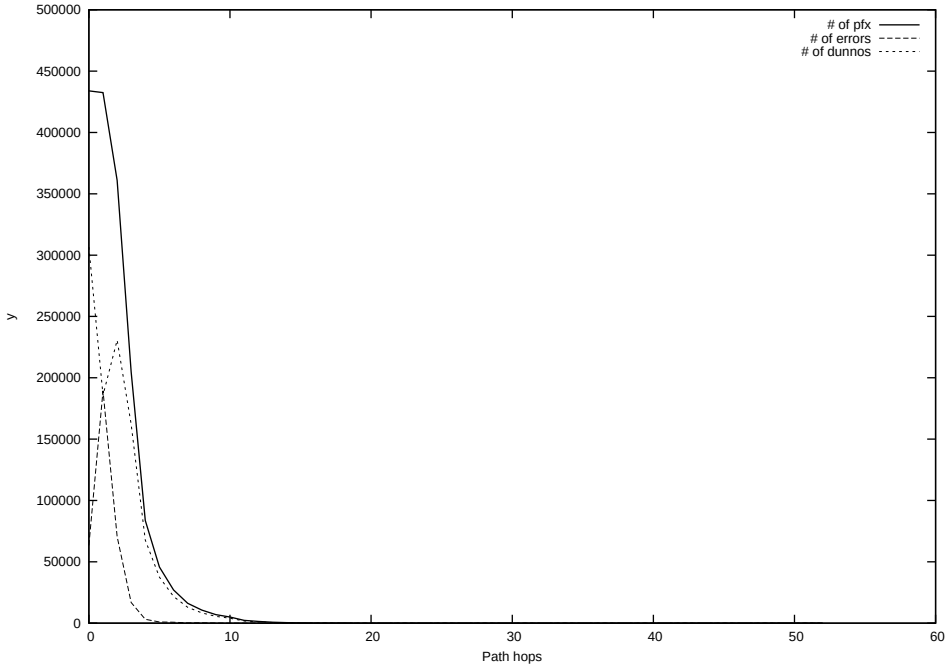
2012-12-15



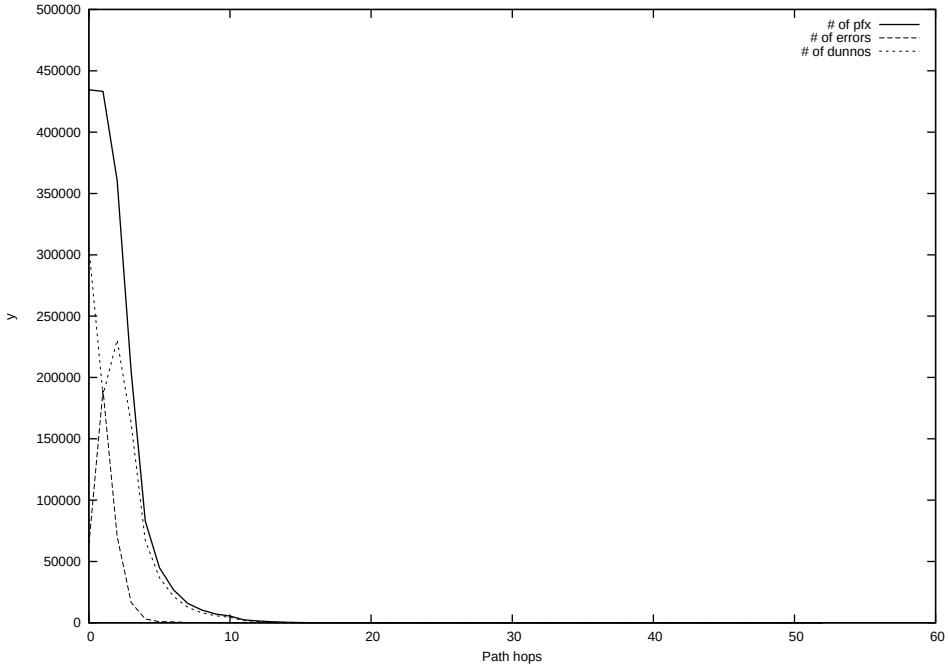
2012-12-16



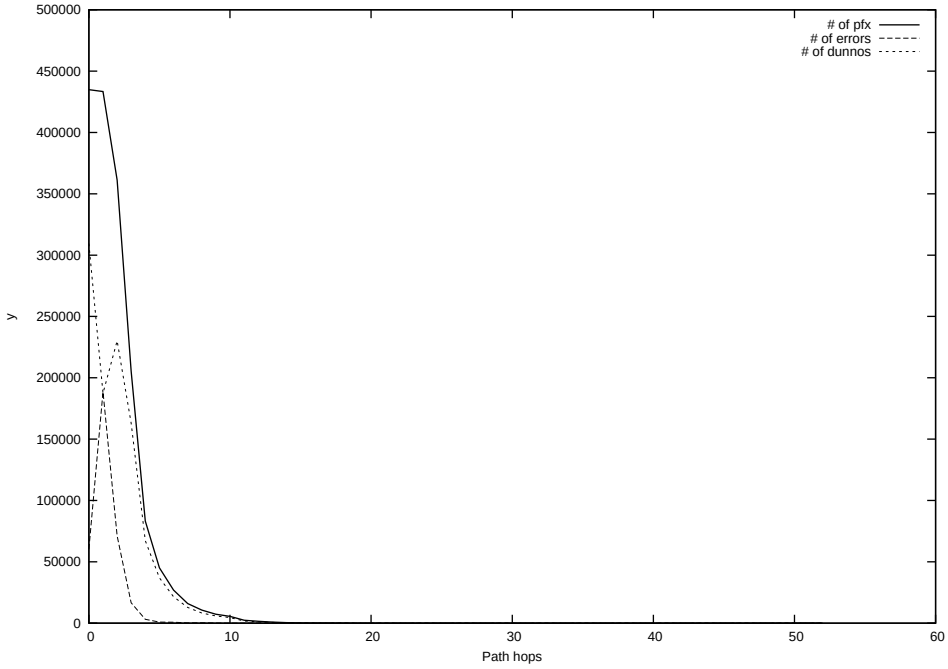
2012-12-17



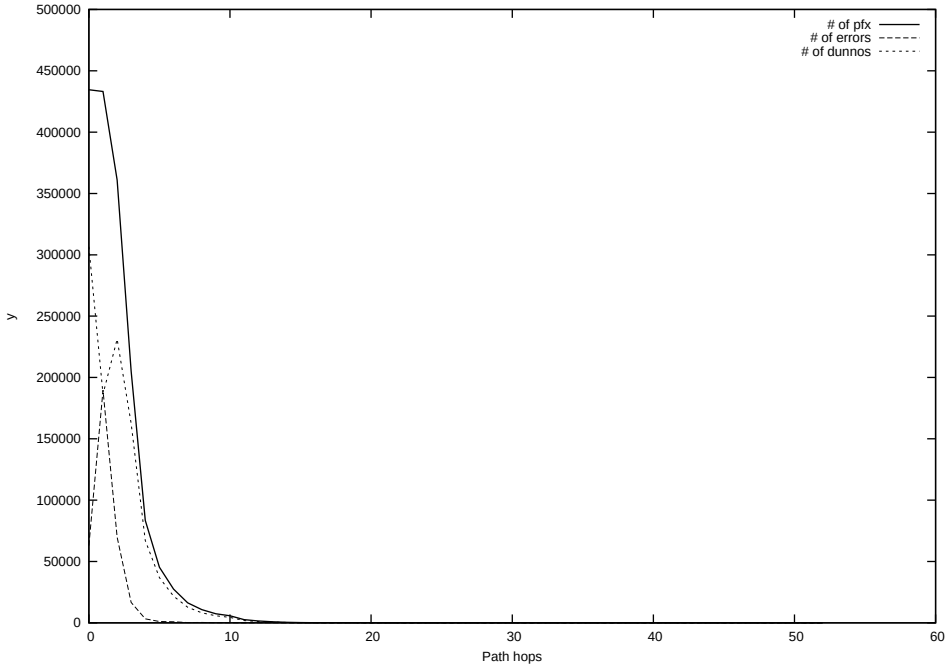
2012-12-18



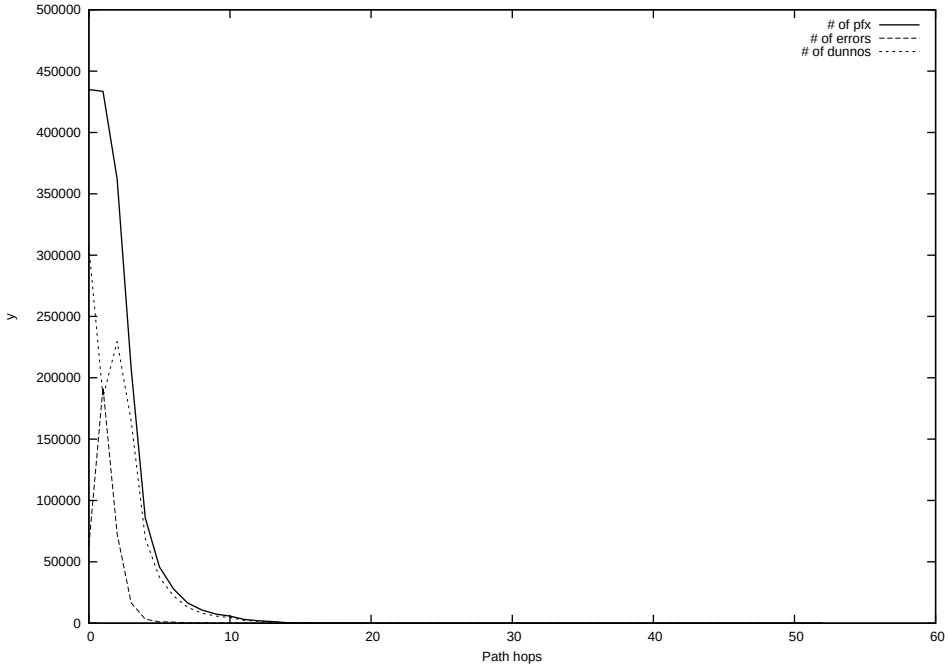
2012-12-19



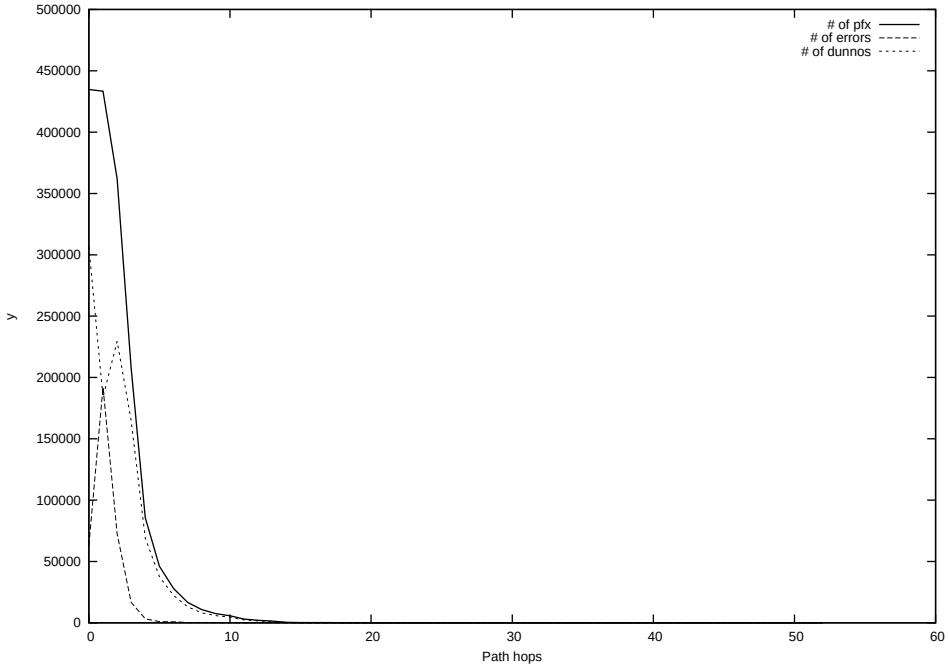
2012-12-20



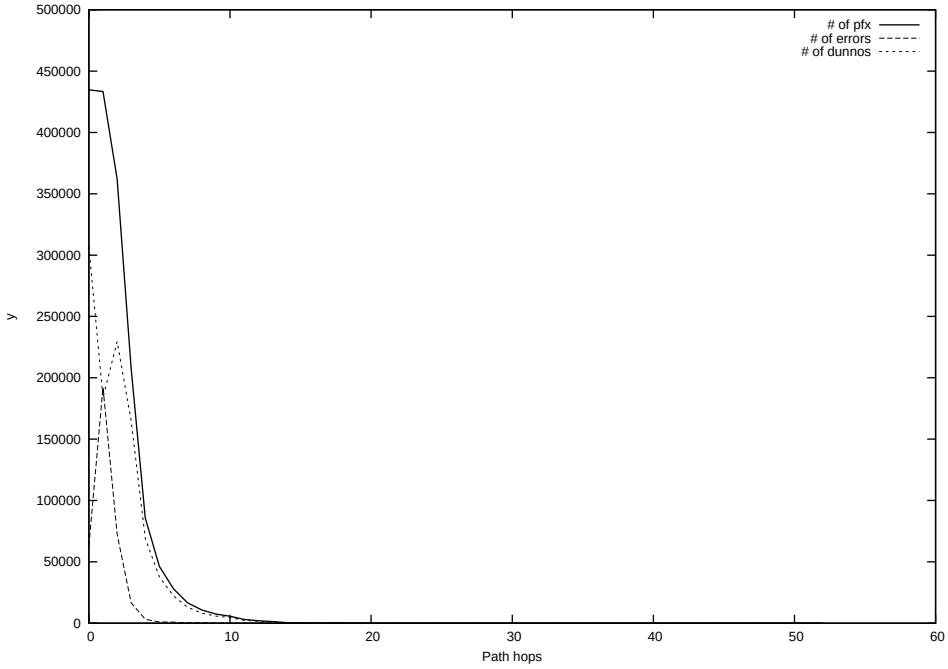
2012-12-21



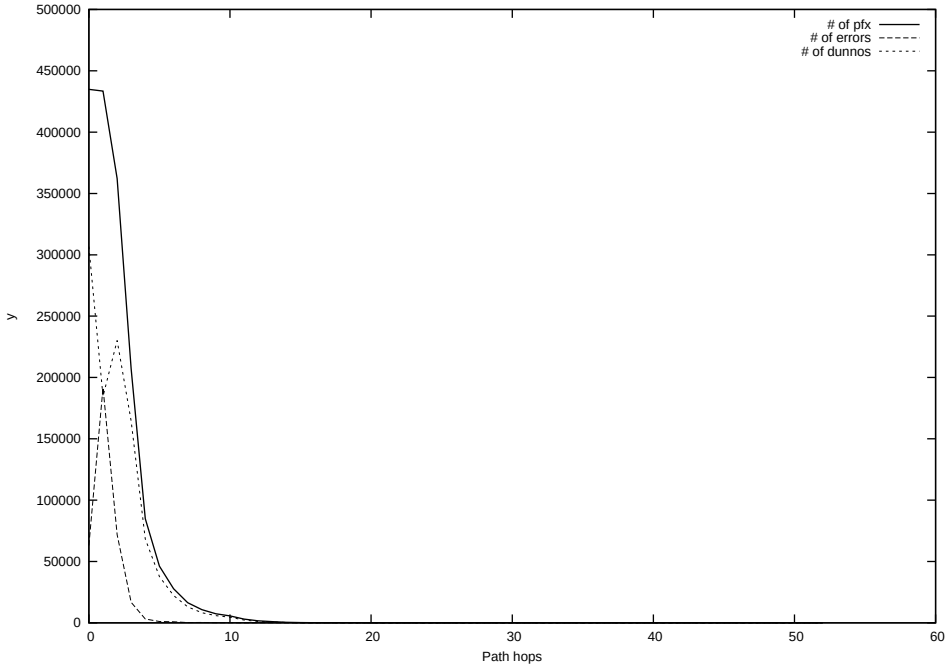
2012-12-22



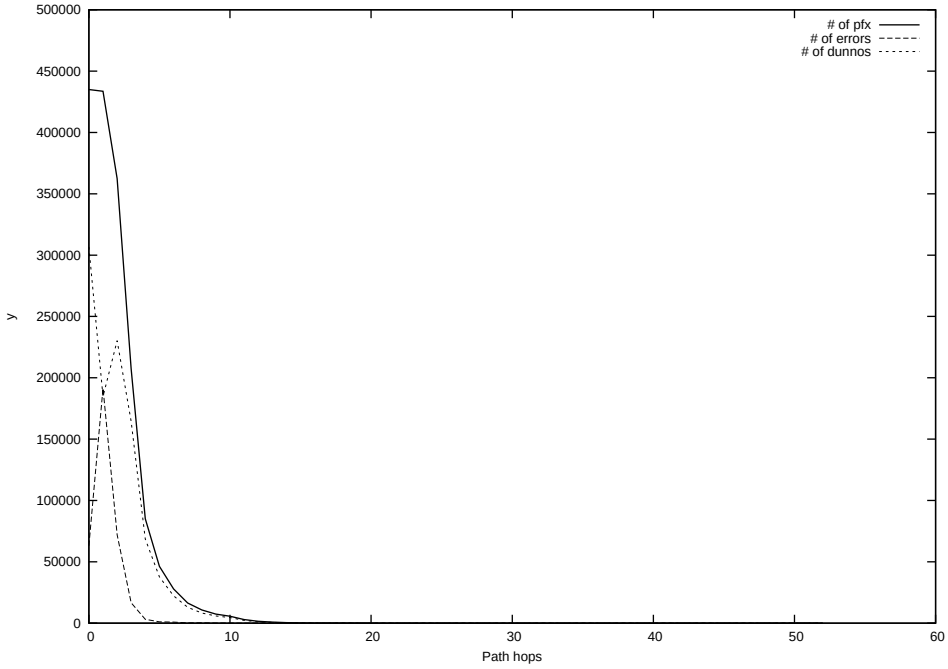
2012-12-23



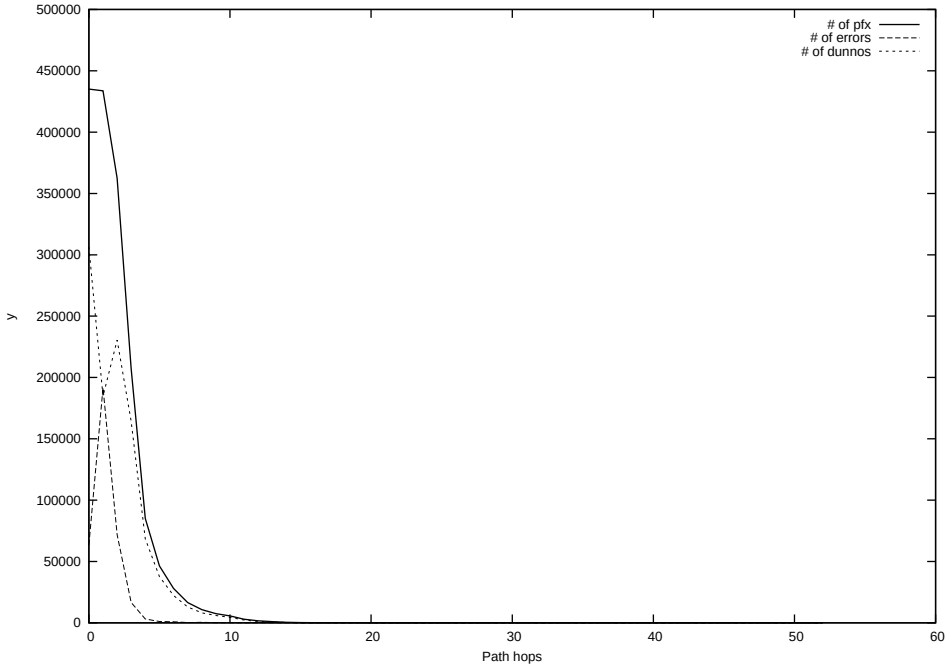
2012-12-24



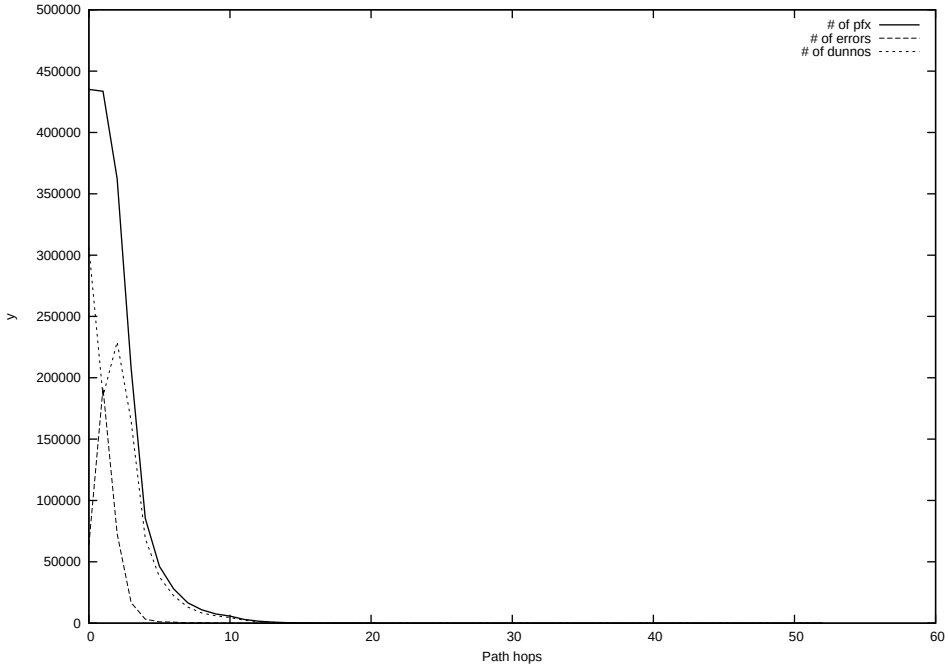
2012-12-25



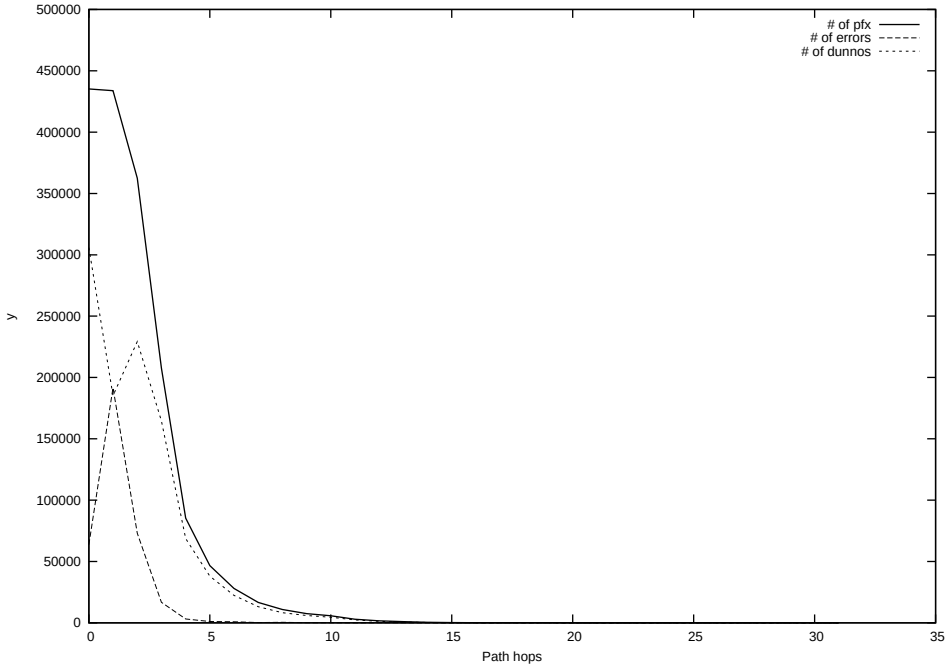
2012-12-26



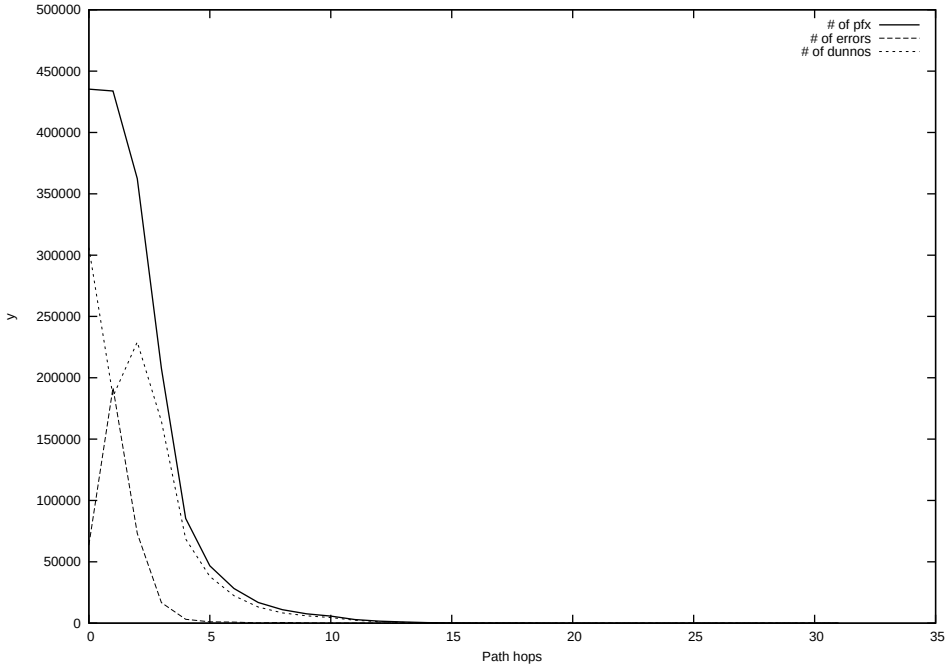
2012-12-27



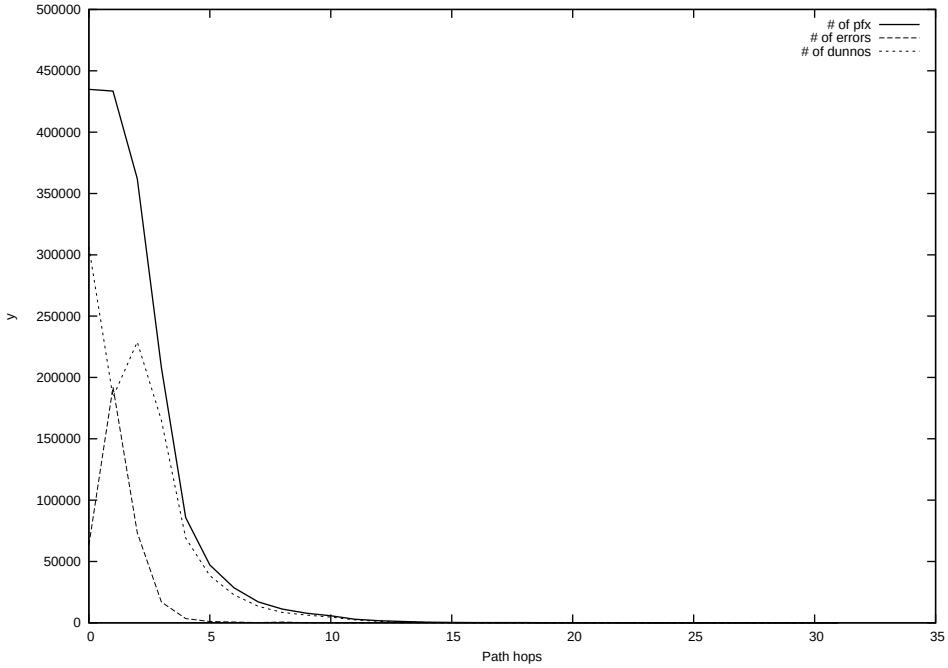
2012-12-28



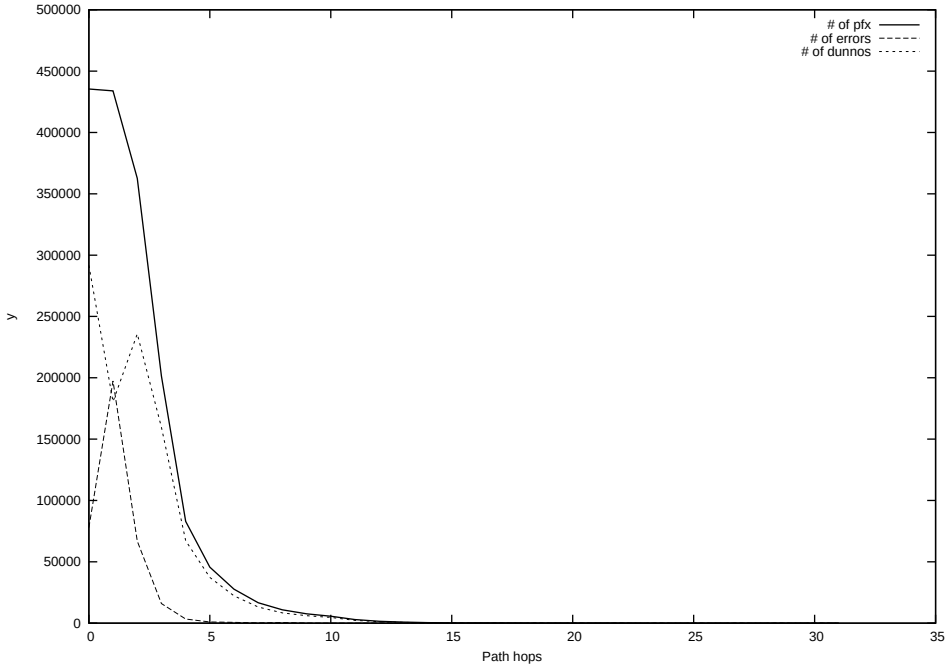
2012-12-29



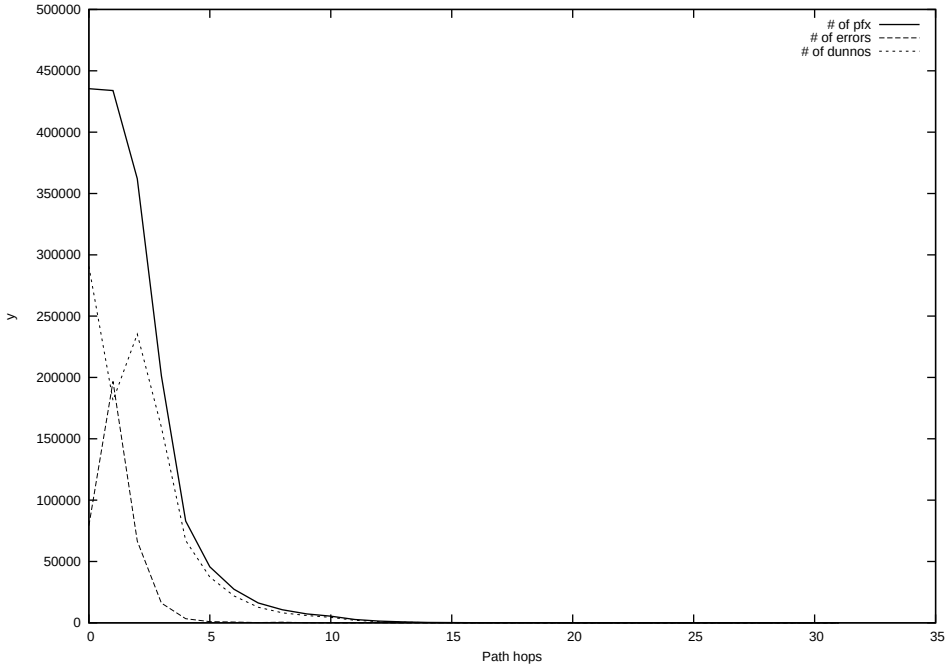
2012-12-30



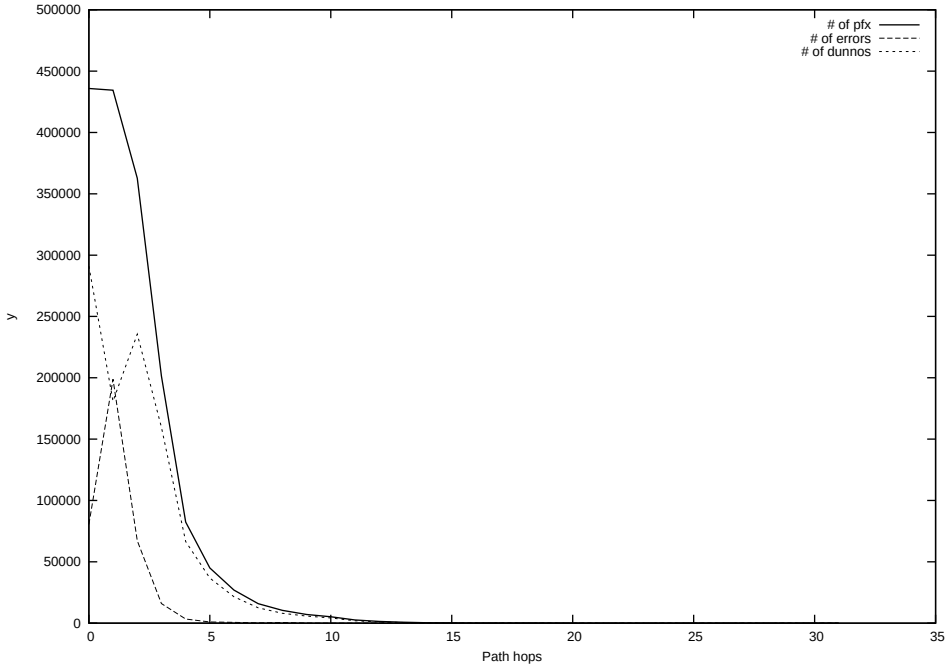
2012-12-31



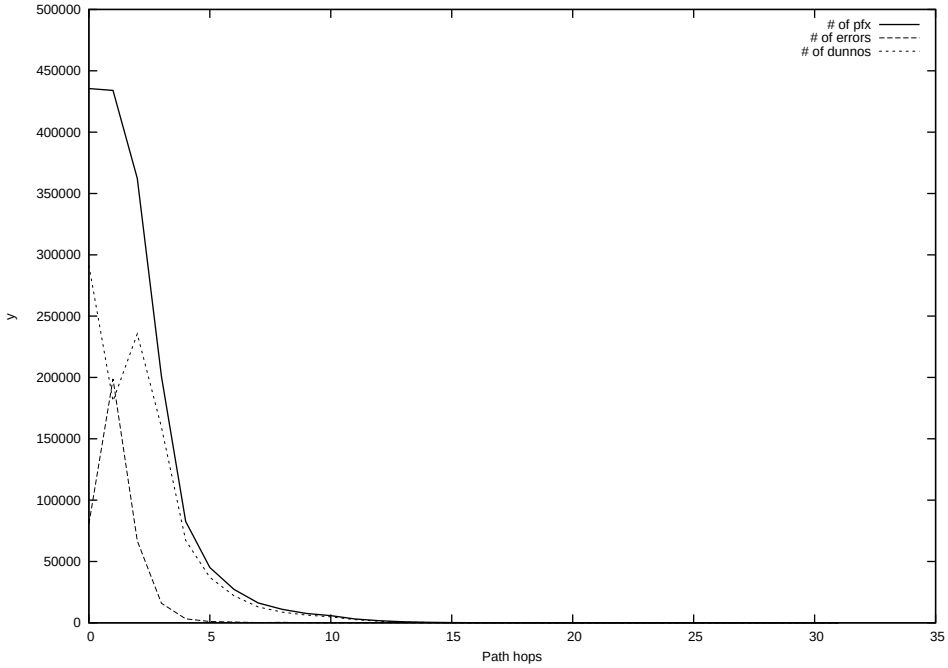
2013-01-01



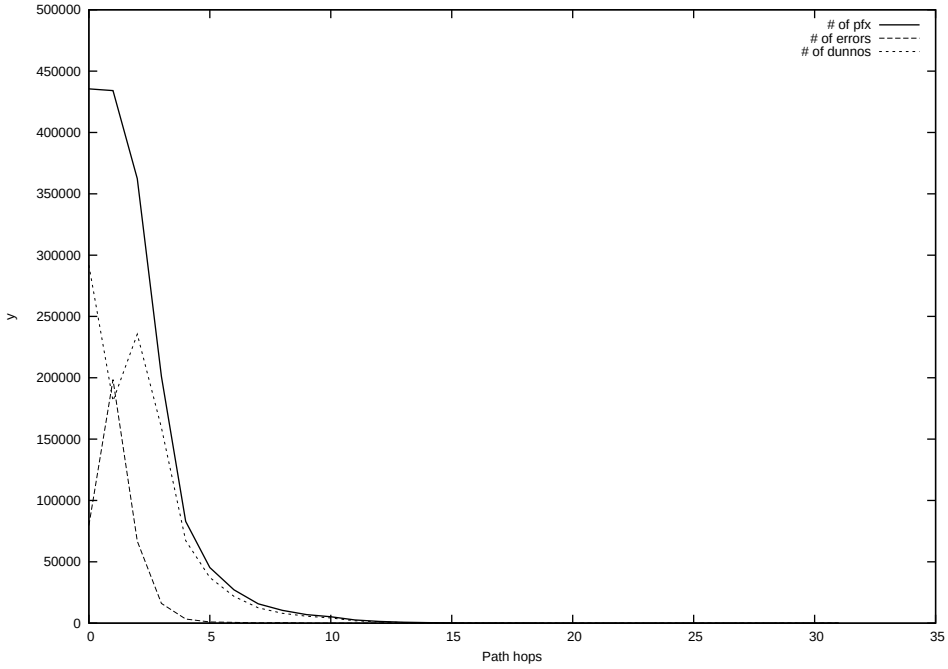
2013-01-02



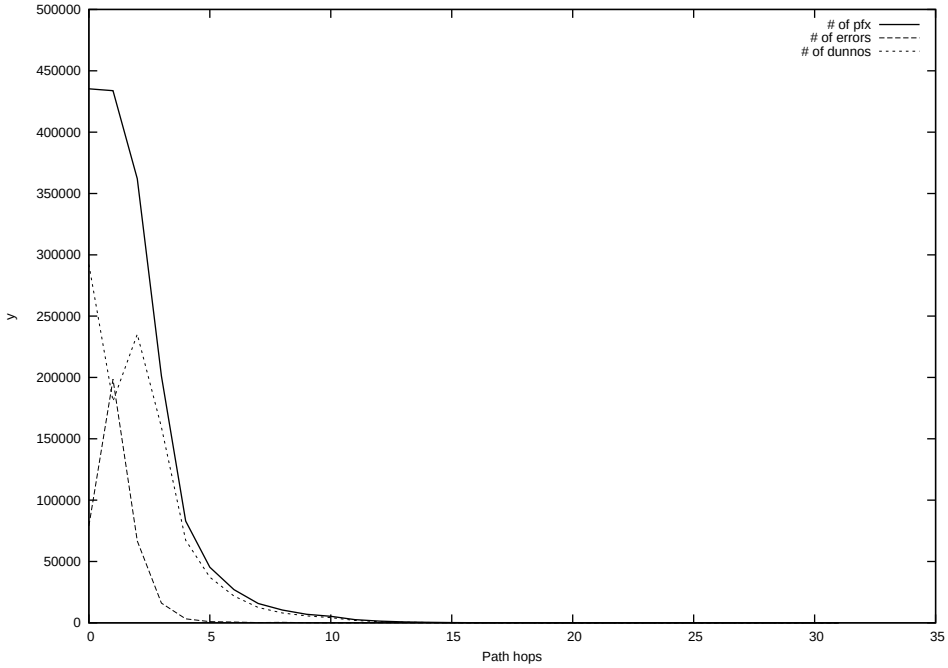
2013-01-03



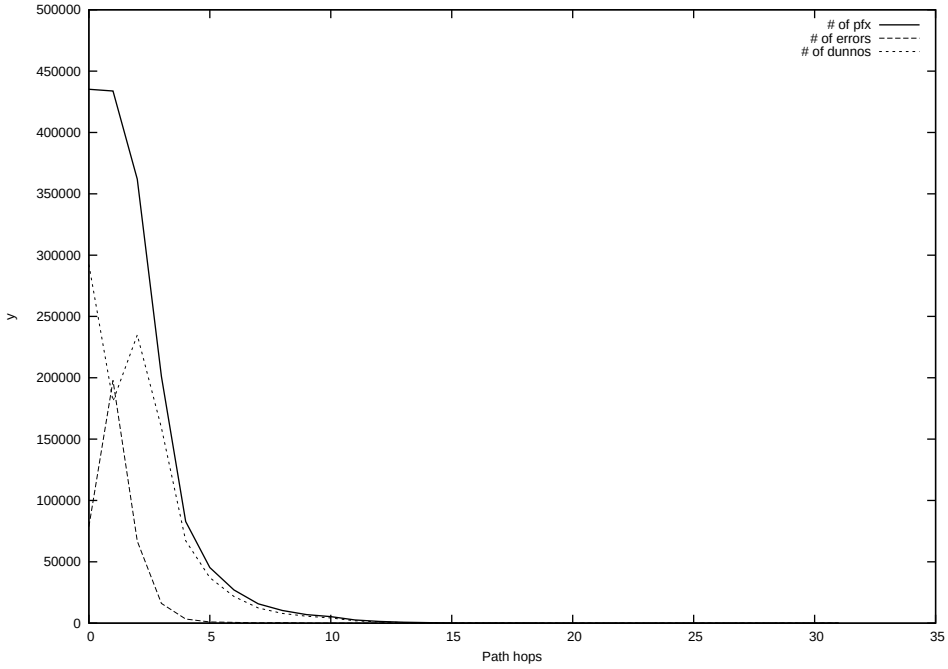
2013-01-04



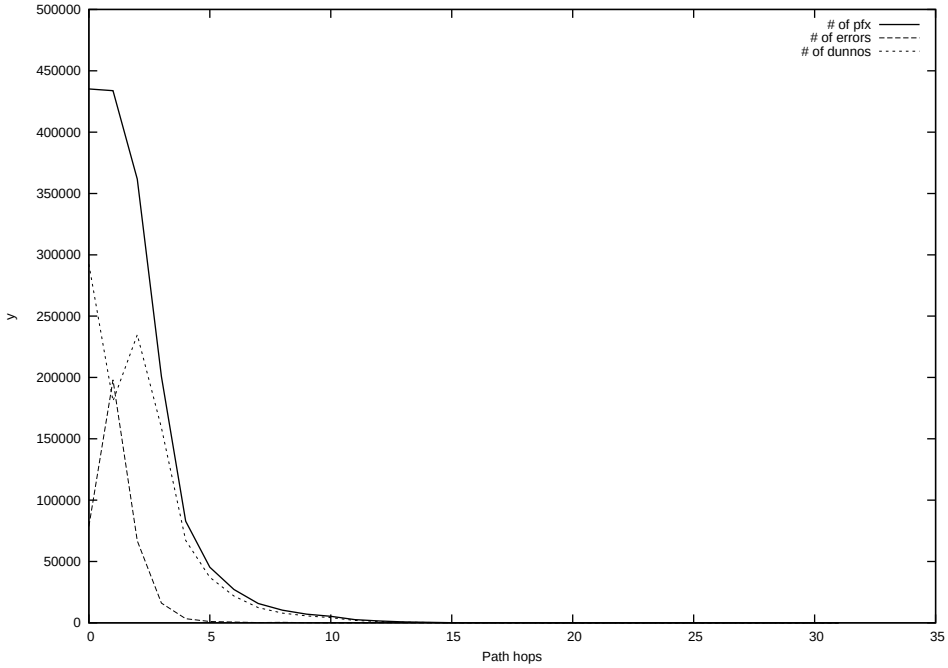
2013-01-05



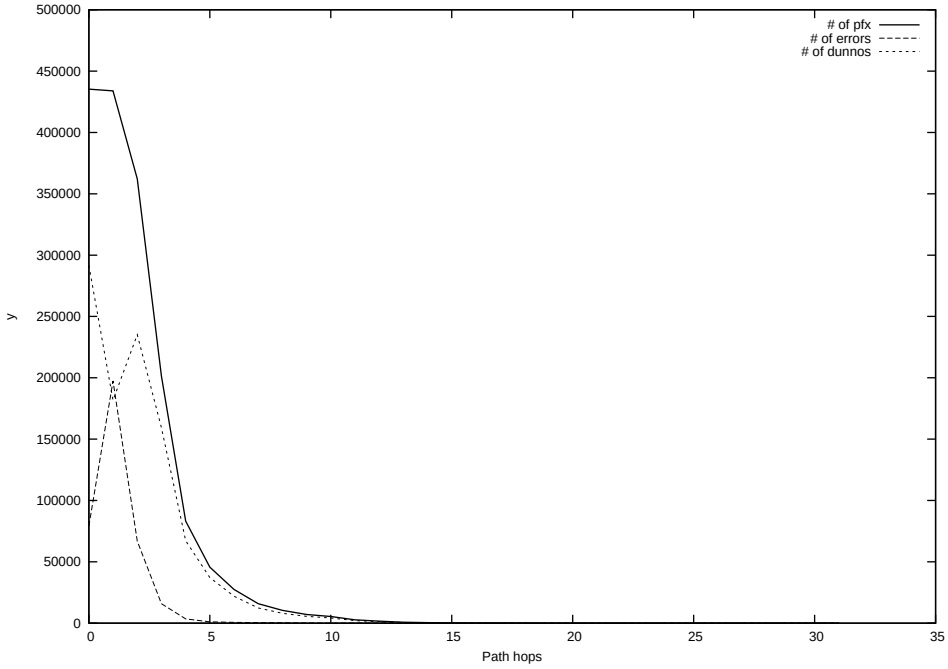
2013-01-06



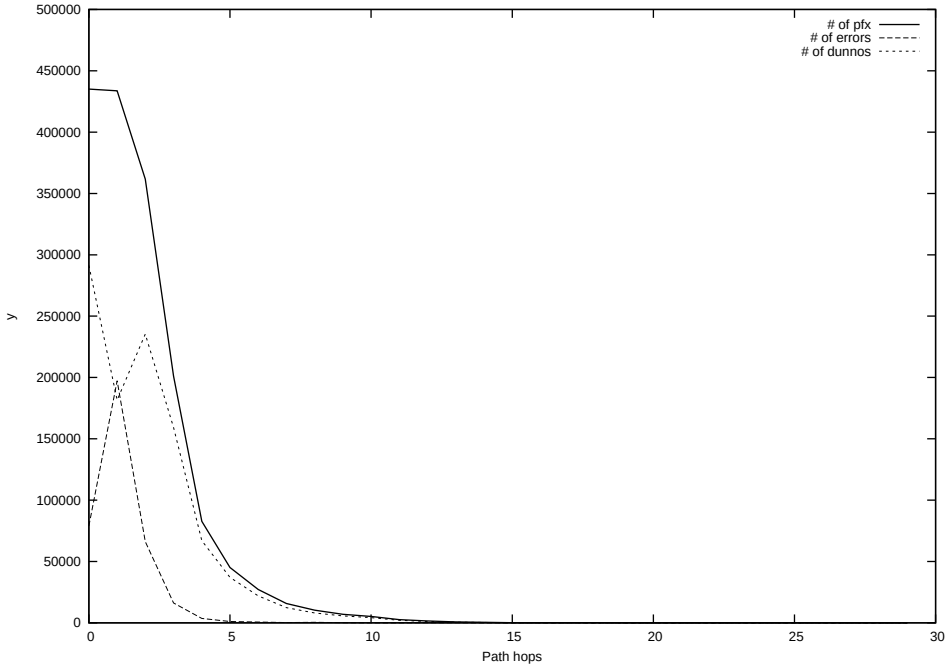
2013-01-07



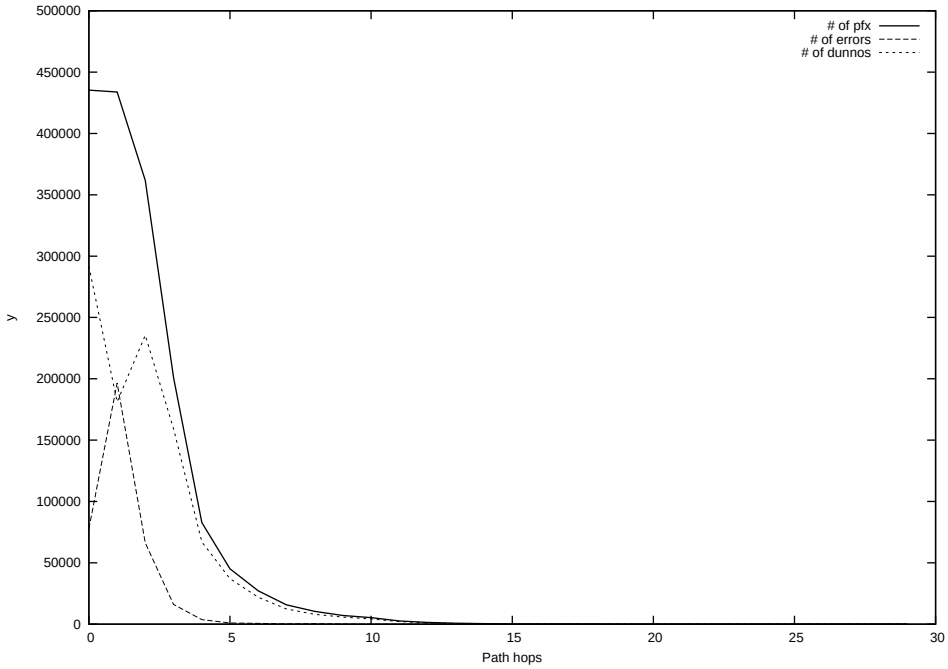
2013-01-08



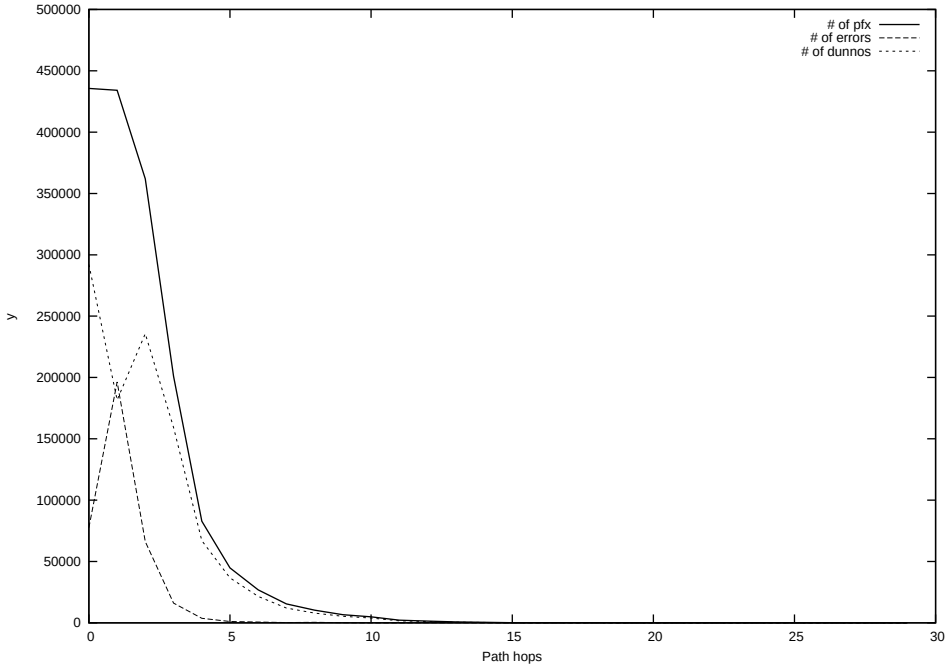
2013-01-09



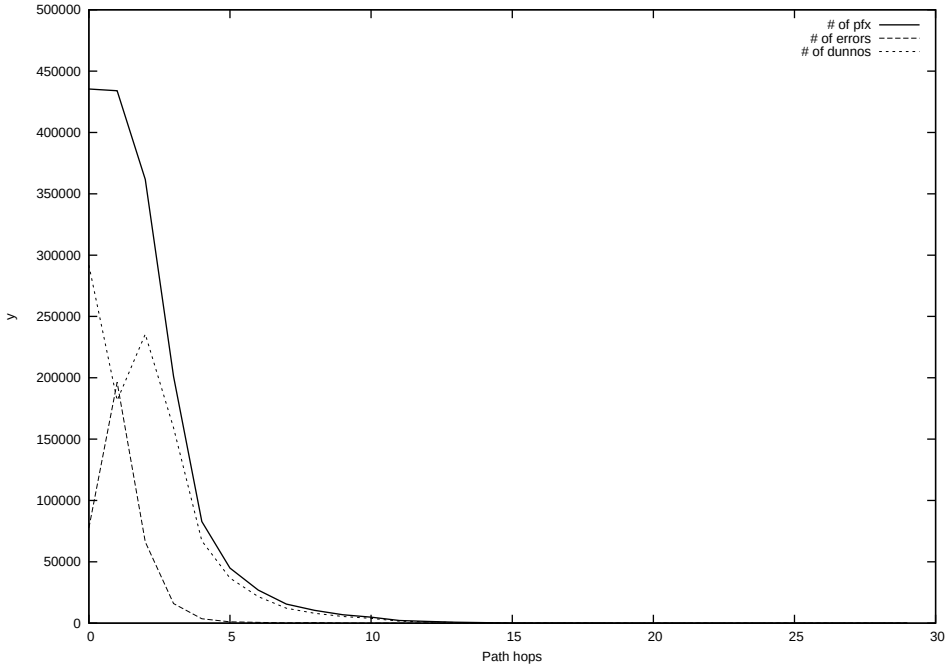
2013-01-10



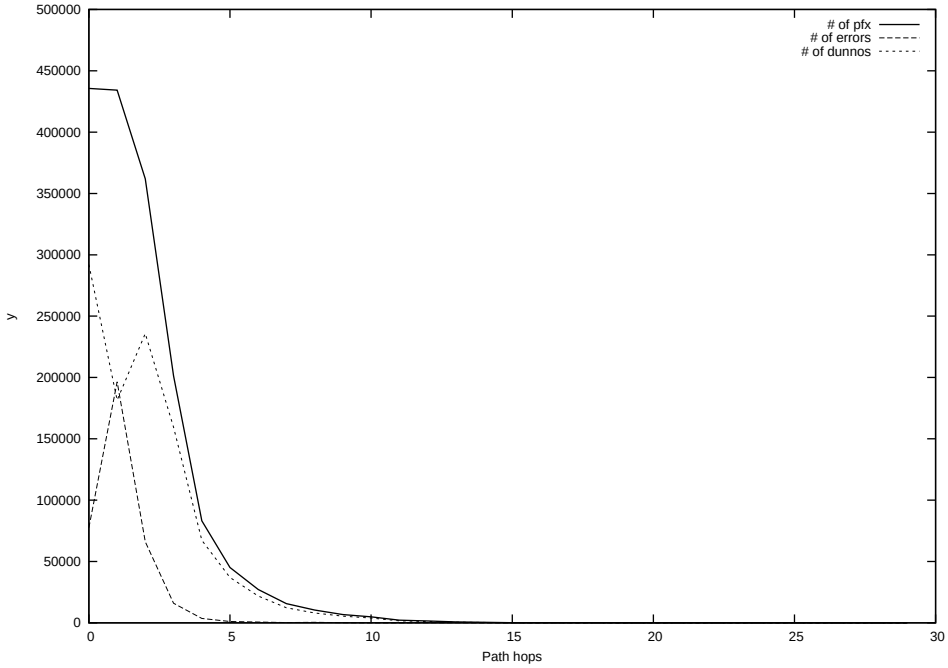
2013-01-11



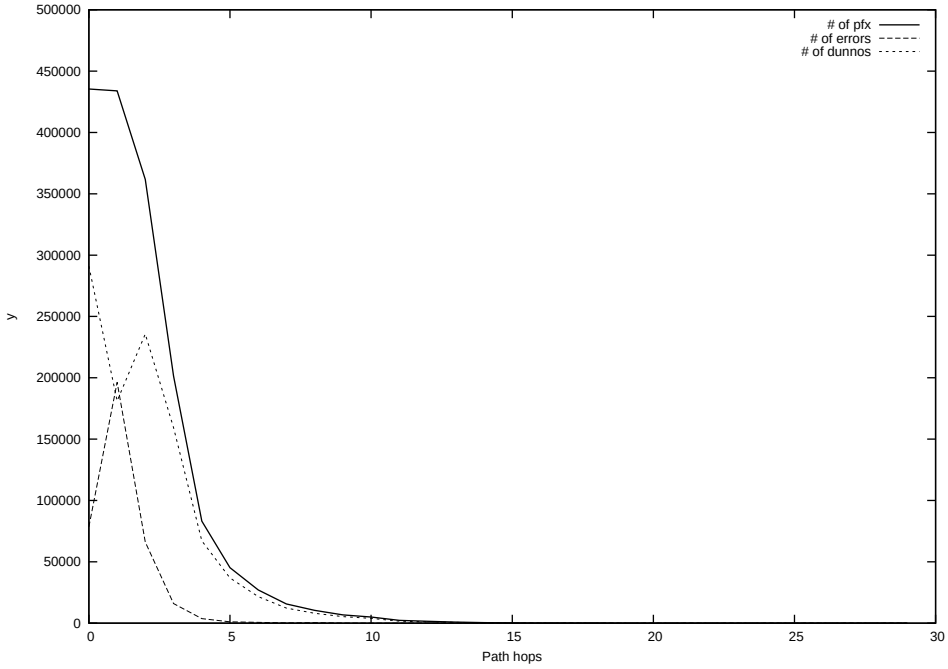
2013-01-12



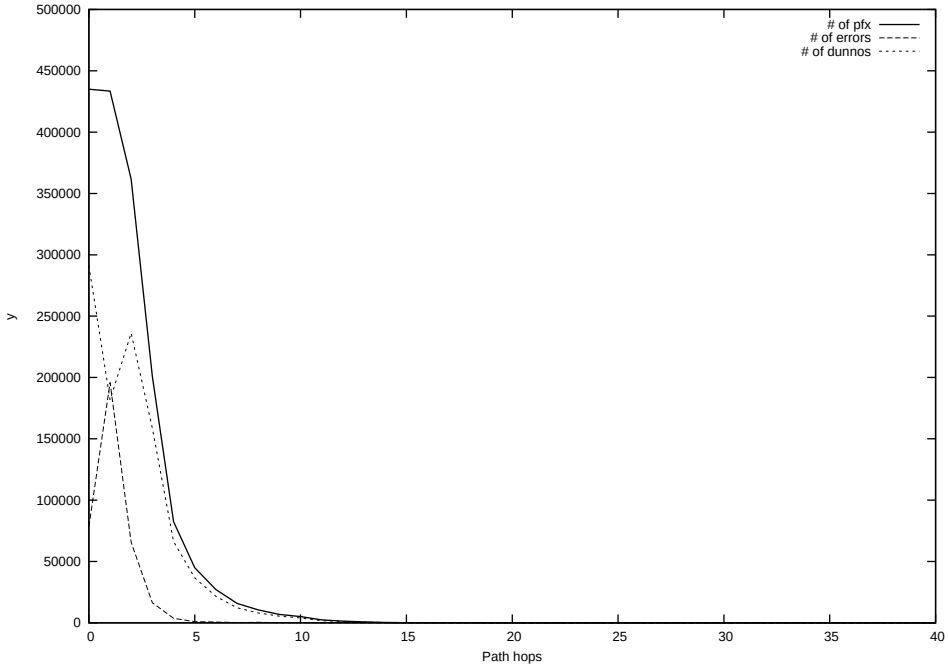
2013-01-13



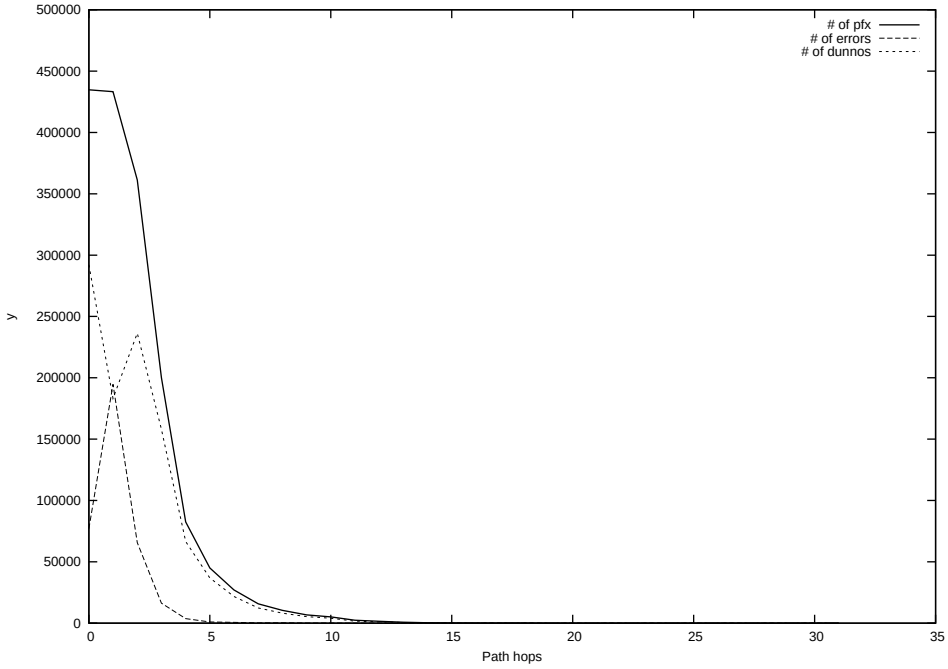
2013-01-14



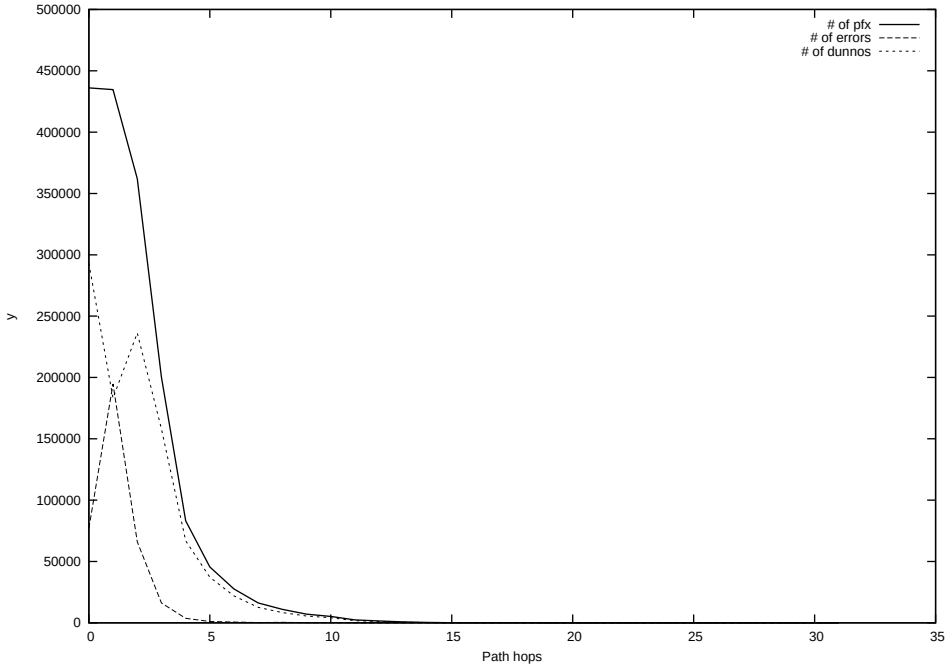
2013-01-15



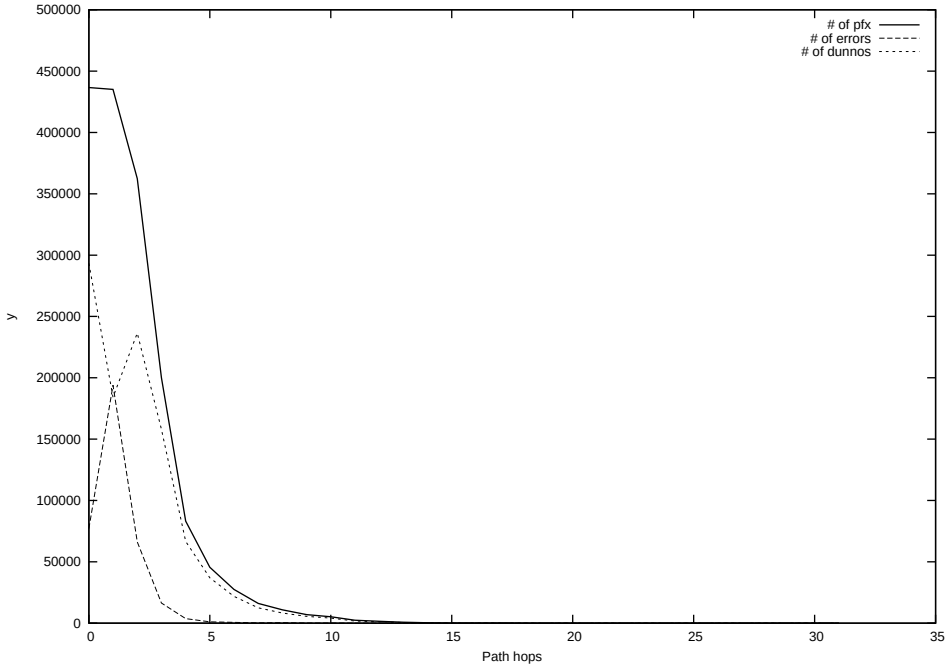
2013-01-16



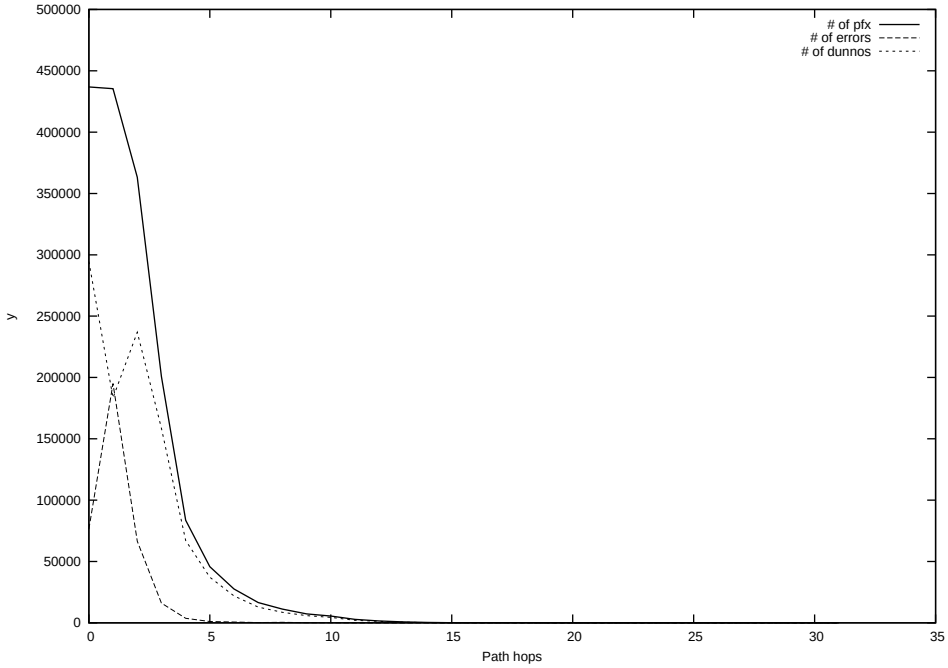
2013-01-17



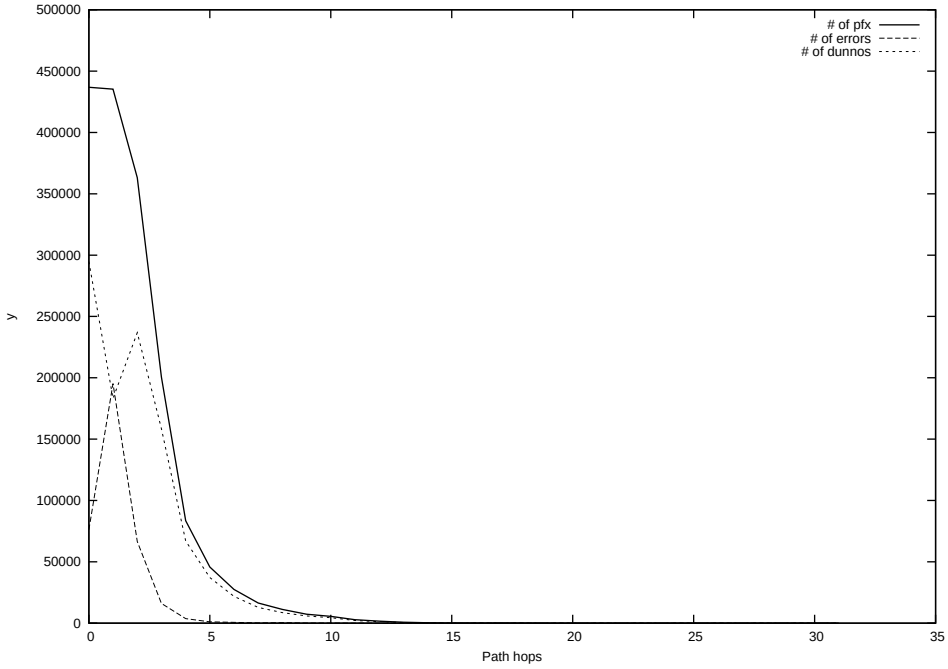
2013-01-18



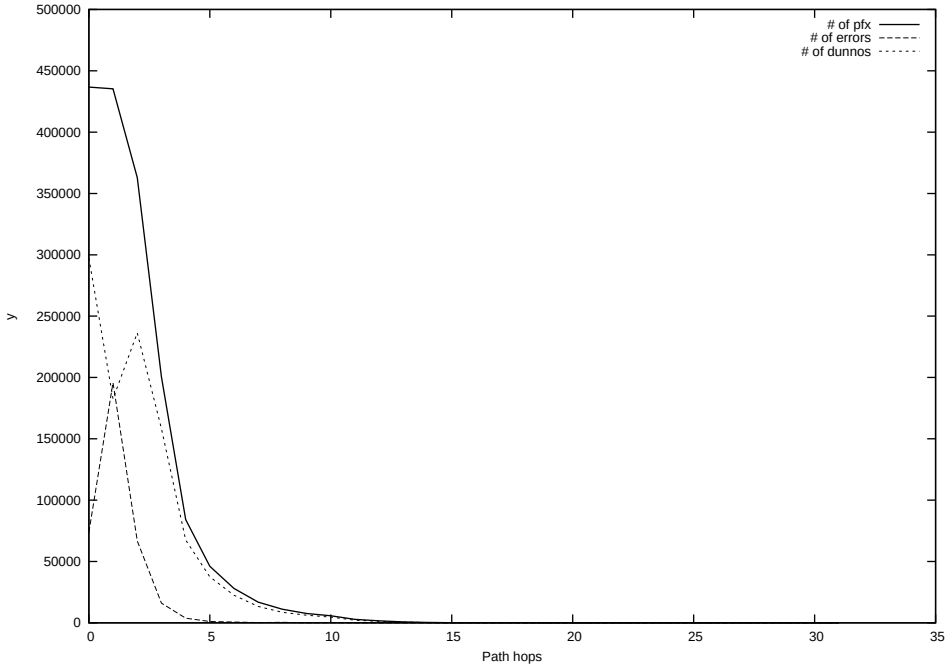
2013-01-19



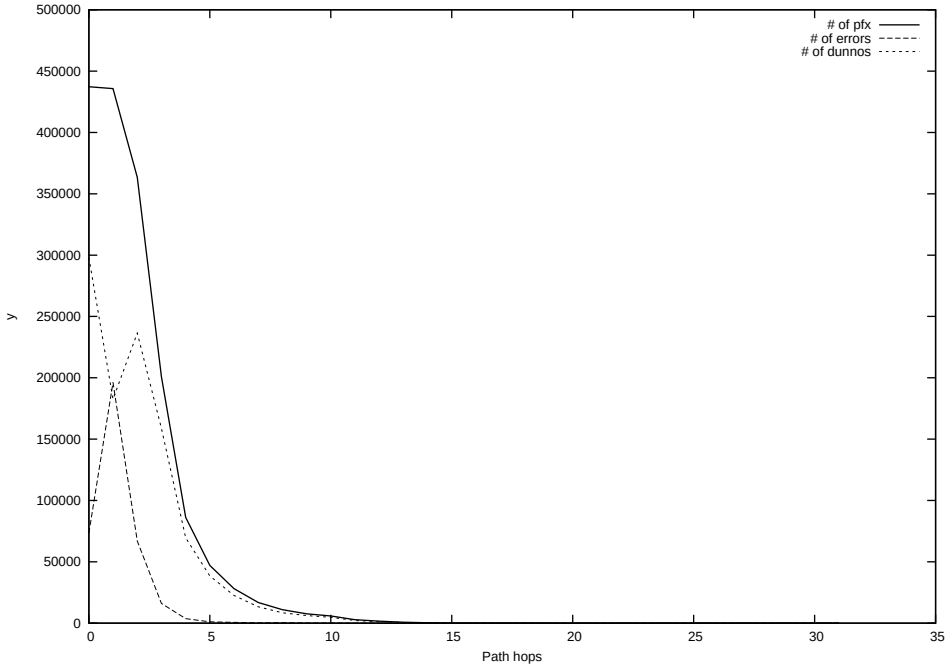
2013-01-20



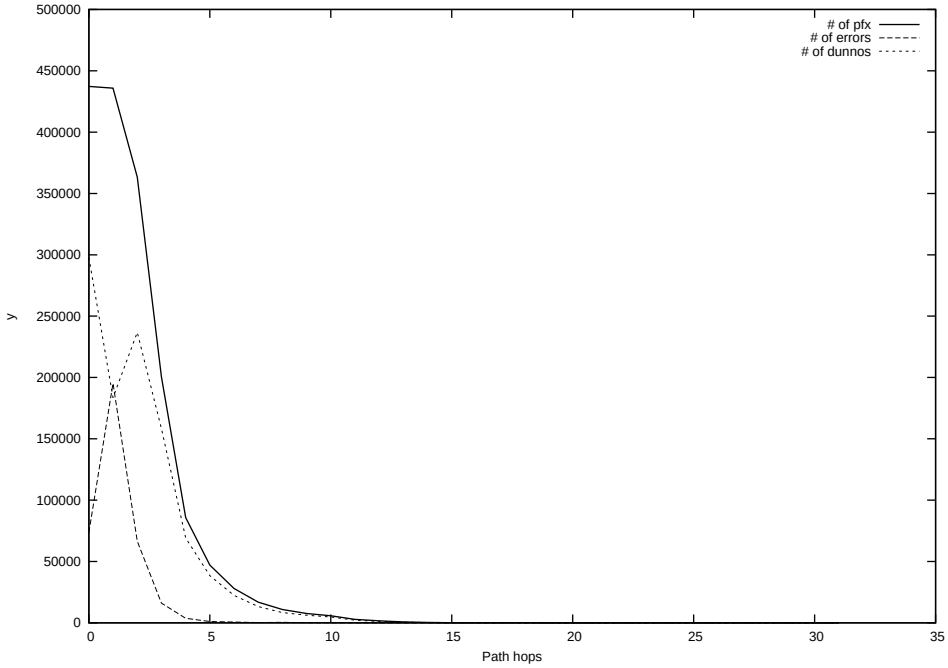
2013-01-21



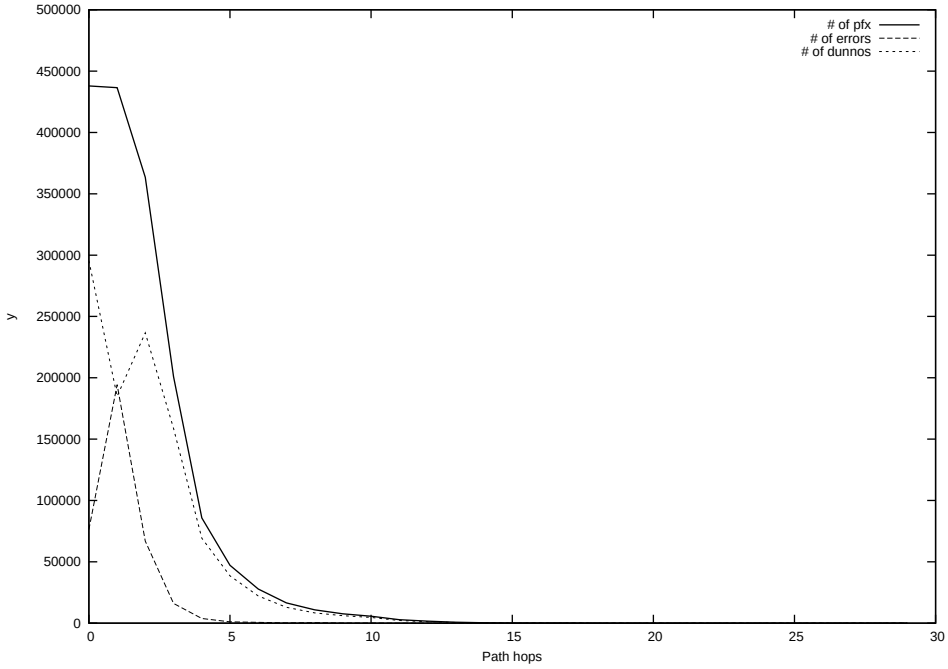
2013-01-22



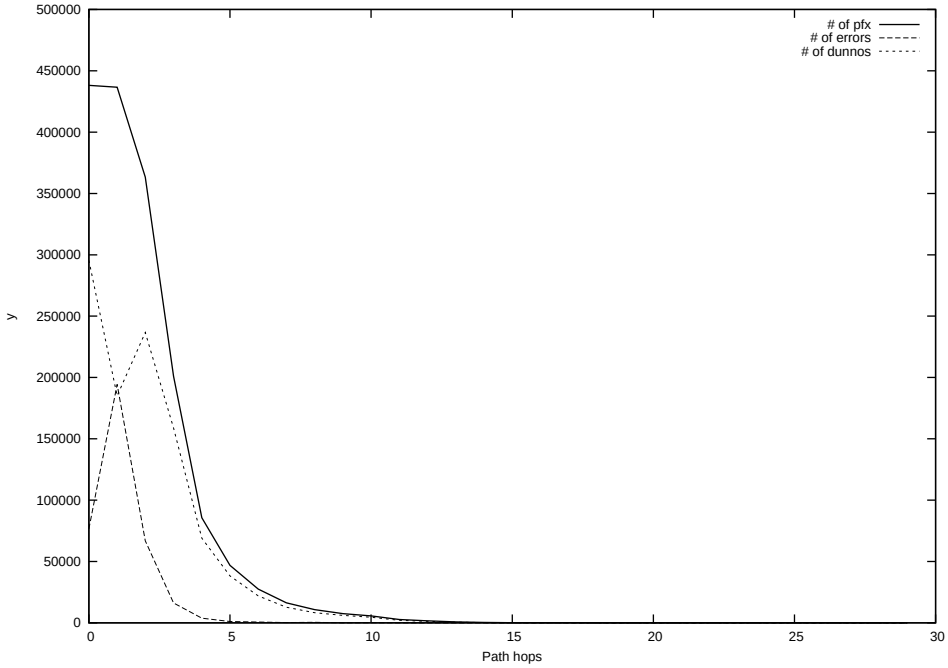
2013-01-23



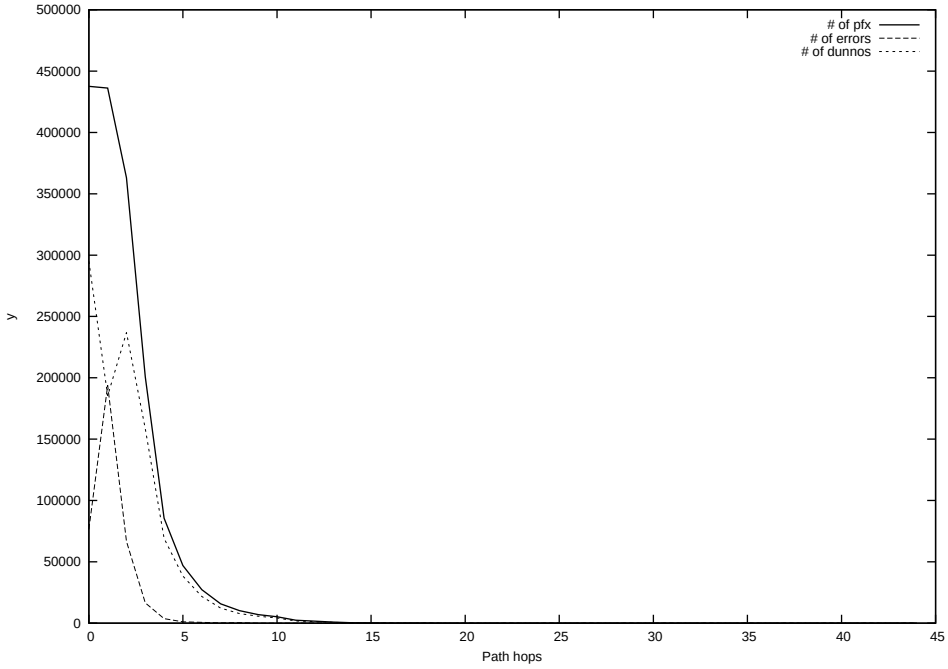
2013-01-24



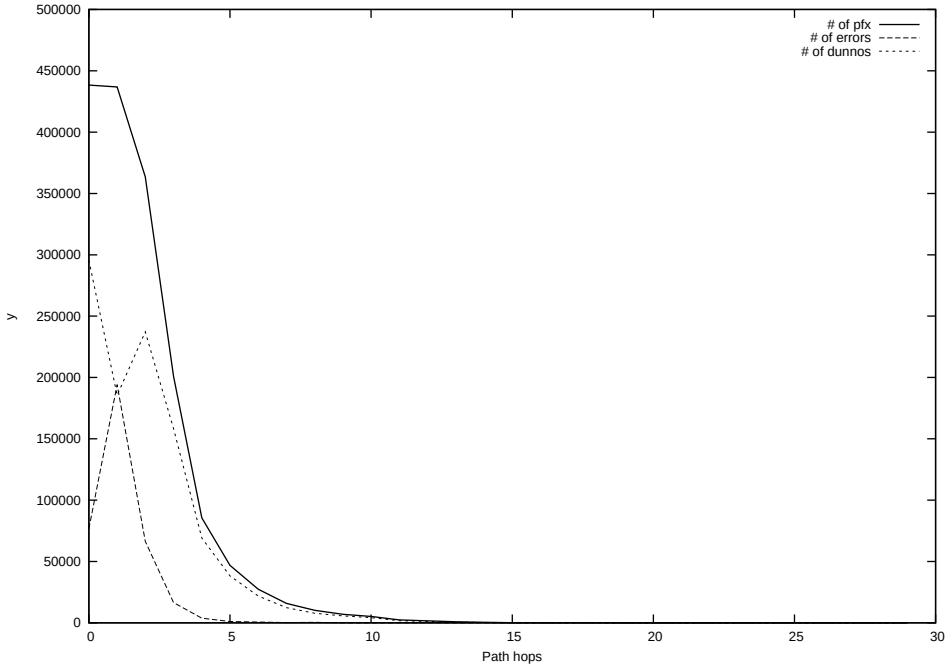
2013-01-25



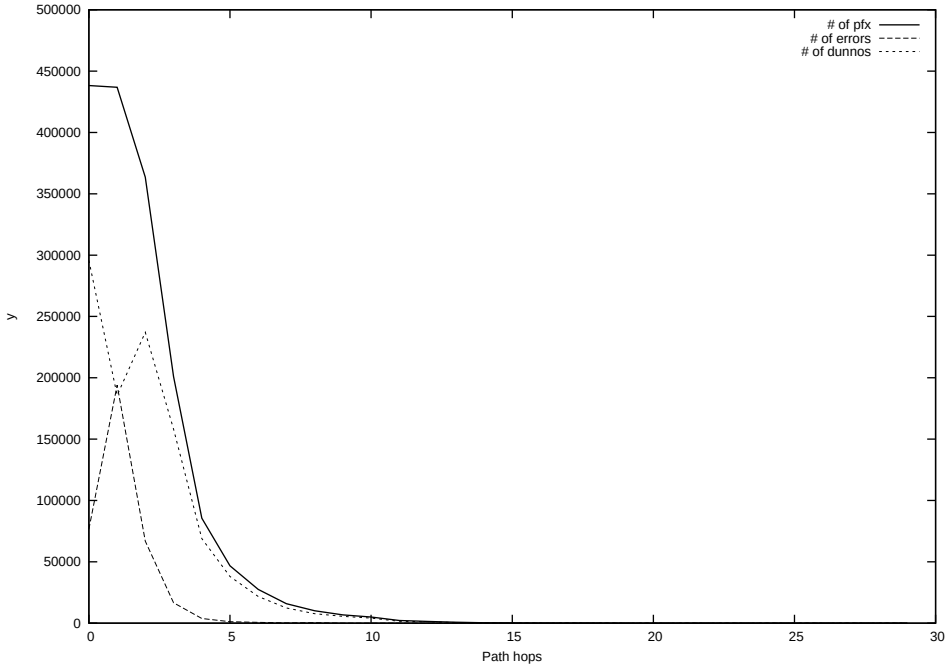
2013-01-26



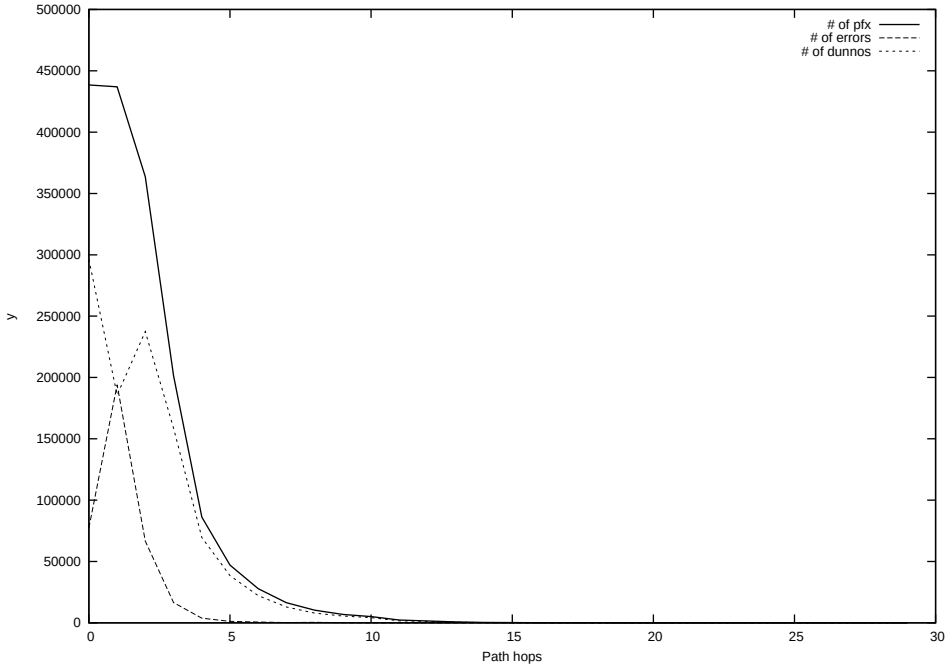
2013-01-27



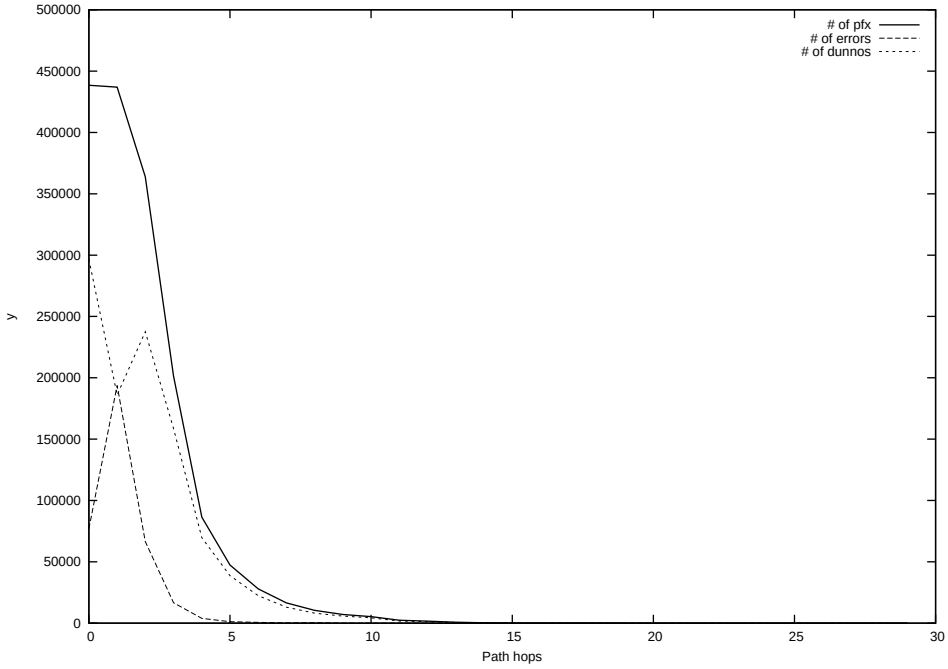
2013-01-28



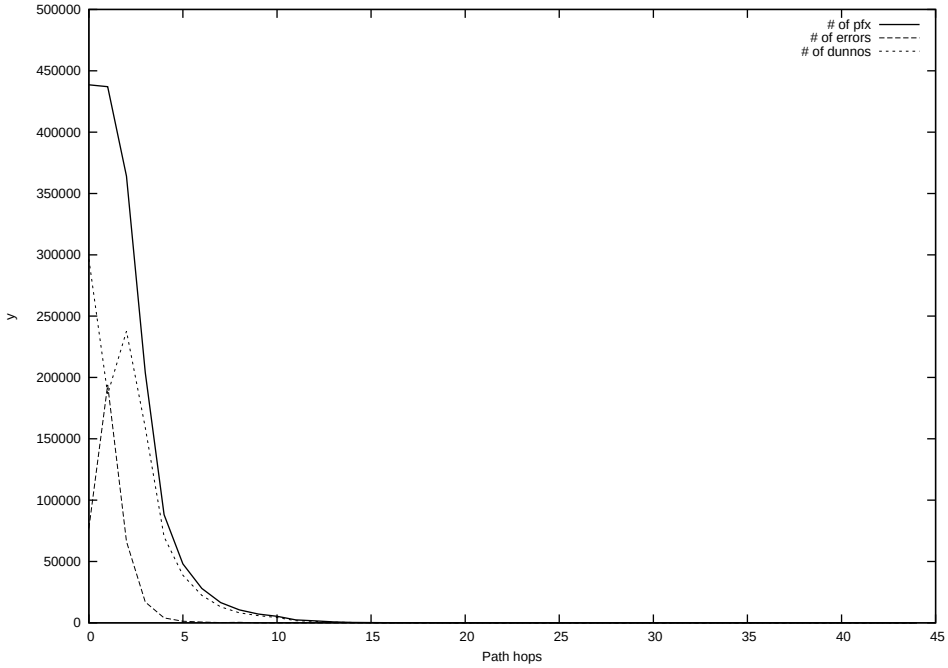
2013-01-29



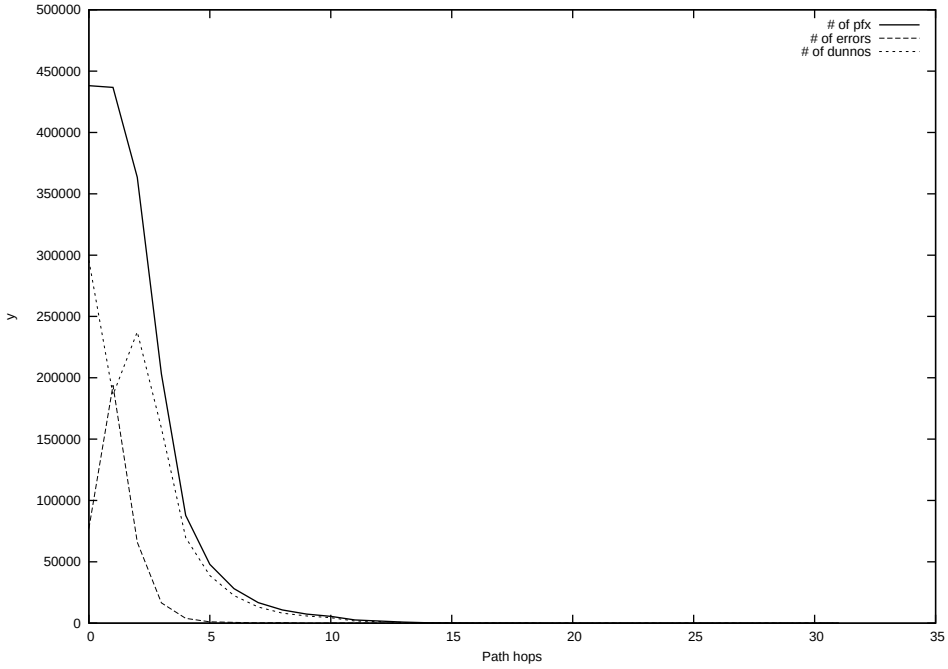
2013-01-30



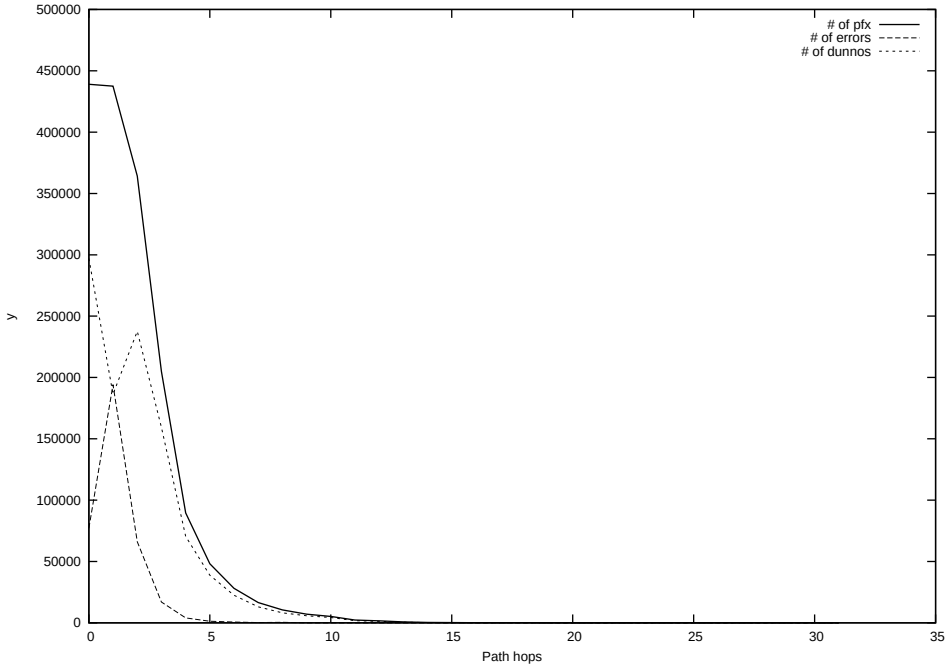
2013-01-31



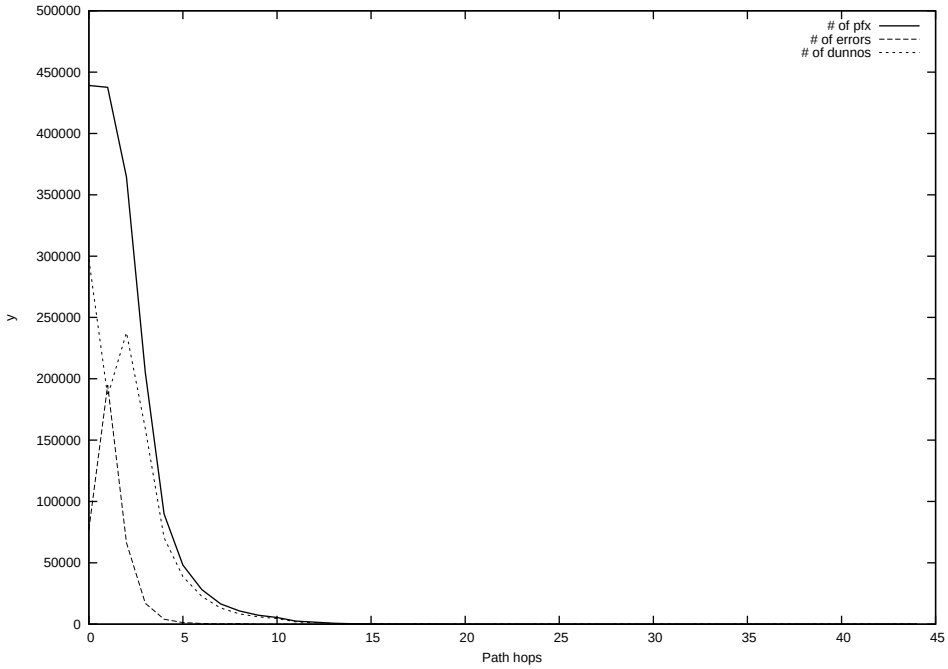
2013-02-01



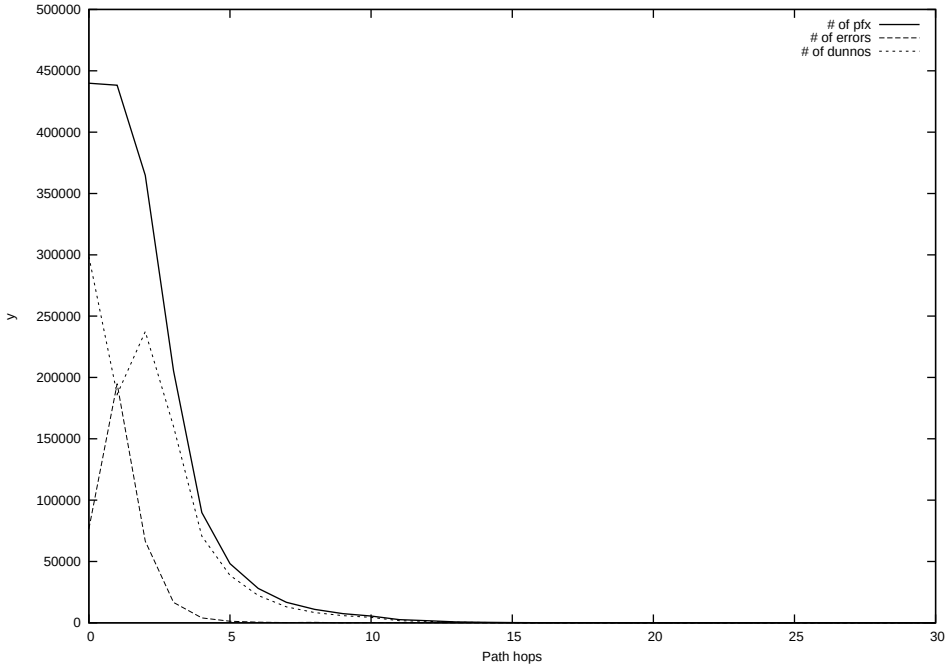
2013-02-02



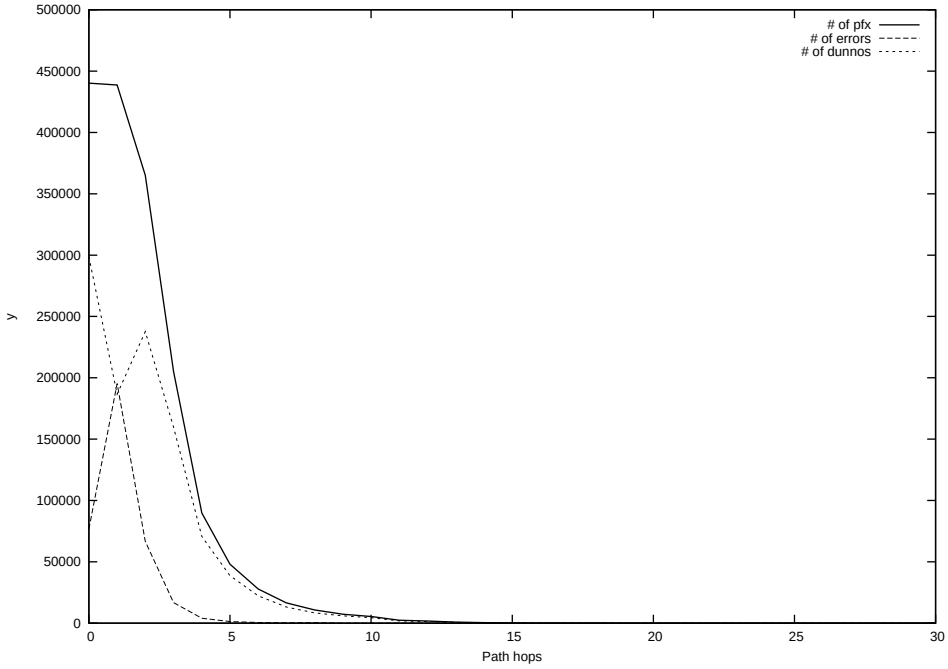
2013-02-03



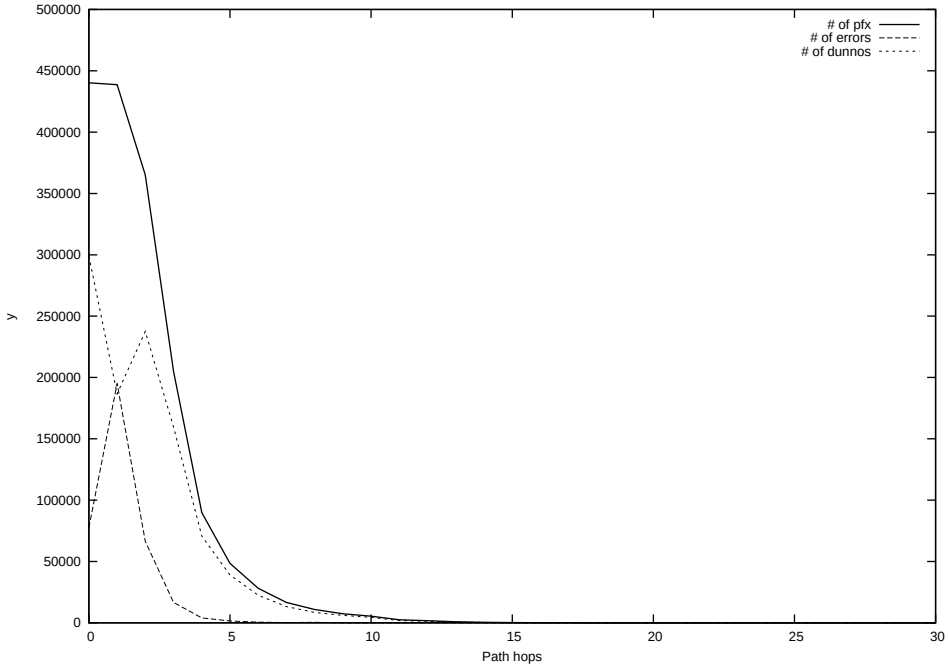
2013-02-04



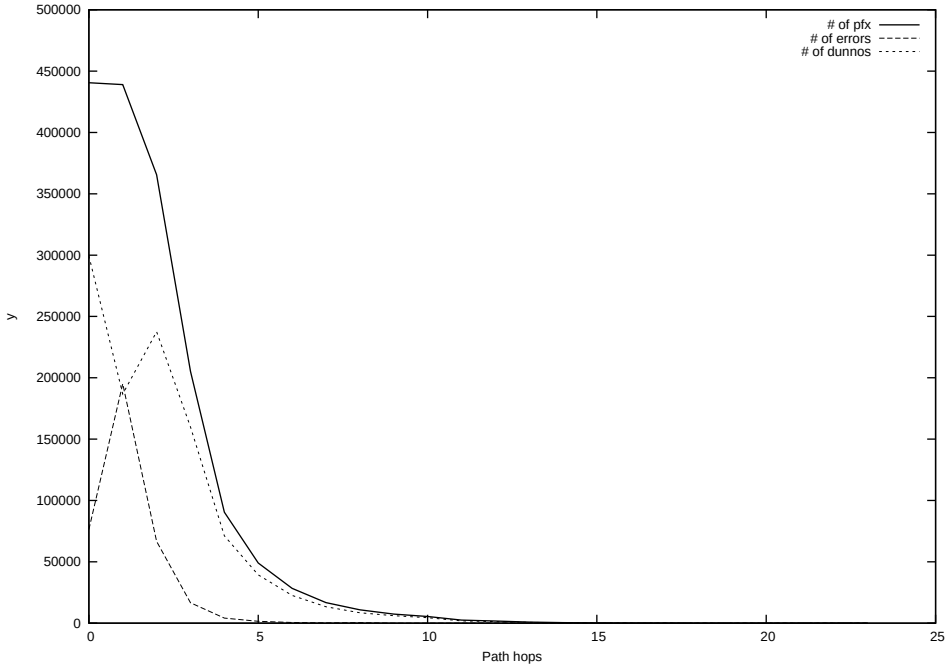
2013-02-05



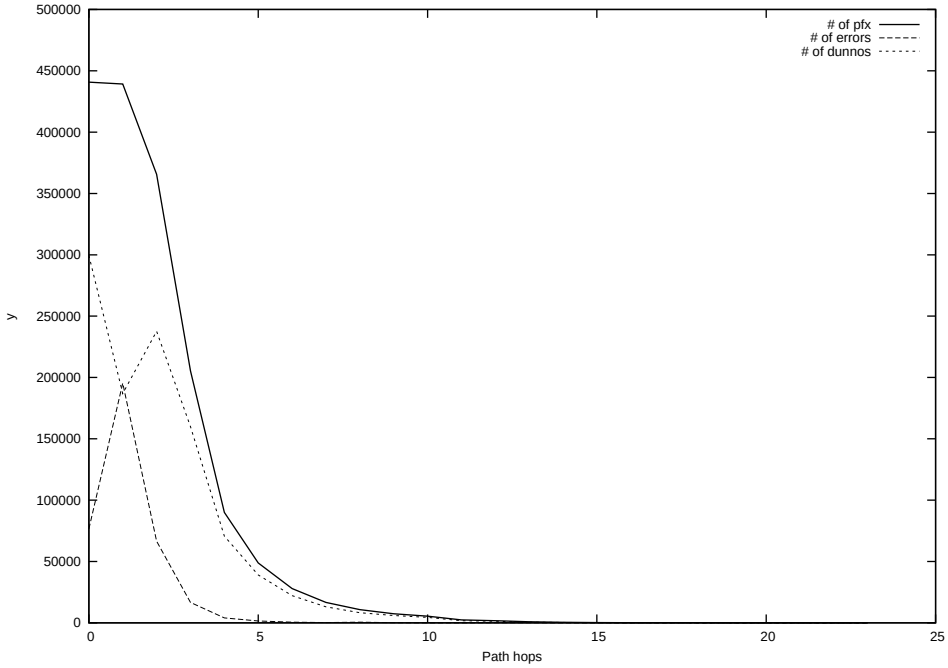
2013-02-06



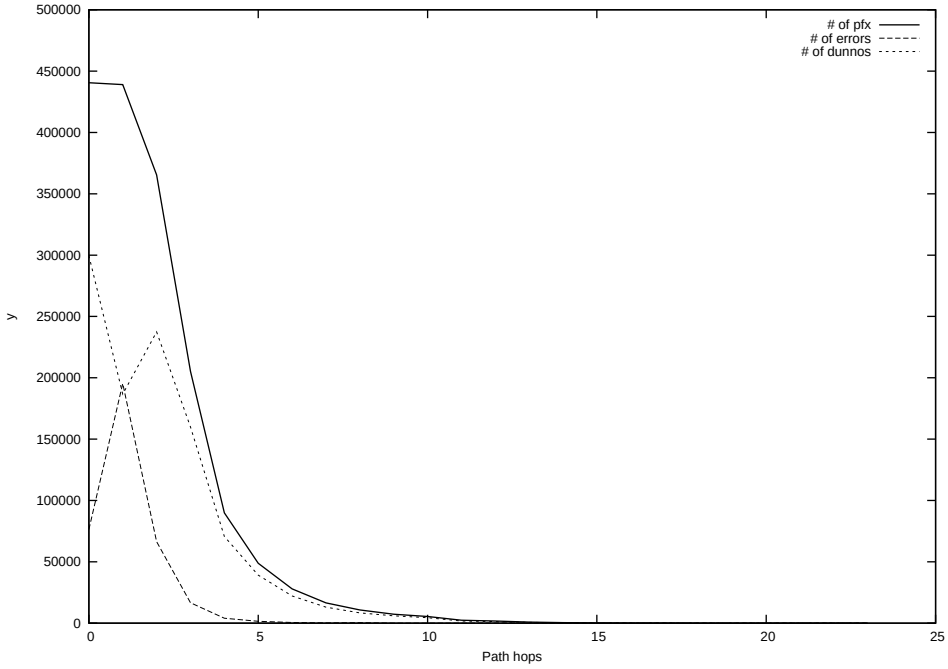
2013-02-07



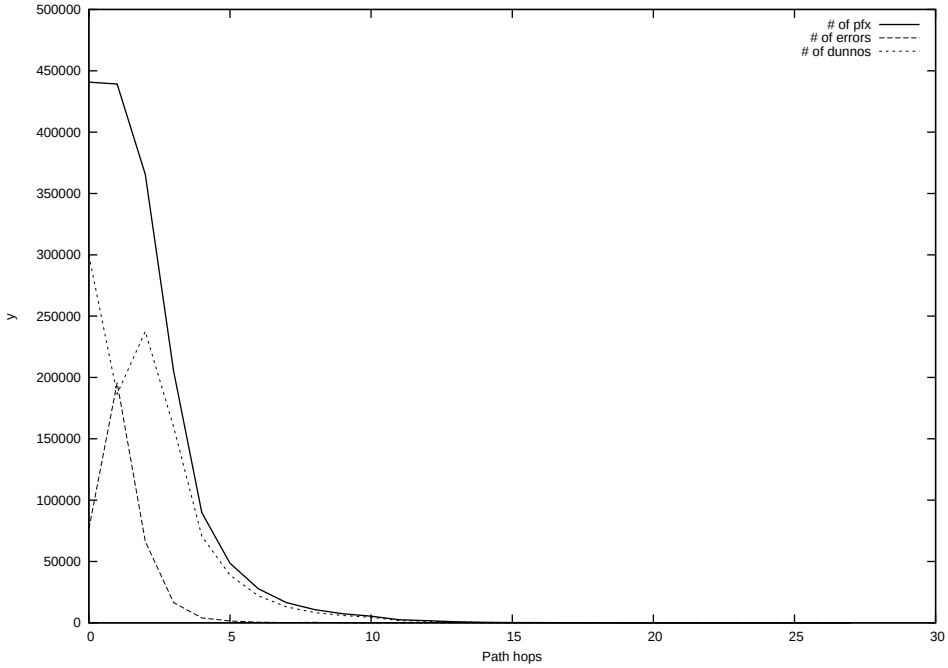
2013-02-08



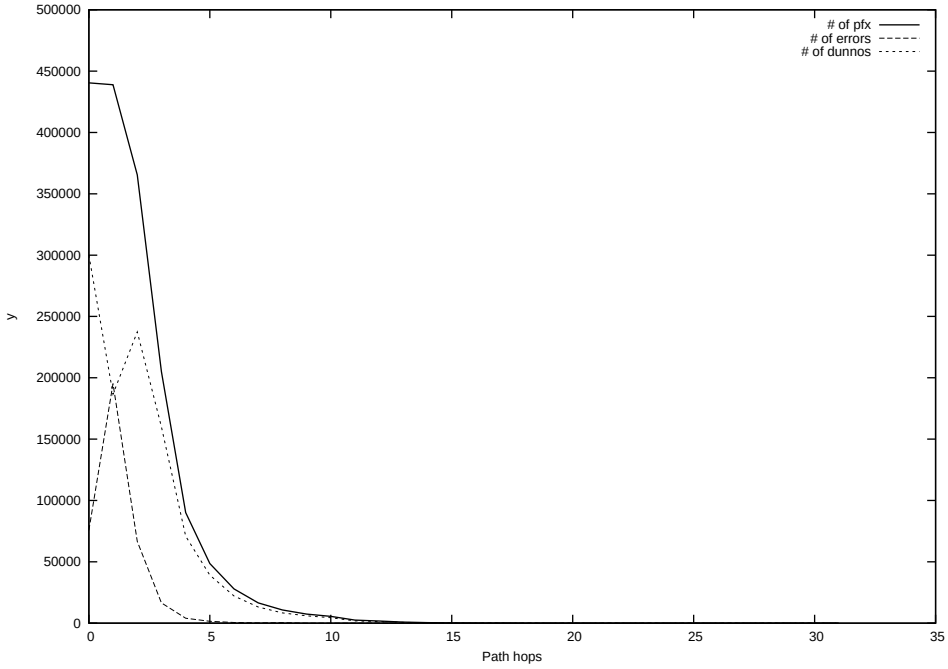
2013-02-09



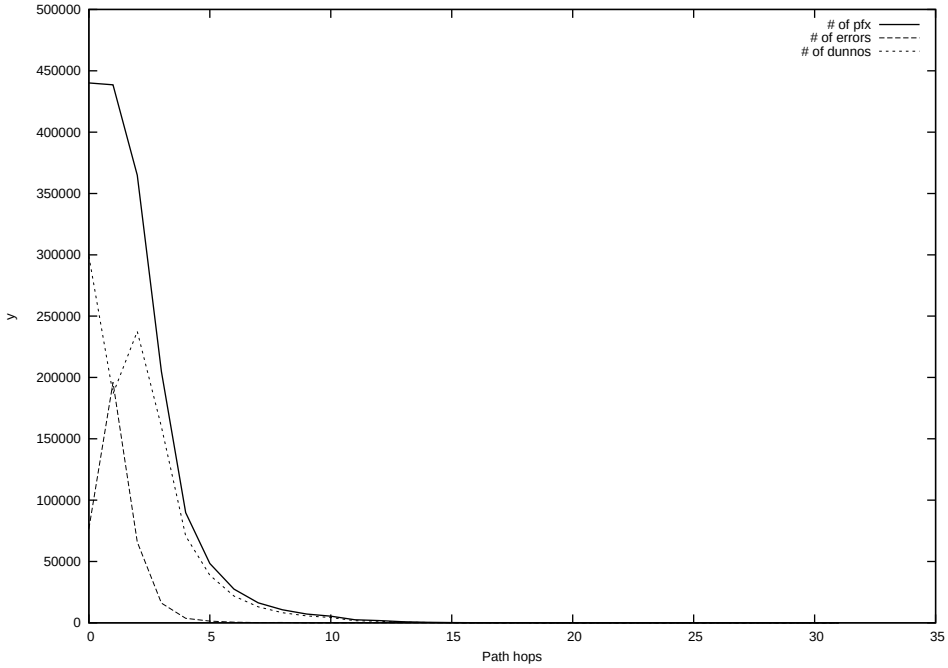
2013-02-10



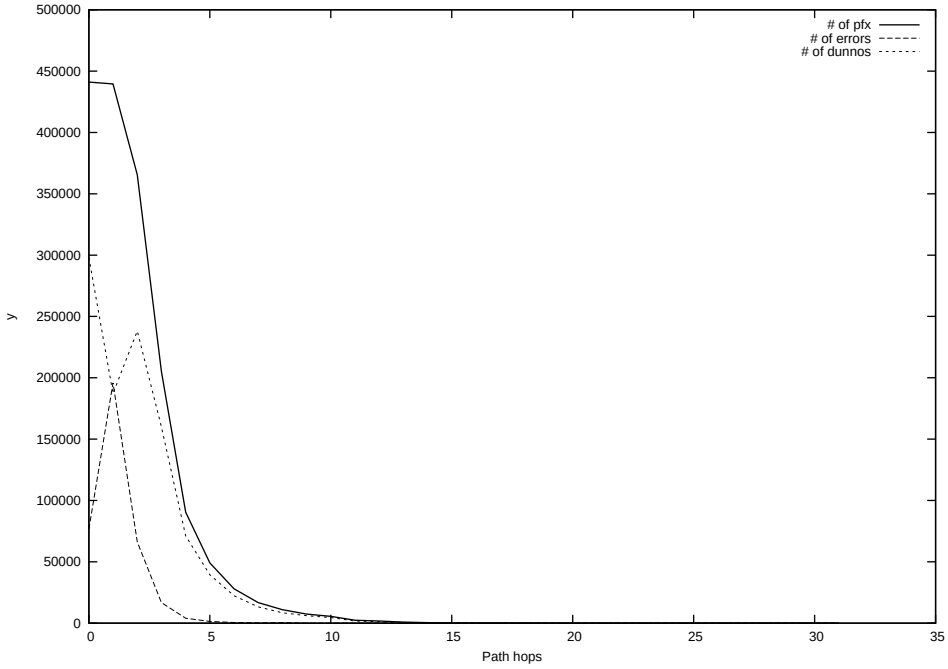
2013-02-11



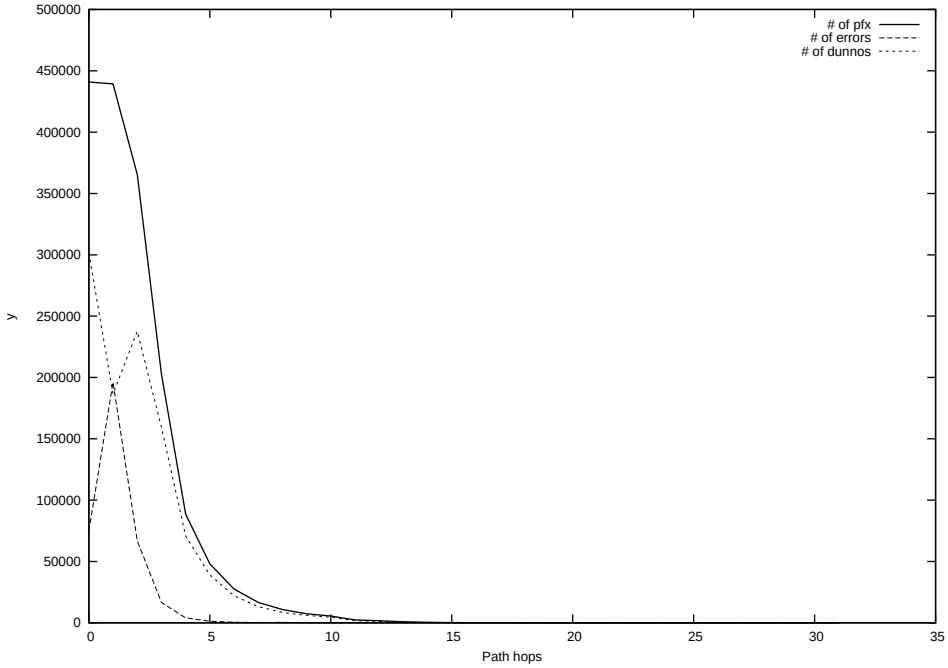
2013-02-12



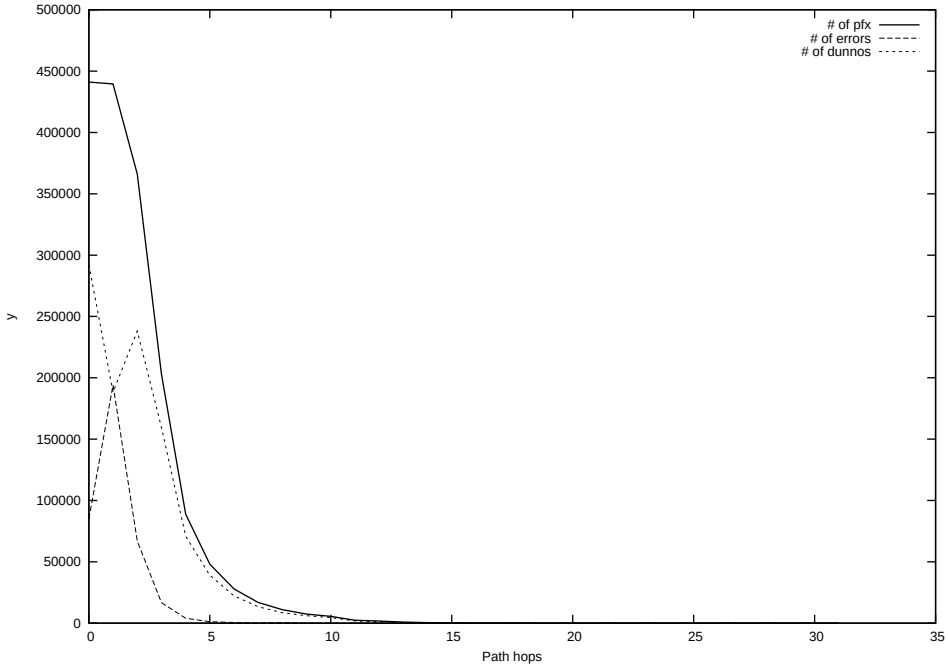
2013-02-13



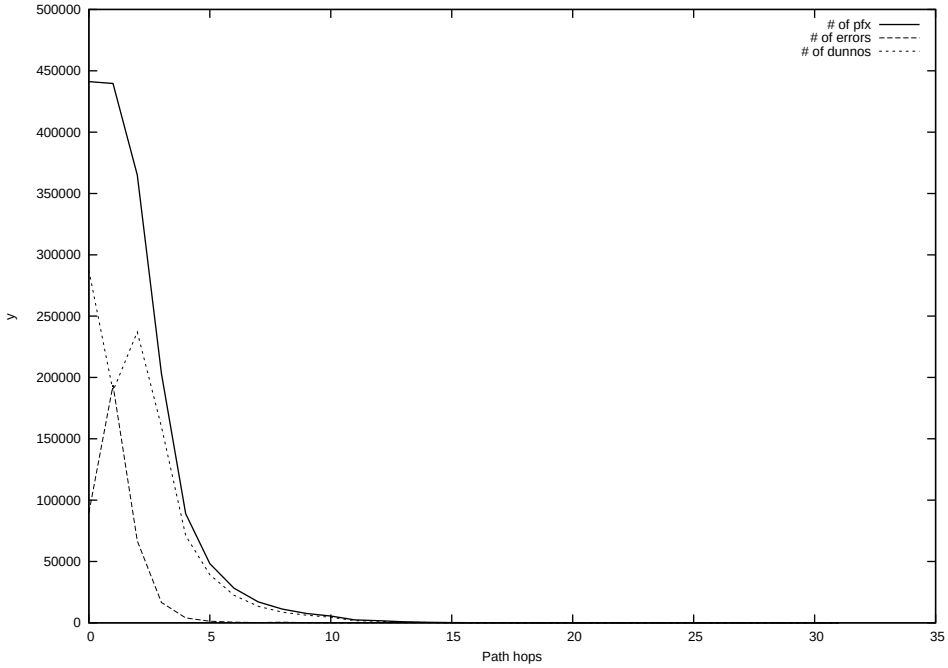
2013-02-14



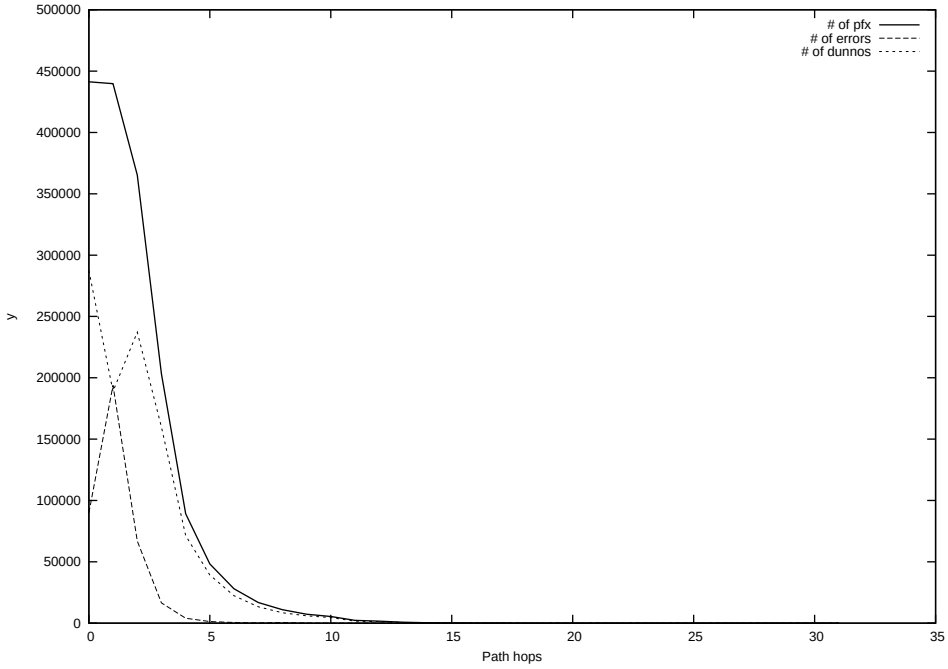
2013-02-15



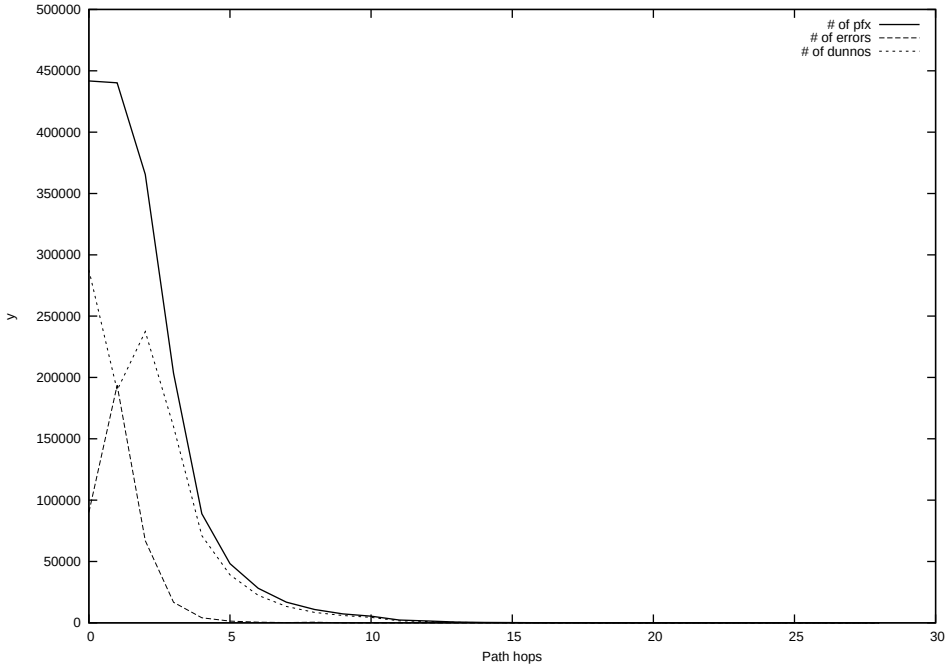
2013-02-16



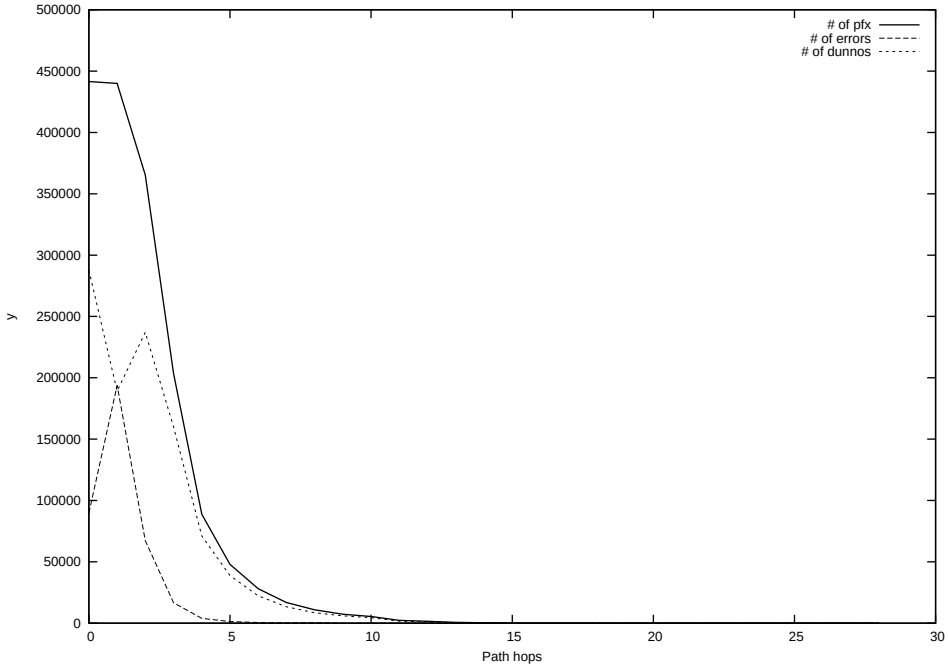
2013-02-17



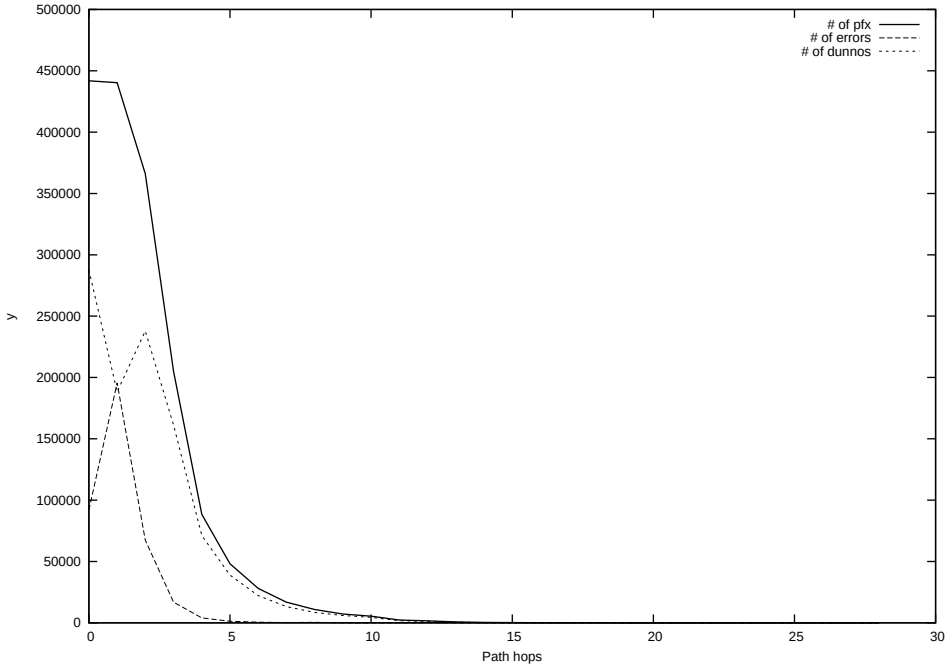
2013-02-18



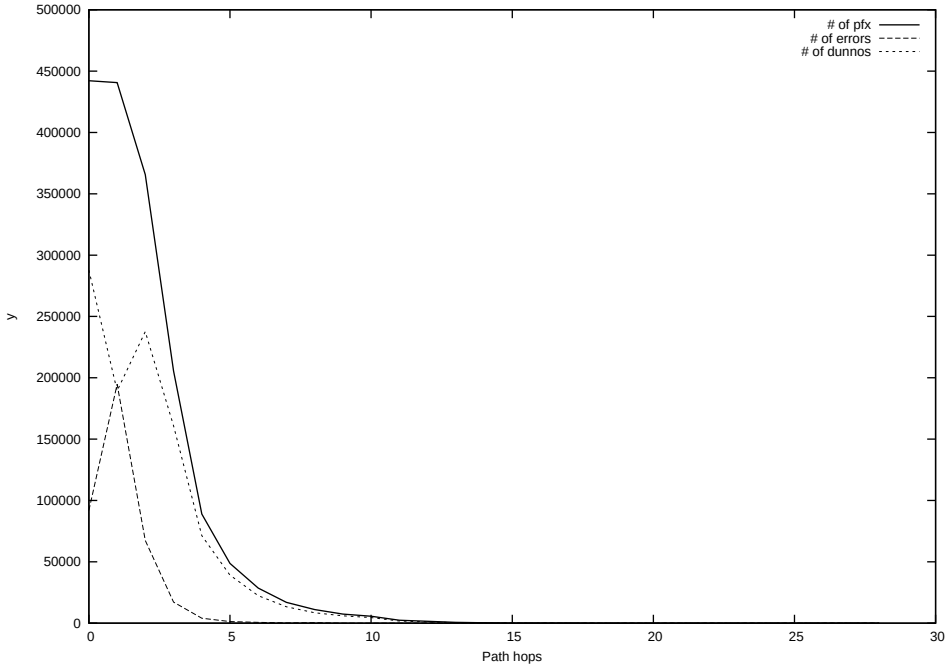
2013-02-19



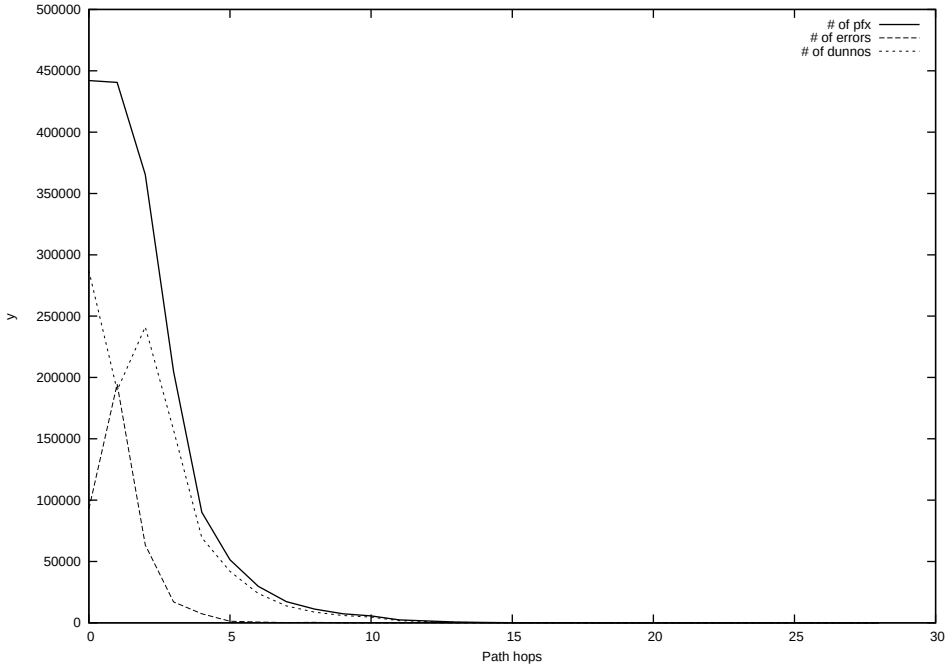
2013-02-20



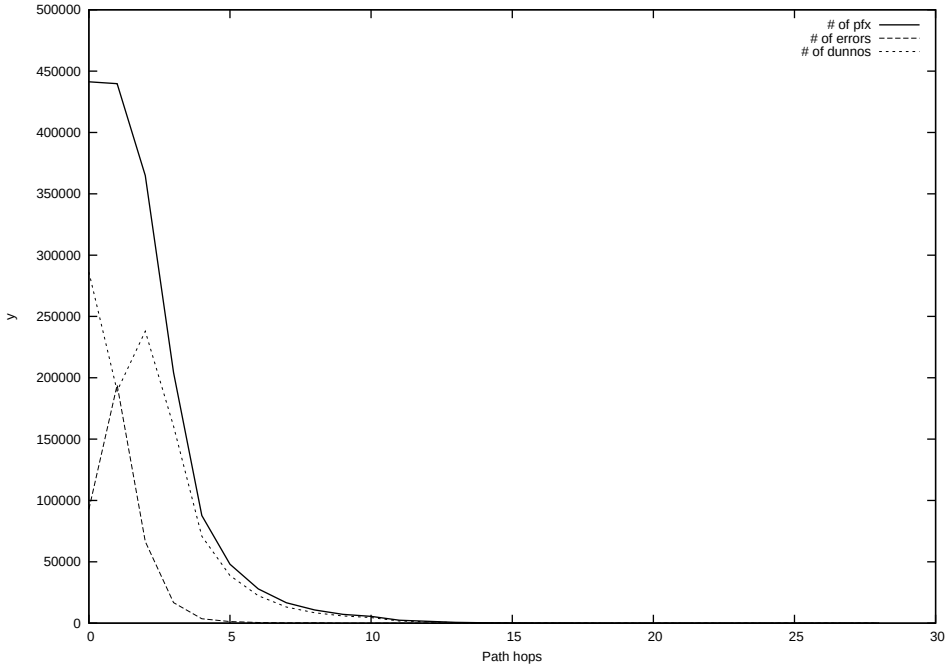
2013-02-21



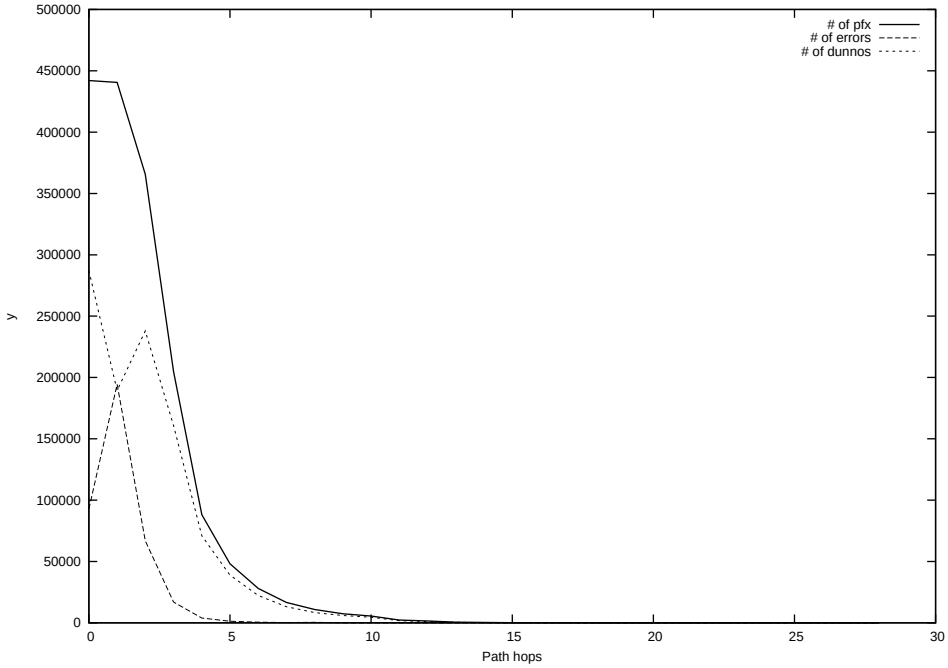
2013-02-22



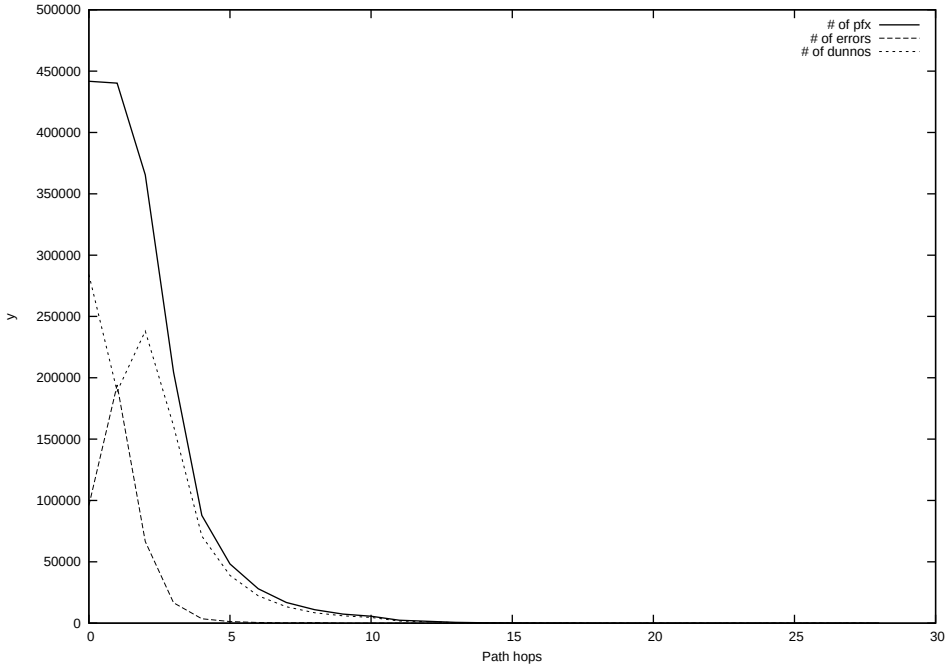
2013-02-23



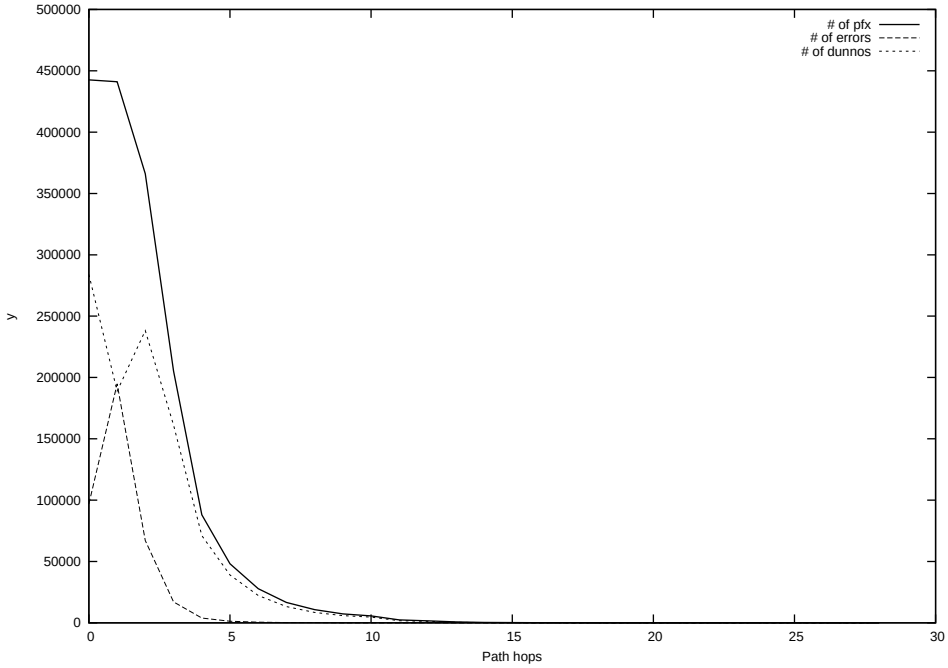
2013-02-24



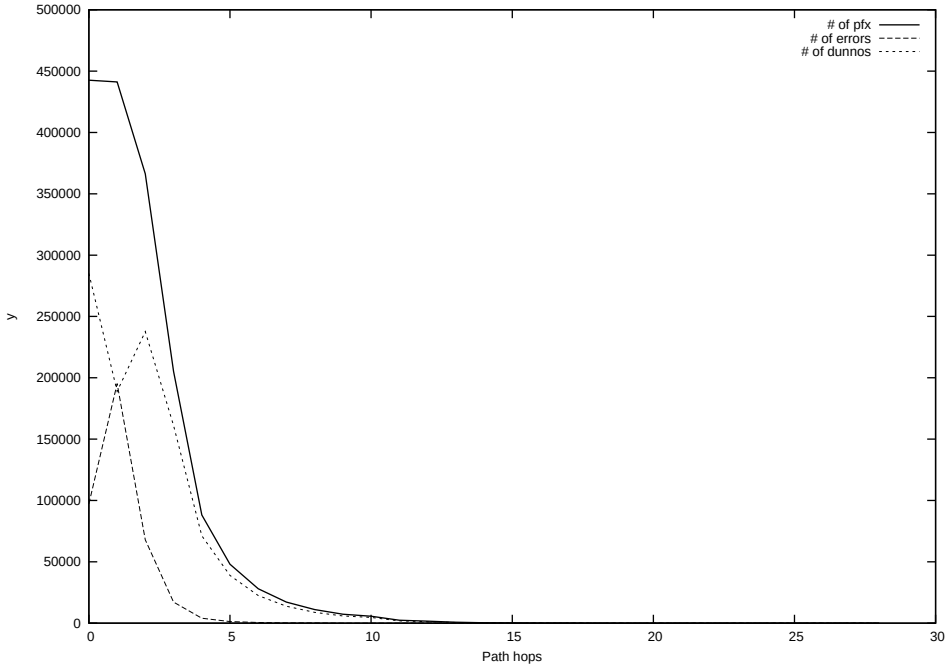
2013-02-25



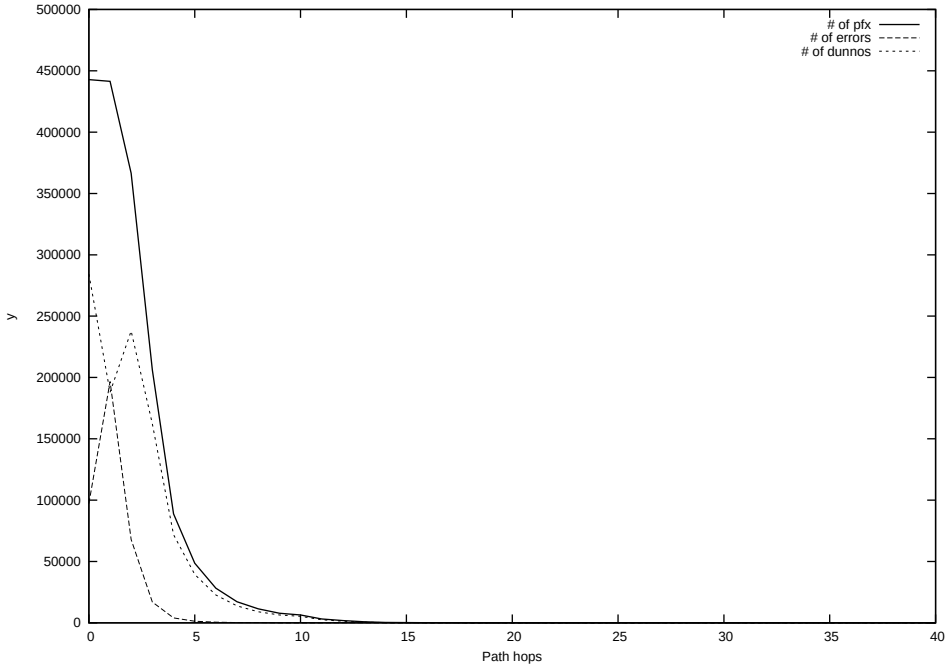
2013-02-26



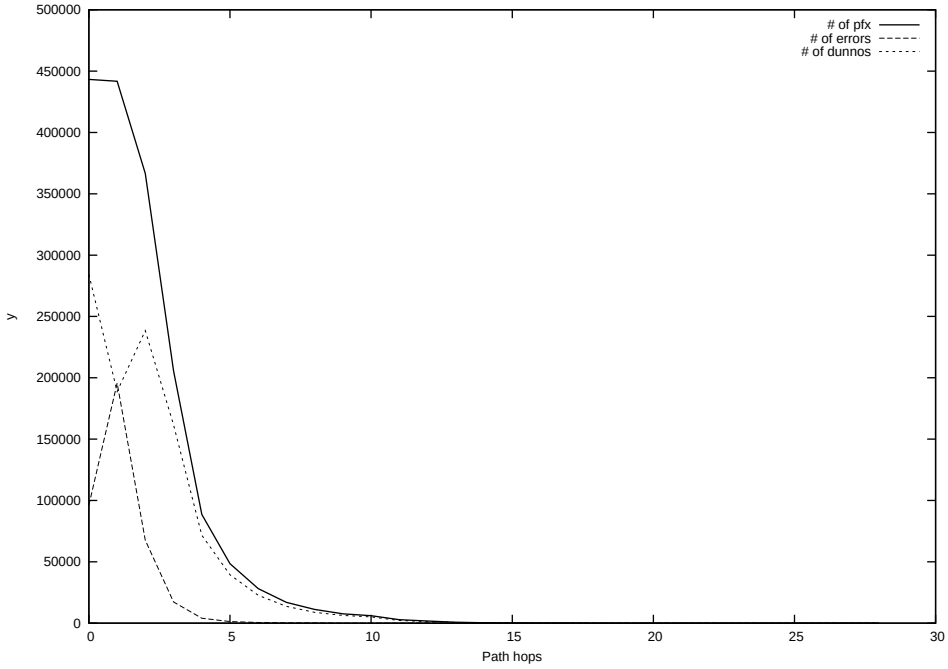
2013-02-27



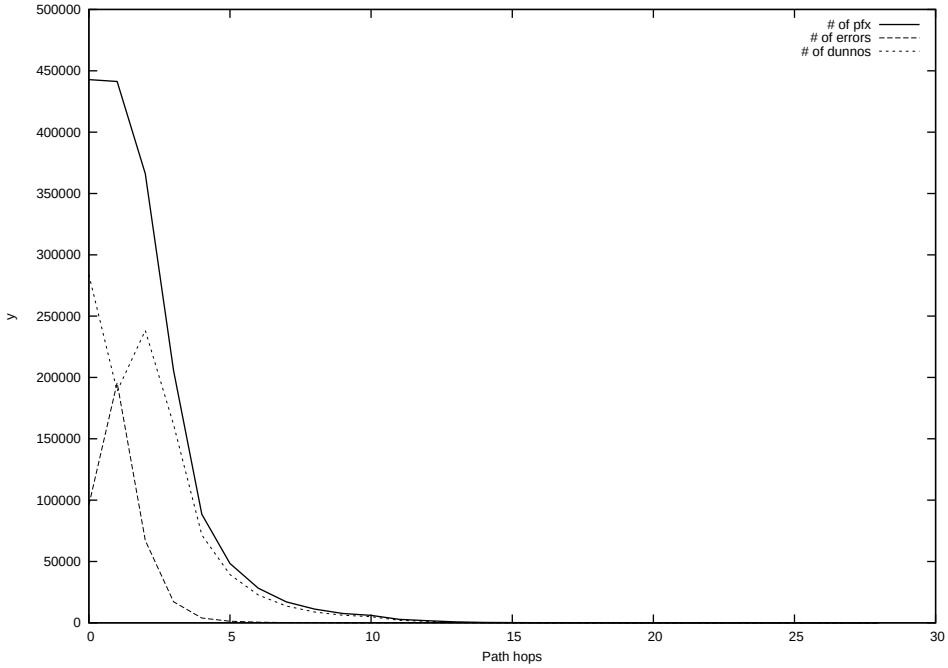
2013-02-28



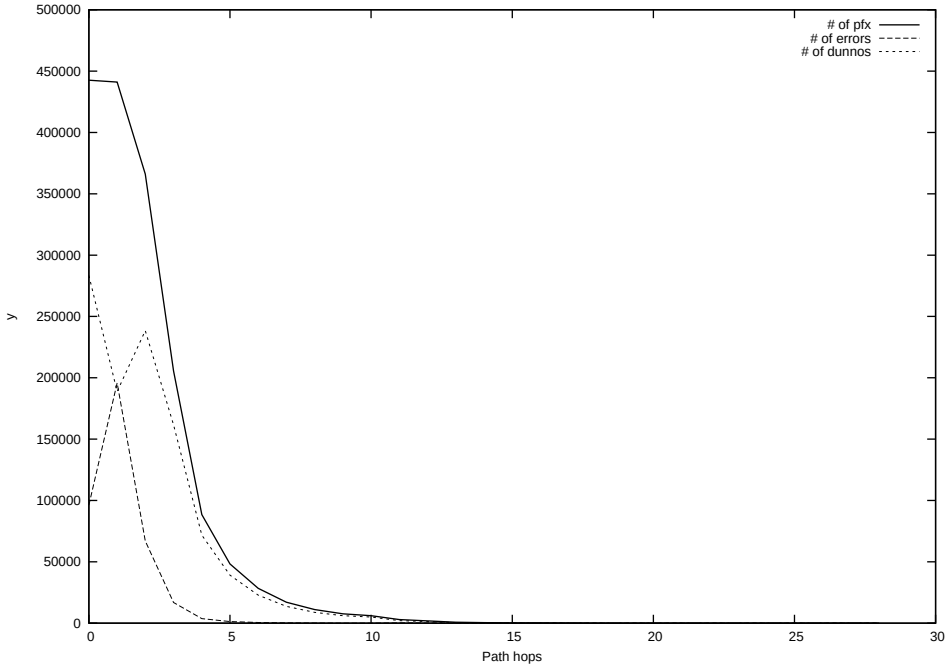
2013-03-01



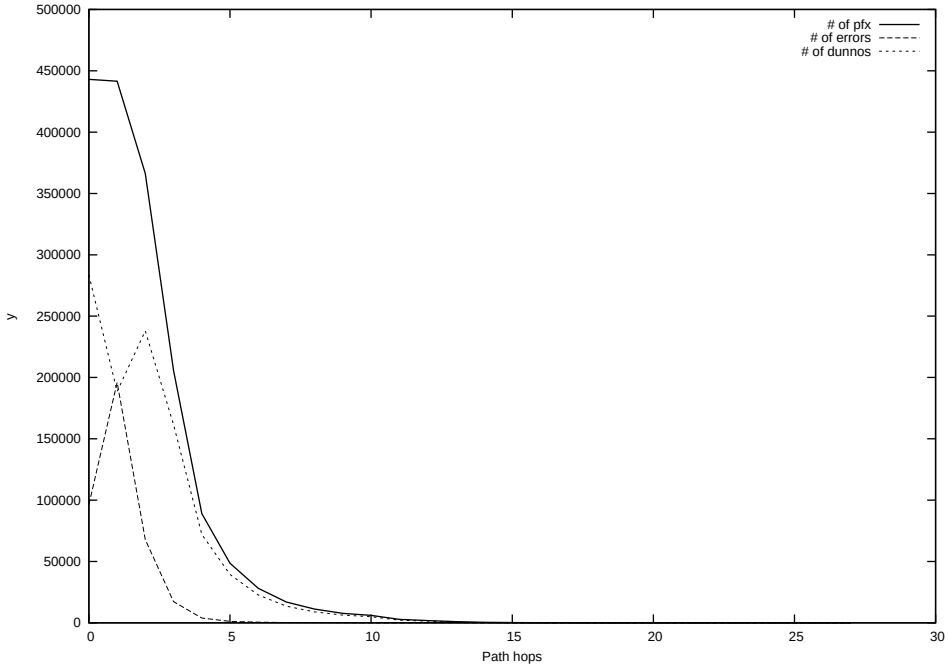
2013-03-02



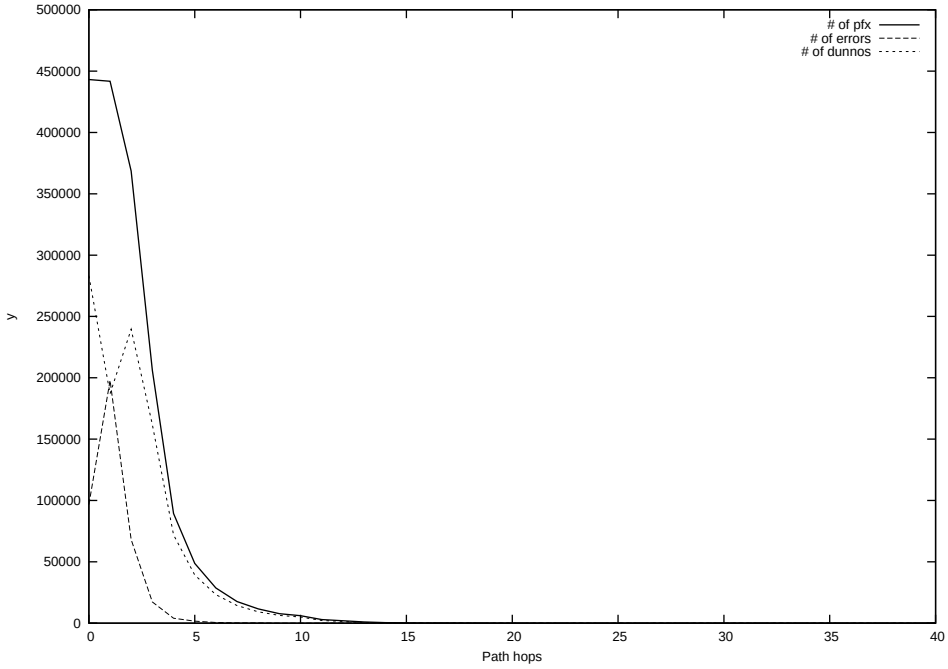
2013-03-03



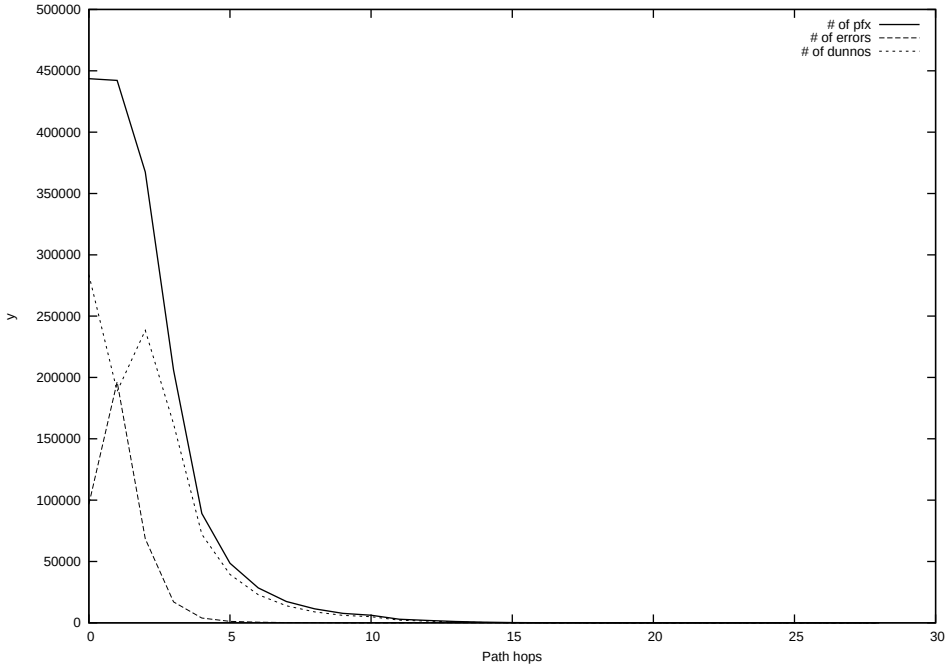
2013-03-04



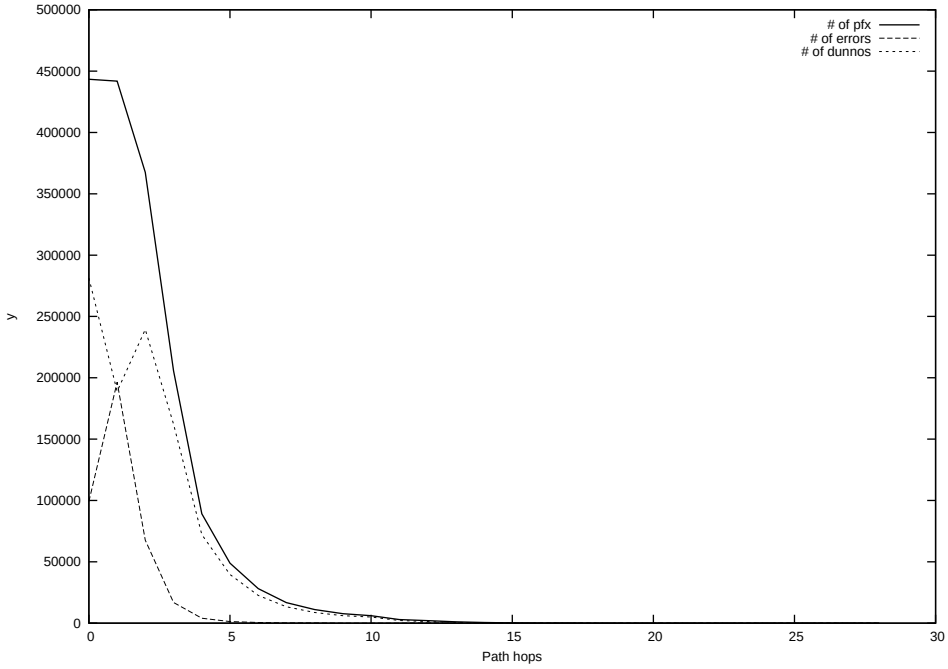
2013-03-05



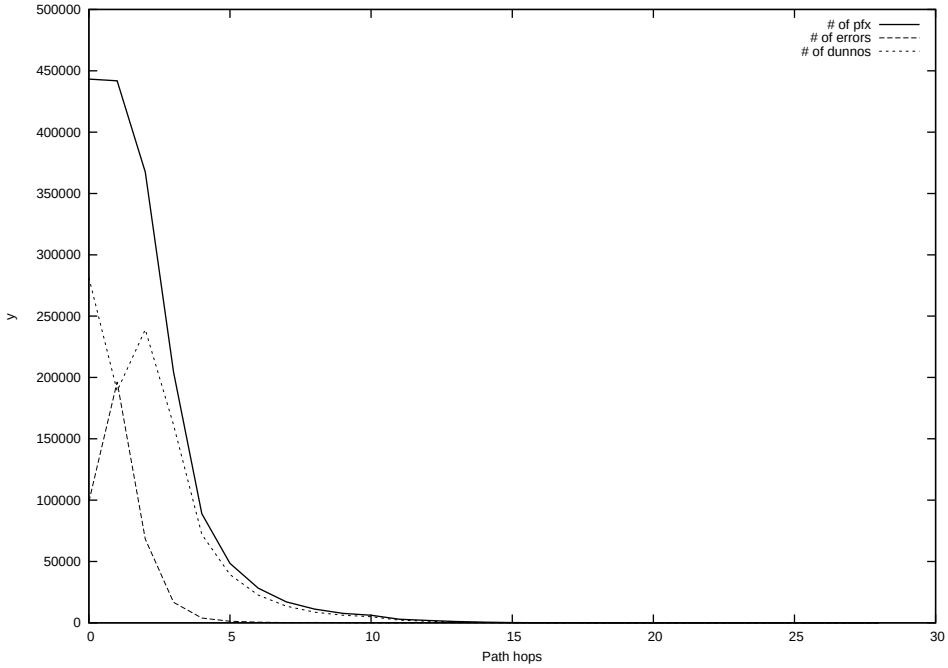
2013-03-06



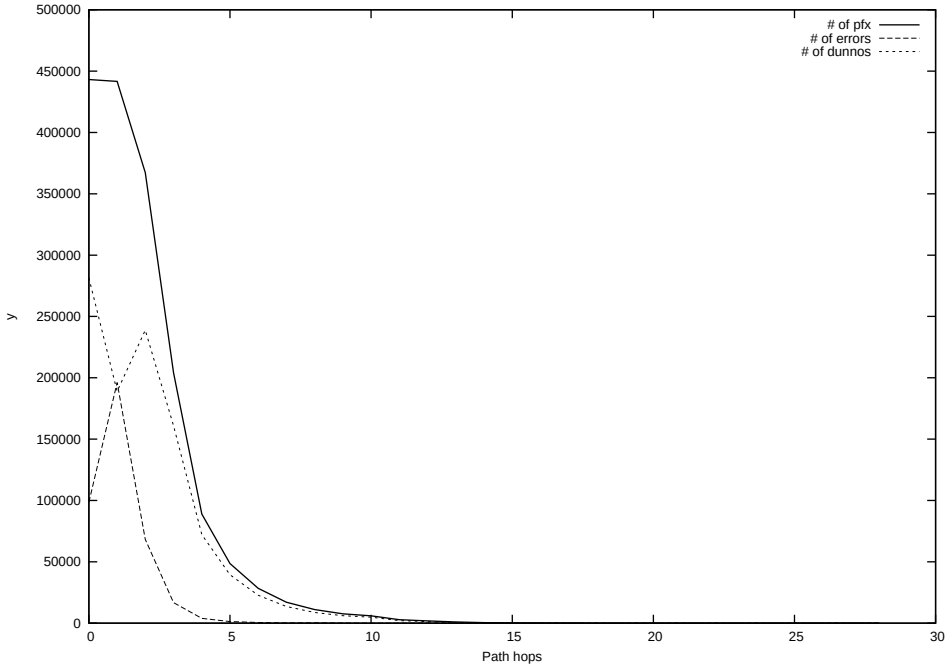
2013-03-07



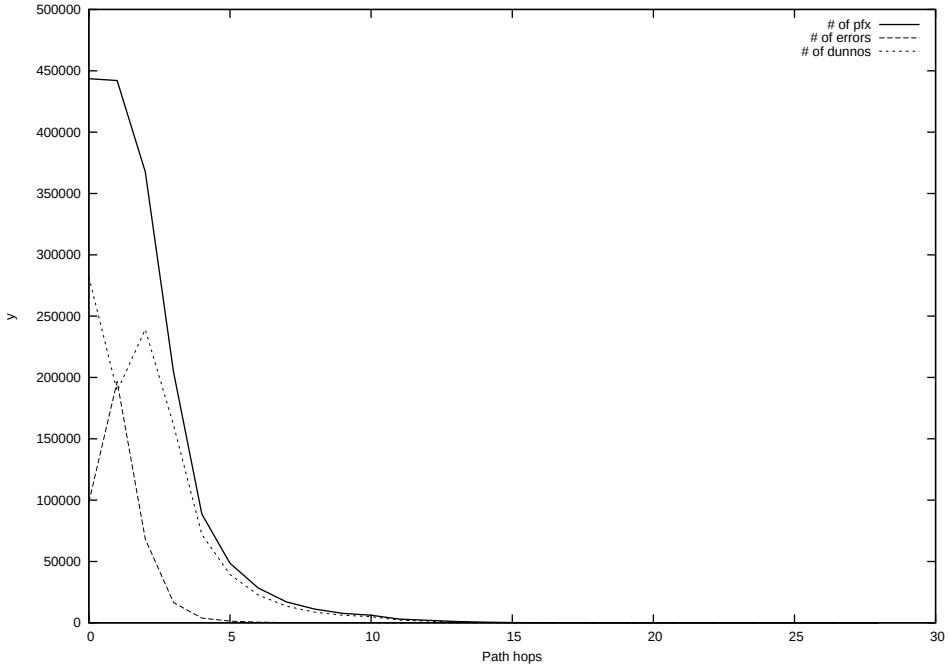
2013-03-08



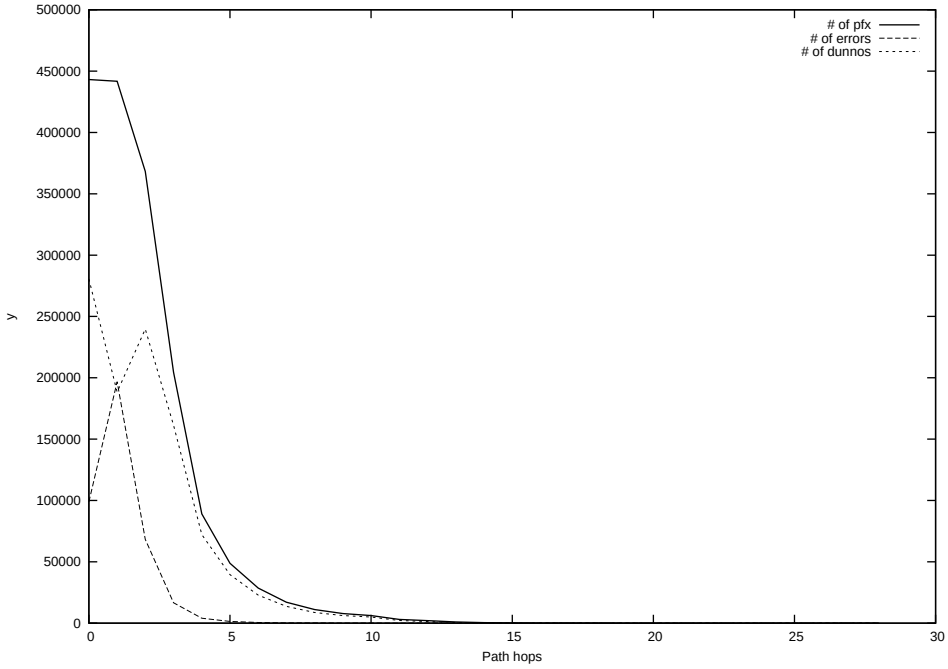
2013-03-09



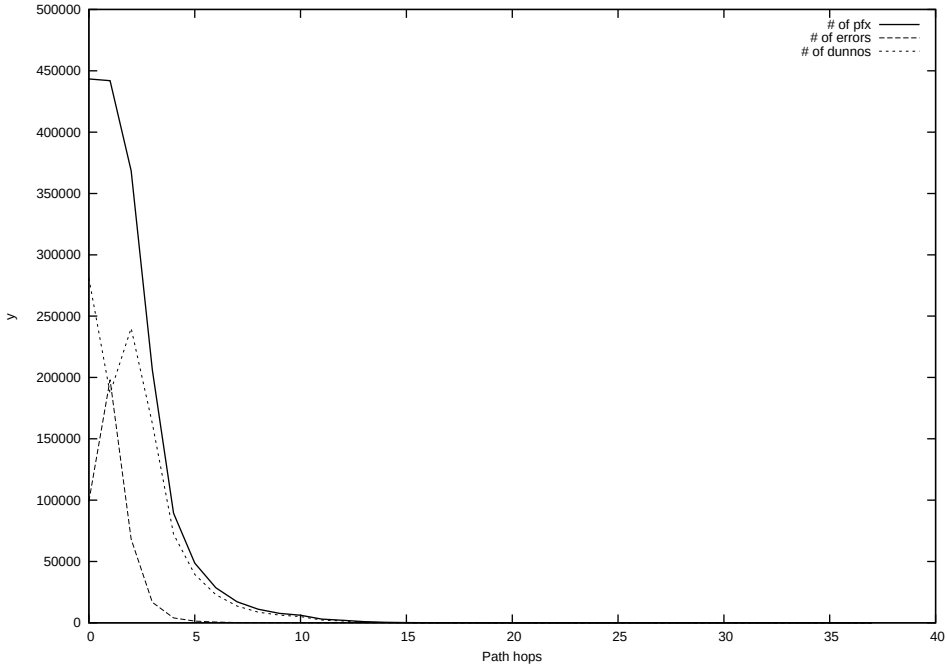
2013-03-10



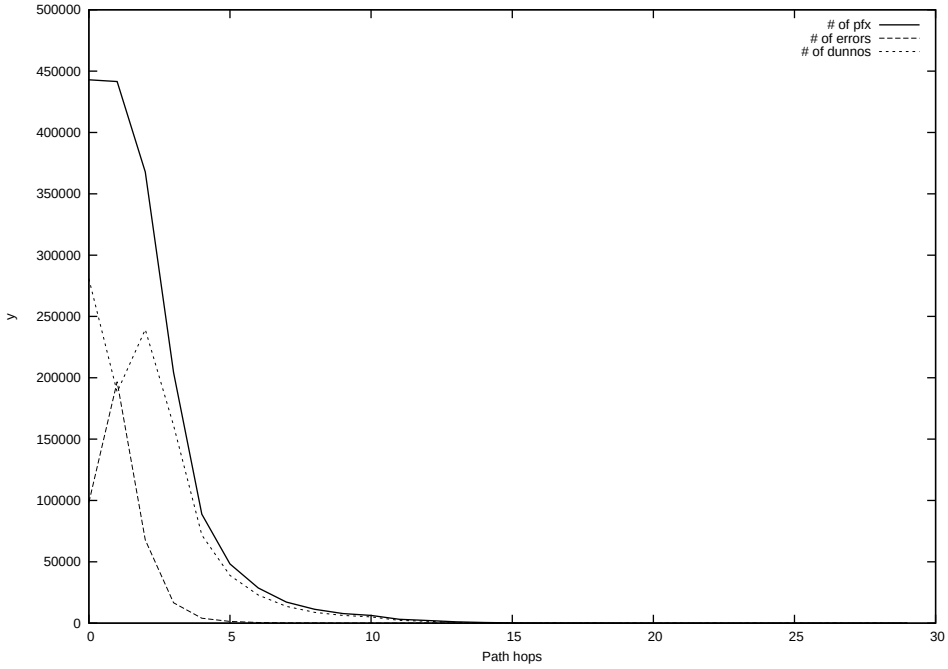
2013-03-11



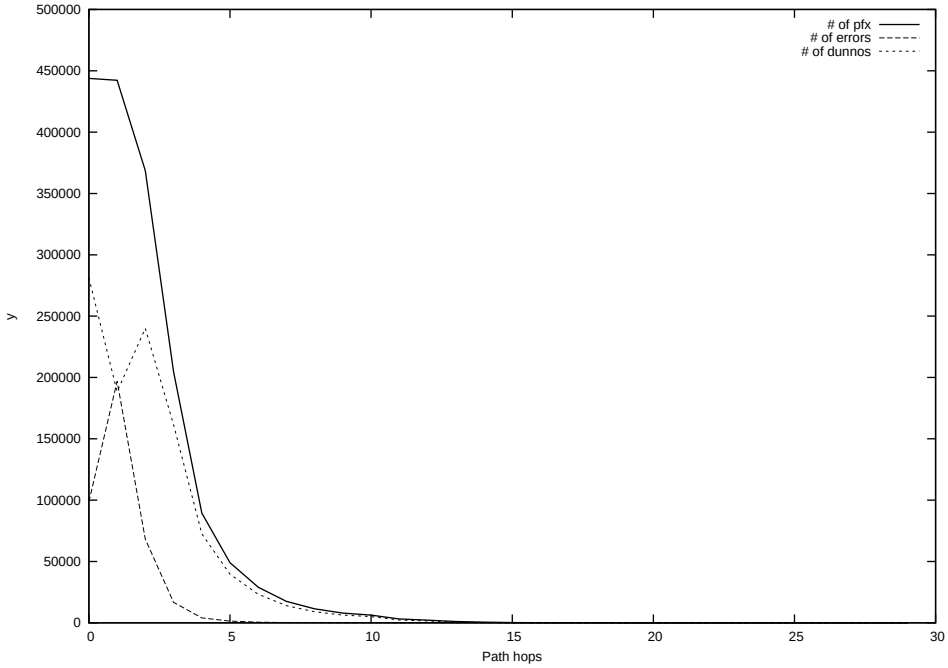
2013-03-12



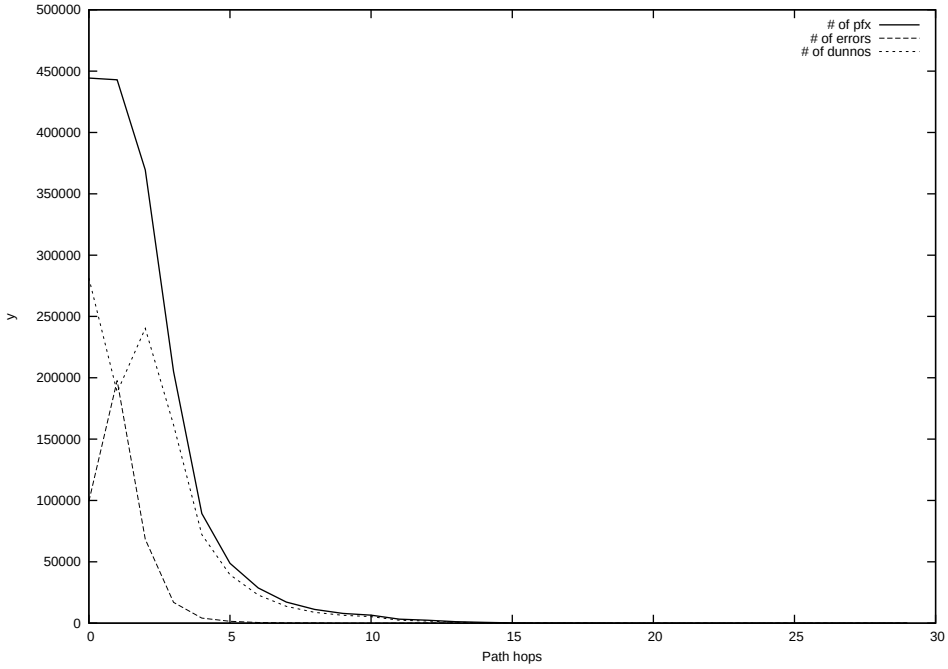
2013-03-13



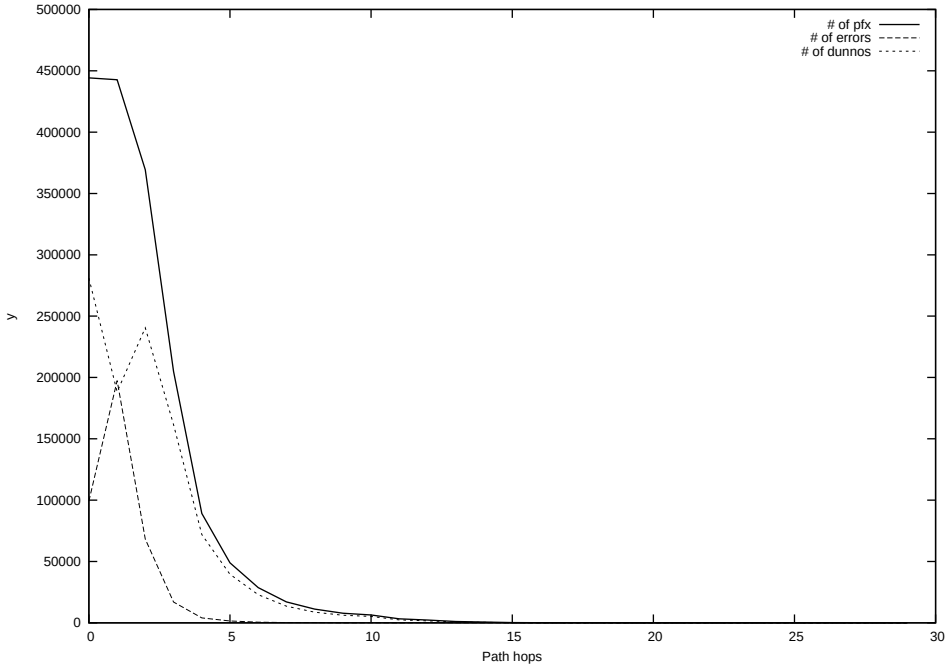
2013-03-14



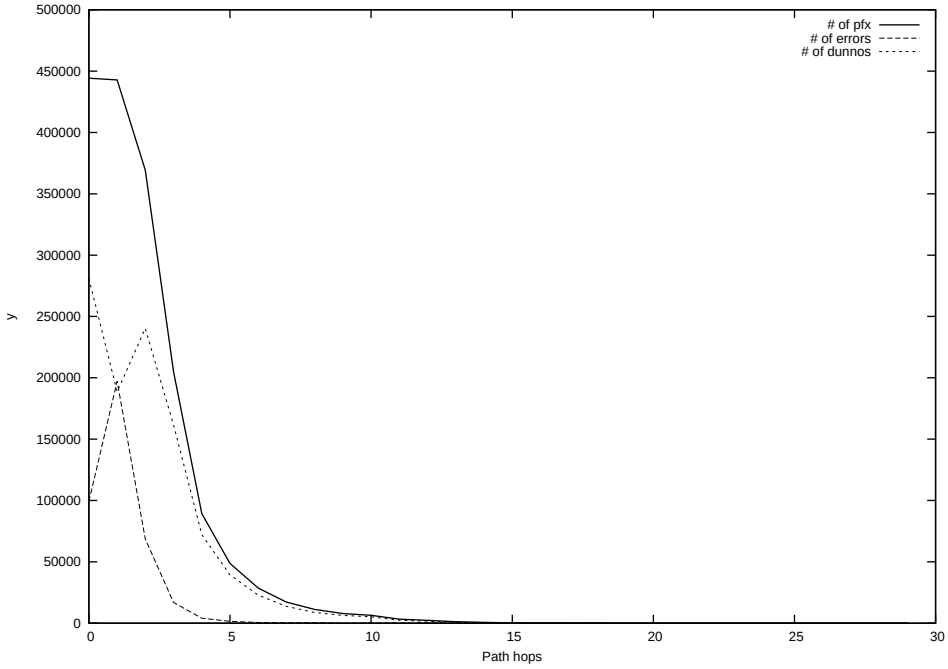
2013-03-15



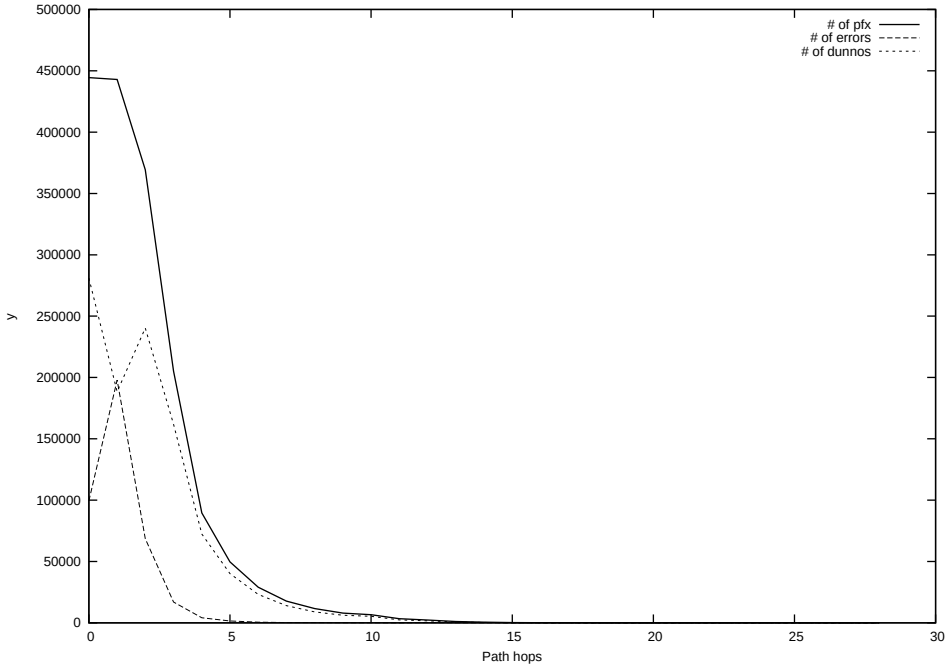
2013-03-16



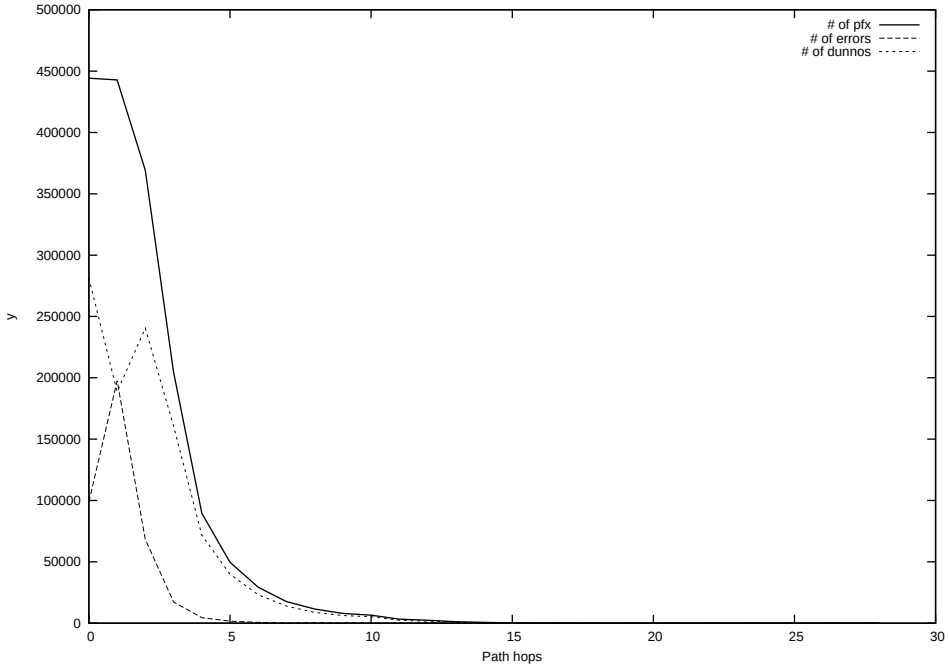
2013-03-17



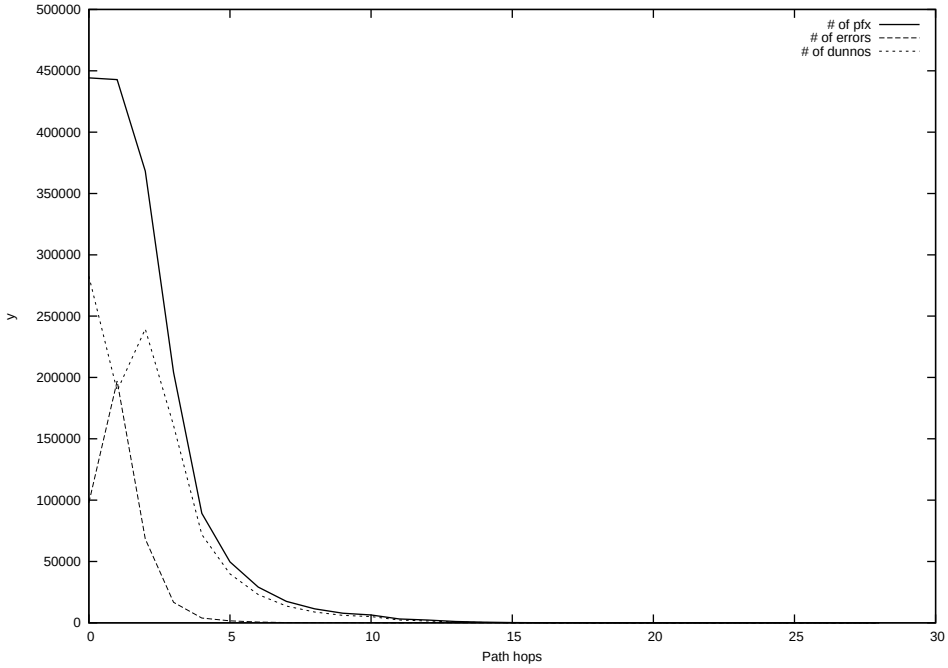
2013-03-18



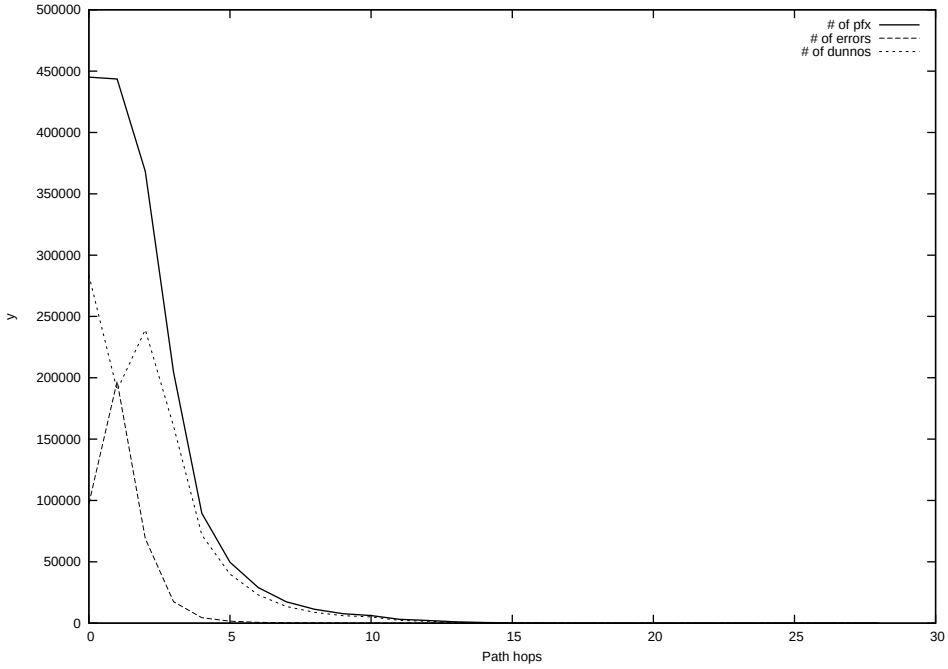
2013-03-19



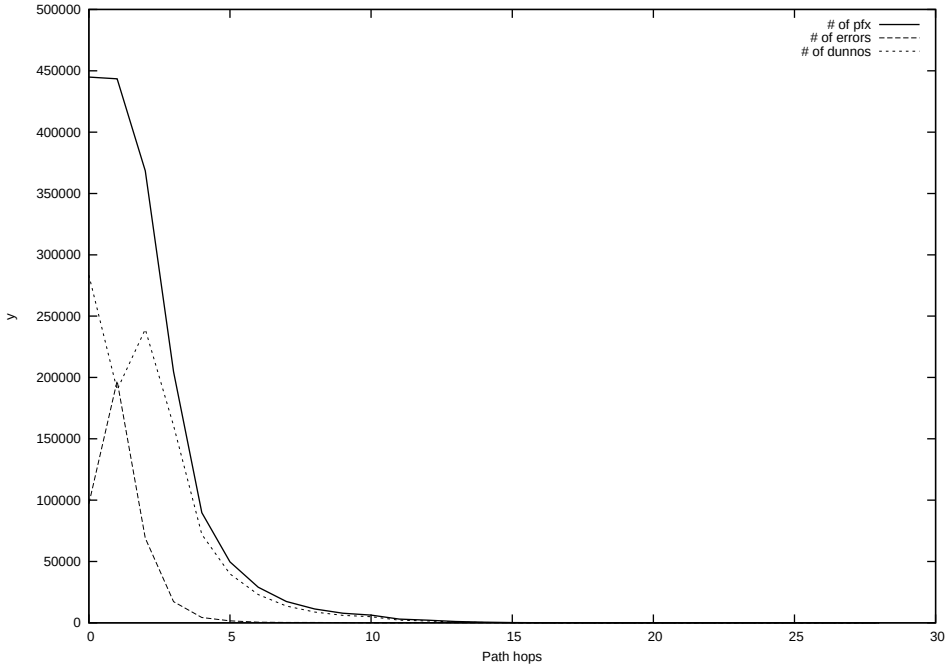
2013-03-20



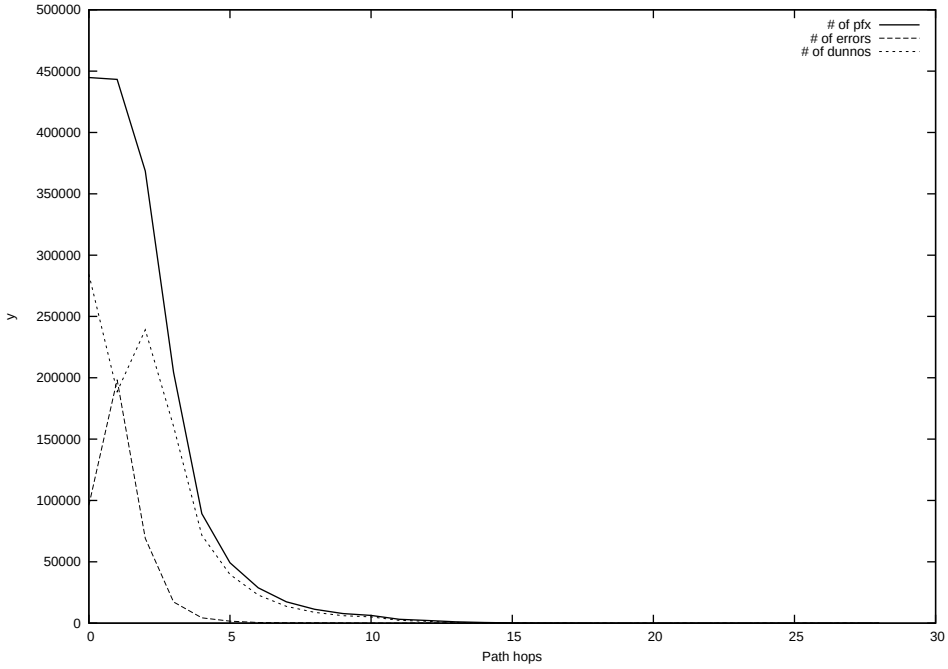
2013-03-21



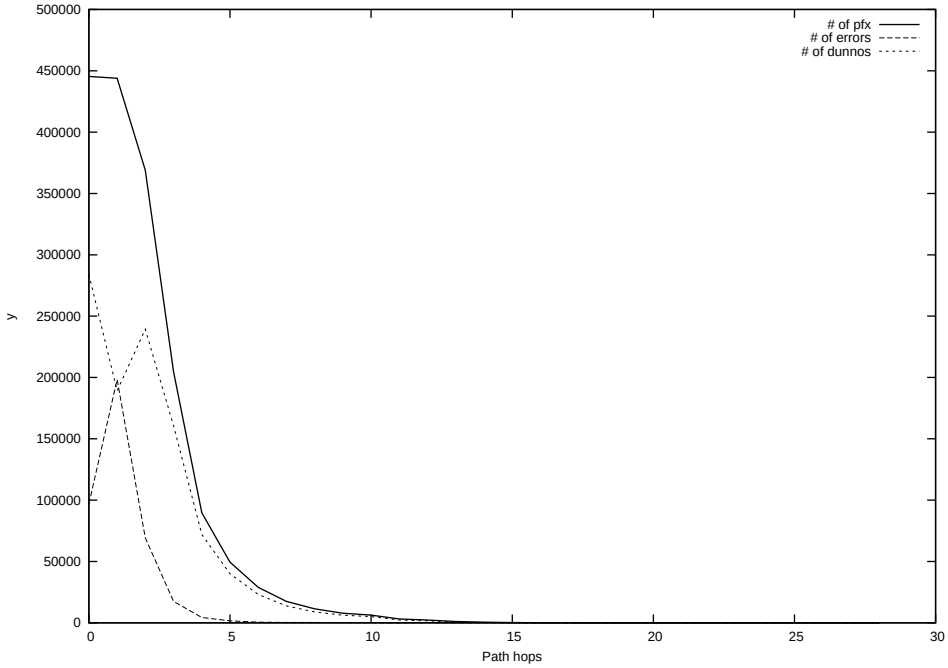
2013-03-22



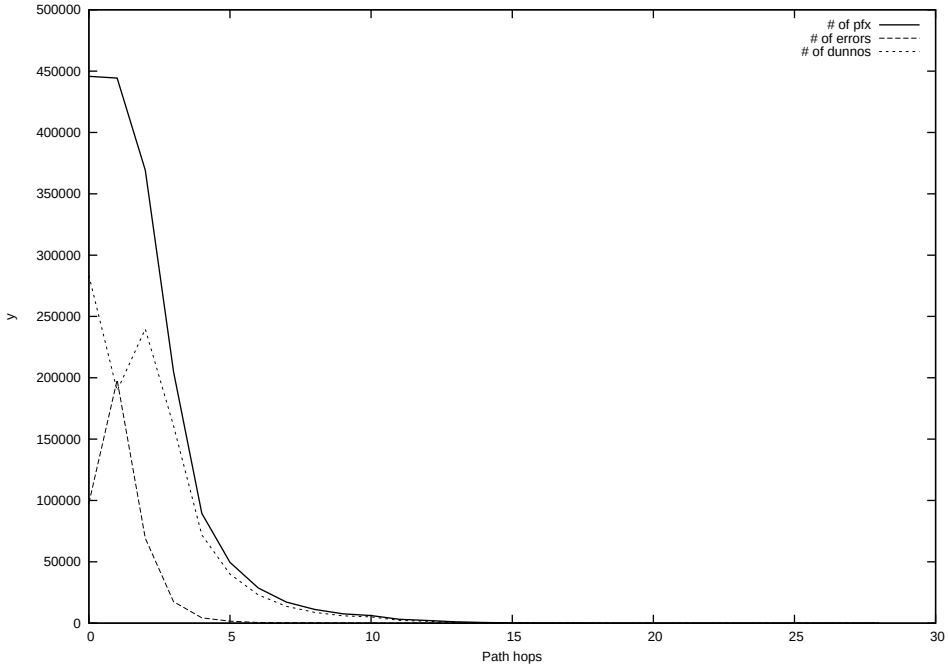
2013-03-23



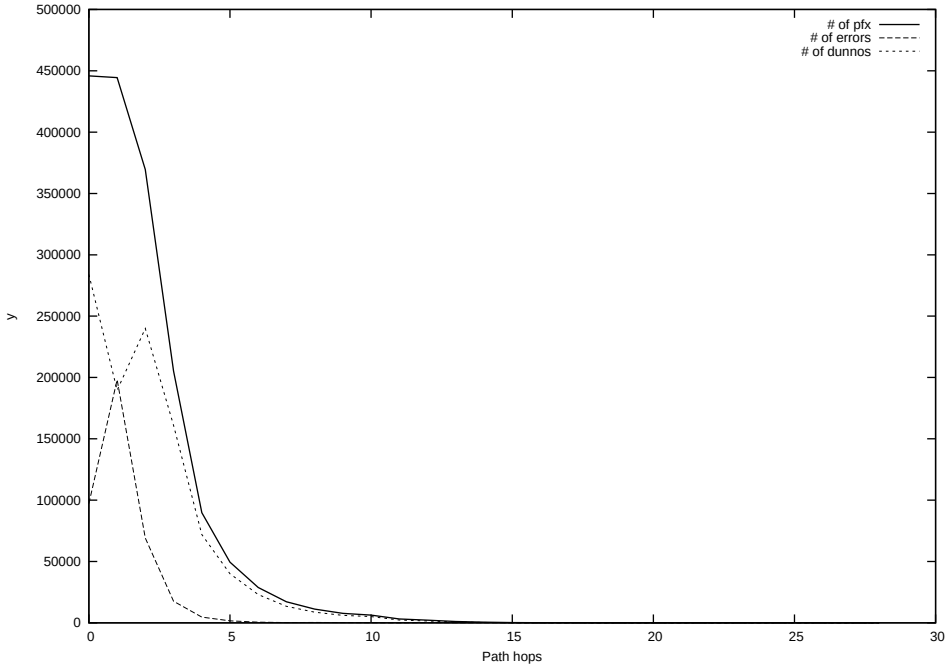
2013-03-24



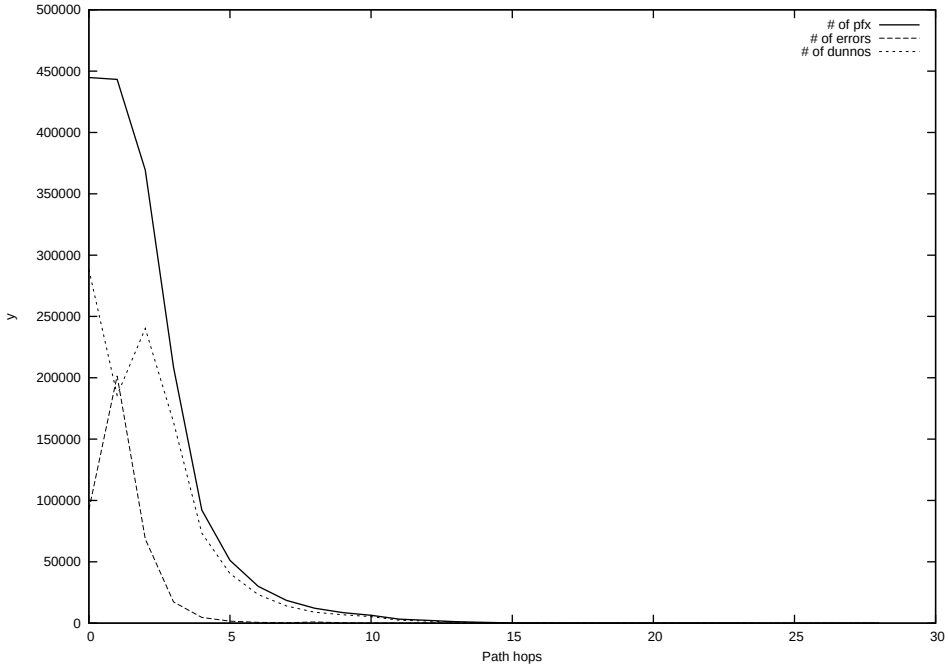
2013-03-25



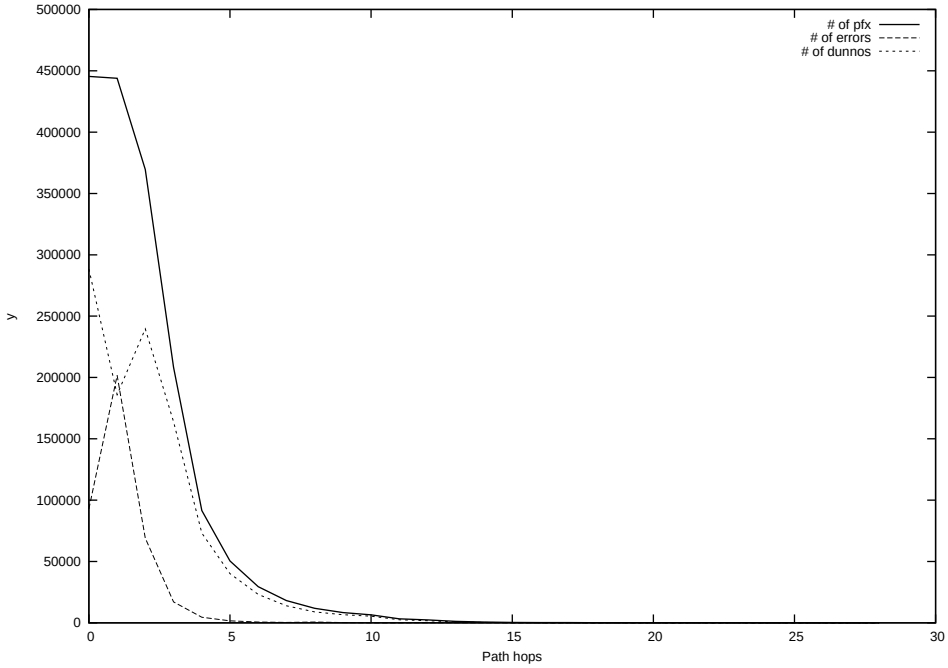
2013-03-26



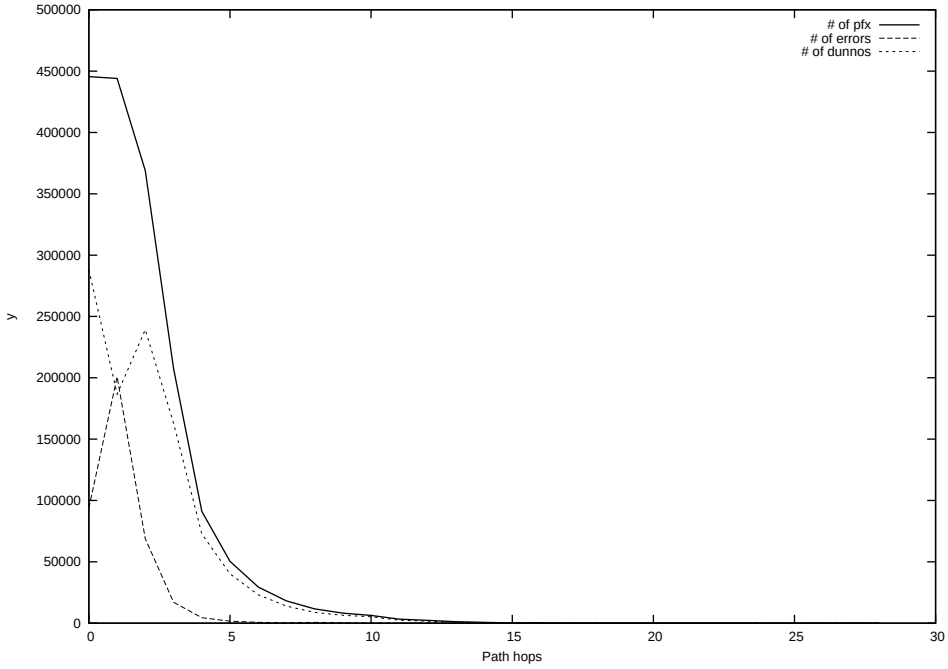
2013-03-27



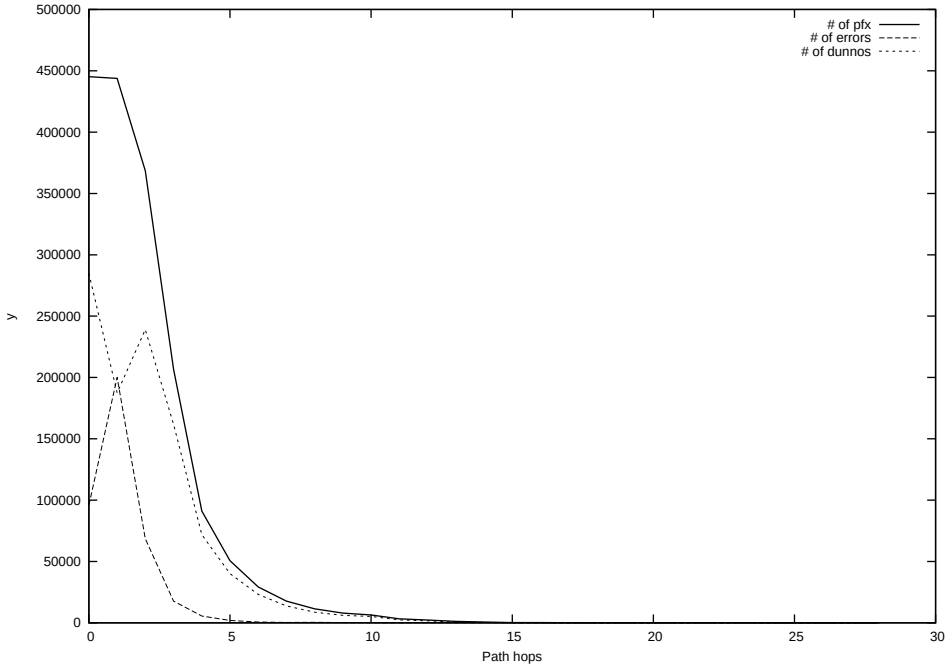
2013-03-28



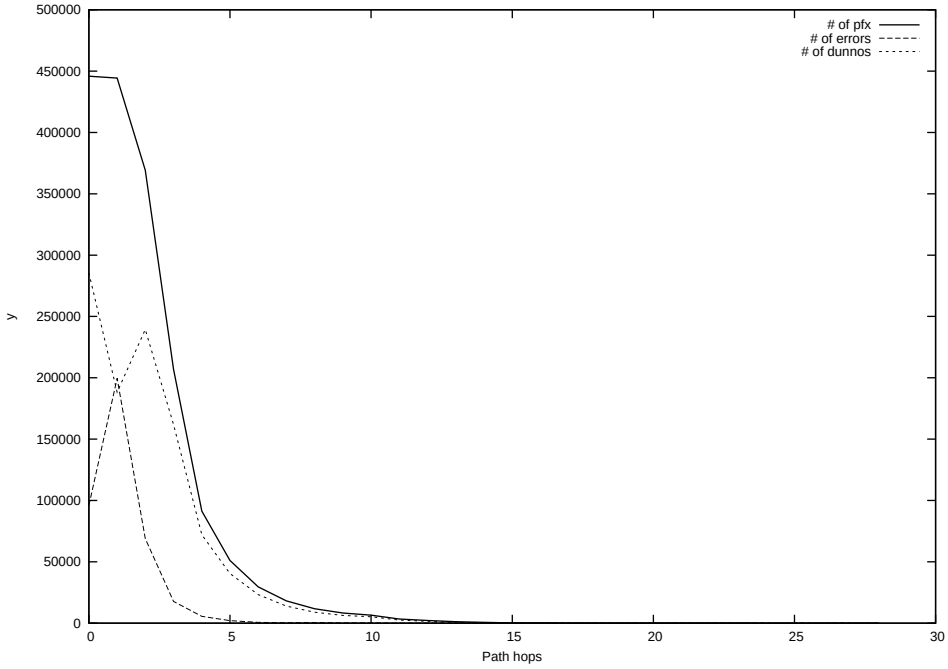
2013-03-29



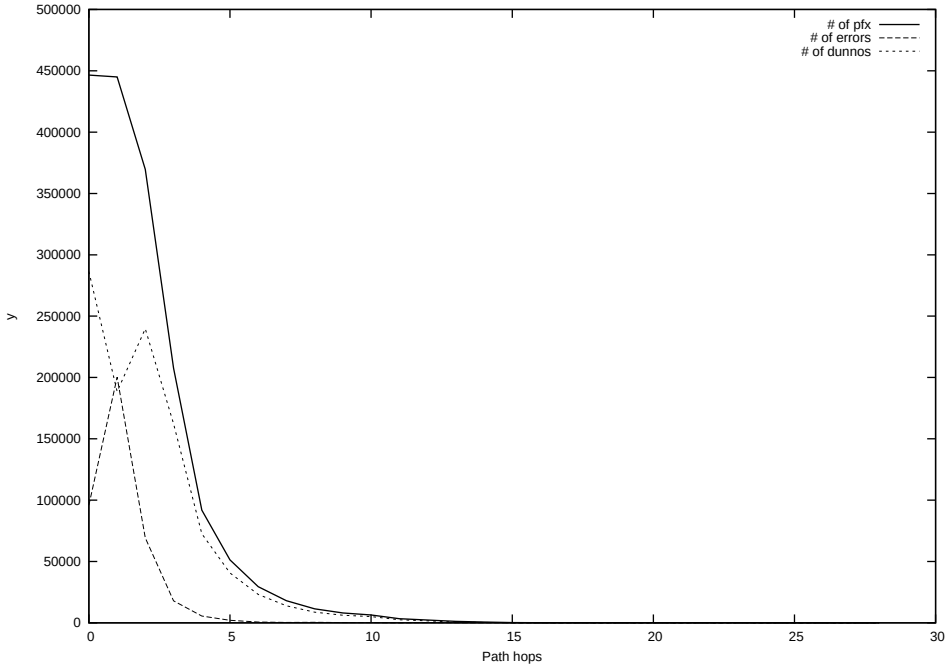
2013-03-30



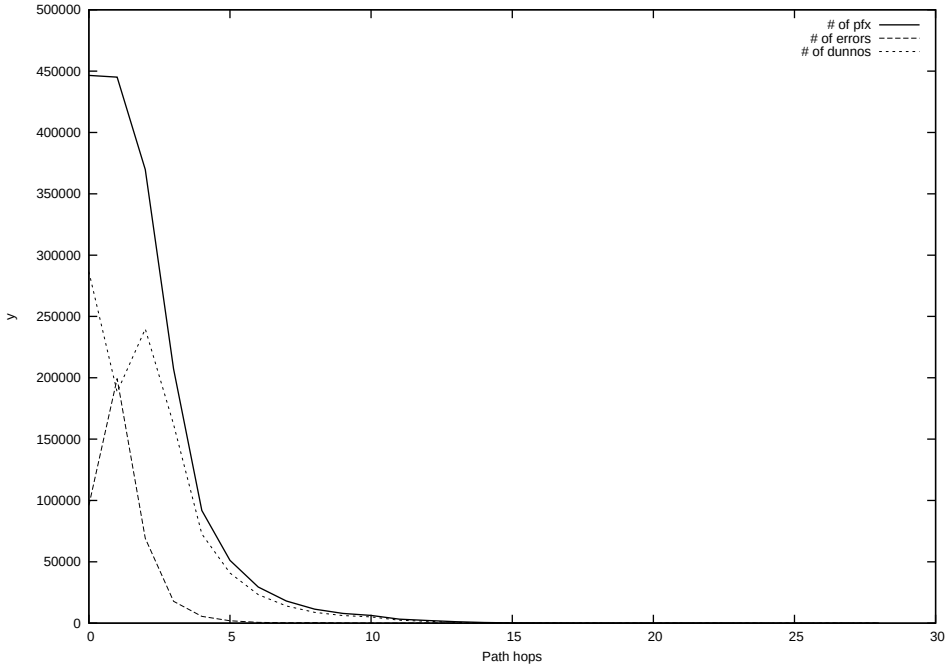
2013-03-31



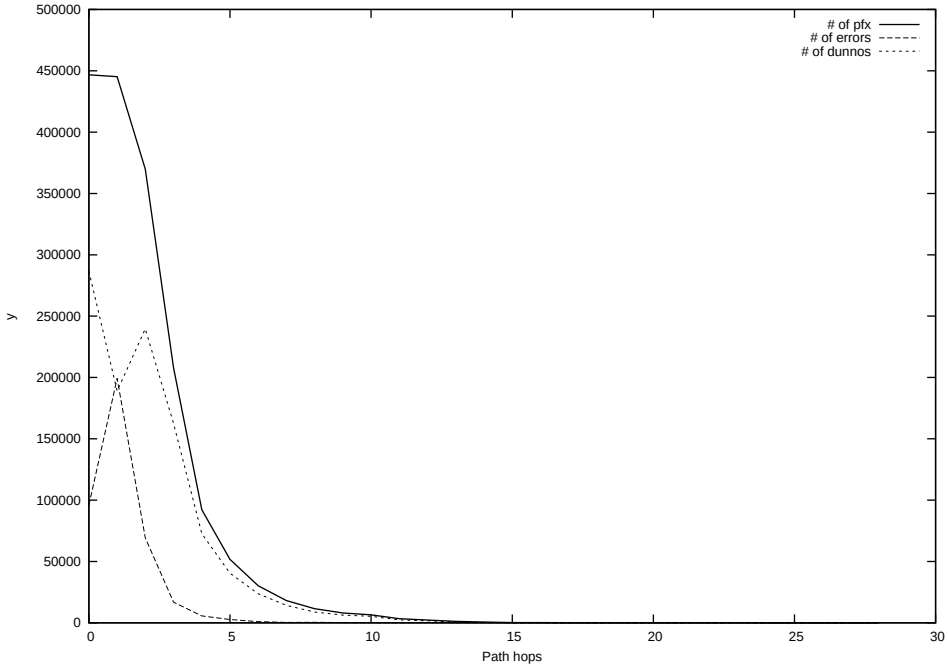
2013-04-01



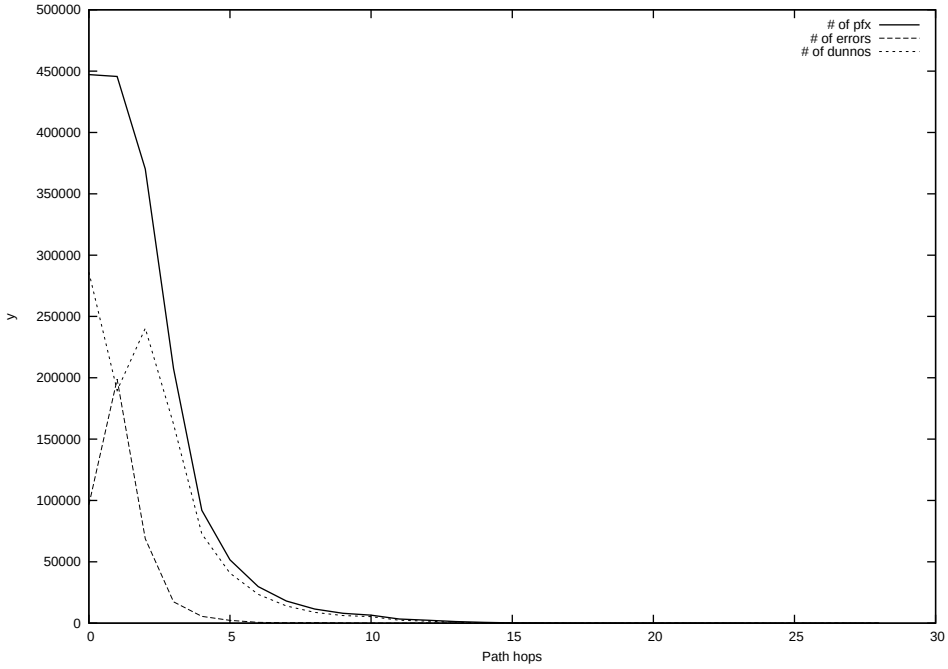
2013-04-02



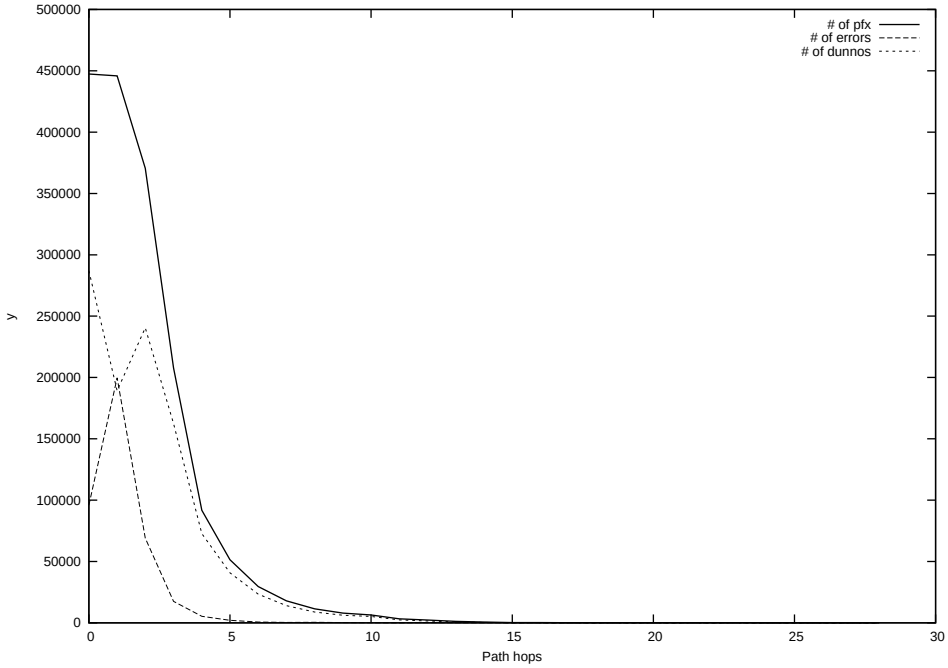
2013-04-03



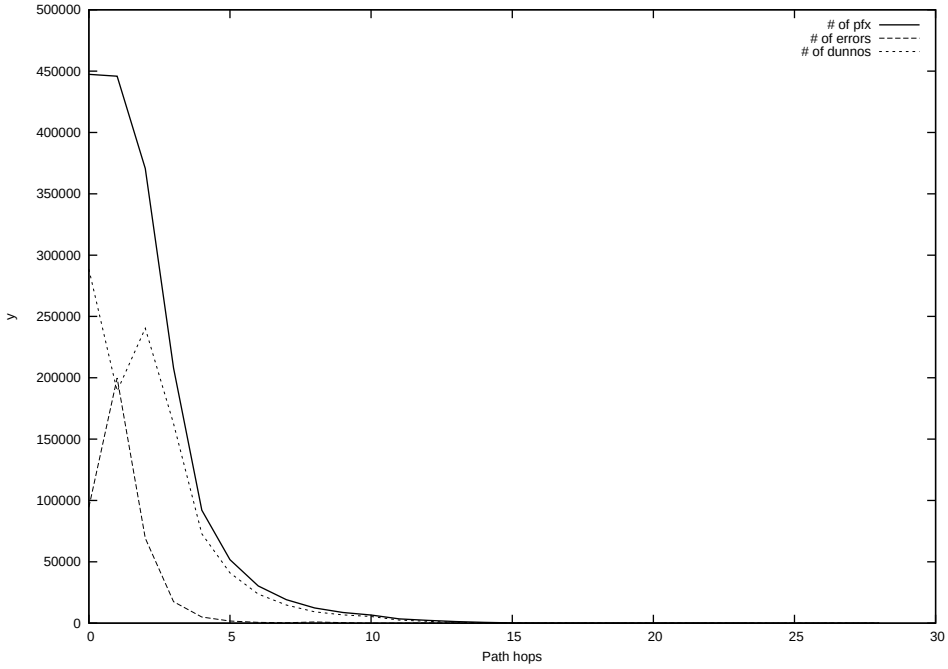
2013-04-04



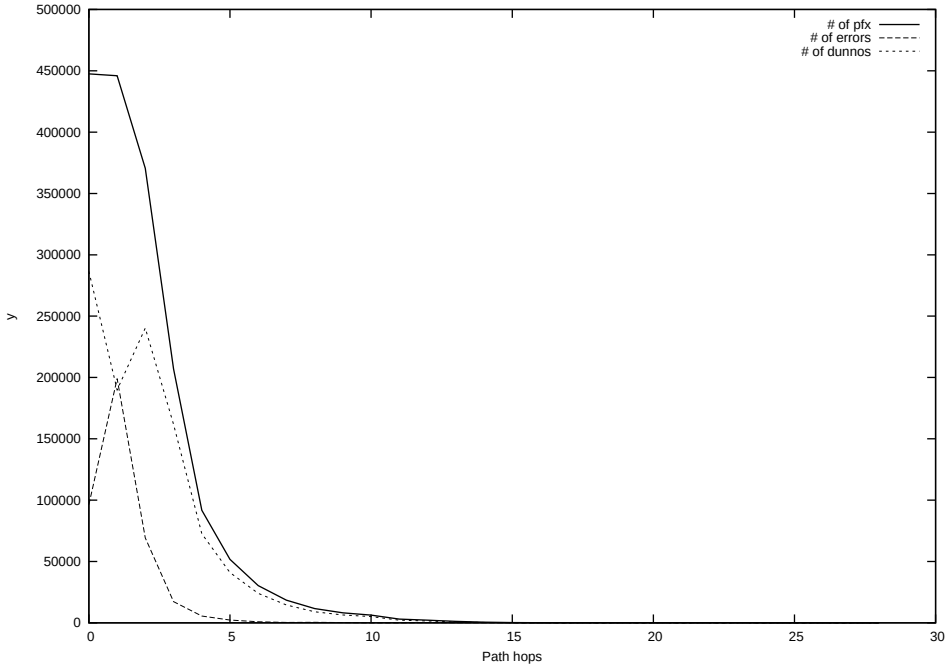
2013-04-05



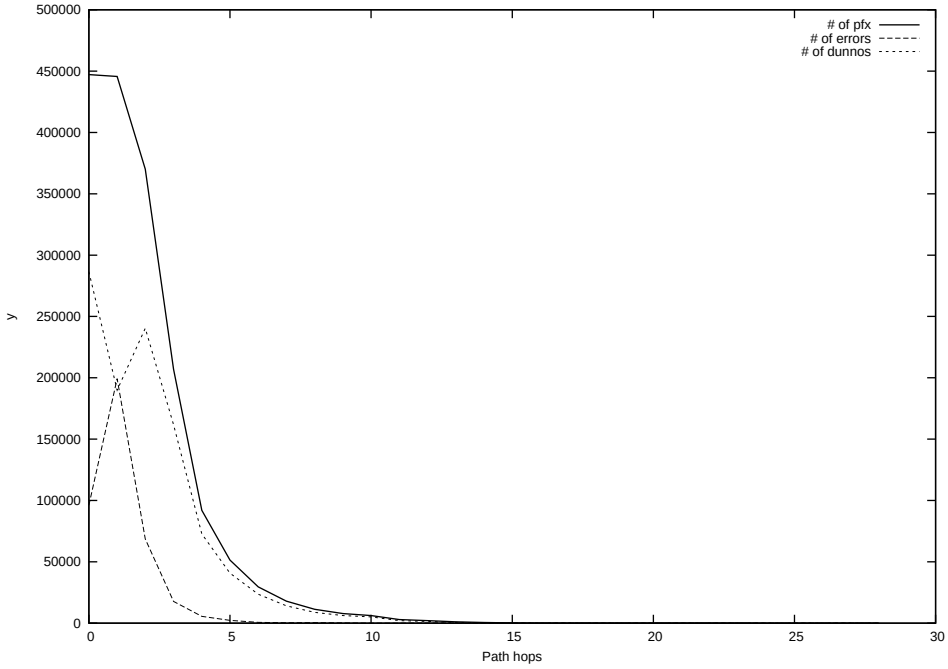
2013-04-06



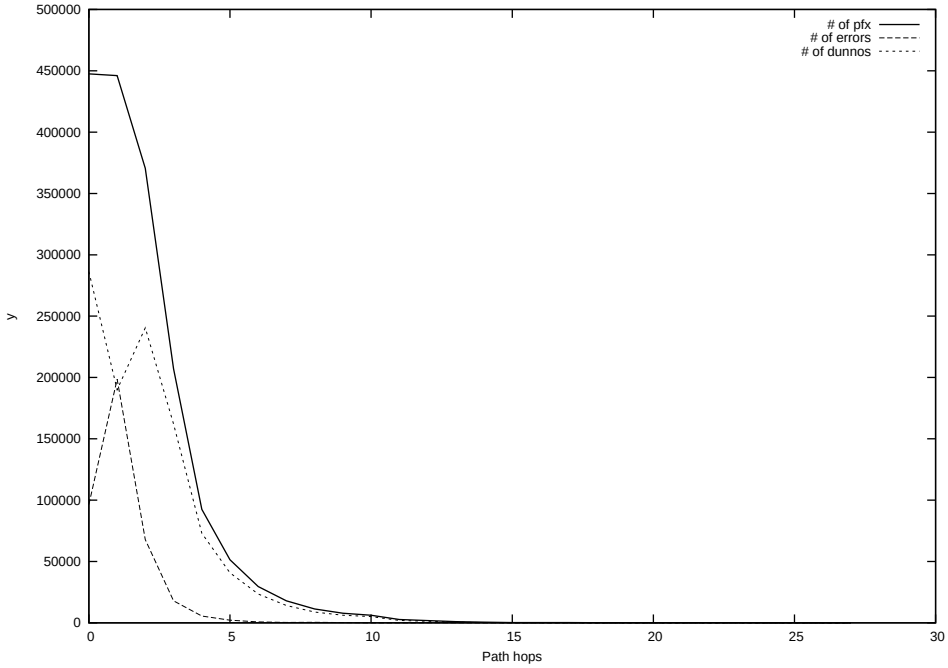
2013-04-07



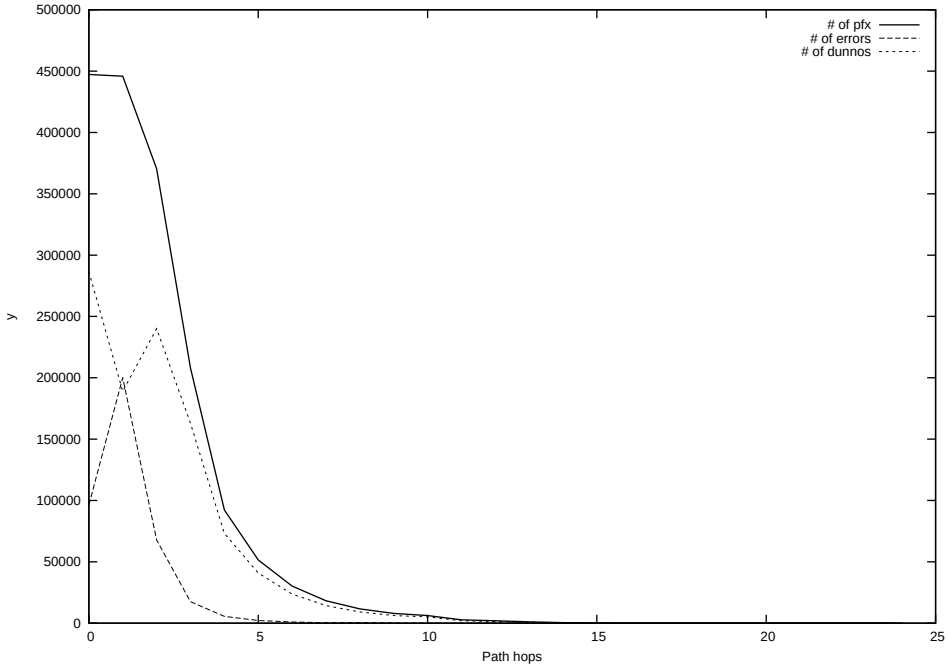
2013-04-08



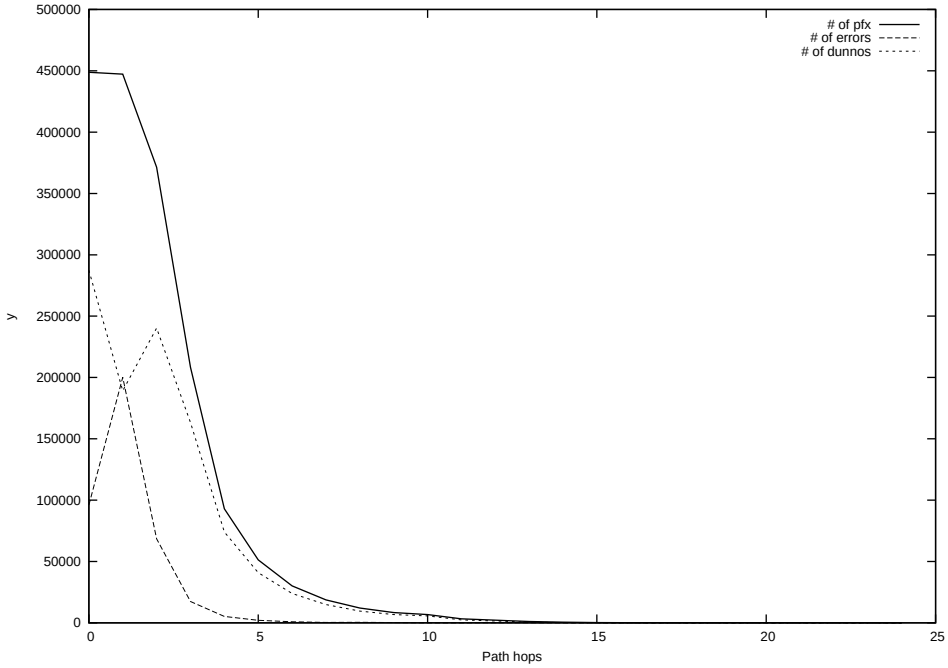
2013-04-09



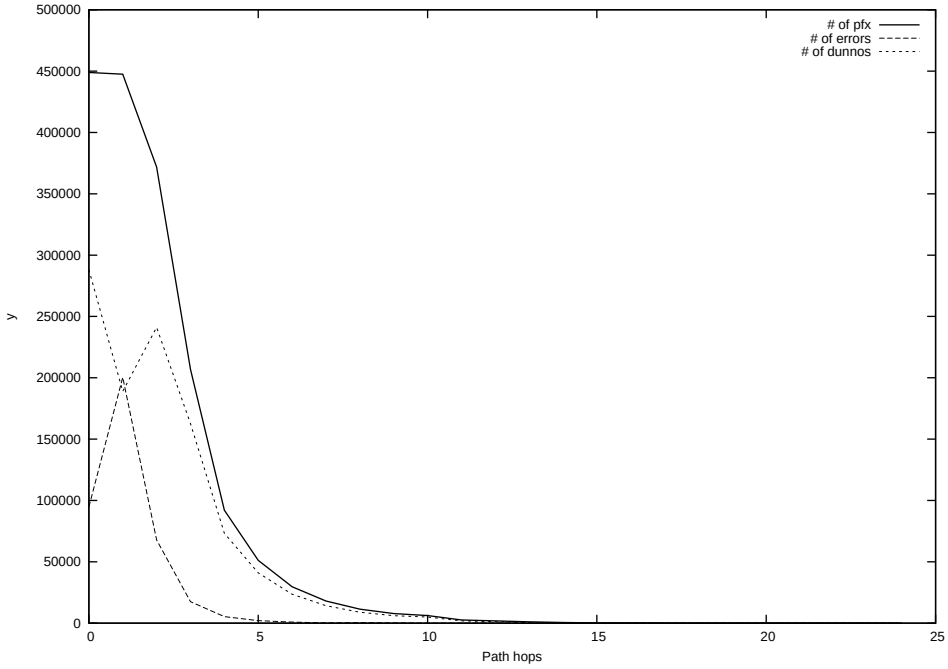
2013-04-10



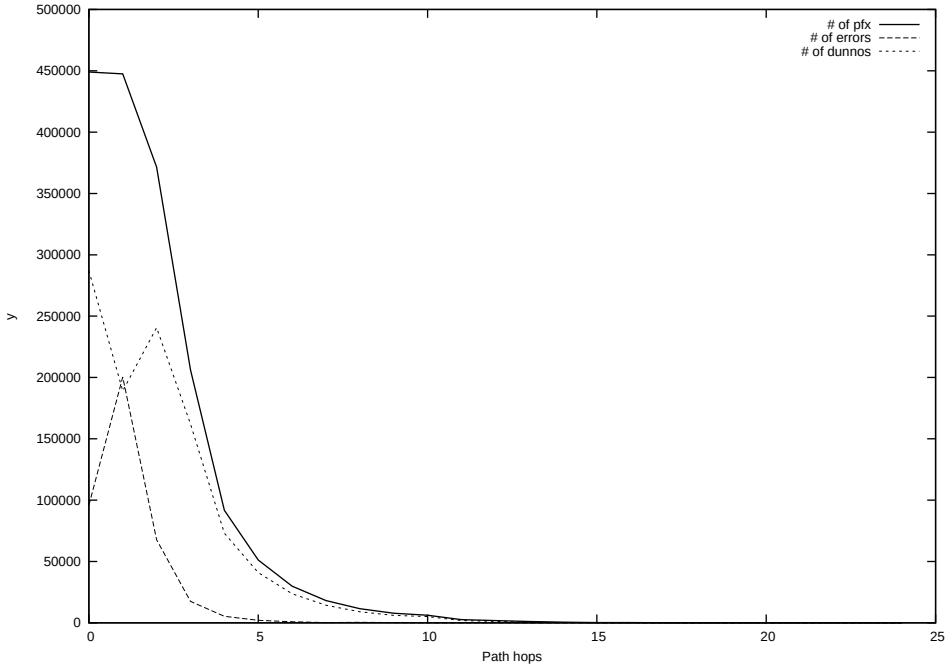
2013-04-11



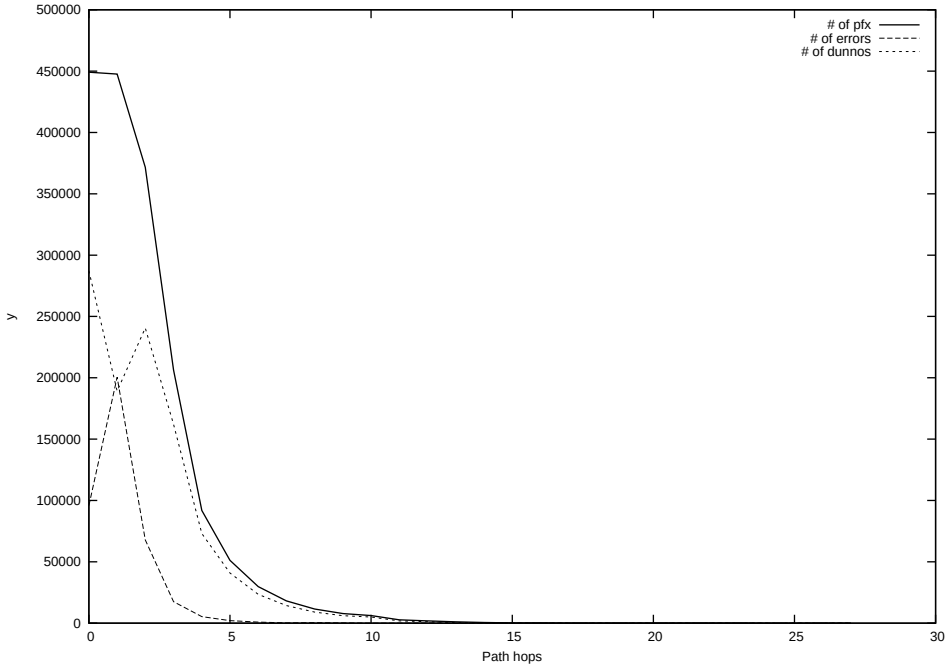
2013-04-12



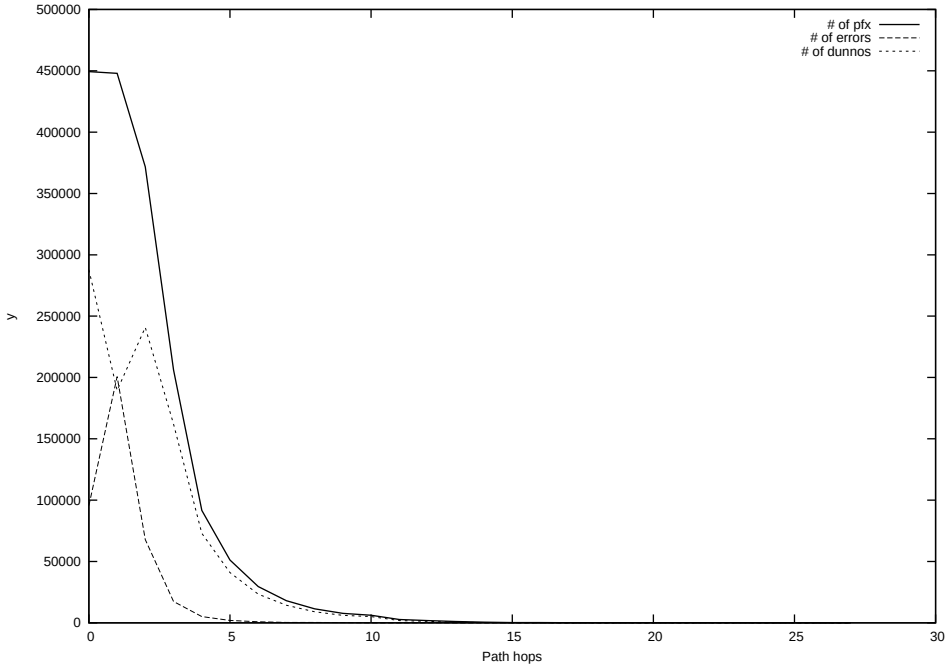
2013-04-13



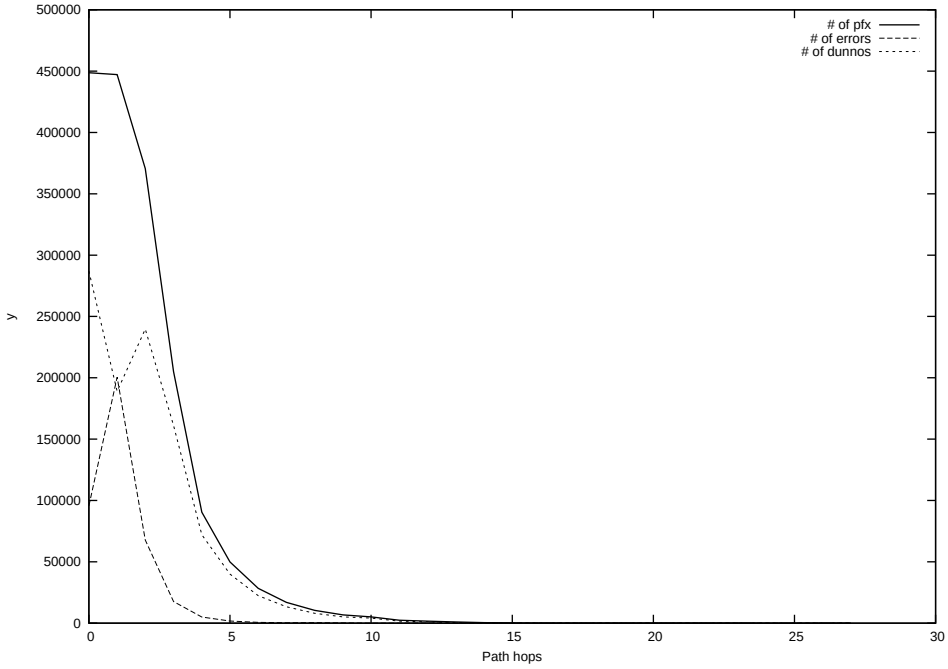
2013-04-14



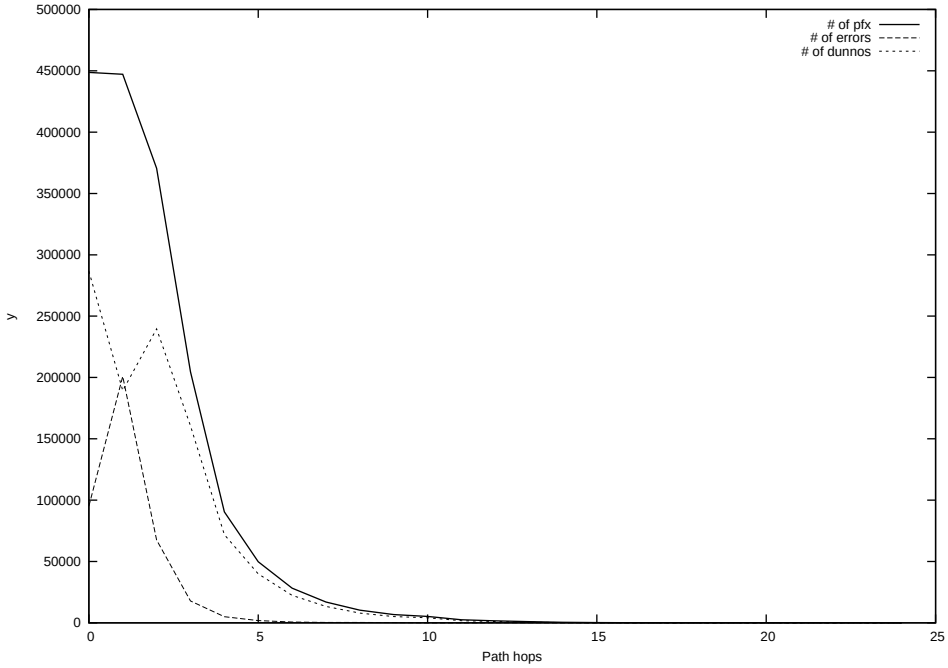
2013-04-15



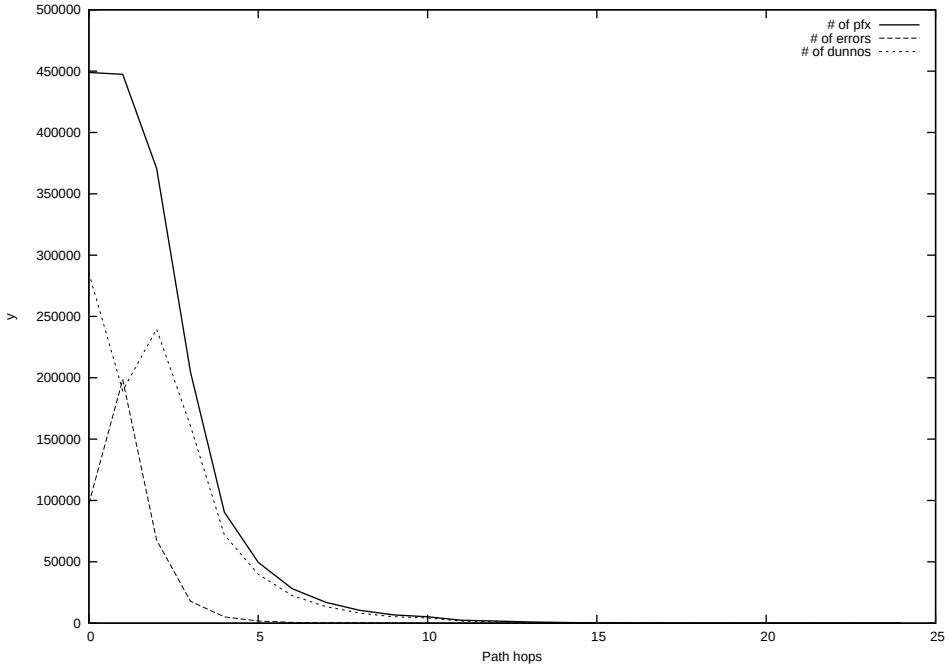
2013-04-16



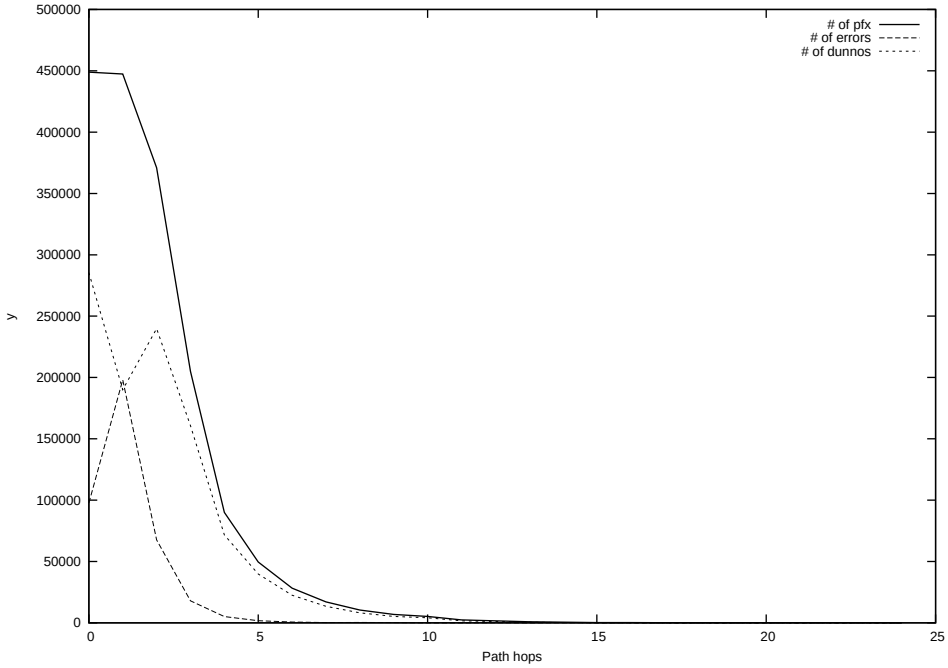
2013-04-17



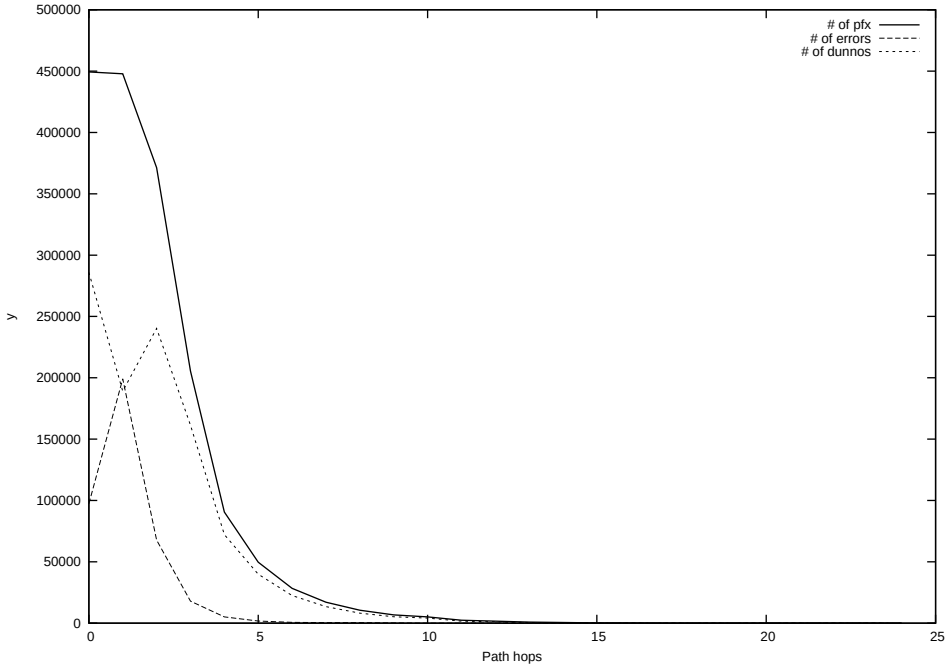
2013-04-18



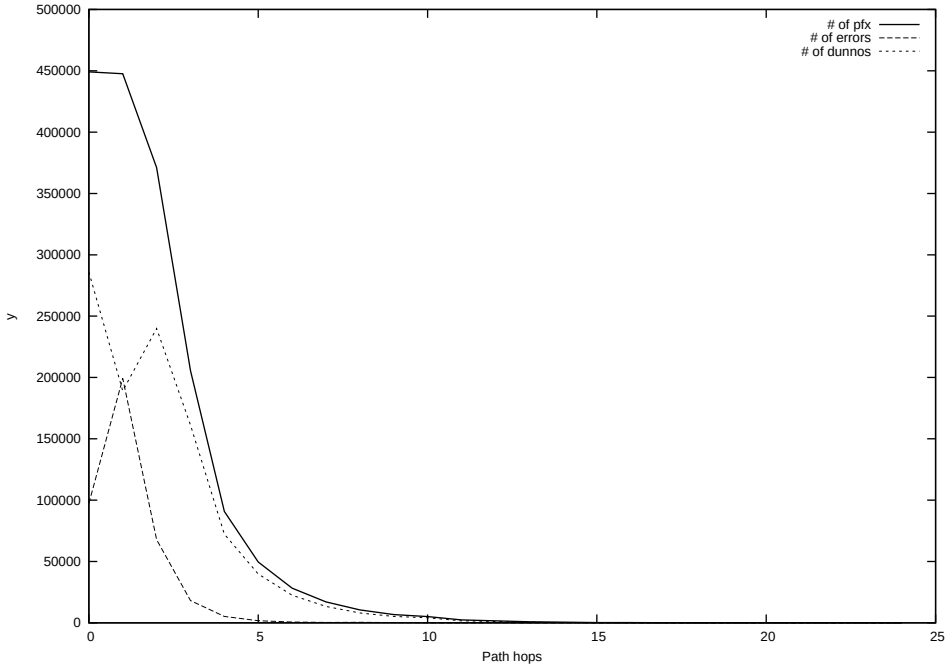
2013-04-19



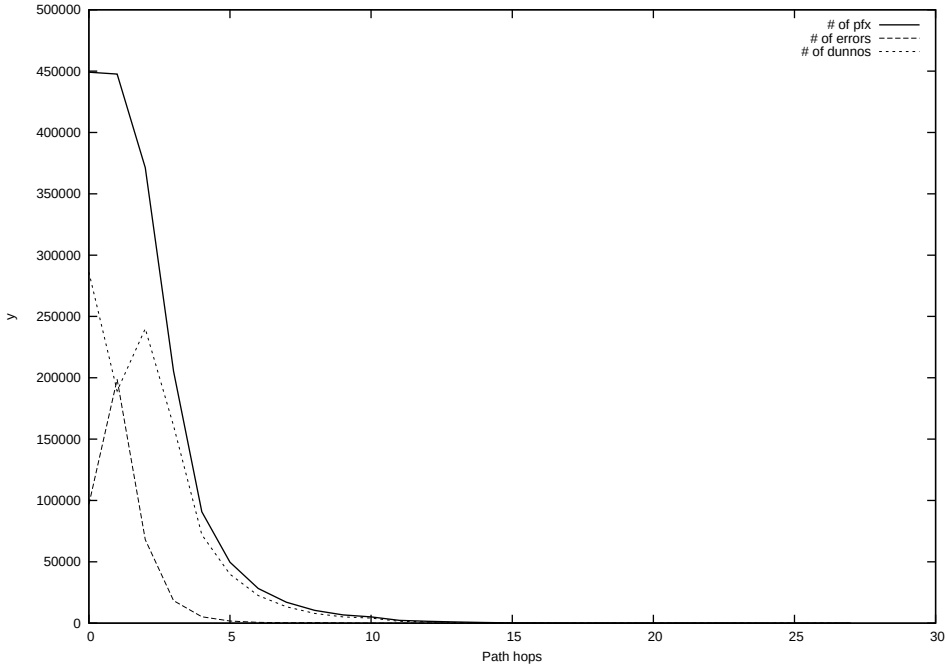
2013-04-20



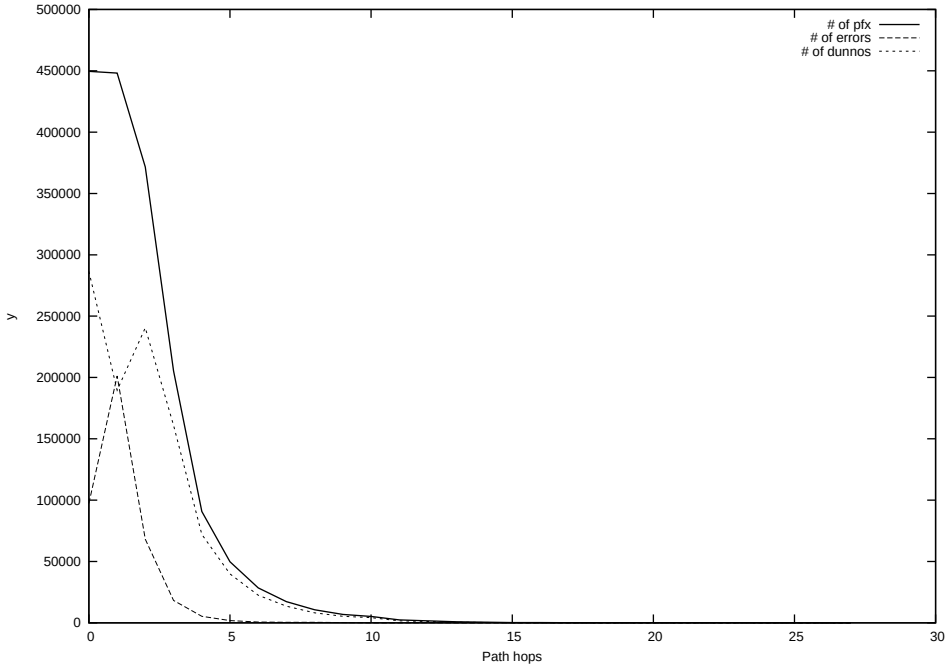
2013-04-21



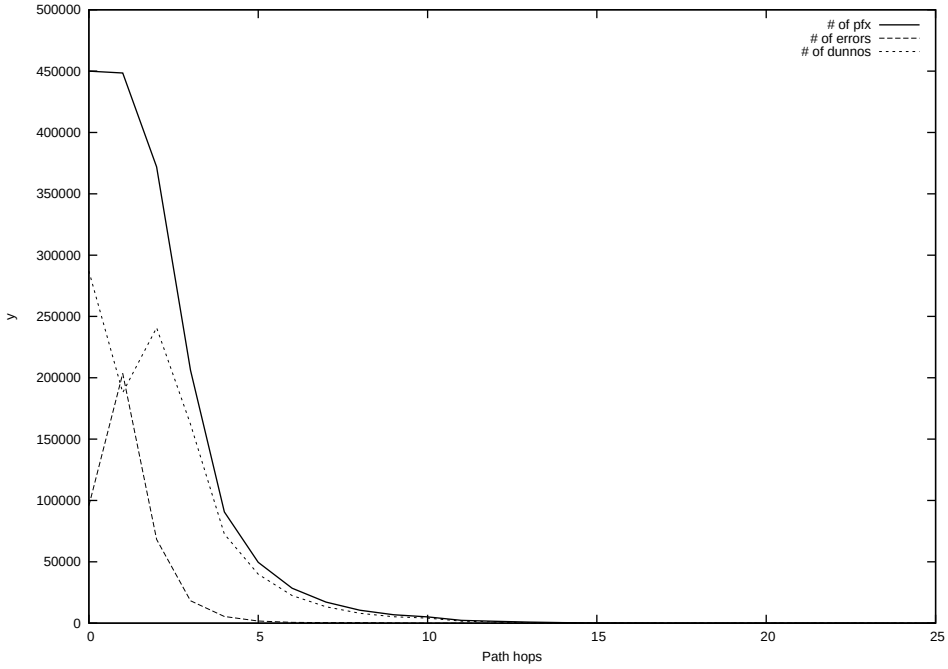
2013-04-22



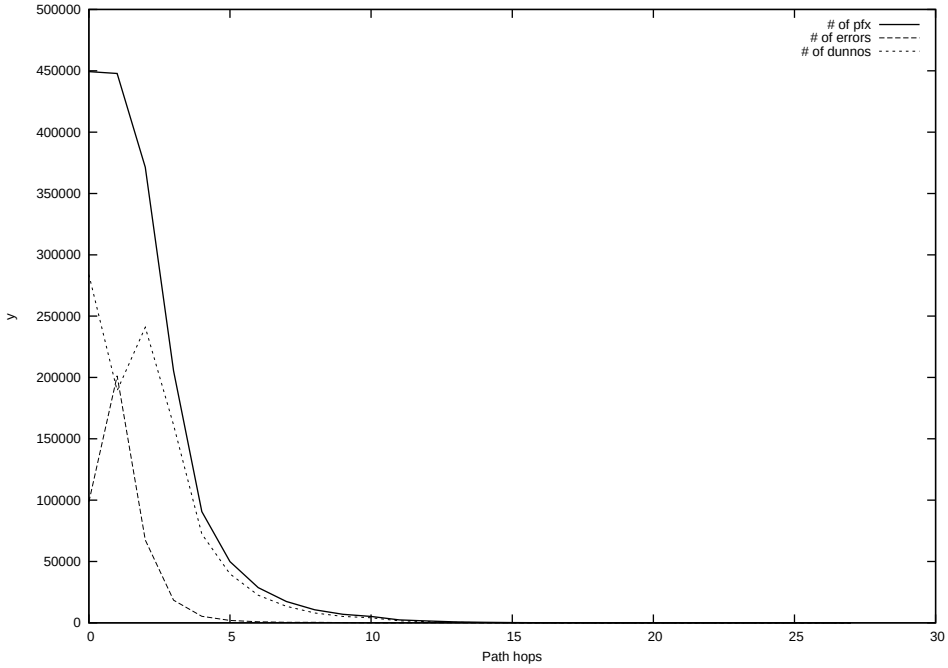
2013-04-23



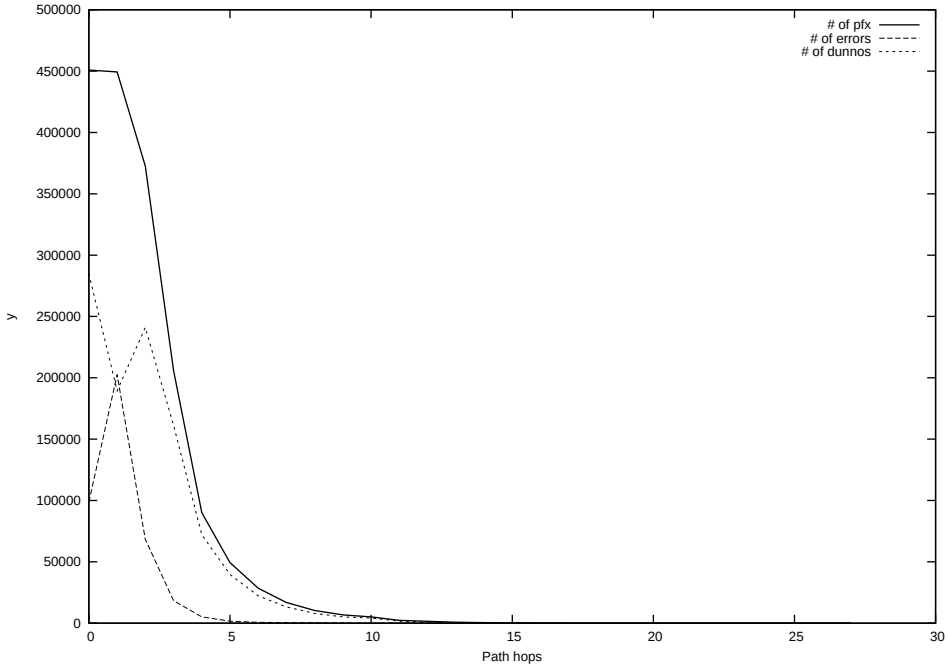
2013-04-24



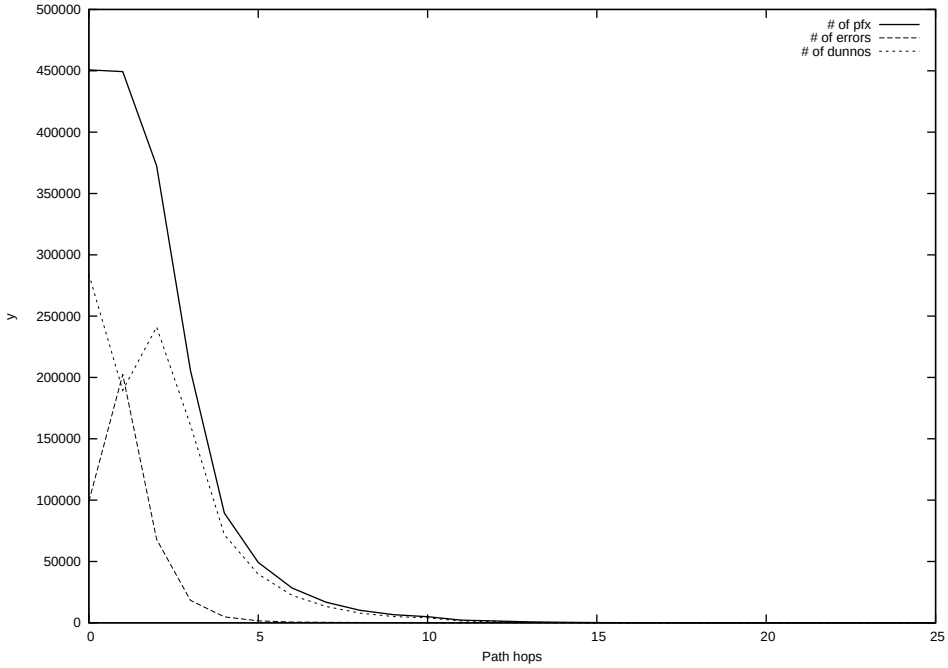
2013-04-25



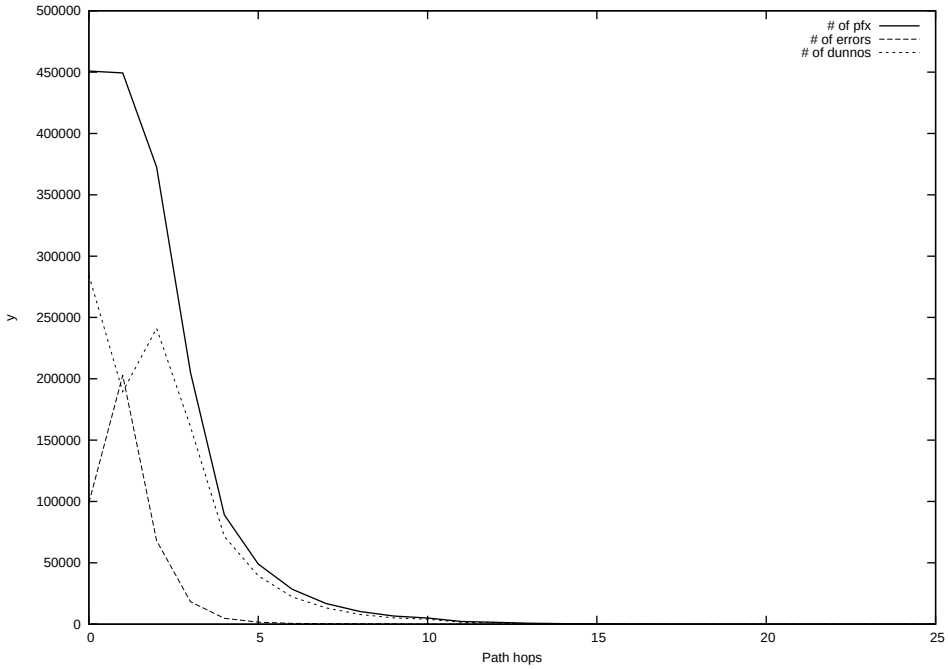
2013-04-26



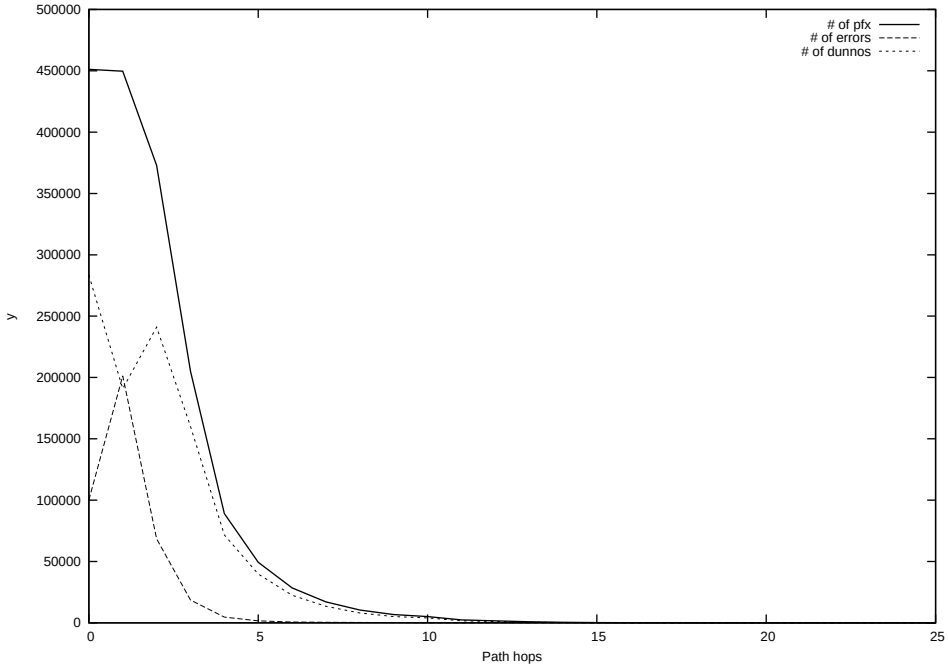
2013-04-27



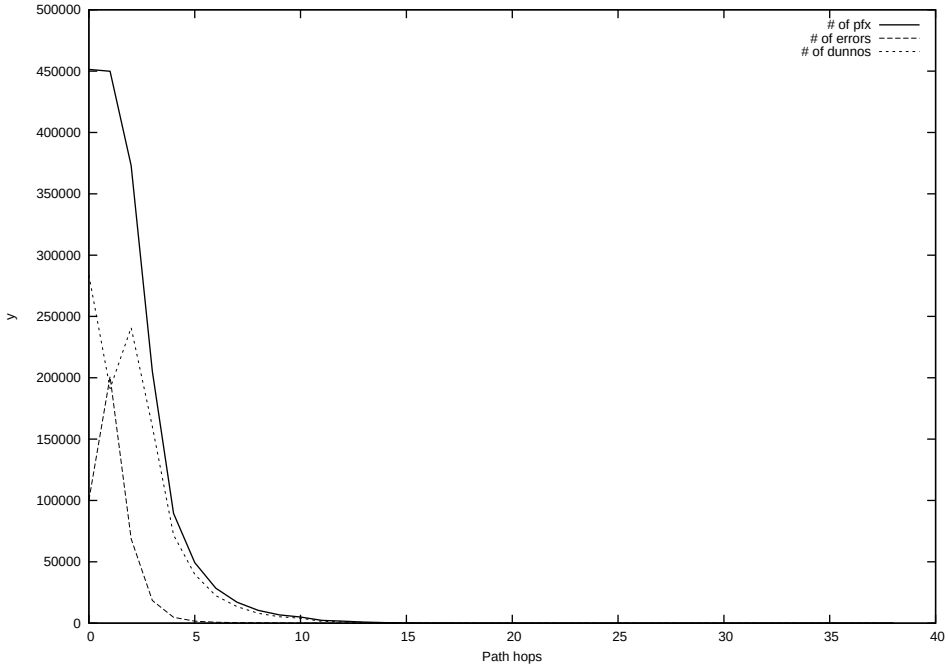
2013-04-28



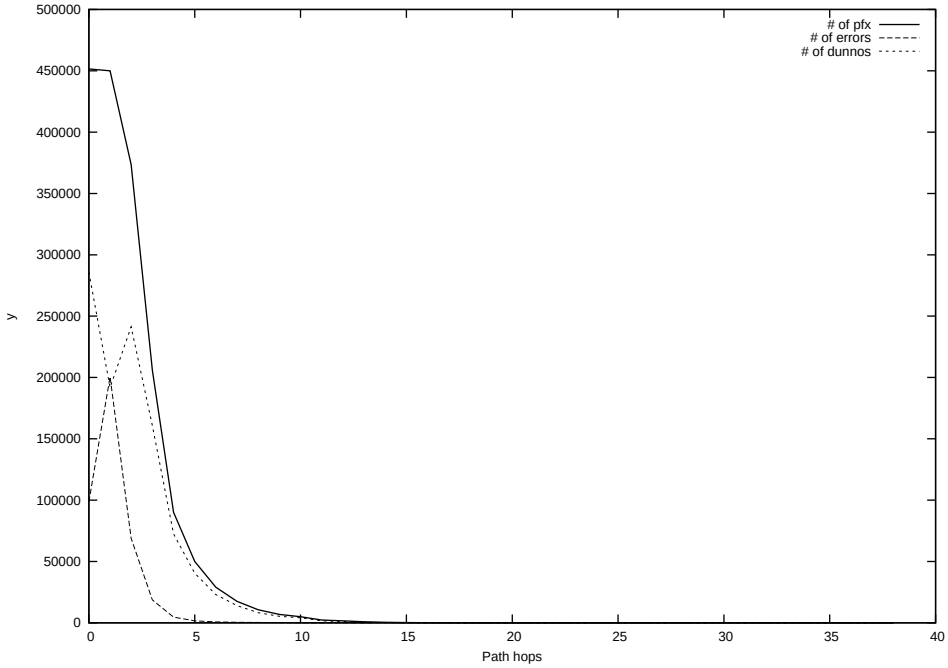
2013-04-29



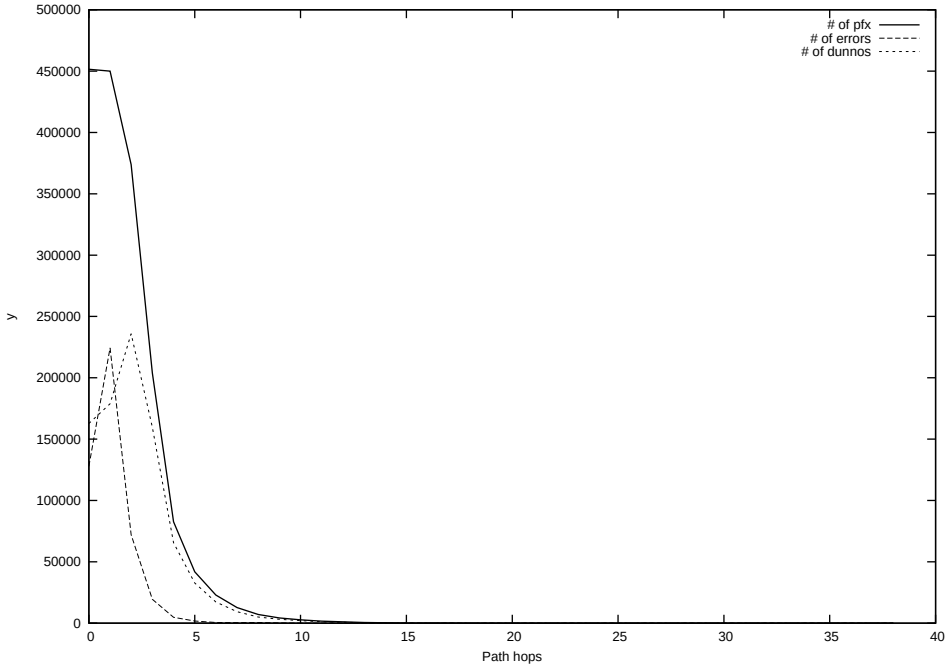
2013-04-30



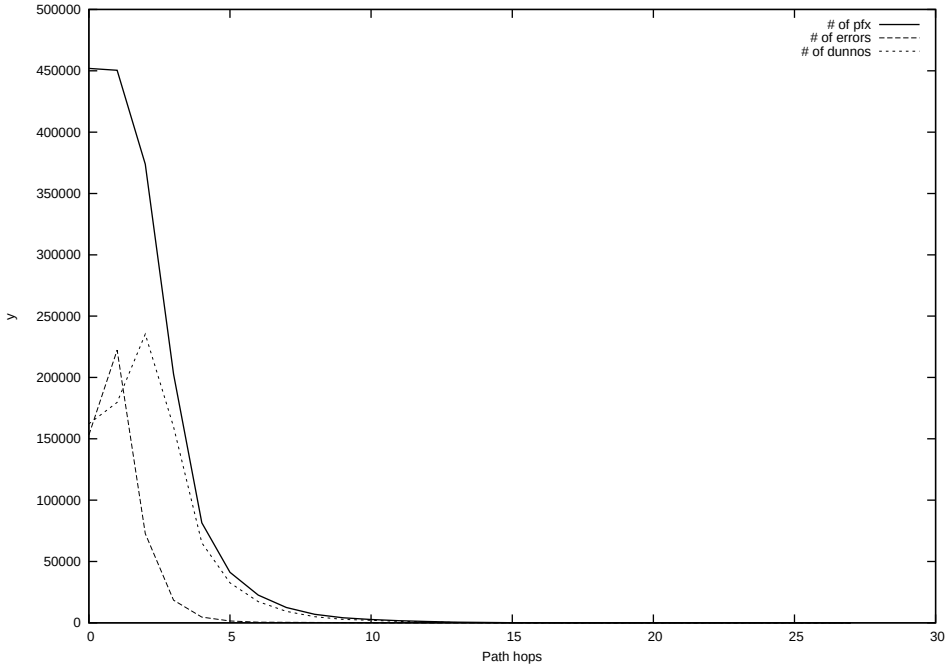
2013-05-01



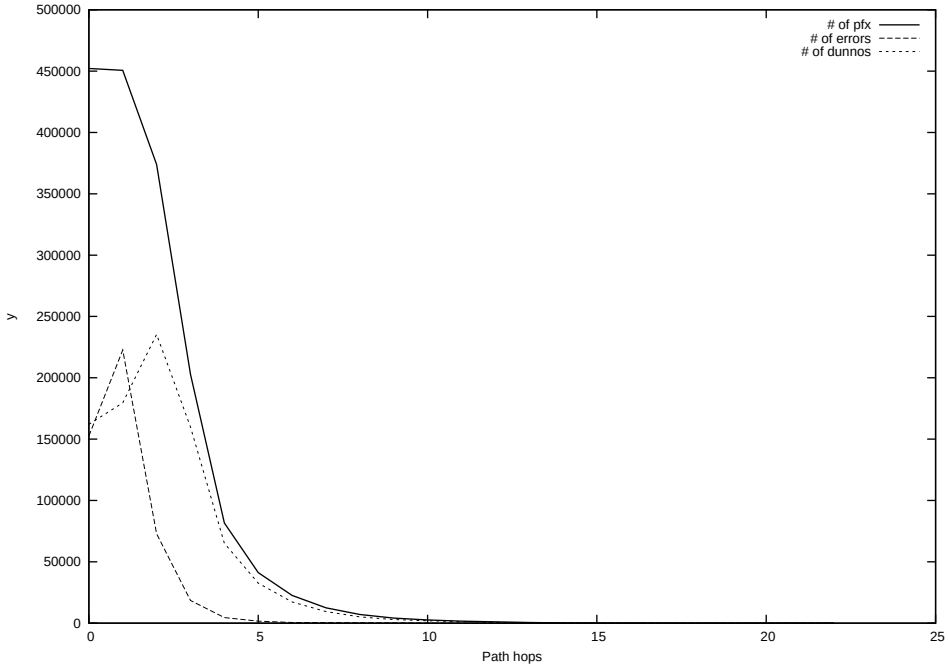
2013-05-02



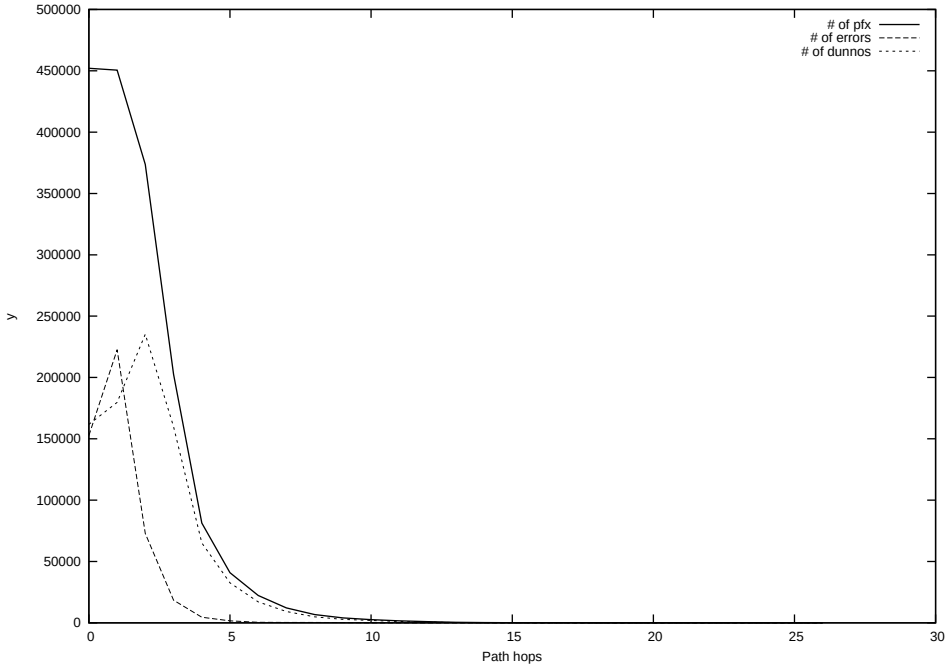
2013-05-03



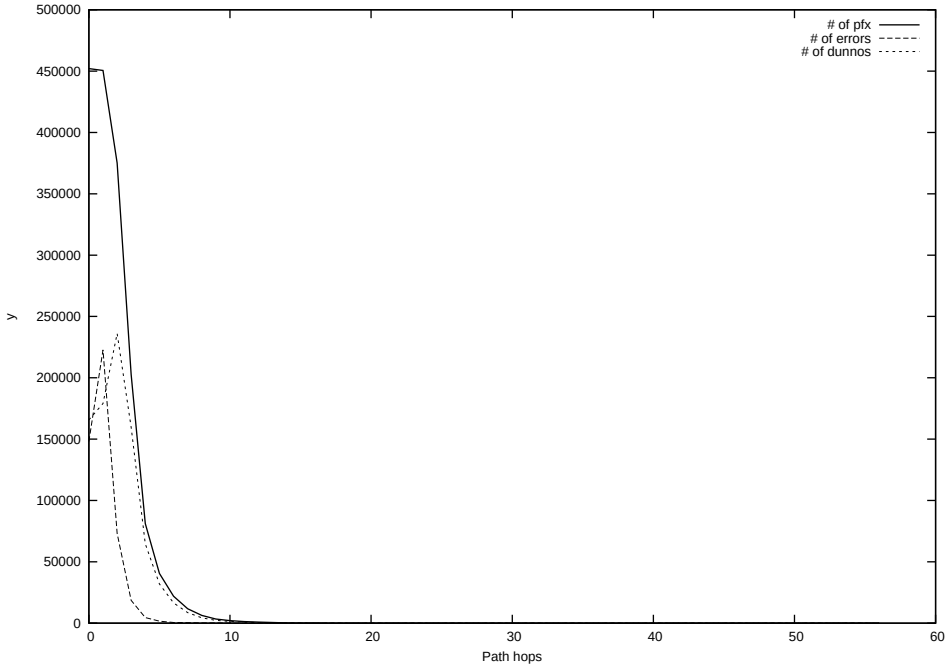
2013-05-04



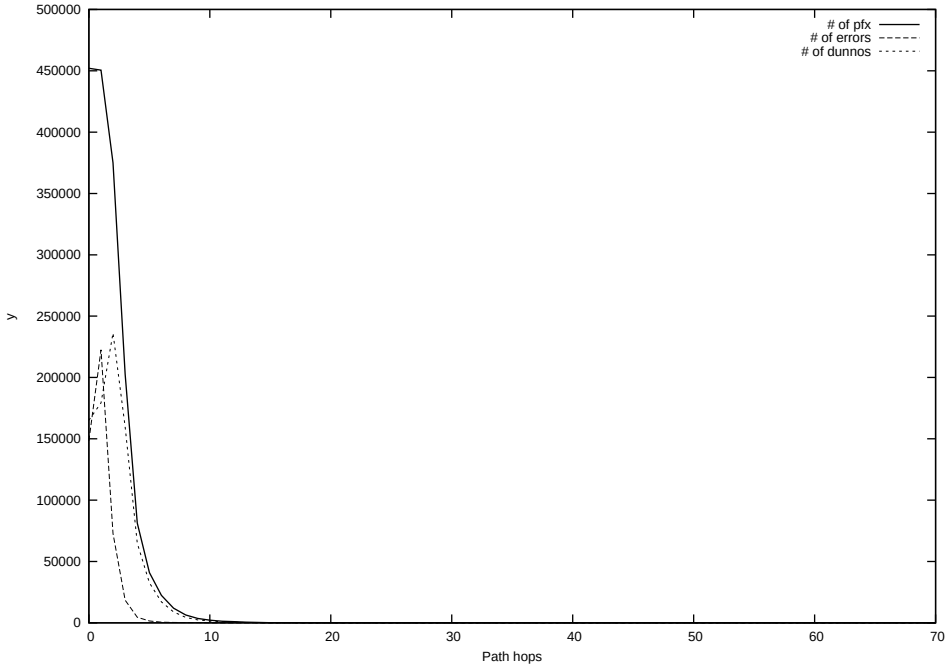
2013-05-05



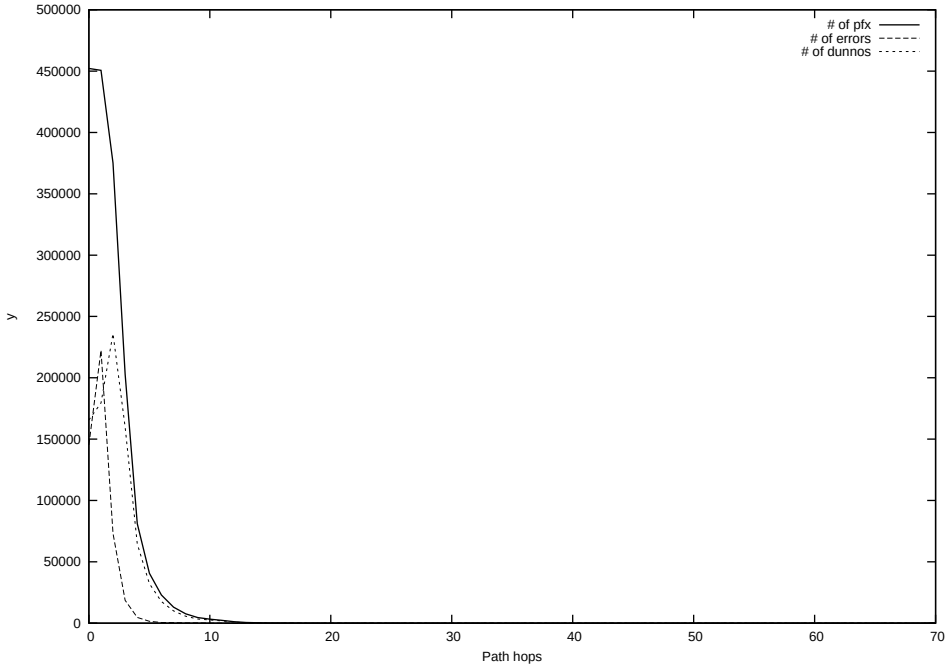
2013-05-06



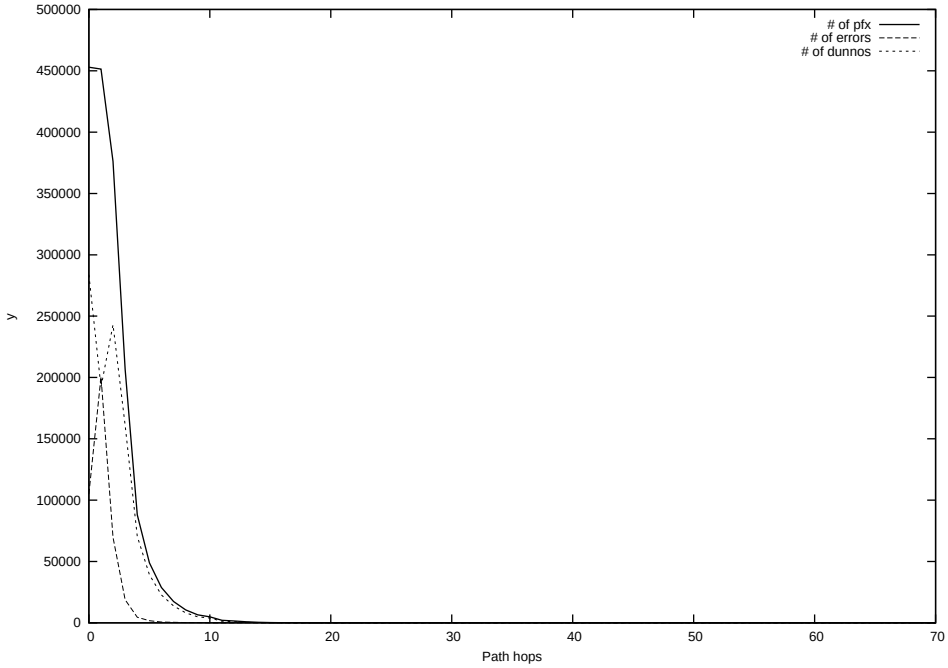
2013-05-07



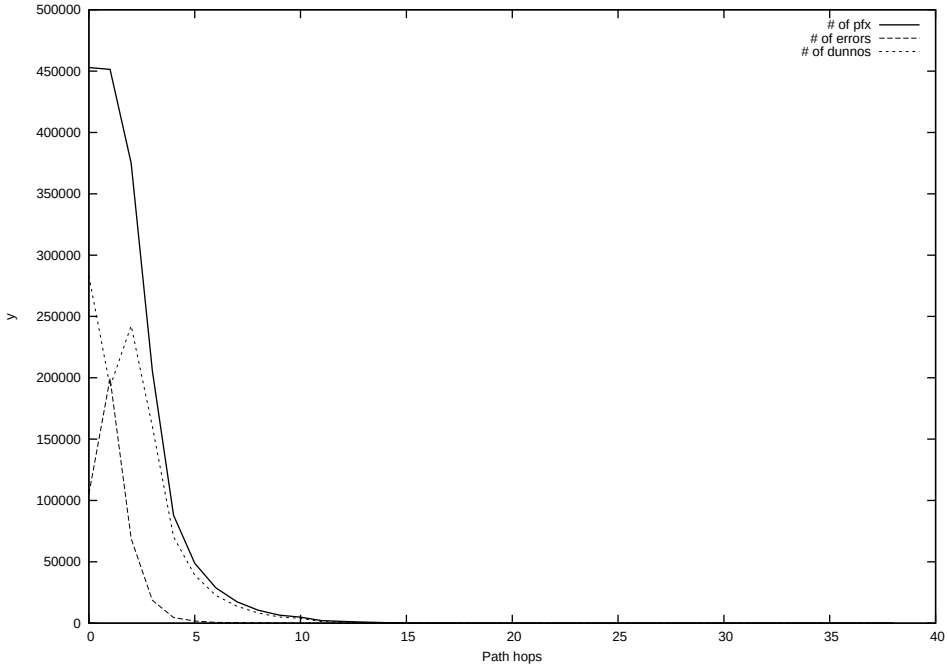
2013-05-08



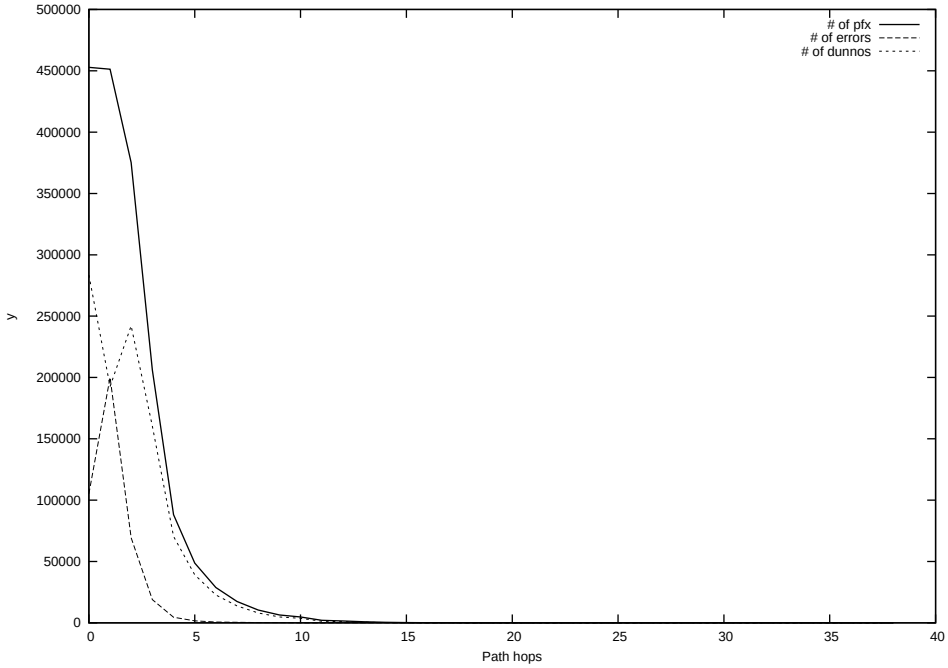
2013-05-09



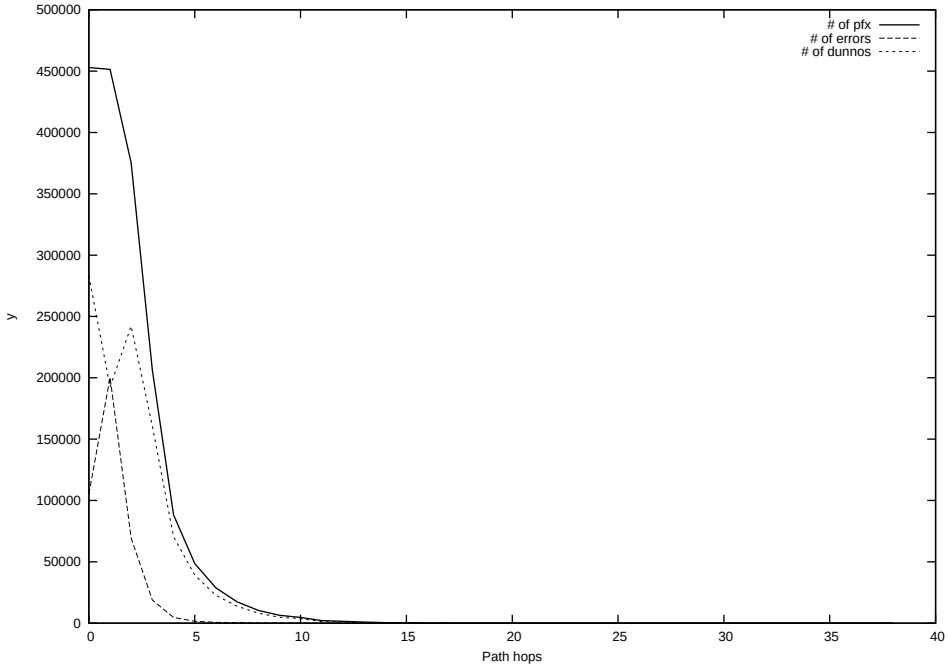
2013-05-10



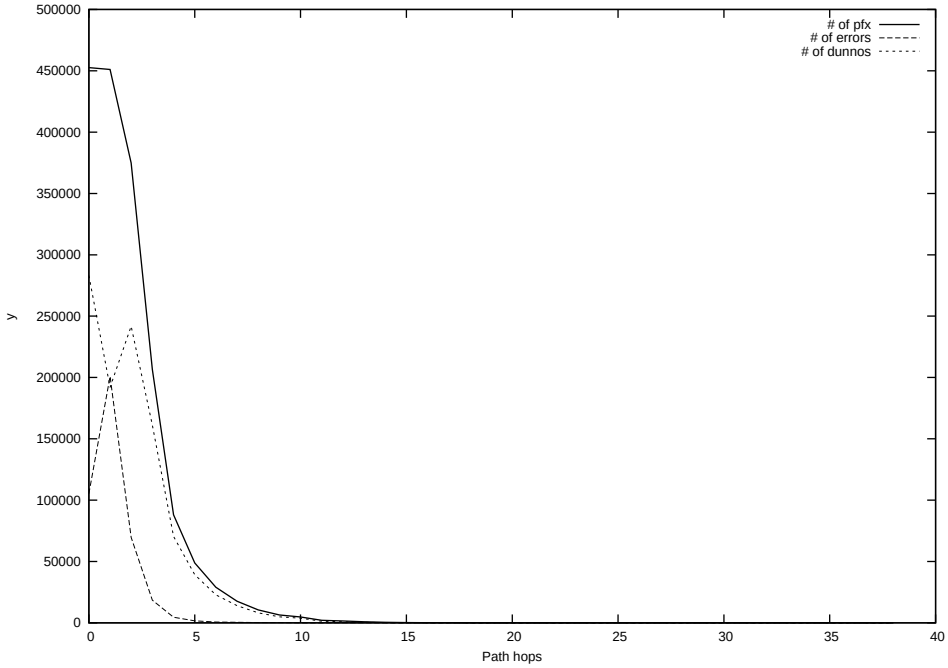
2013-05-11



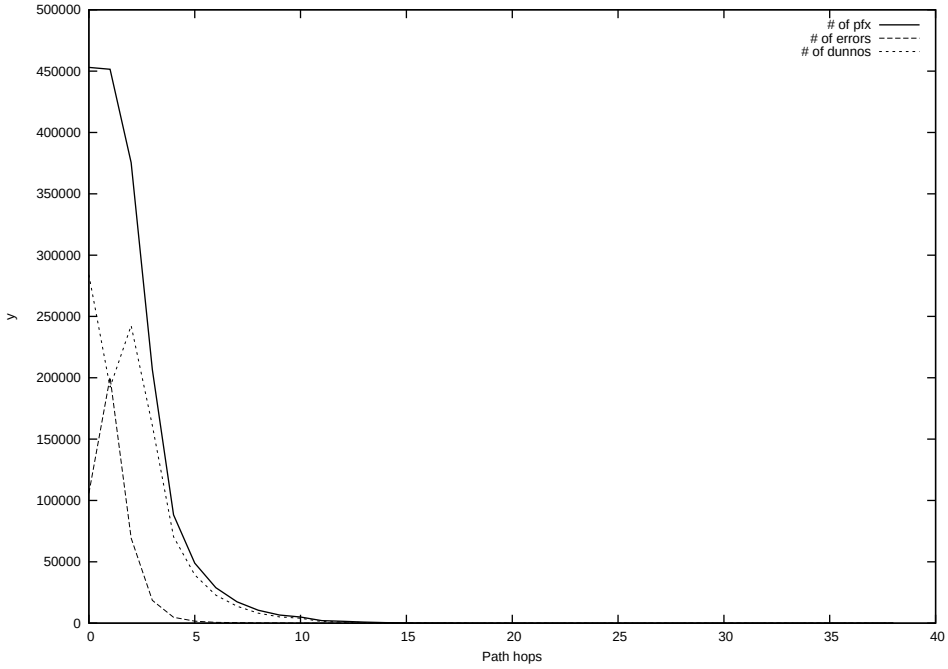
2013-05-12



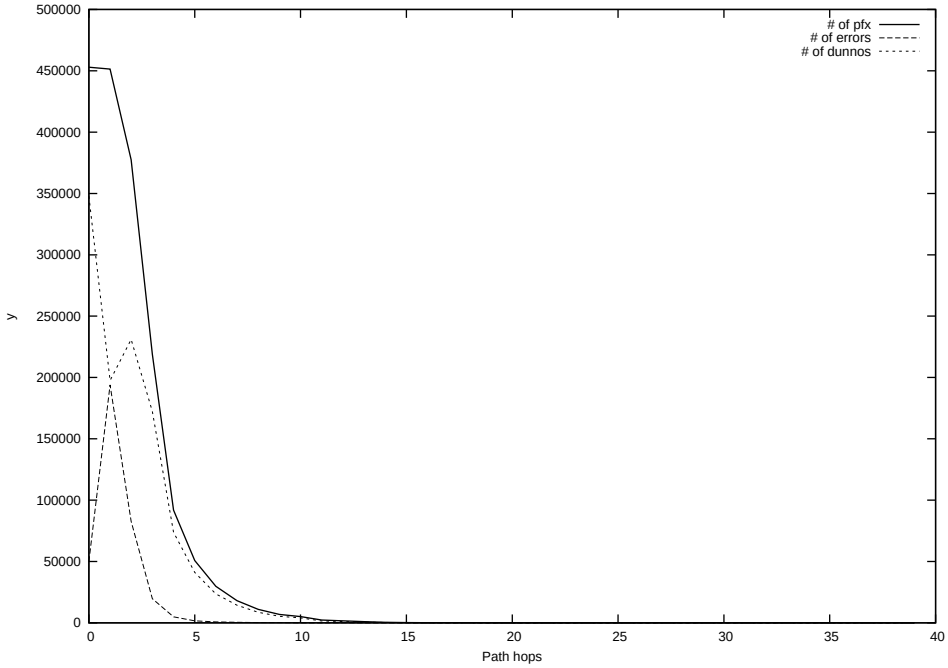
2013-05-13



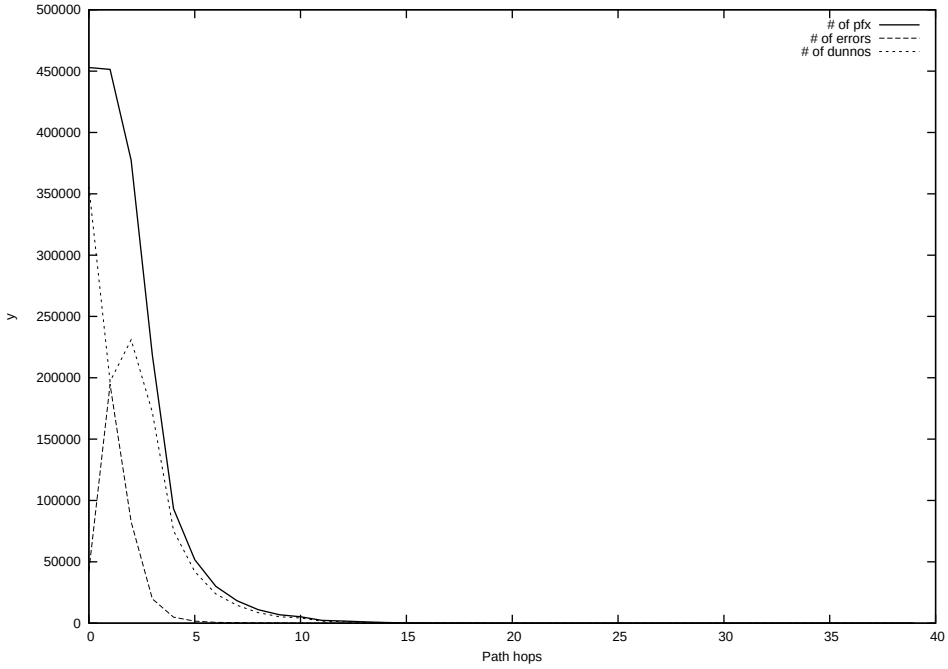
2013-05-14



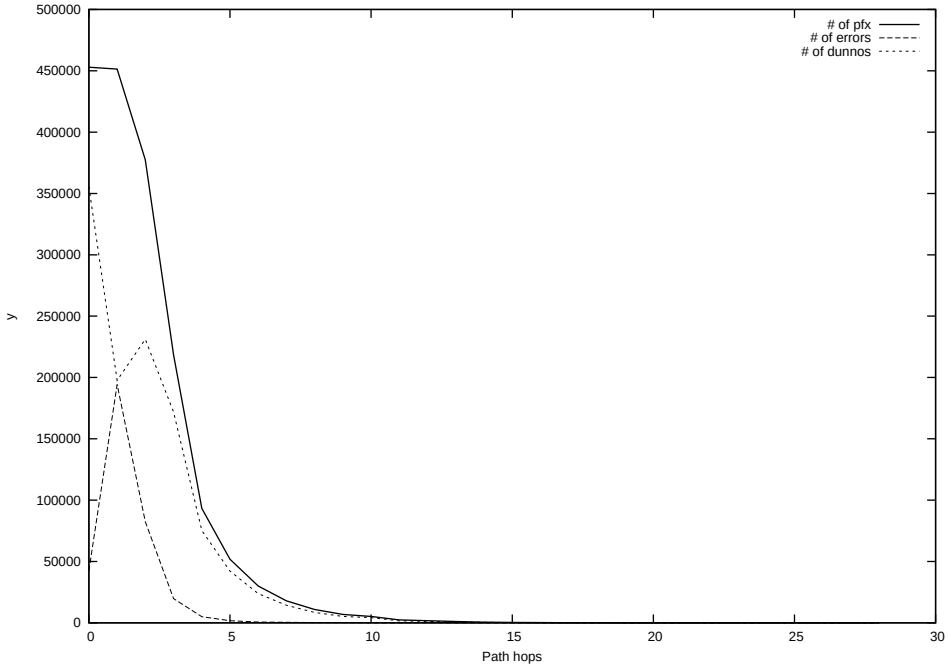
2013-05-15



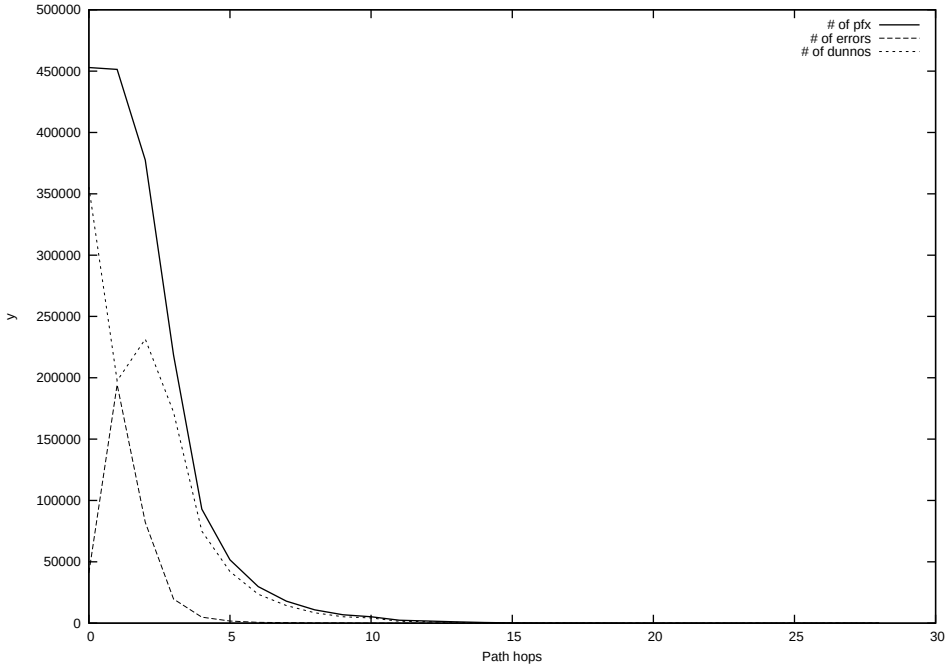
2013-05-16



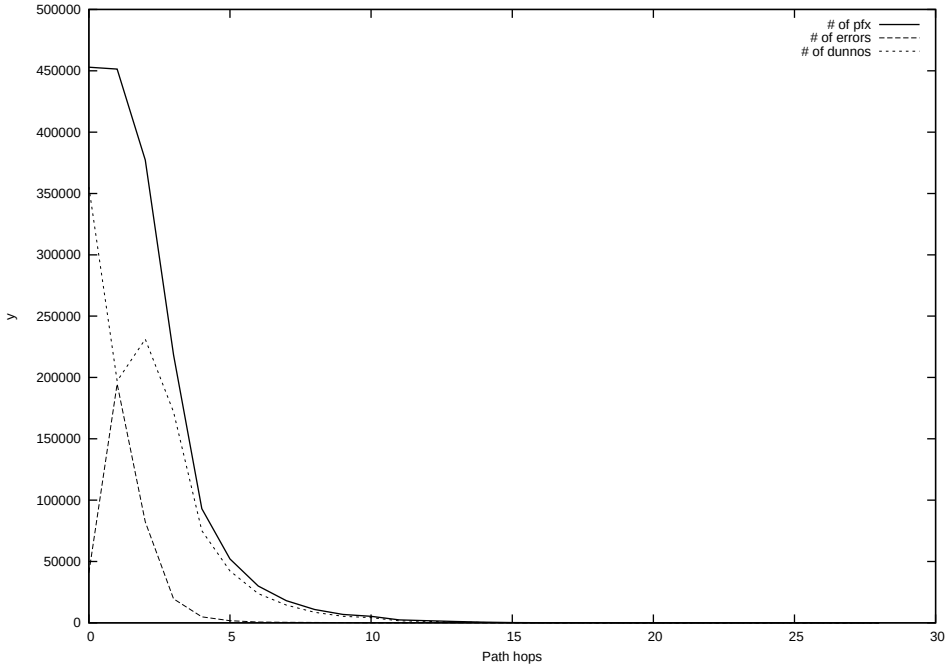
2013-05-17



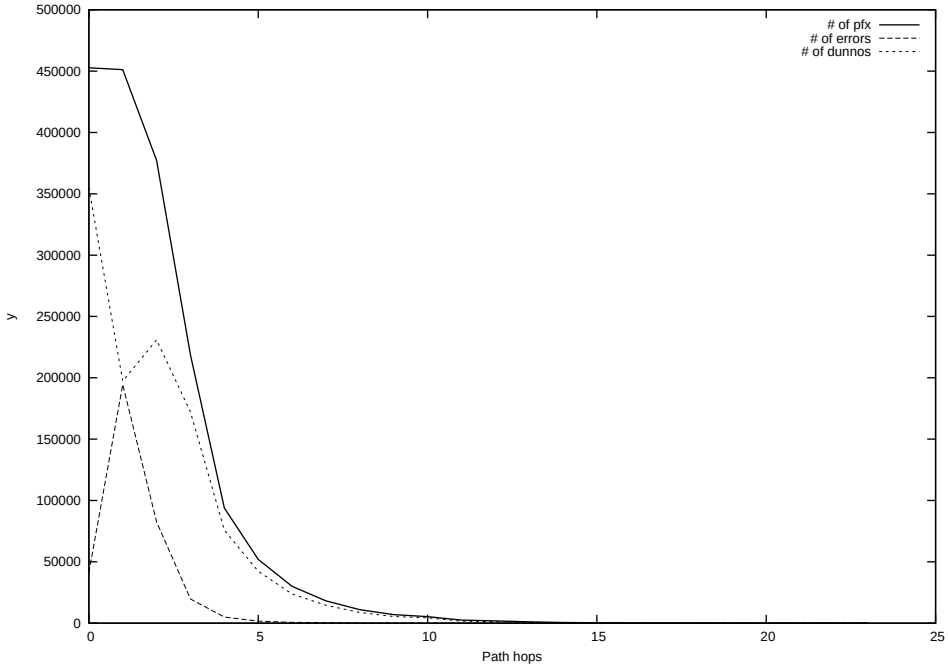
2013-05-18



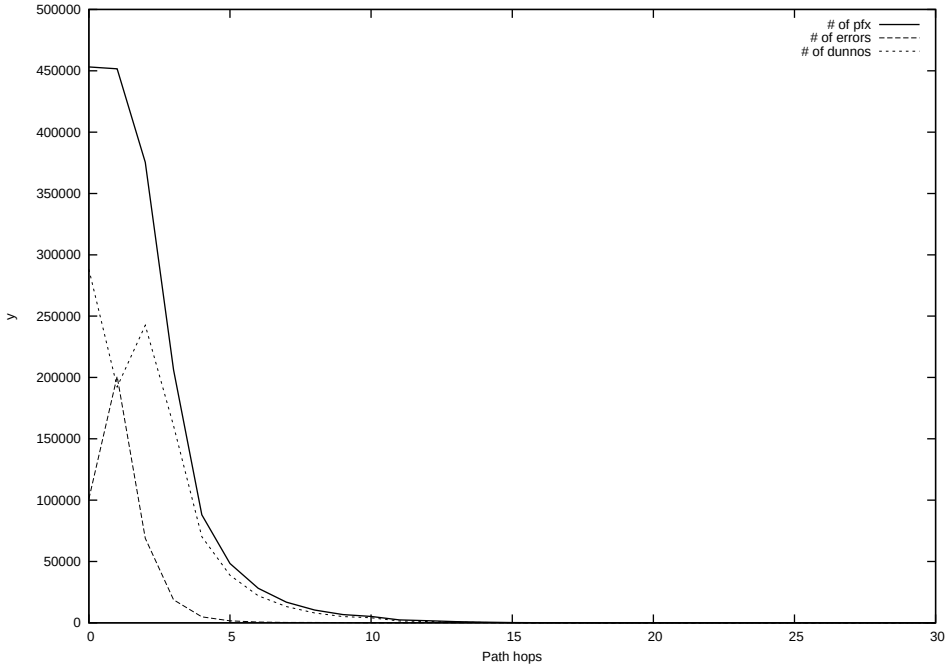
2013-05-19



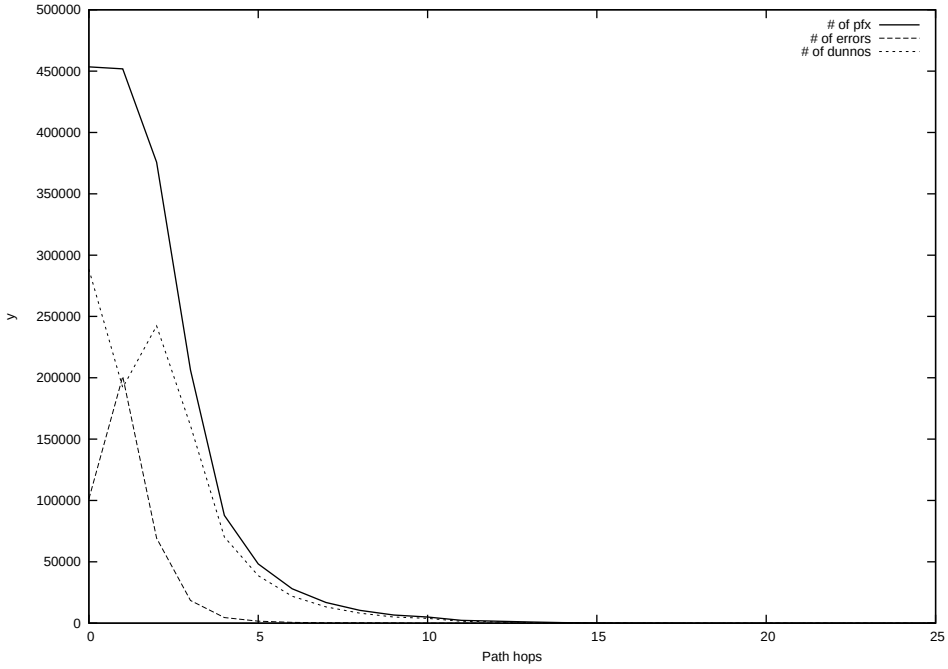
2013-05-20



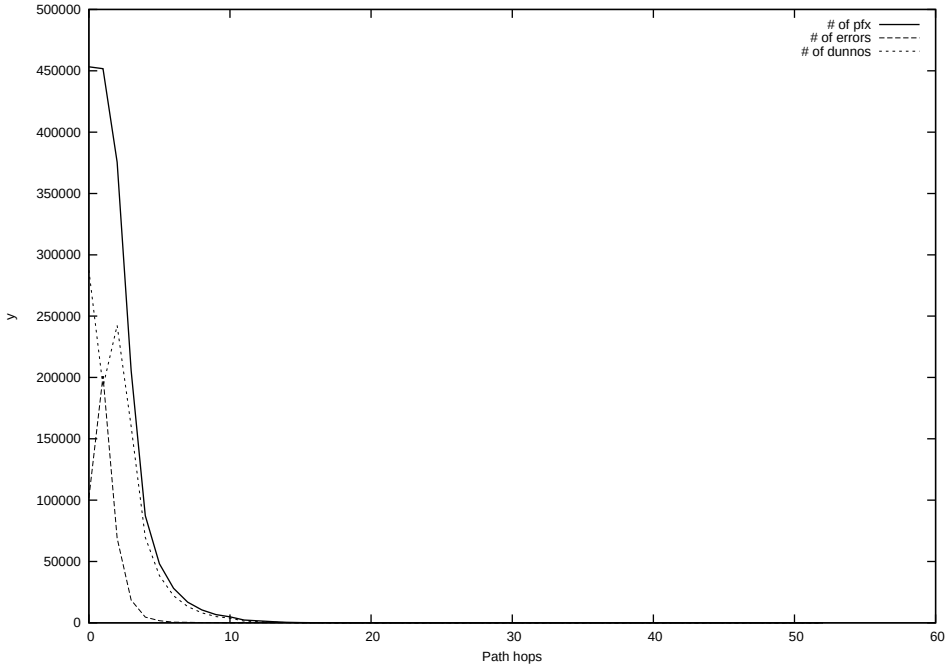
2013-05-21



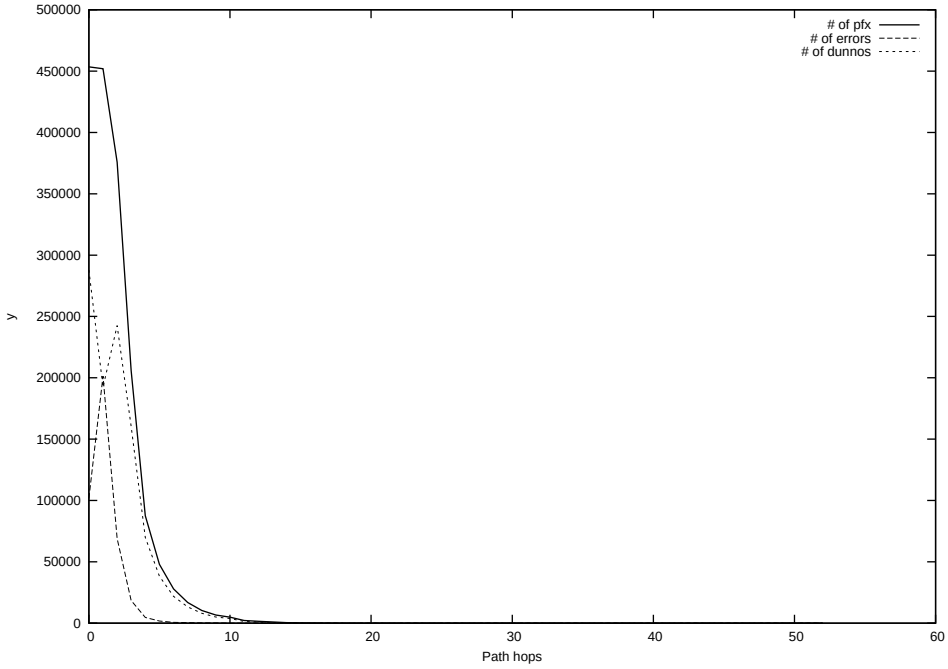
2013-05-22



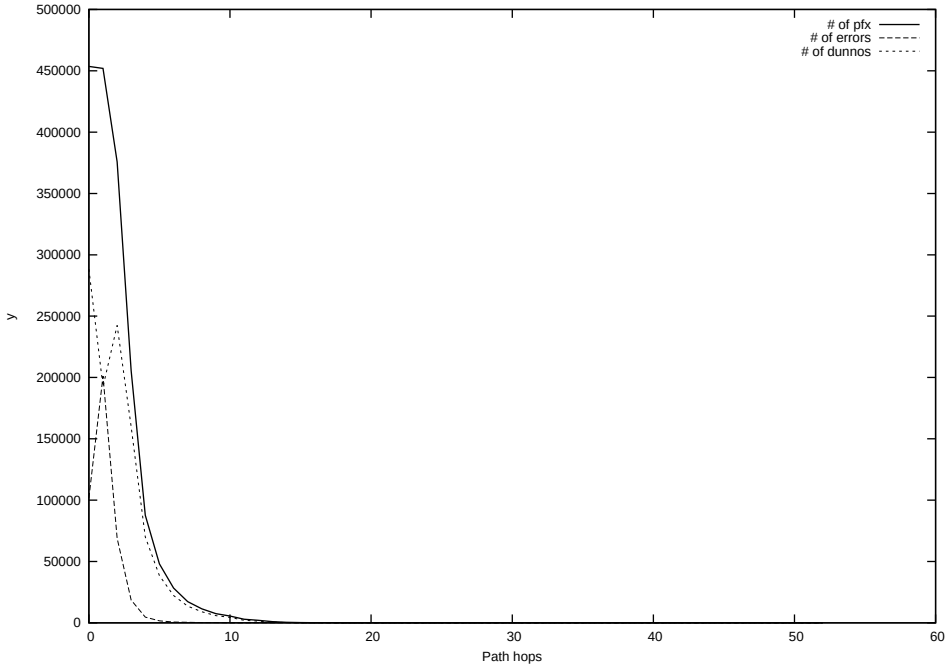
2013-05-23



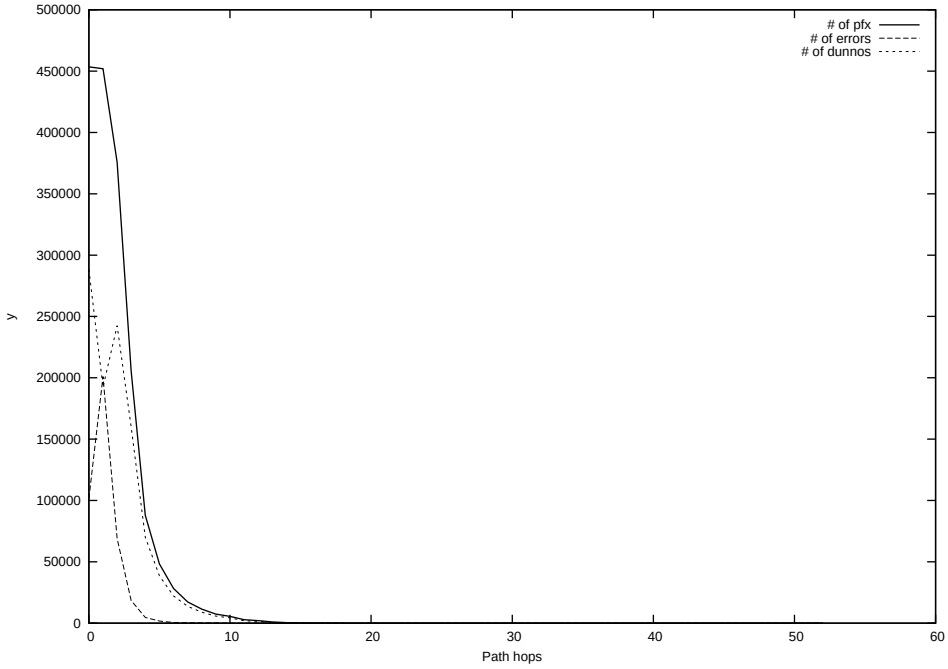
2013-05-24



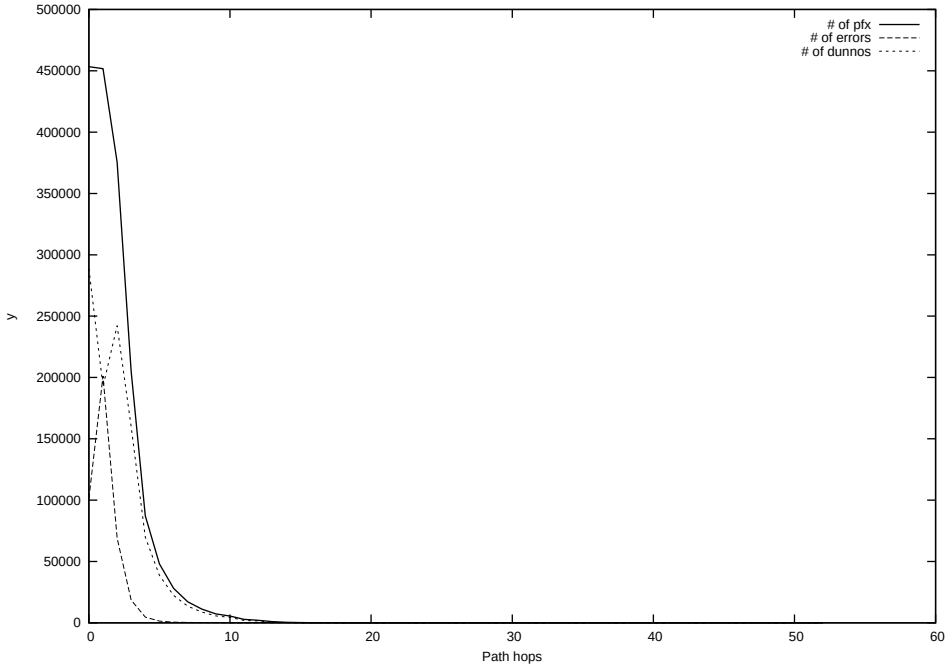
2013-05-25



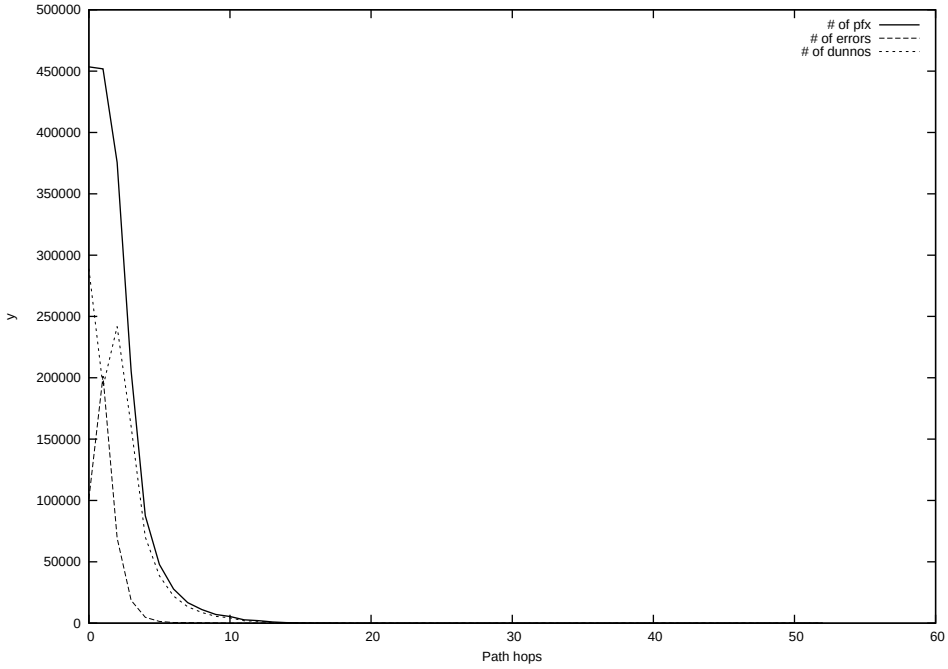
2013-05-26



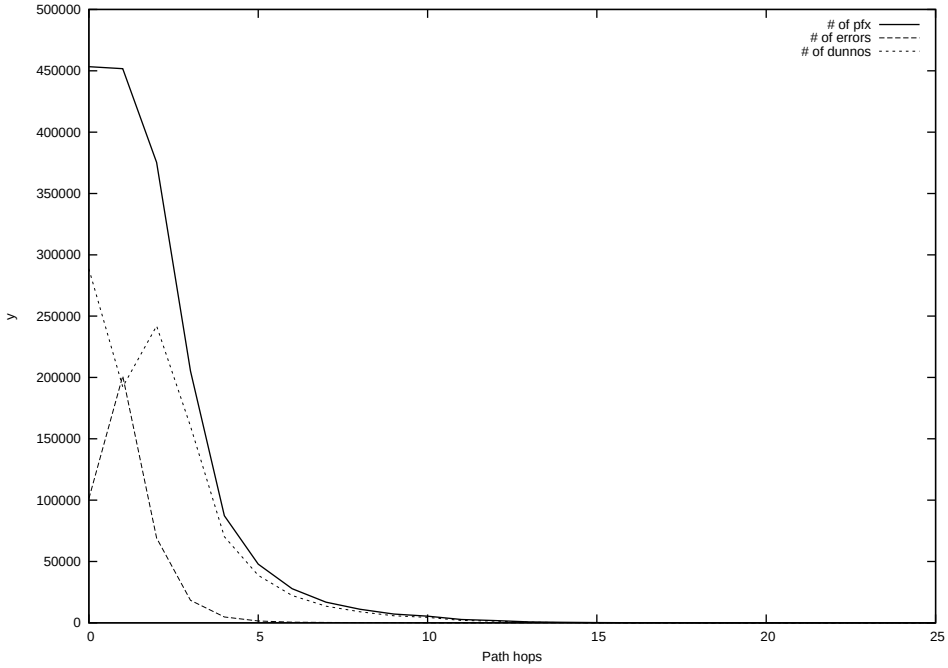
2013-05-27



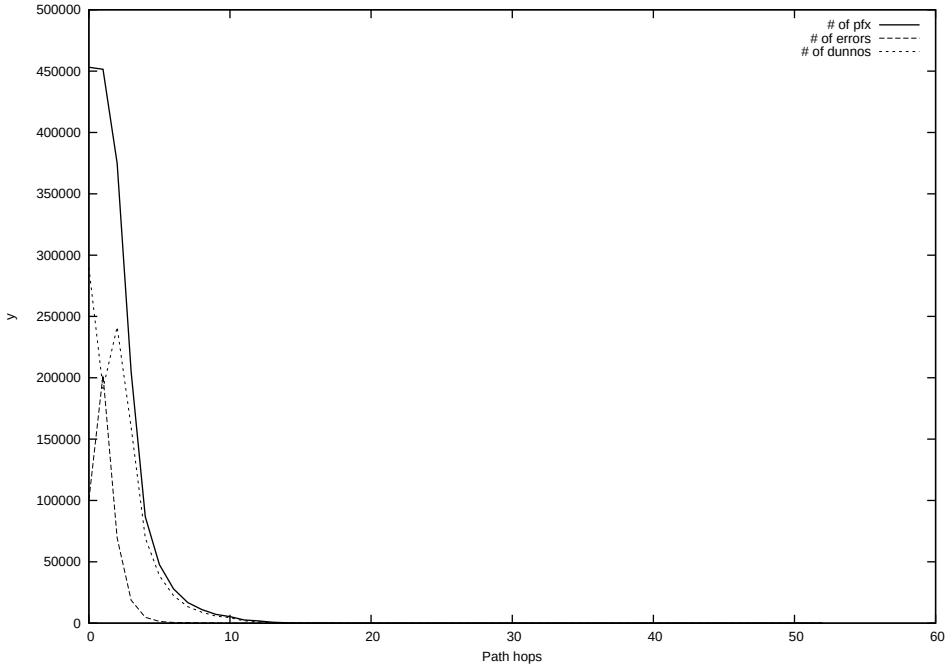
2013-05-28



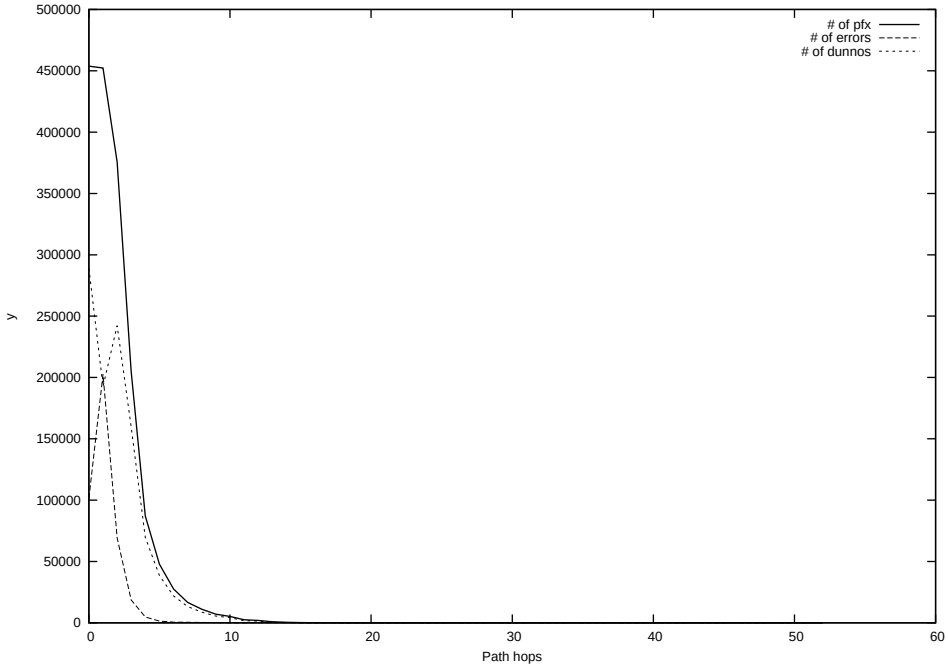
2013-05-29



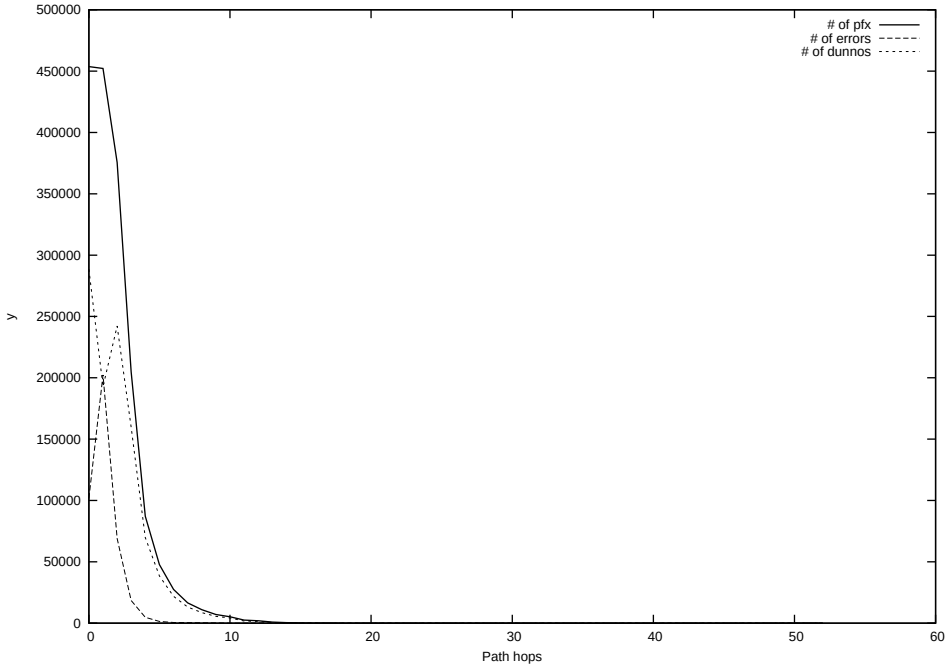
2013-05-30



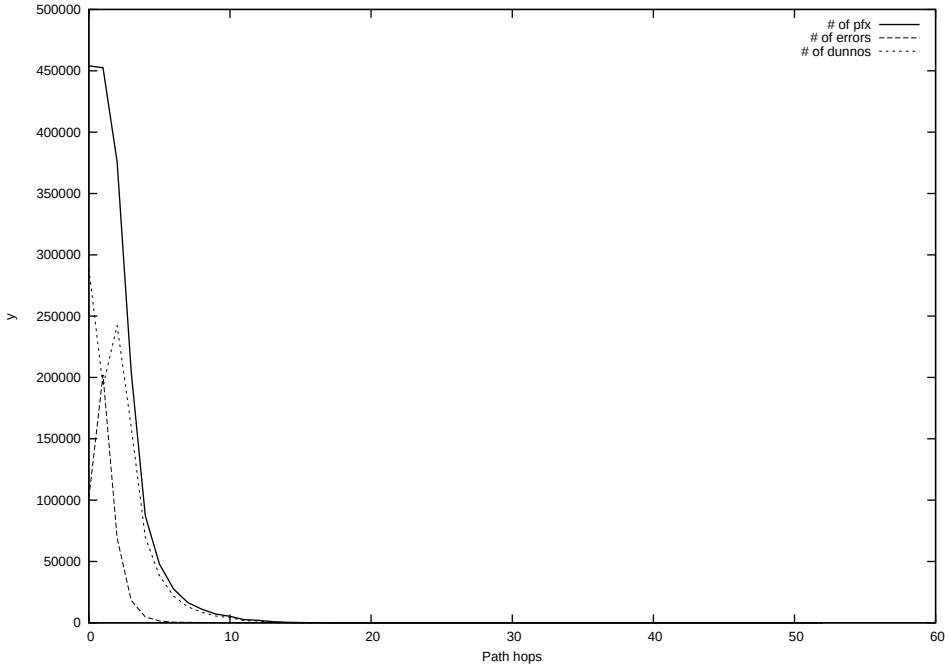
2013-05-31



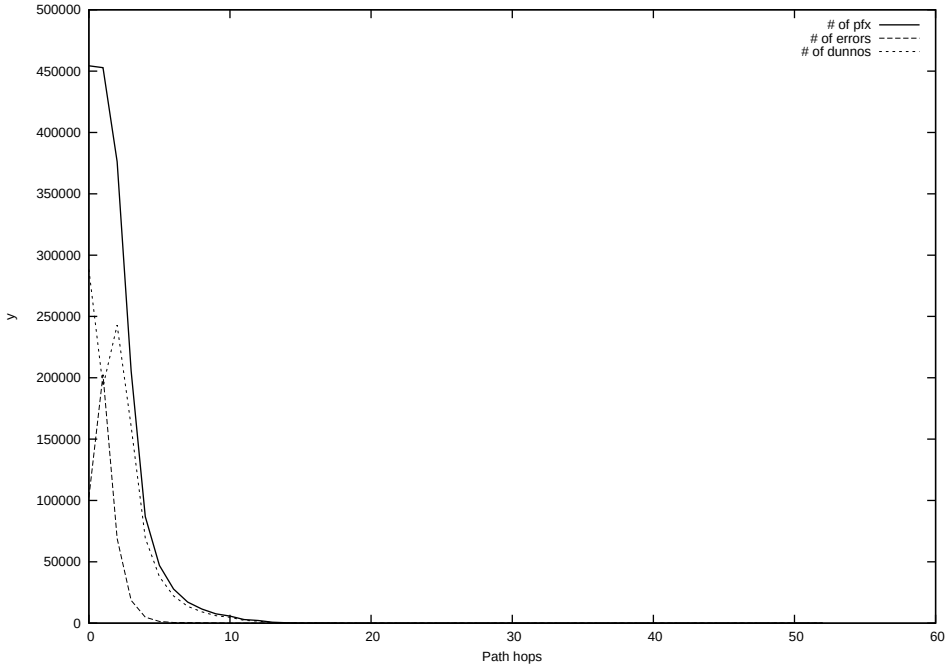
2013-06-01



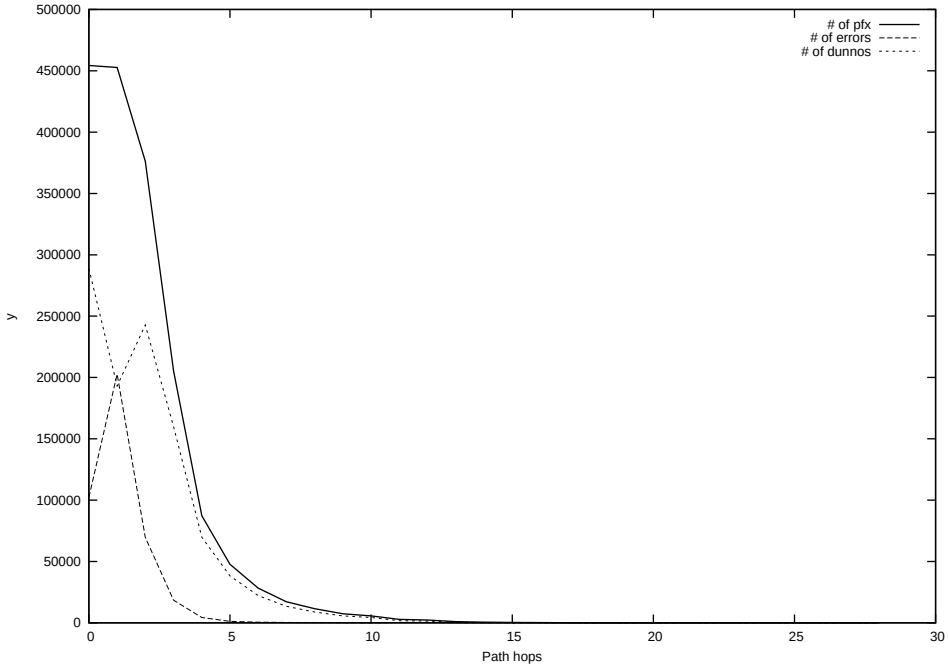
2013-06-02



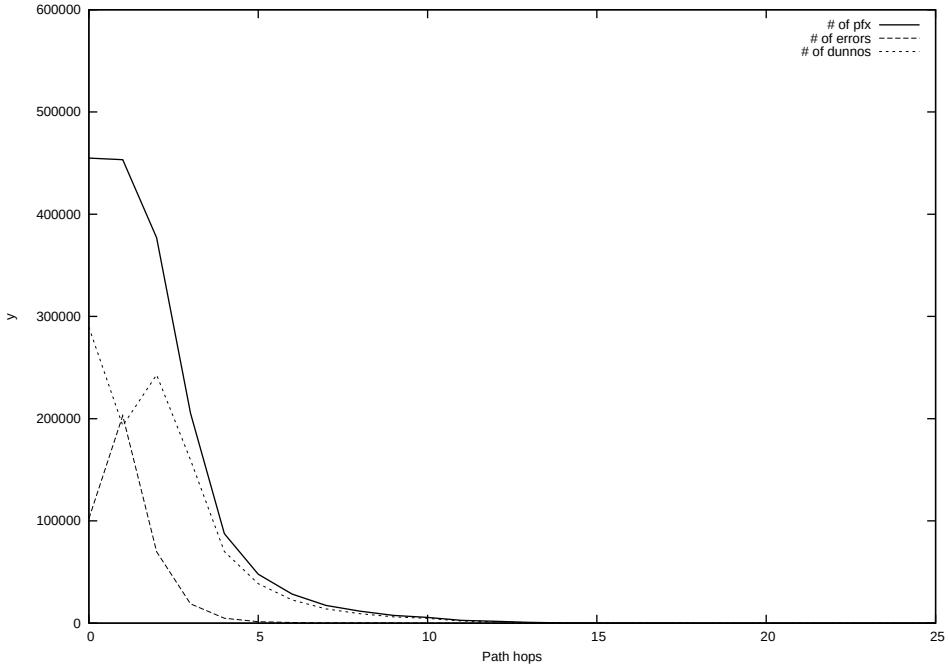
2013-06-03



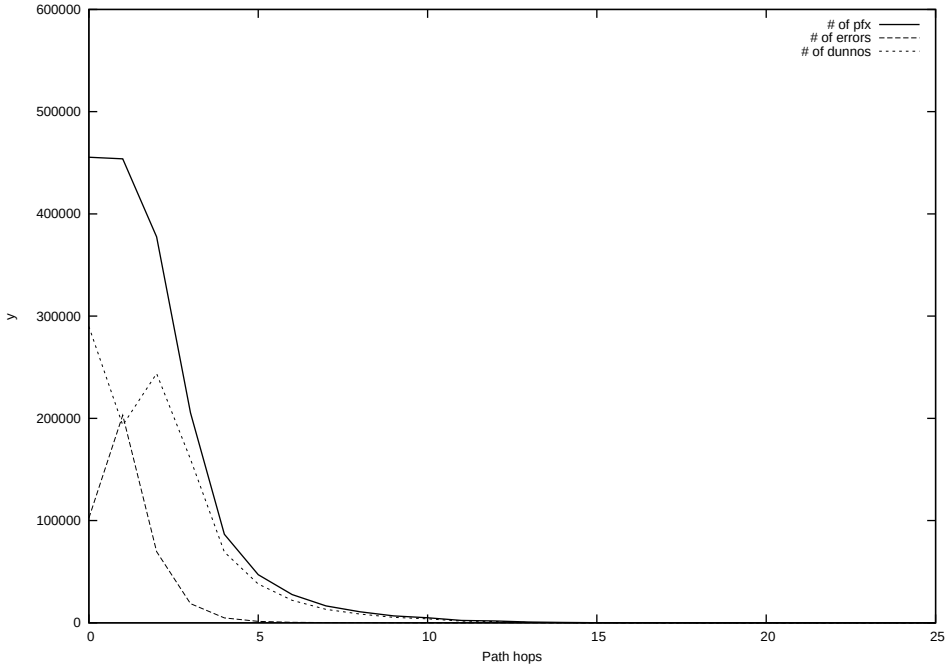
2013-06-04



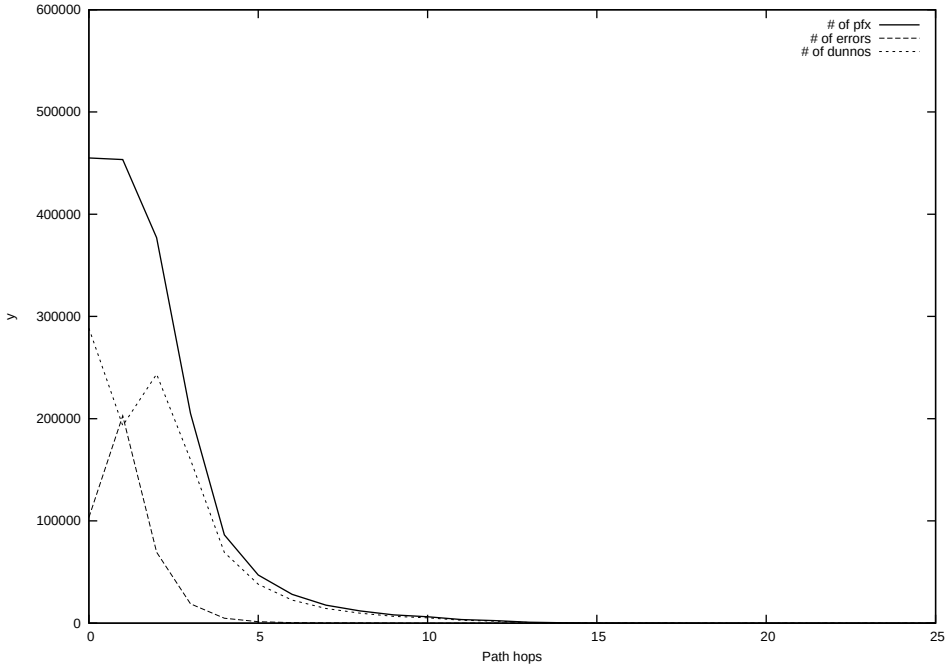
2013-06-05



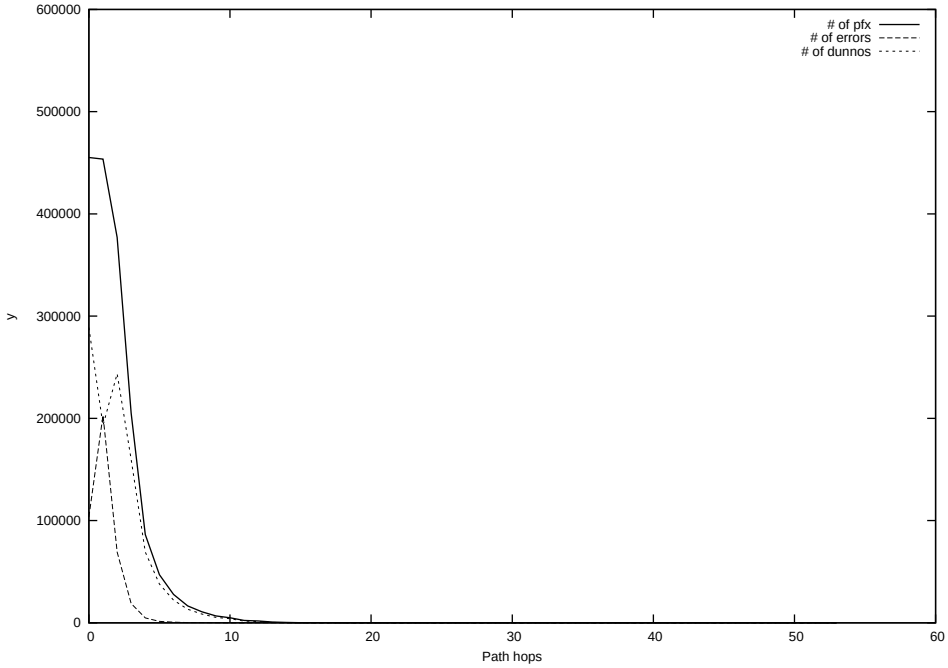
2013-06-06



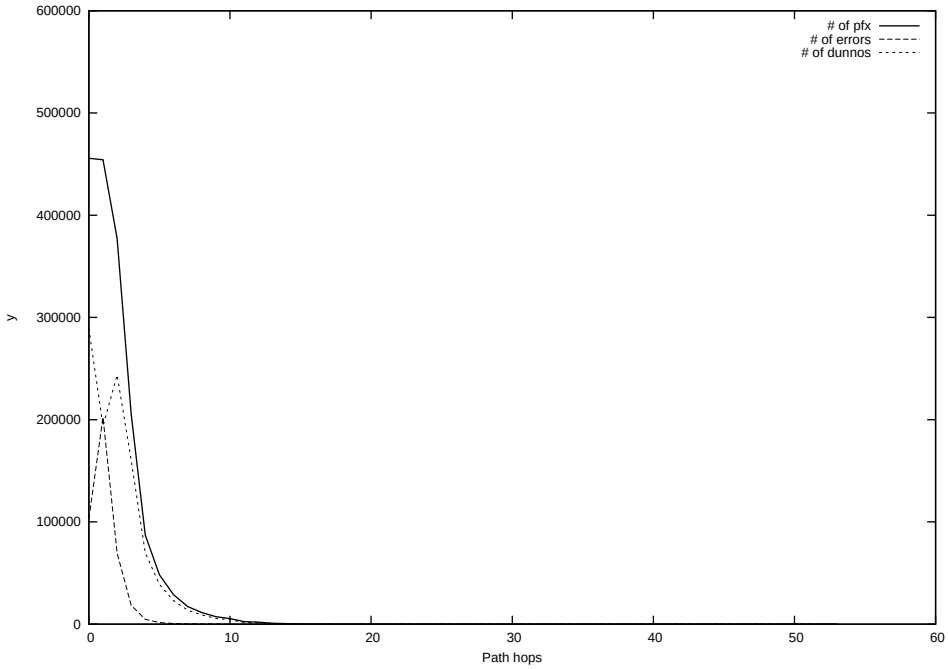
2013-06-07



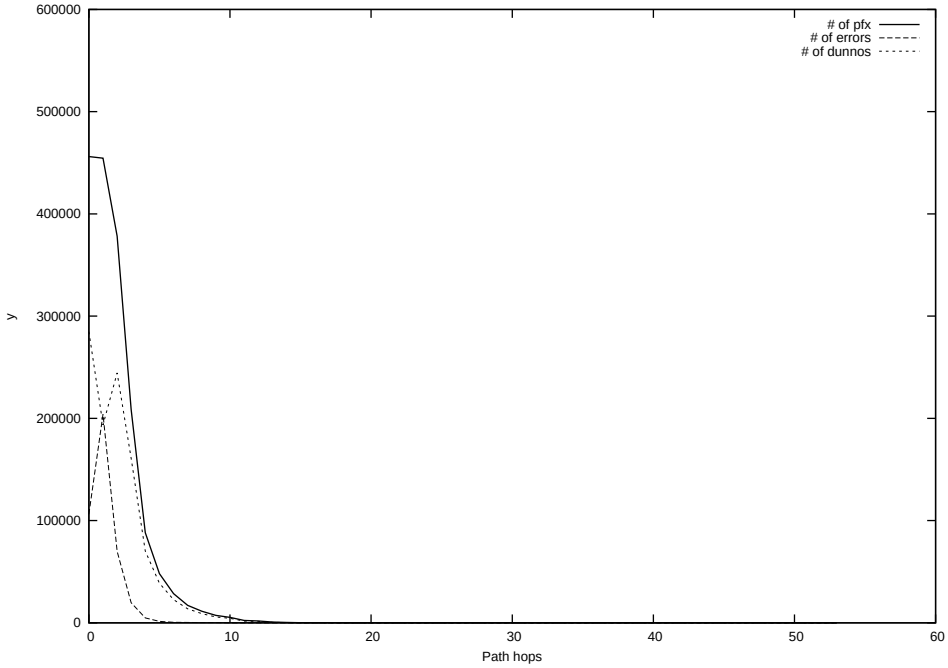
2013-06-09



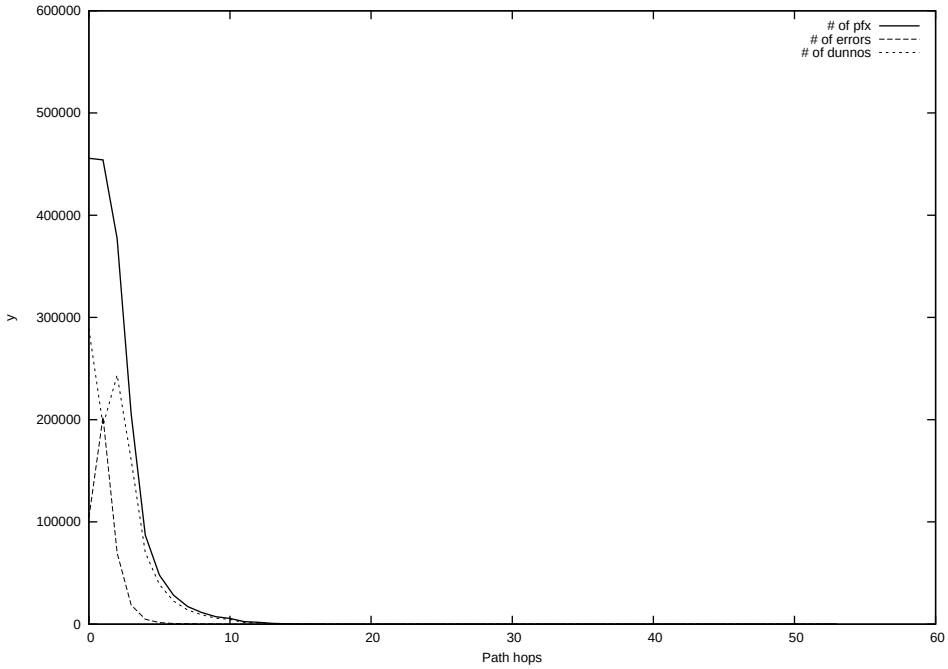
2013-06-10



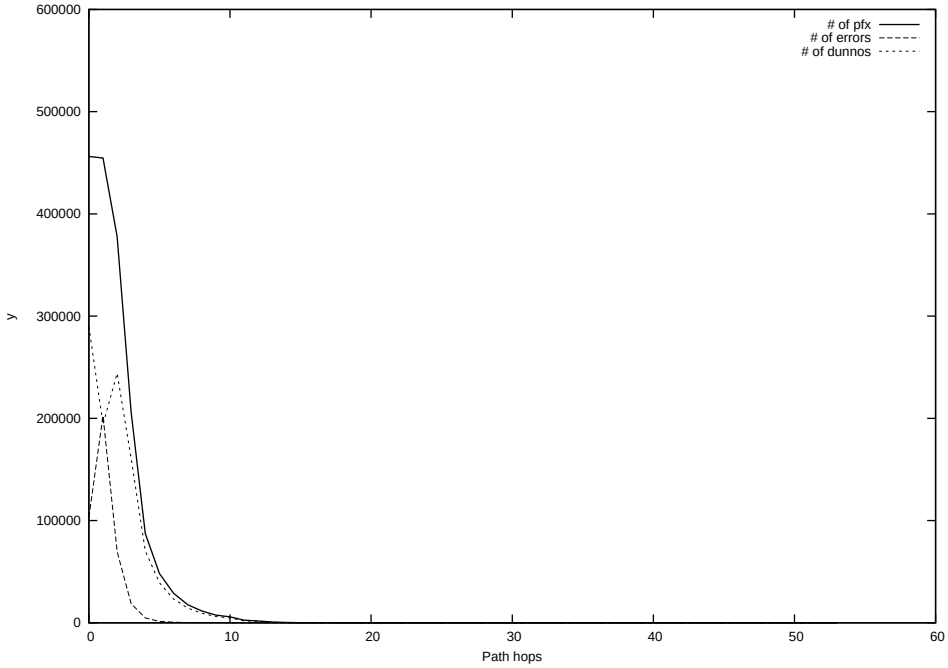
2013-06-12



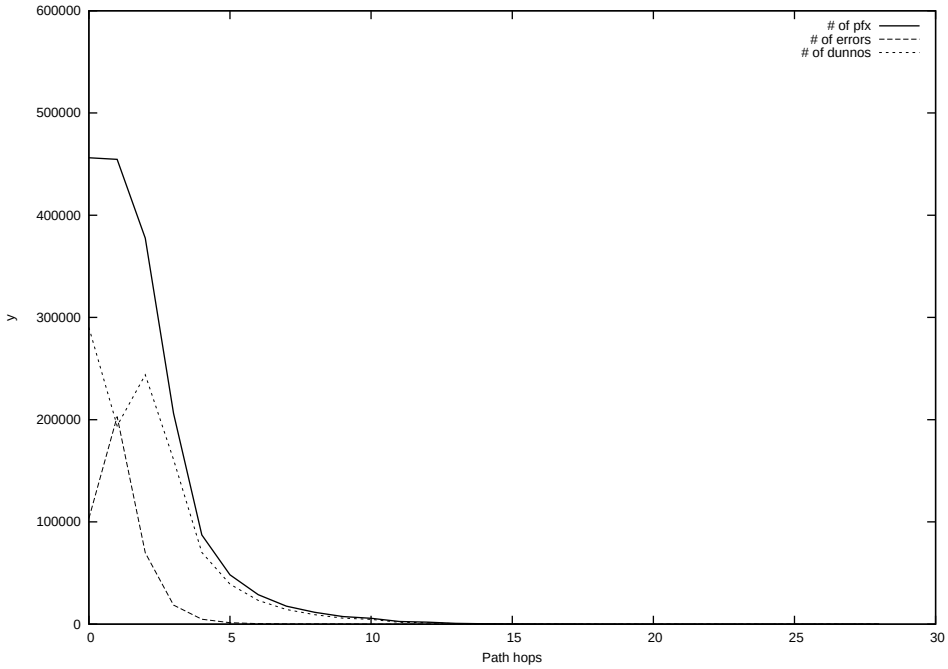
2013-06-13



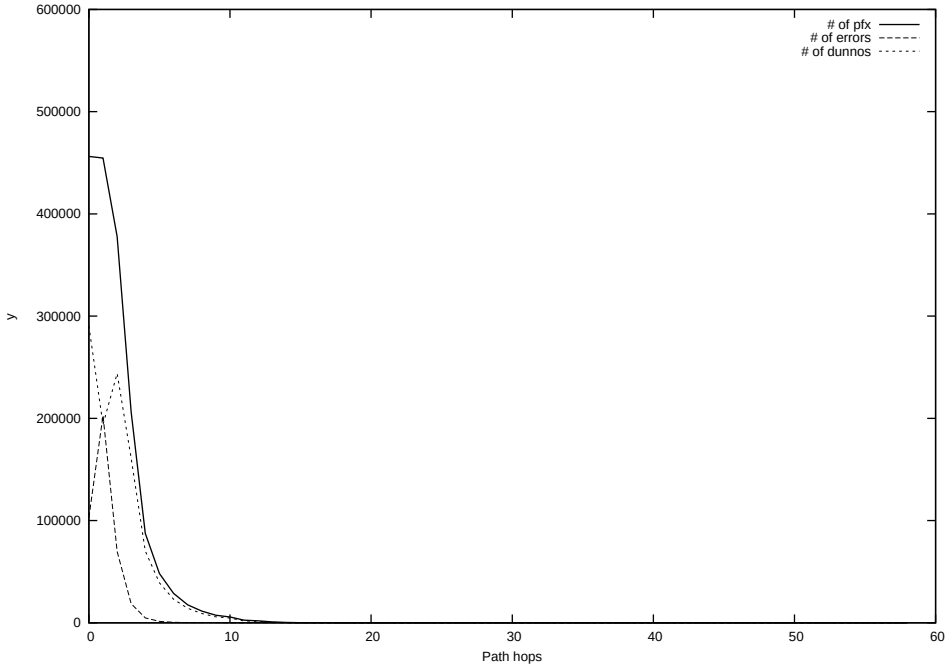
2013-06-14



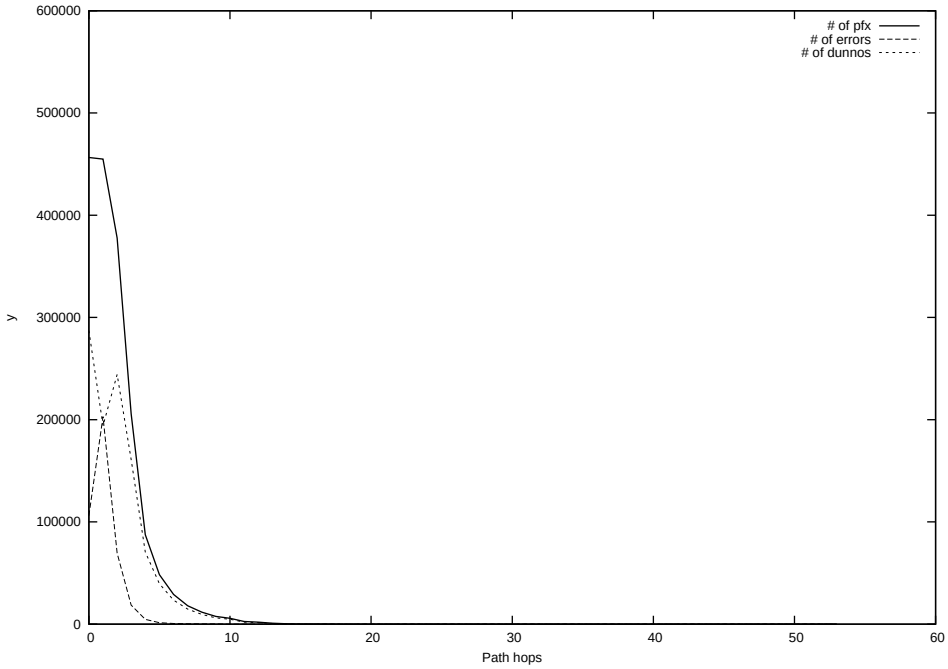
2013-06-15



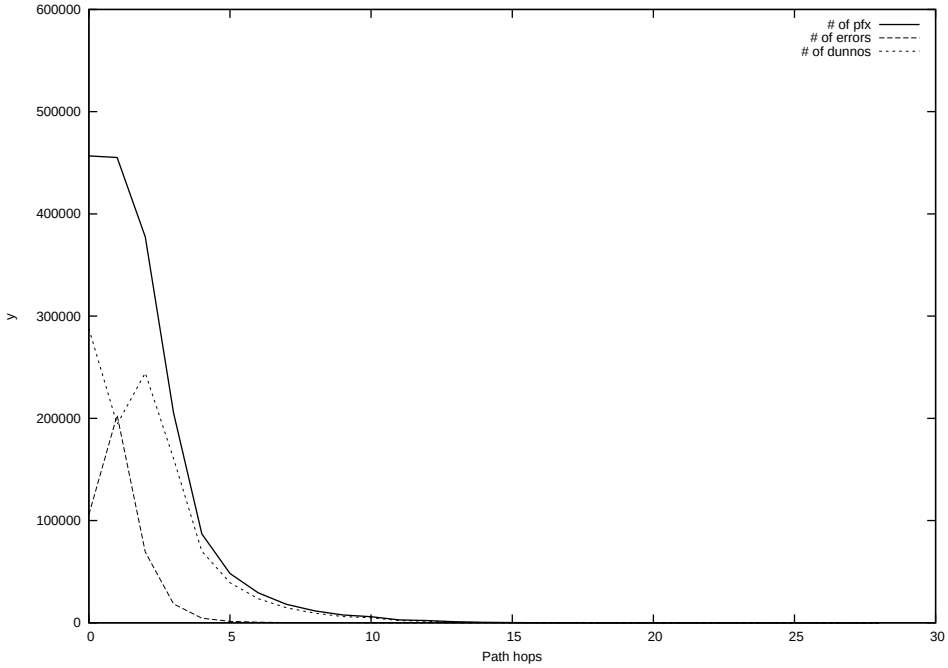
2013-06-16



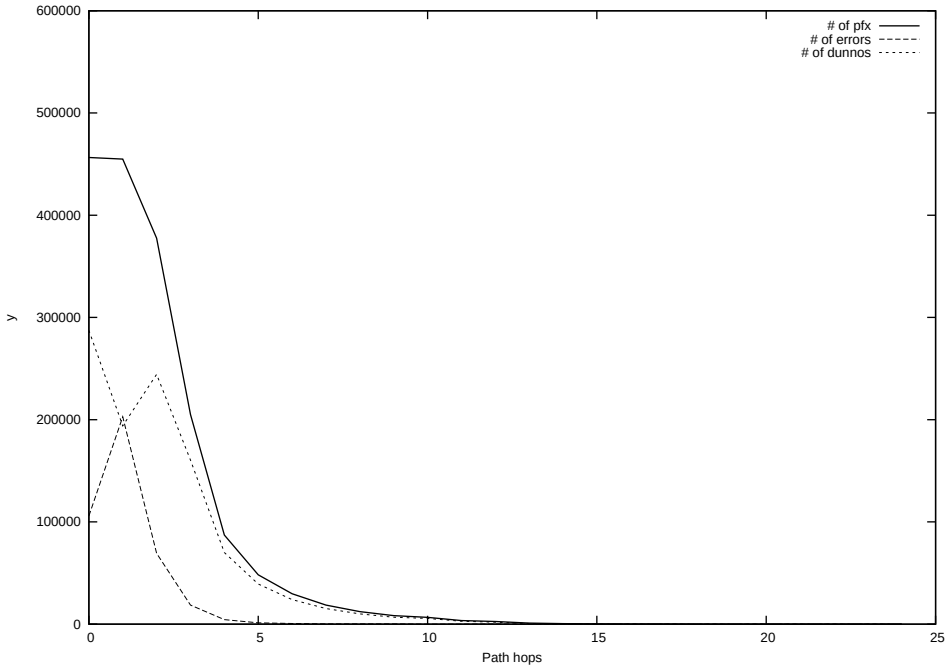
2013-06-17



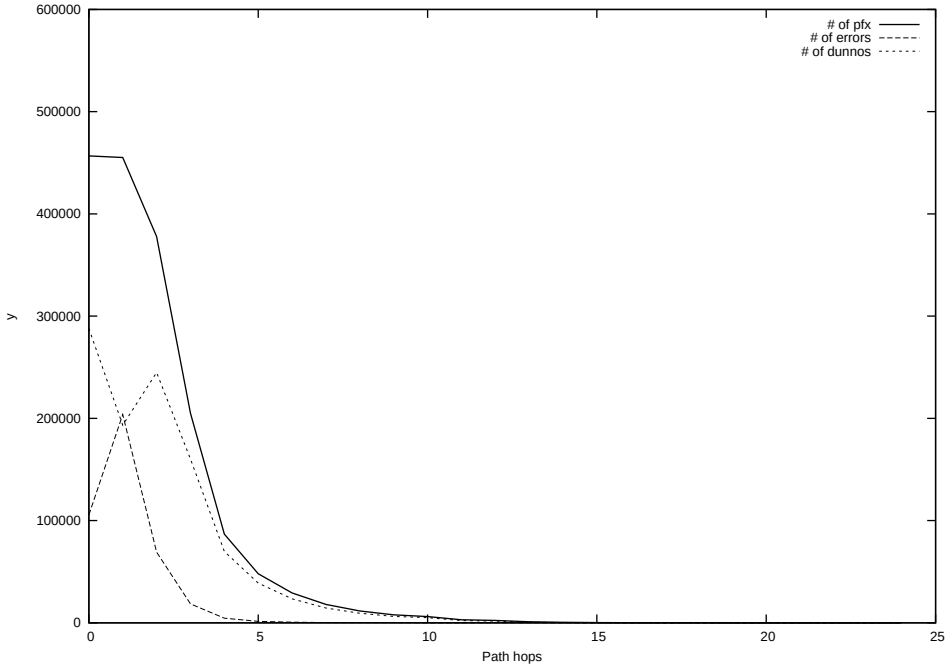
2013-06-18



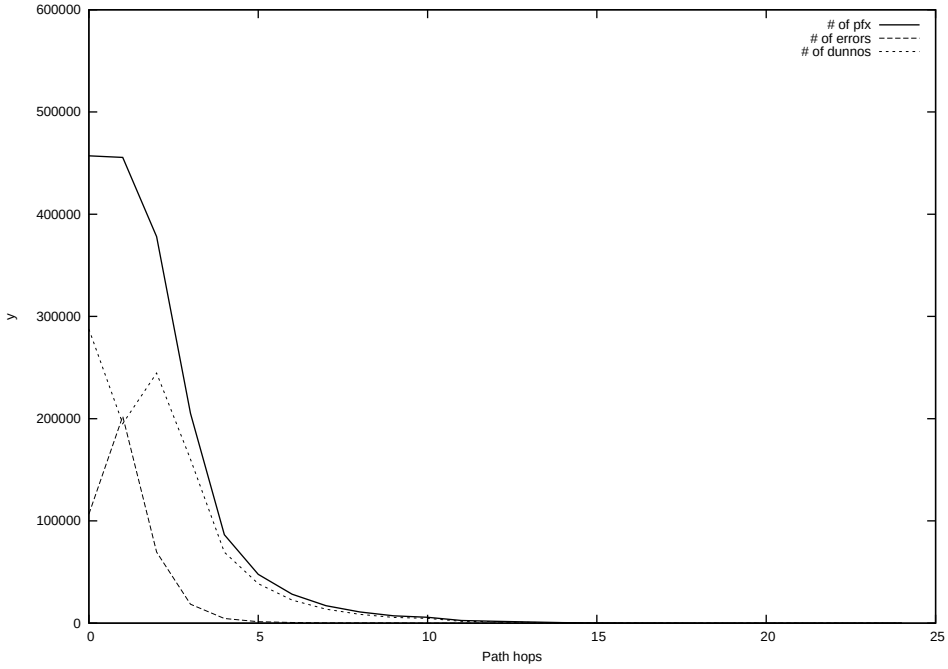
2013-06-19



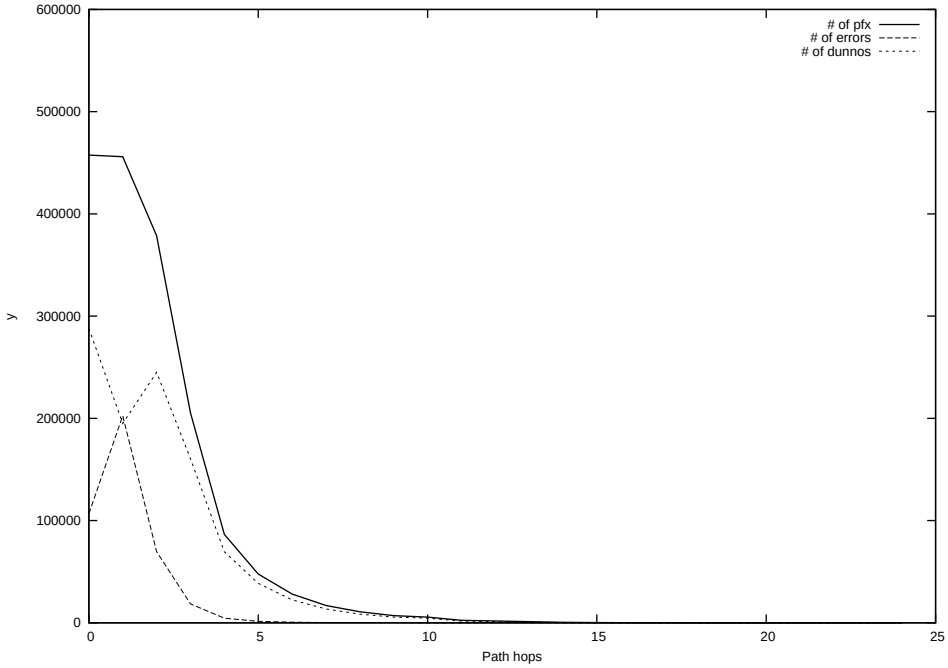
2013-06-20



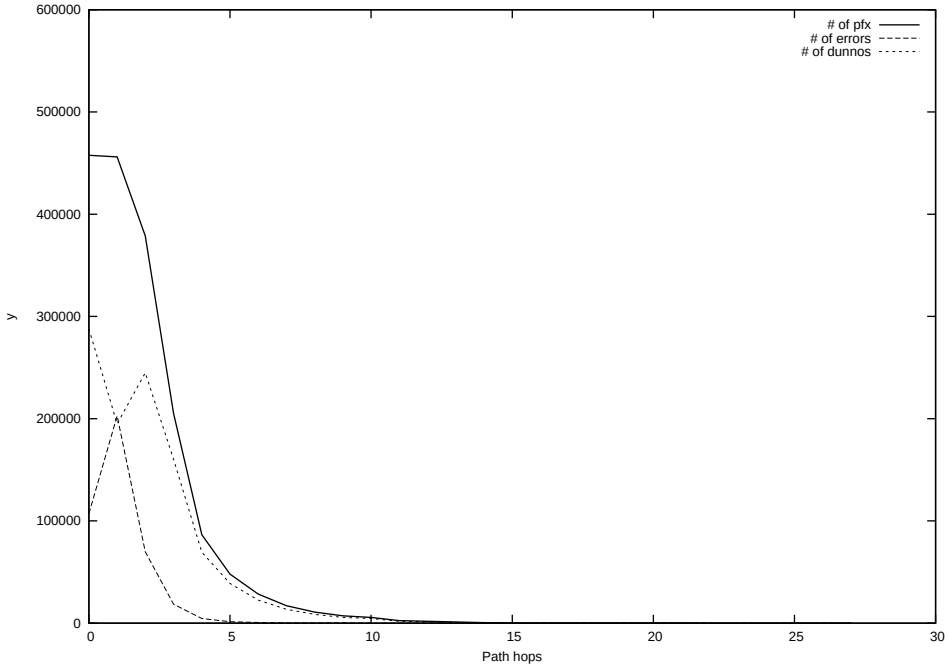
2013-06-21



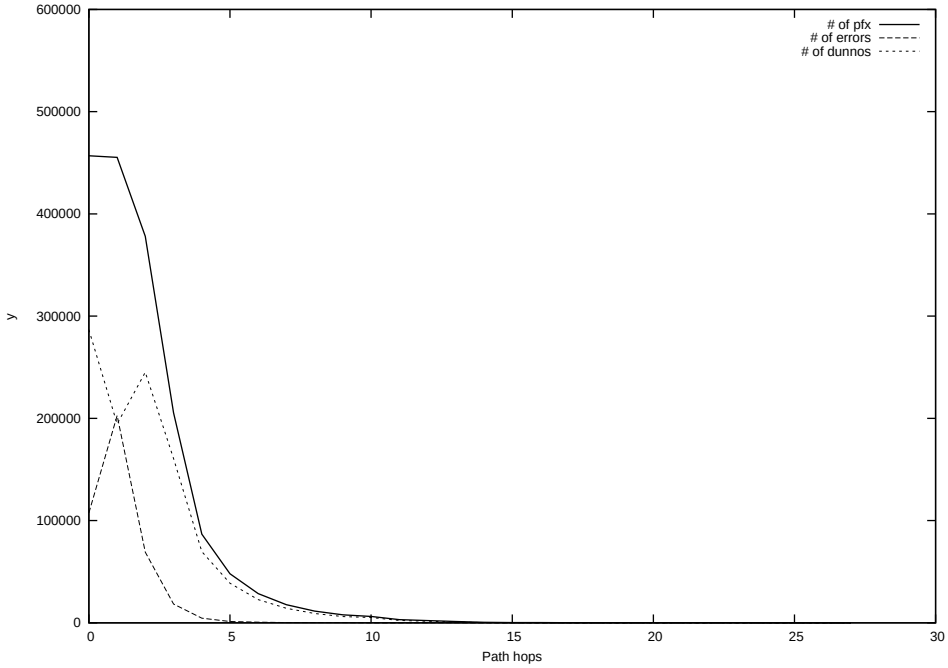
2013-06-22



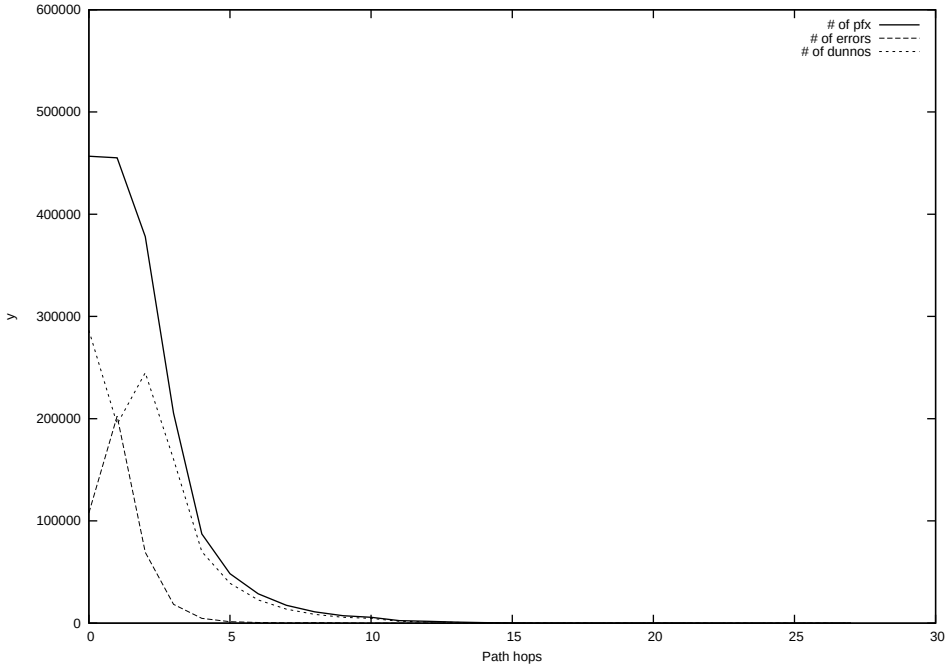
2013-06-23



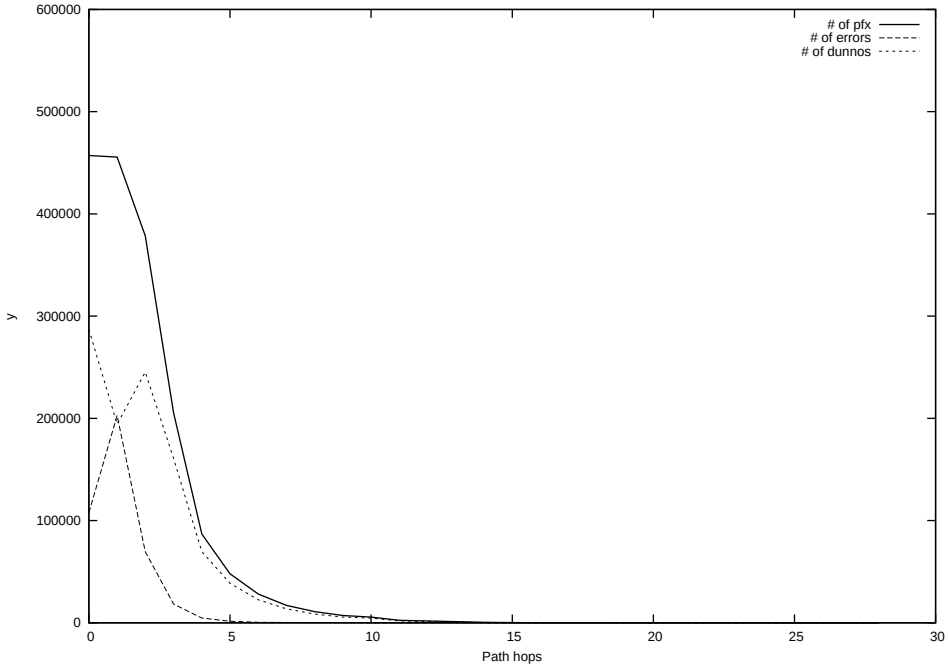
2013-06-24



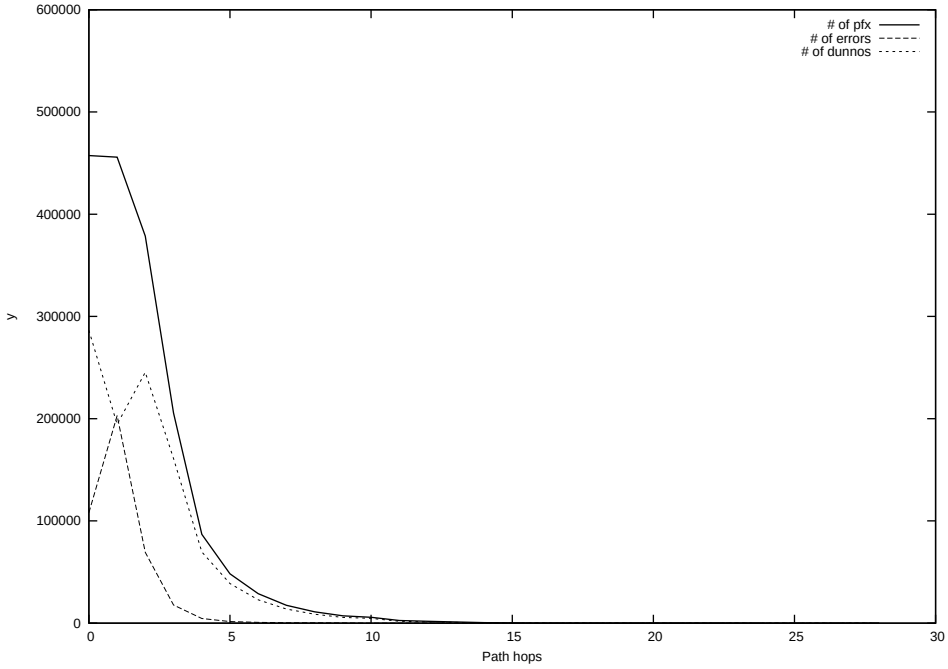
2013-06-25



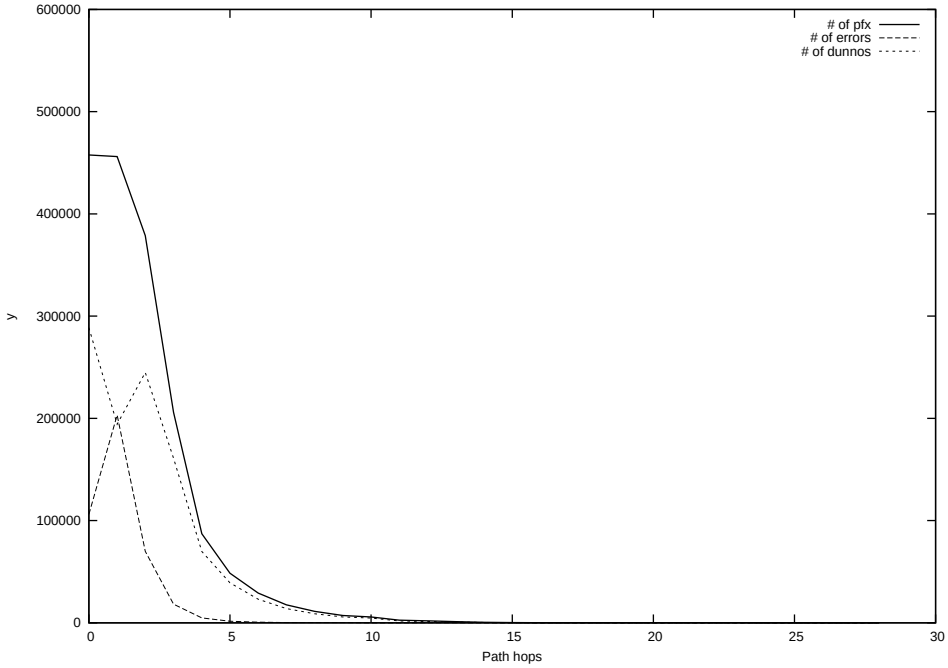
2013-06-26



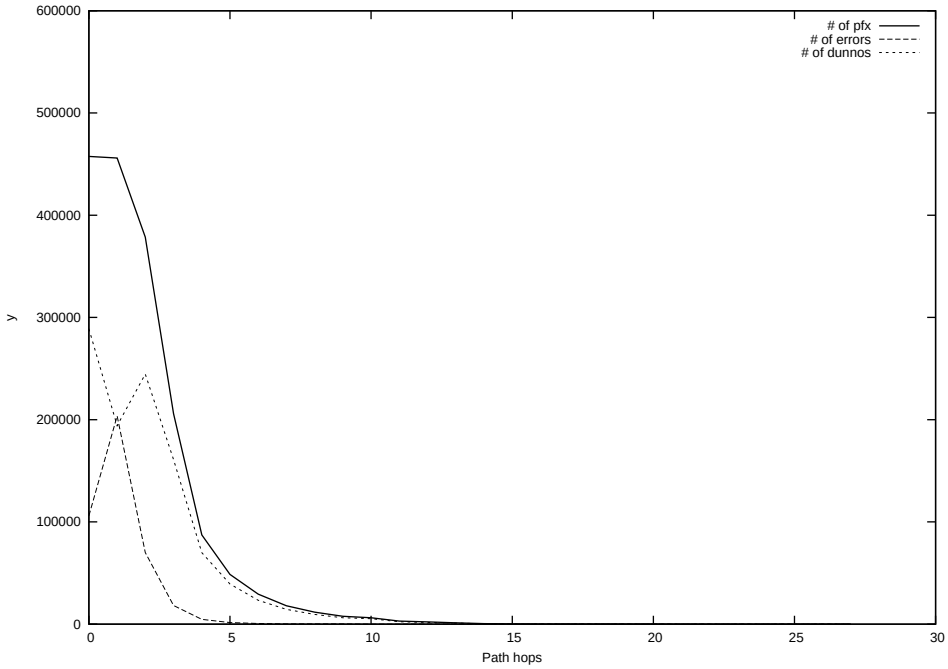
2013-06-27



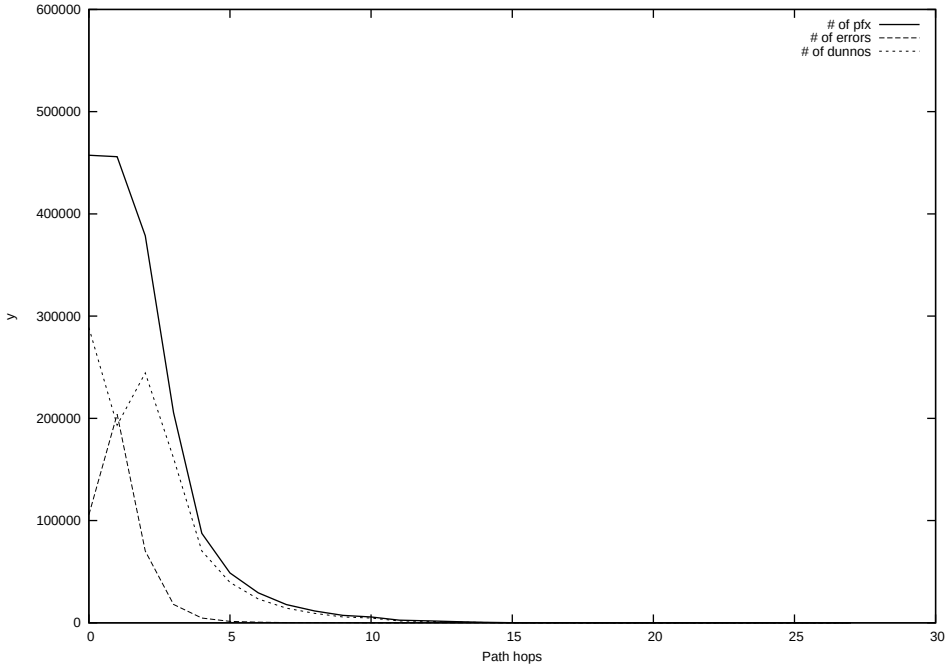
2013-06-28



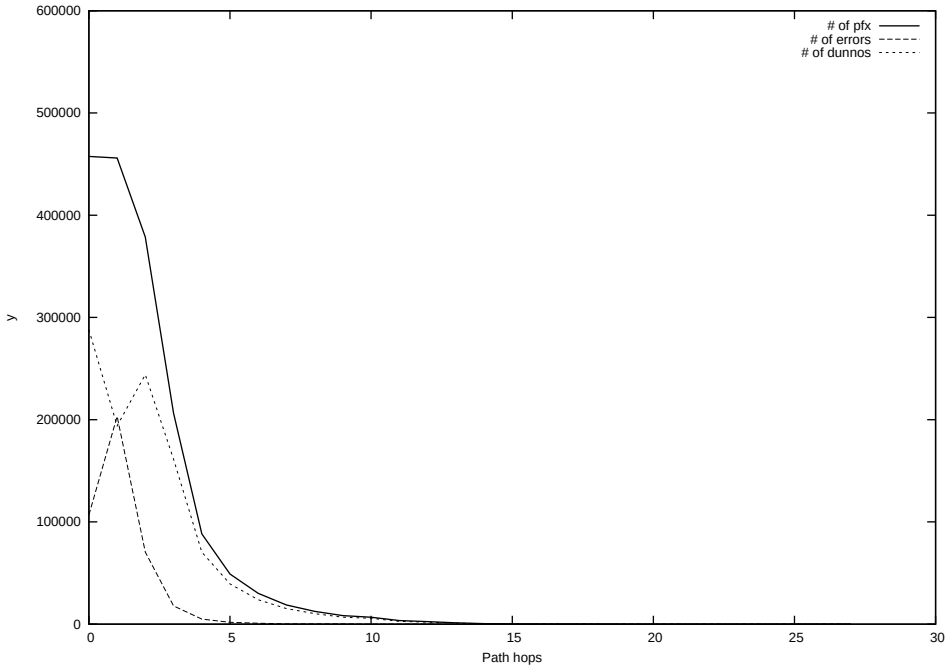
2013-06-29



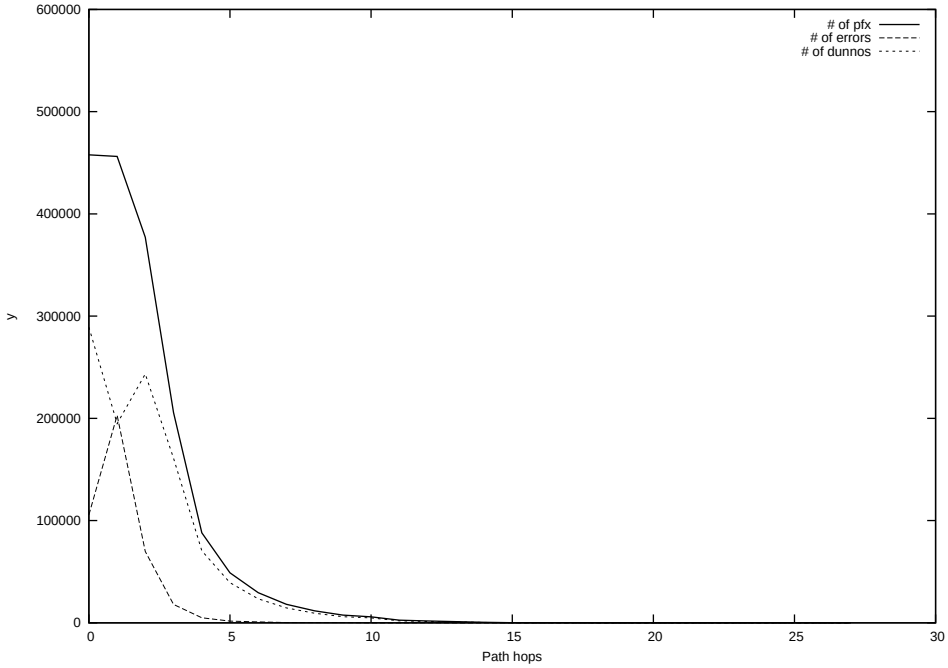
2013-06-30



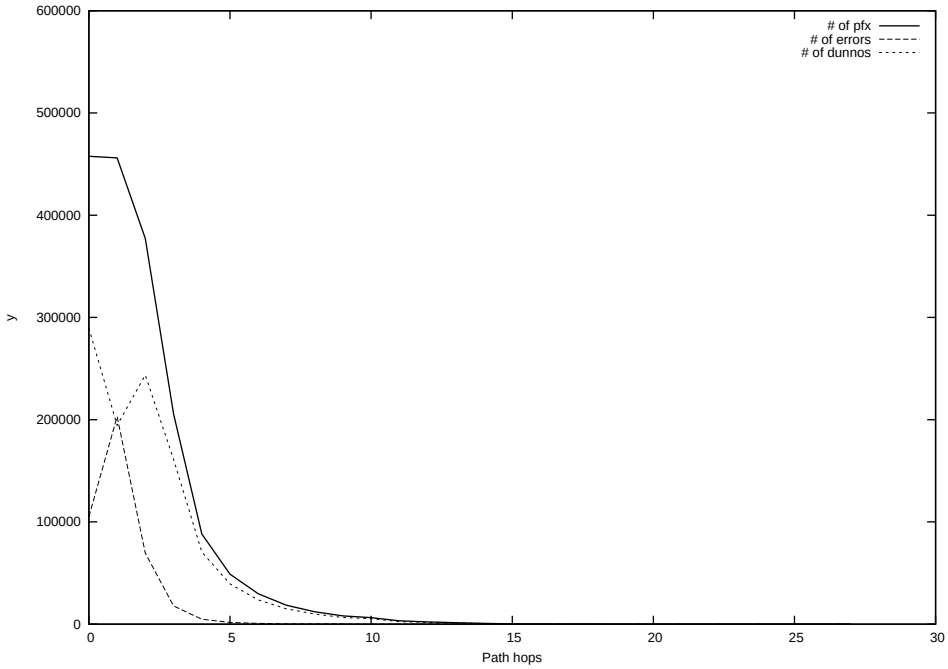
2013-07-01



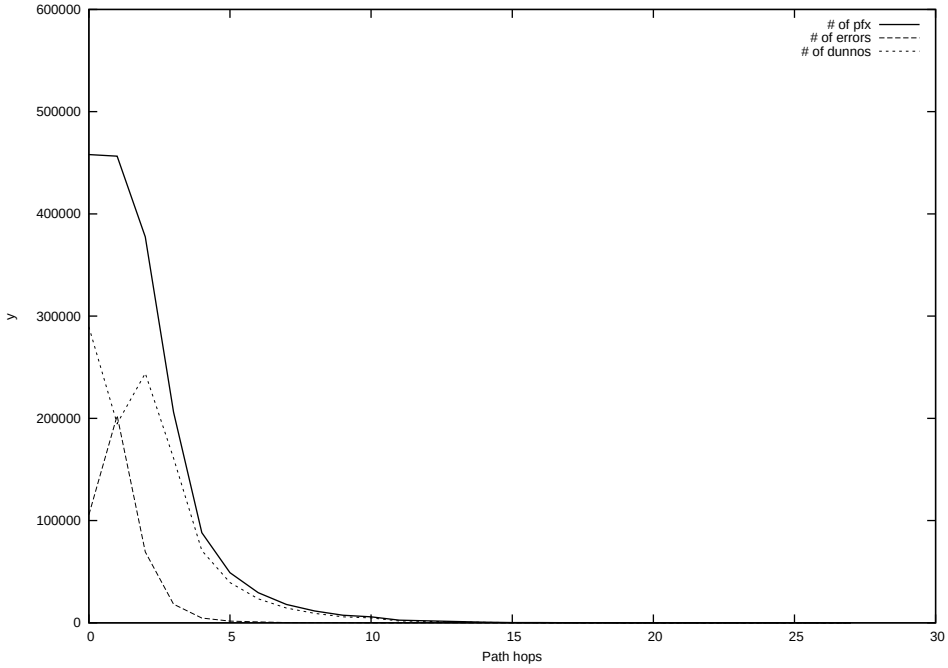
2013-07-02



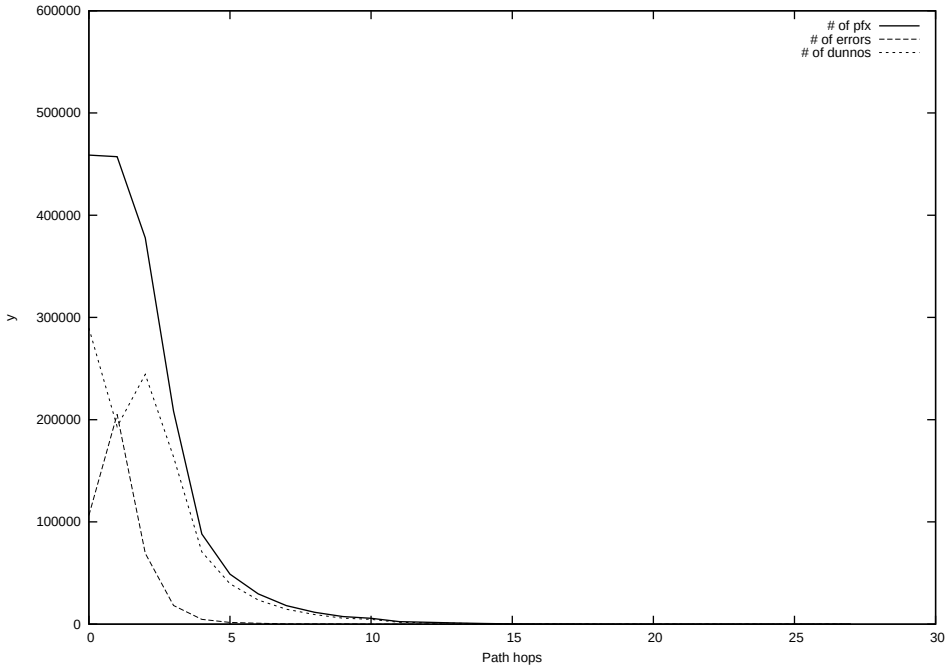
2013-07-03



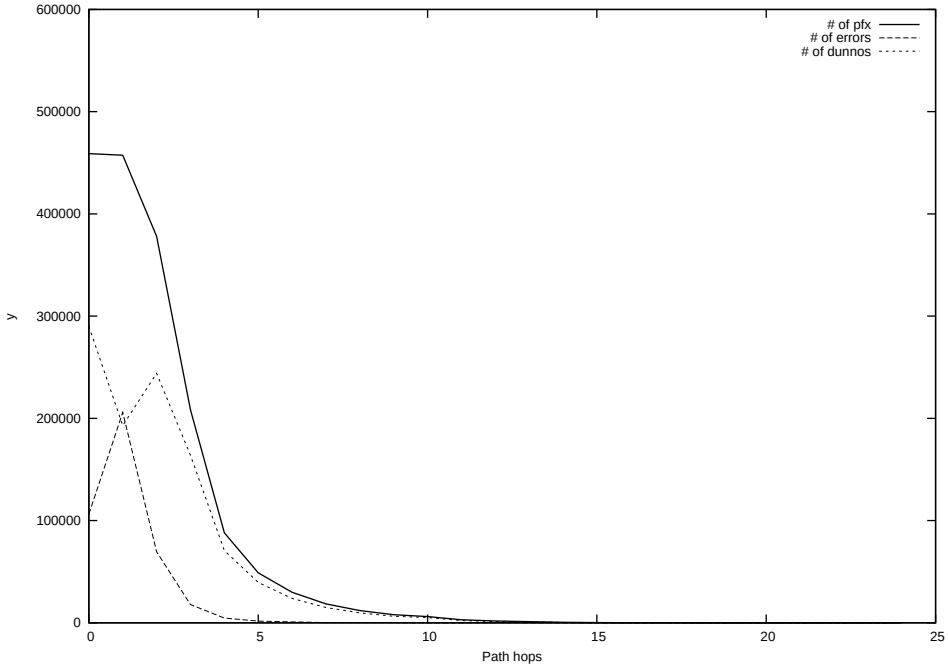
2013-07-04



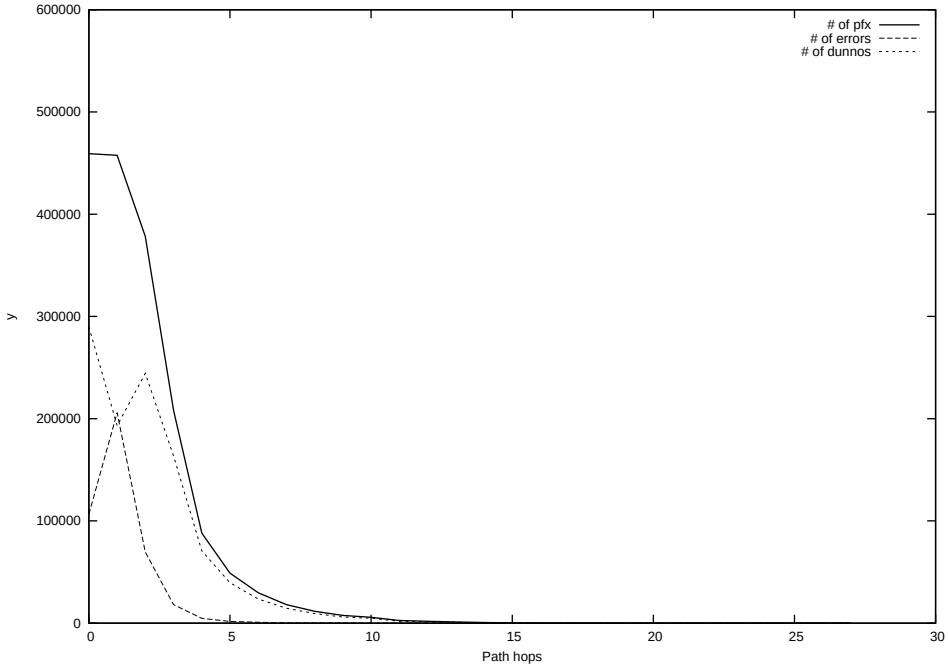
2013-07-05



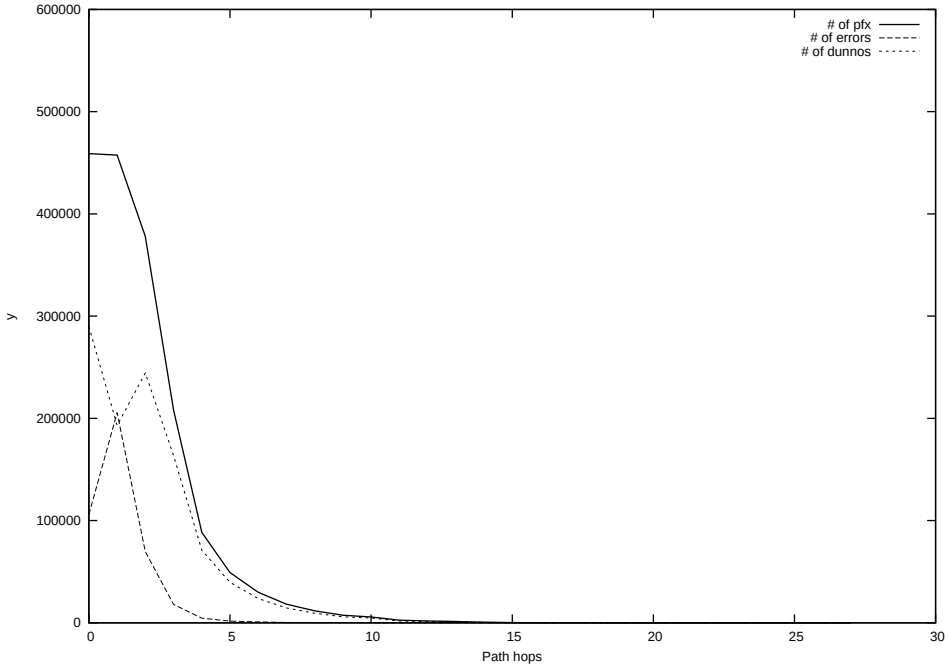
2013-07-06



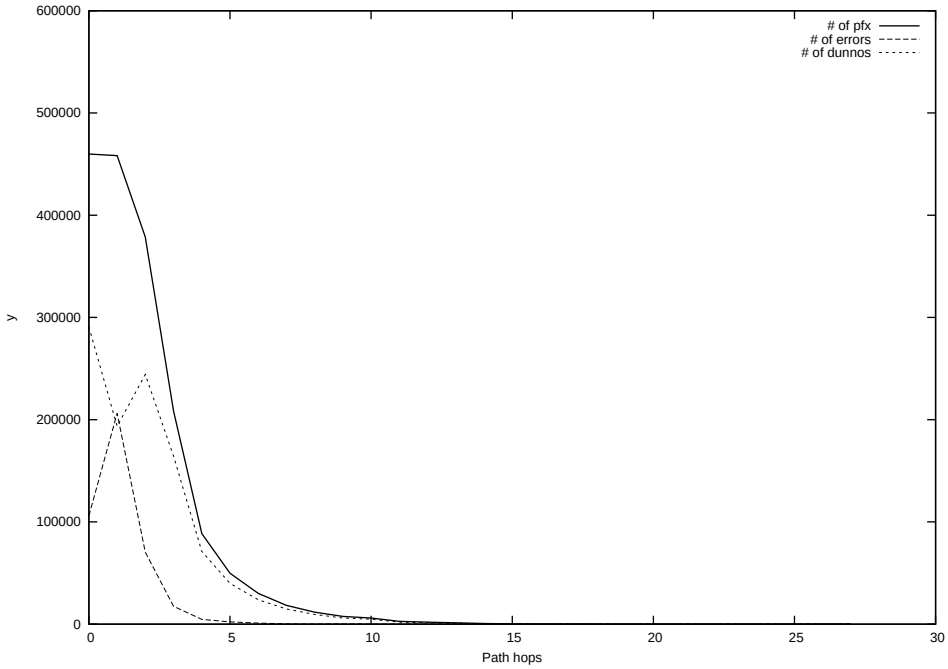
2013-07-07



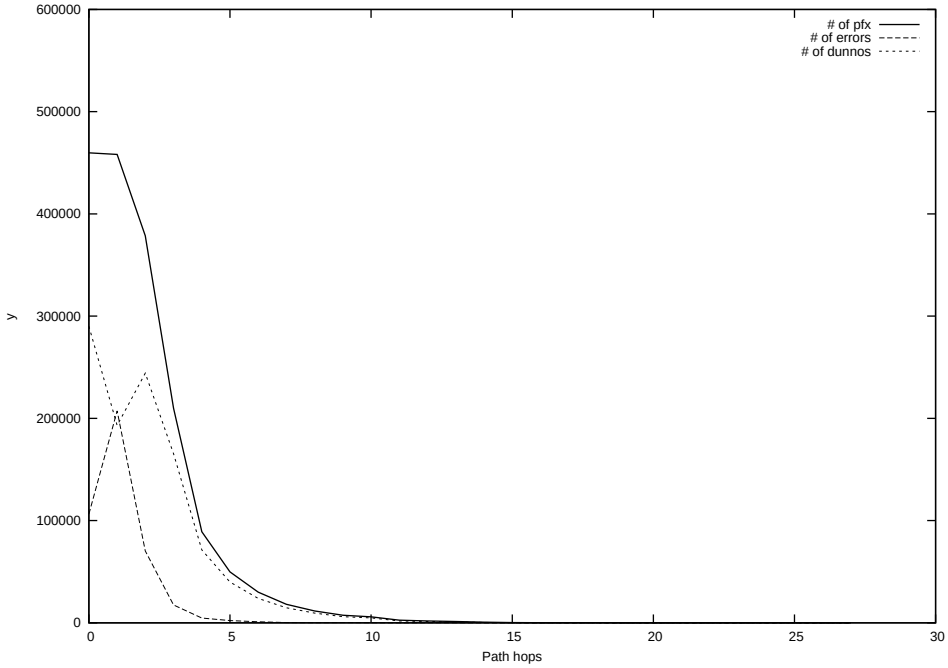
2013-07-08



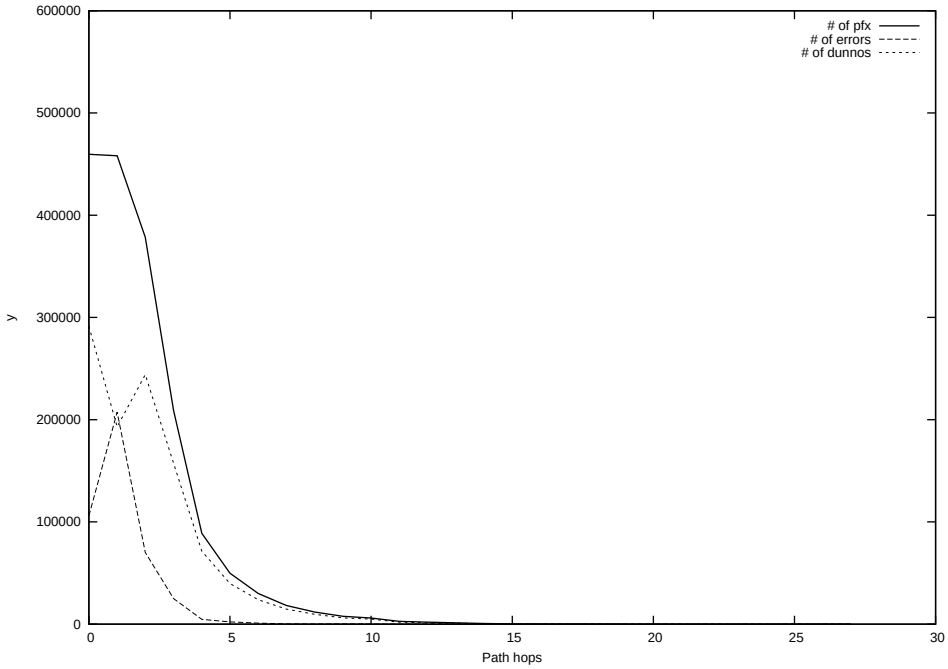
2013-07-09



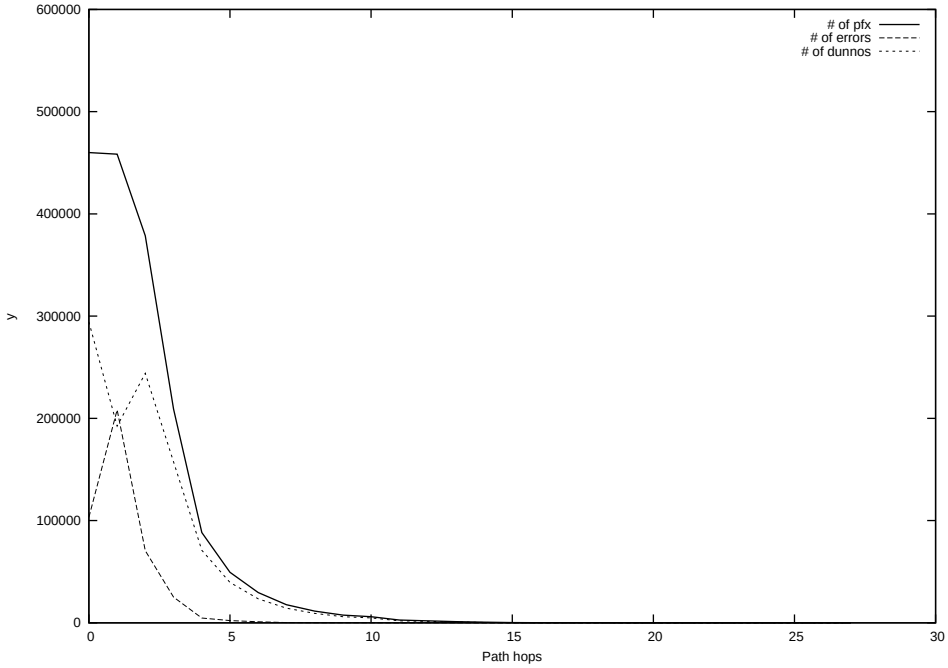
2013-07-10



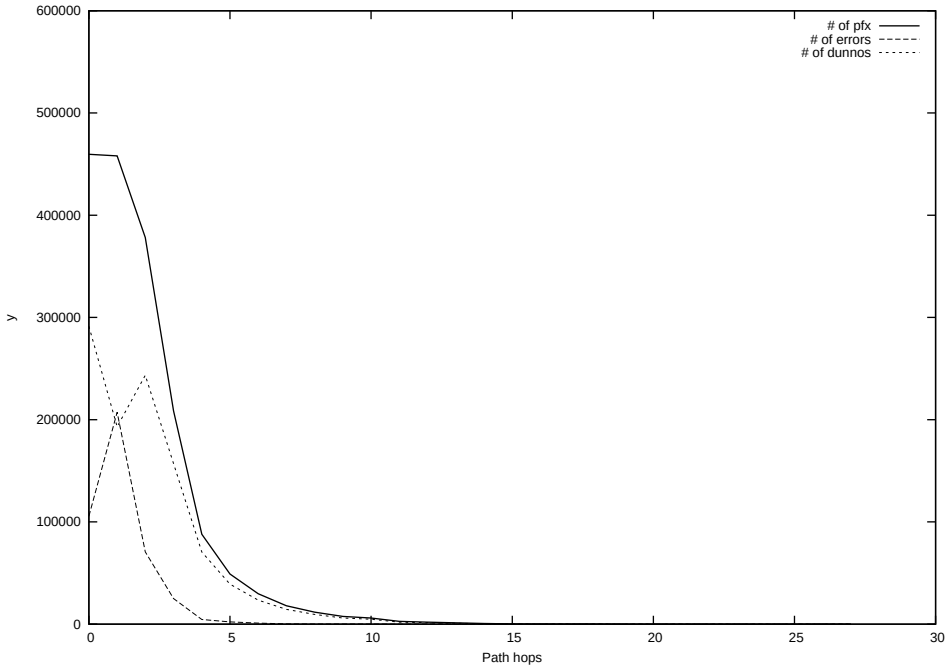
2013-07-11



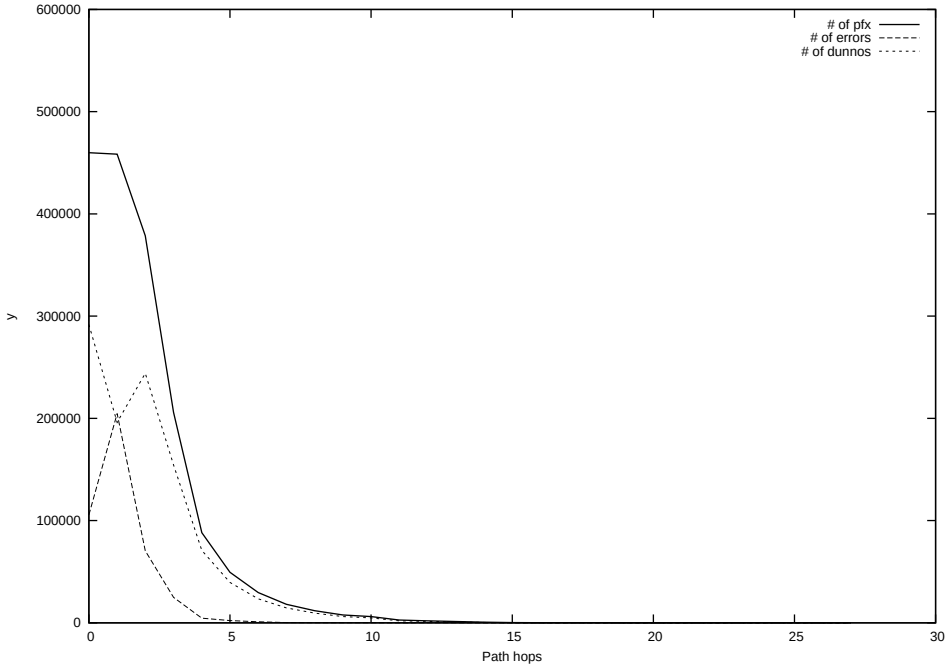
2013-07-12



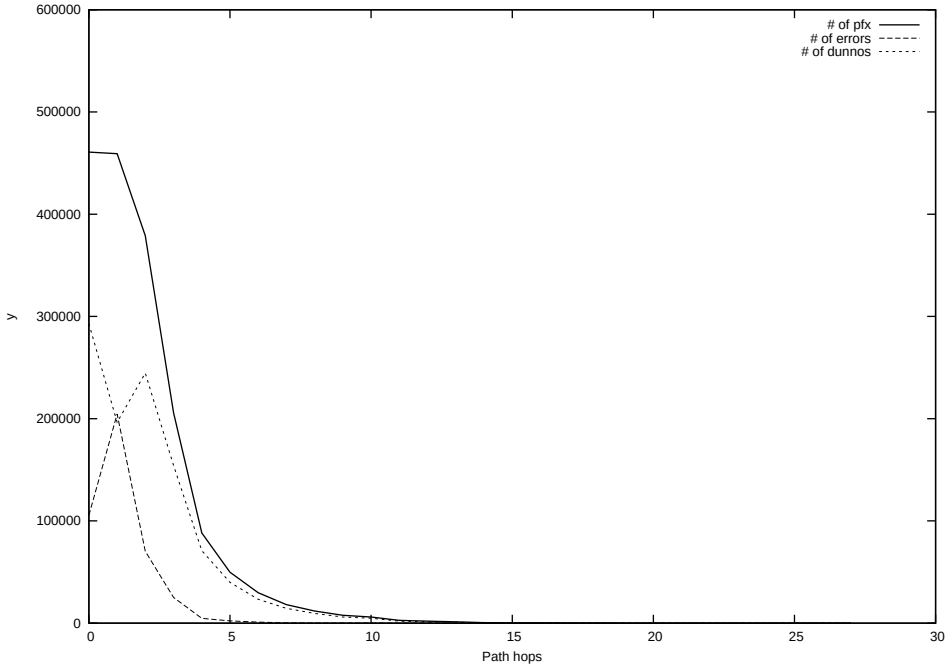
2013-07-13



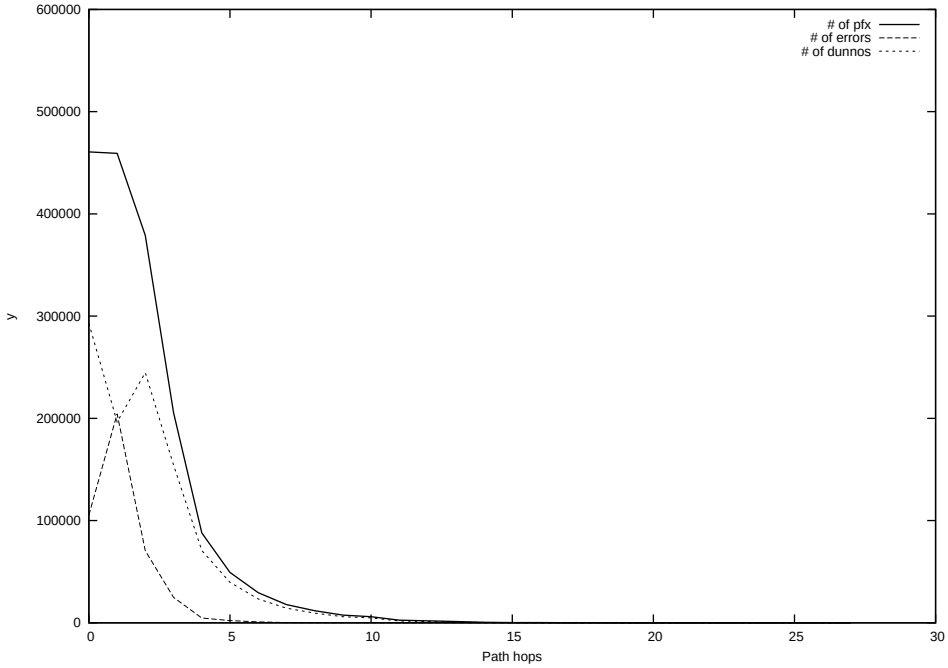
2013-07-14



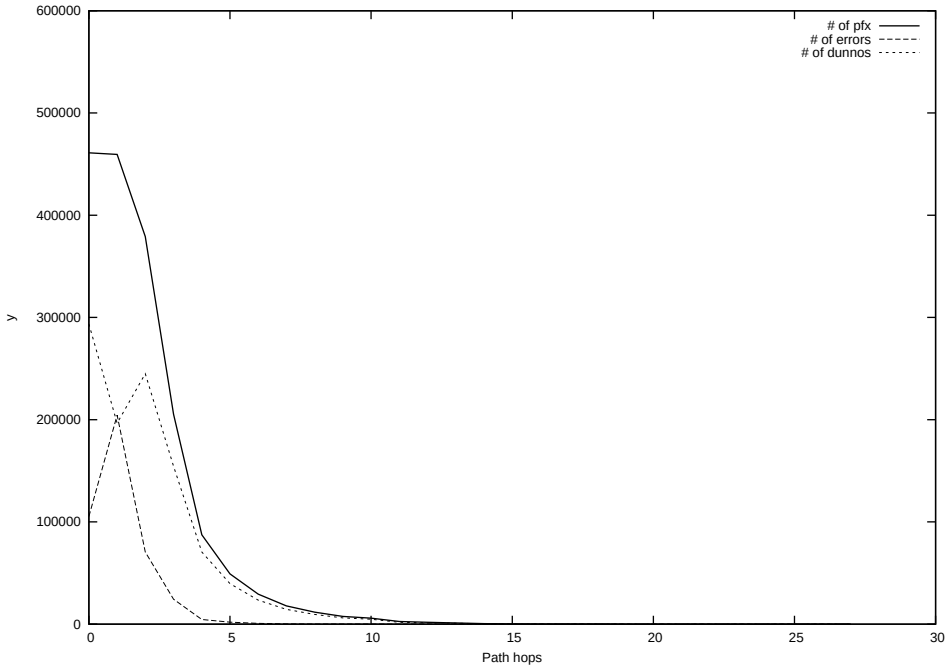
2013-07-15



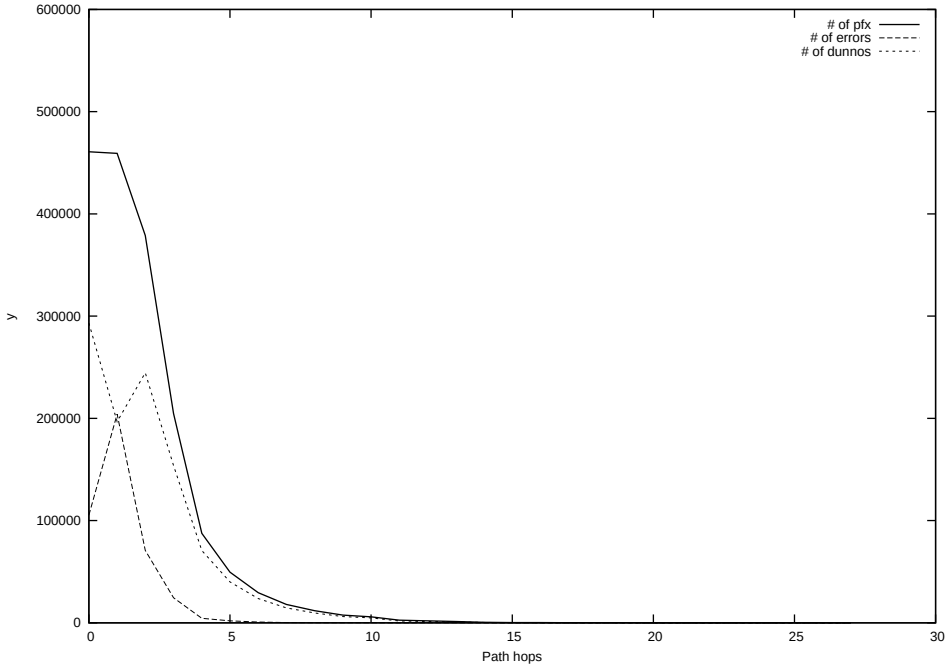
2013-07-16



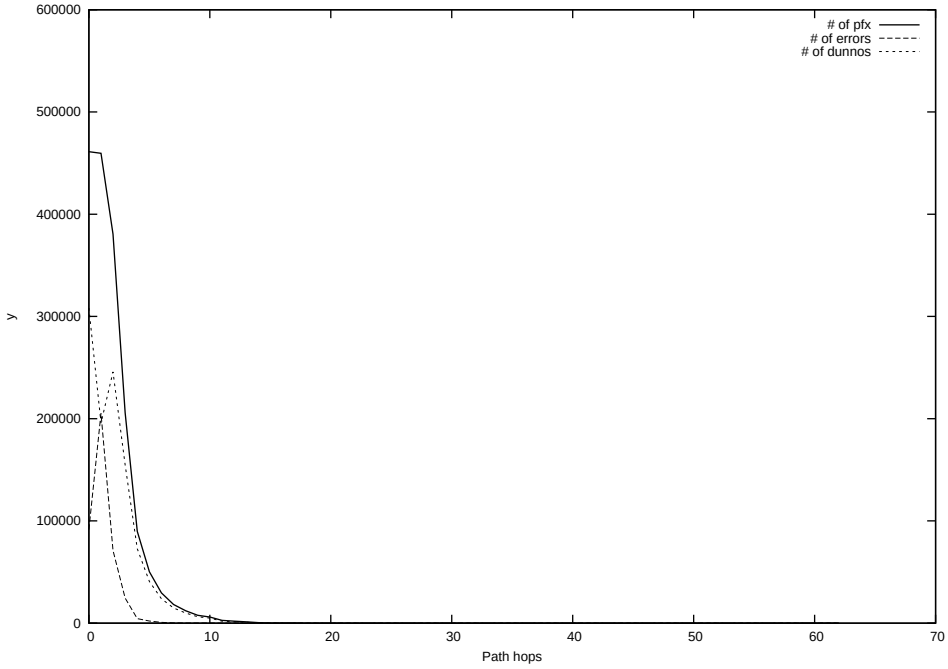
2013-07-17



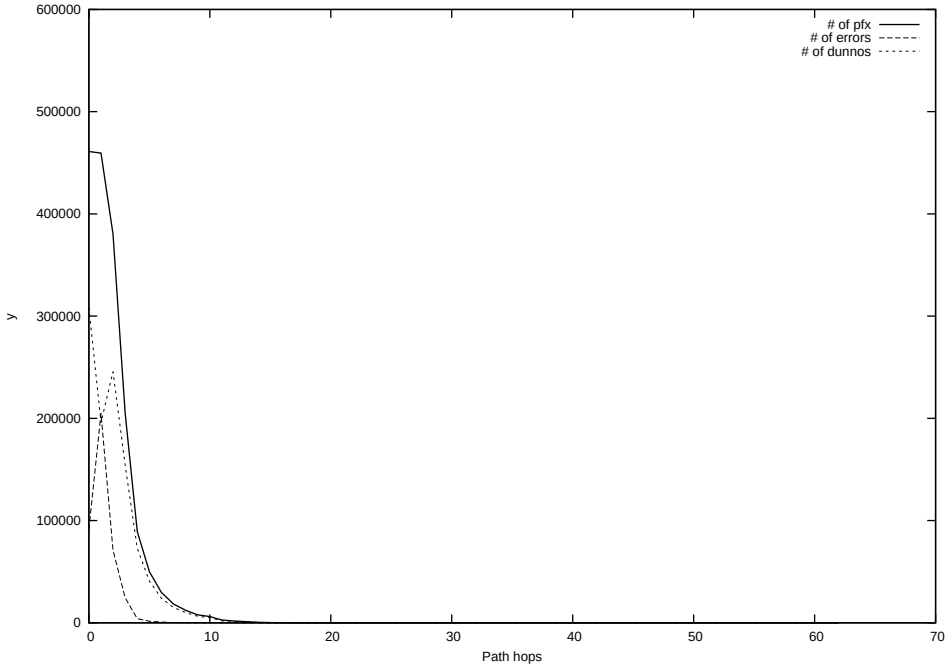
2013-07-18



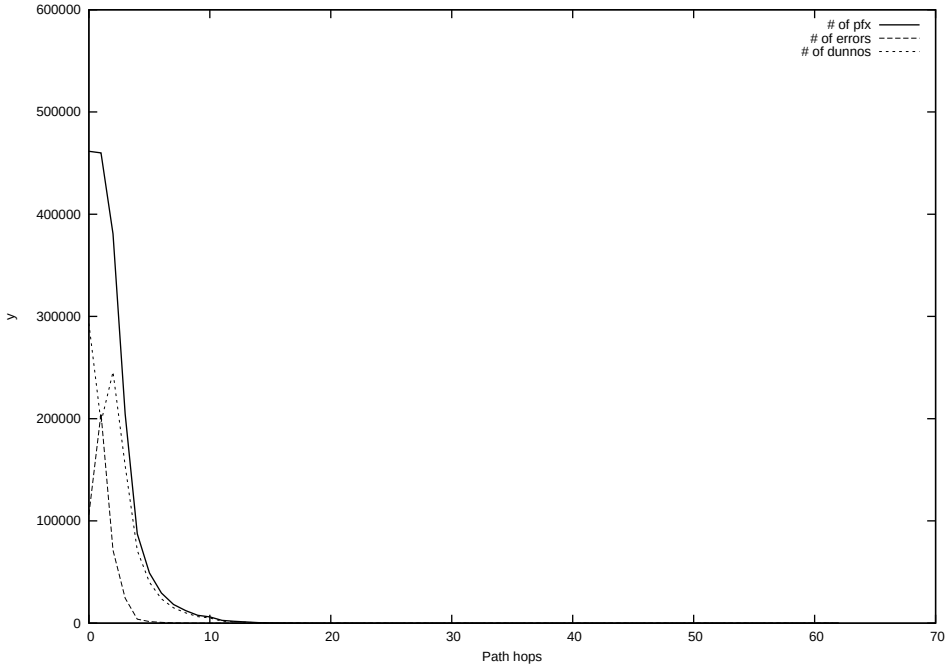
2013-07-19



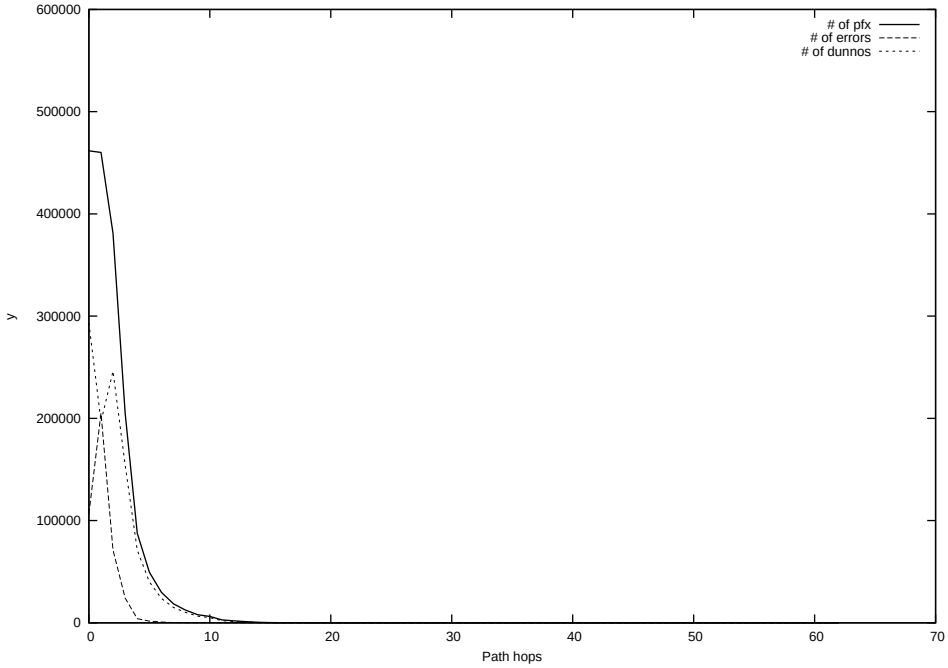
2013-07-20



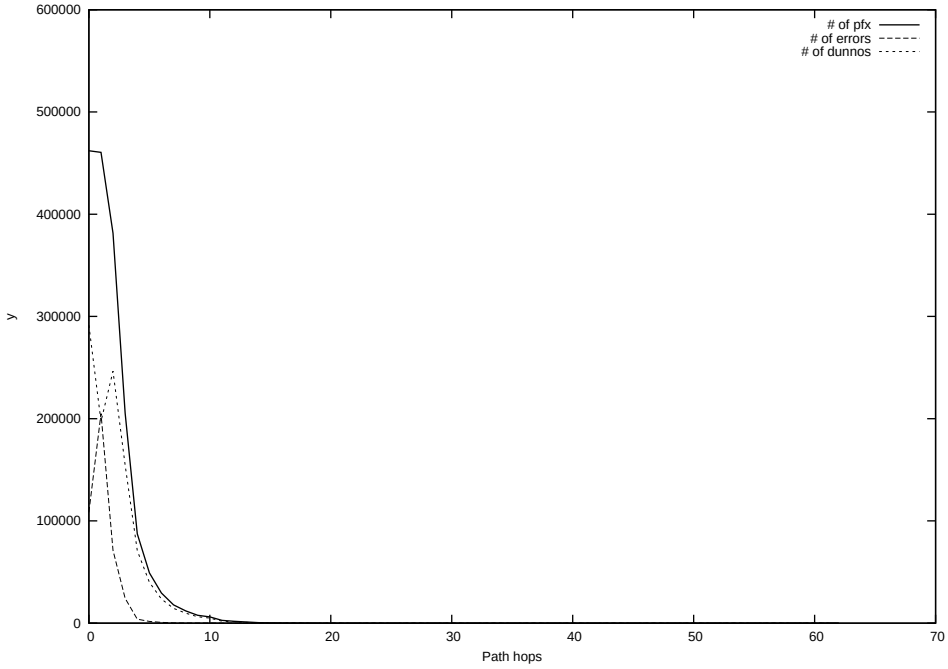
2013-07-21



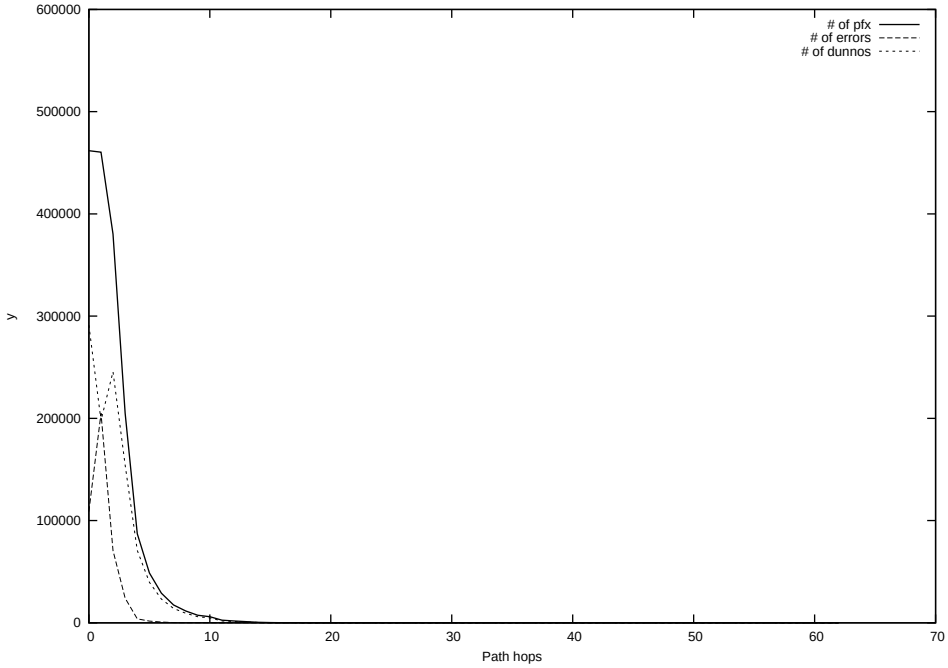
2013-07-22



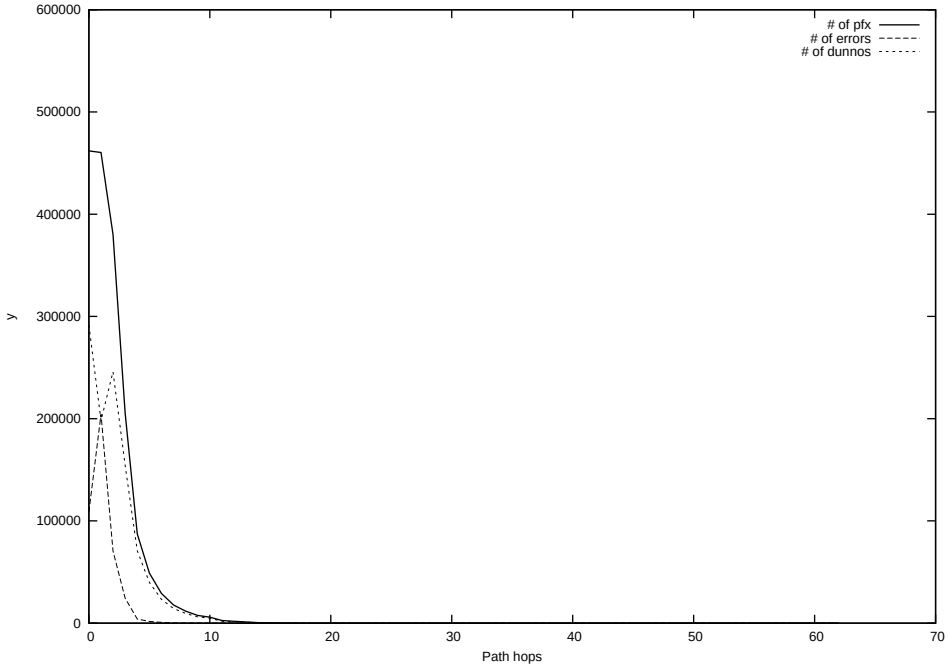
2013-07-23



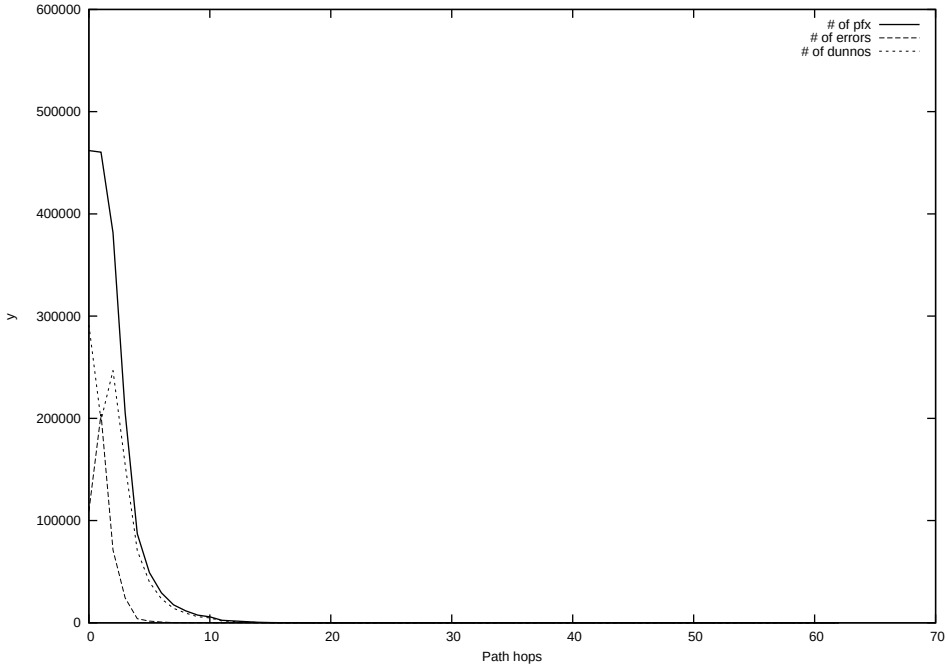
2013-07-24



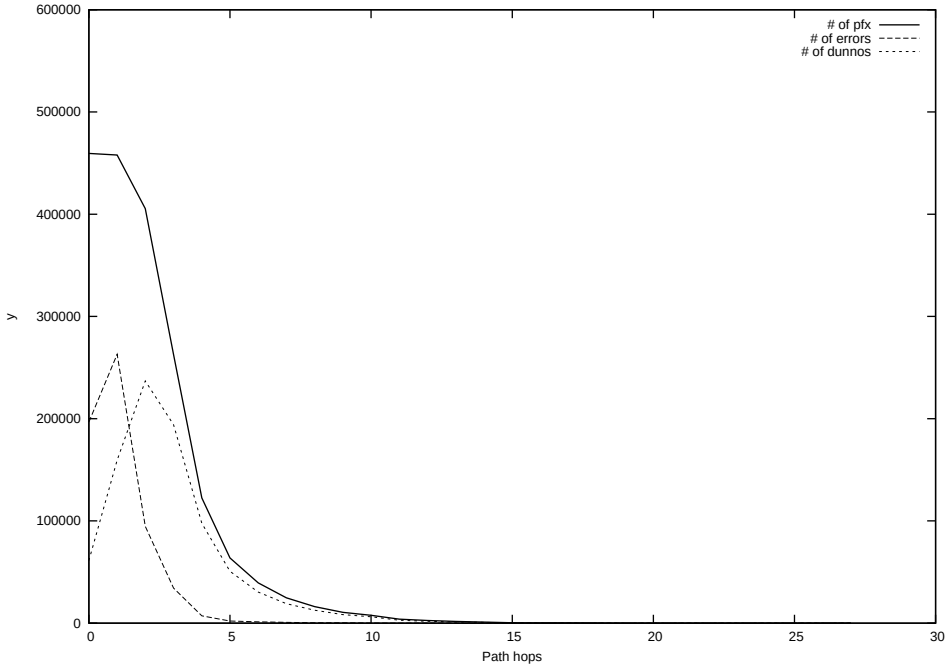
2013-07-25



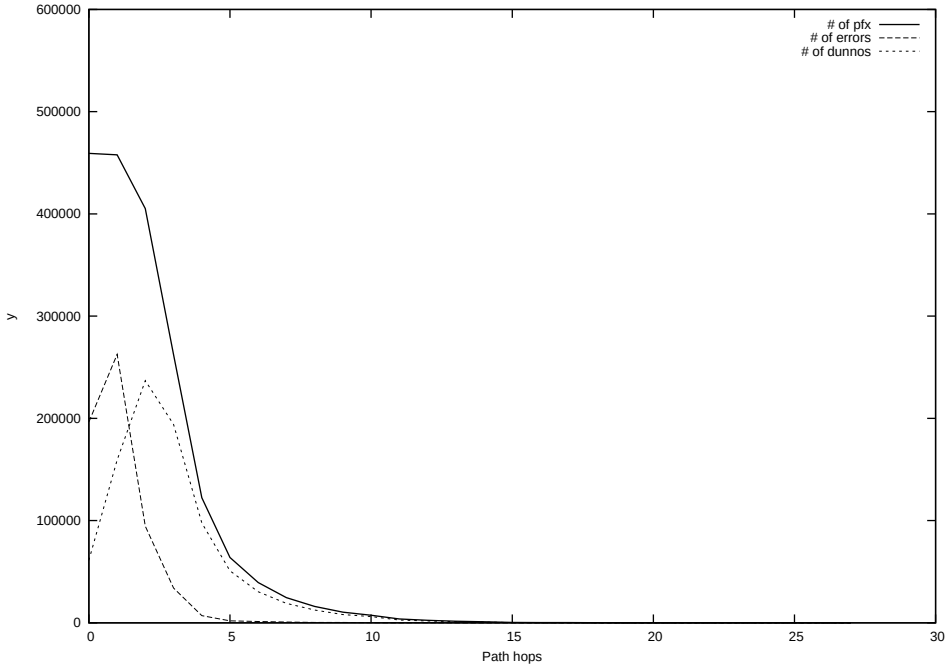
2013-07-26



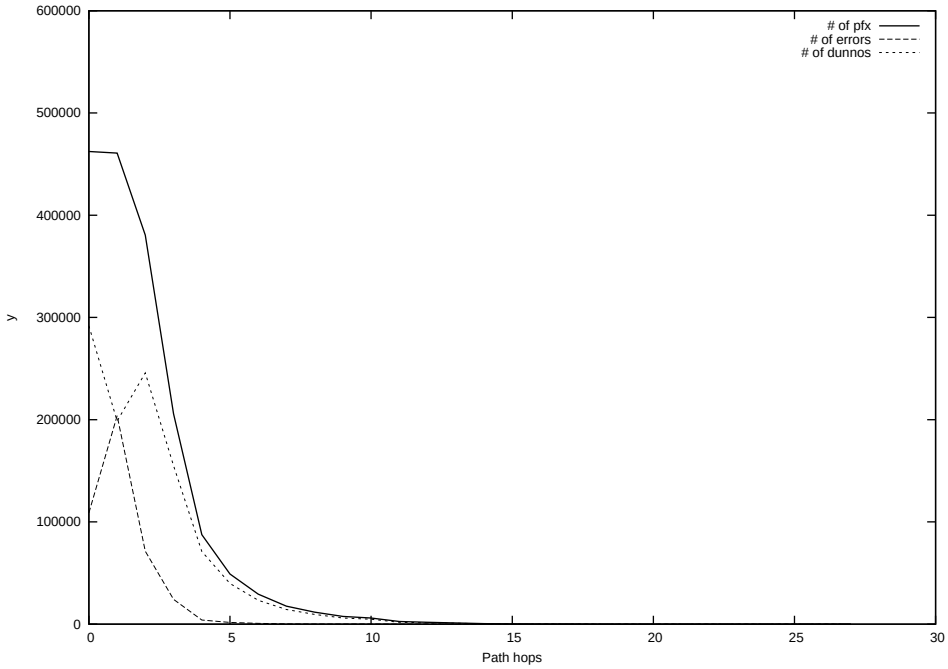
2013-07-27



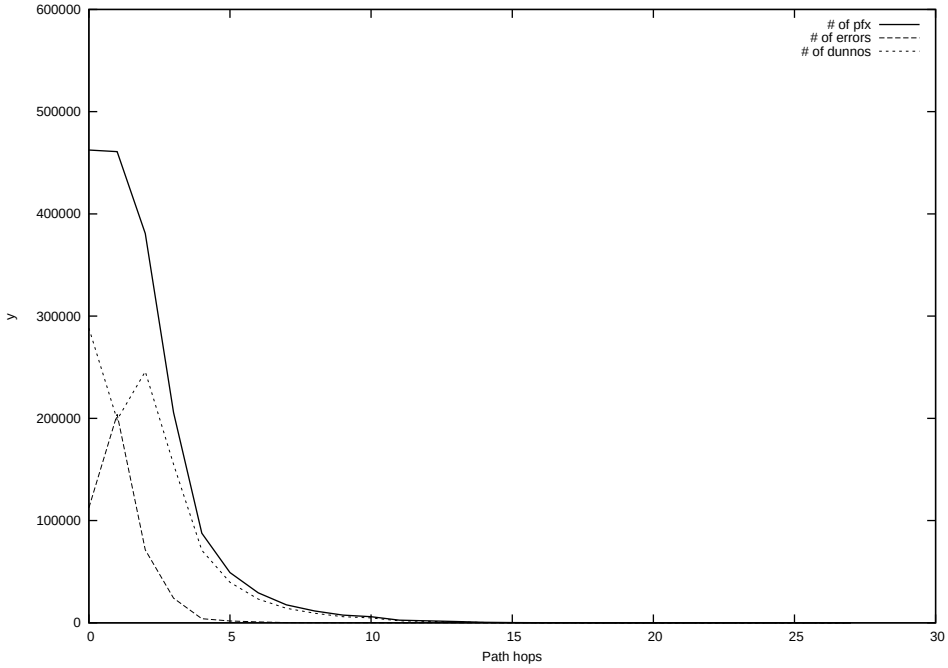
2013-07-28



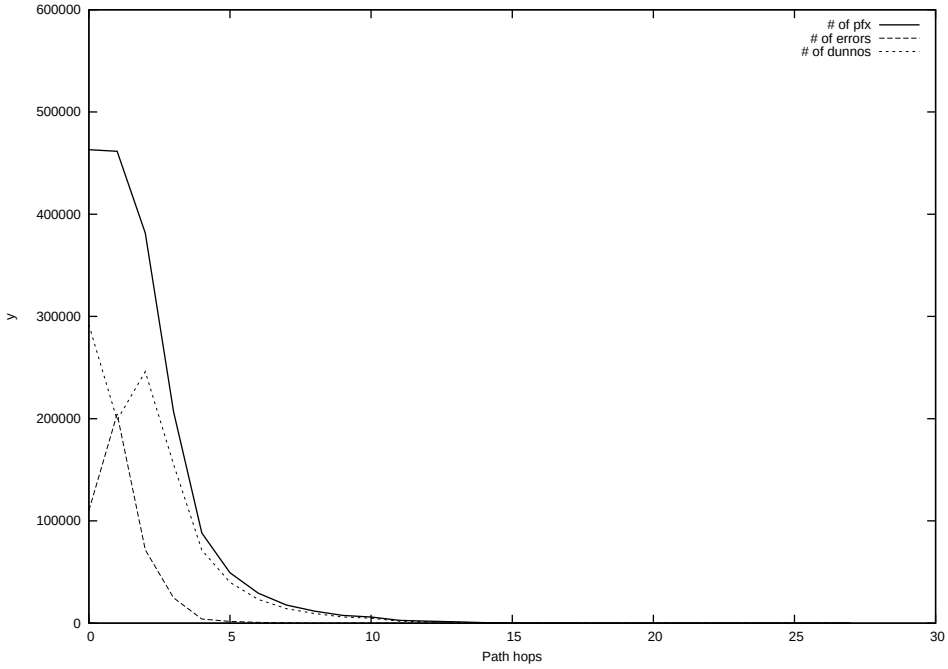
2013-07-29



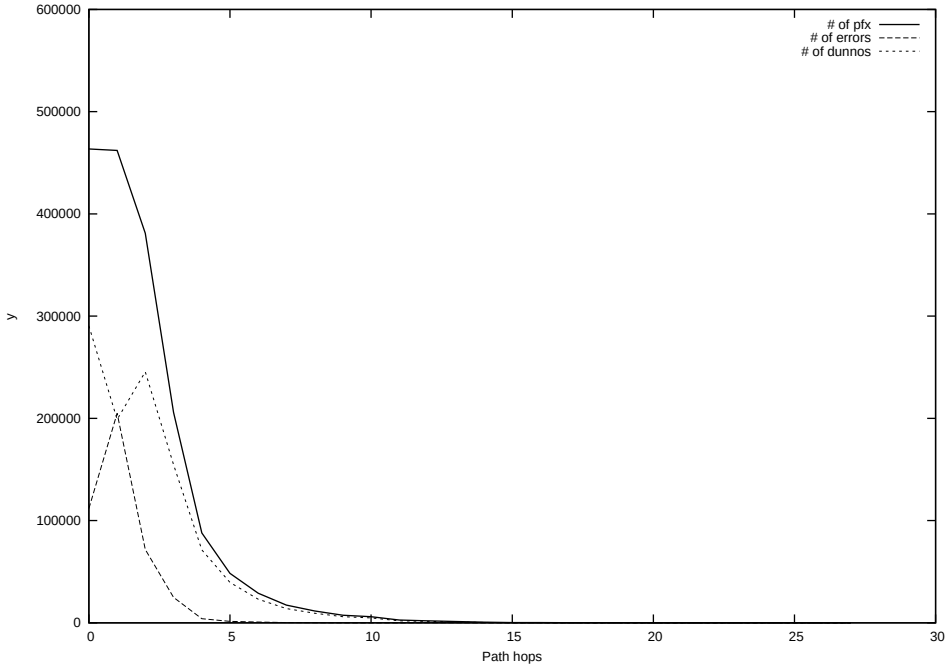
2013-07-30



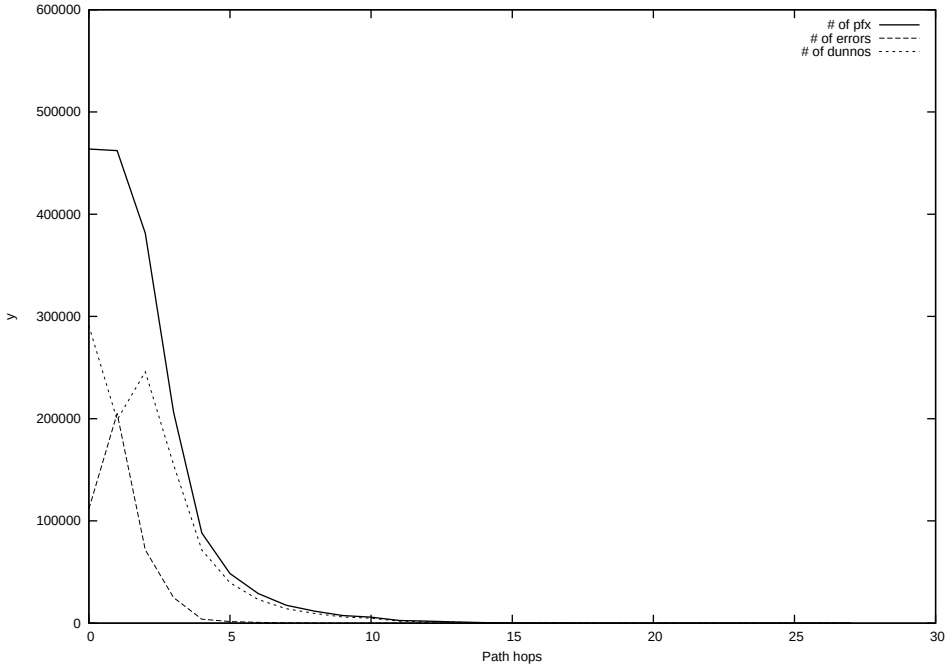
2013-07-31



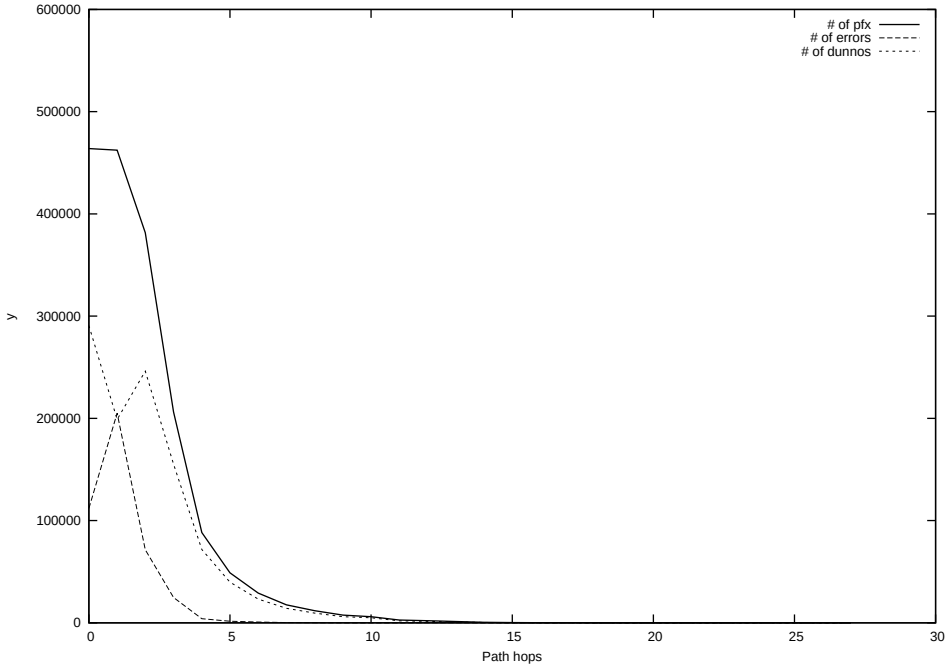
2013-08-01



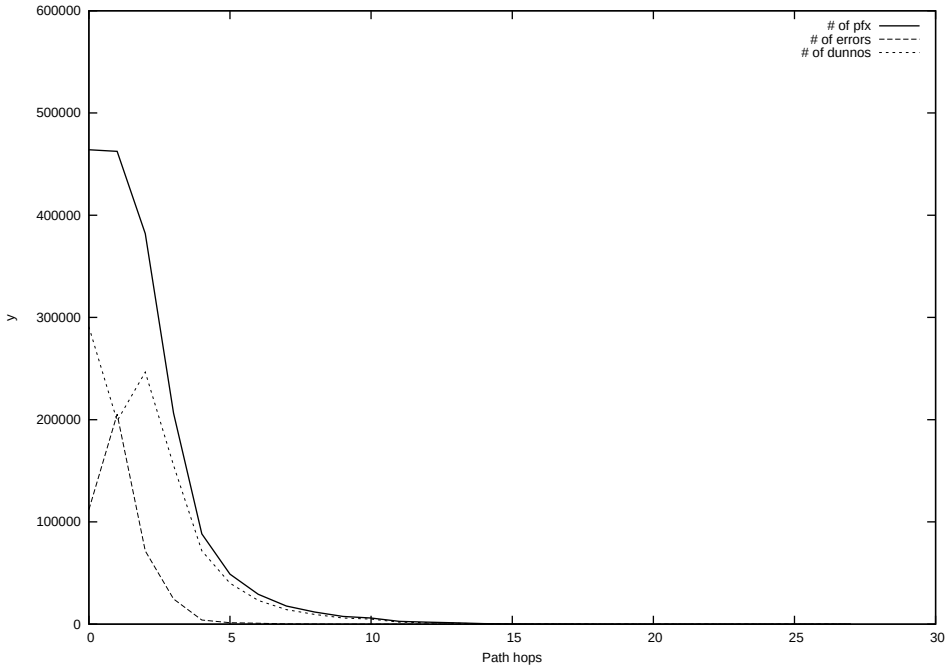
2013-08-02



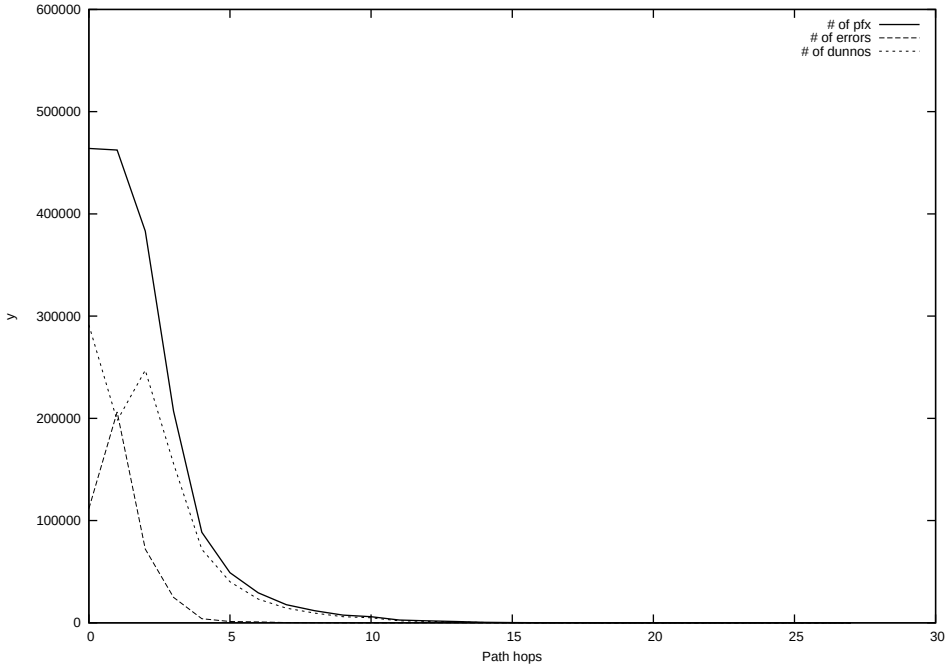
2013-08-03



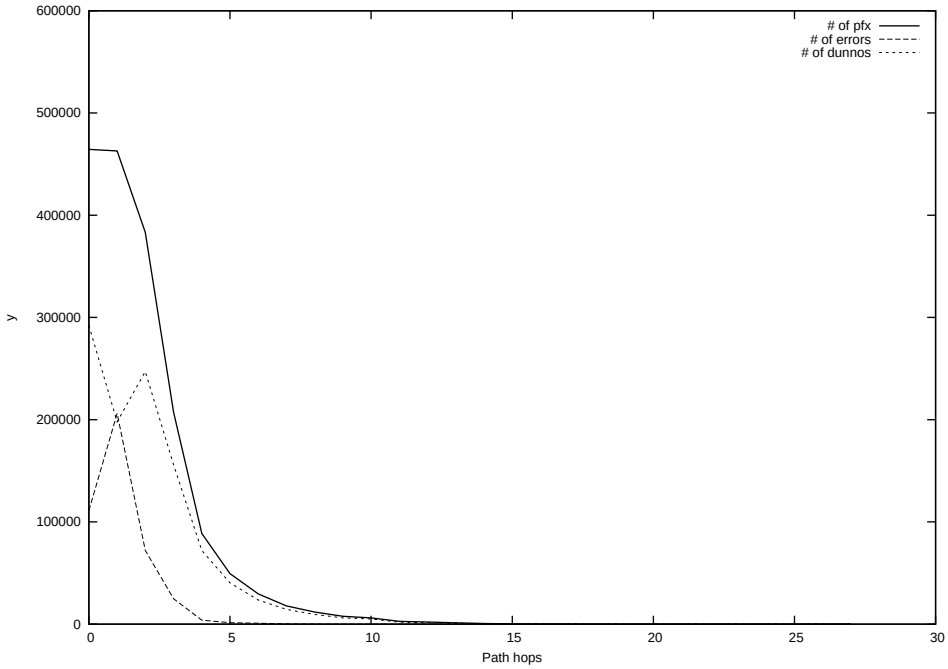
2013-08-04



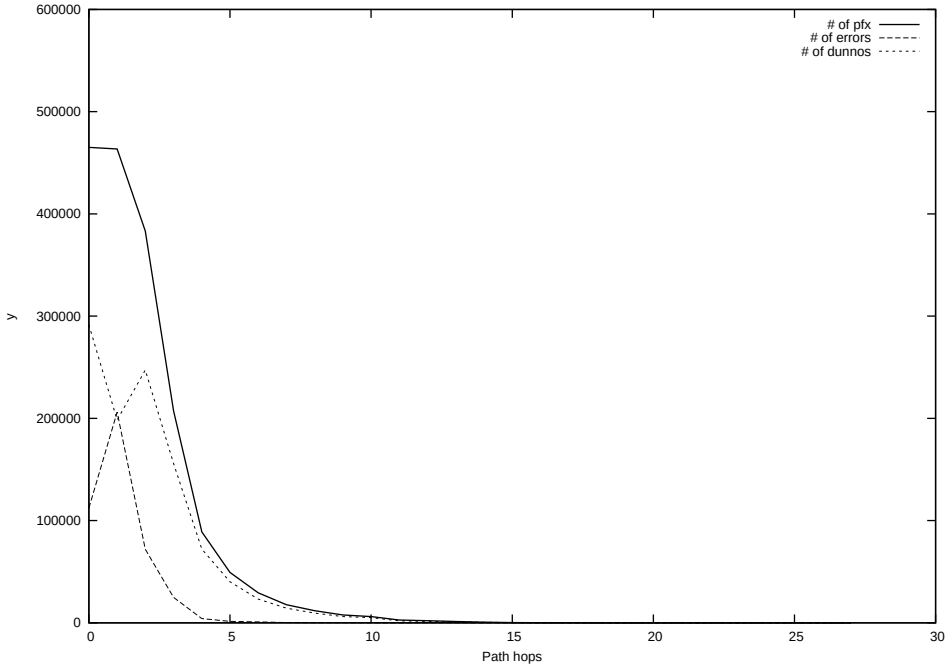
2013-08-05



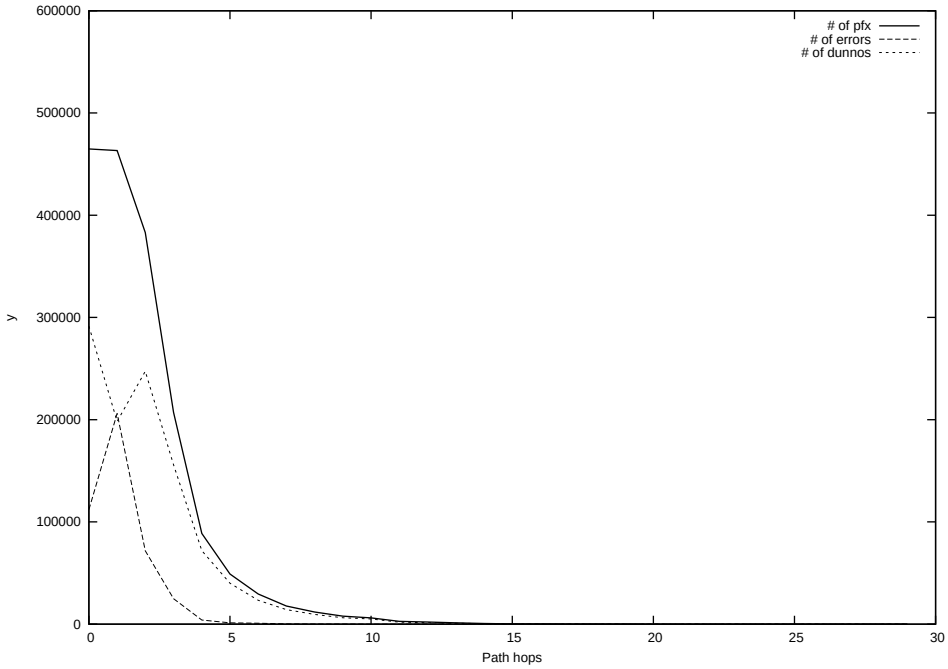
2013-08-06



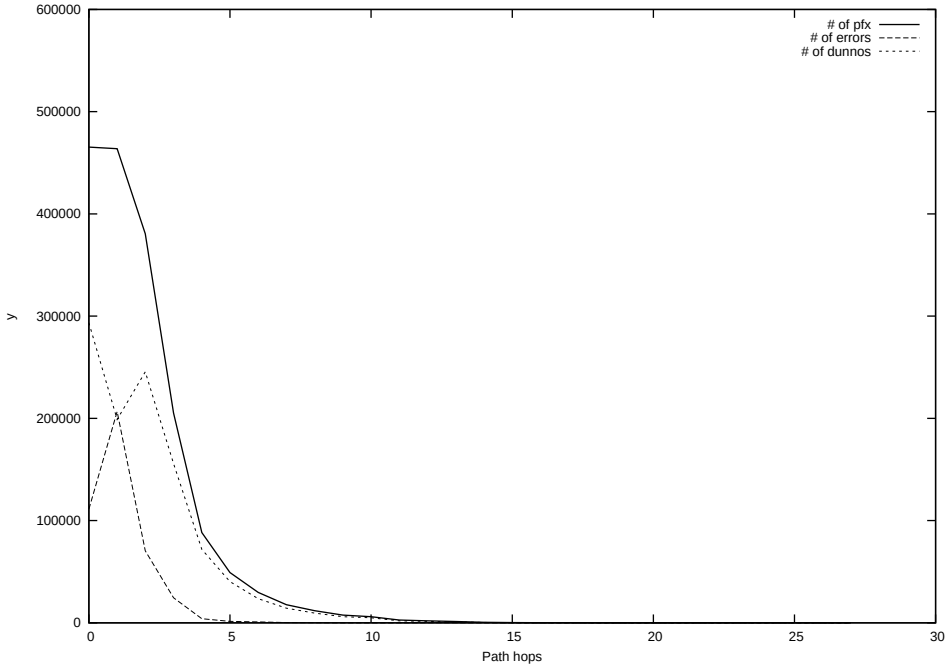
2013-08-07



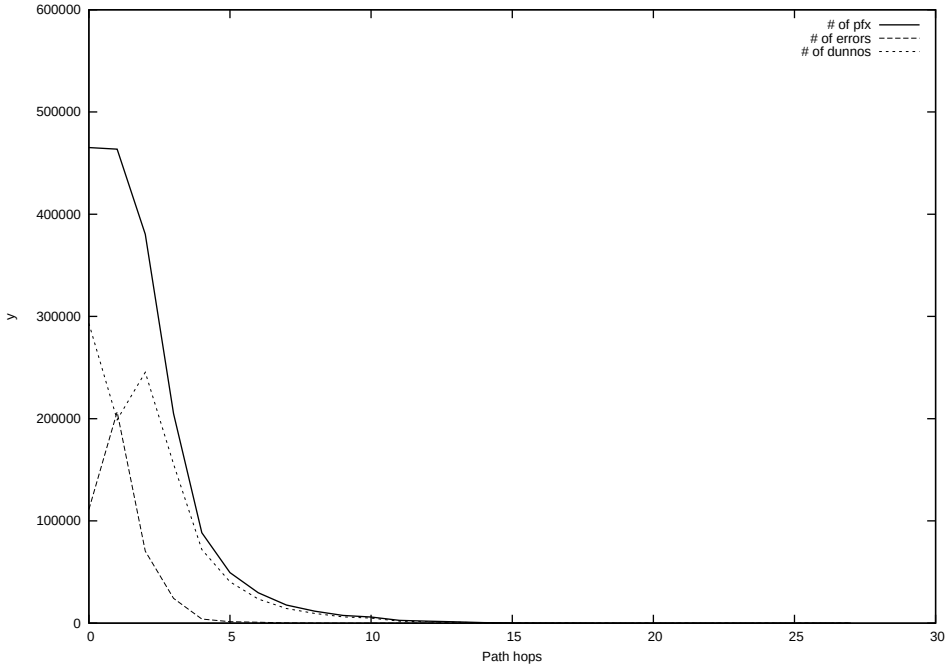
2013-08-08



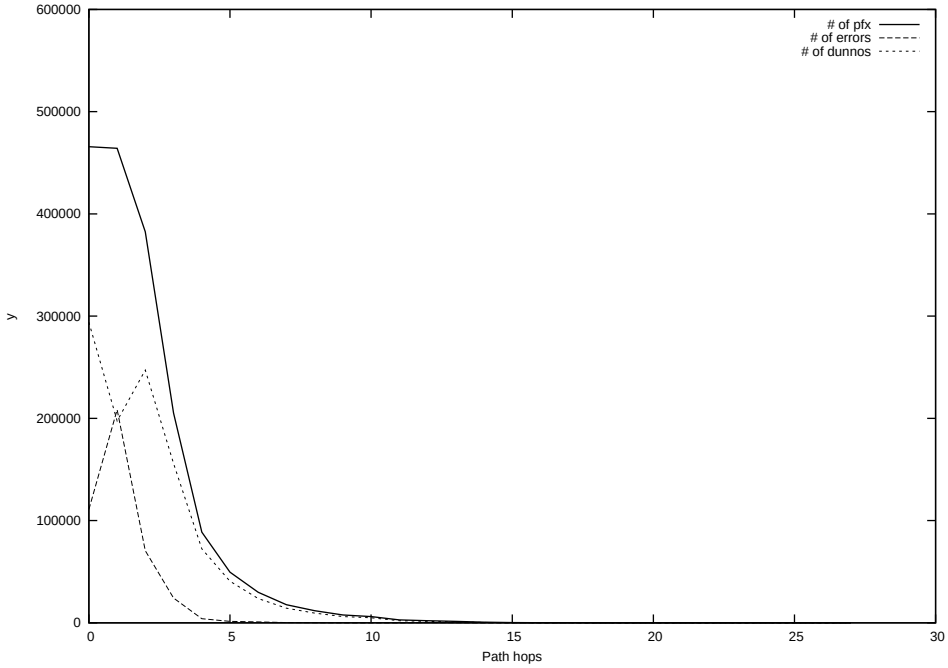
2013-08-09



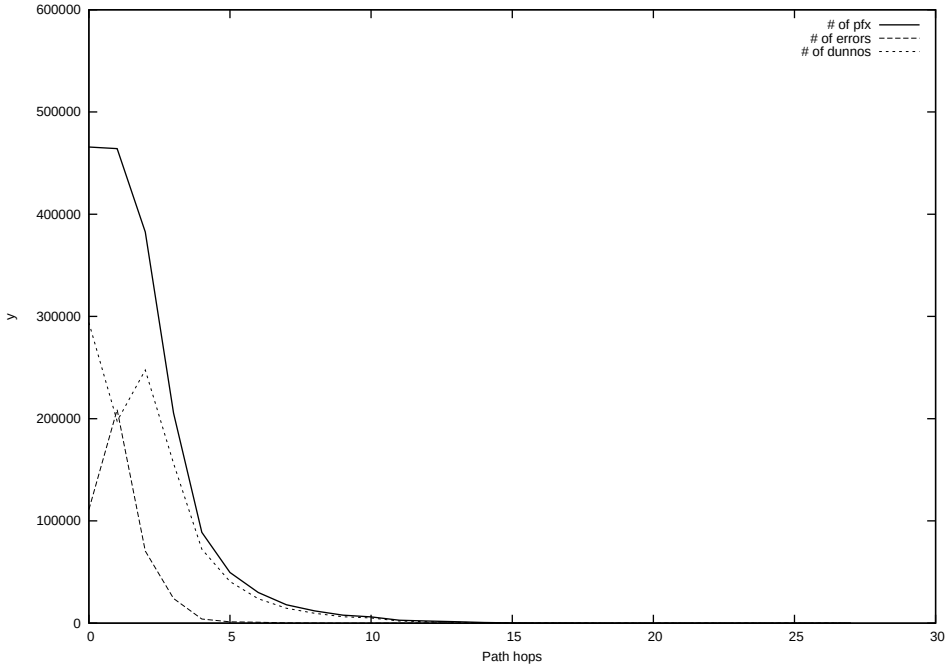
2013-08-10



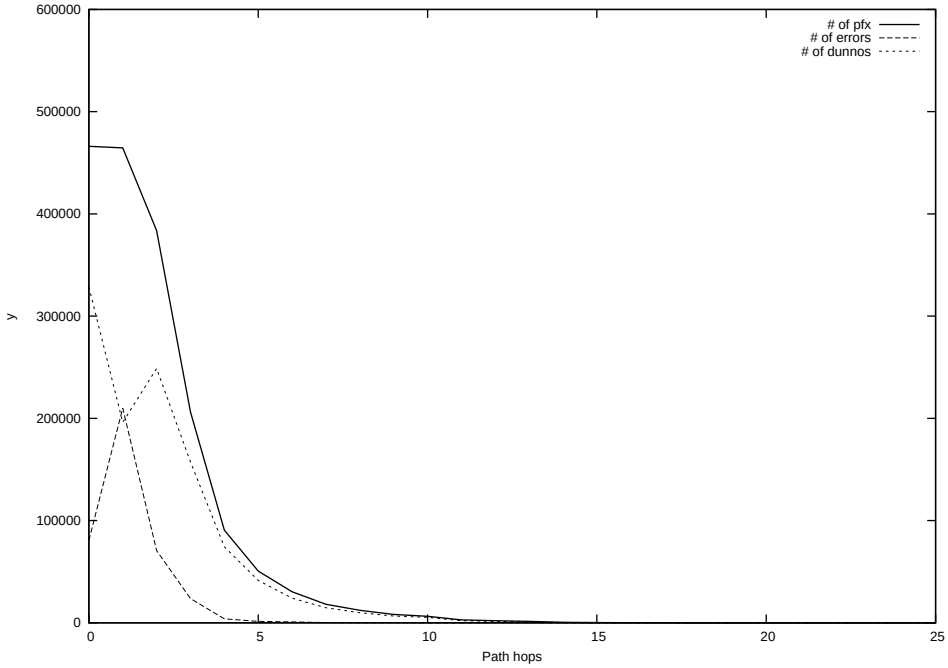
2013-08-11



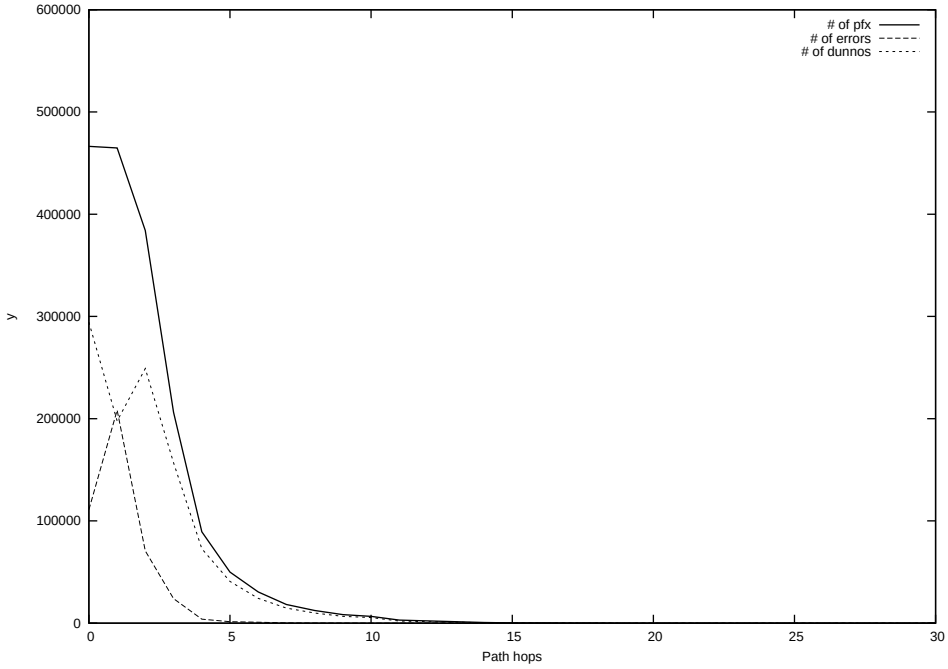
2013-08-12



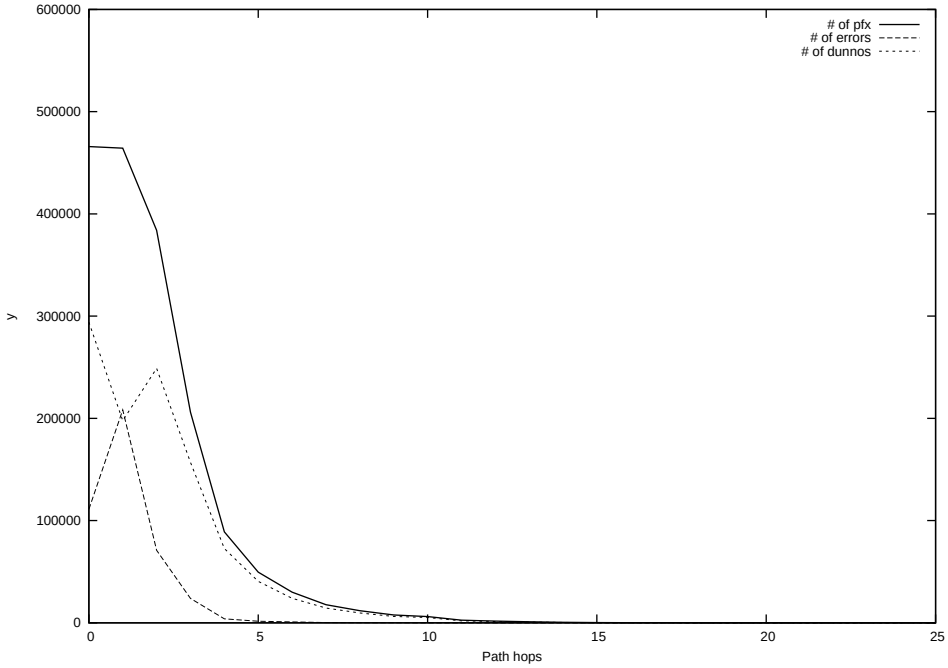
2013-08-13



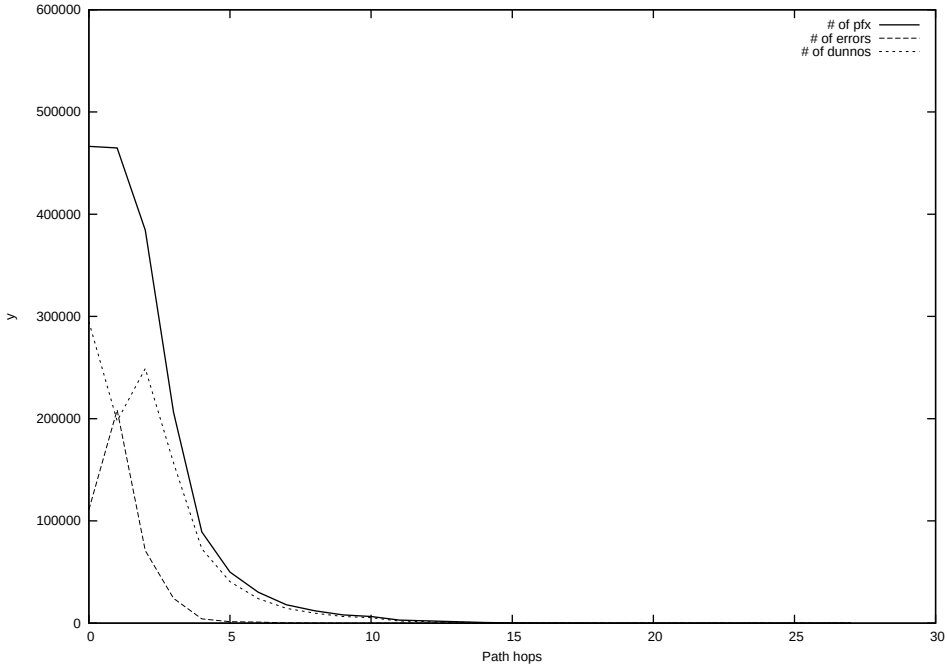
2013-08-15



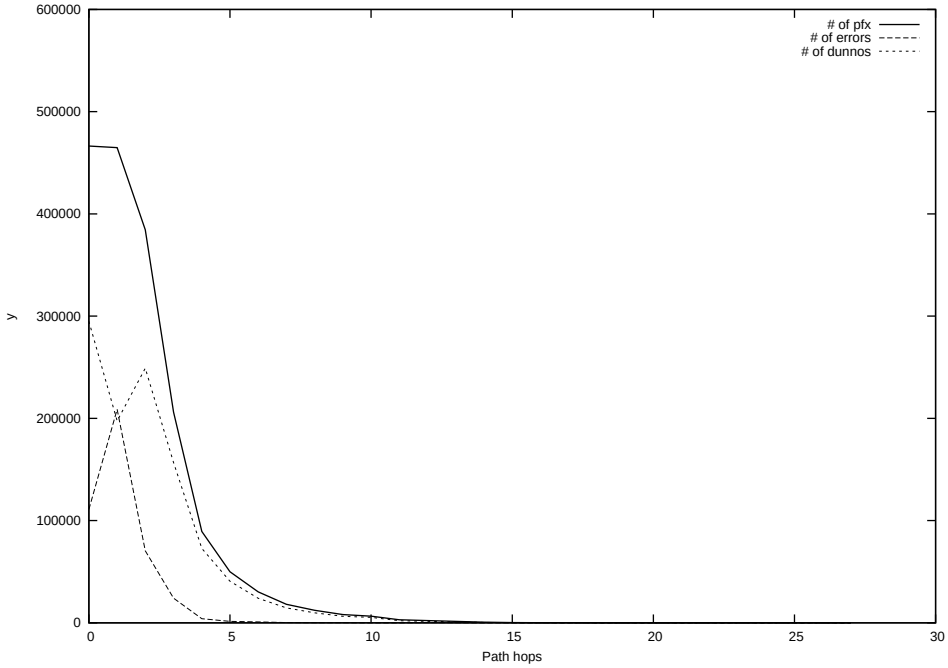
2013-08-16



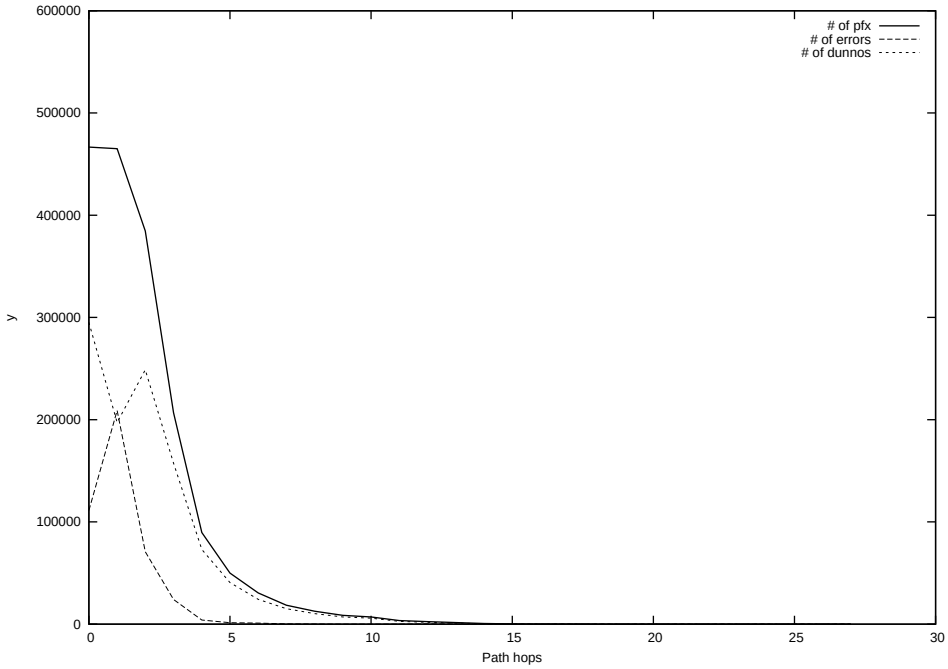
2013-08-17



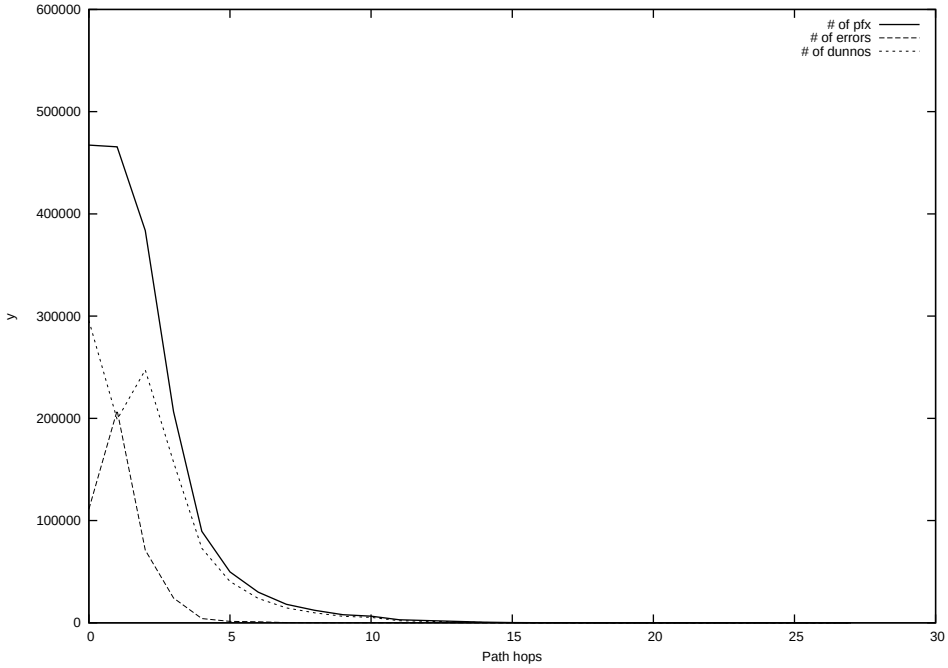
2013-08-18



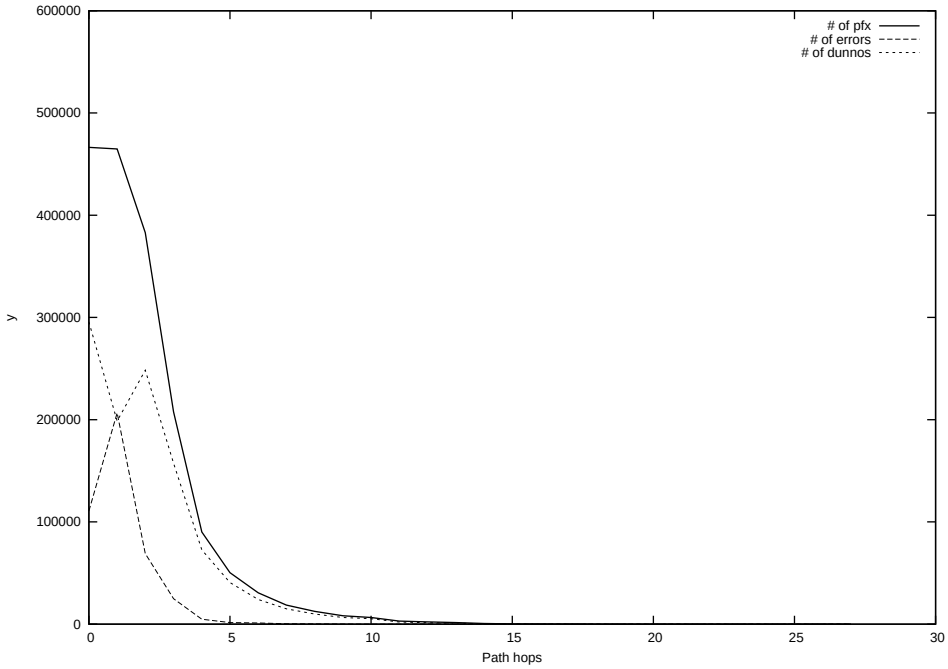
2013-08-19



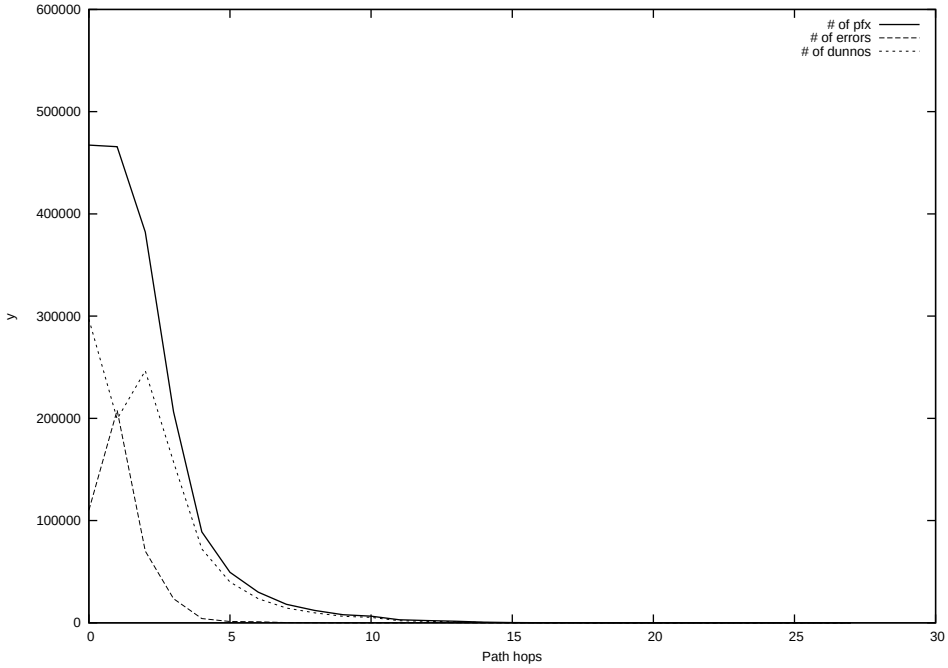
2013-08-20



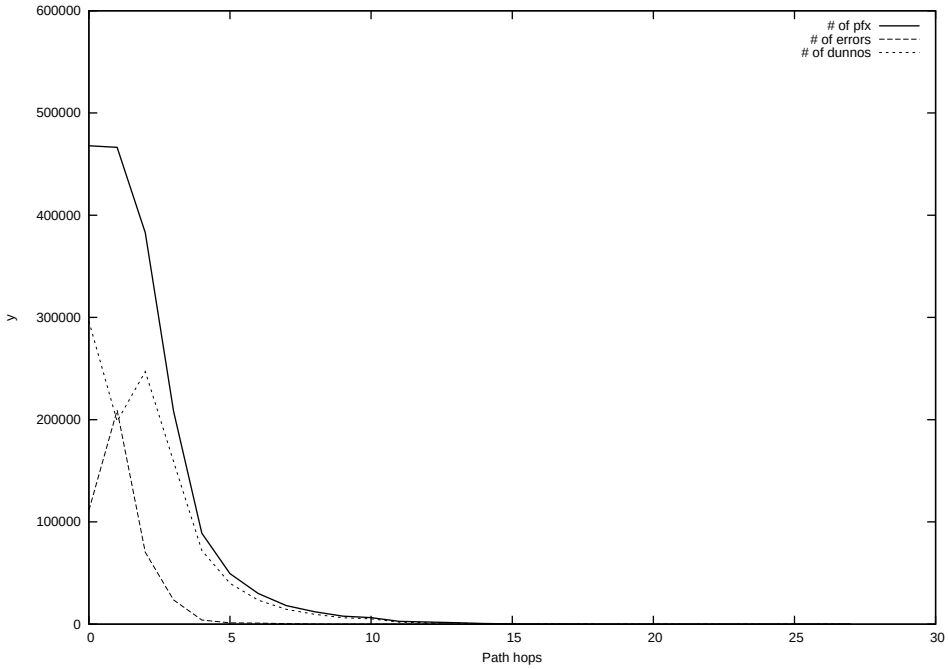
2013-08-21



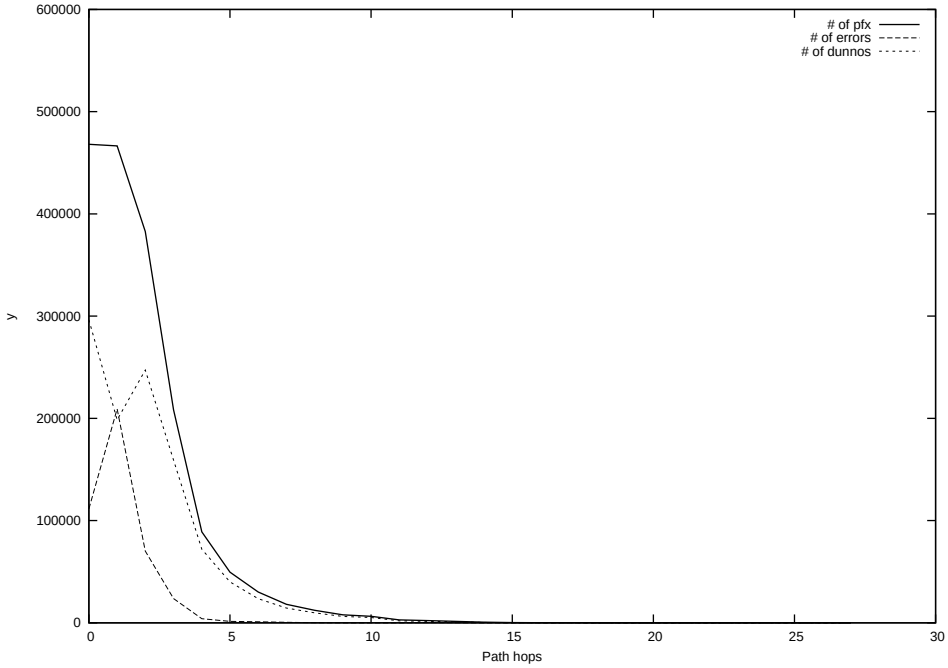
2013-08-22



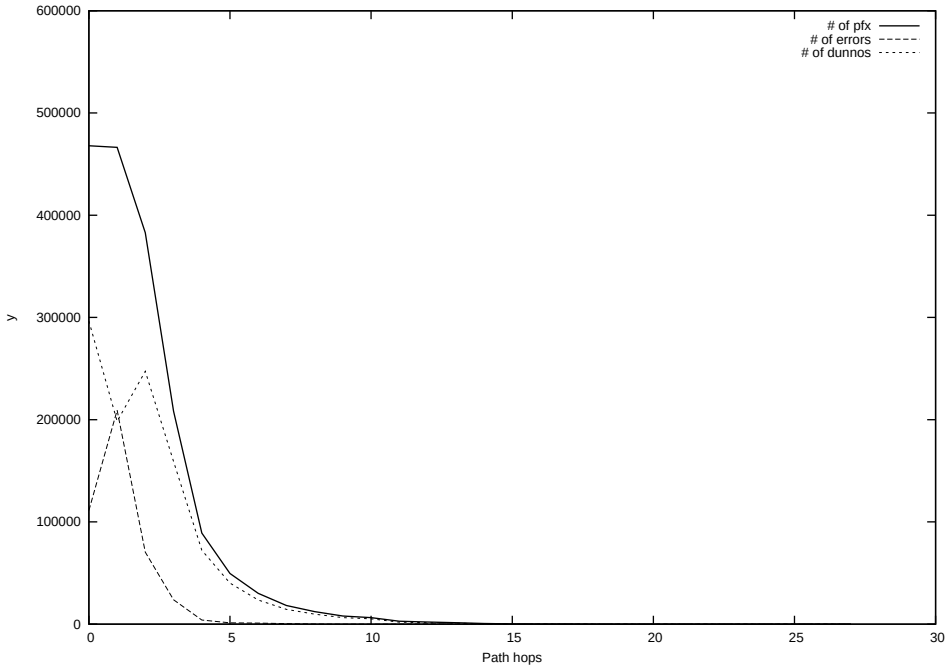
2013-08-23



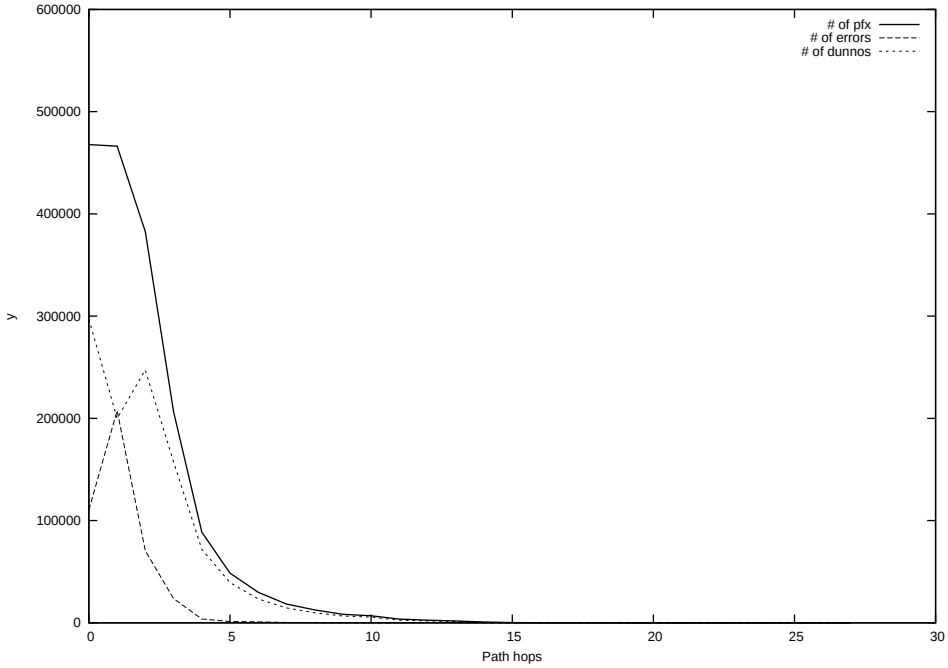
2013-08-24



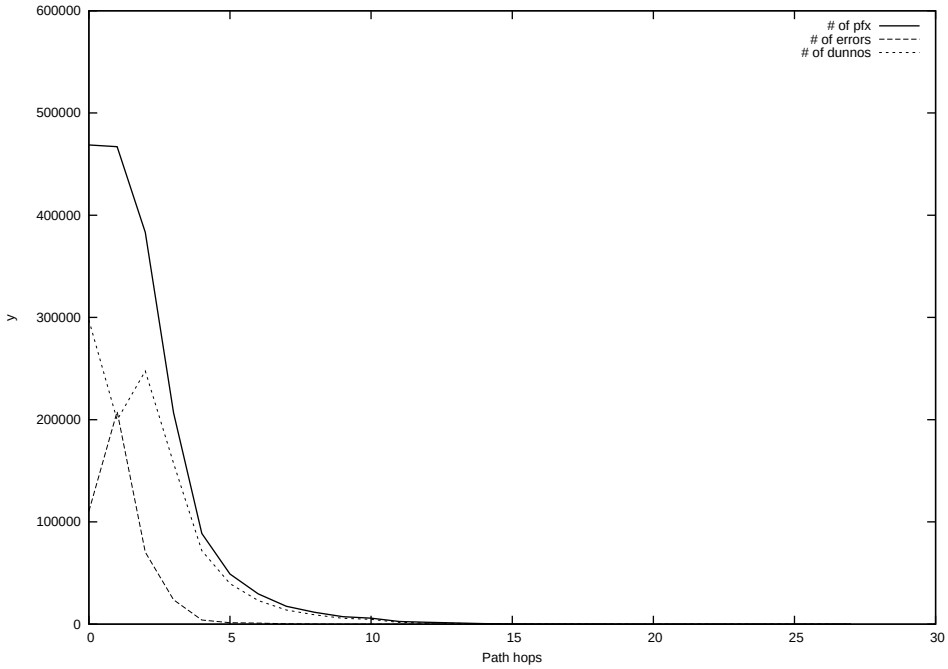
2013-08-25



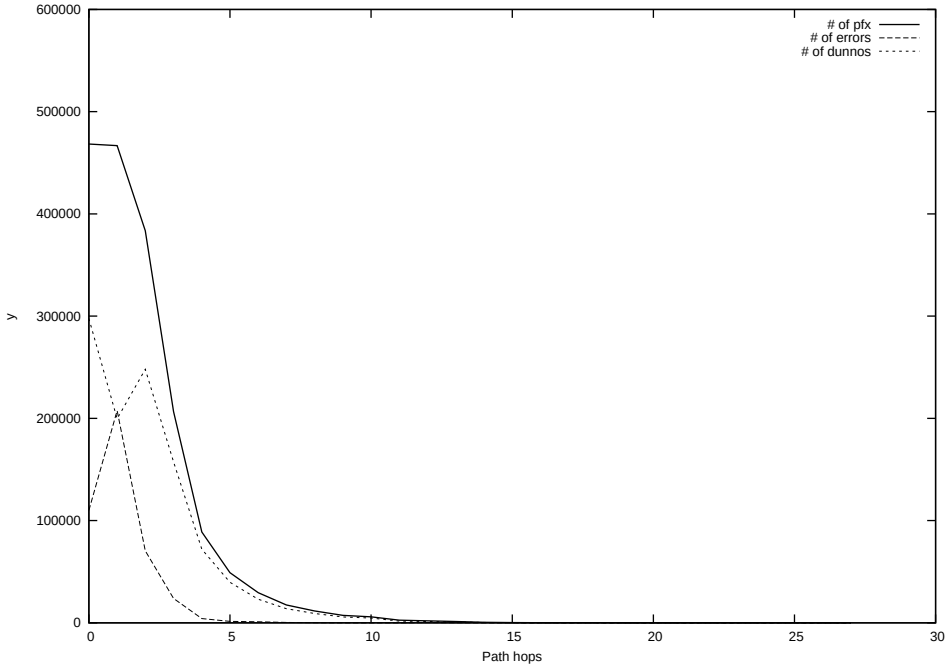
2013-08-26



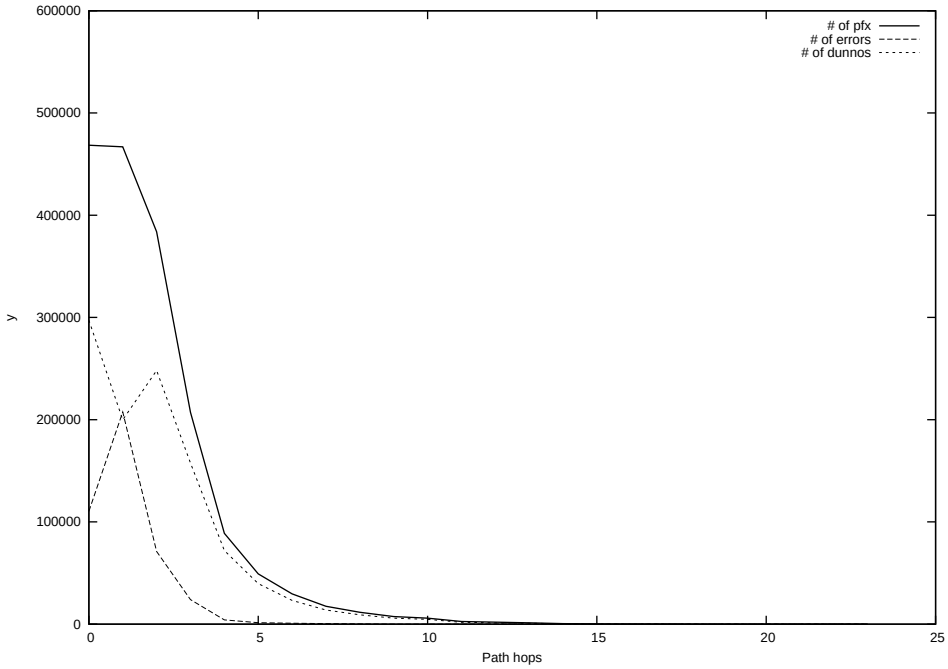
2013-08-27



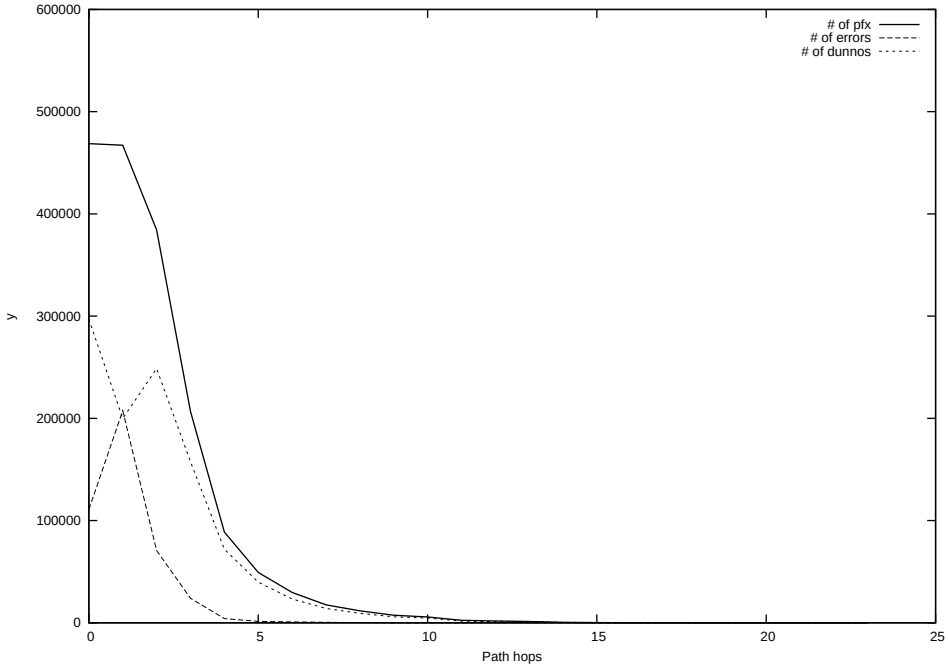
2013-08-28



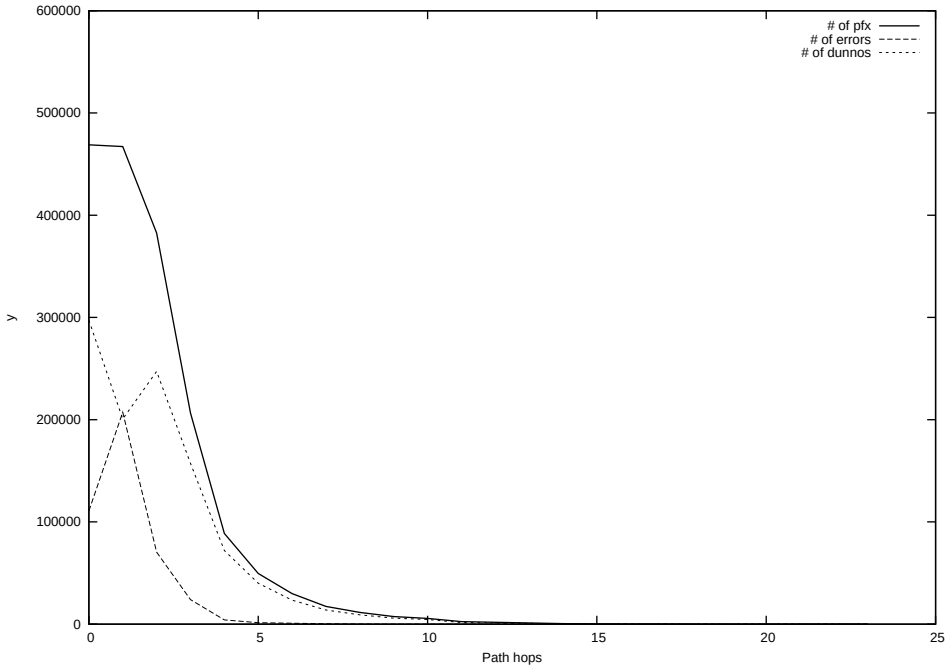
2013-08-29



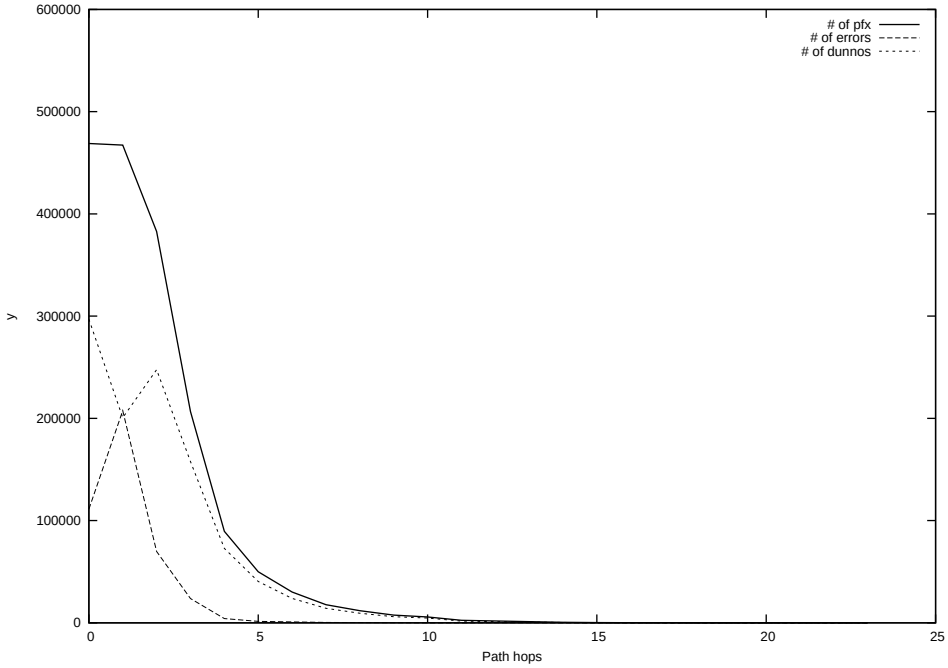
2013-08-30



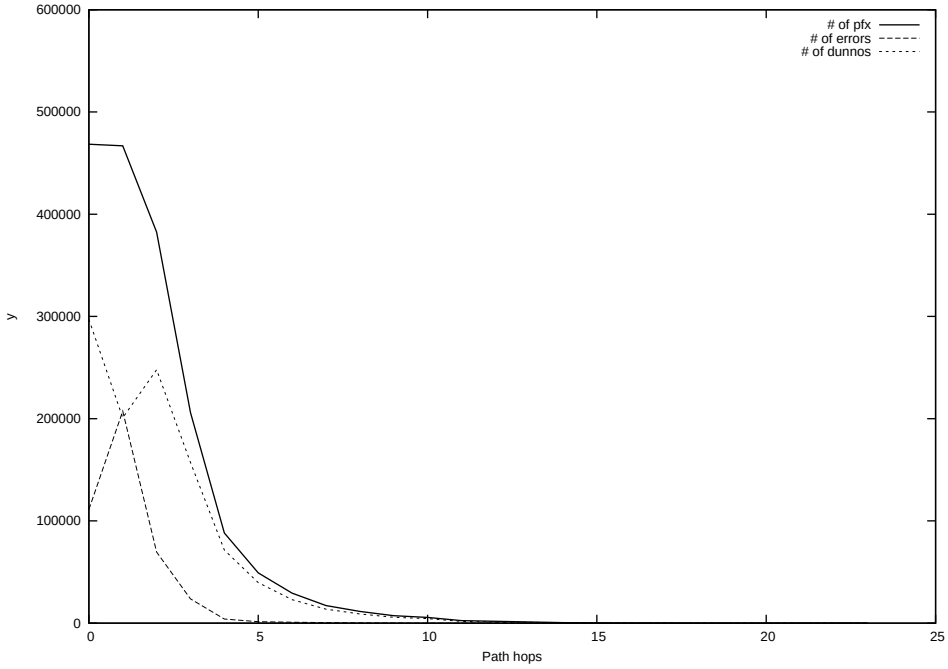
2013-08-31



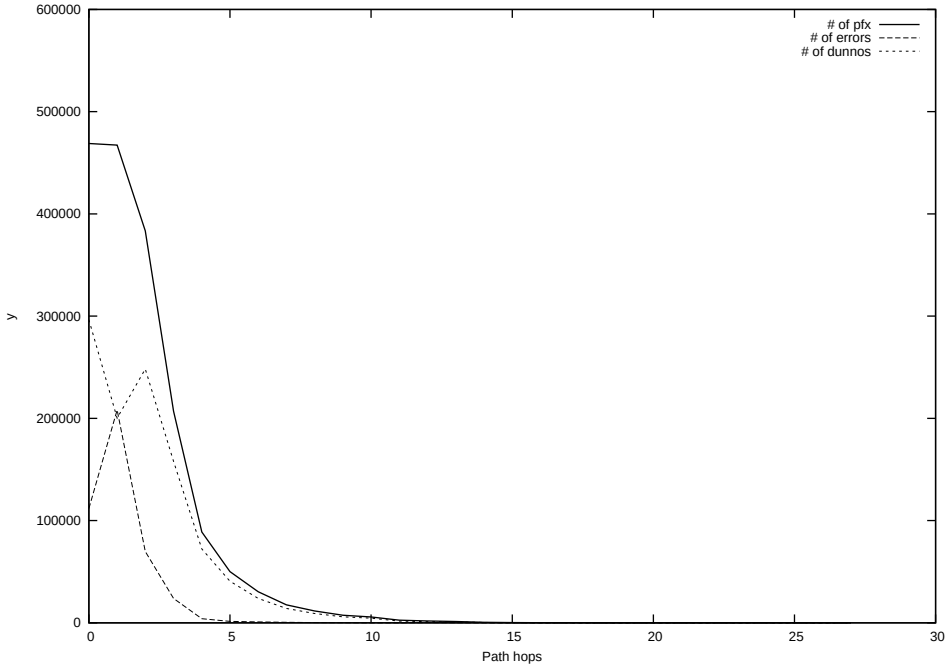
2013-09-01



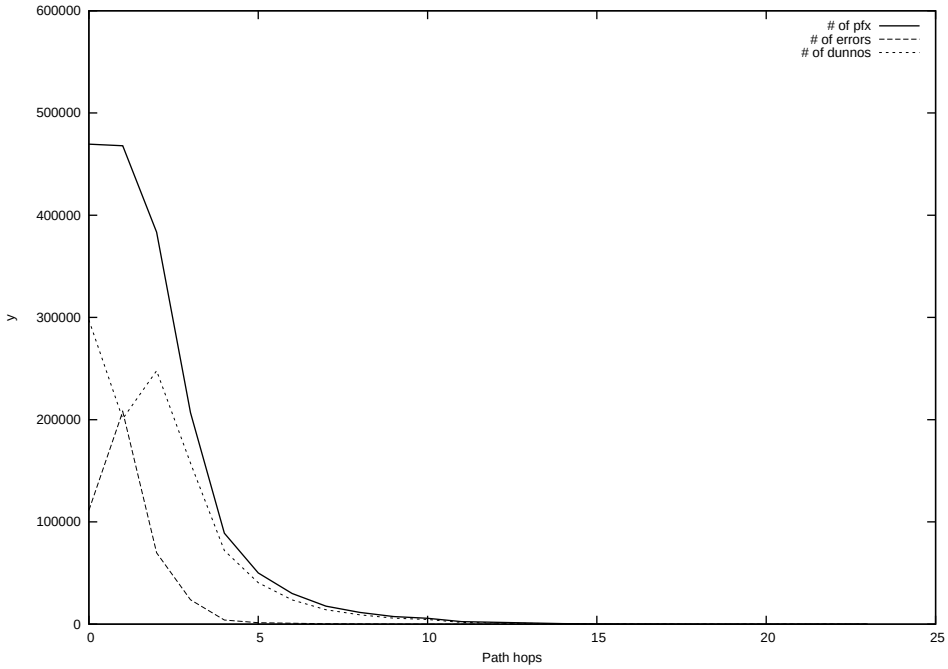
2013-09-02



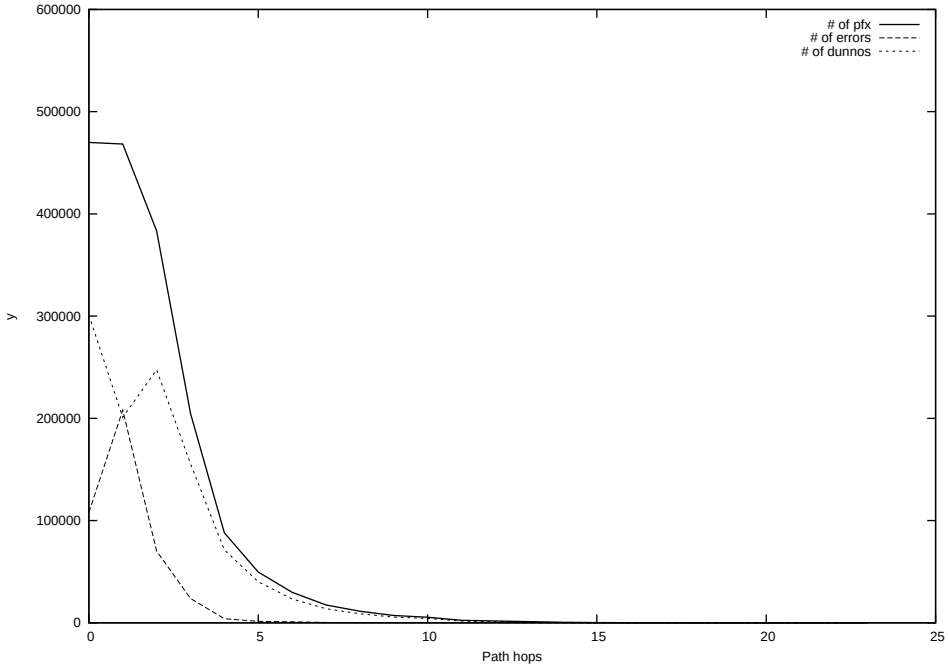
2013-09-03



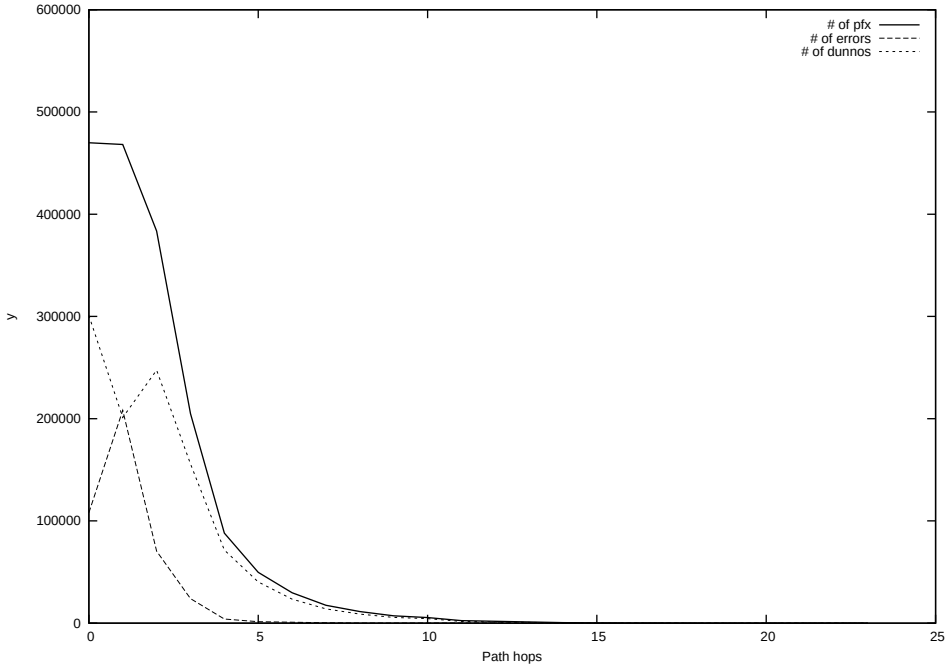
2013-09-04



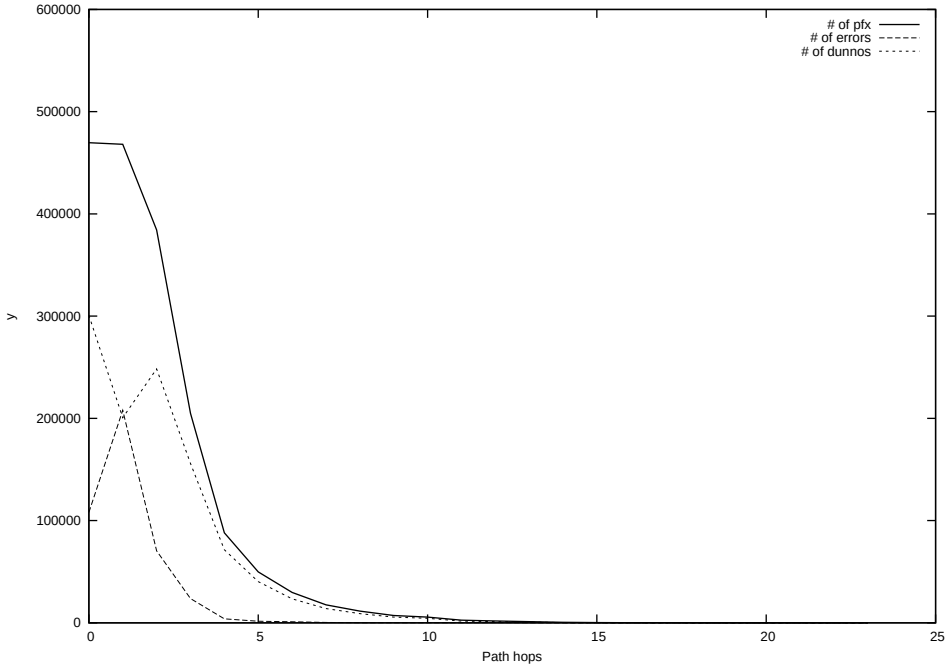
2013-09-05



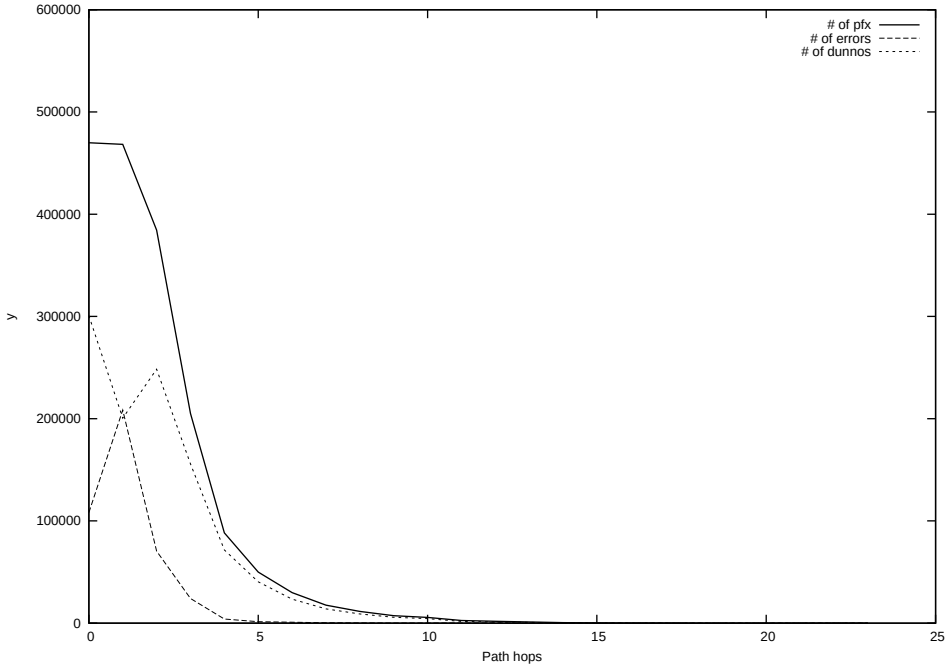
2013-09-06



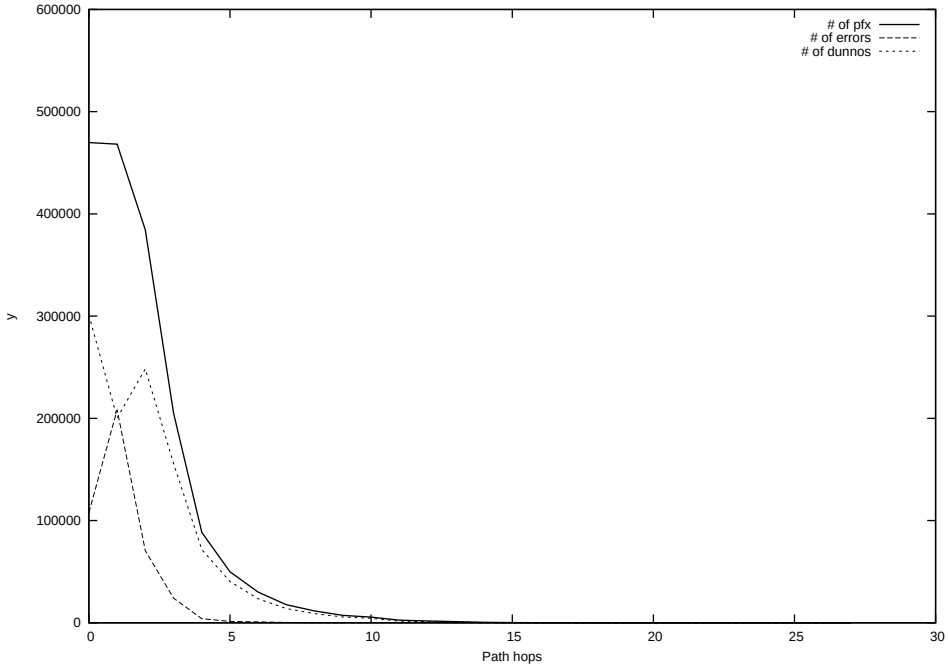
2013-09-07



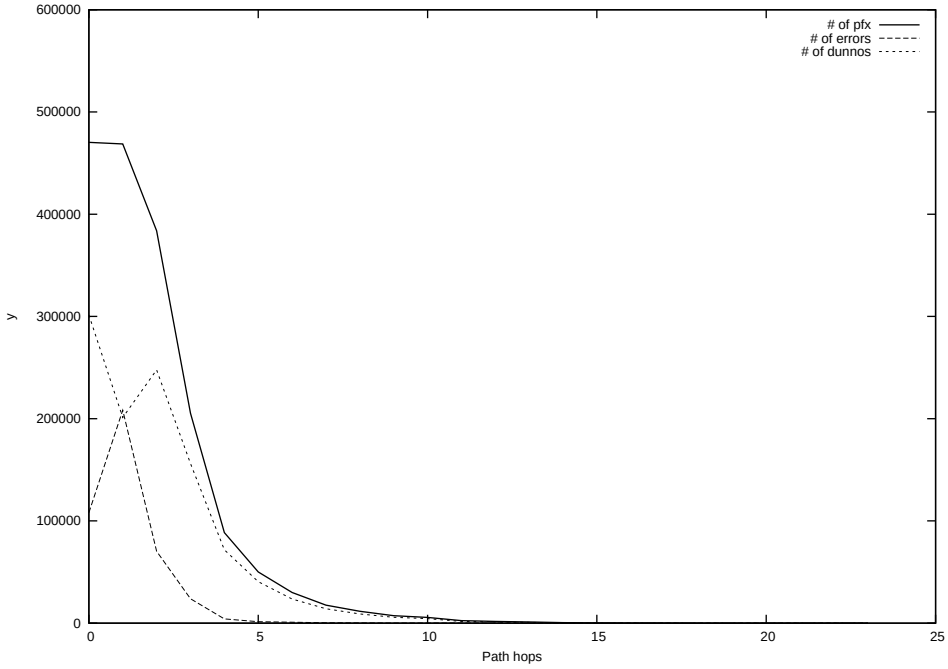
2013-09-08



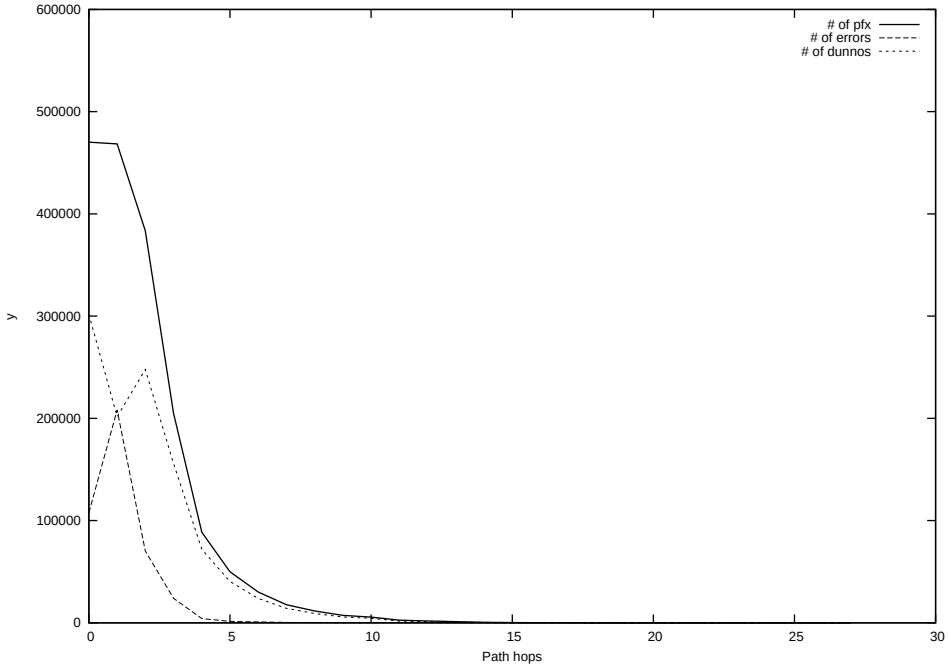
2013-09-09



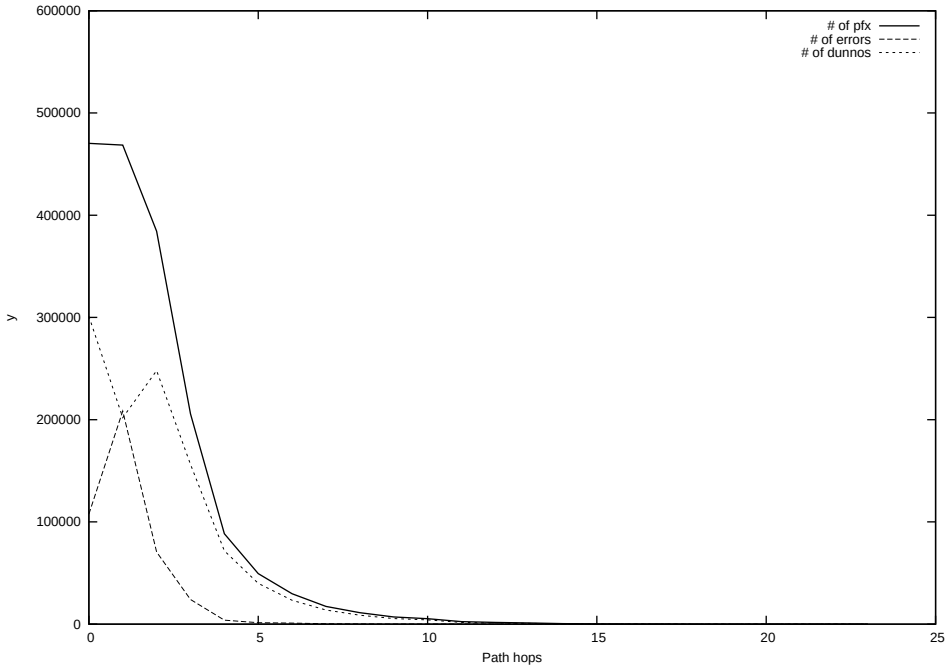
2013-09-10



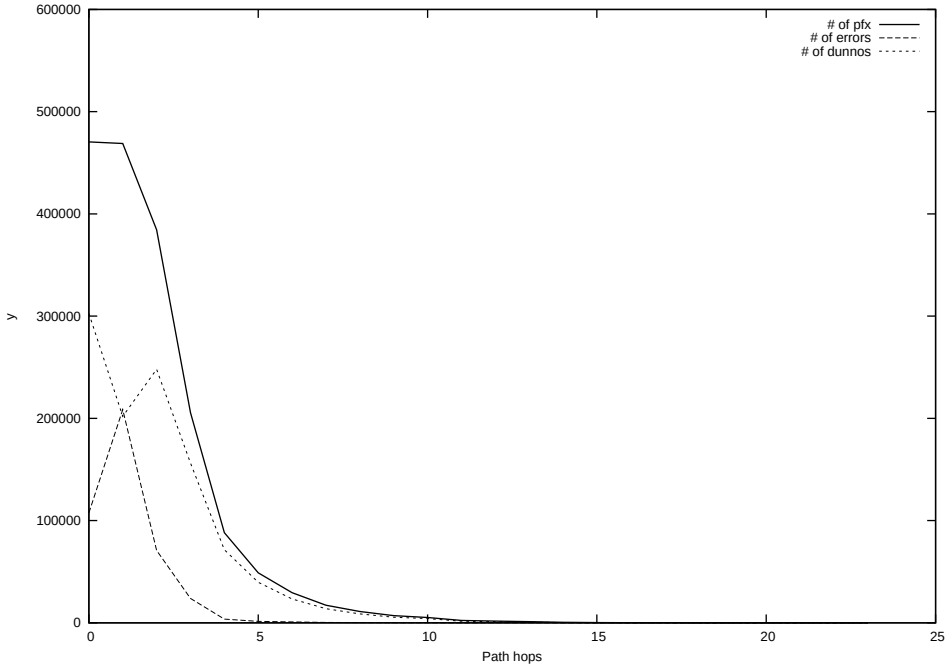
2013-09-11



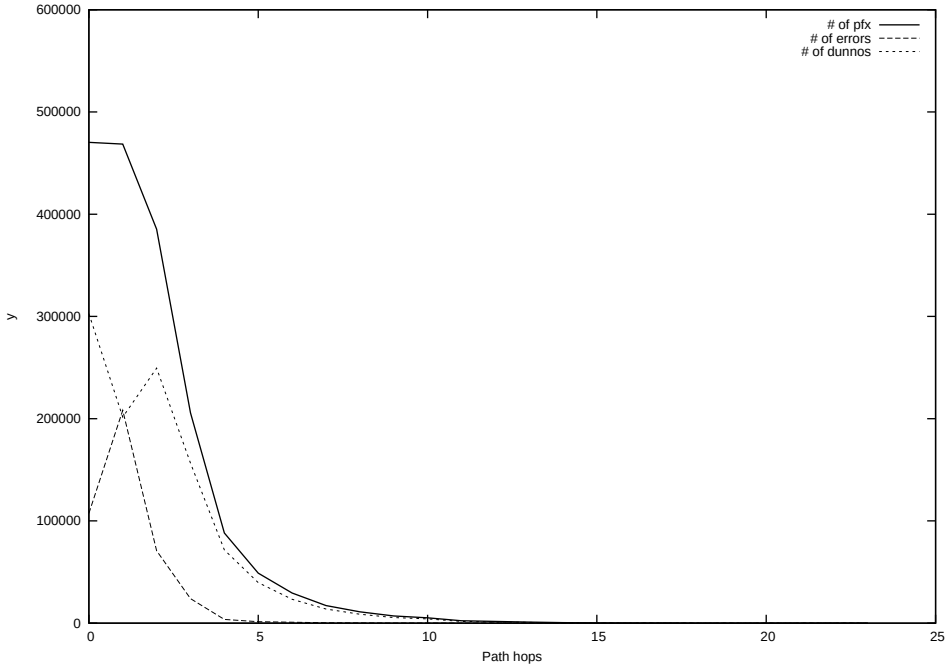
2013-09-12



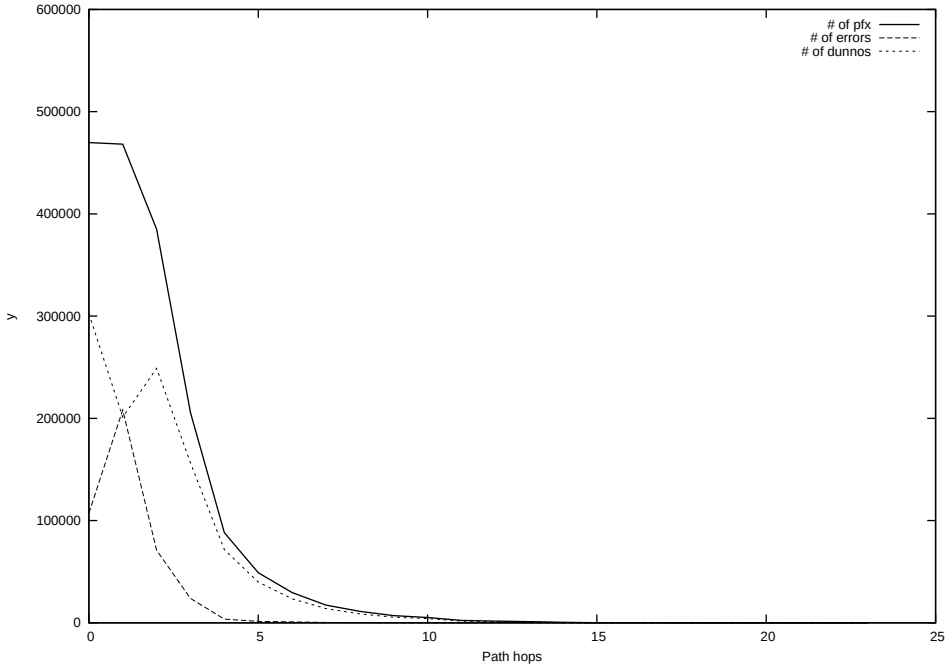
2013-09-13



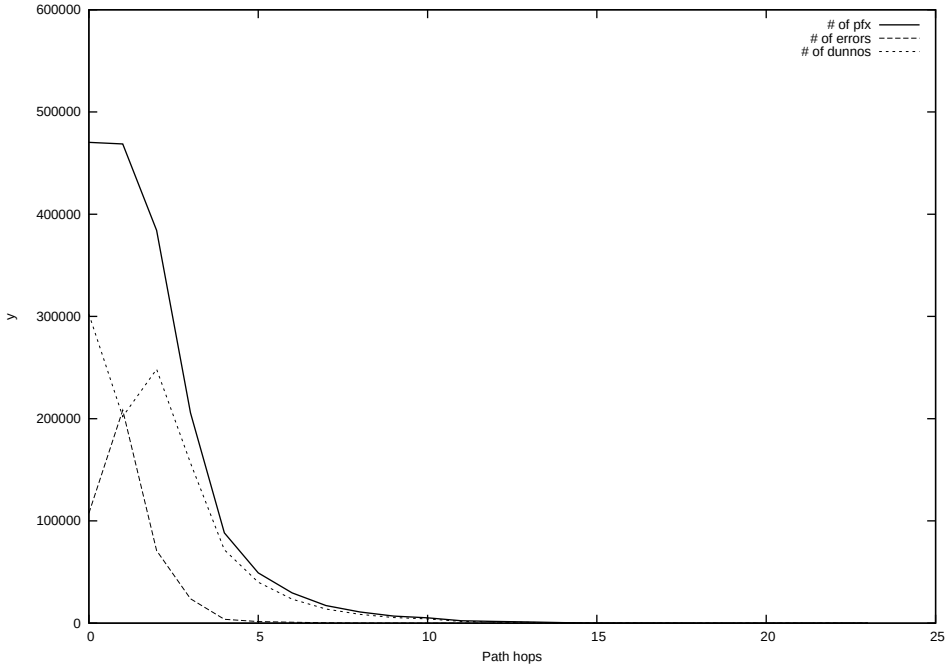
2013-09-14



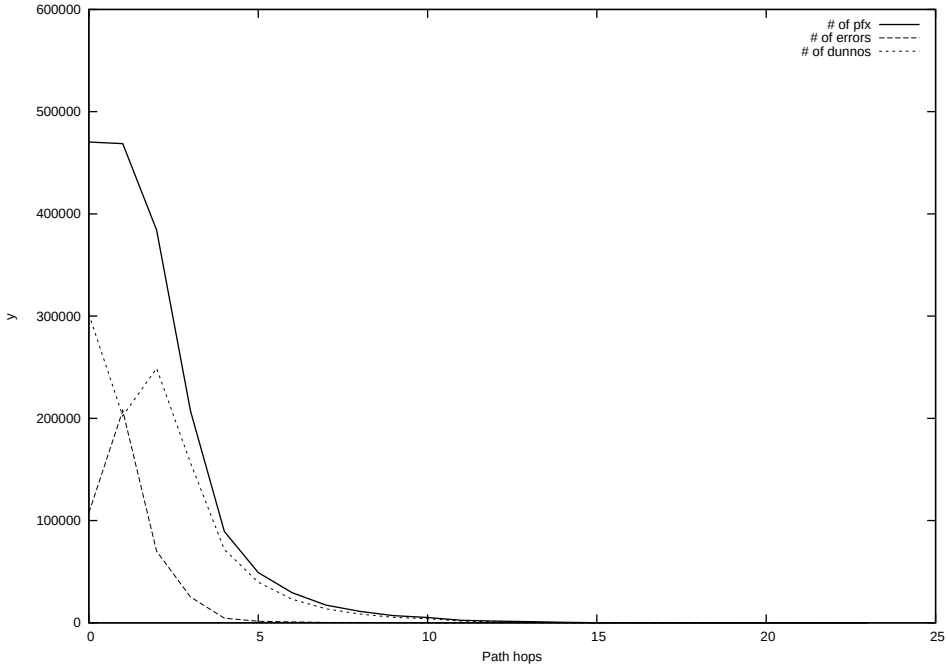
2013-09-15



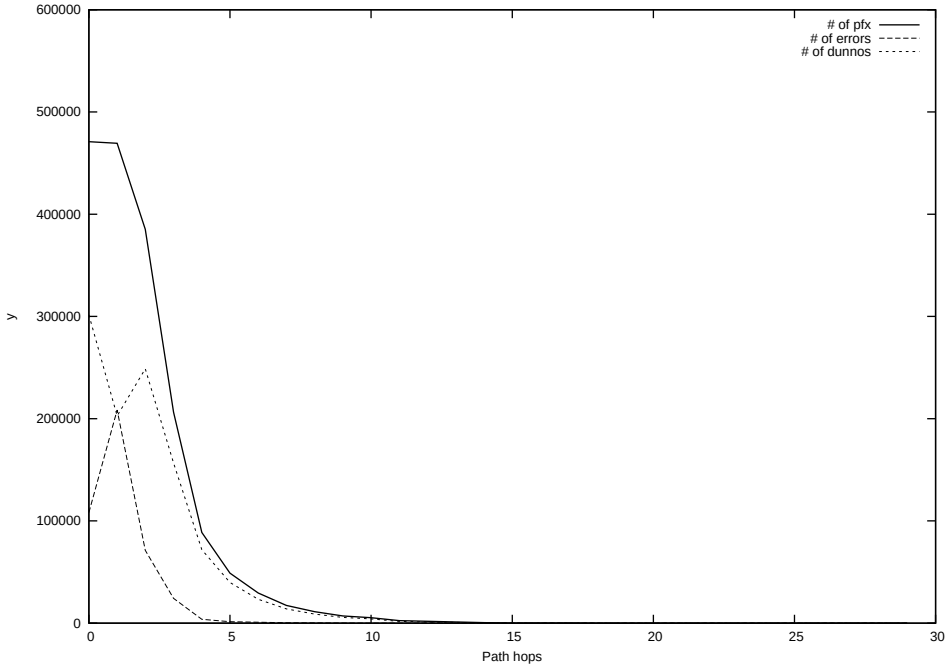
2013-09-16



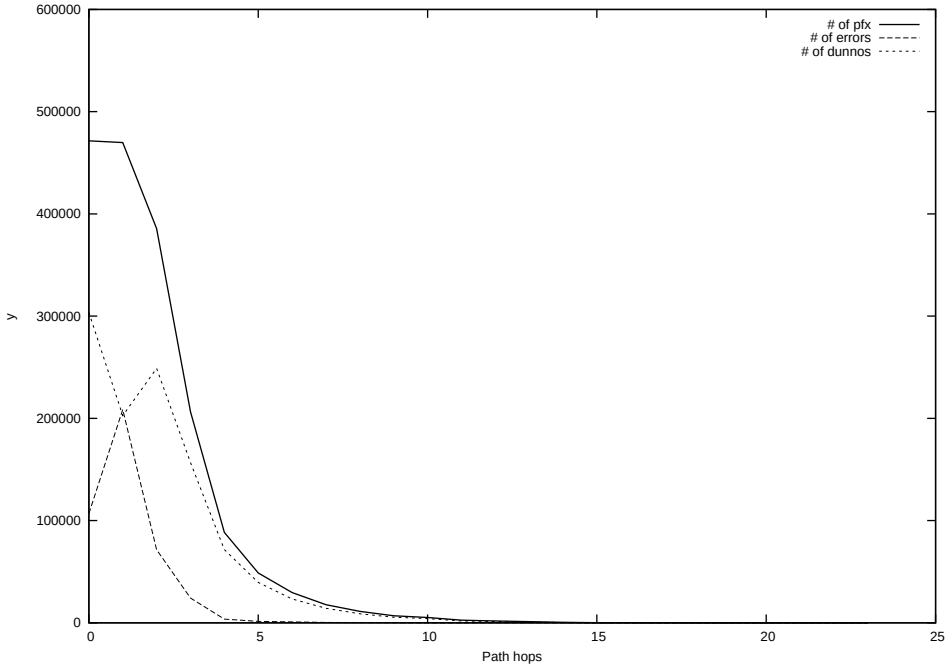
2013-09-17



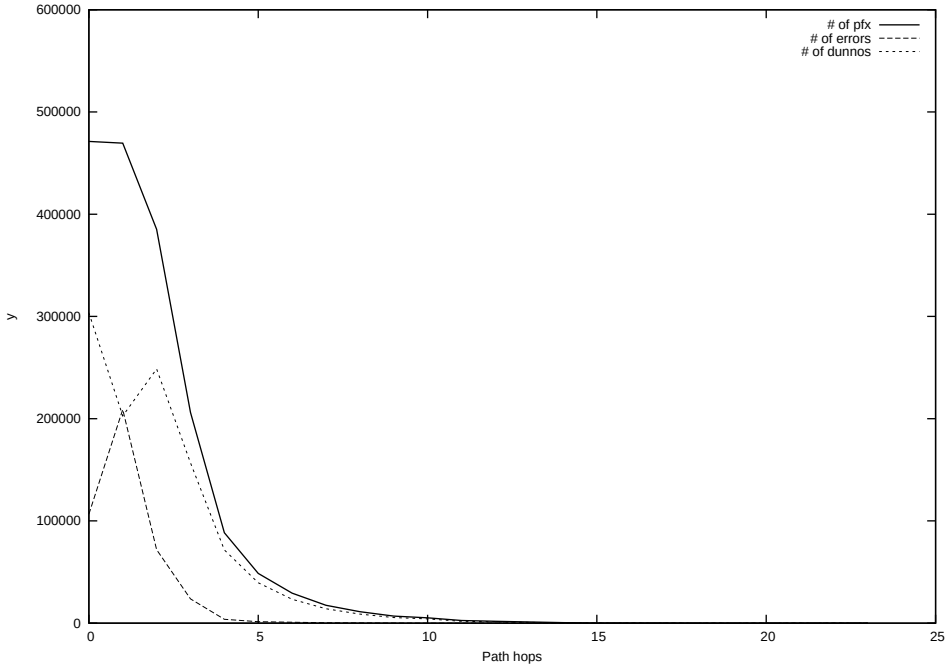
2013-09-18



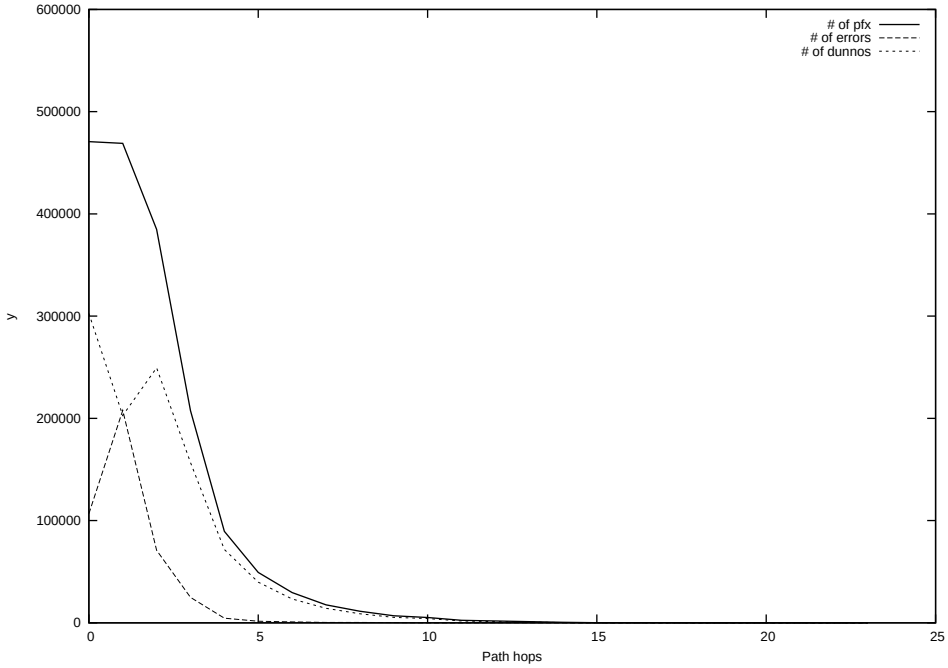
2013-09-19



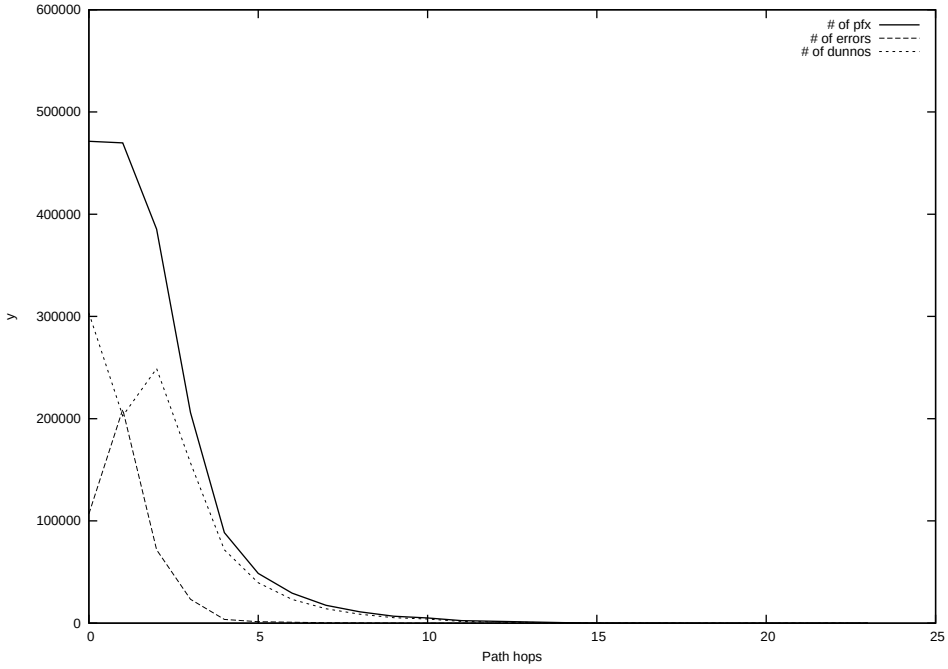
2013-09-20



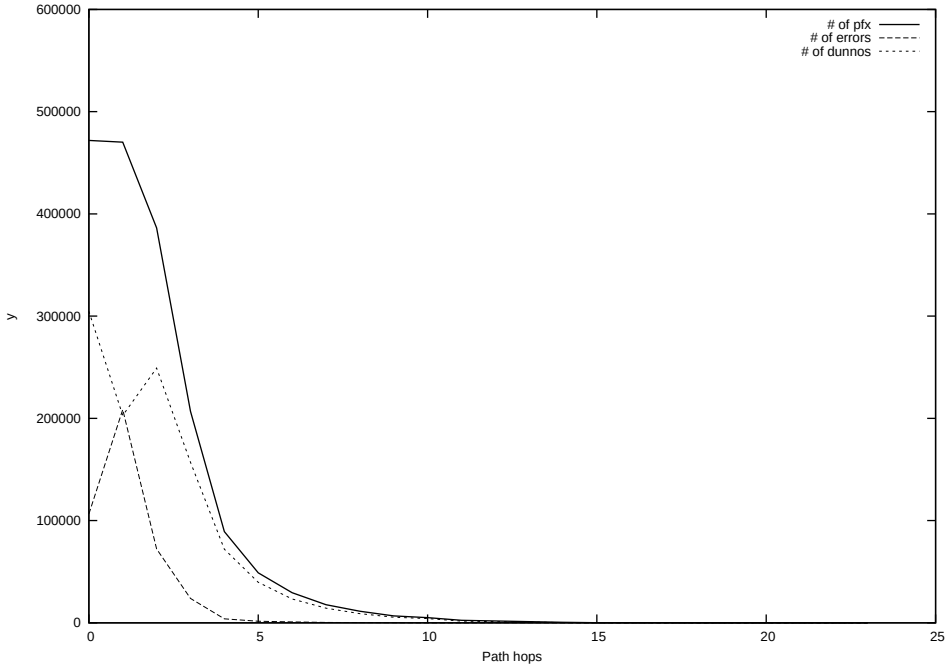
2013-09-21



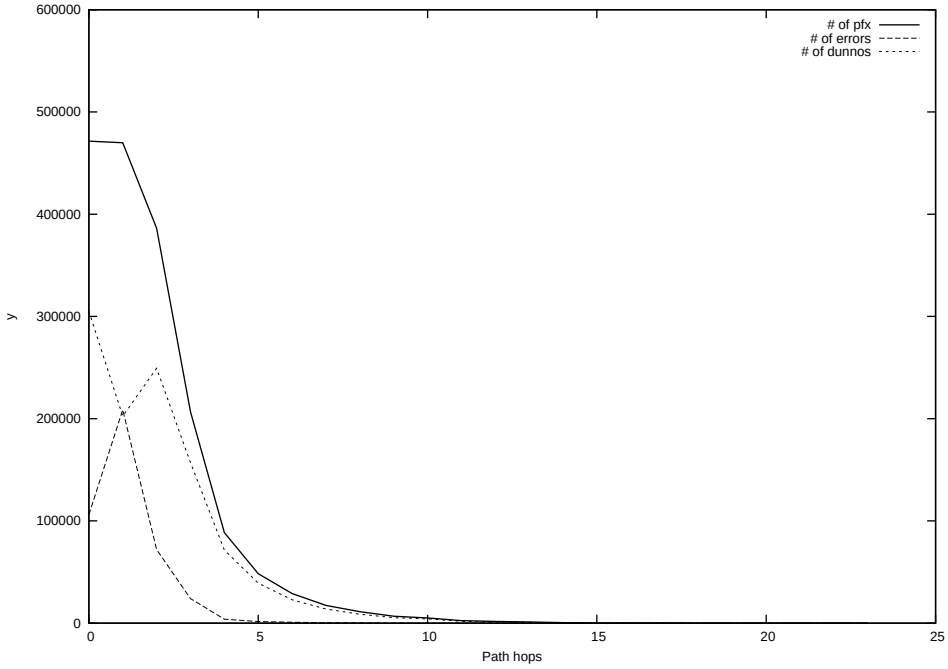
2013-09-22



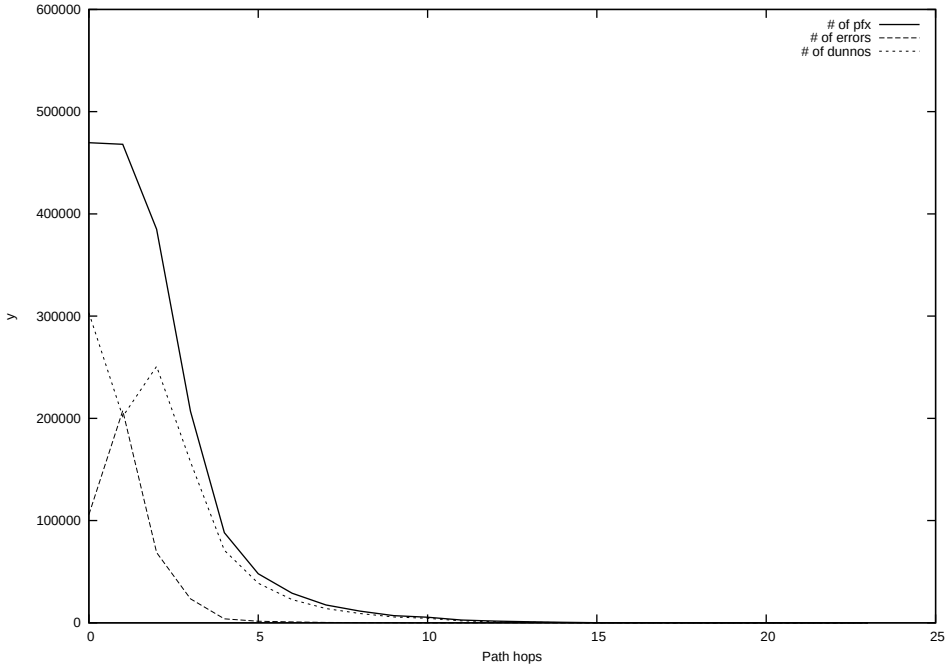
2013-09-23



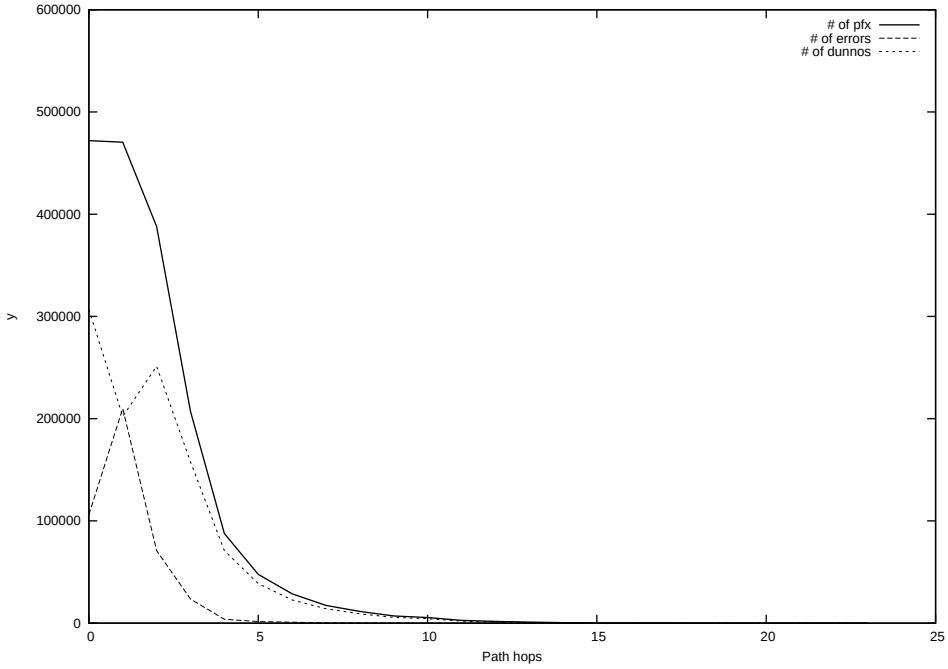
2013-09-24



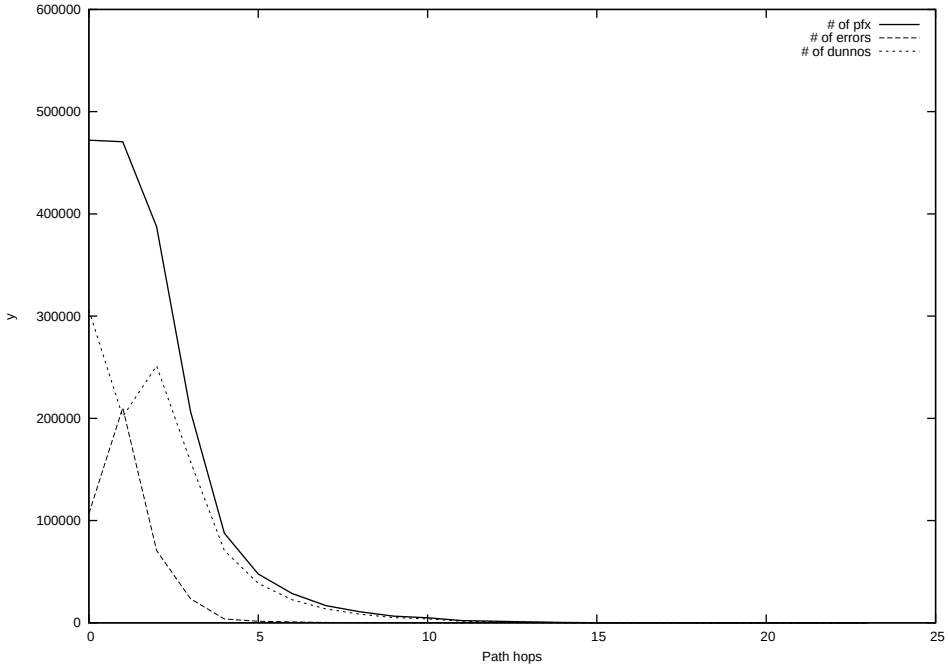
2013-09-25



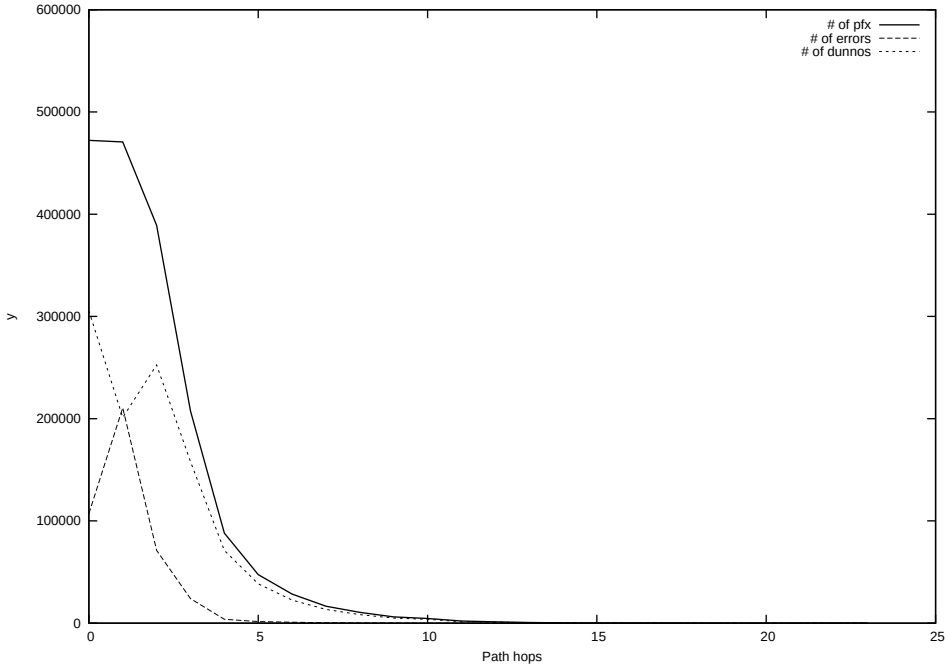
2013-09-26



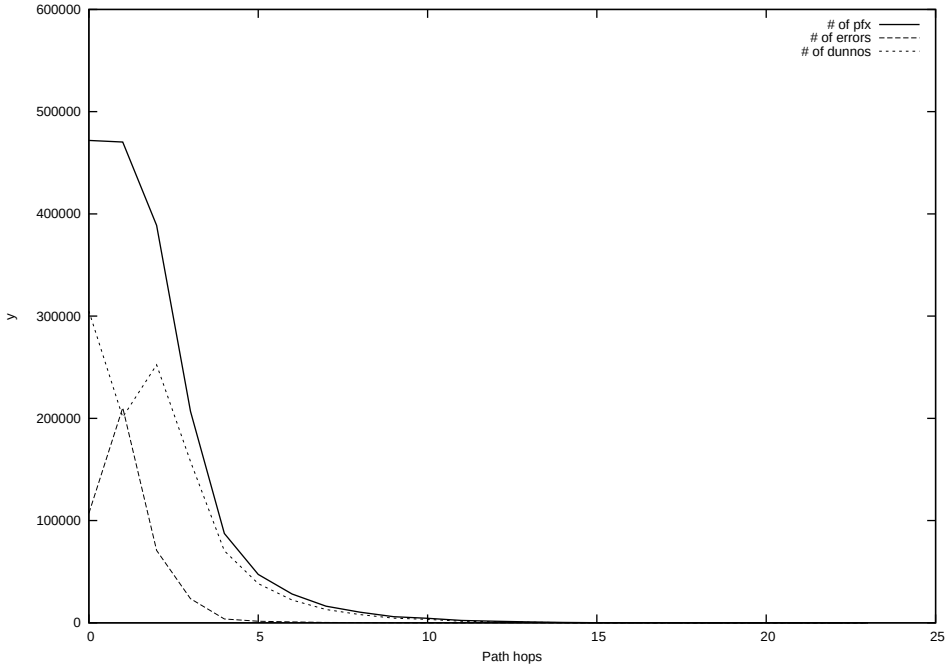
2013-09-27



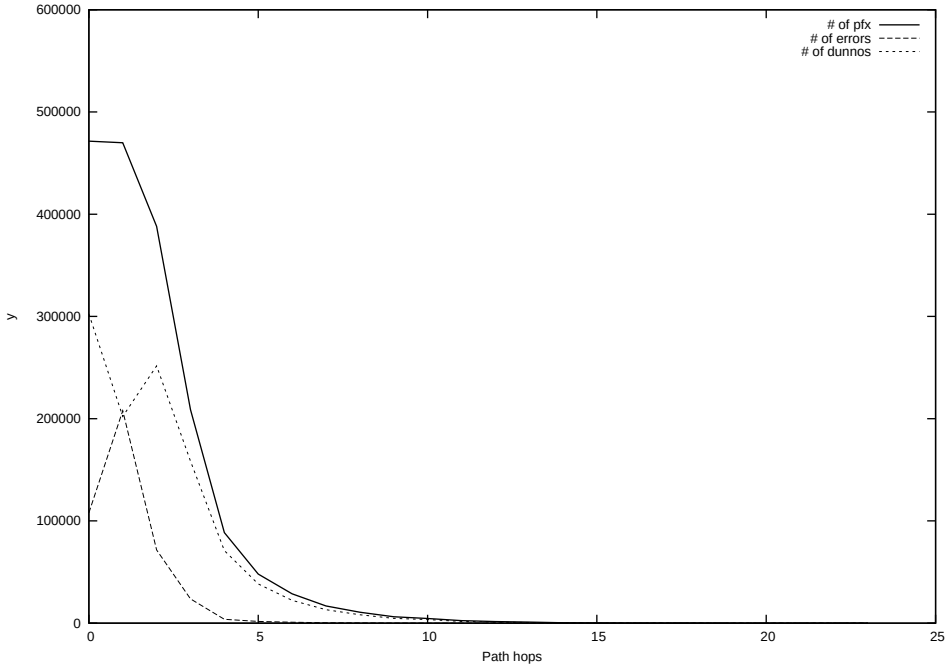
2013-09-28



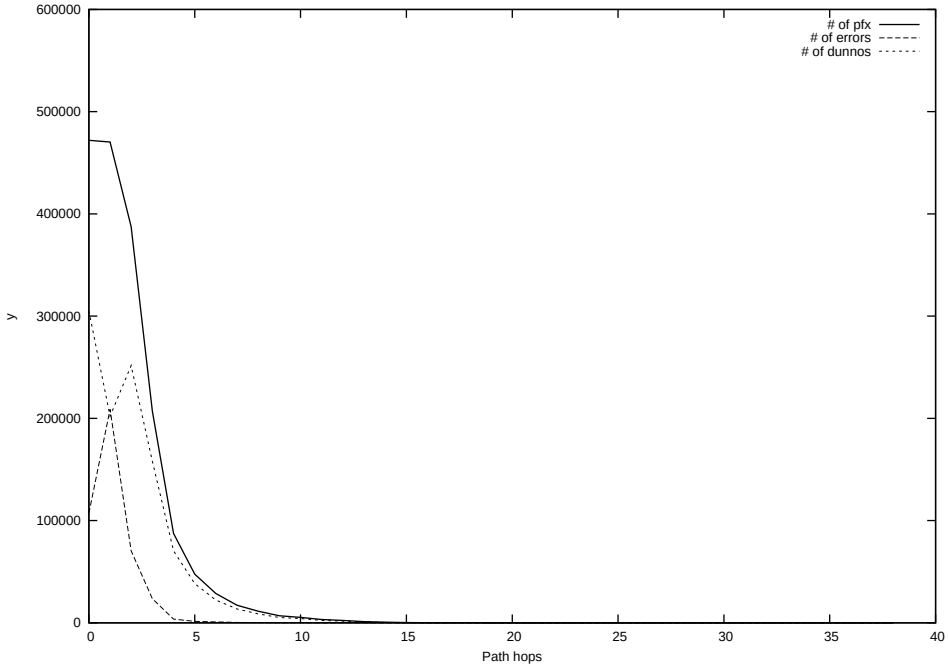
2013-09-29



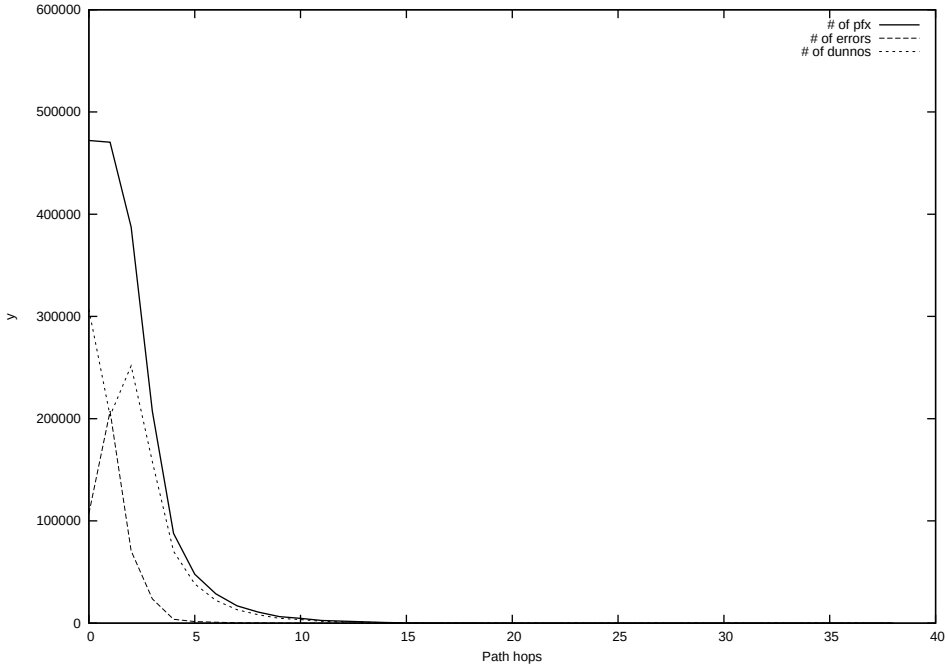
2013-09-30



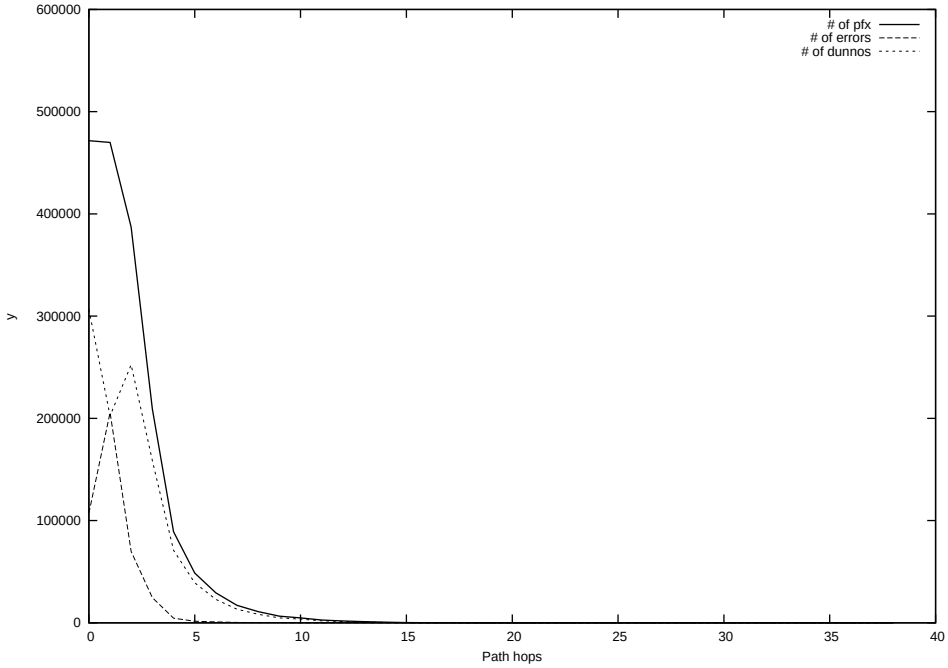
2013-10-01



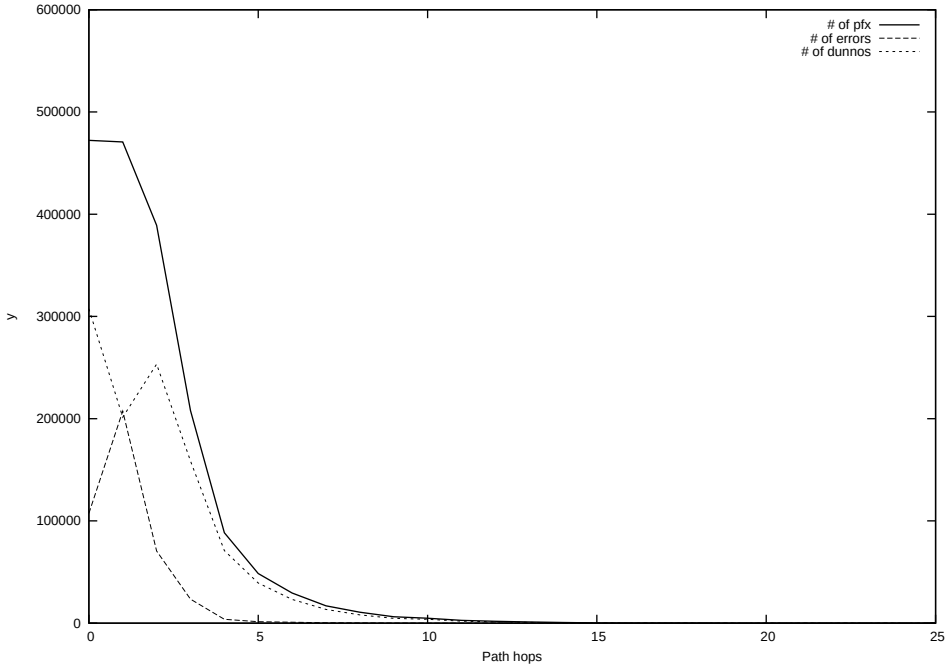
2013-10-02



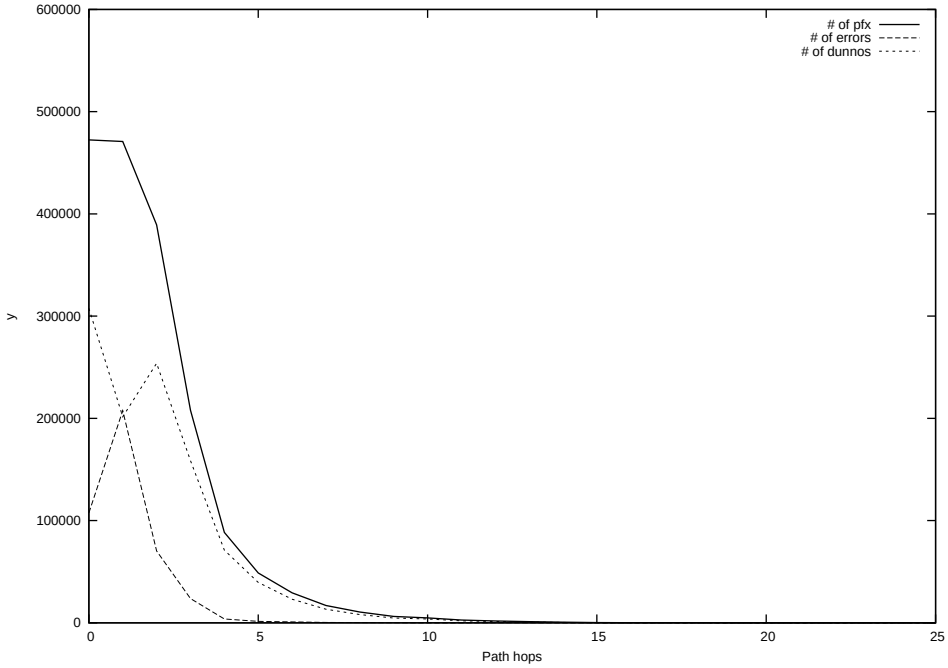
2013-10-03



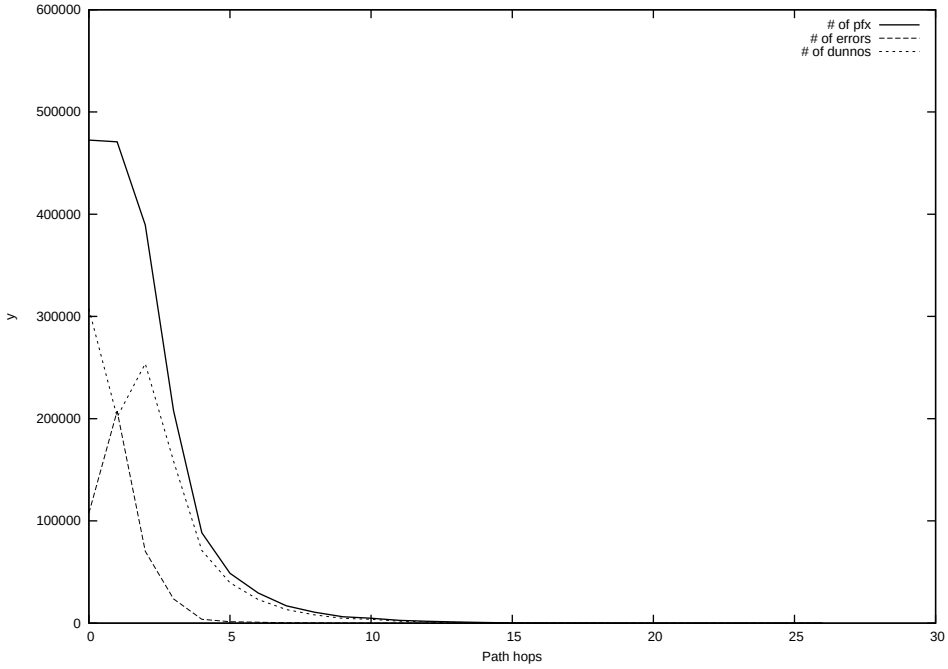
2013-10-04



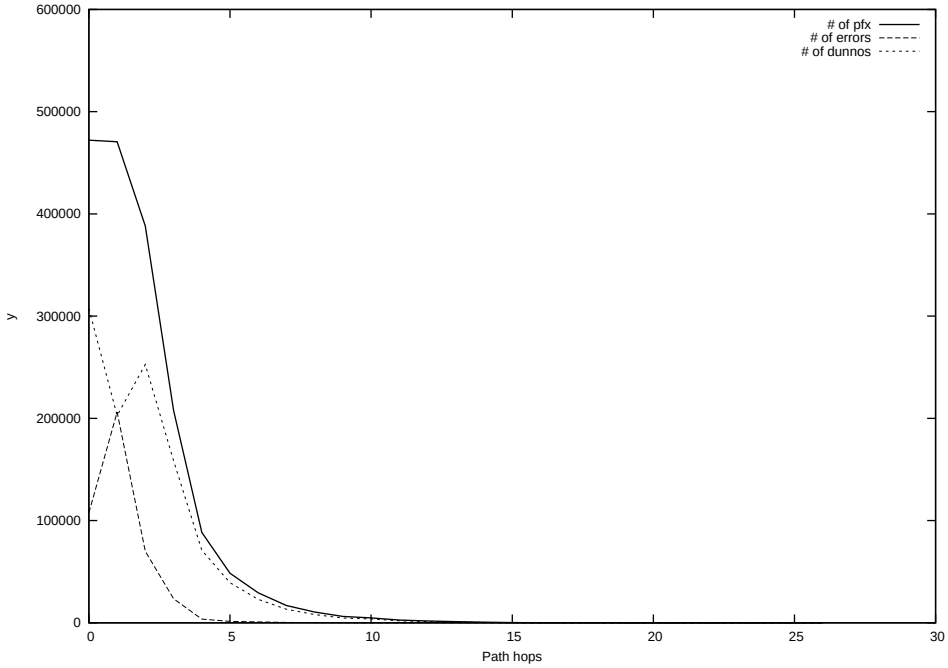
2013-10-05



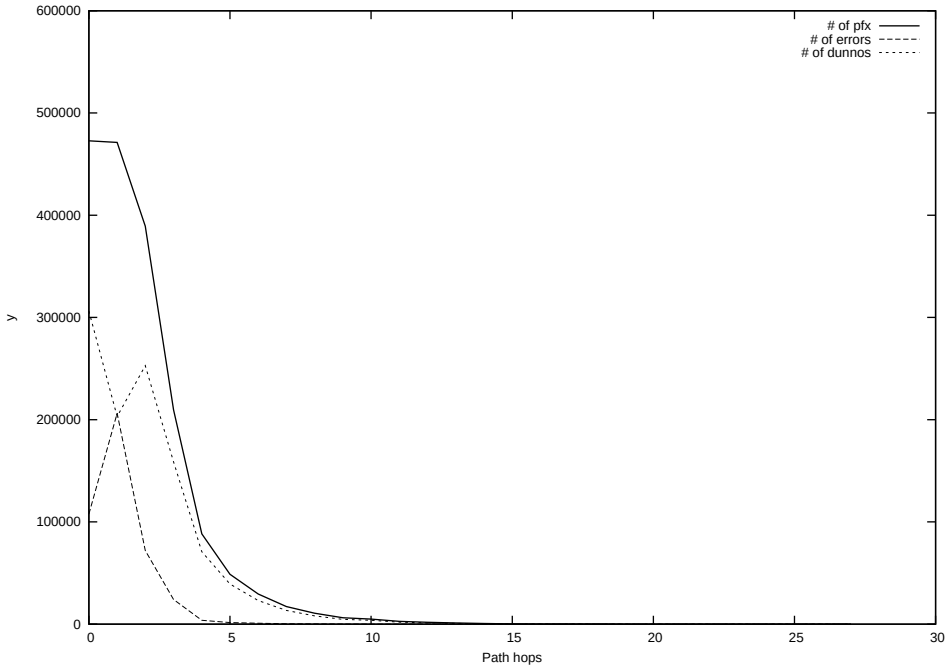
2013-10-06



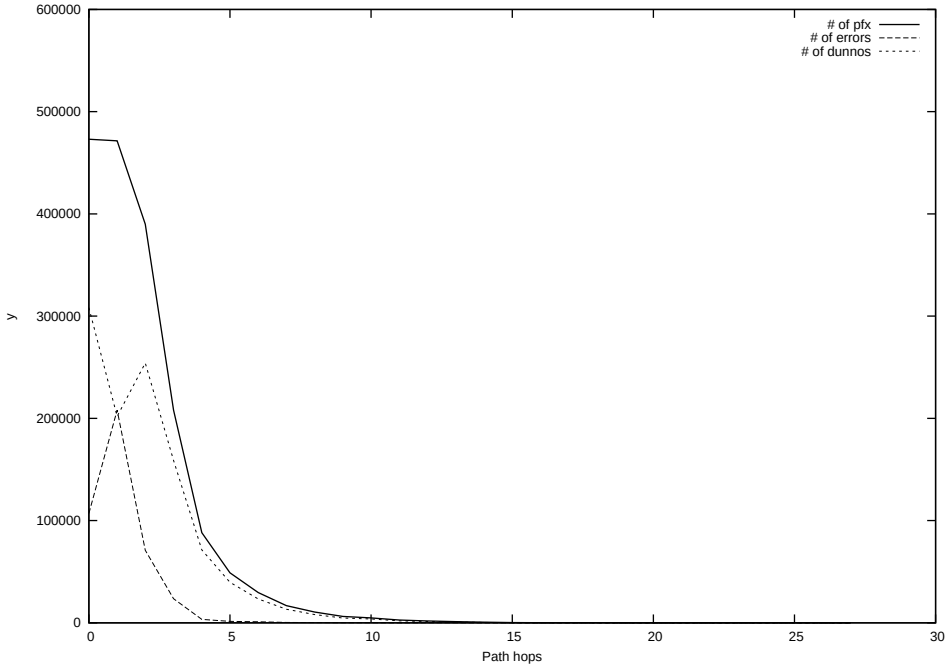
2013-10-07



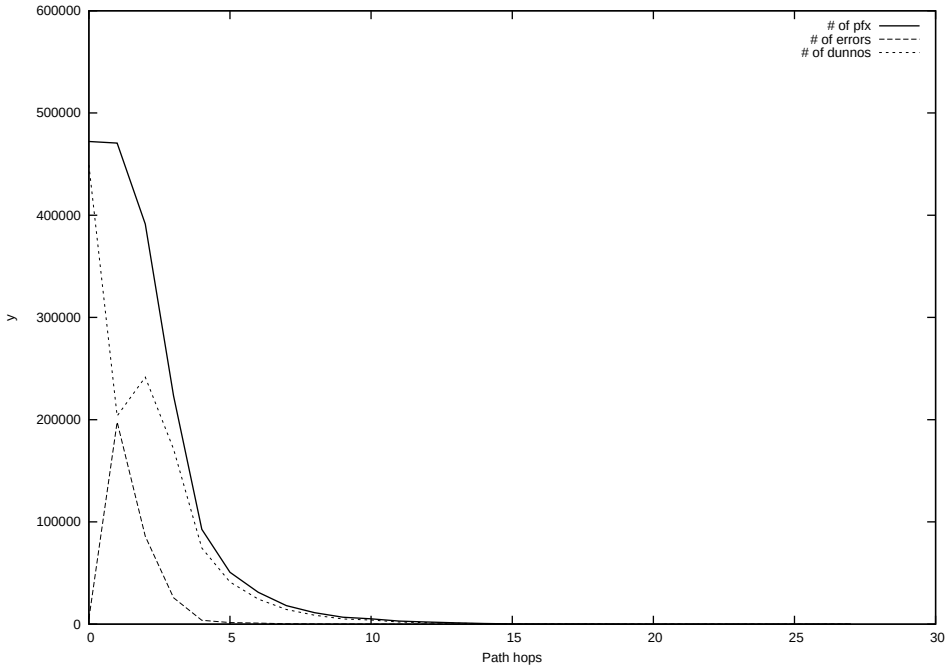
2013-10-08



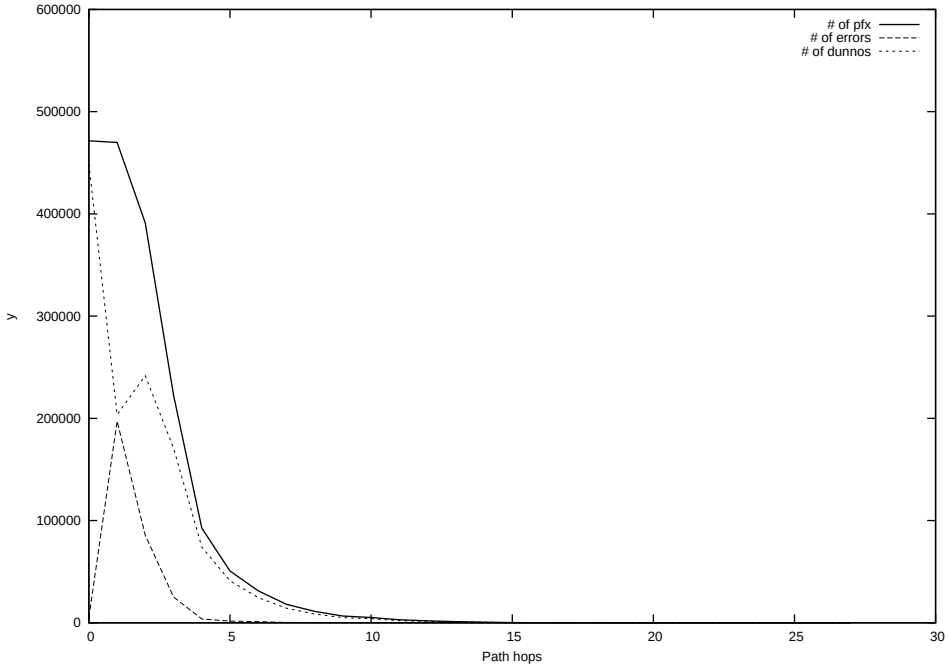
2013-10-09



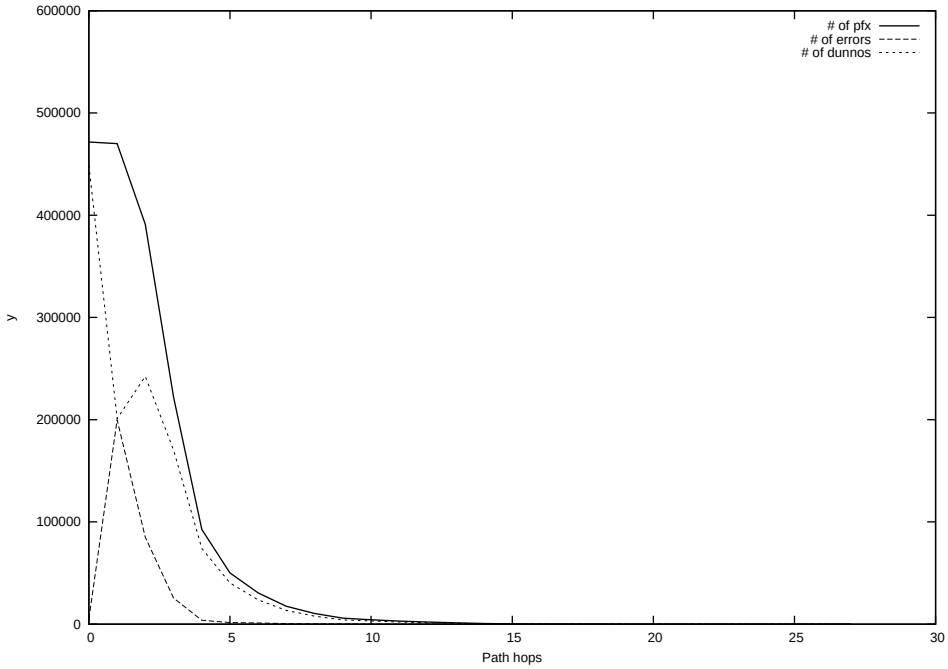
2013-10-10



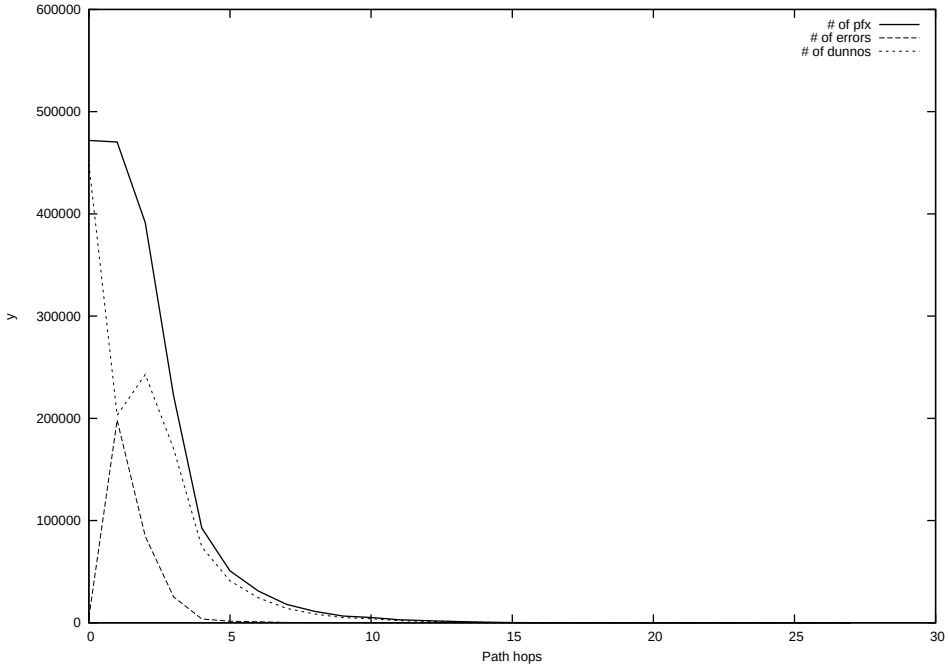
2013-10-11



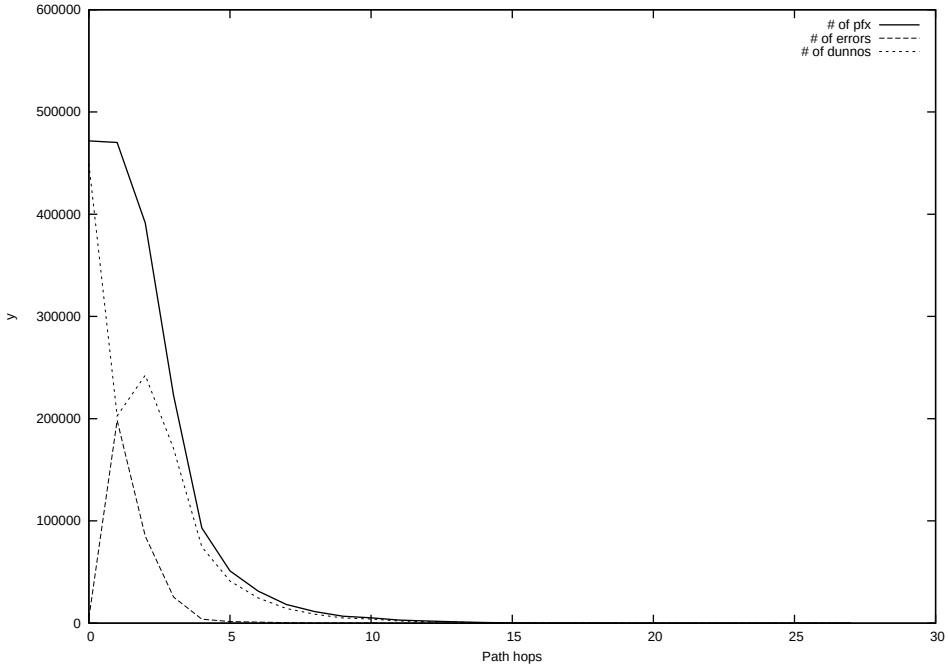
2013-10-12



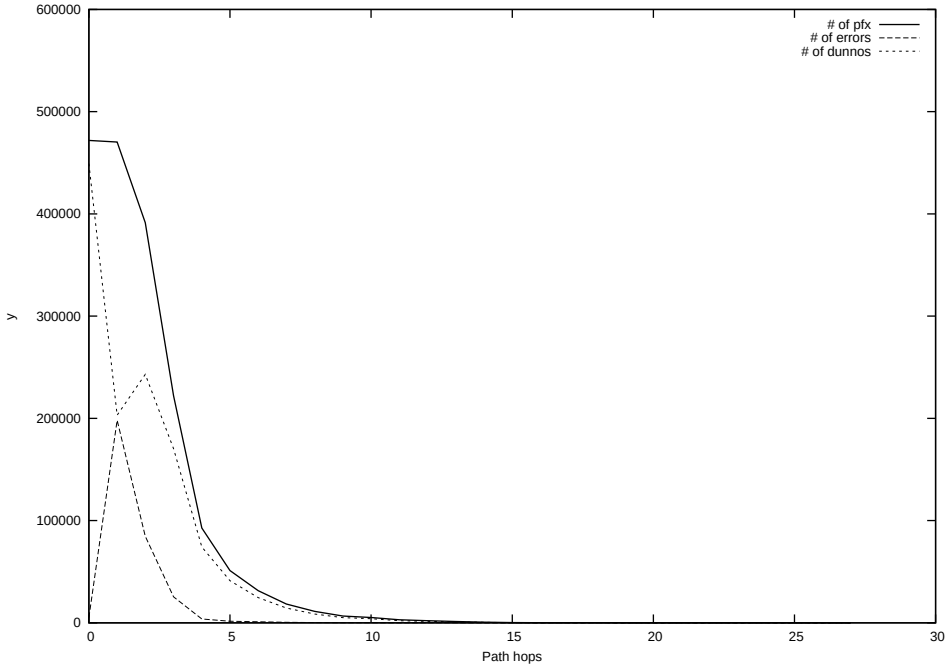
2013-10-13



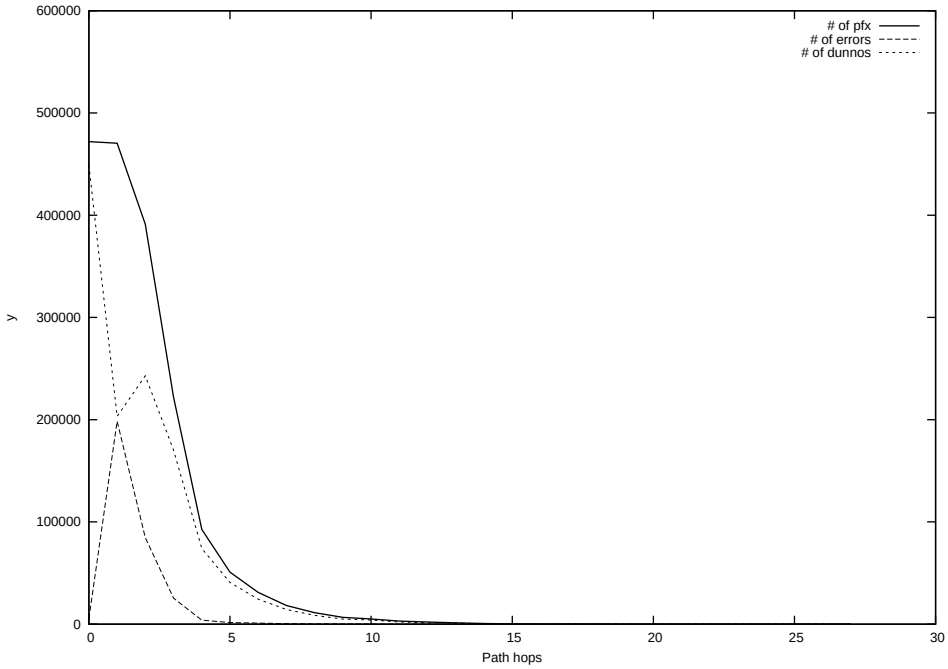
2013-10-14



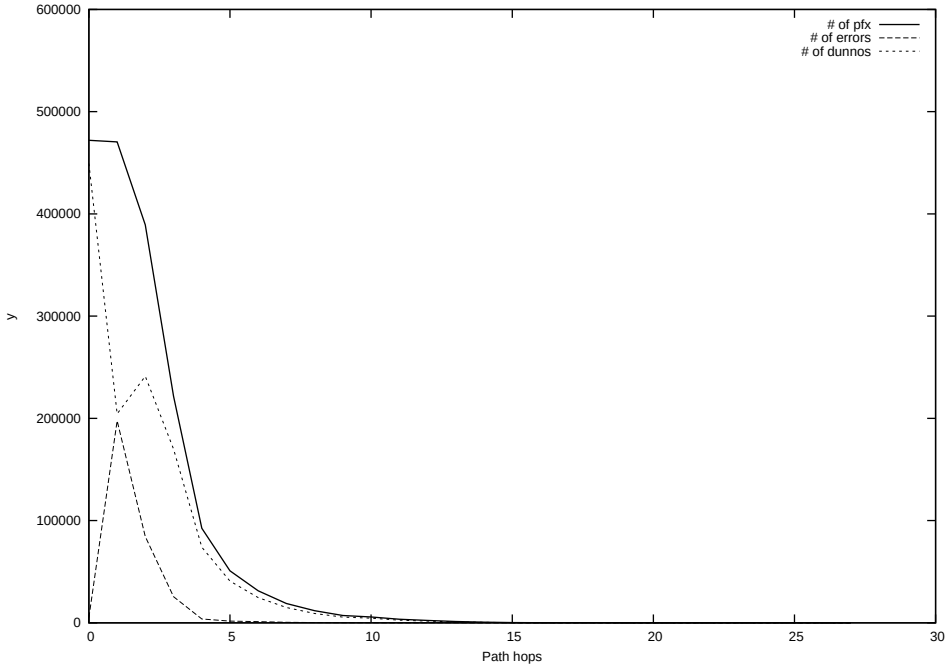
2013-10-15



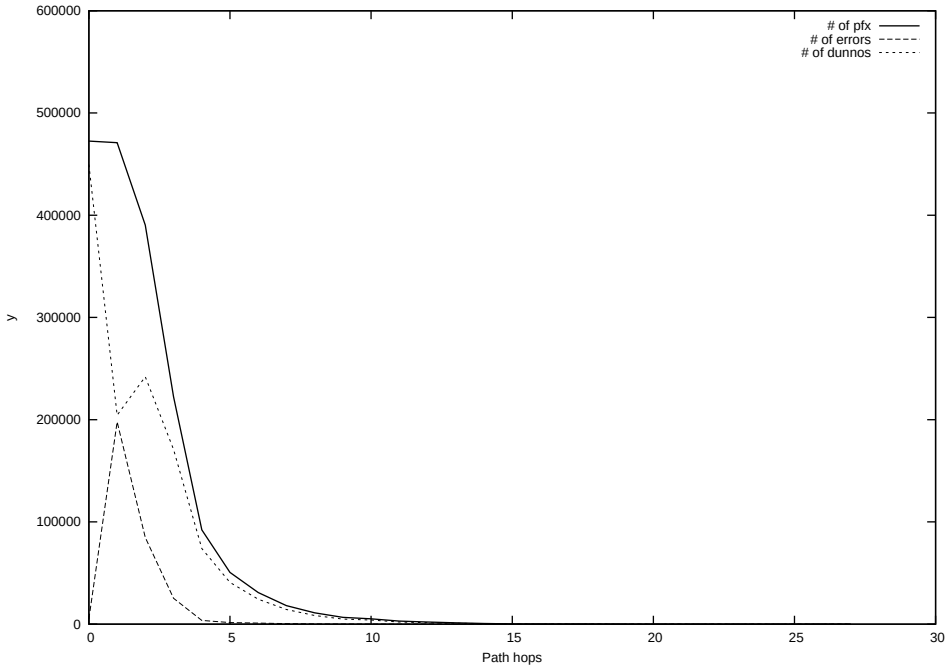
2013-10-16



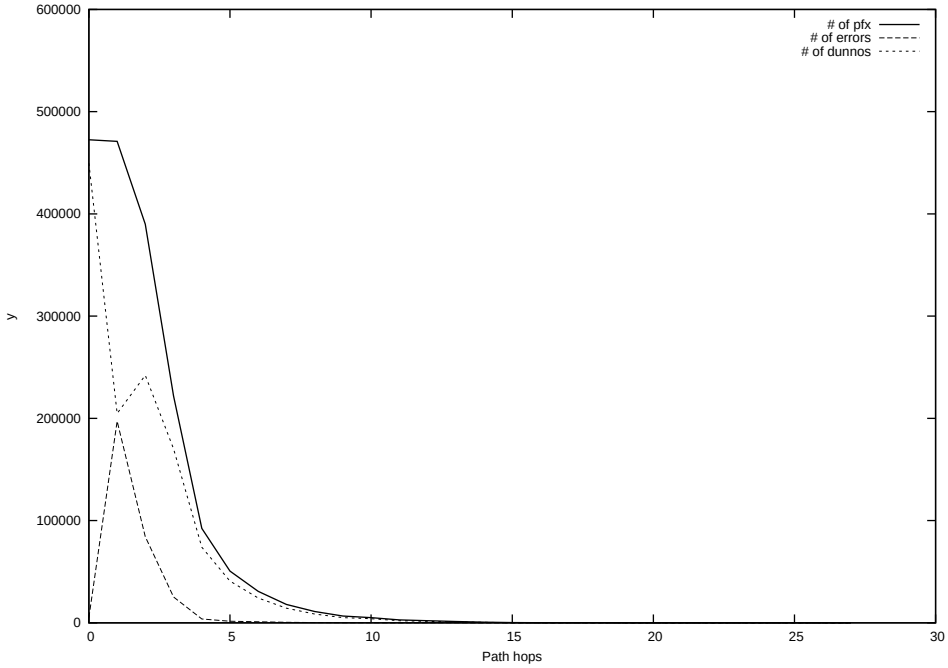
2013-10-17



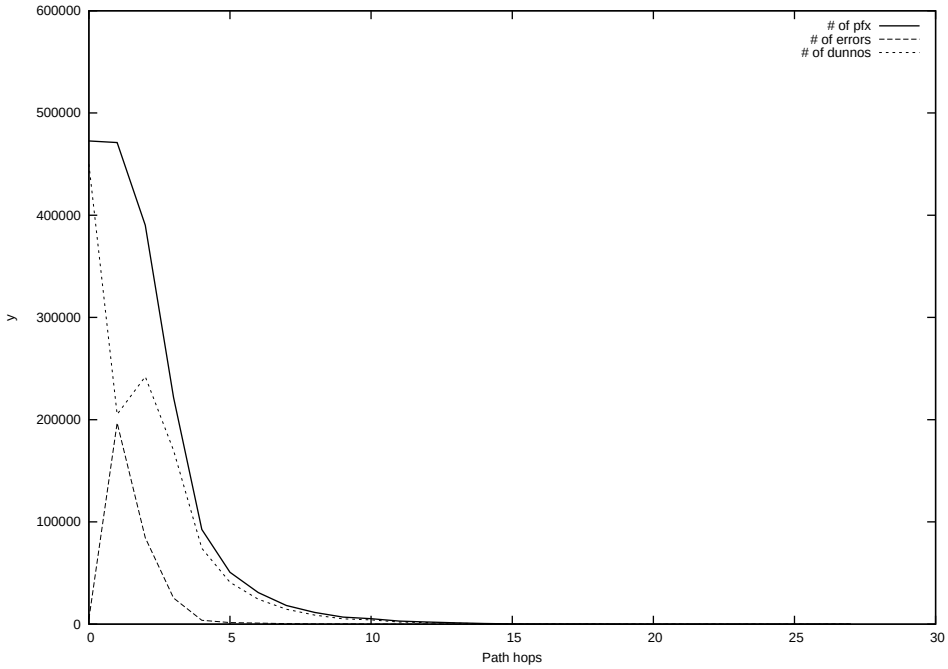
2013-10-18



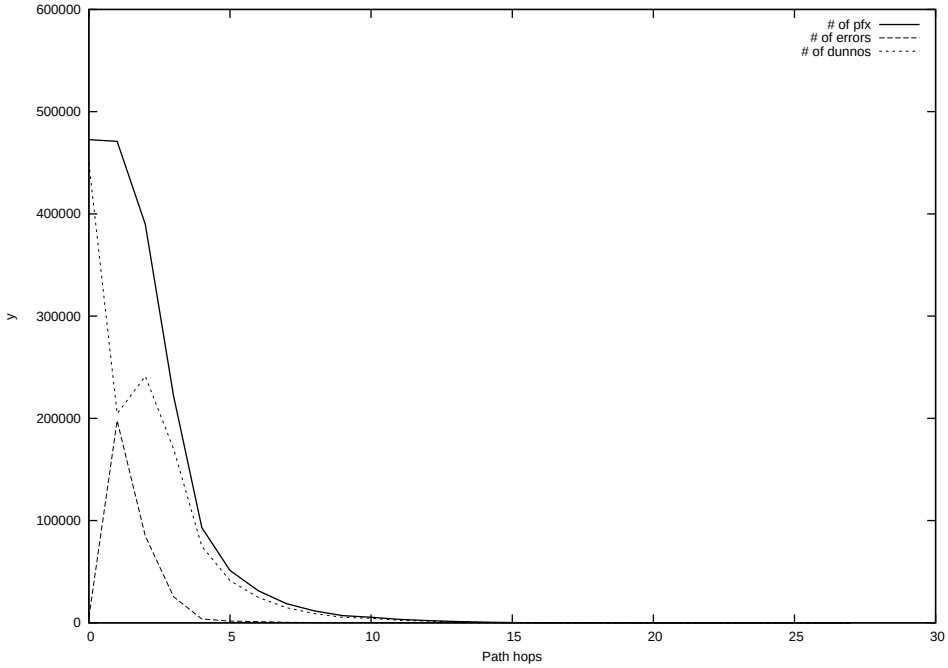
2013-10-19



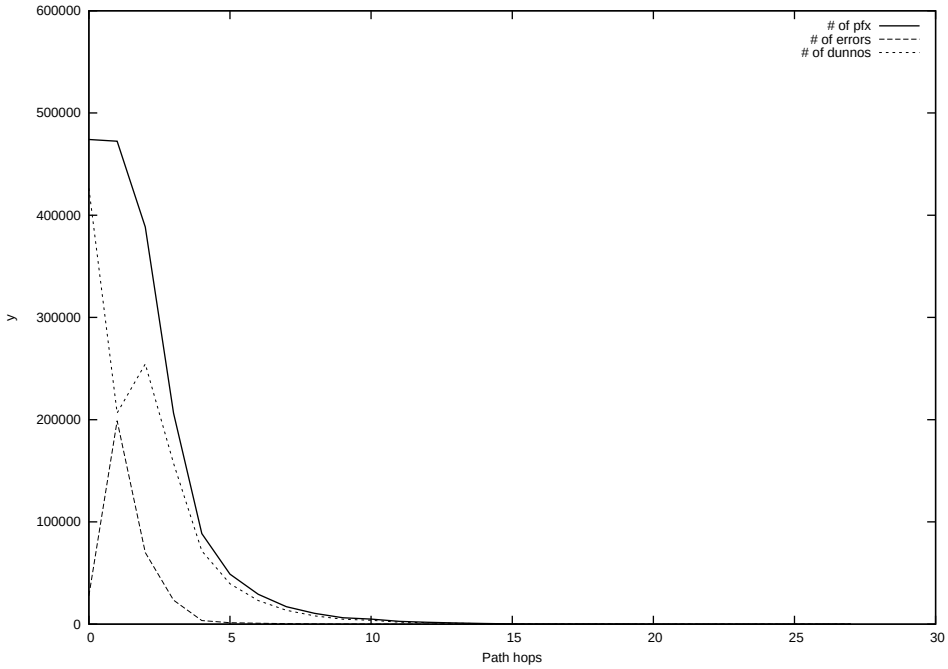
2013-10-20



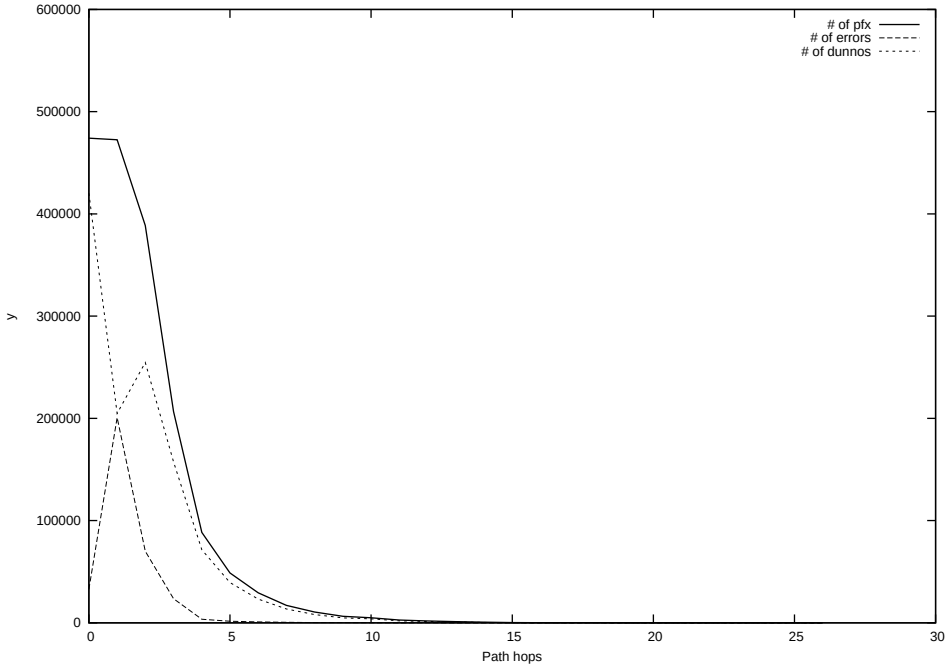
2013-10-21



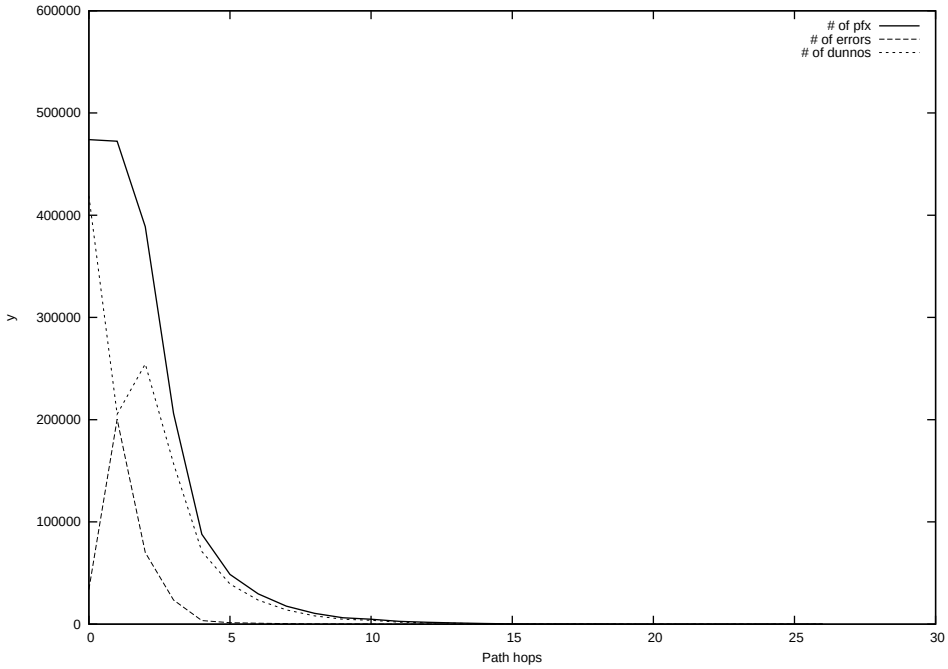
2013-10-22



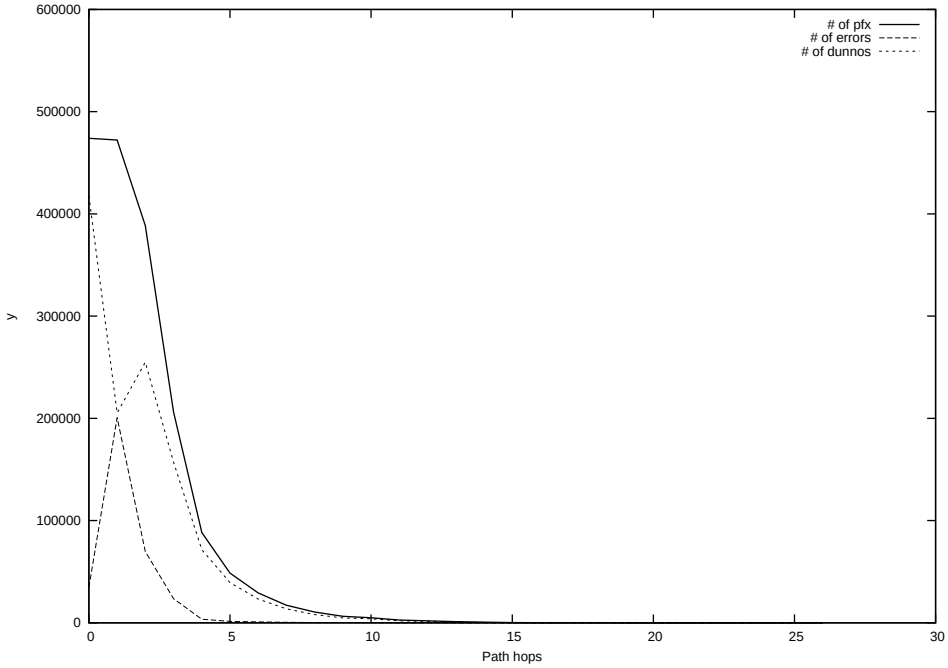
2013-10-23



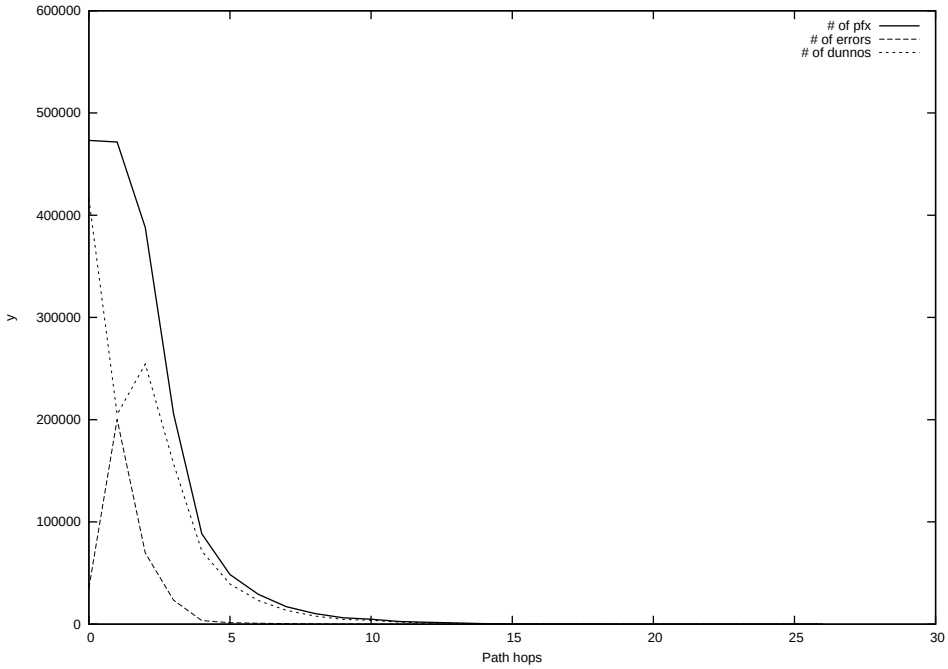
2013-10-24



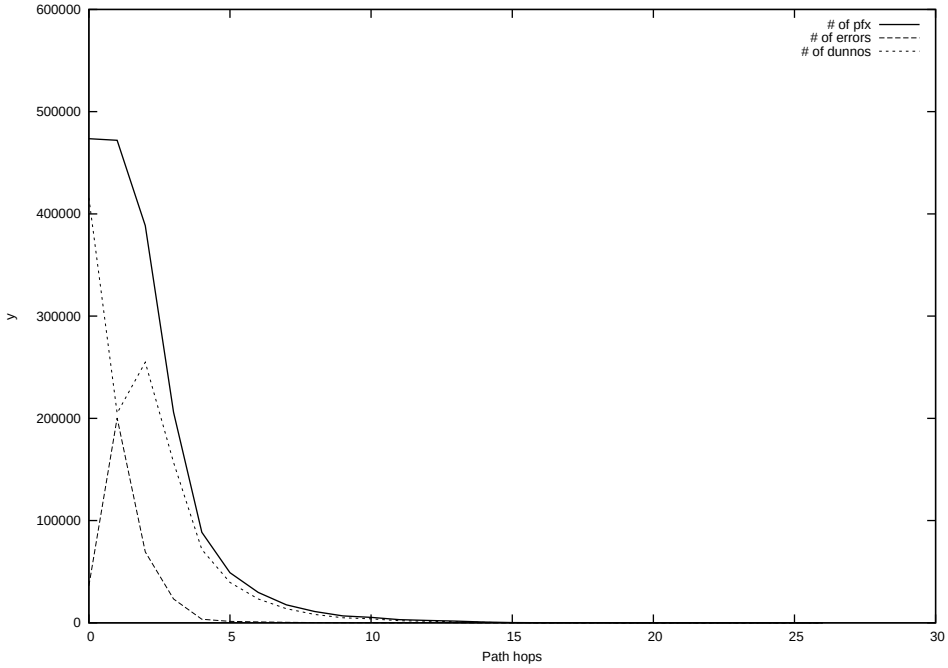
2013-10-25



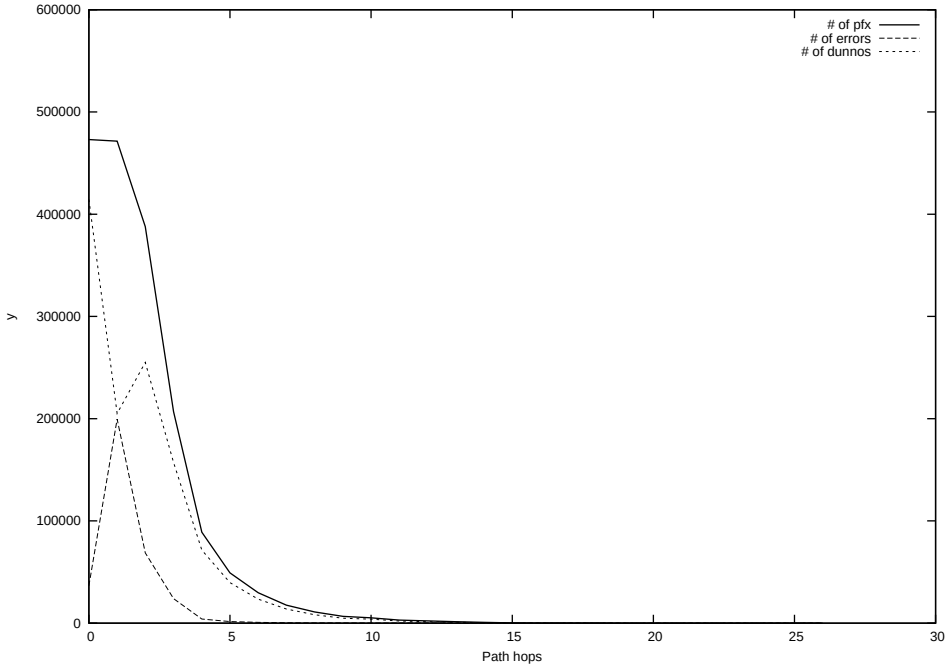
2013-10-26



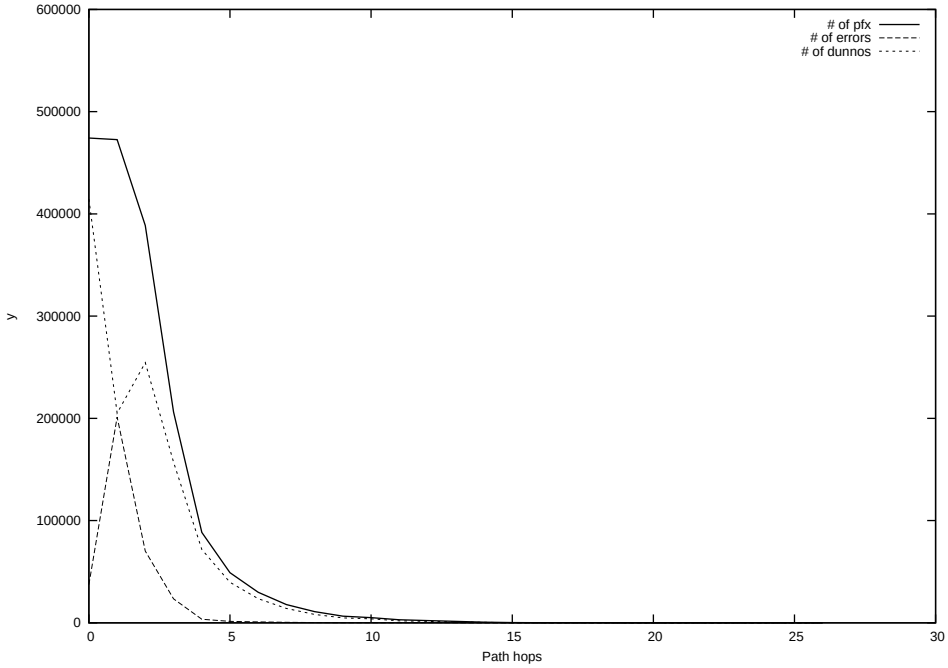
2013-10-27



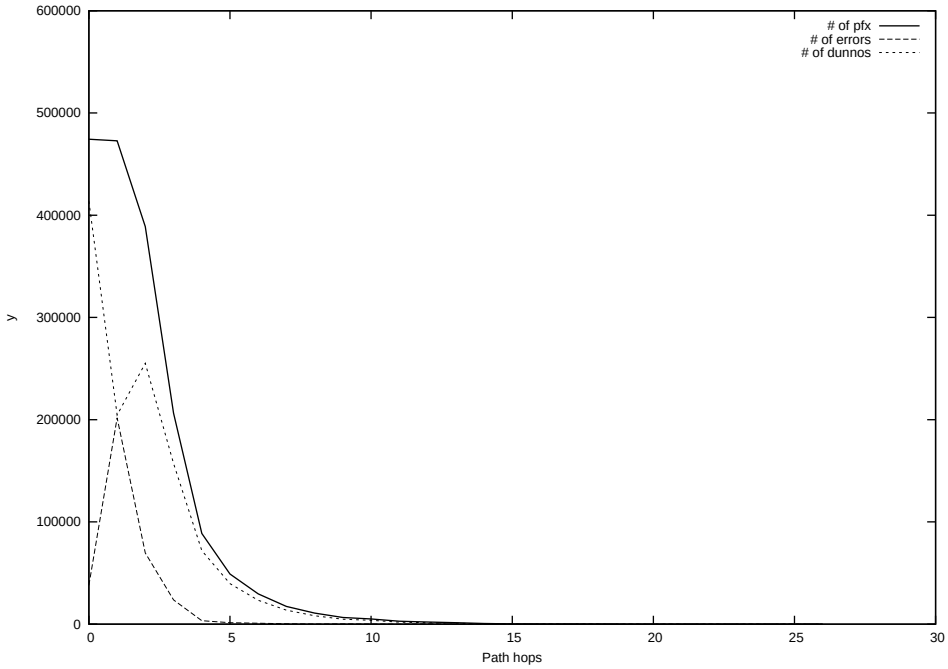
2013-10-28



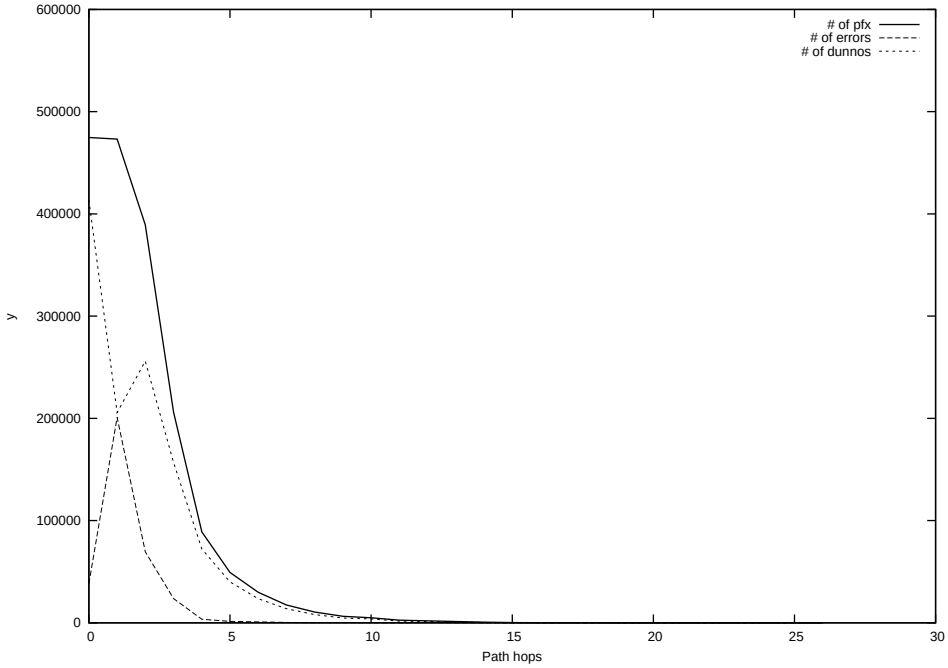
2013-10-29



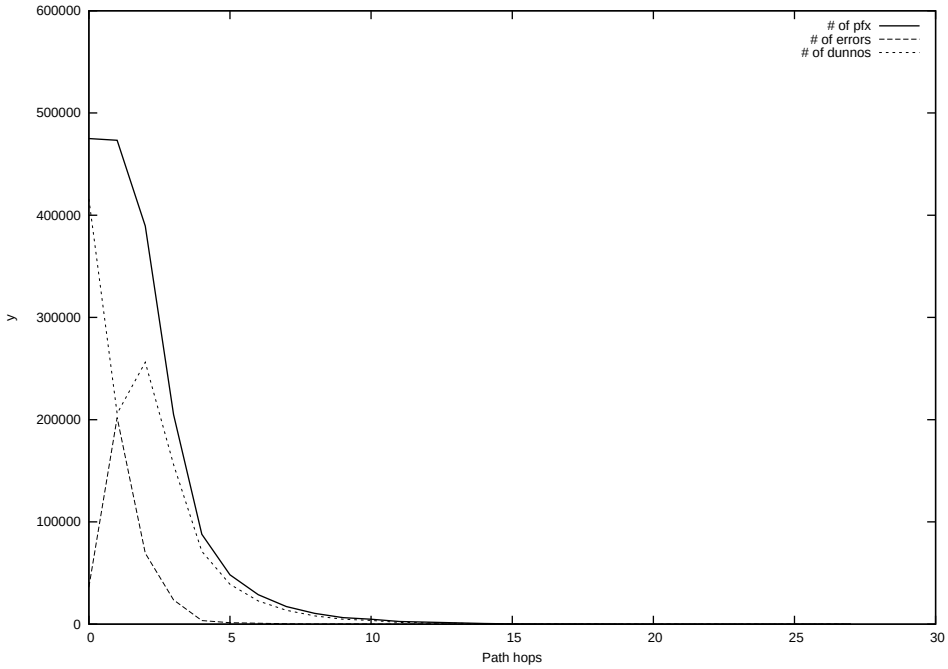
2013-10-30



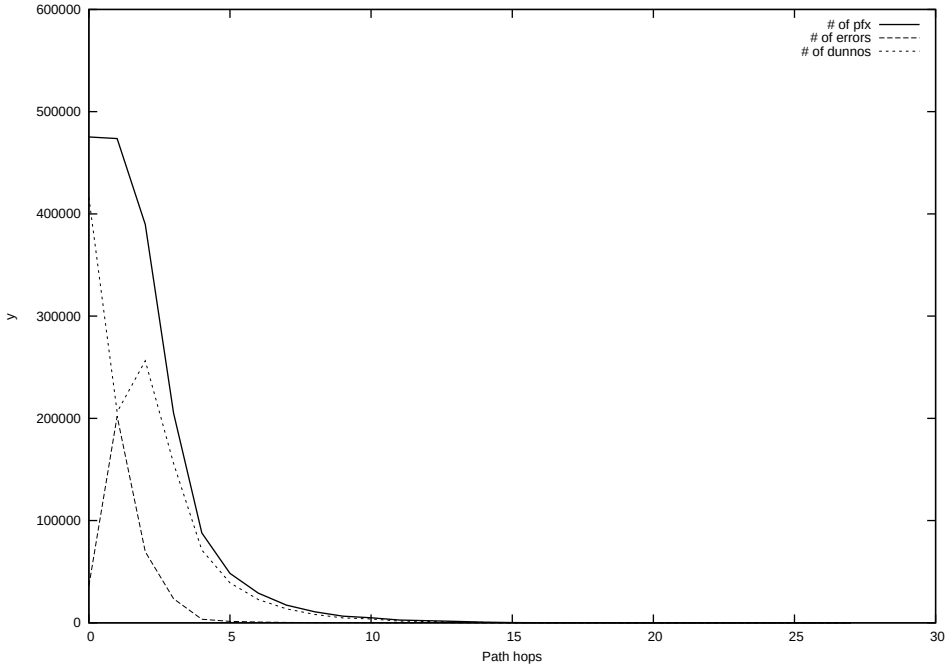
2013-10-31



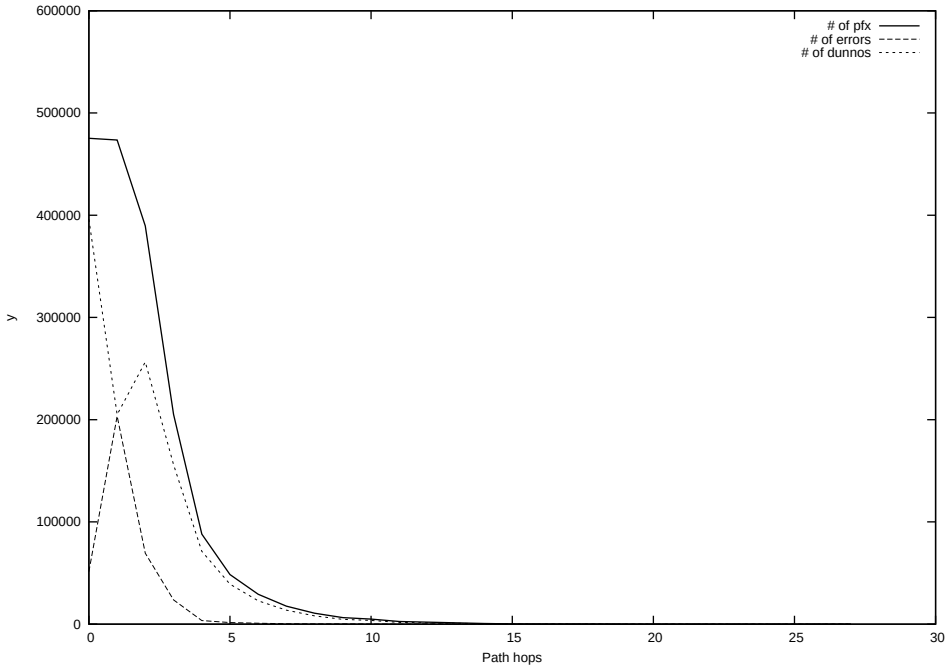
2013-11-01



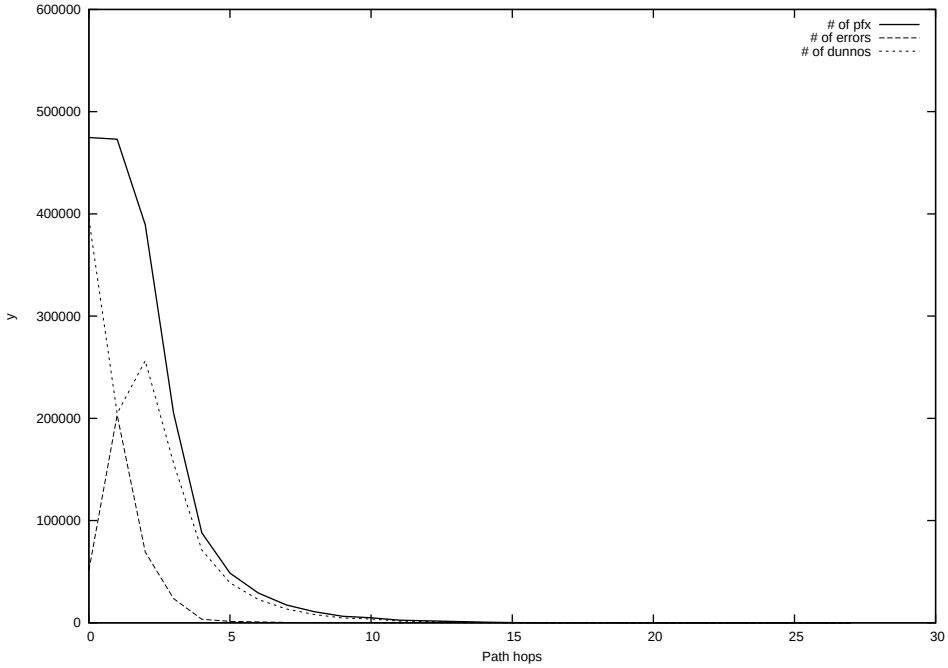
2013-11-02



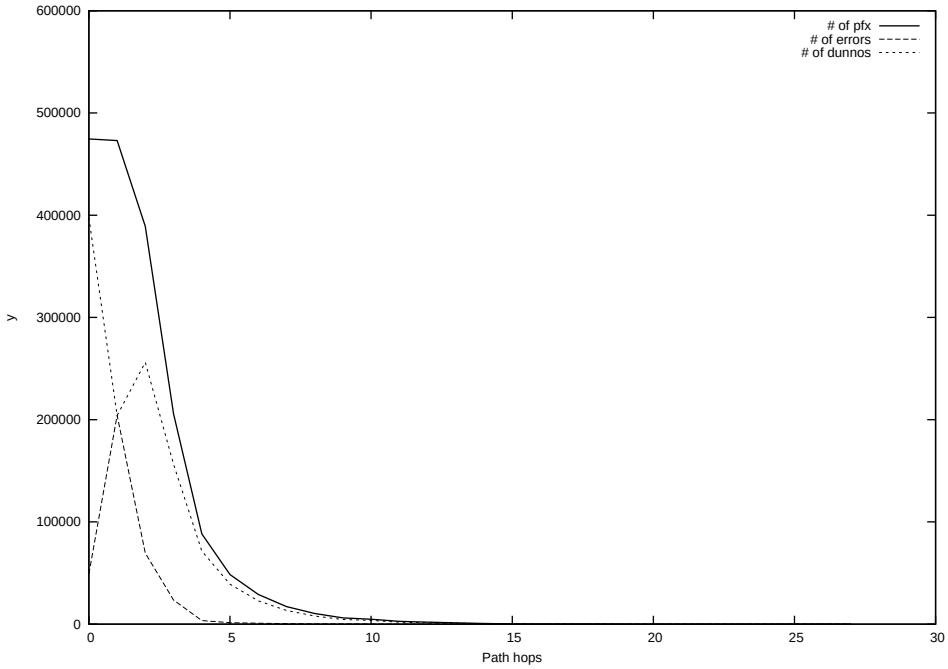
2013-11-03



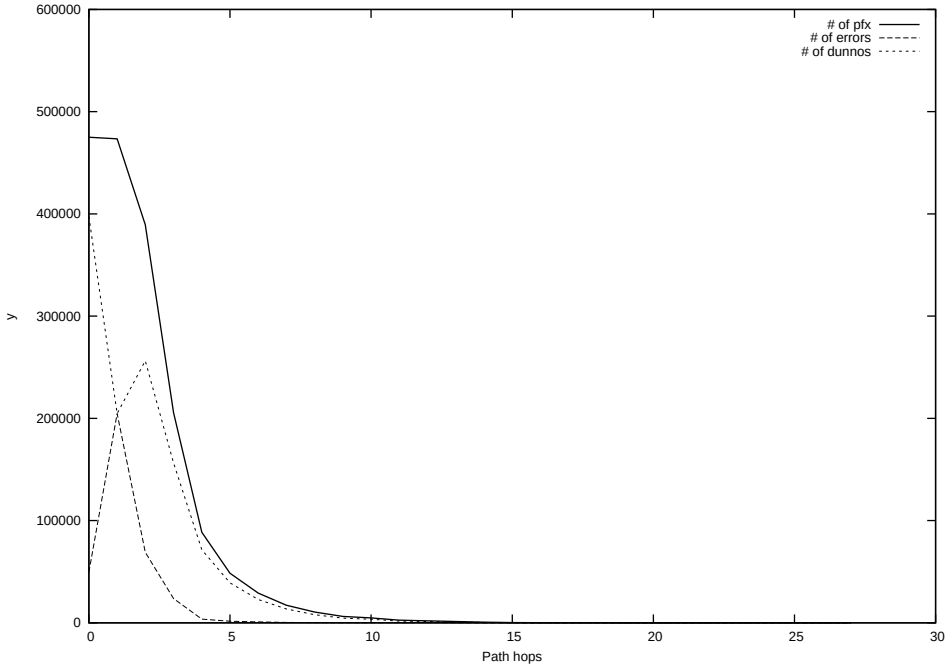
2013-11-04



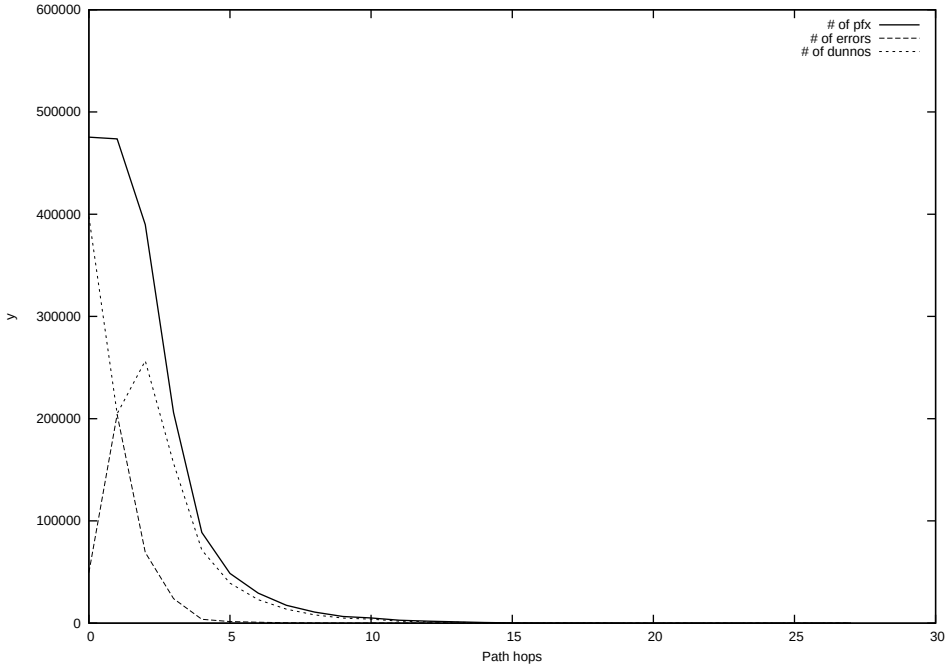
2013-11-05



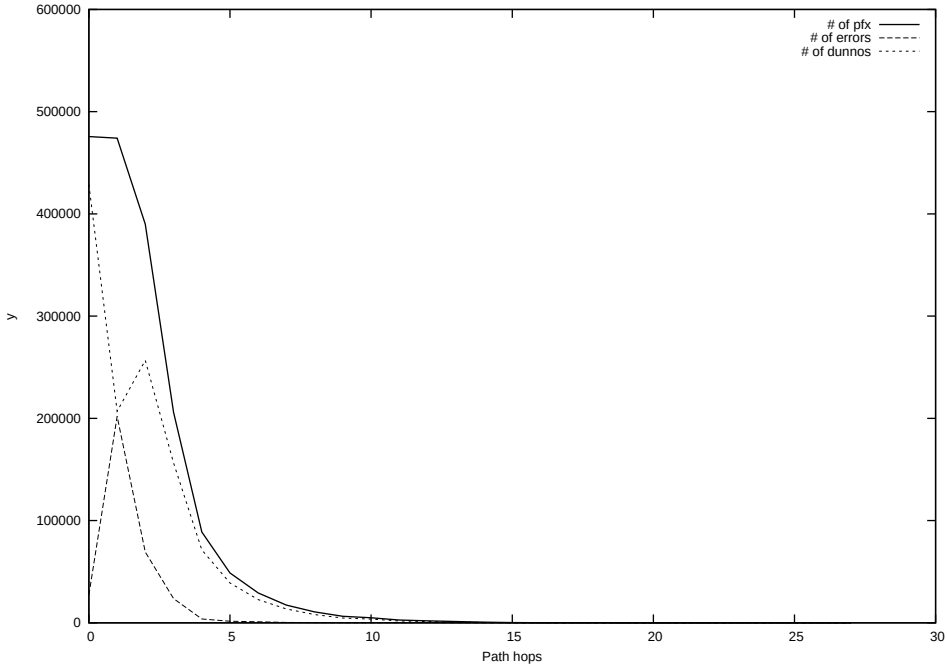
2013-11-06



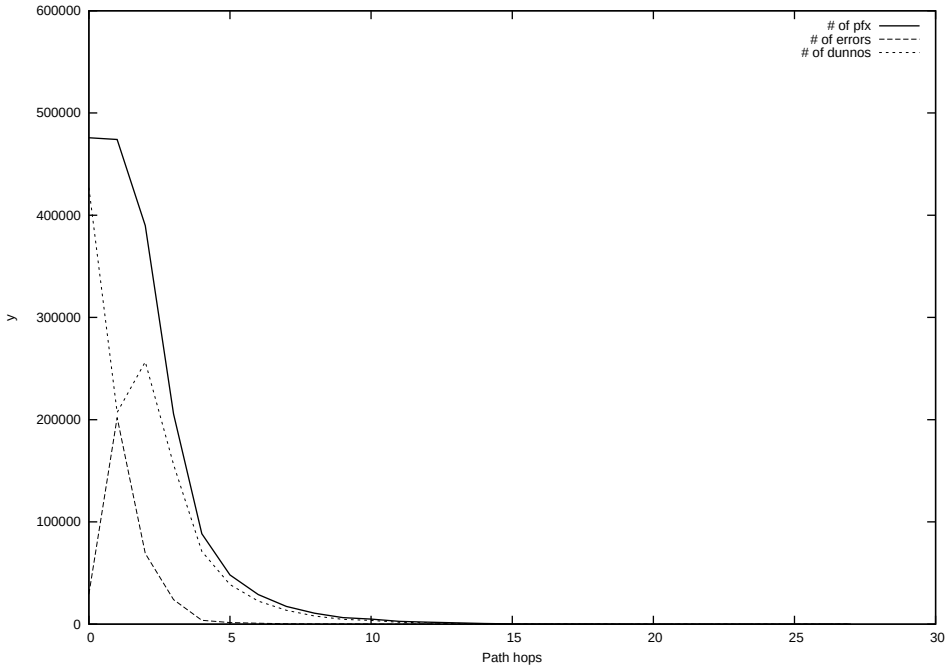
2013-11-07



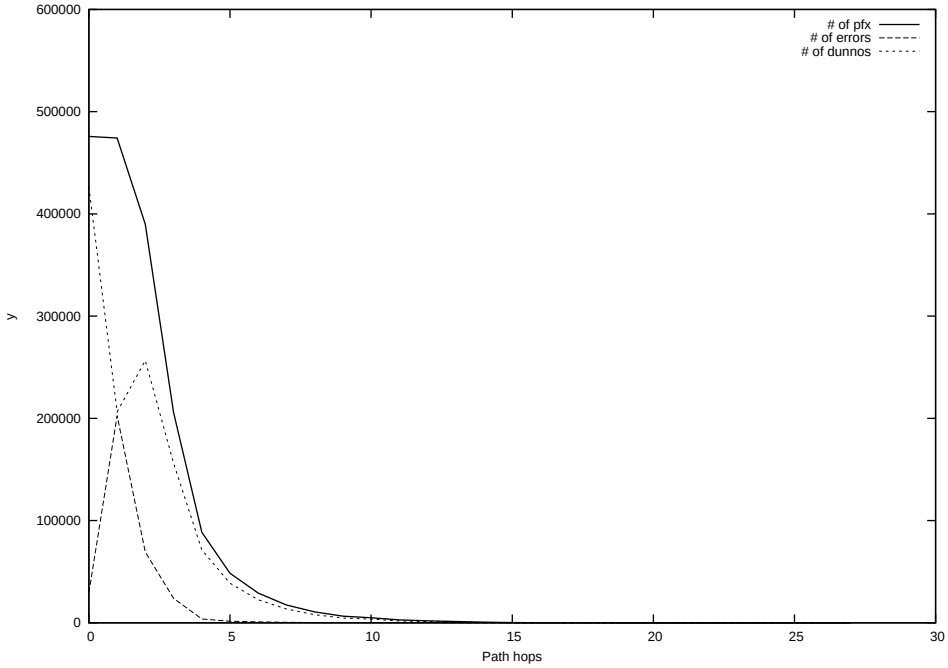
2013-11-08



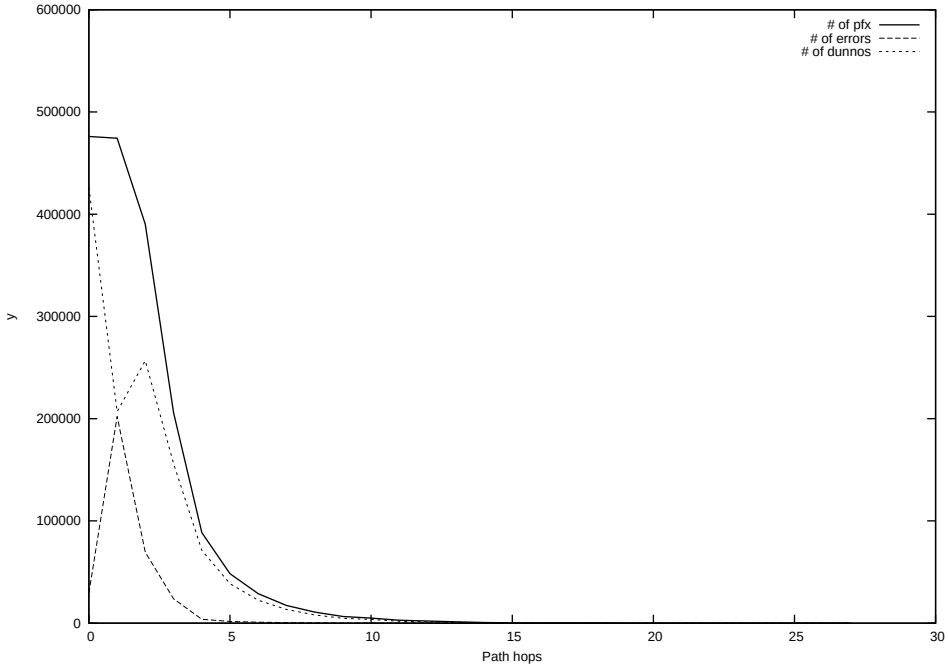
2013-11-09



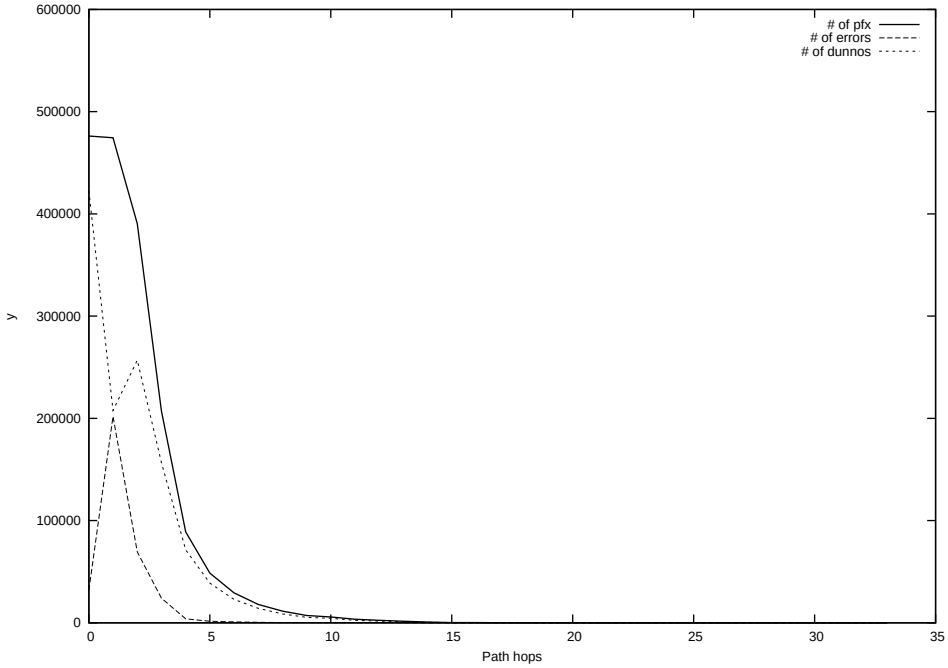
2013-11-10



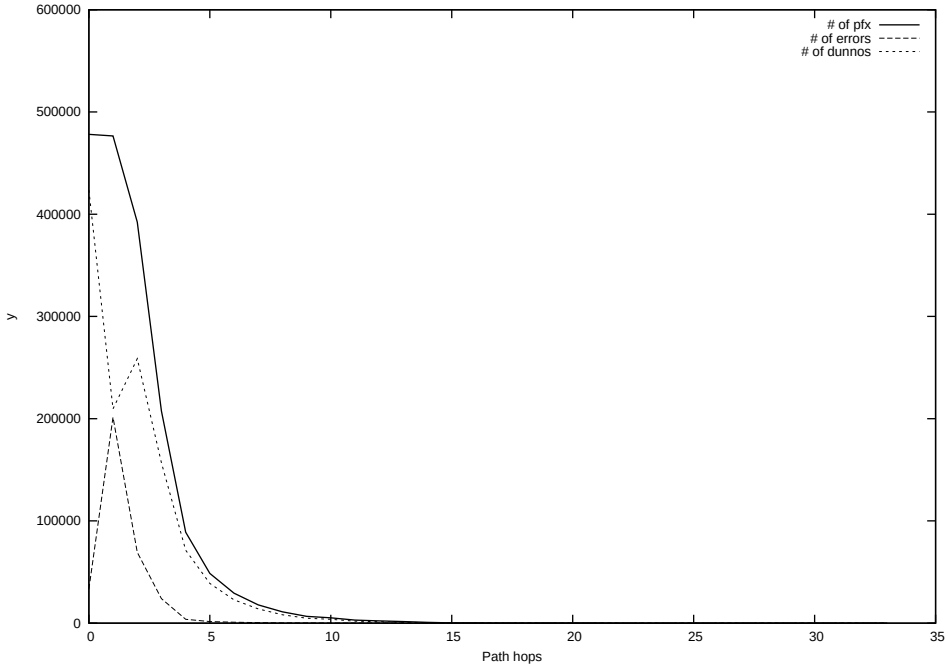
2013-11-11



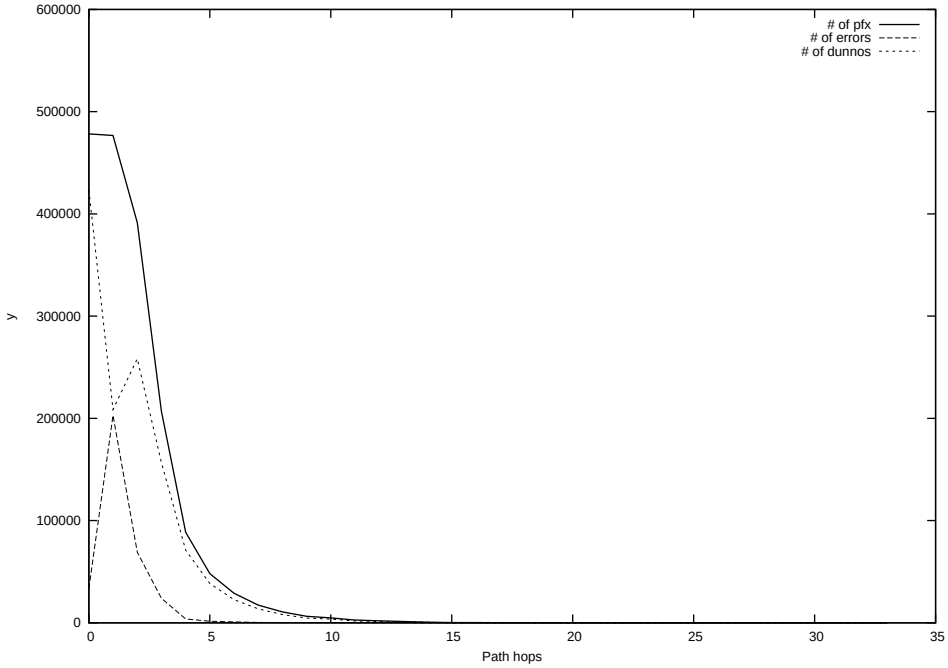
2013-11-12



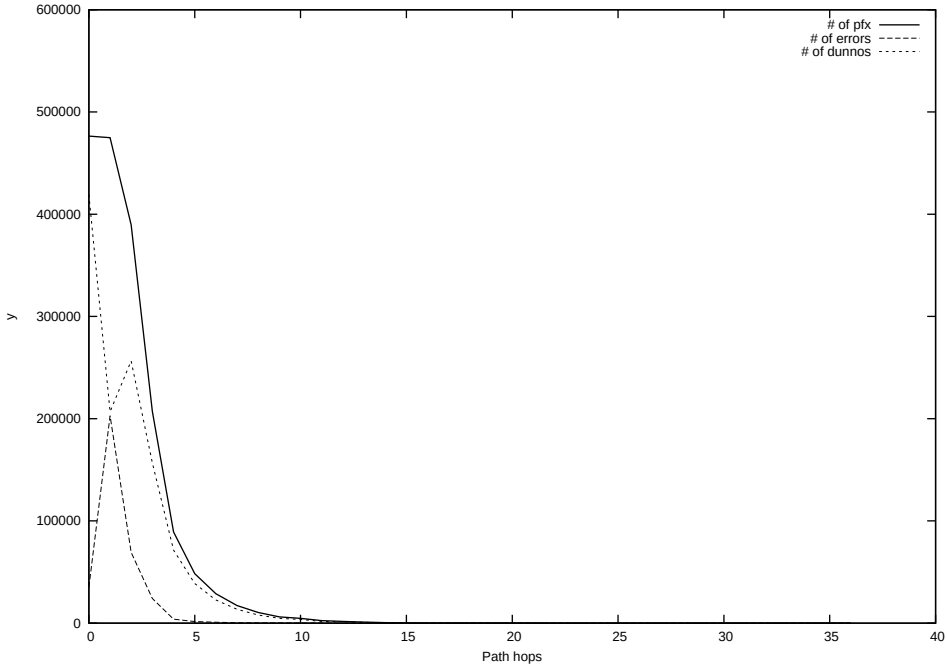
2013-11-13



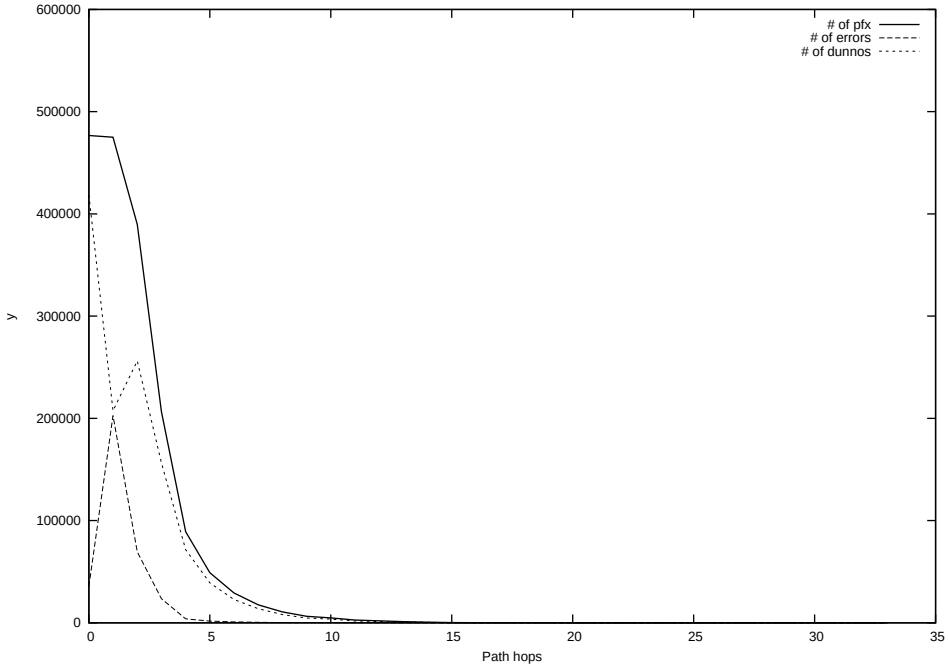
2013-11-14



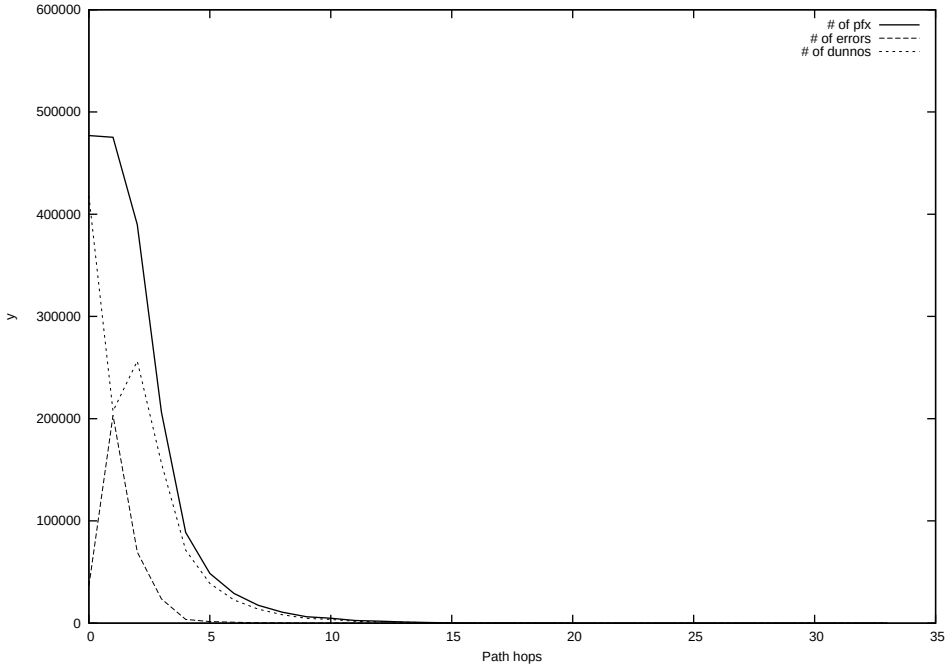
2013-11-15



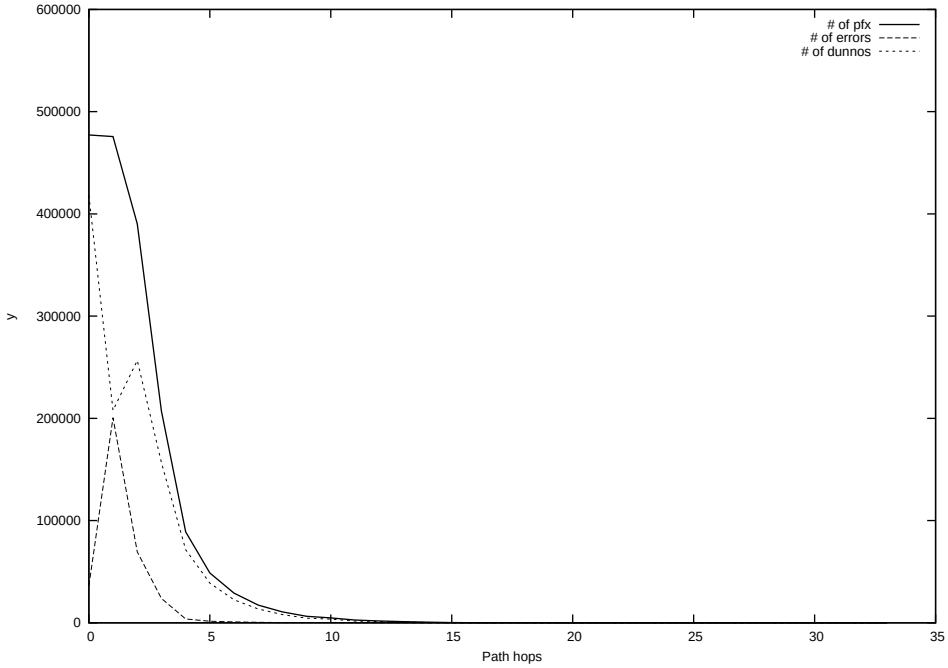
2013-11-16



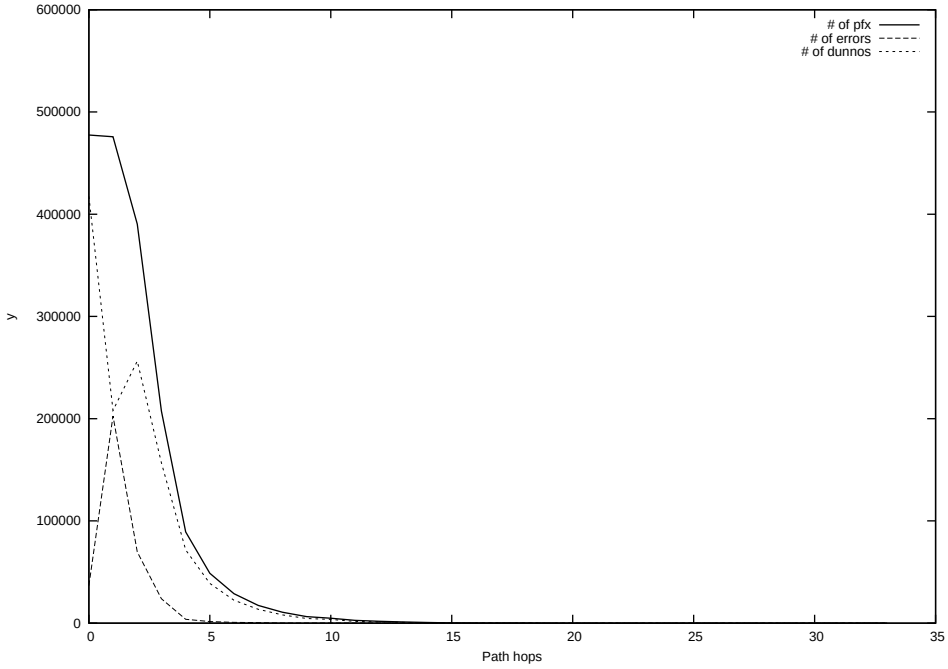
2013-11-17



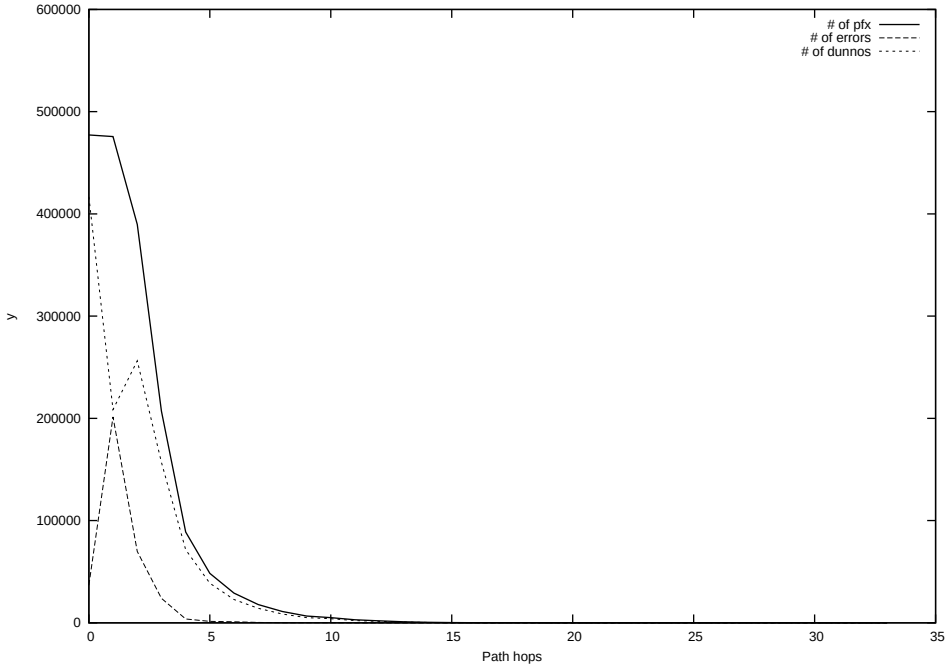
2013-11-18



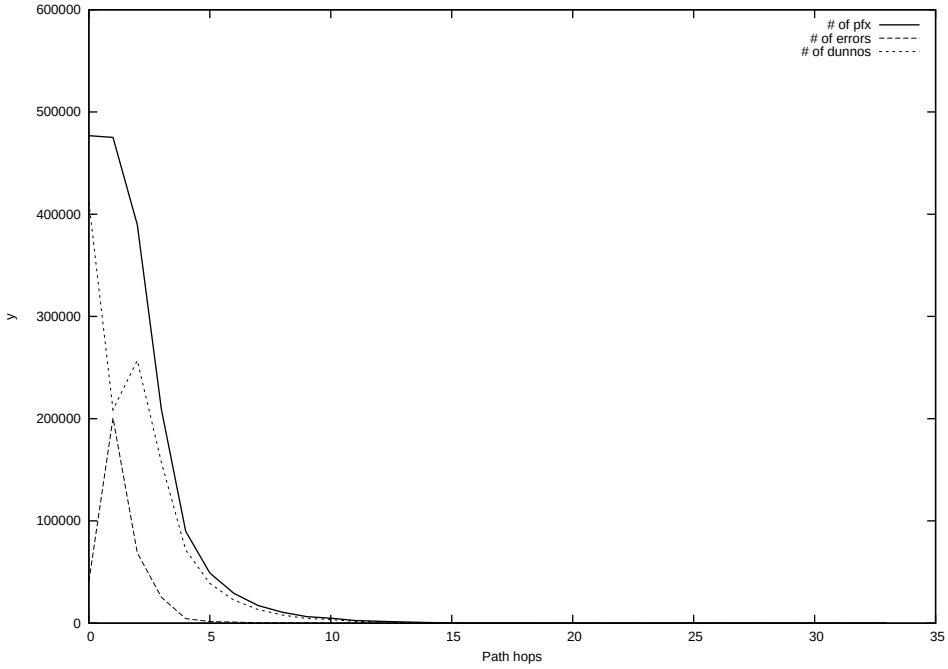
2013-11-19



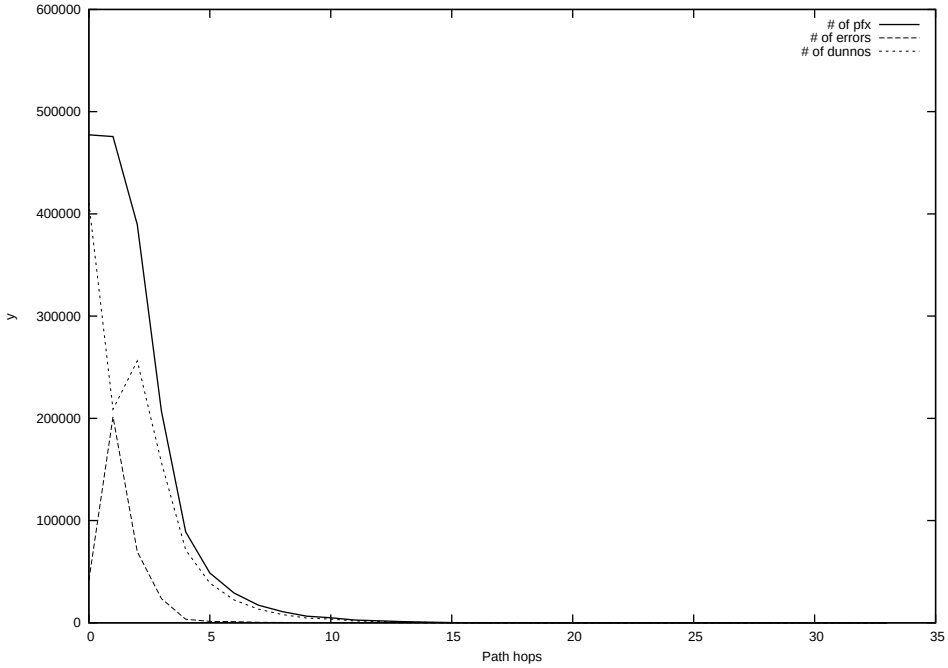
2013-11-20



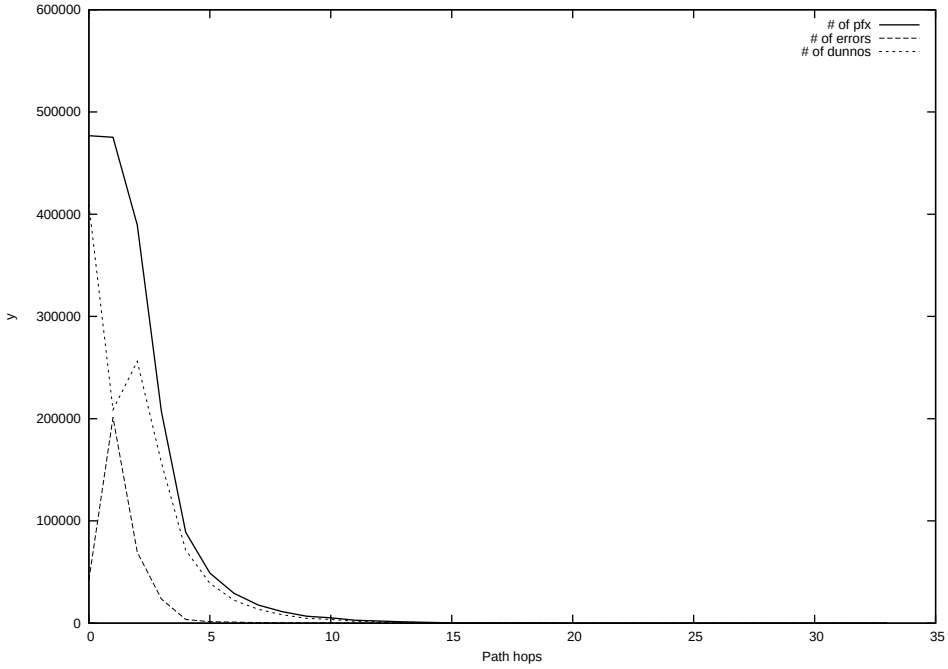
2013-11-21



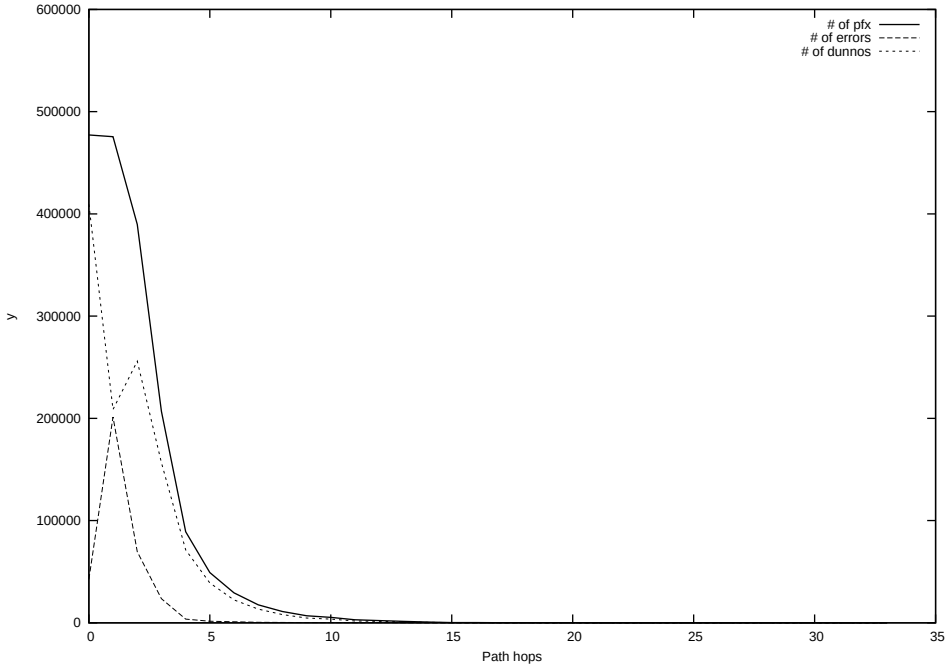
2013-11-22



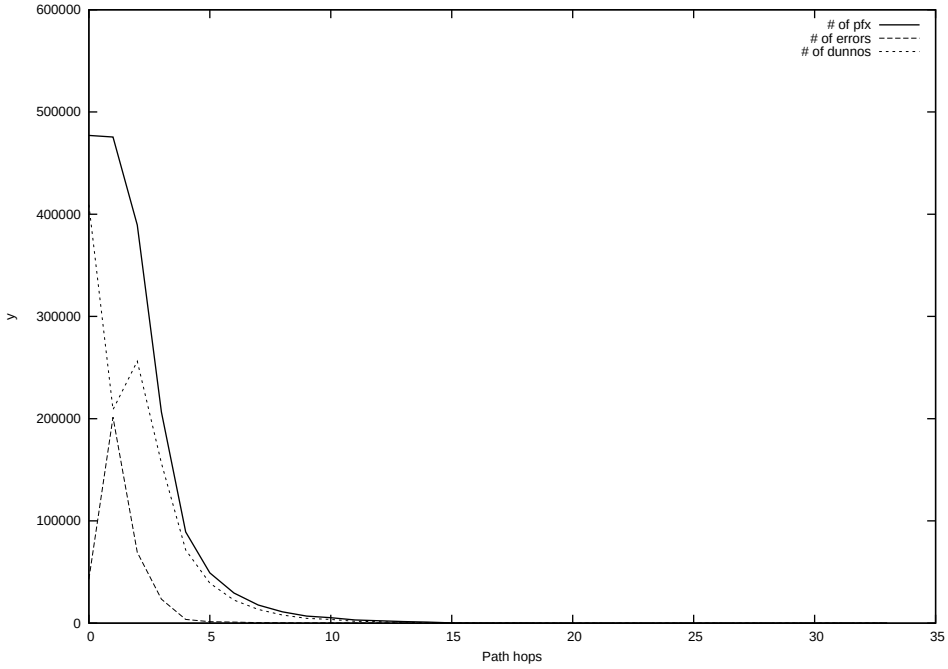
2013-11-23



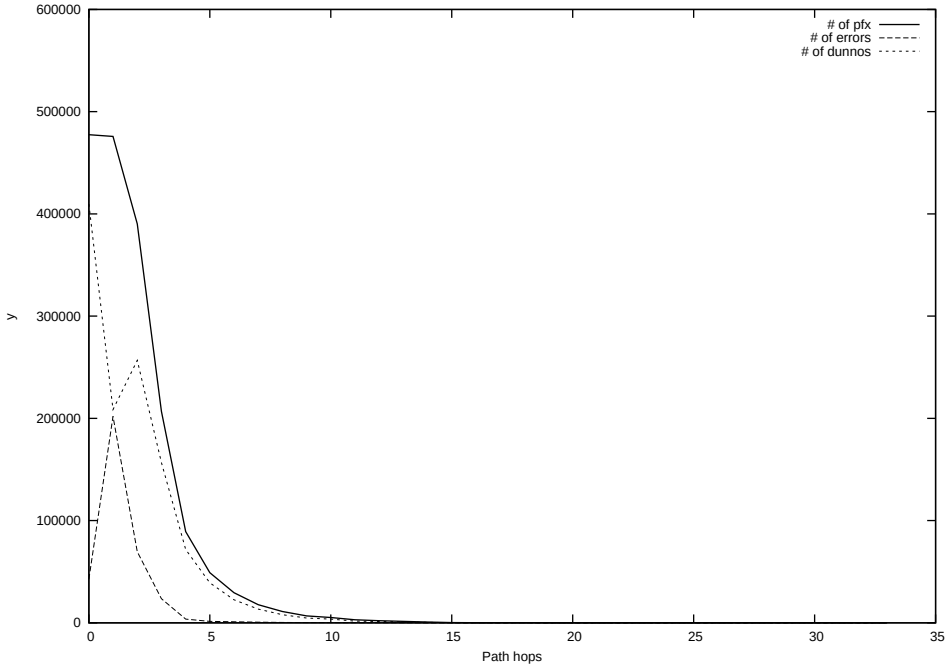
2013-11-24



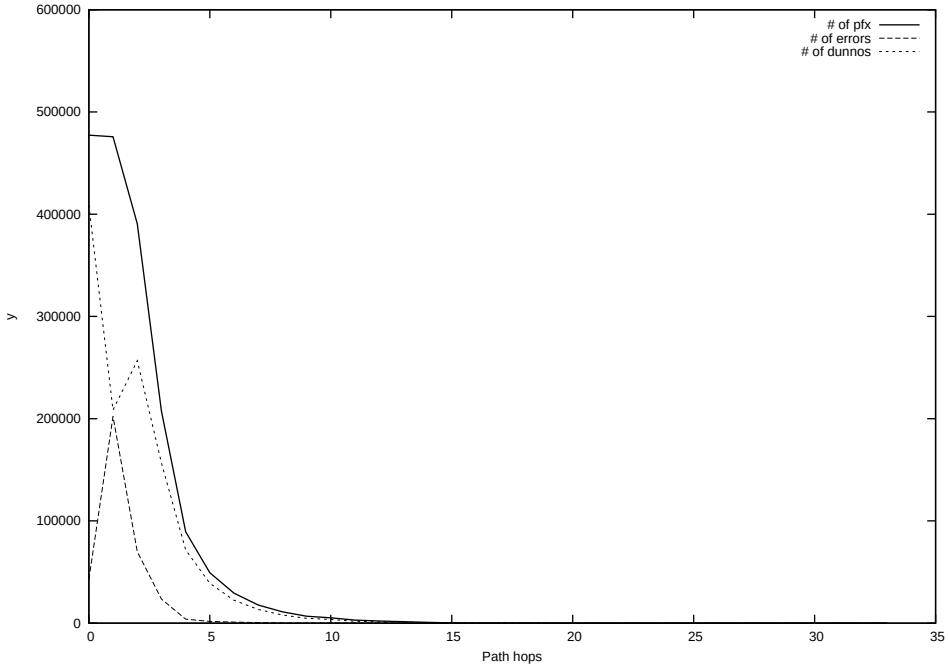
2013-11-25



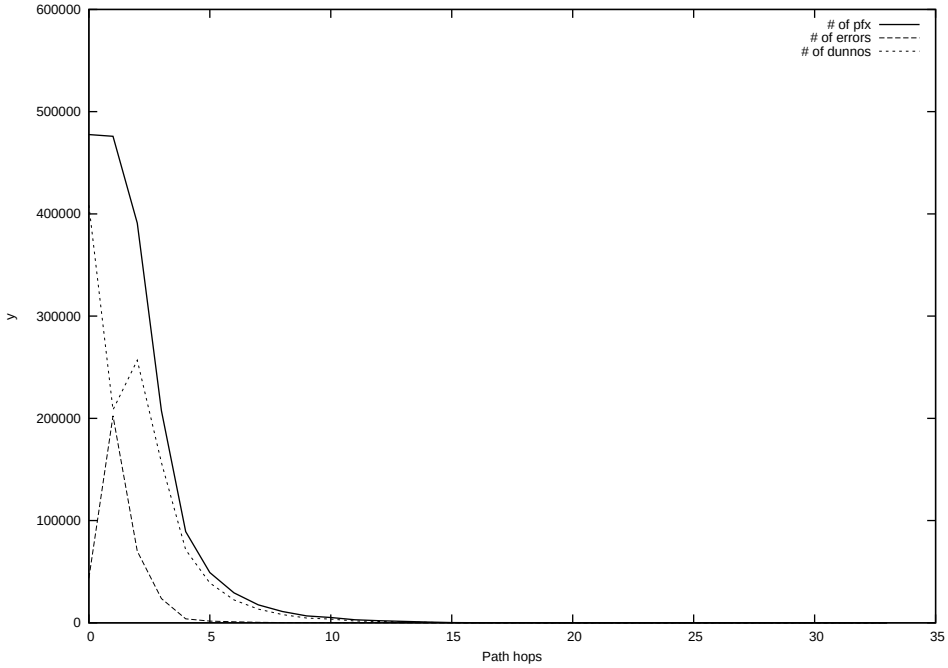
2013-11-26



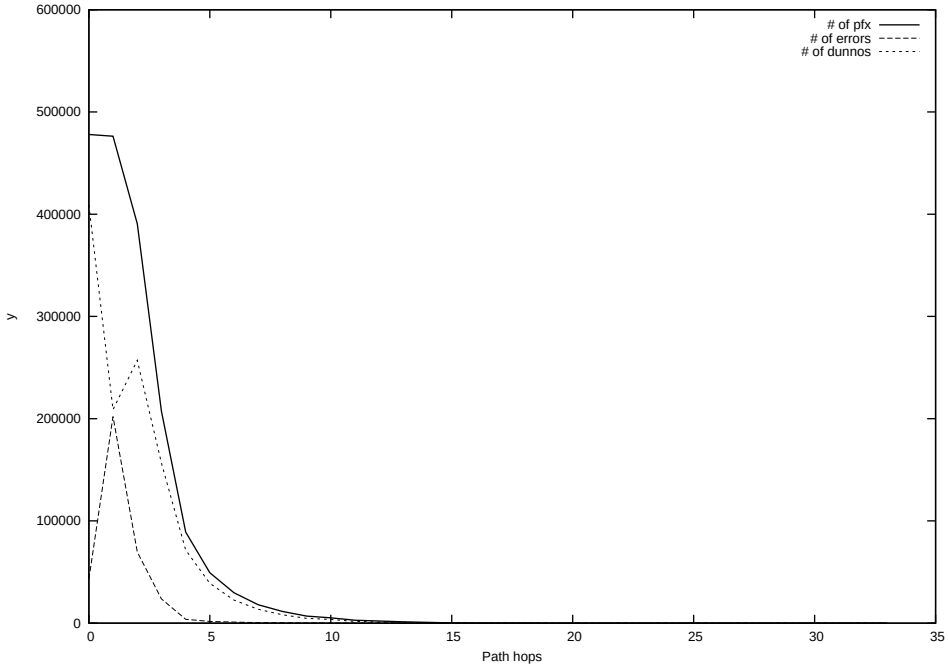
2013-11-27



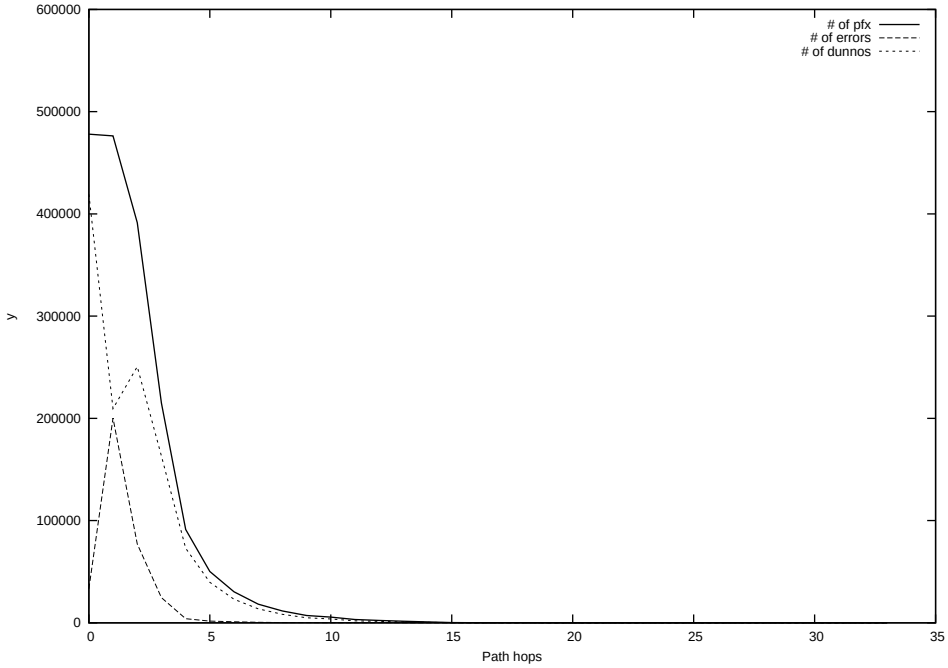
2013-11-28



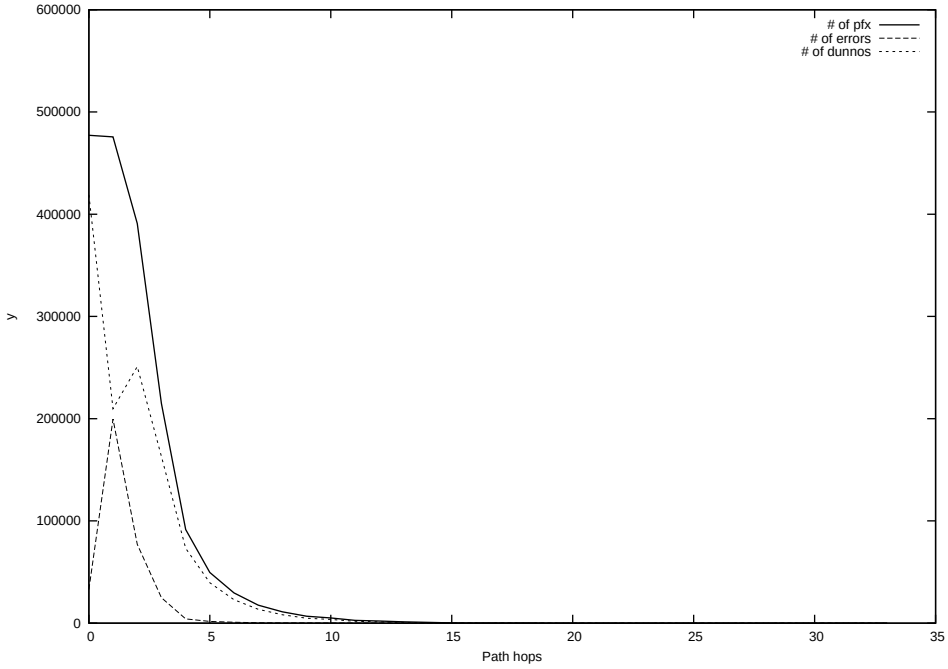
2013-11-29



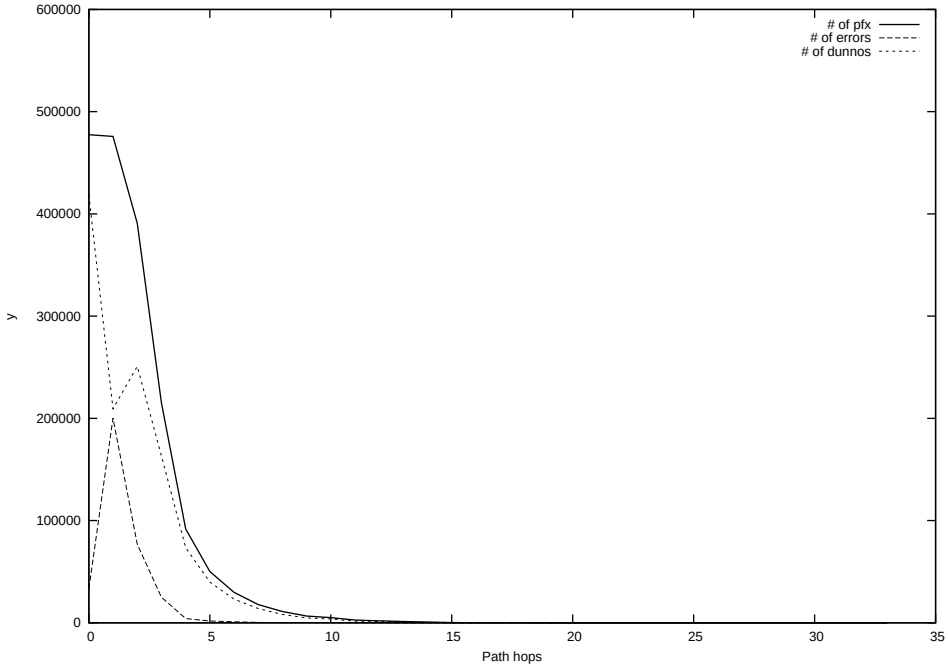
2013-11-30



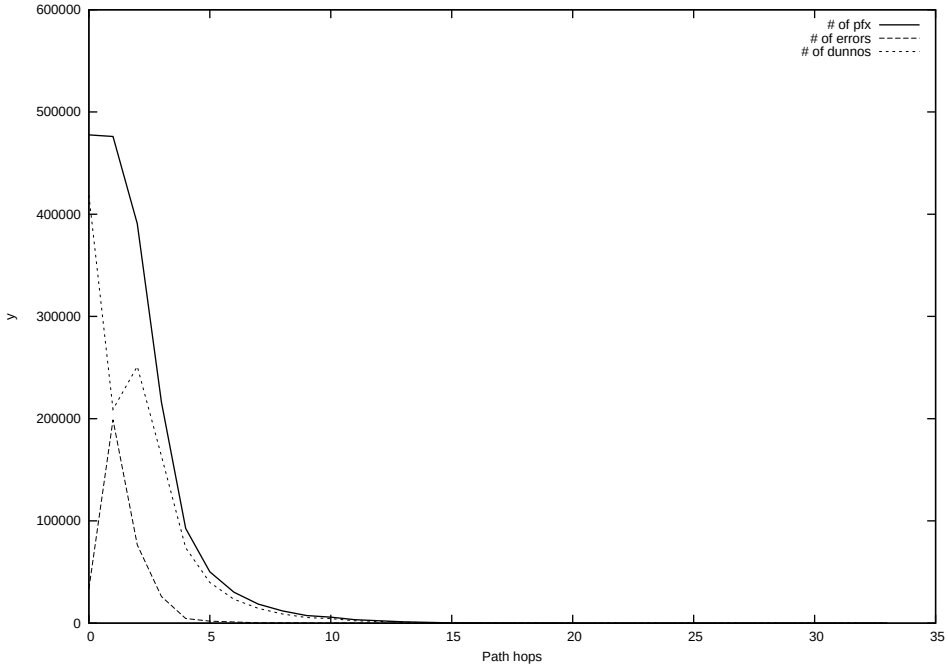
2013-12-01



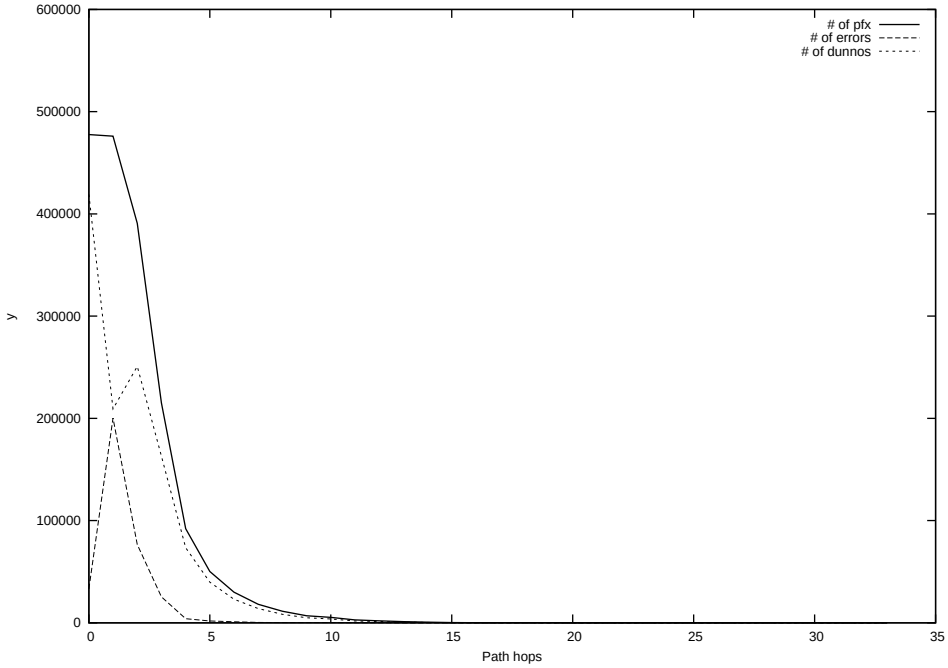
2013-12-02



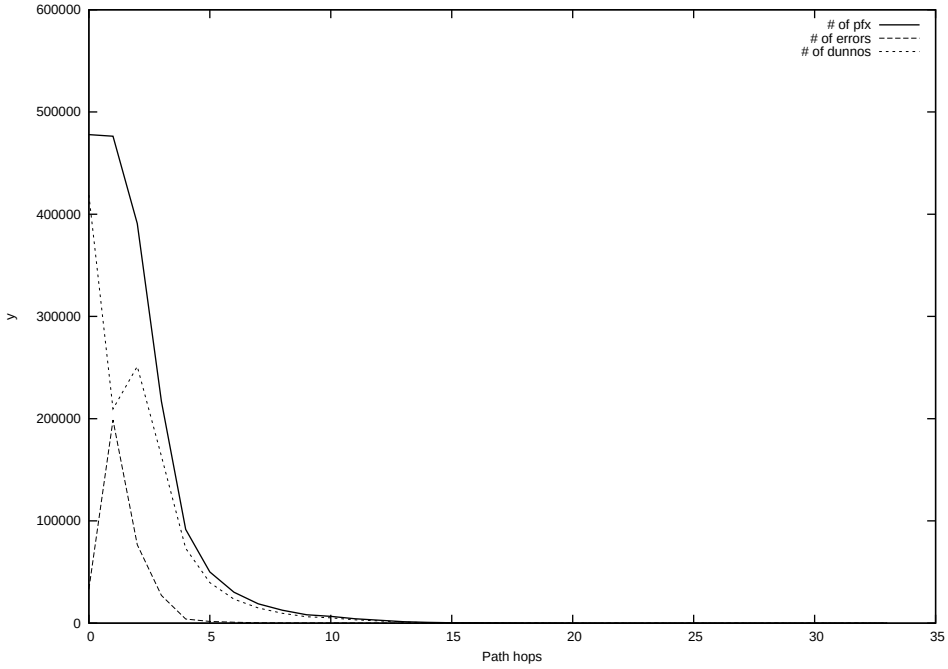
2013-12-03



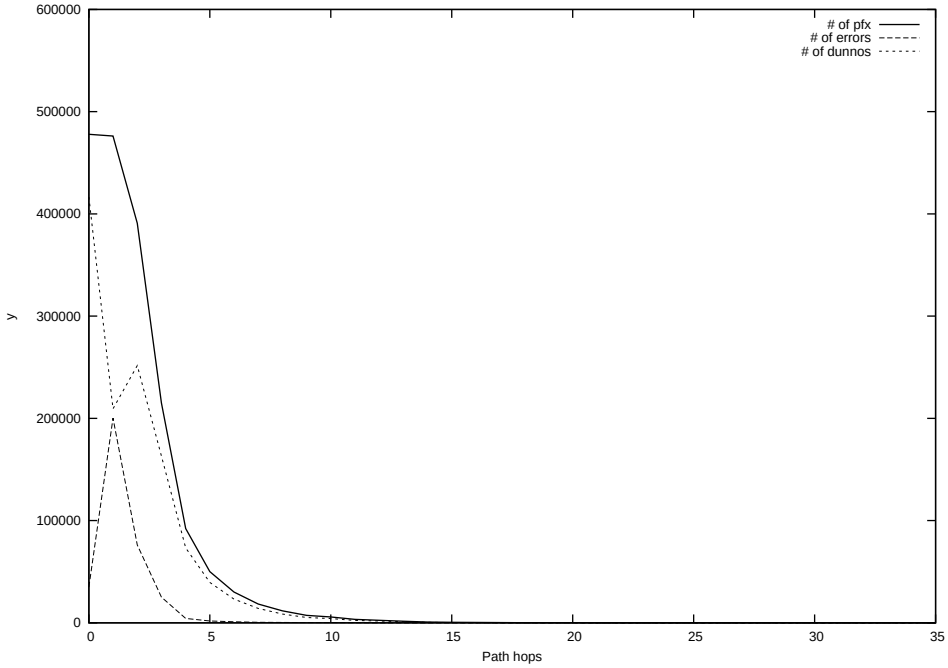
2013-12-04



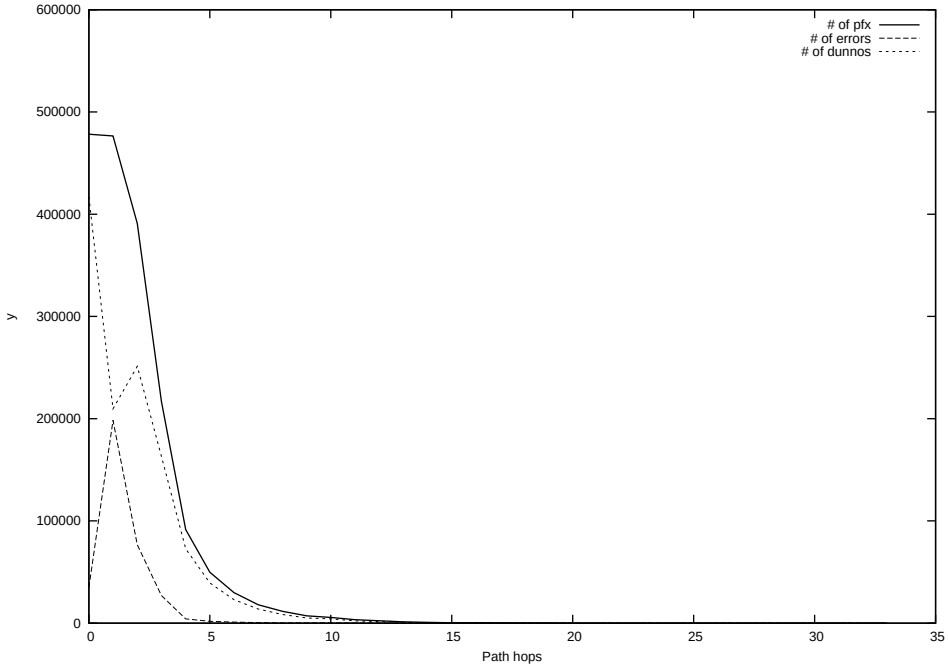
2013-12-05



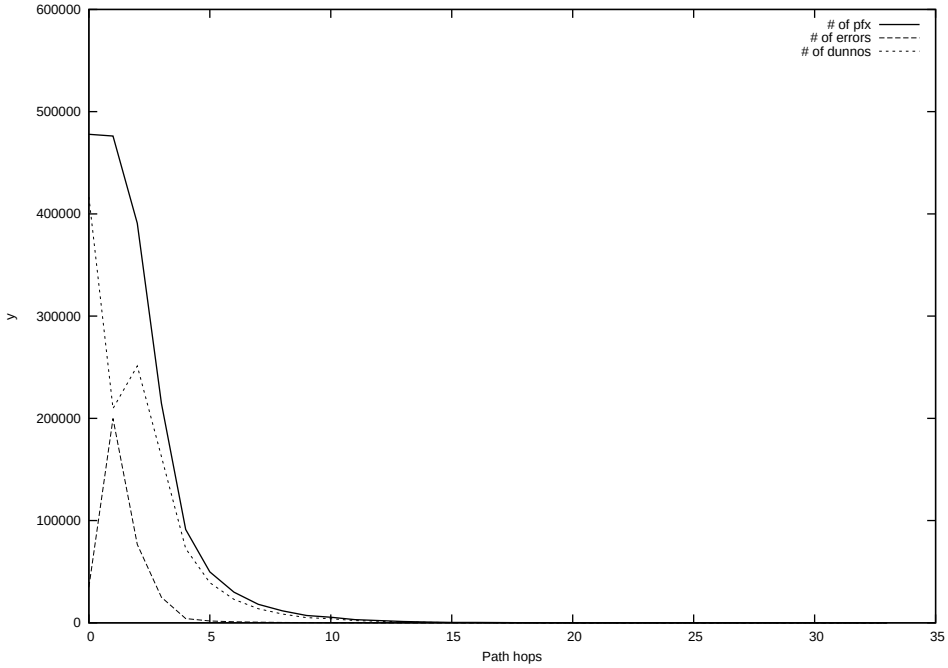
2013-12-06



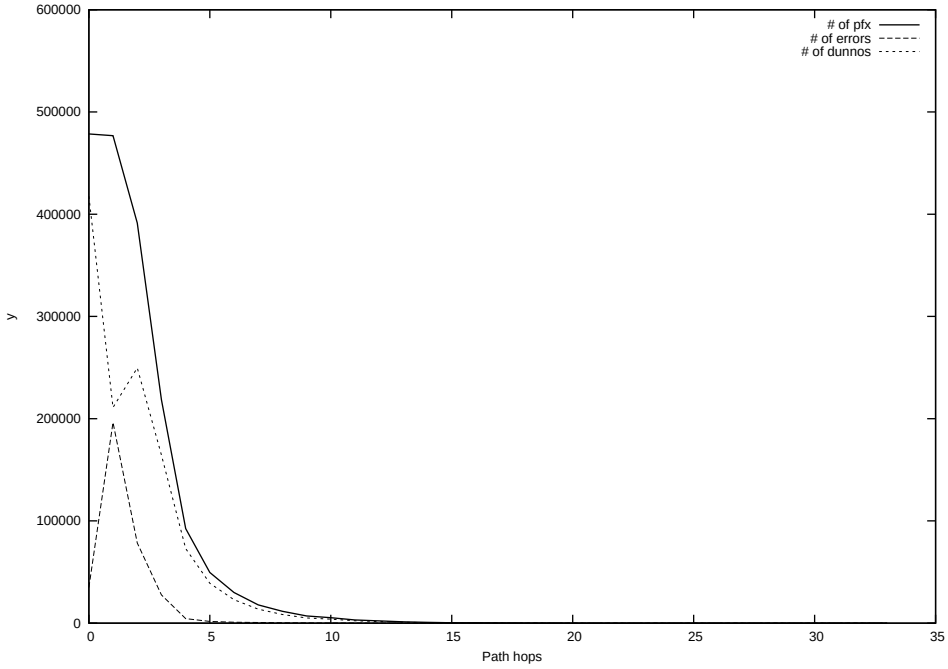
2013-12-07



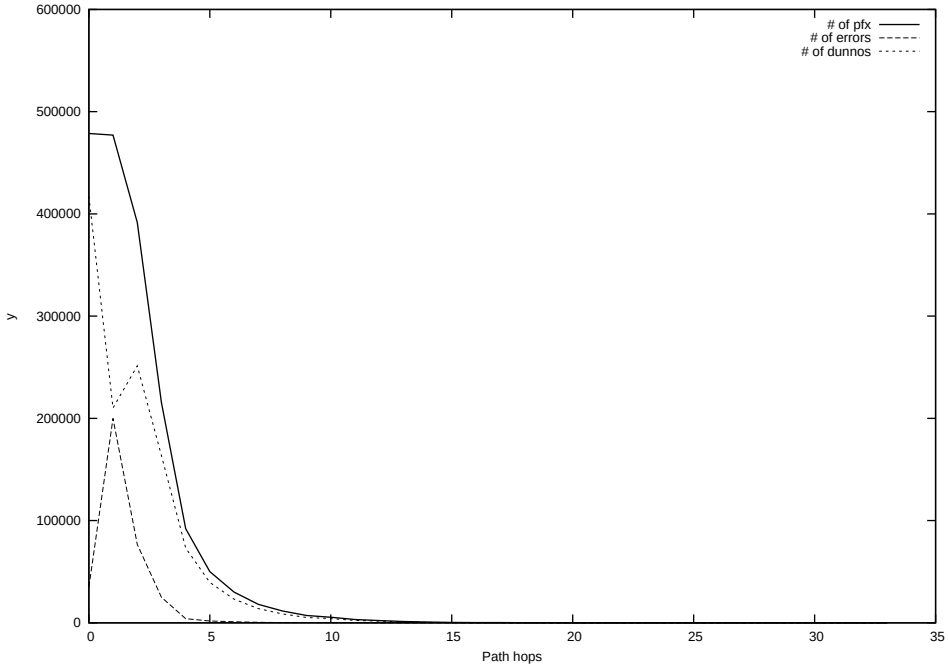
2013-12-08



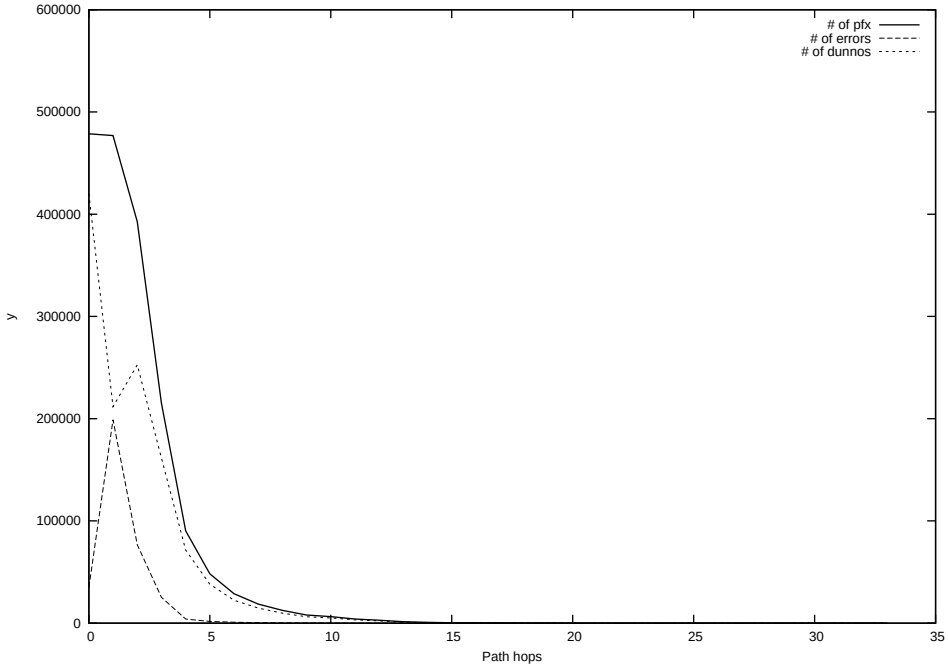
2013-12-09



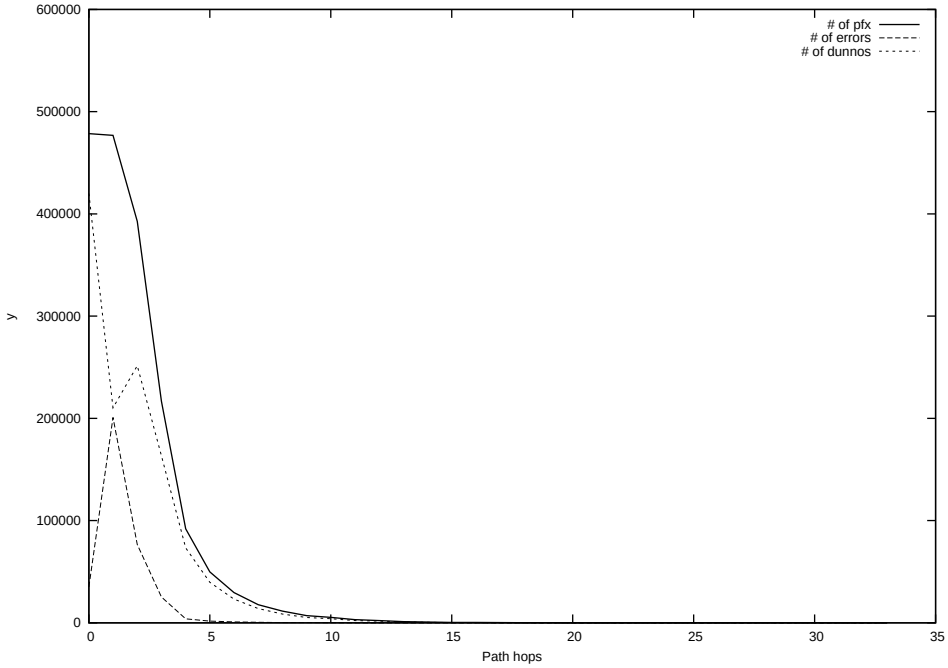
2013-12-10



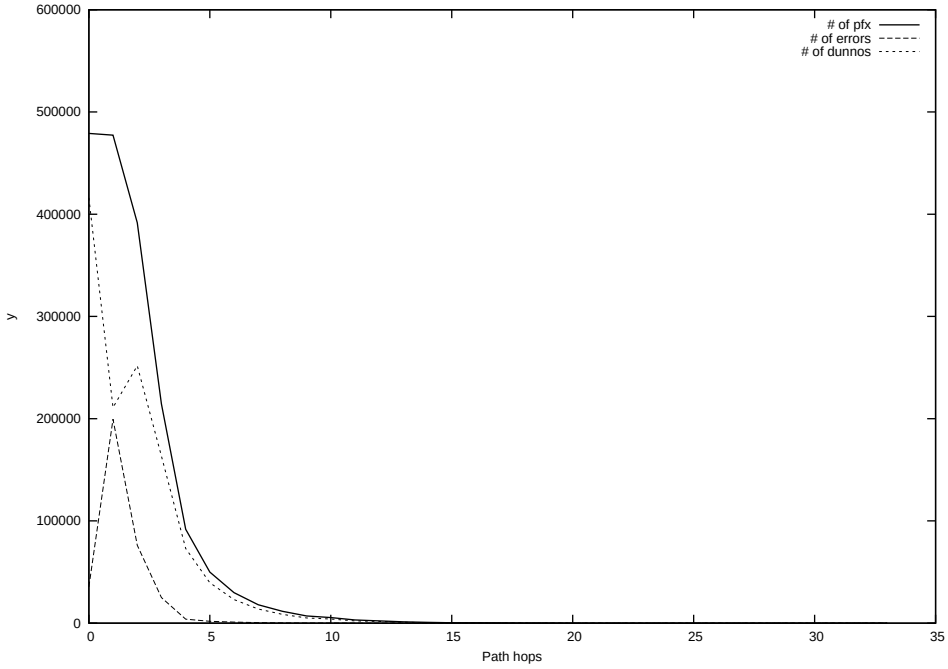
2013-12-11



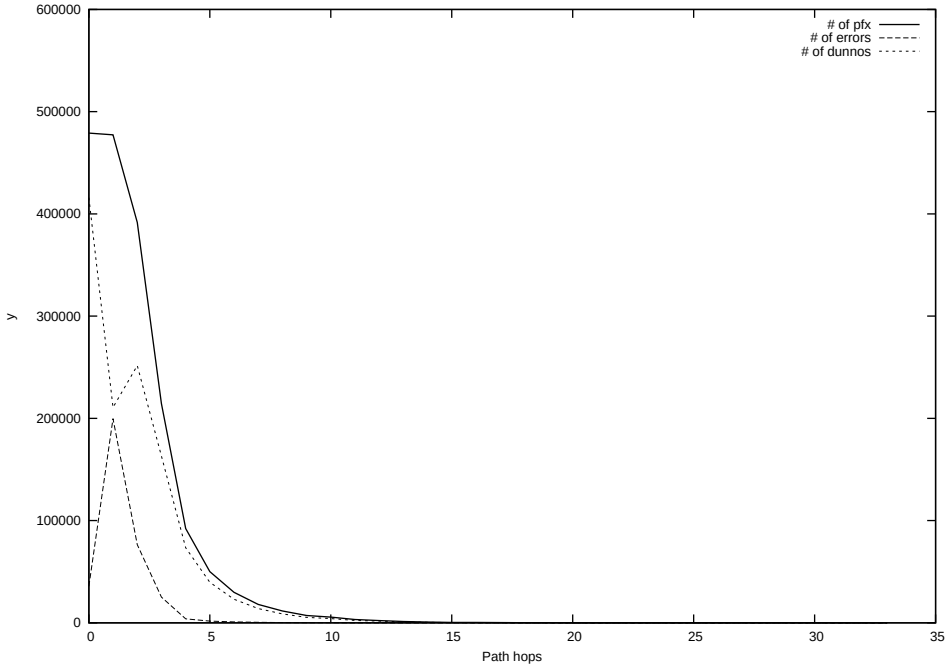
2013-12-12



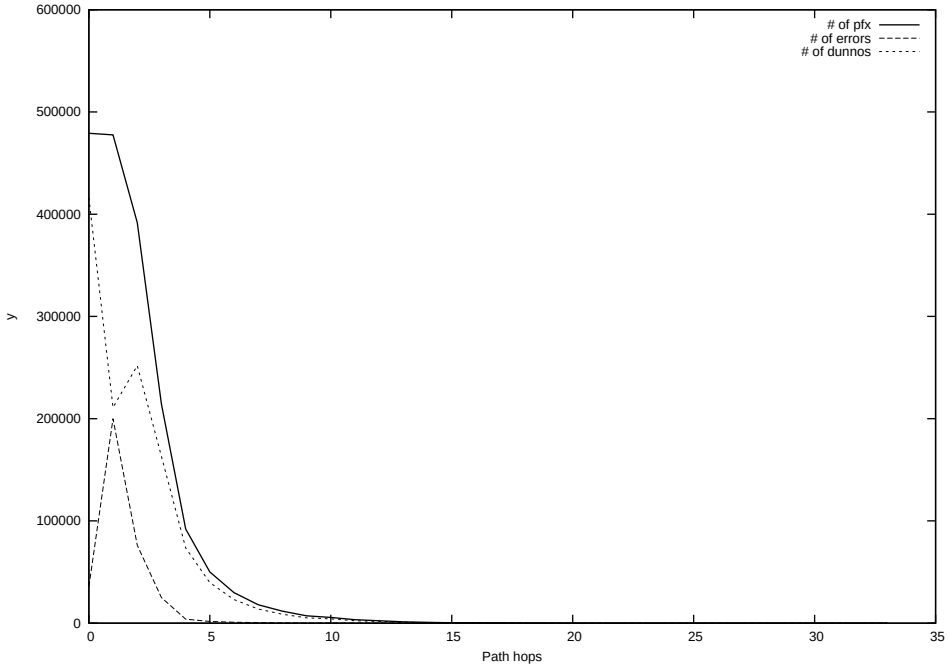
2013-12-13



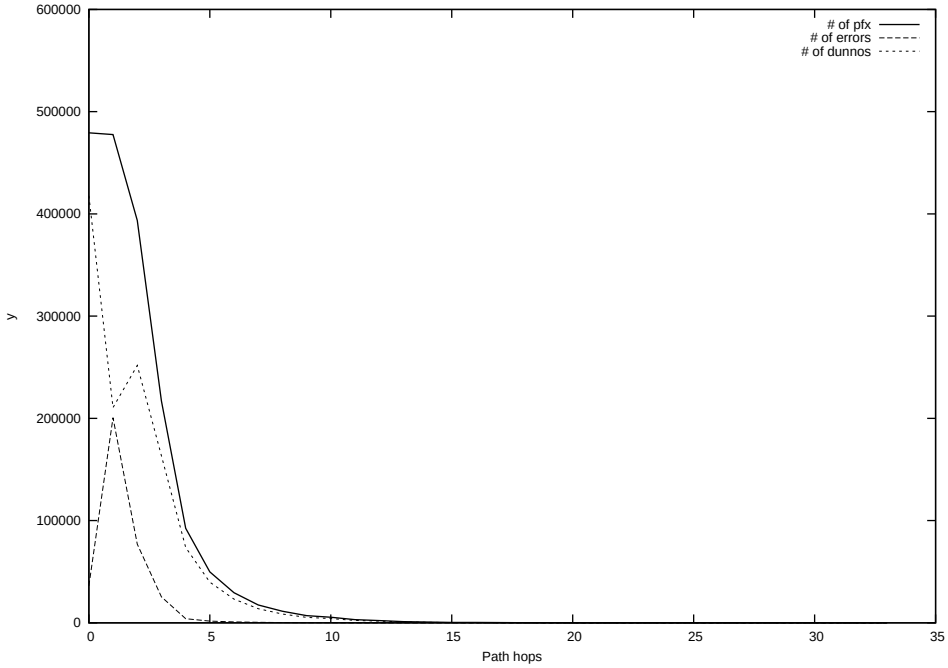
2013-12-14



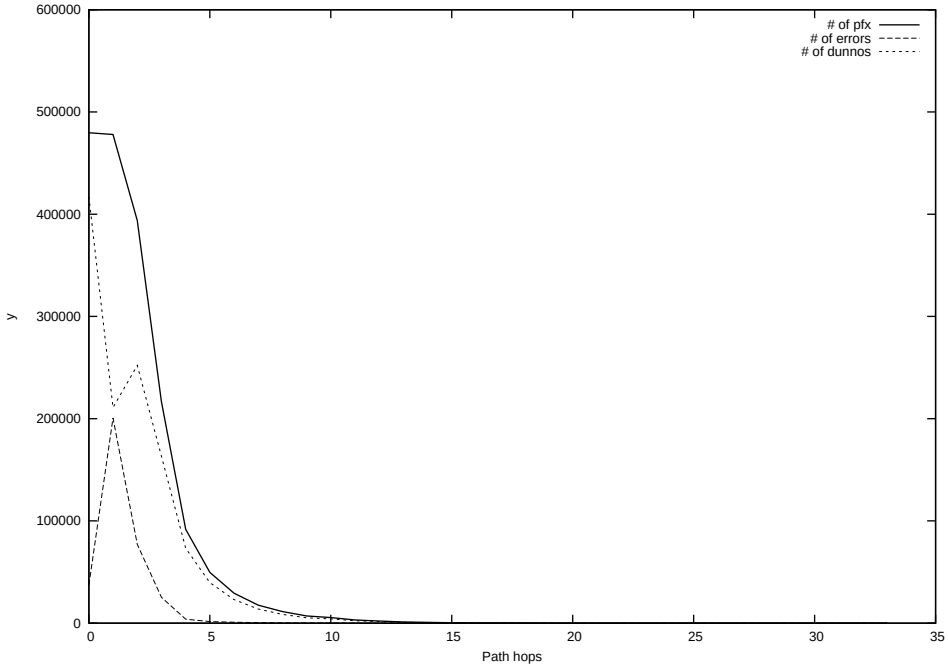
2013-12-15



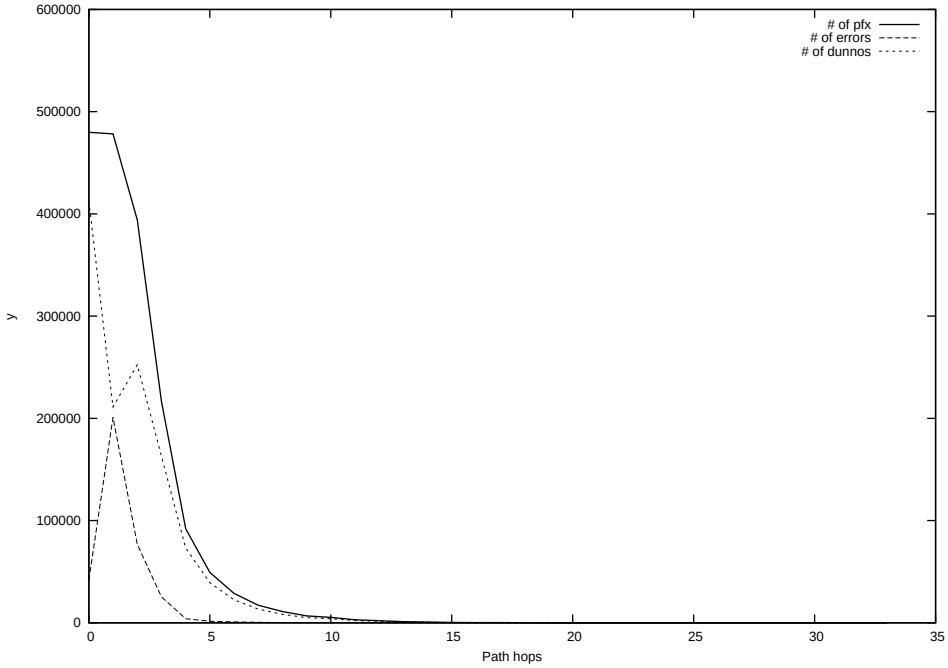
2013-12-16



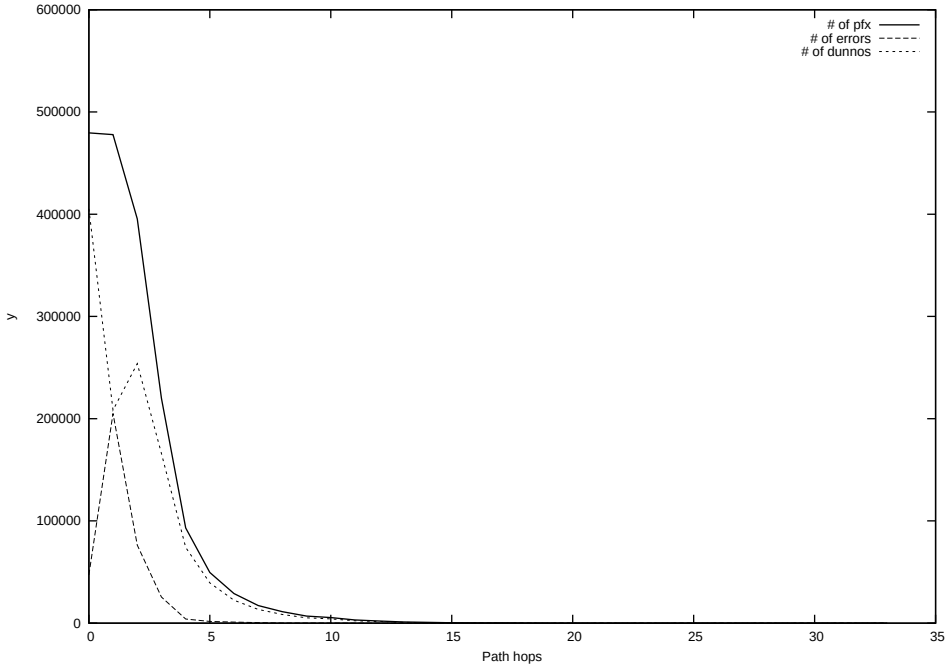
2013-12-17



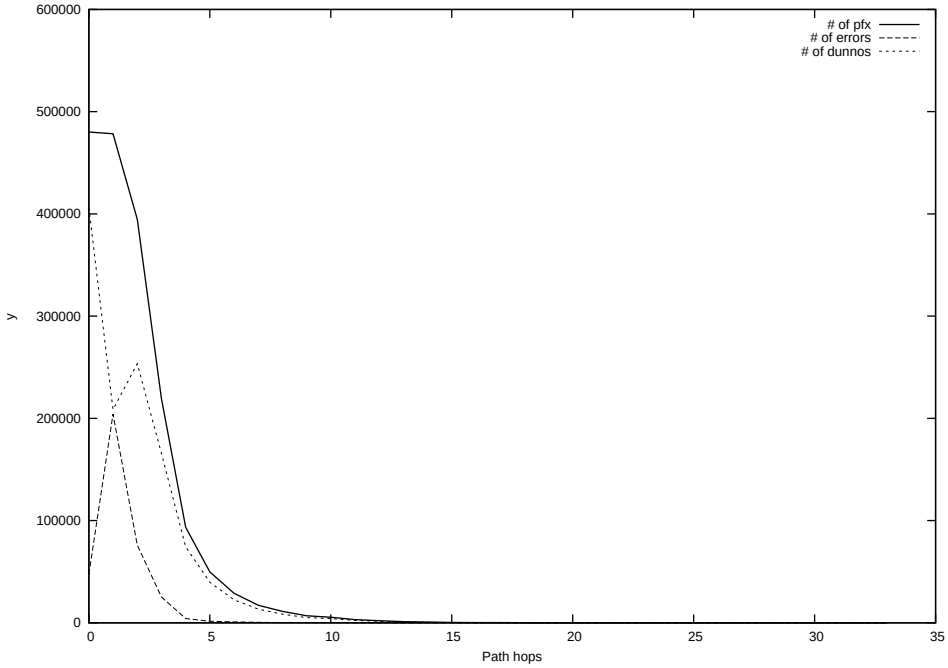
2013-12-18



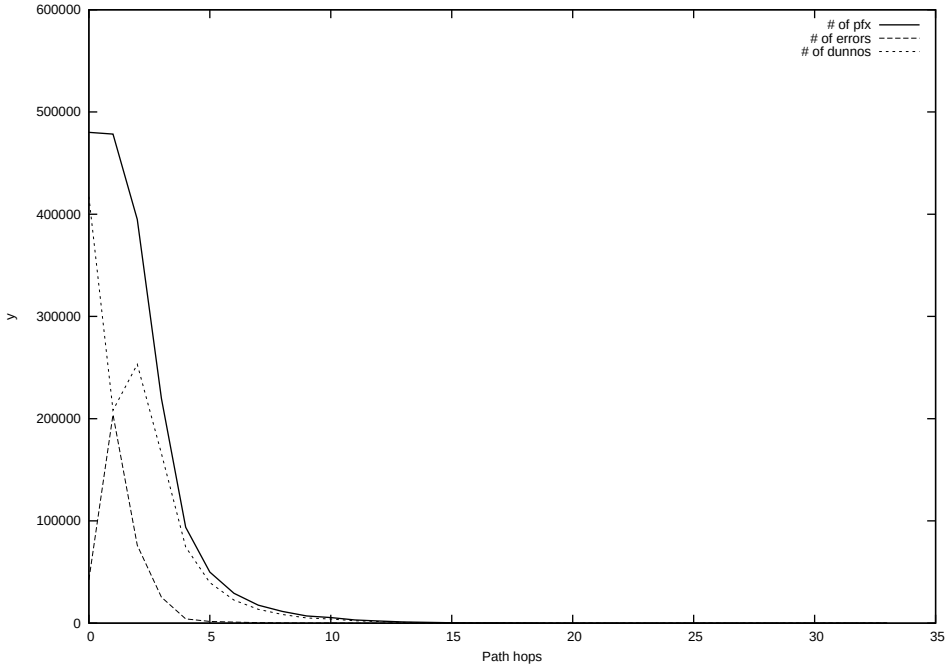
2013-12-19



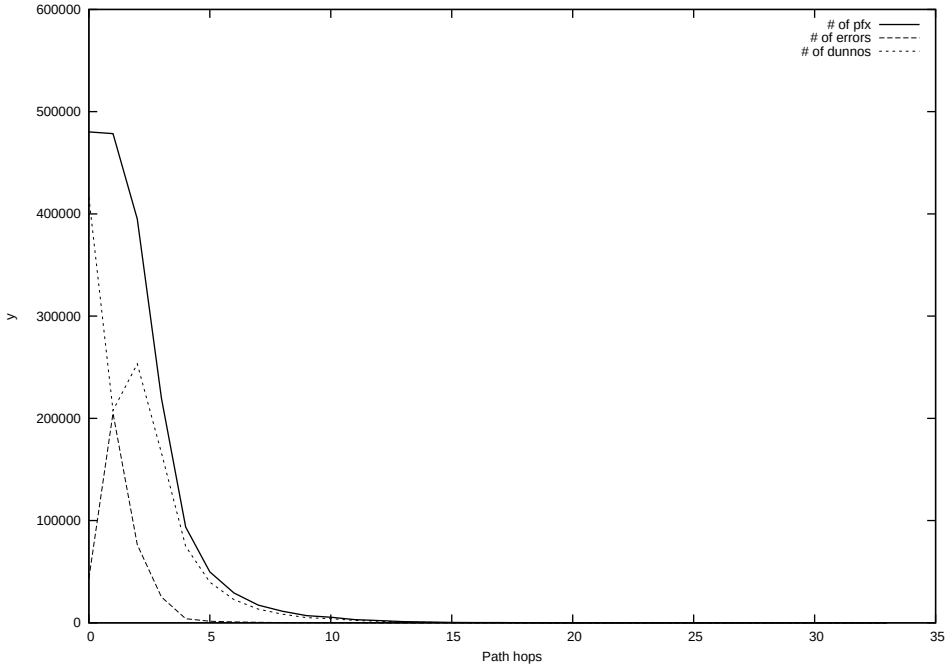
2013-12-20



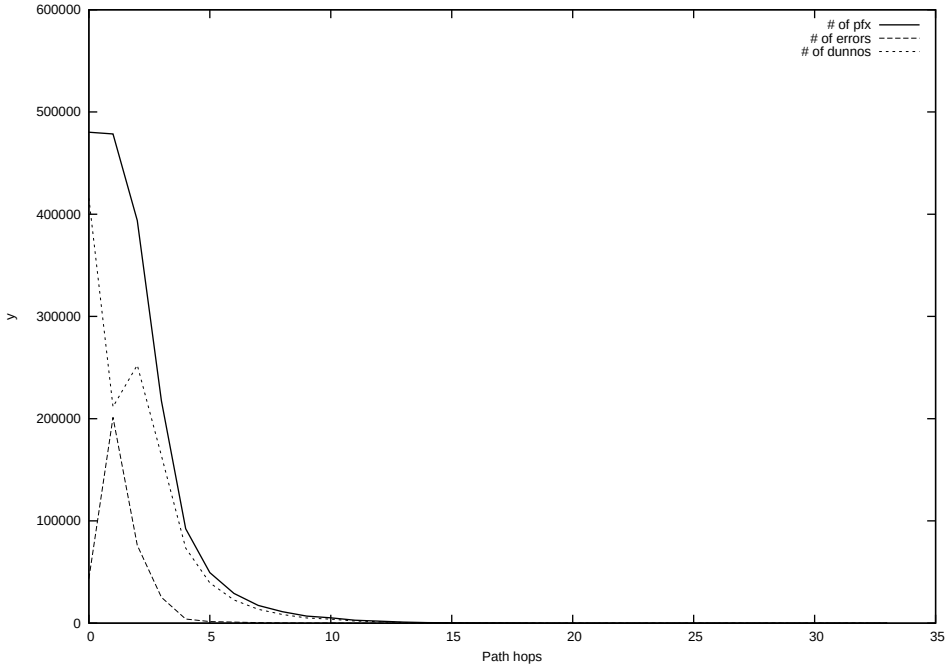
2013-12-21



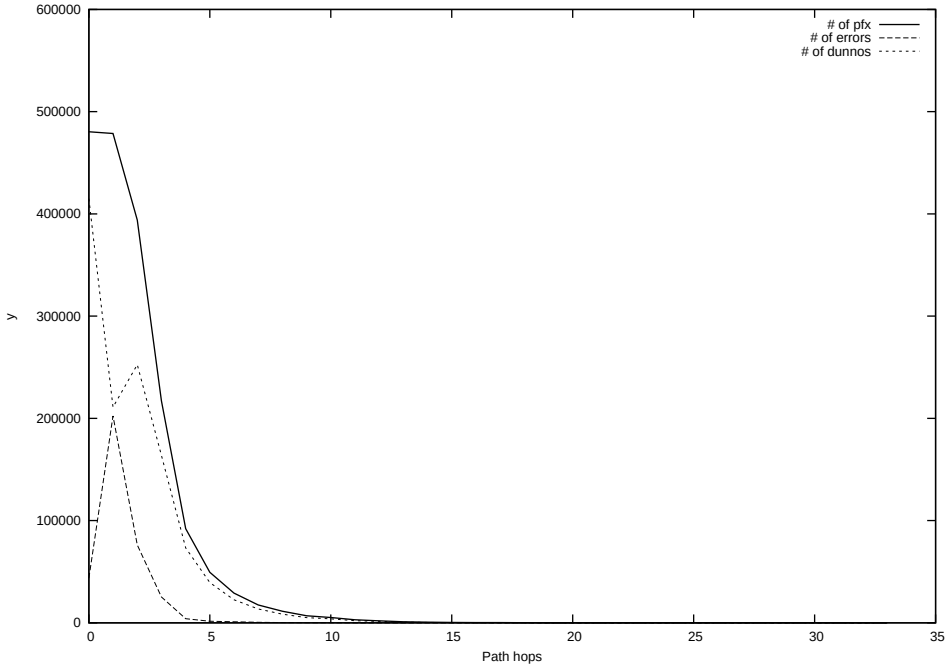
2013-12-22



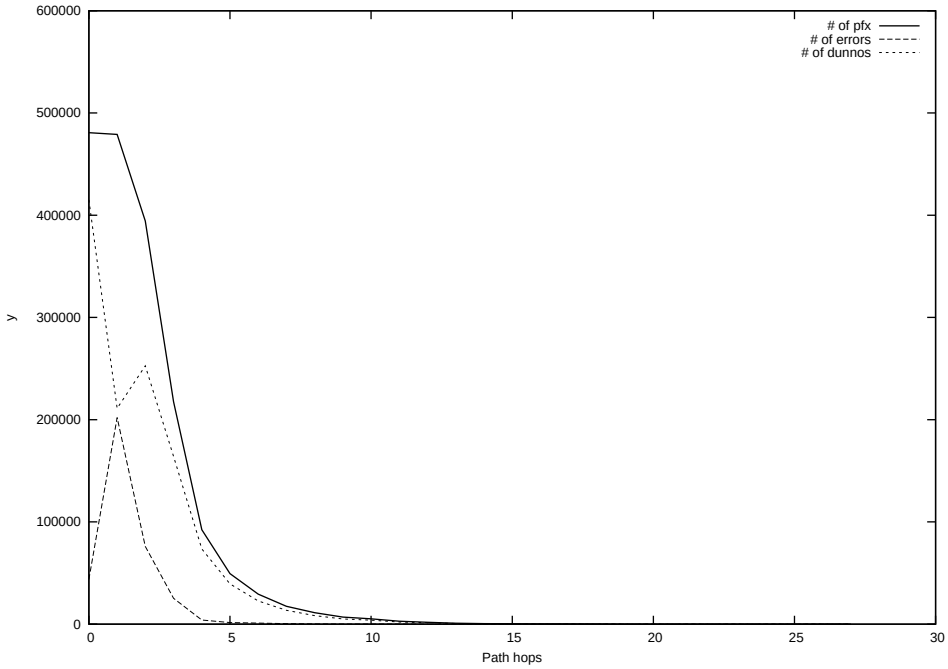
2013-12-23



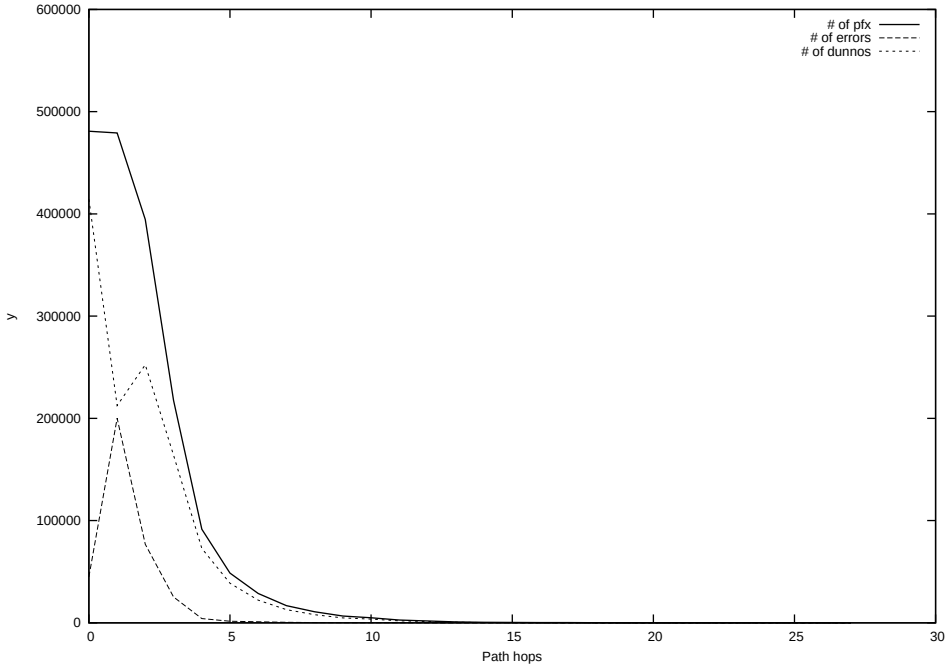
2013-12-24



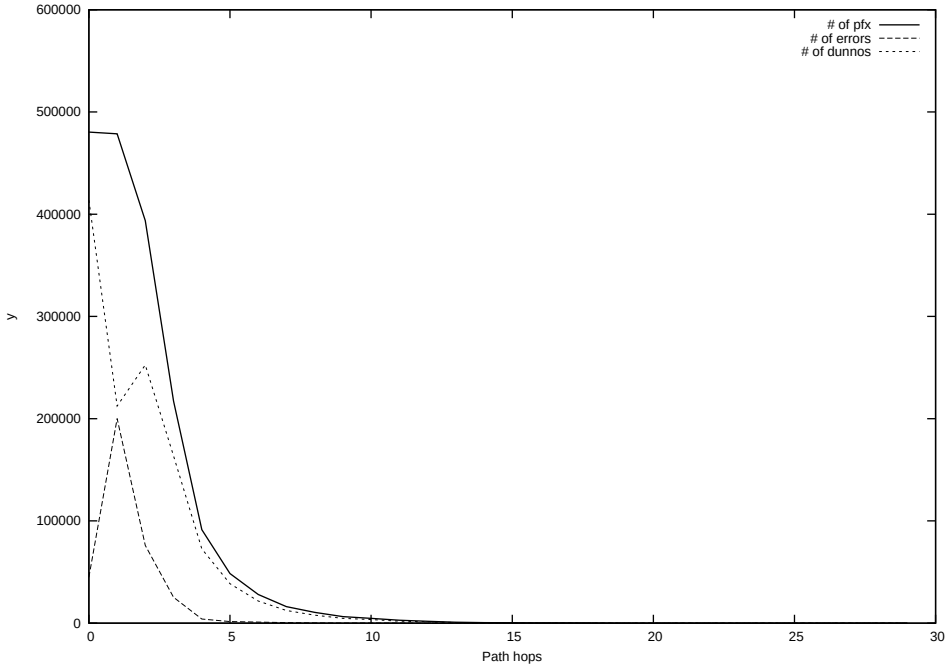
2013-12-25



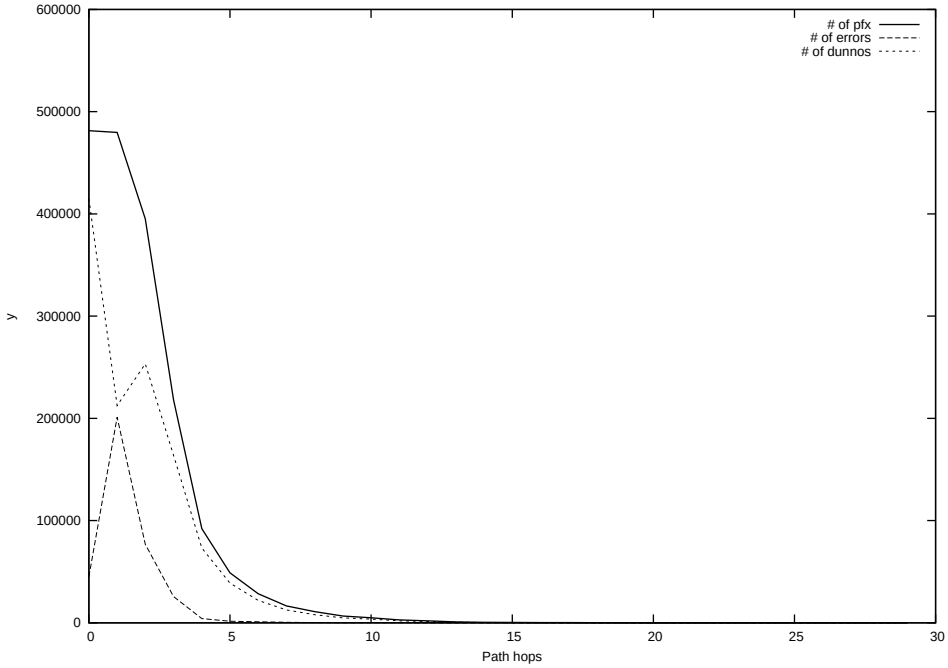
2013-12-26



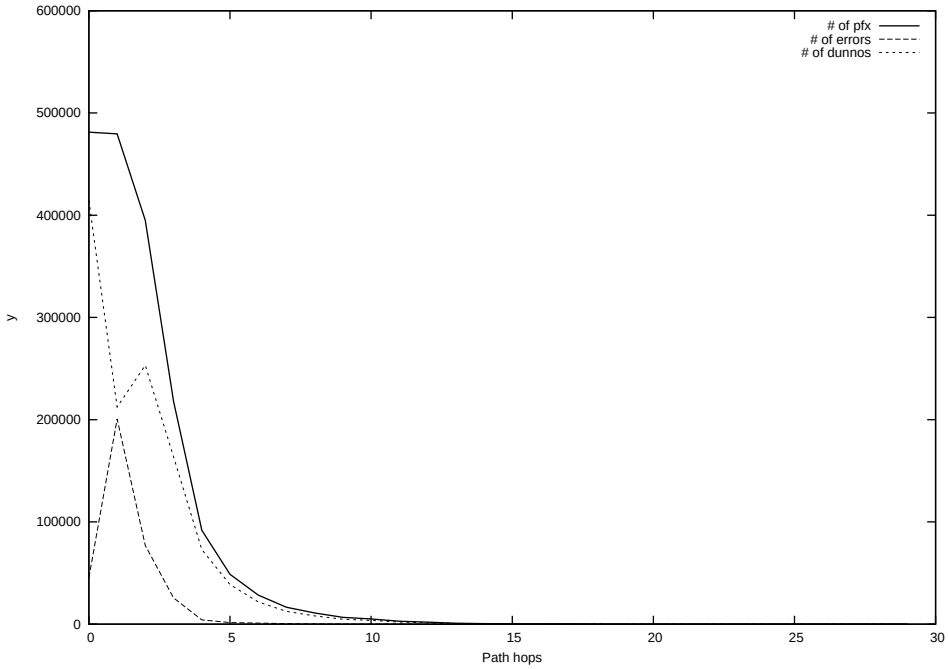
2013-12-27



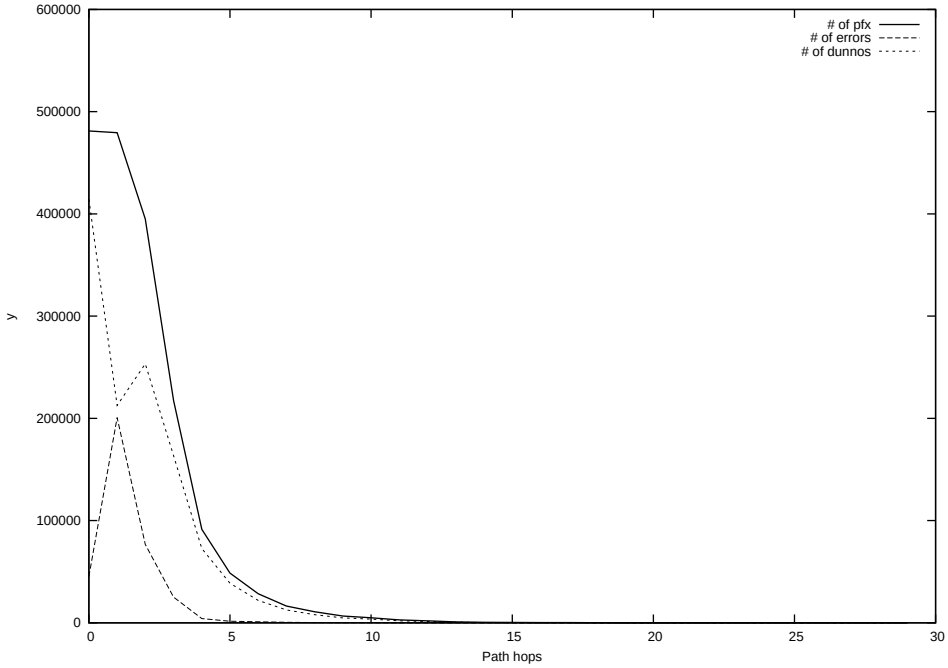
2013-12-28



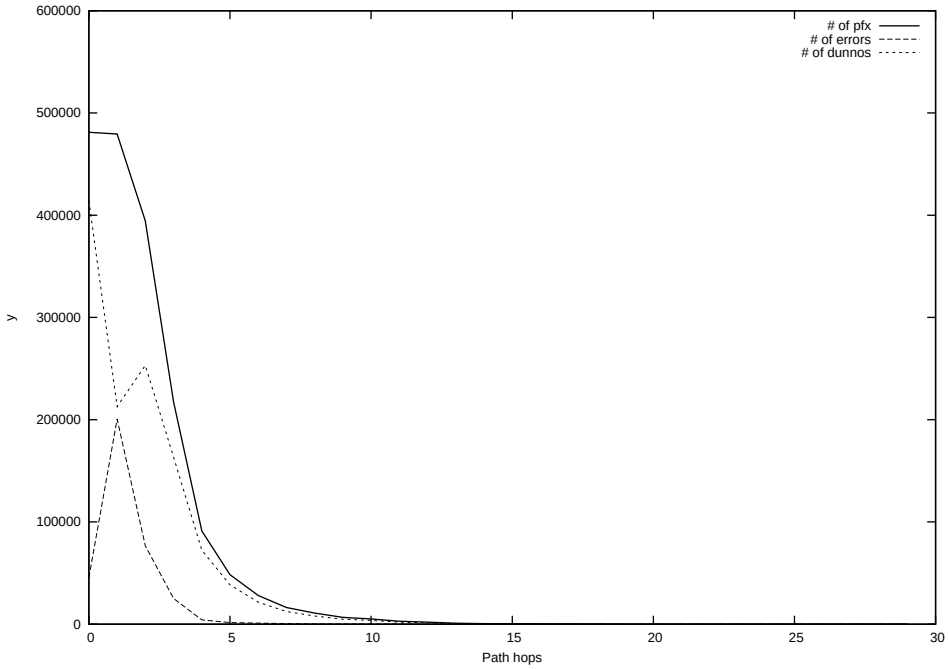
2013-12-29



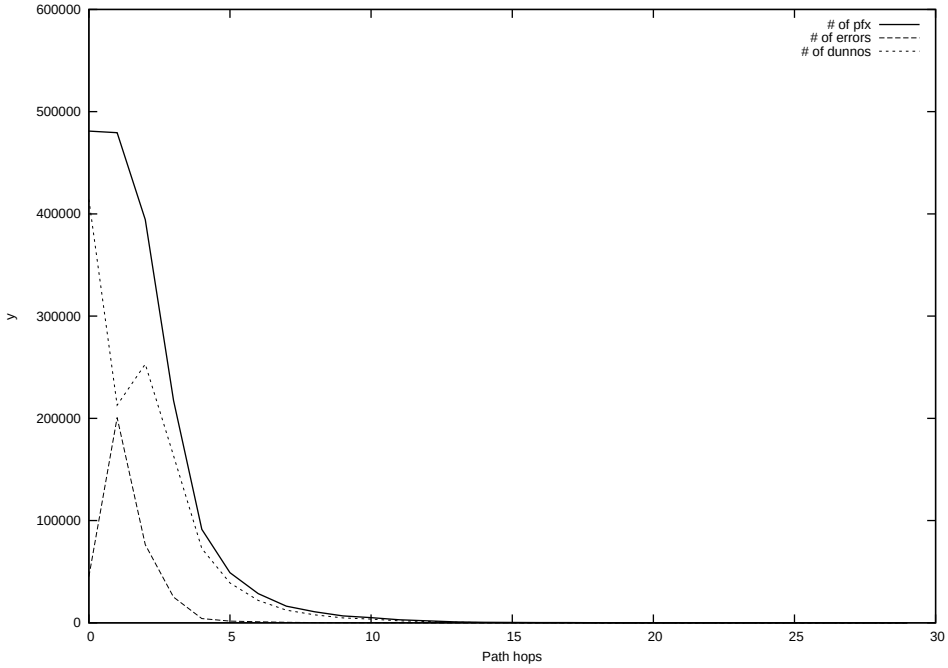
2013-12-30



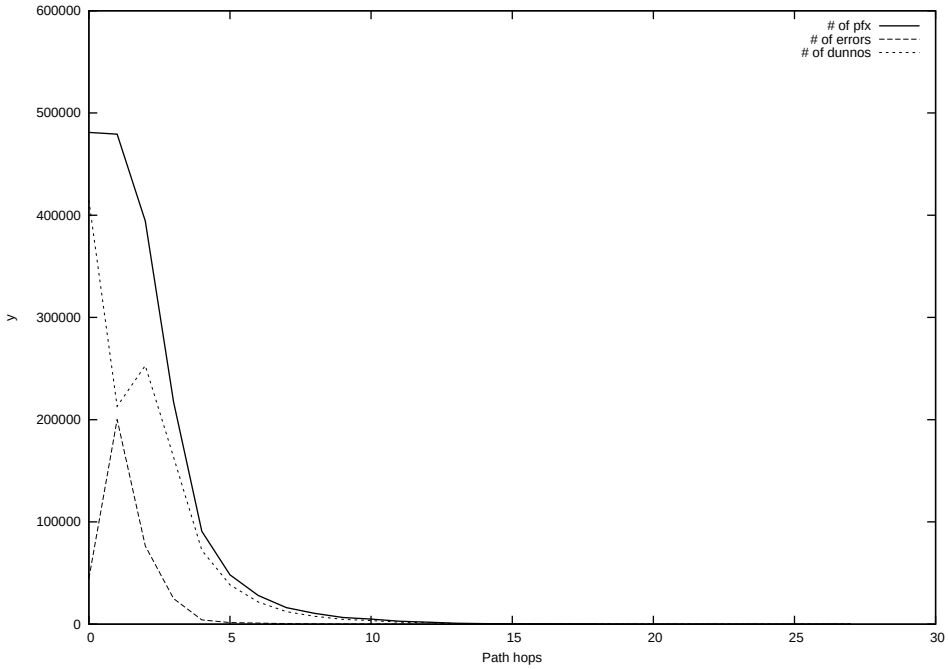
2013-12-31



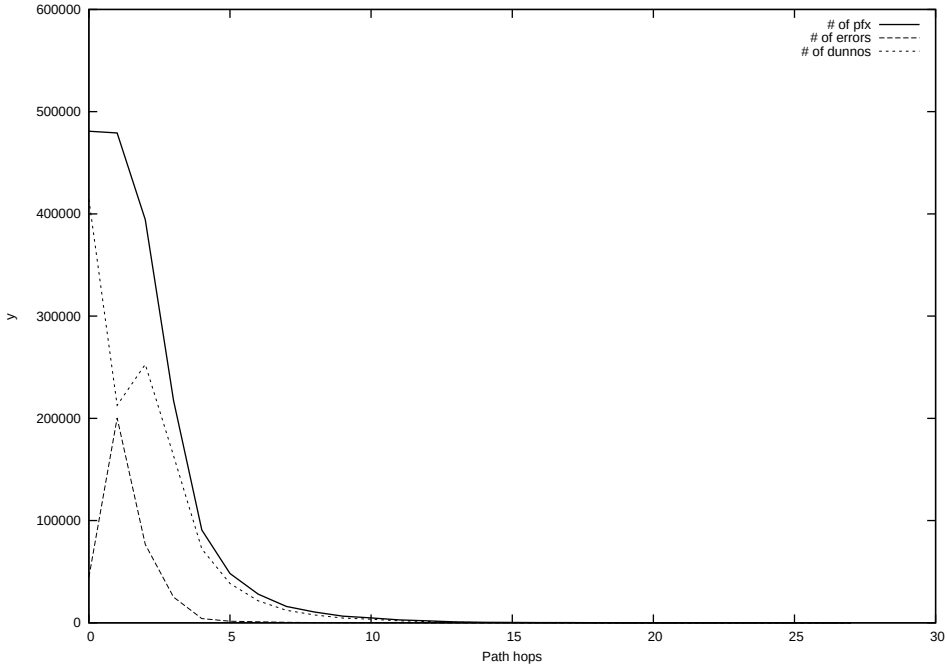
2014-01-01



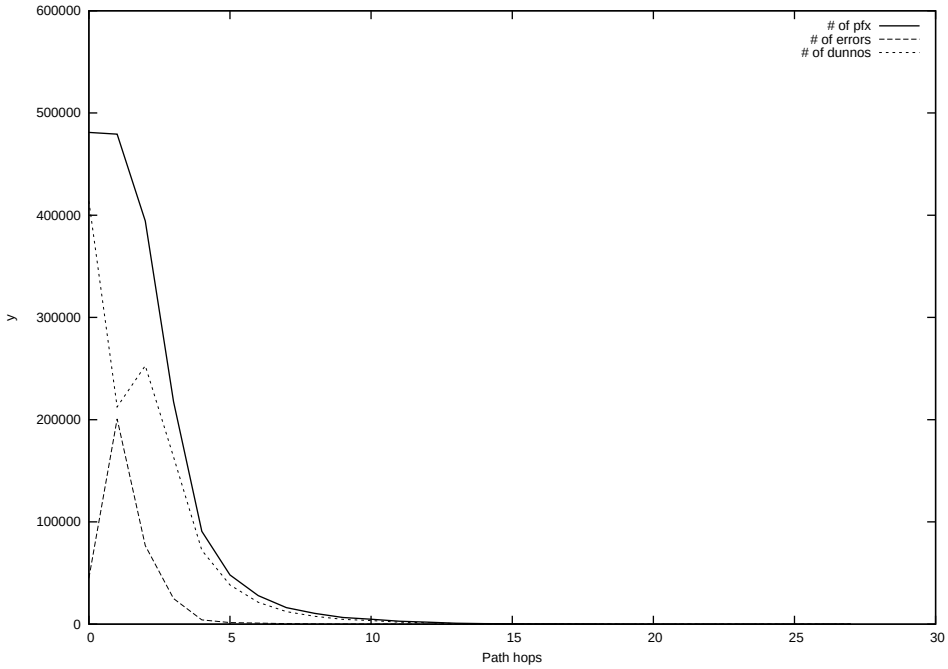
2014-01-02



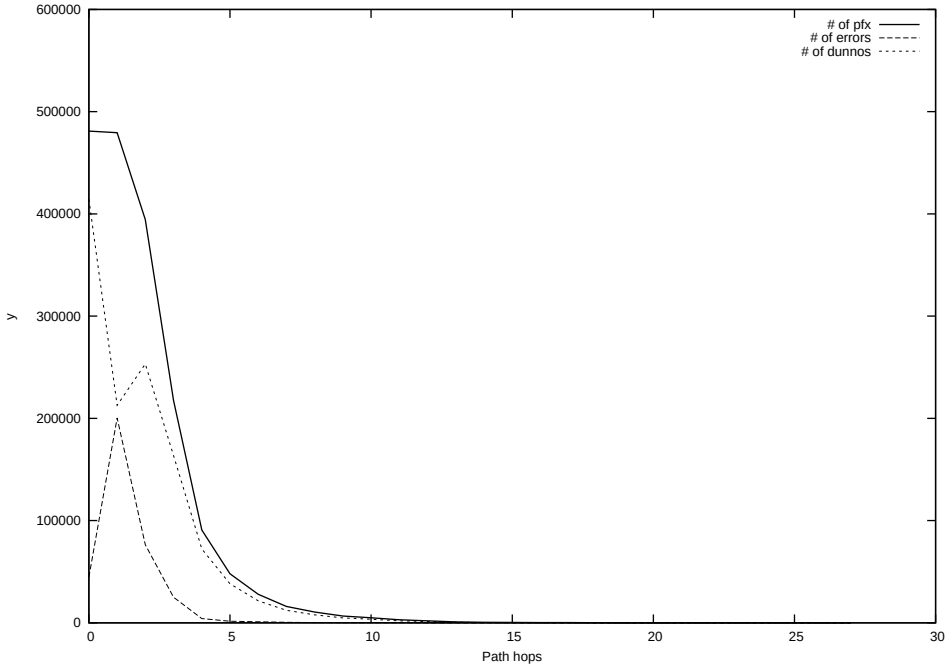
2014-01-03



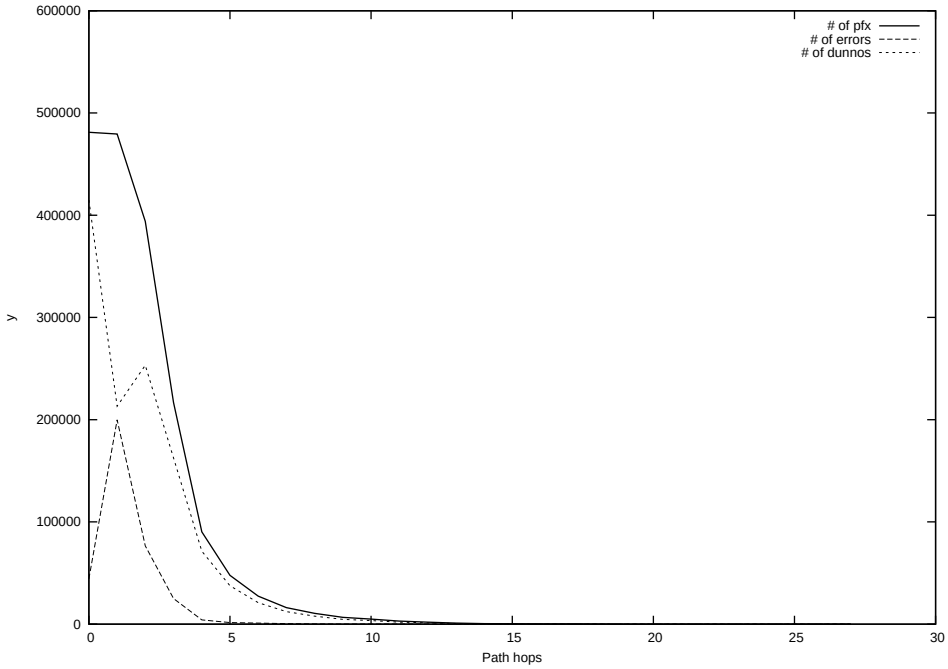
2014-01-04



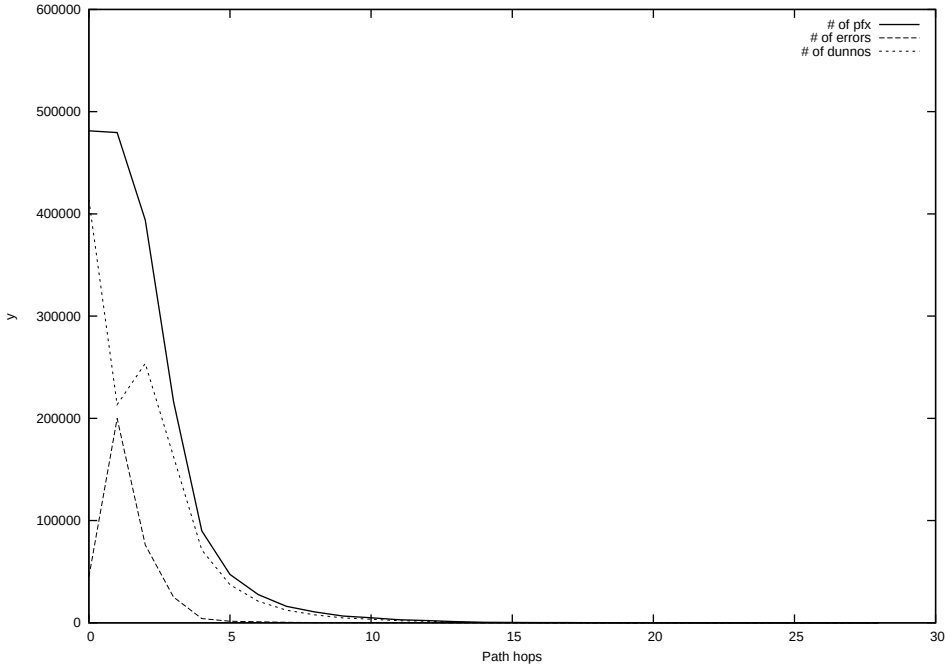
2014-01-05



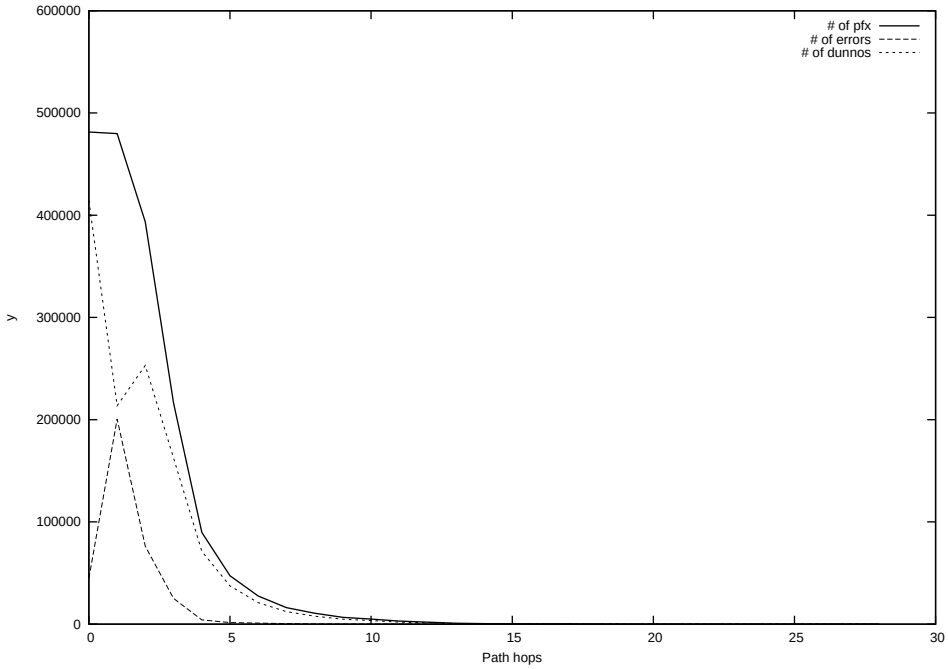
2014-01-06



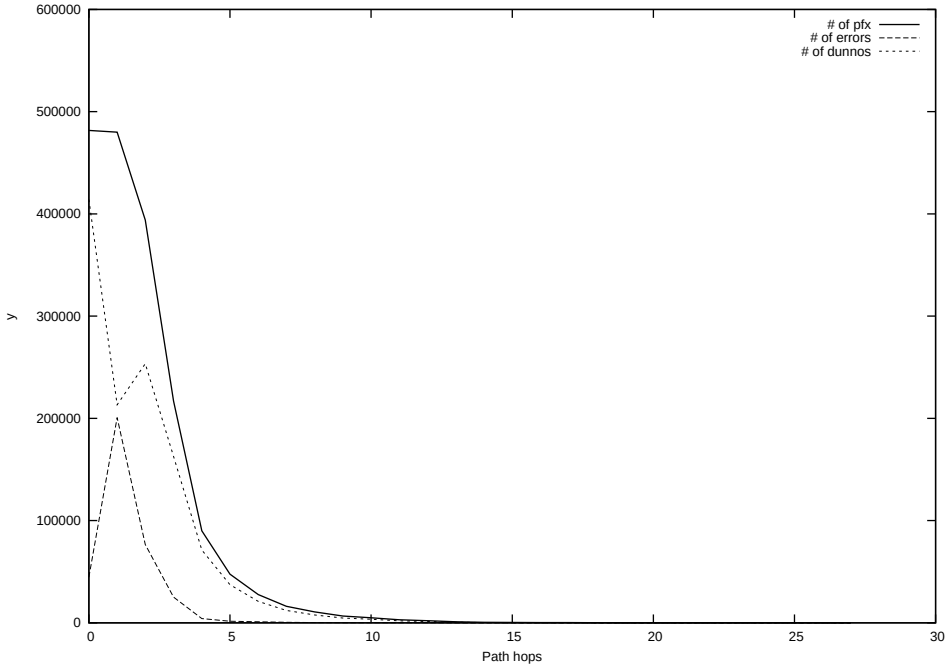
2014-01-07



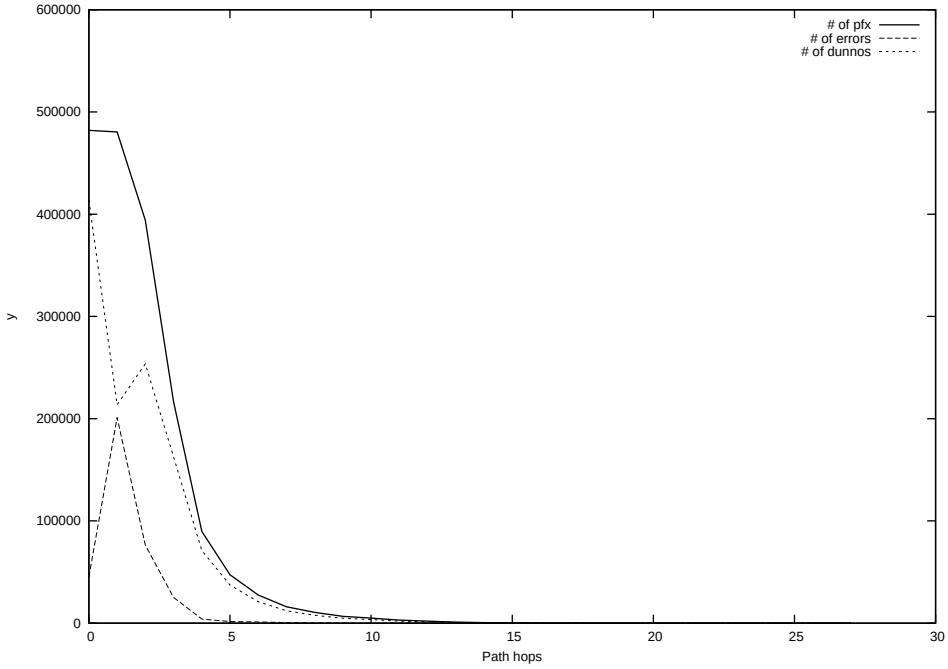
2014-01-08



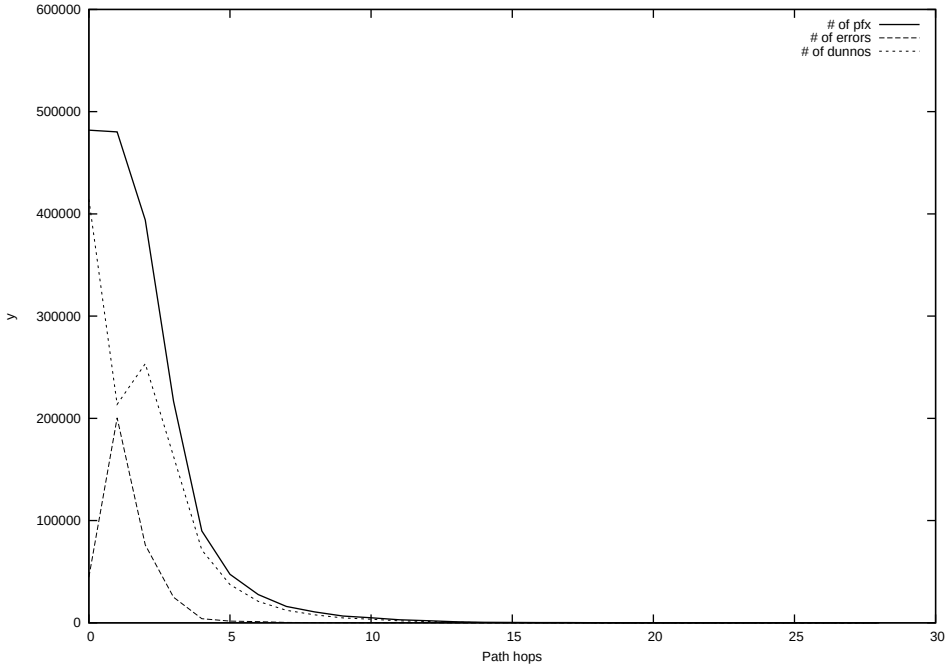
2014-01-09



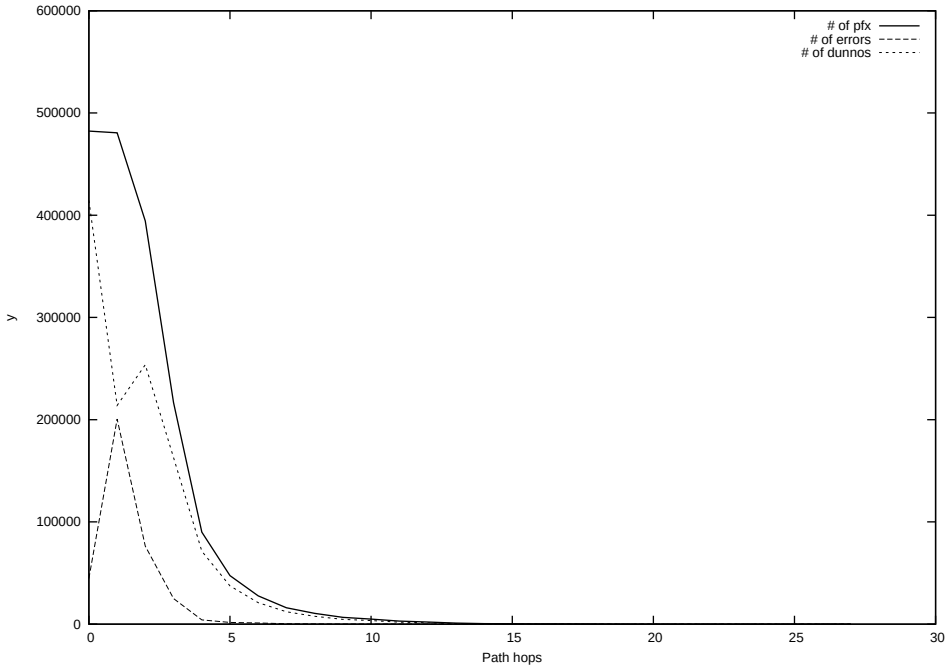
2014-01-10



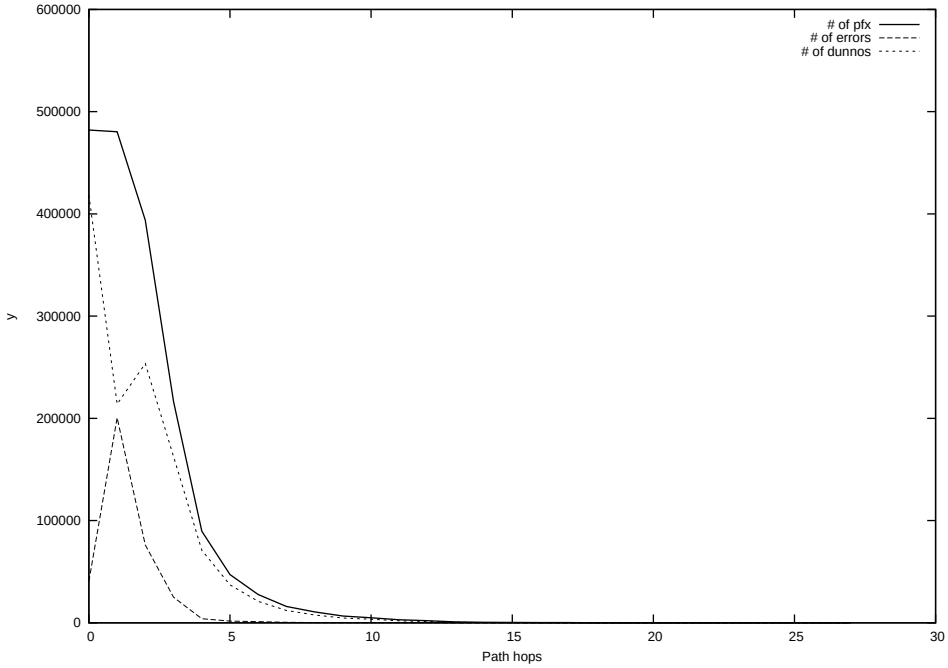
2014-01-11



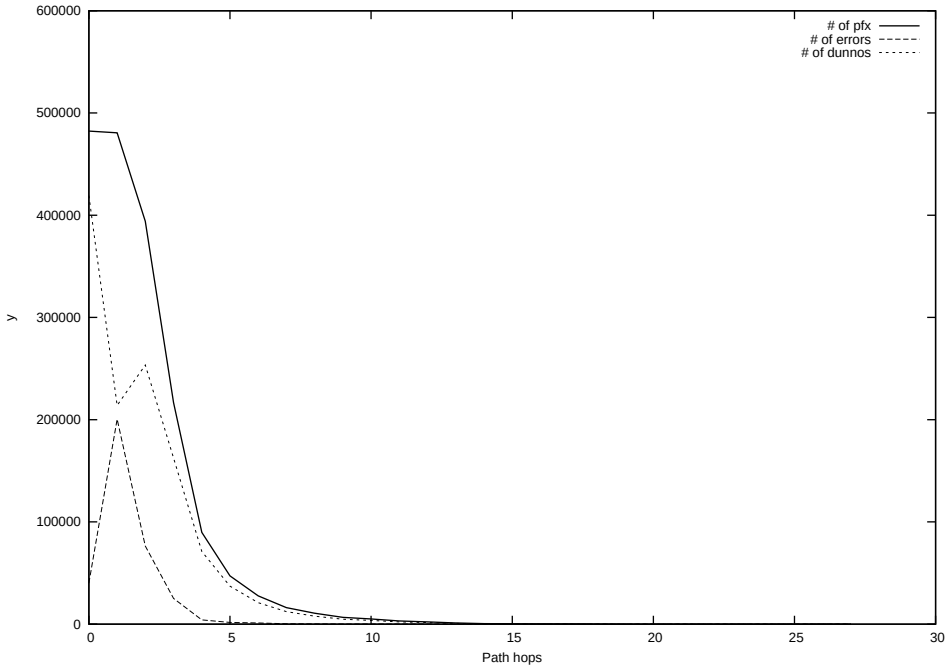
2014-01-12



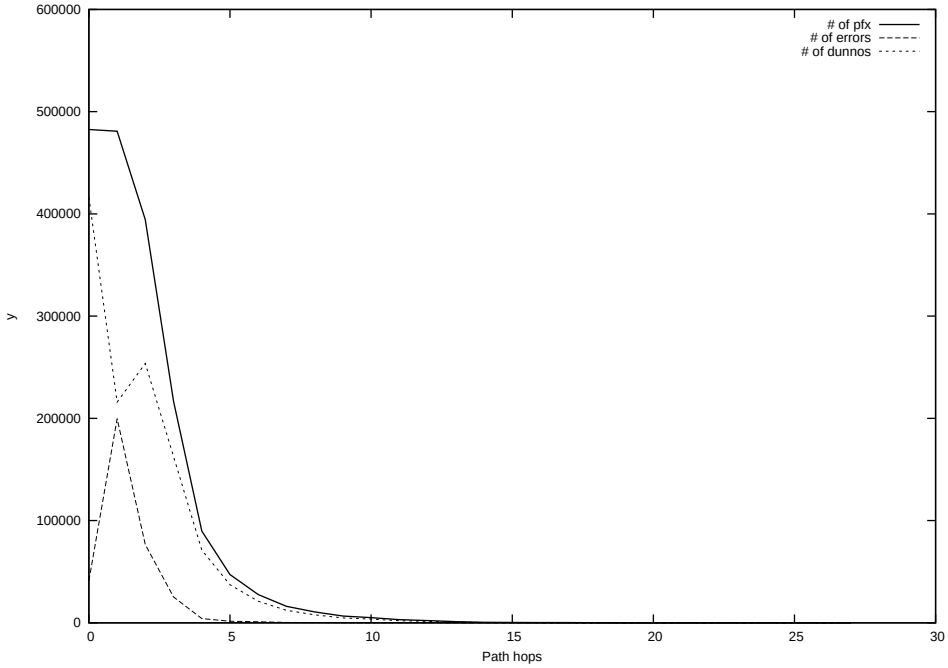
2014-01-13



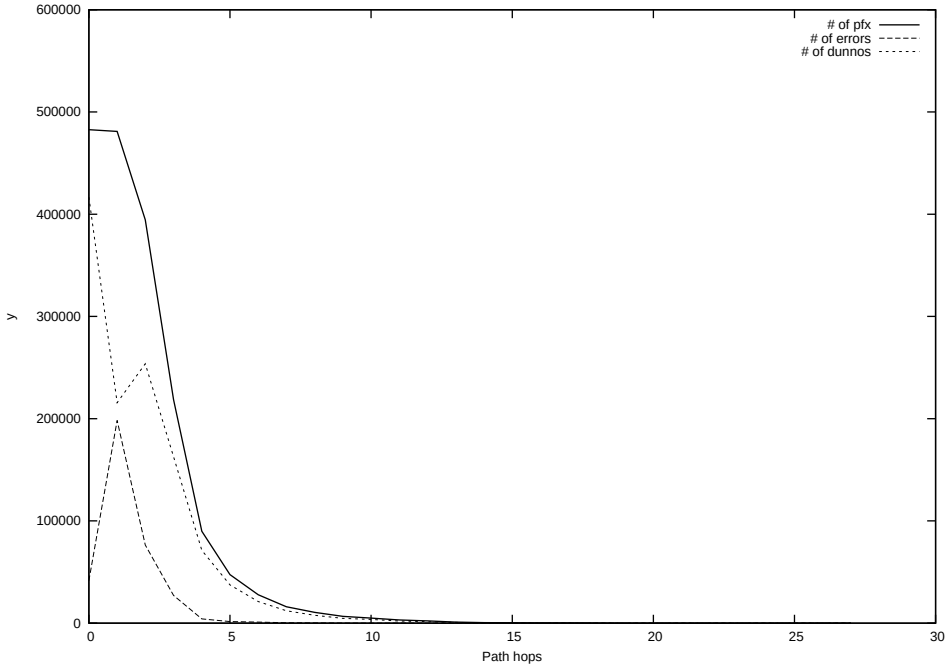
2014-01-14



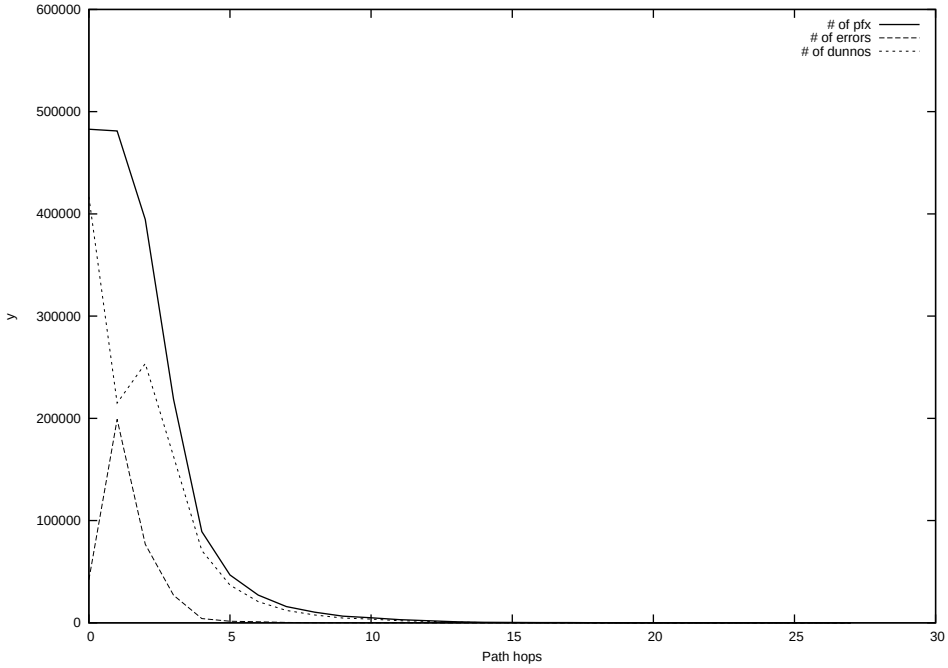
2014-01-15



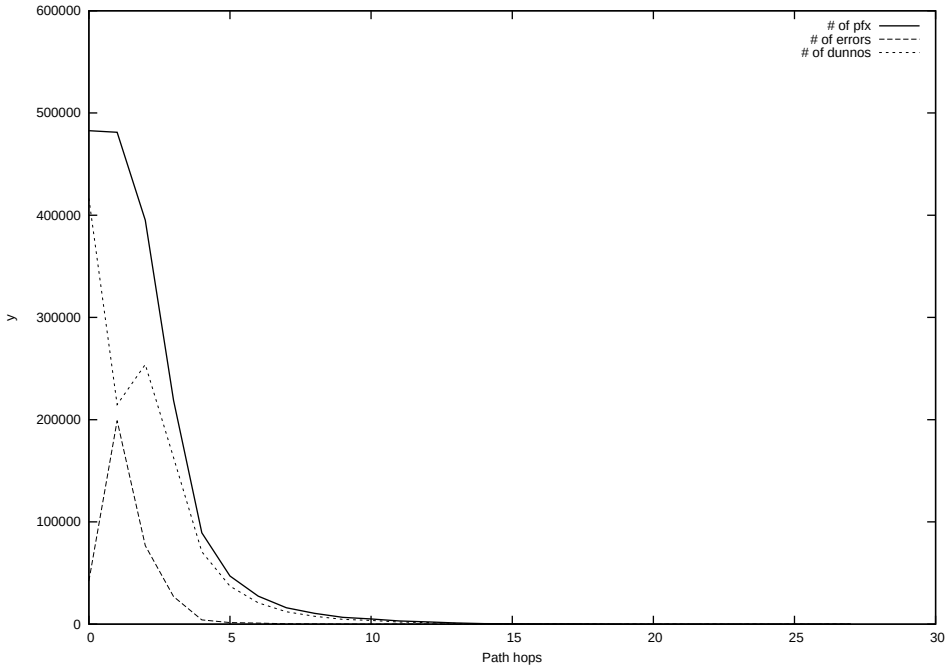
2014-01-16



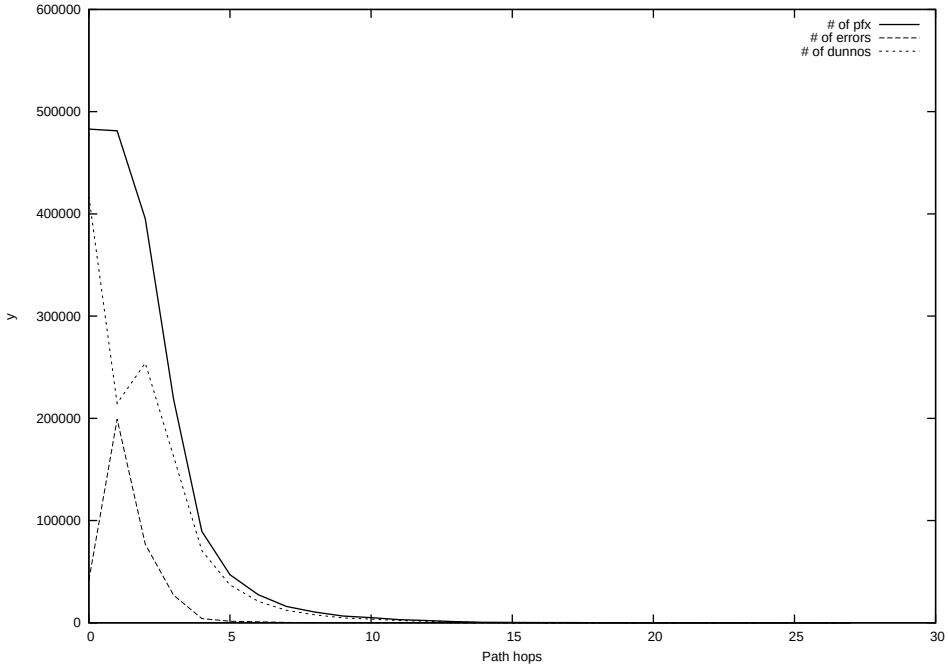
2014-01-17



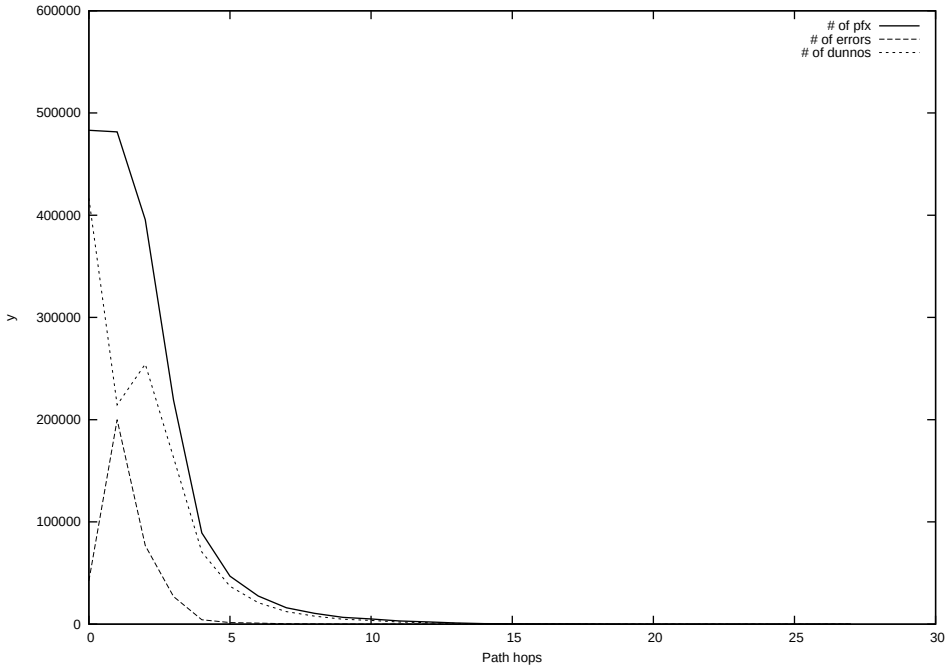
2014-01-18



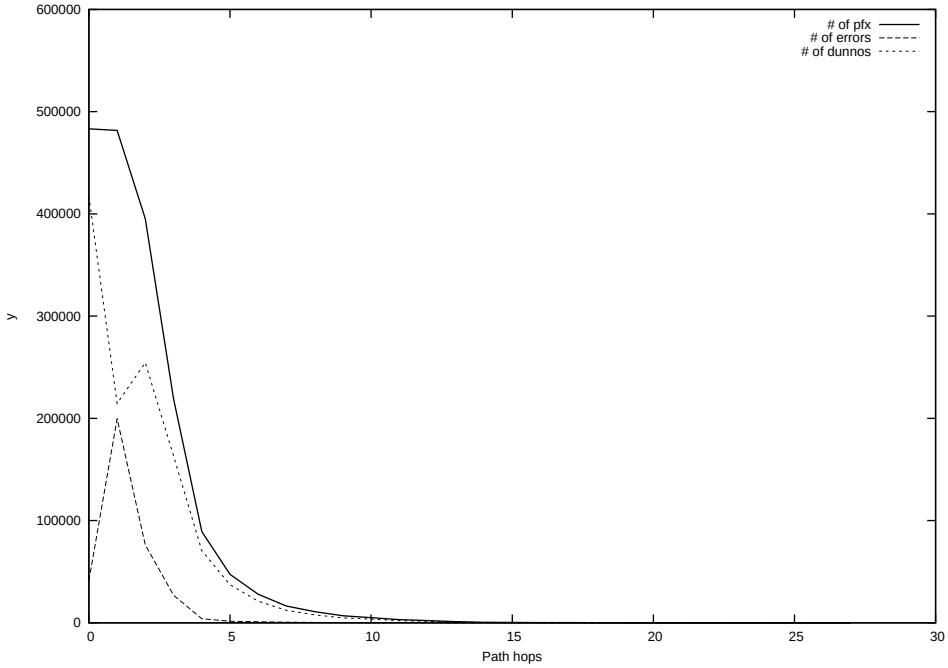
2014-01-19



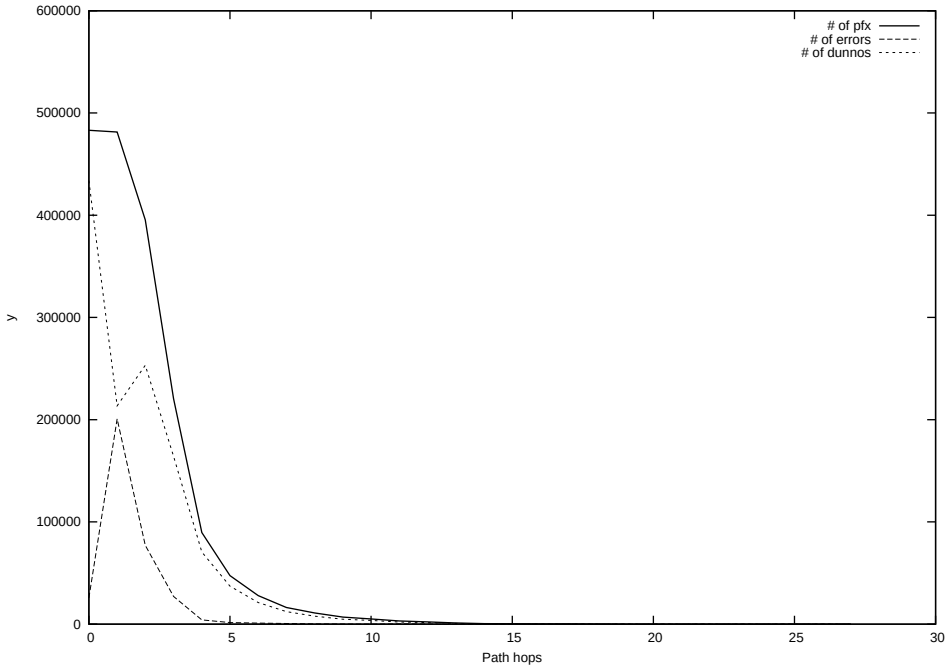
2014-01-20



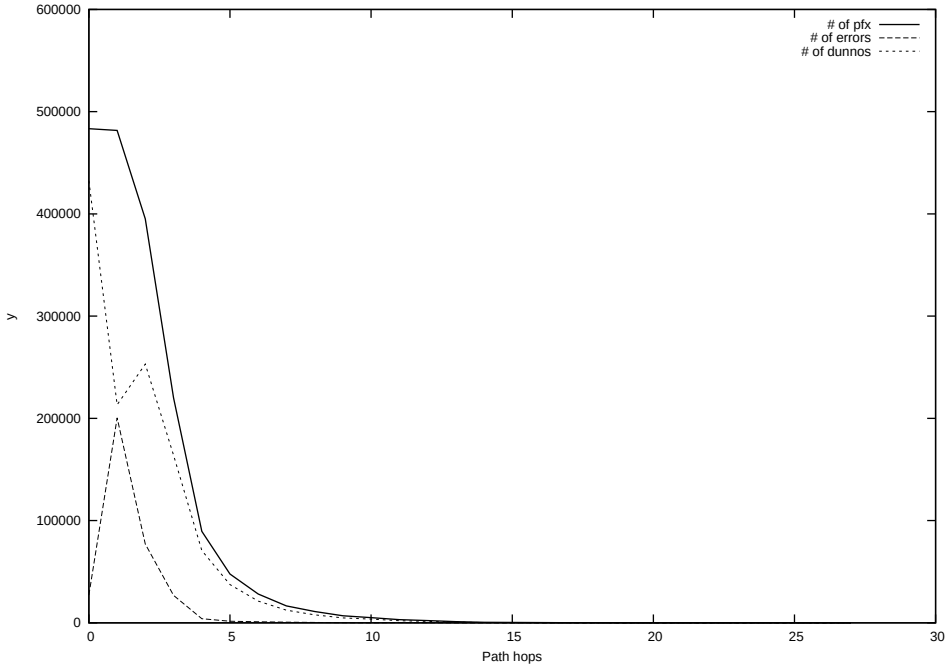
2014-01-21



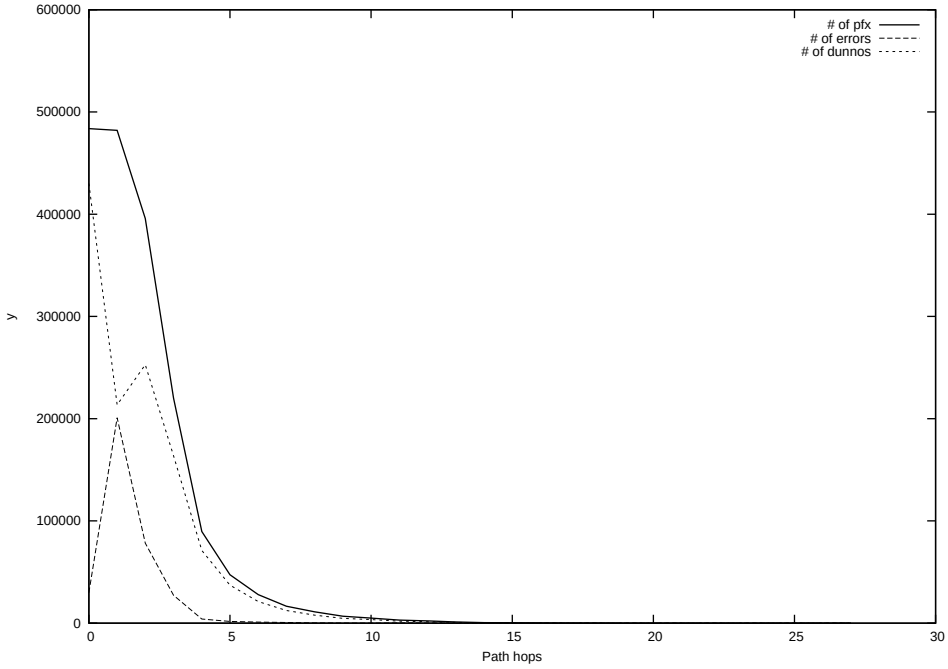
2014-01-22



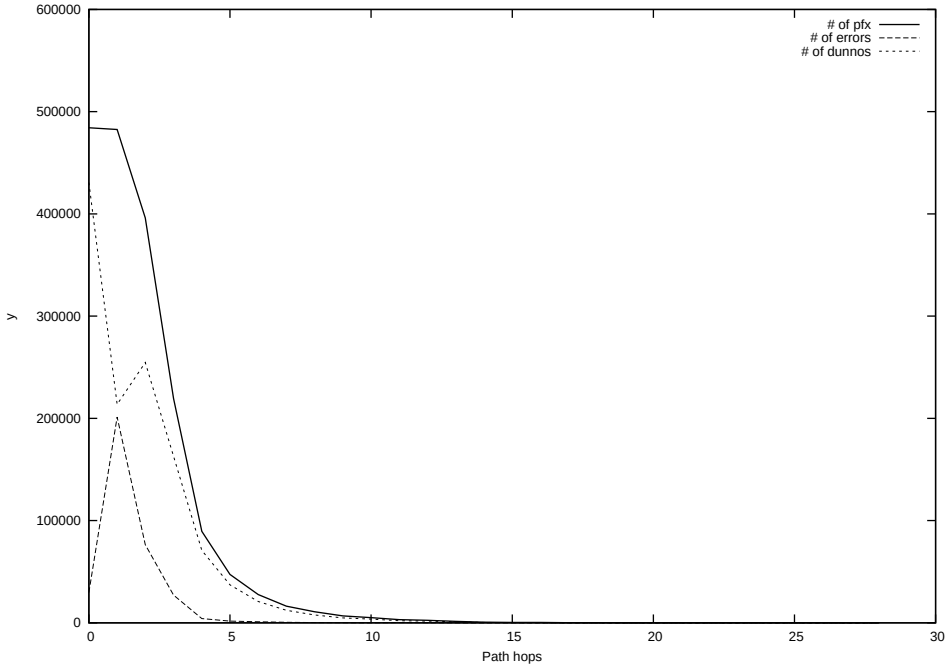
2014-01-23



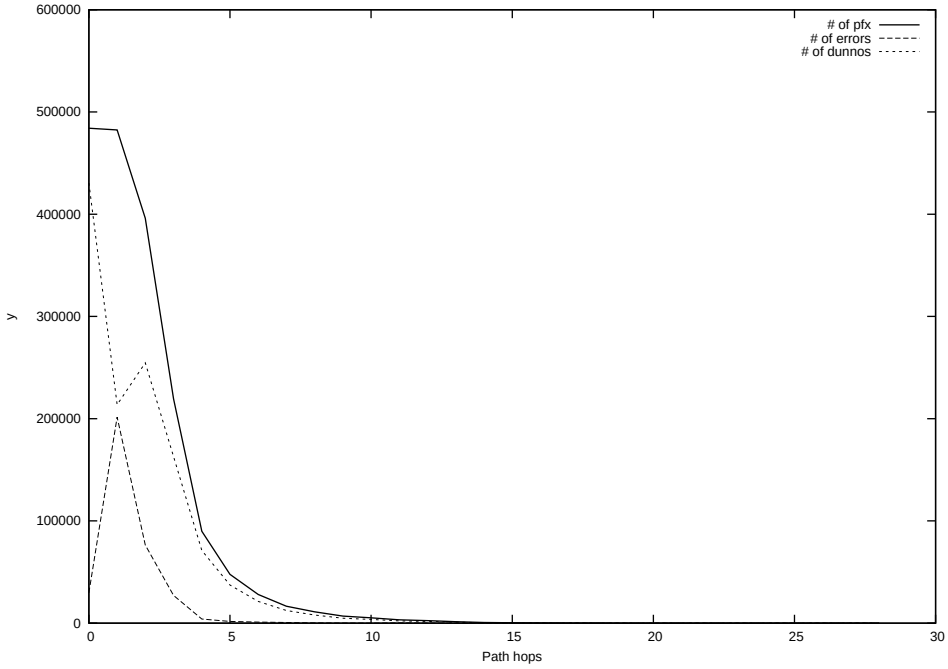
2014-01-24



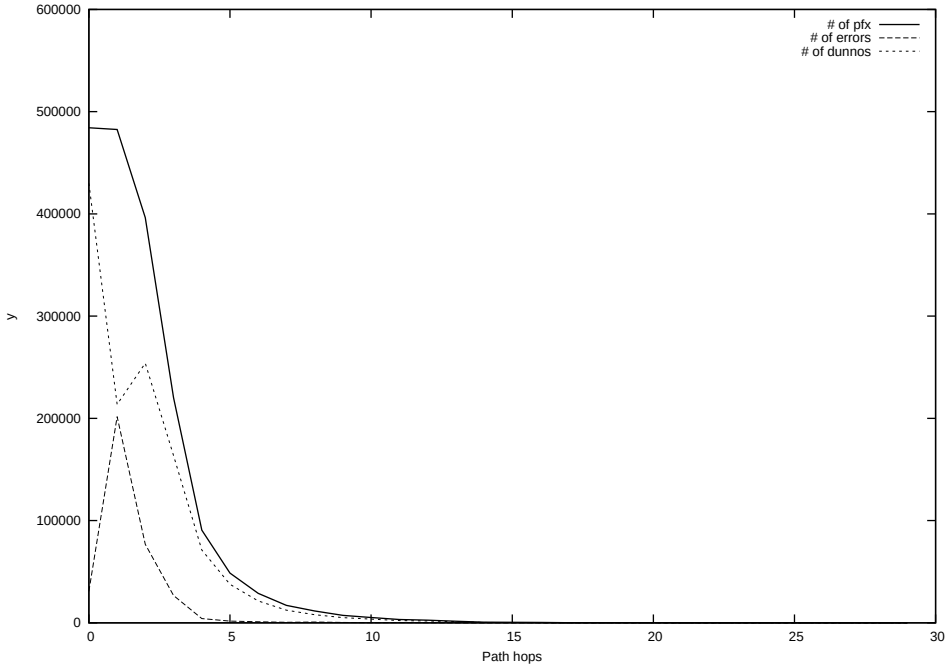
2014-01-25



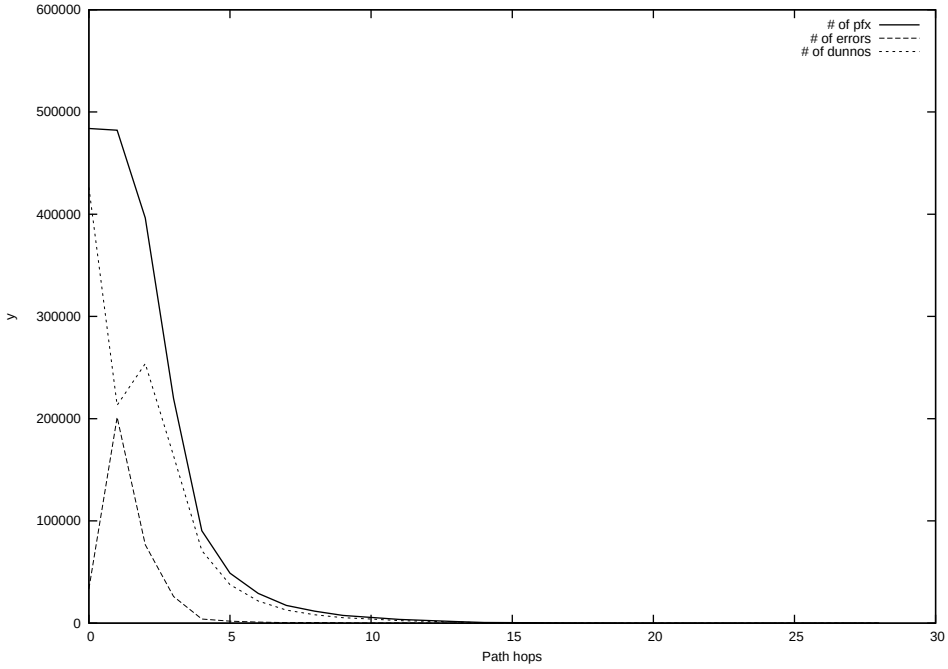
2014-01-26



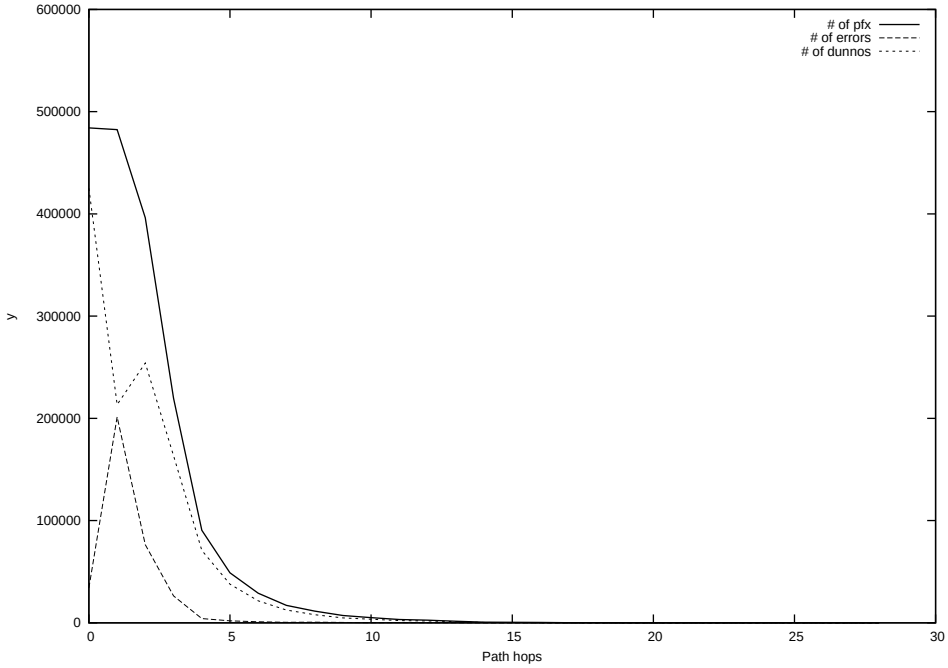
2014-01-27



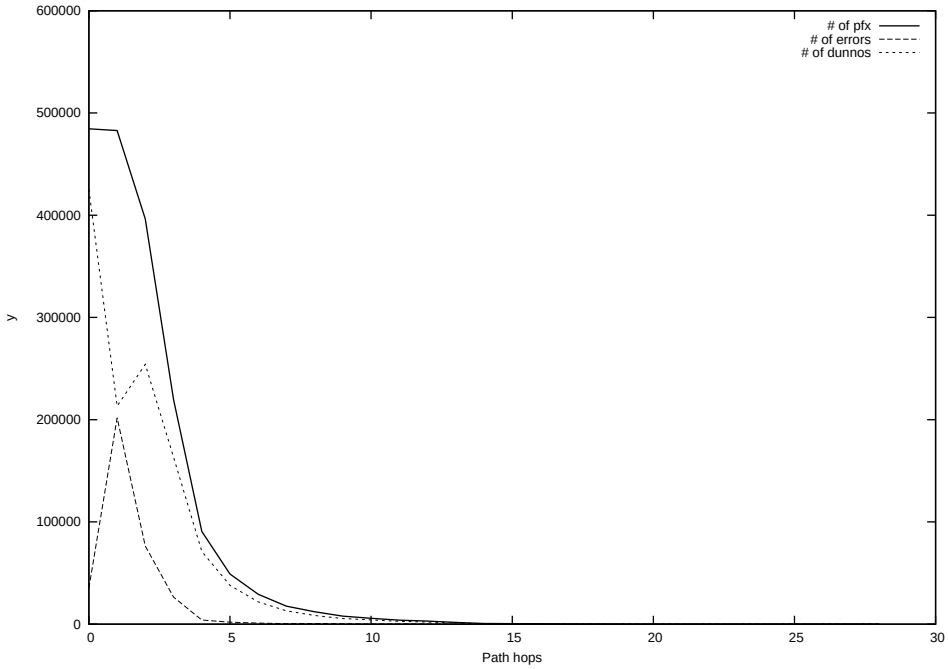
2014-01-28



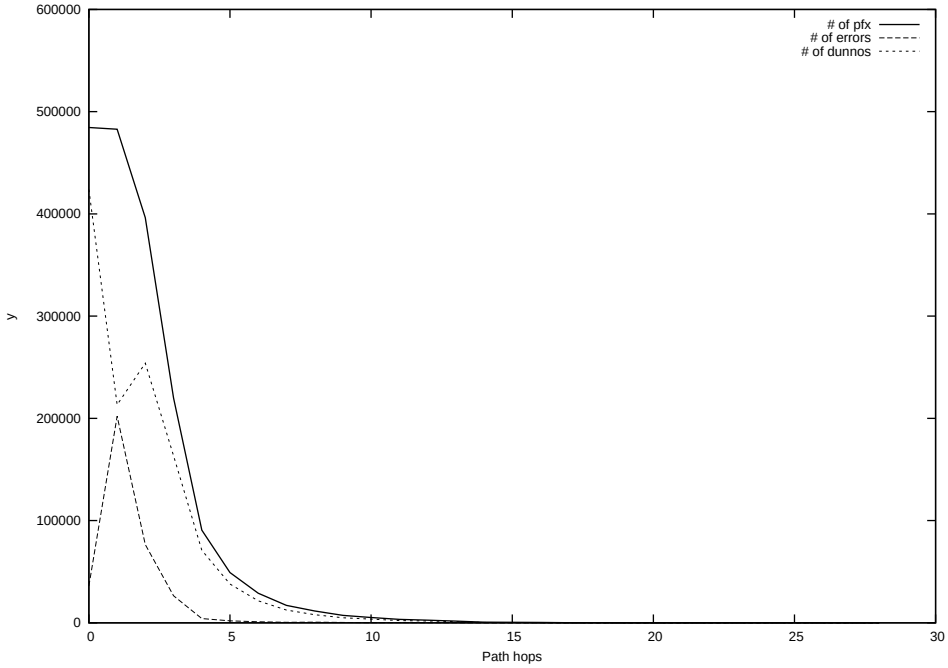
2014-01-29



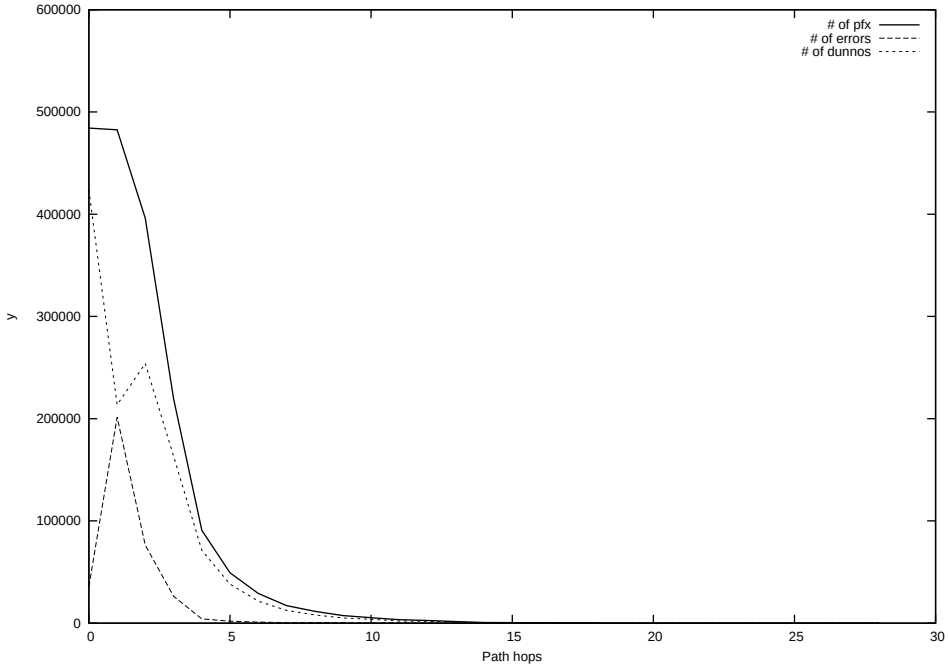
2014-01-30



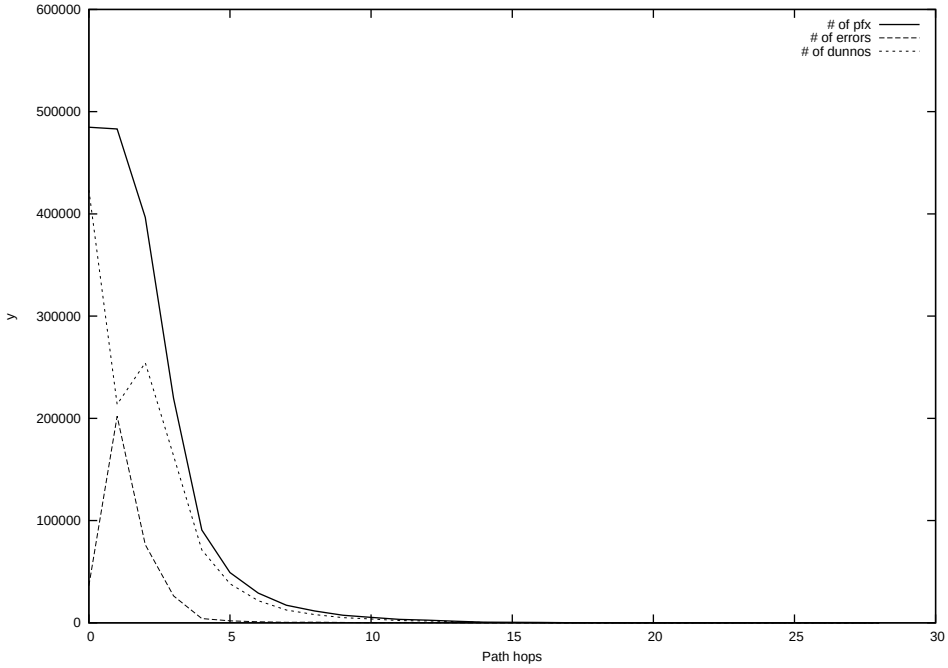
2014-01-31



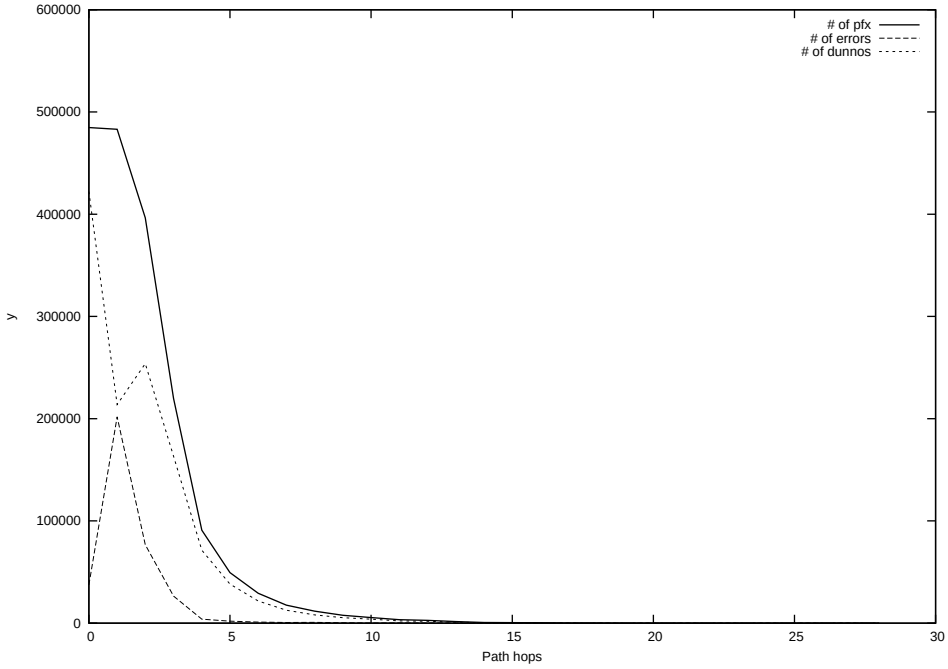
2014-02-01



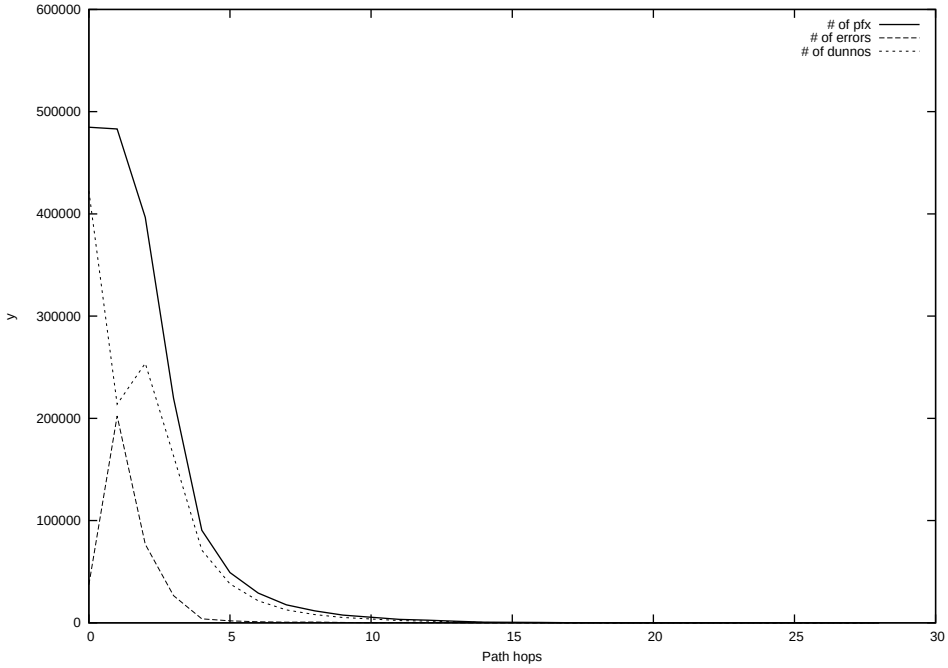
2014-02-02



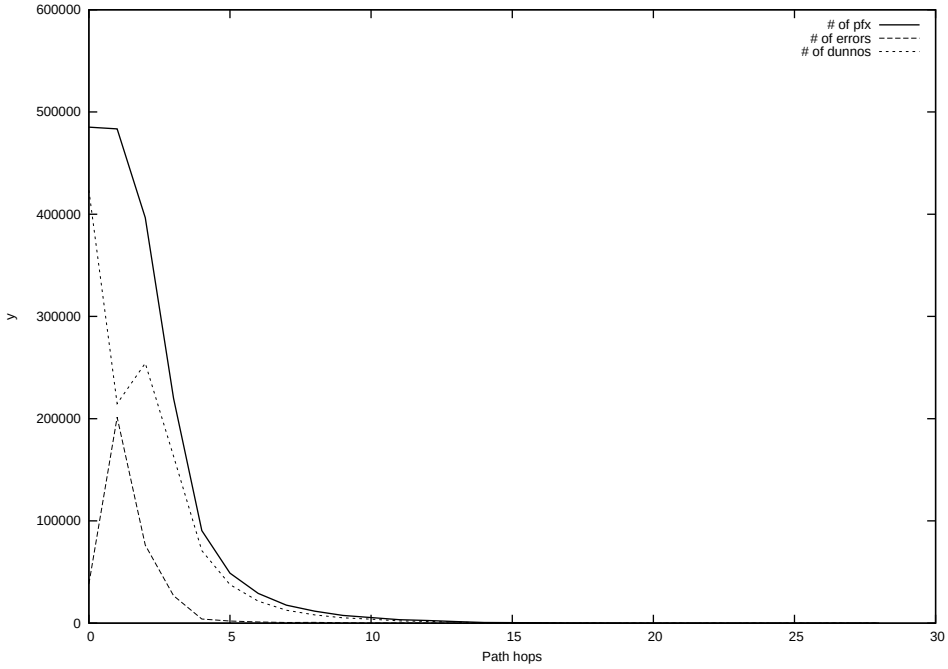
2014-02-03



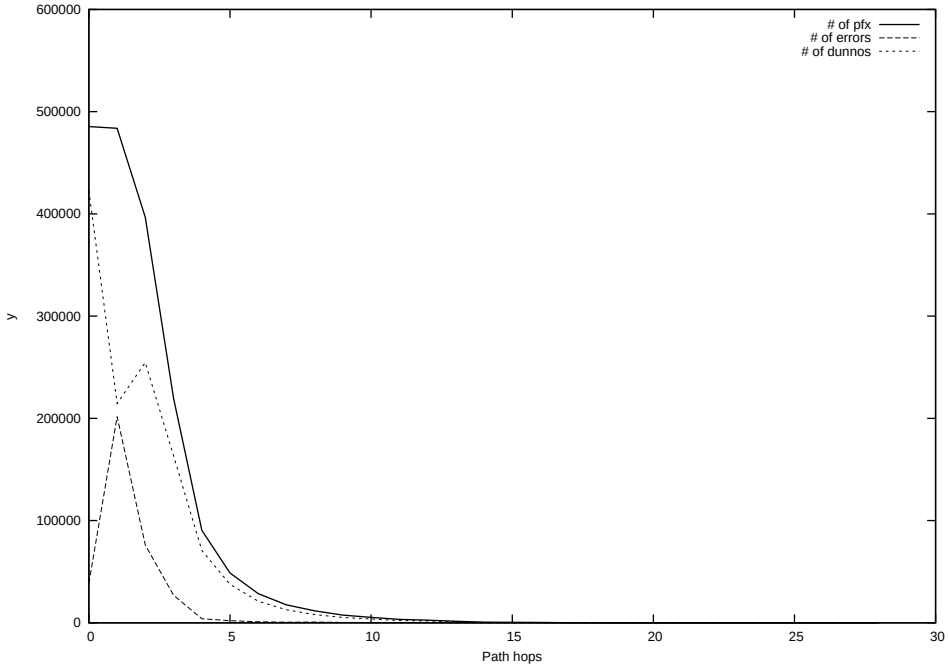
2014-02-04



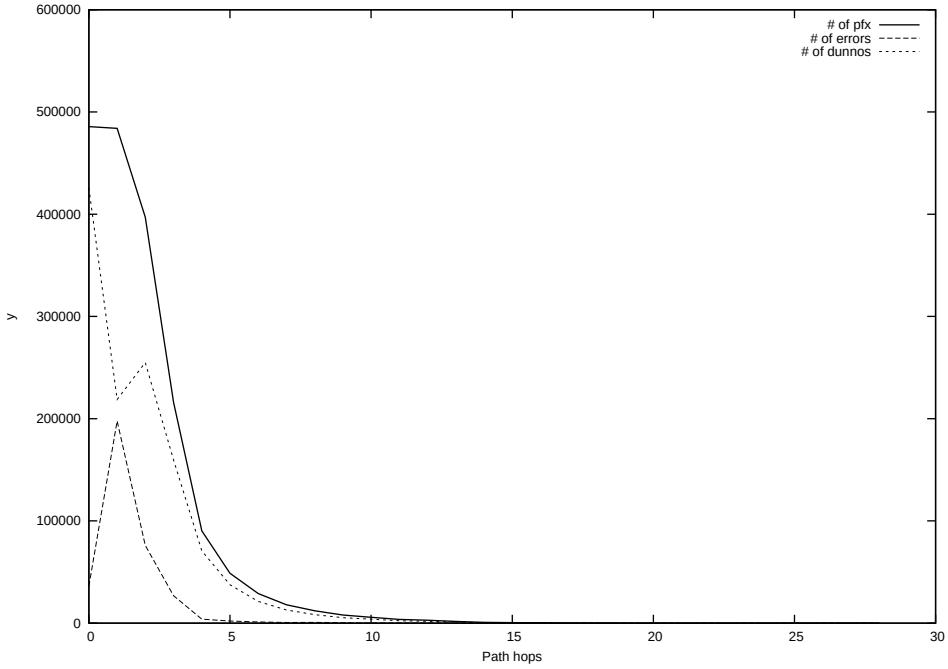
2014-02-05



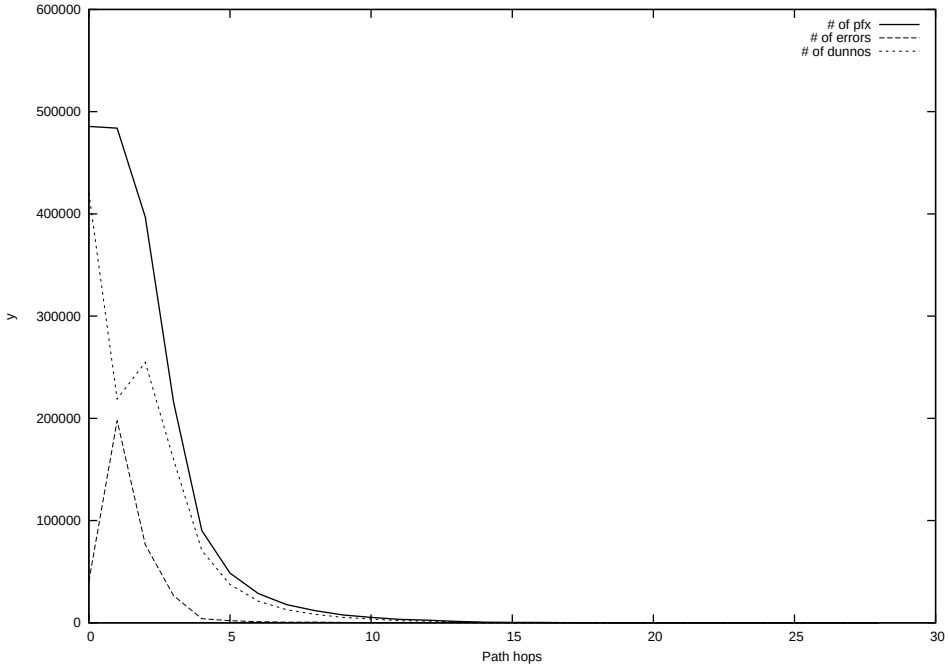
2014-02-06



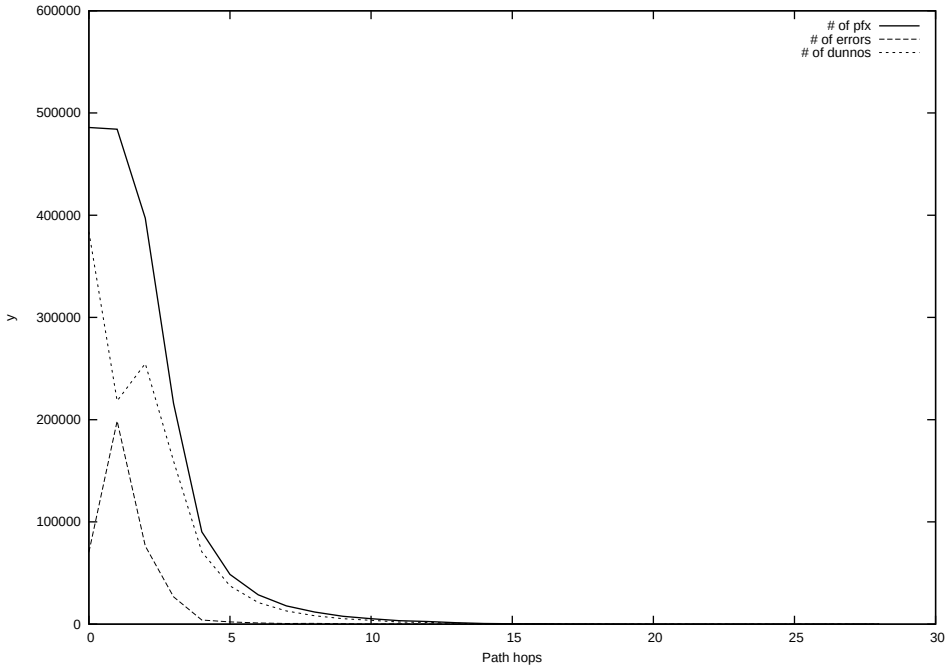
2014-02-07



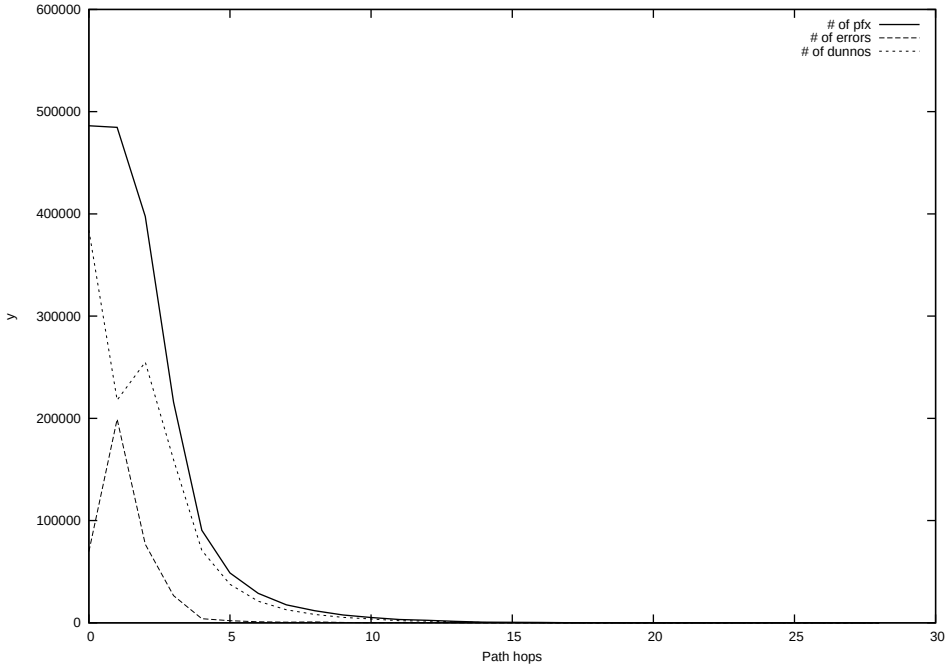
2014-02-08



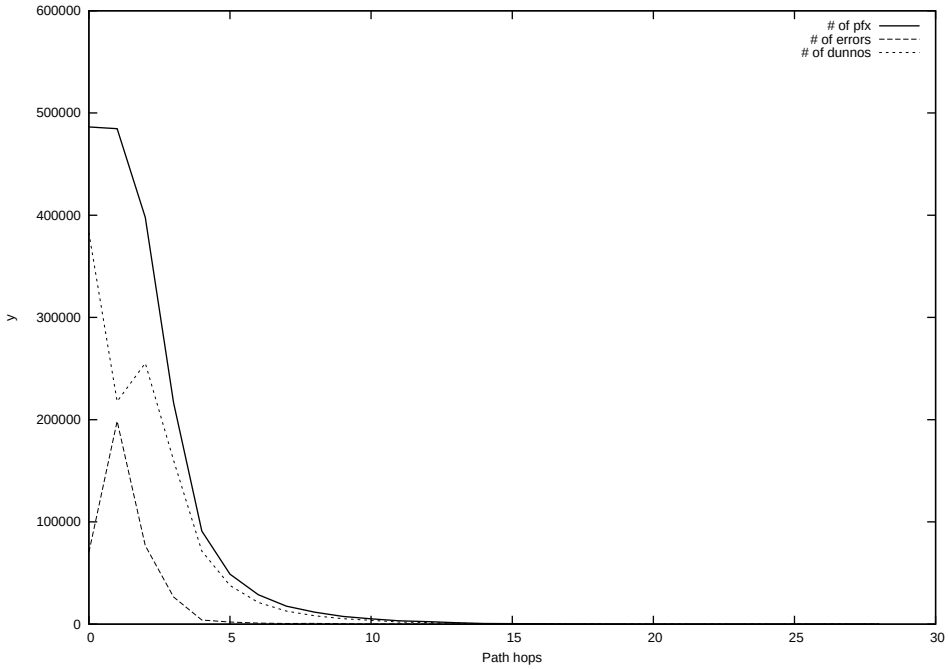
2014-02-09



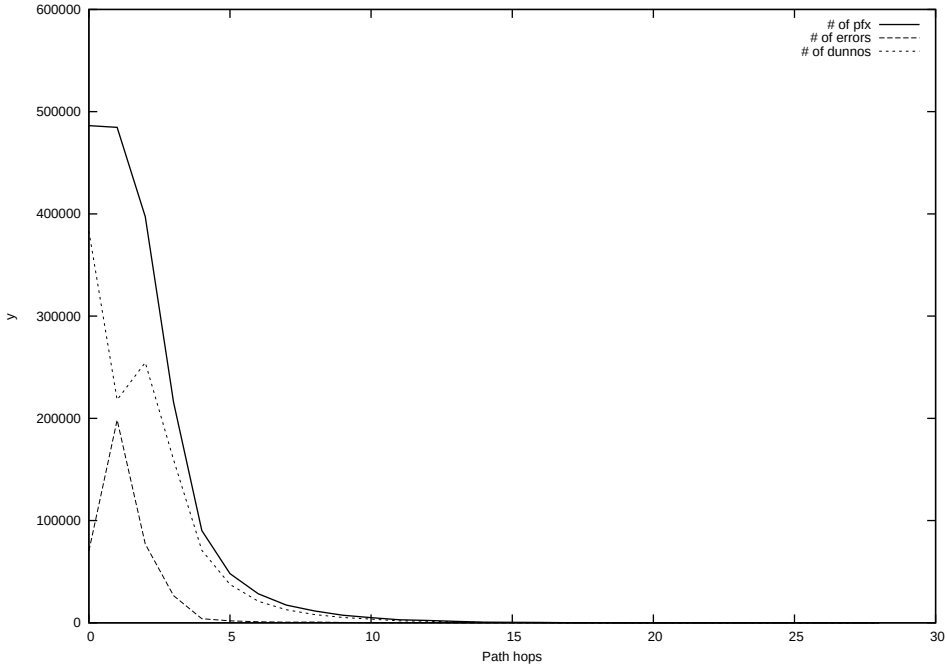
2014-02-10



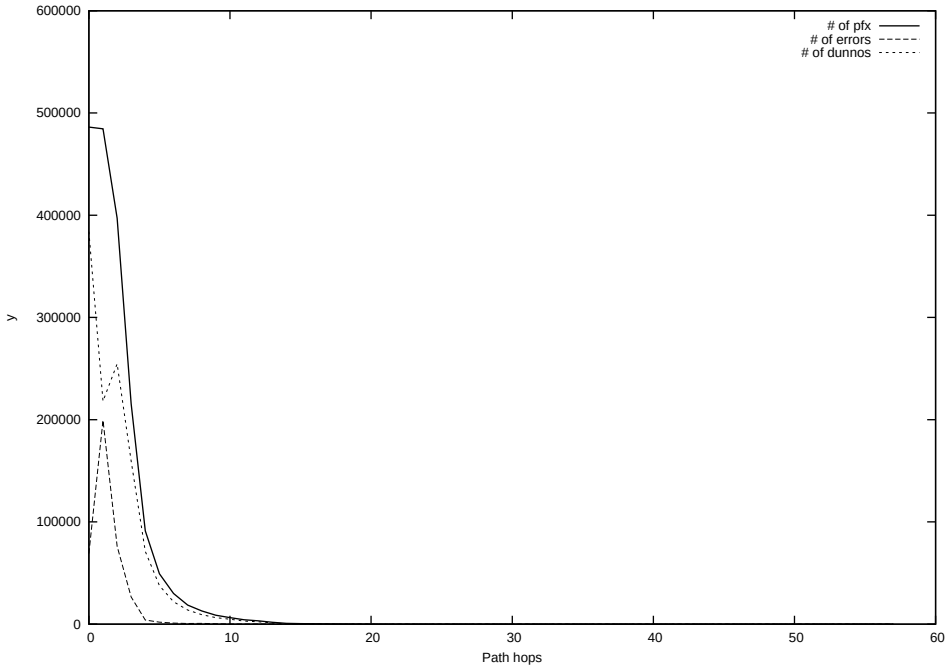
2014-02-11



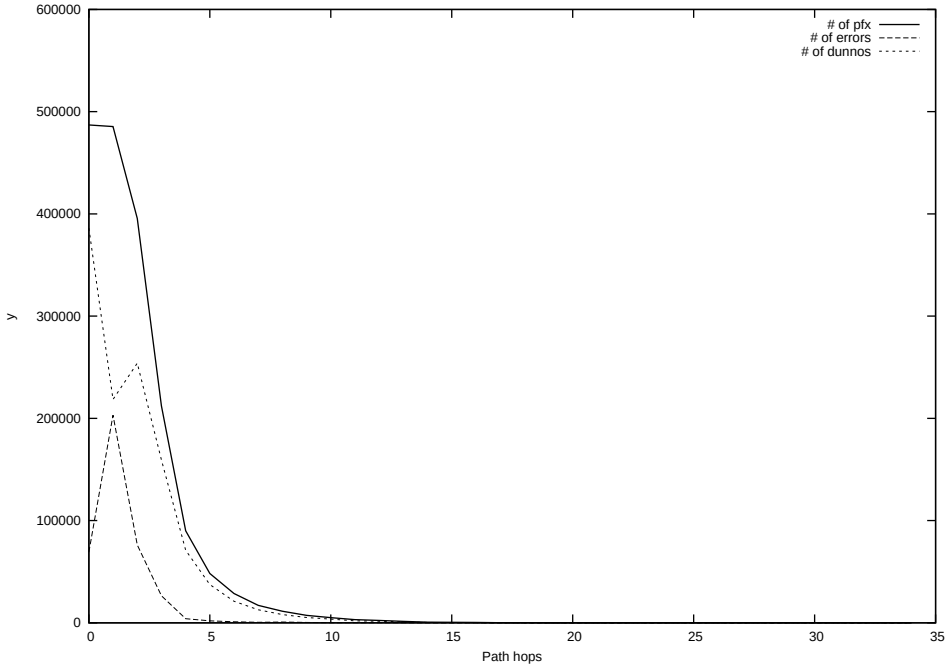
2014-02-12



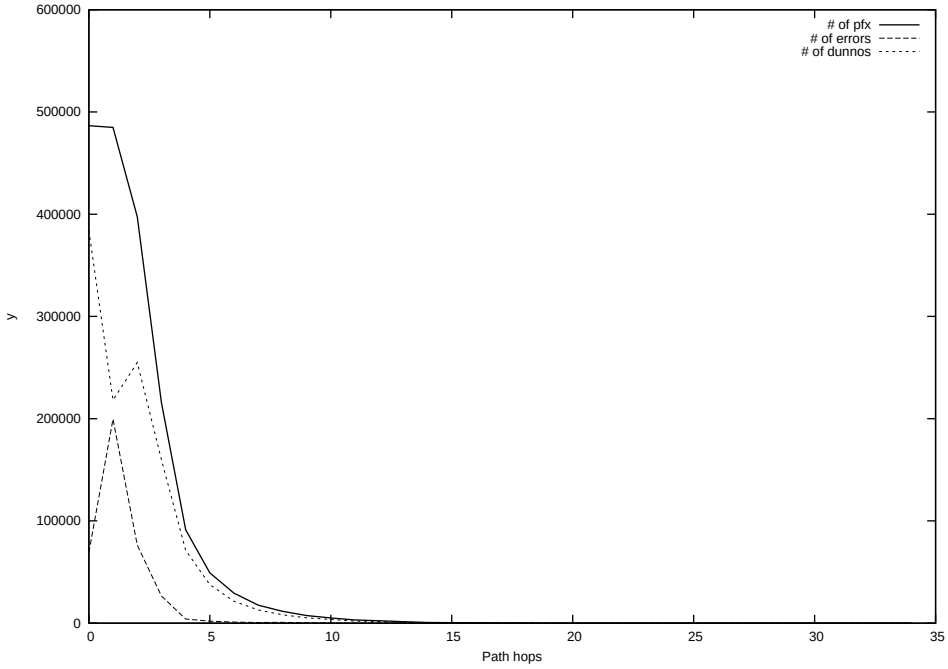
2014-02-13



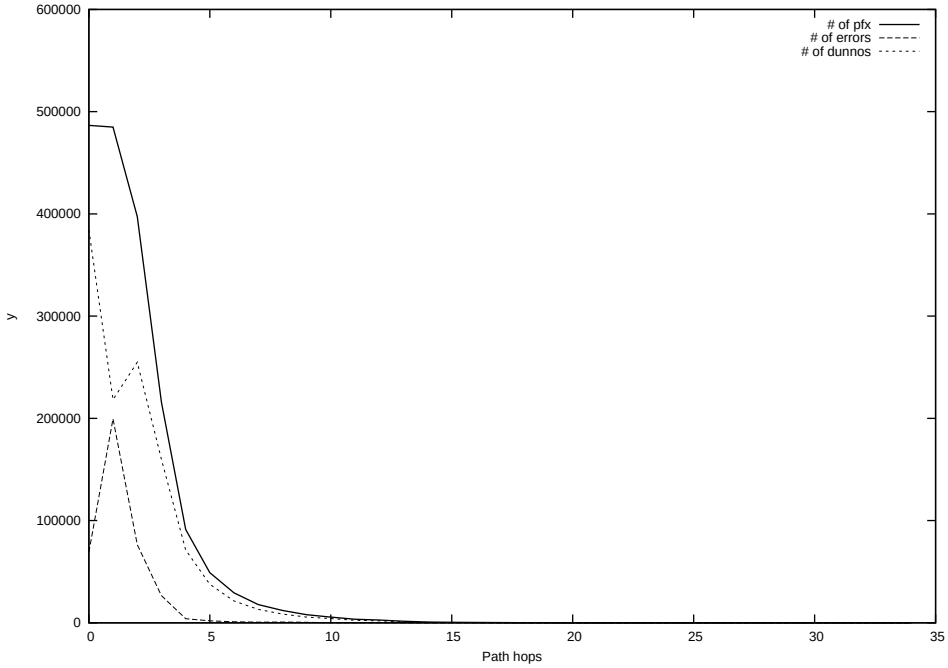
2014-02-14



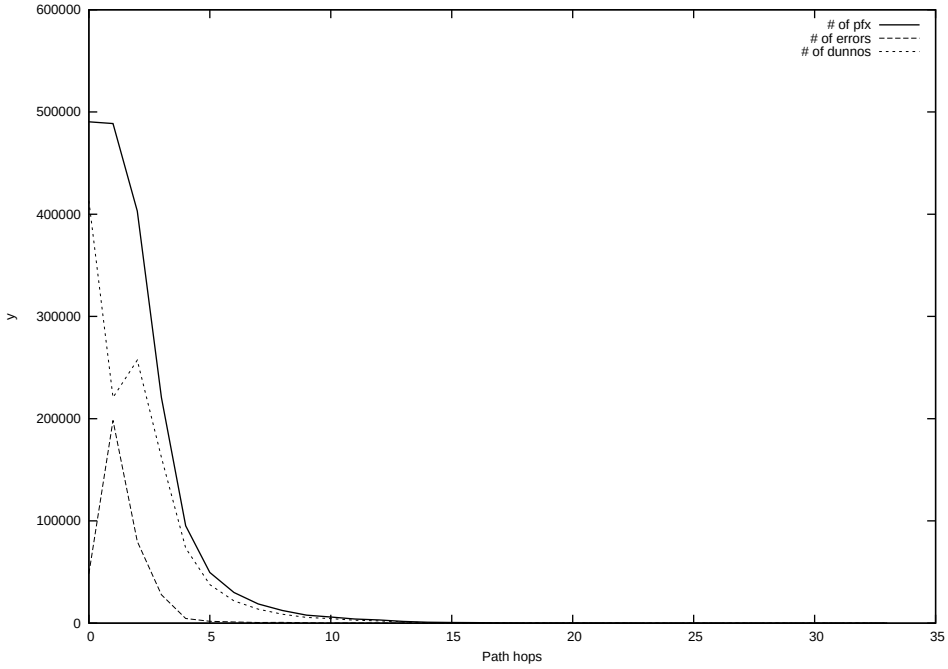
2014-02-15



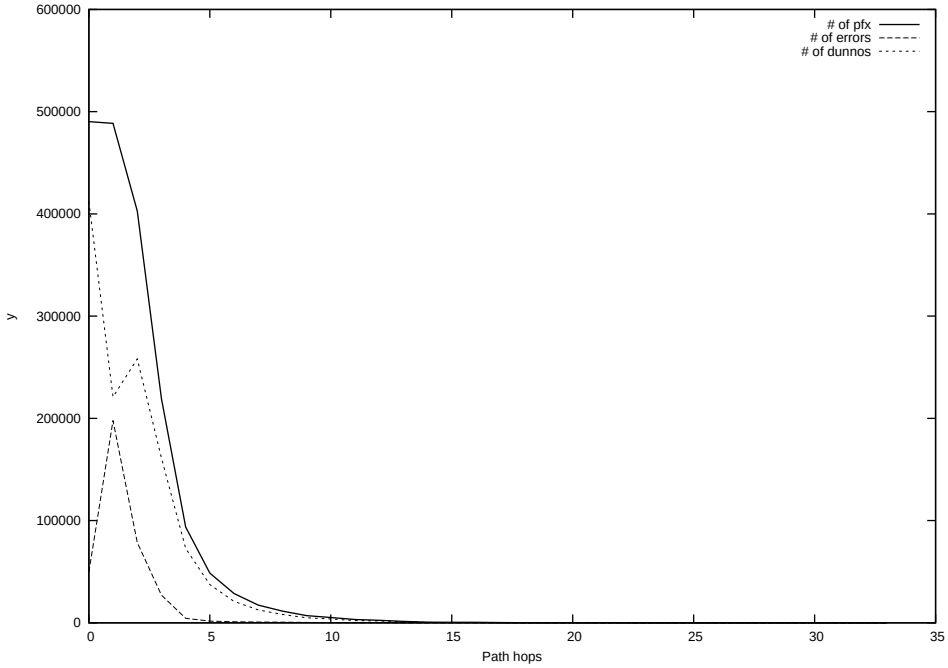
2014-02-16



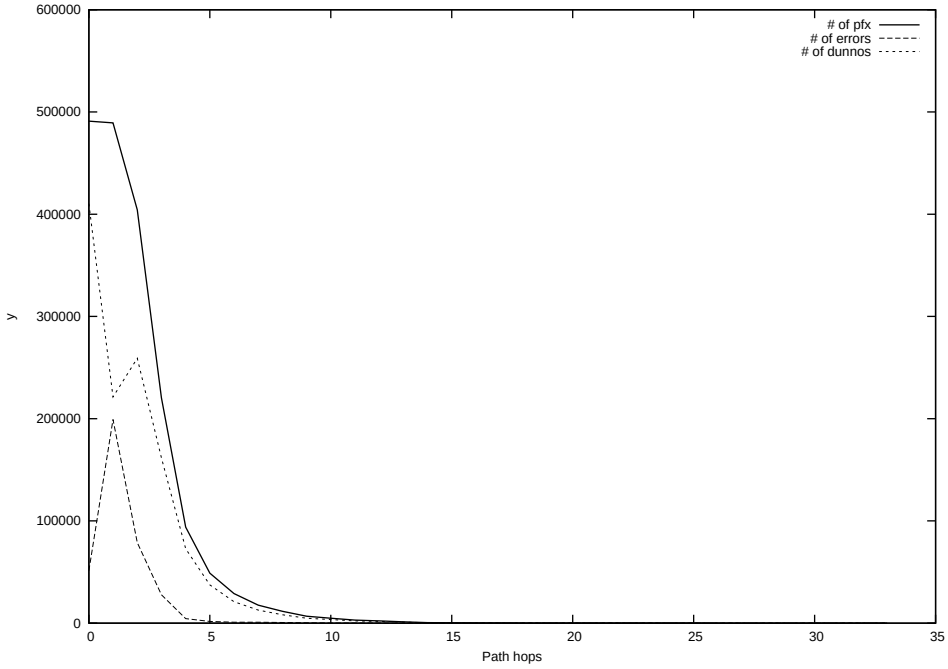
2014-02-17



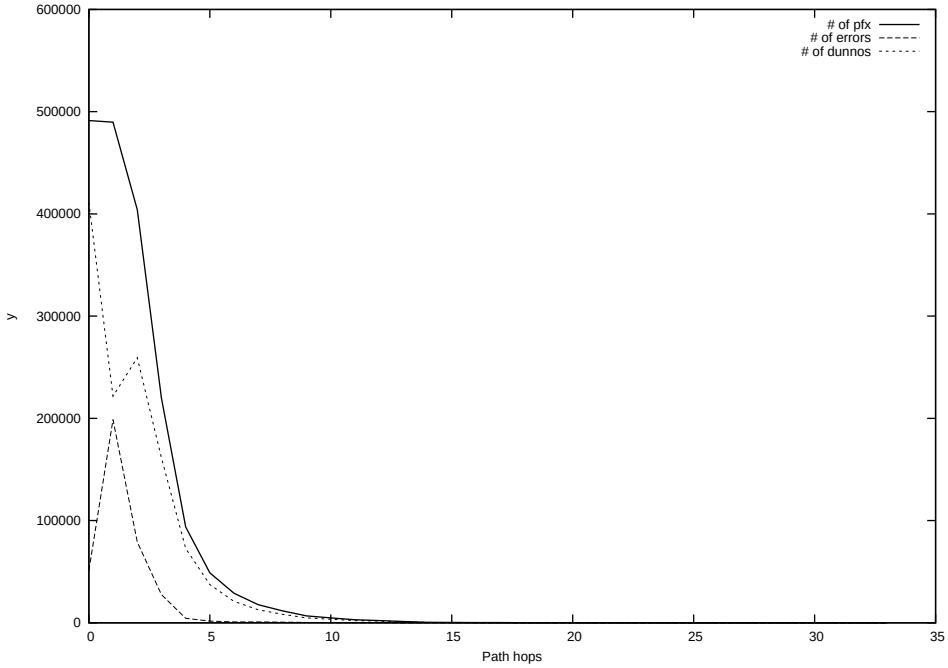
2014-03-20



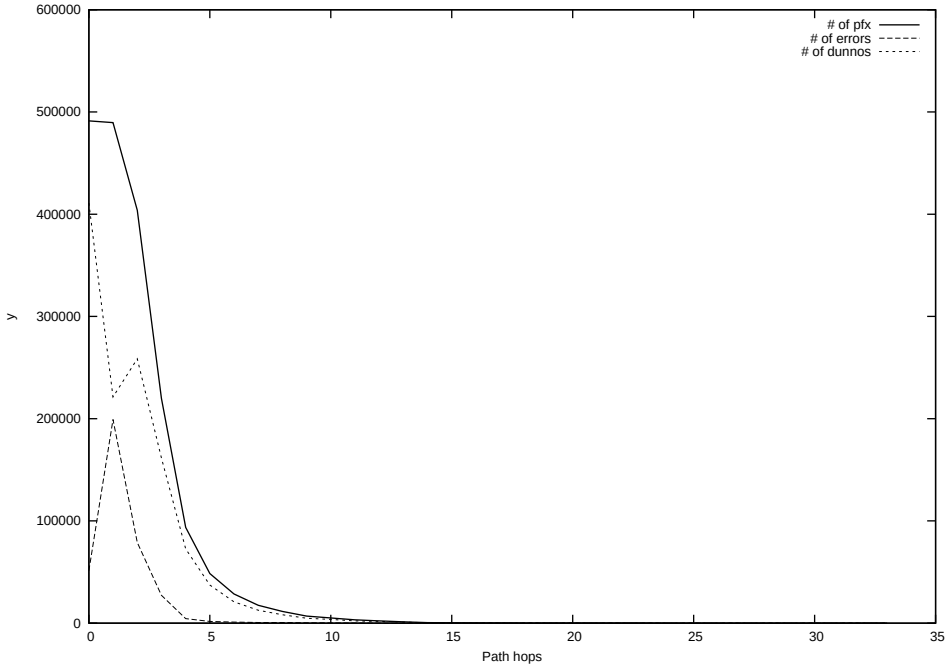
2014-03-21



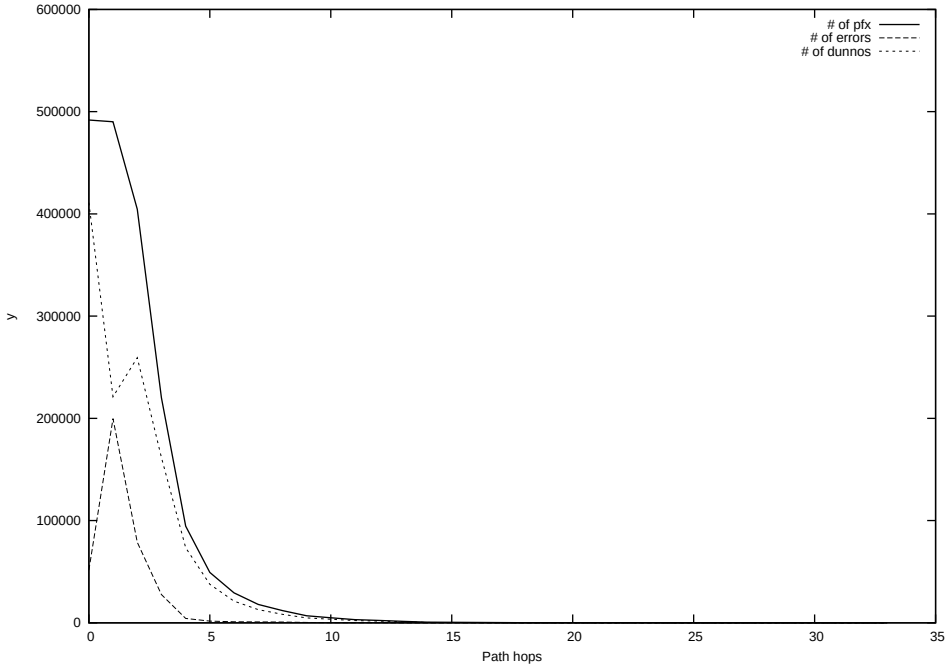
2014-03-22



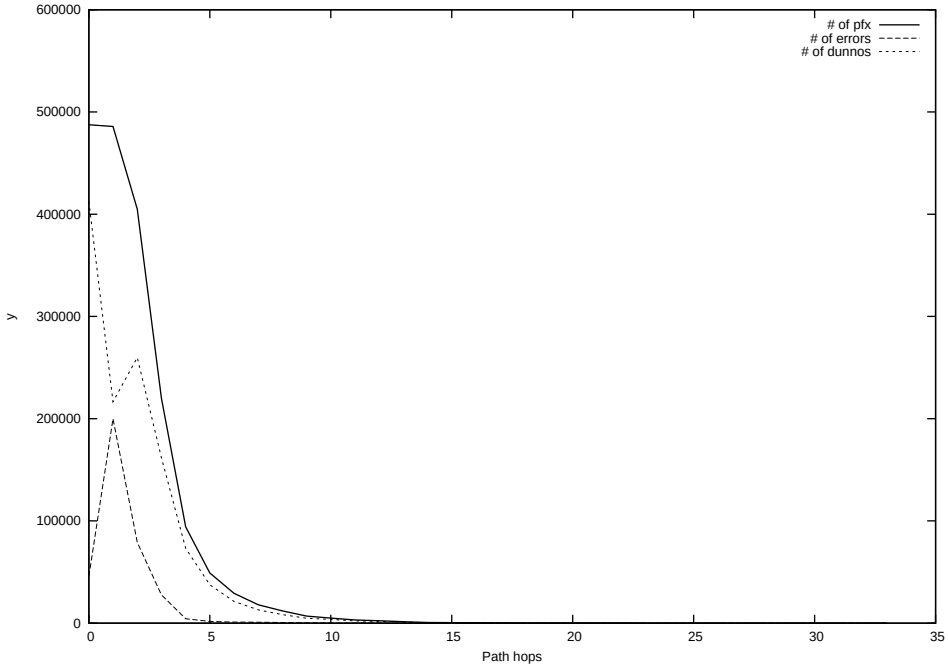
2014-03-23



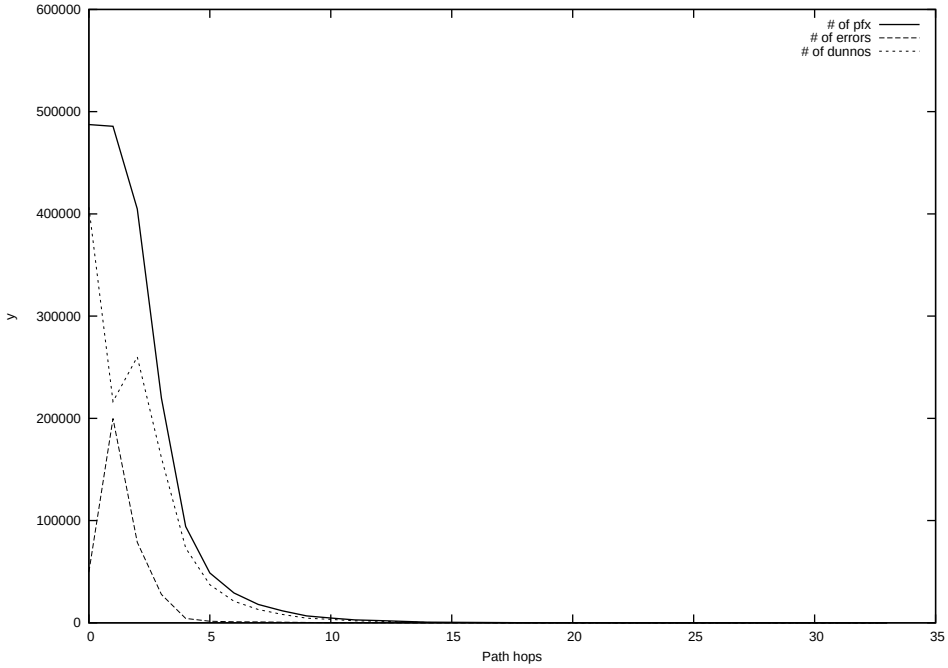
2014-03-24



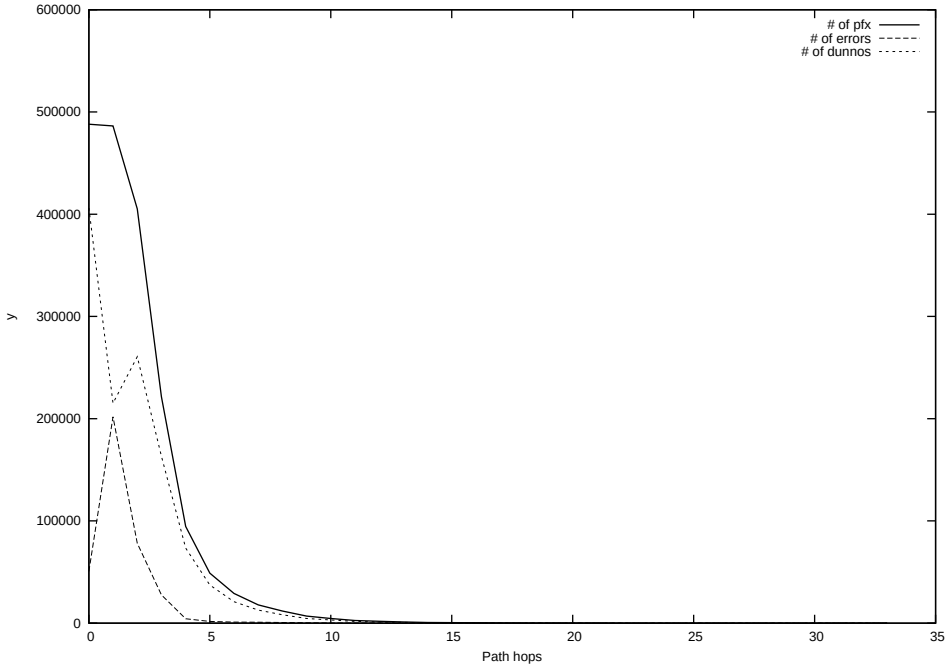
2014-03-25



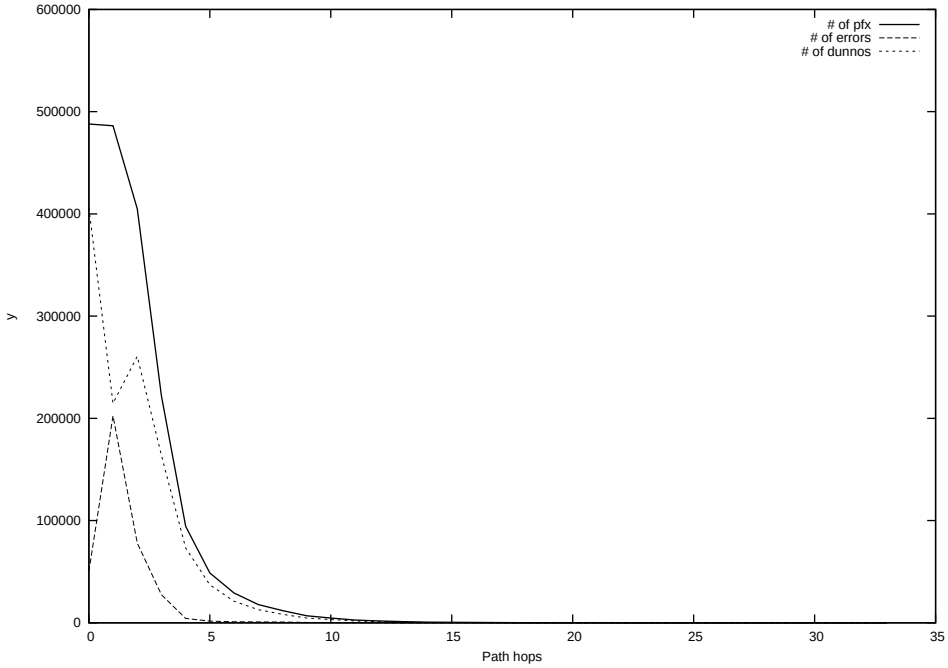
2014-03-26



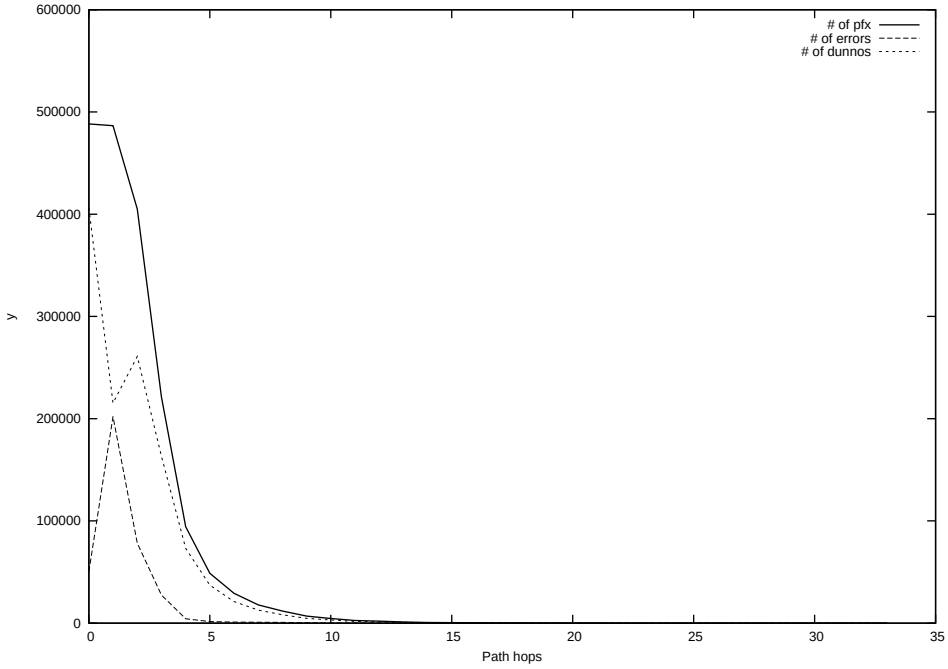
2014-03-27



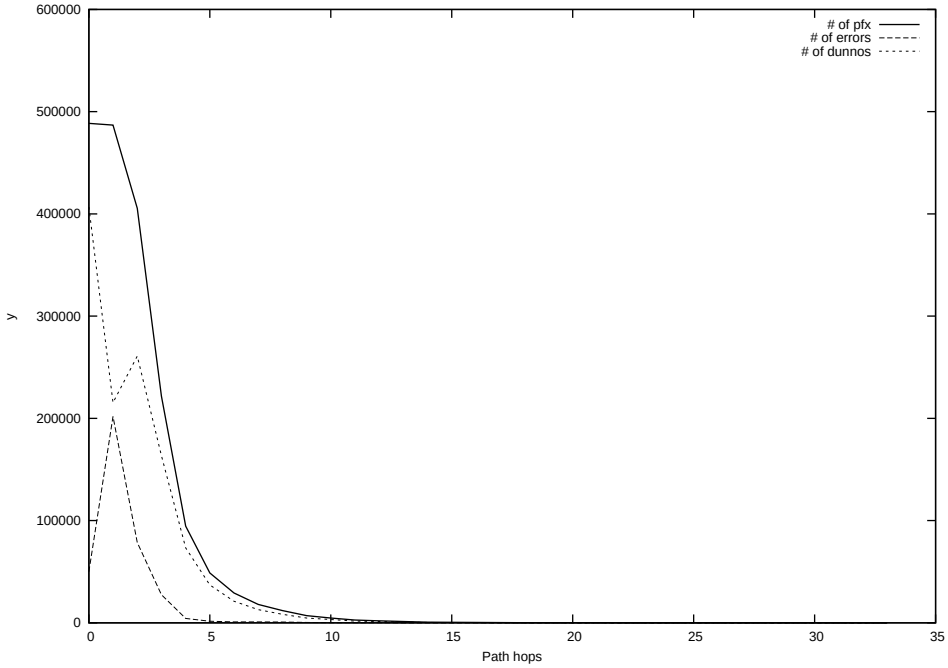
2014-03-28



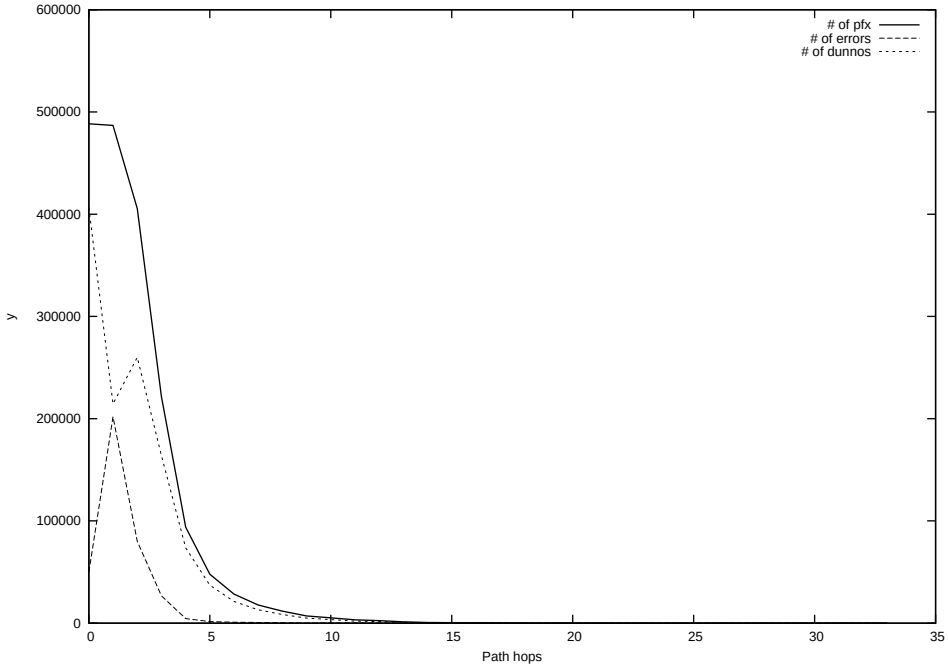
2014-03-29



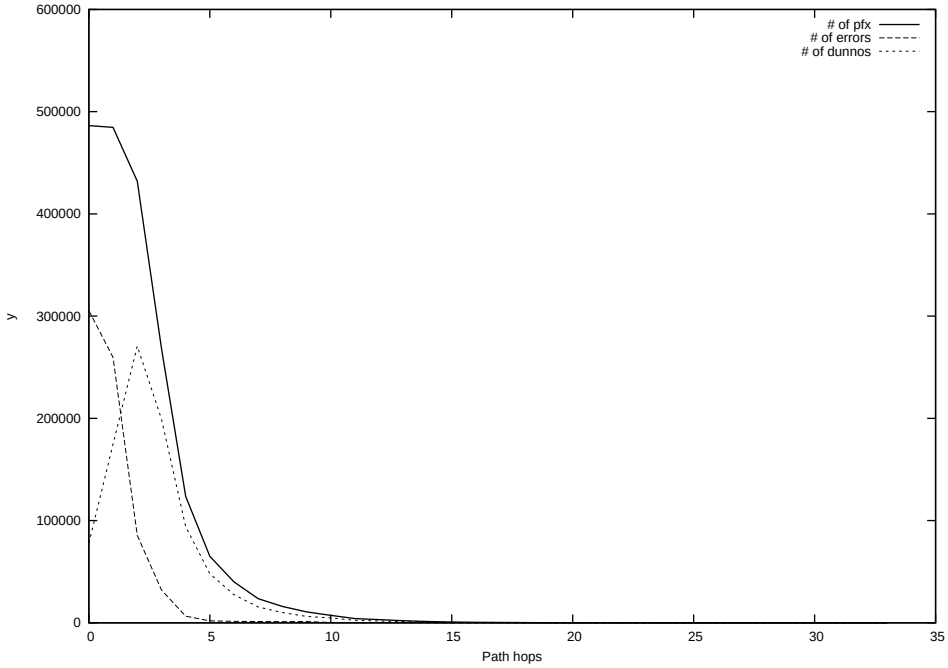
2014-03-30



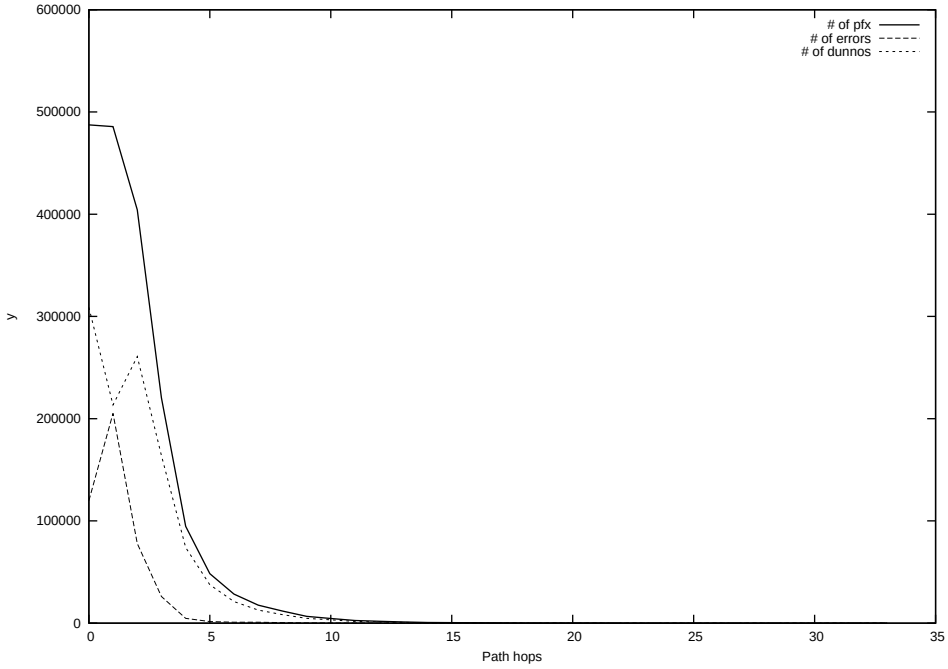
2014-03-31



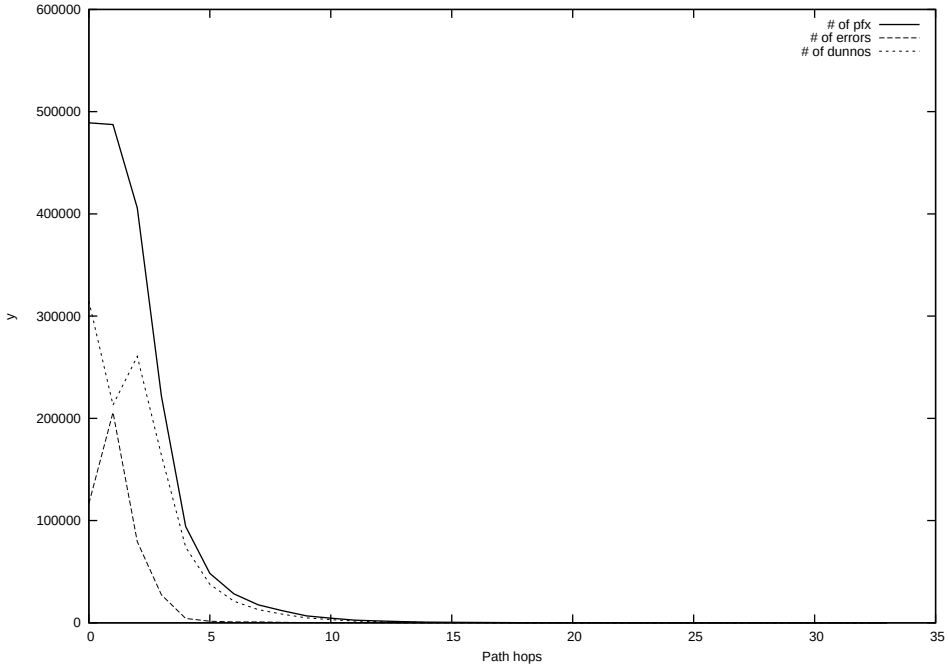
2014-04-01



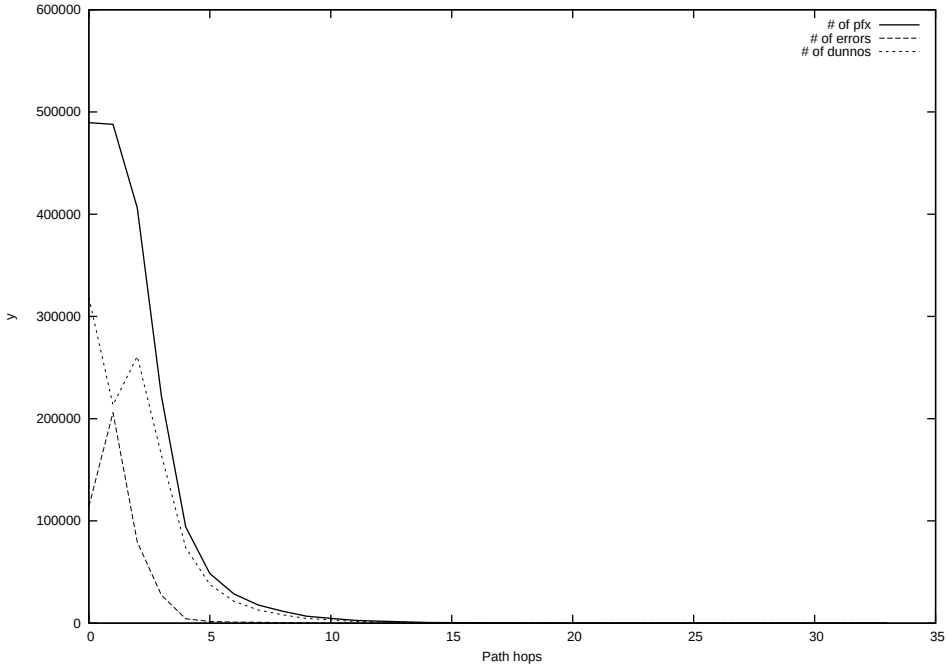
2014-04-02



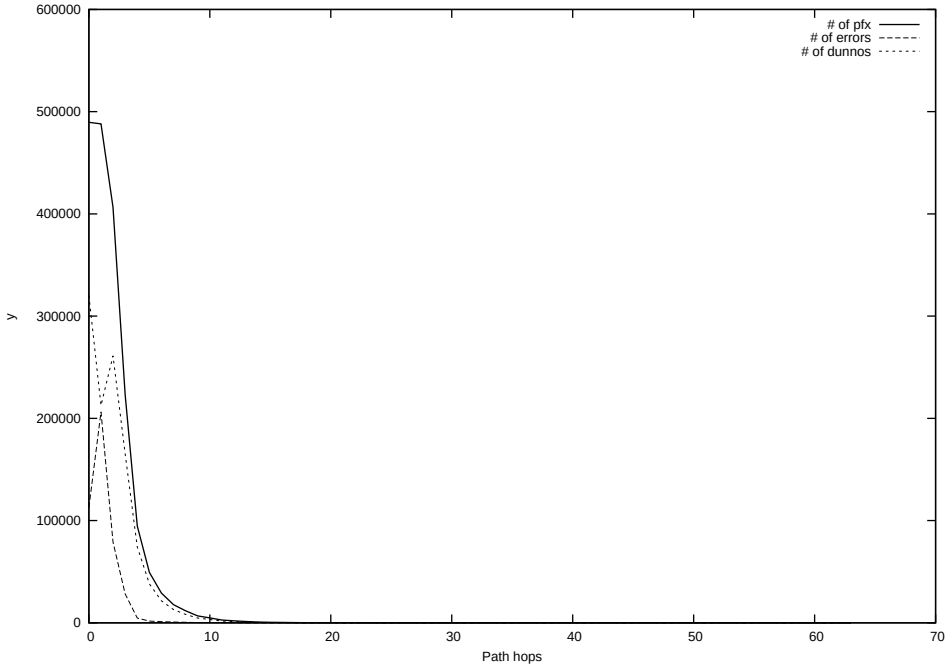
2014-04-03



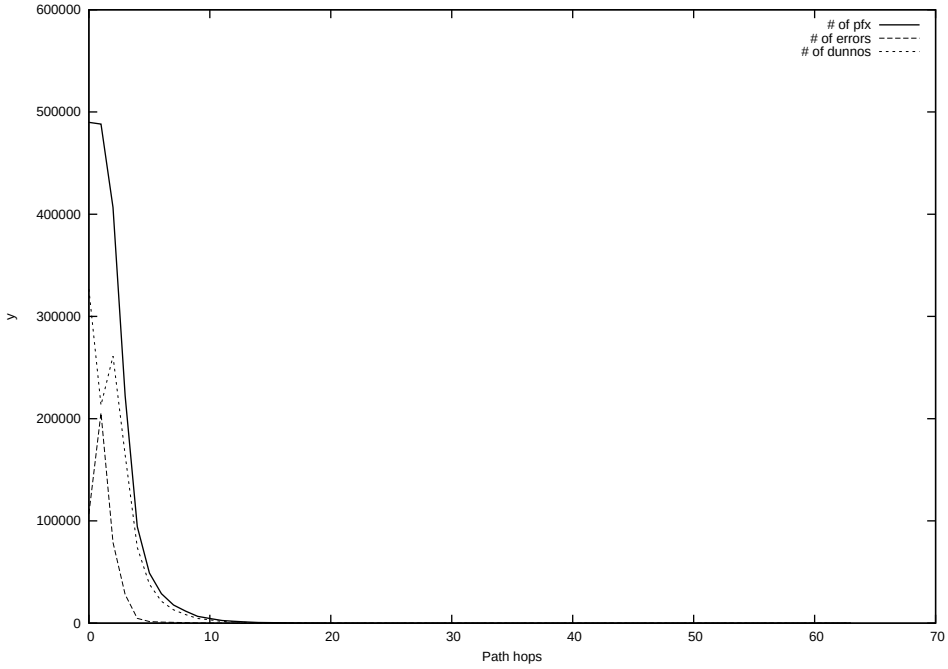
2014-04-04



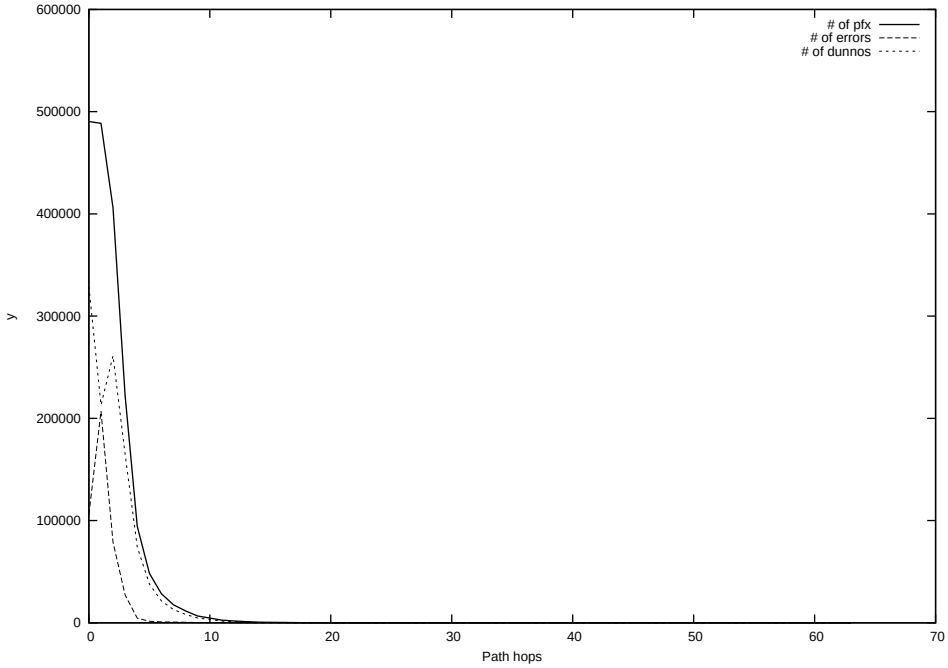
2014-04-05



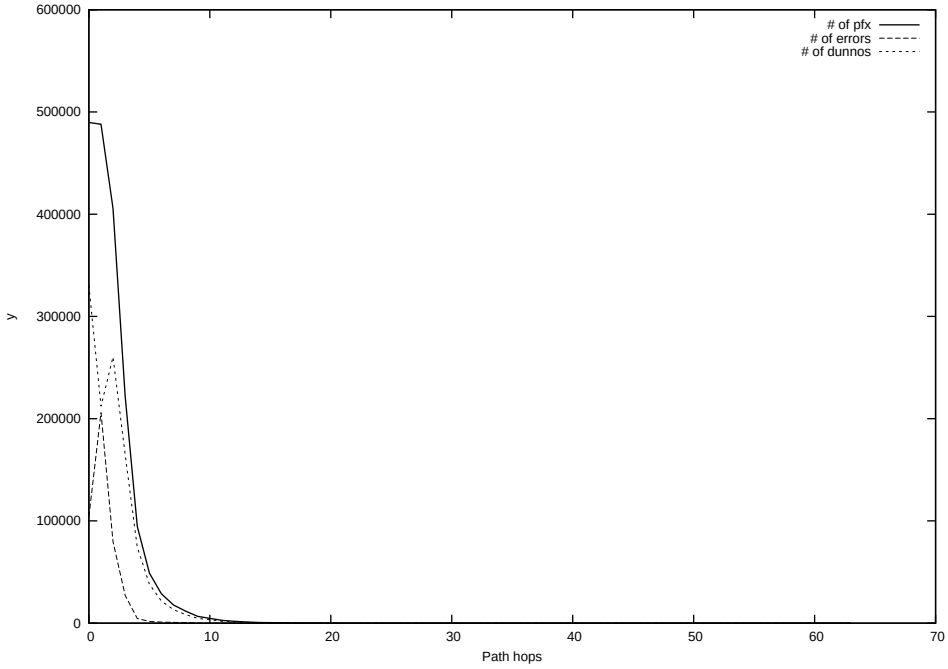
2014-04-06



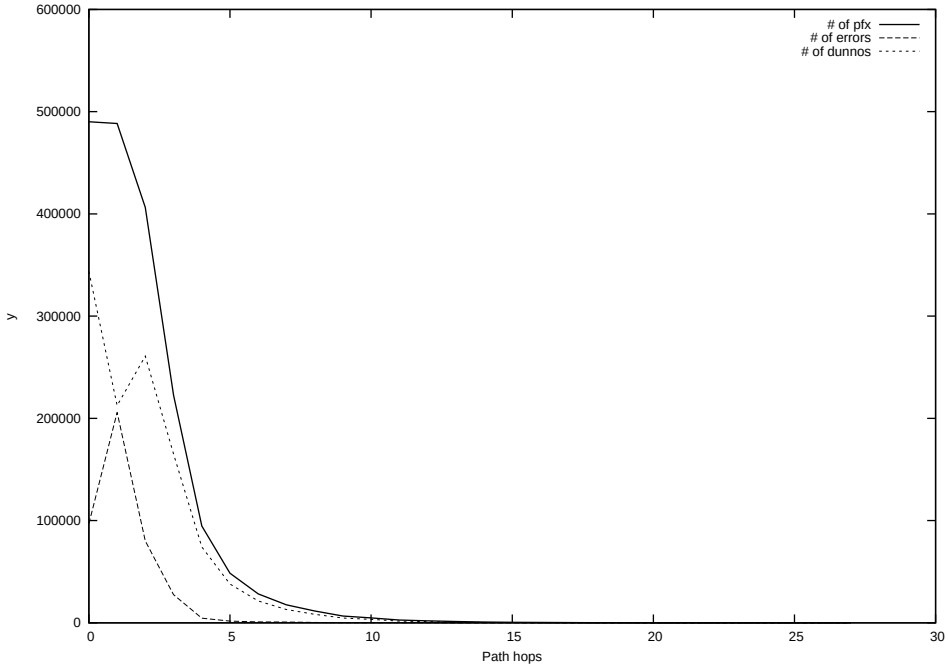
2014-04-07



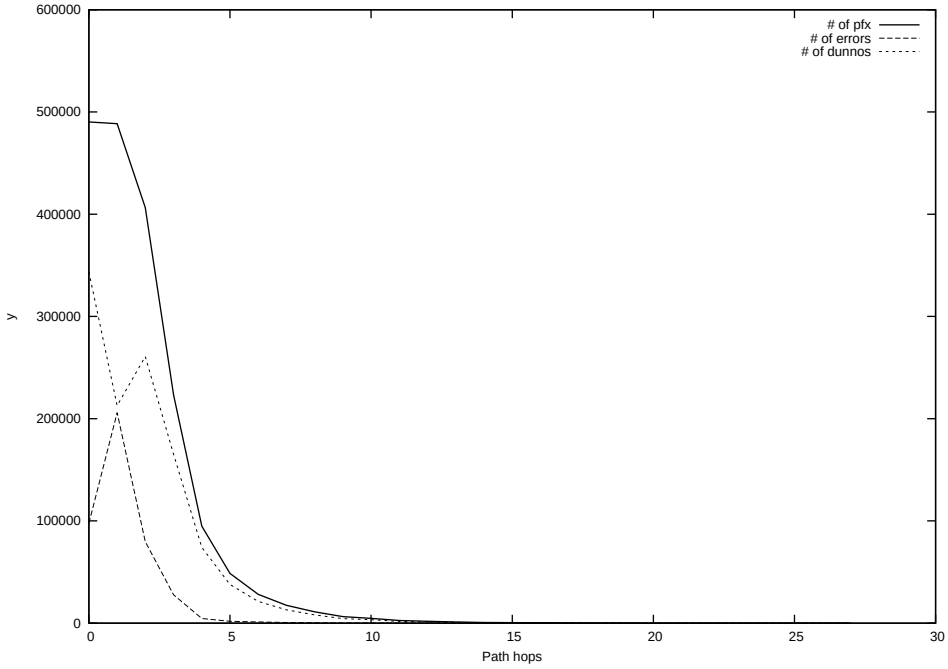
2014-04-08



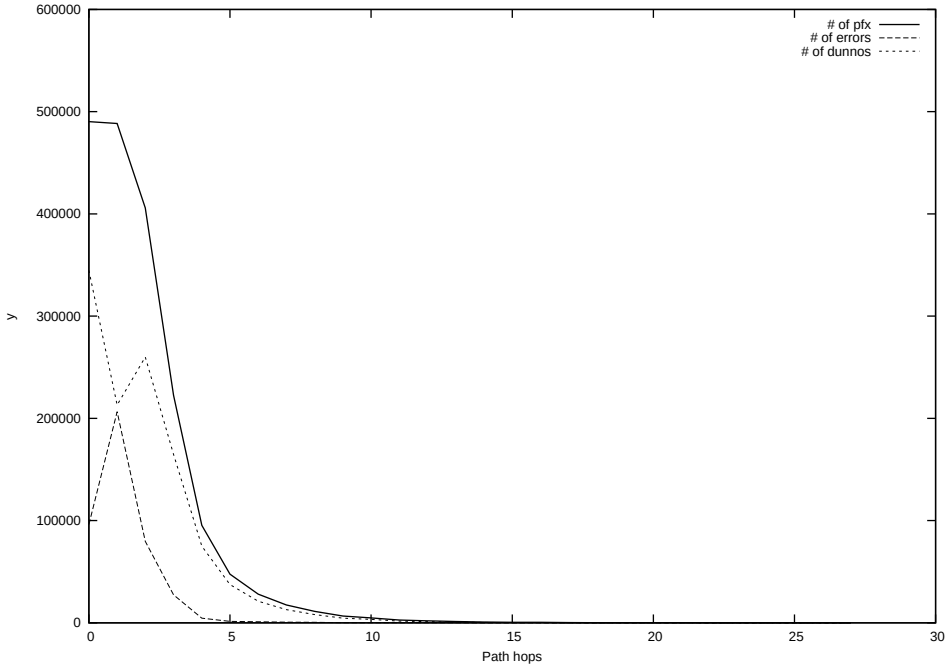
2014-04-09



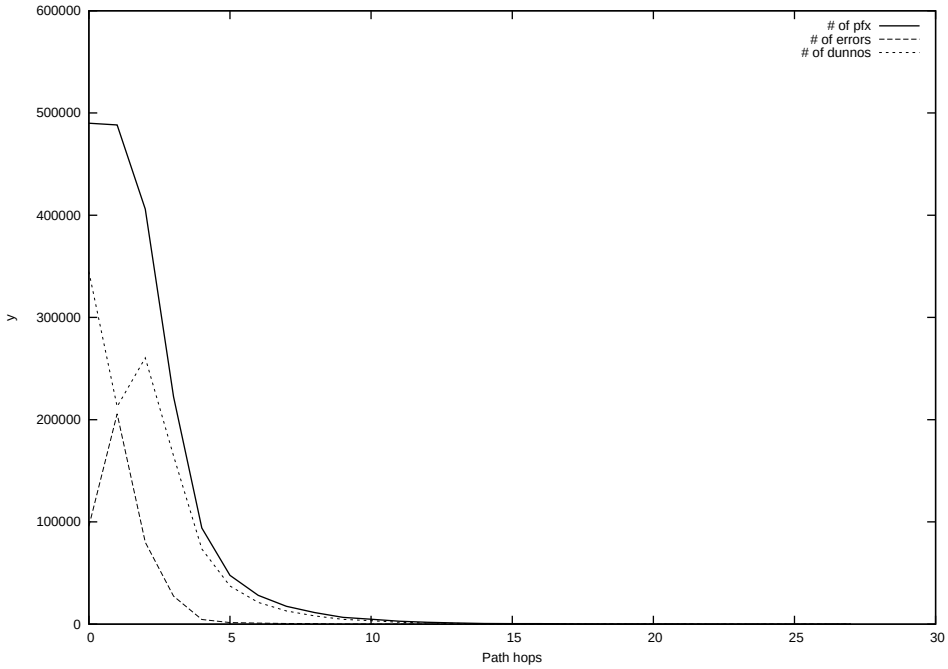
2014-04-10



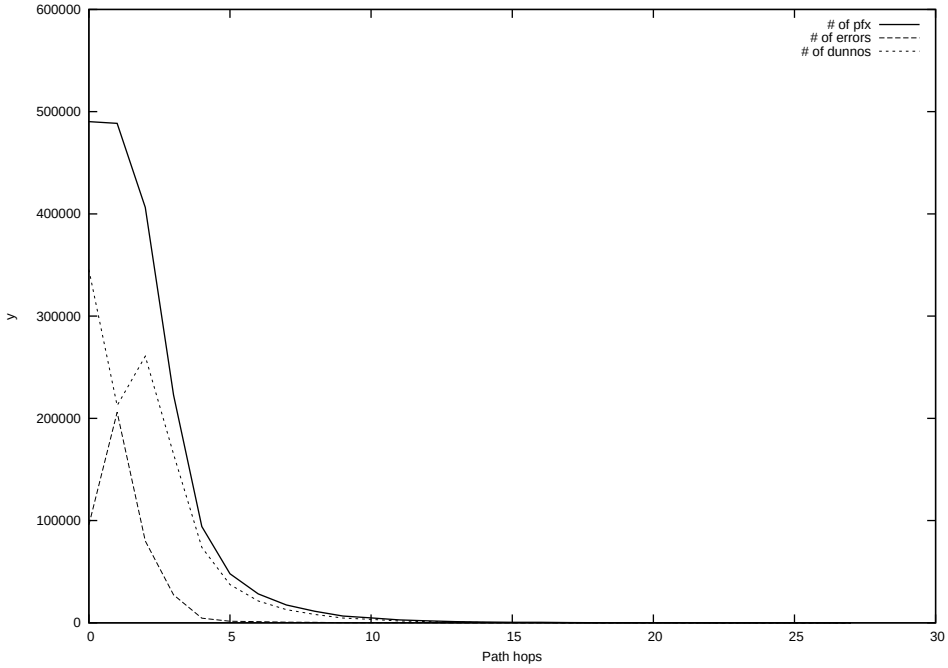
2014-04-11



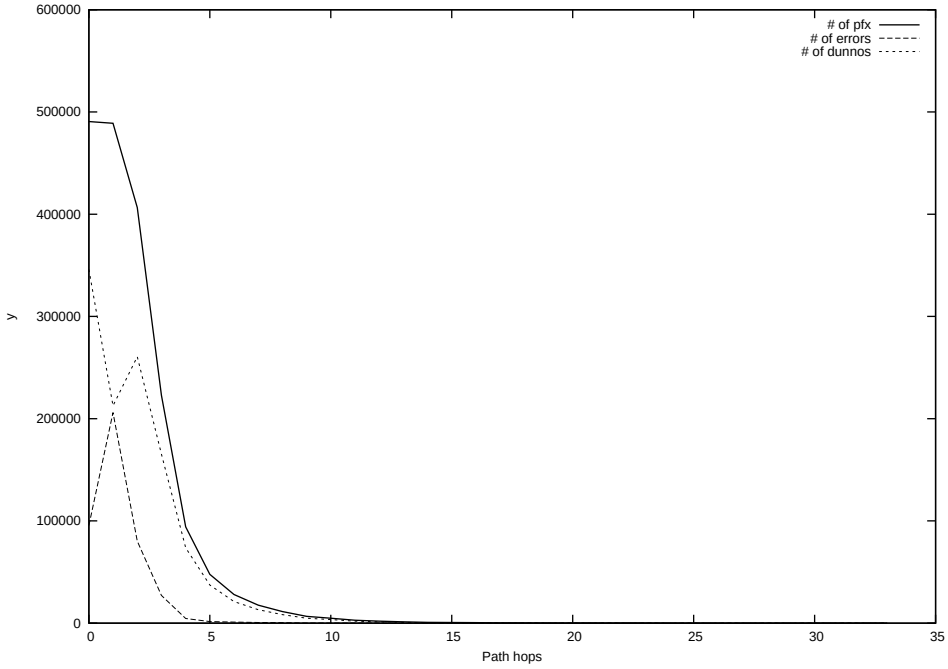
2014-04-12



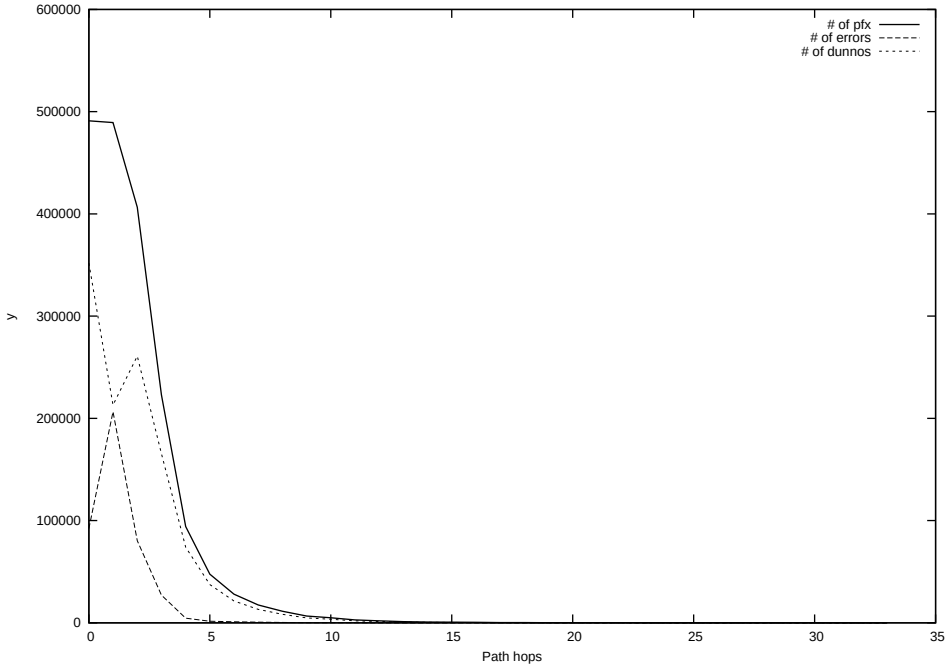
2014-04-13



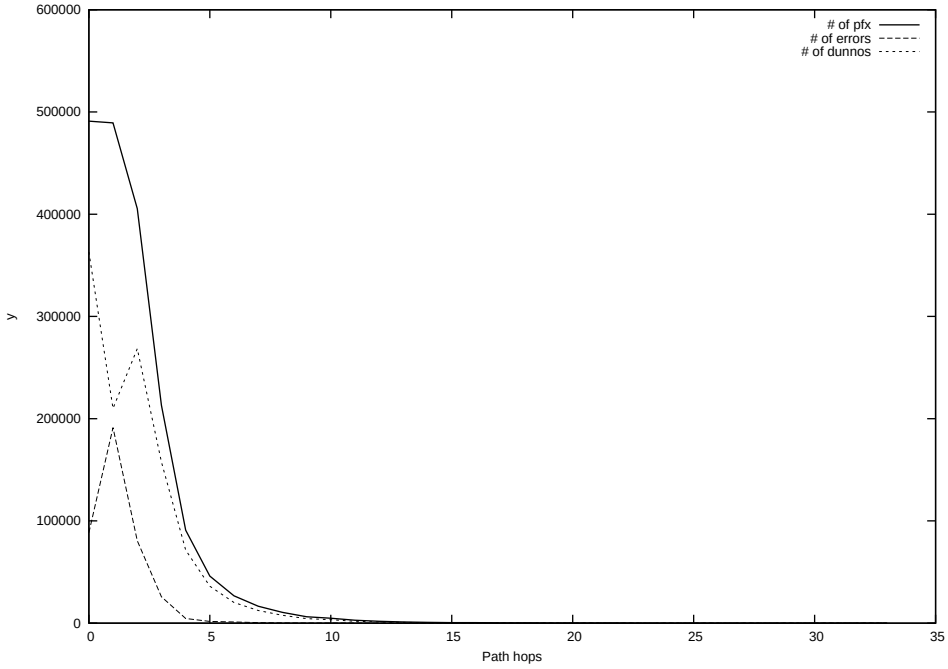
2014-04-14



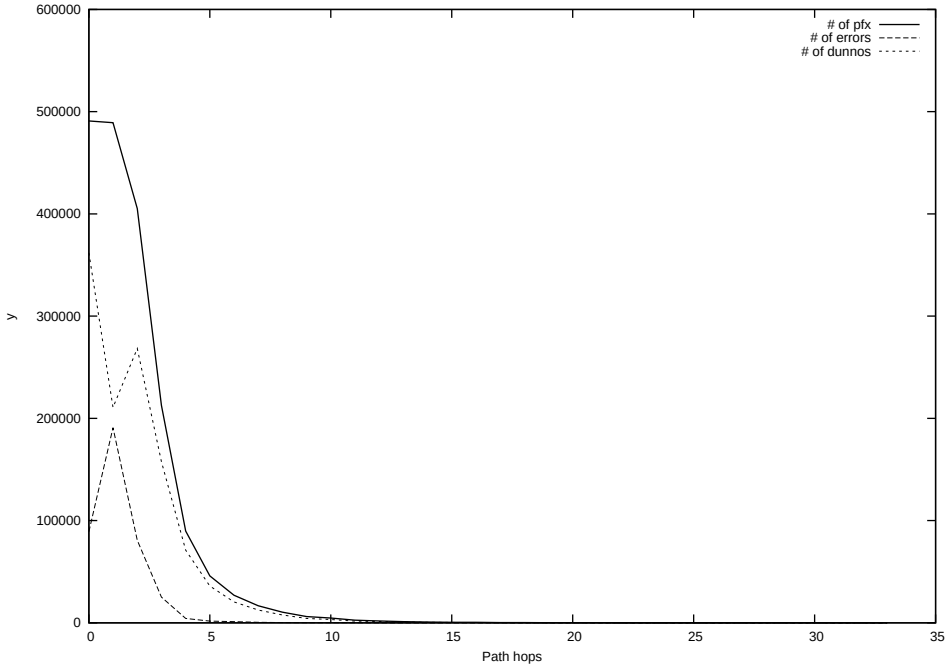
2014-04-15



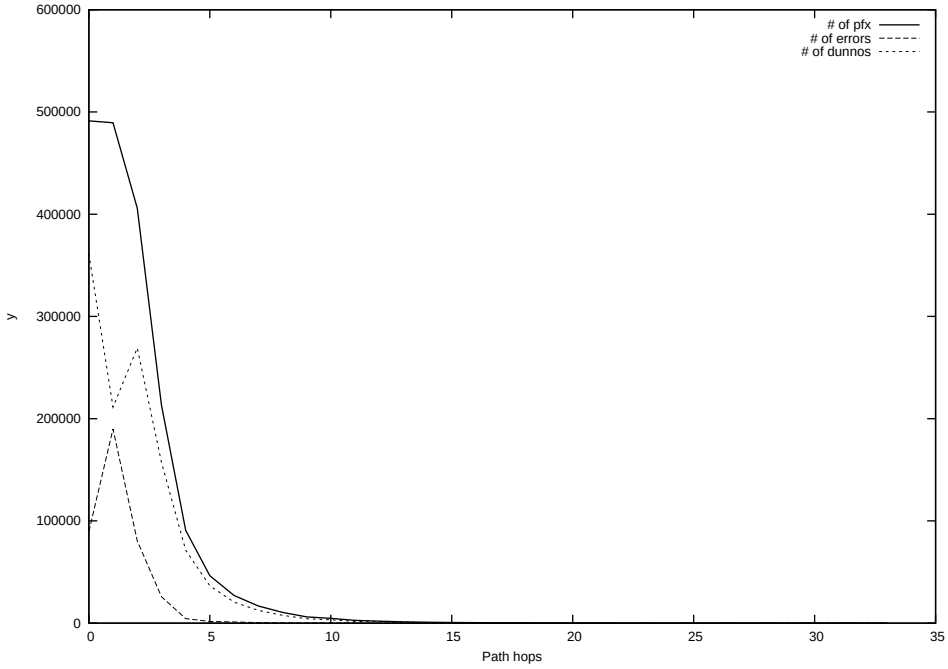
2014-04-16



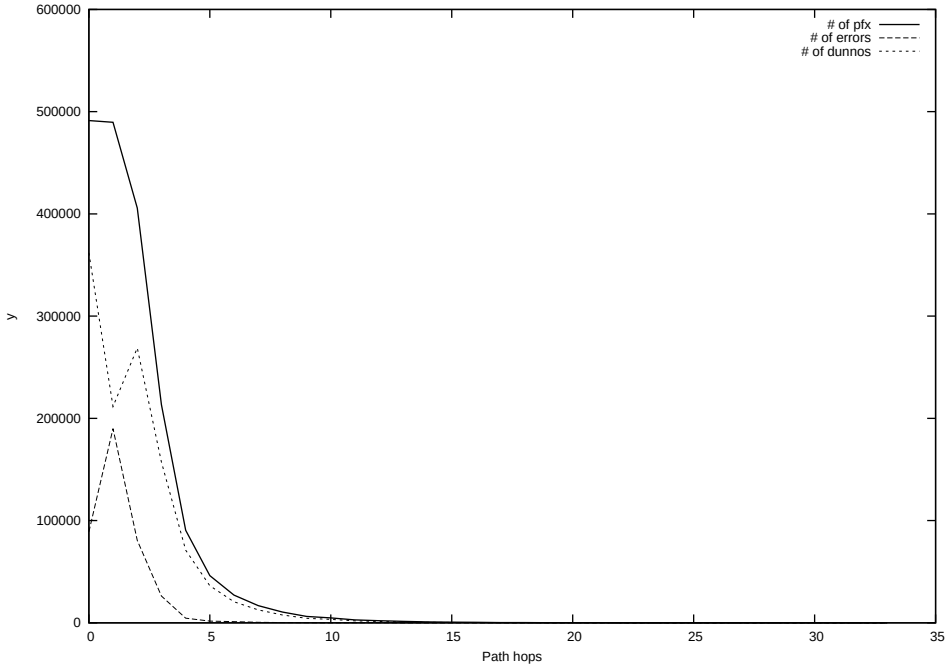
2014-04-17



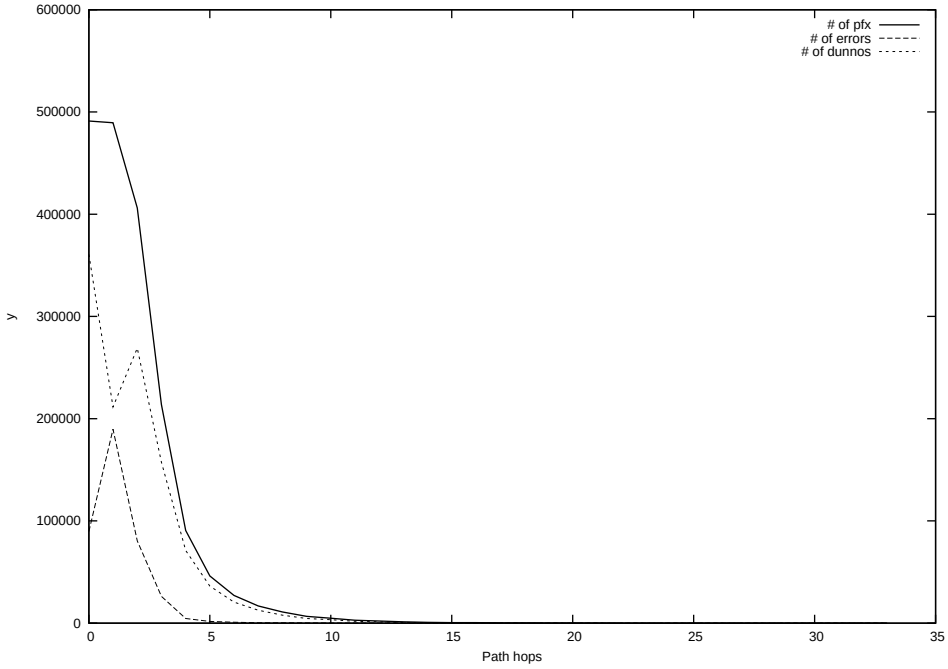
2014-04-18



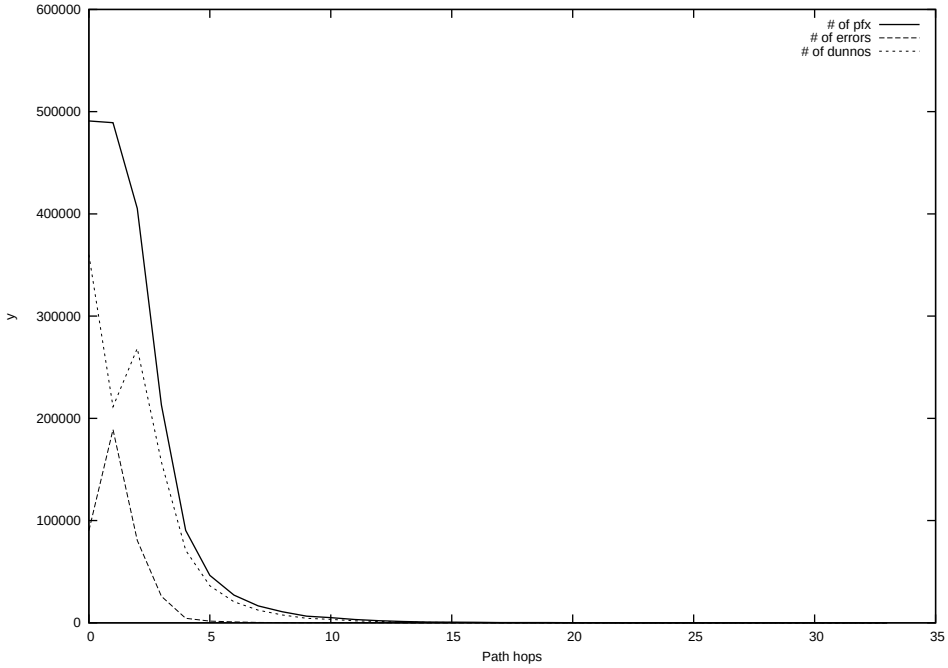
2014-04-19



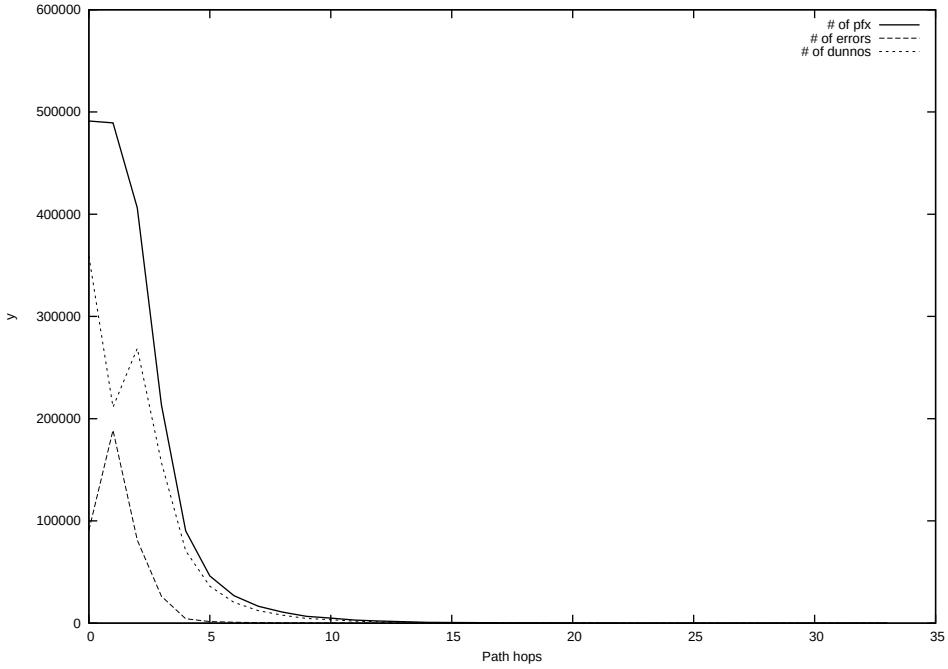
2014-04-20



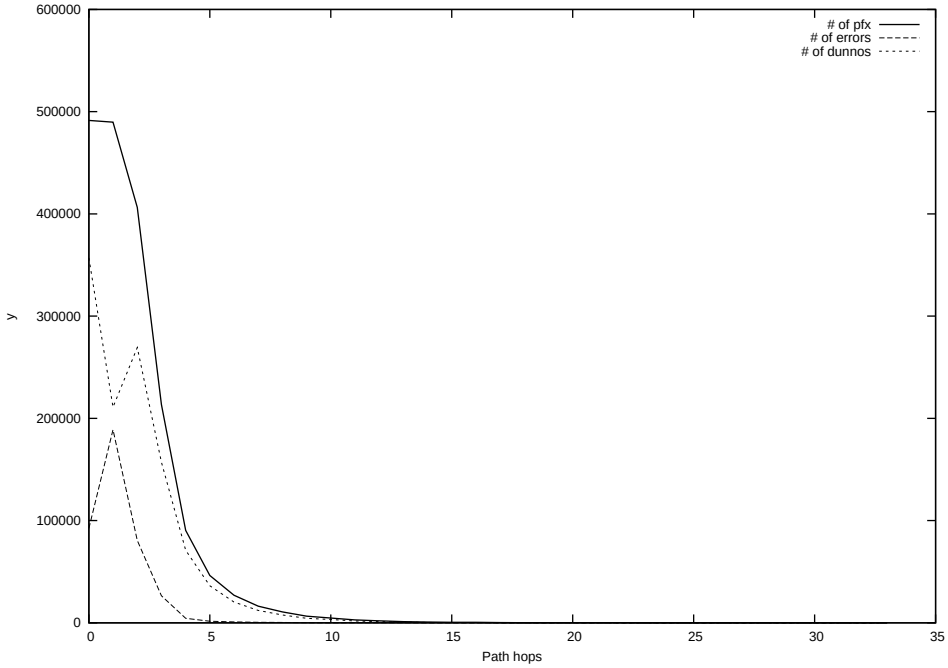
2014-04-21



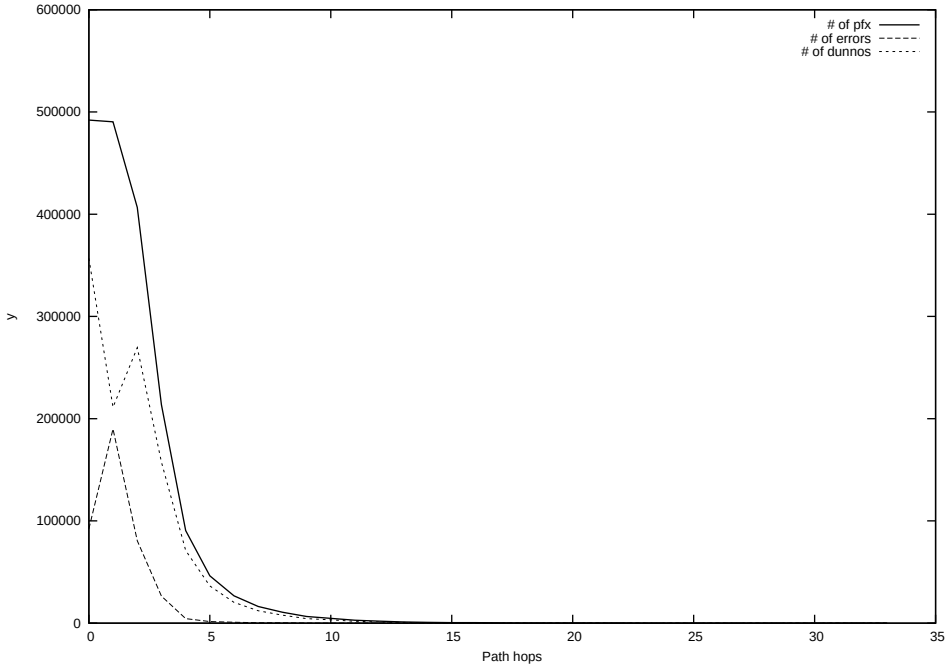
2014-04-22



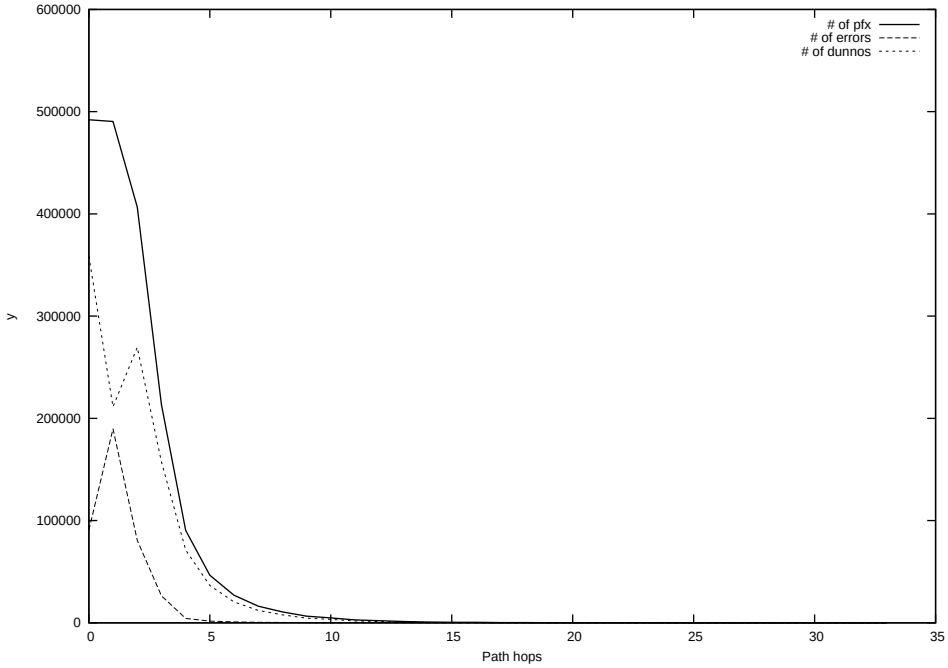
2014-04-23



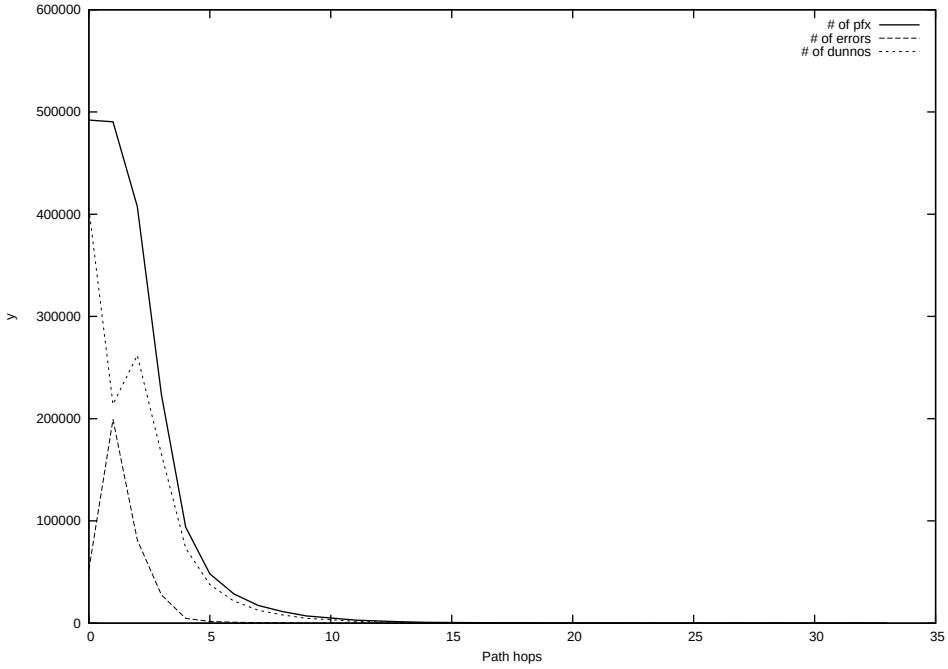
2014-04-24



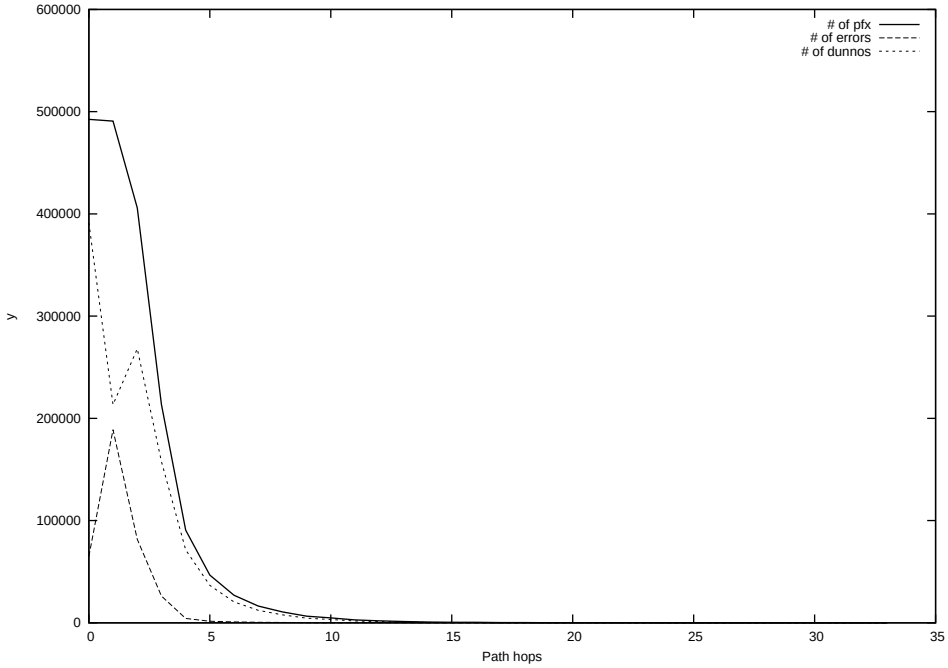
2014-04-25



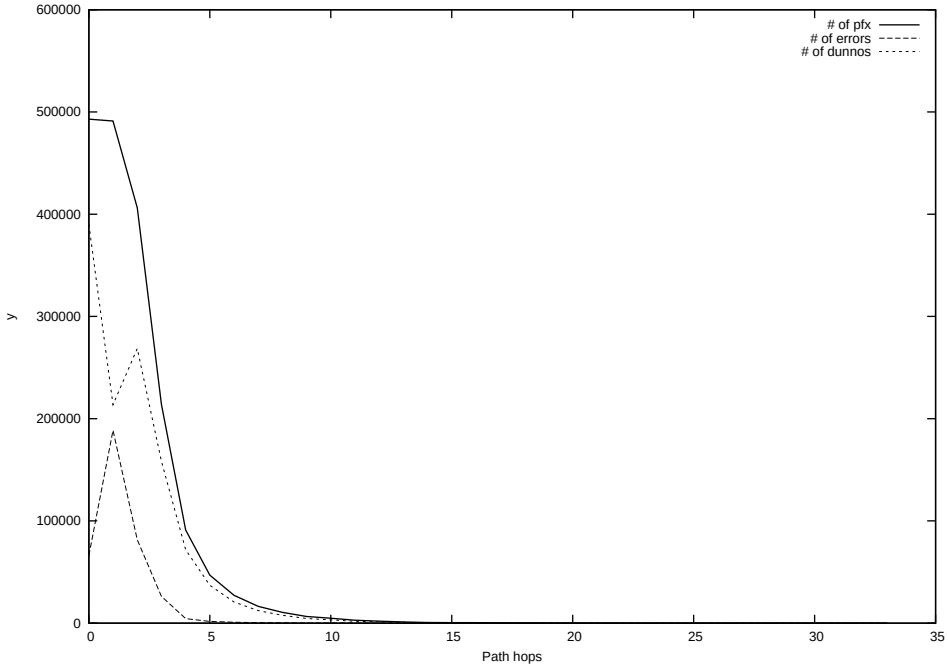
2014-04-26



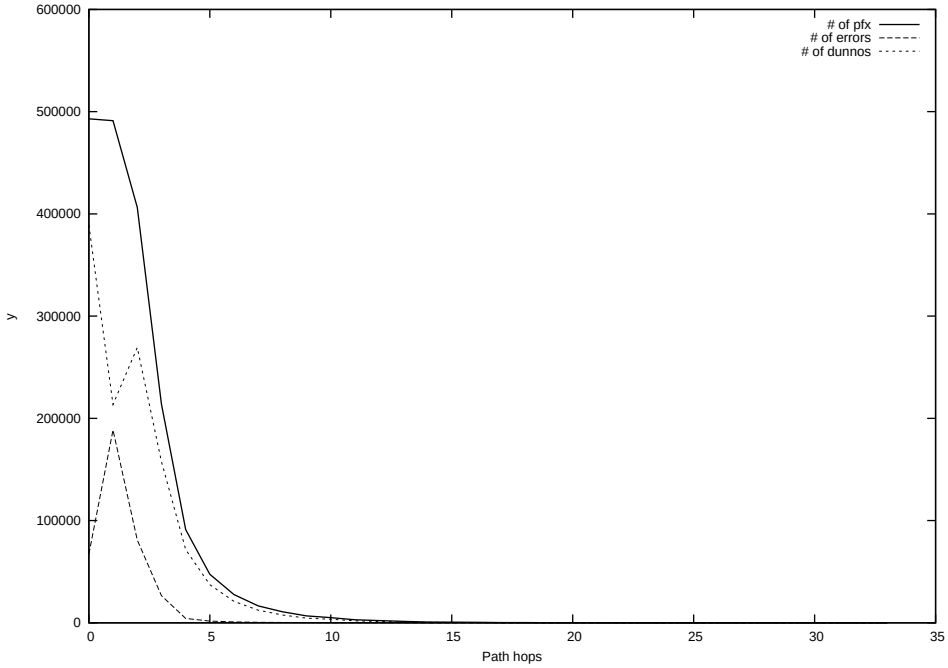
2014-04-27



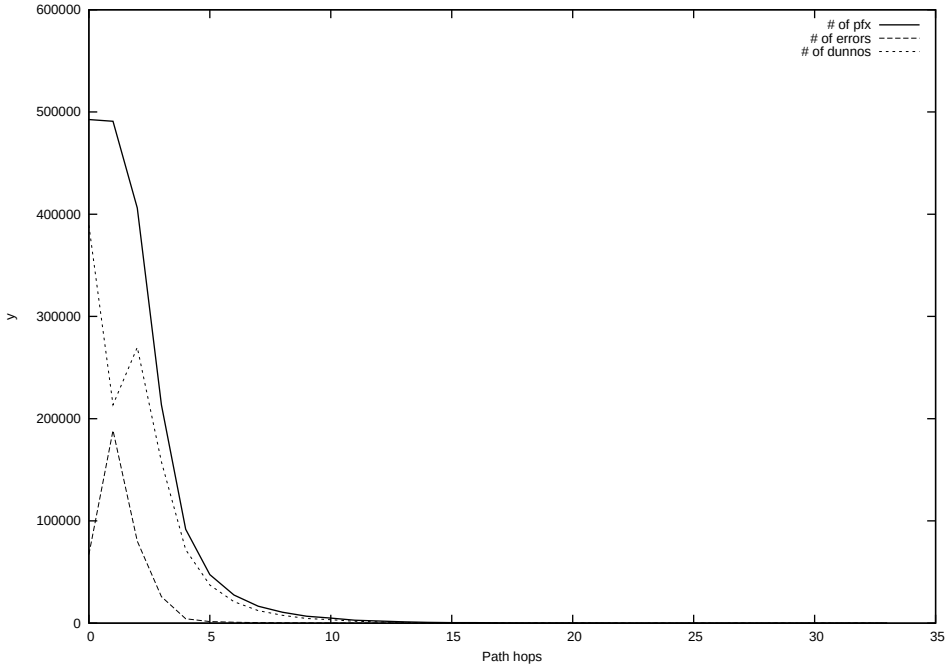
2014-04-28



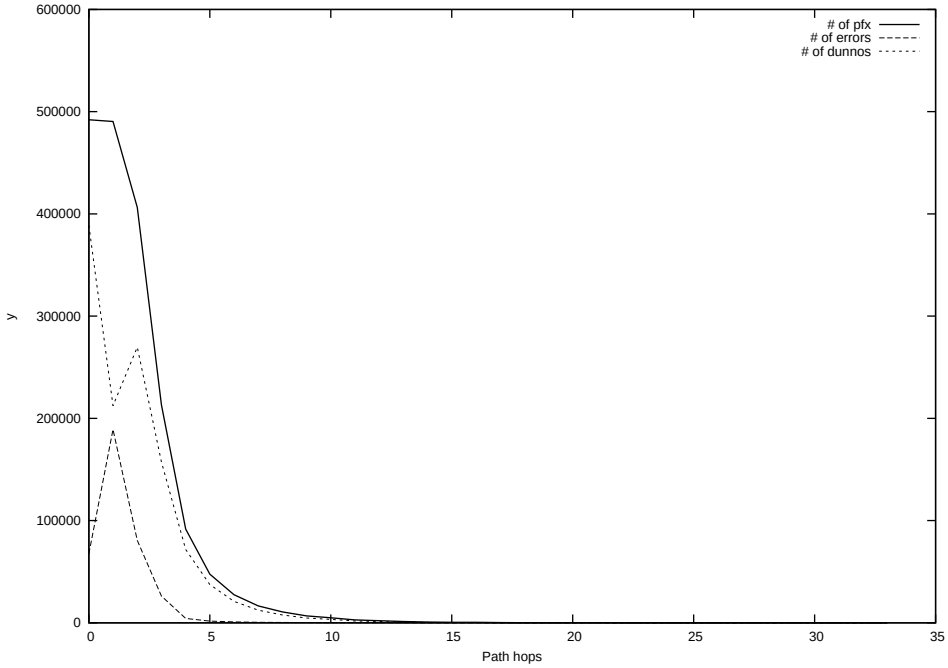
2014-04-29



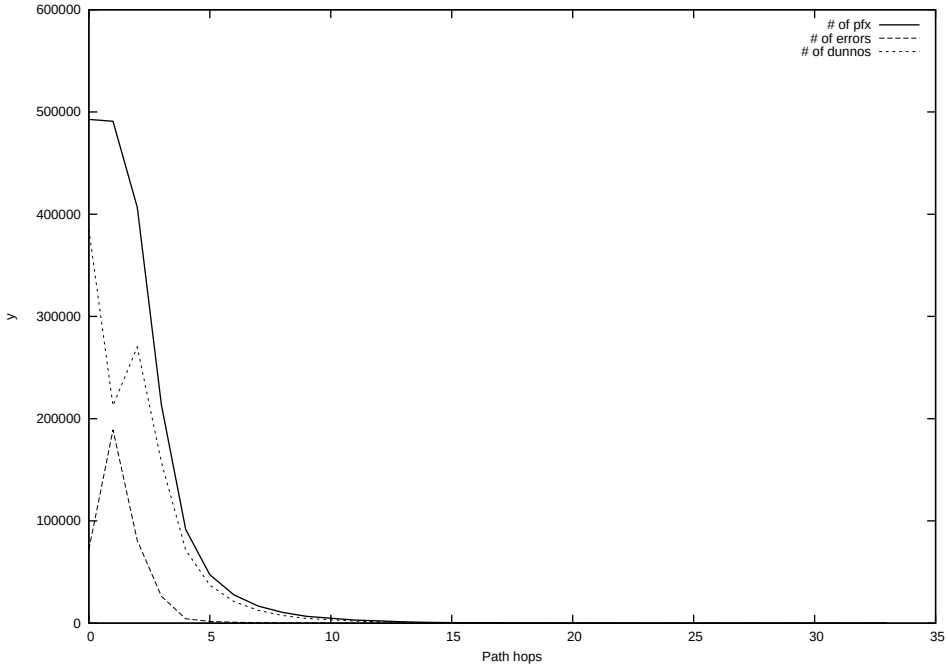
2014-04-30



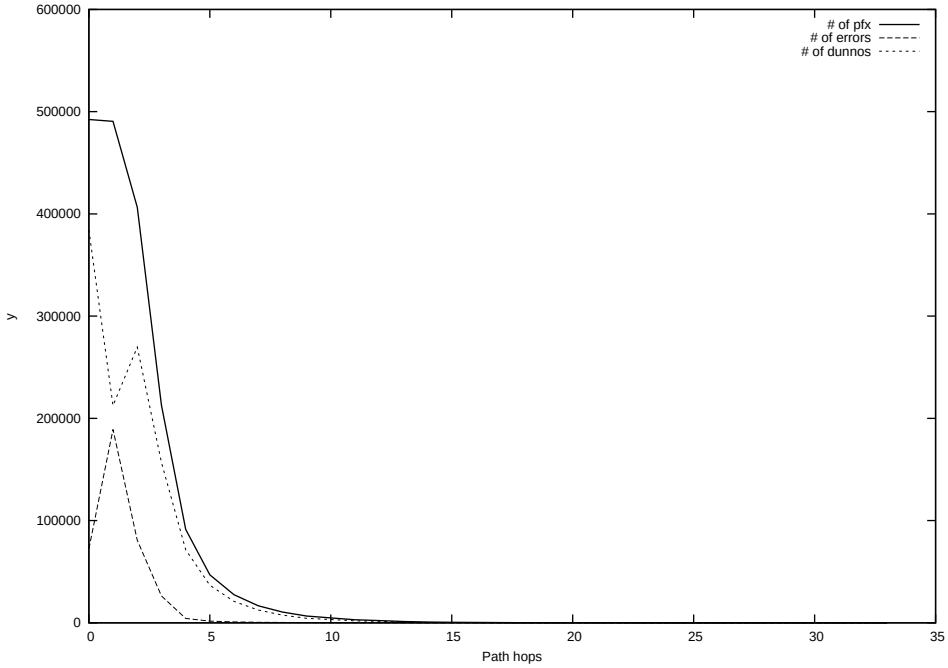
2014-05-01



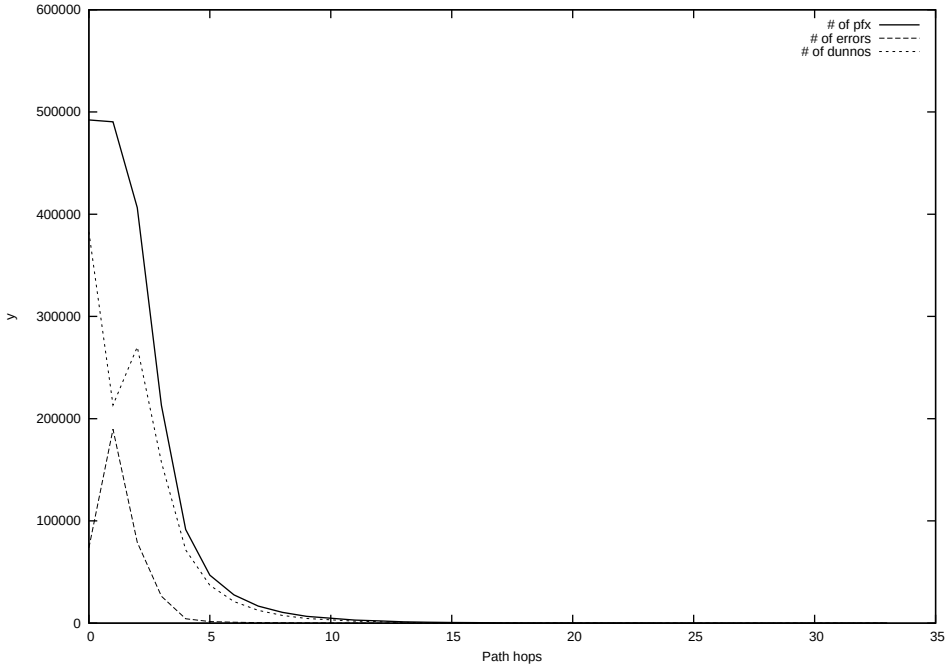
2014-05-02



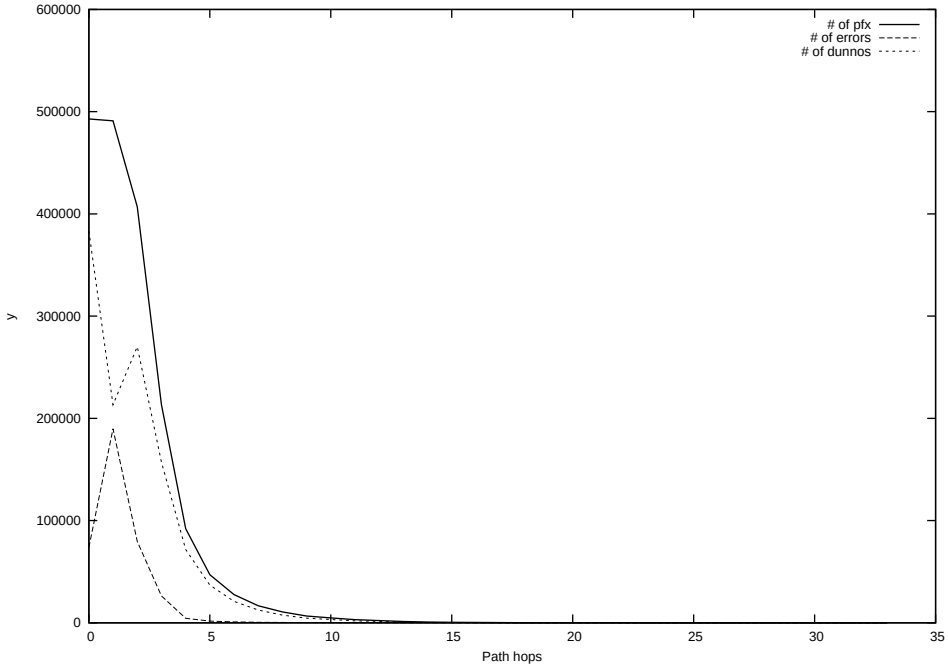
2014-05-03



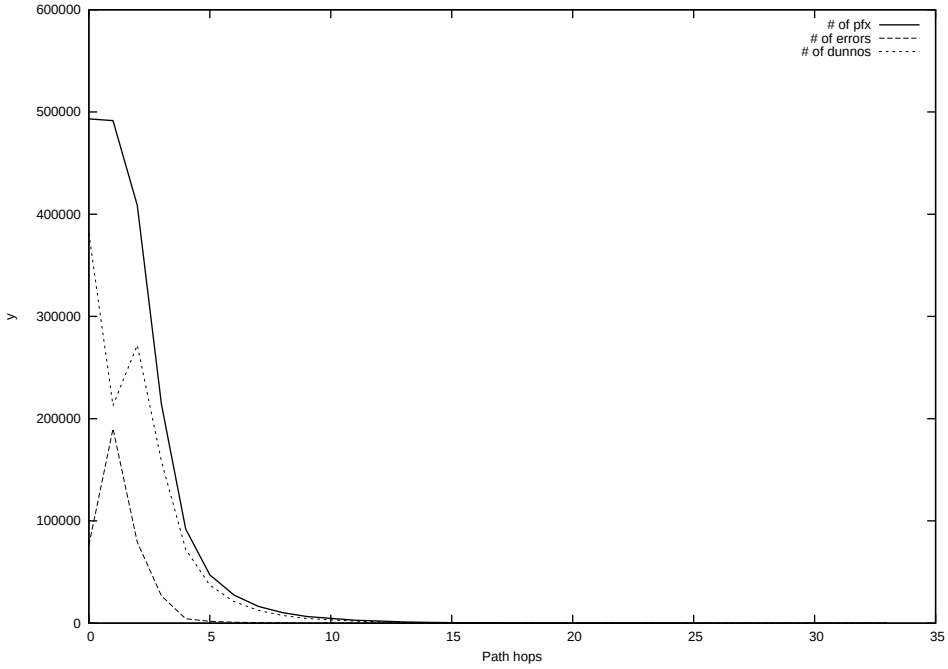
2014-05-04



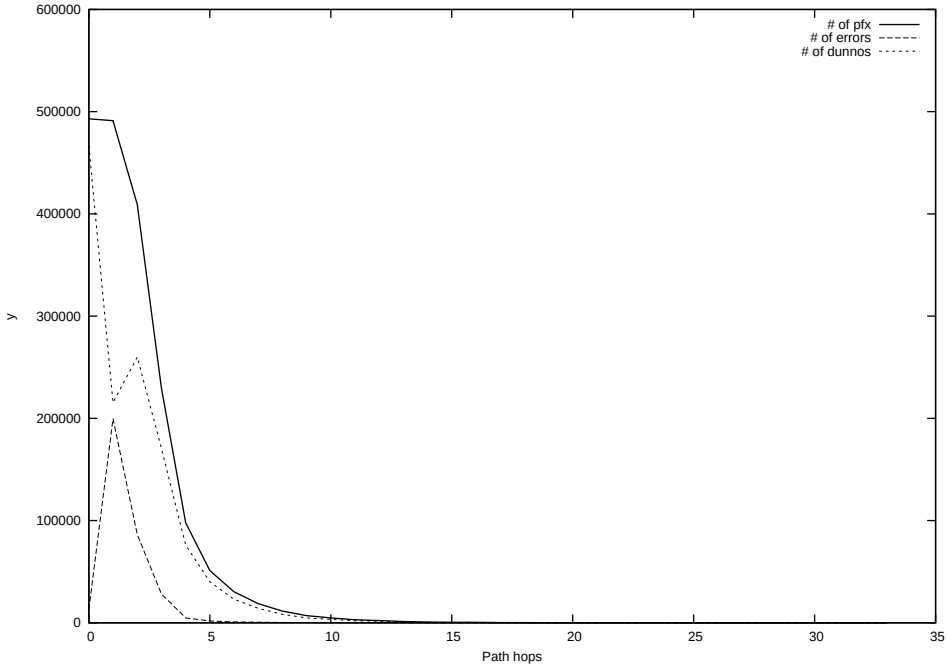
2014-05-05



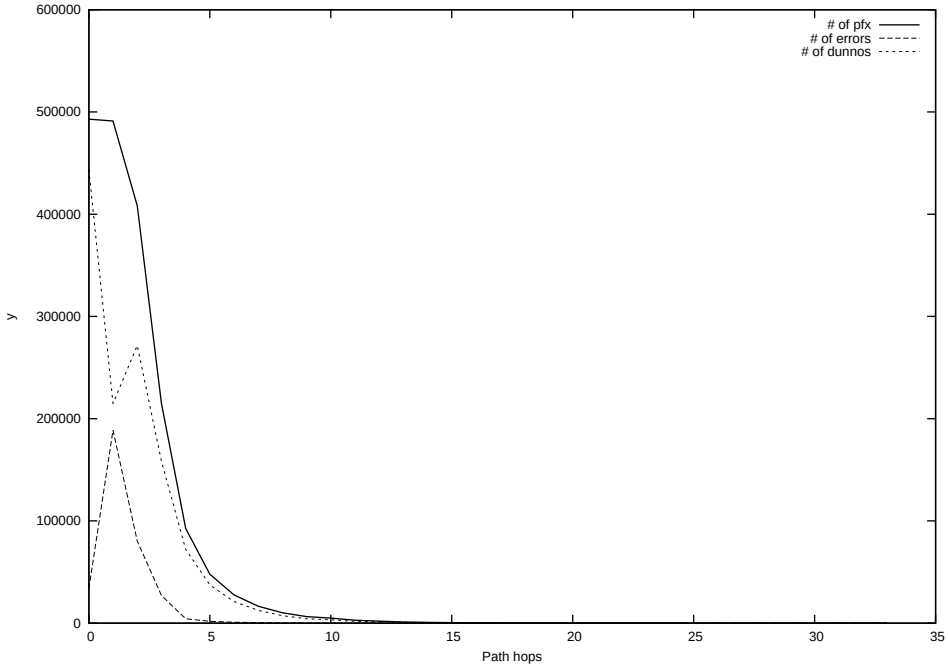
2014-05-06



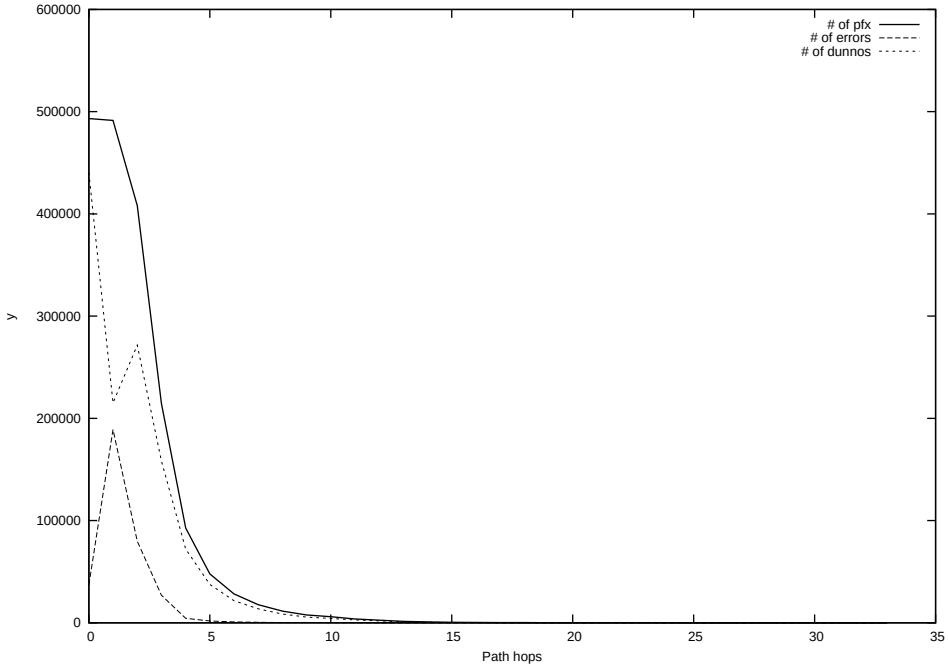
2014-05-07



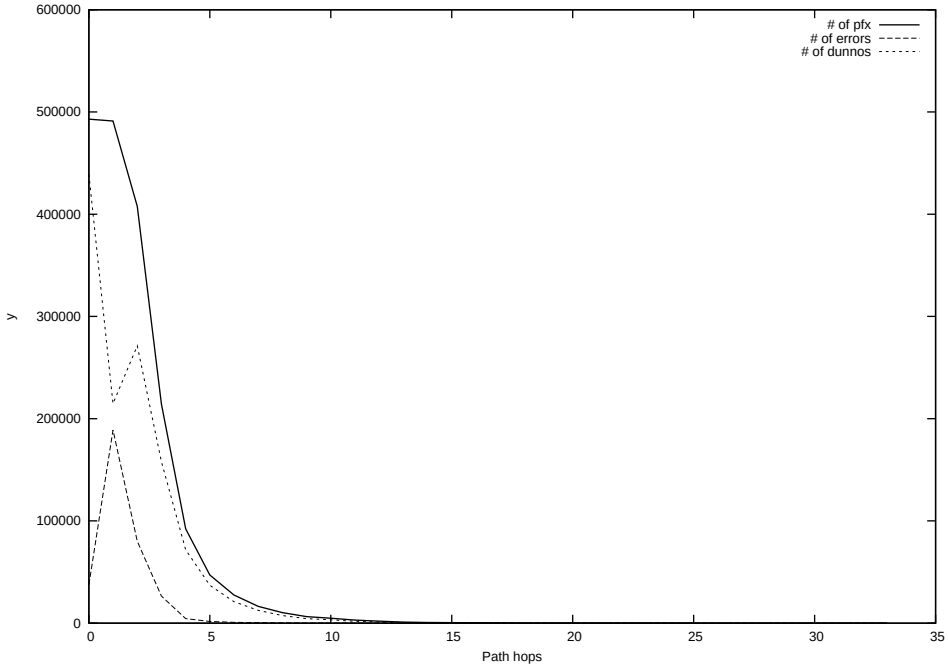
2014-05-08



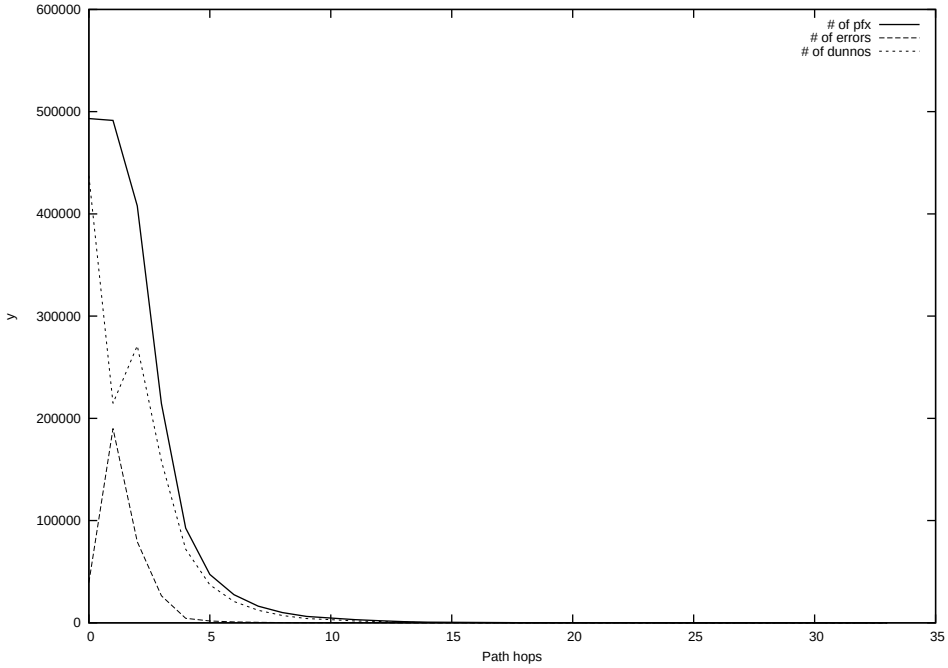
2014-05-09



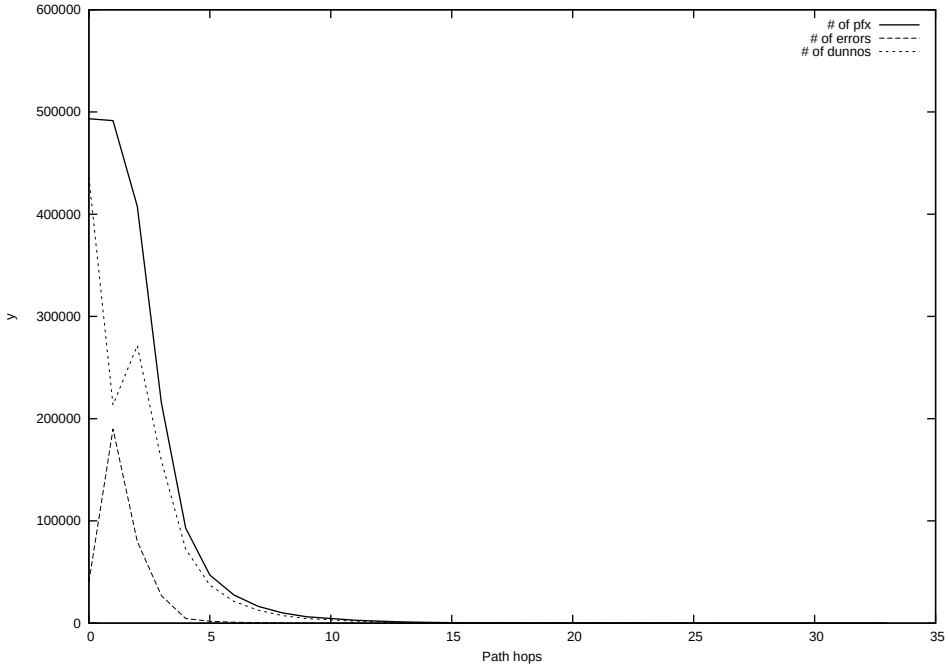
2014-05-10



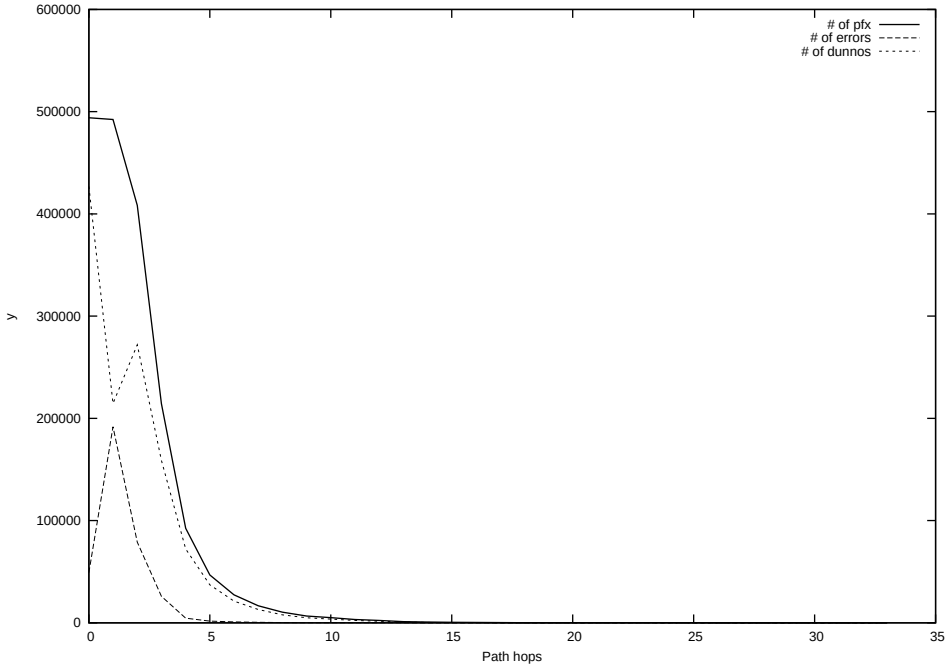
2014-05-11



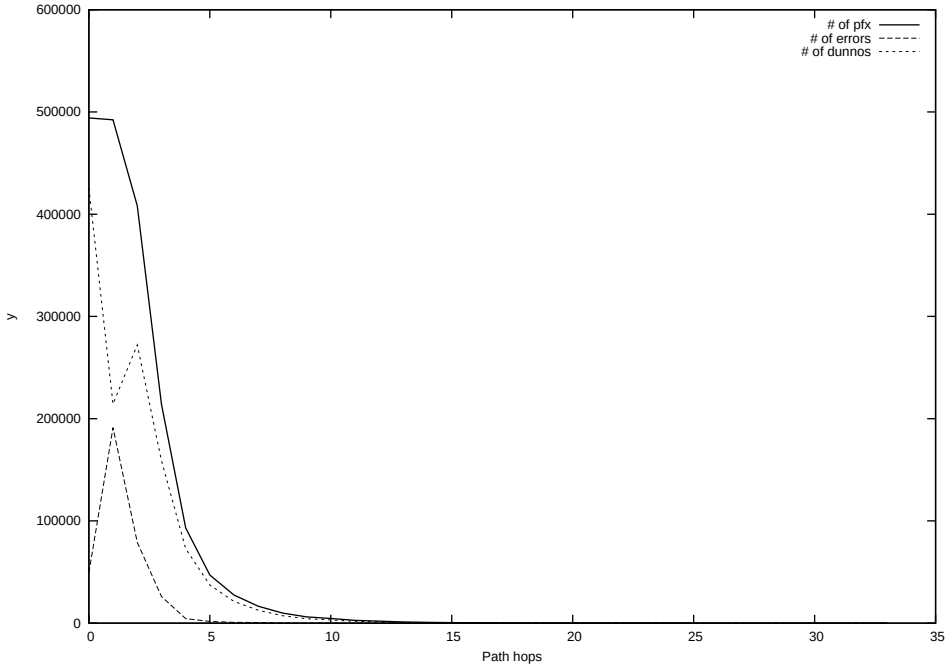
2014-05-12



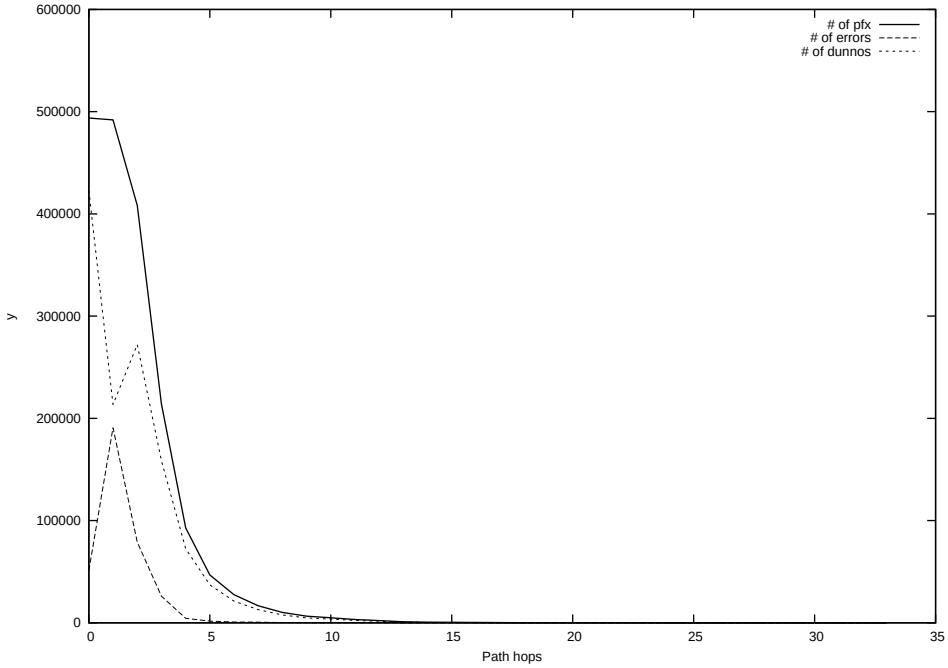
2014-05-13



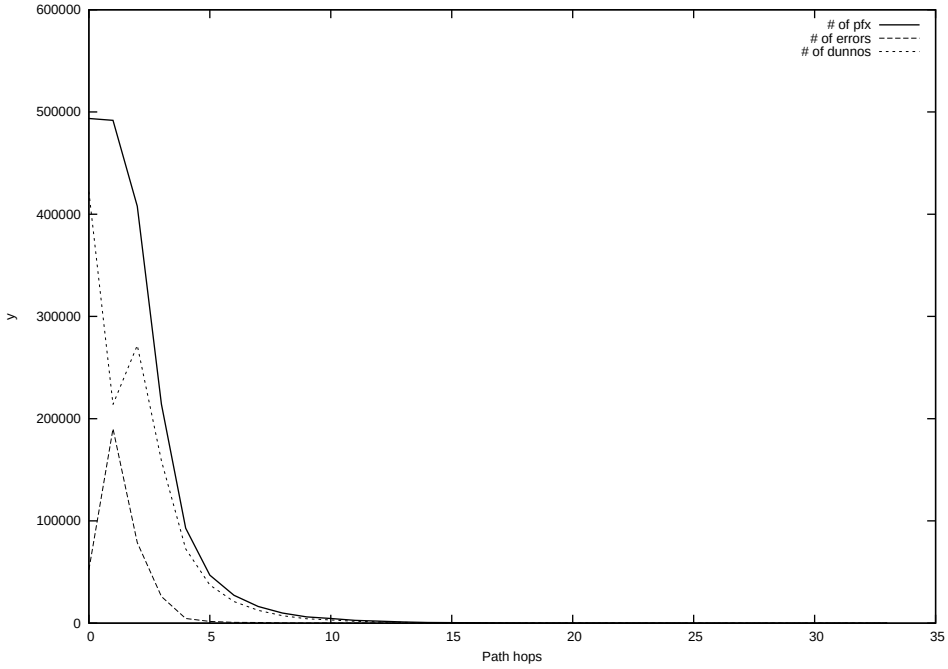
2014-05-14



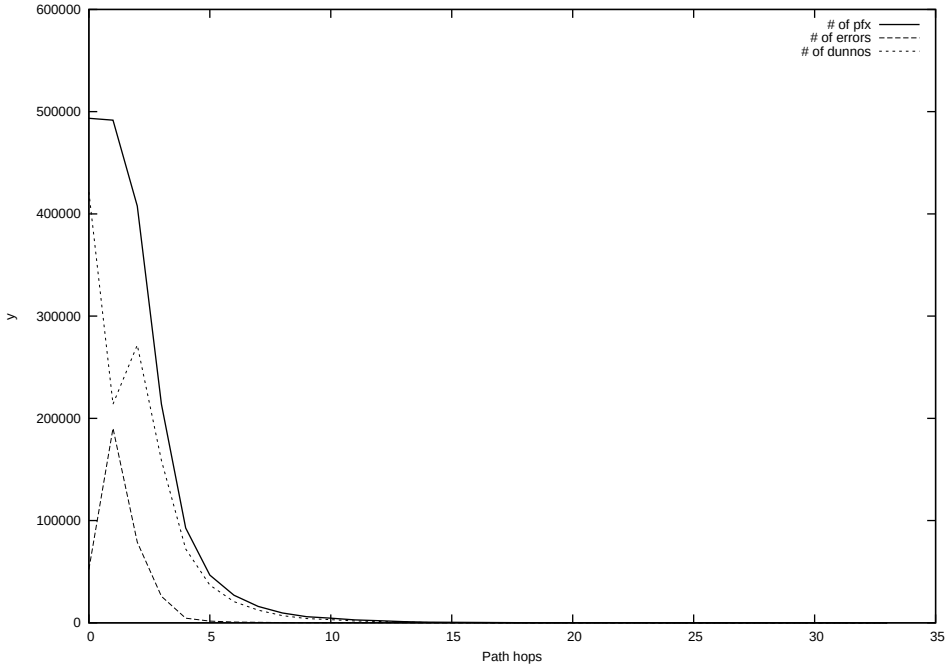
2014-05-15



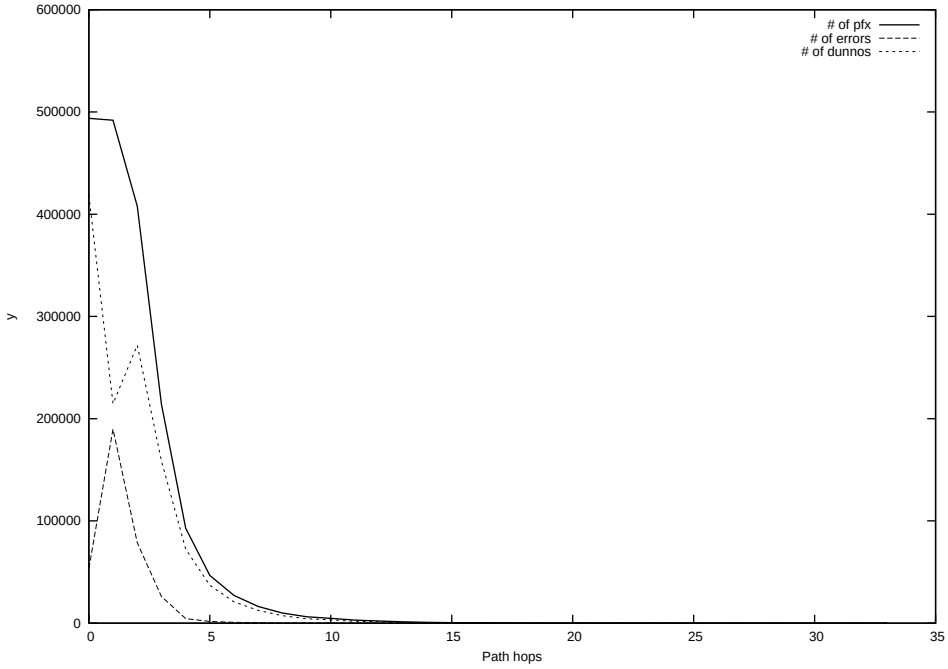
2014-05-16



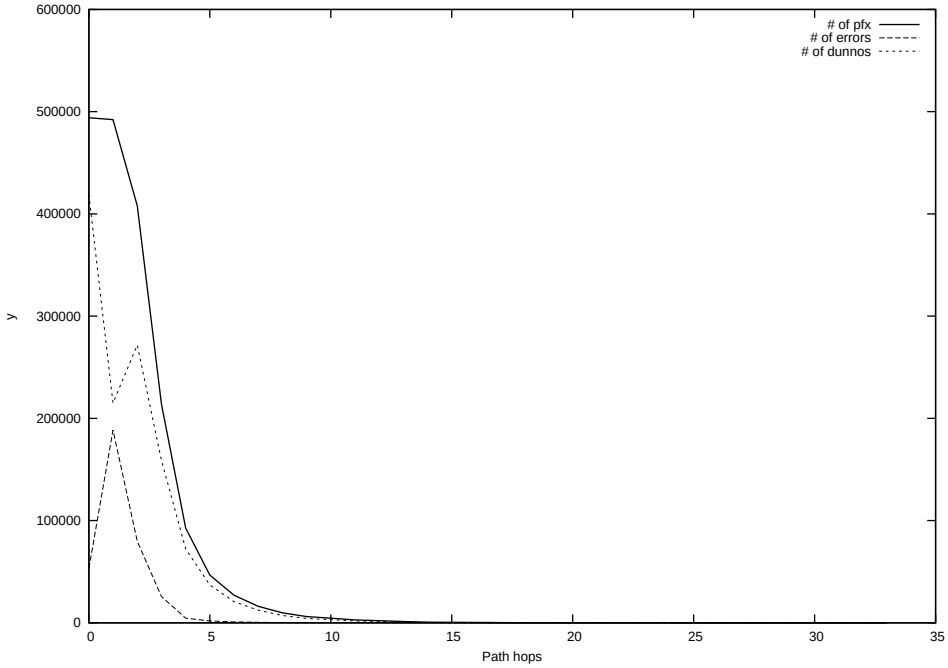
2014-05-17



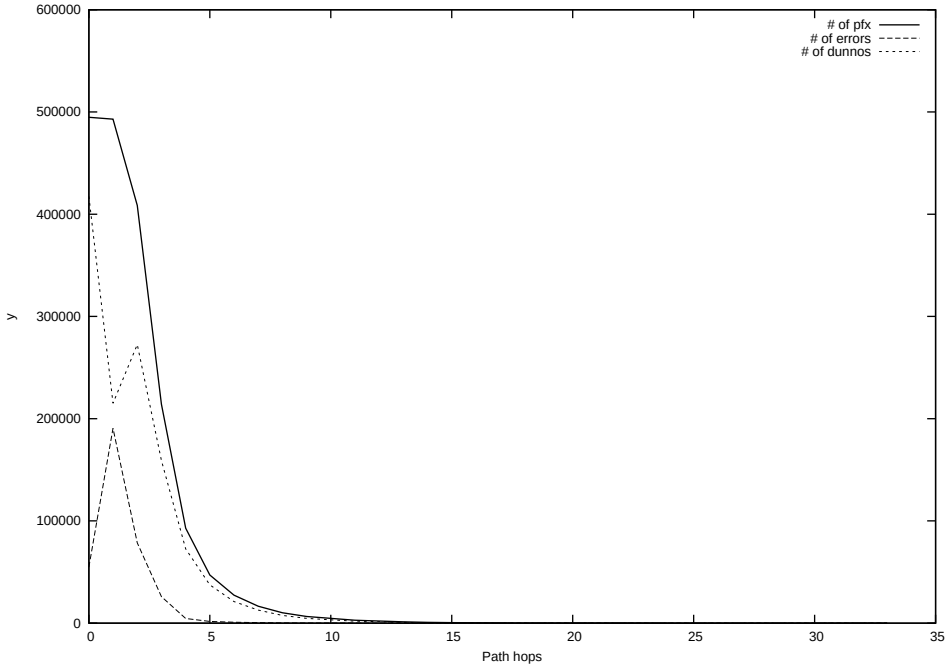
2014-05-18



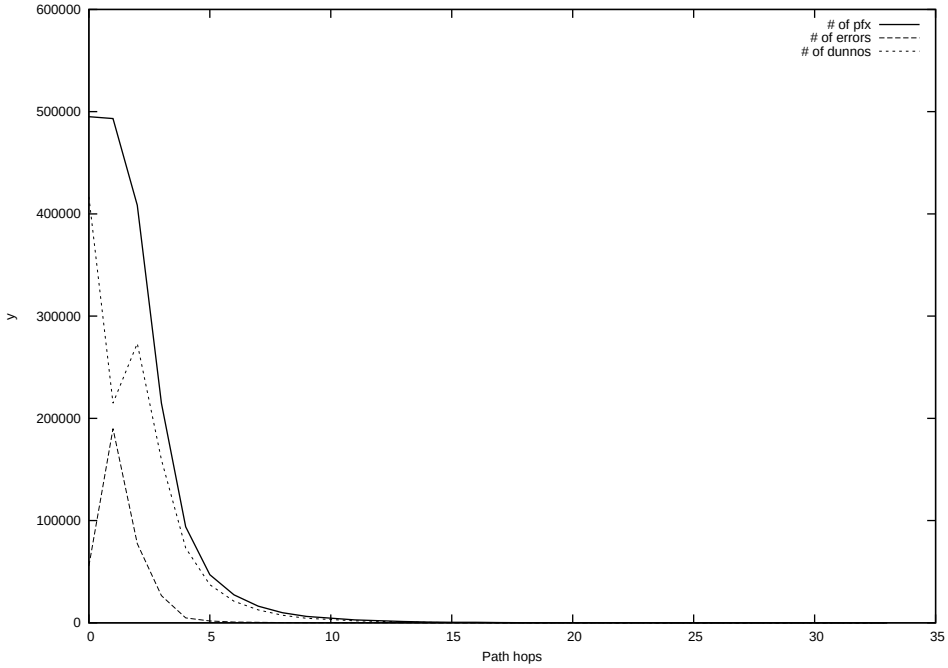
2014-05-19



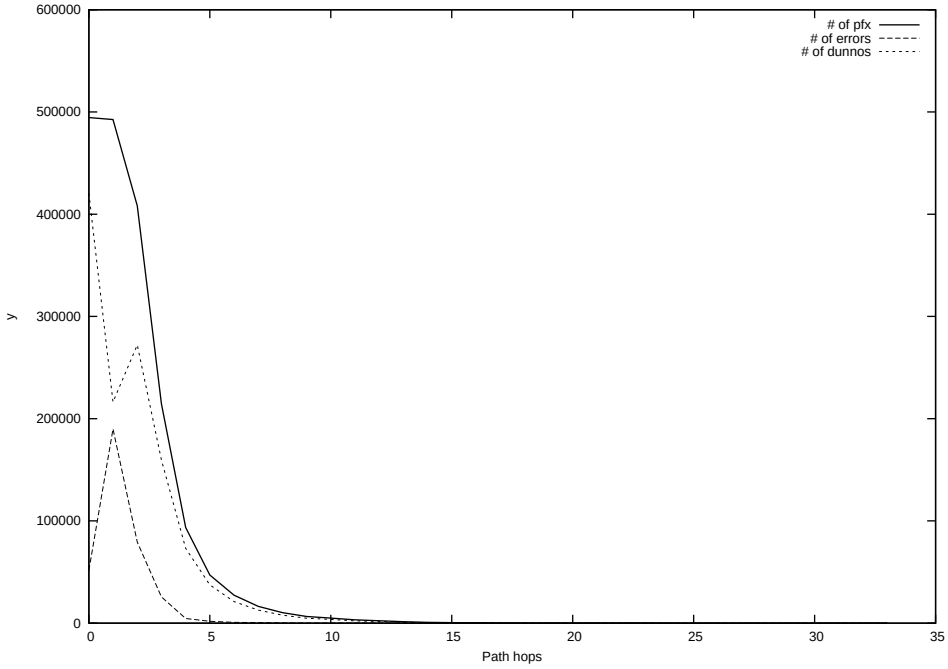
2014-05-20



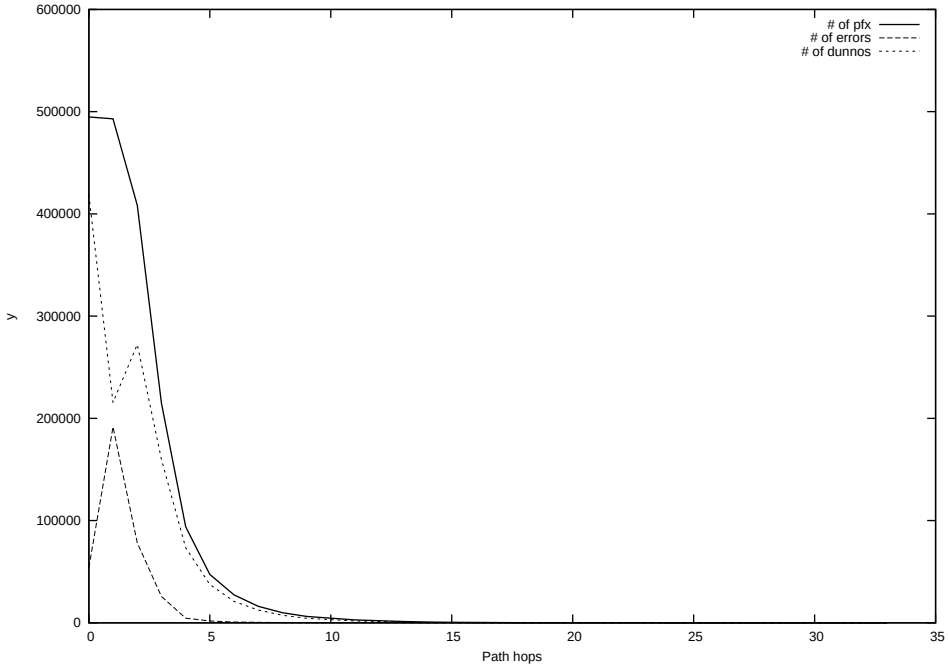
2014-05-21



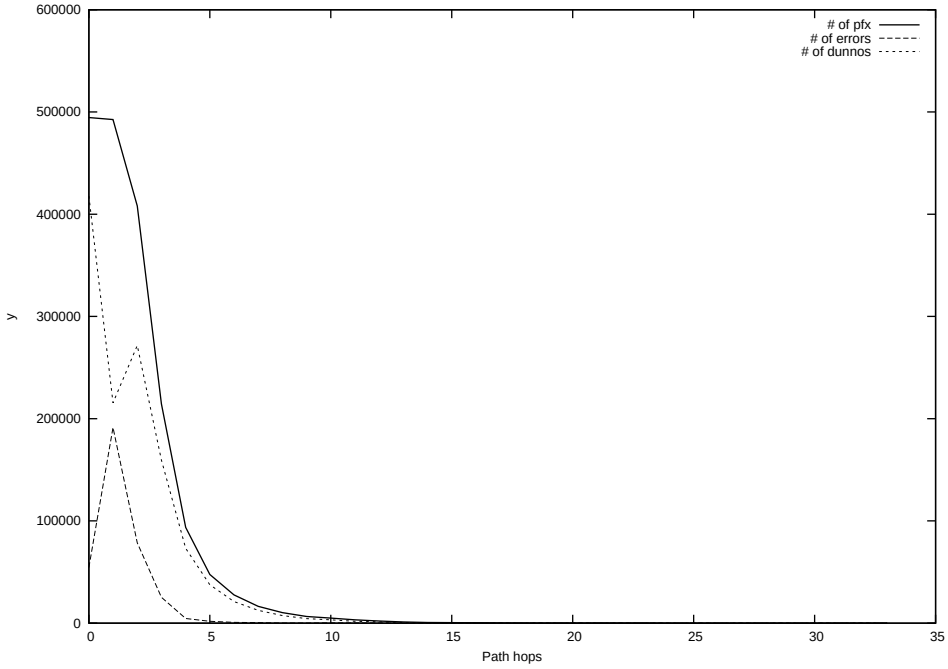
2014-05-22



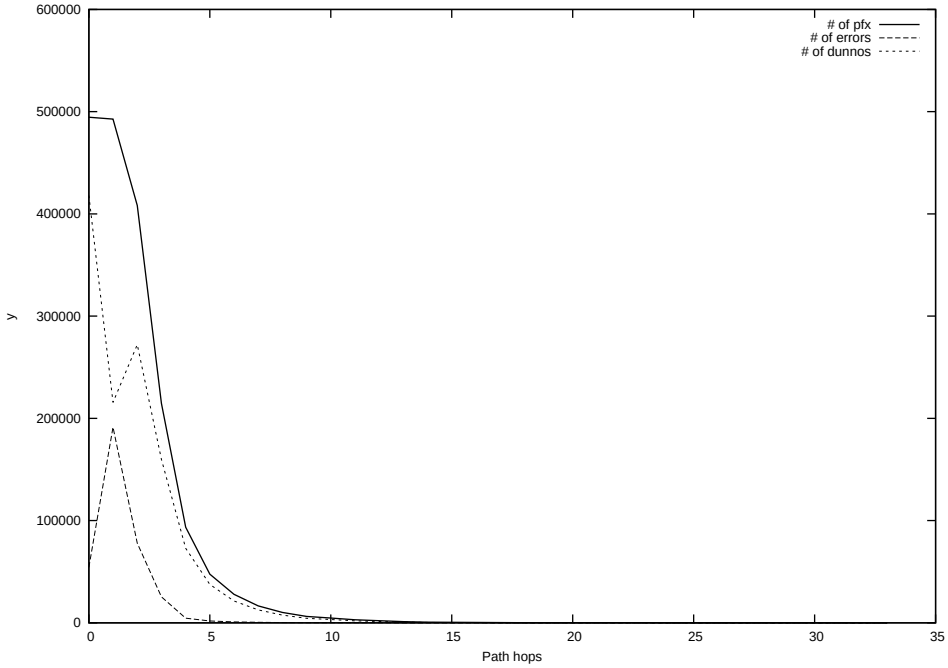
2014-05-23



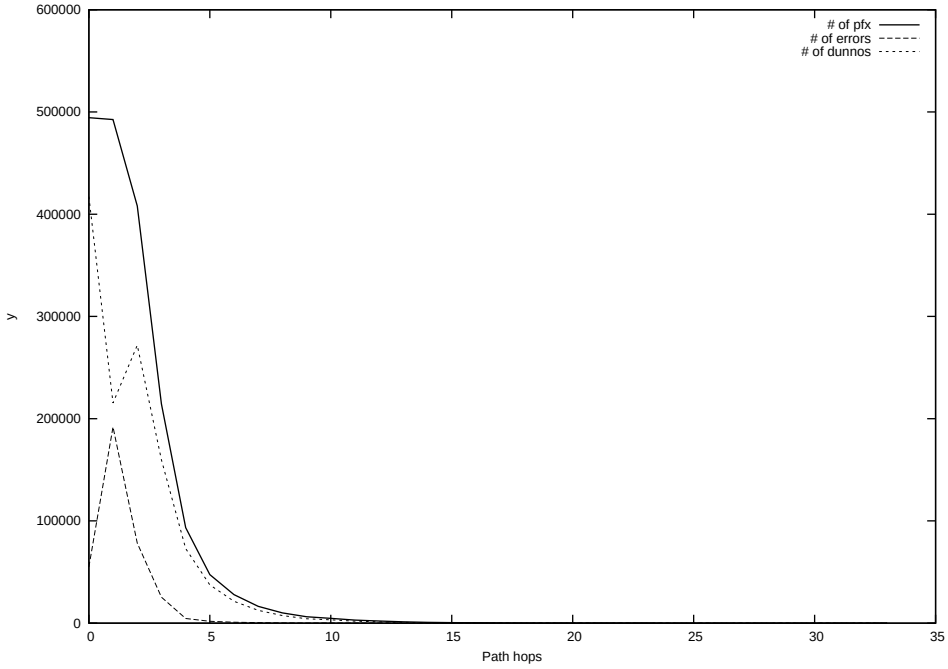
2014-05-24



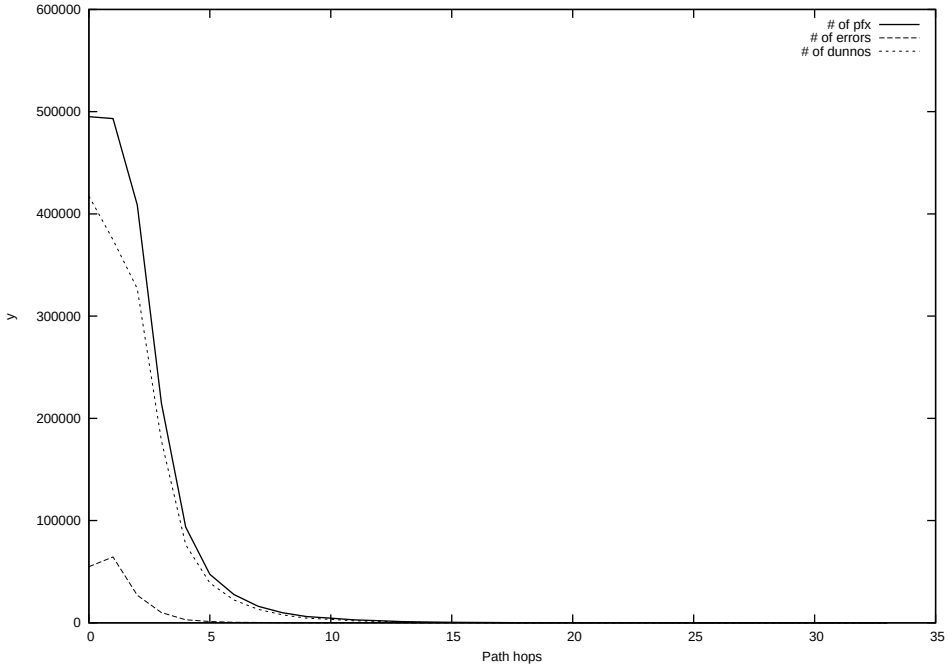
2014-05-25



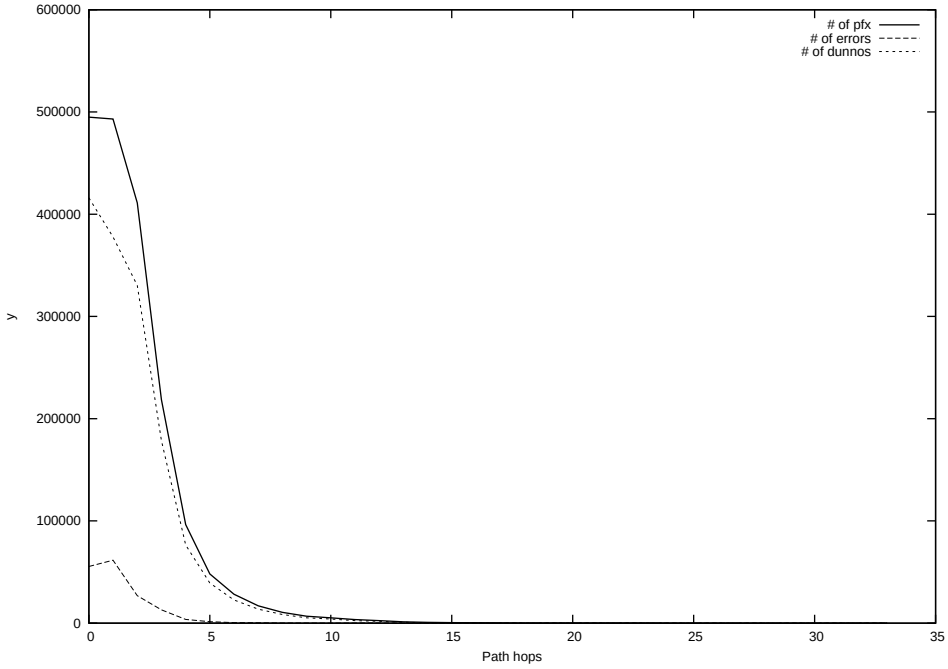
2014-05-26



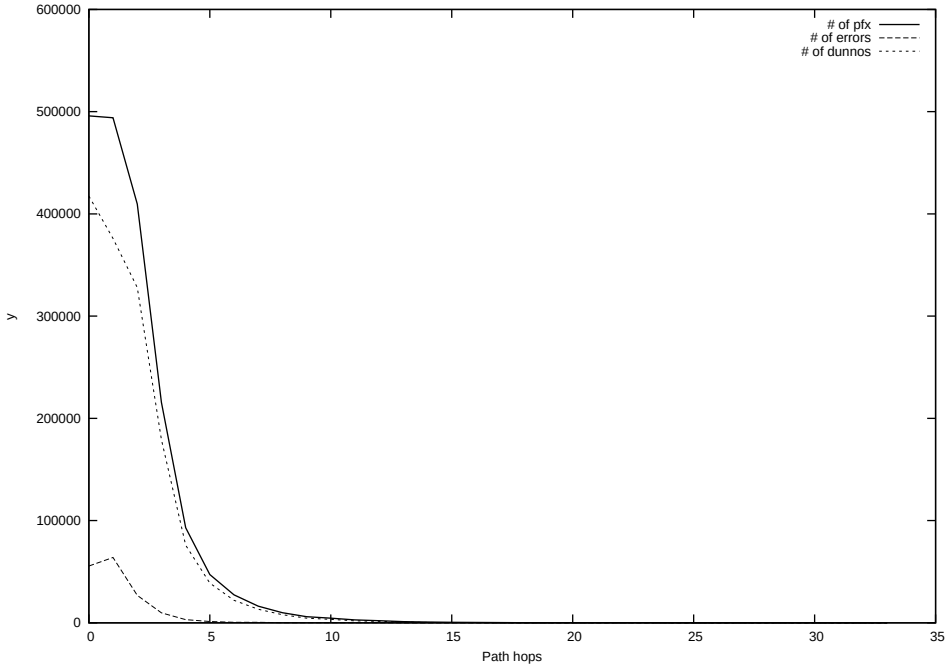
2014-05-27



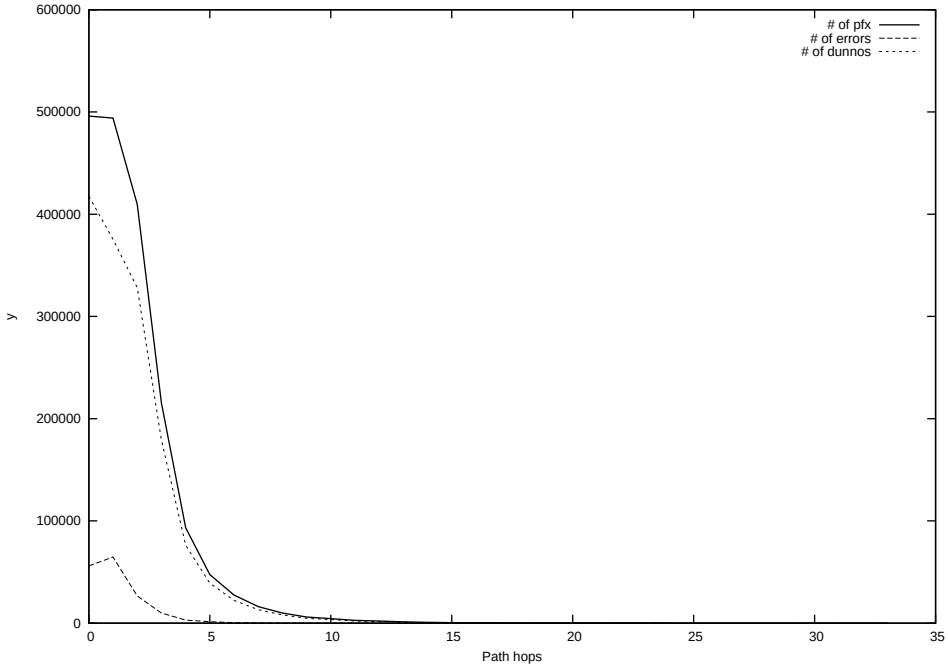
2014-05-28



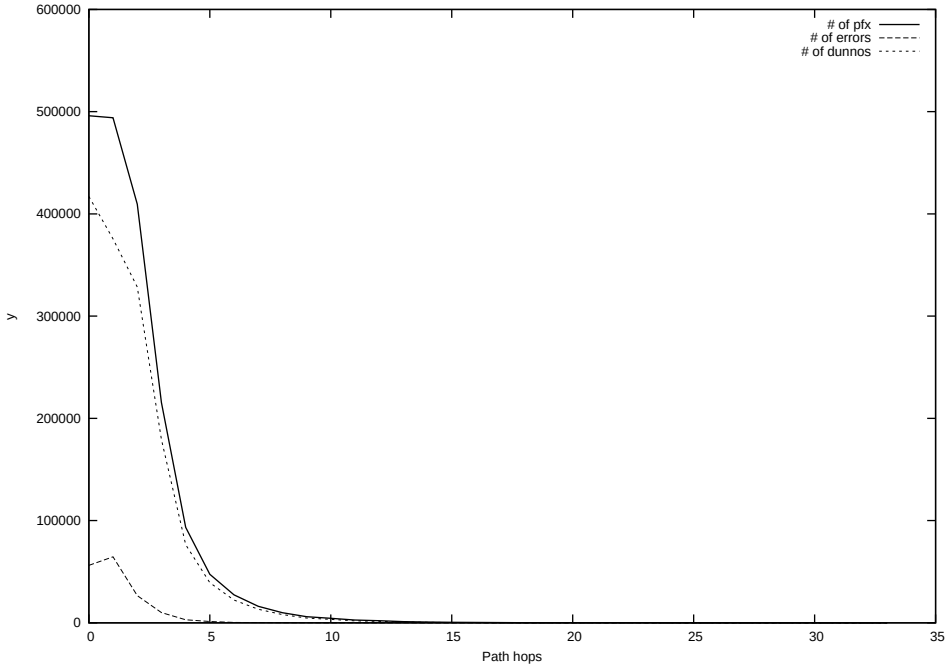
2014-05-29



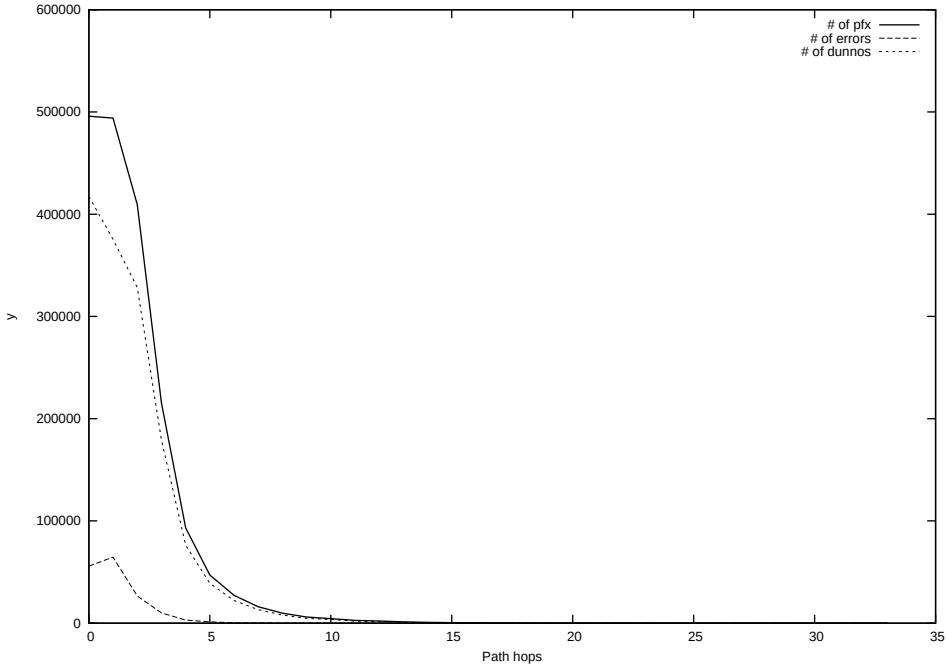
2014-05-30



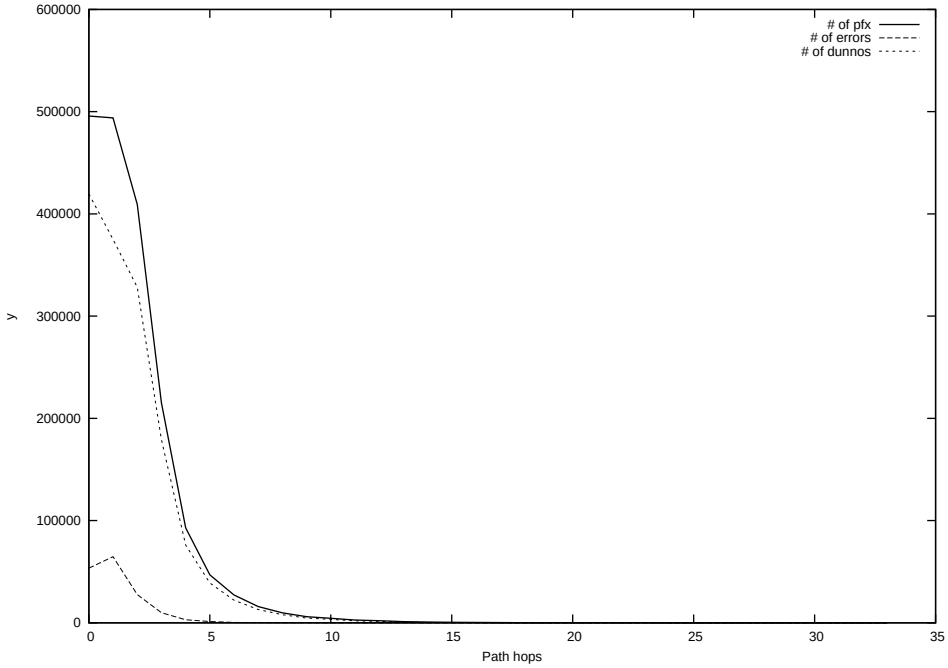
2014-05-31



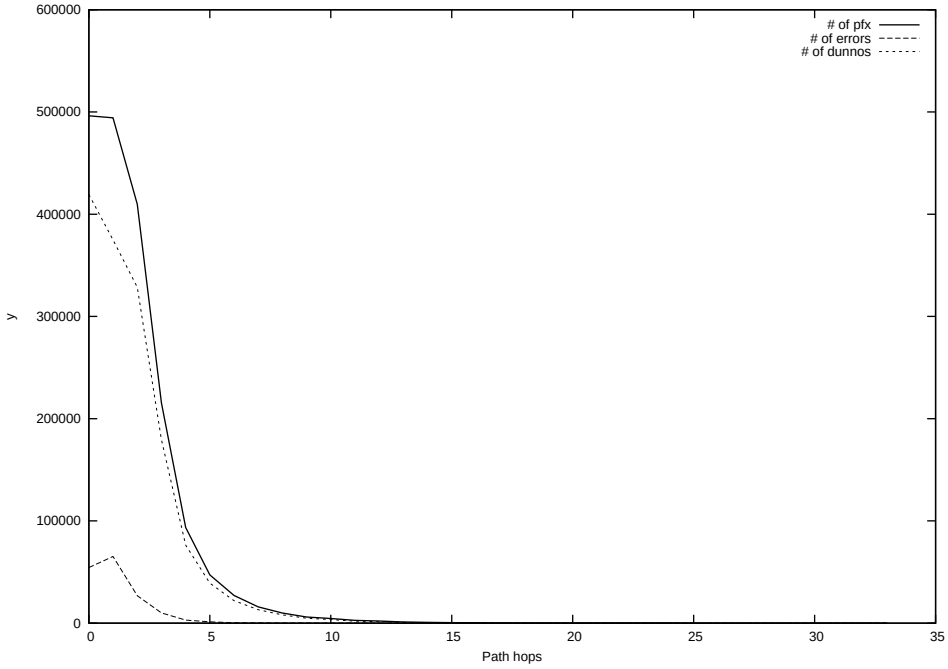
2014-06-01



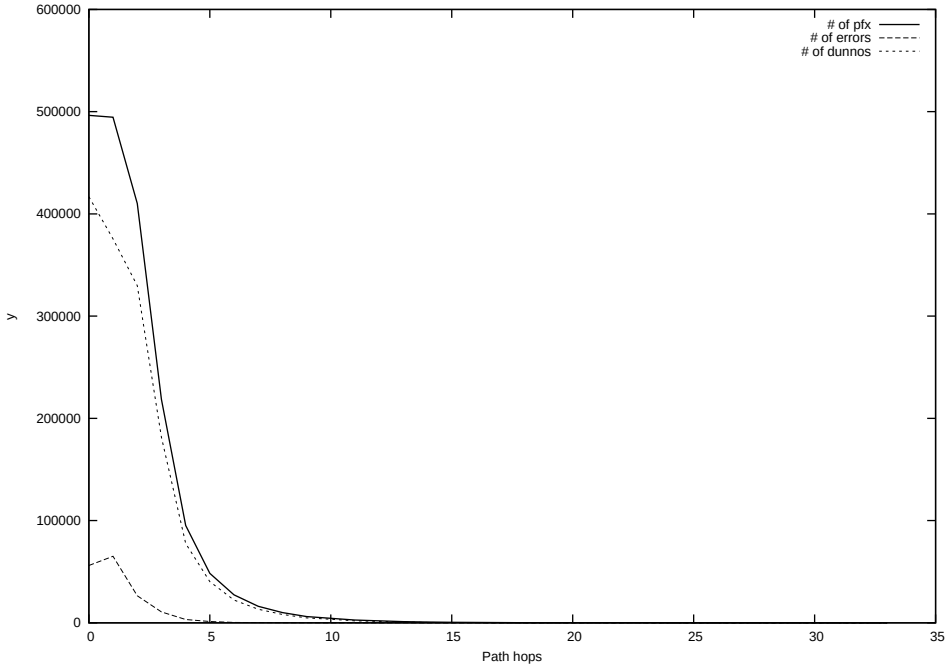
2014-06-02



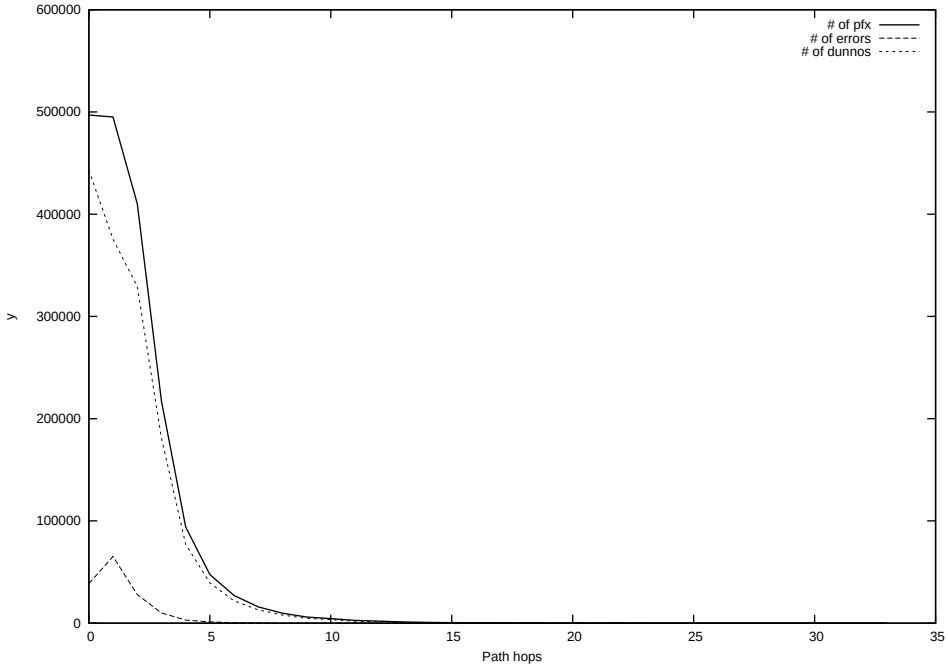
2014-06-03



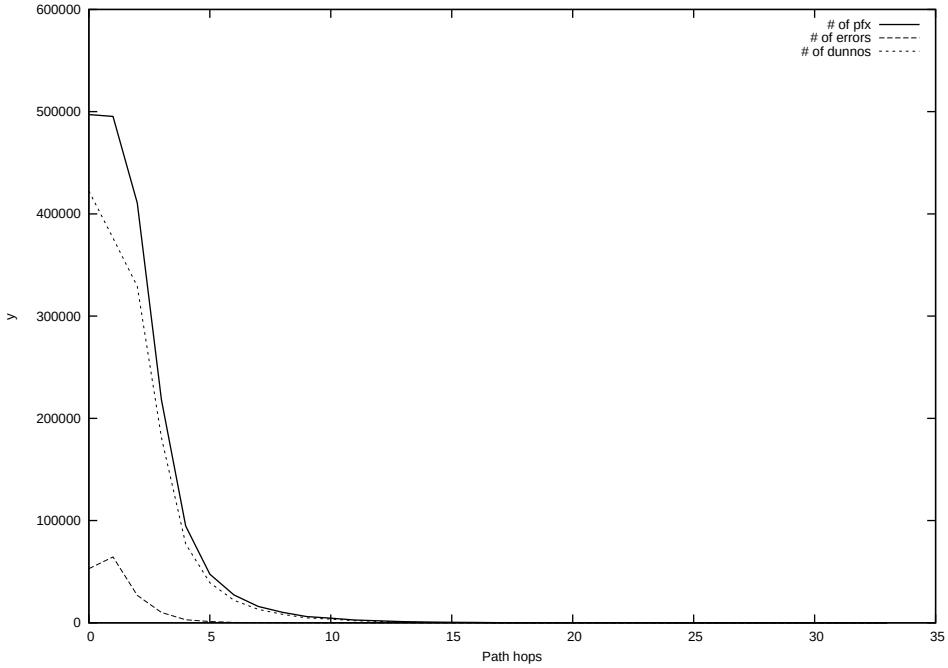
2014-06-04



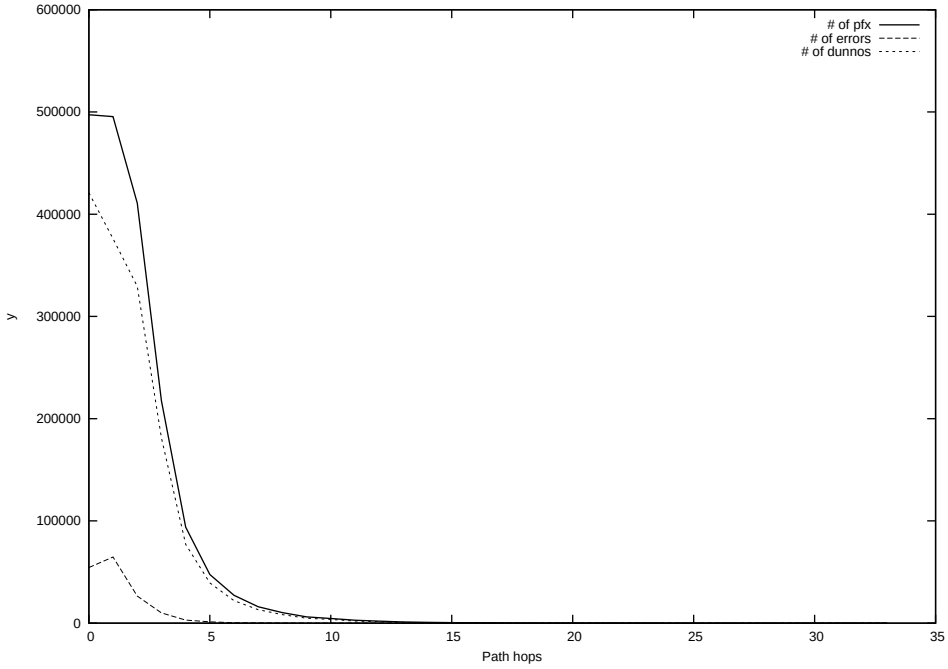
2014-06-05



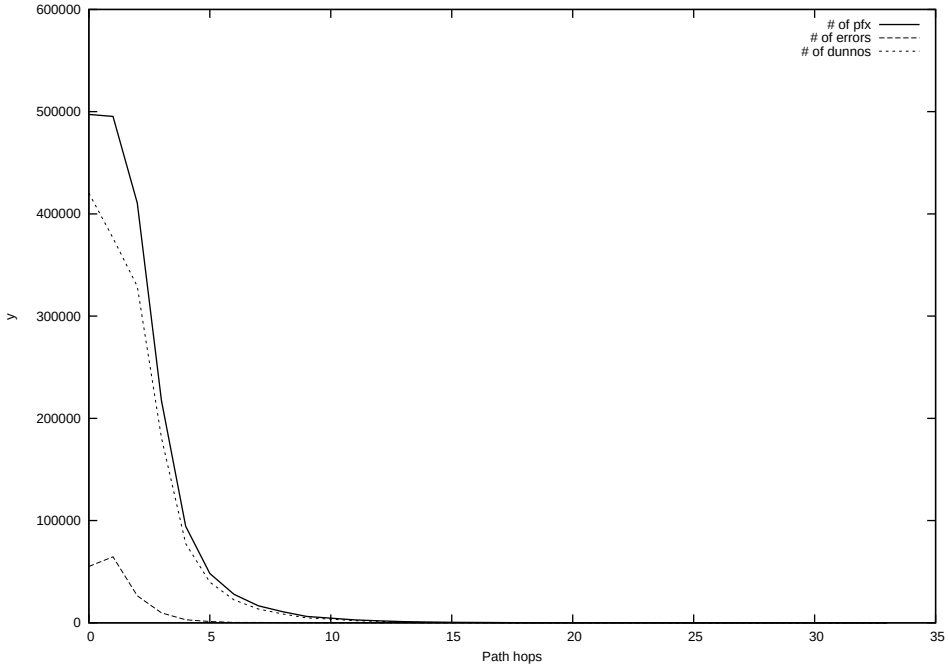
2014-06-06



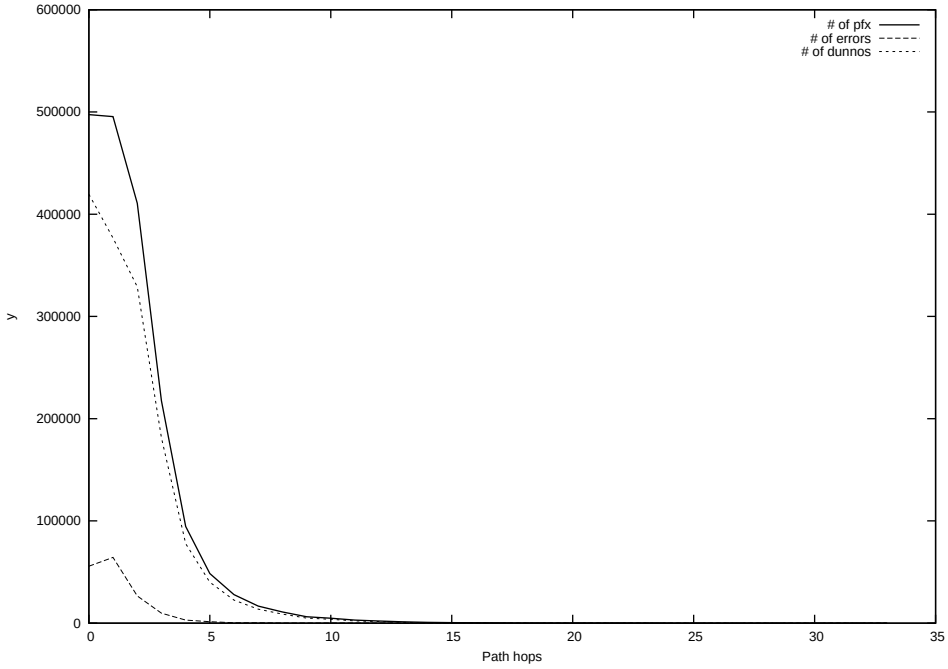
2014-06-07



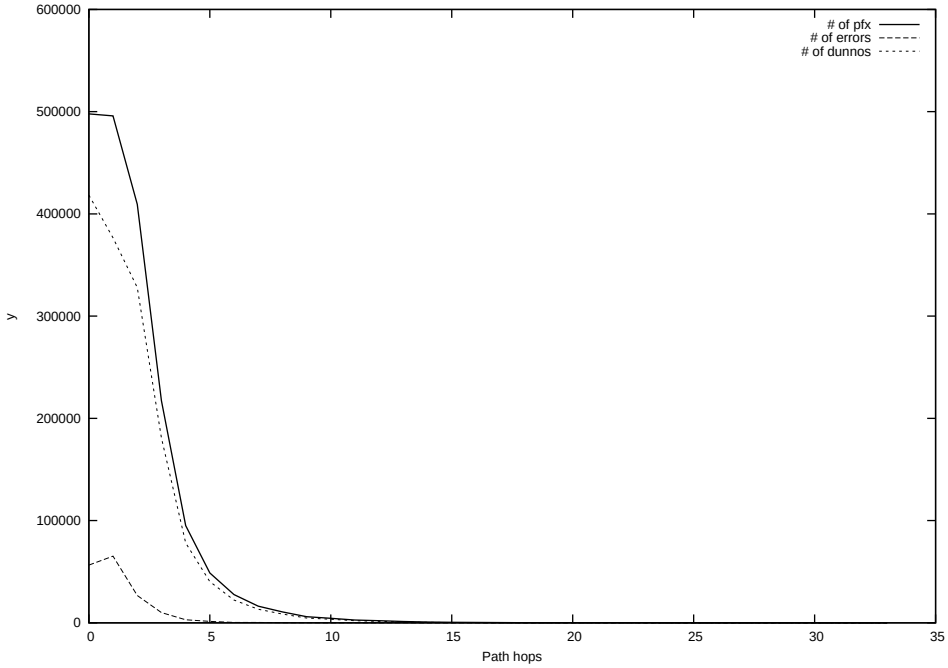
2014-06-08



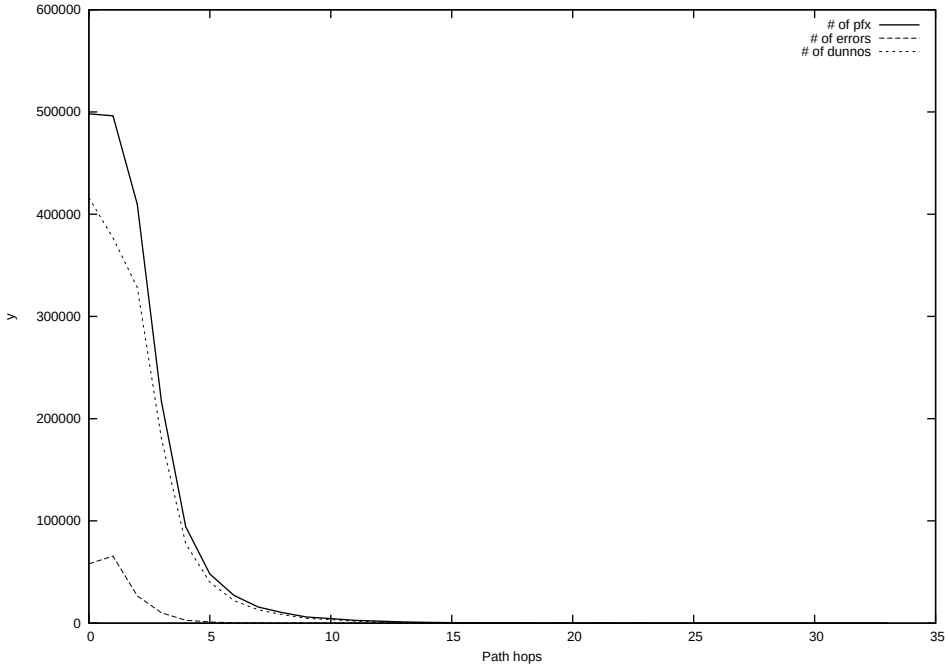
2014-06-09



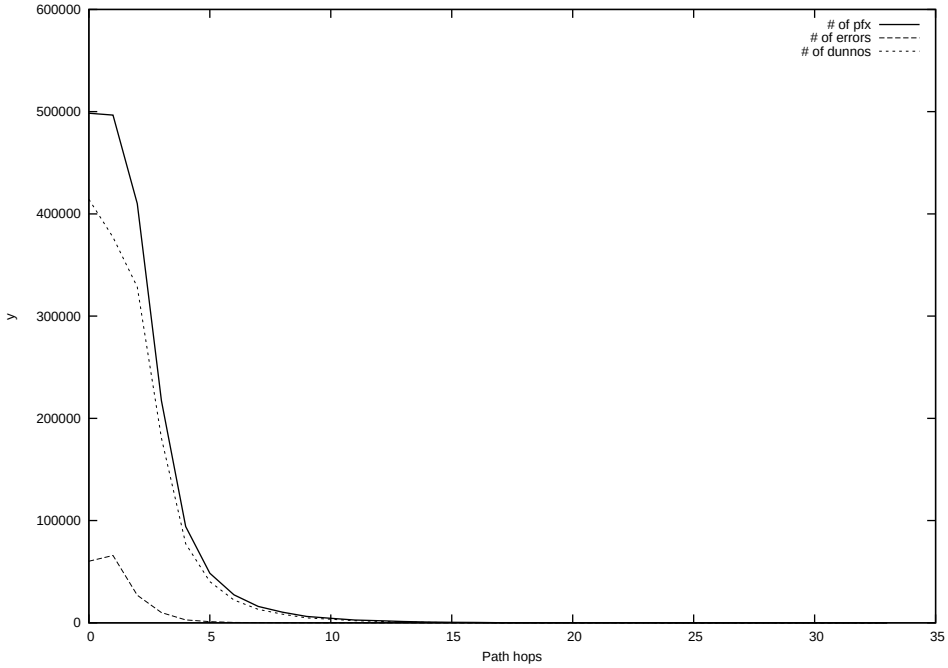
2014-06-10



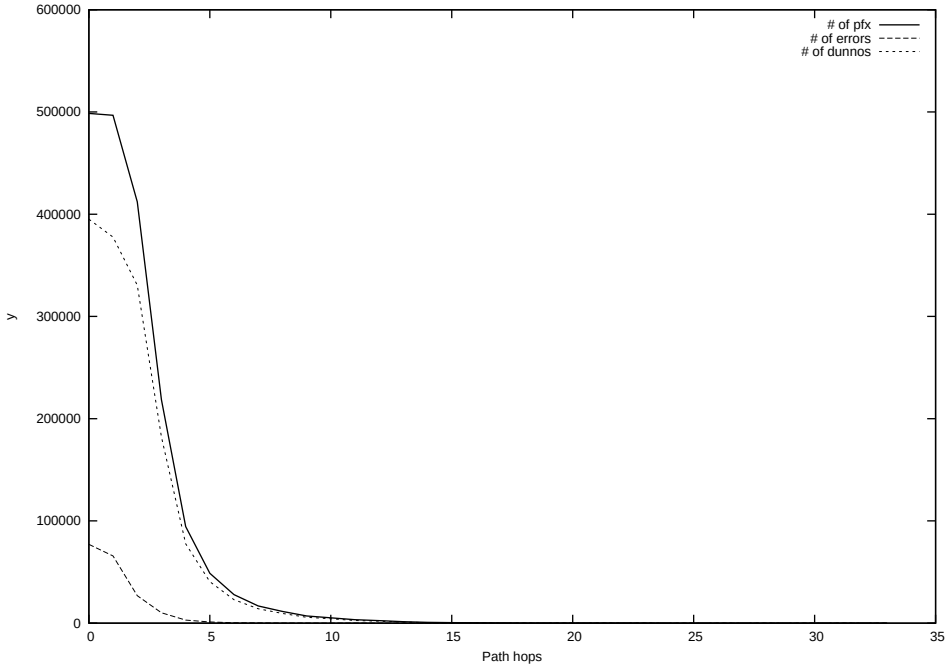
2014-06-11



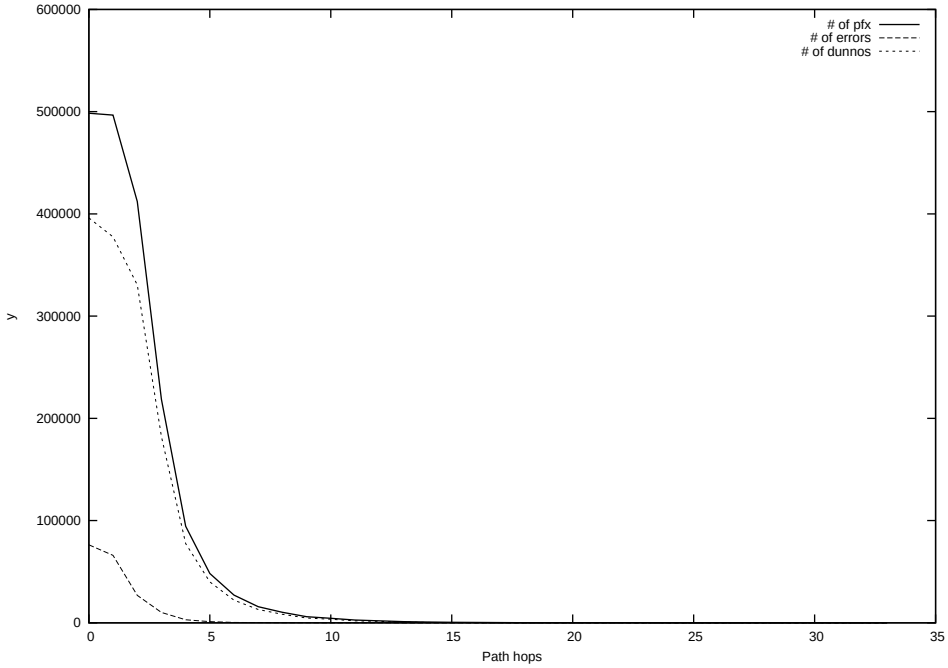
2014-06-12



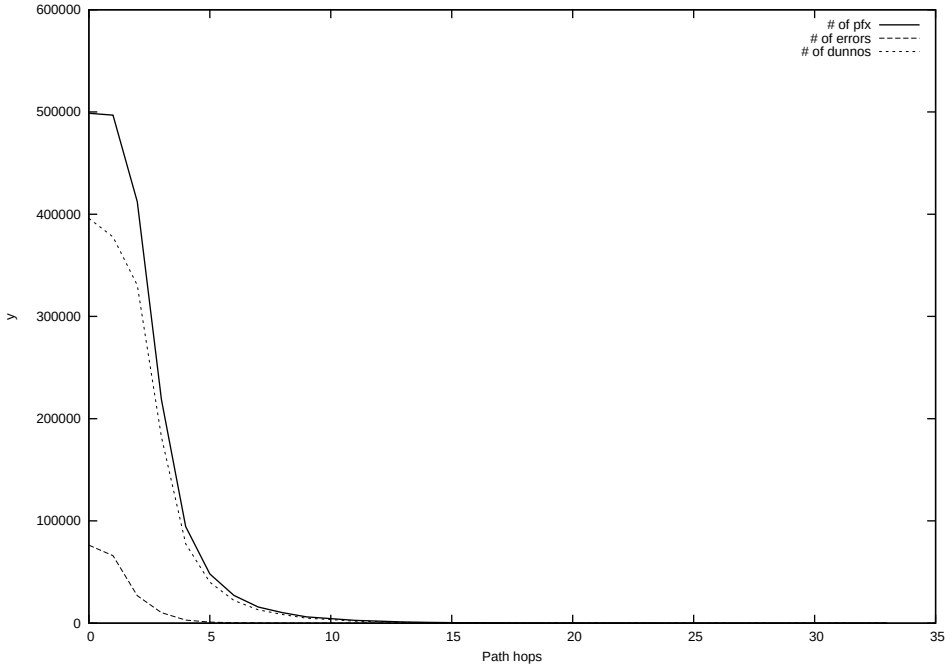
2014-06-13



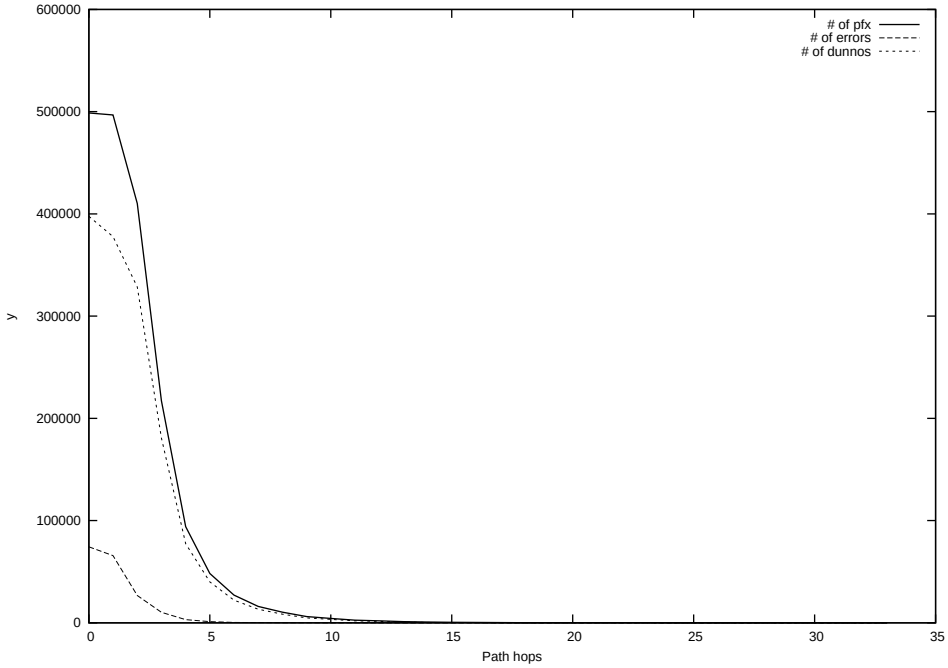
2014-06-14



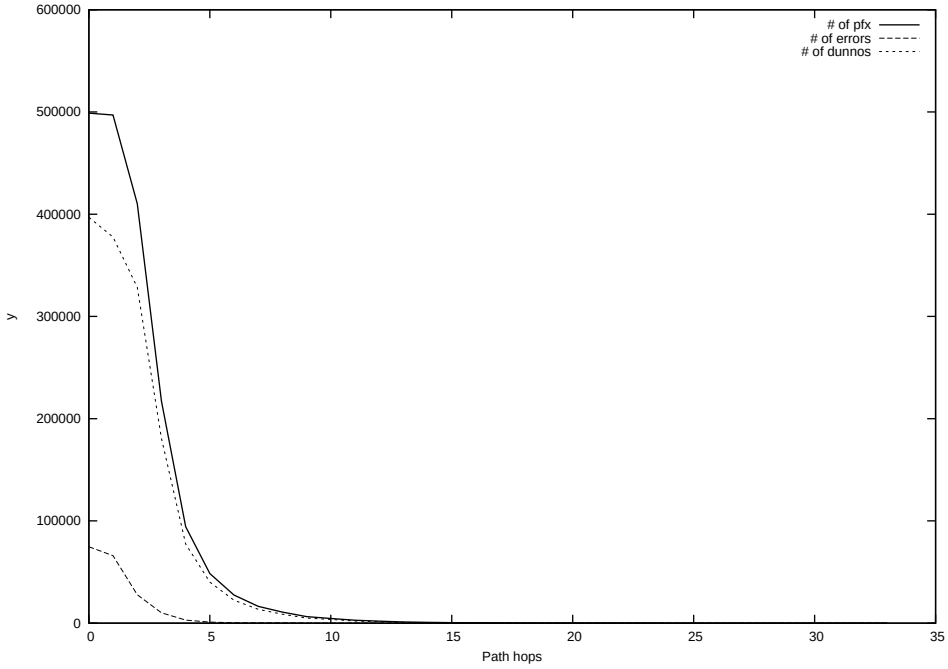
2014-06-15



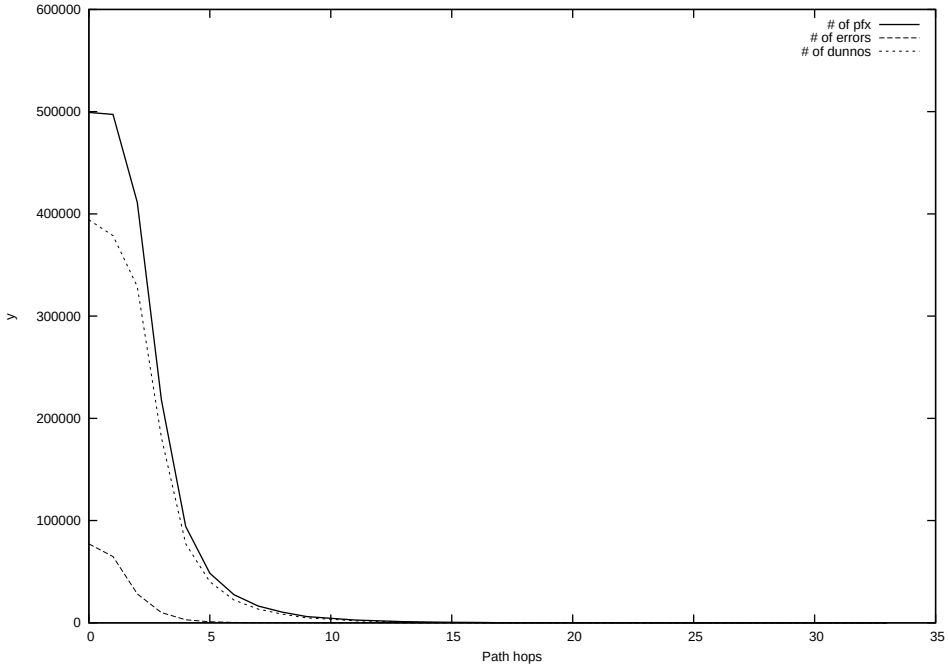
2014-06-16



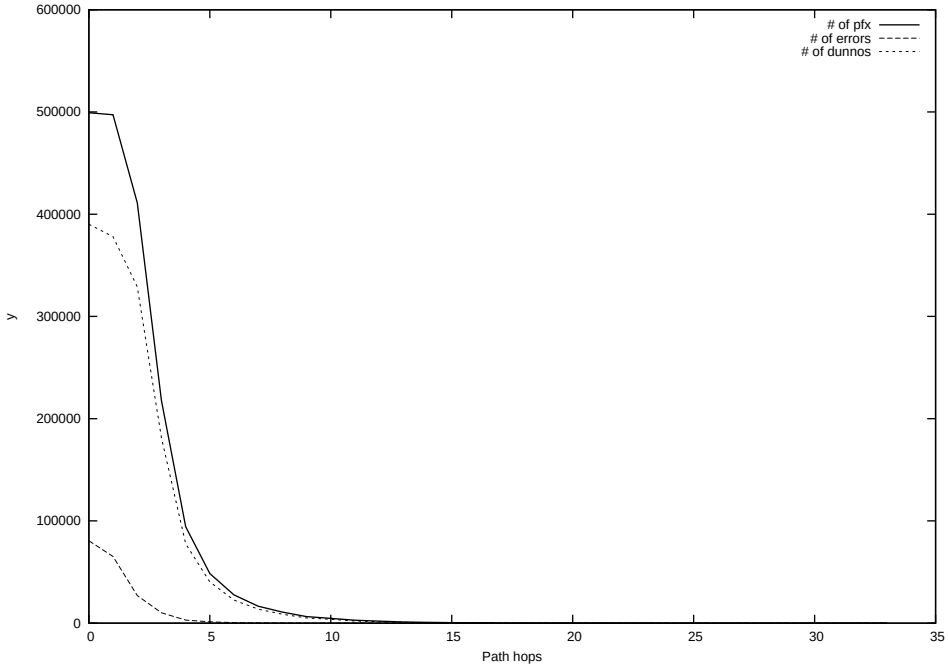
2014-06-17



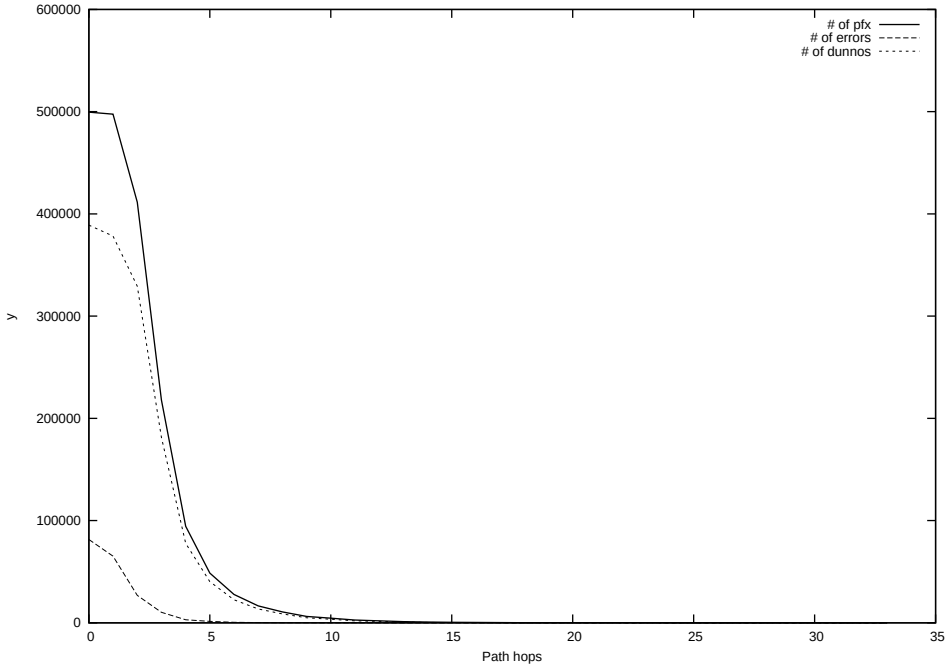
2014-06-18



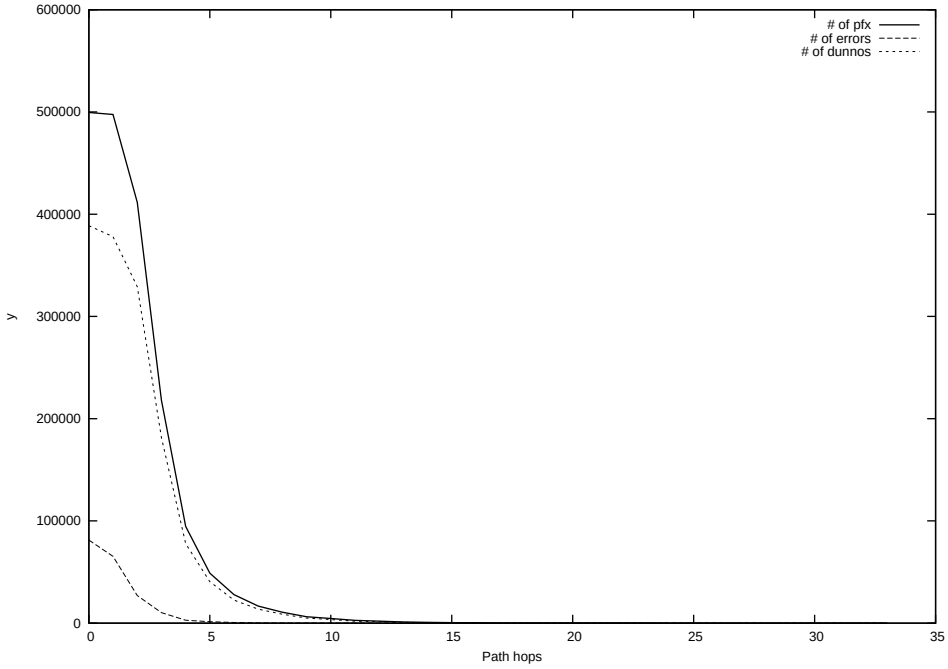
2014-06-19



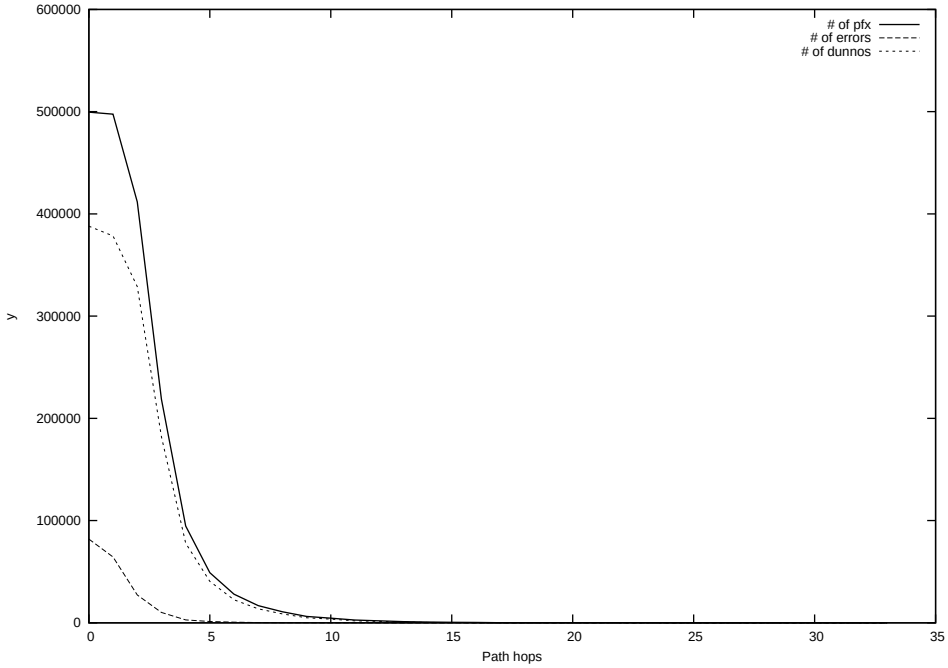
2014-06-20



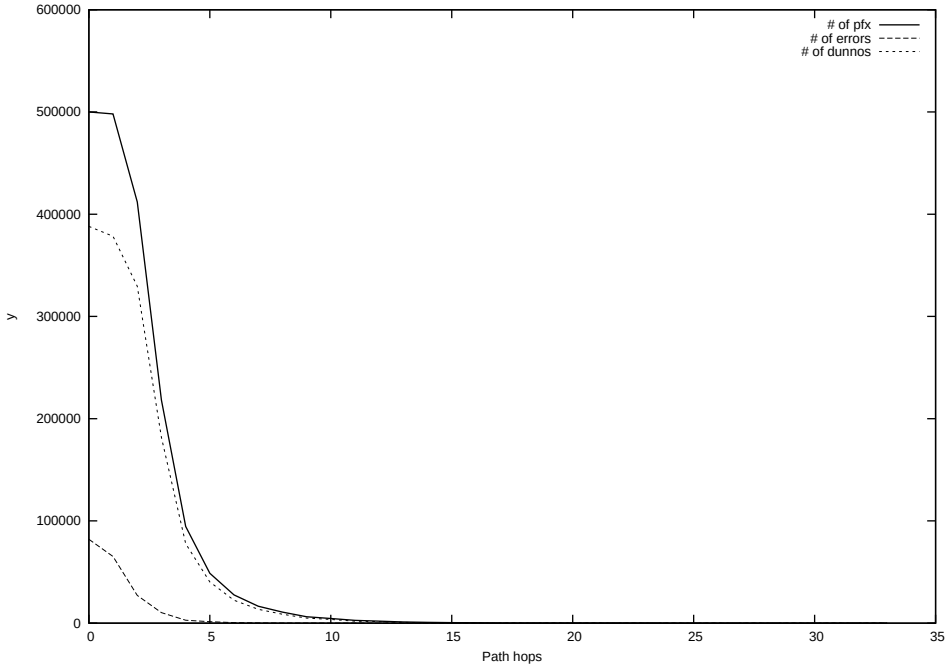
2014-06-21



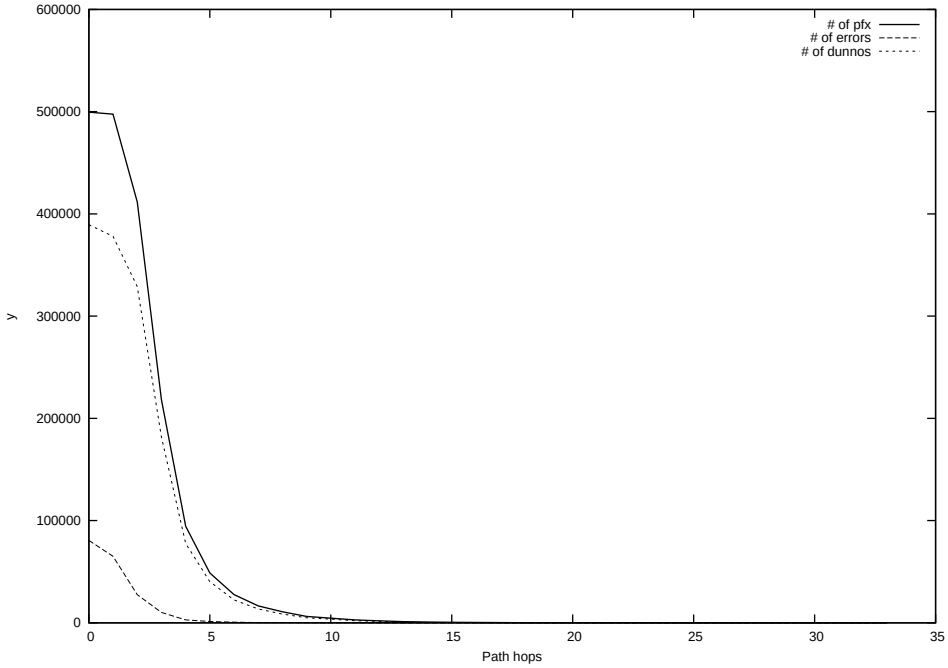
2014-06-22



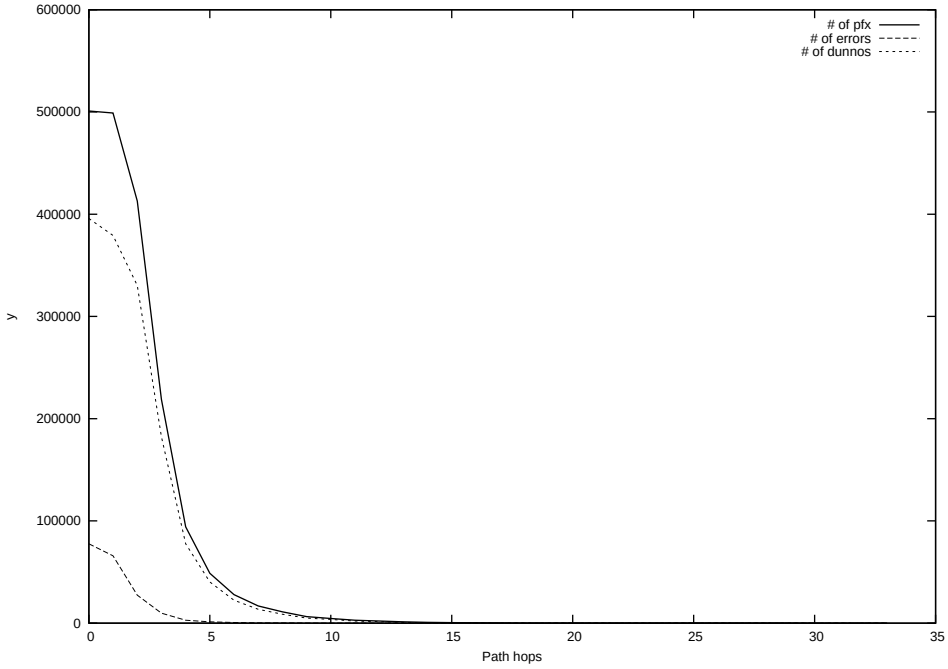
2014-06-23



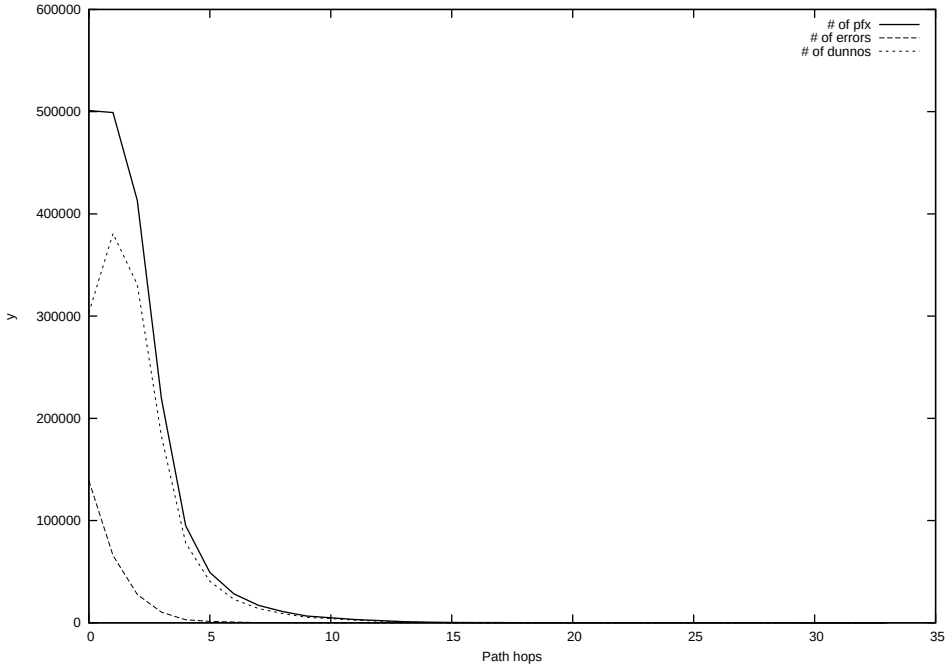
2014-06-24



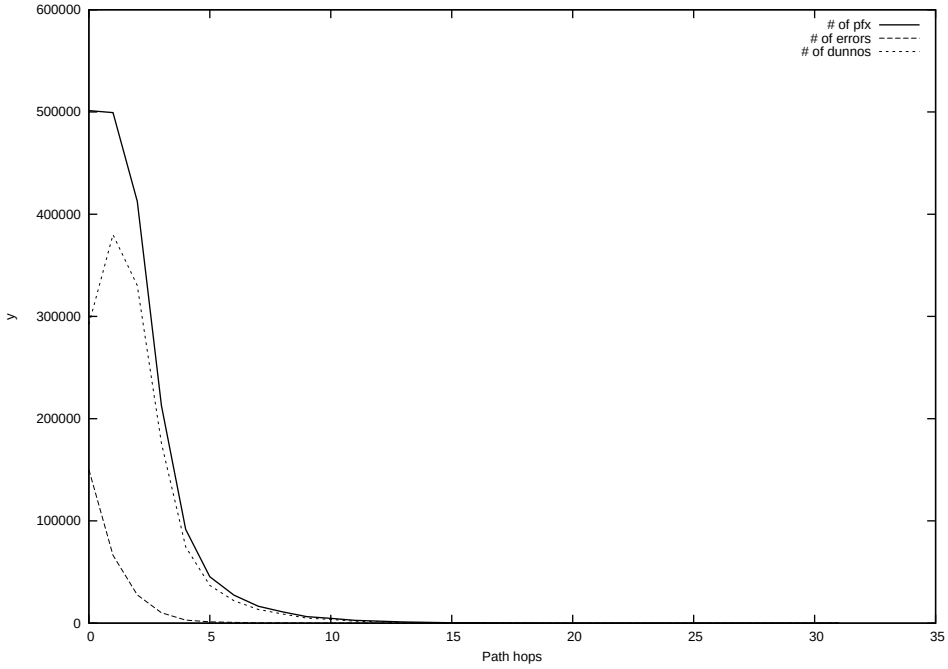
2014-06-25



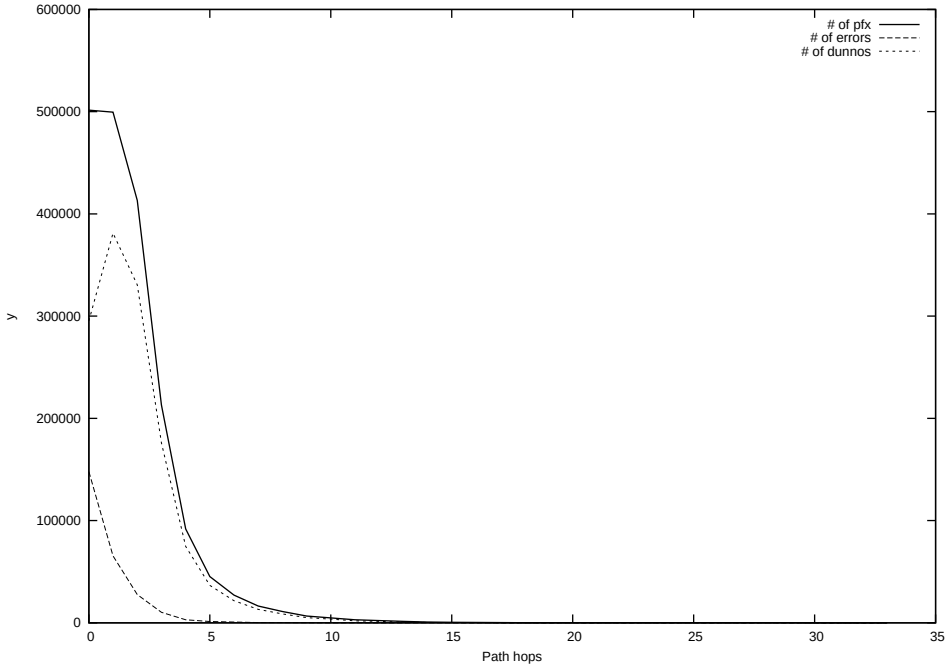
2014-06-26



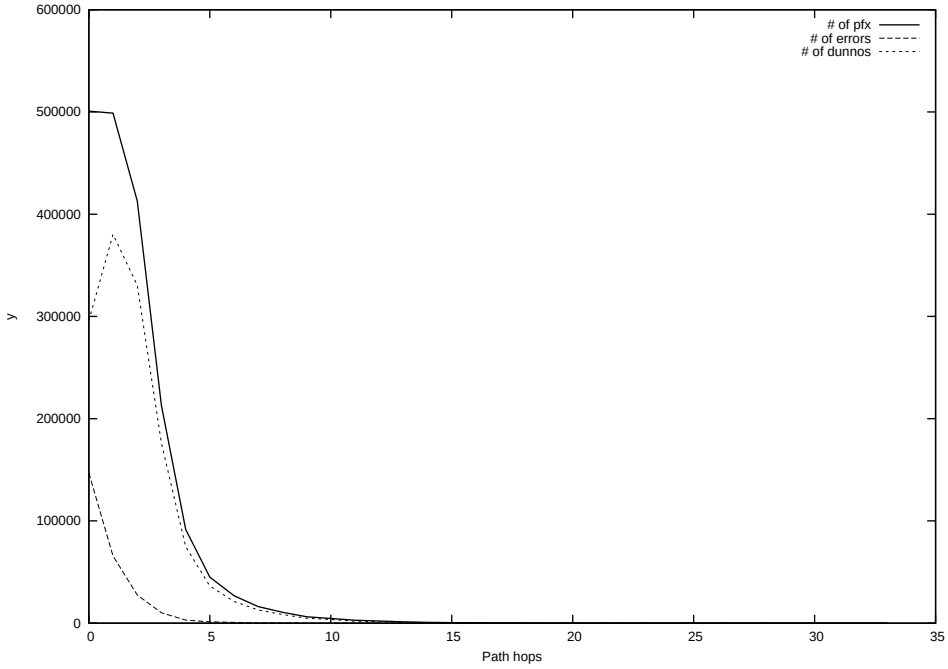
2014-06-27



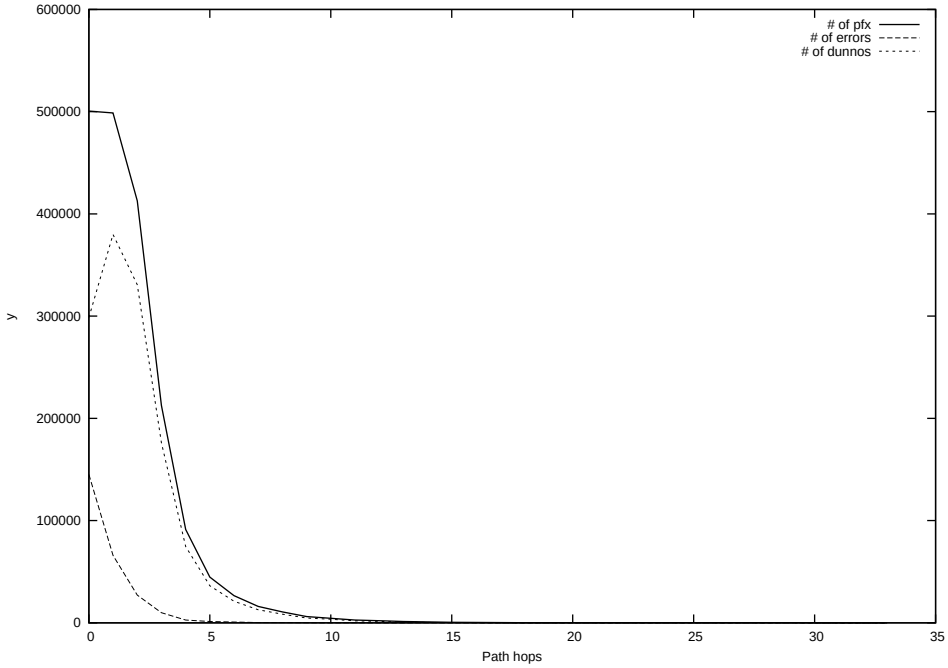
2014-06-28



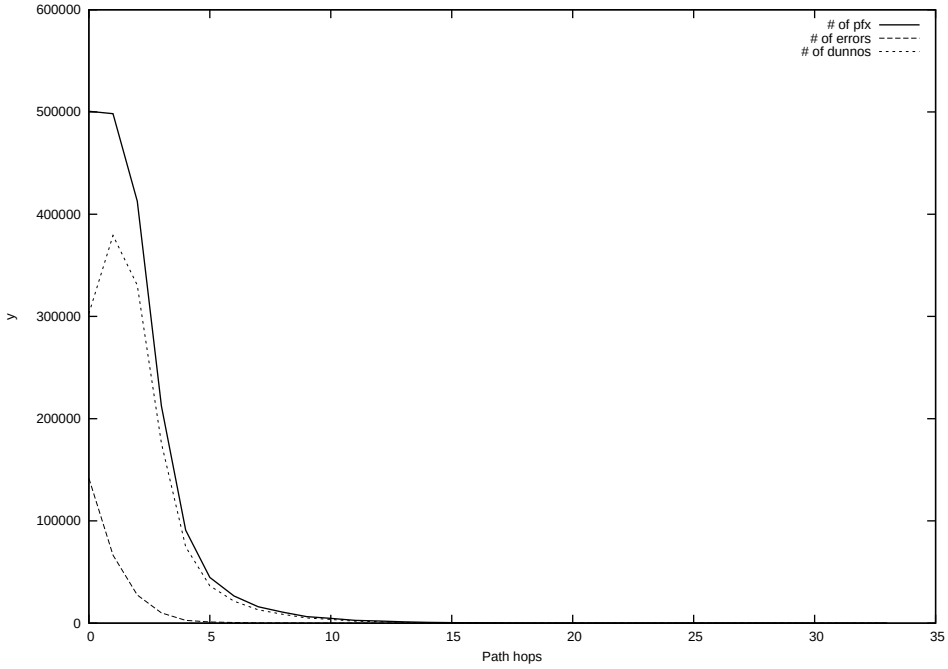
2014-06-29



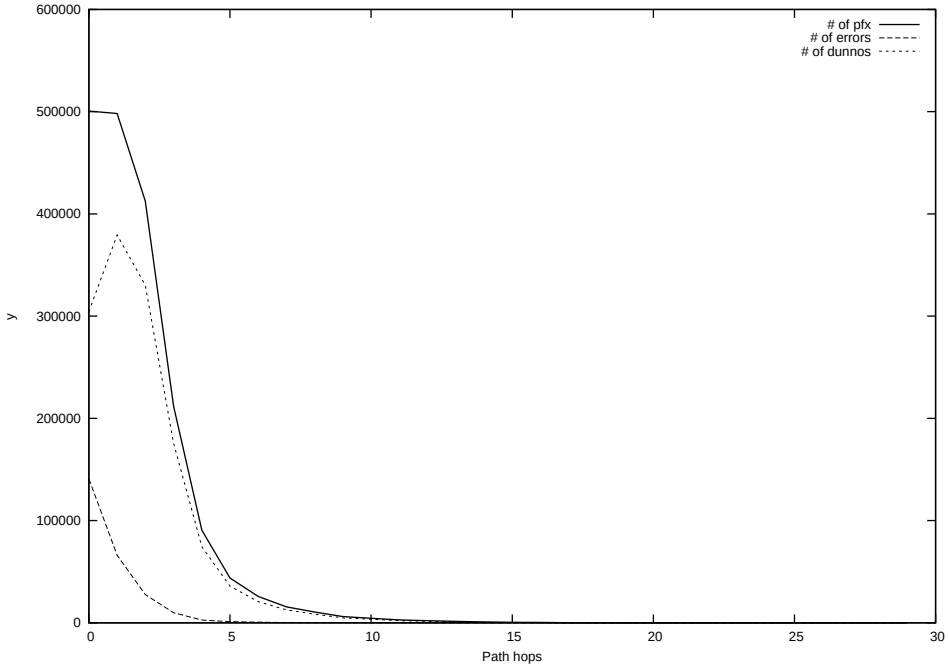
2014-06-30



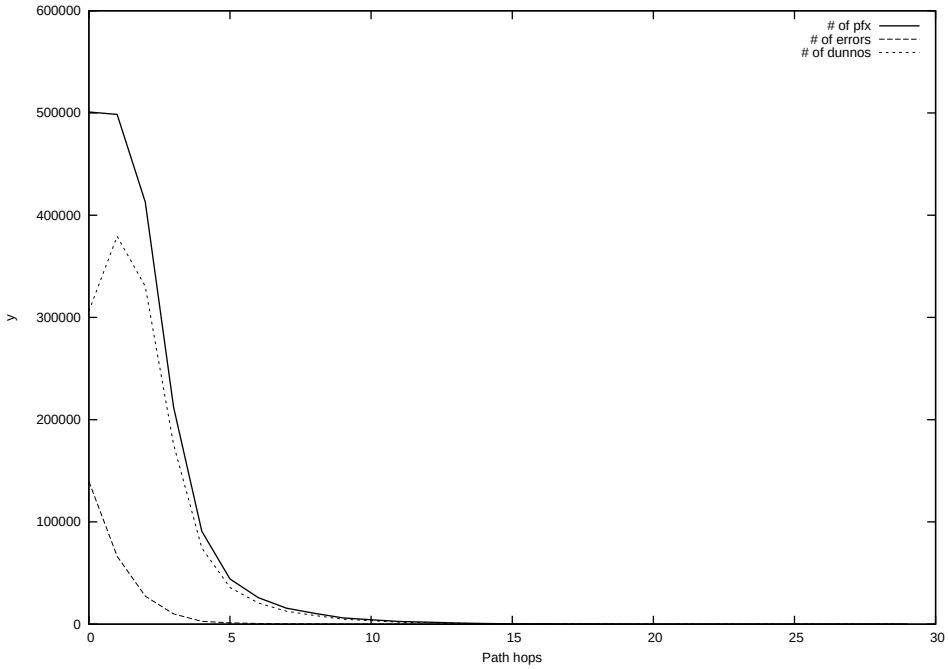
2014-07-01



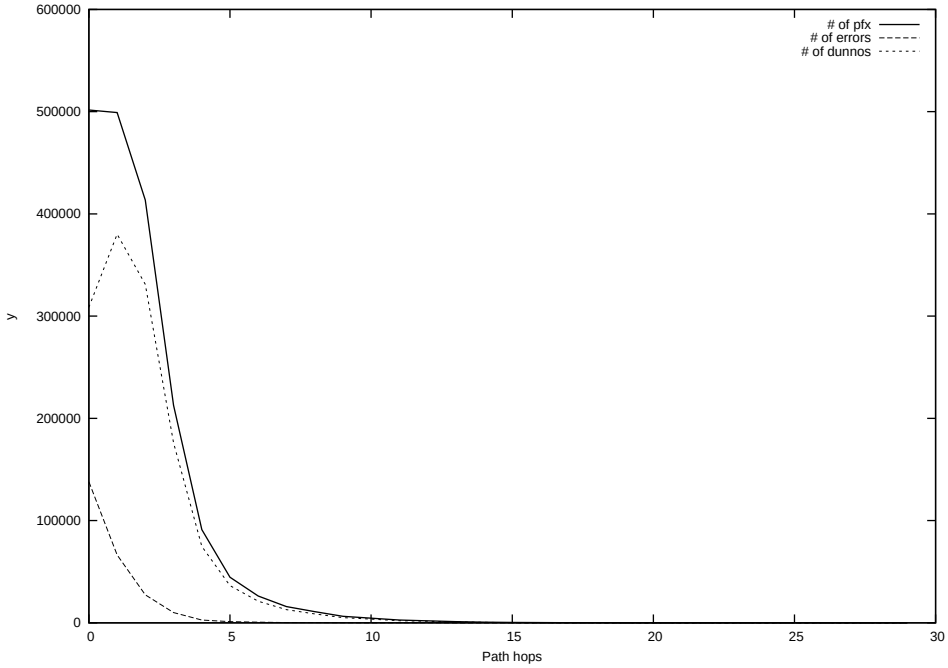
2014-07-02



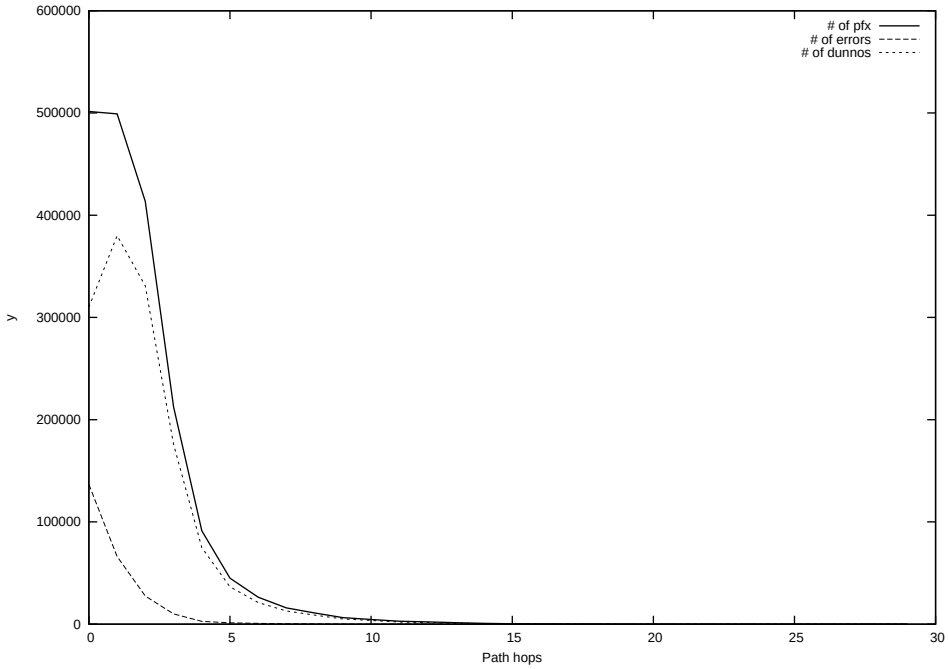
2014-07-03



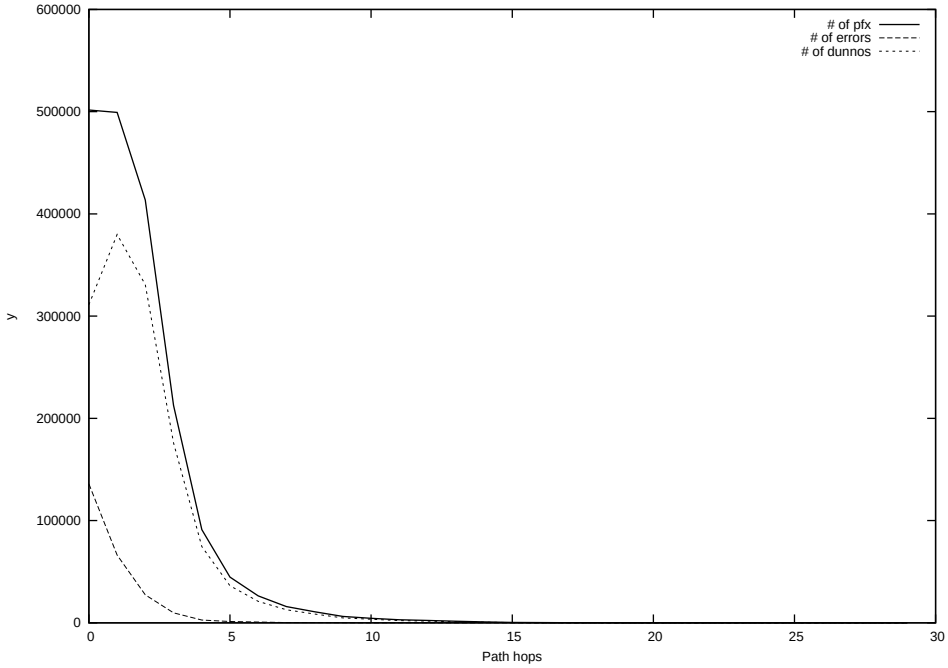
2014-07-04



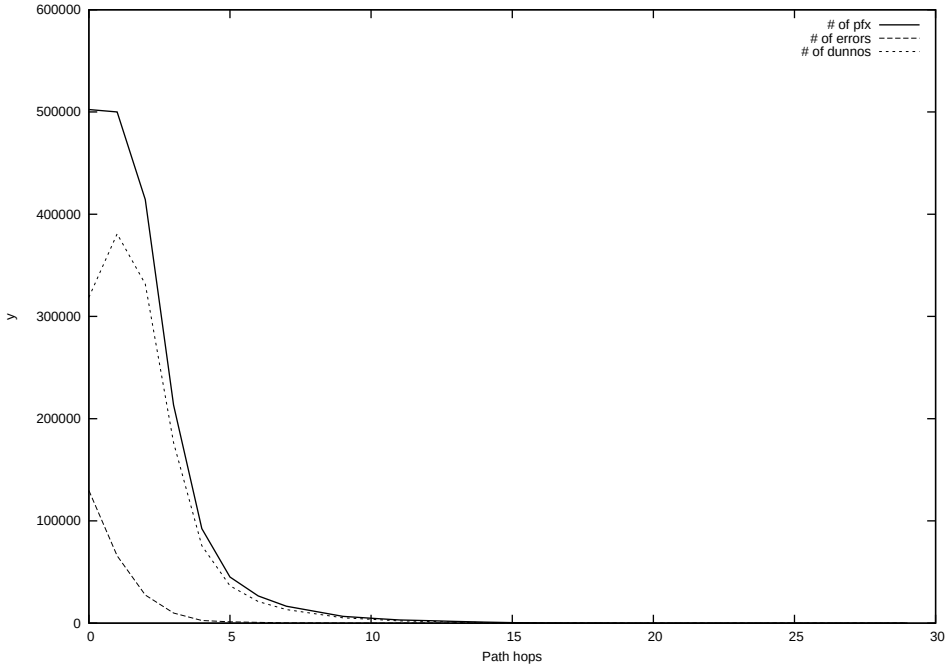
2014-07-05



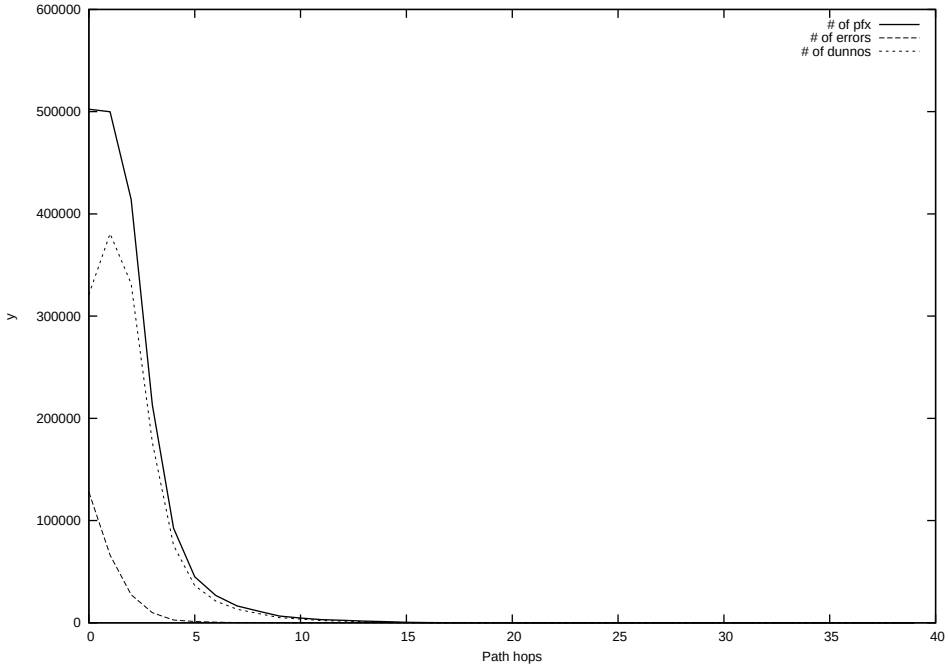
2014-07-06



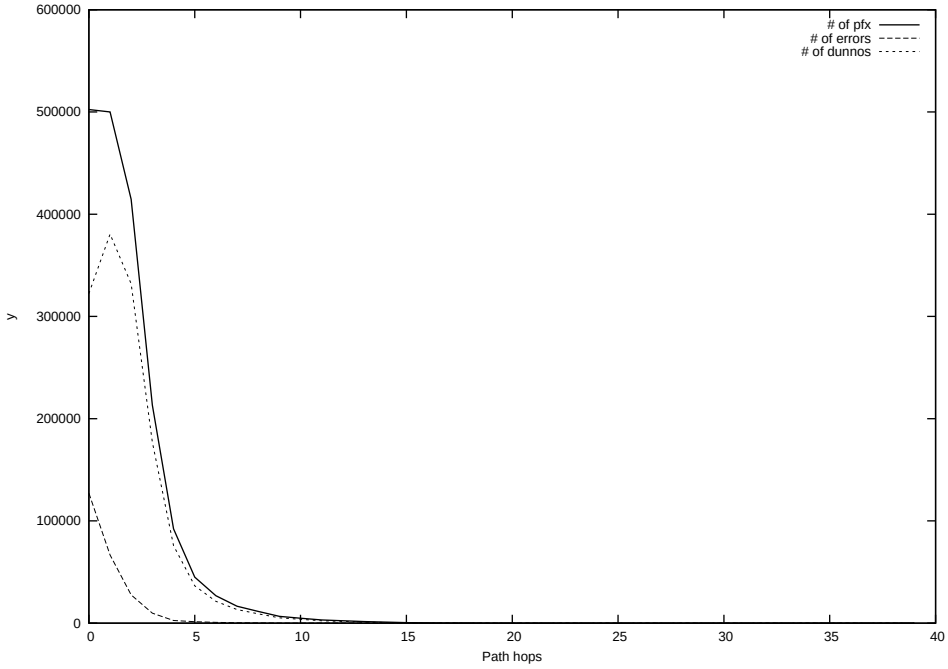
2014-07-07



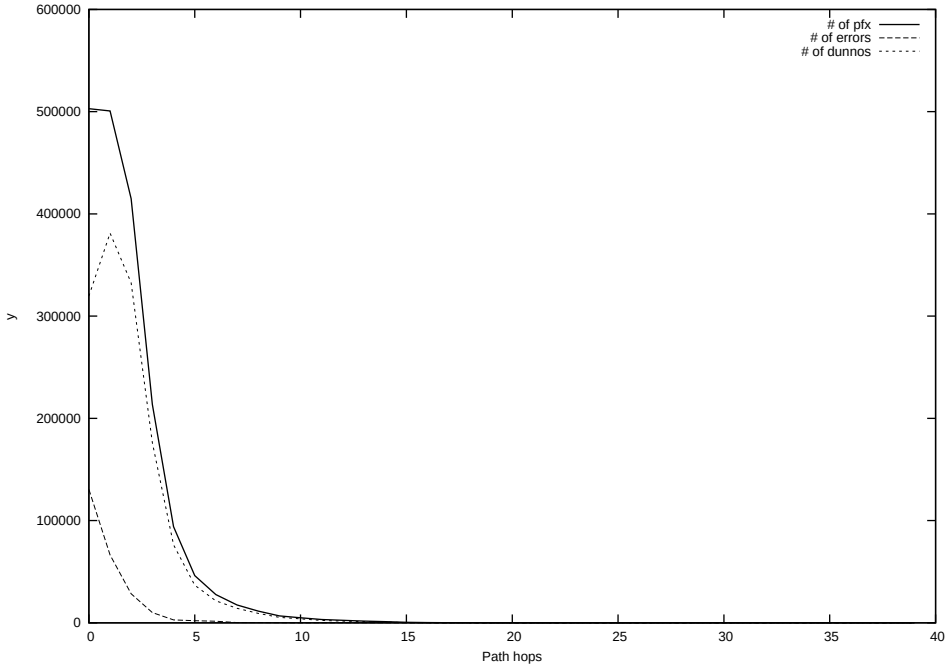
2014-07-08



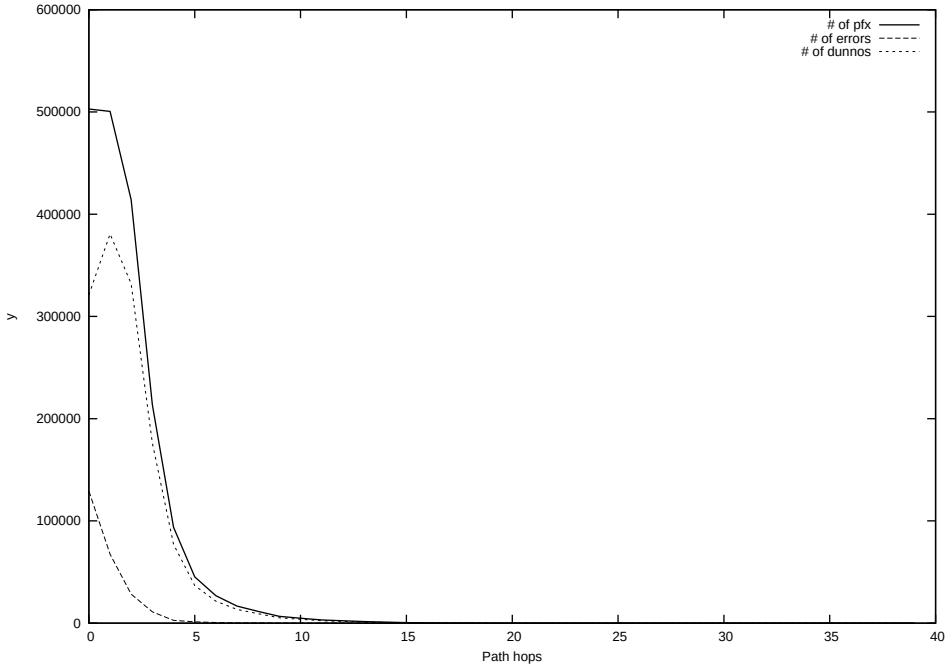
2014-07-09



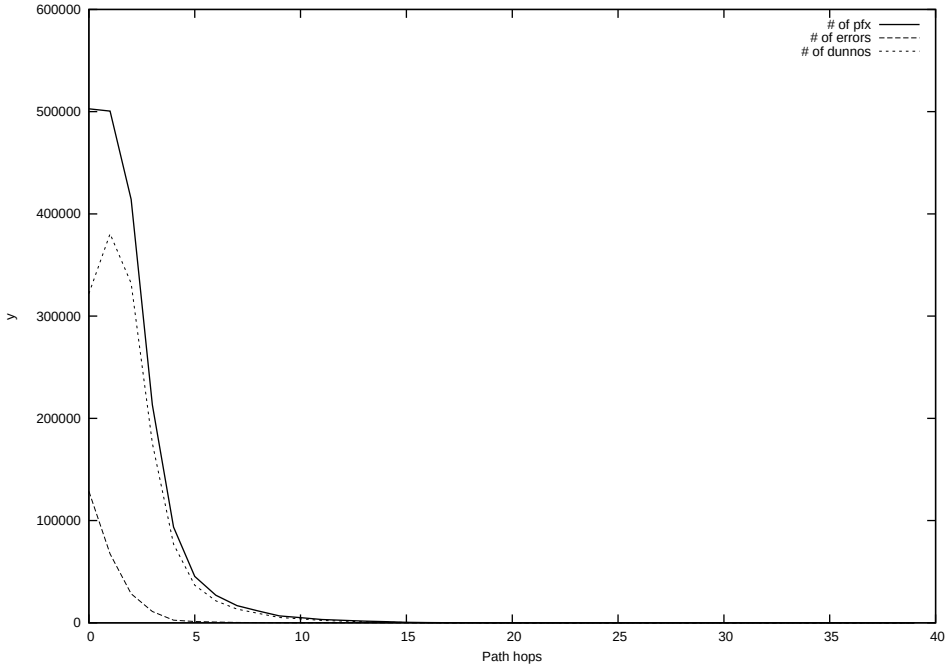
2014-07-10



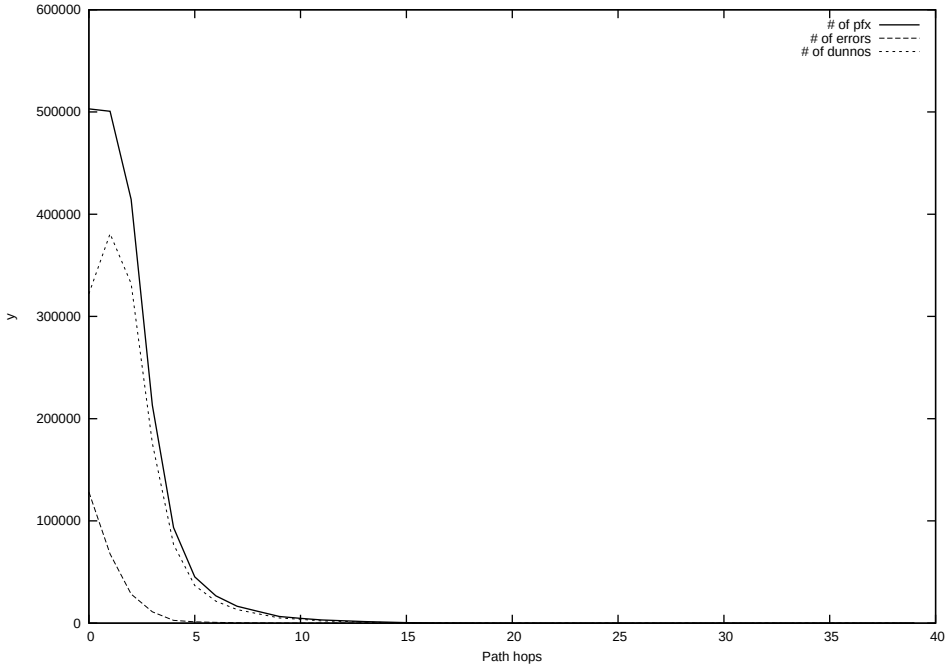
2014-07-11



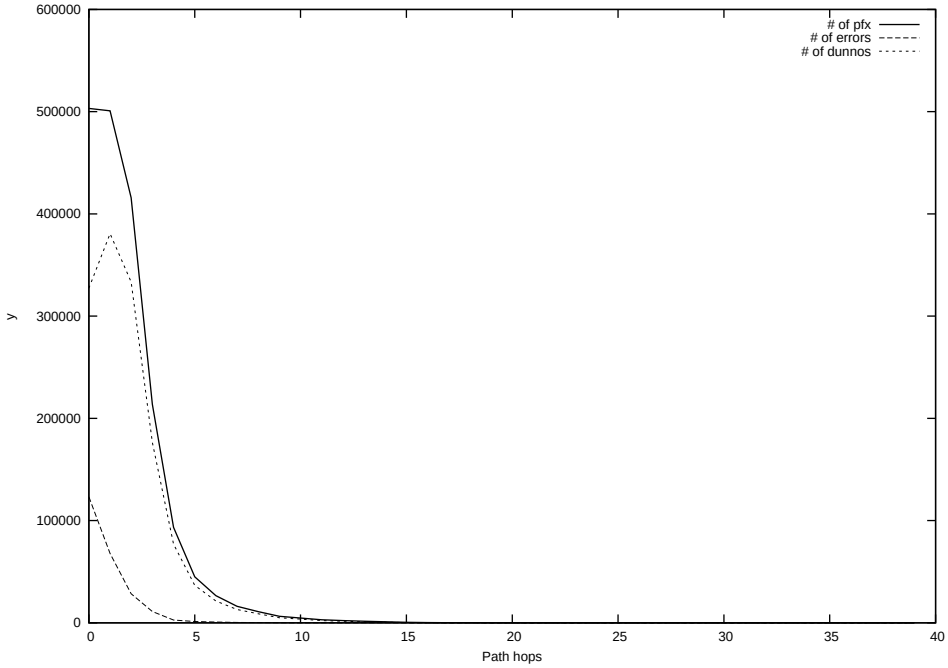
2014-07-12



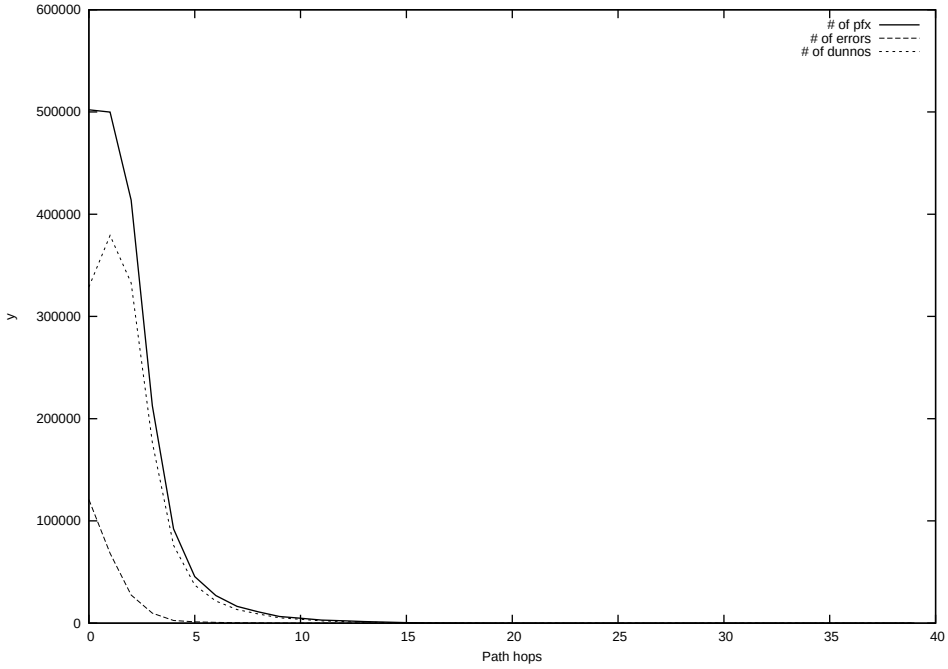
2014-07-13



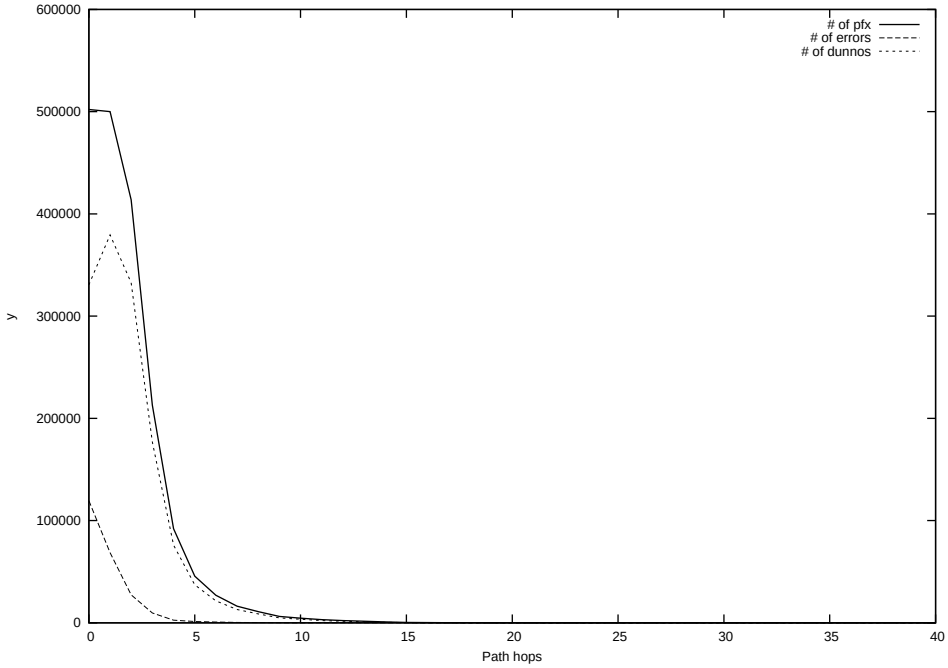
2014-07-14



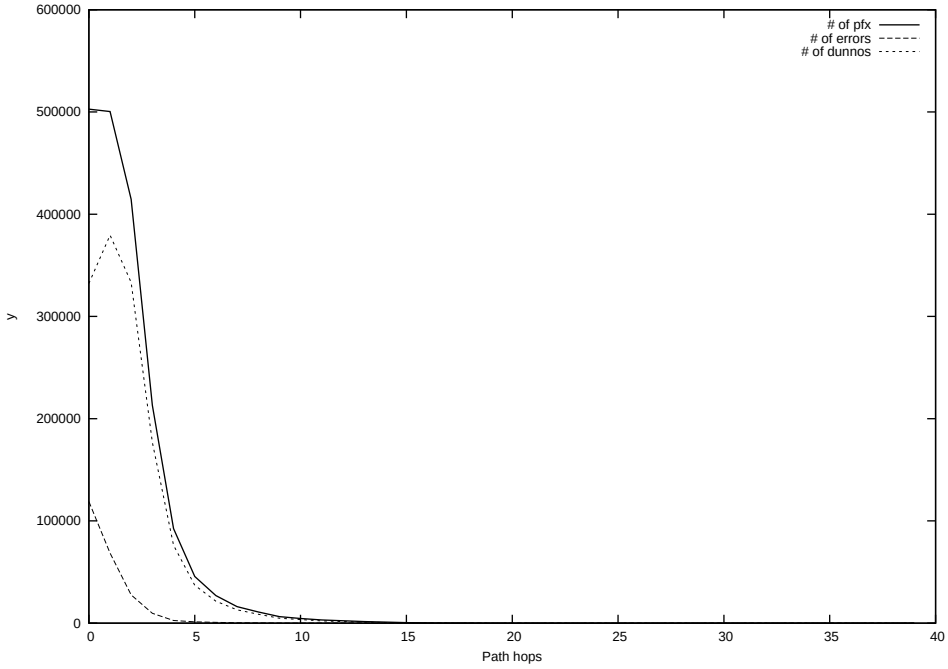
2014-07-15



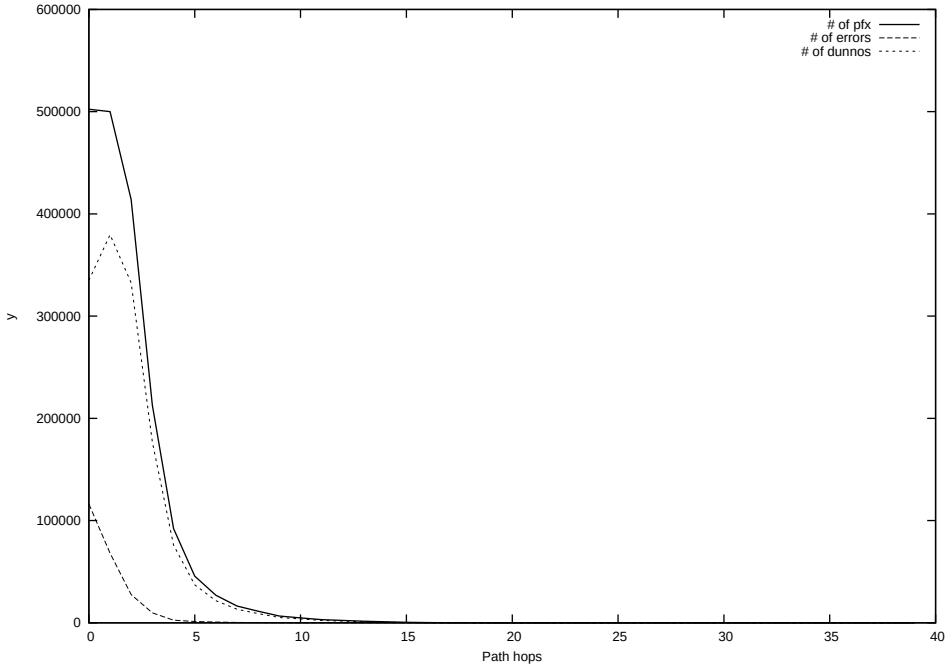
2014-07-16



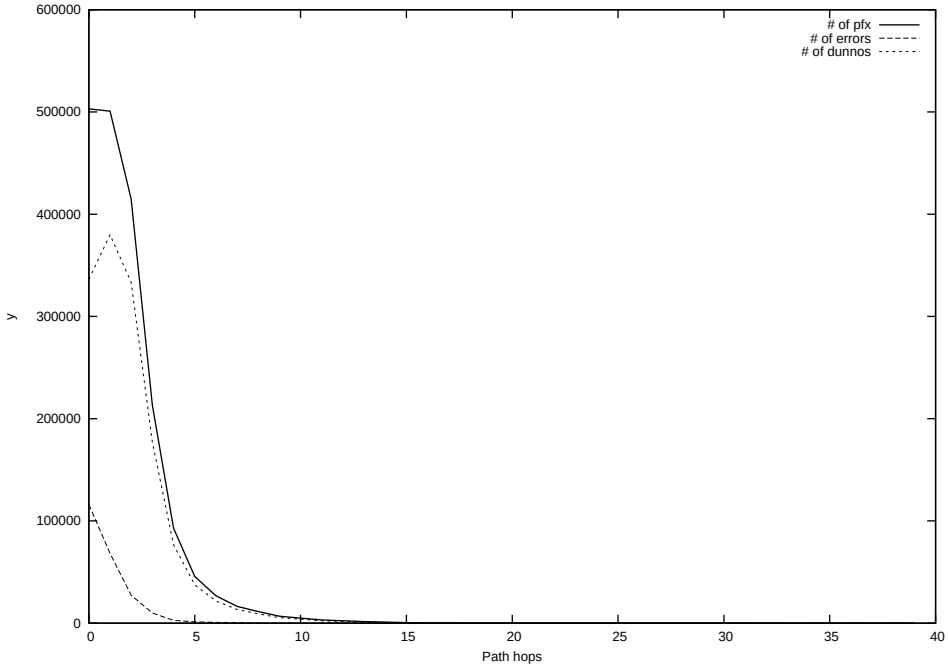
2014-07-17



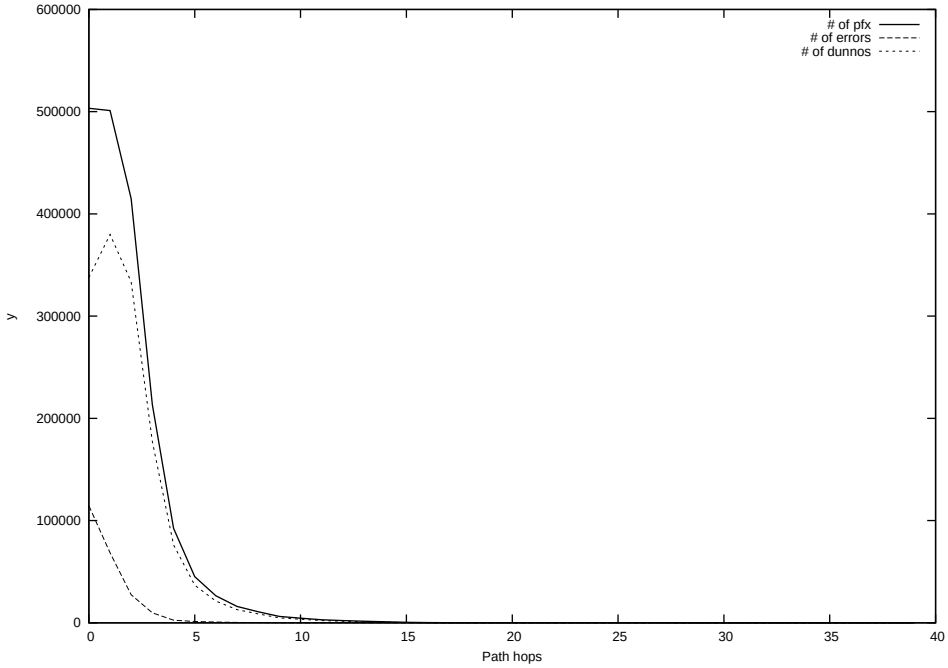
2014-07-18



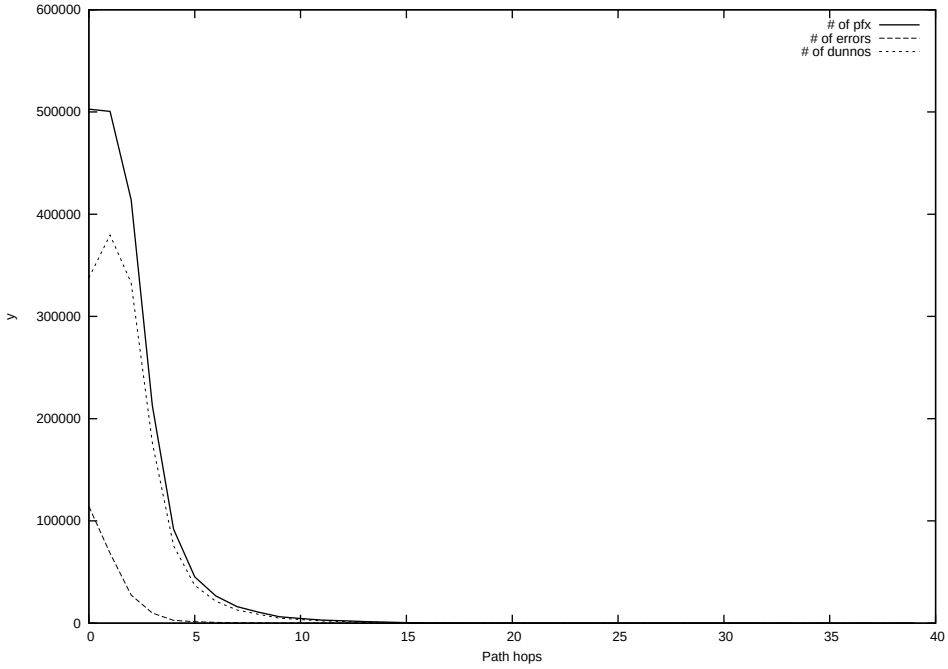
2014-07-19



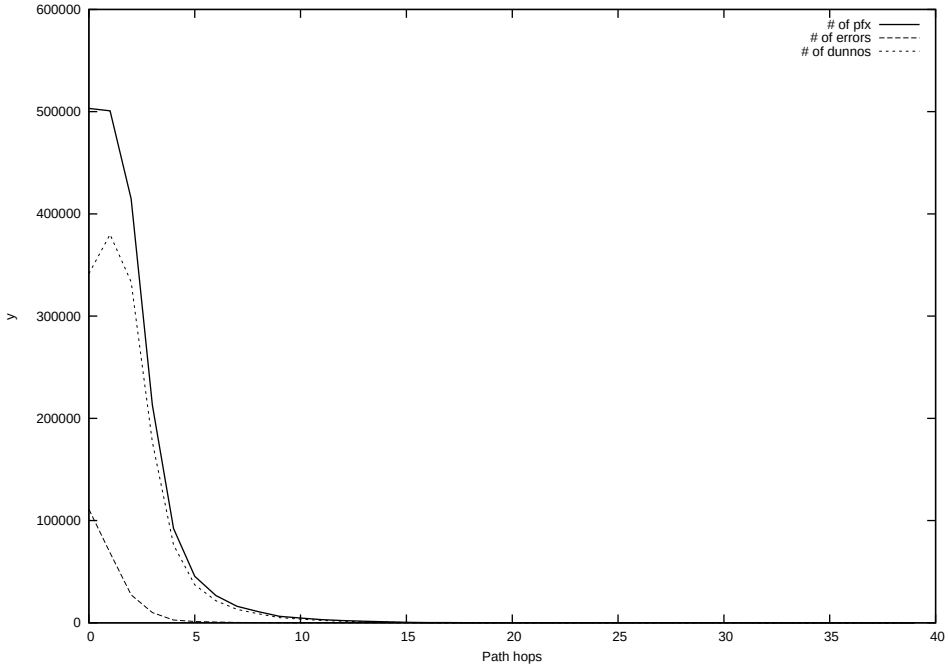
2014-07-20



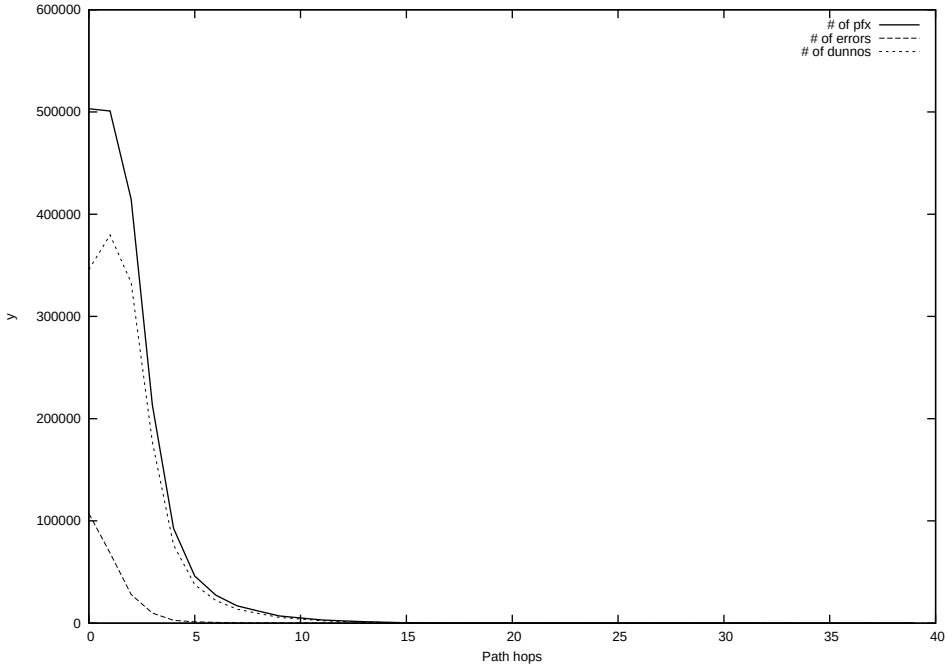
2014-07-21



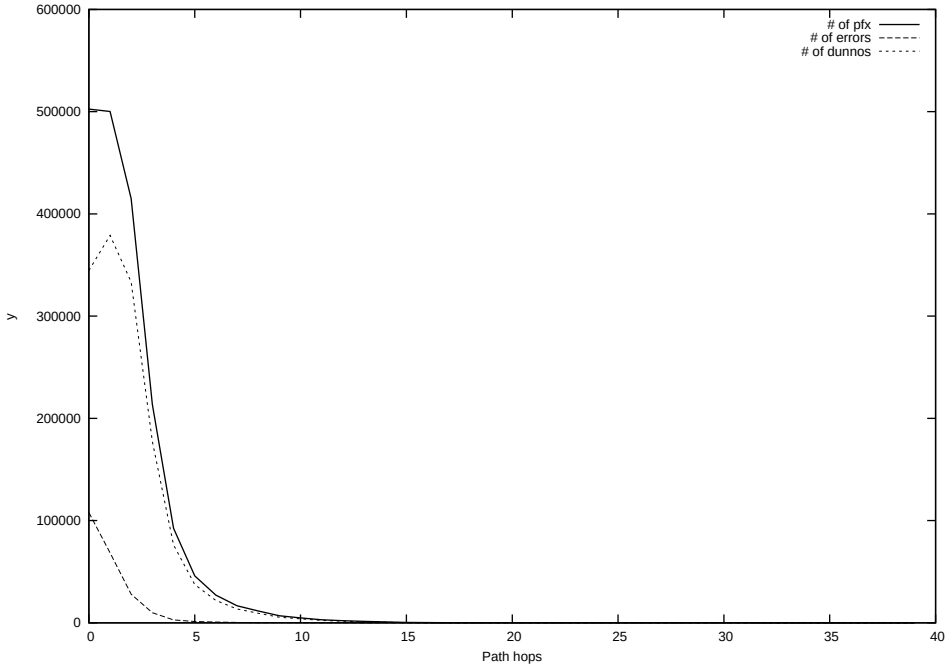
2014-07-22



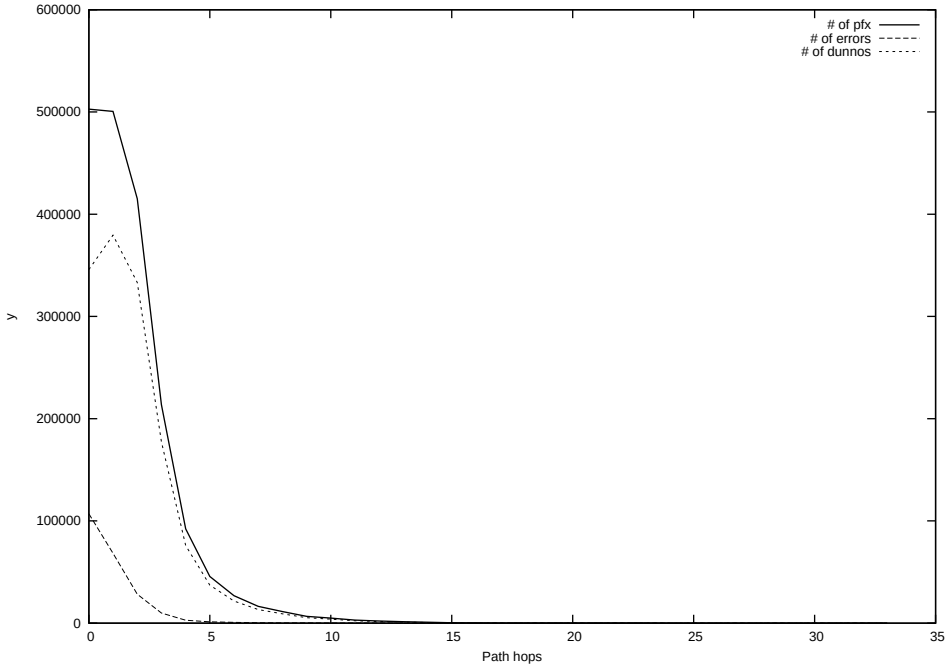
2014-07-23



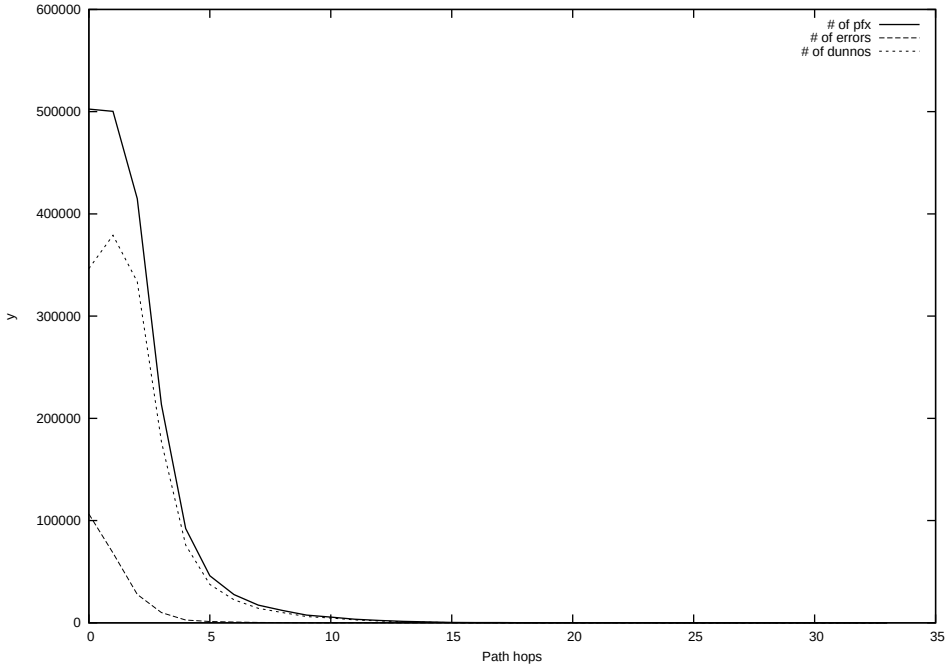
2014-07-24



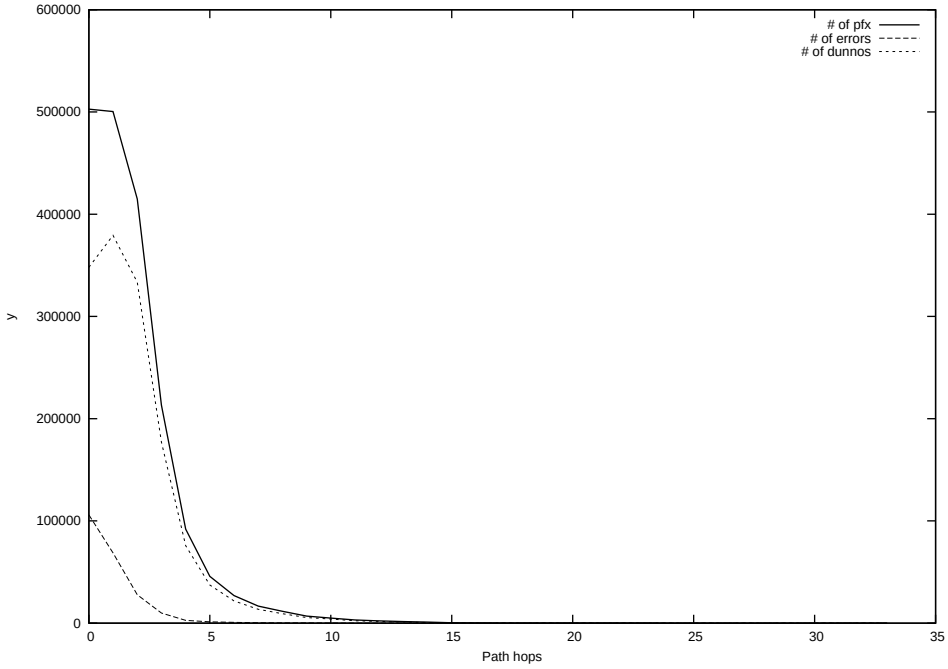
2014-07-25



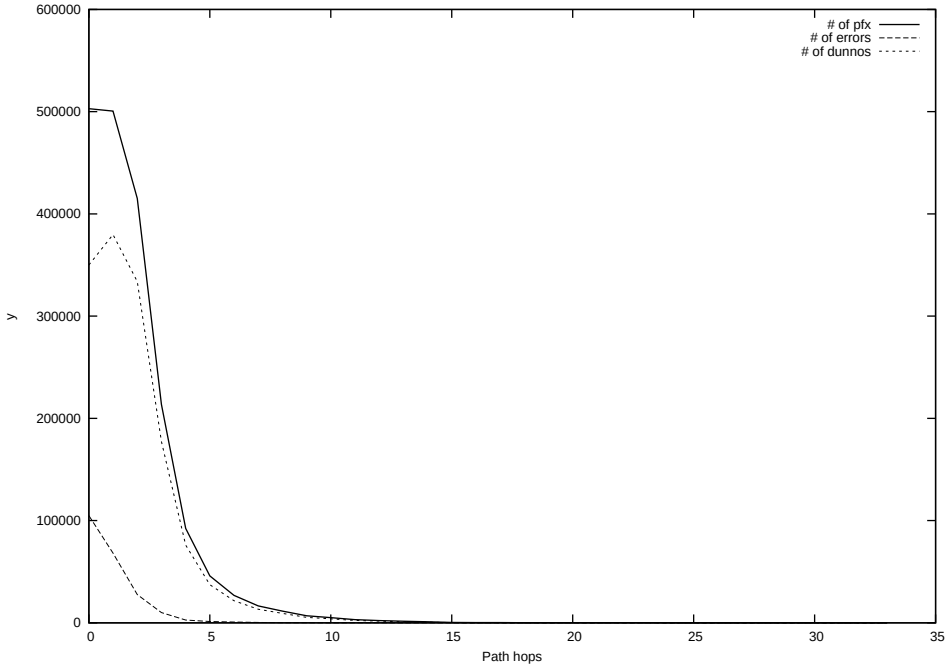
2014-07-26



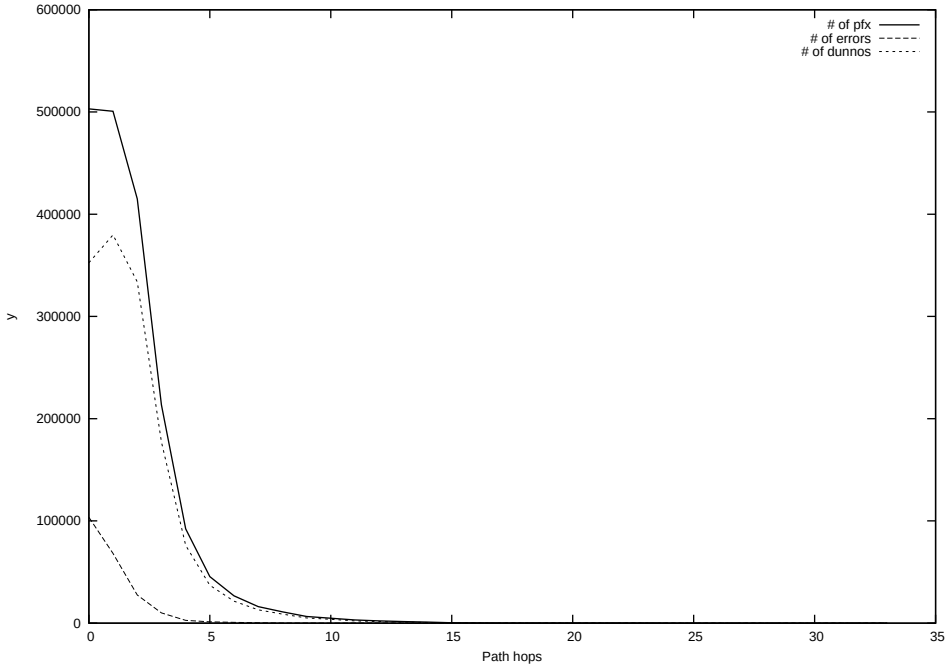
2014-07-27



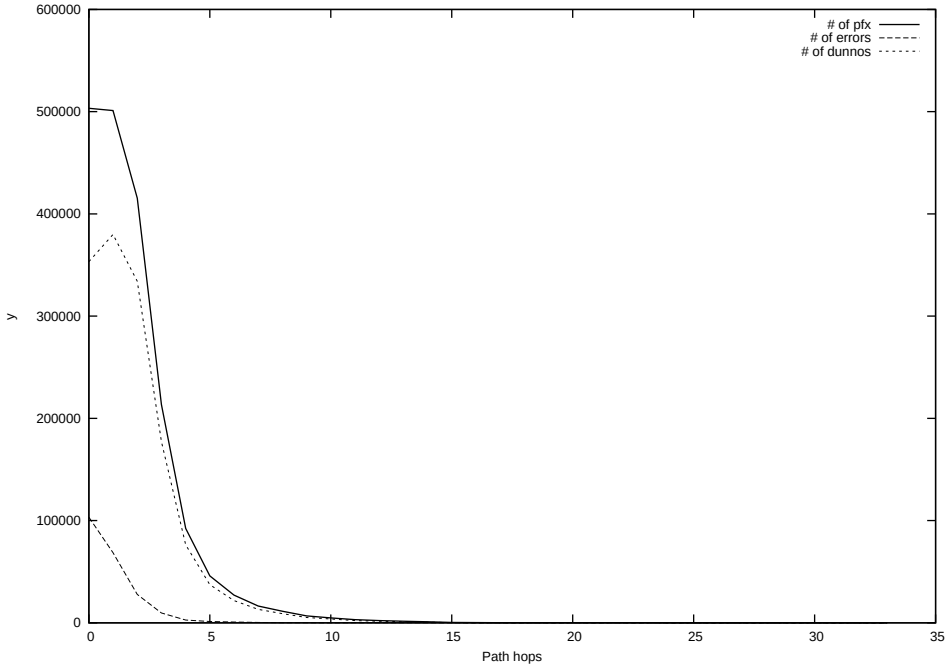
2014-07-28



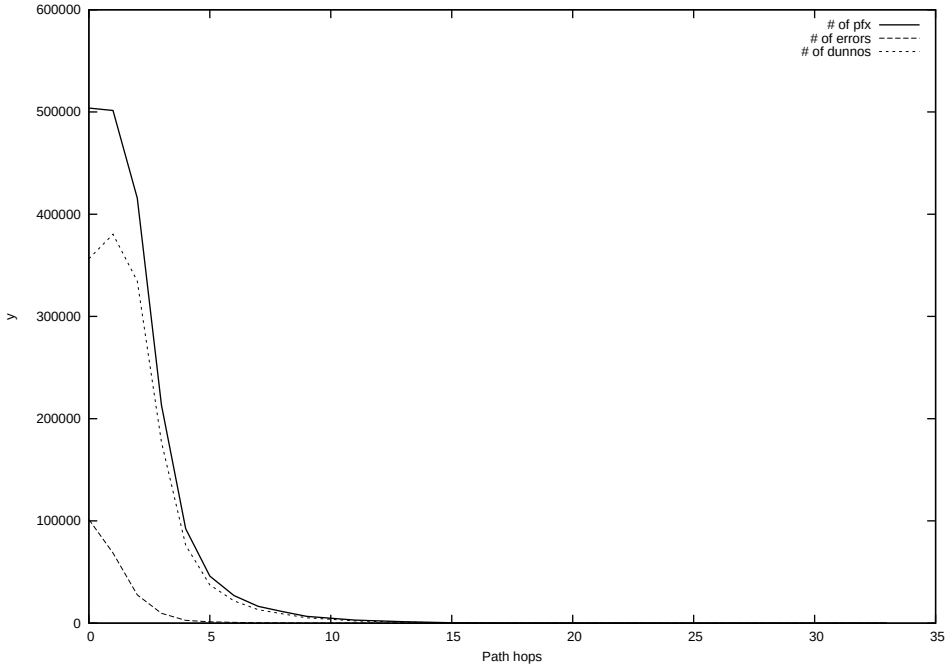
2014-07-29



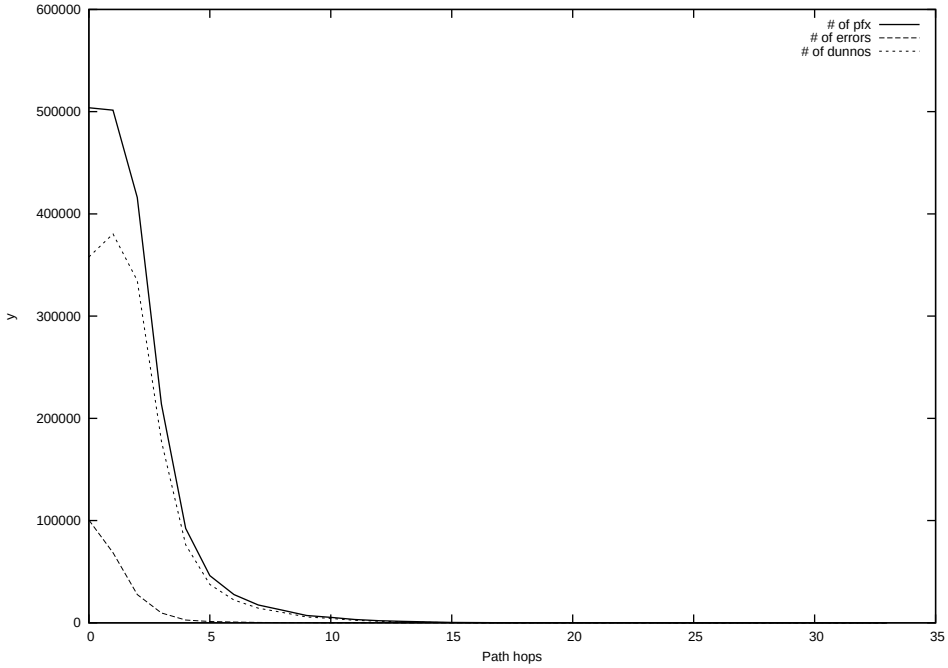
2014-07-30



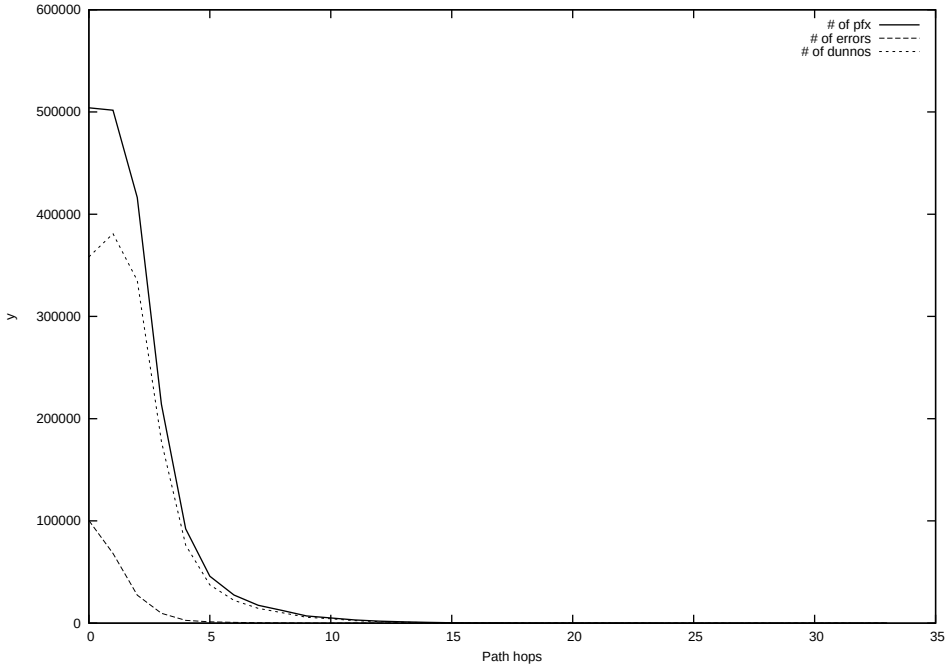
2014-07-31



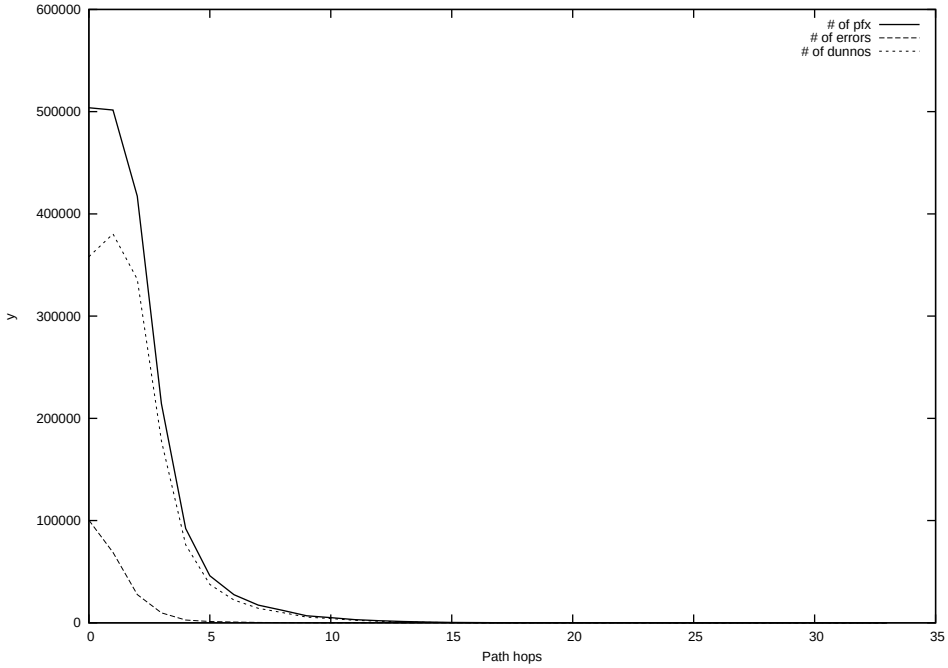
2014-08-01



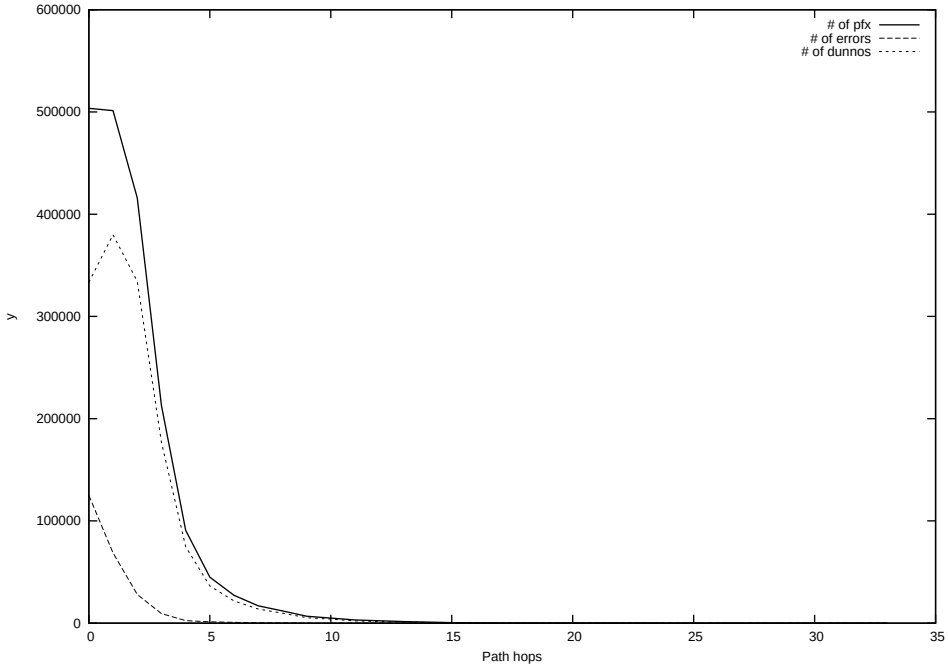
2014-08-02



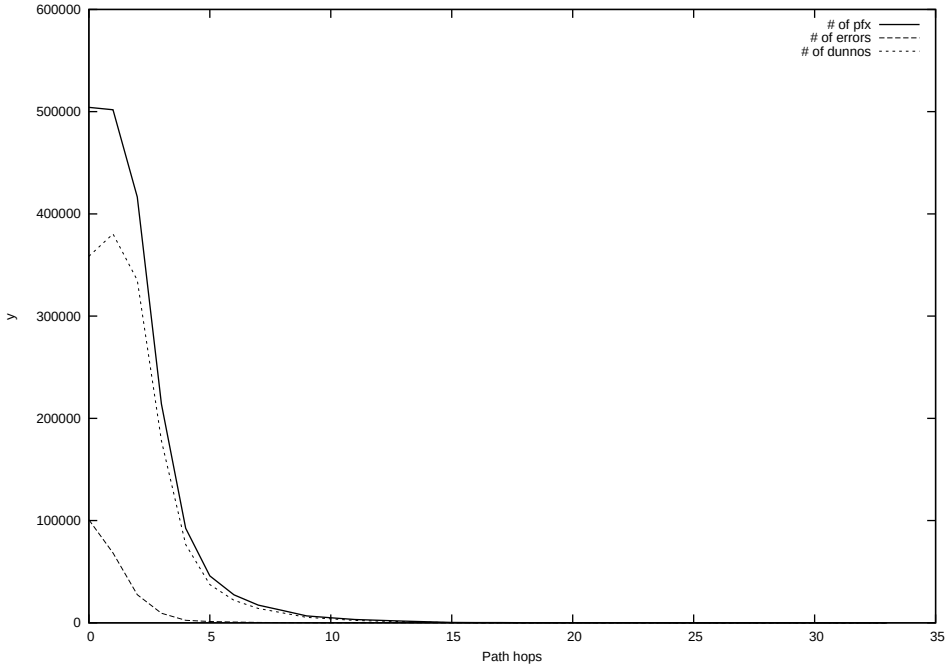
2014-08-03



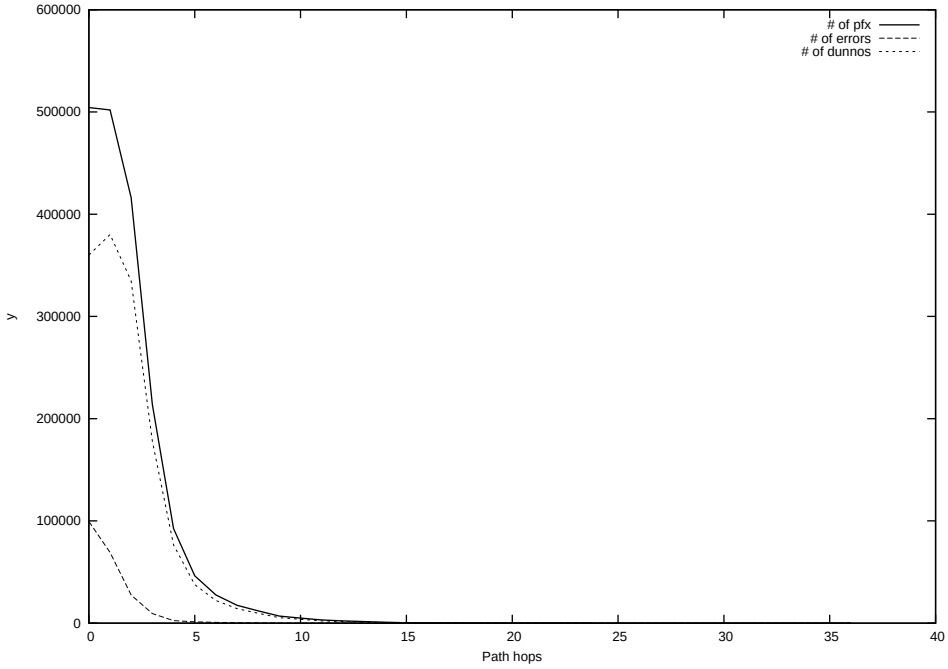
2014-08-04



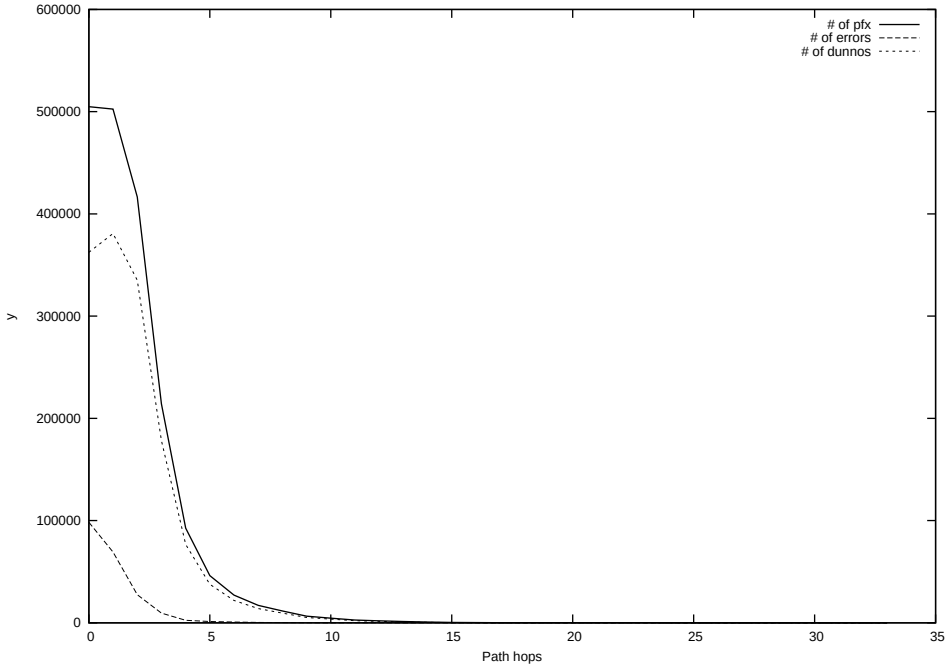
2014-08-05



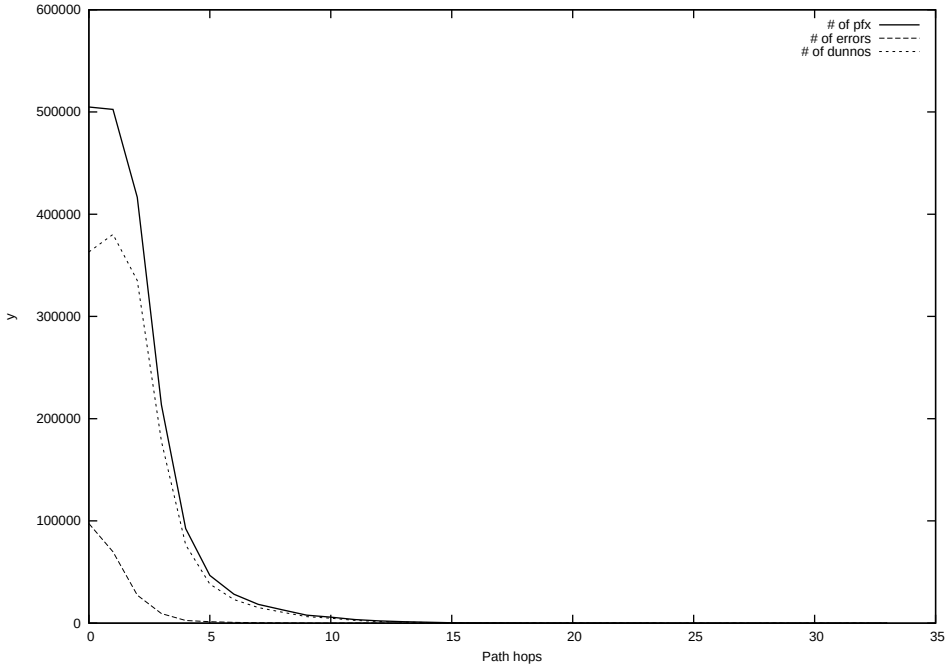
2014-08-06



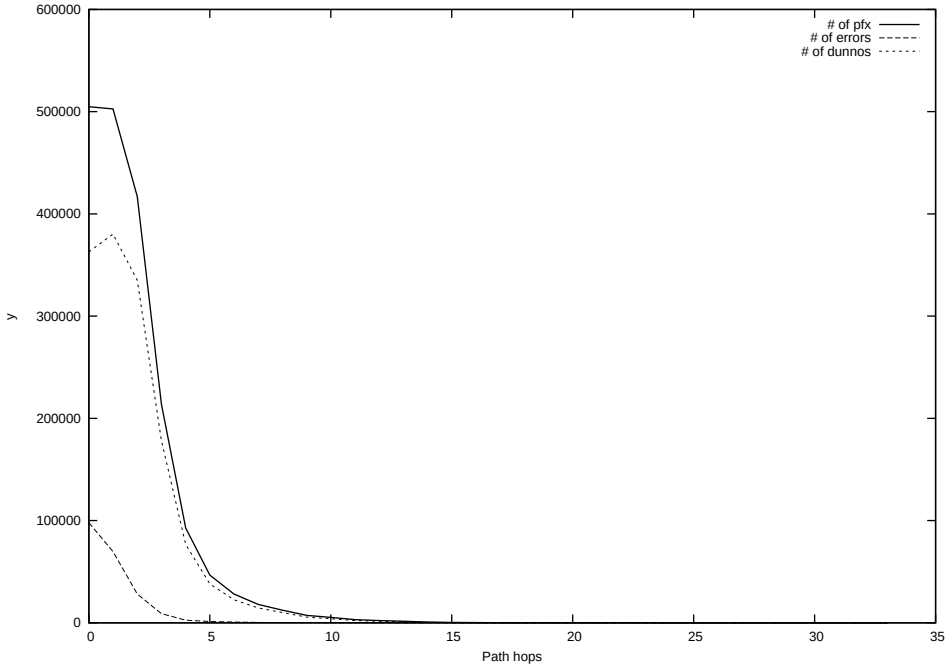
2014-08-07



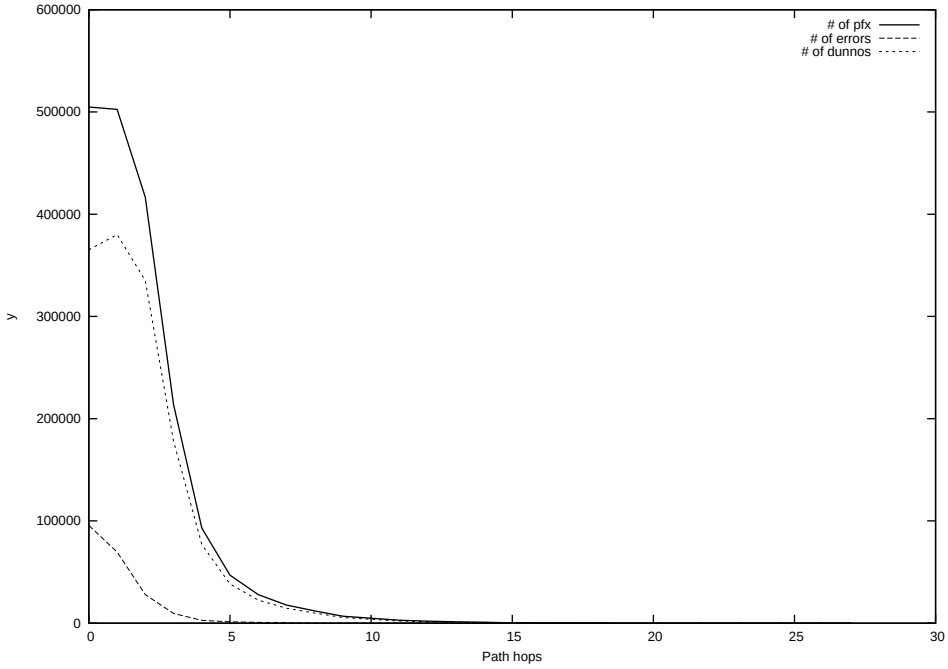
2014-08-08



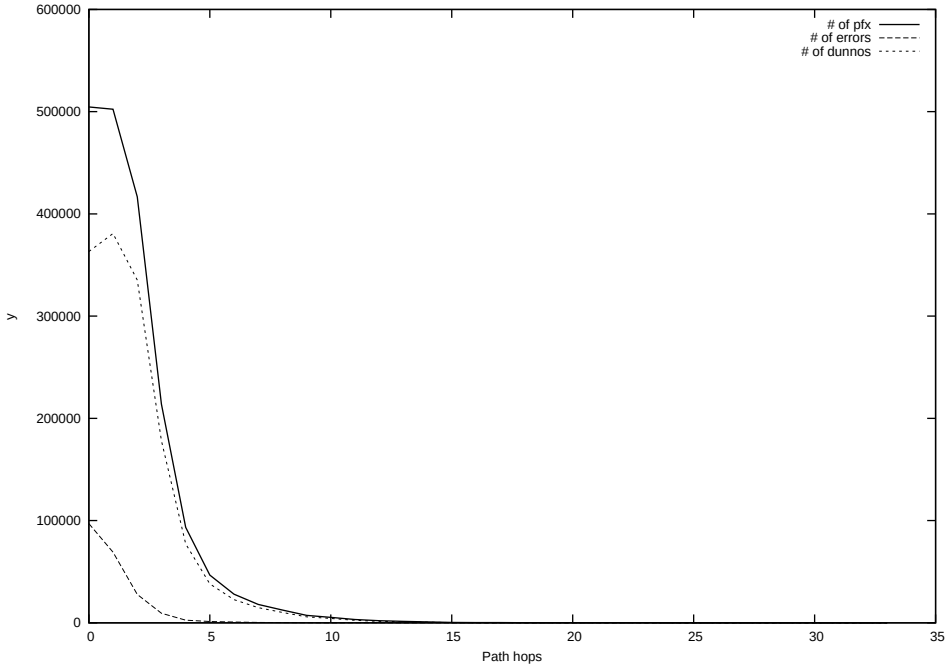
2014-08-09



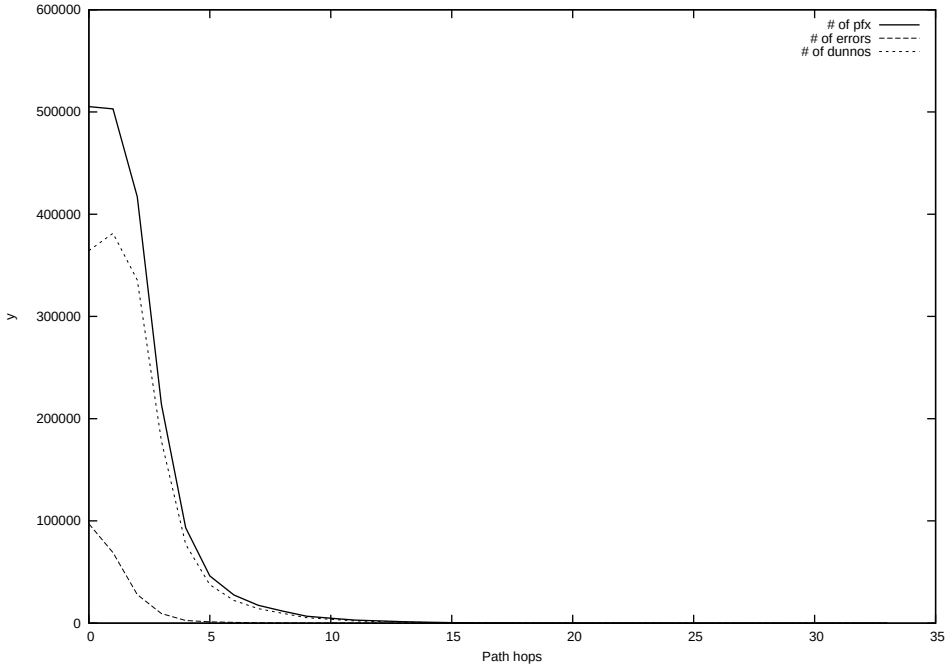
2014-08-10



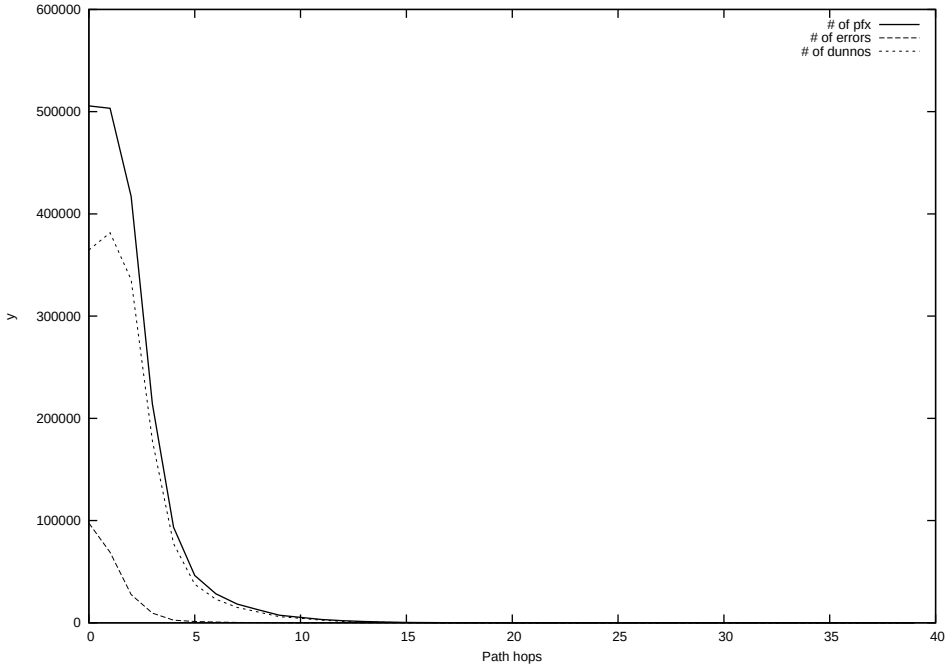
2014-08-11



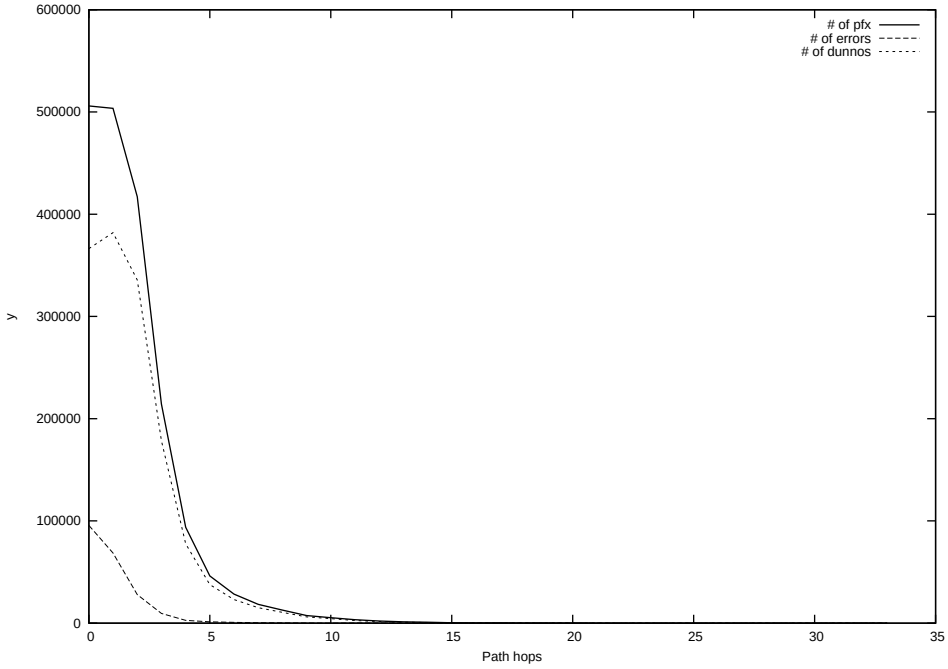
2014-08-12



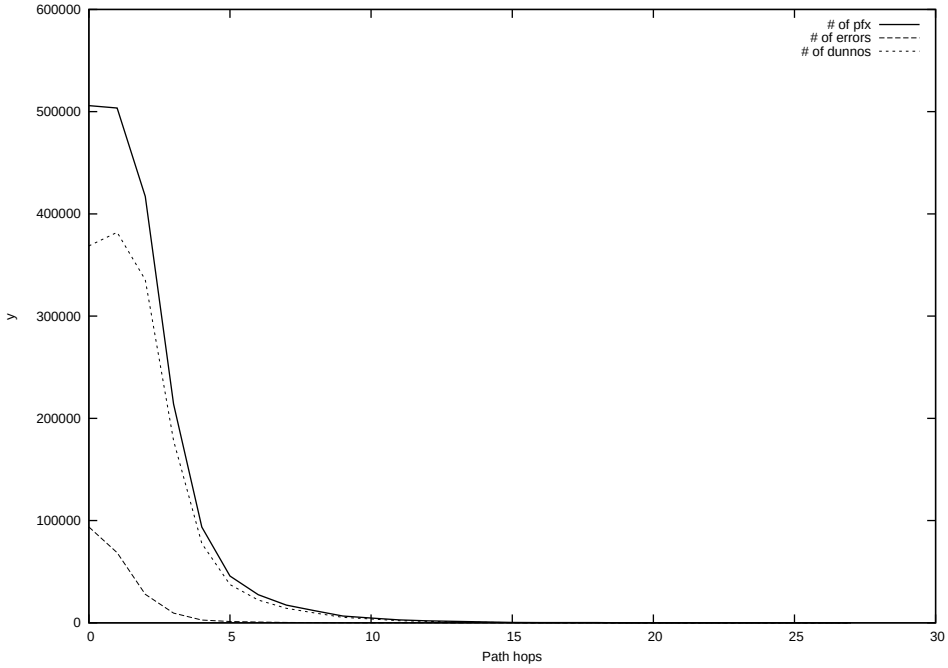
2014-08-13



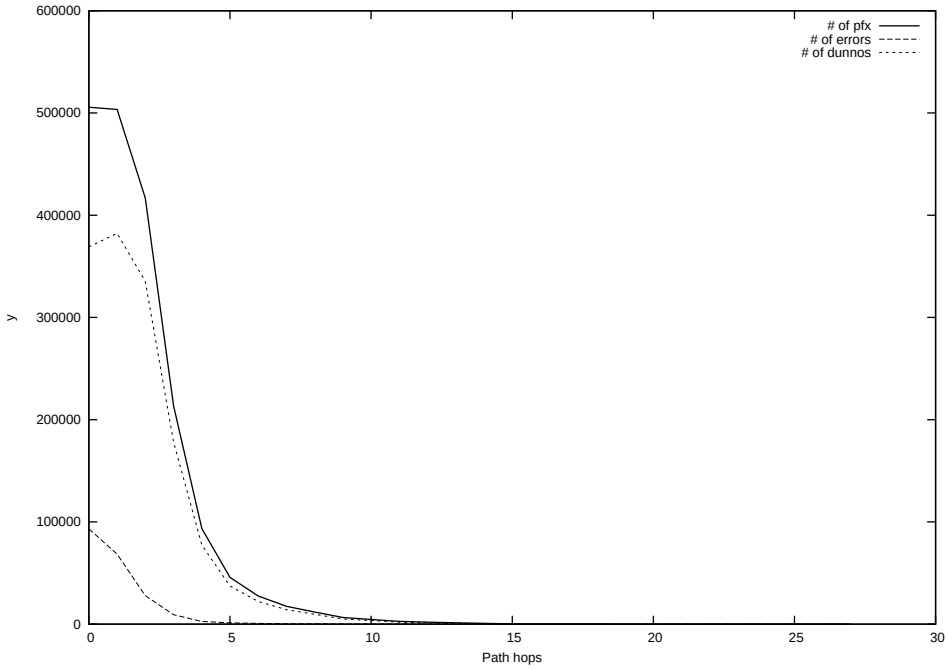
2014-08-14



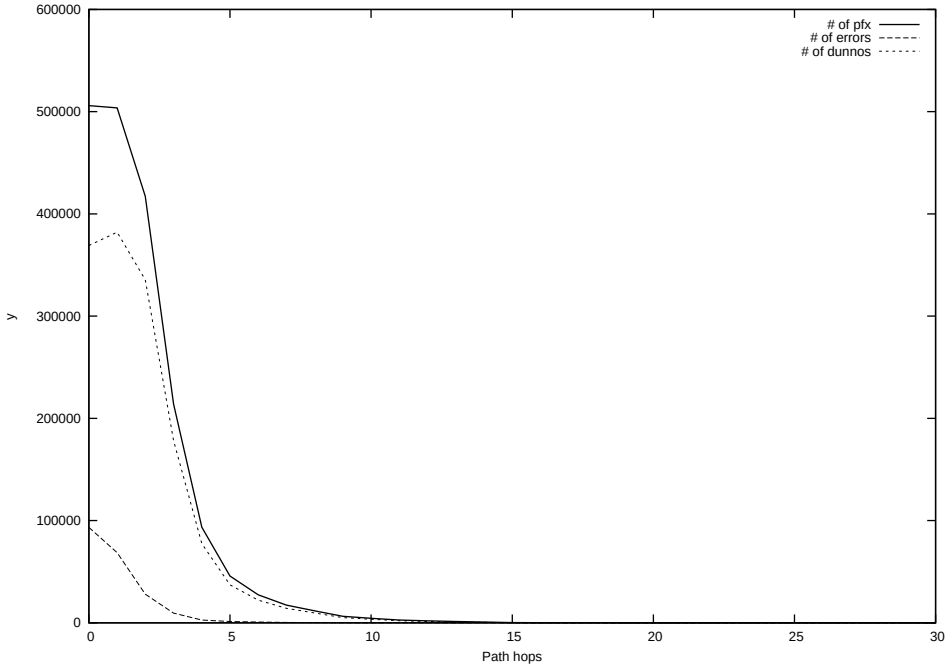
2014-08-15



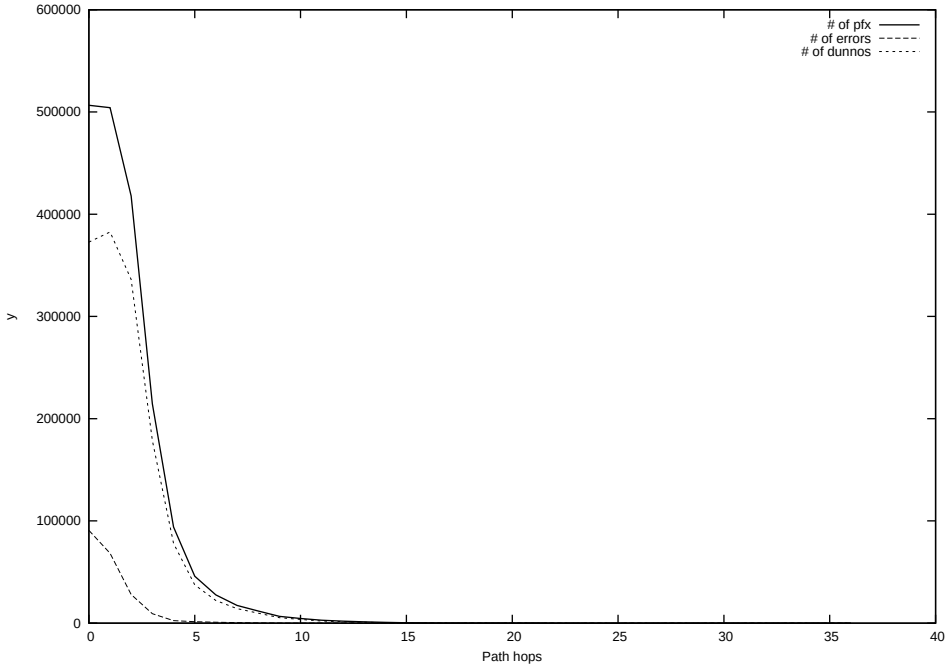
2014-08-16



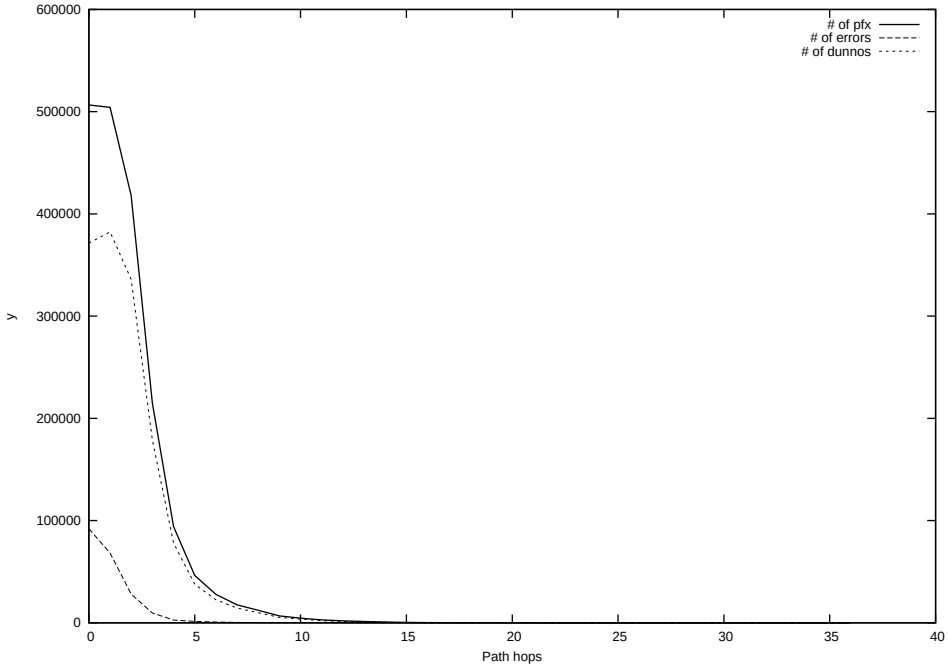
2014-08-17



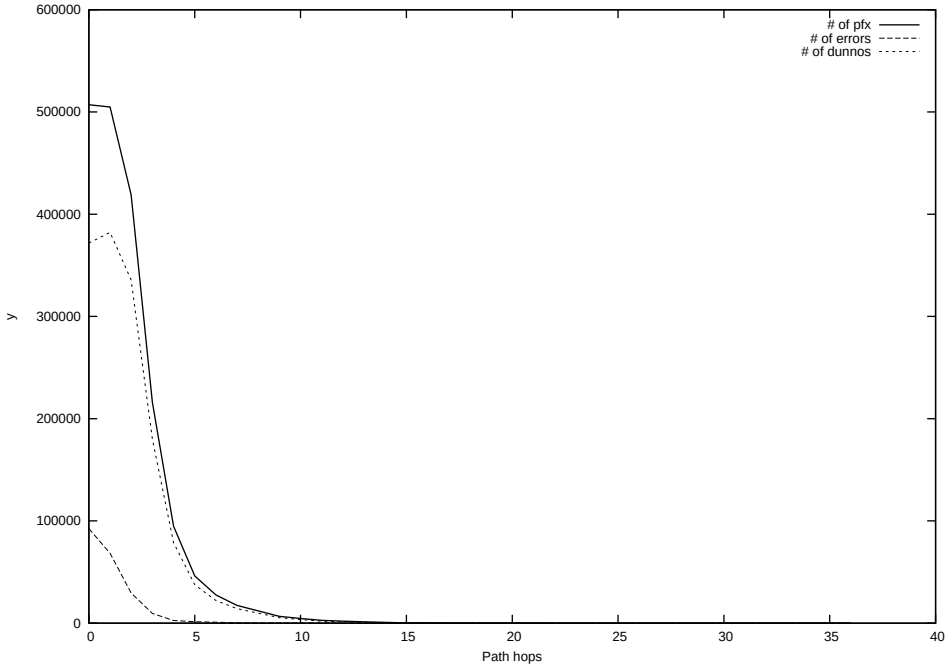
2014-08-18



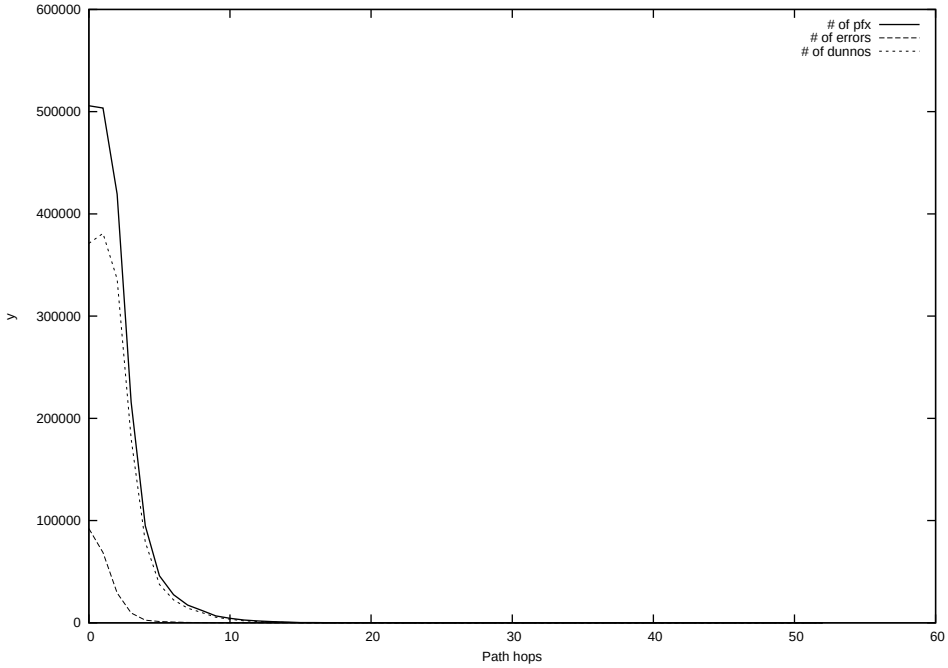
2014-08-19



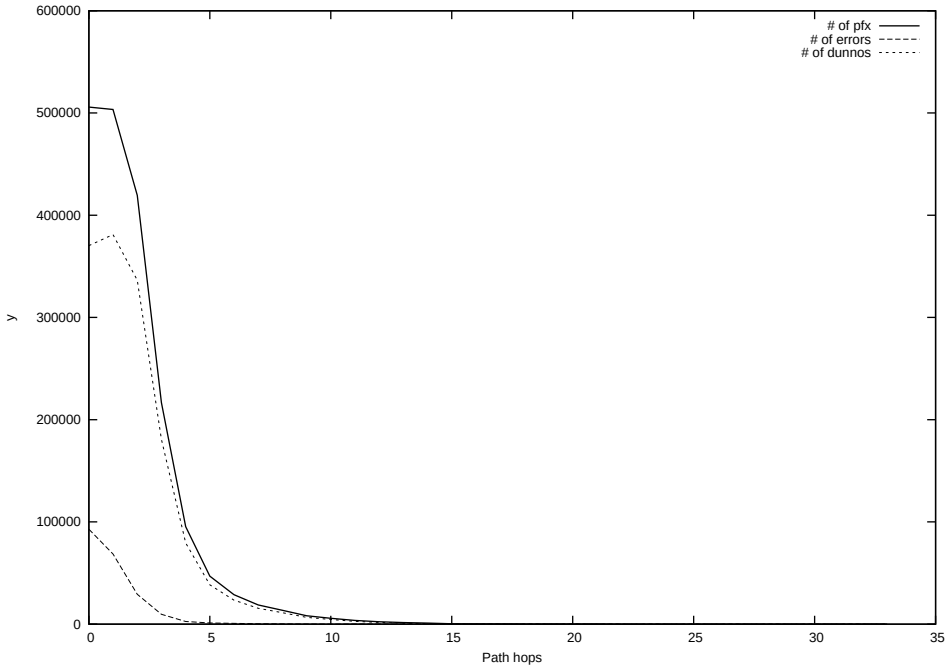
2014-08-20



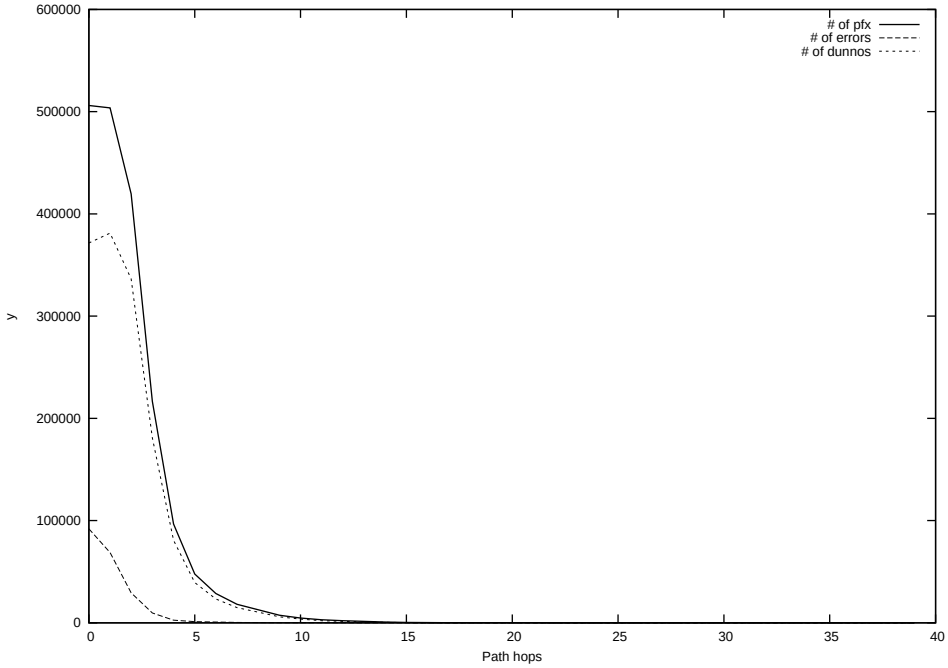
2014-08-21



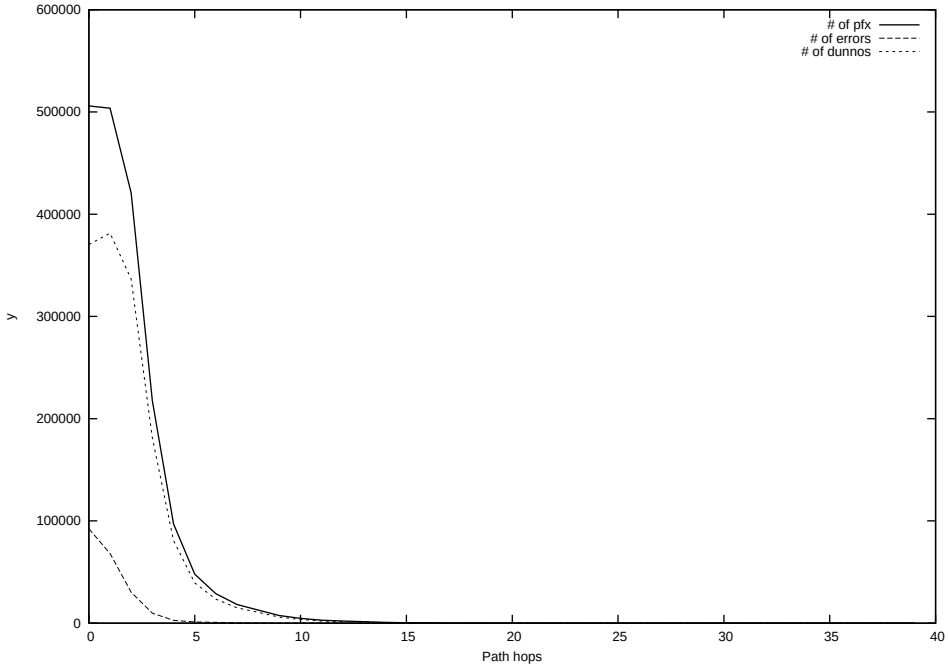
2014-08-22



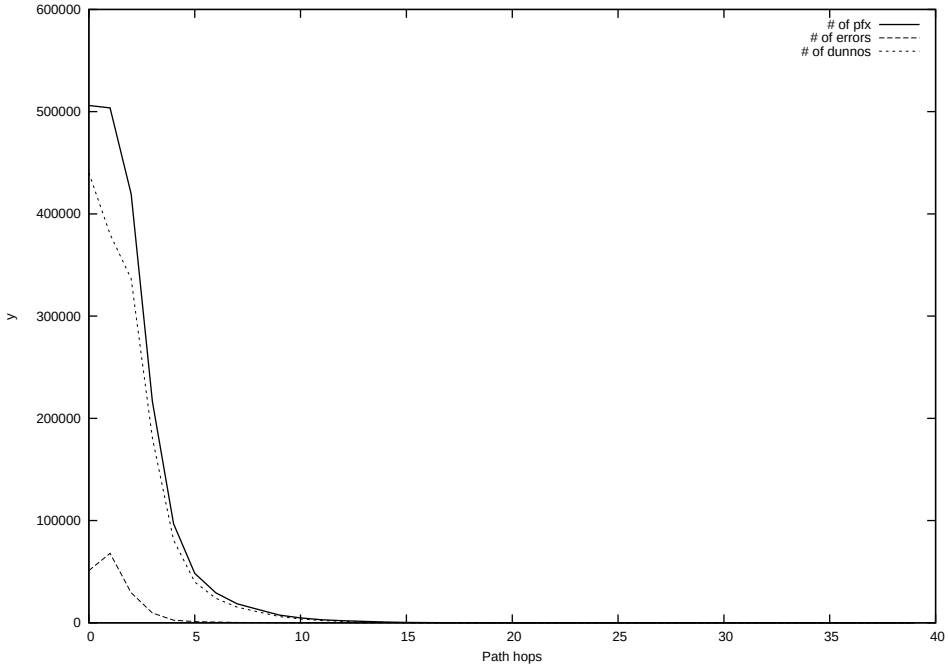
2014-08-23



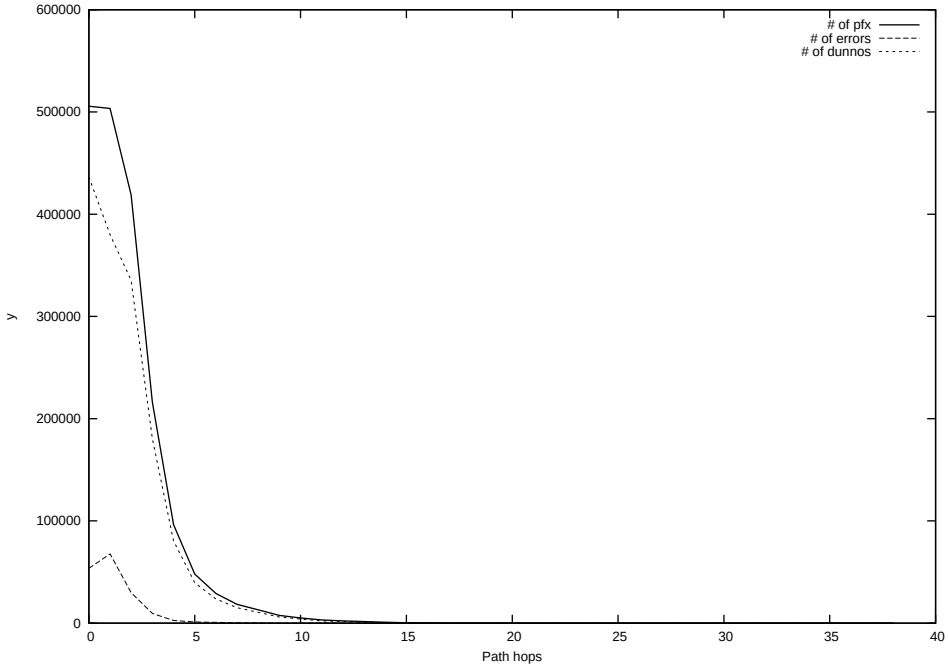
2014-08-24



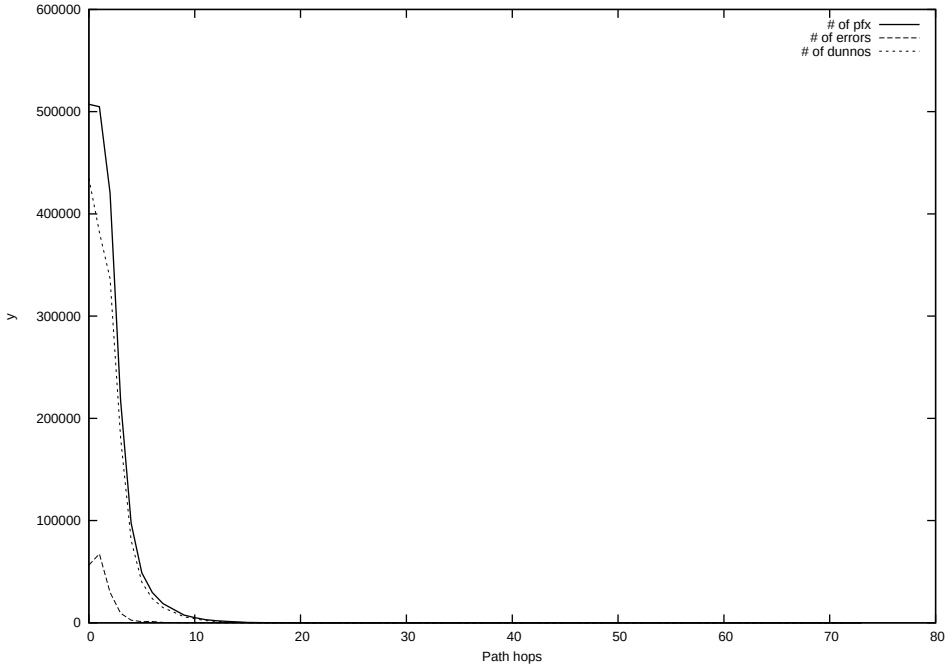
2014-08-25



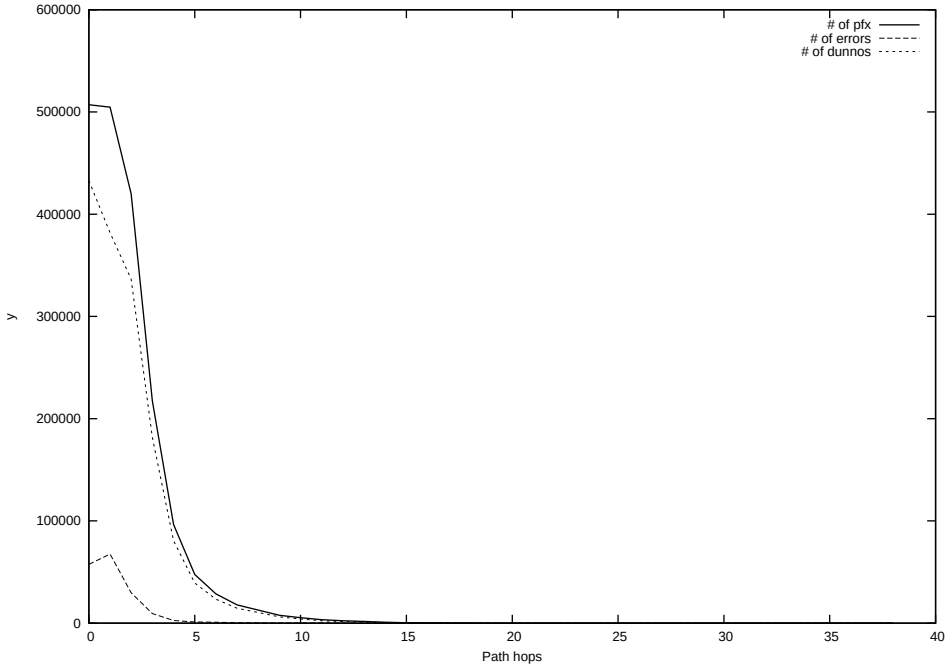
2014-08-26



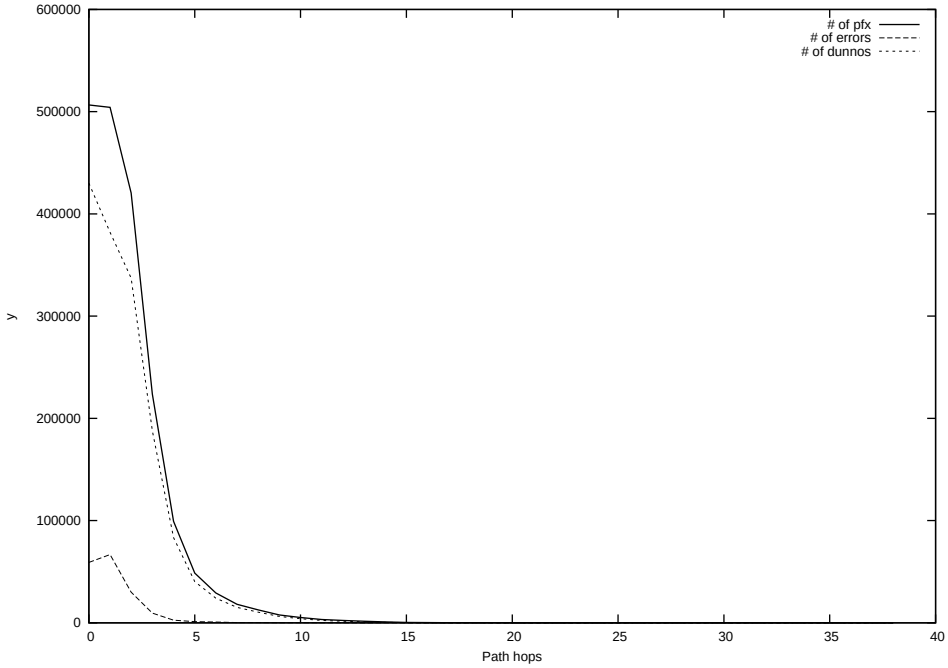
2014-08-27



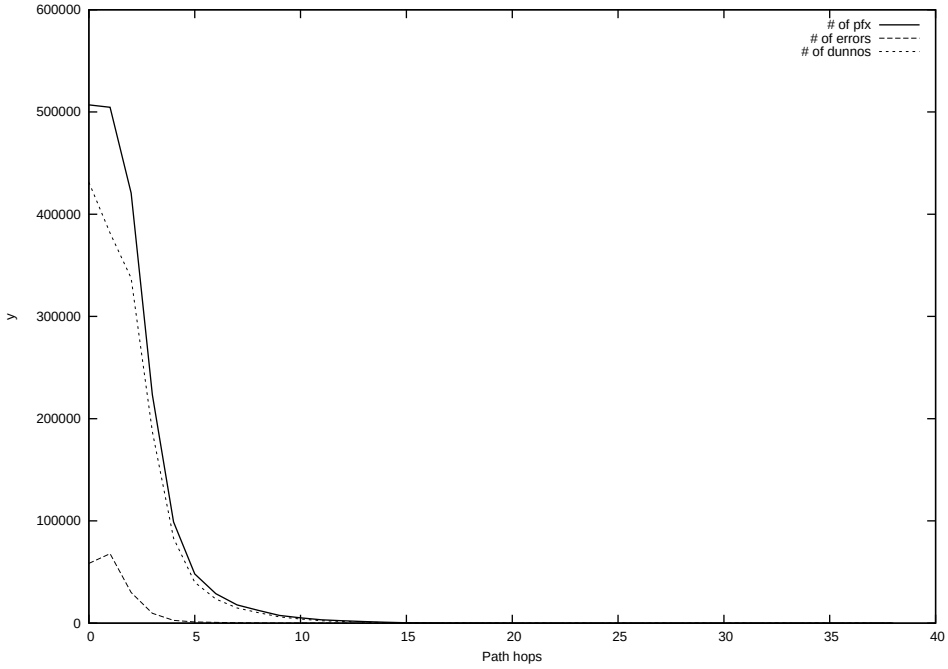
2014-08-28



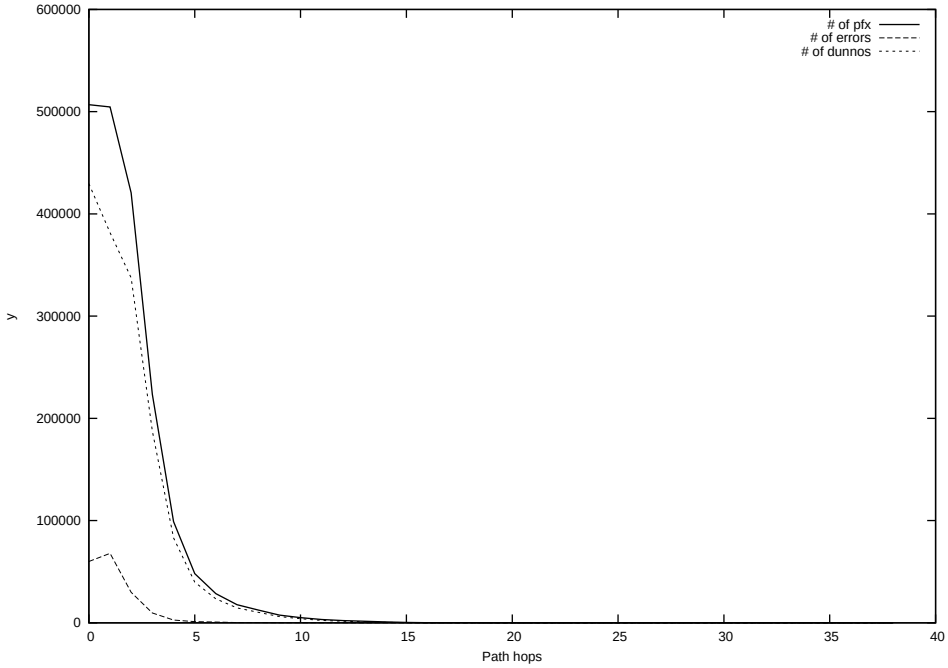
2014-08-29



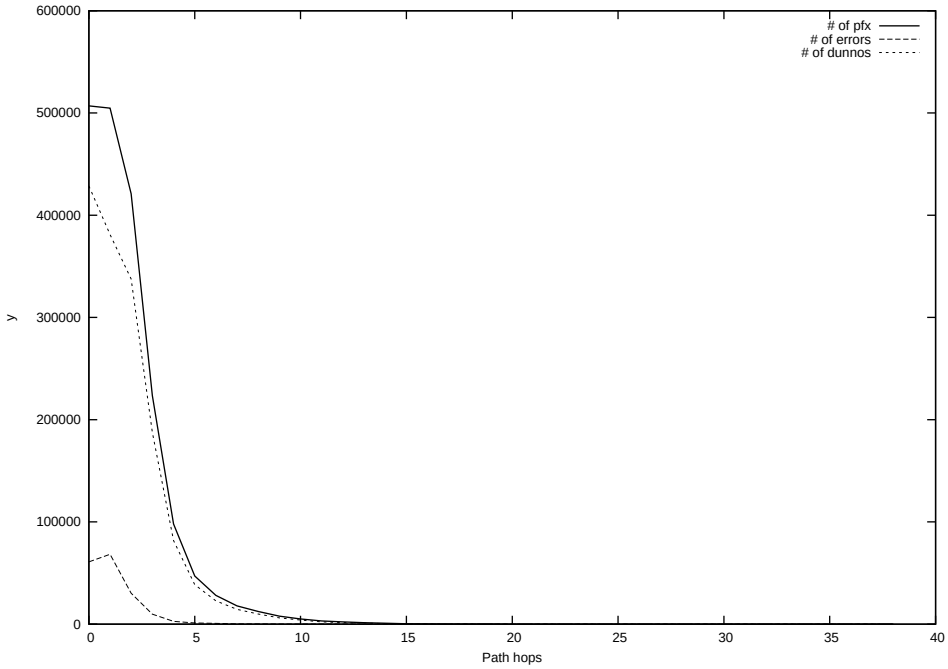
2014-08-30



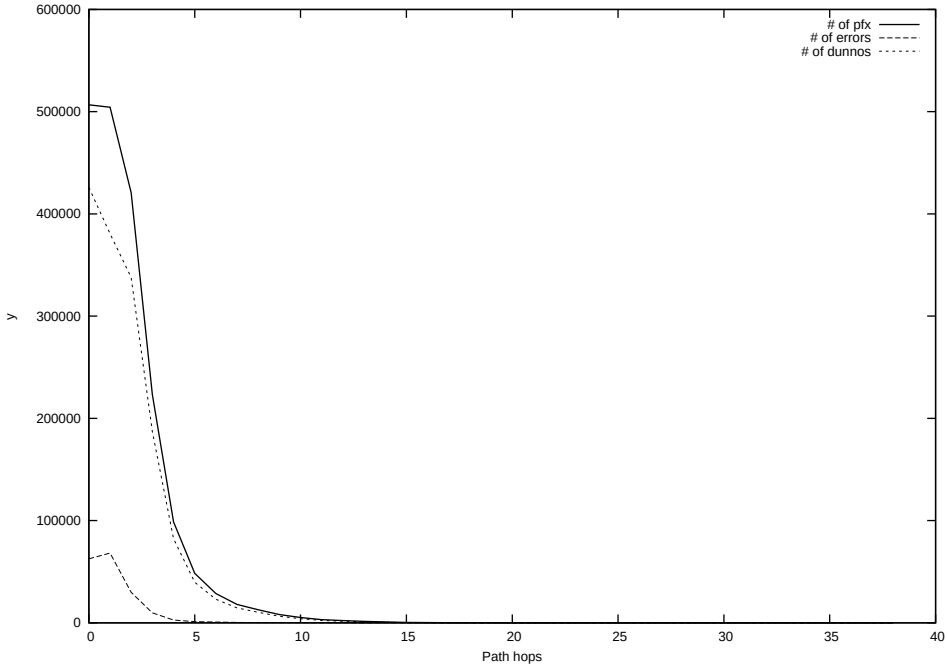
2014-08-31



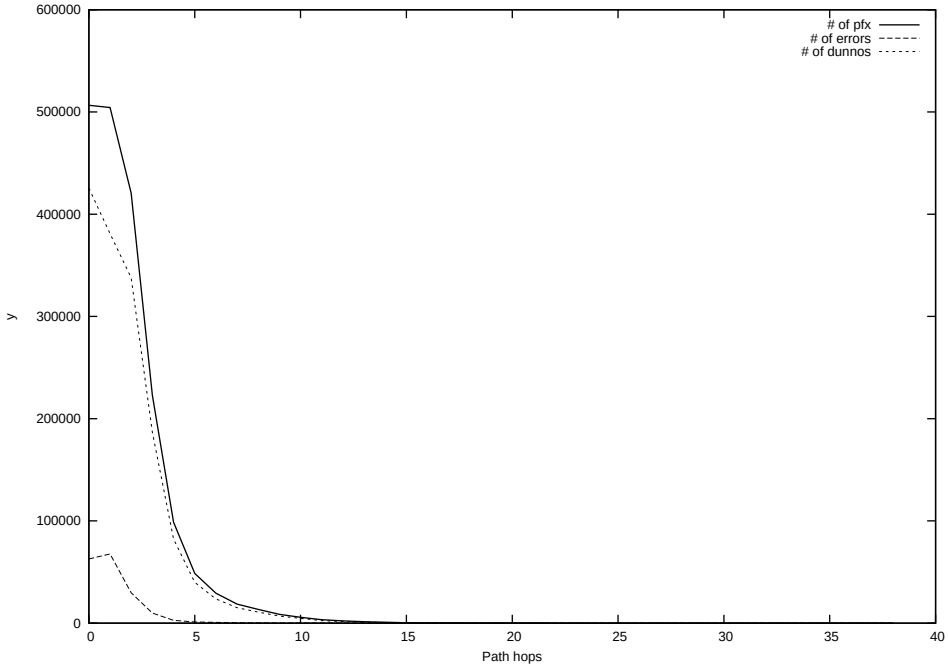
2014-09-01



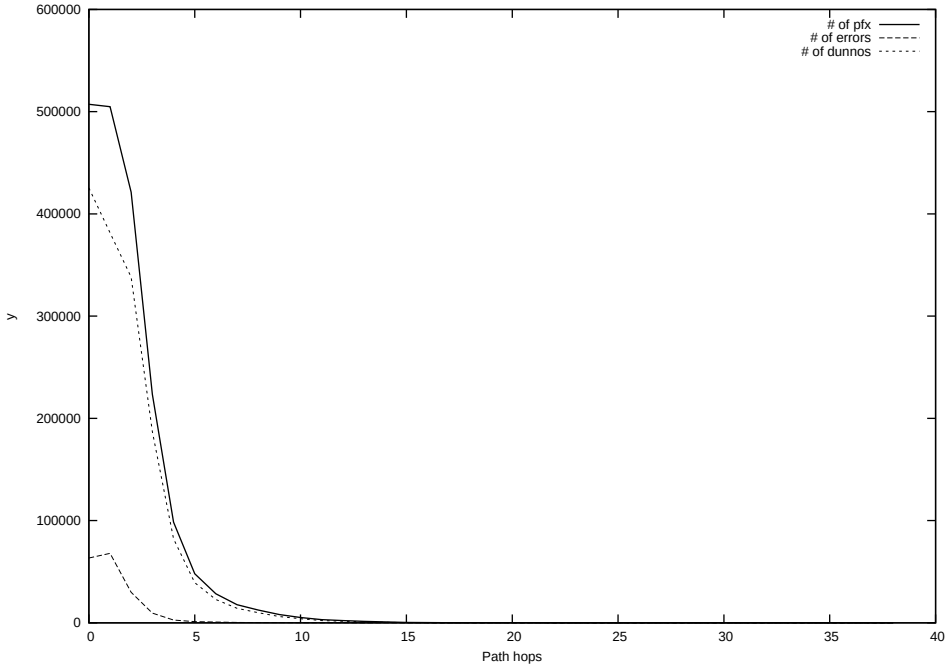
2014-09-02



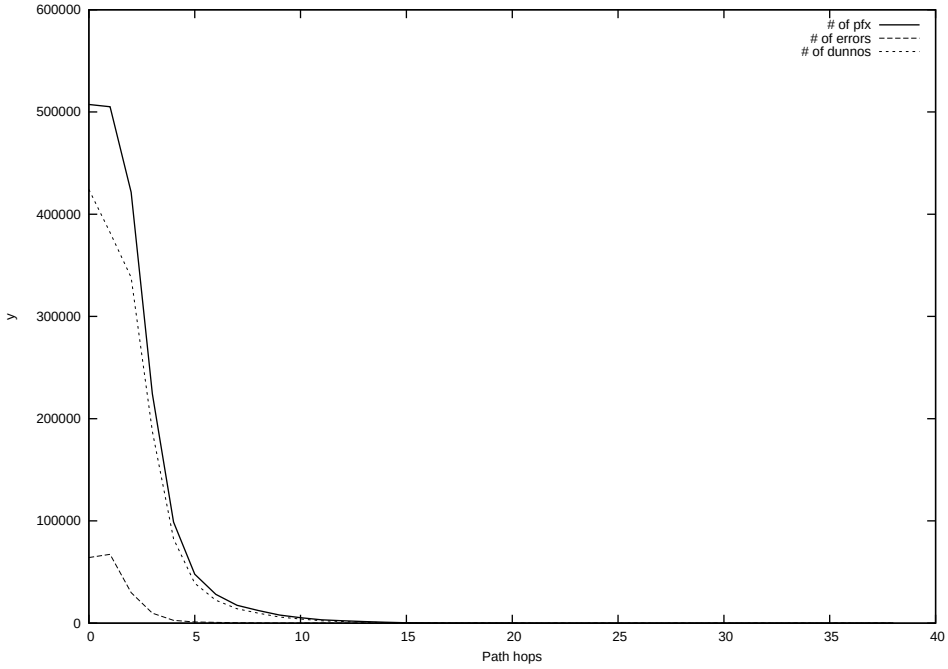
2014-09-03



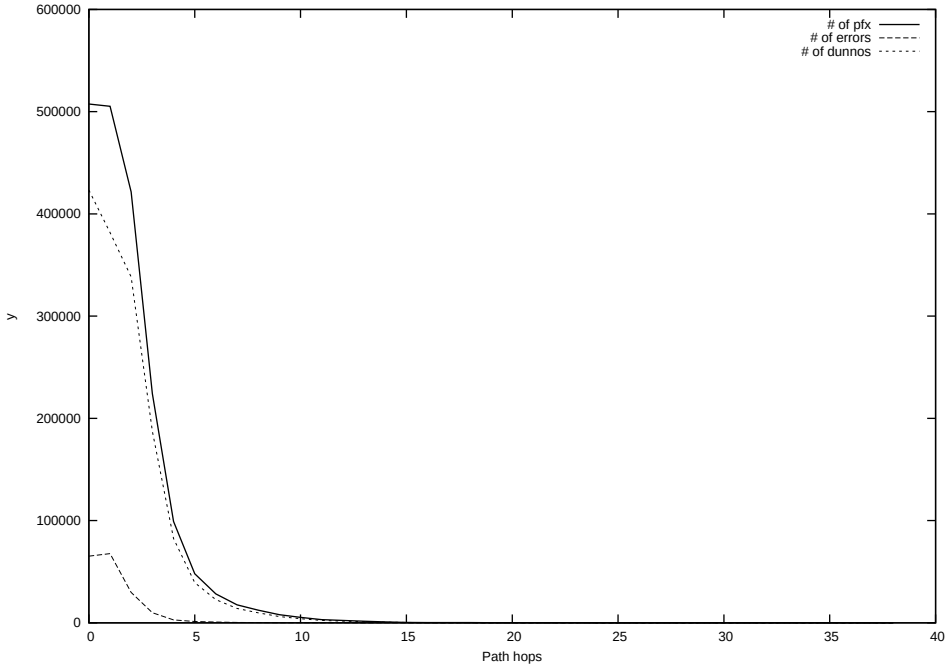
2014-09-04



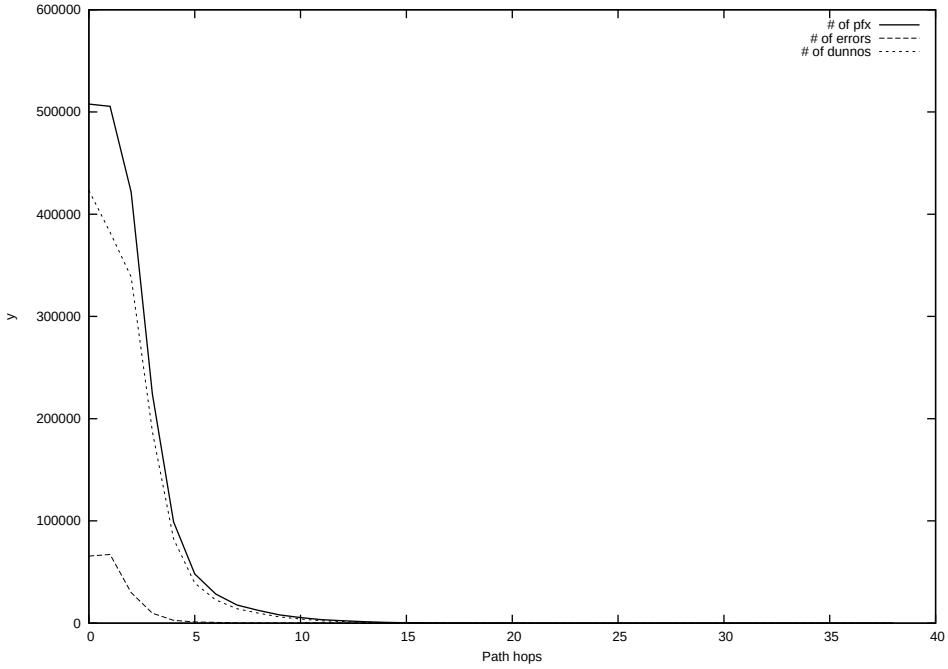
2014-09-05



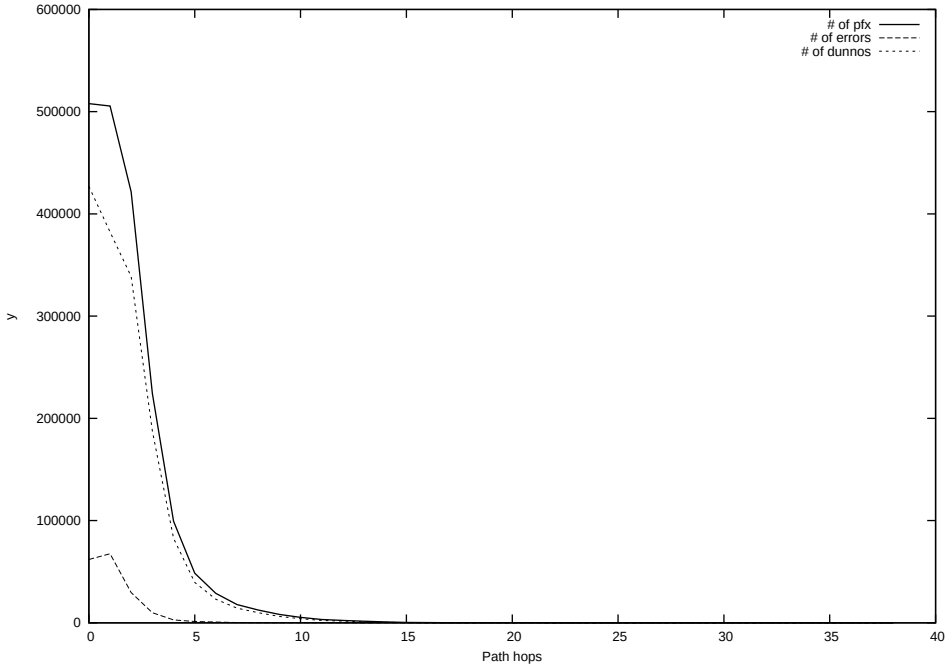
2014-09-06



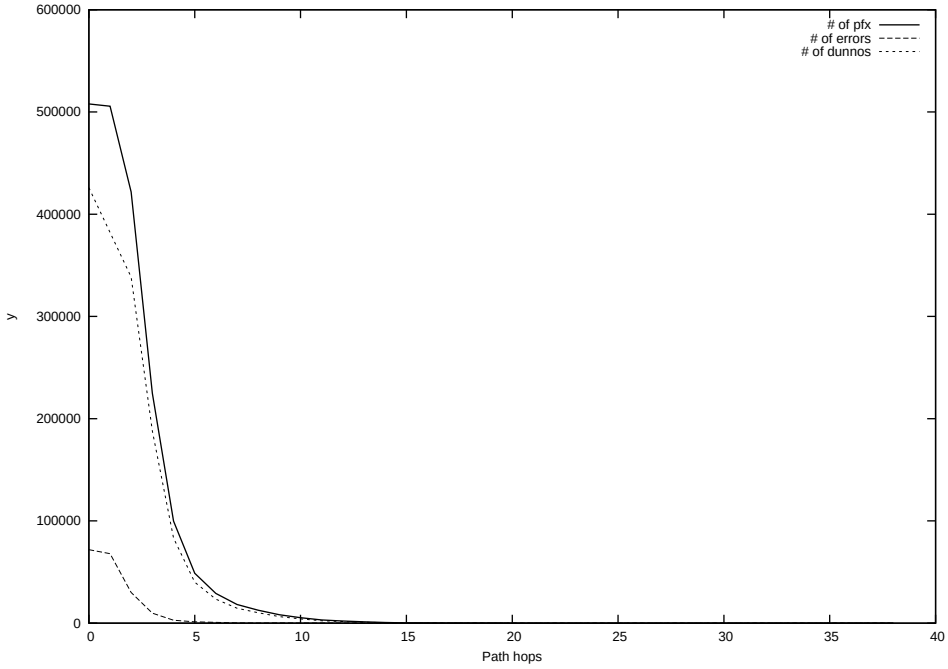
2014-09-07



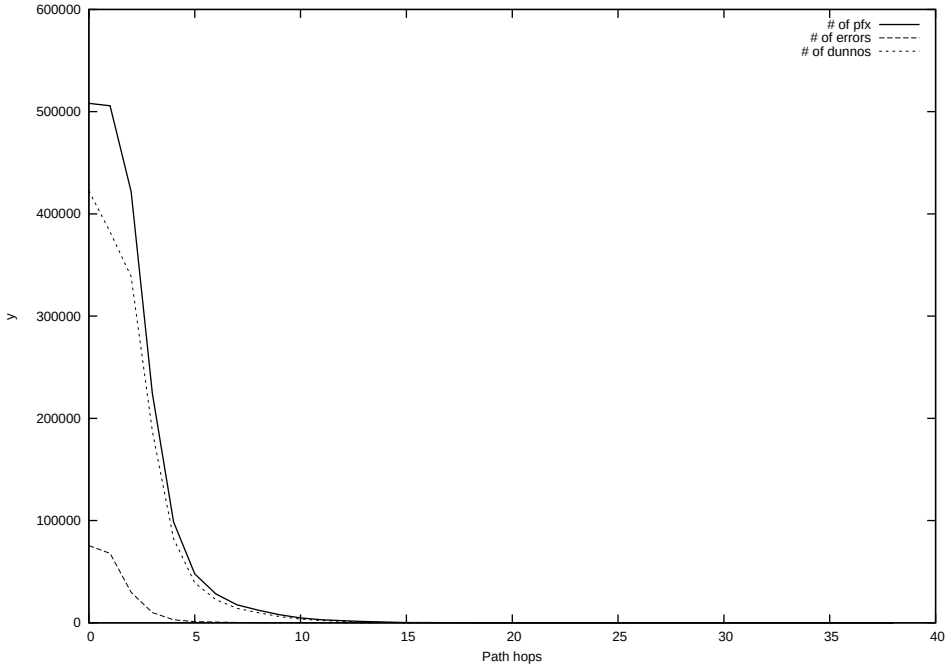
2014-09-08



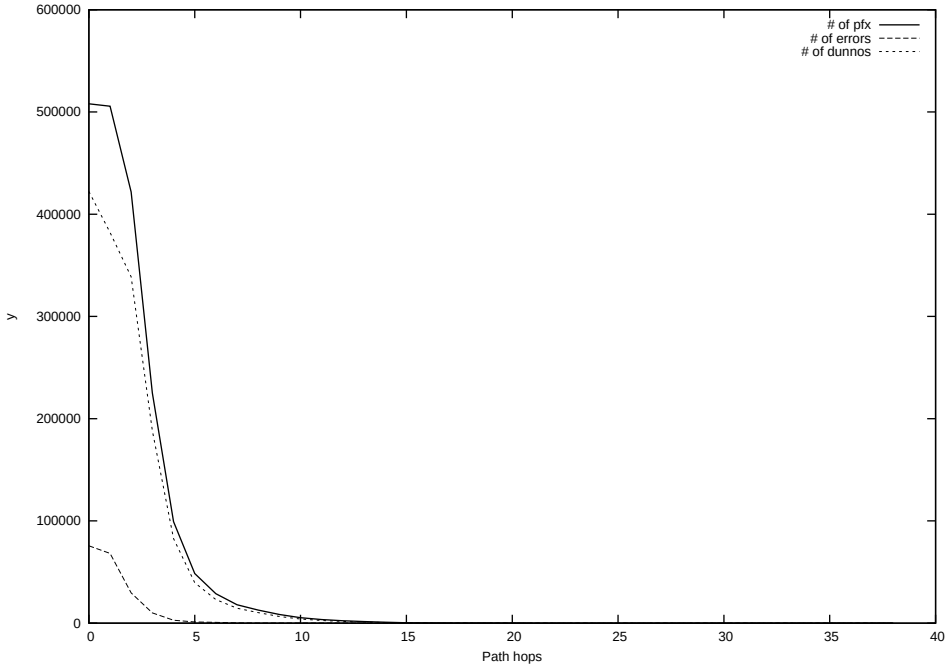
2014-09-09



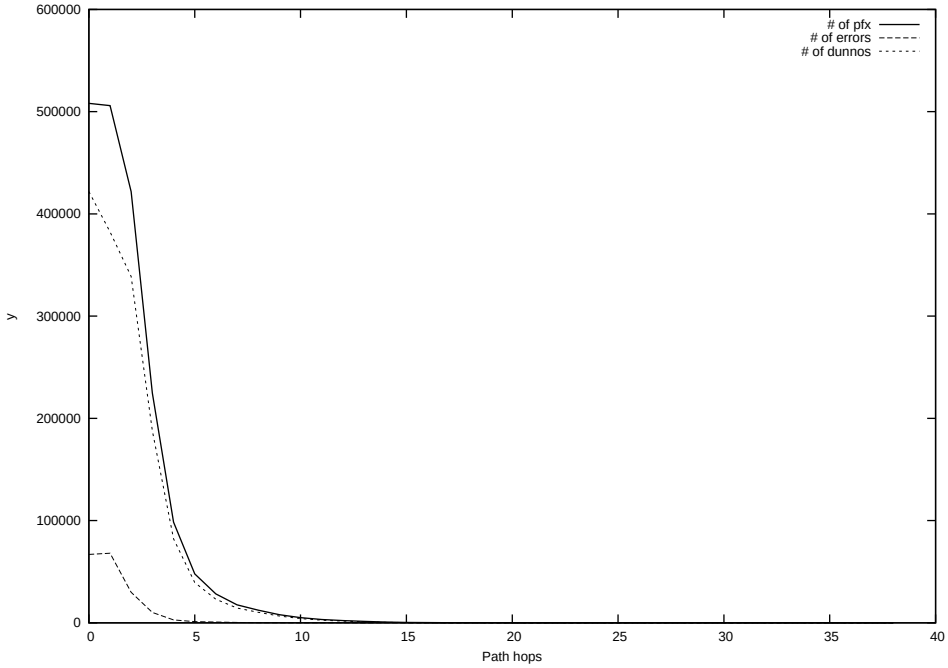
2014-09-10



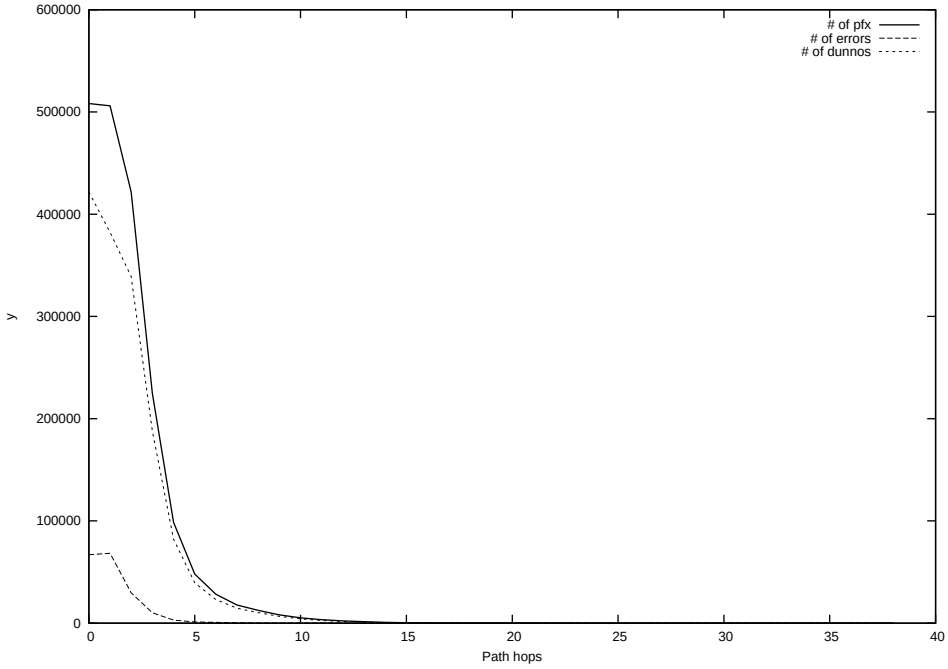
2014-09-11



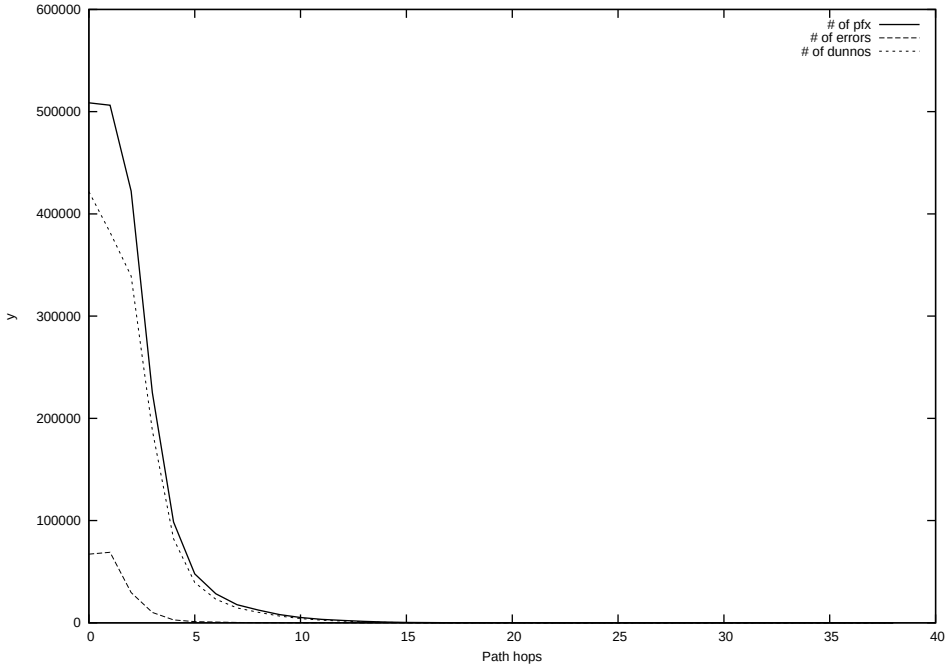
2014-09-12



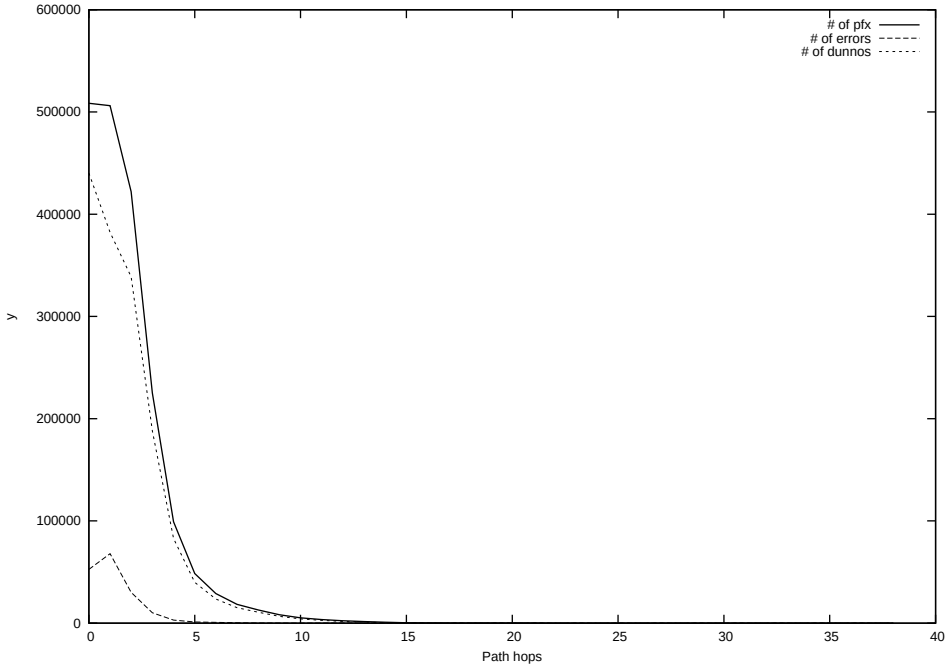
2014-09-13



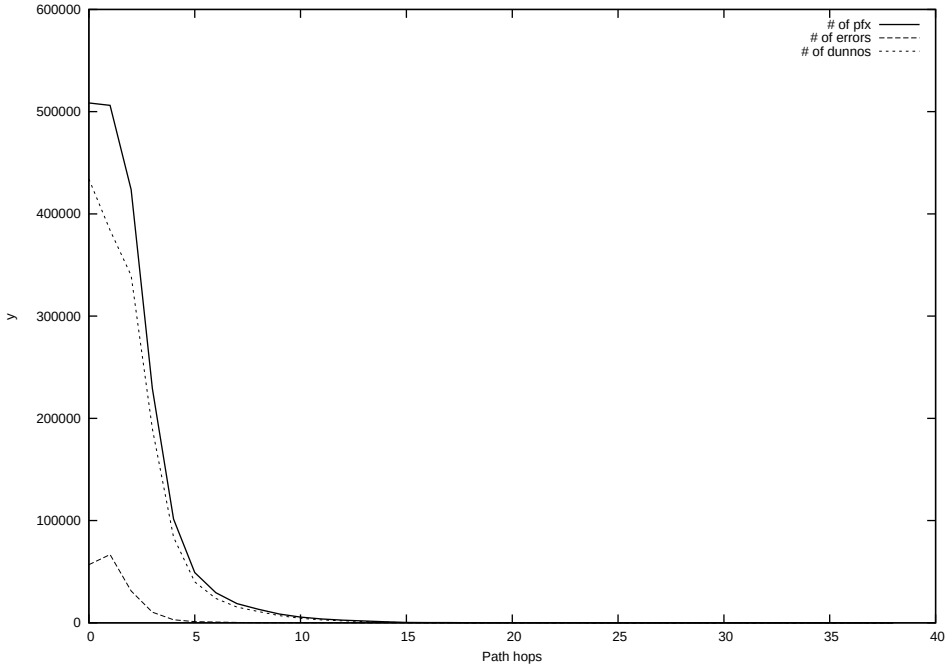
2014-09-14



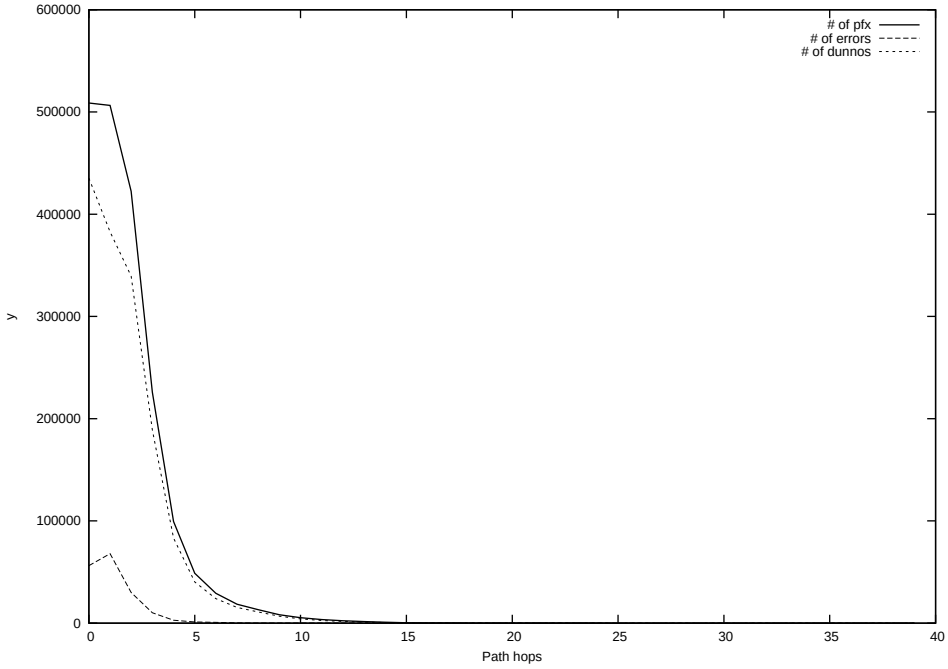
2014-09-15



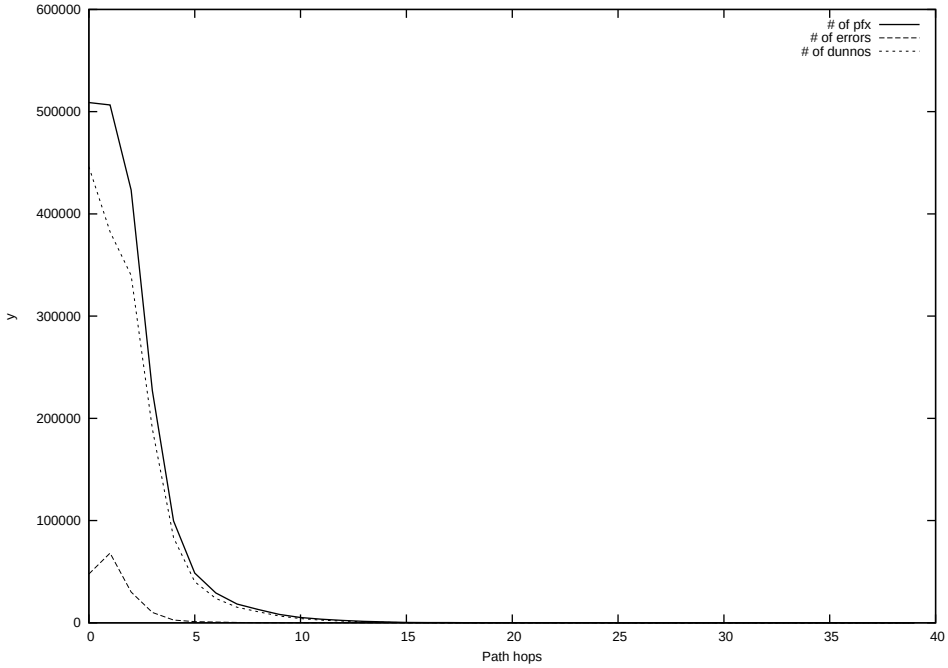
2014-09-16



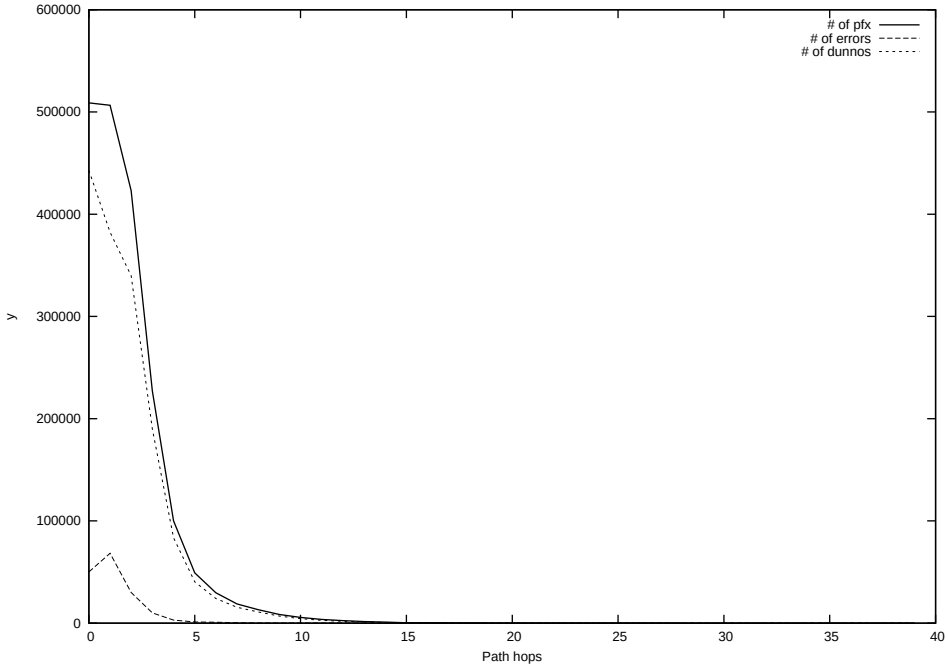
2014-09-17



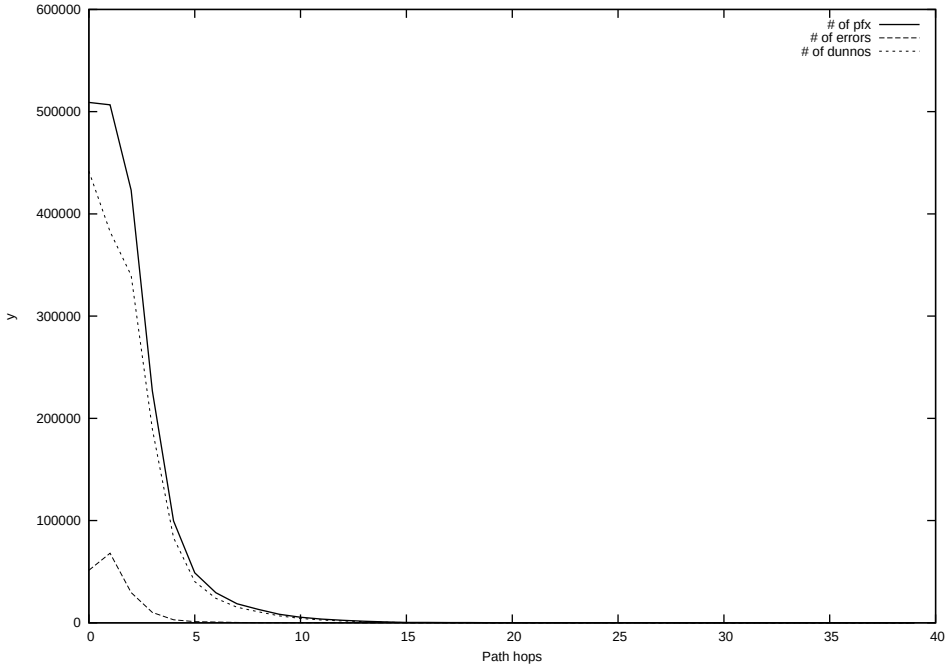
2014-09-18



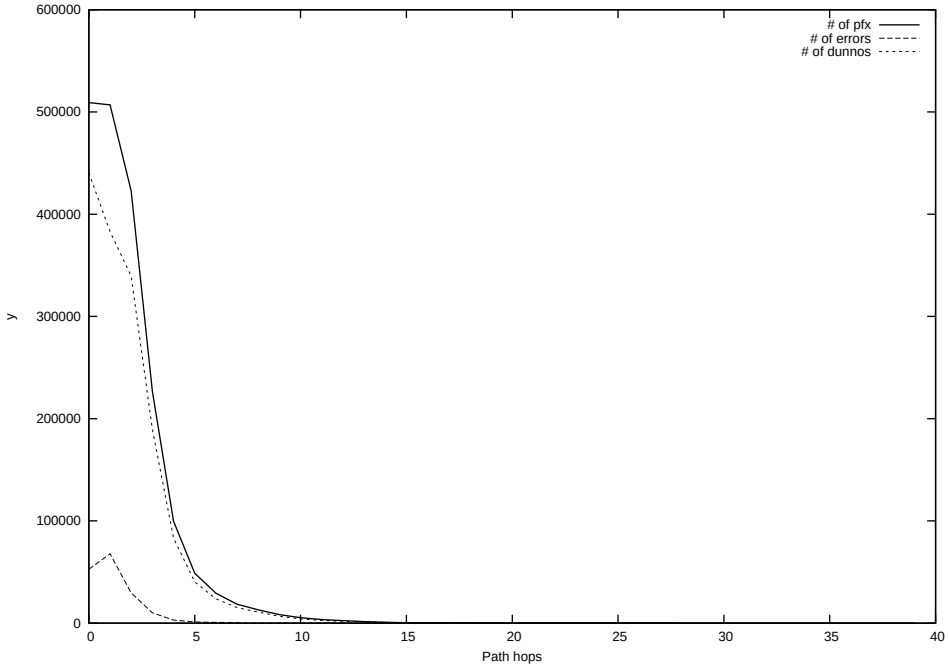
2014-09-19



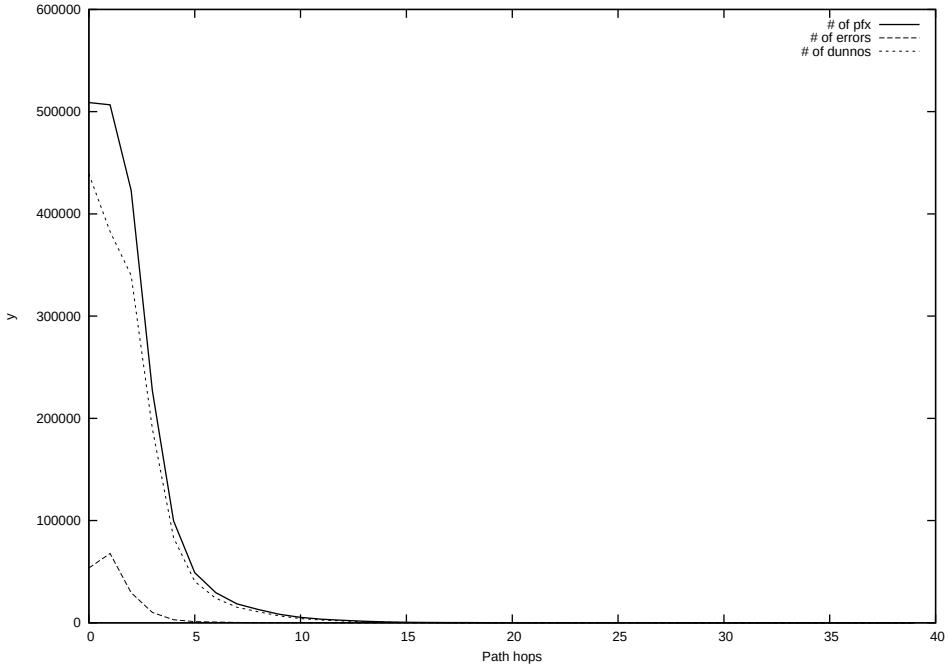
2014-09-20



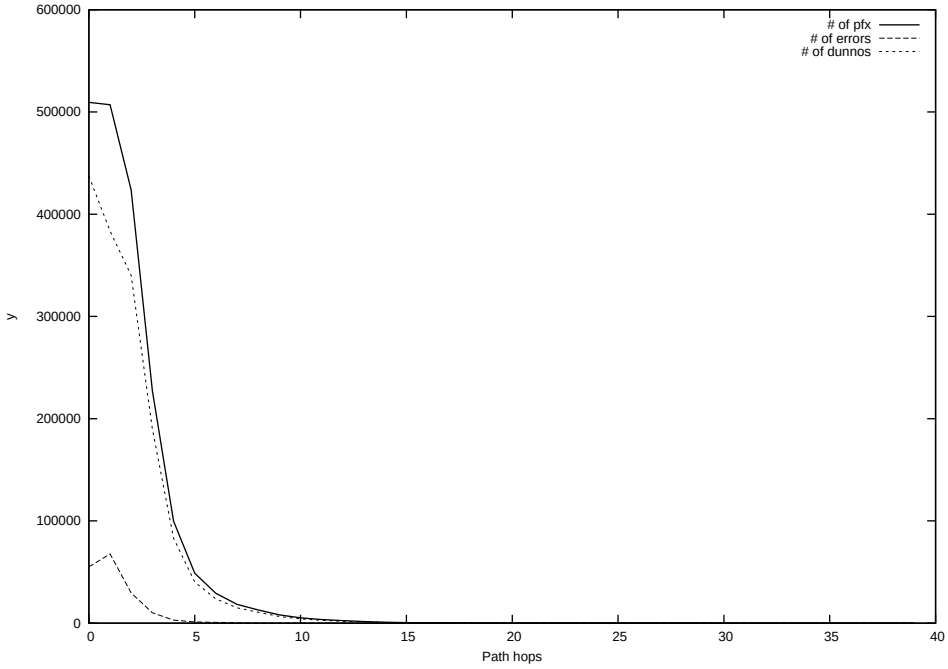
2014-09-21



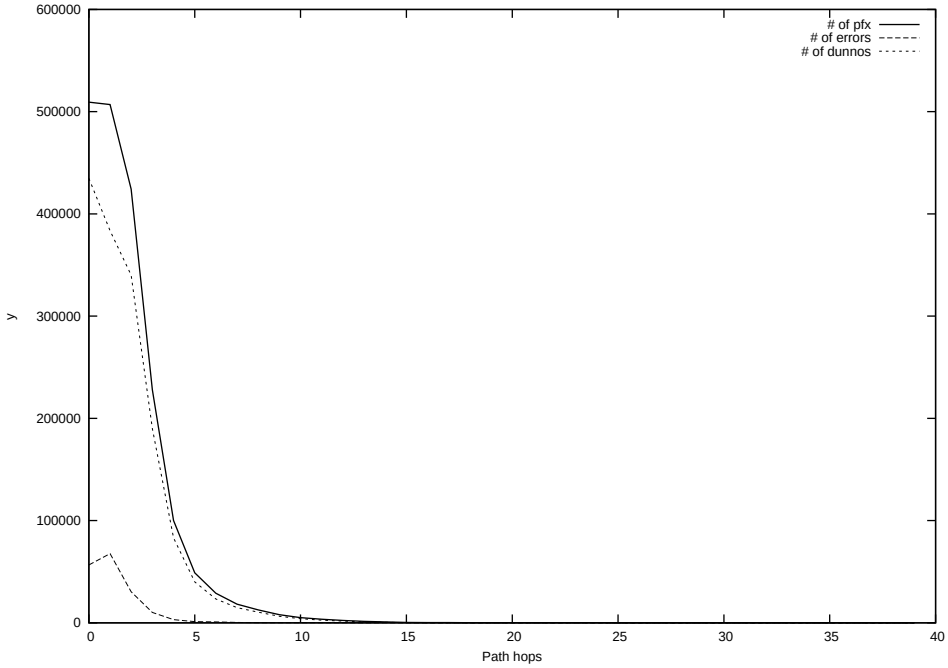
2014-09-22



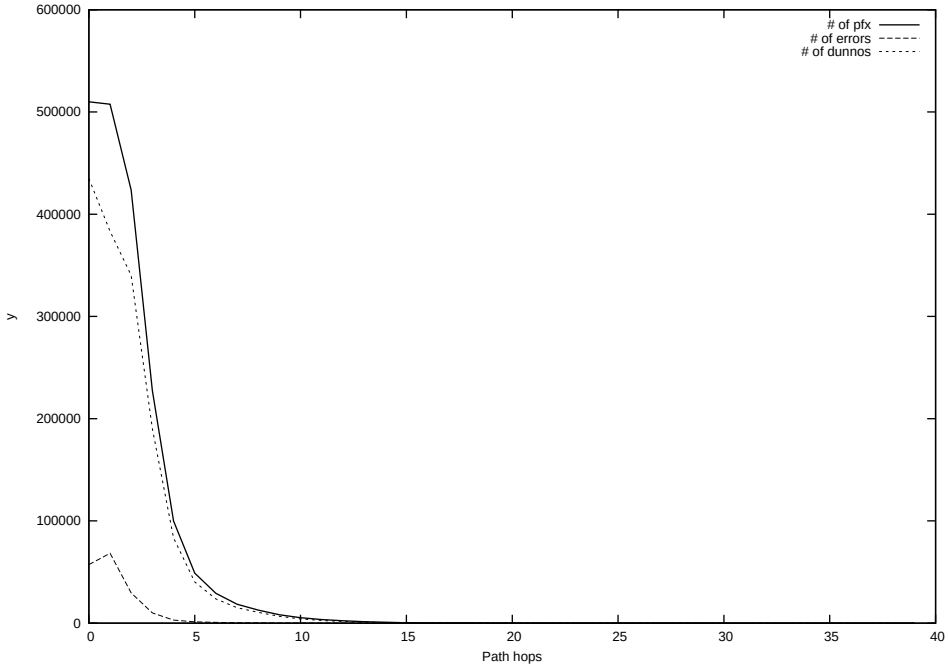
2014-09-23



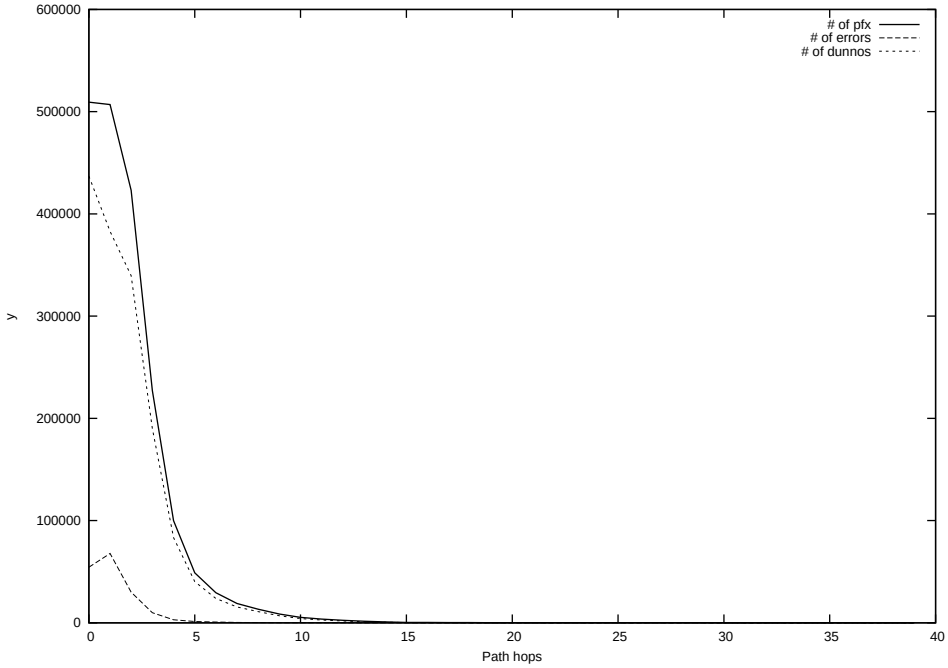
2014-09-24



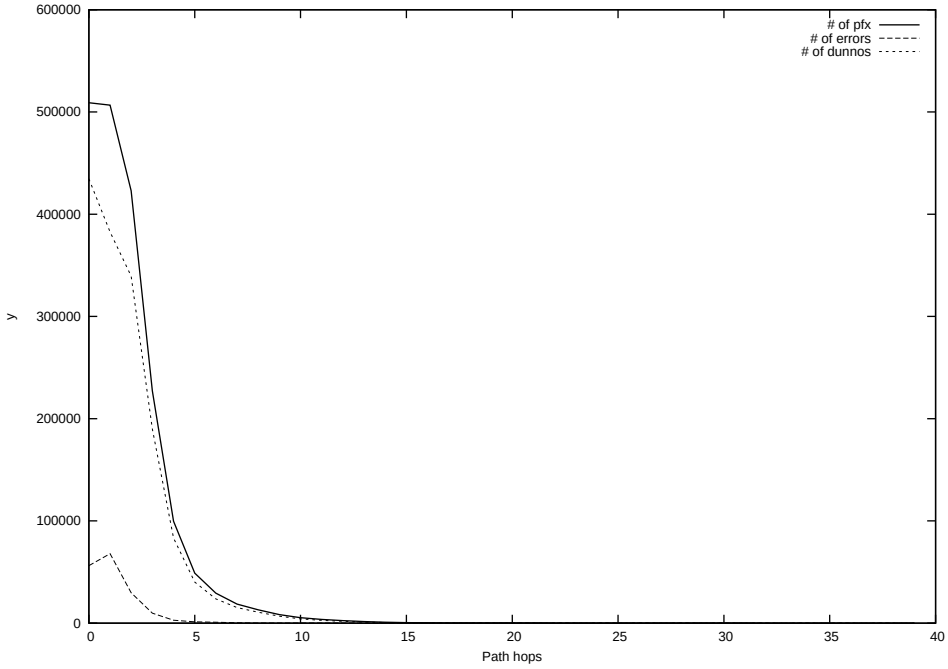
2014-09-25



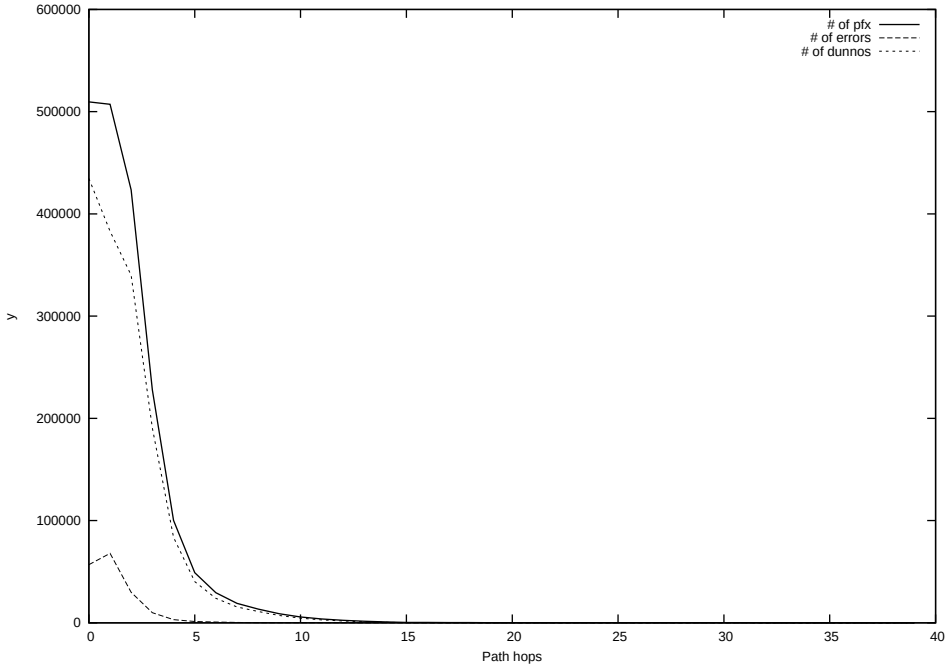
2014-09-26



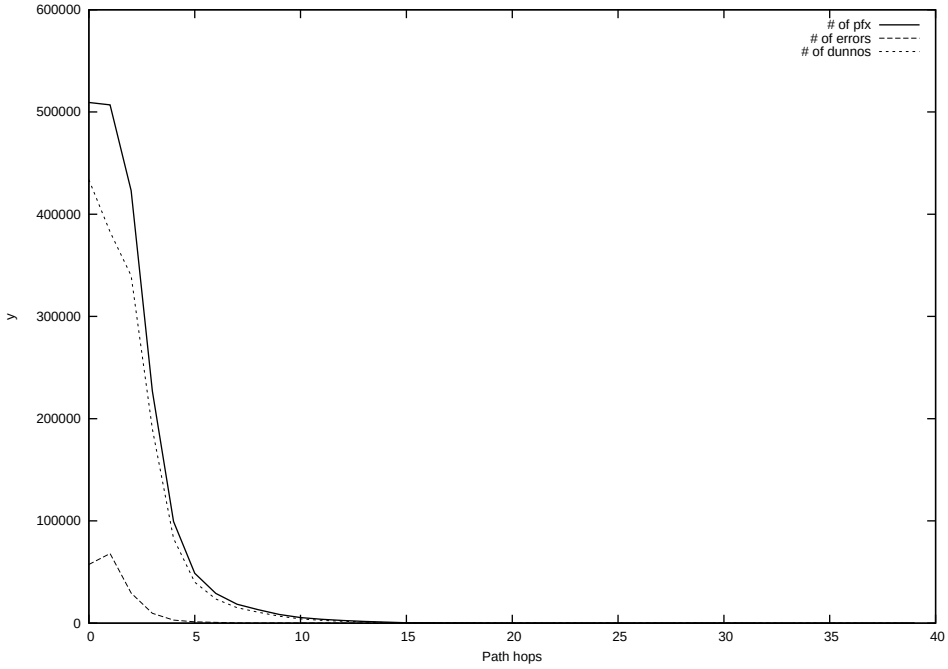
2014-09-27



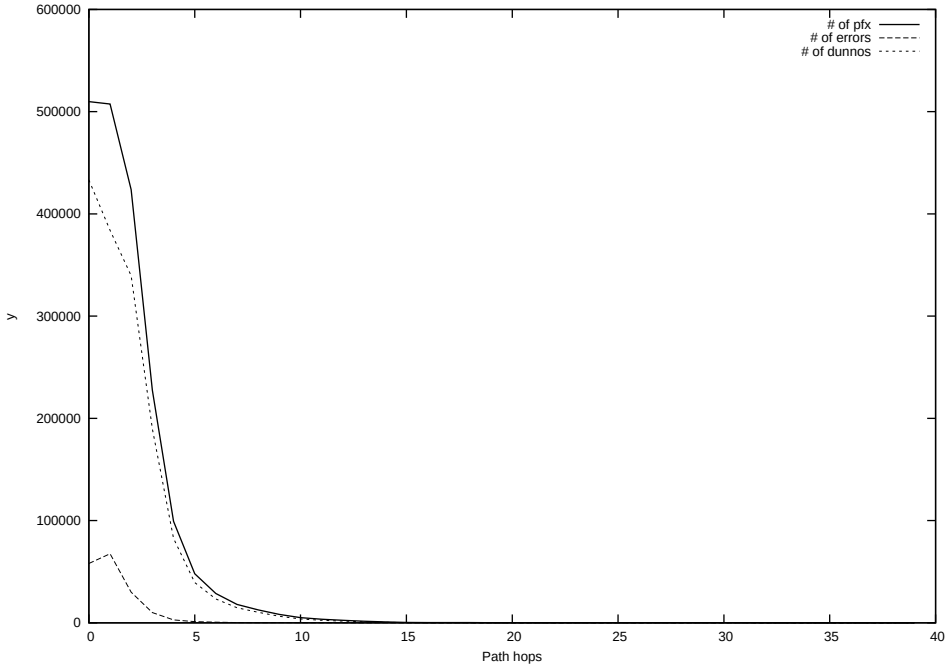
2014-09-28



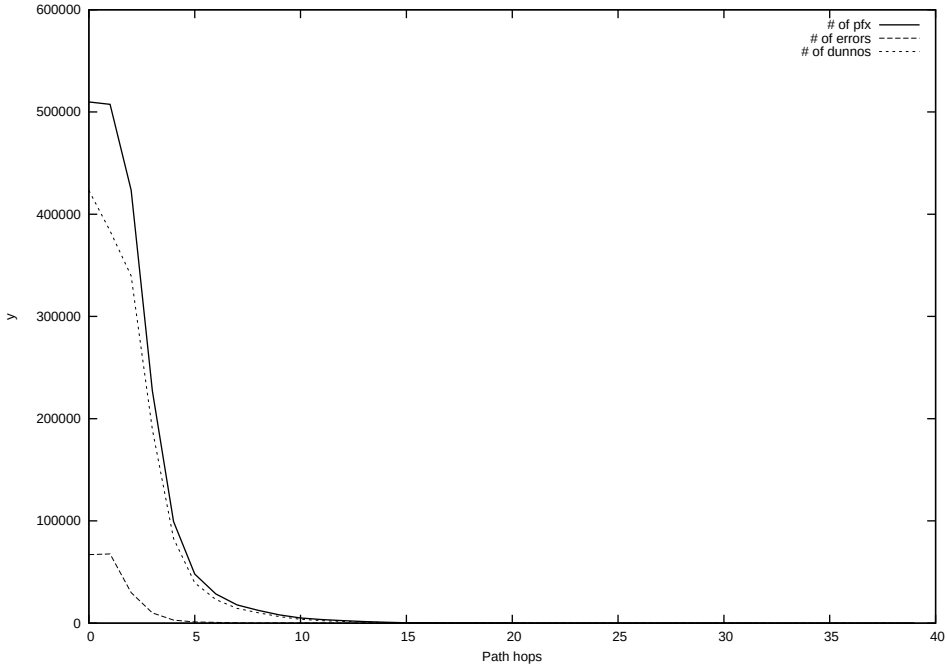
2014-09-29



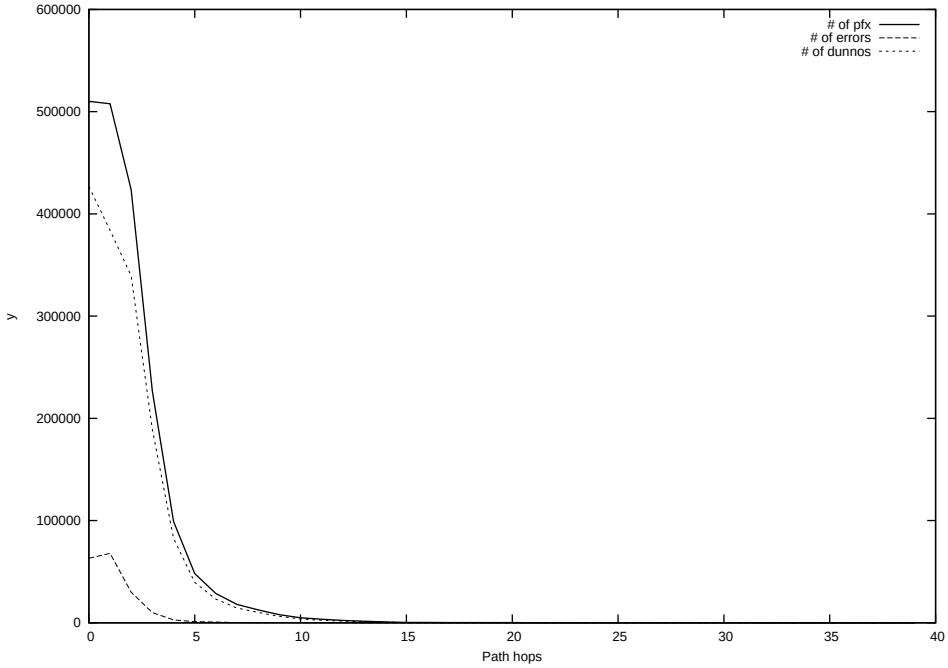
2014-09-30



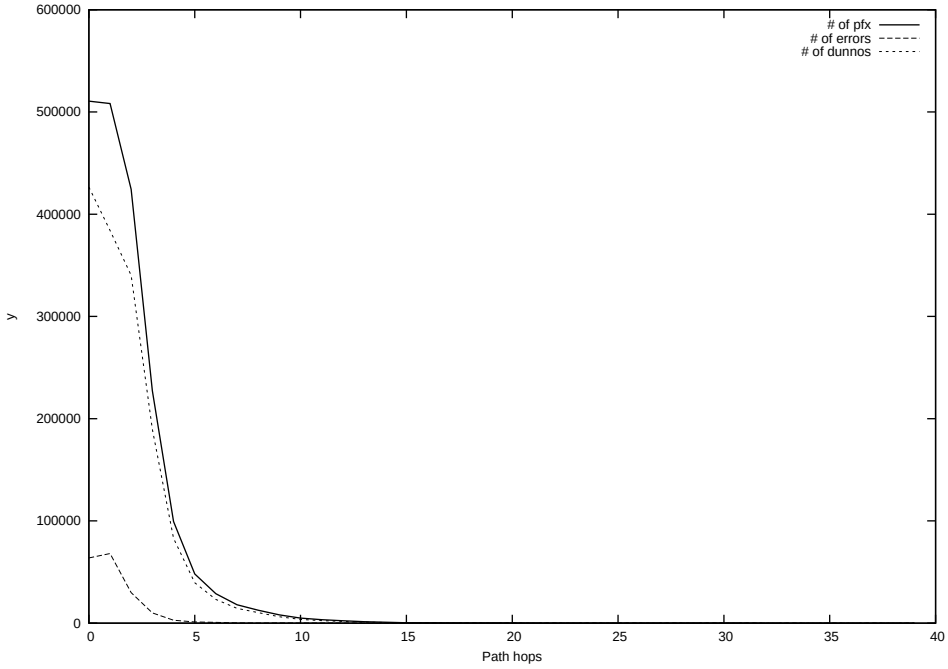
2014-10-01



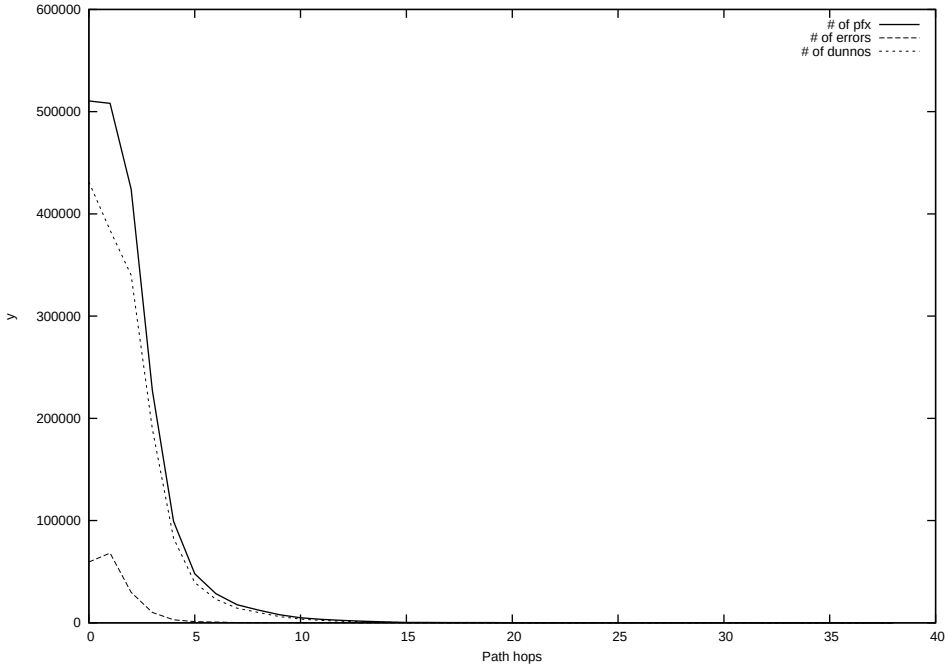
2014-10-02



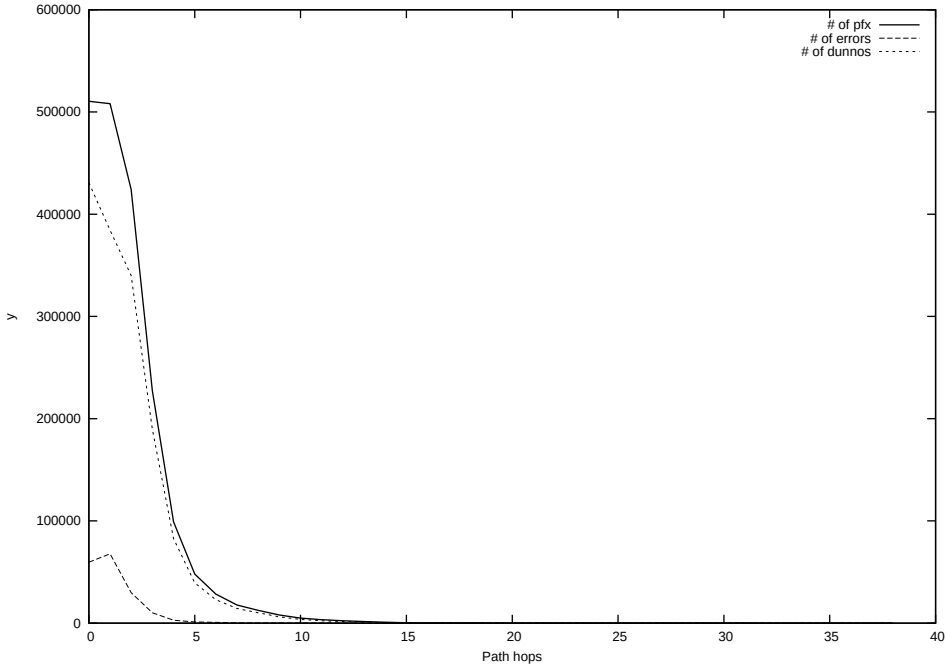
2014-10-03



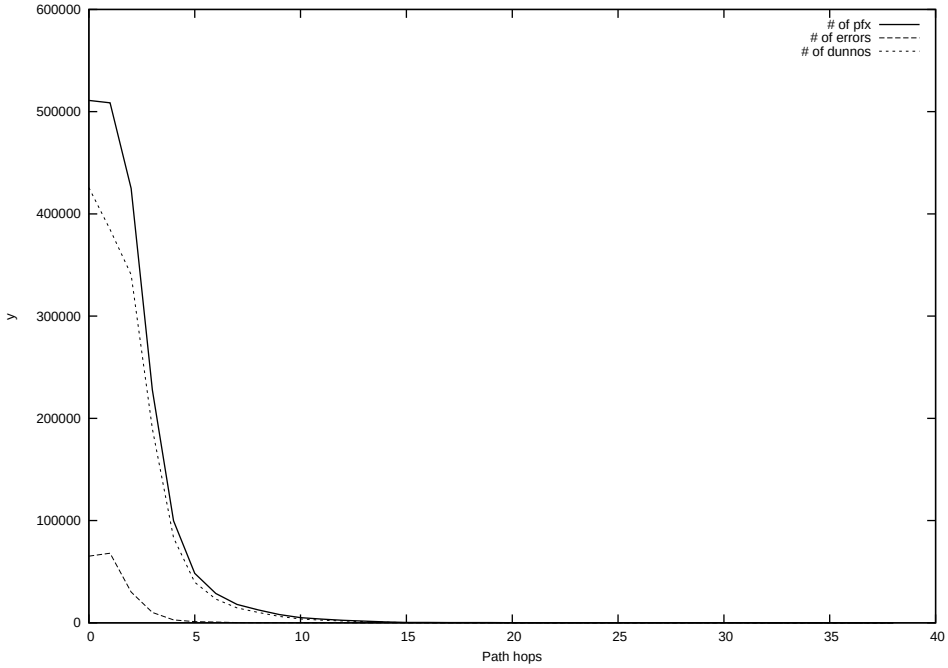
2014-10-04



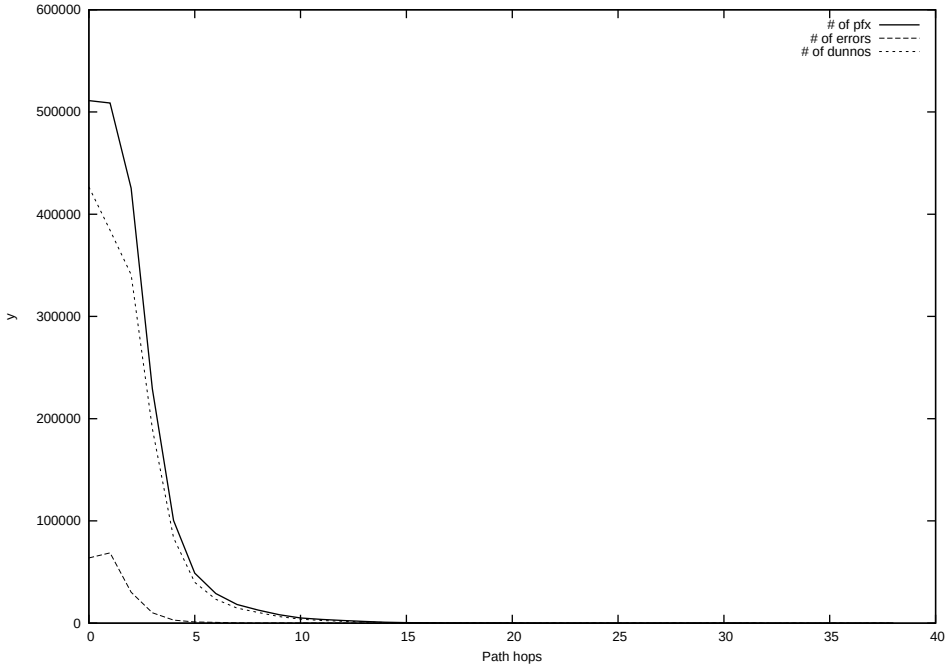
2014-10-05



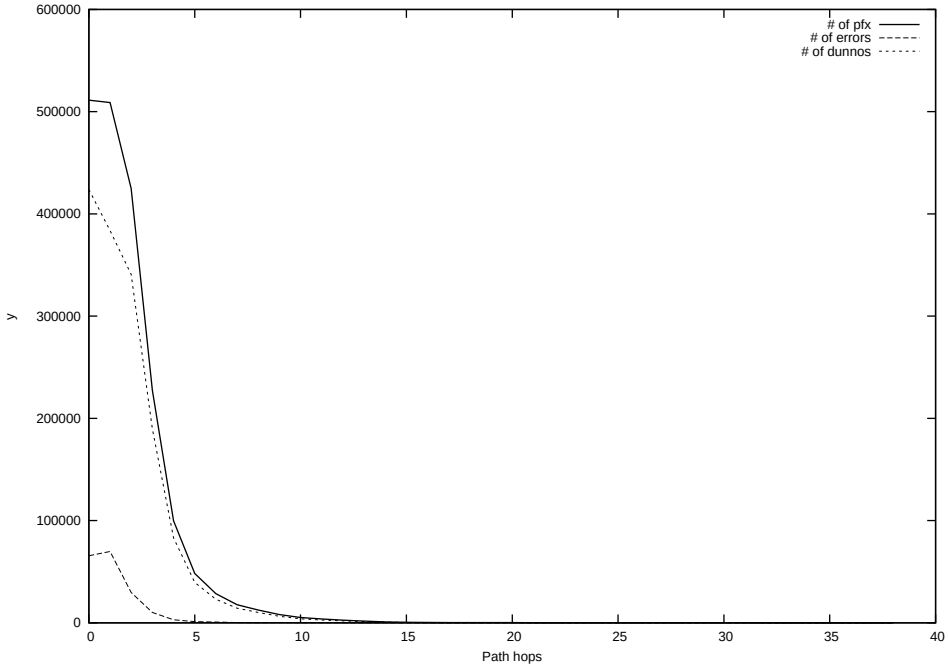
2014-10-06



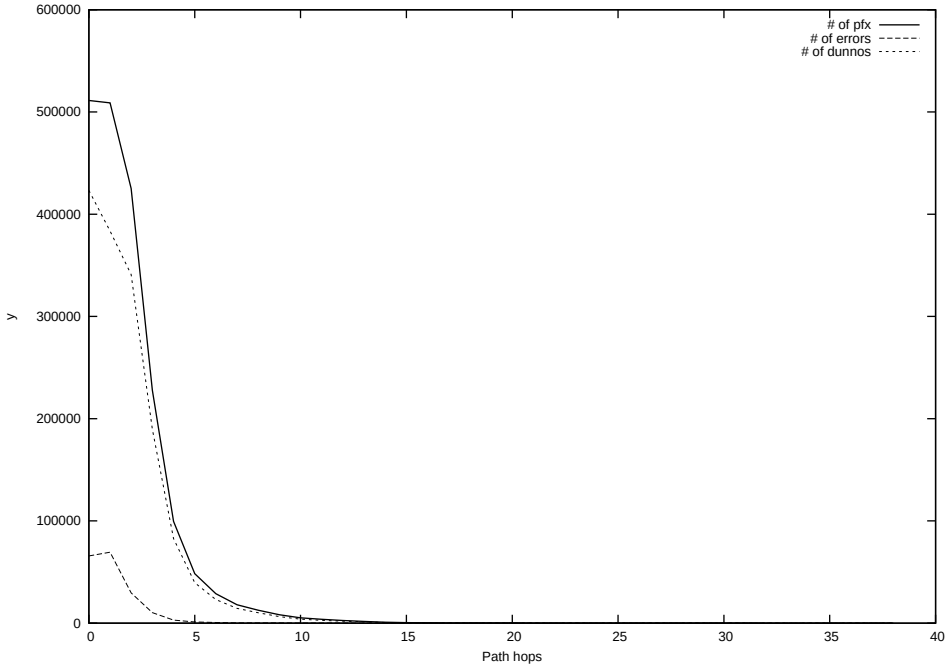
2014-10-07



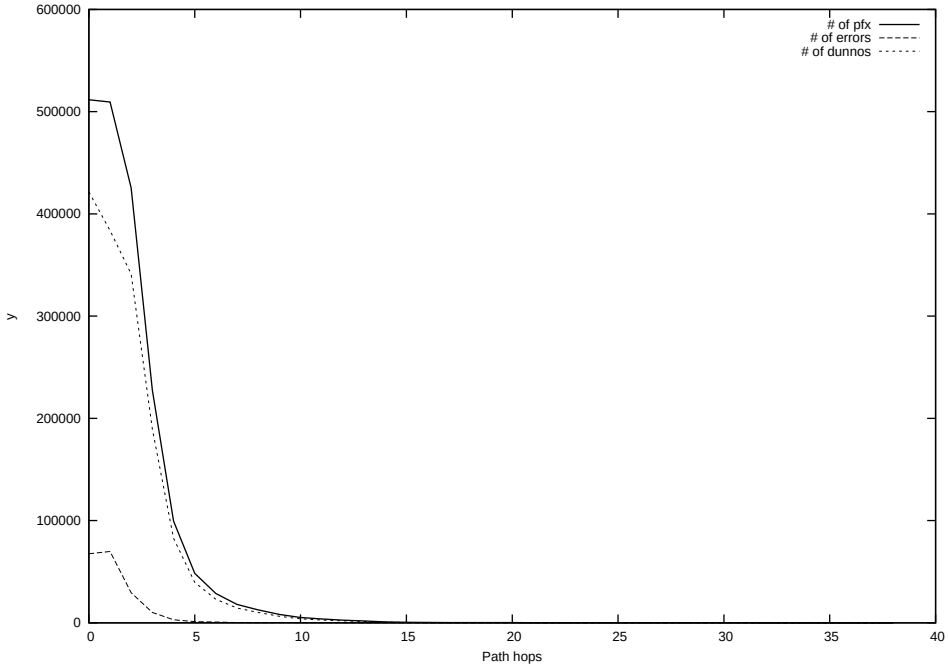
2014-10-08



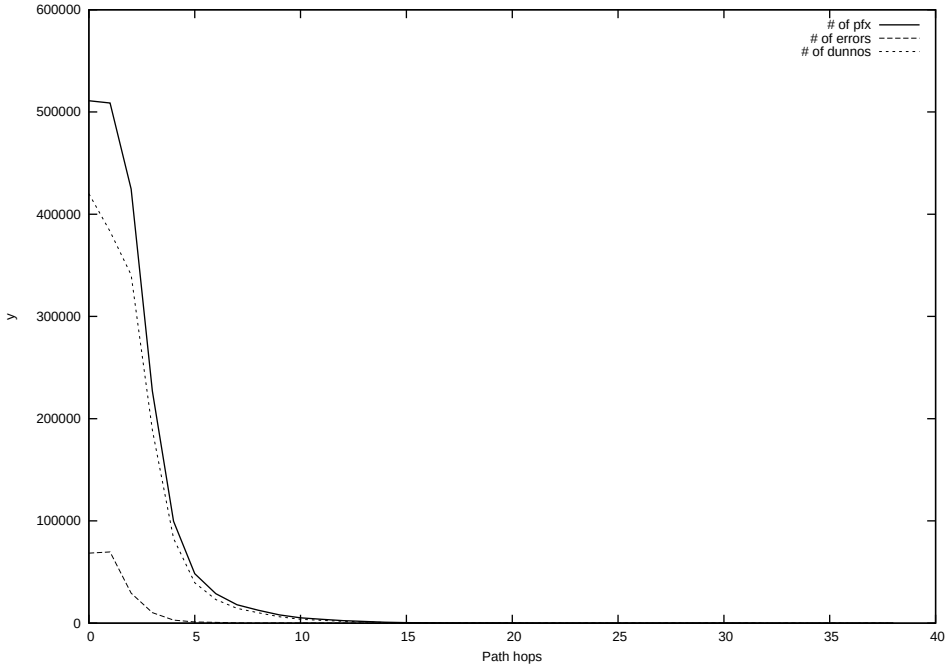
2014-10-09



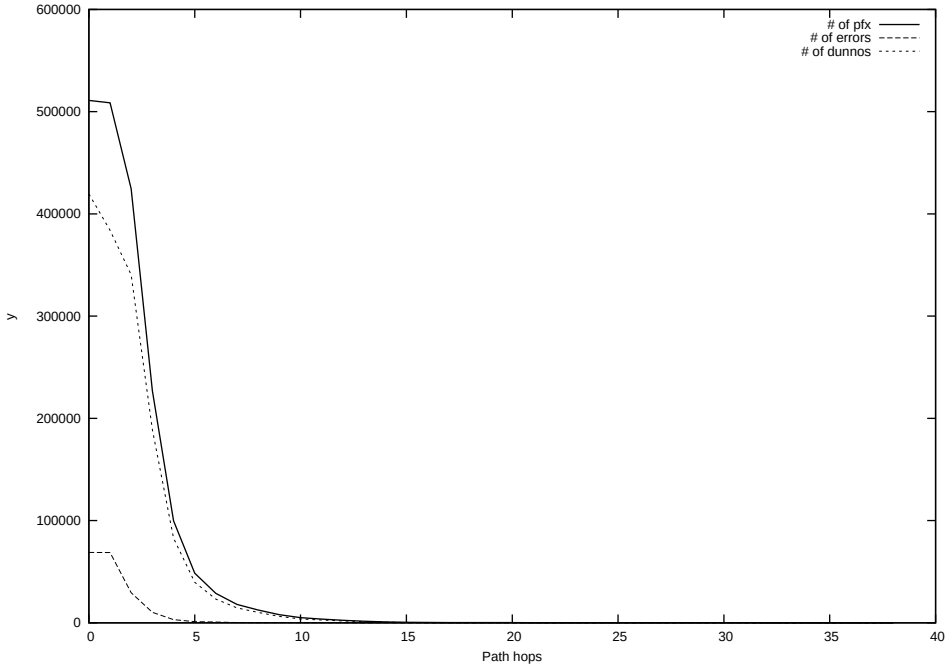
2014-10-10



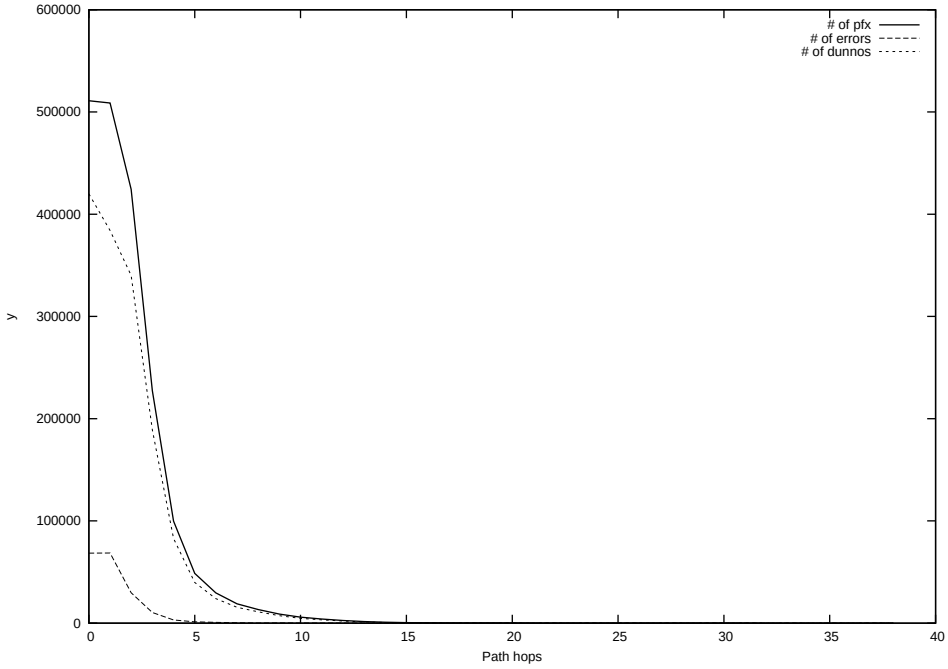
2014-10-11



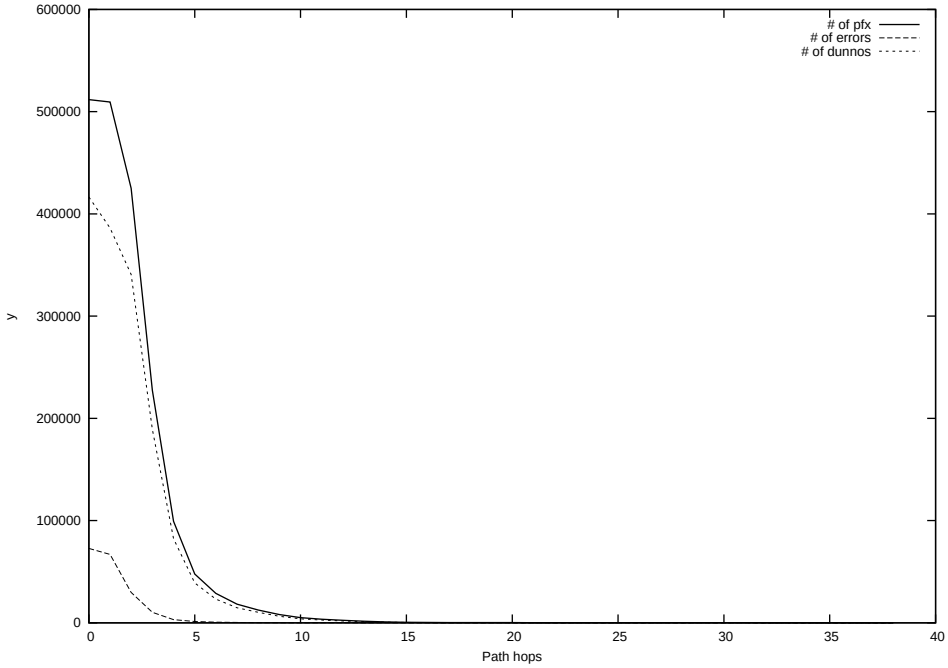
2014-10-12



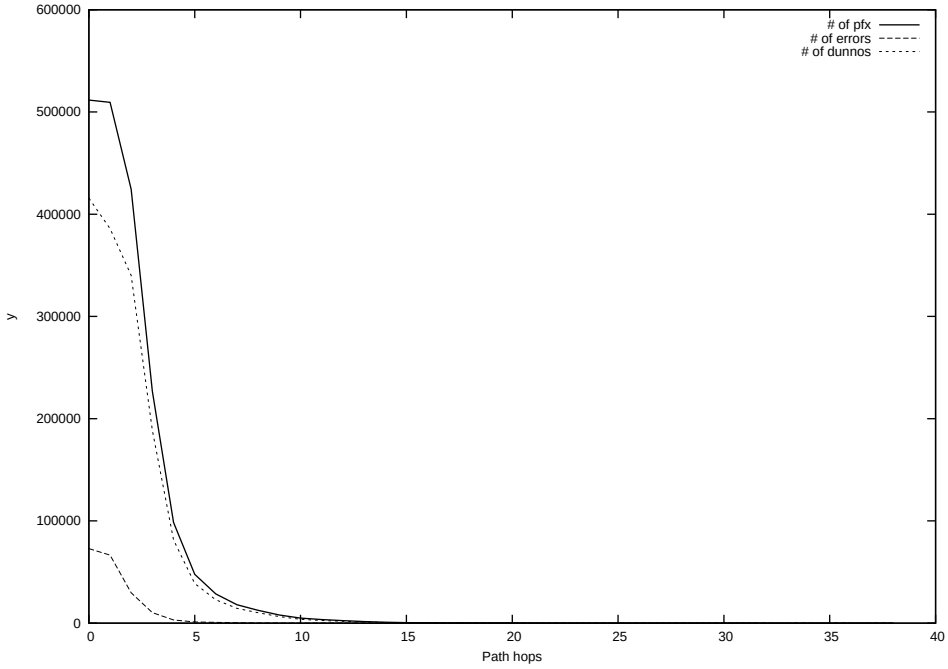
2014-10-13



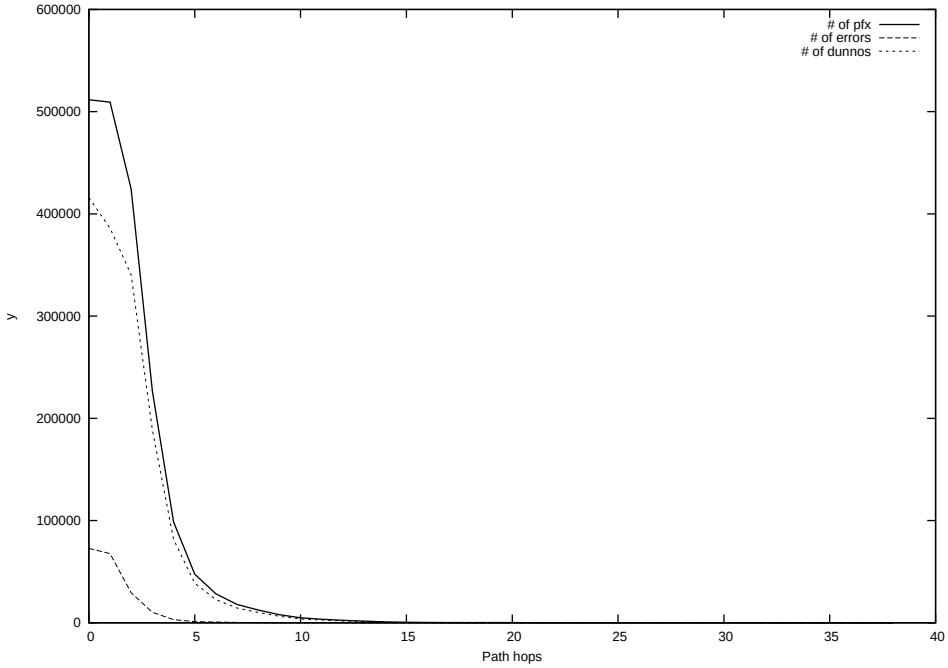
2014-10-14



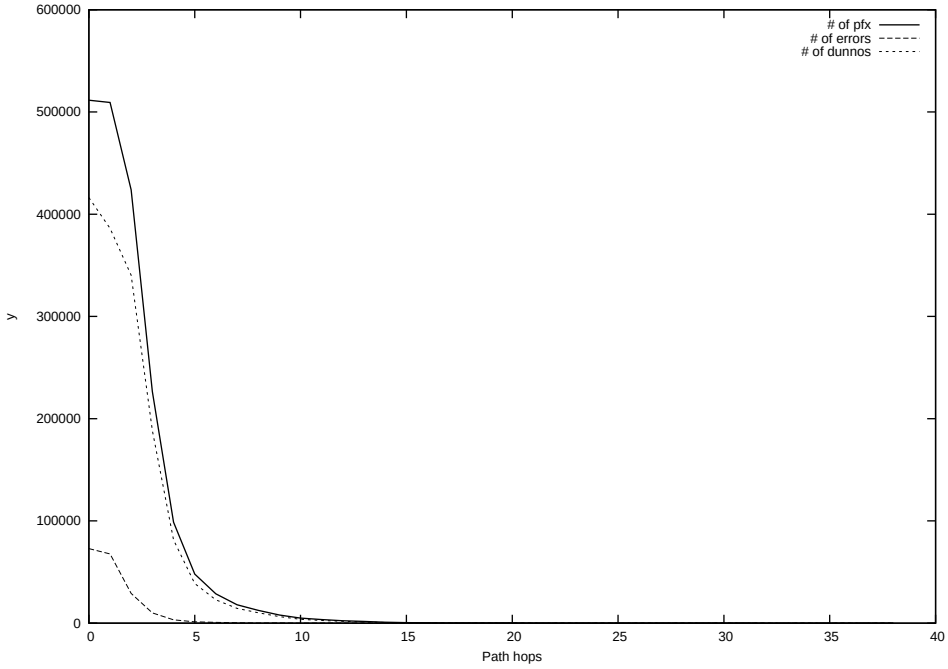
2014-10-15



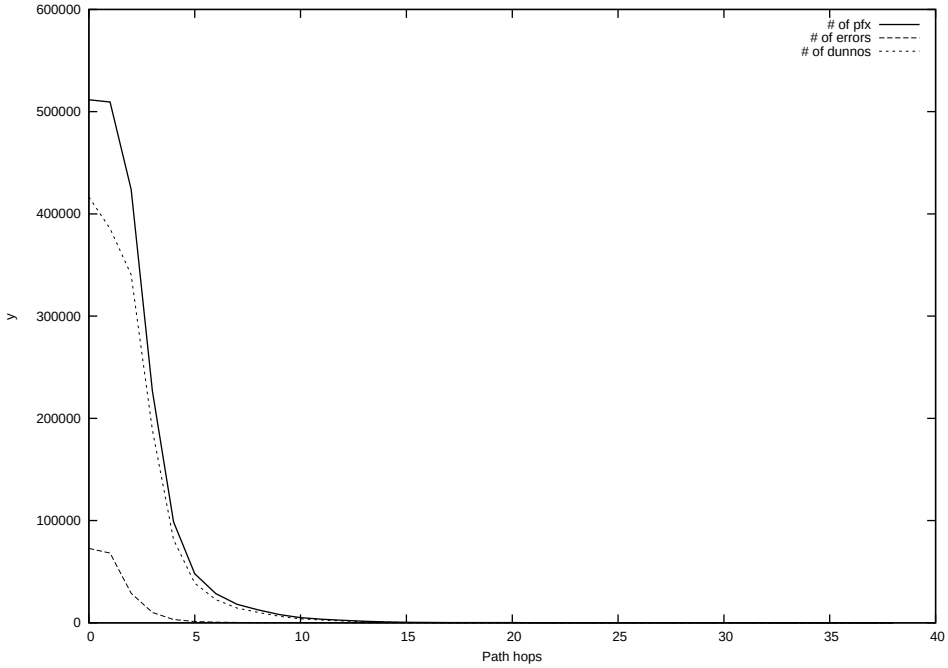
2014-10-16



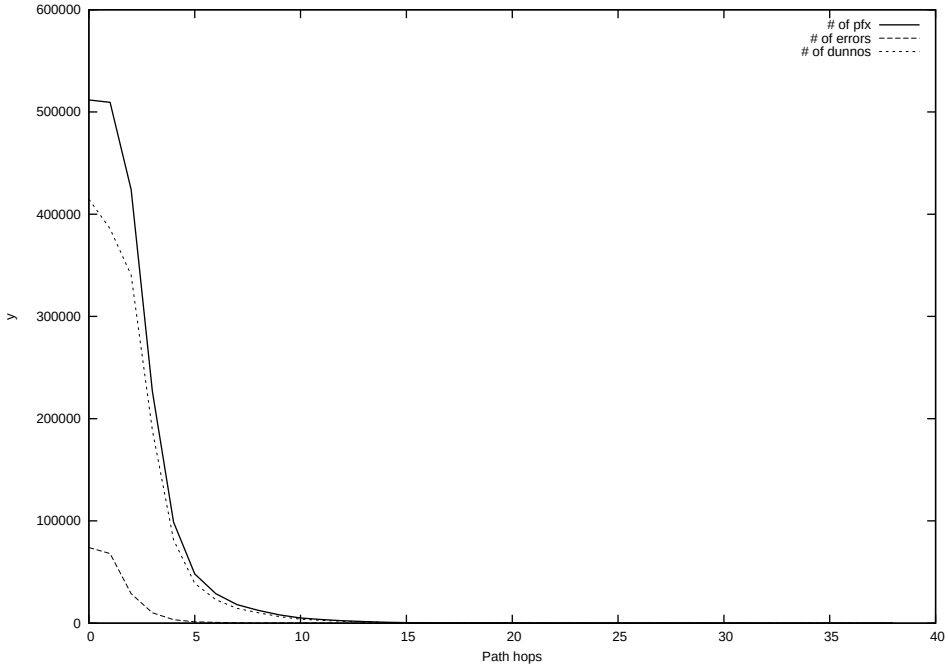
2014-10-17



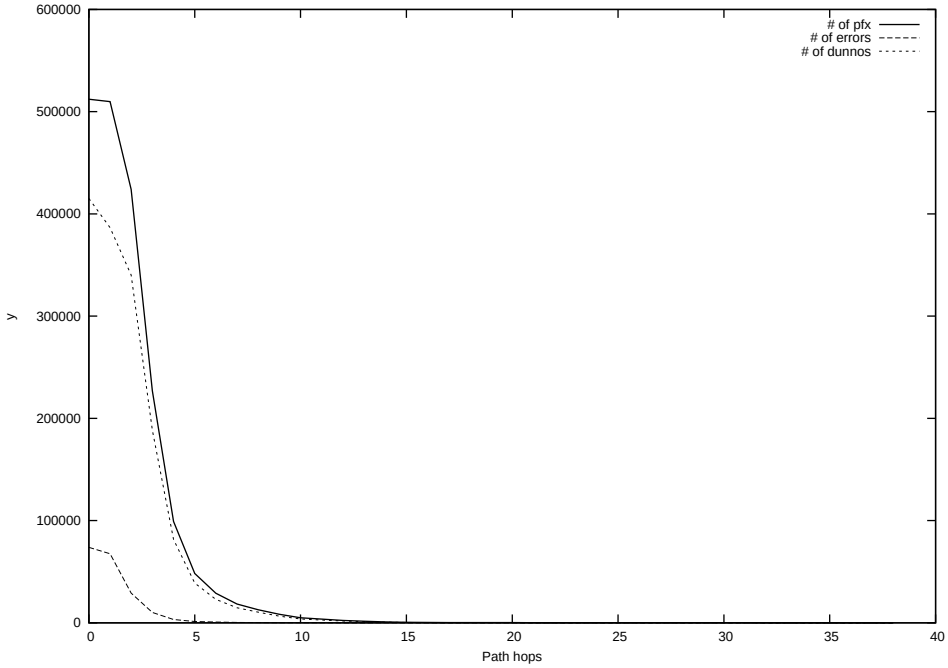
2014-10-18



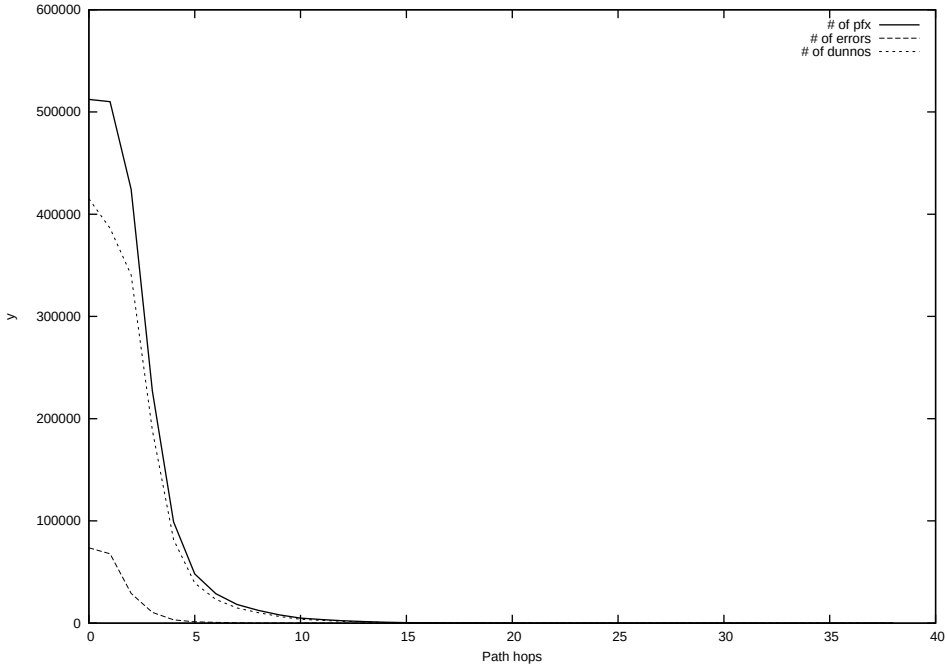
2014-10-19



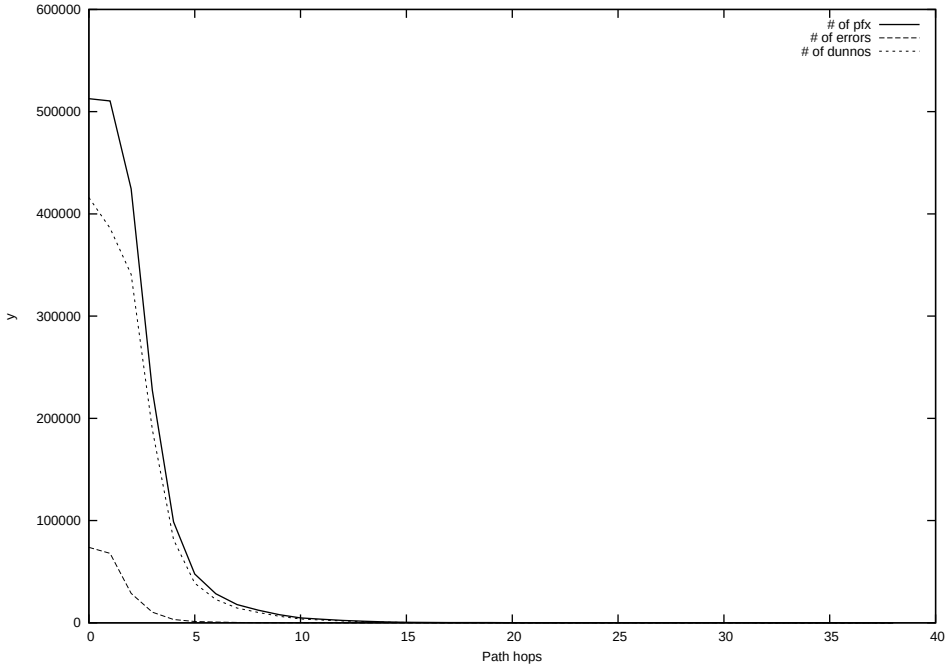
2014-10-20



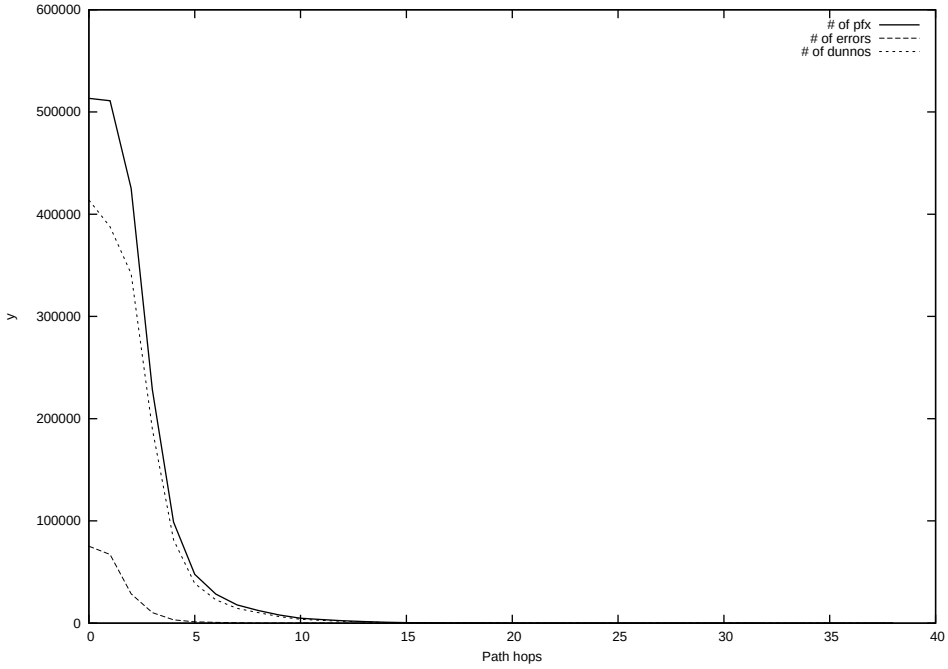
2014-10-21



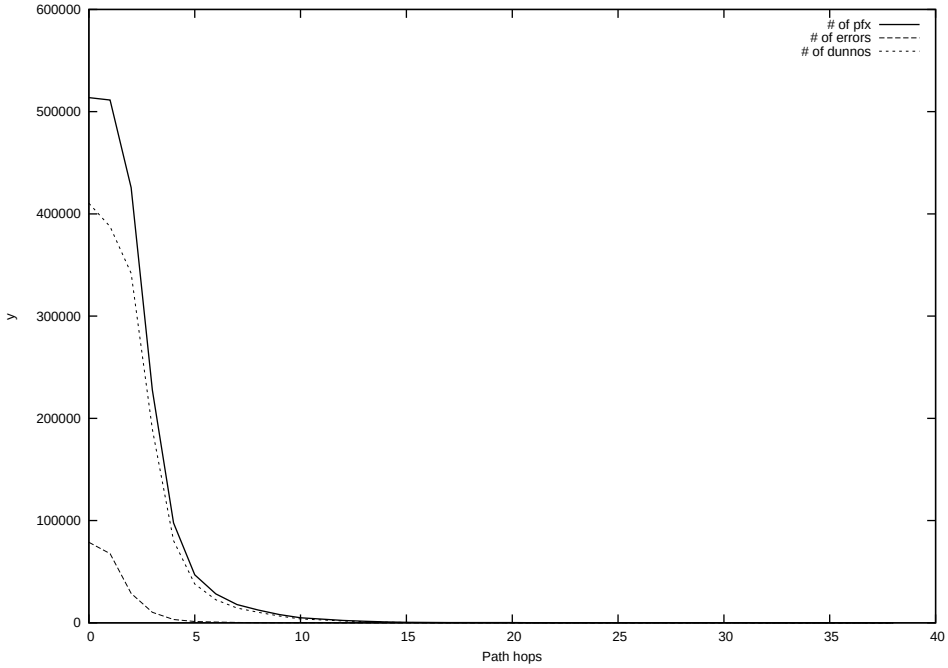
2014-10-22



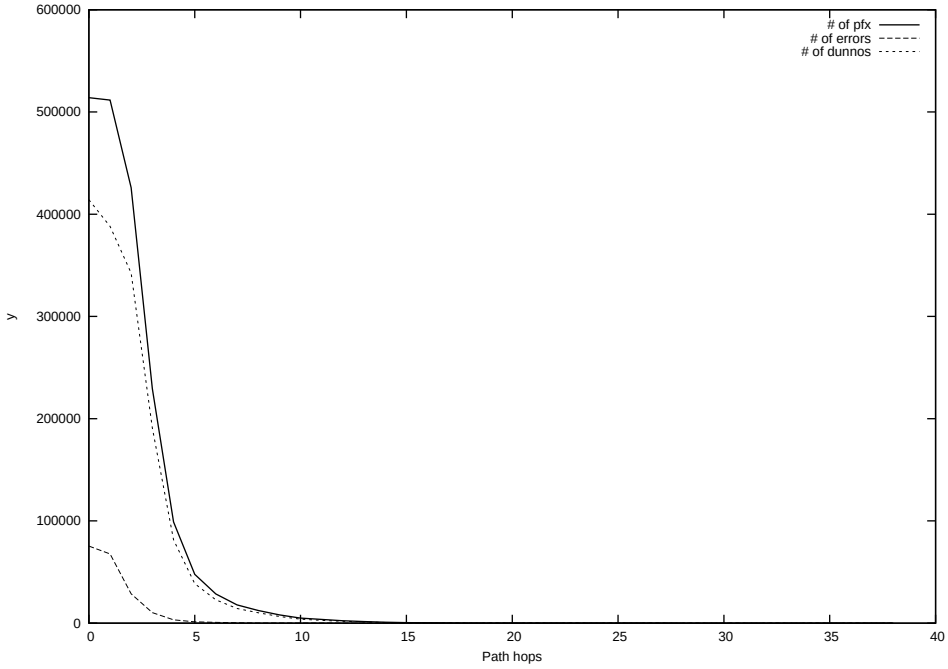
2014-10-23



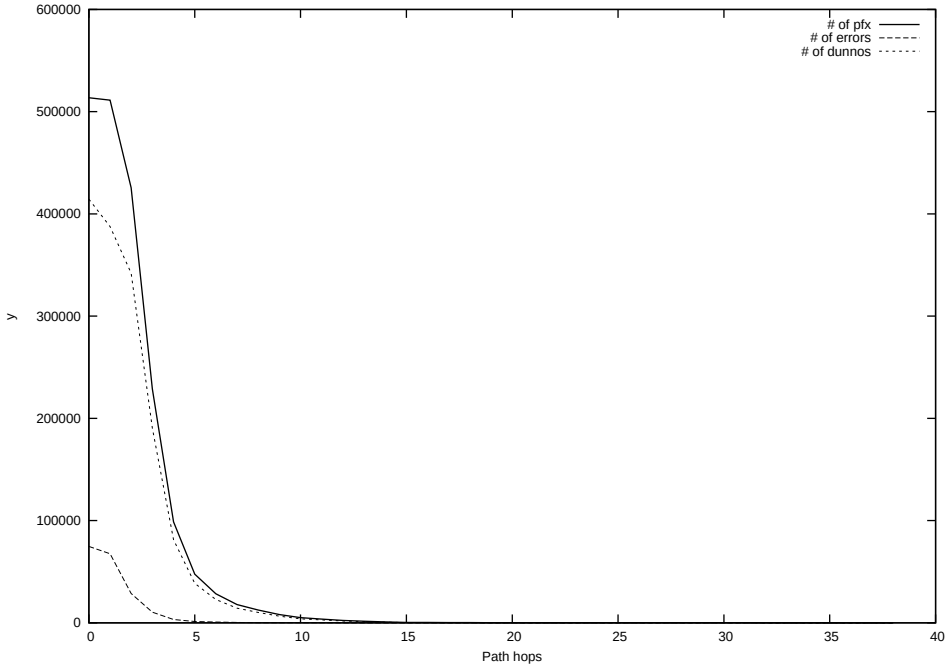
2014-10-24



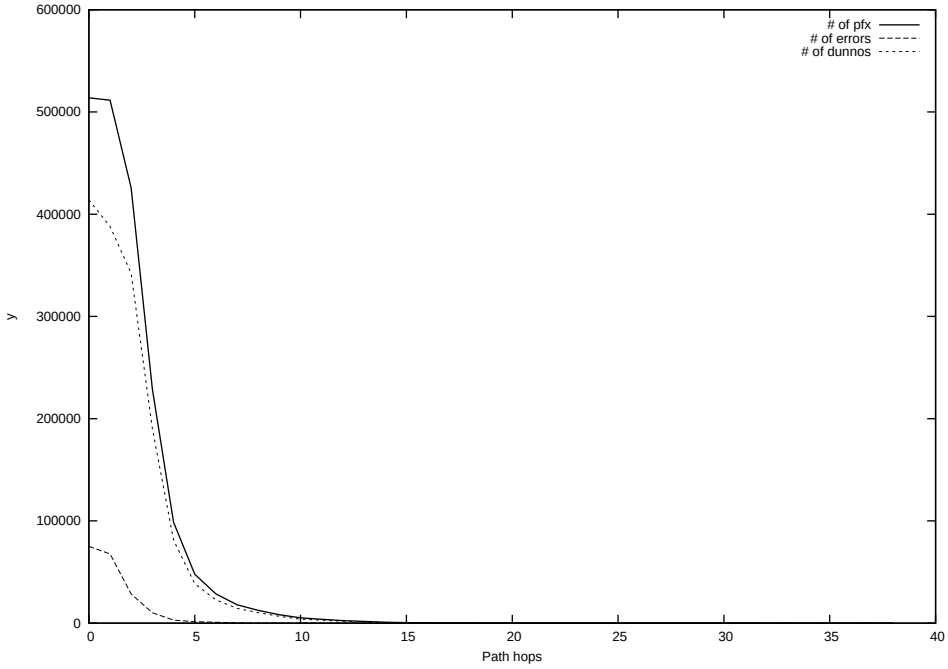
2014-10-25



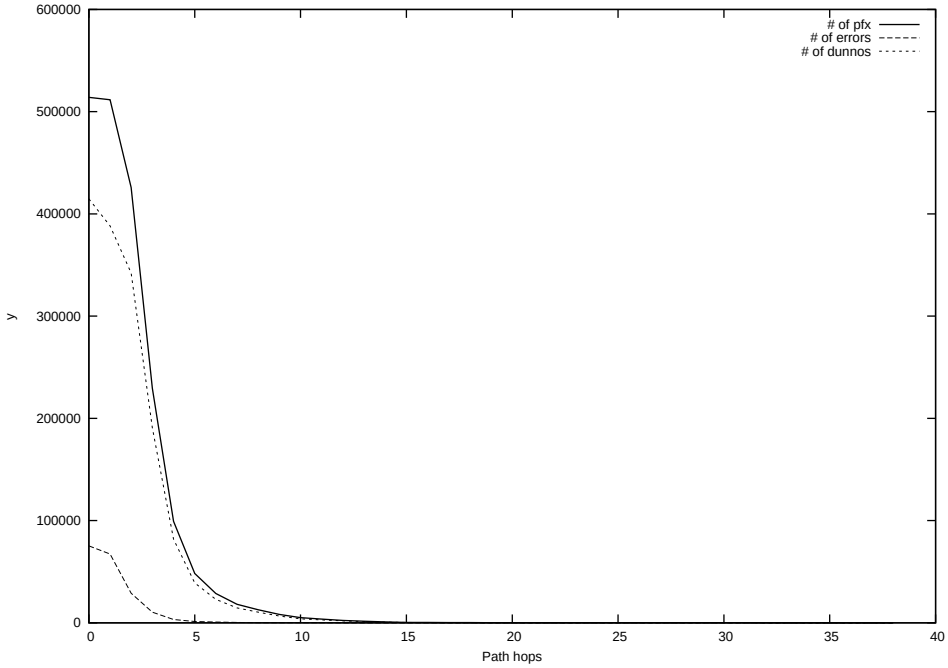
2014-10-26



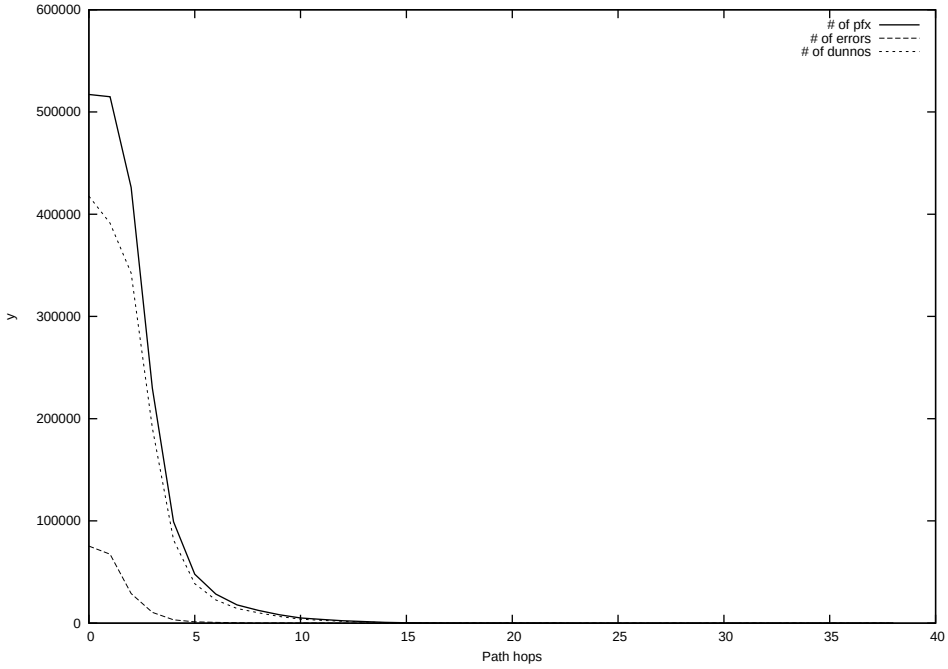
2014-10-27



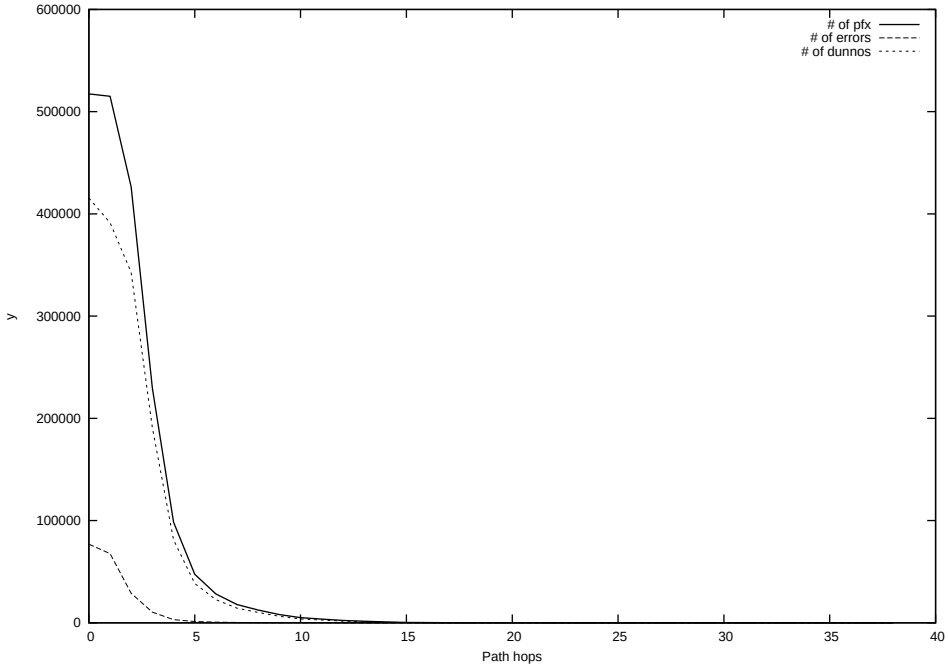
2014-10-28



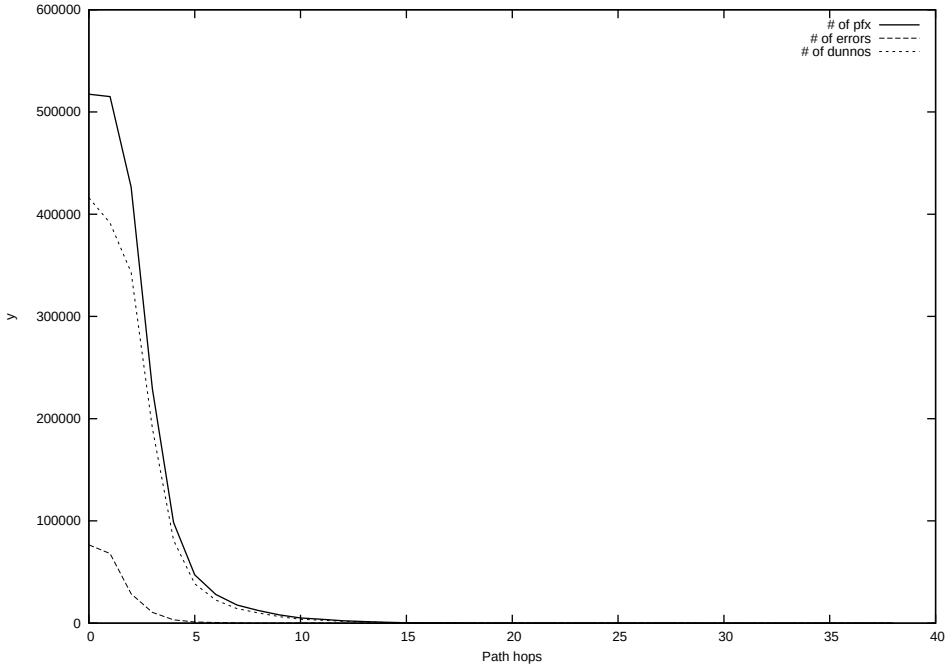
2014-10-29



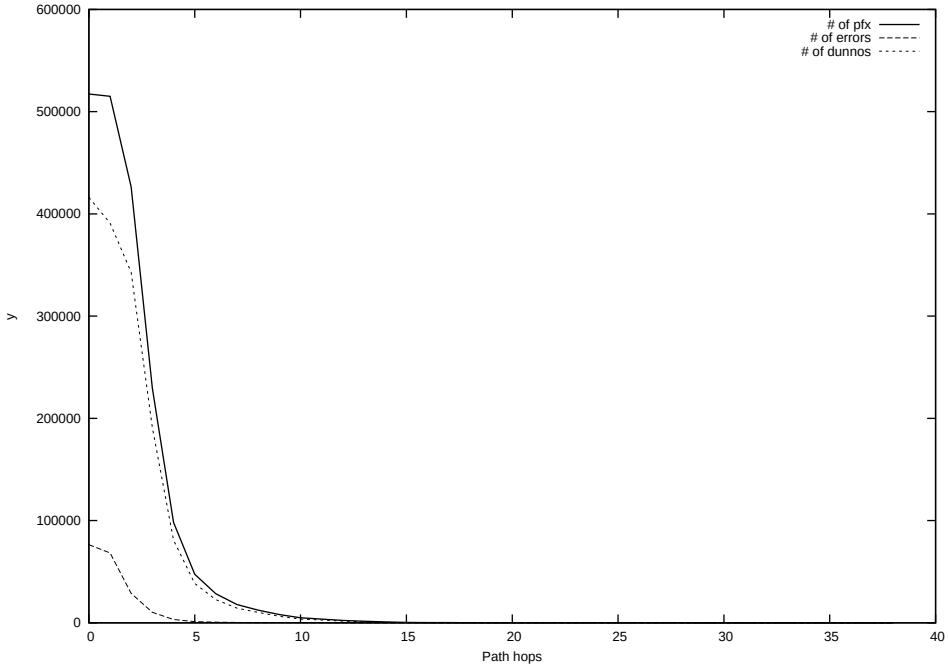
2014-10-30



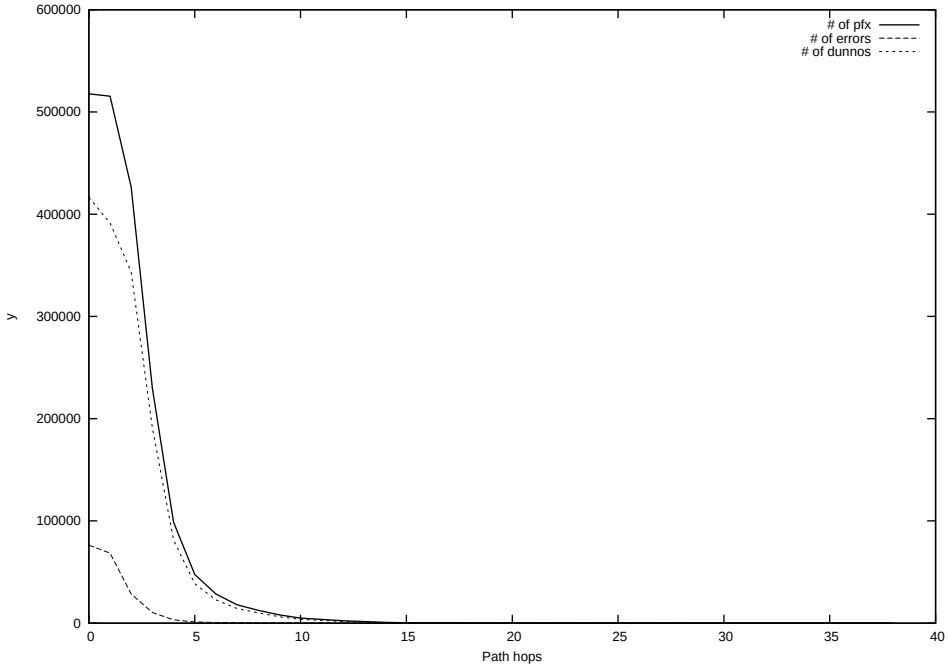
2014-10-31



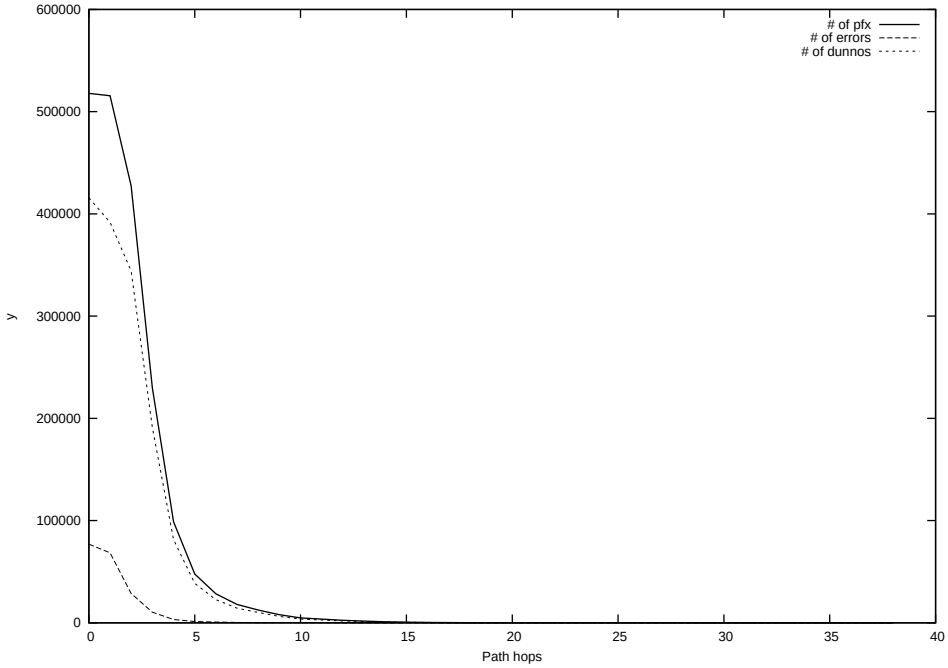
2014-11-02



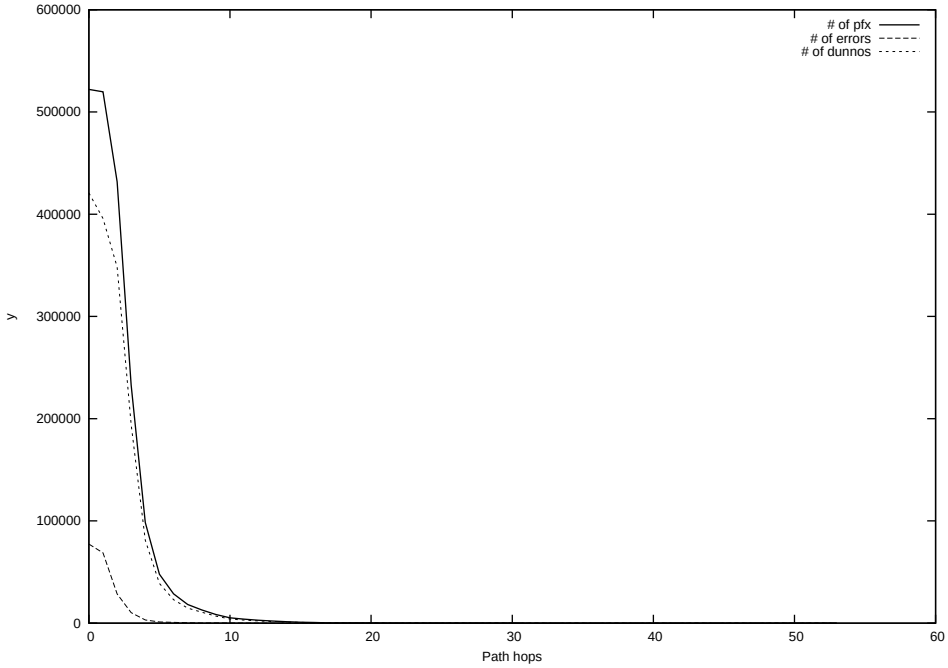
2014-11-03



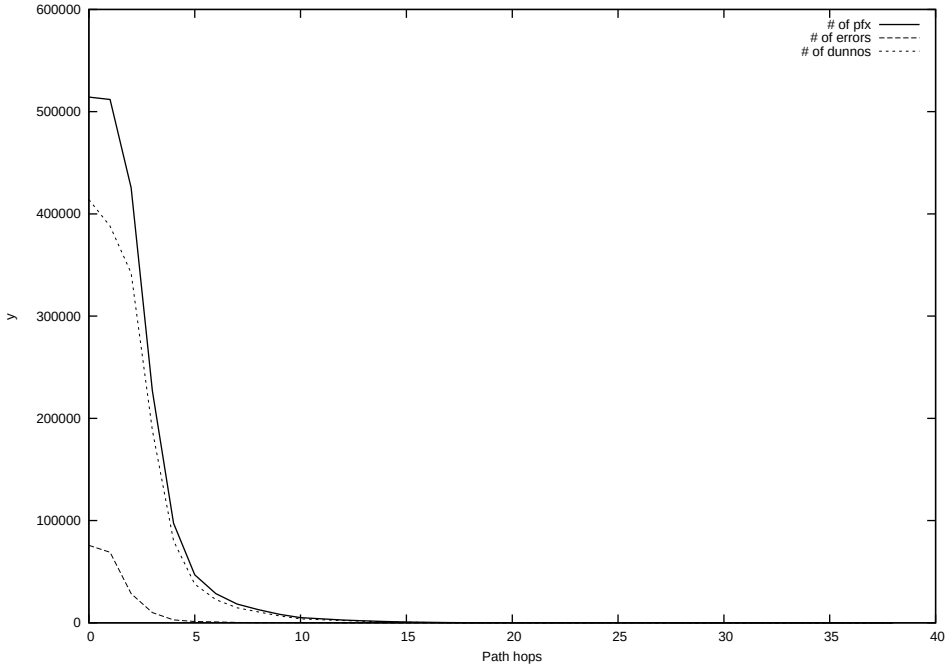
2014-11-04



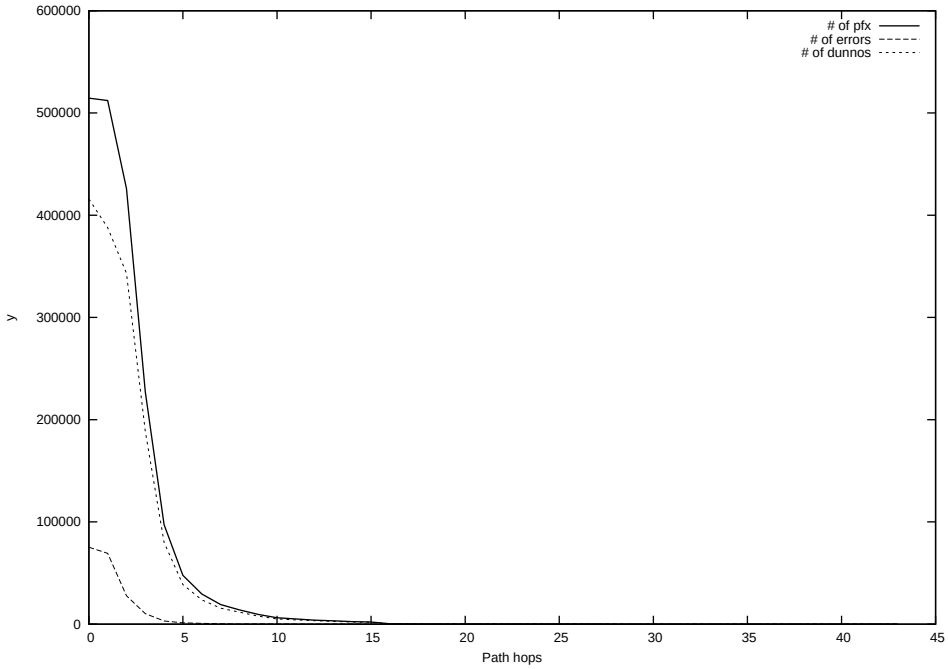
2014-11-05



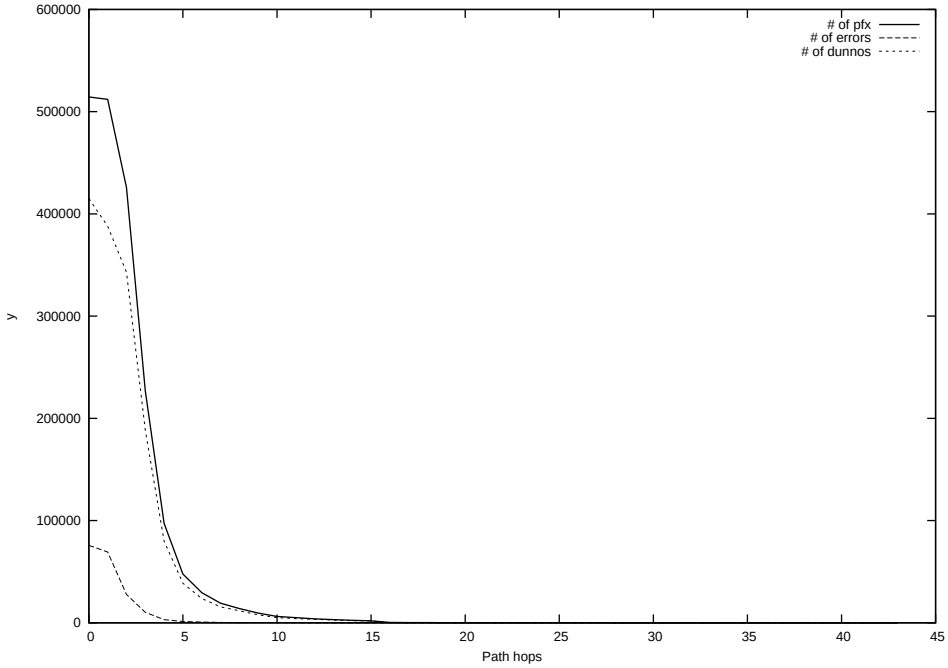
2014-11-06



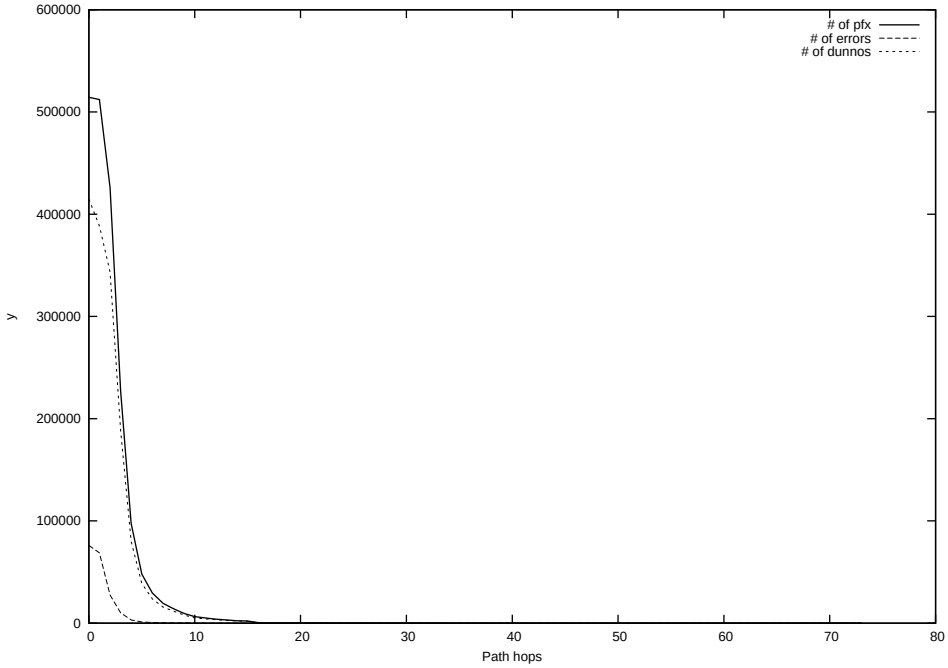
2014-11-07



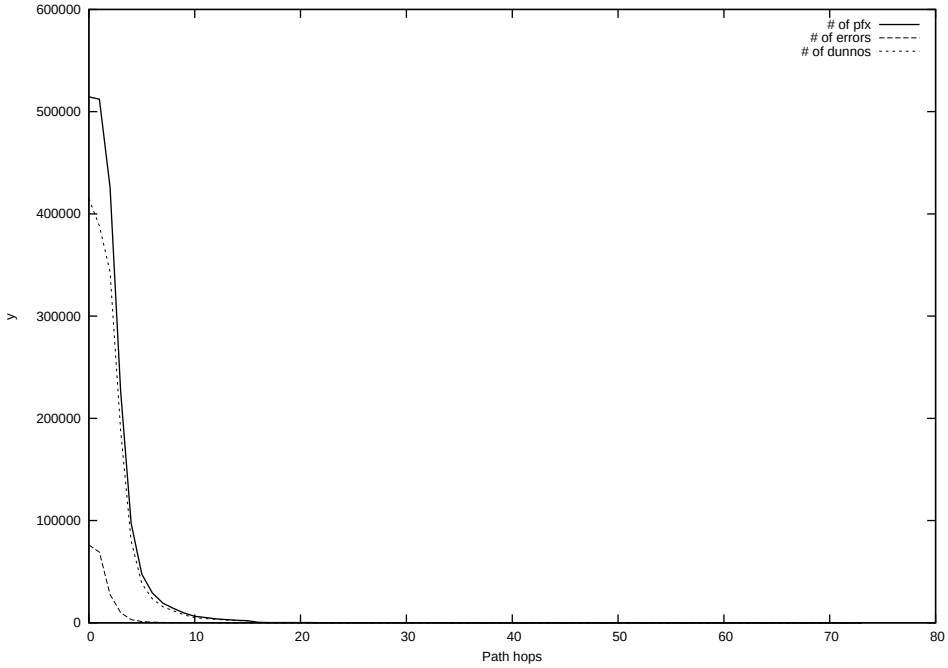
2014-11-08



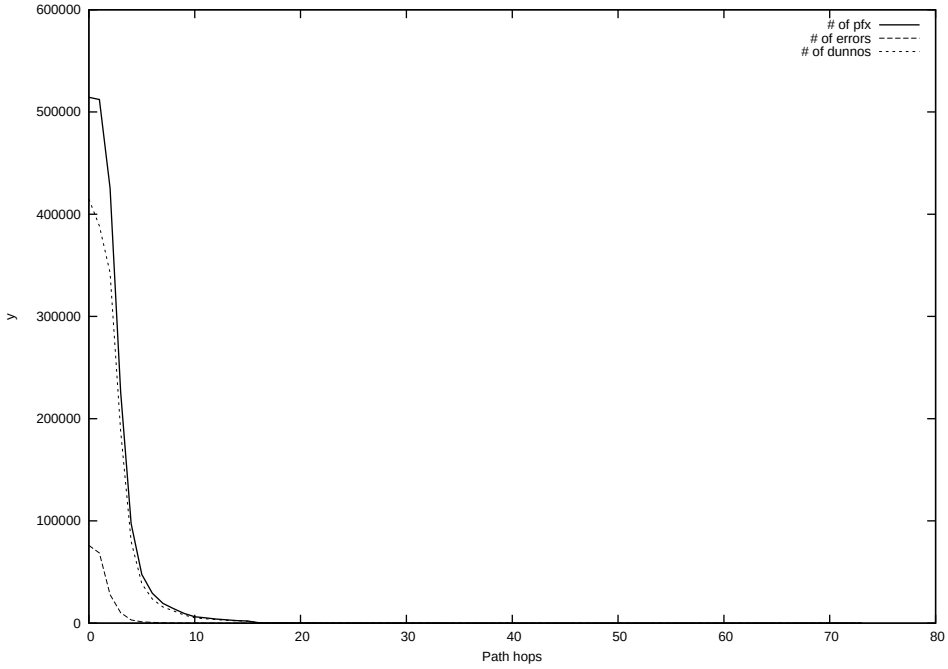
2014-11-09



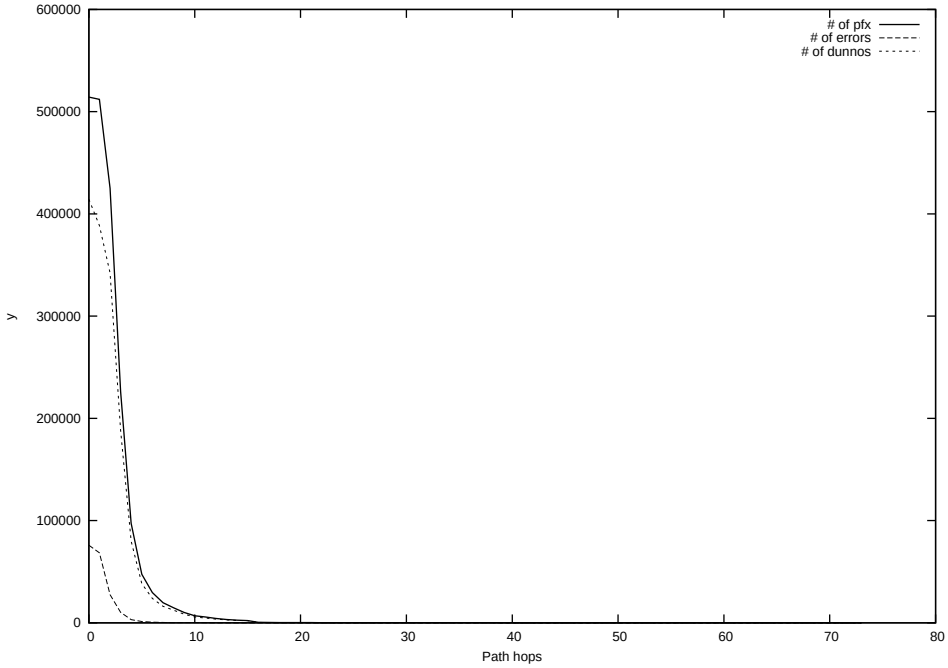
2014-11-10



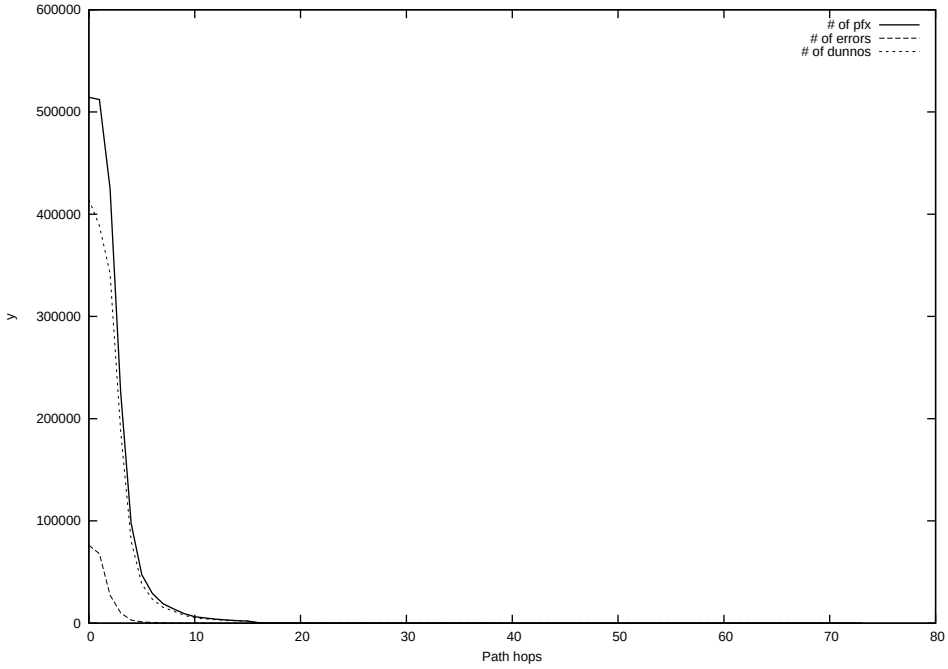
2014-11-11



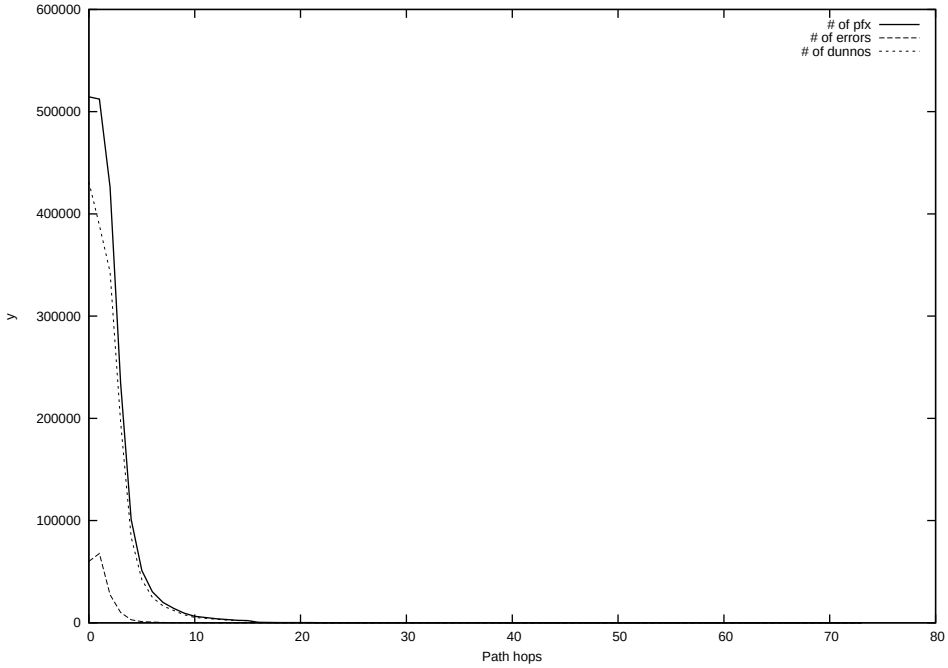
2014-11-12



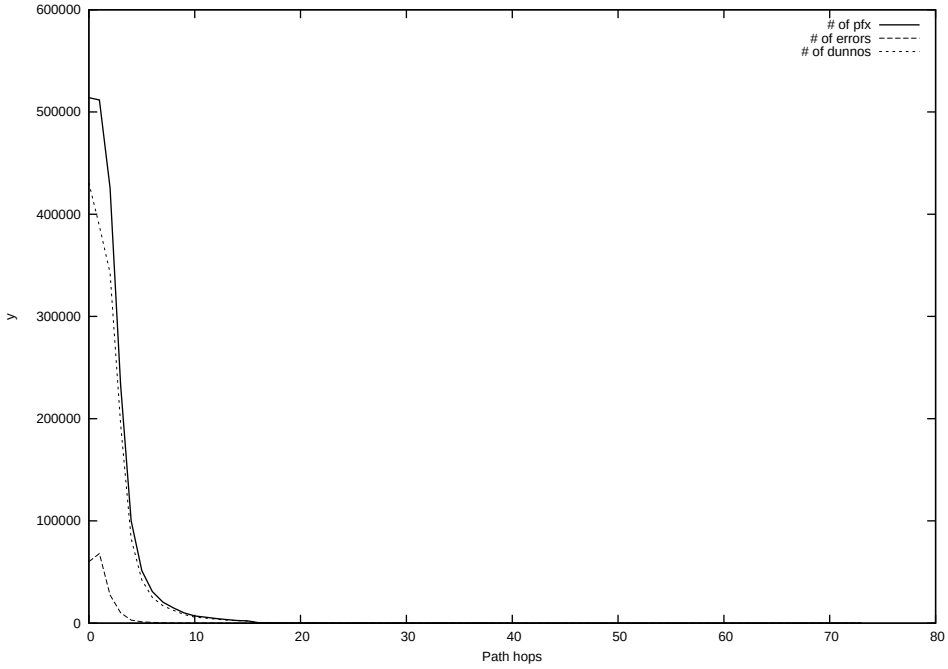
2014-11-13



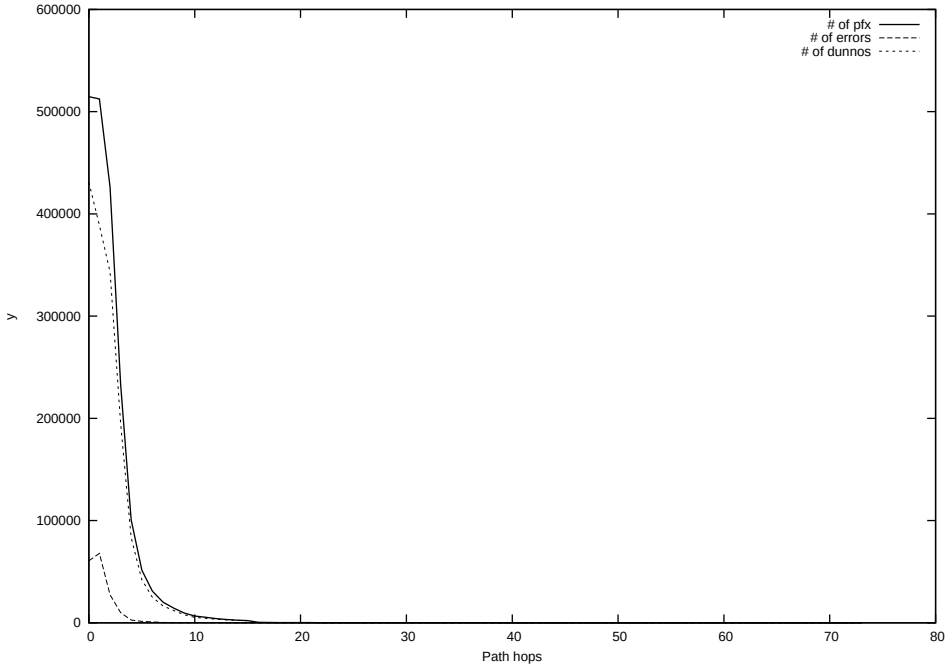
2014-11-14



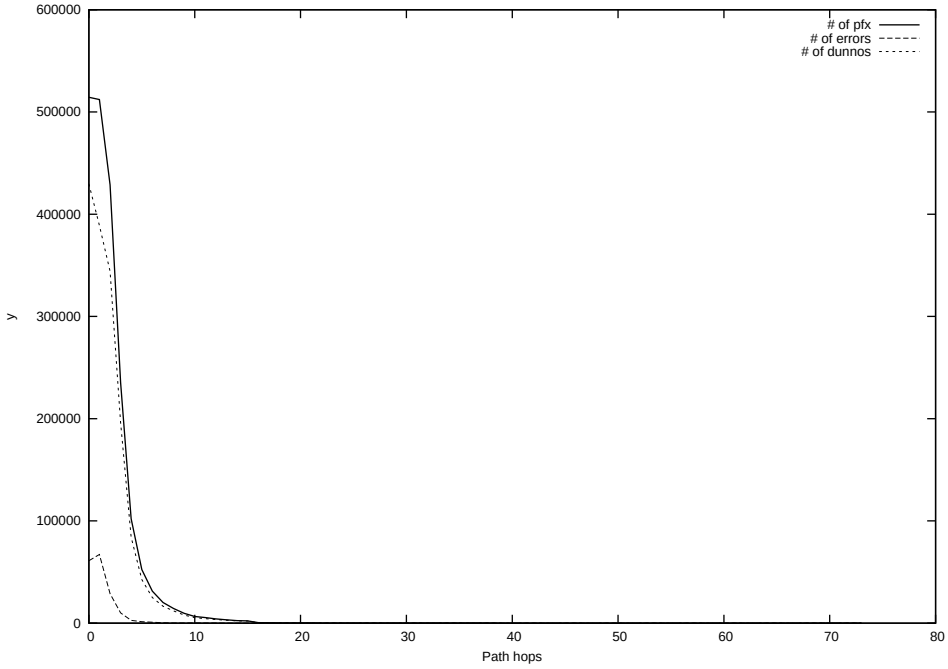
2014-11-15



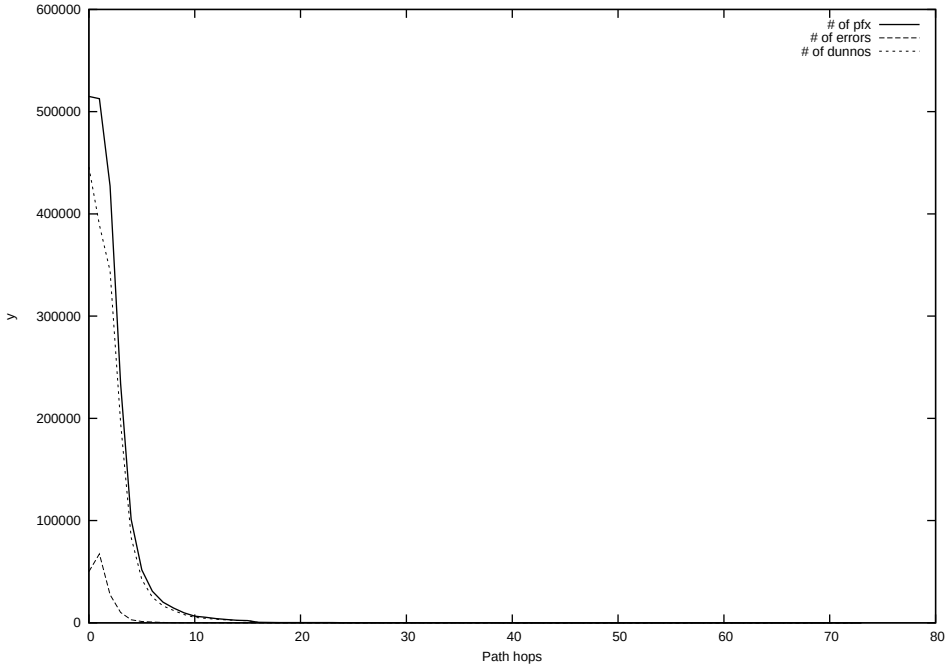
2014-11-16



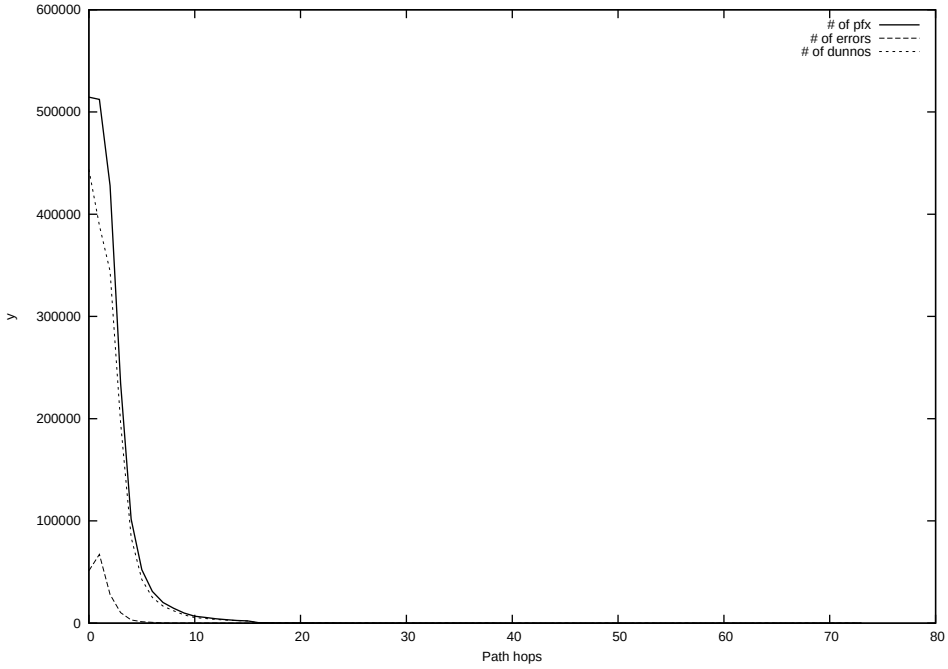
2014-11-17



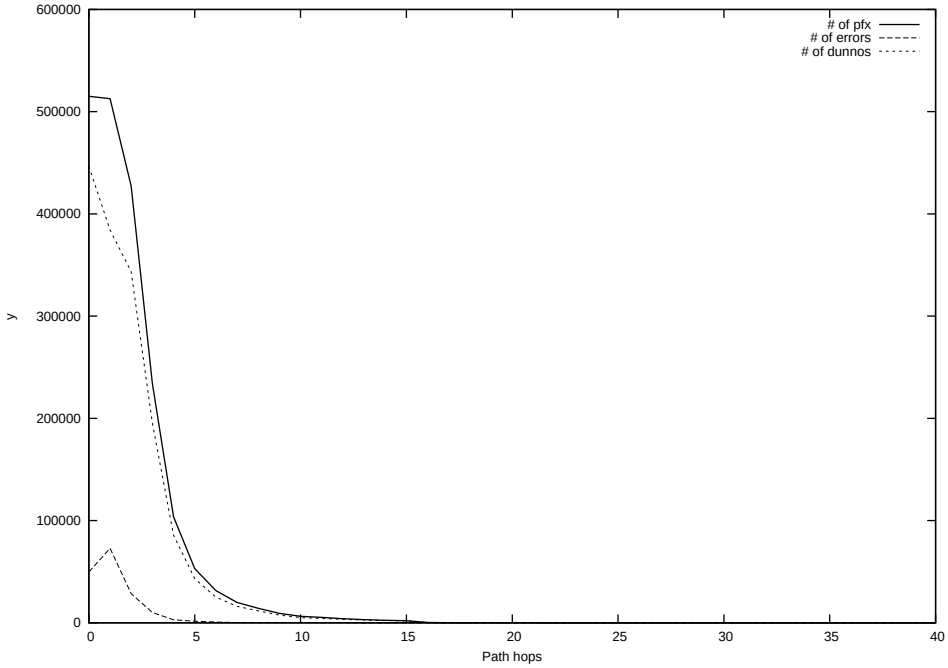
2014-11-18



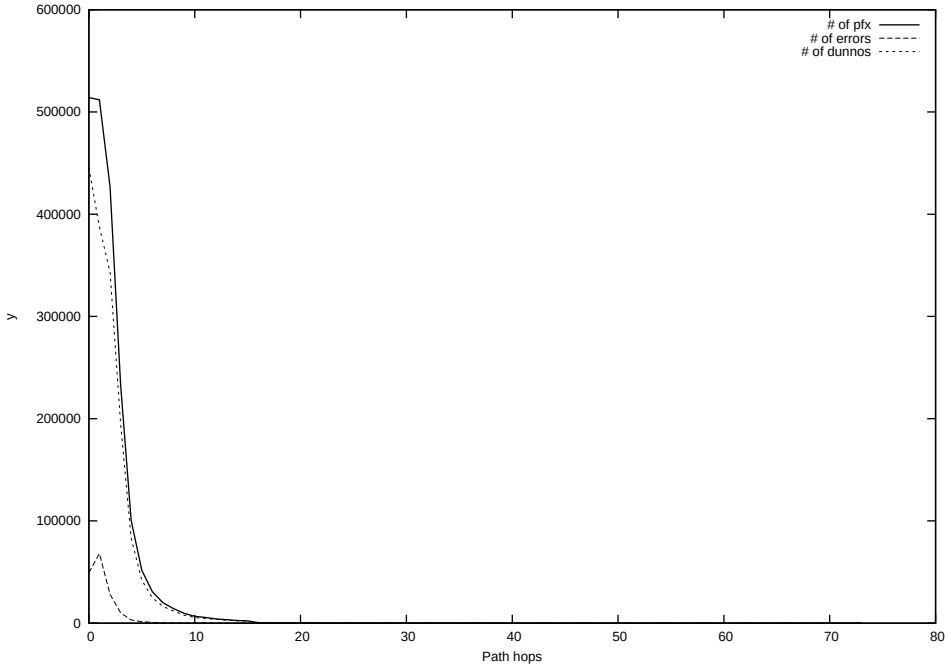
2014-11-19



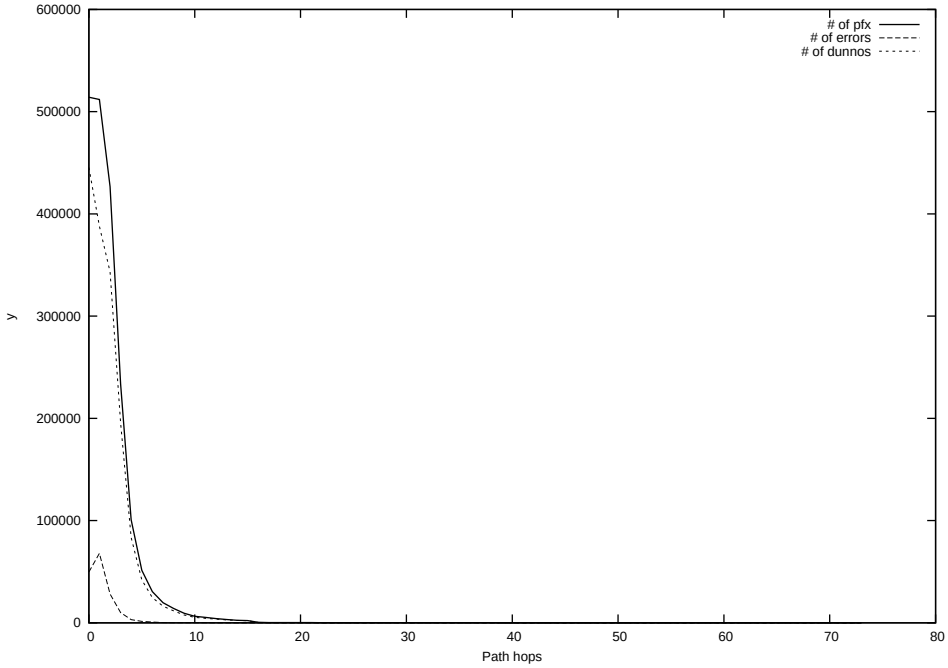
2014-11-20



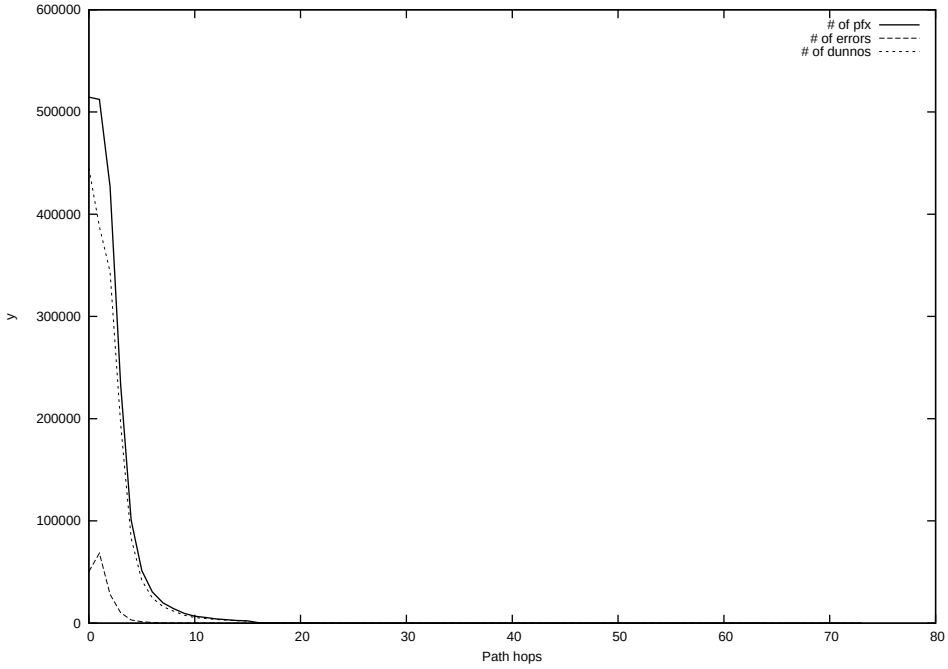
2014-11-21



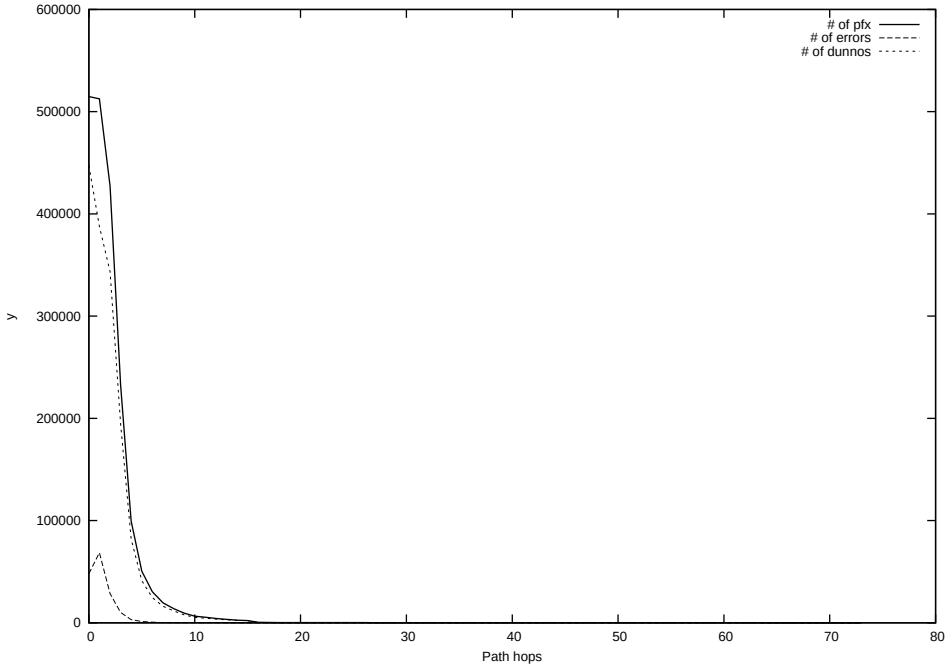
2014-11-22



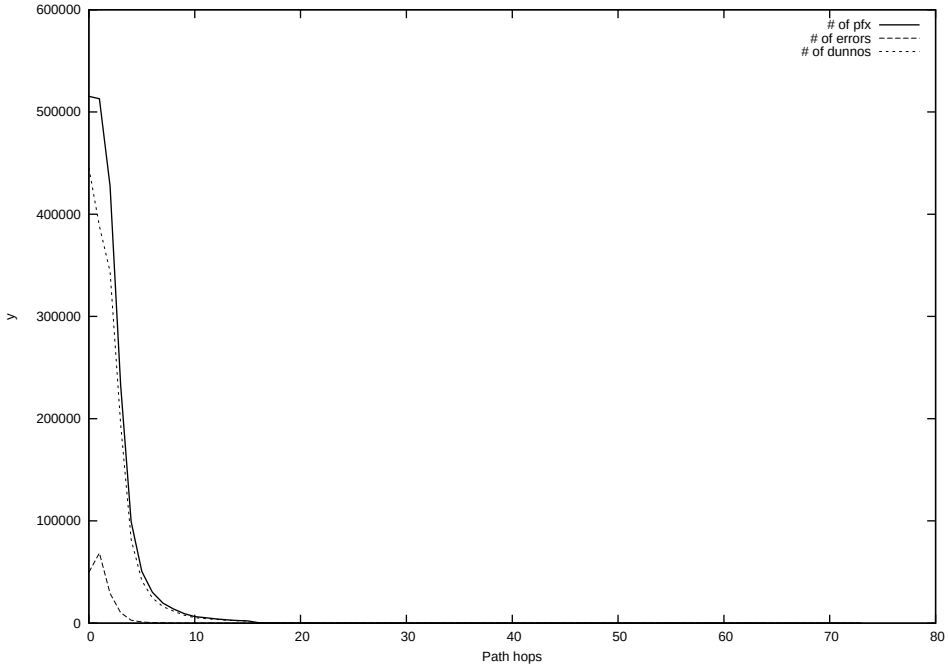
2014-11-23



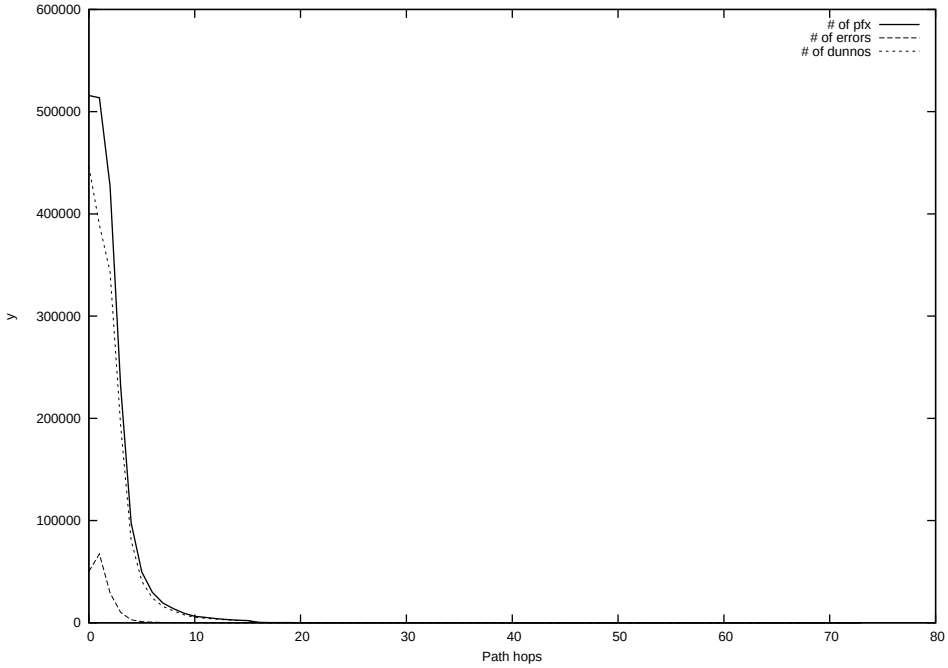
2014-11-24



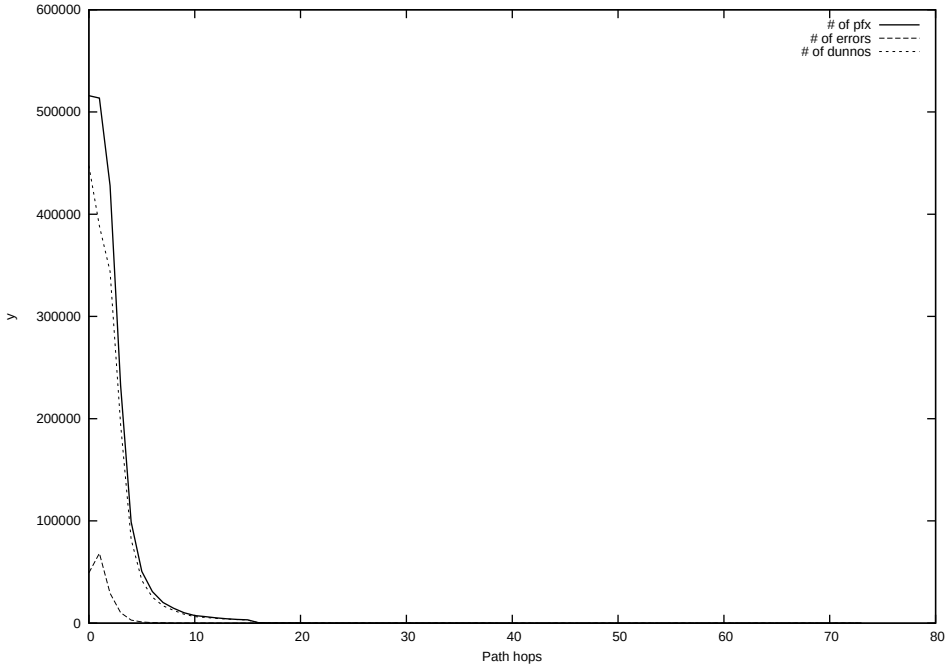
2014-11-25



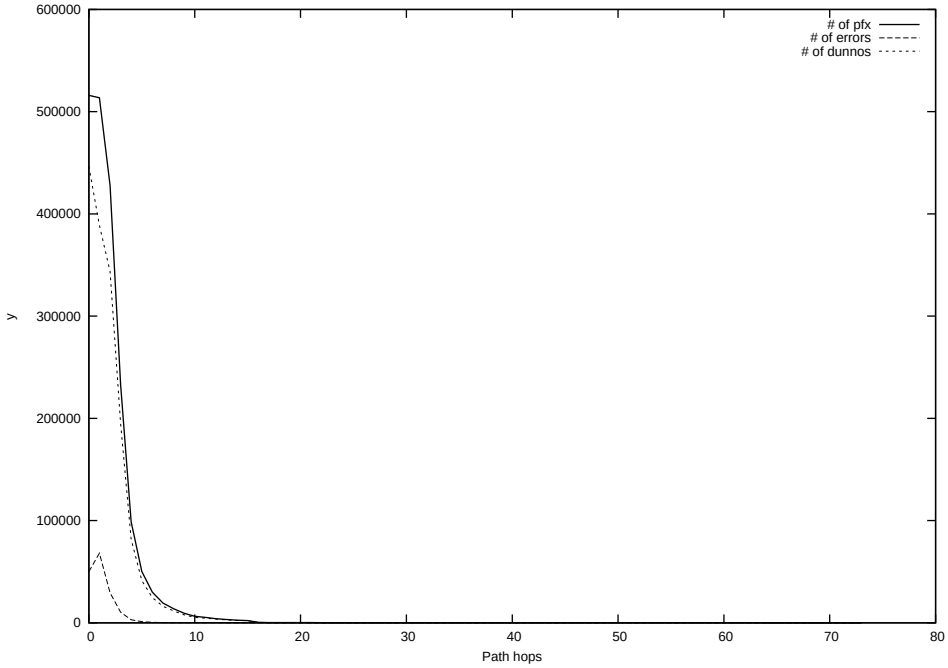
2014-11-26



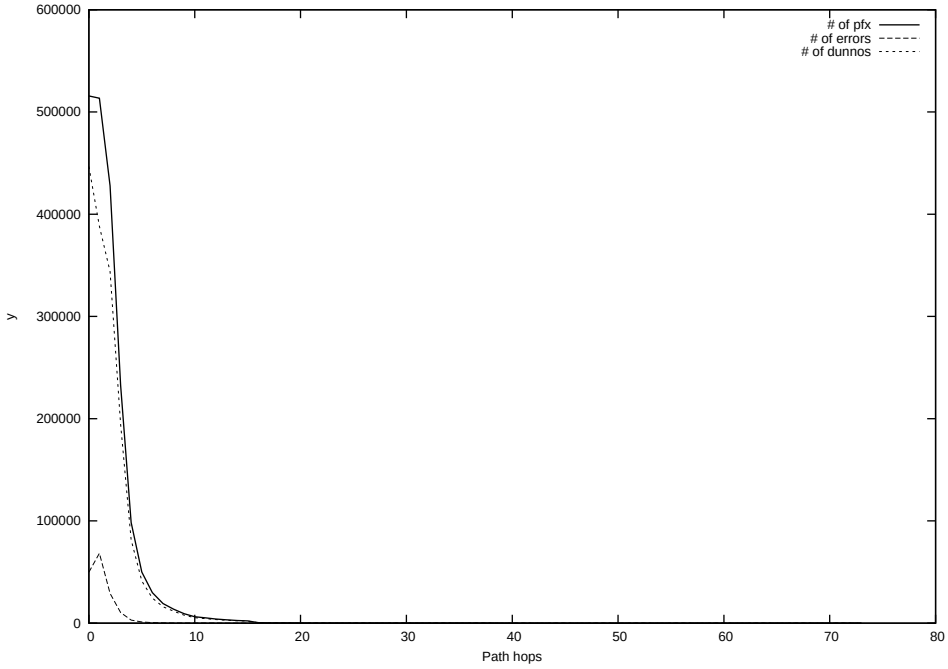
2014-11-27



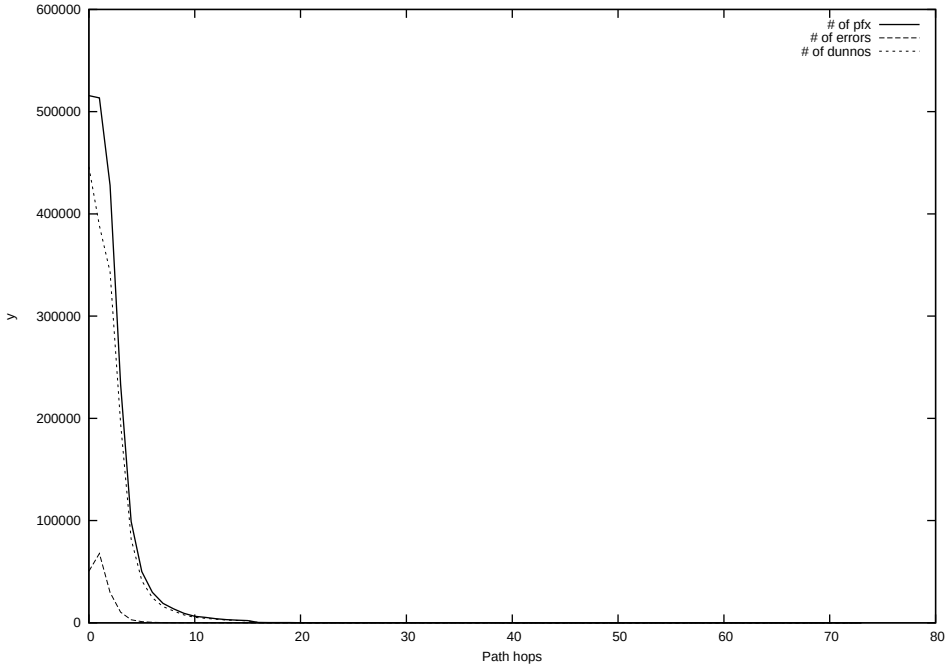
2014-11-28



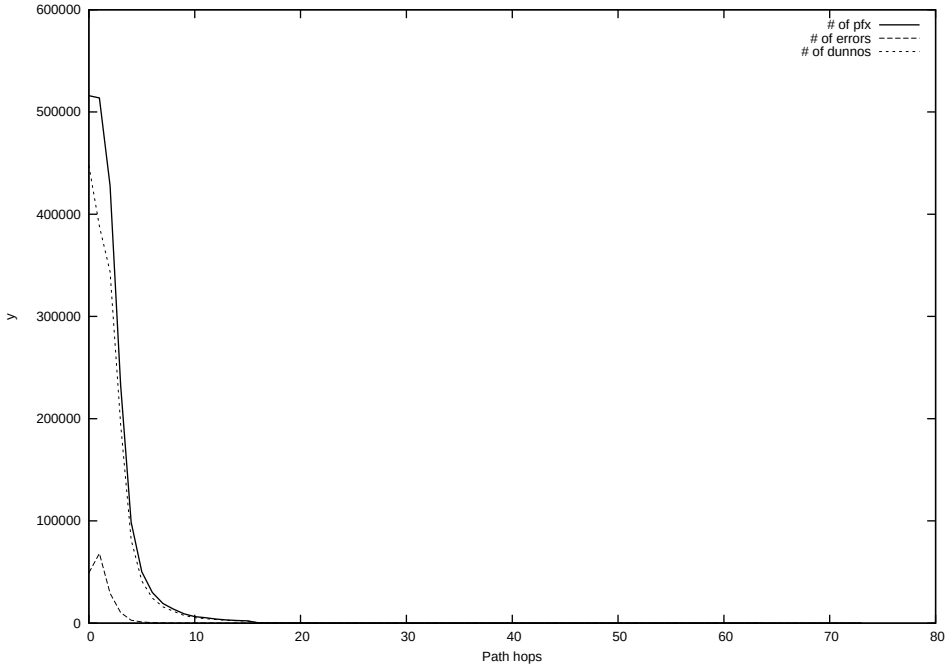
2014-11-29



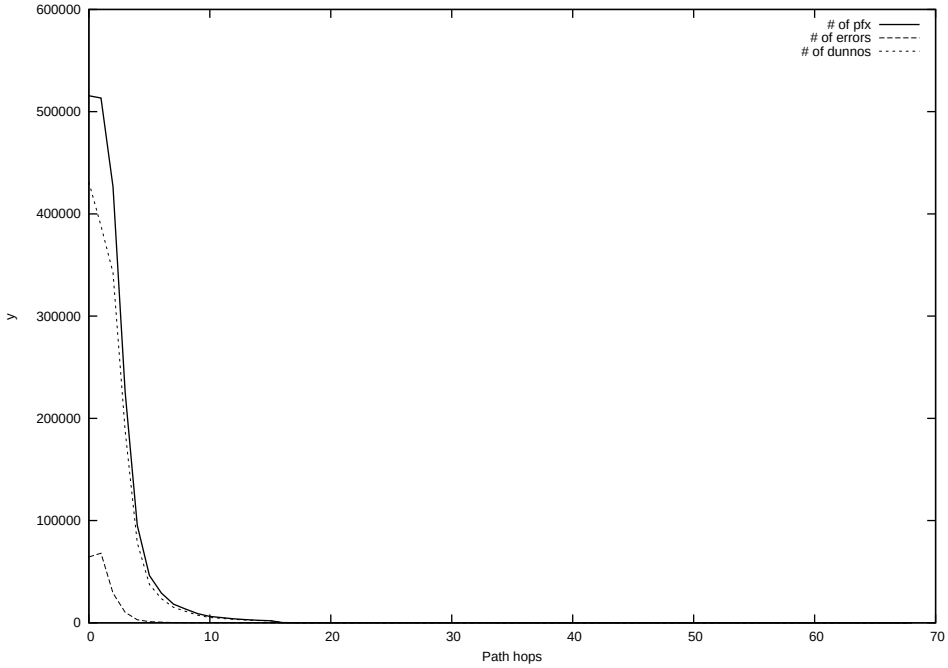
2014-11-30



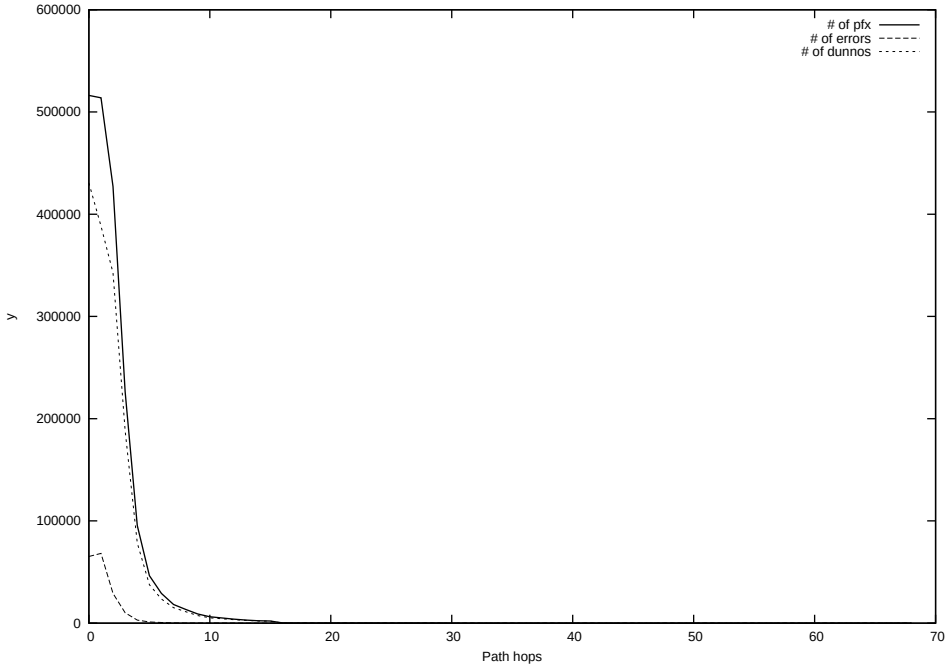
2014-12-01



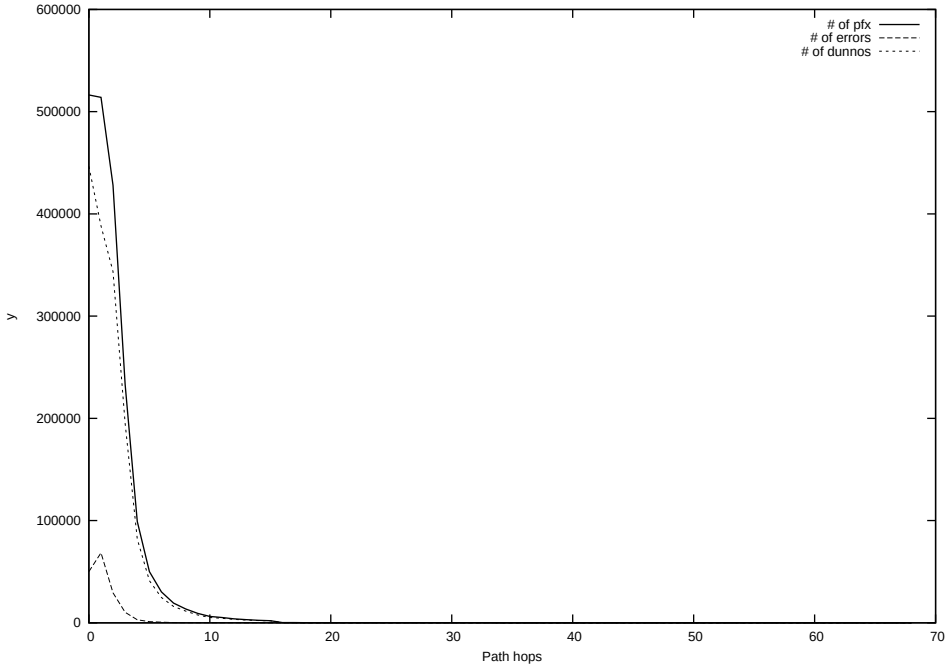
2014-12-02



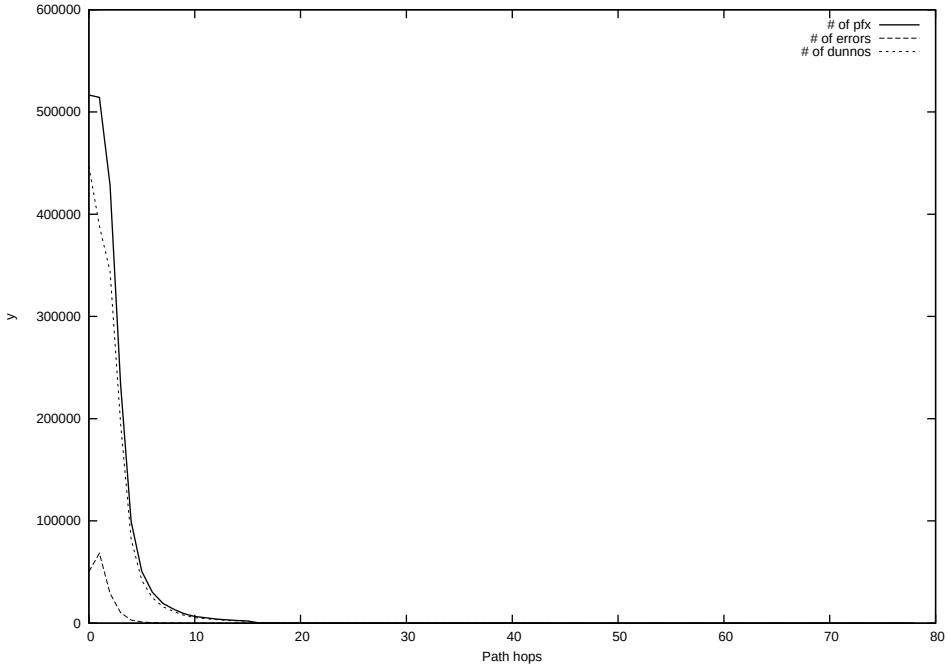
2014-12-03



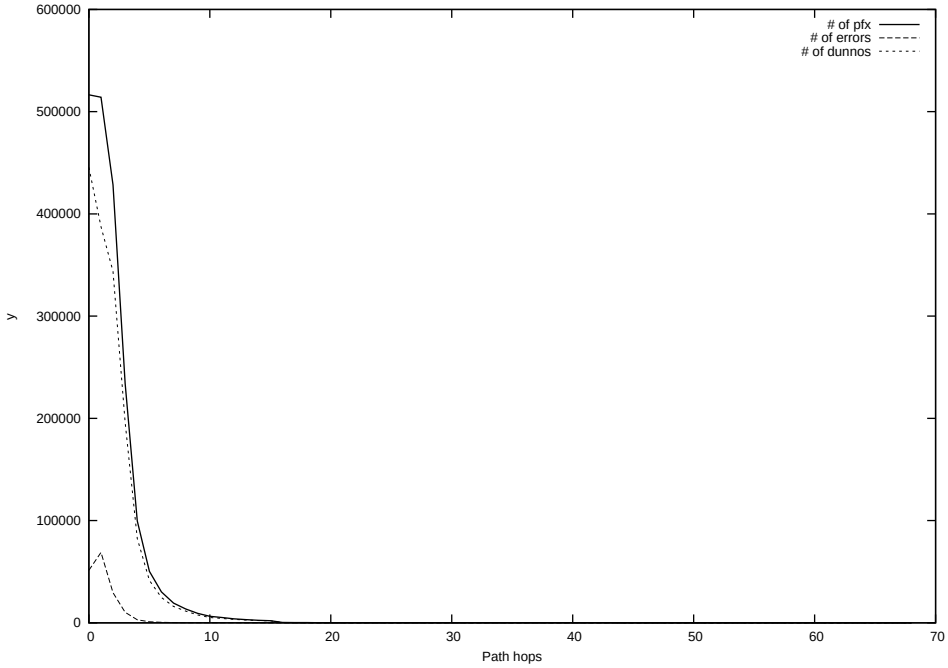
2014-12-04



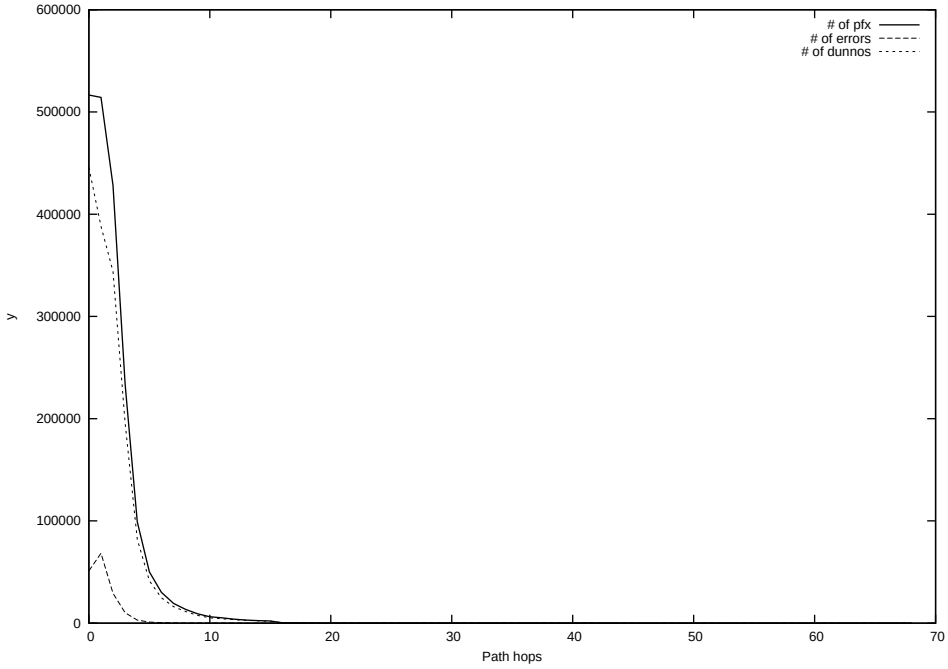
2014-12-05



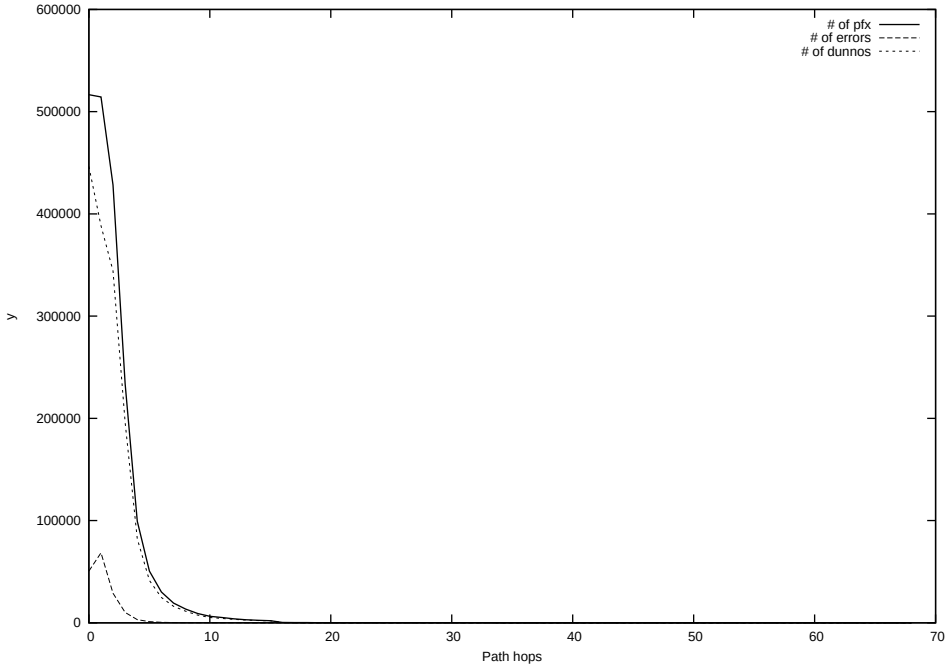
2014-12-06



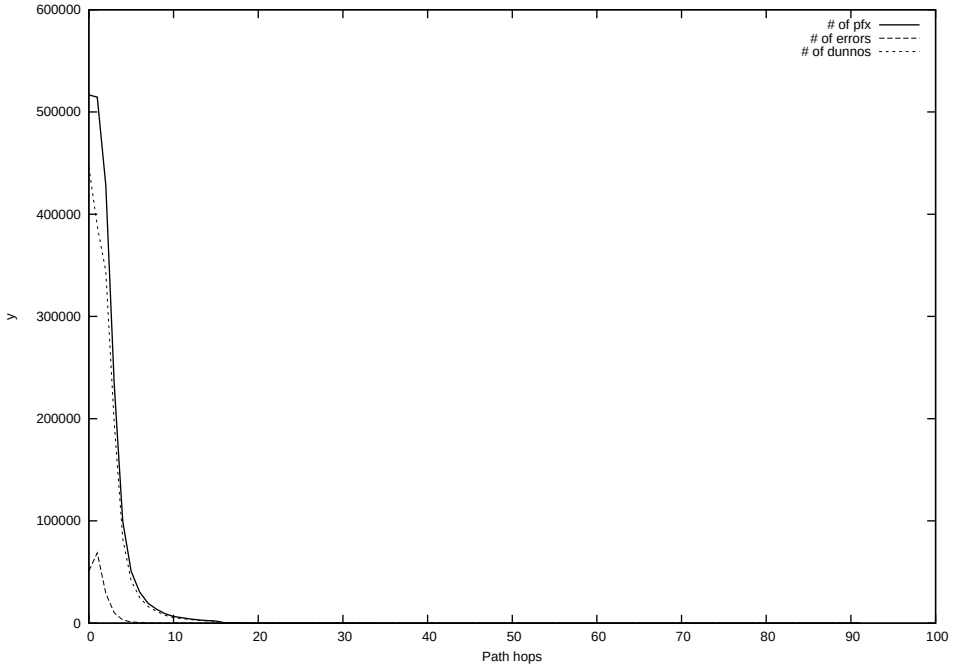
2014-12-07



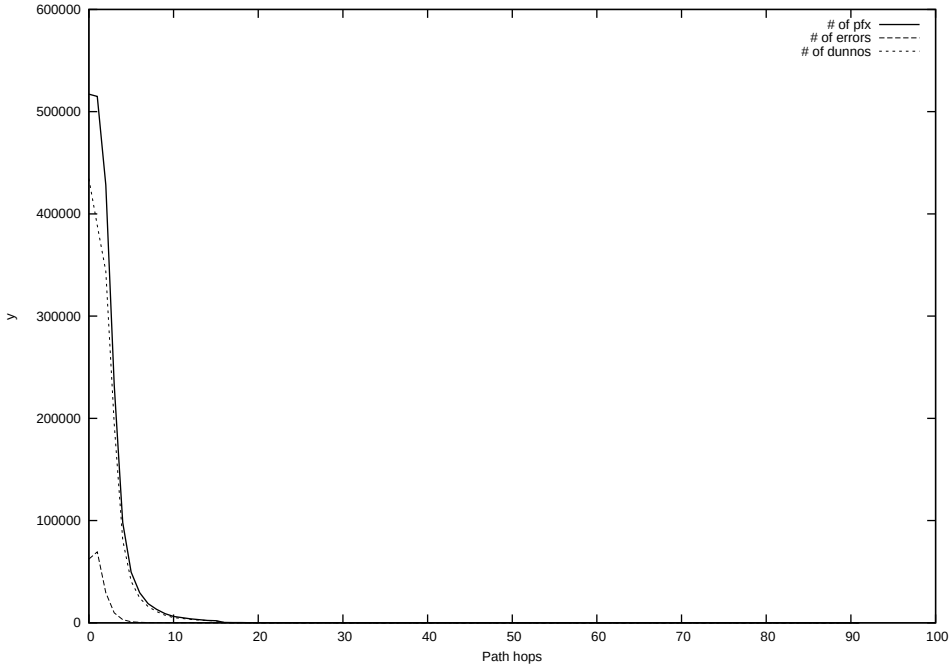
2014-12-08



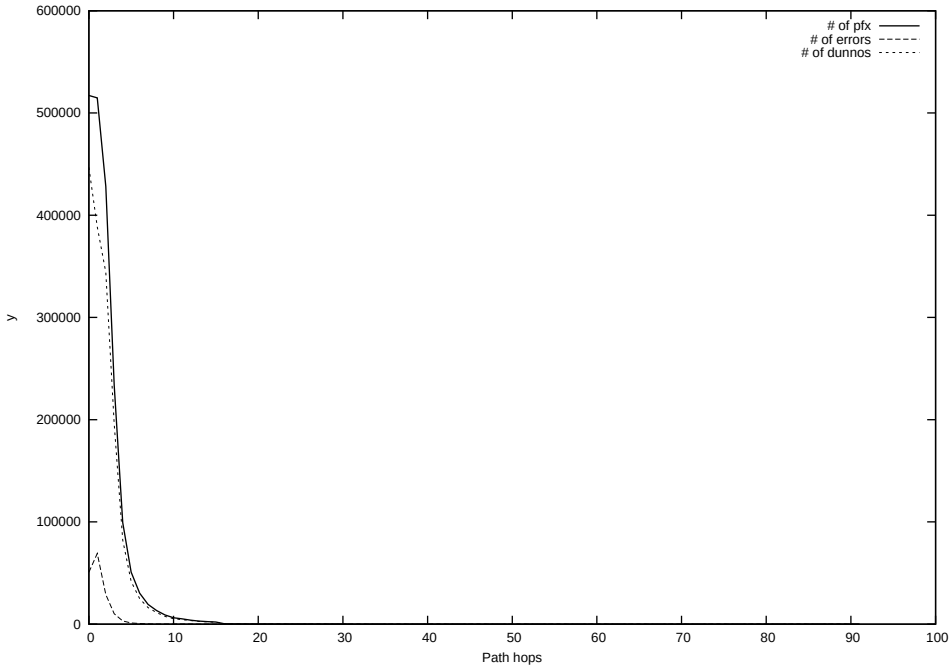
2014-12-09



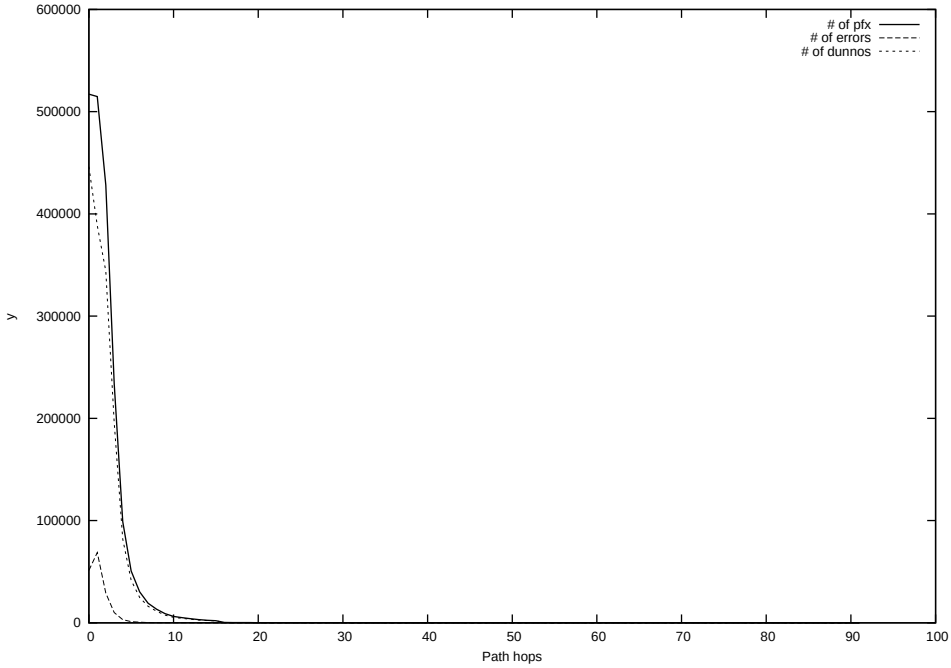
2014-12-10



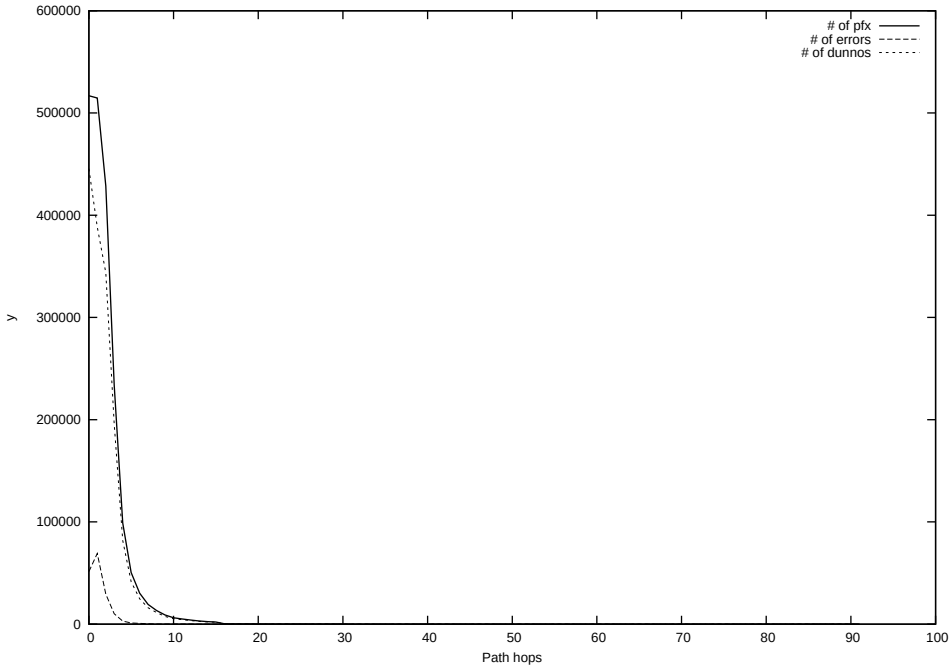
2014-12-11



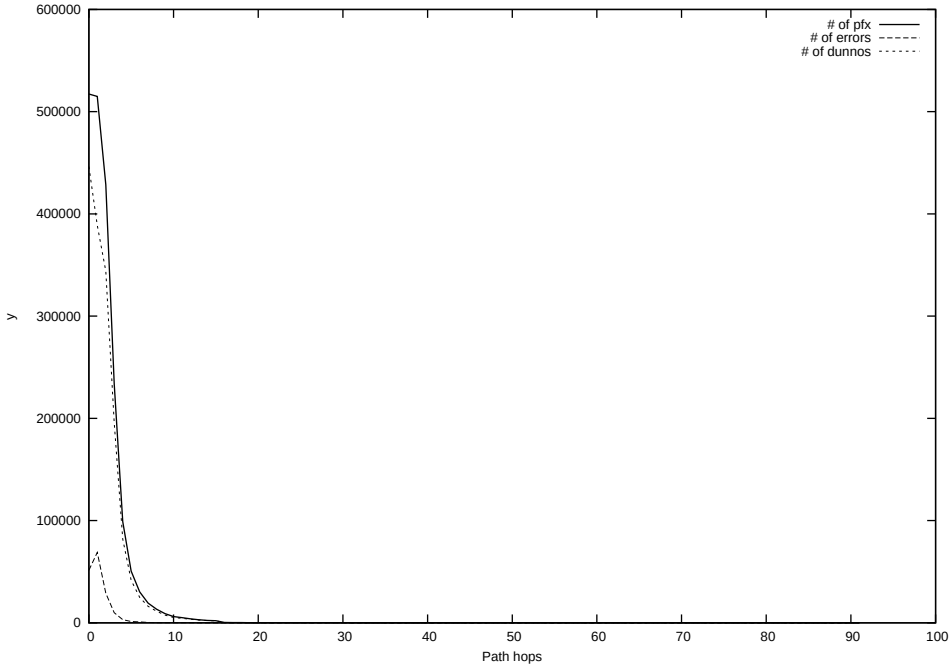
2014-12-12



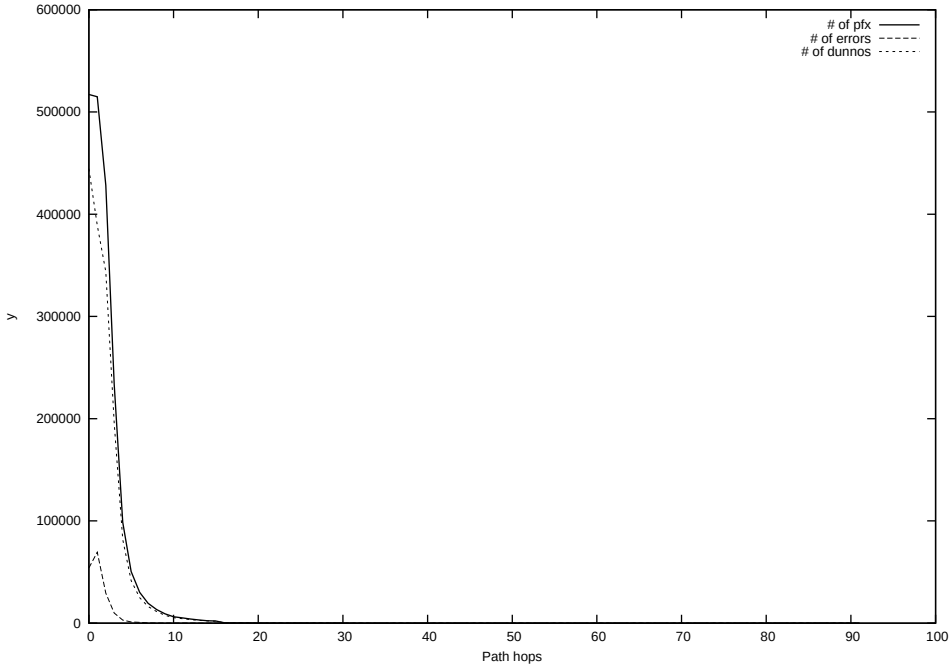
2014-12-13



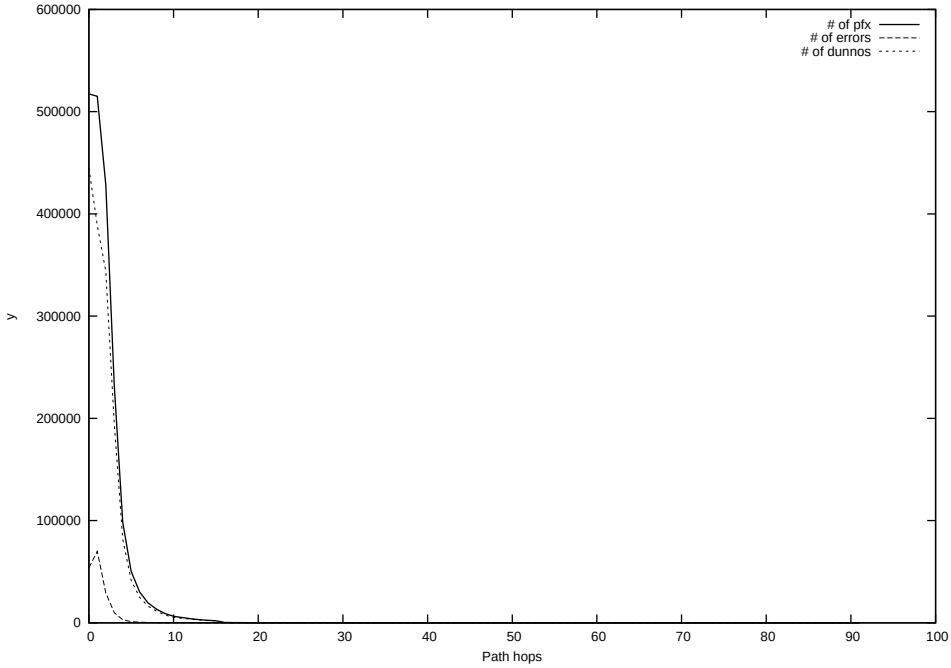
2014-12-14



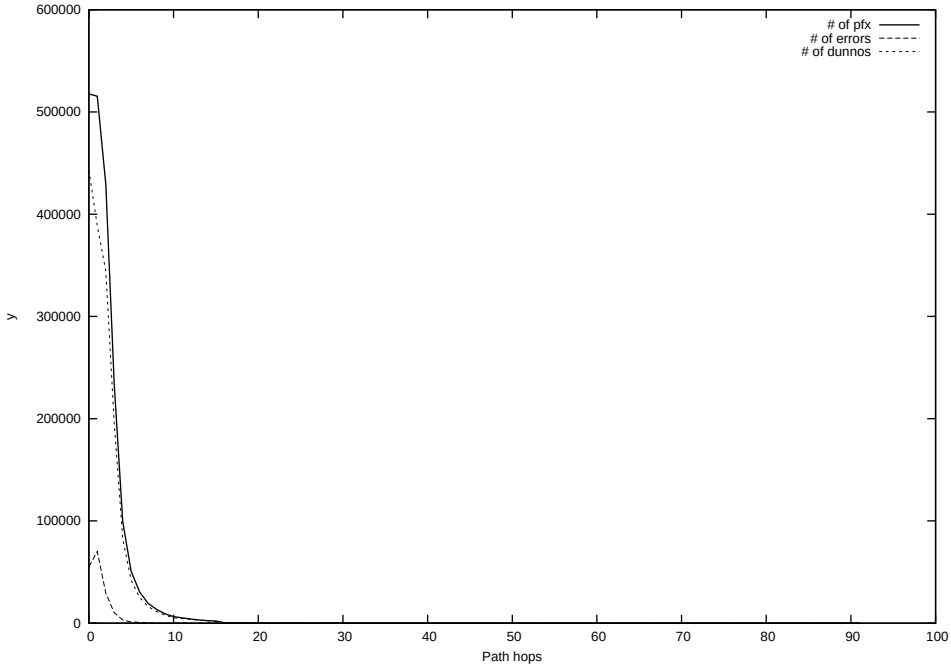
2014-12-15



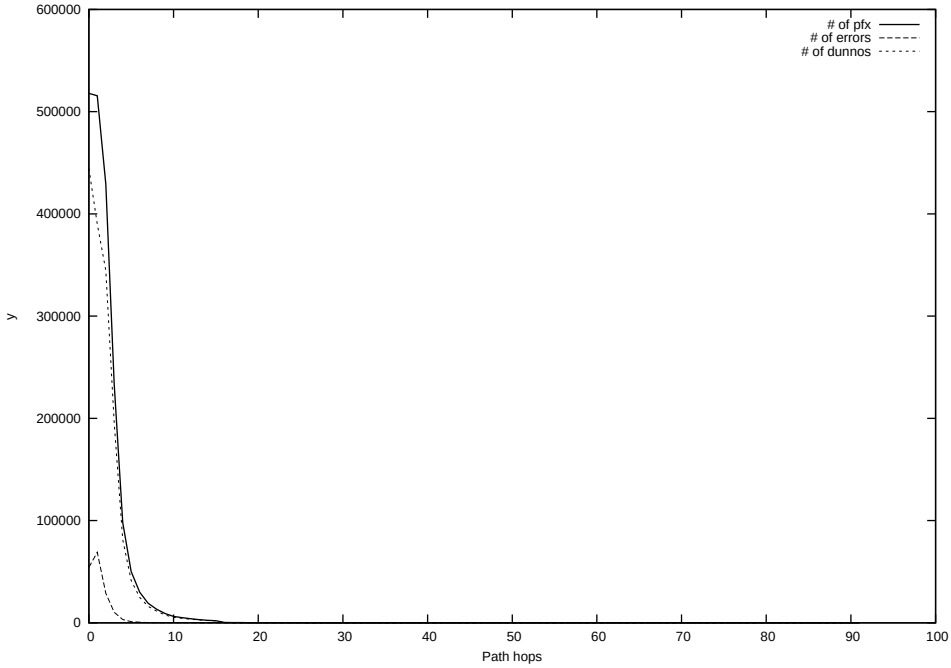
2014-12-16



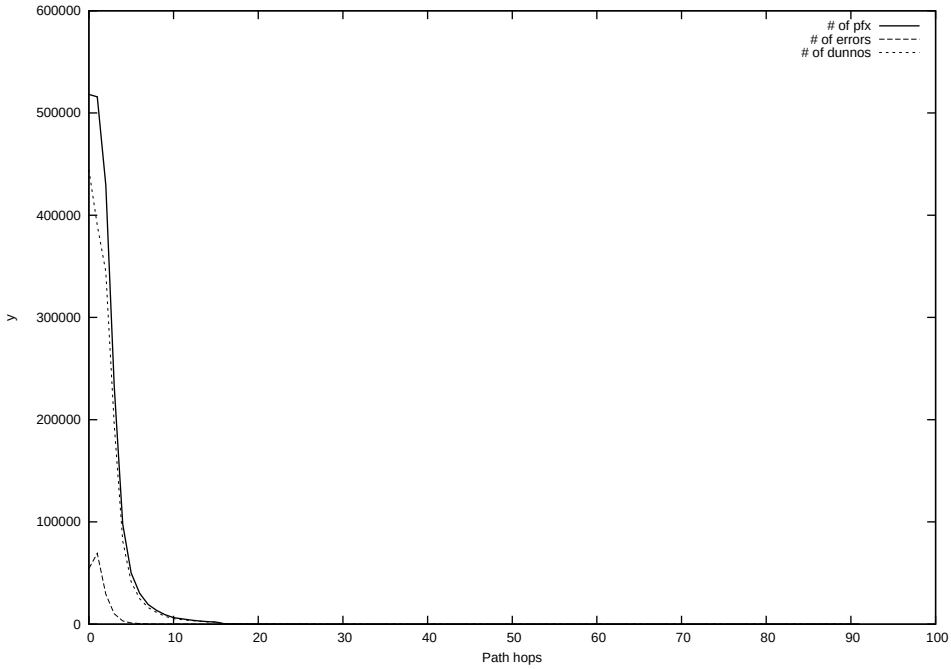
2014-12-17



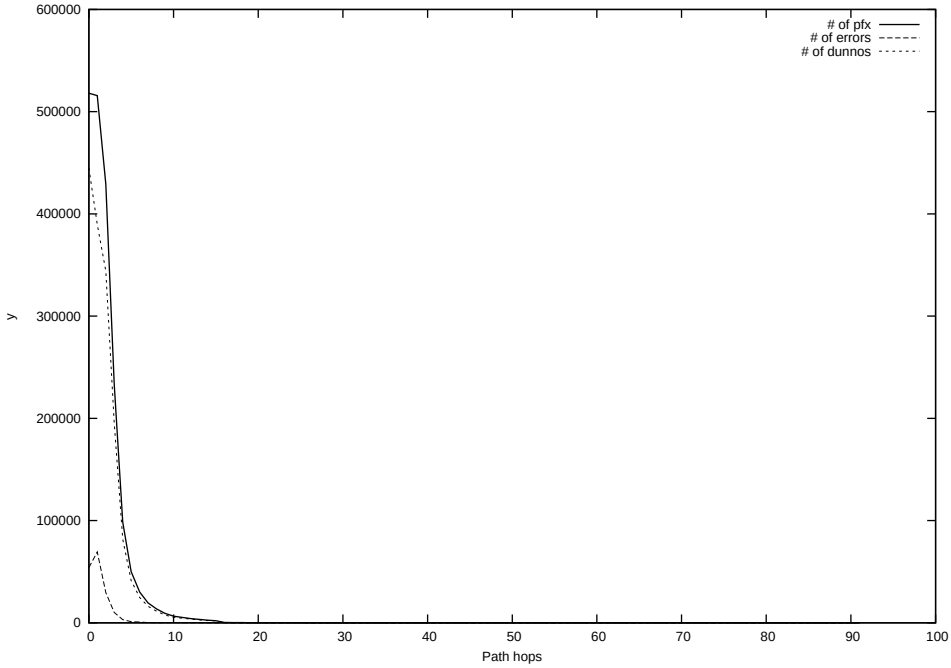
2014-12-18



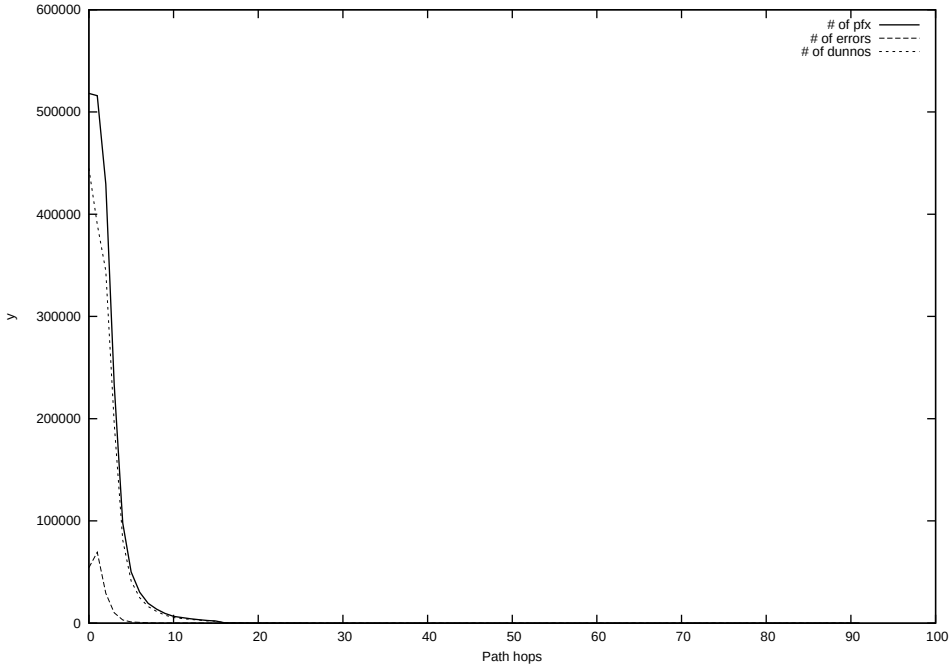
2014-12-19



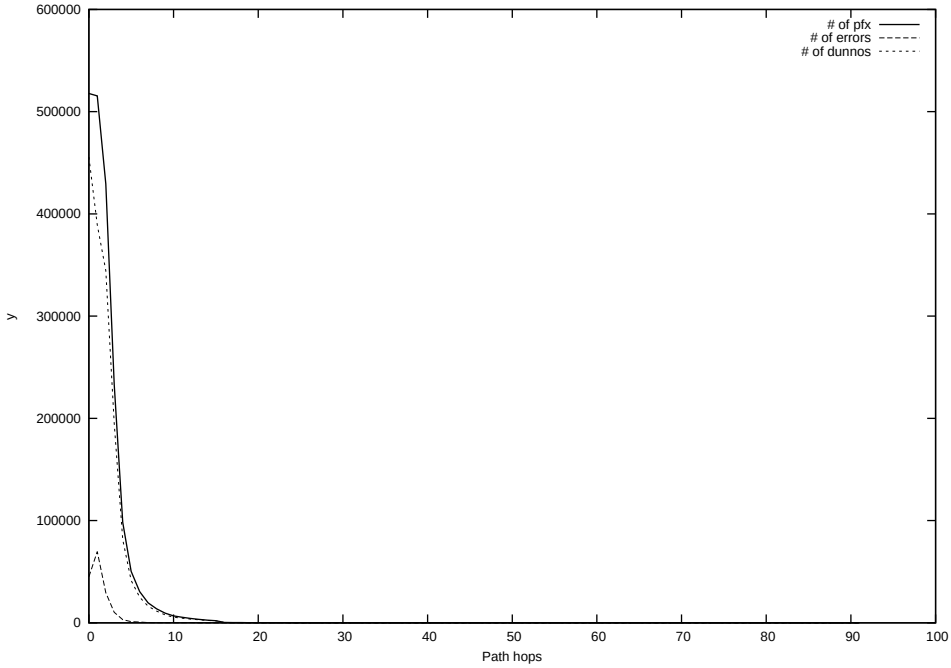
2014-12-20



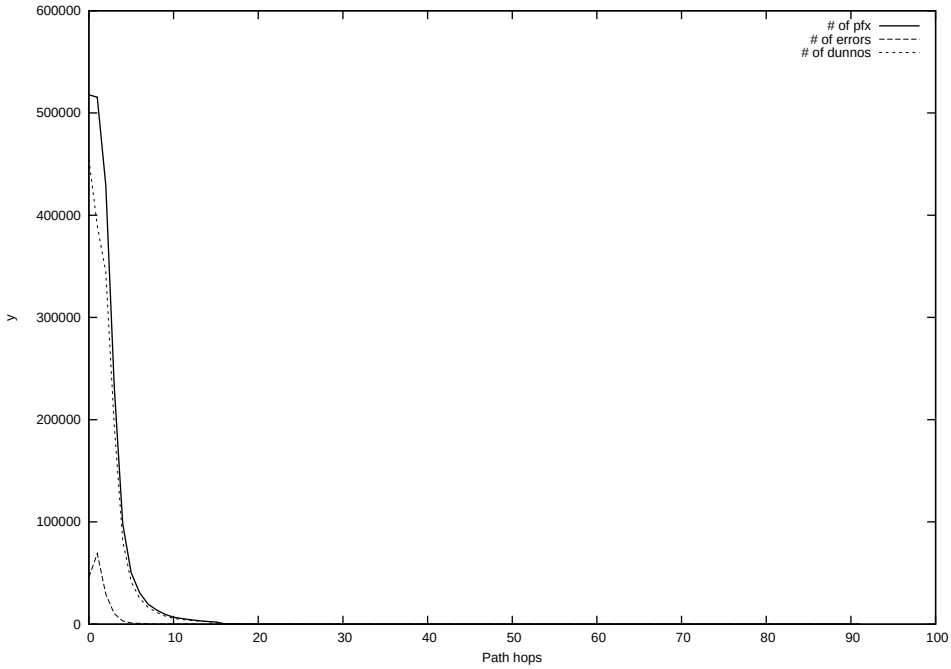
2014-12-21



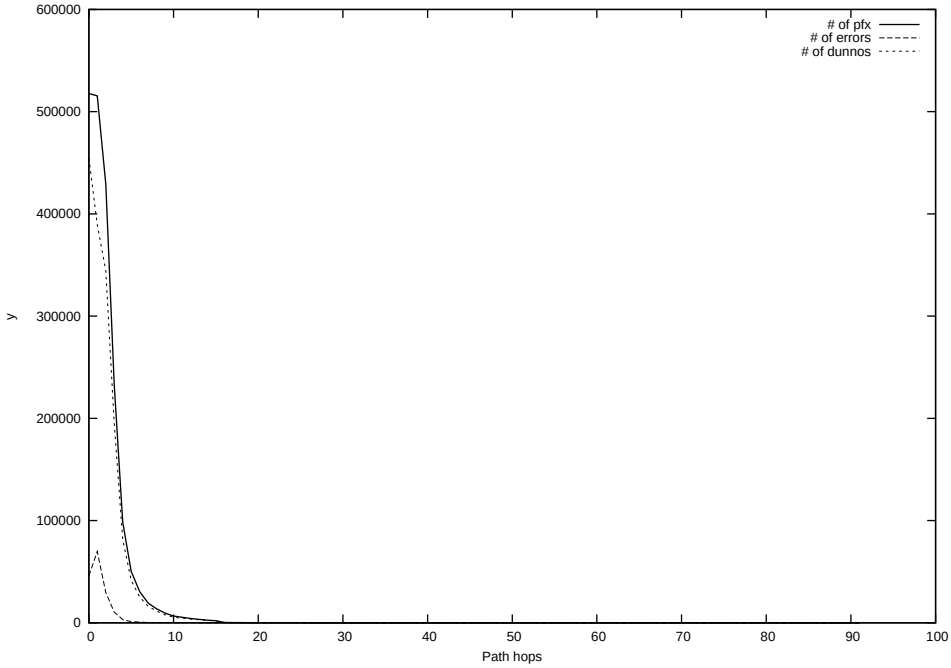
2014-12-22



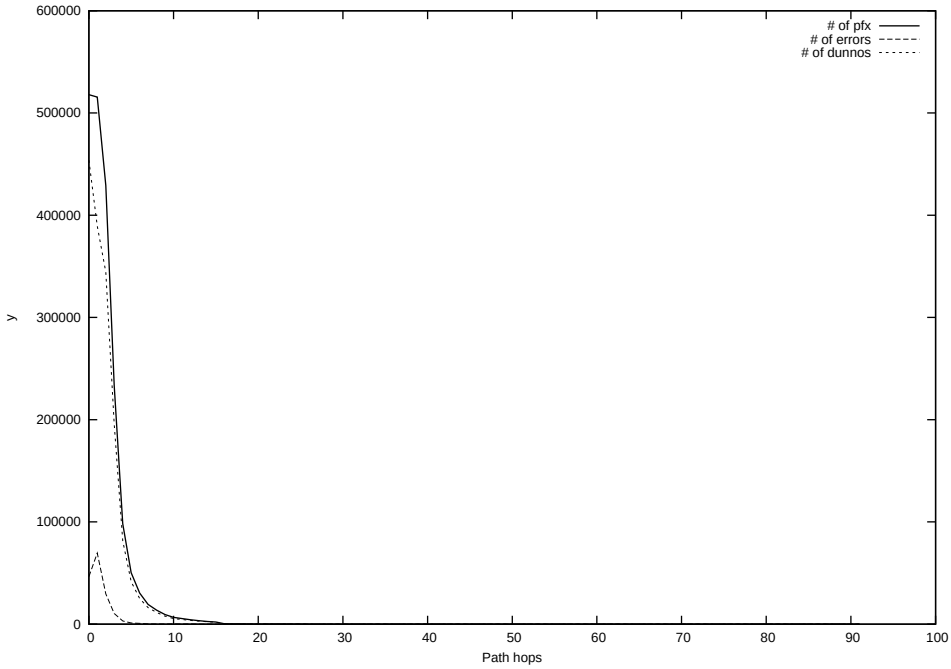
2014-12-23



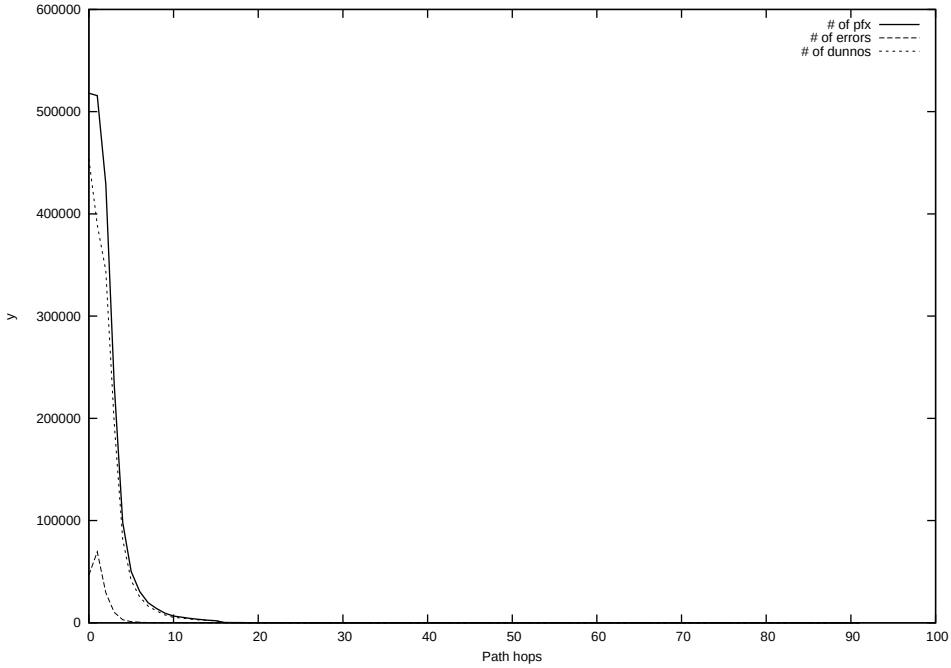
2014-12-24



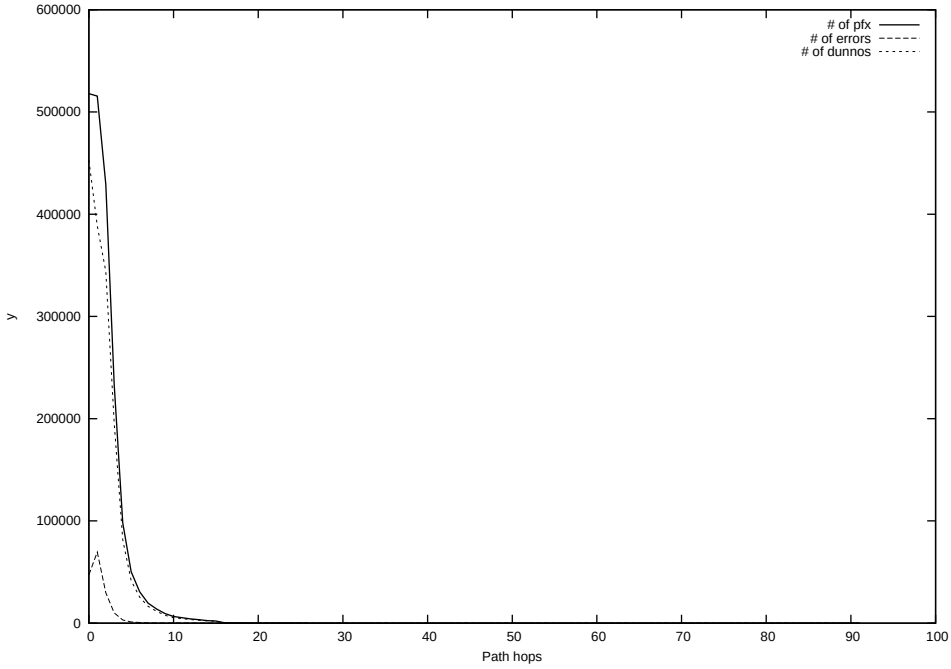
2014-12-25



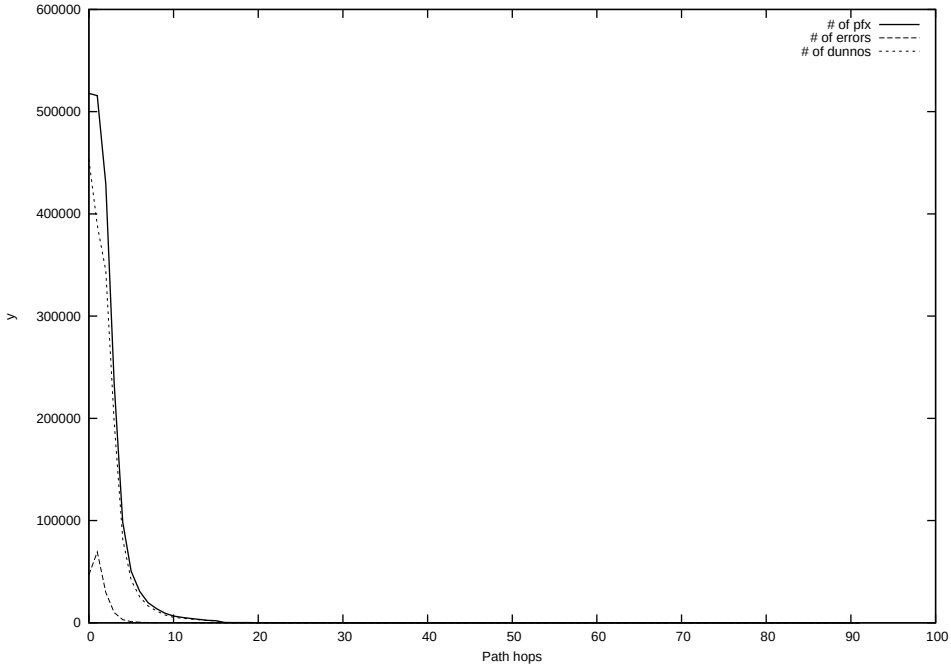
2014-12-26



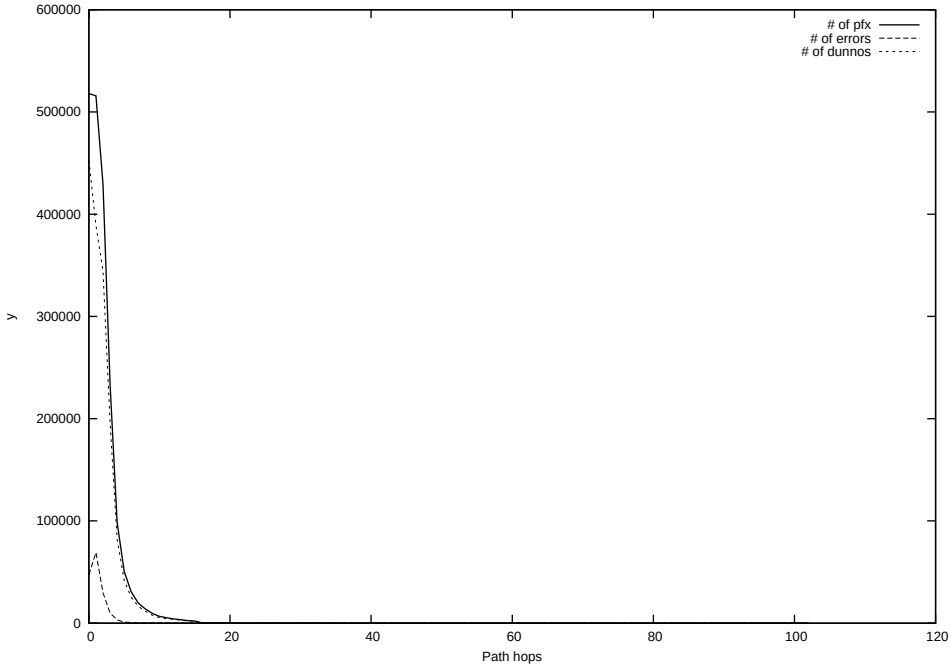
2014-12-27



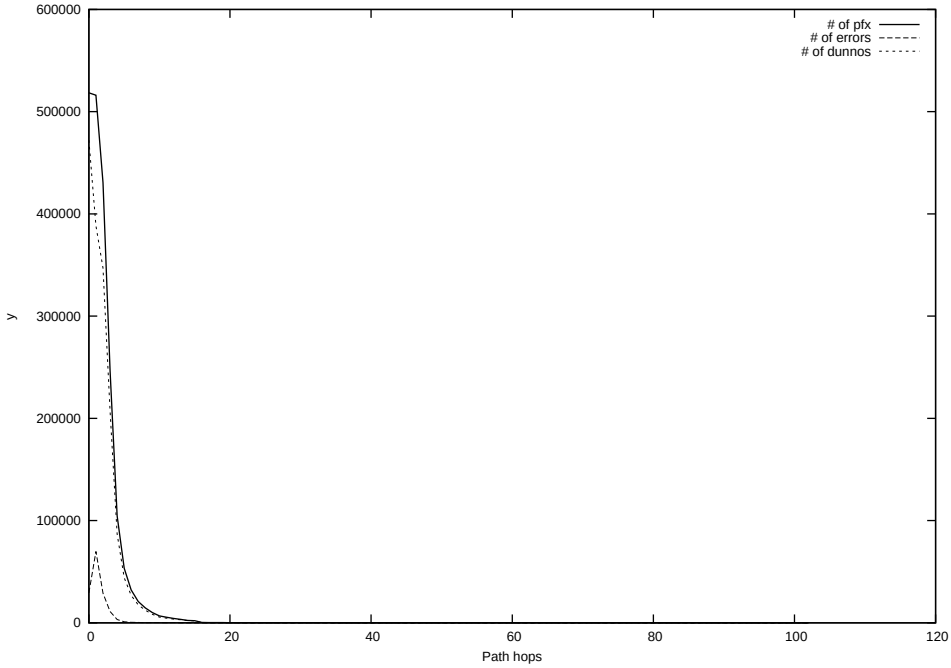
2014-12-28



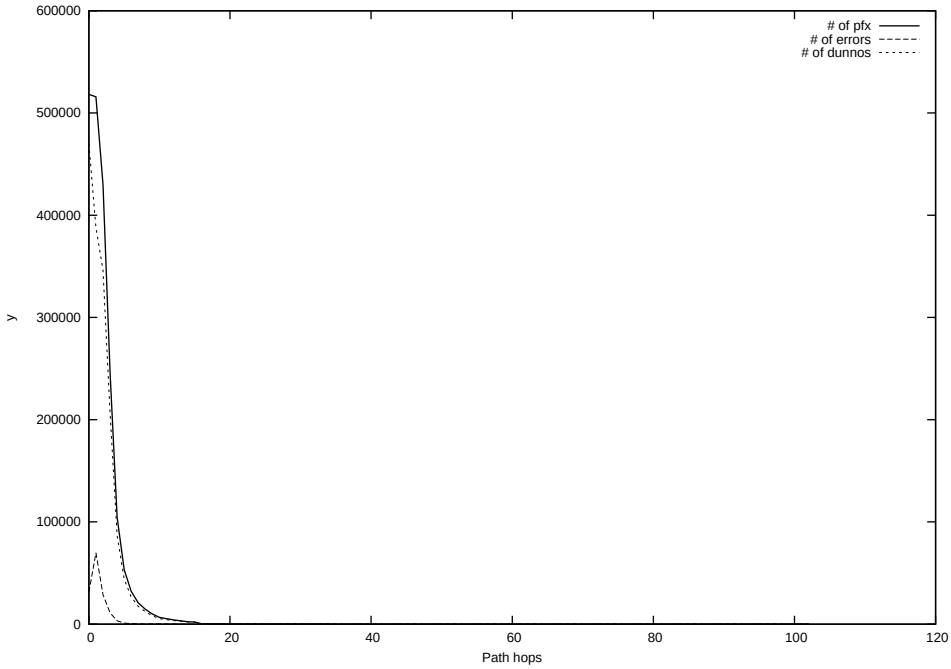
2014-12-29



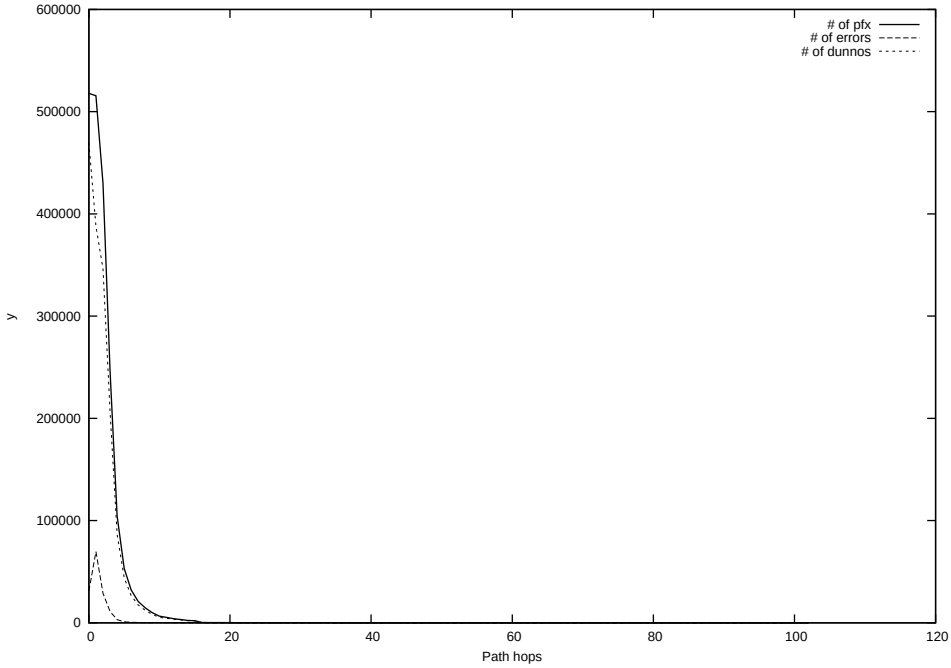
2014-12-30



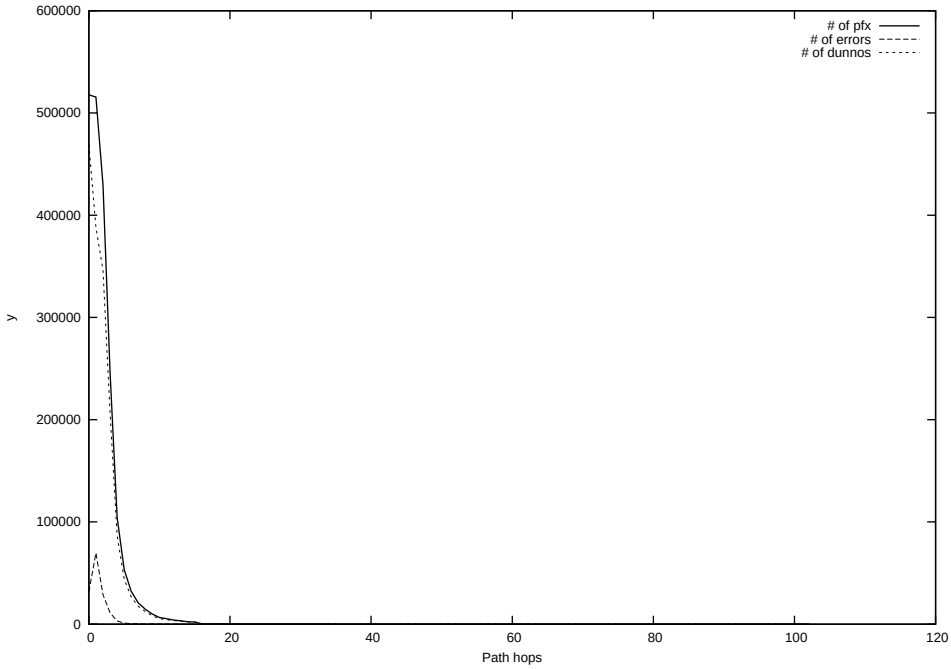
2014-12-31



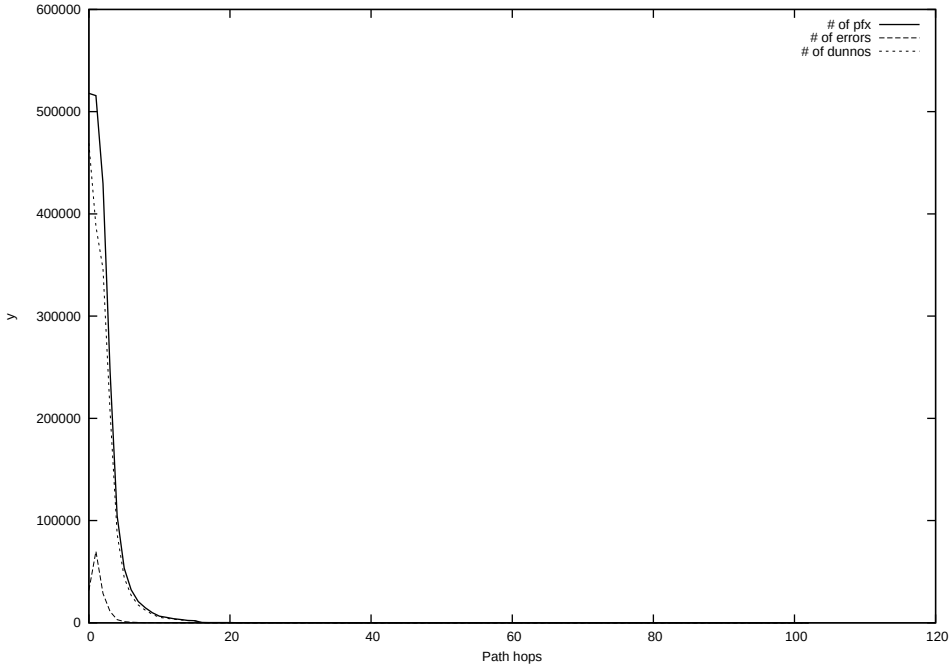
2015-01-01



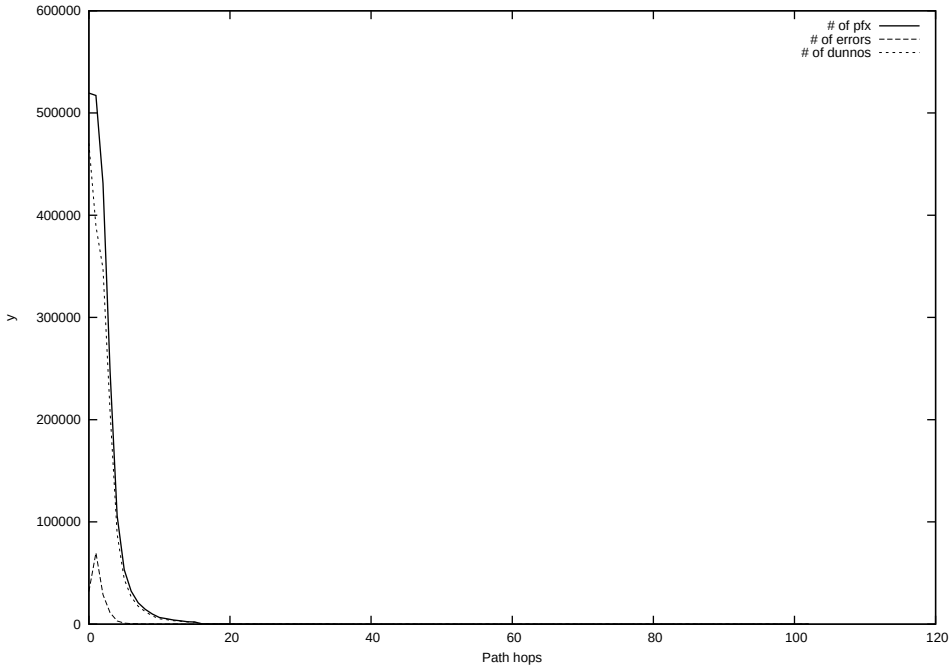
2015-01-02



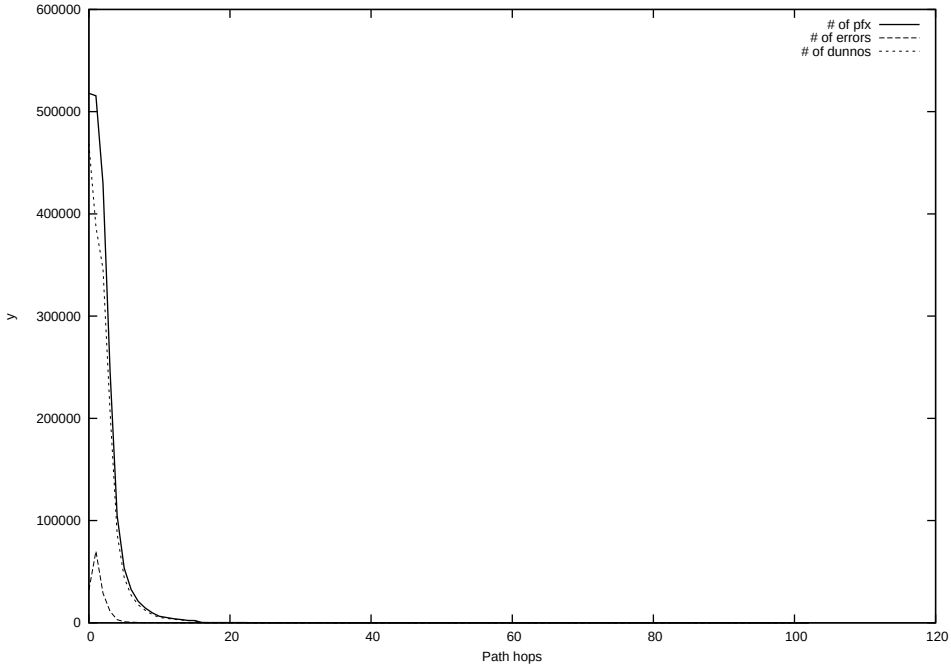
2015-01-03



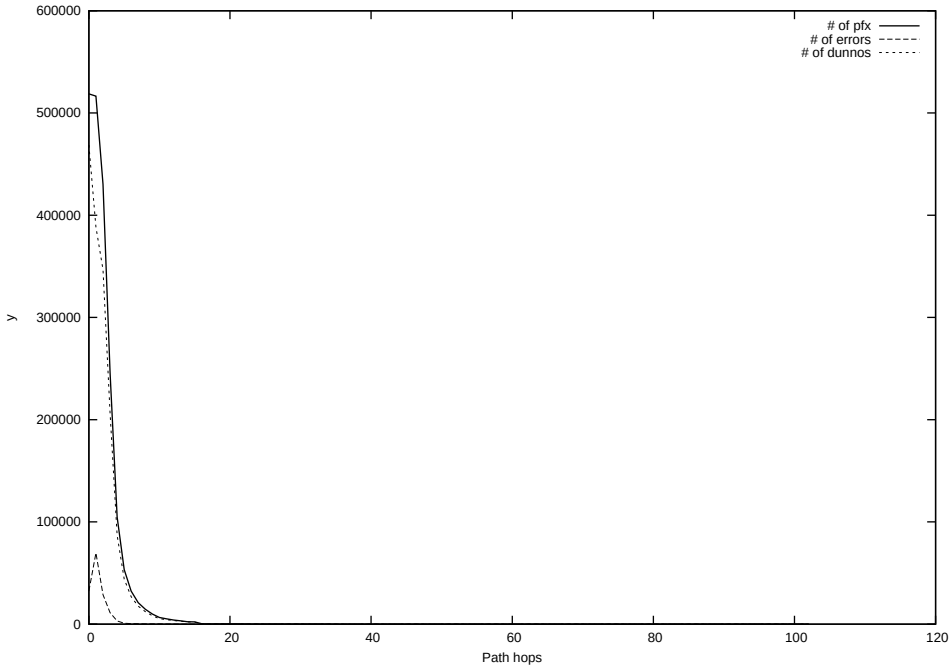
2015-01-04



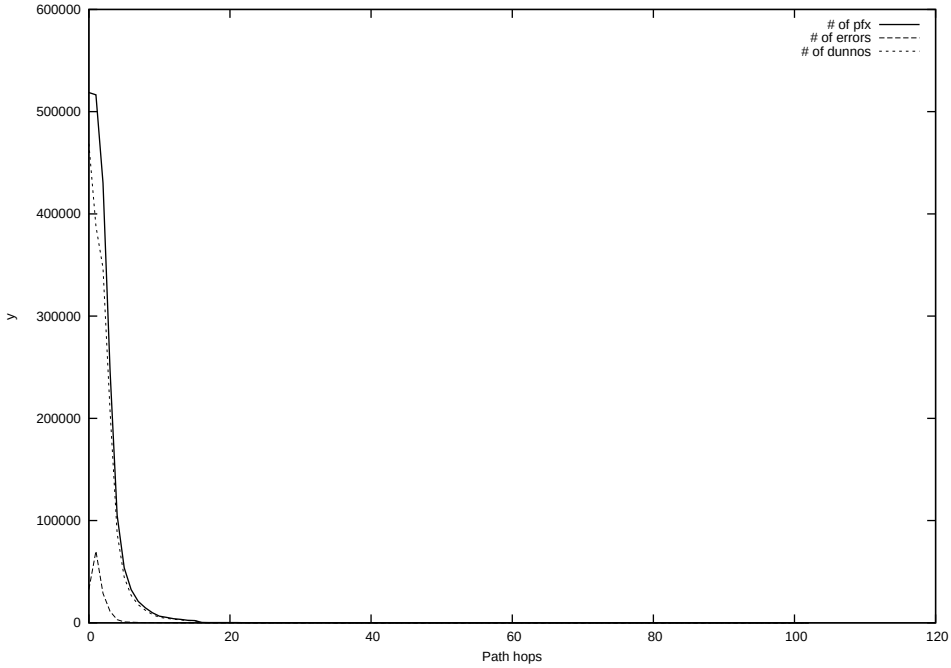
2015-01-05



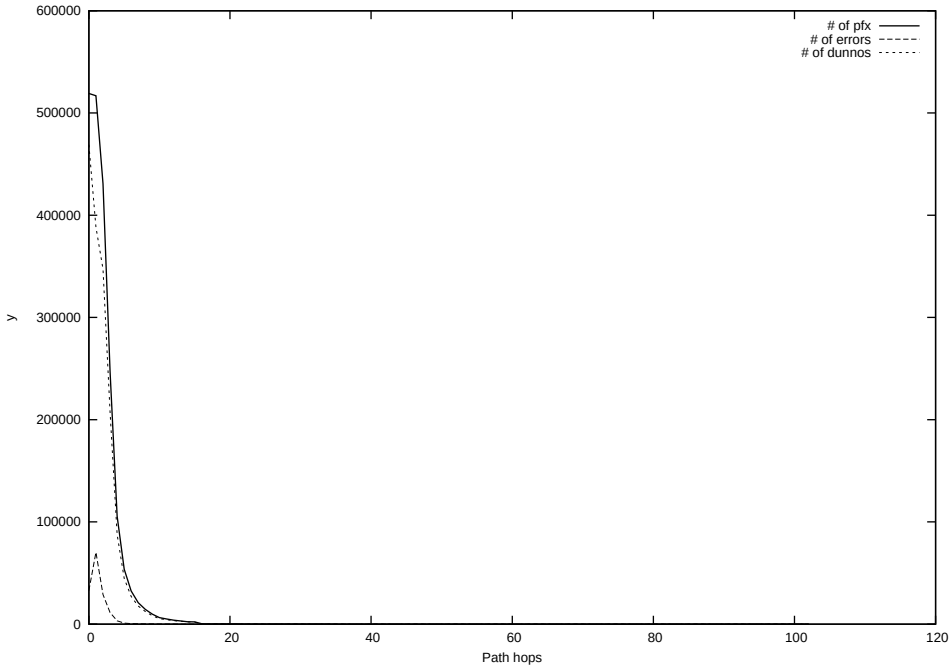
2015-01-06



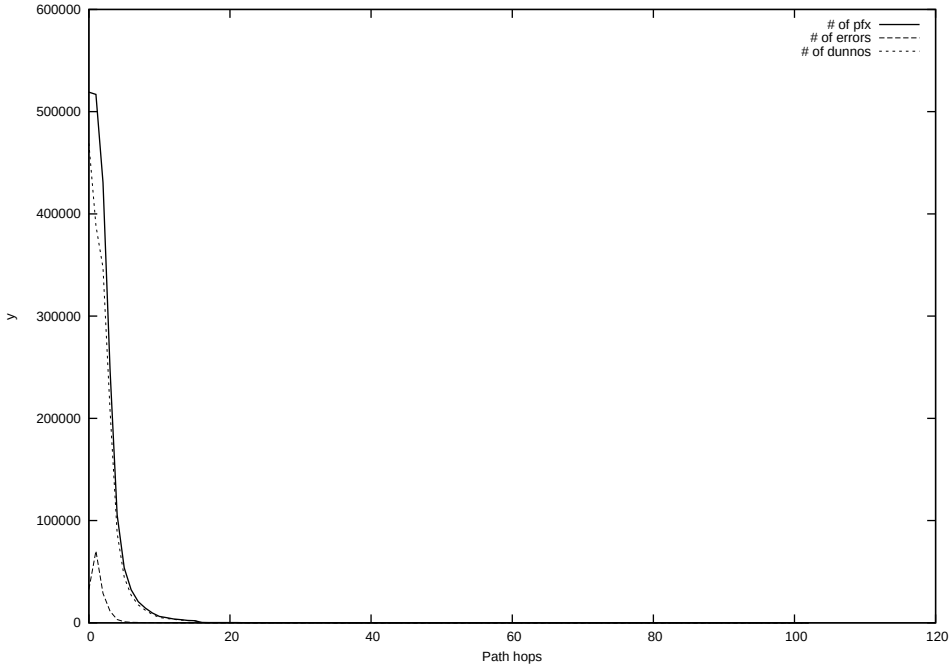
2015-01-07



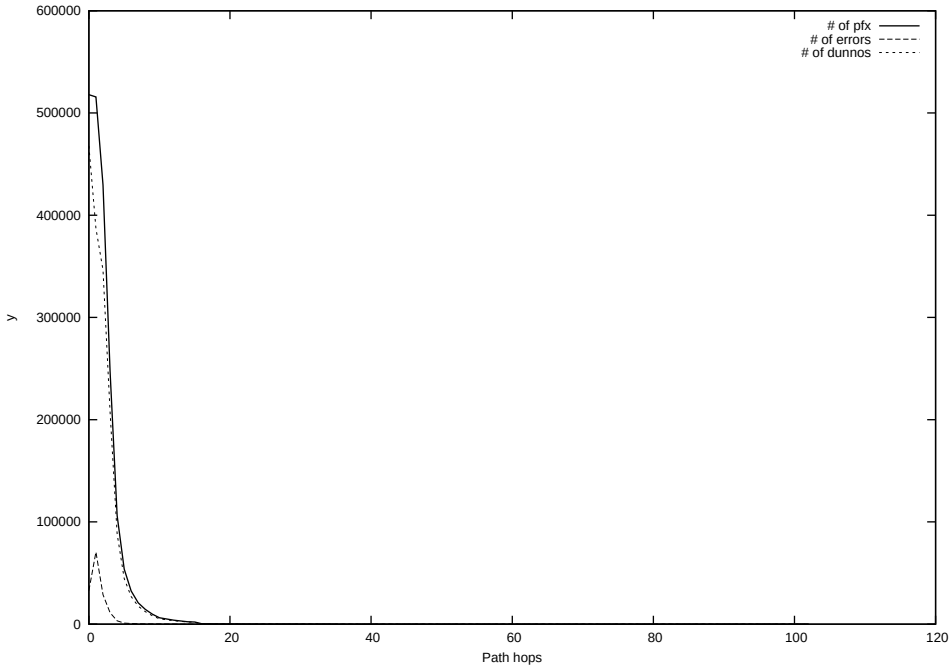
2015-01-08



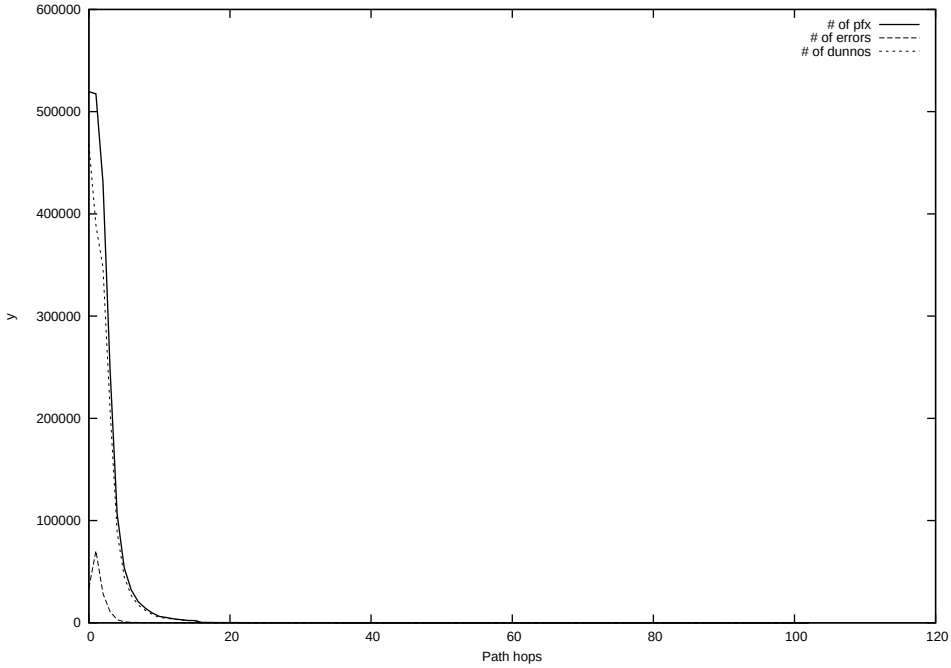
2015-01-09



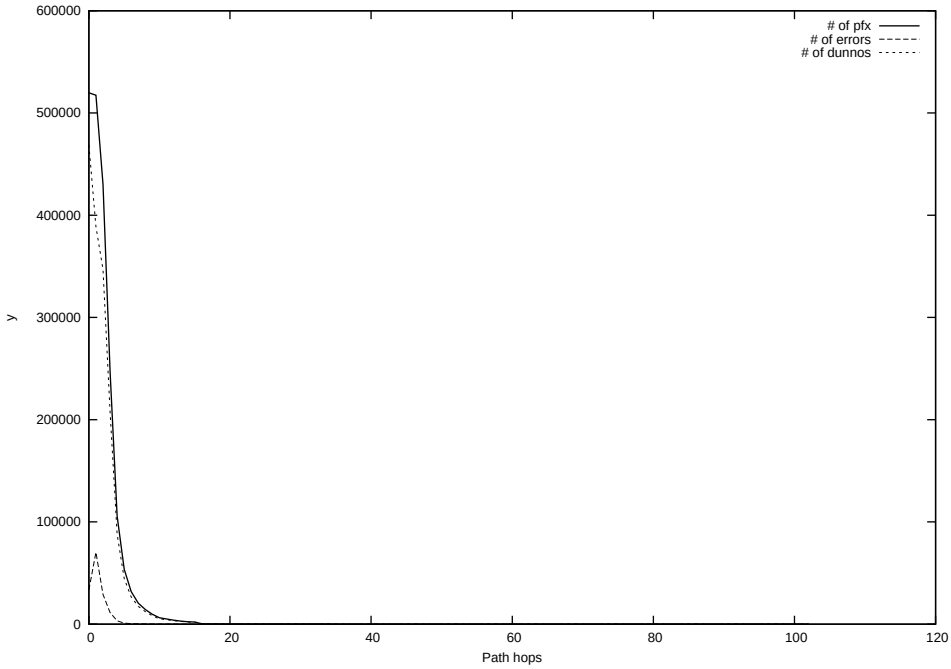
2015-01-10



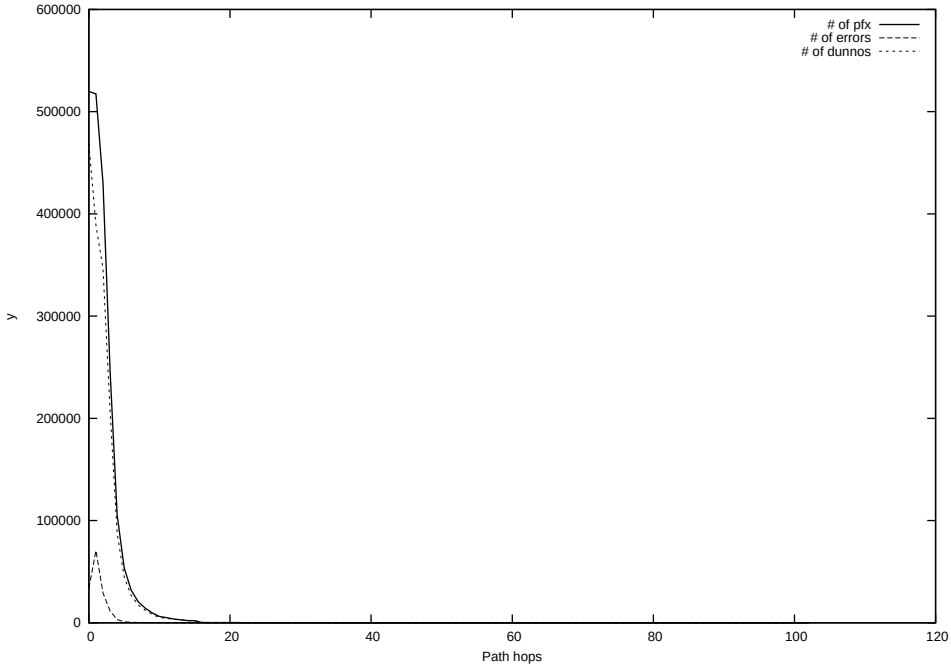
2015-01-11



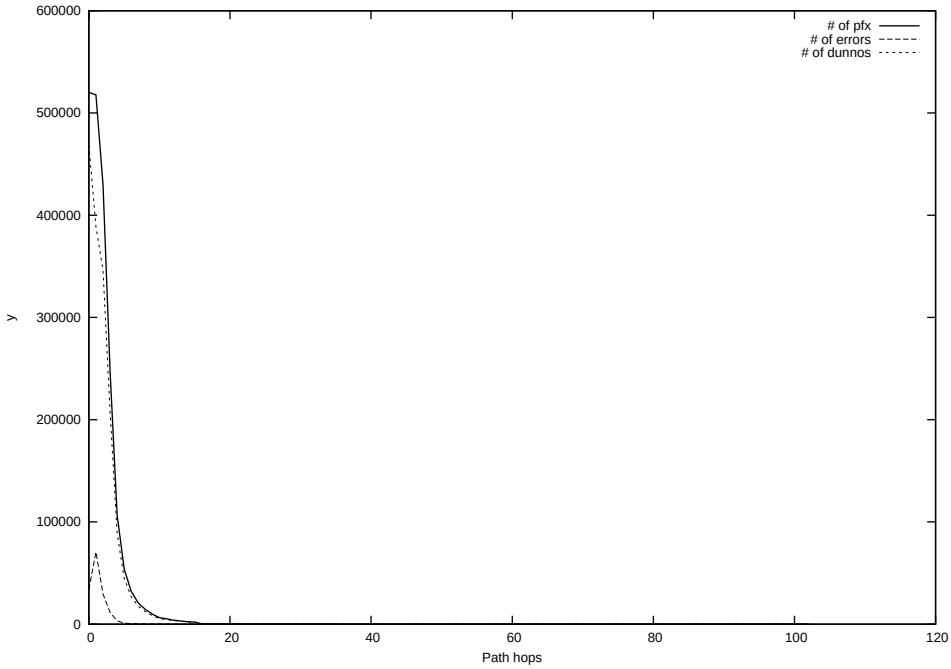
2015-01-12



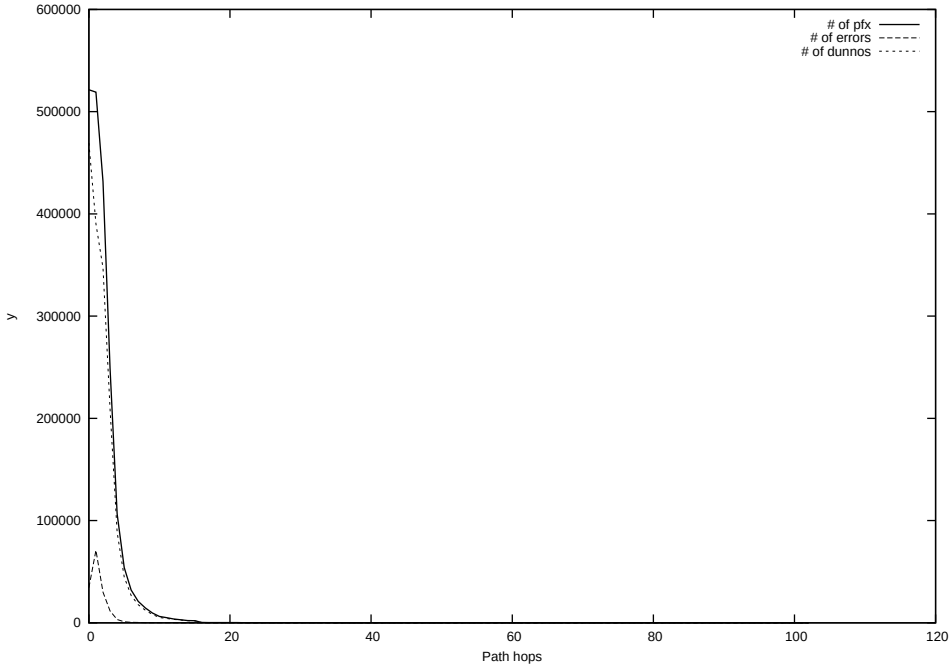
2015-01-13



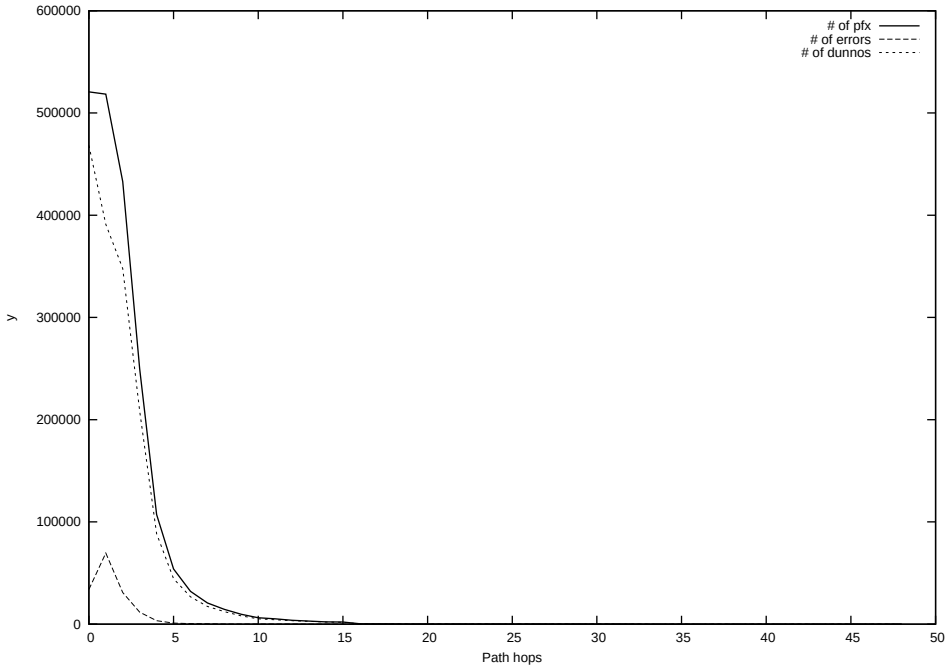
2015-01-14



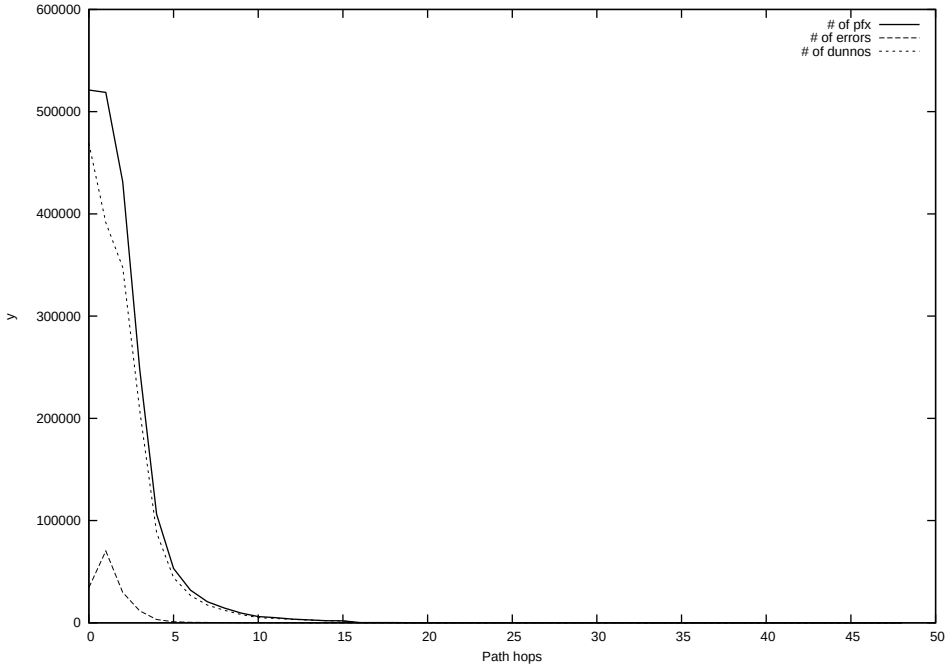
2015-01-15



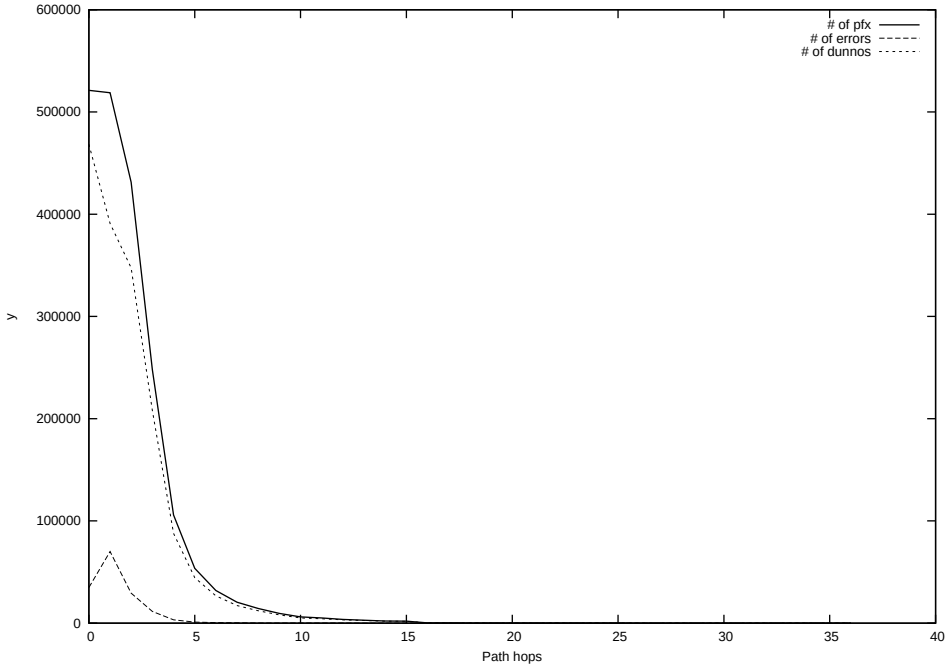
2015-01-16



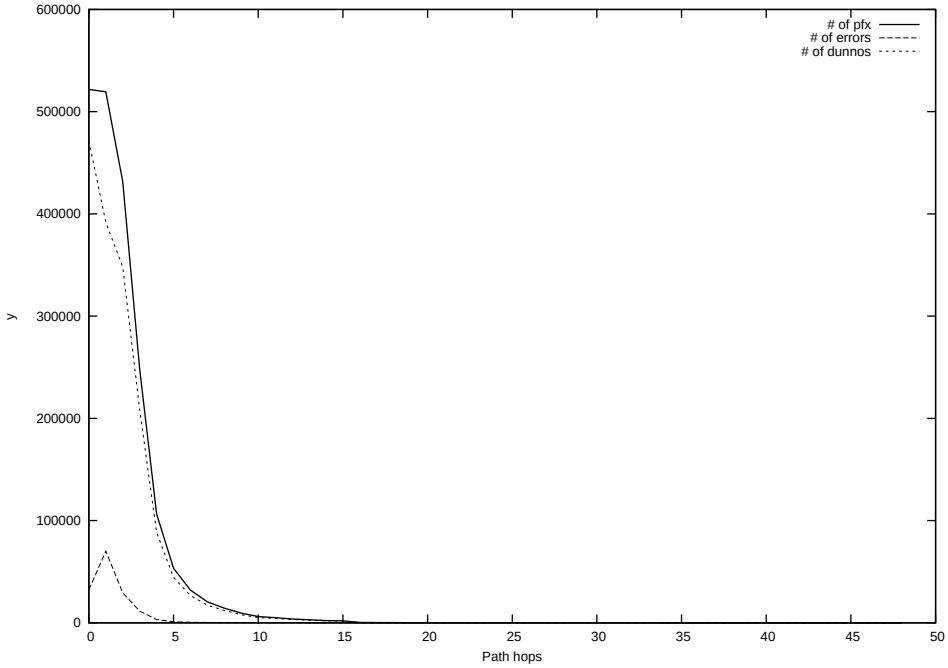
2015-01-17



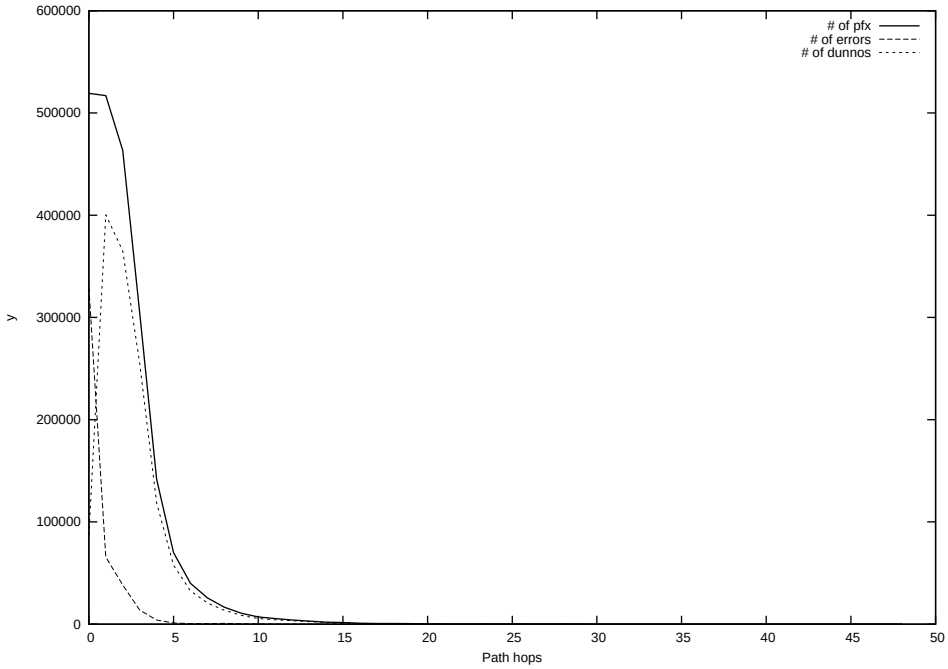
2015-01-18



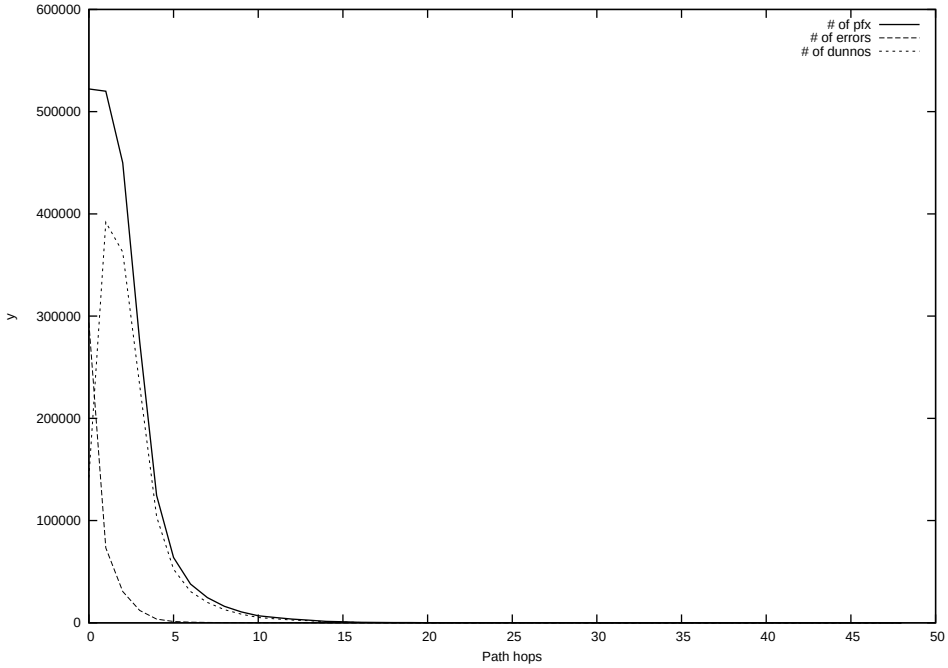
2015-01-19



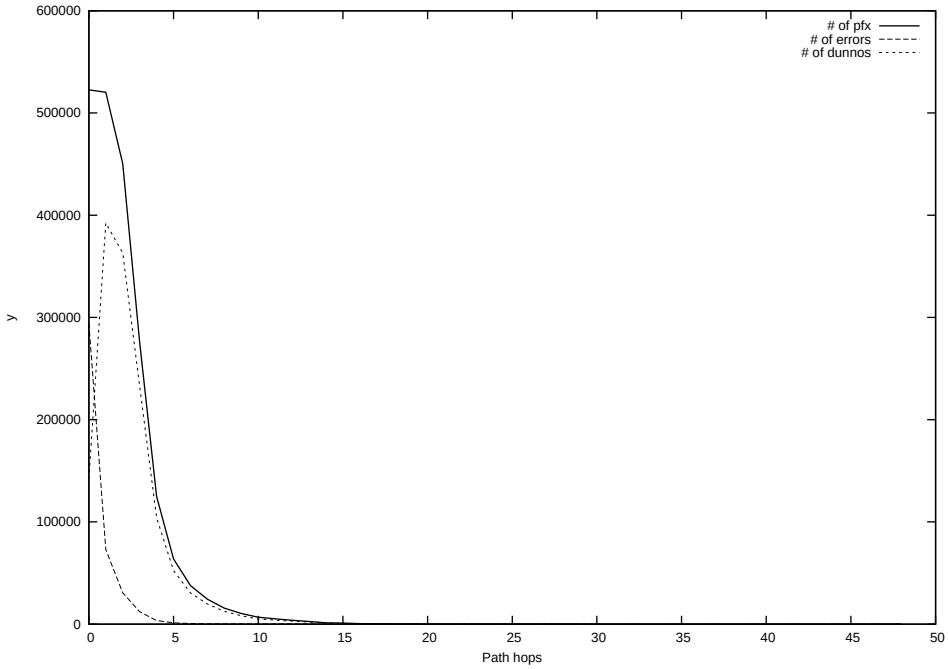
2015-01-20



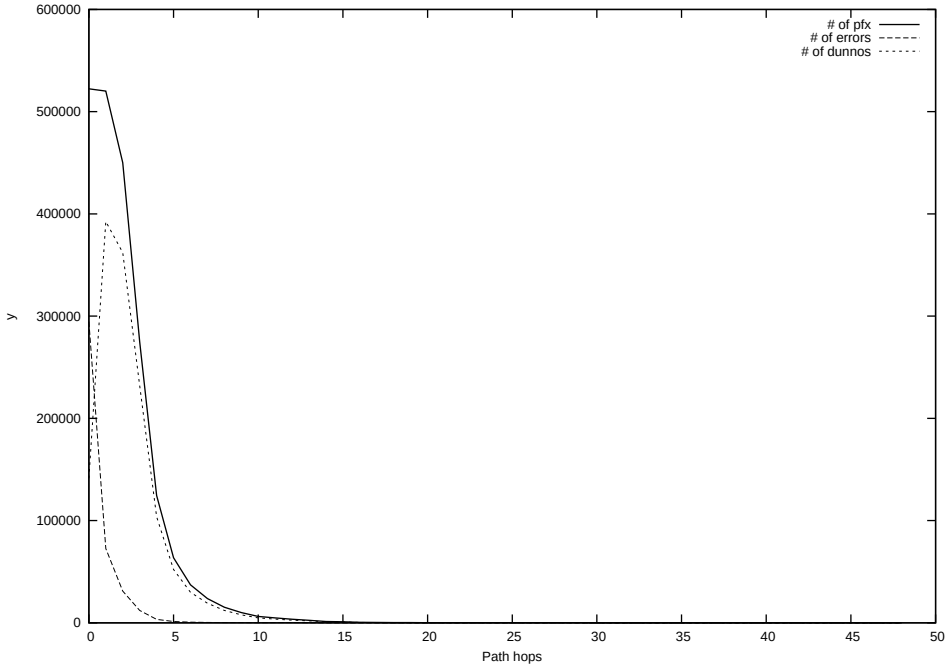
2015-01-21



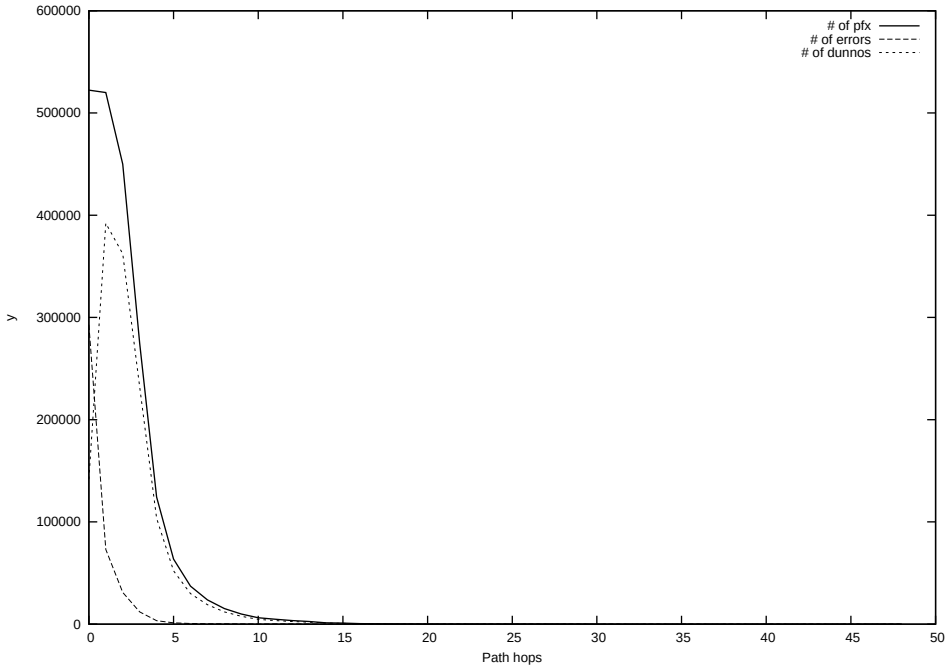
2015-01-22



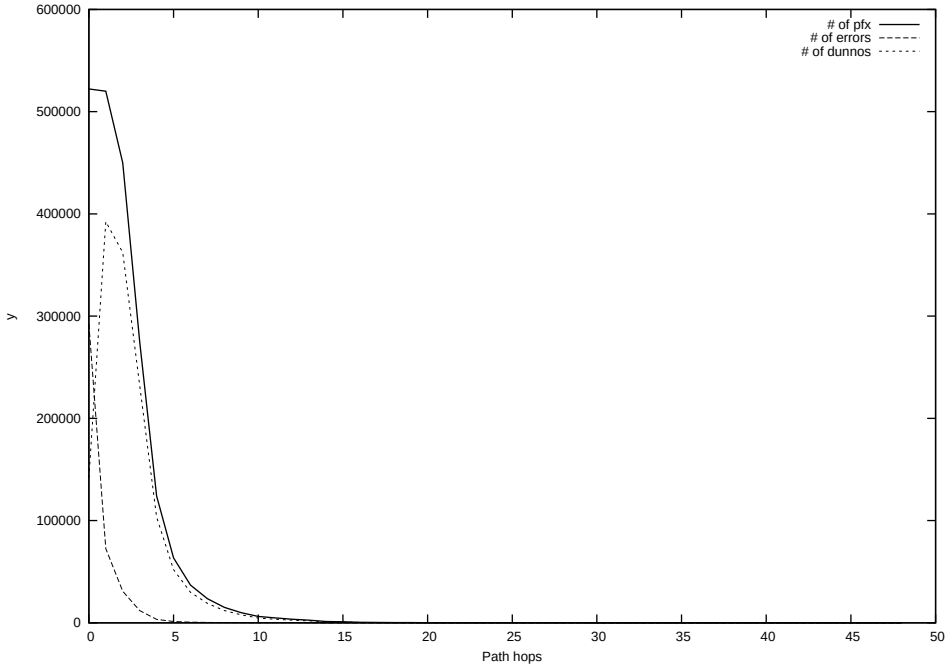
2015-01-23



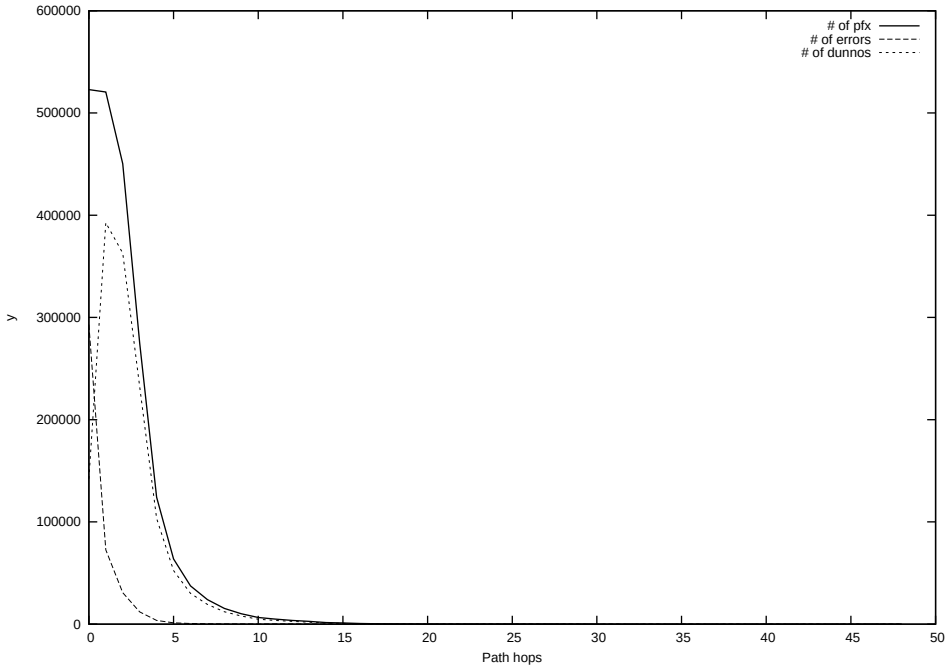
2015-01-24



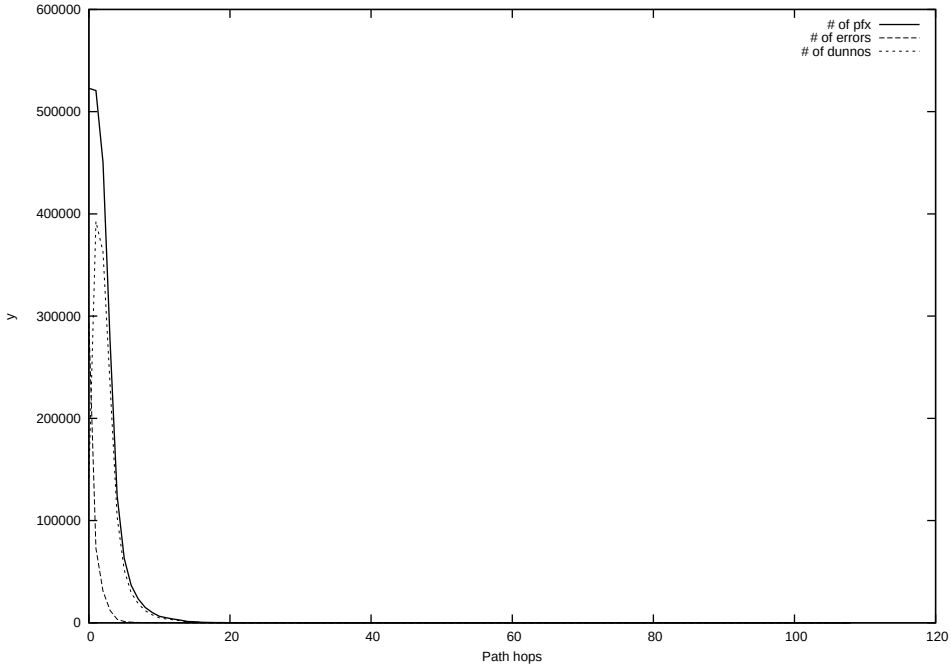
2015-01-25



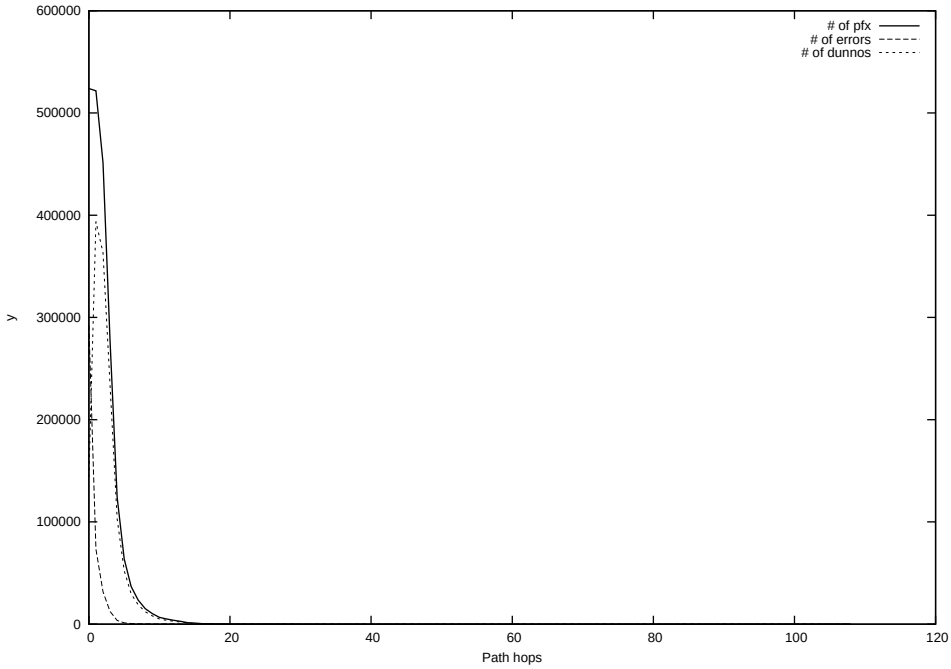
2015-01-26



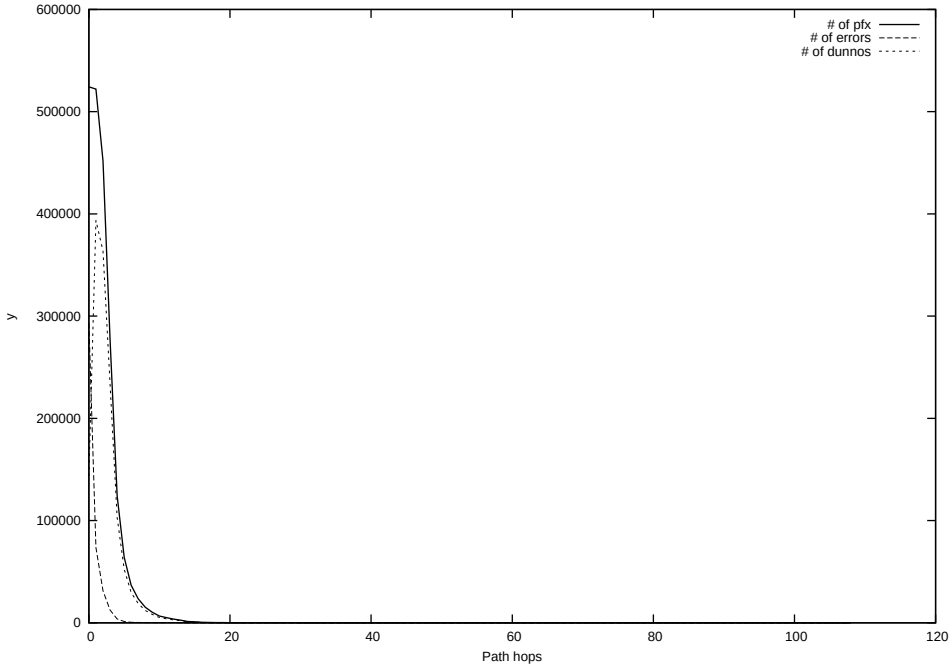
2015-01-27



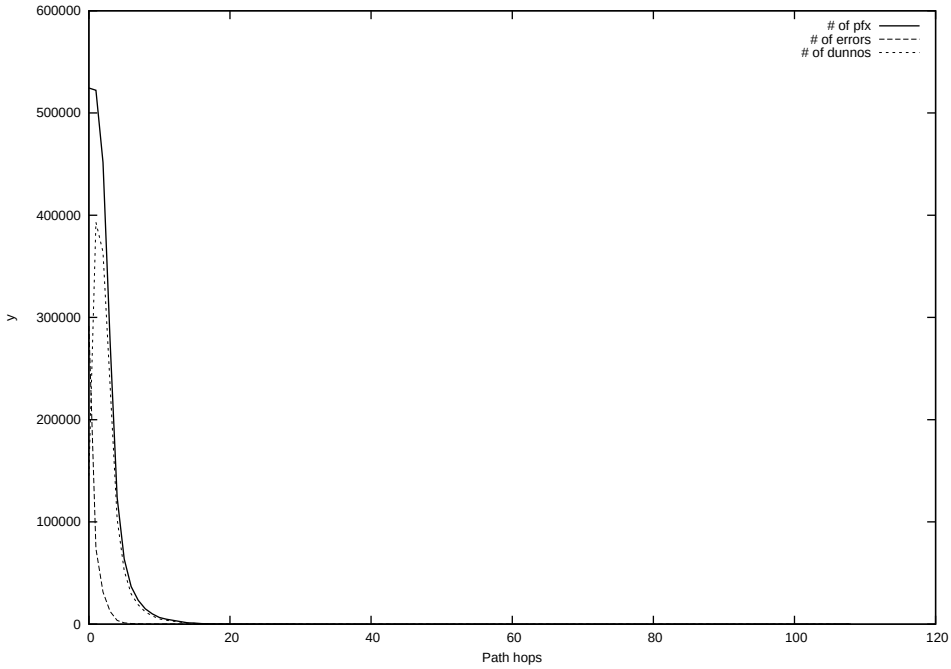
2015-01-28



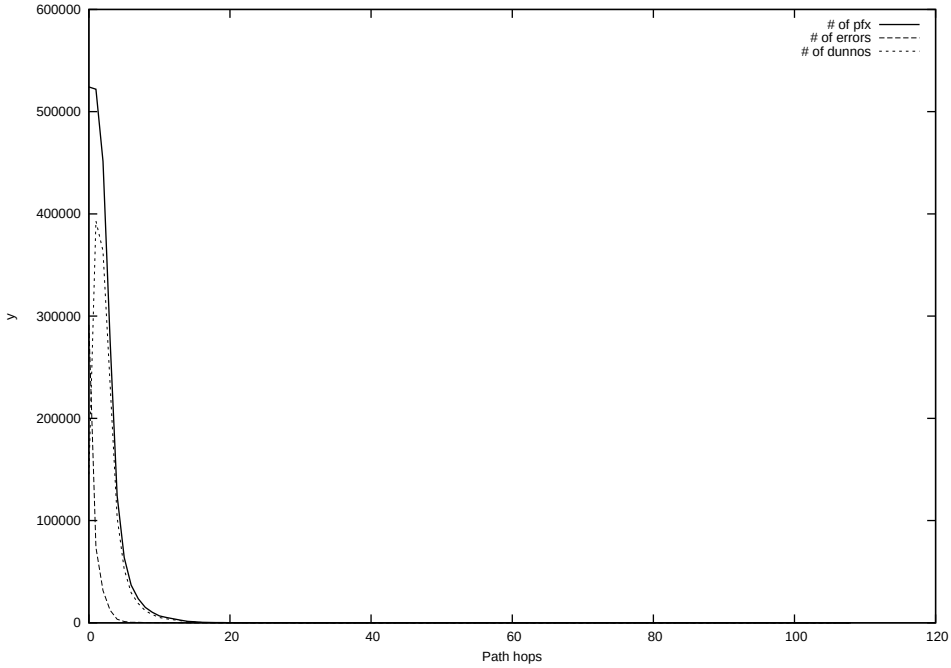
2015-01-29



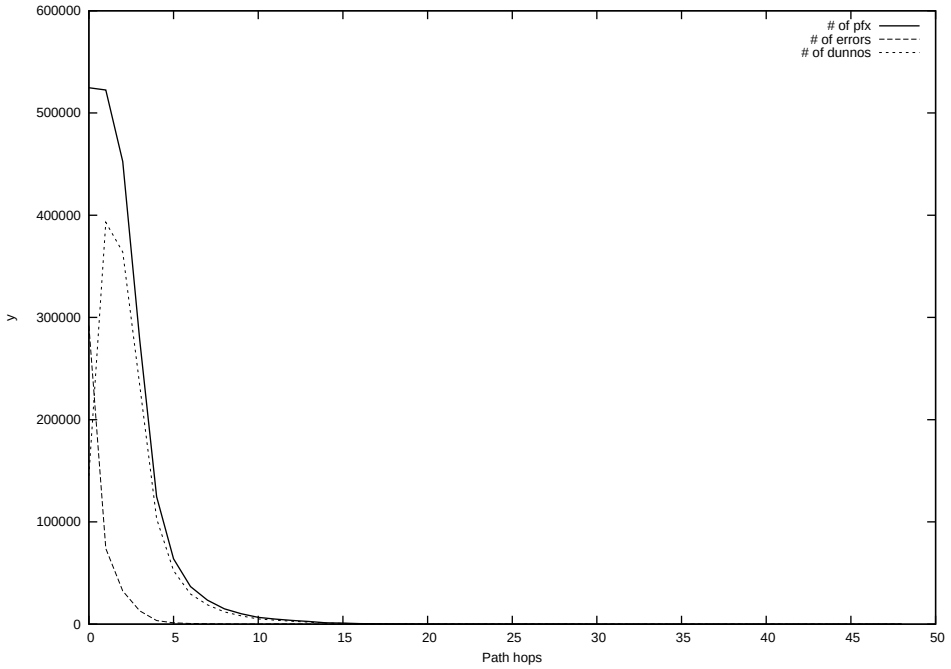
2015-01-30



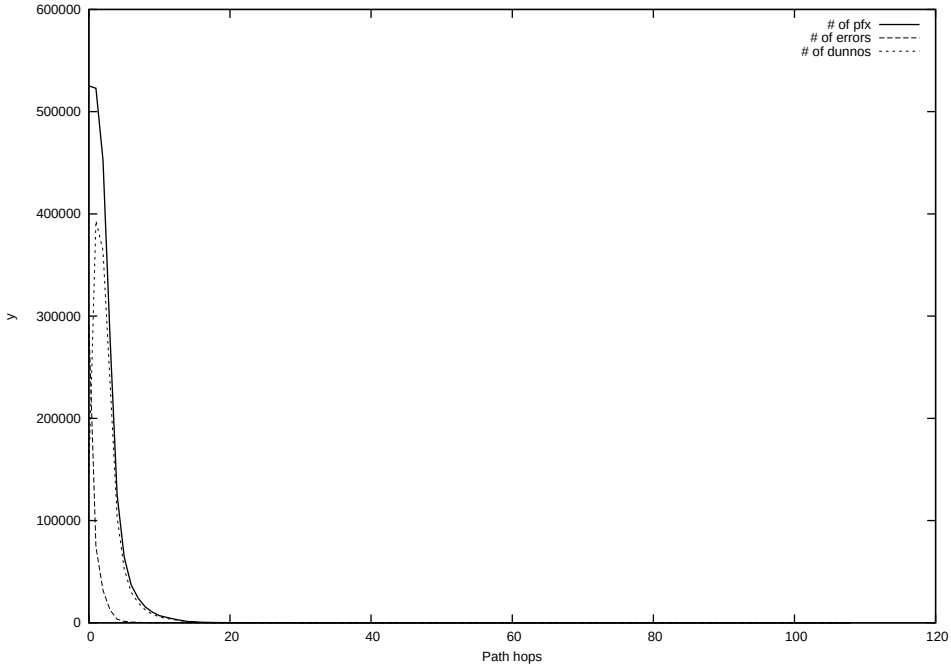
2015-01-31



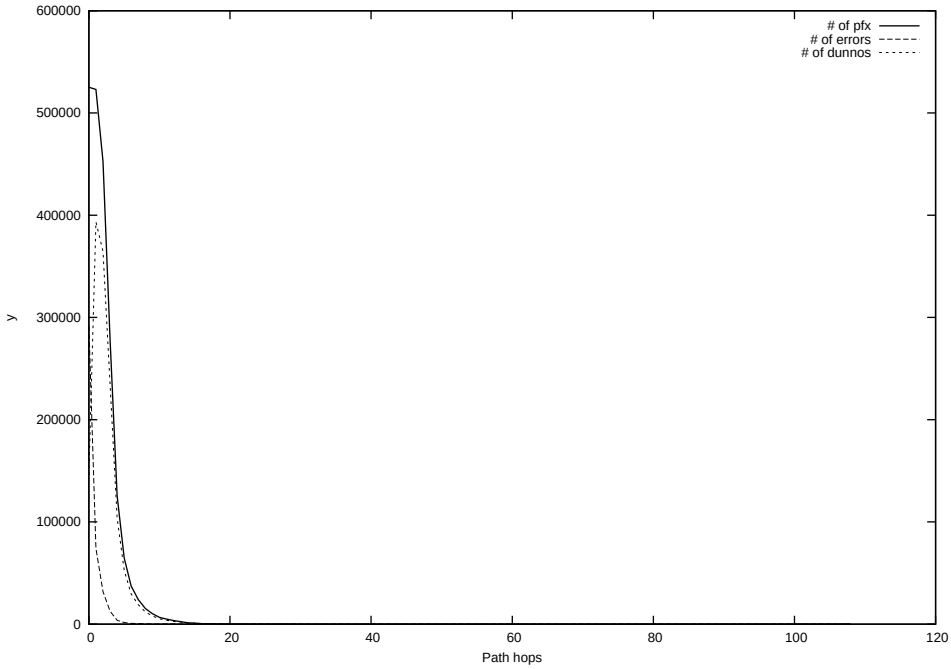
2015-02-01



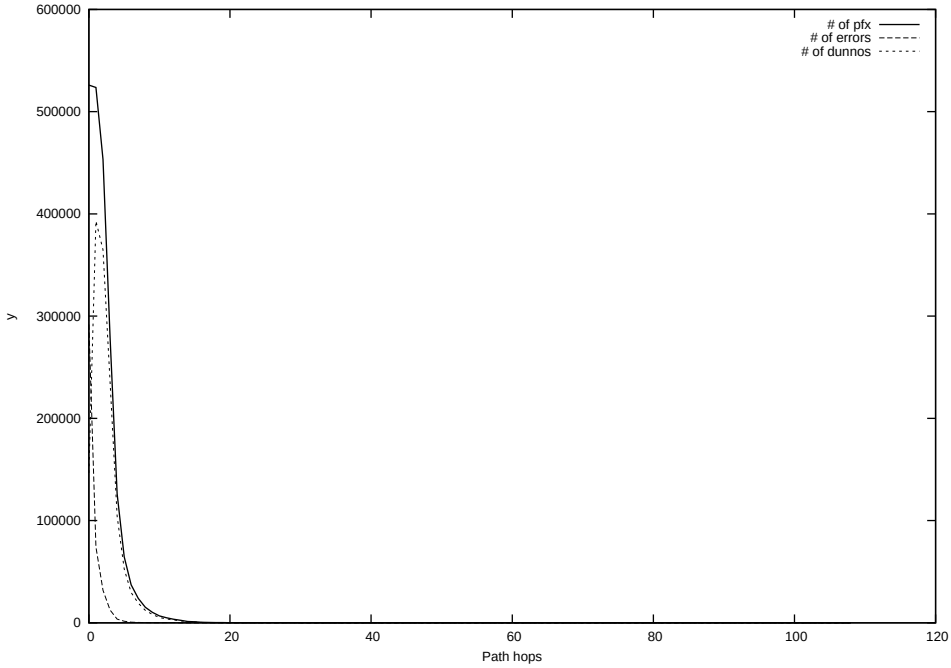
2015-02-02



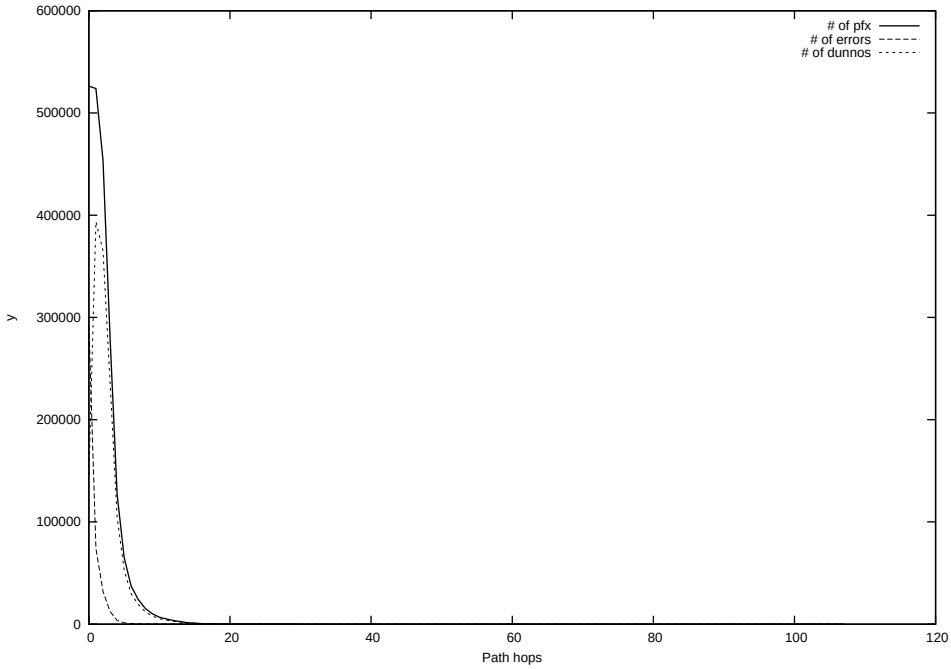
2015-02-03



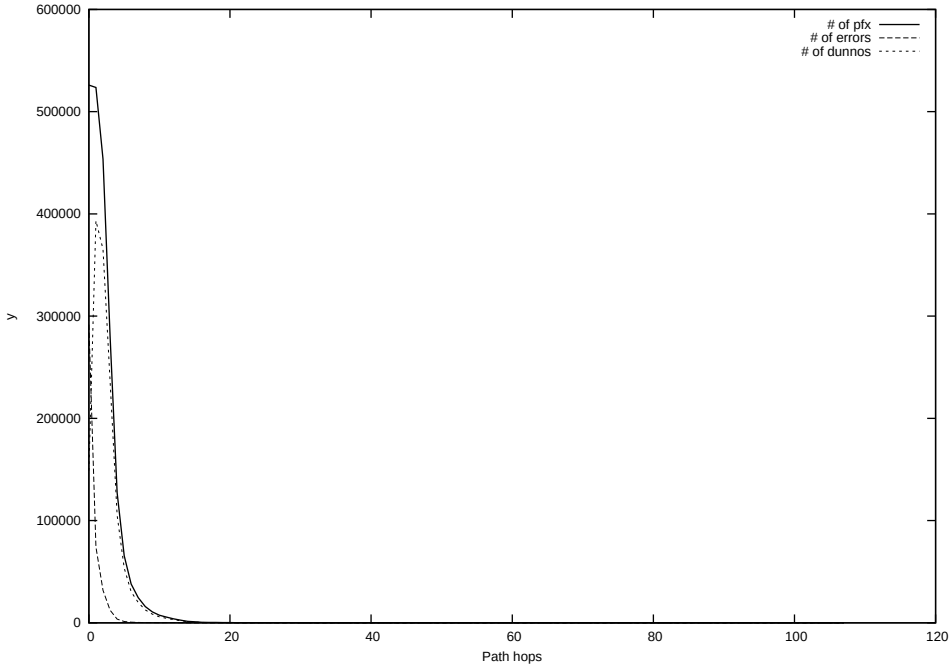
2015-02-04



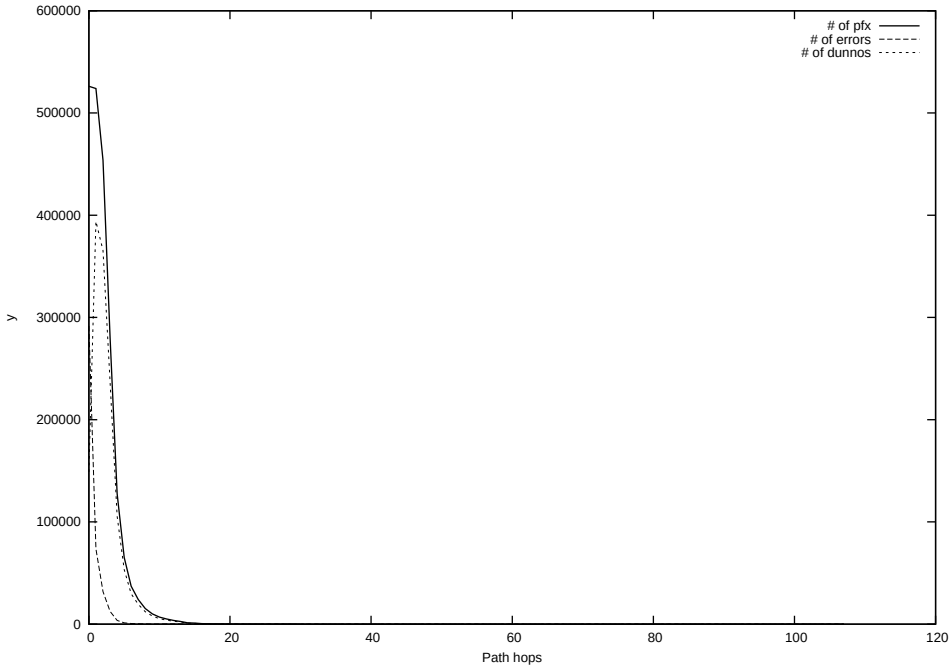
2015-02-05



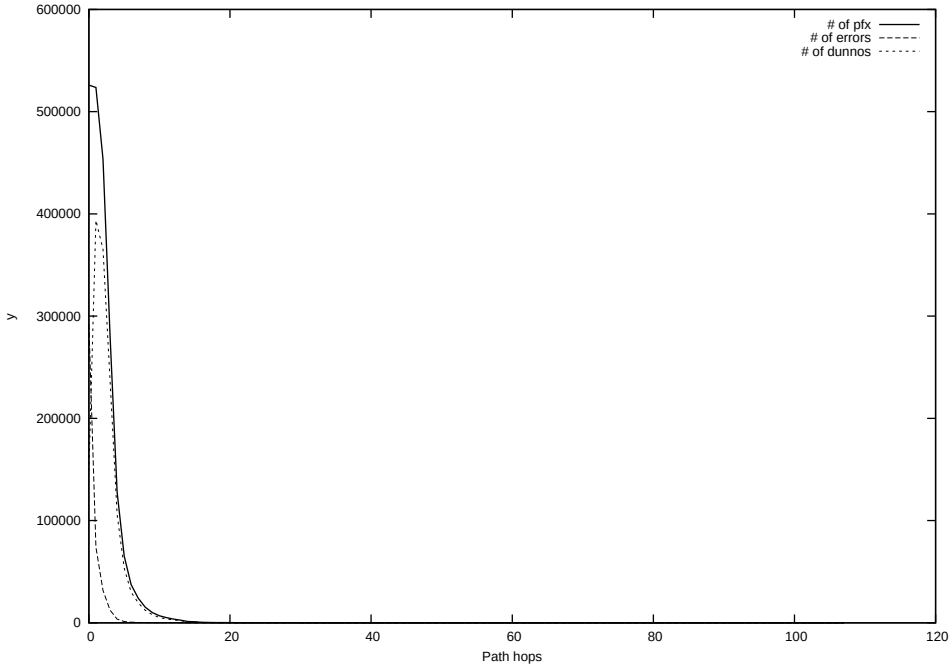
2015-02-06



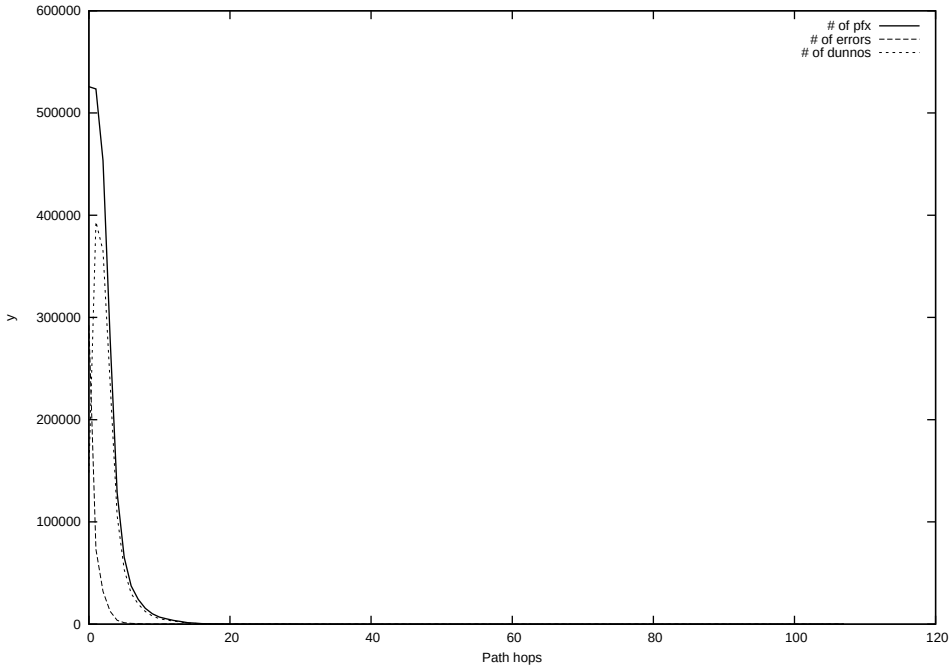
2015-02-07



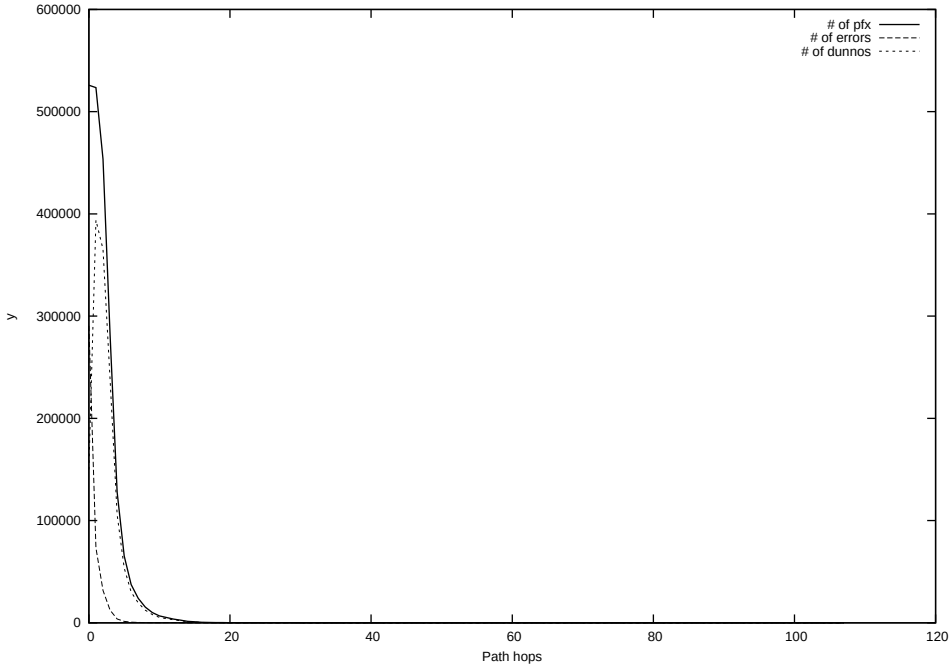
2015-02-08



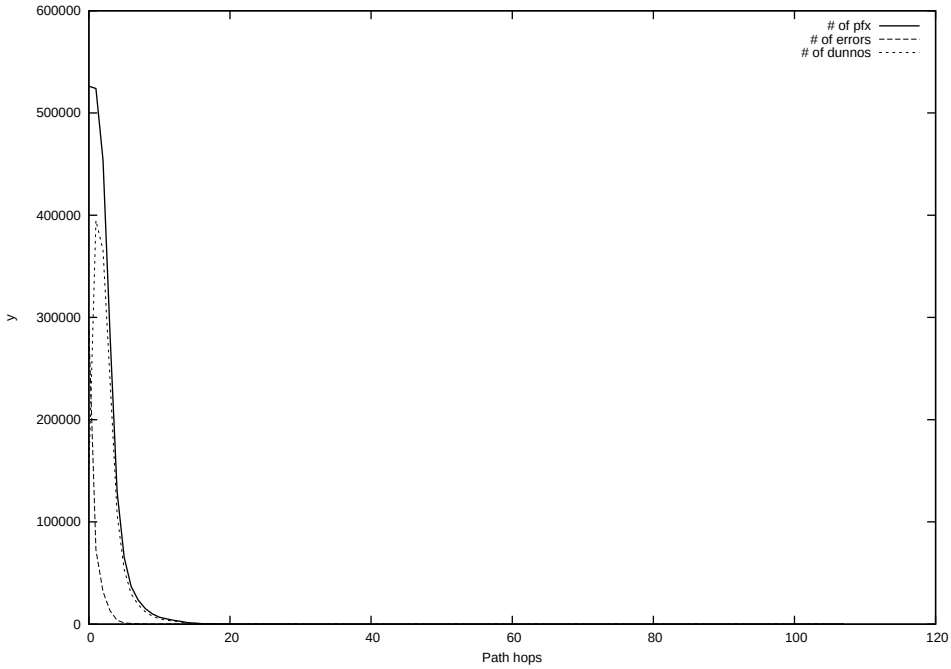
2015-02-09



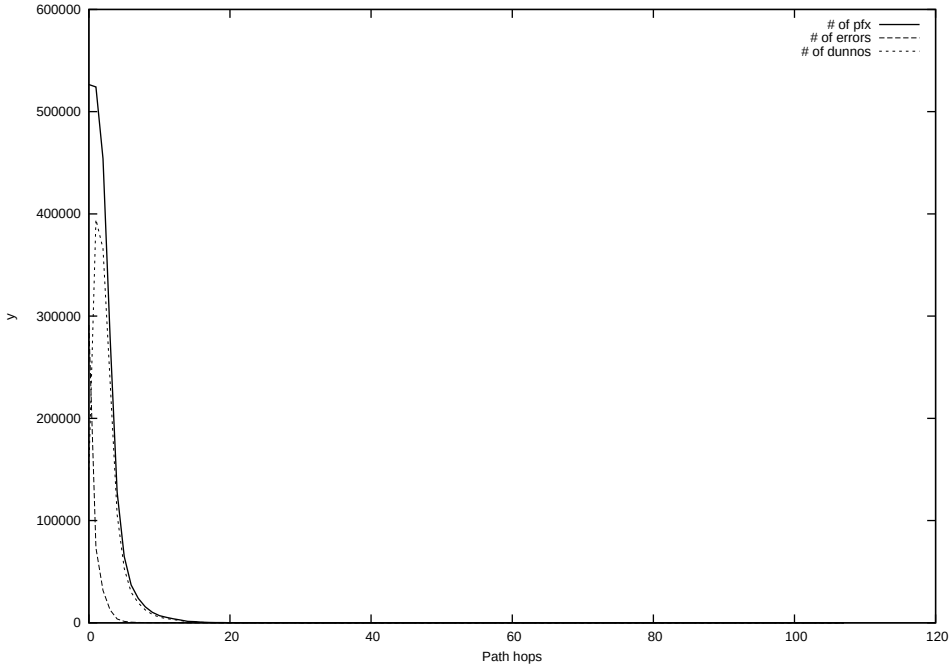
2015-02-10



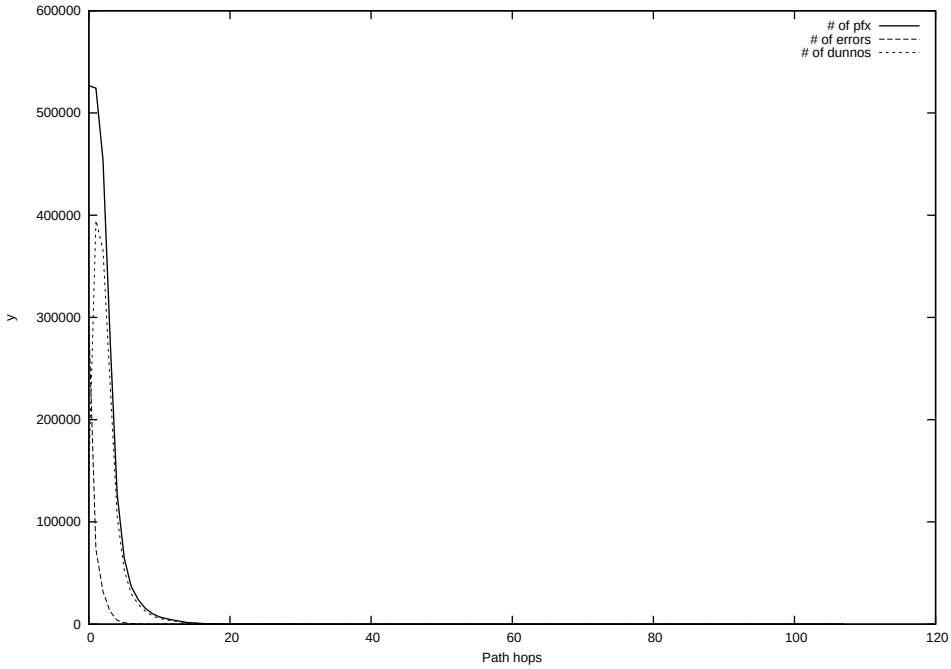
2015-02-11



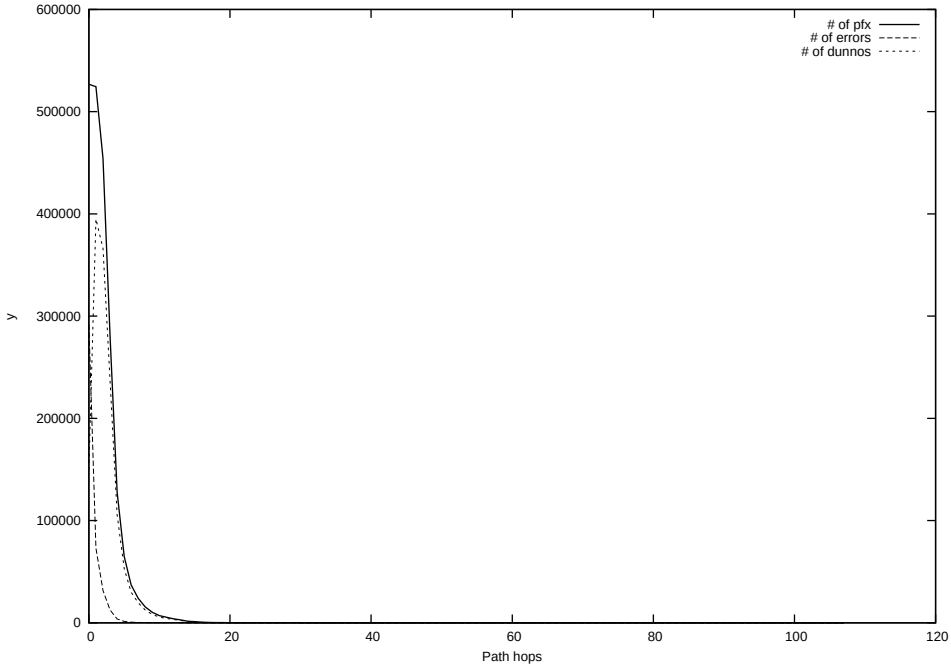
2015-02-12



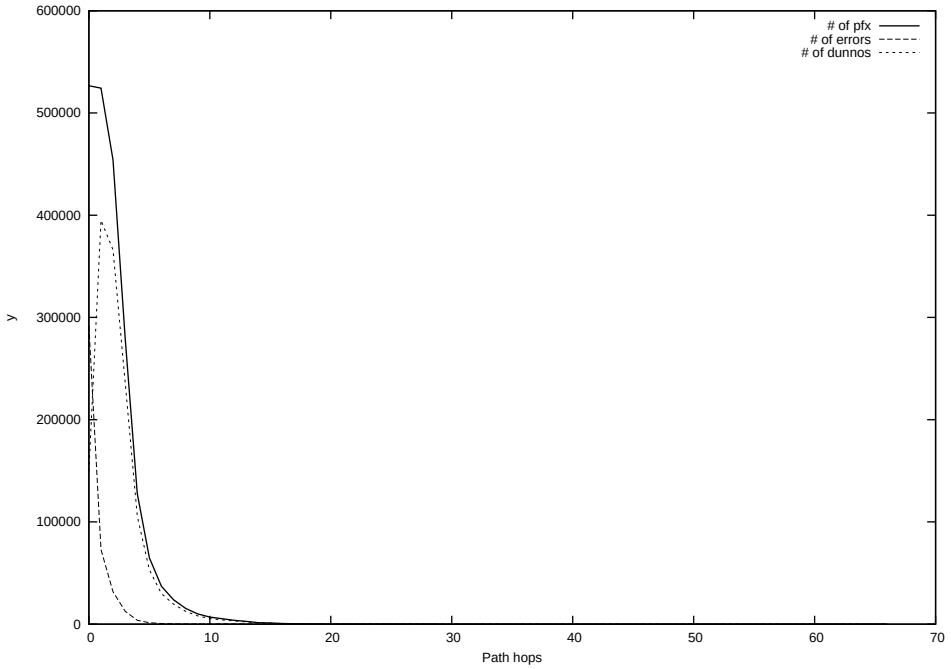
2015-02-13



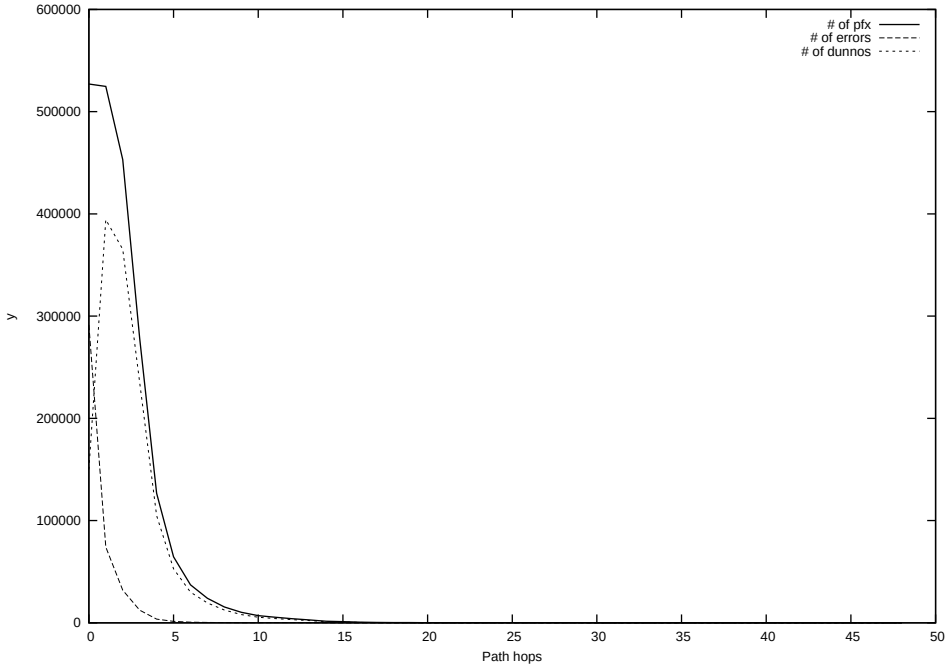
2015-02-14



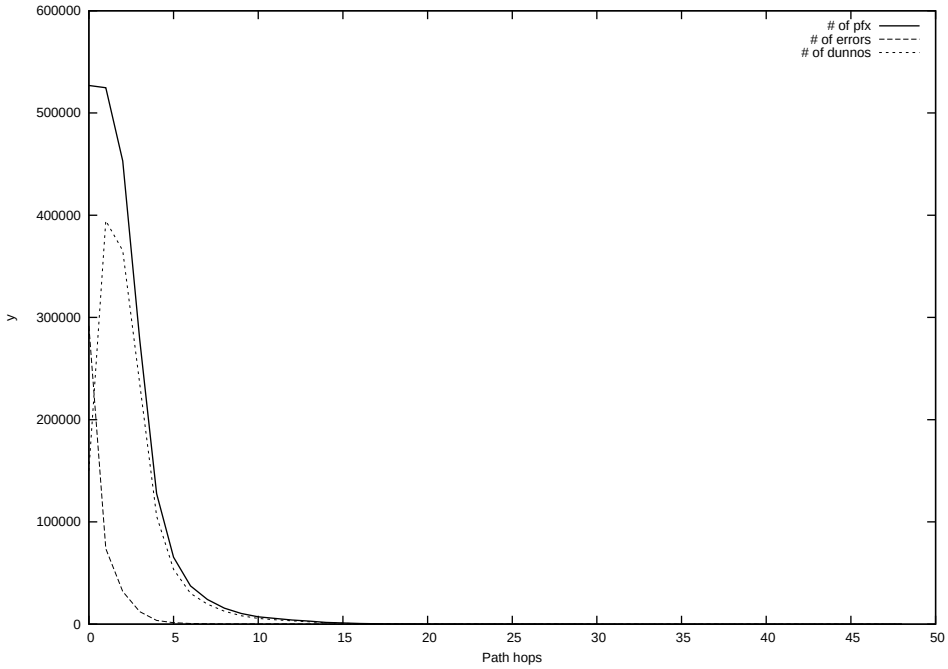
2015-02-15



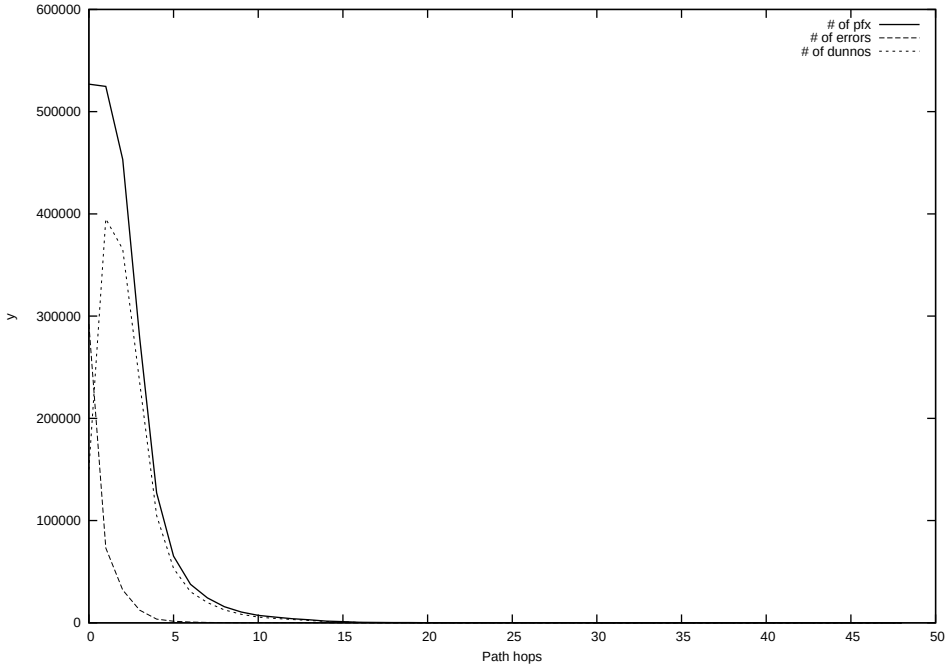
2015-02-16



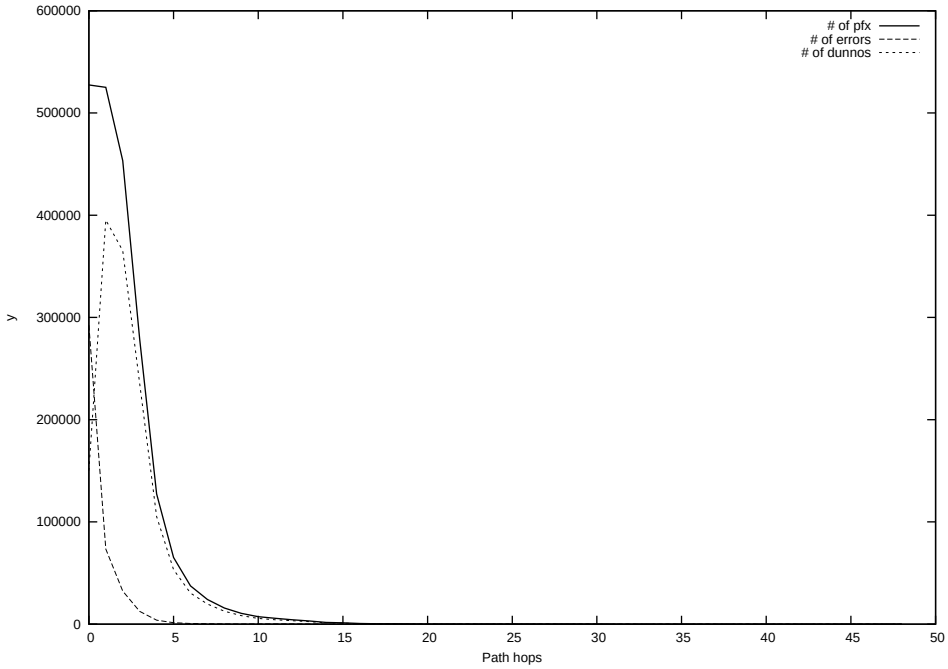
2015-02-17



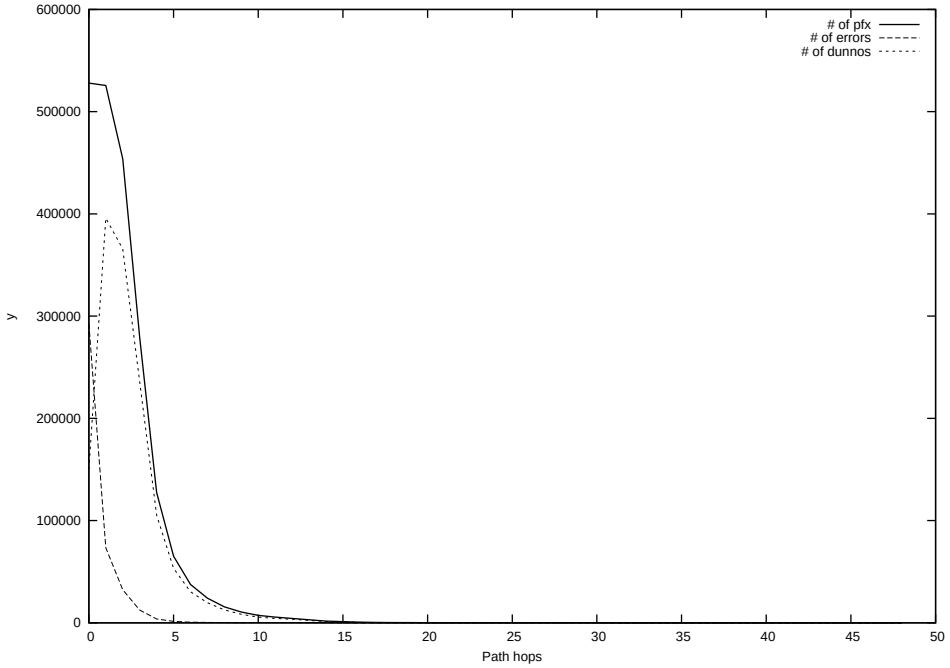
2015-02-18



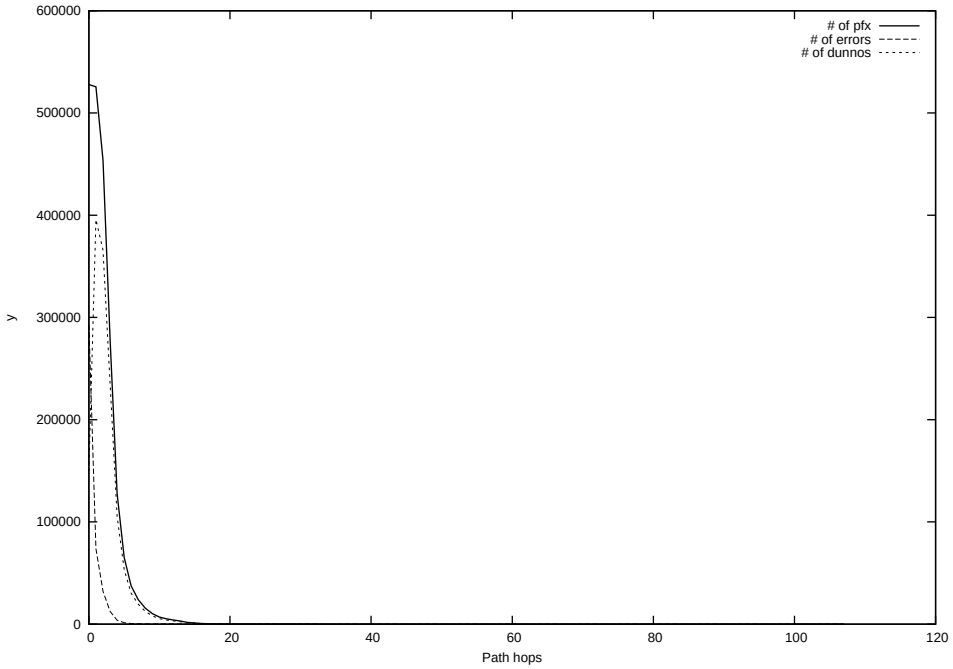
2015-02-19



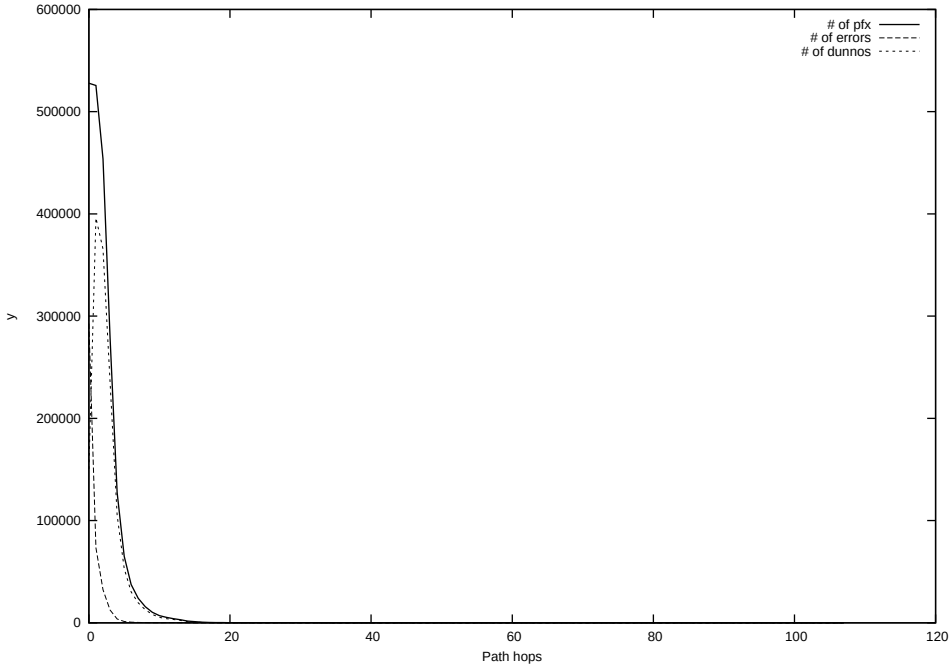
2015-02-20



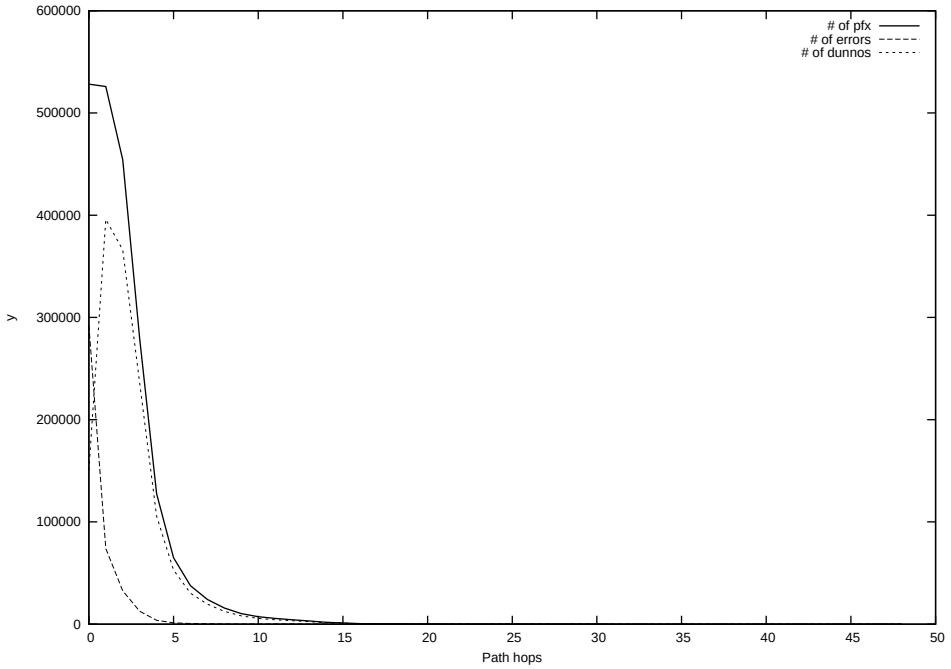
2015-02-21



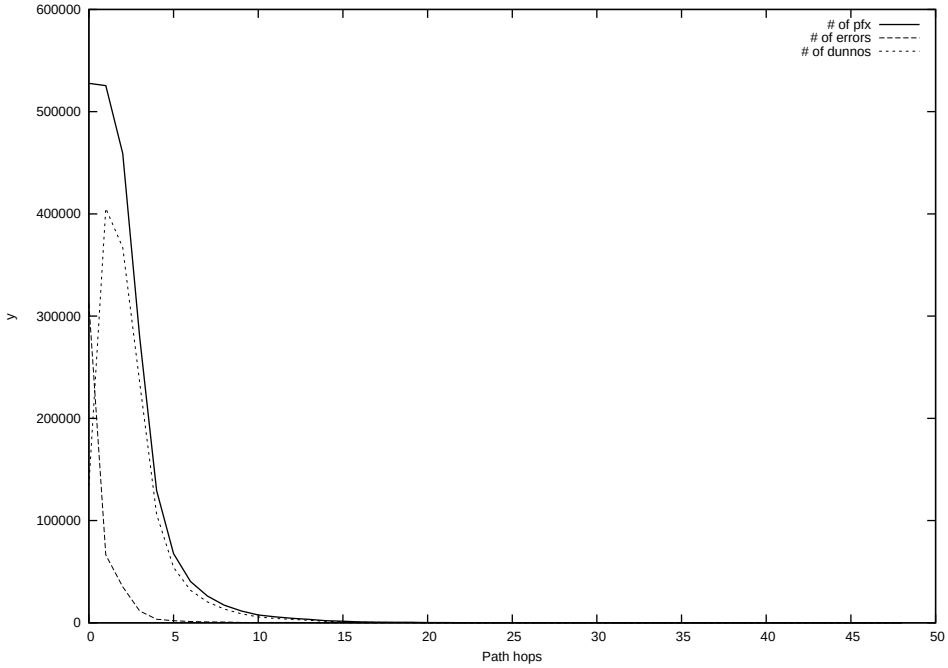
2015-02-22



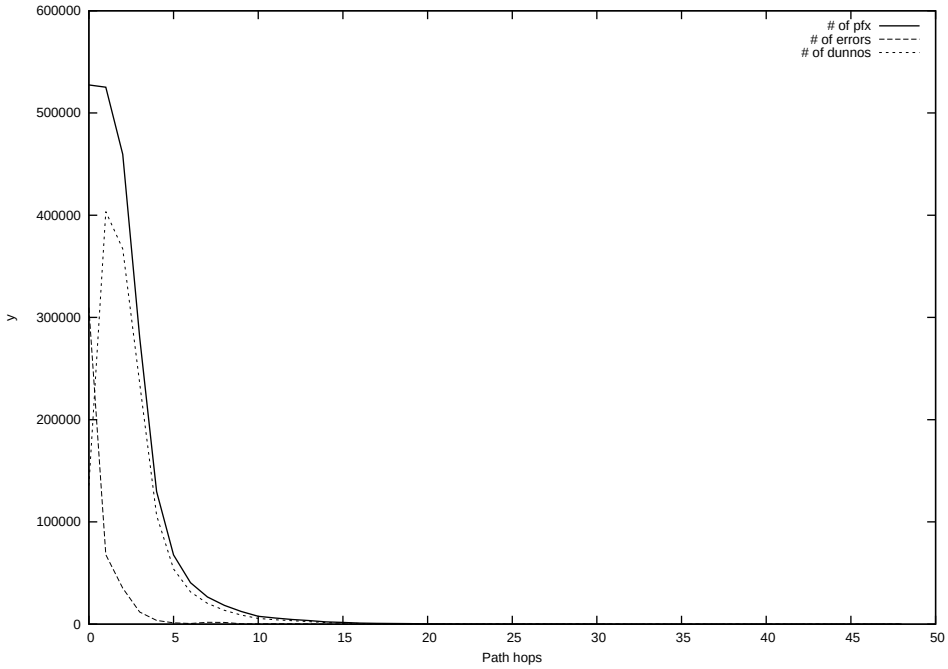
2015-02-23



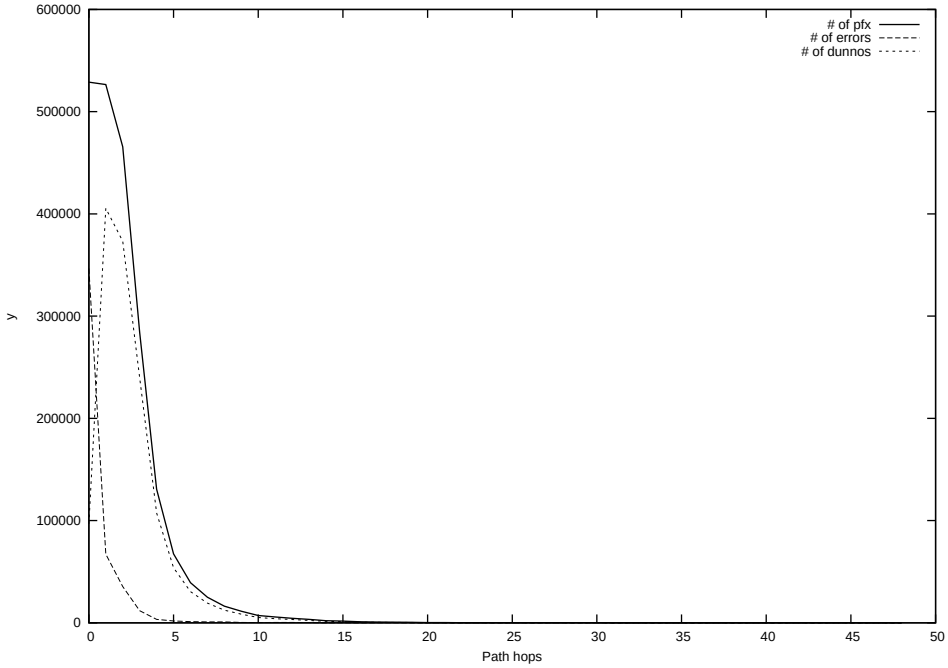
2015-02-24



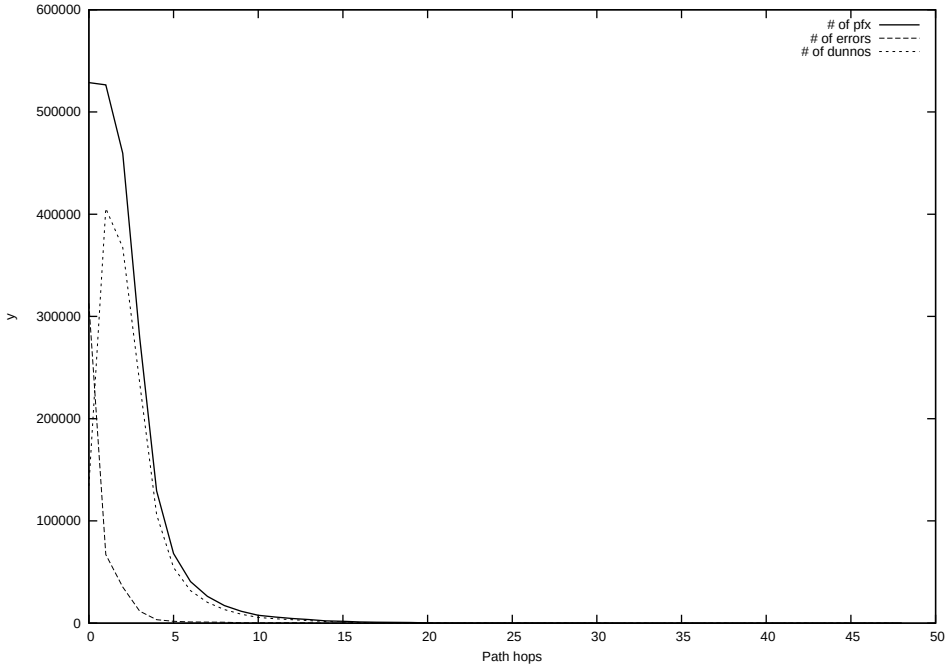
2015-02-25



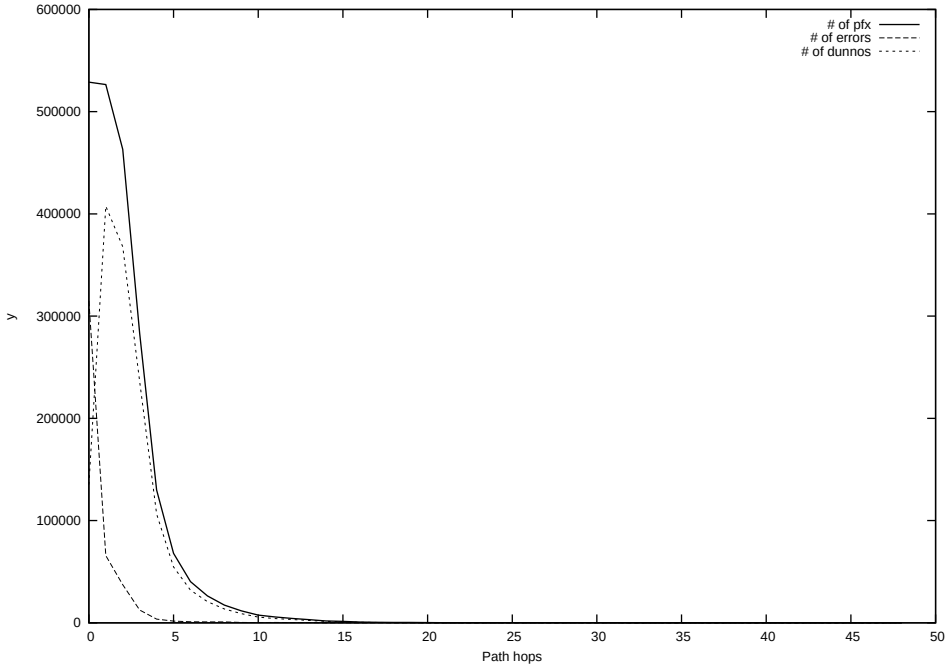
2015-02-26



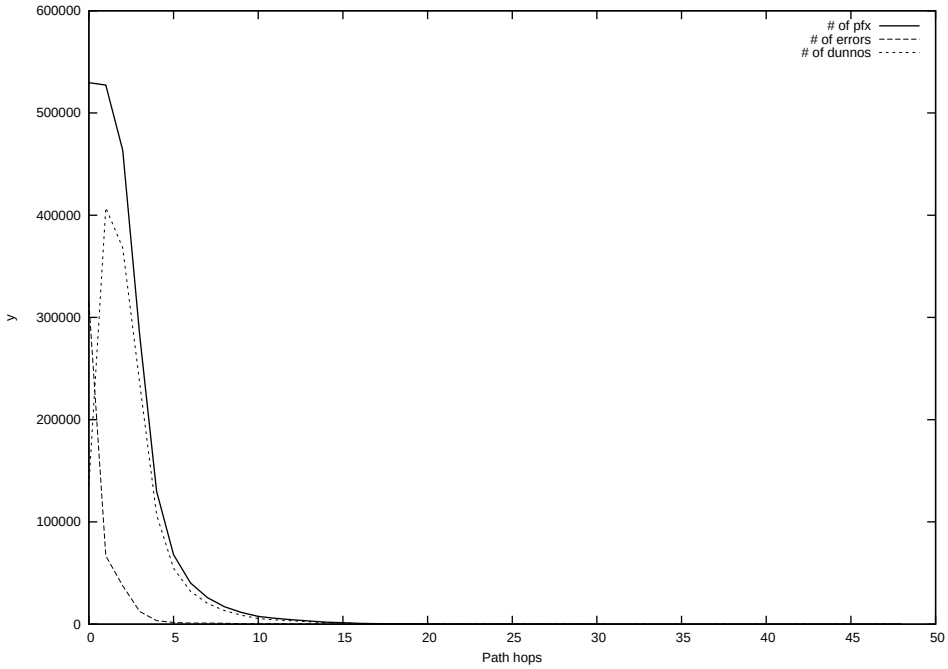
2015-02-27



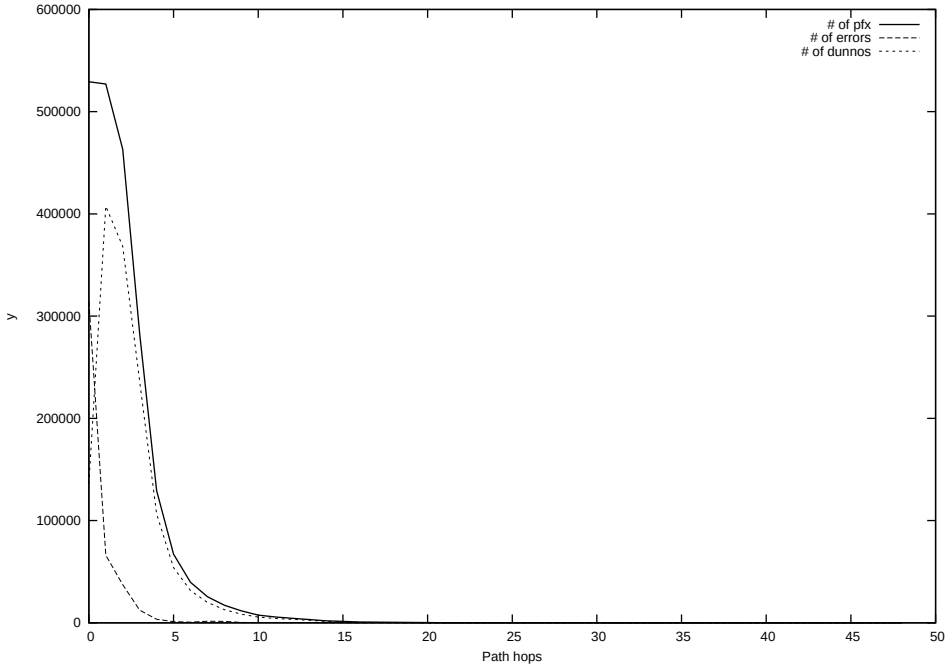
2015-02-28



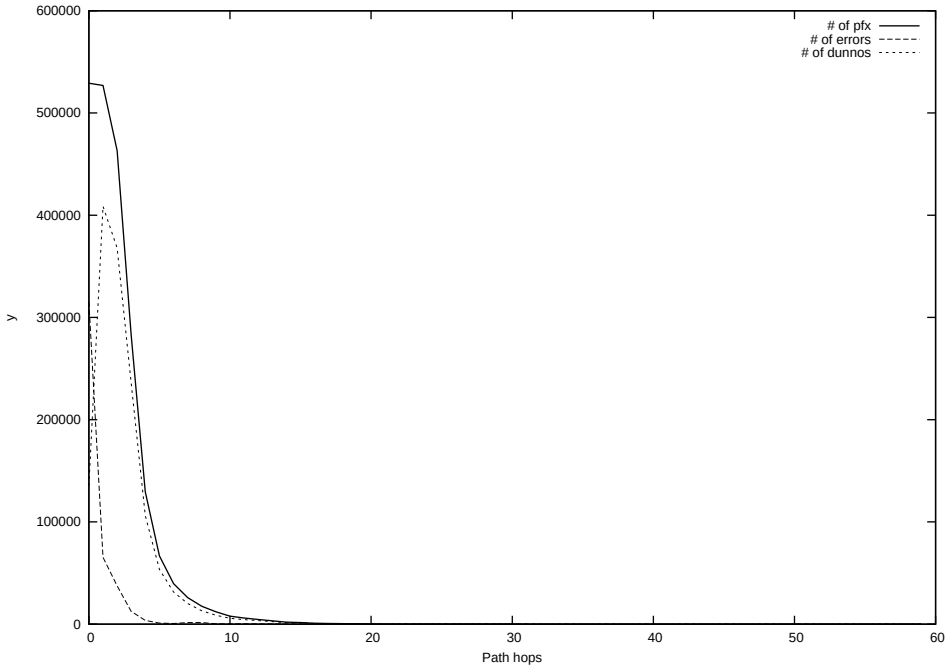
2015-03-01



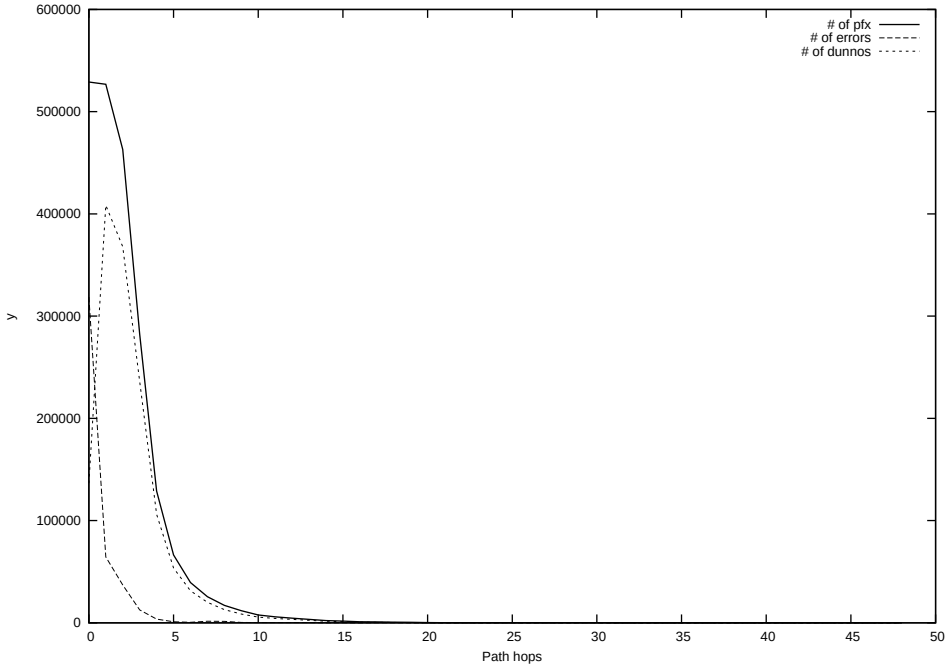
2015-03-02



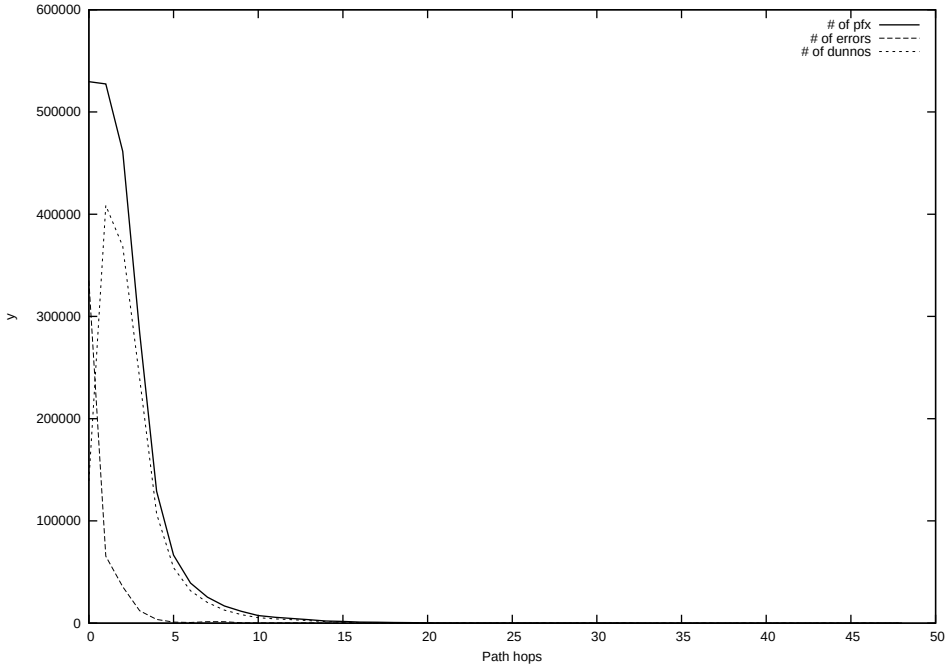
2015-03-03



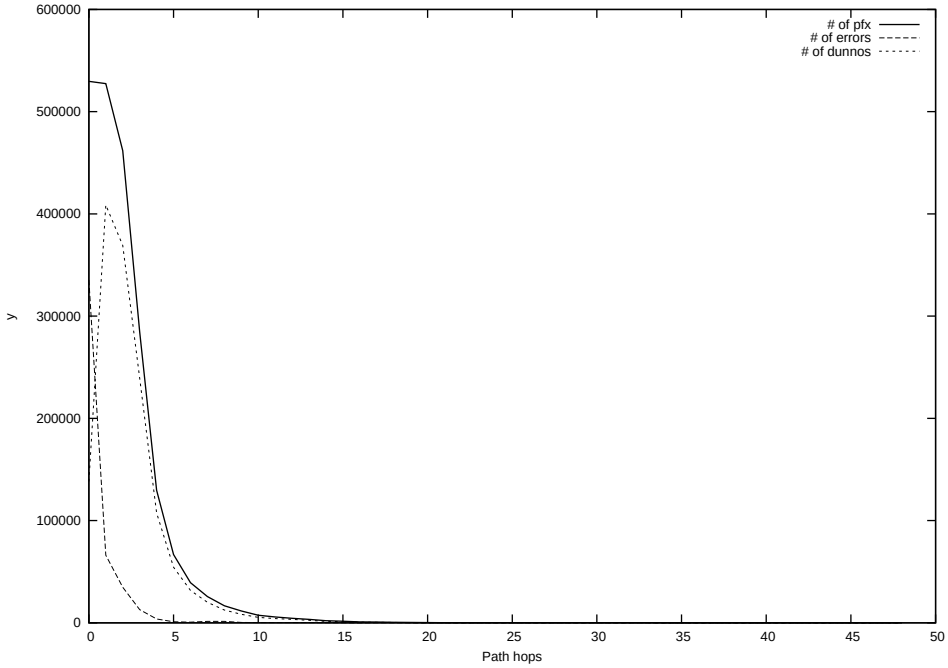
2015-03-04



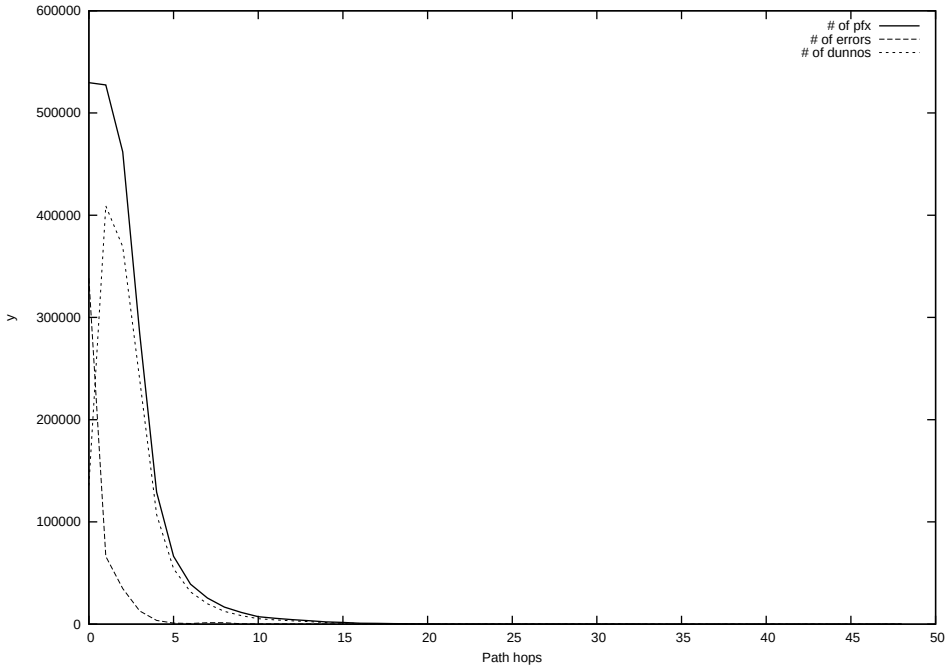
2015-03-05



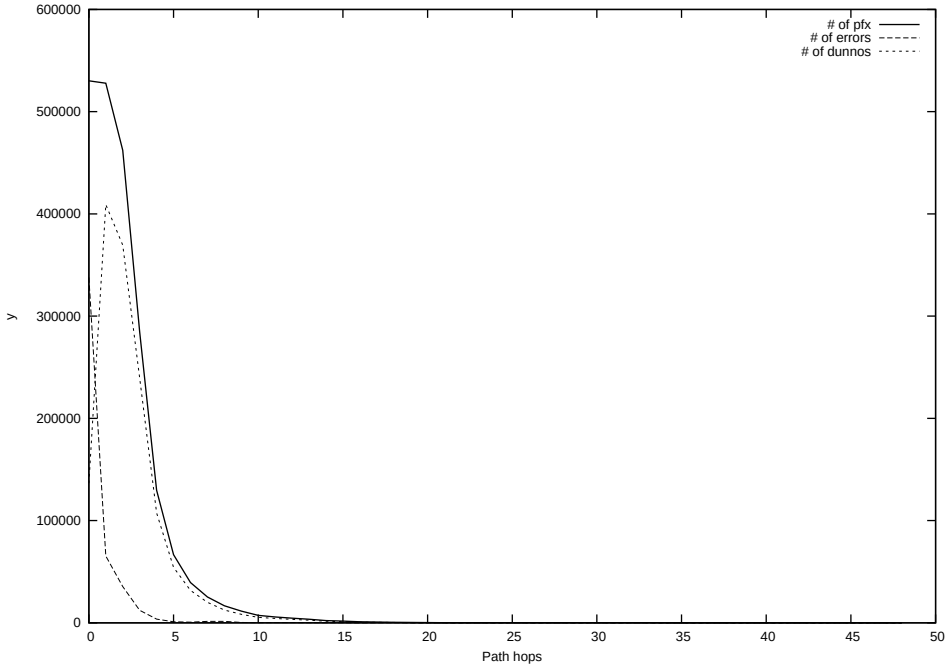
2015-03-06



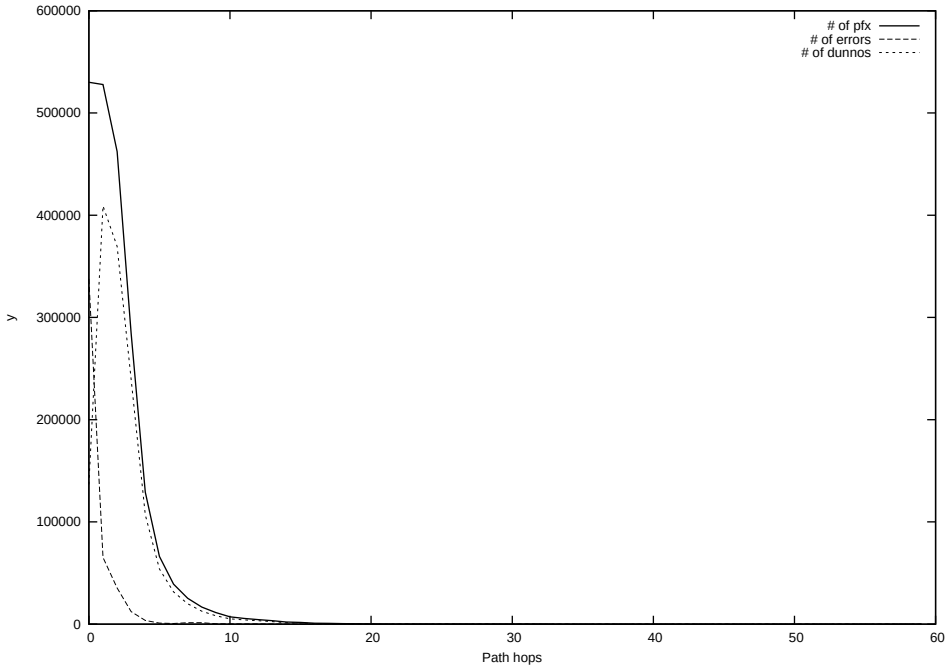
2015-03-07



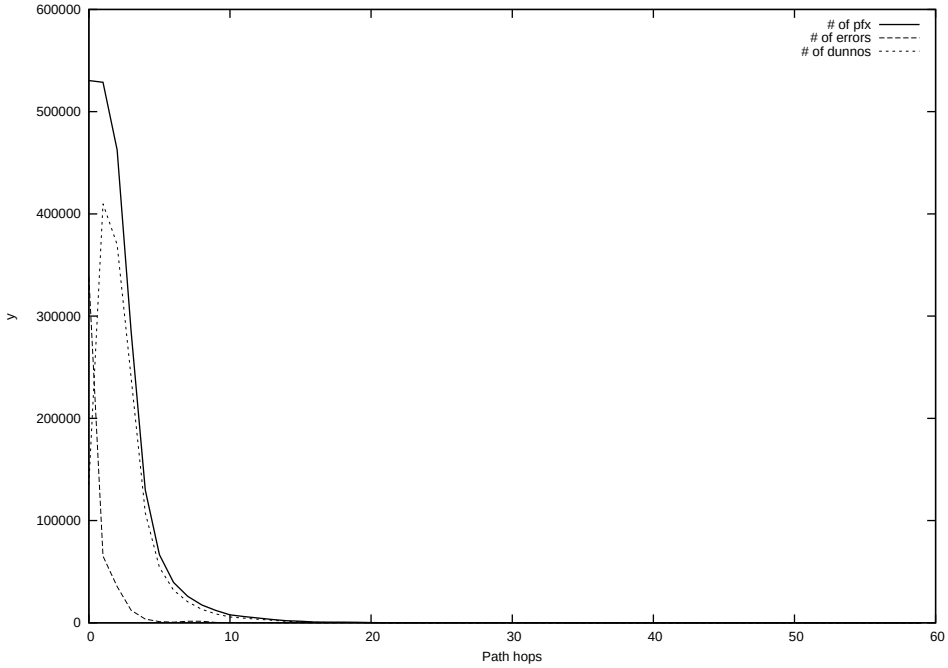
2015-03-08



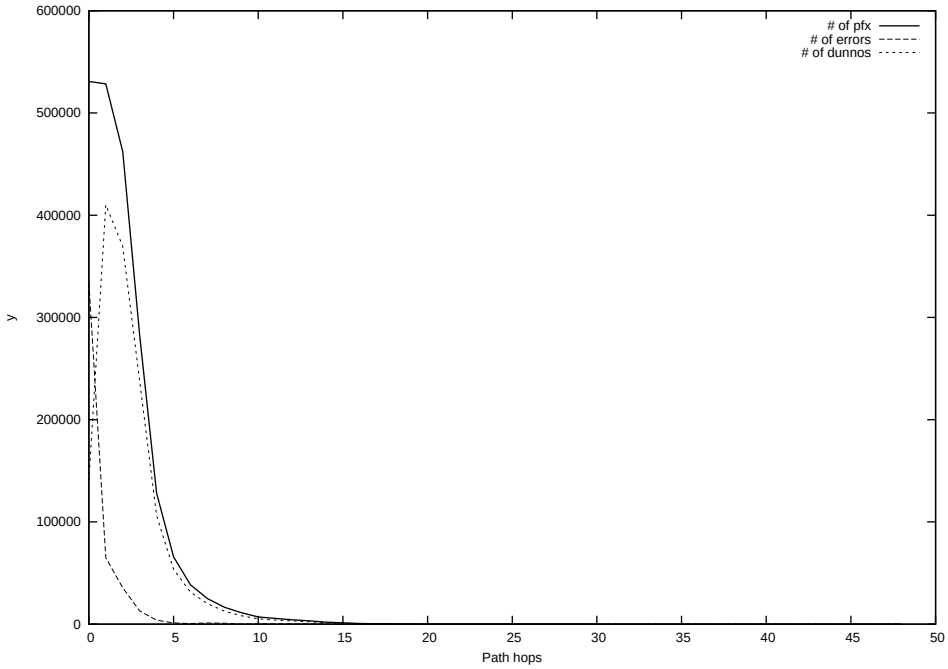
2015-03-09



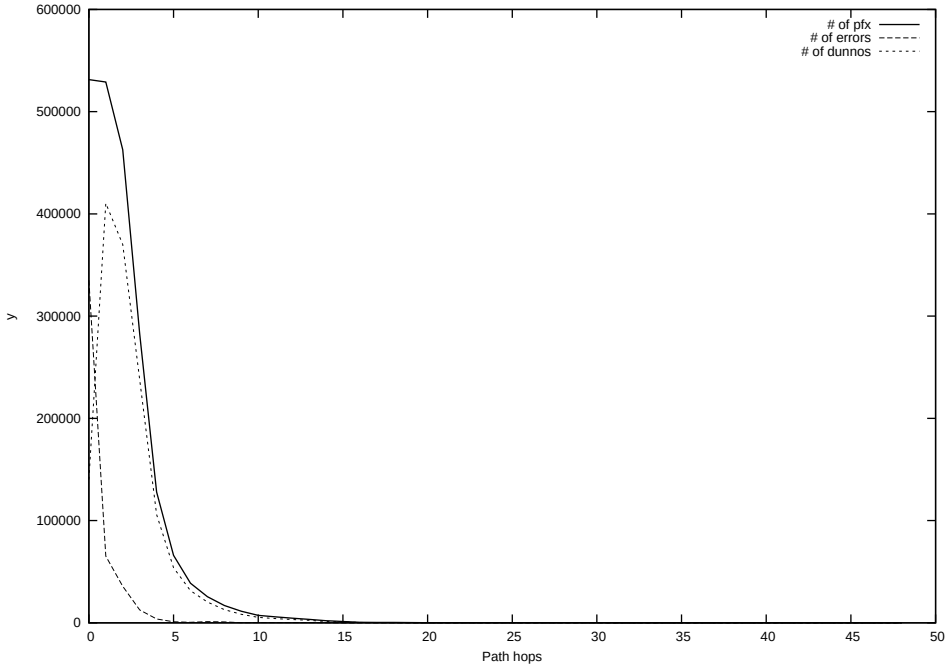
2015-03-10



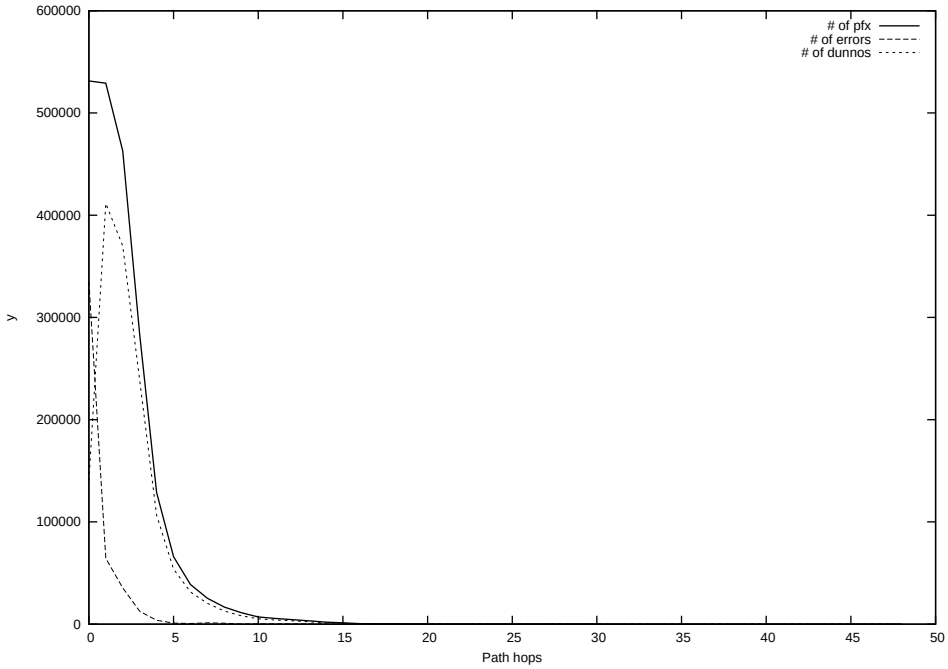
2015-03-11



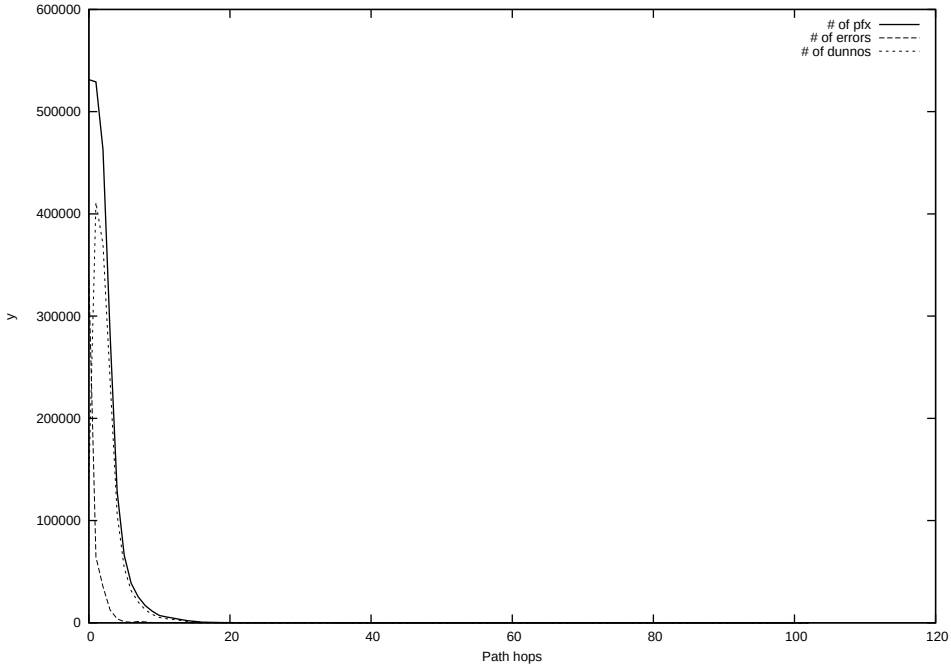
2015-03-12



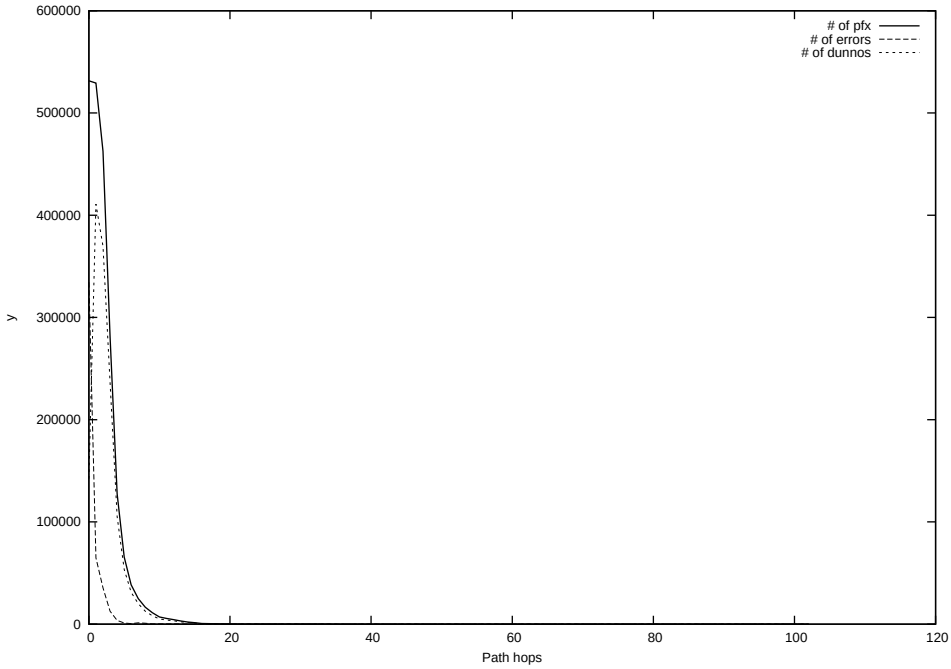
2015-03-13



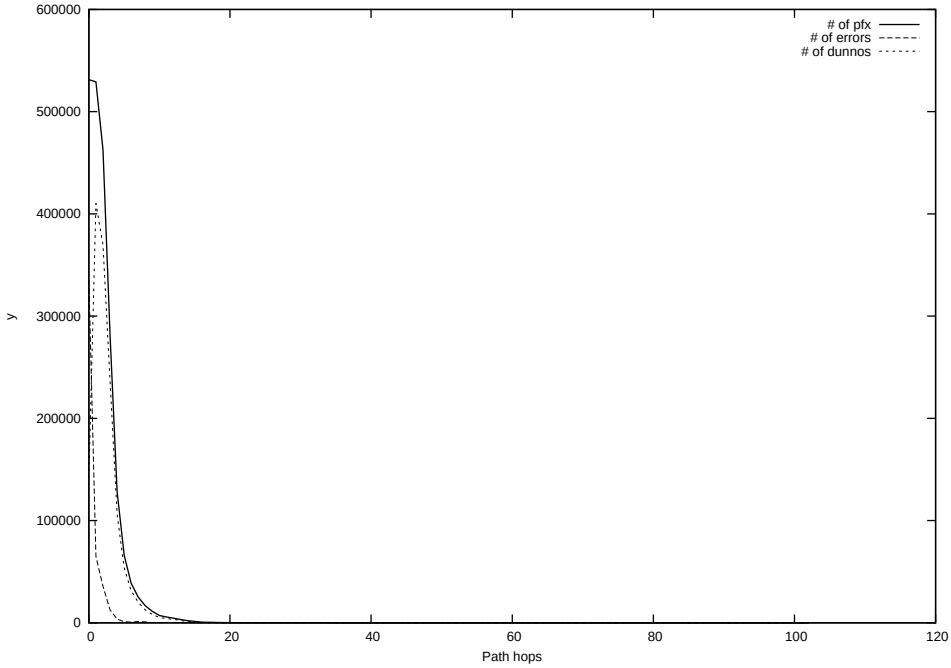
2015-03-14



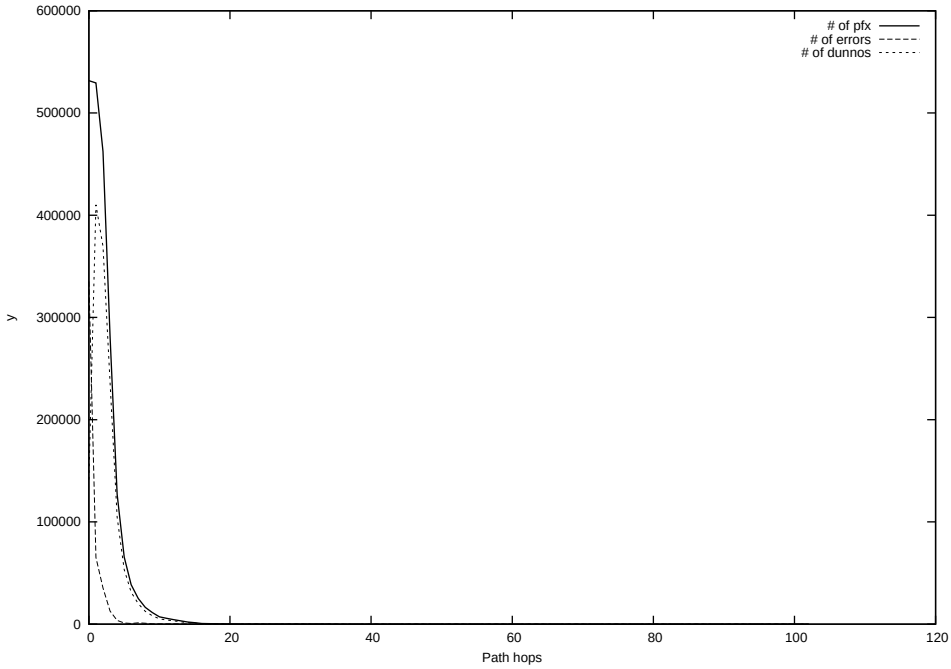
2015-03-15



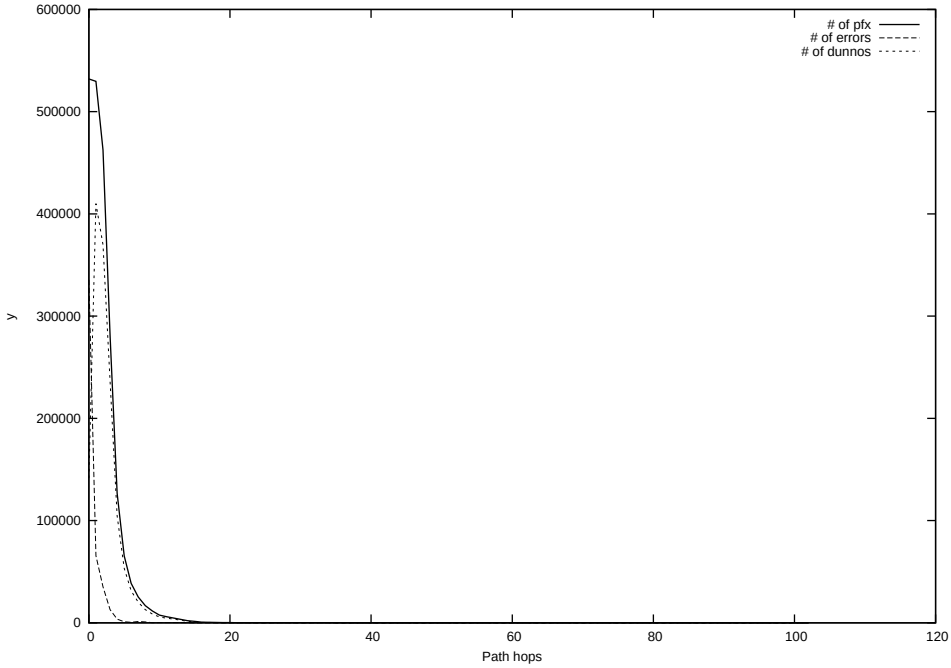
2015-03-16



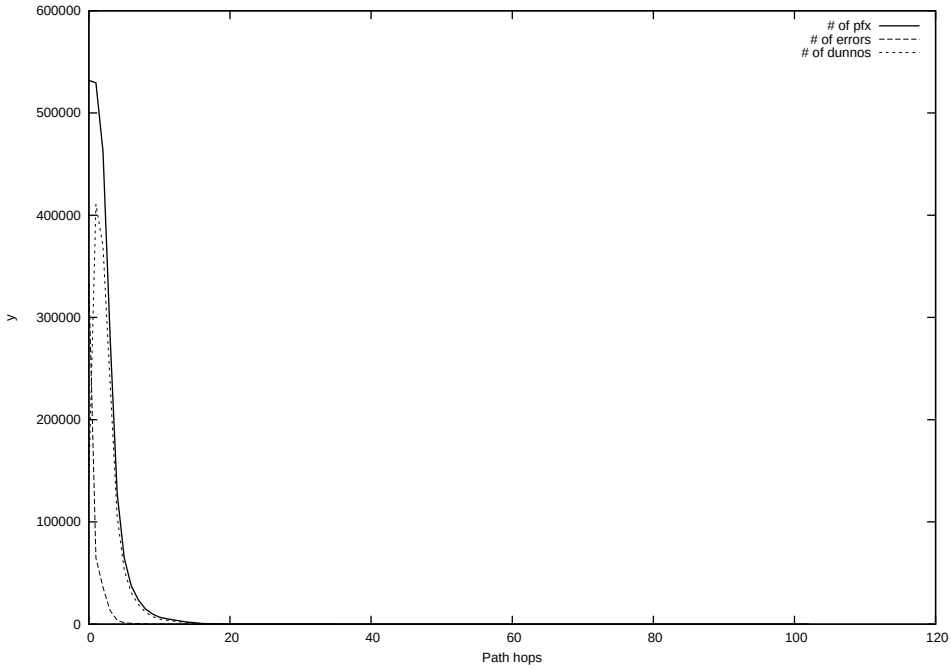
2015-03-17



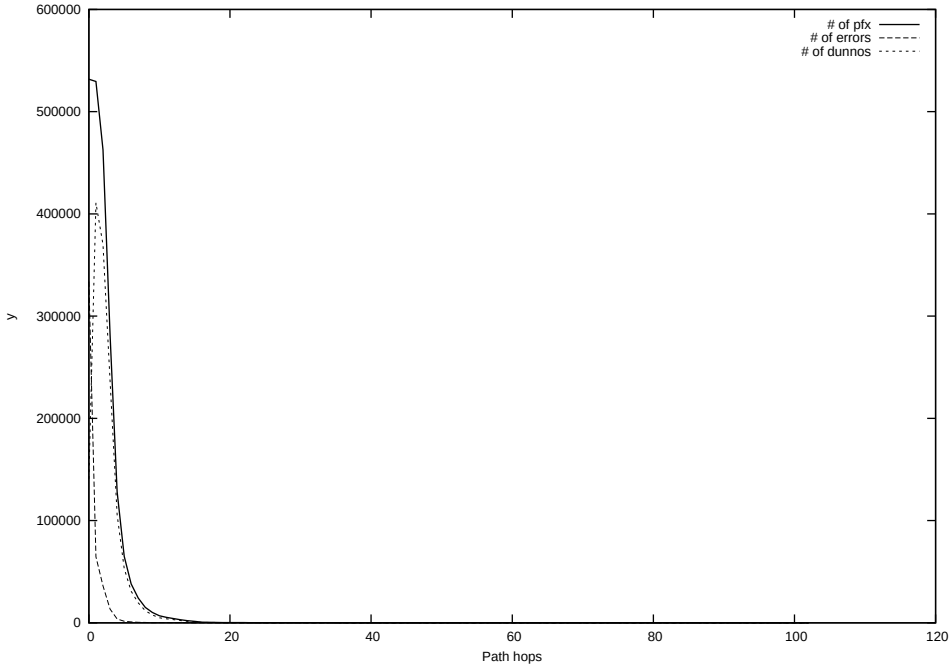
2015-03-18



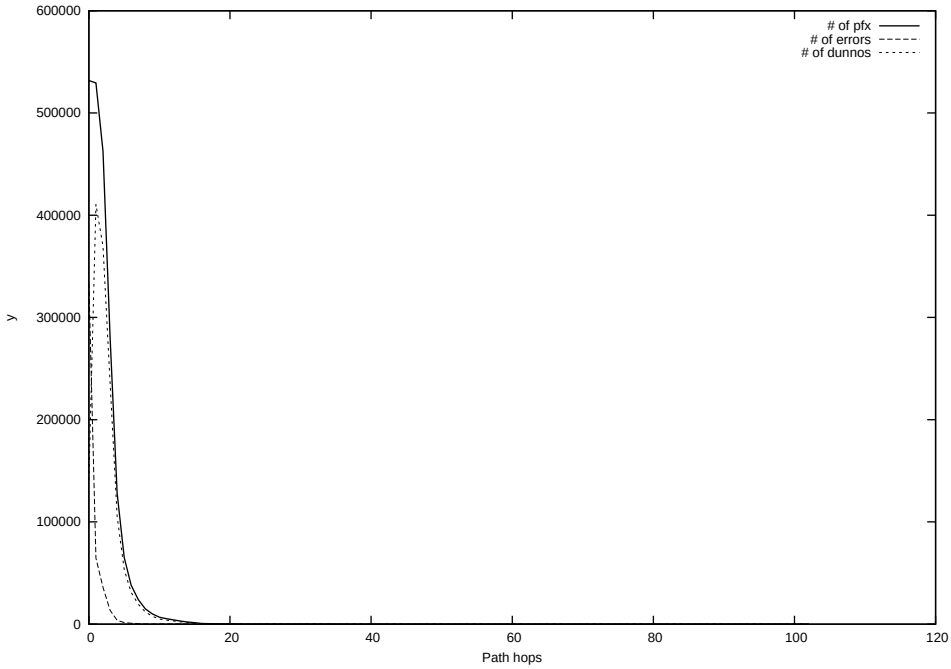
2015-03-19



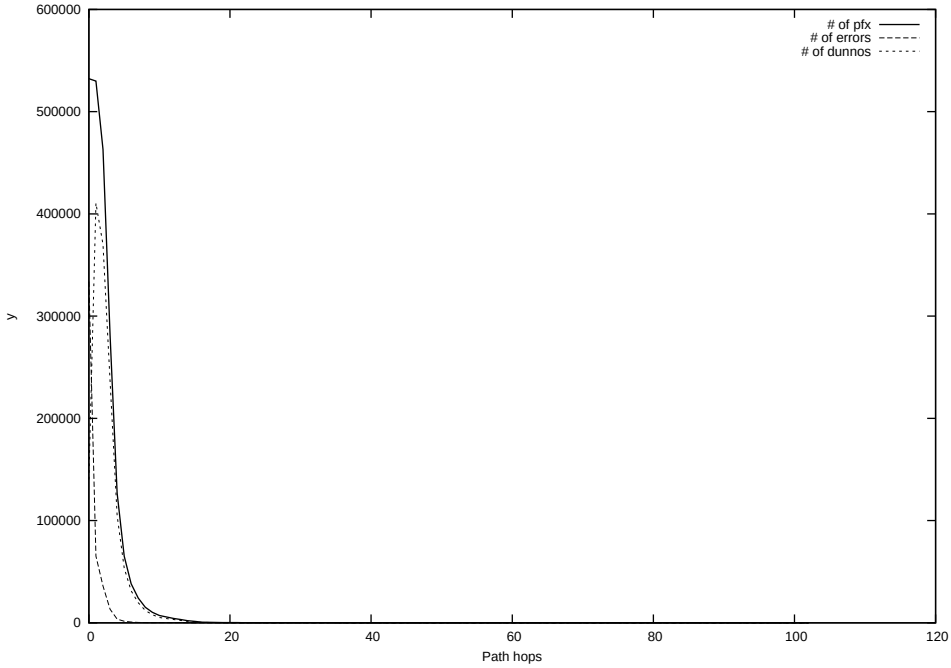
2015-03-20



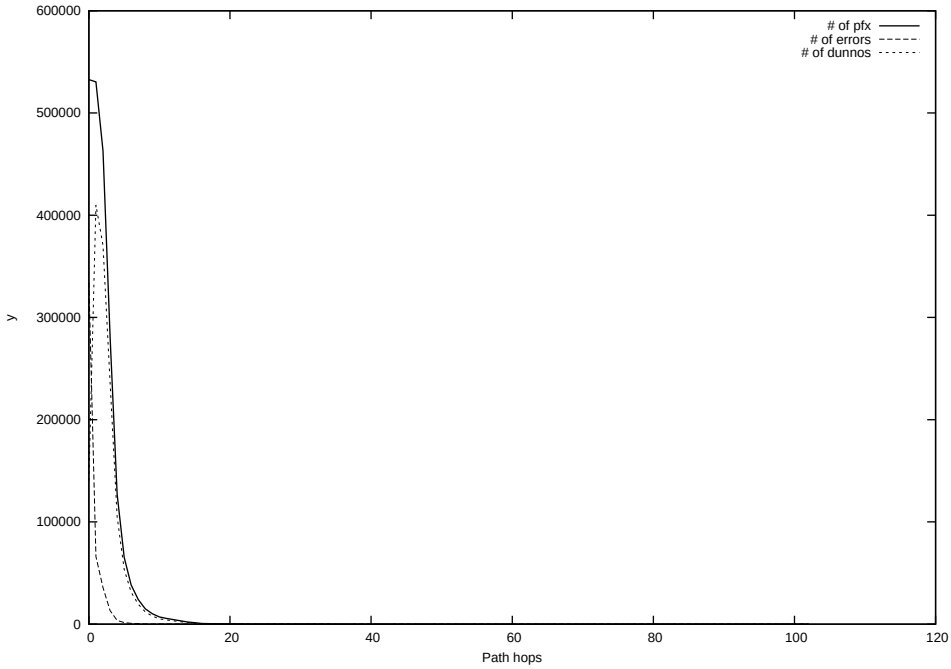
2015-03-21



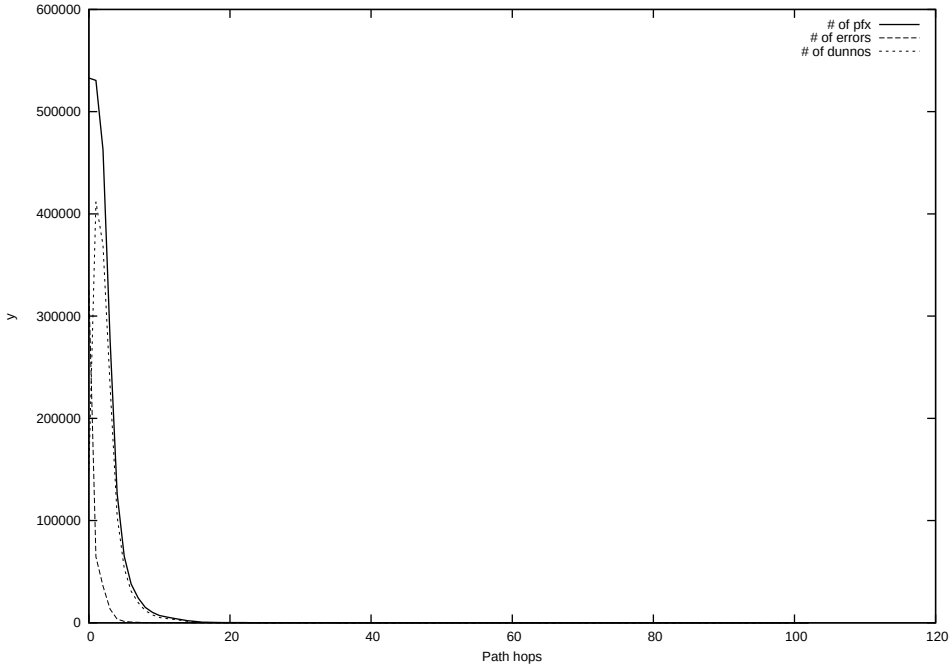
2015-03-22



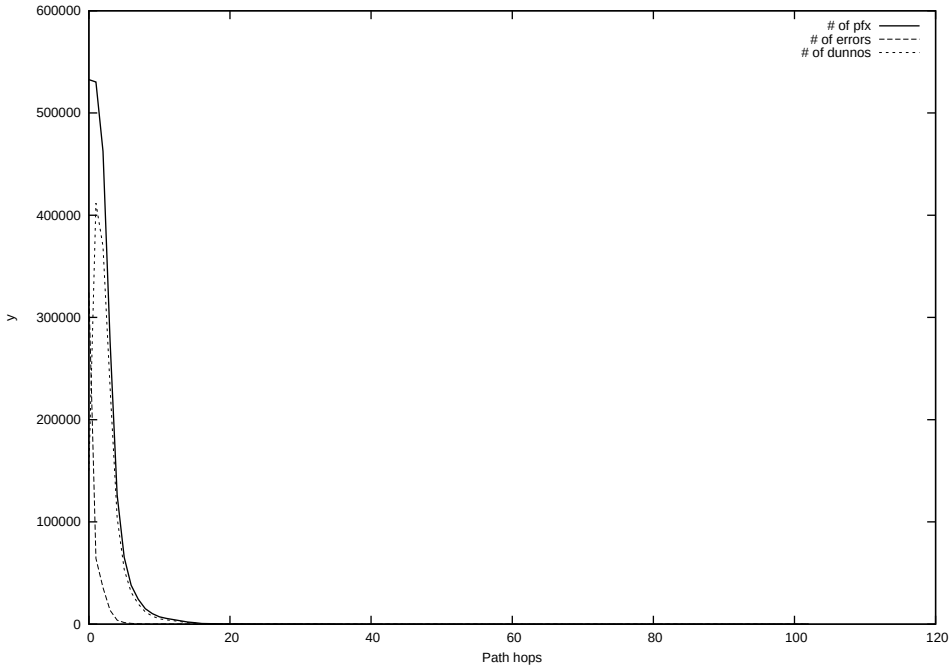
2015-03-23



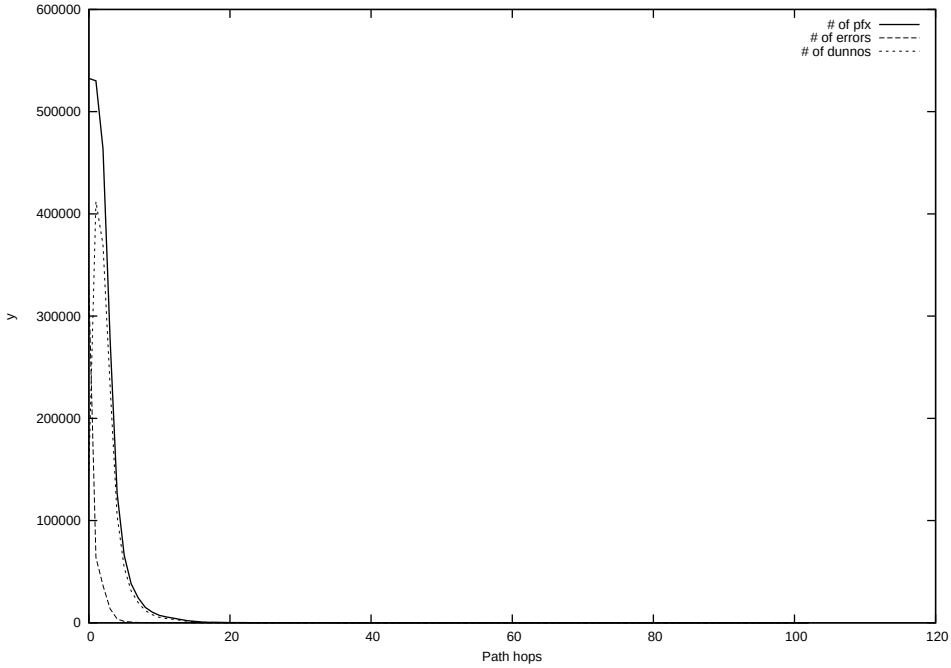
2015-03-24



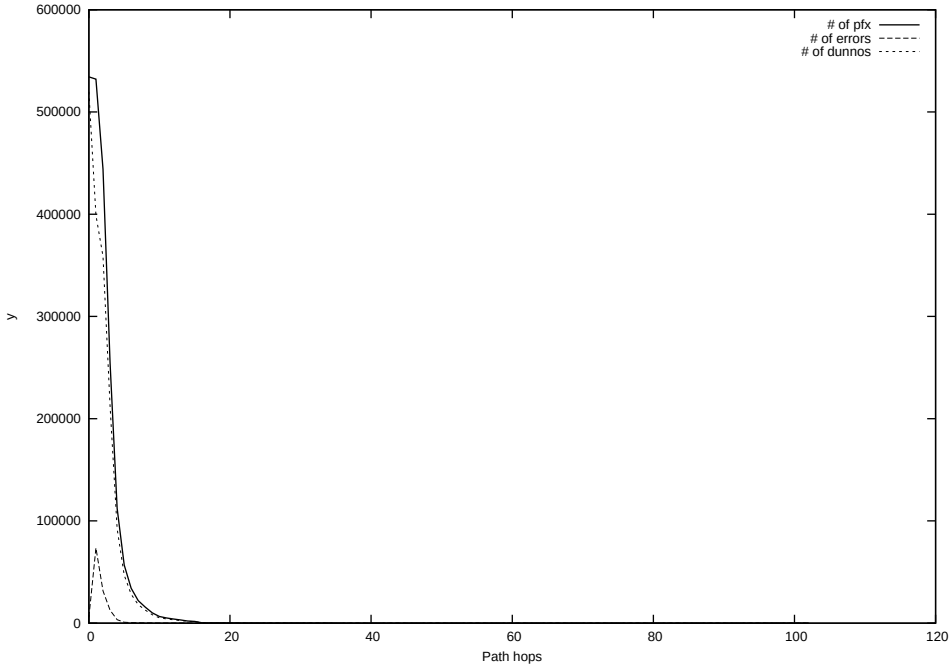
2015-03-25



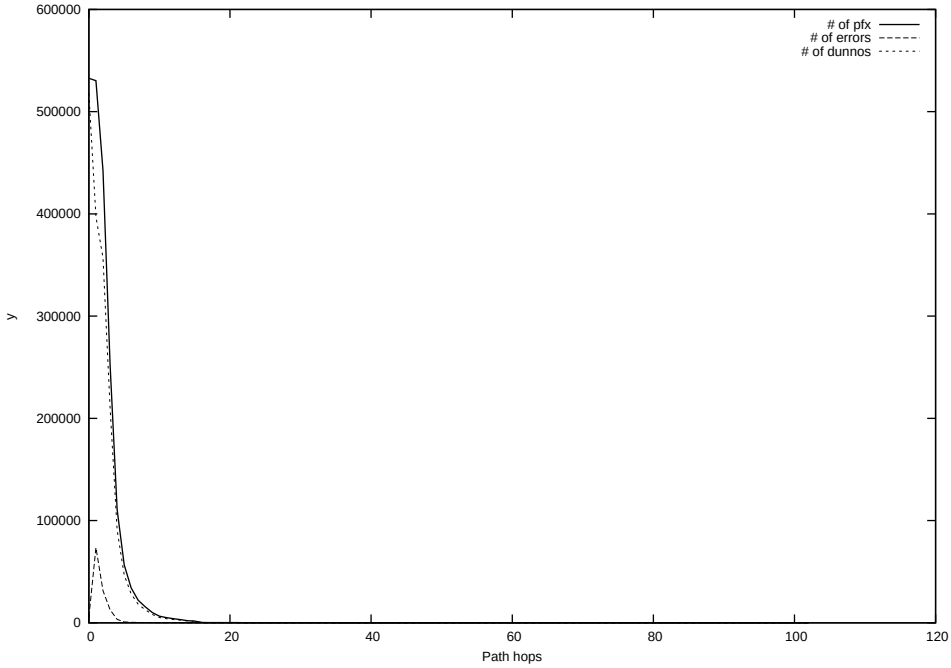
2015-03-26



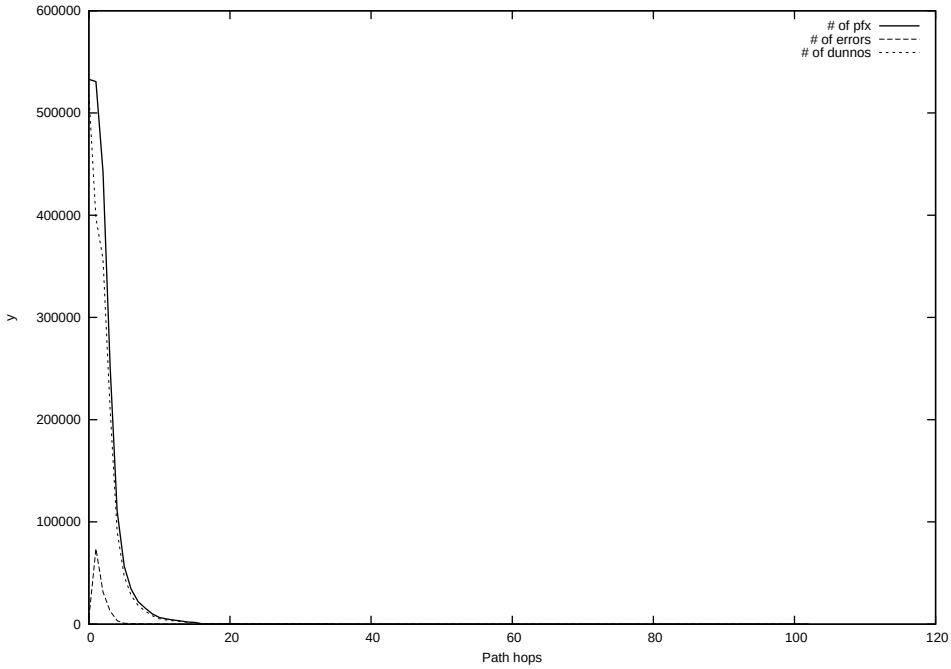
2015-03-27



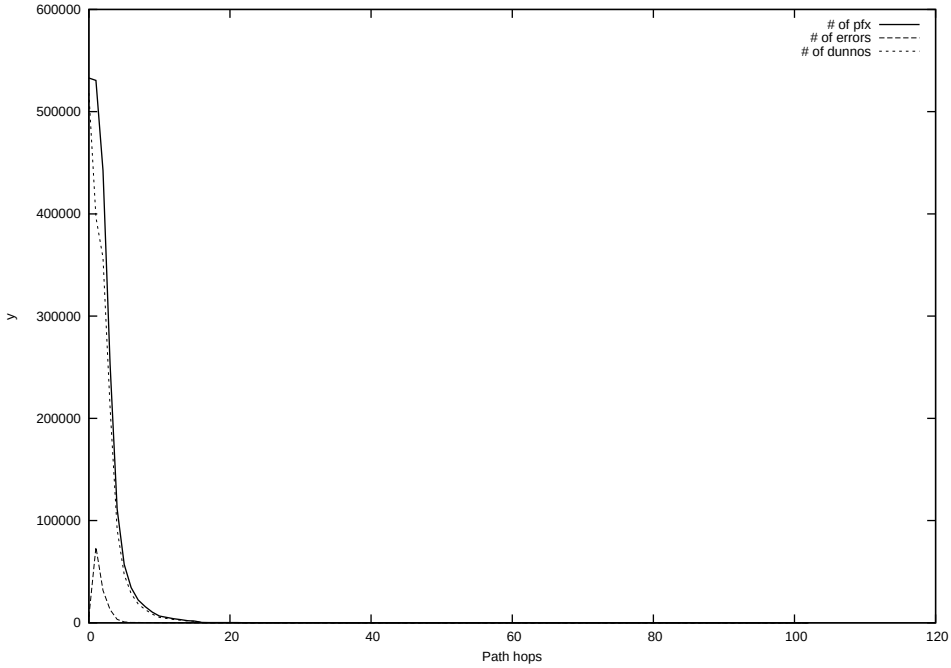
2015-03-28



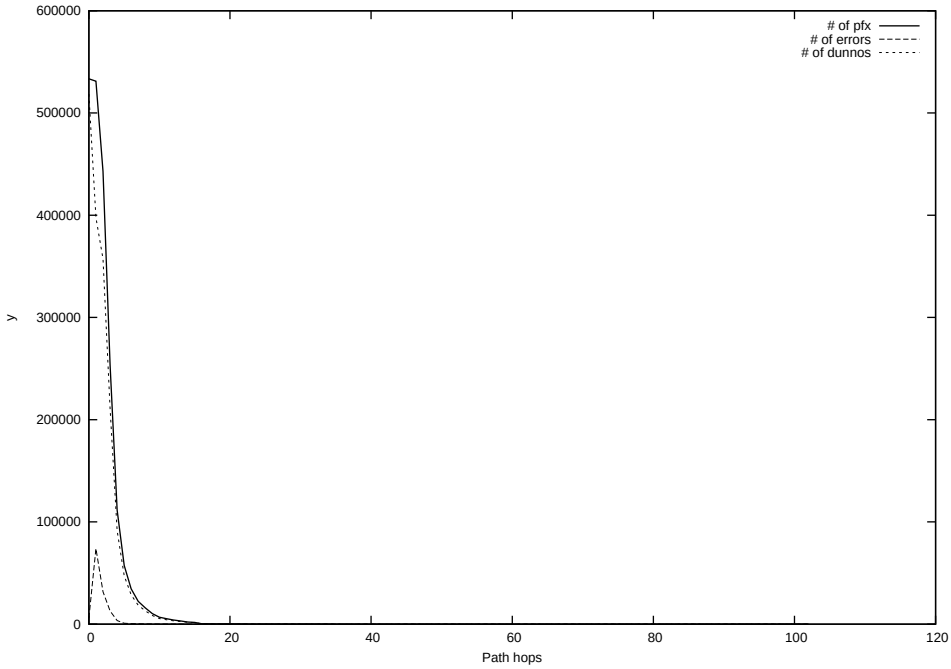
2015-03-29



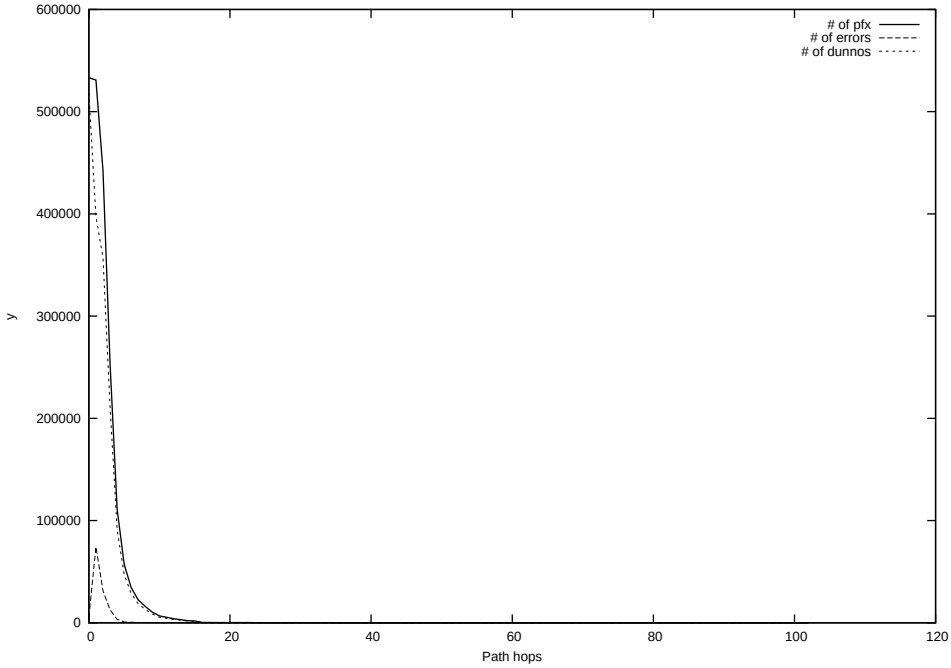
2015-03-30



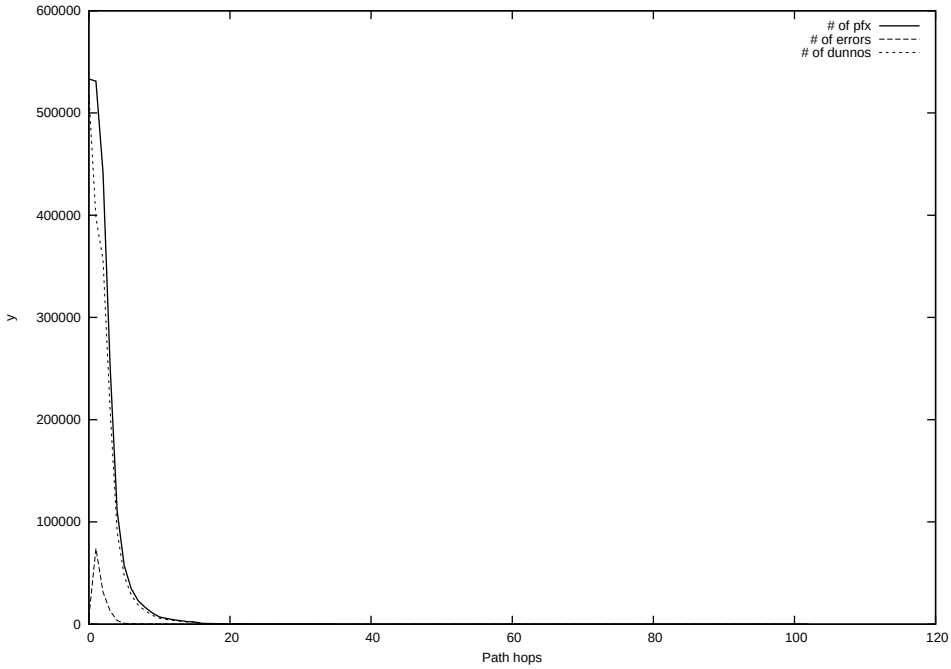
2015-03-31



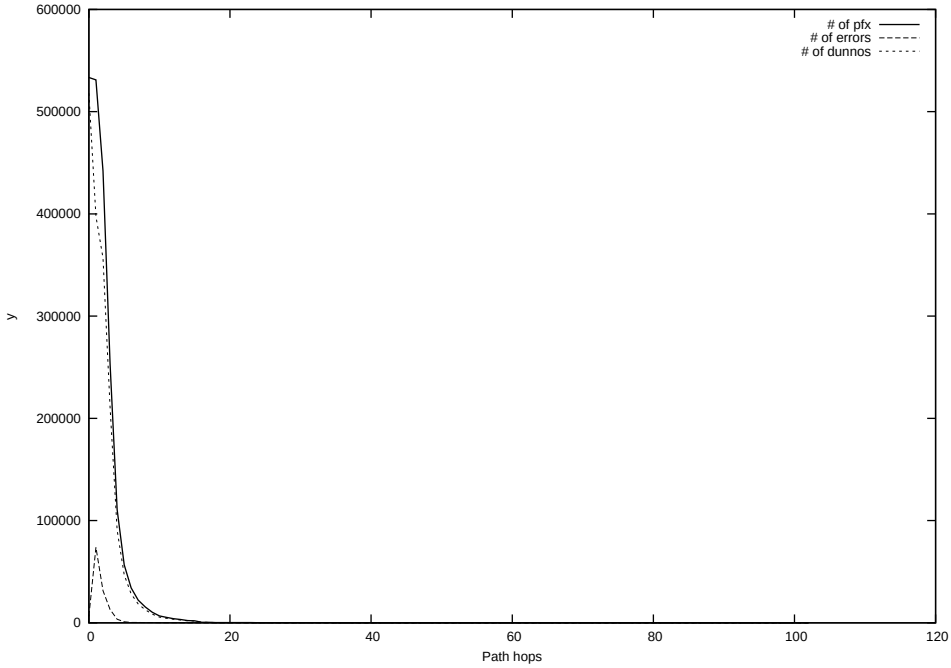
2015-04-01



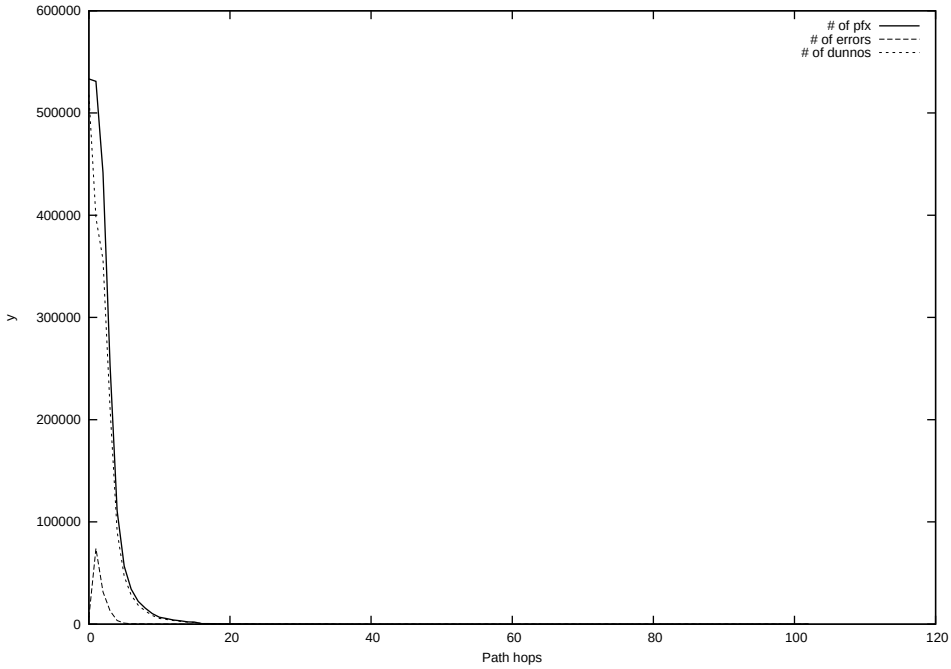
2015-04-02



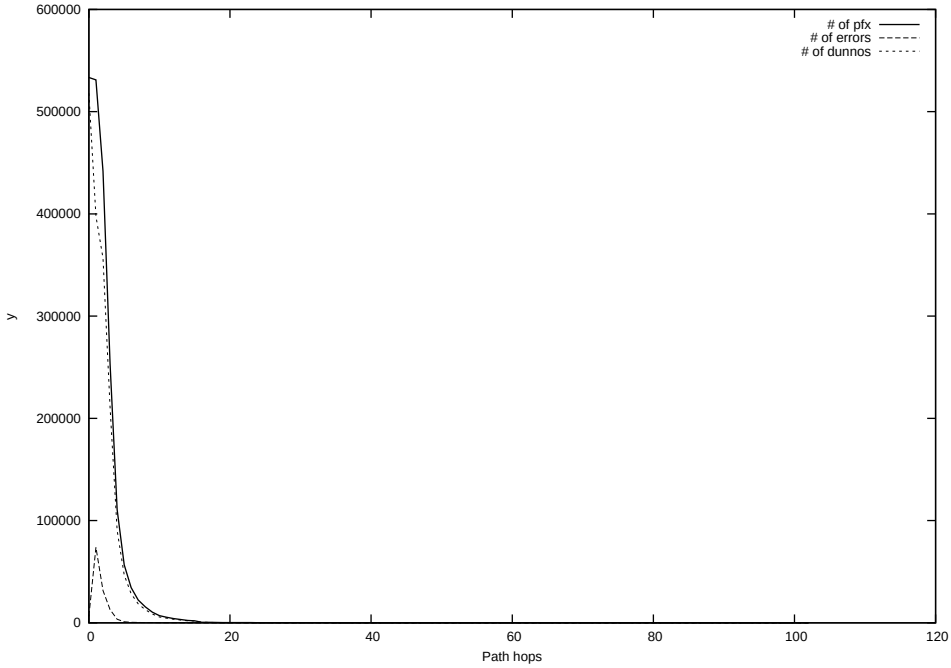
2015-04-03



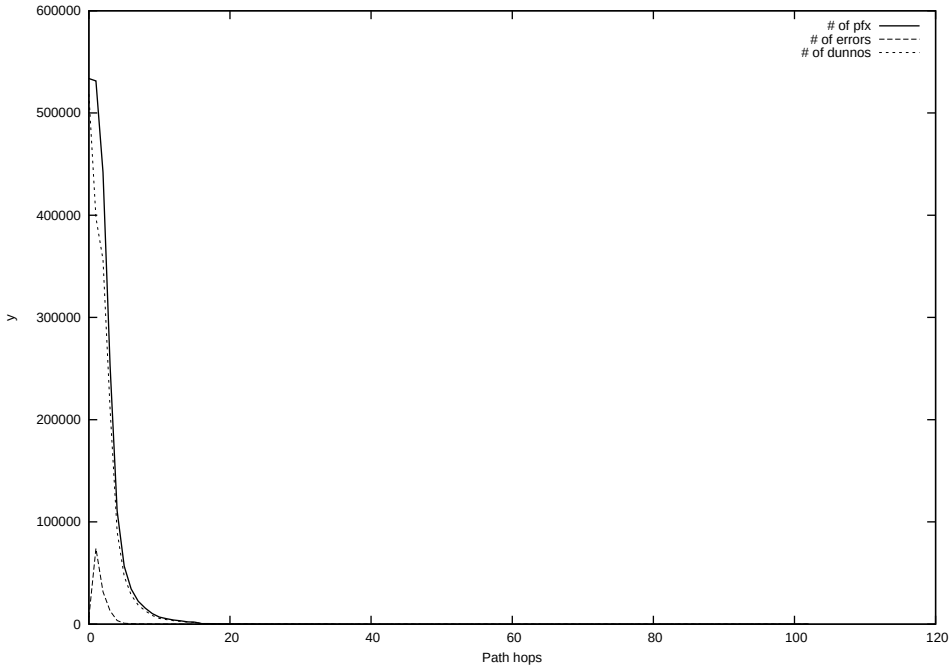
2015-04-04



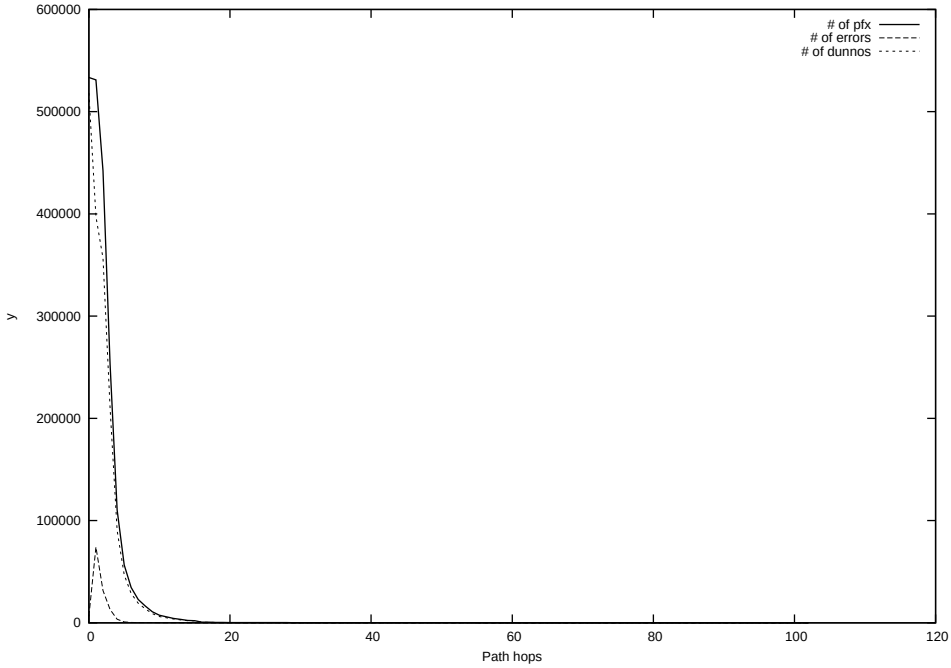
2015-04-05



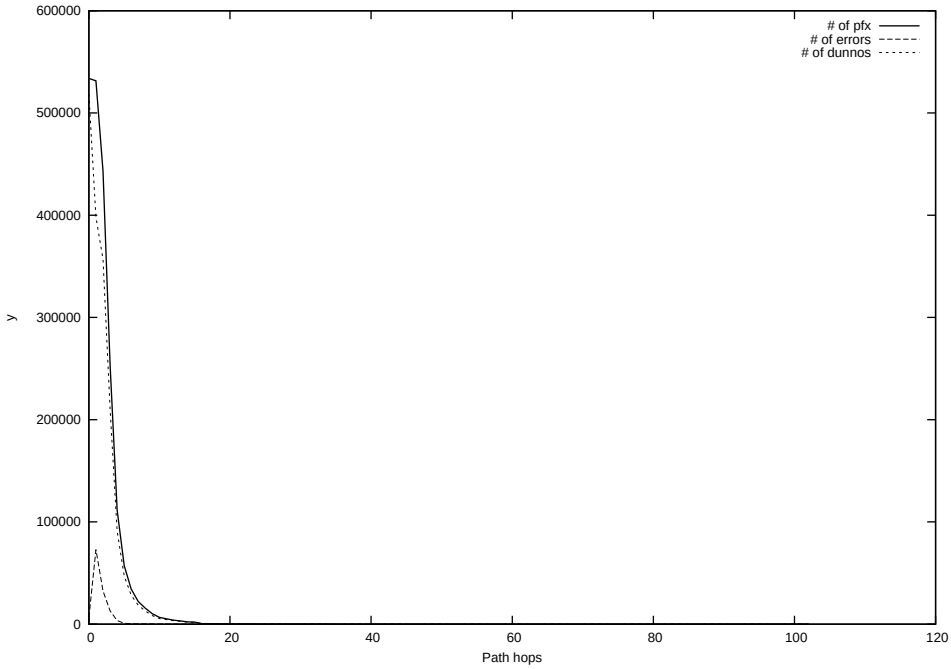
2015-04-06



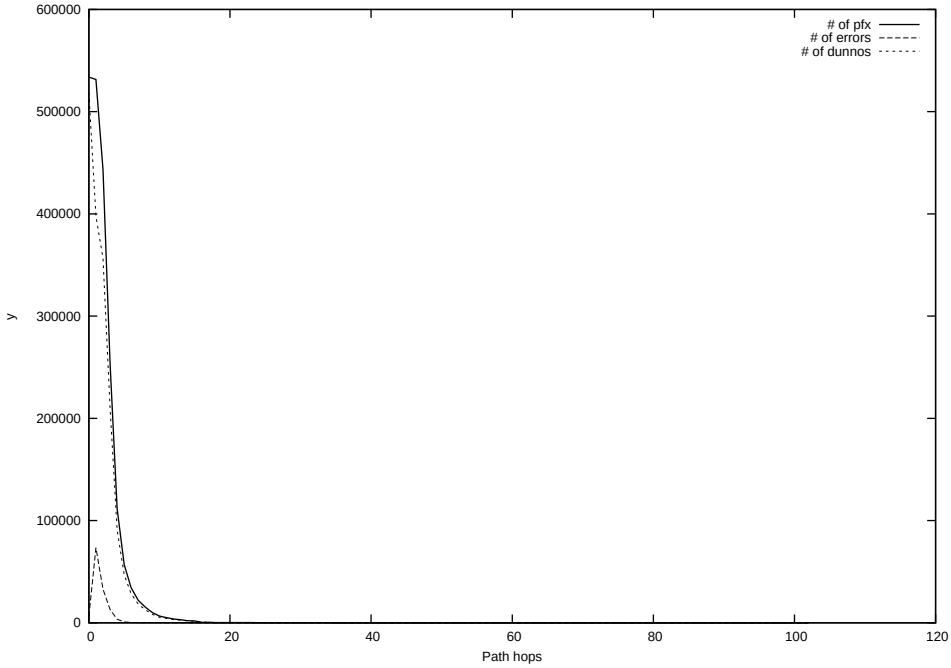
2015-04-07



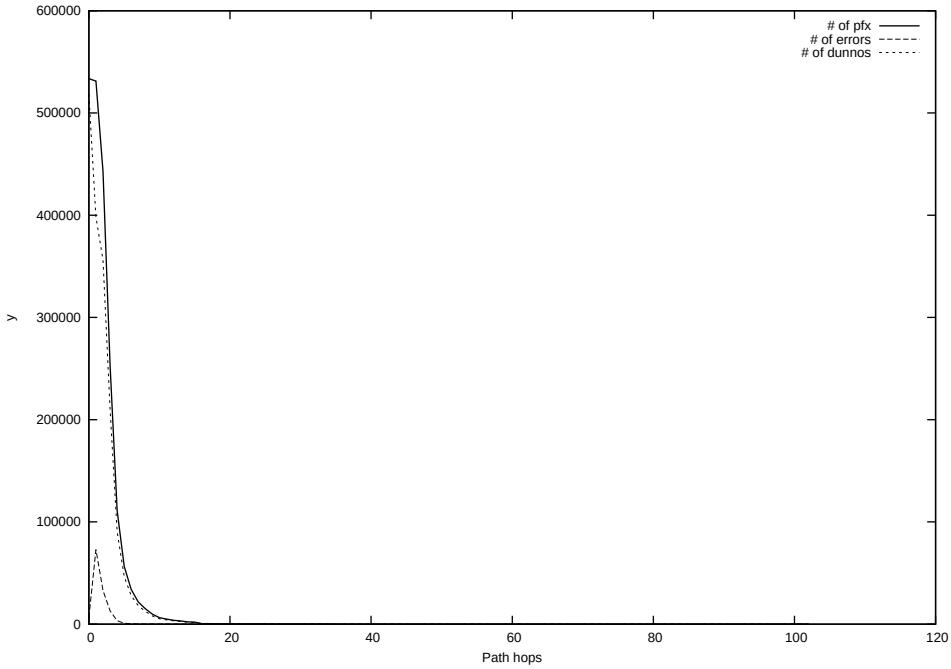
2015-04-08



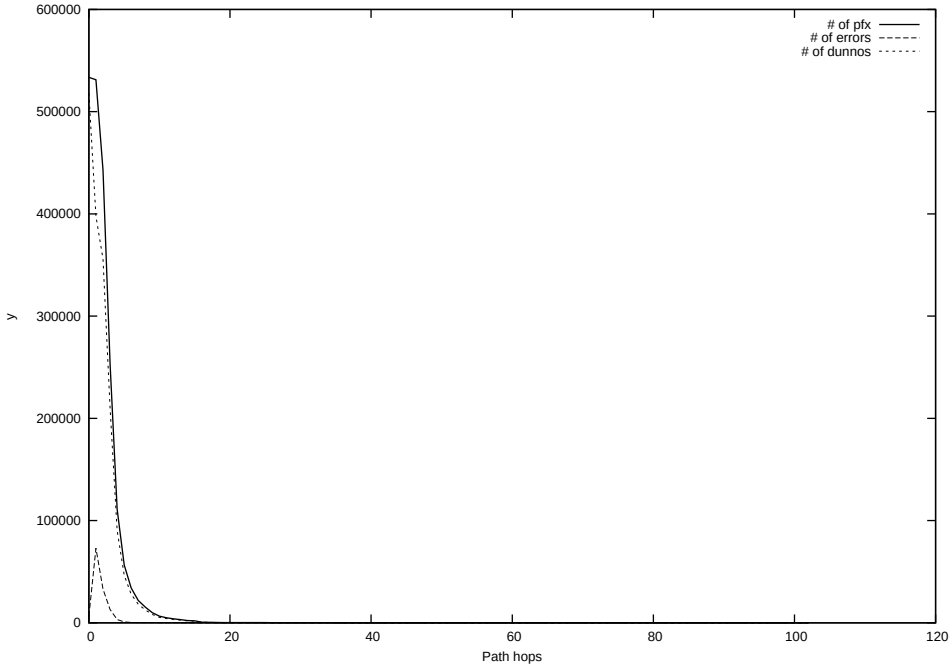
2015-04-09



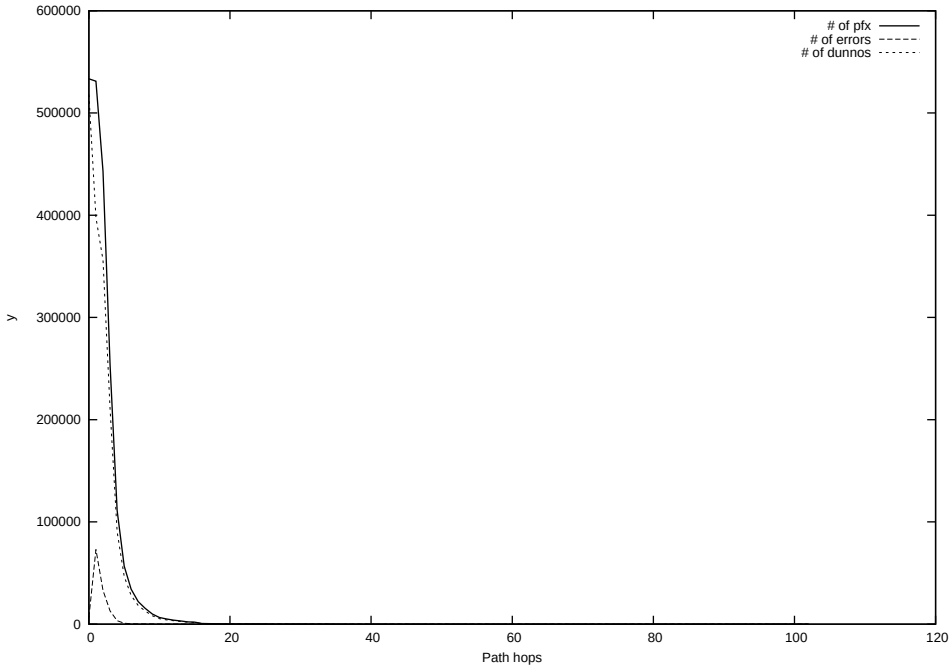
2015-04-10



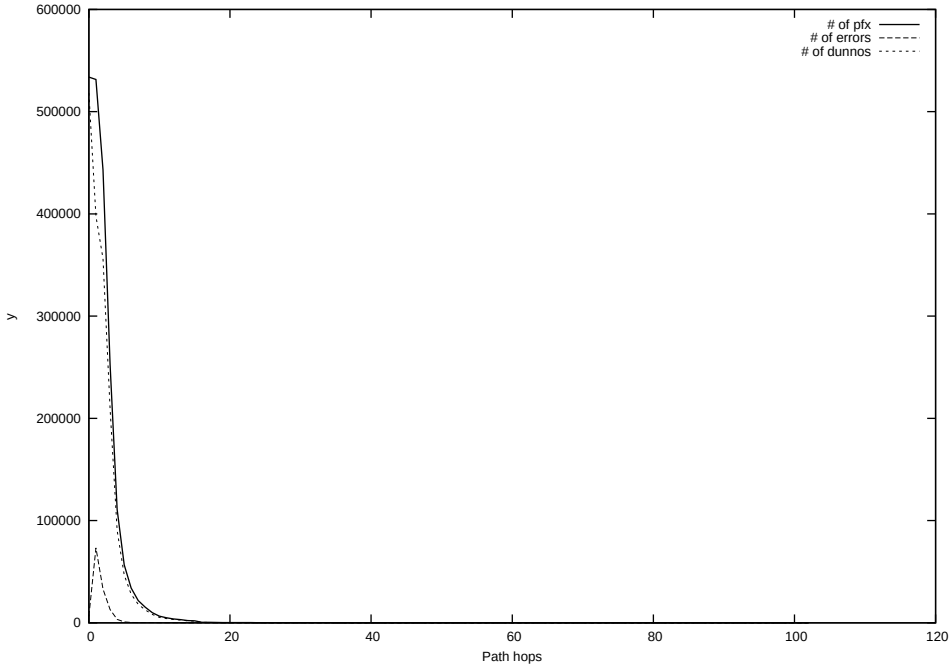
2015-04-11



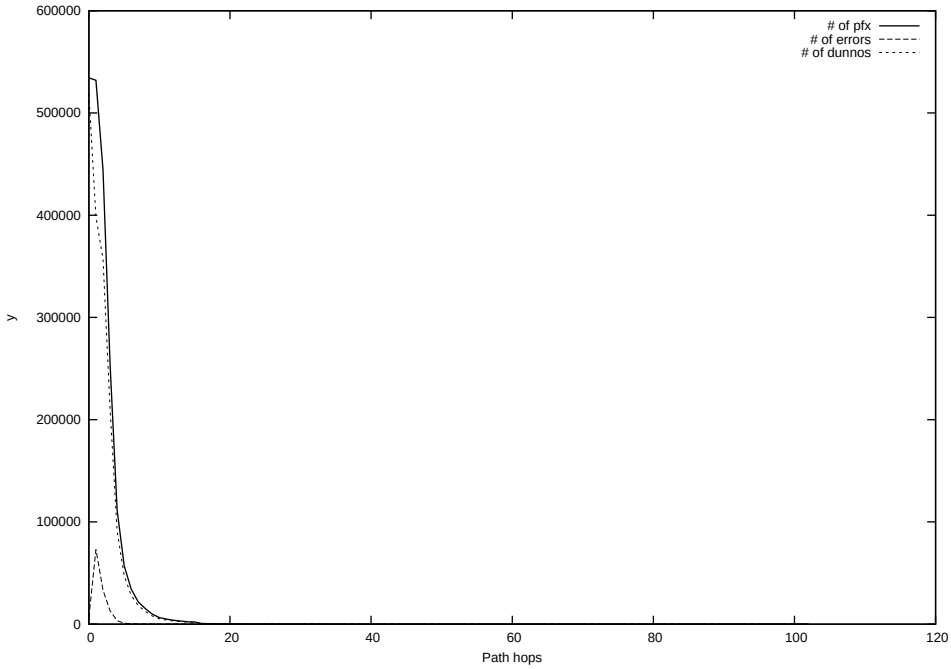
2015-04-12



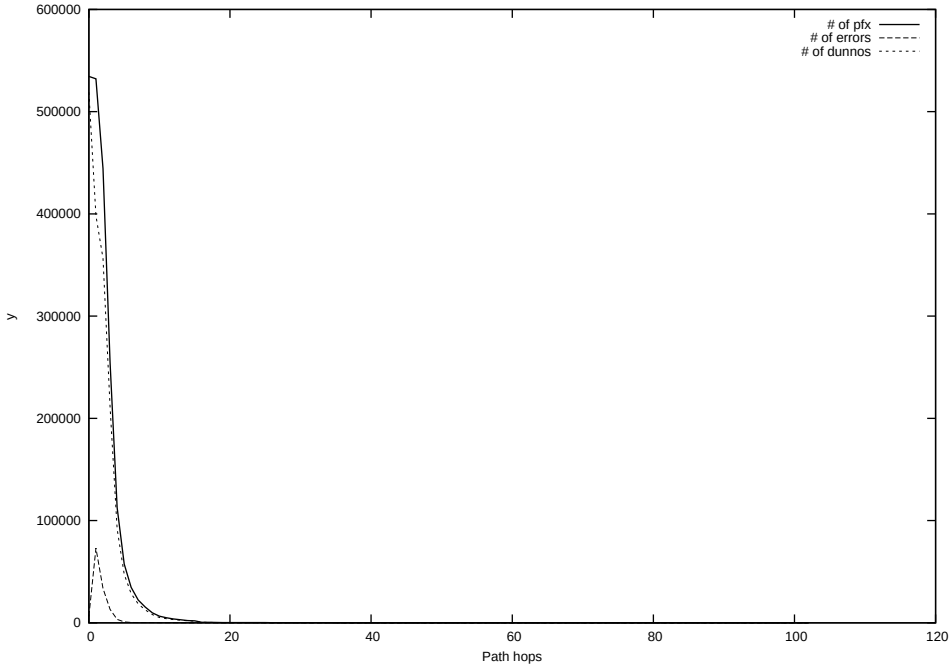
2015-04-13



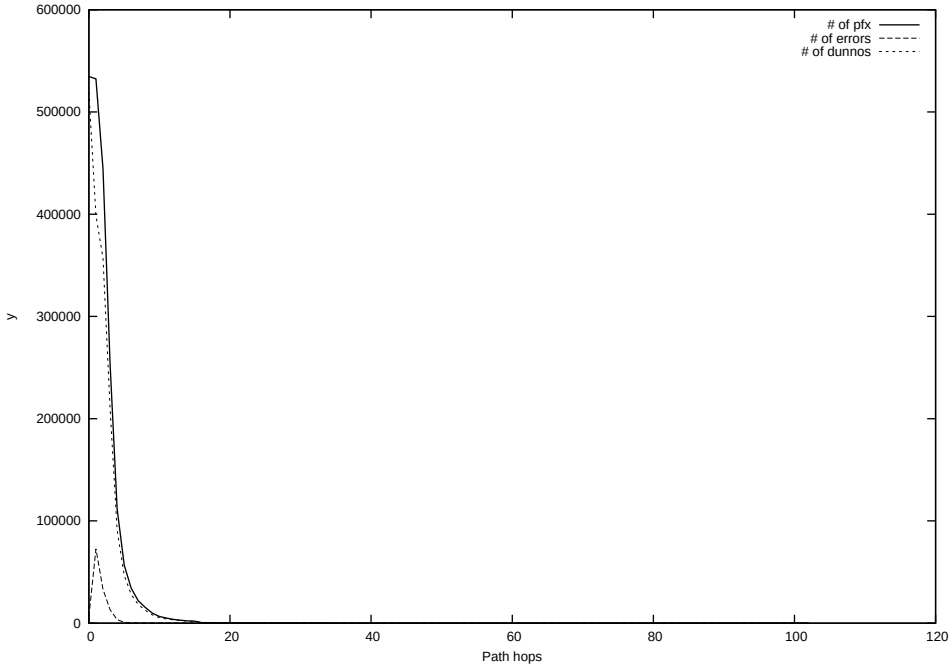
2015-04-14



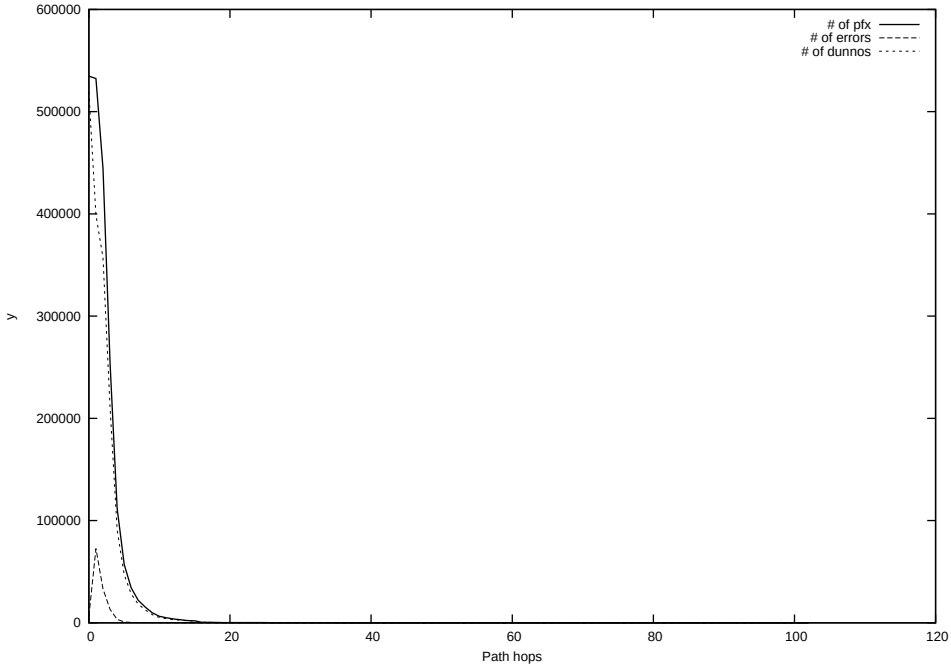
2015-04-15



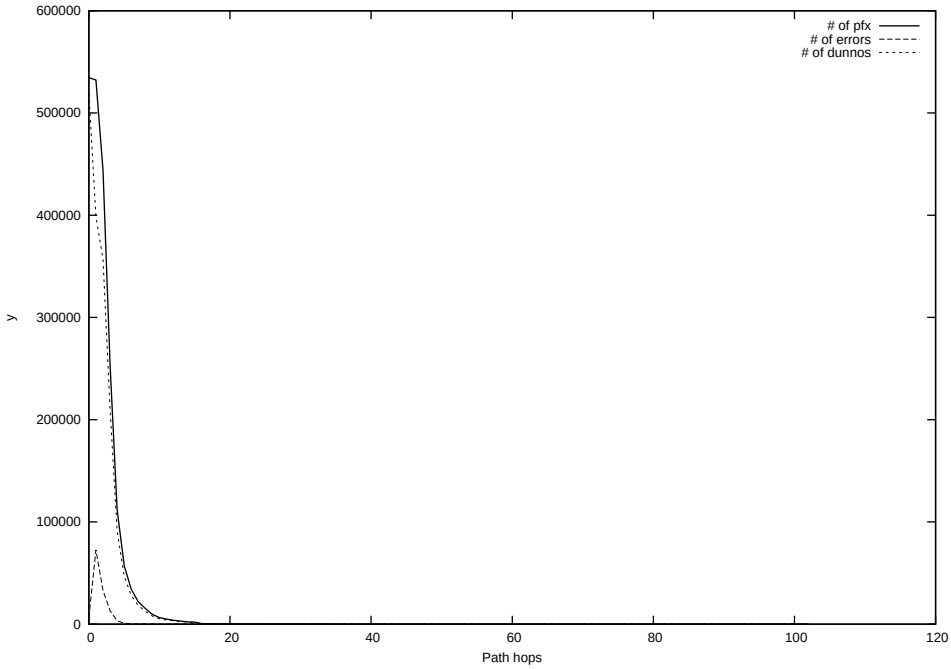
2015-04-16



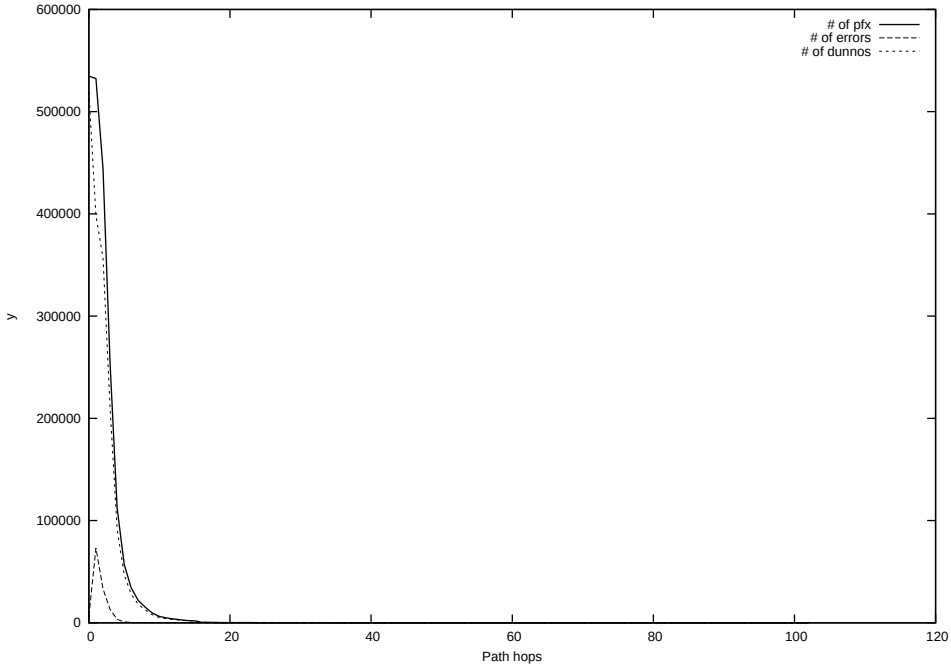
2015-04-17



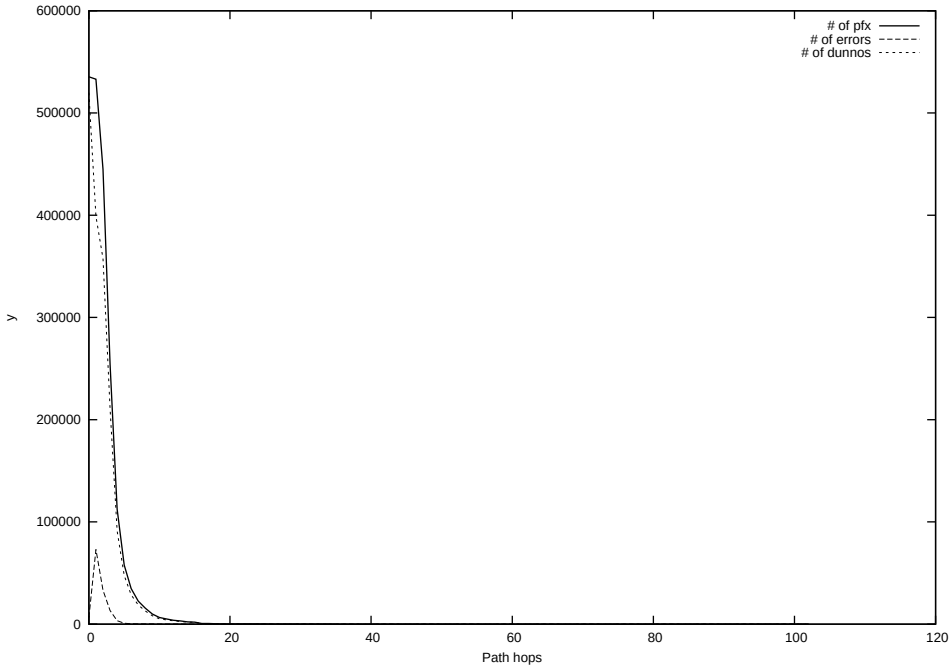
2015-04-18



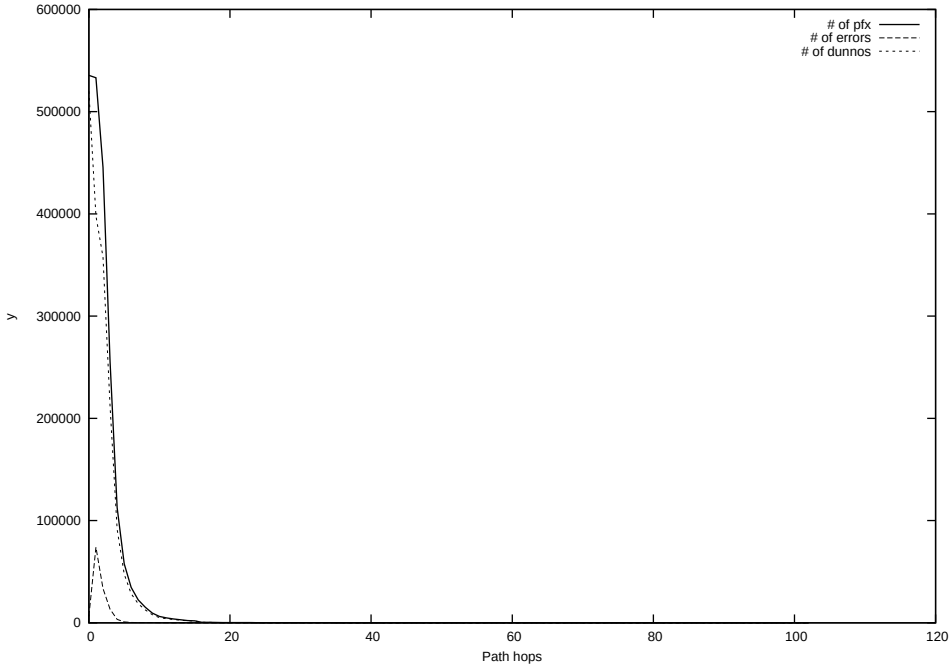
2015-04-19



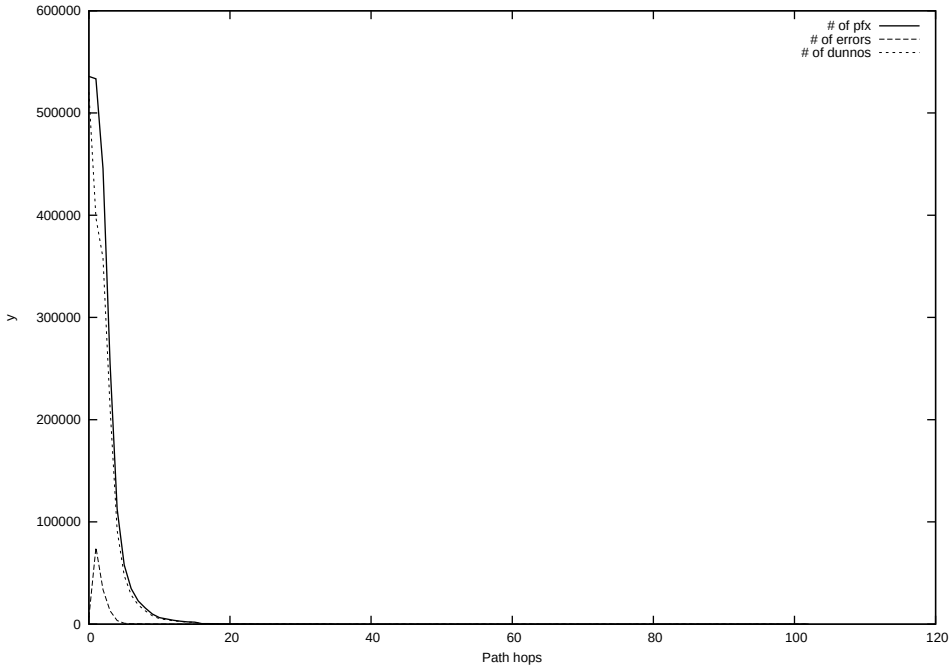
2015-04-20



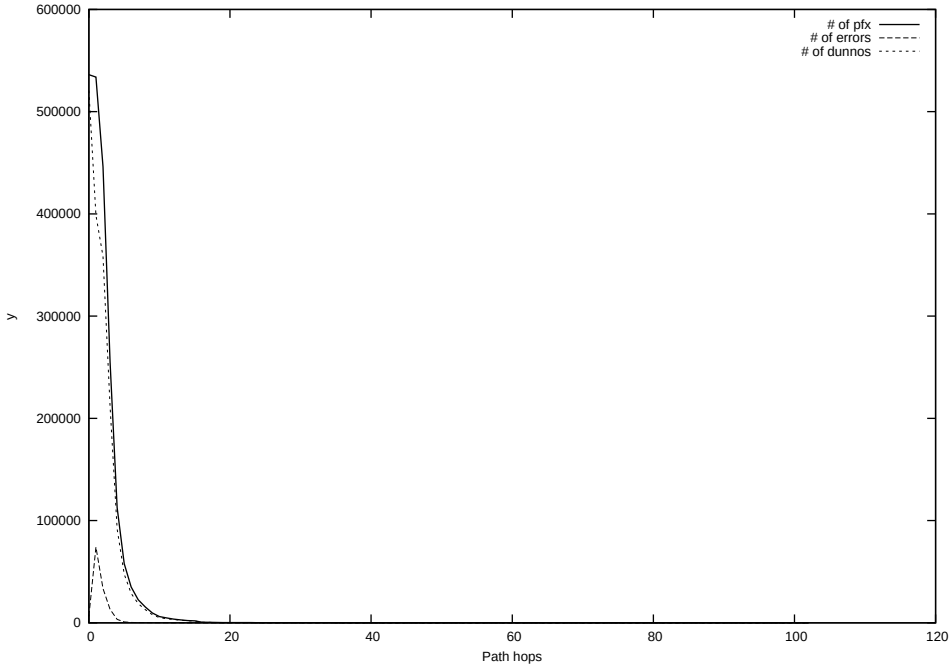
2015-04-21



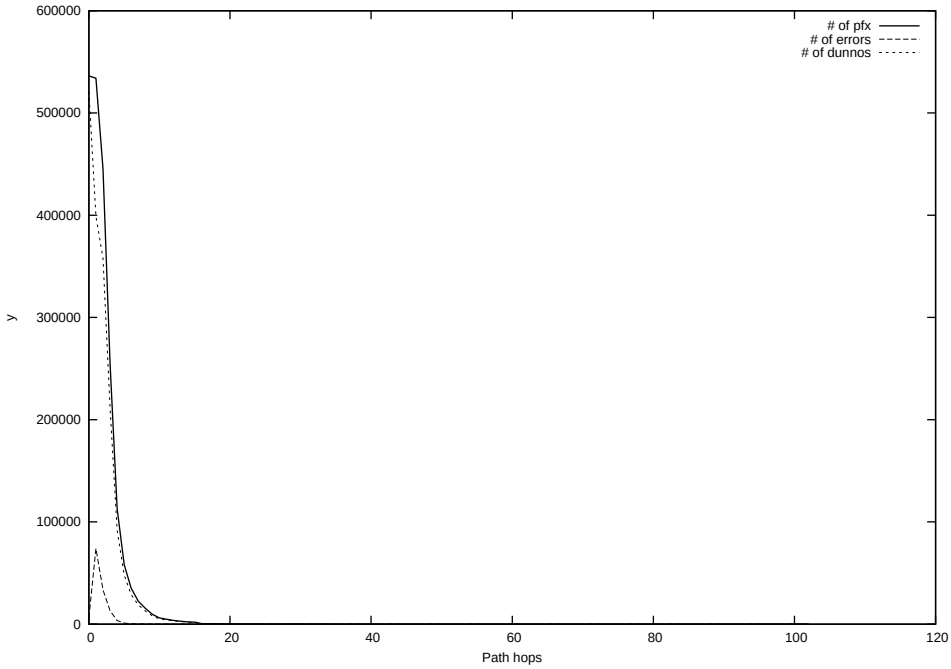
2015-04-22



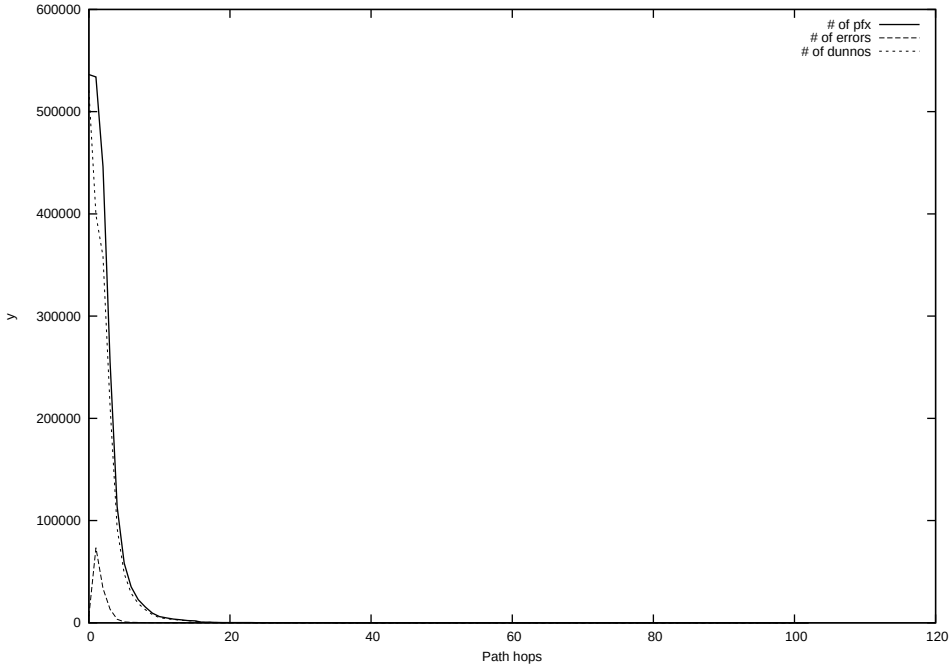
2015-04-23



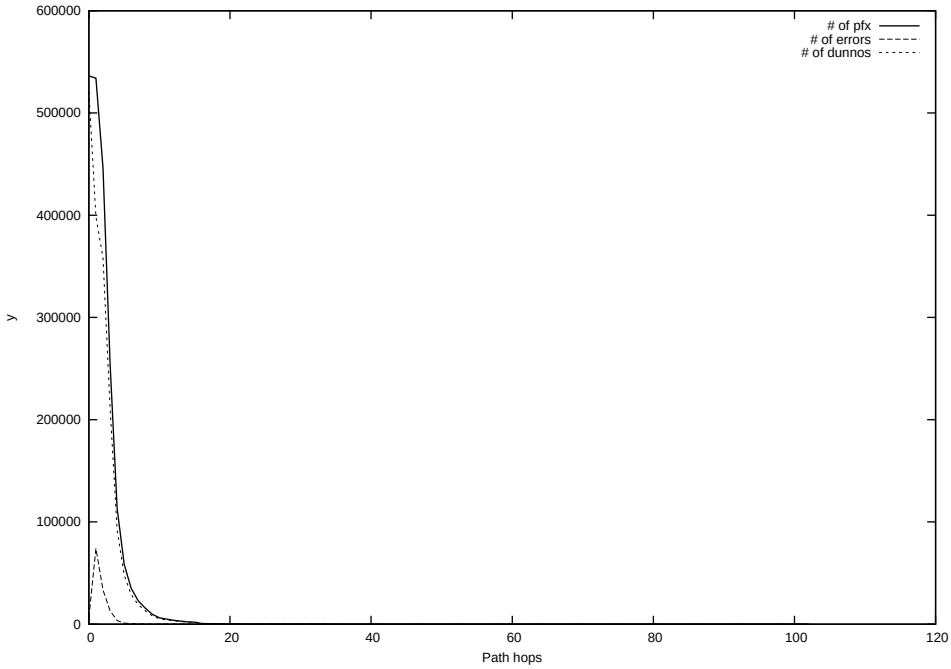
2015-04-24



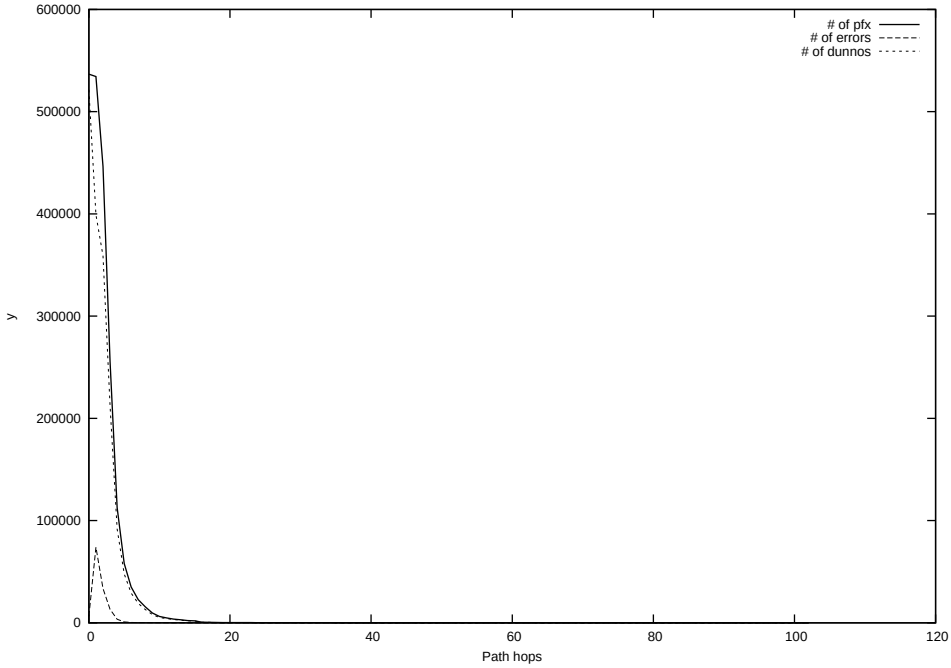
2015-04-25



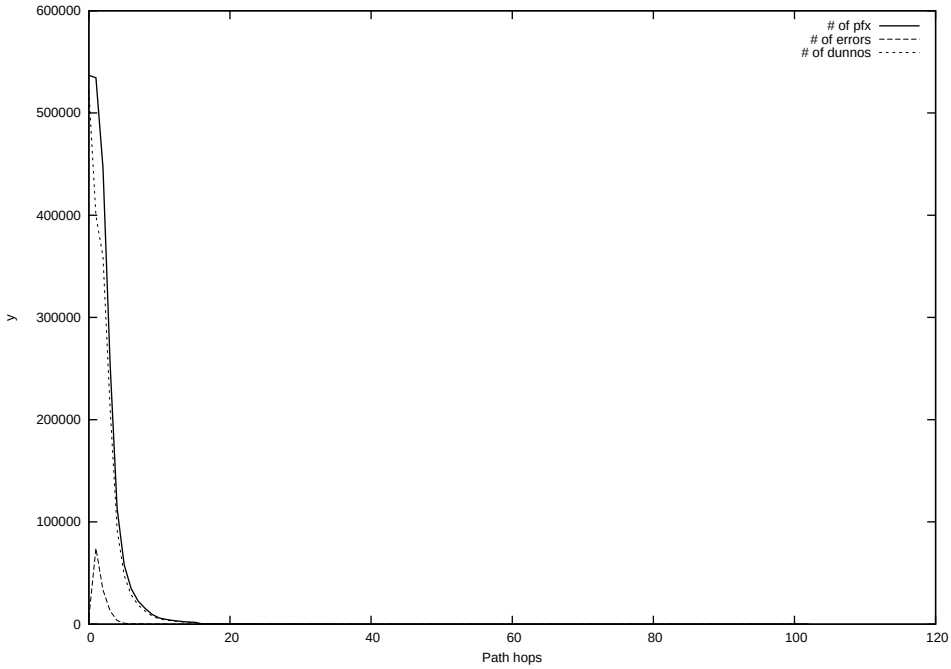
2015-04-26



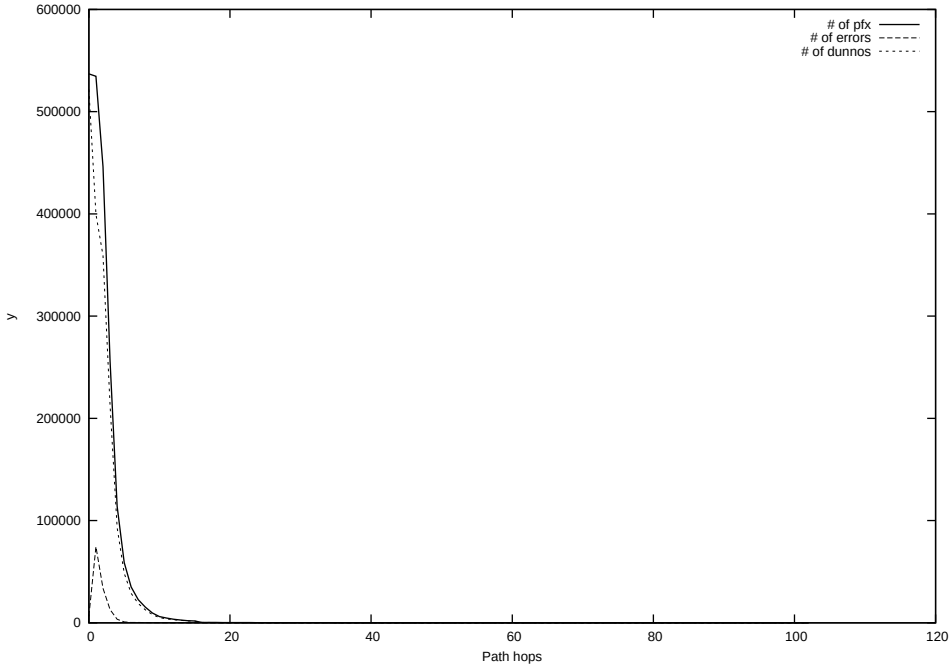
2015-04-27



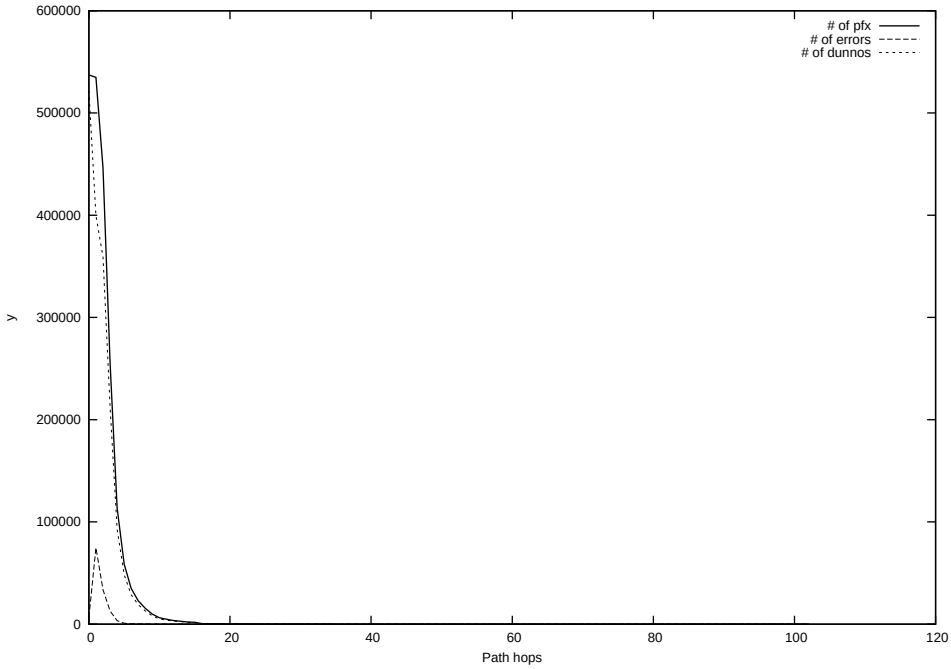
2015-04-28



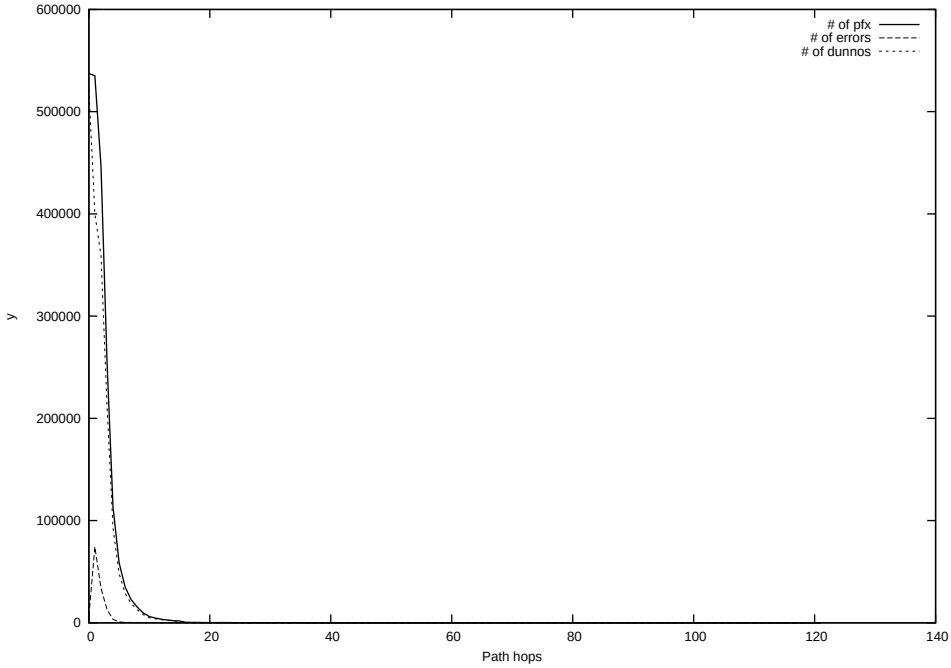
2015-04-29



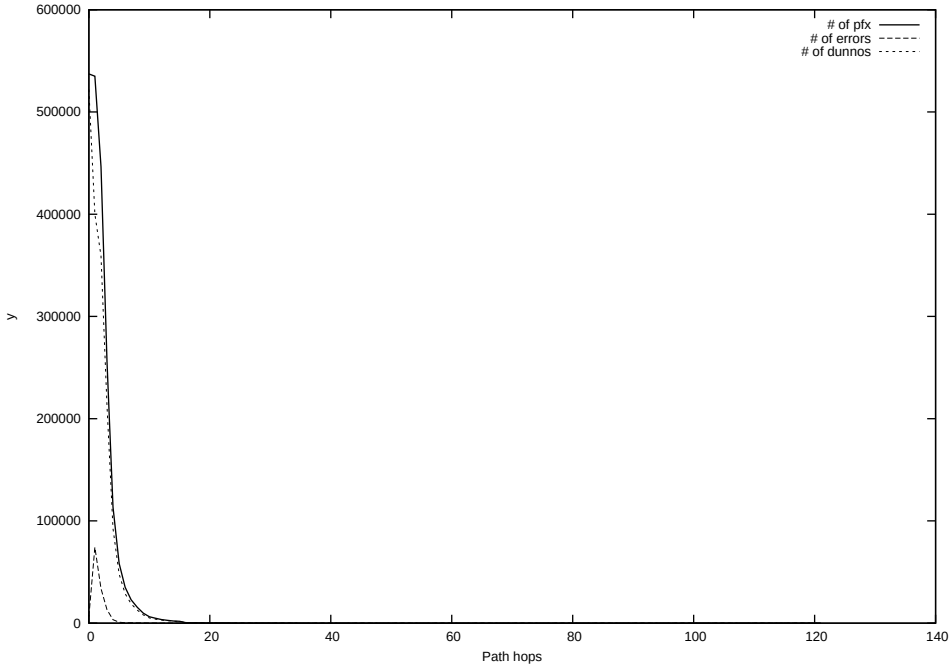
2015-04-30



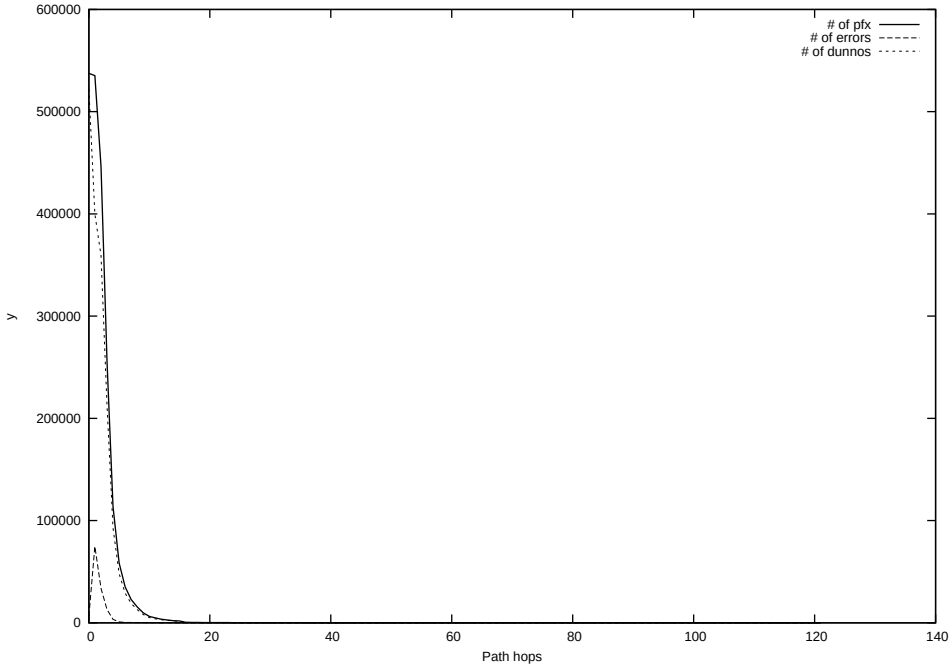
2015-05-01



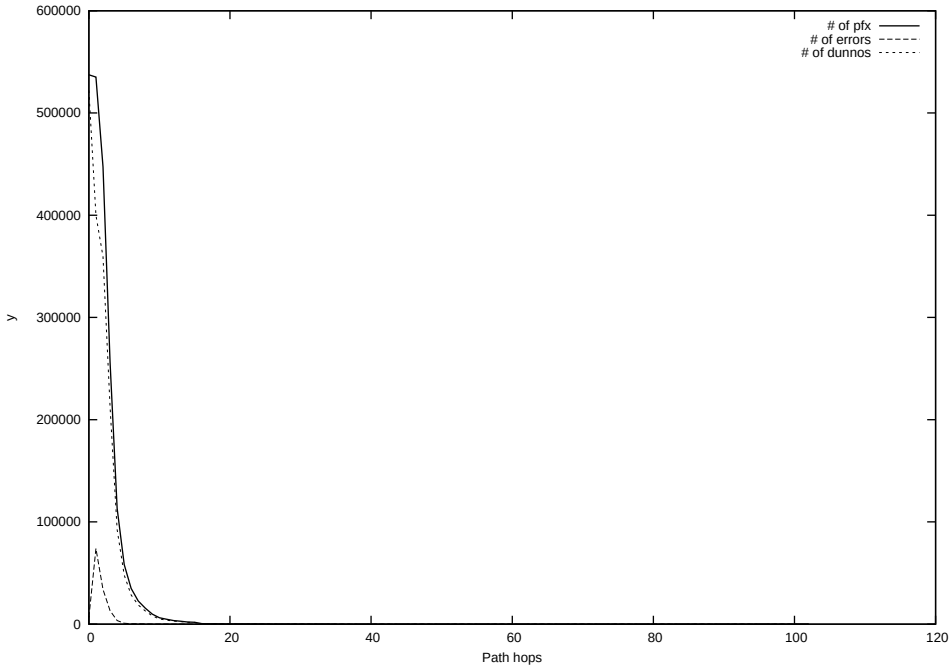
2015-05-02



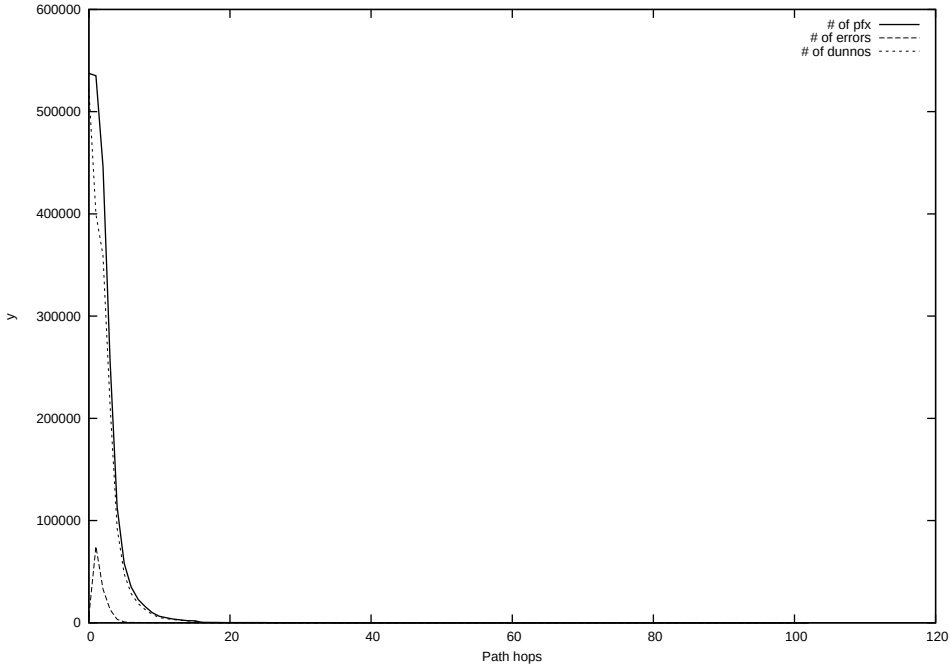
2015-05-03



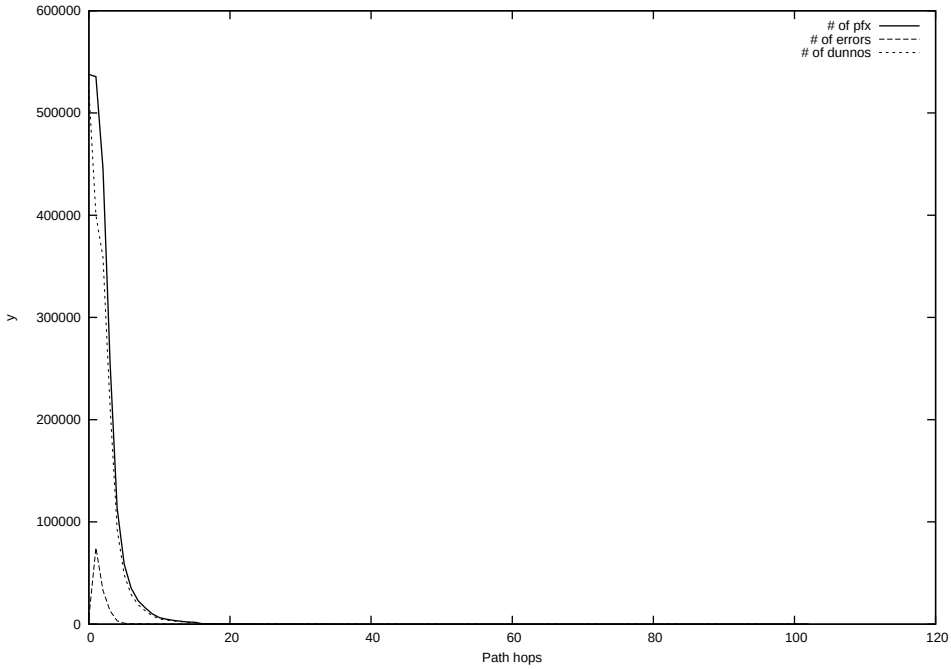
2015-05-04



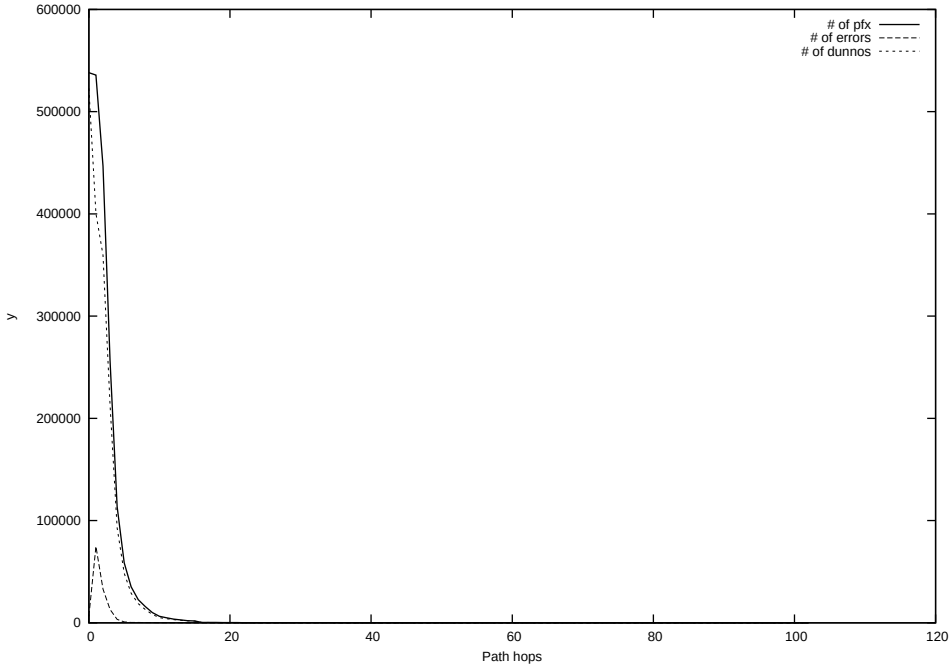
2015-05-05



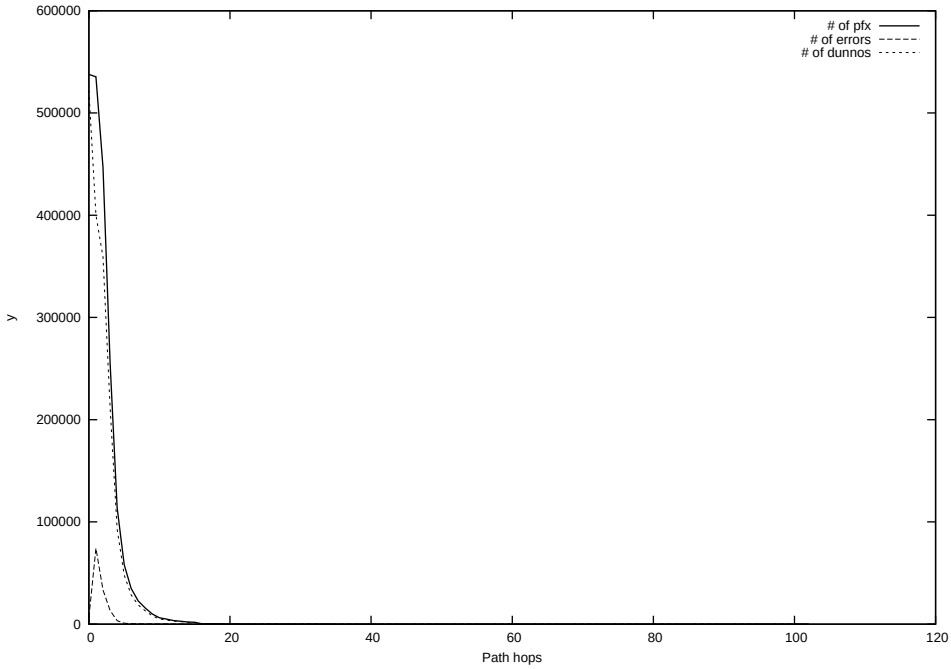
2015-05-06



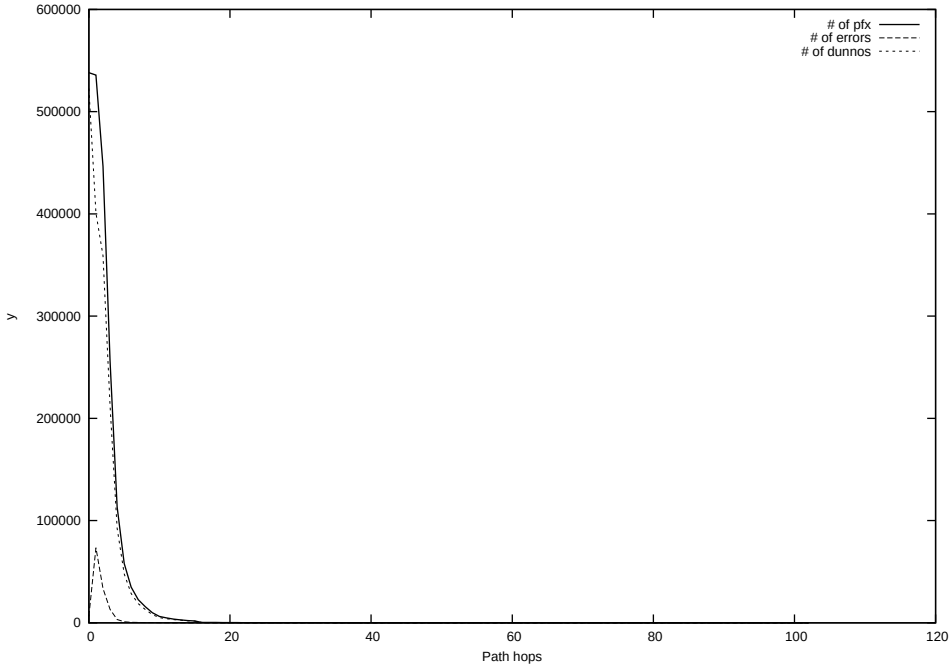
2015-05-07



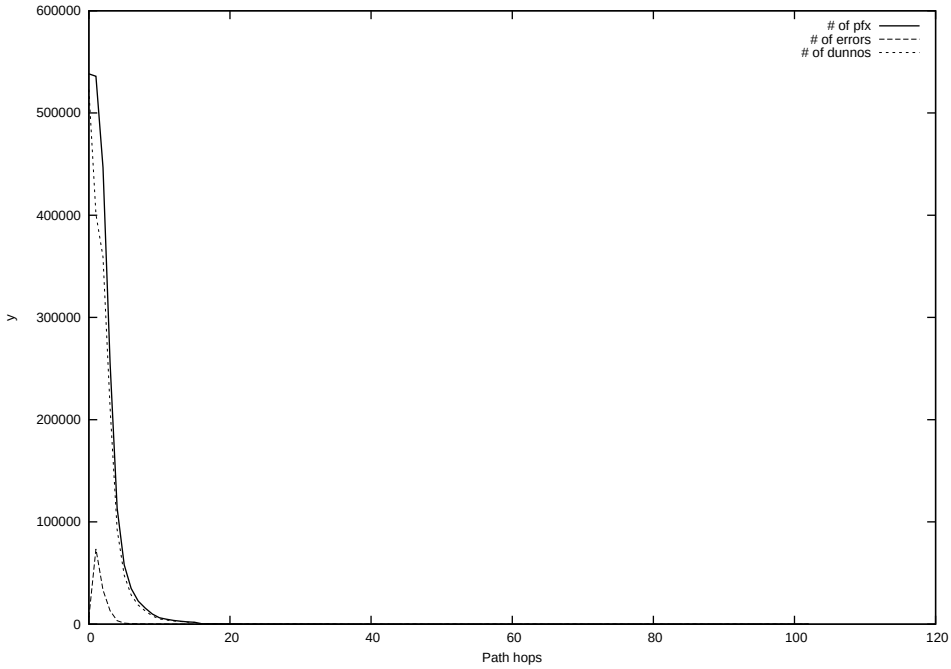
2015-05-08



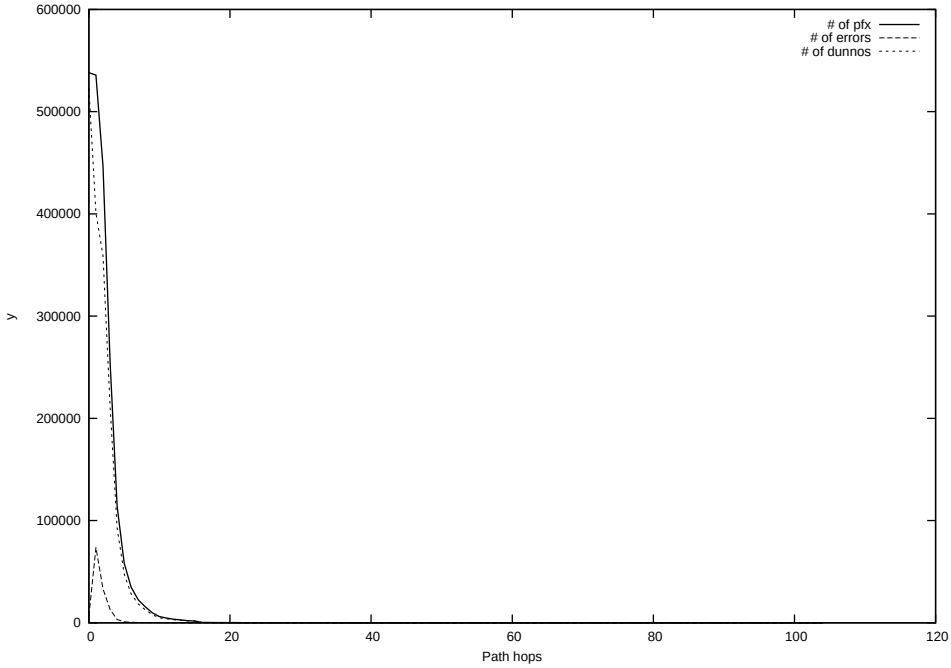
2015-05-09



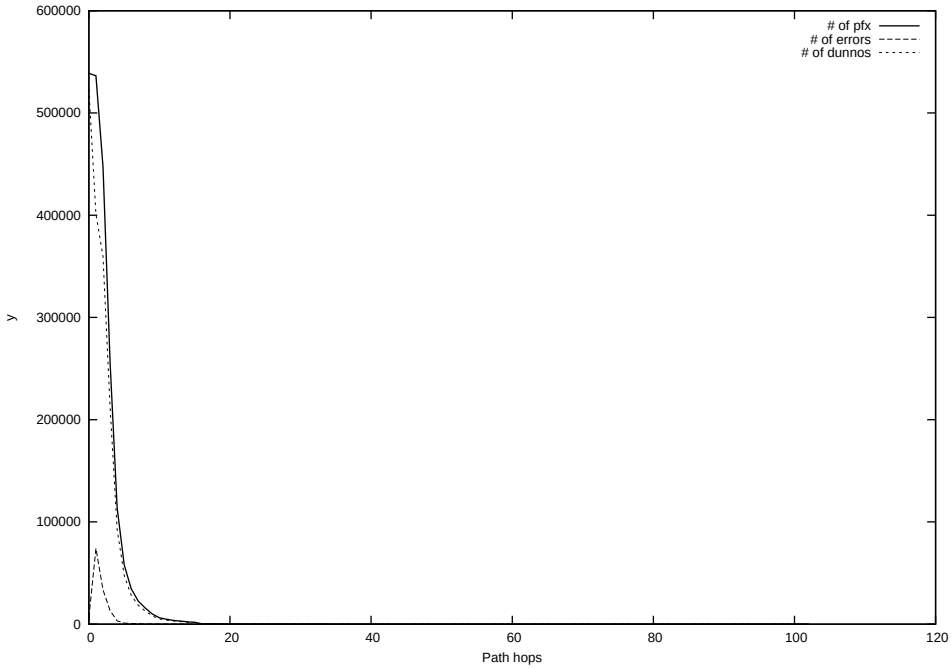
2015-05-10



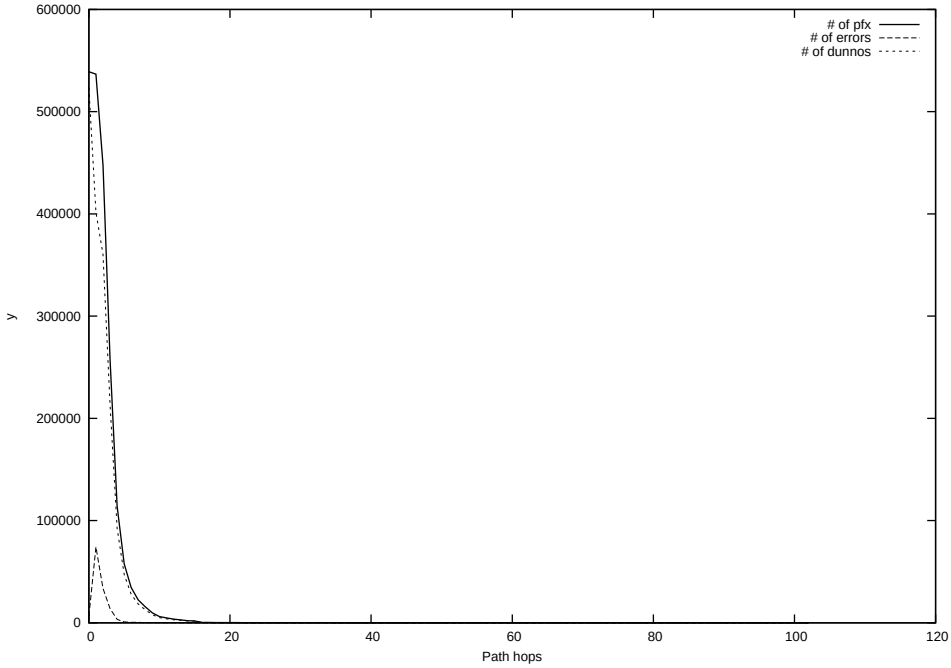
2015-05-11



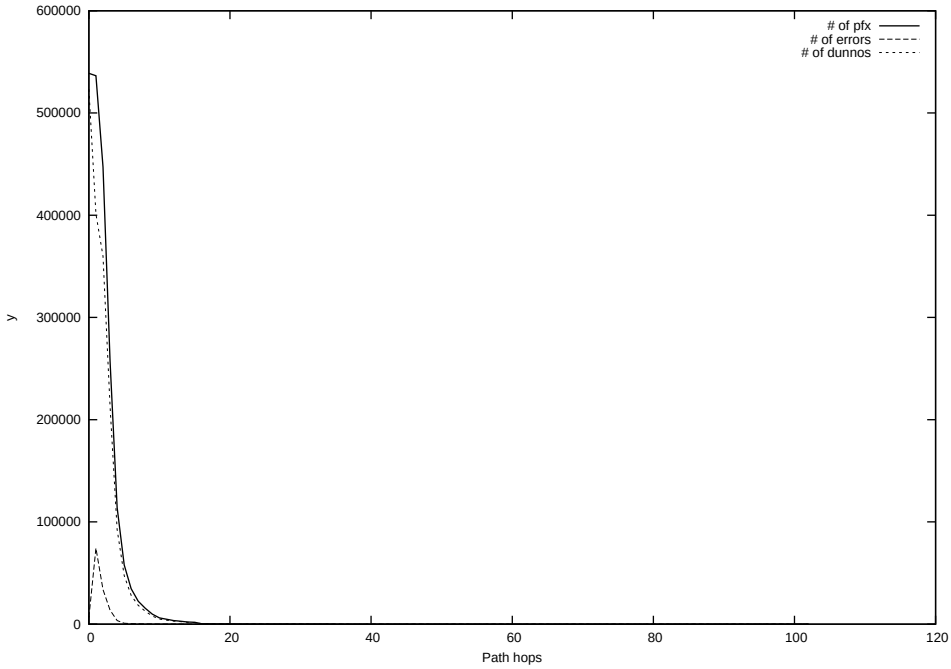
2015-05-12



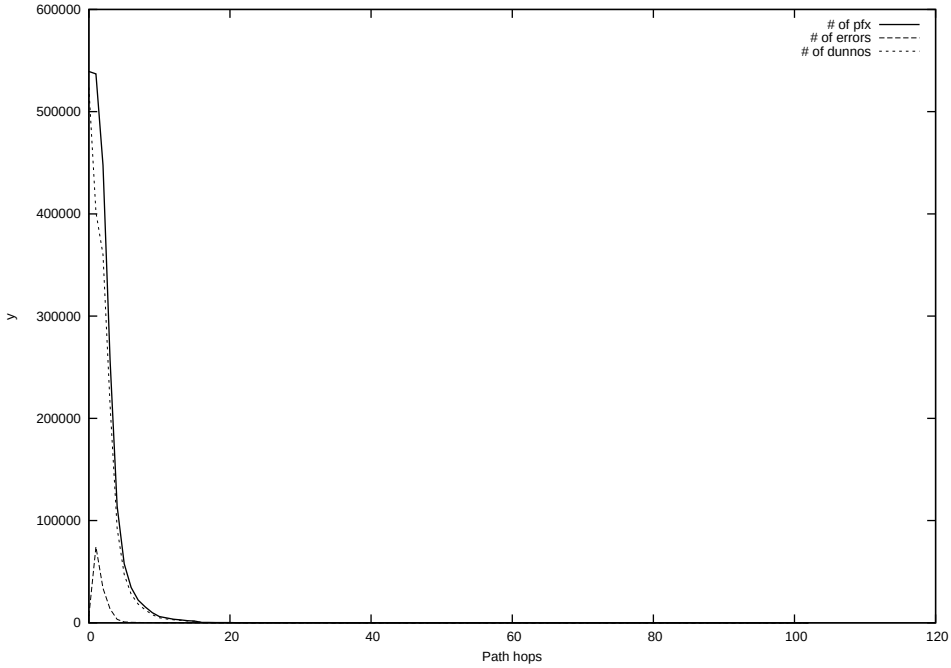
2015-05-13



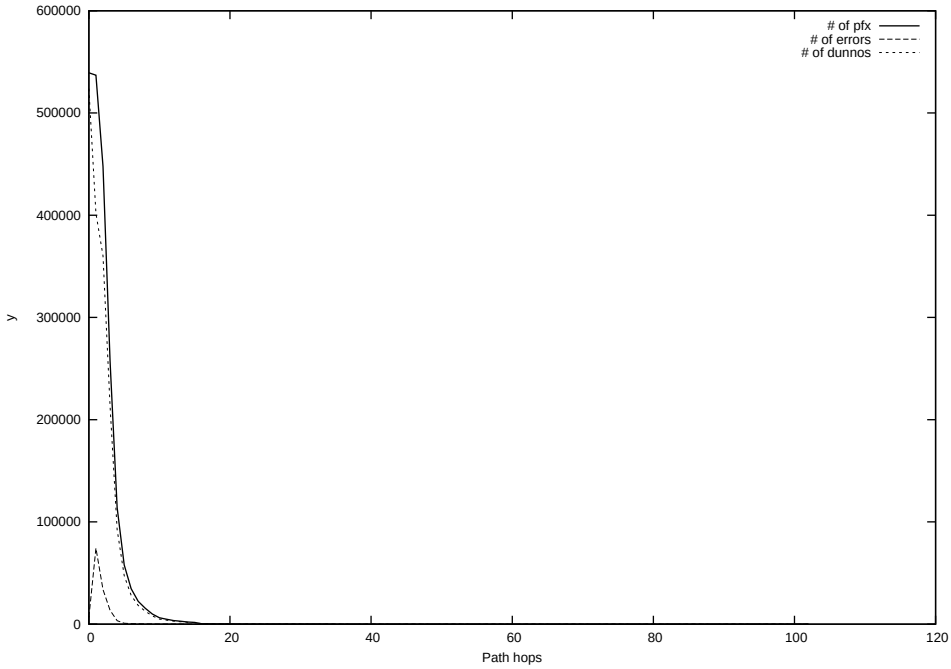
2015-05-14



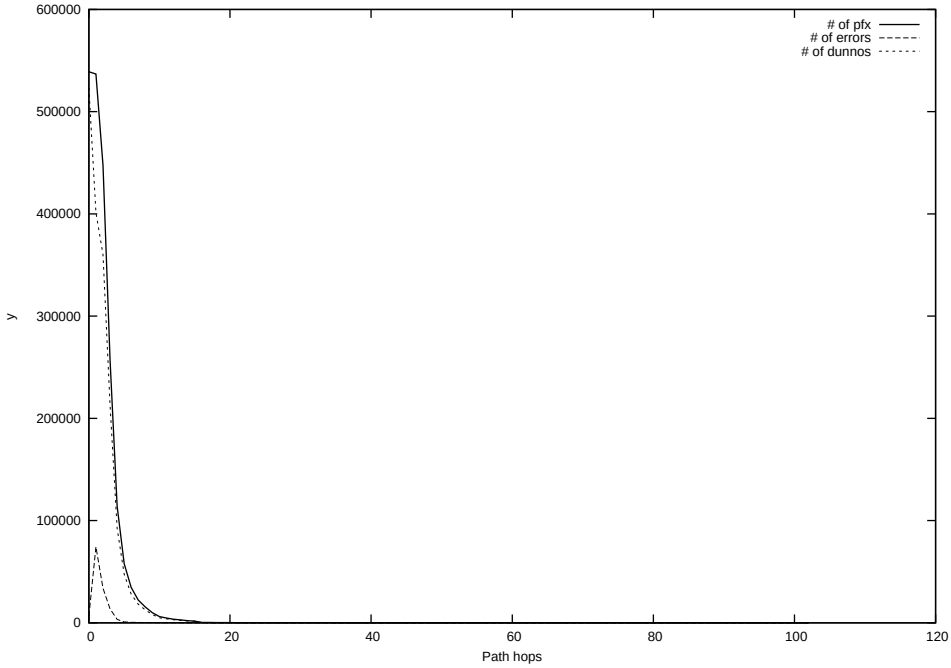
2015-05-15



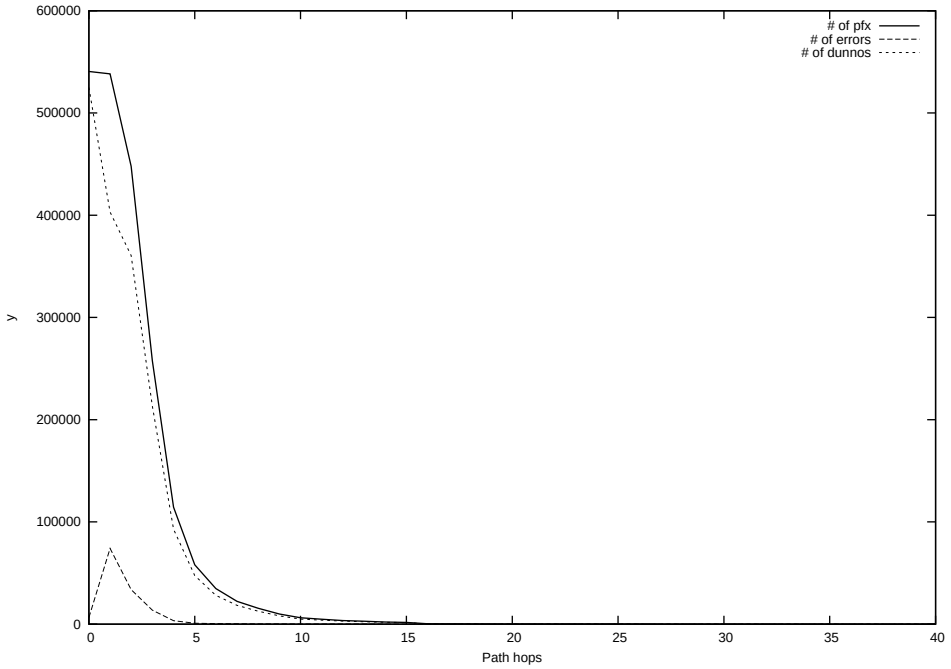
2015-05-16



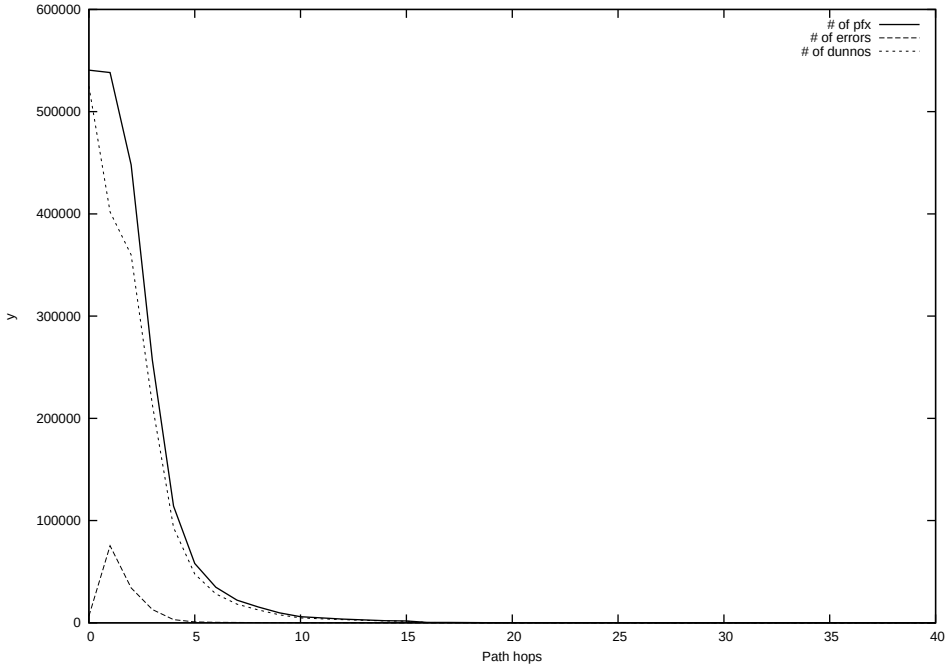
2015-05-17



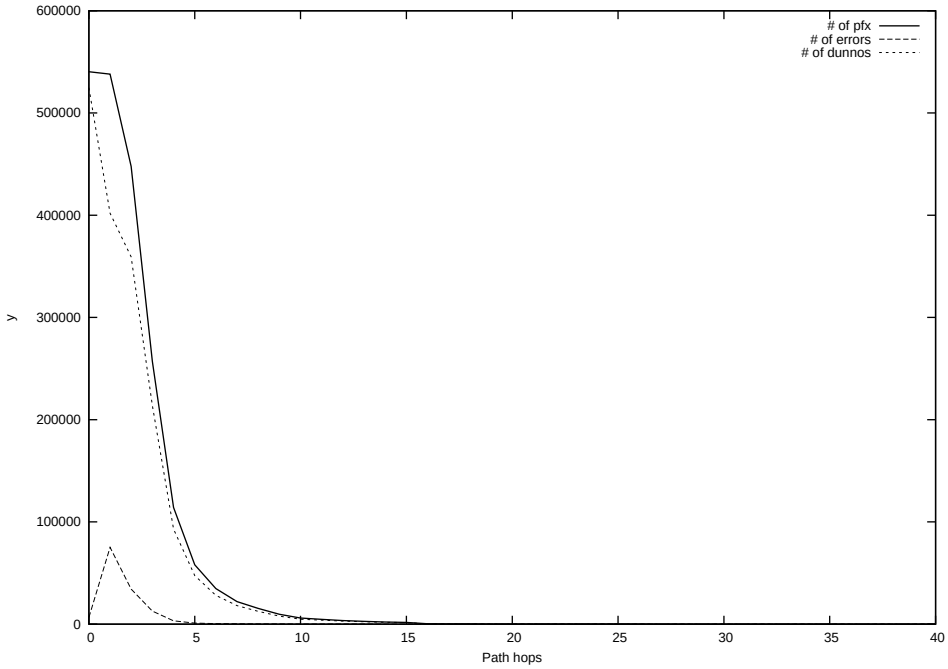
2015-05-18



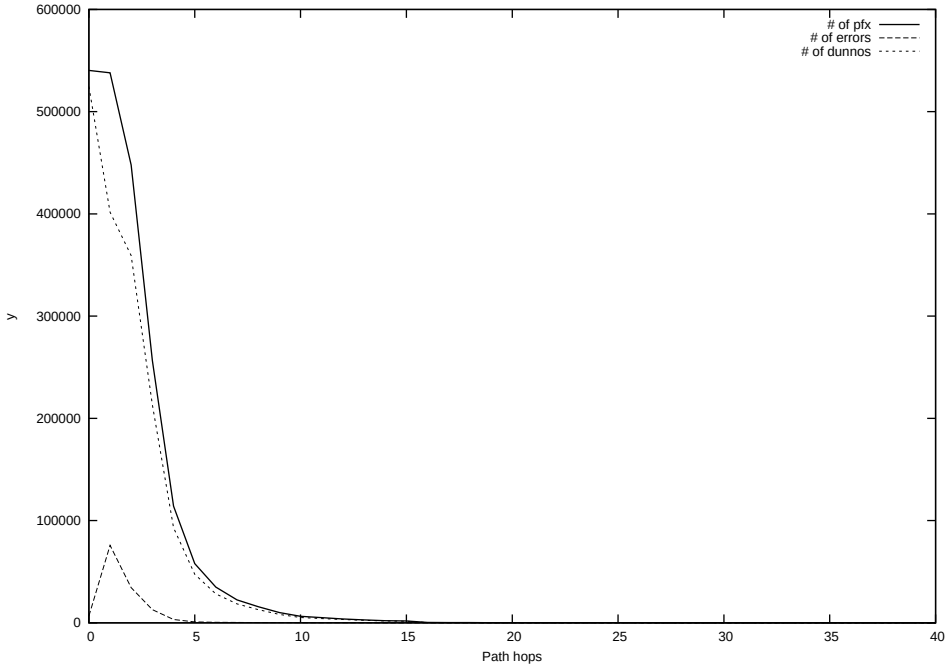
2015-05-19



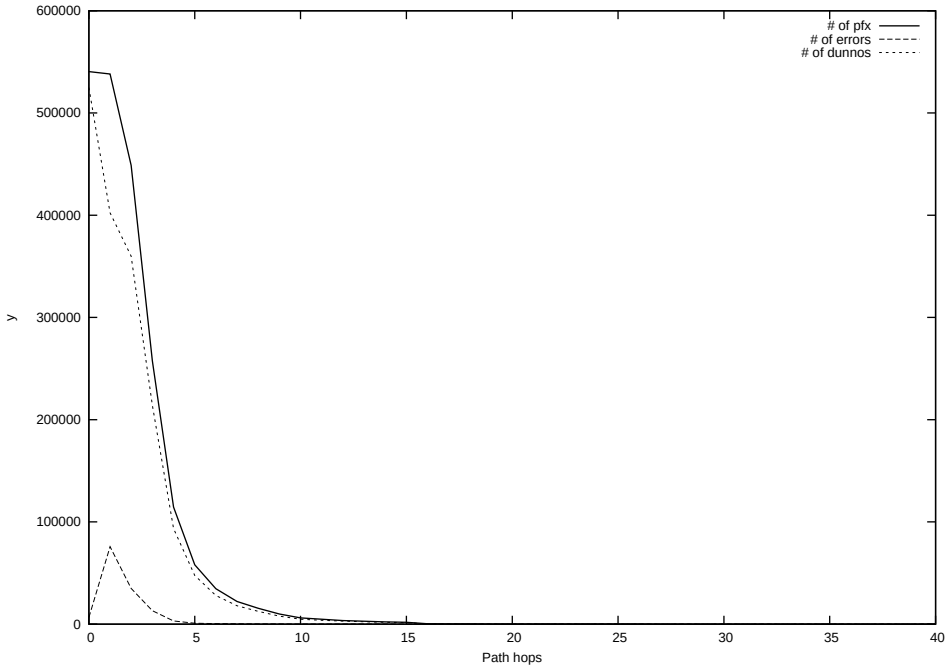
2015-05-20



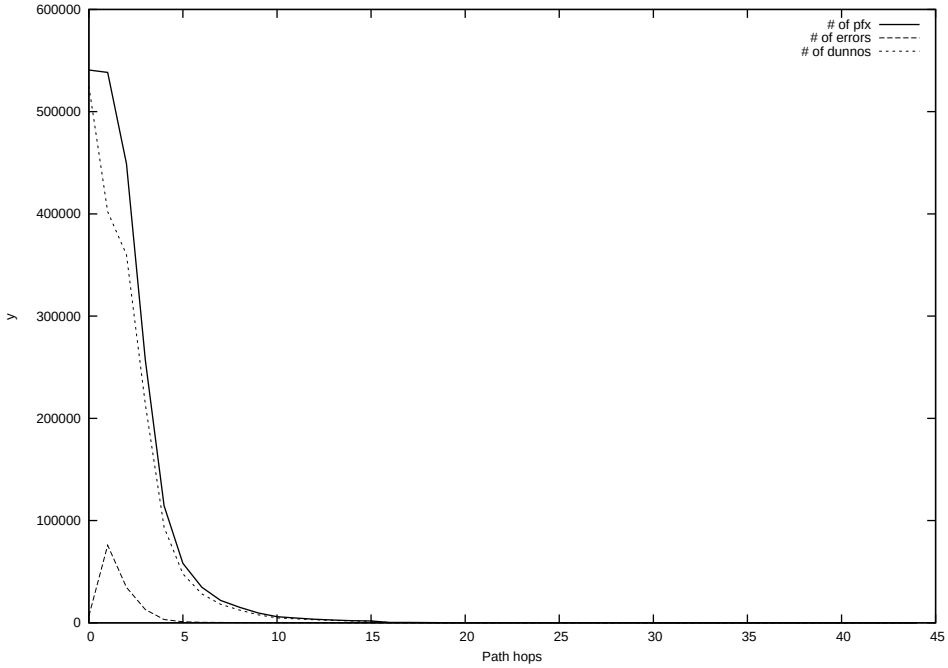
2015-05-21



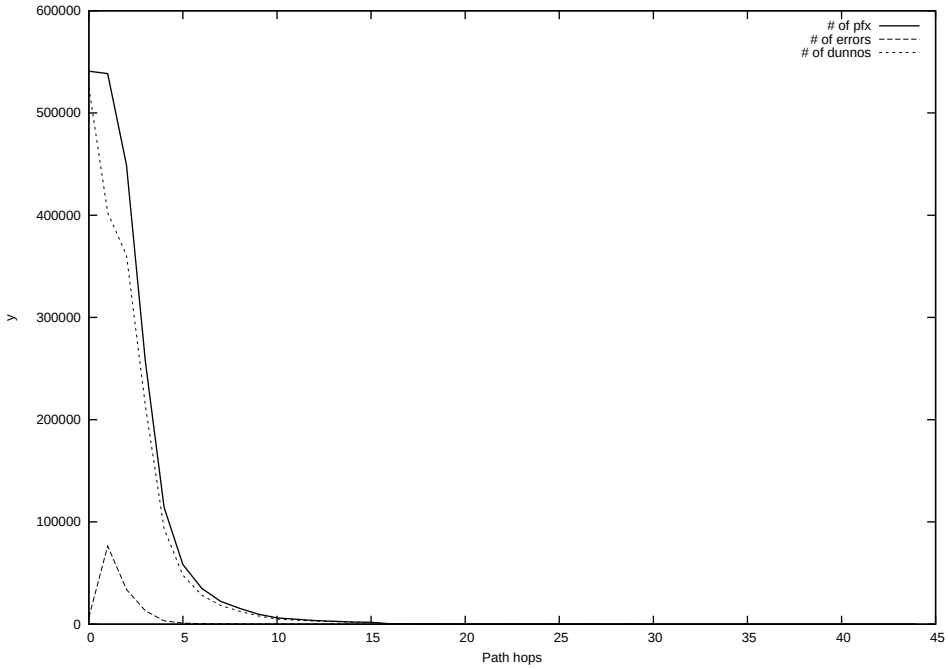
2015-05-22



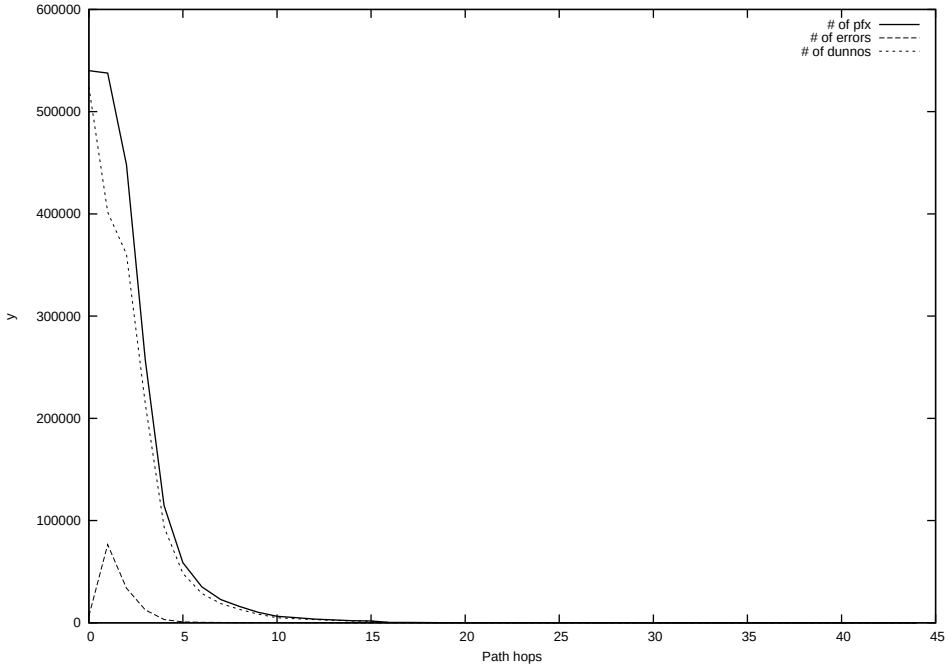
2015-05-23



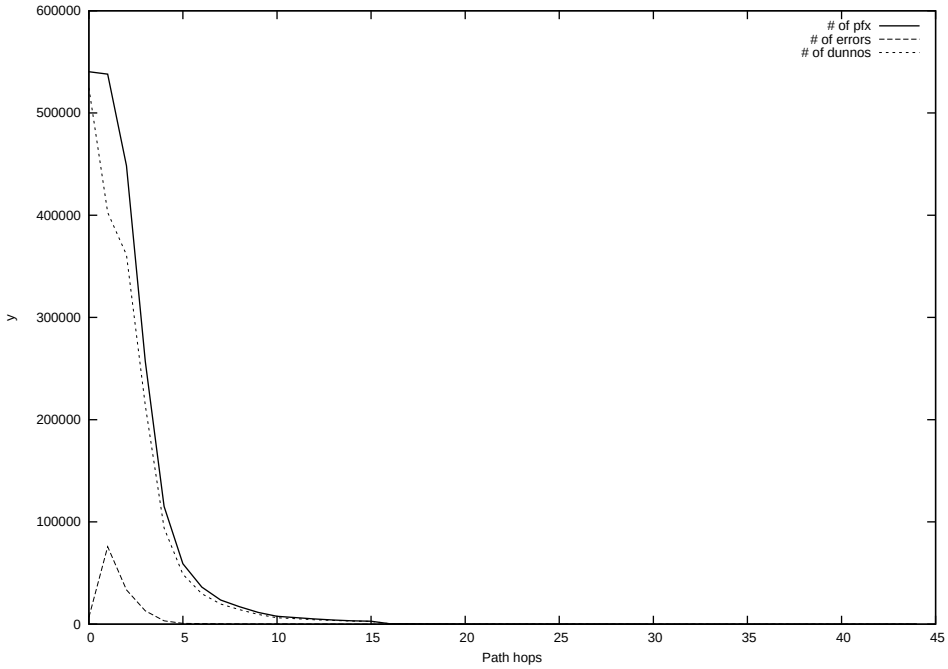
2015-05-24



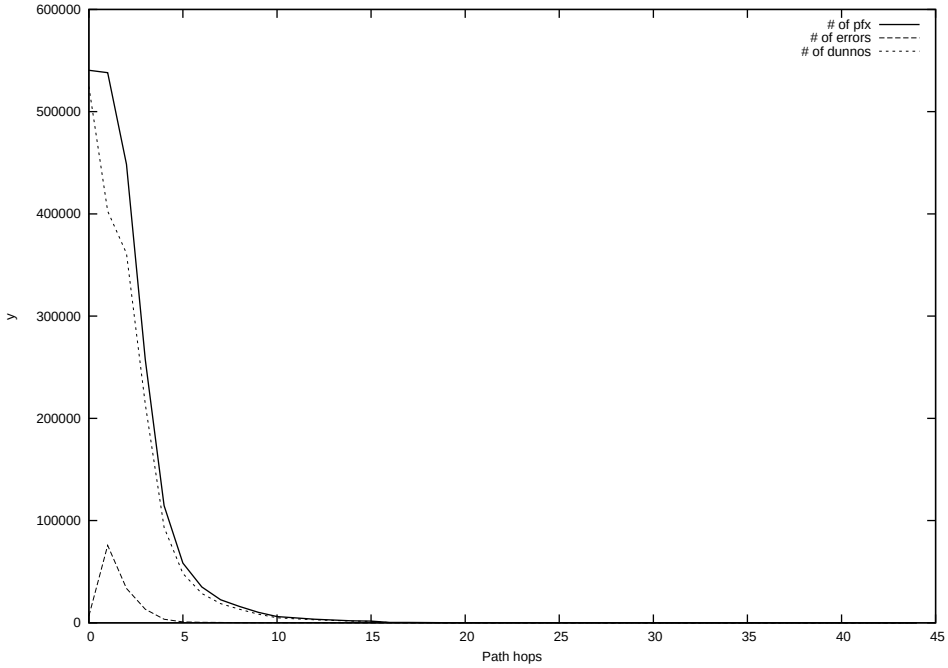
2015-05-25



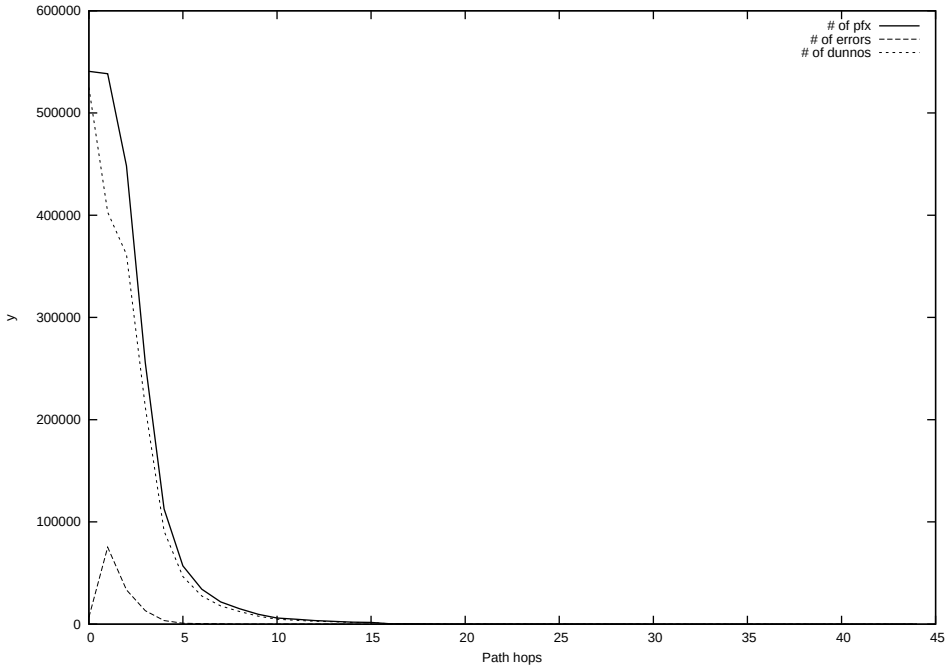
2015-05-26



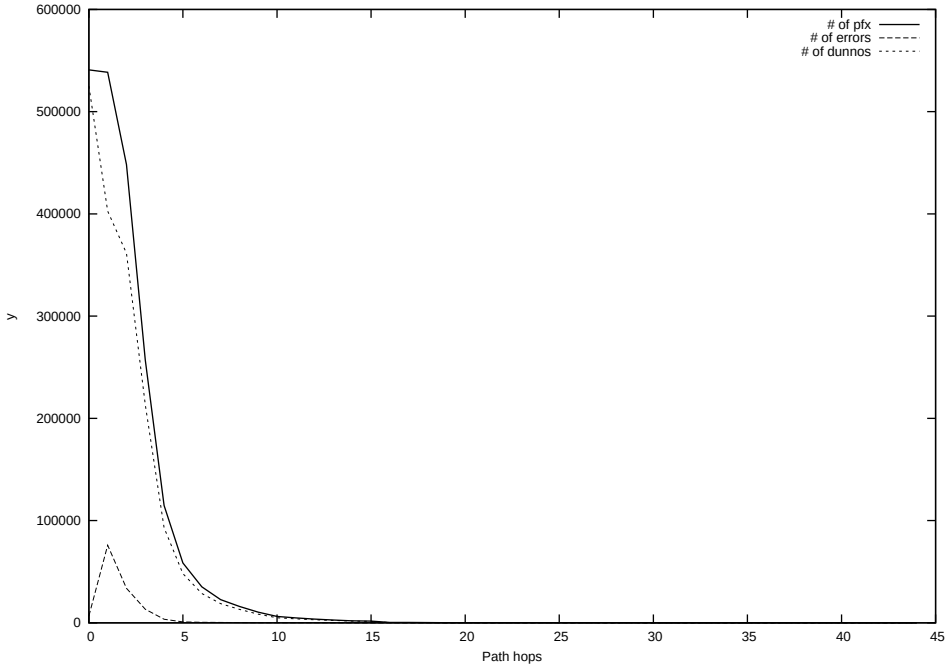
2015-05-27



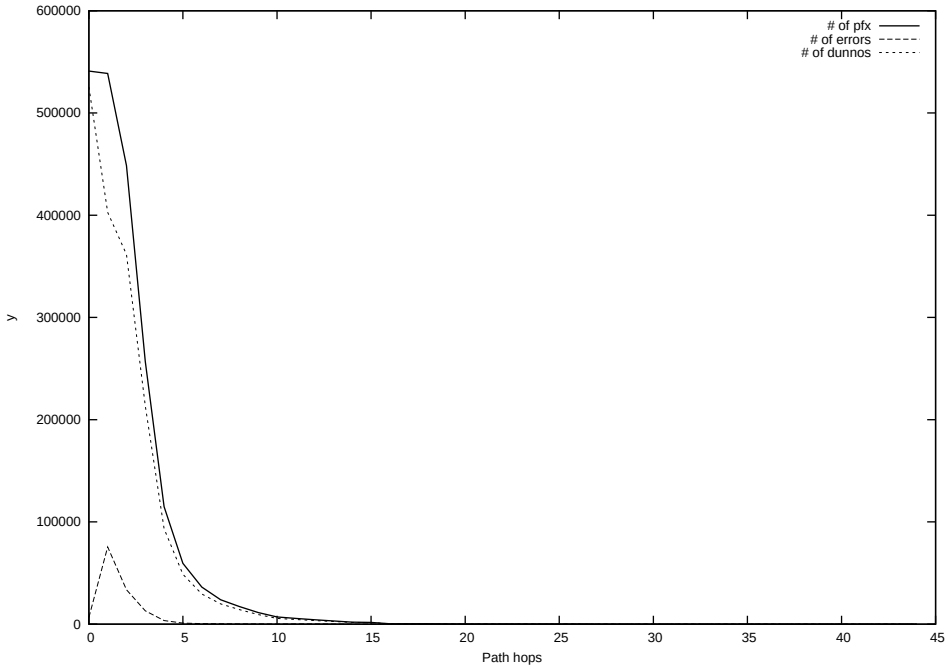
2015-05-28



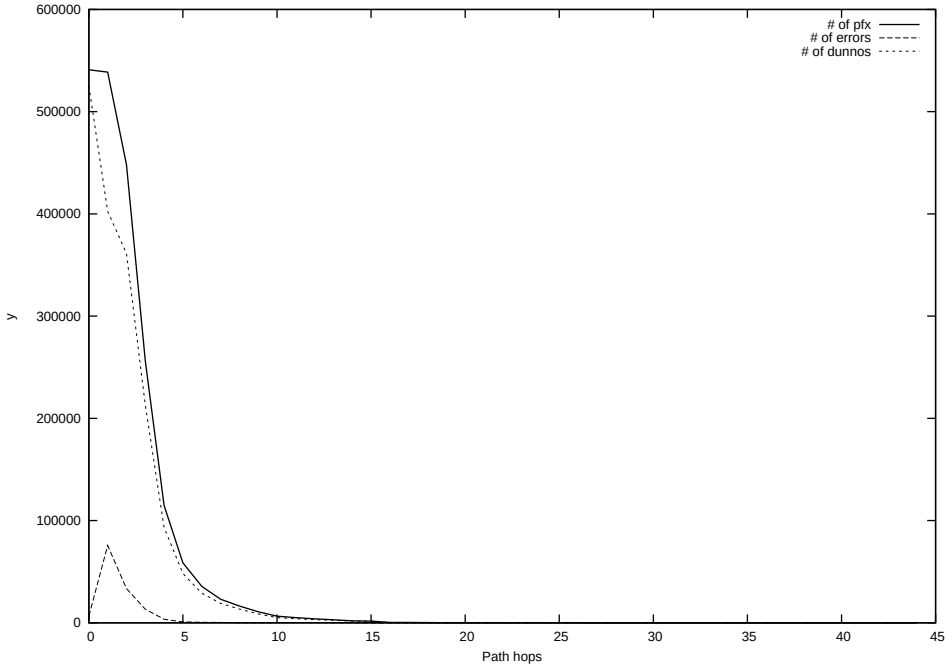
2015-05-29



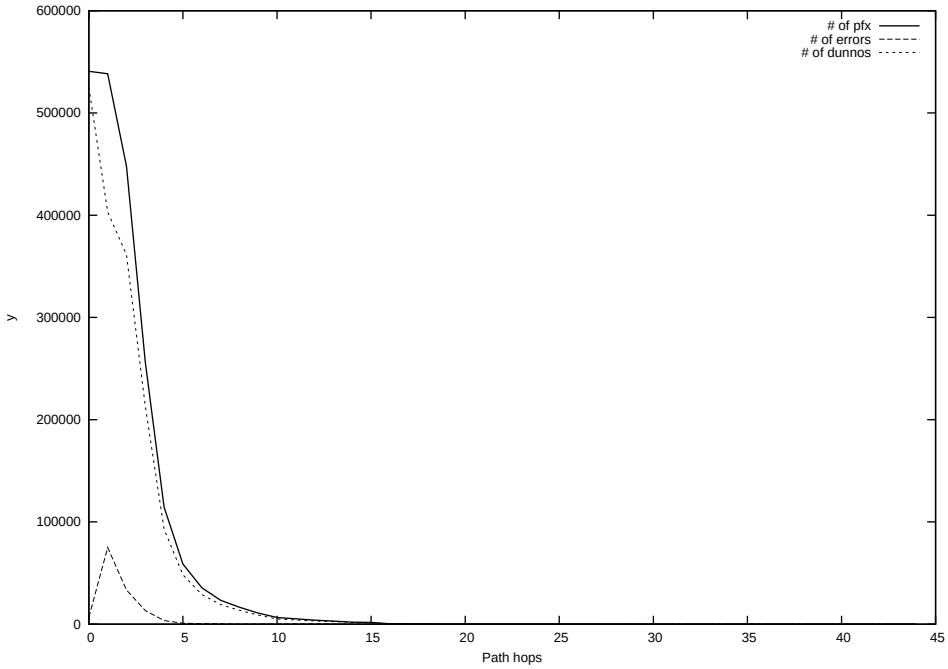
2015-05-30



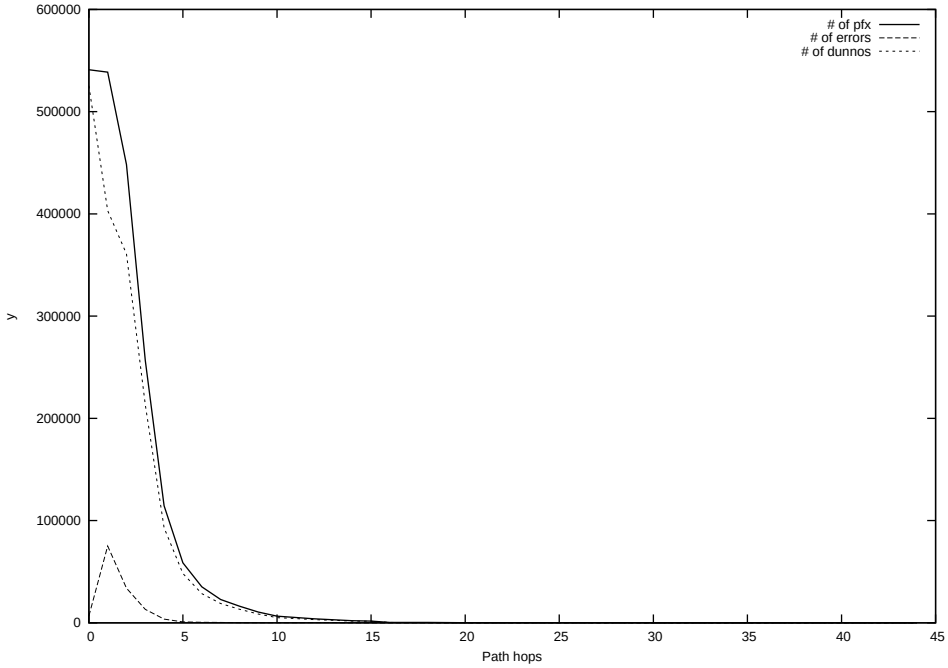
2015-05-31



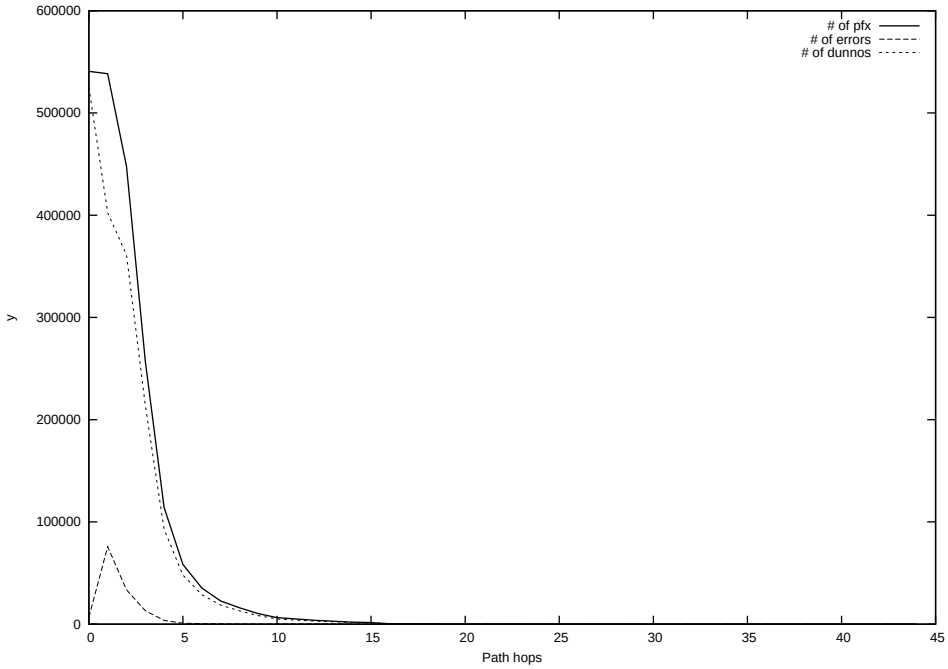
2015-06-01



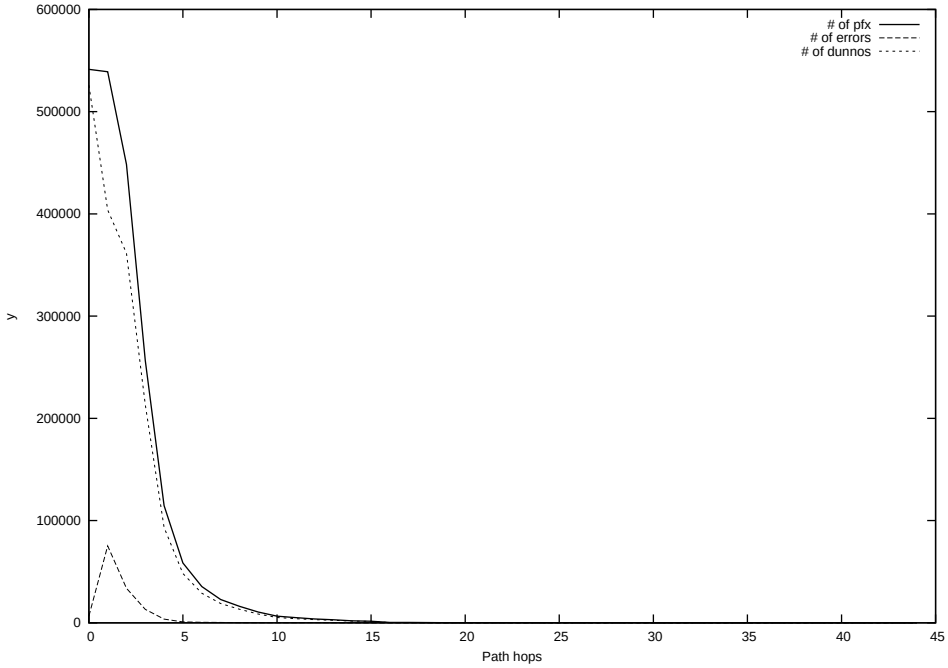
2015-06-02



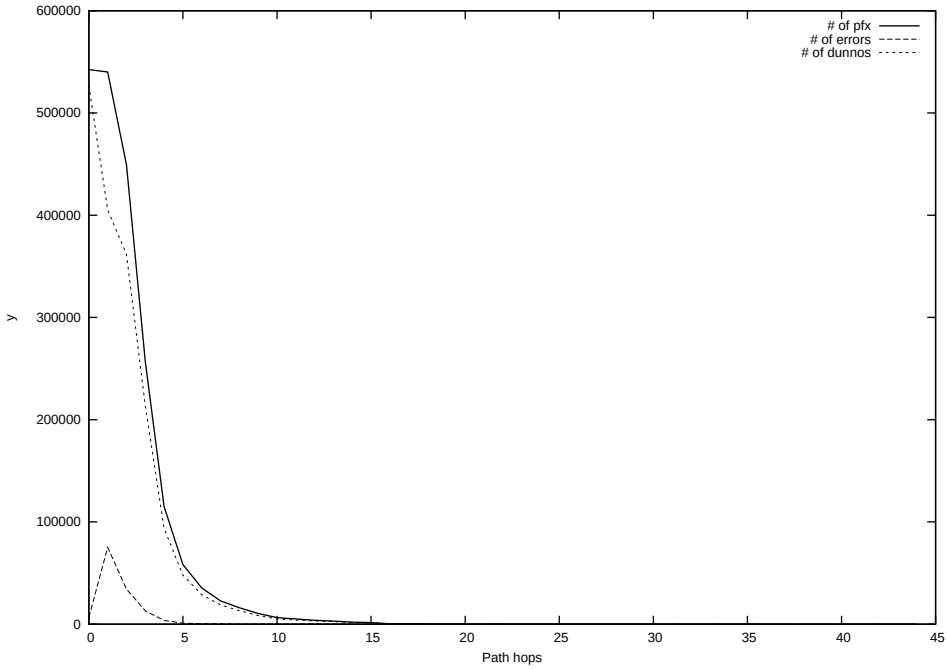
2015-06-03



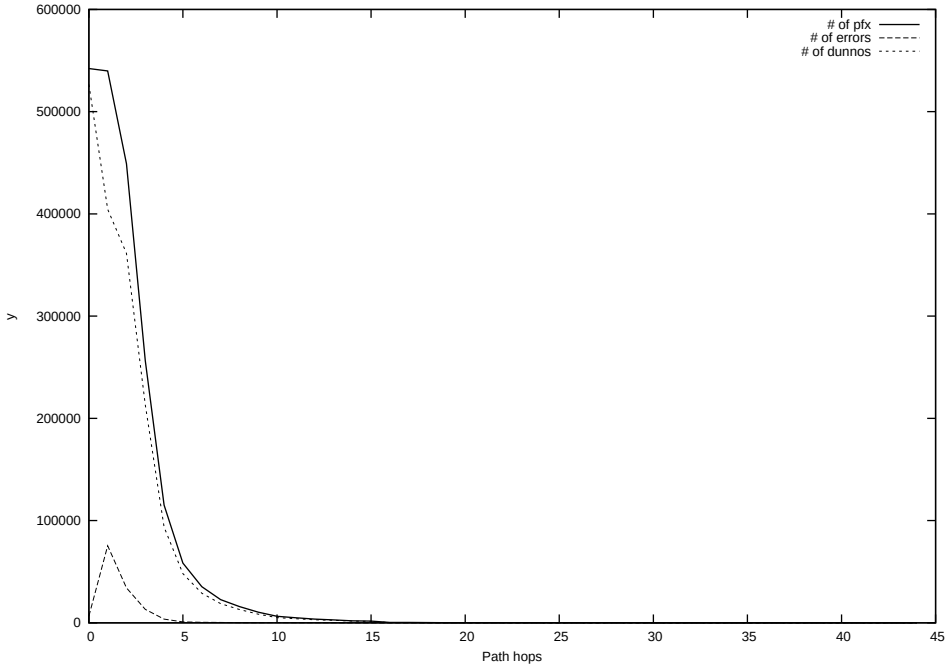
2015-06-04



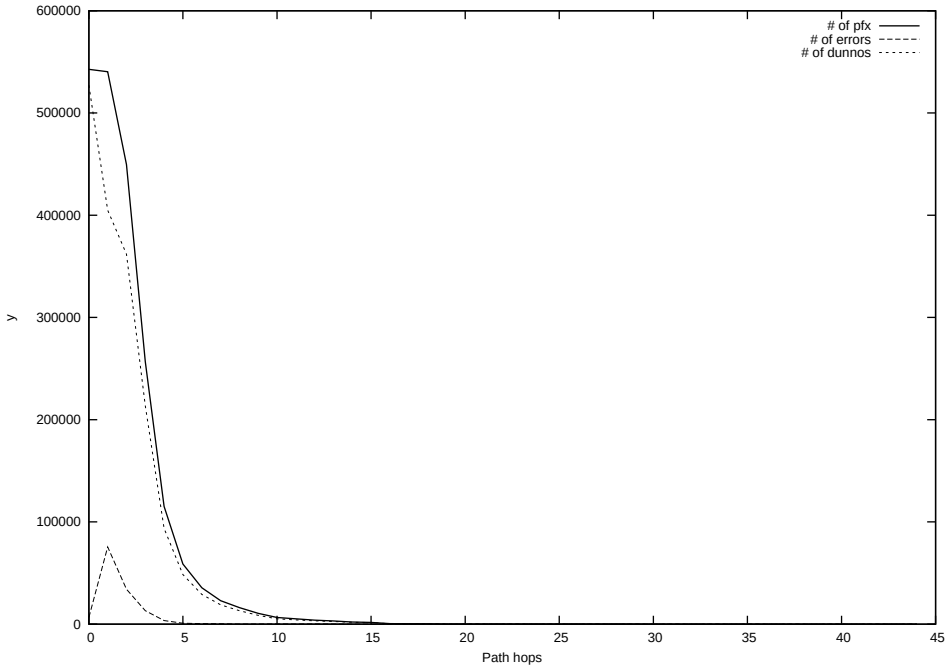
2015-06-05



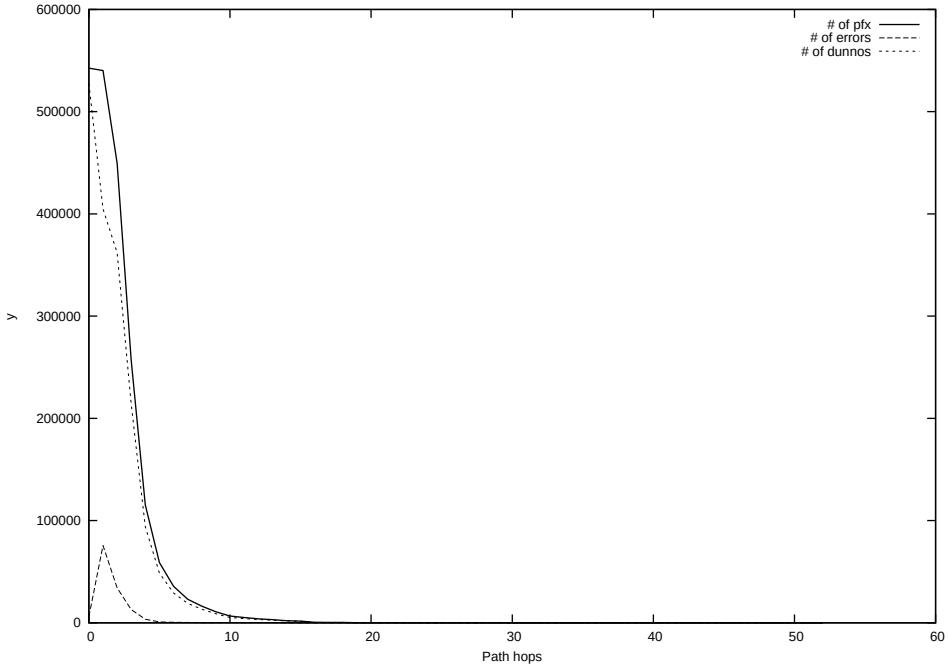
2015-06-06



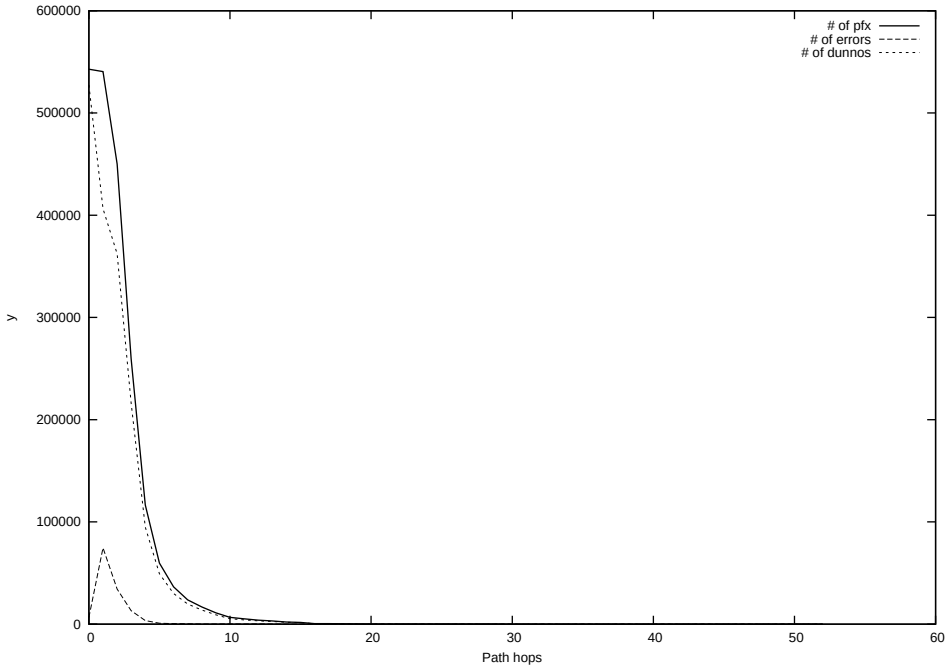
2015-06-07



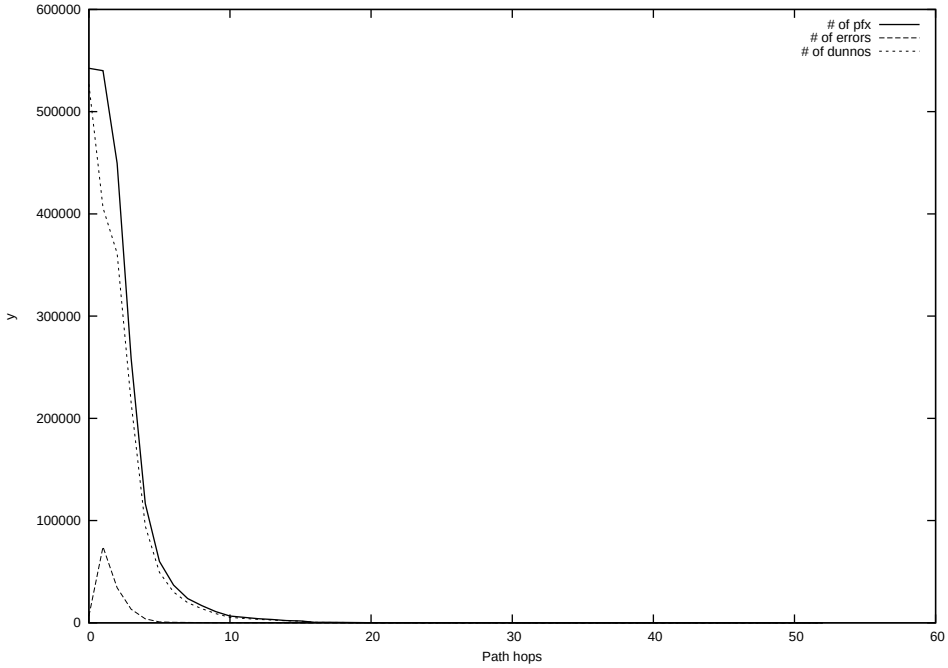
2015-06-08



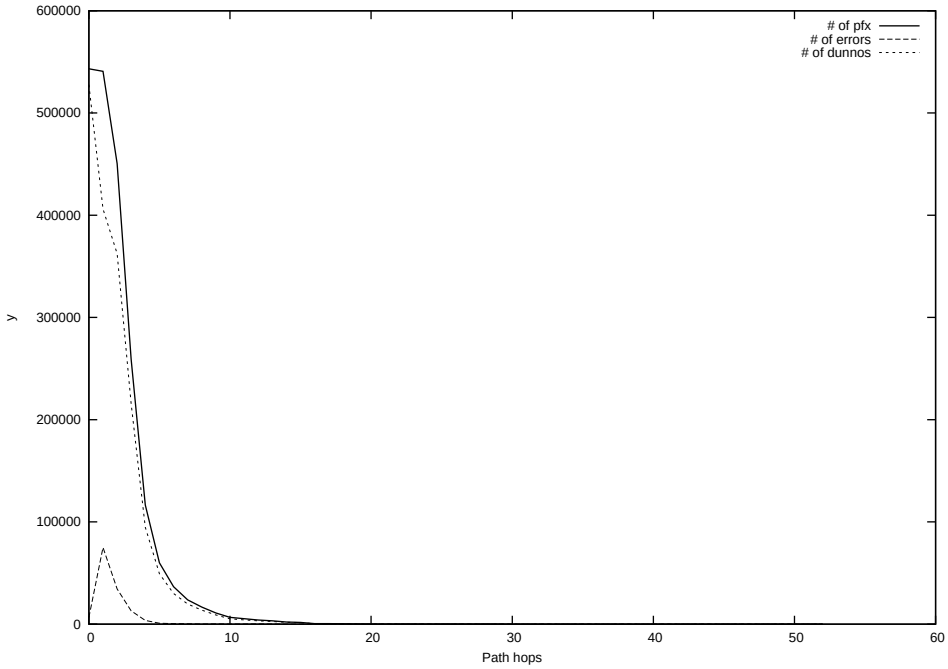
2015-06-09



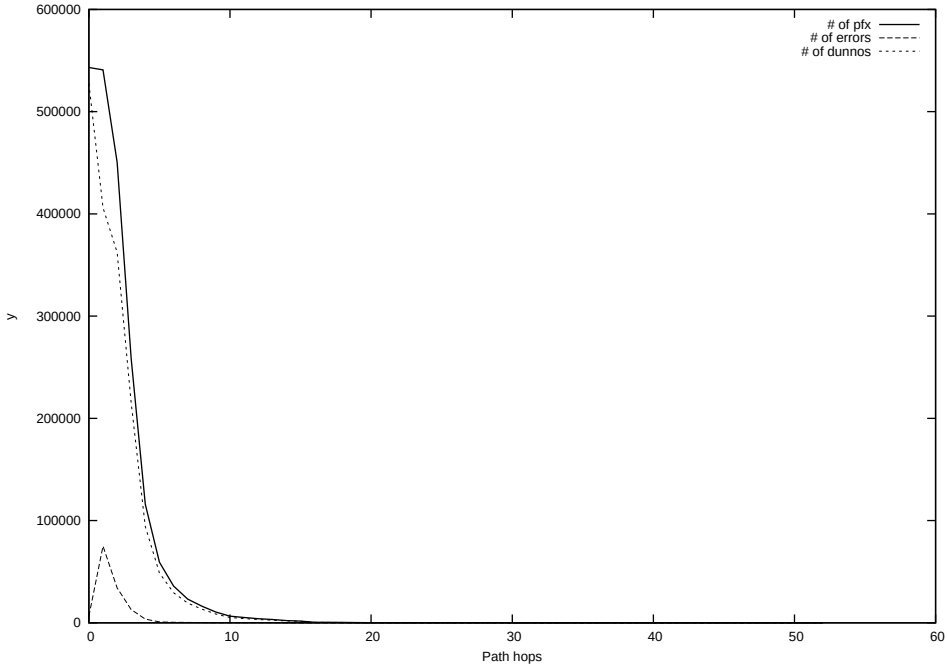
2015-06-10



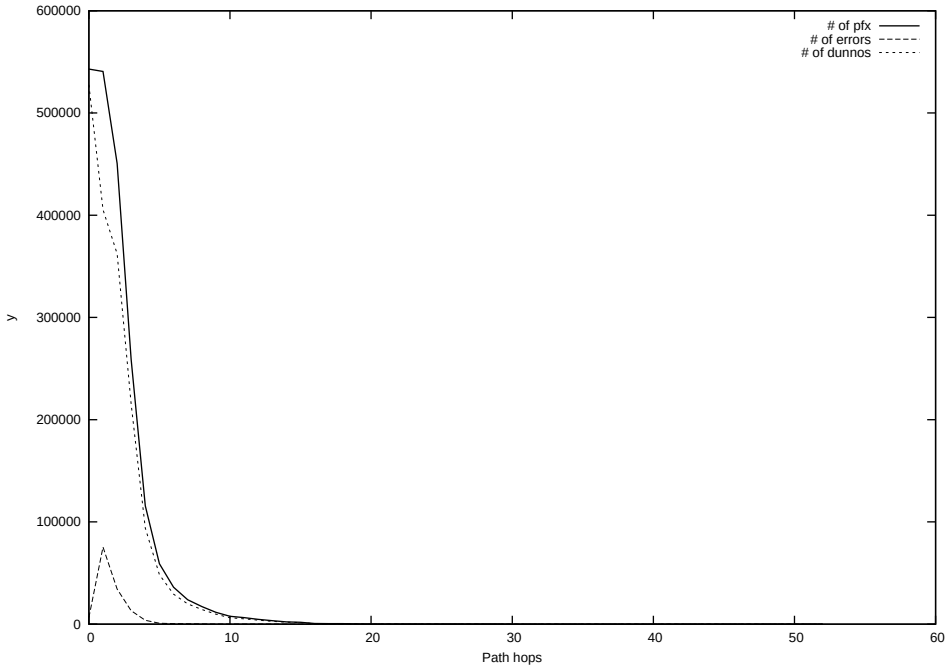
2015-06-11



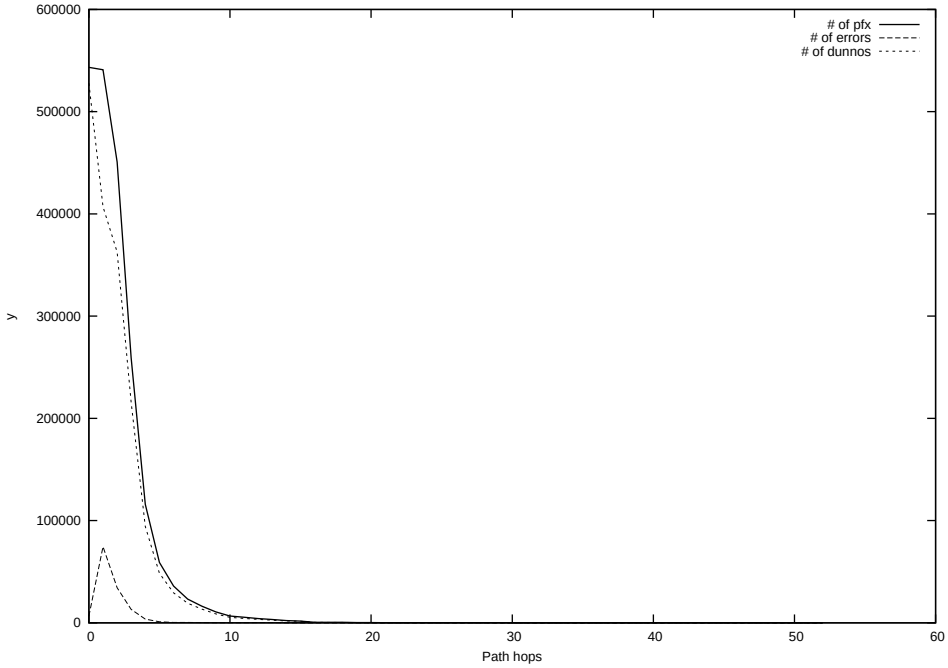
2015-06-12



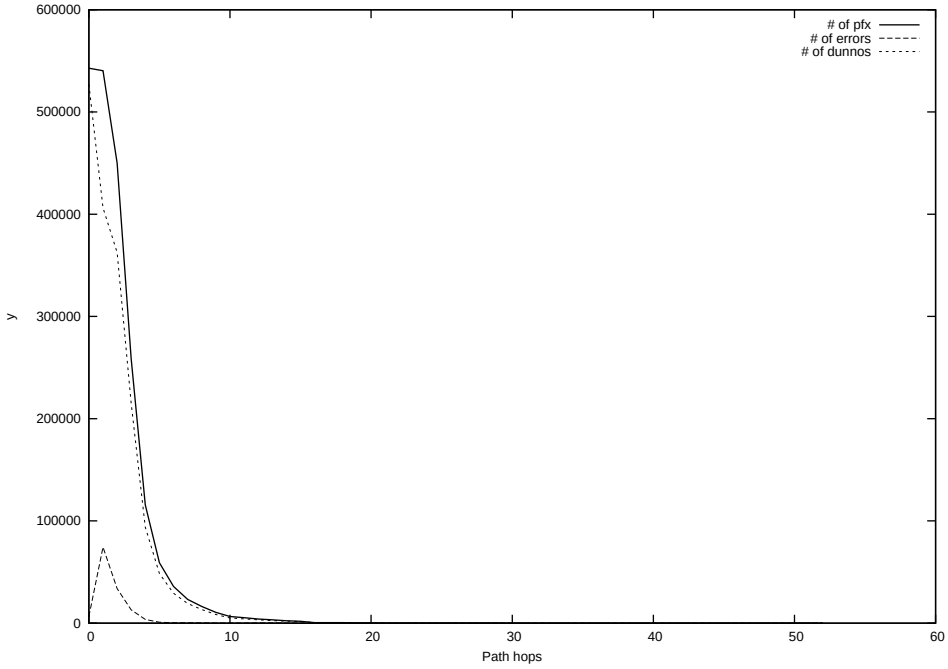
2015-06-13



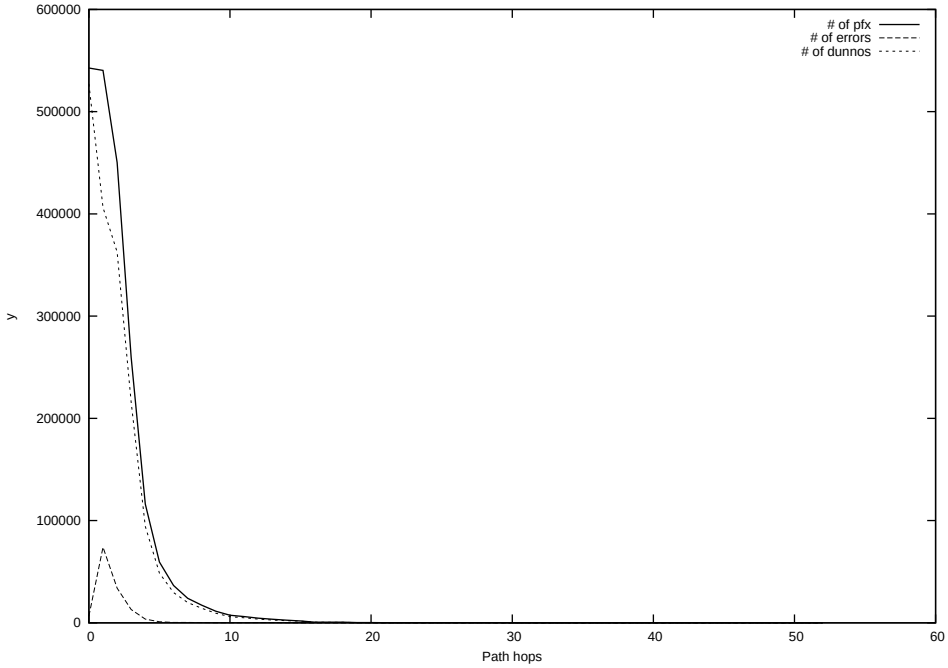
2015-06-14



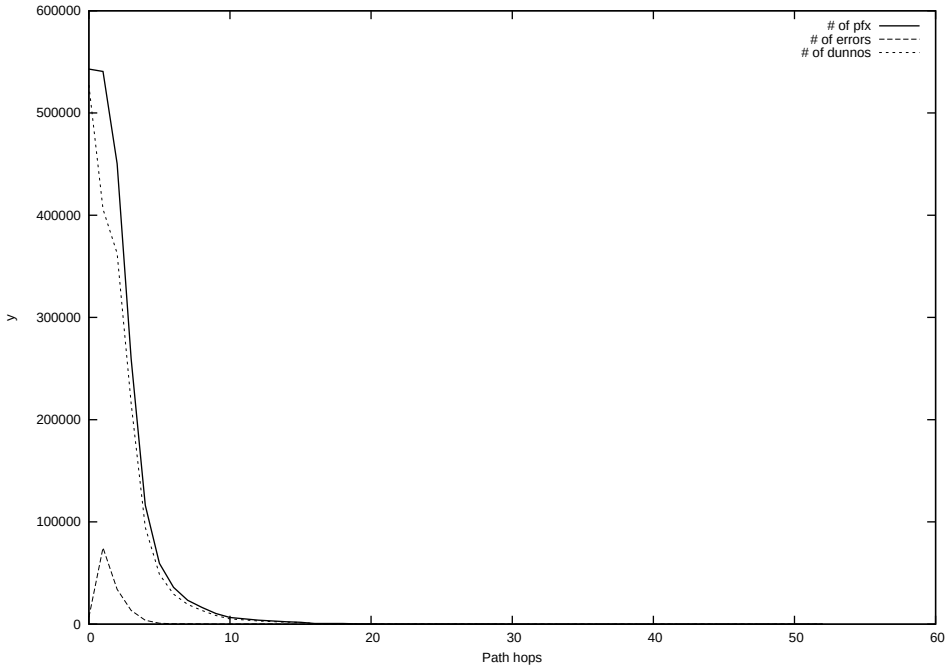
2015-06-15



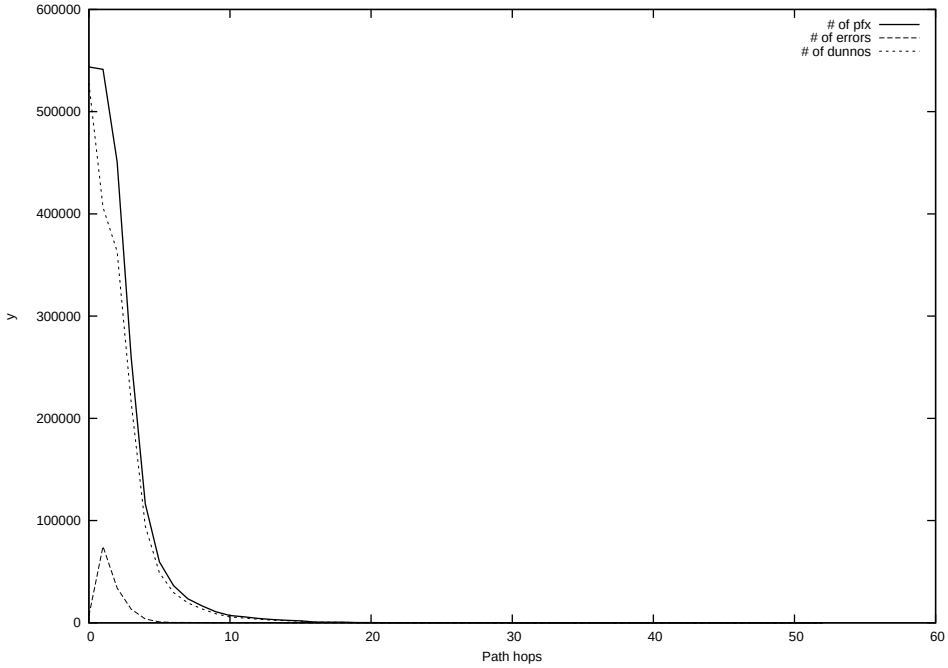
2015-06-16



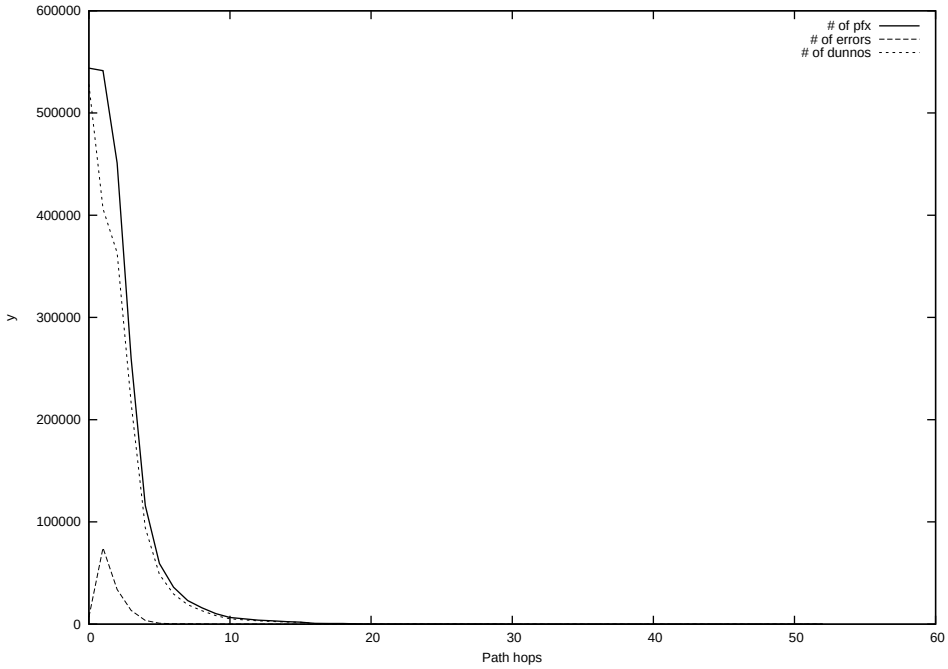
2015-06-17



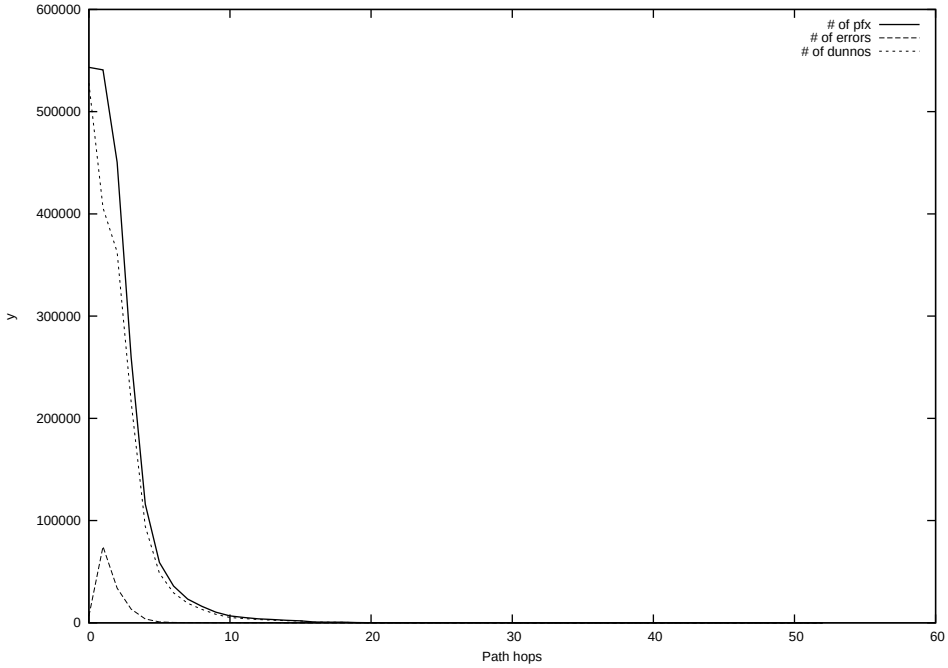
2015-06-18



2015-06-19

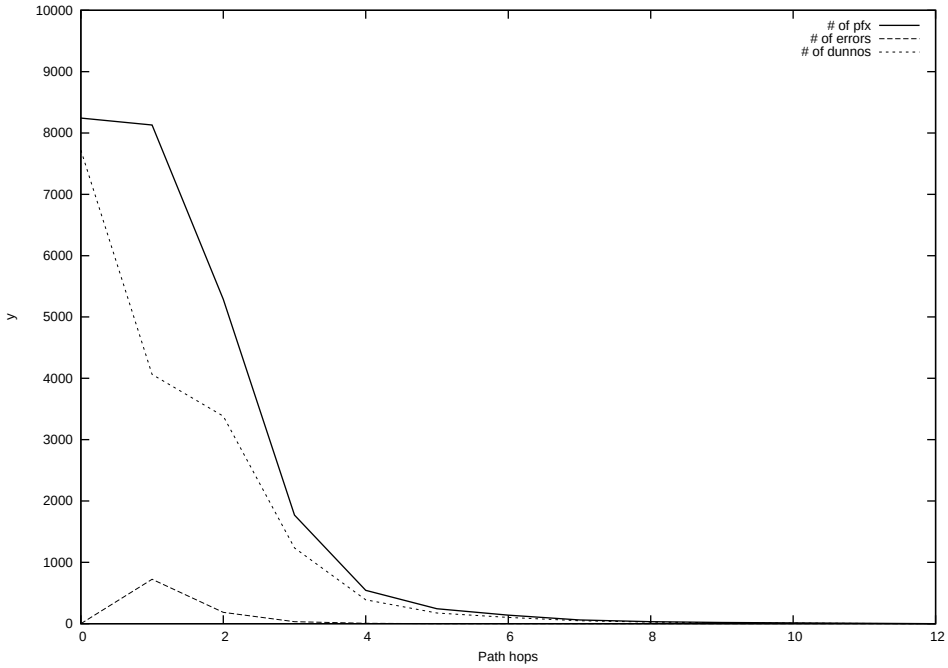


2015-06-20

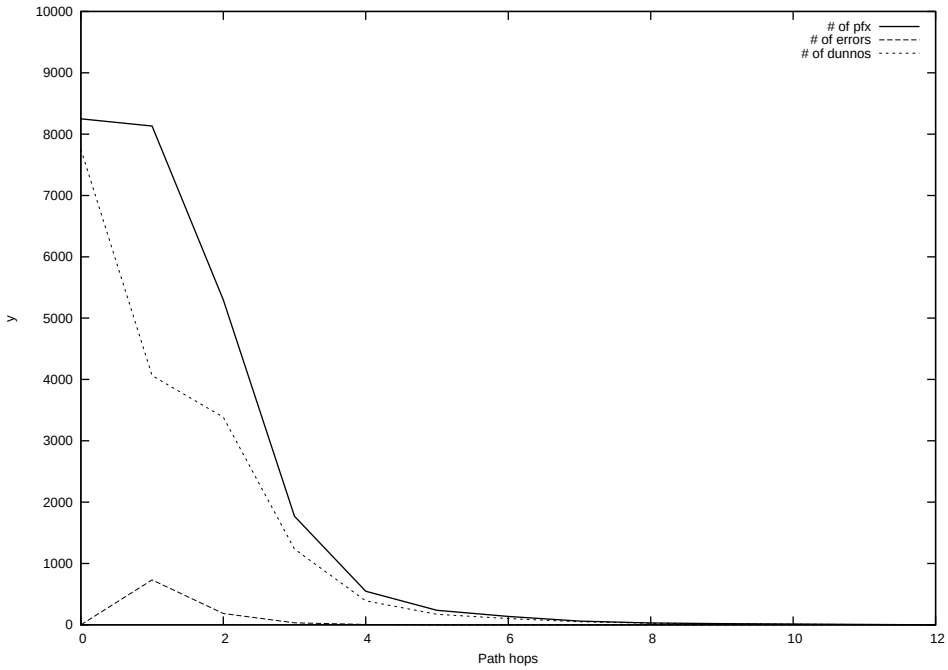


2015-06-21

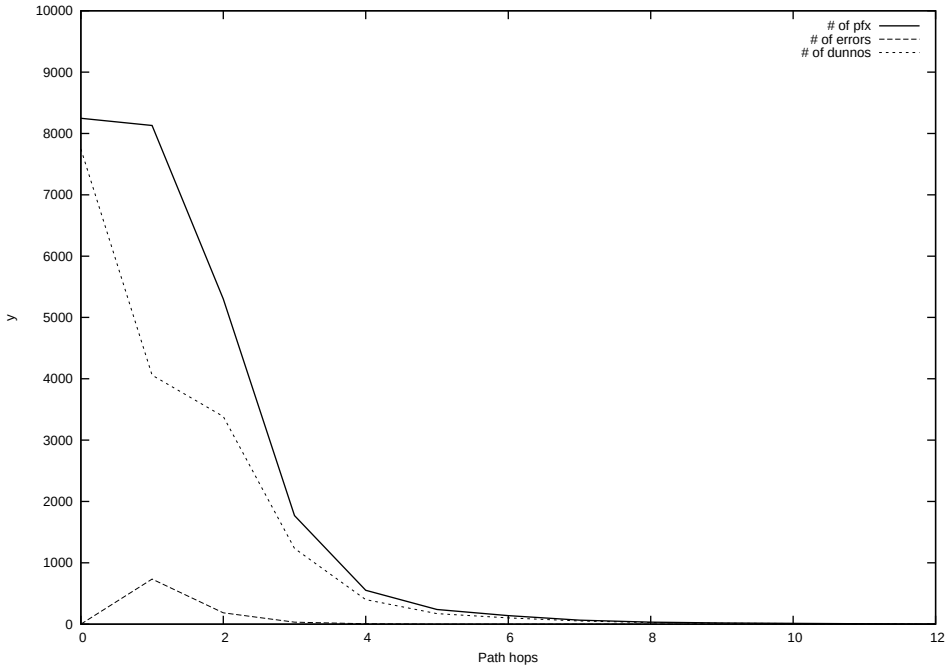
A.12 IPv6 BGP full paths matched against RIPE DB



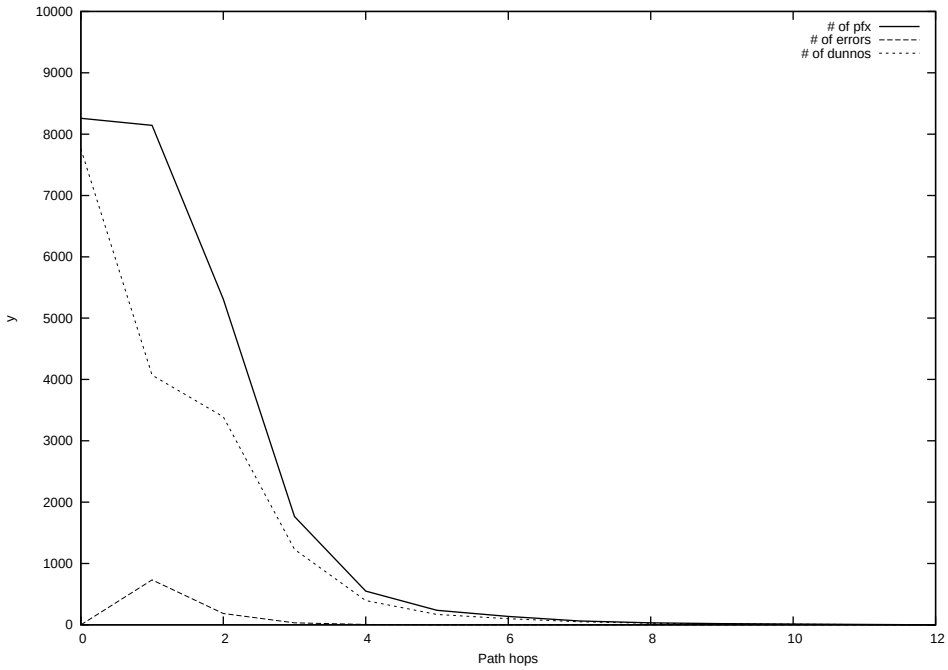
2012-03-22



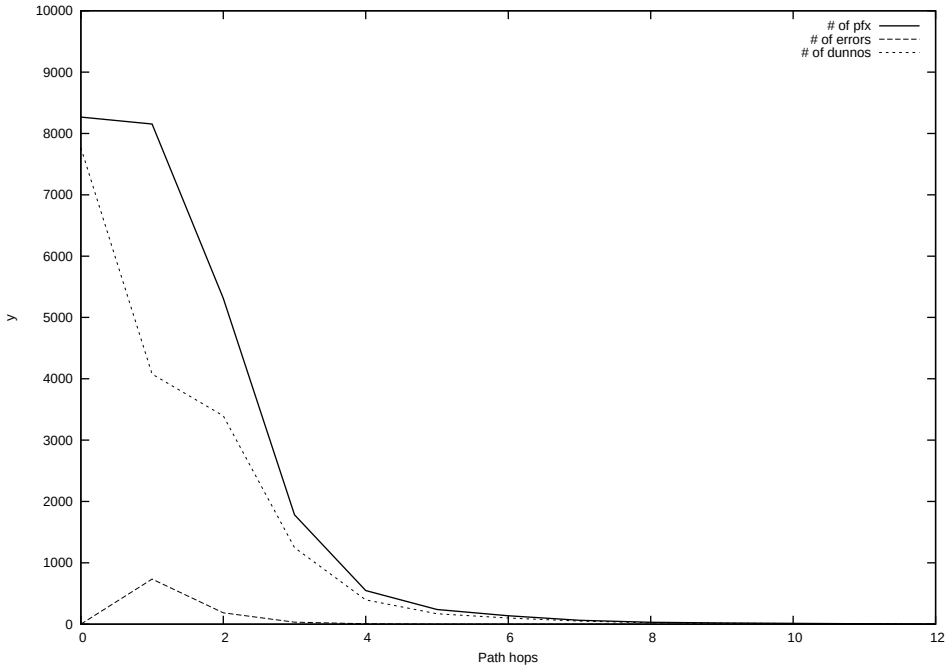
2012-03-23



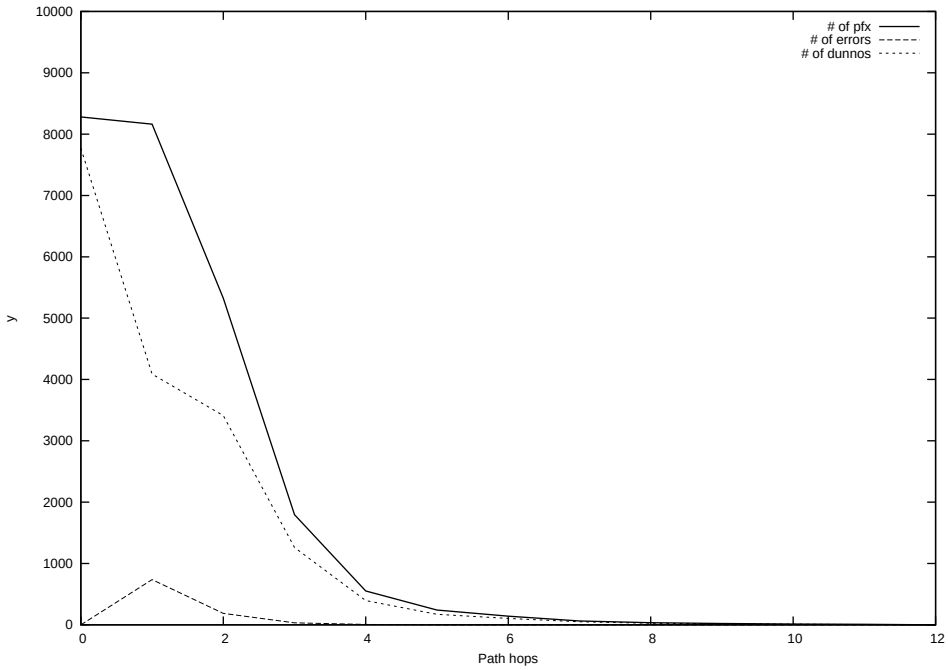
2012-03-24



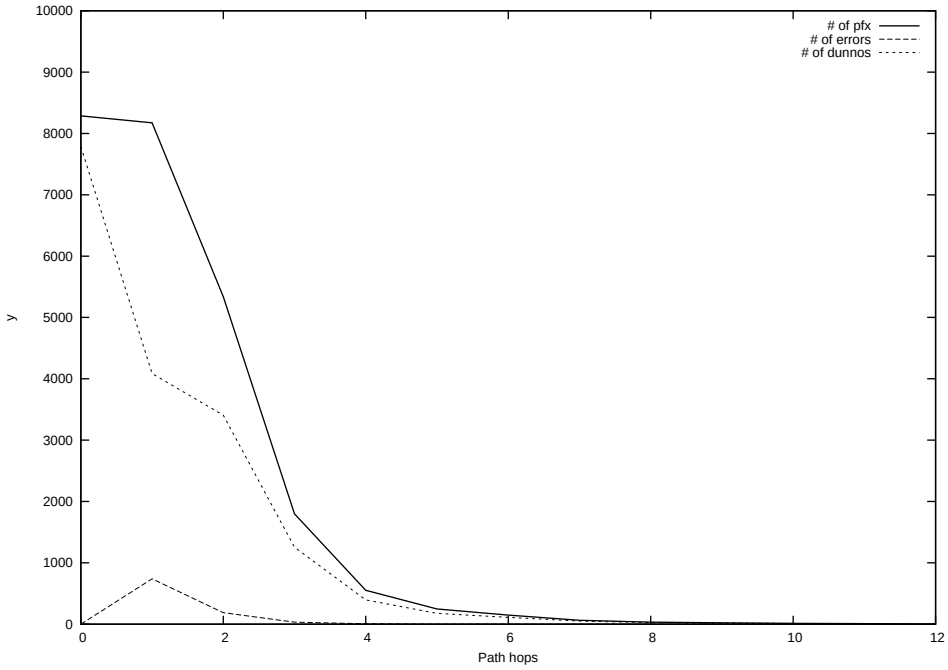
2012-03-25



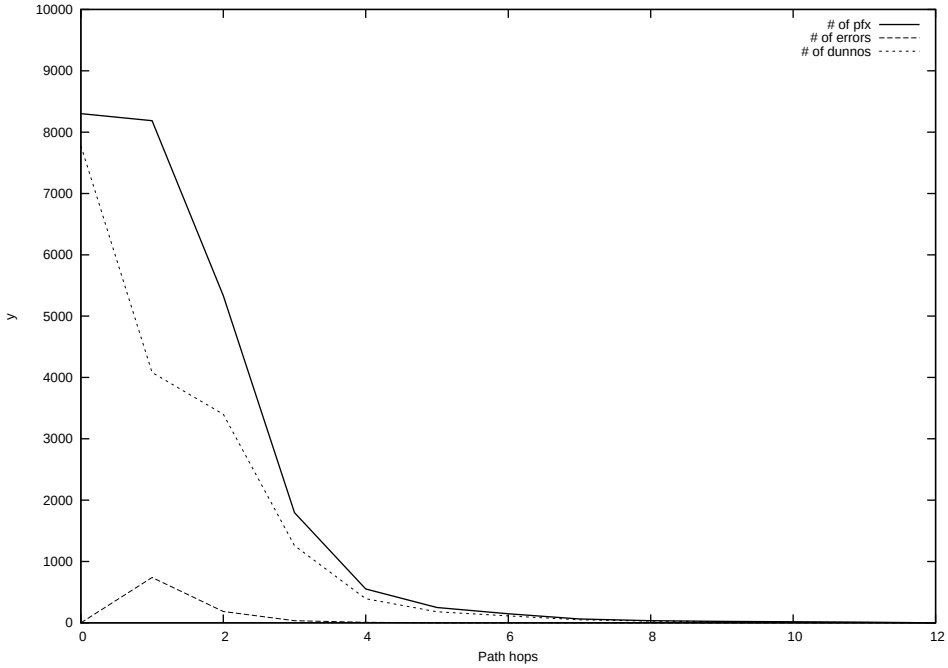
2012-03-26



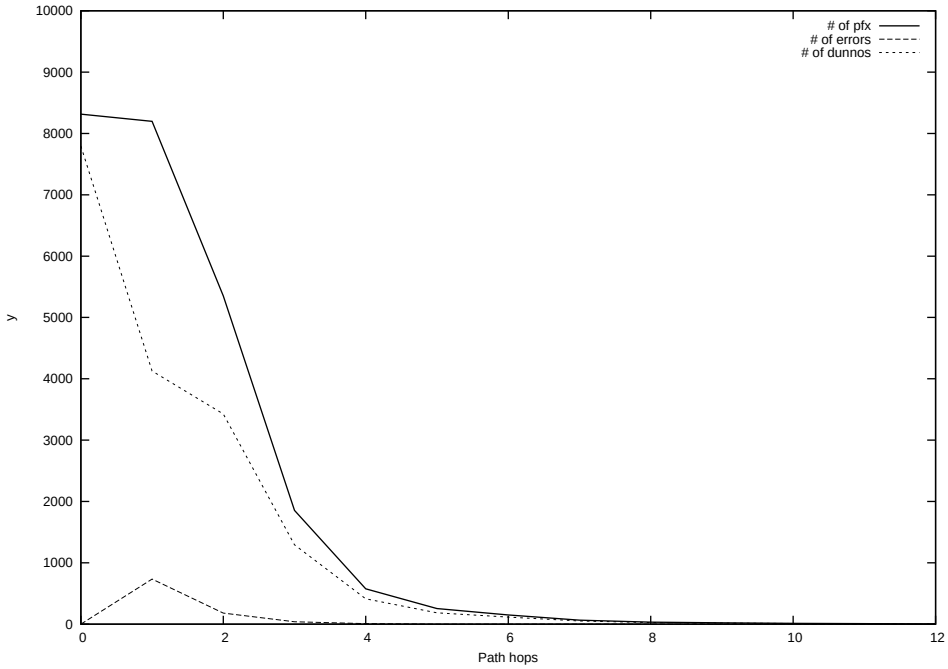
2012-03-27



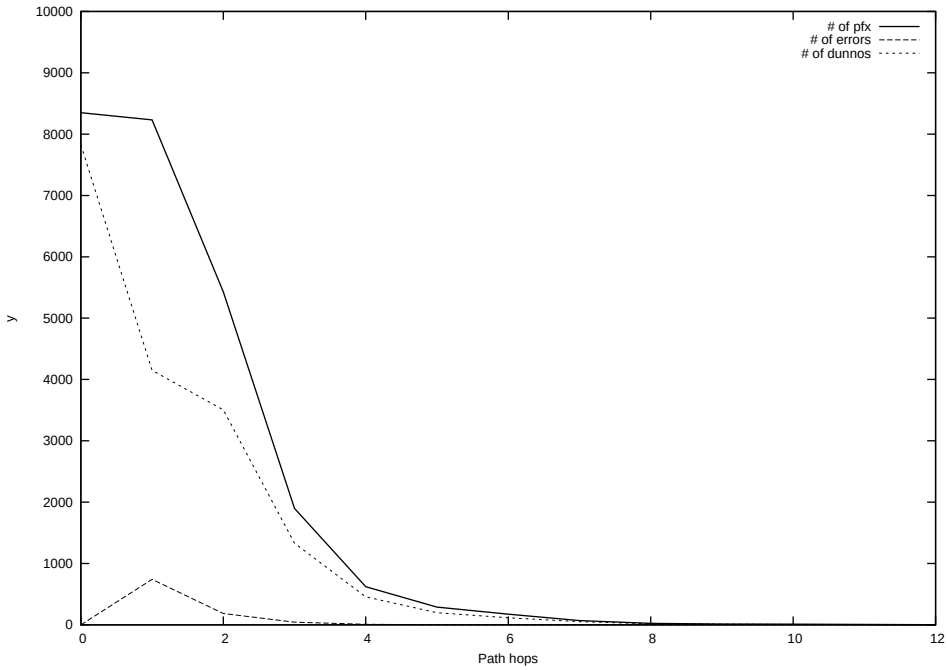
2012-03-28



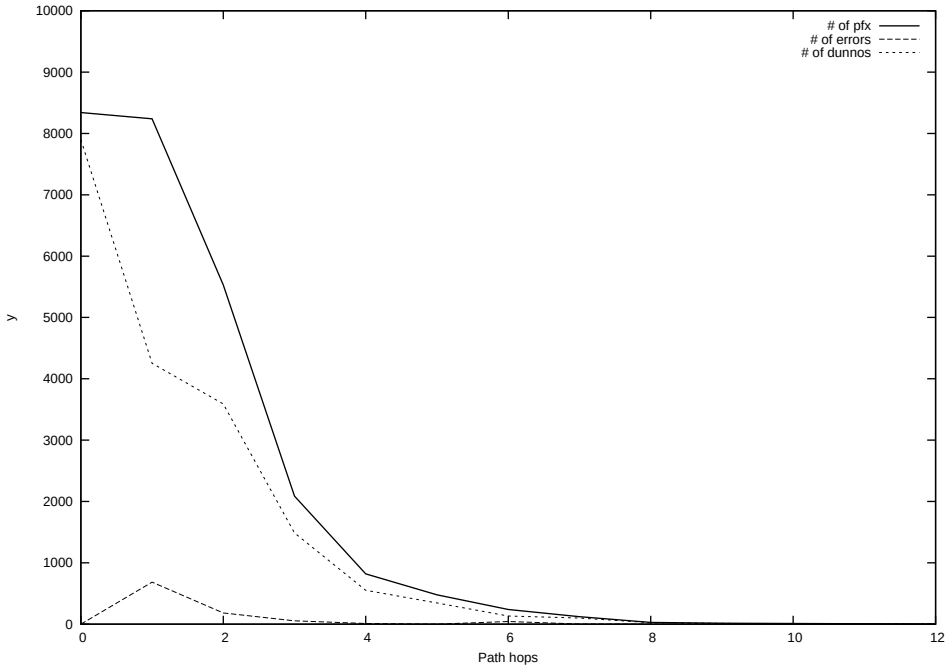
2012-03-29



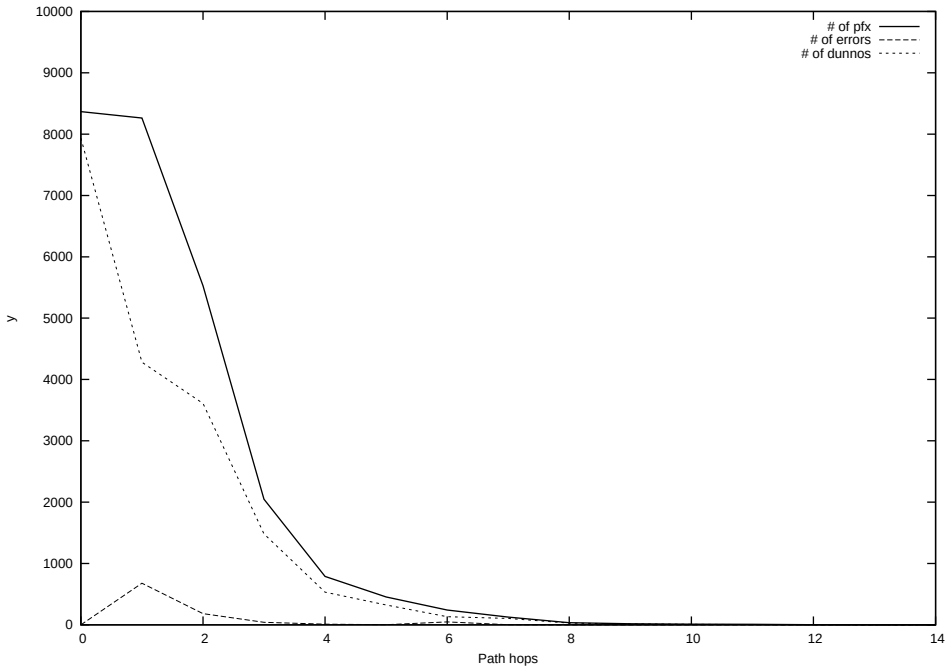
2012-03-30



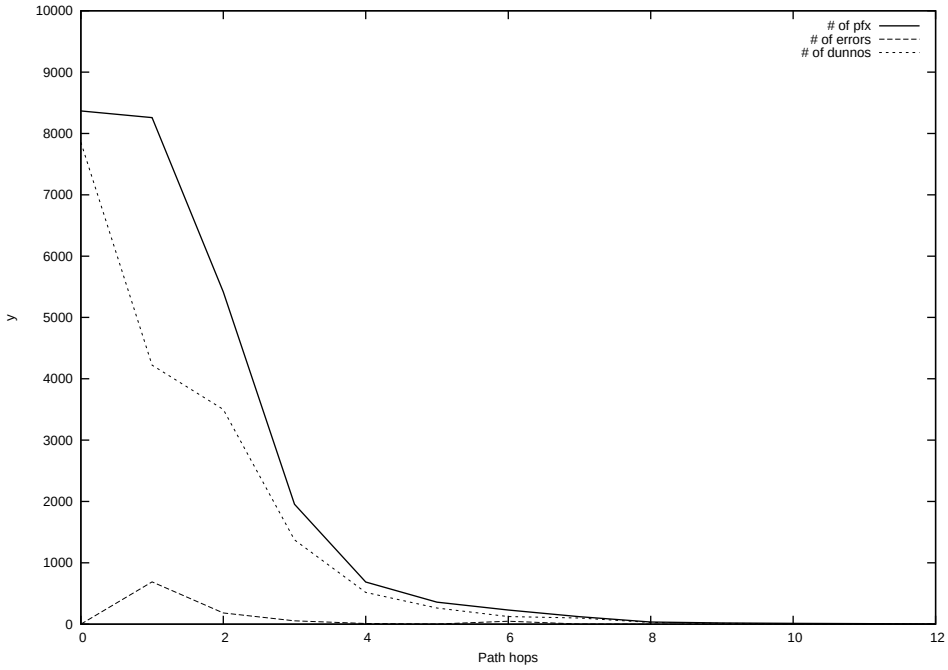
2012-03-31



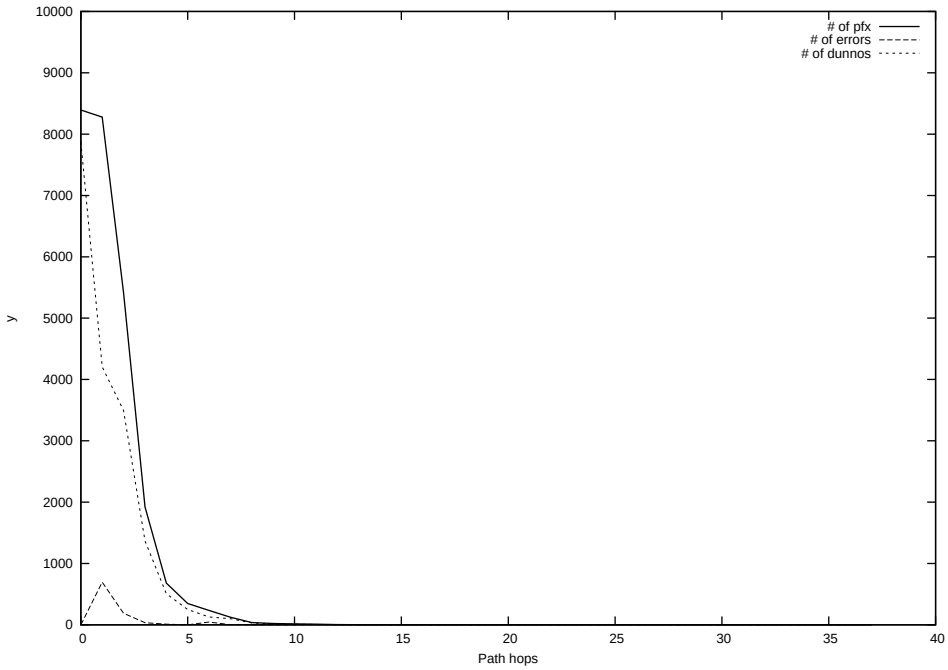
2012-04-01



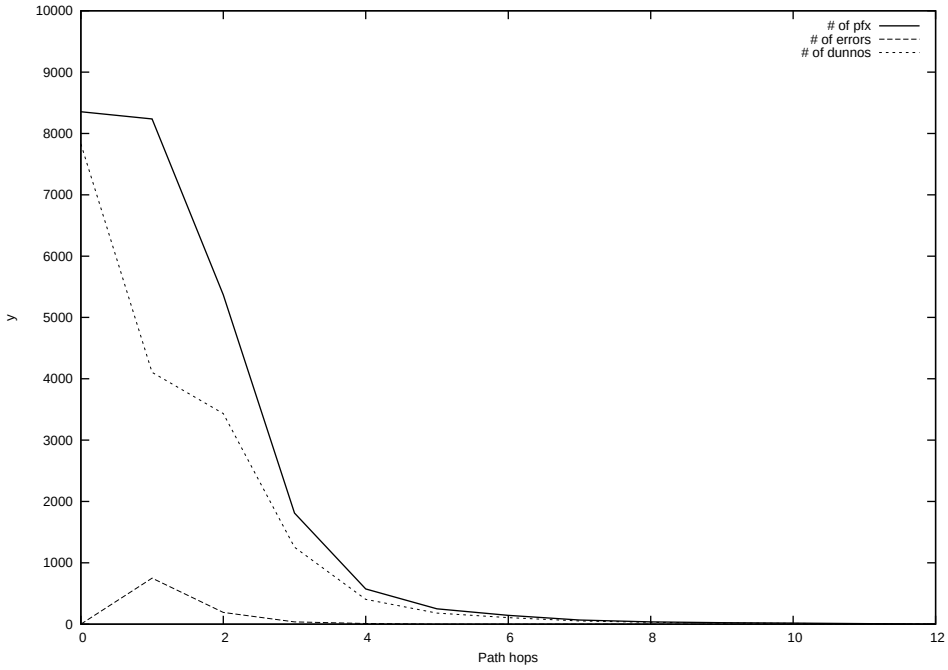
2012-04-02



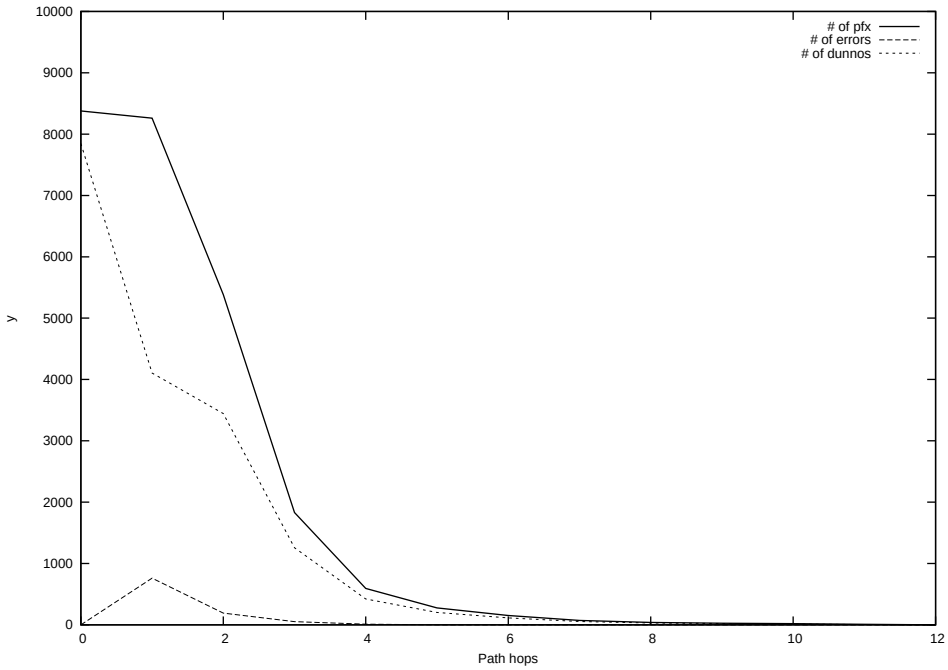
2012-04-03



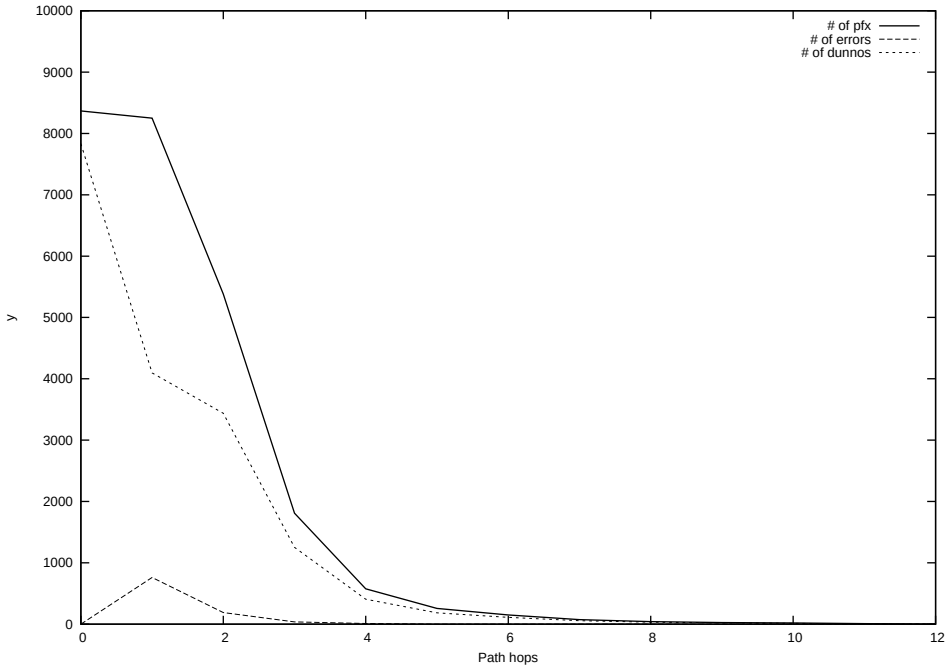
2012-04-04



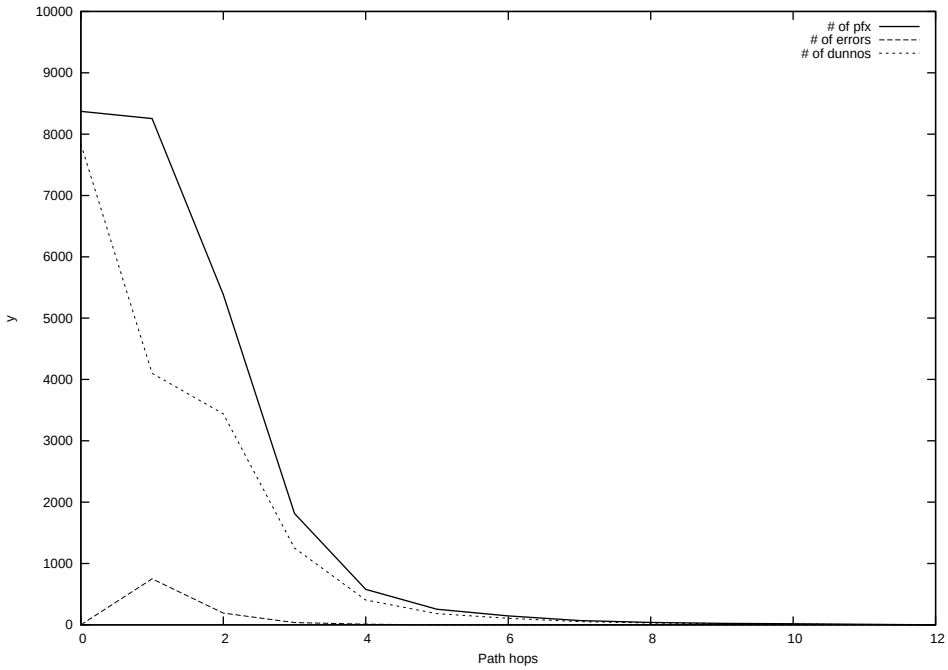
2012-04-05



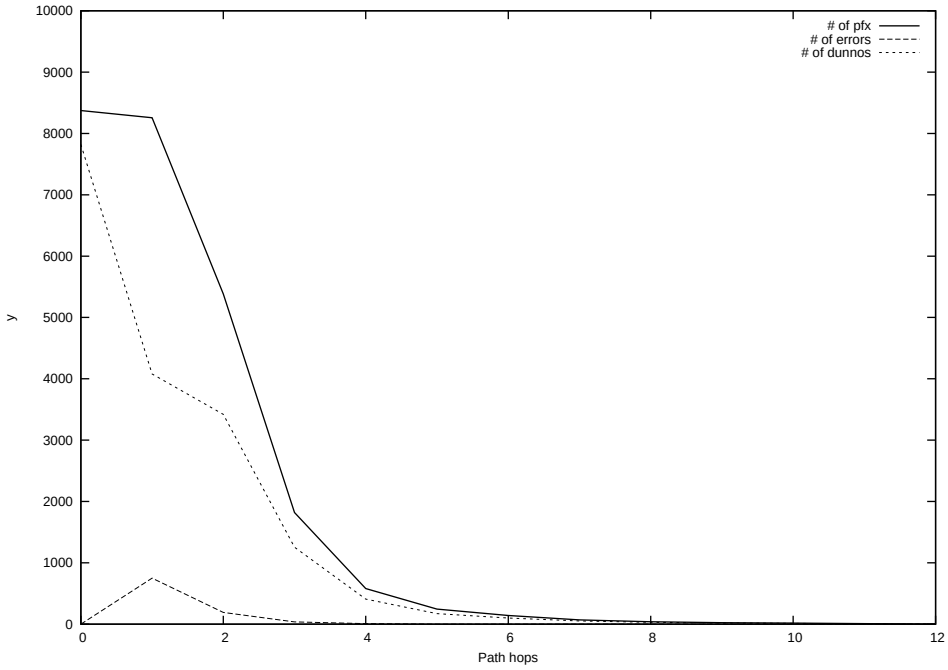
2012-04-06



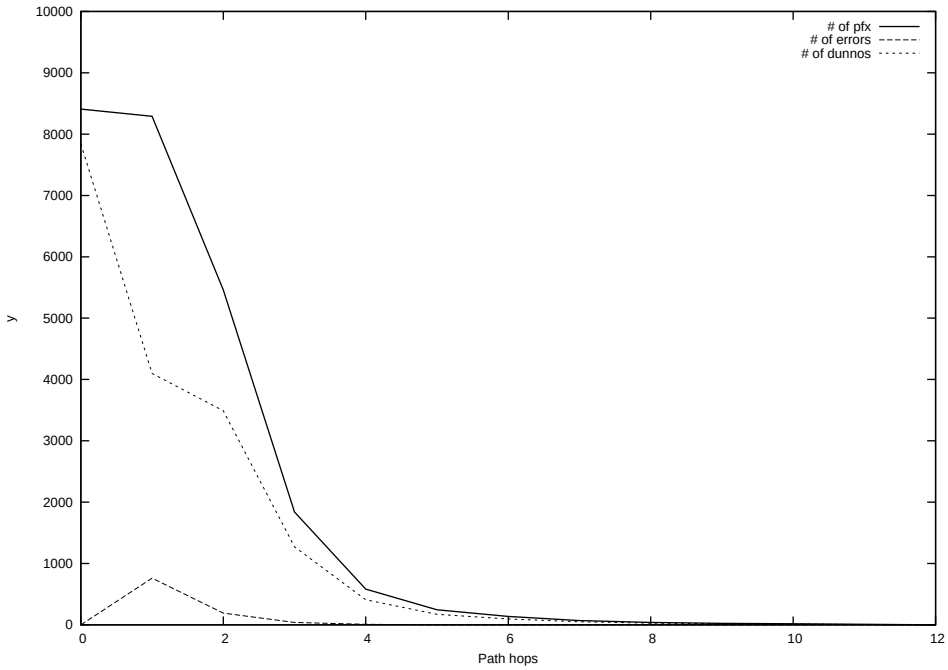
2012-04-07



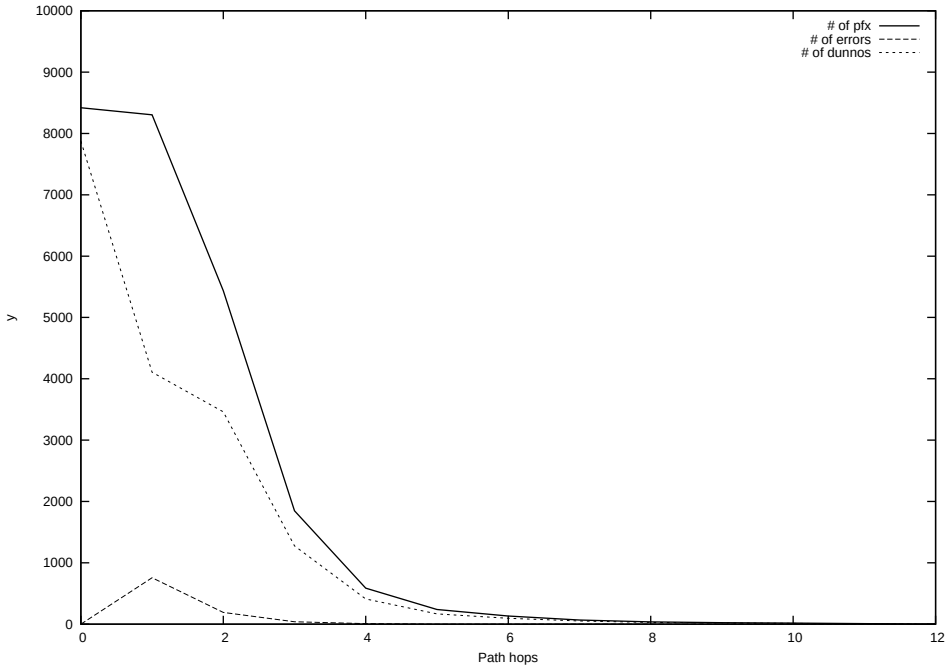
2012-04-08



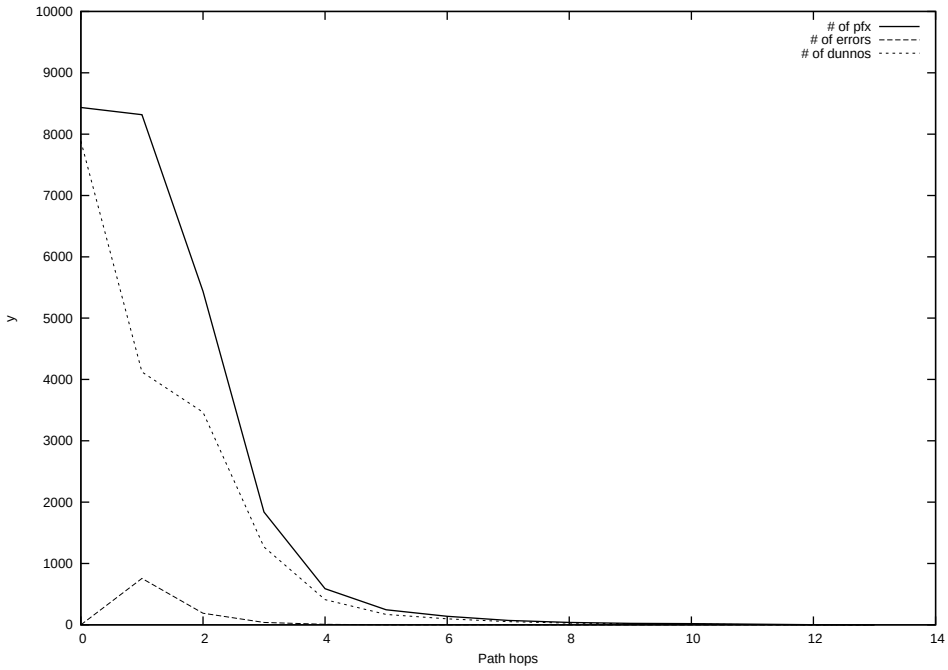
2012-04-09



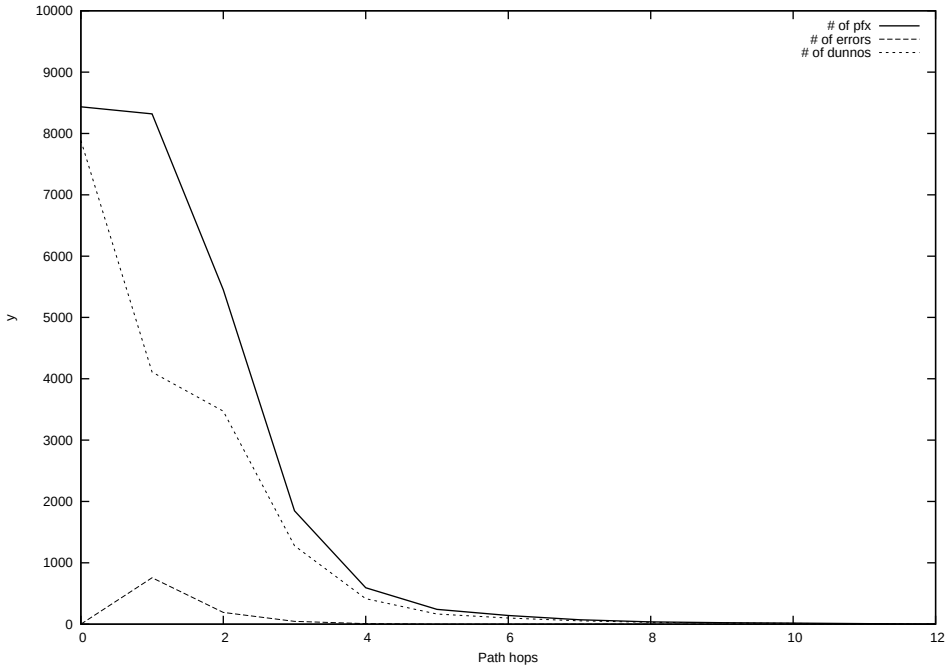
2012-04-10



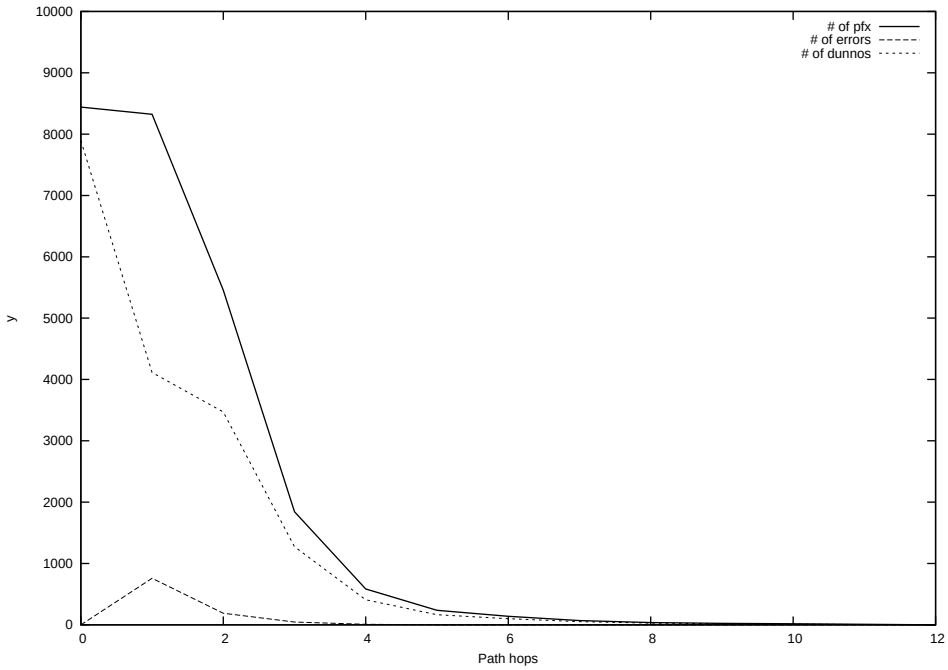
2012-04-11



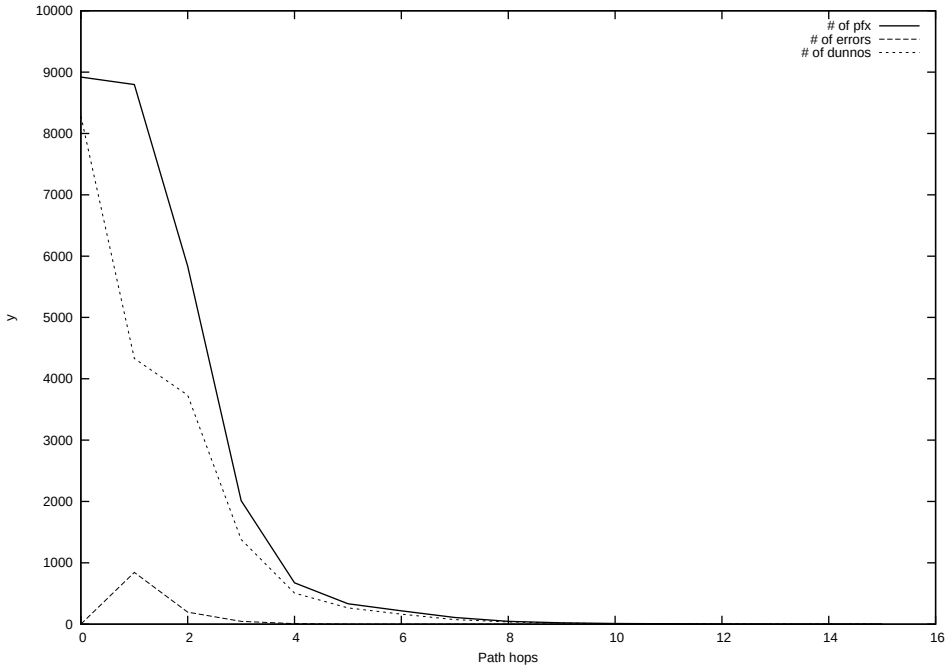
2012-04-12



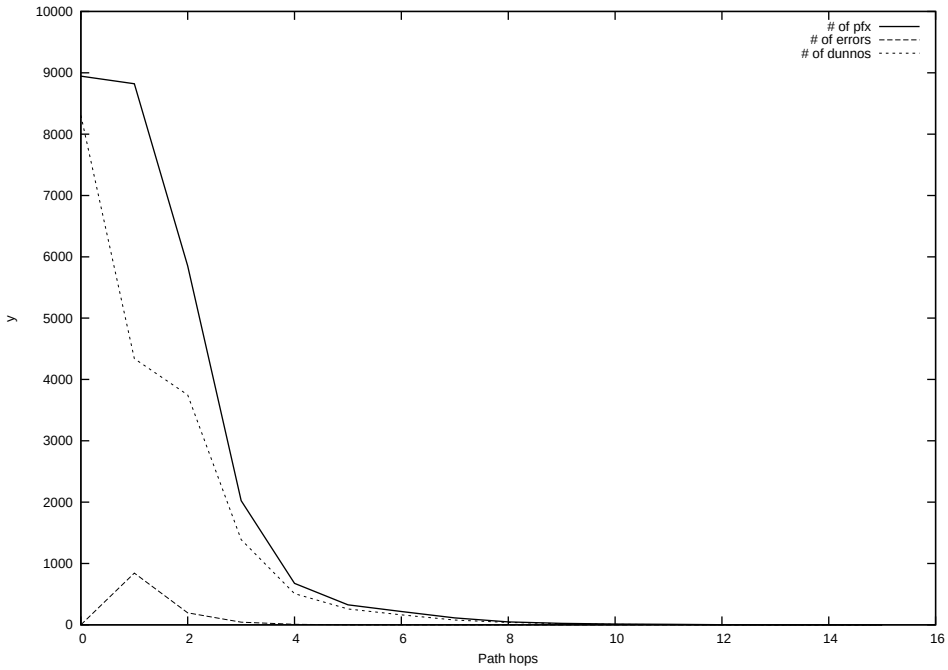
2012-04-13



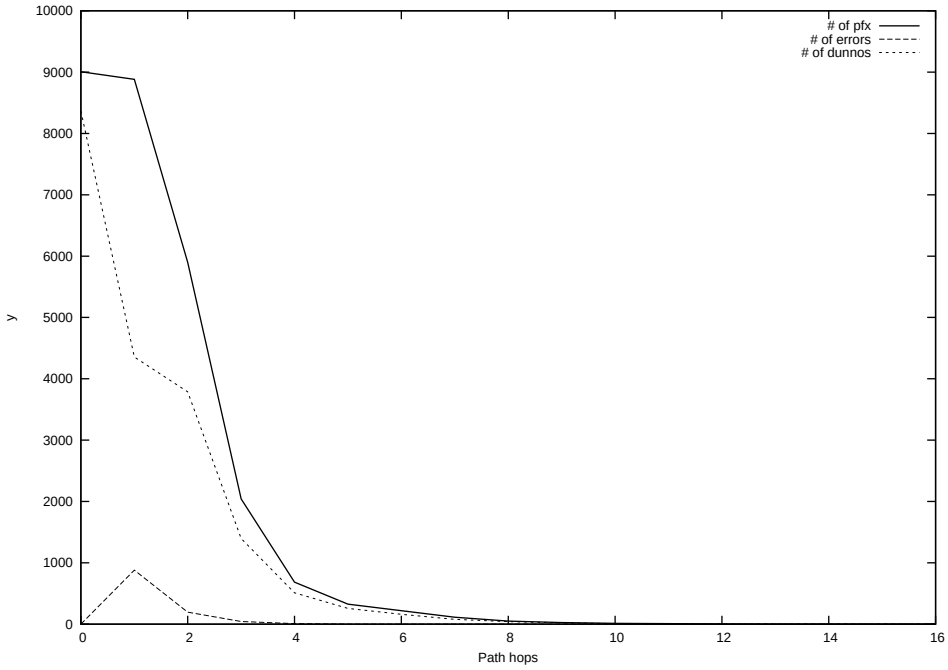
2012-04-14



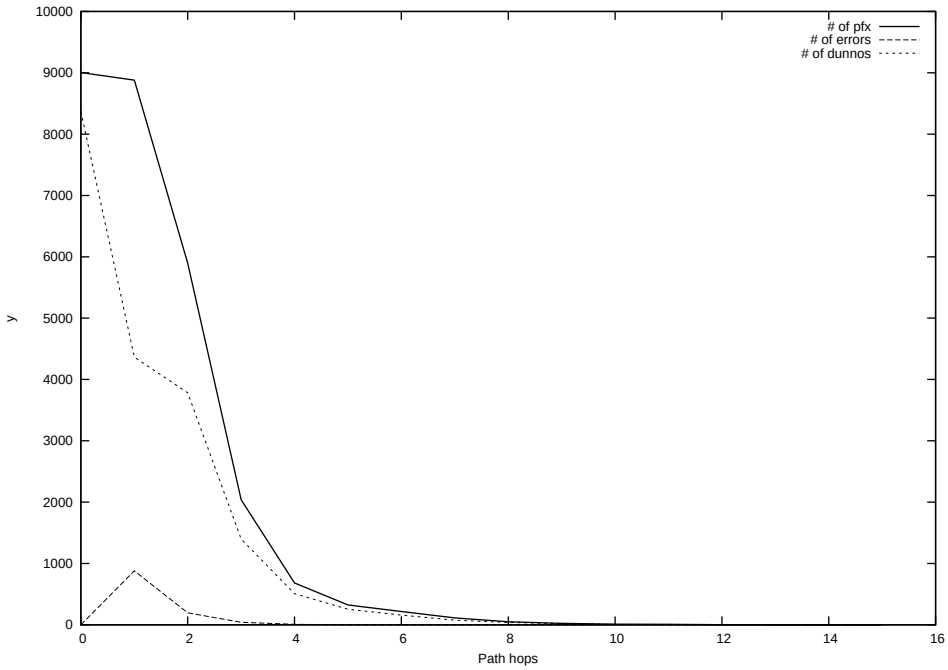
2012-05-16



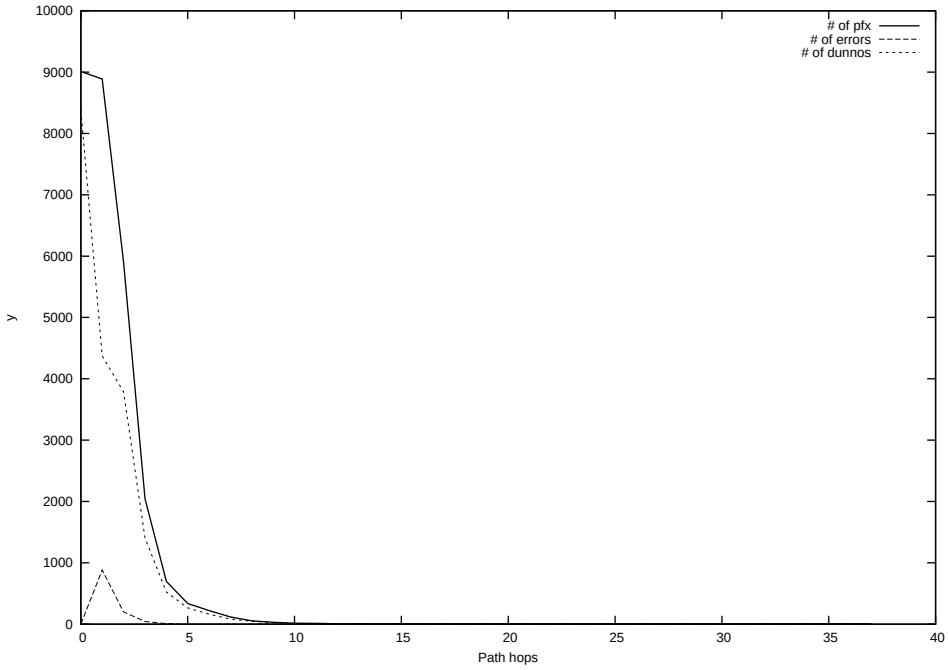
2012-05-17



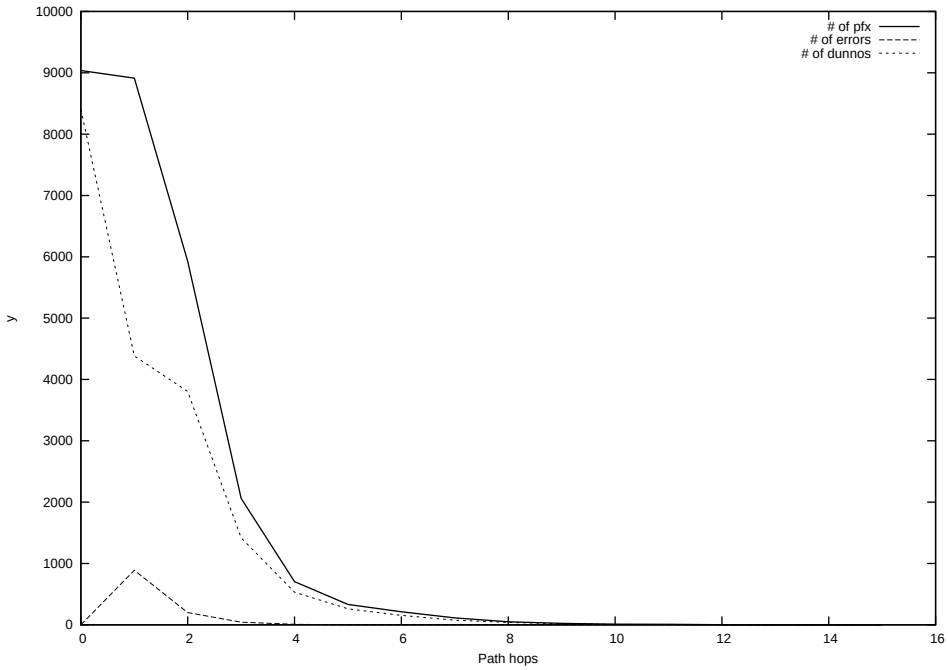
2012-05-19



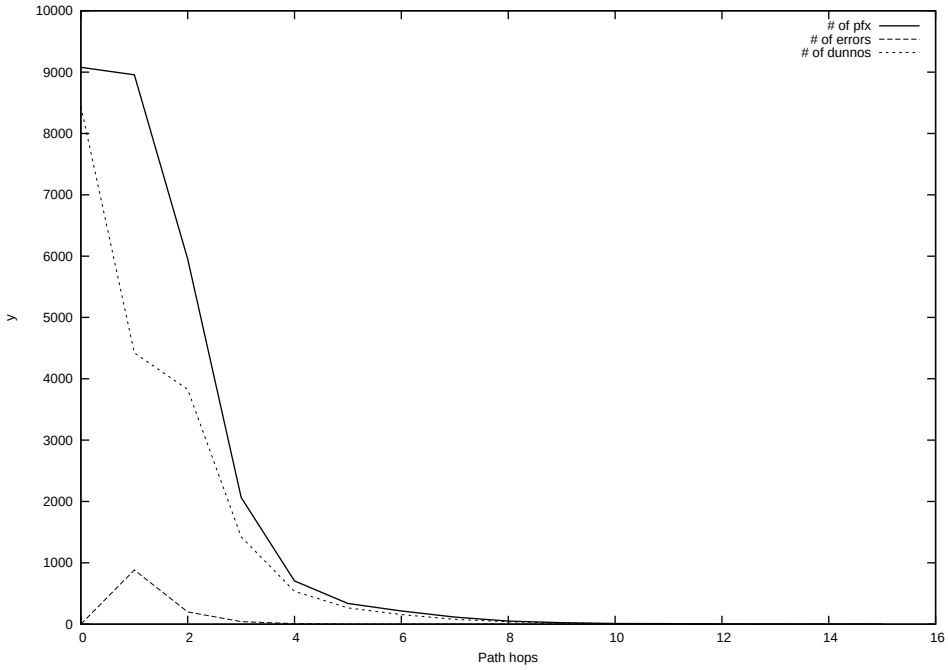
2012-05-20



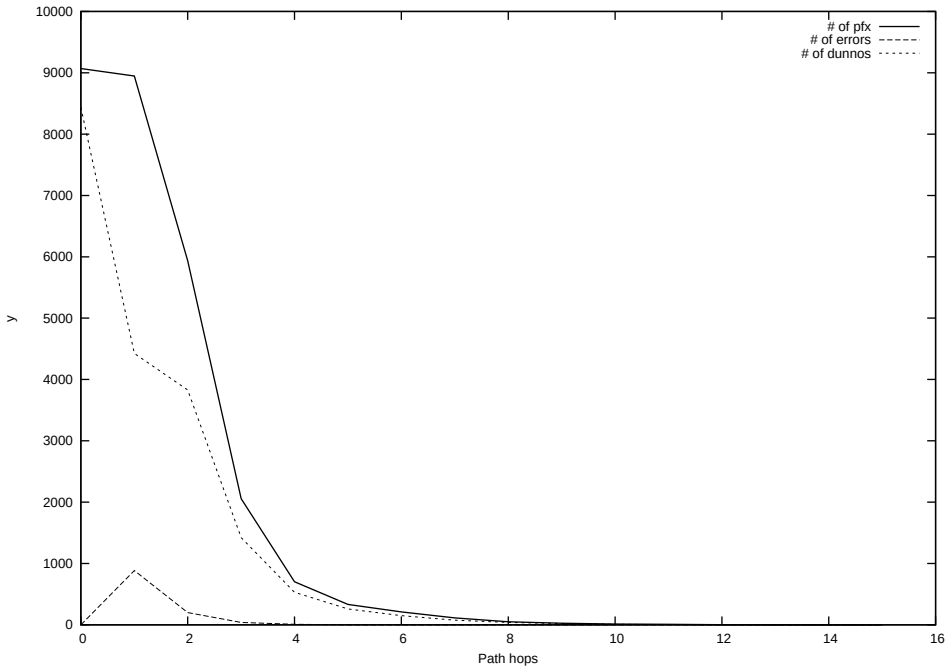
2012-05-21



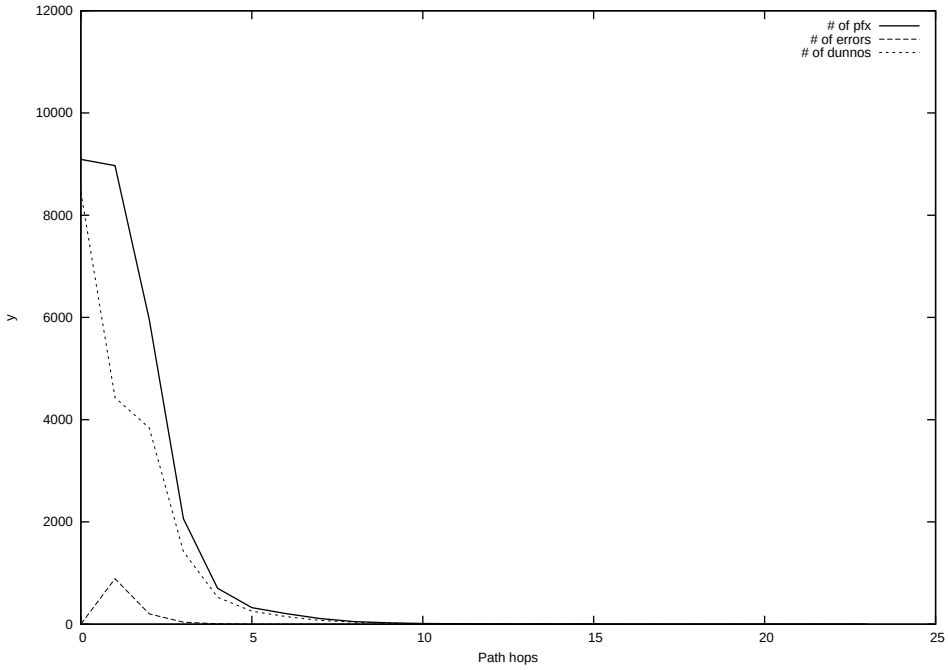
2012-05-22



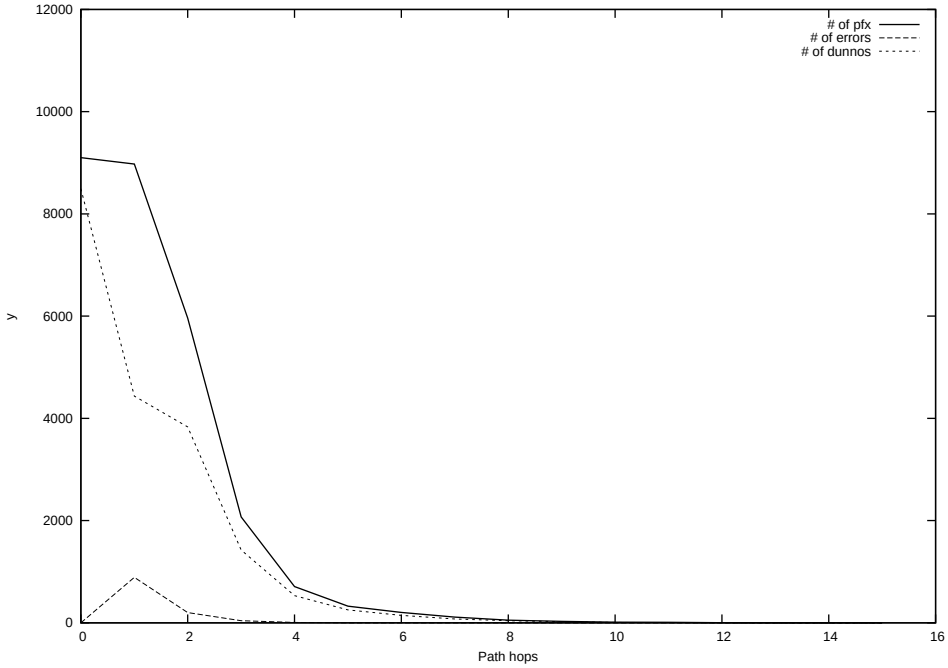
2012-05-23



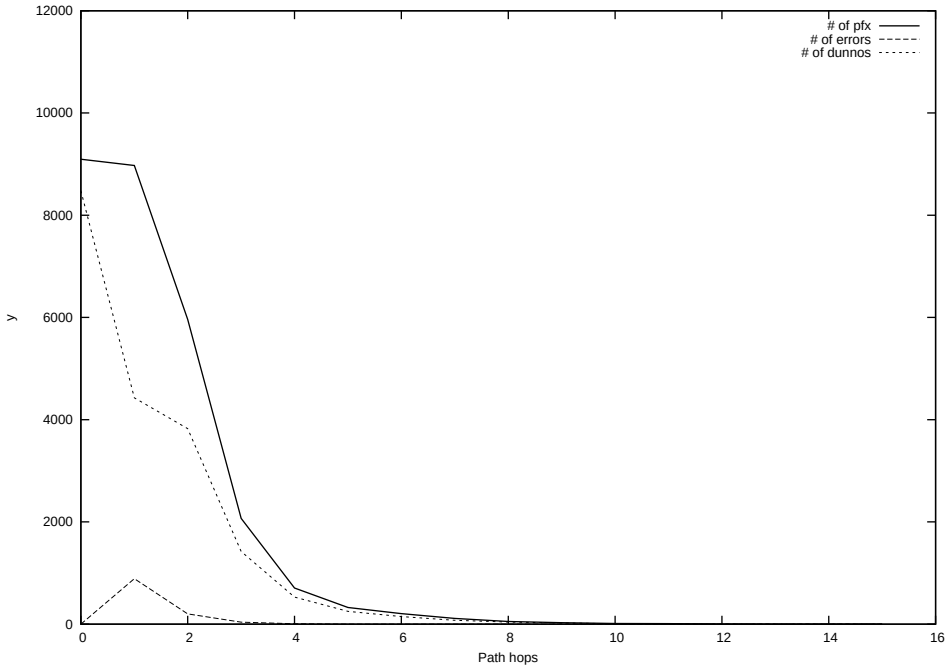
2012-05-24



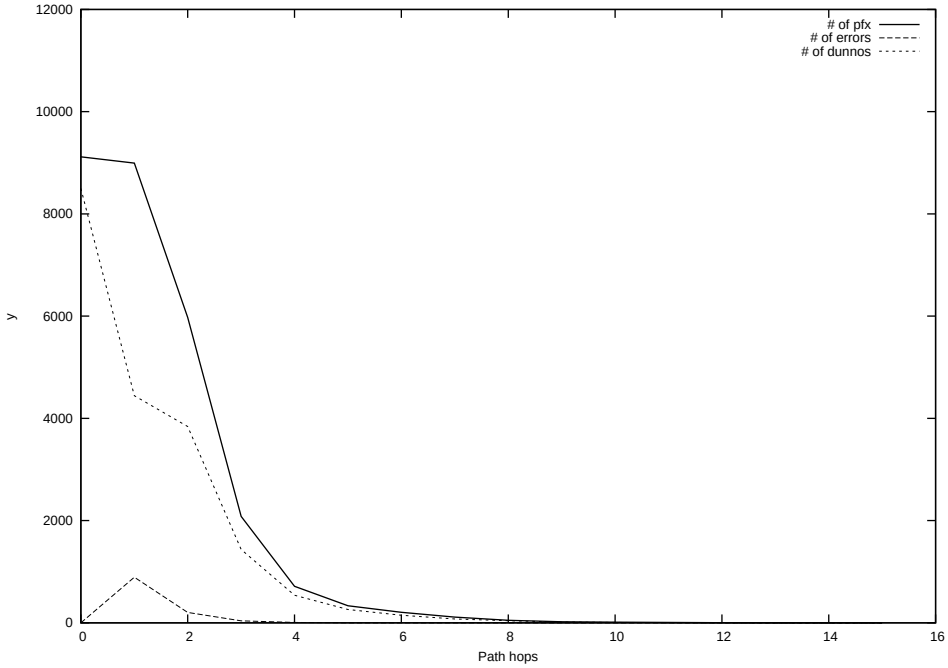
2012-05-25



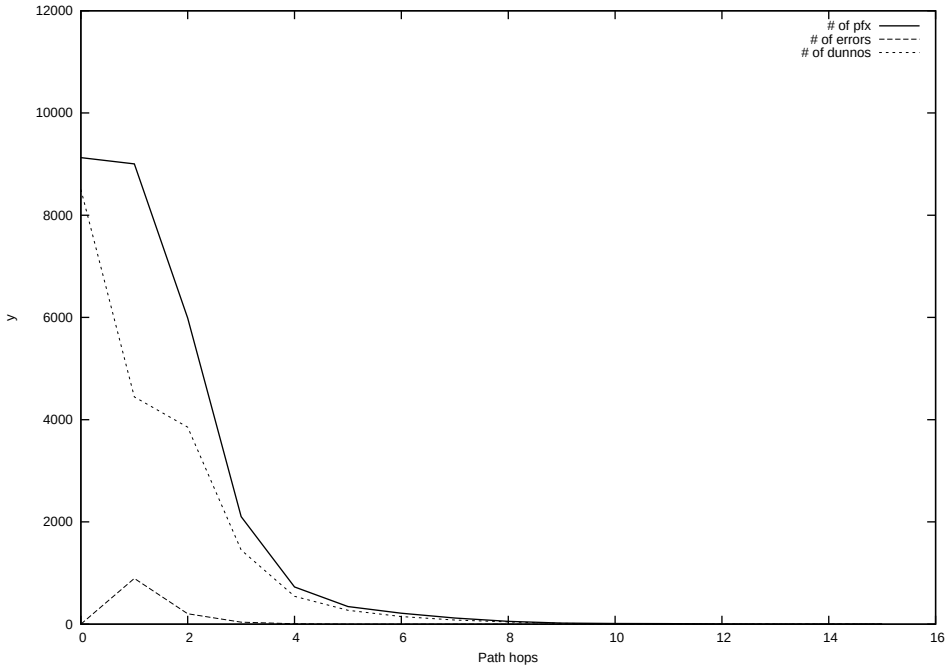
2012-05-26



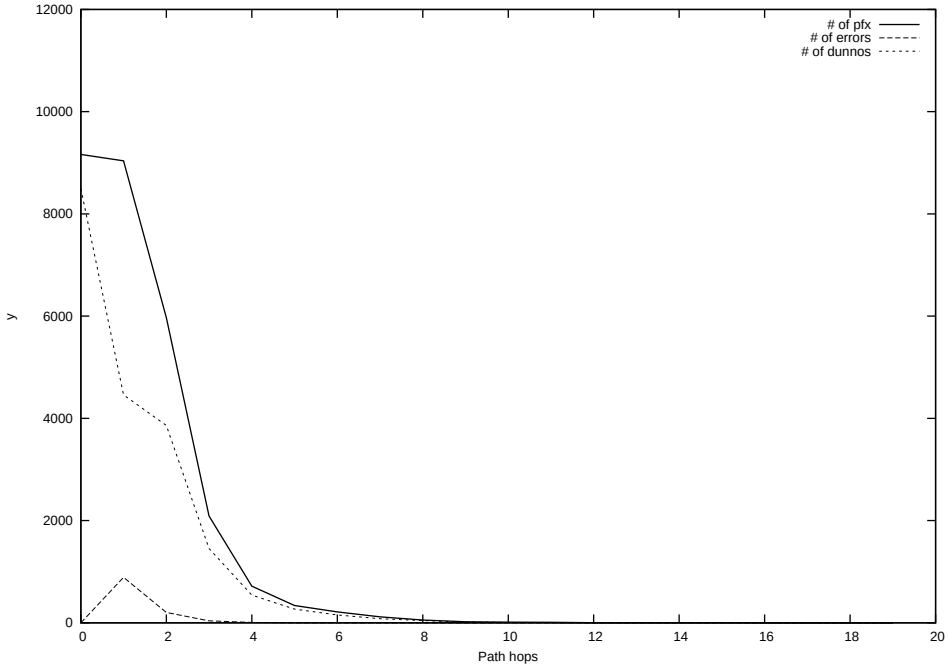
2012-05-27



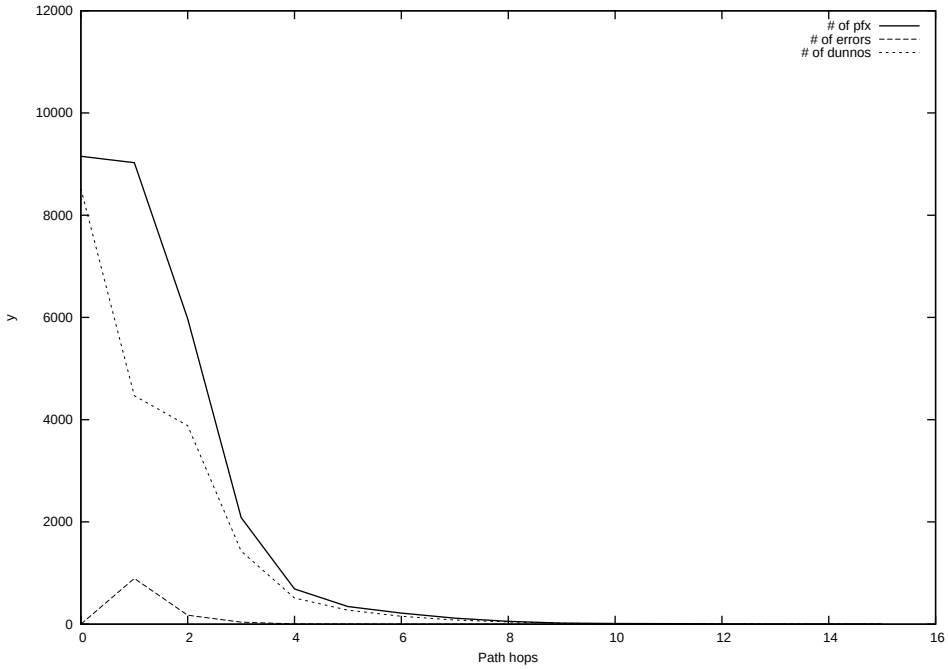
2012-05-28



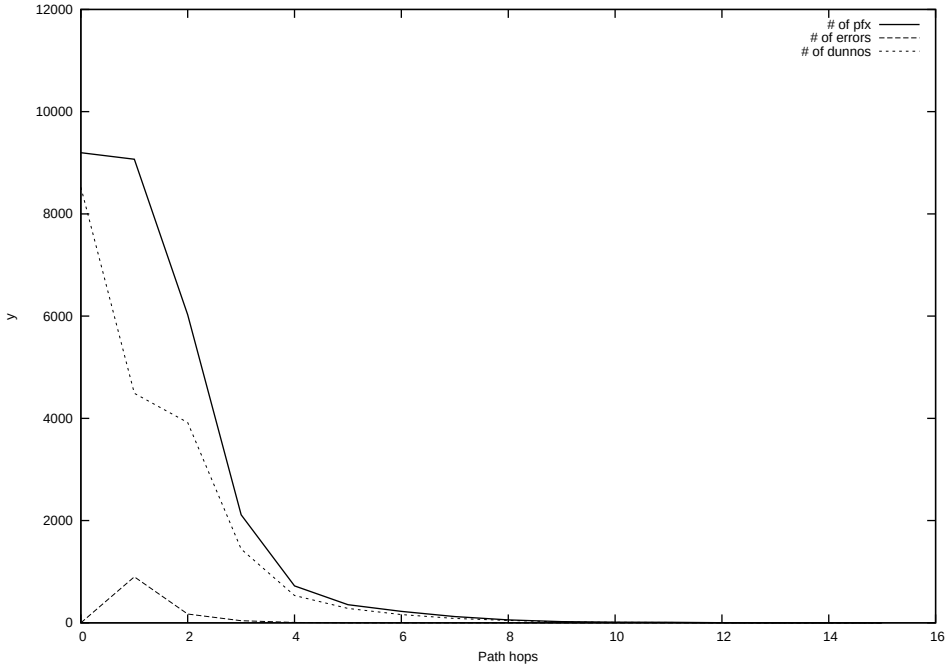
2012-05-29



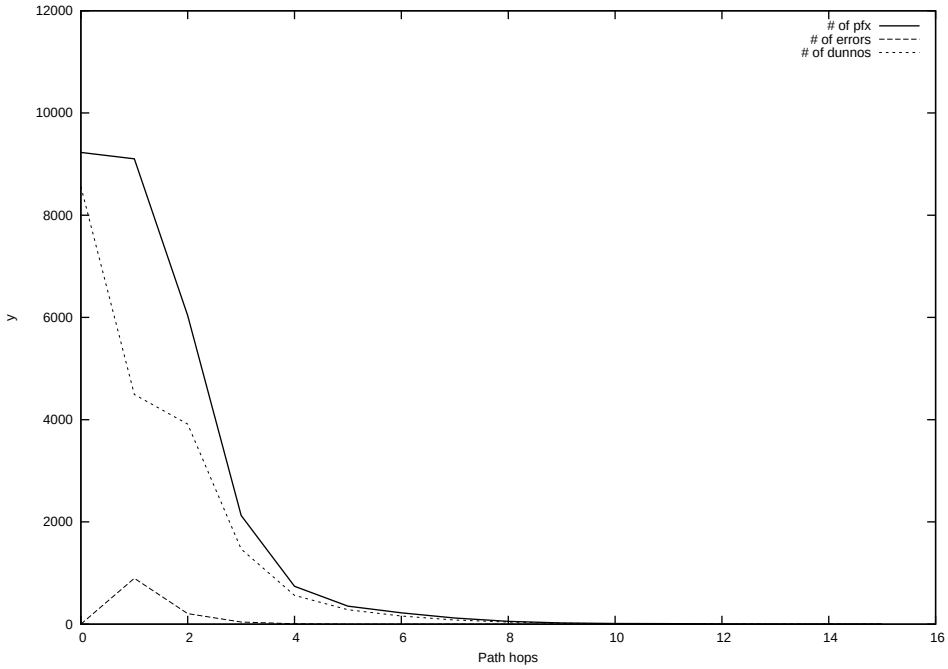
2012-05-30



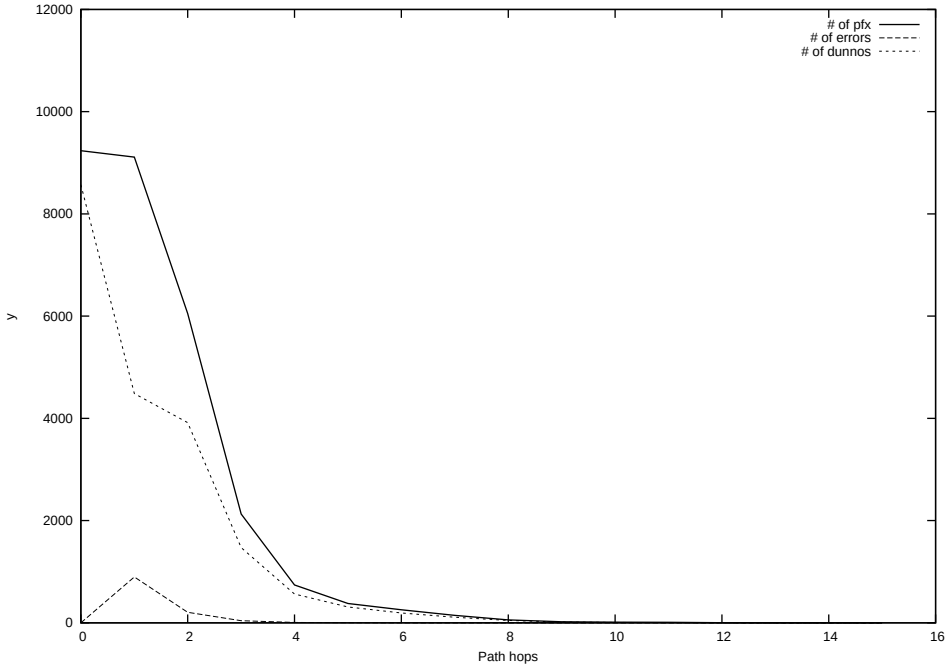
2012-05-31



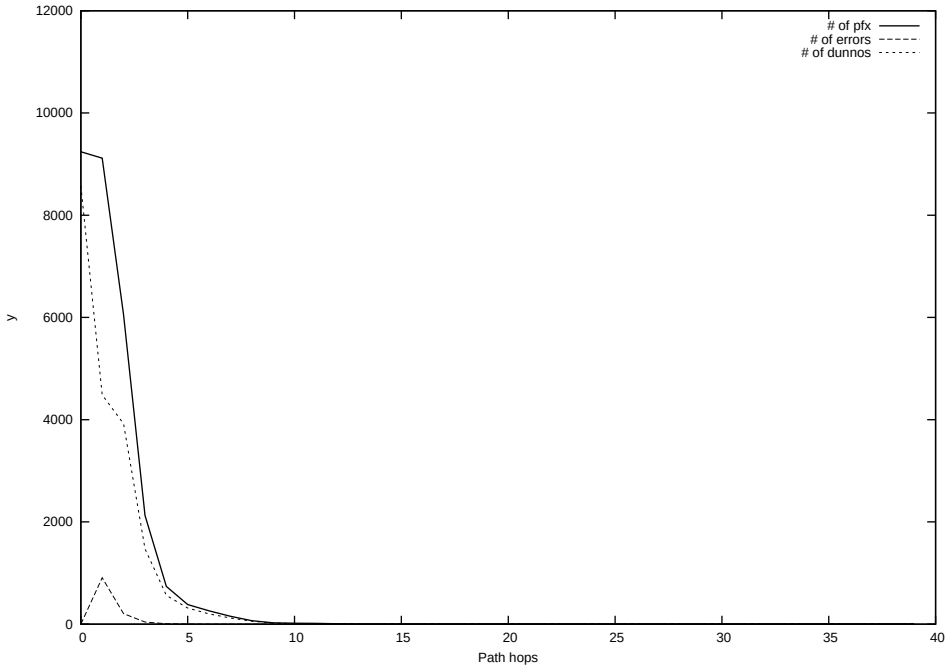
2012-06-01



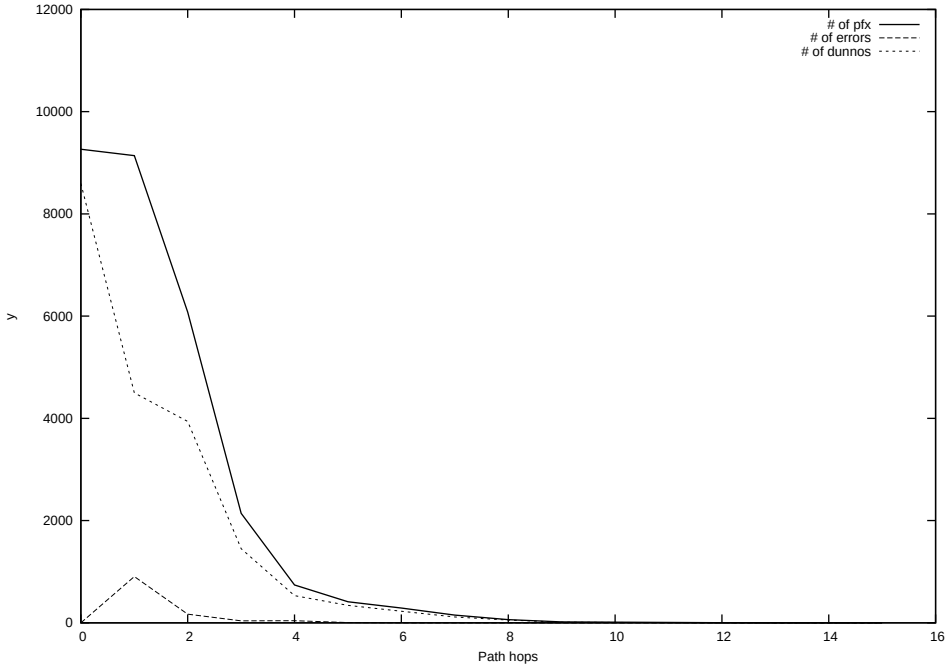
2012-06-02



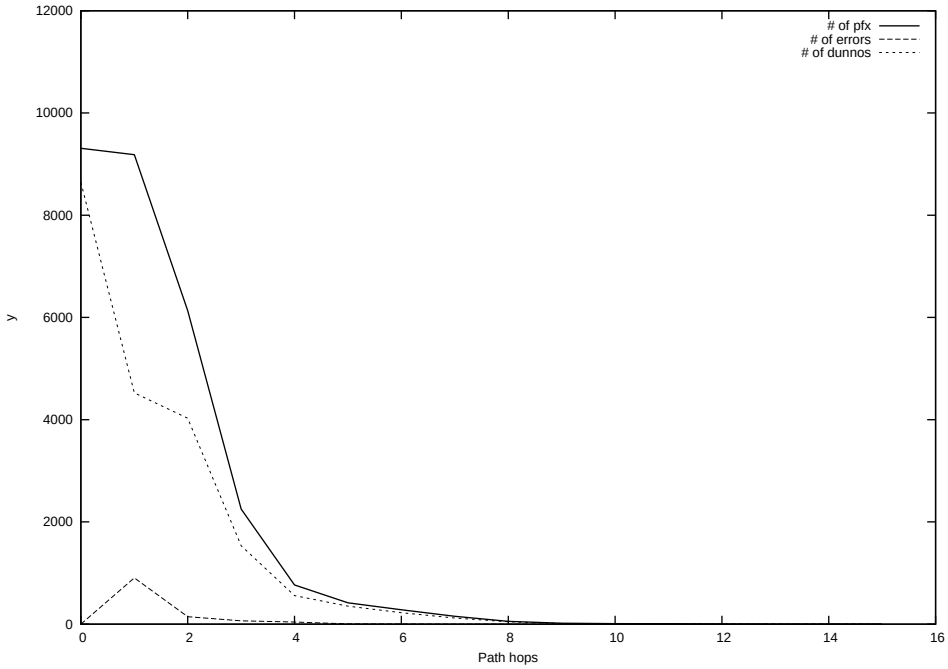
2012-06-03



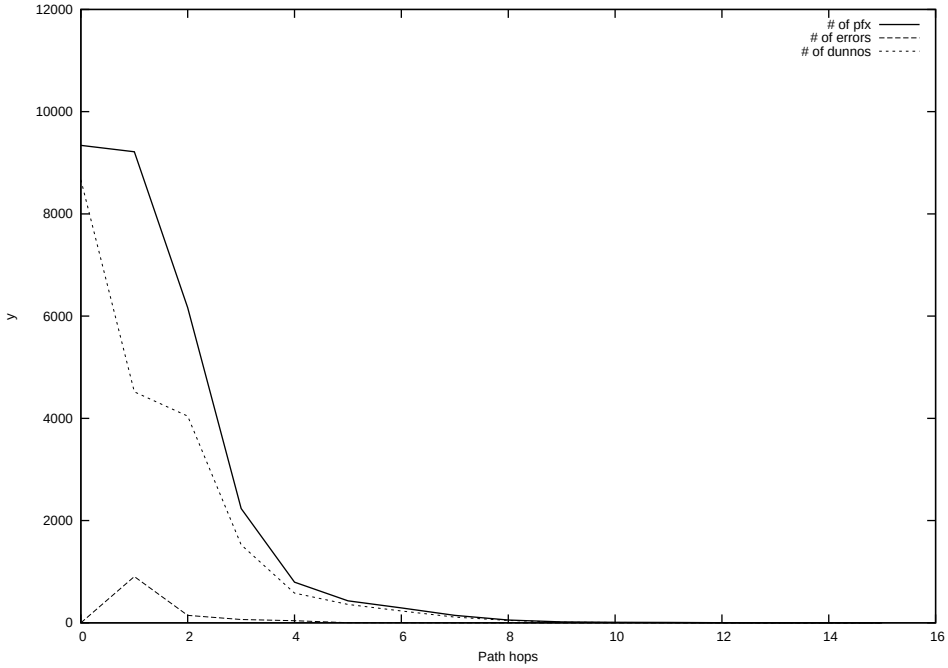
2012-06-04



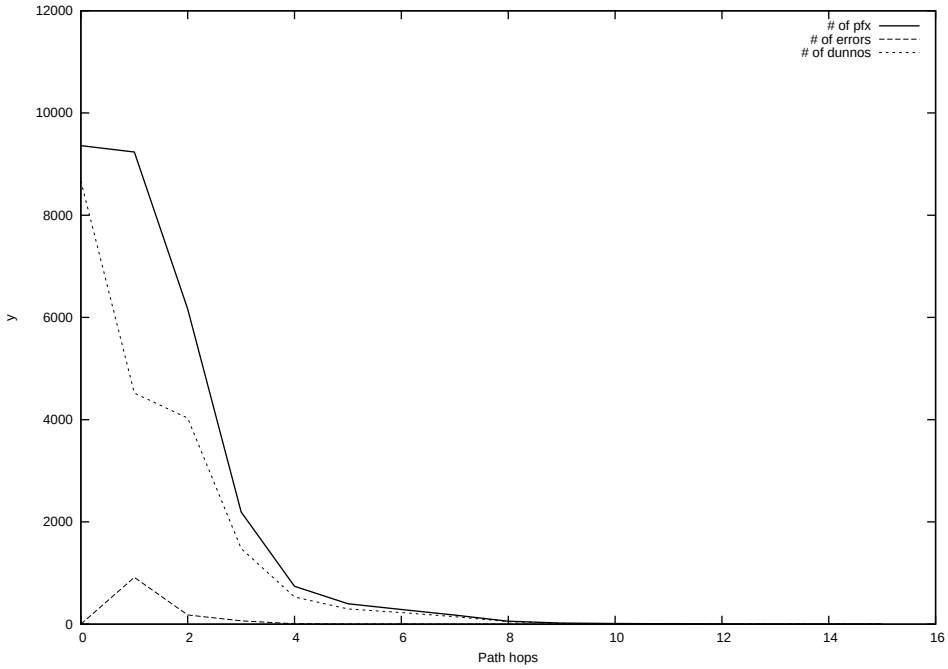
2012-06-05



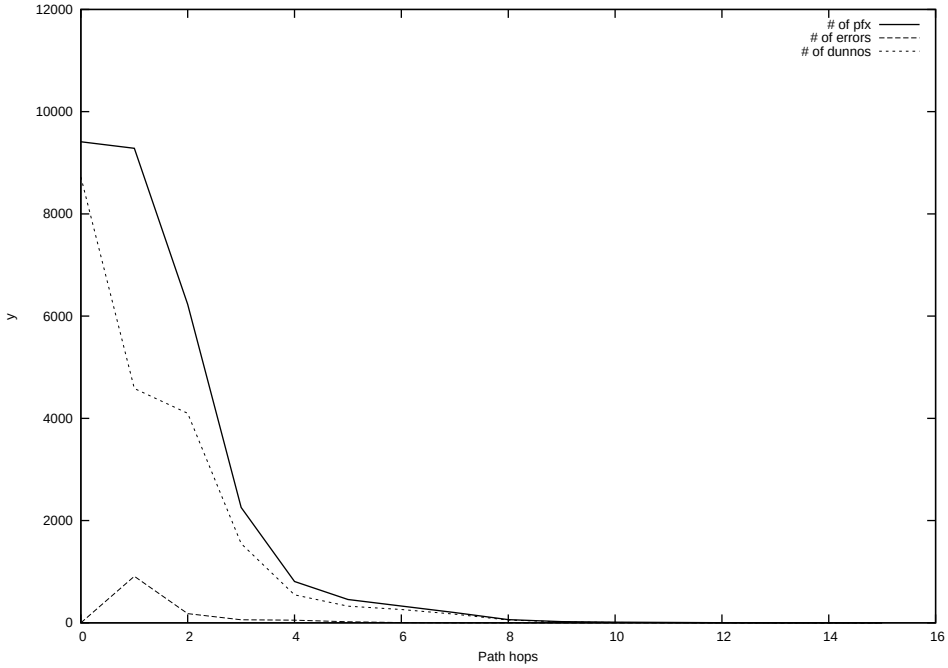
2012-06-06



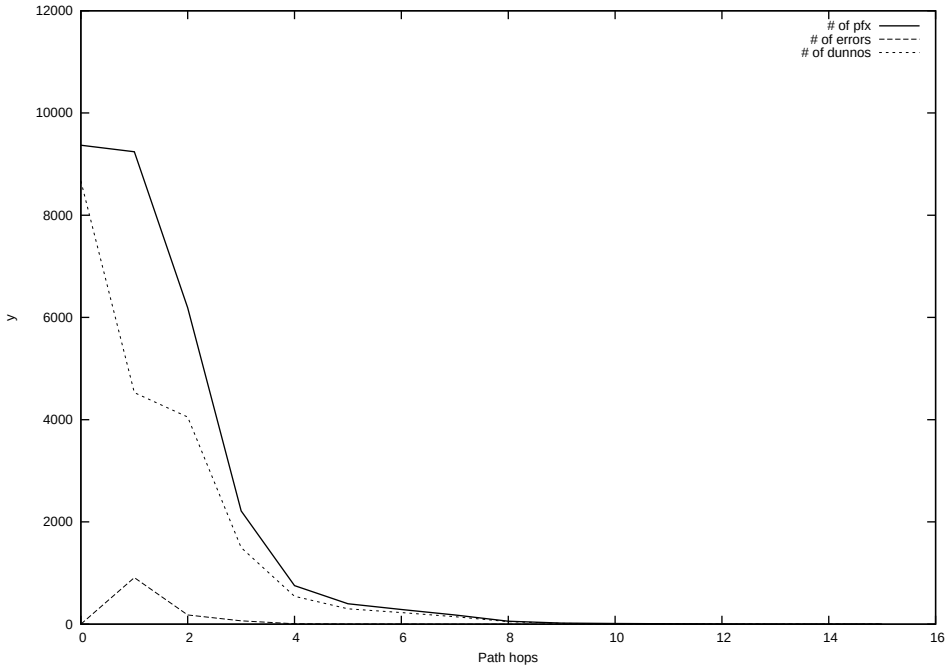
2012-06-07



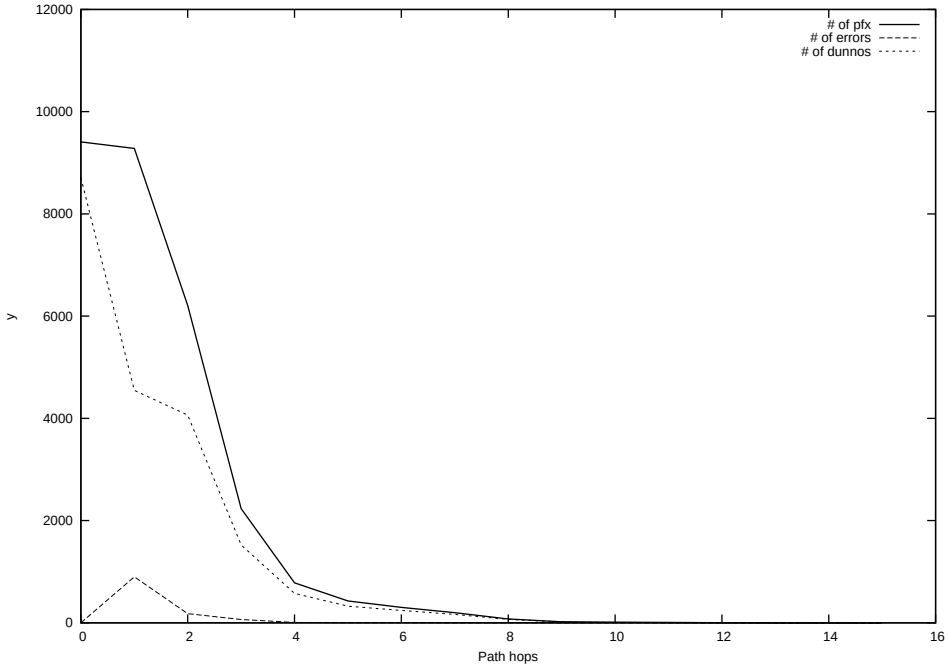
2012-06-08



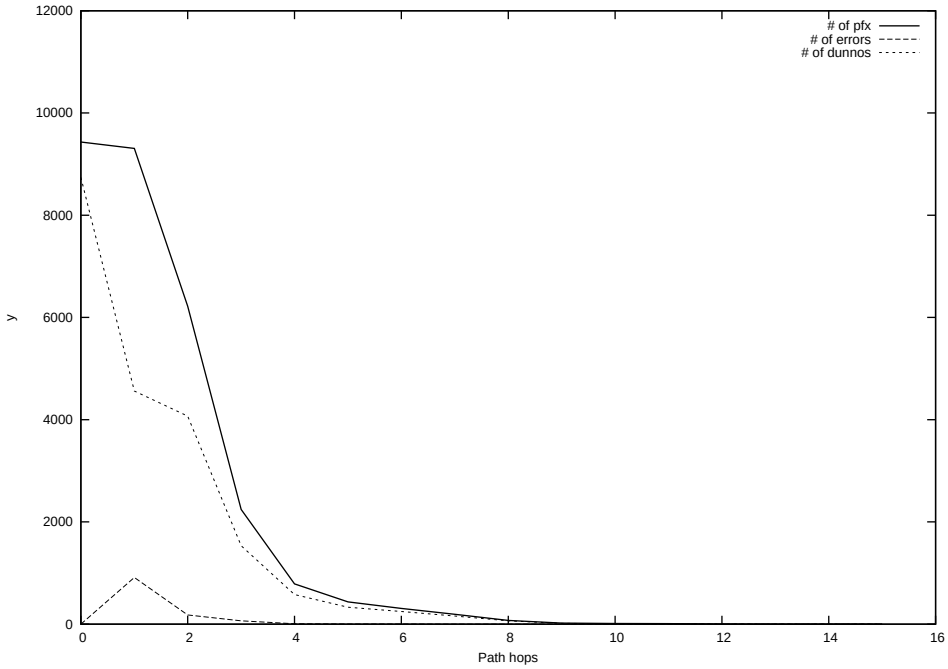
2012-06-10



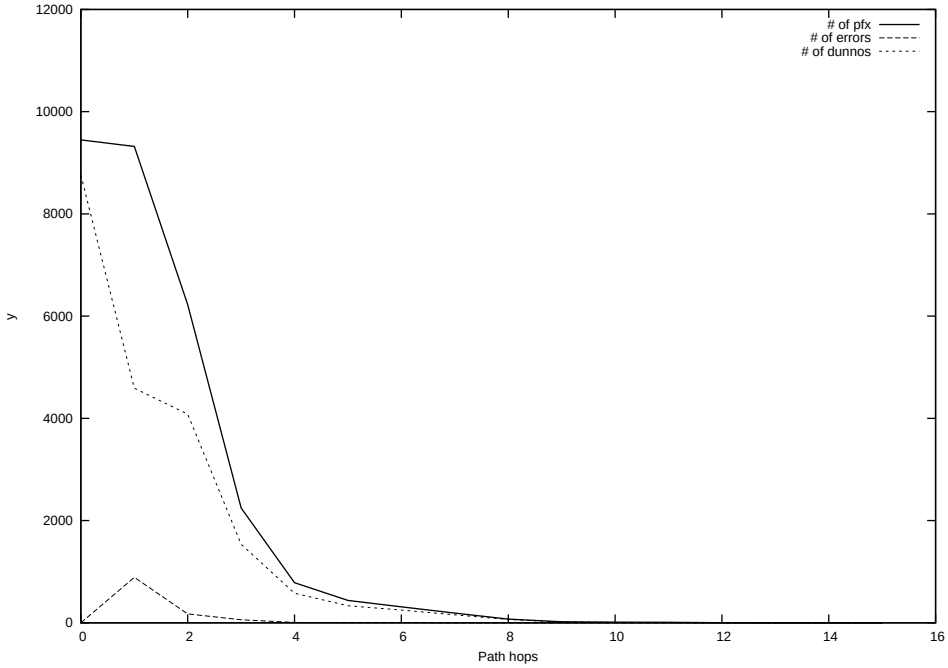
2012-06-11



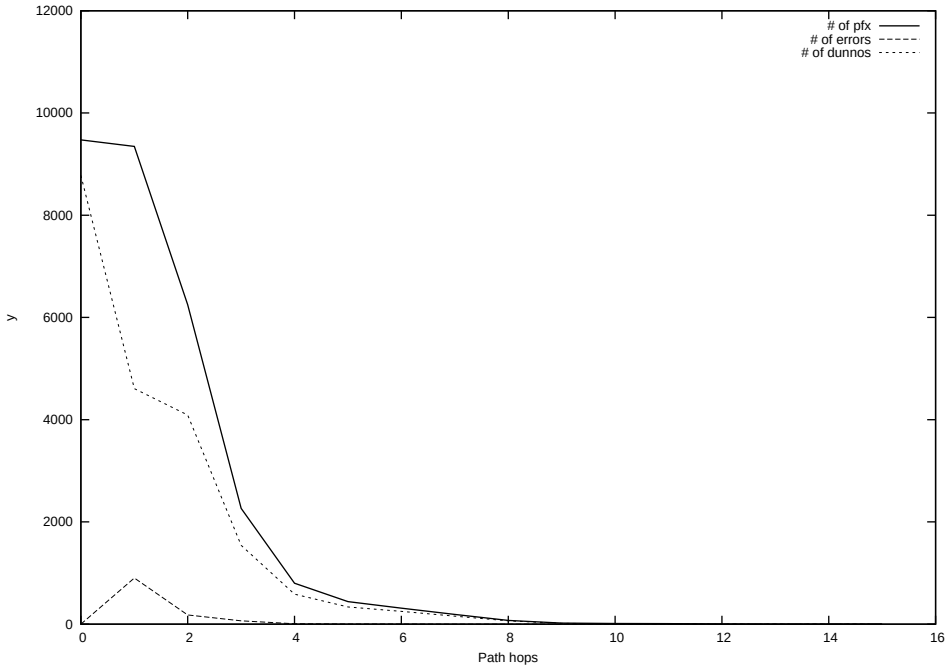
2012-06-12



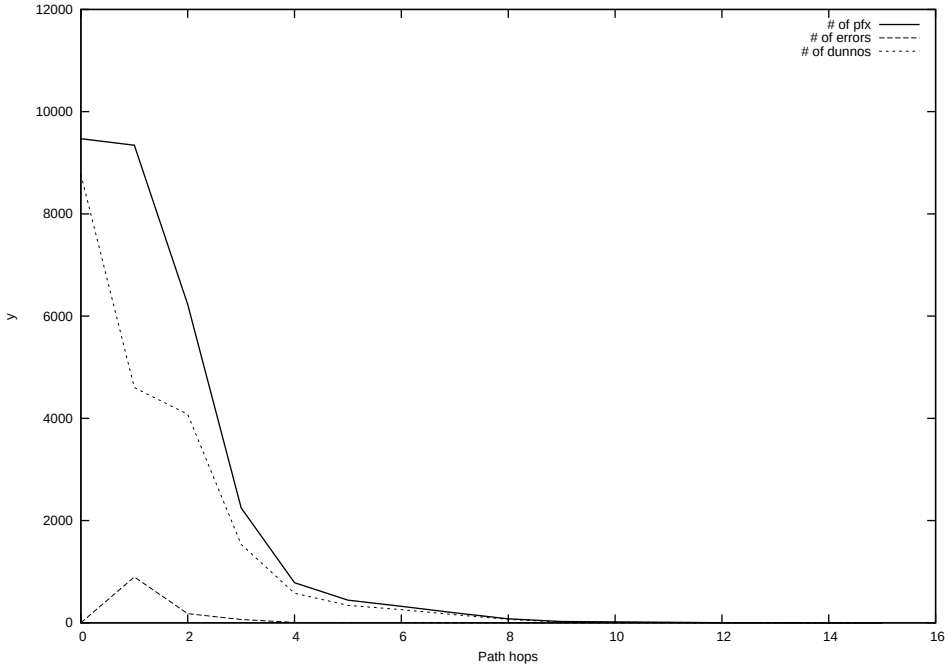
2012-06-13



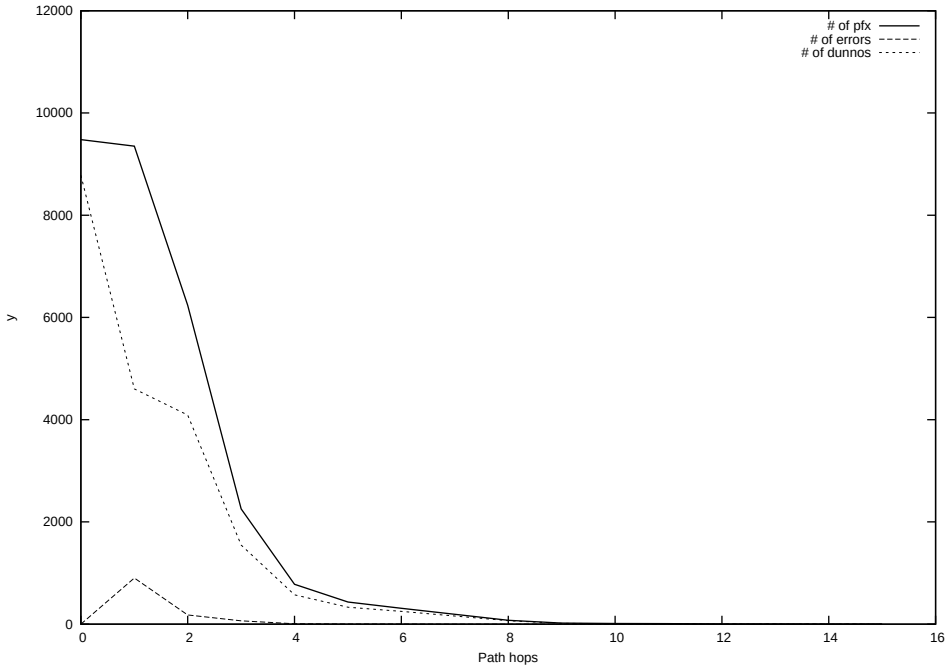
2012-06-14



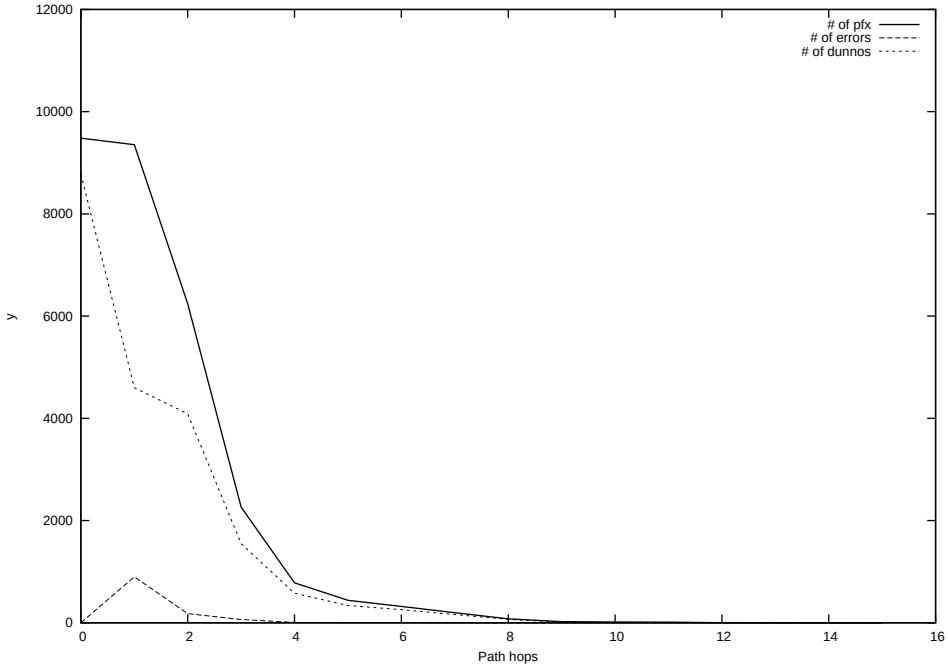
2012-06-15



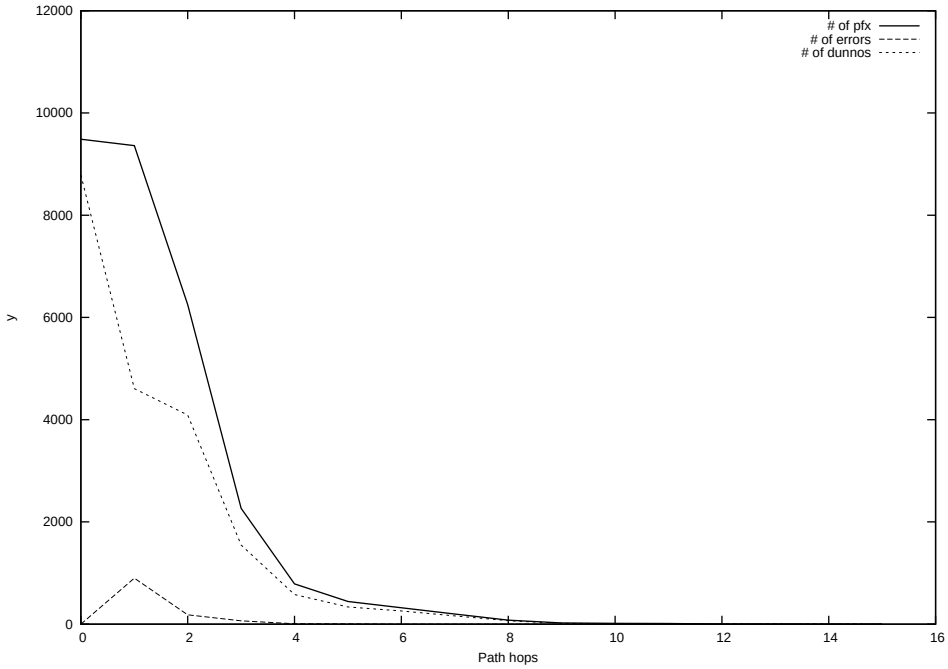
2012-06-16



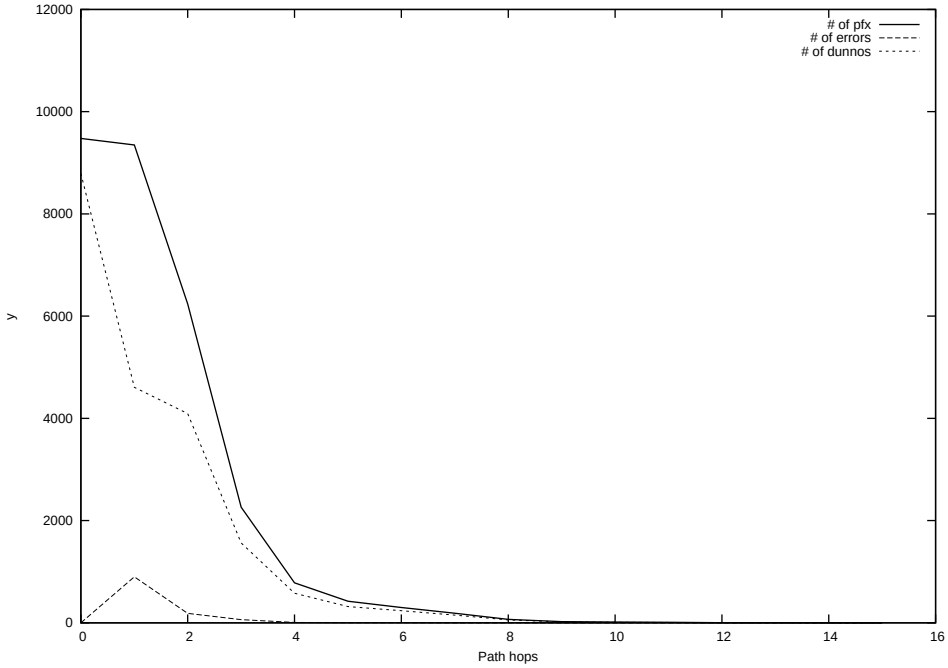
2012-06-17



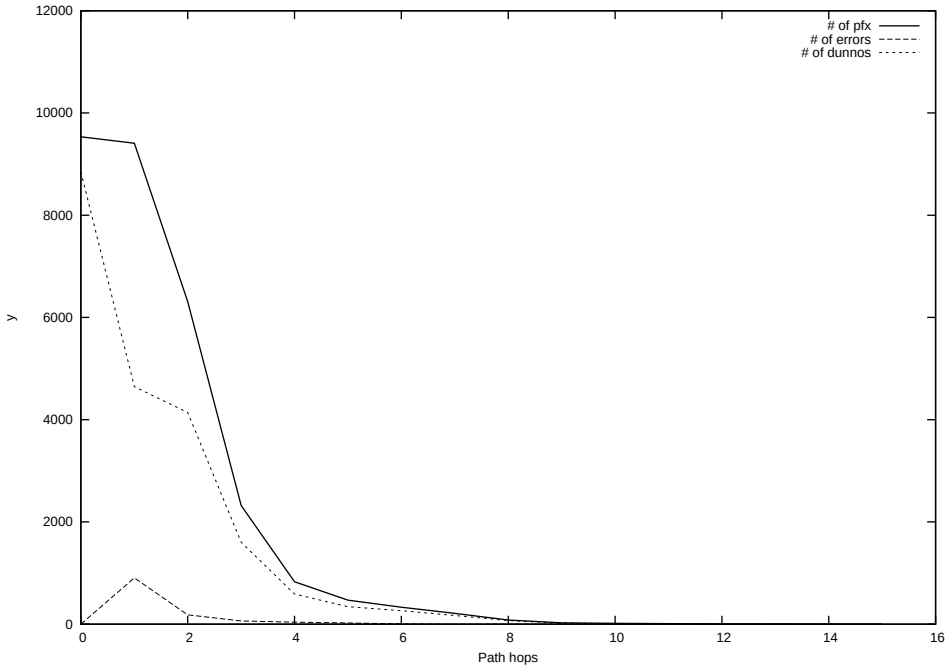
2012-06-18



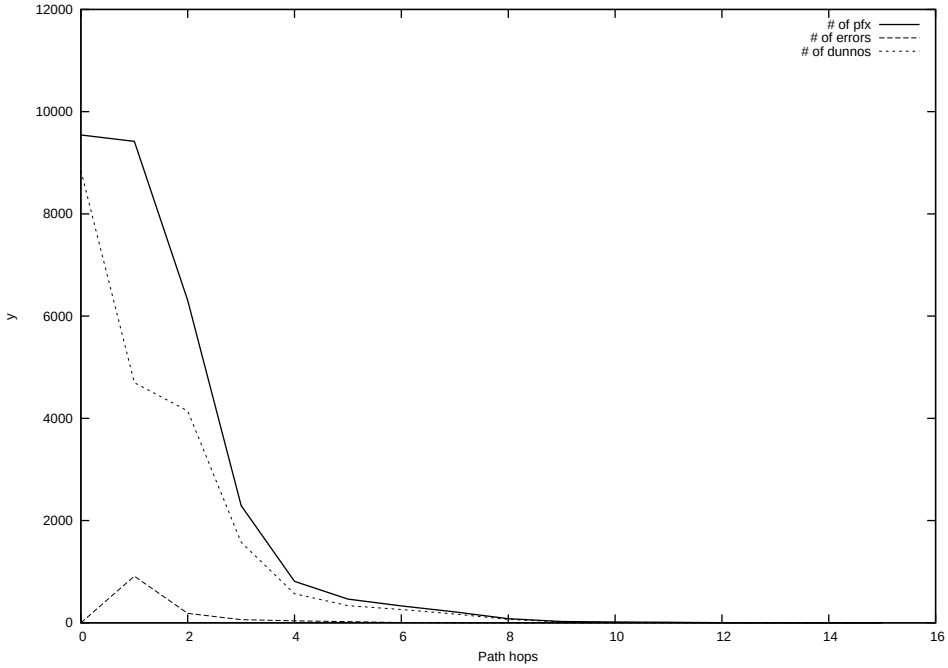
2012-06-19



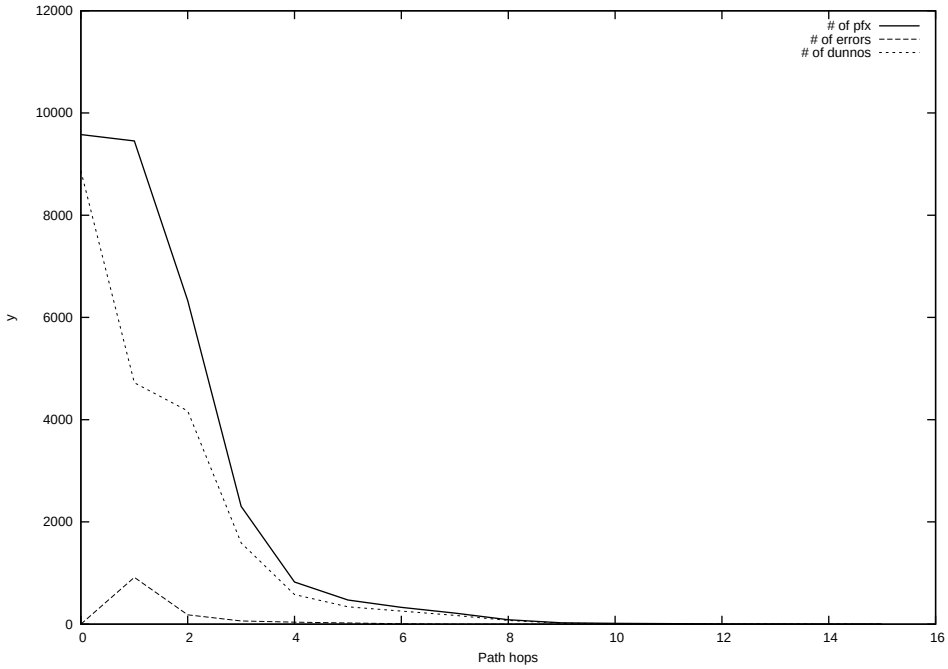
2012-06-20



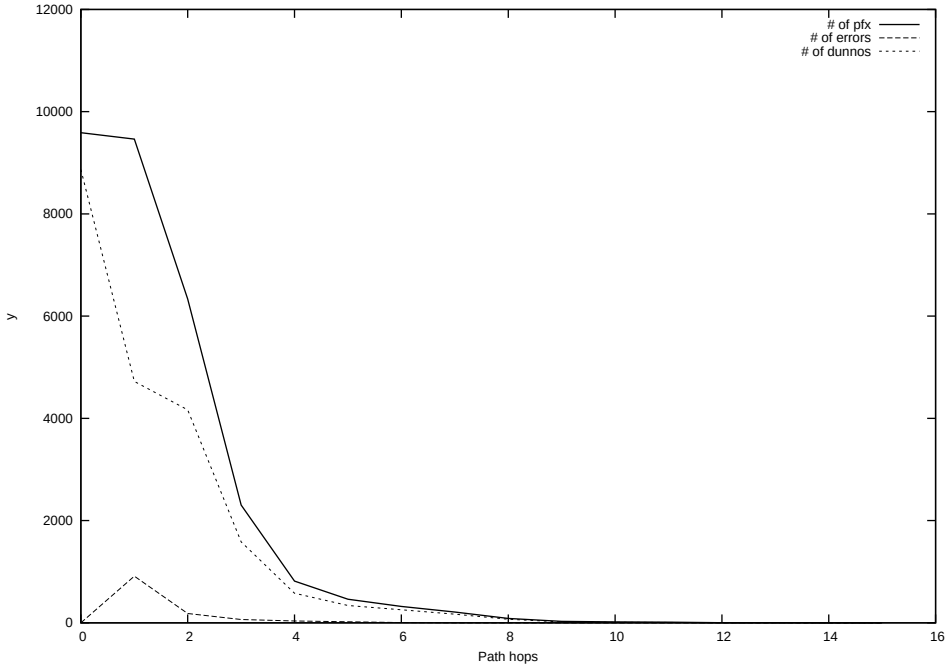
2012-06-21



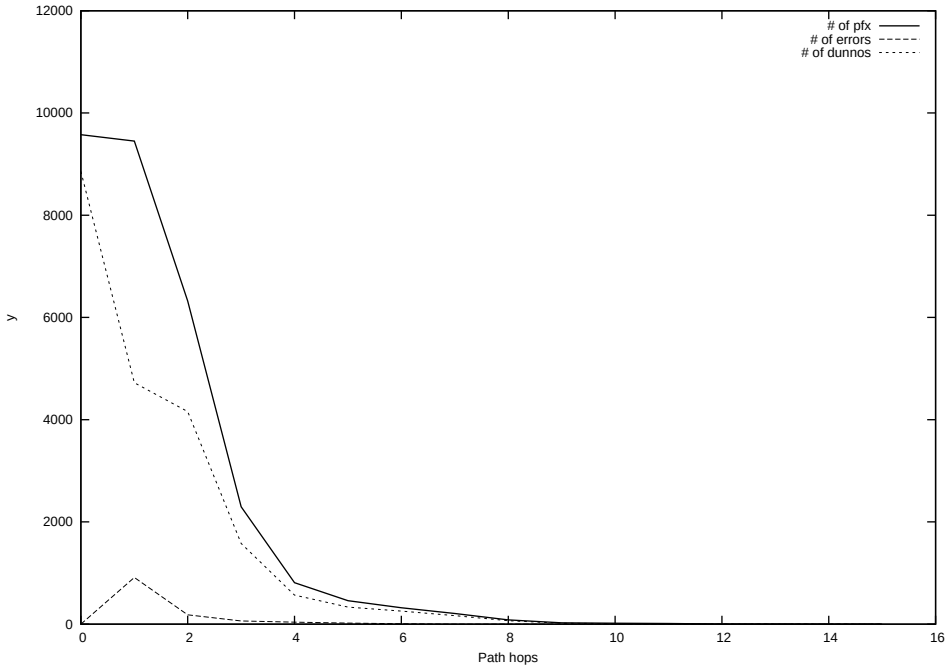
2012-06-22



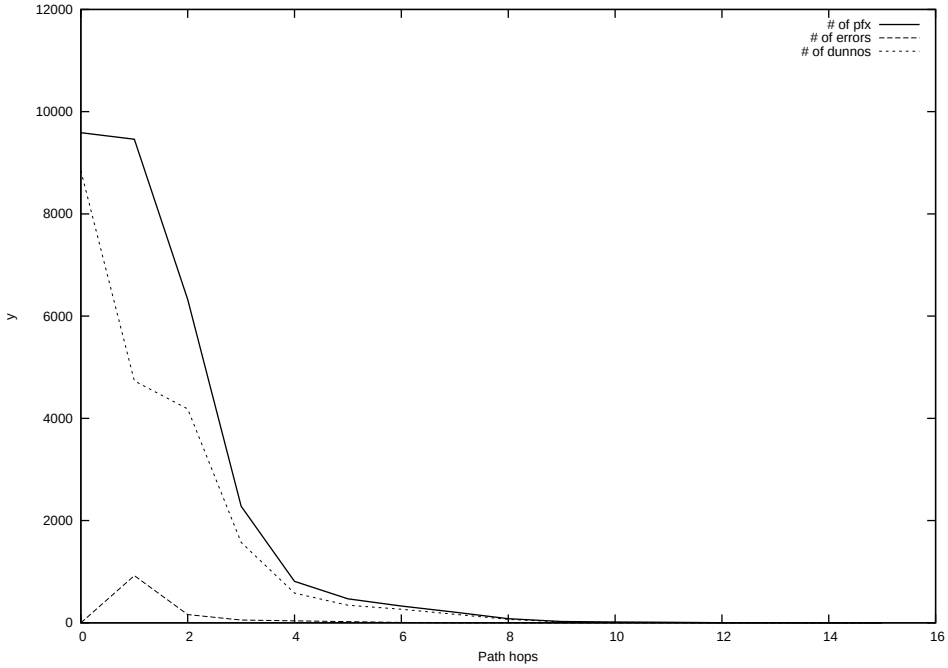
2012-06-23



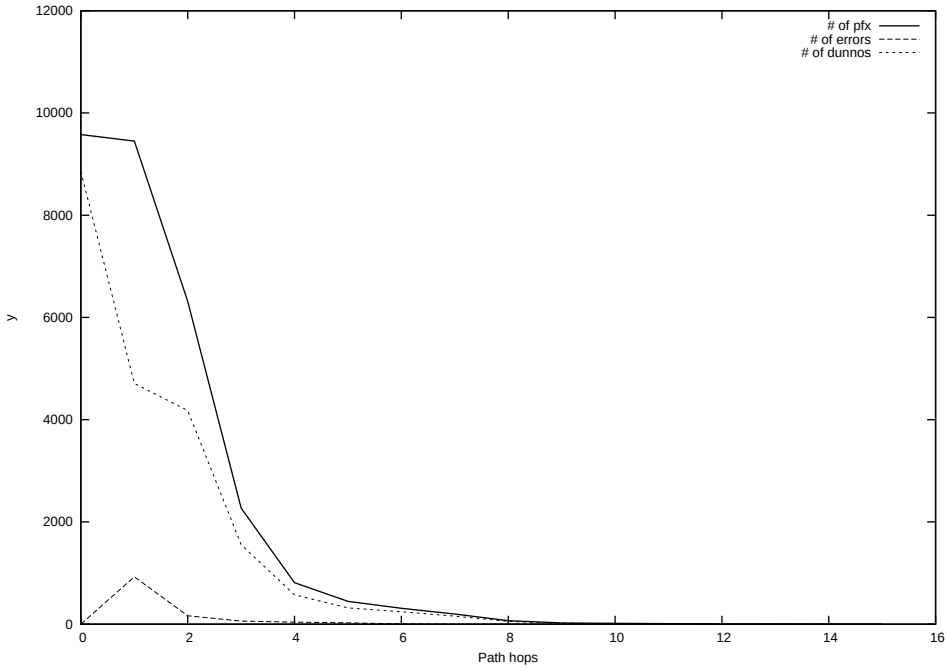
2012-06-24



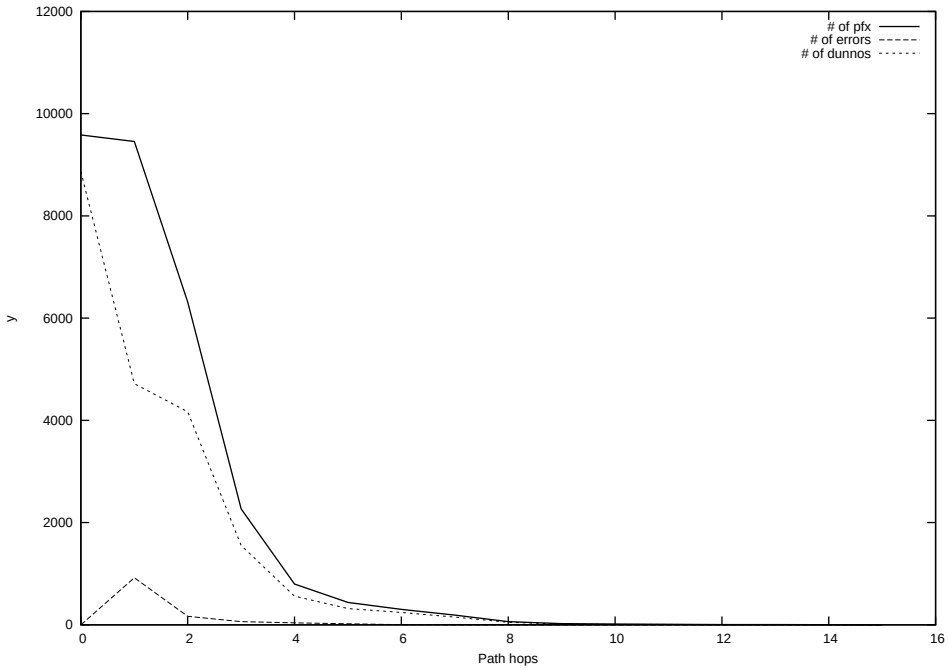
2012-06-25



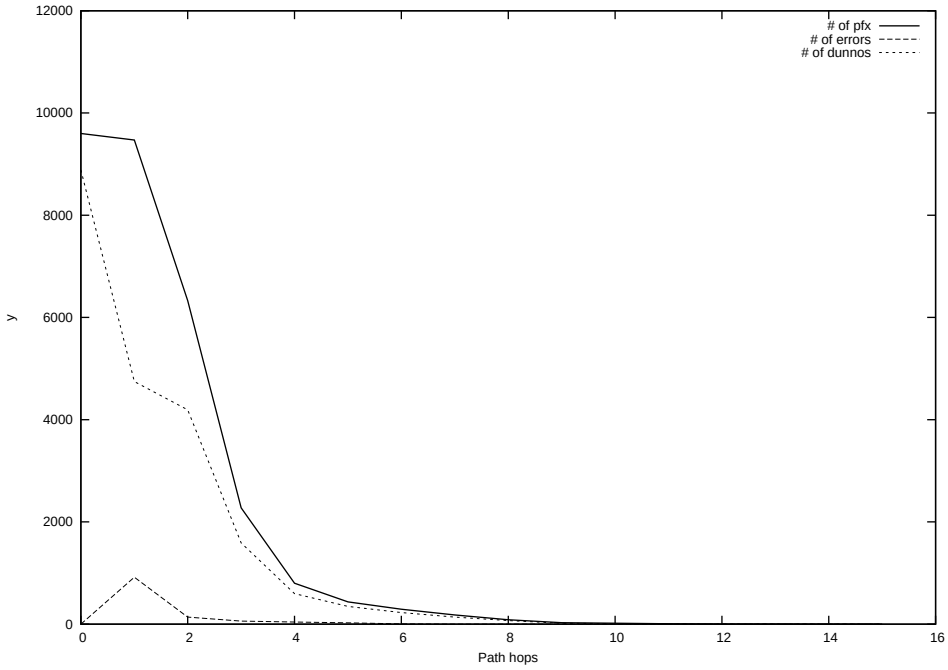
2012-06-26



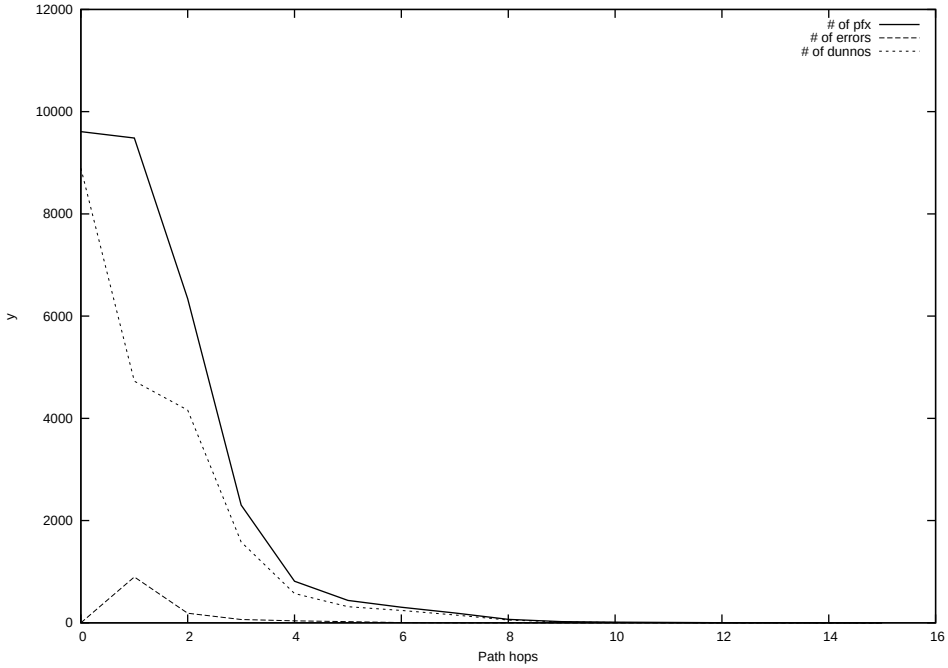
2012-06-27



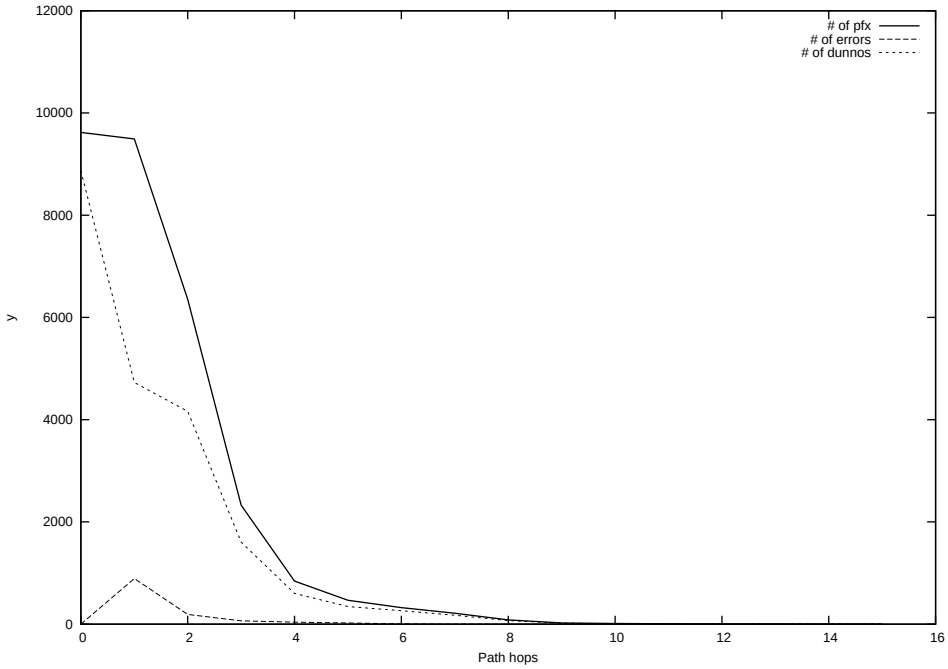
2012-06-28



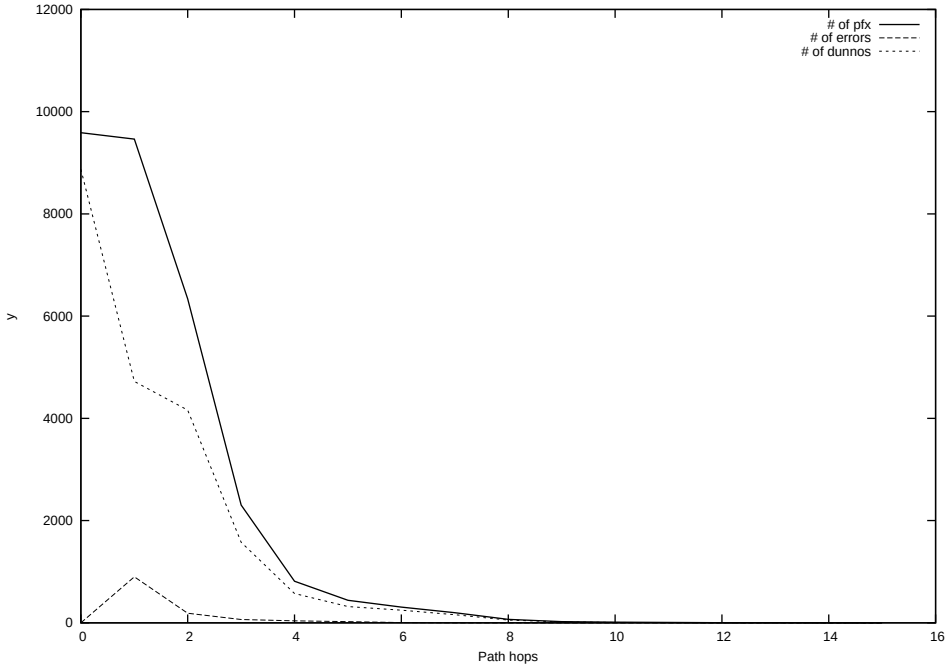
2012-06-29



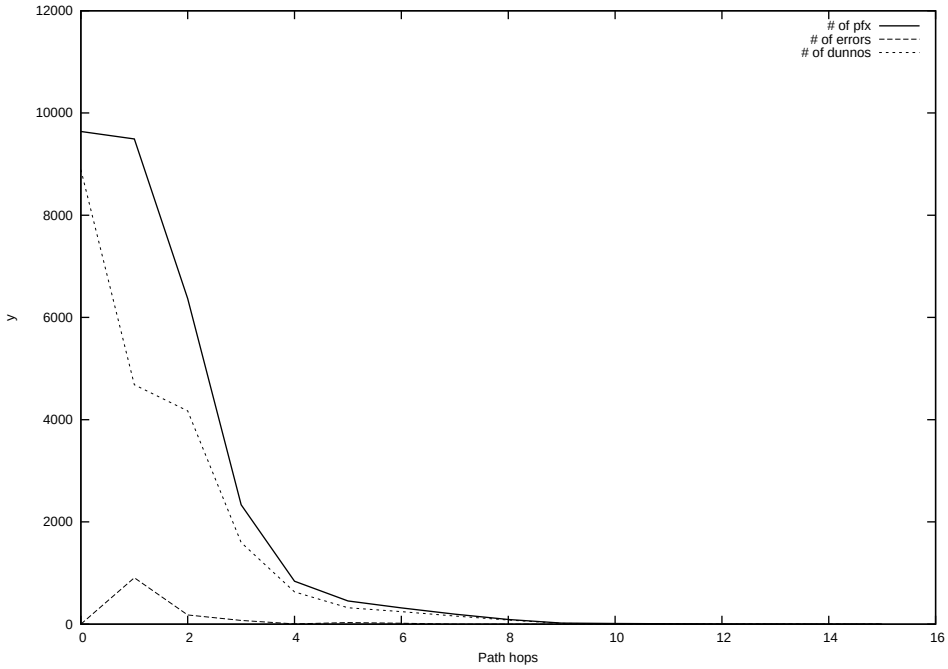
2012-06-30



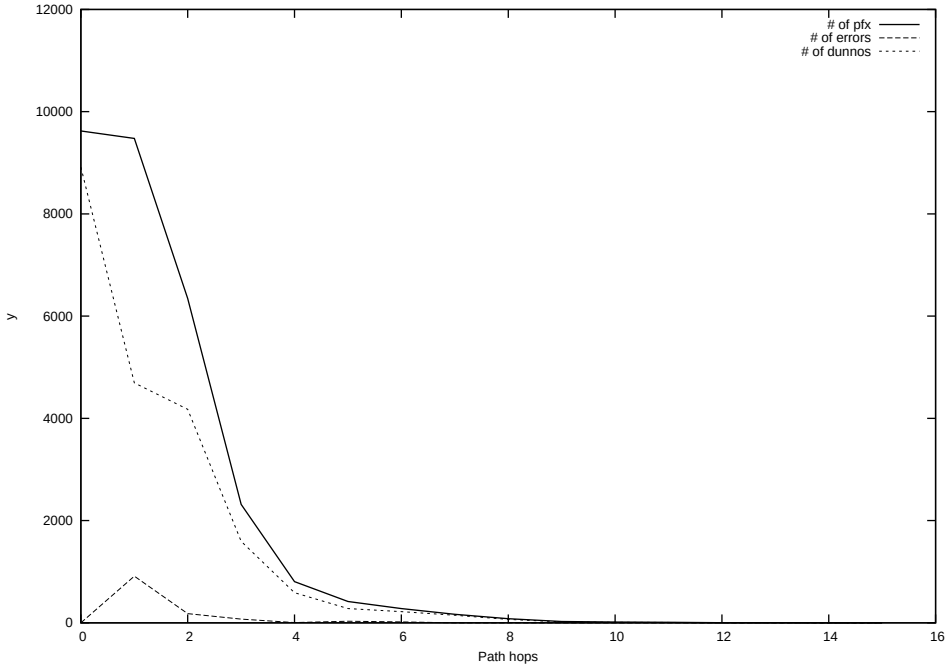
2012-07-01



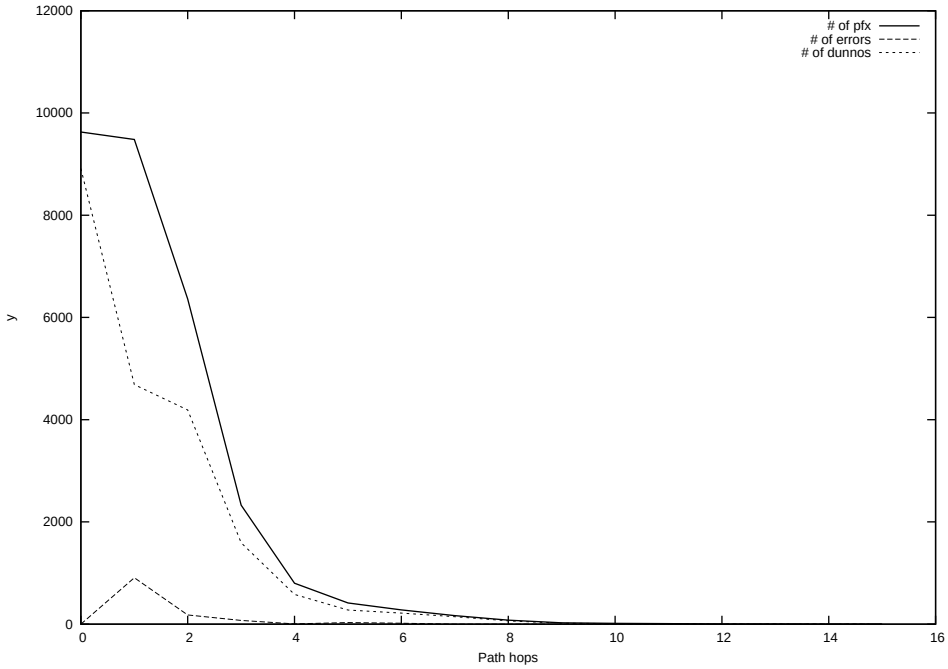
2012-07-02



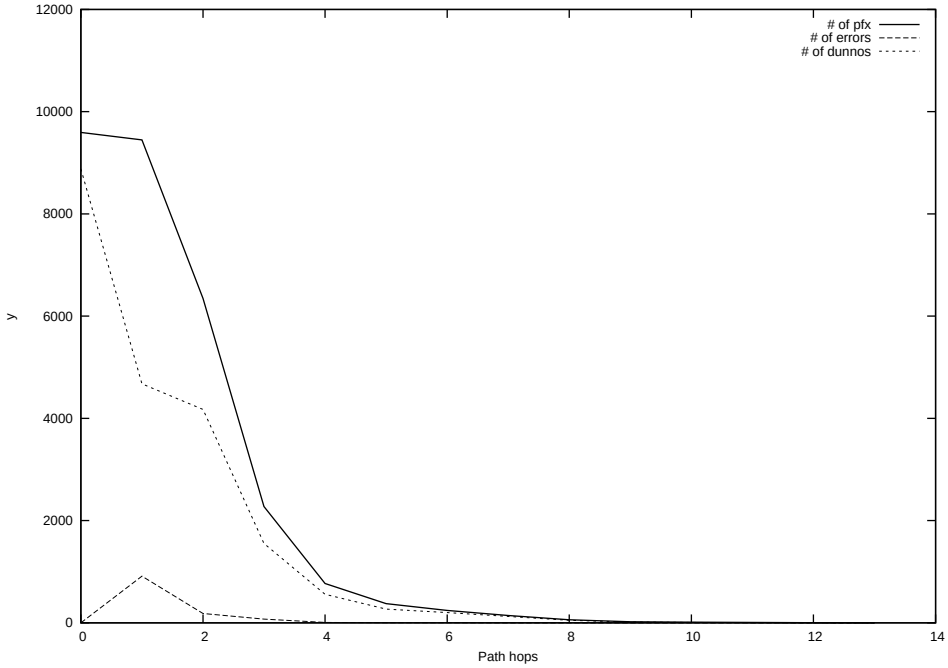
2012-07-03



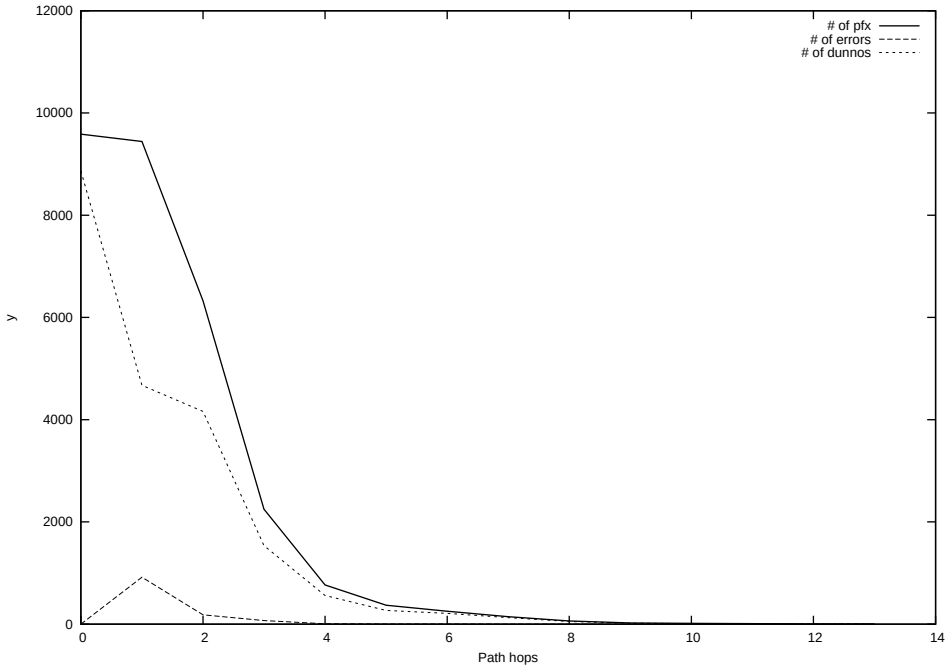
2012-07-04



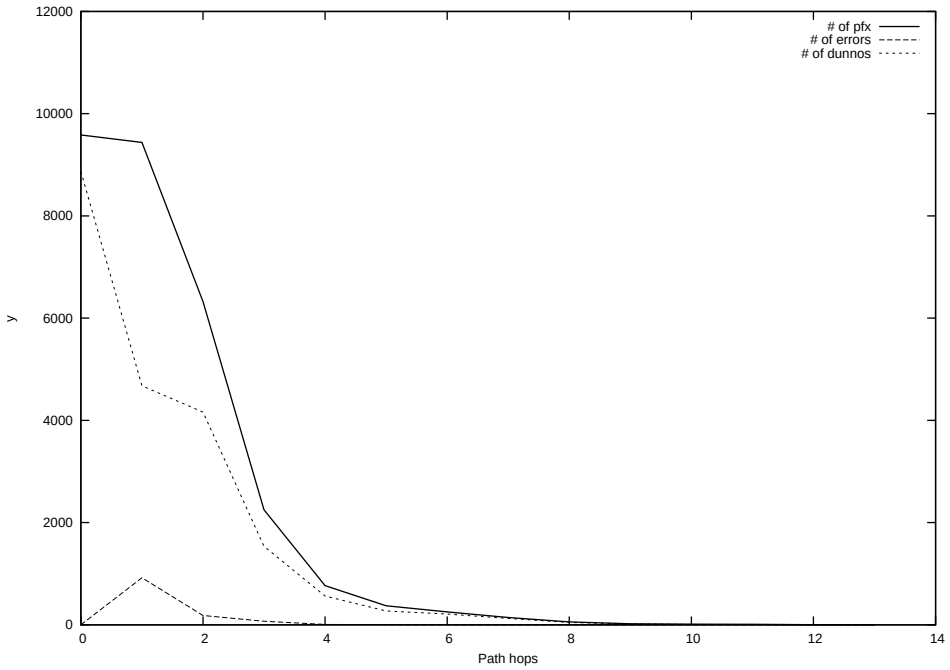
2012-07-05



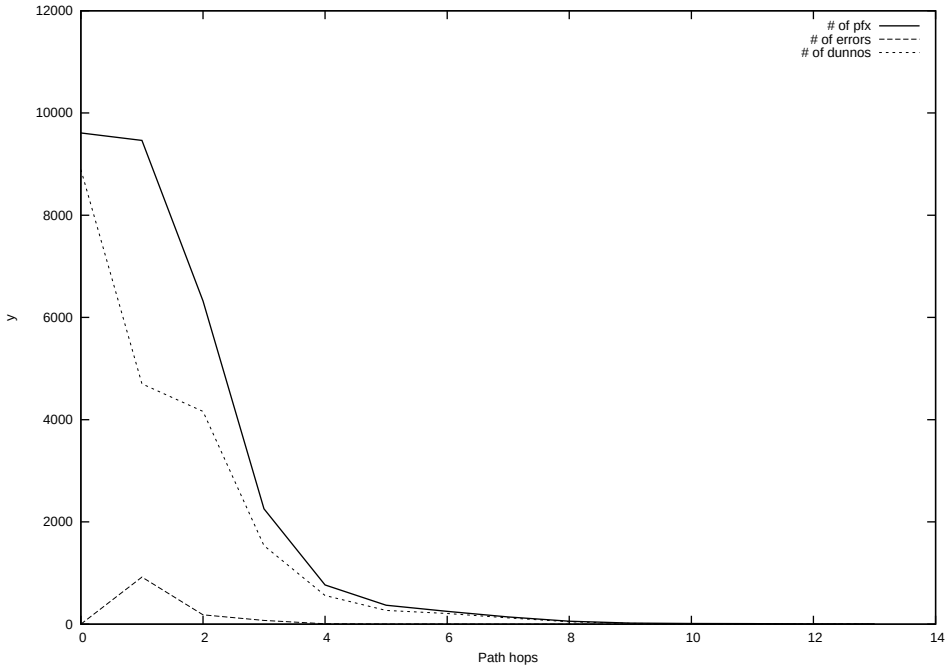
2012-07-06



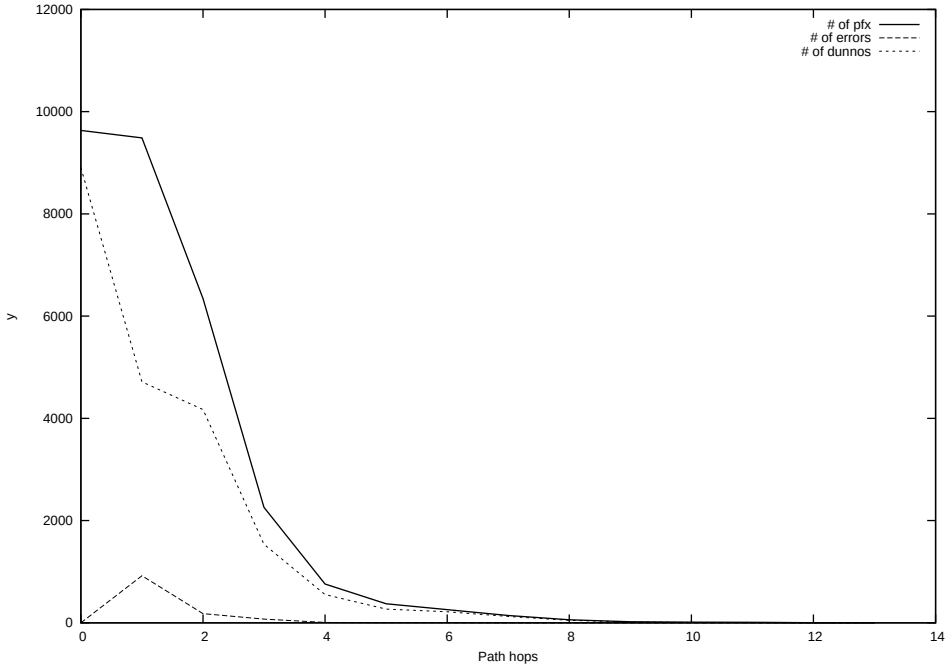
2012-07-07



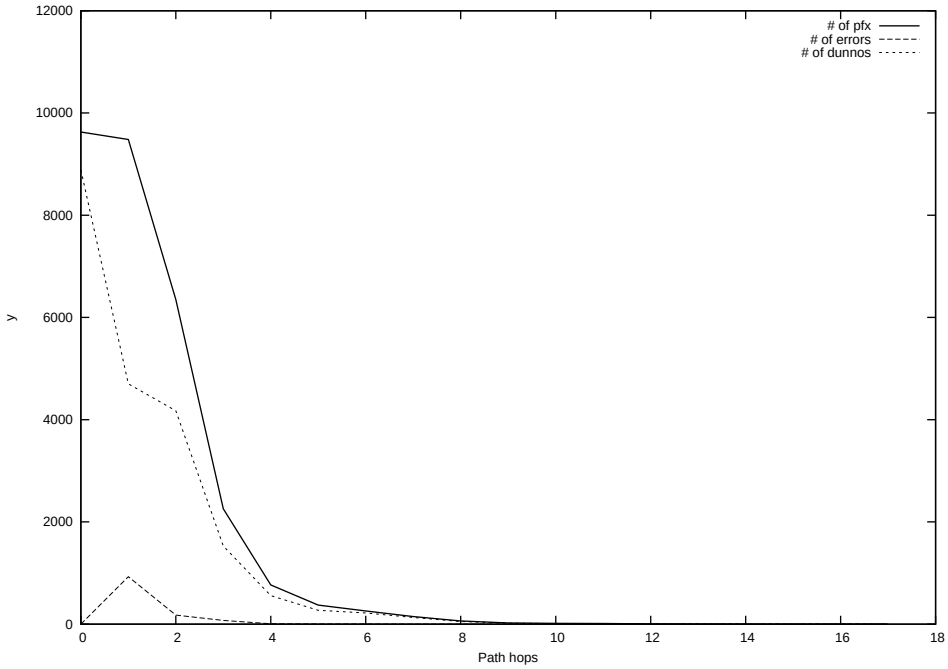
2012-07-08



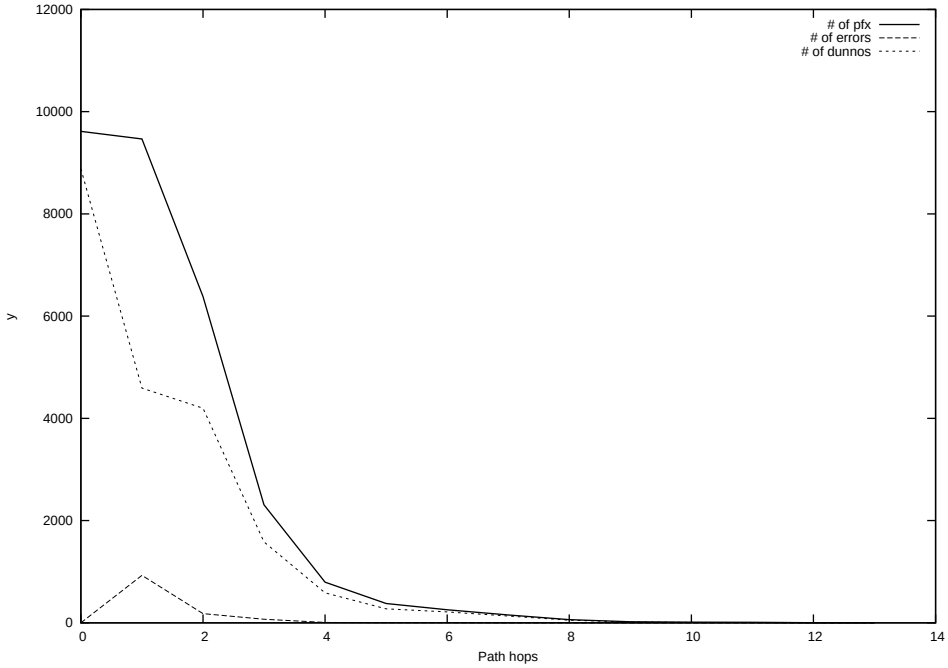
2012-07-09



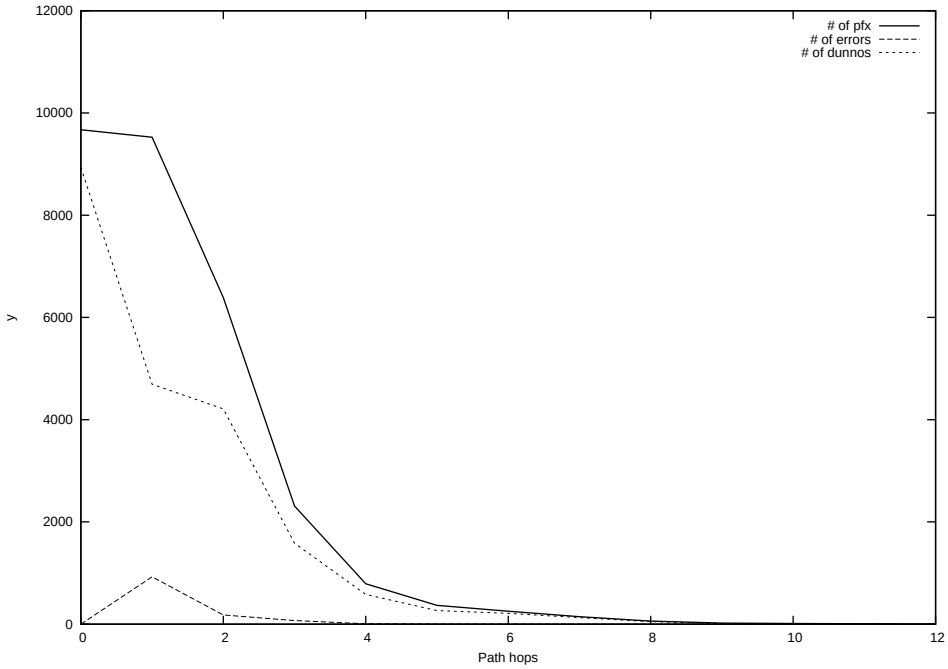
2012-07-10



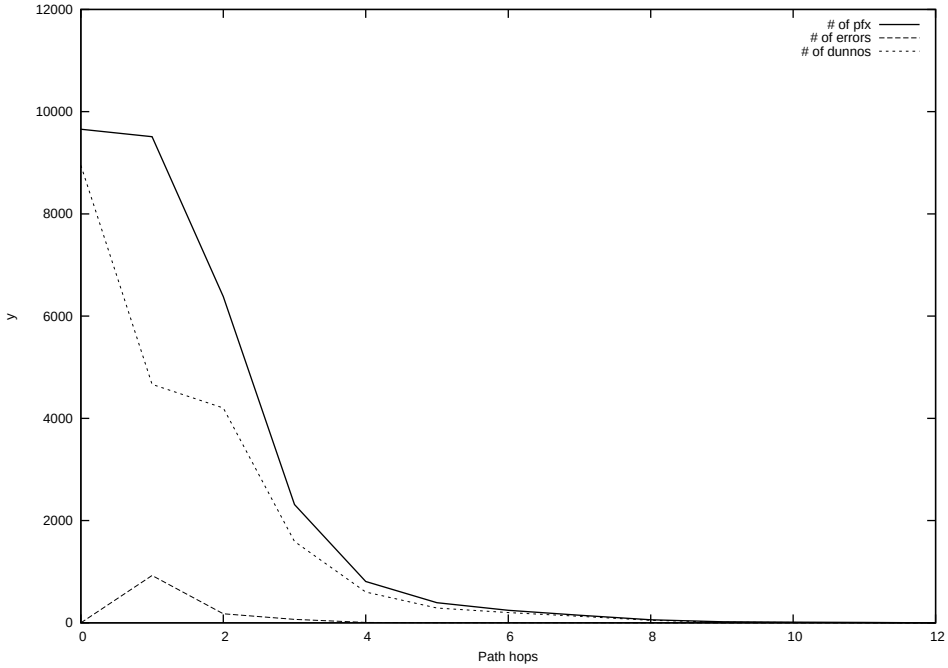
2012-07-11



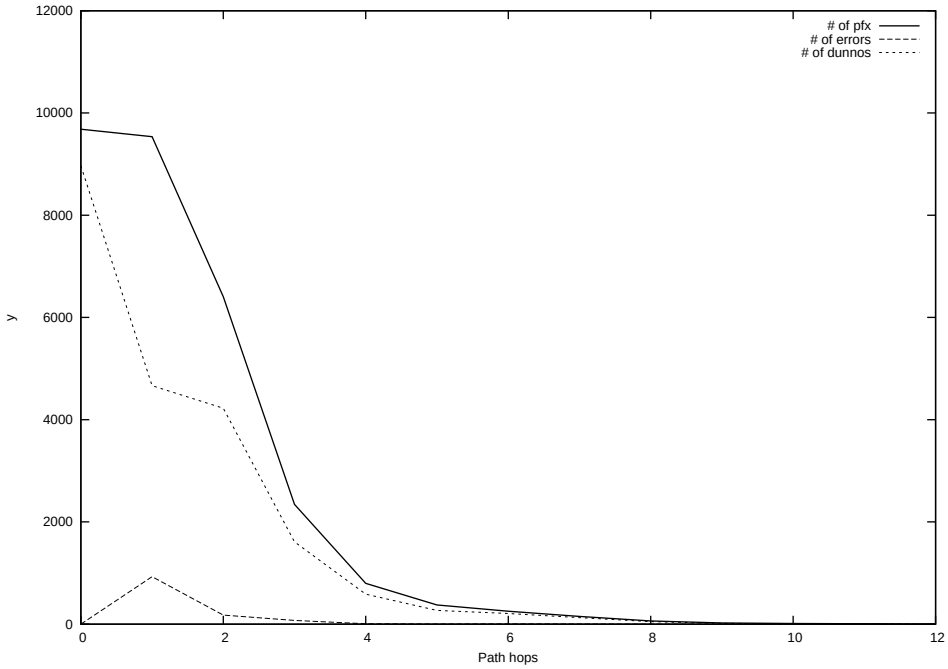
2012-07-12



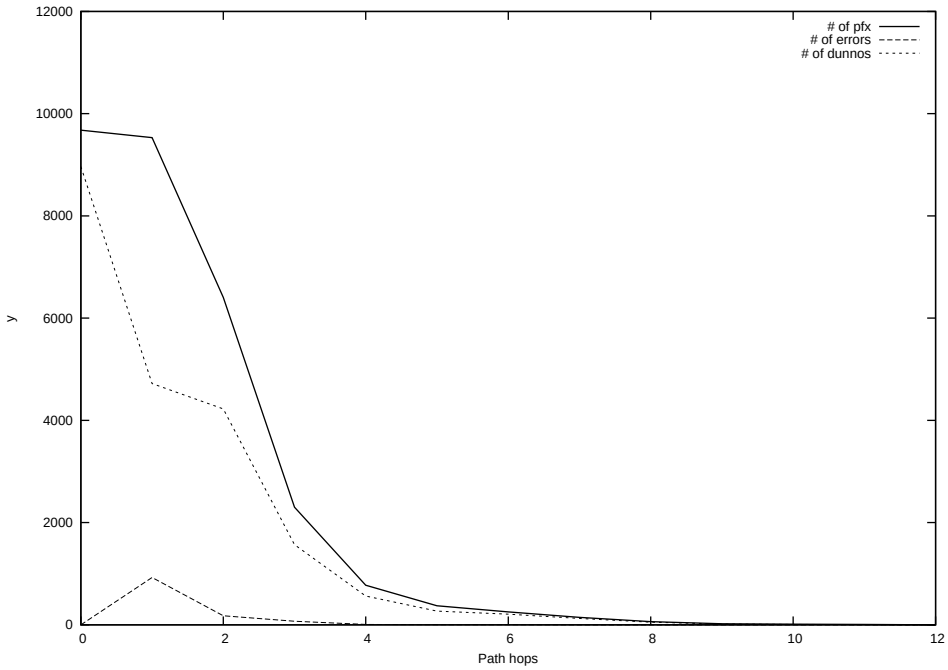
2012-07-13



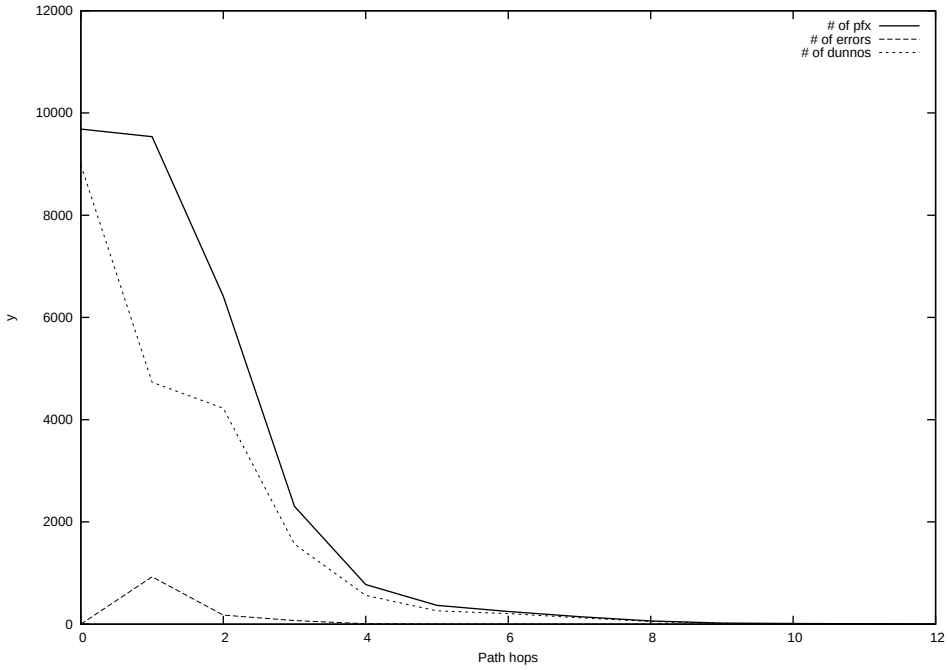
2012-07-14



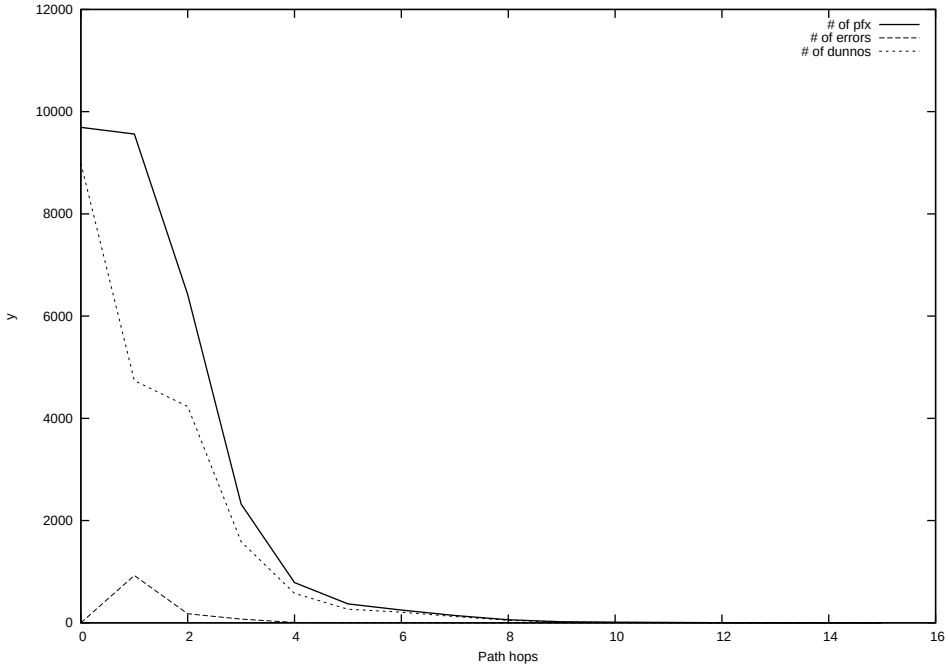
2012-07-15



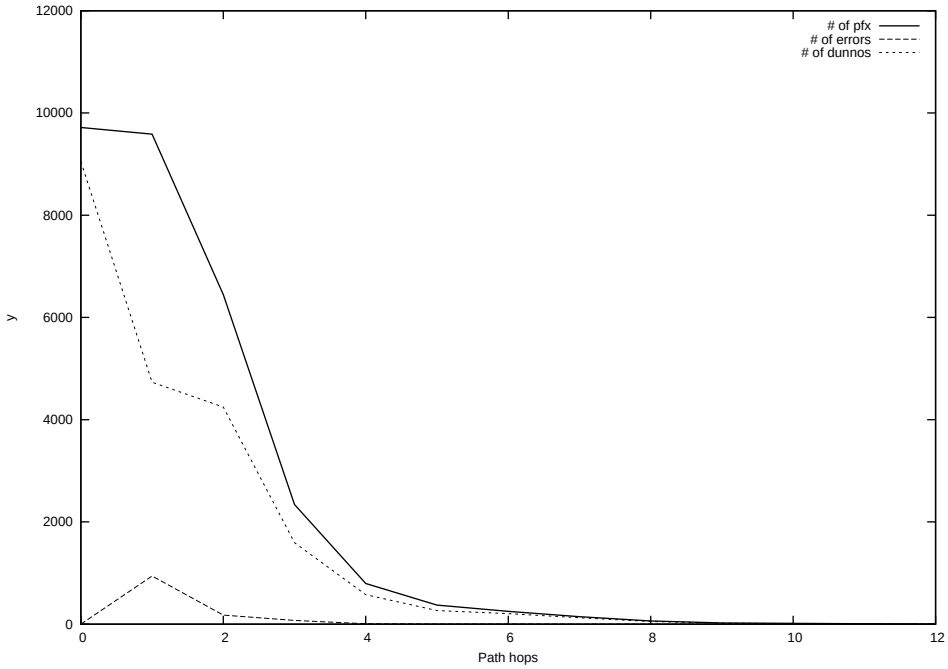
2012-07-16



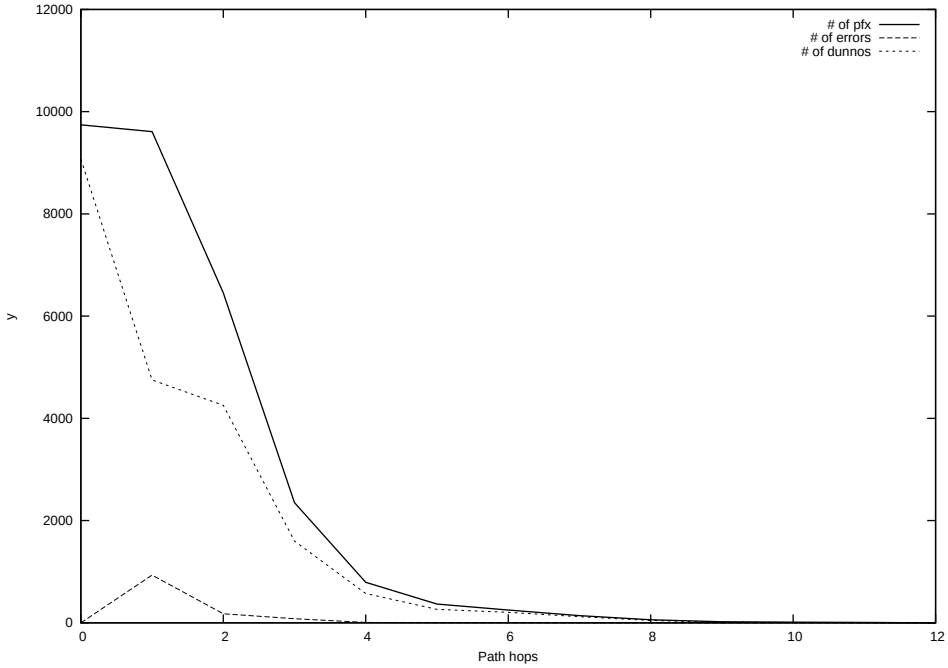
2012-07-17



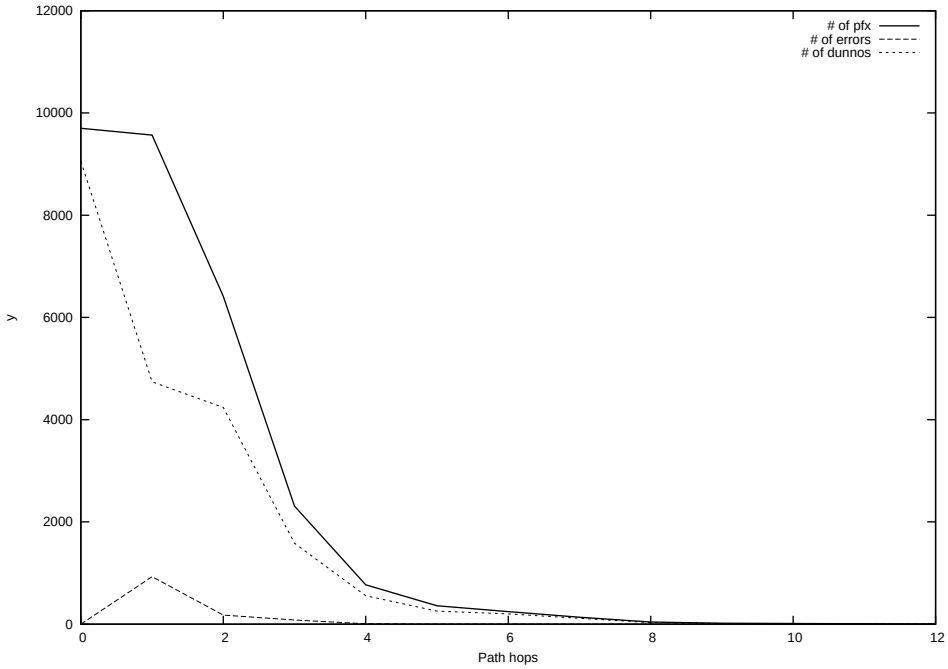
2012-07-18



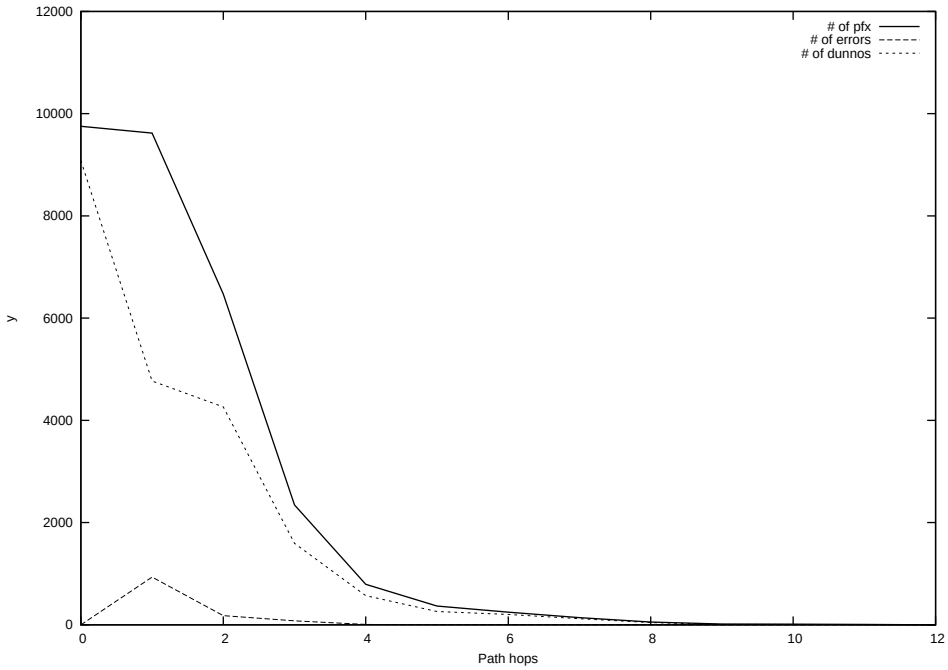
2012-07-19



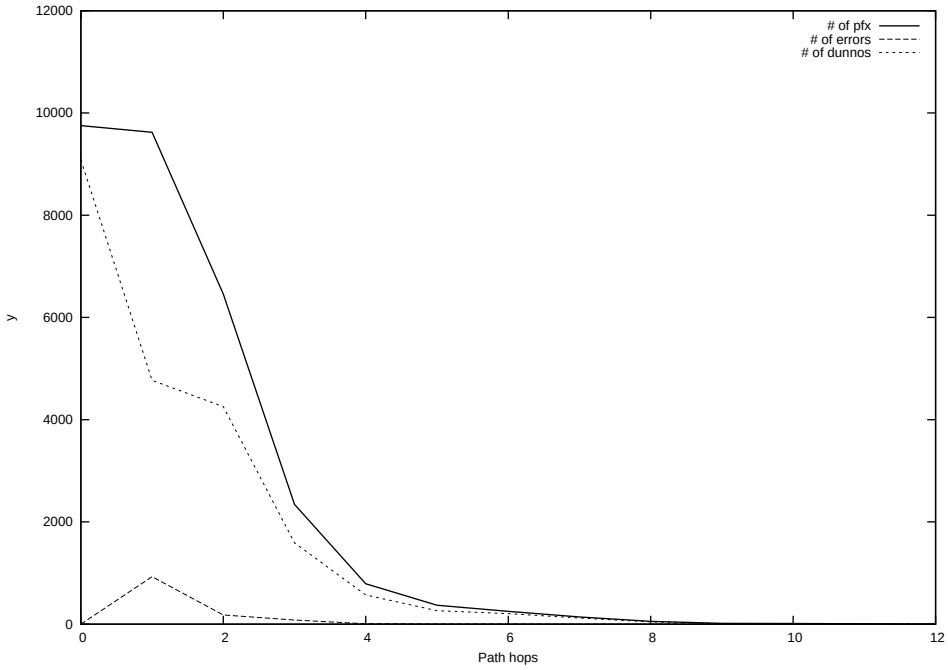
2012-07-20



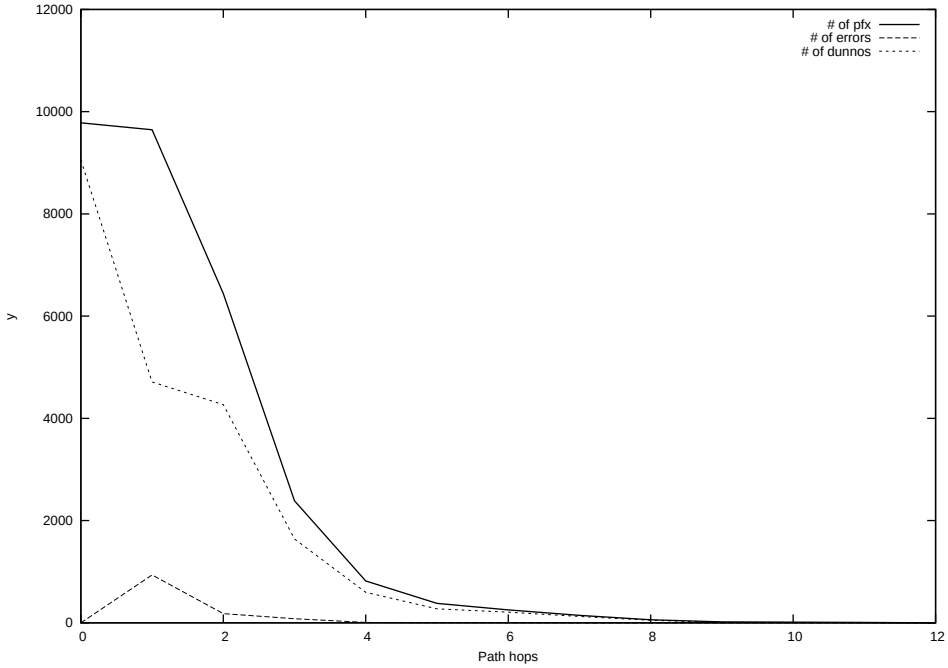
2012-07-21



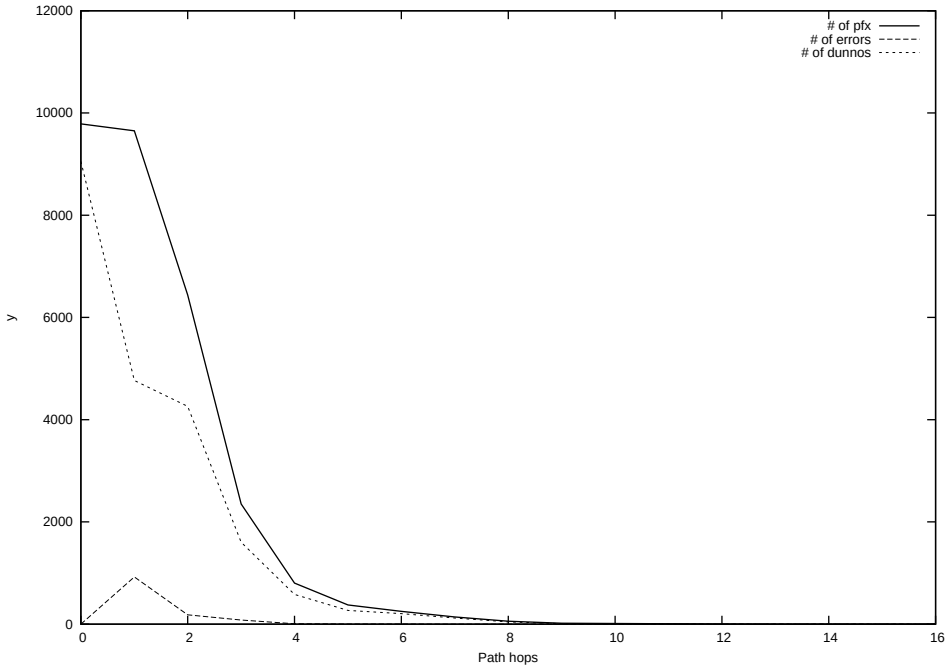
2012-07-22



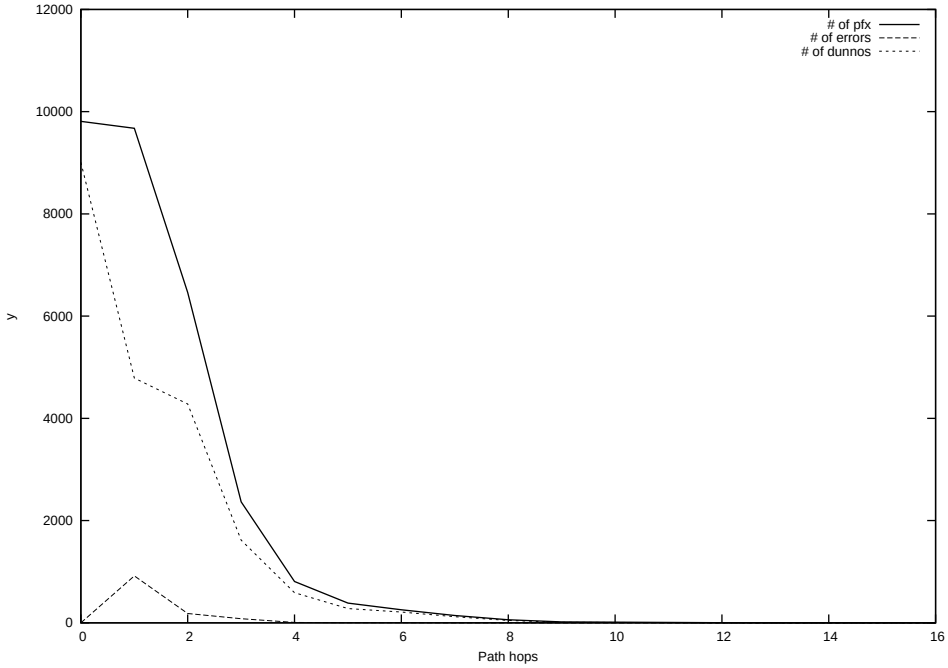
2012-07-23



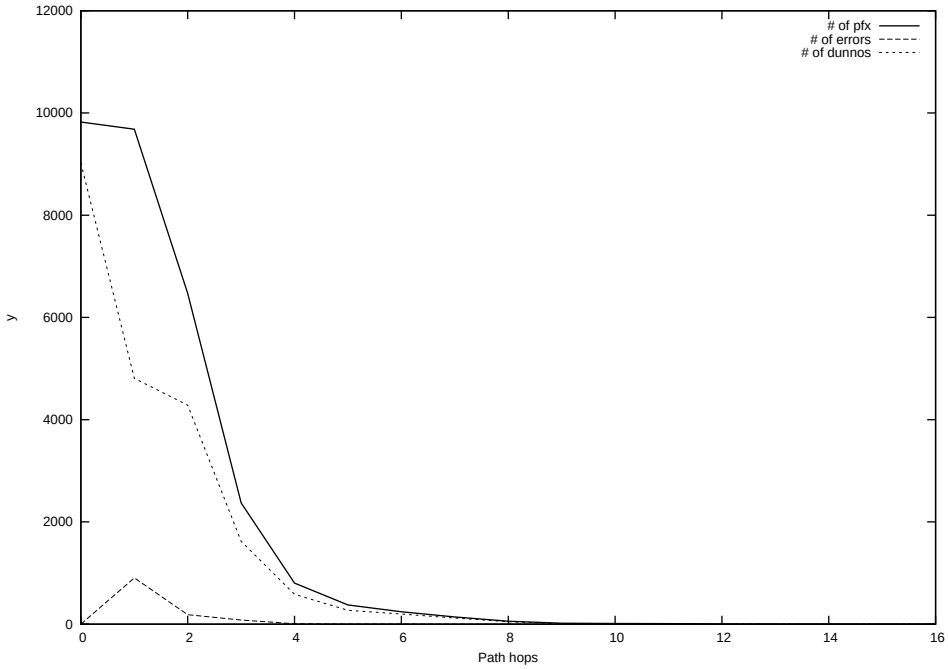
2012-07-24



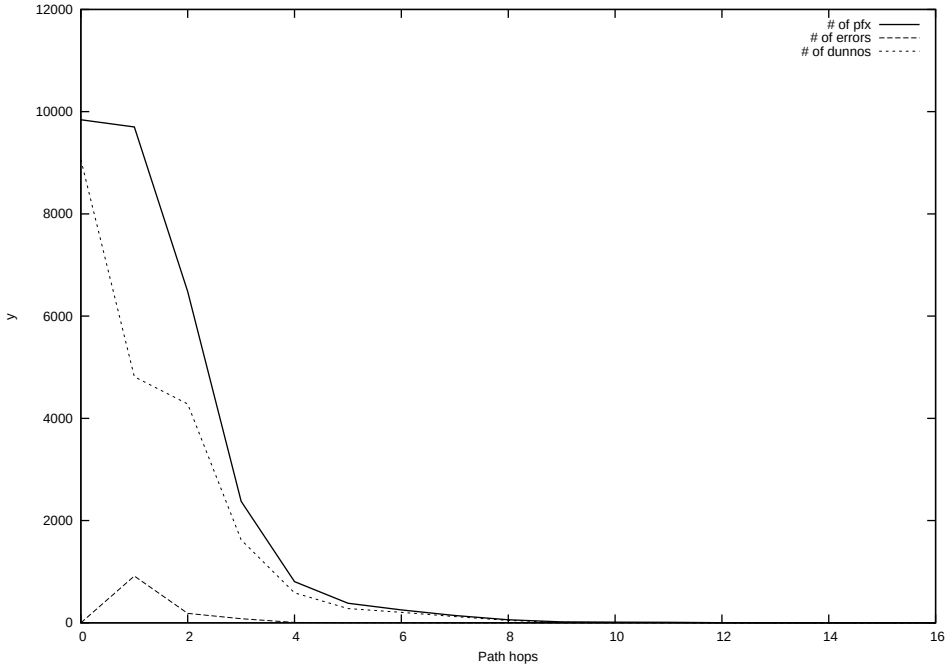
2012-07-25



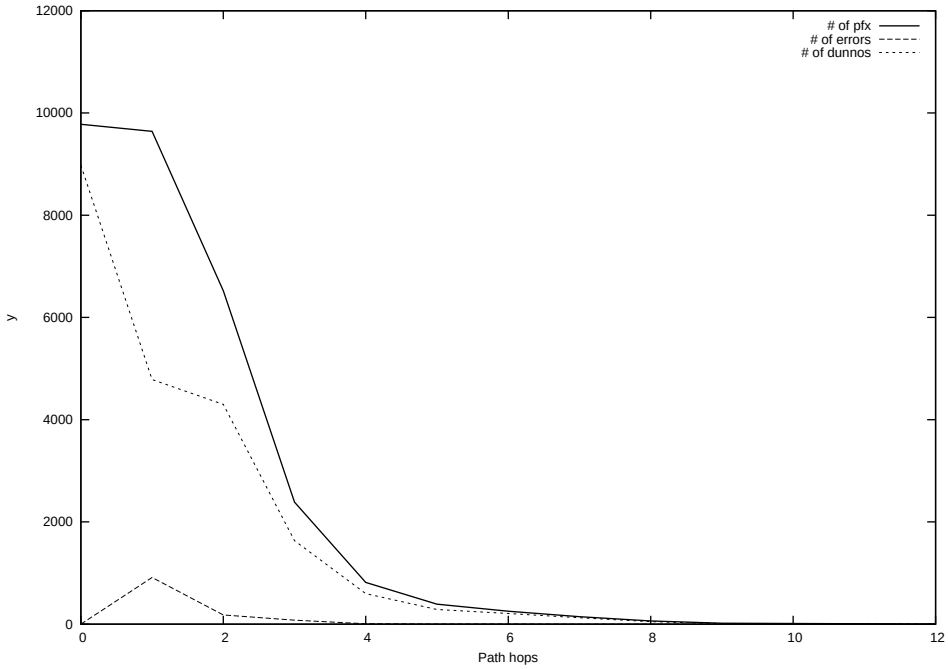
2012-07-26



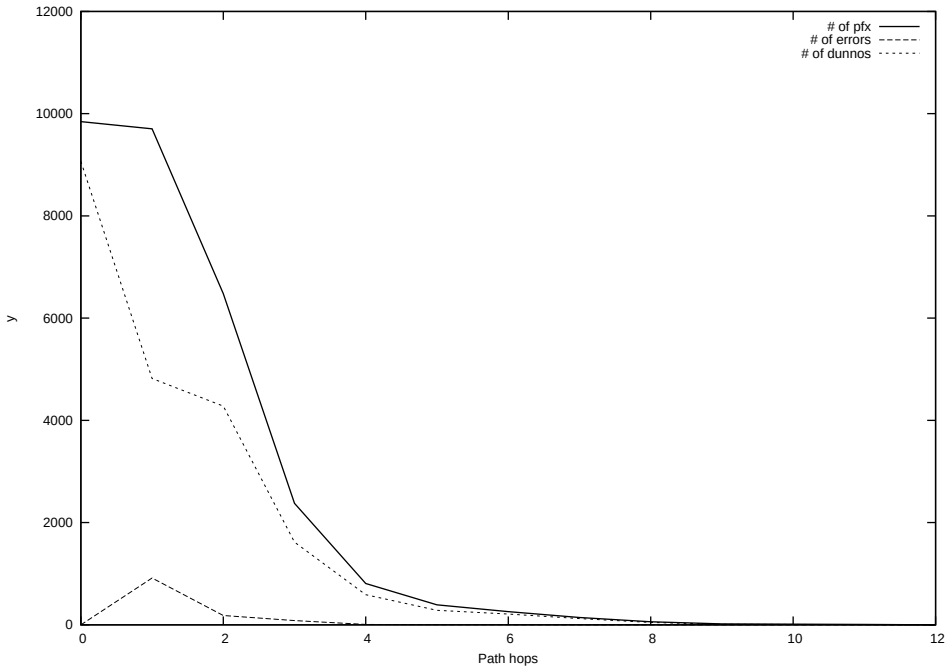
2012-07-27



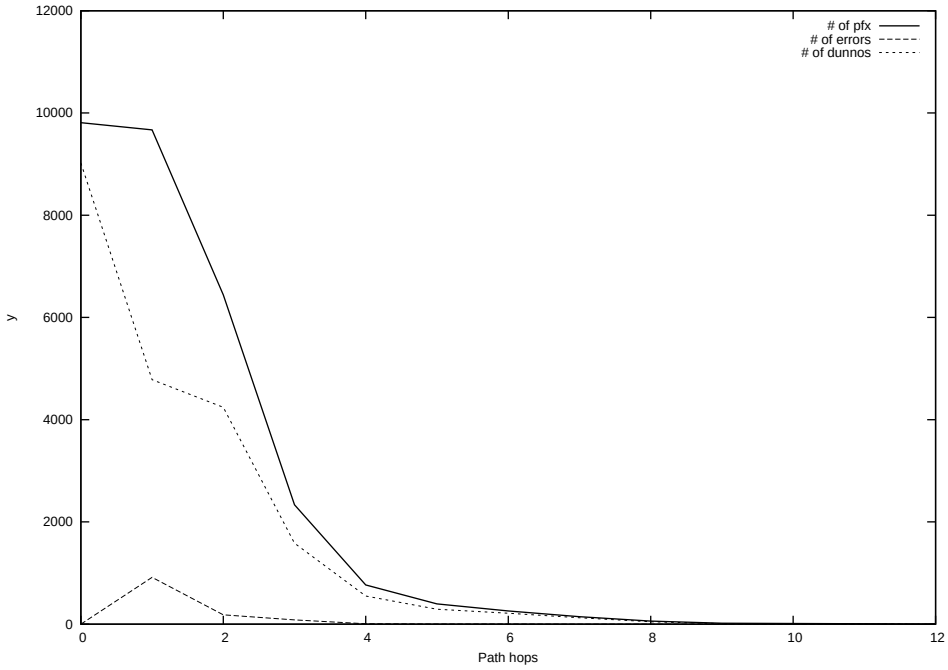
2012-07-28



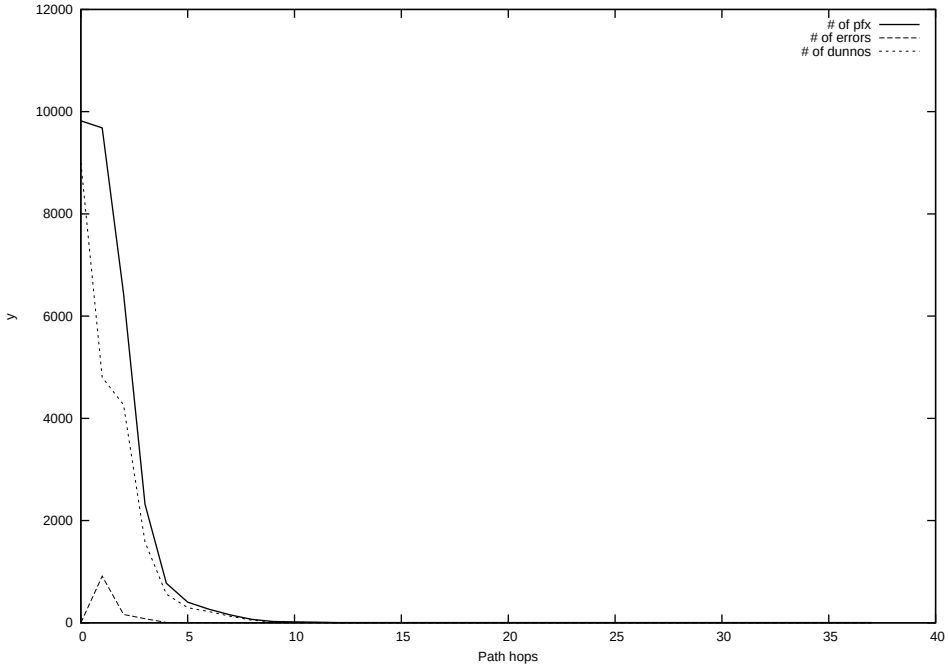
2012-07-29



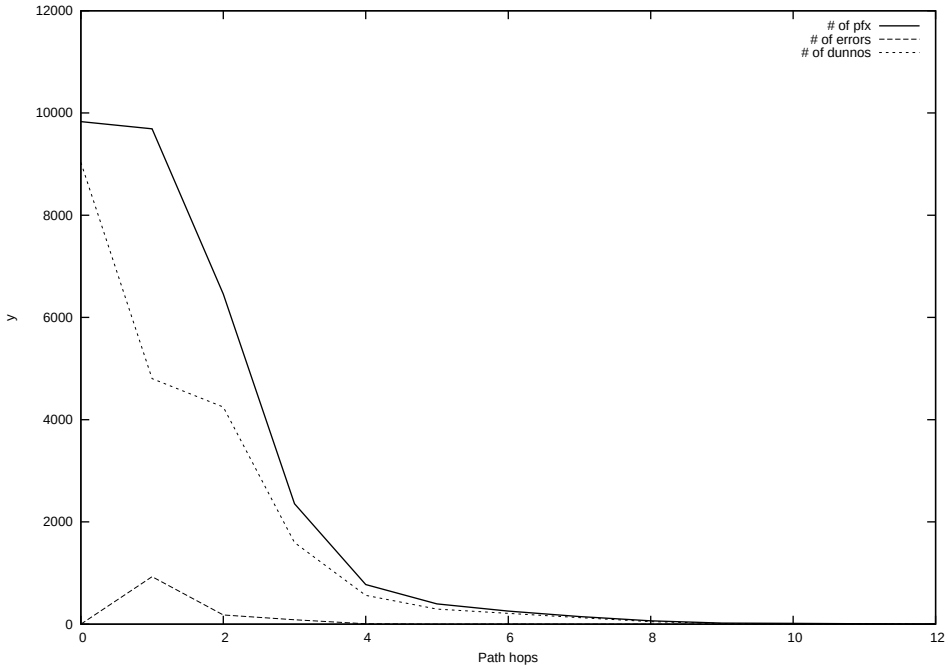
2012-07-30



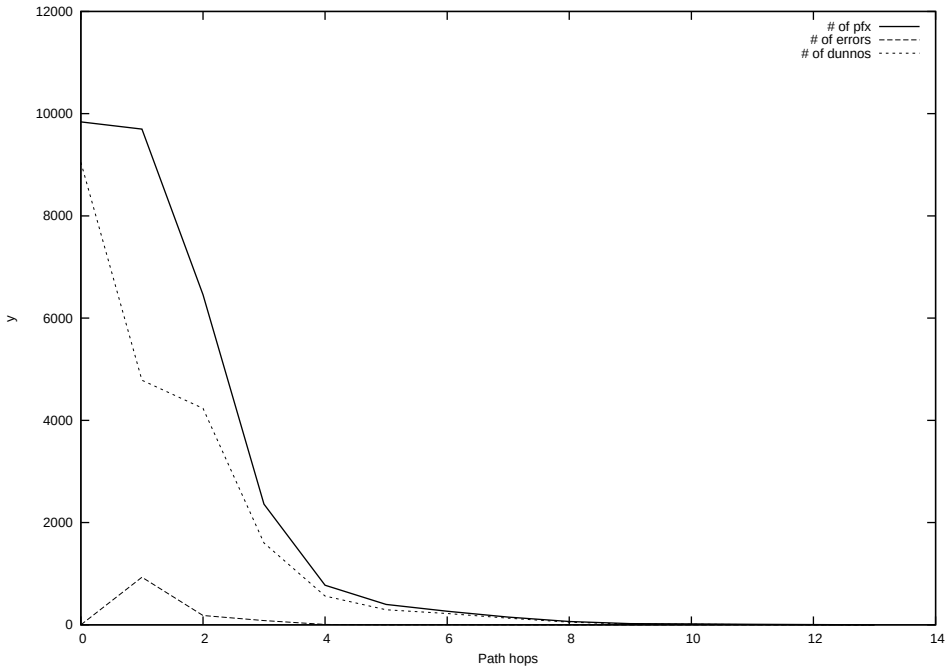
2012-07-31



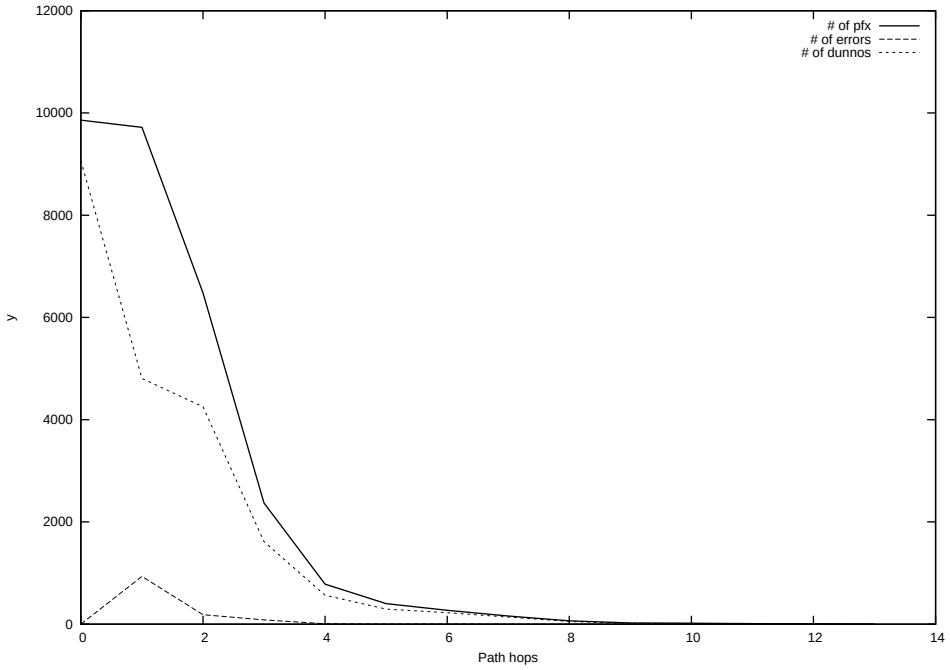
2012-08-01



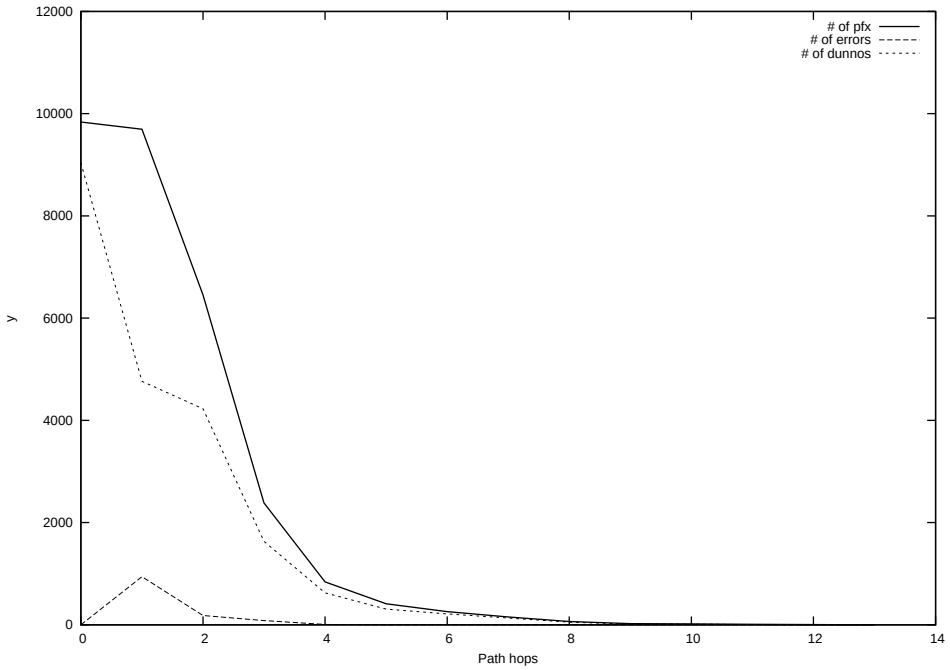
2012-08-02



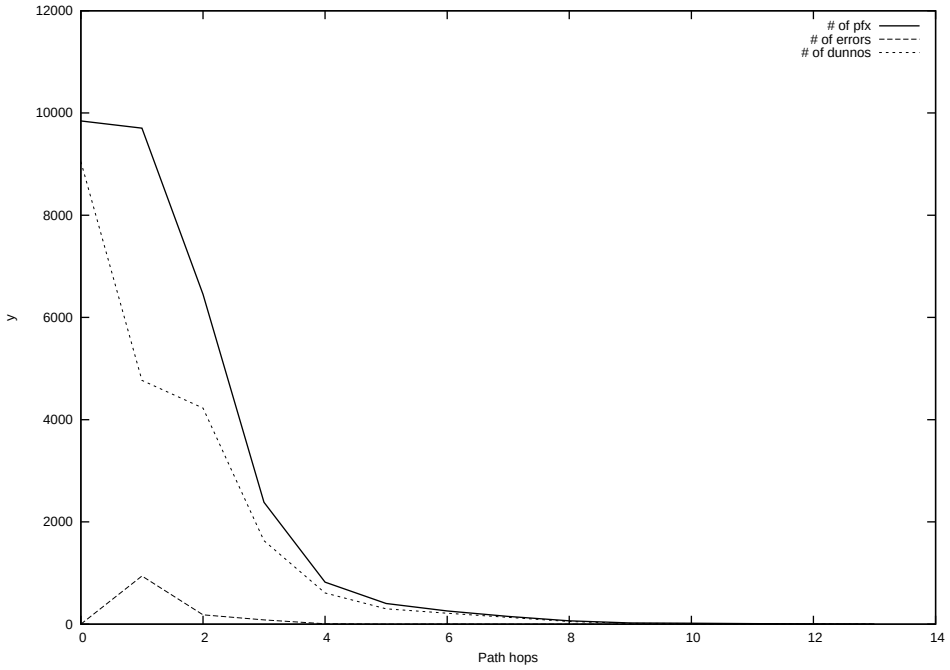
2012-08-03



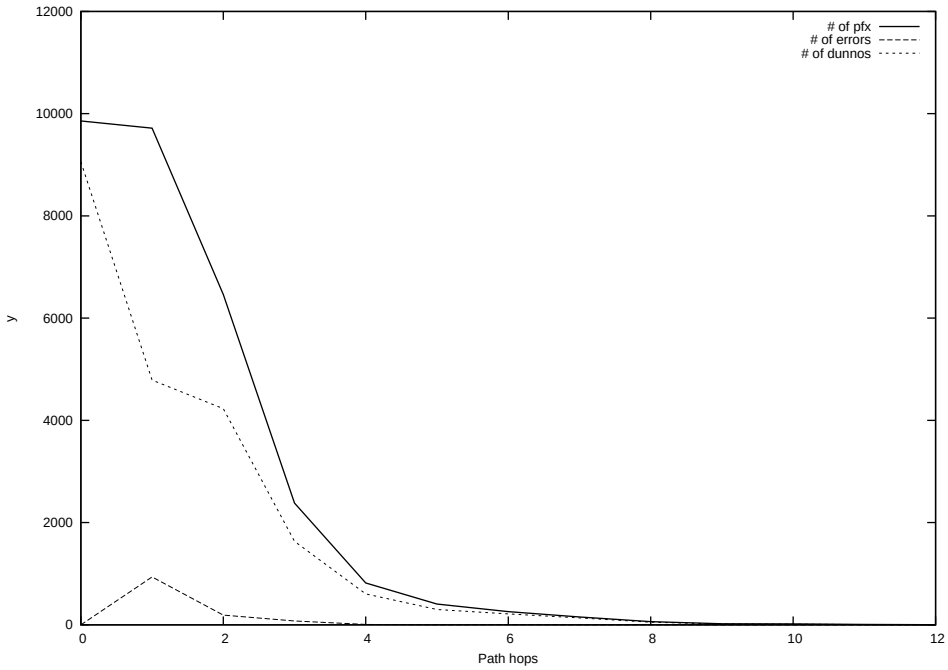
2012-08-04



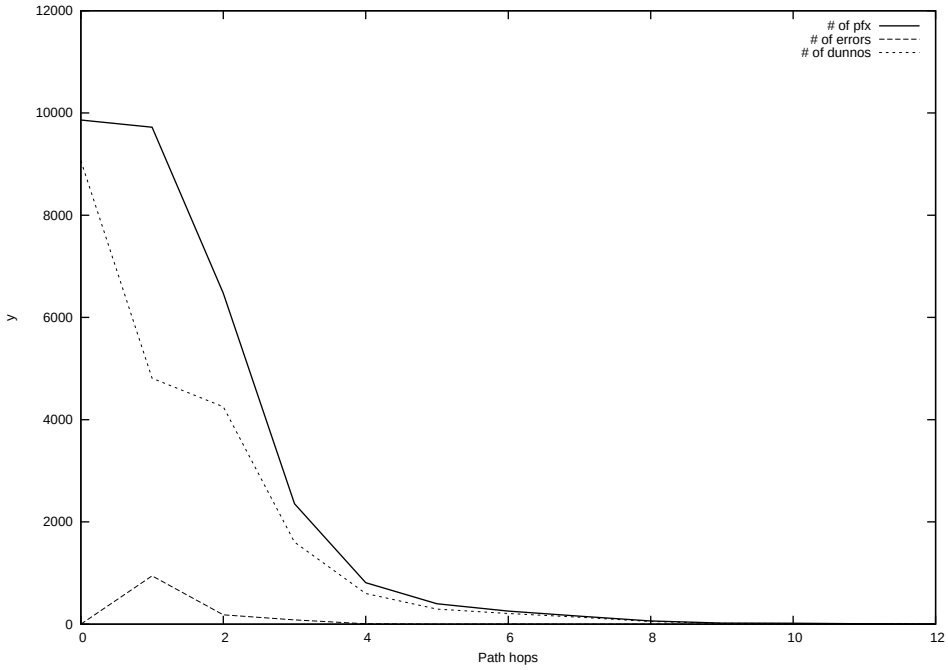
2012-08-05



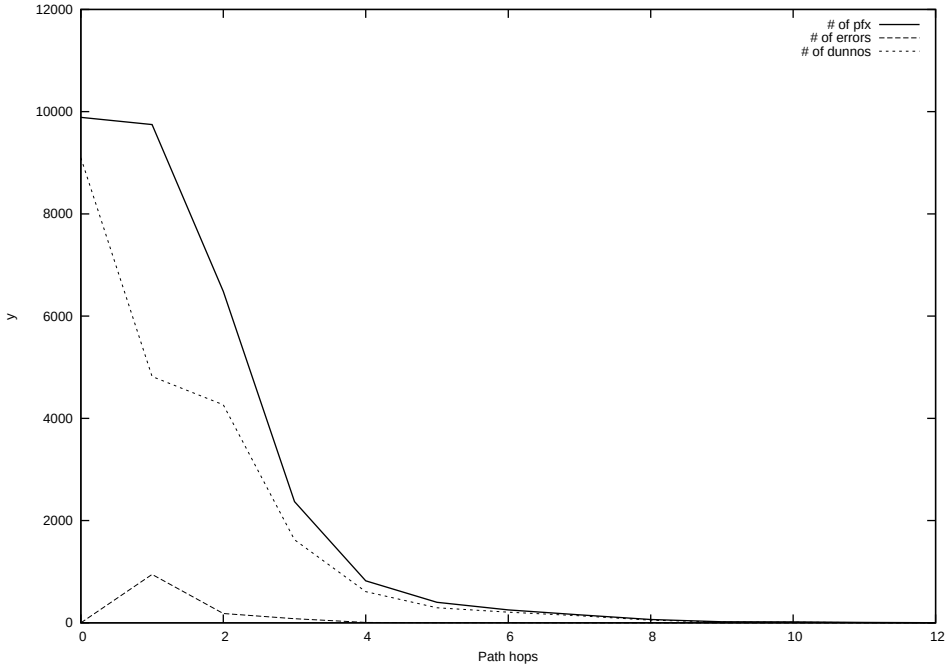
2012-08-06



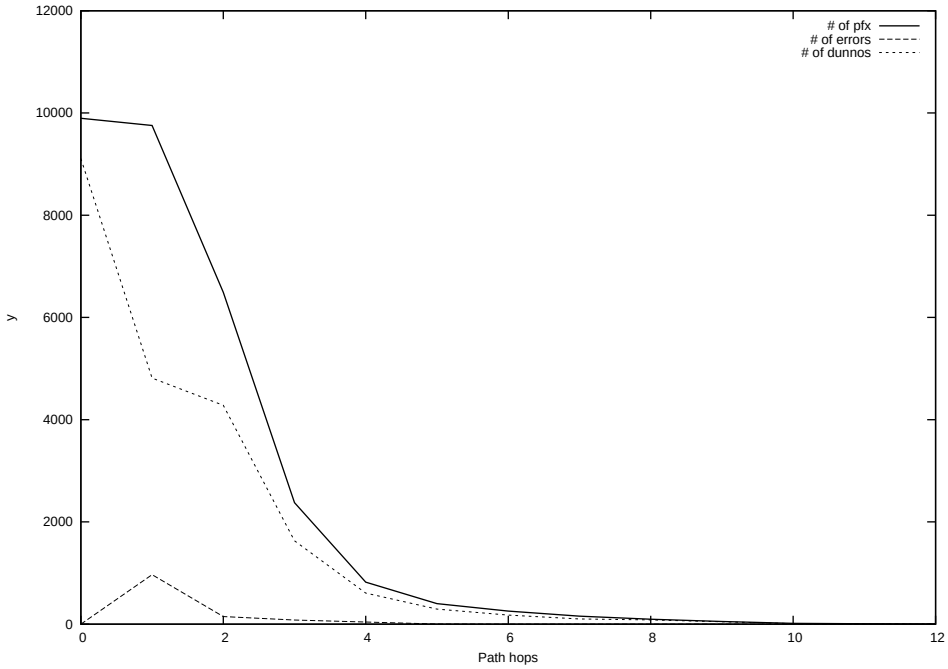
2012-08-07



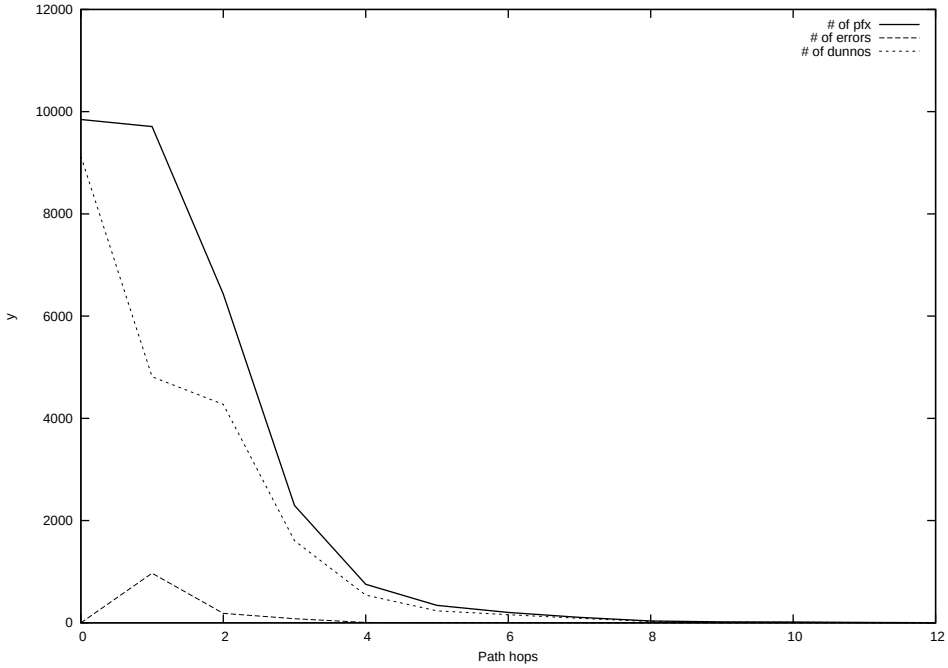
2012-08-08



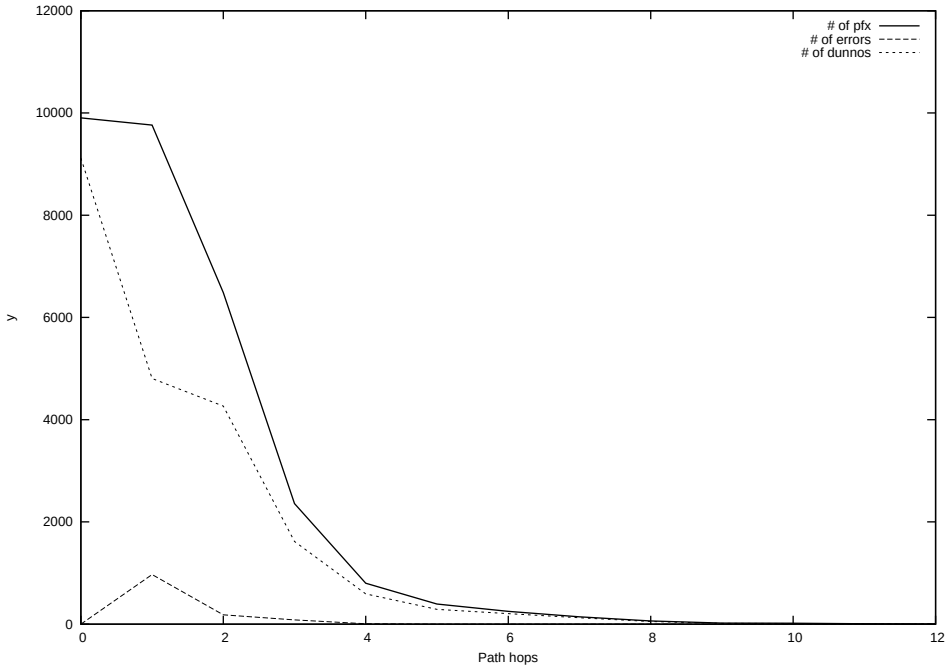
2012-08-09



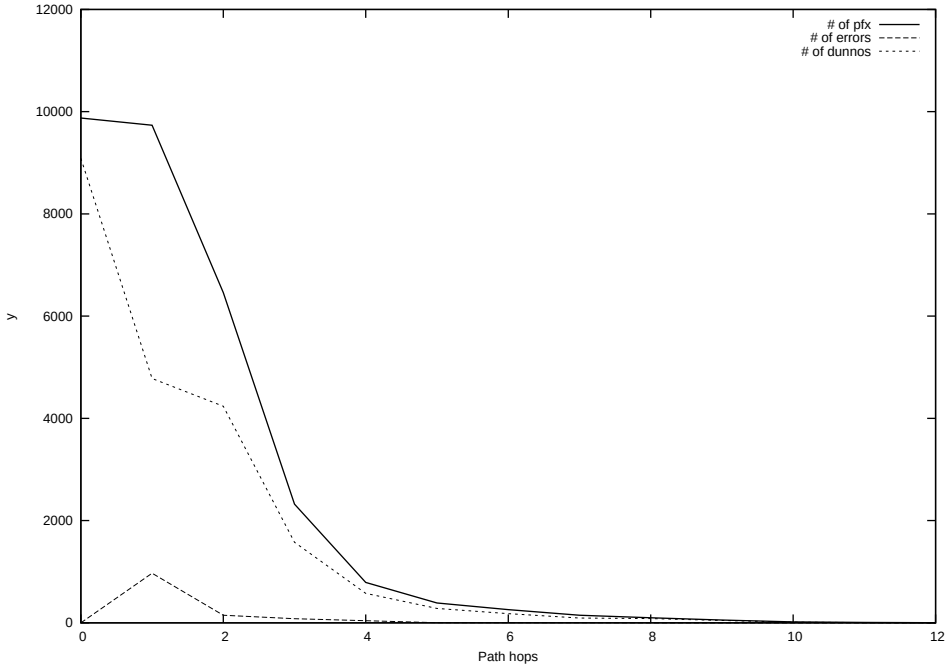
2012-08-10



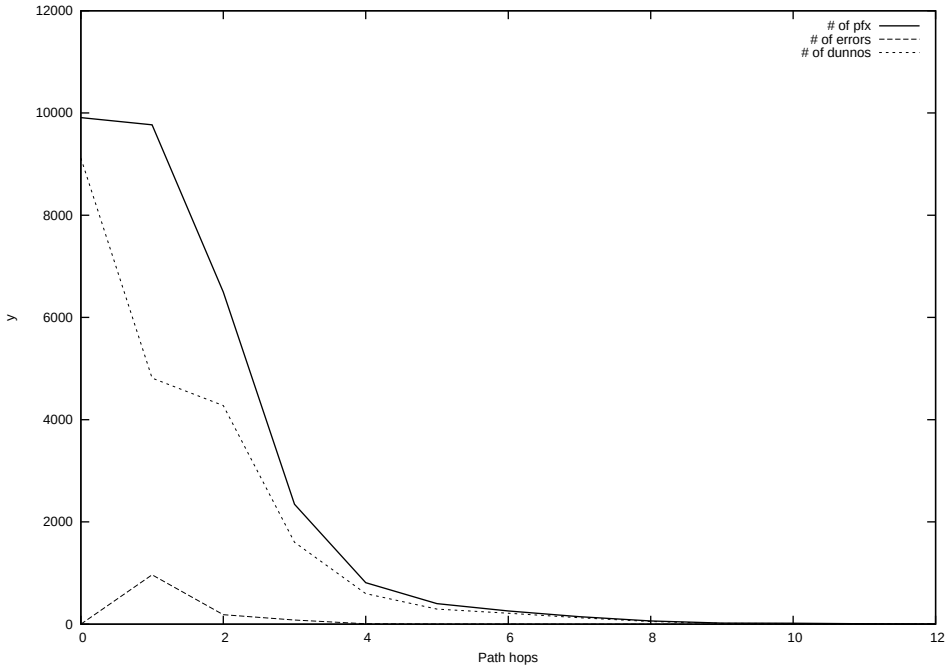
2012-08-11



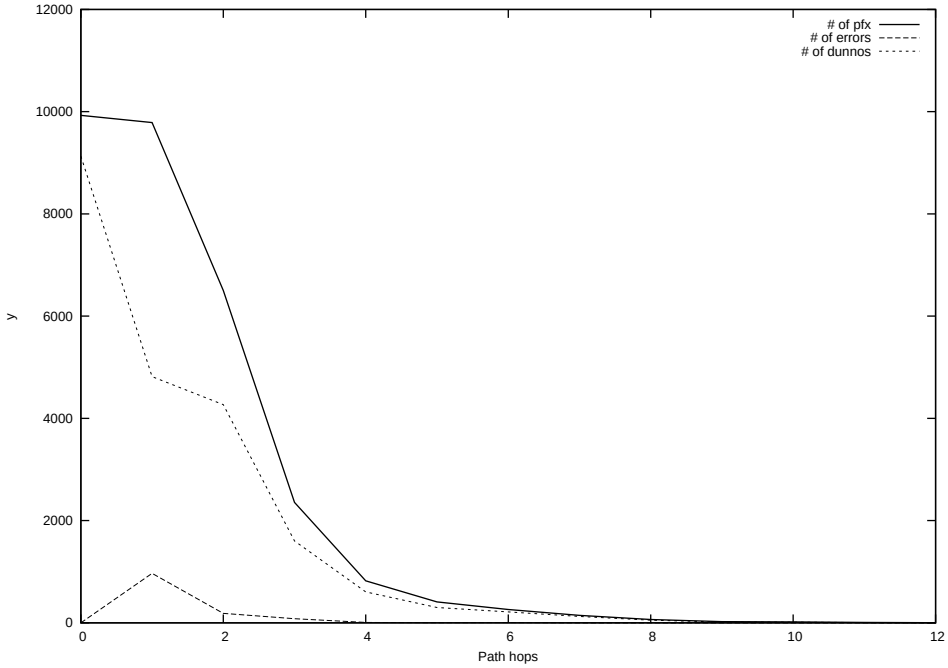
2012-08-12



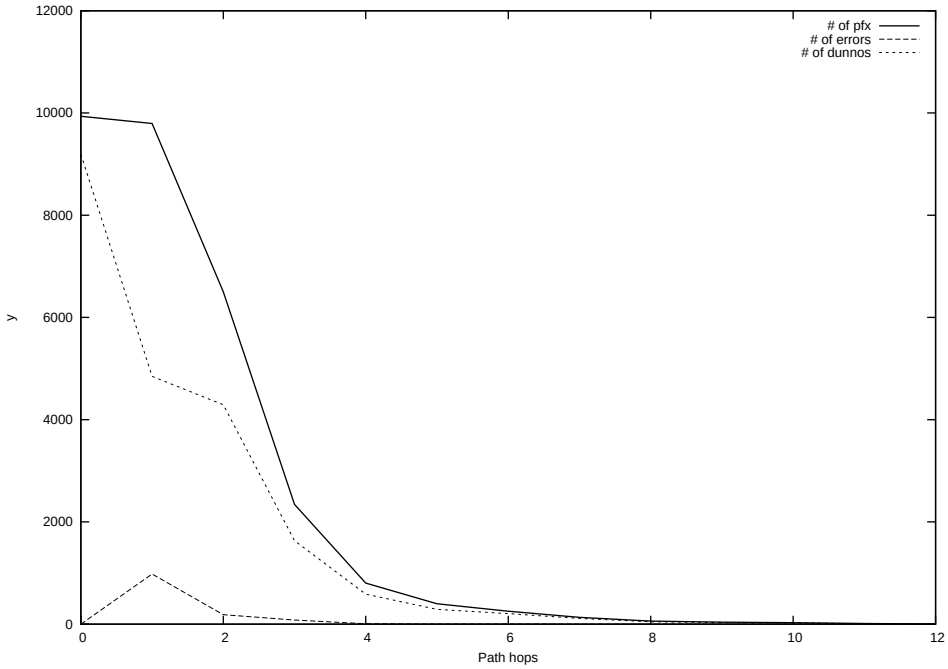
2012-08-13



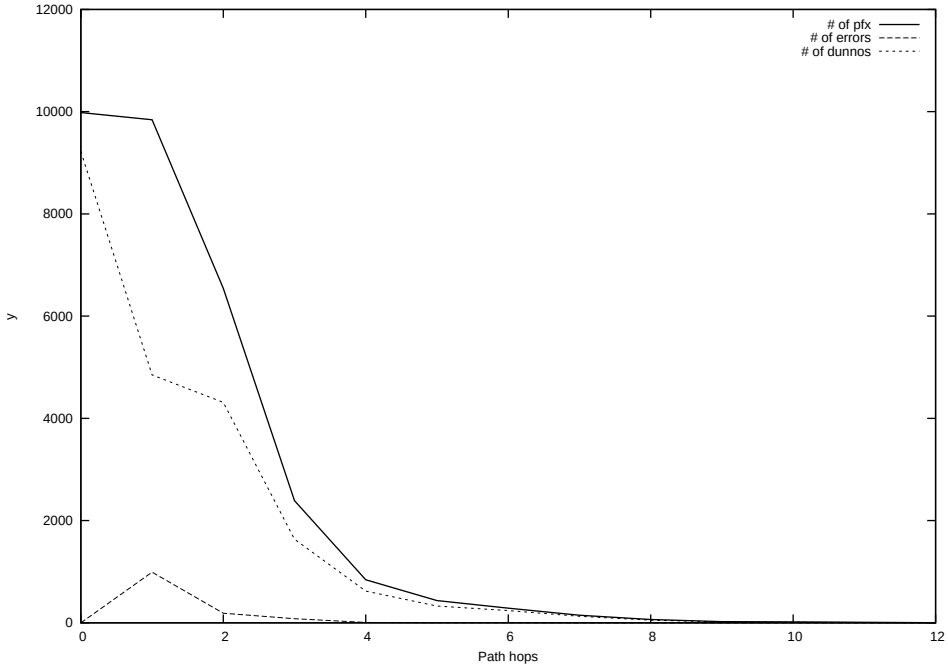
2012-08-14



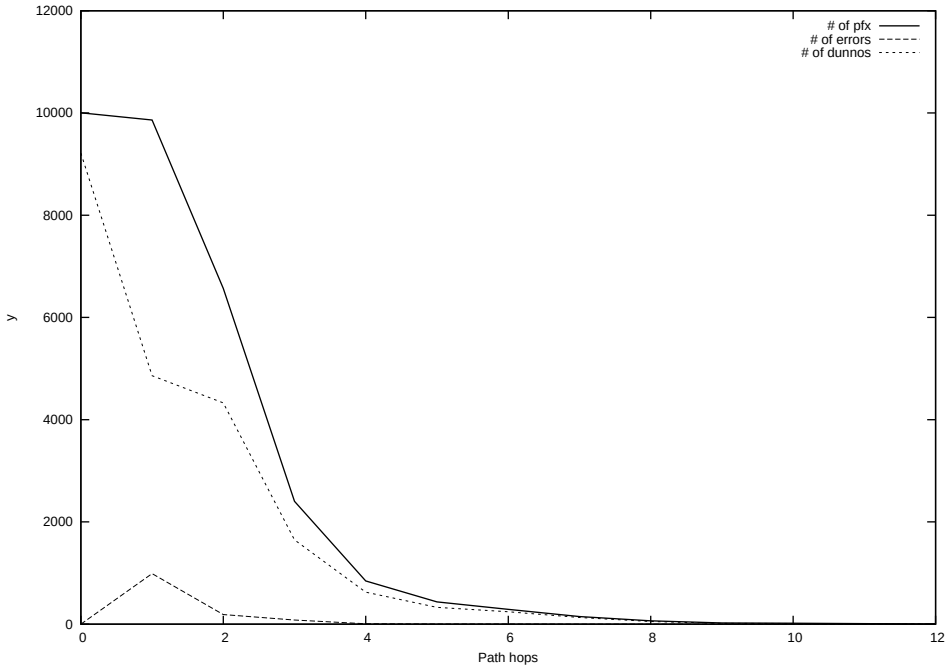
2012-08-15



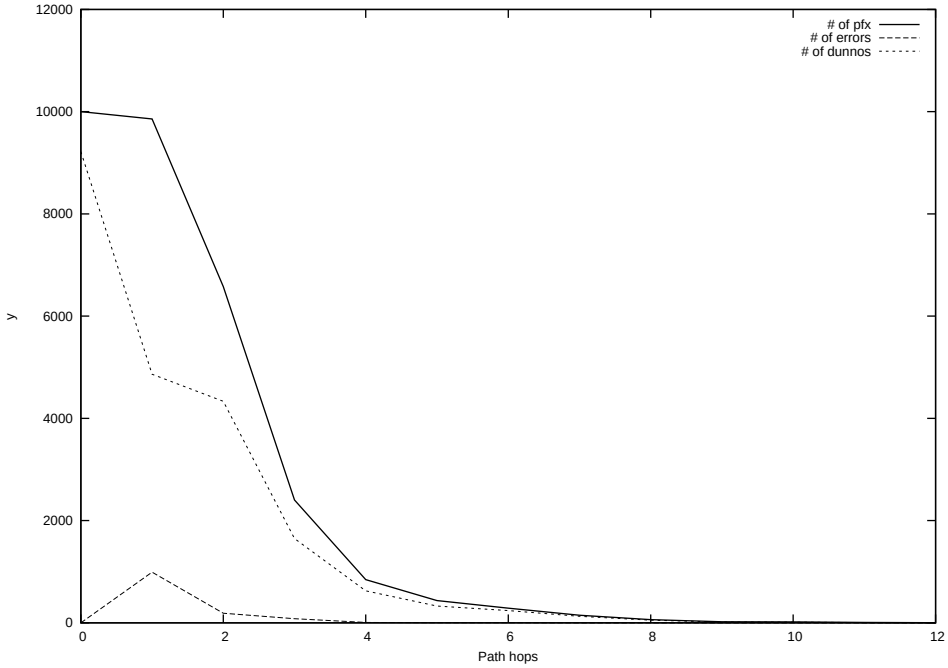
2012-08-16



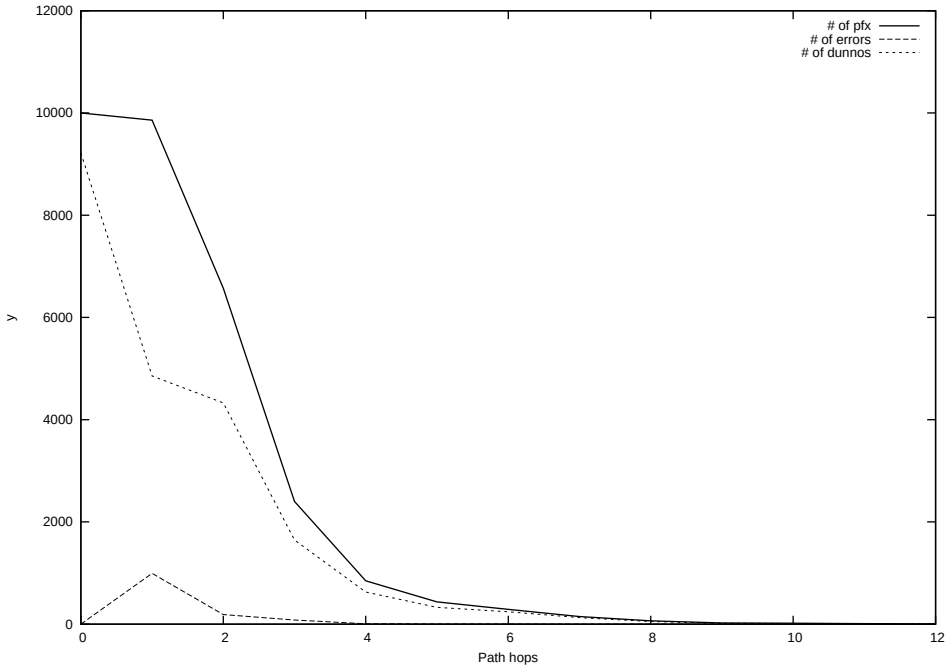
2012-08-17



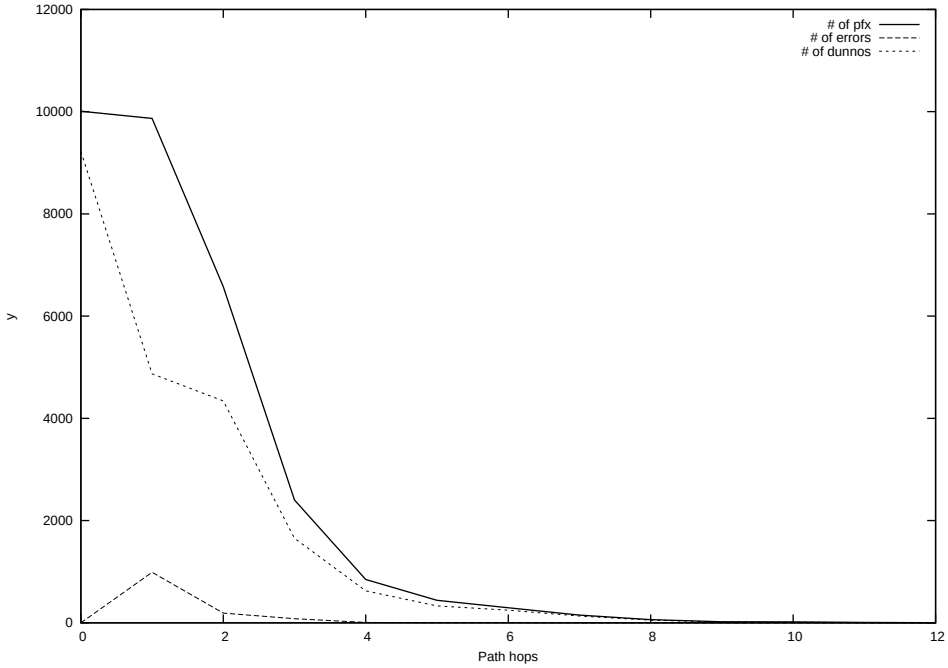
2012-08-18



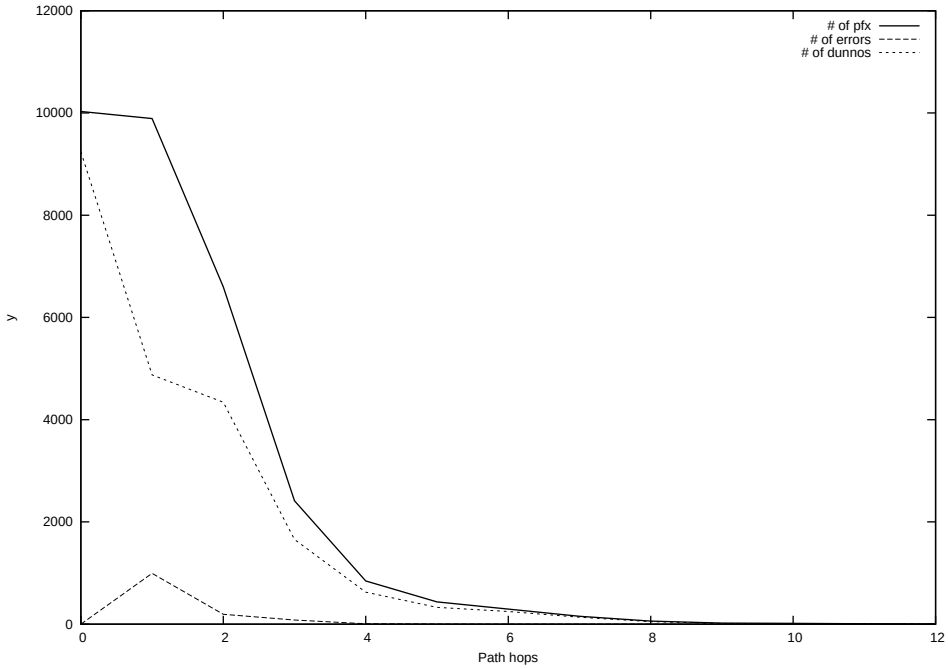
2012-08-19



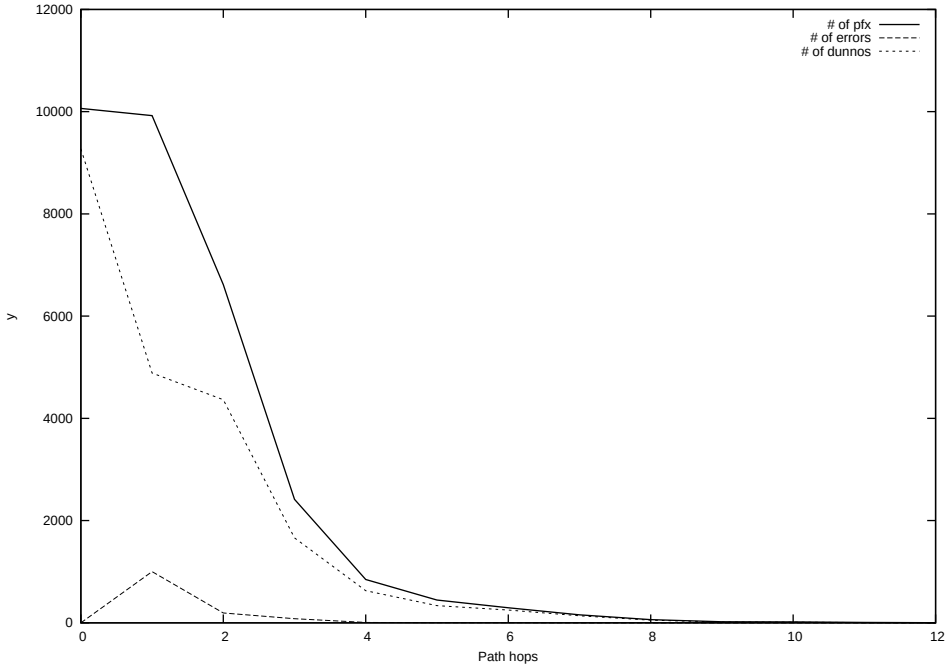
2012-08-20



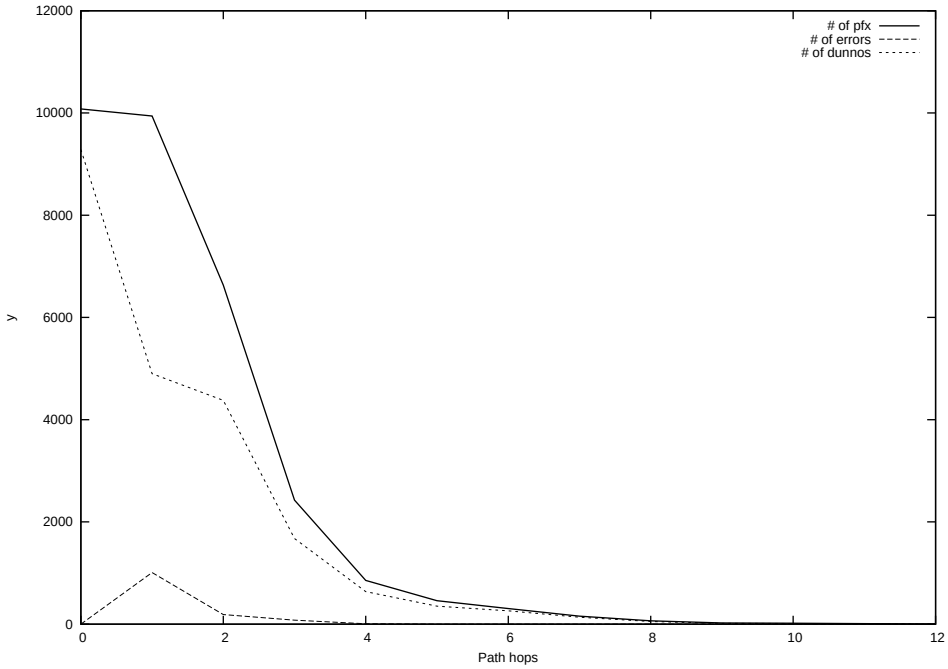
2012-08-21



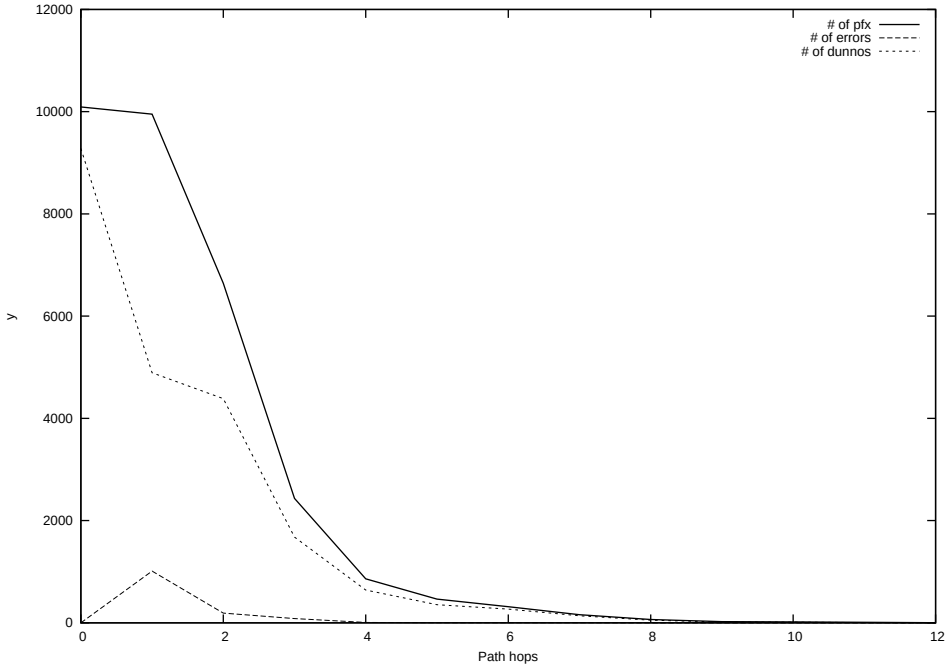
2012-08-22



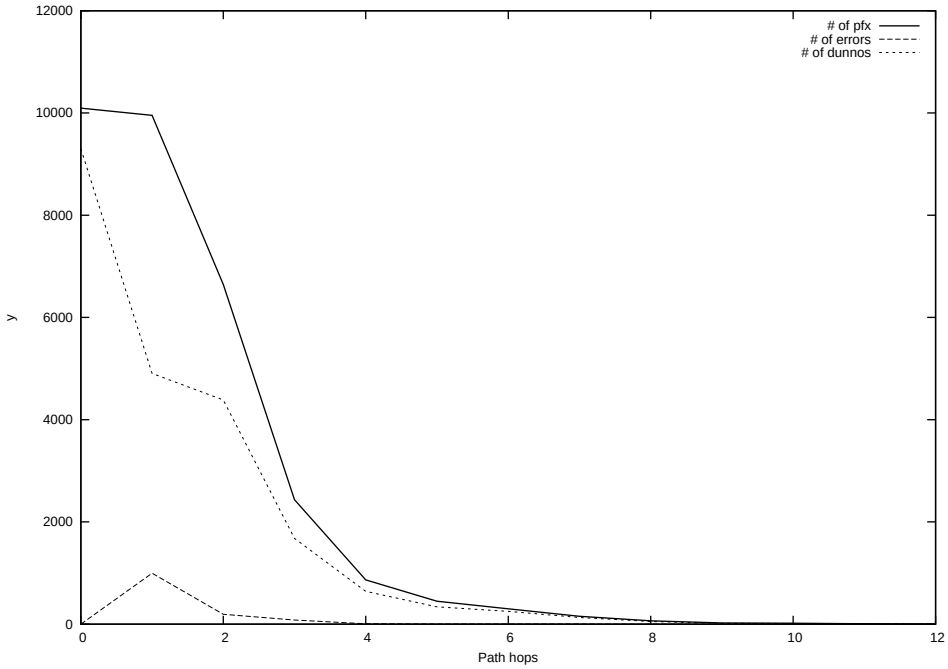
2012-08-23



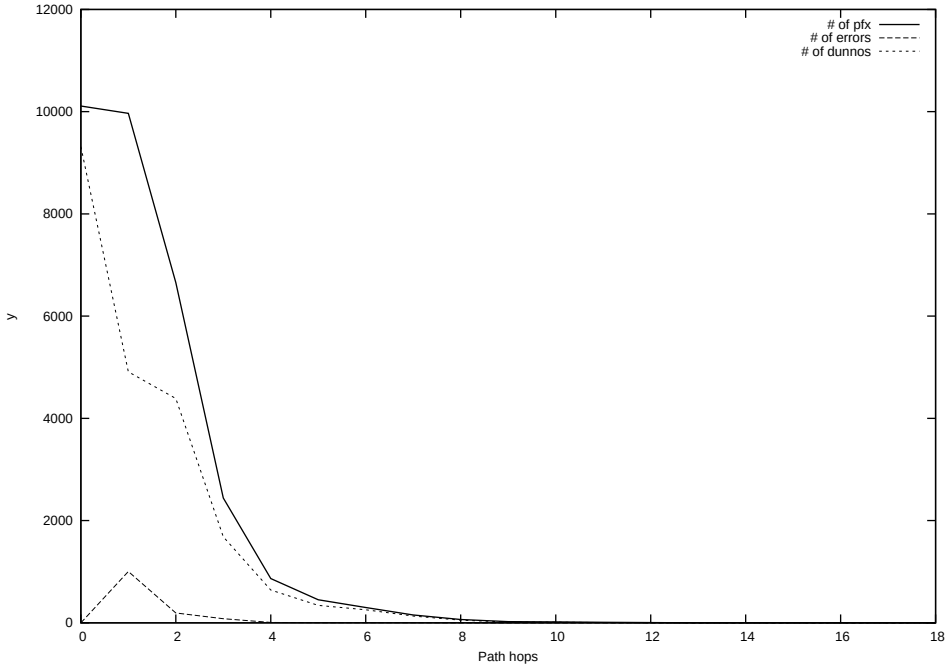
2012-08-24



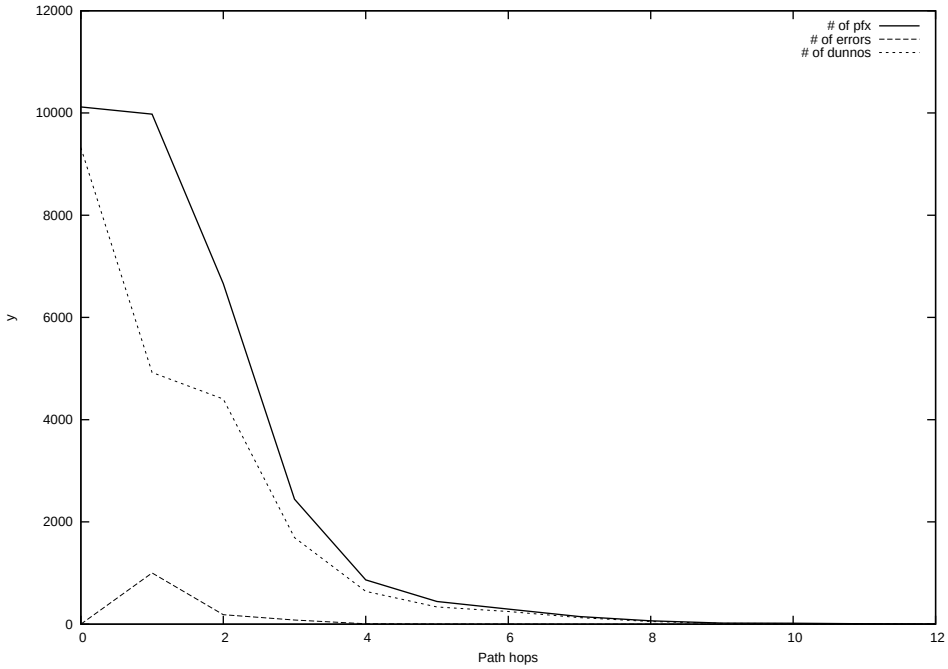
2012-08-25



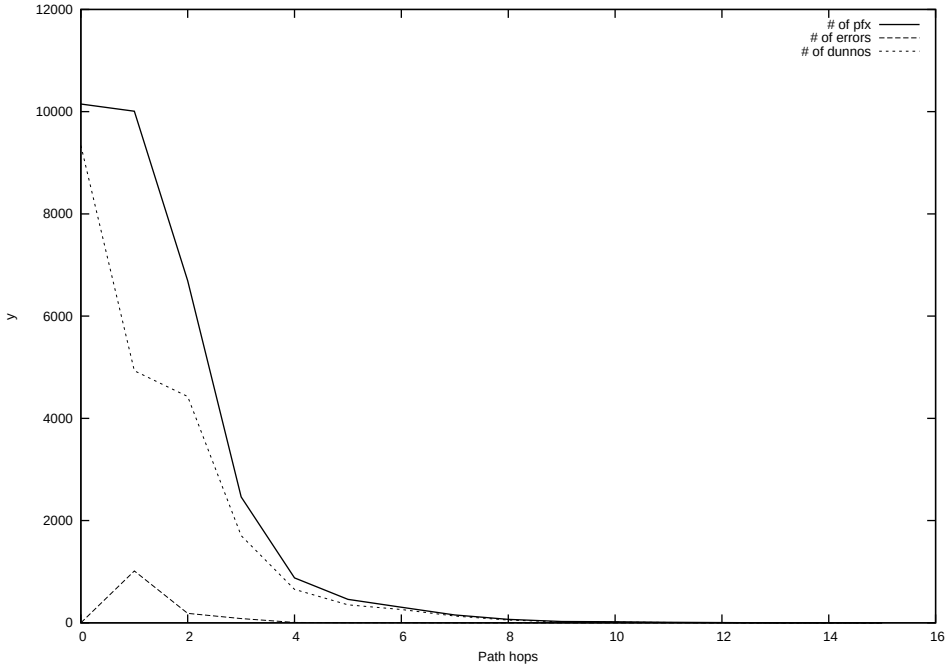
2012-08-26



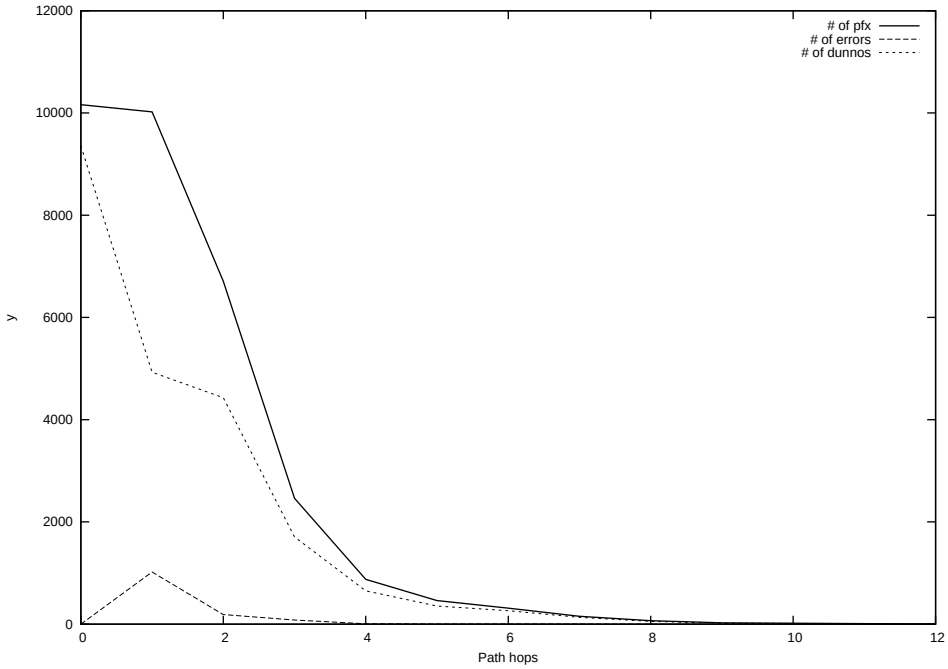
2012-08-27



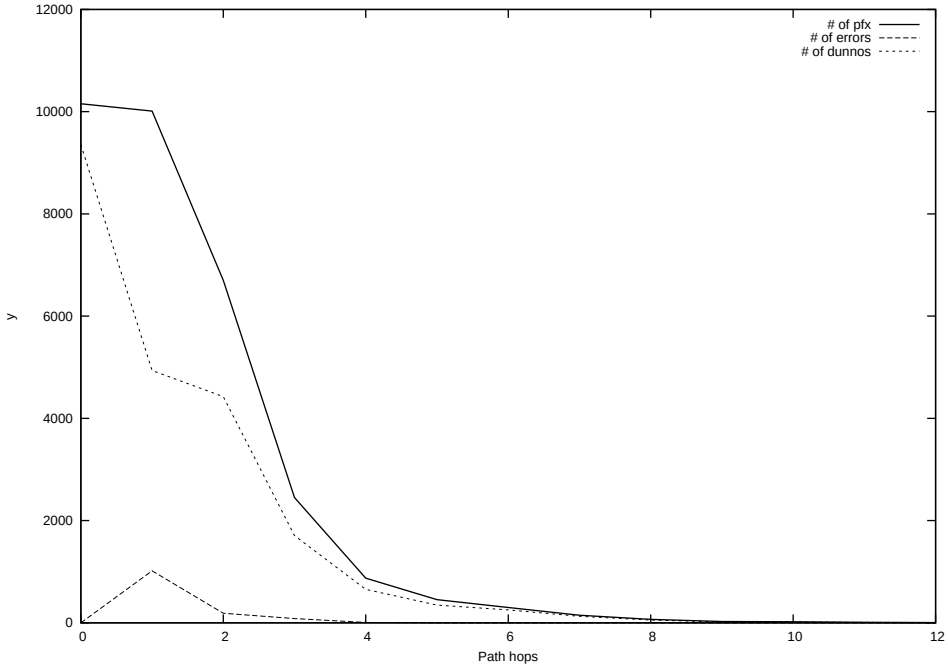
2012-08-28



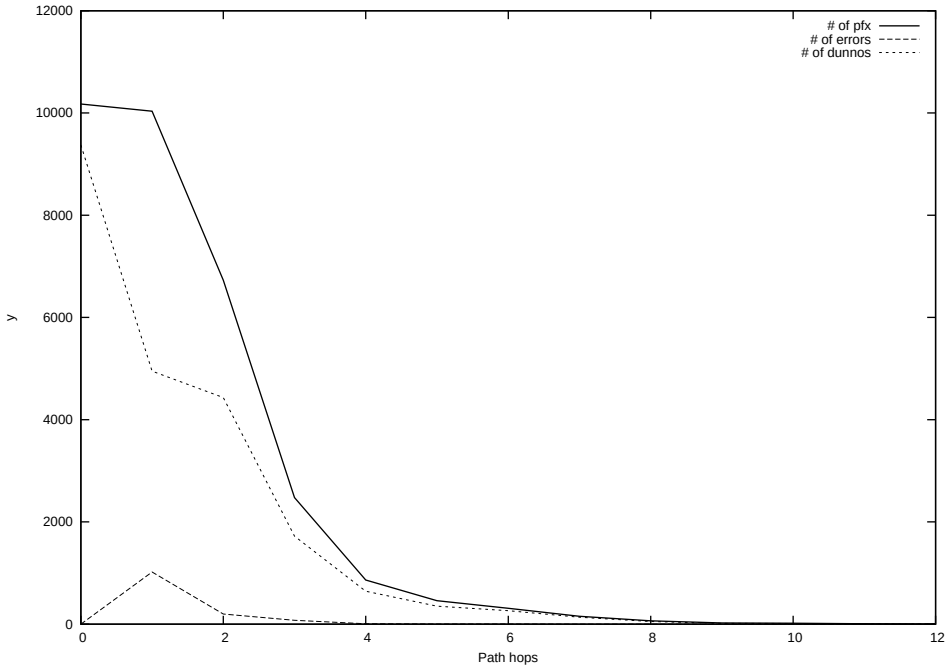
2012-08-29



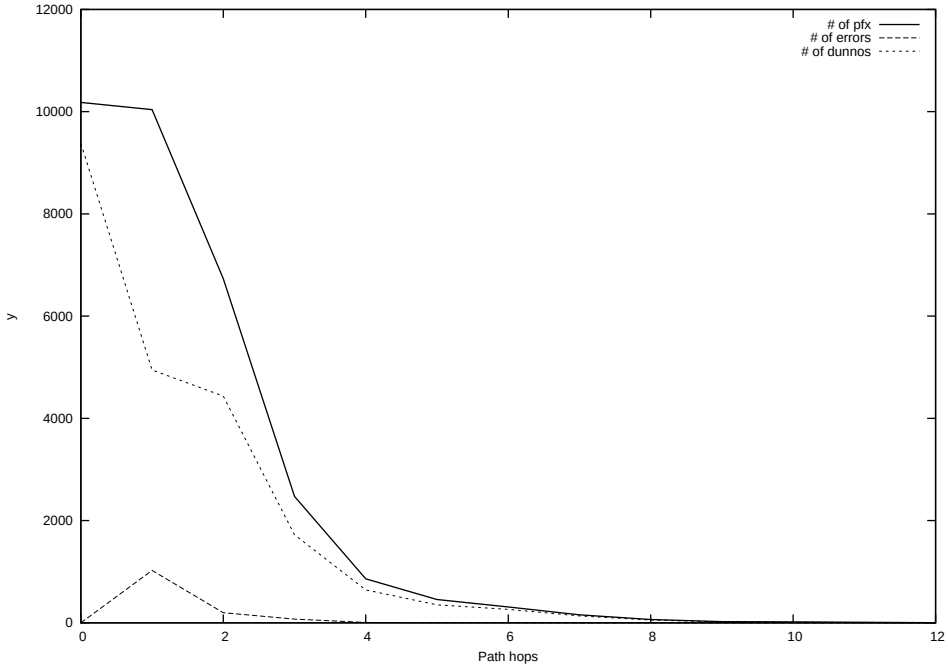
2012-08-30



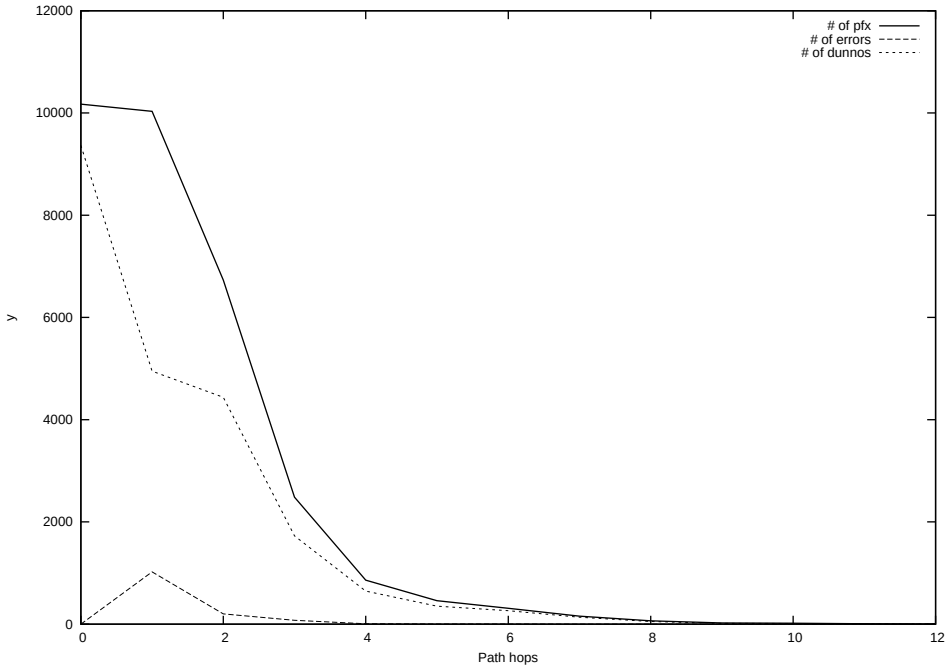
2012-08-31



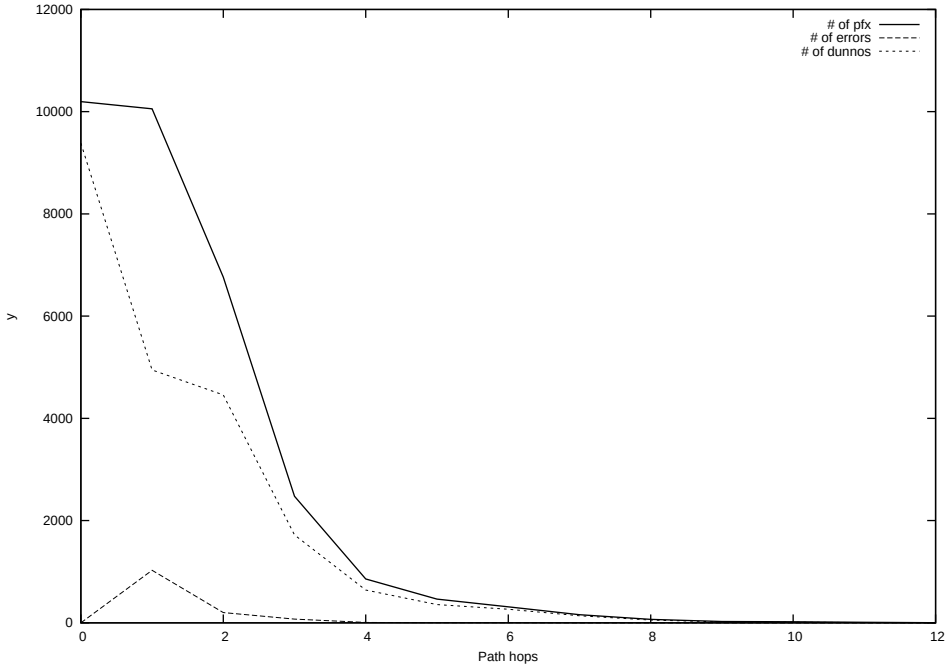
2012-09-01



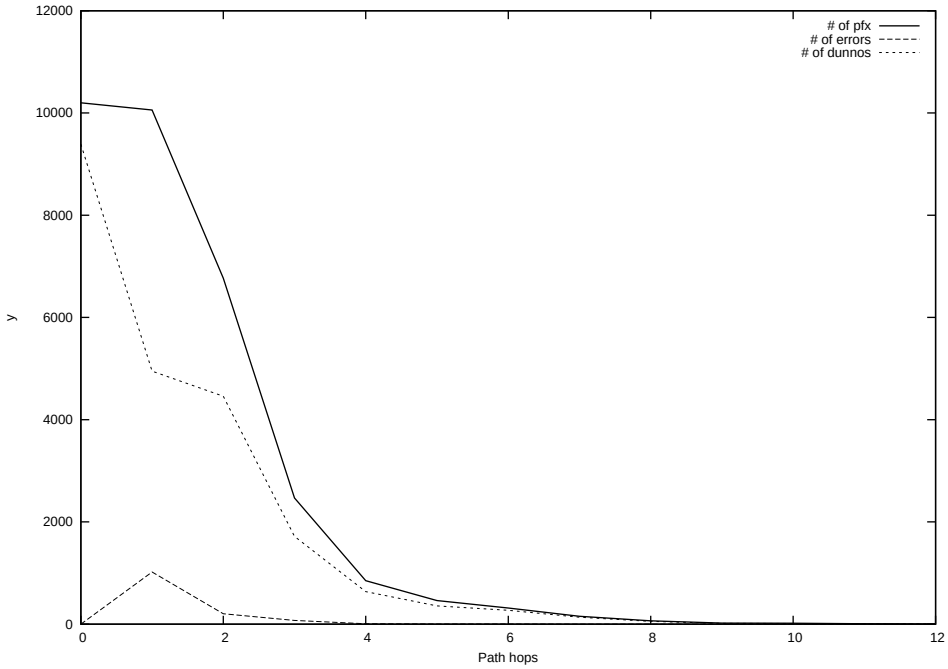
2012-09-02



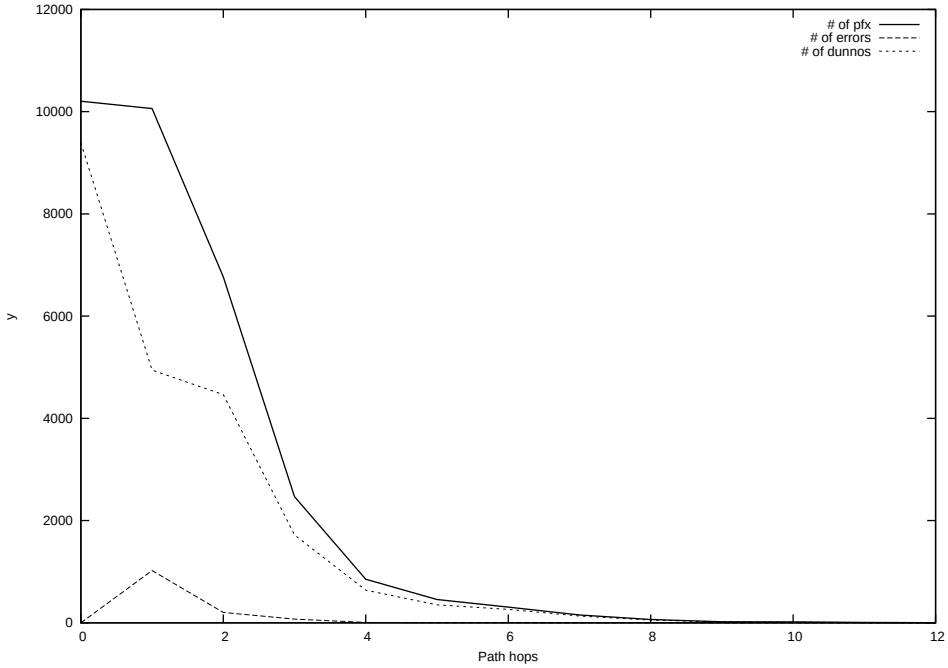
2012-09-03



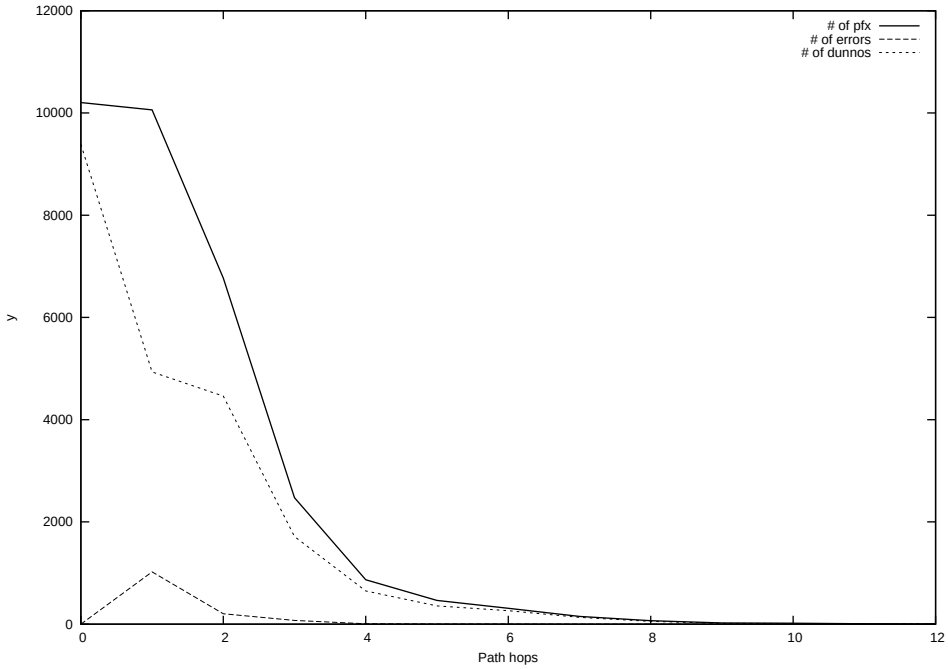
2012-09-04



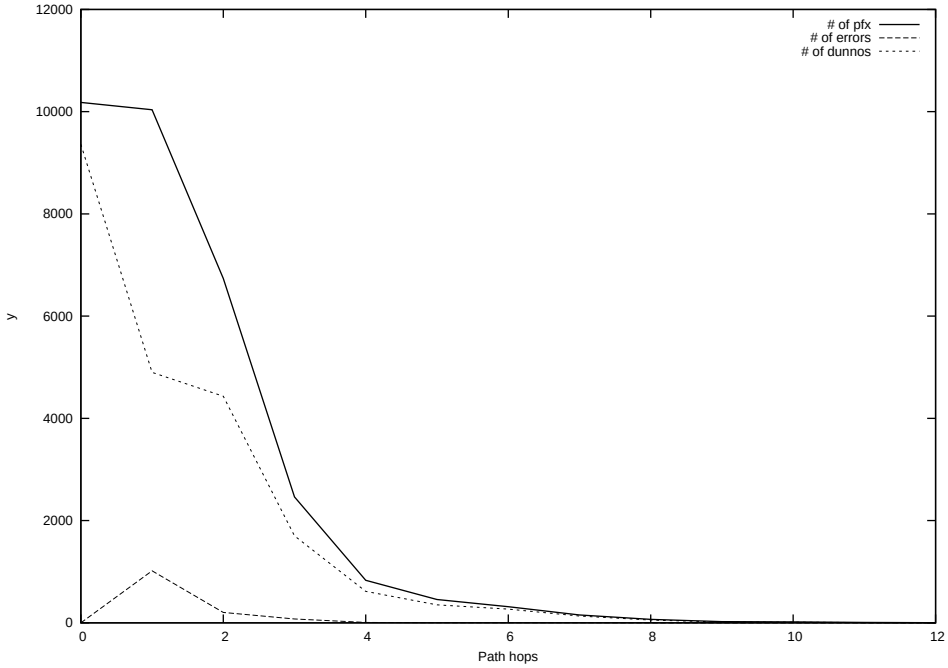
2012-09-05



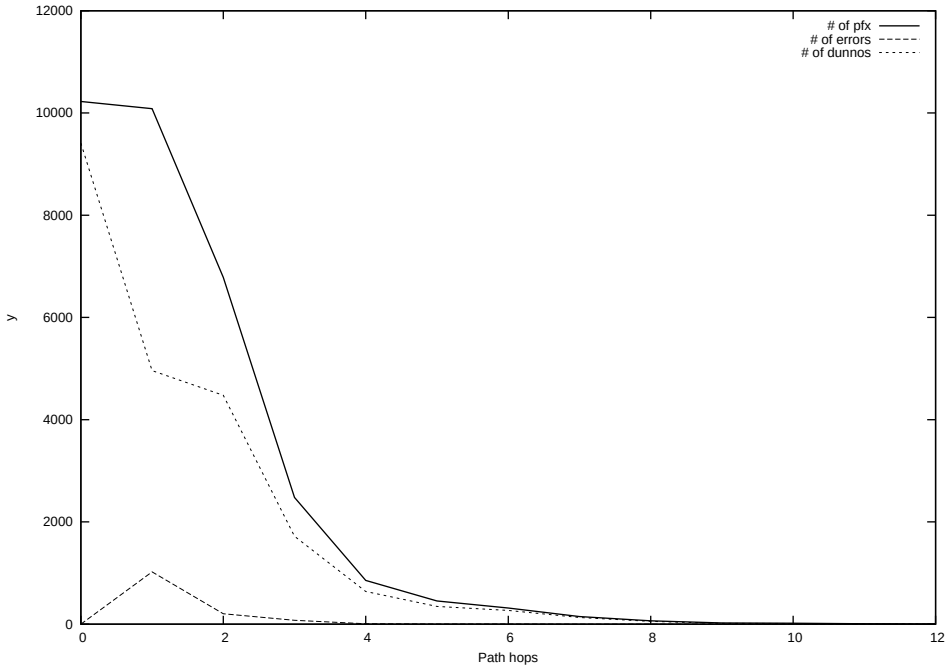
2012-09-06



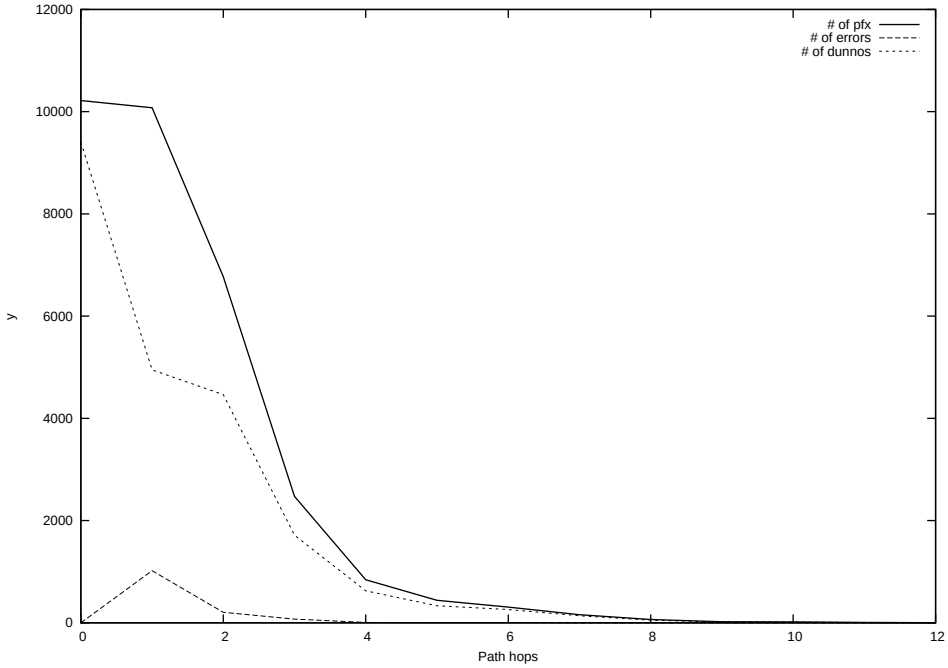
2012-09-07



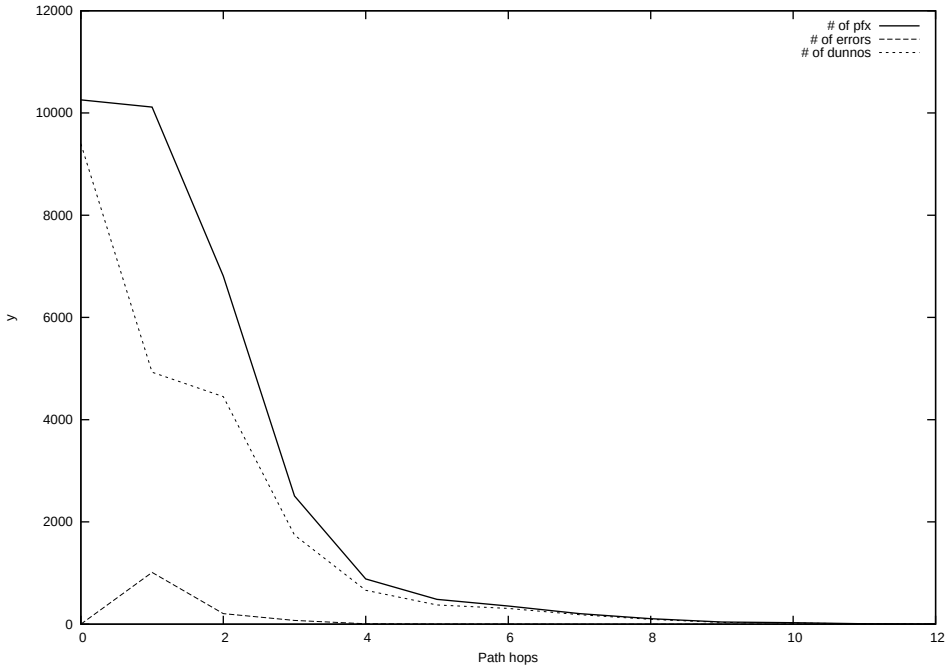
2012-09-08



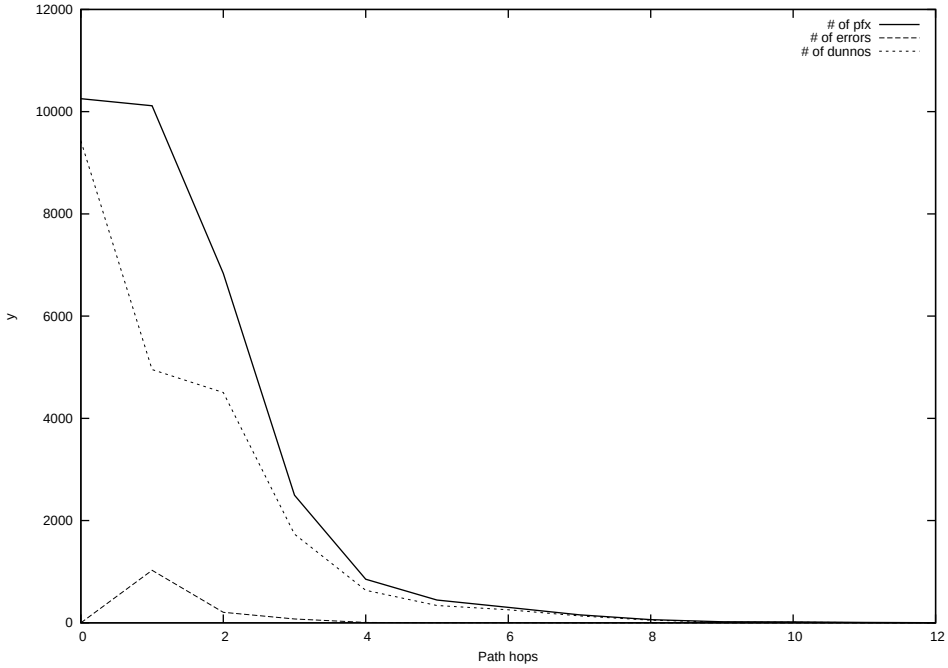
2012-09-09



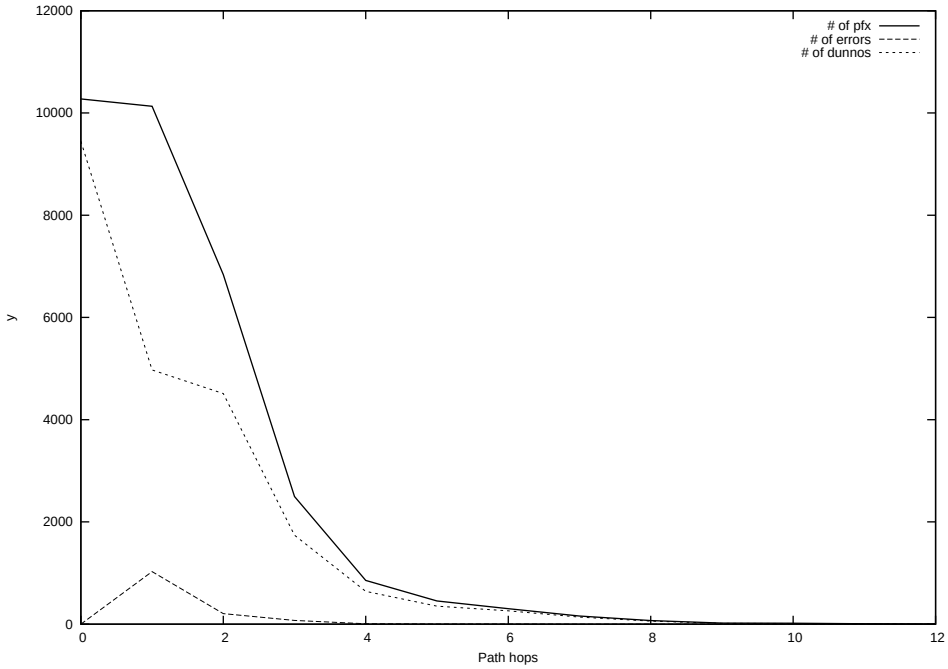
2012-09-10



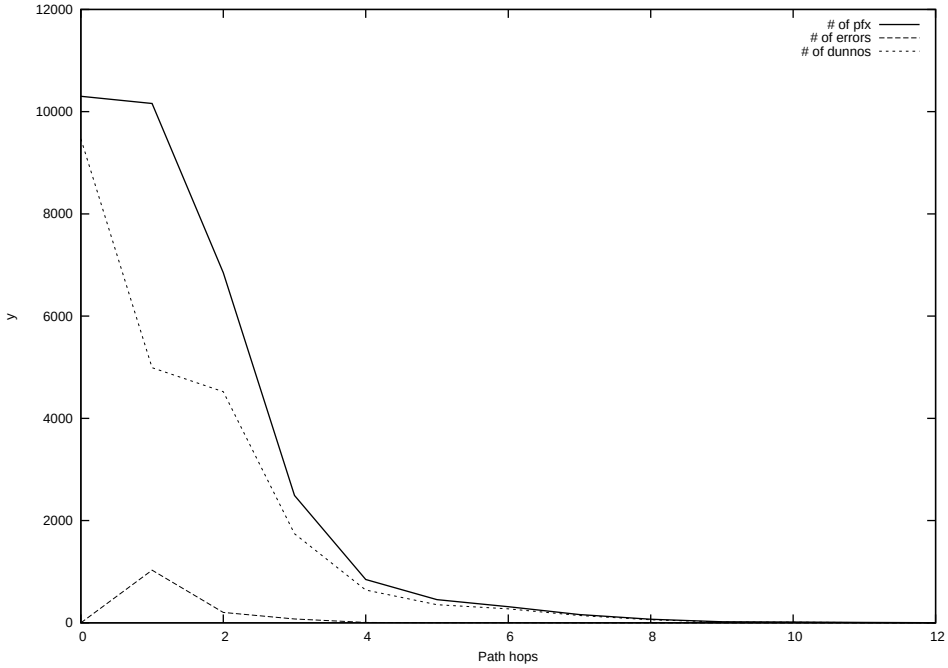
2012-09-11



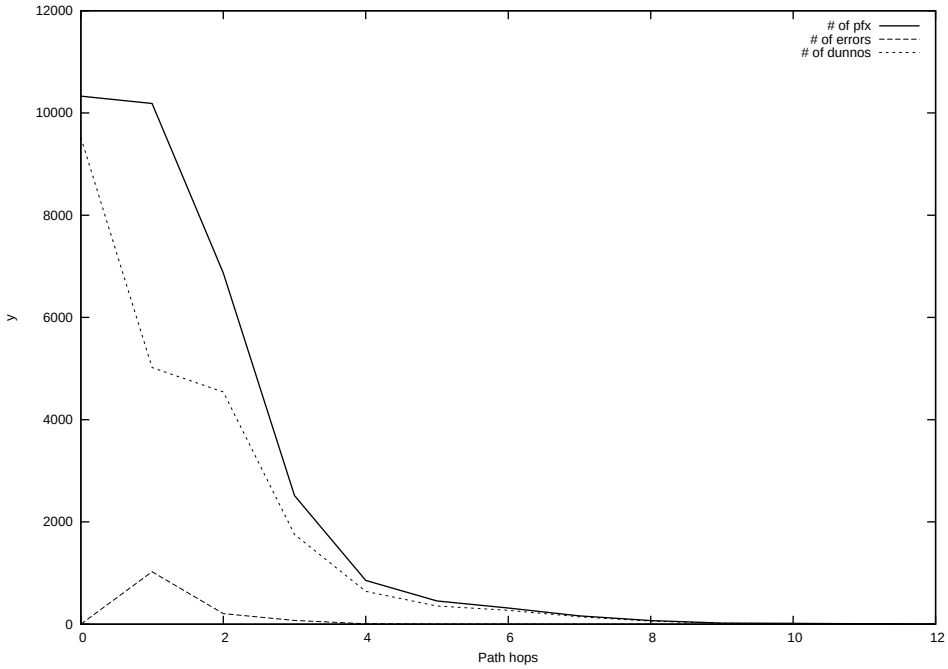
2012-09-12



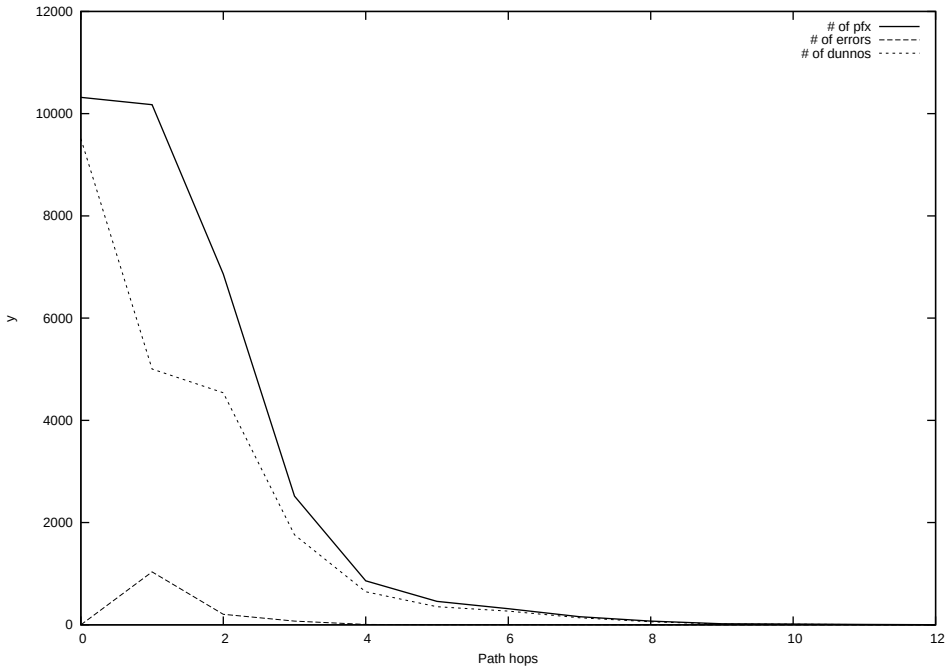
2012-09-13



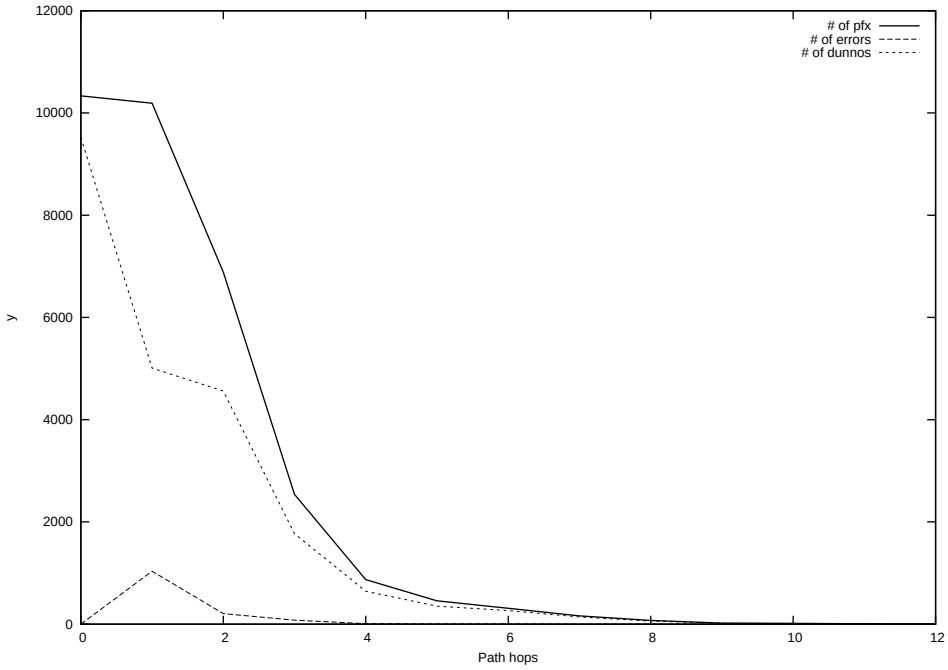
2012-09-14



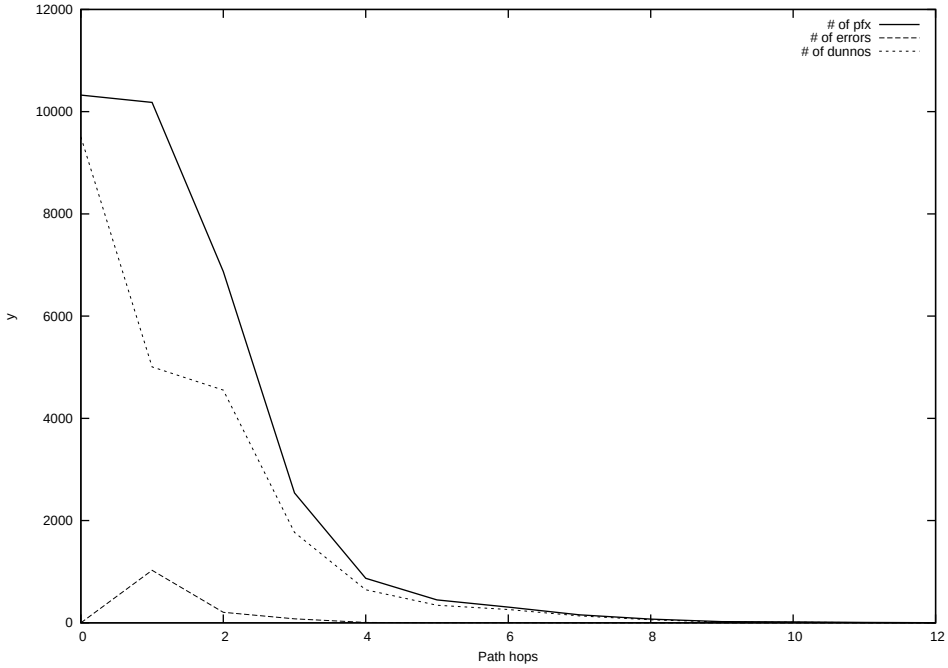
2012-09-15



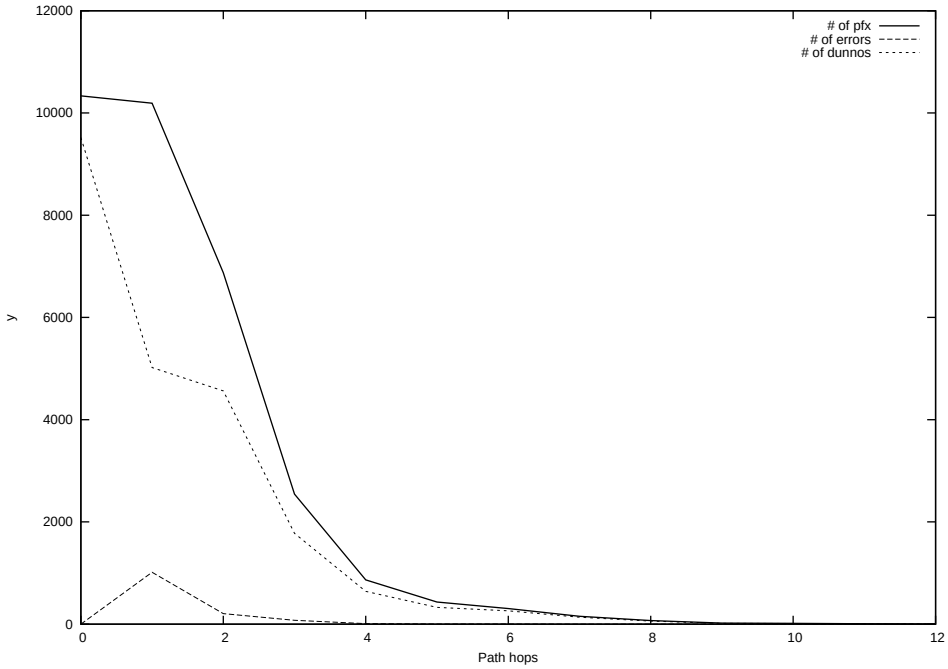
2012-09-16



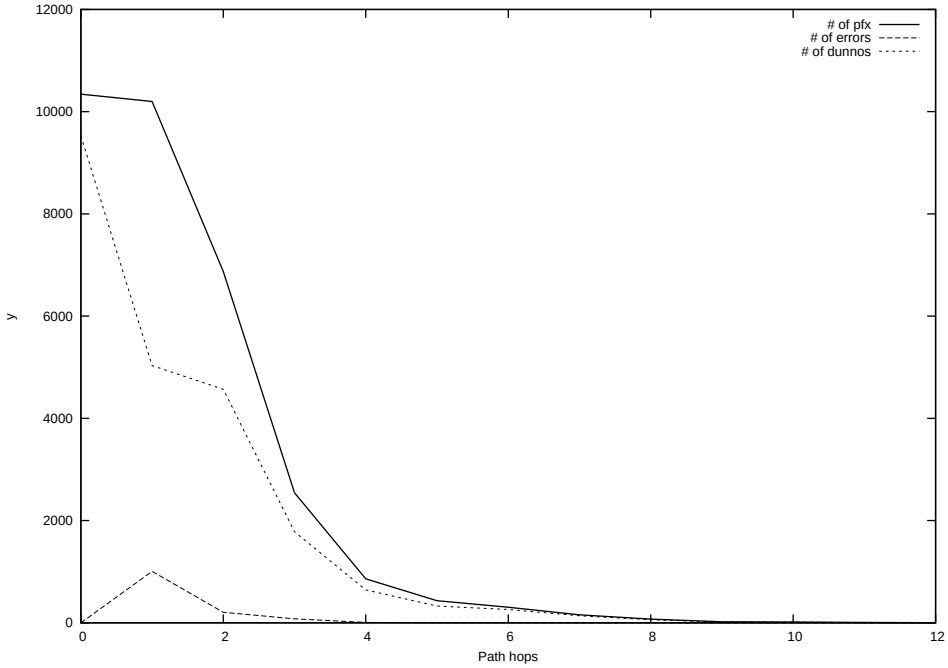
2012-09-17



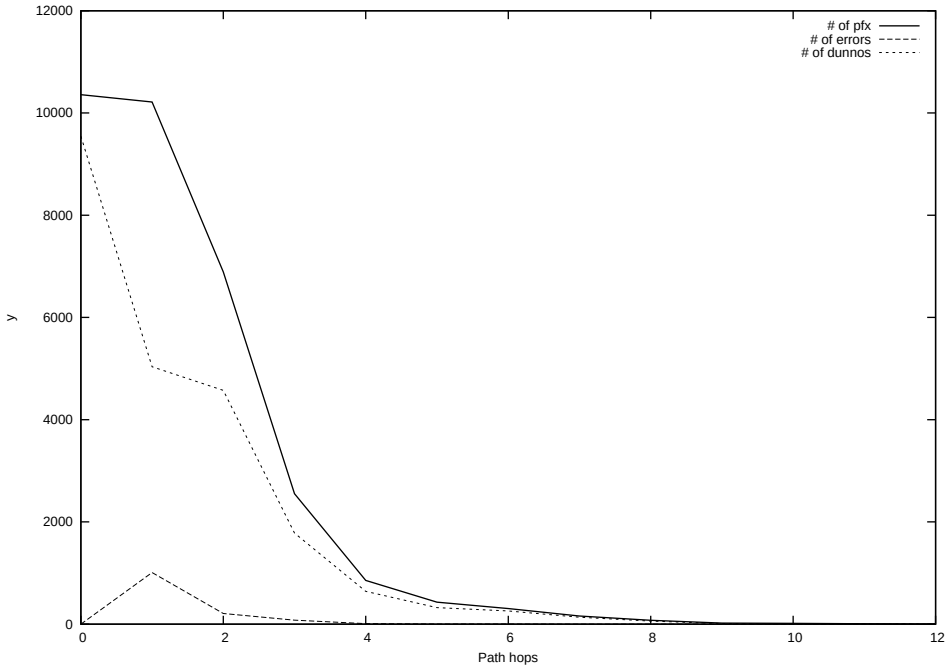
2012-09-18



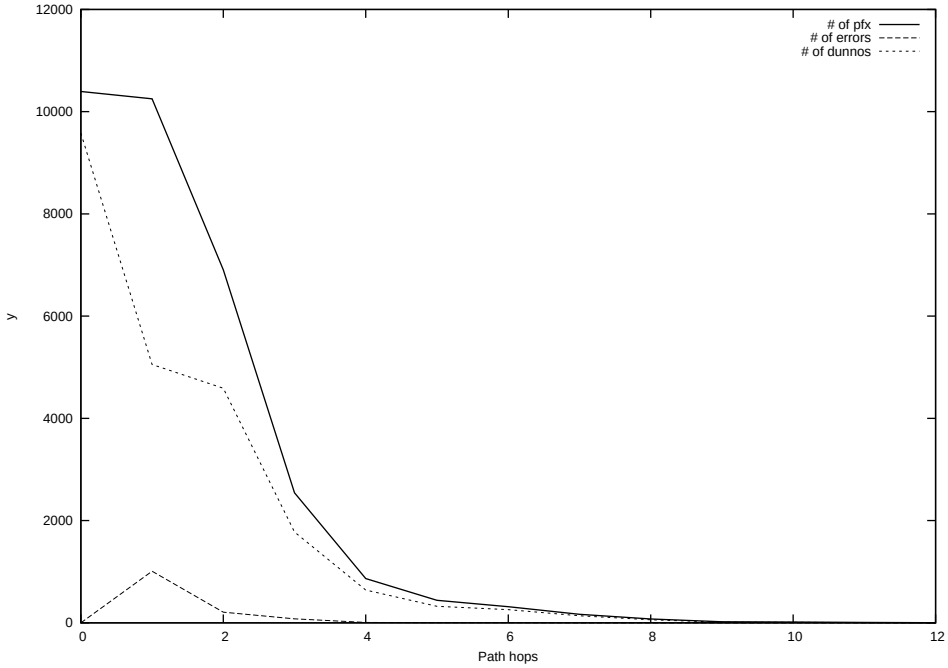
2012-09-19



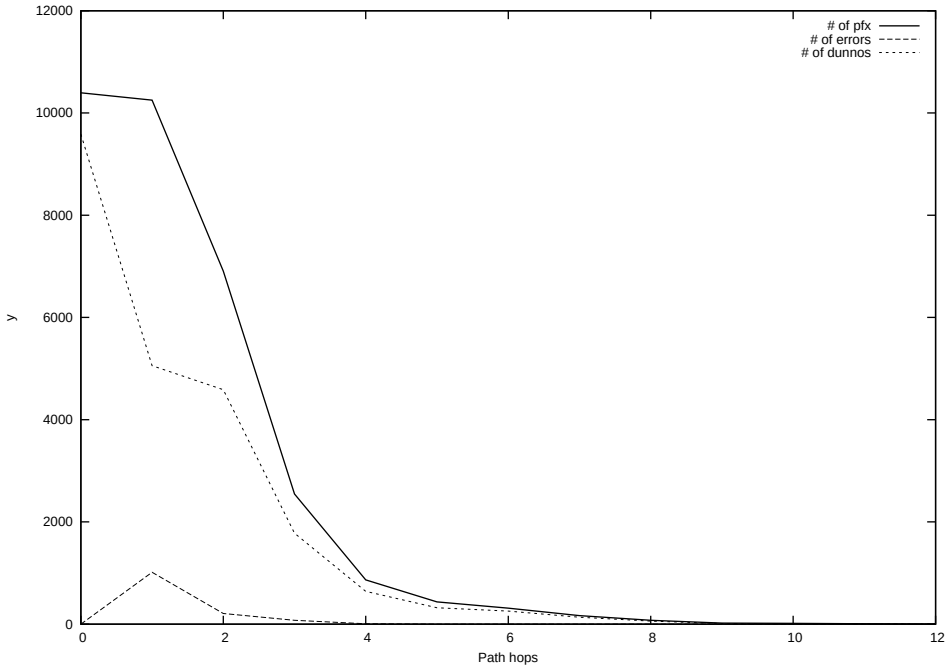
2012-09-20



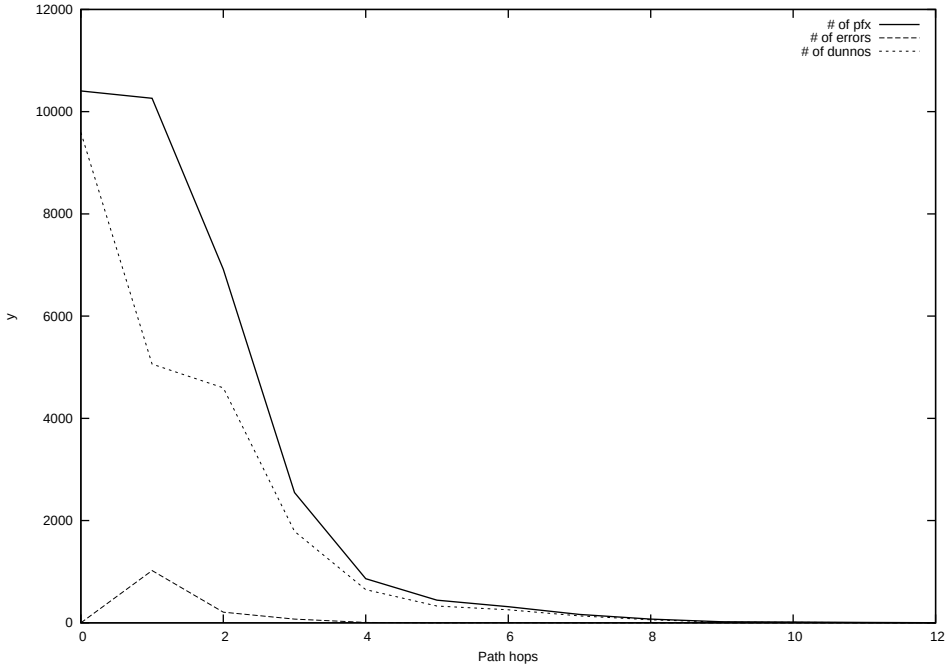
2012-09-21



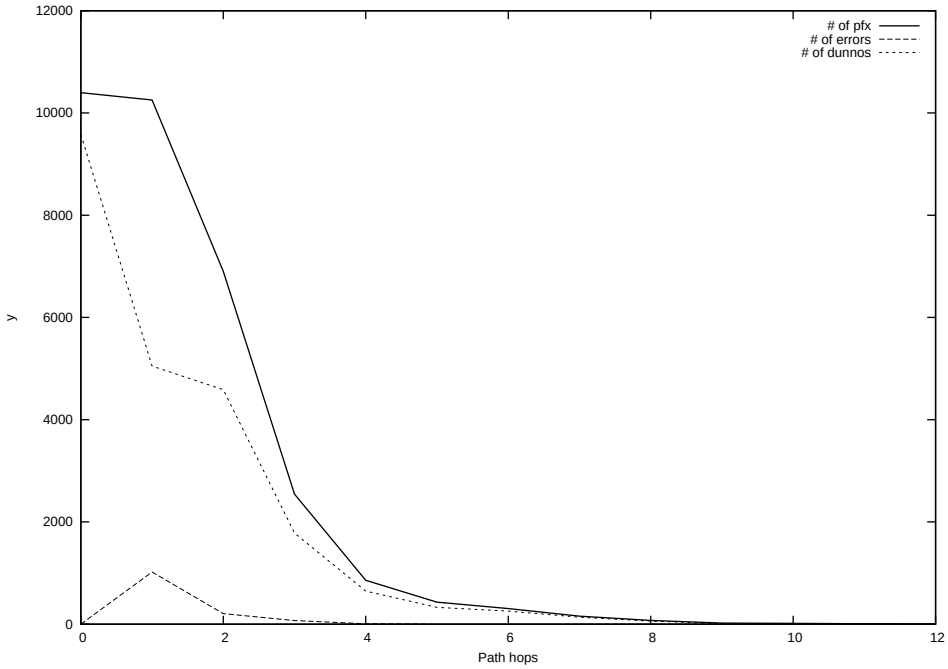
2012-09-22



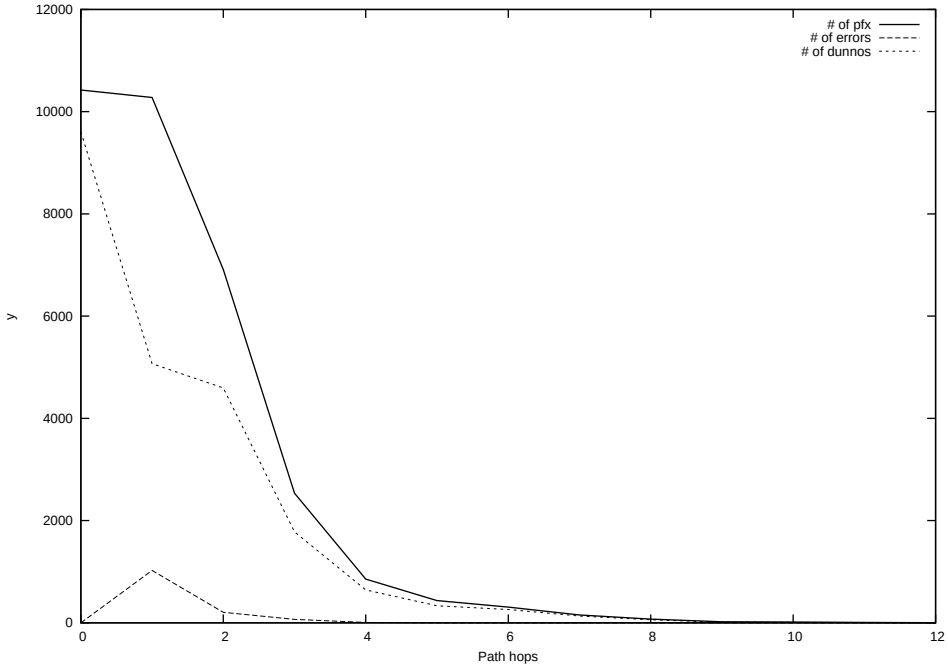
2012-09-23



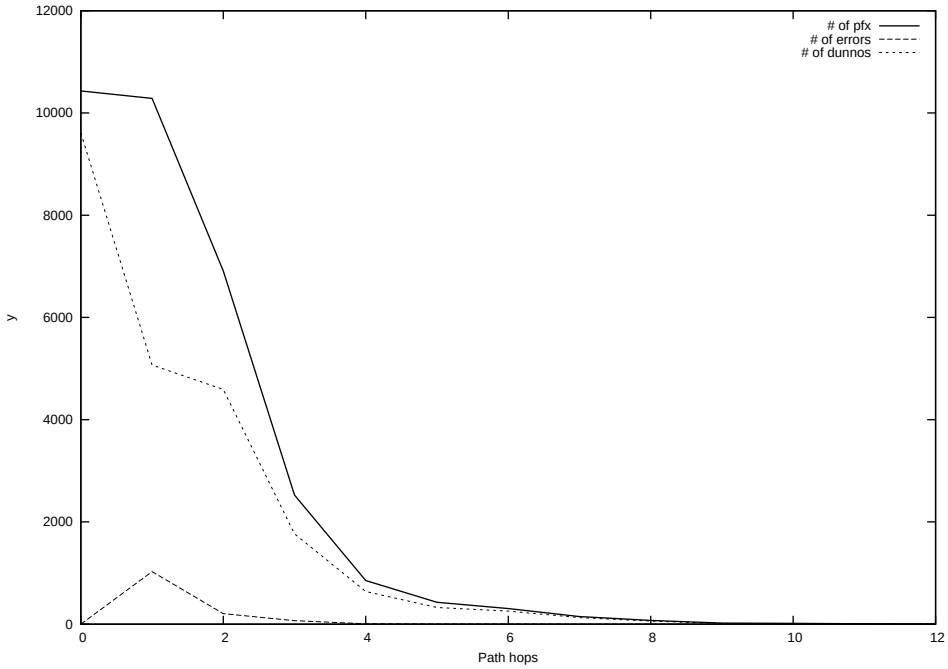
2012-09-24



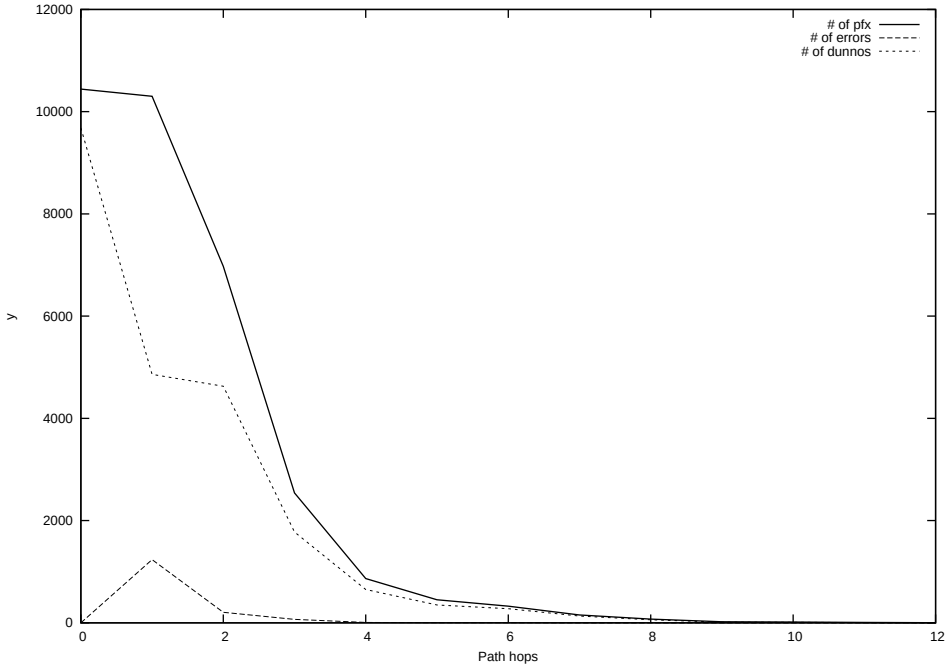
2012-09-25



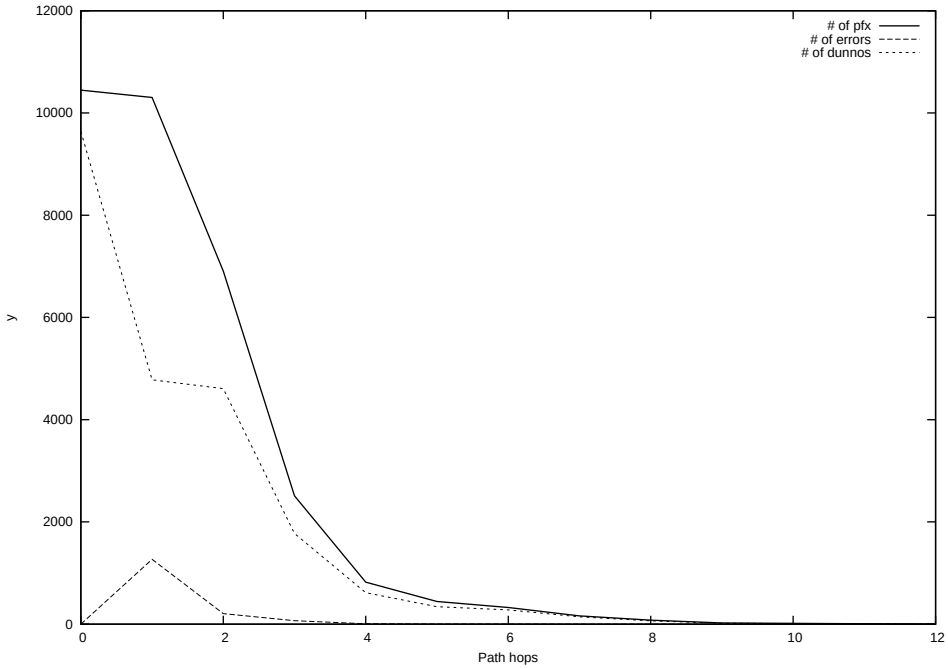
2012-09-26



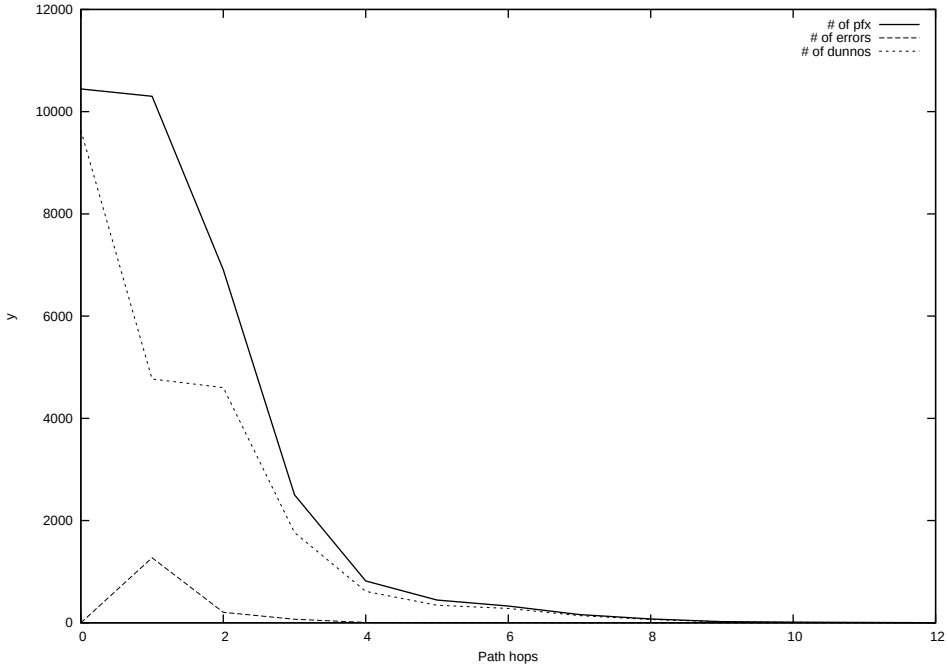
2012-09-27



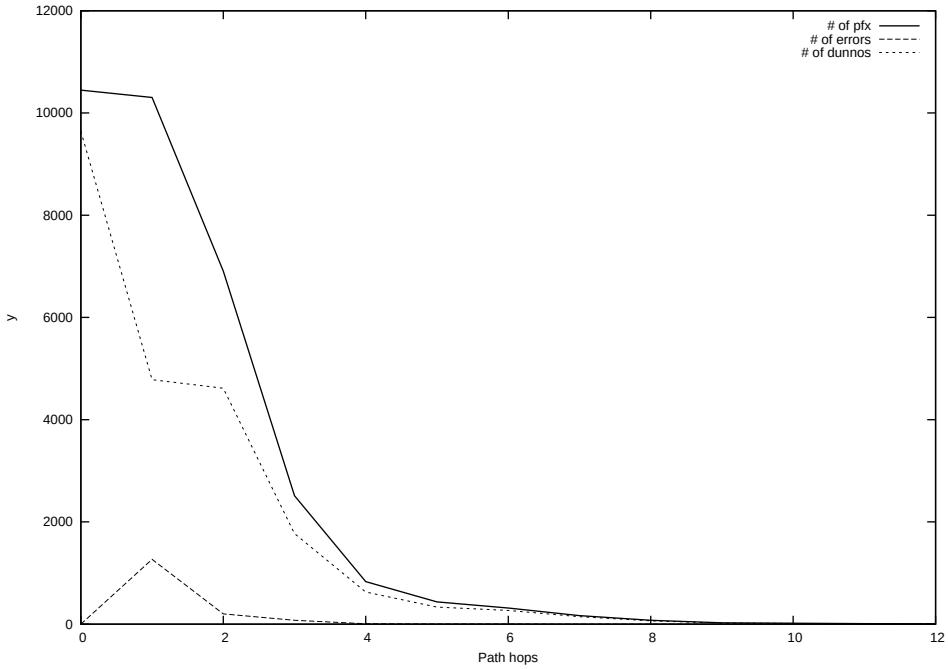
2012-09-28



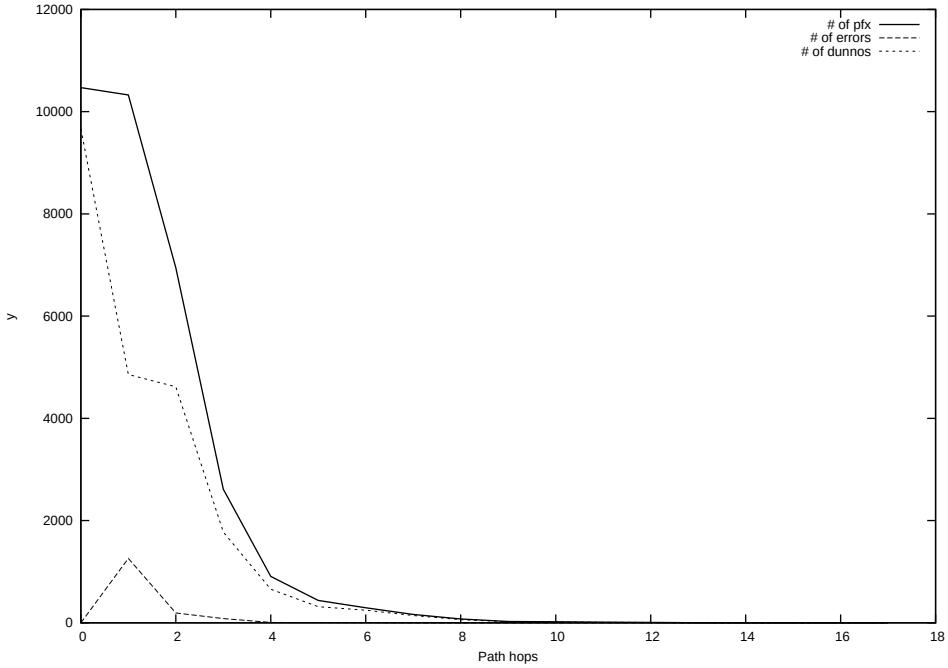
2012-09-29



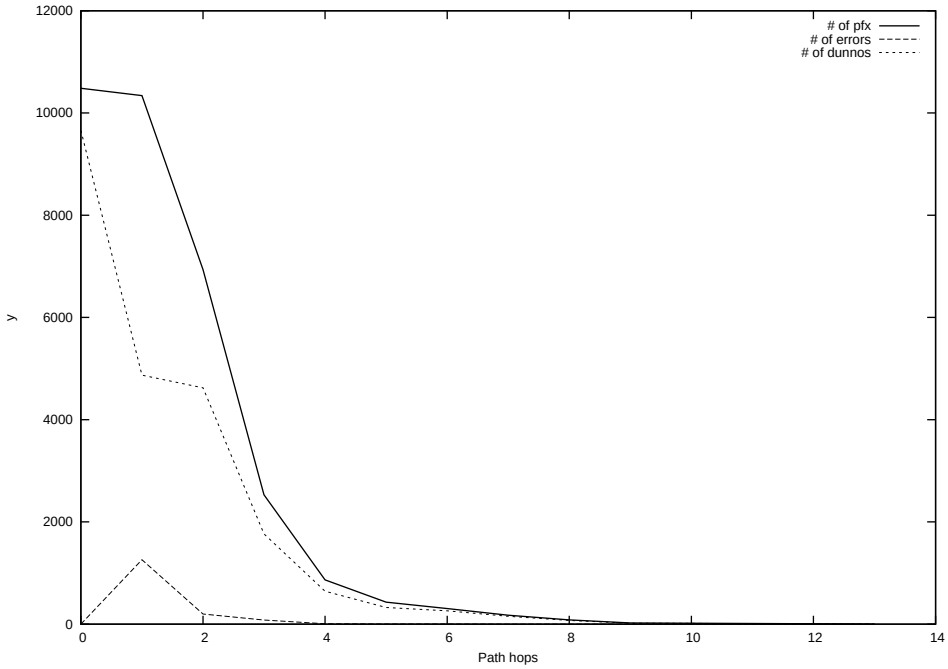
2012-09-30



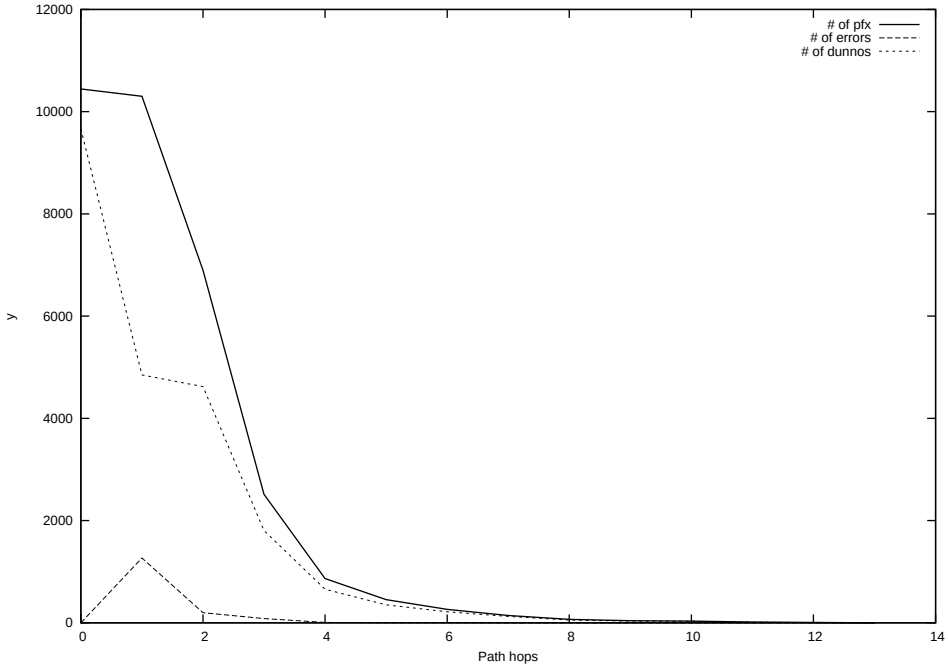
2012-10-01



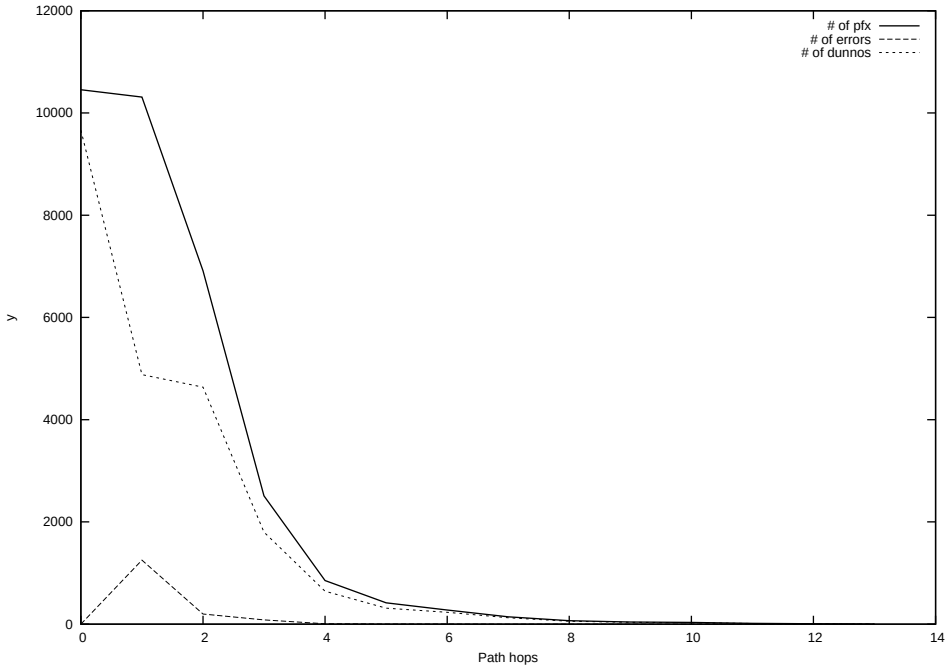
2012-10-02



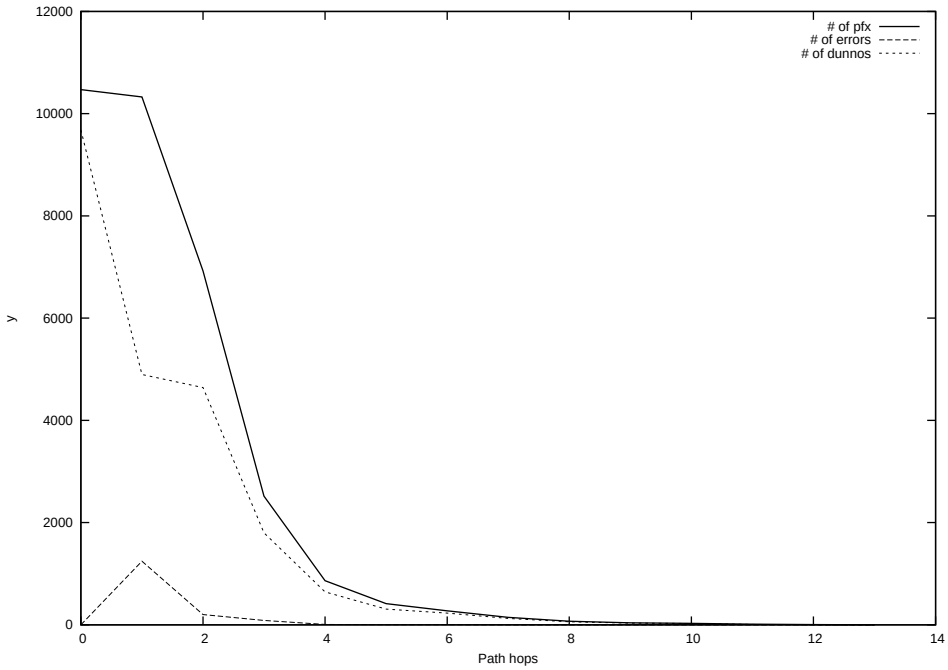
2012-10-03



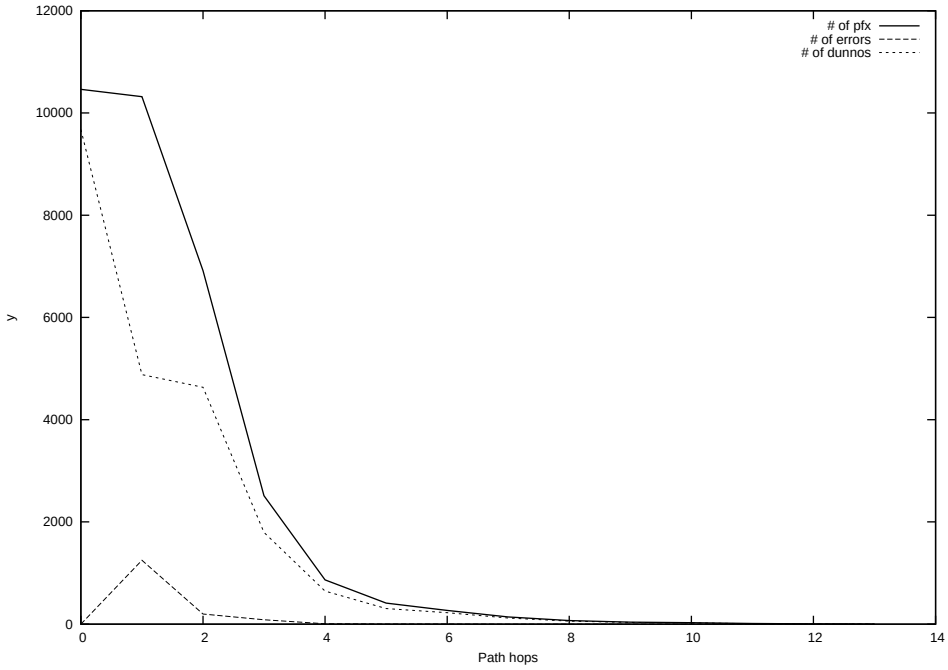
2012-10-04



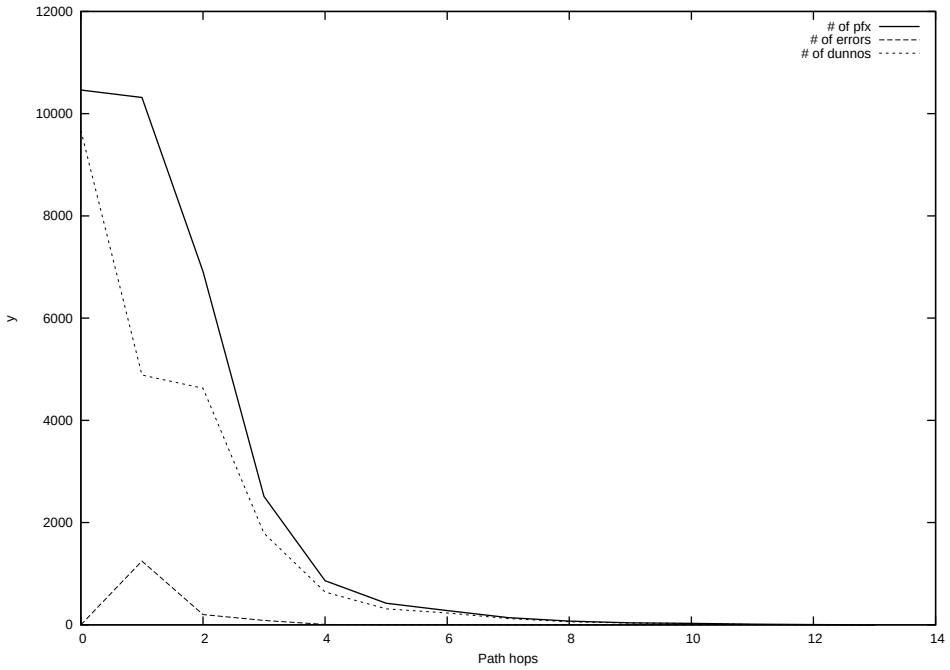
2012-10-05



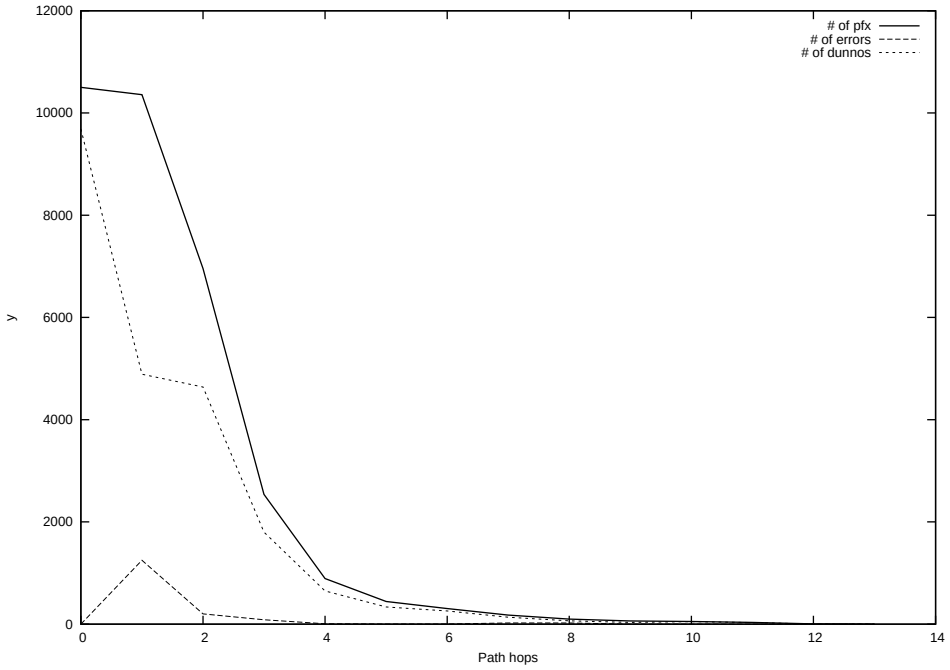
2012-10-06



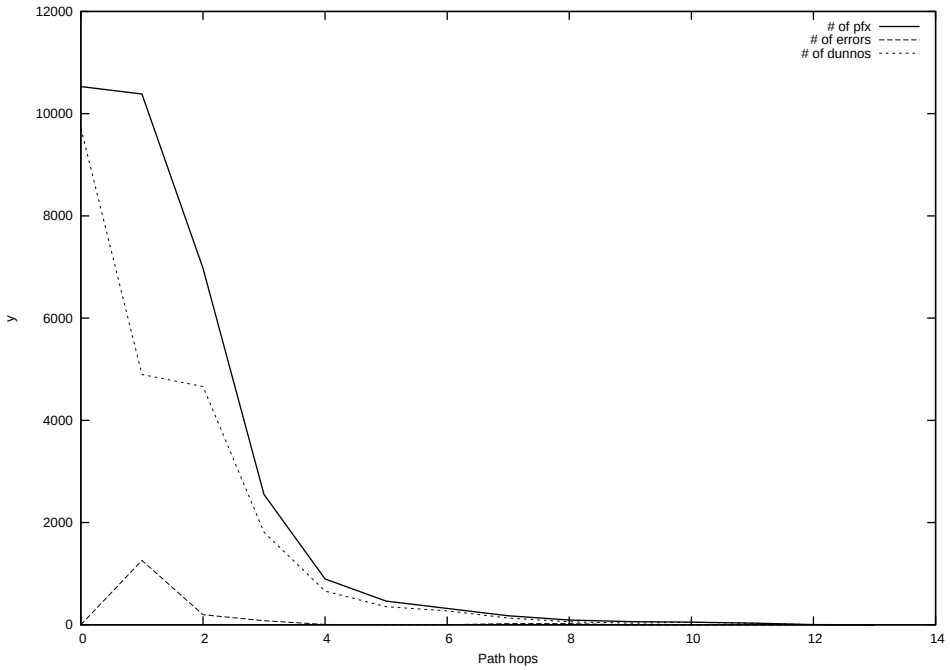
2012-10-07



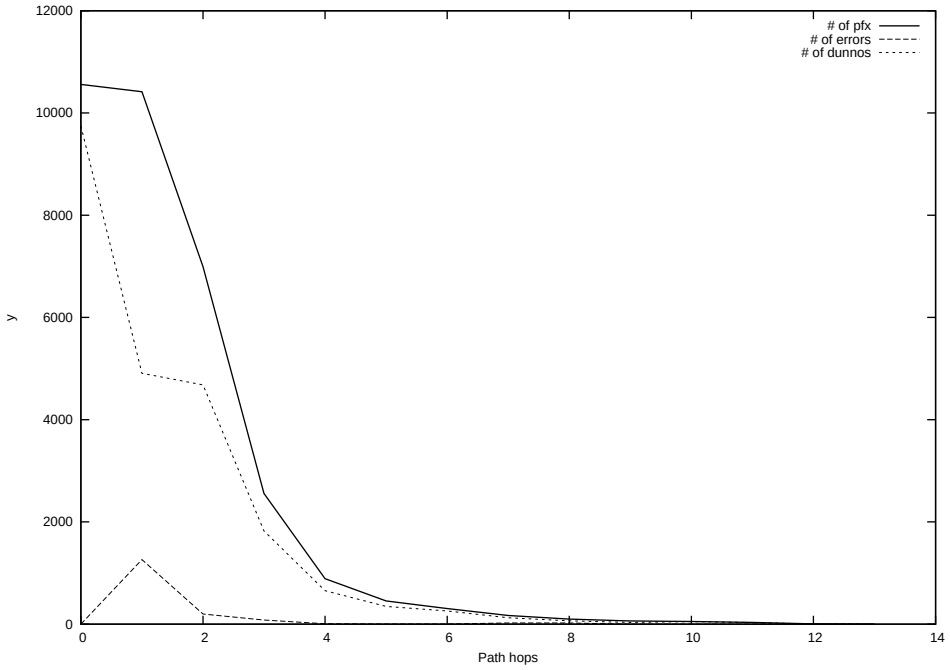
2012-10-08



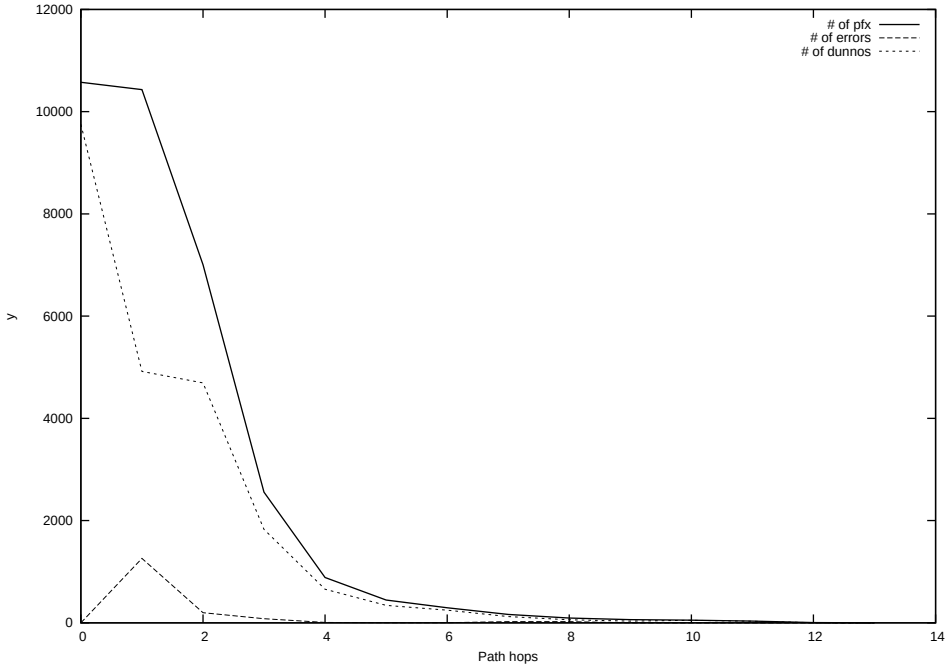
2012-10-09



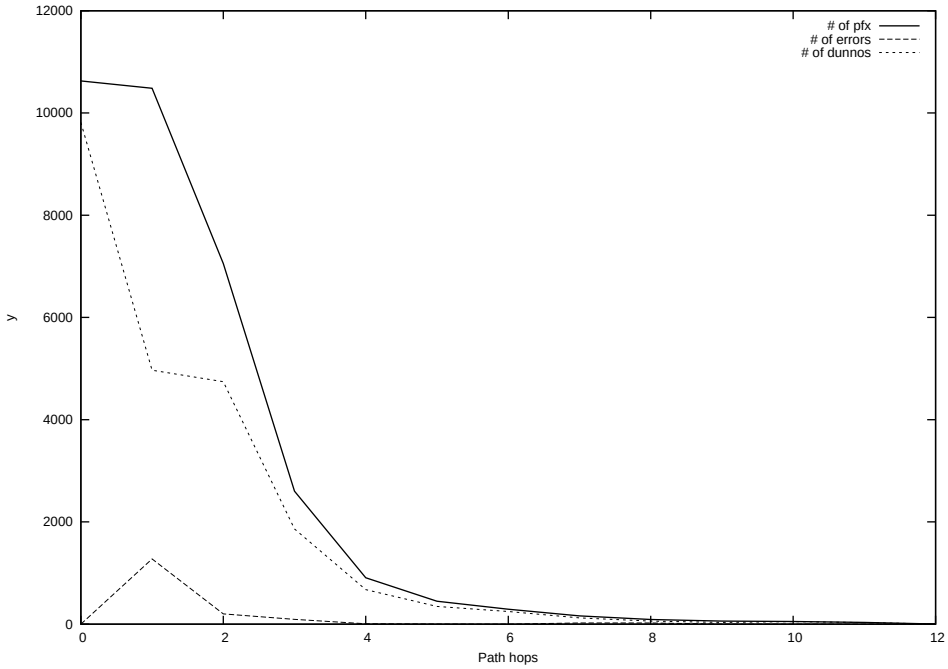
2012-10-10



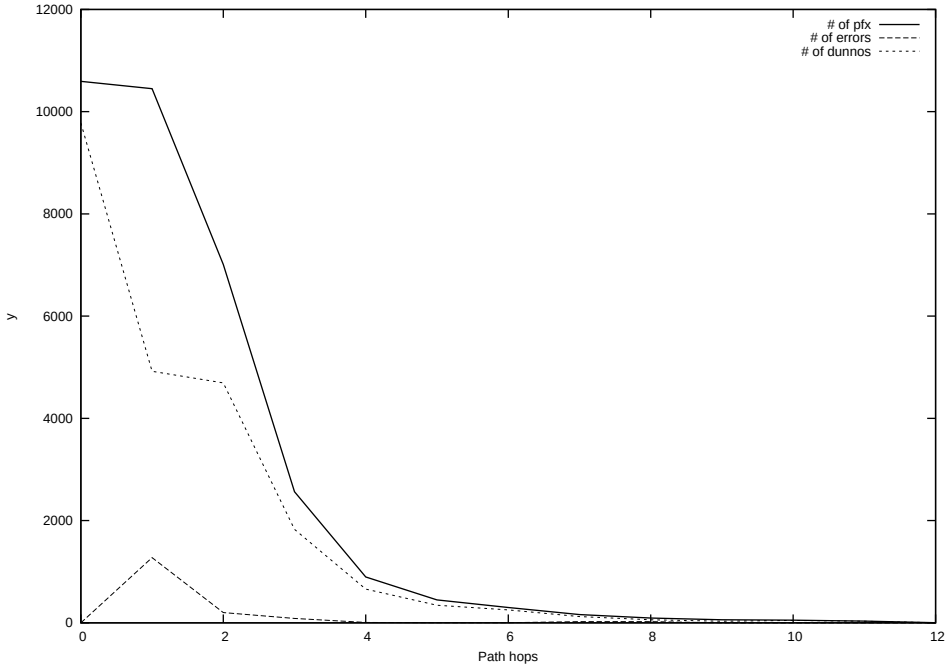
2012-10-11



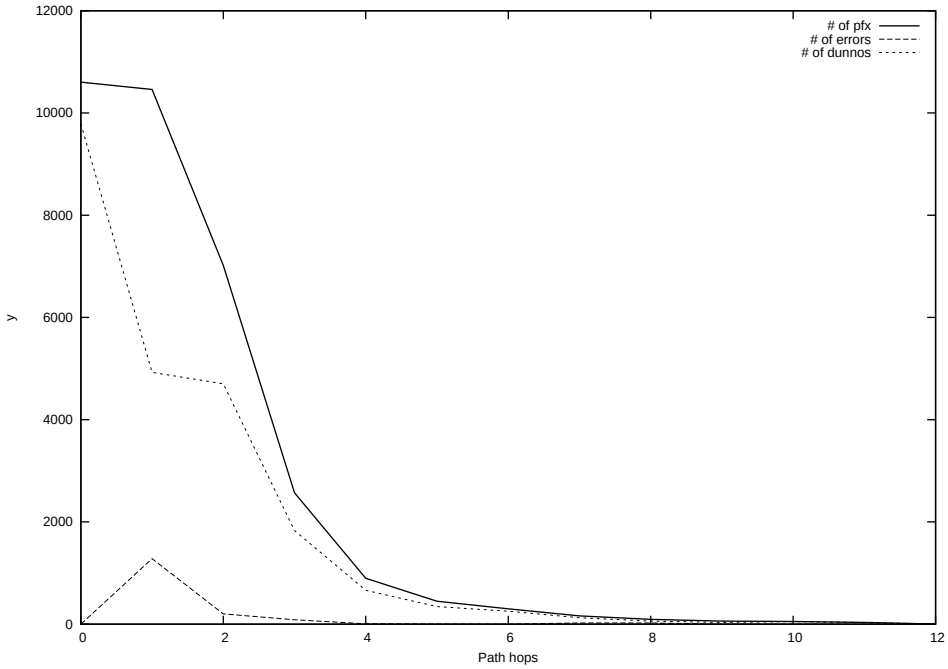
2012-10-12



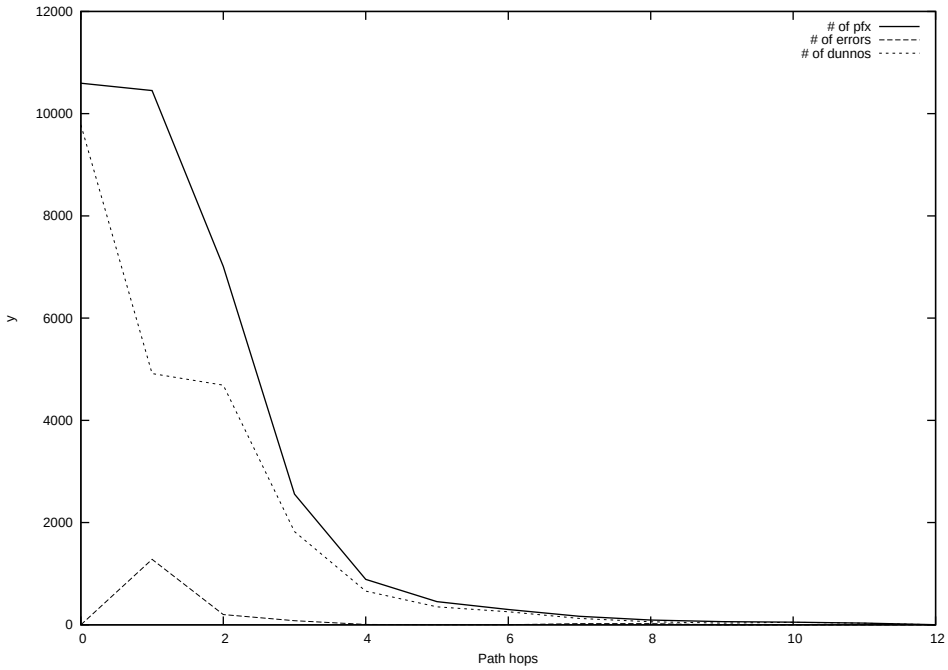
2012-10-13



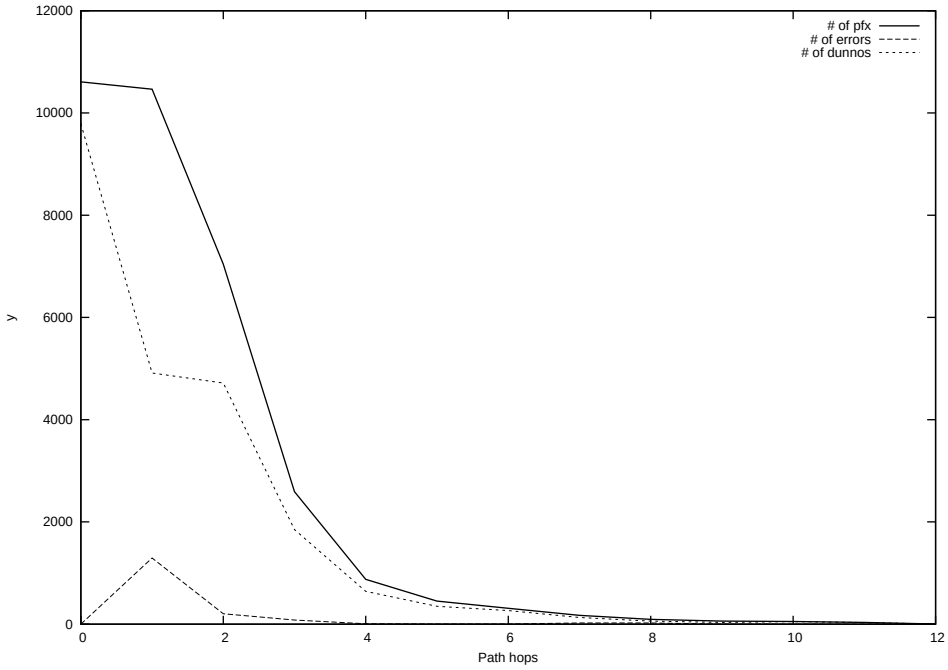
2012-10-14



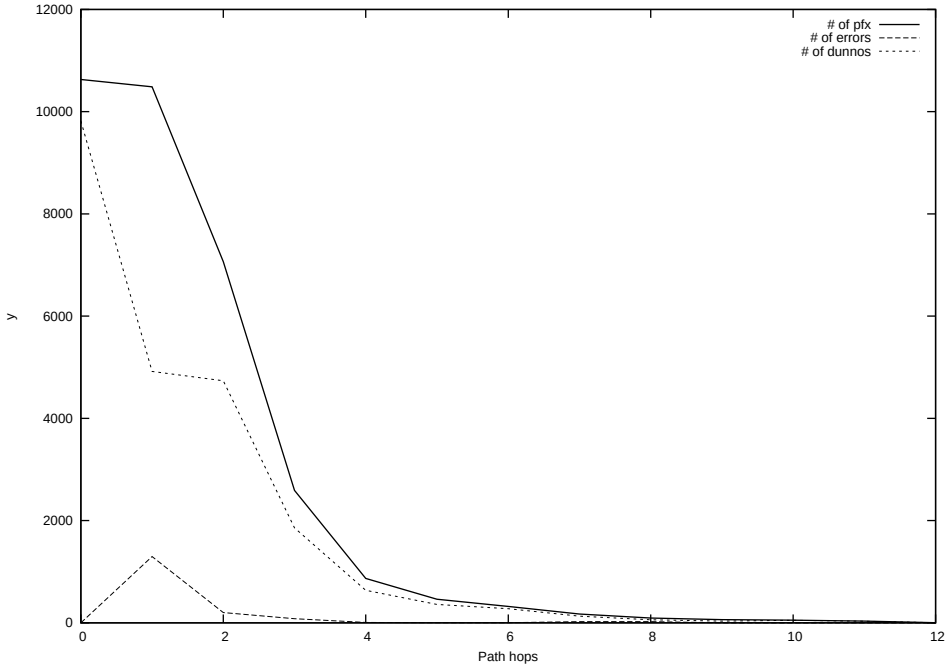
2012-10-15



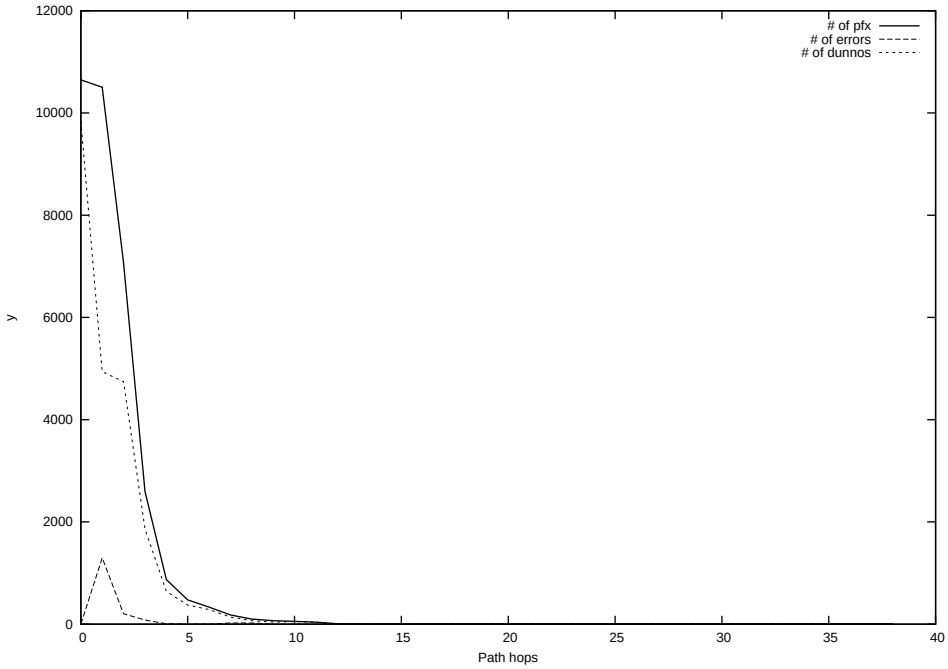
2012-10-16



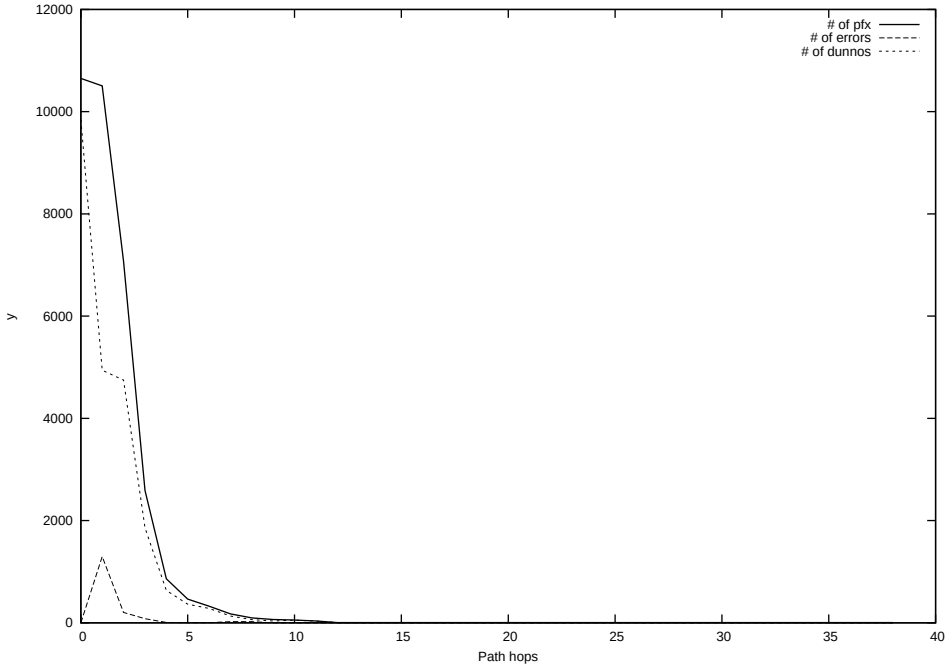
2012-10-18



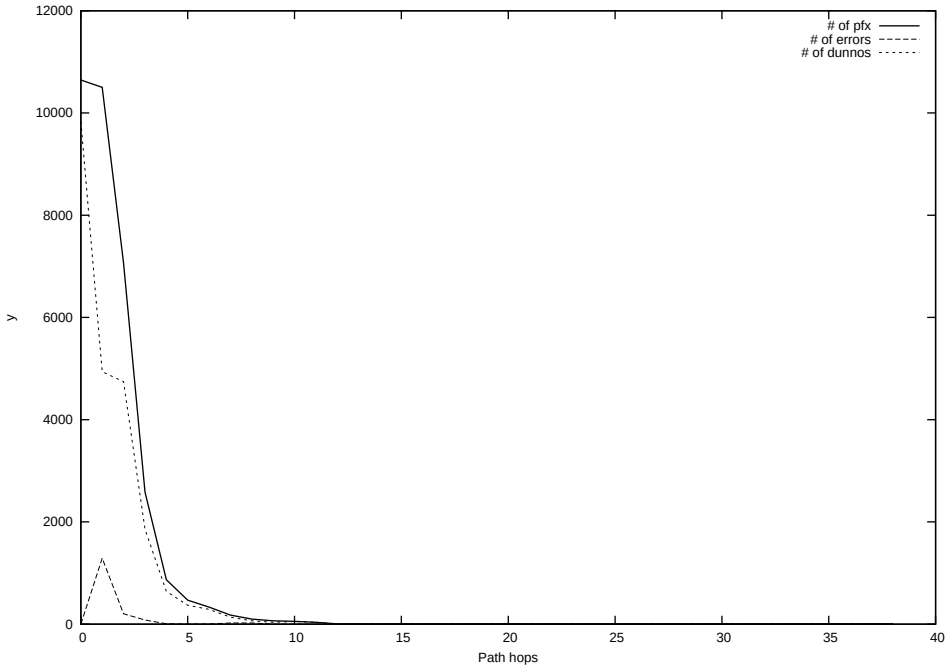
2012-10-19



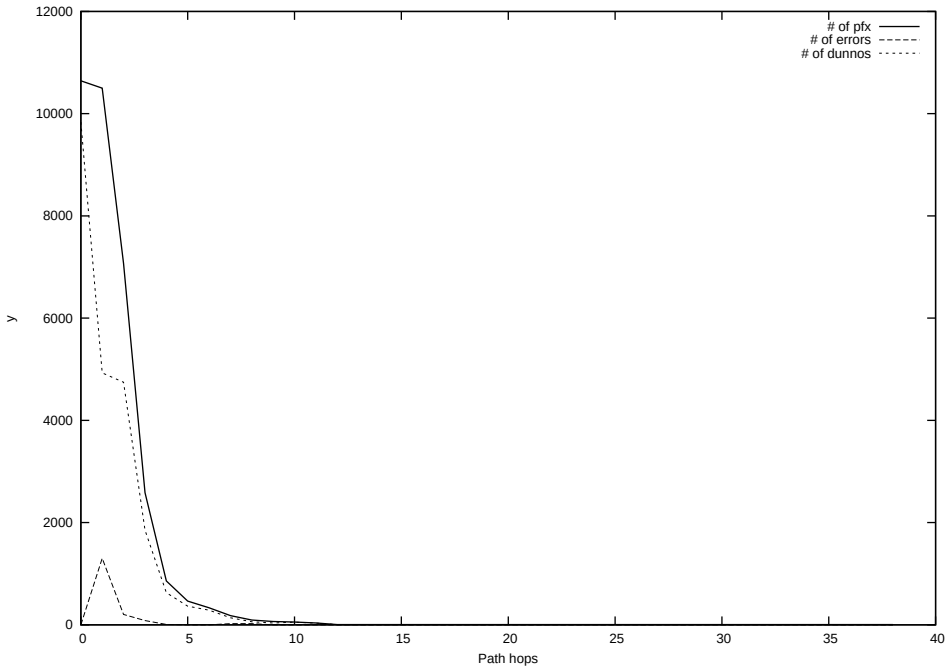
2012-10-20



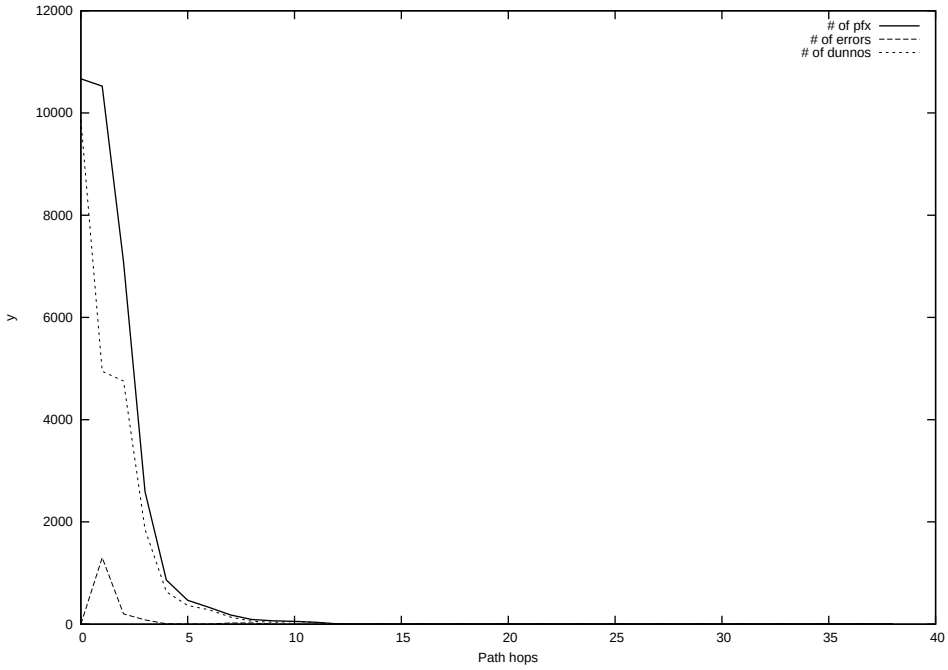
2012-10-21



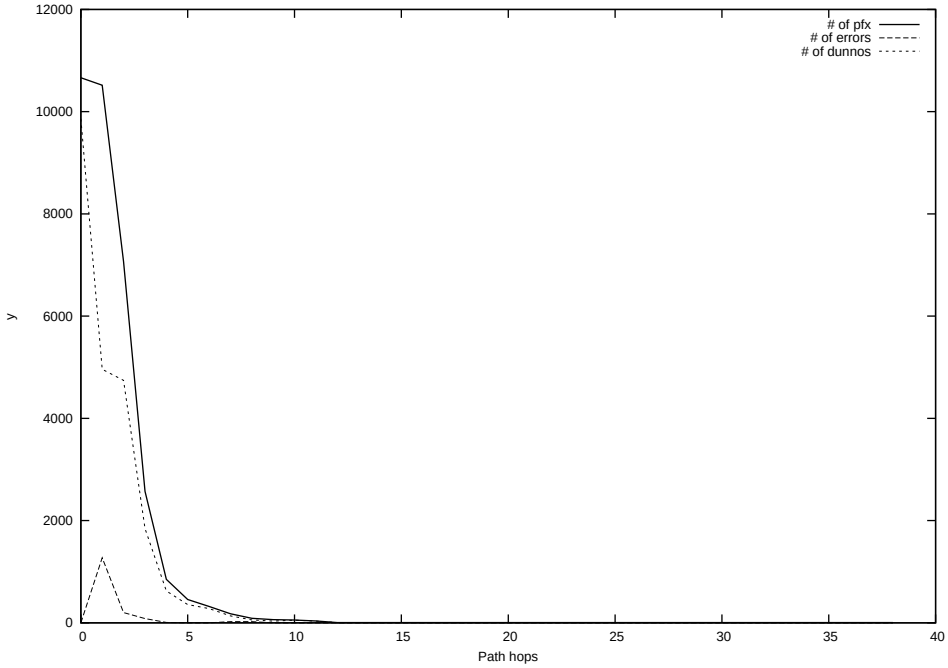
2012-10-22



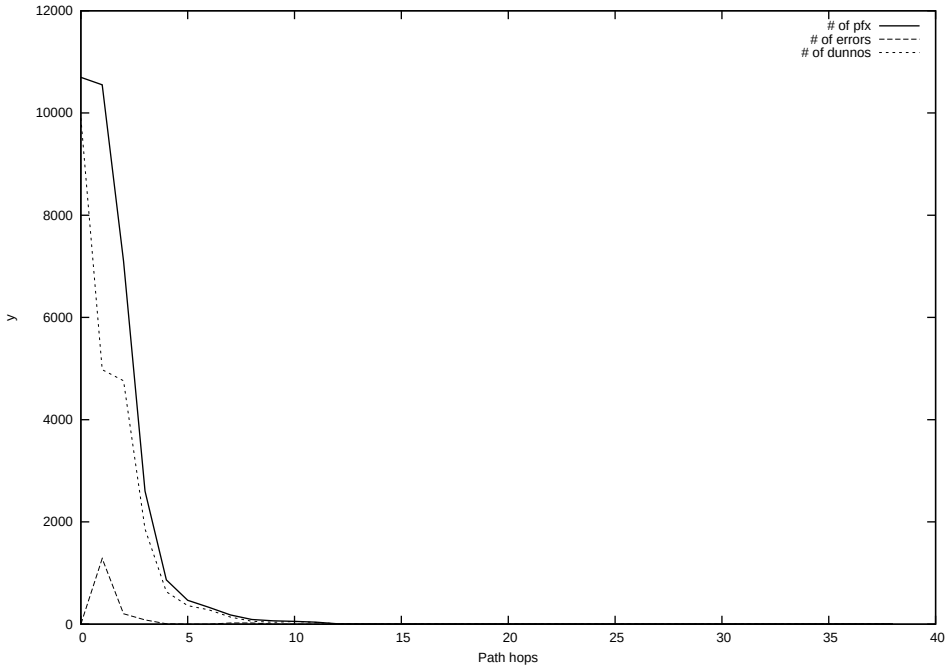
2012-10-23



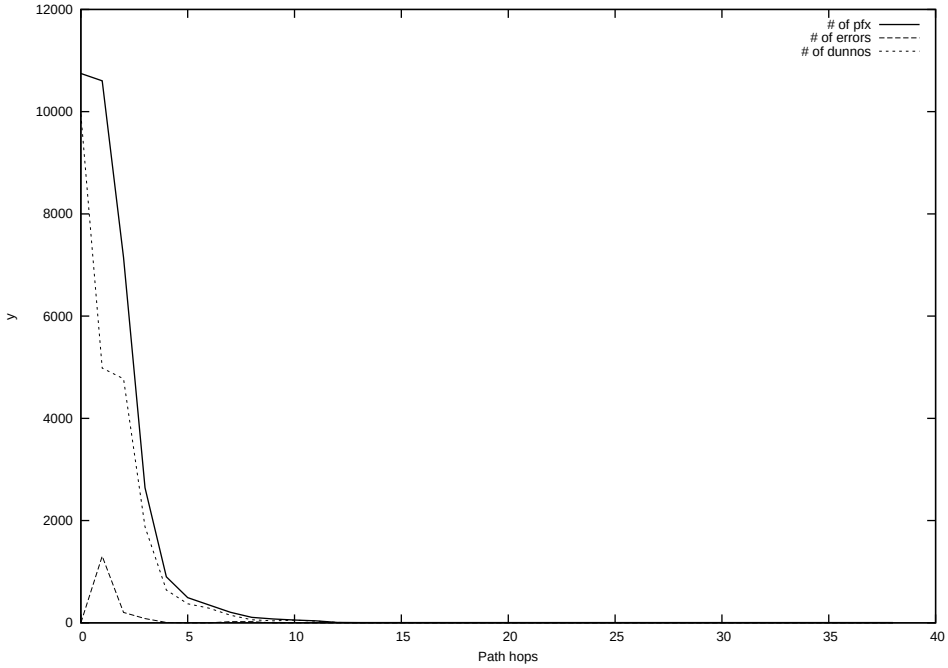
2012-10-24



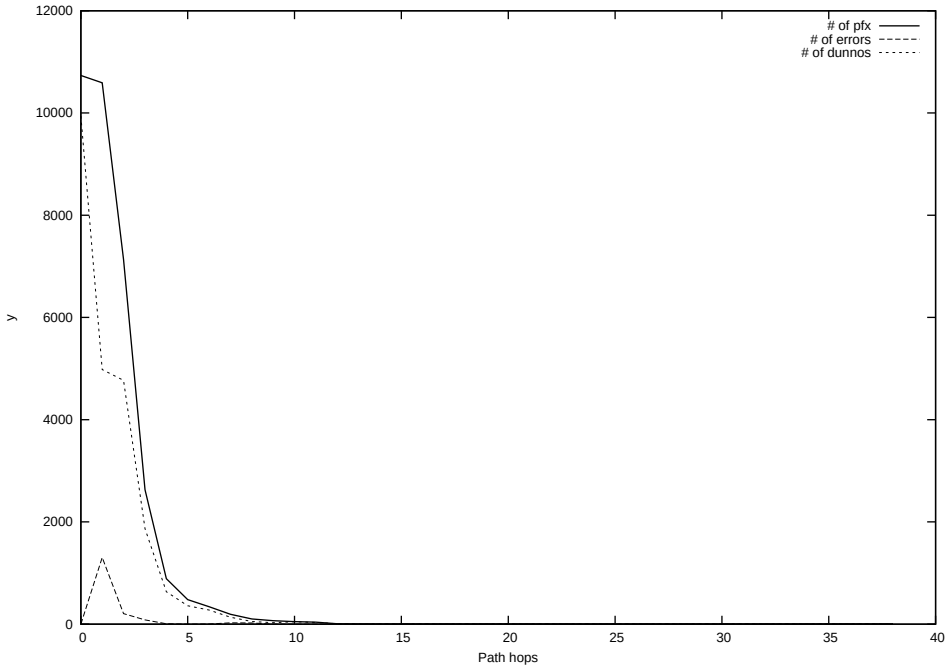
2012-10-25



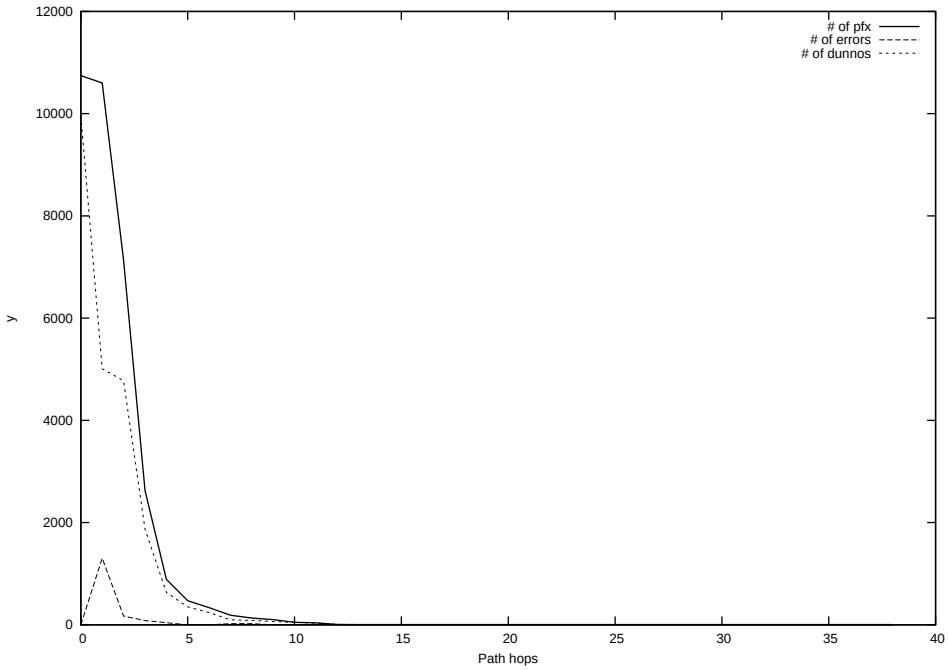
2012-10-26



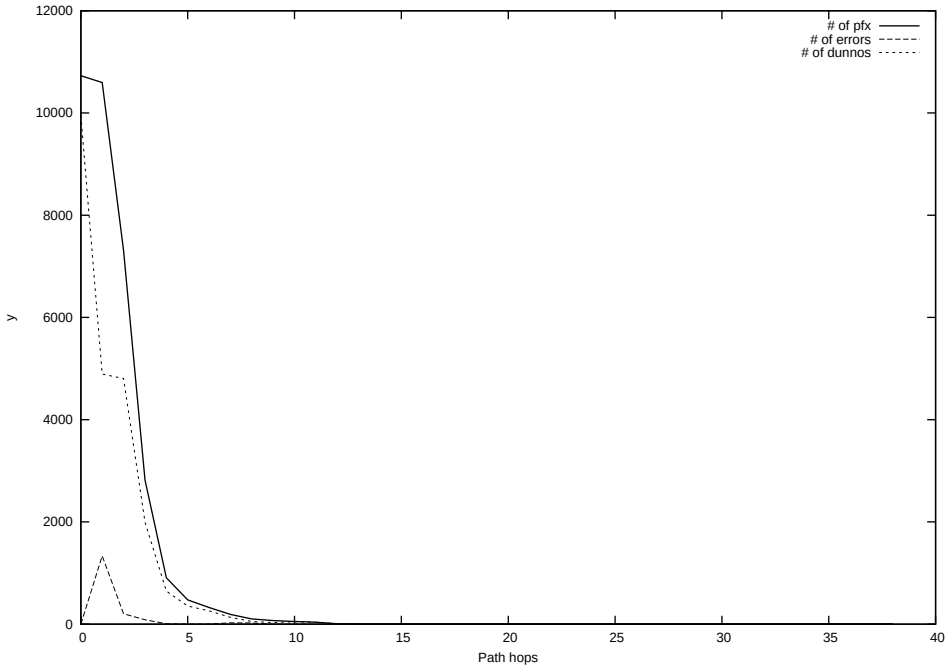
2012-10-27



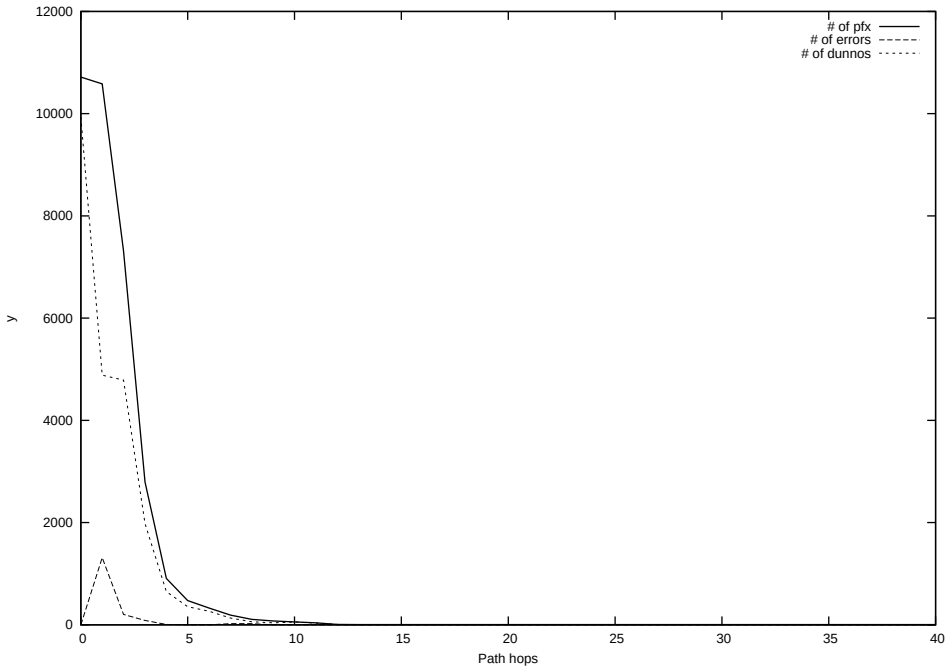
2012-10-28



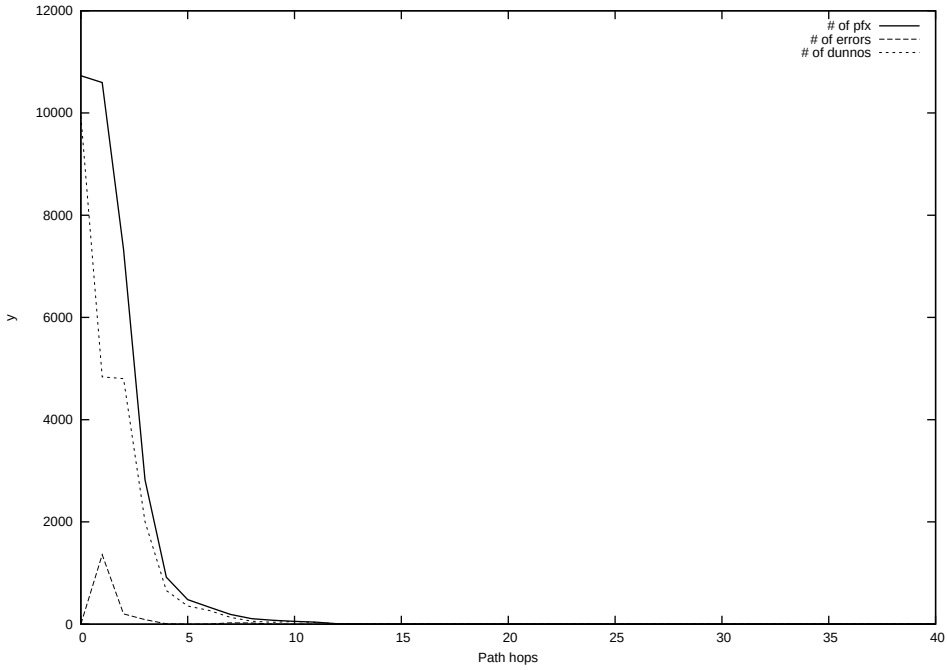
2012-10-29



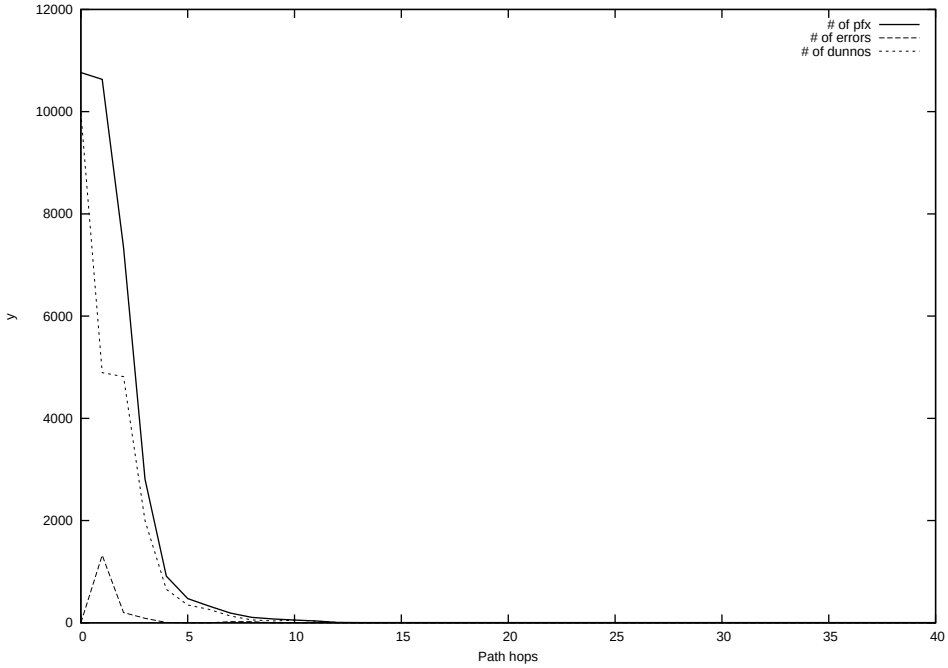
2012-10-30



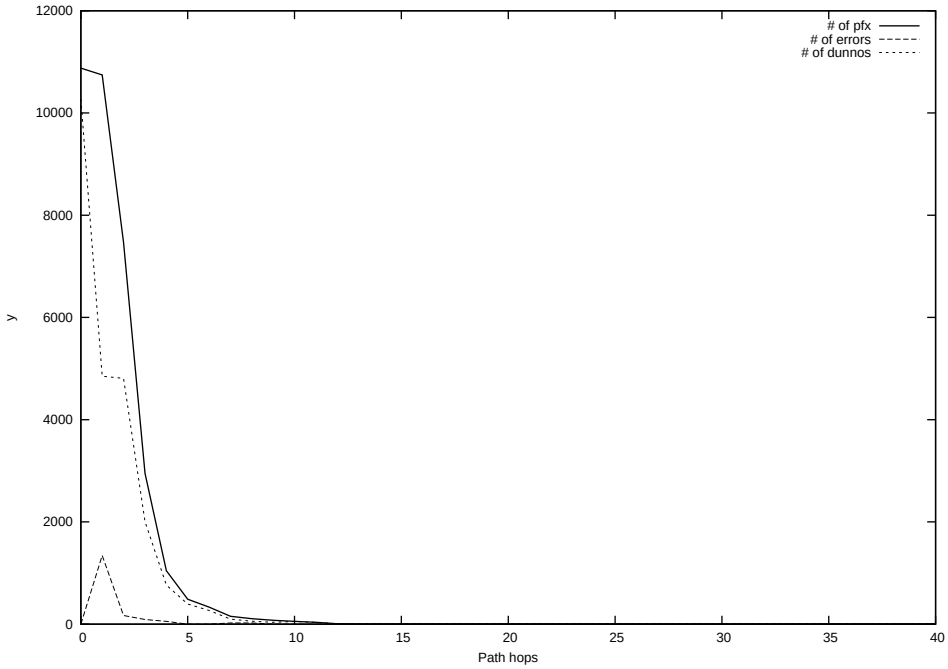
2012-10-31



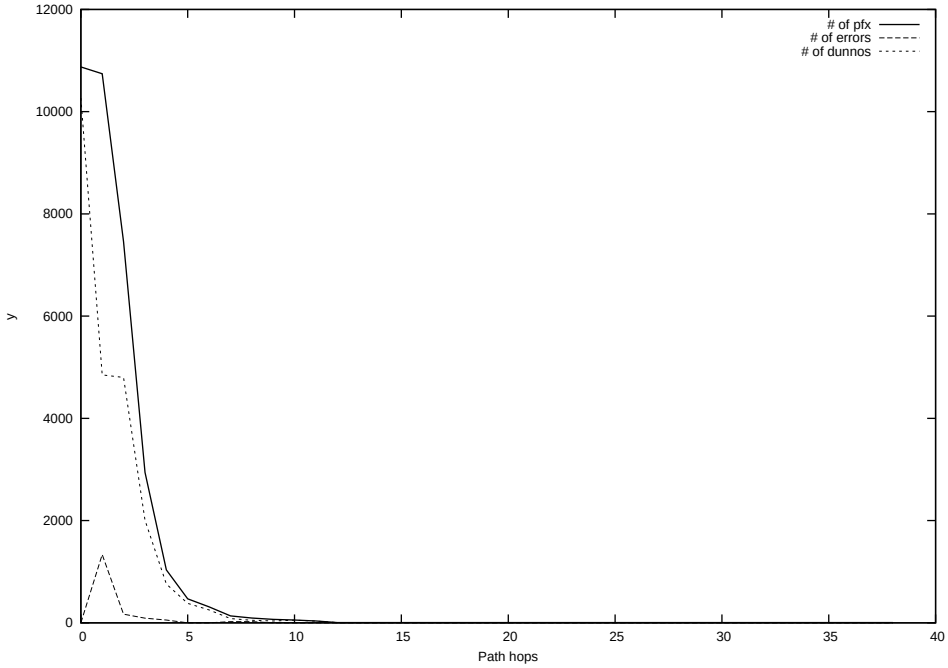
2012-11-01



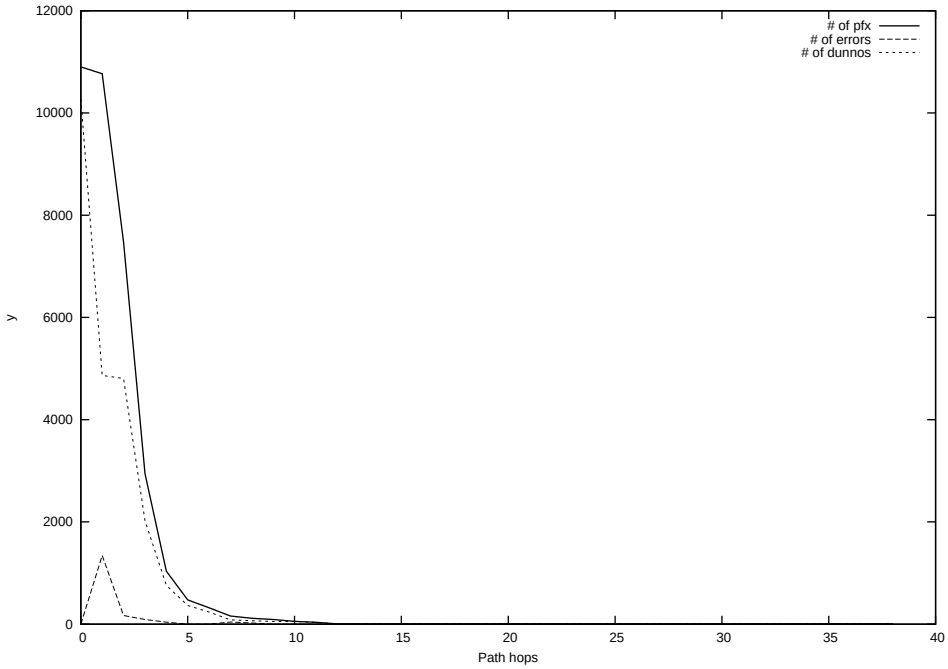
2012-11-02



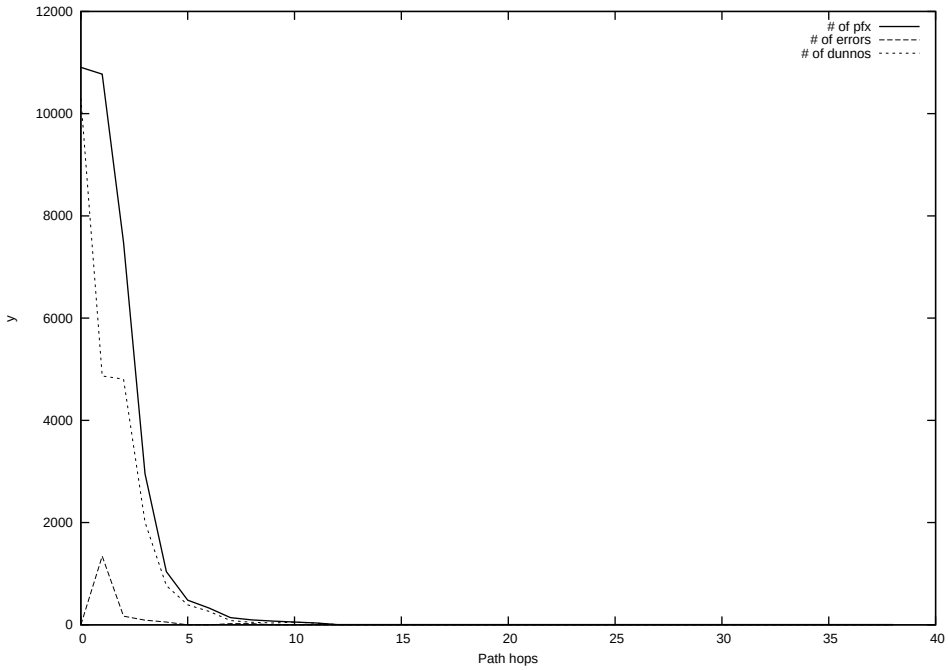
2012-11-03



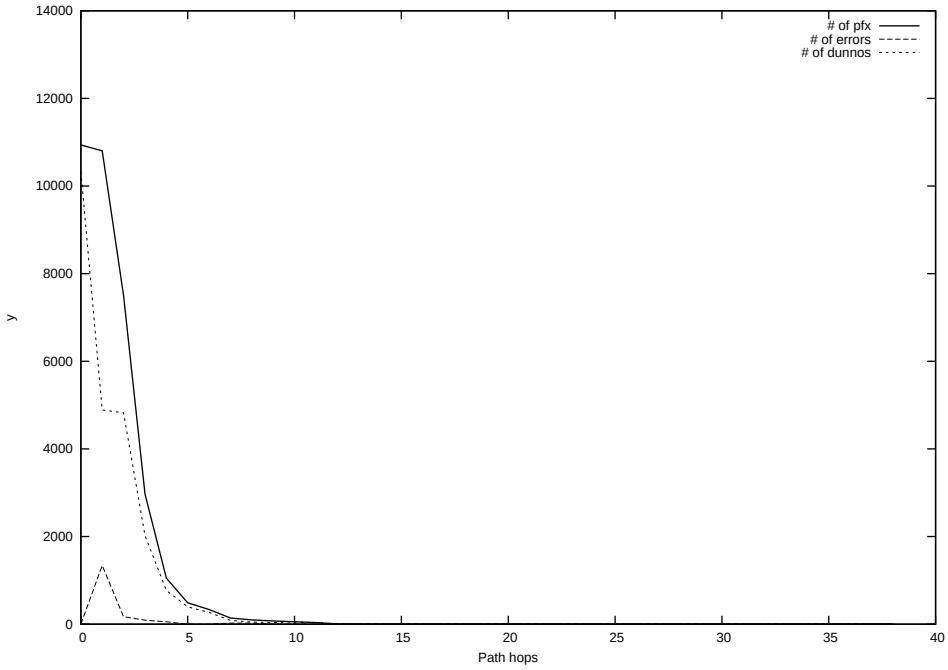
2012-11-04



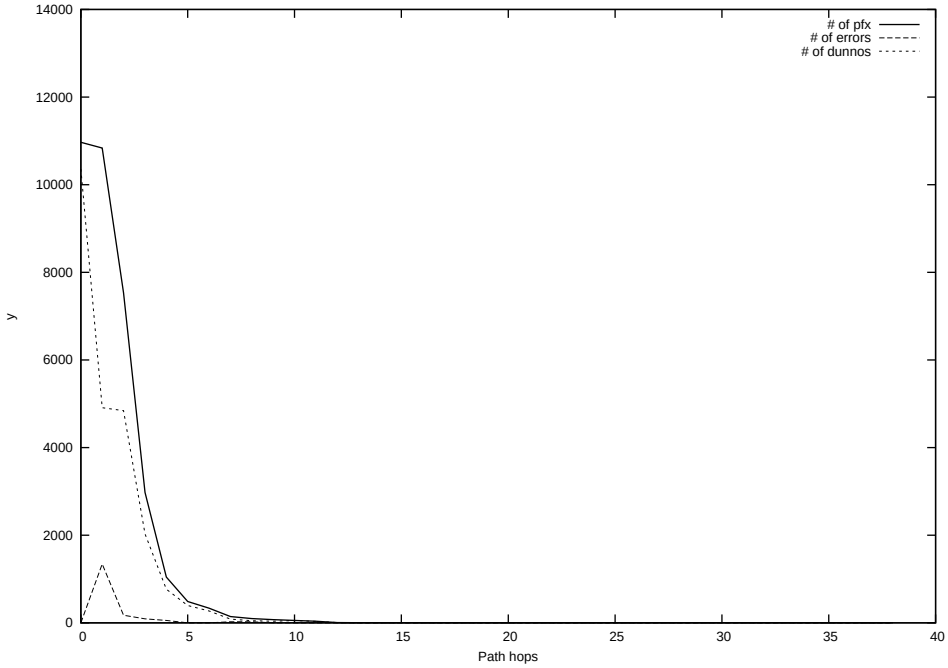
2012-11-05



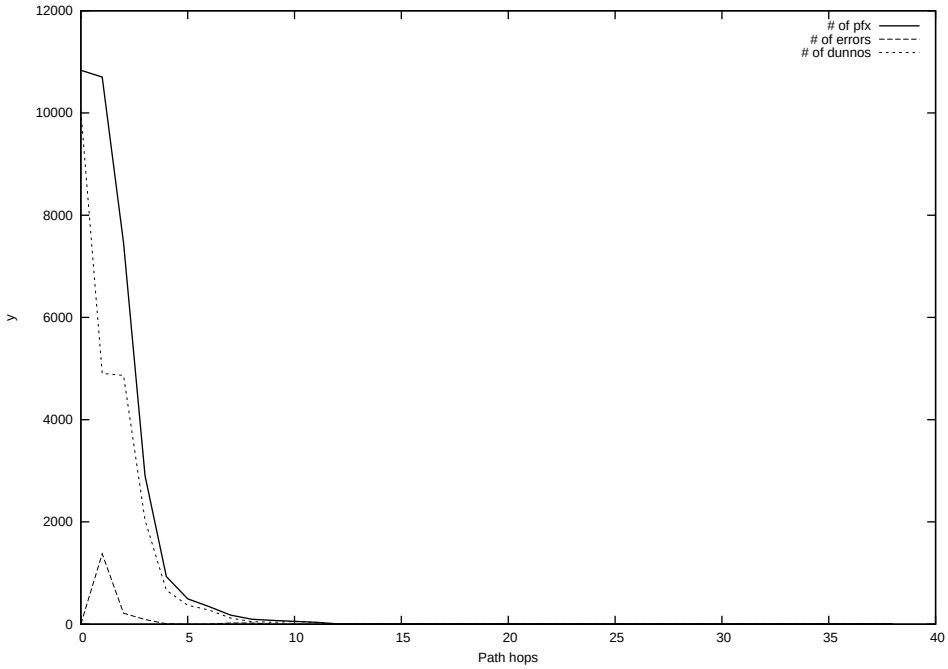
2012-11-06



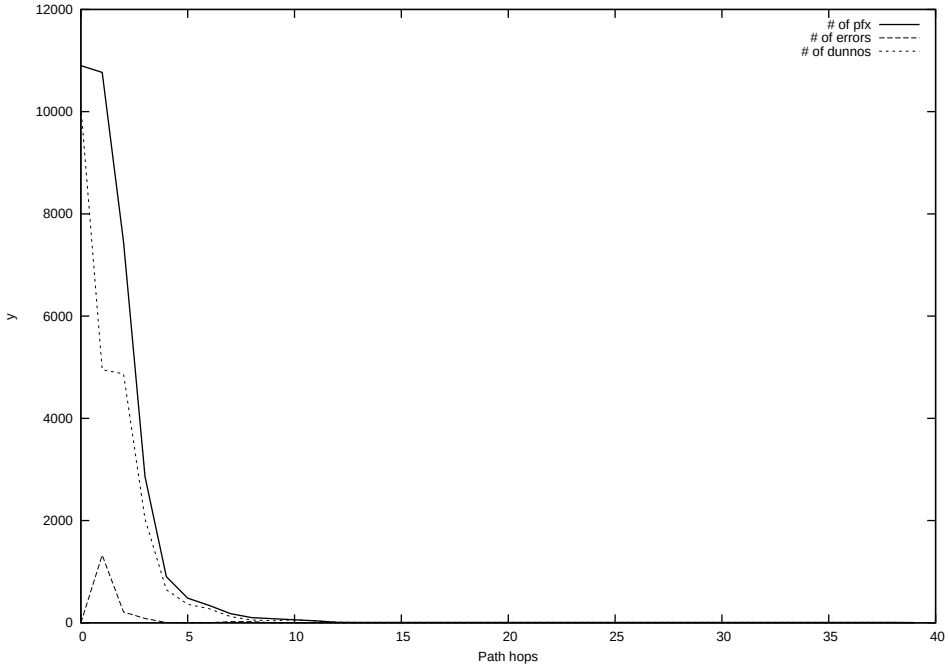
2012-11-07



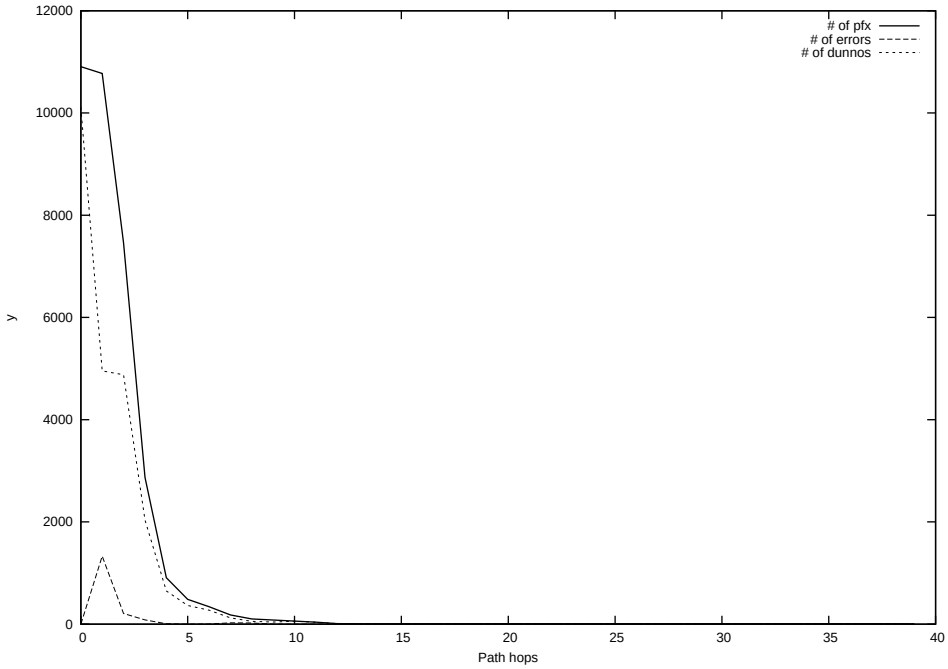
2012-11-08



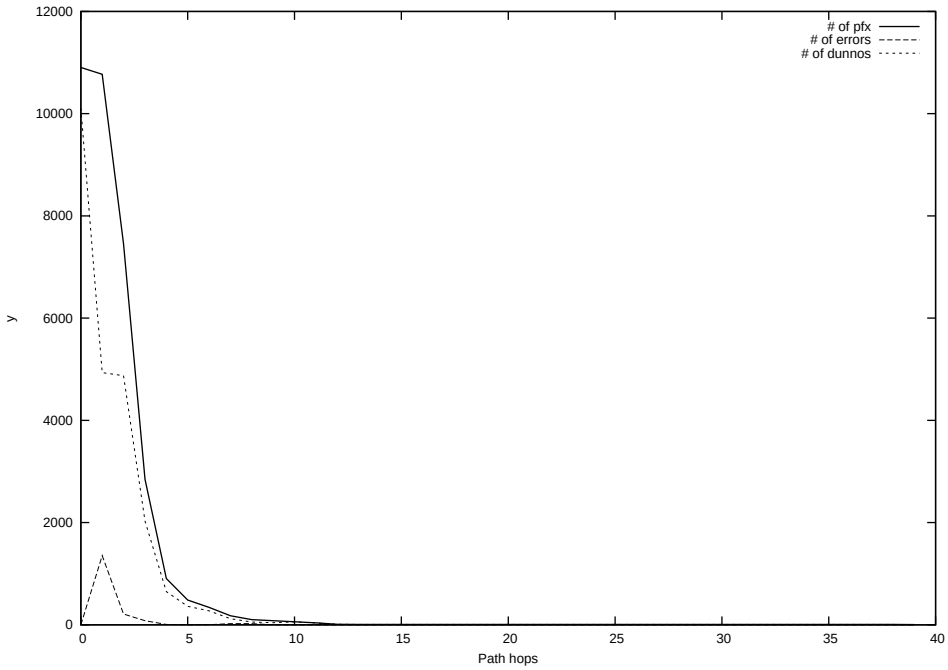
2012-11-09



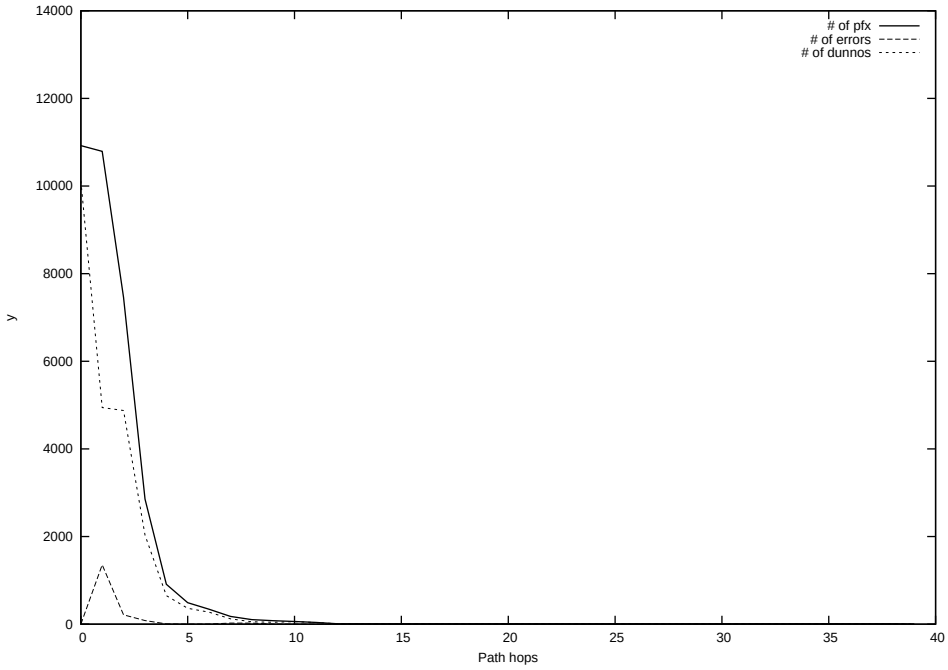
2012-11-10



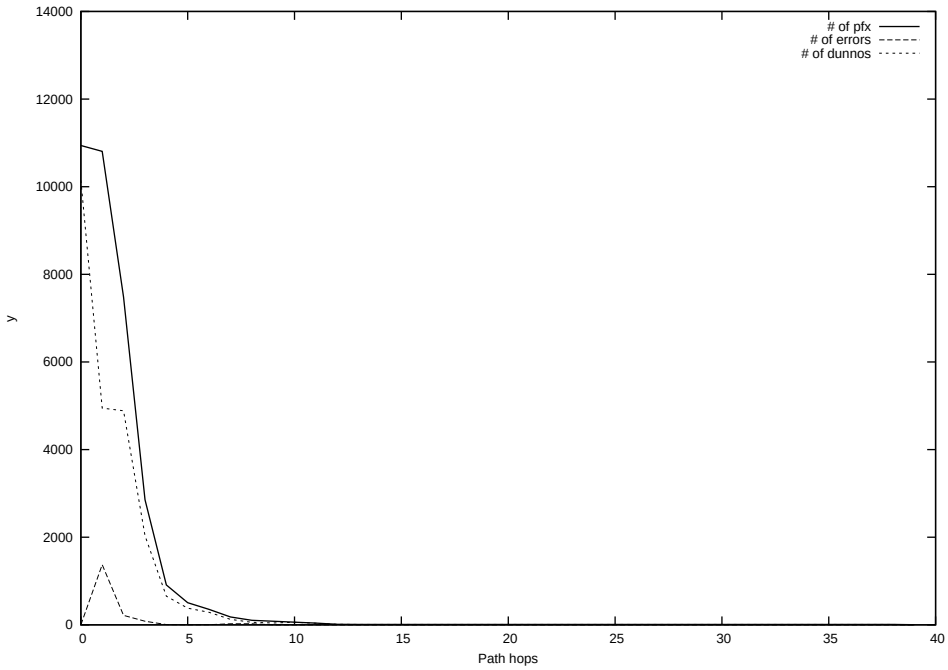
2012-11-11



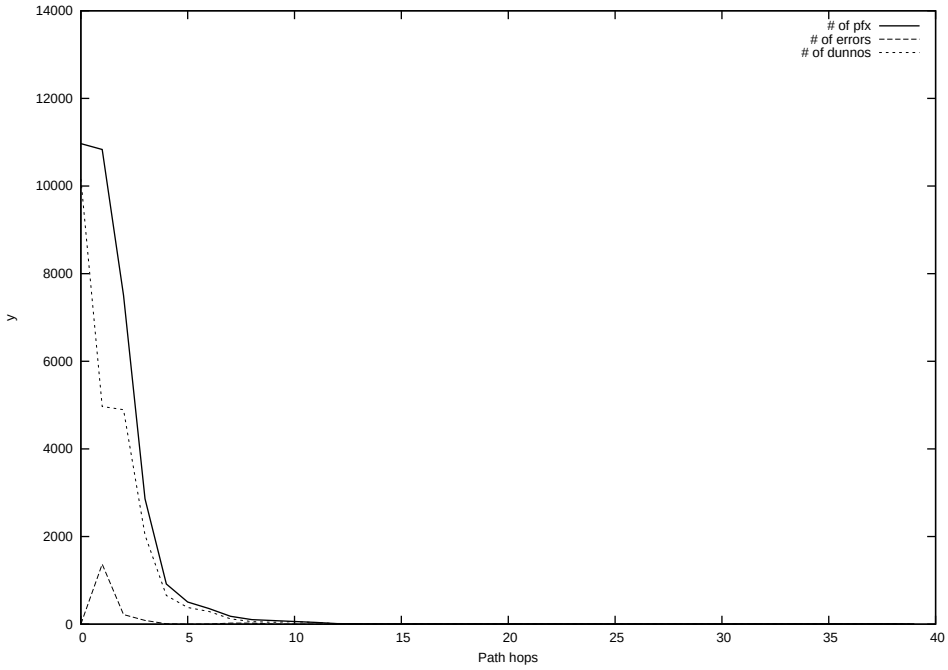
2012-11-12



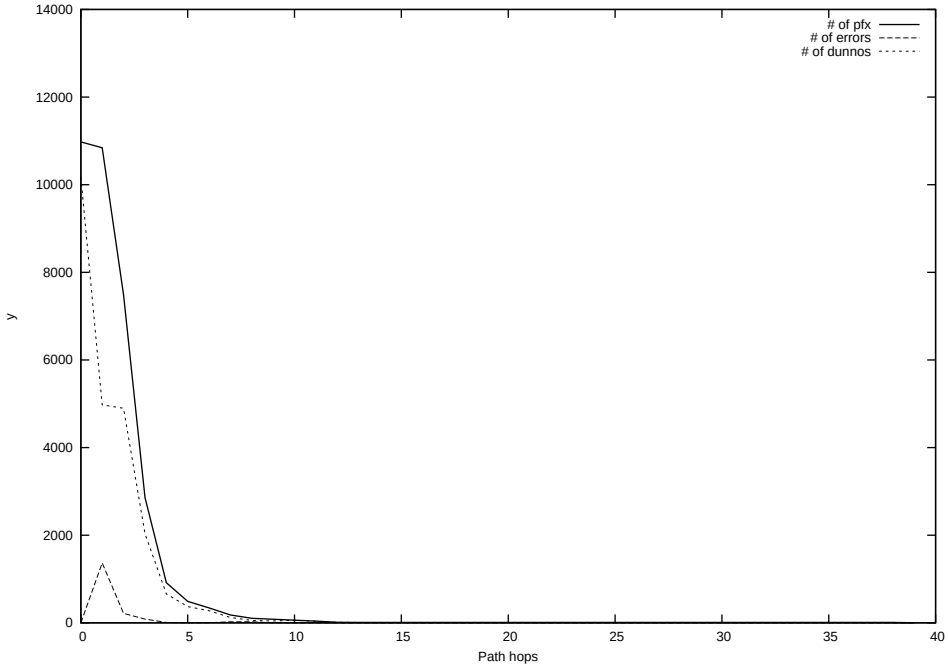
2012-11-13



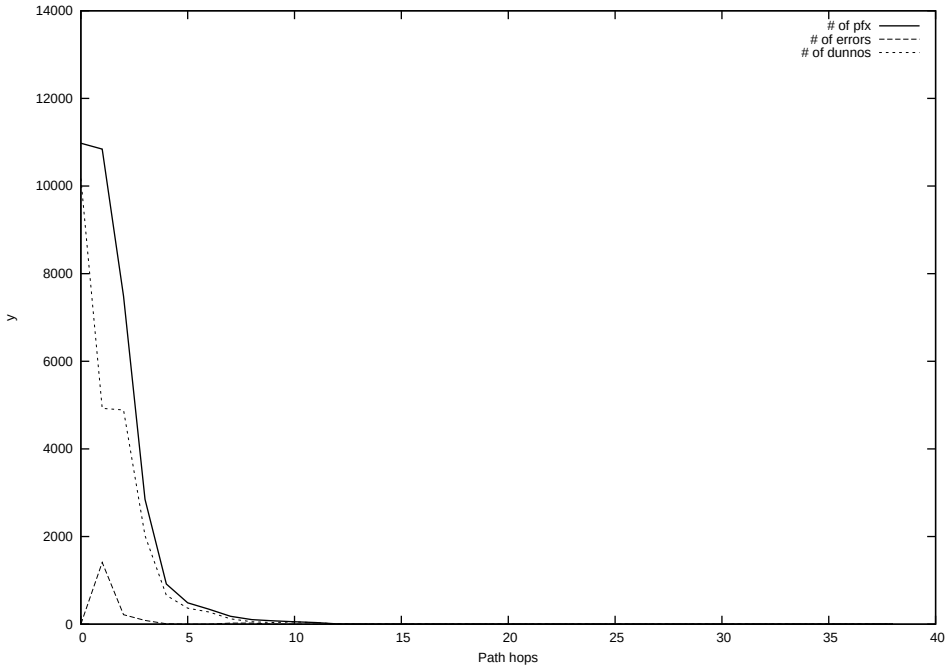
2012-11-14



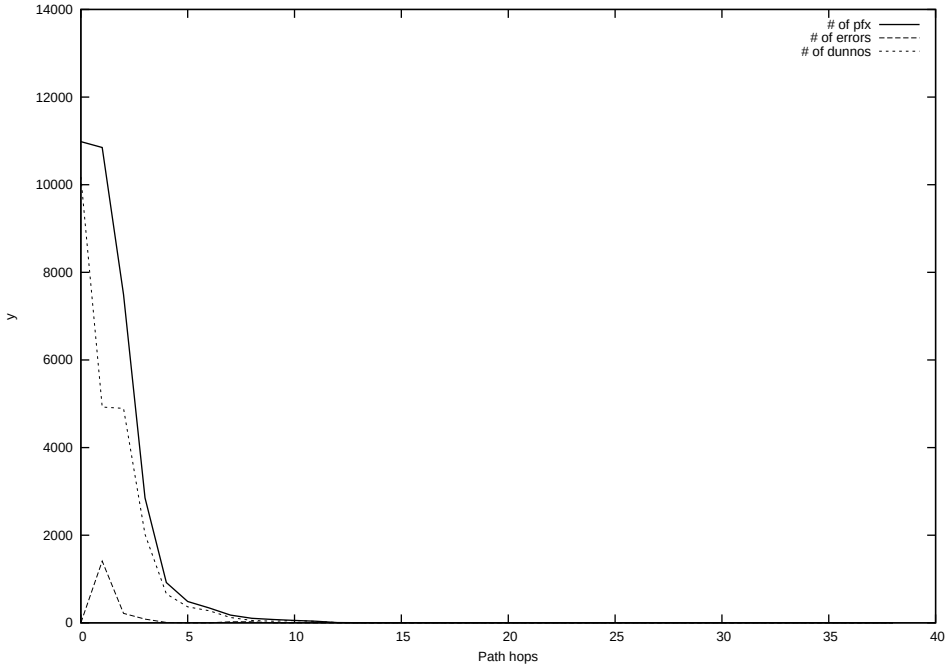
2012-11-15



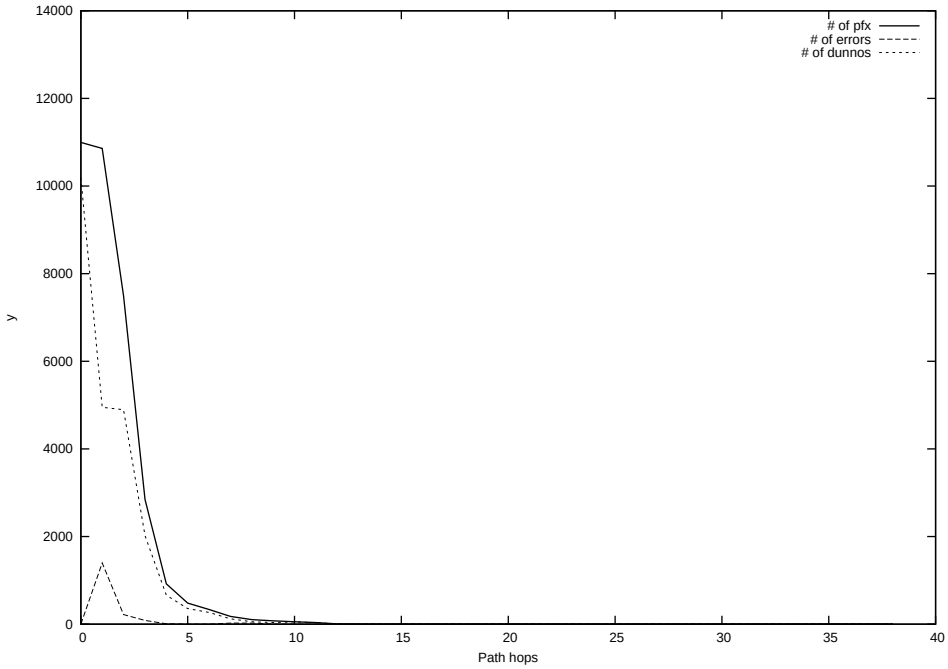
2012-11-16



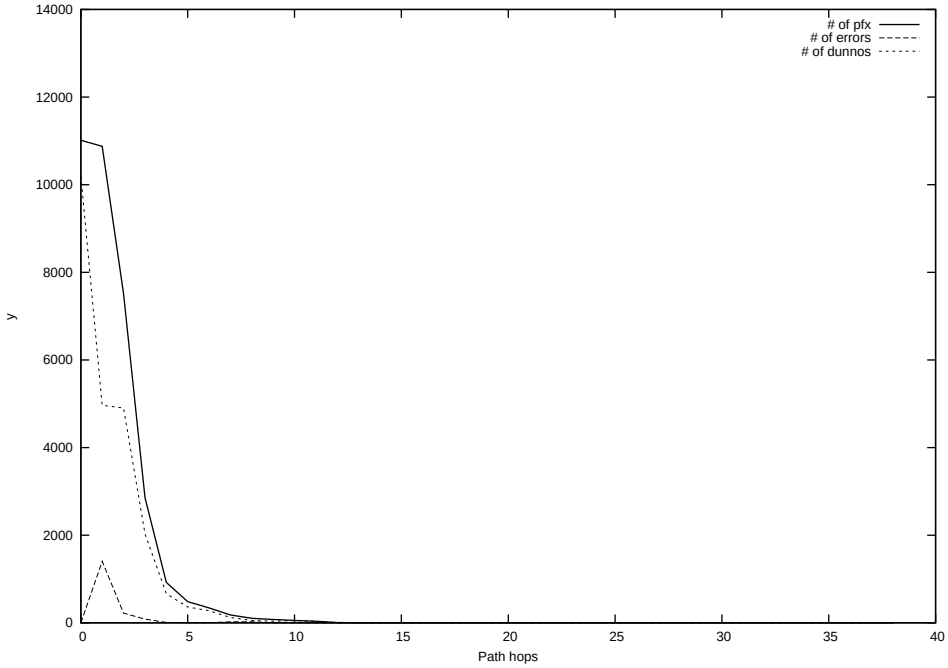
2012-11-17



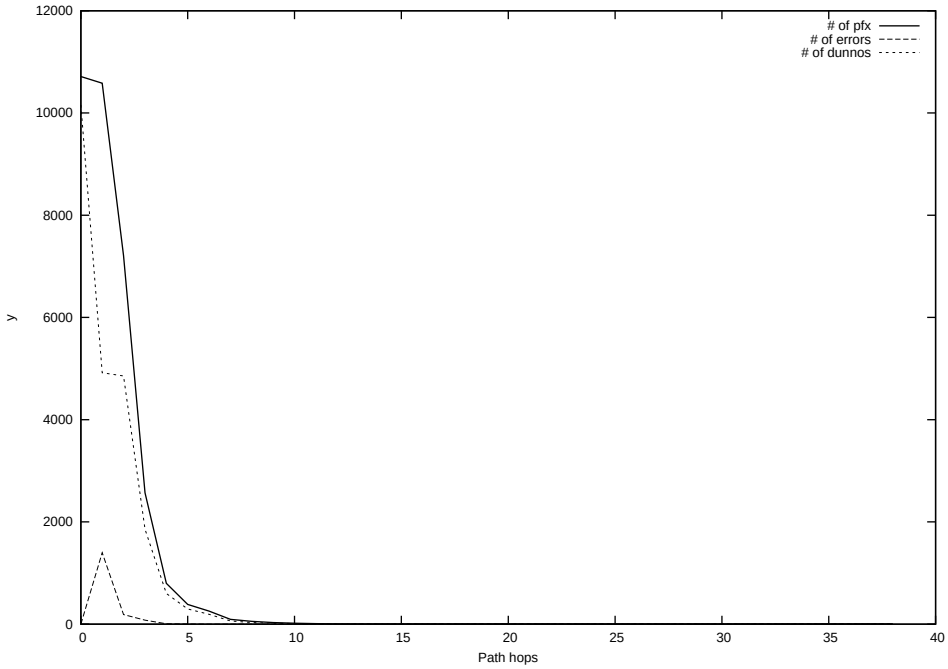
2012-11-18



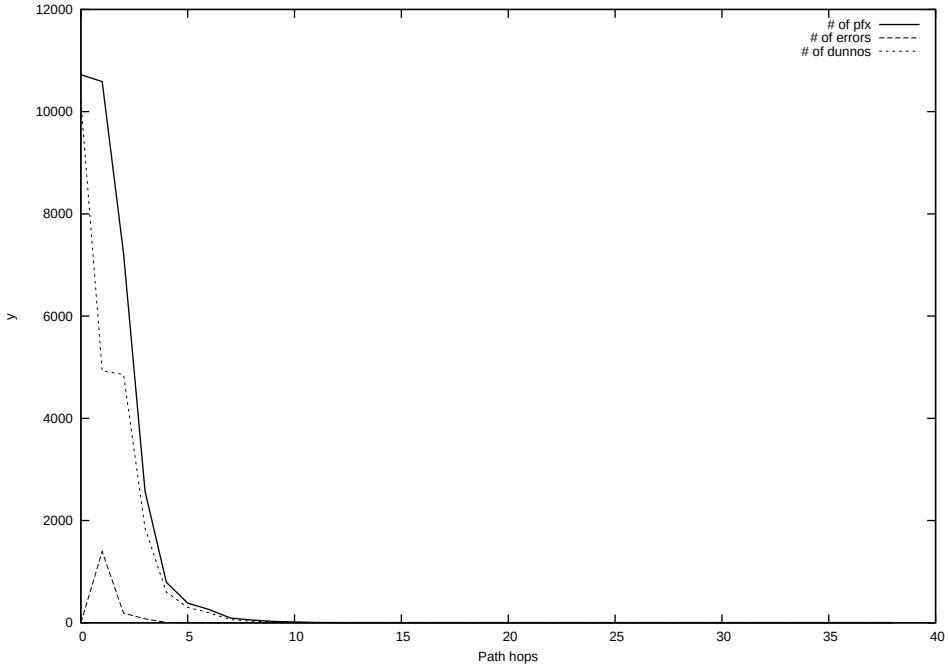
2012-11-19



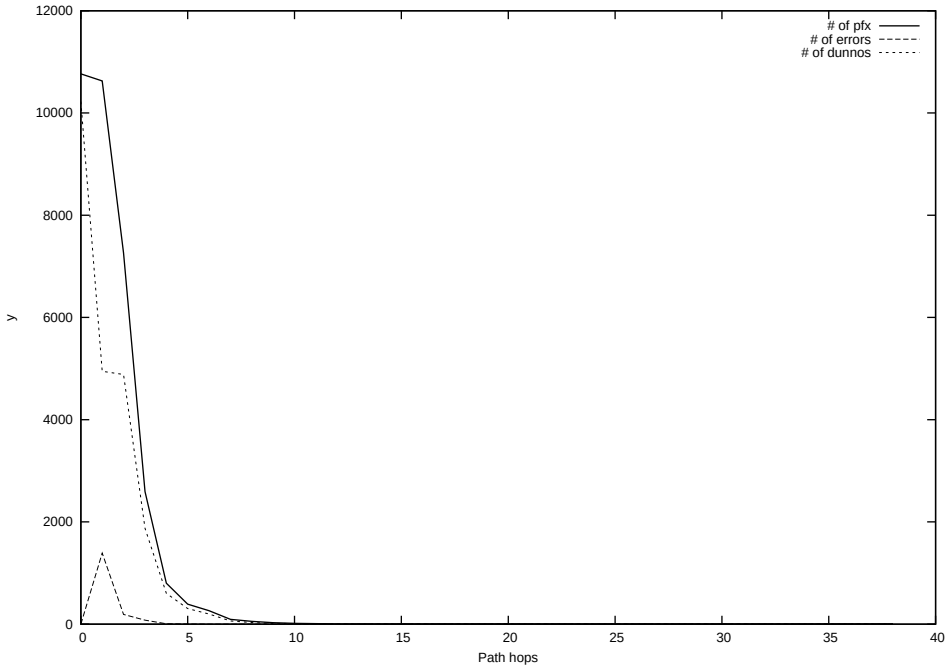
2012-11-20



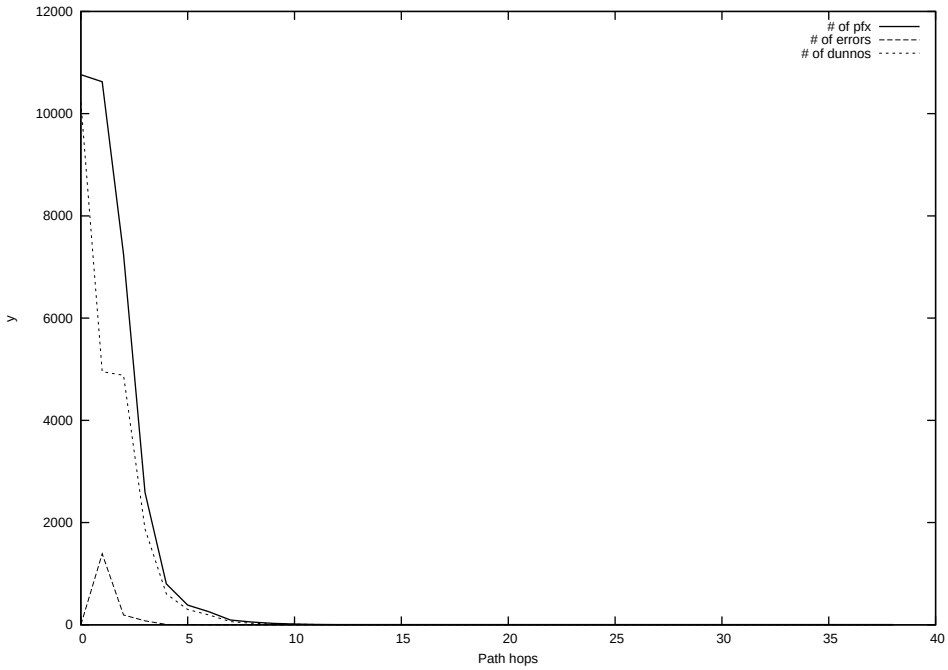
2012-11-21



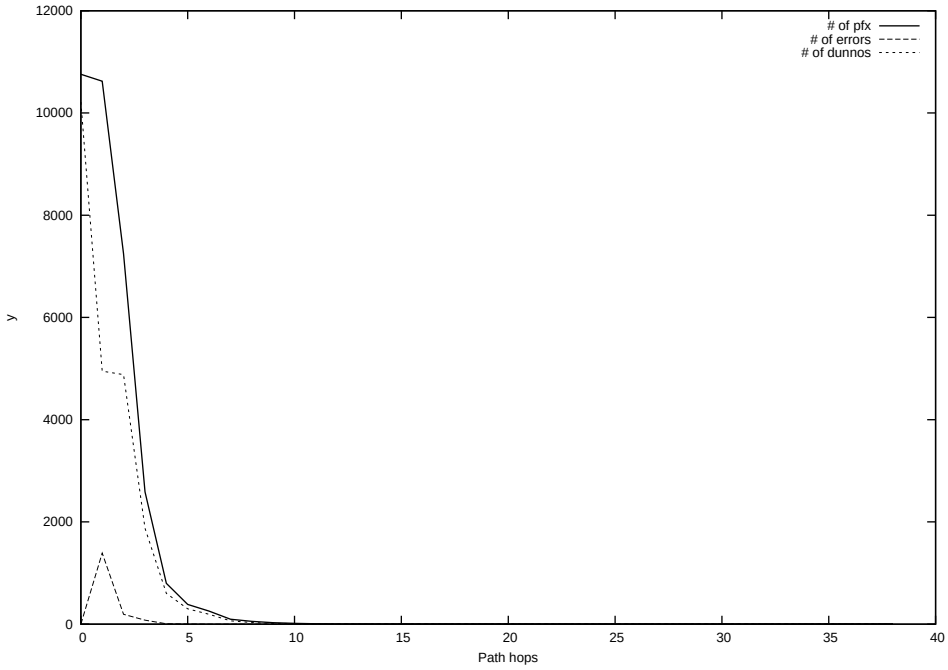
2012-11-22



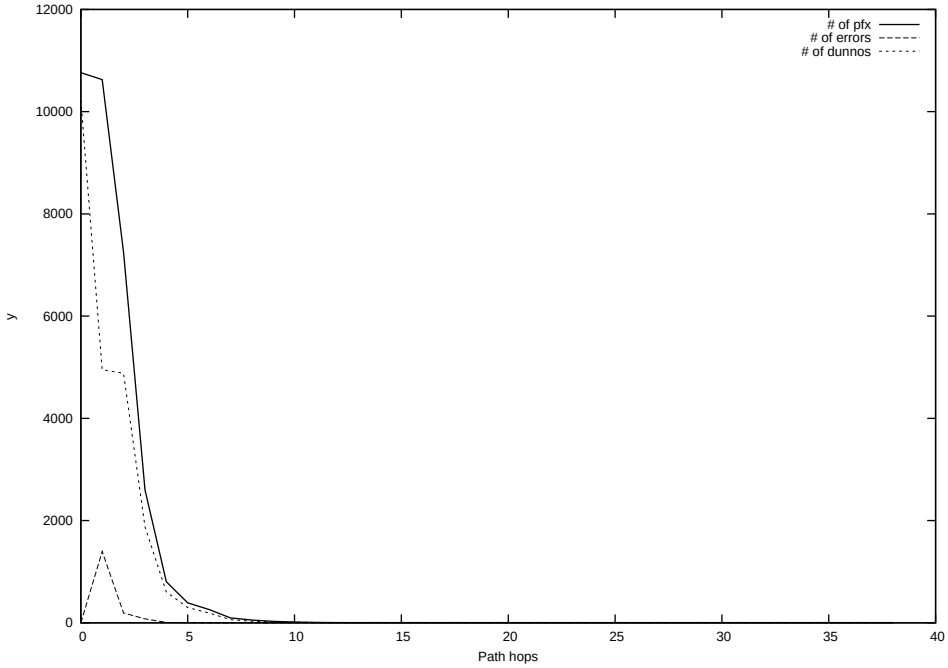
2012-11-23



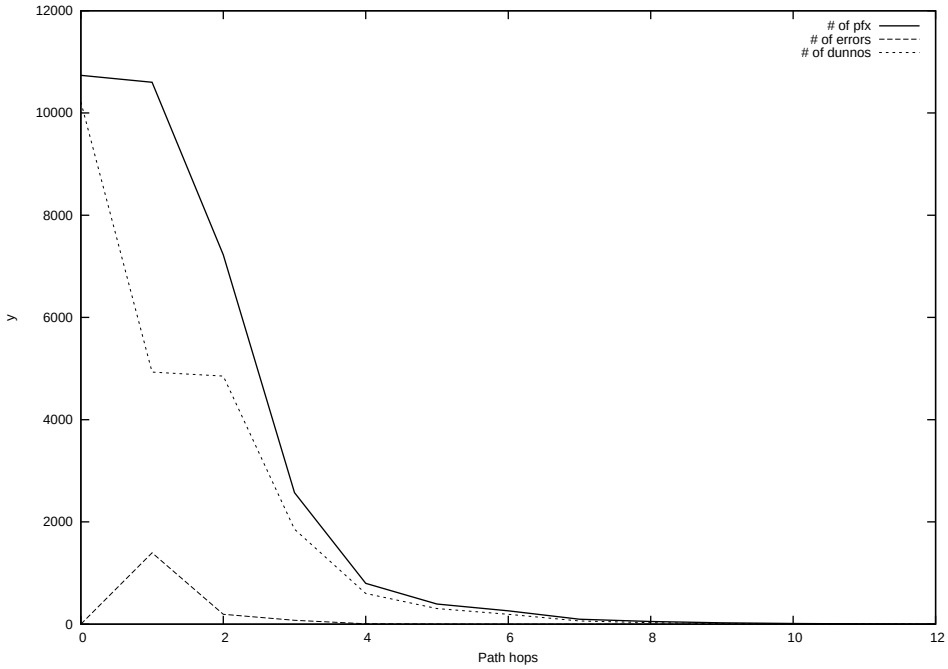
2012-11-24



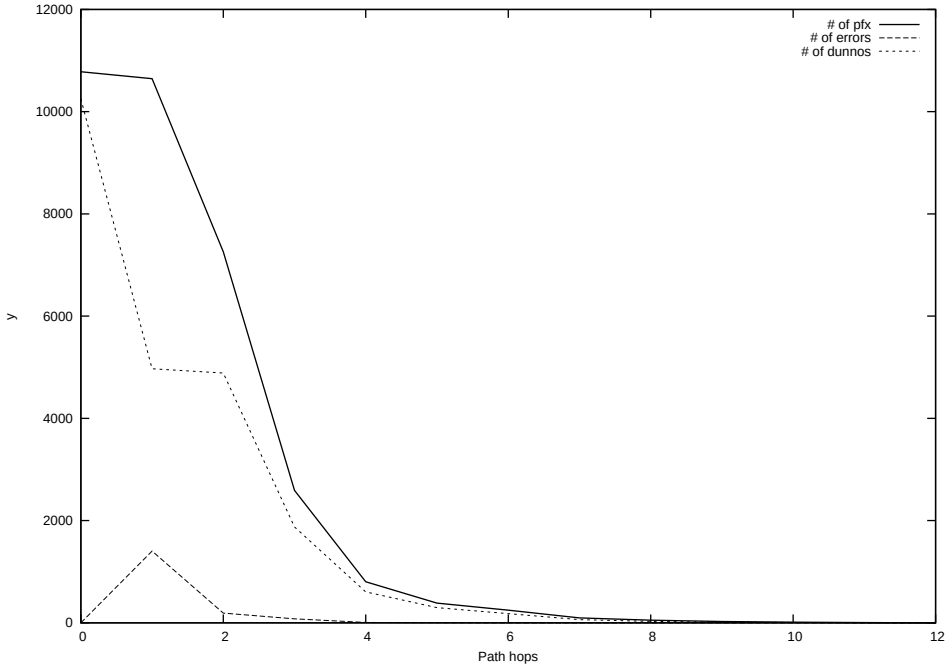
2012-11-25



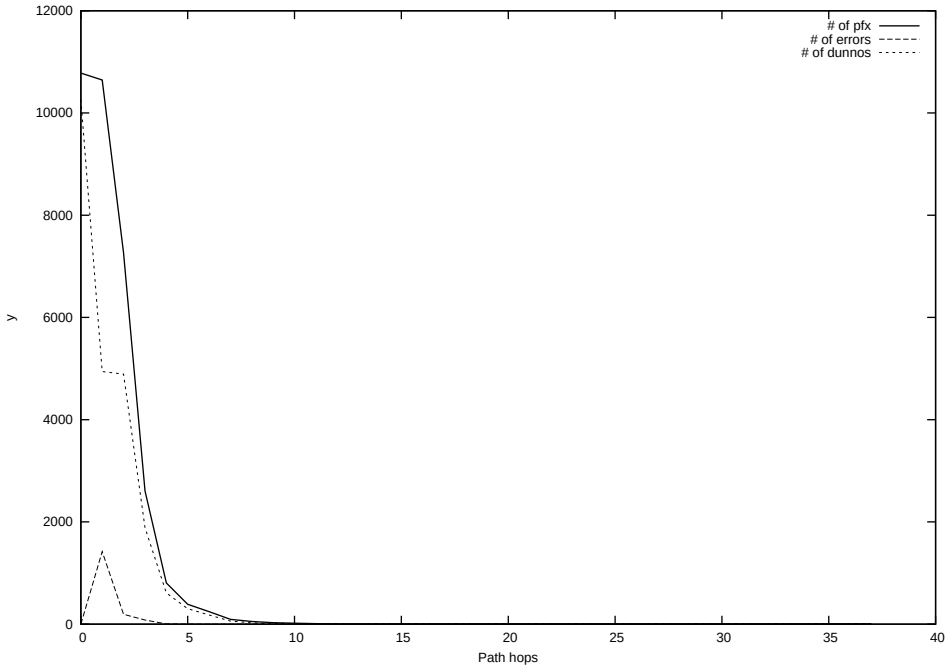
2012-11-26



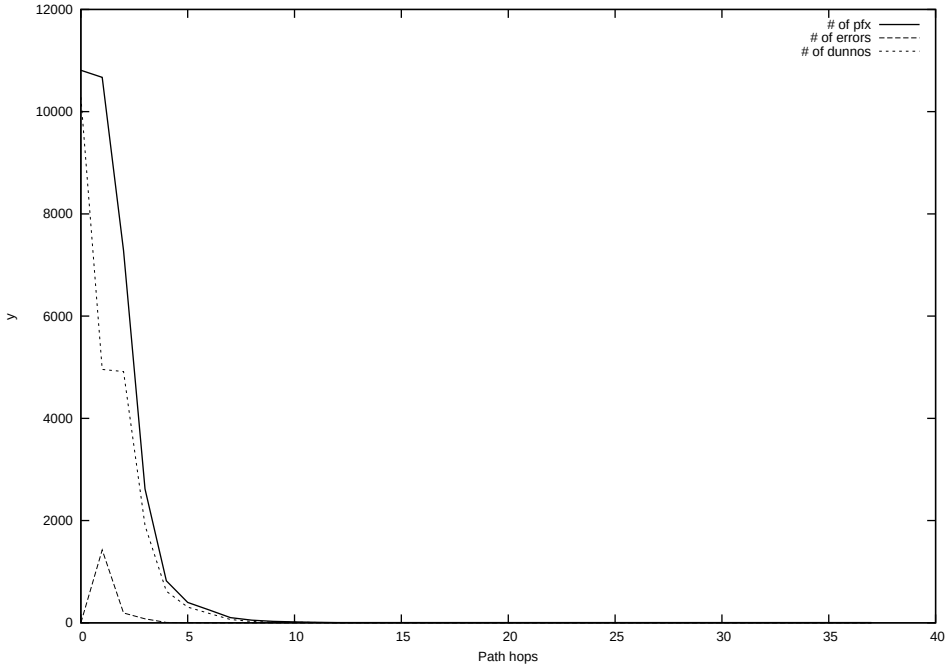
2012-11-27



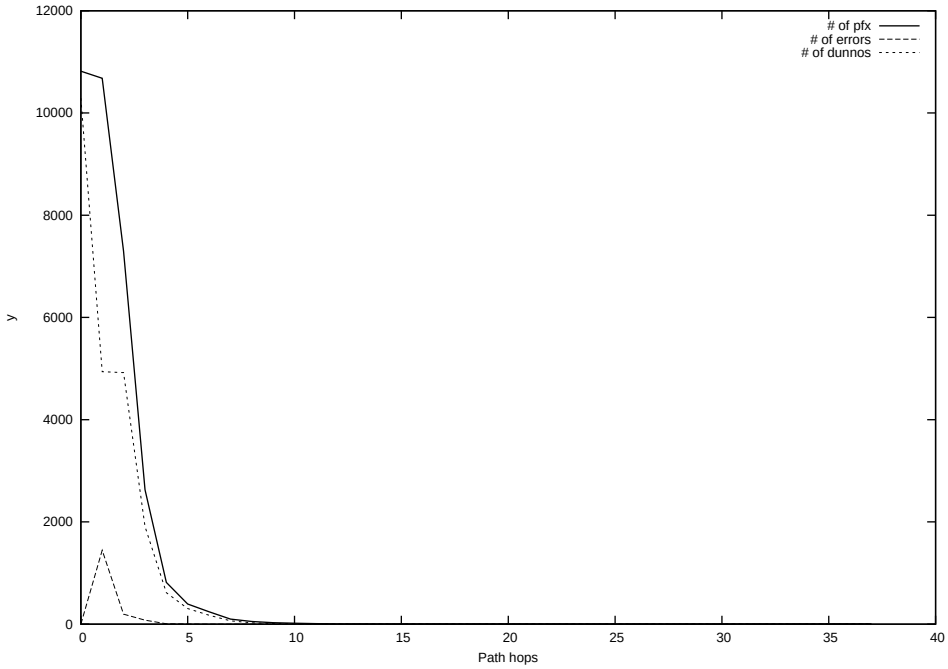
2012-11-28



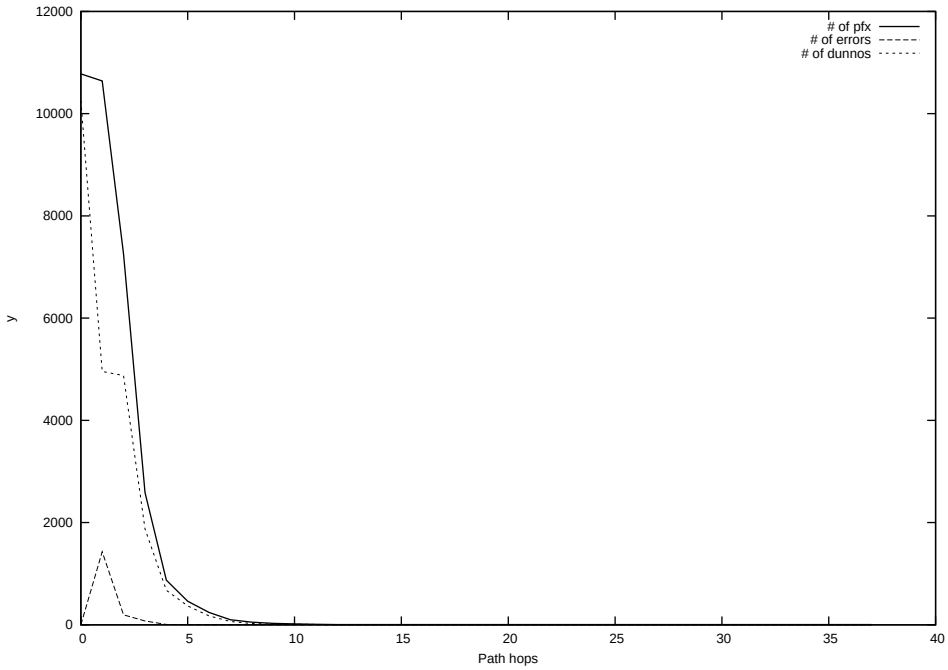
2012-11-29



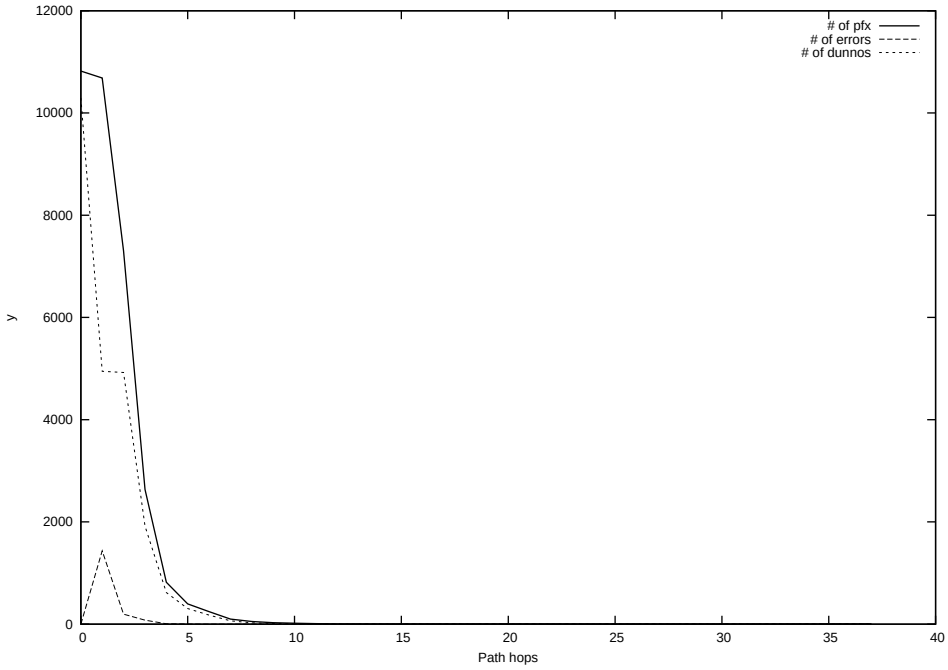
2012-11-30



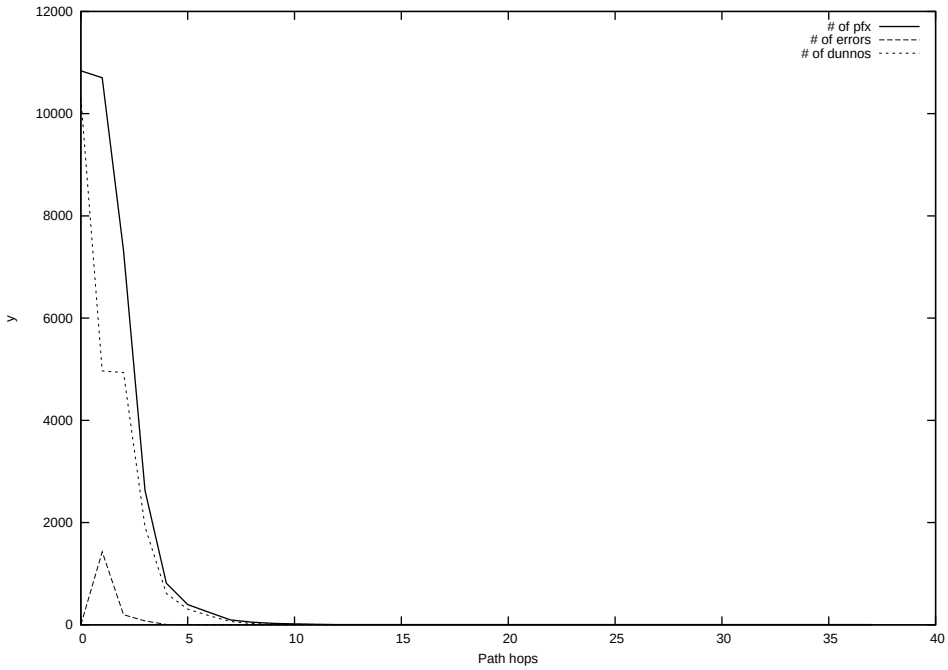
2012-12-01



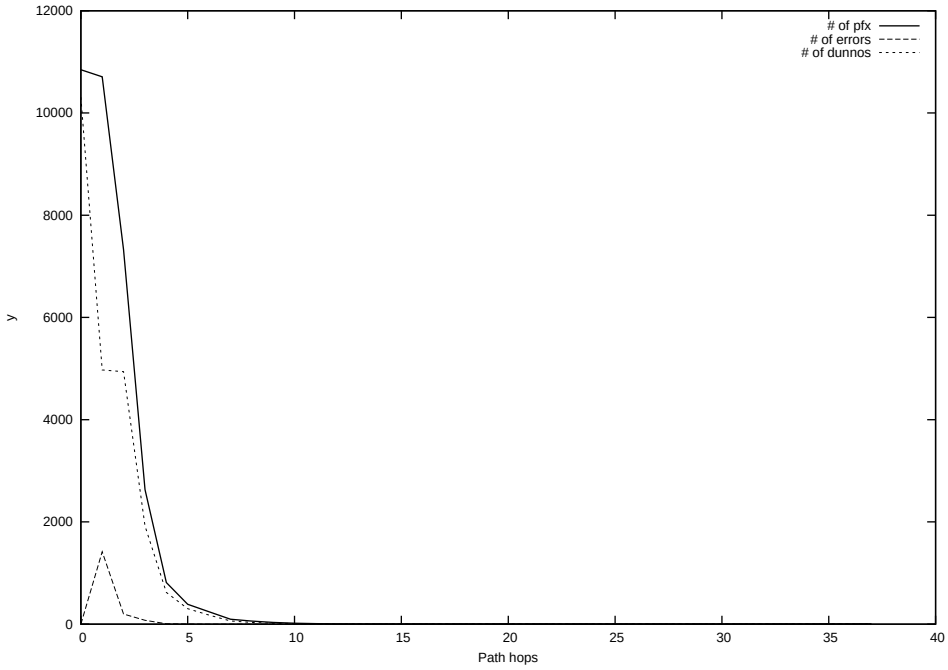
2012-12-02



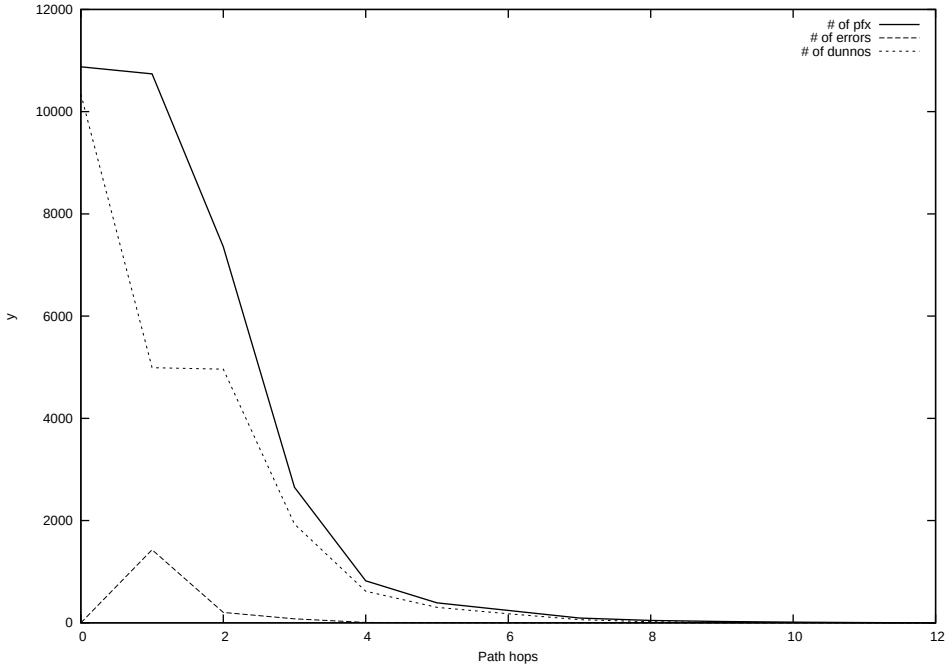
2012-12-03



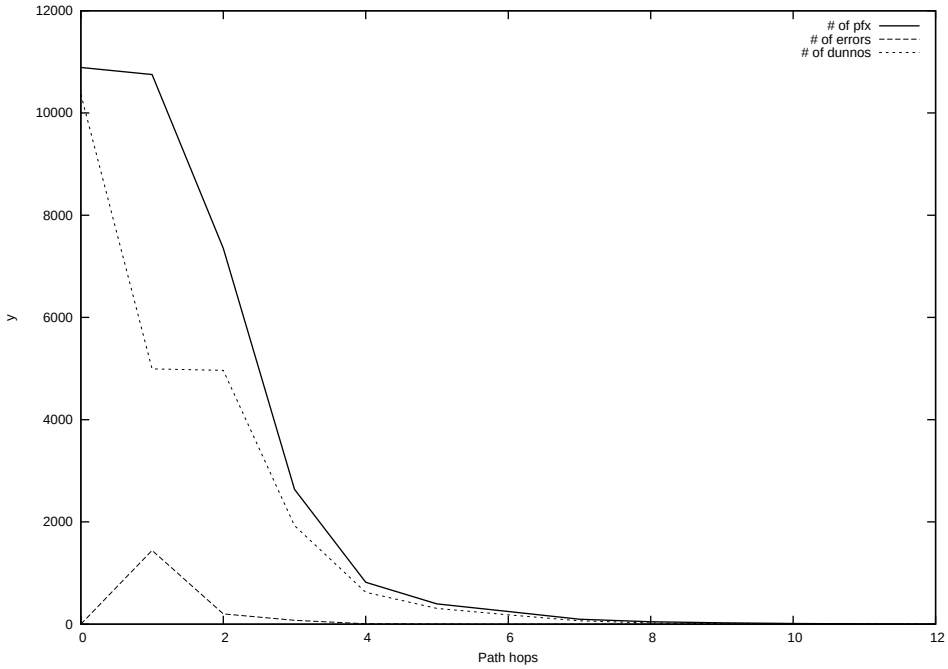
2012-12-04



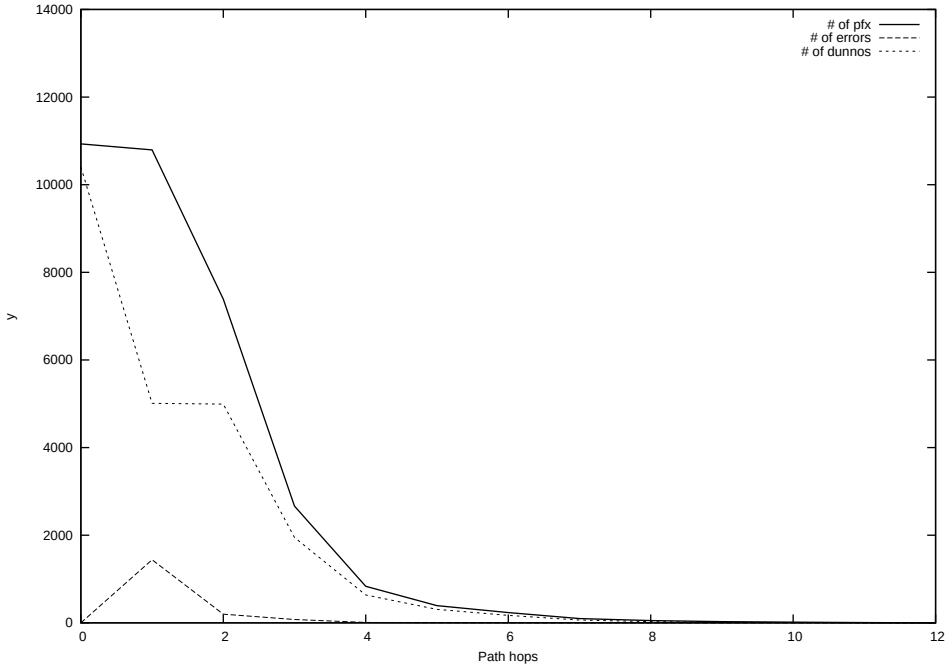
2012-12-05



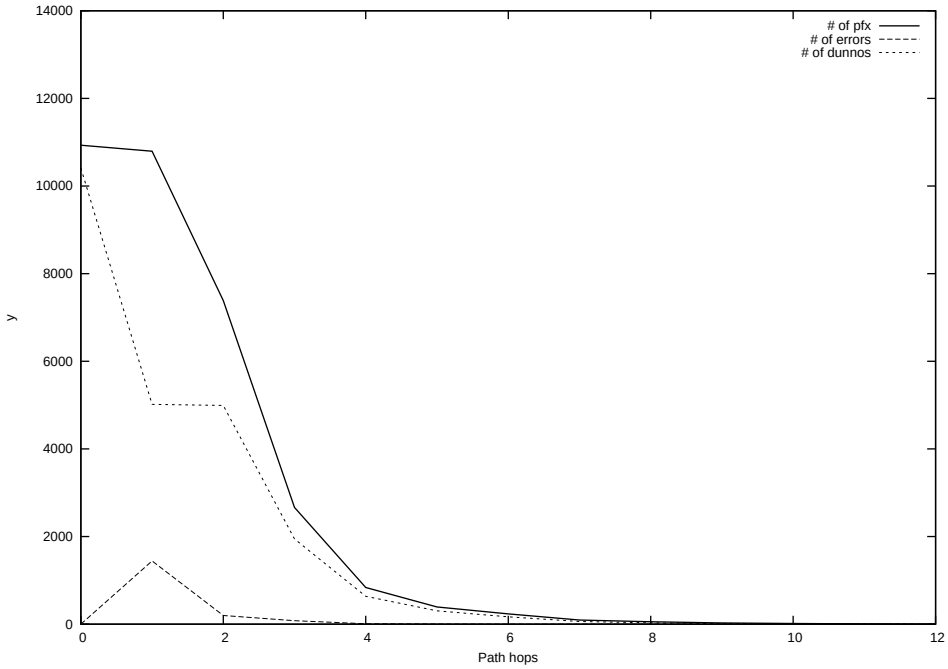
2012-12-06



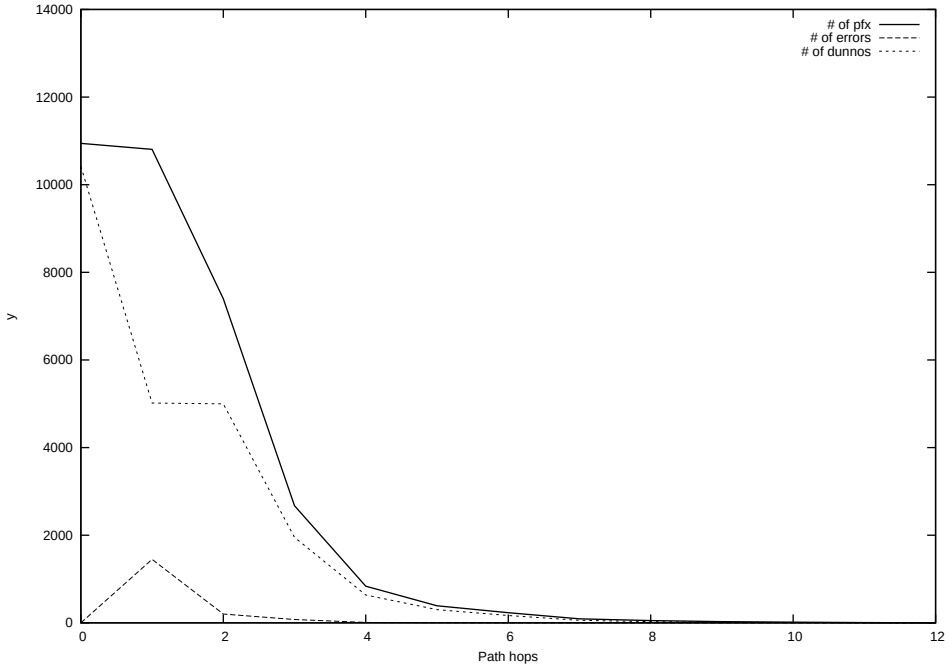
2012-12-07



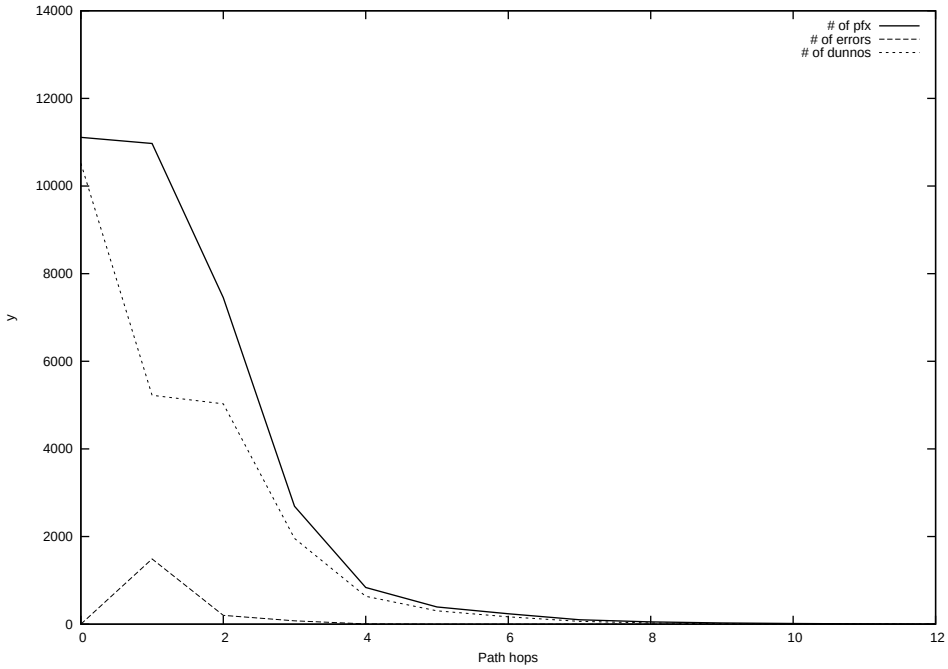
2012-12-08



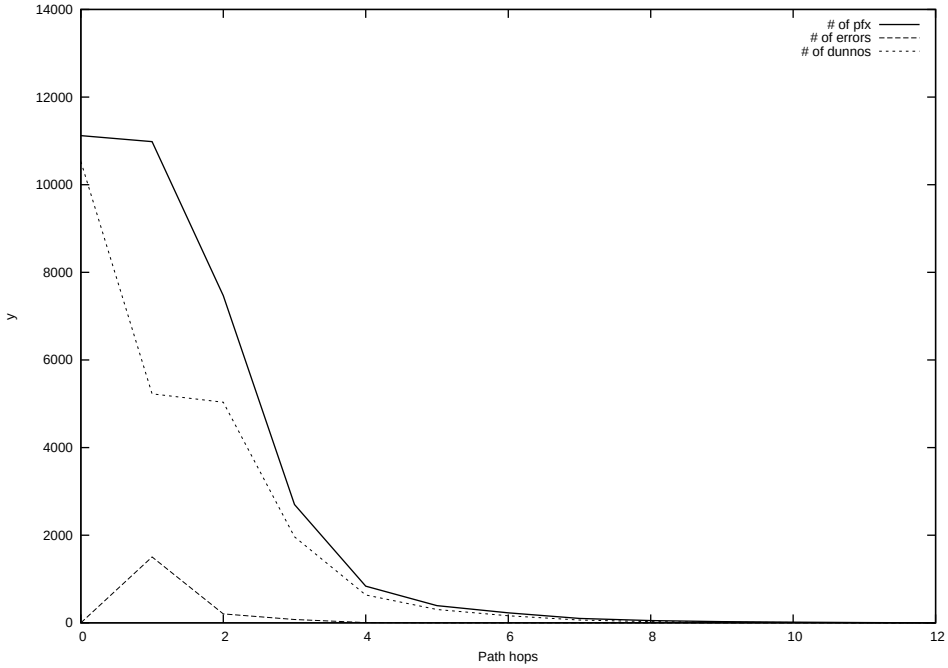
2012-12-09



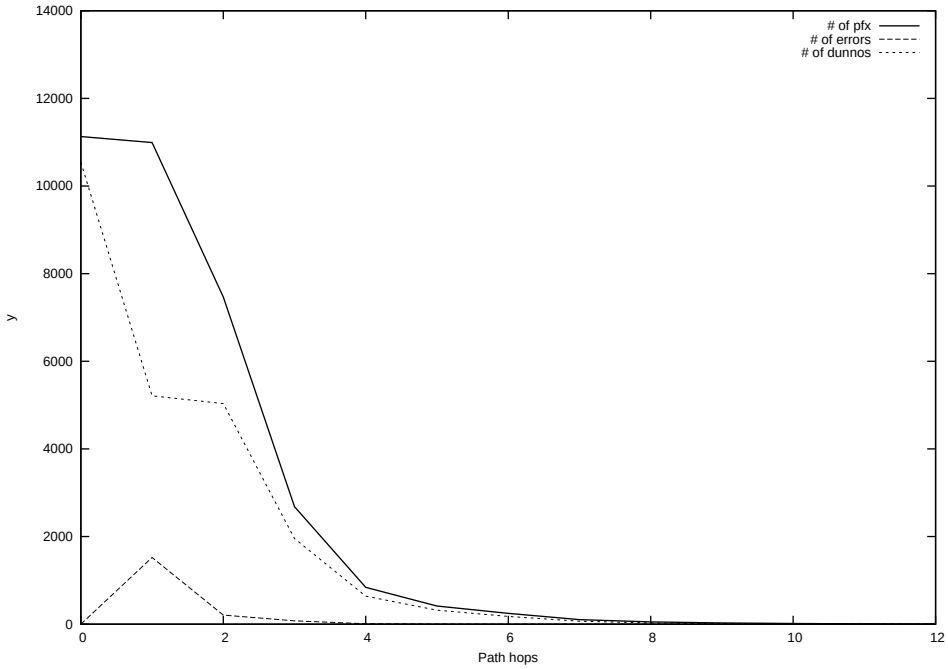
2012-12-10



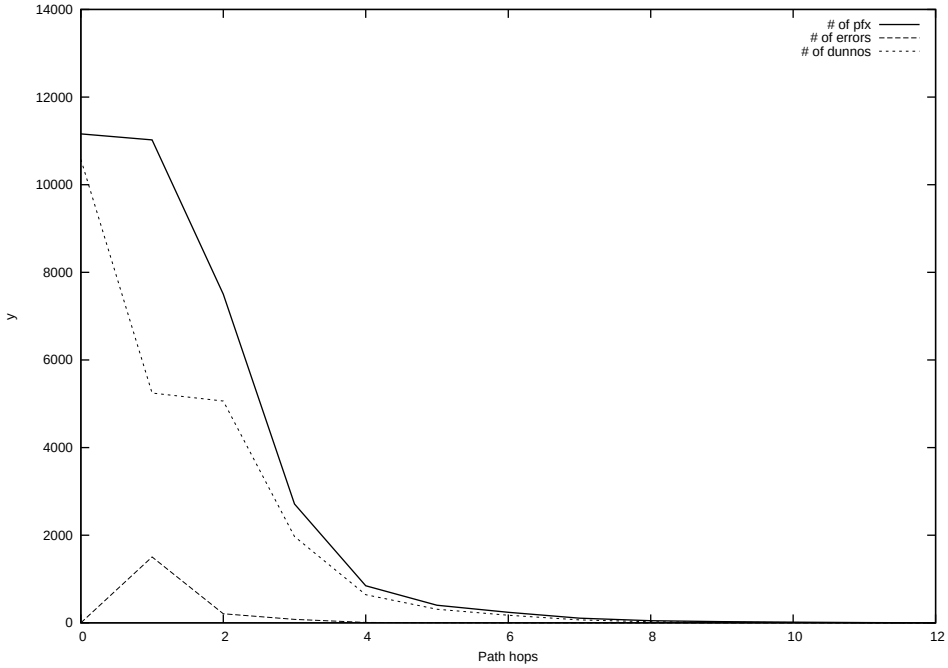
2012-12-11



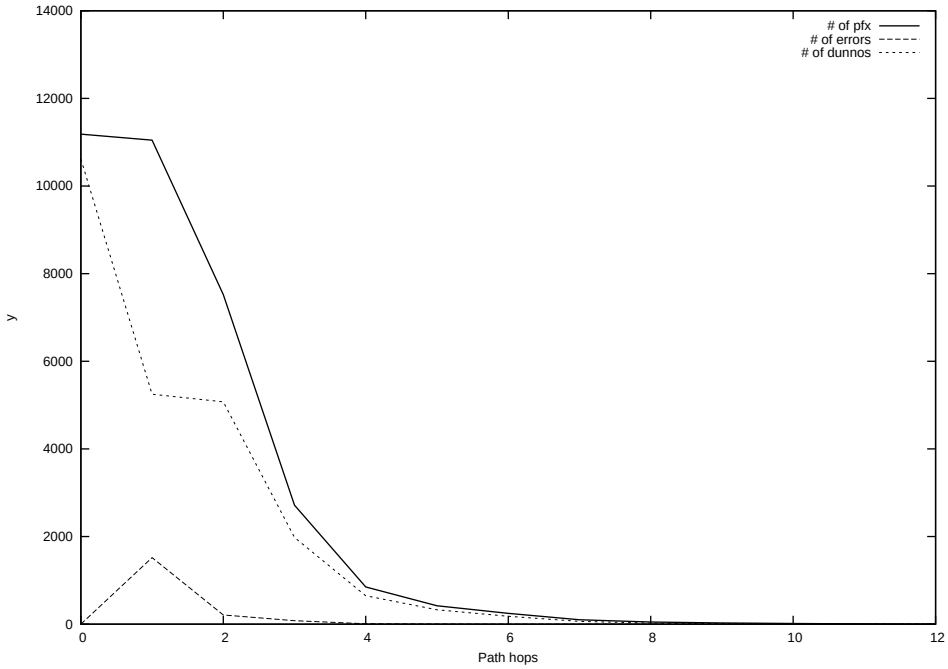
2012-12-12



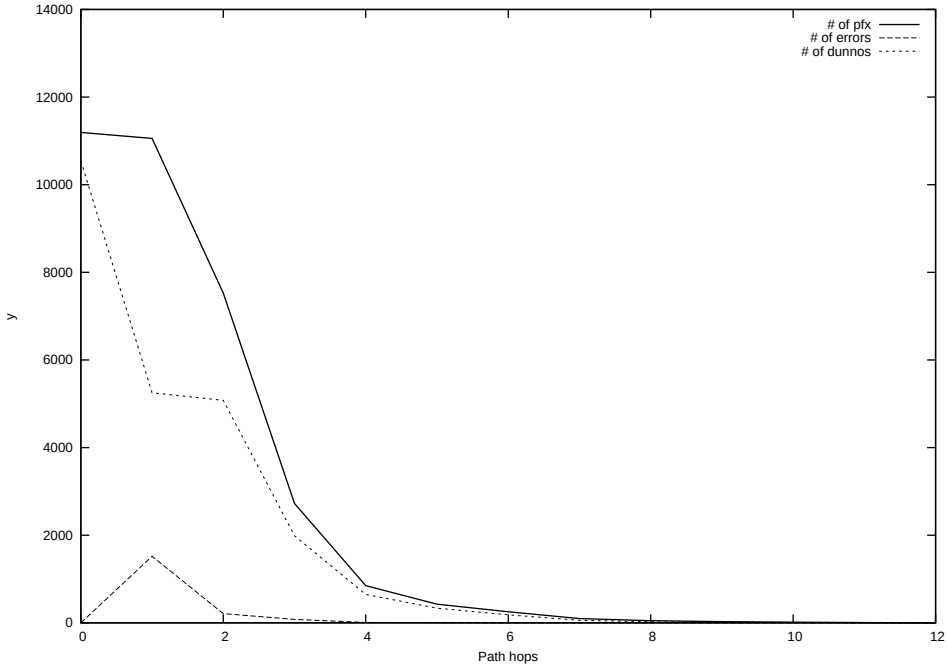
2012-12-13



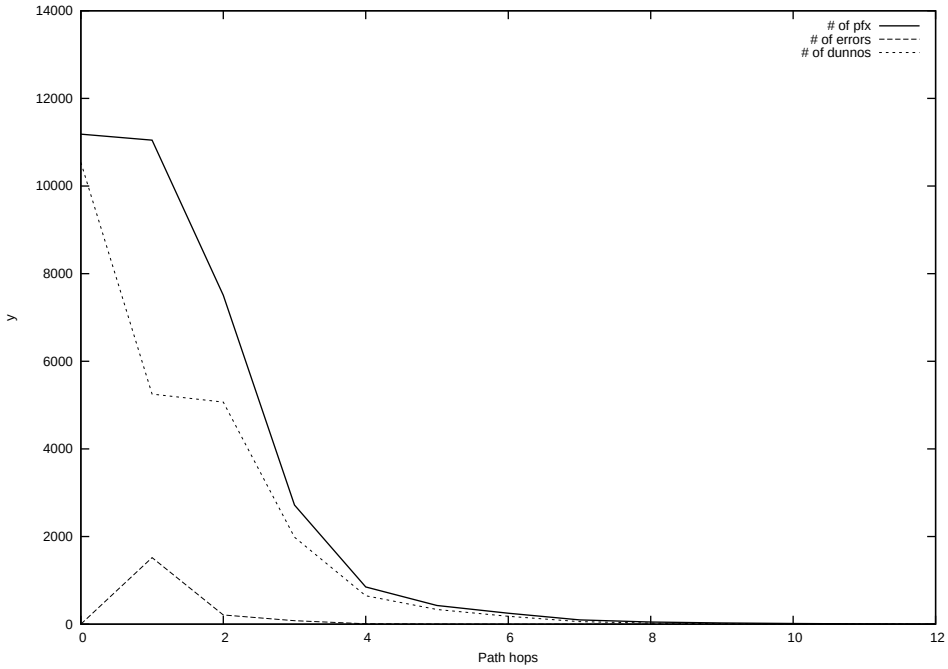
2012-12-14



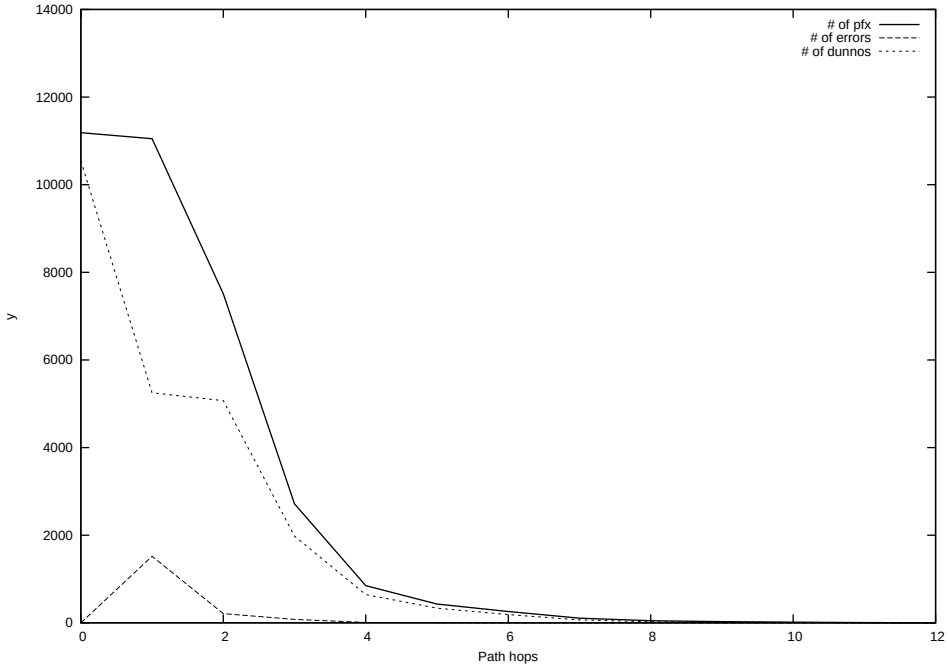
2012-12-15



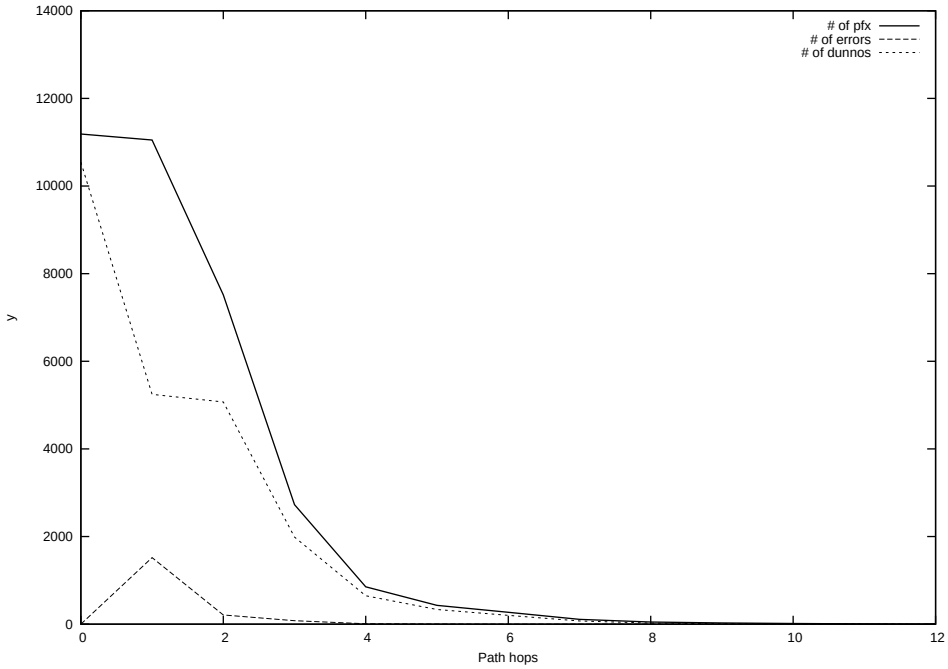
2012-12-16



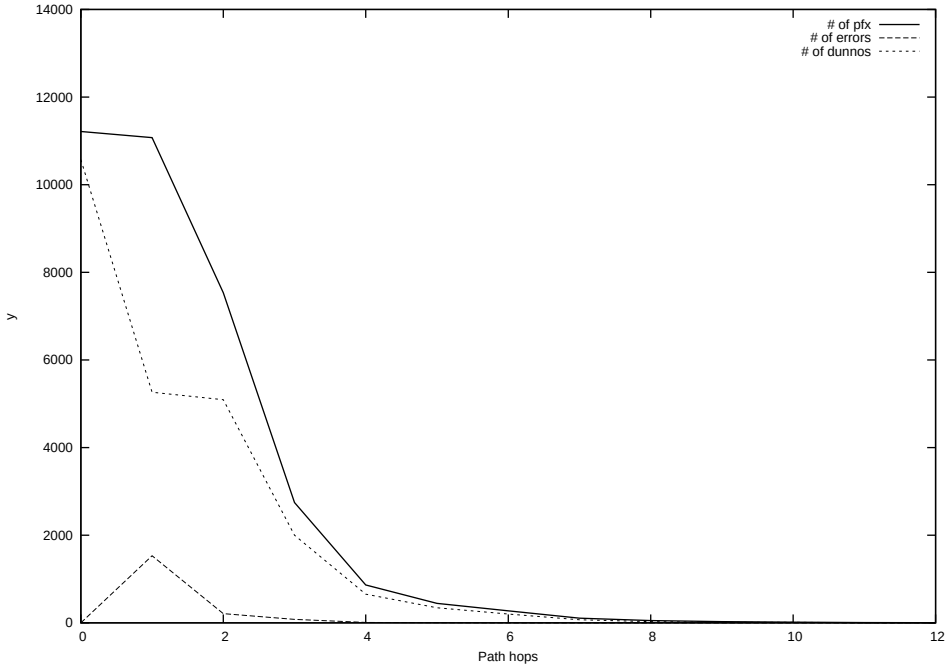
2012-12-17



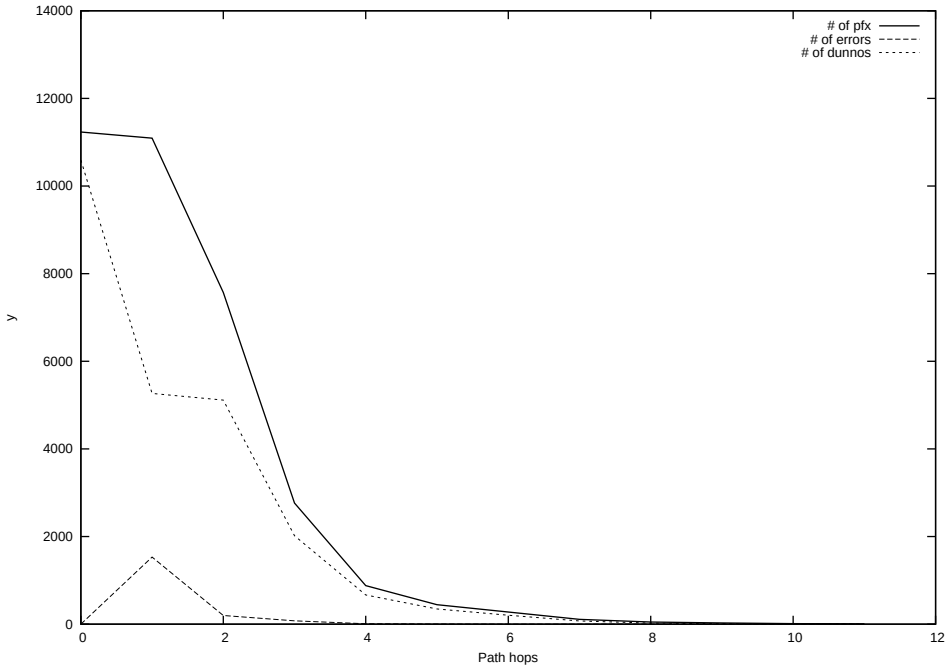
2012-12-18



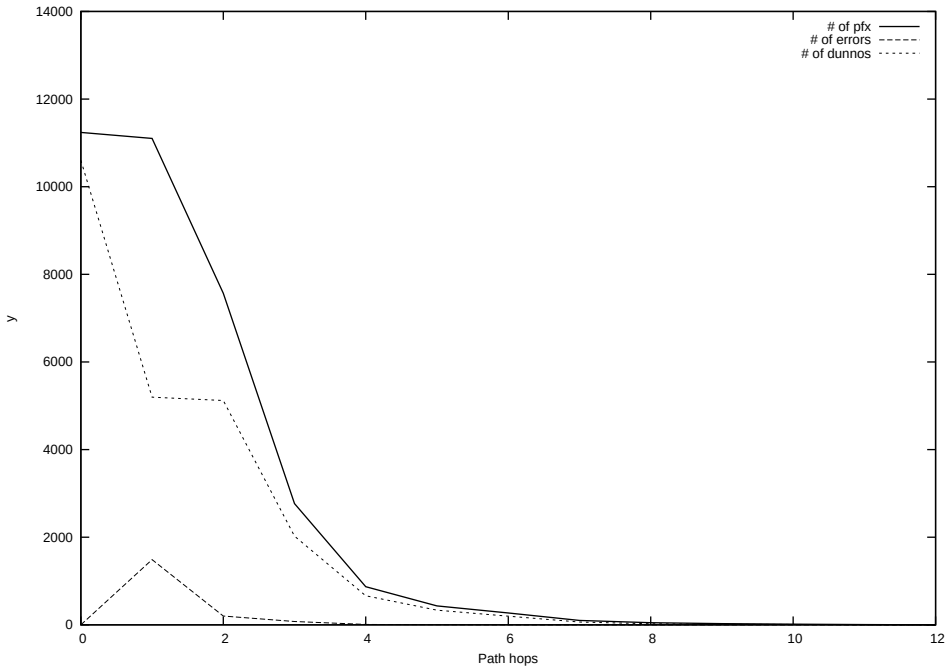
2012-12-19



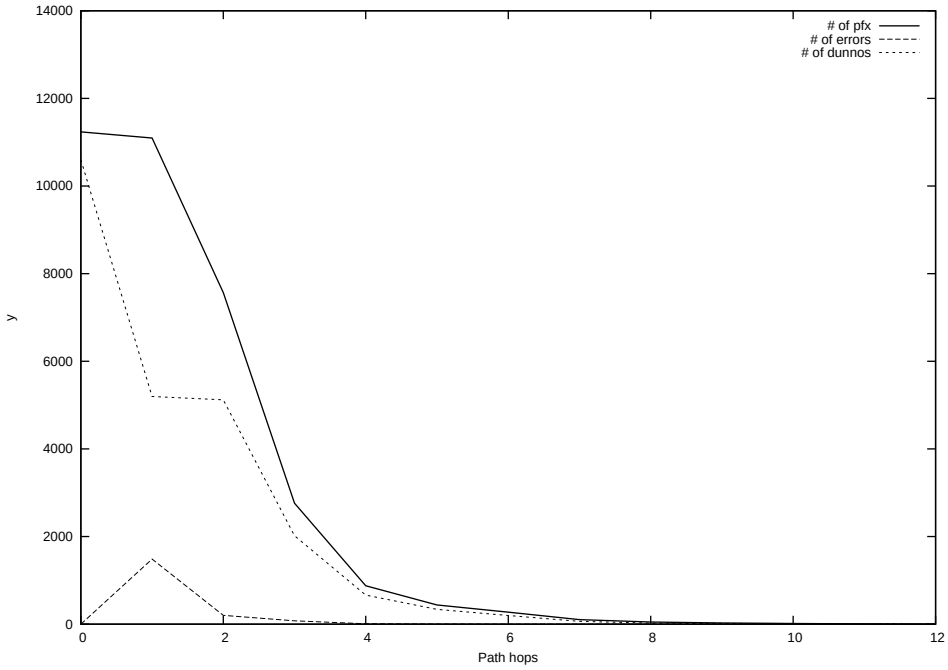
2012-12-20



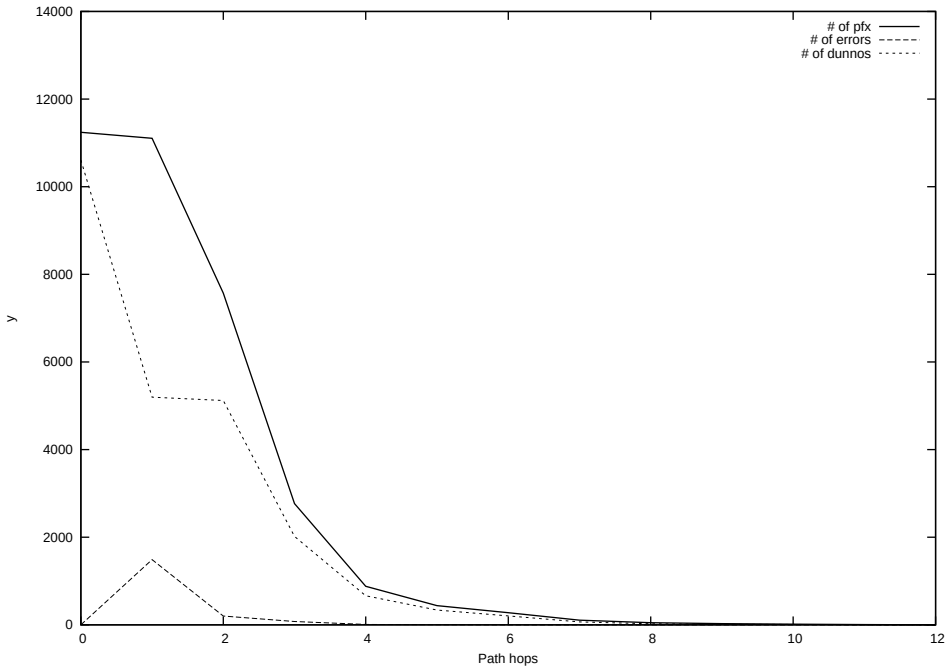
2012-12-21



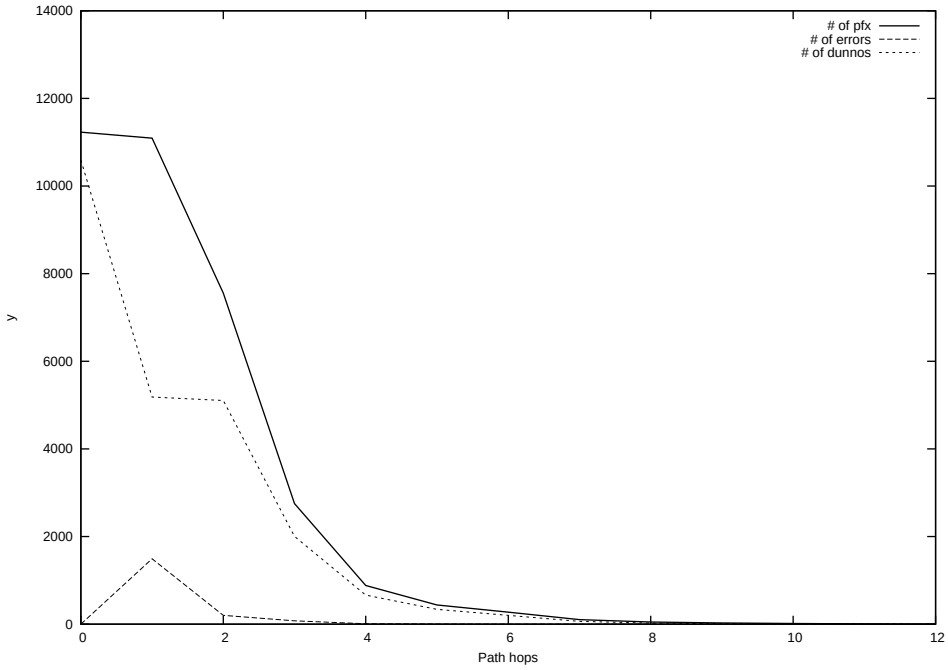
2012-12-22



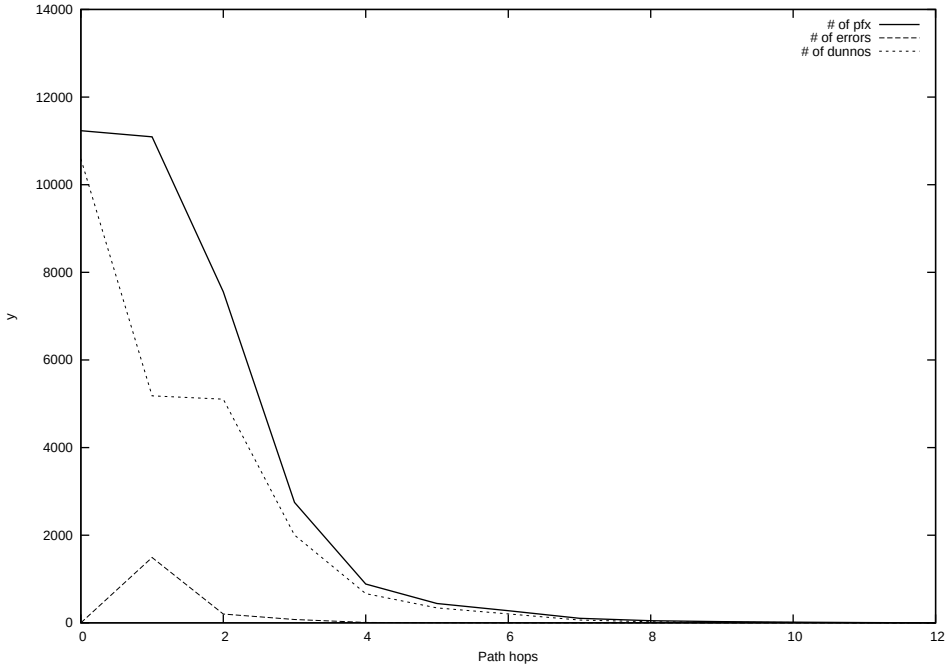
2012-12-23



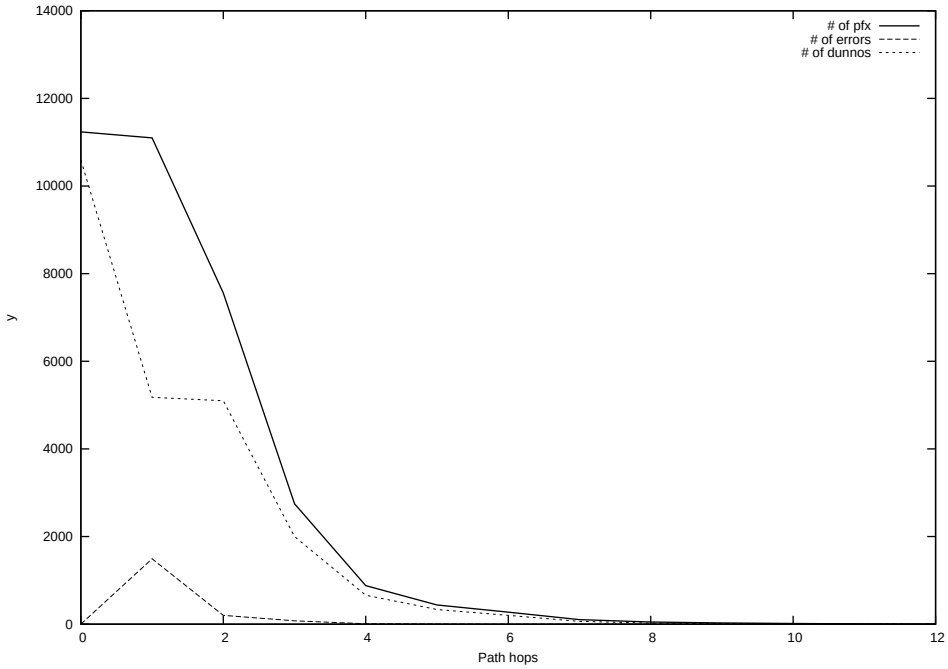
2012-12-24



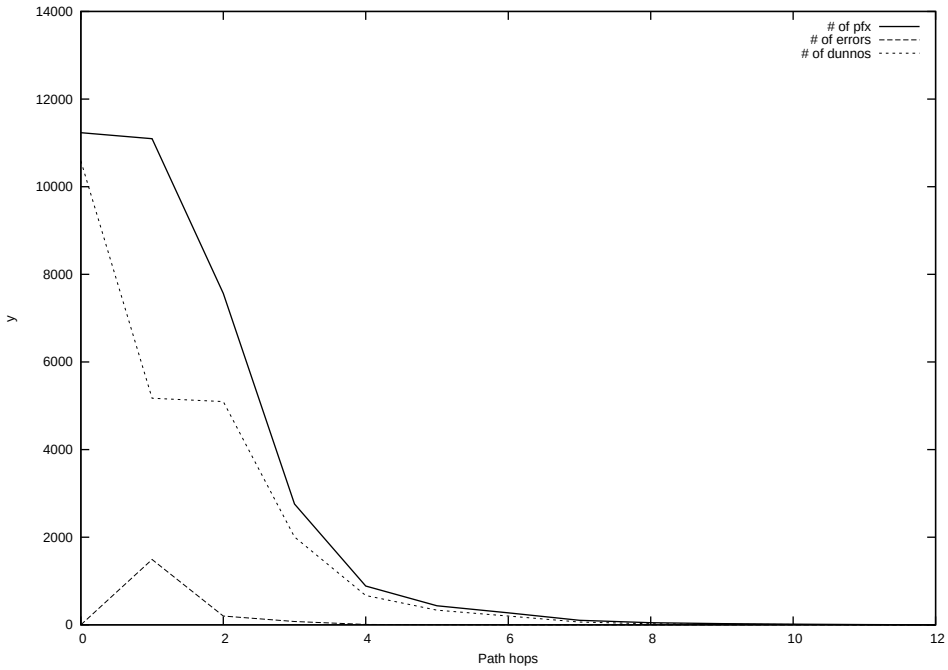
2012-12-25



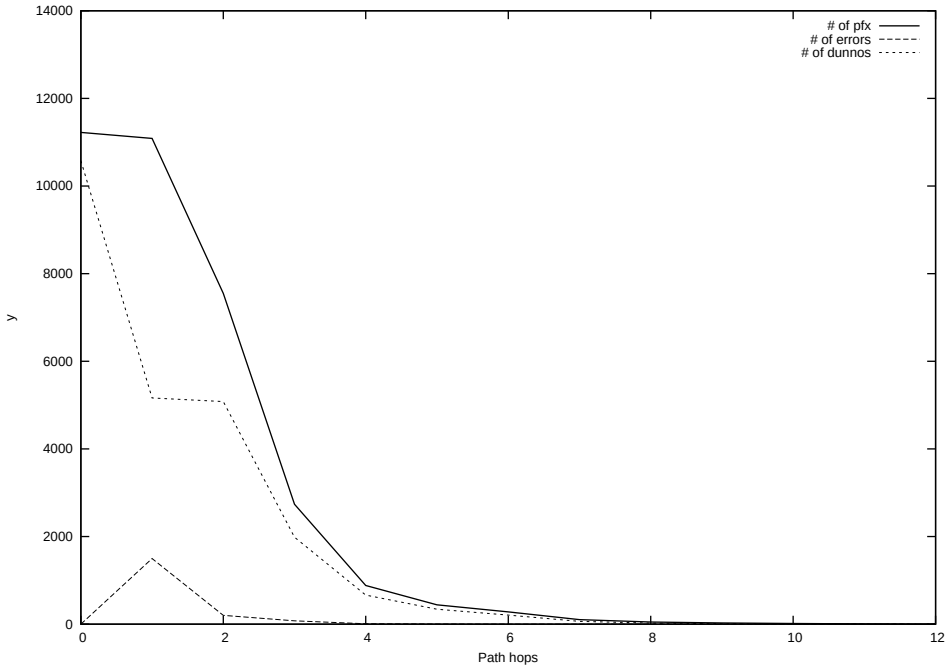
2012-12-26



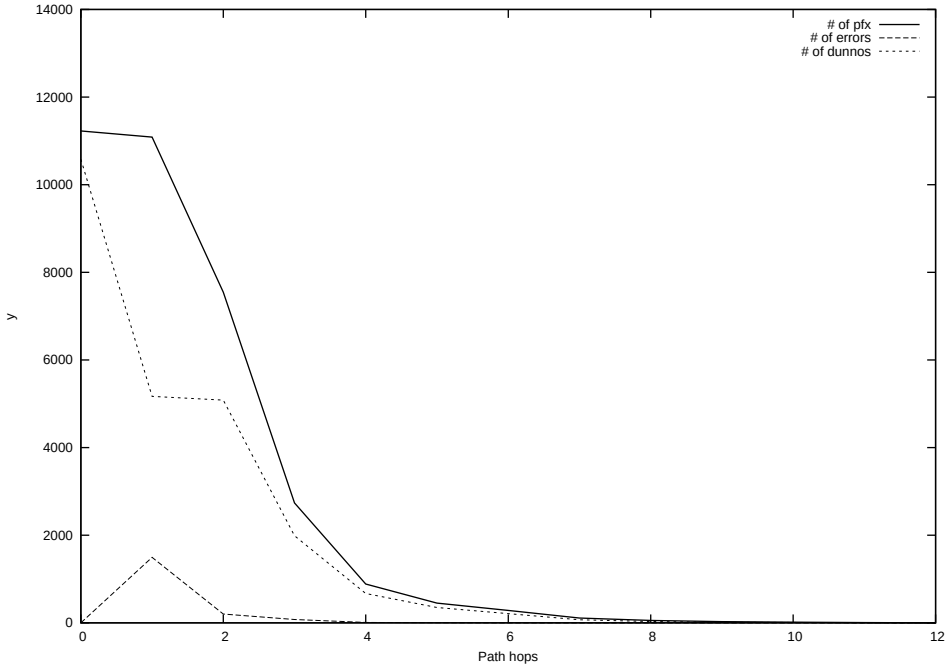
2012-12-27



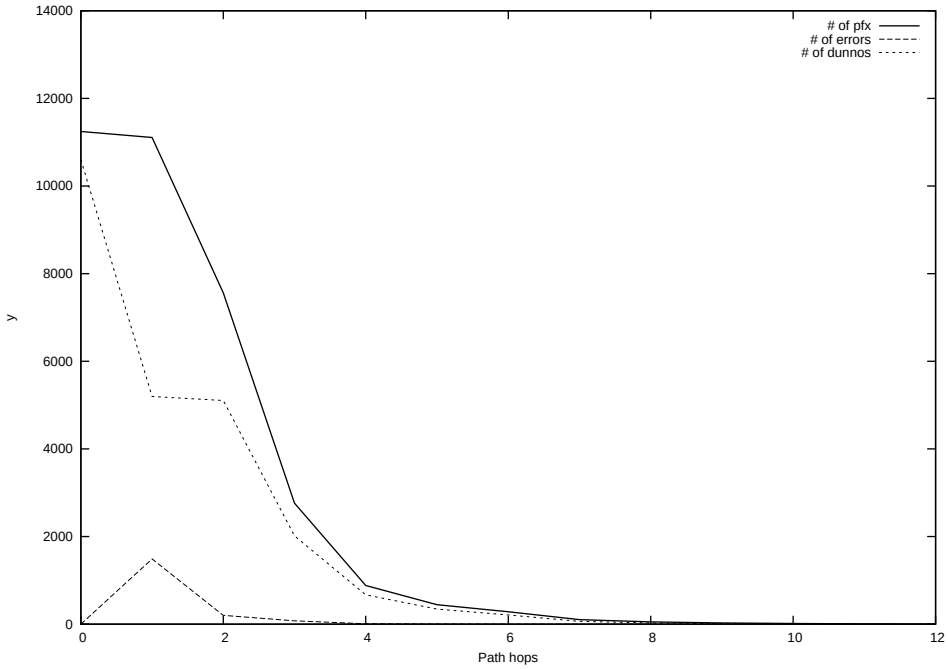
2012-12-28



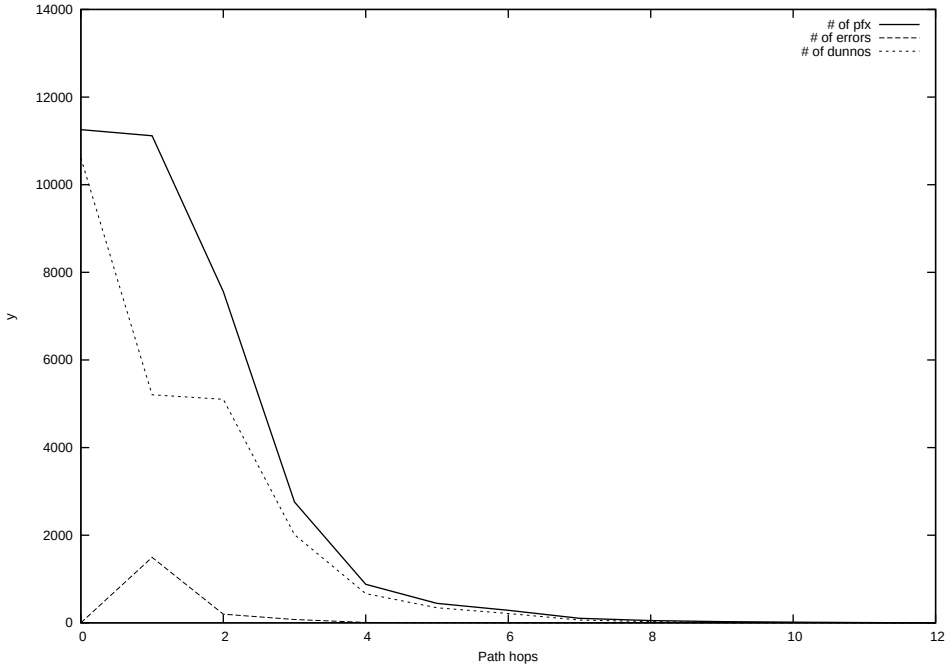
2012-12-29



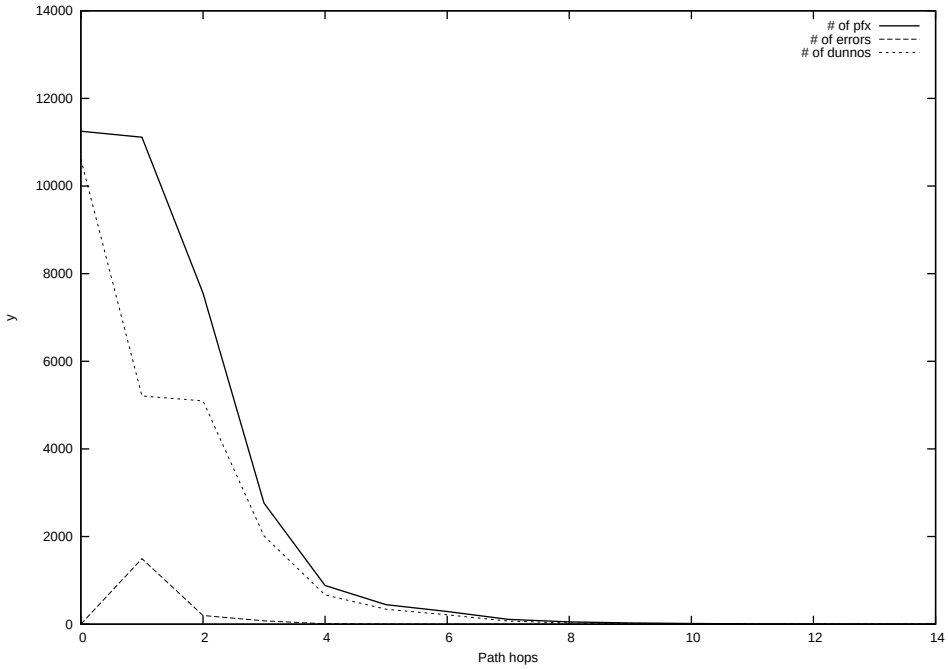
2012-12-30



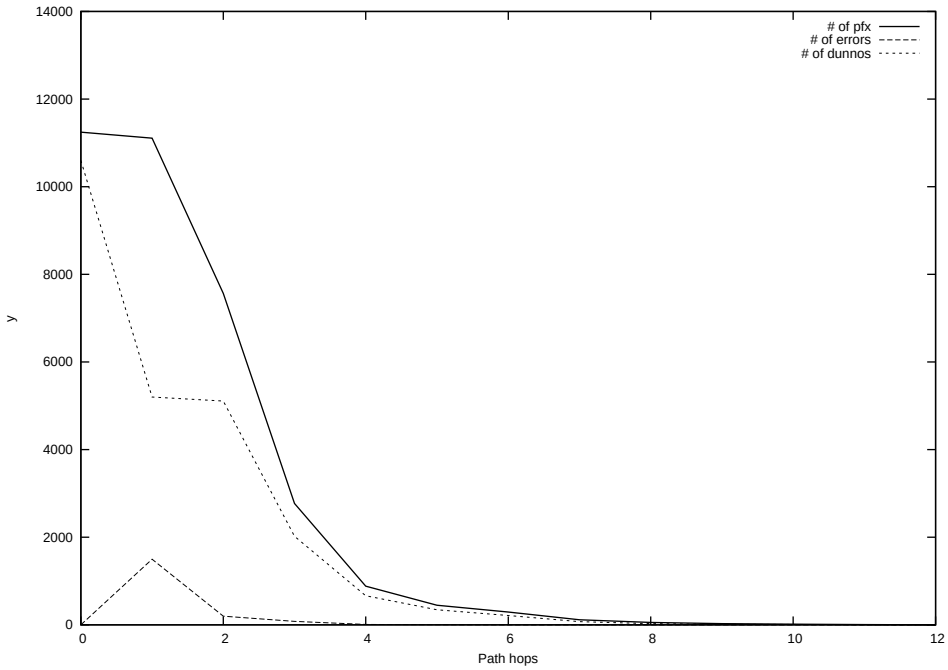
2012-12-31



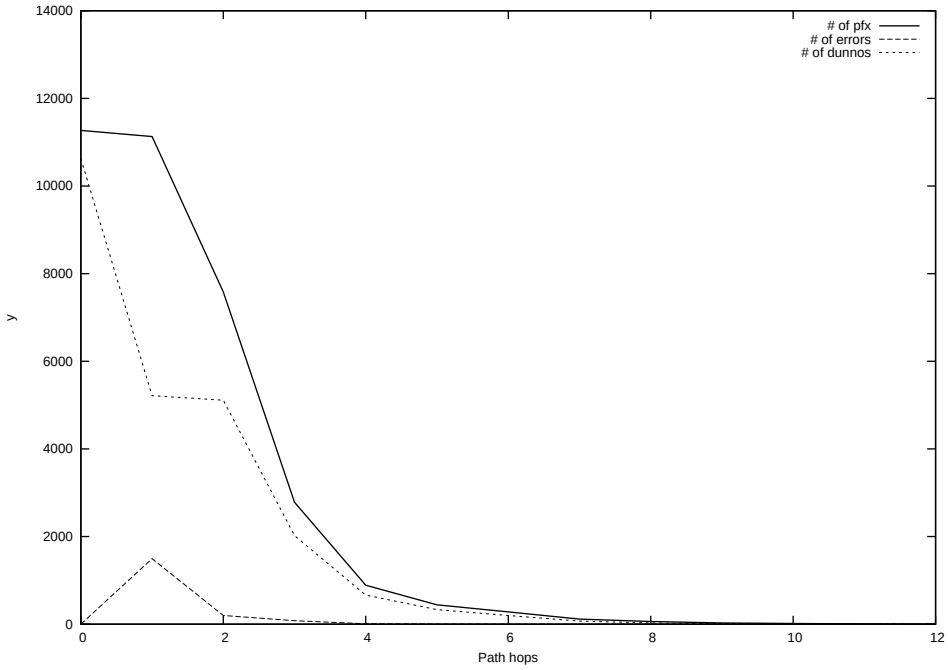
2013-01-01



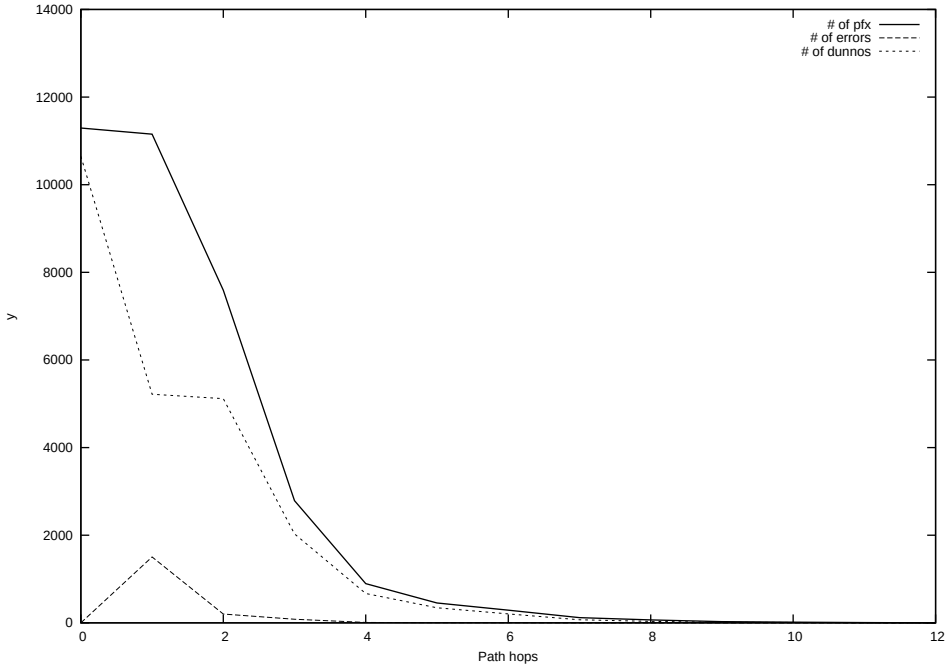
2013-01-02



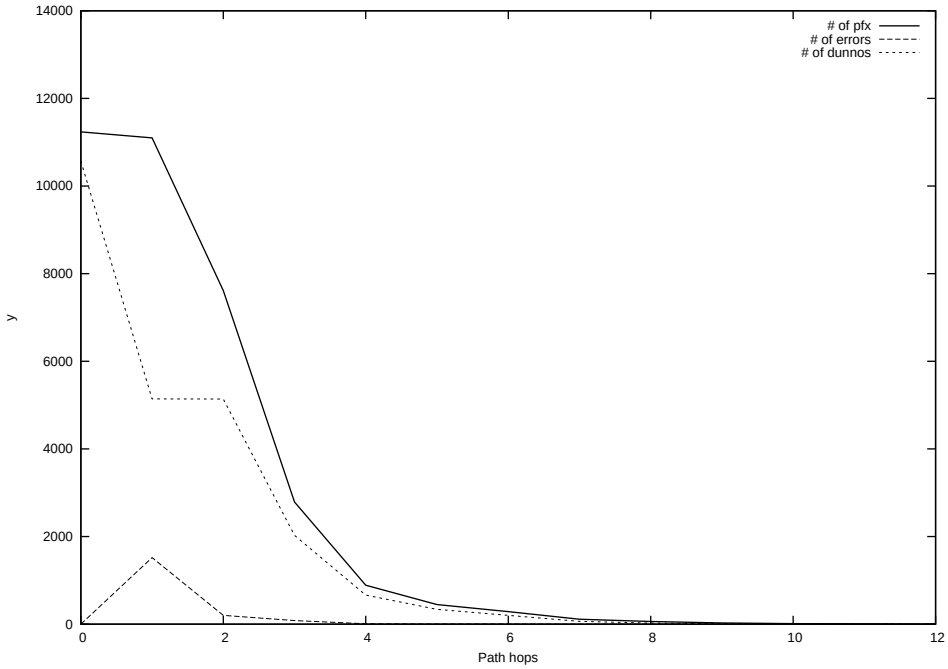
2013-01-03



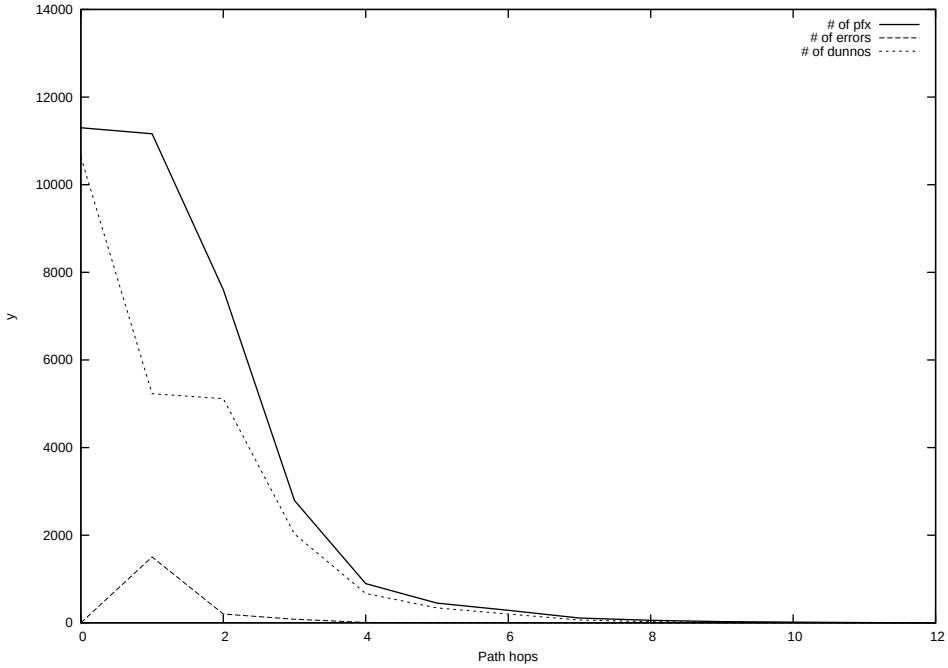
2013-01-04



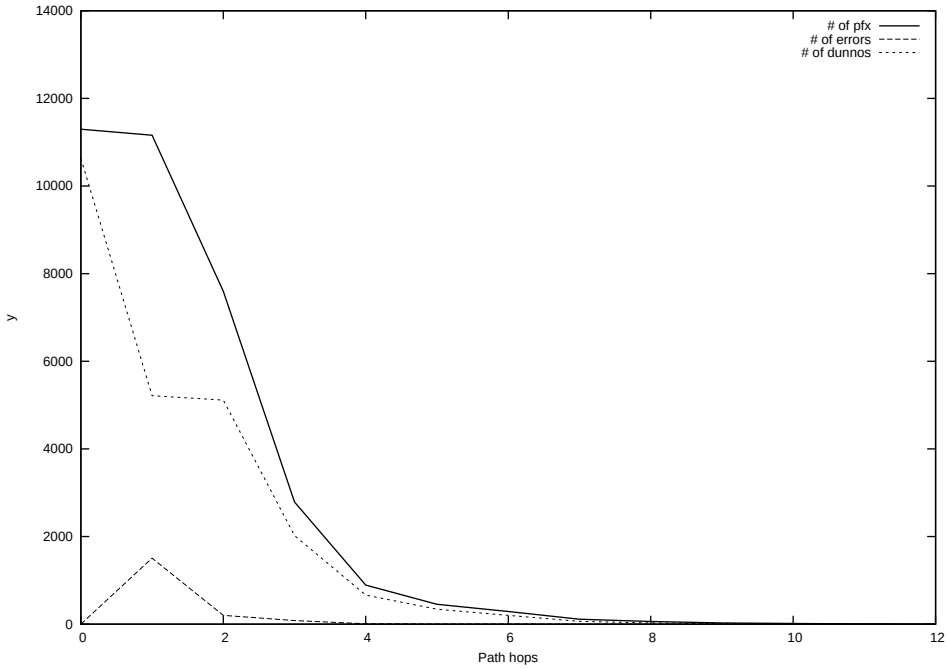
2013-01-05



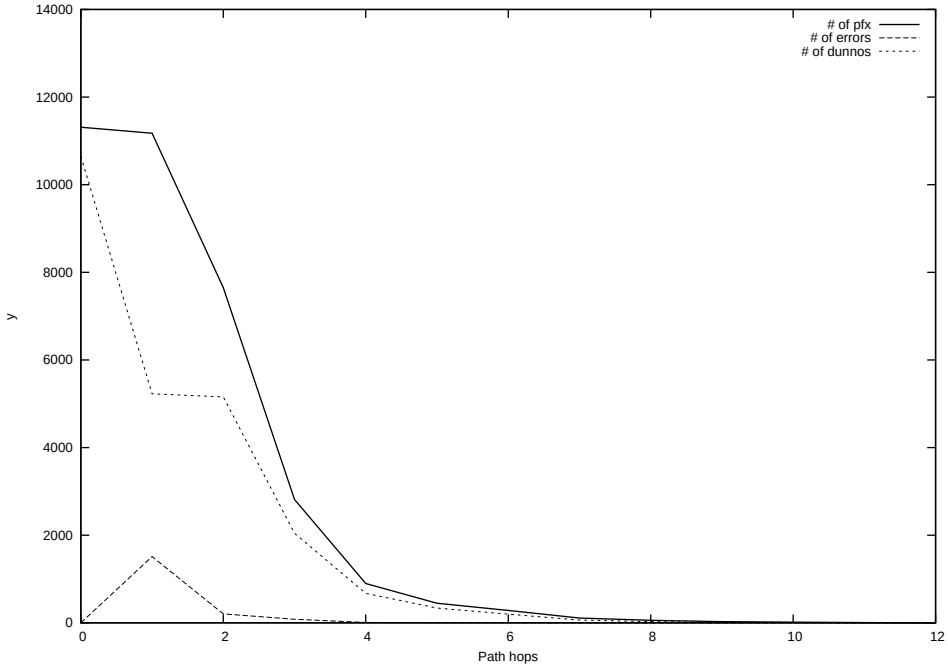
2013-01-06



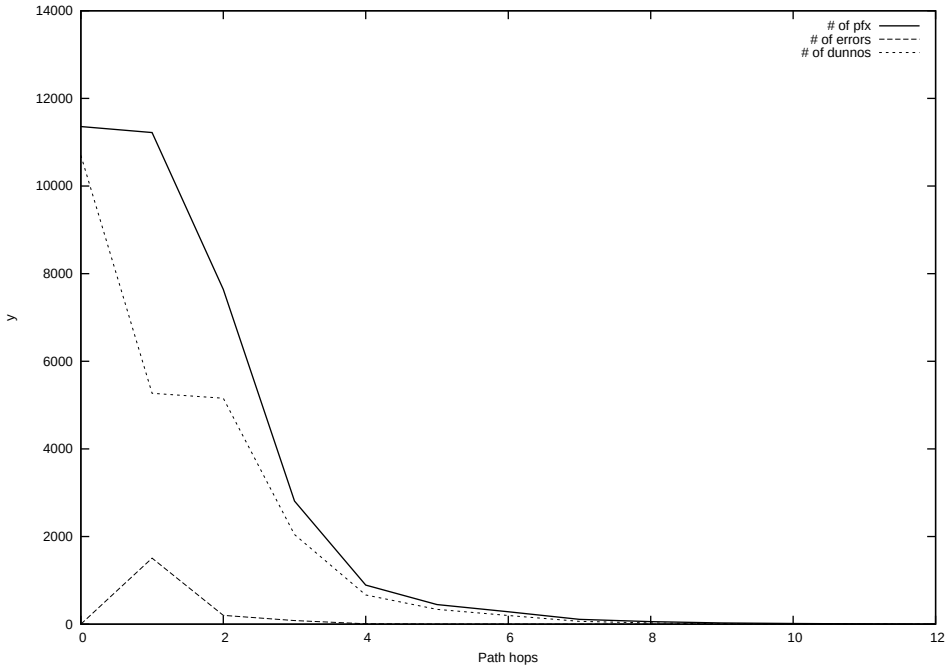
2013-01-07



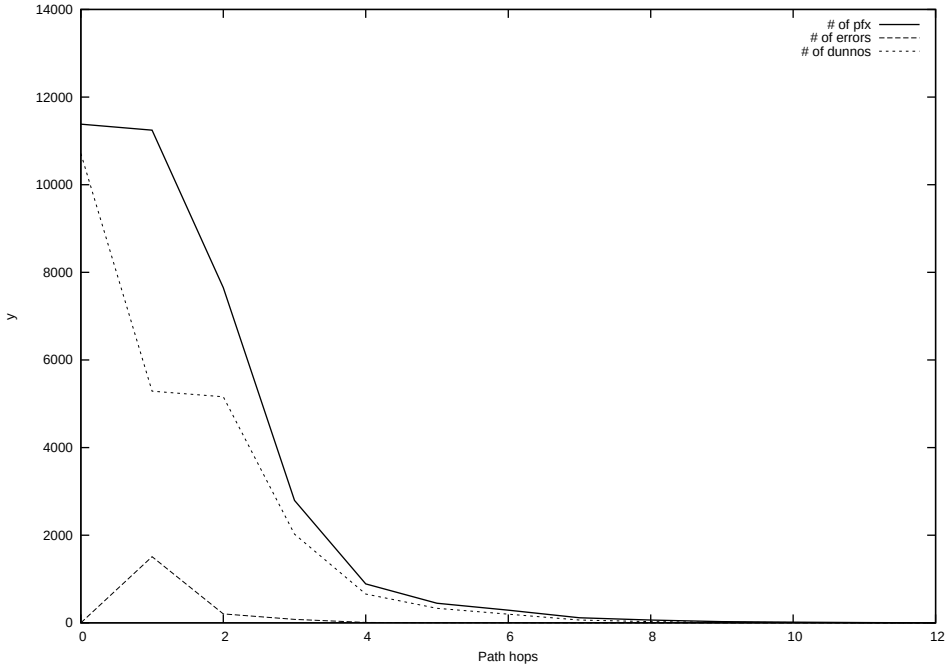
2013-01-08



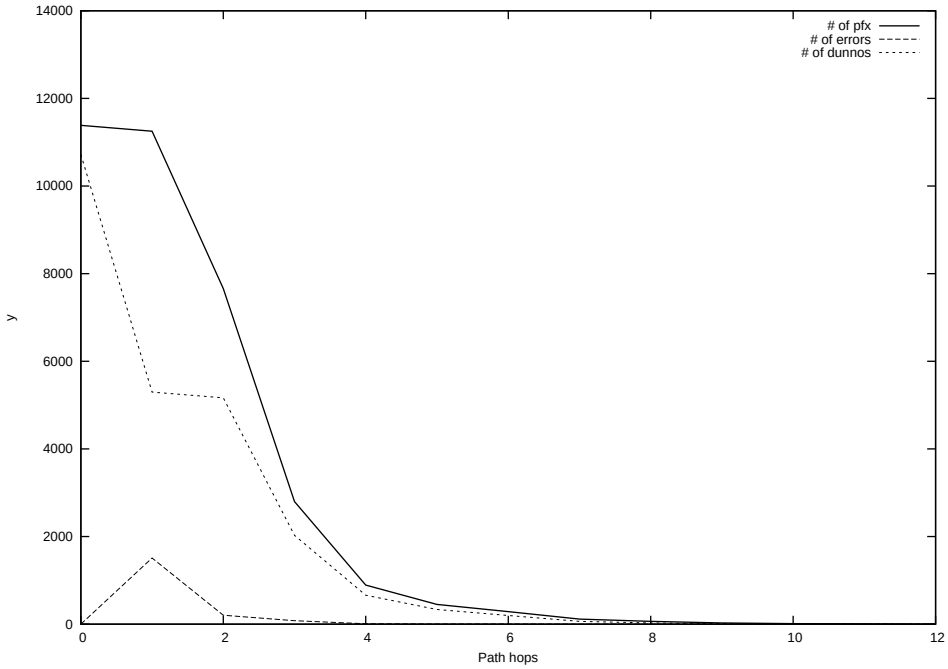
2013-01-09



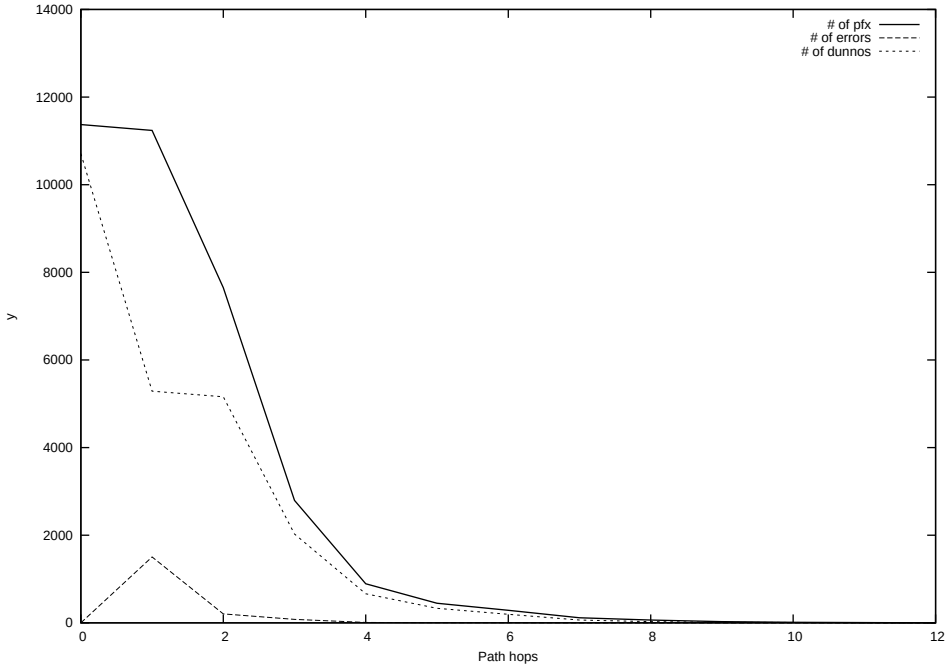
2013-01-10



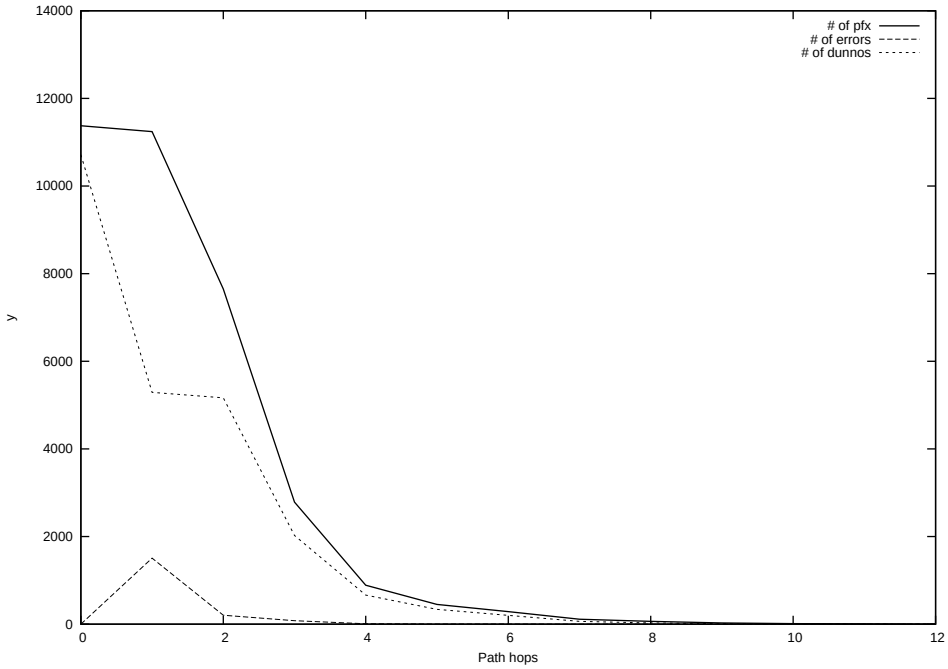
2013-01-11



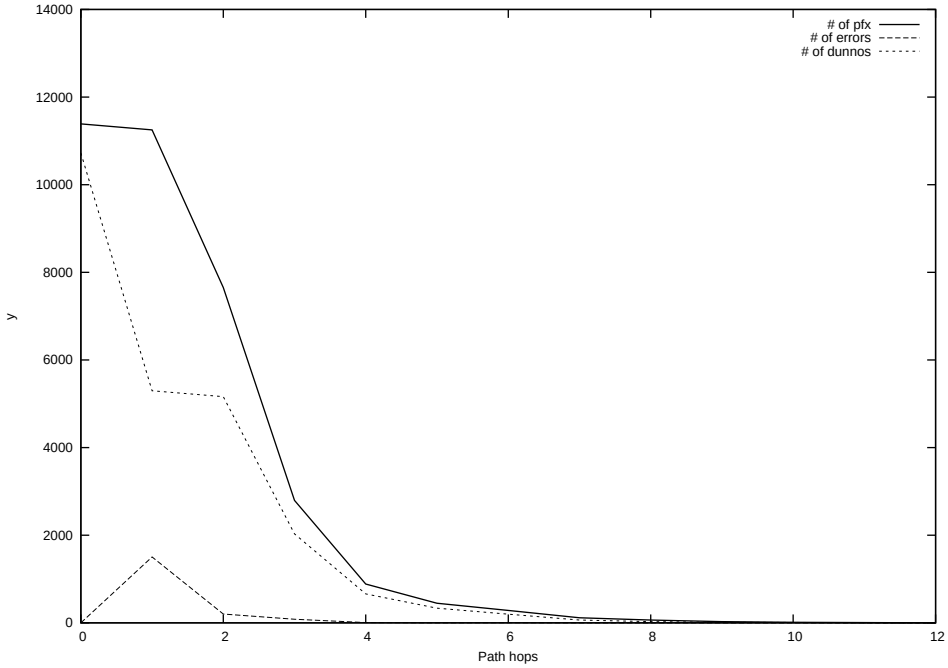
2013-01-12



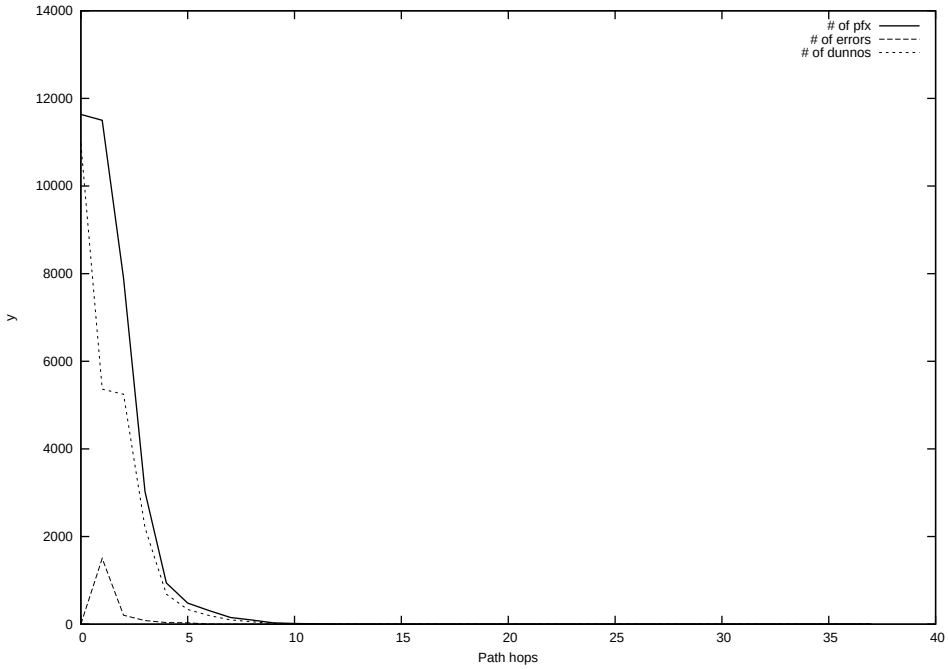
2013-01-13



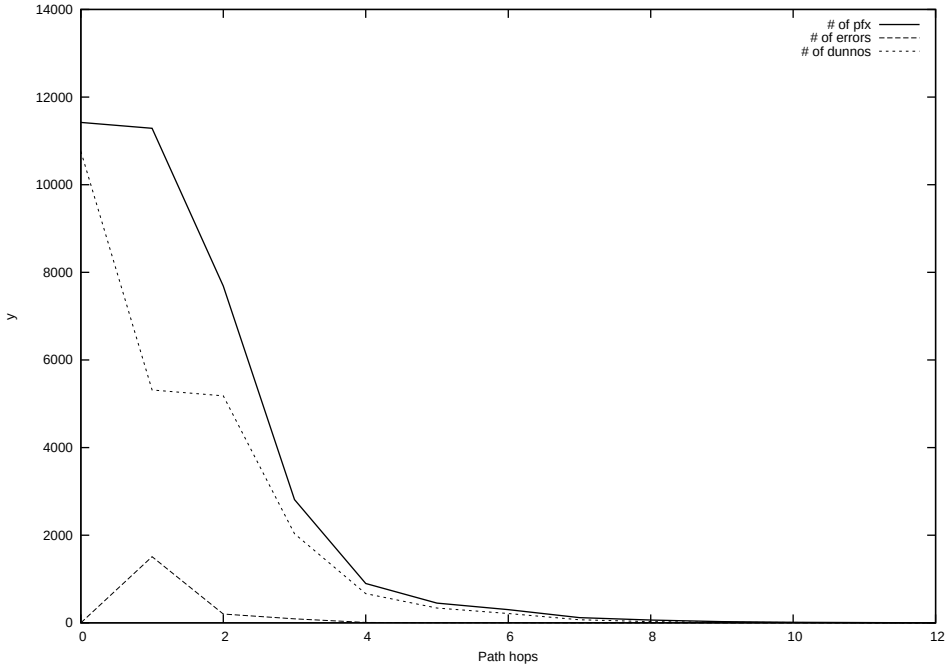
2013-01-14



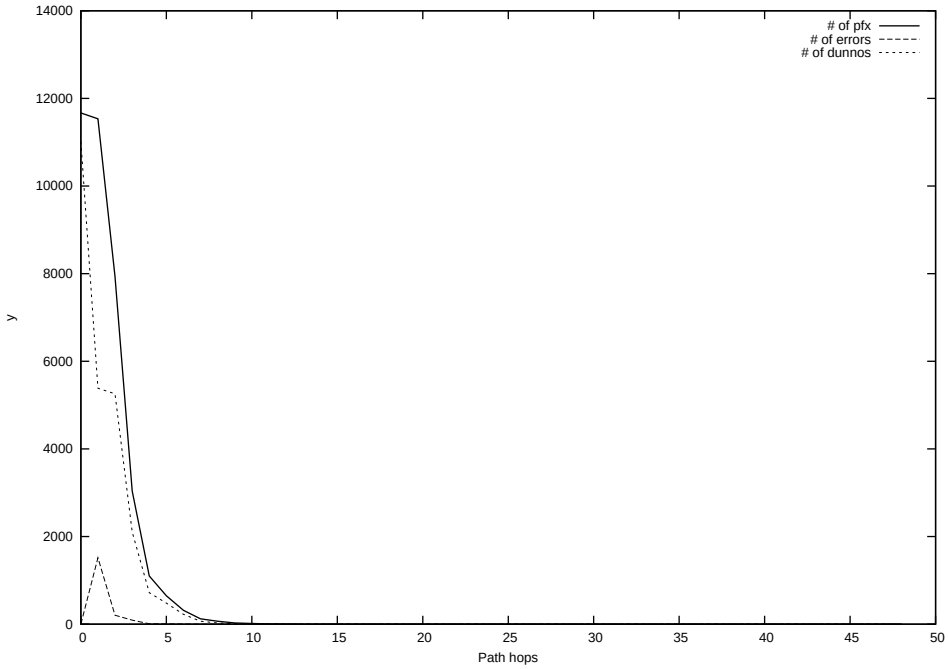
2013-01-15



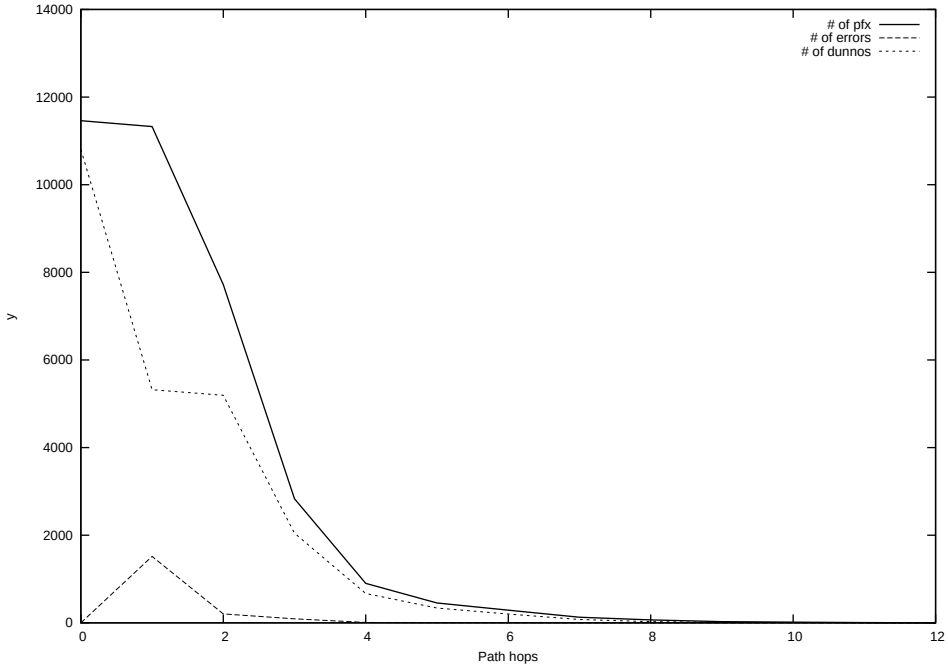
2013-01-16



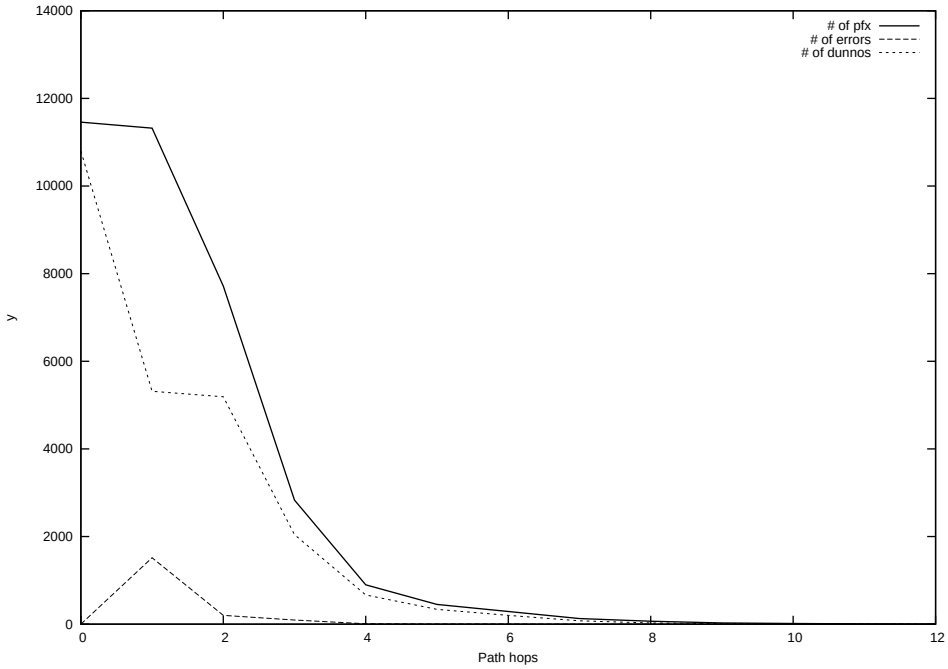
2013-01-17



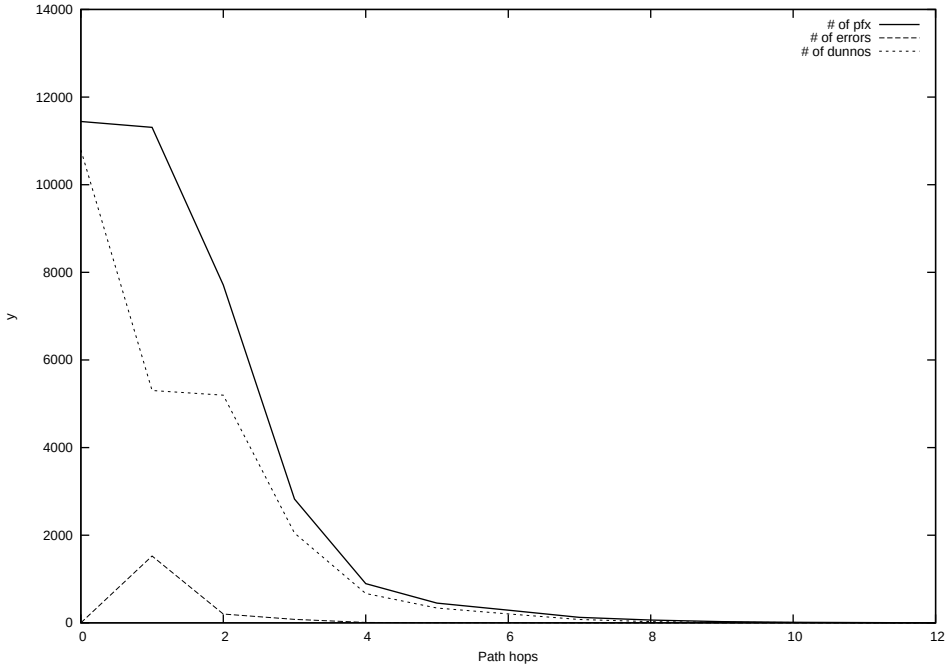
2013-01-18



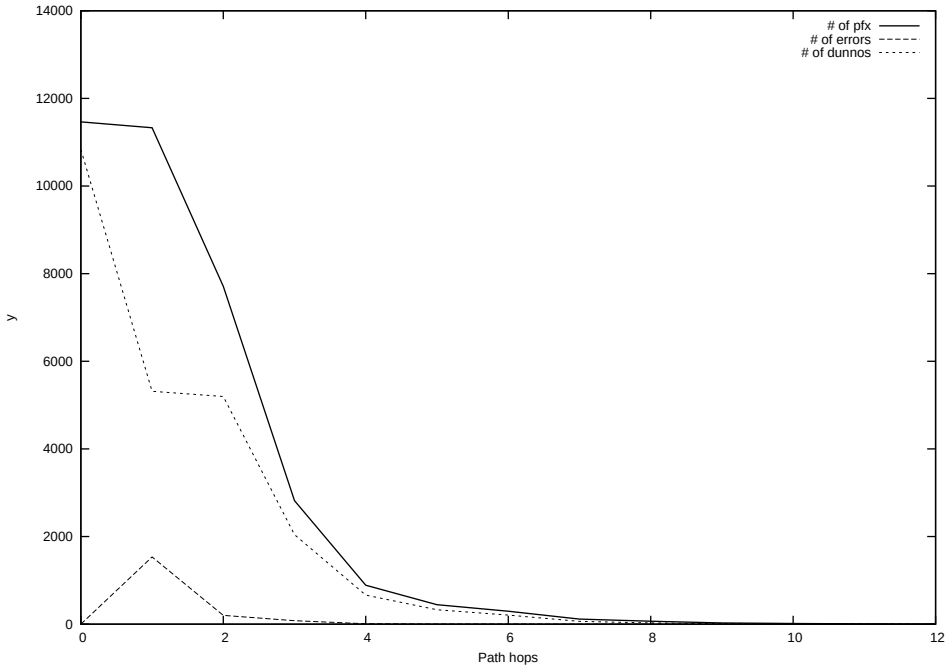
2013-01-19



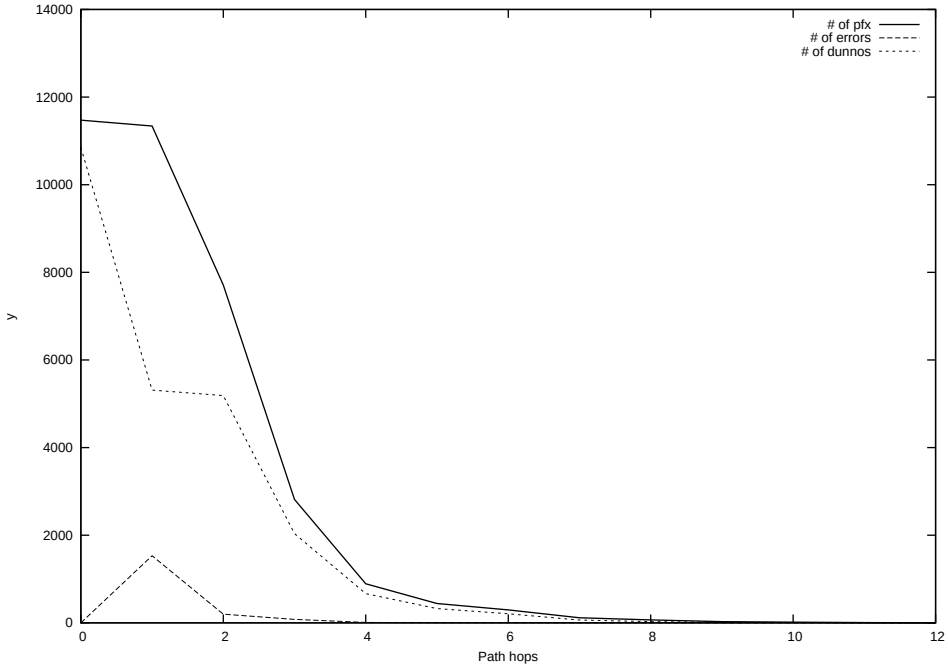
2013-01-20



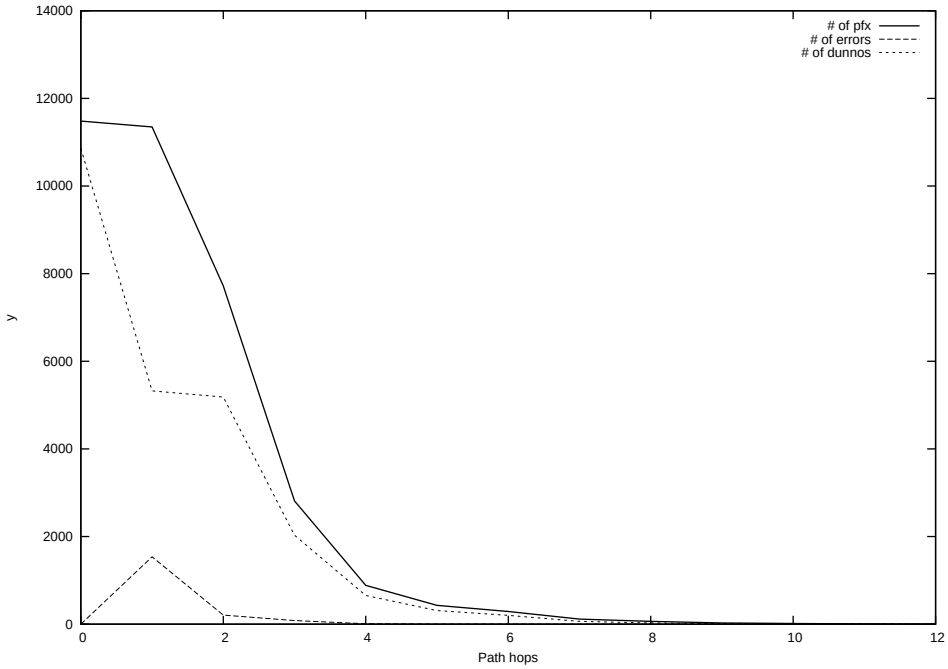
2013-01-21



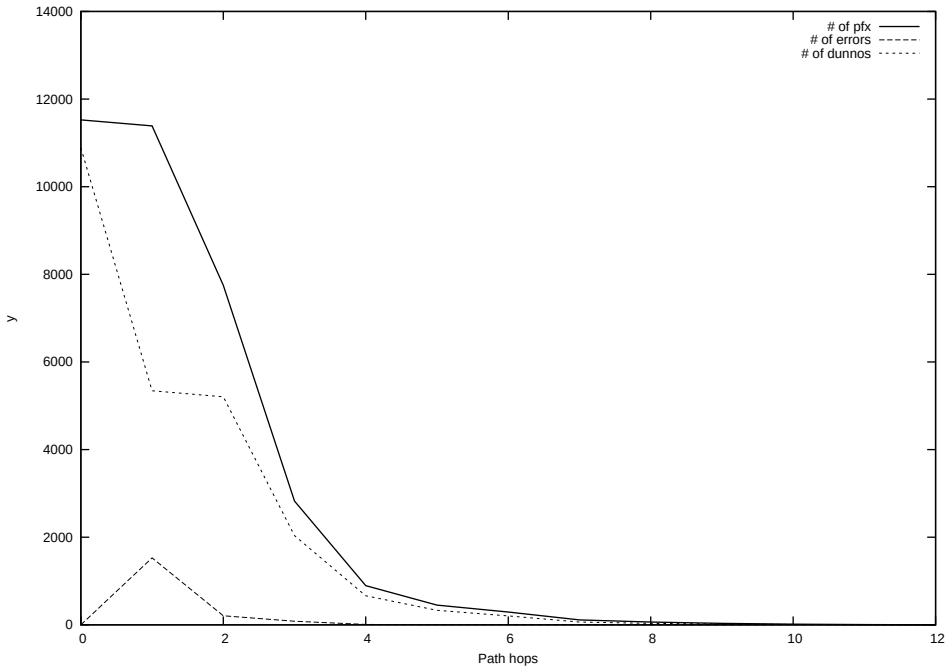
2013-01-22



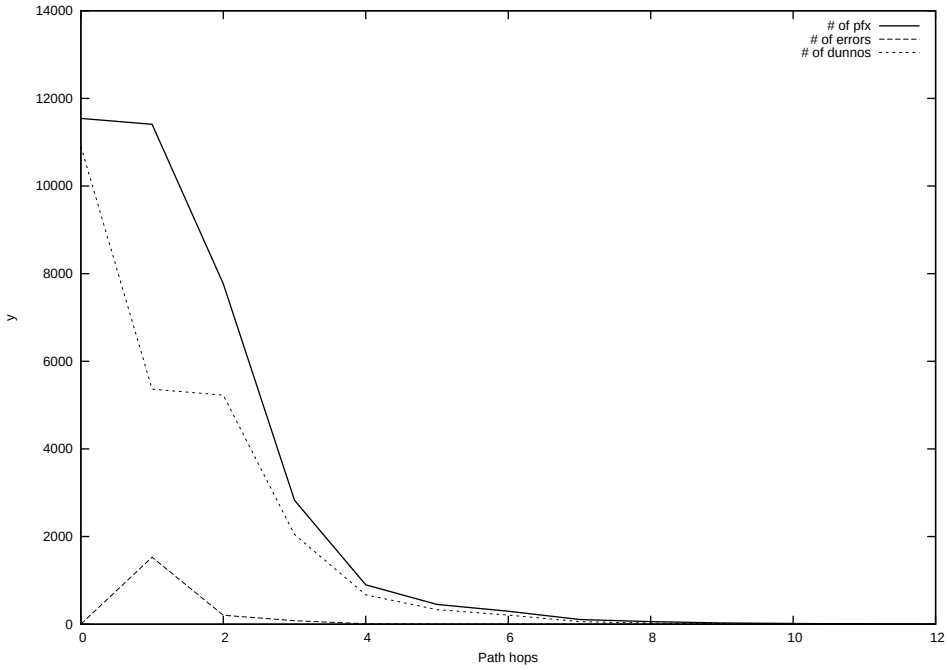
2013-01-23



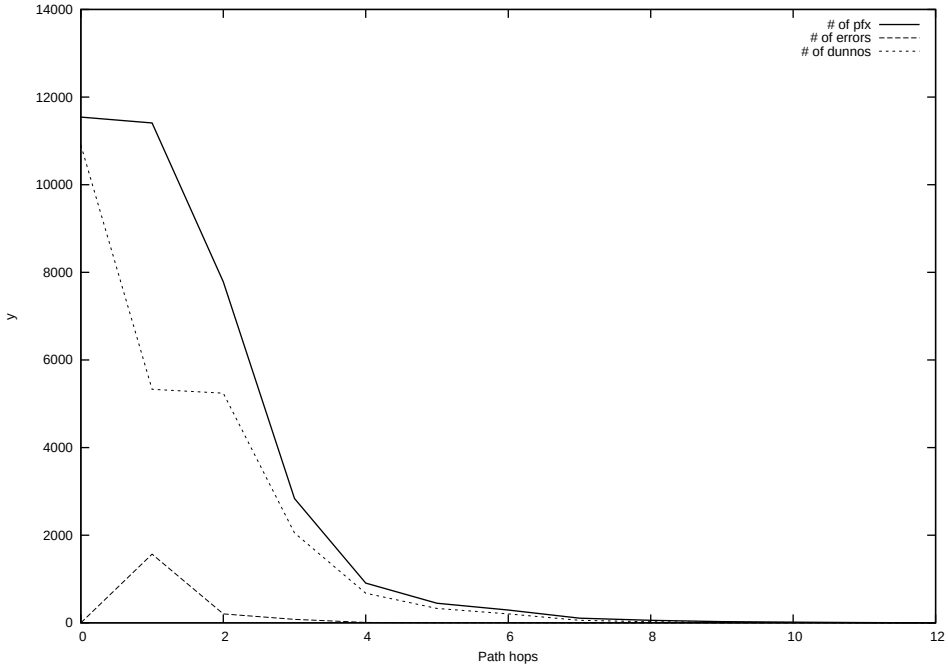
2013-01-24



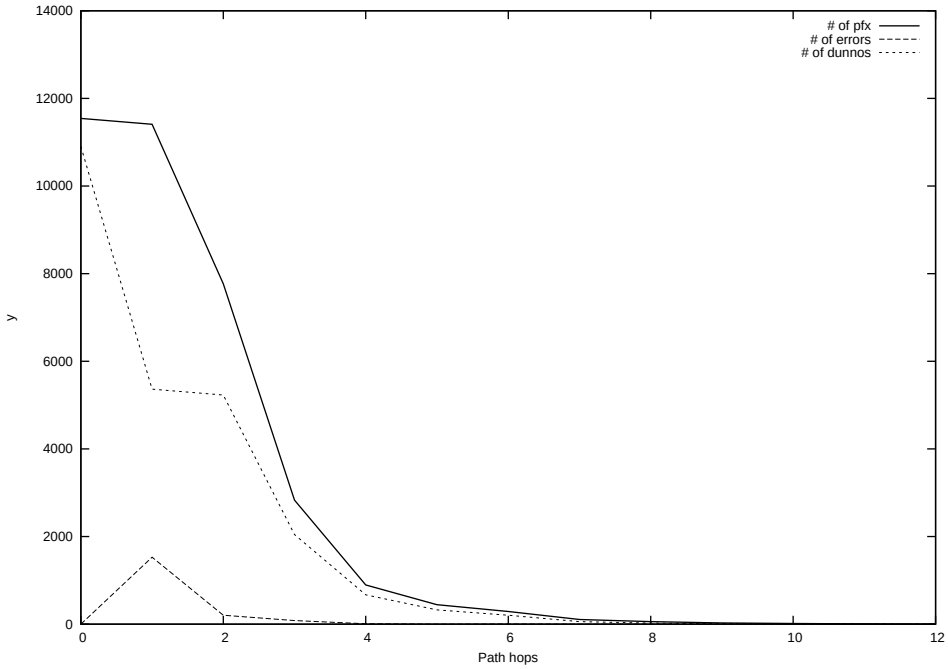
2013-01-25



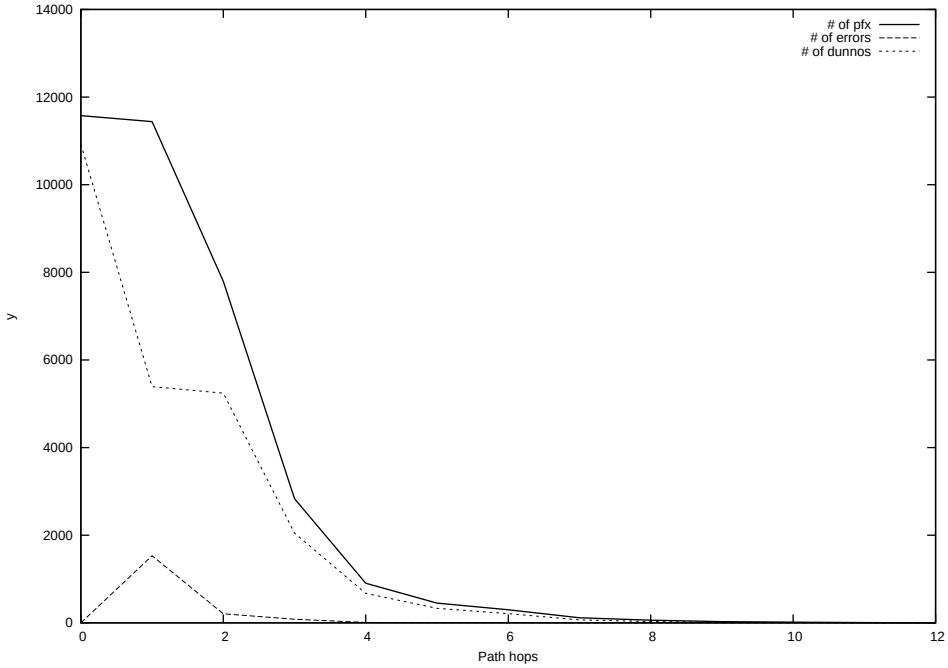
2013-01-26



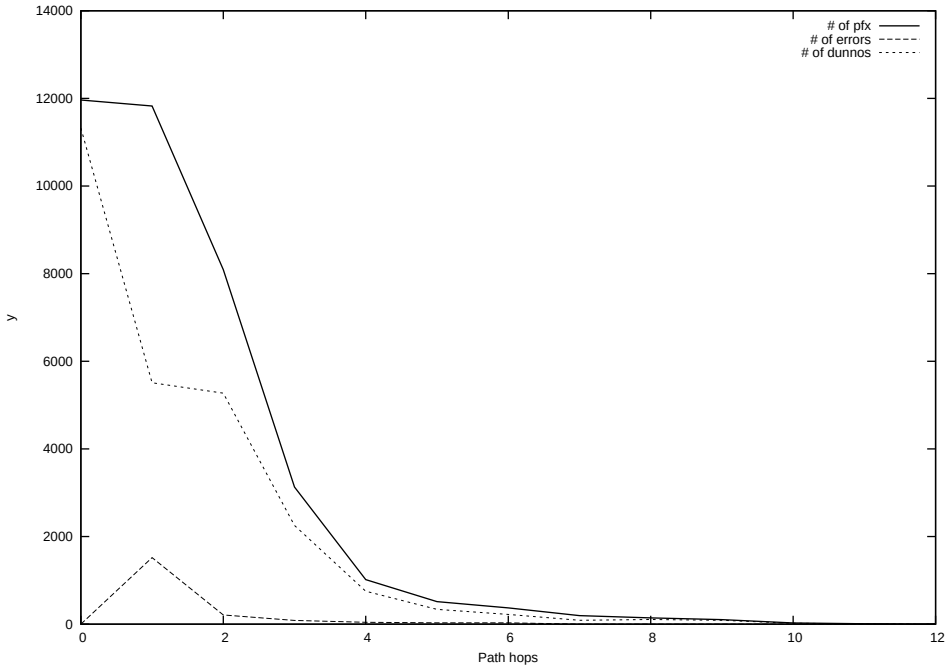
2013-01-27



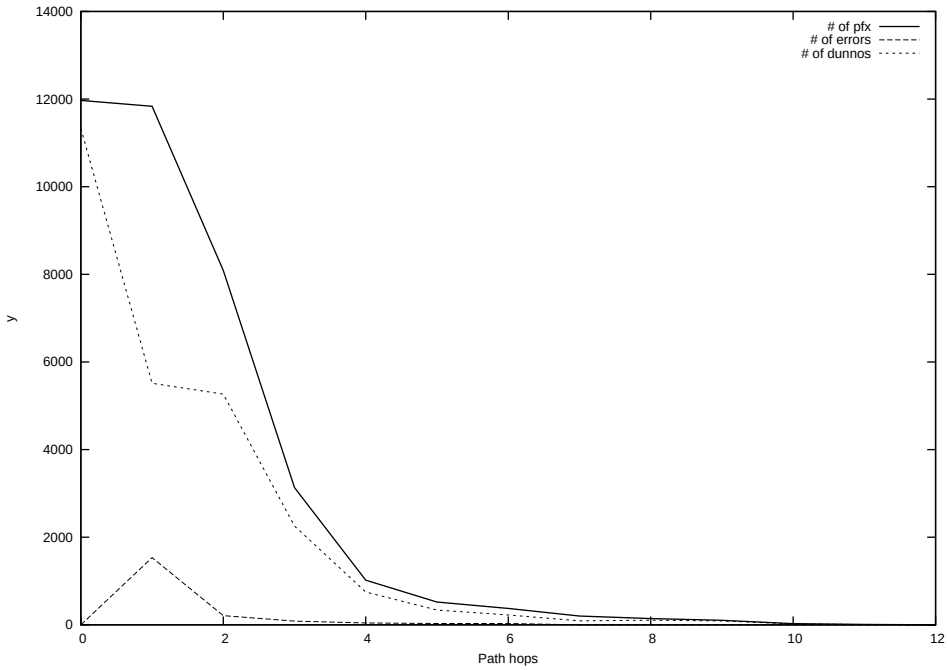
2013-01-28



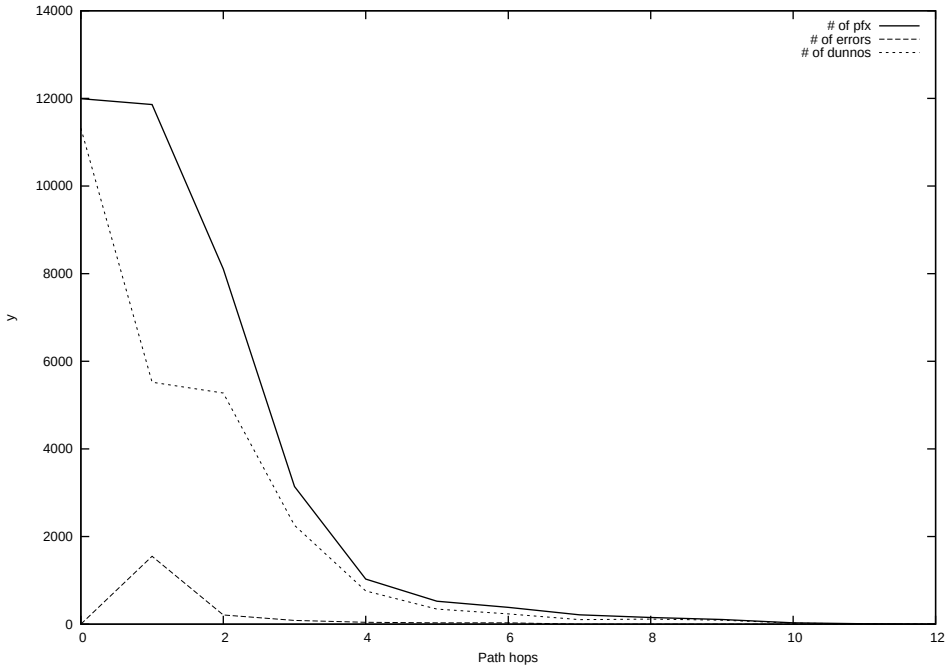
2013-01-29



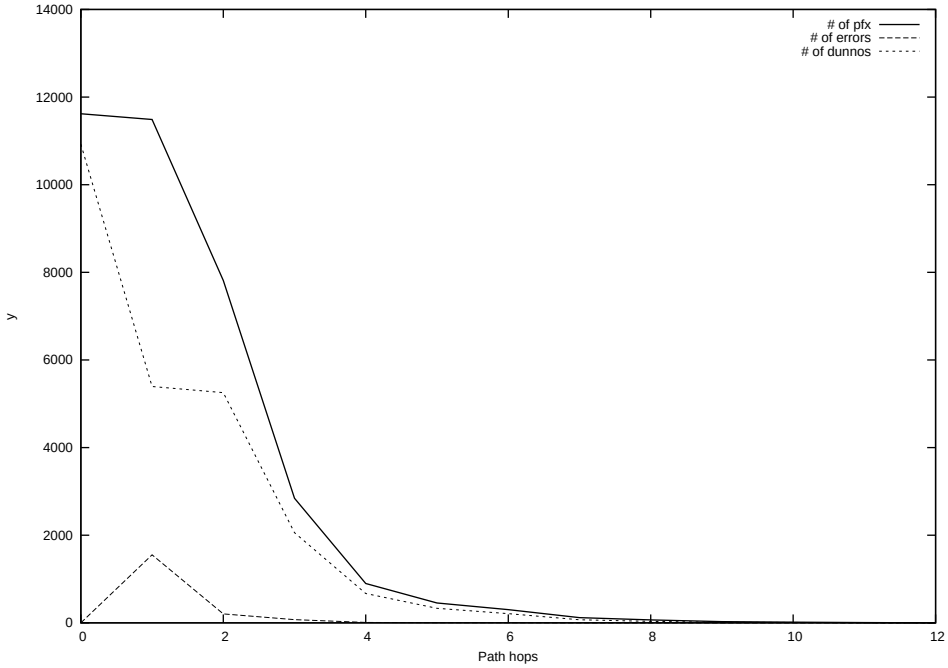
2013-01-30



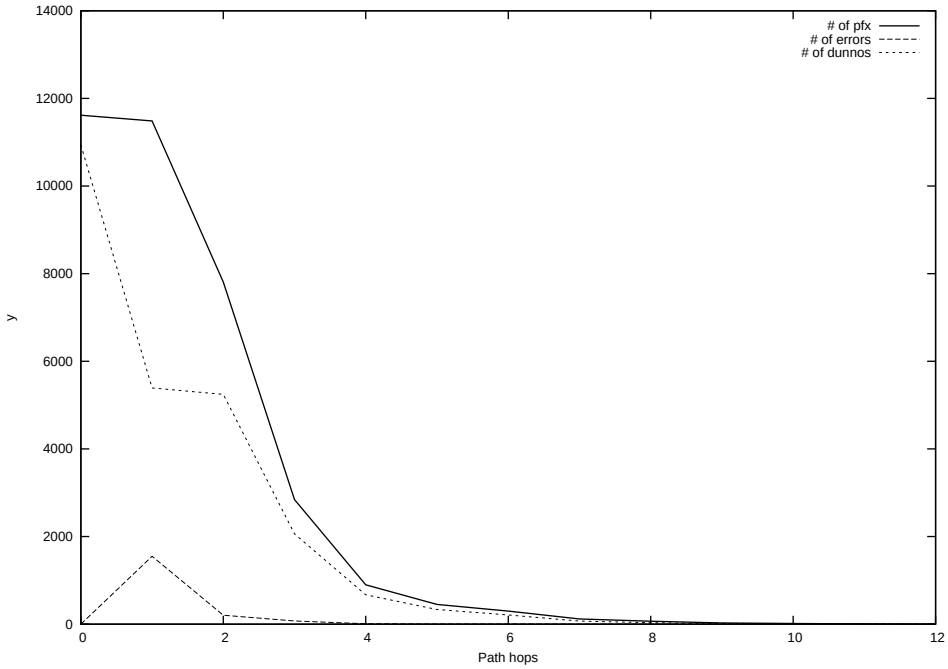
2013-01-31



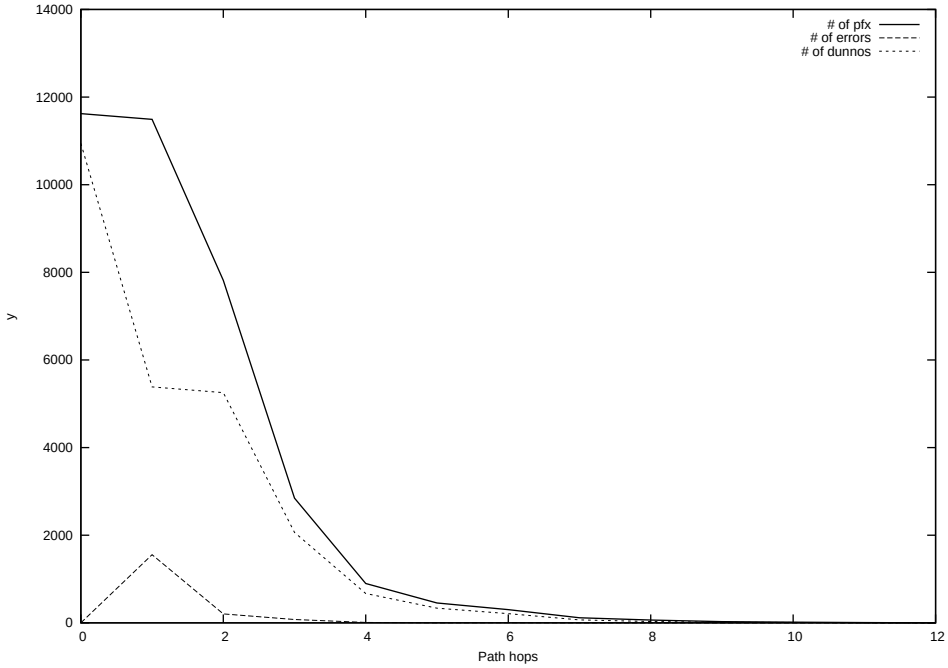
2013-02-01



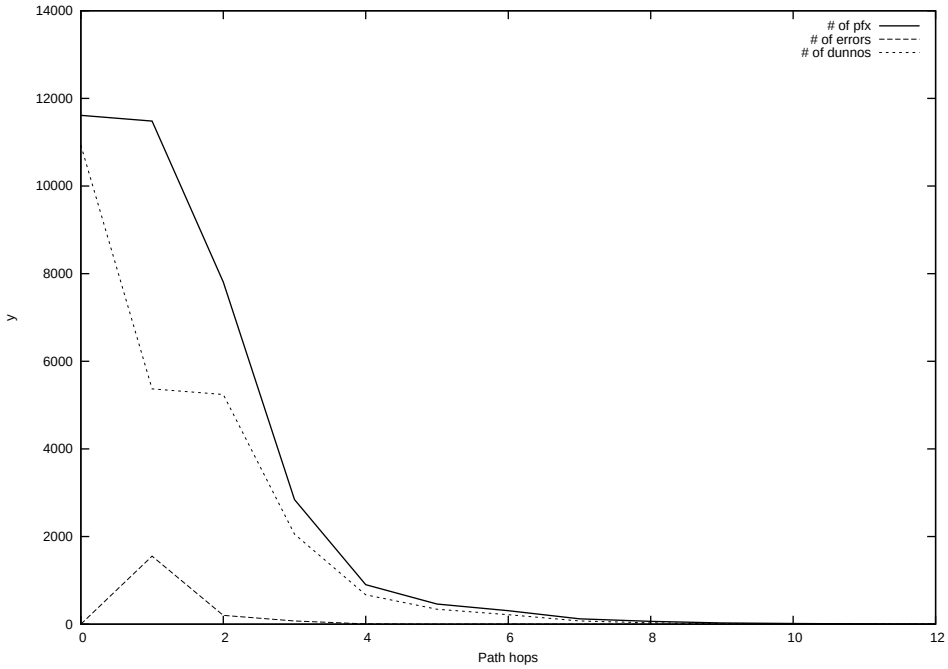
2013-02-02



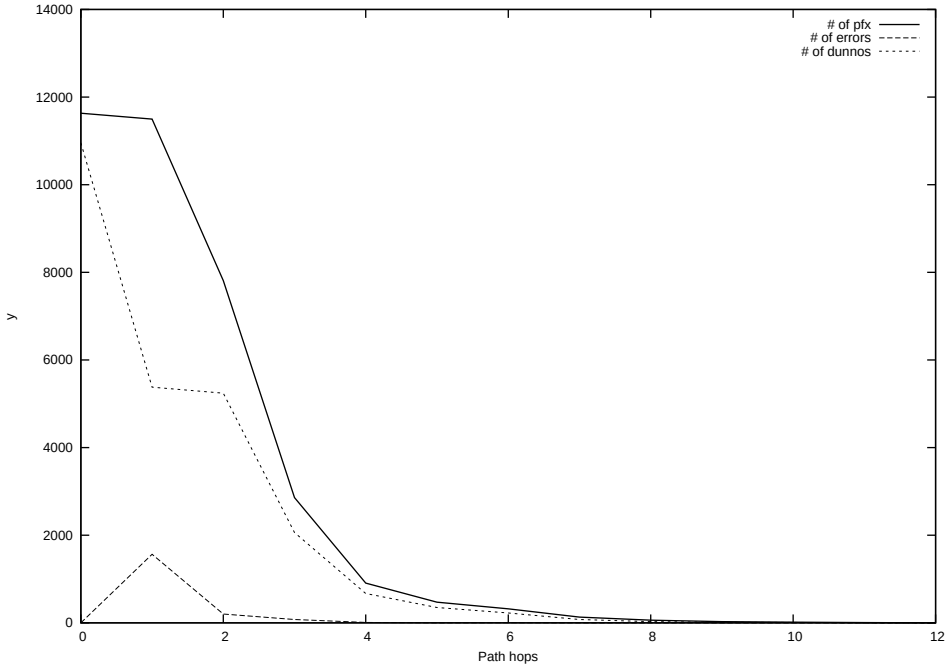
2013-02-03



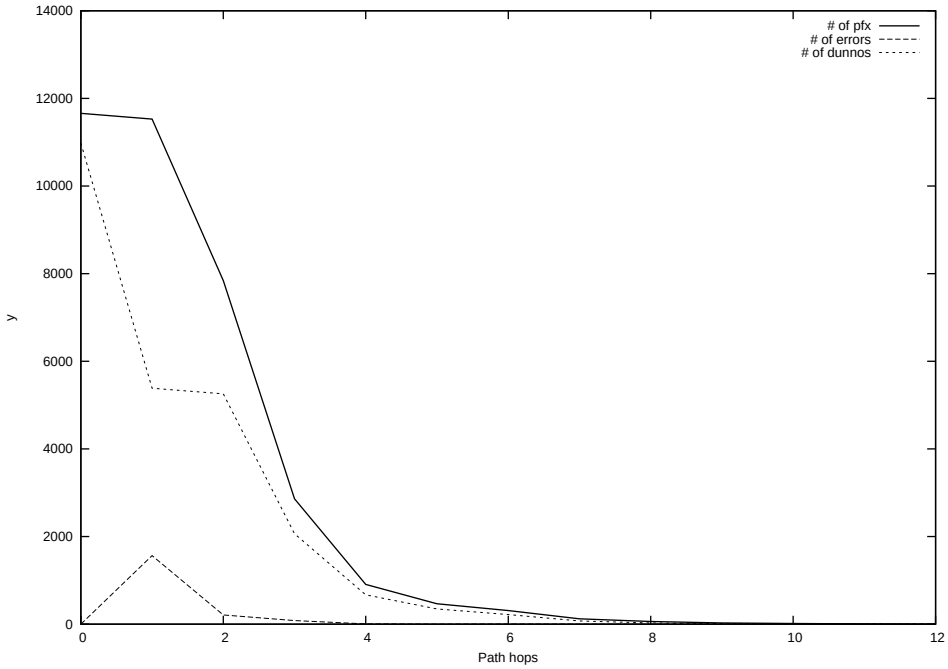
2013-02-04



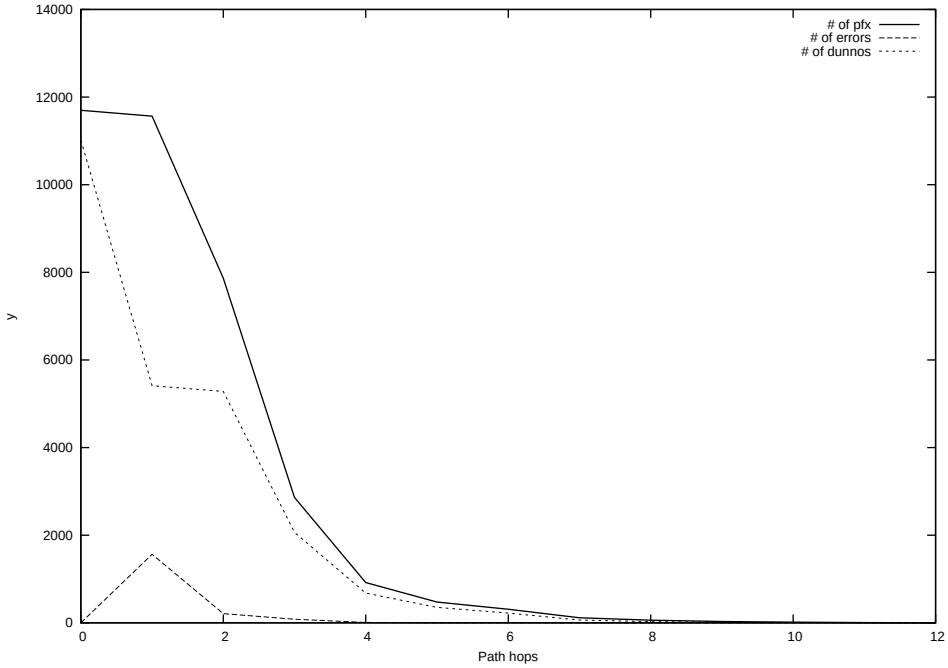
2013-02-05



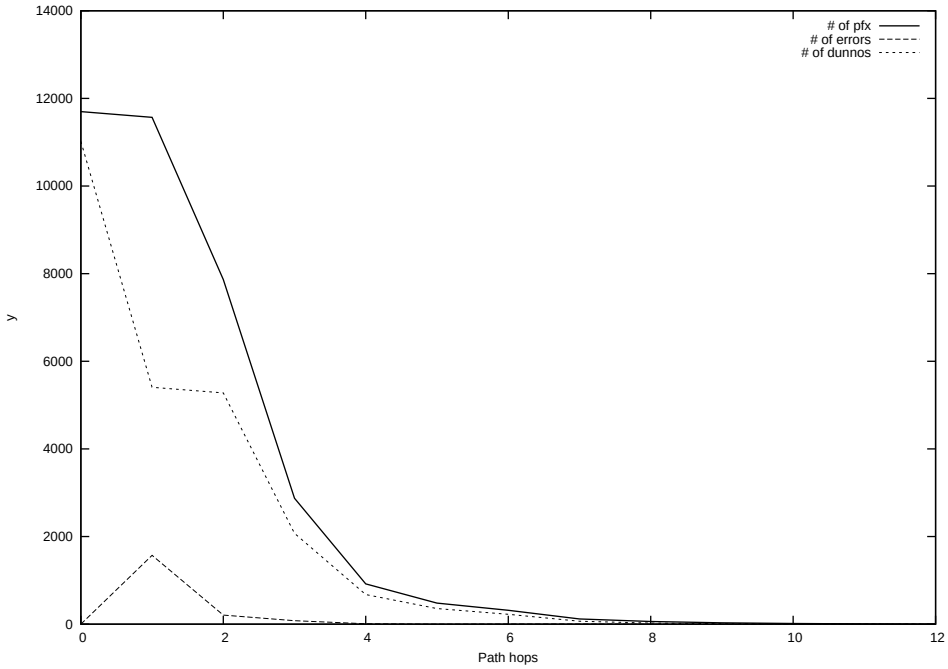
2013-02-06



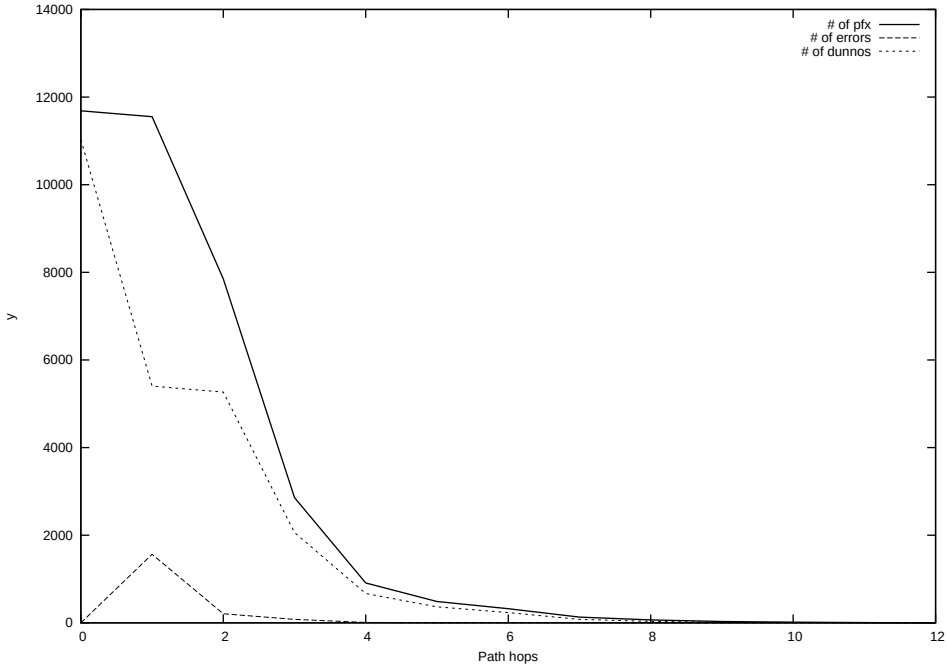
2013-02-07



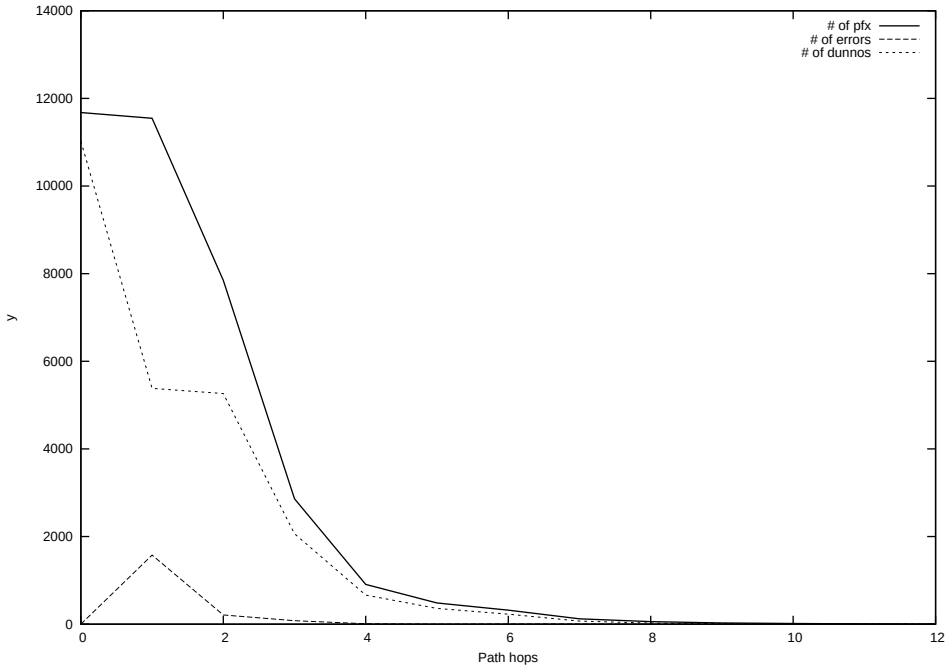
2013-02-08



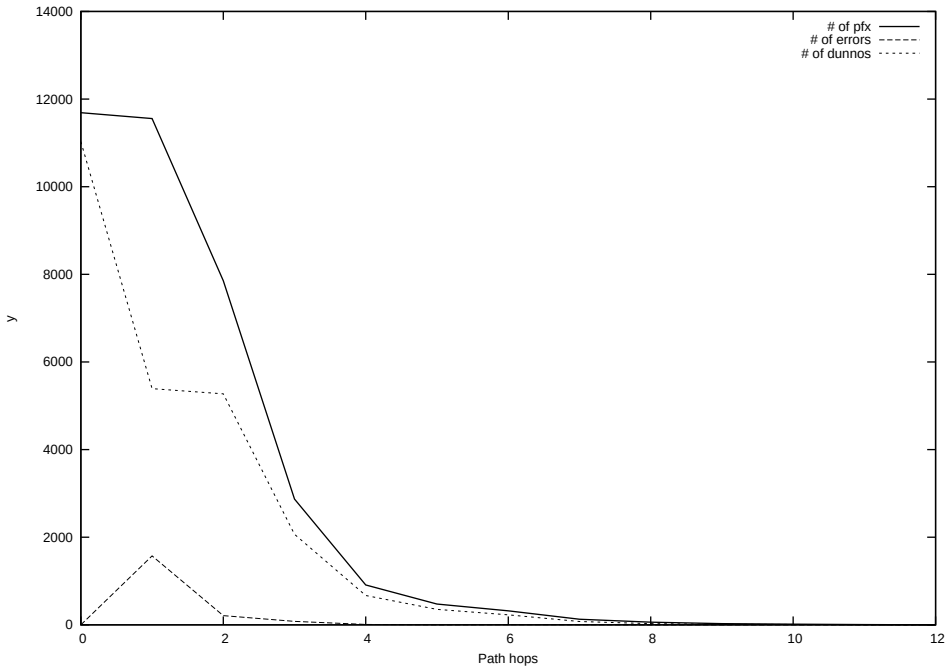
2013-02-09



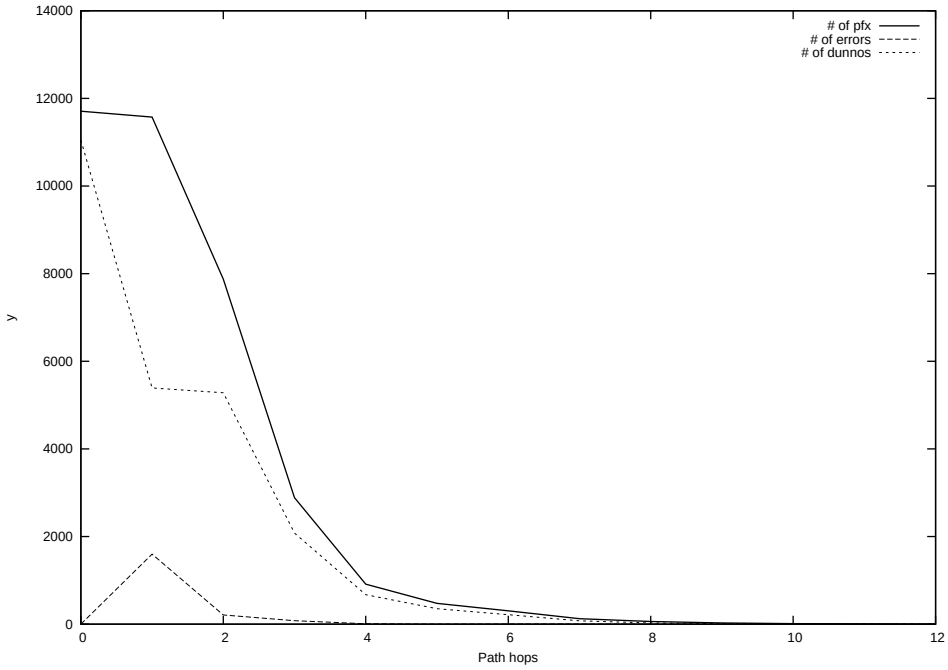
2013-02-10



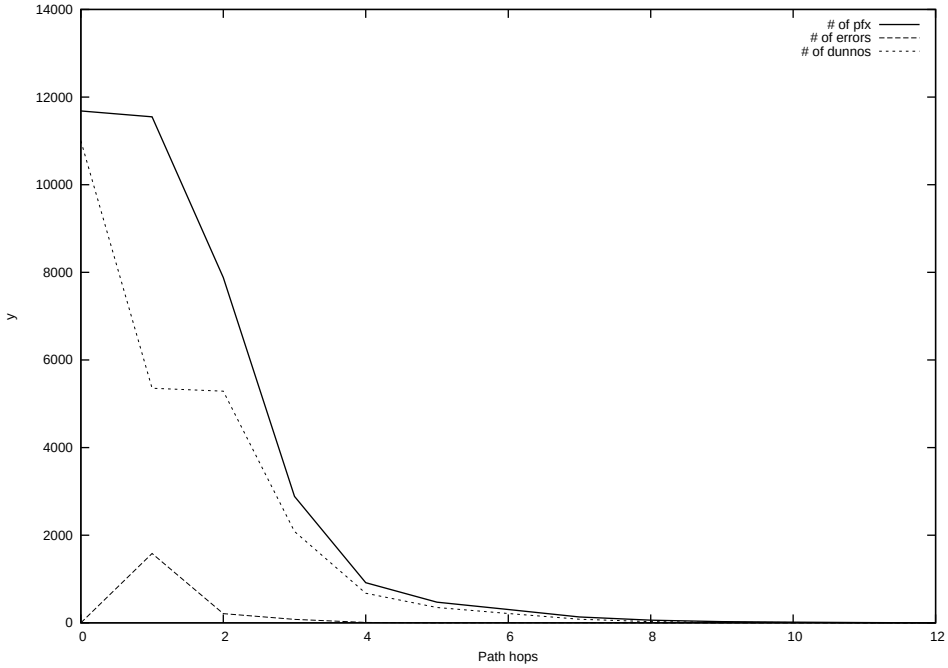
2013-02-11



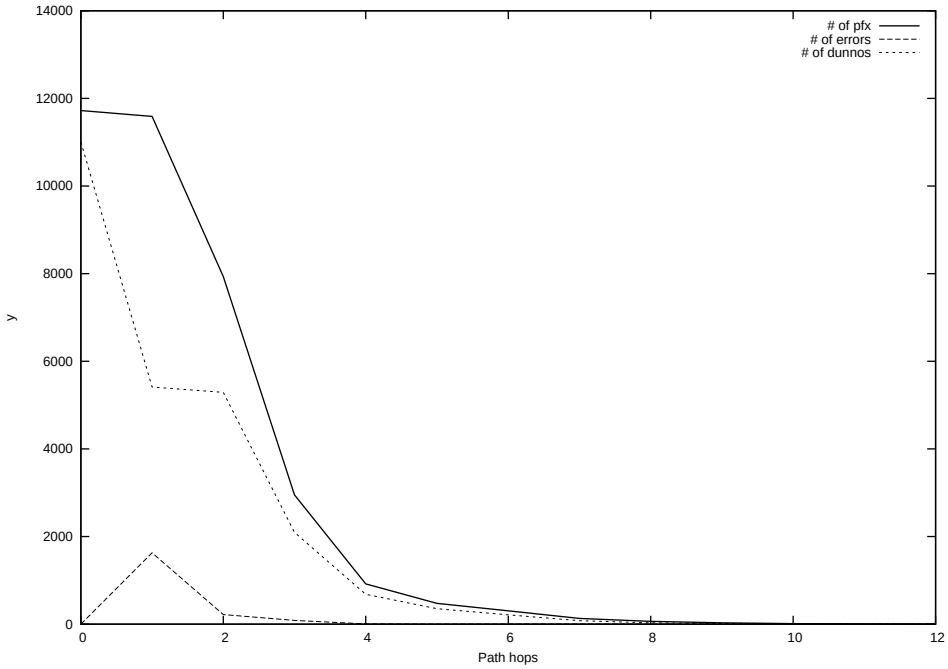
2013-02-12



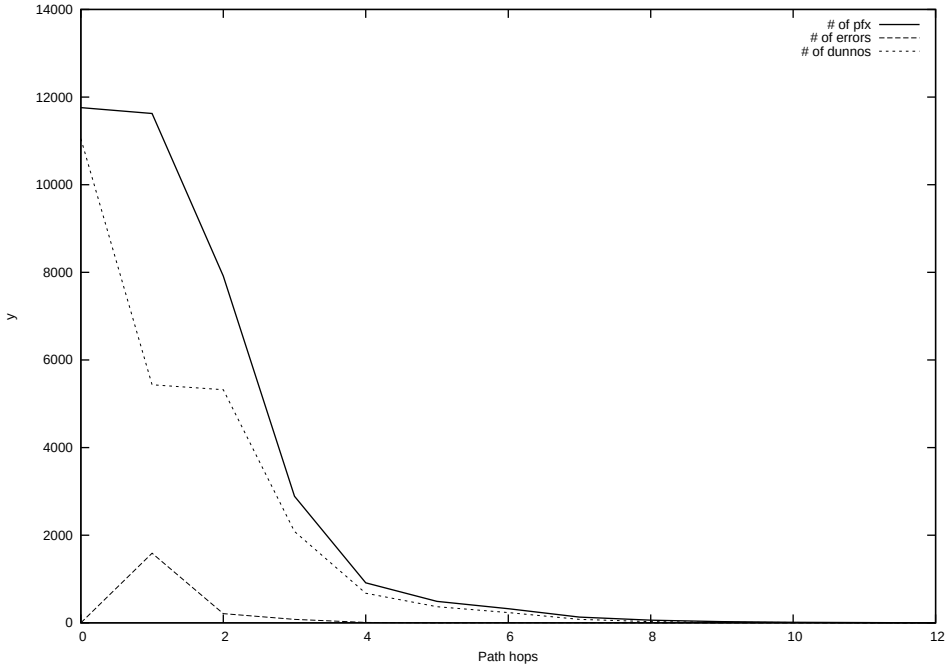
2013-02-13



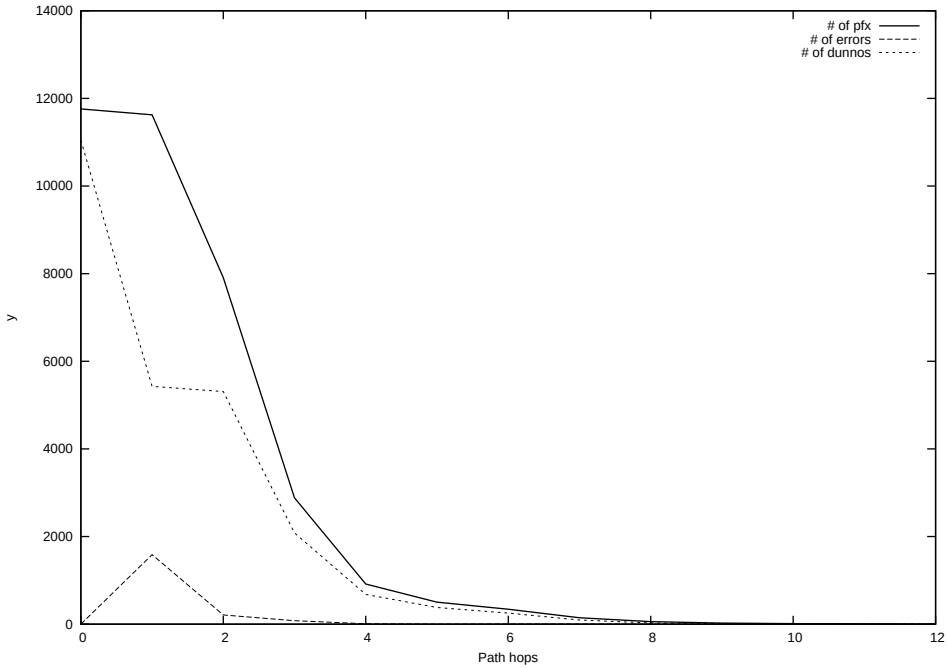
2013-02-14



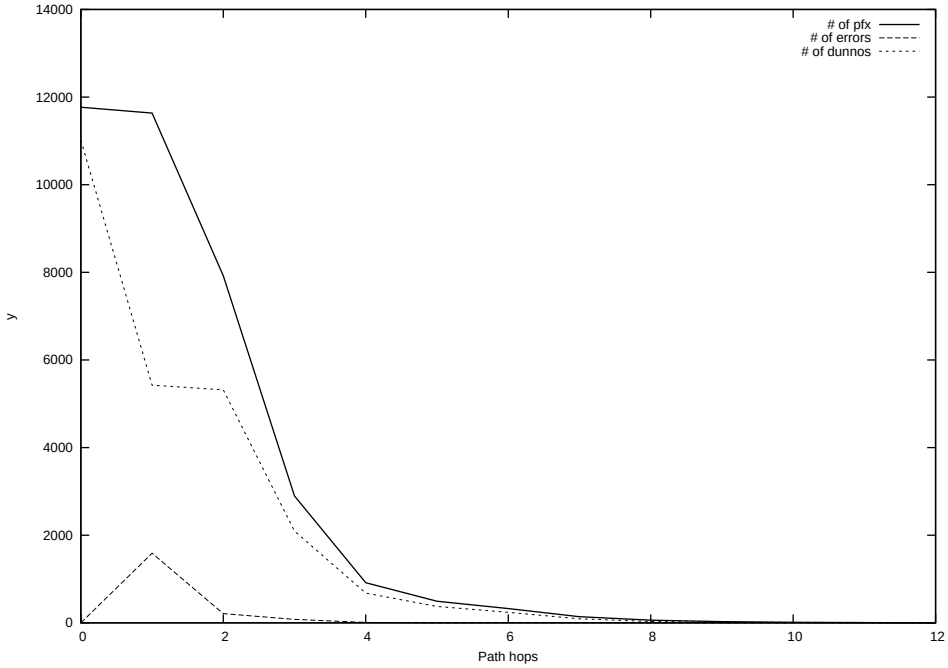
2013-02-15



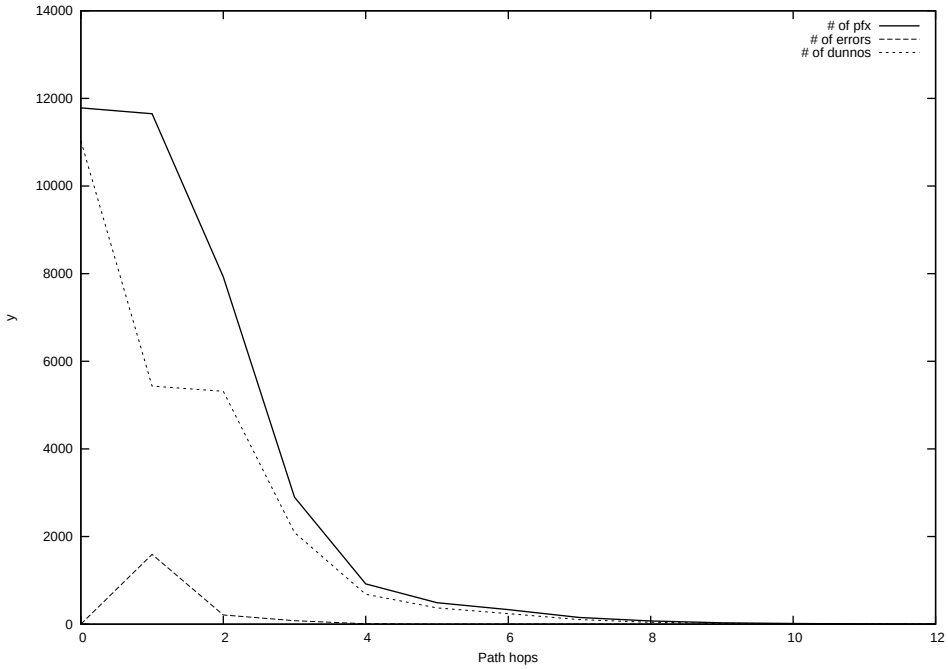
2013-02-16



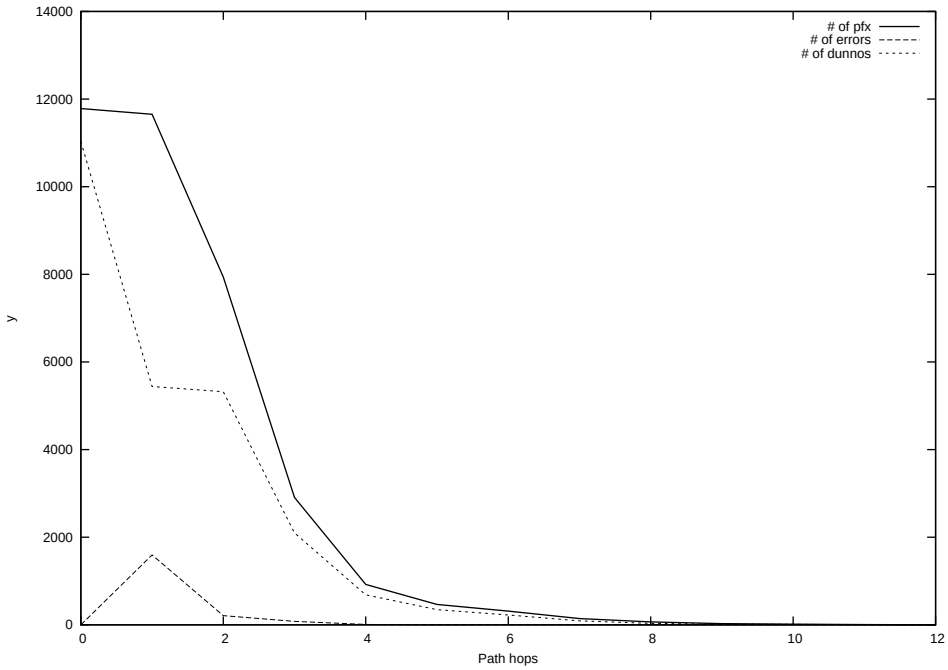
2013-02-17



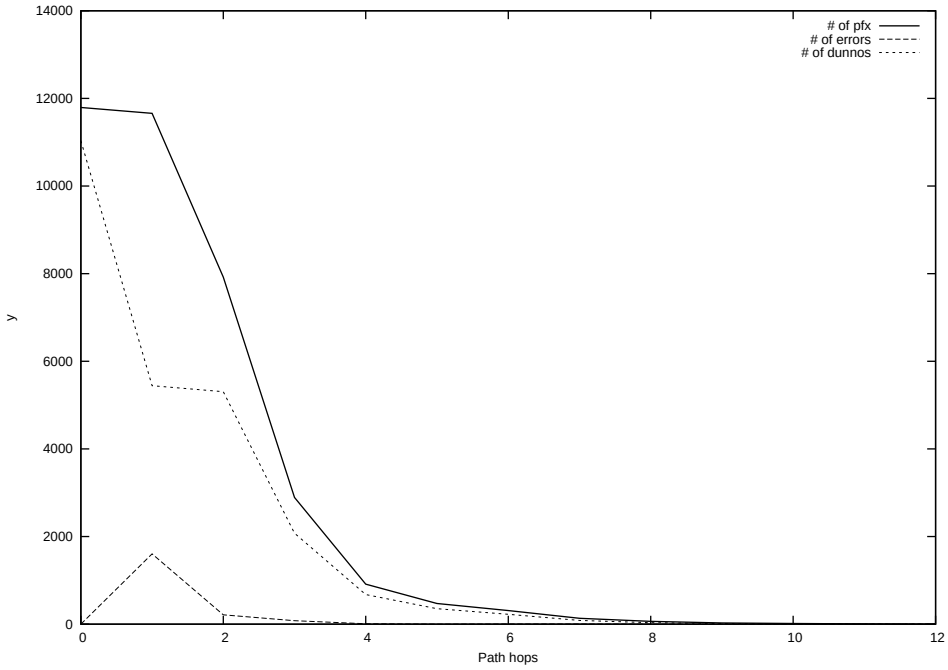
2013-02-18



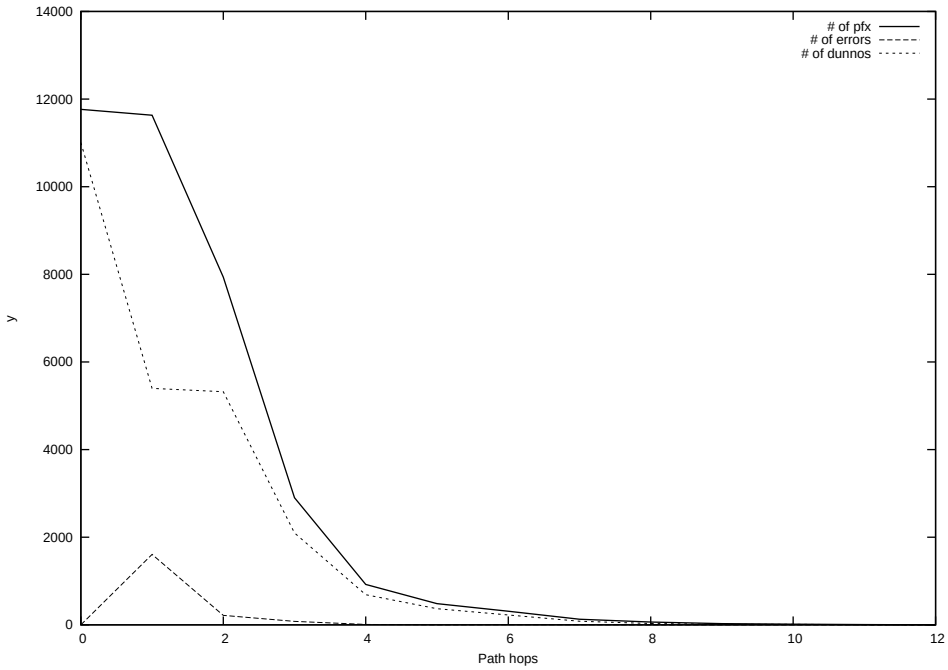
2013-02-19



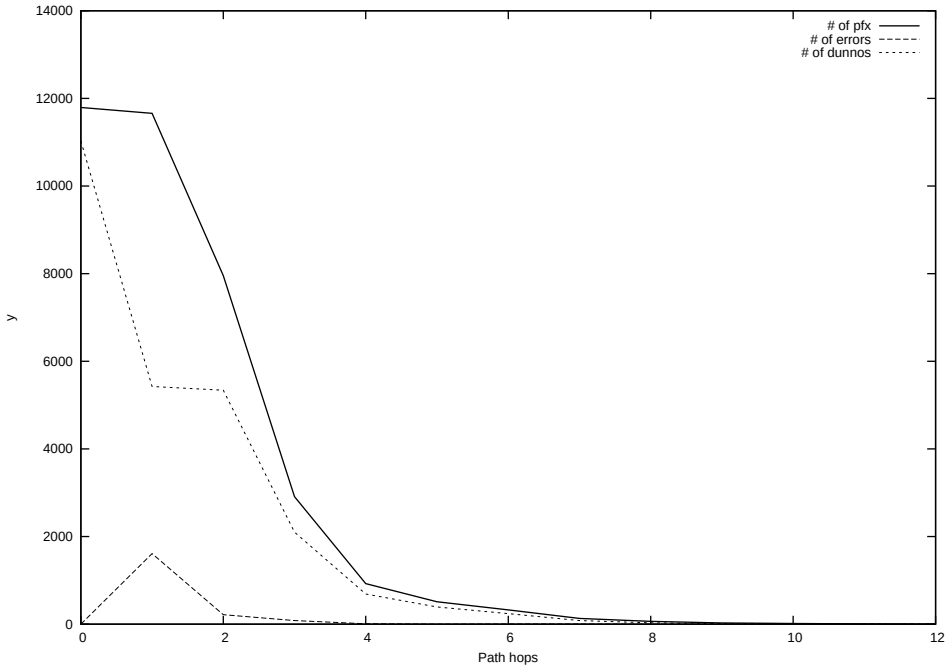
2013-02-20



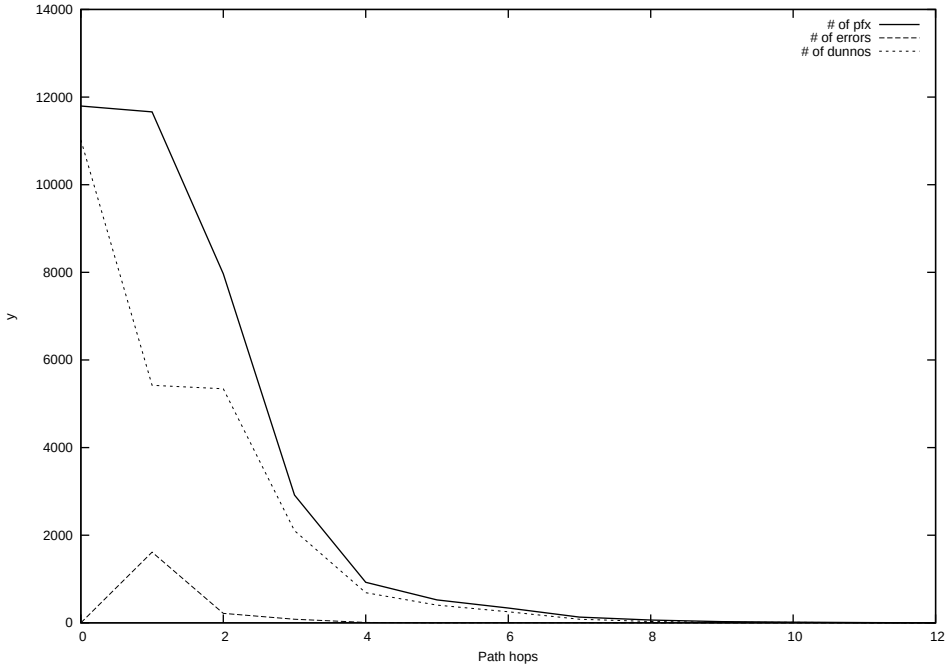
2013-02-21



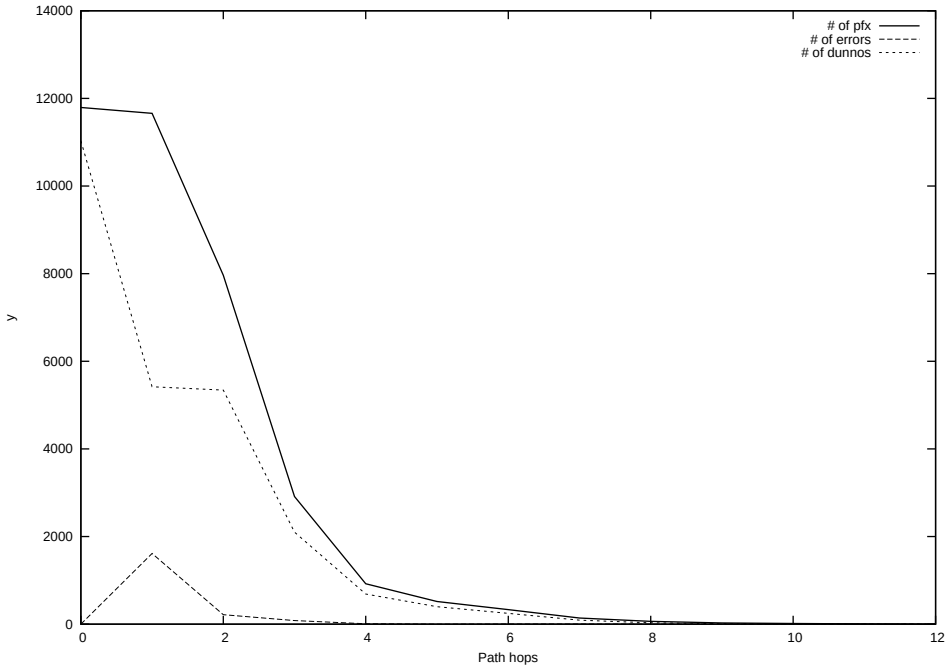
2013-02-22



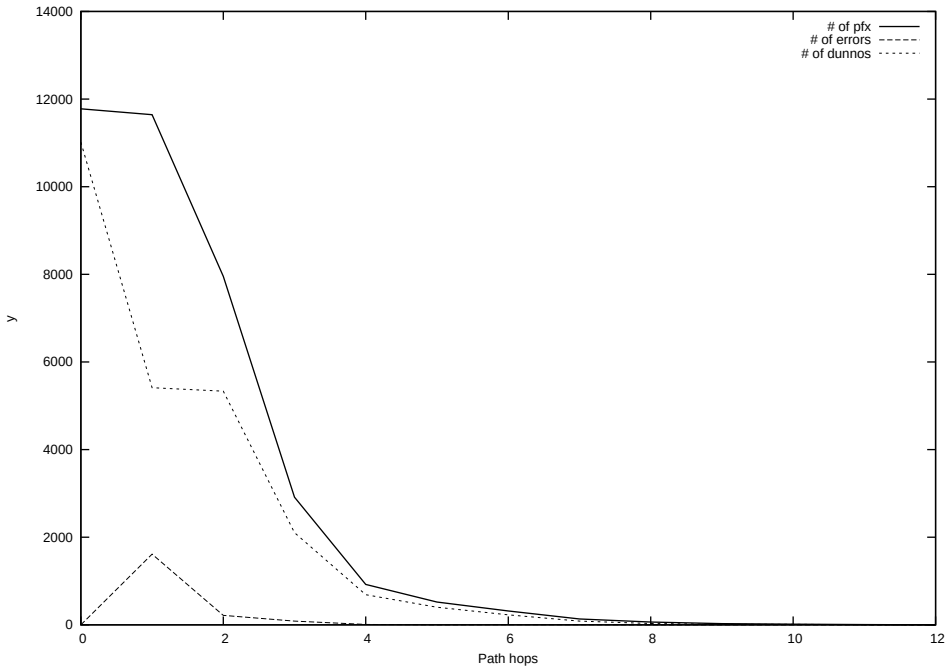
2013-02-23



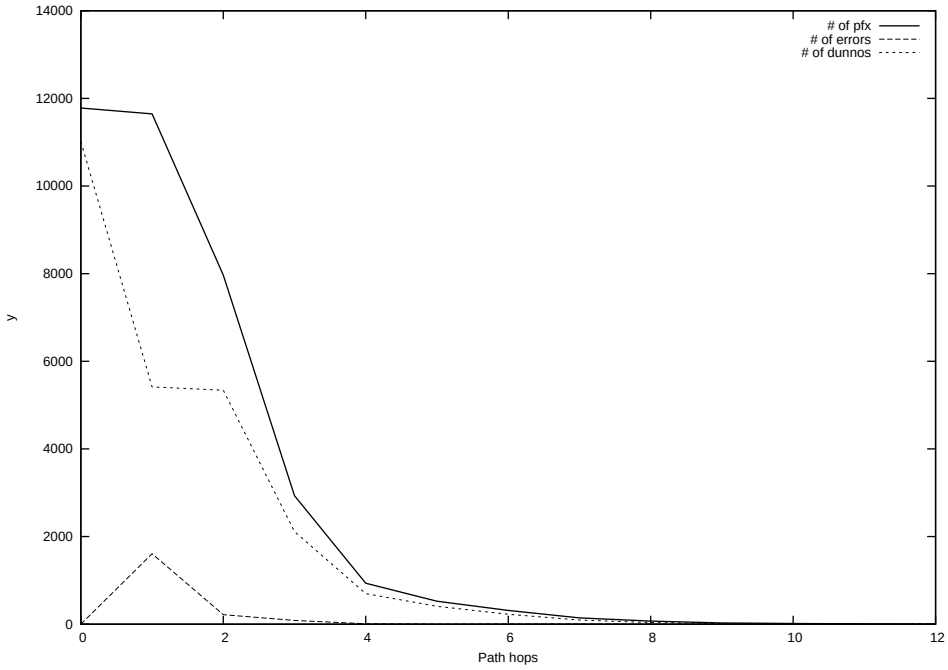
2013-02-24



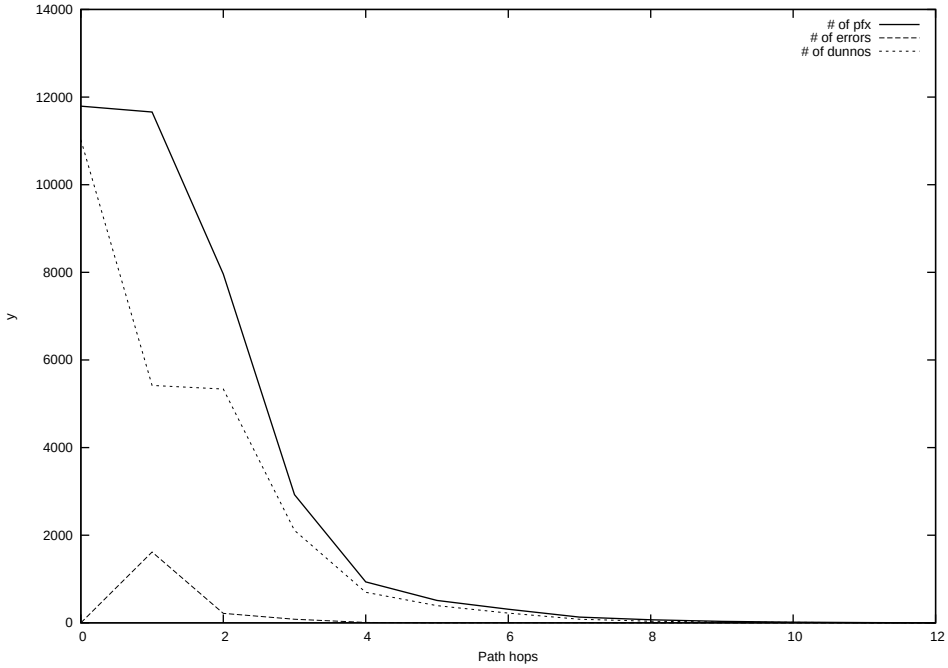
2013-02-25



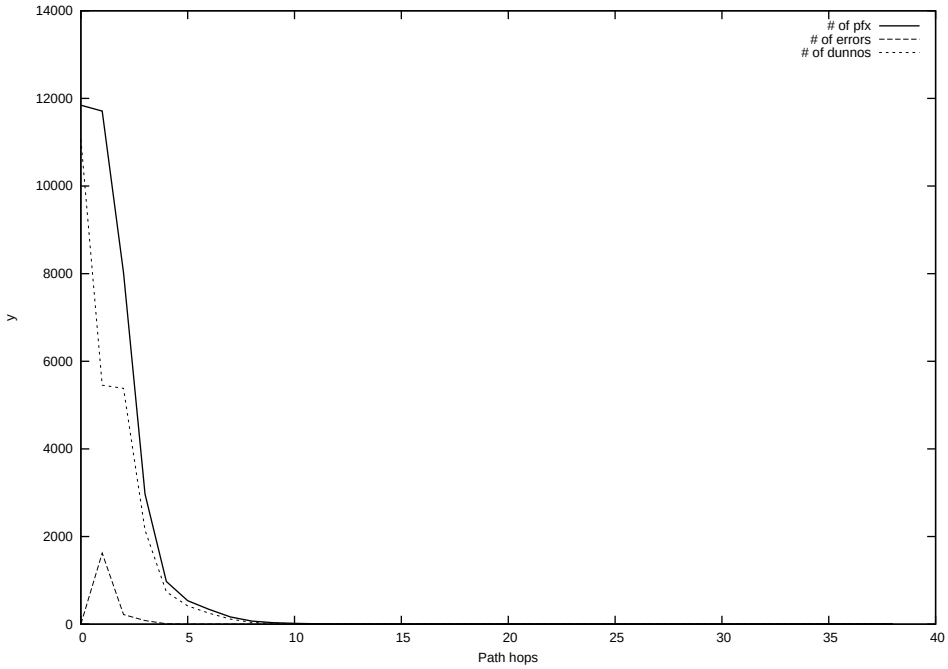
2013-02-26



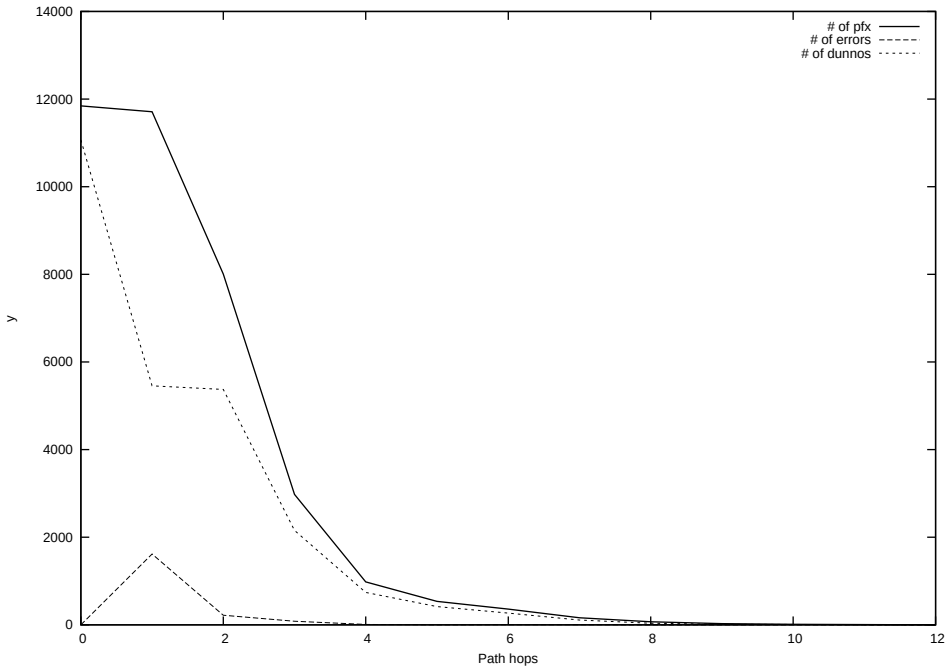
2013-02-27



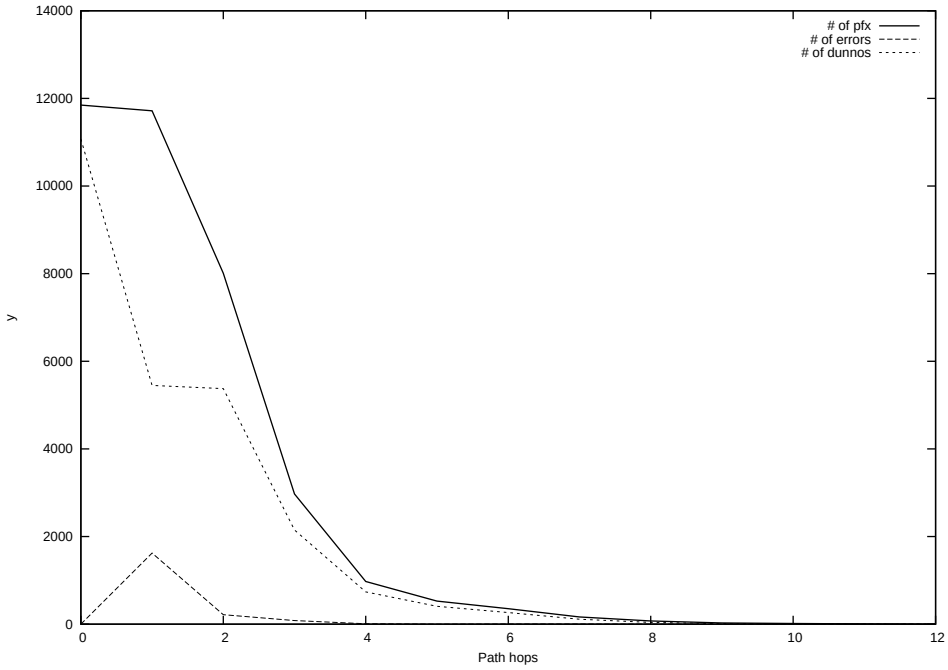
2013-02-28



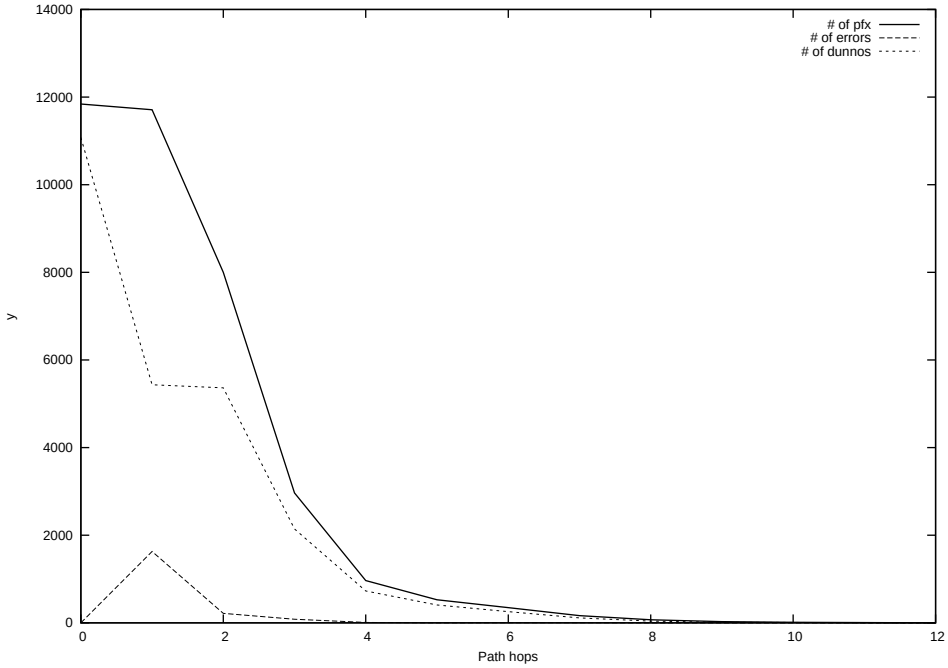
2013-03-01



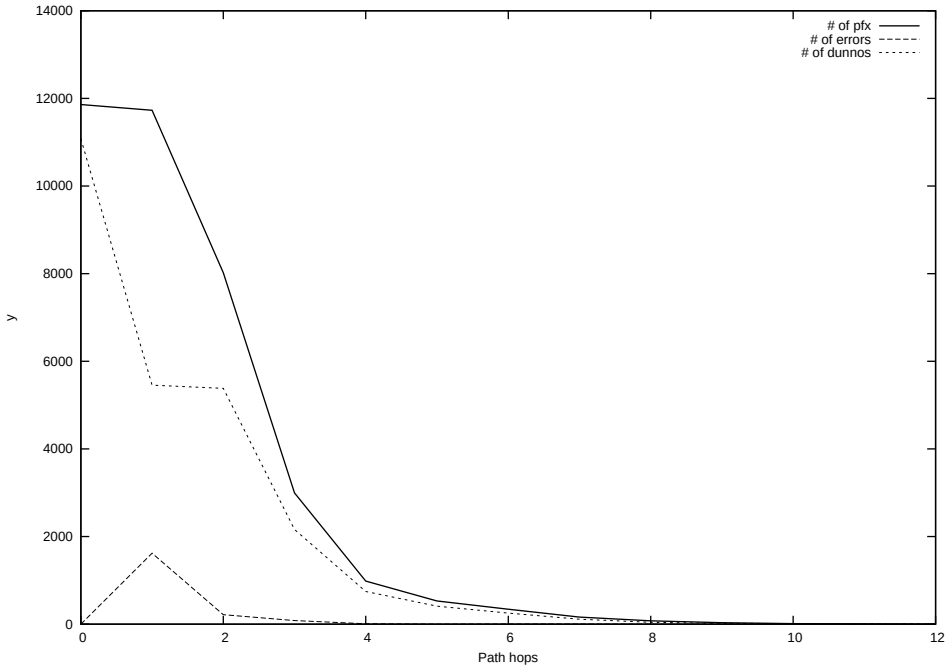
2013-03-02



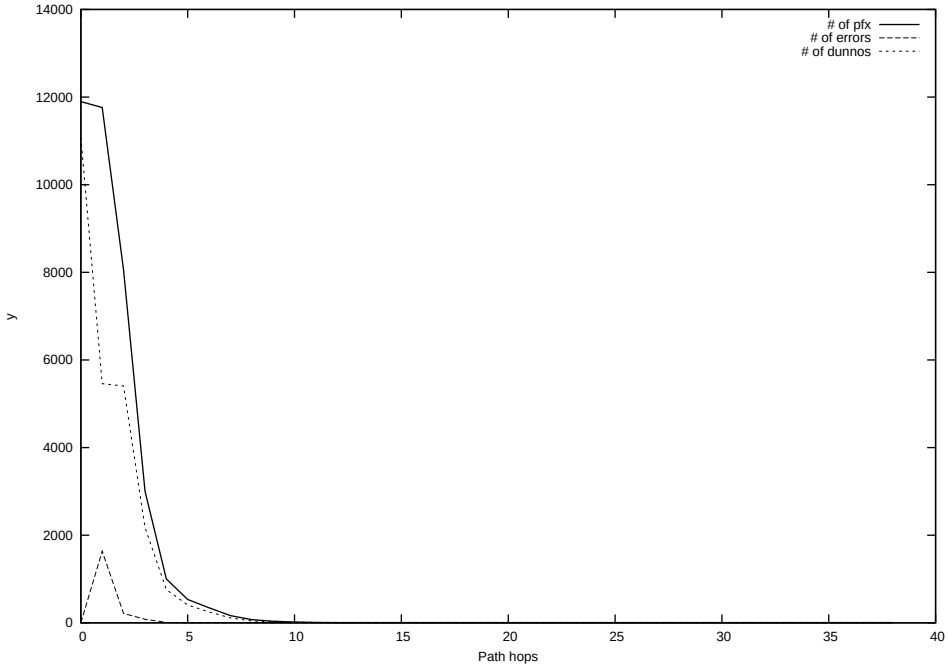
2013-03-03



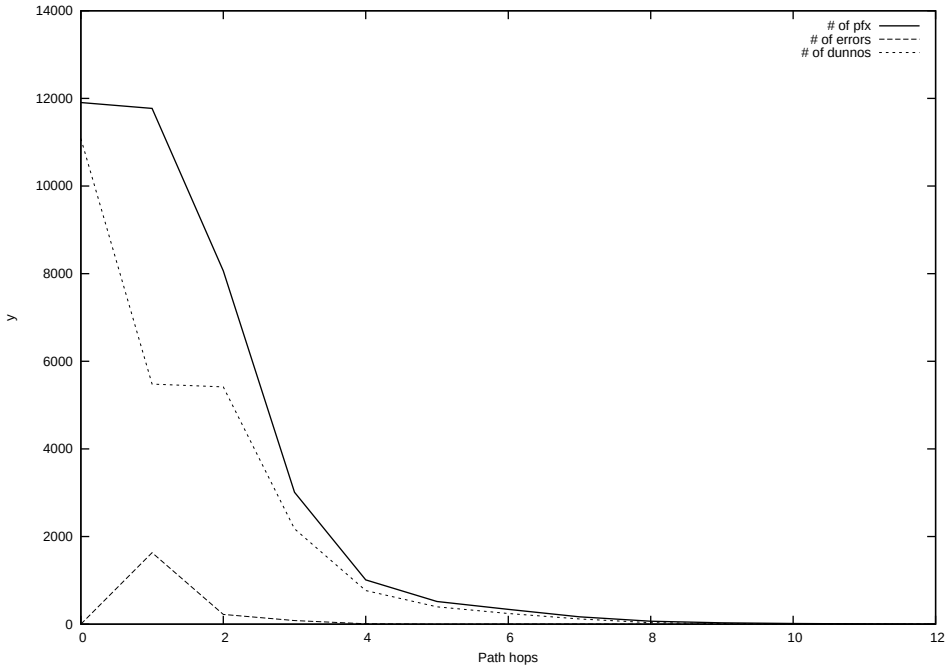
2013-03-04



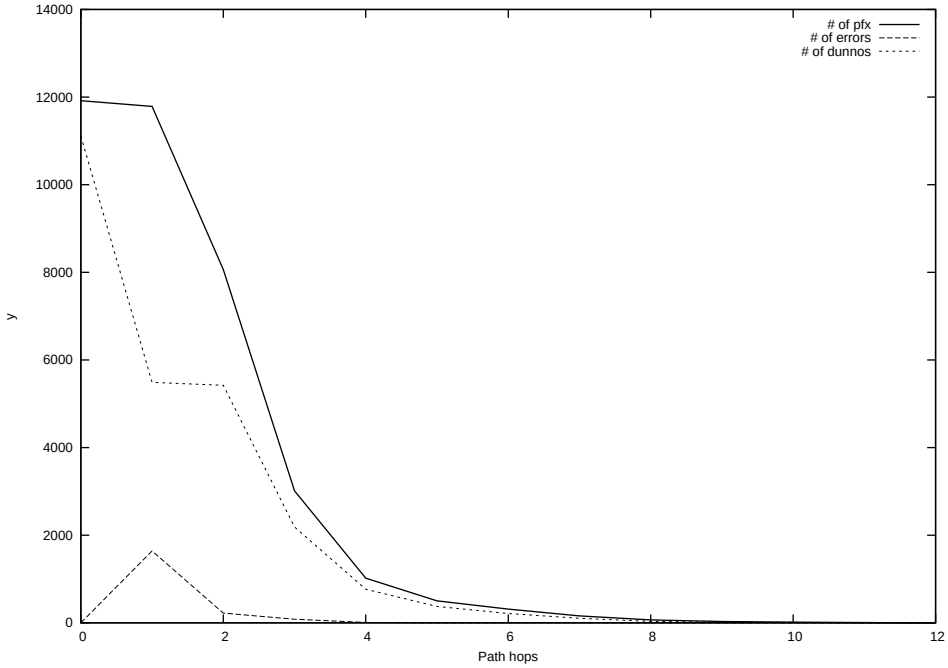
2013-03-05



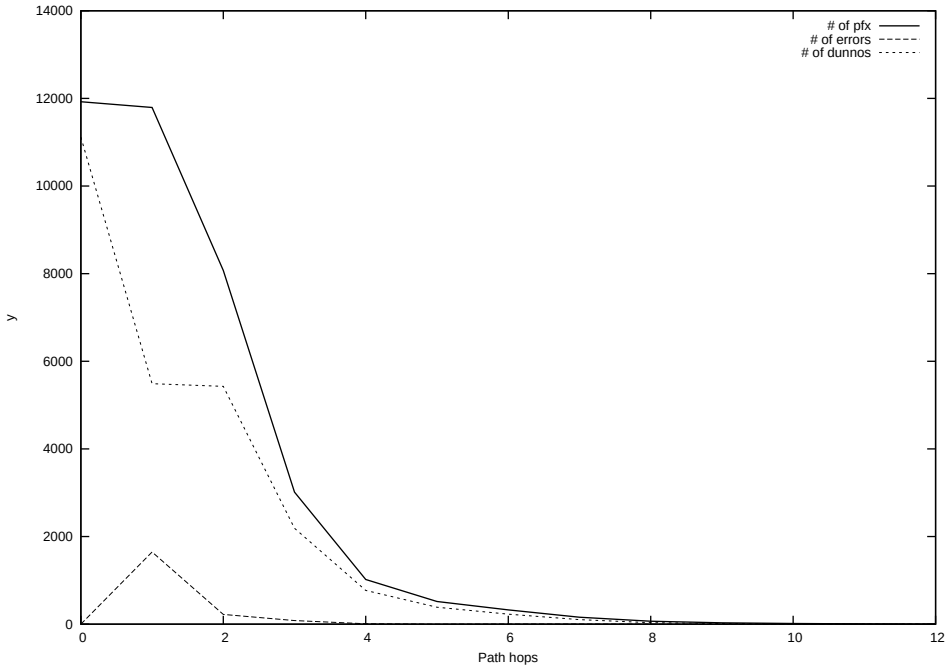
2013-03-06



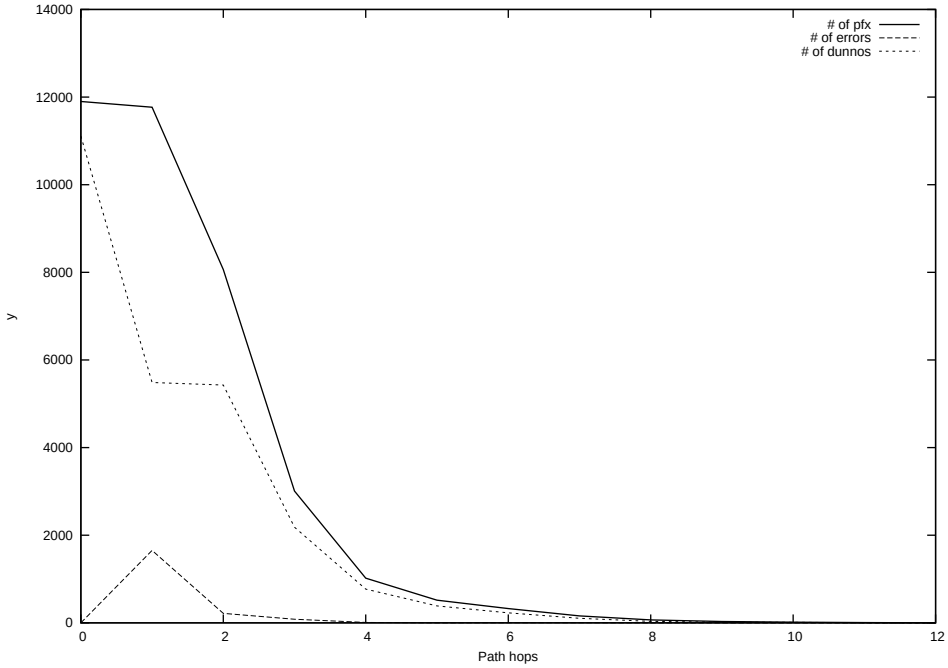
2013-03-07



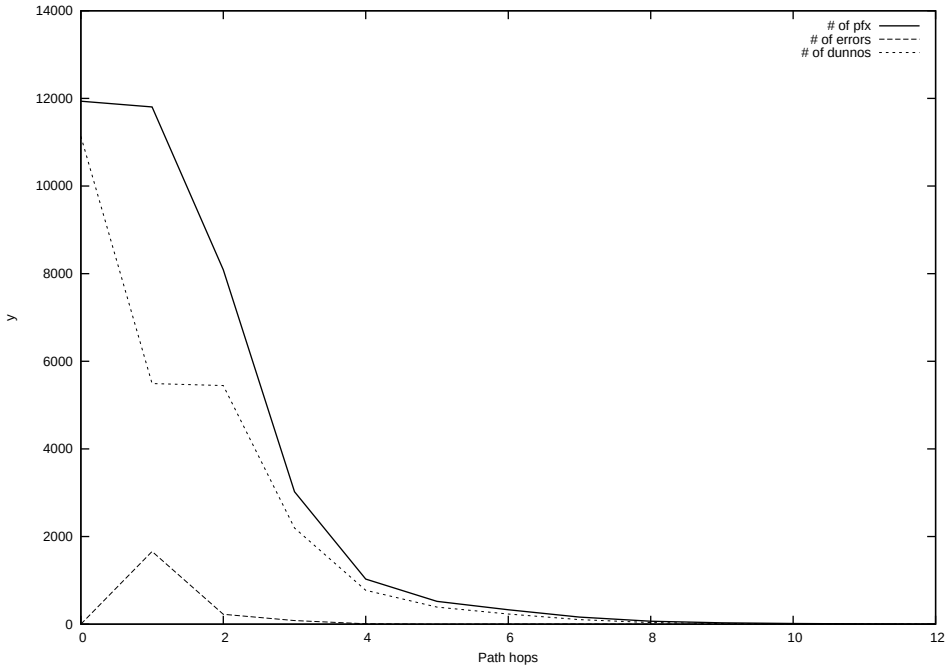
2013-03-08



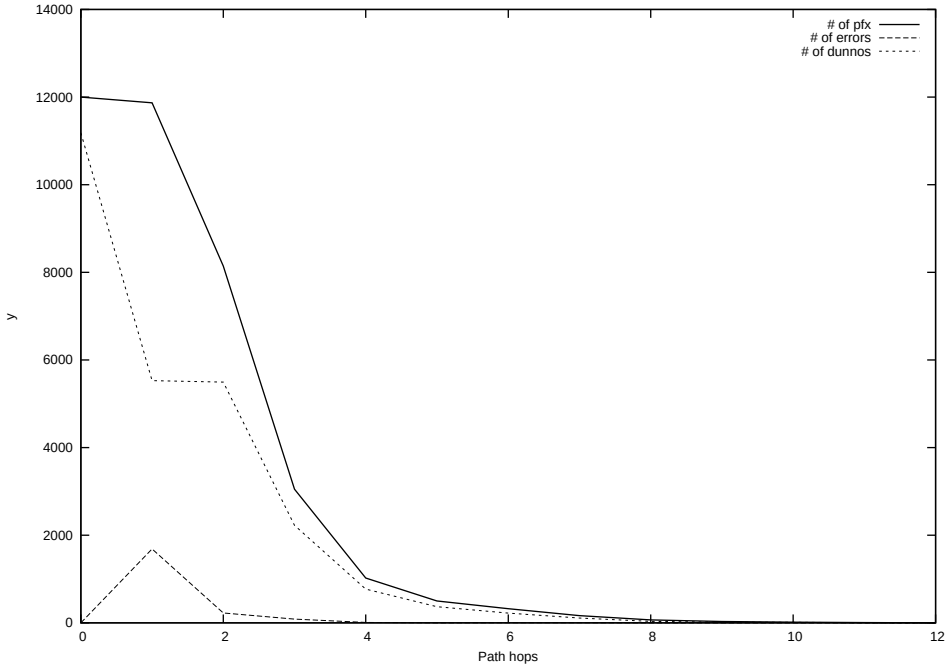
2013-03-09



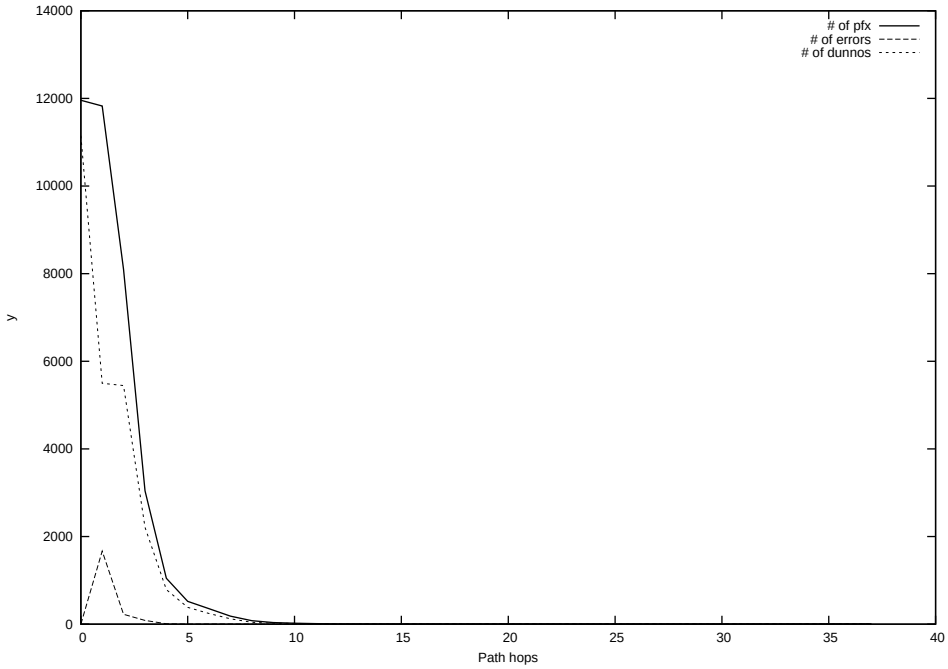
2013-03-10



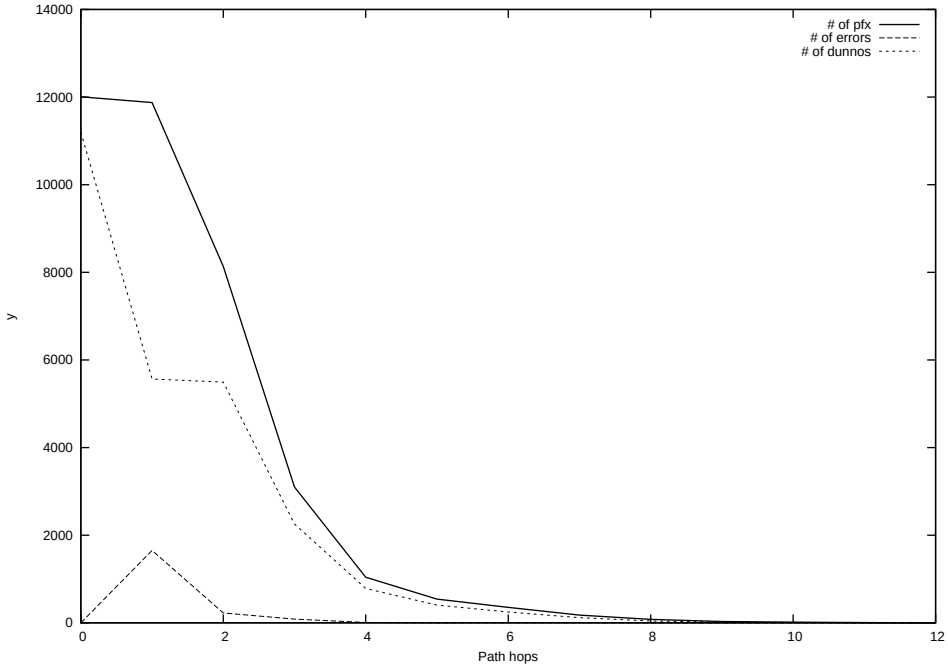
2013-03-11



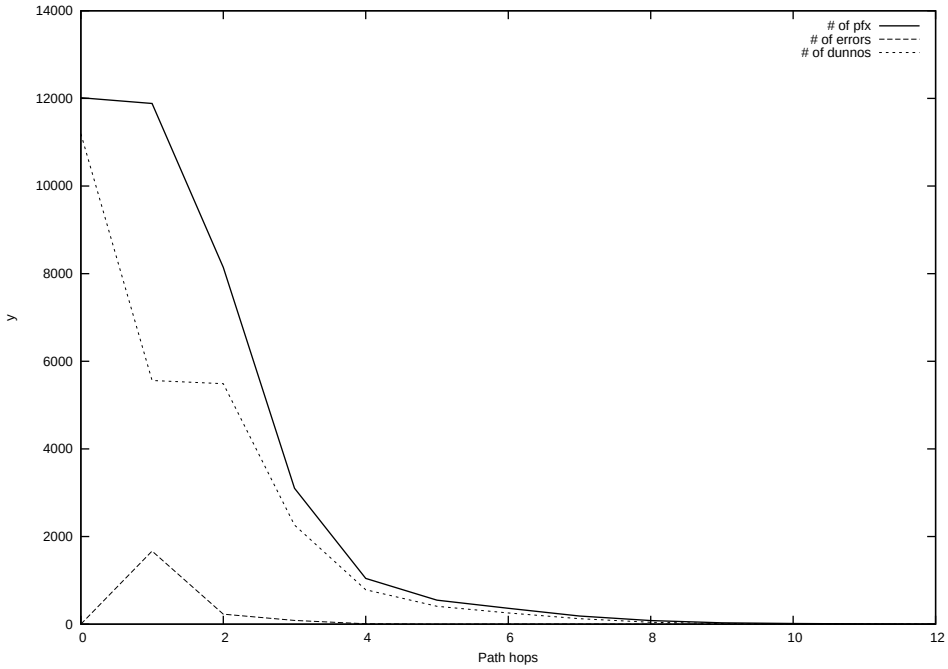
2013-03-12



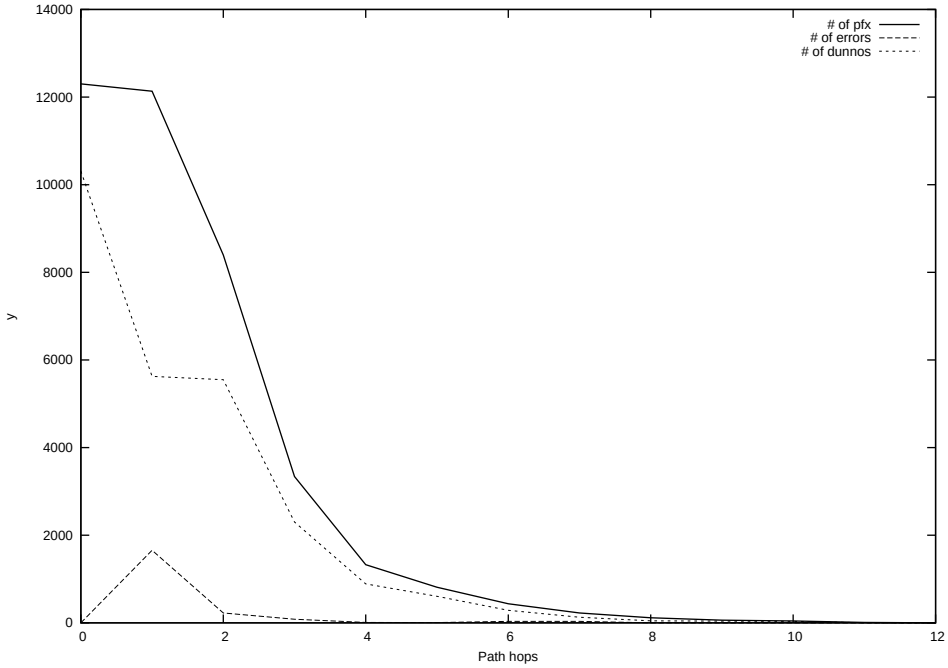
2013-03-13



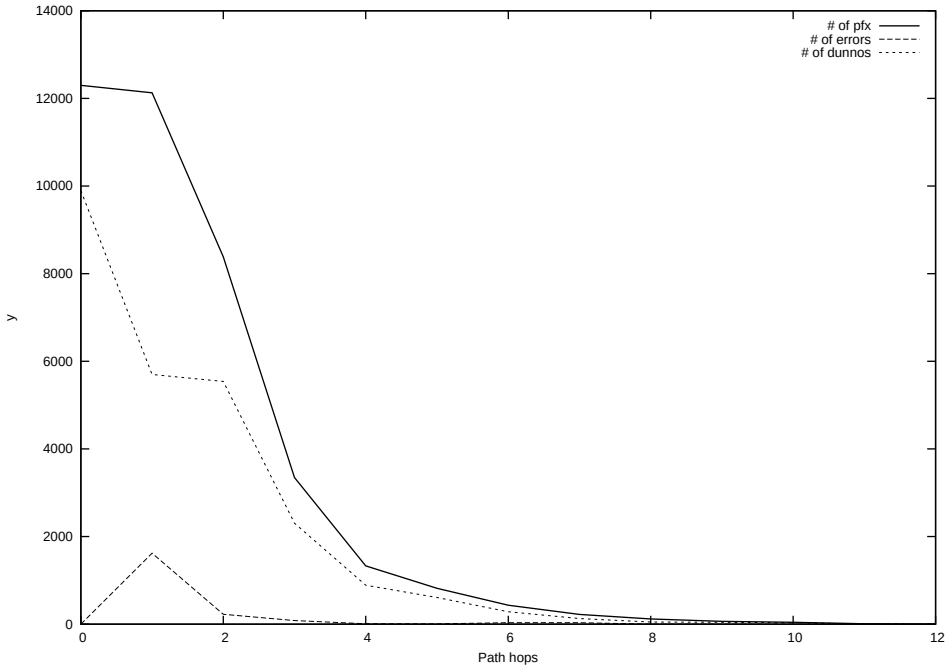
2013-03-14



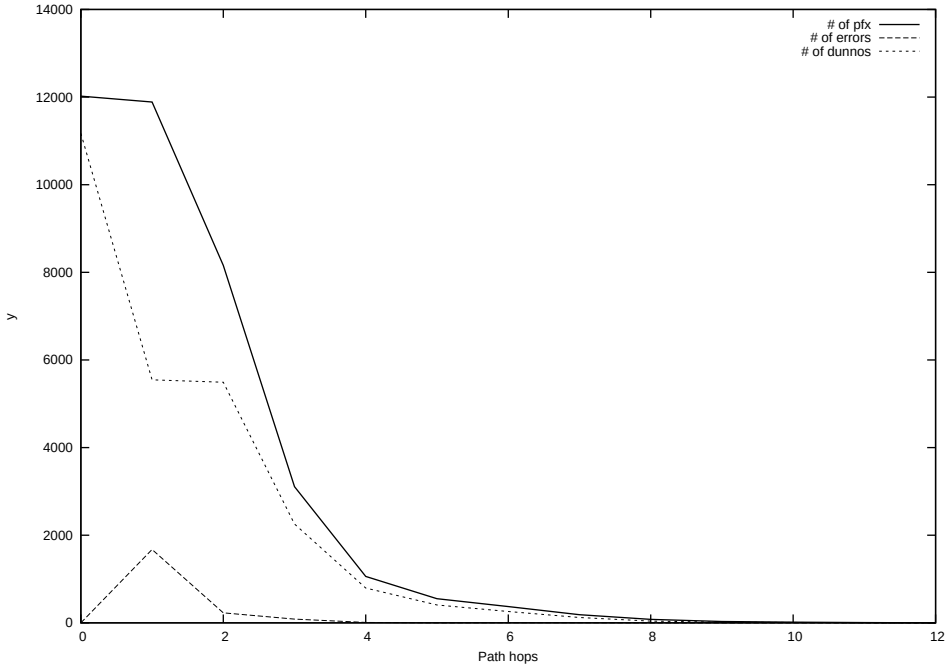
2013-03-15



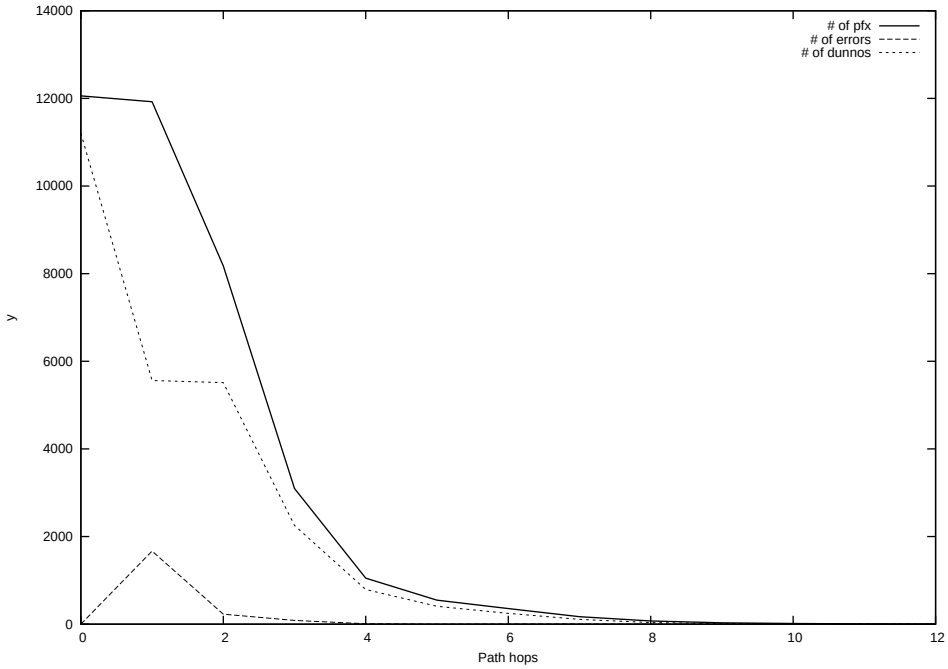
2013-03-16



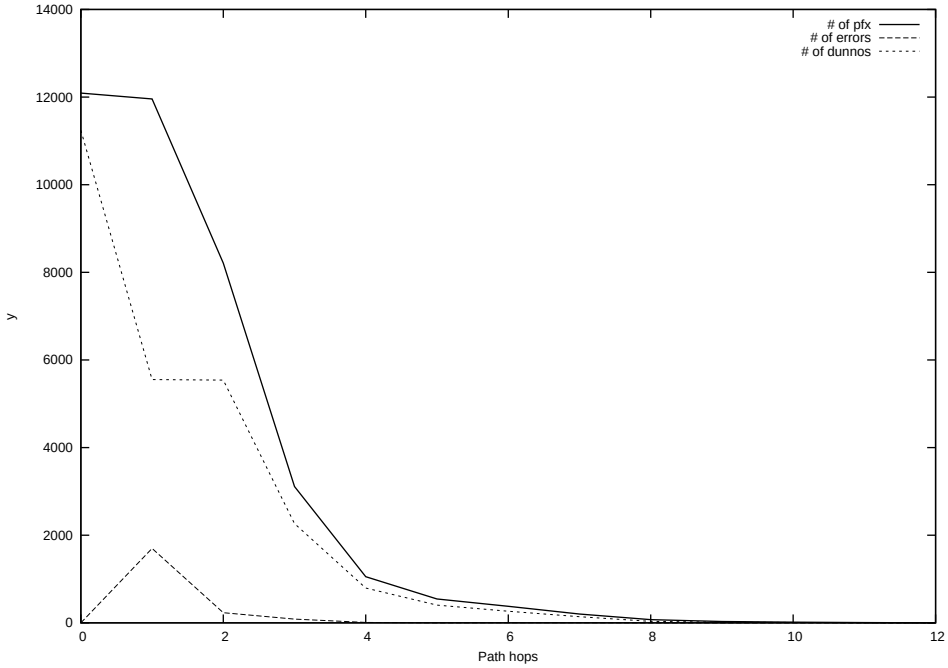
2013-03-17



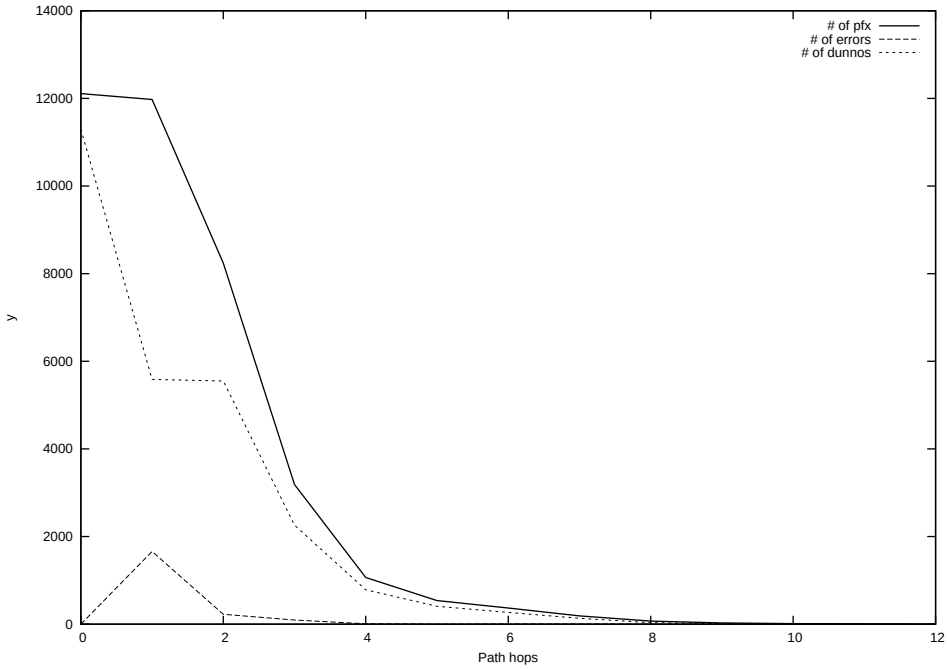
2013-03-18



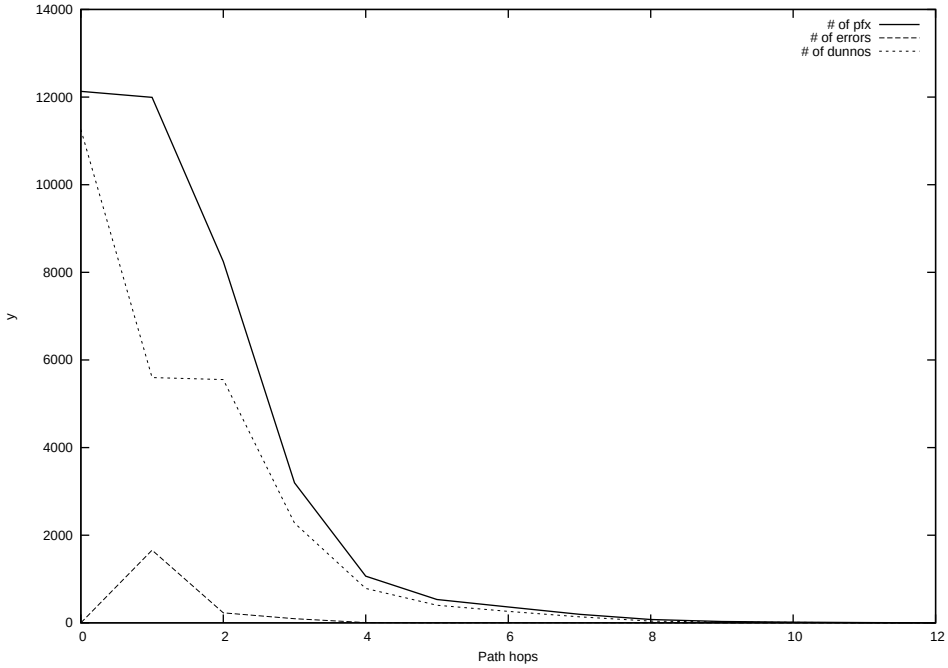
2013-03-19



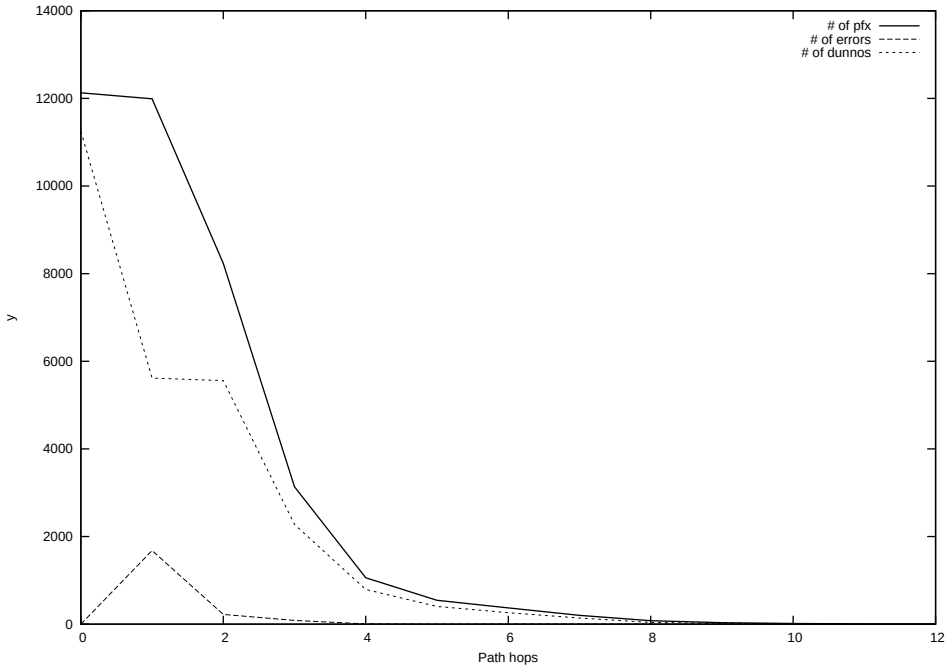
2013-03-20



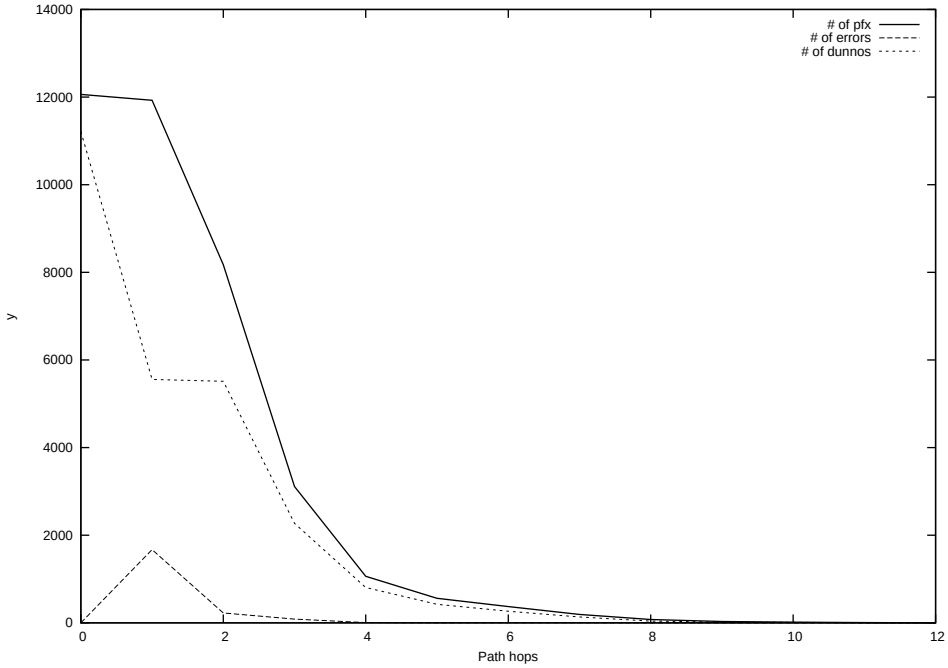
2013-03-21



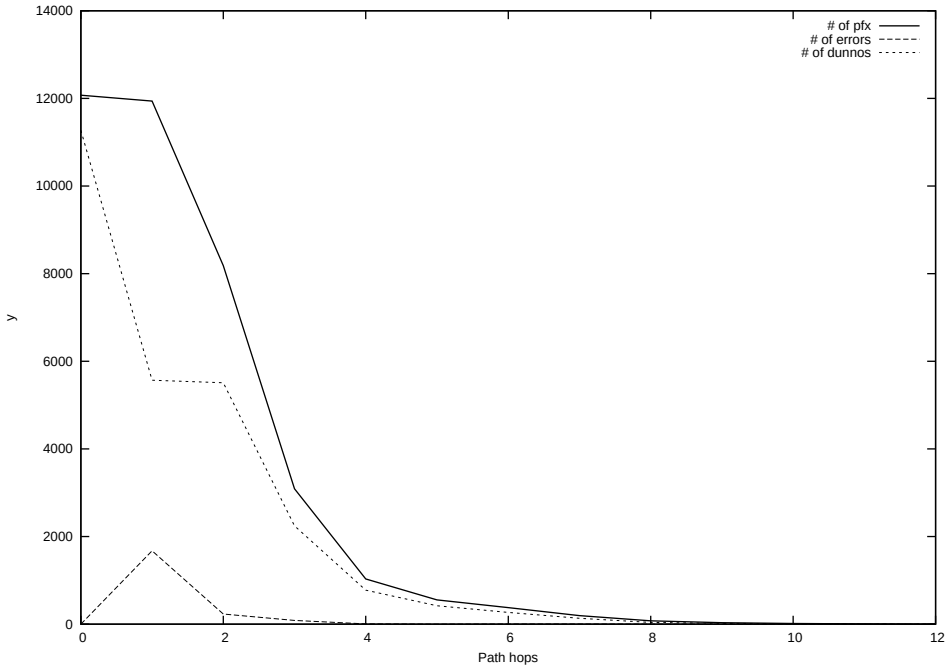
2013-03-22



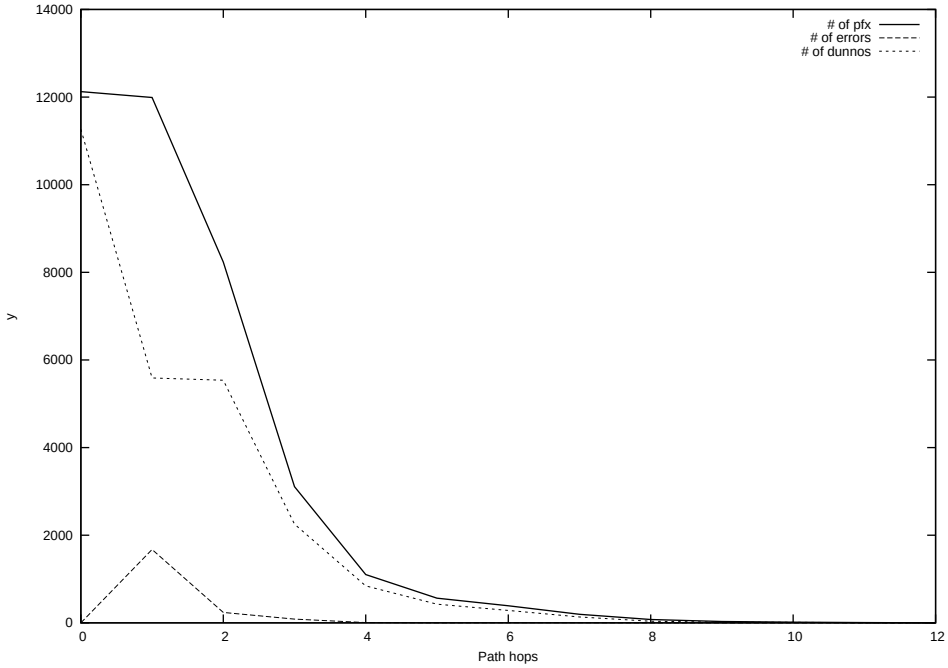
2013-03-23



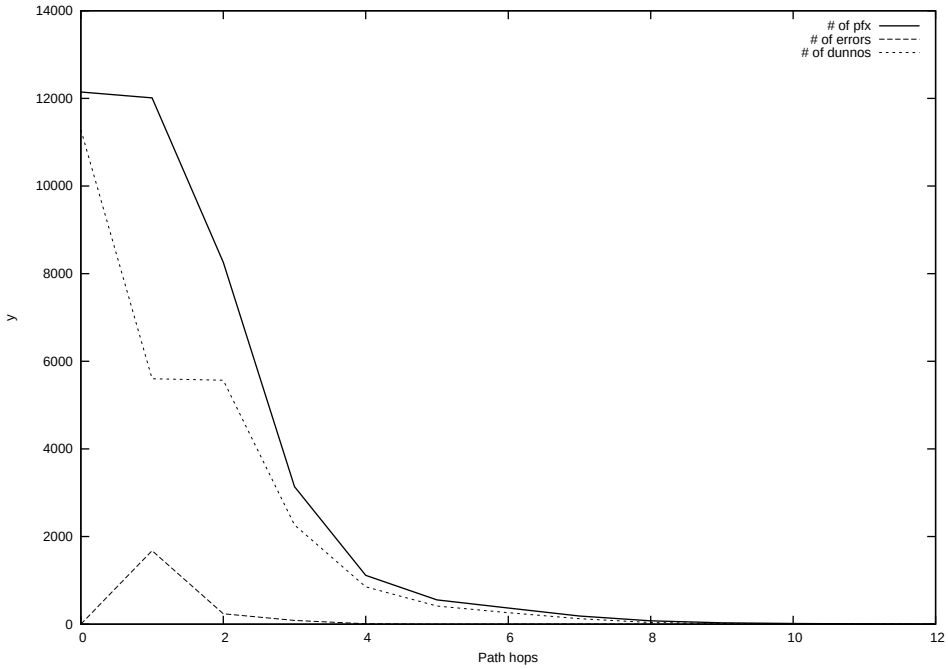
2013-03-24



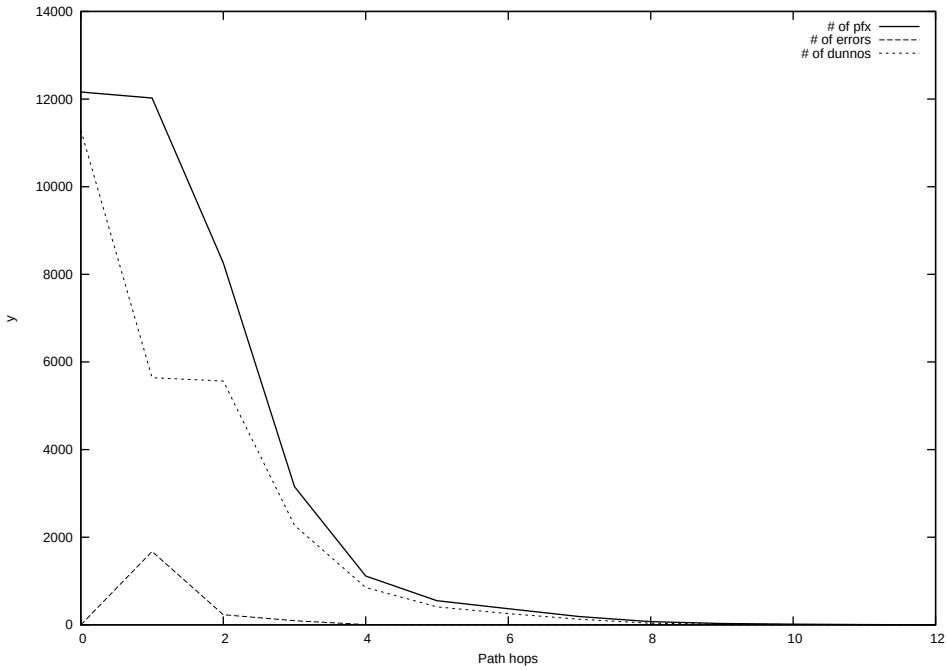
2013-03-25



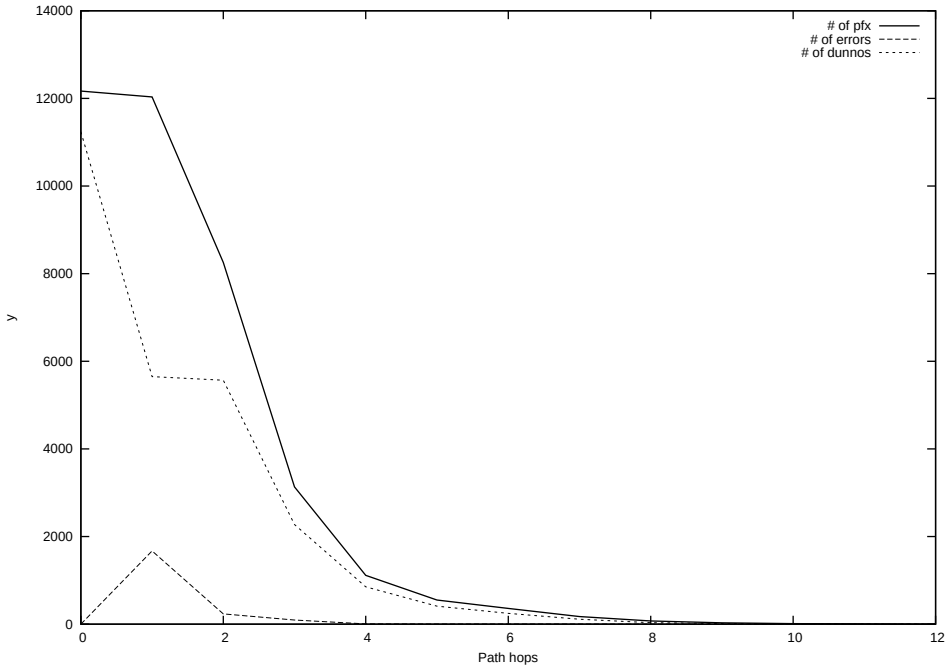
2013-03-26



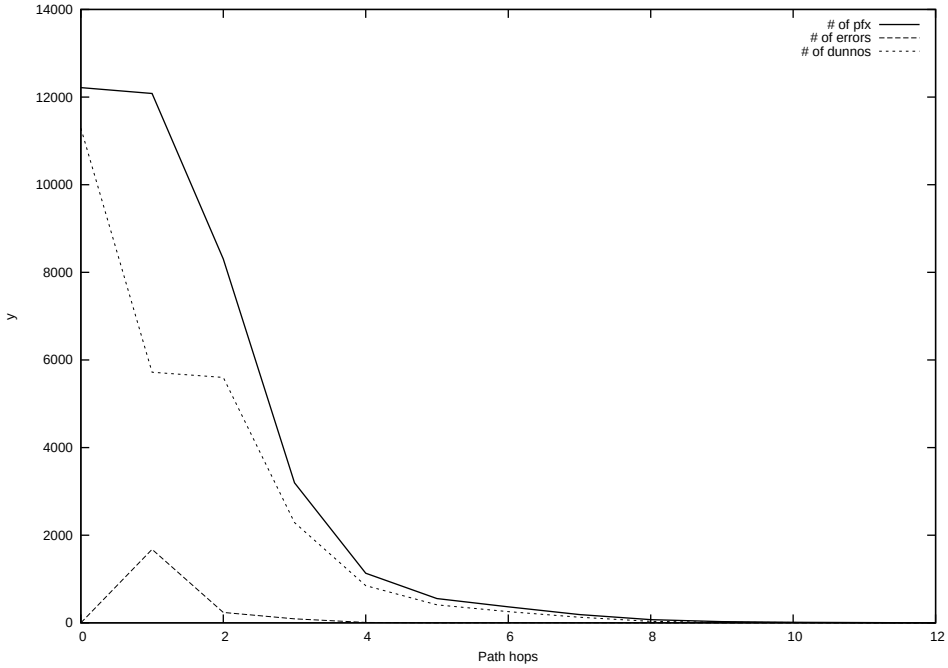
2013-03-27



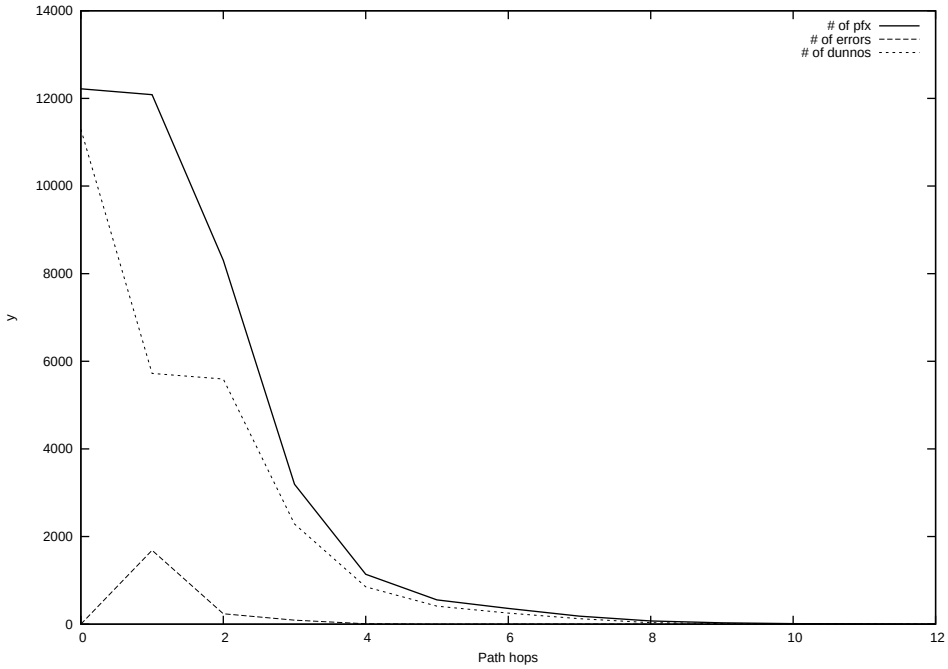
2013-03-28



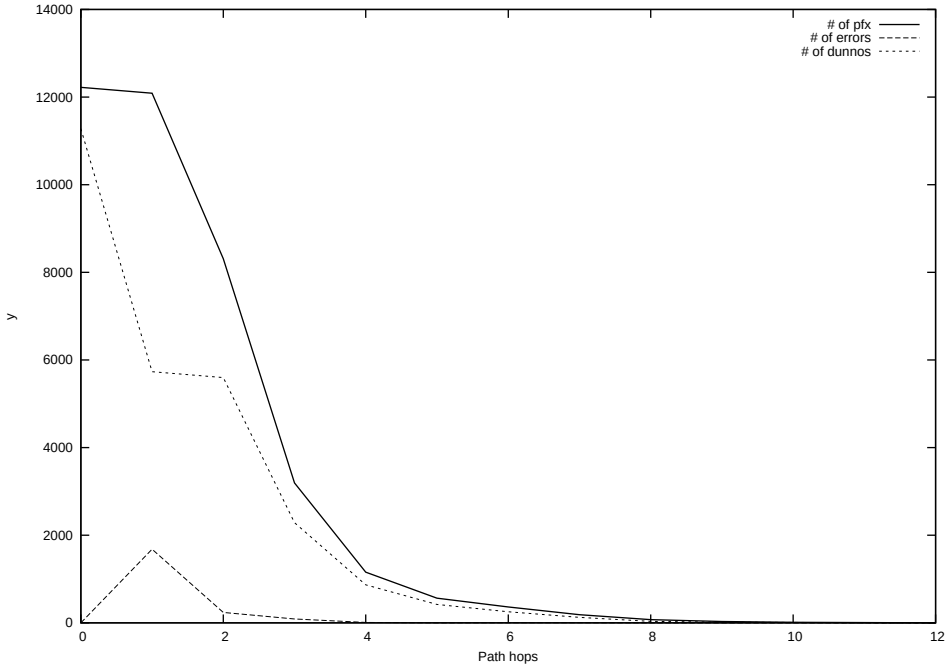
2013-03-29



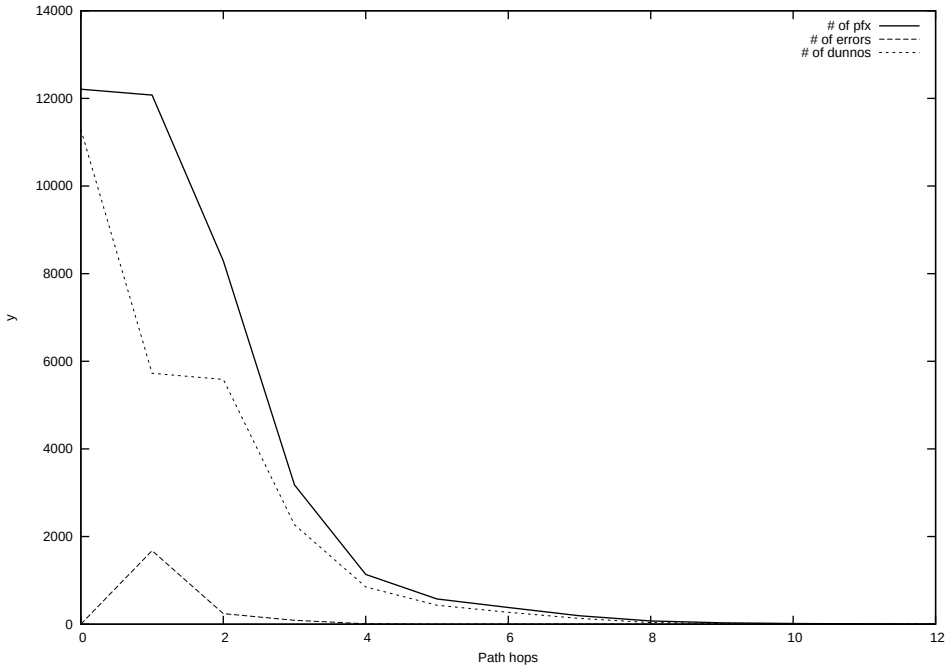
2013-03-30



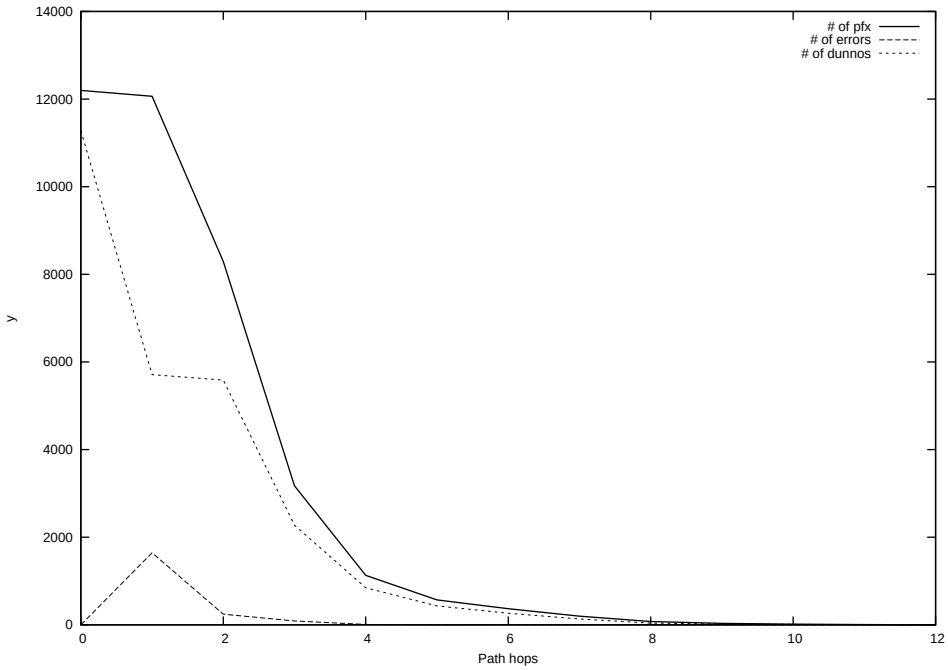
2013-03-31



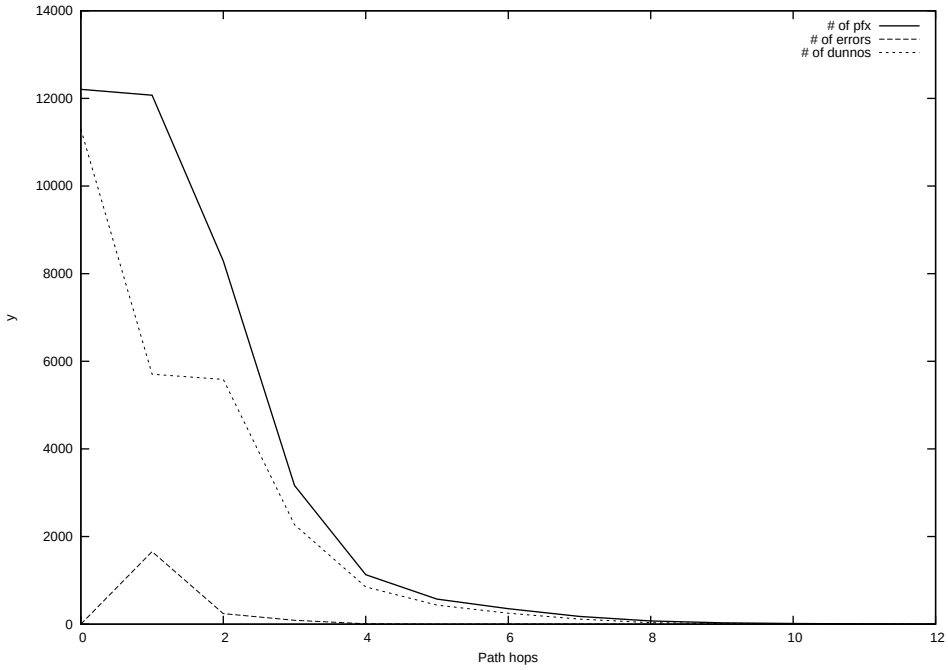
2013-04-01



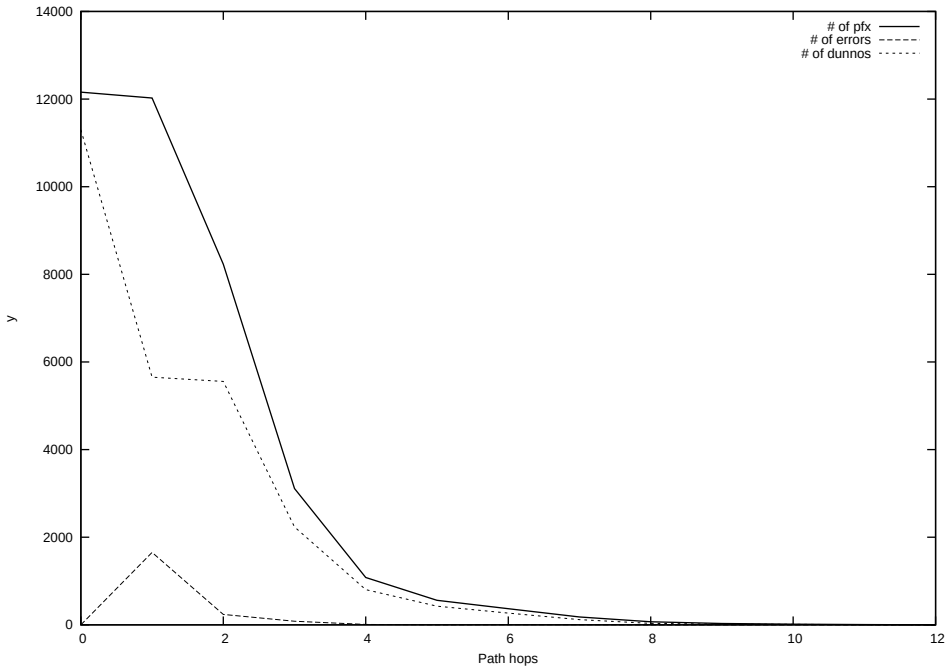
2013-04-02



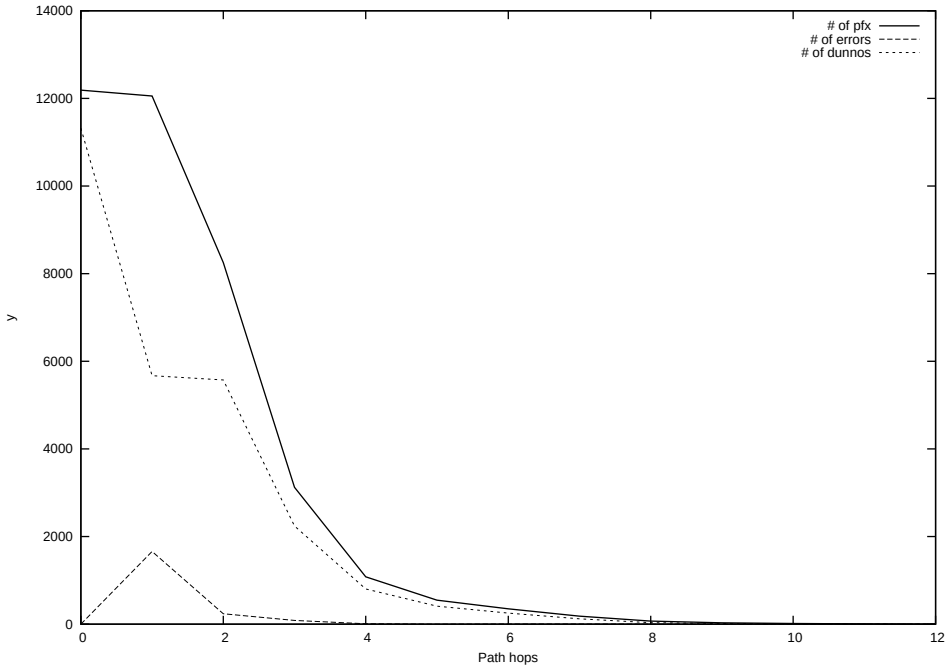
2013-04-03



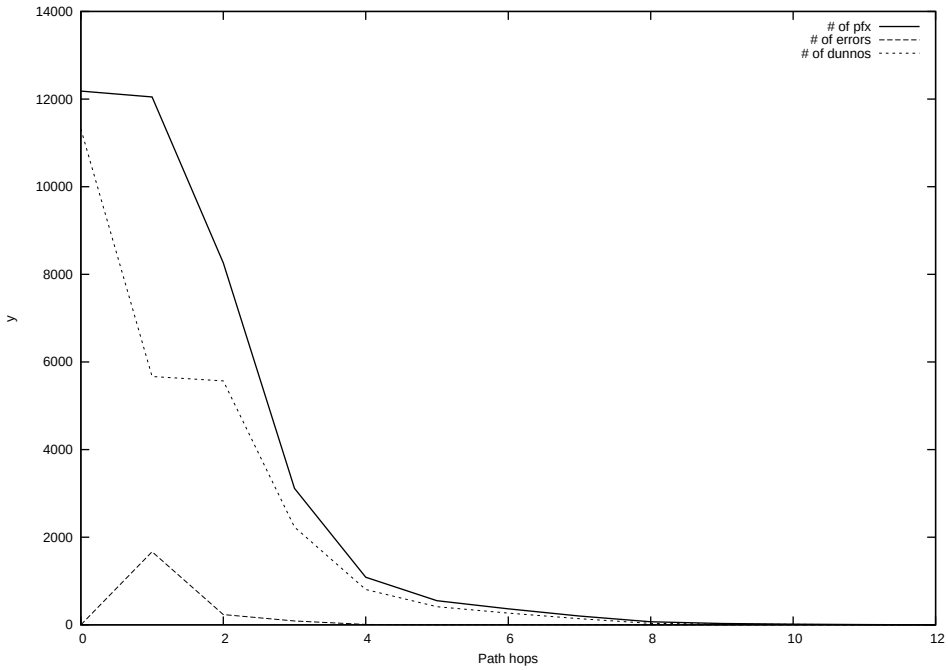
2013-04-04



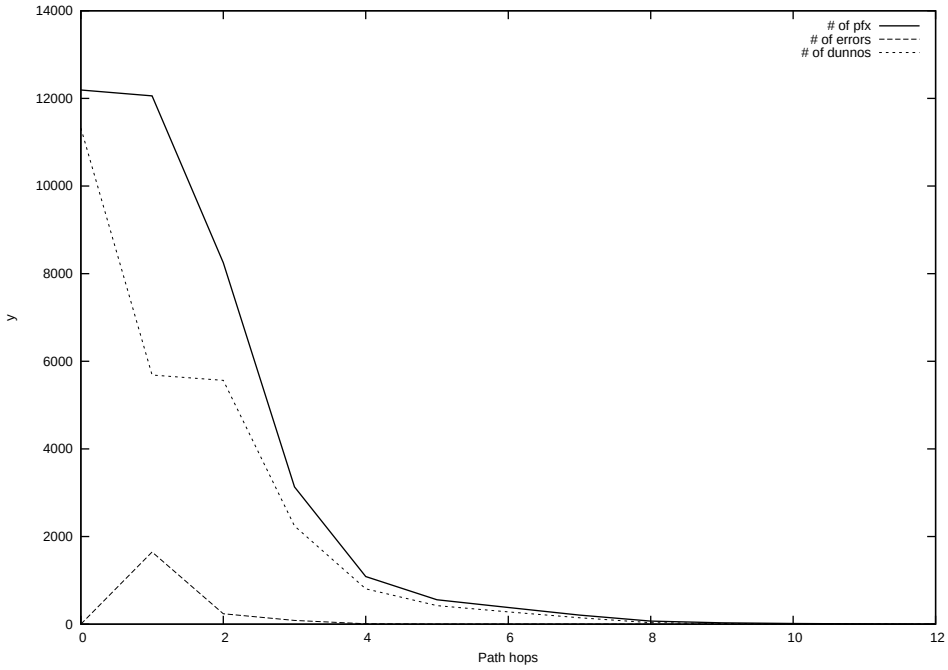
2013-04-05



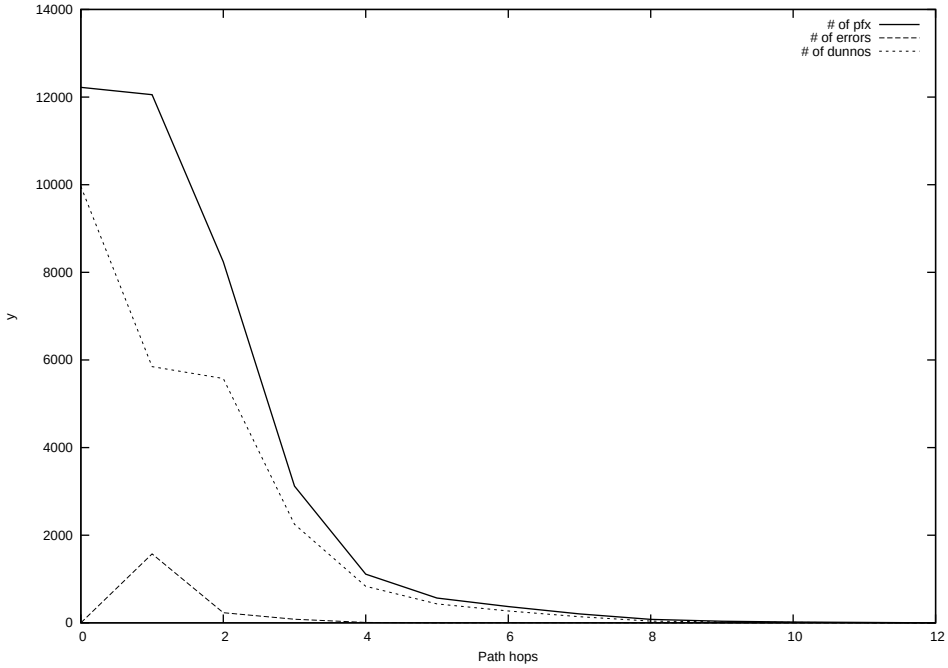
2013-04-06



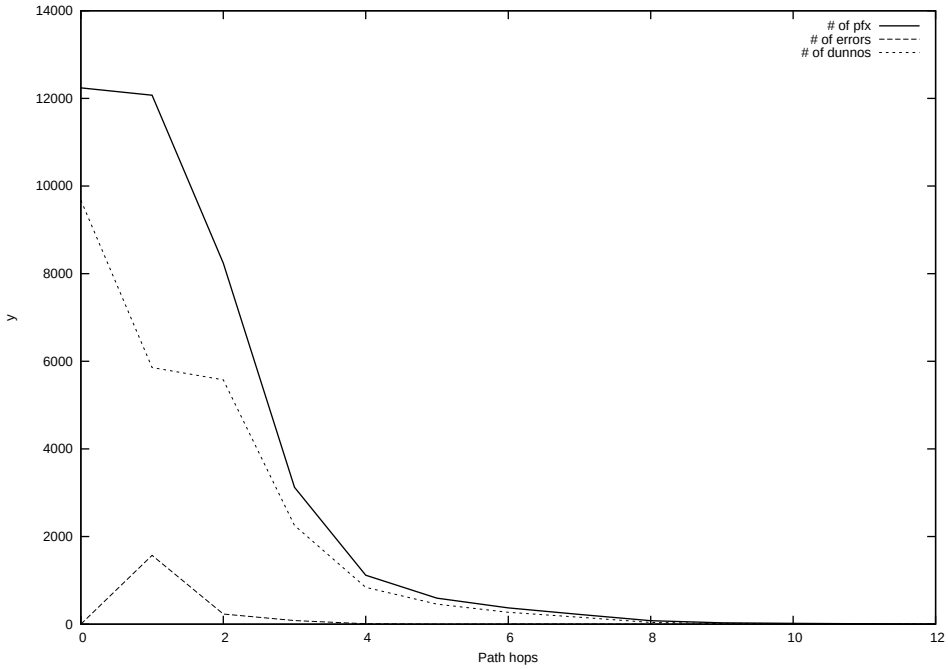
2013-04-07



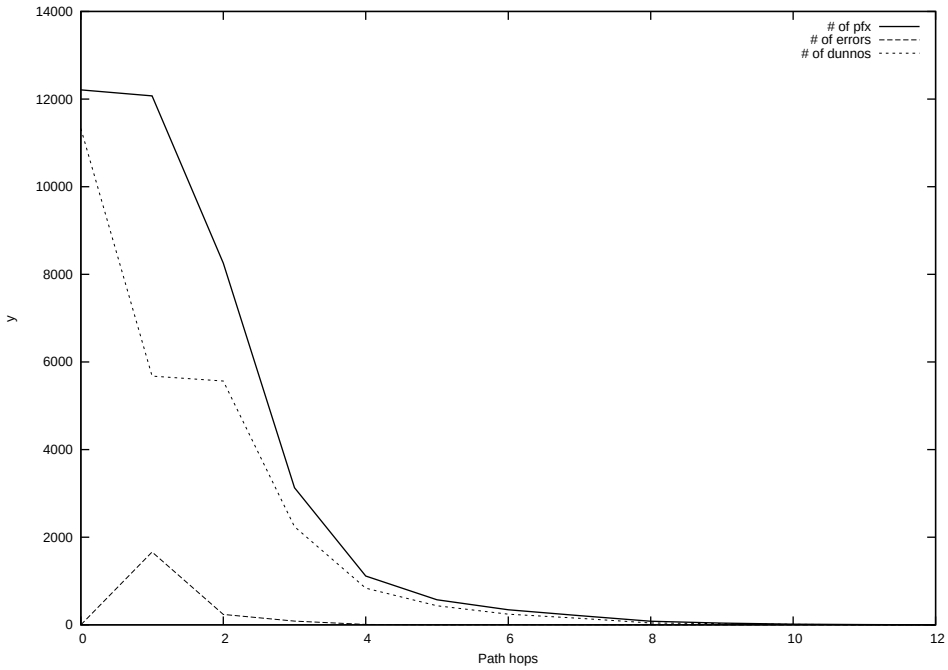
2013-04-08



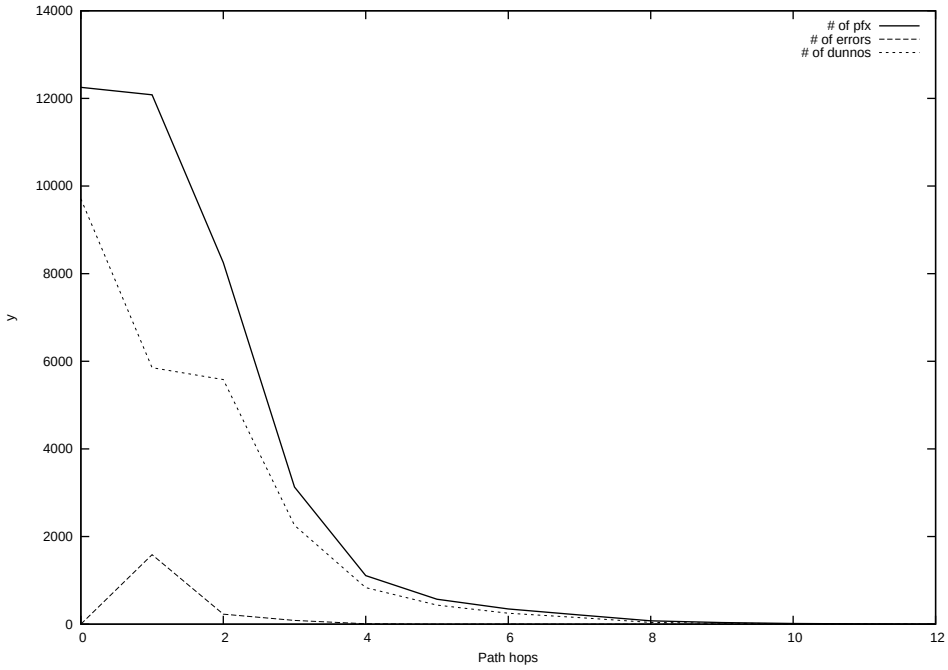
2013-04-09



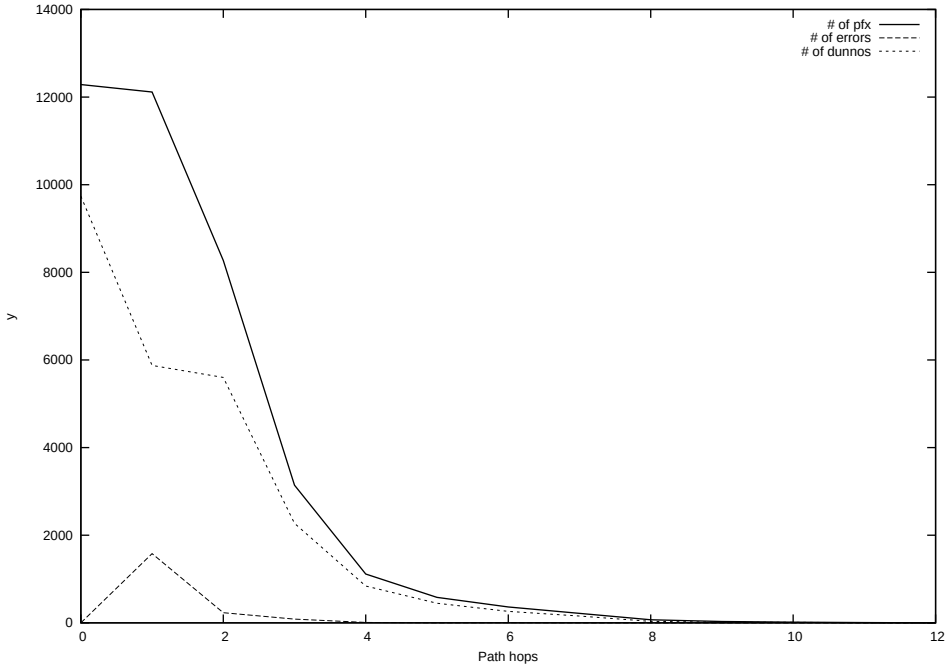
2013-04-10



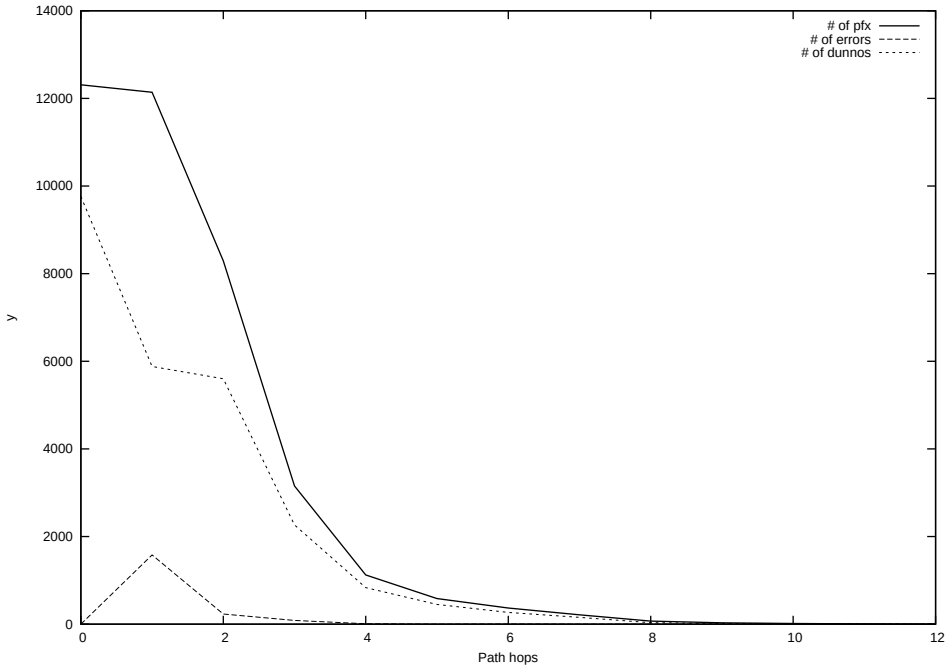
2013-04-11



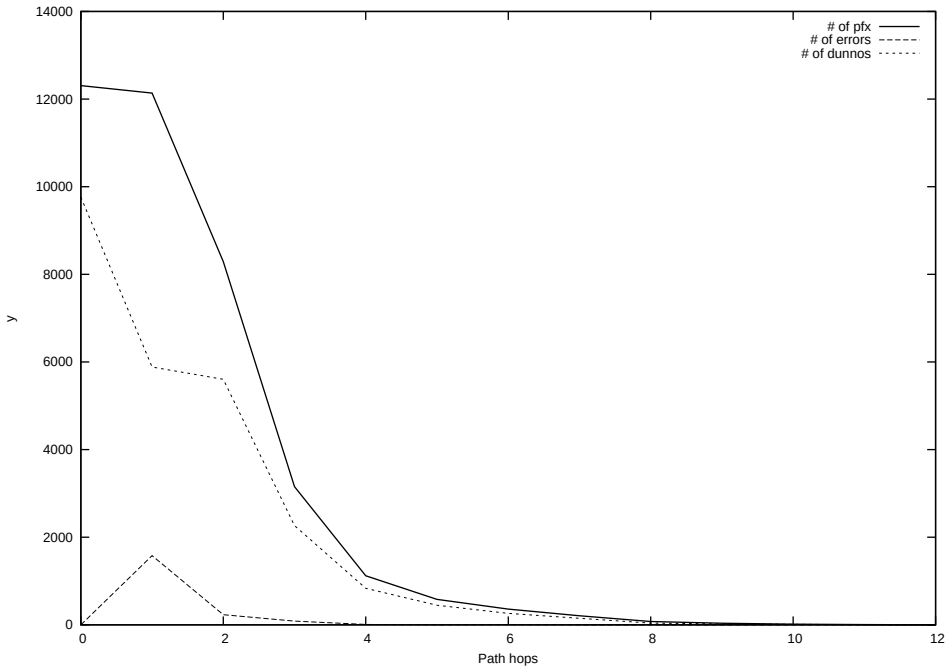
2013-04-12



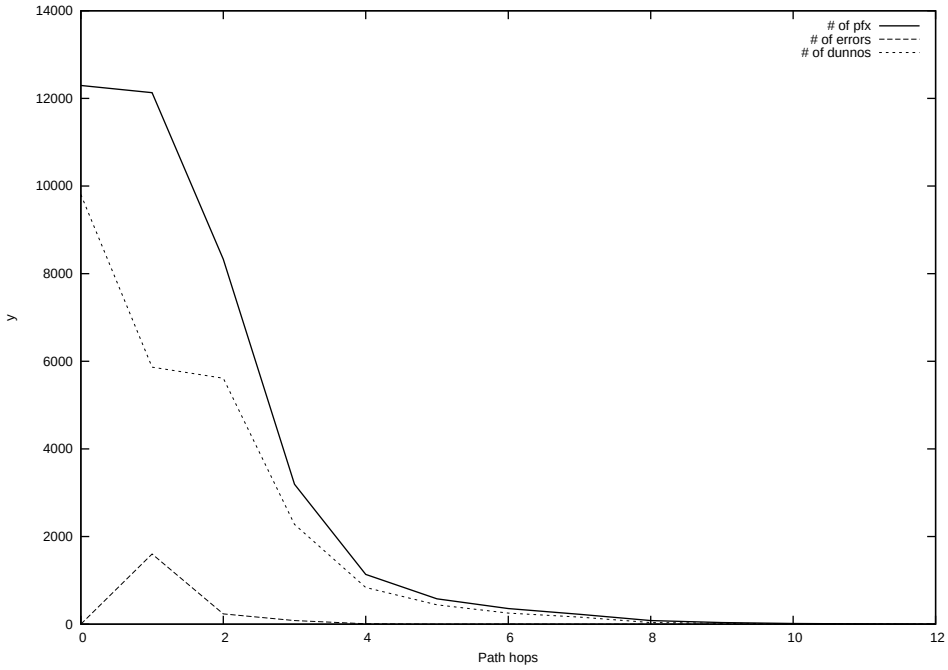
2013-04-13



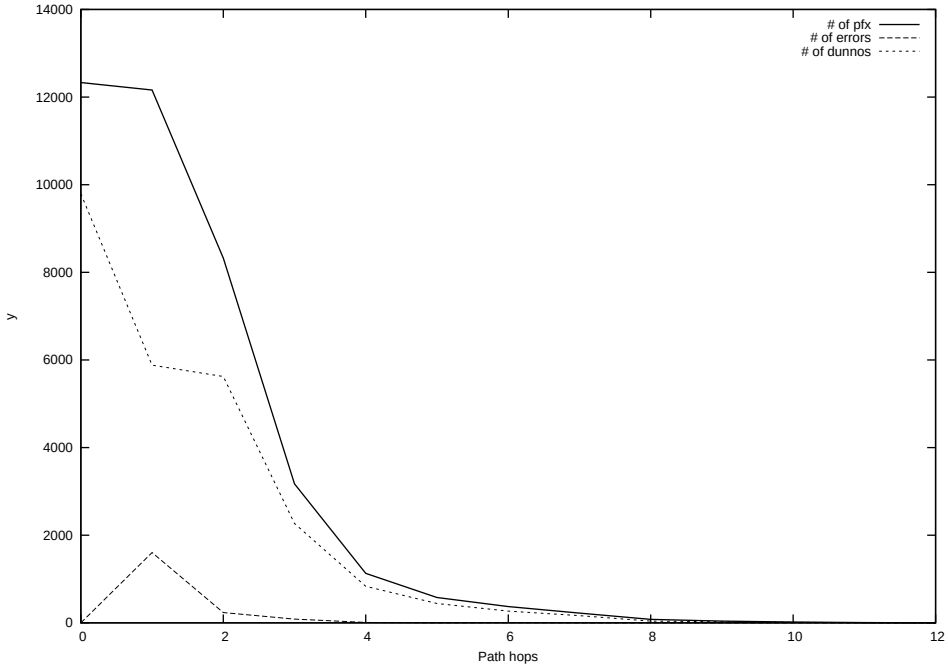
2013-04-14



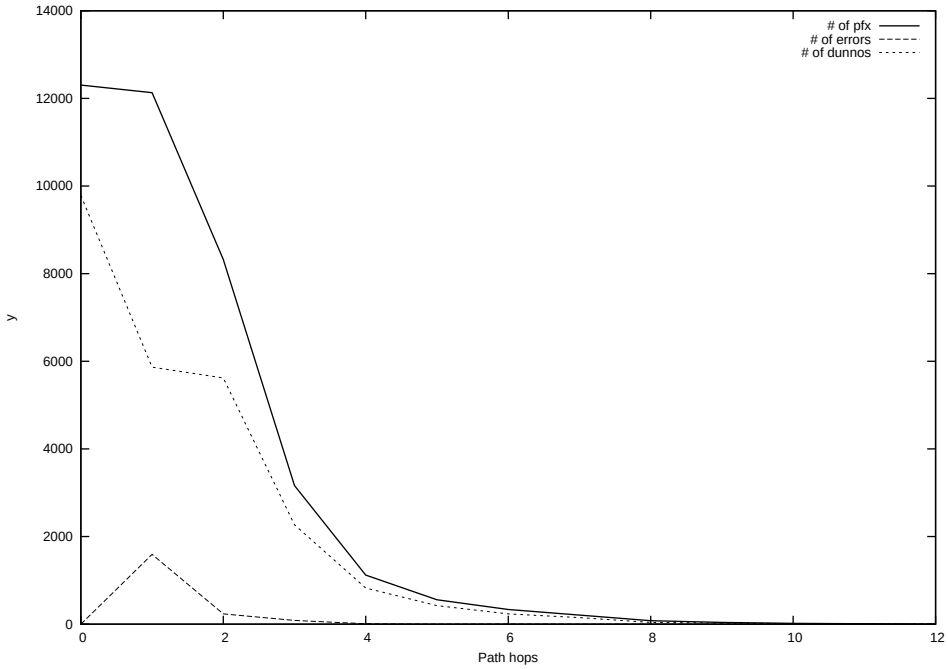
2013-04-15



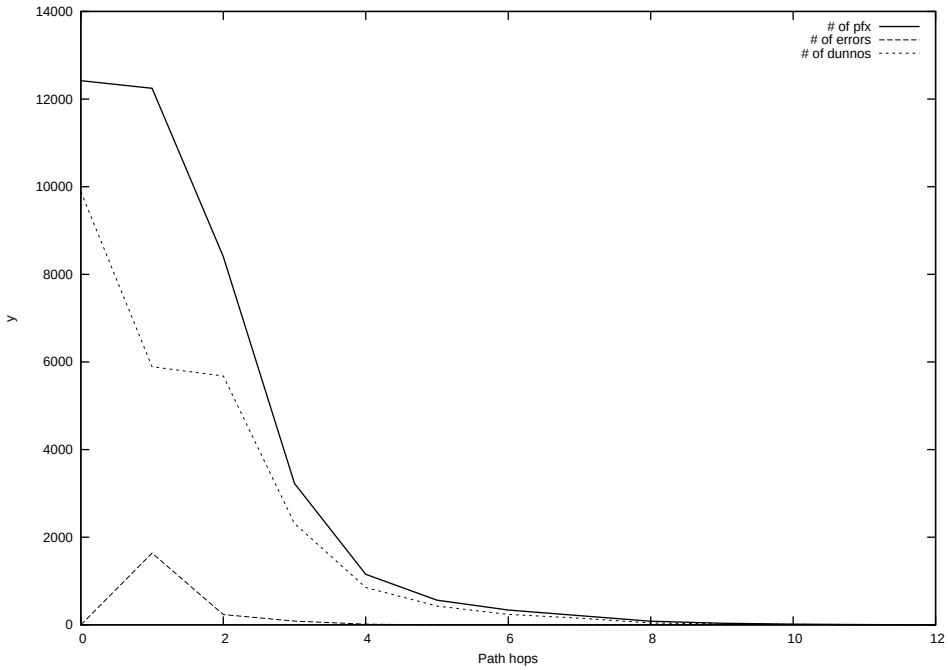
2013-04-16



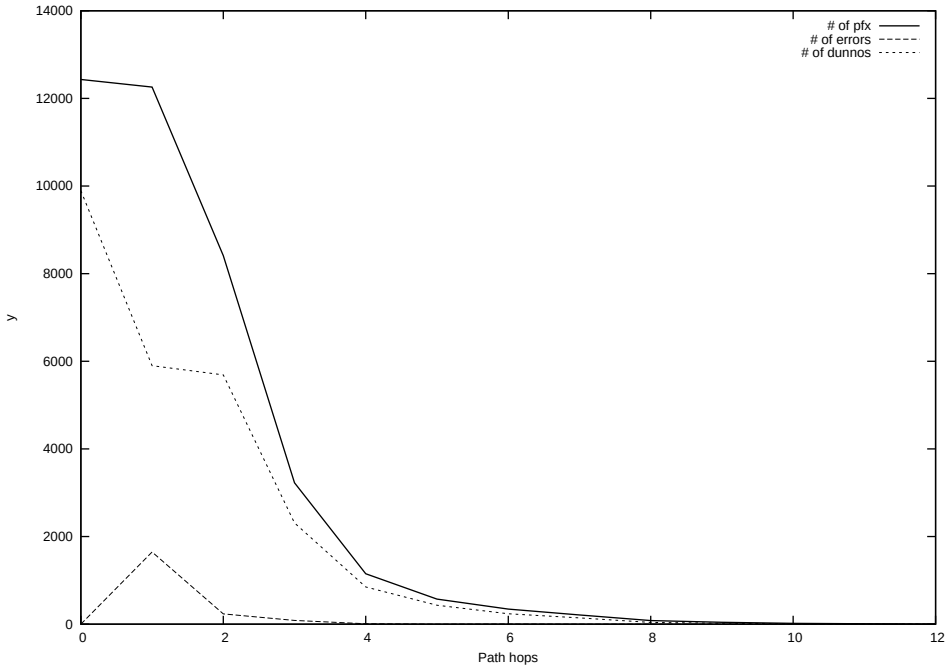
2013-04-17



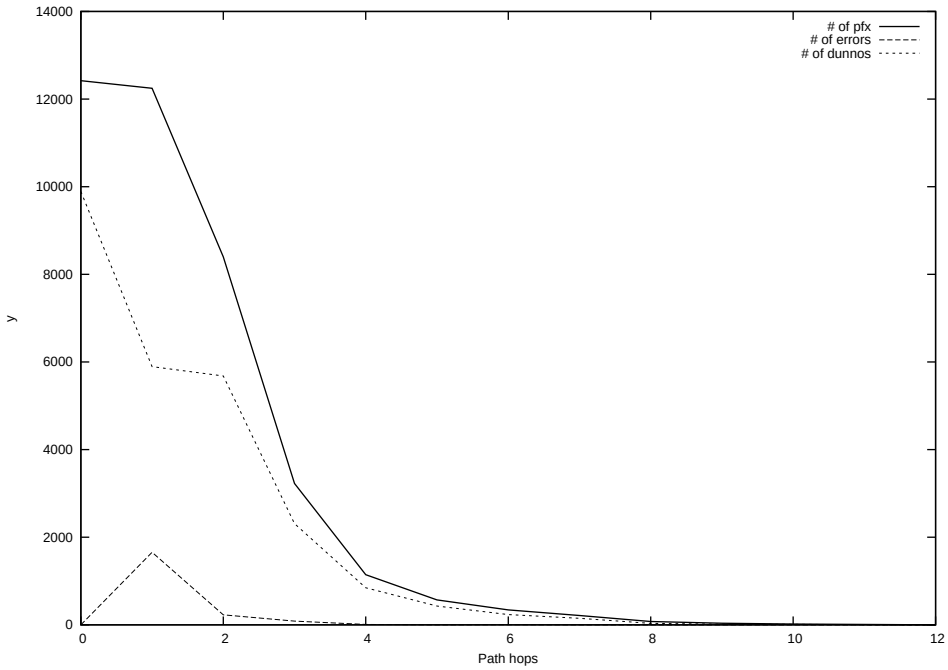
2013-04-18



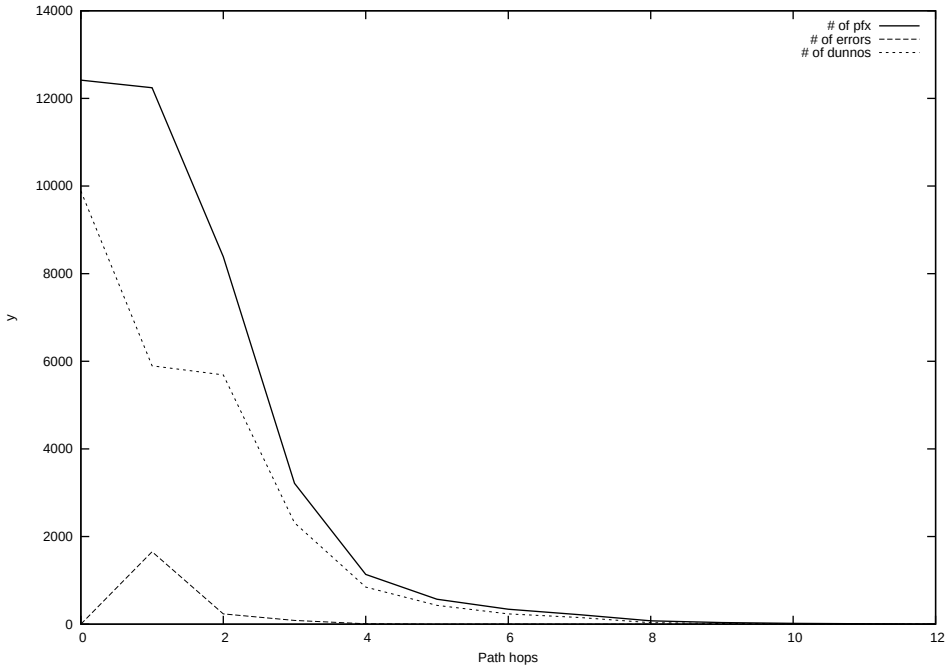
2013-04-19



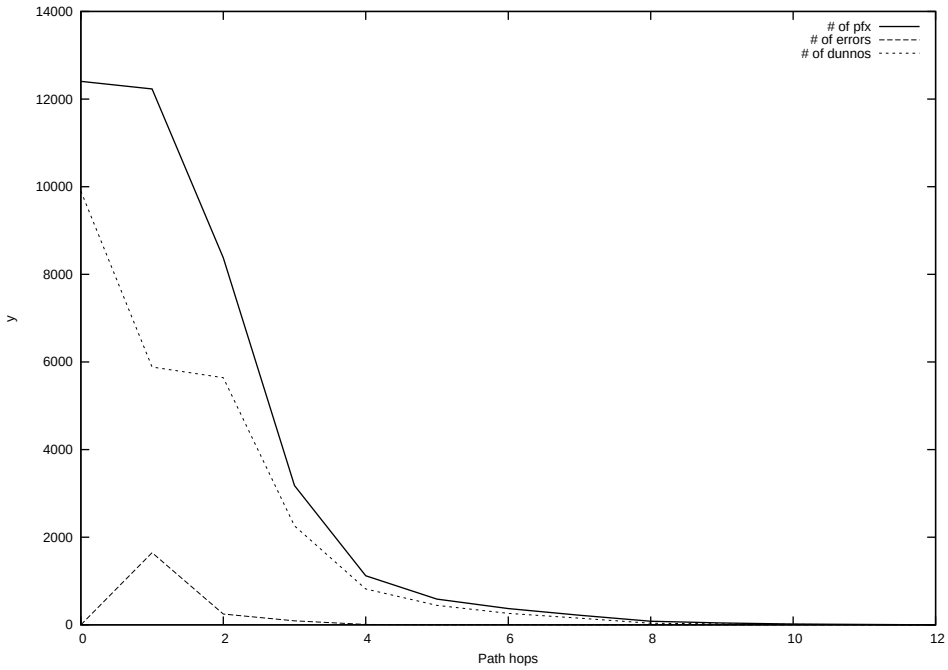
2013-04-20



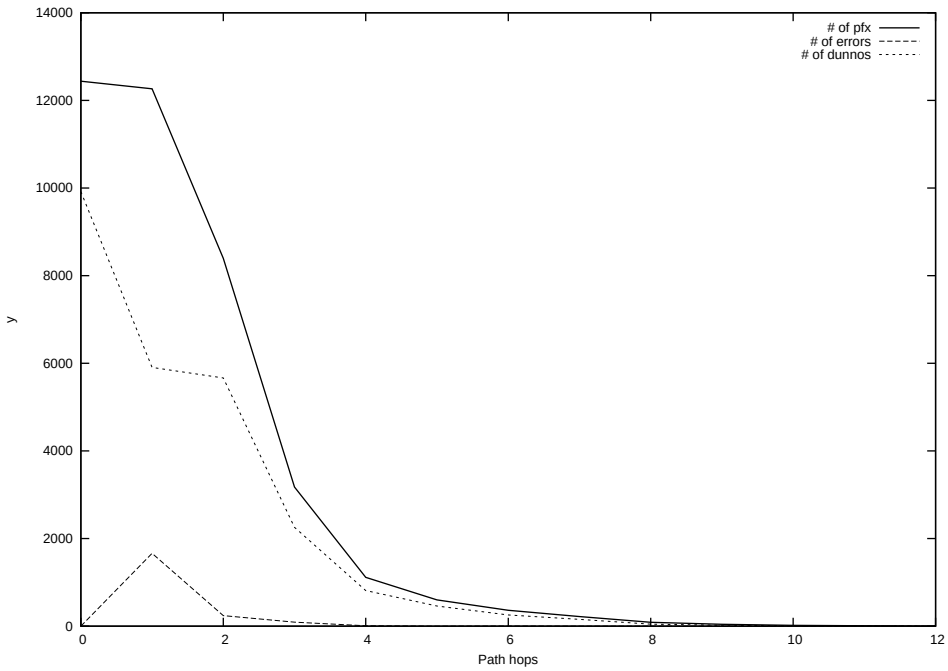
2013-04-21



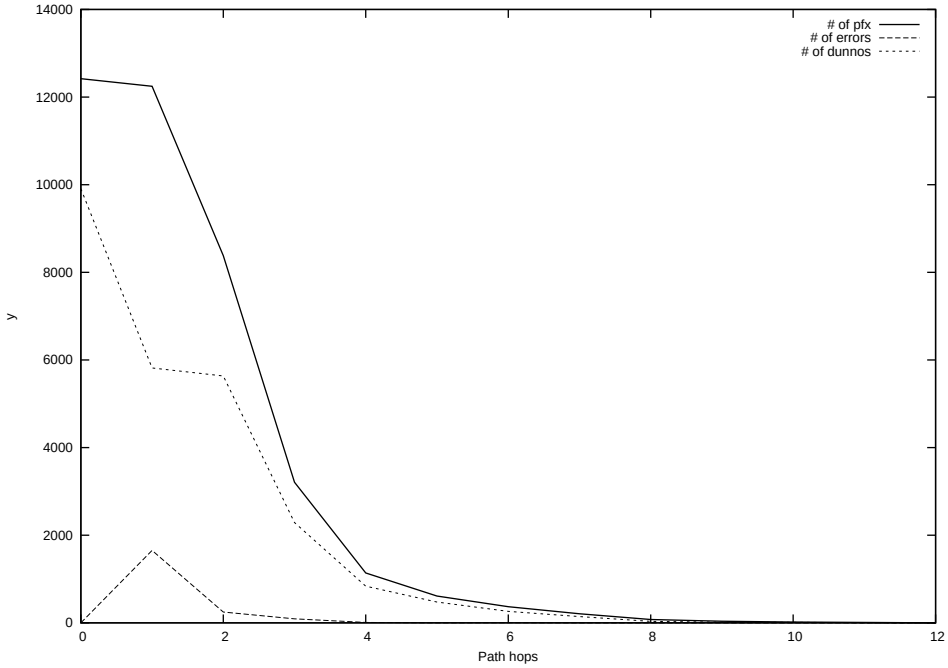
2013-04-22



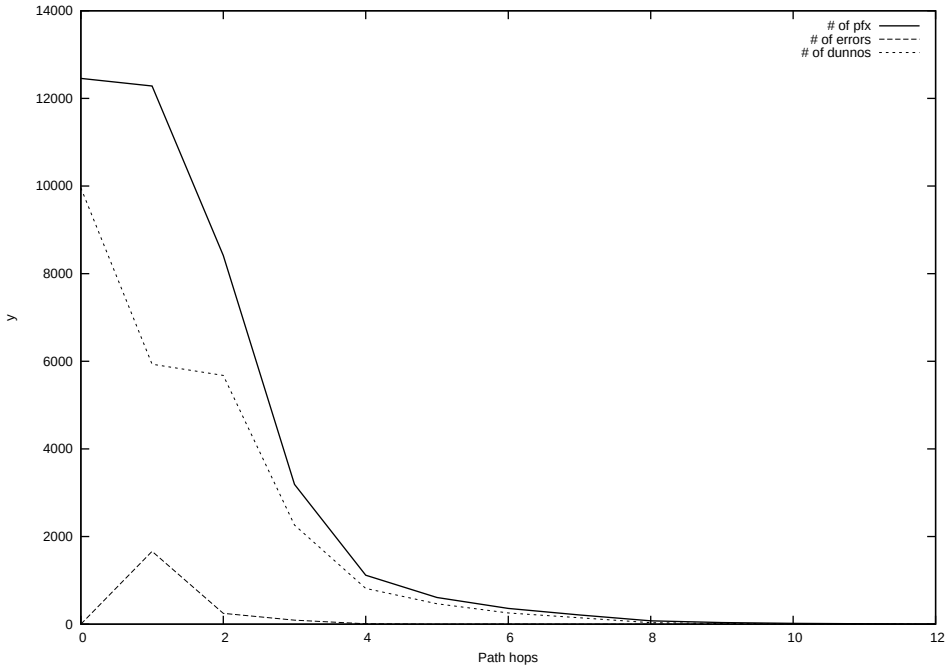
2013-04-23



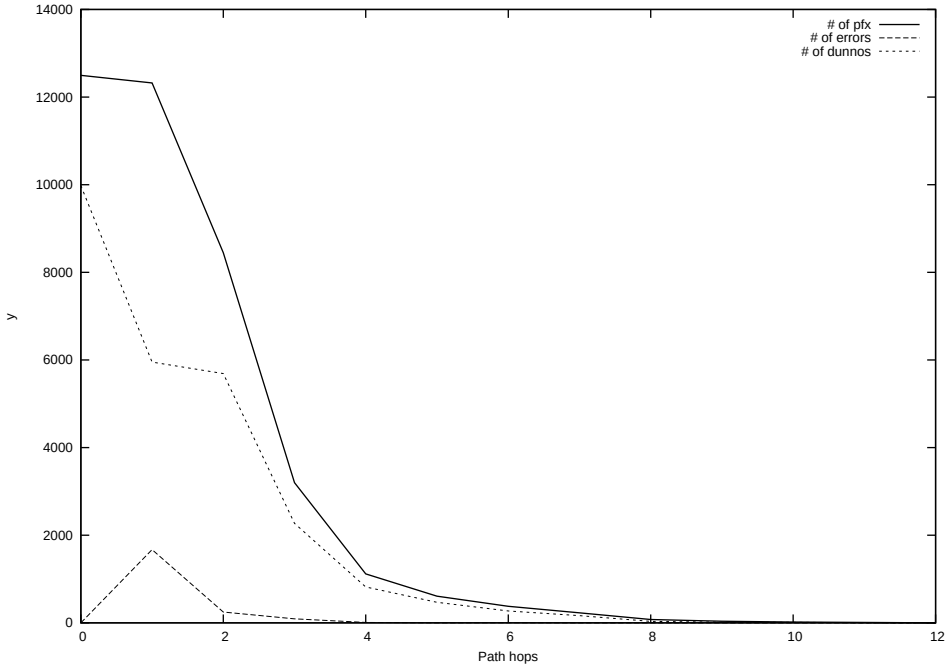
2013-04-24



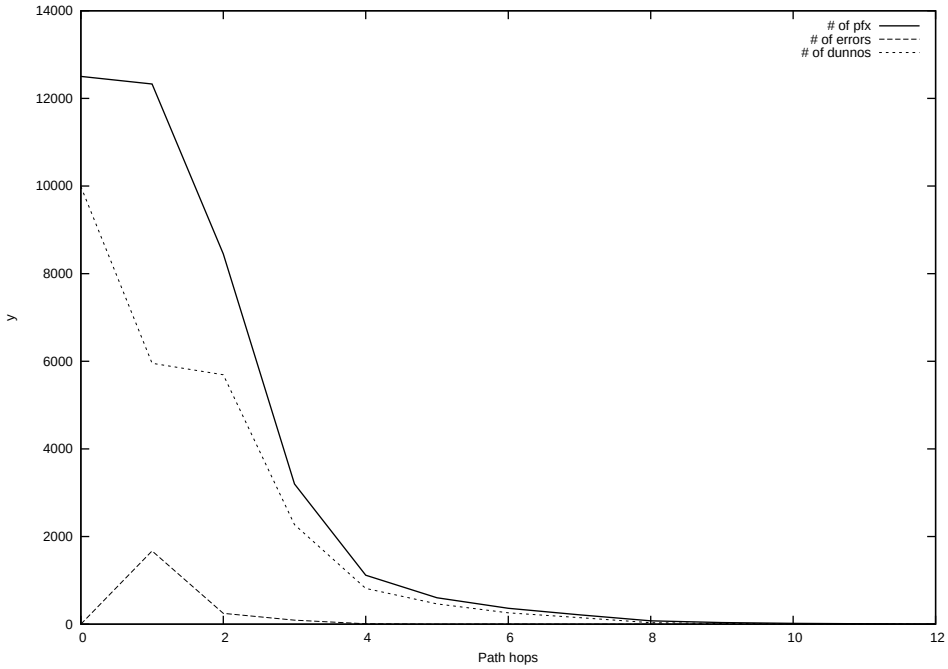
2013-04-25



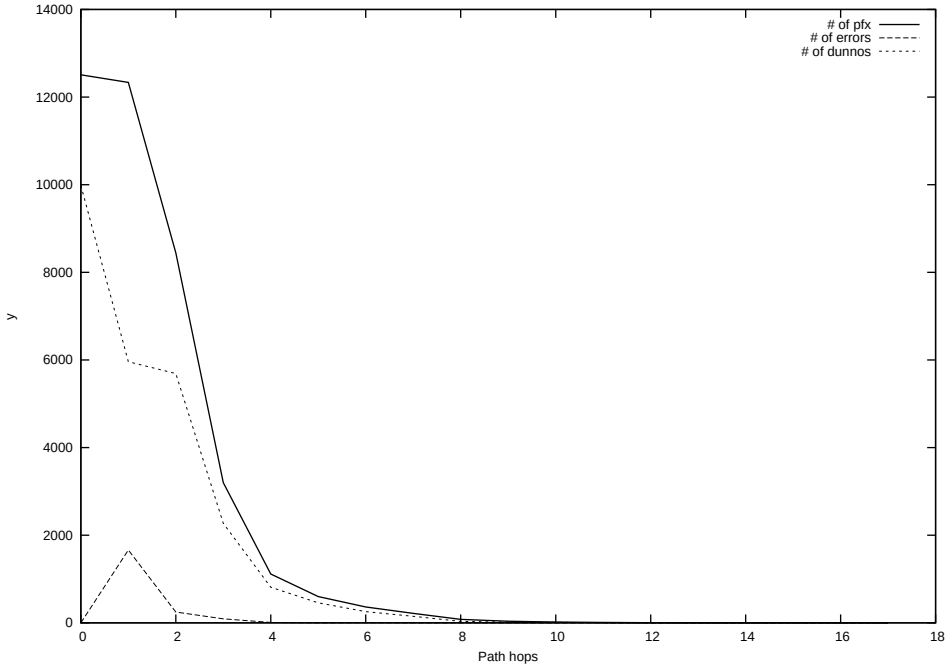
2013-04-26



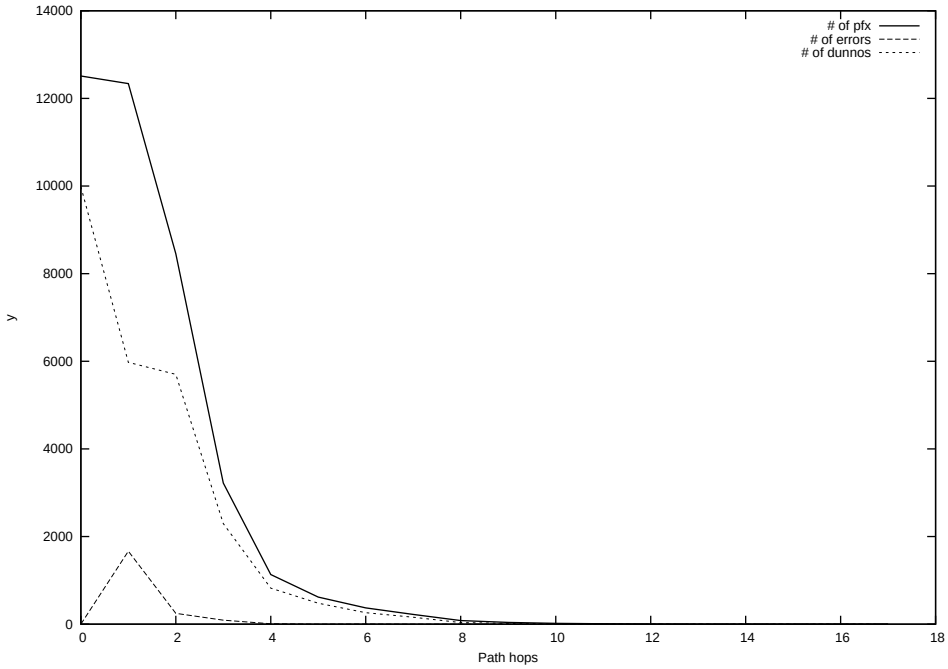
2013-04-27



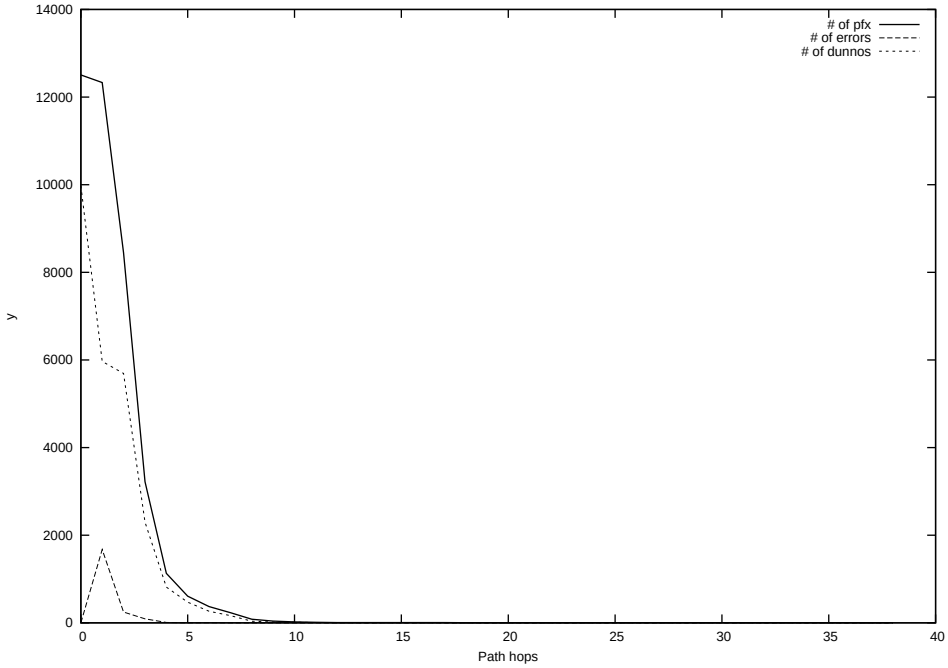
2013-04-28



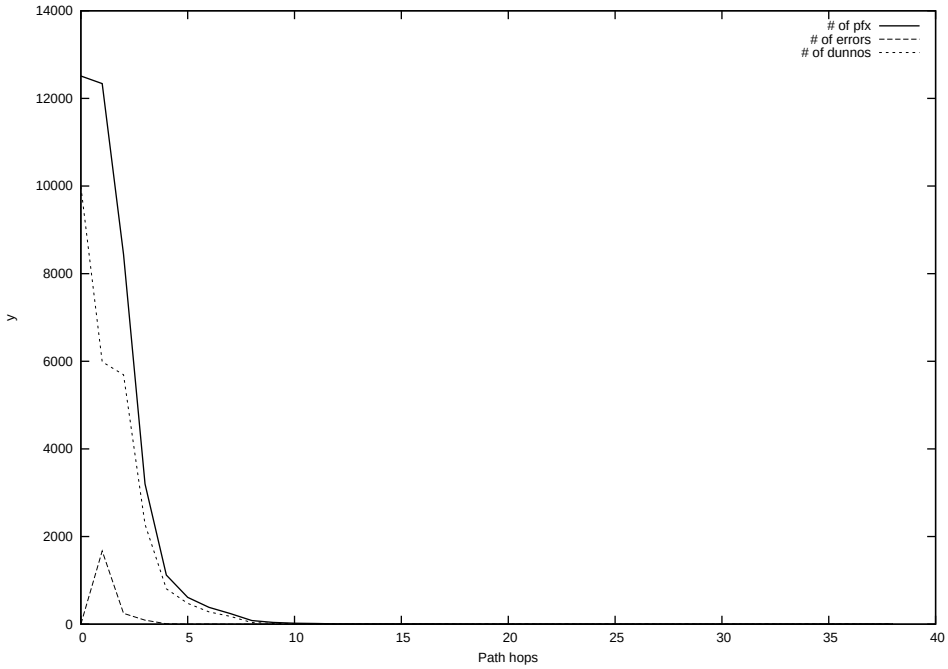
2013-04-29



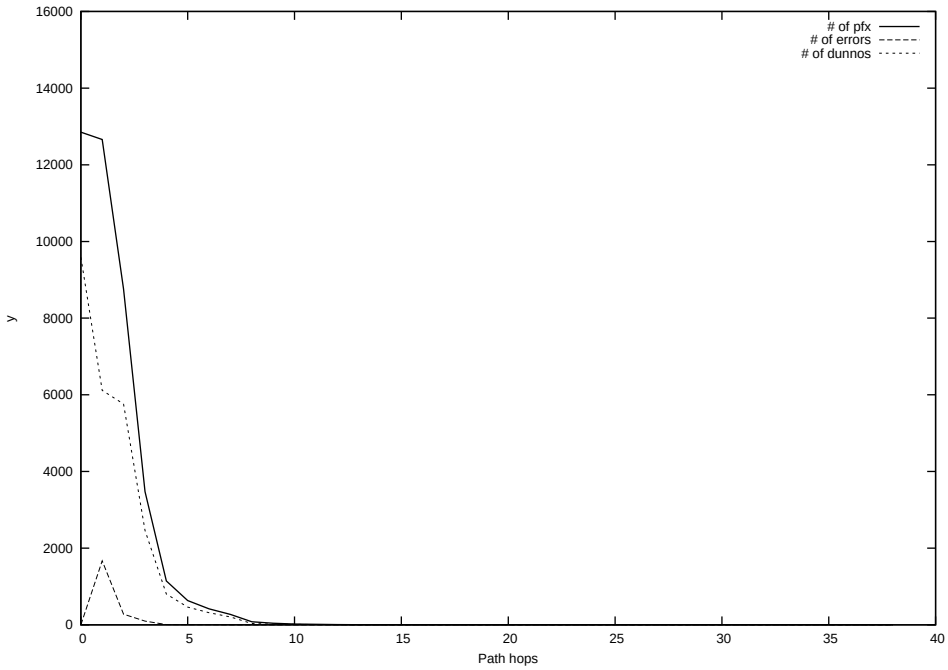
2013-04-30



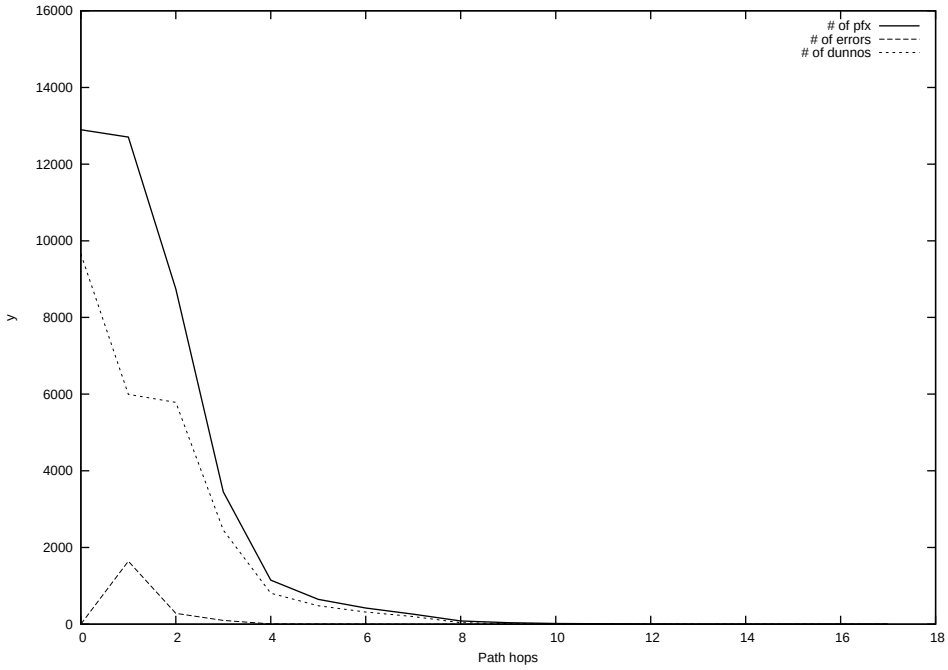
2013-05-01



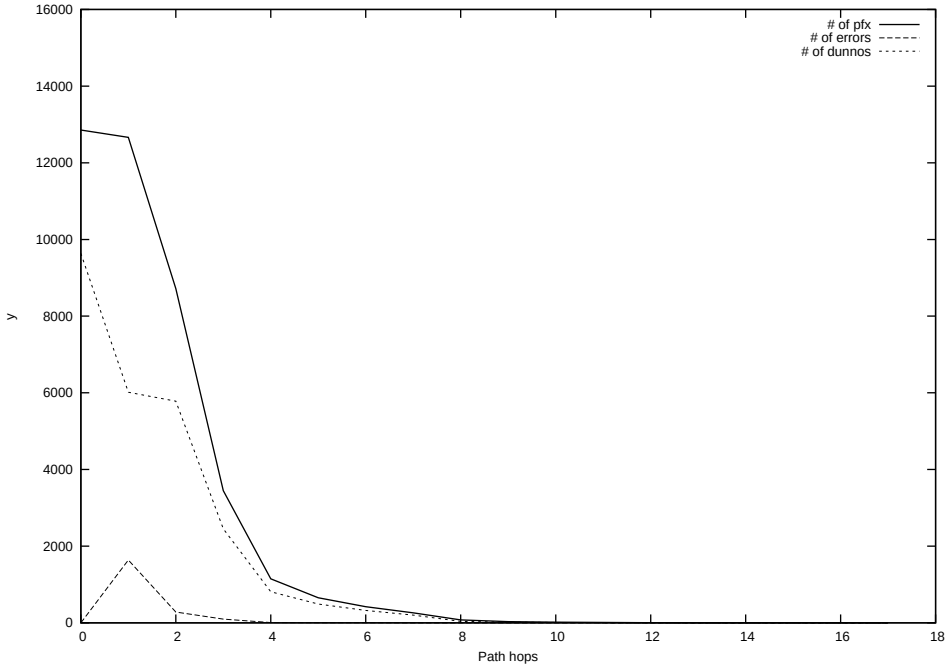
2013-05-02



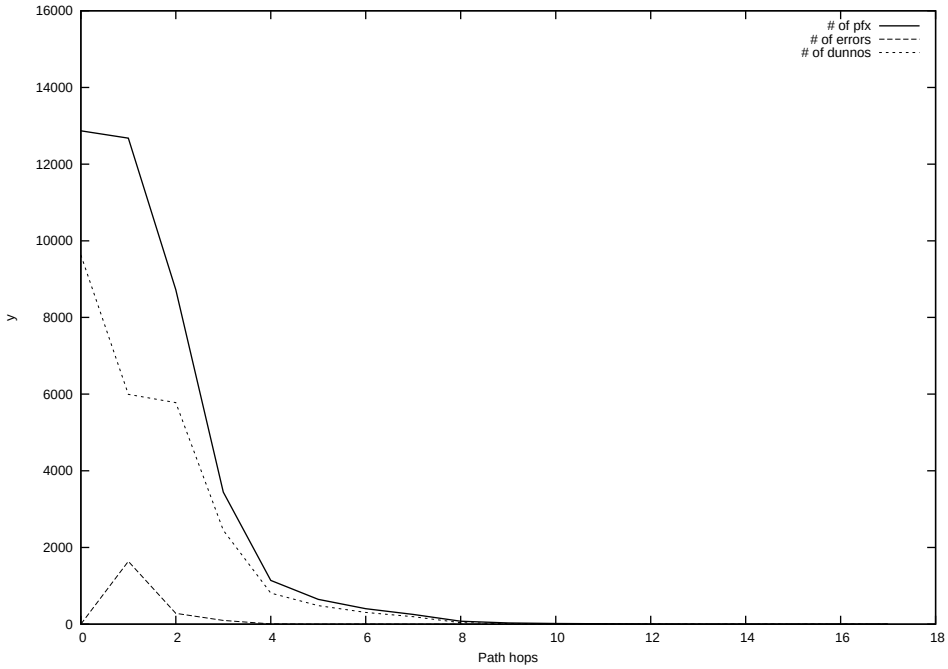
2013-05-03



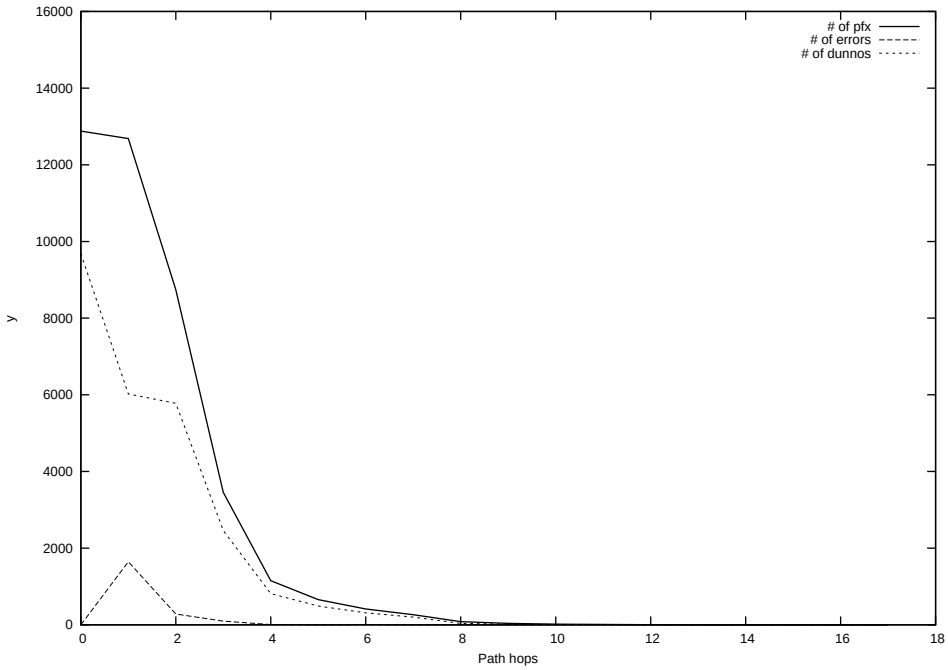
2013-05-04



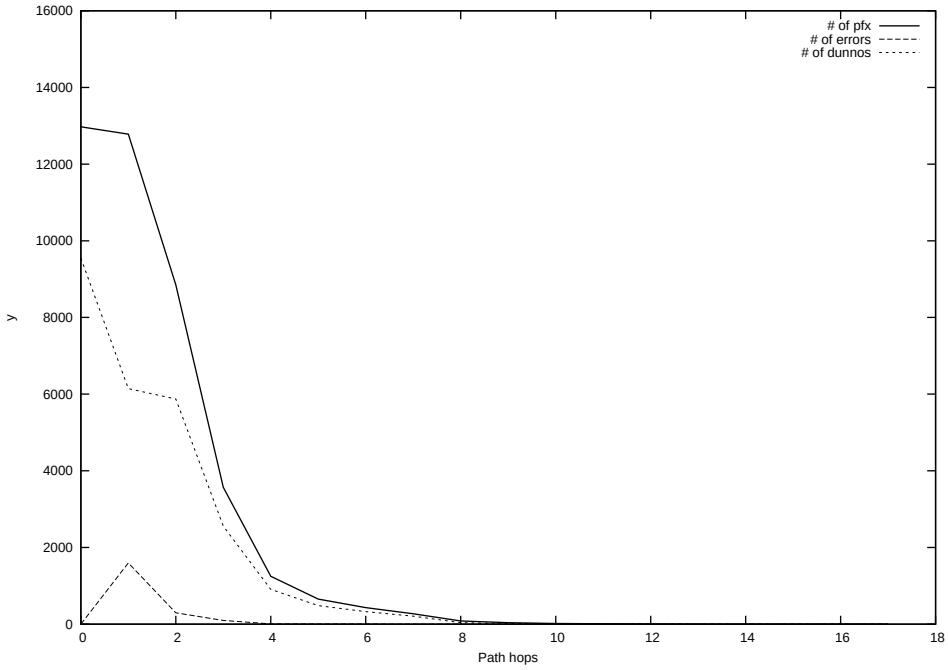
2013-05-05



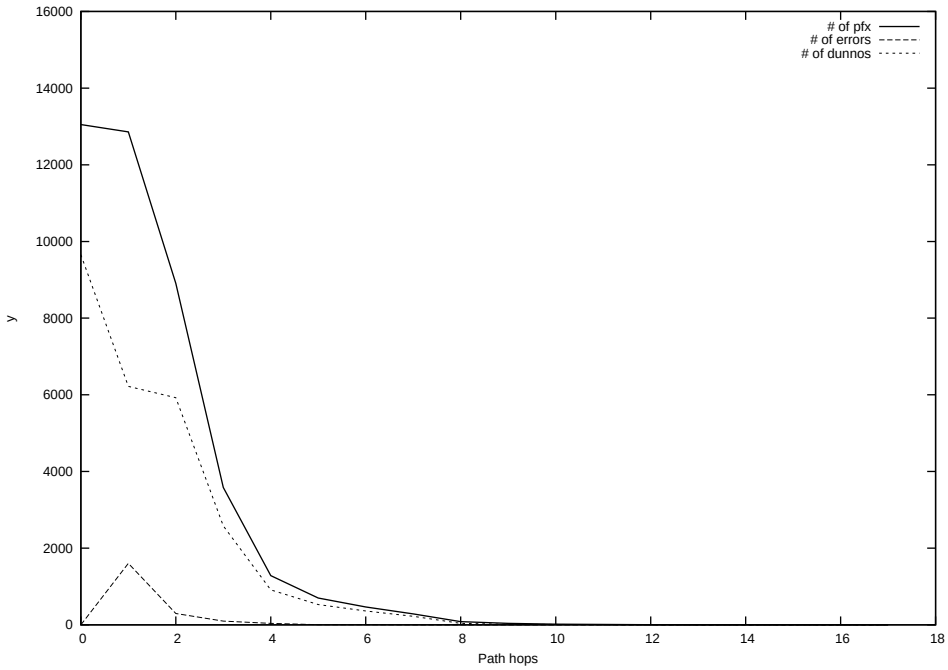
2013-05-06



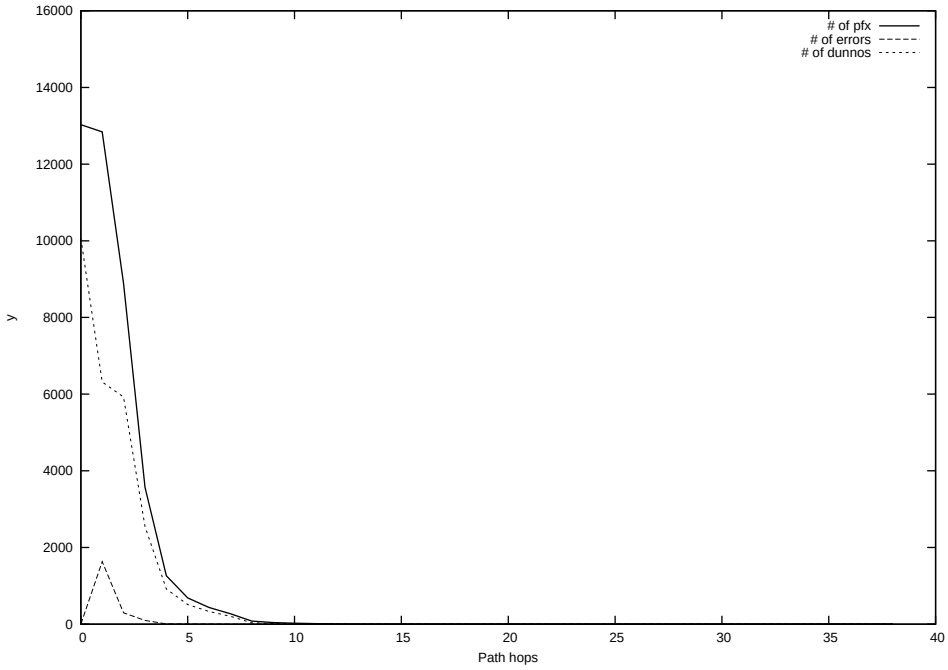
2013-05-07



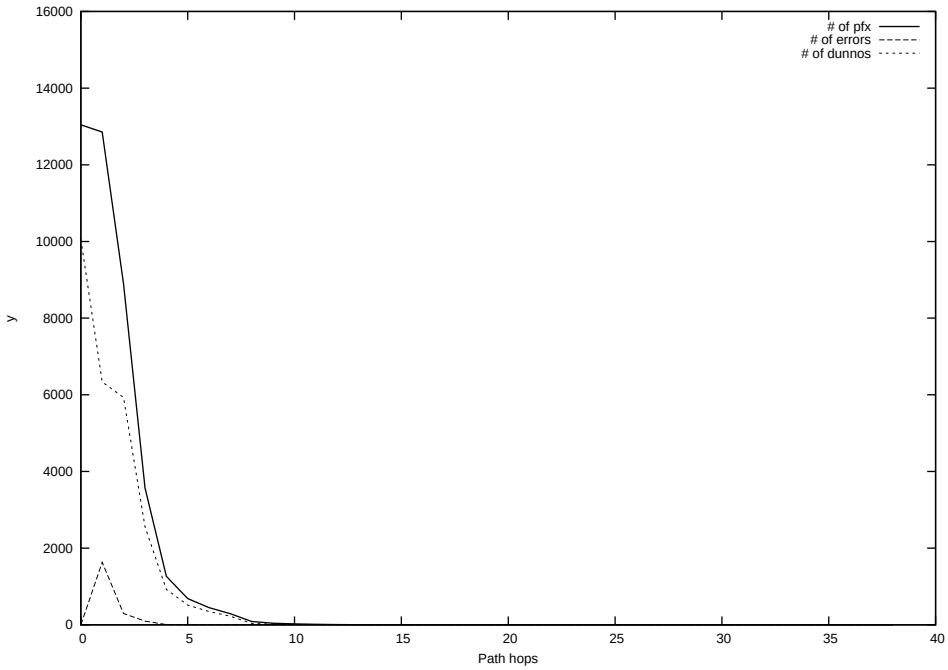
2013-05-08



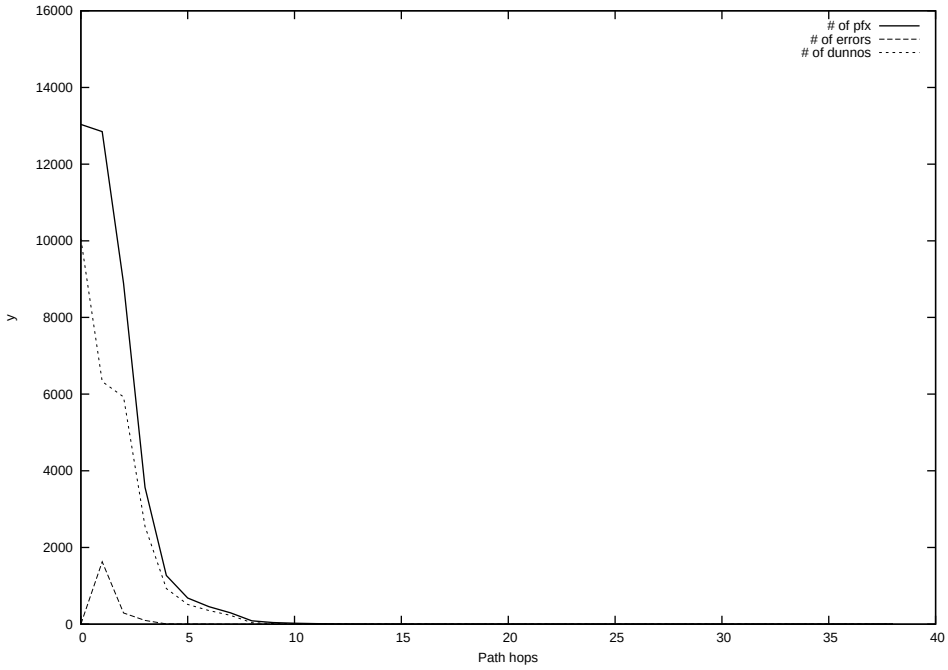
2013-05-09



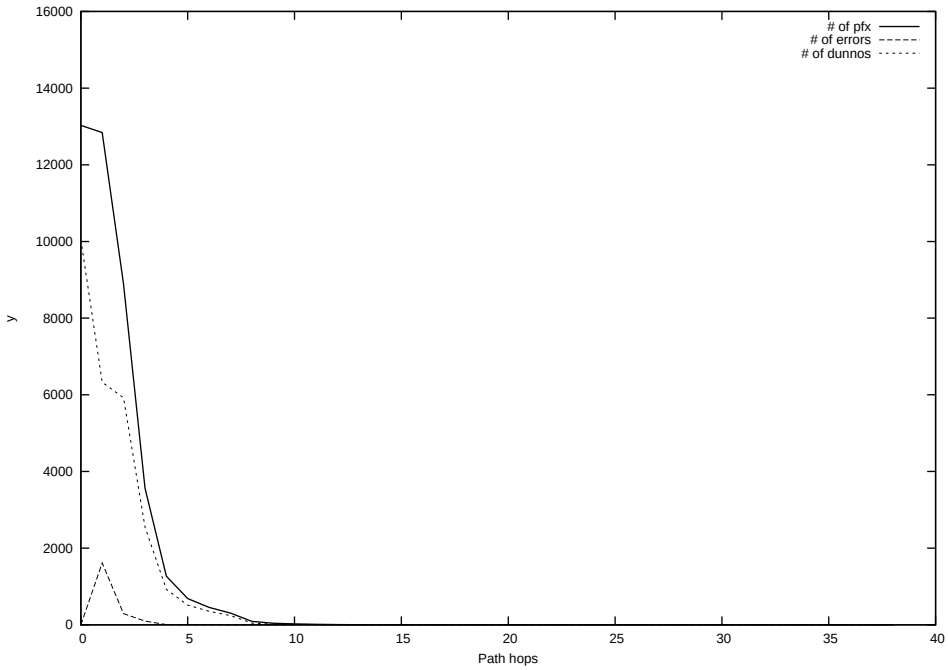
2013-05-10



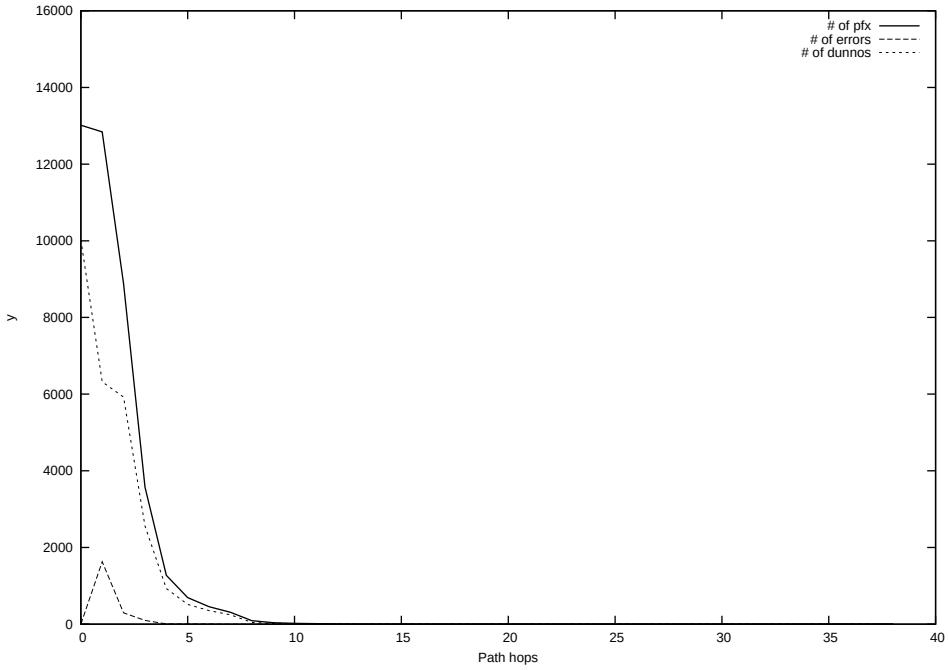
2013-05-11



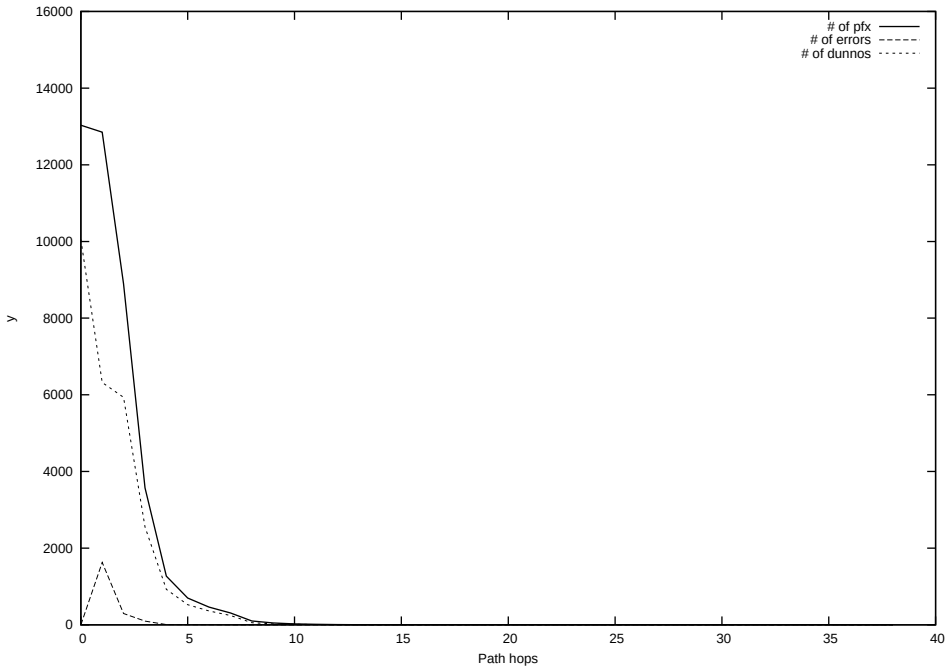
2013-05-12



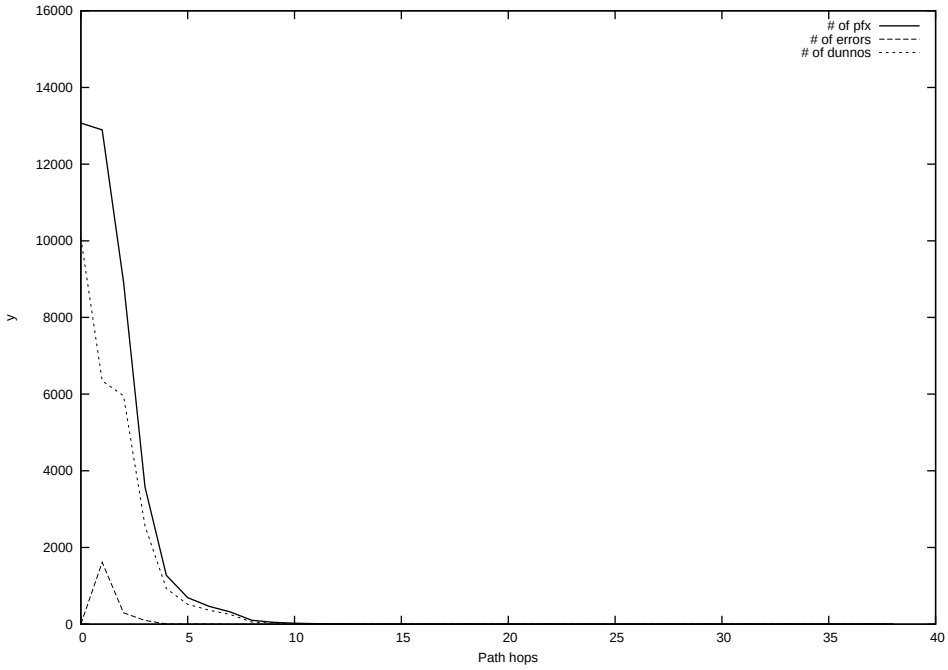
2013-05-13



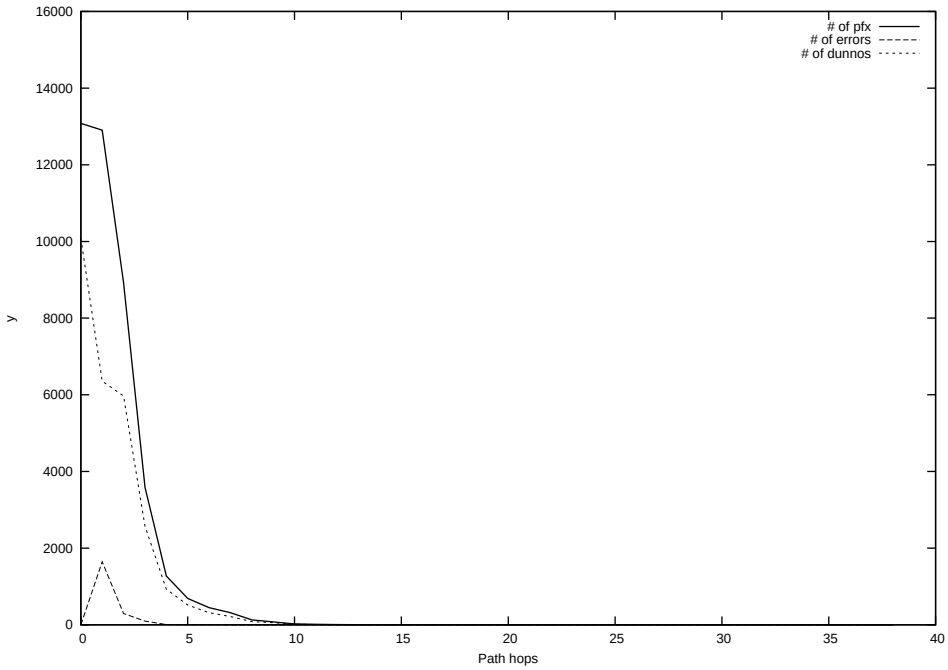
2013-05-14



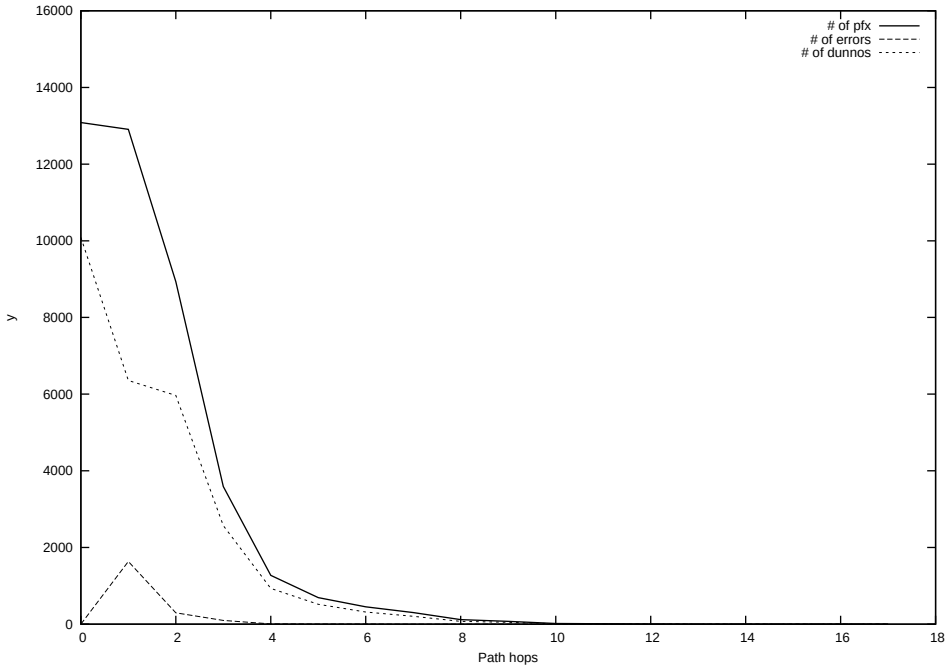
2013-05-15



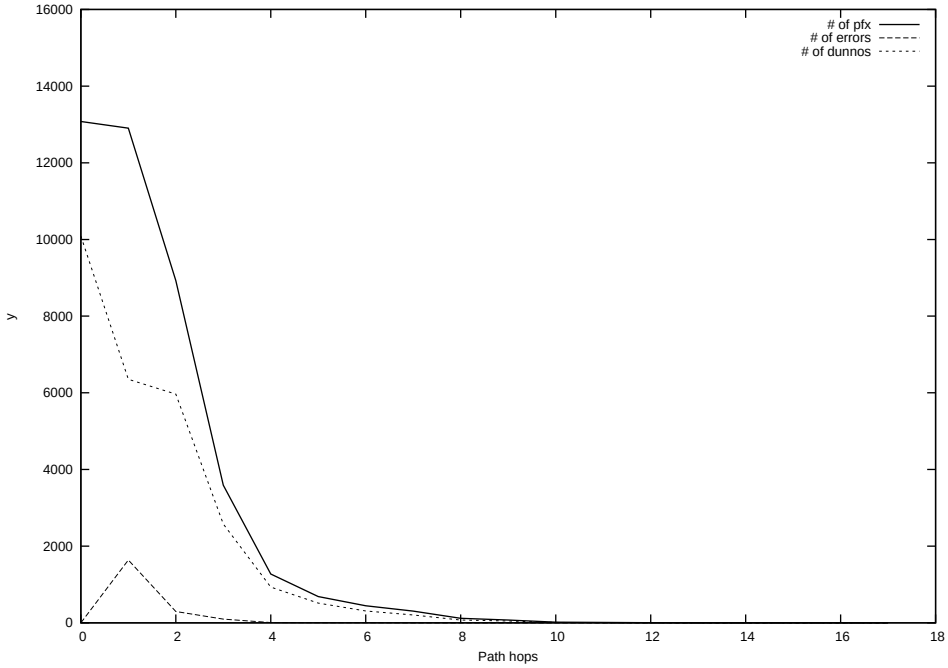
2013-05-16



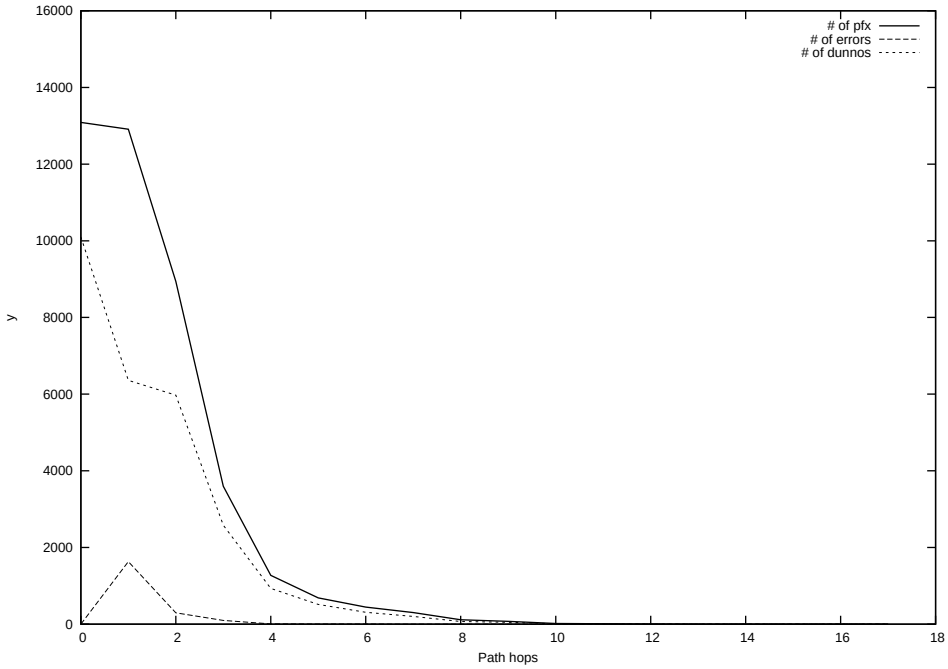
2013-05-17



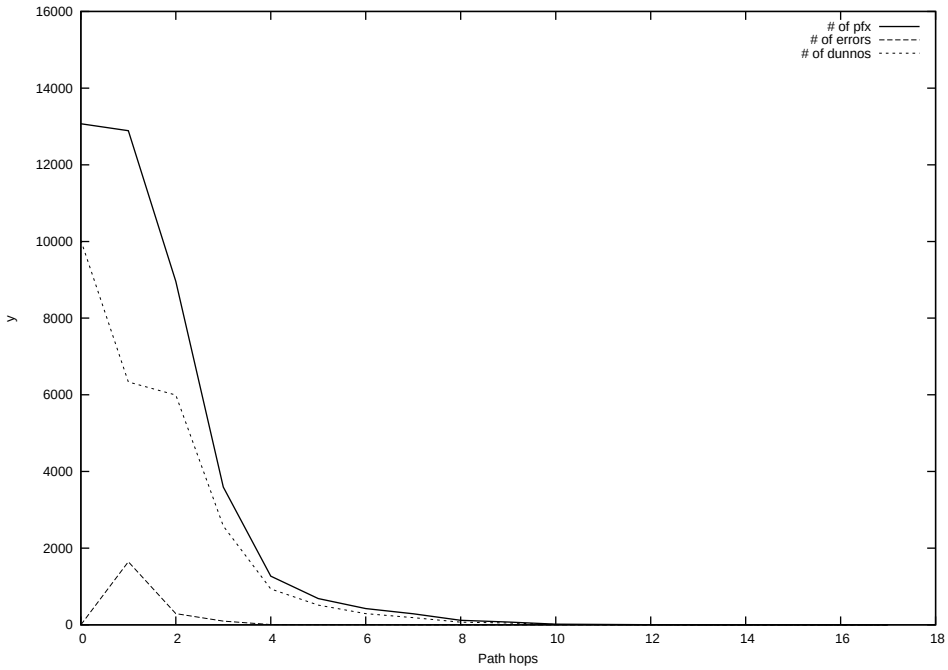
2013-05-18



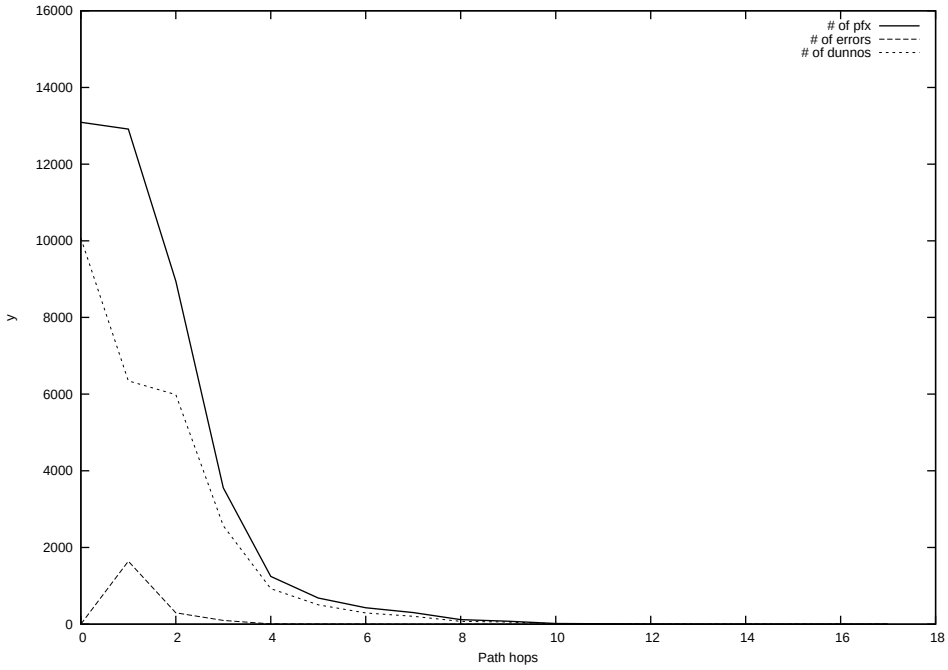
2013-05-19



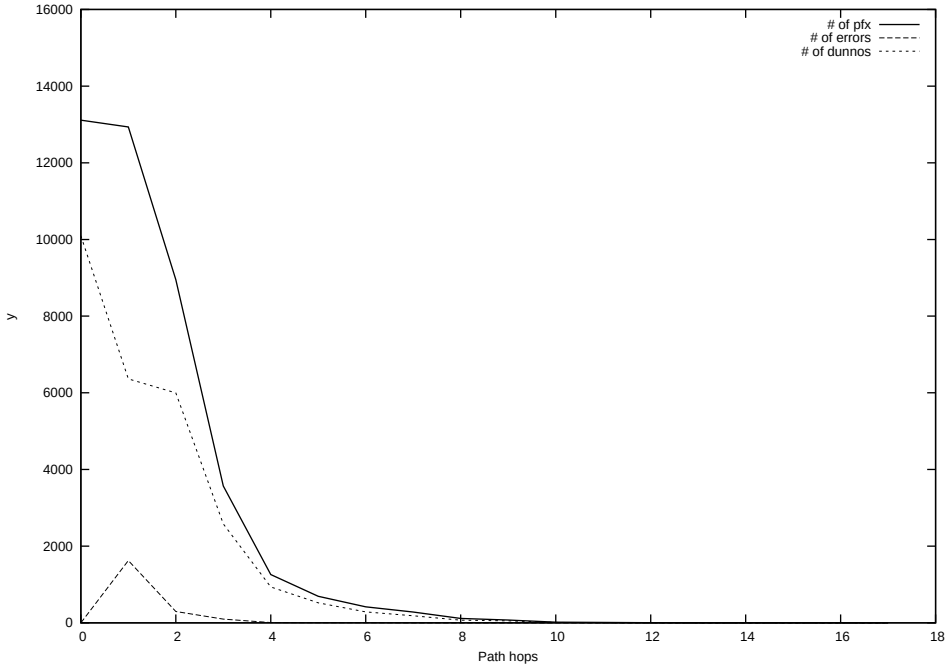
2013-05-20



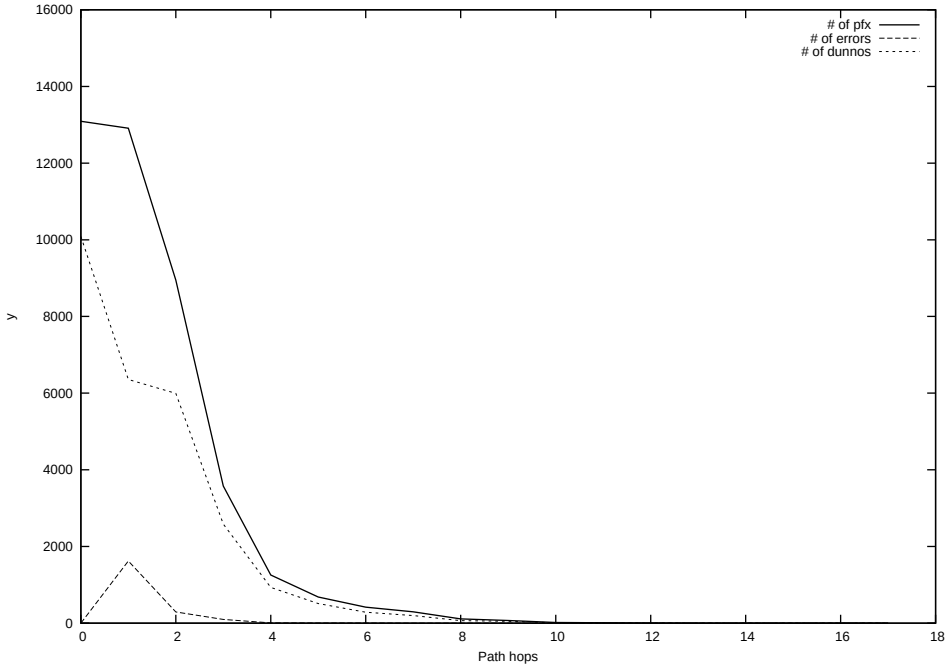
2013-05-21



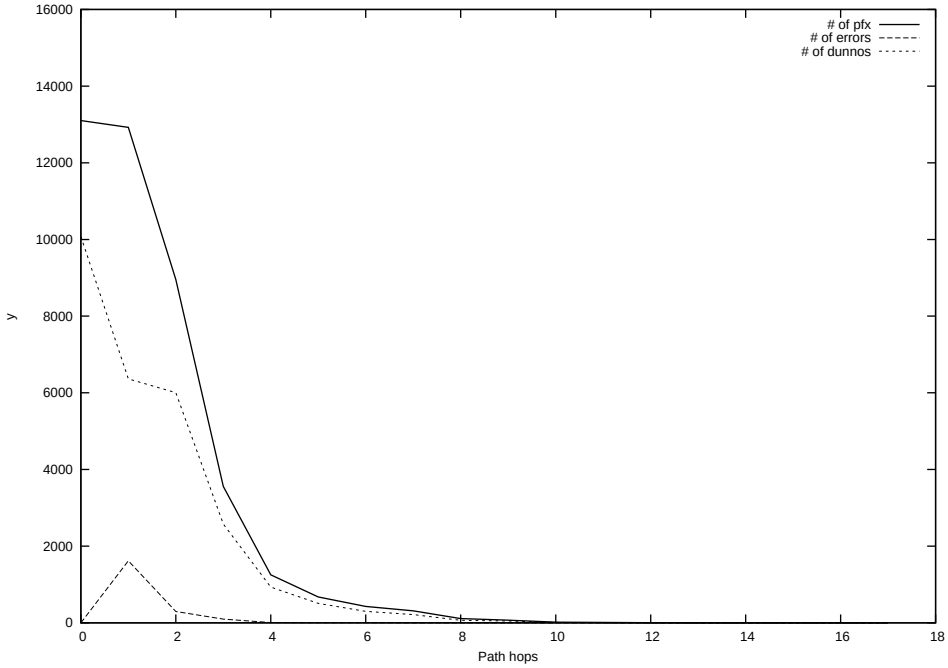
2013-05-22



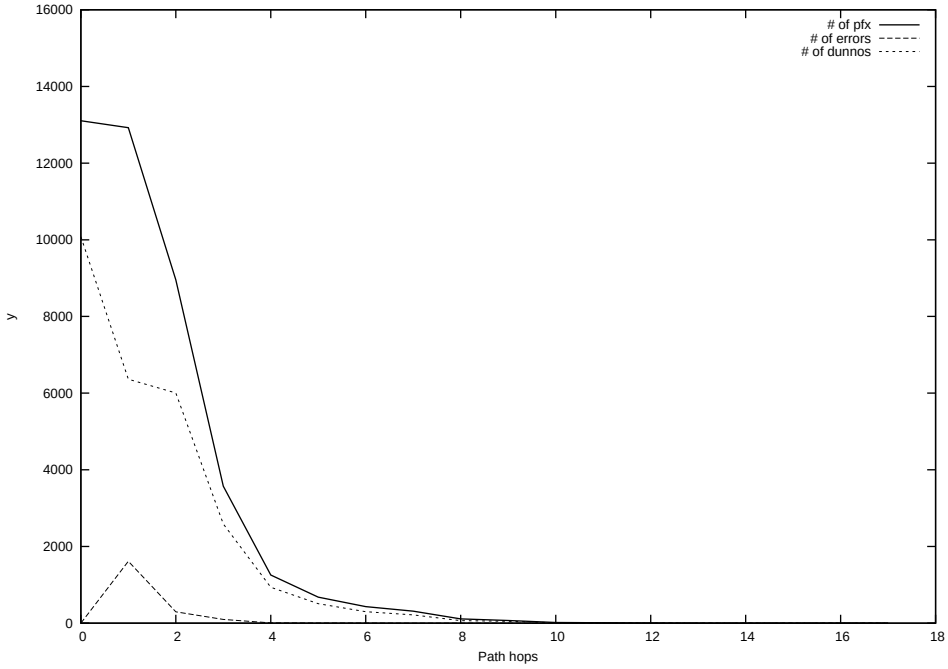
2013-05-23



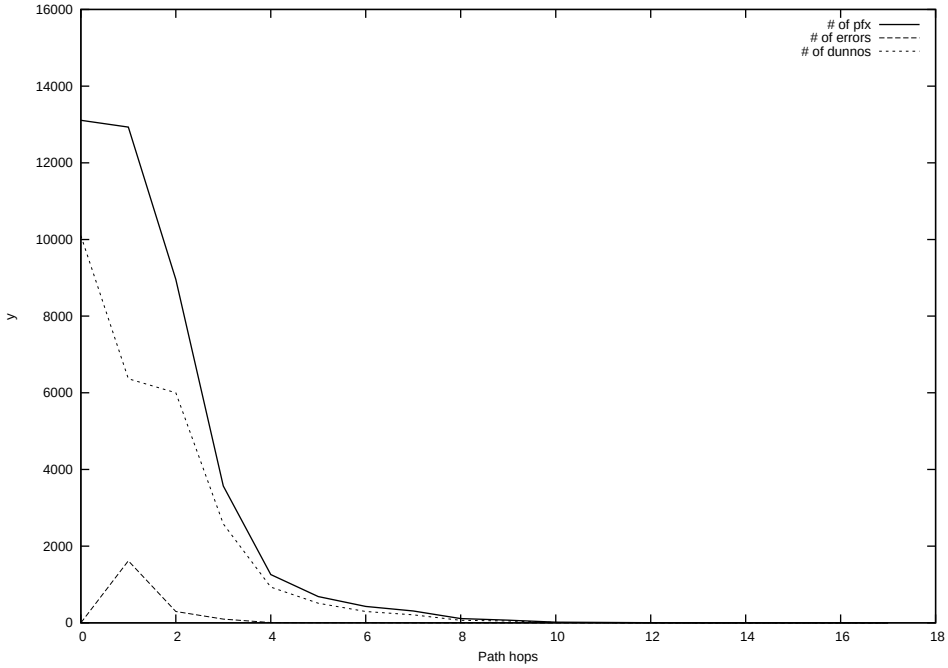
2013-05-24



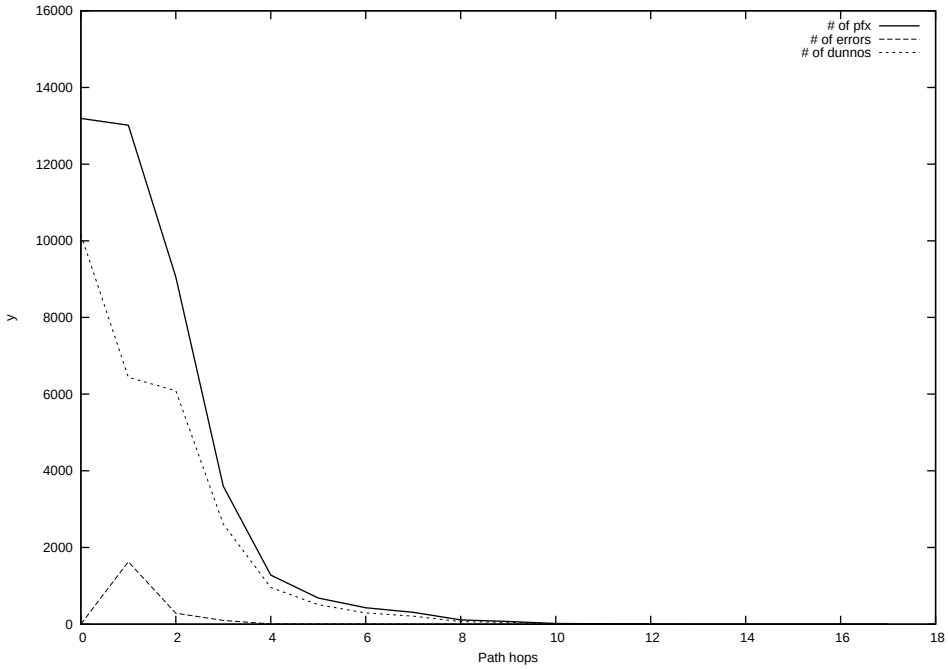
2013-05-25



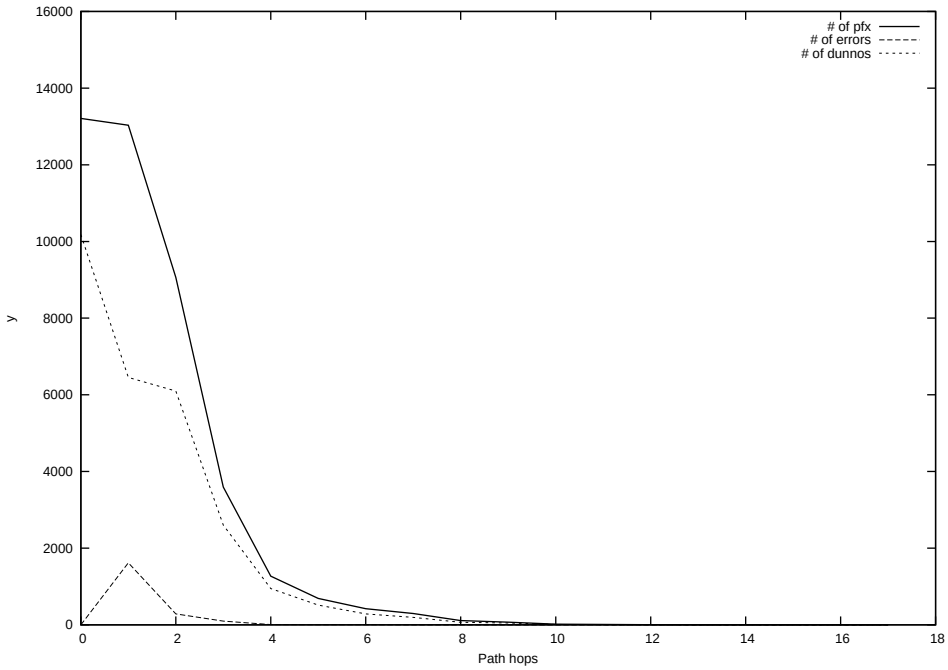
2013-05-26



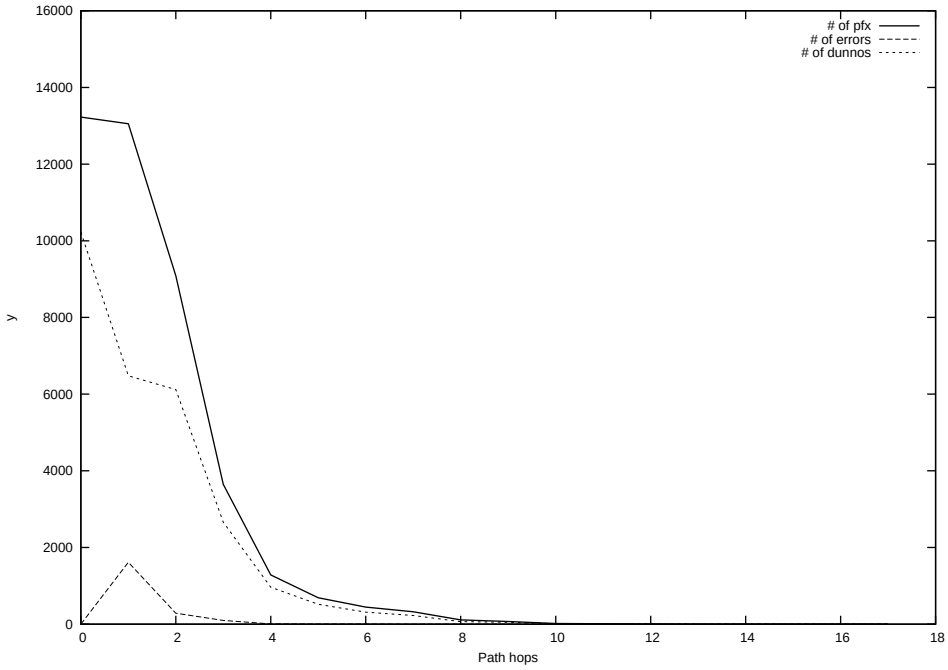
2013-05-27



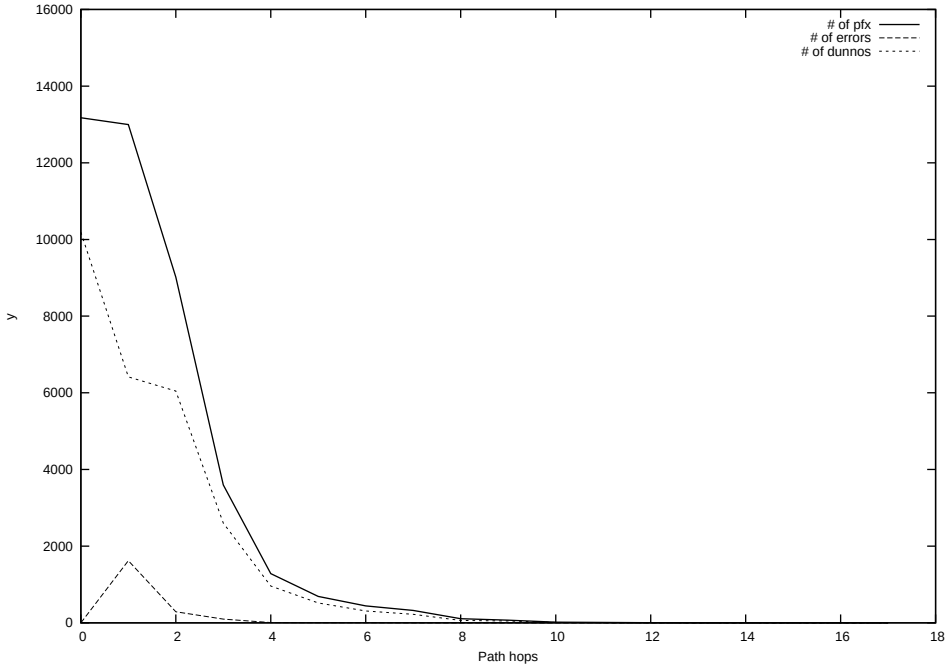
2013-05-28



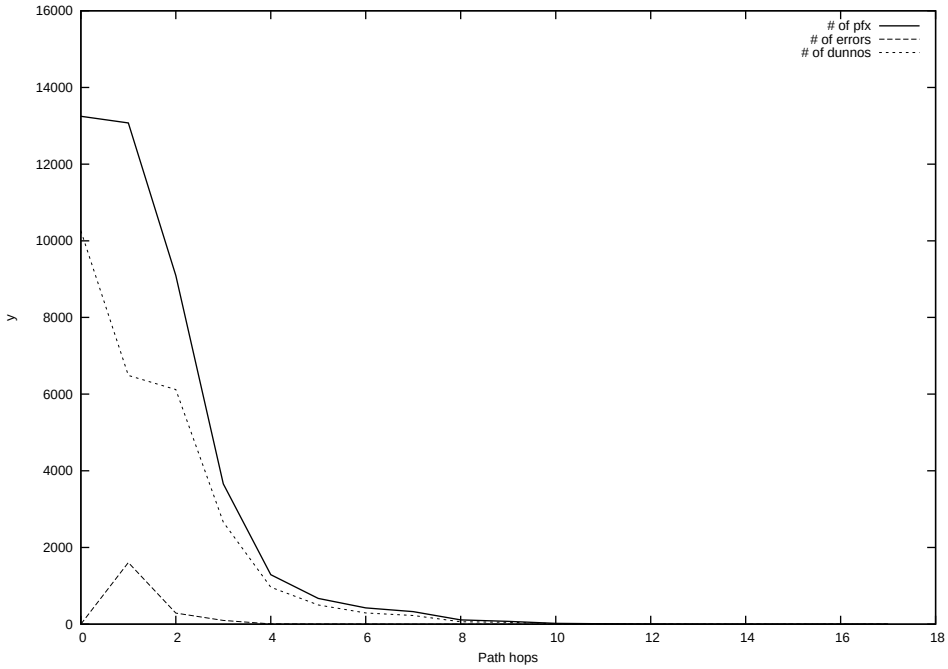
2013-05-29



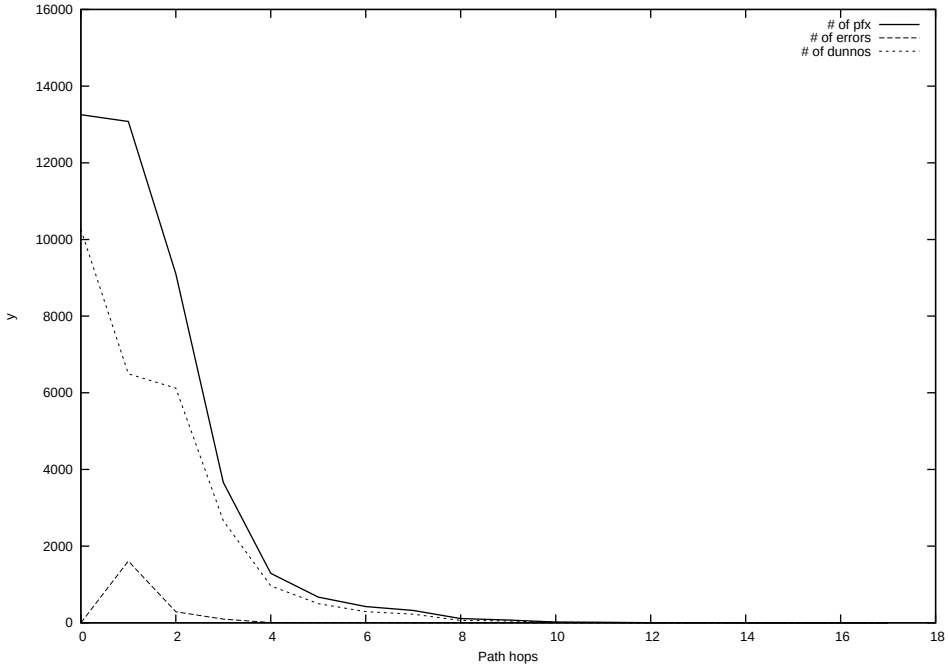
2013-05-30



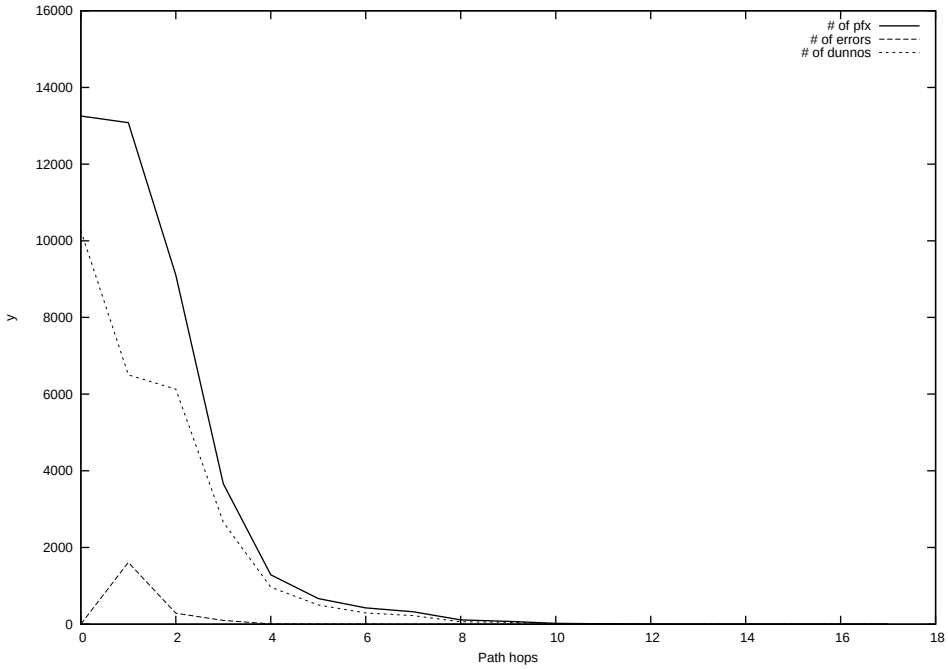
2013-05-31



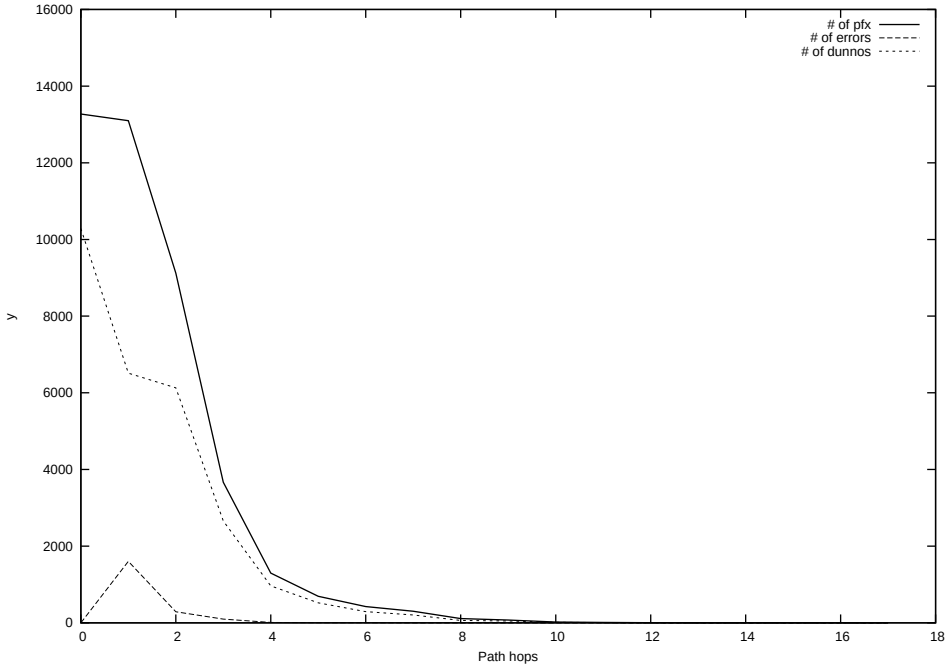
2013-06-01



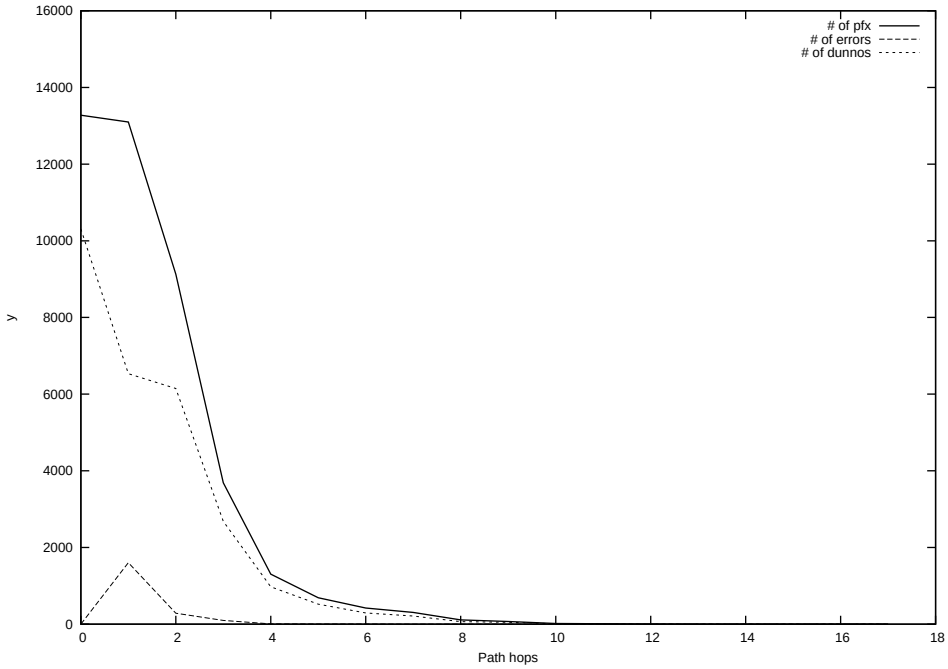
2013-06-02



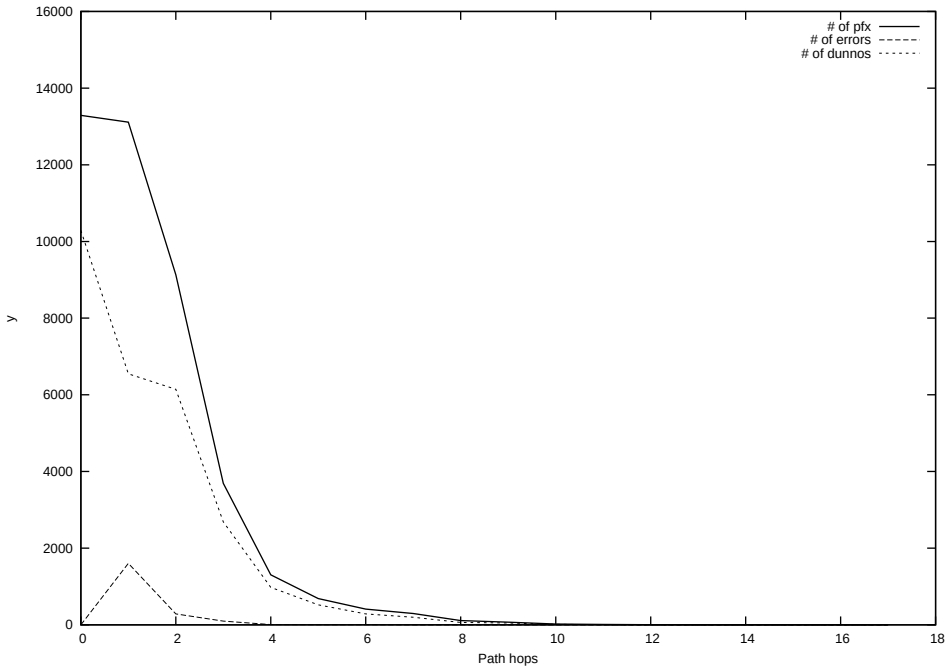
2013-06-03



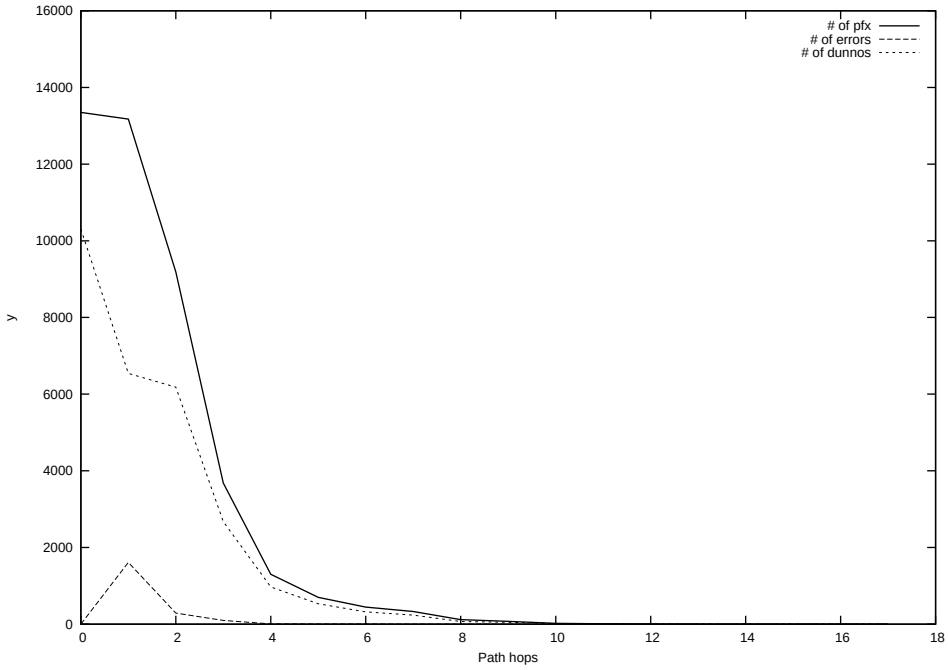
2013-06-04



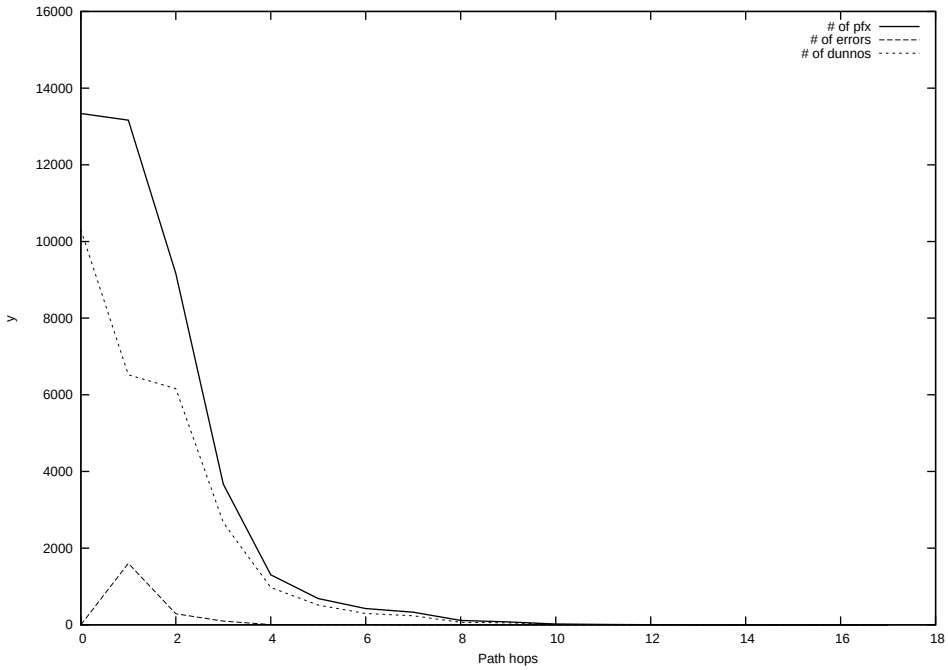
2013-06-05



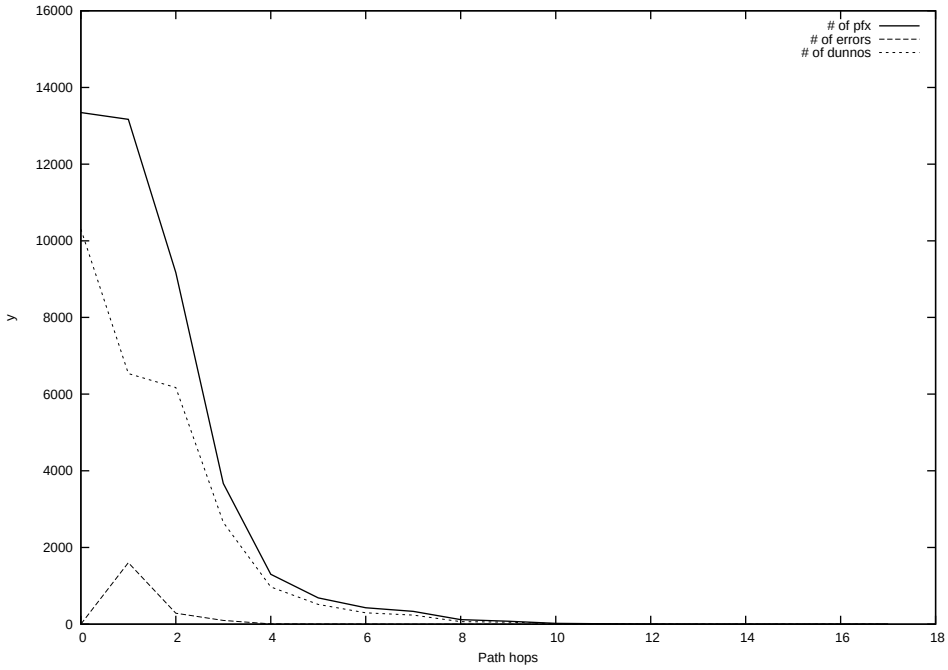
2013-06-06



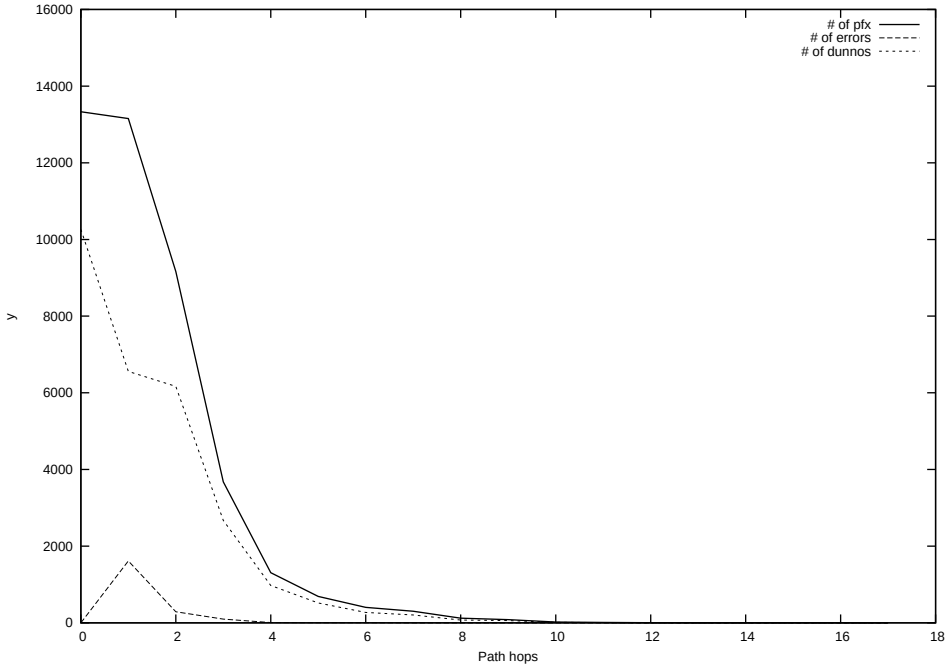
2013-06-07



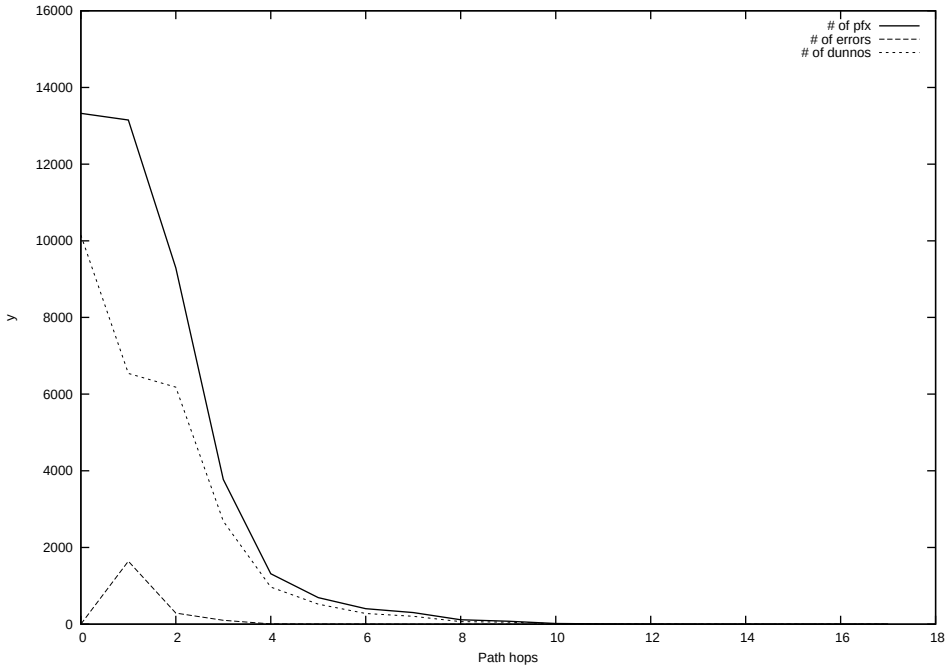
2013-06-09



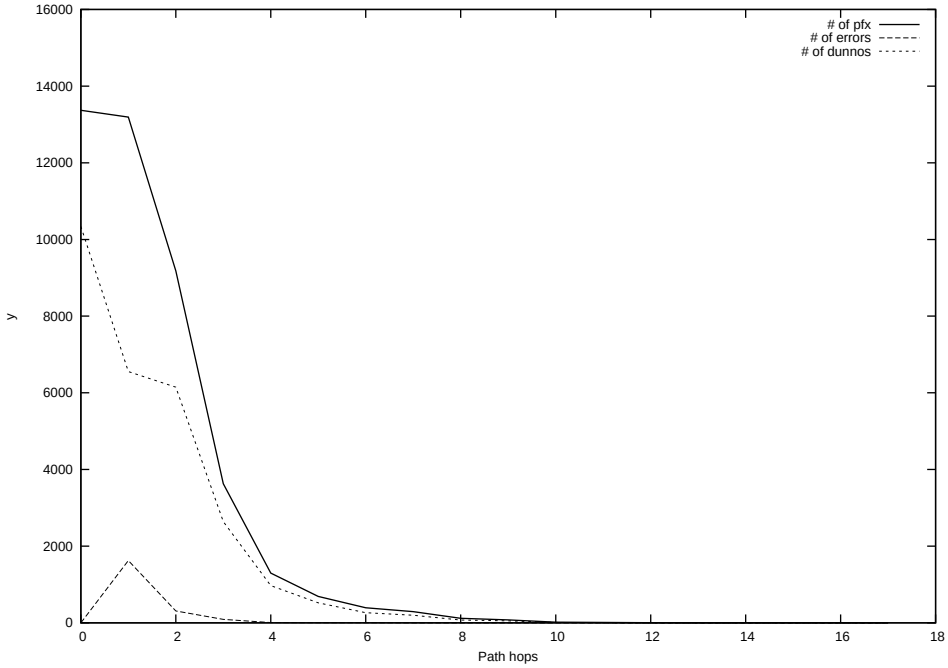
2013-06-10



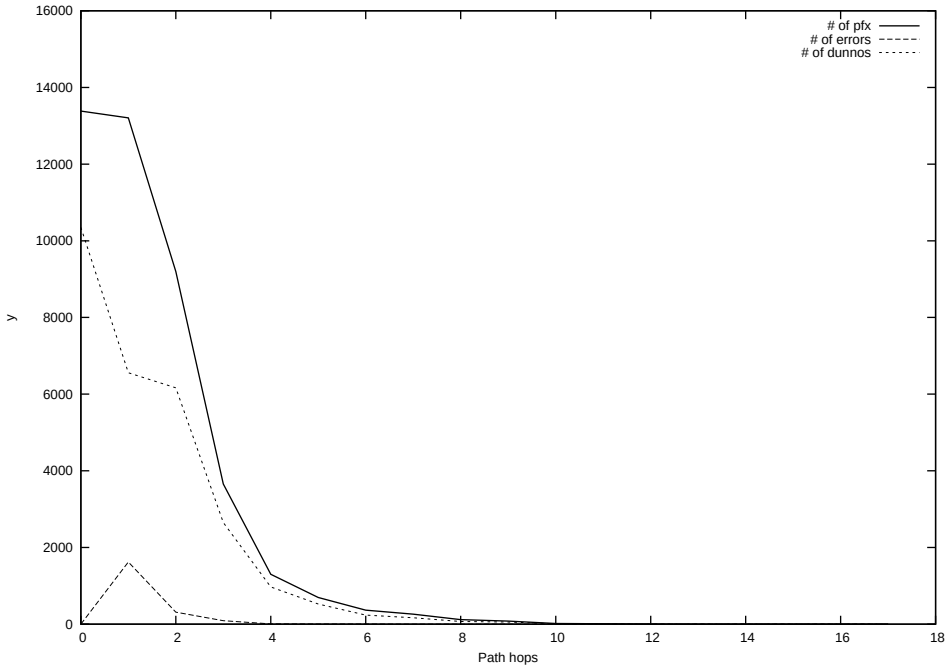
2013-06-12



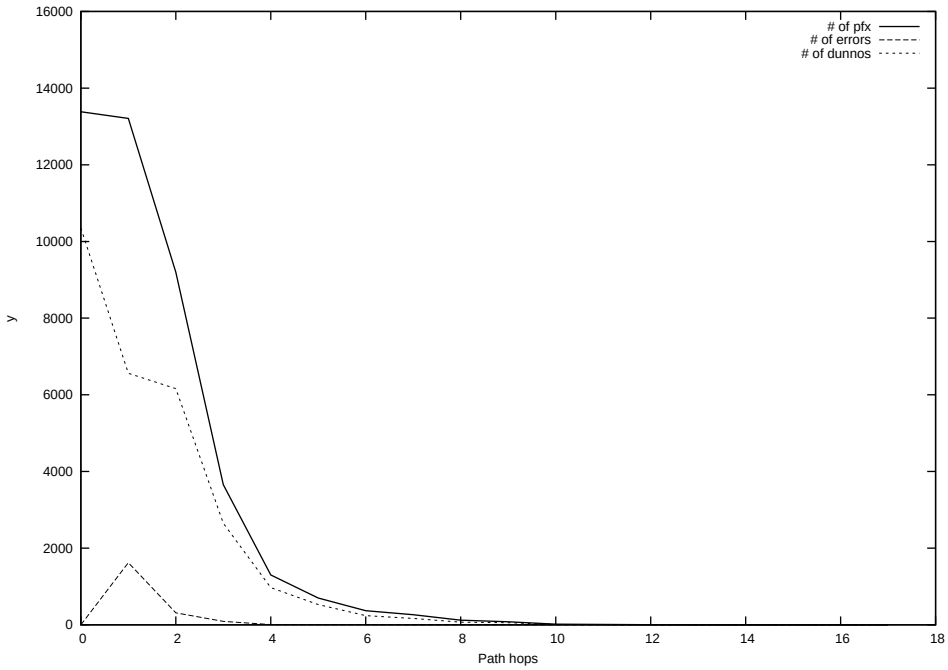
2013-06-13



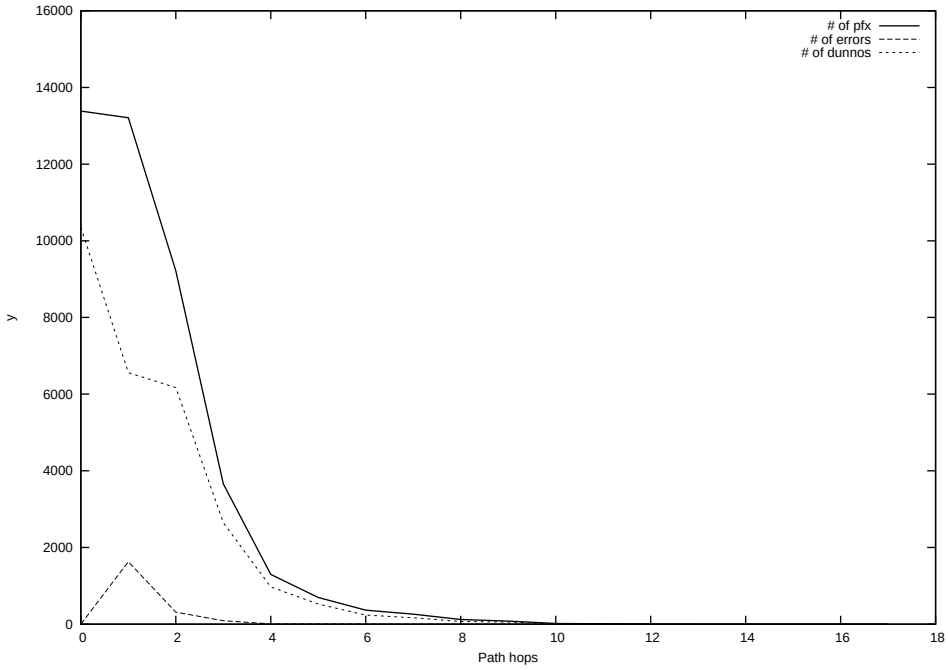
2013-06-14



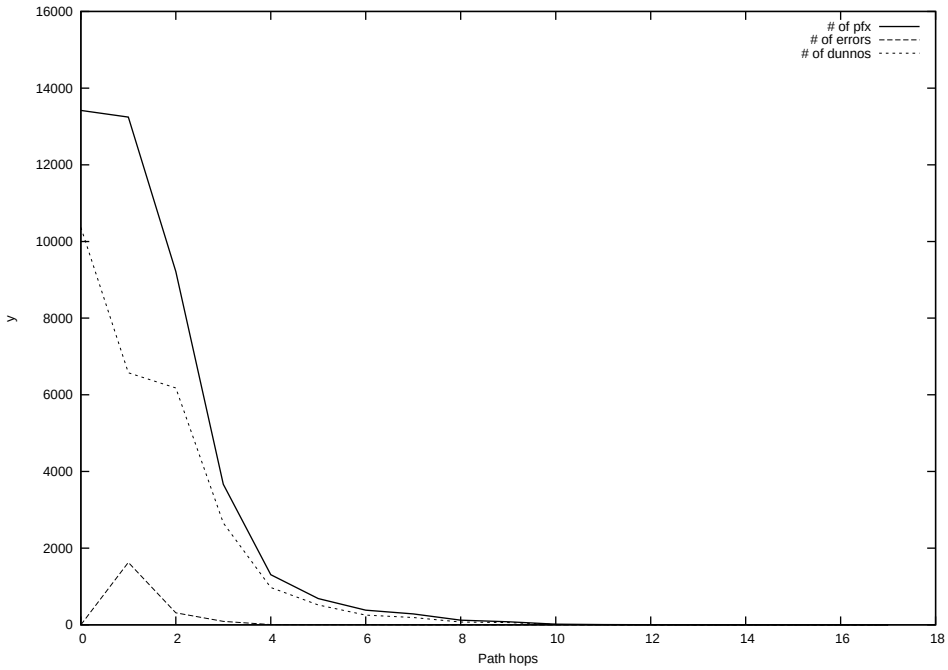
2013-06-15



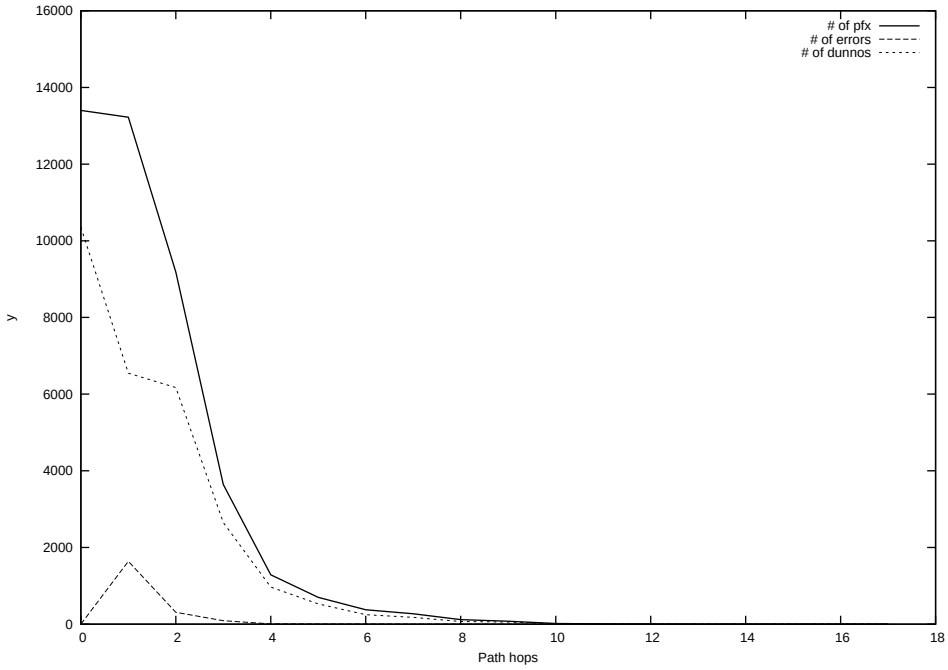
2013-06-16



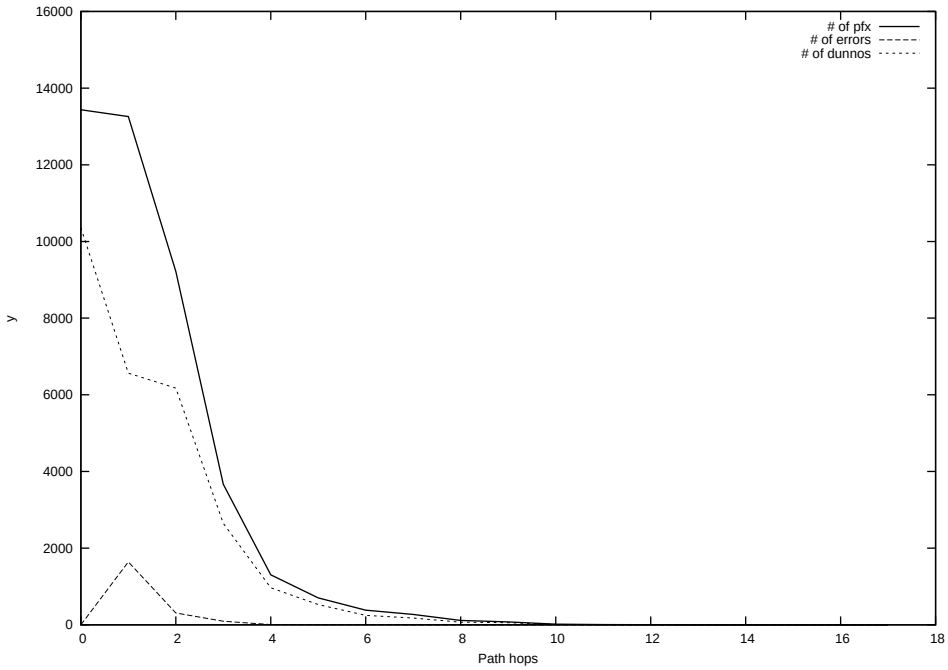
2013-06-17



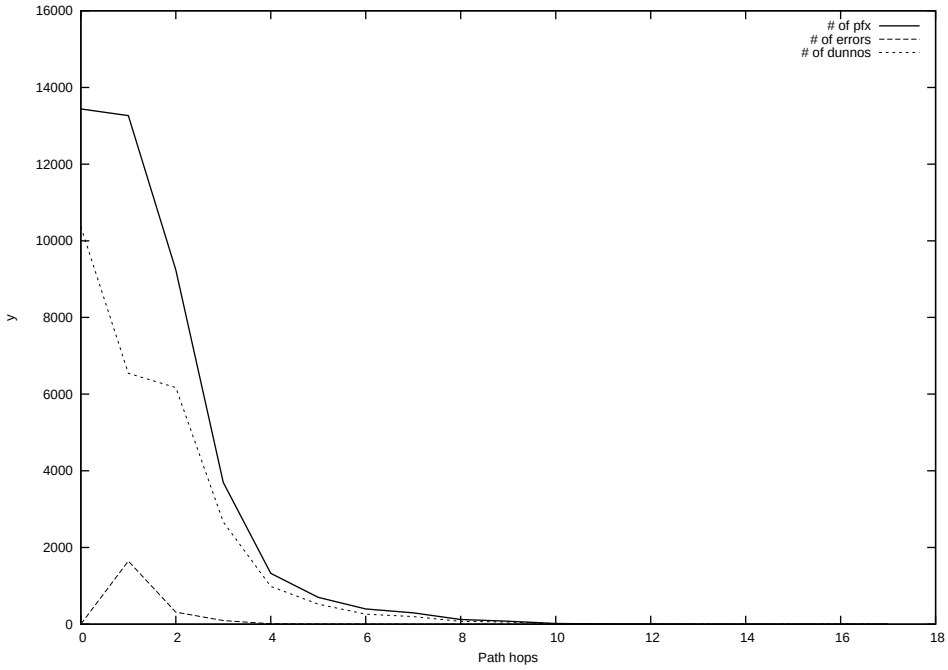
2013-06-18



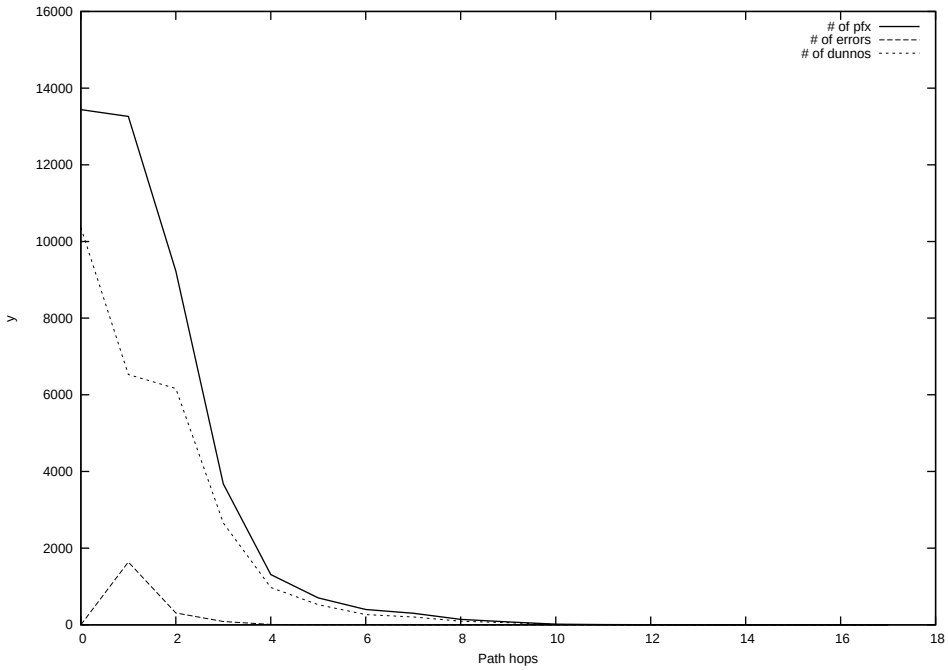
2013-06-19



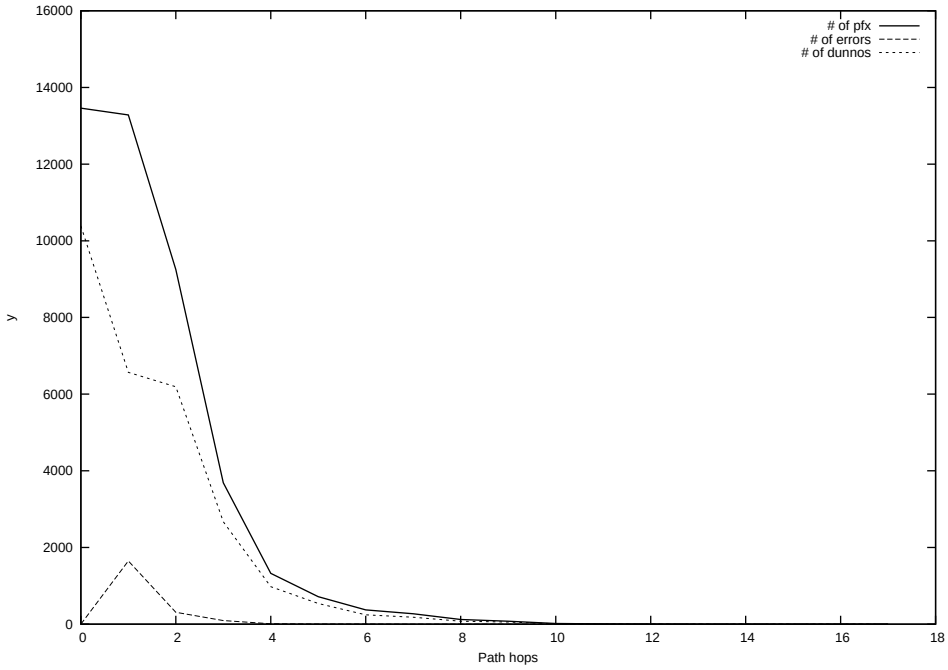
2013-06-20



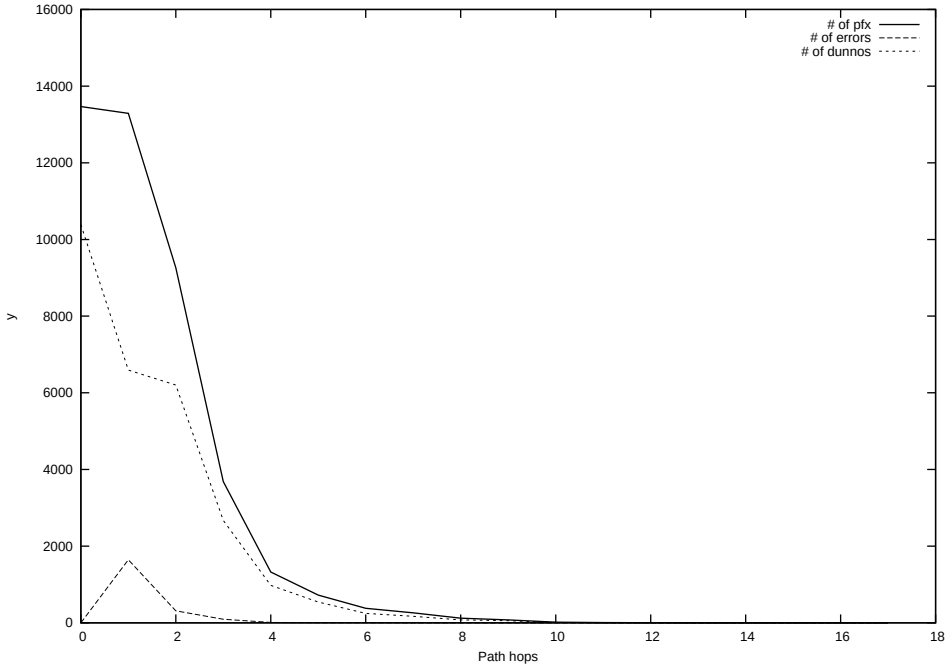
2013-06-21



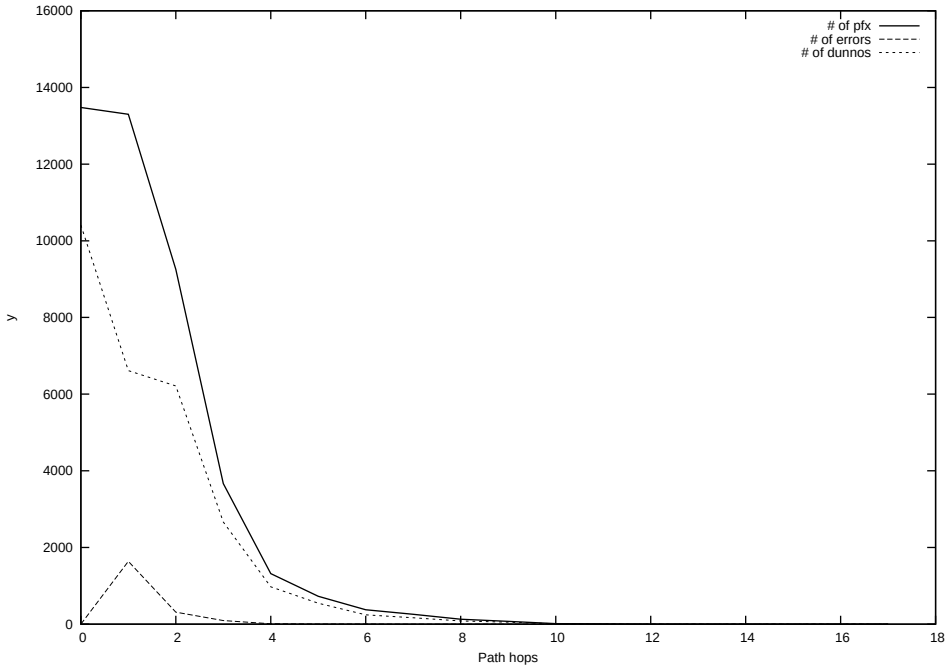
2013-06-22



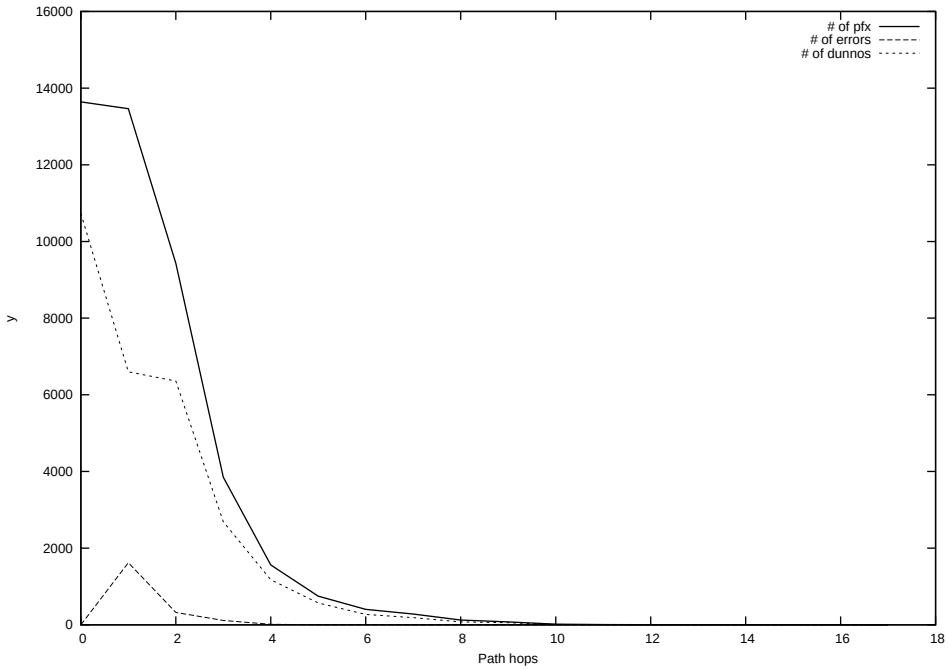
2013-06-23



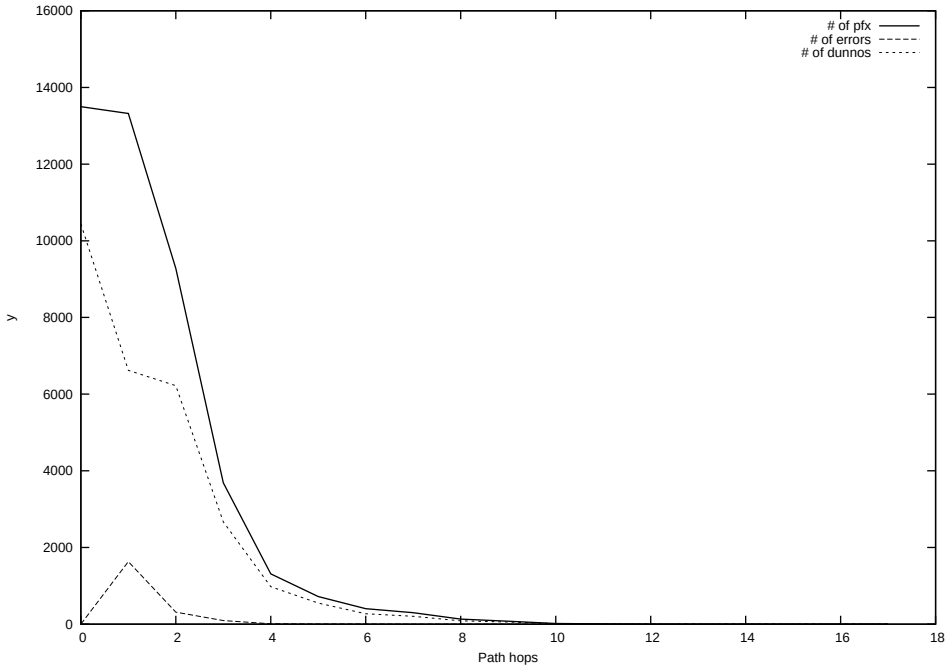
2013-06-24



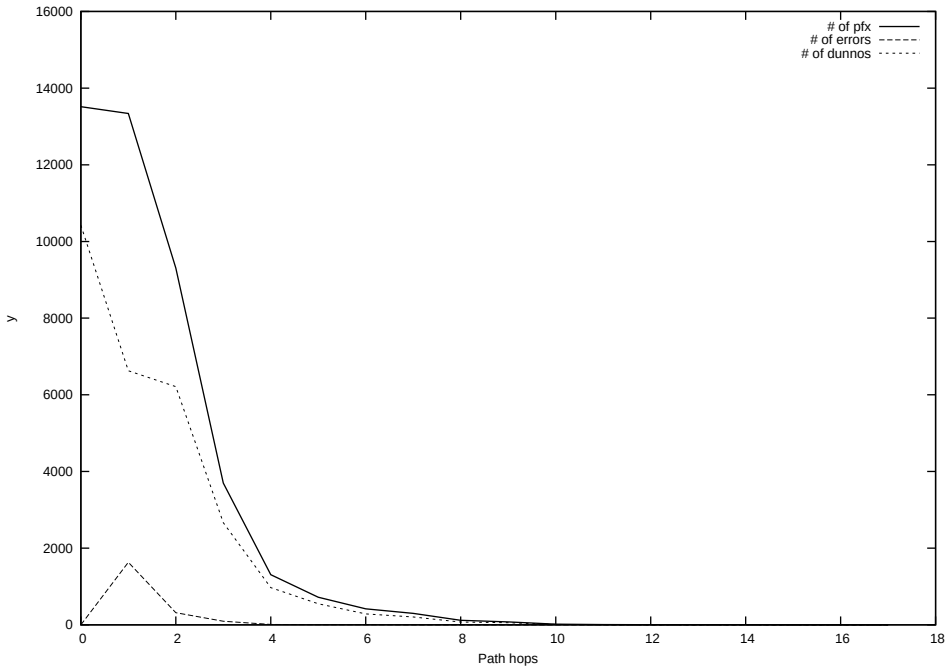
2013-06-25



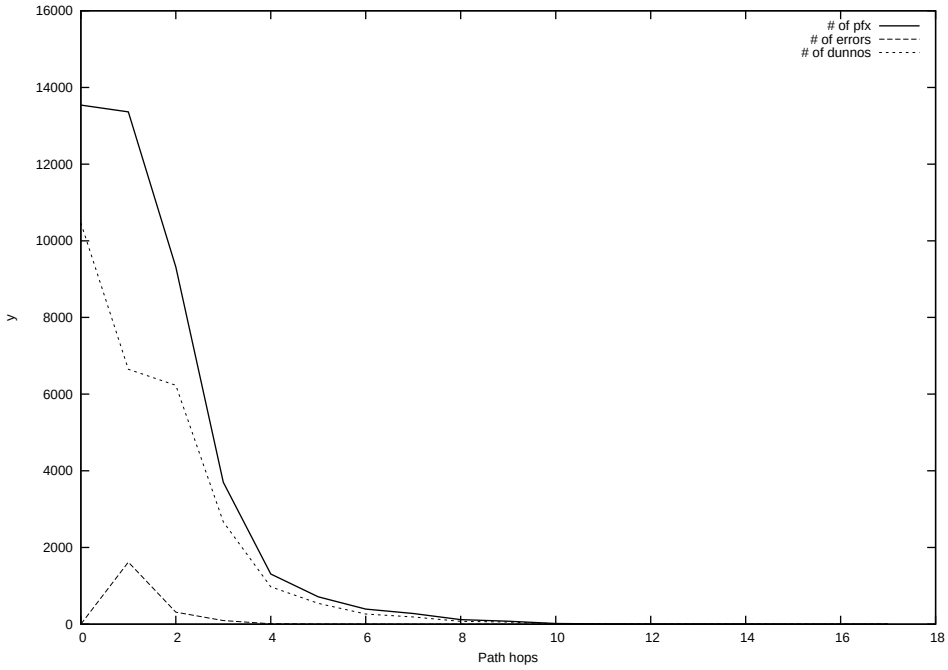
2013-06-26



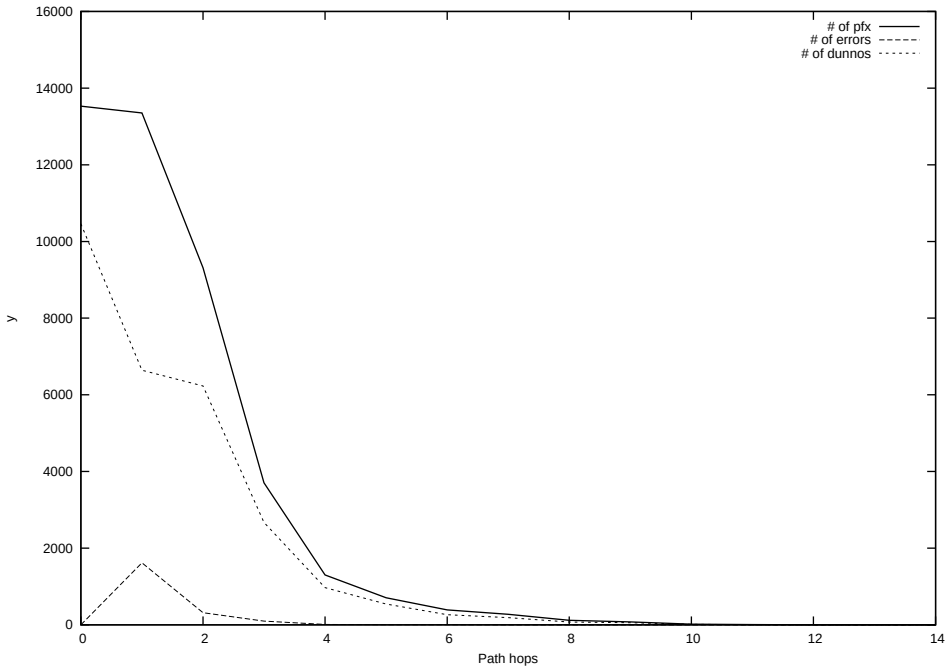
2013-06-27



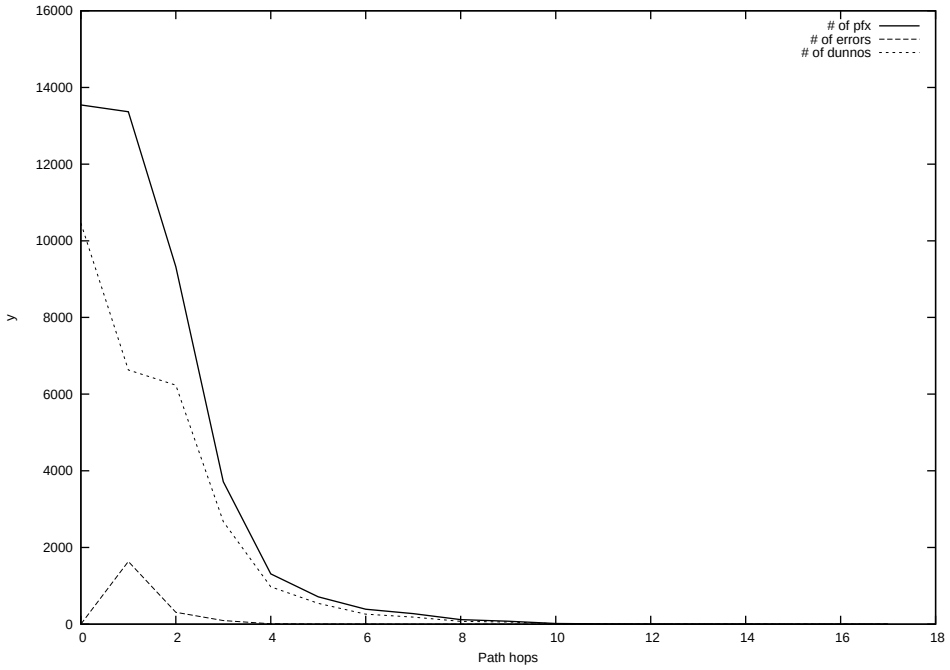
2013-06-28



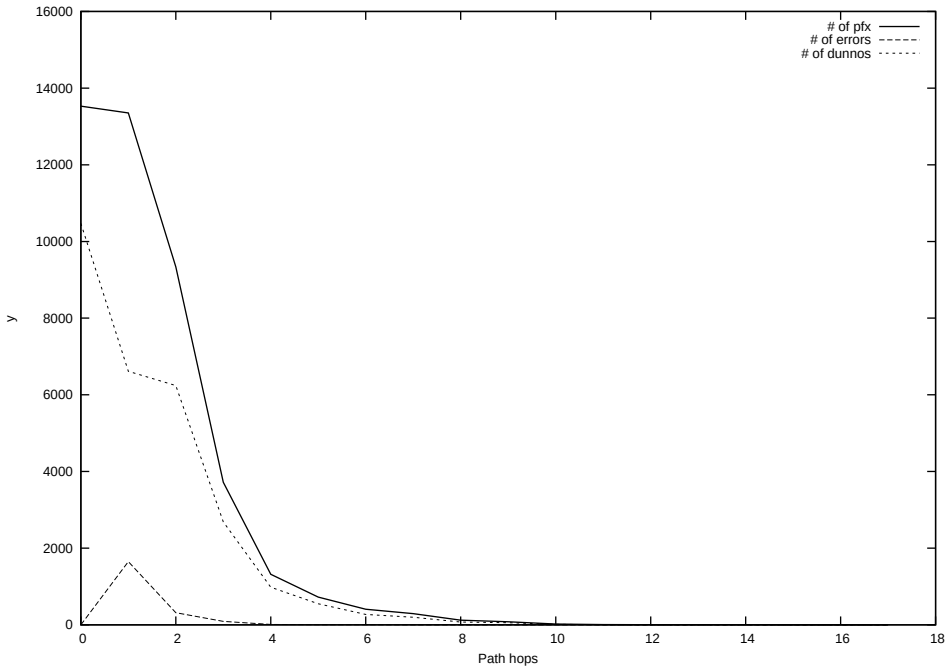
2013-06-29



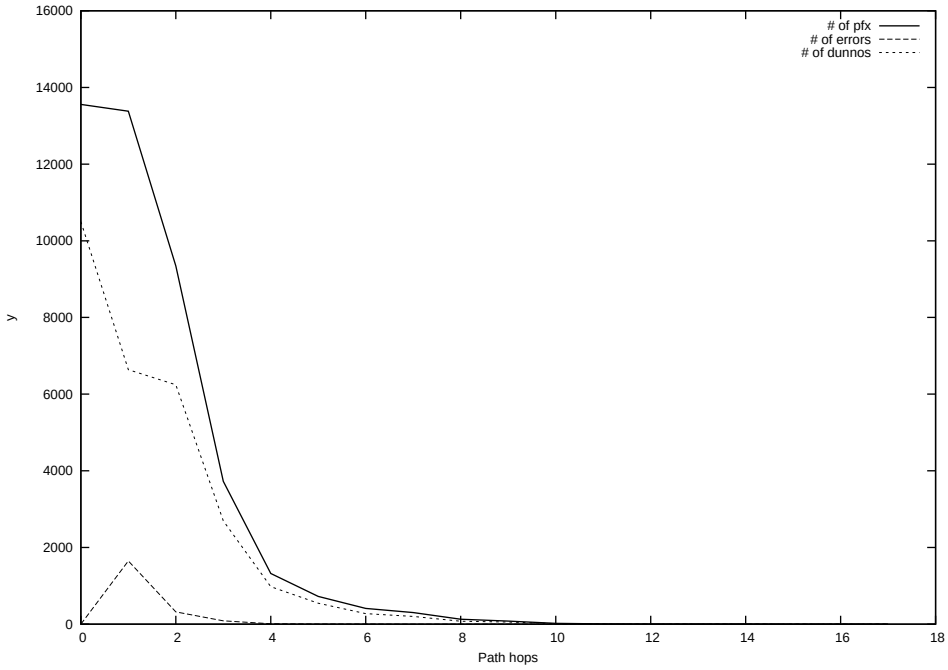
2013-06-30



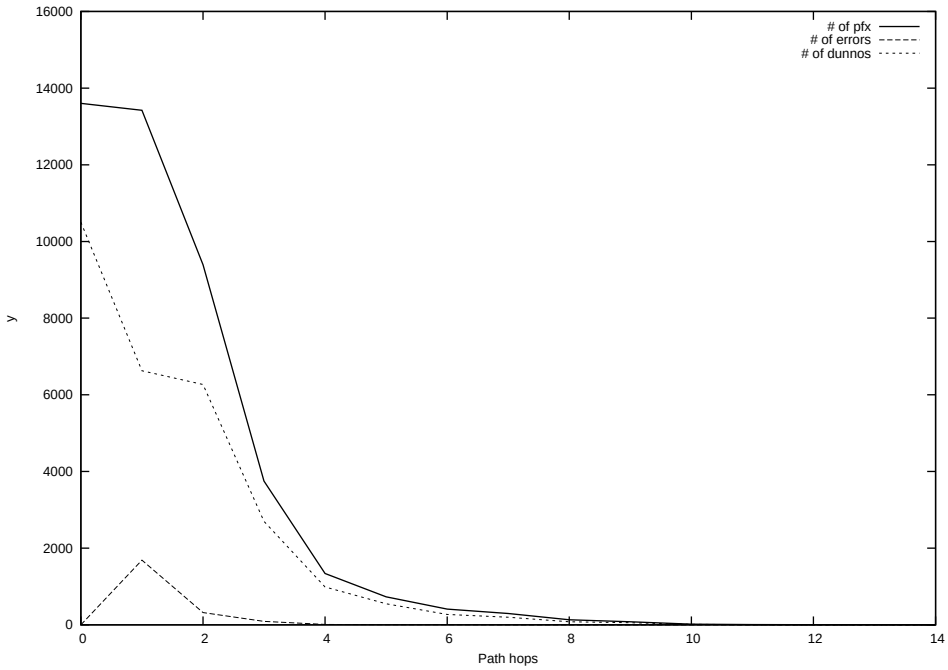
2013-07-01



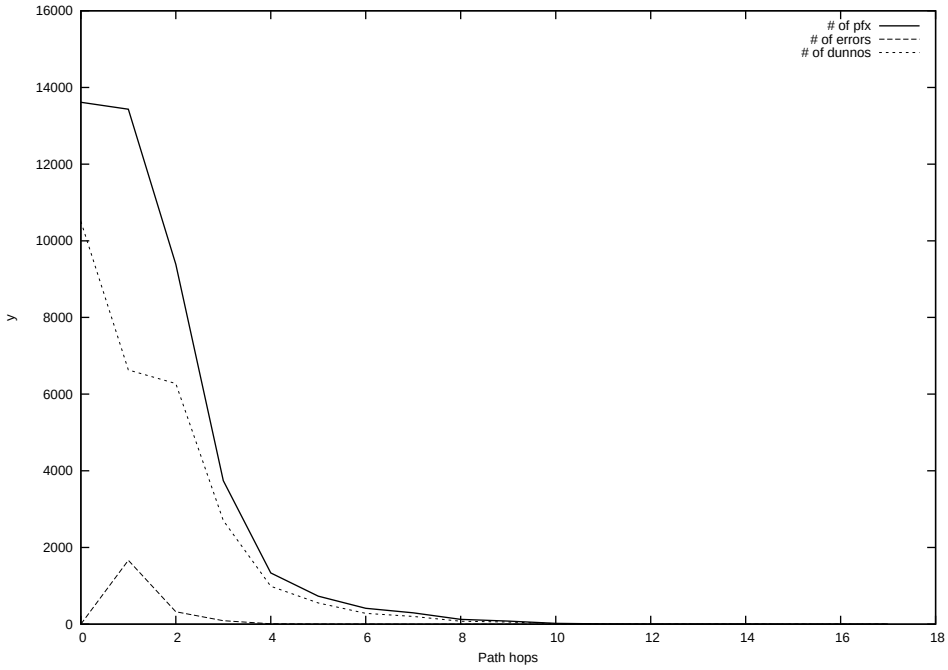
2013-07-02



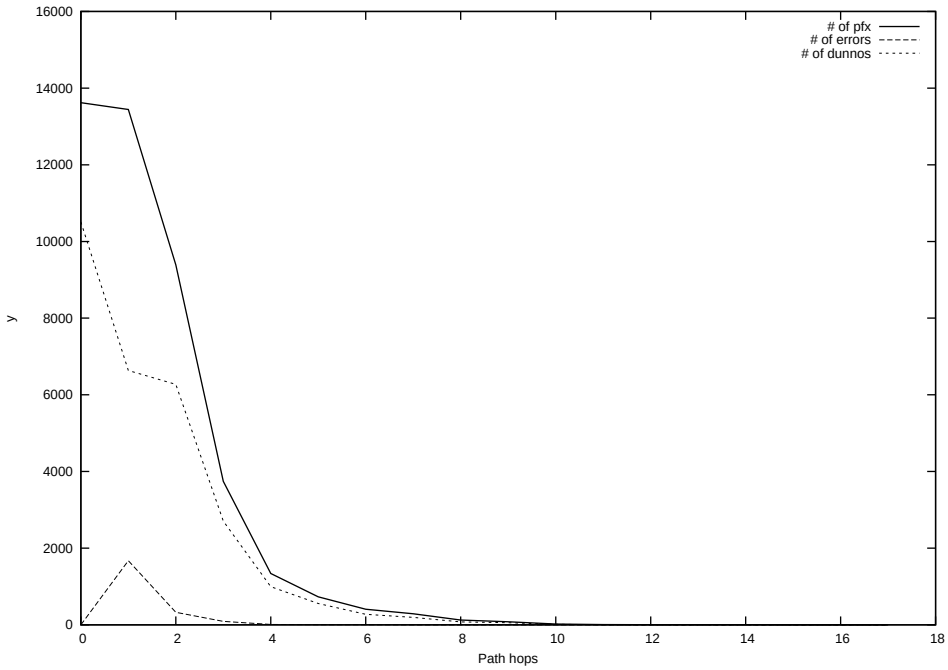
2013-07-03



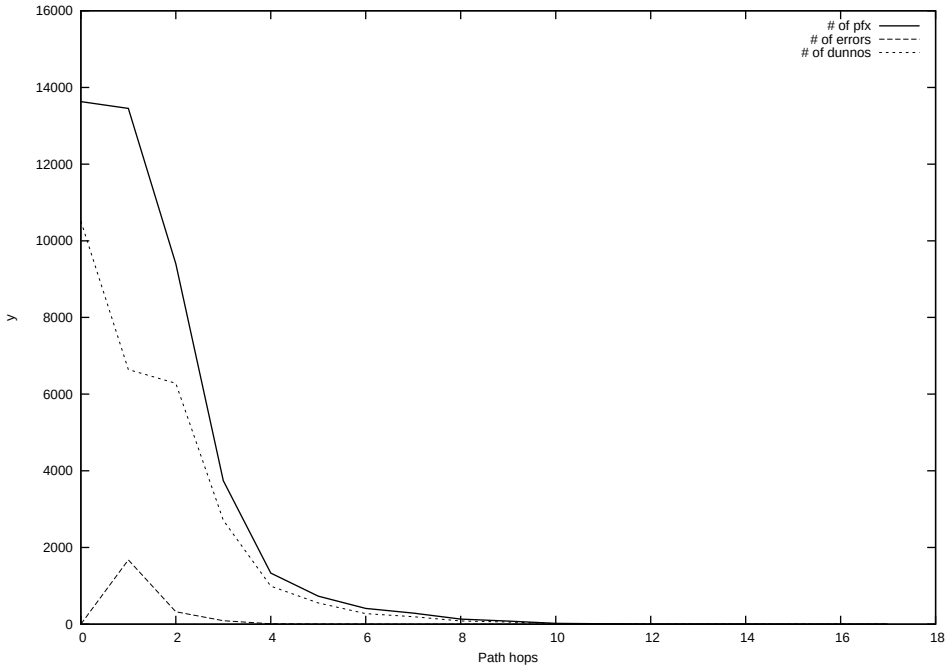
2013-07-04



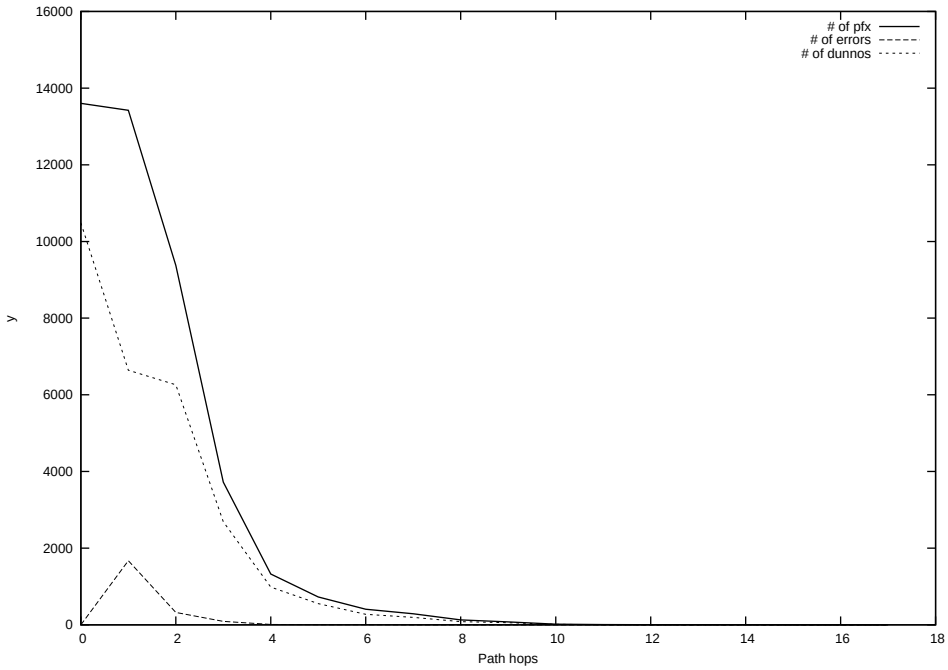
2013-07-05



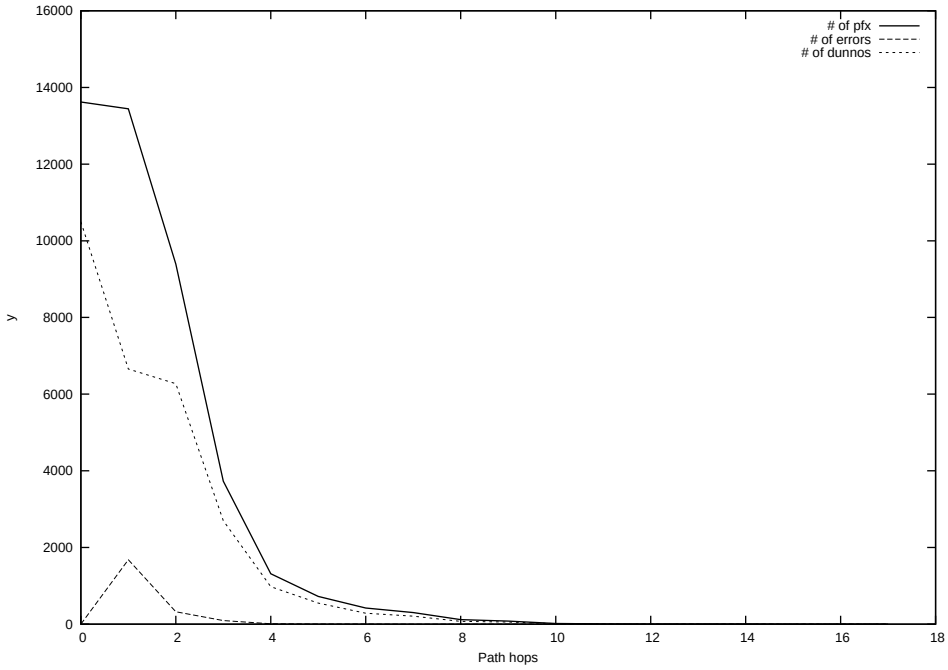
2013-07-06



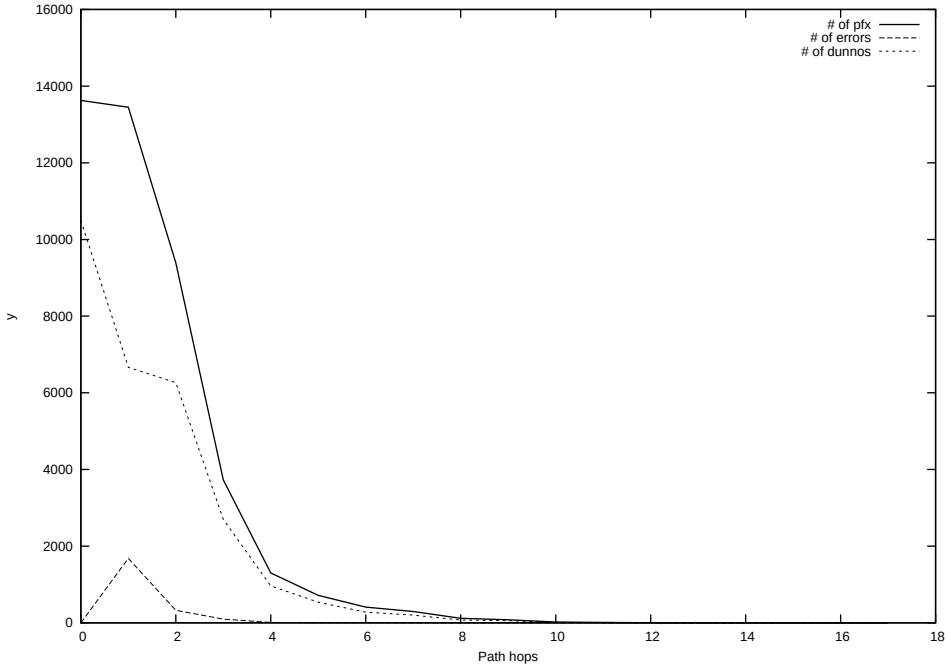
2013-07-07



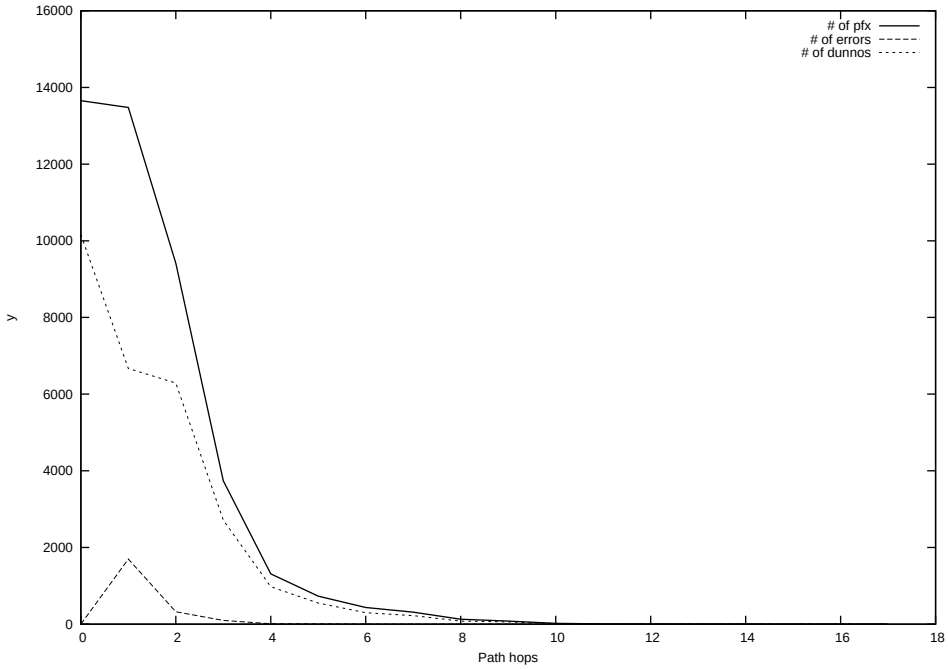
2013-07-08



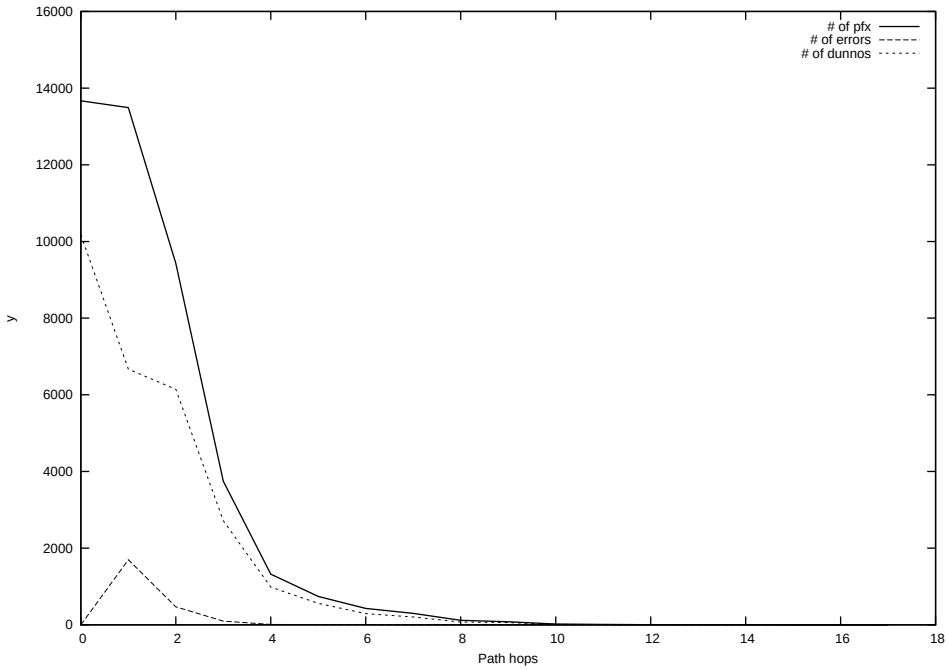
2013-07-09



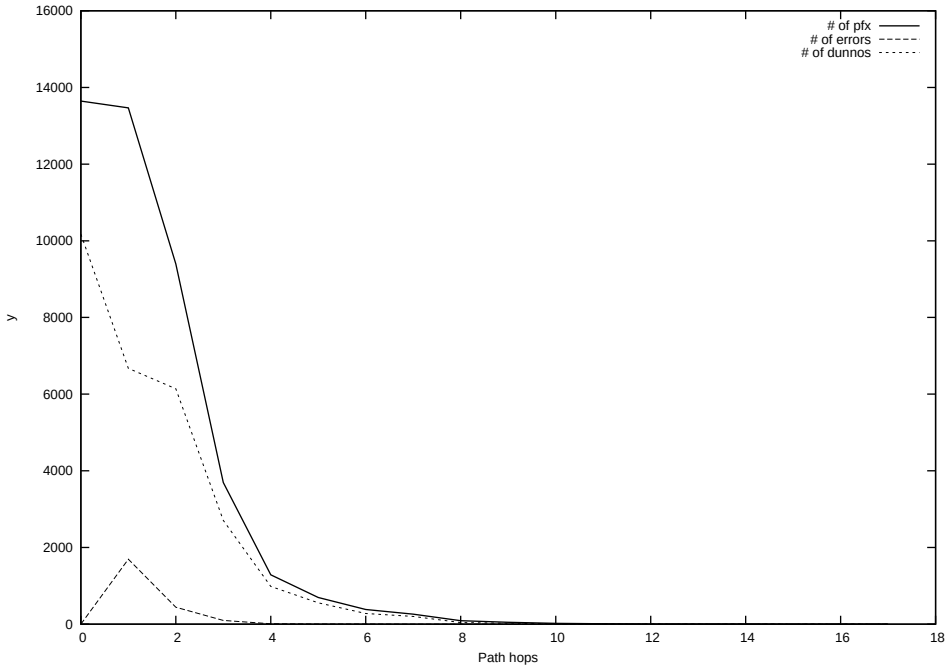
2013-07-10



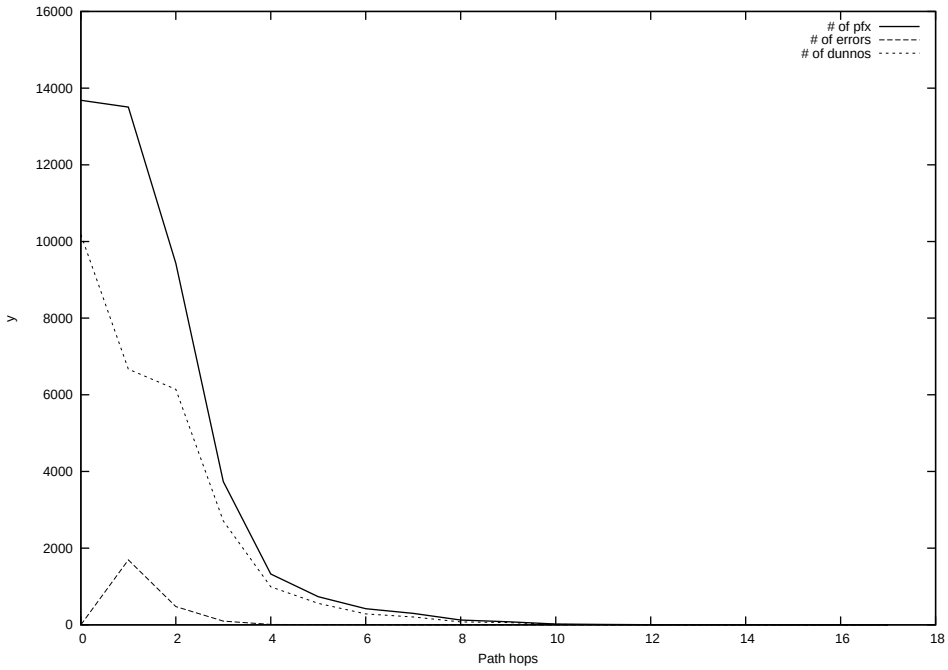
2013-07-11



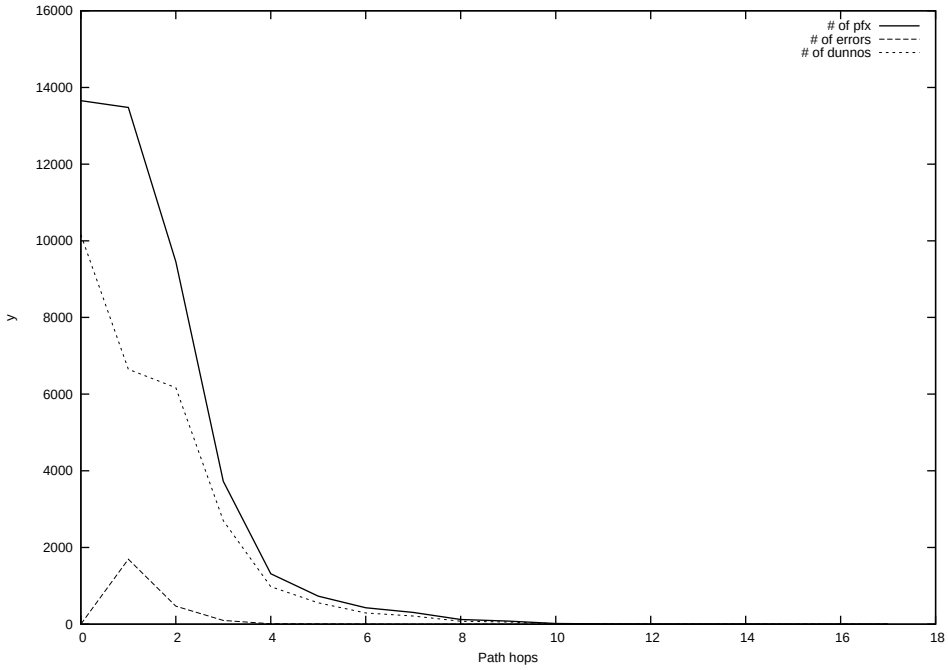
2013-07-12



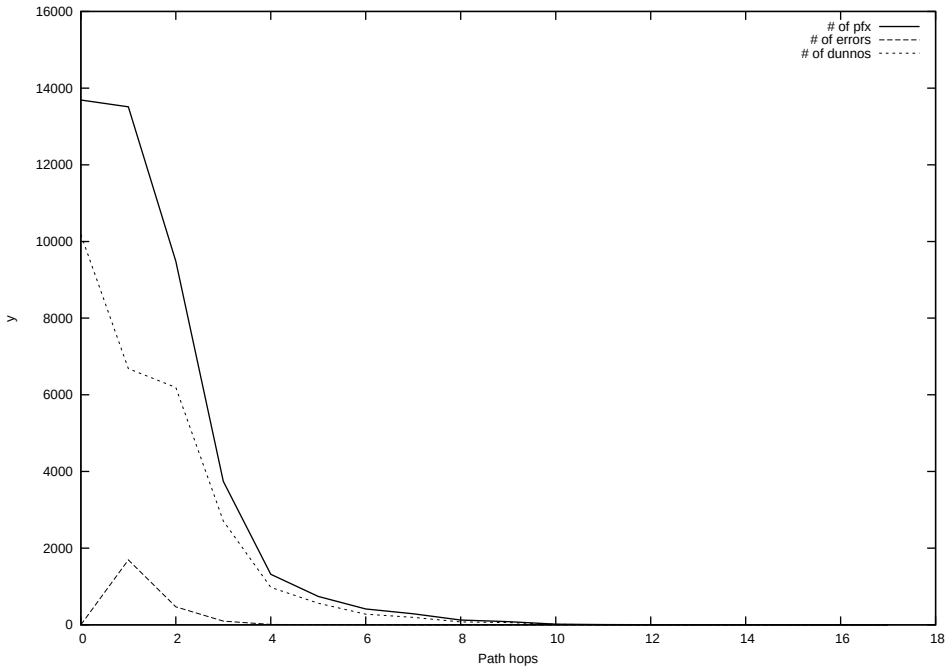
2013-07-13



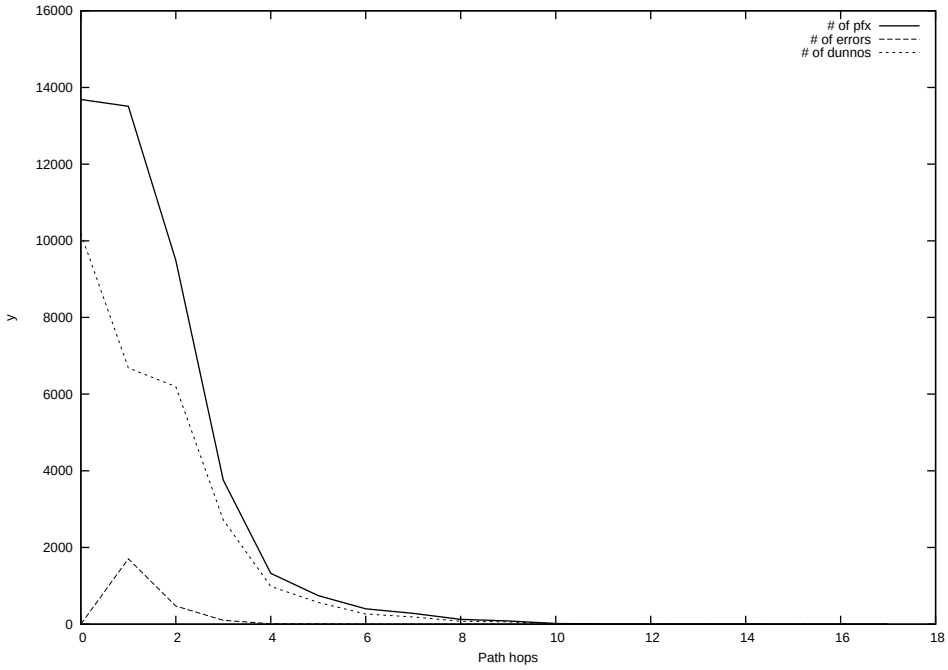
2013-07-14



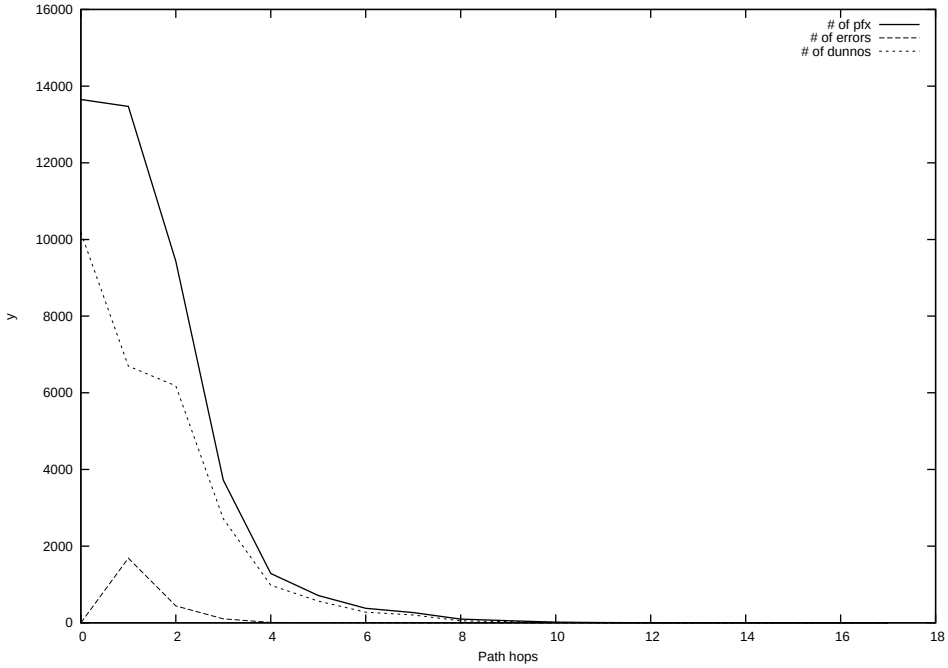
2013-07-15



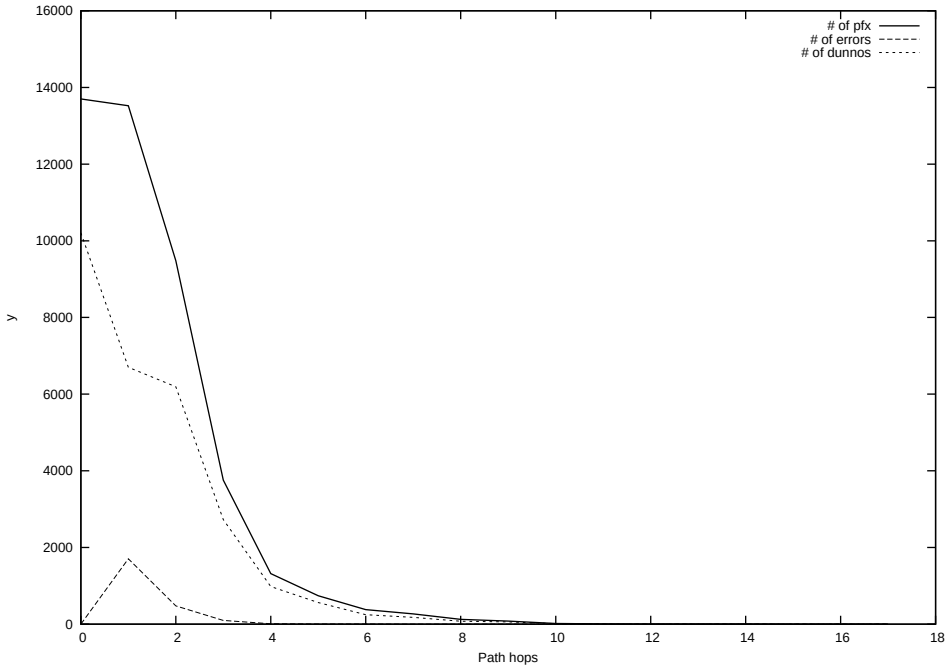
2013-07-16



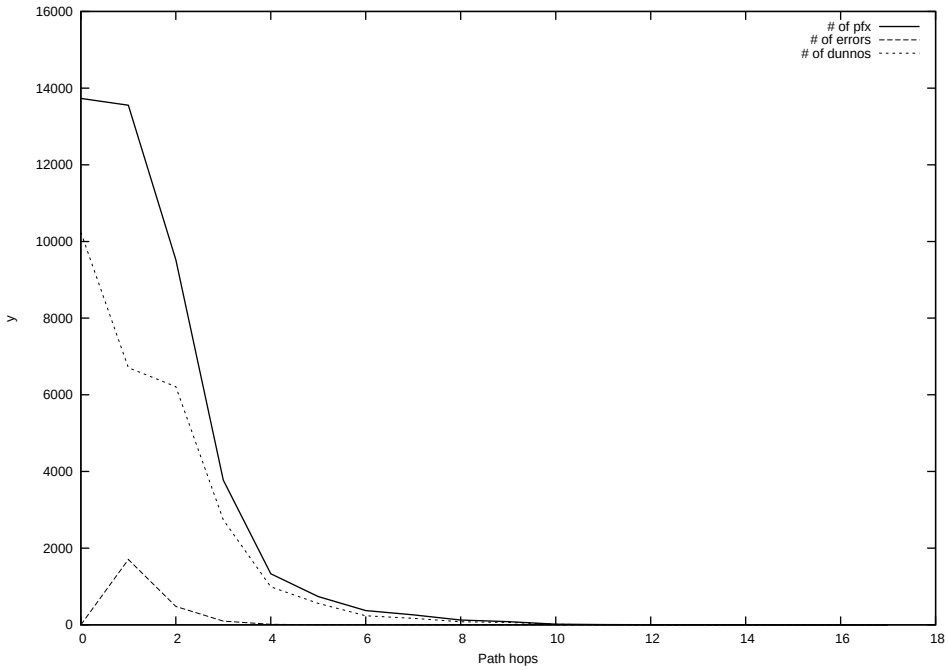
2013-07-17



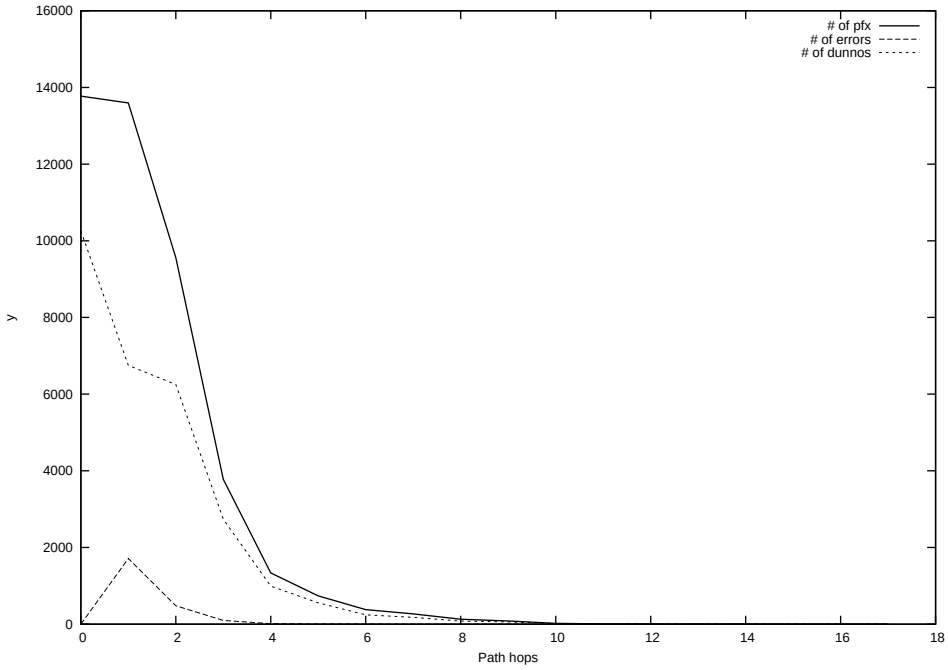
2013-07-18



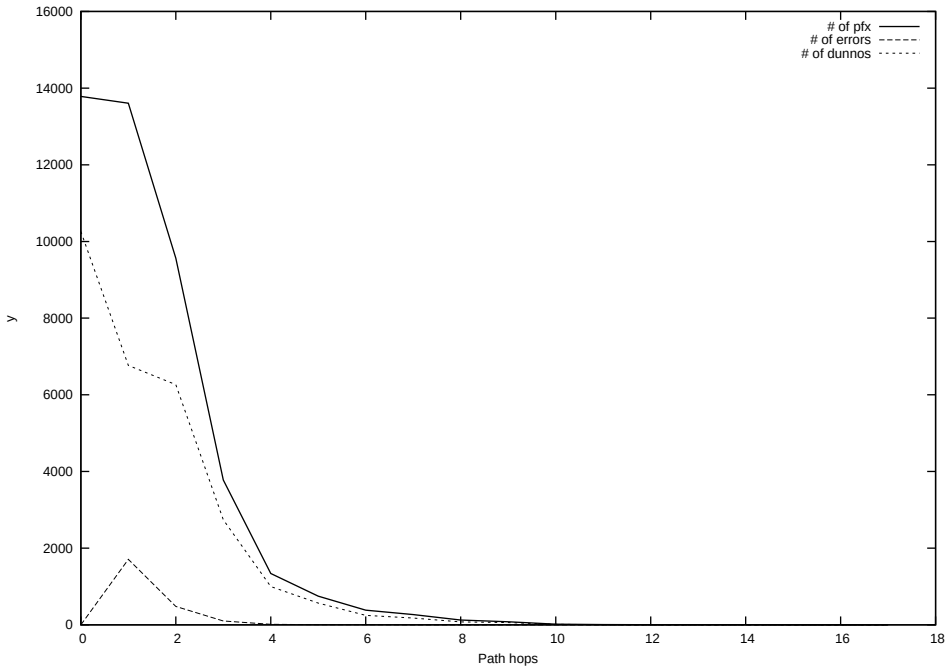
2013-07-19



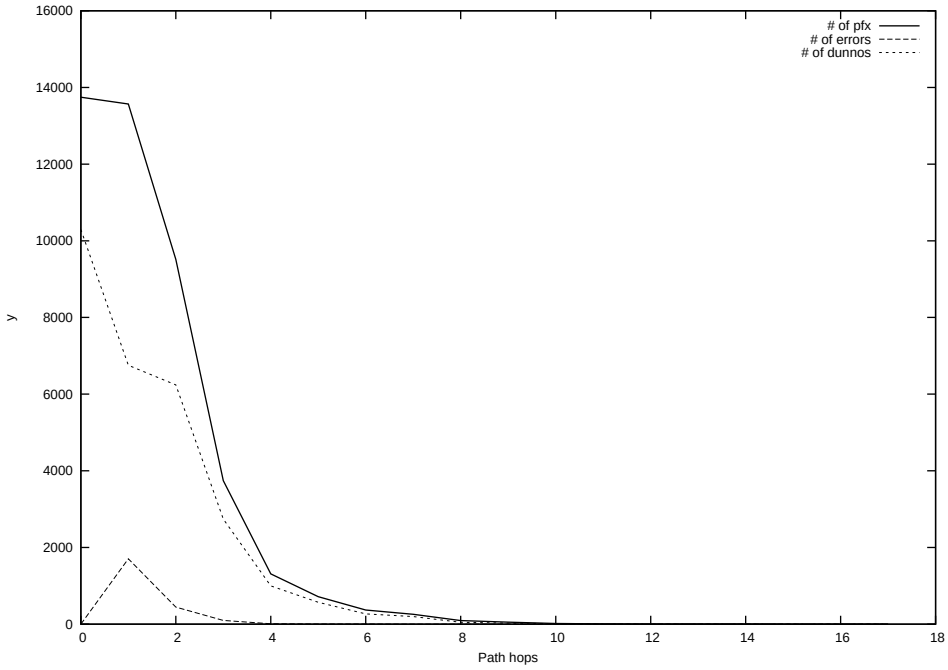
2013-07-20



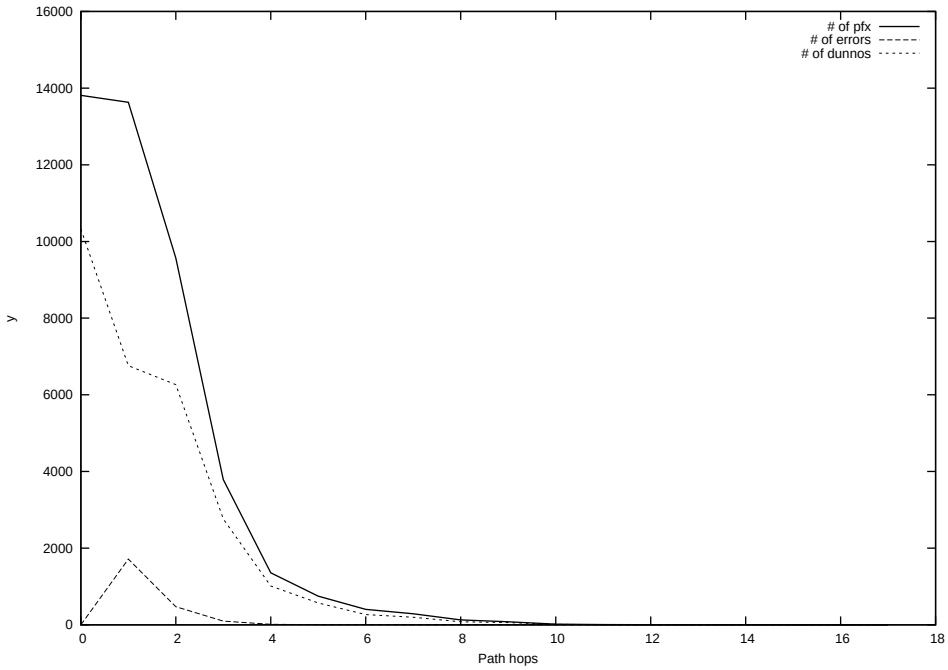
2013-07-21



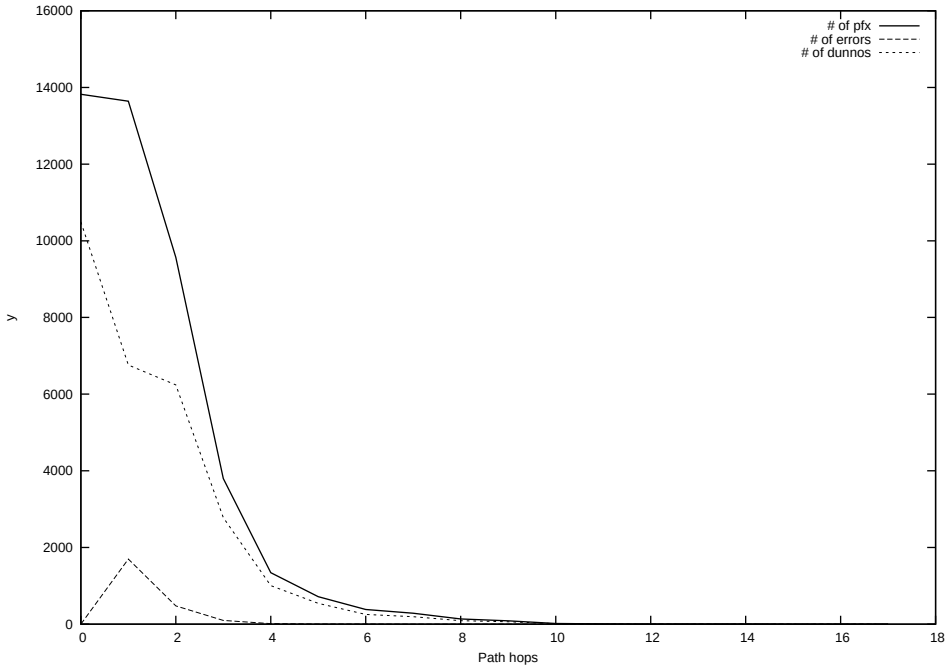
2013-07-22



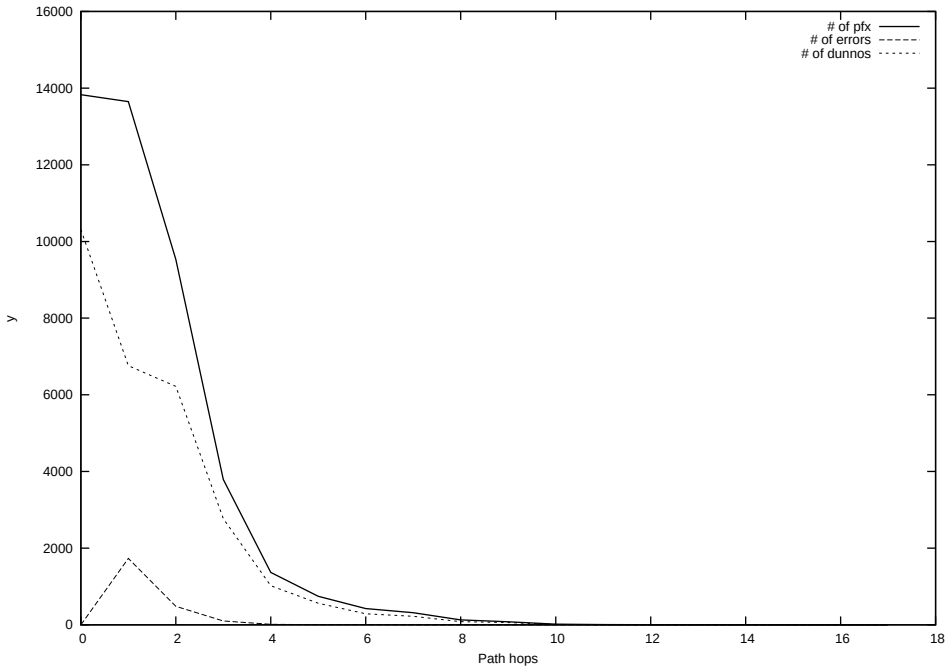
2013-07-23



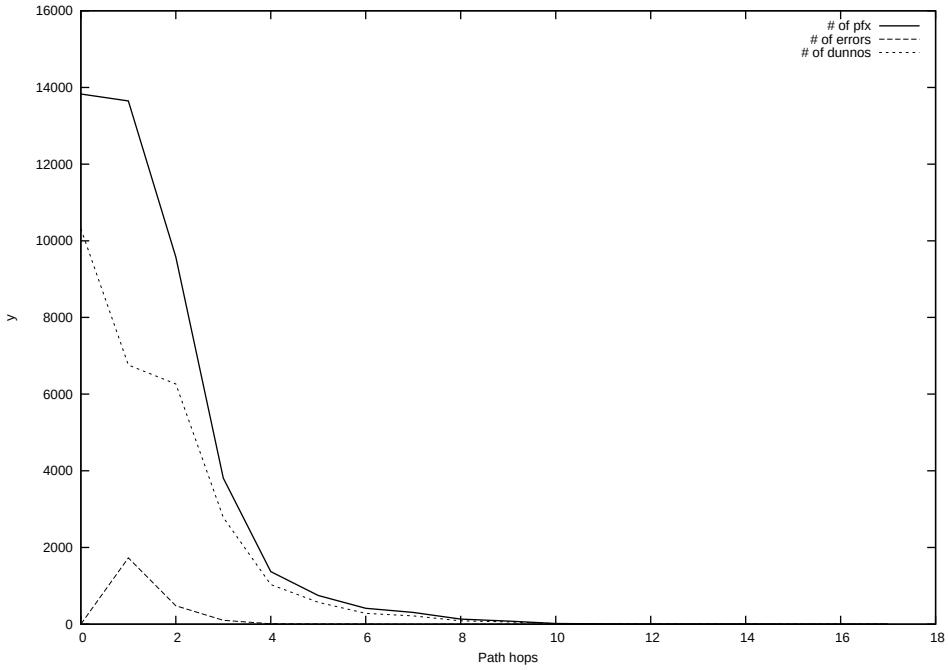
2013-07-24



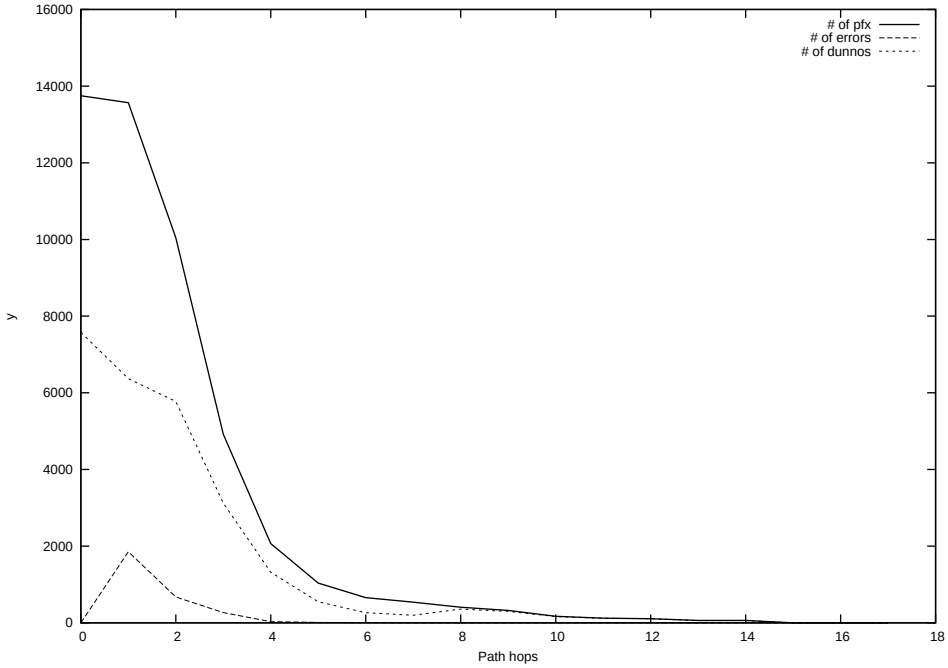
2013-07-25



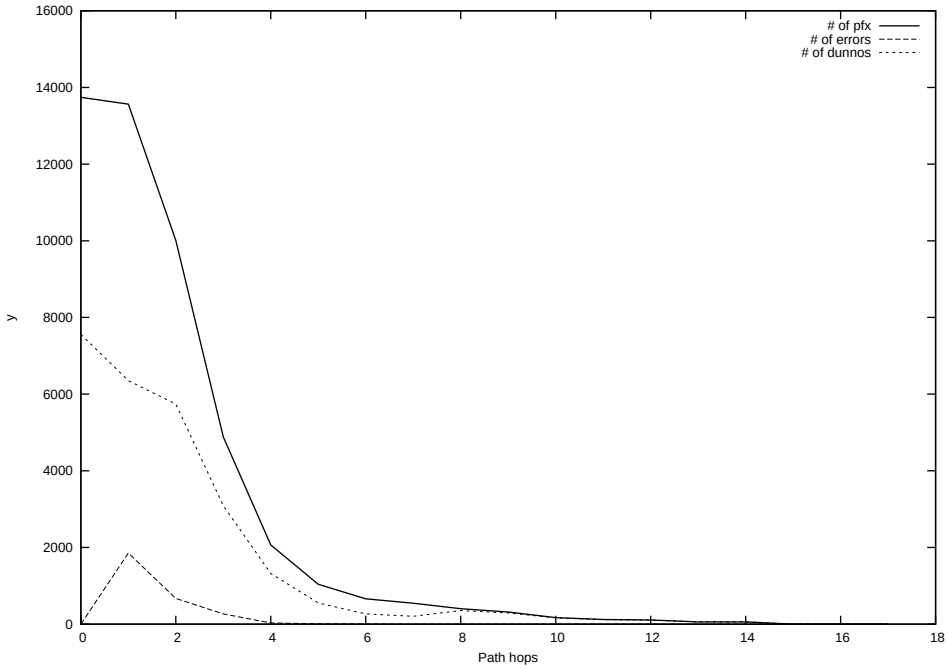
2013-07-26



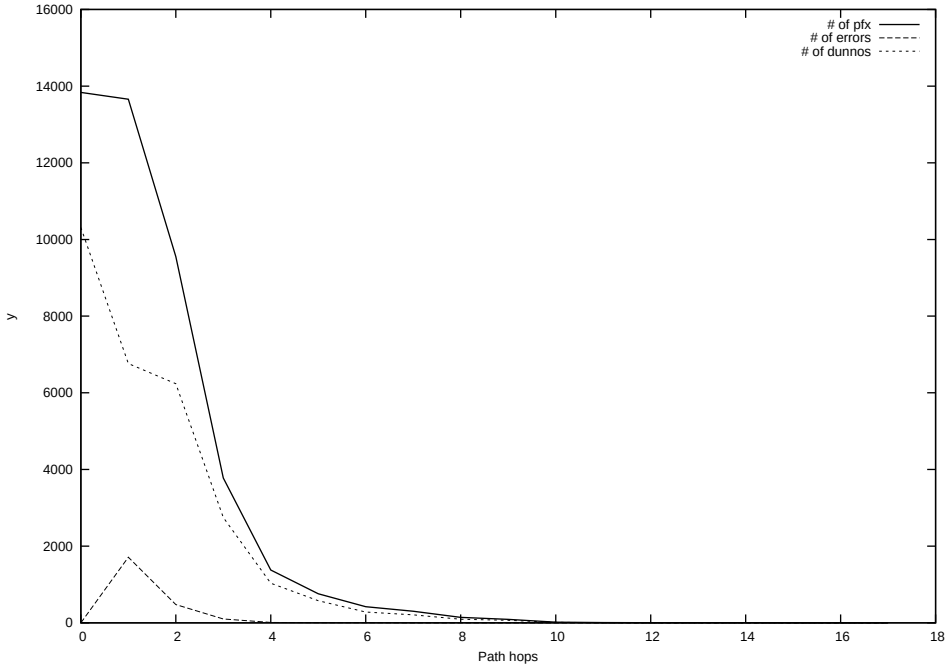
2013-07-27



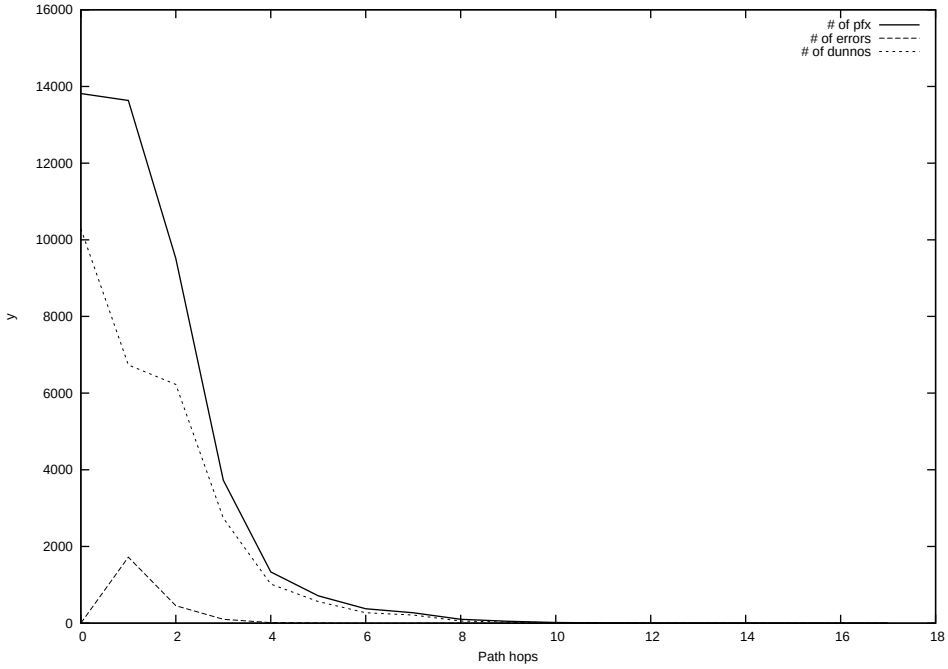
2013-07-28



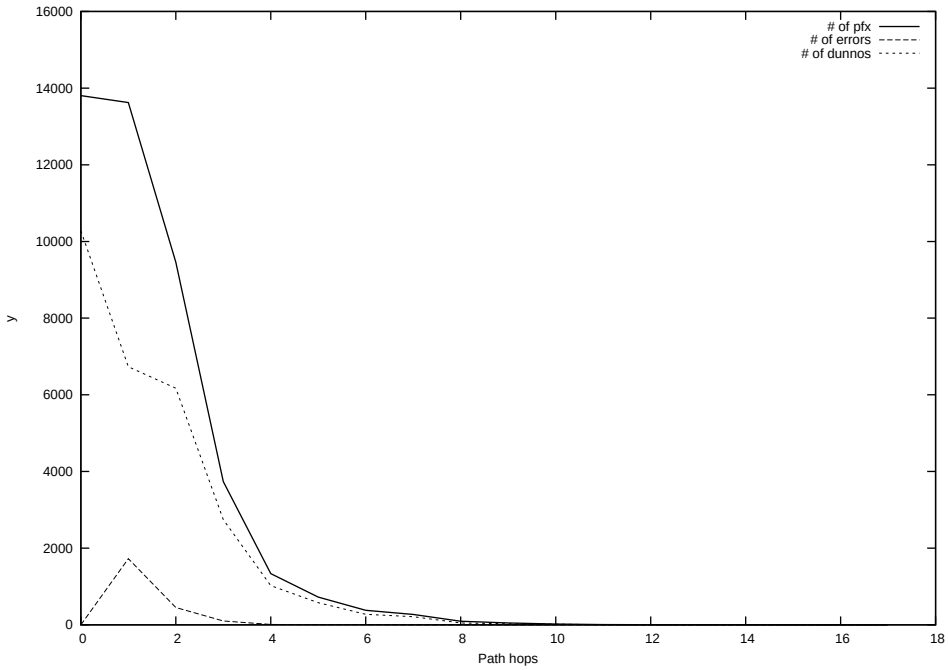
2013-07-29



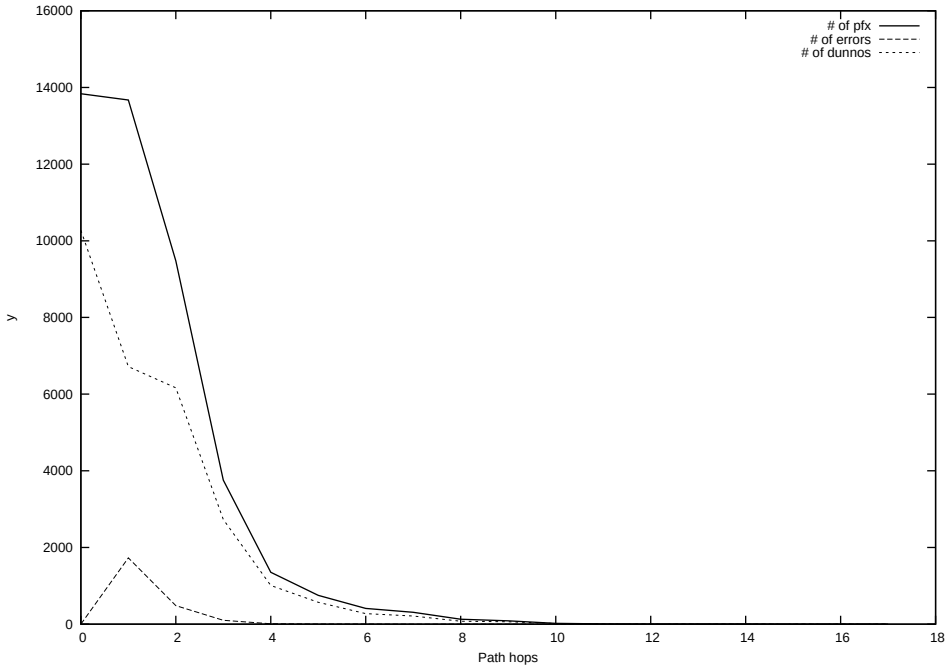
2013-07-30



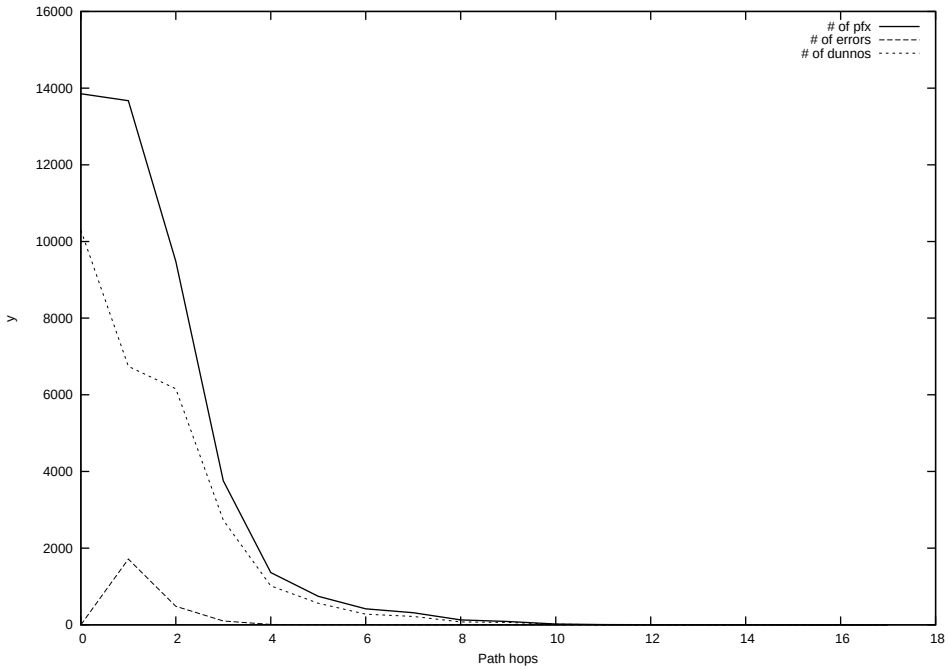
2013-07-31



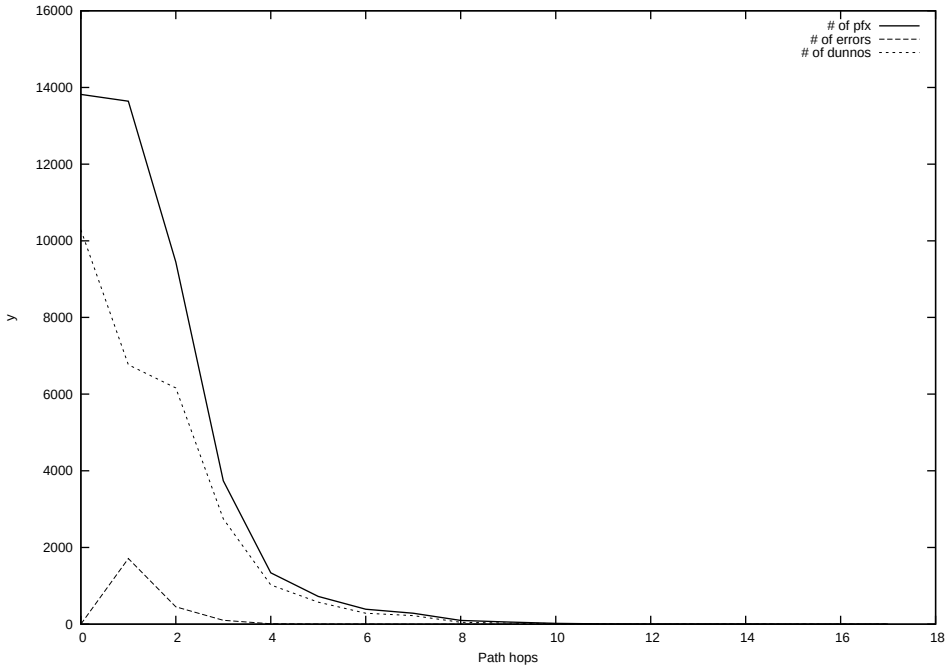
2013-08-01



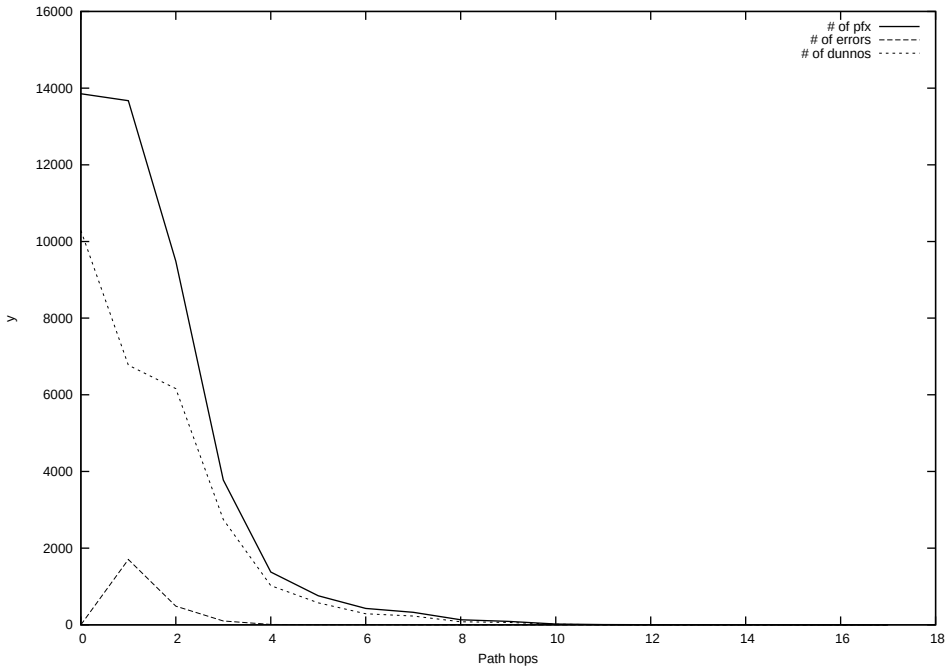
2013-08-02



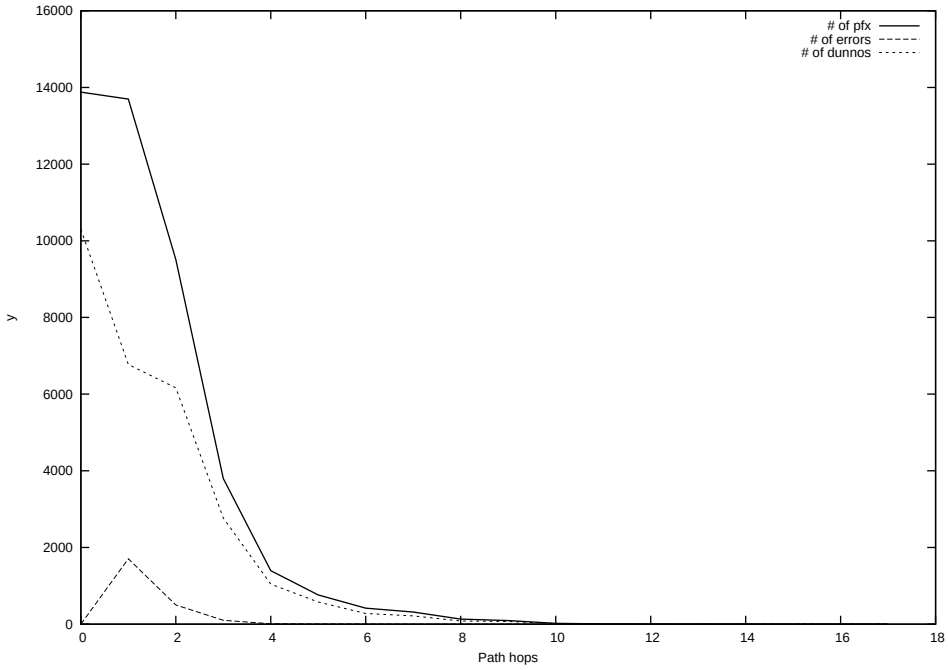
2013-08-03



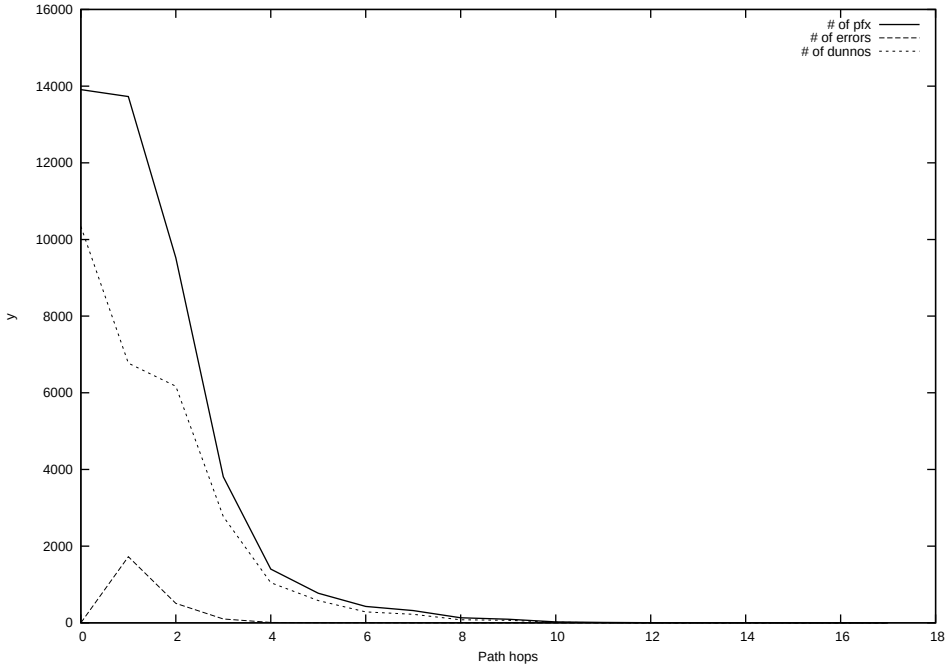
2013-08-04



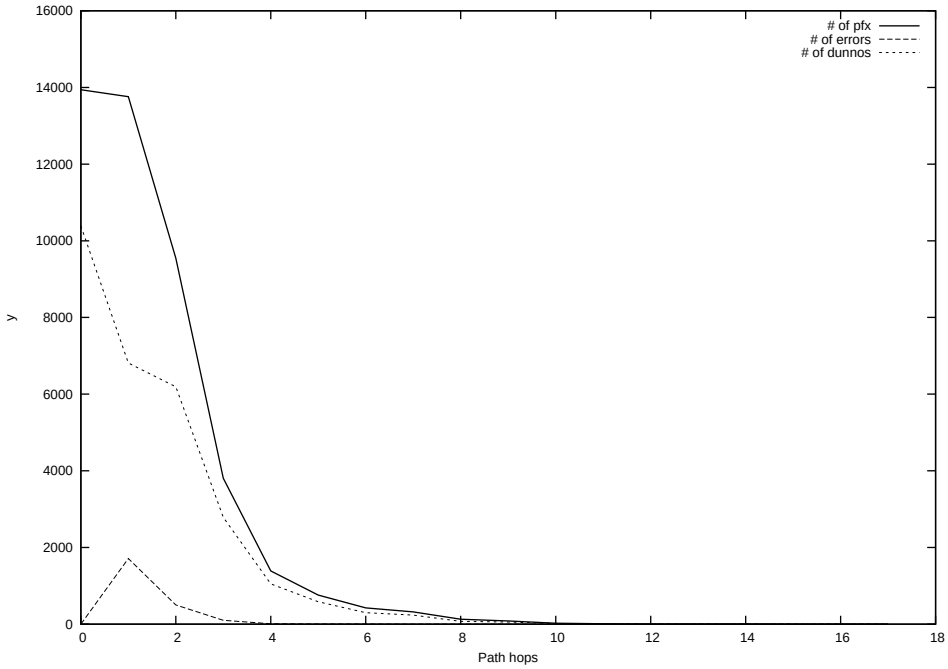
2013-08-05



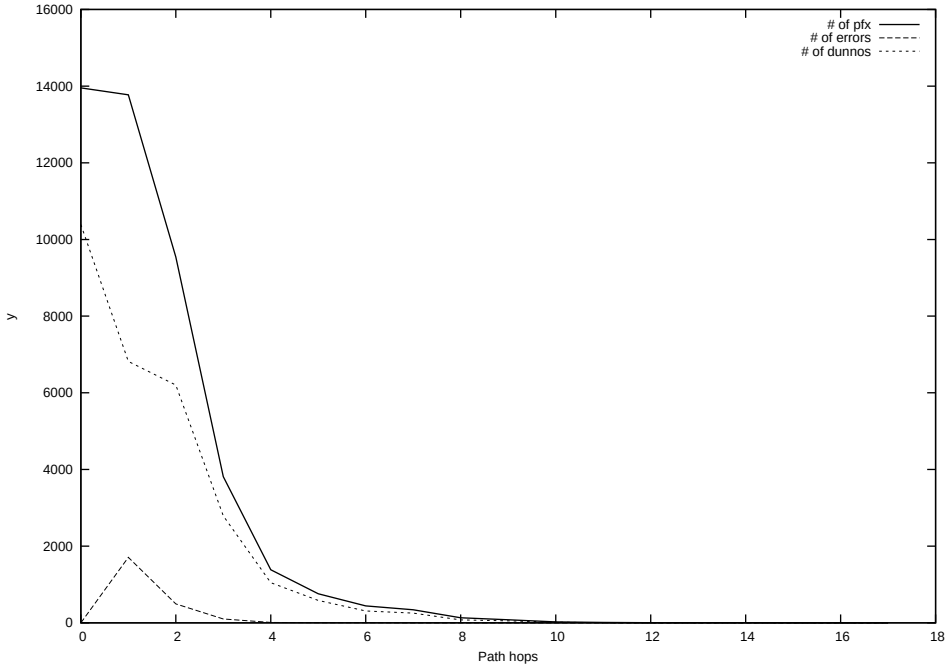
2013-08-06



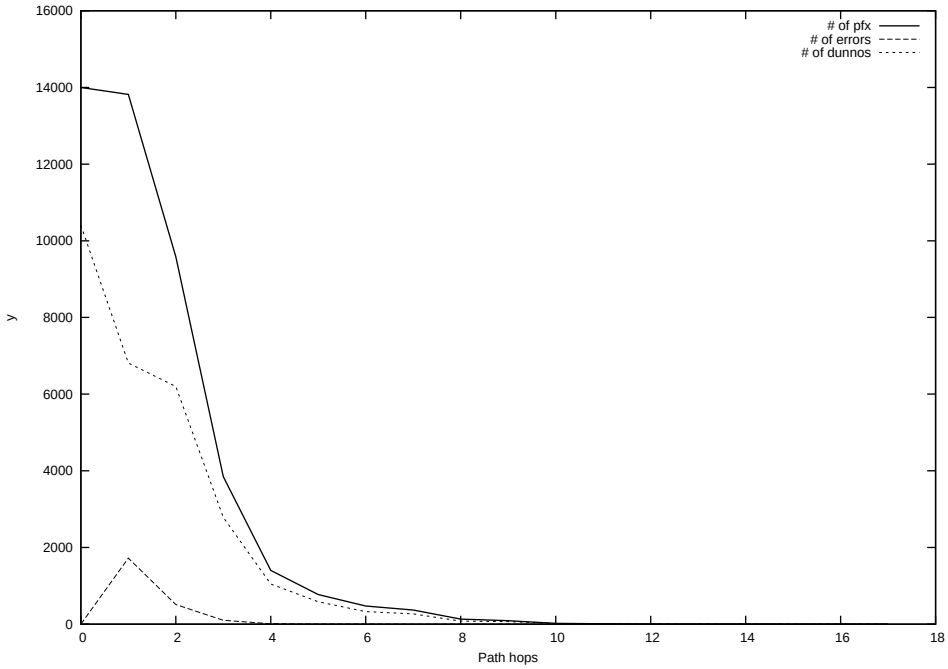
2013-08-07



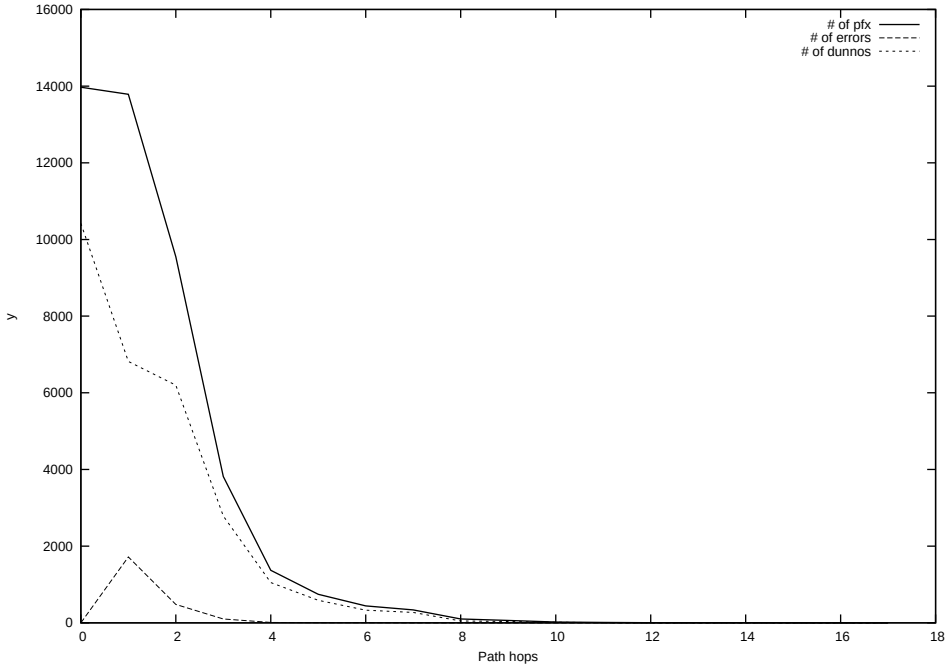
2013-08-08



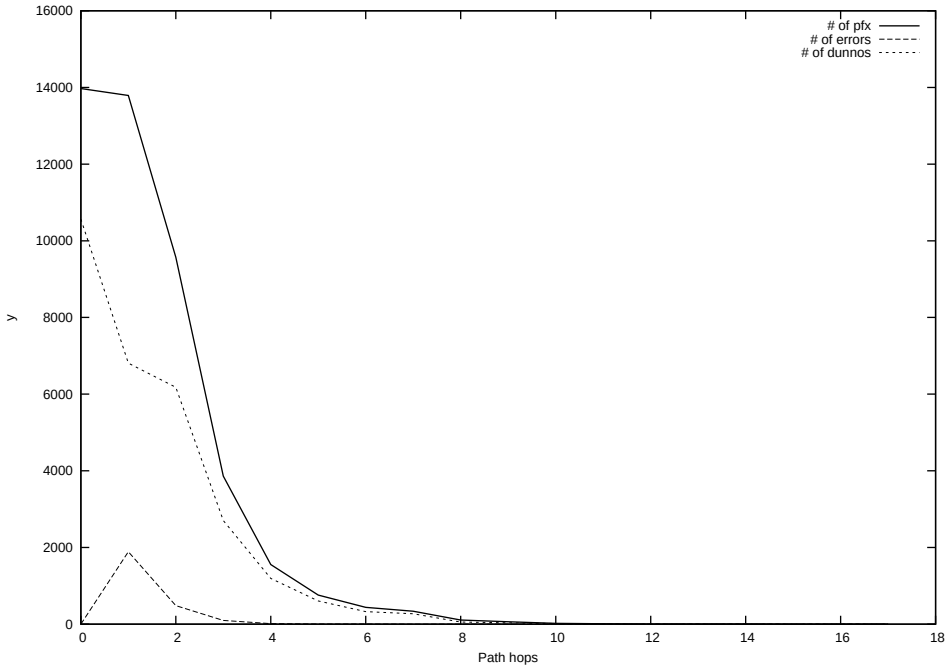
2013-08-09



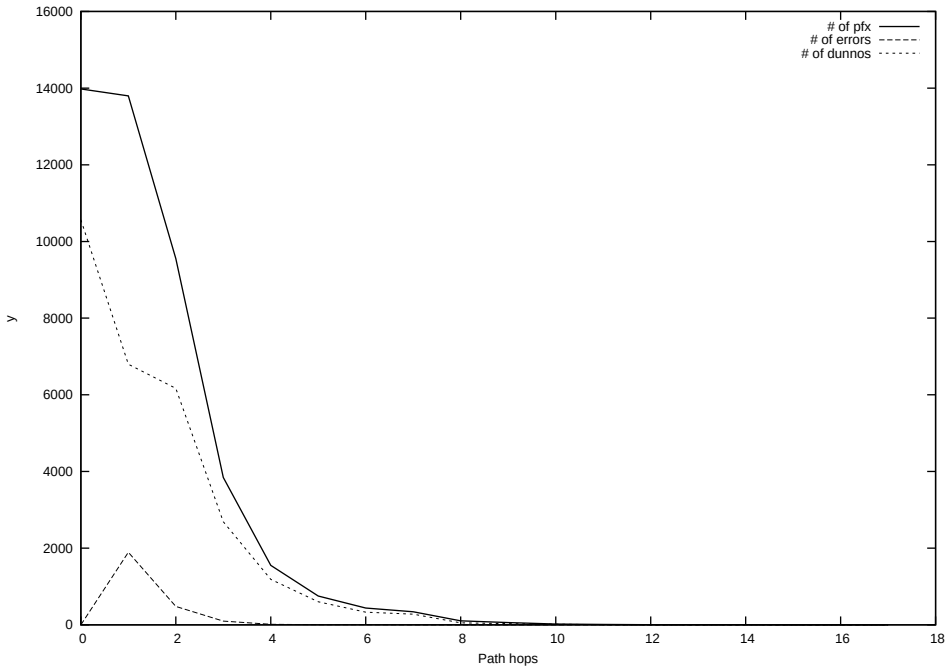
2013-08-10



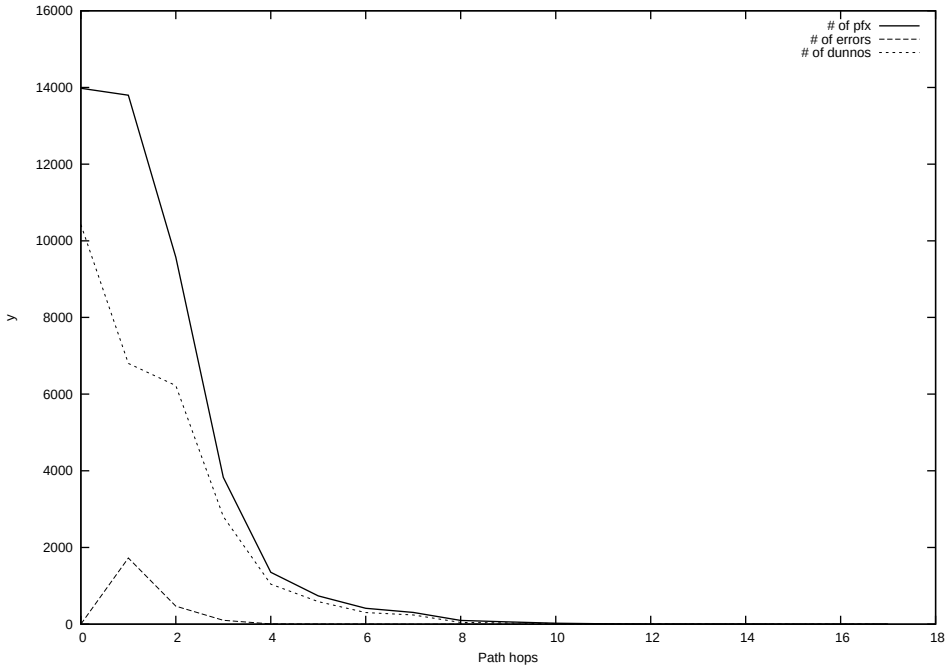
2013-08-11



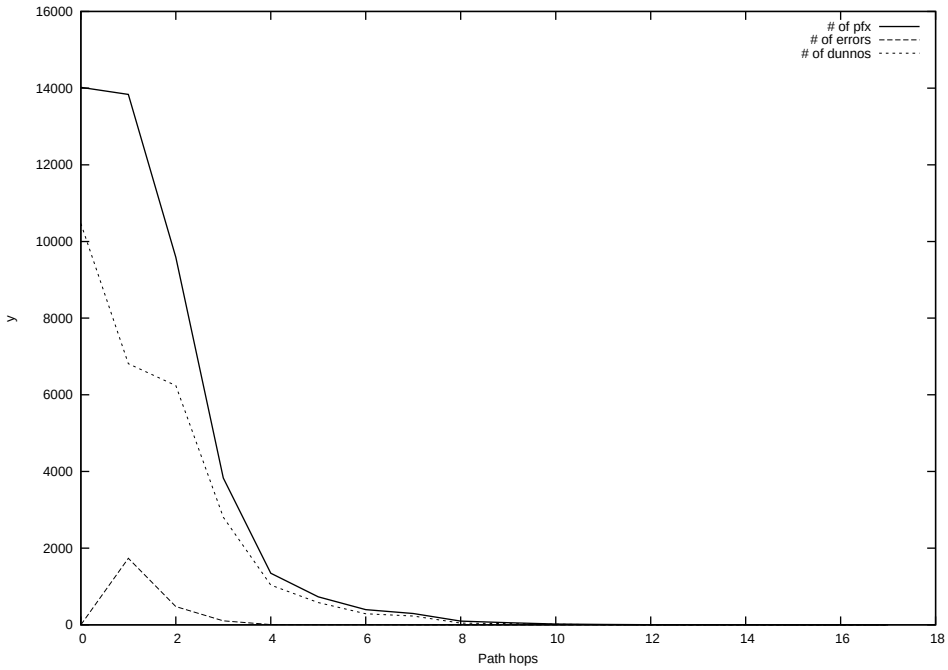
2013-08-12



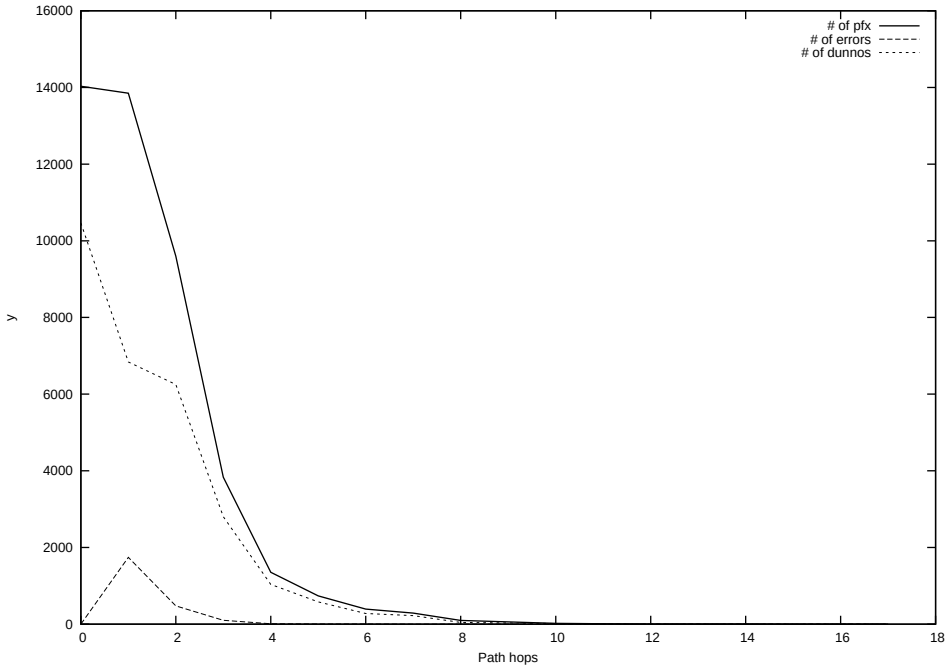
2013-08-13



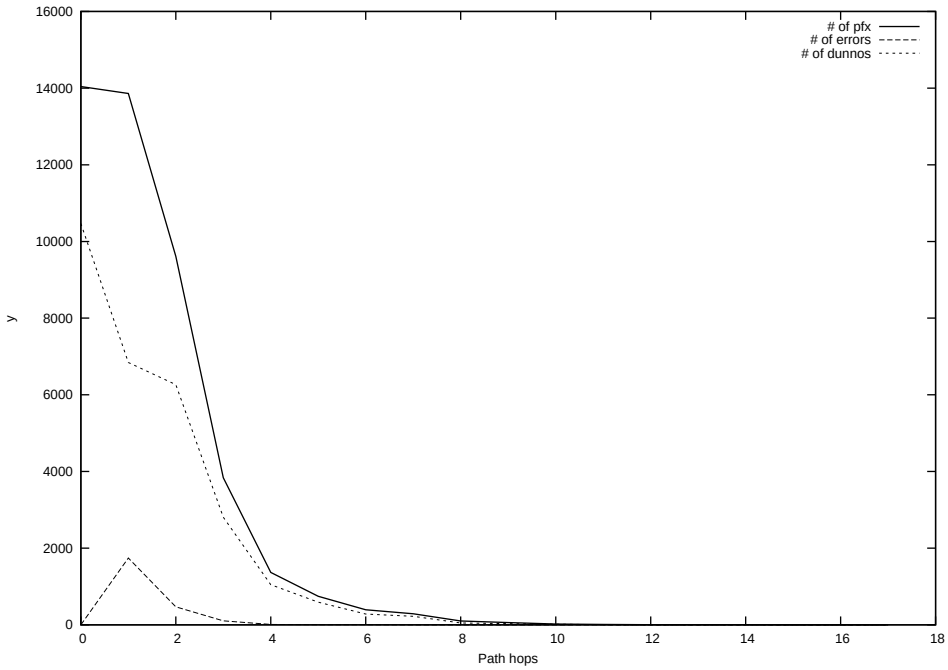
2013-08-15



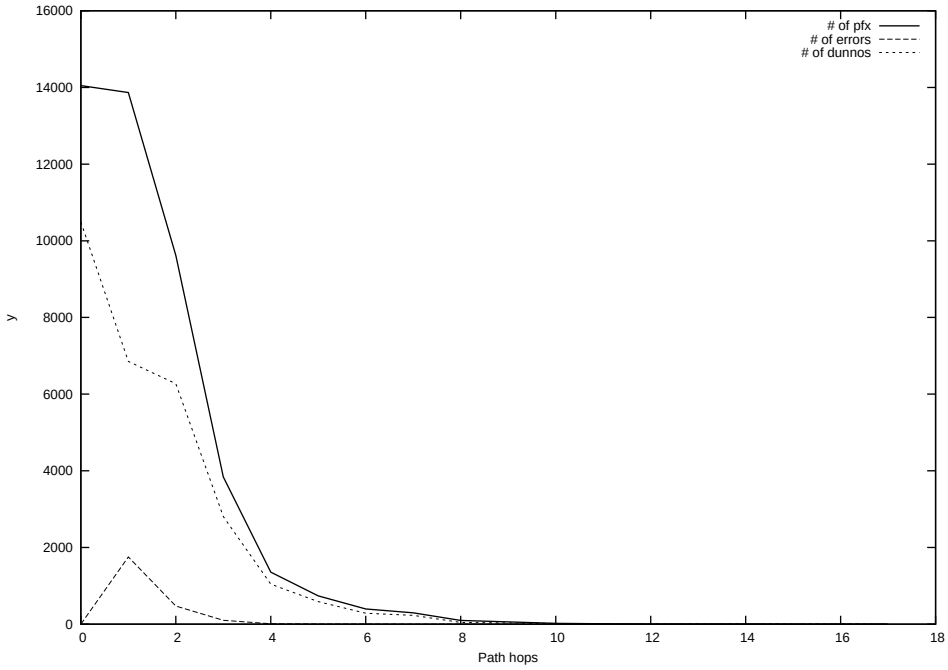
2013-08-16



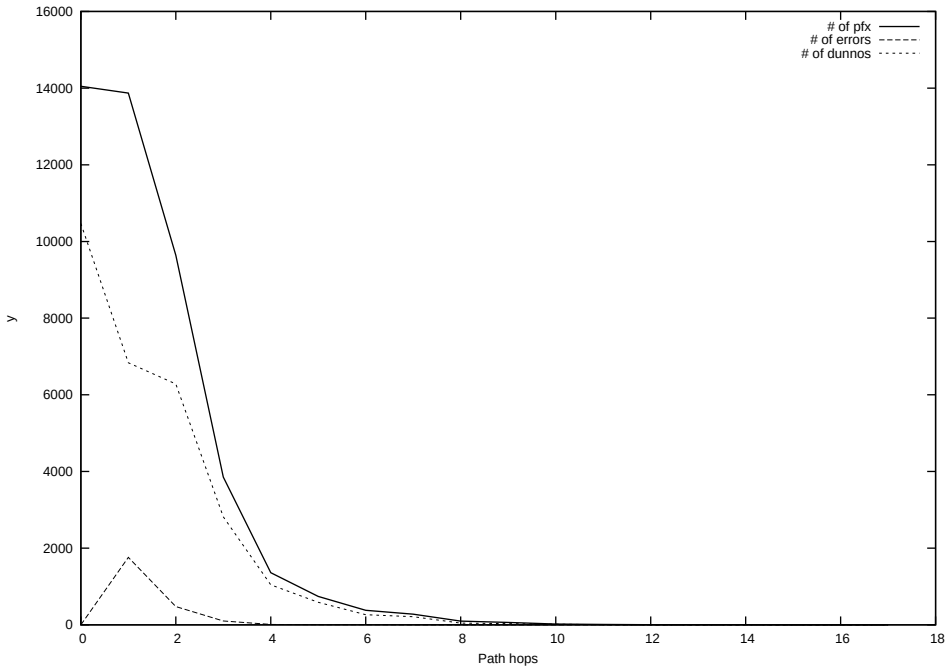
2013-08-17



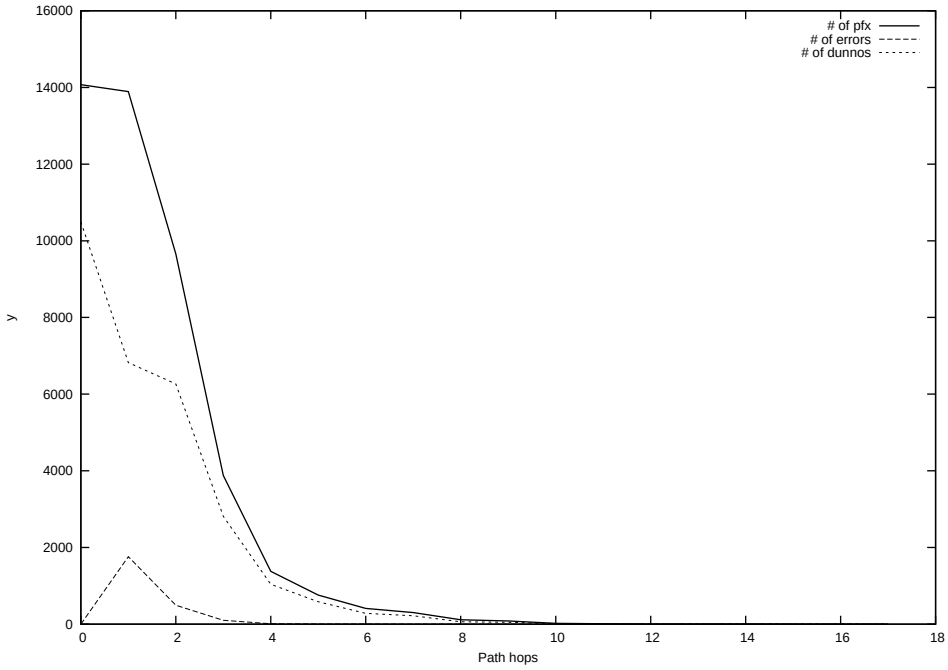
2013-08-18



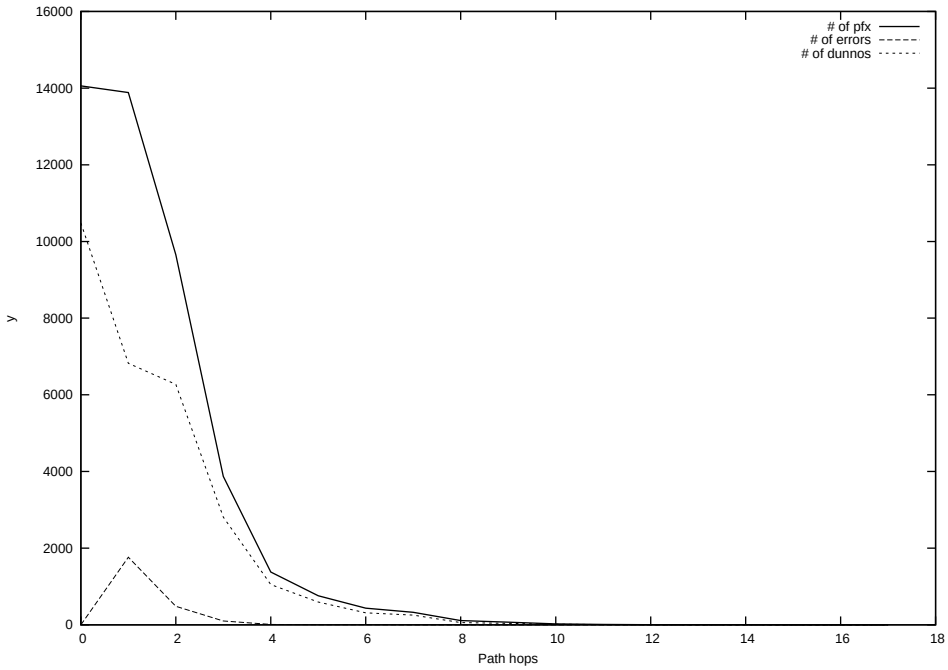
2013-08-19



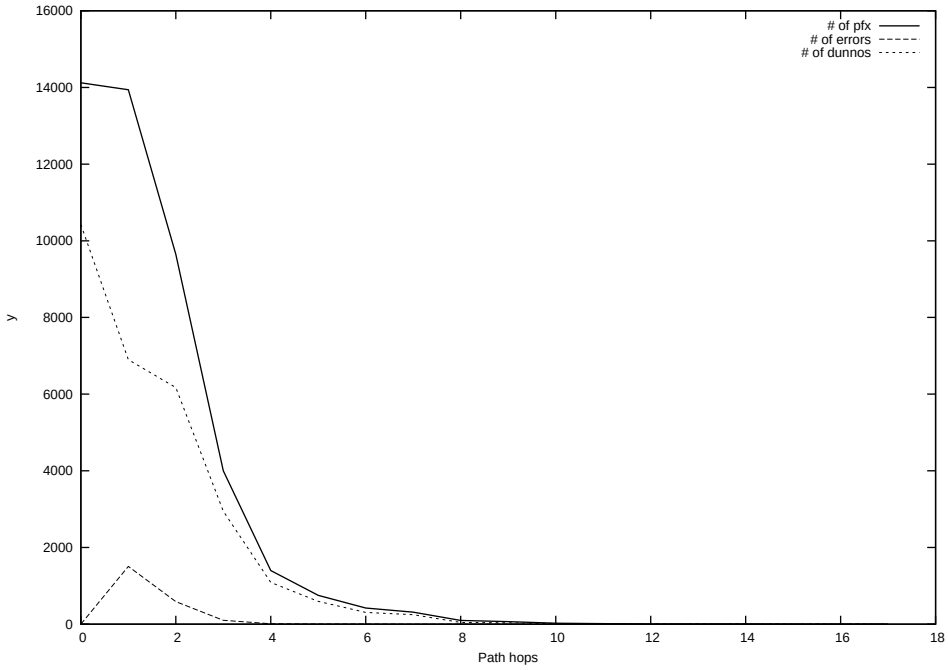
2013-08-20



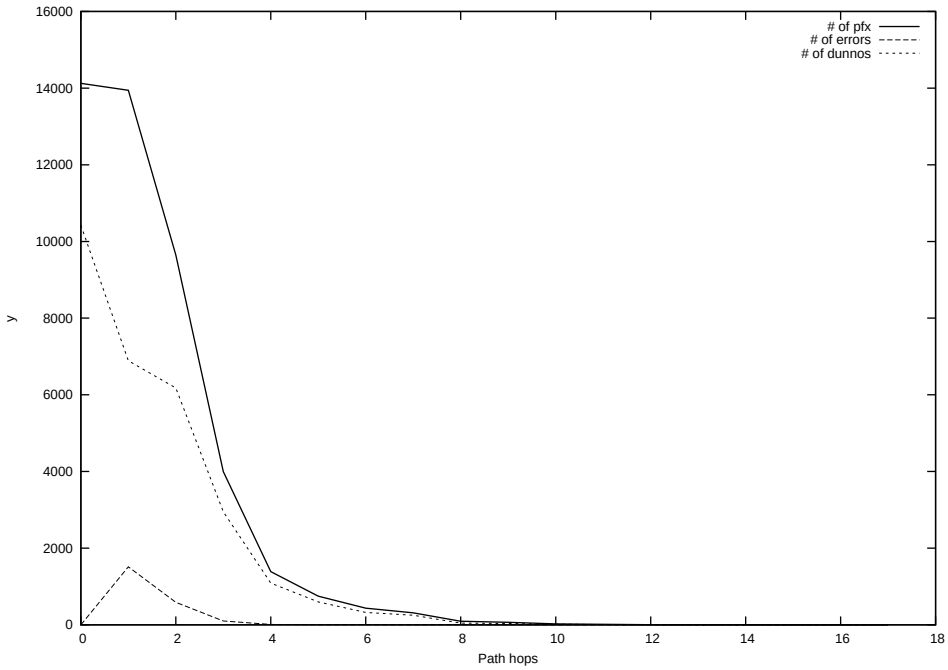
2013-08-21



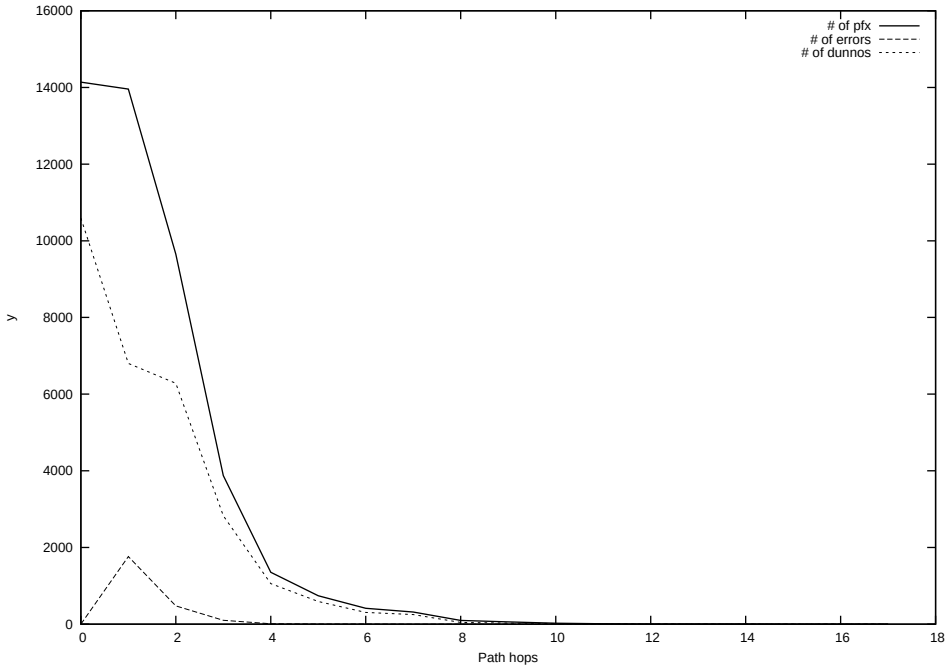
2013-08-22



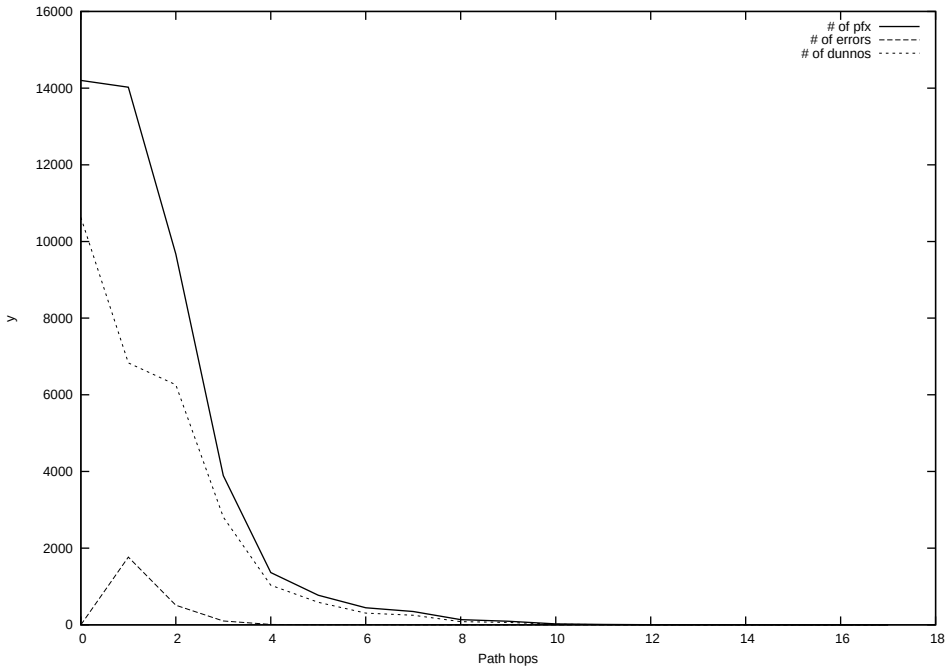
2013-08-23



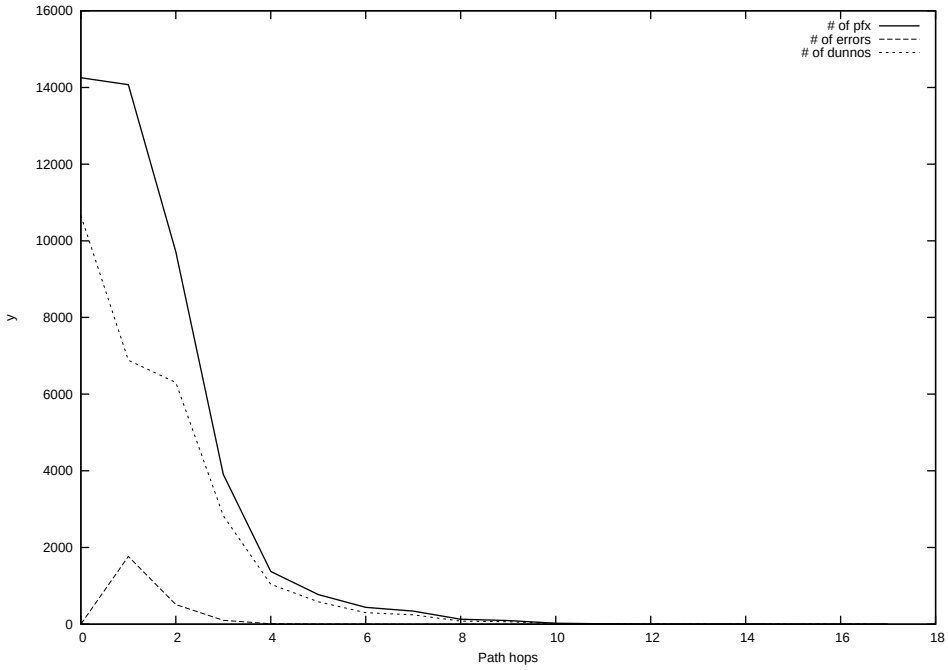
2013-08-24



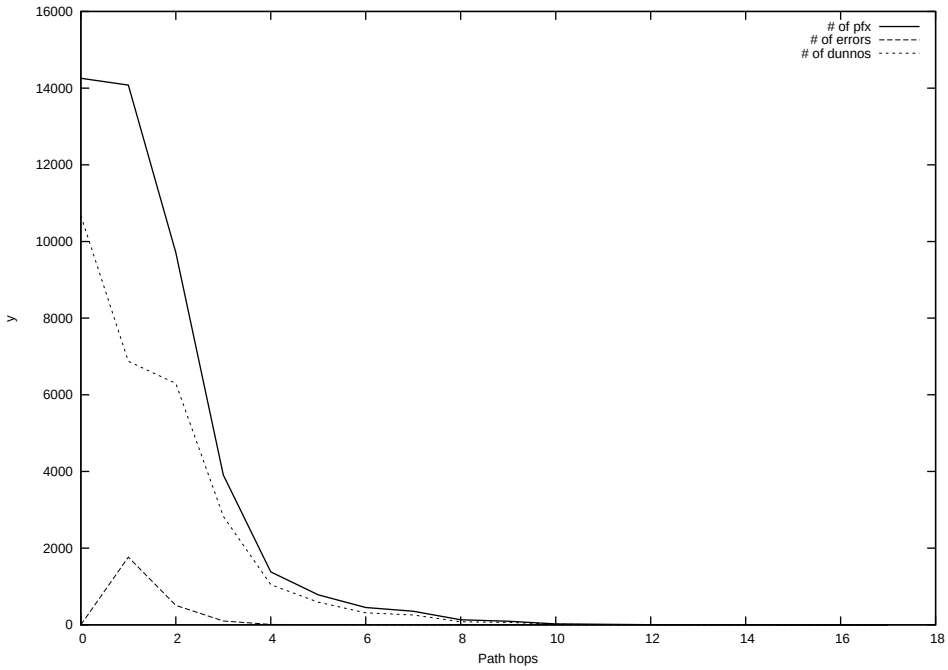
2013-08-25



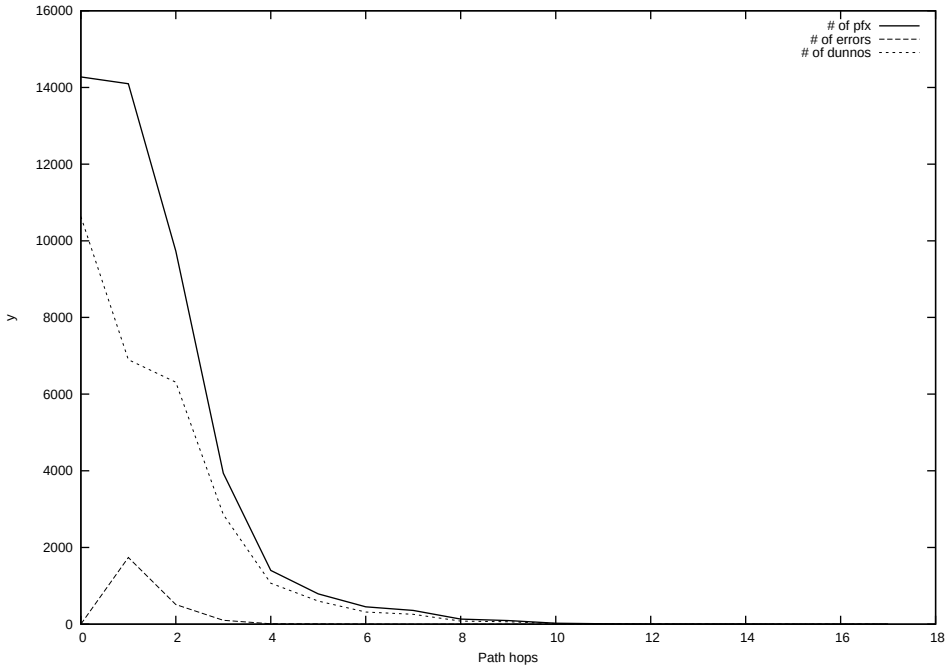
2013-08-26



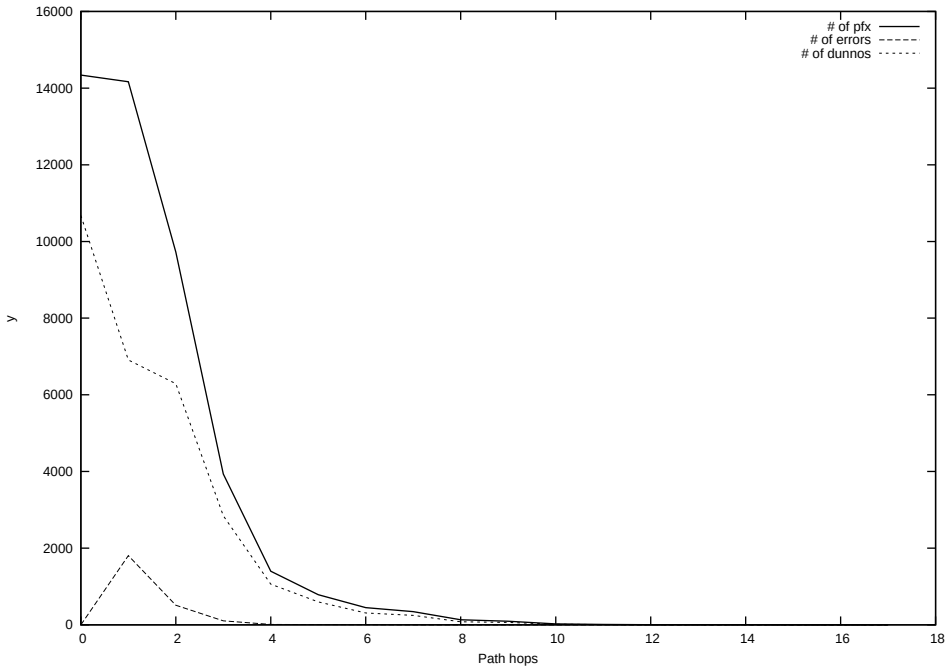
2013-08-27



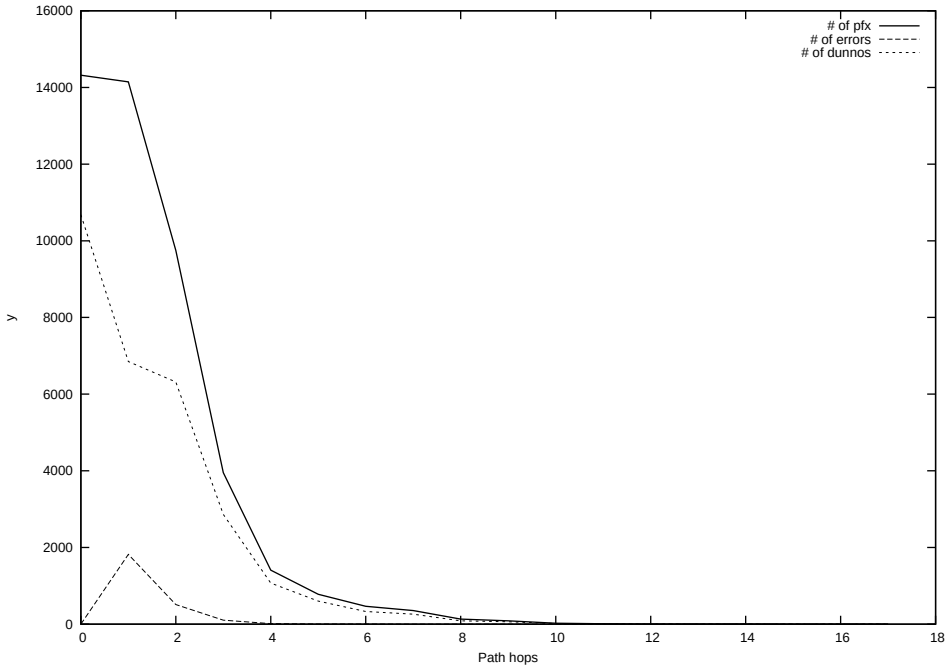
2013-08-28



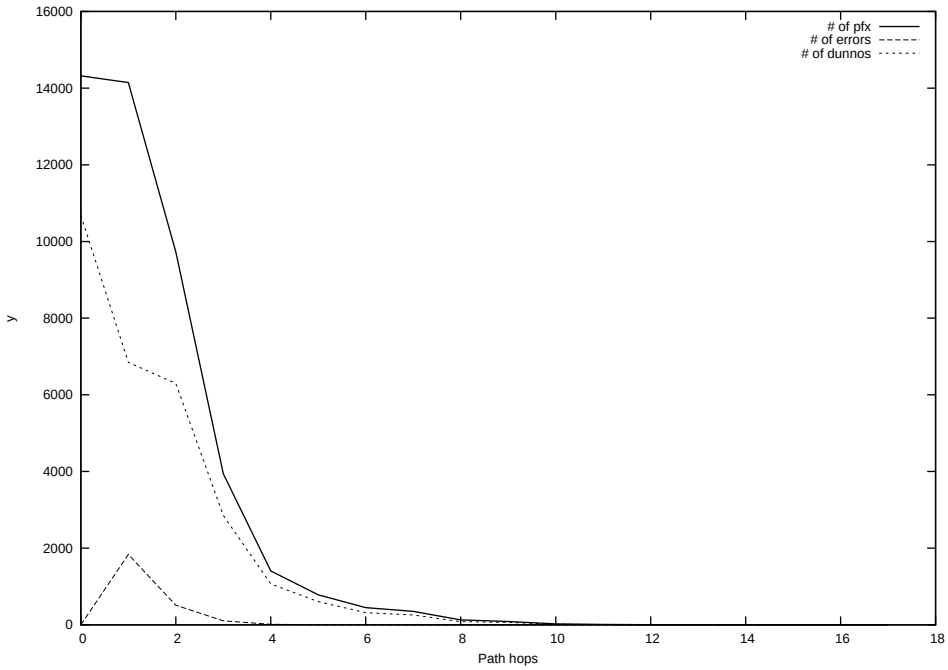
2013-08-29



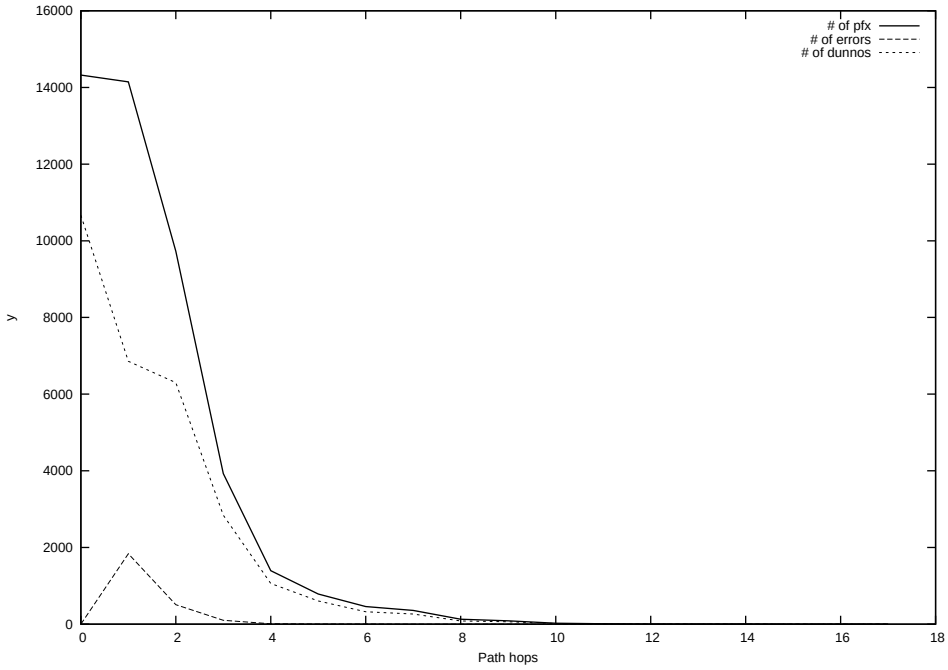
2013-08-30



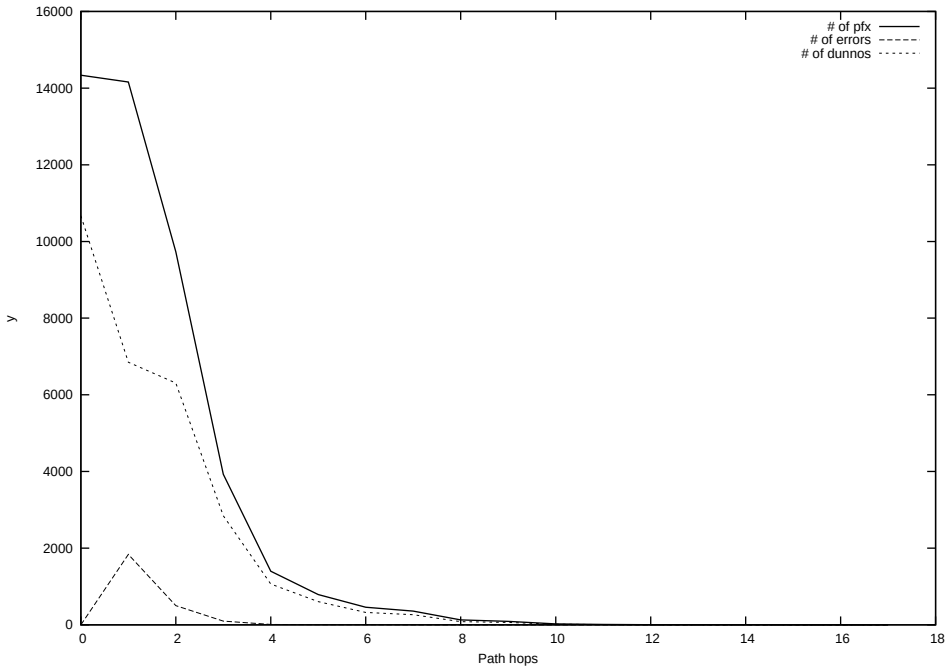
2013-08-31



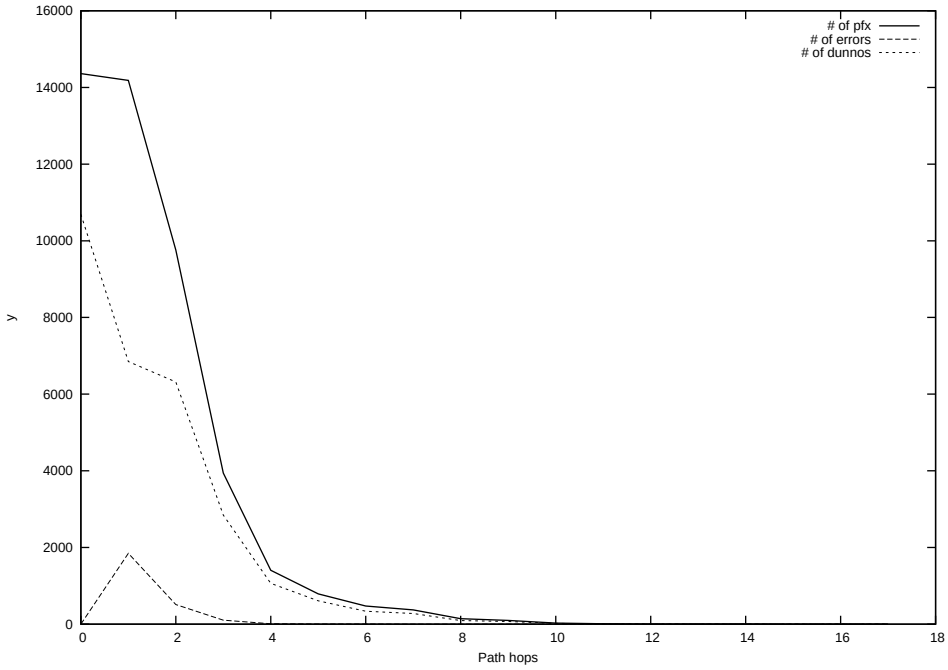
2013-09-01



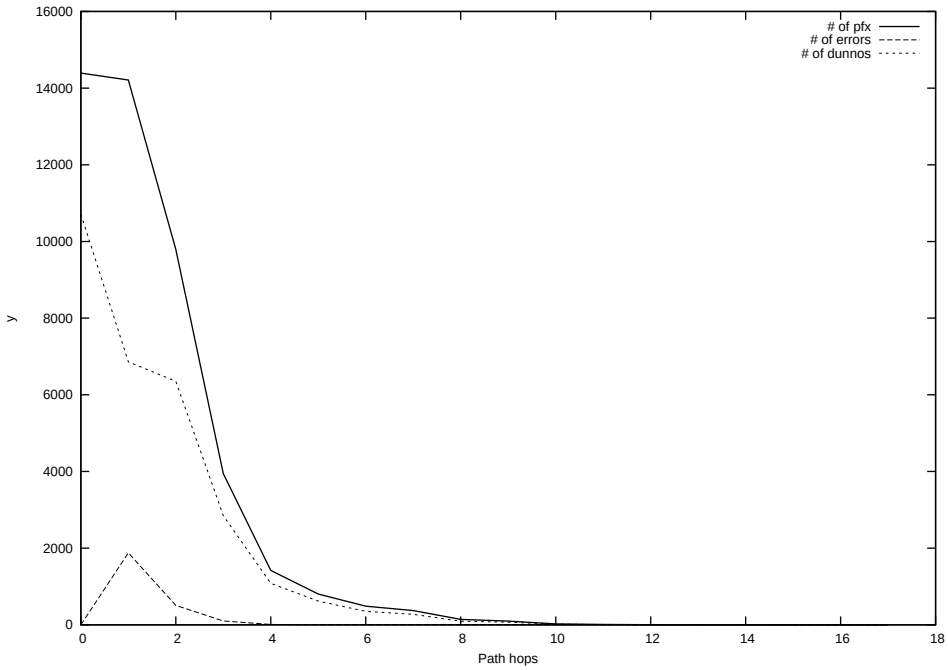
2013-09-02



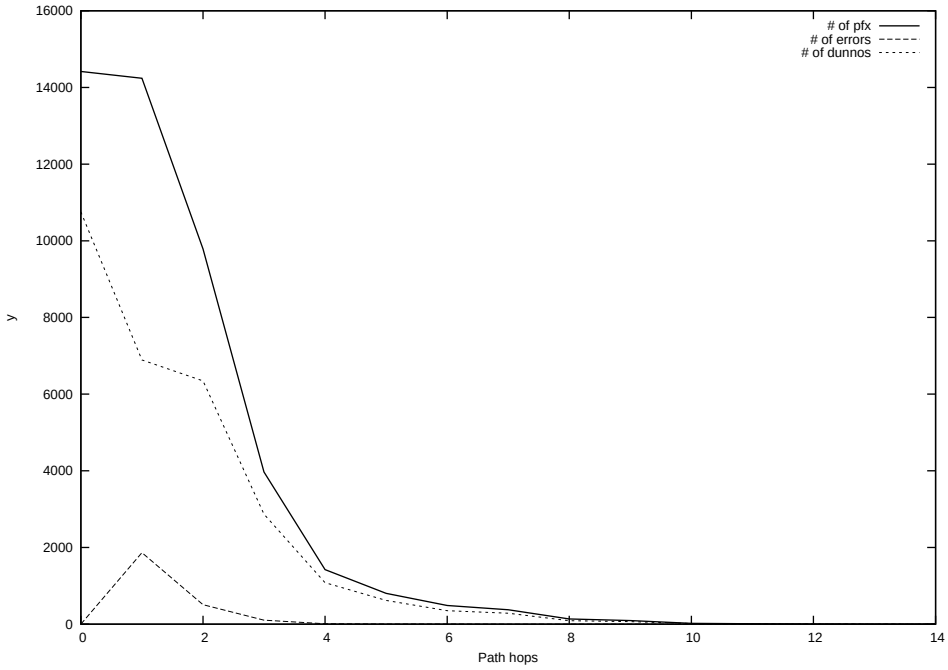
2013-09-03



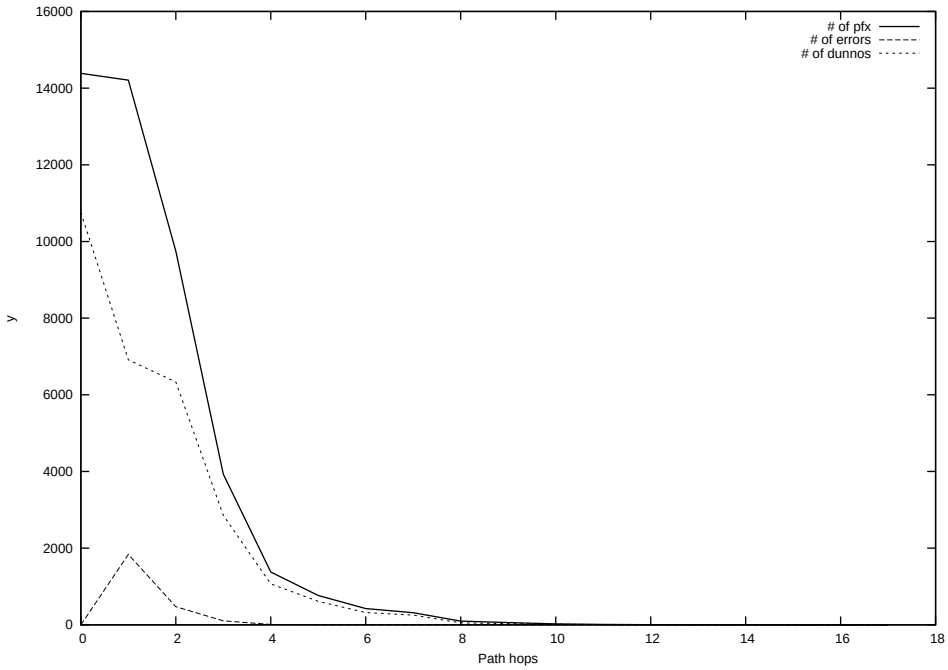
2013-09-04



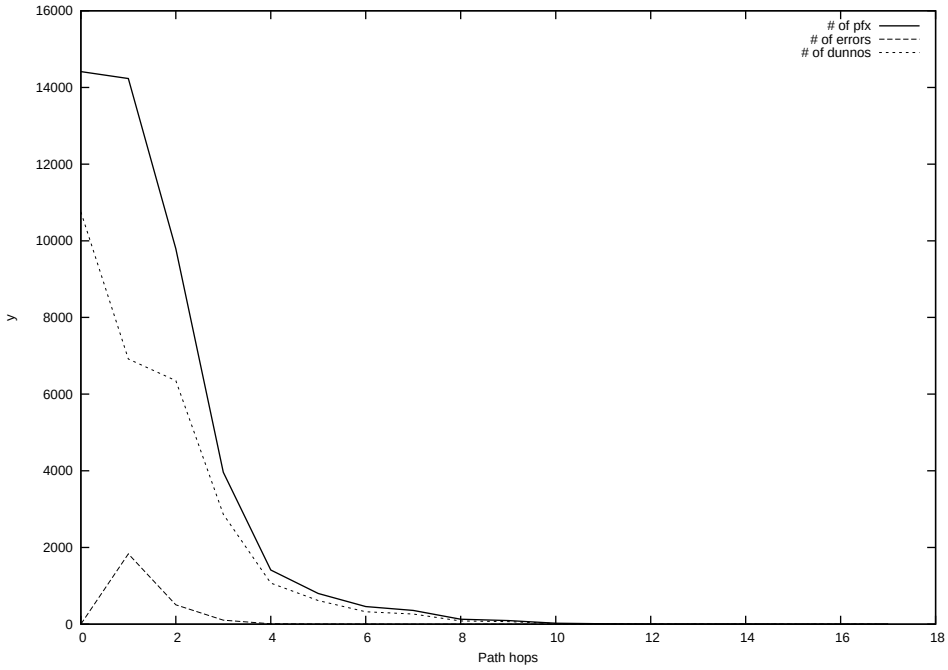
2013-09-05



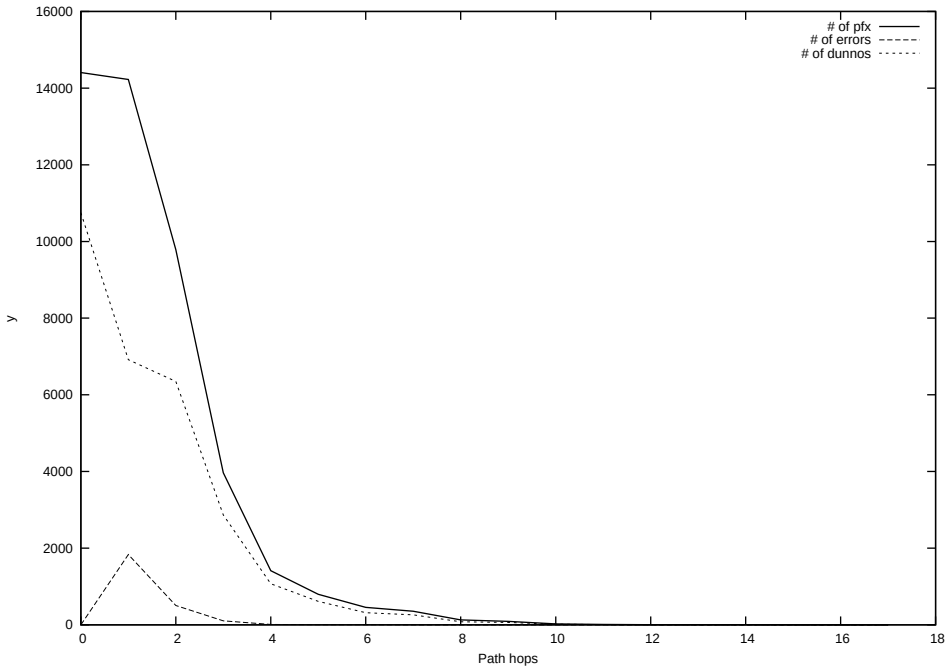
2013-09-06



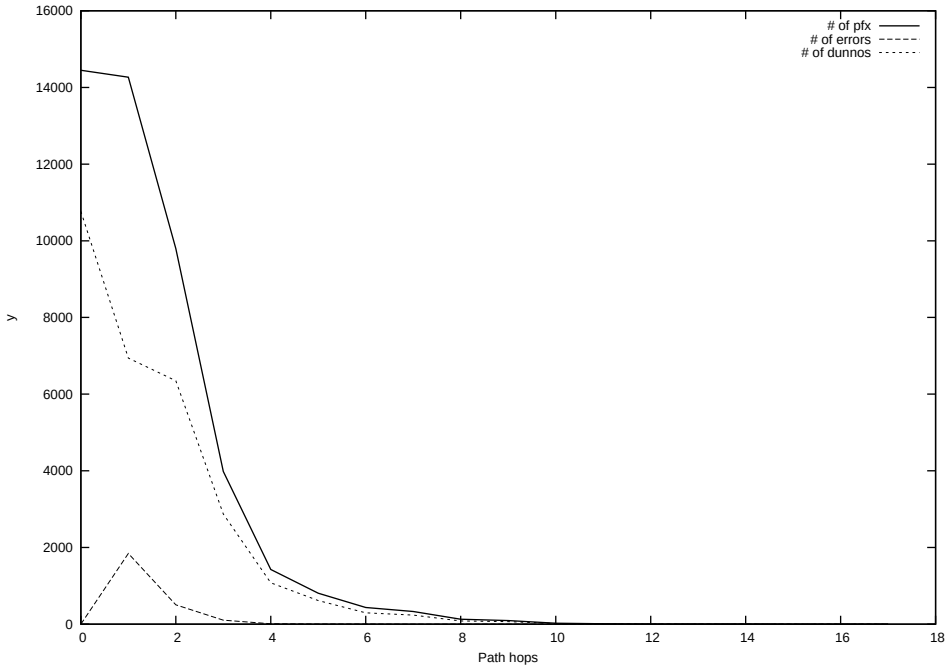
2013-09-07



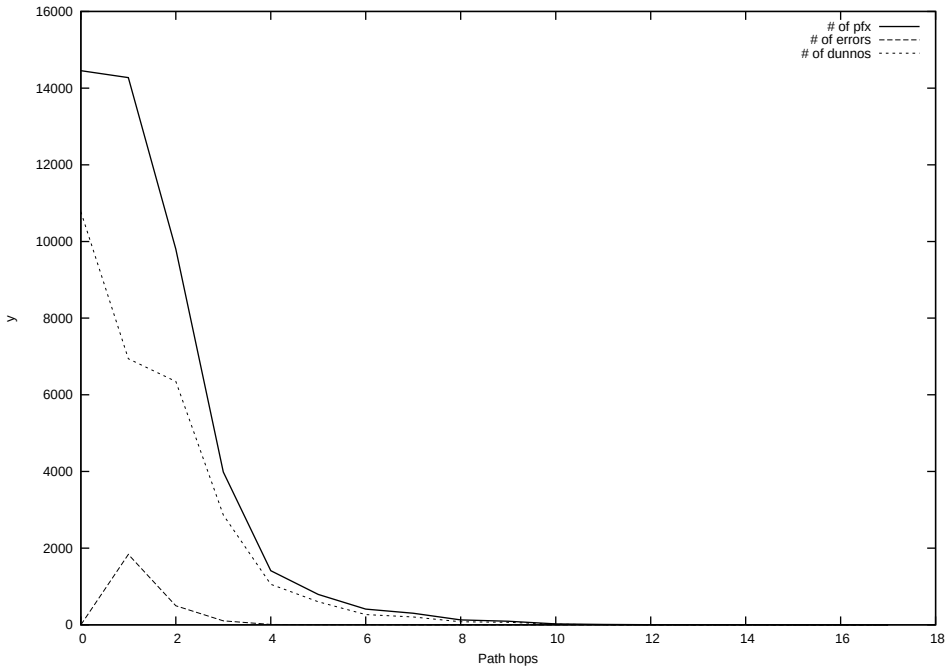
2013-09-08



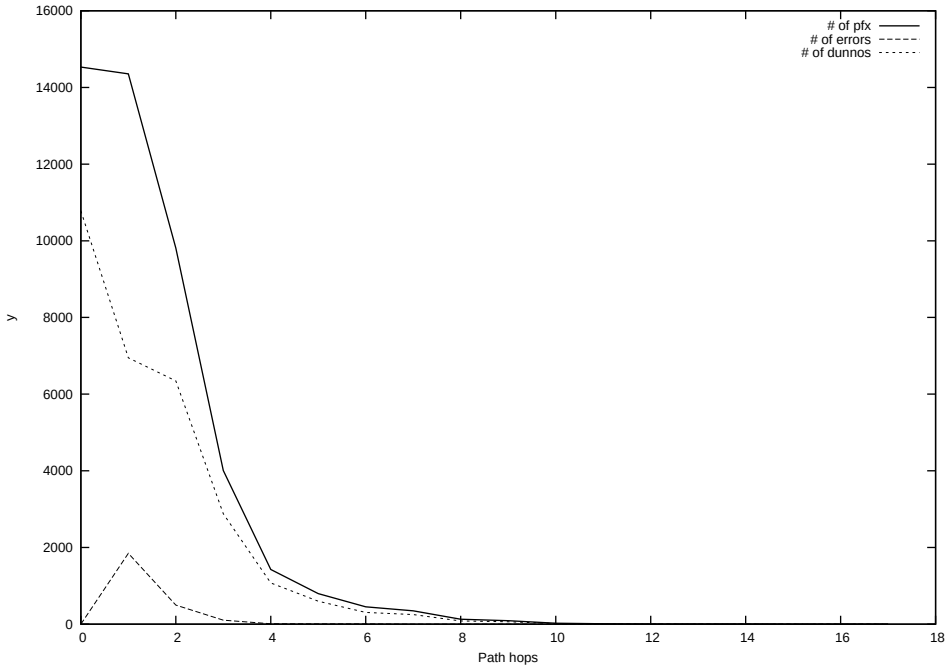
2013-09-09



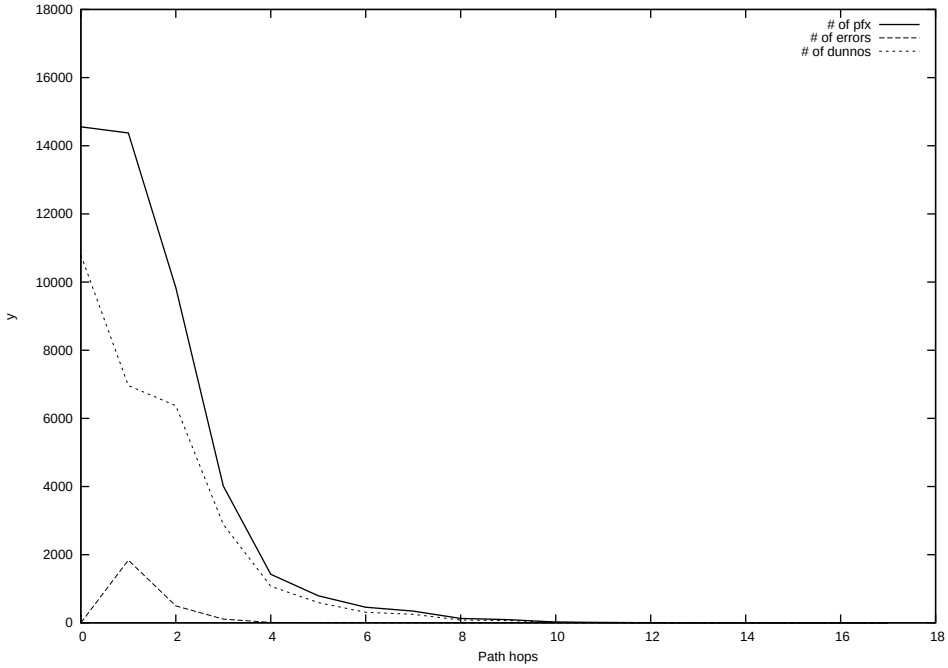
2013-09-10



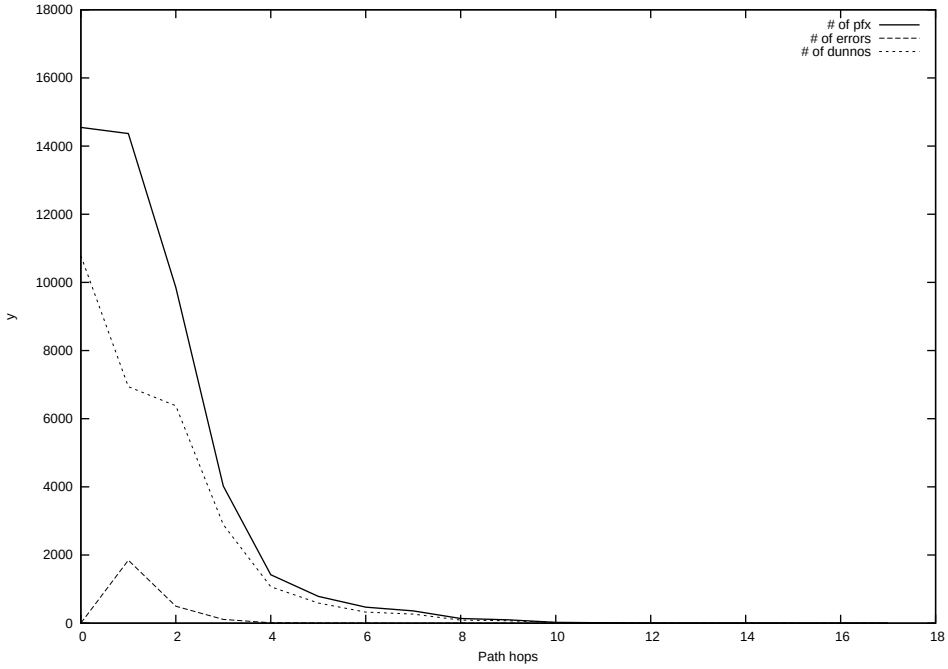
2013-09-11



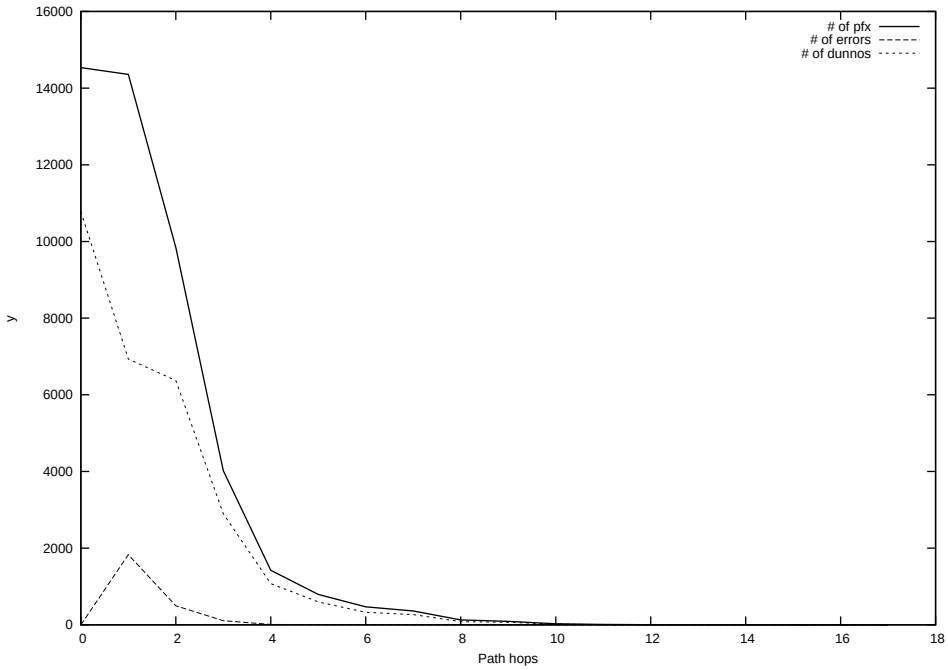
2013-09-12



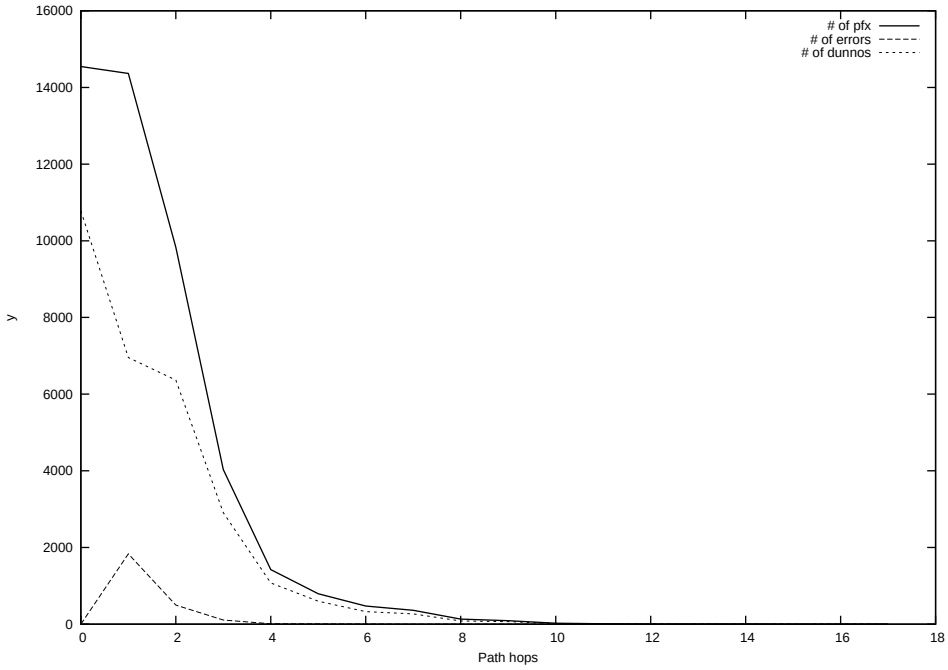
2013-09-13



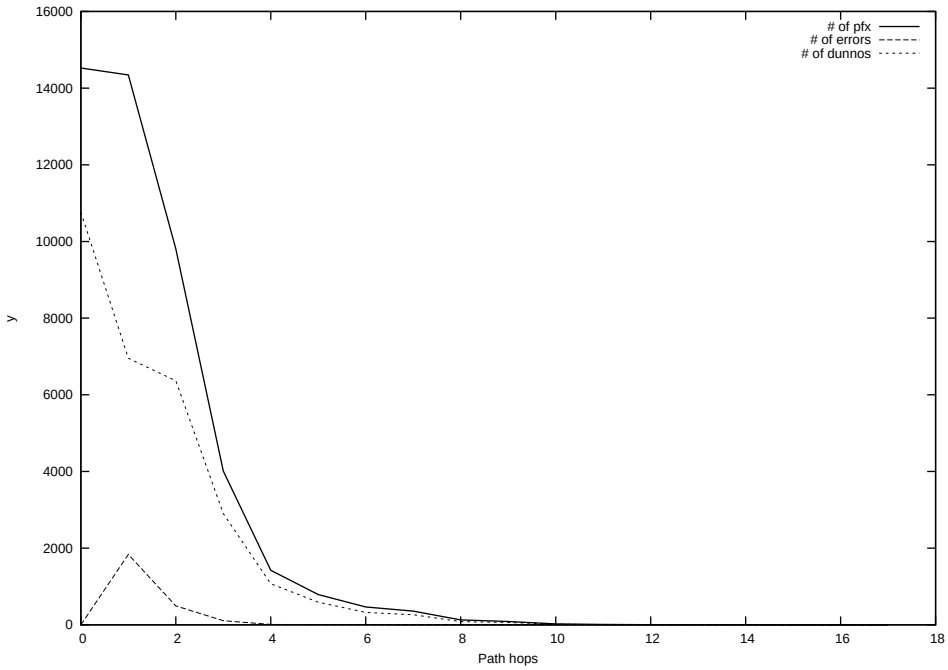
2013-09-14



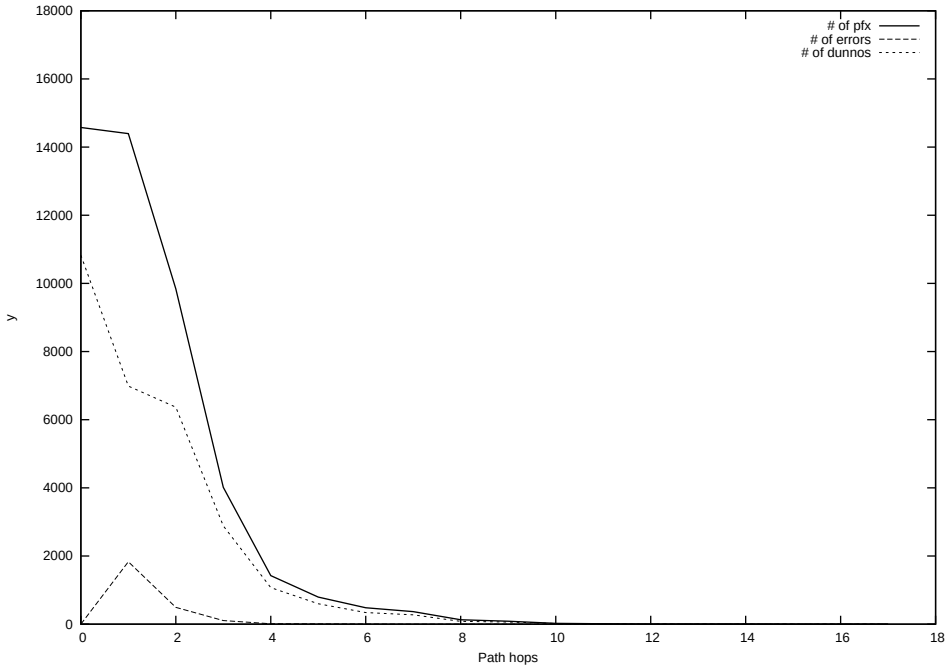
2013-09-15



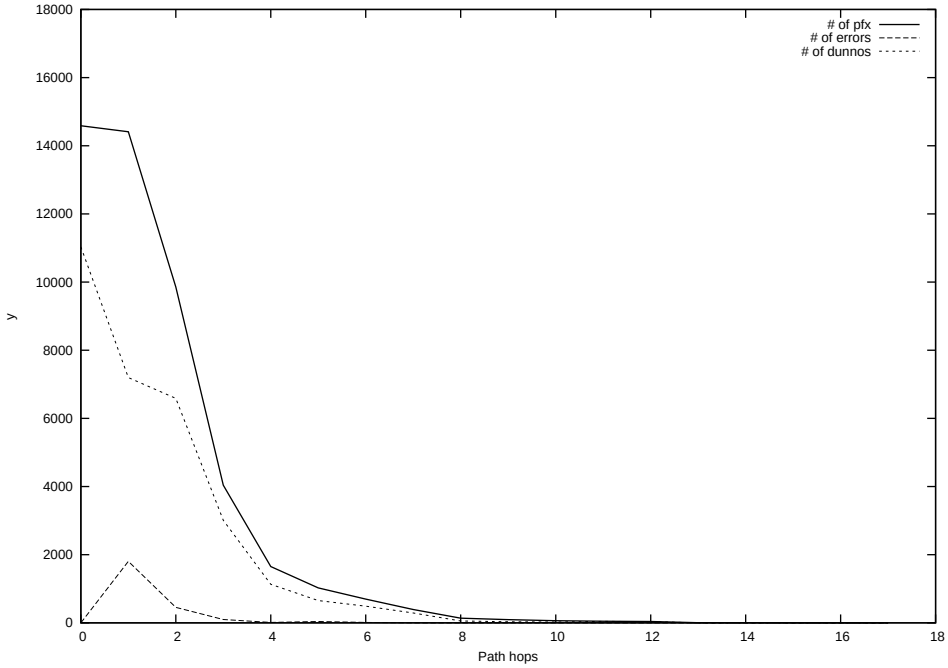
2013-09-16



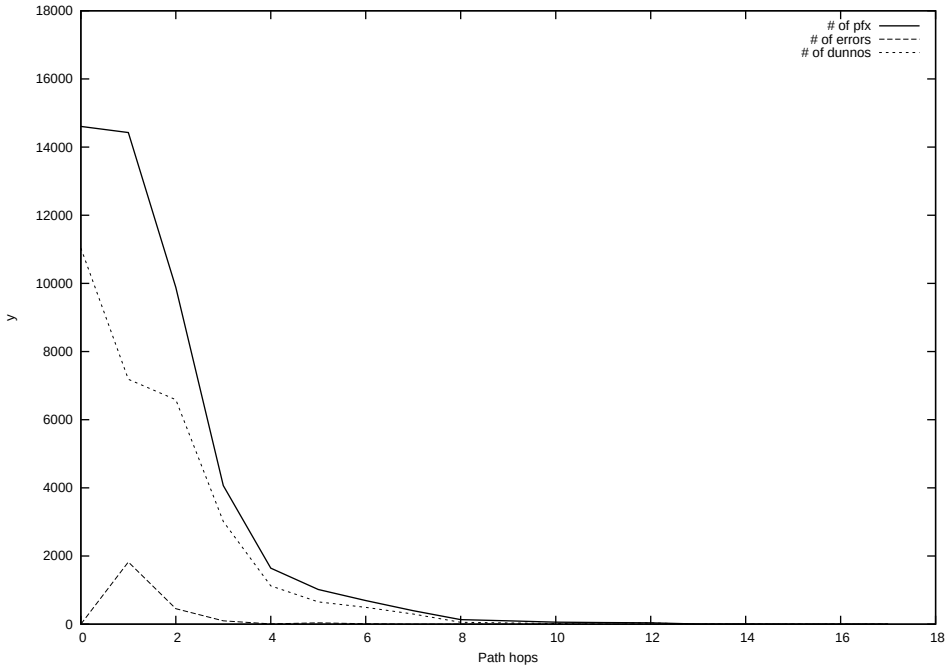
2013-09-17



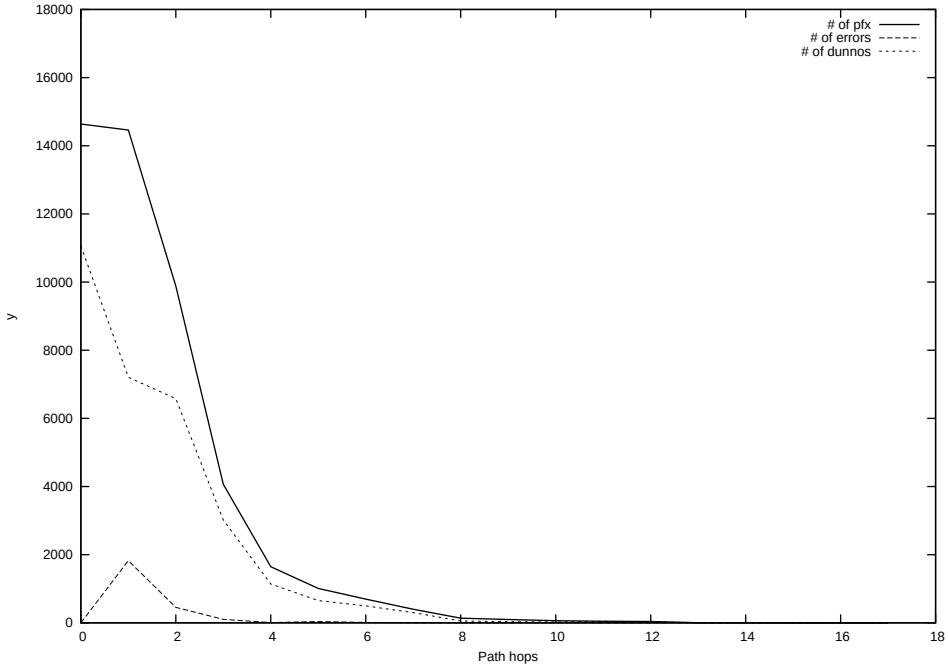
2013-09-18



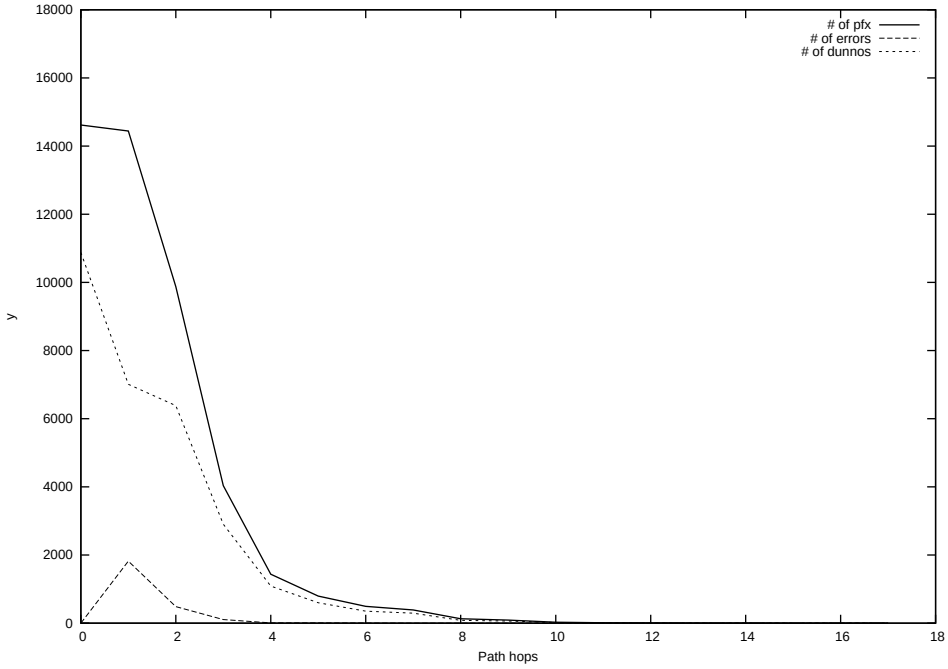
2013-09-19



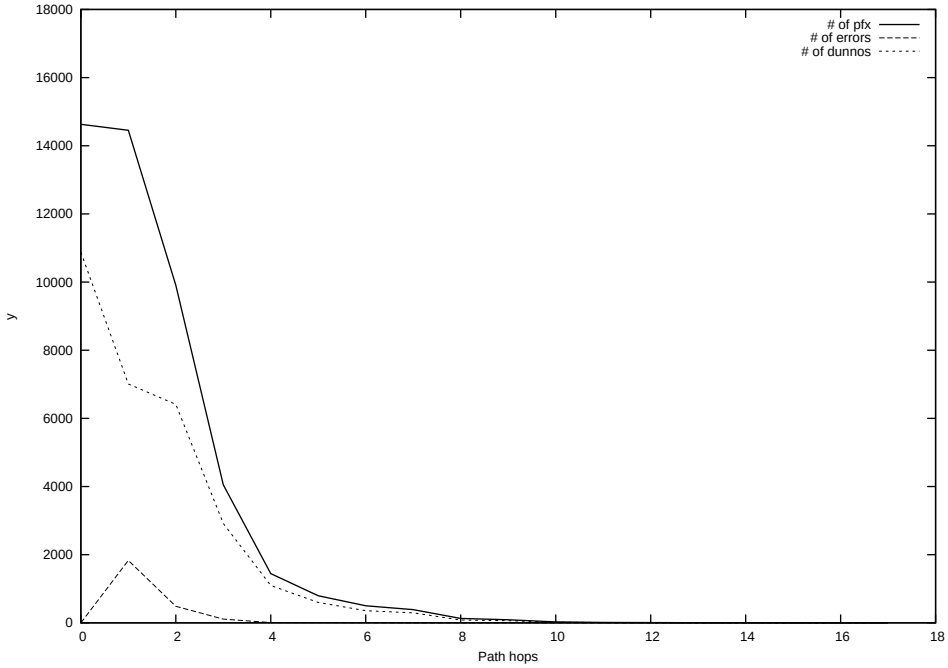
2013-09-20



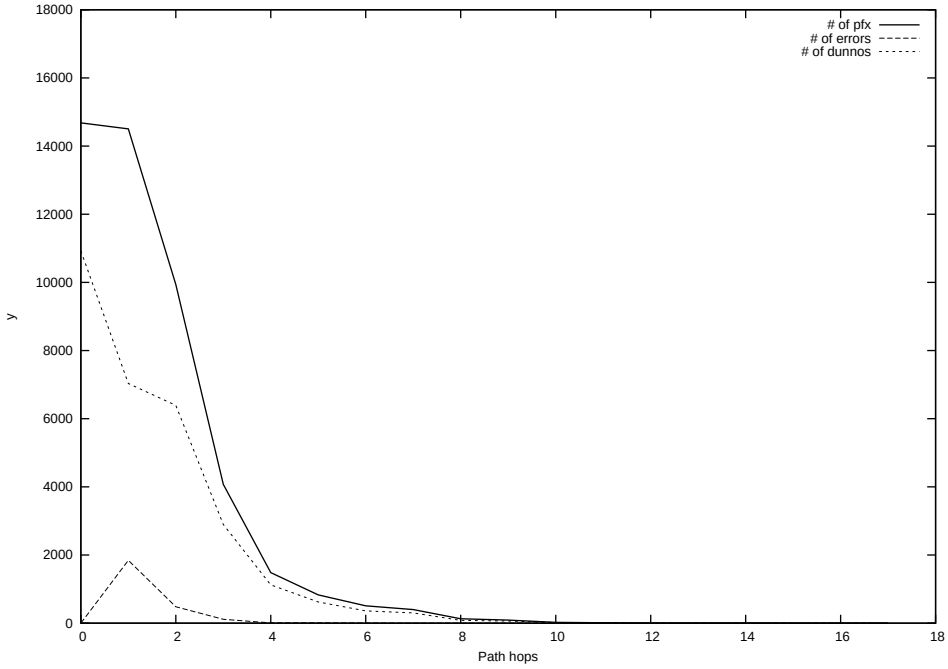
2013-09-21



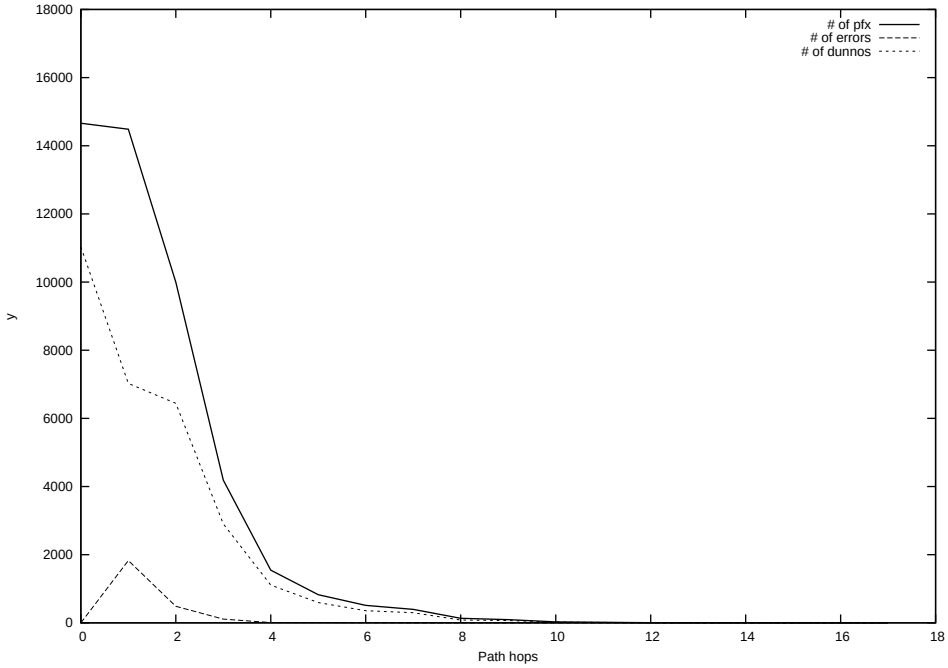
2013-09-22



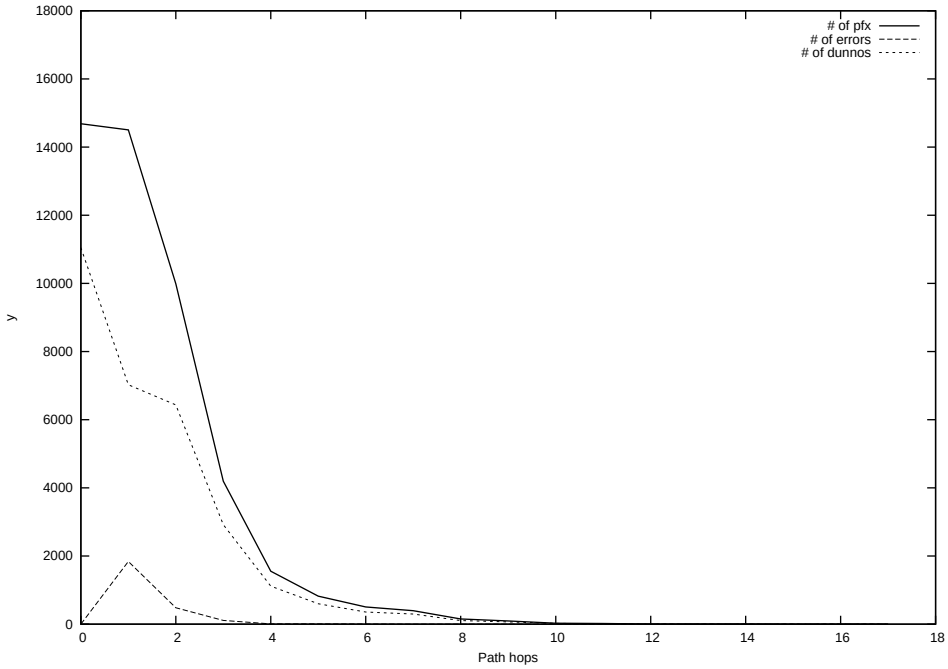
2013-09-23



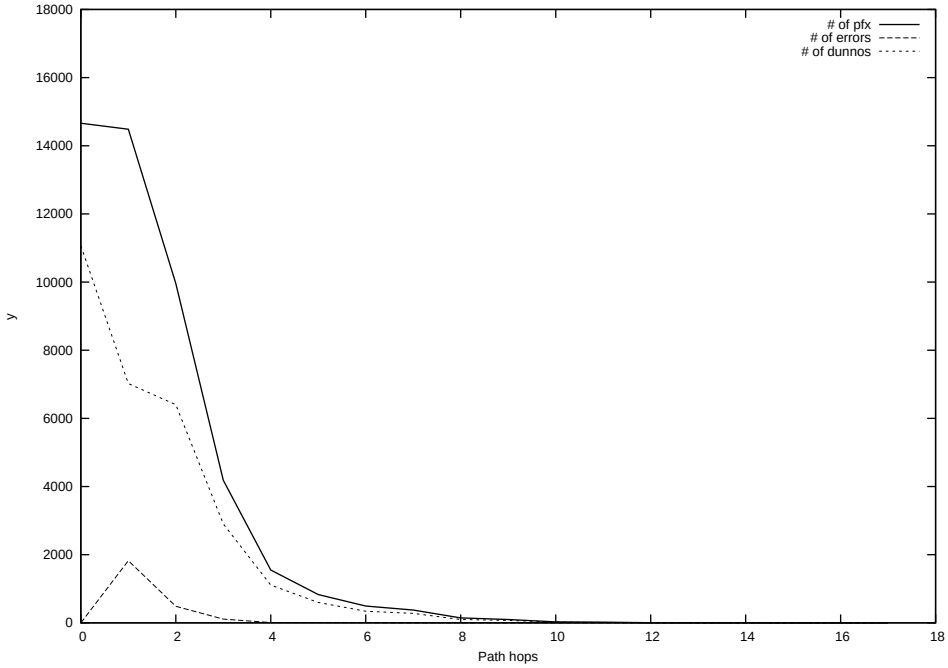
2013-09-24



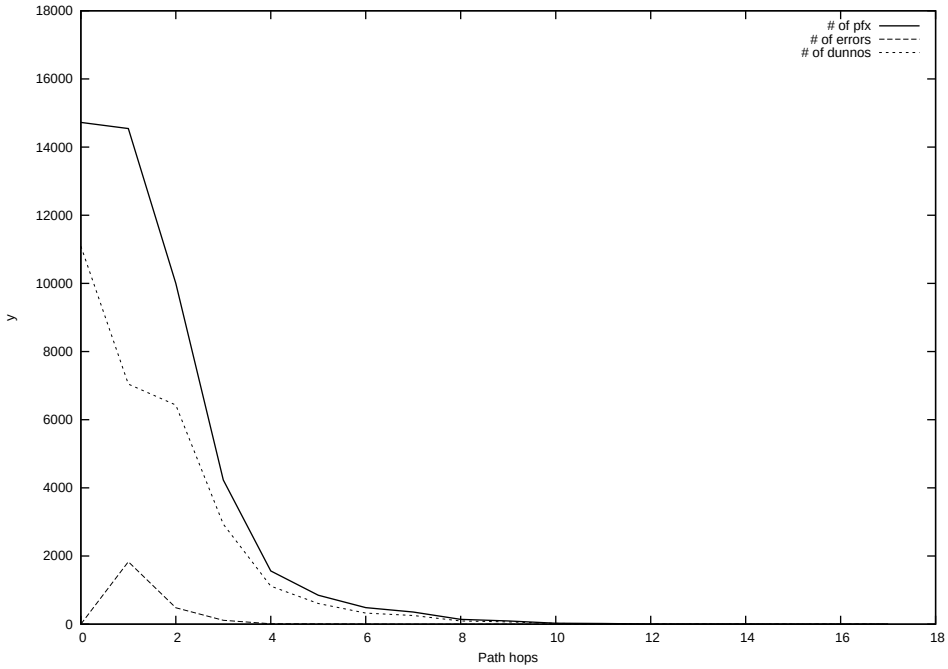
2013-09-25



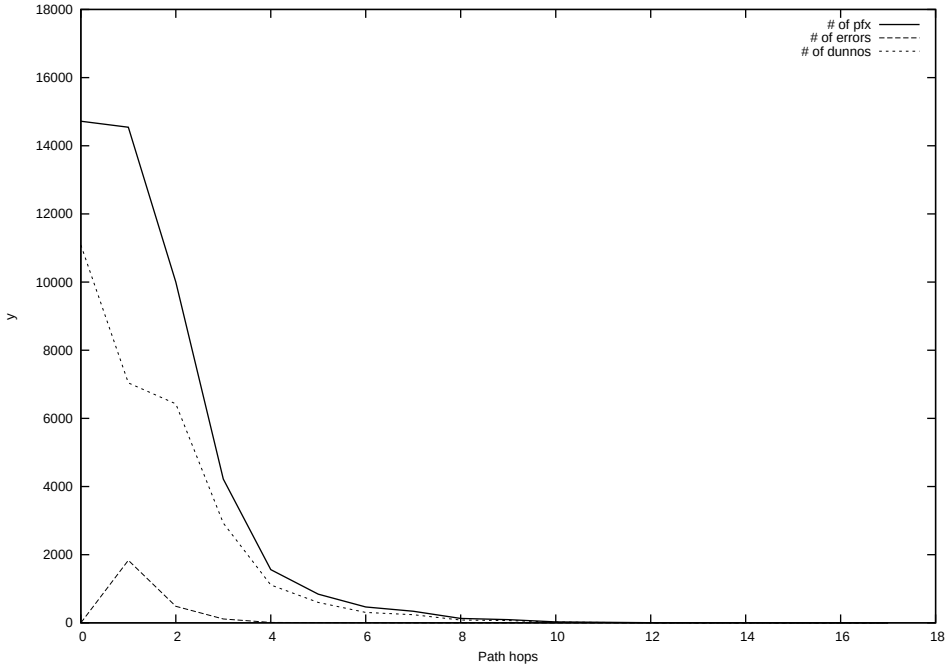
2013-09-26



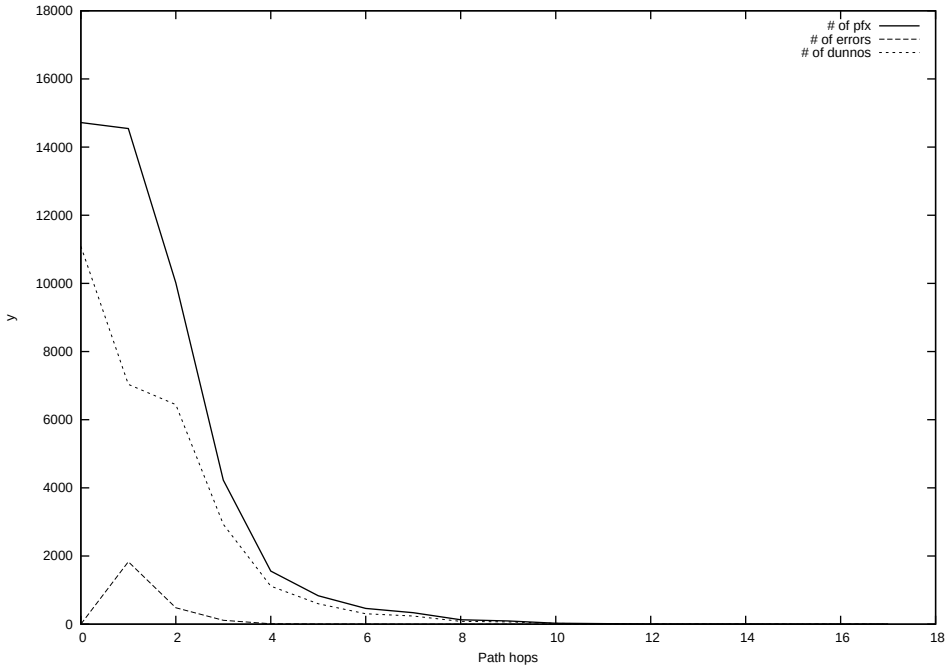
2013-09-27



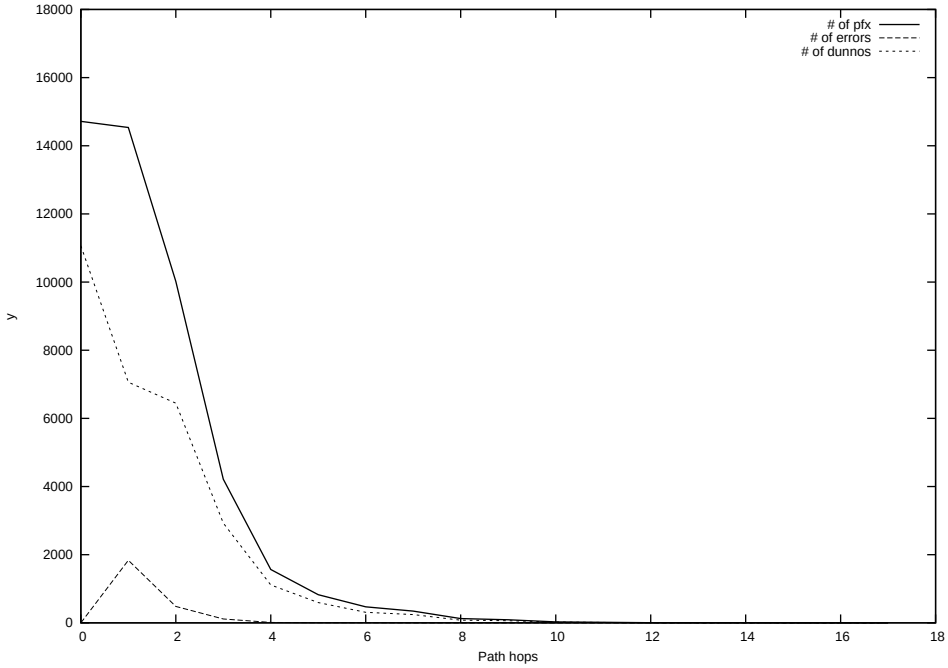
2013-09-28



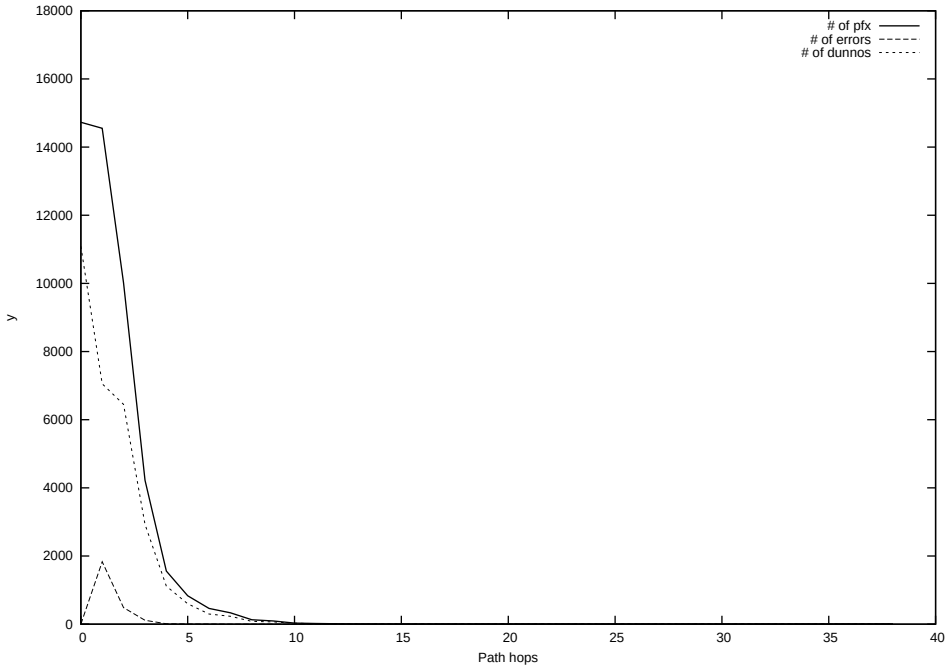
2013-09-29



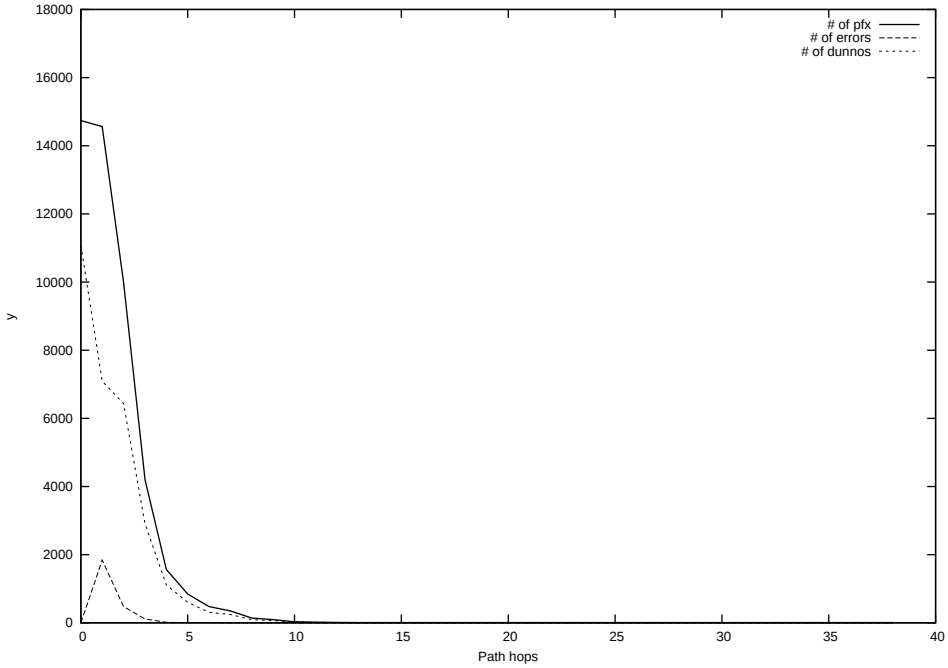
2013-09-30



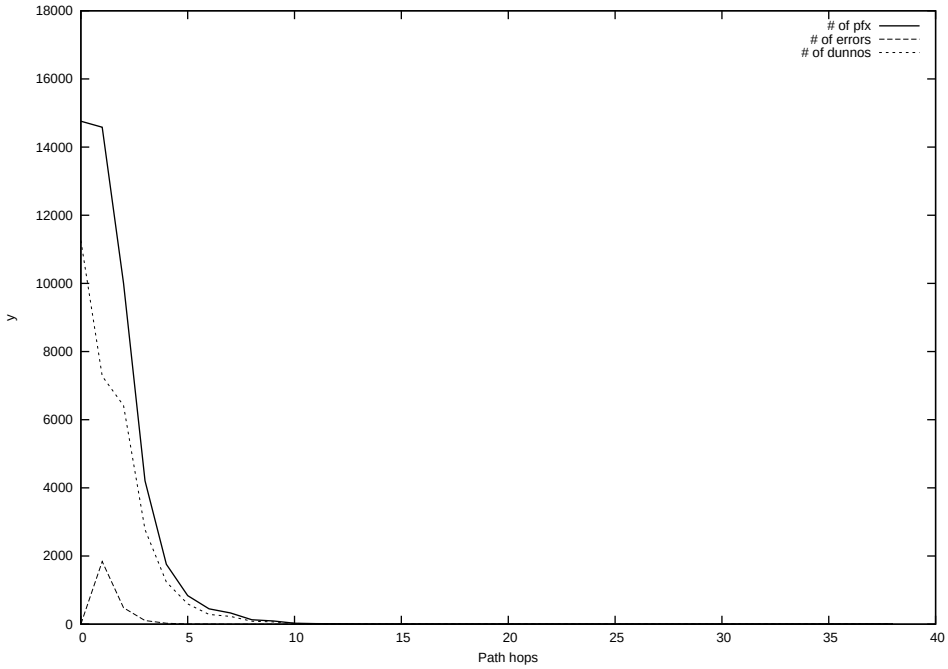
2013-10-01



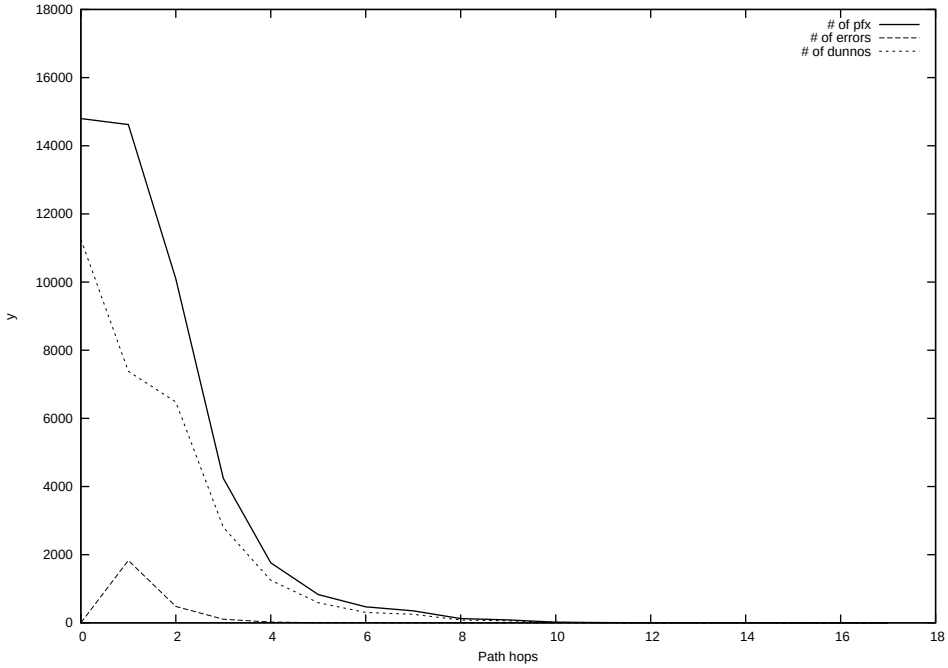
2013-10-02



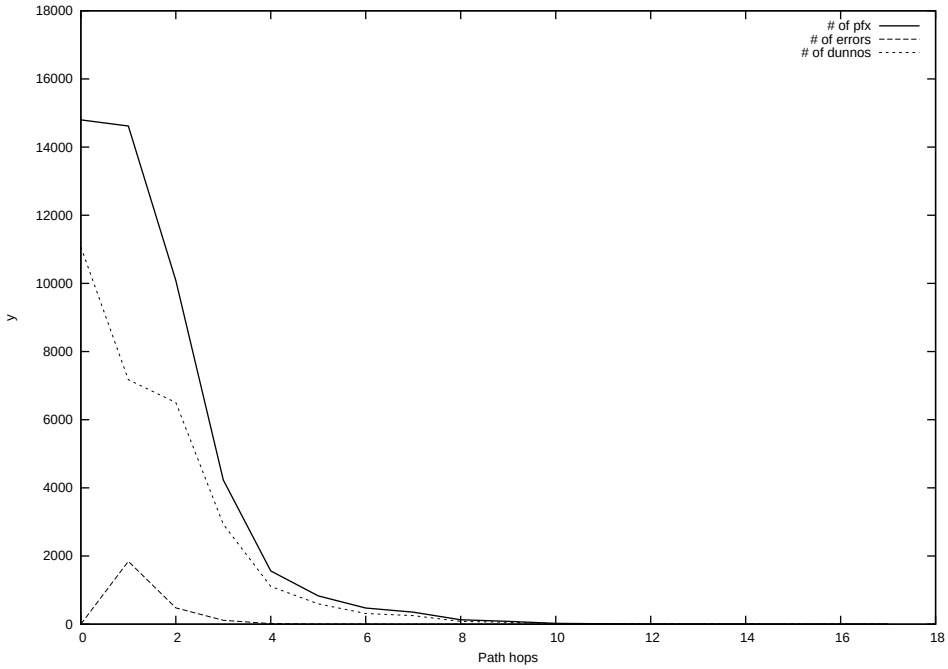
2013-10-03



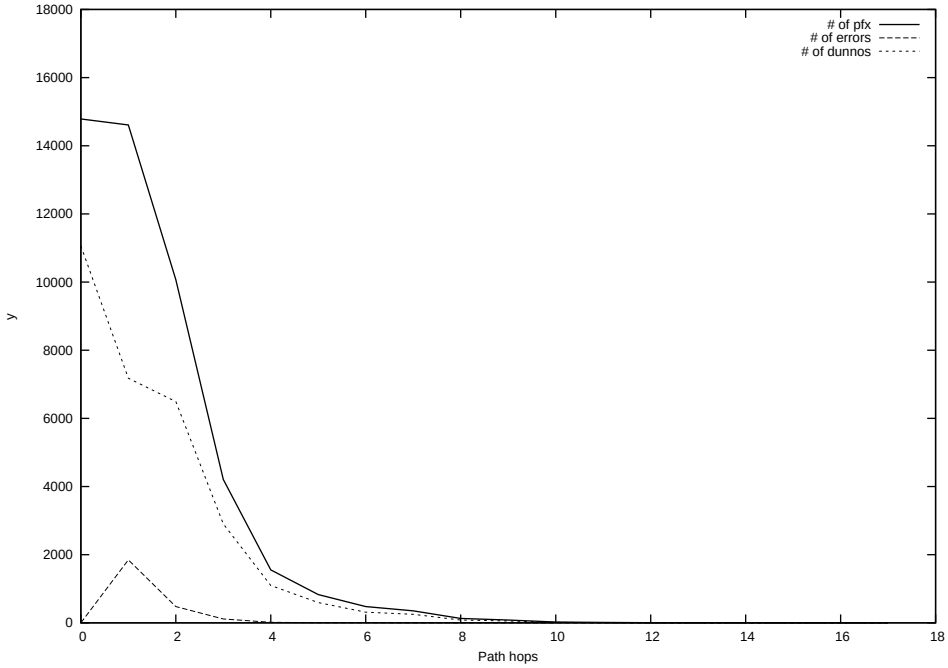
2013-10-04



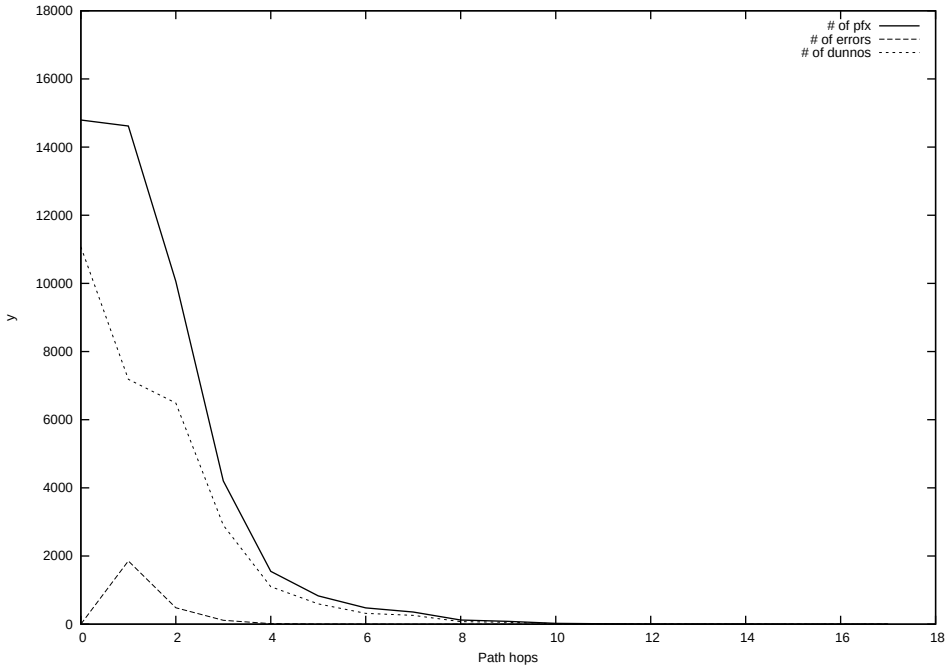
2013-10-05



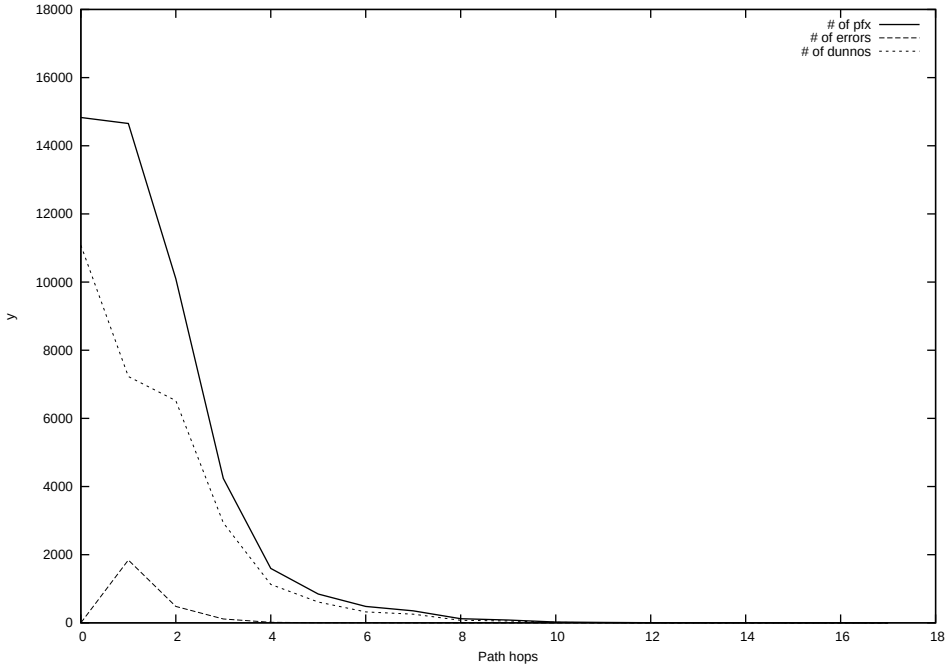
2013-10-06



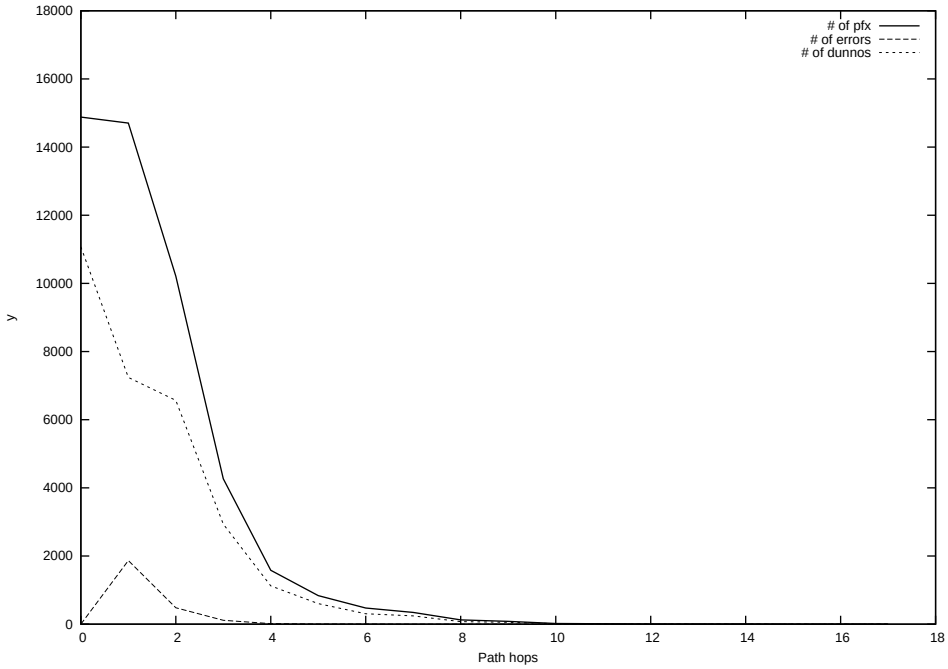
2013-10-07



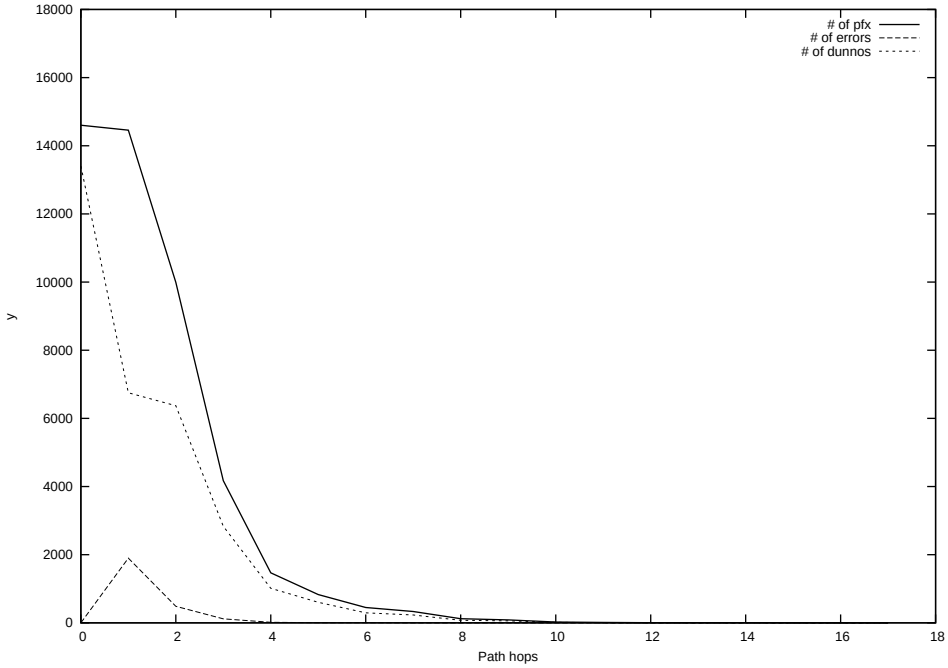
2013-10-08



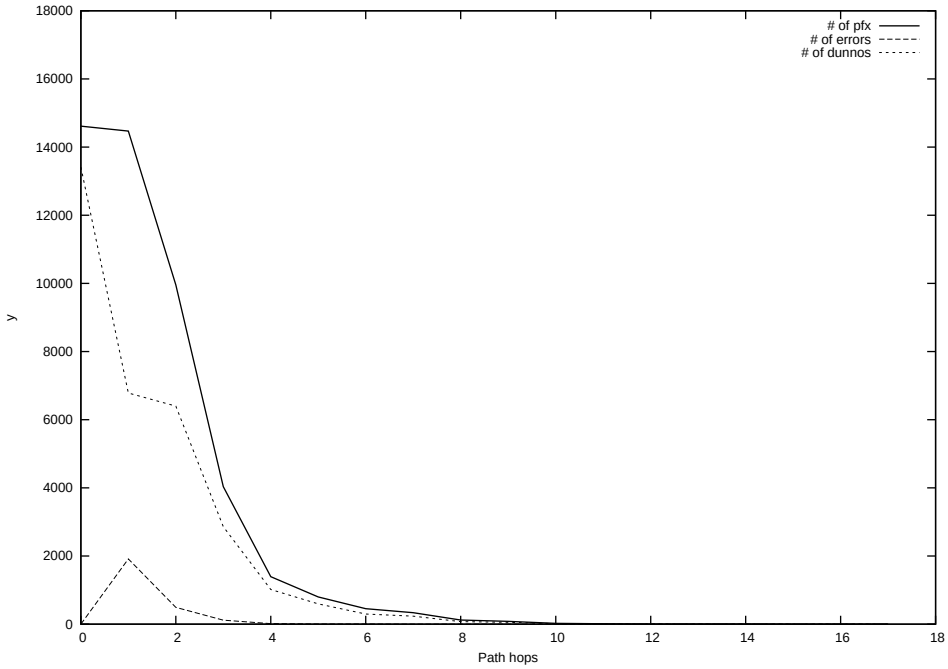
2013-10-09



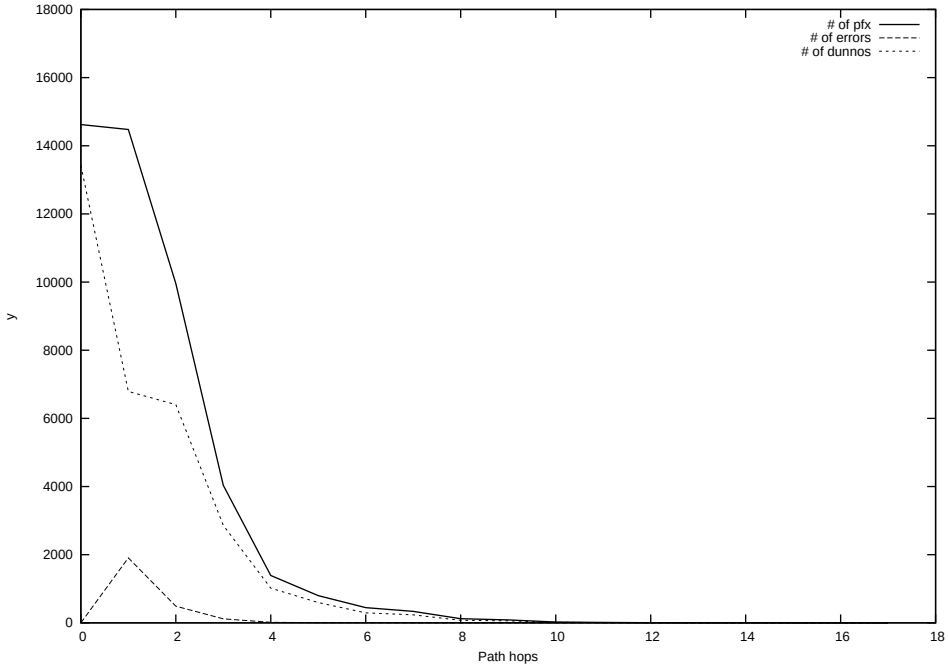
2013-10-10



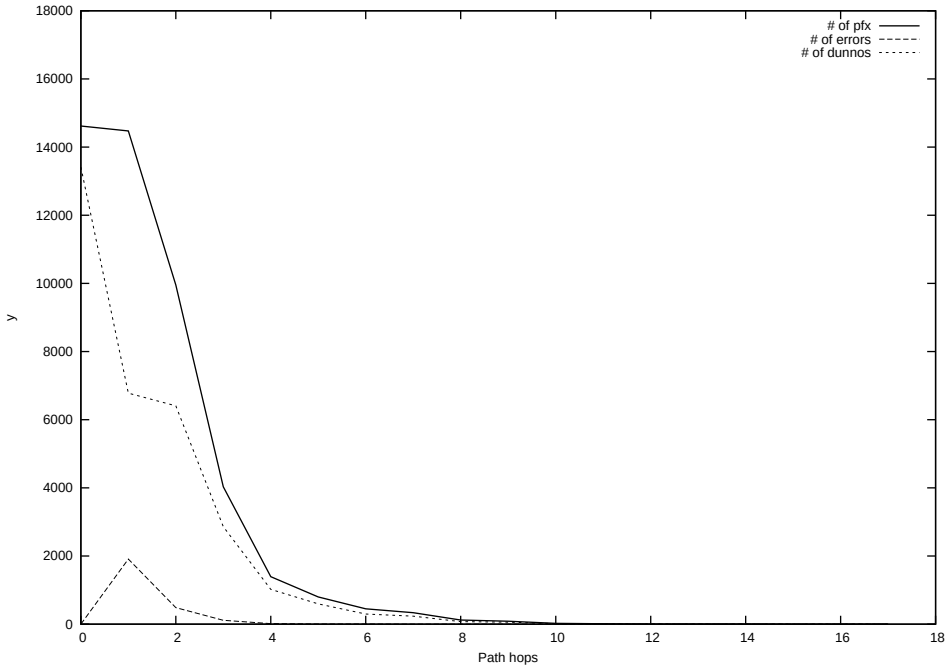
2013-10-11



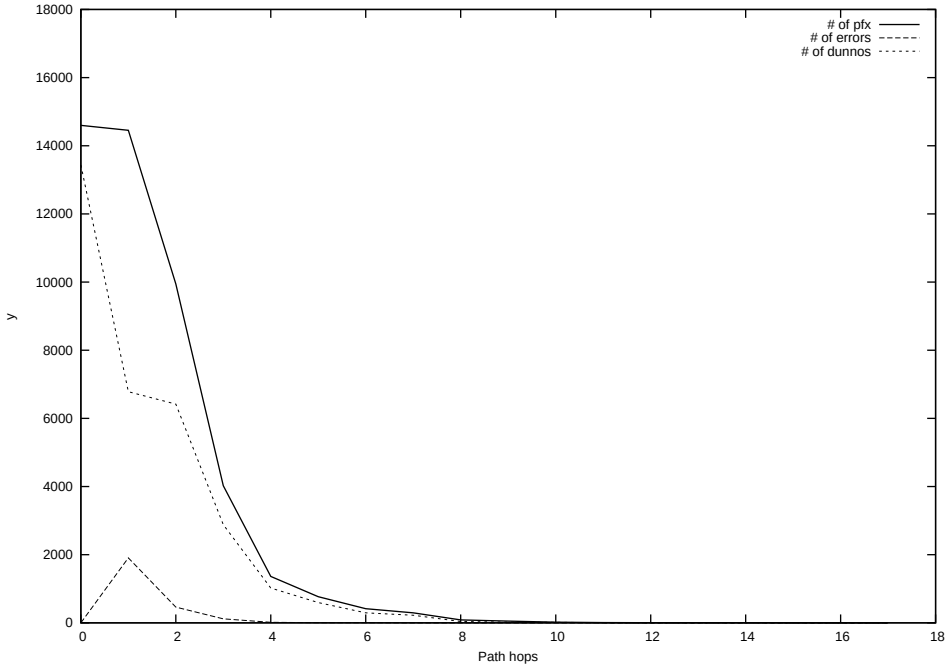
2013-10-12



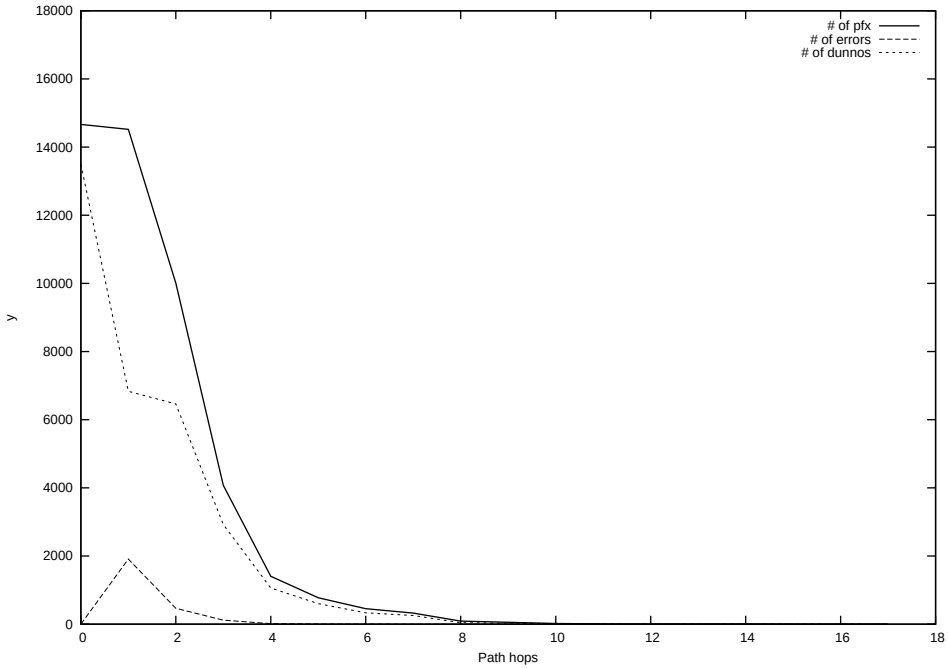
2013-10-13



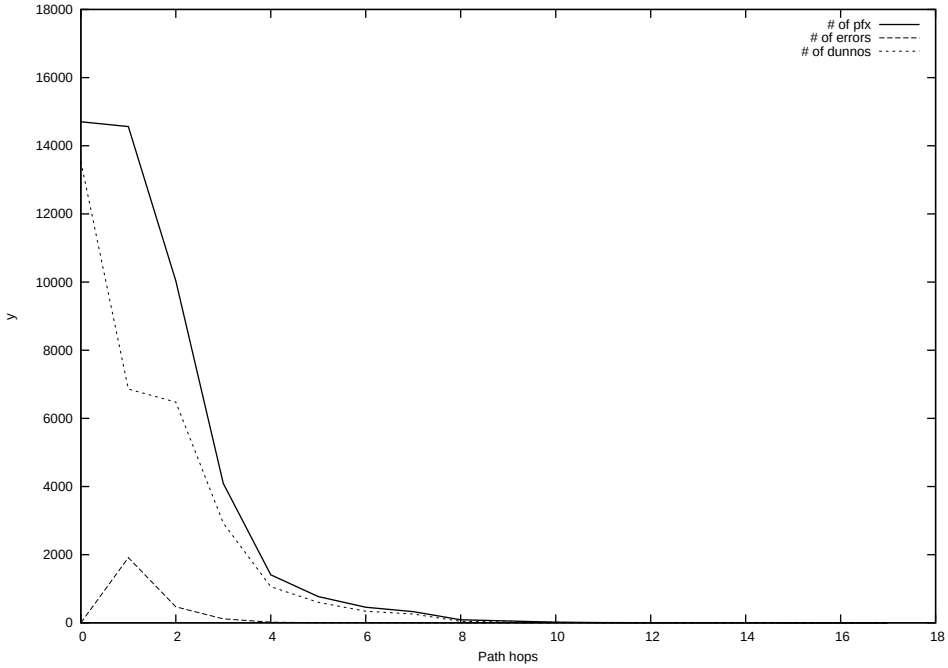
2013-10-14



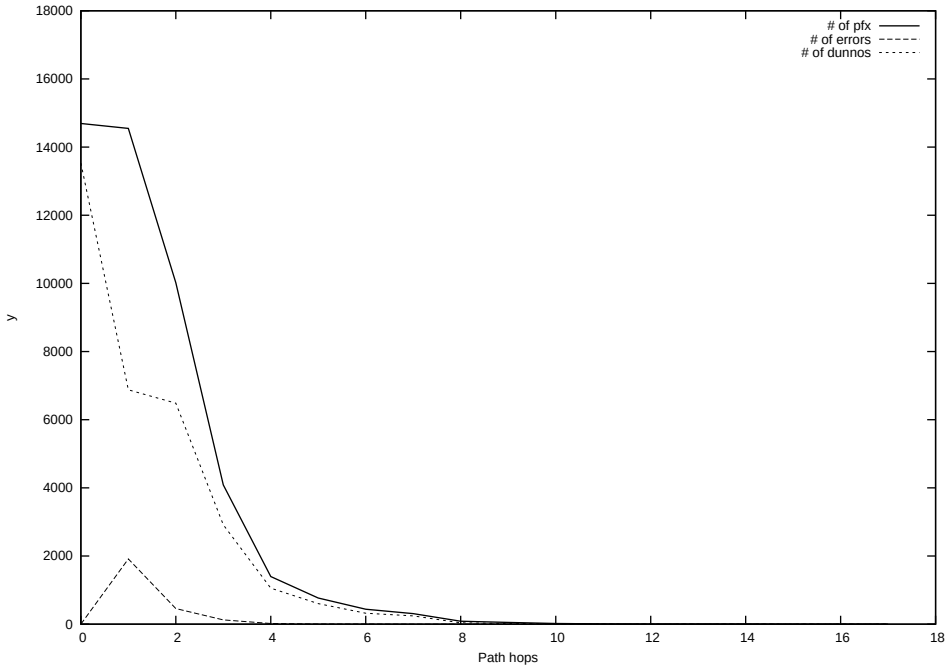
2013-10-15



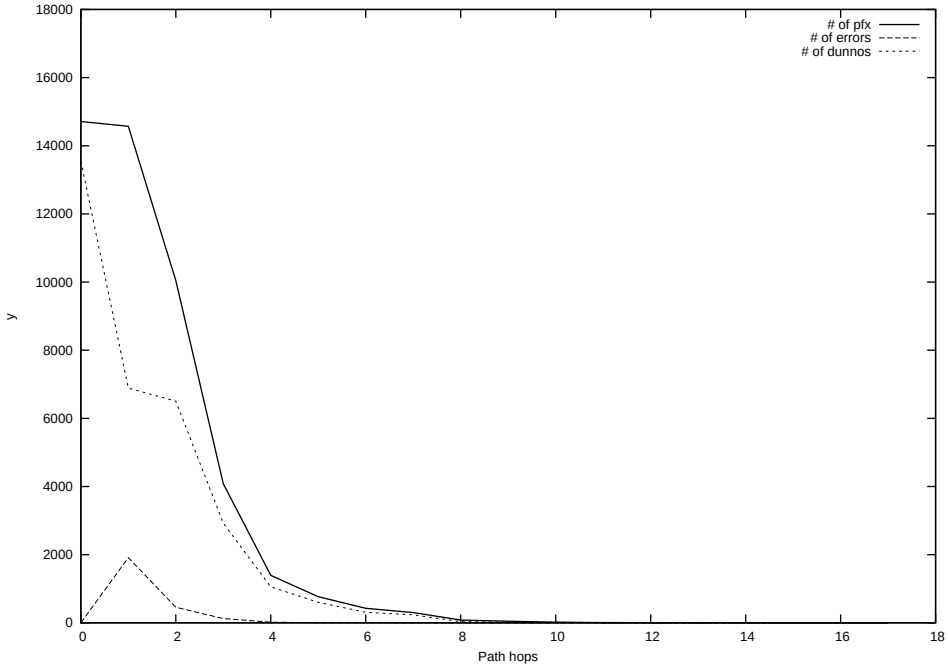
2013-10-16



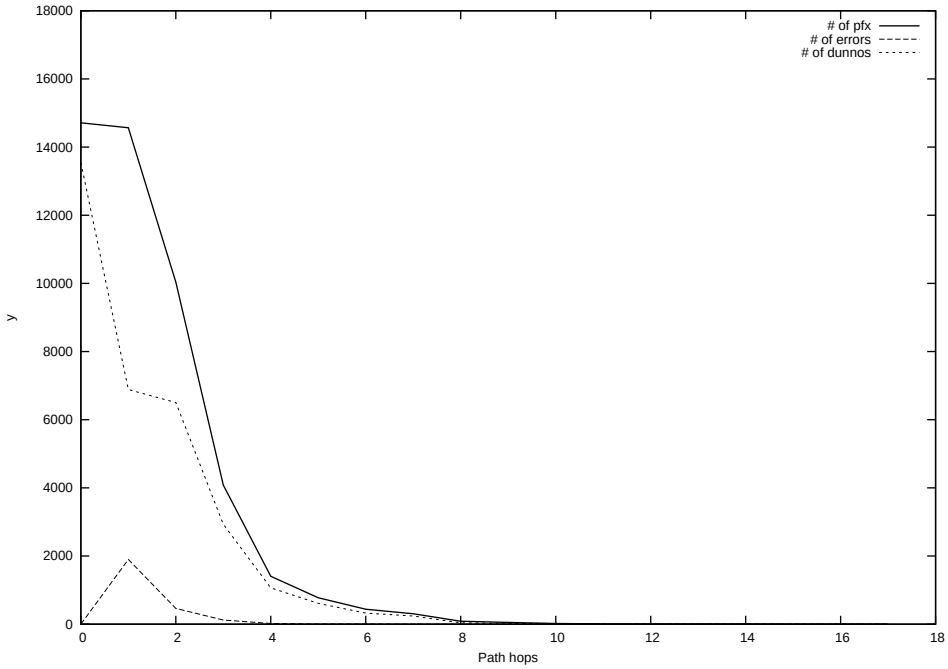
2013-10-17



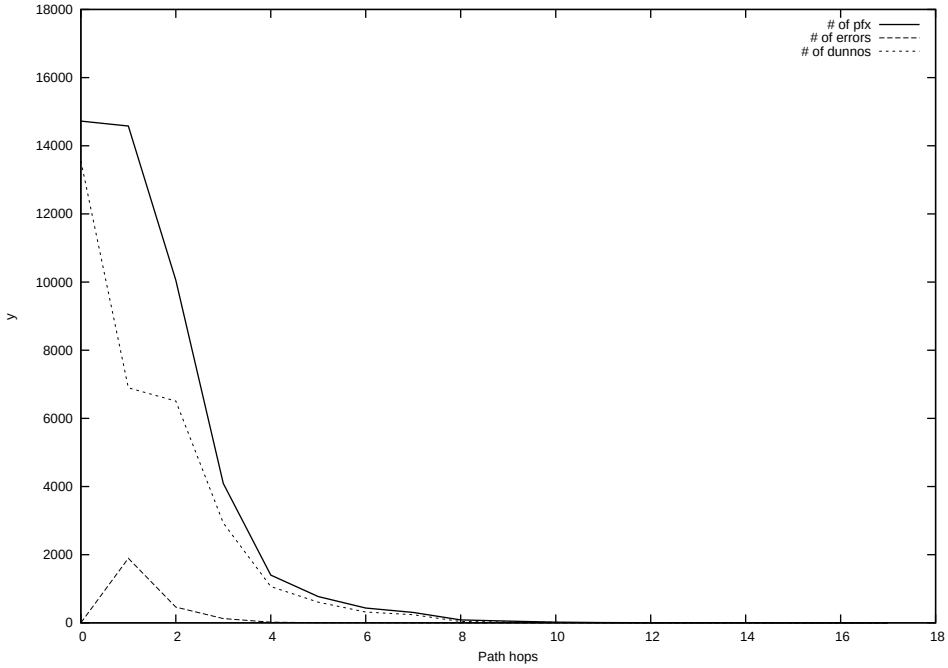
2013-10-18



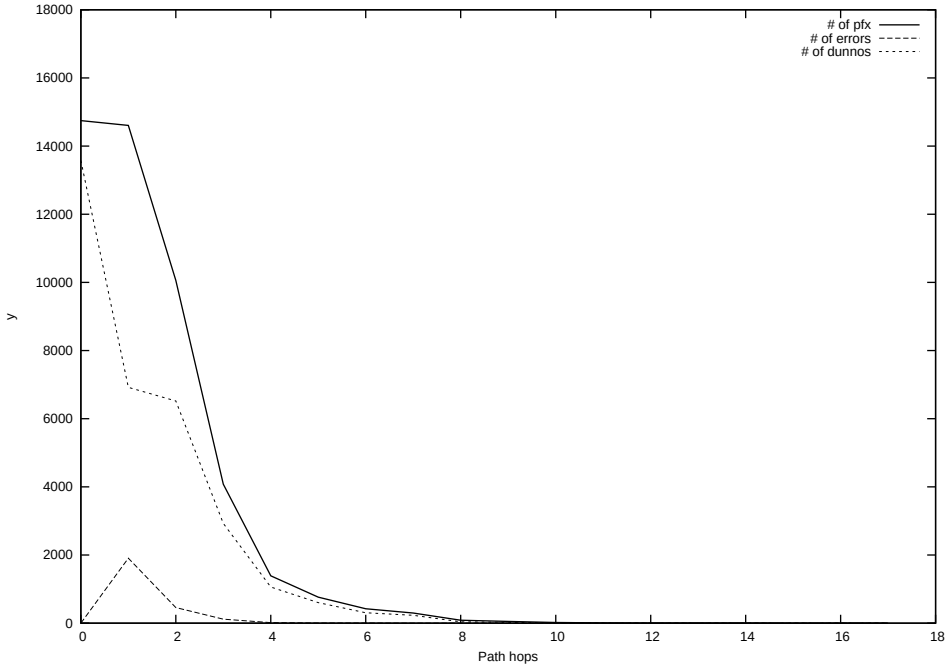
2013-10-19



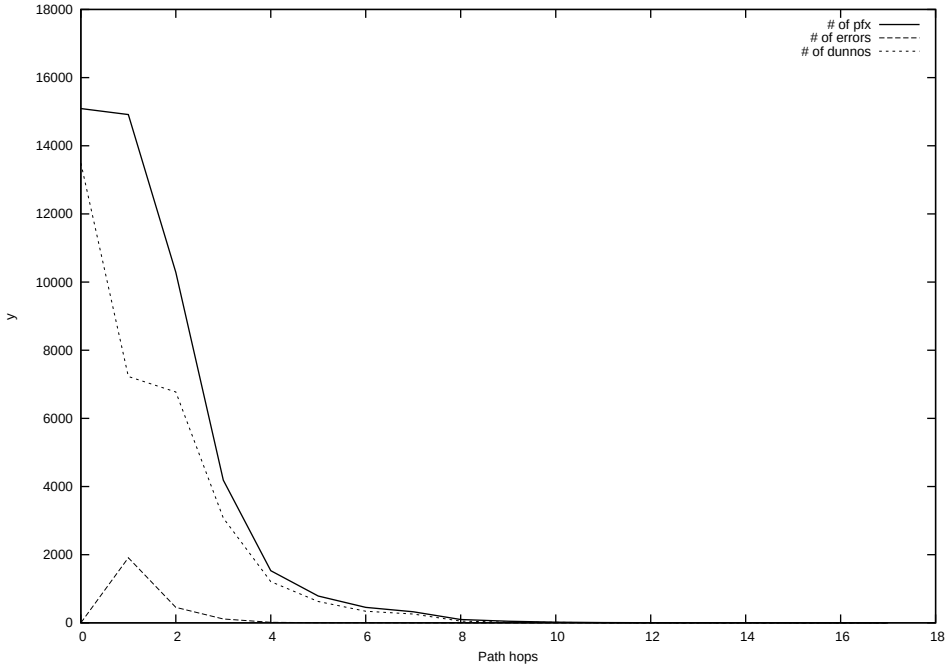
2013-10-20



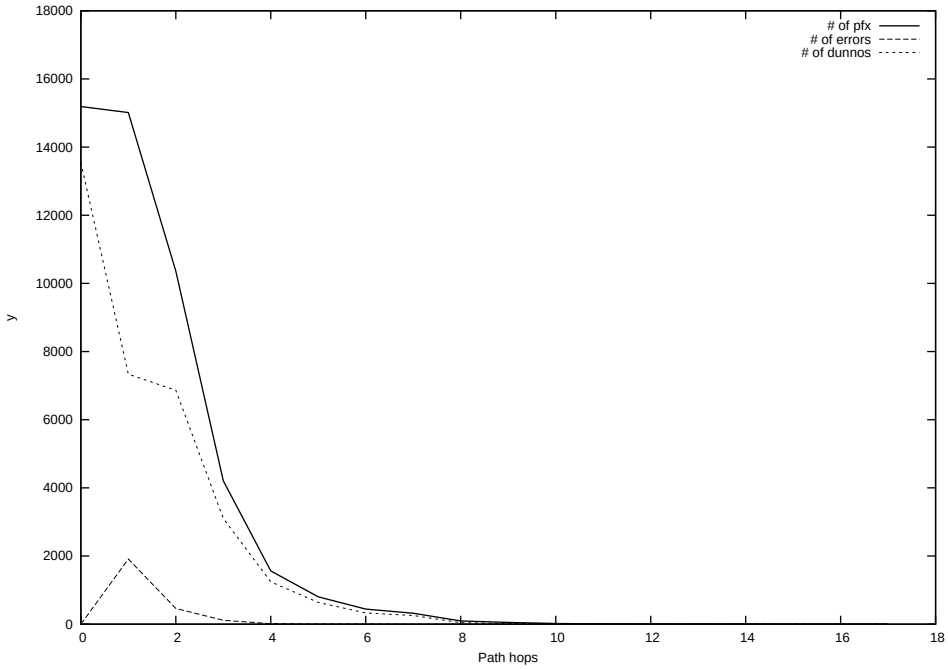
2013-10-21



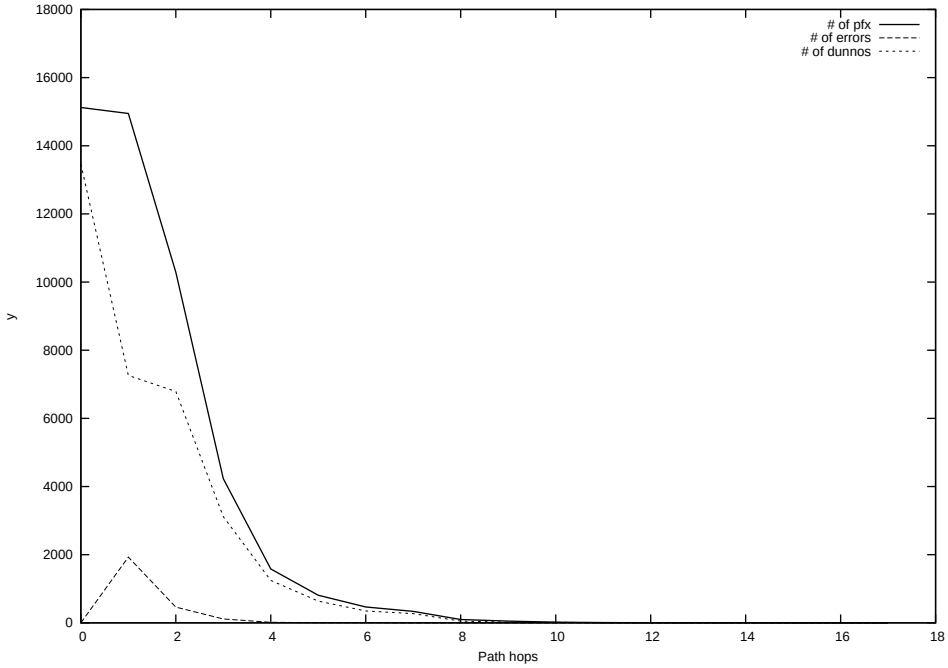
2013-10-22



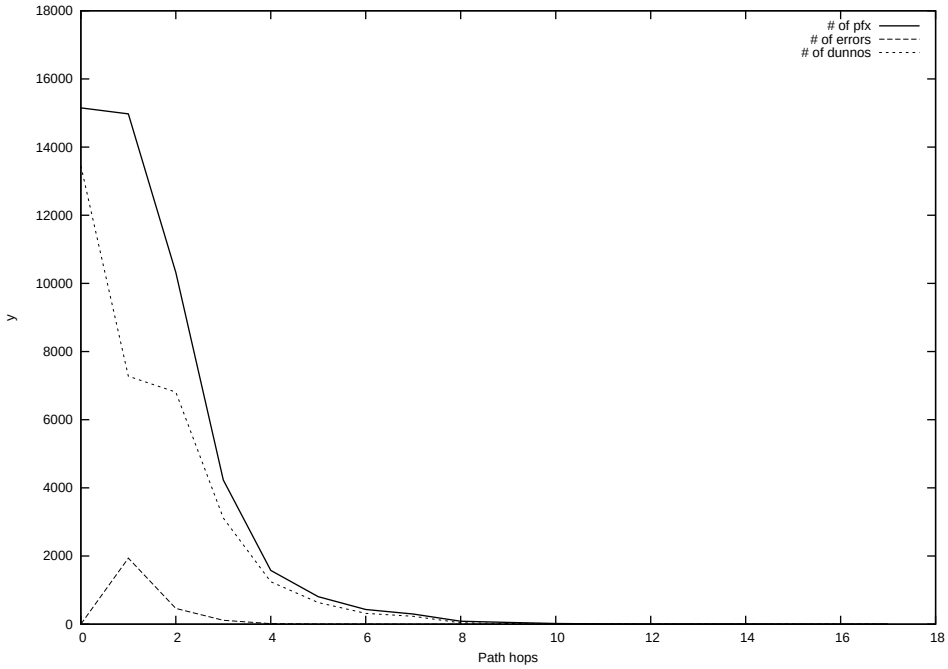
2013-10-23



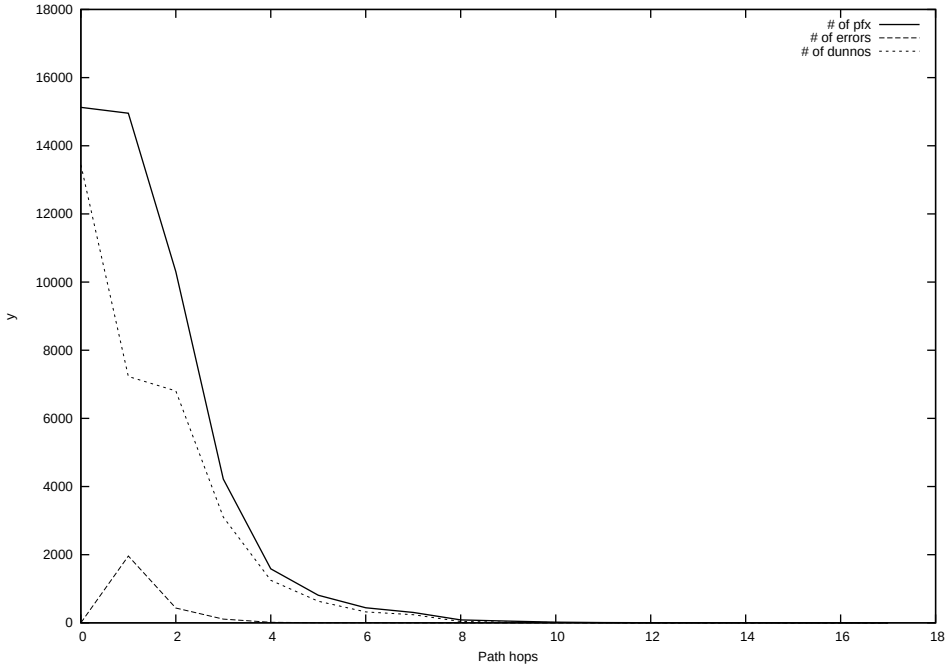
2013-10-24



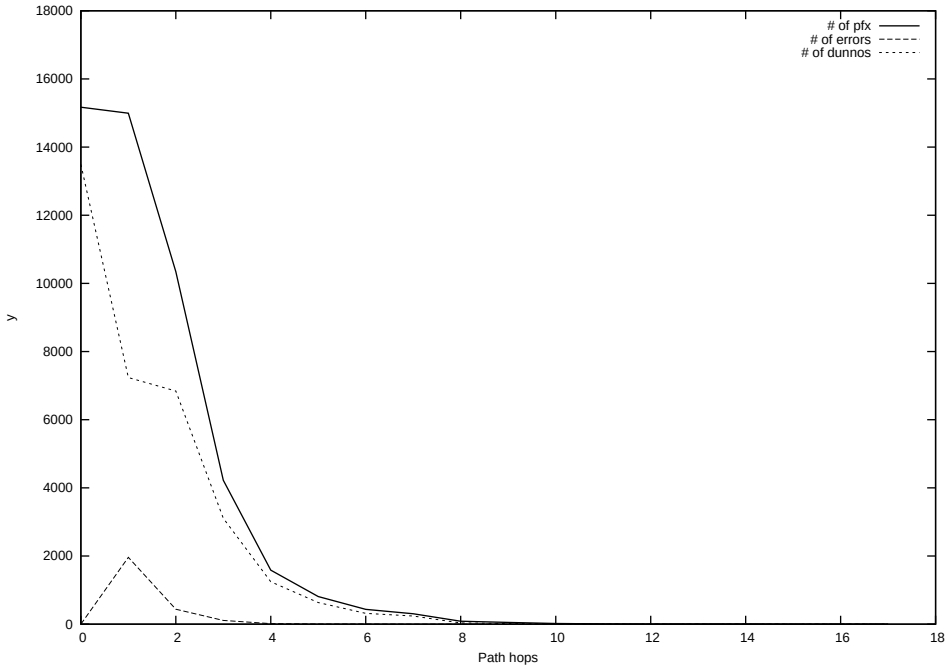
2013-10-25



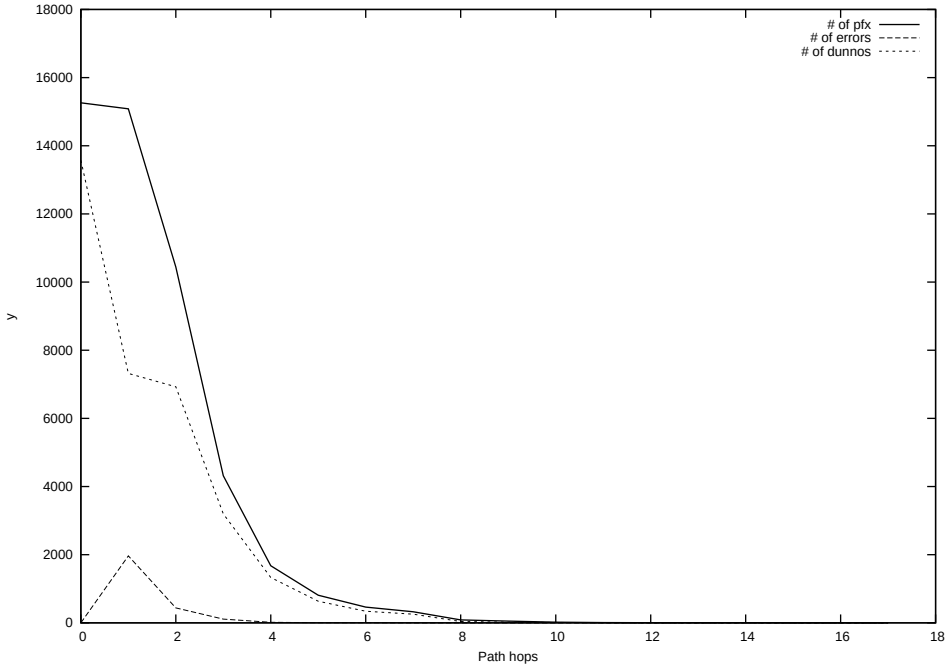
2013-10-26



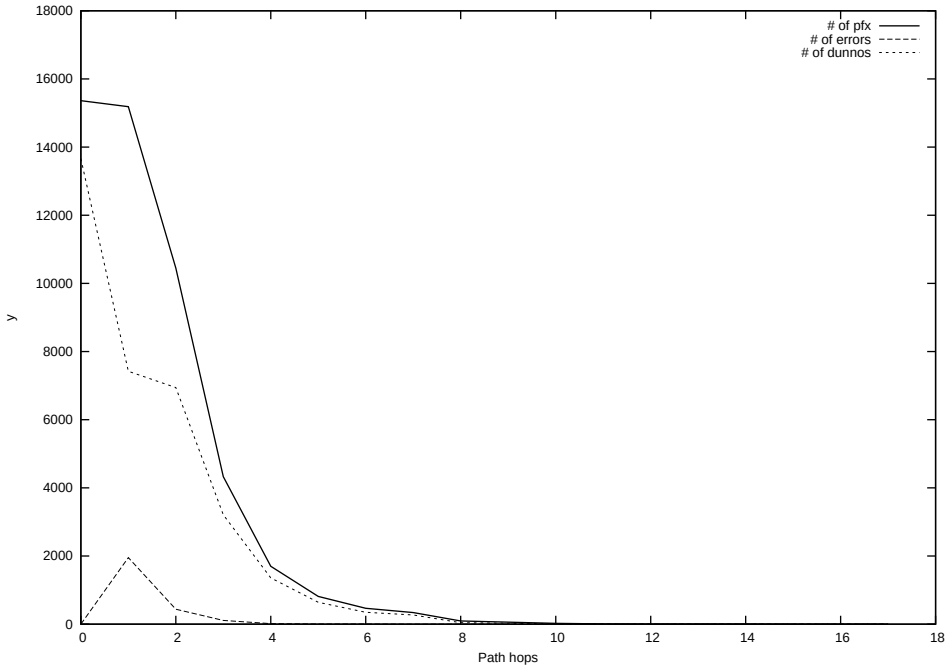
2013-10-27



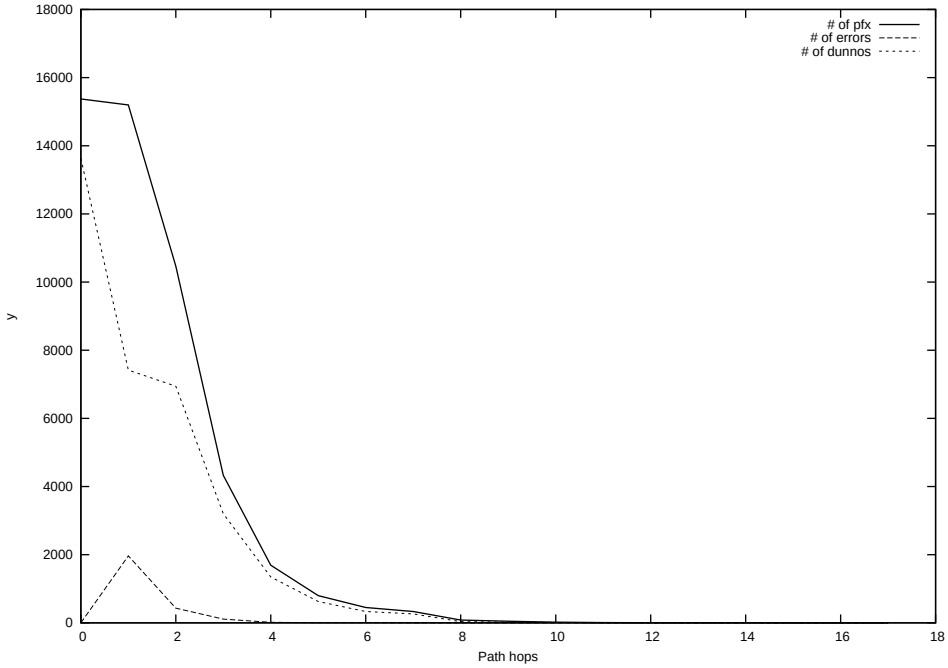
2013-10-28



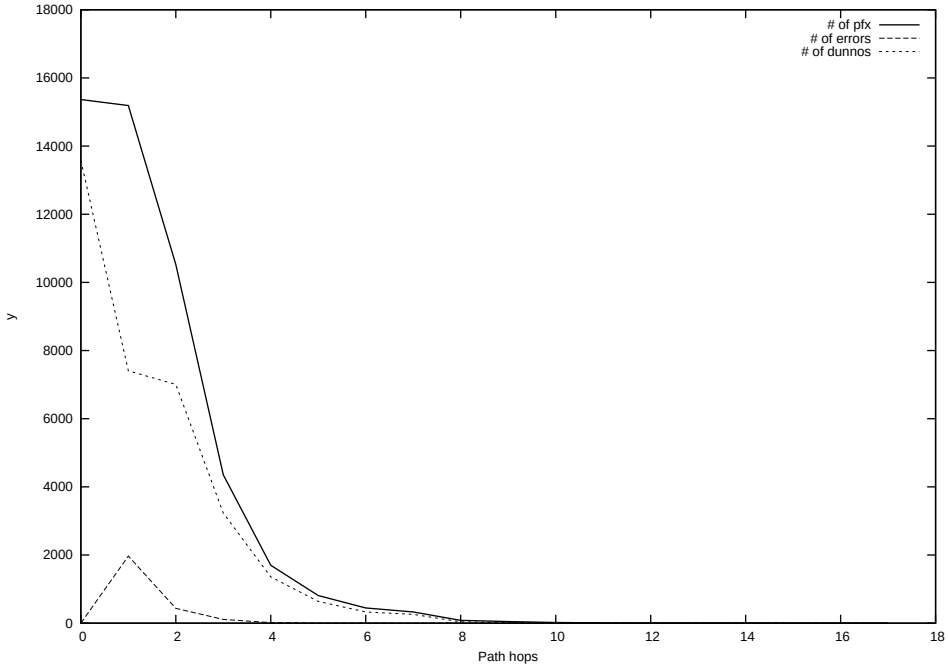
2013-10-29



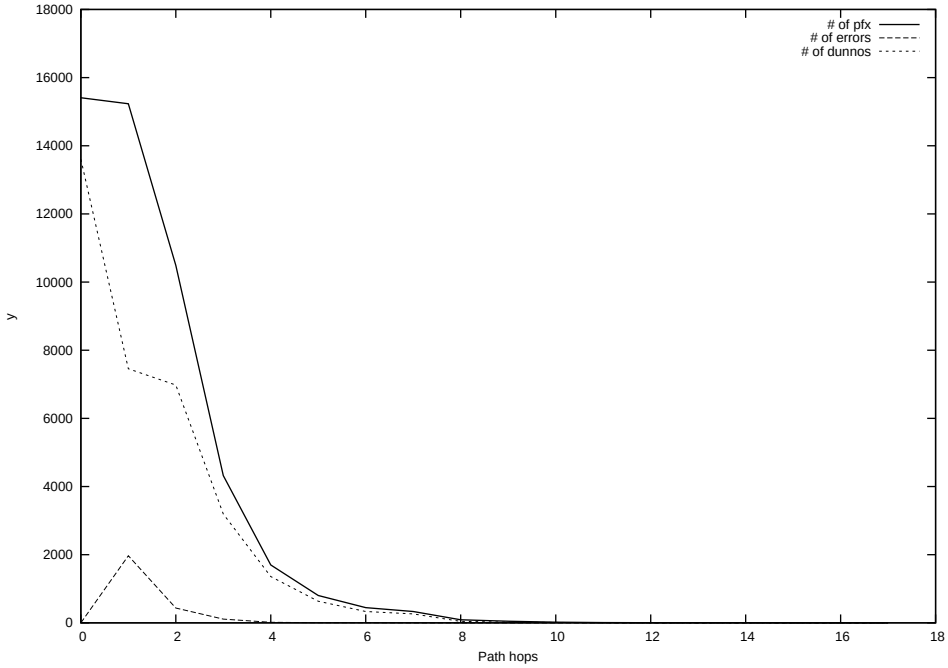
2013-10-30



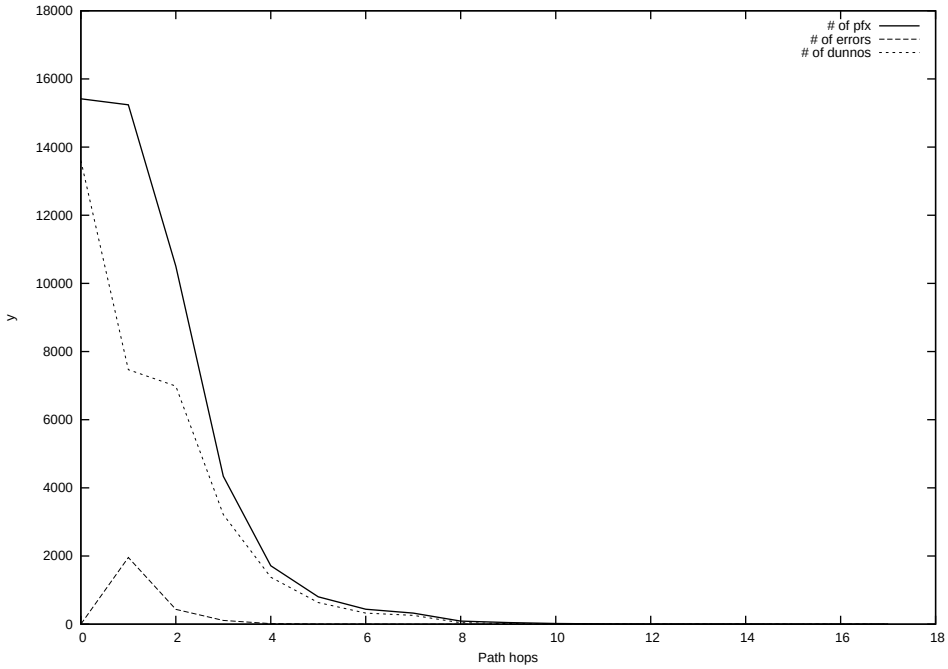
2013-10-31



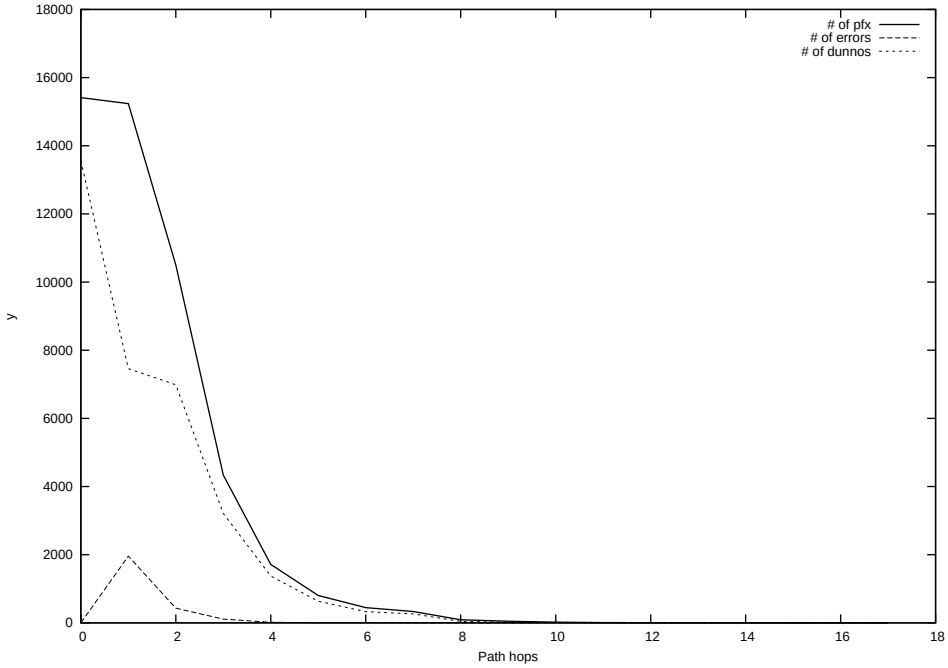
2013-11-01



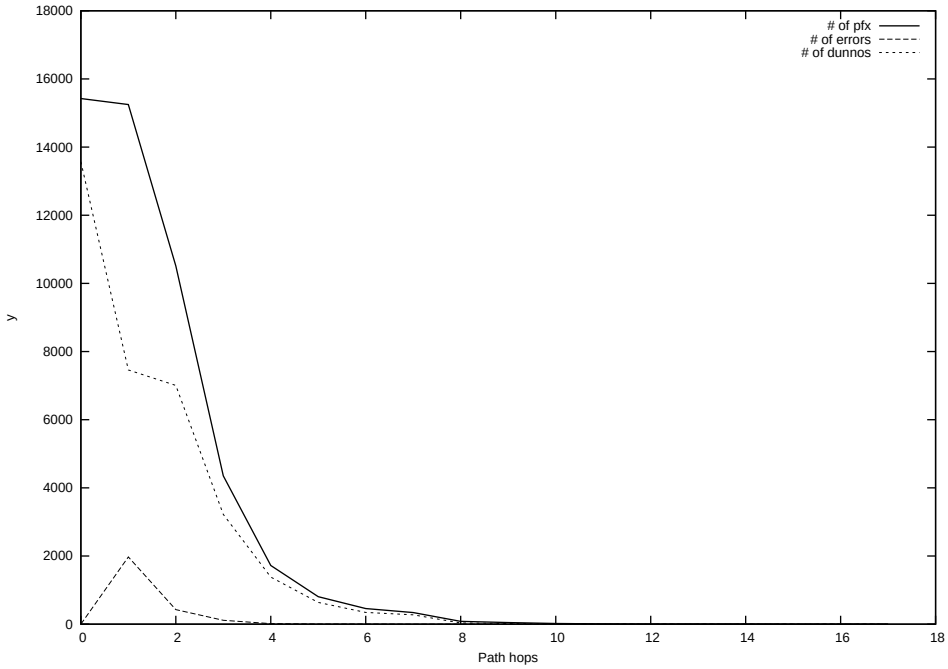
2013-11-02



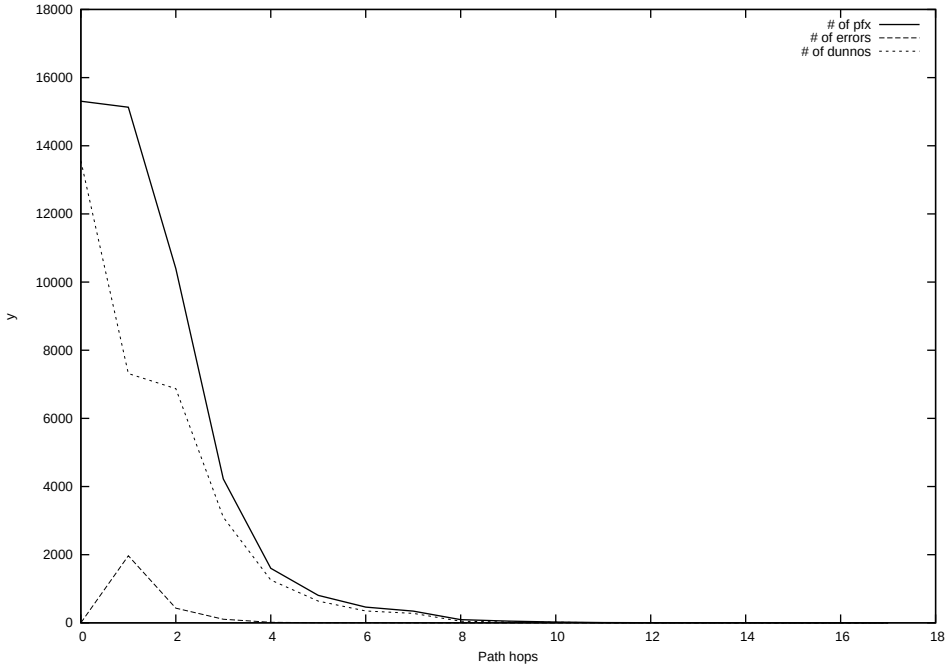
2013-11-03



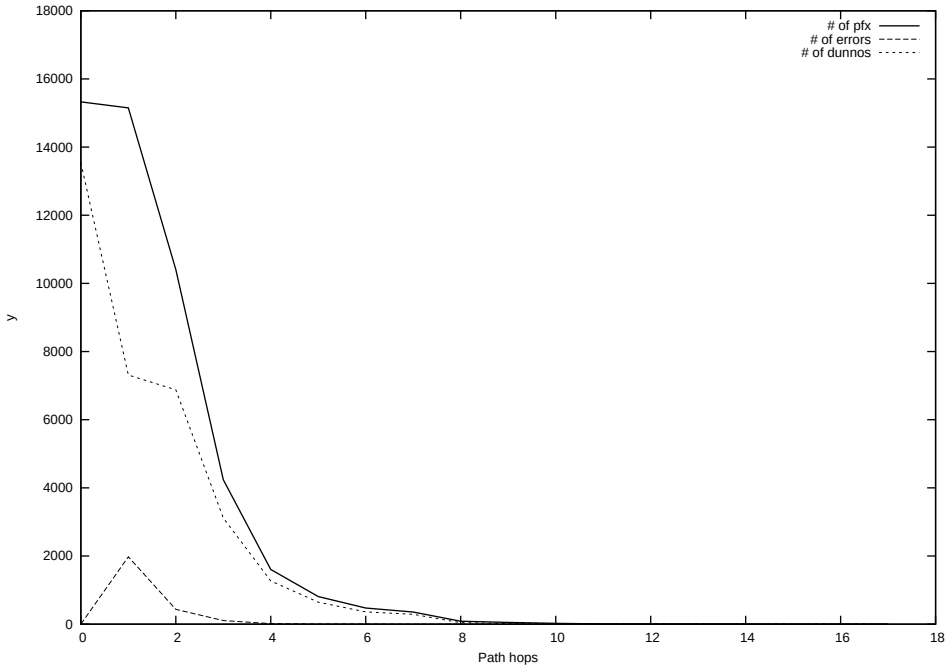
2013-11-04



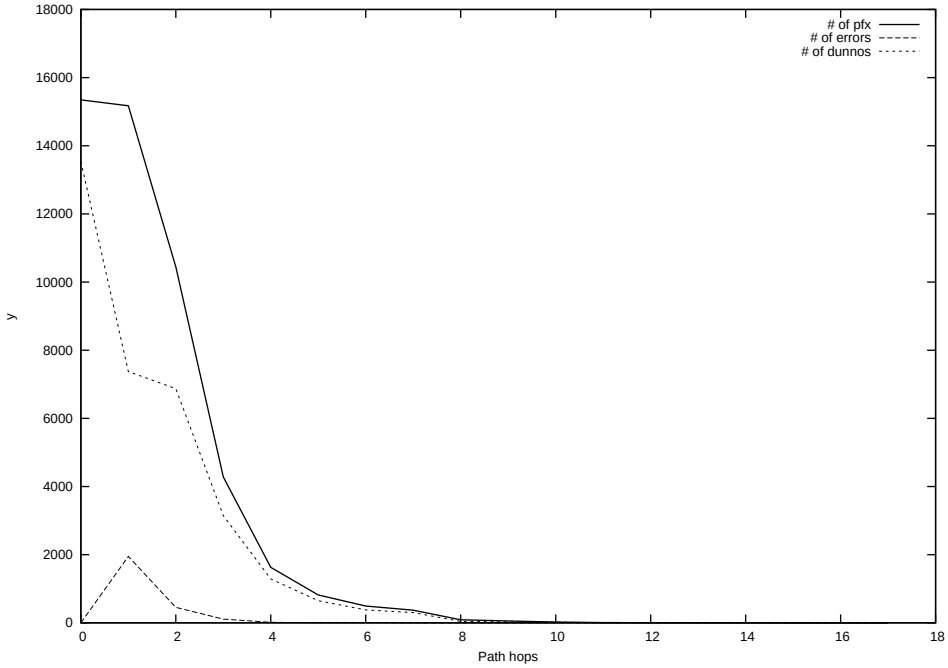
2013-11-05



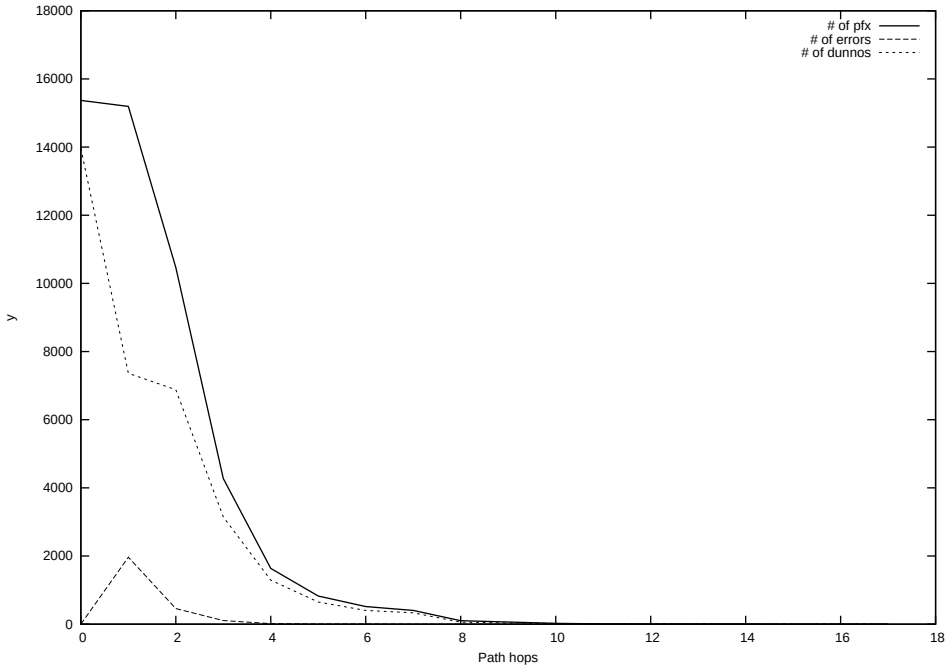
2013-11-06



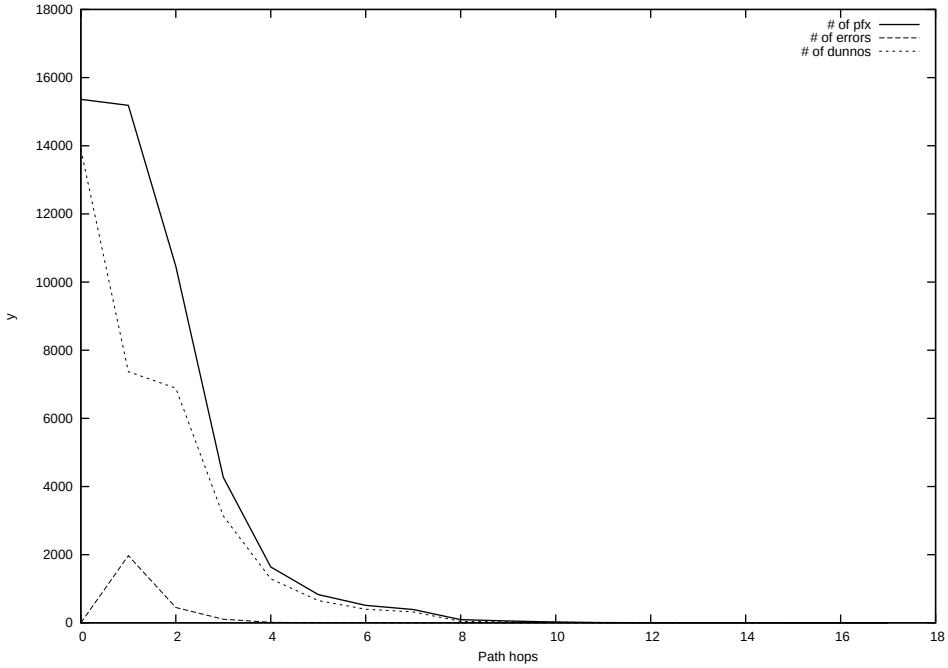
2013-11-07



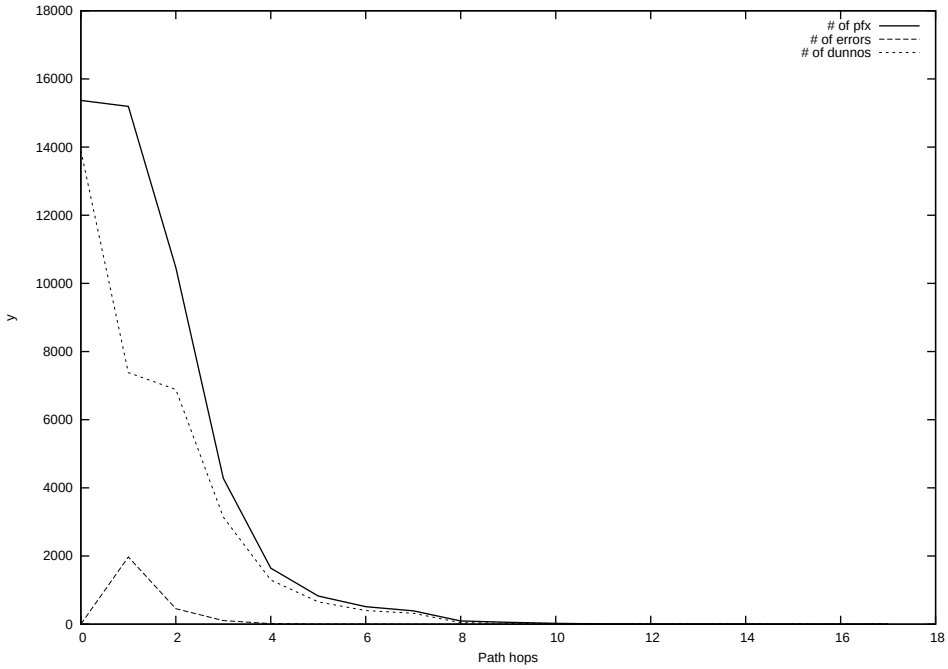
2013-11-08



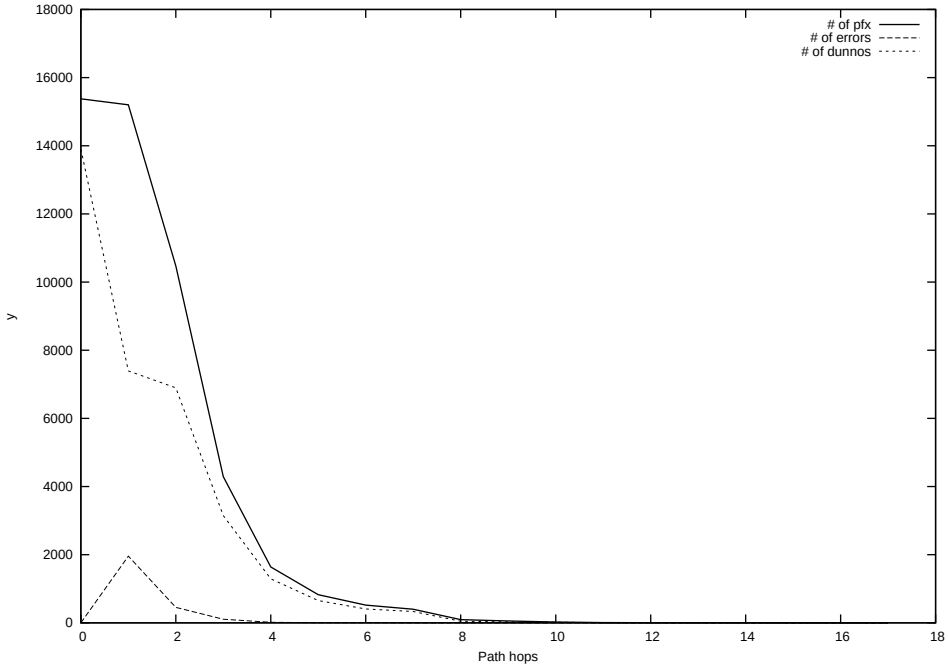
2013-11-09



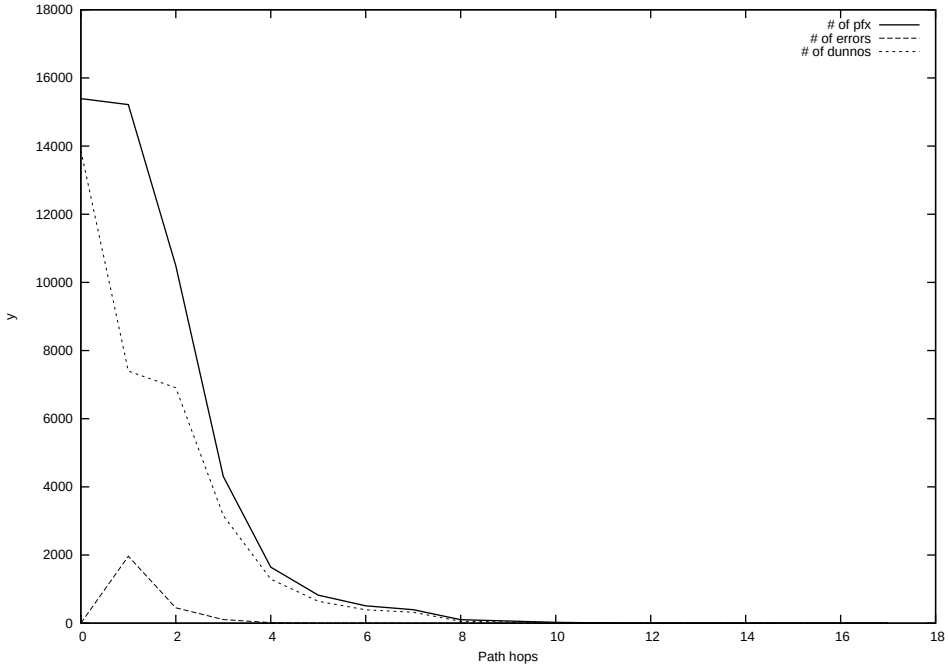
2013-11-10



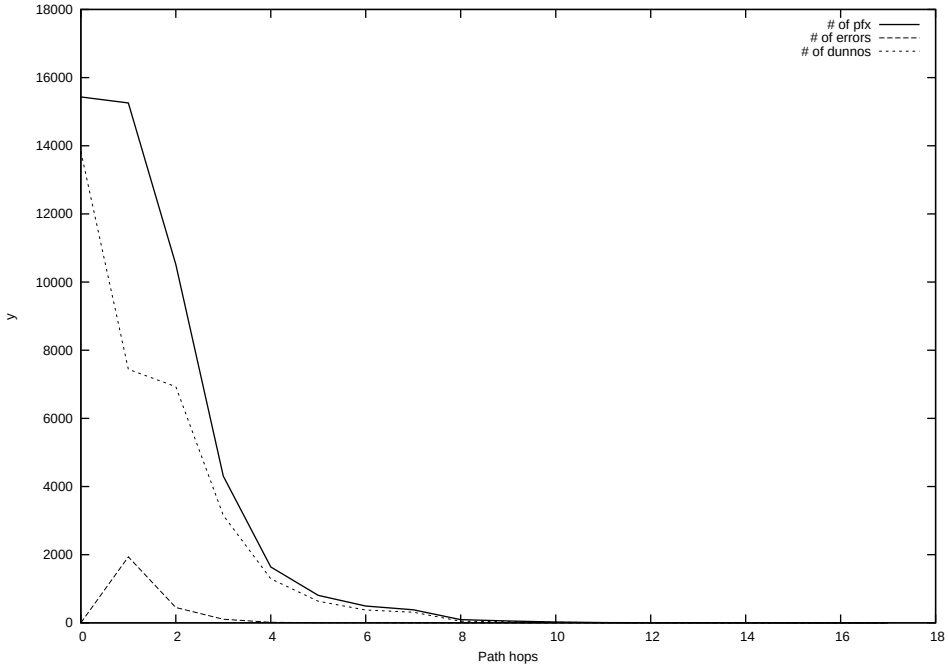
2013-11-11



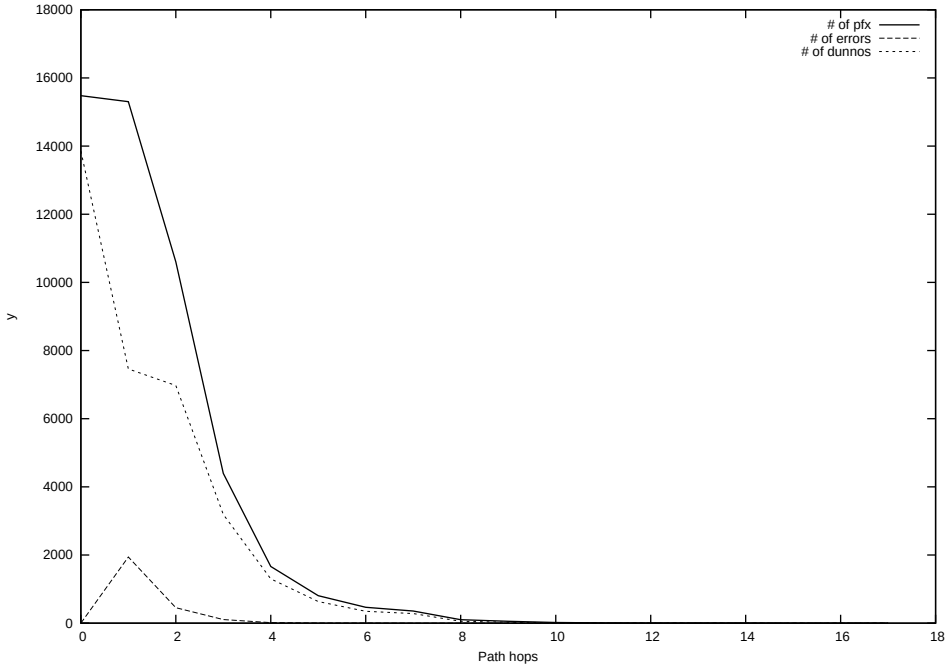
2013-11-12



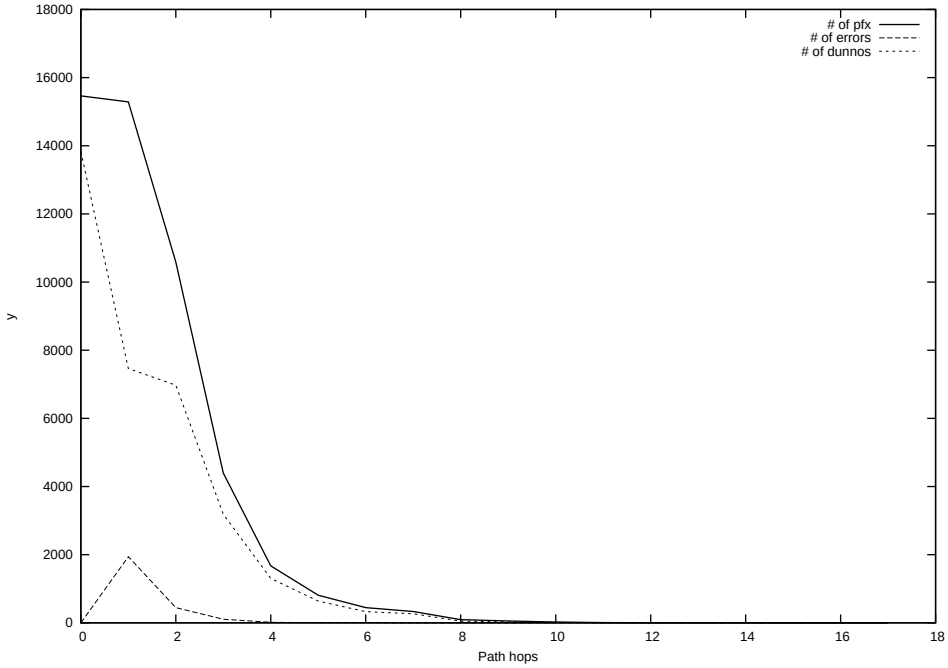
2013-11-13



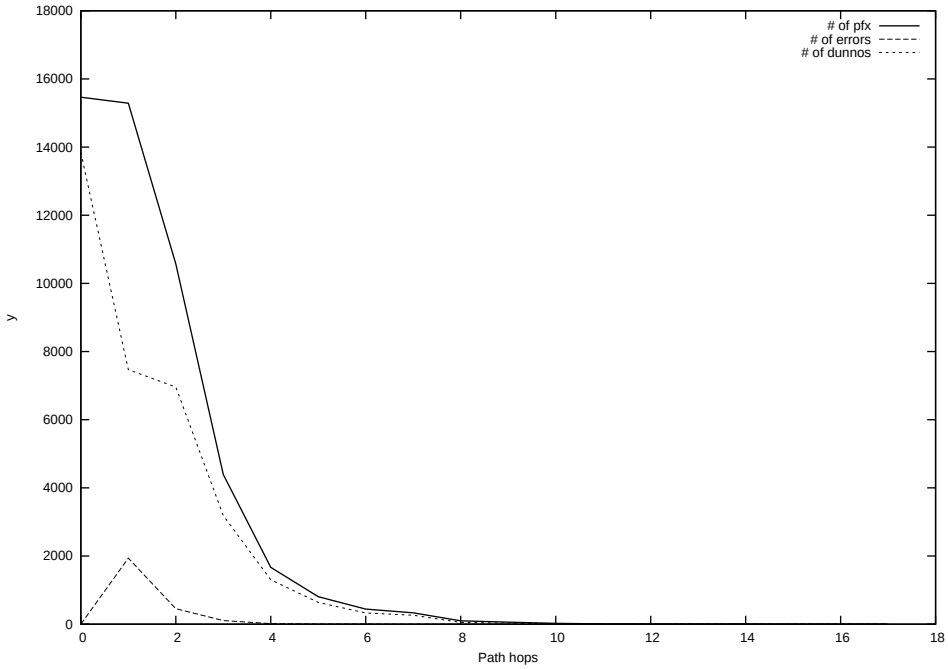
2013-11-14



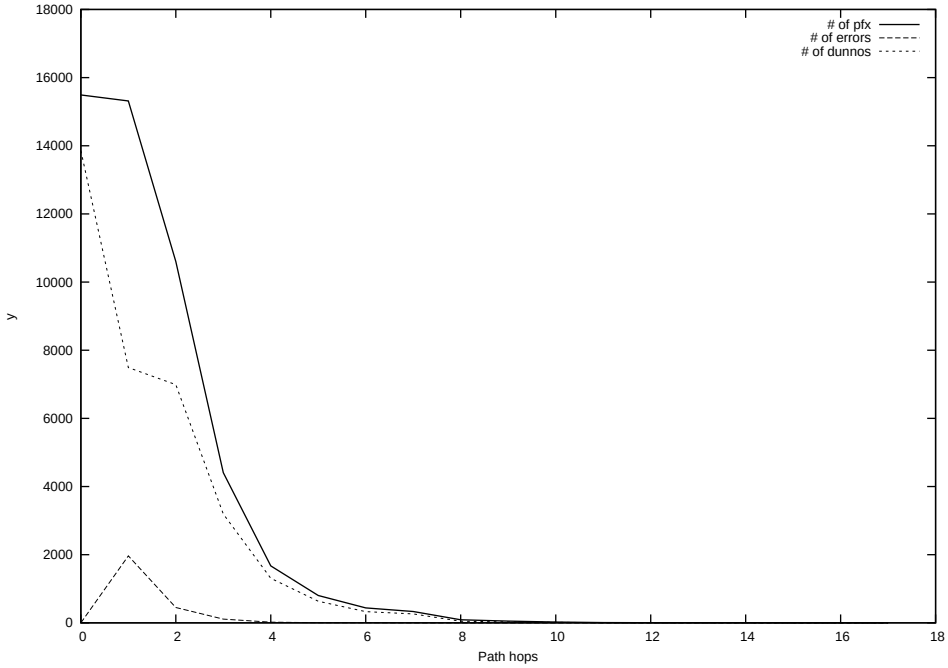
2013-11-15



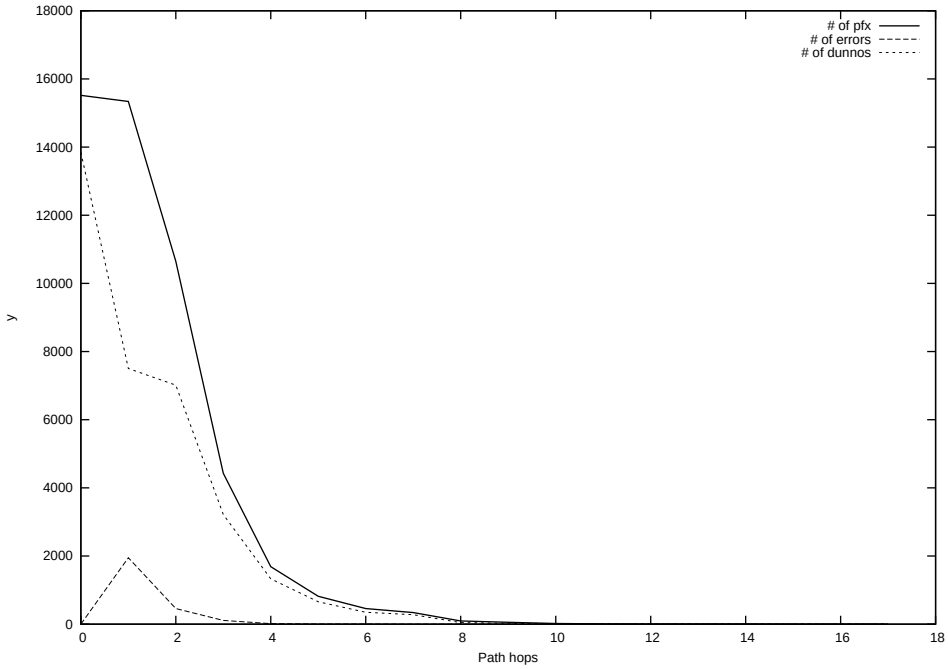
2013-11-16



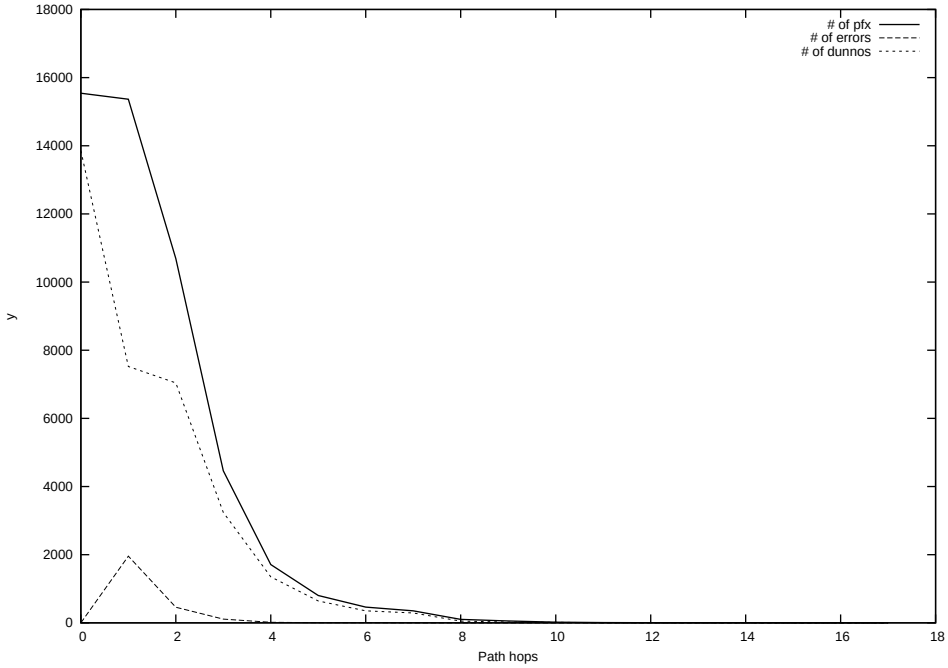
2013-11-17



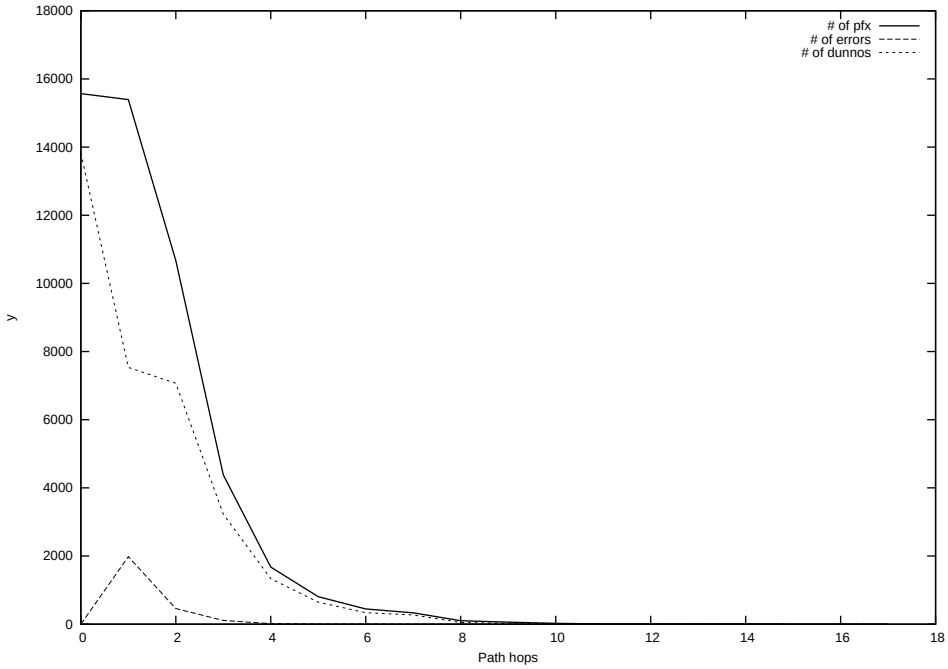
2013-11-18



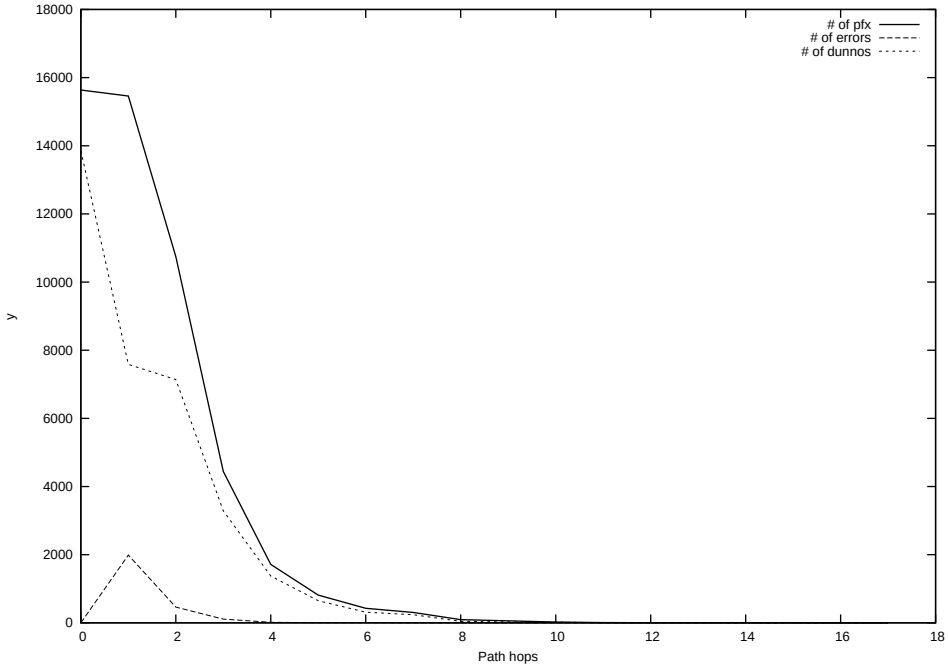
2013-11-19



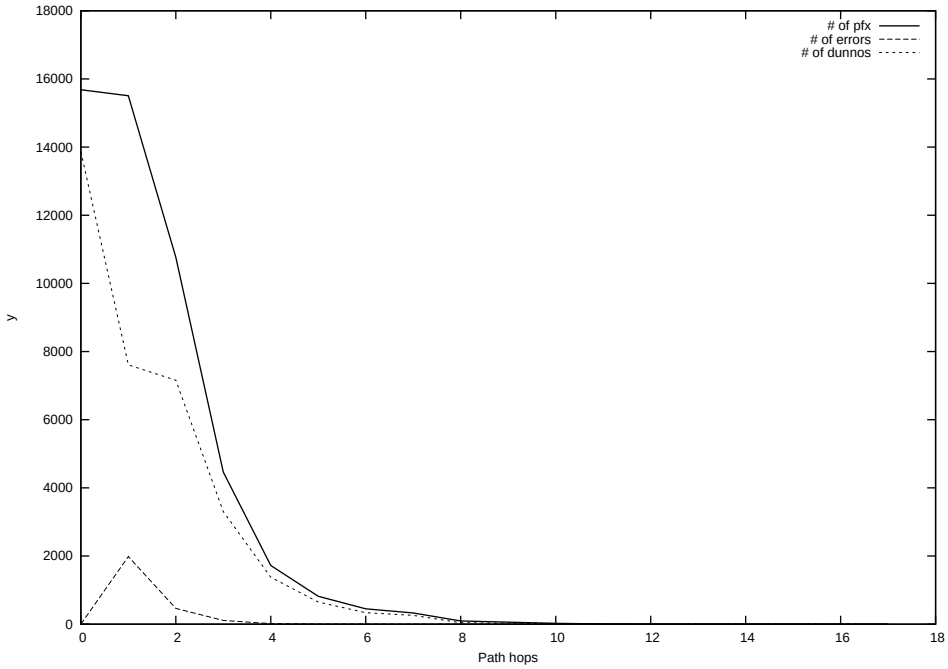
2013-11-20



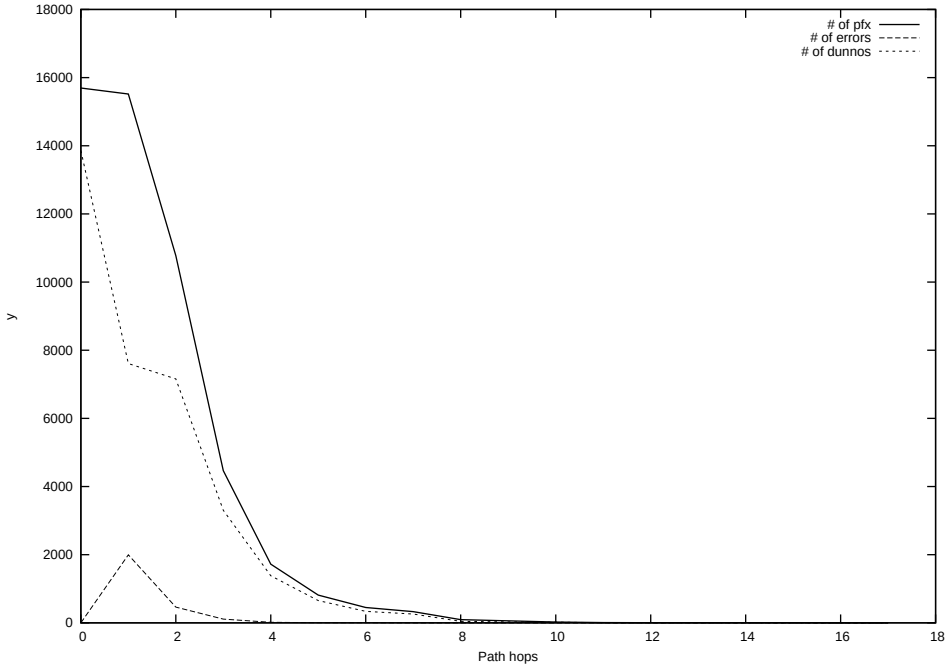
2013-11-21



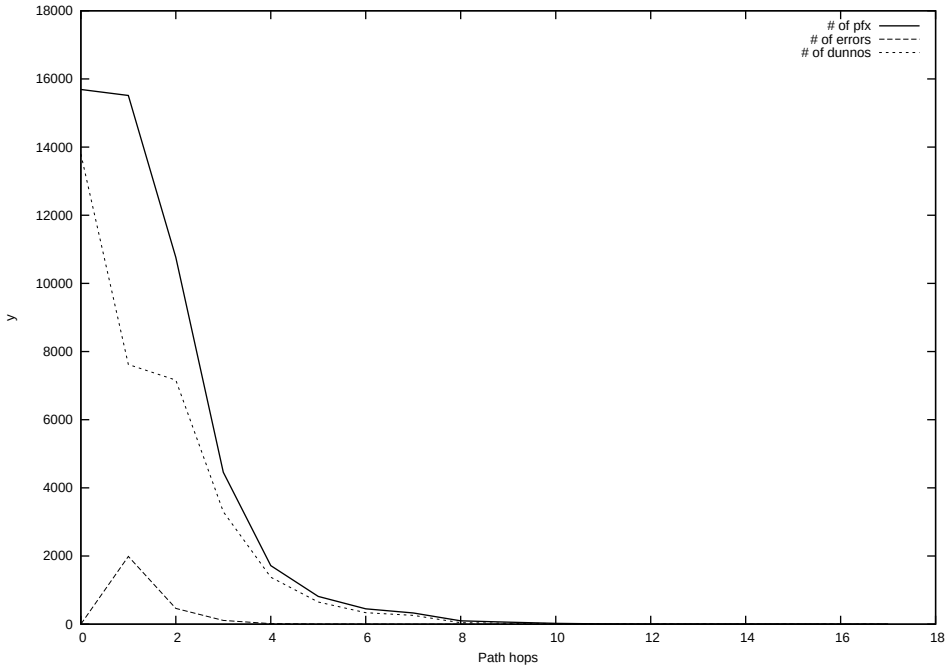
2013-11-22



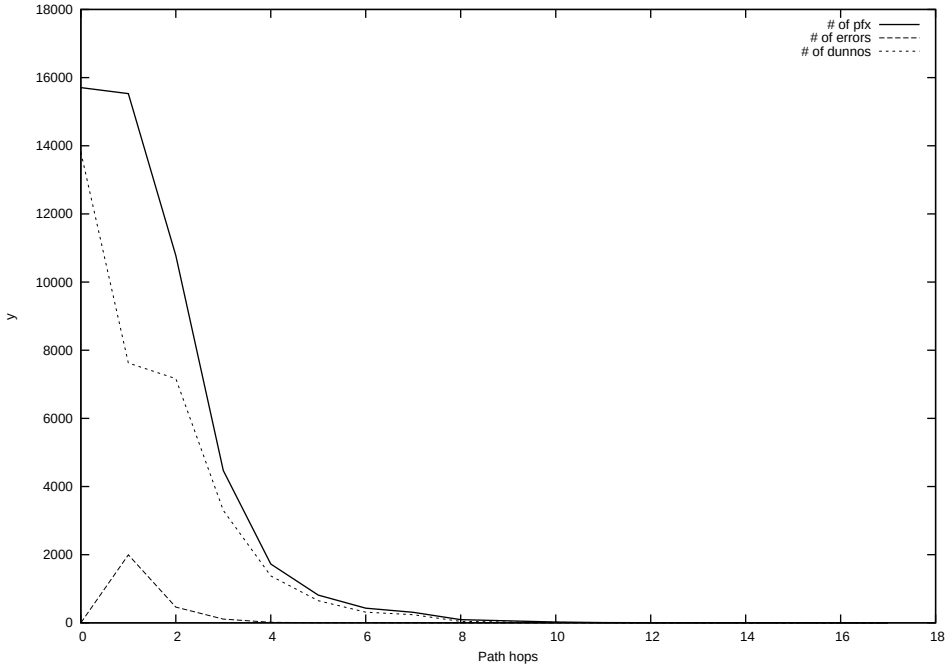
2013-11-23



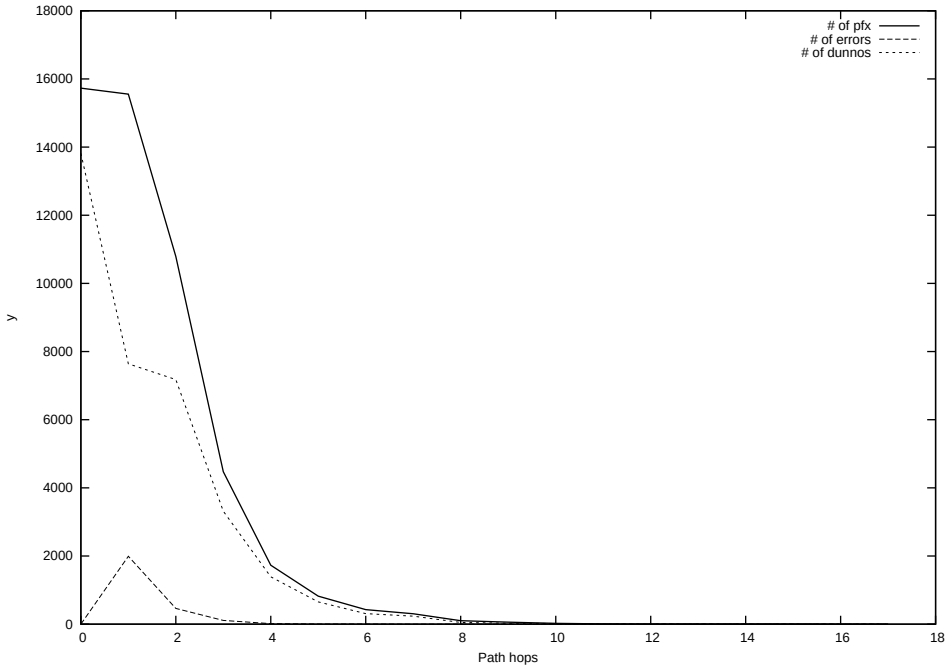
2013-11-24



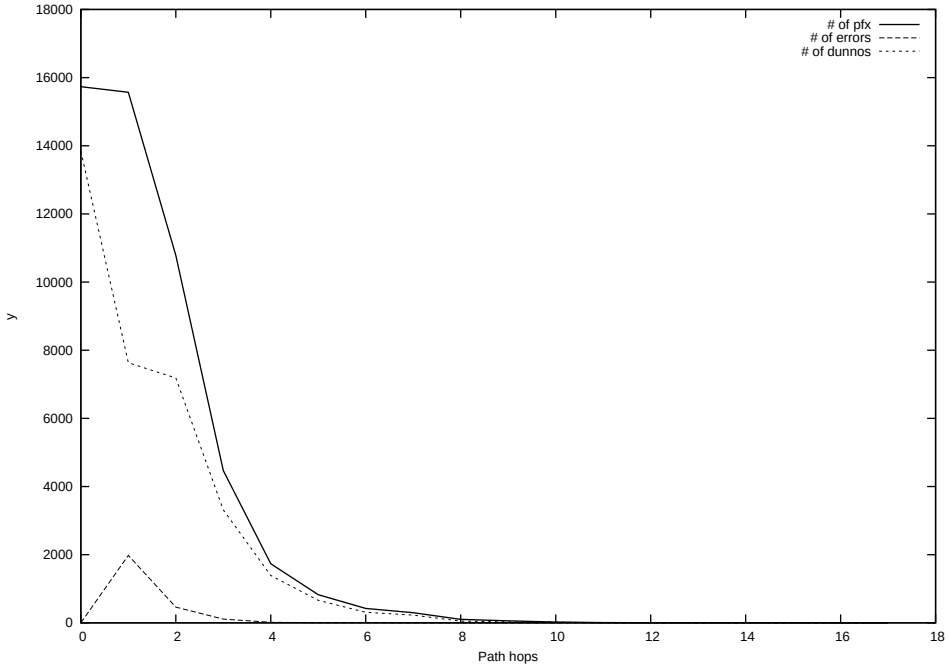
2013-11-25



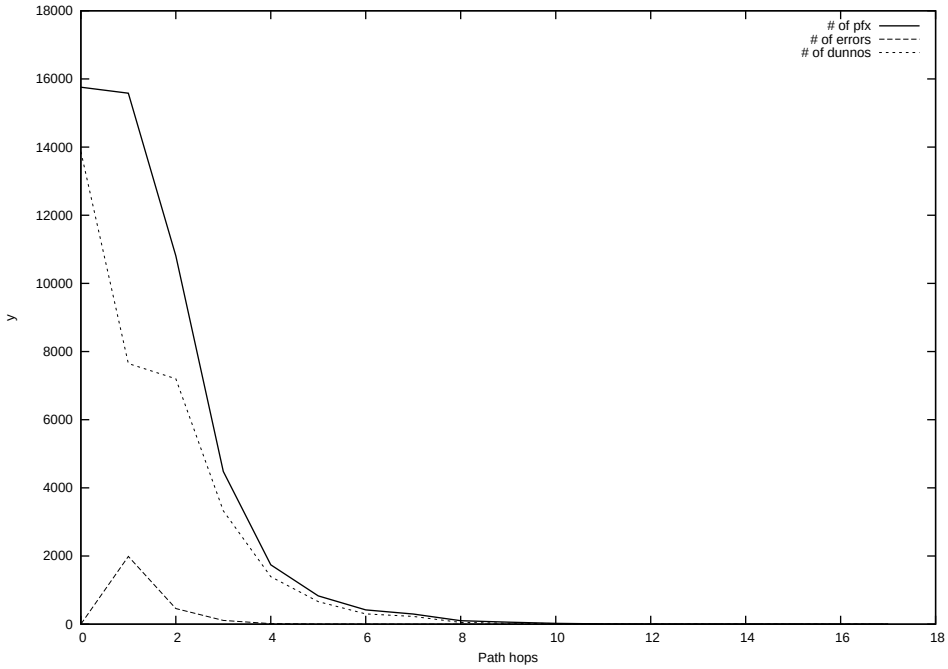
2013-11-26



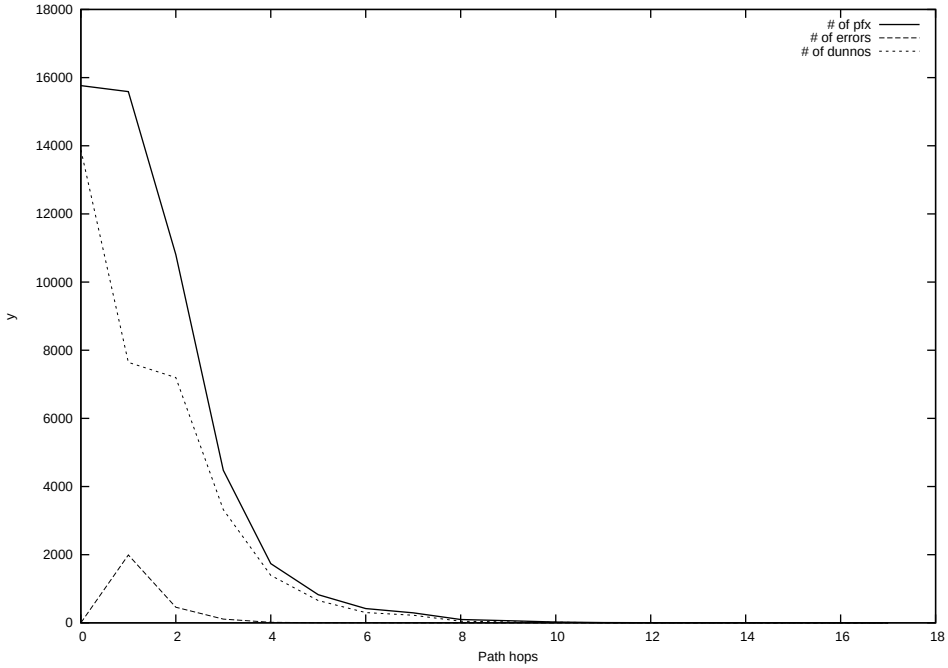
2013-11-27



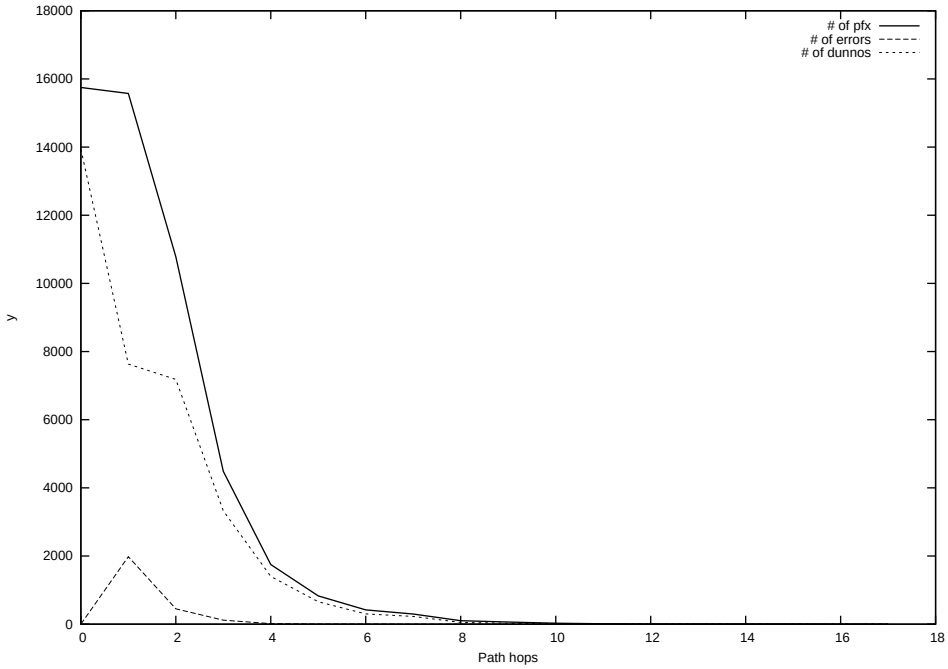
2013-11-28



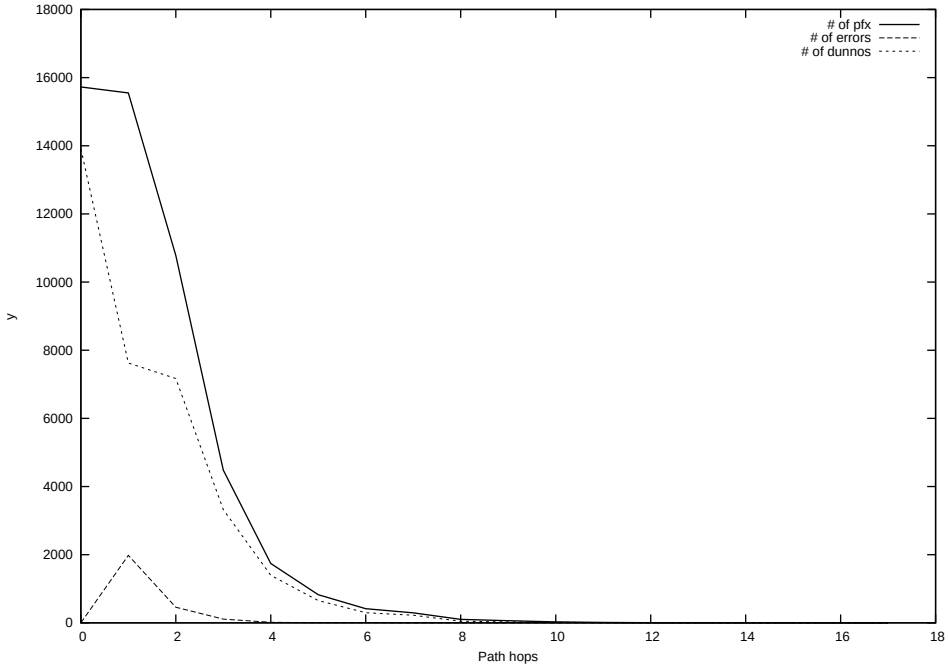
2013-11-29



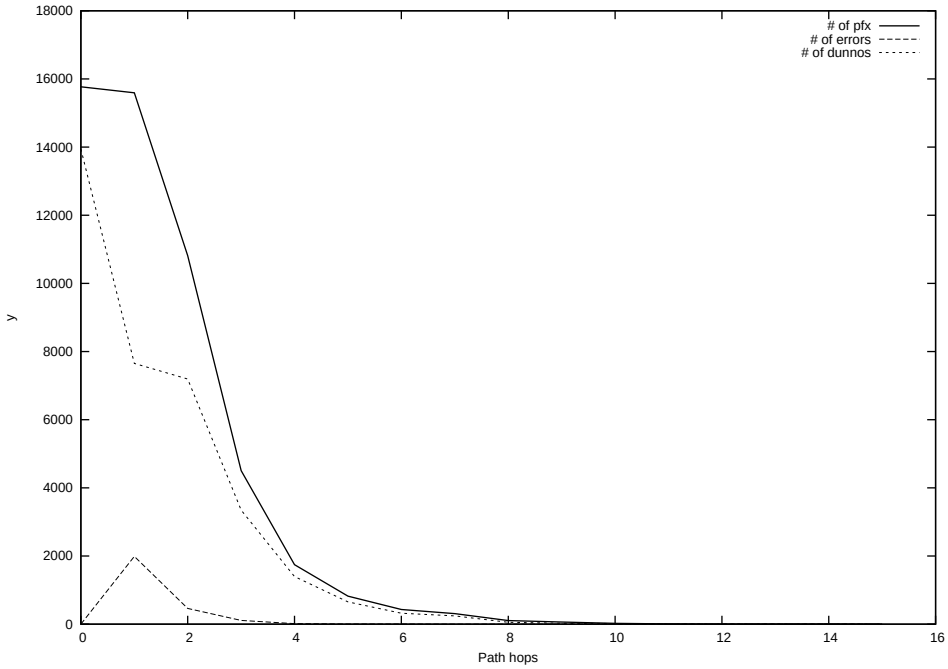
2013-11-30



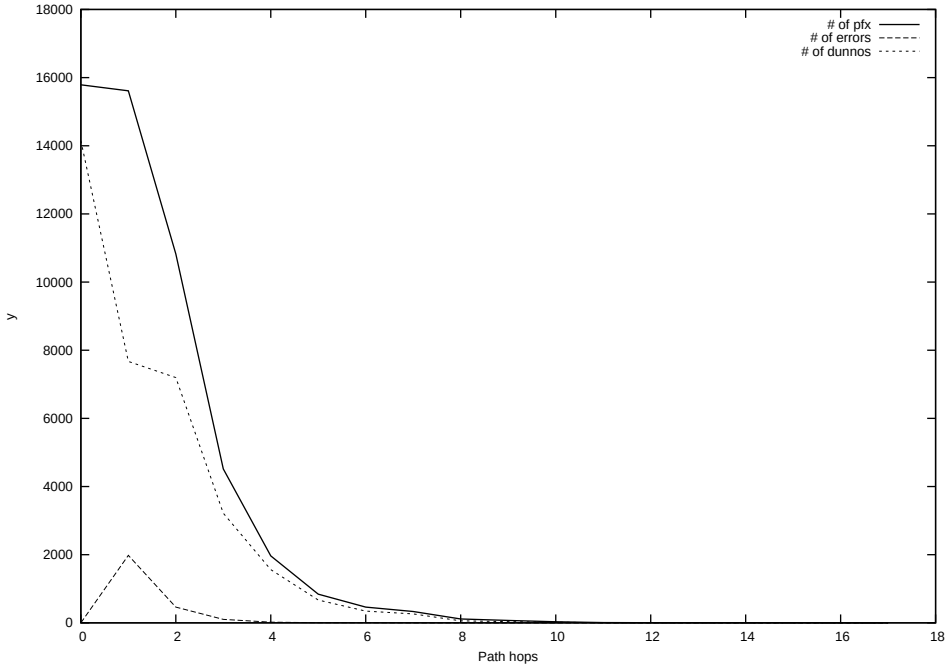
2013-12-01



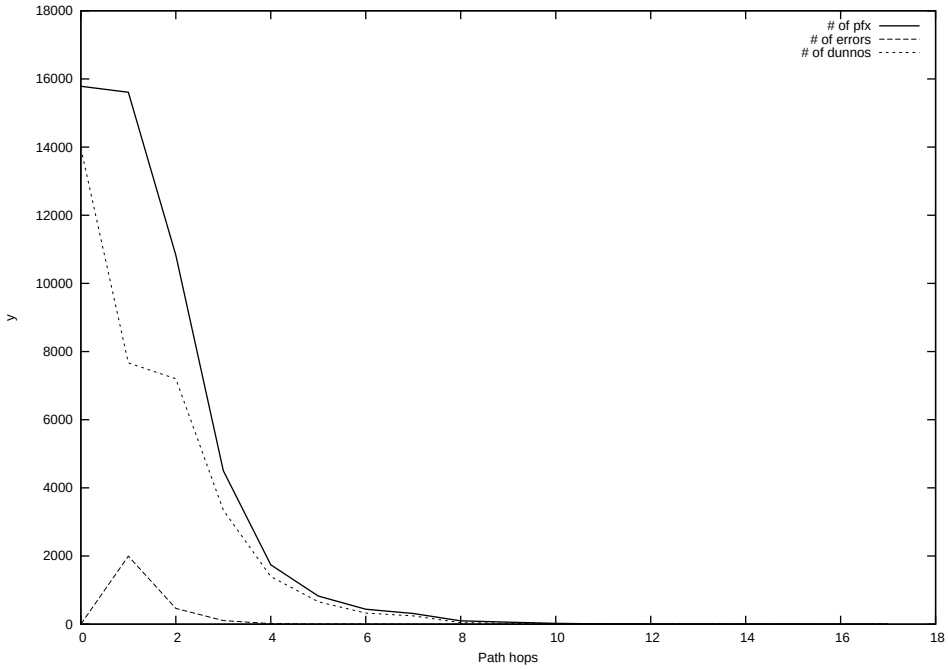
2013-12-02



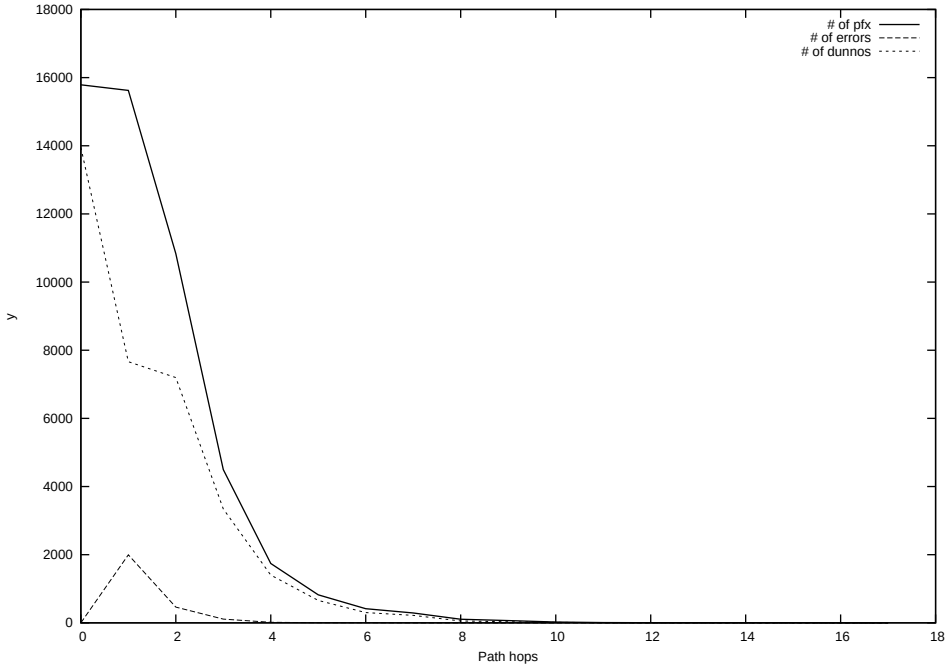
2013-12-03



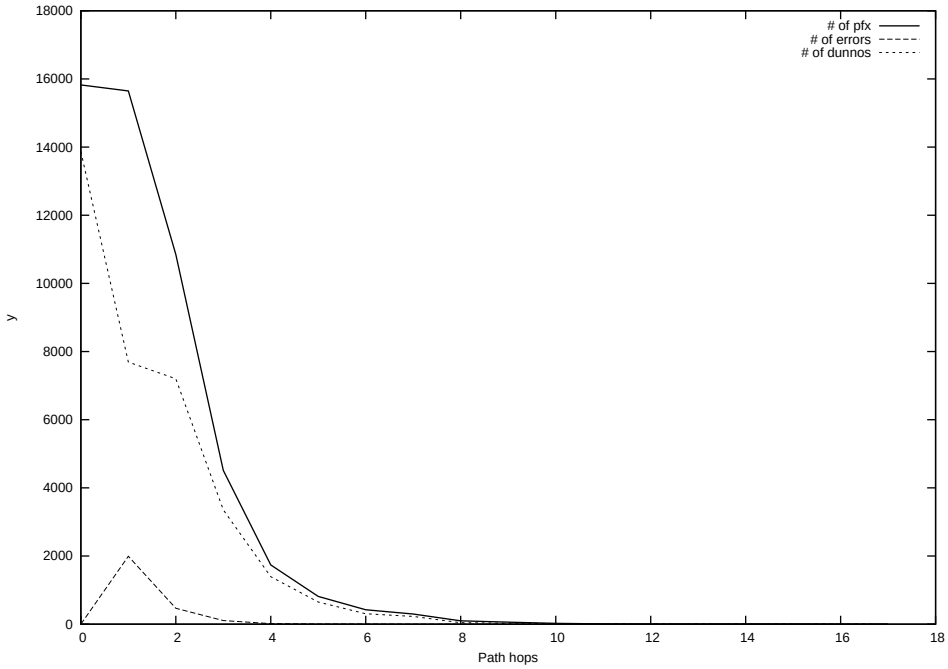
2013-12-04



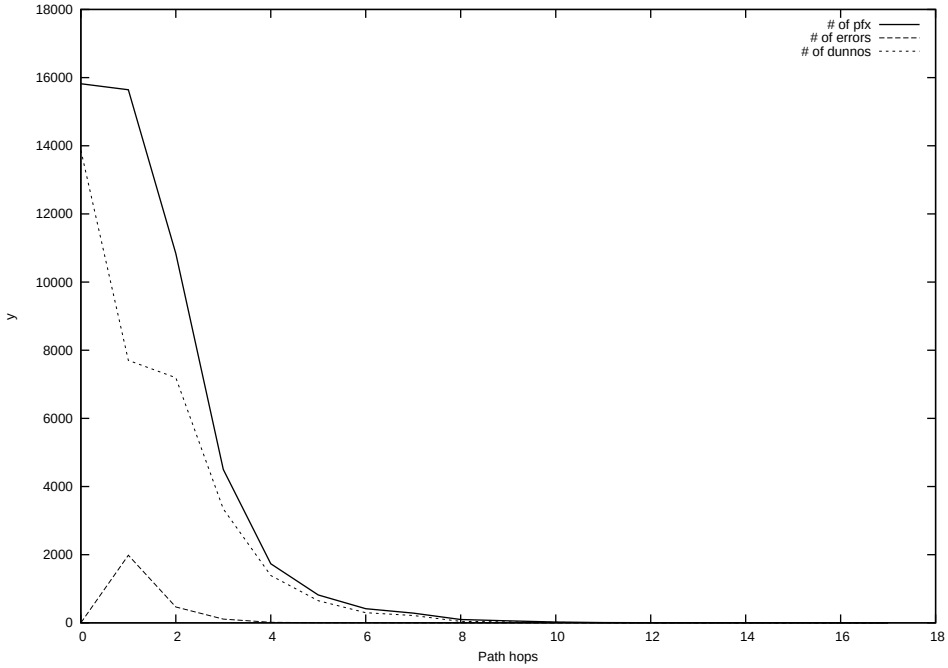
2013-12-05



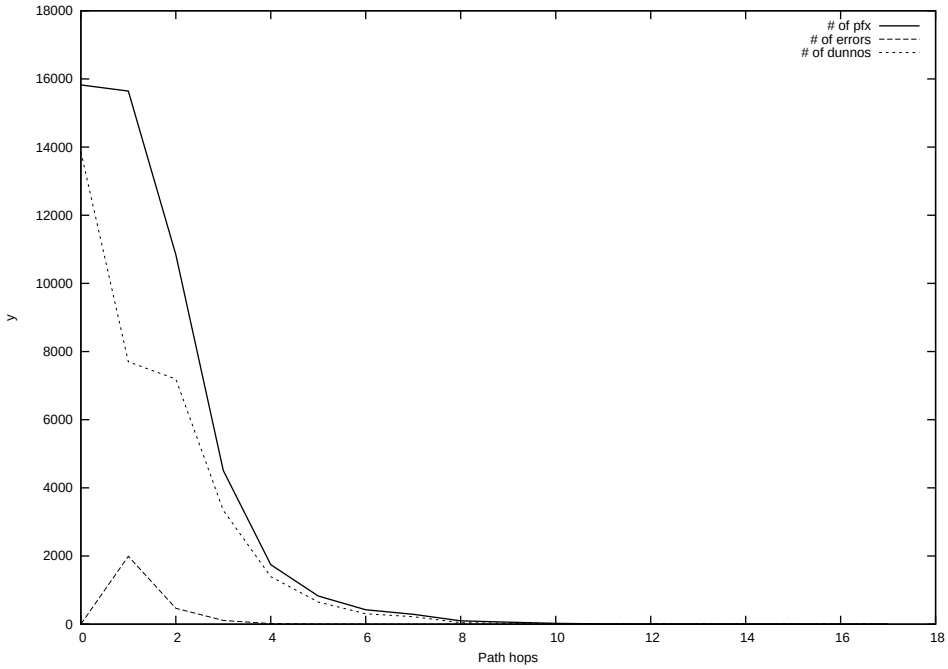
2013-12-06



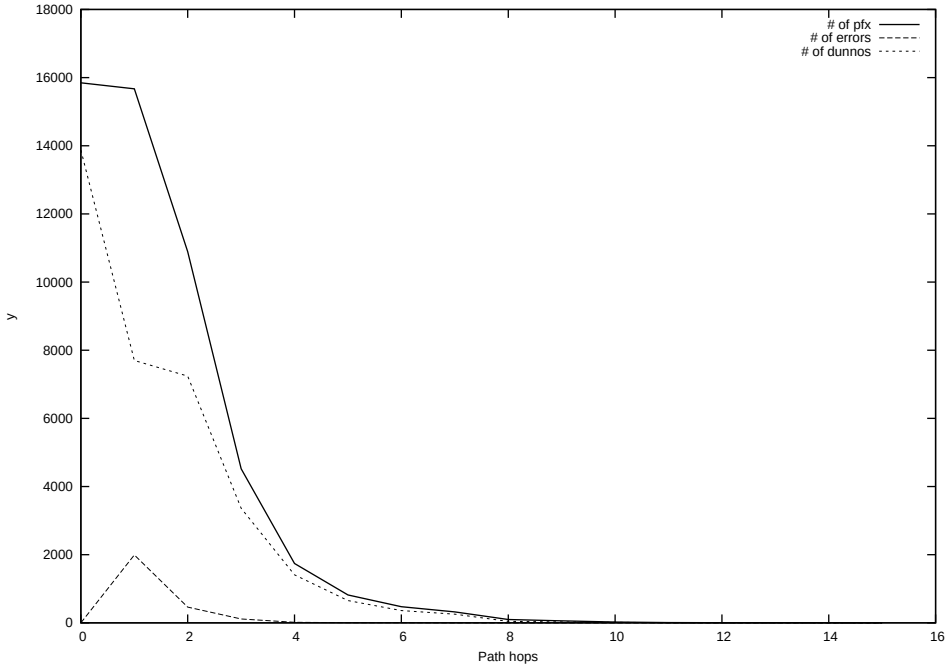
2013-12-07



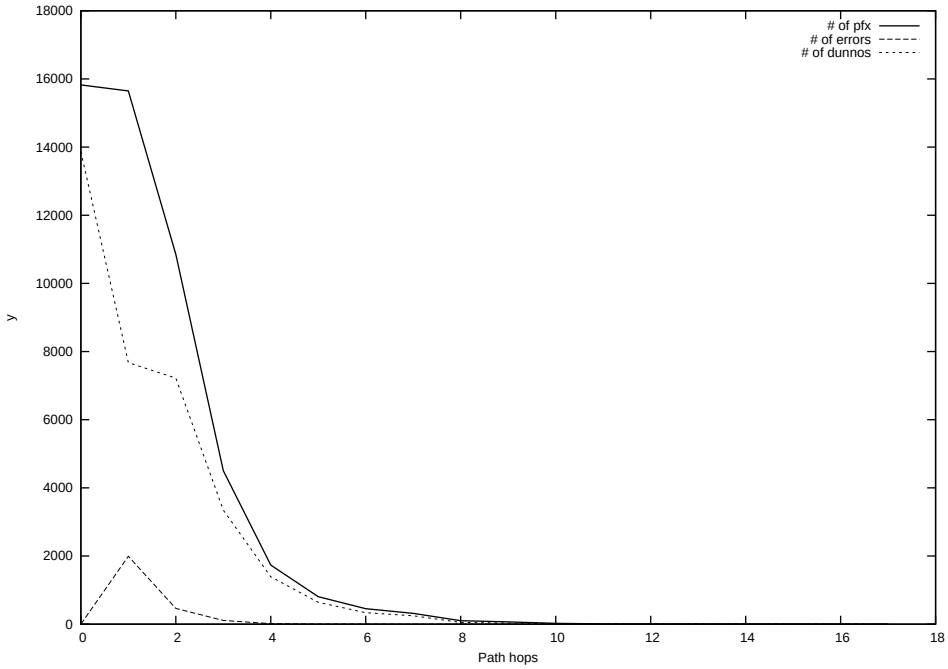
2013-12-08



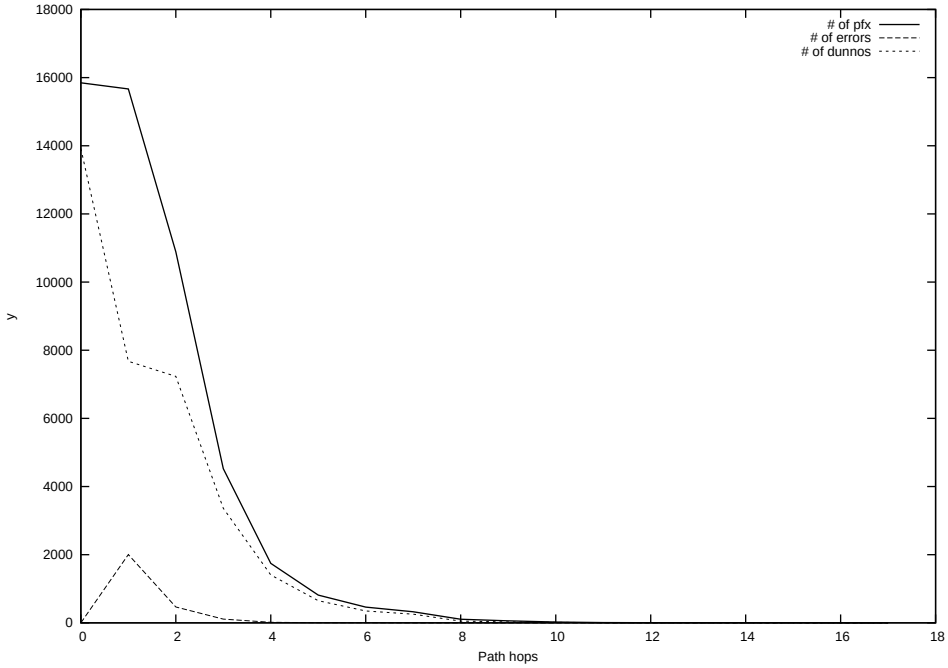
2013-12-09



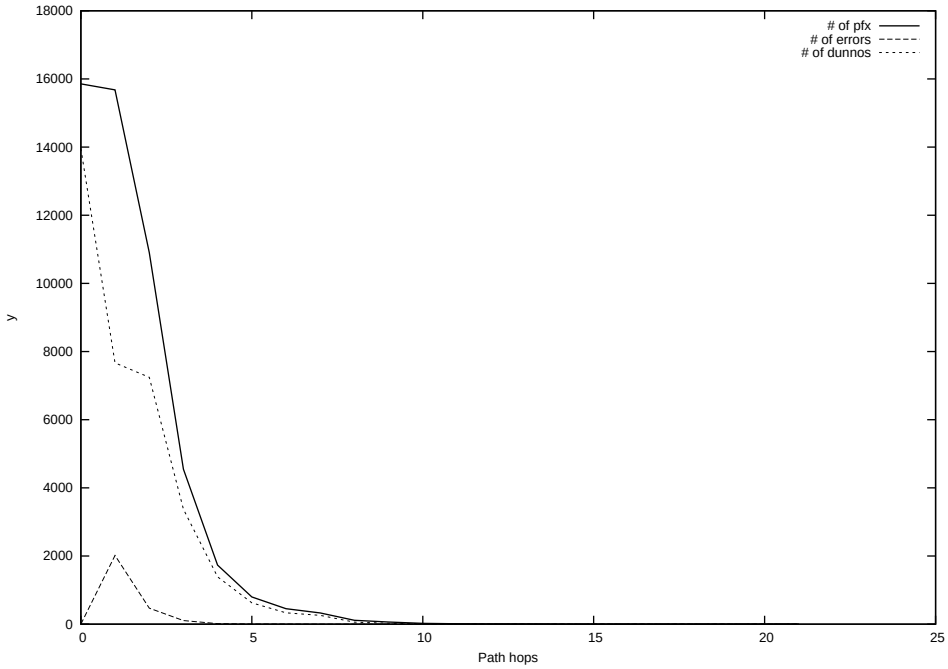
2013-12-10



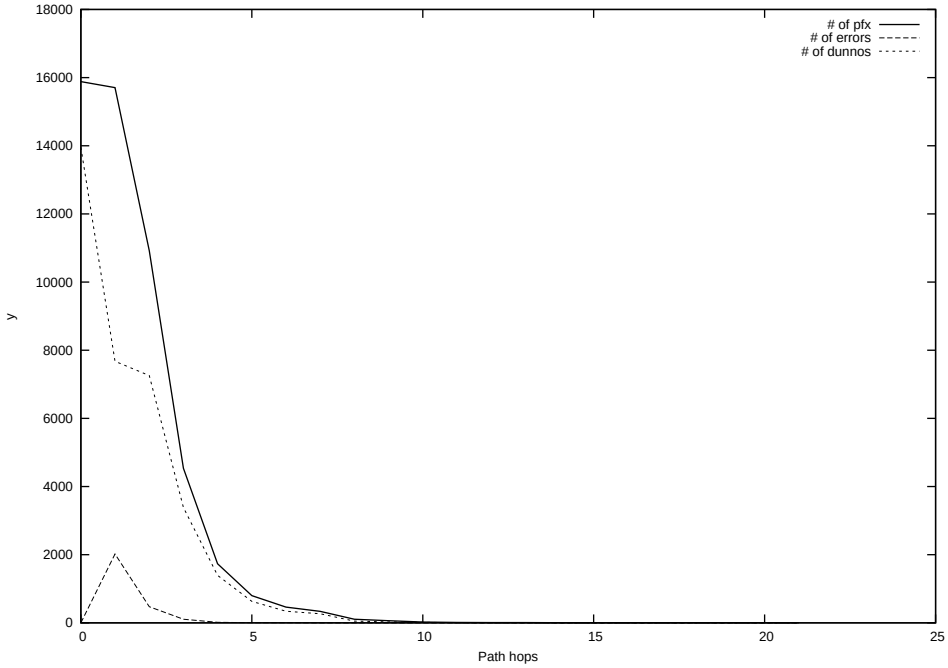
2013-12-11



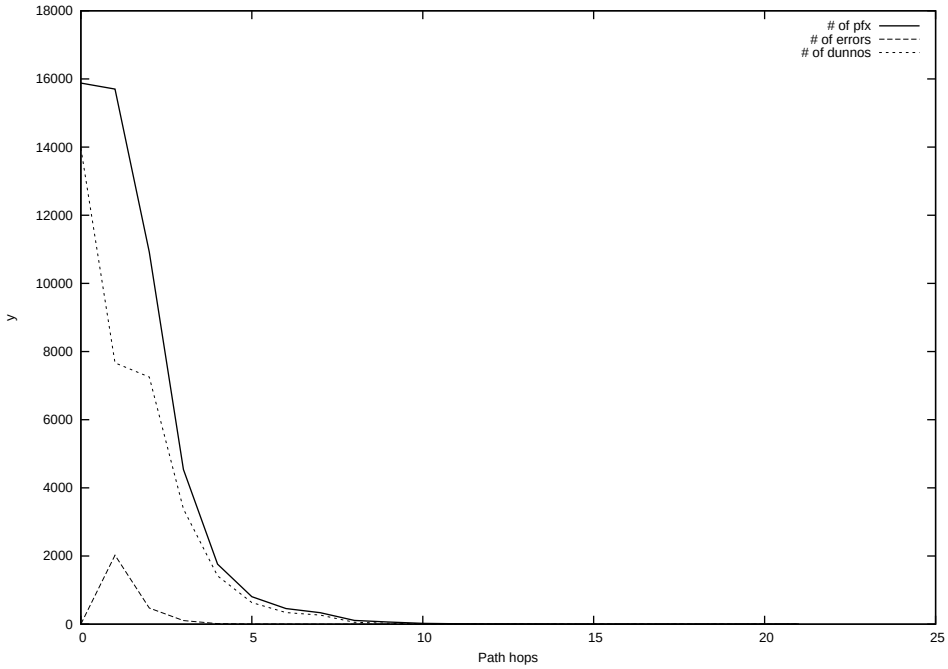
2013-12-12



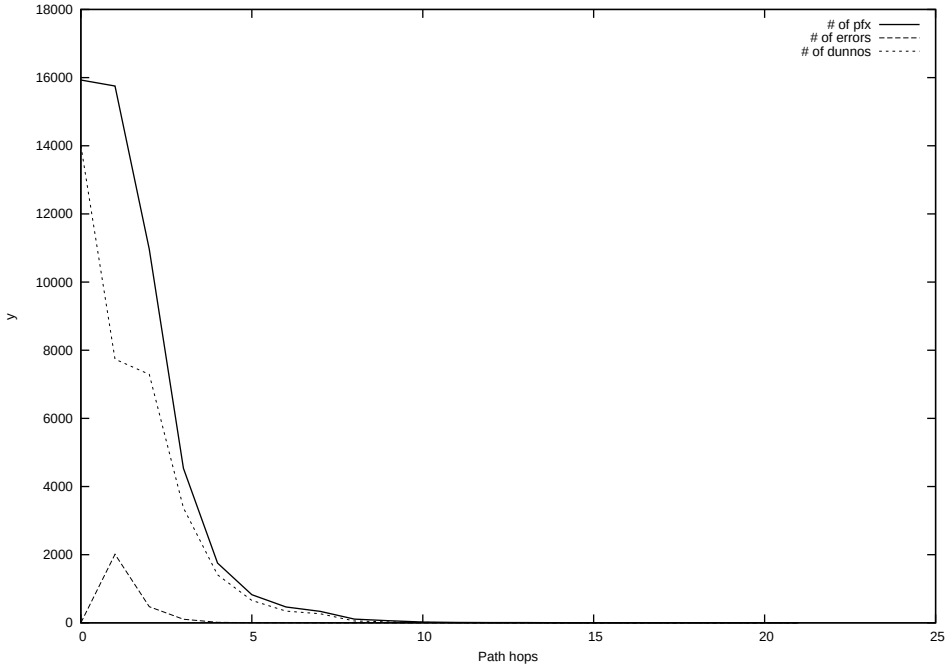
2013-12-13



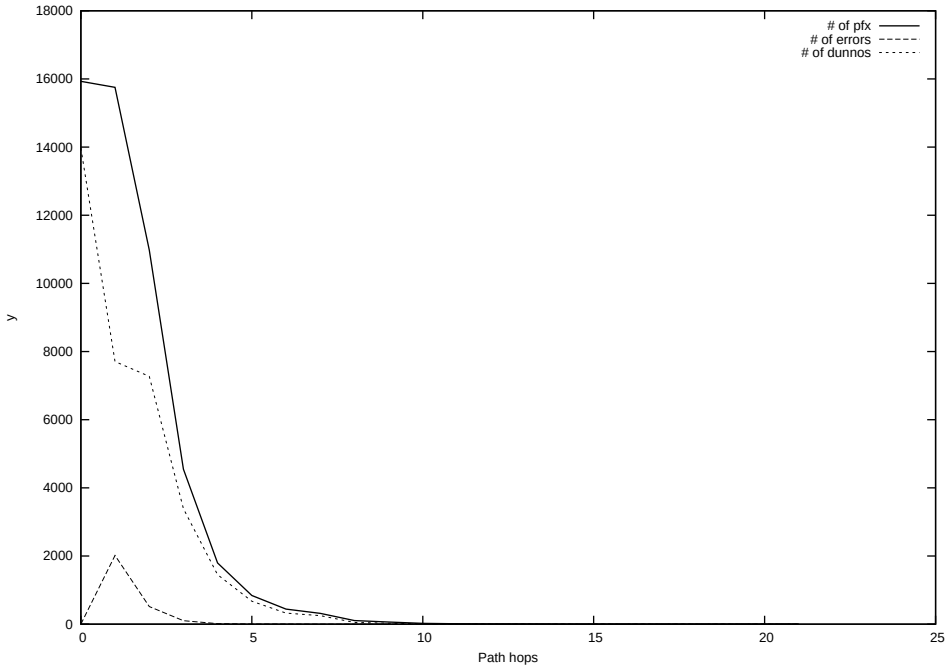
2013-12-14



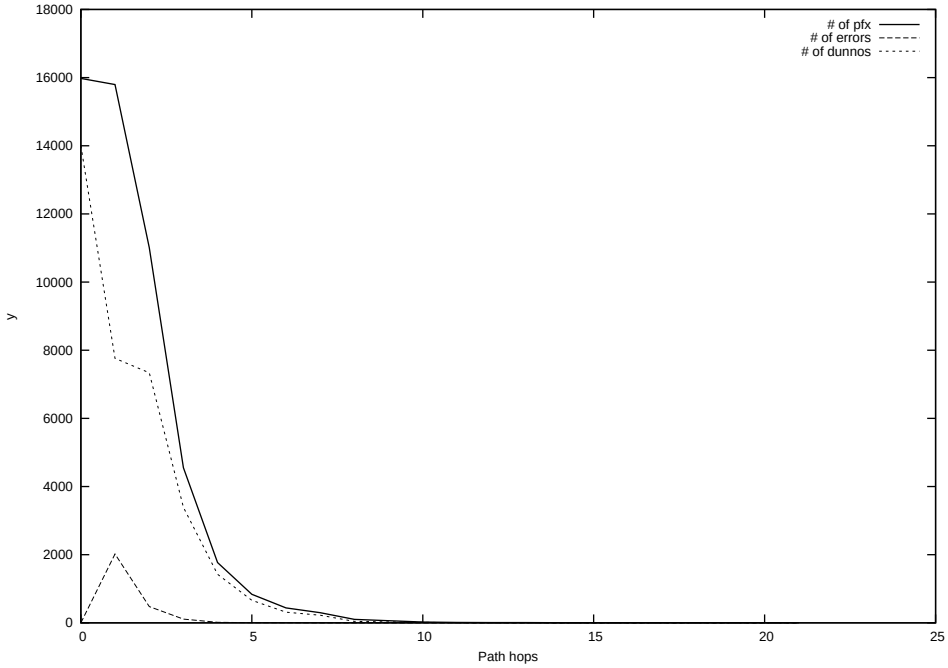
2013-12-15



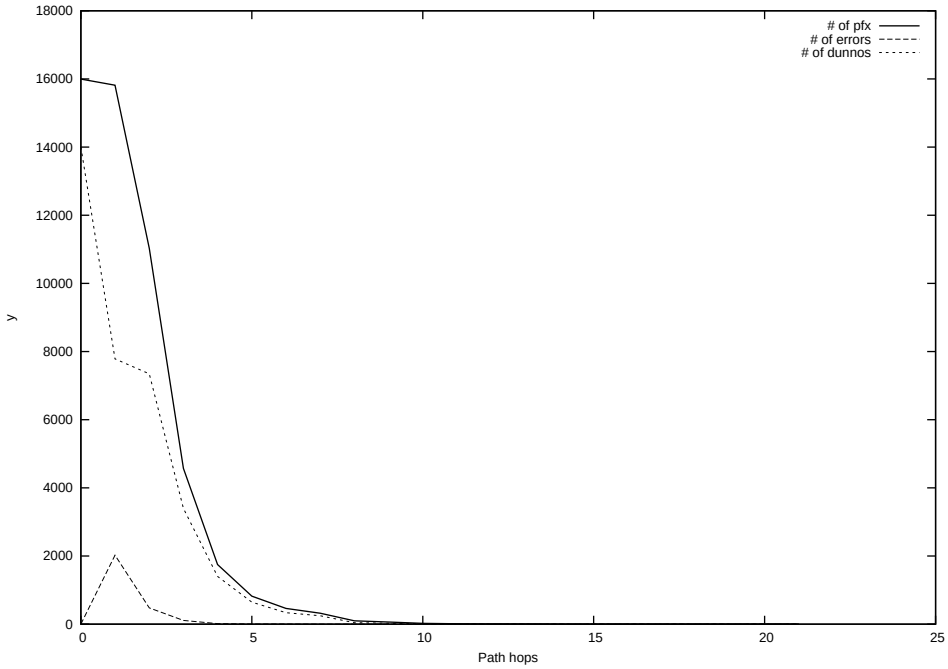
2013-12-16



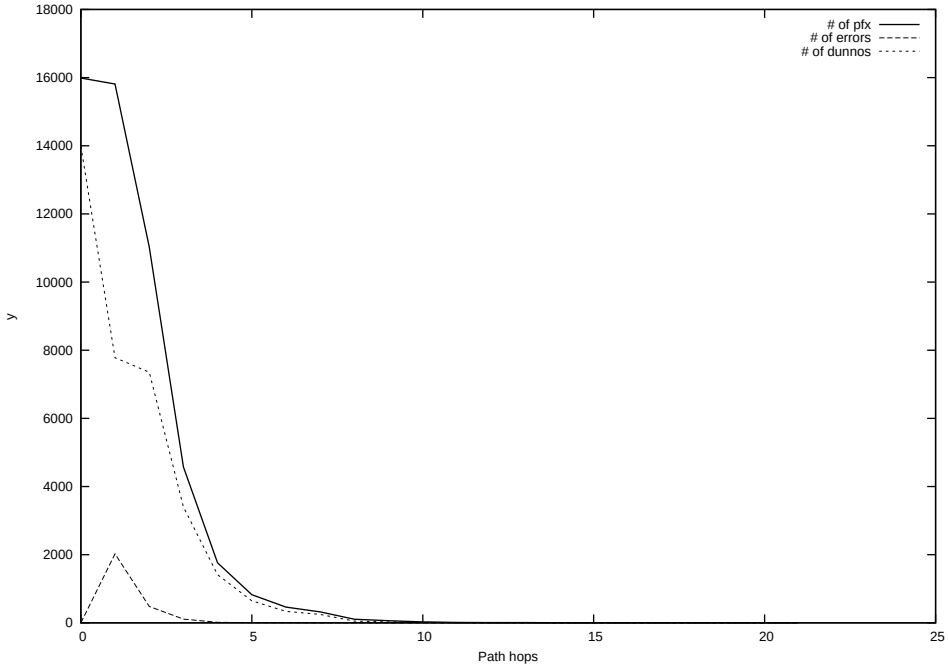
2013-12-17



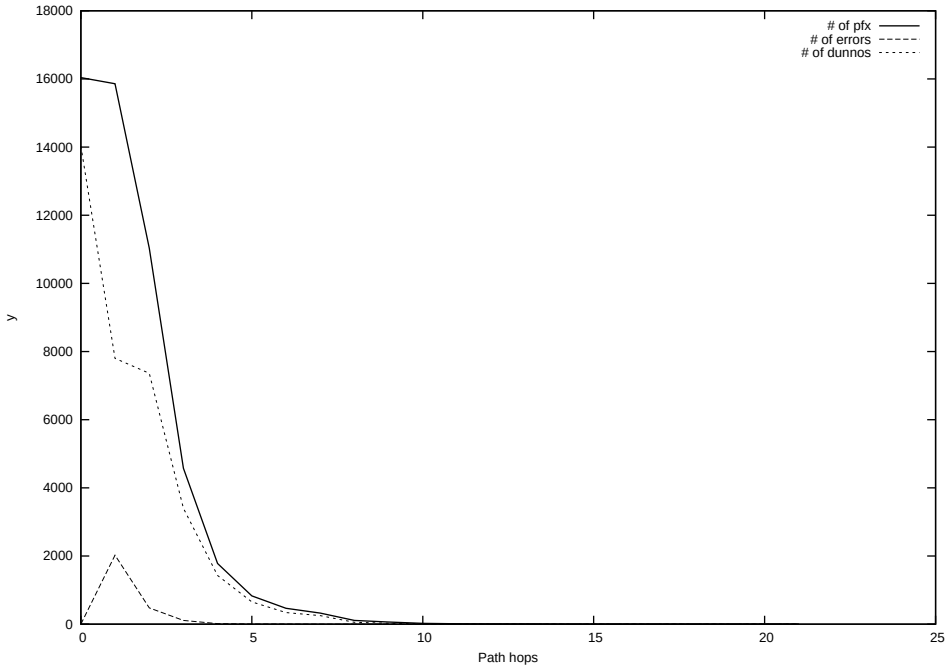
2013-12-18



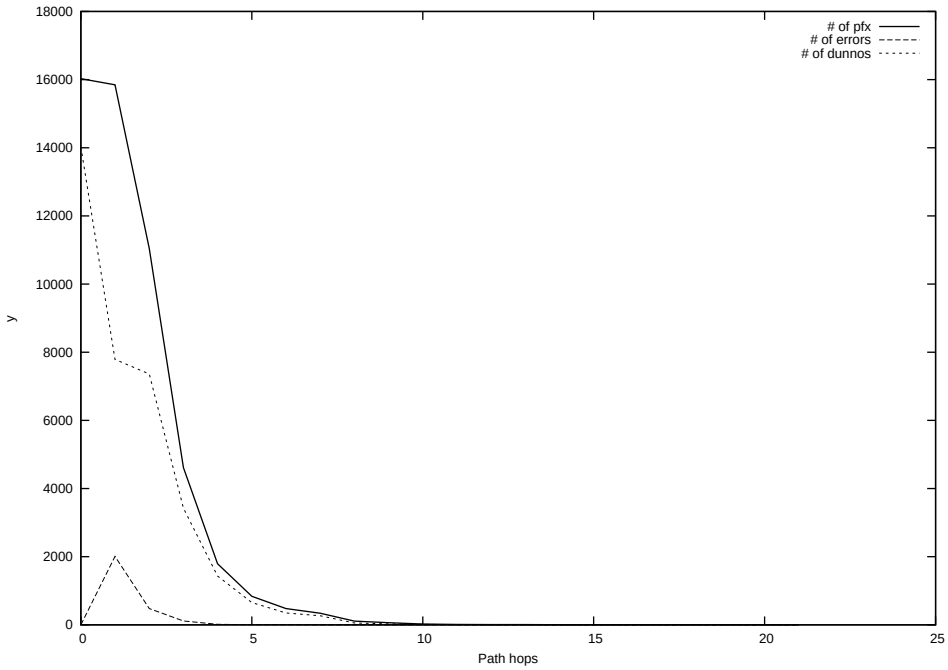
2013-12-19



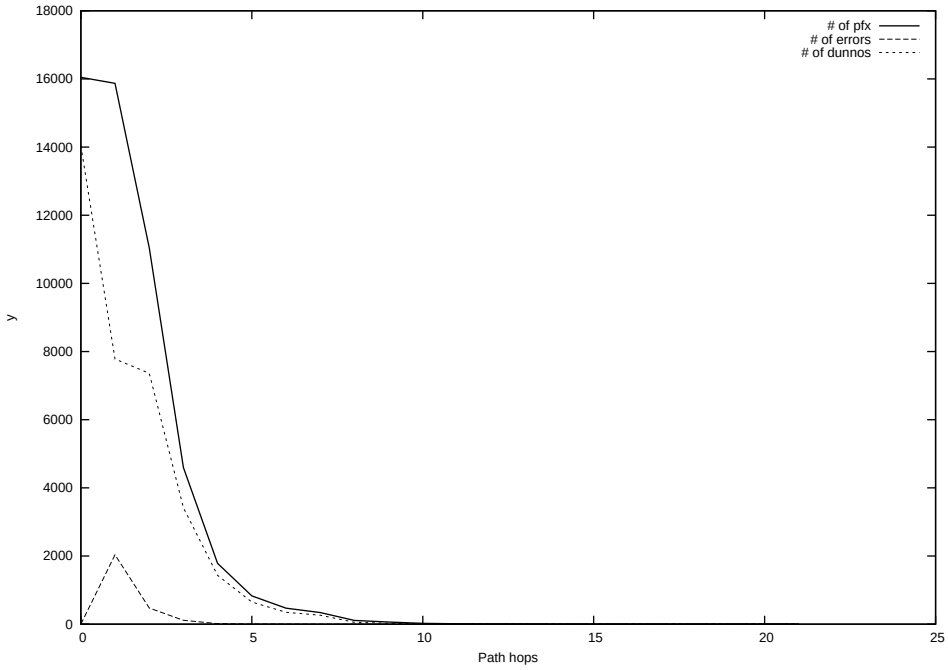
2013-12-20



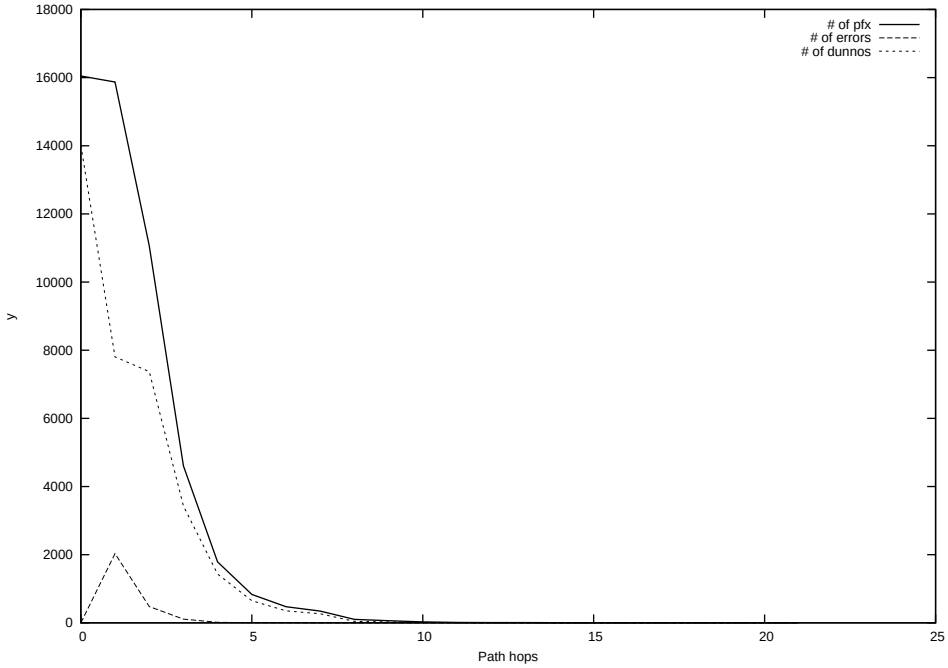
2013-12-21



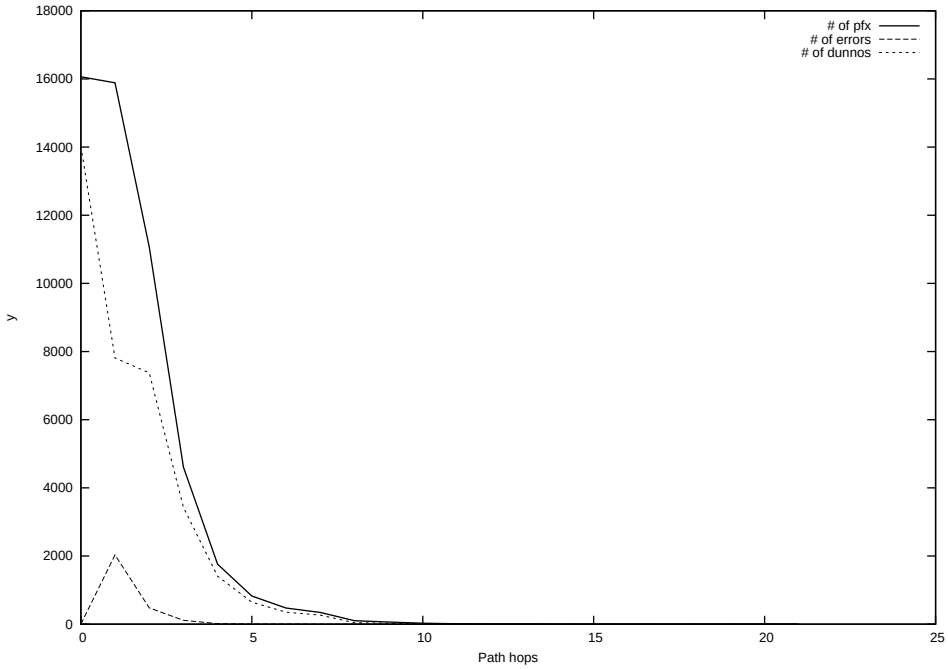
2013-12-22



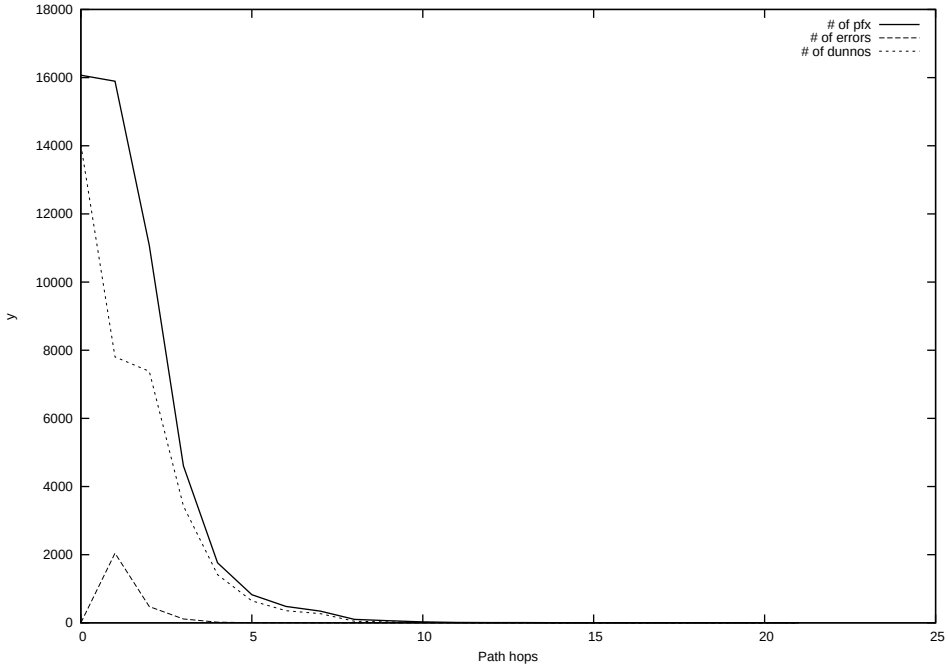
2013-12-23



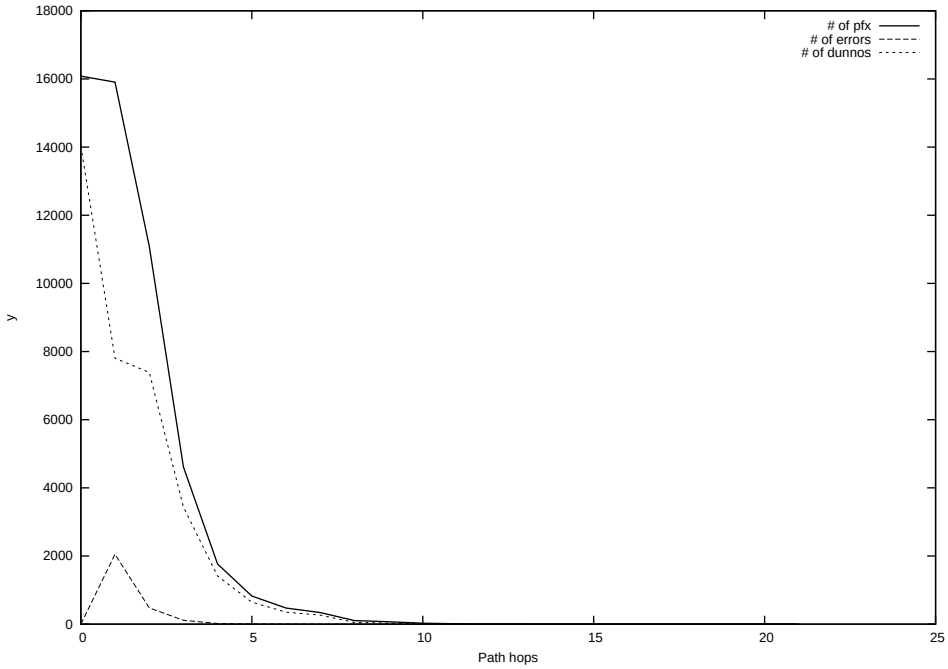
2013-12-24



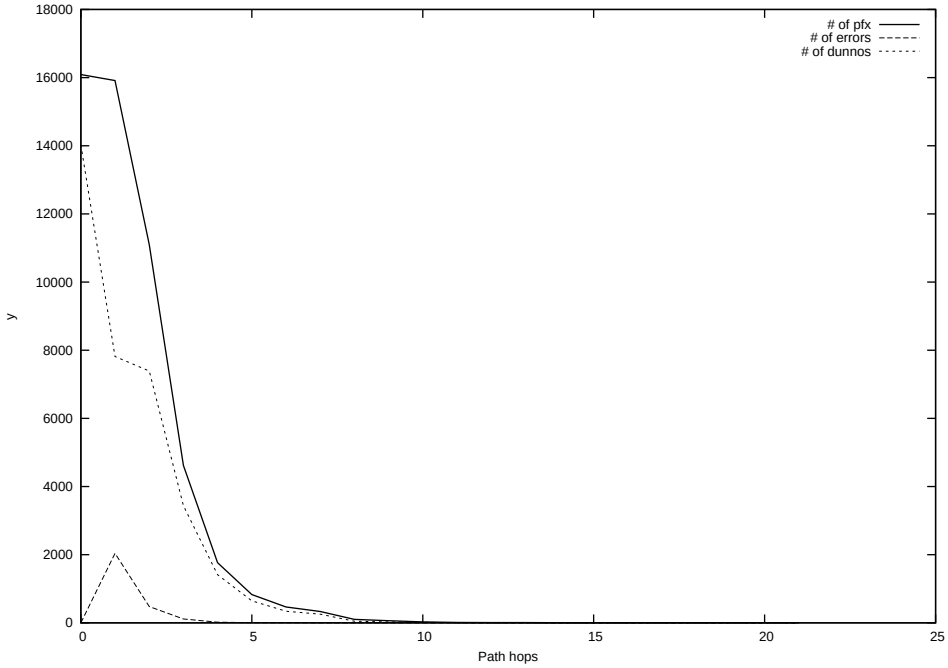
2013-12-25



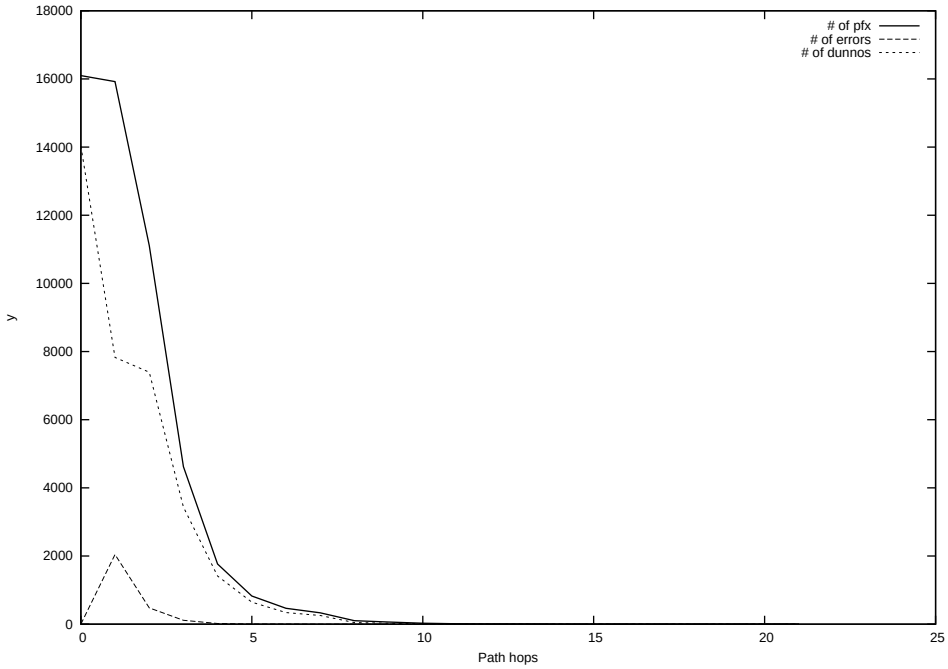
2013-12-26



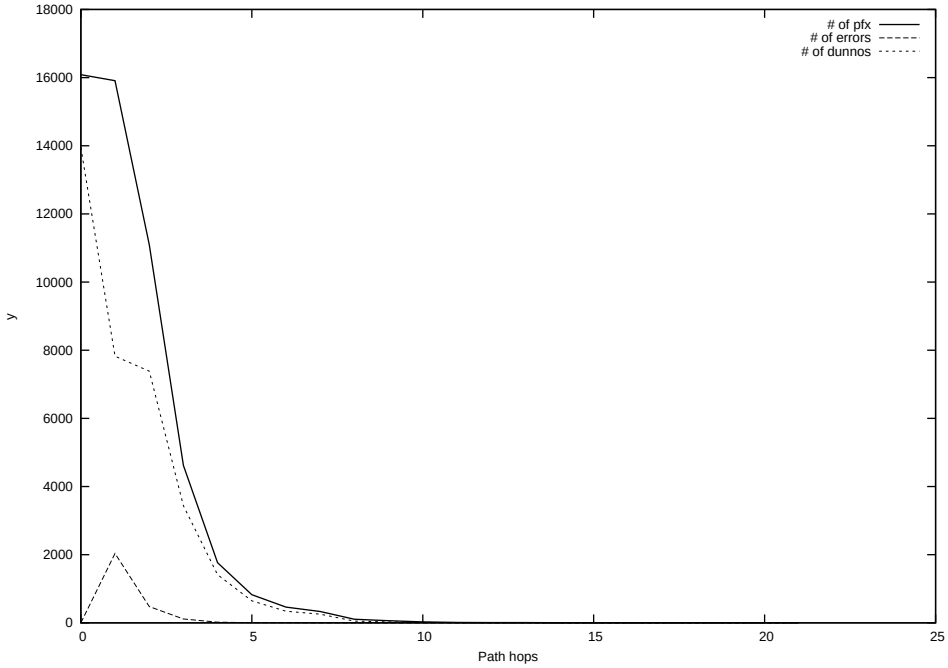
2013-12-27



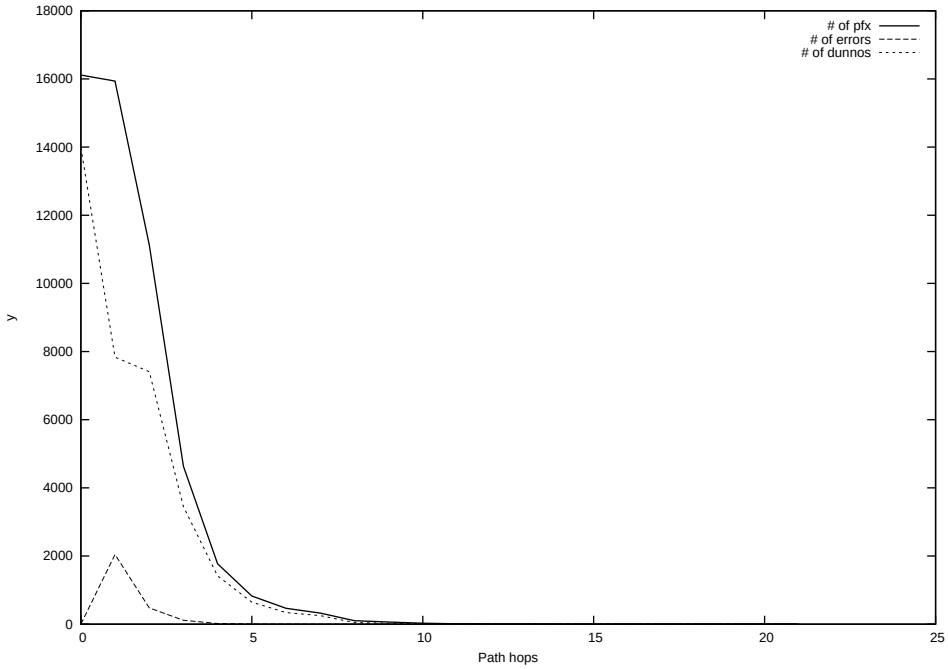
2013-12-28



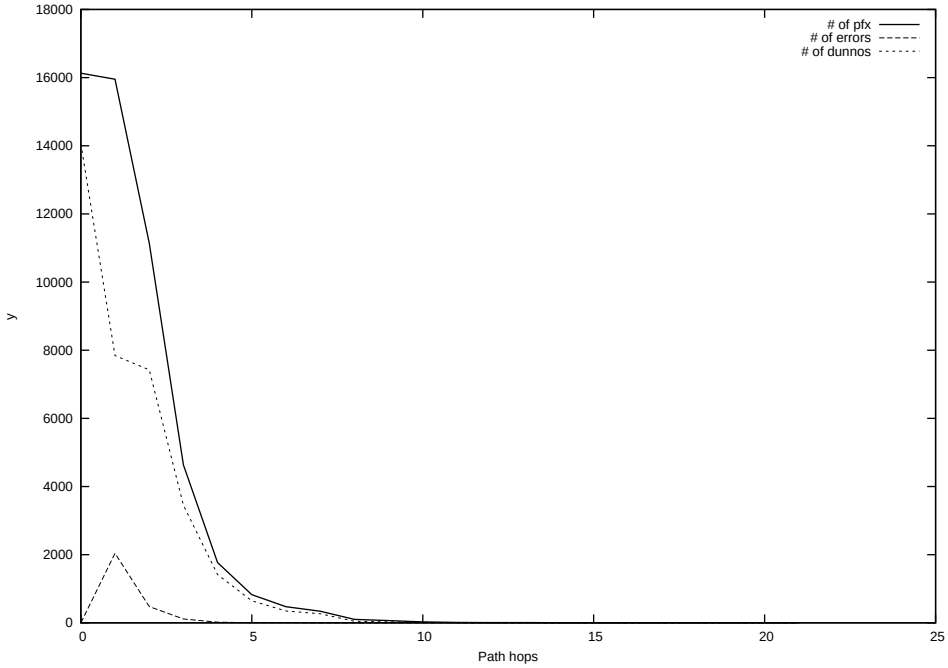
2013-12-29



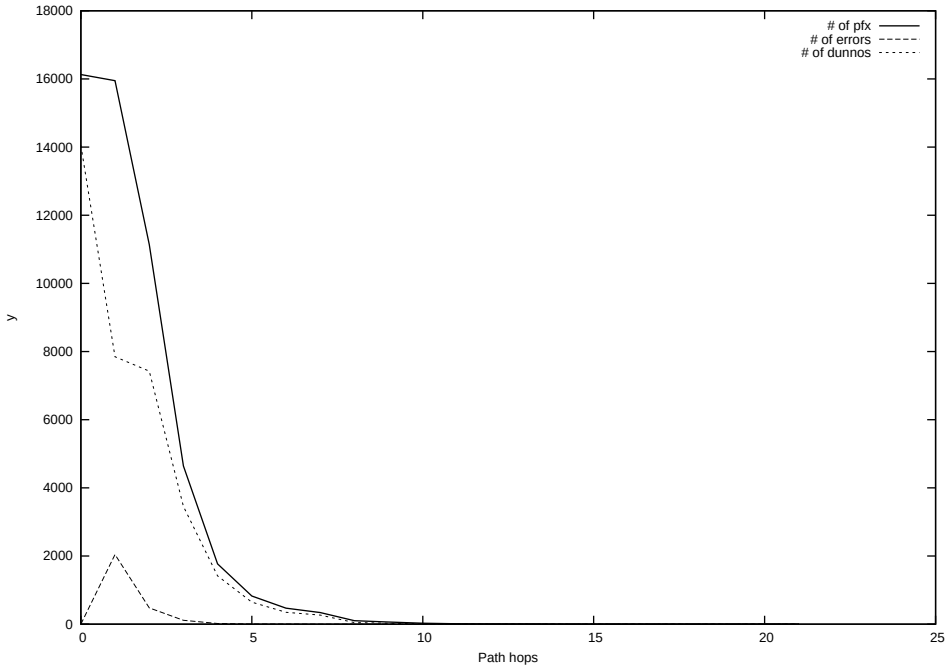
2013-12-30



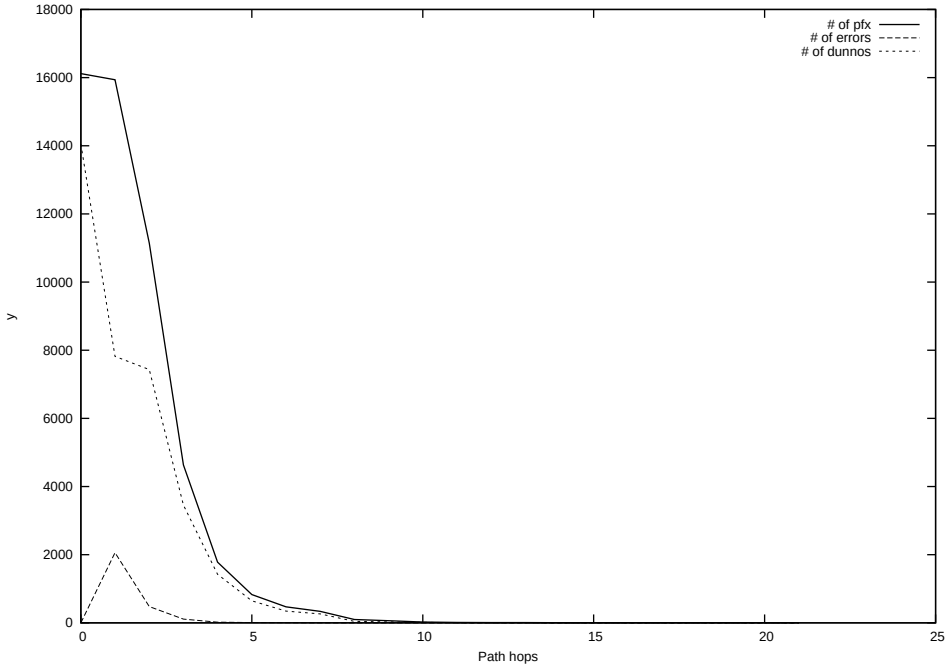
2013-12-31



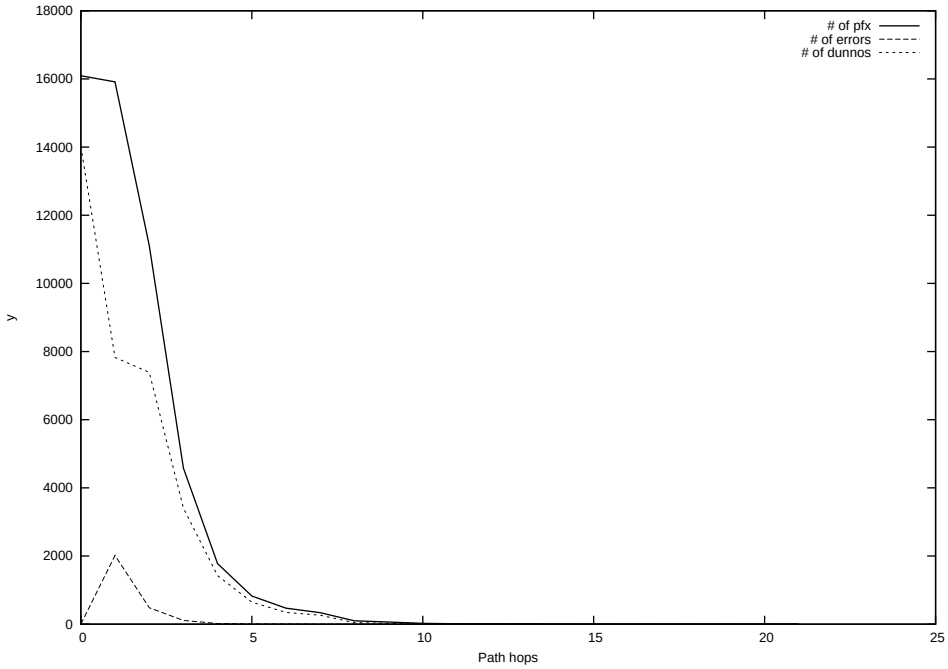
2014-01-01



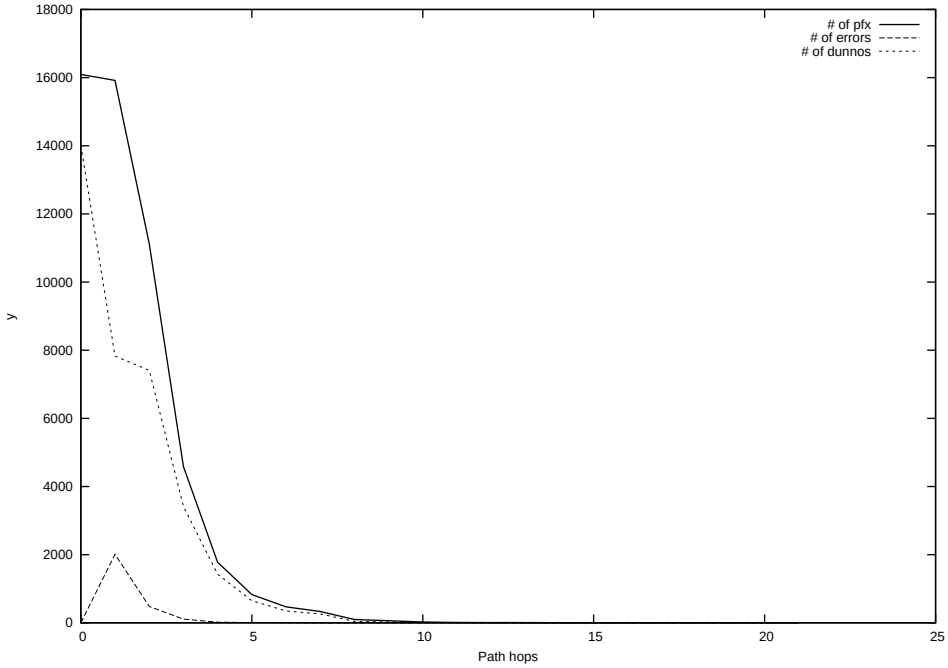
2014-01-02



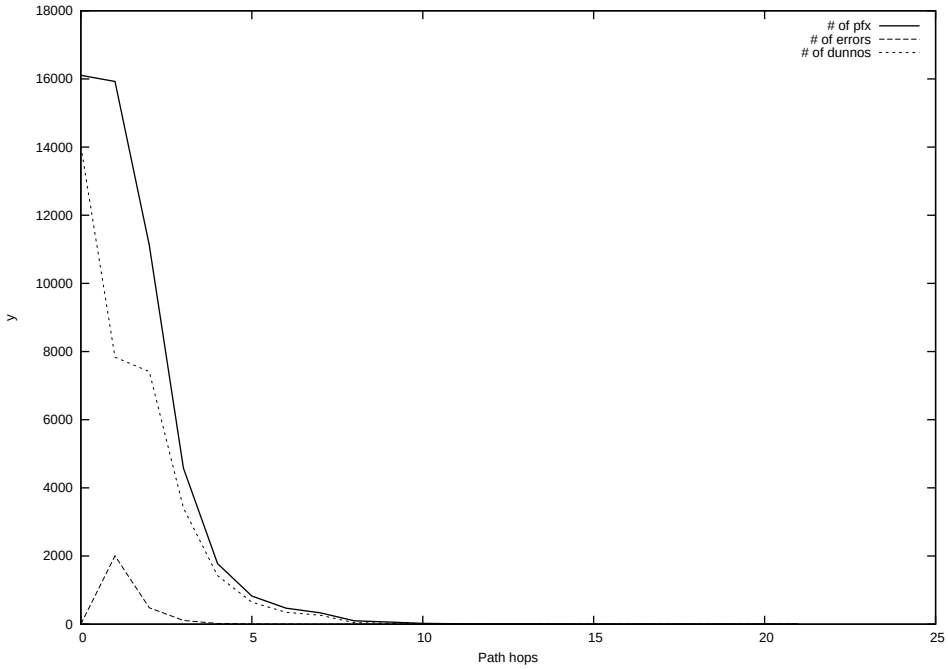
2014-01-03



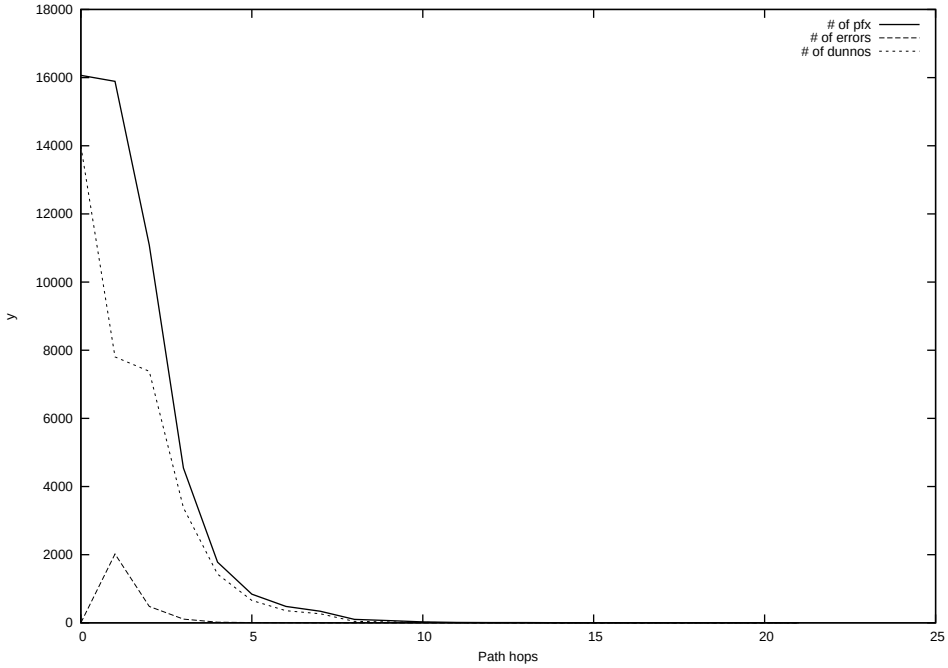
2014-01-04



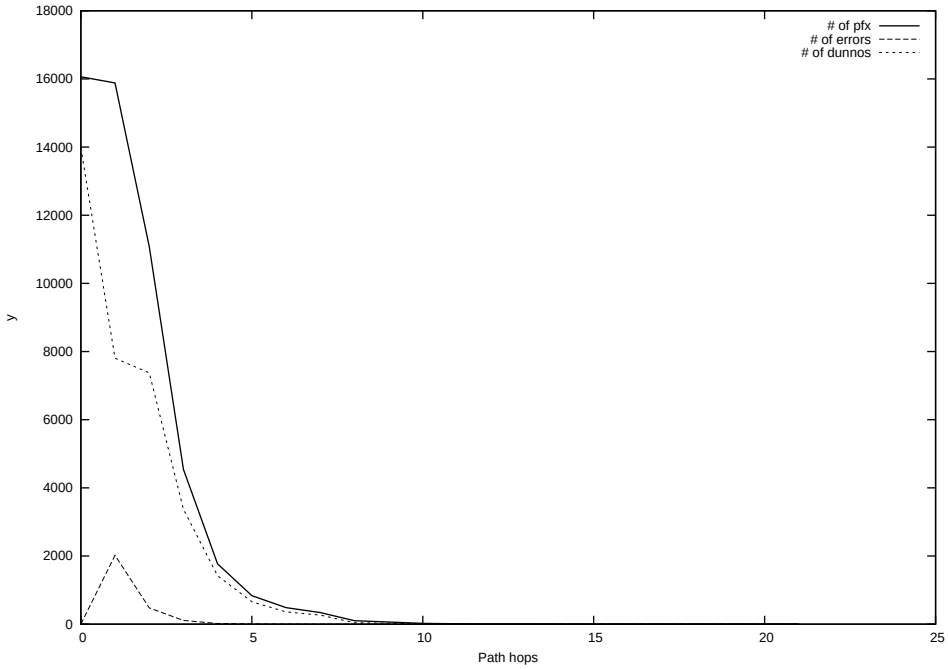
2014-01-05



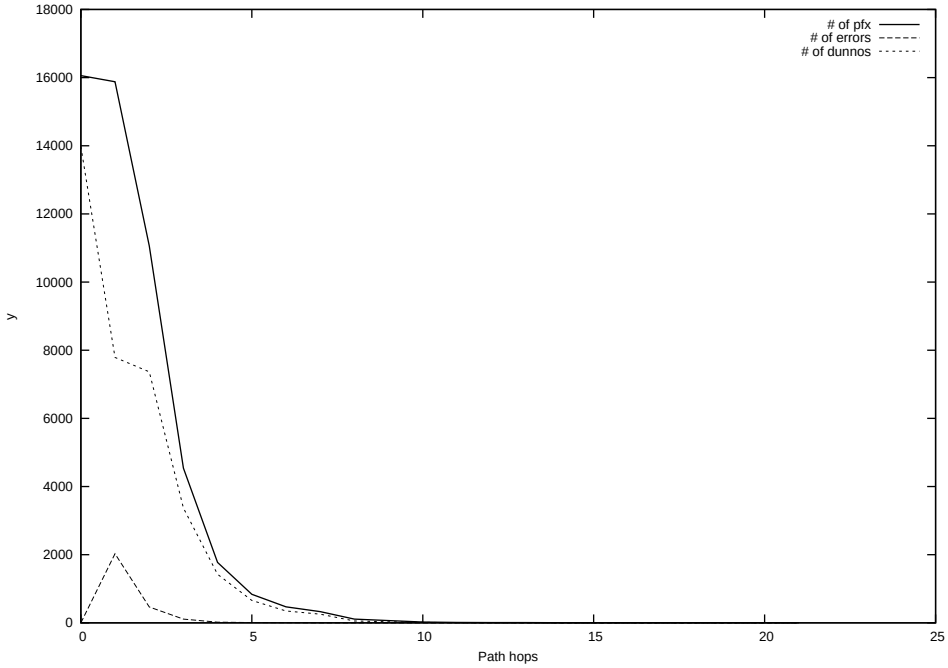
2014-01-06



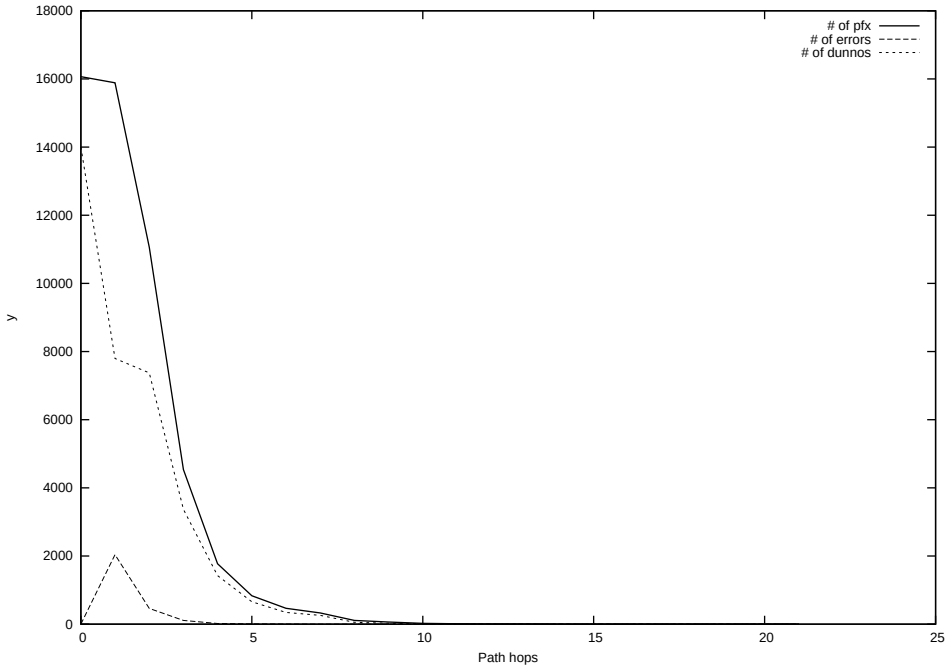
2014-01-07



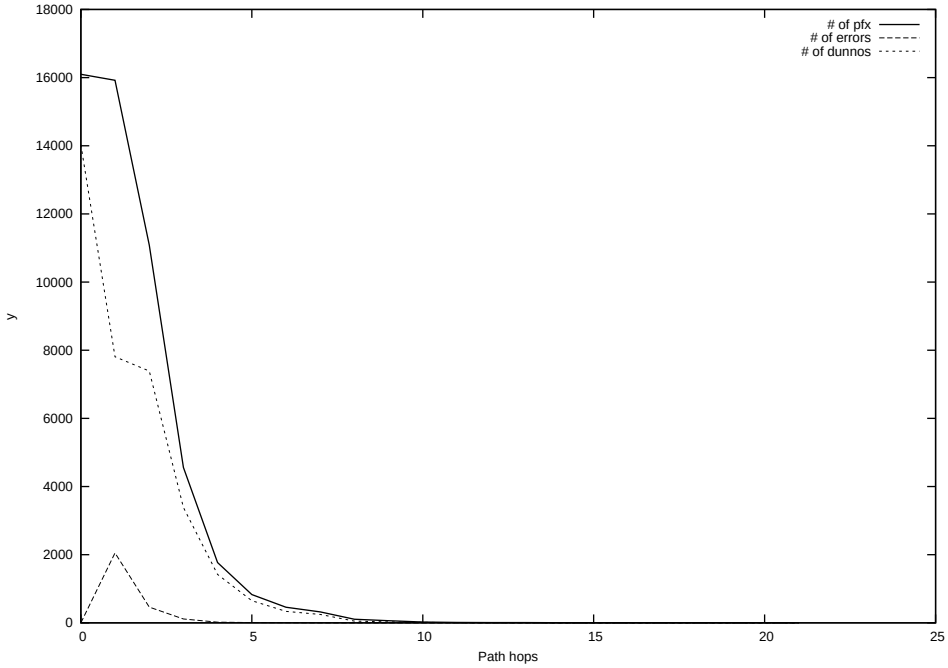
2014-01-08



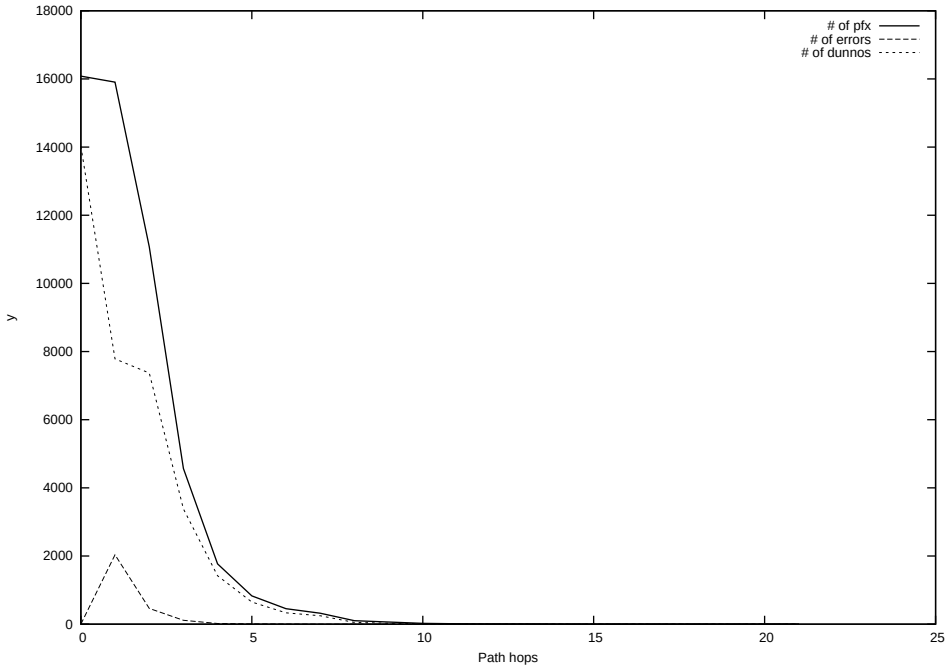
2014-01-09



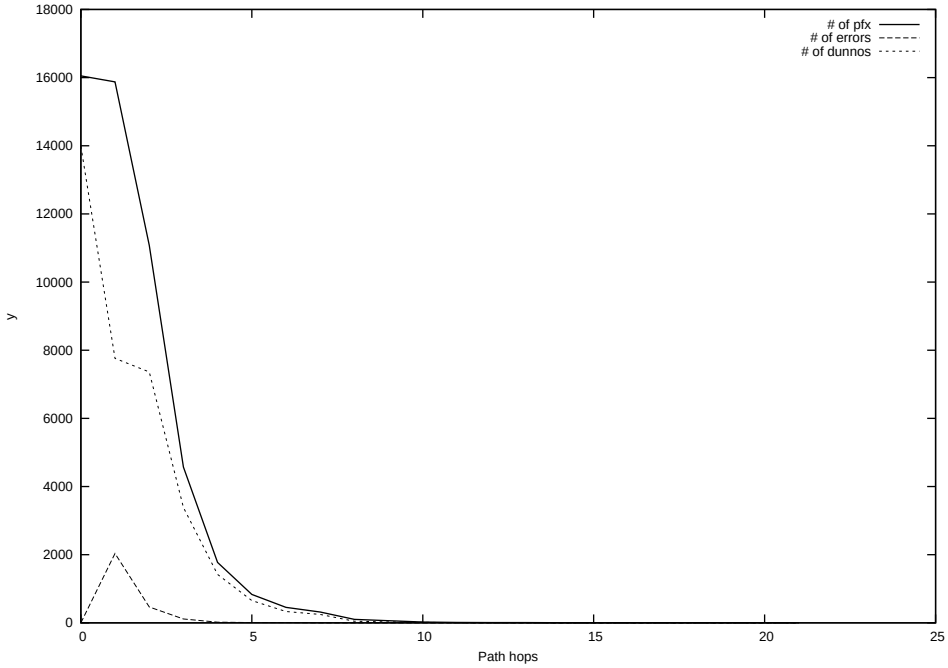
2014-01-10



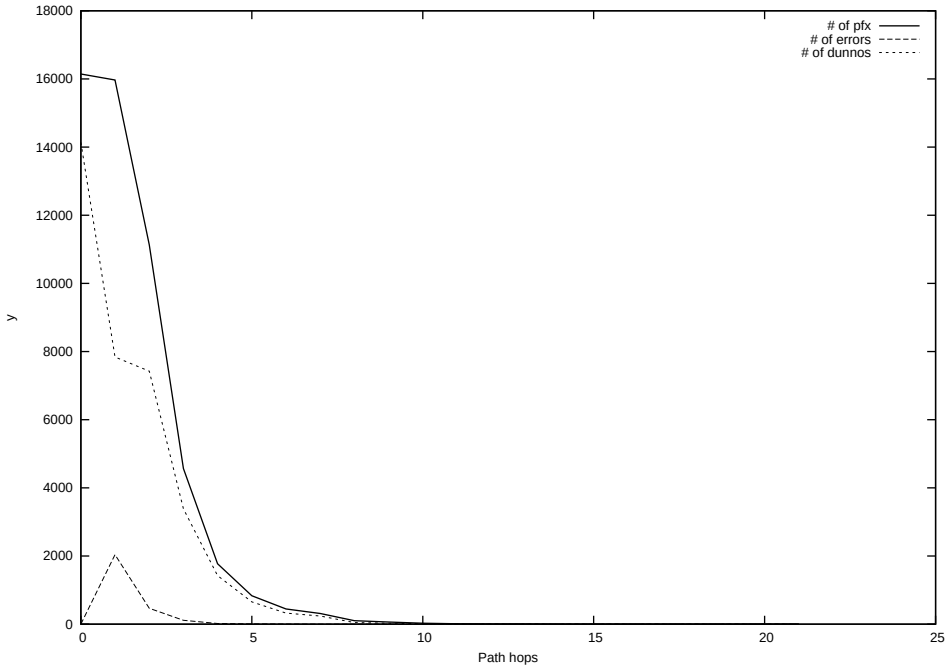
2014-01-11



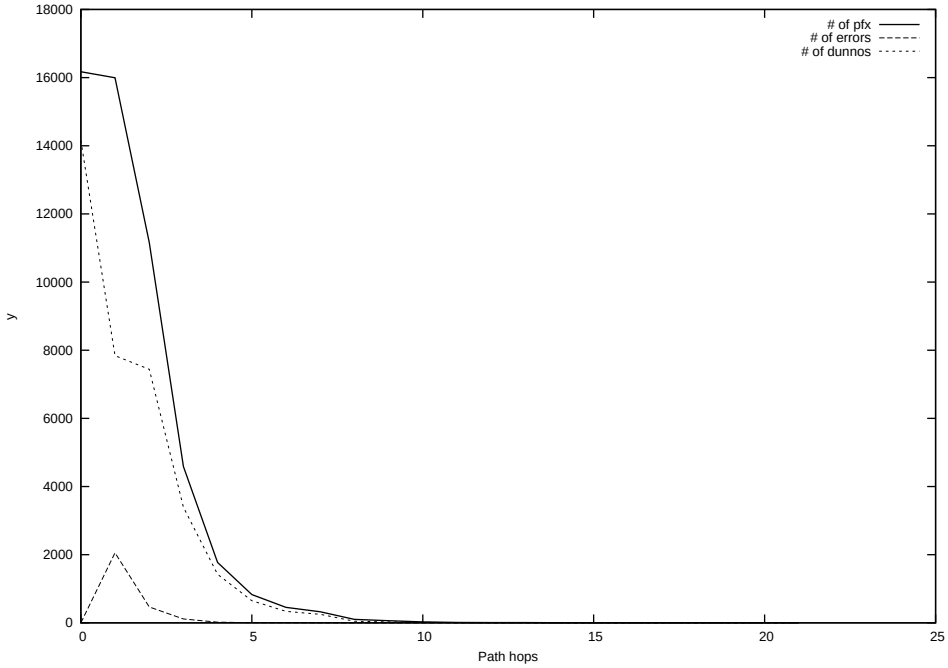
2014-01-12



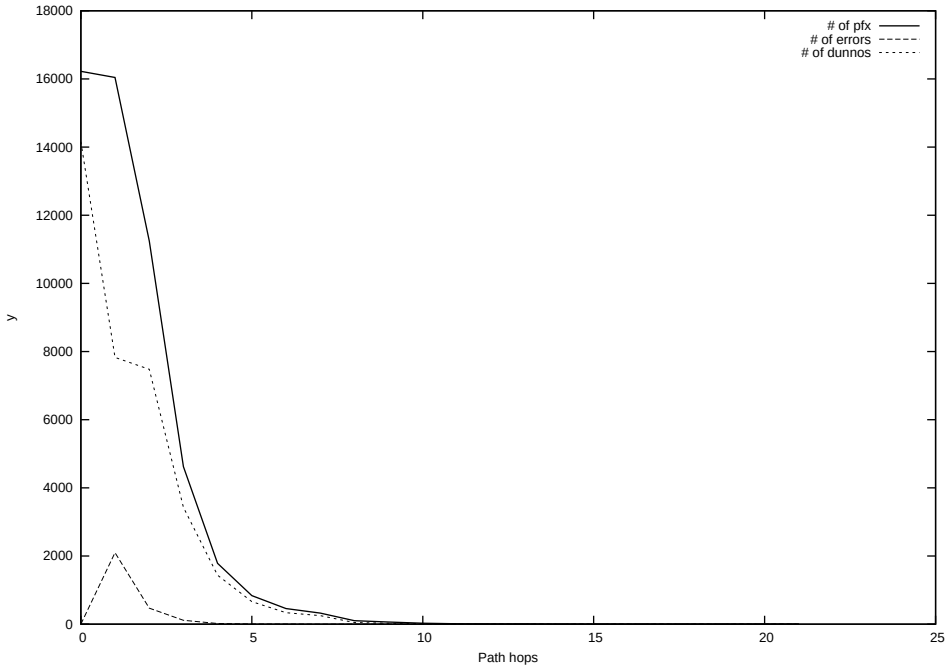
2014-01-13



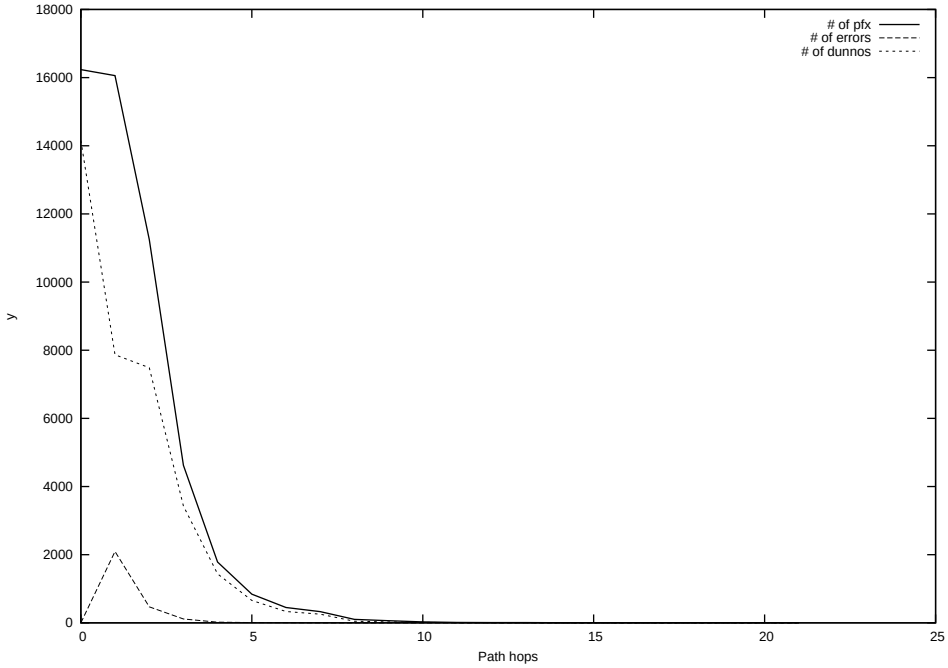
2014-01-14



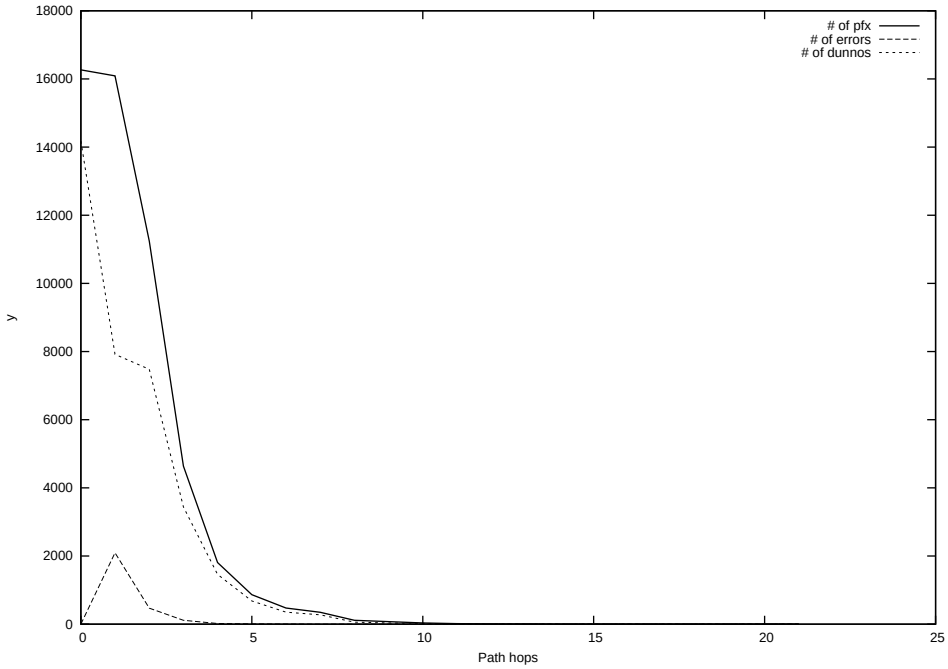
2014-01-15



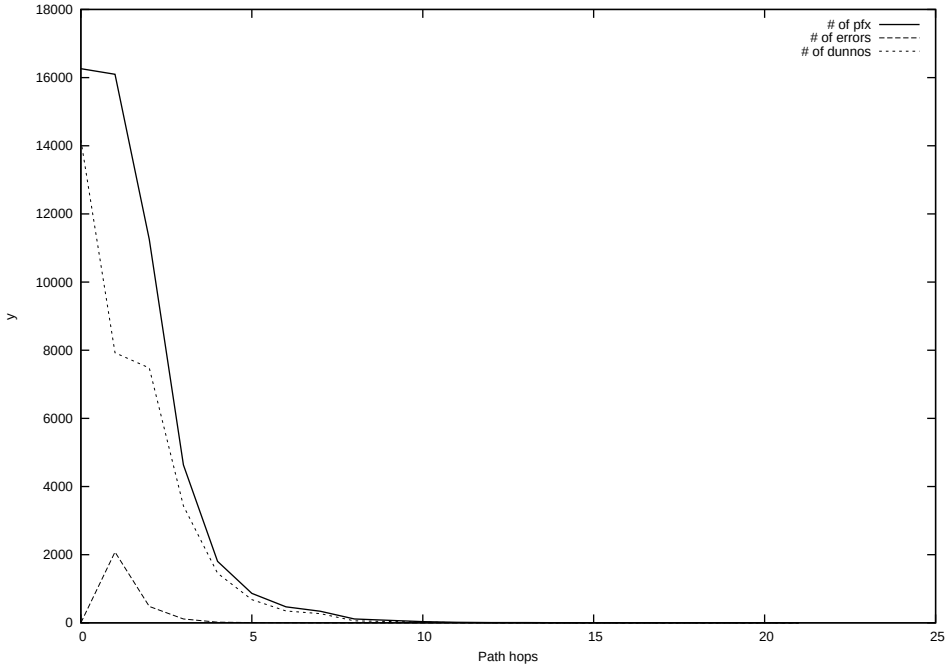
2014-01-16



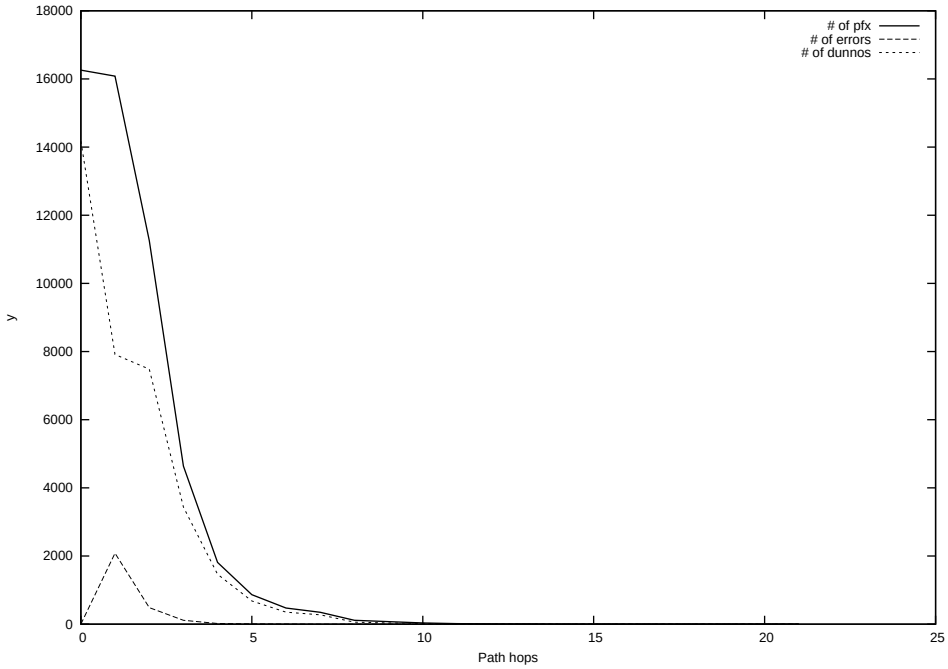
2014-01-17



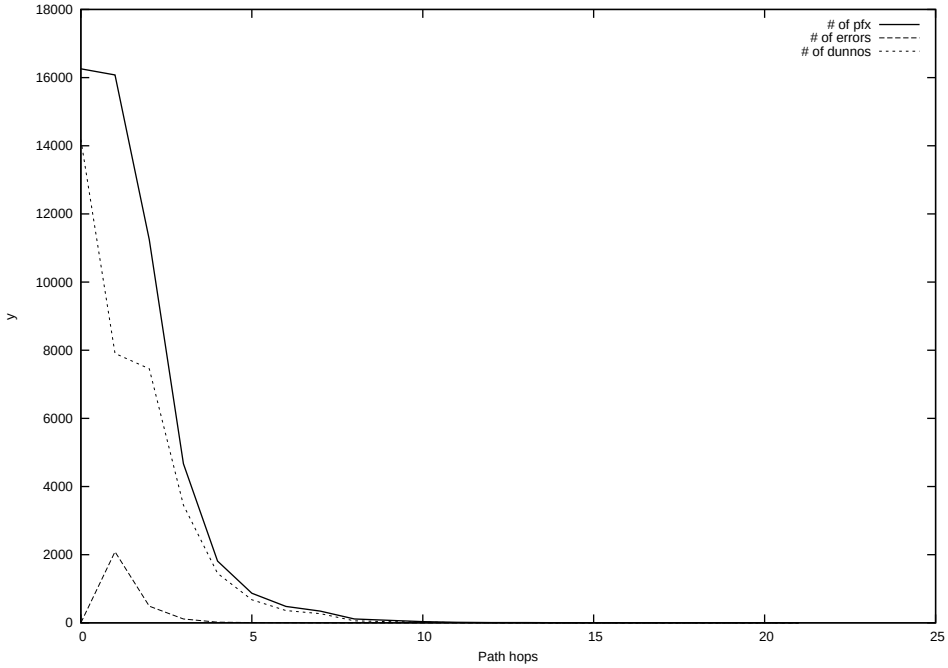
2014-01-18



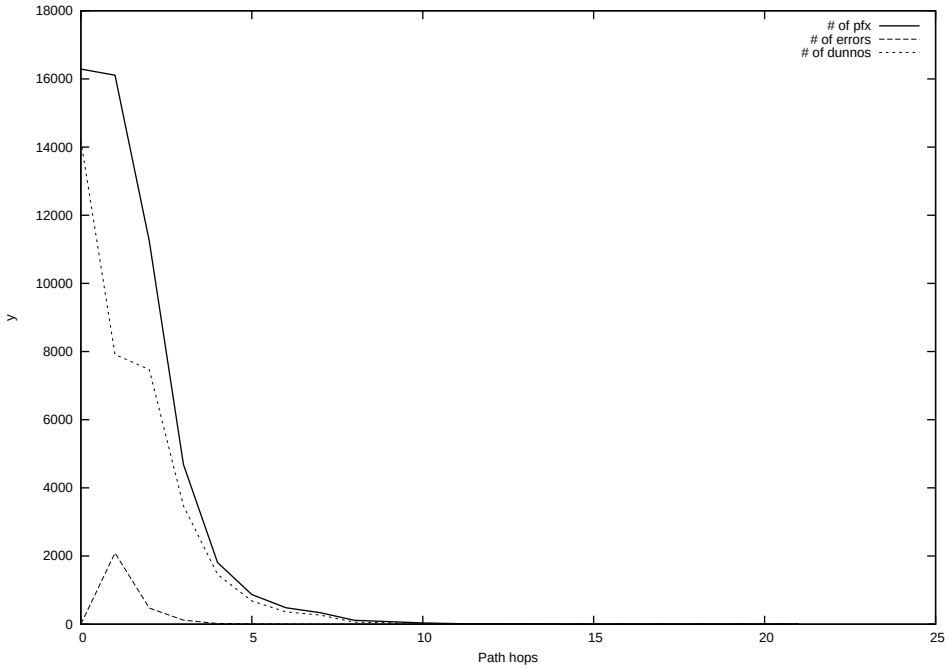
2014-01-19



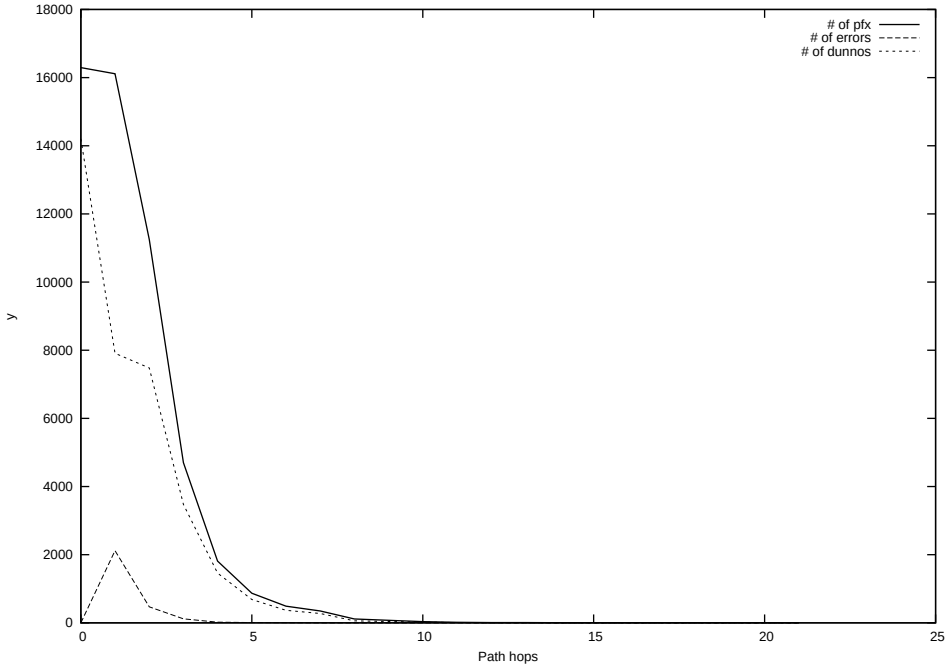
2014-01-20



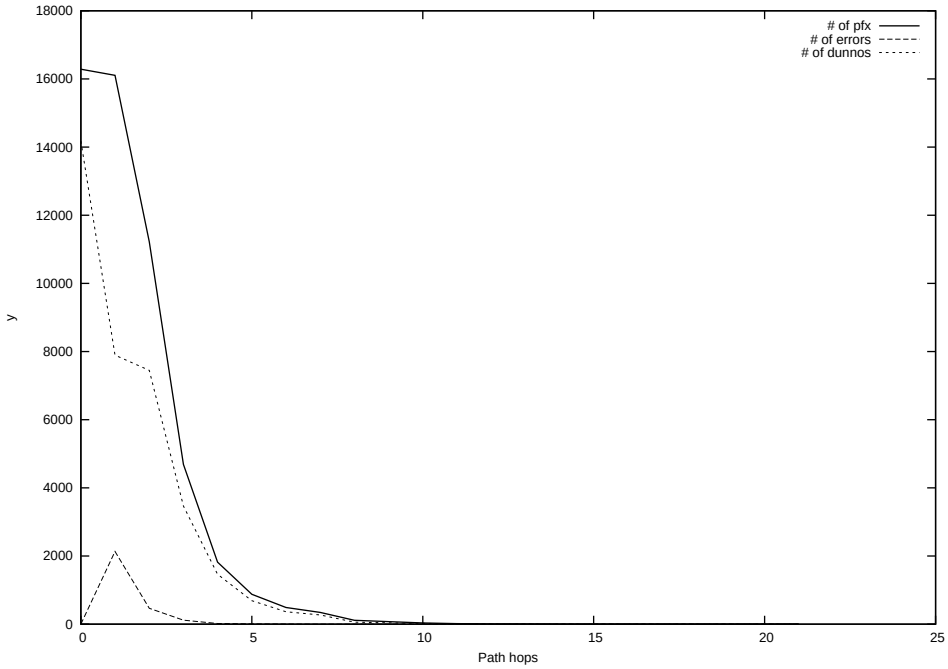
2014-01-21



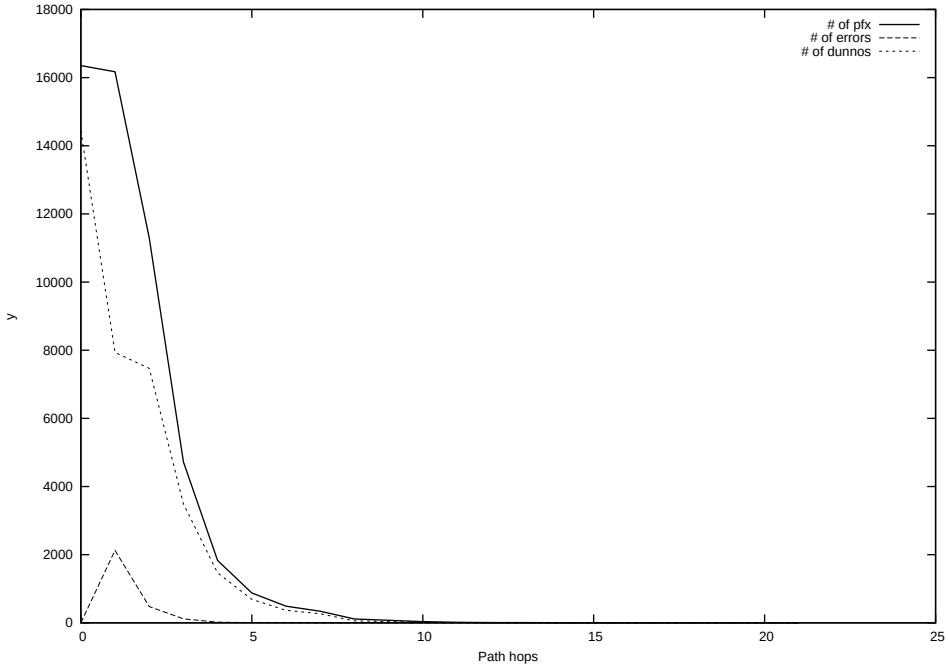
2014-01-22



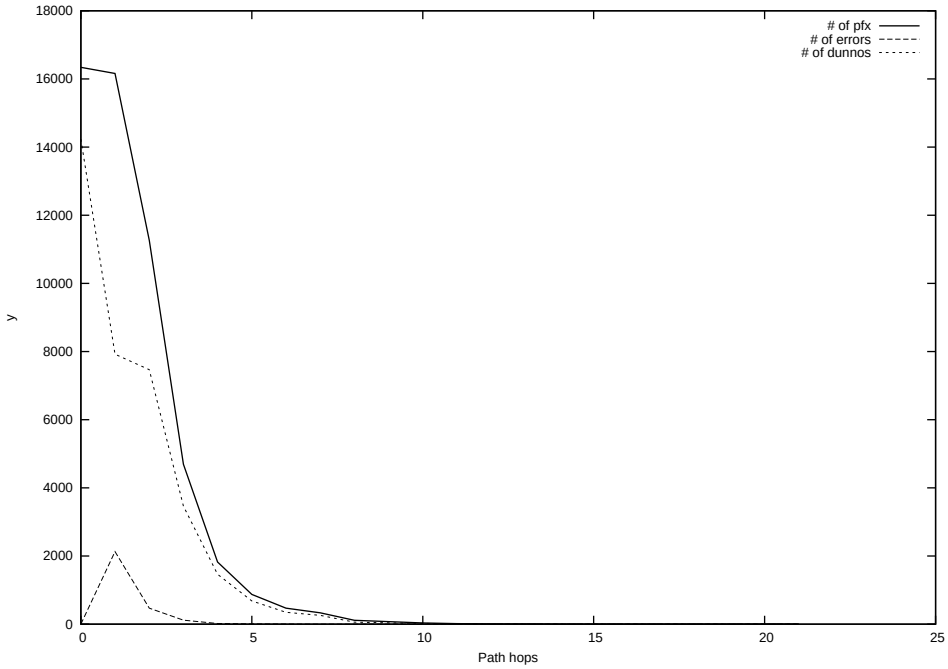
2014-01-23



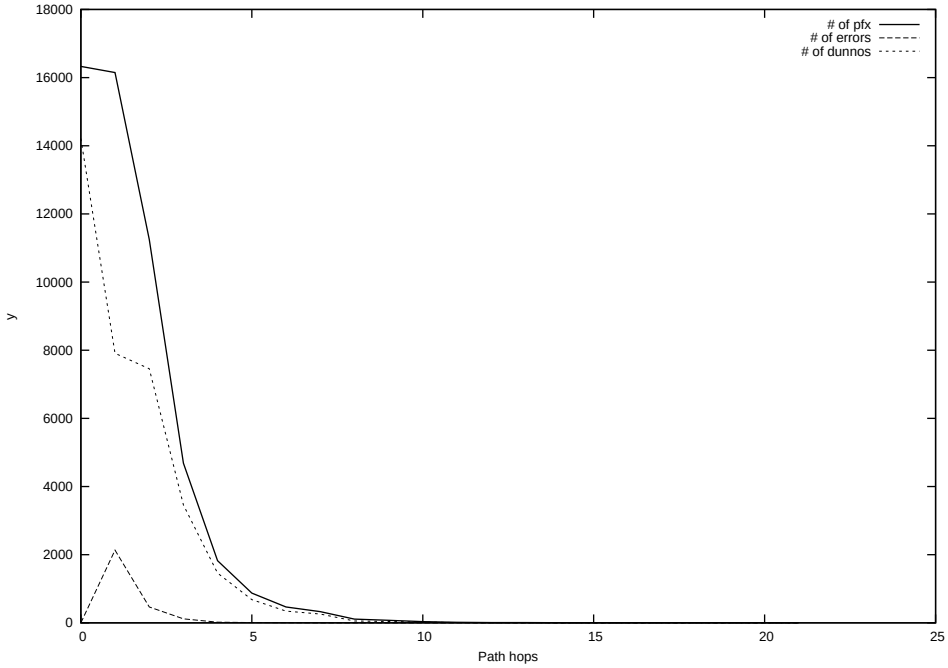
2014-01-24



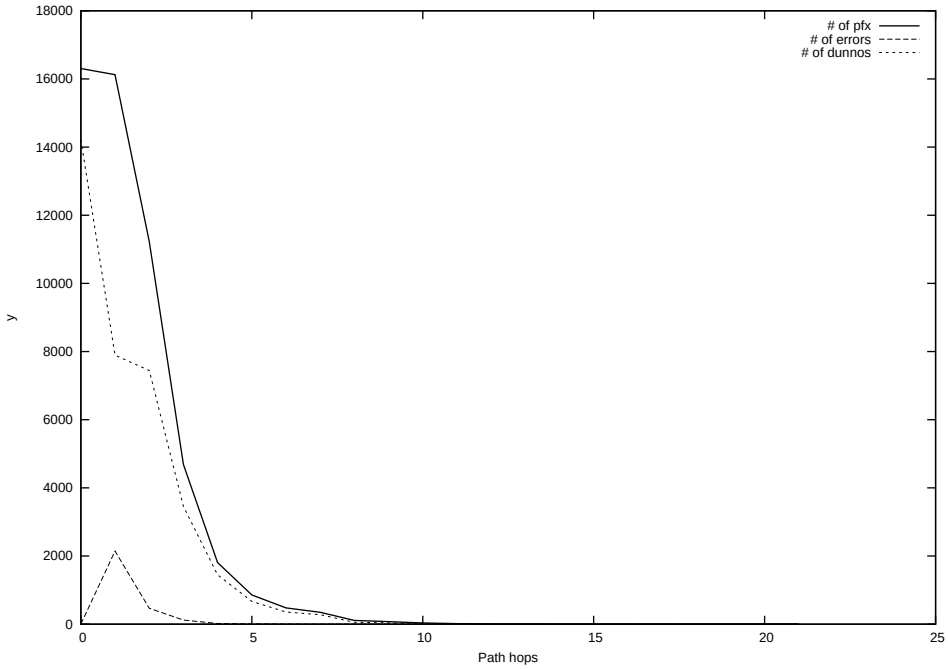
2014-01-25



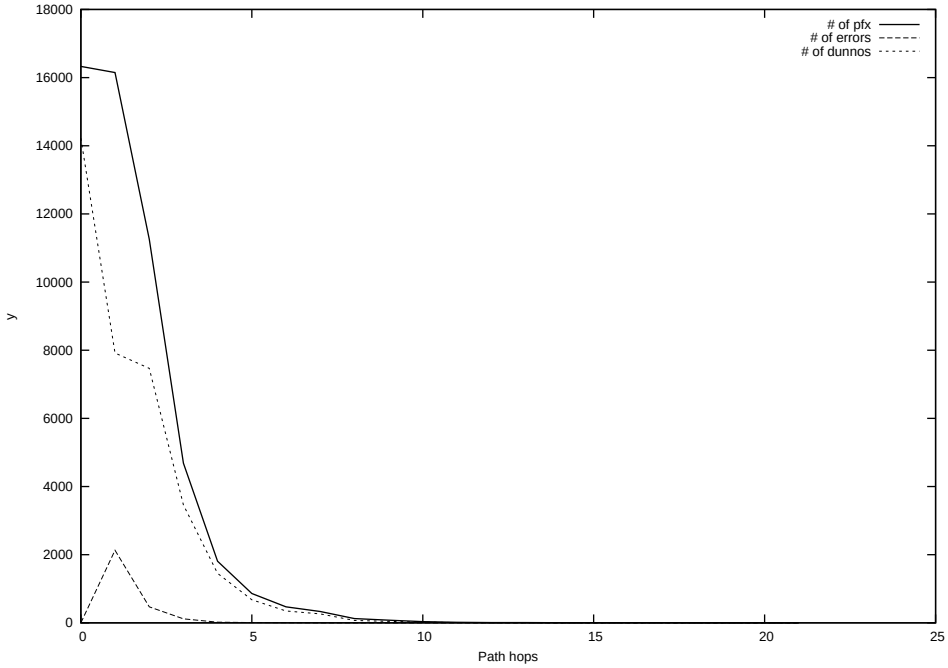
2014-01-26



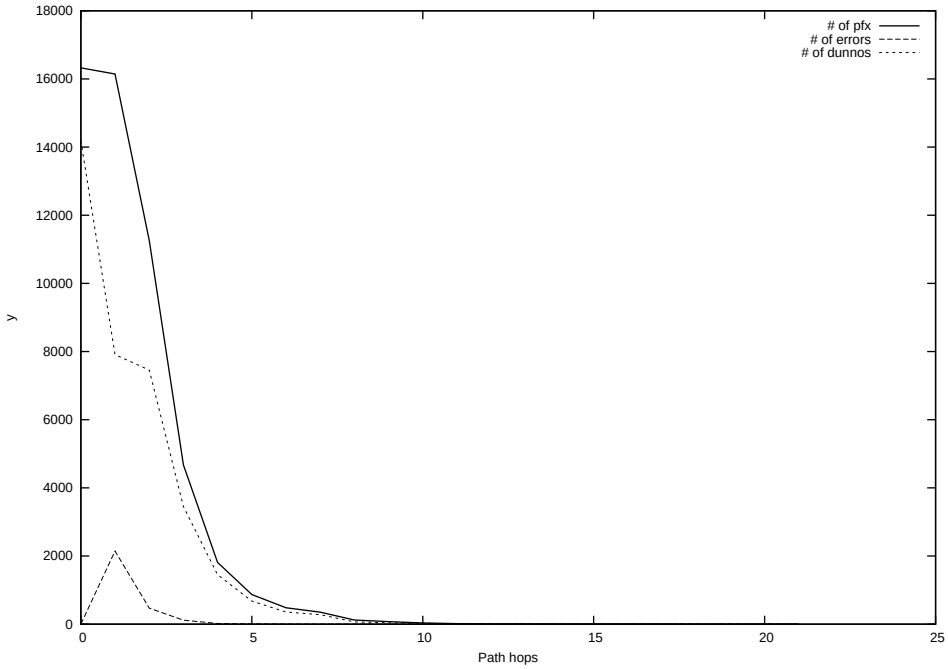
2014-01-27



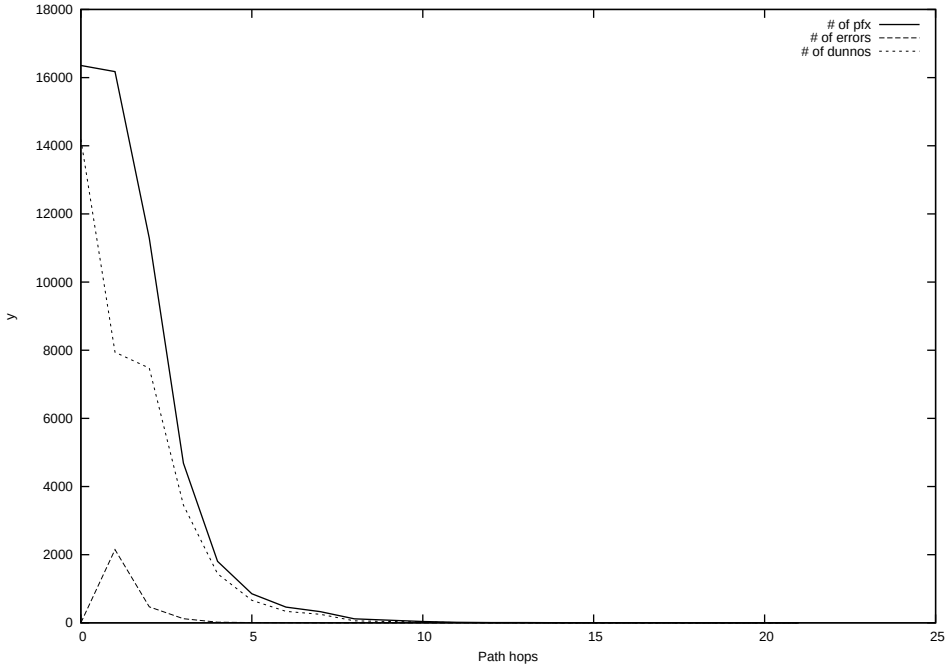
2014-01-28



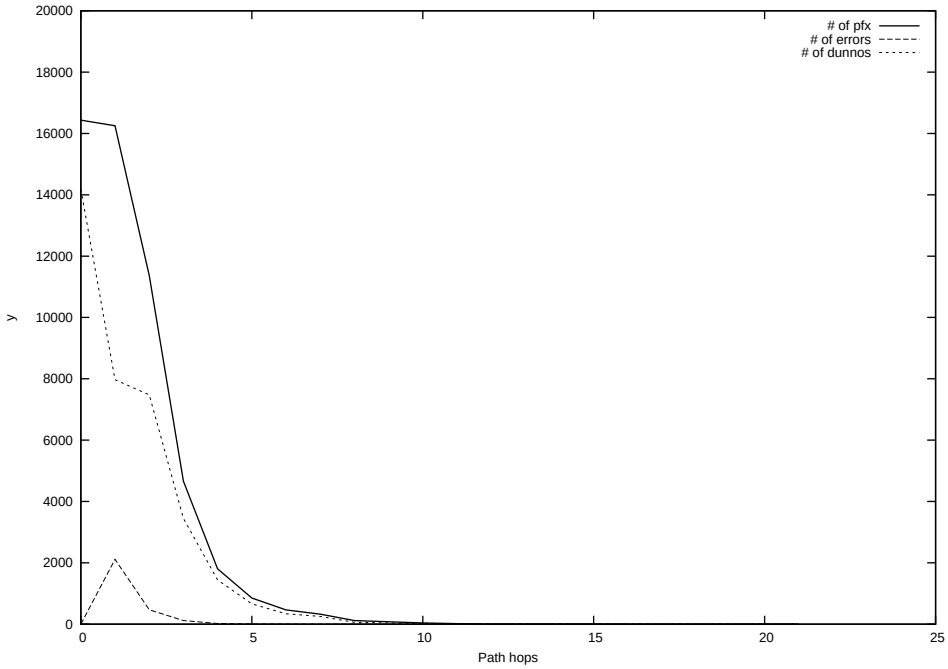
2014-01-29



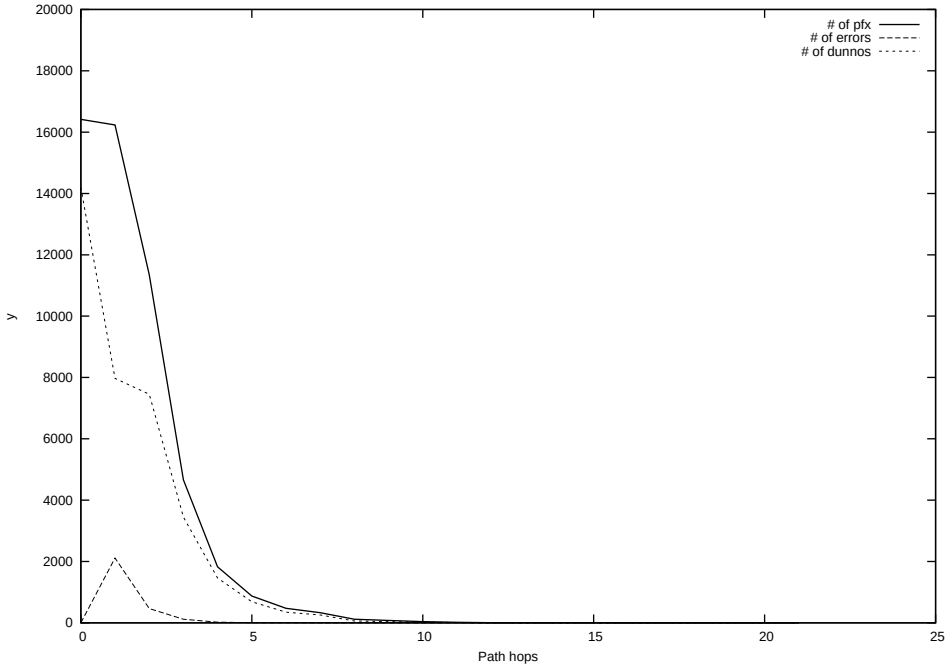
2014-01-30



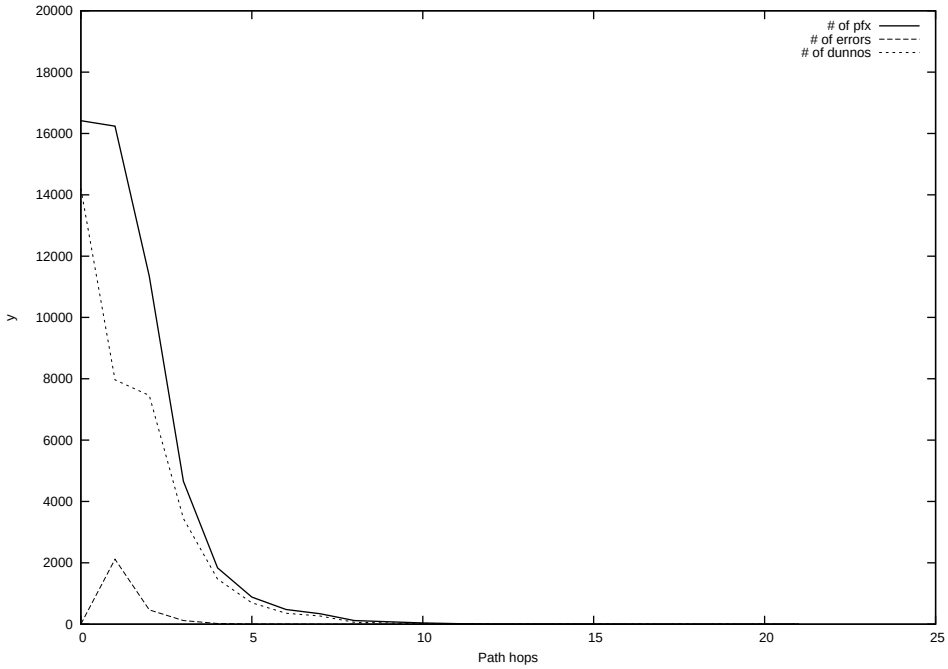
2014-01-31



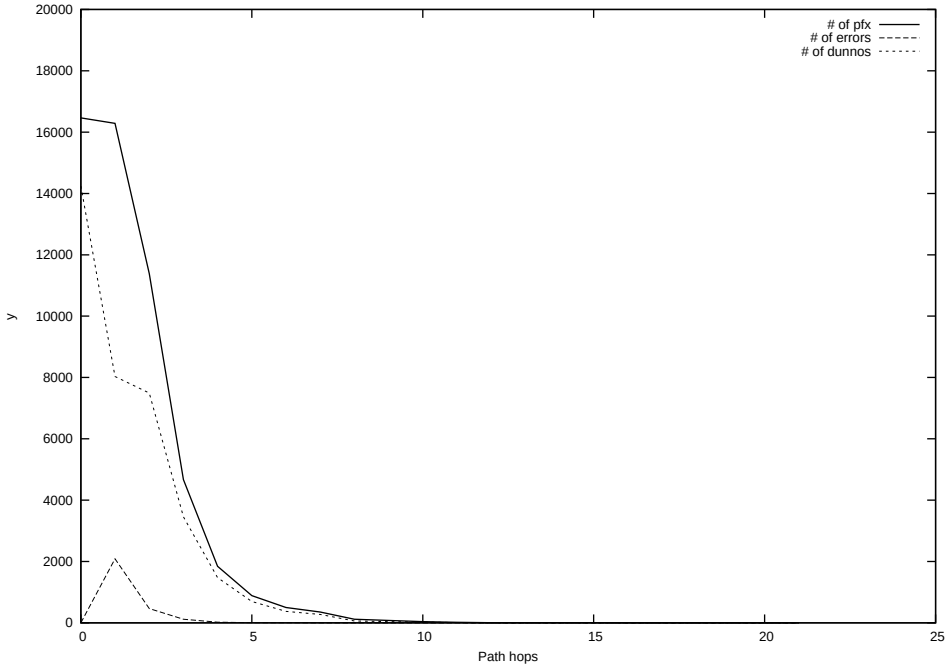
2014-02-01



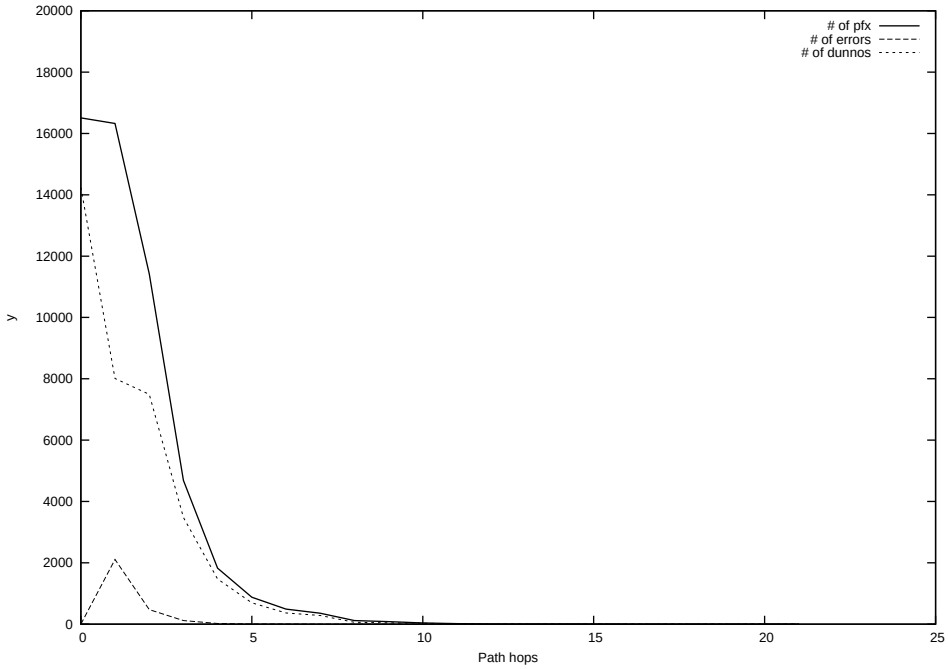
2014-02-02



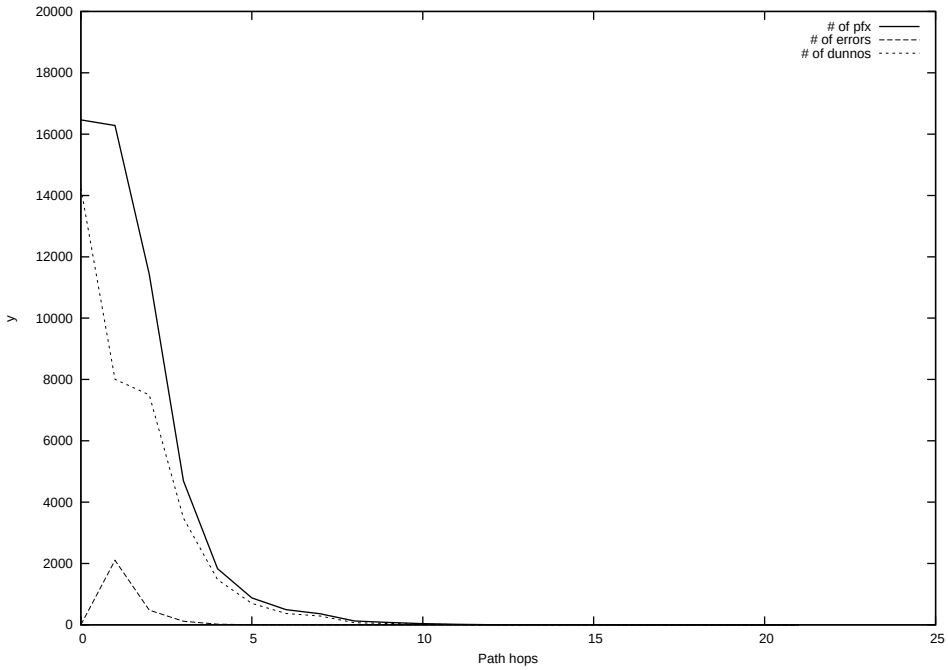
2014-02-03



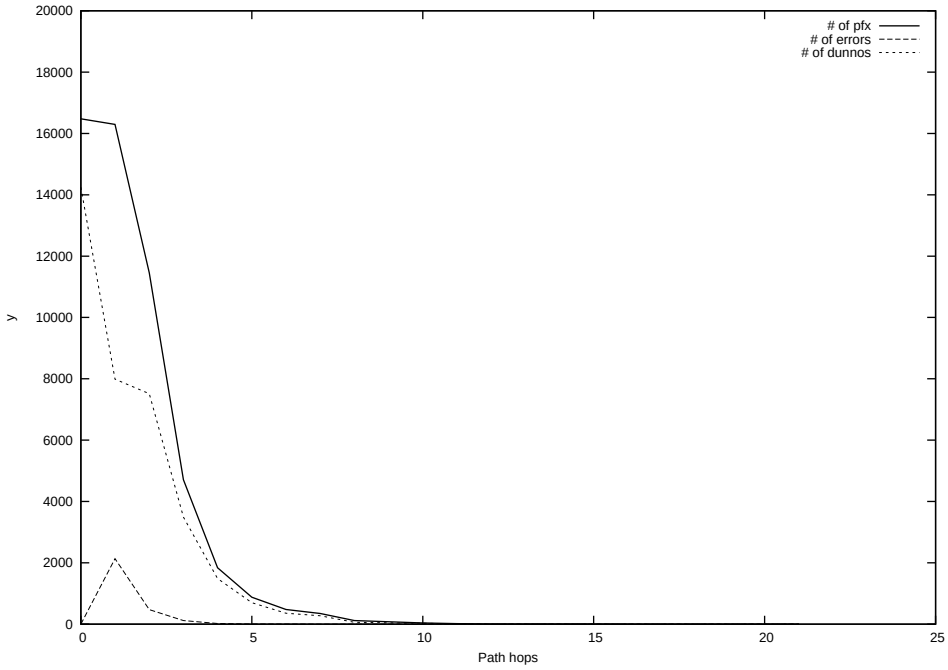
2014-02-04



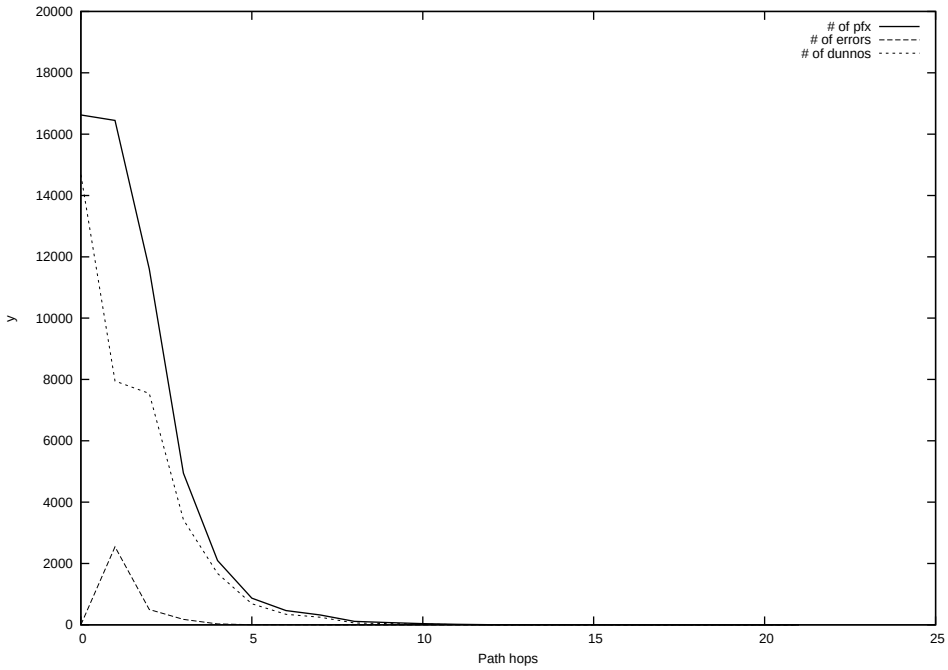
2014-02-05



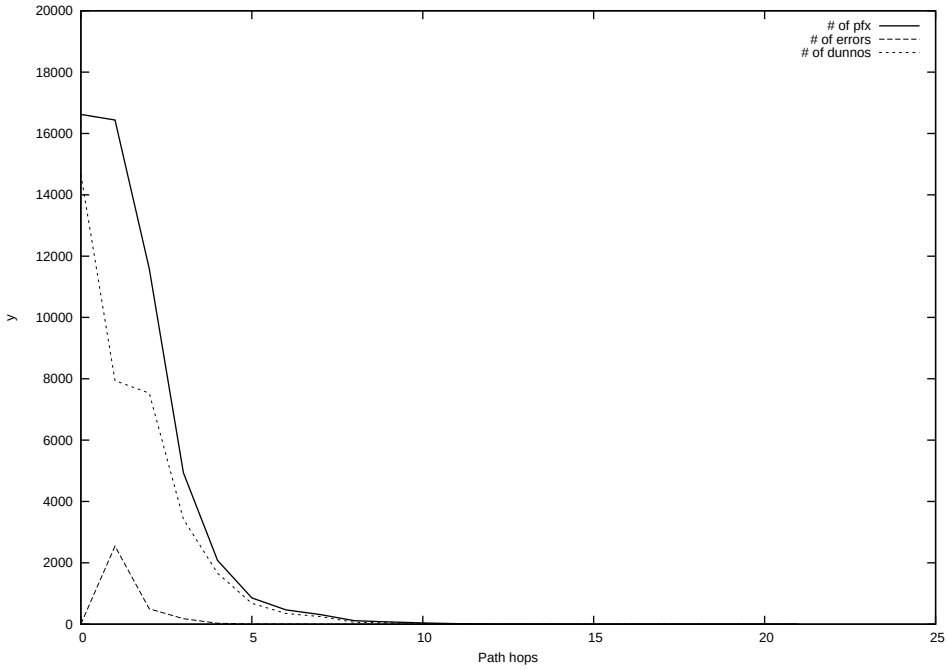
2014-02-06



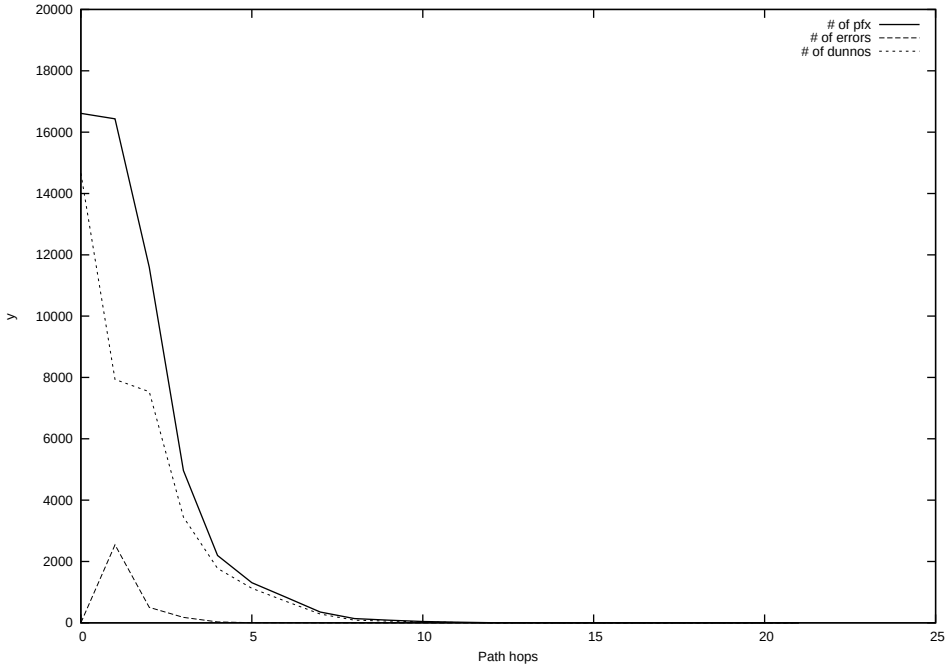
2014-02-07



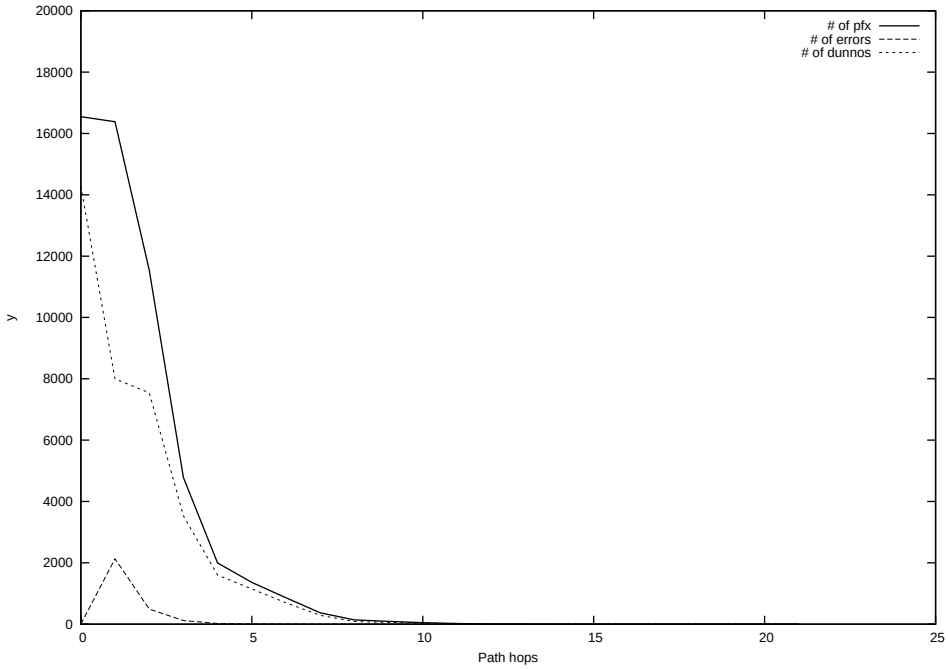
2014-02-08



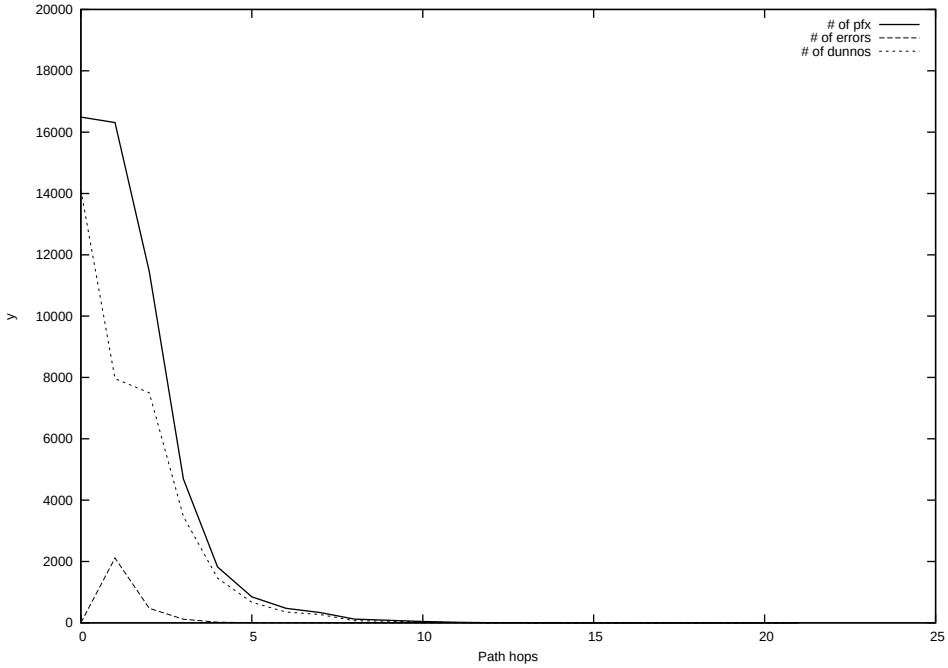
2014-02-09



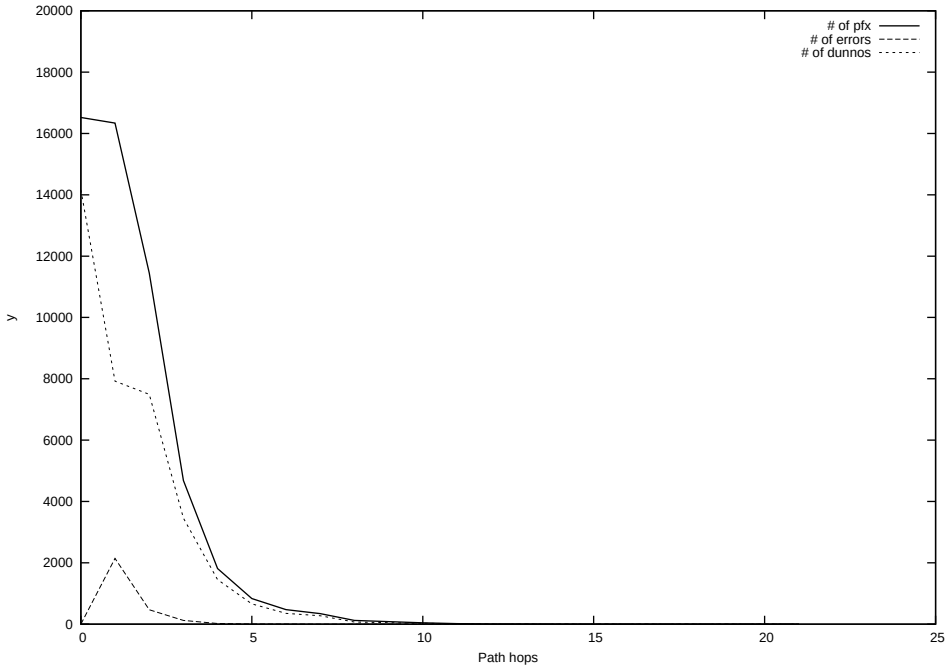
2014-02-10



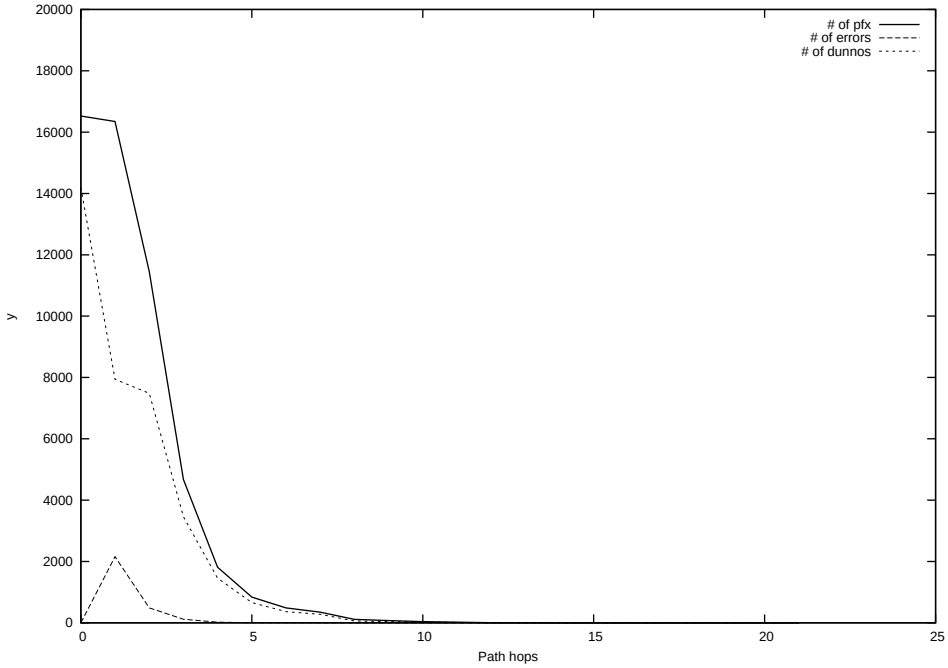
2014-02-11



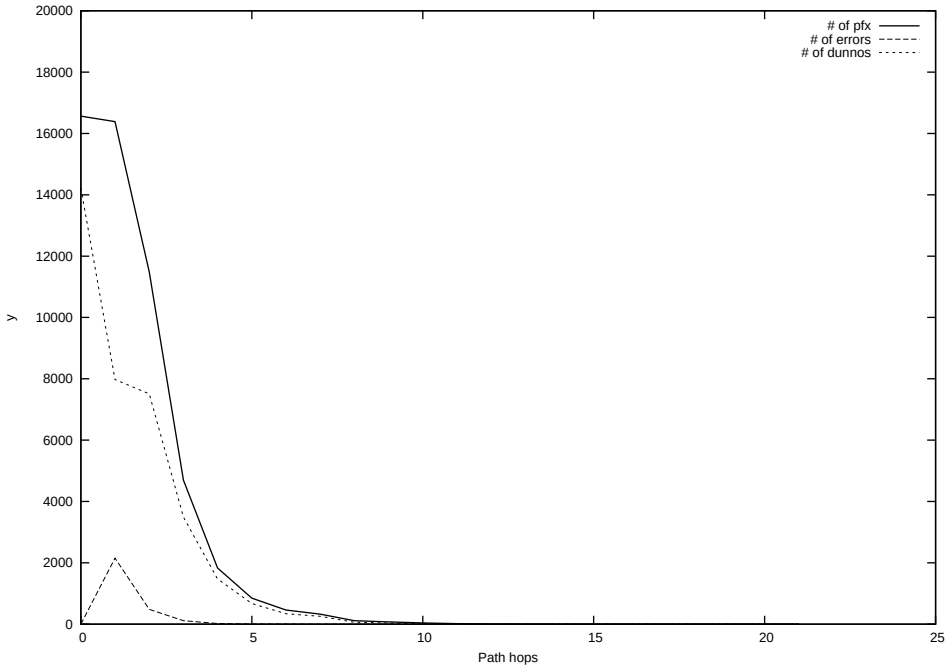
2014-02-12



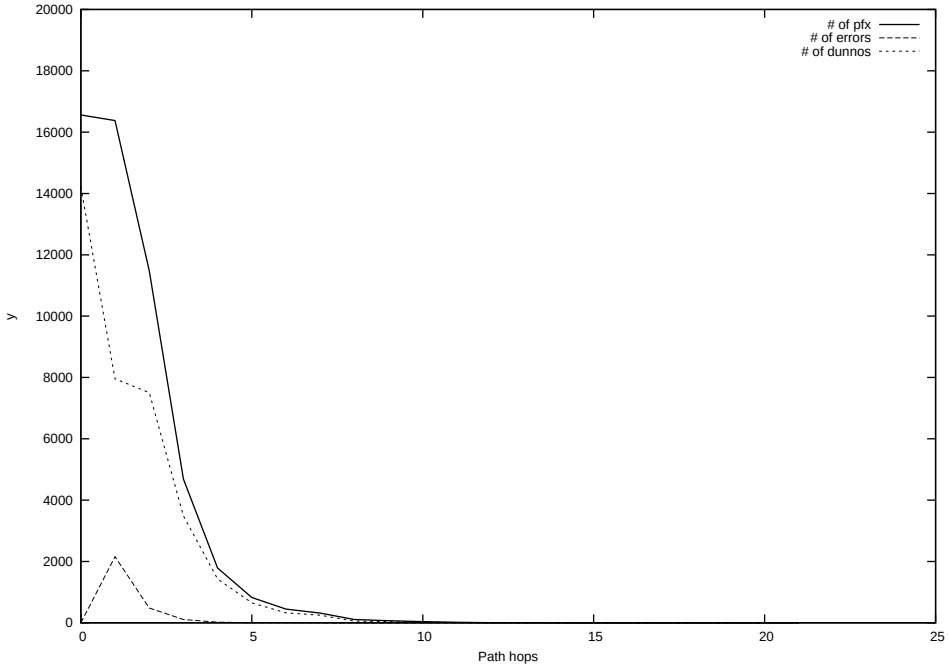
2014-02-13



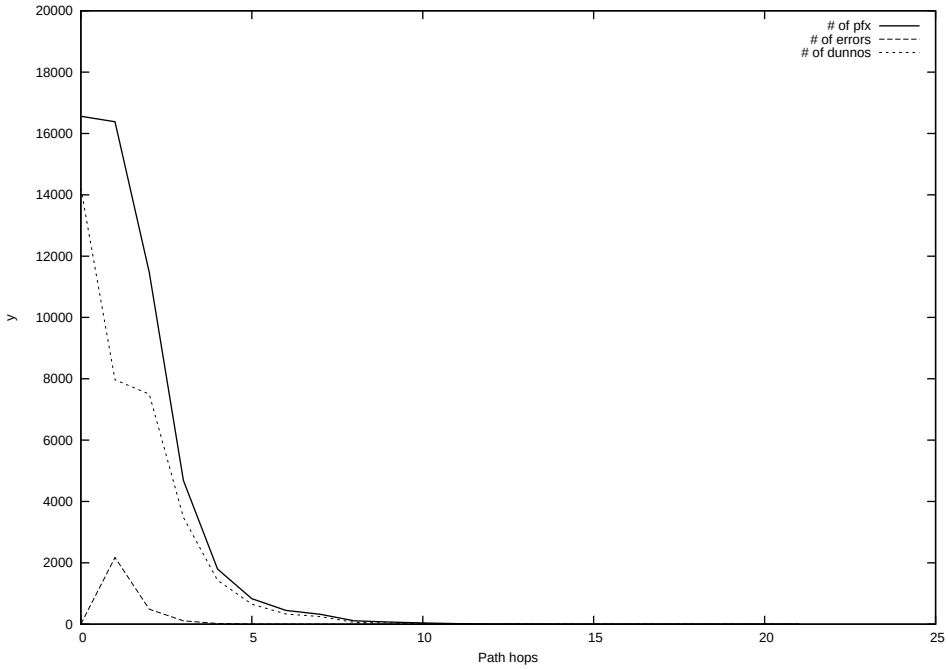
2014-02-14



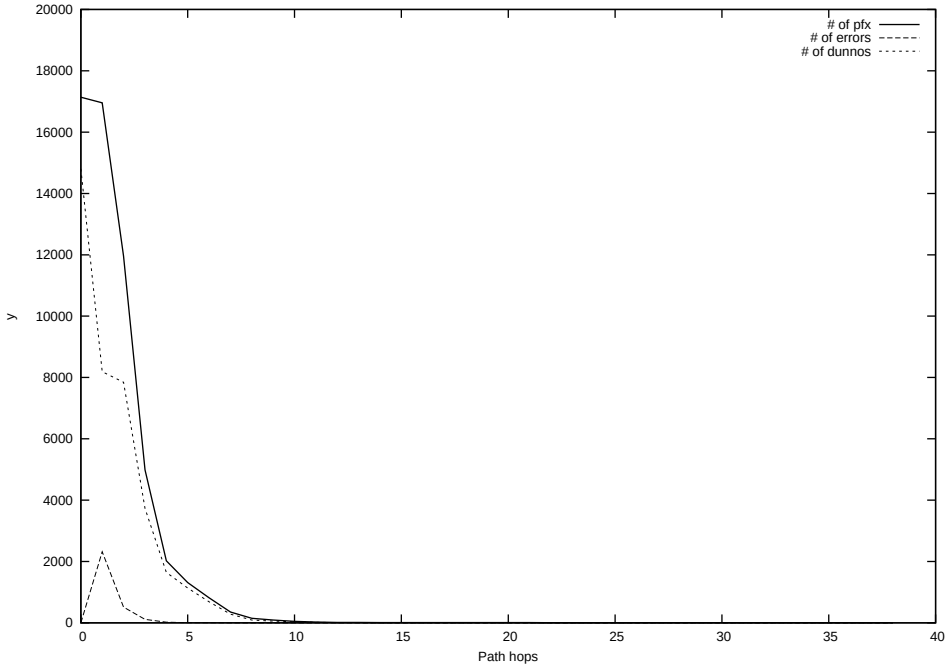
2014-02-15



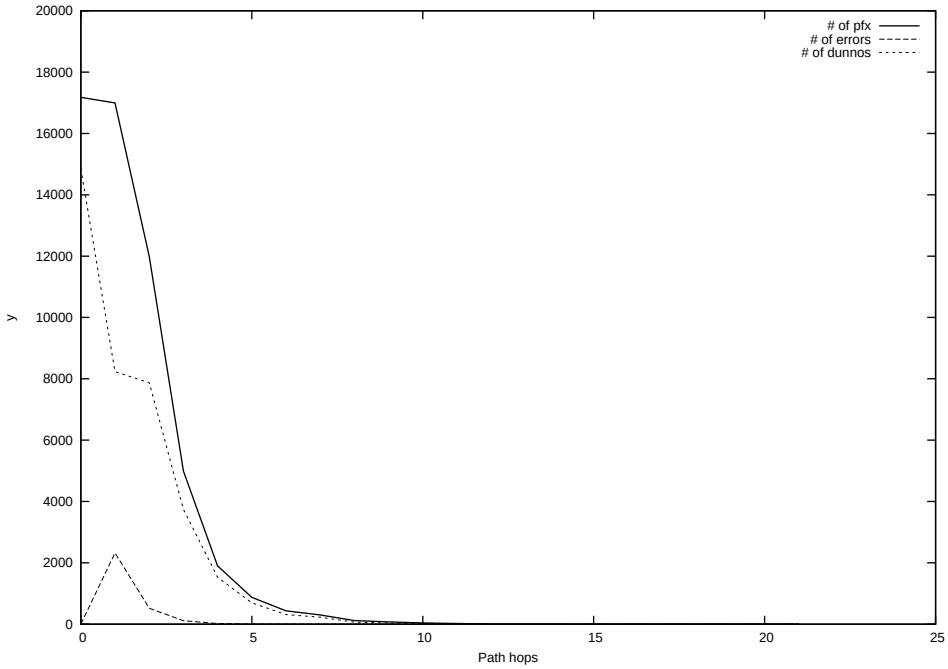
2014-02-16



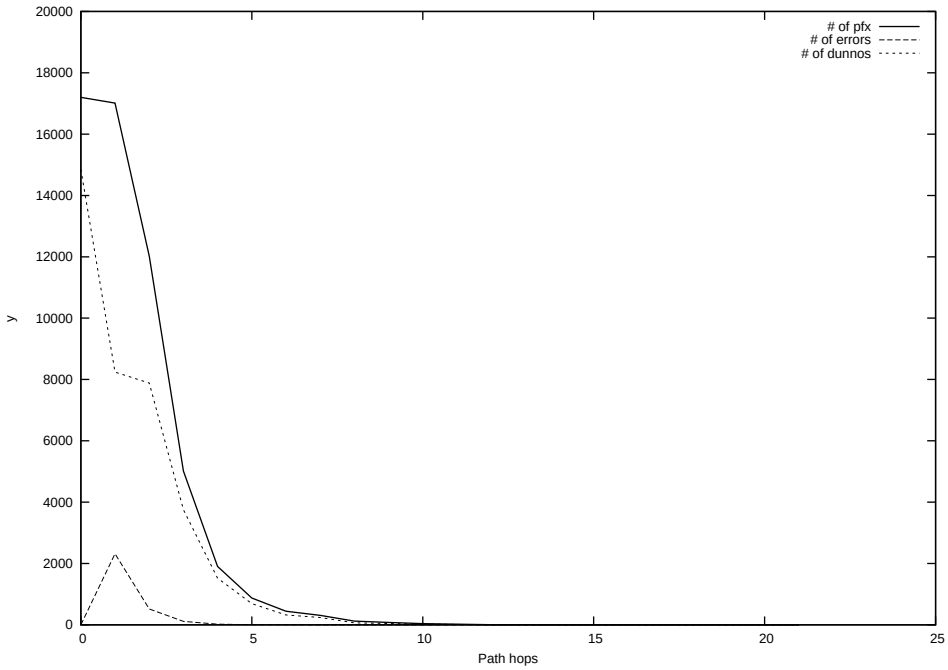
2014-02-17



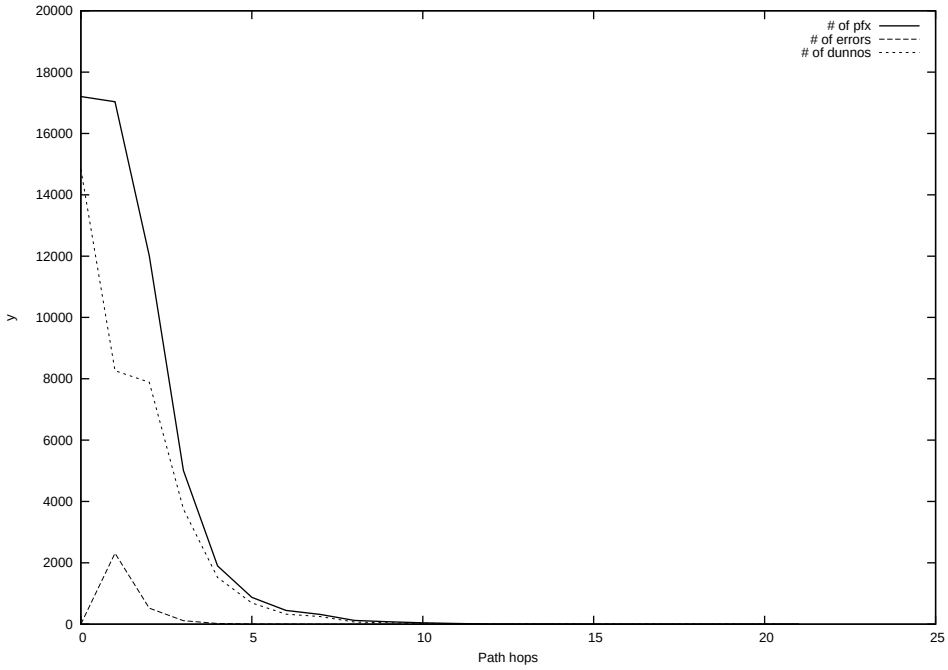
2014-03-20



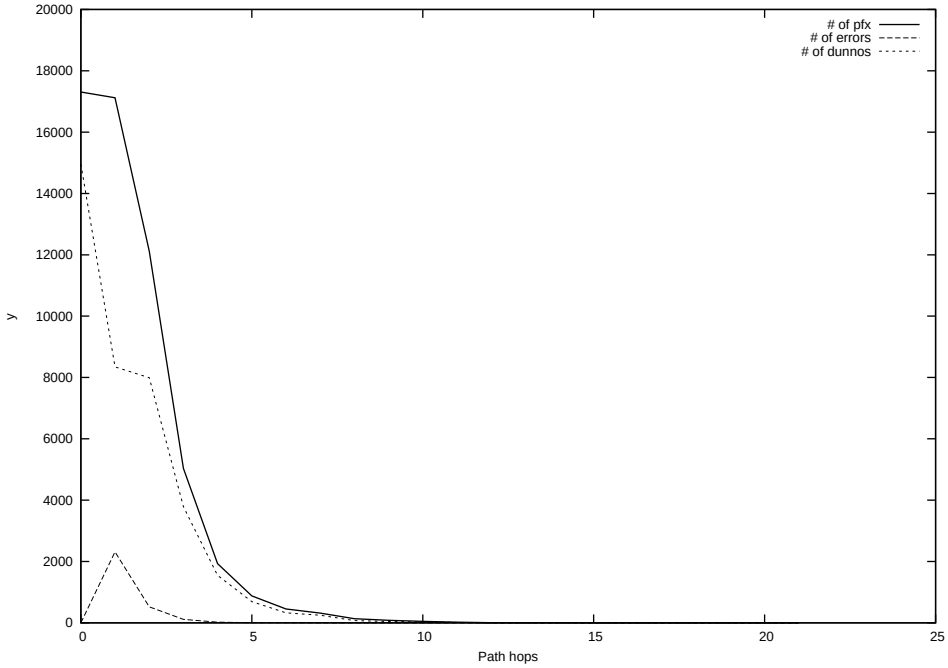
2014-03-21



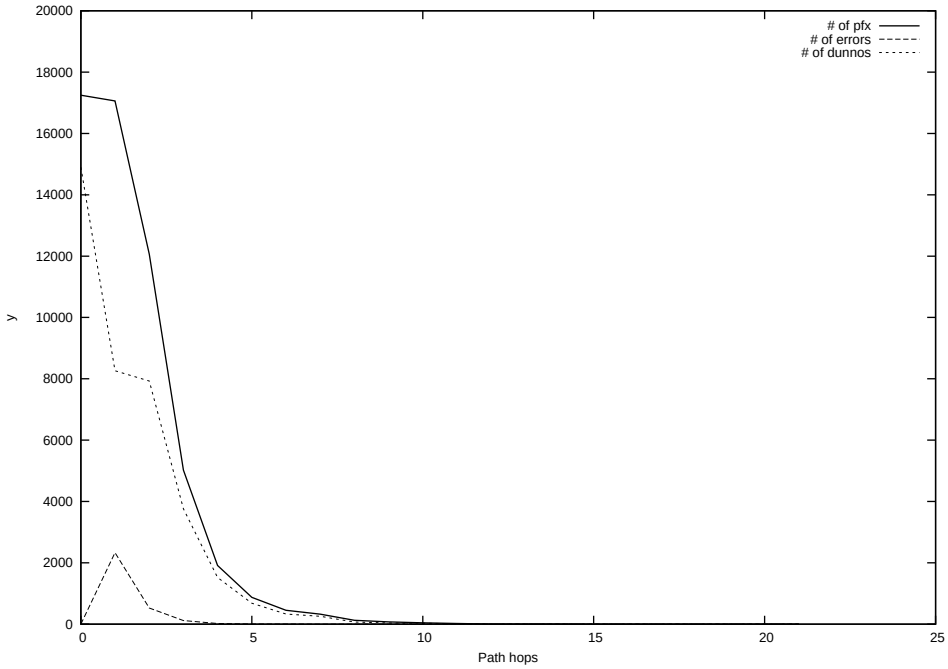
2014-03-22



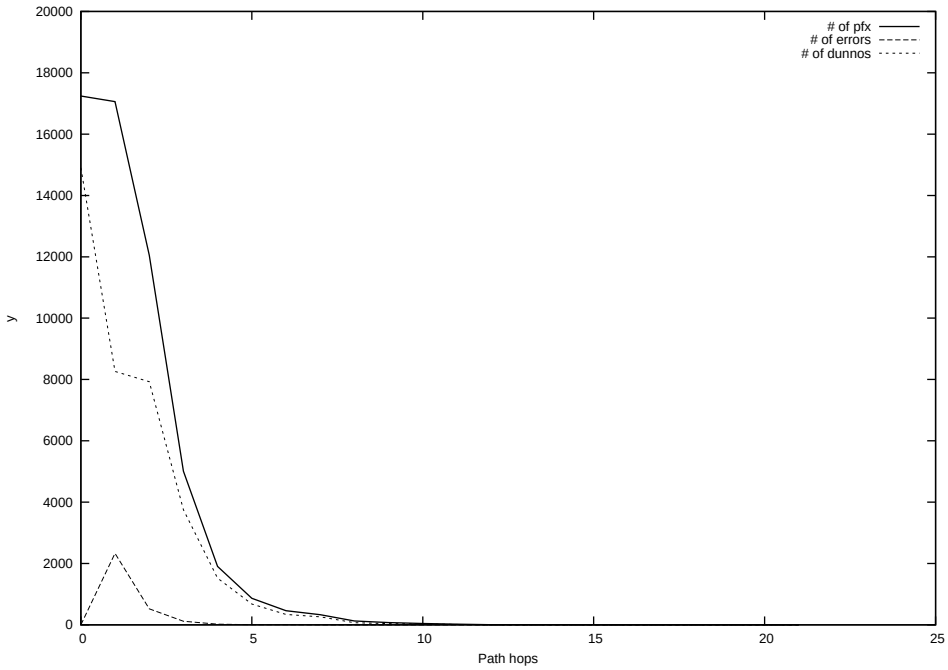
2014-03-23



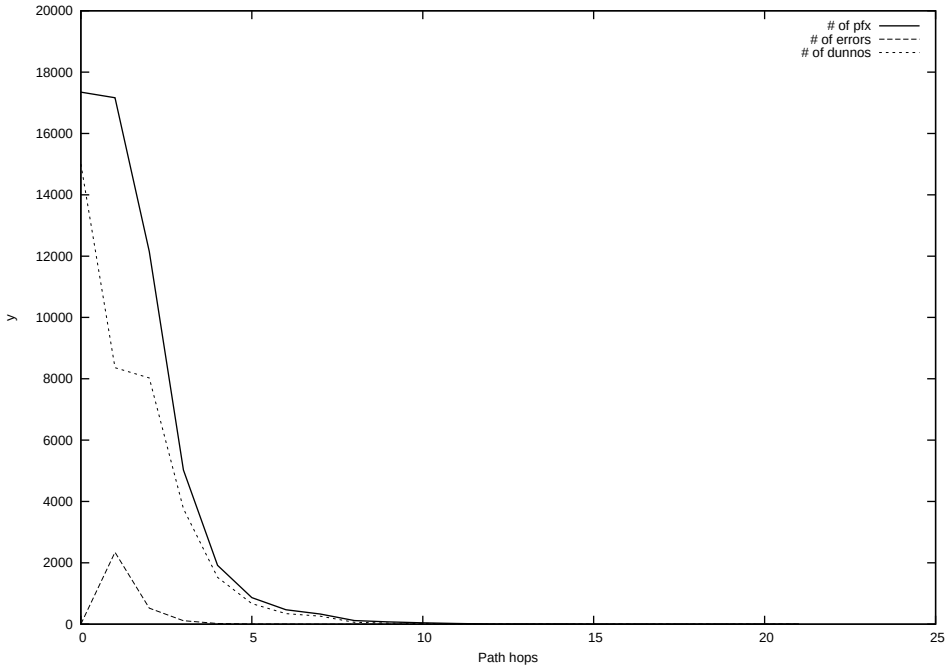
2014-03-24



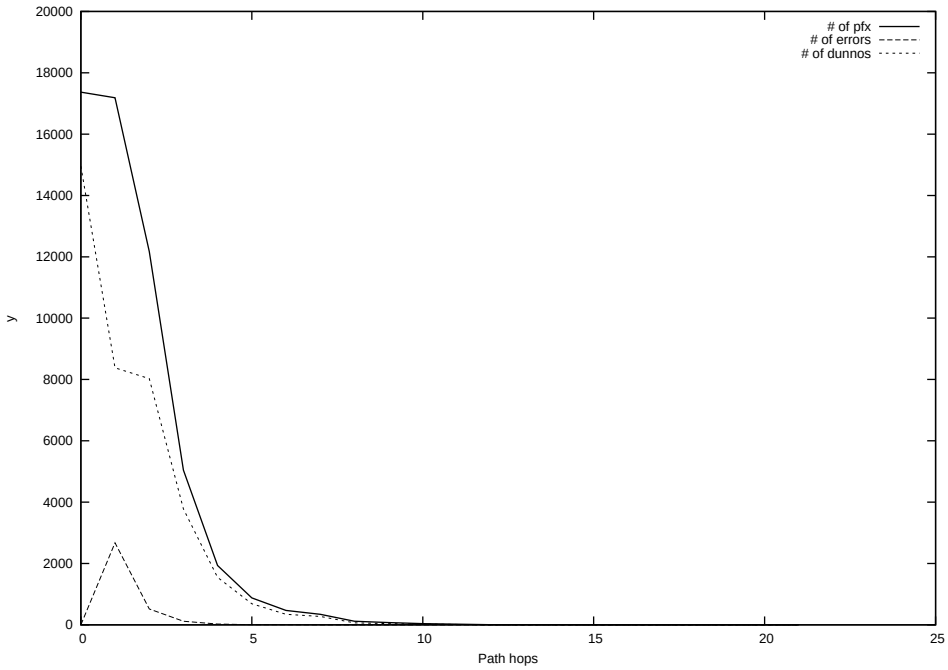
2014-03-25



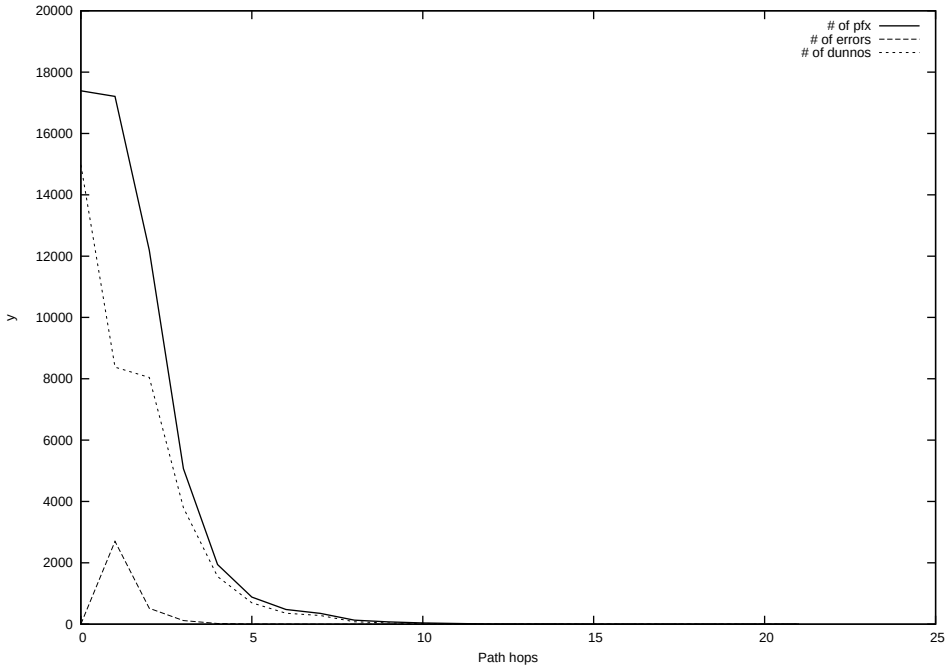
2014-03-26



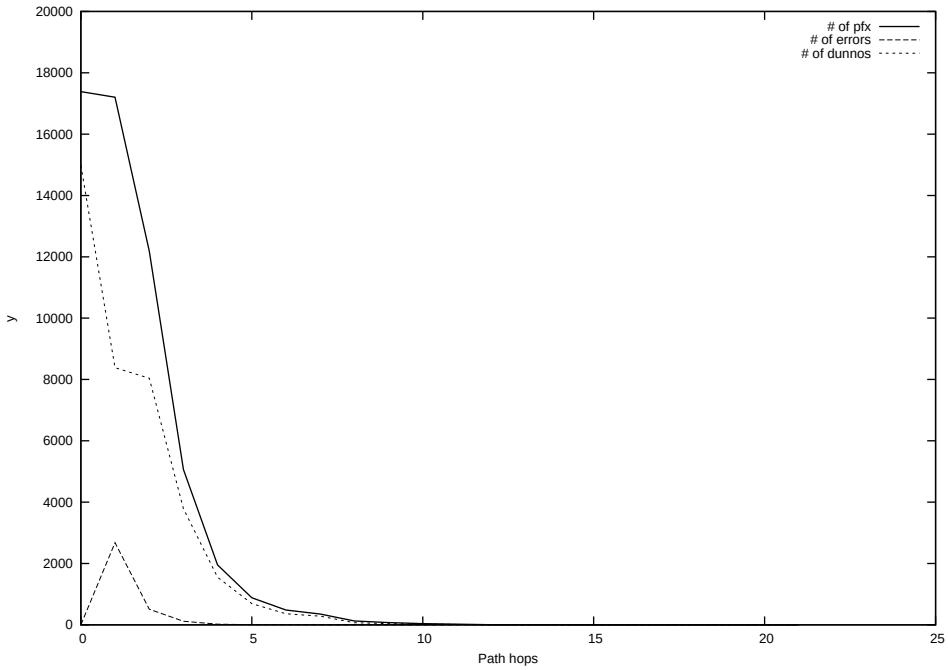
2014-03-27



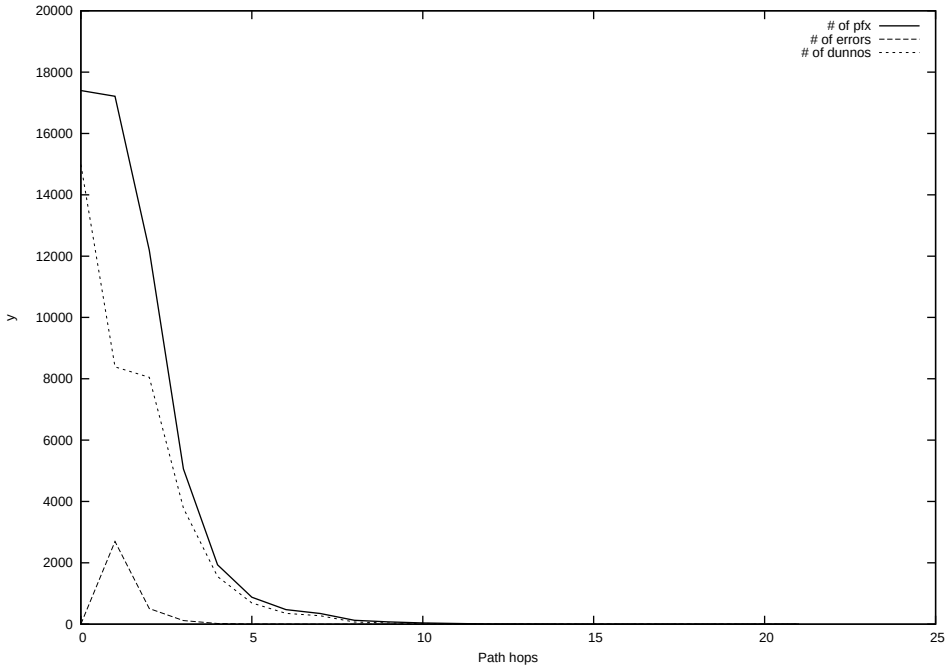
2014-03-28



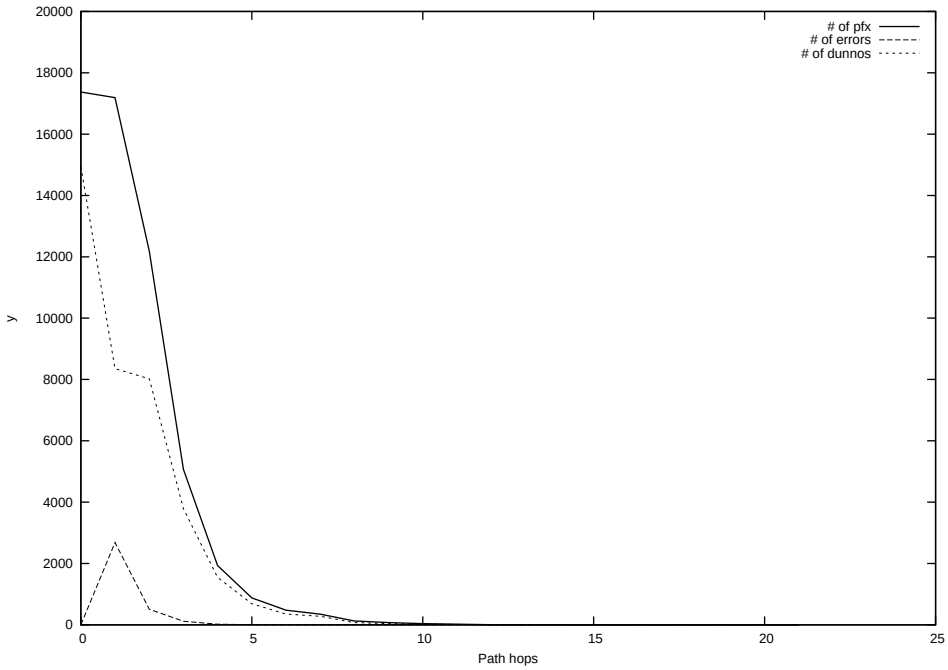
2014-03-29



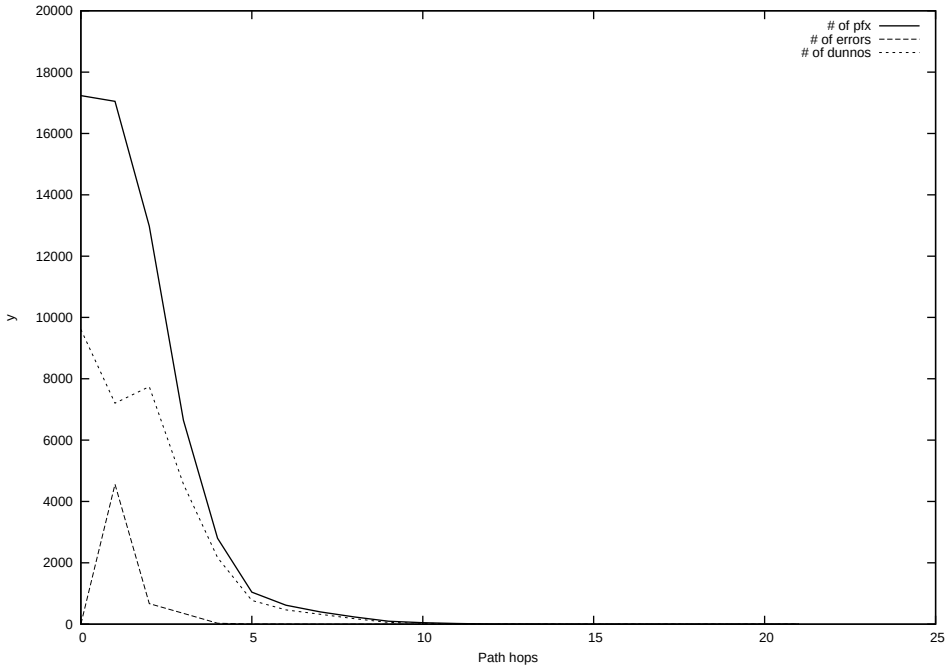
2014-03-30



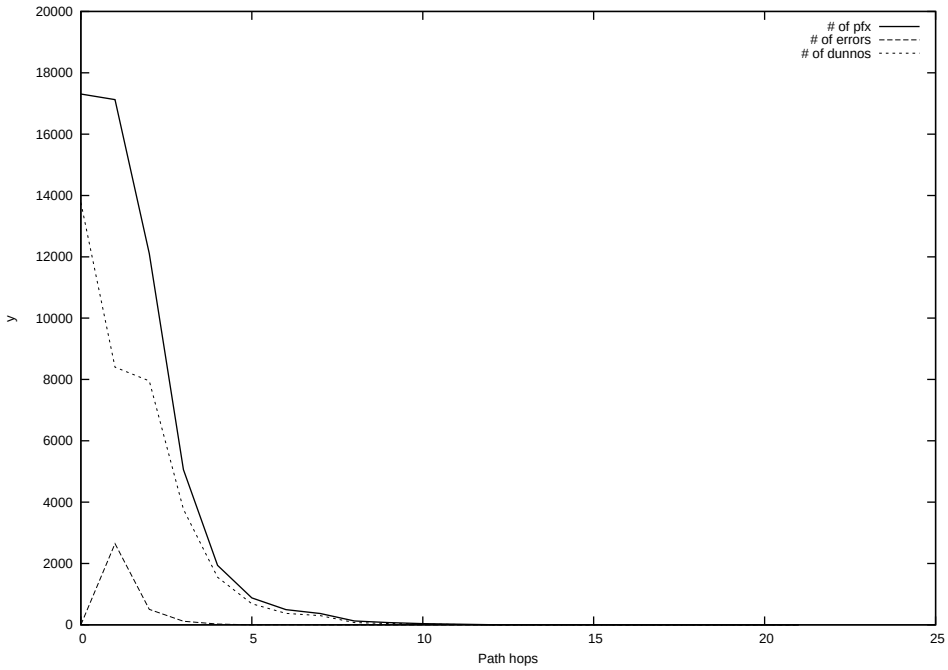
2014-03-31



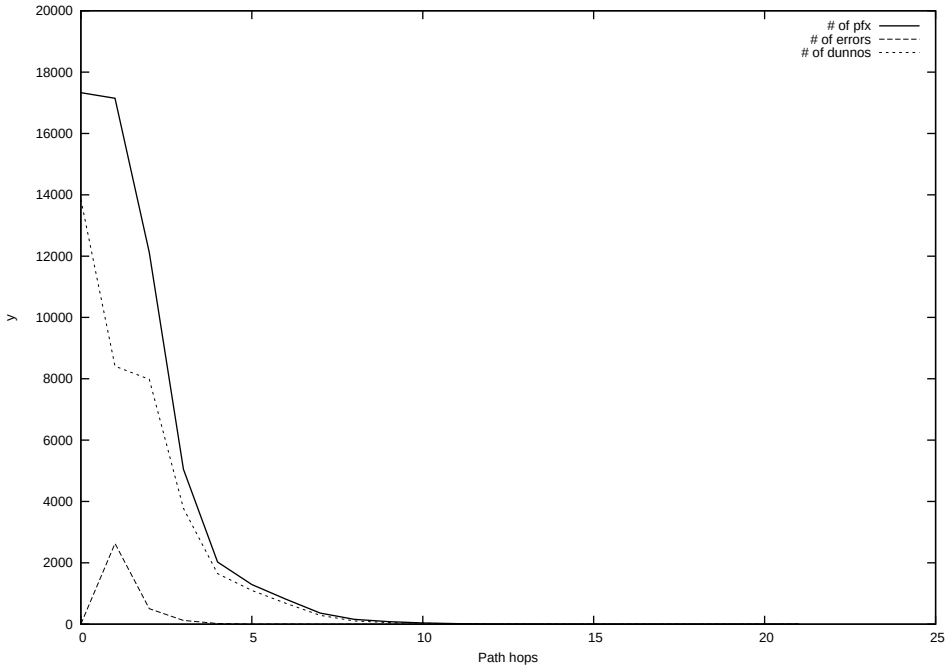
2014-04-01



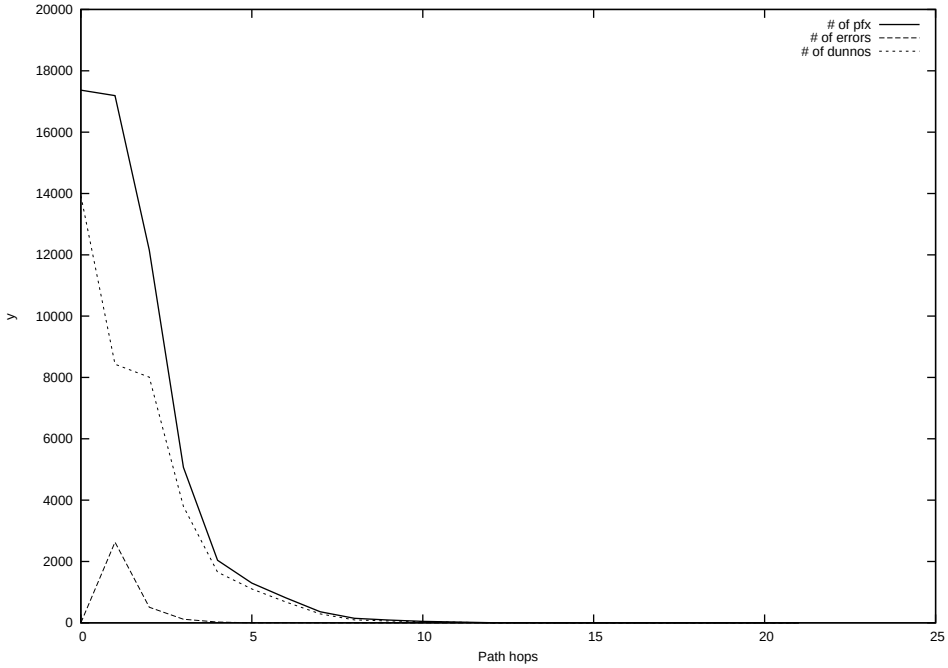
2014-04-02



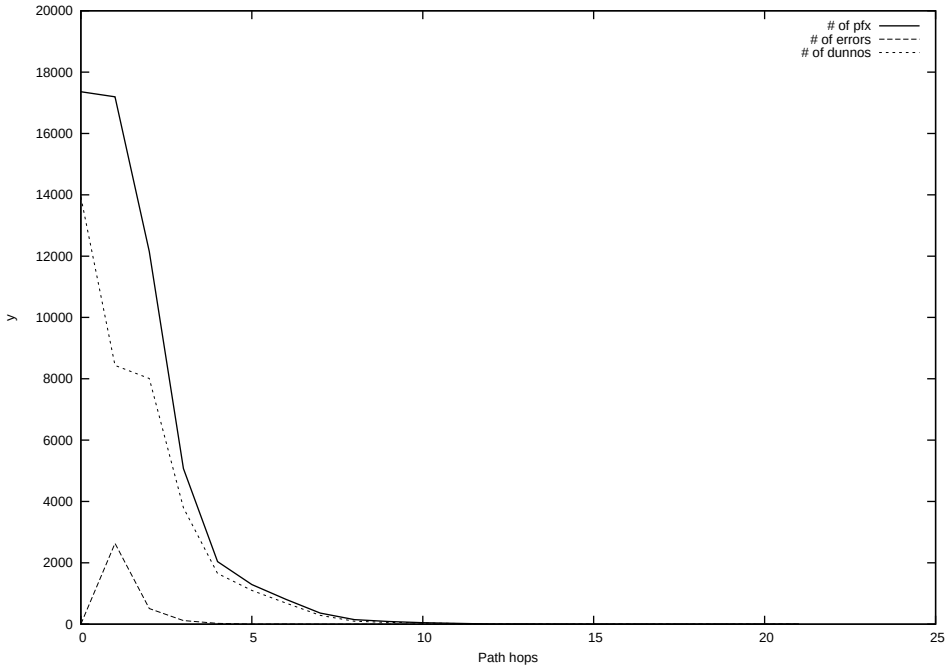
2014-04-03



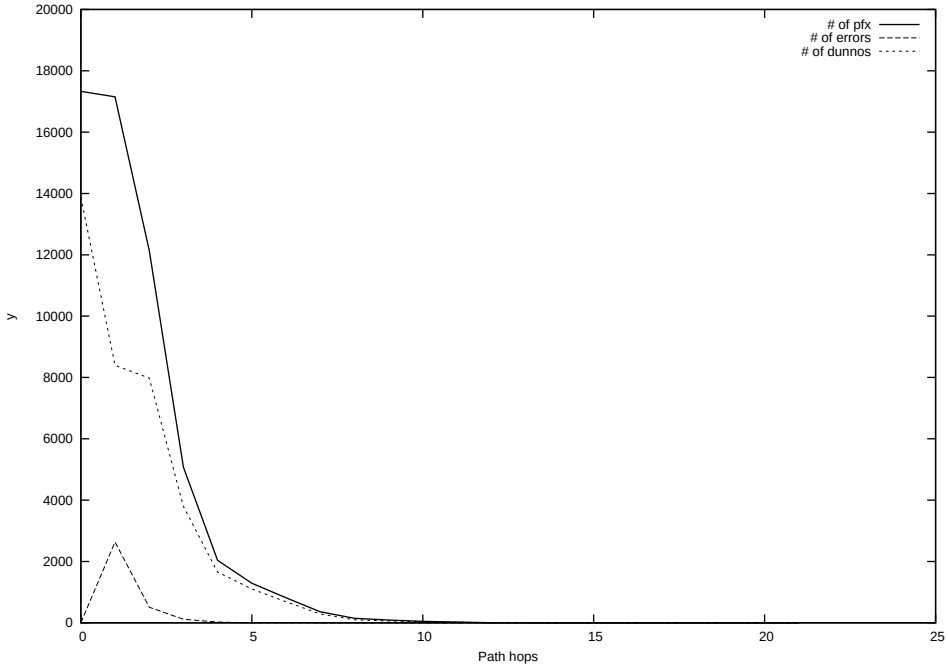
2014-04-04



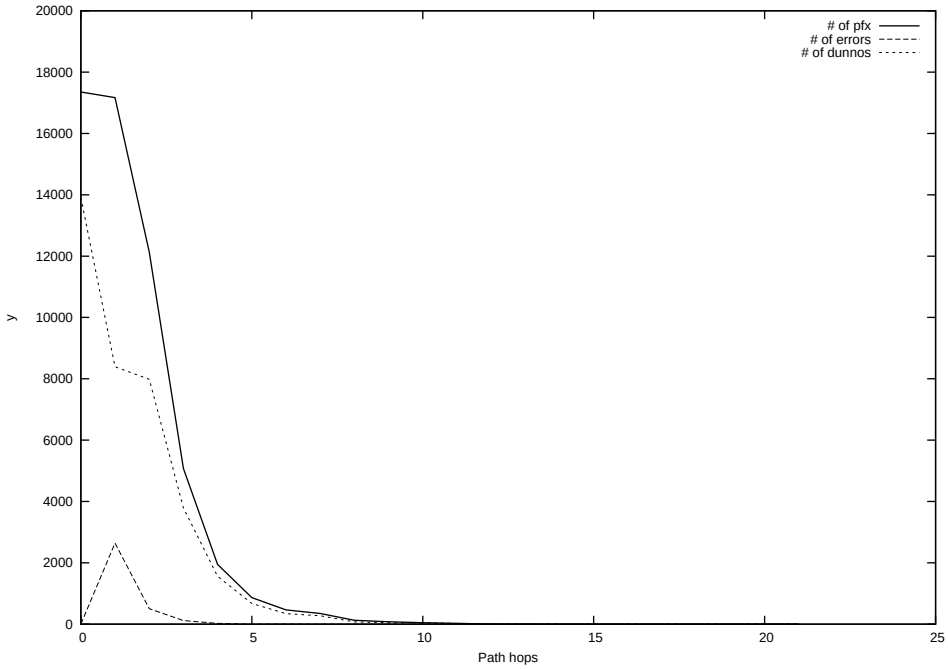
2014-04-05



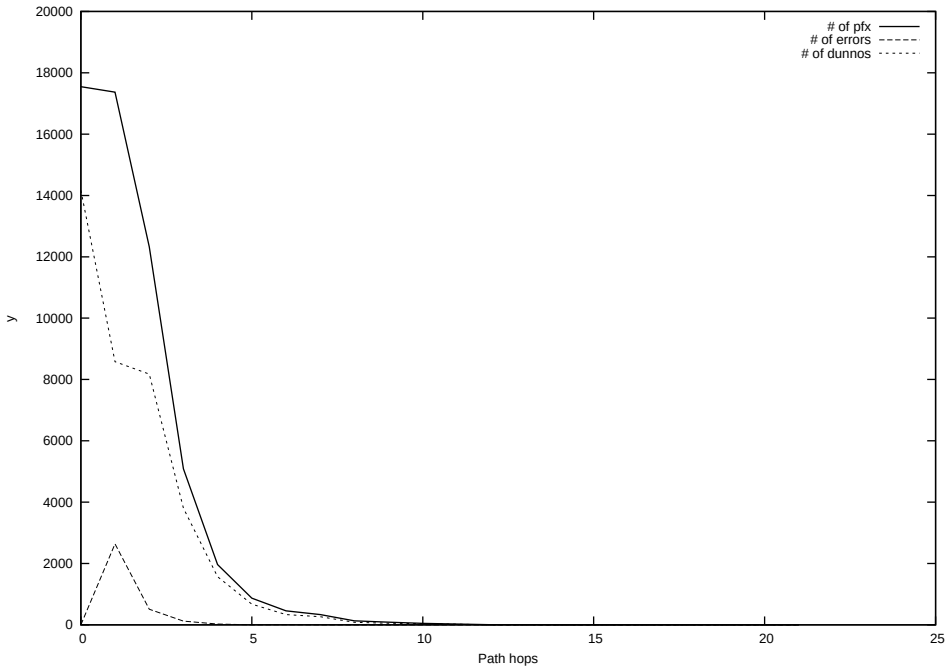
2014-04-06



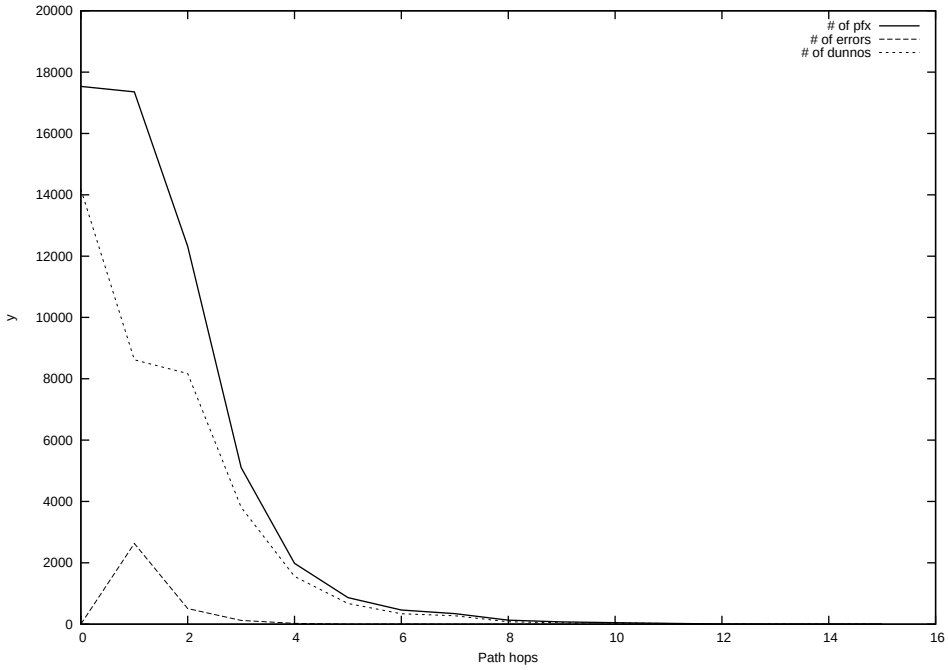
2014-04-07



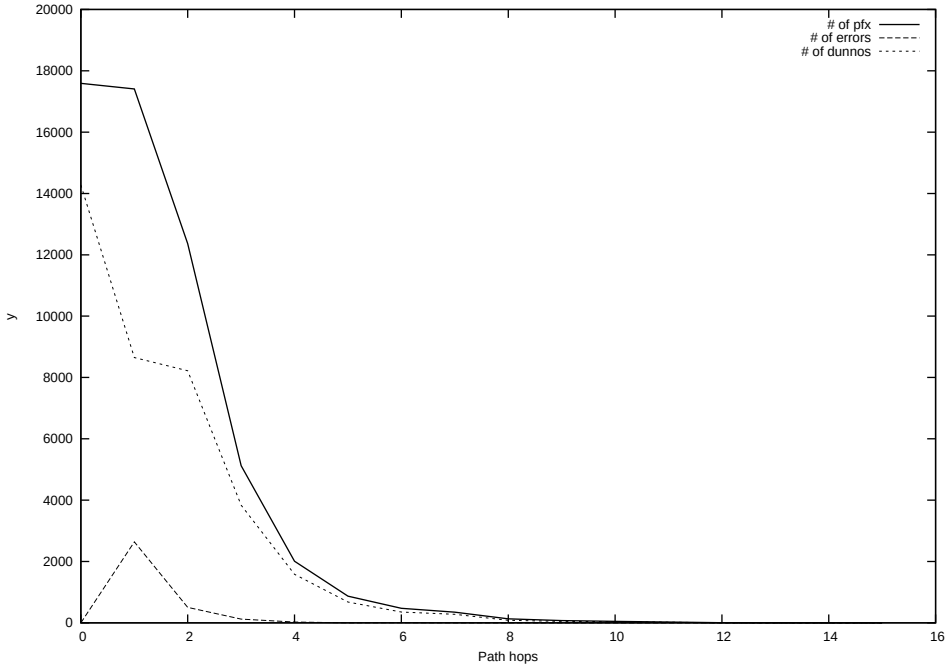
2014-04-08



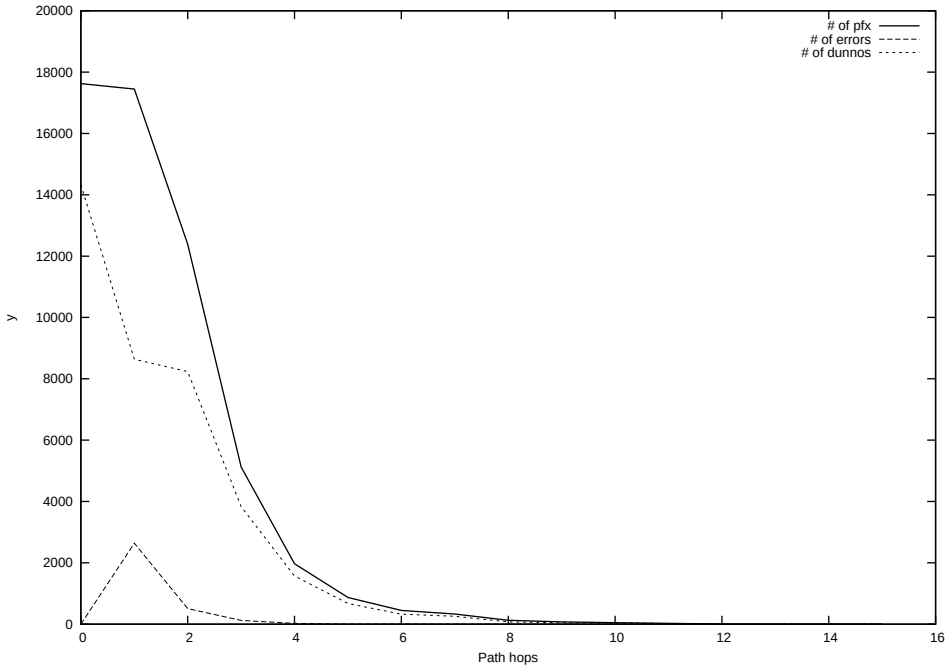
2014-04-09



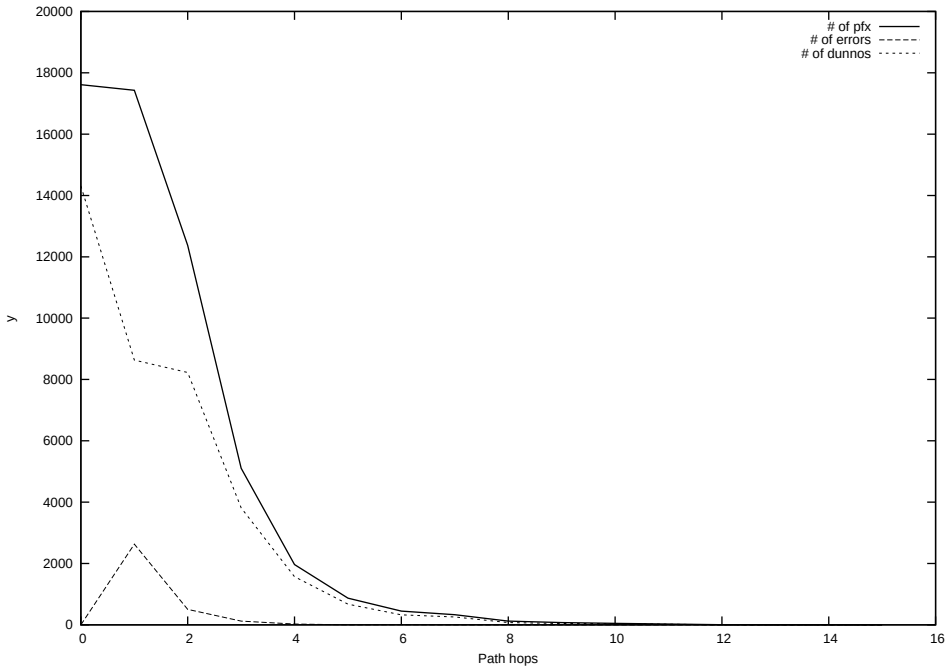
2014-04-10



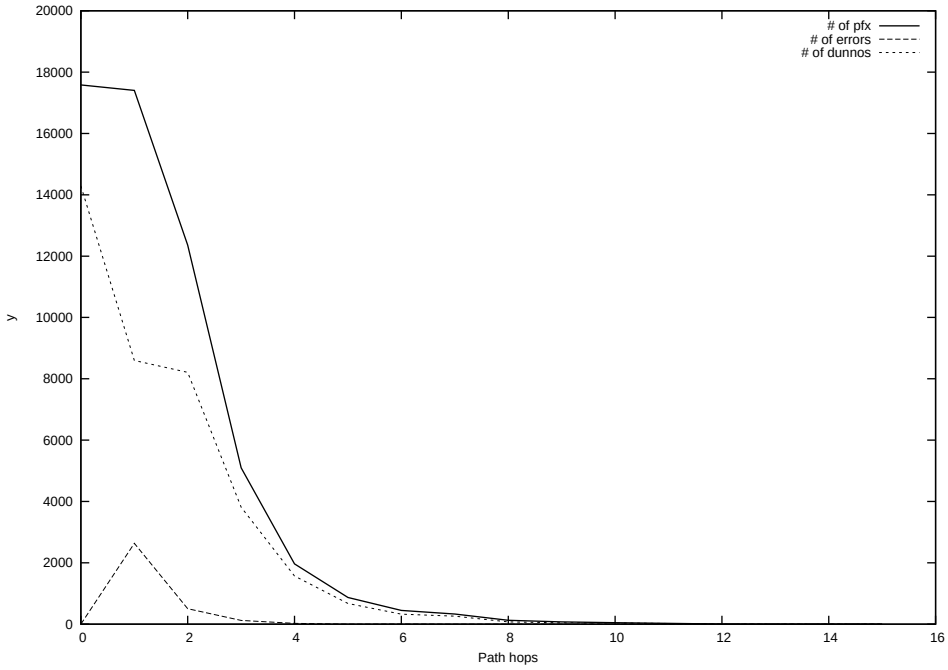
2014-04-11



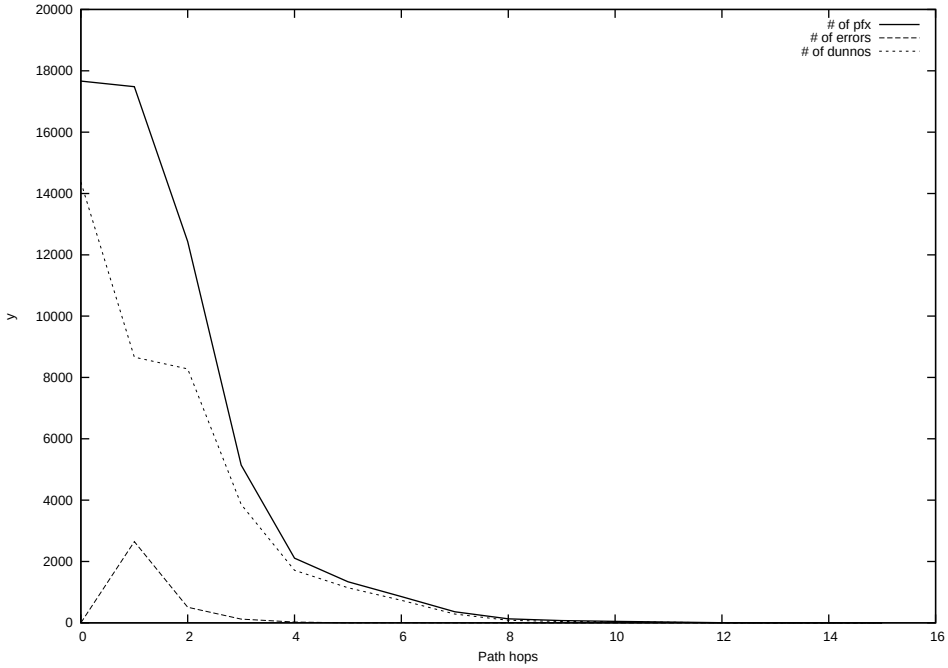
2014-04-12



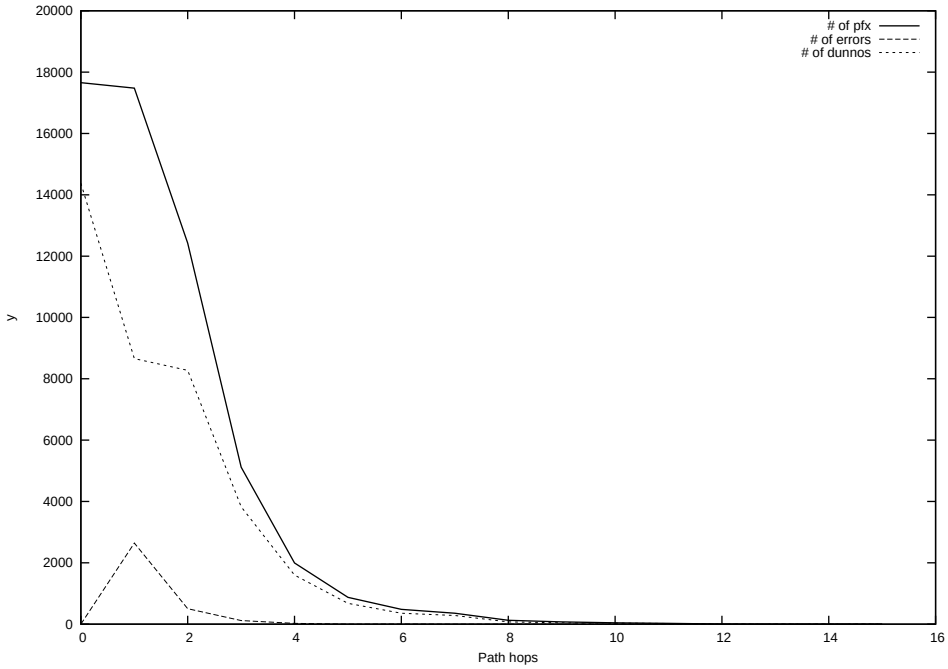
2014-04-13



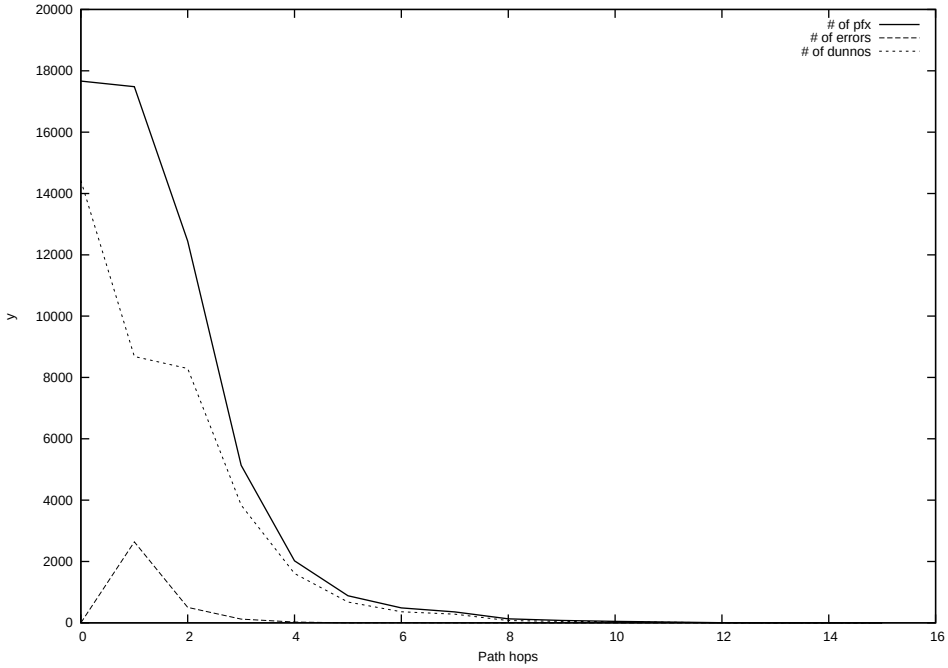
2014-04-14



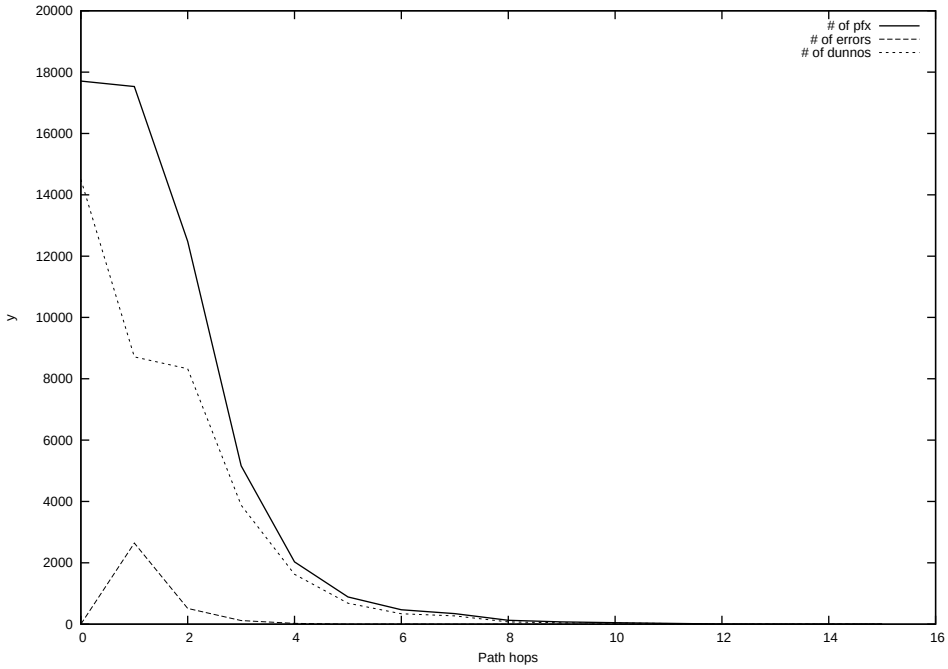
2014-04-15



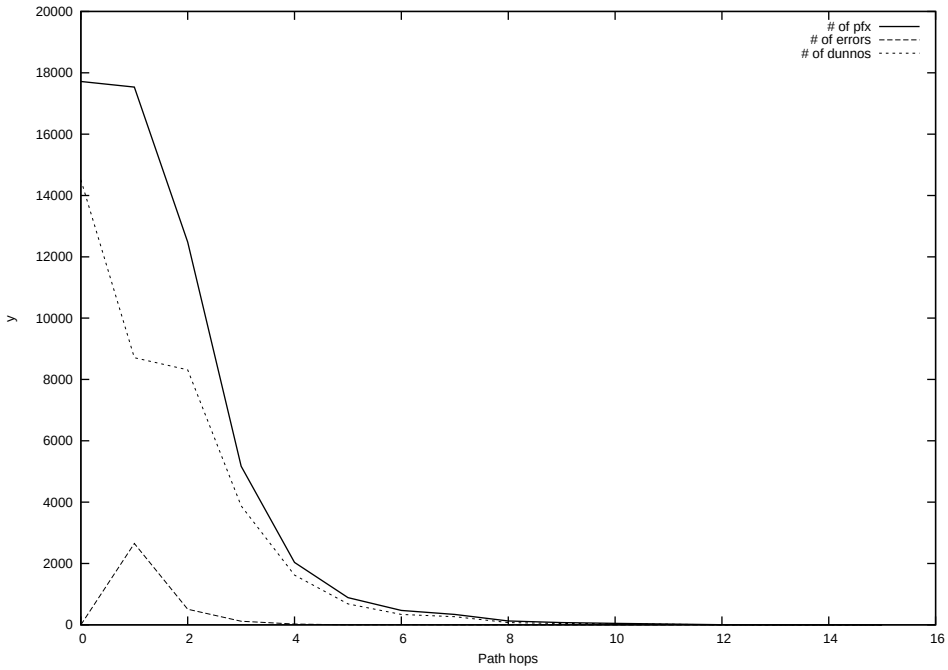
2014-04-16



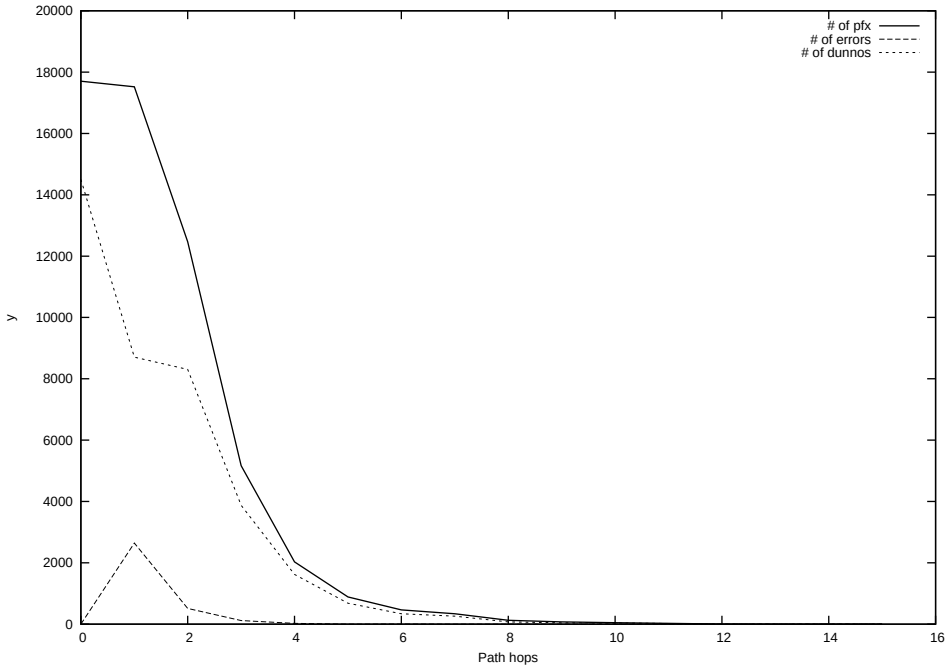
2014-04-17



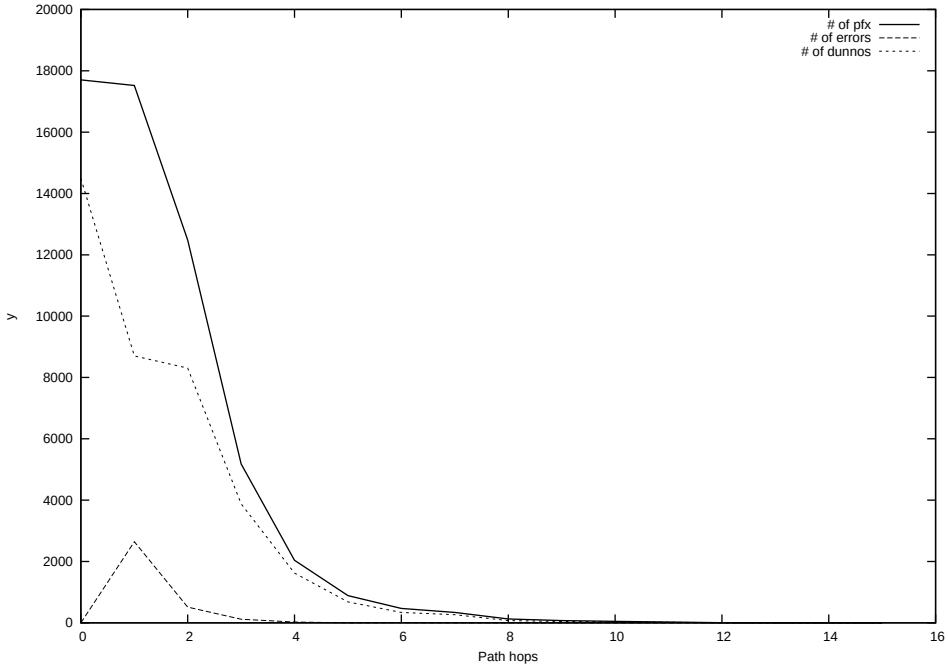
2014-04-18



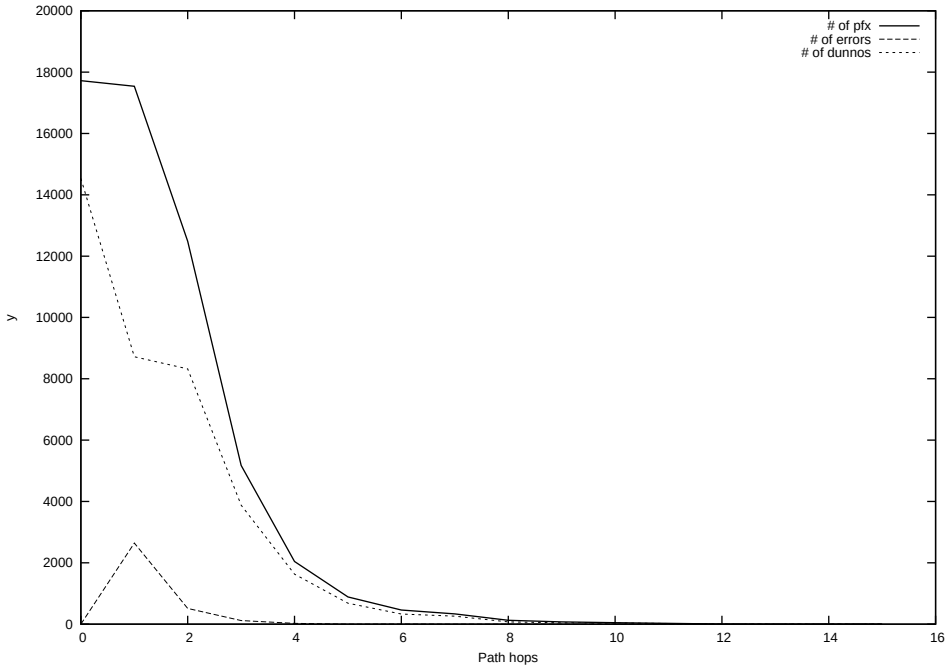
2014-04-19



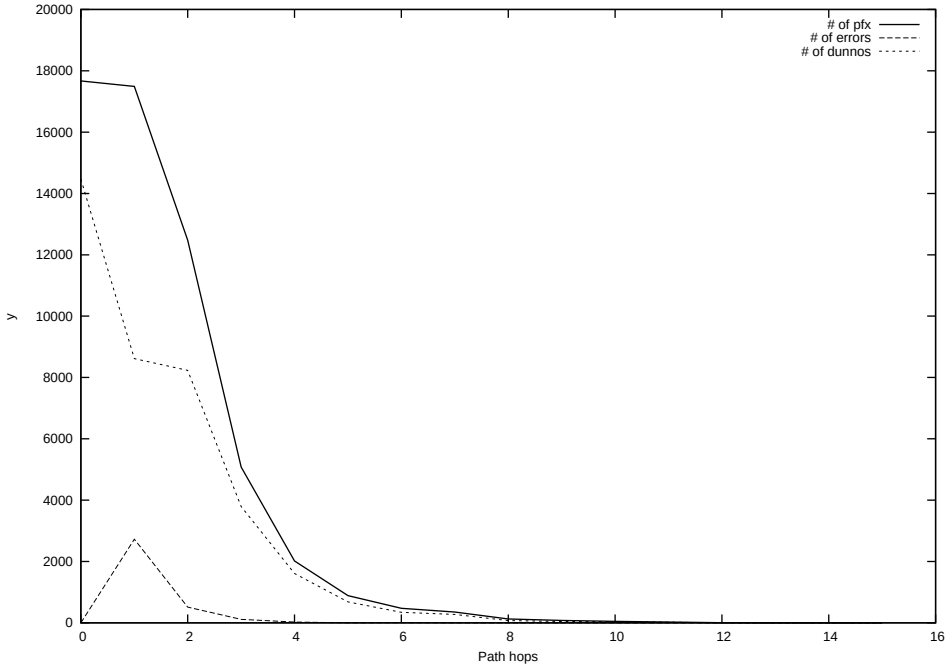
2014-04-20



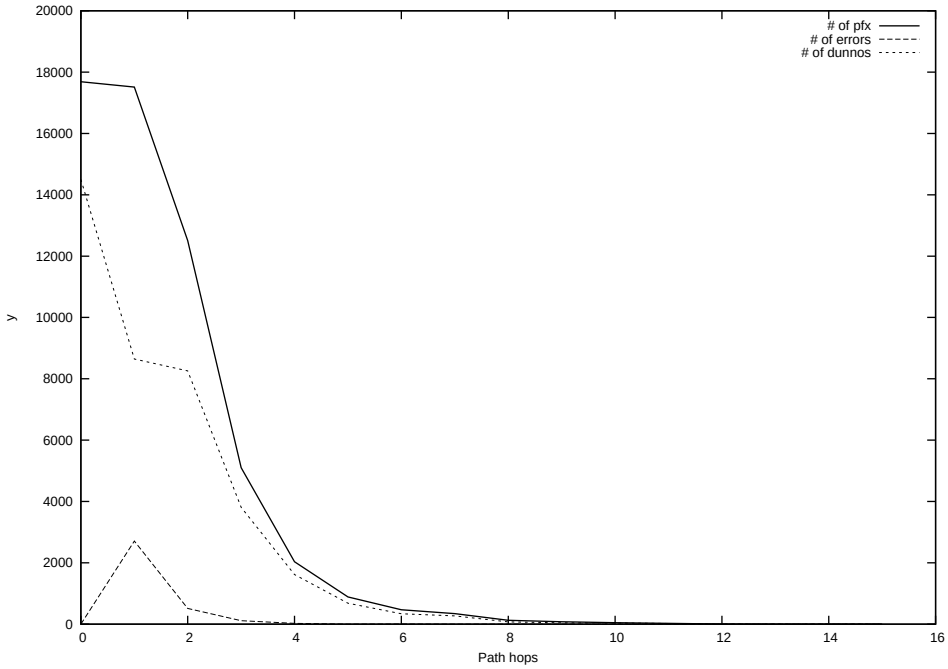
2014-04-21



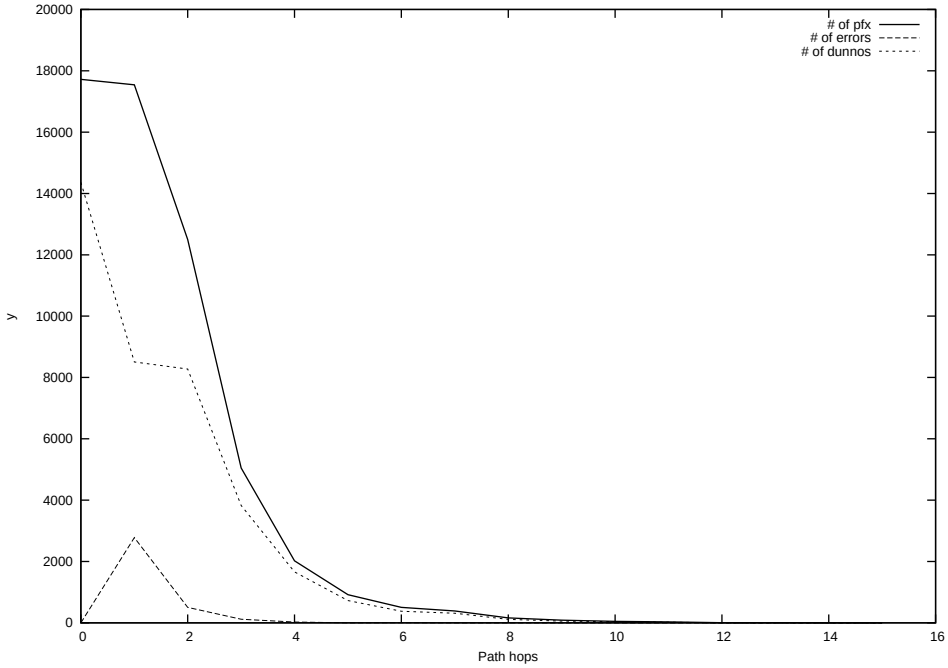
2014-04-22



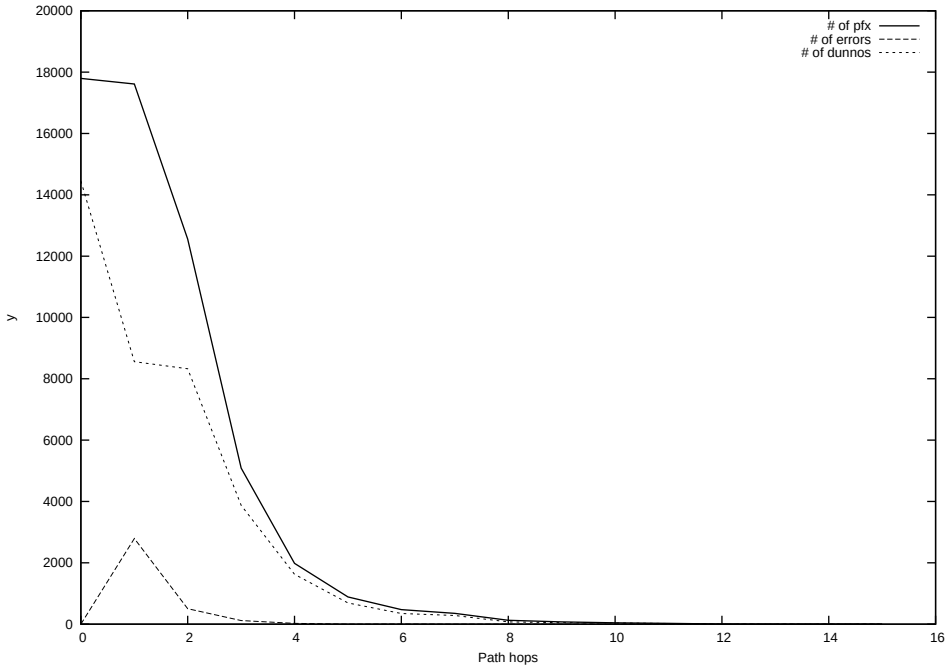
2014-04-23



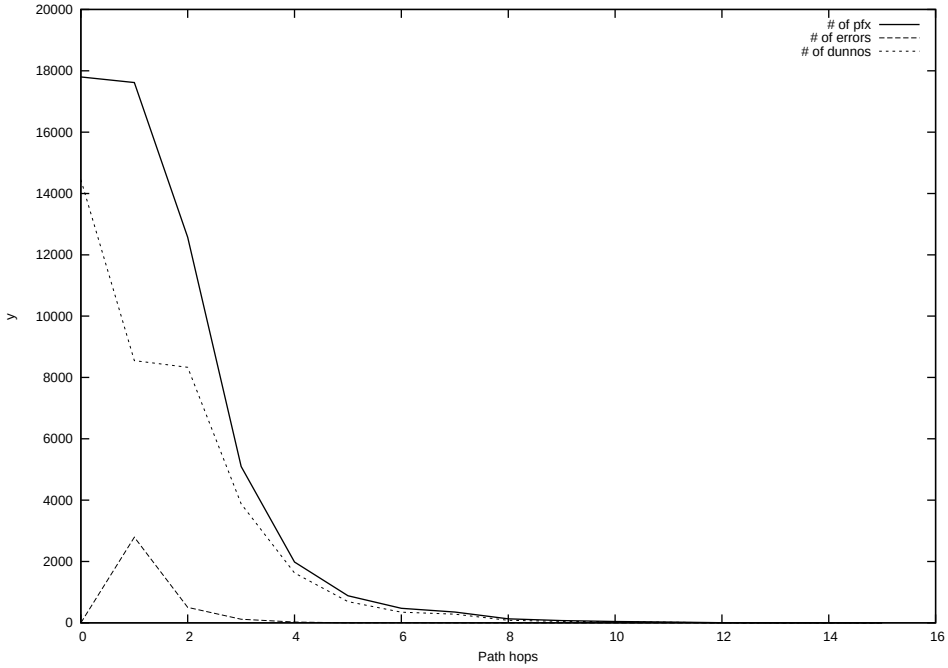
2014-04-24



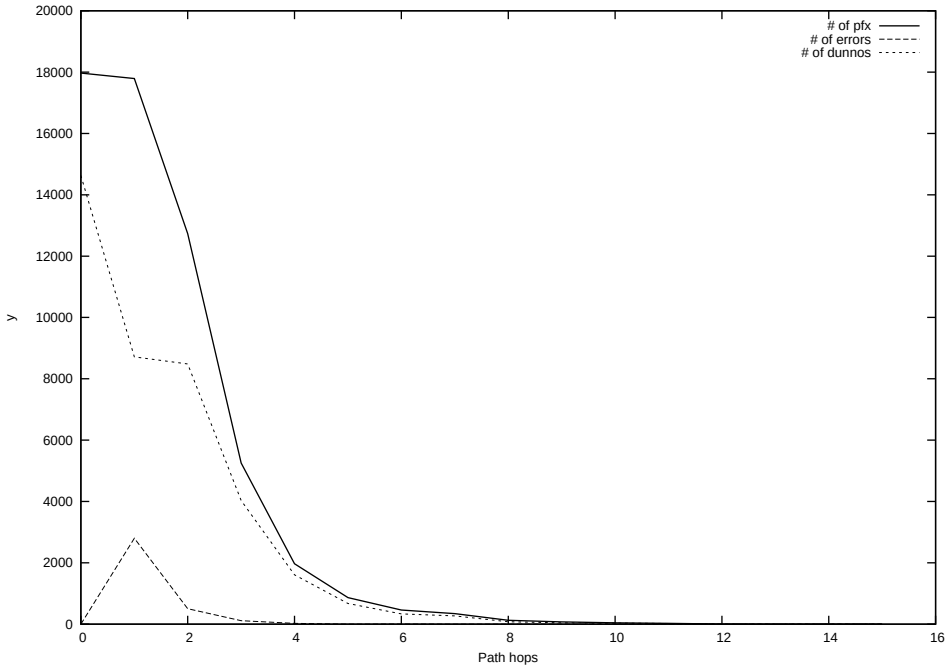
2014-04-25



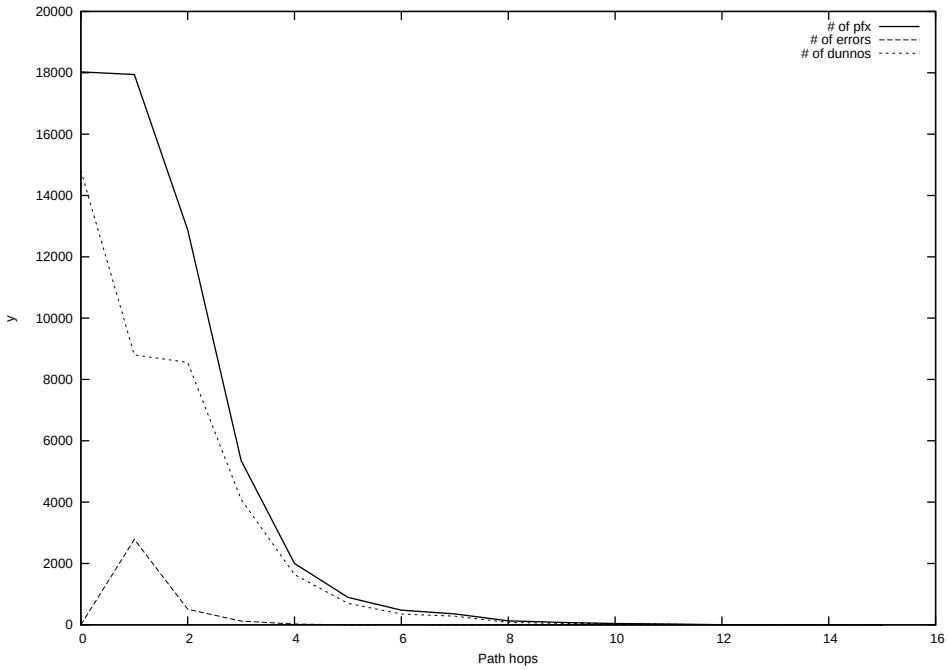
2014-04-26



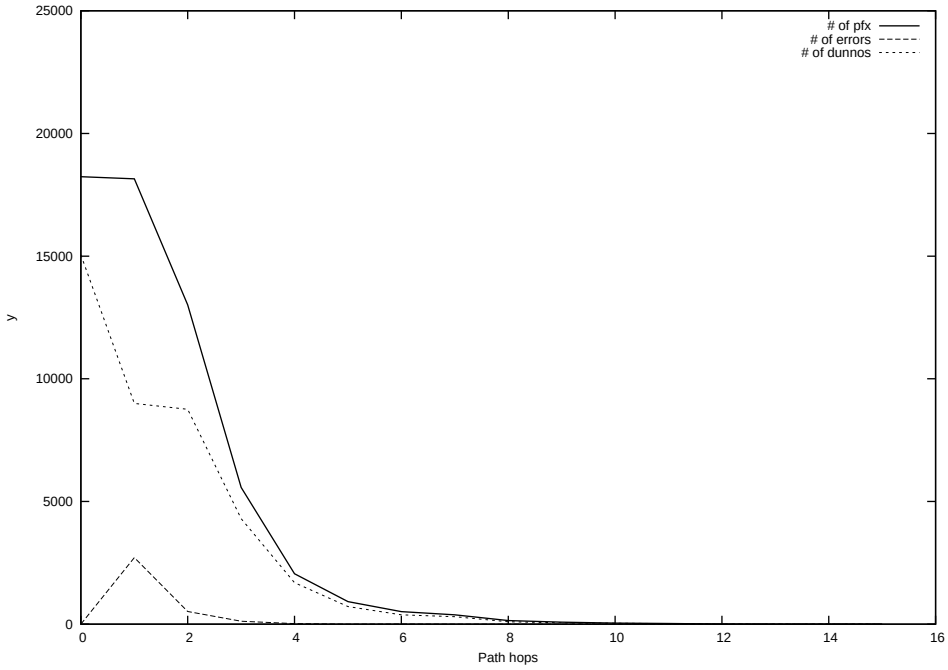
2014-04-27



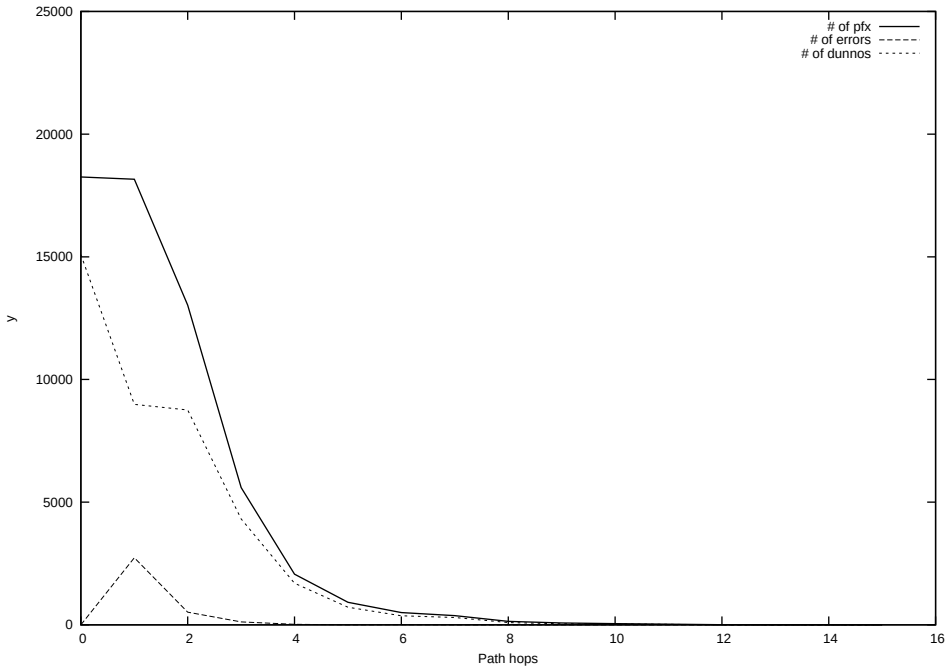
2014-04-28



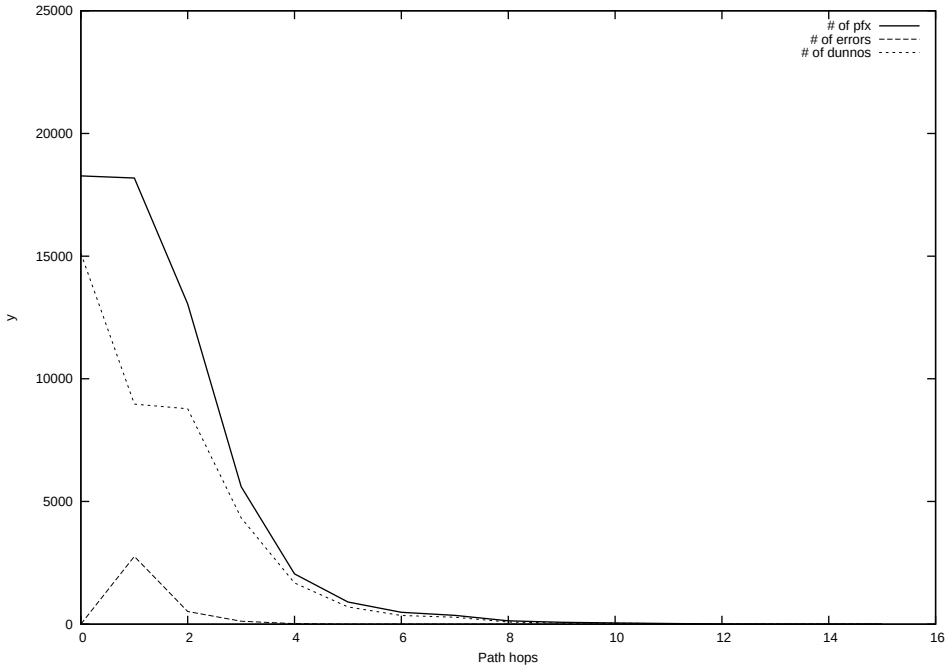
2014-04-29



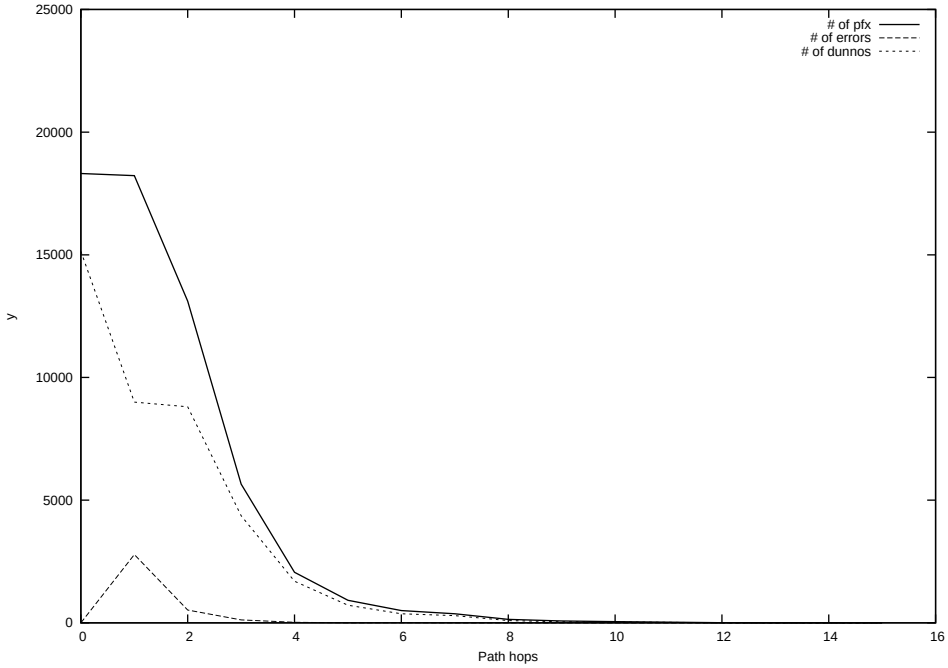
2014-04-30



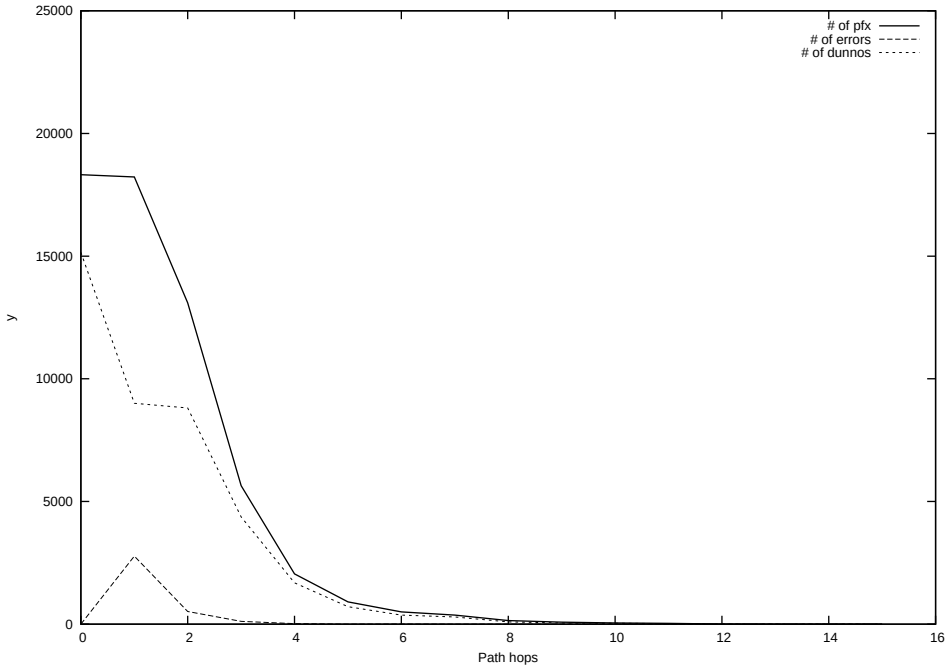
2014-05-01



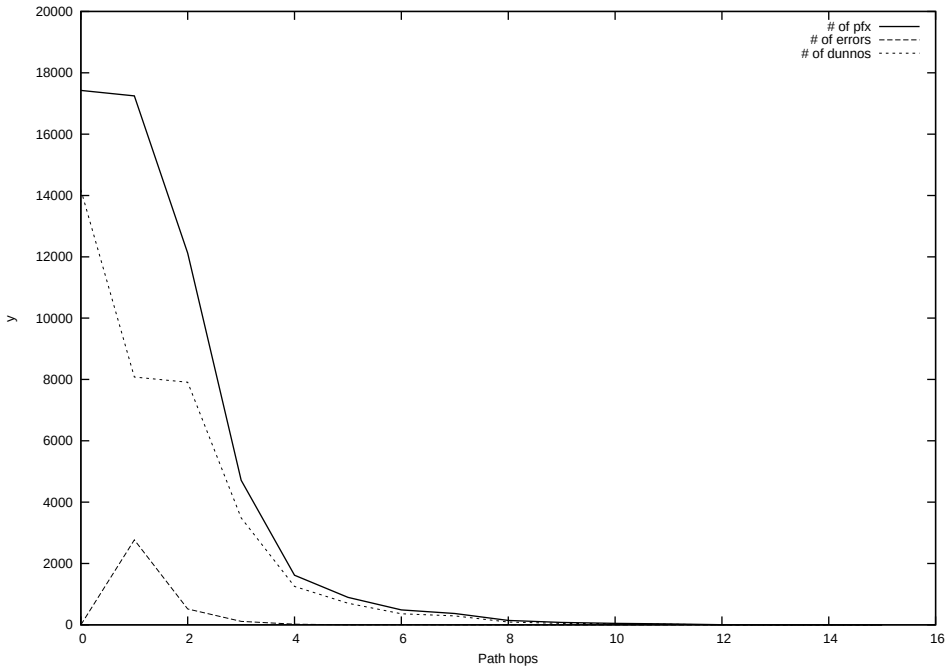
2014-05-02



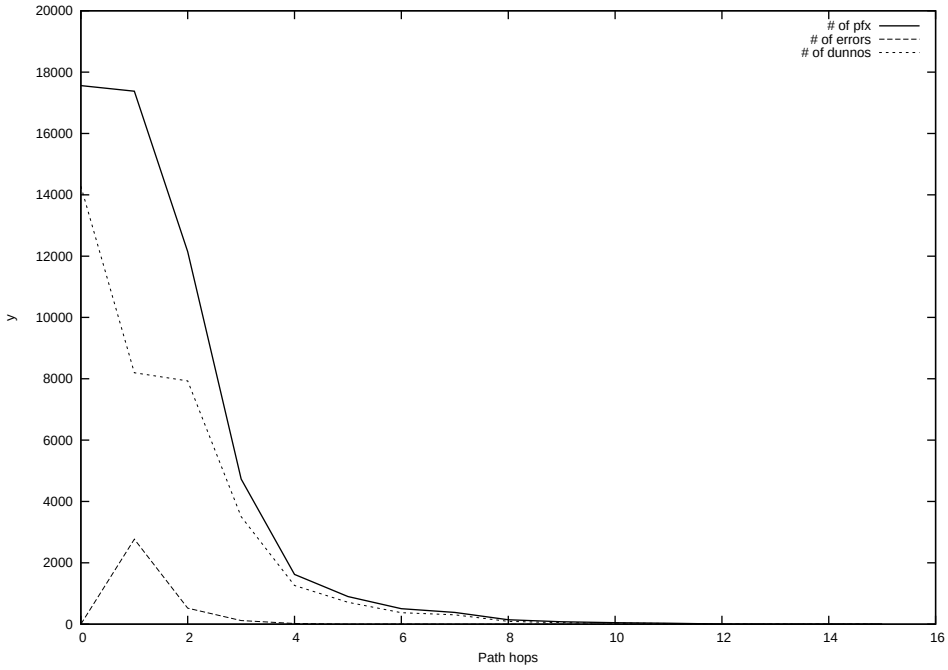
2014-05-03



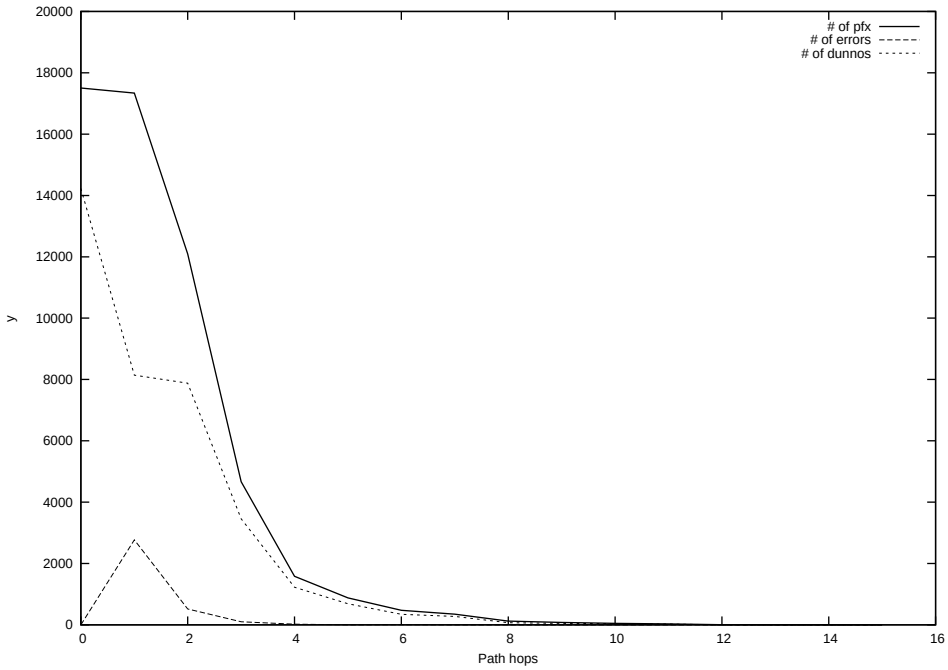
2014-05-04



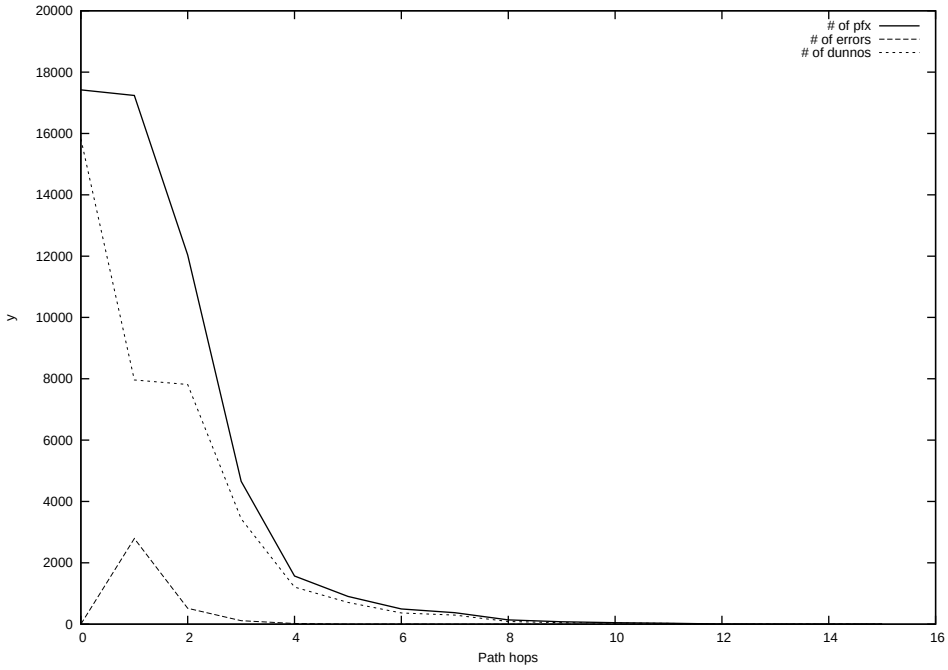
2014-05-05



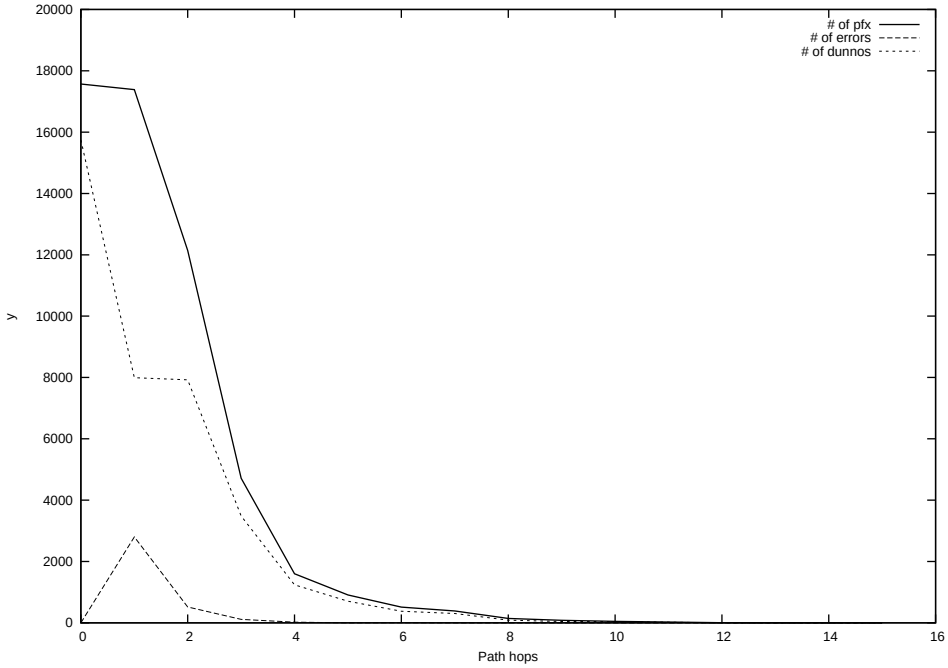
2014-05-06



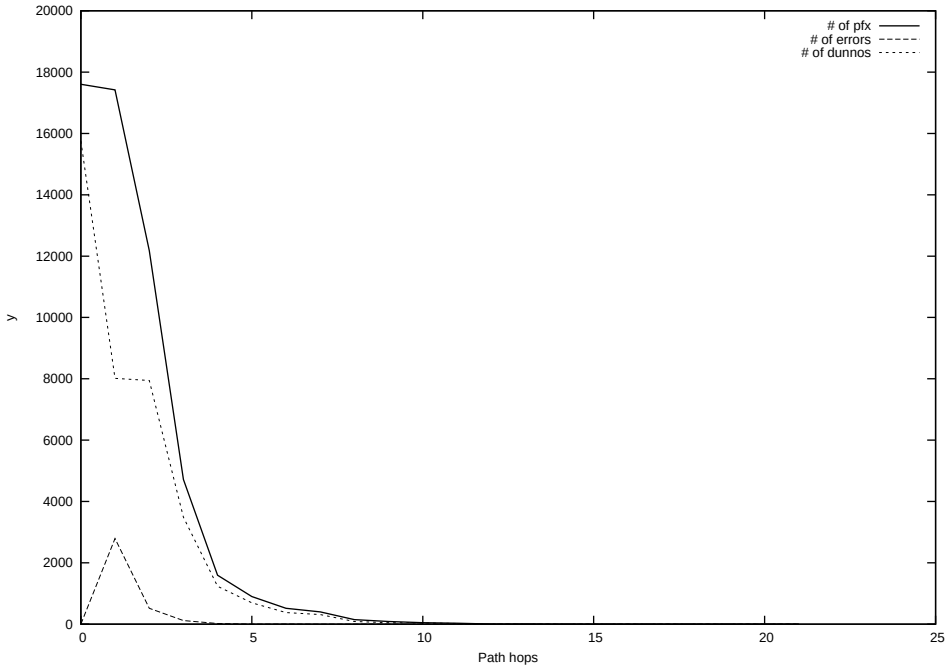
2014-05-07



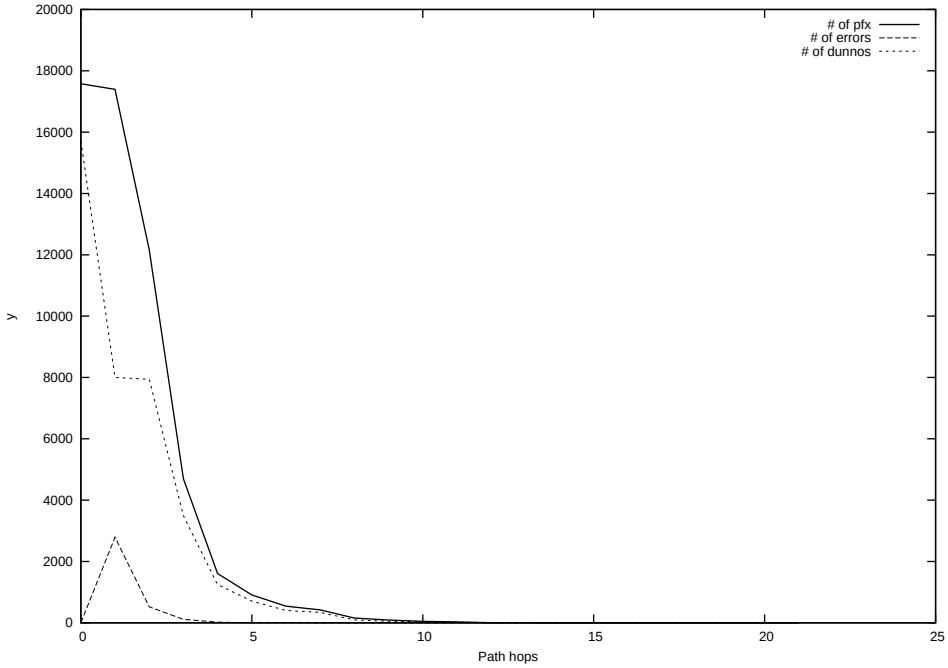
2014-05-08



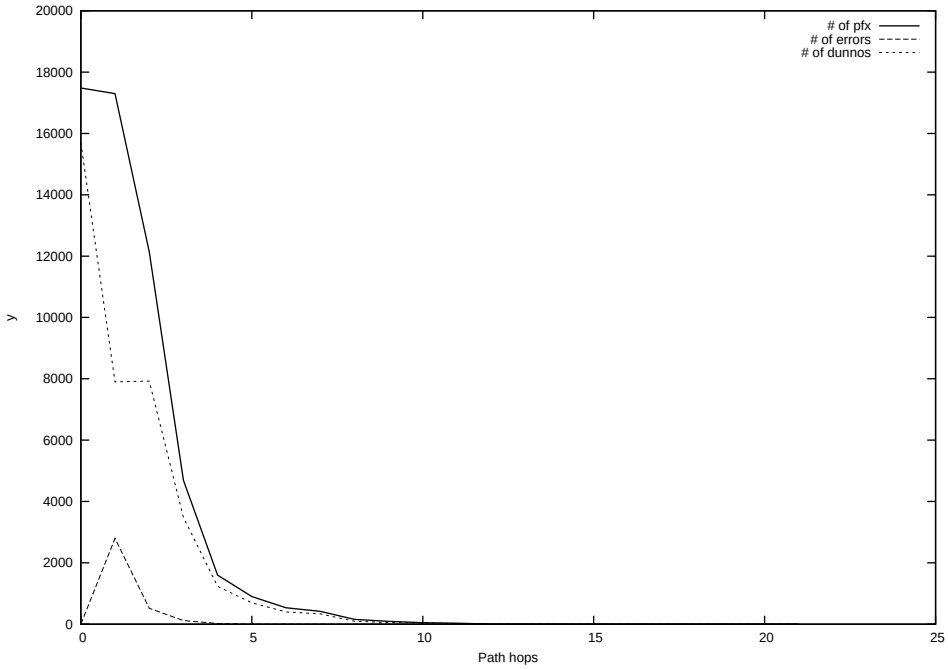
2014-05-09



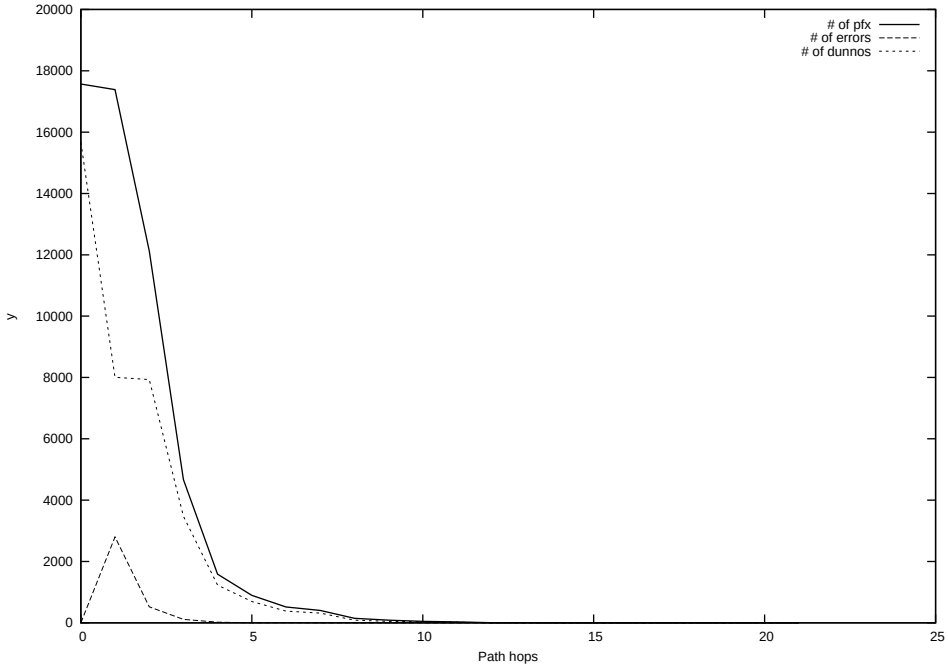
2014-05-10



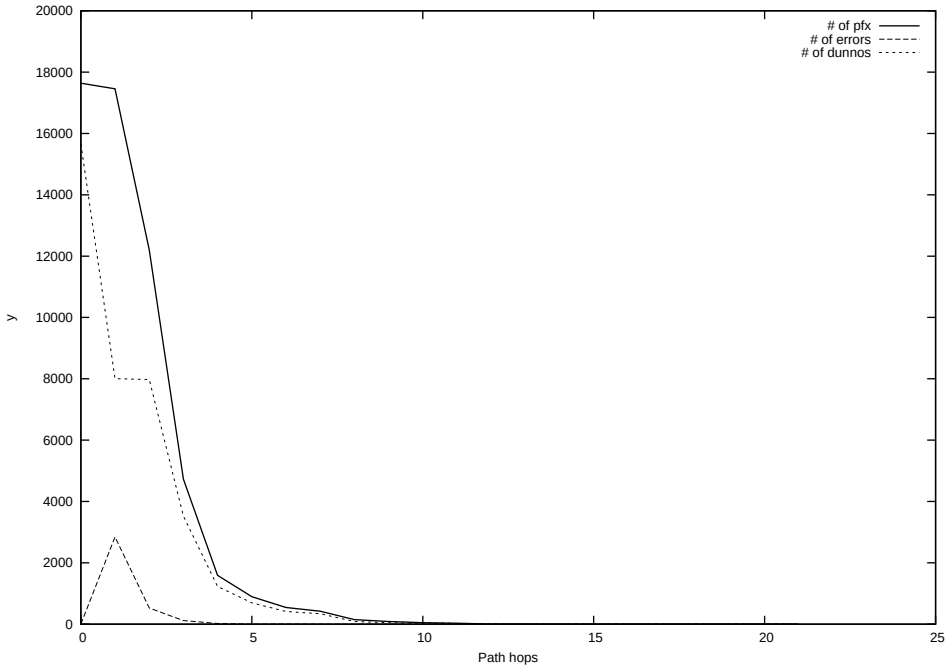
2014-05-11



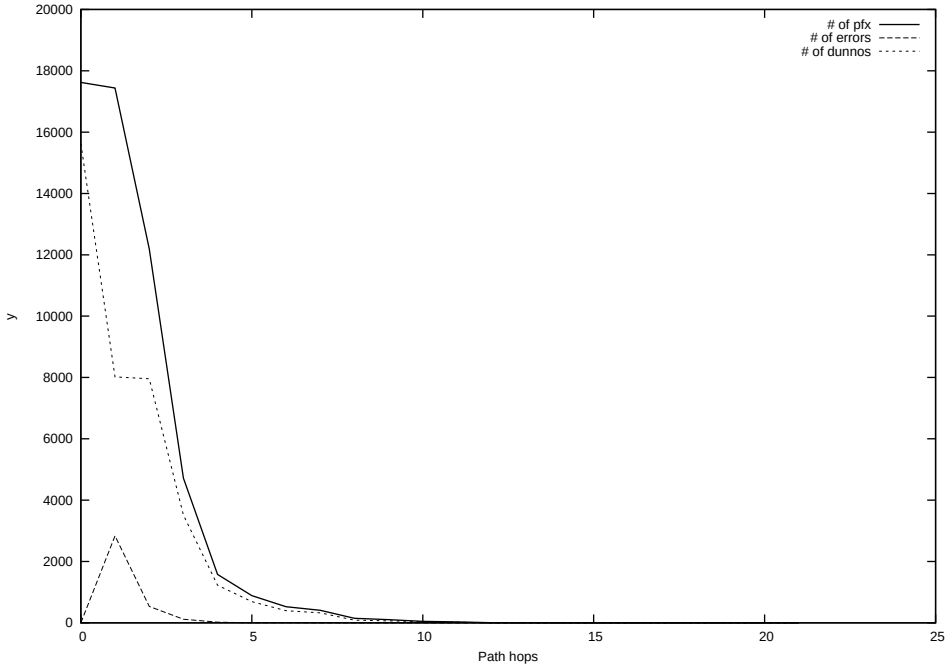
2014-05-12



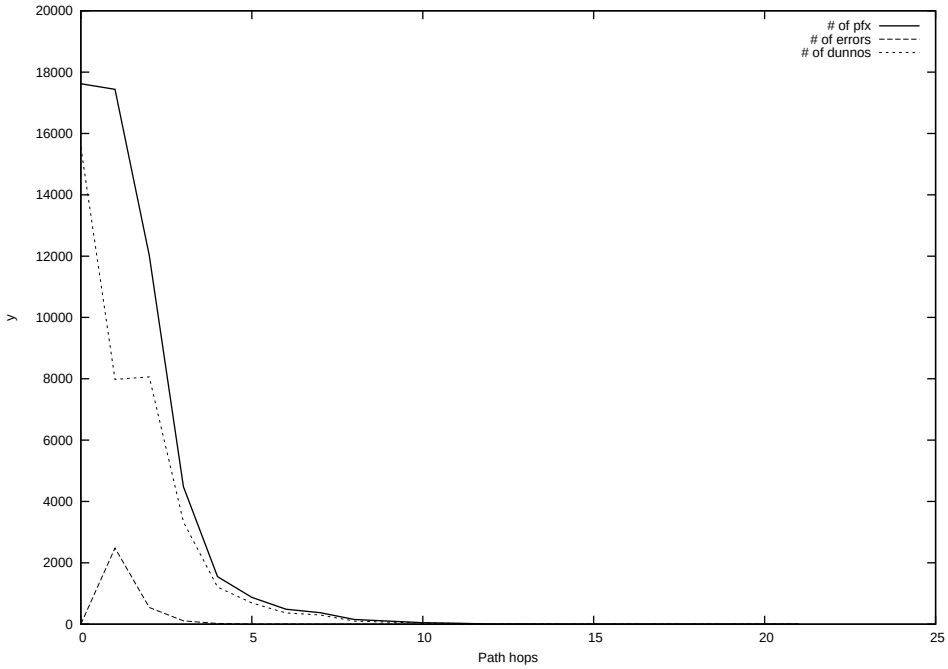
2014-05-13



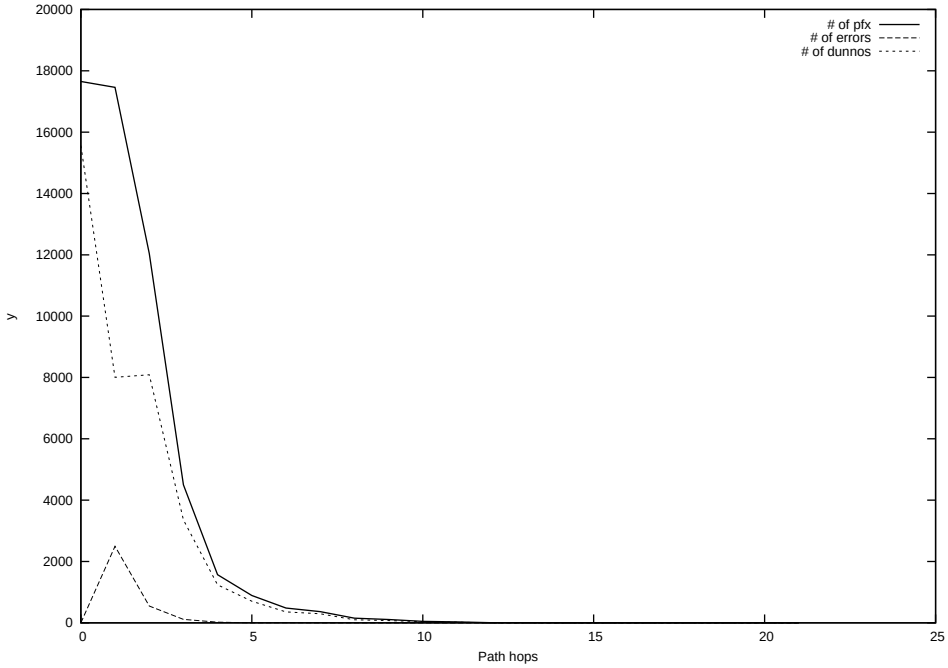
2014-05-14



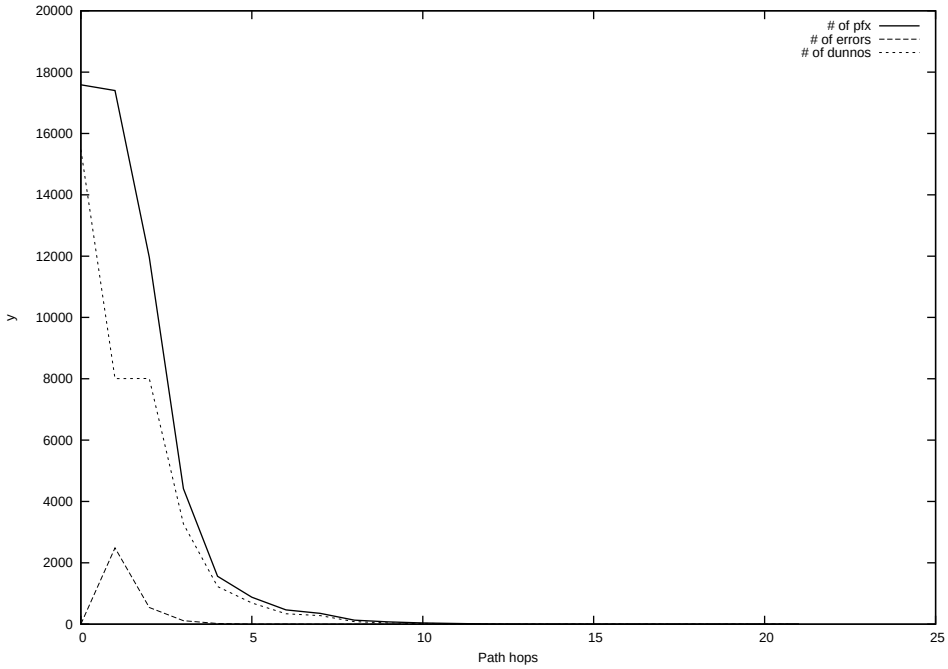
2014-05-15



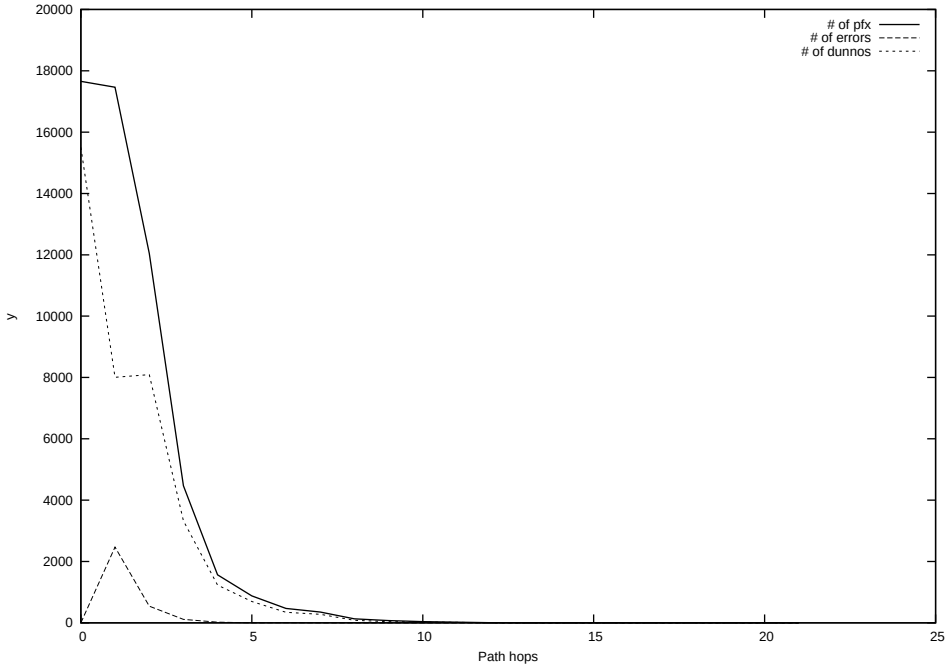
2014-05-16



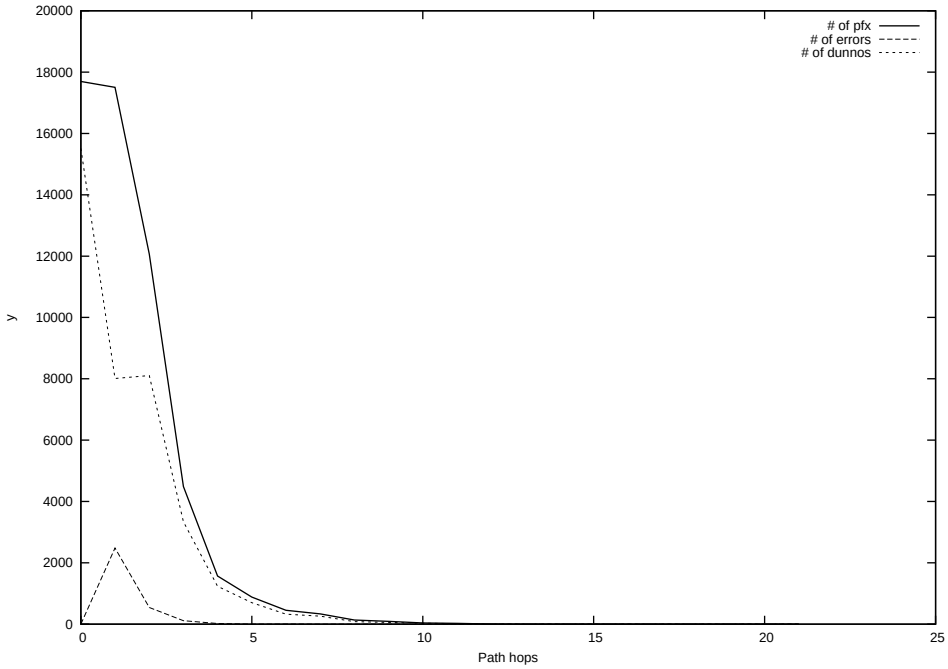
2014-05-17



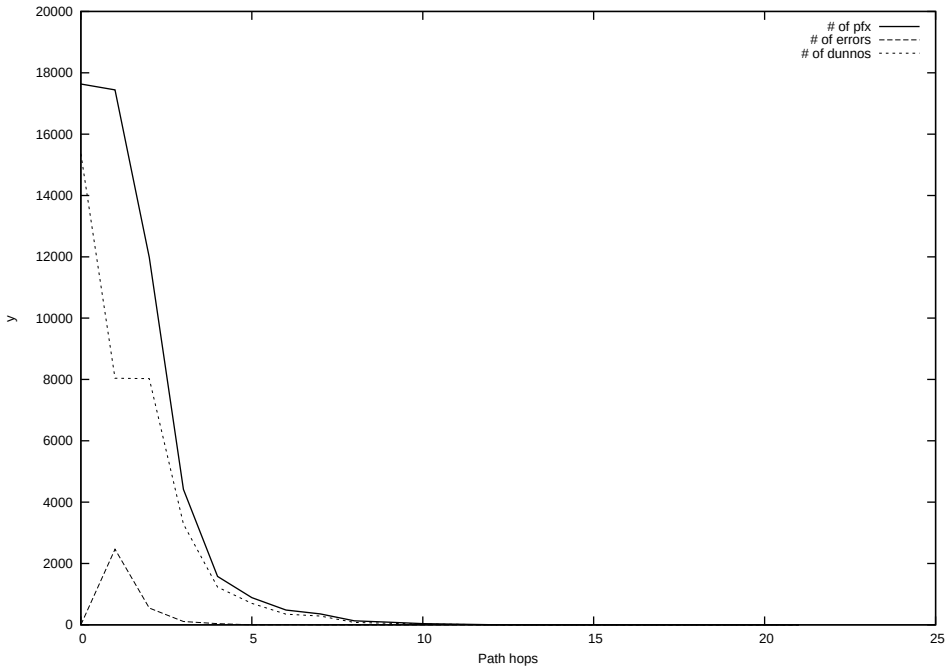
2014-05-18



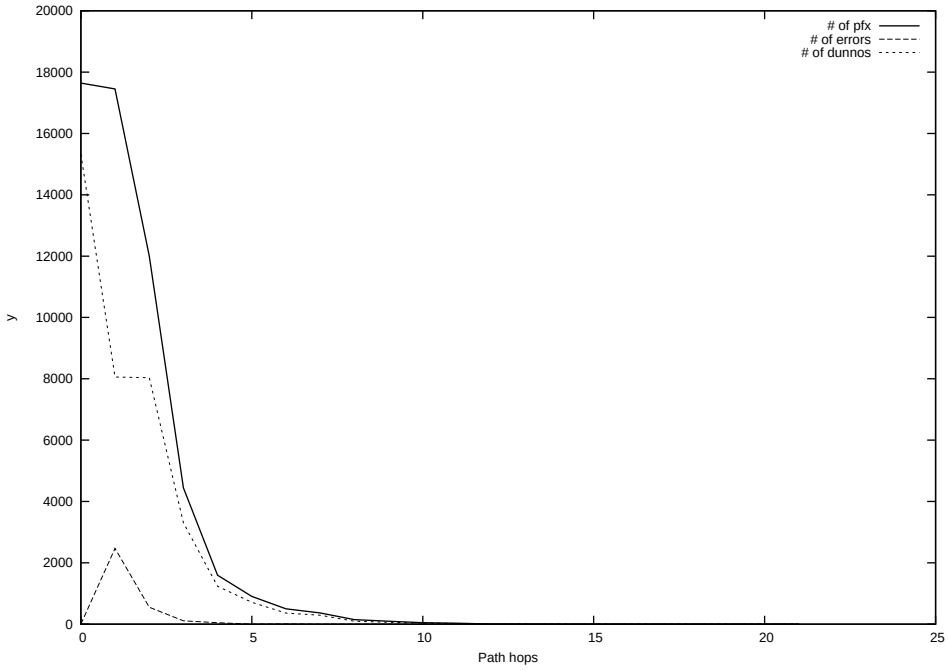
2014-05-19



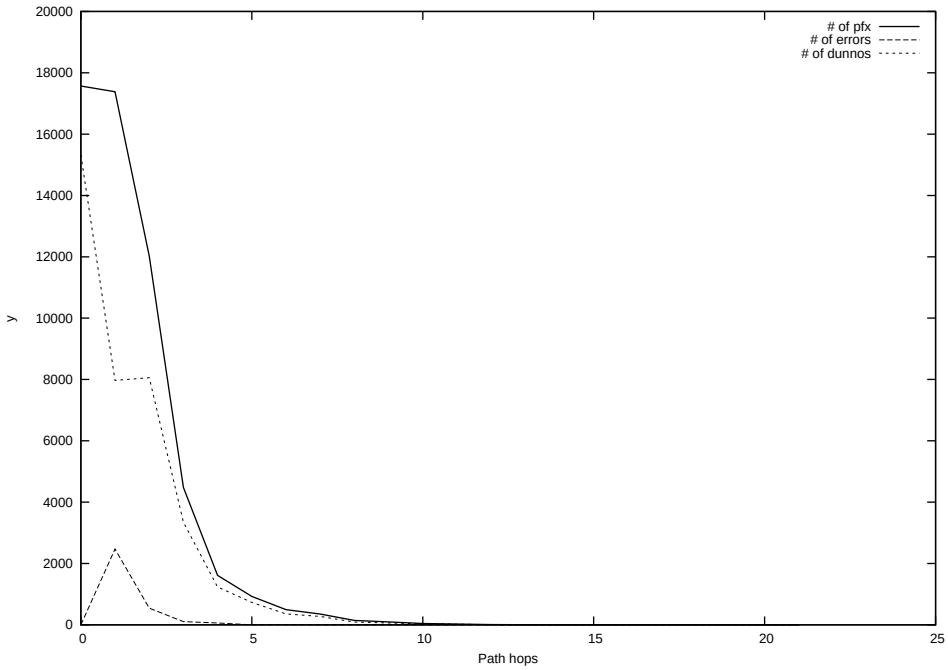
2014-05-20



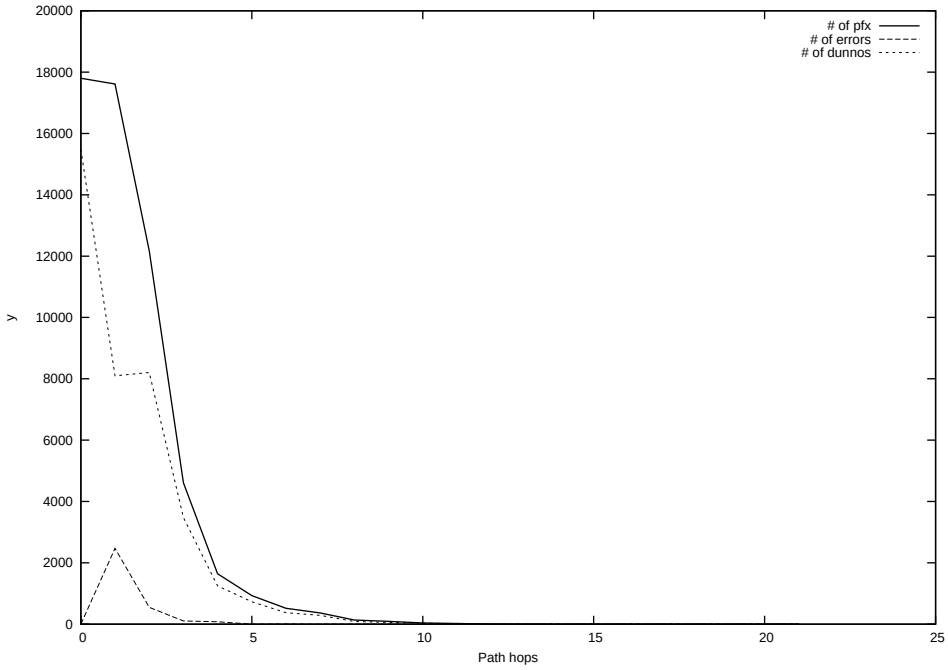
2014-05-21



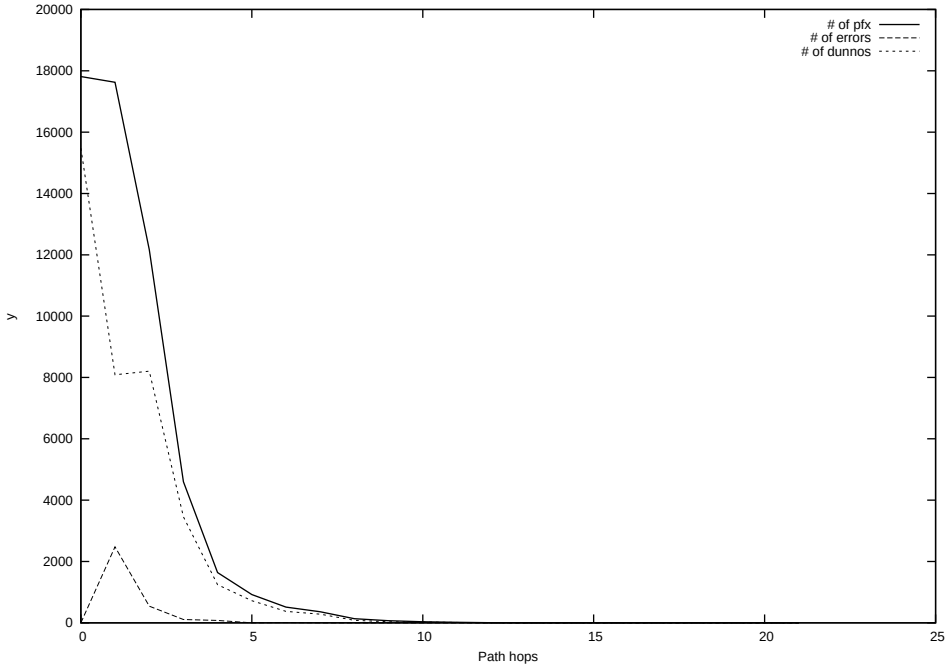
2014-05-22



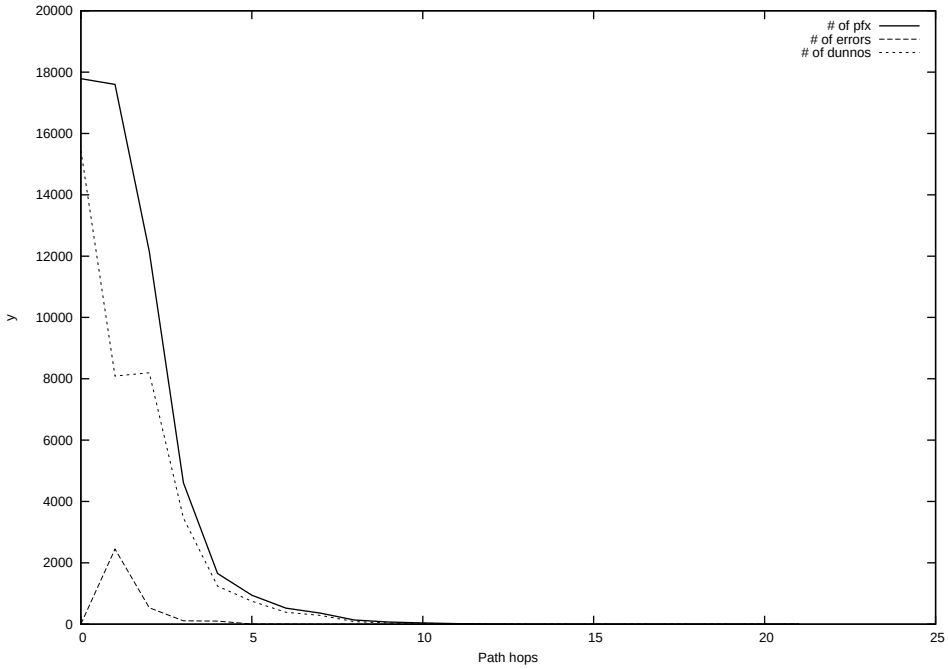
2014-05-23



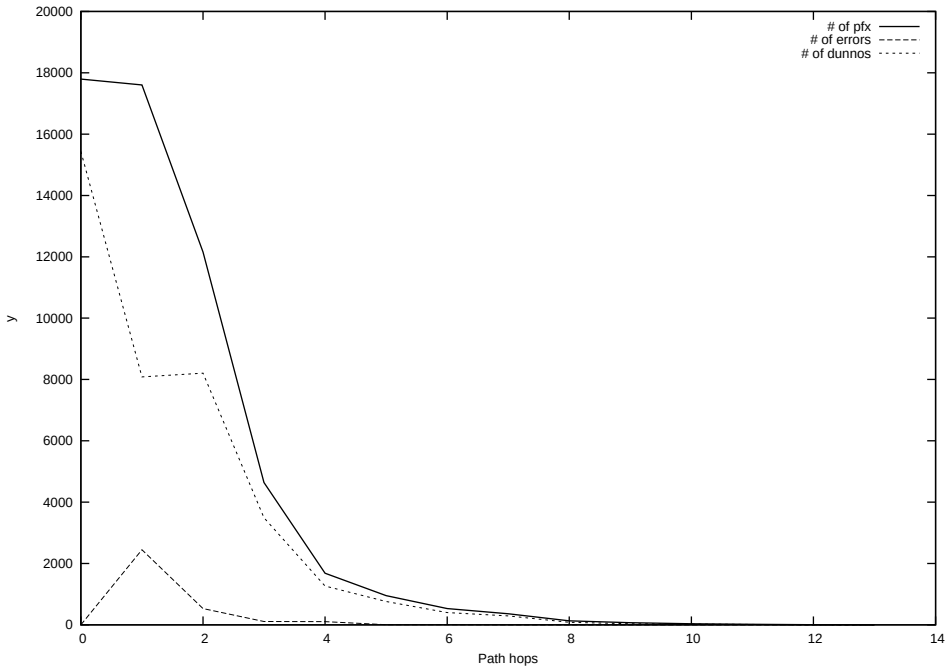
2014-05-24



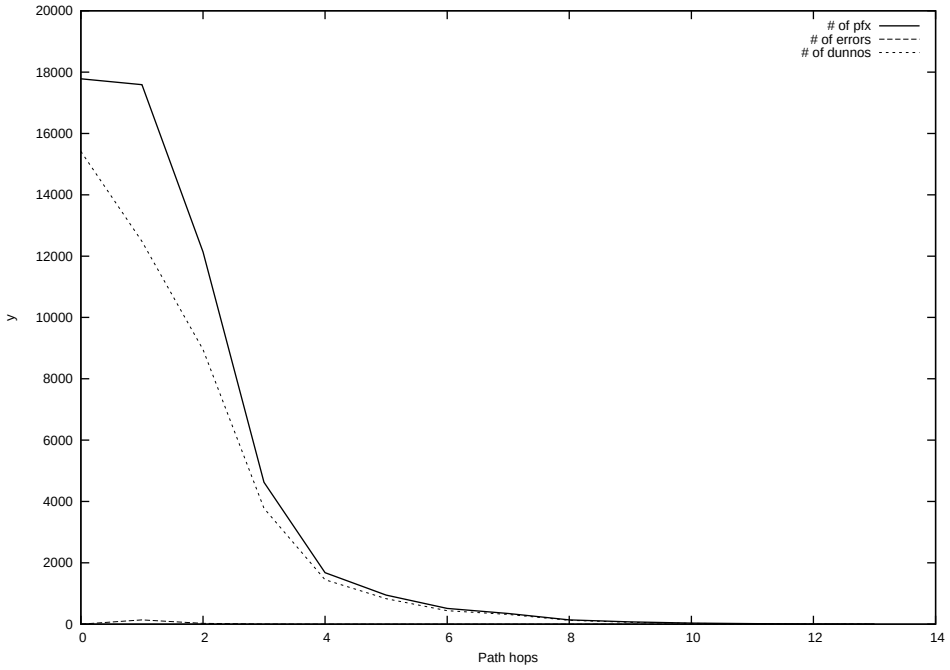
2014-05-25



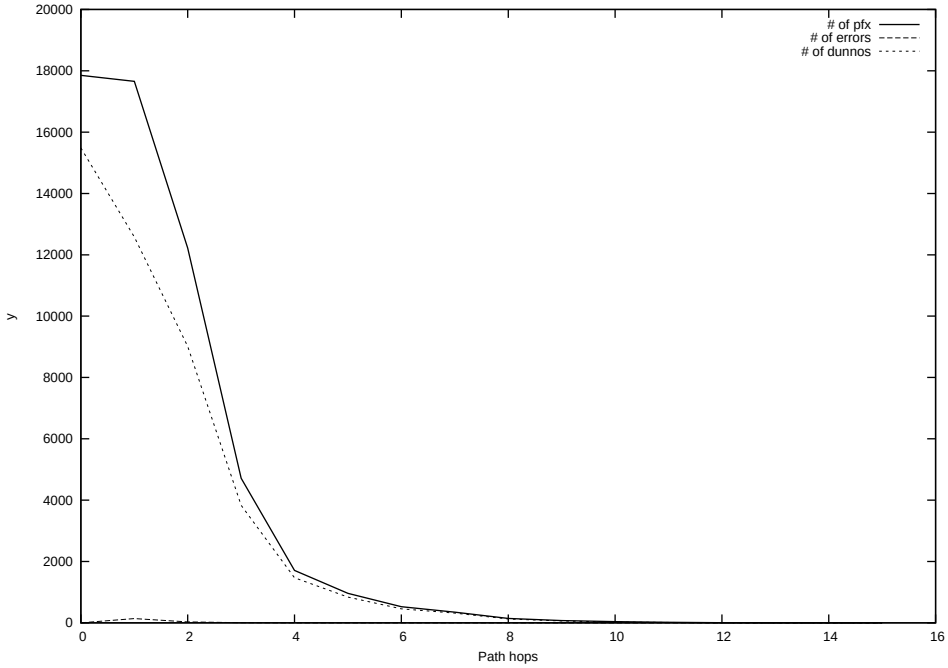
2014-05-26



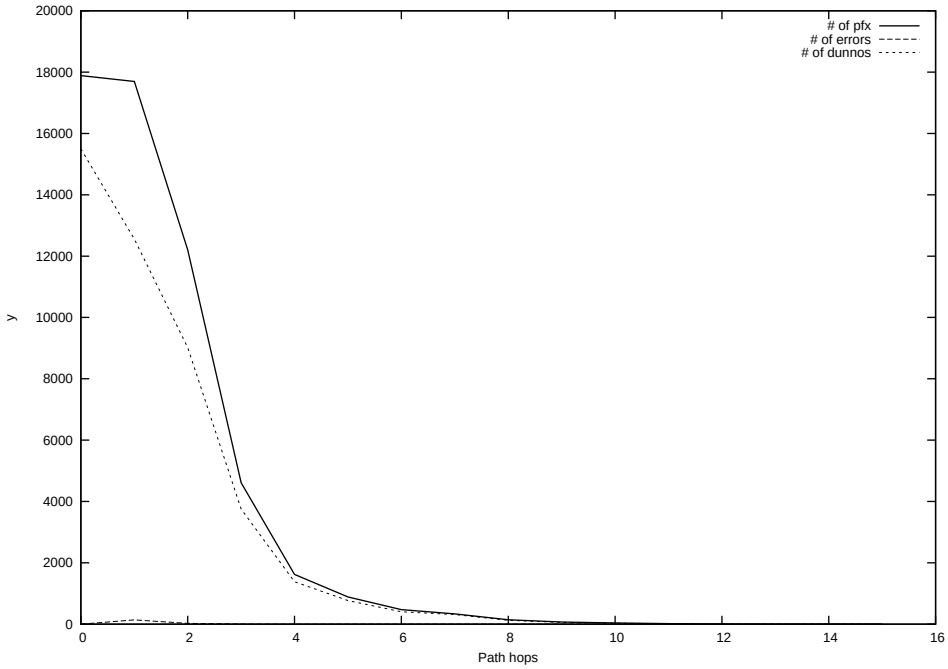
2014-05-27



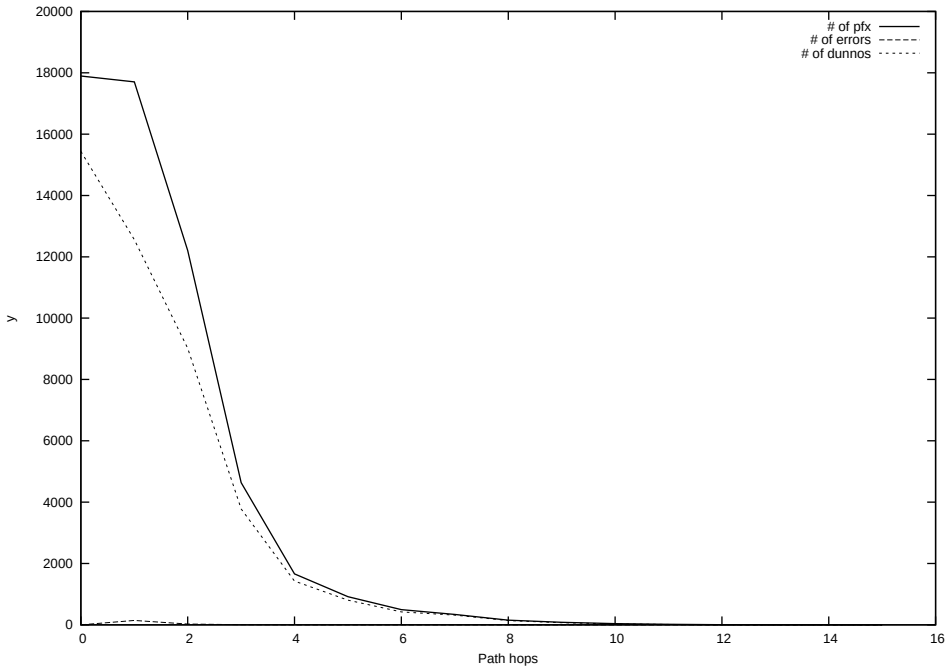
2014-05-28



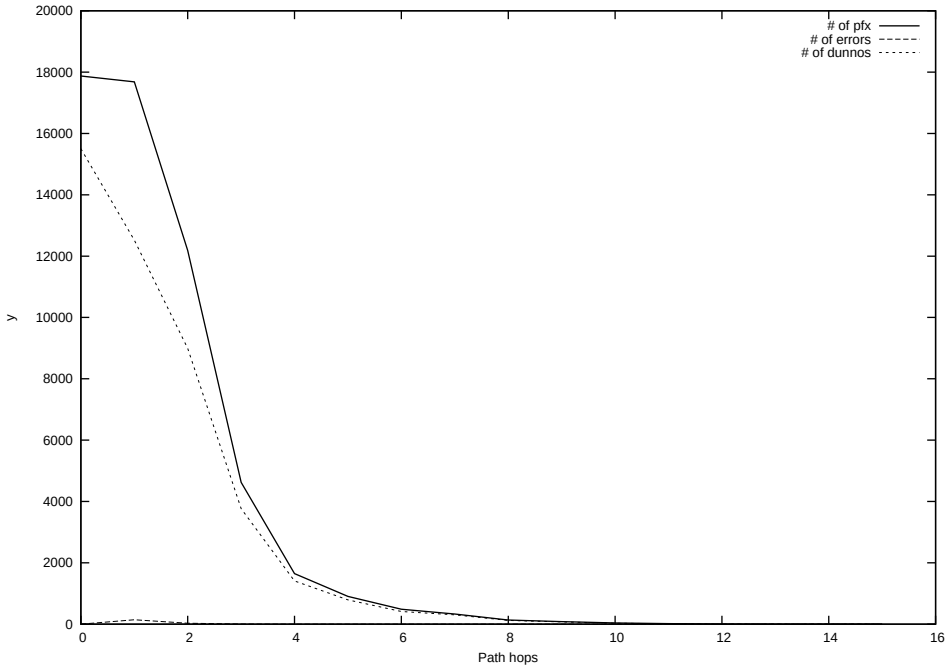
2014-05-29



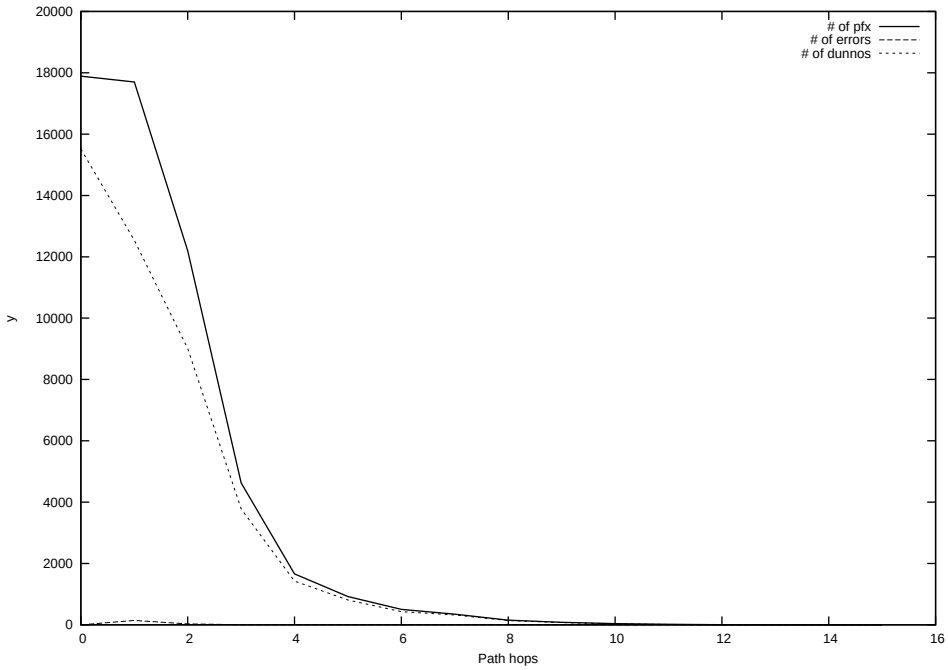
2014-05-30



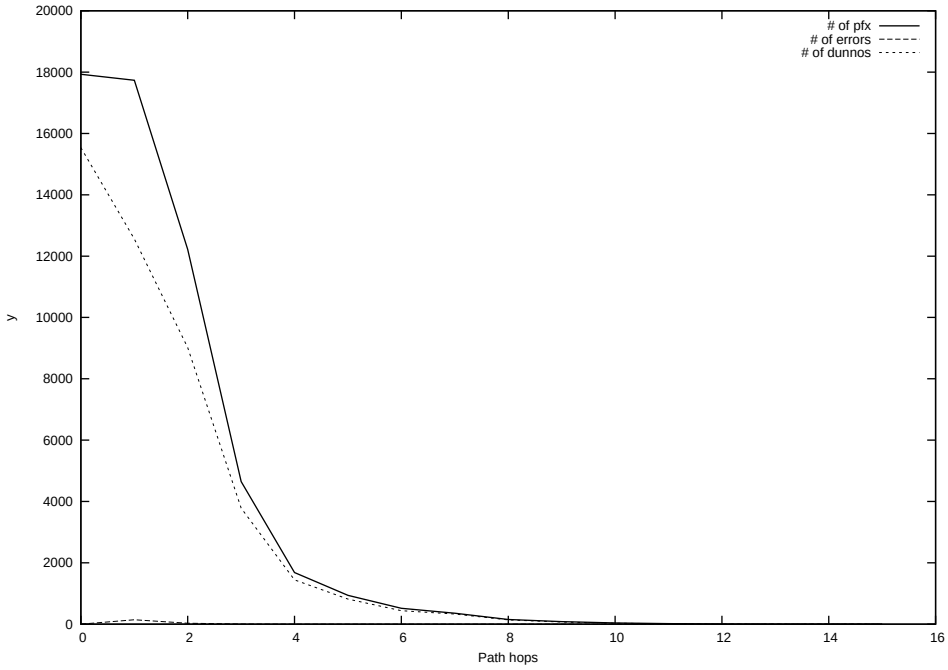
2014-05-31



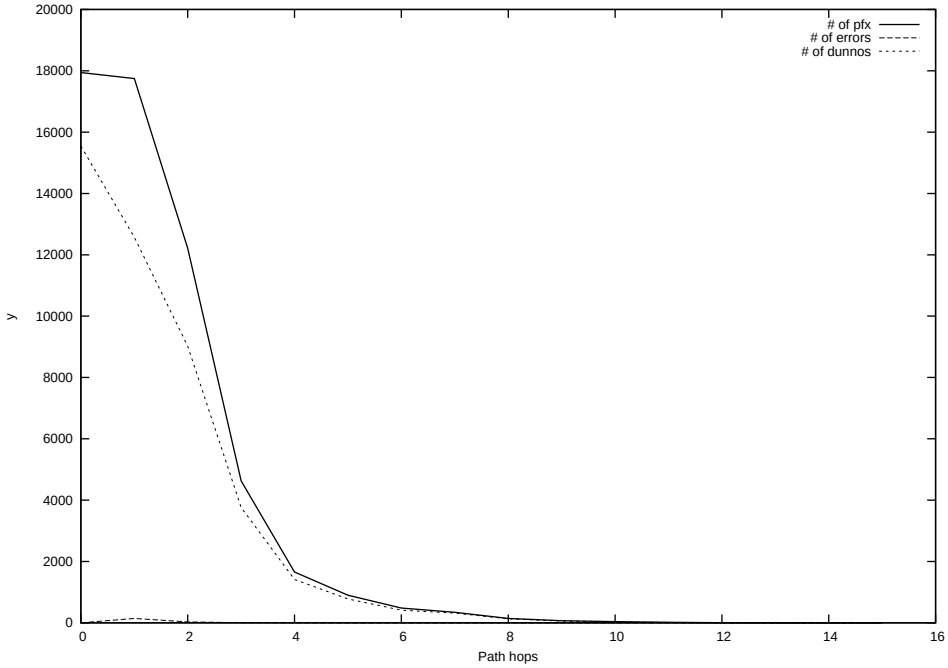
2014-06-01



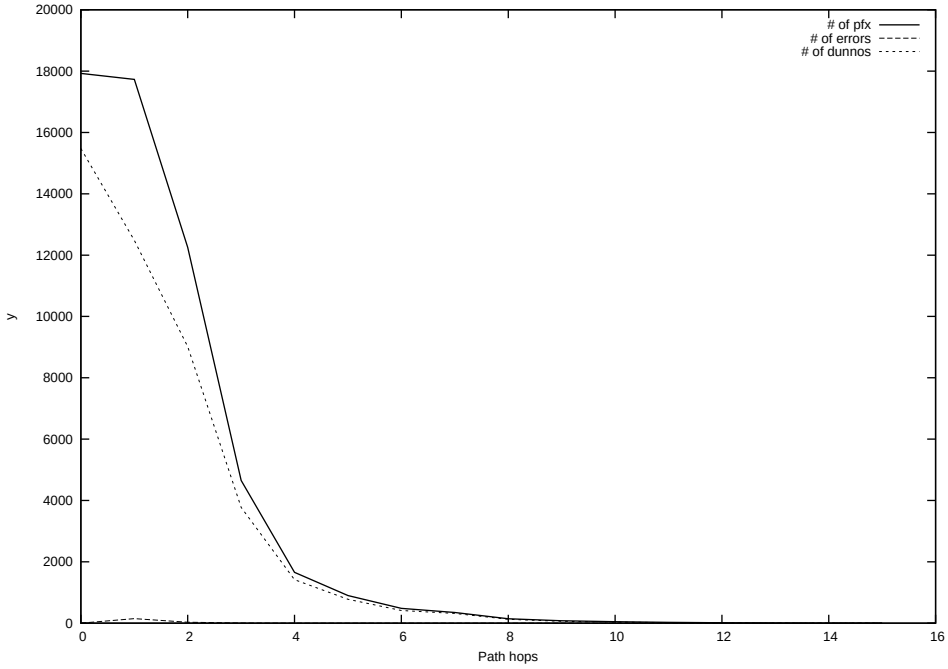
2014-06-02



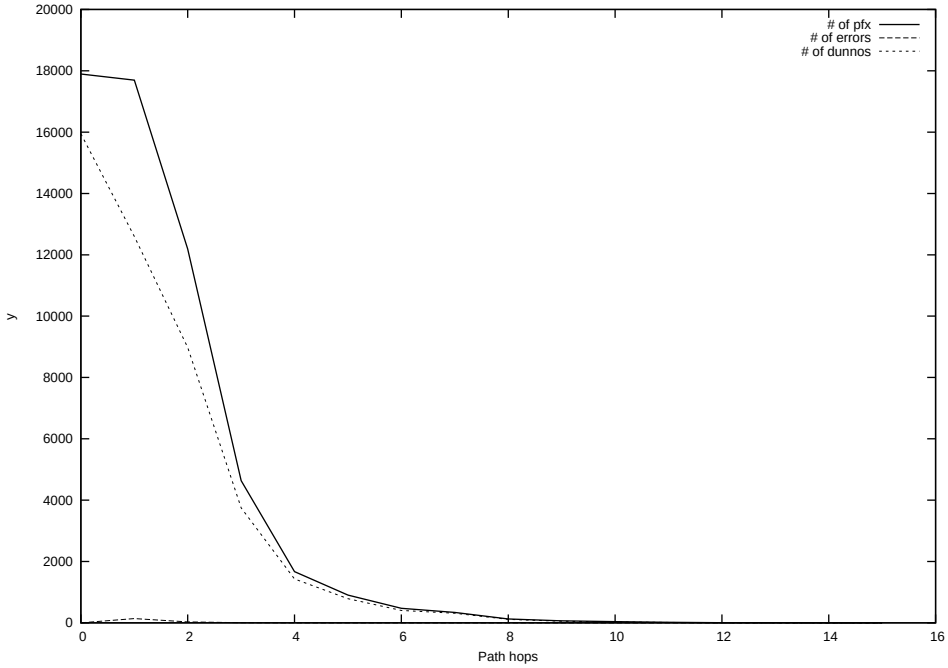
2014-06-03



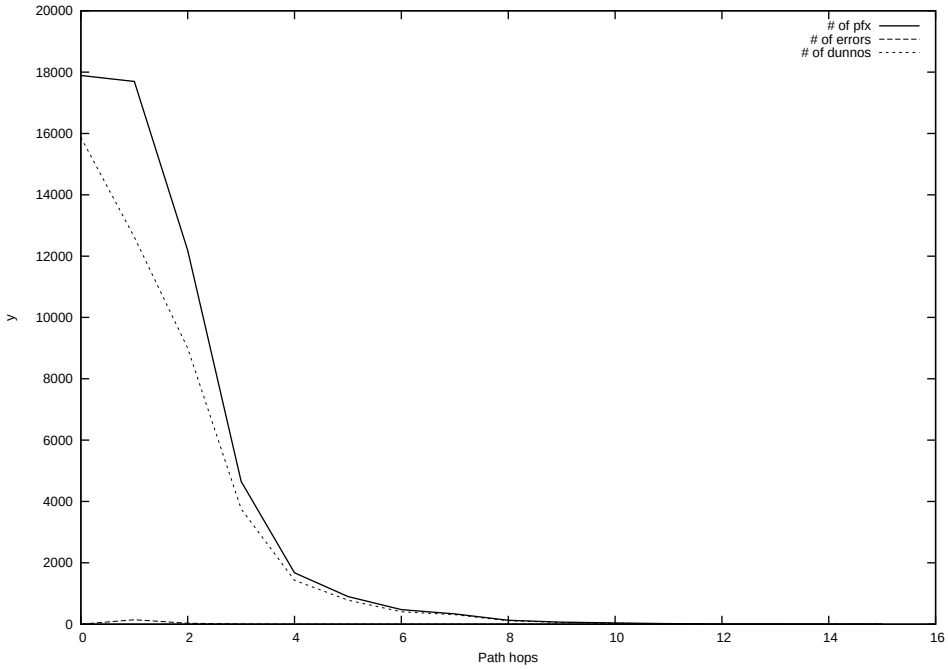
2014-06-04



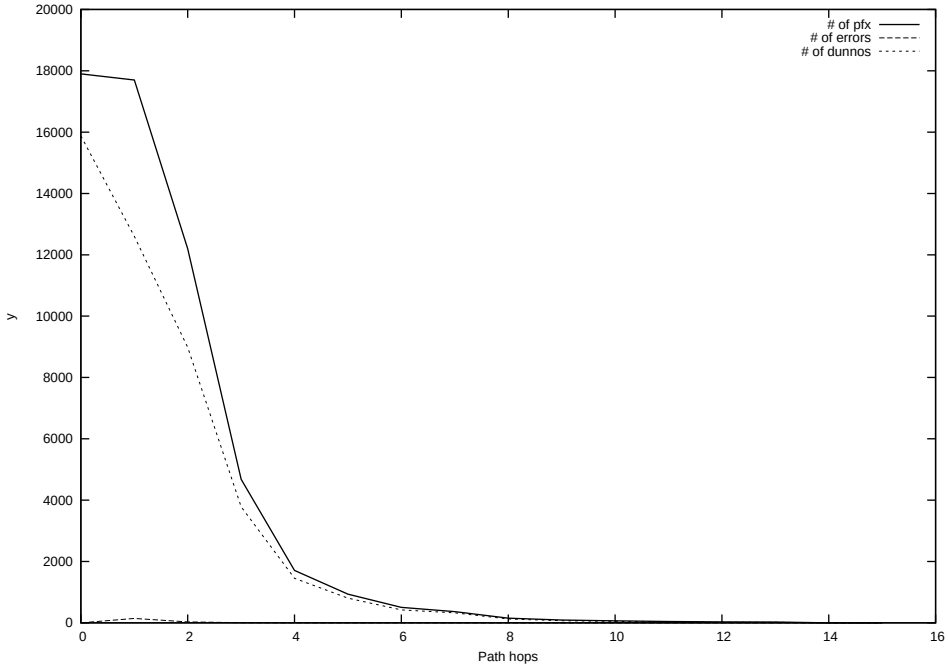
2014-06-05



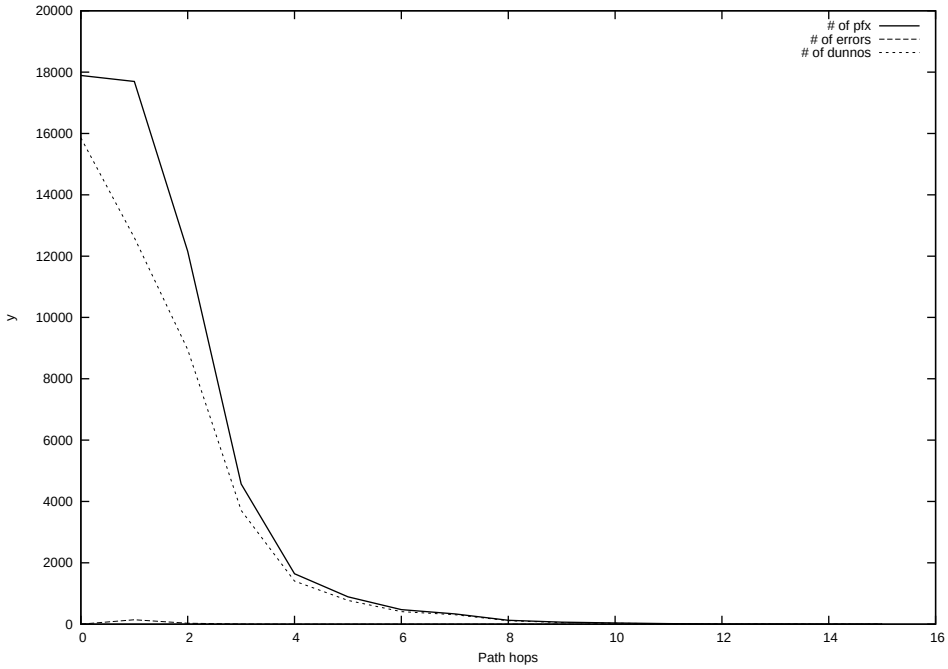
2014-06-06



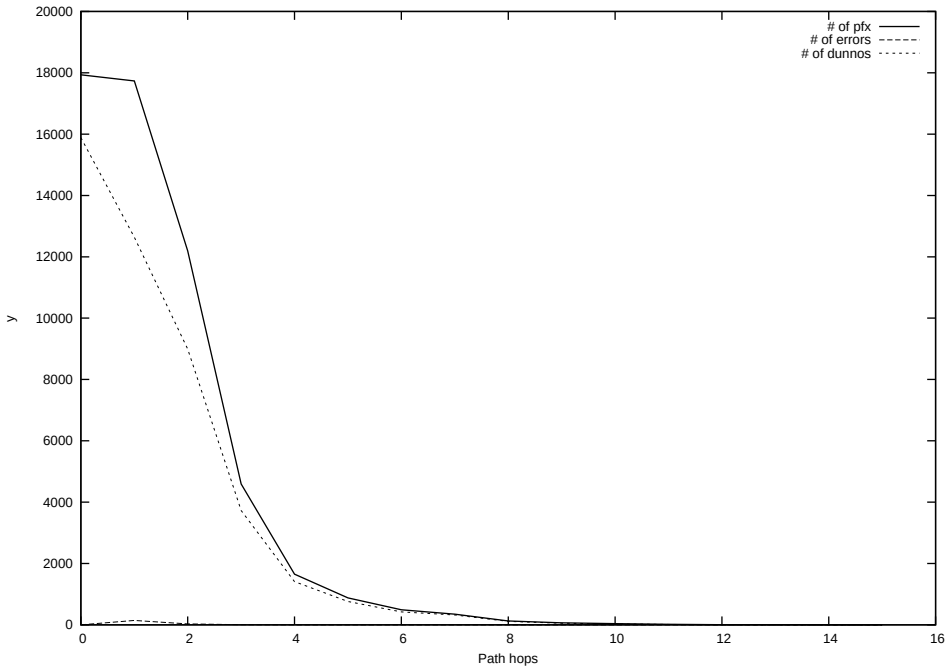
2014-06-07



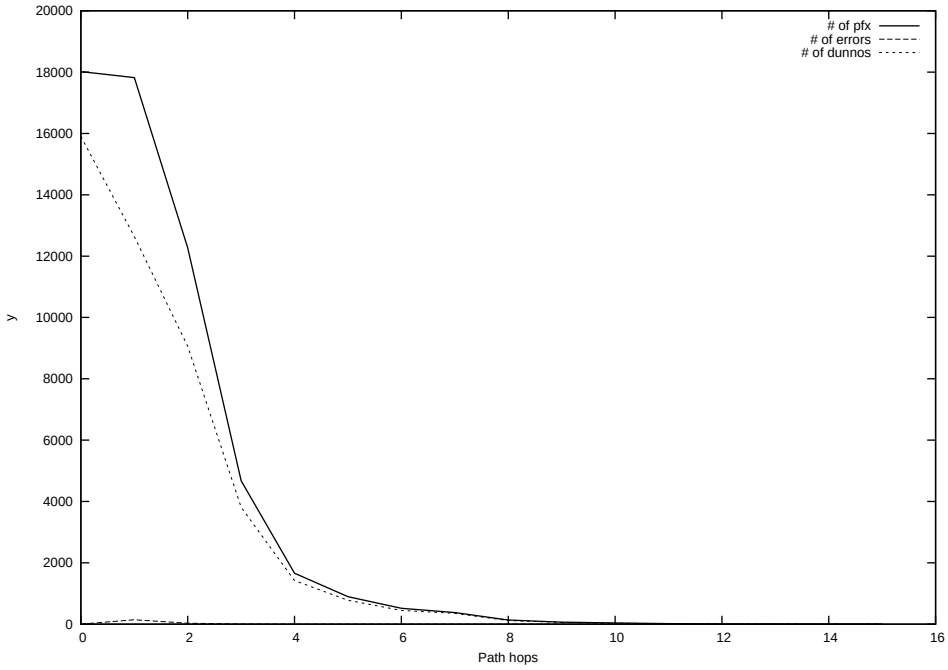
2014-06-08



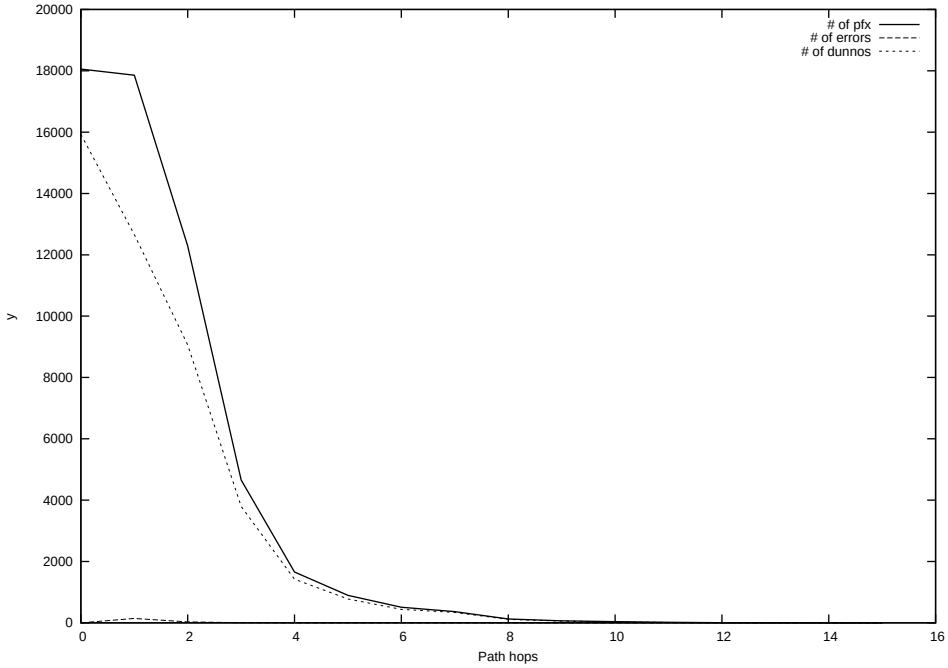
2014-06-09



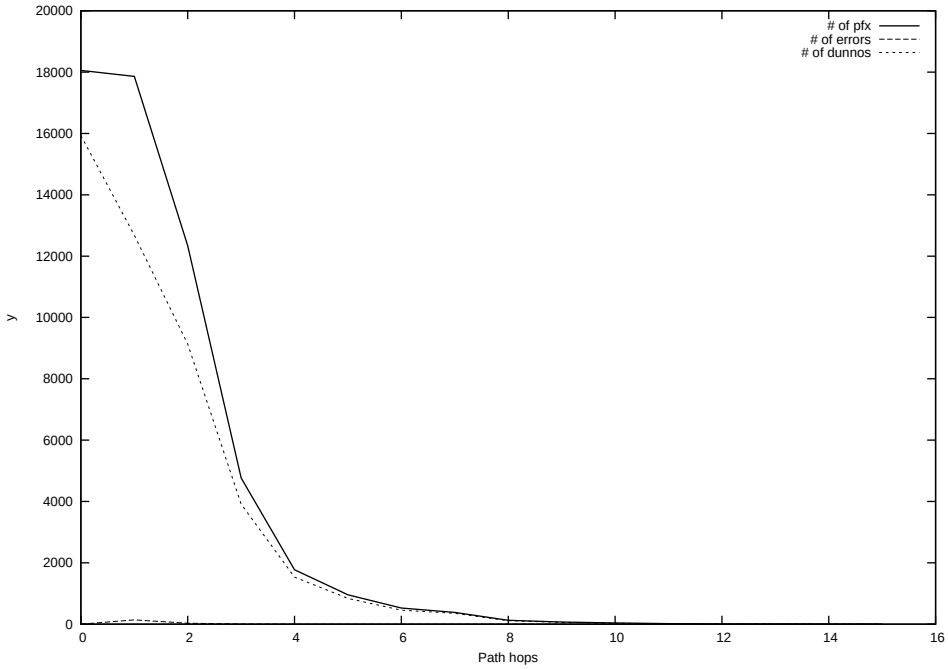
2014-06-10



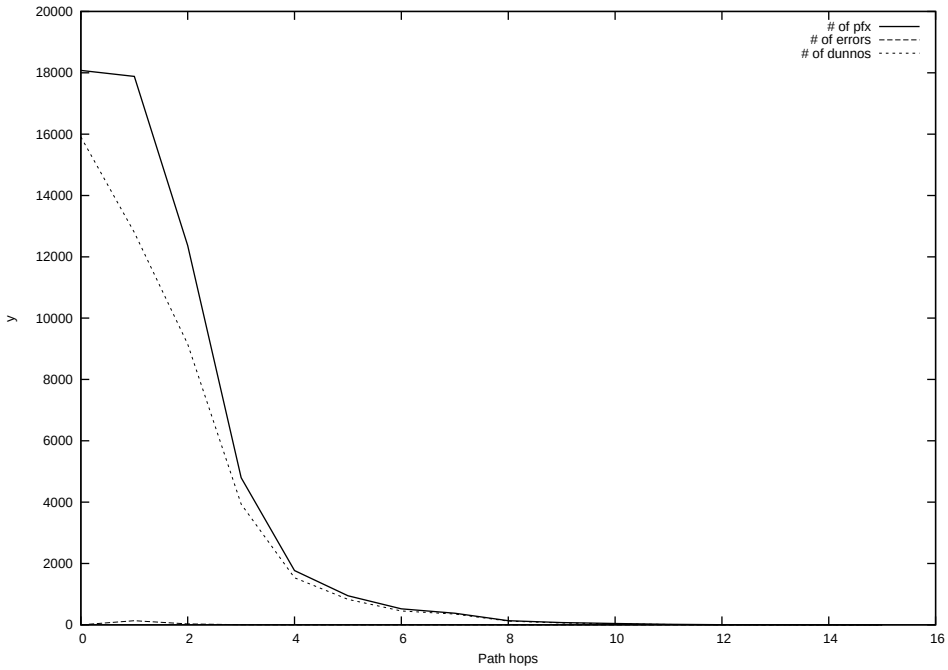
2014-06-11



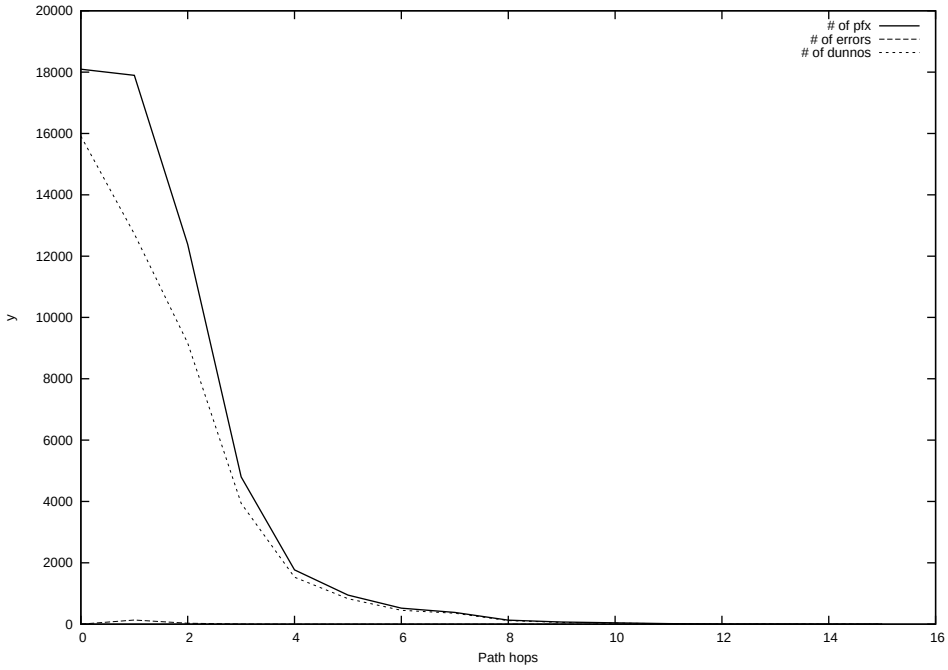
2014-06-12



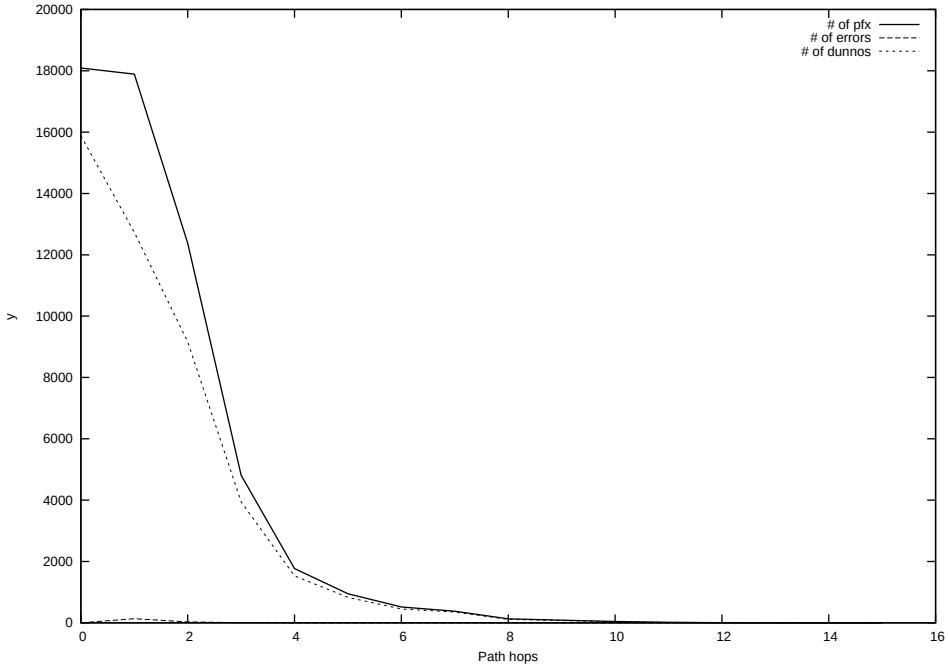
2014-06-13



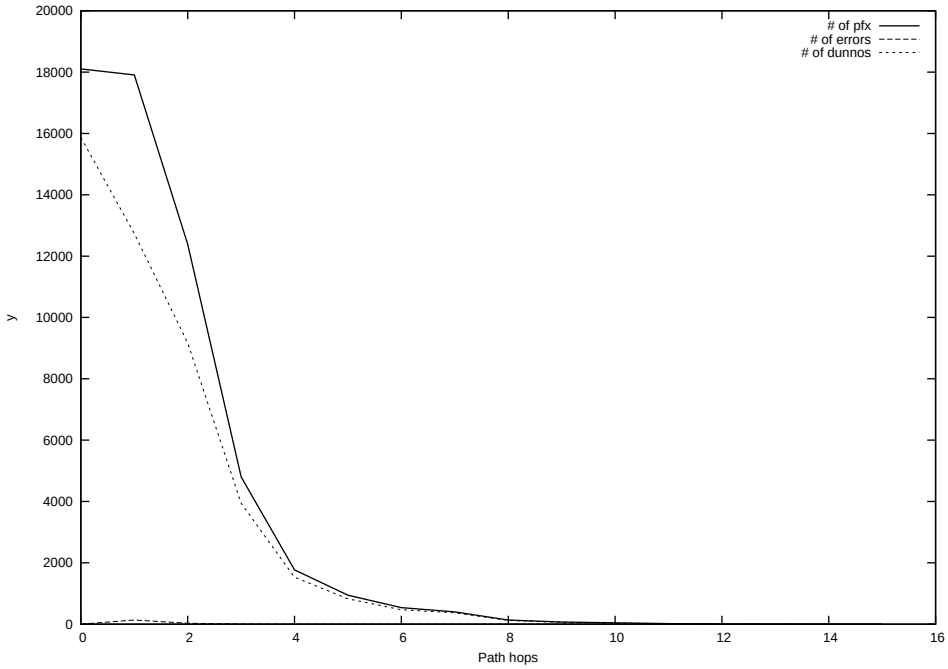
2014-06-14



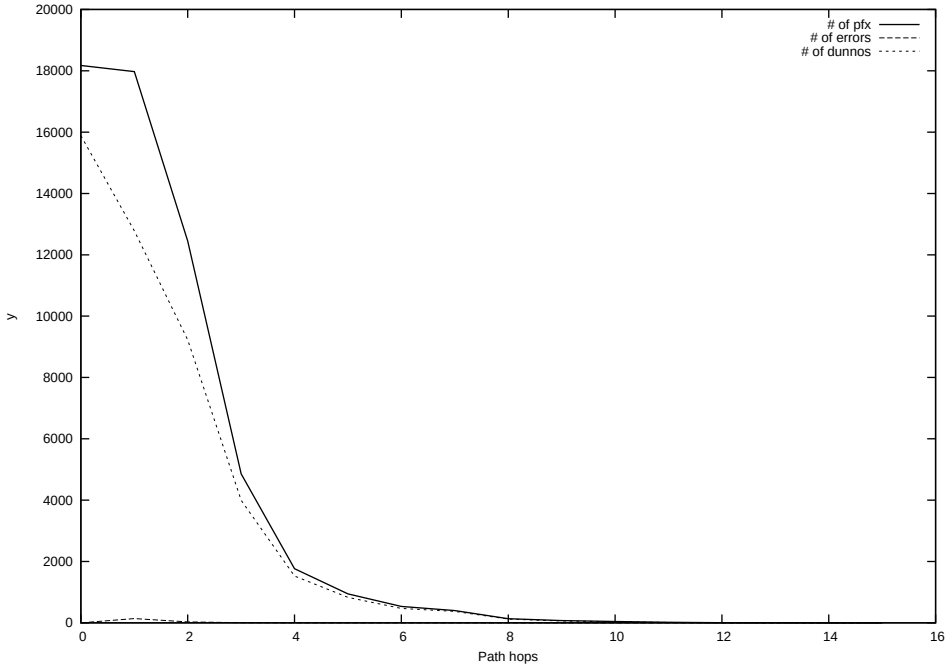
2014-06-15



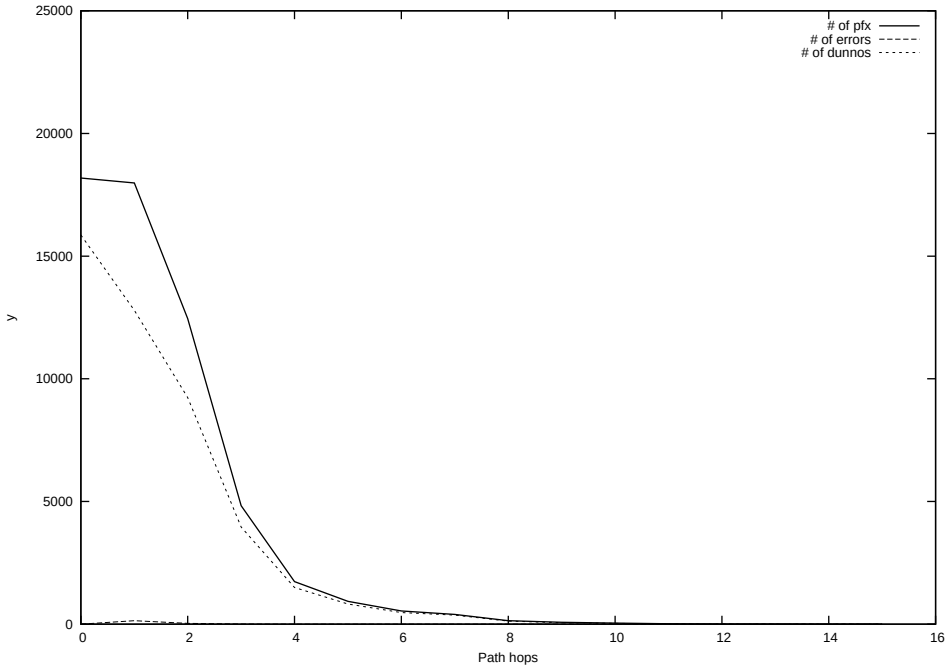
2014-06-16



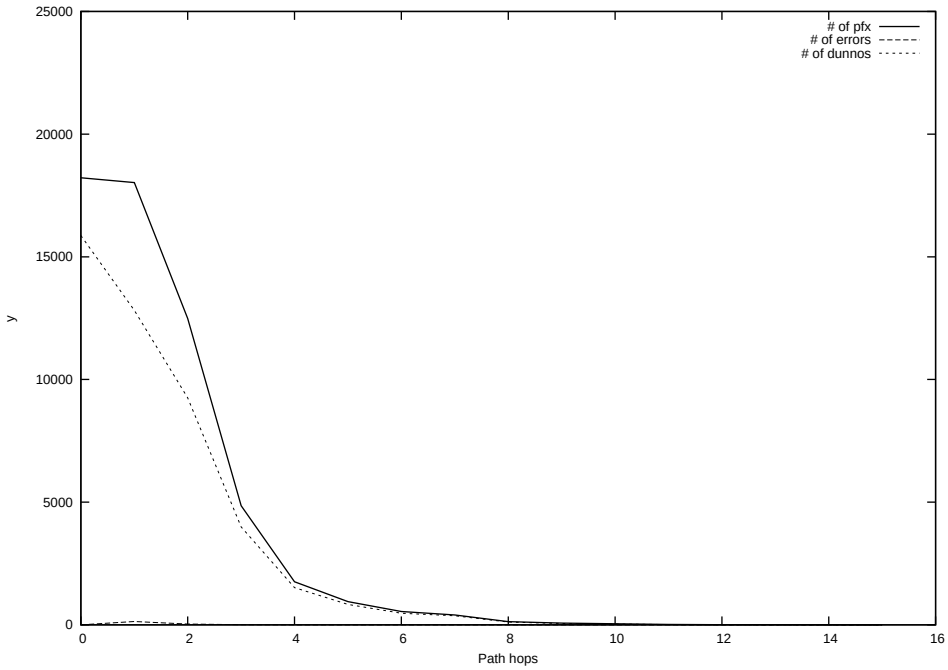
2014-06-17



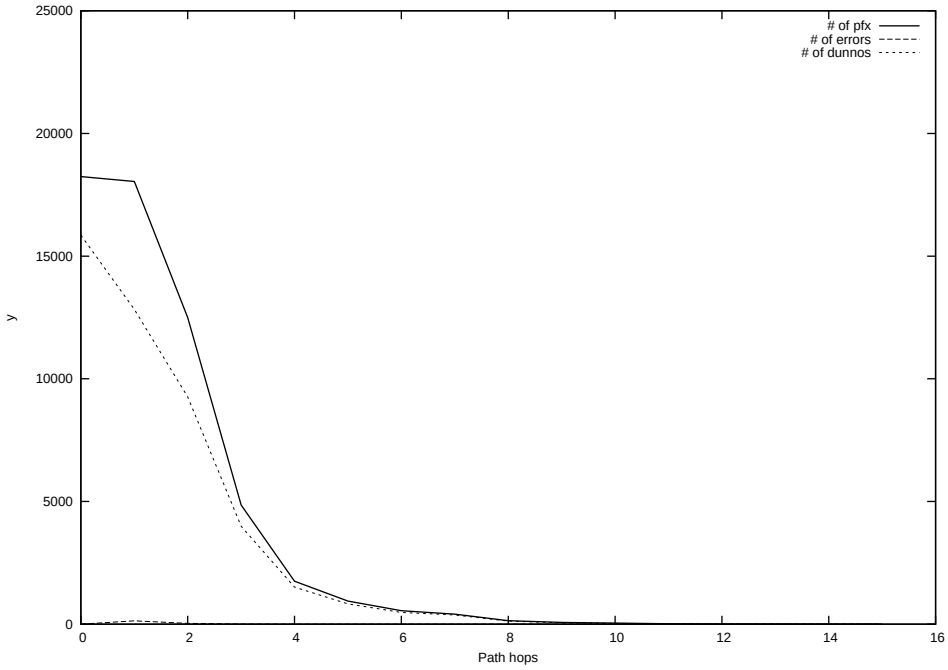
2014-06-18



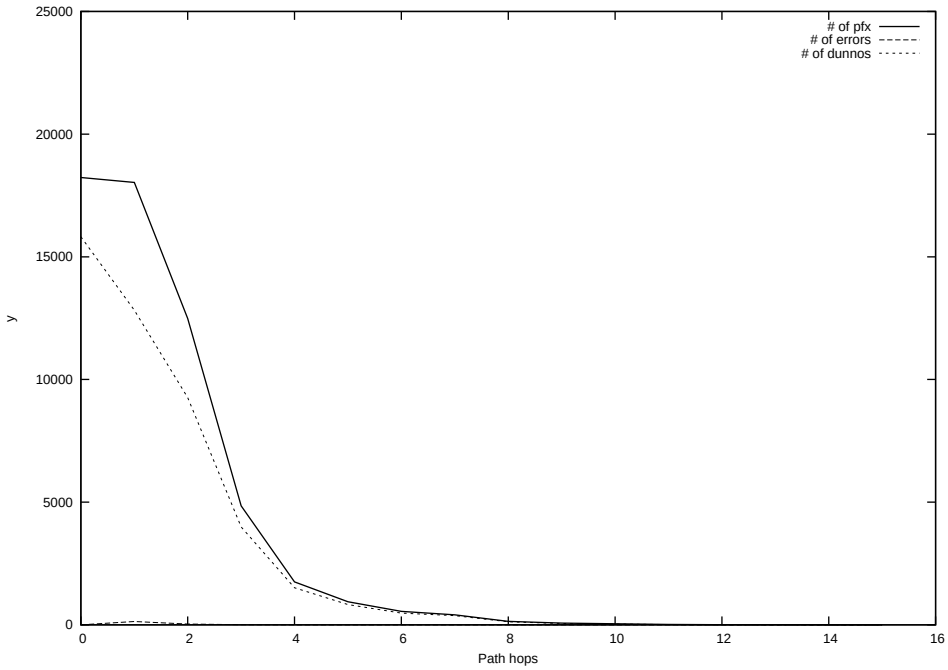
2014-06-19



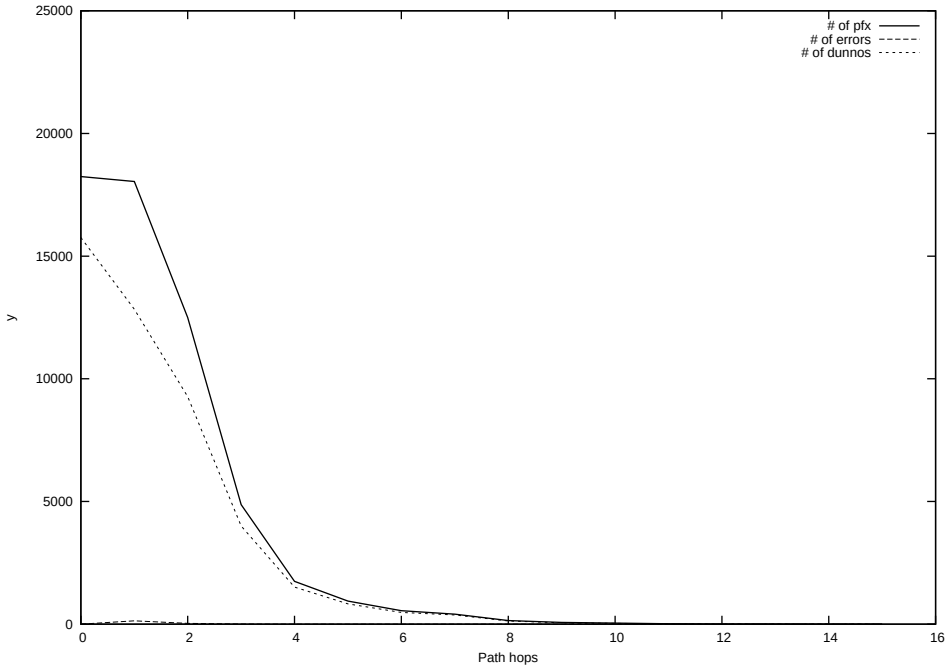
2014-06-20



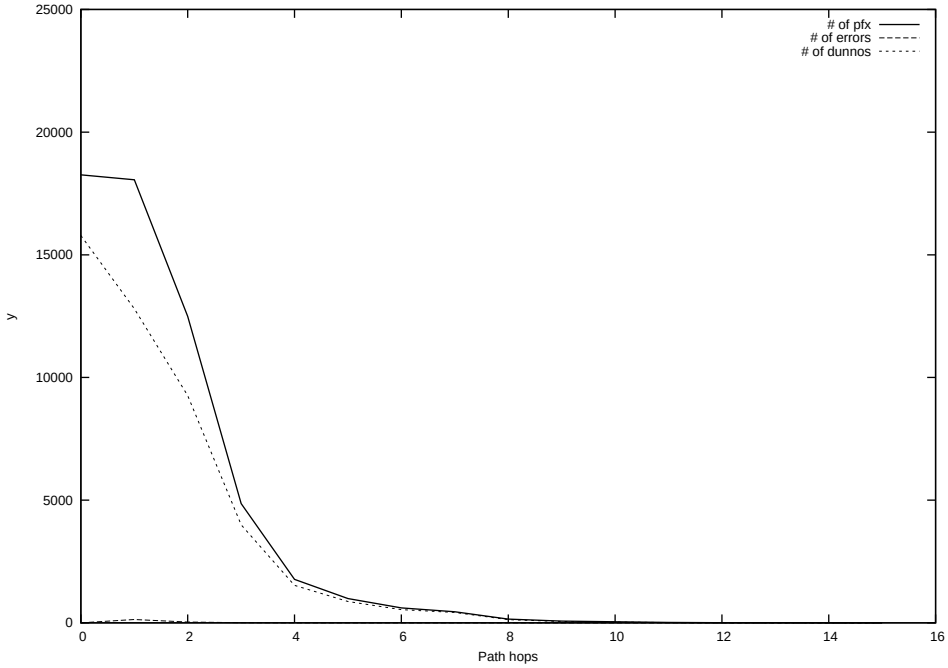
2014-06-21



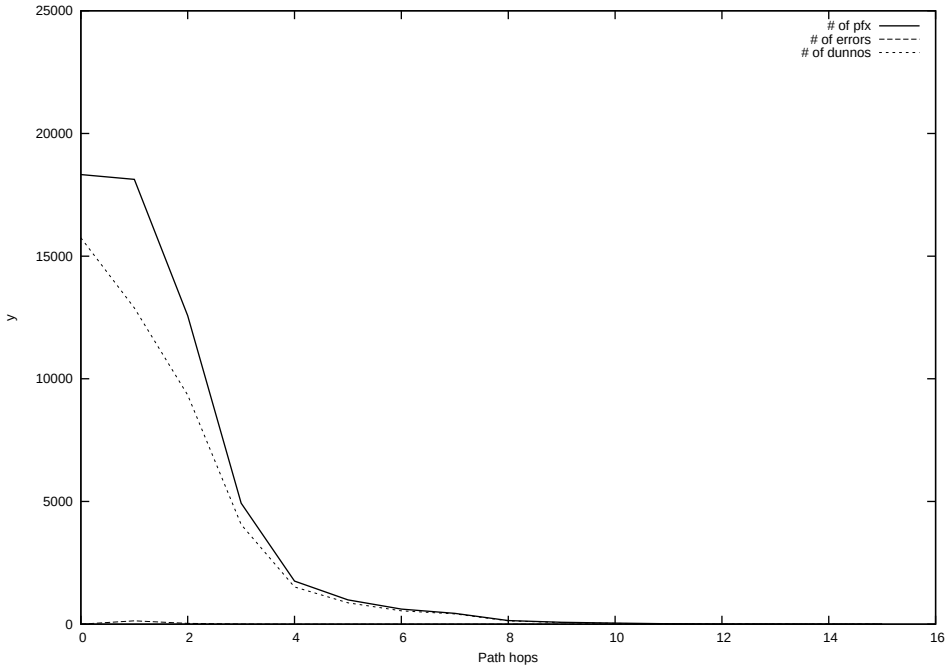
2014-06-22



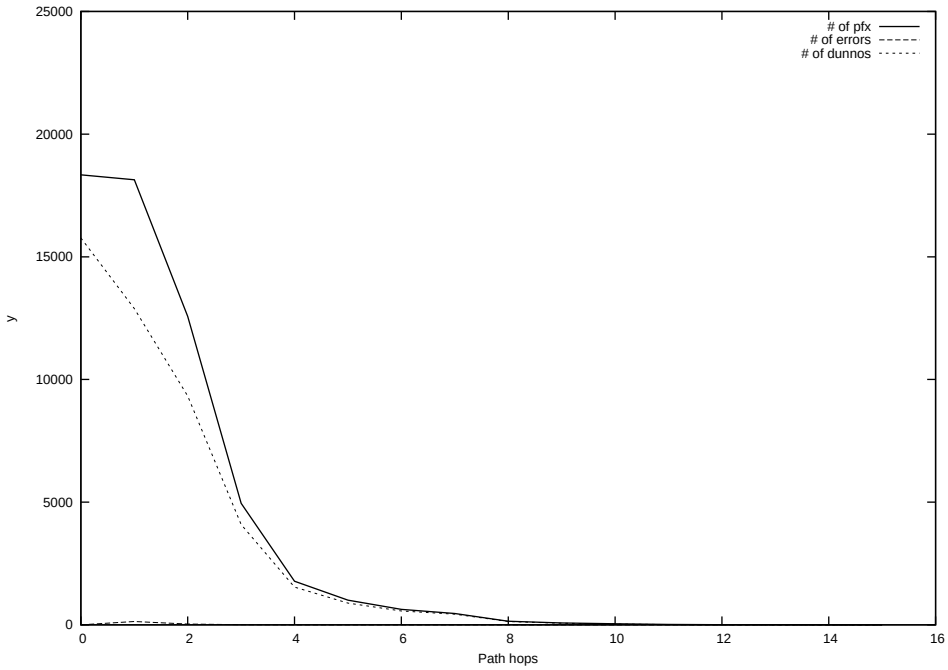
2014-06-23



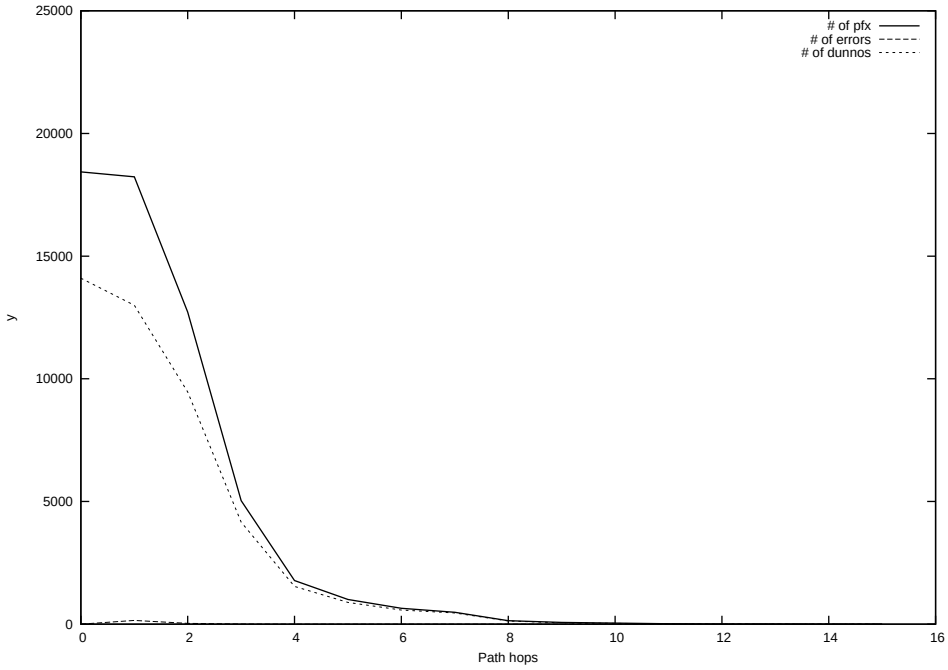
2014-06-24



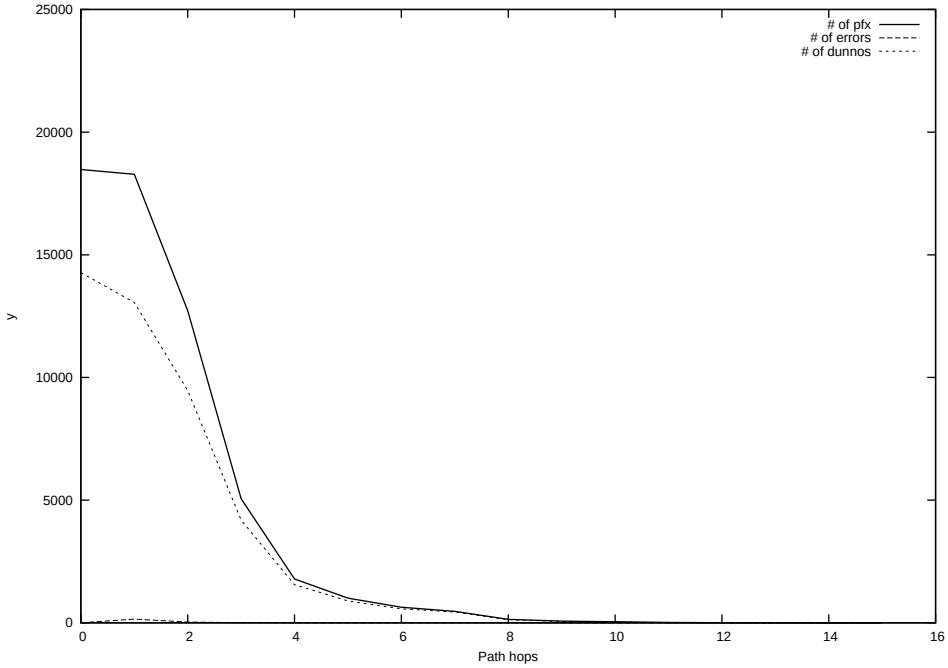
2014-06-25



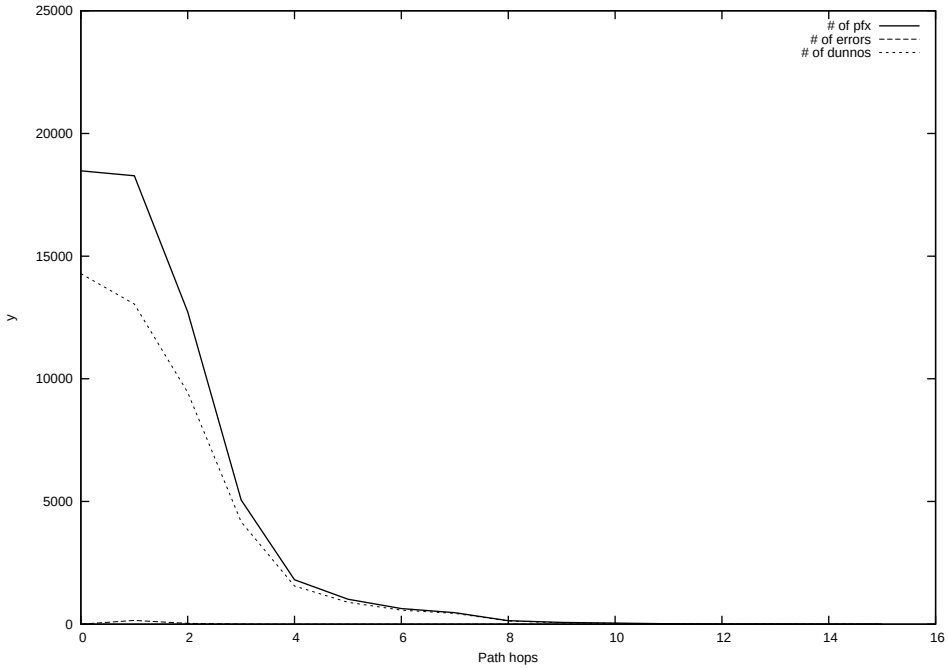
2014-06-26



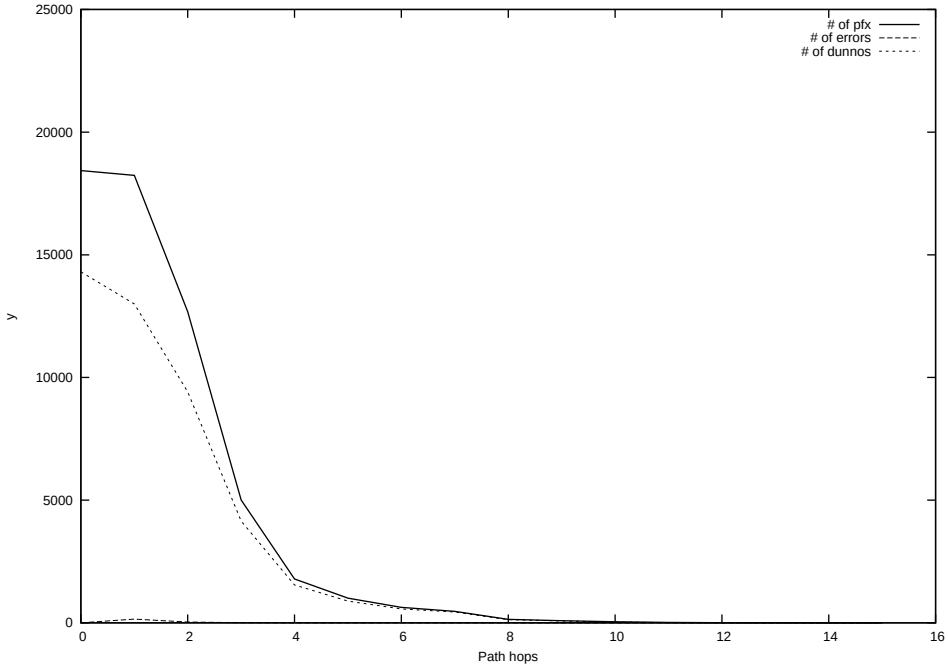
2014-06-27



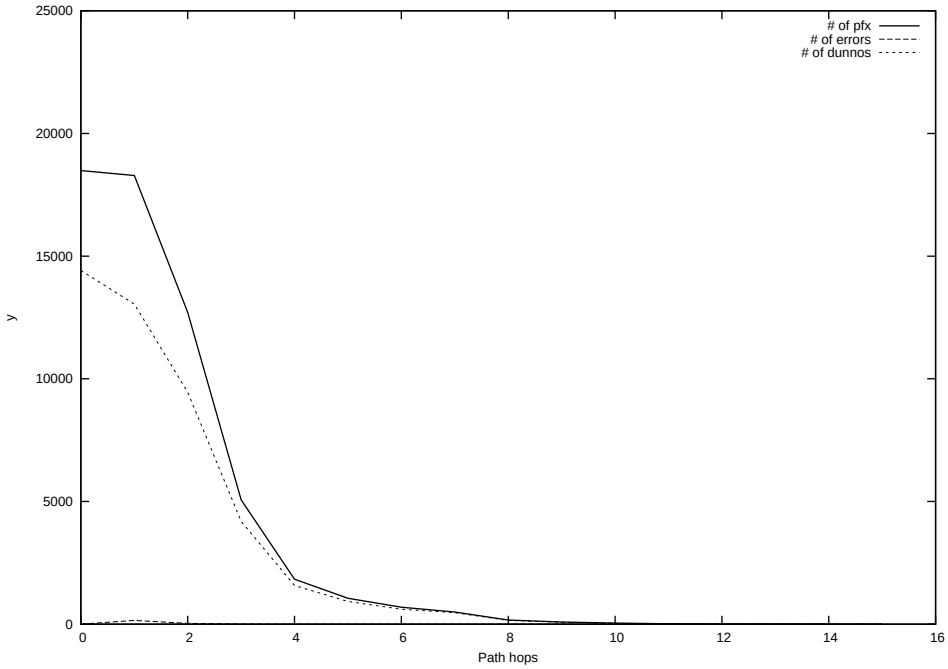
2014-06-28



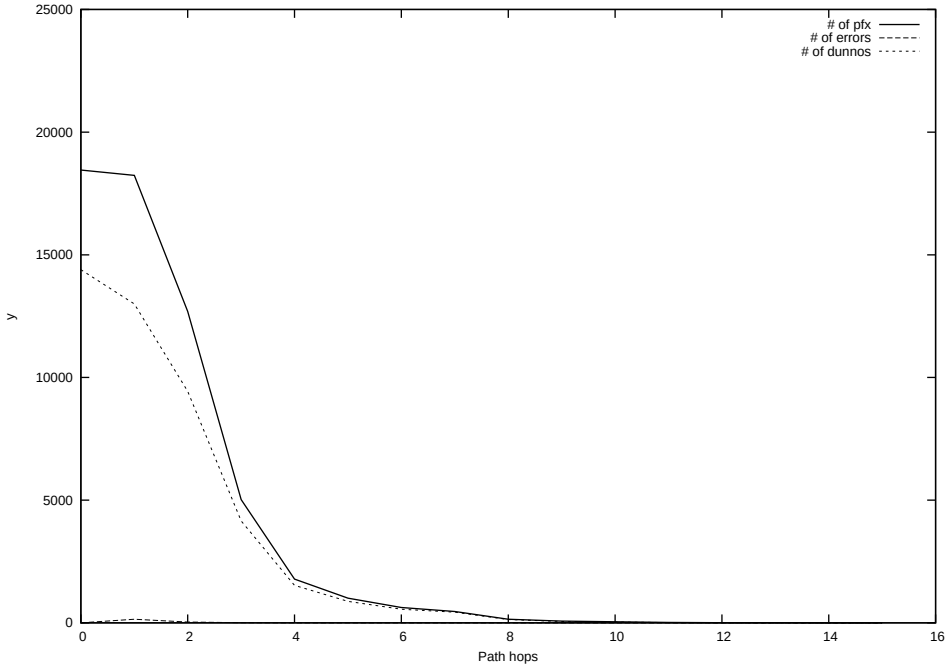
2014-06-29



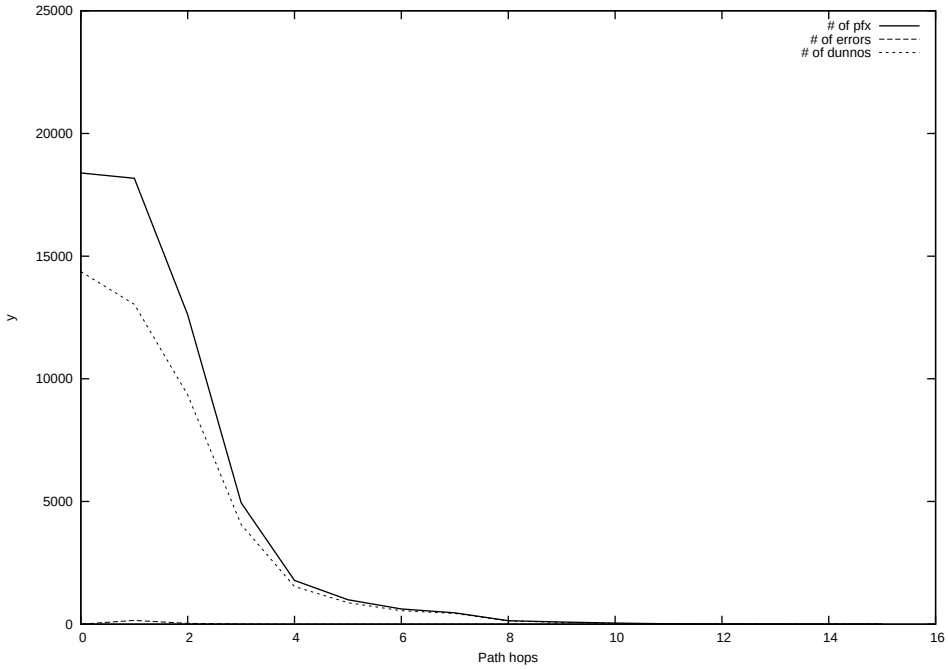
2014-06-30



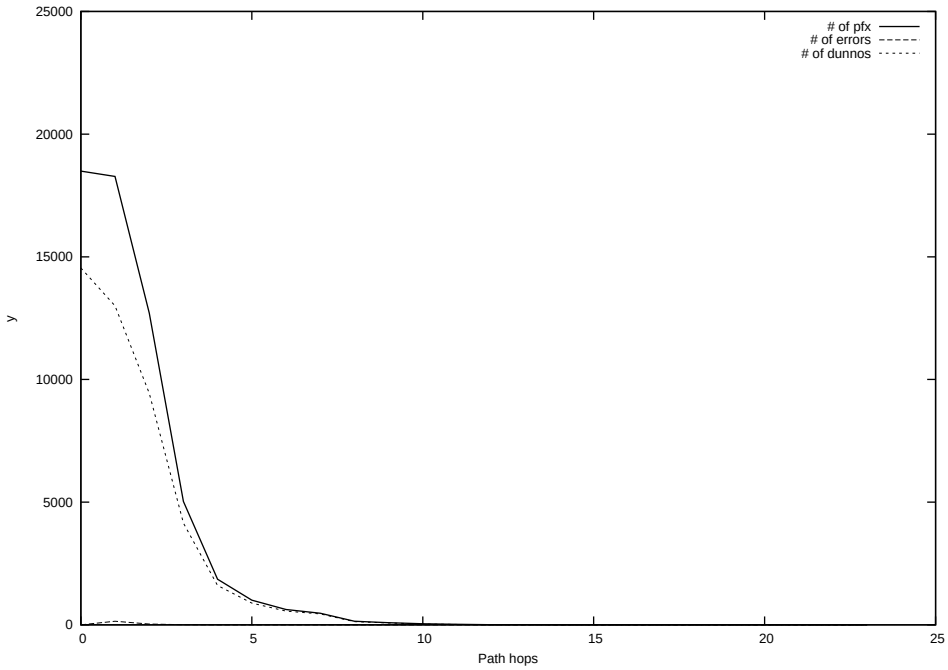
2014-07-01



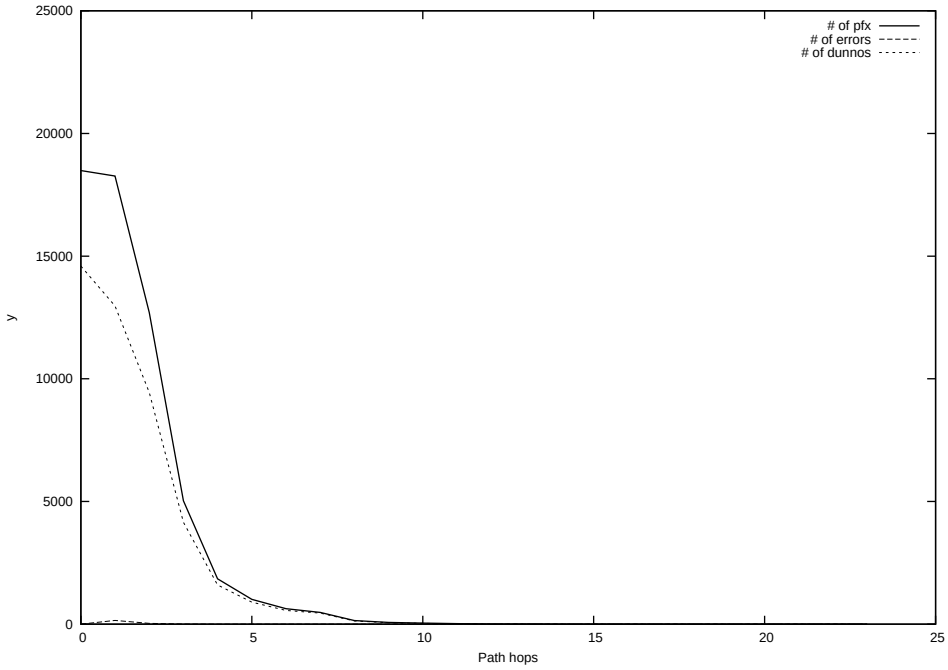
2014-07-02



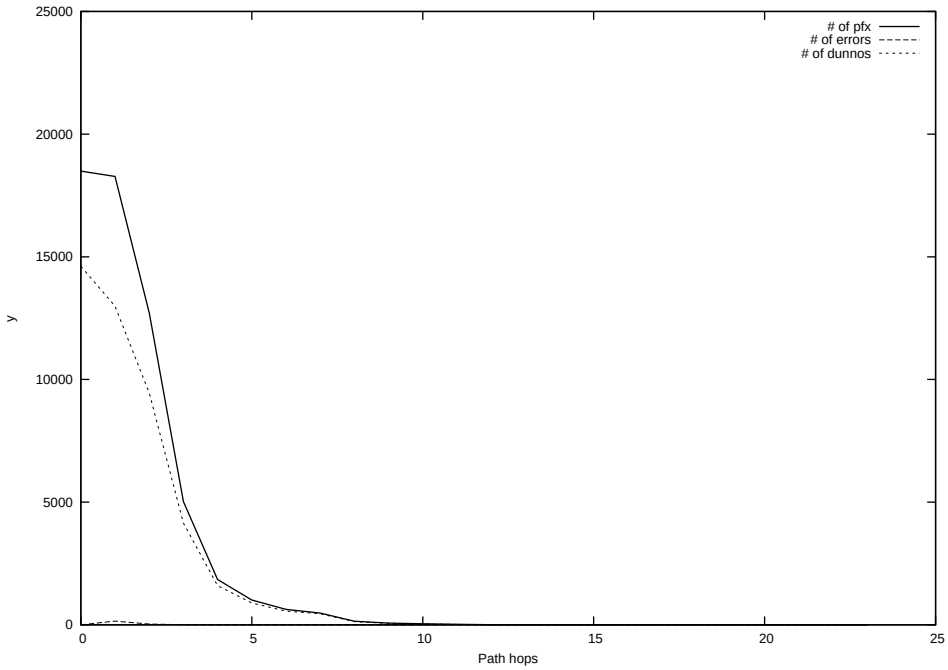
2014-07-03



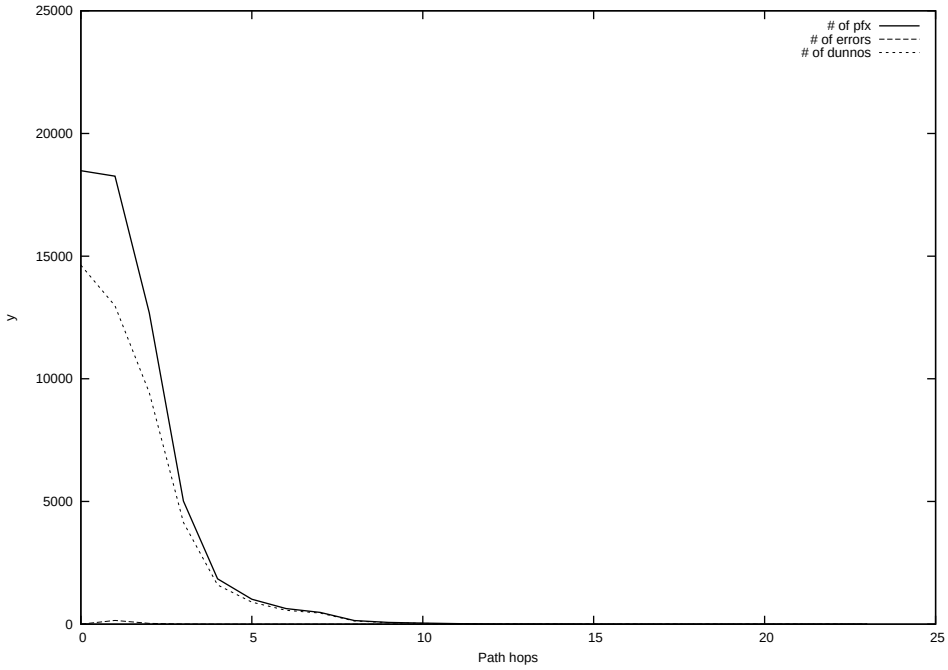
2014-07-04



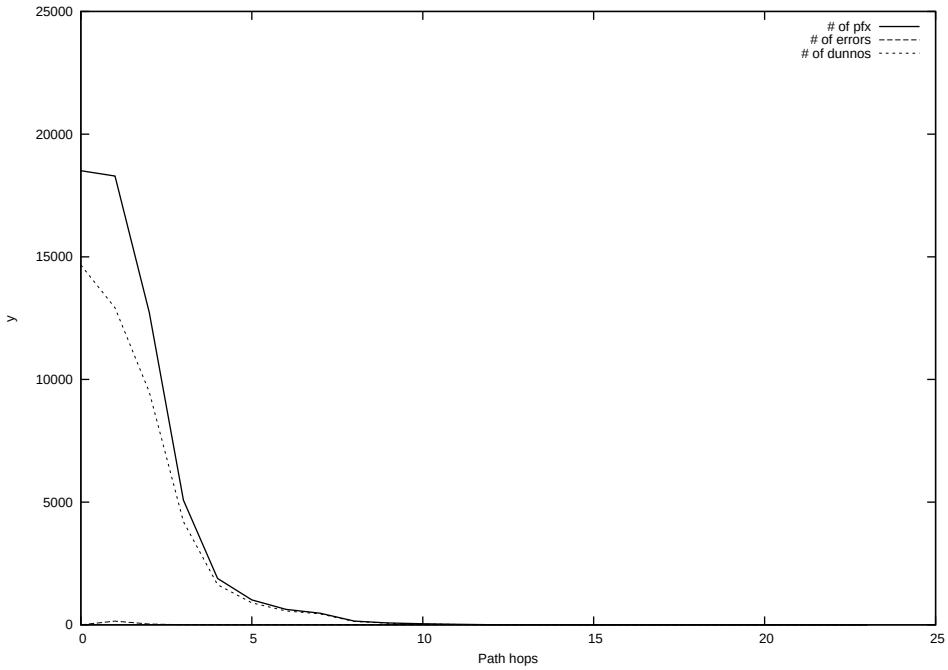
2014-07-05



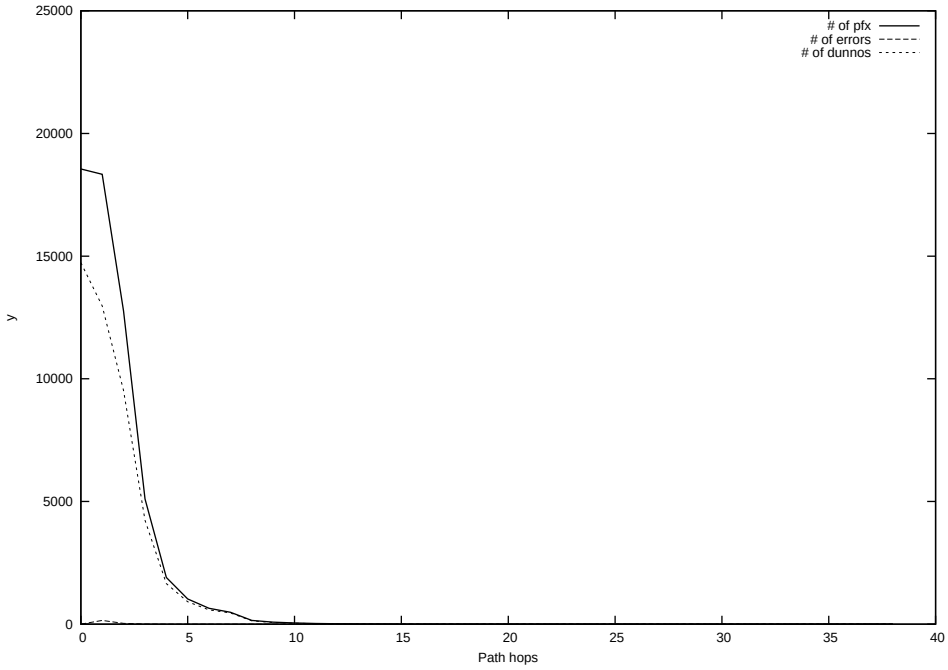
2014-07-06



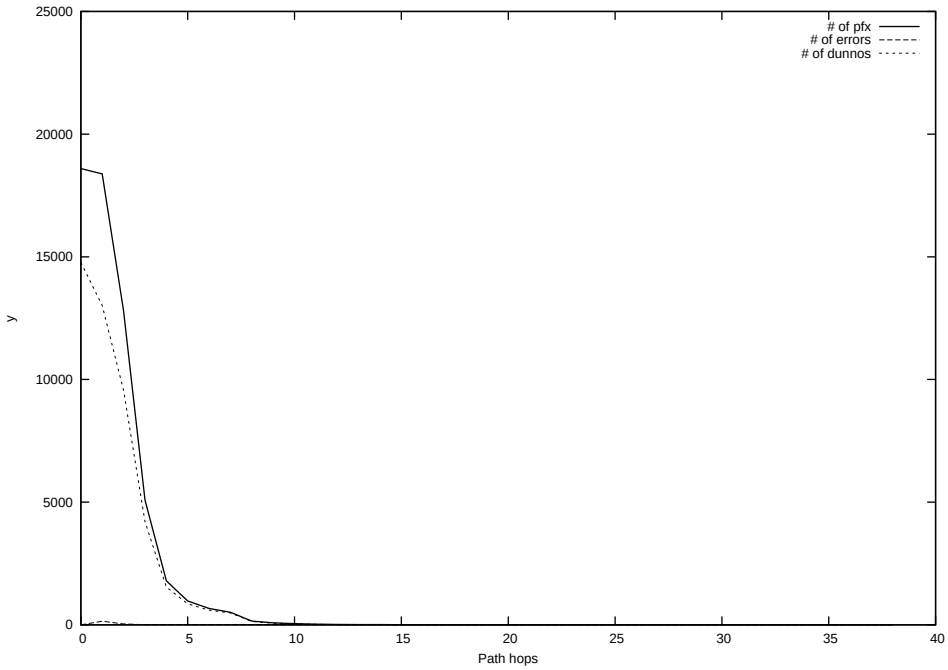
2014-07-07



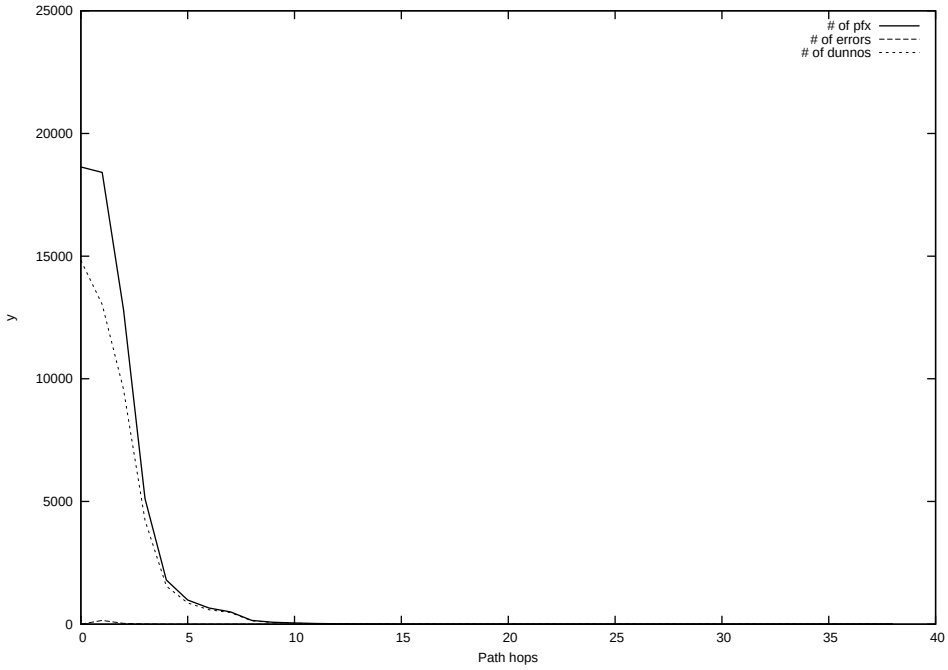
2014-07-08



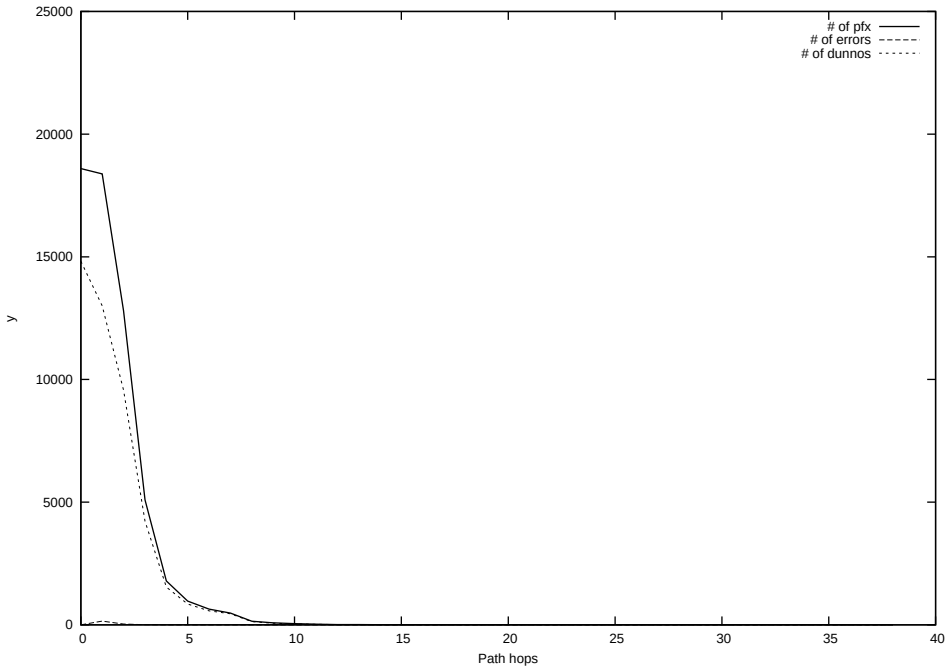
2014-07-09



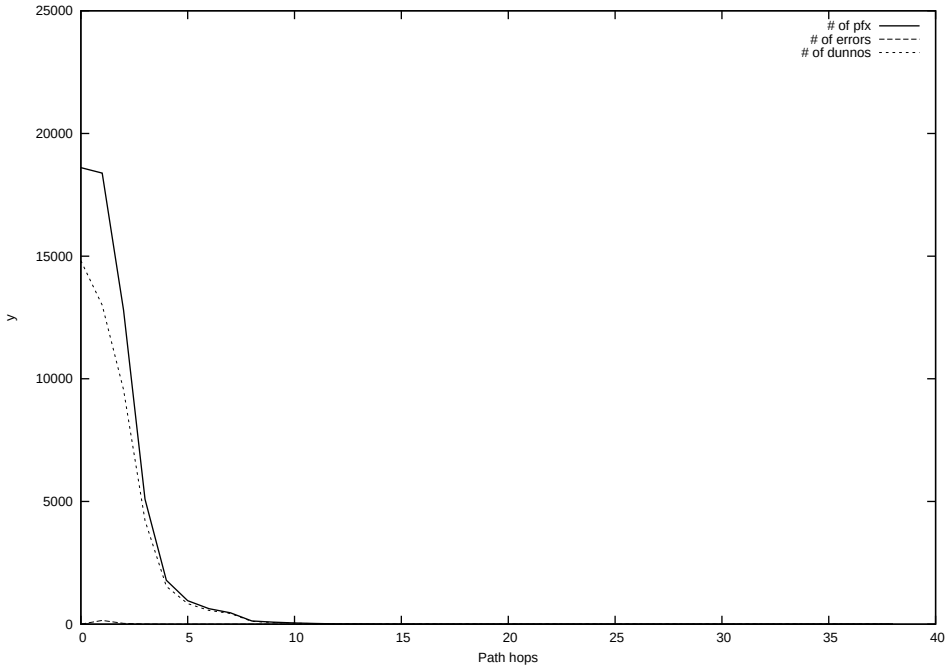
2014-07-10



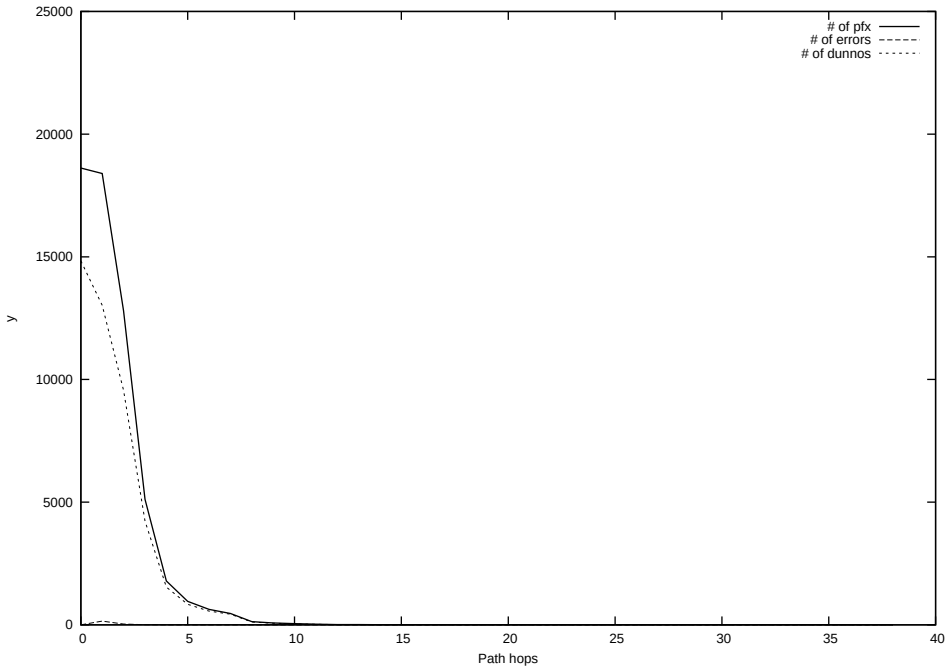
2014-07-11



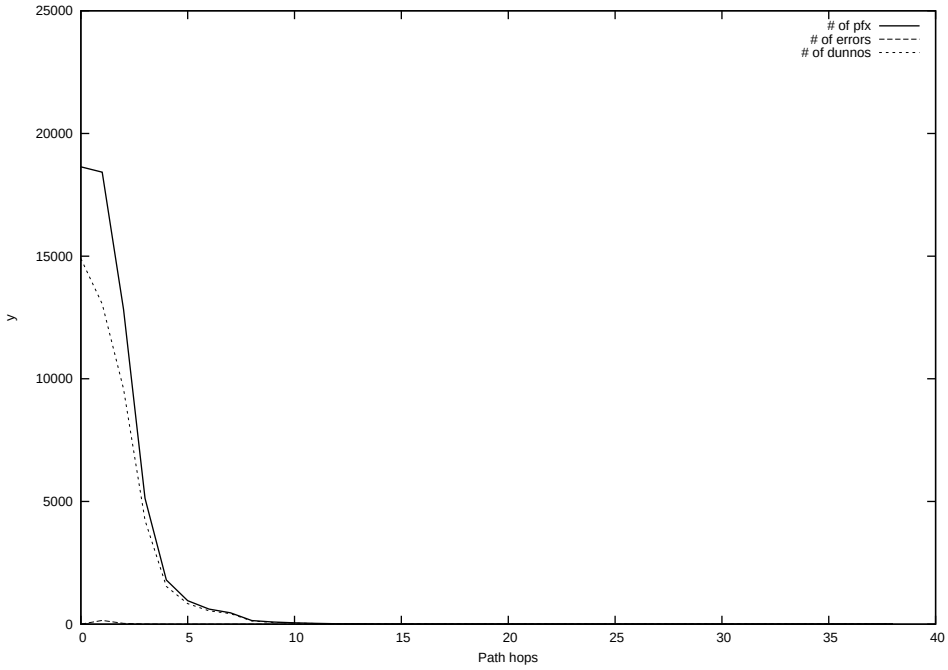
2014-07-12



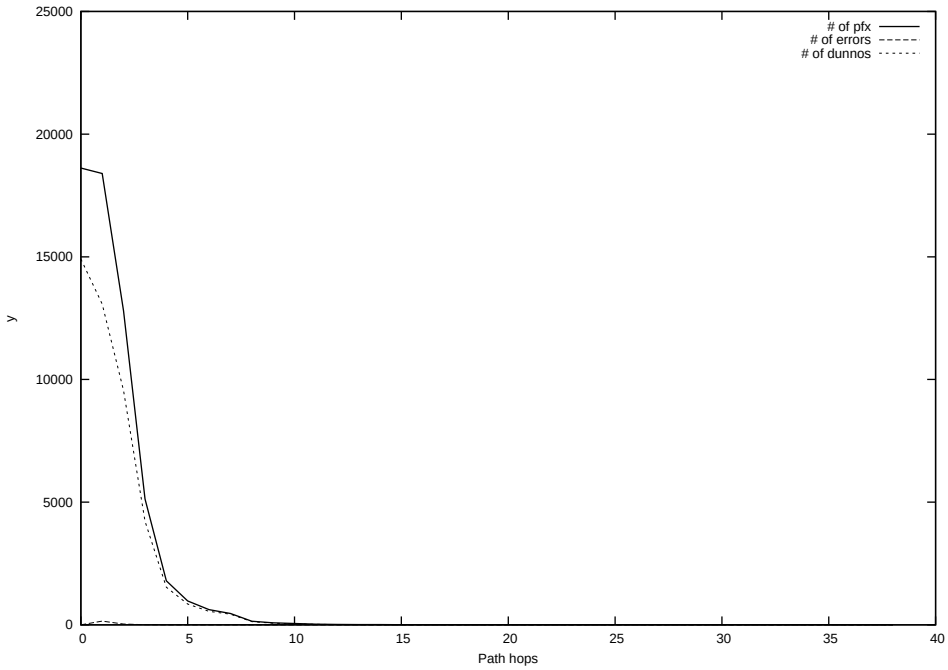
2014-07-13



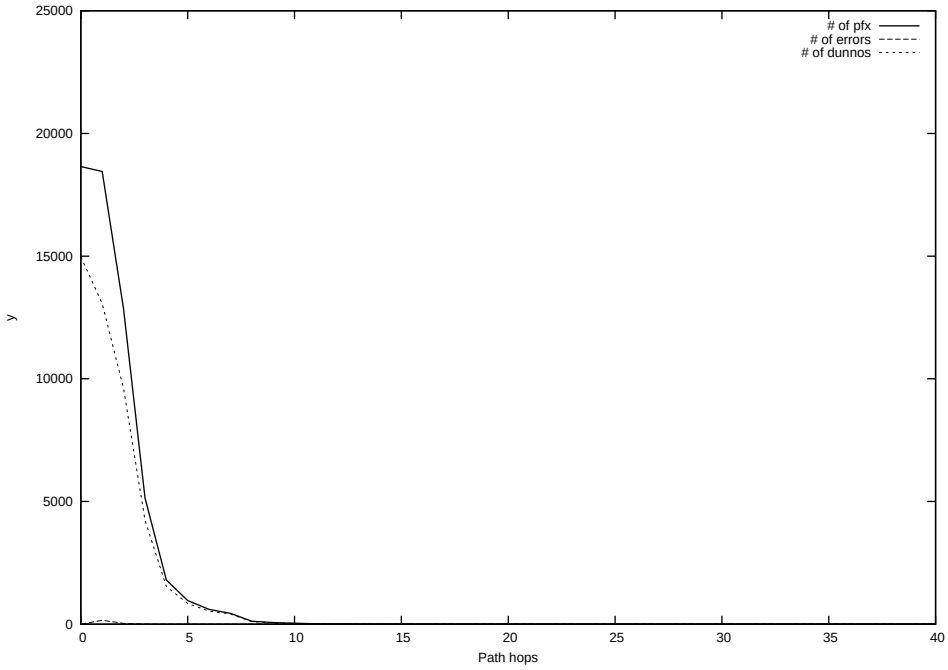
2014-07-14



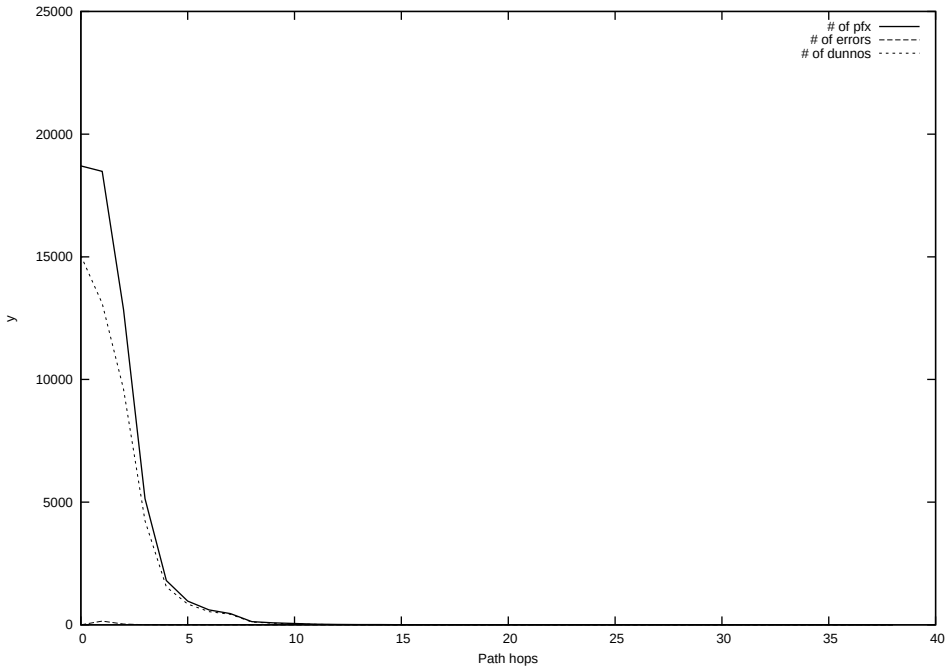
2014-07-15



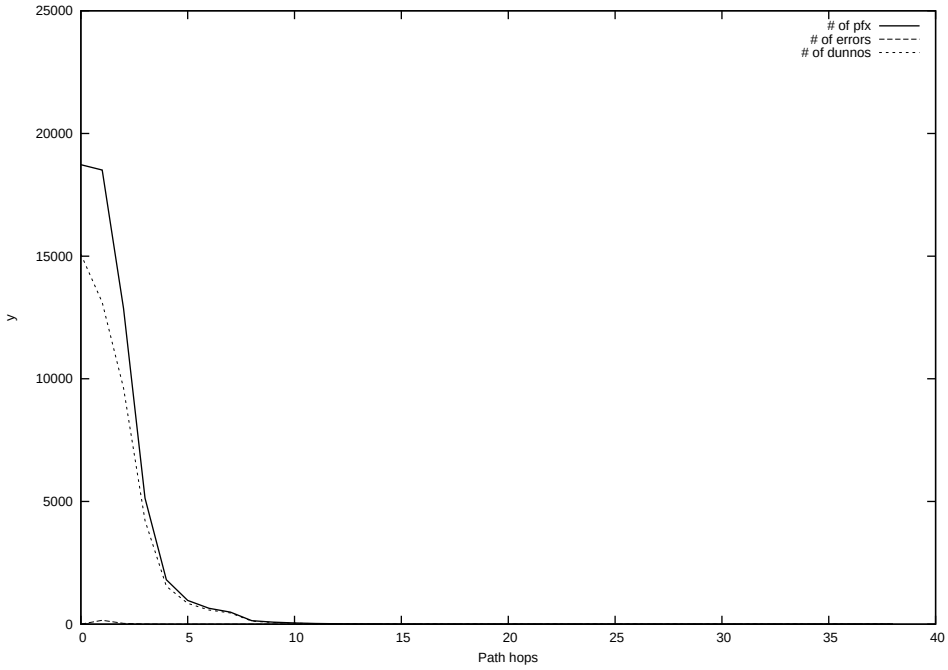
2014-07-16



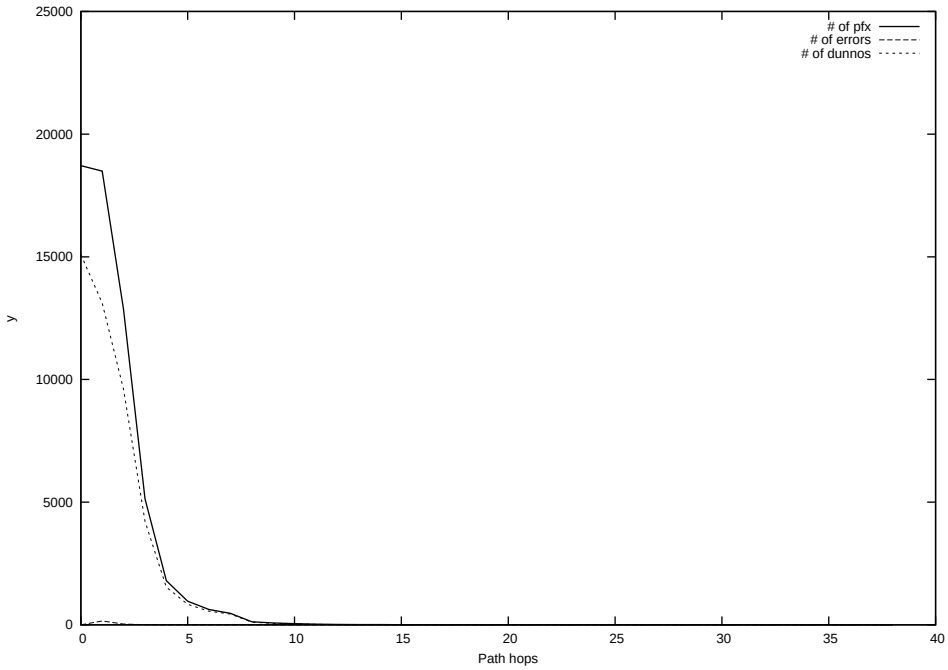
2014-07-17



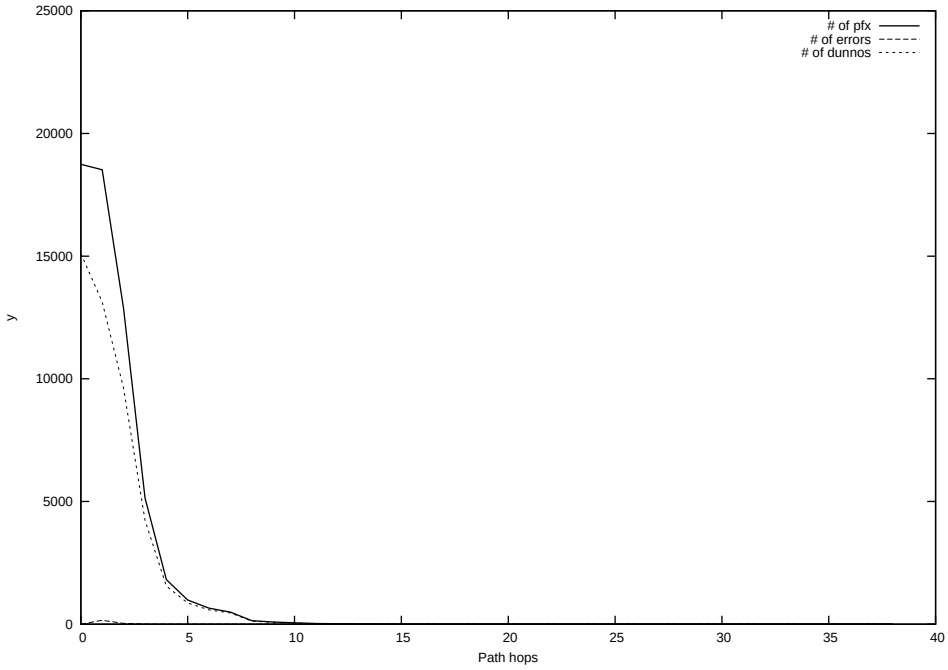
2014-07-18



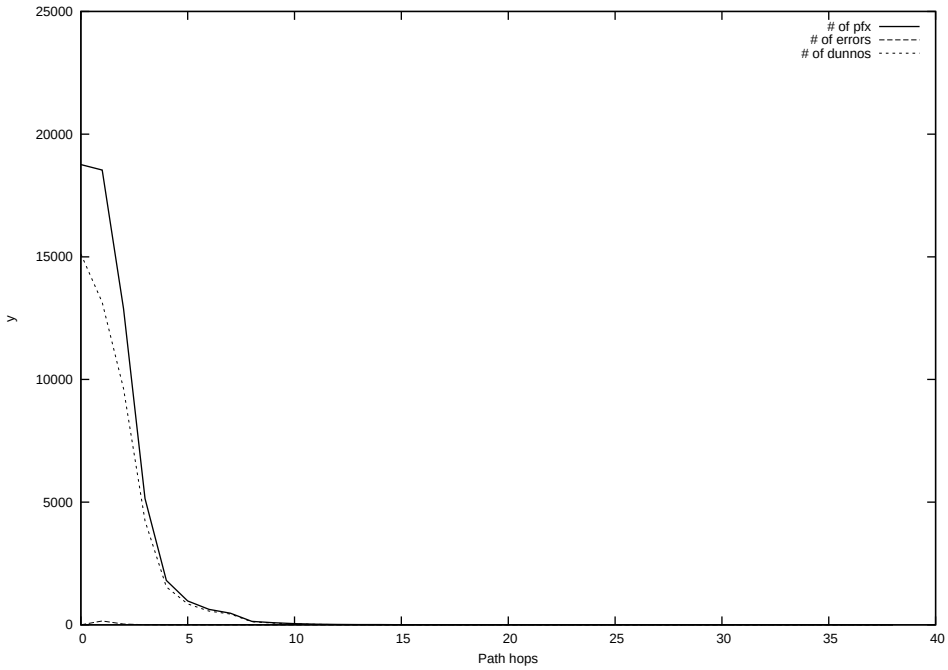
2014-07-19



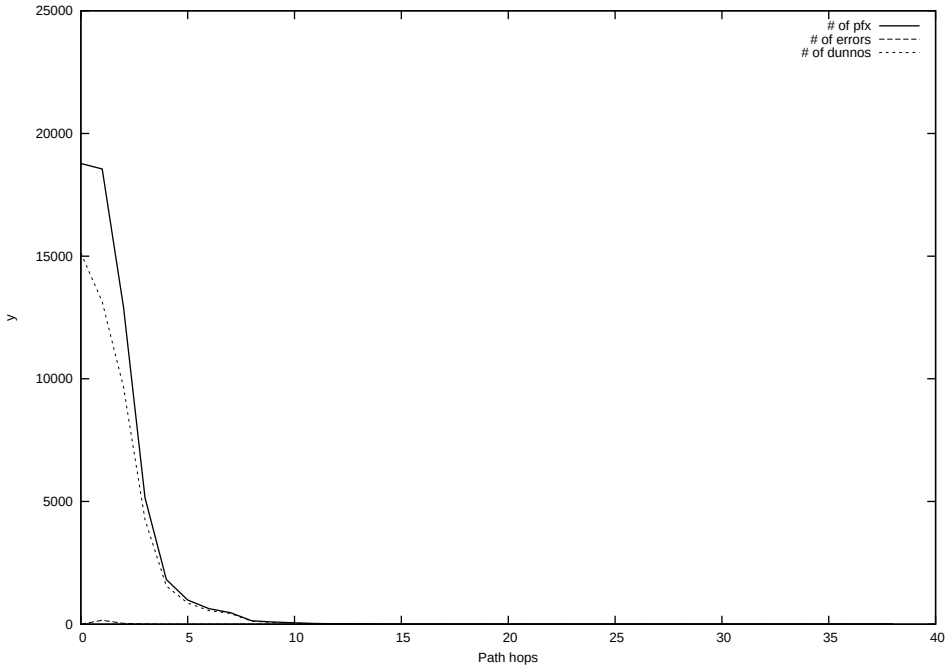
2014-07-20



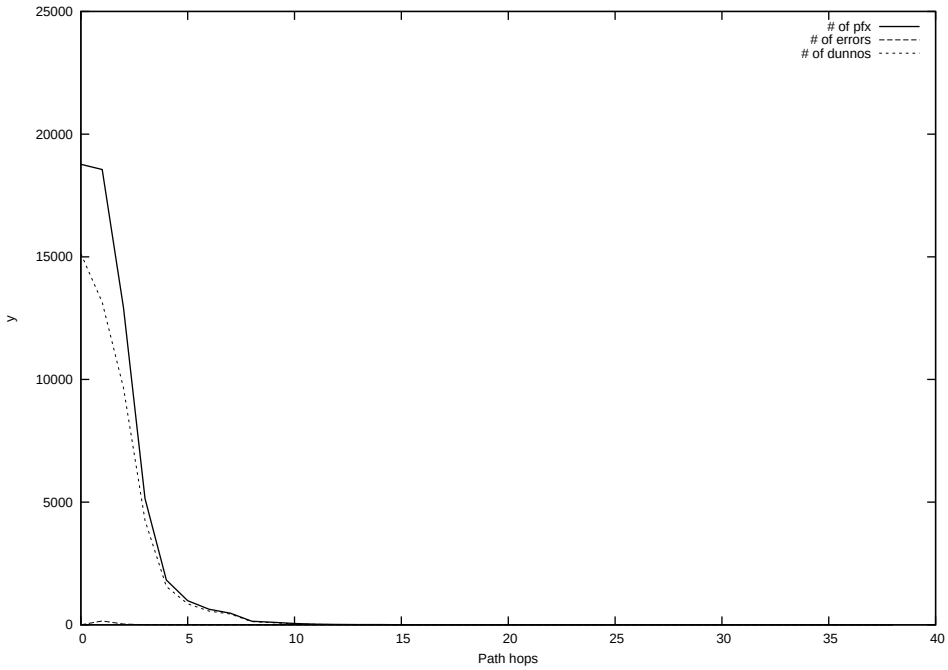
2014-07-21



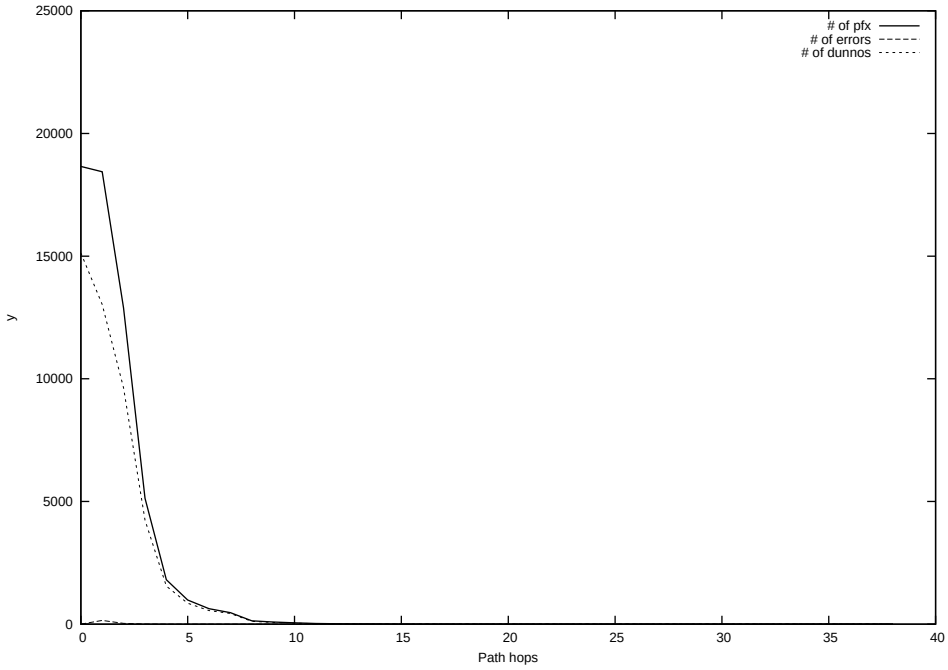
2014-07-22



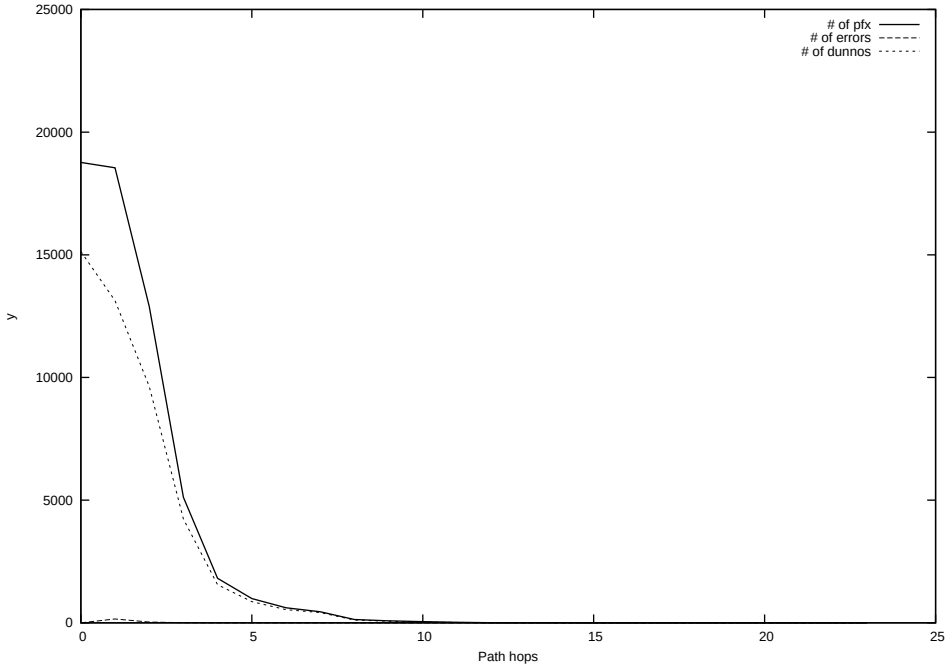
2014-07-23



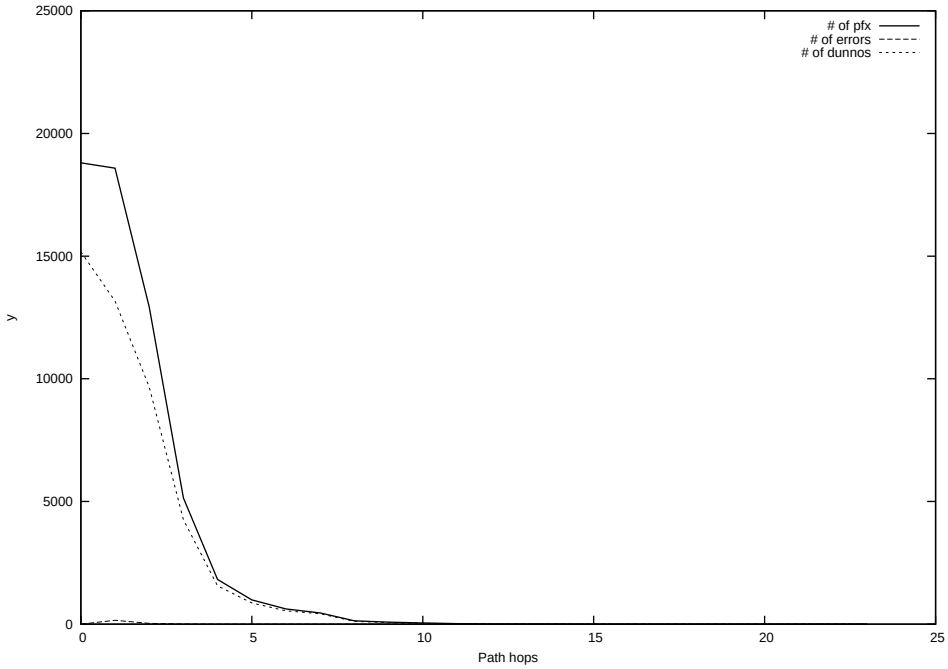
2014-07-24



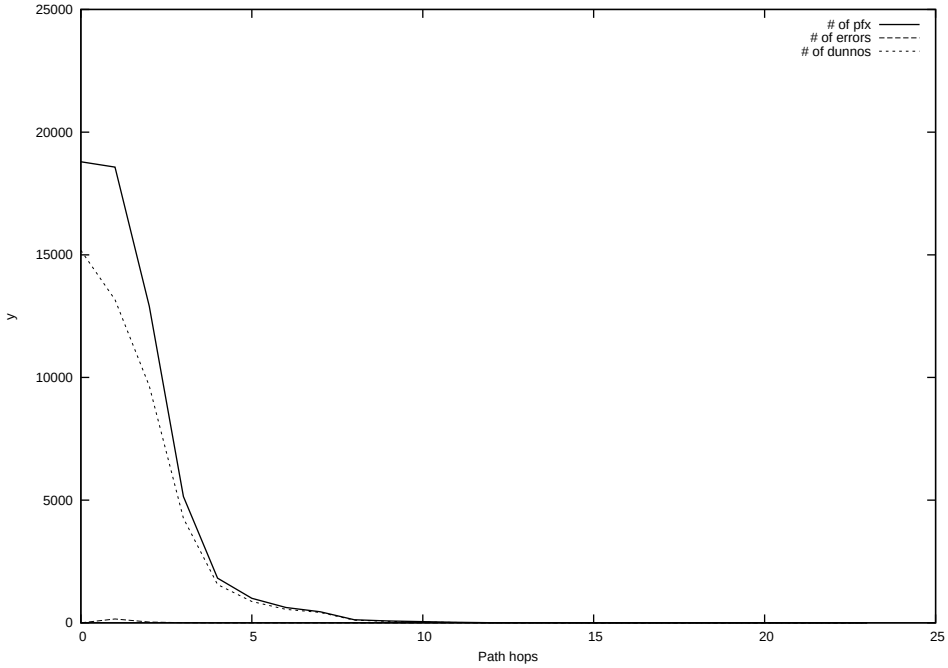
2014-07-25



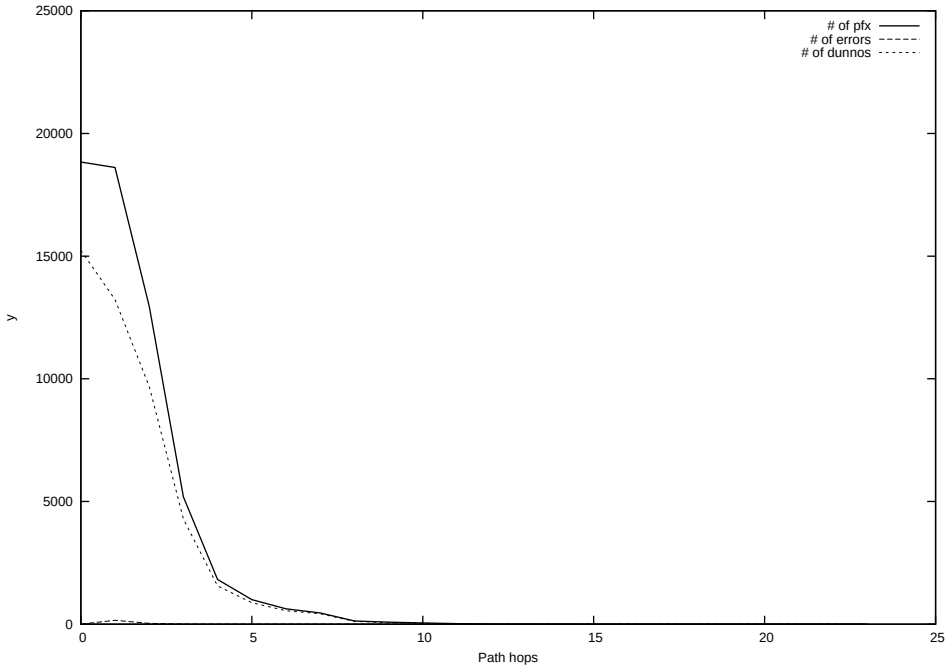
2014-07-26



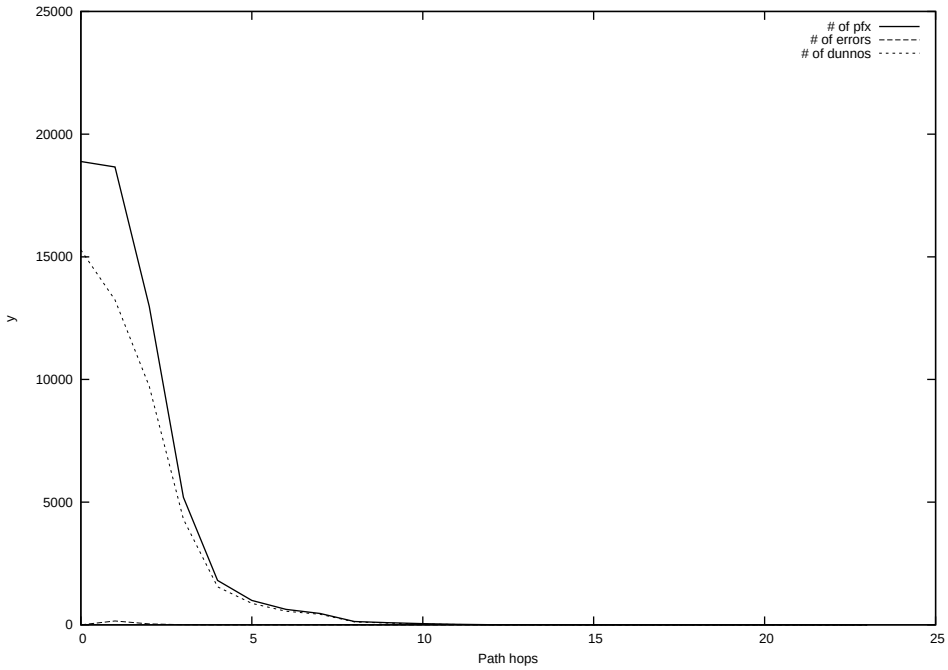
2014-07-27



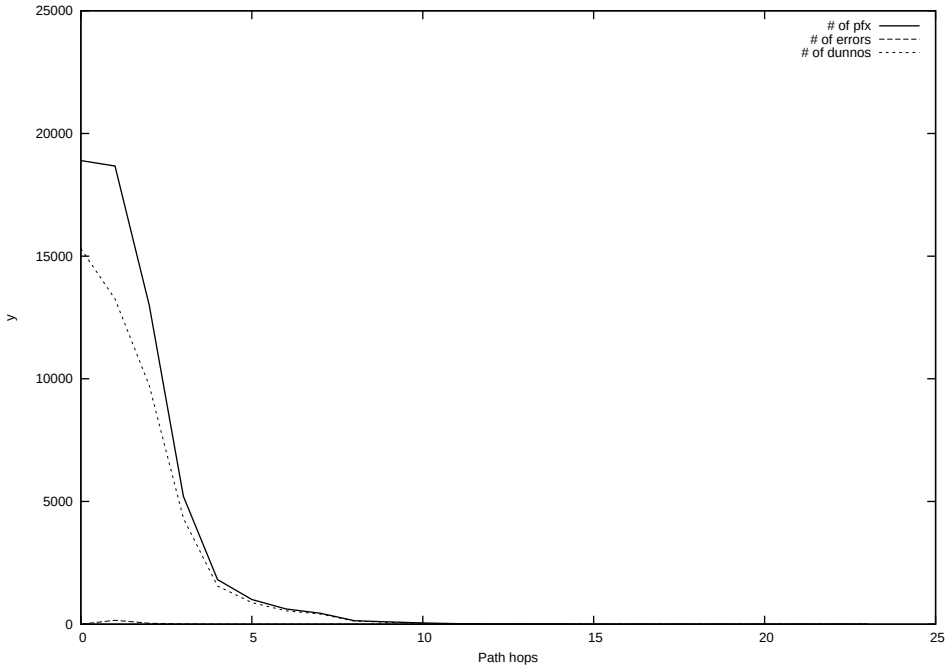
2014-07-28



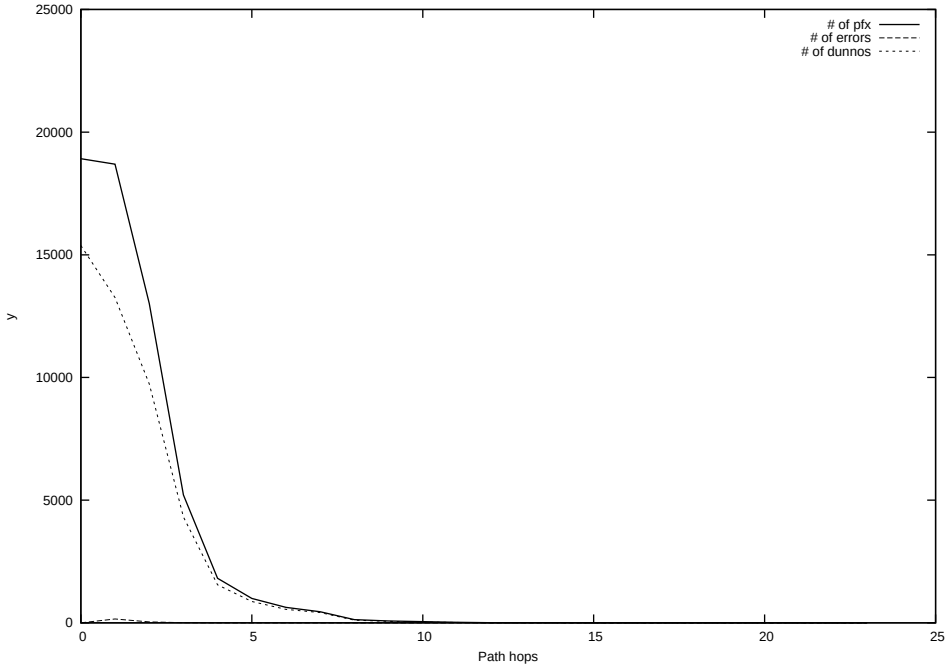
2014-07-29



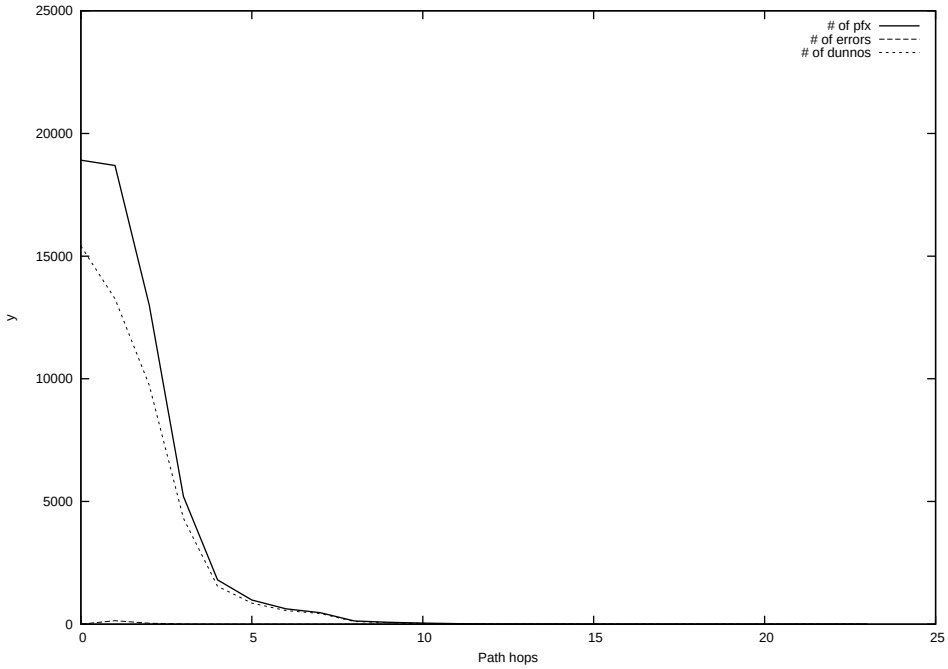
2014-07-30



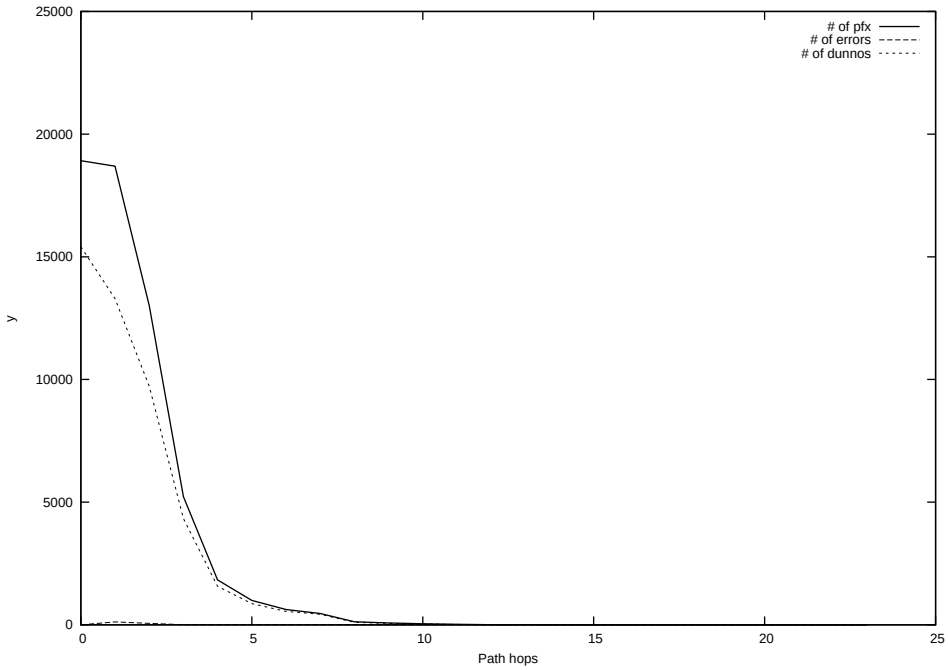
2014-07-31



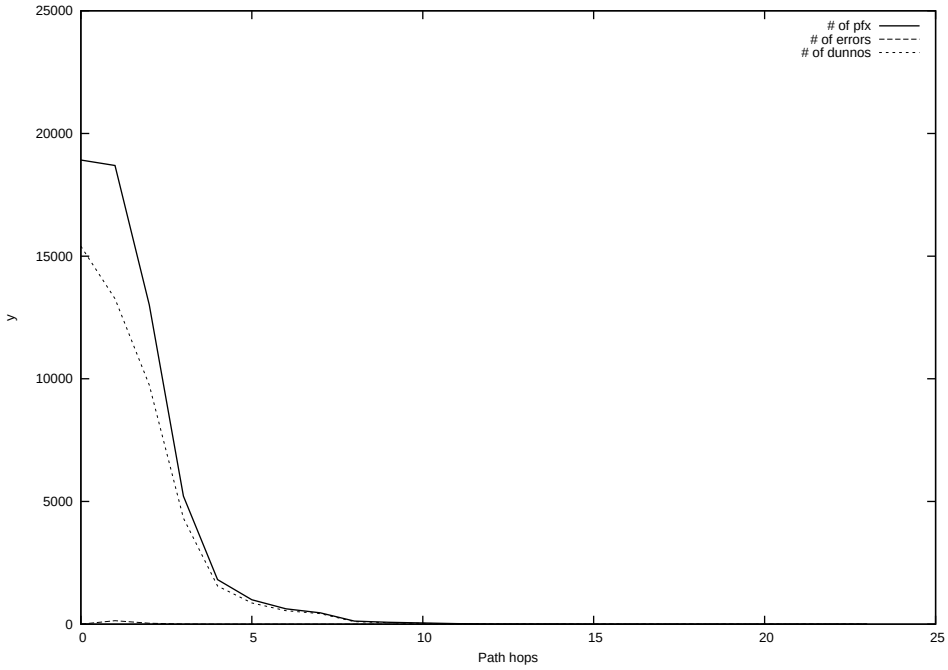
2014-08-01



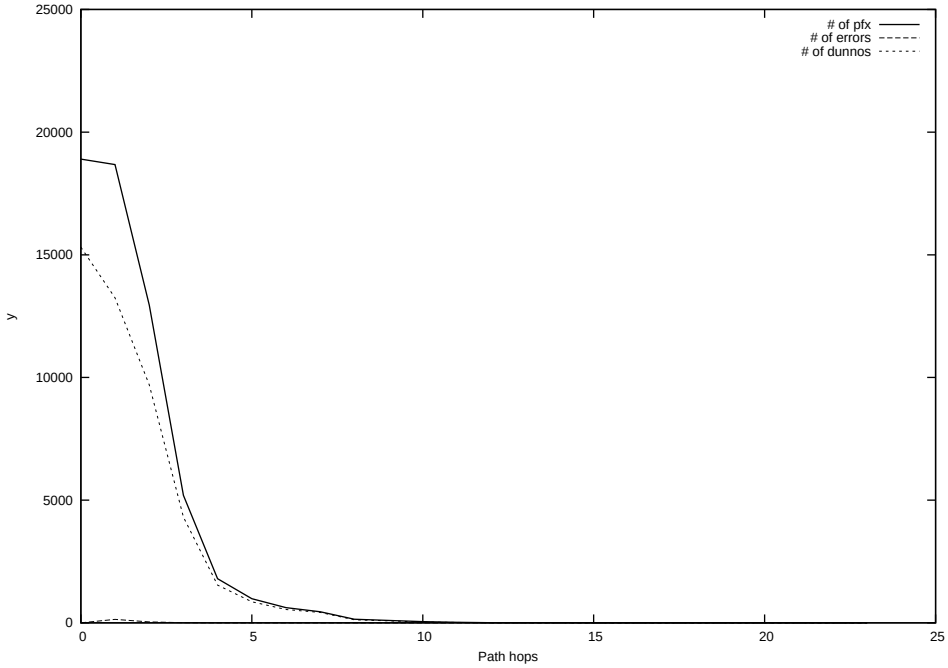
2014-08-02



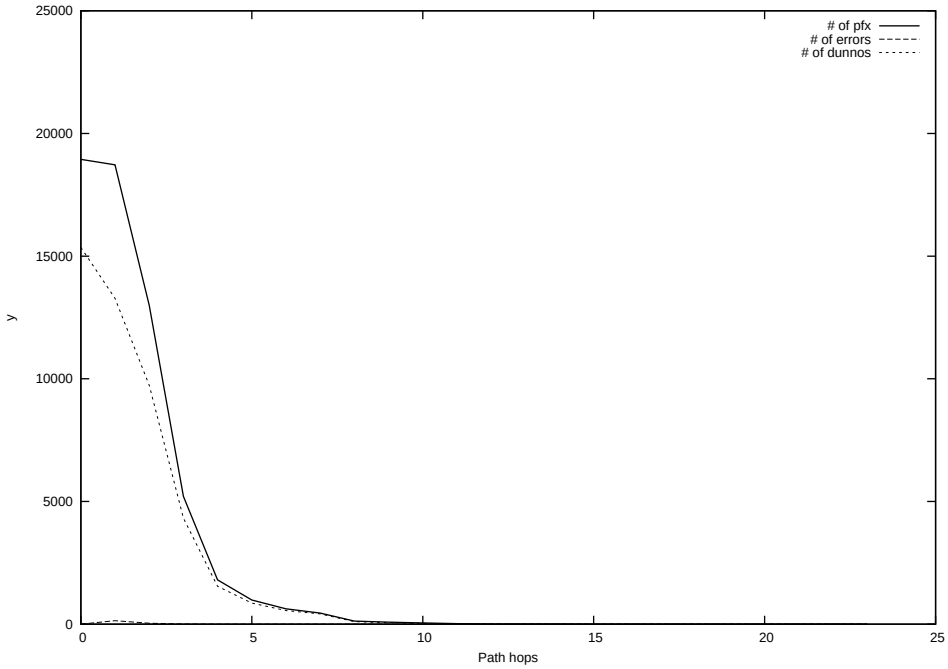
2014-08-03



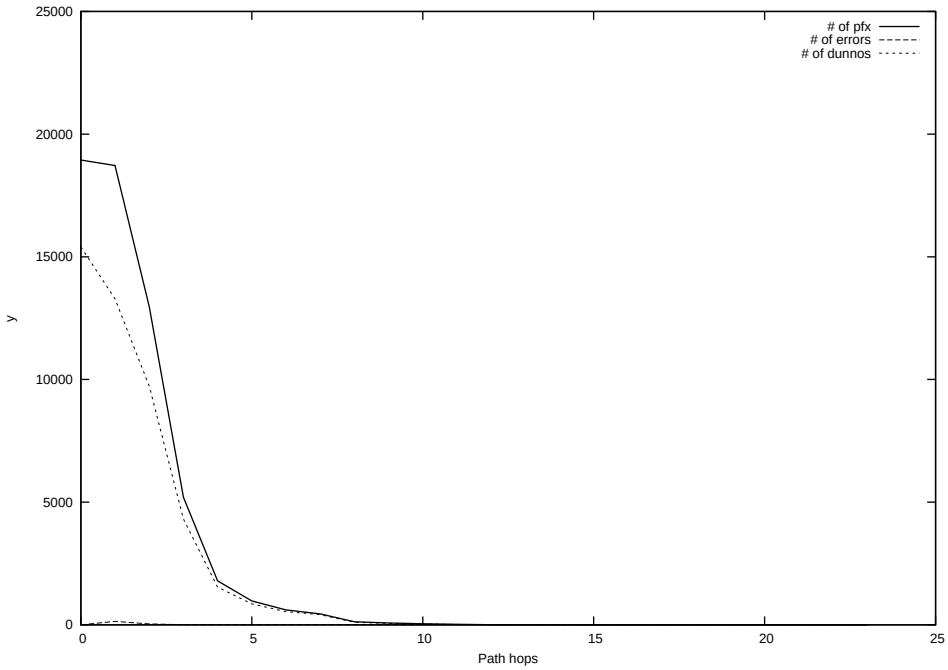
2014-08-04



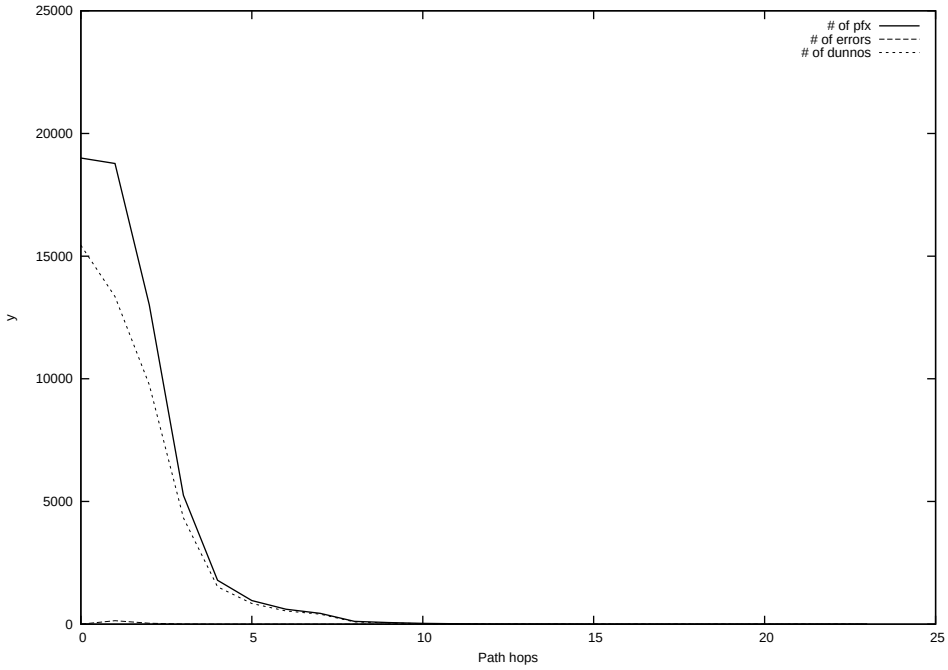
2014-08-05



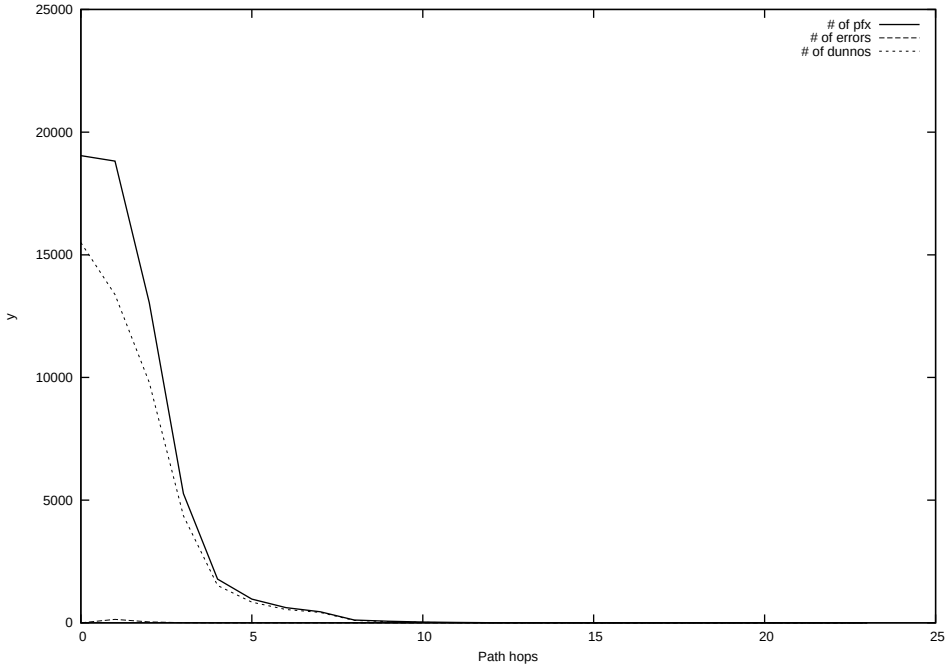
2014-08-06



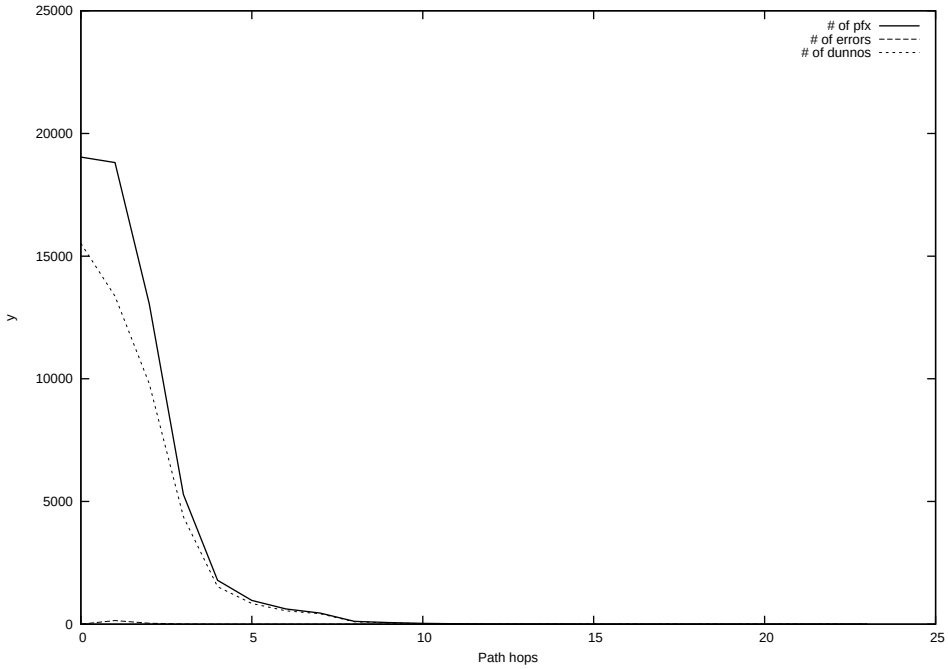
2014-08-07



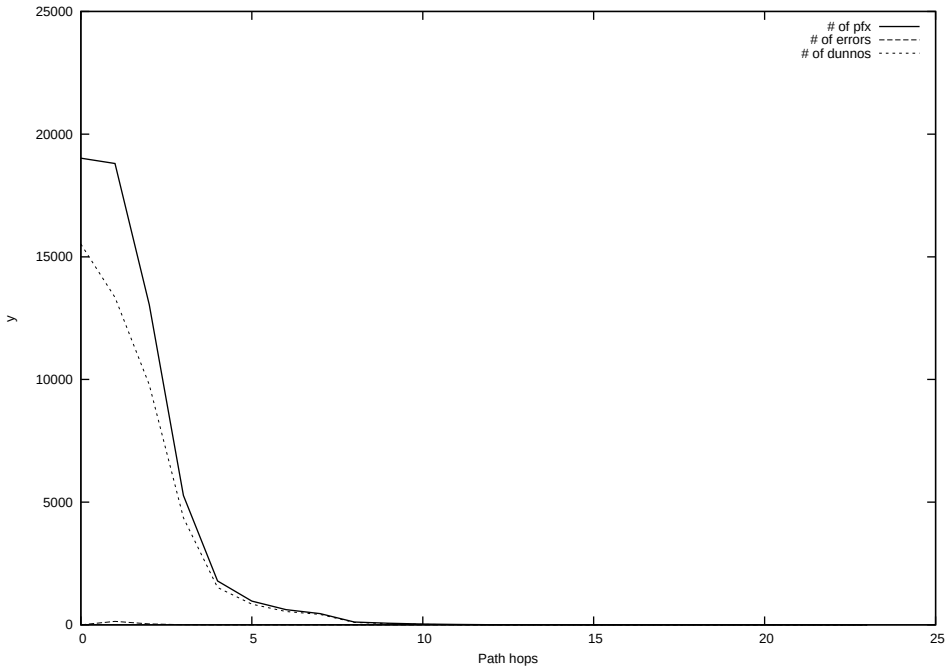
2014-08-08



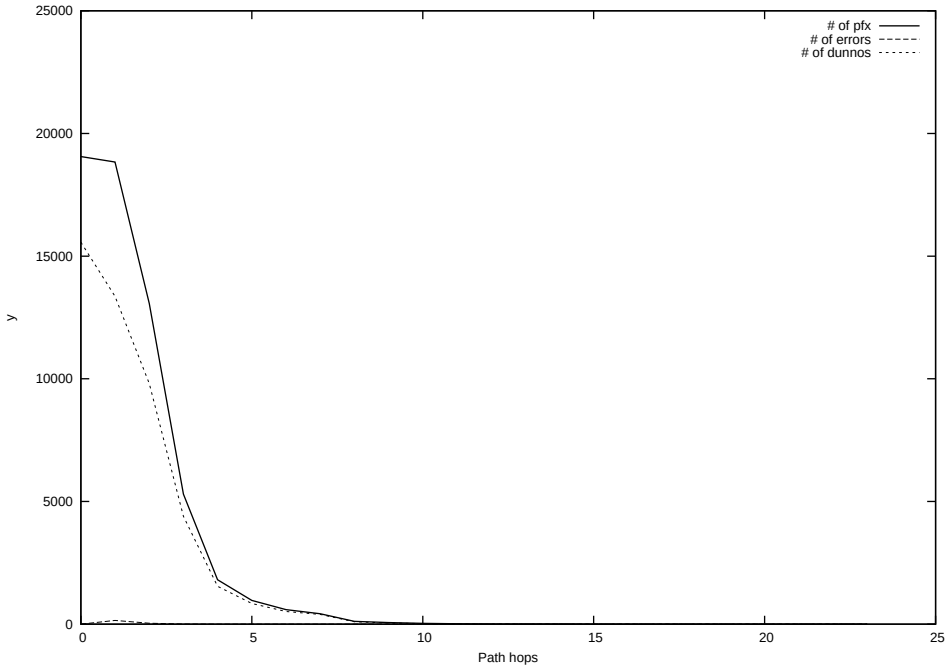
2014-08-09



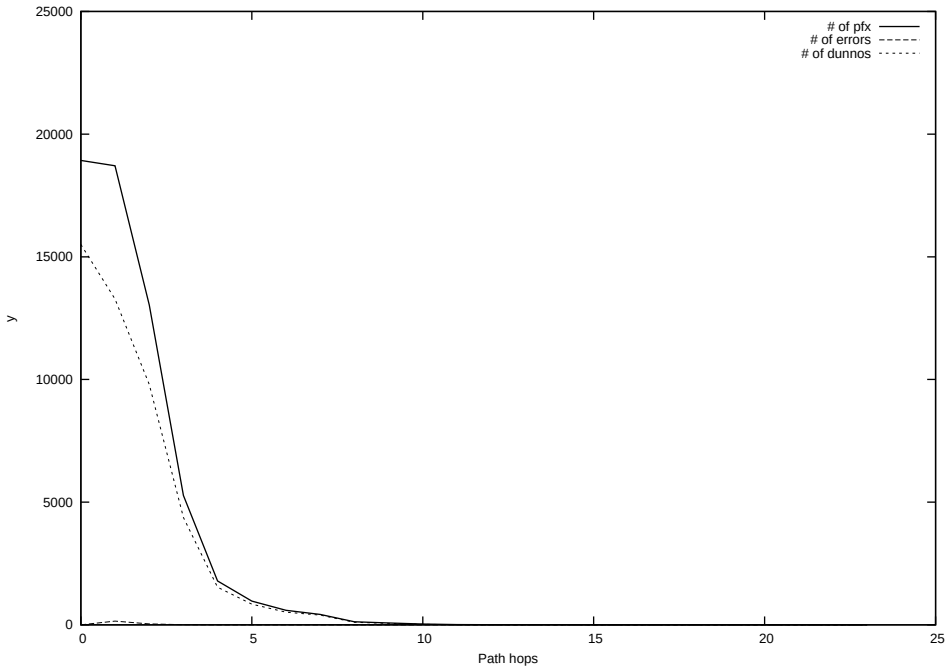
2014-08-10



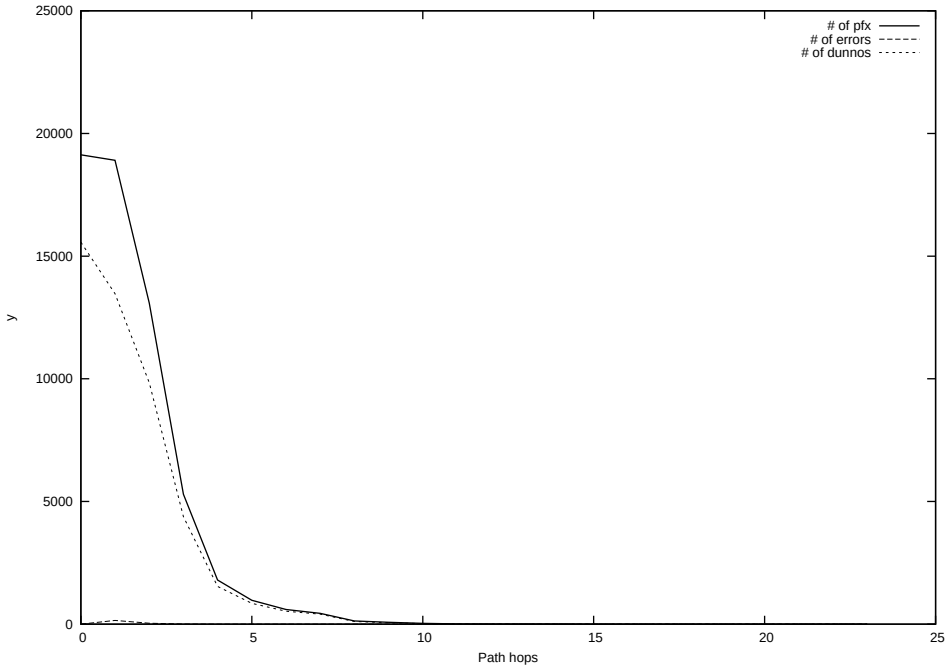
2014-08-11



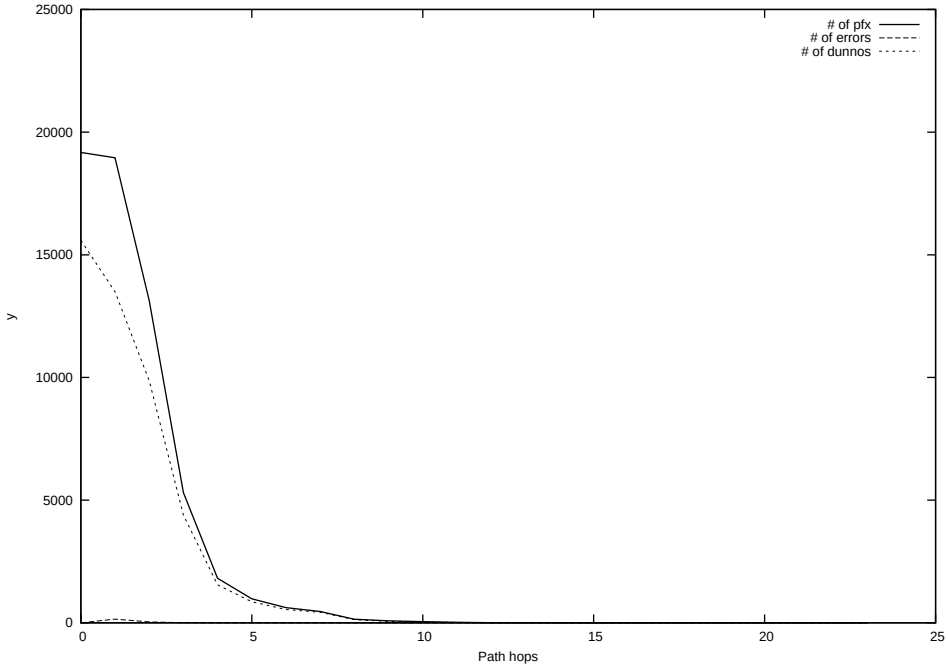
2014-08-12



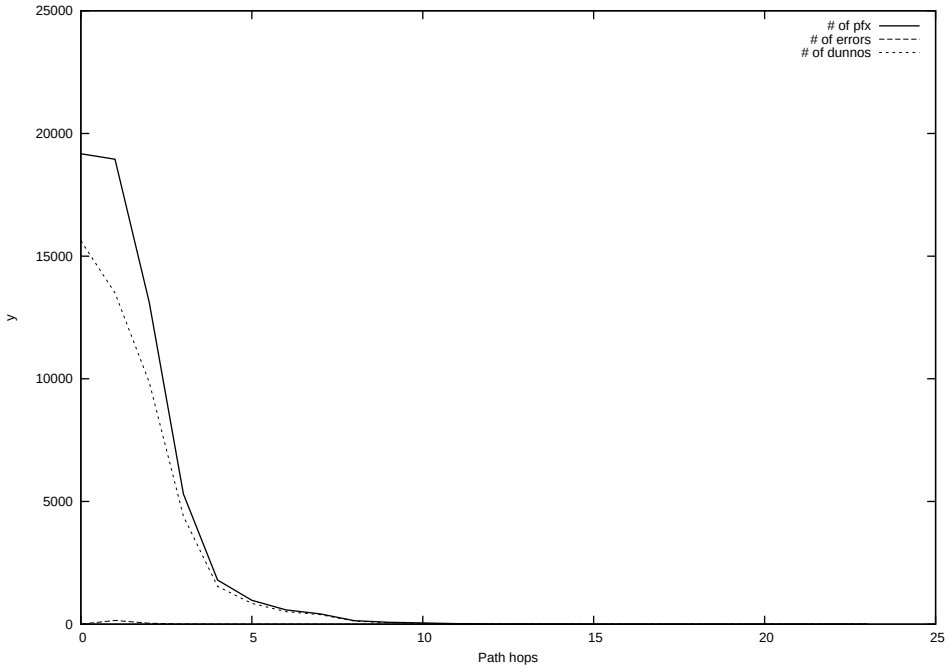
2014-08-13



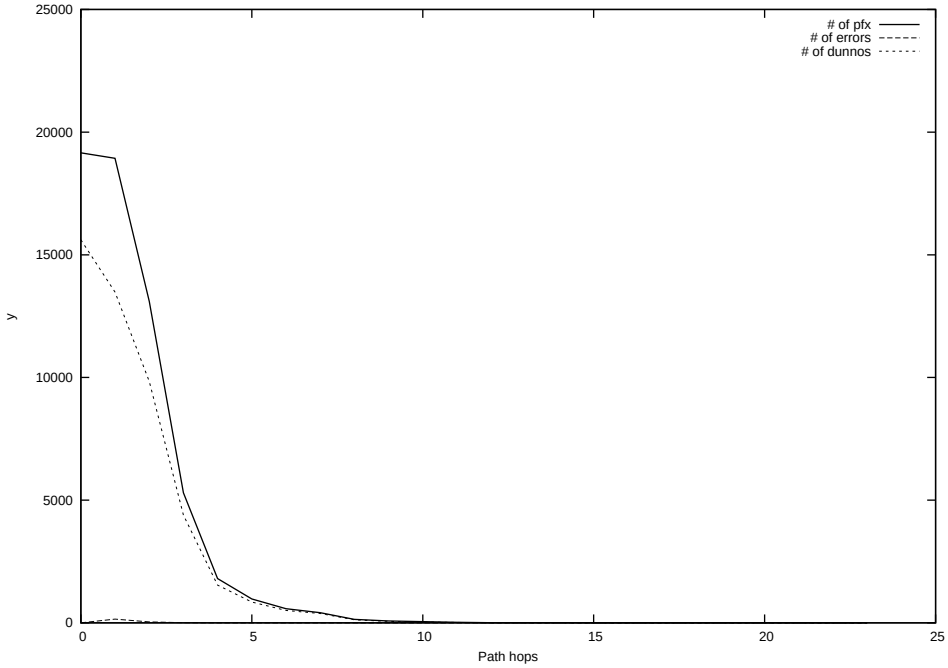
2014-08-14



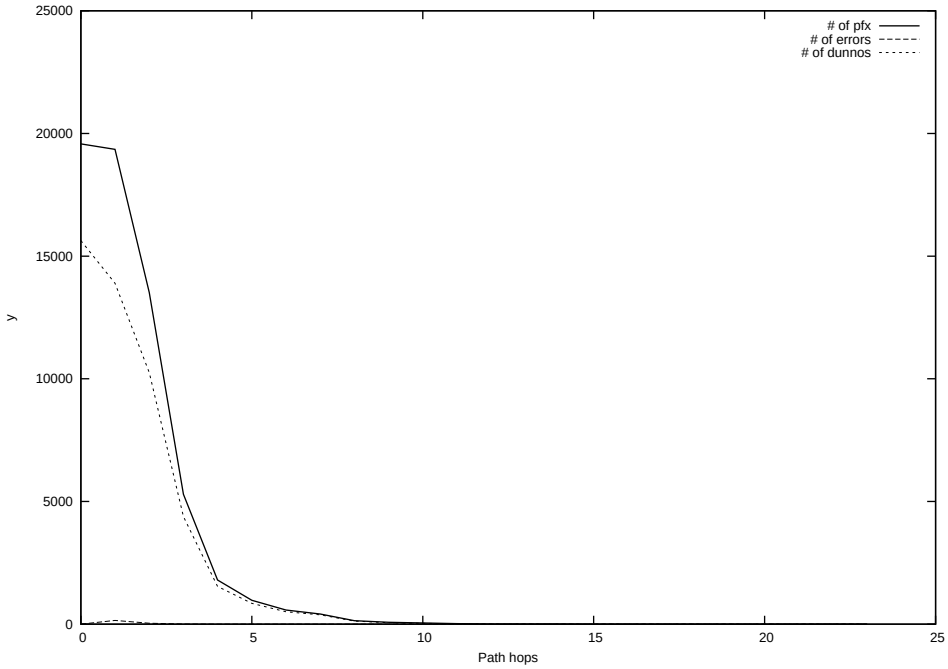
2014-08-15



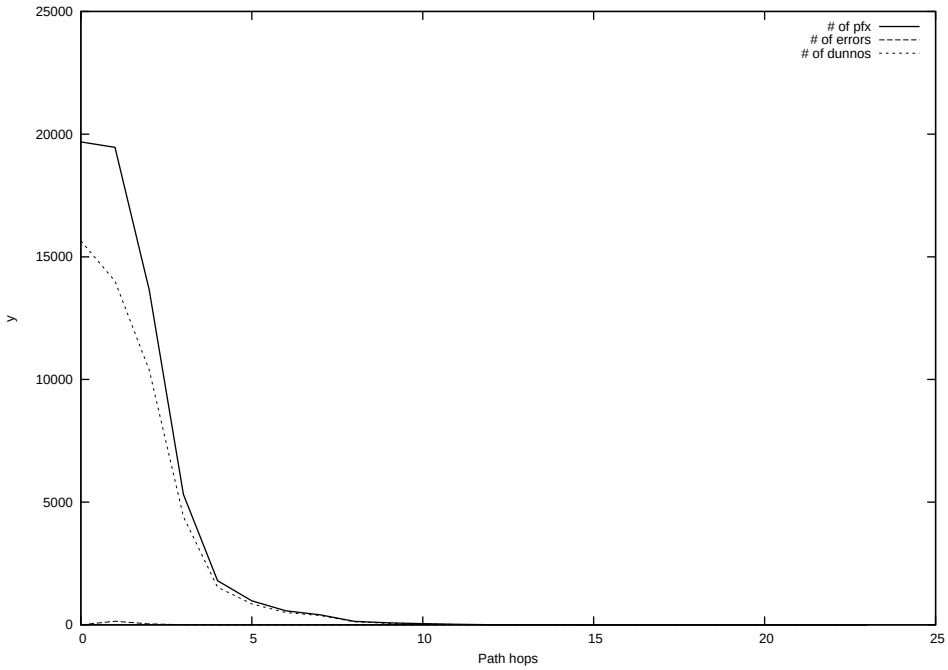
2014-08-16



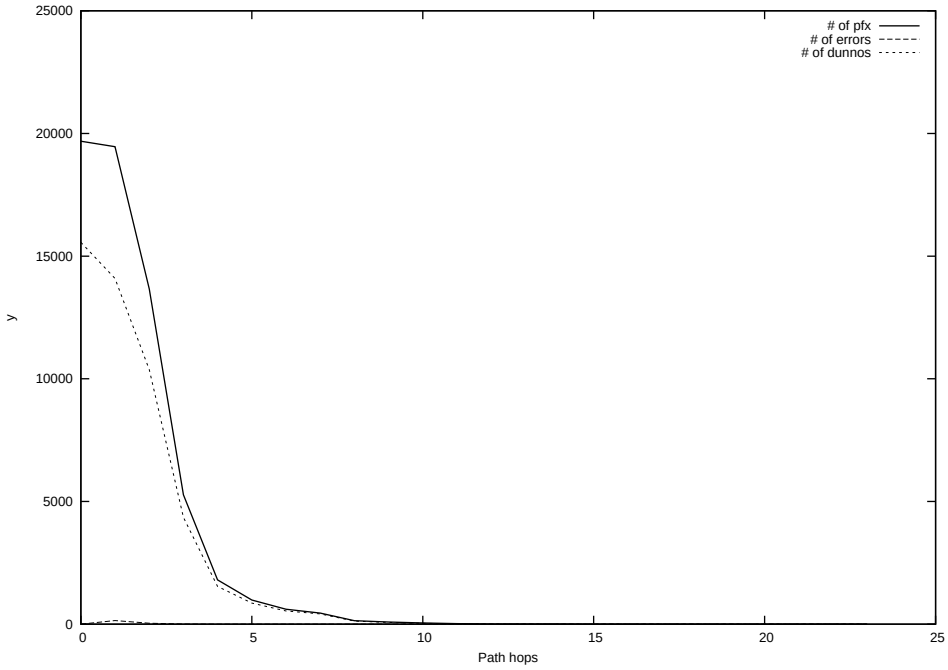
2014-08-17



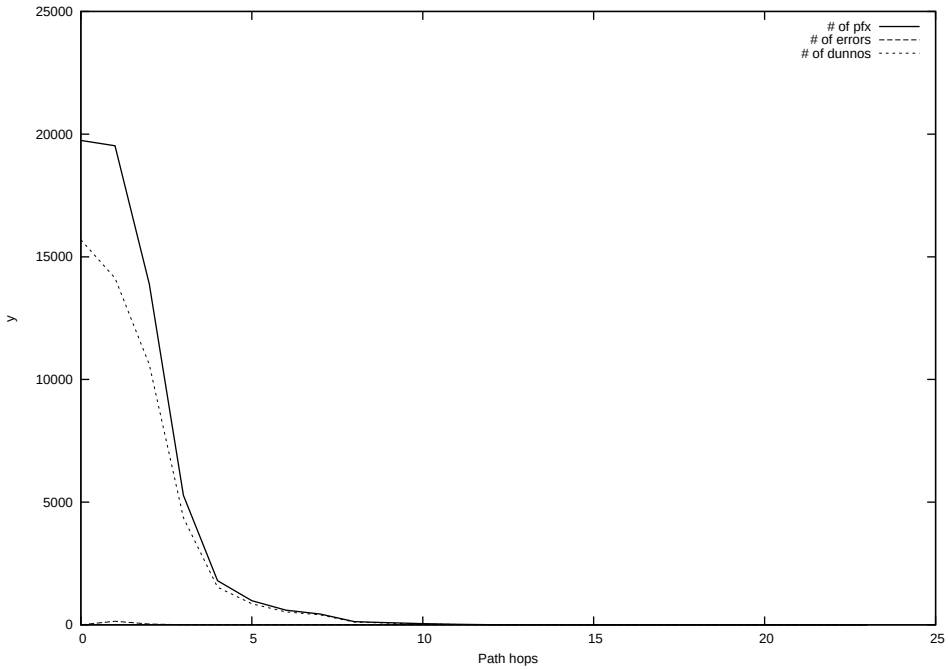
2014-08-18



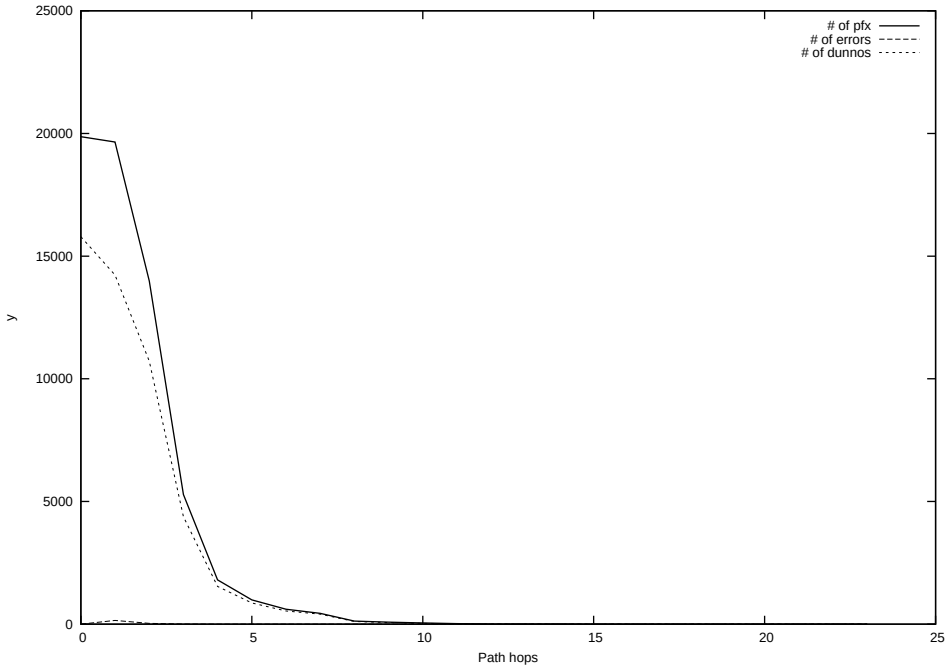
2014-08-19



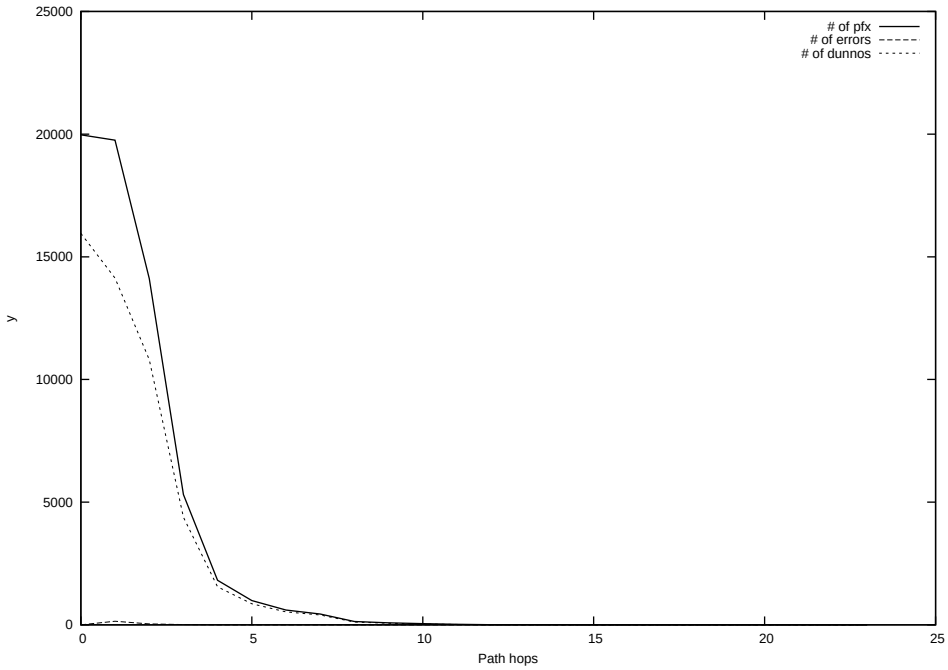
2014-08-20



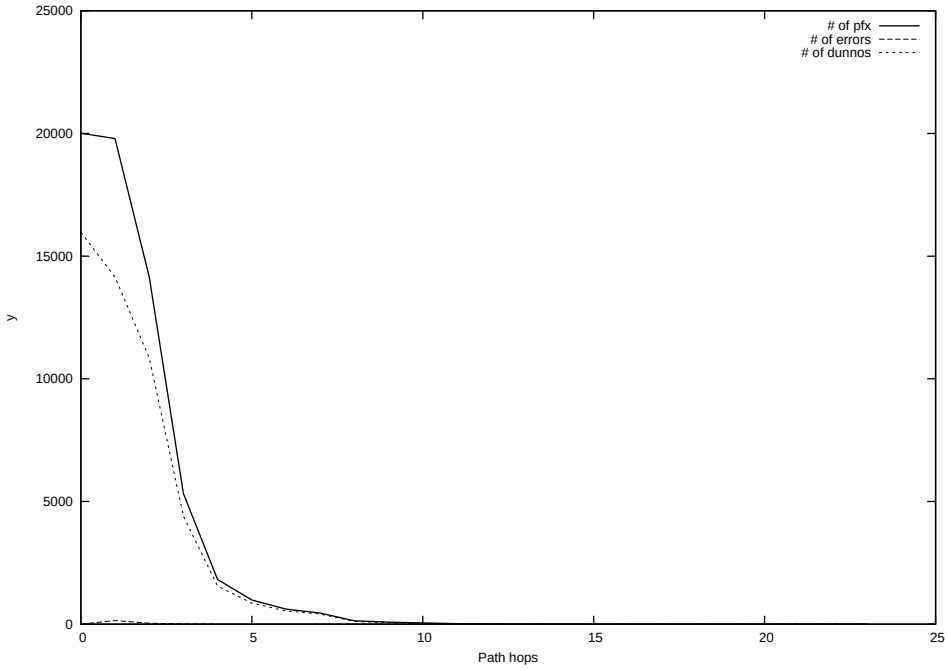
2014-08-21



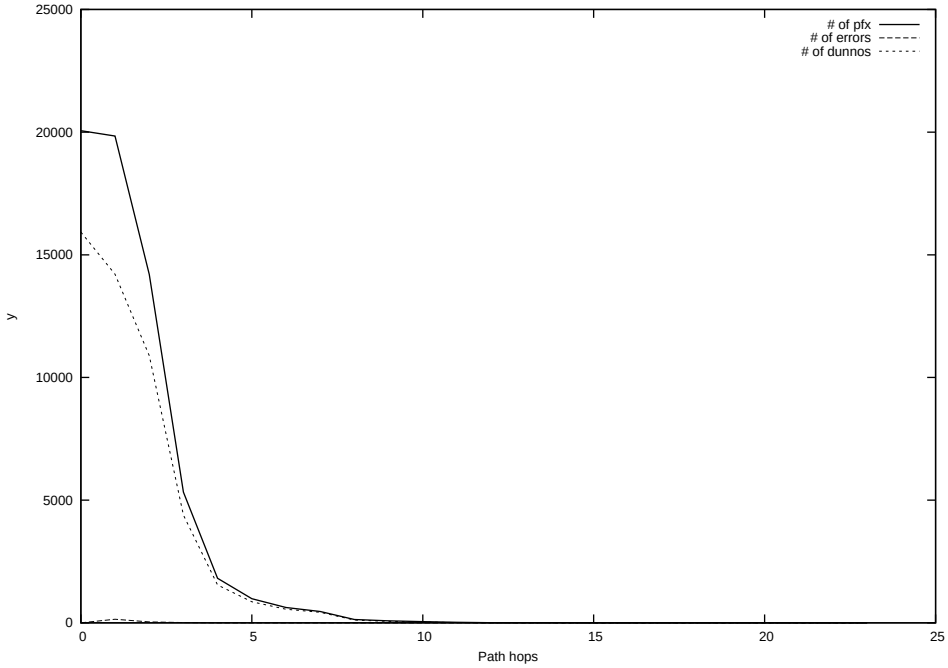
2014-08-22



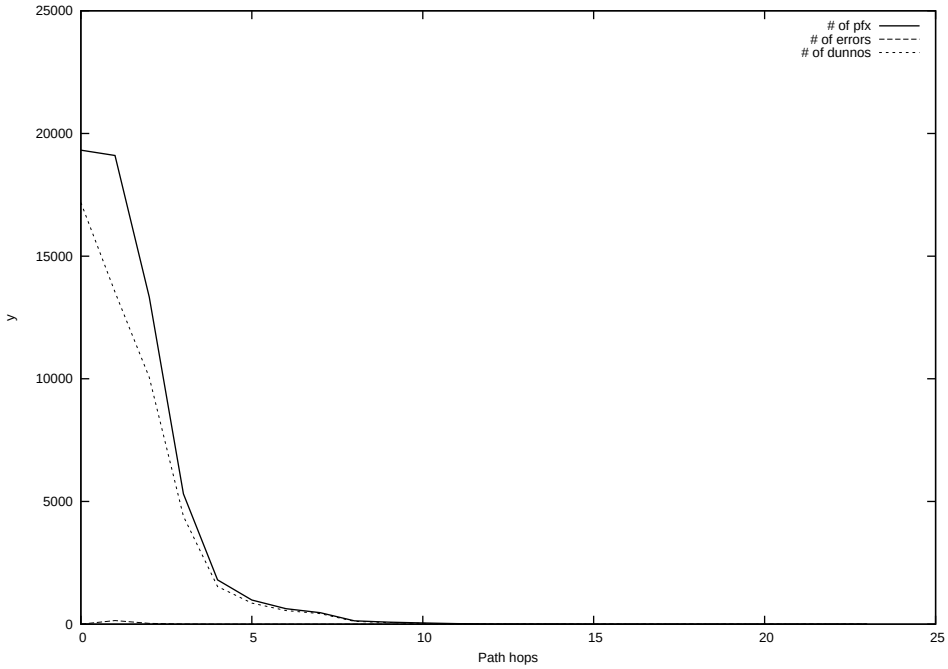
2014-08-23



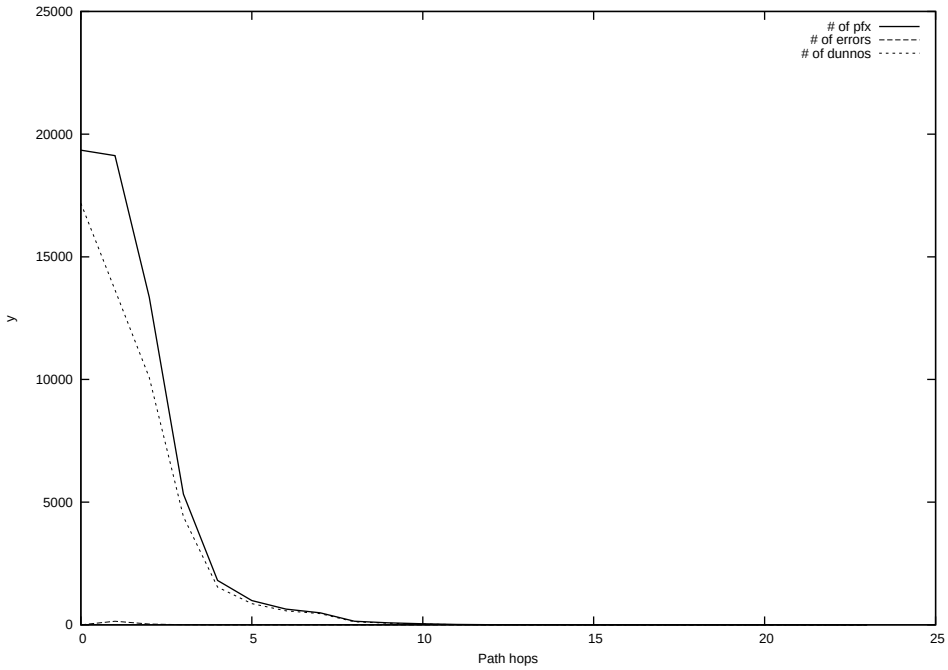
2014-08-24



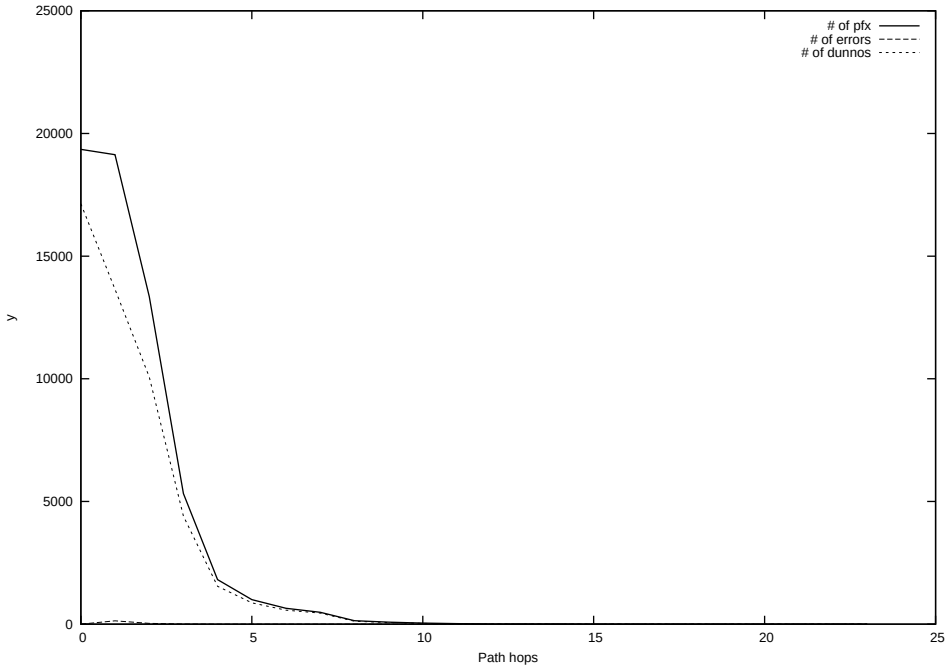
2014-08-25



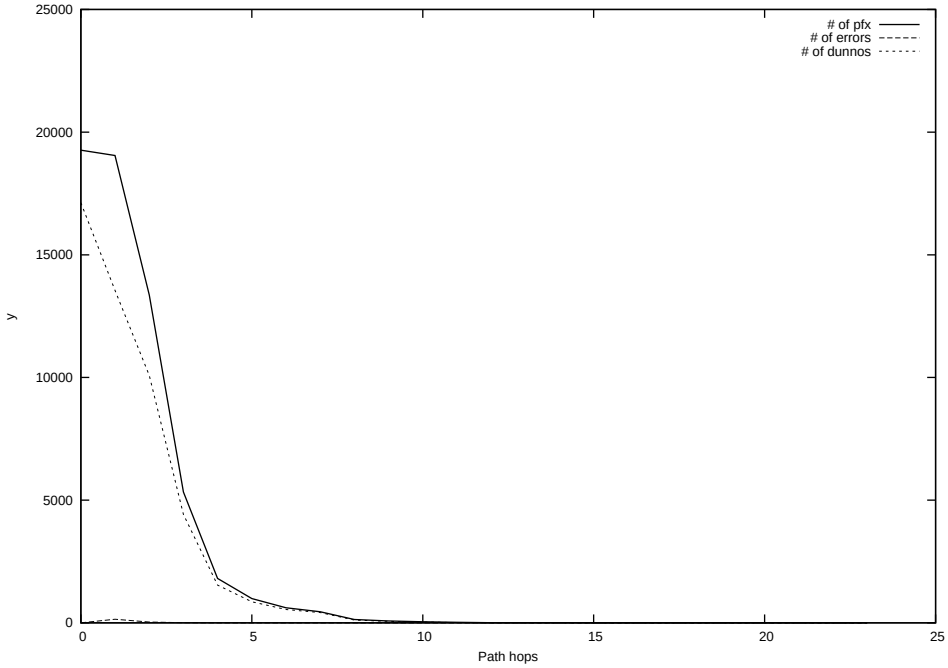
2014-08-26



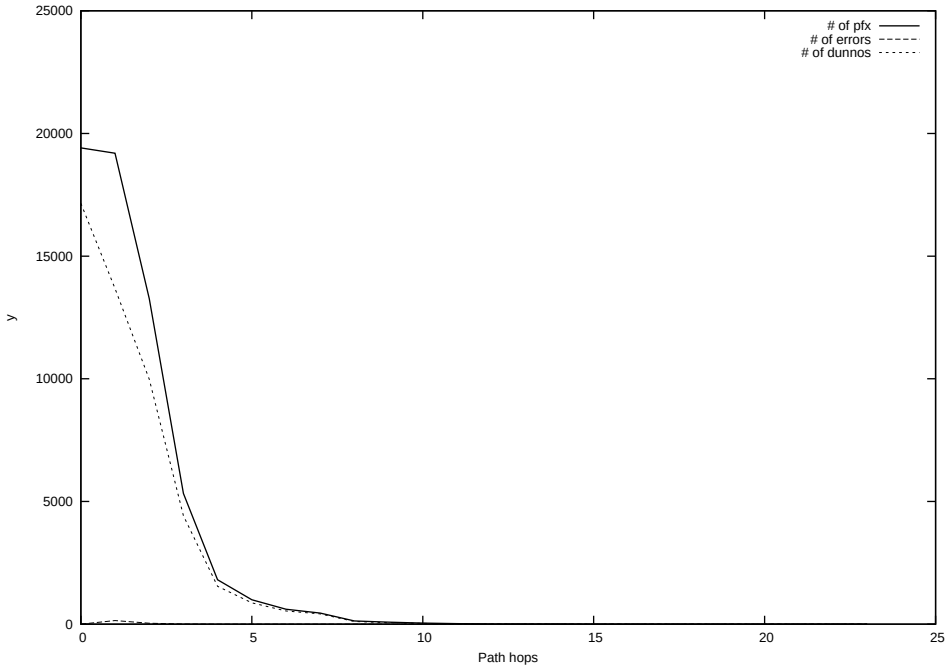
2014-08-27



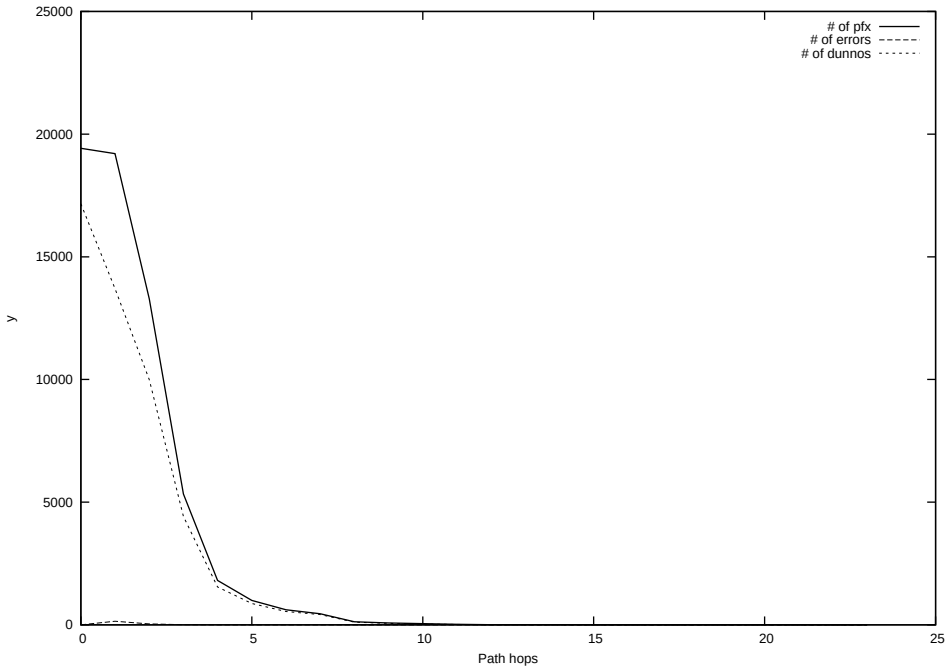
2014-08-28



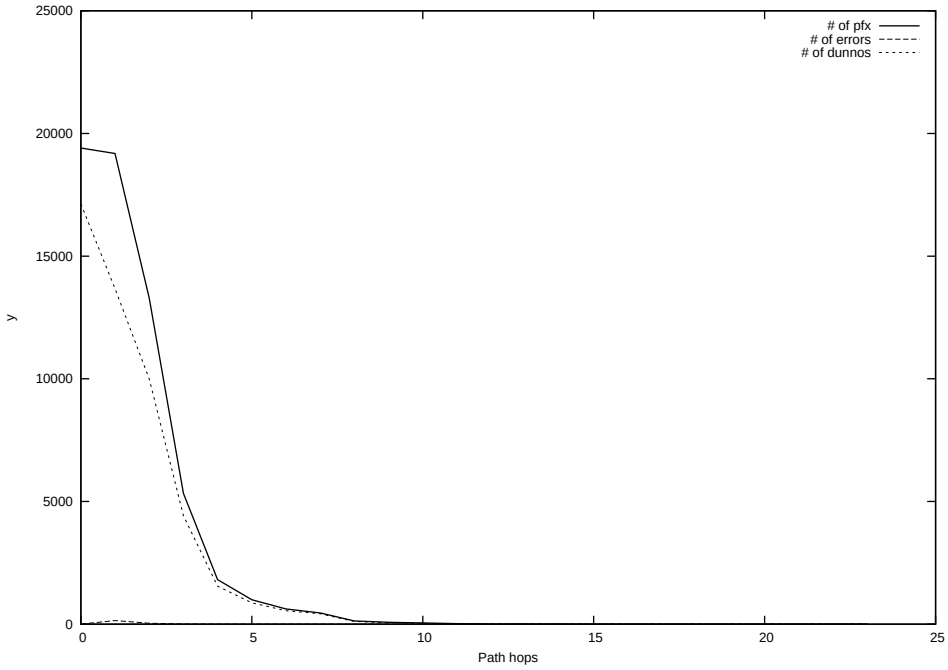
2014-08-29



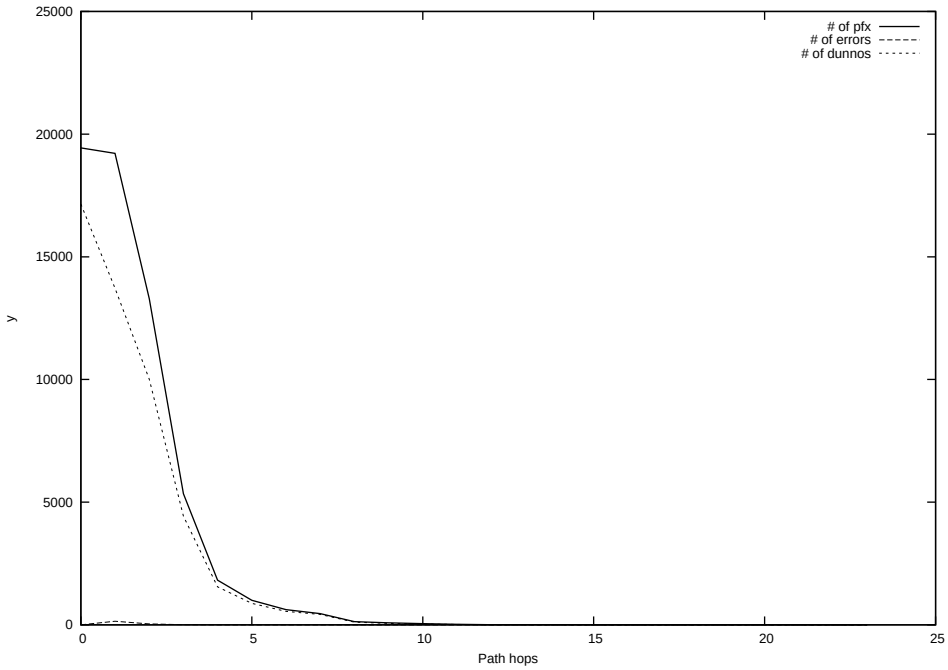
2014-08-30



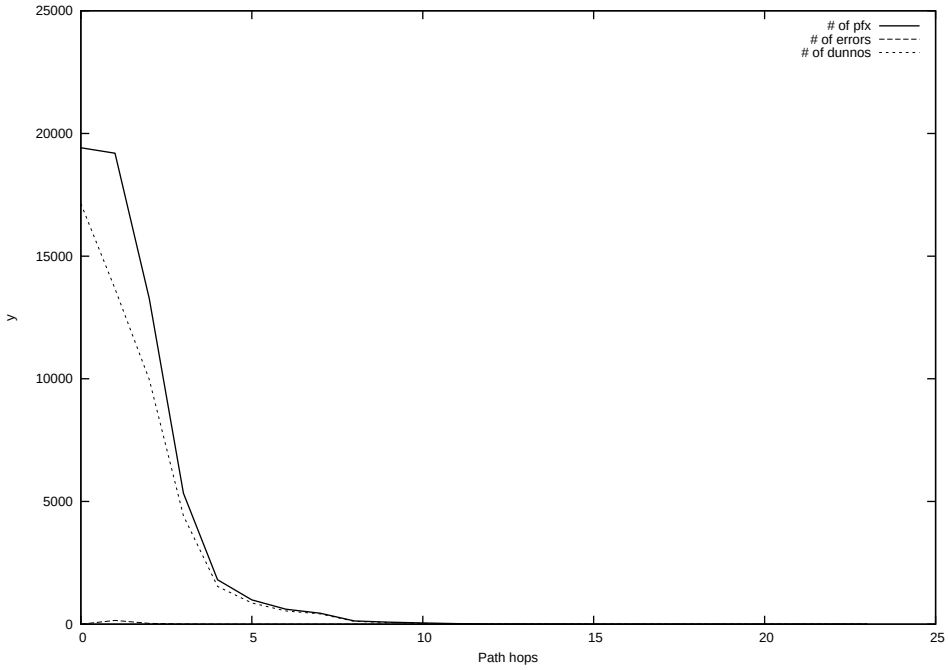
2014-08-31



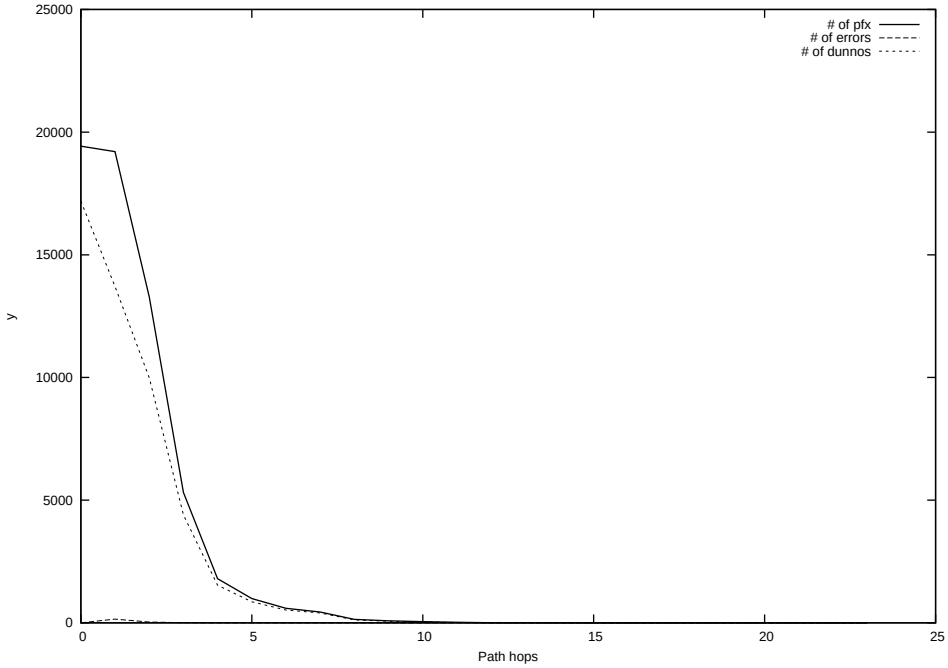
2014-09-01



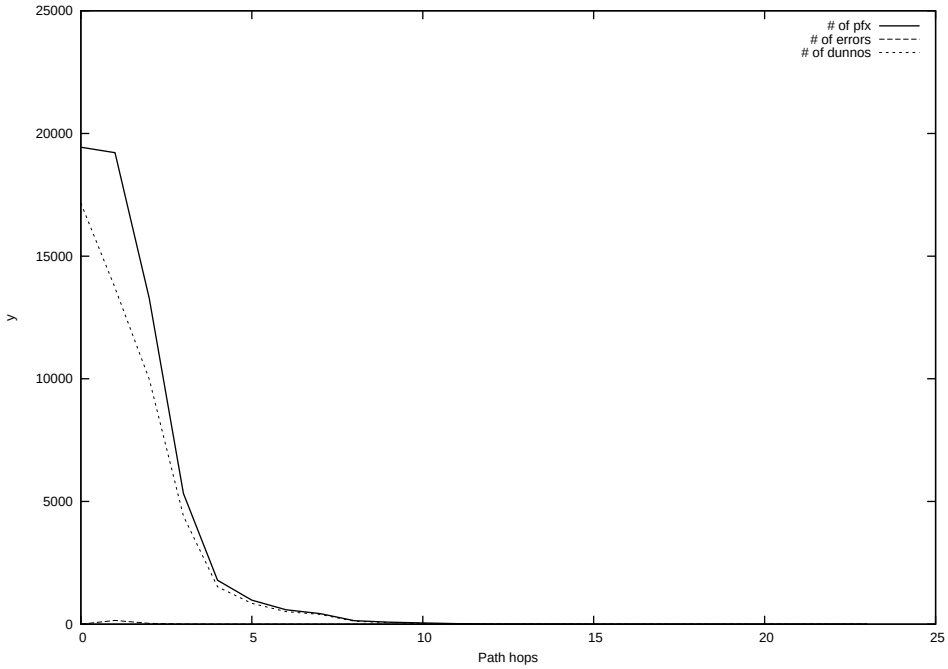
2014-09-02



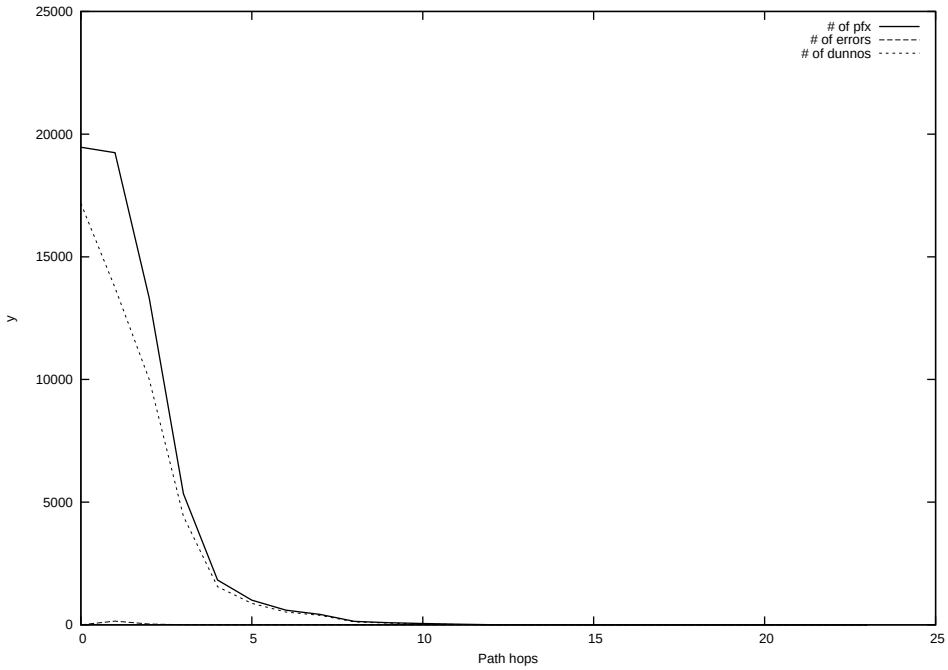
2014-09-03



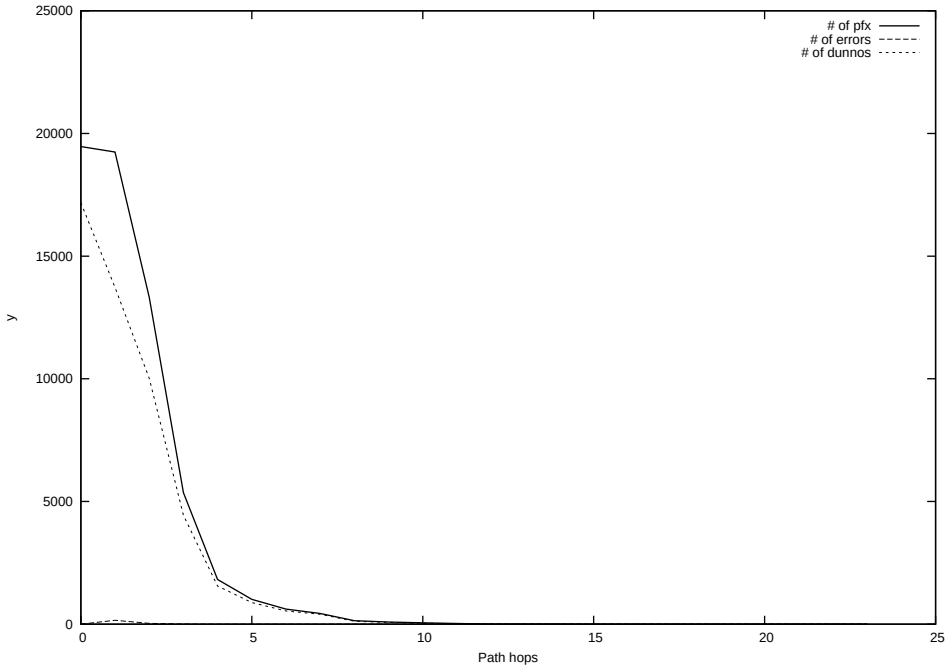
2014-09-04



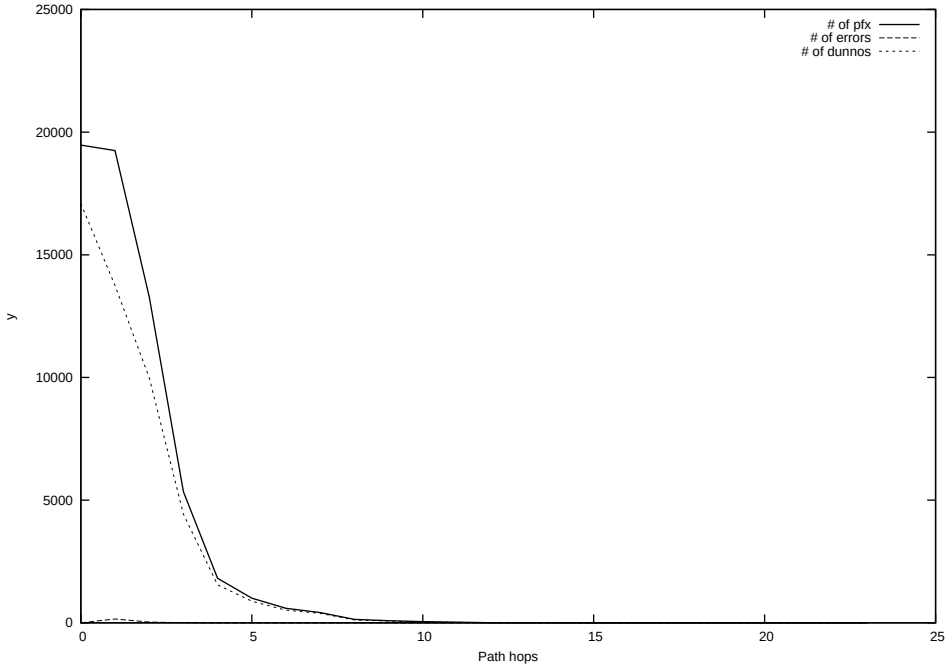
2014-09-05



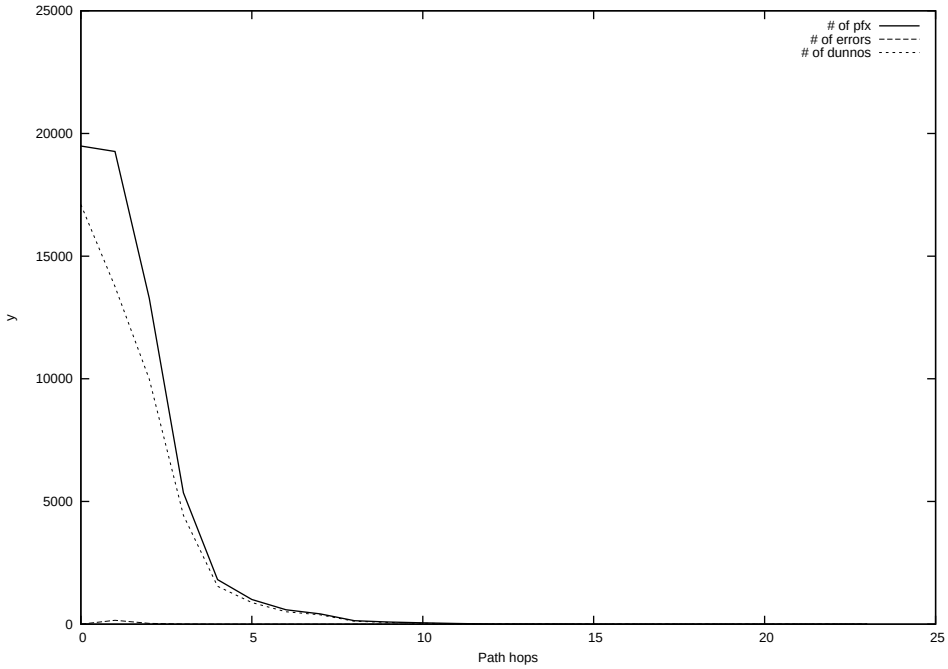
2014-09-06



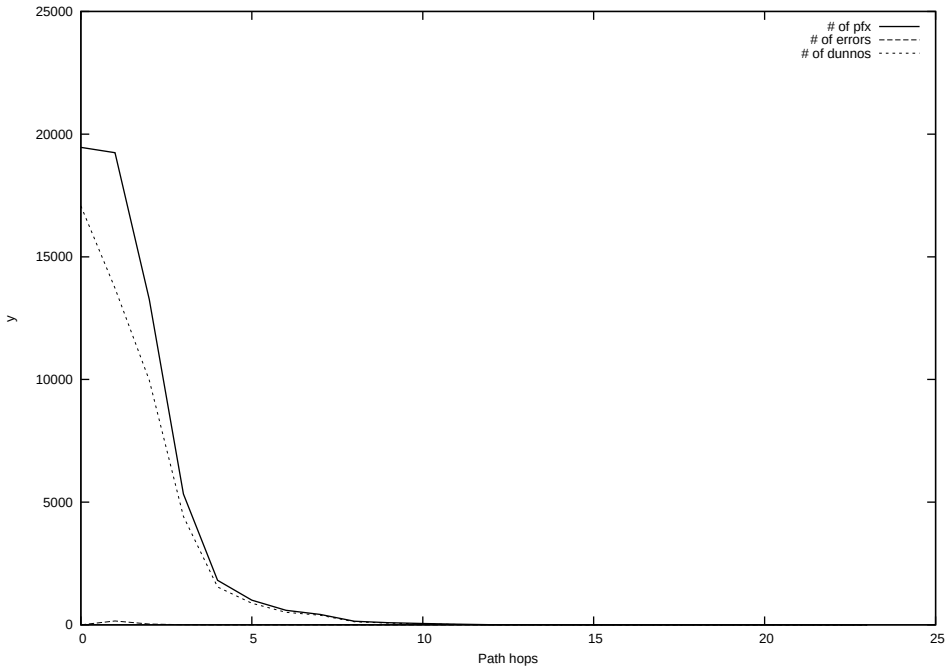
2014-09-07



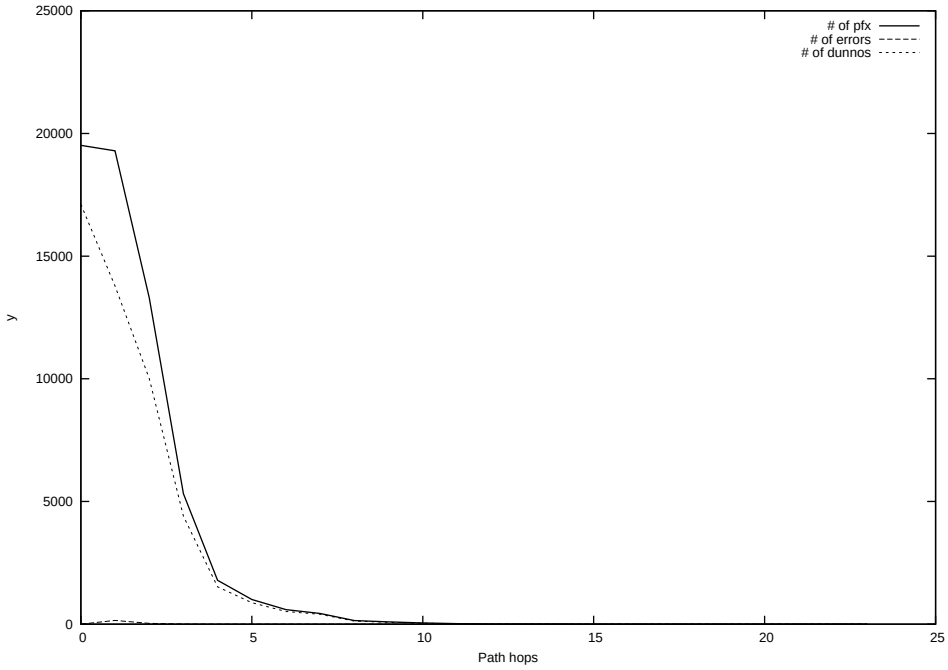
2014-09-08



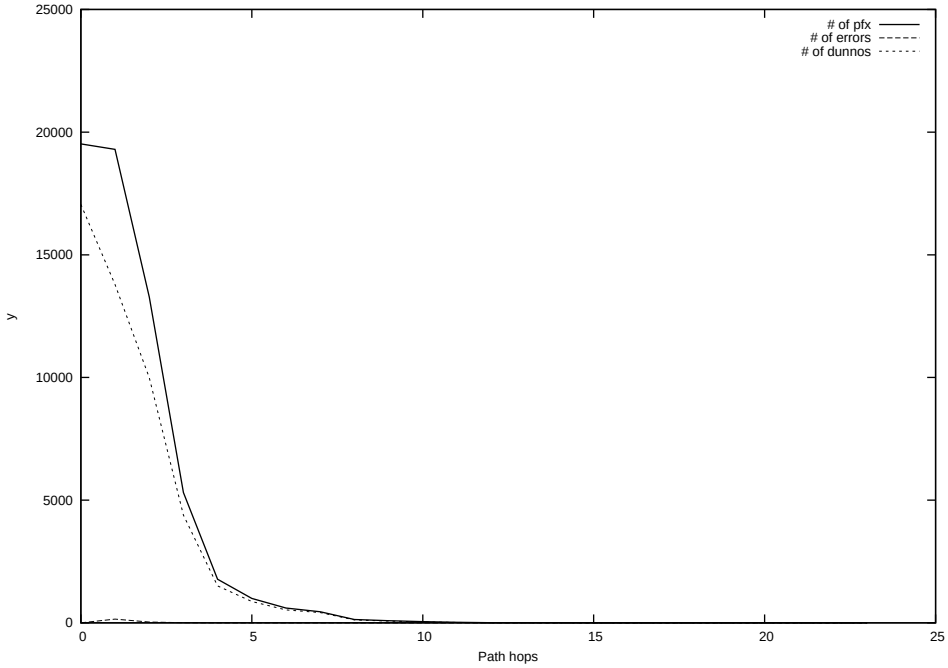
2014-09-09



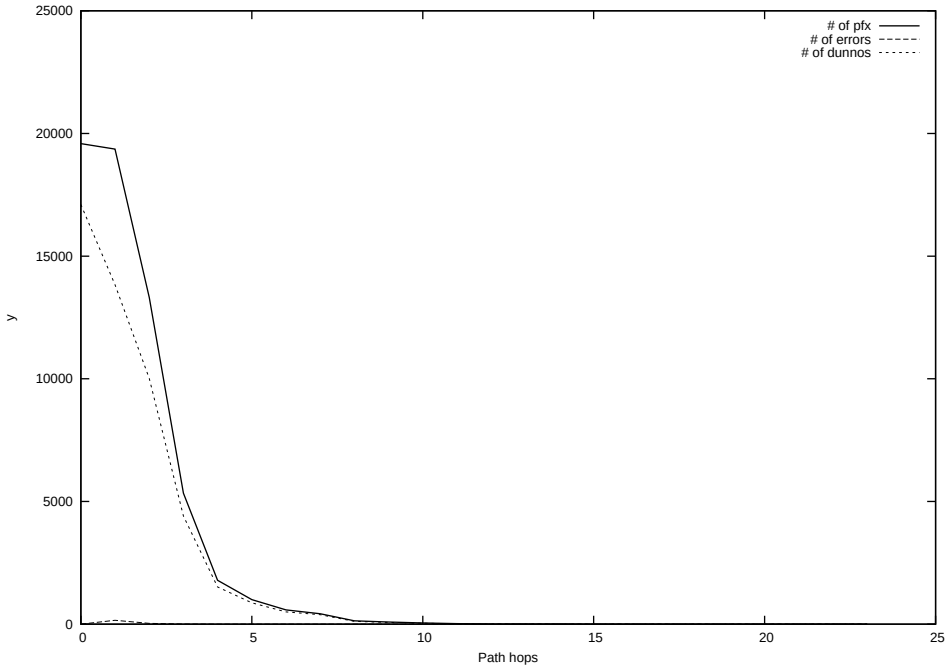
2014-09-10



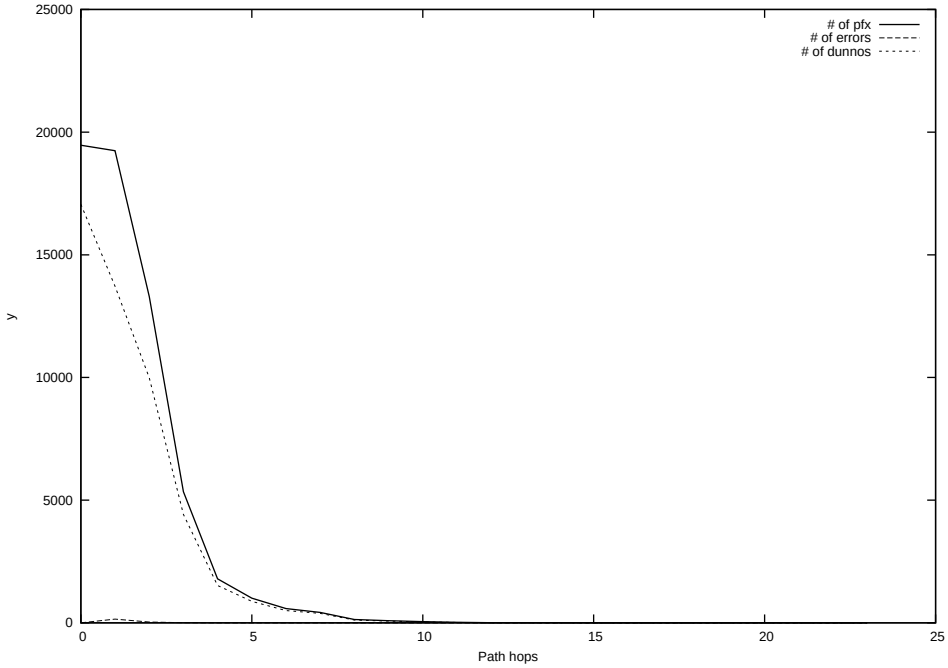
2014-09-11



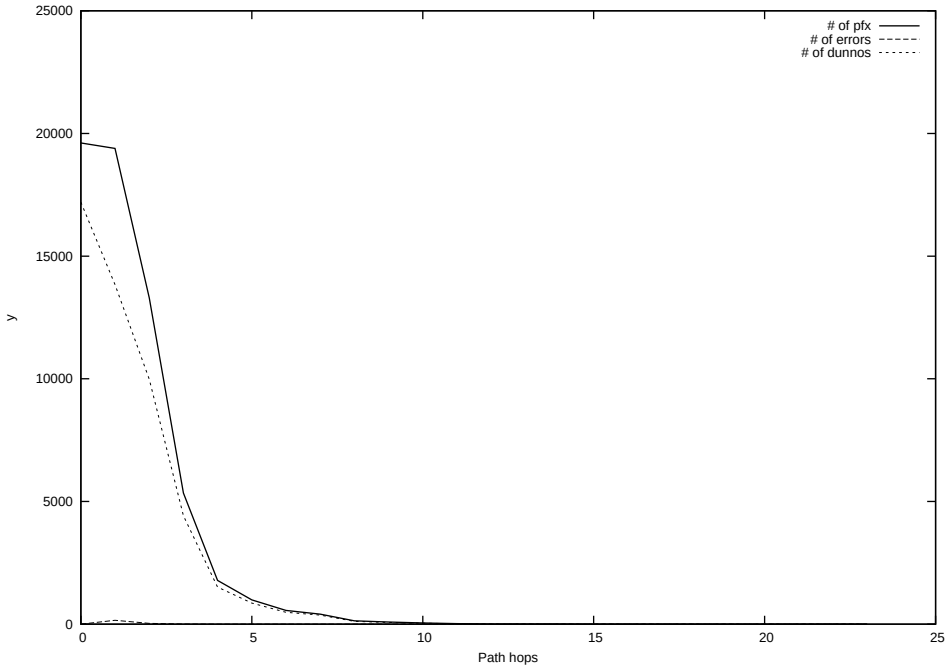
2014-09-12



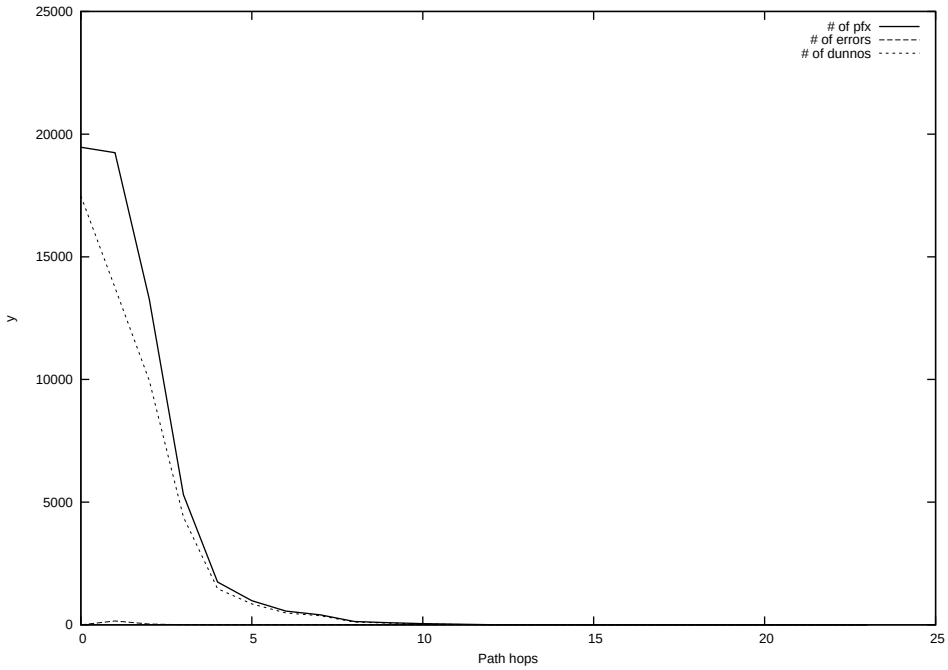
2014-09-13



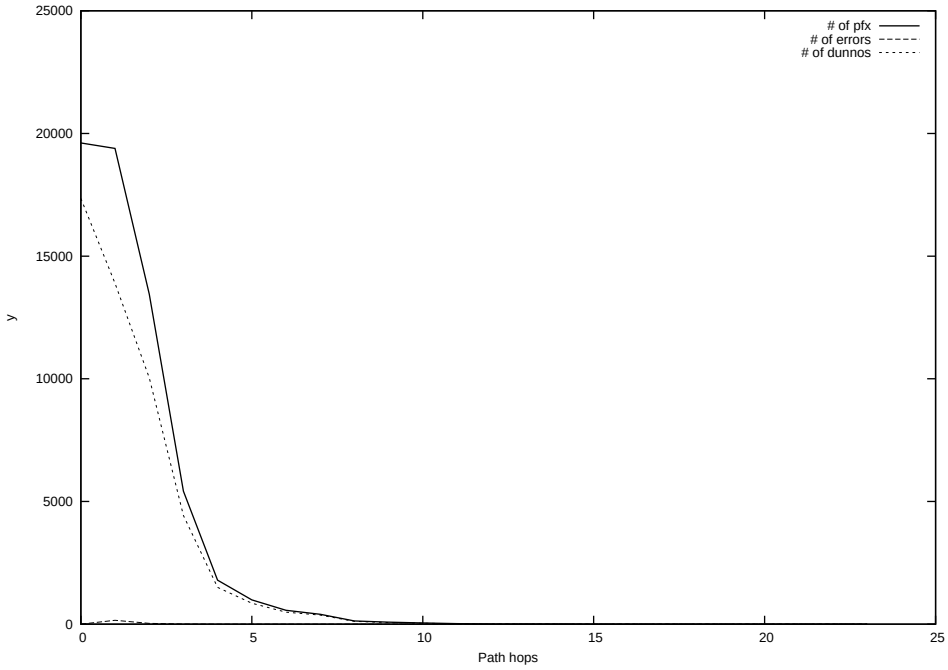
2014-09-14



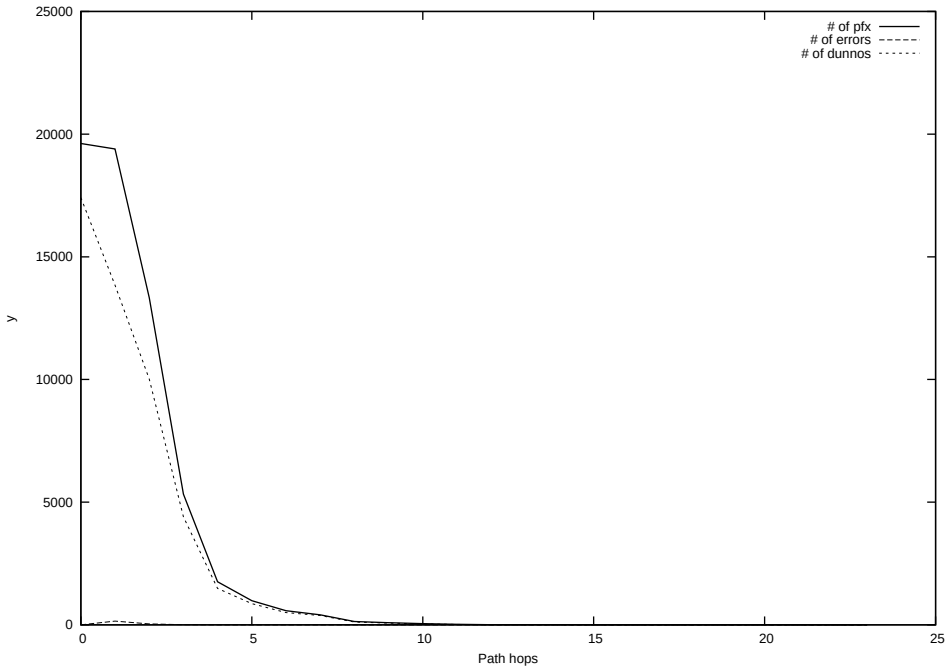
2014-09-15



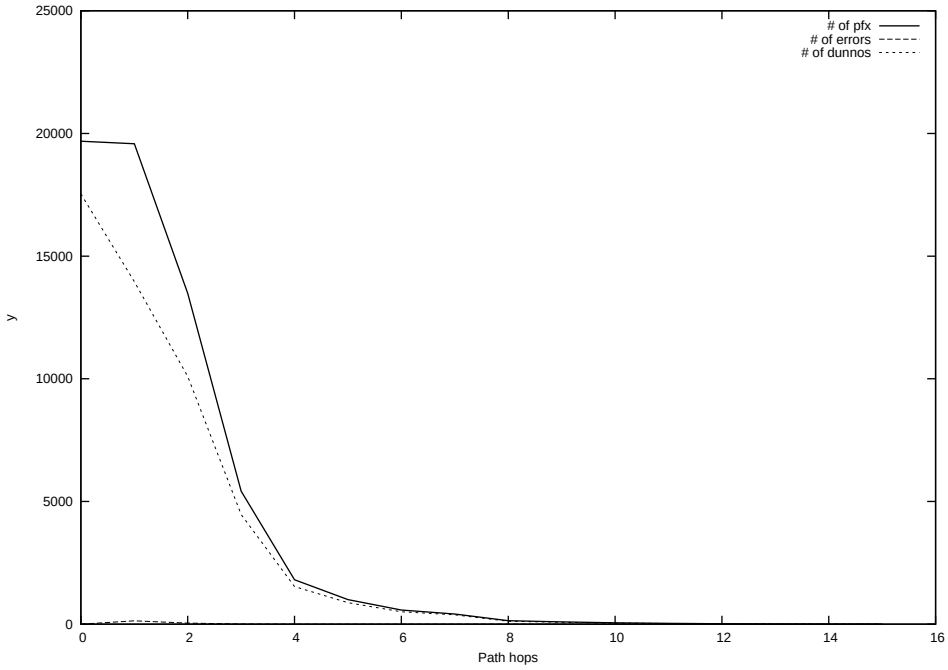
2014-09-16



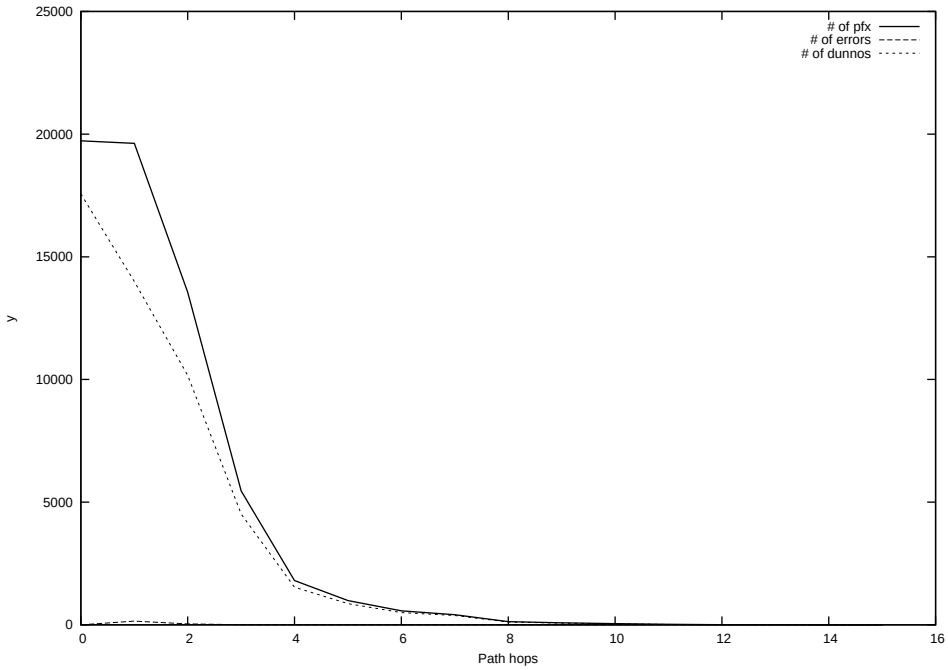
2014-09-17



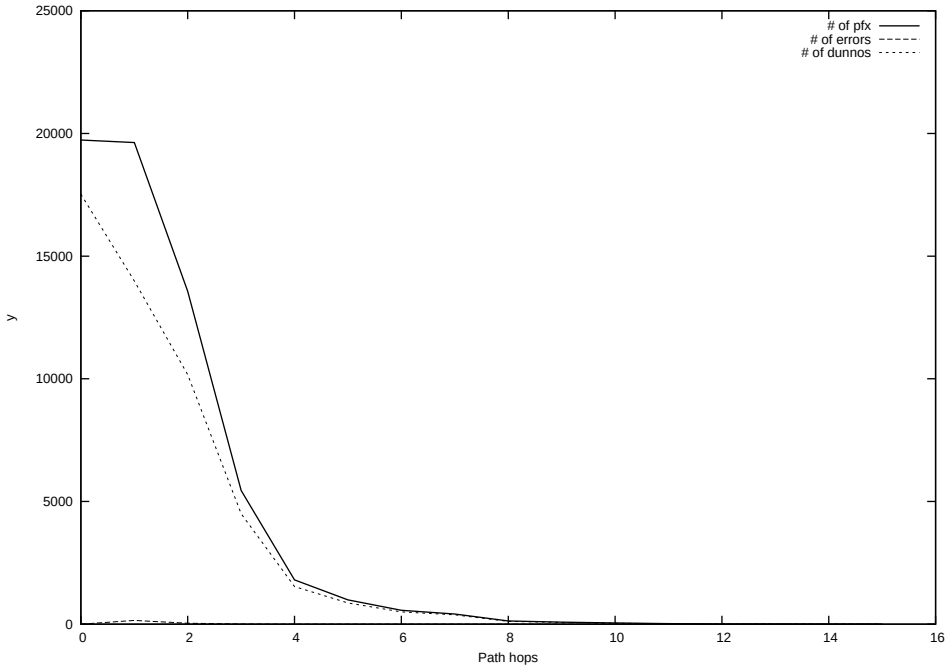
2014-09-18



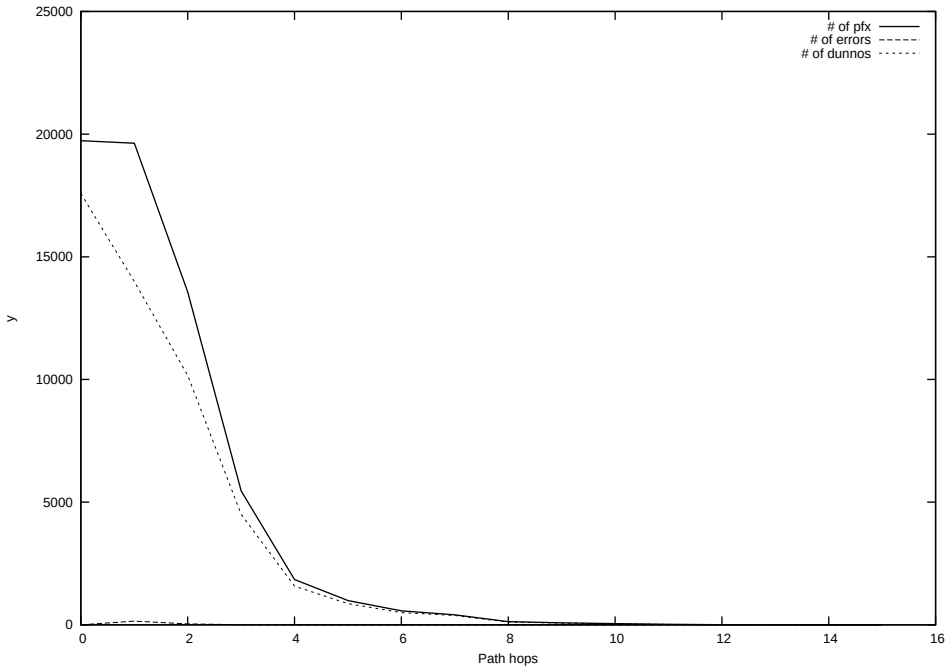
2014-09-19



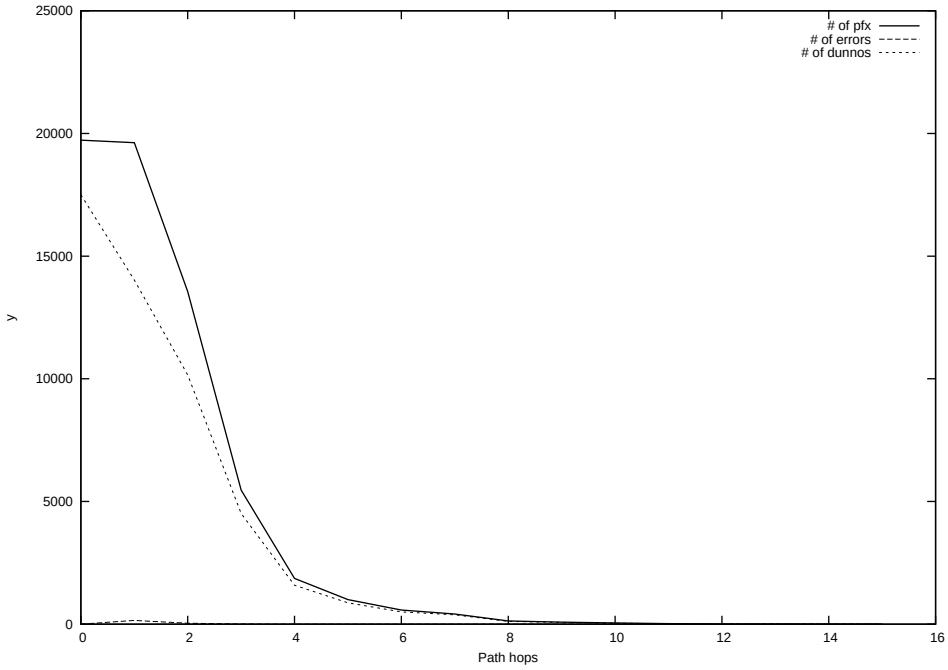
2014-09-20



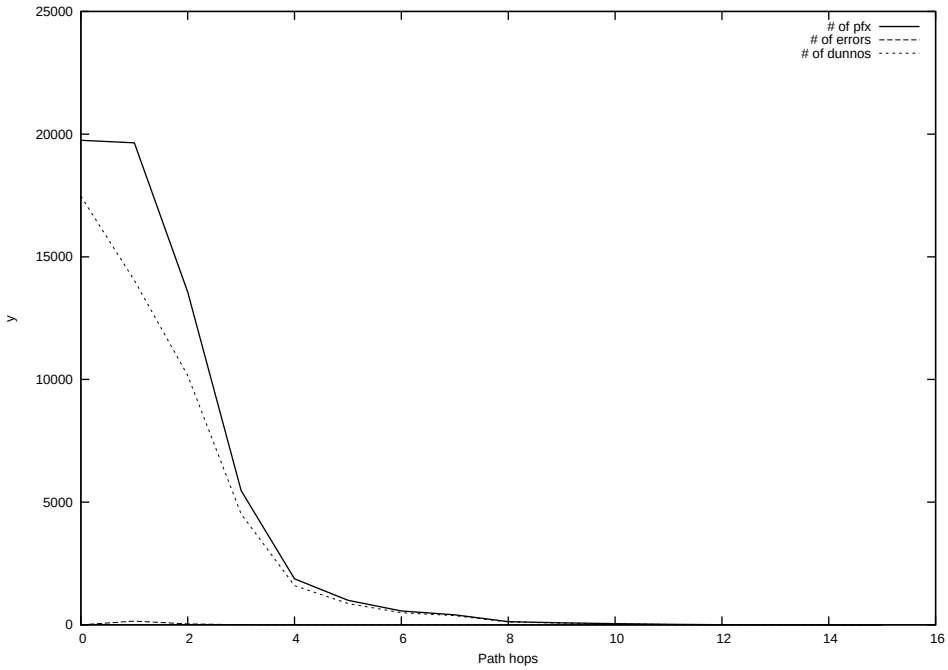
2014-09-21



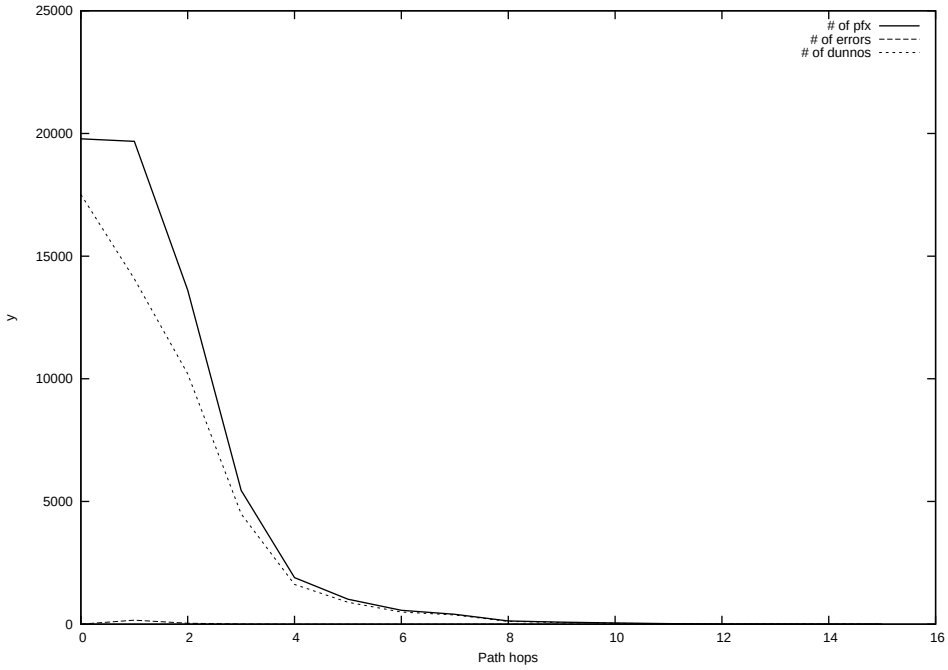
2014-09-22



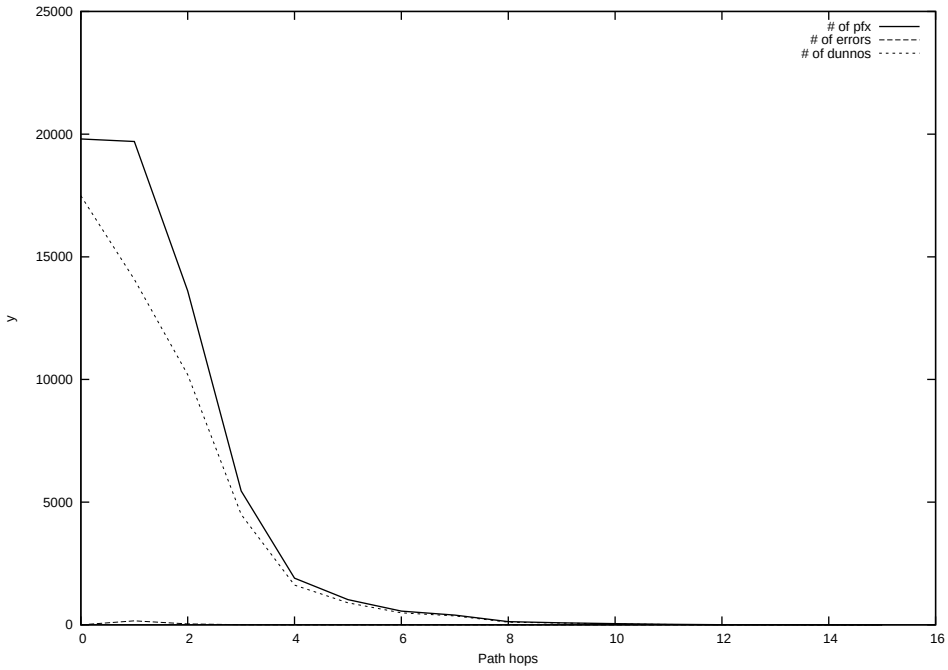
2014-09-23



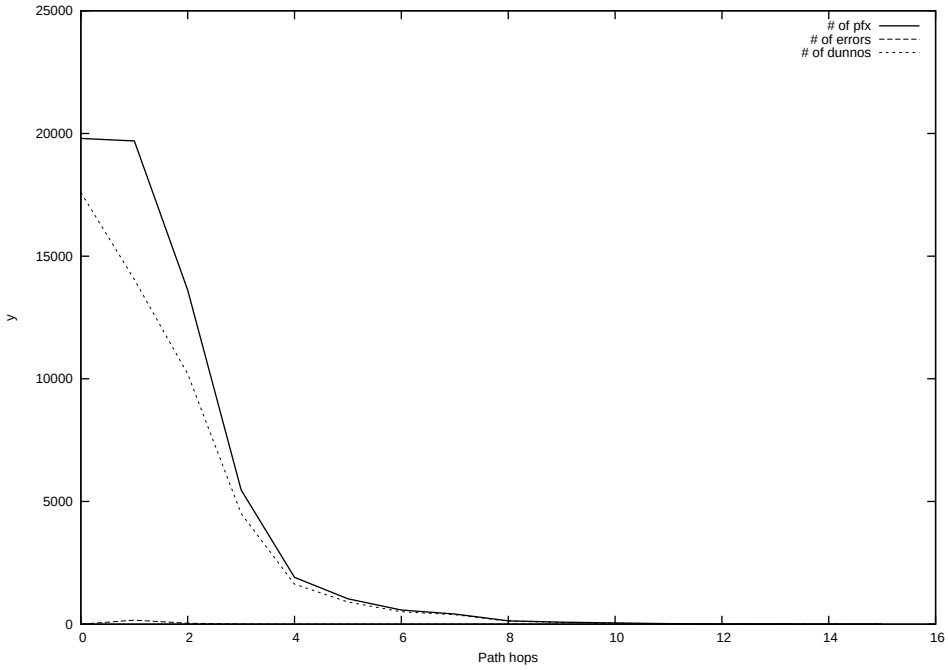
2014-09-24



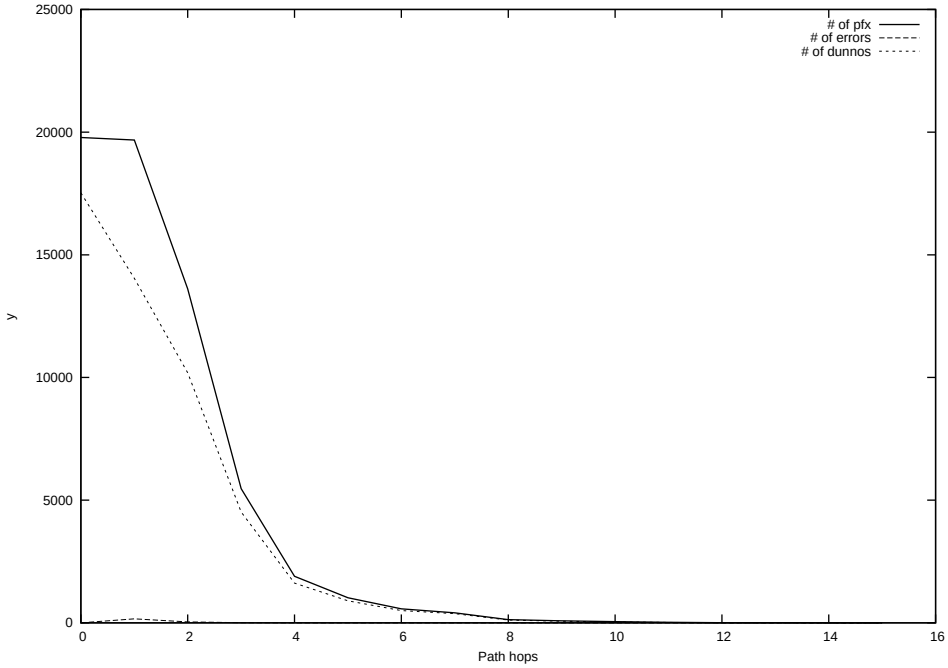
2014-09-25



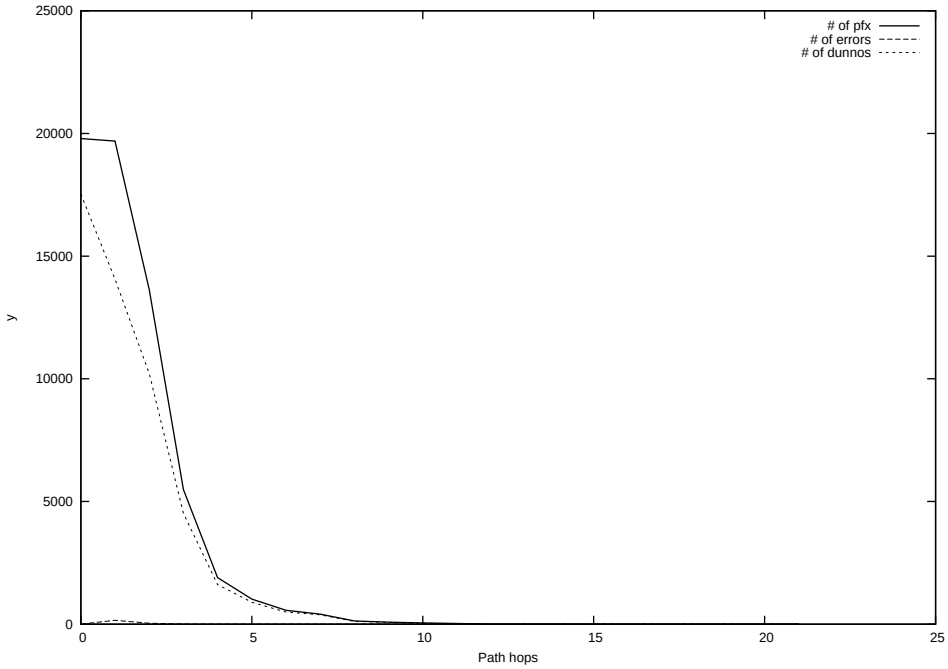
2014-09-26



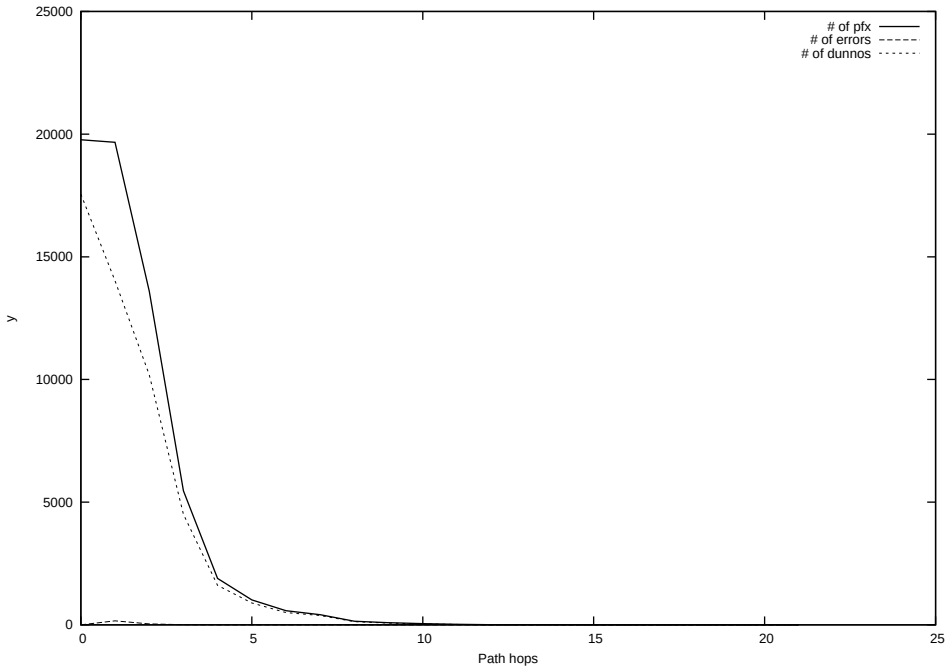
2014-09-27



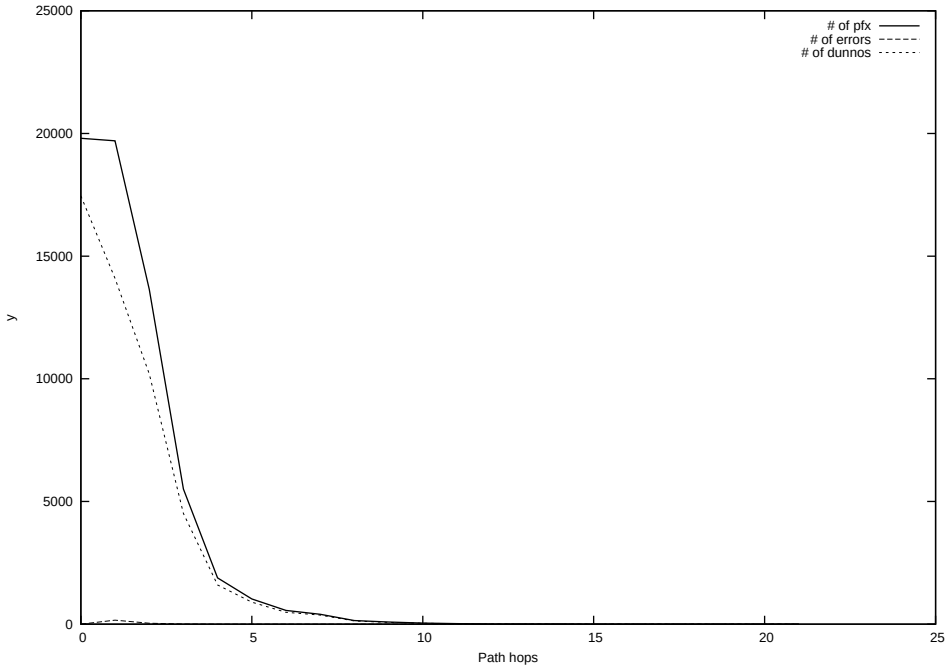
2014-09-28



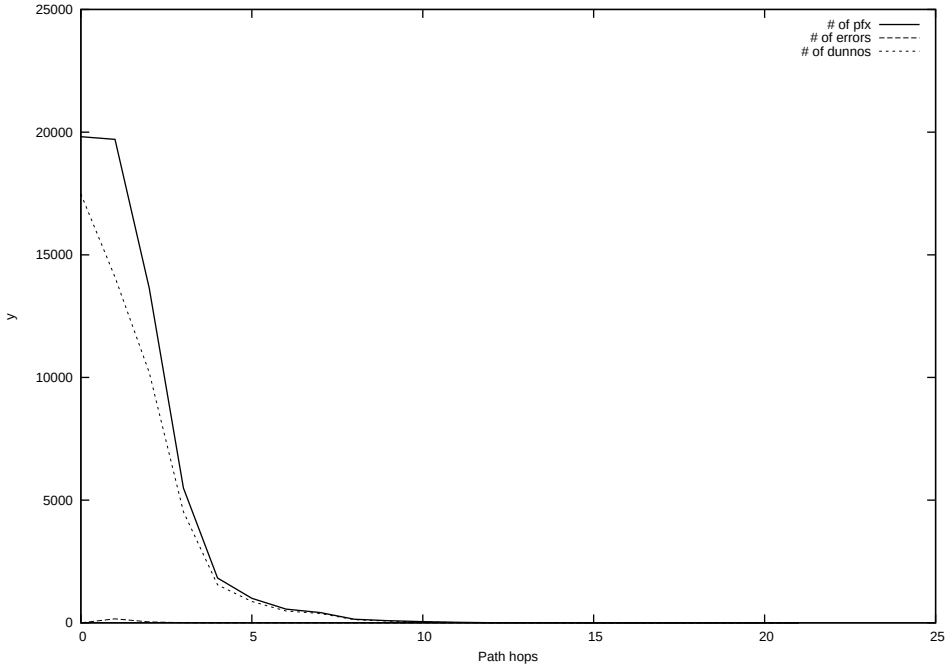
2014-09-29



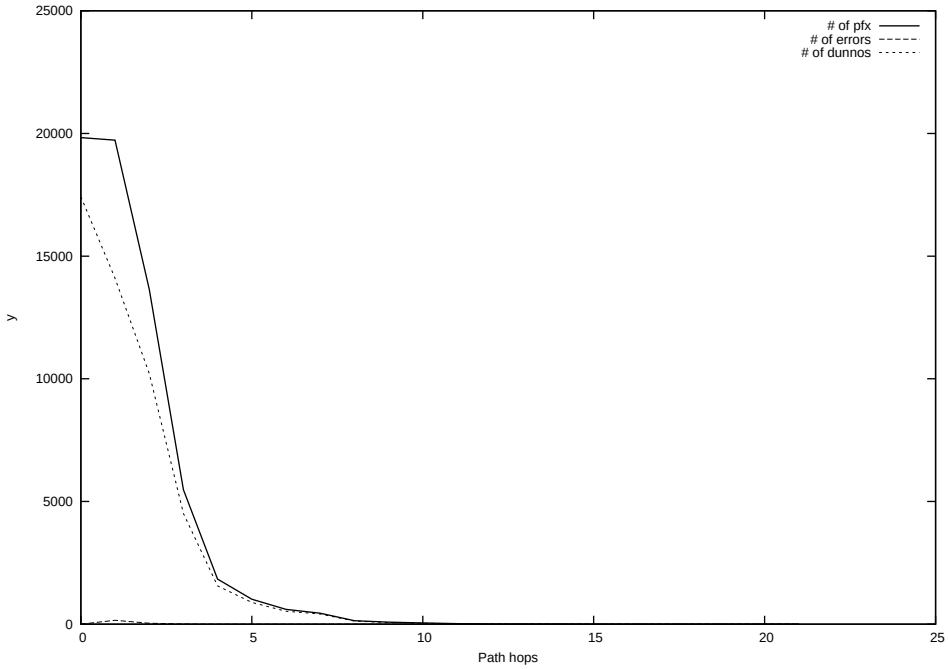
2014-09-30



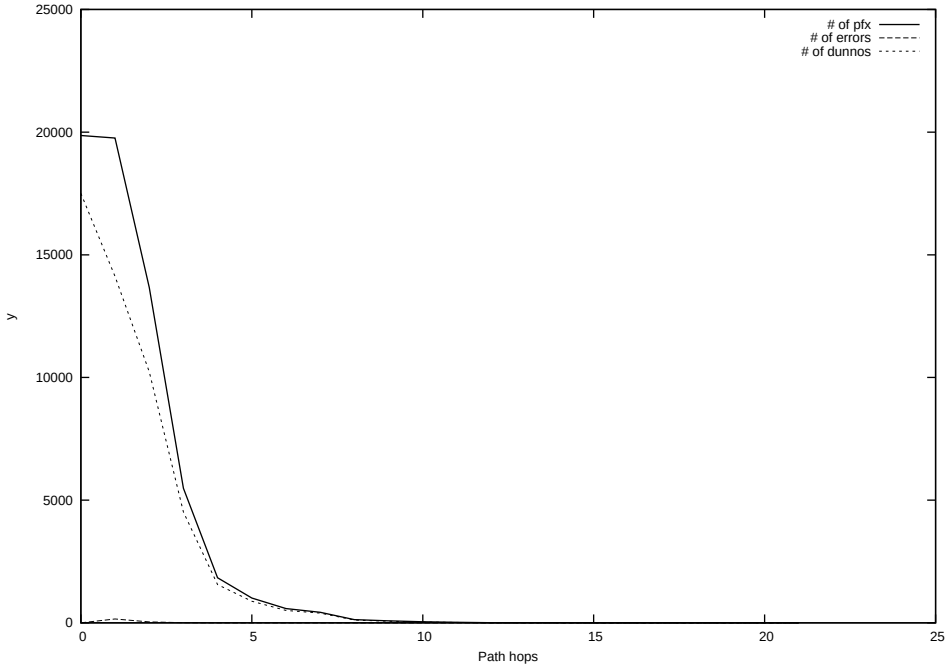
2014-10-01



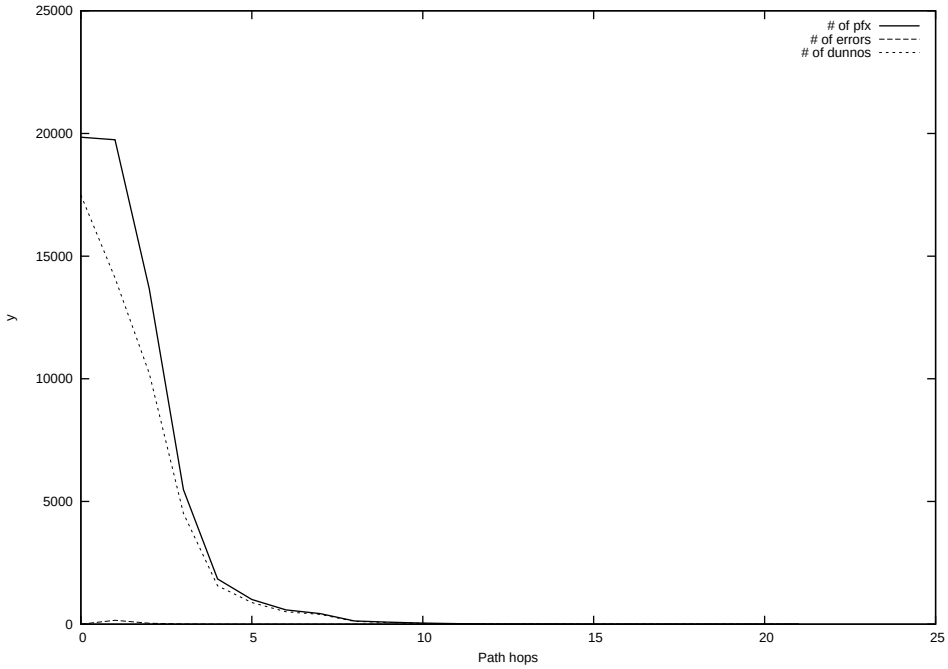
2014-10-02



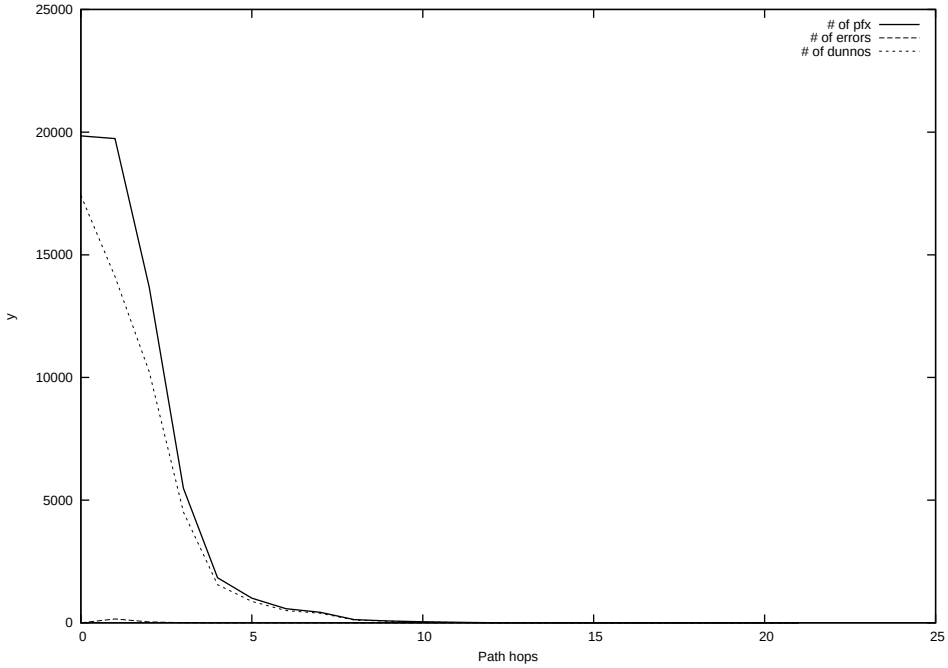
2014-10-03



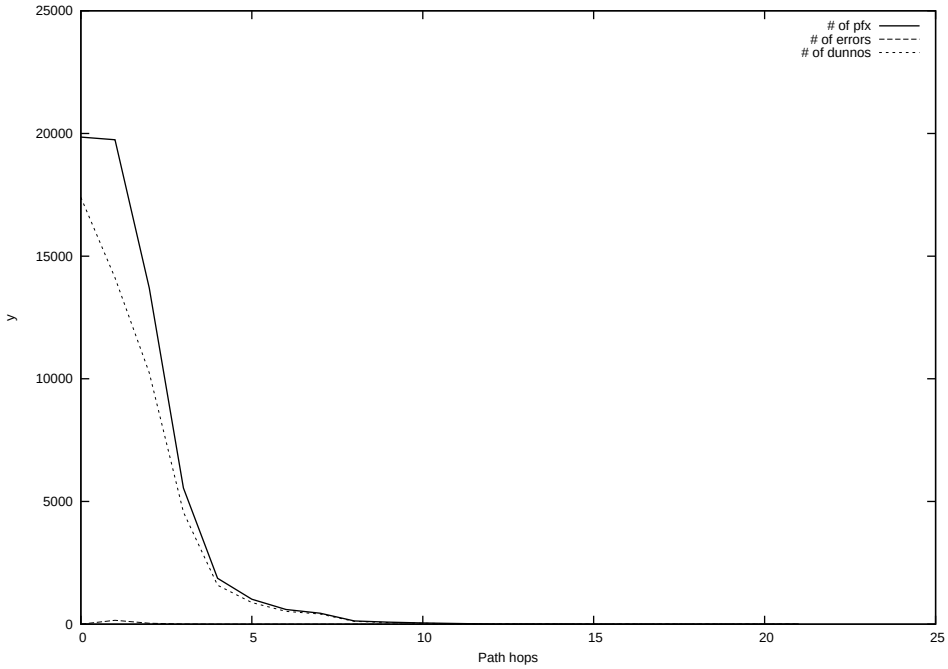
2014-10-04



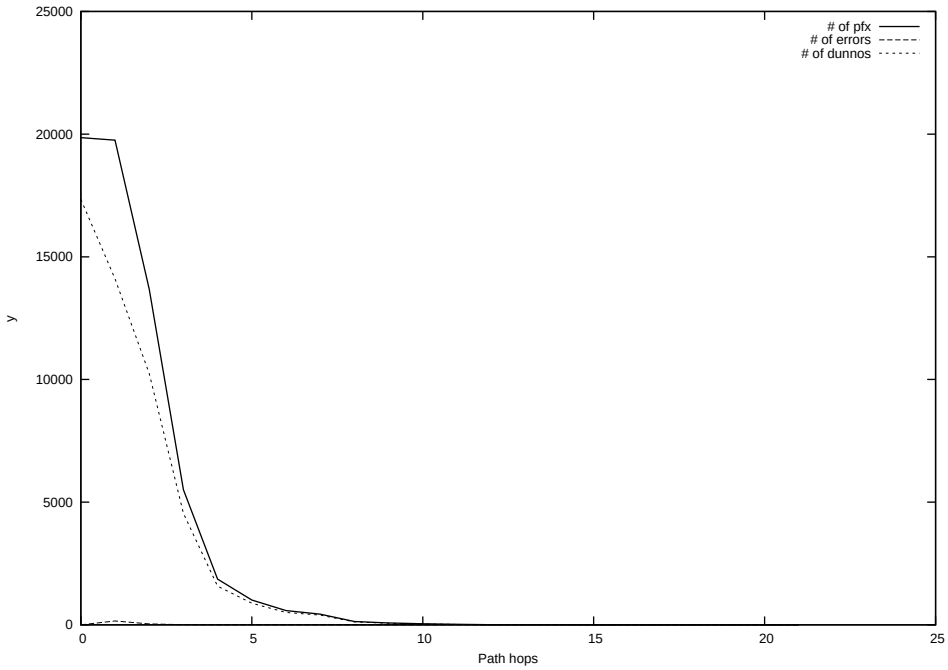
2014-10-05



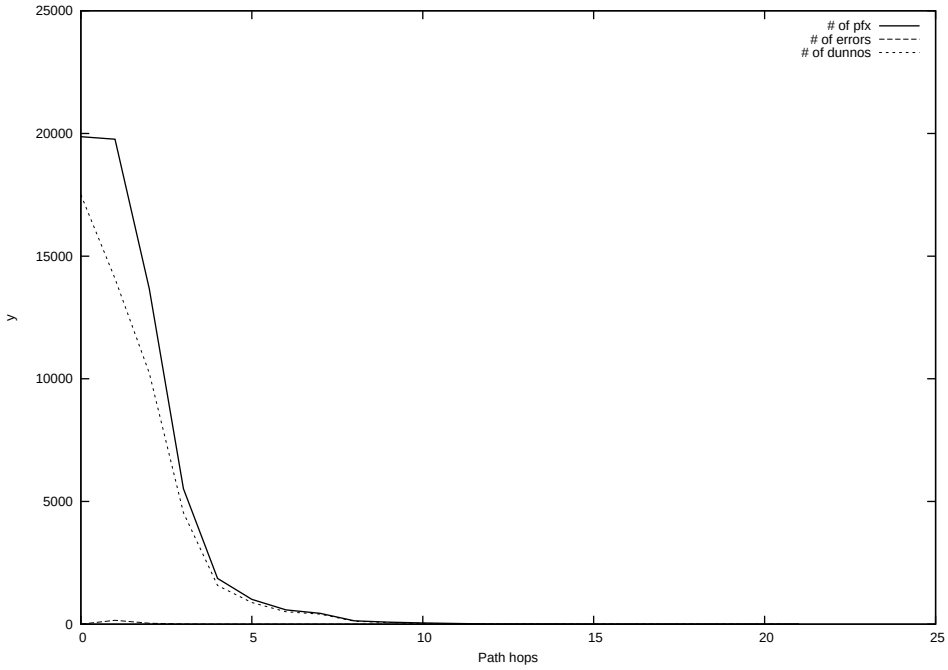
2014-10-06



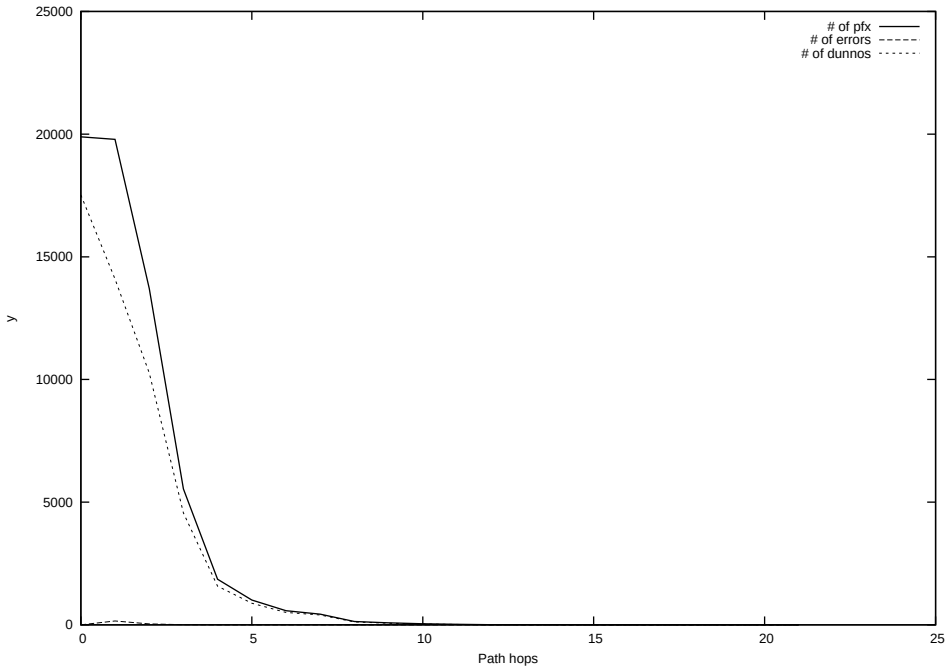
2014-10-07



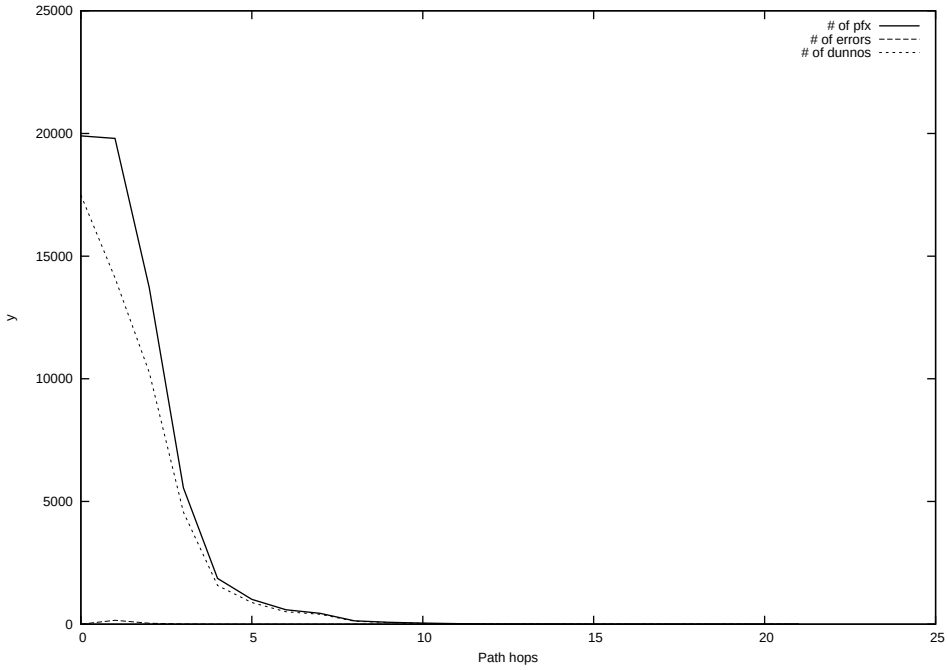
2014-10-08



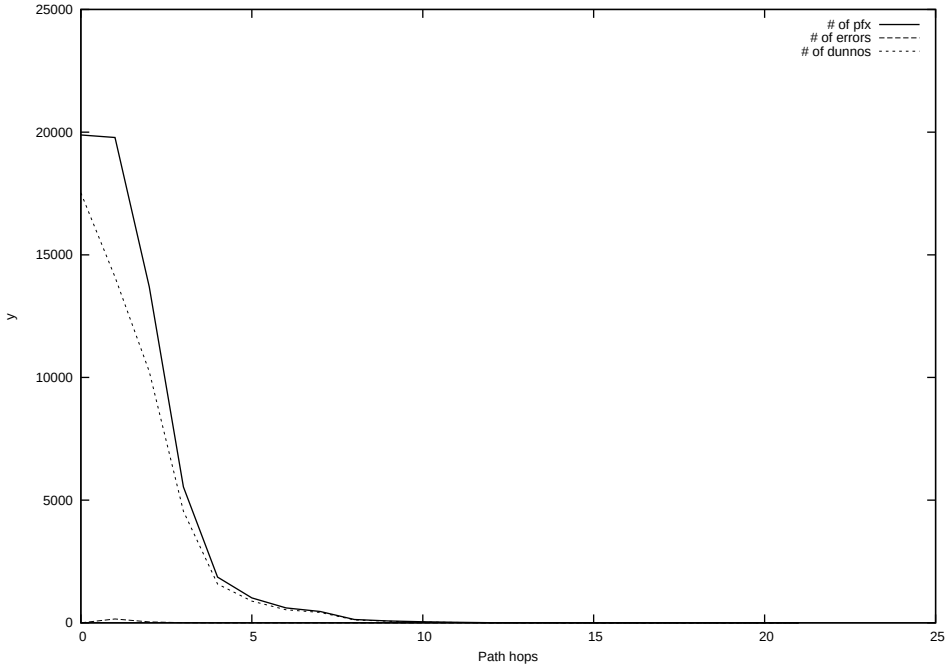
2014-10-09



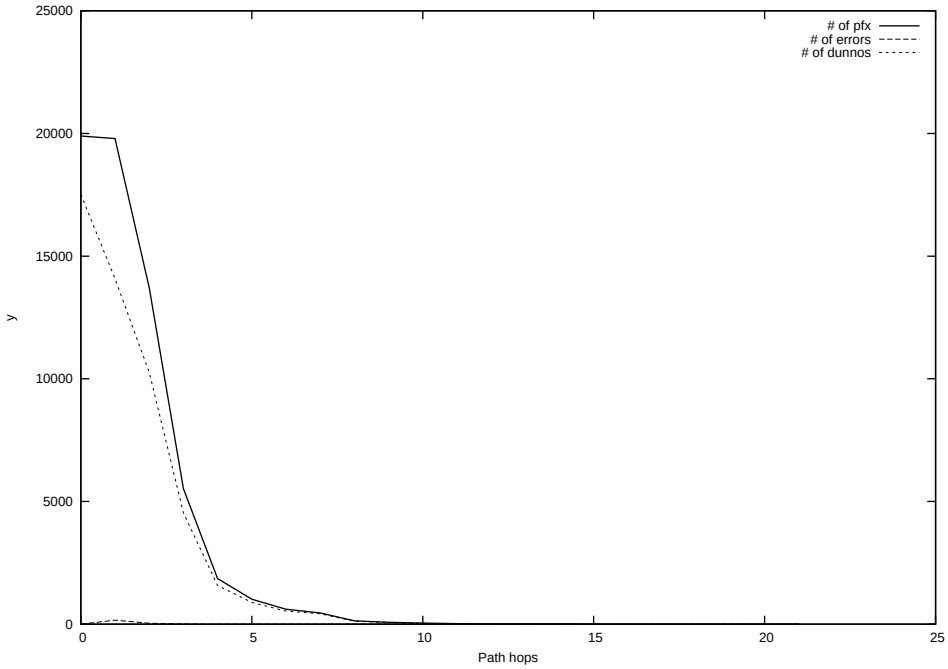
2014-10-10



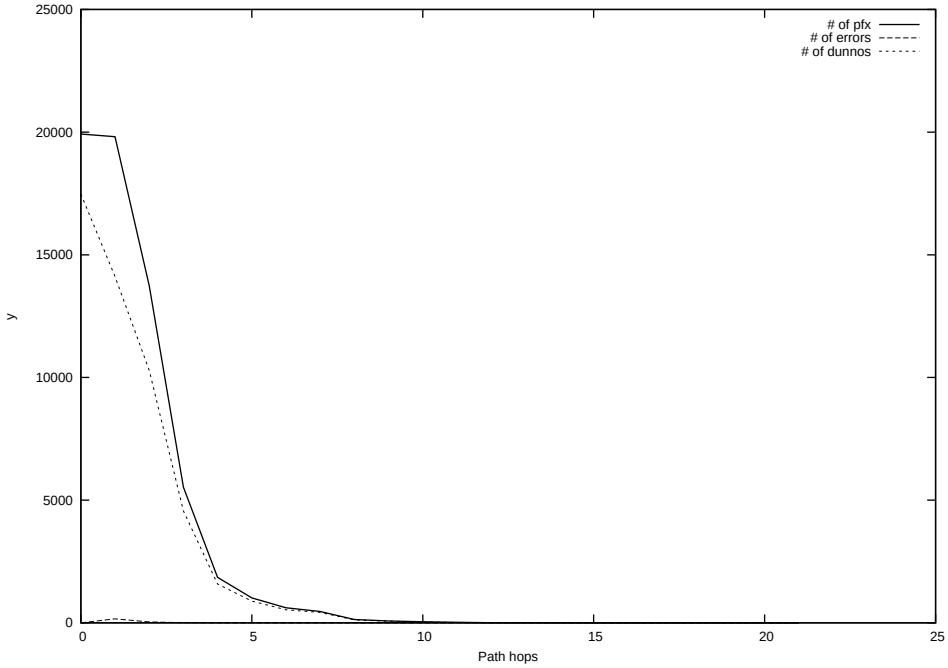
2014-10-11



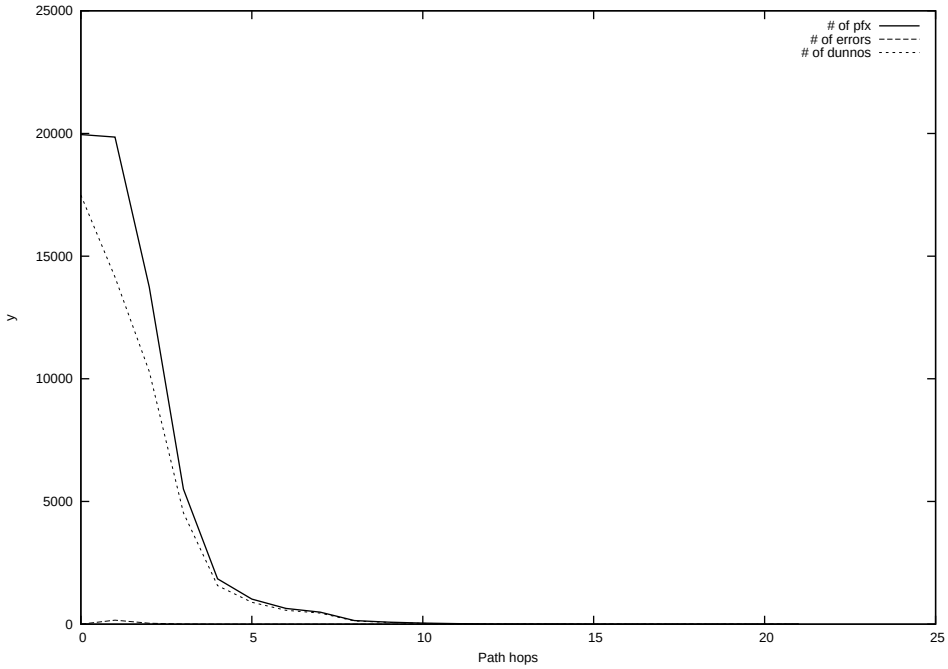
2014-10-12



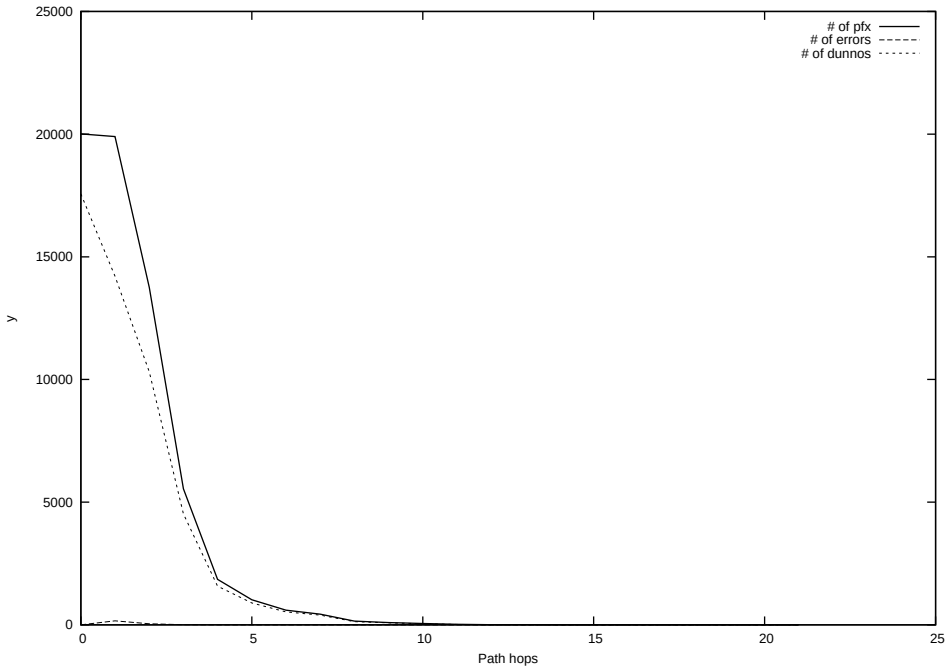
2014-10-13



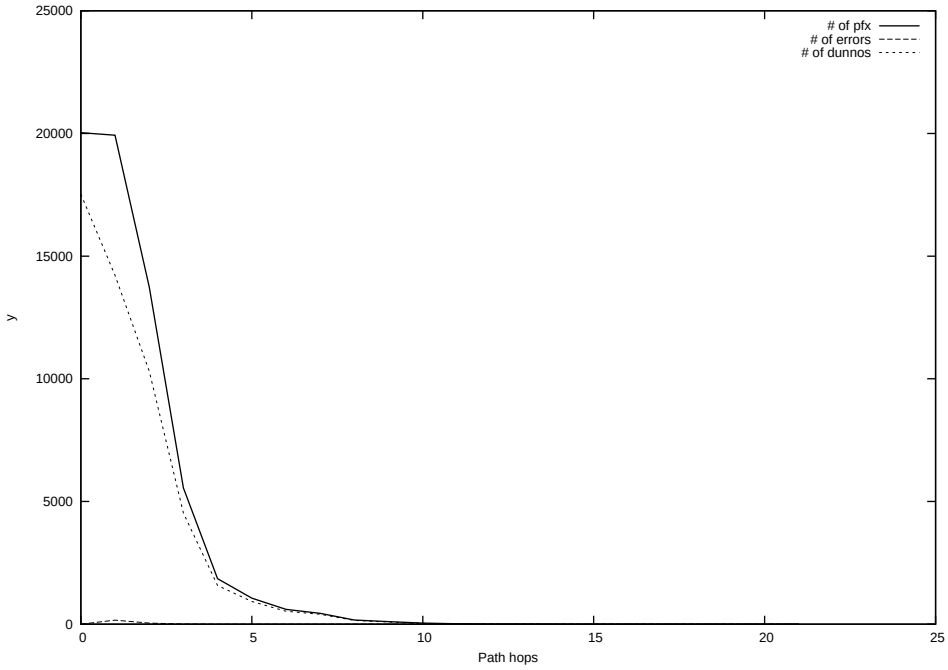
2014-10-14



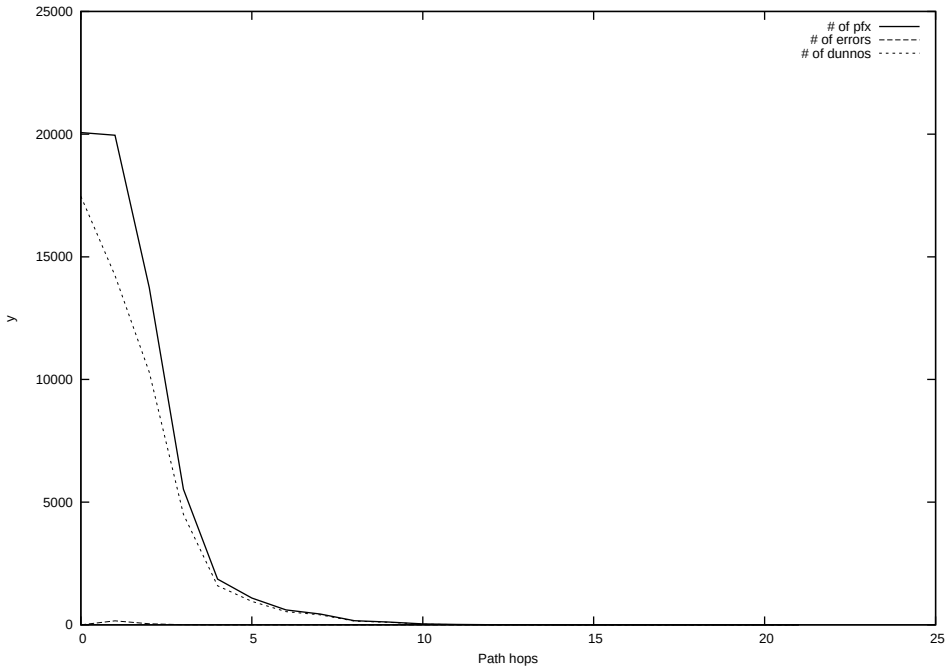
2014-10-15



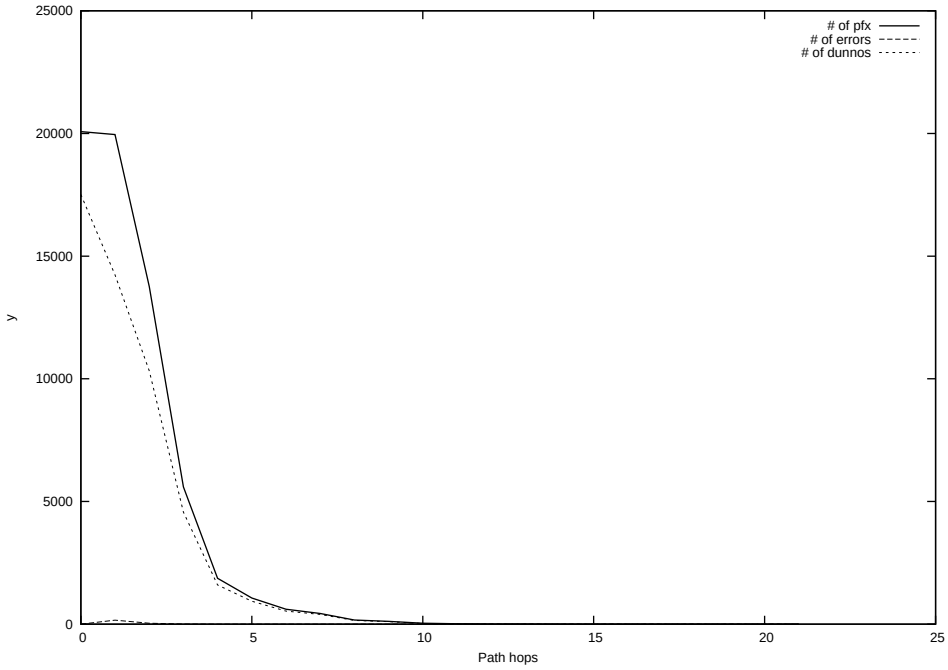
2014-10-16



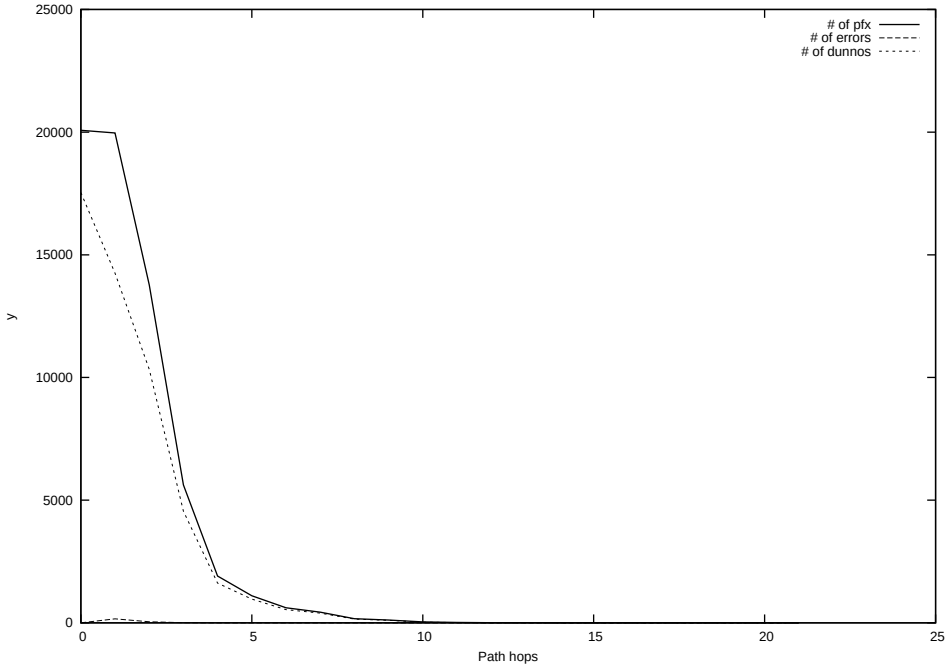
2014-10-17



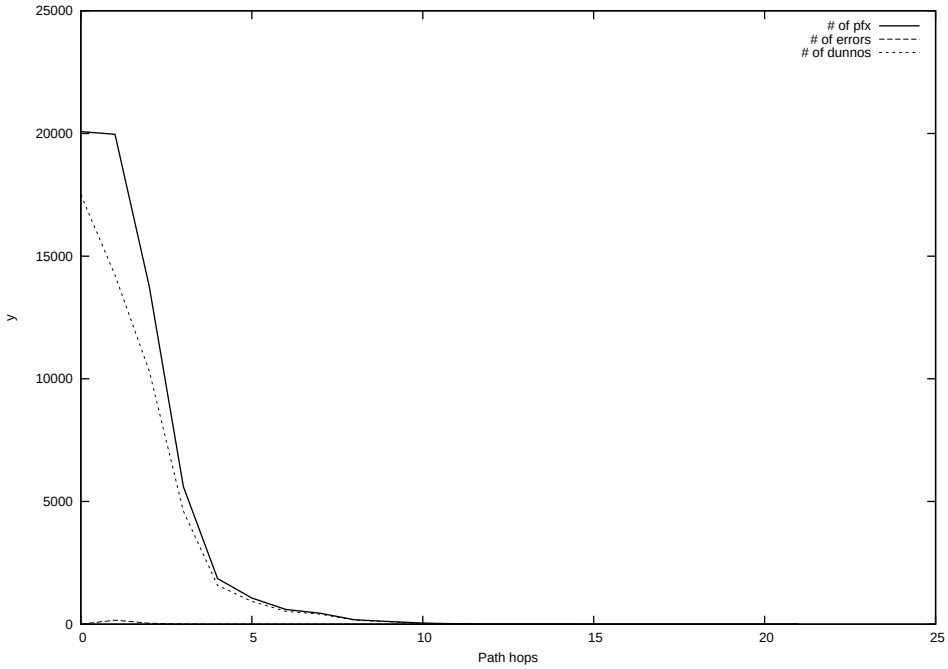
2014-10-18



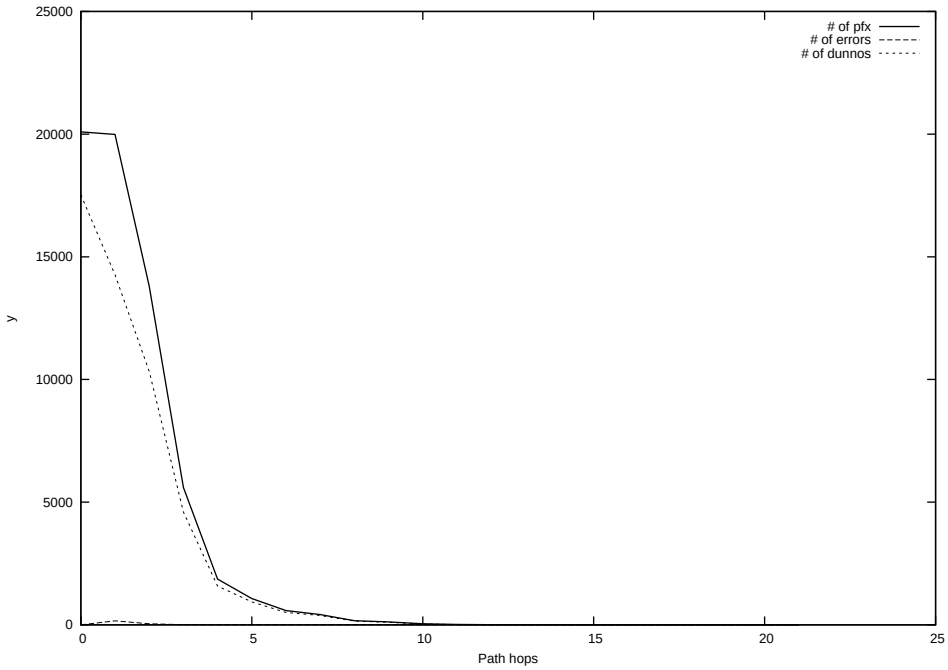
2014-10-19



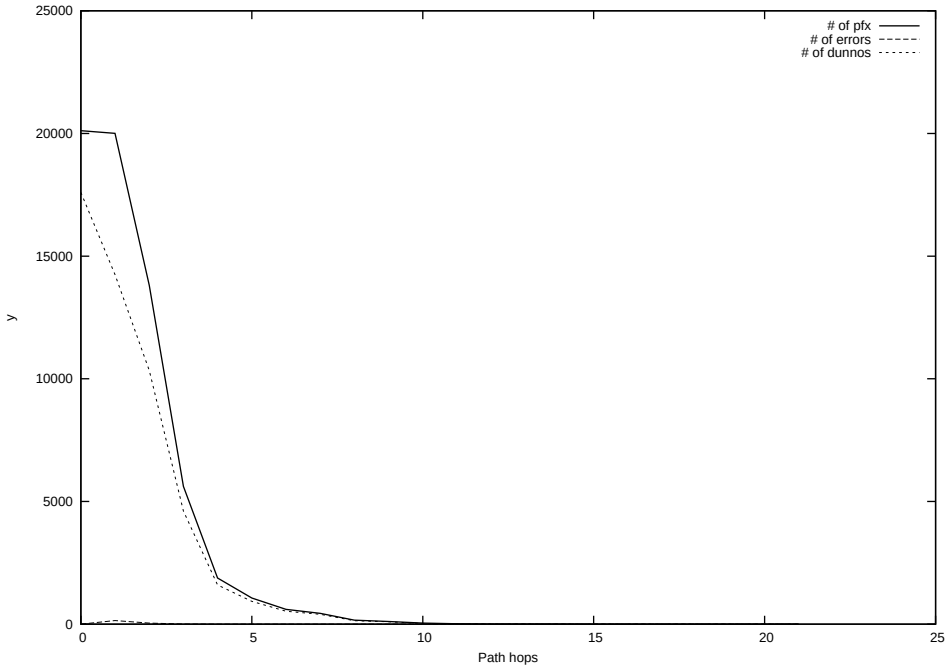
2014-10-20



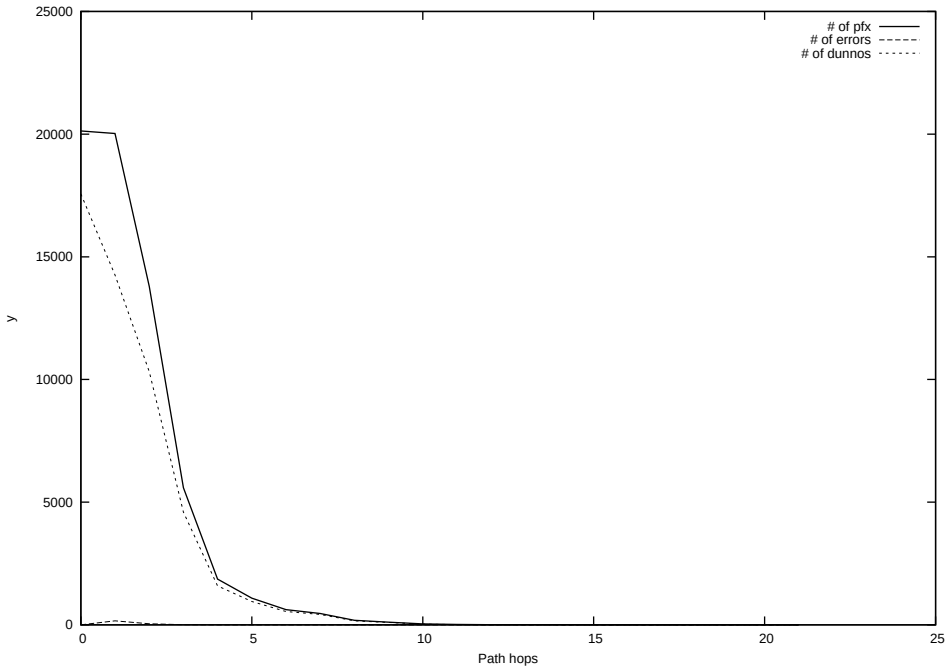
2014-10-21



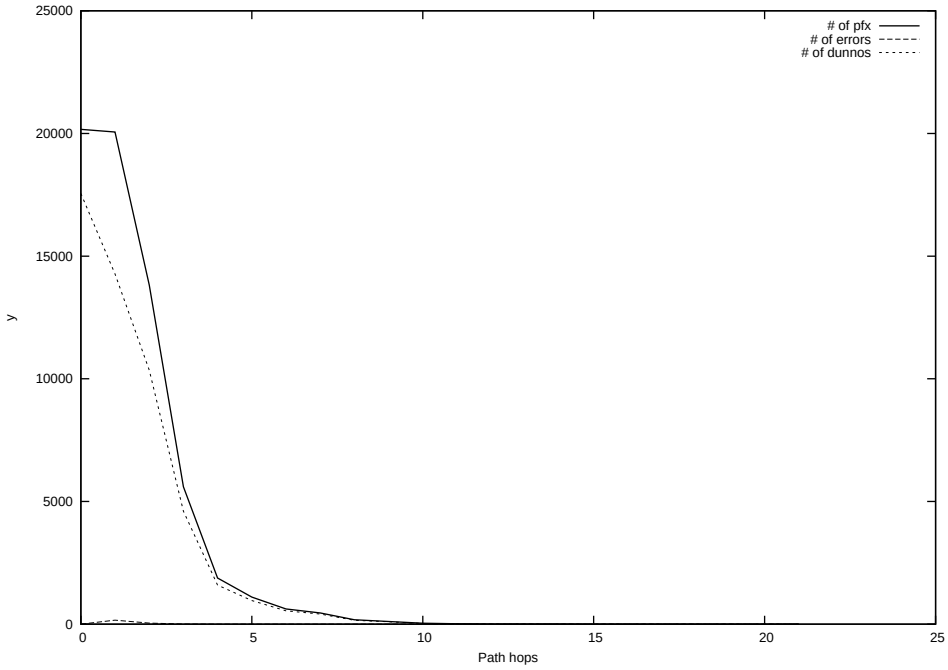
2014-10-22



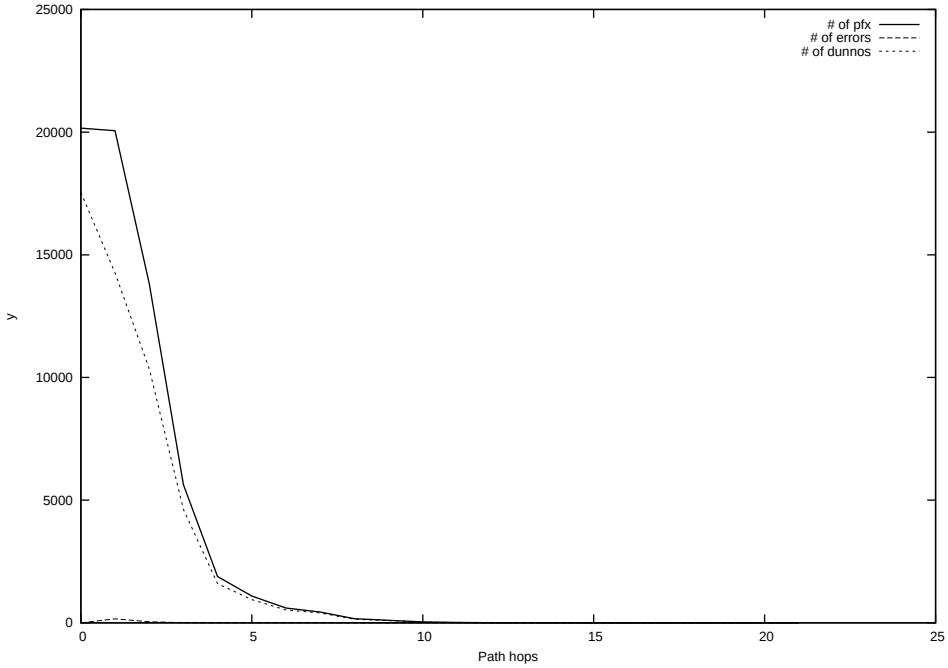
2014-10-23



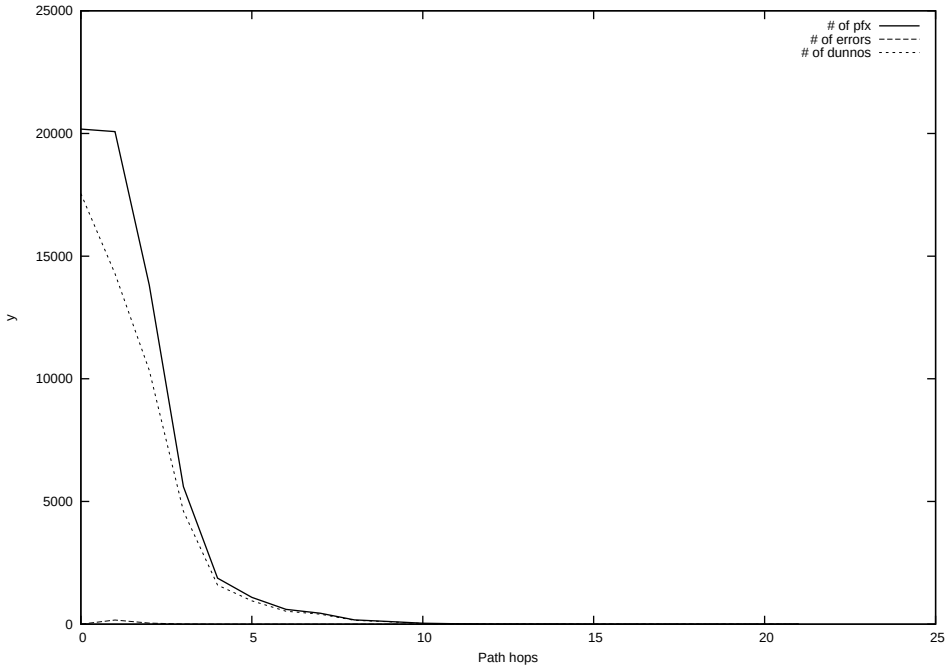
2014-10-24



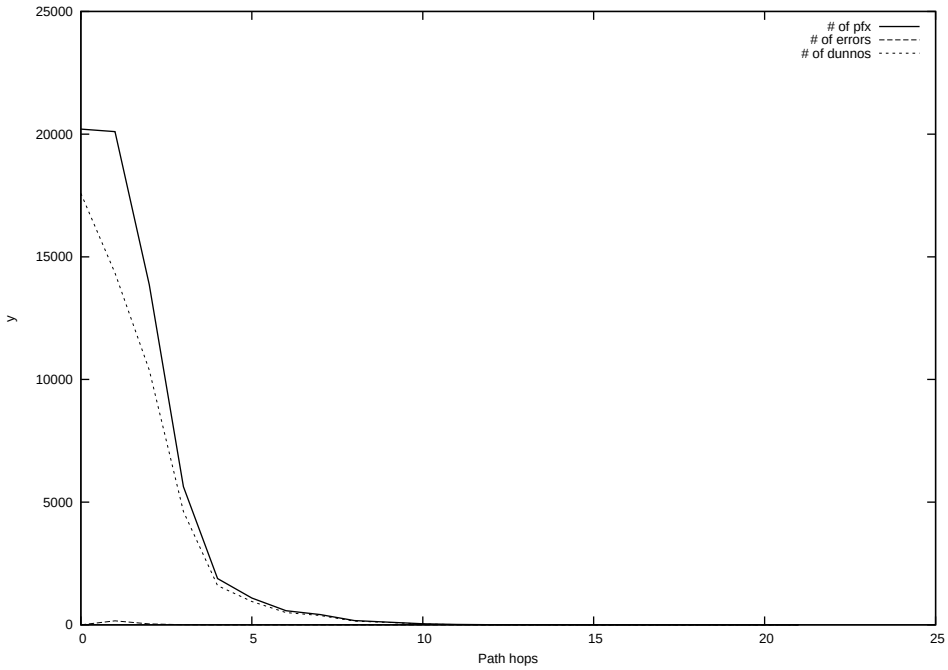
2014-10-25



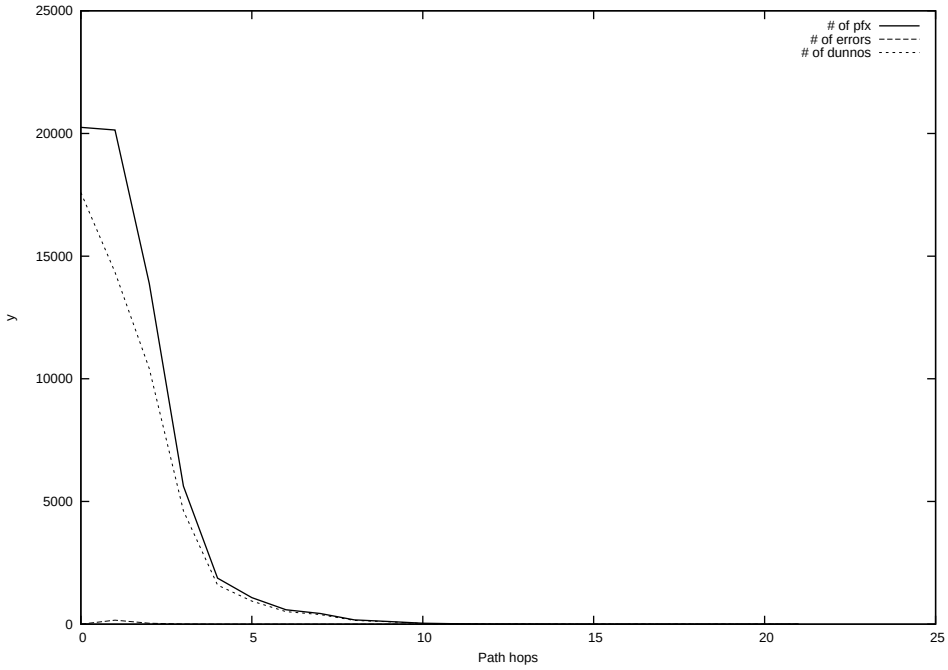
2014-10-26



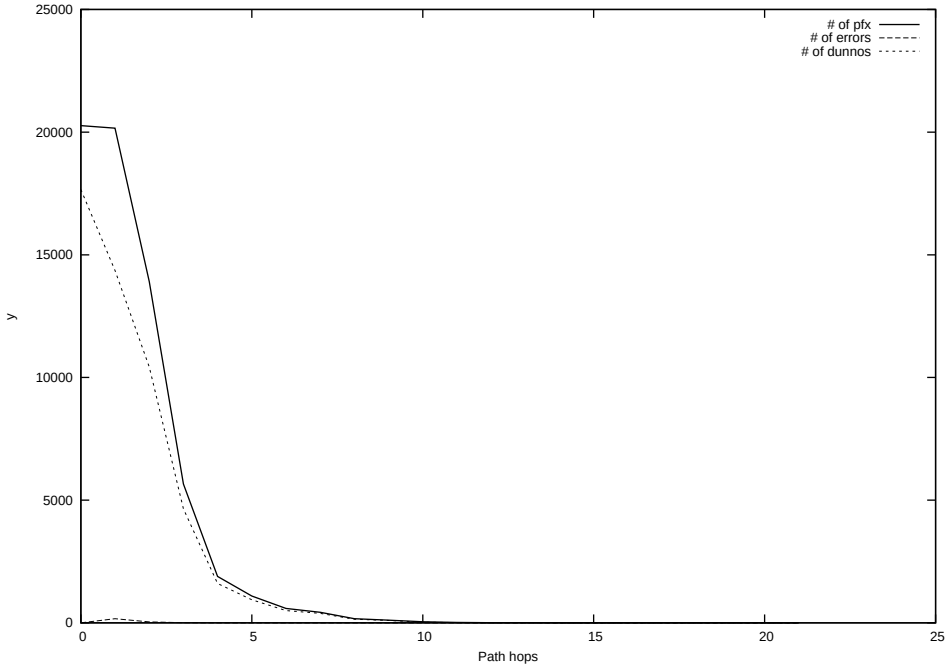
2014-10-27



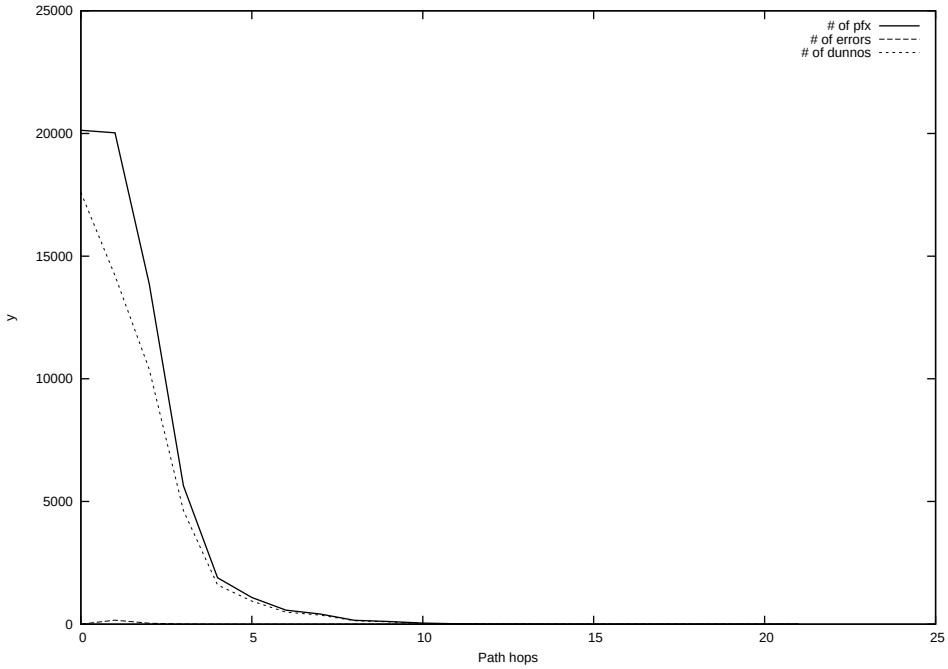
2014-10-28



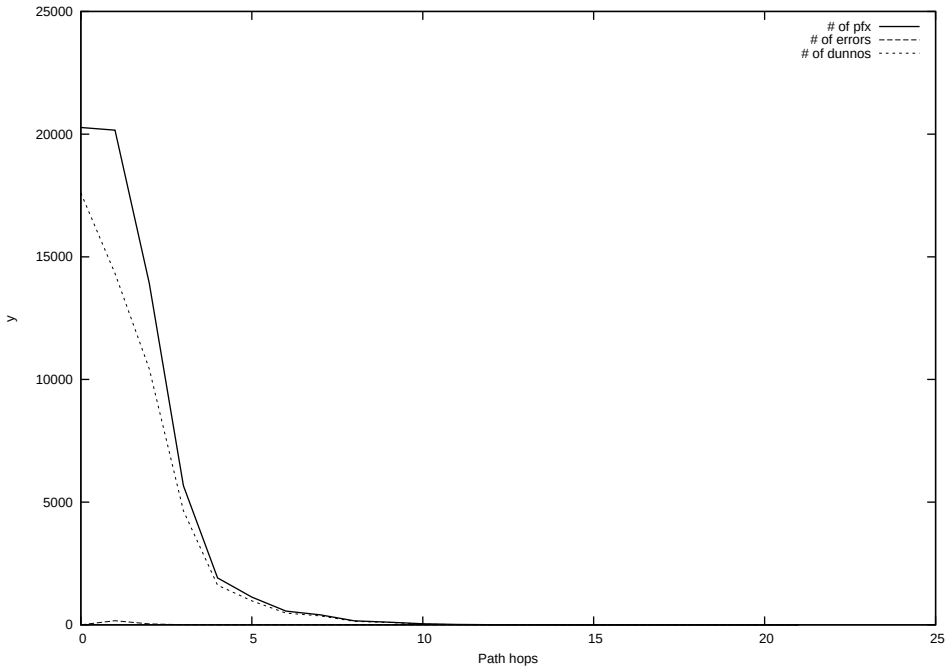
2014-10-29



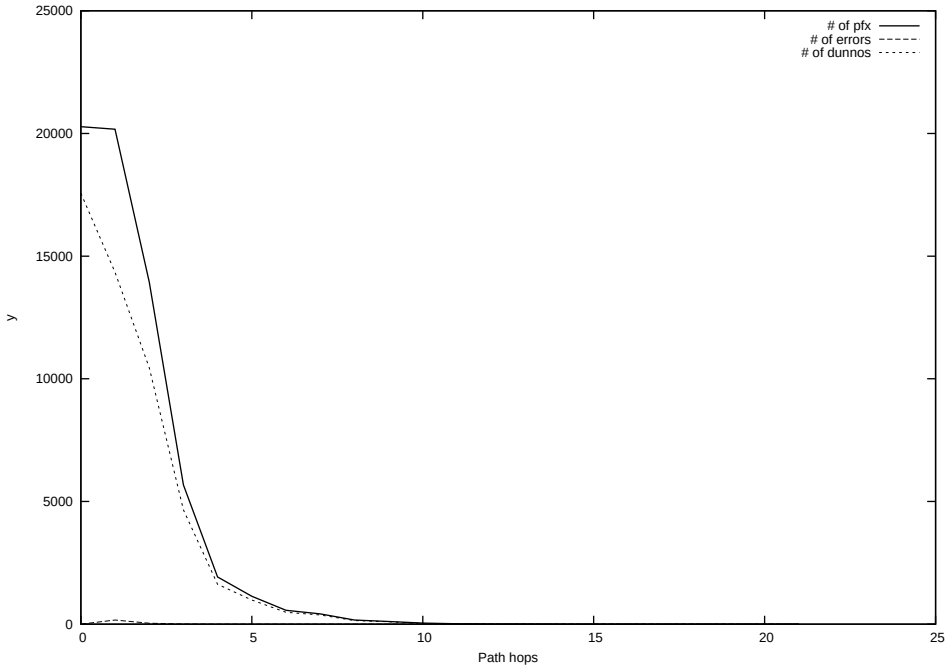
2014-10-30



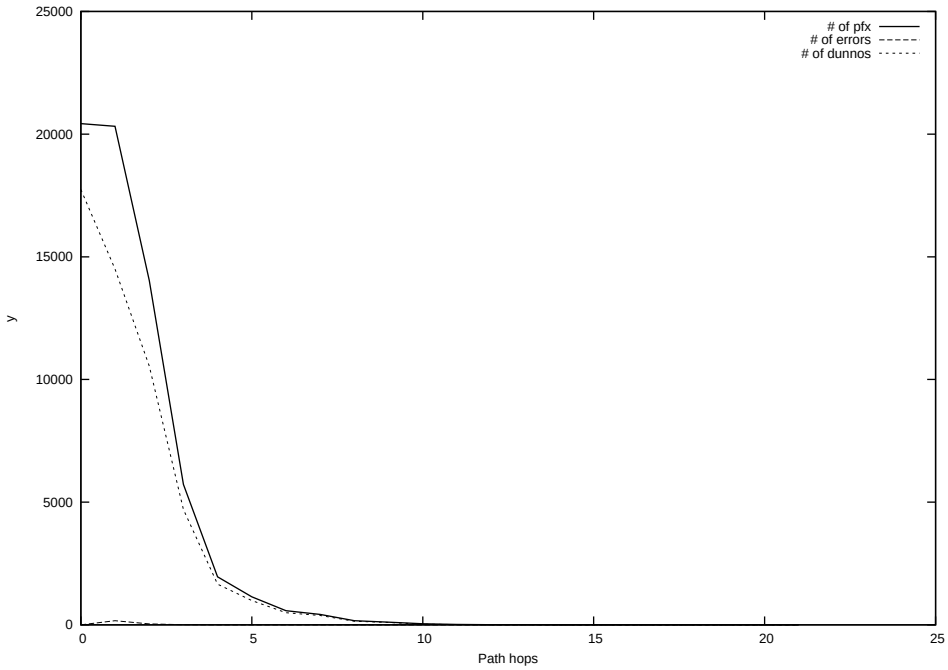
2014-10-31



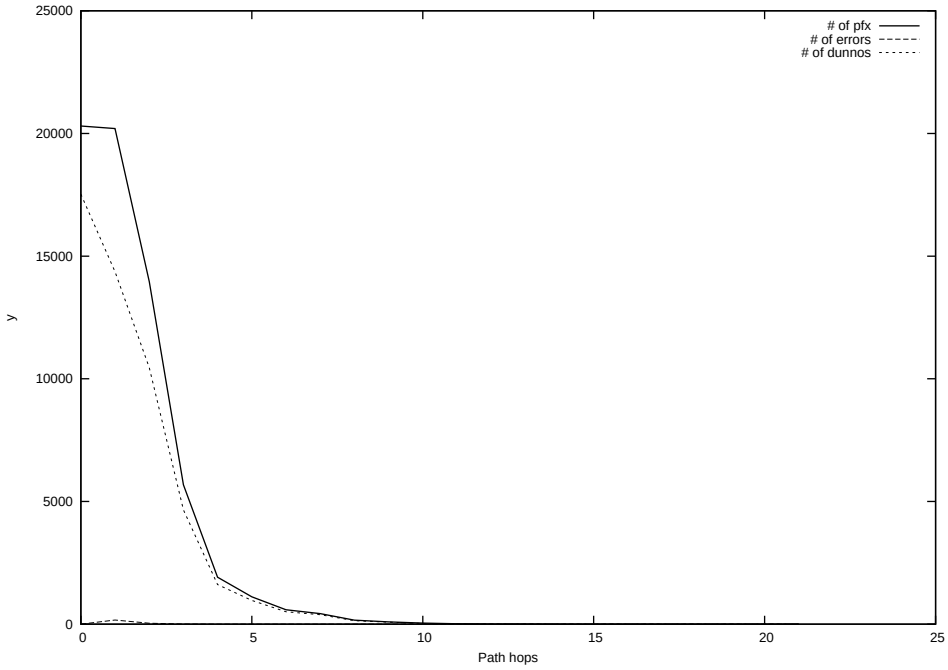
2014-11-02



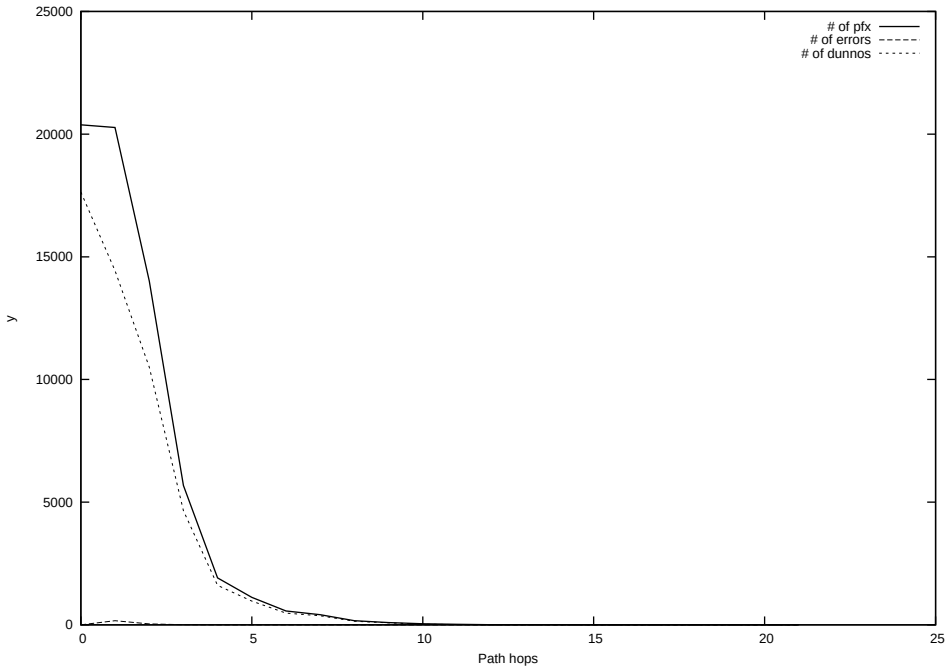
2014-11-03



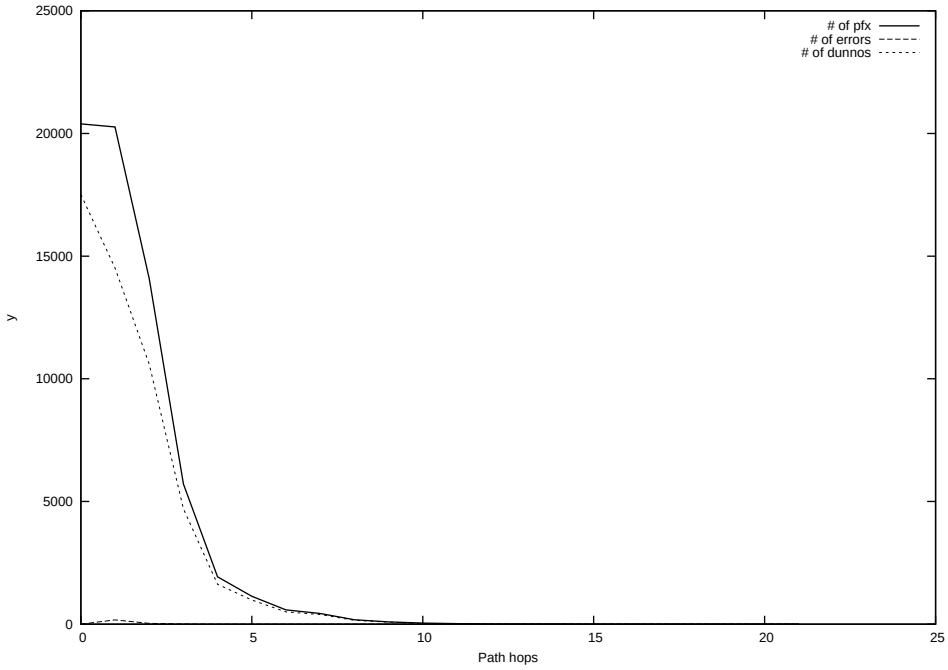
2014-11-04



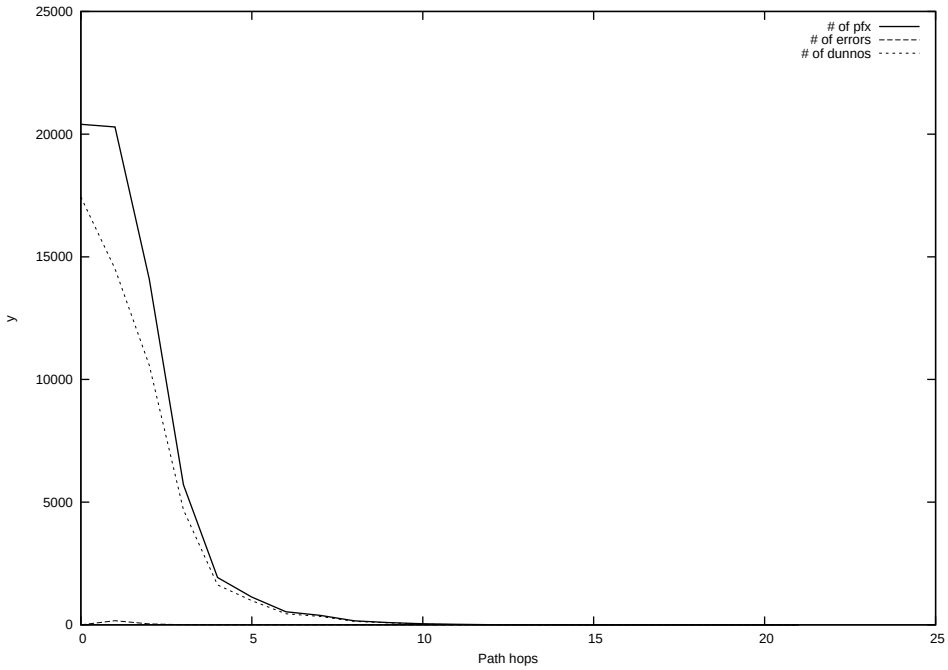
2014-11-05



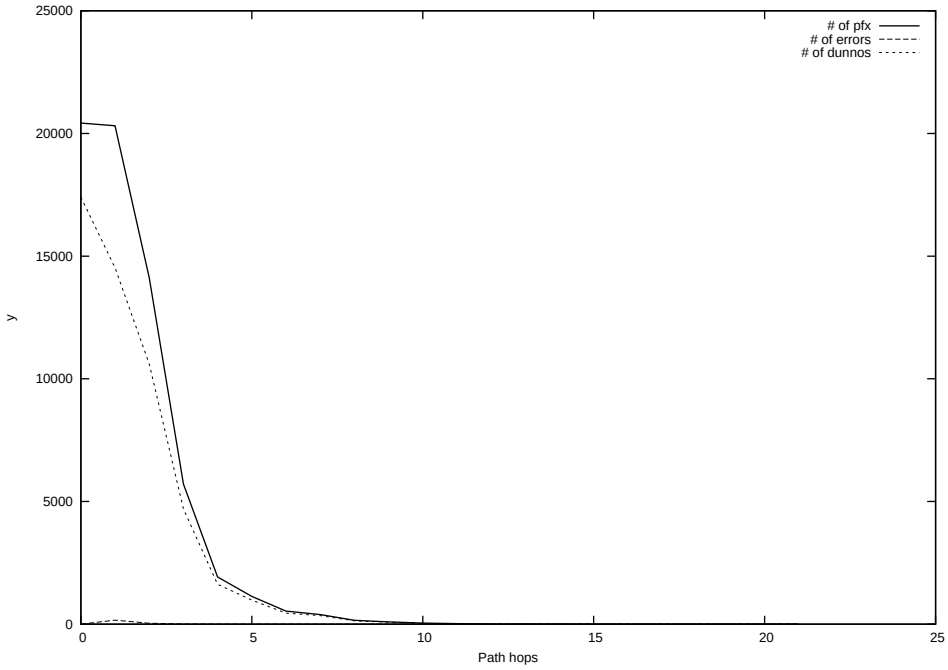
2014-11-06



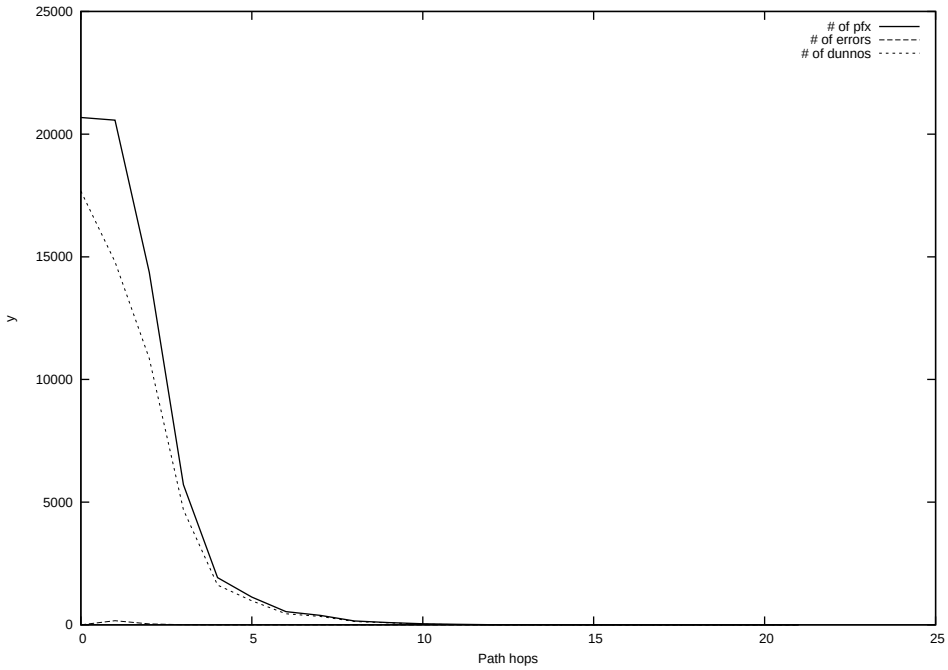
2014-11-07



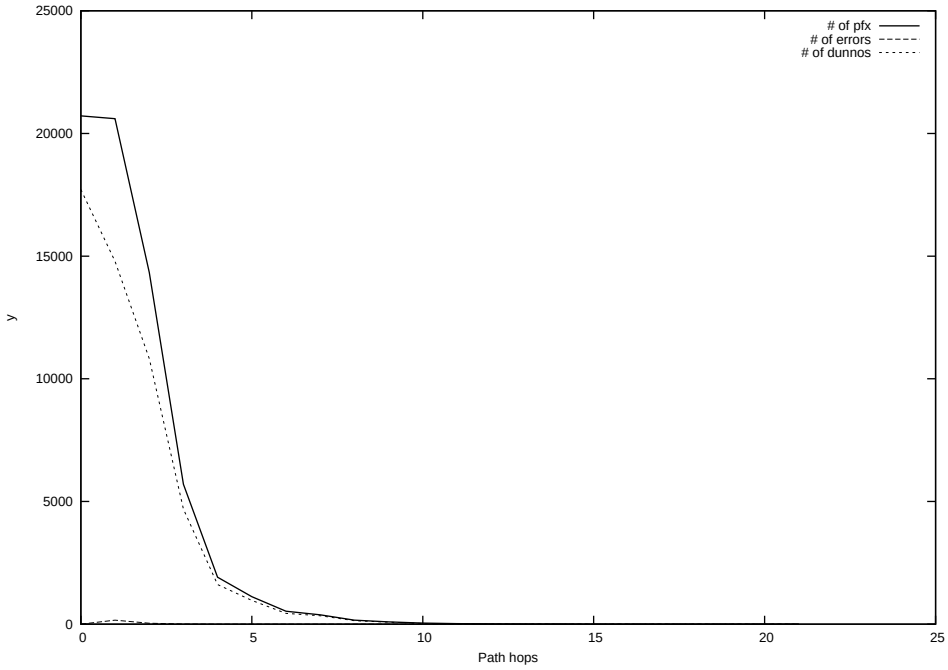
2014-11-08



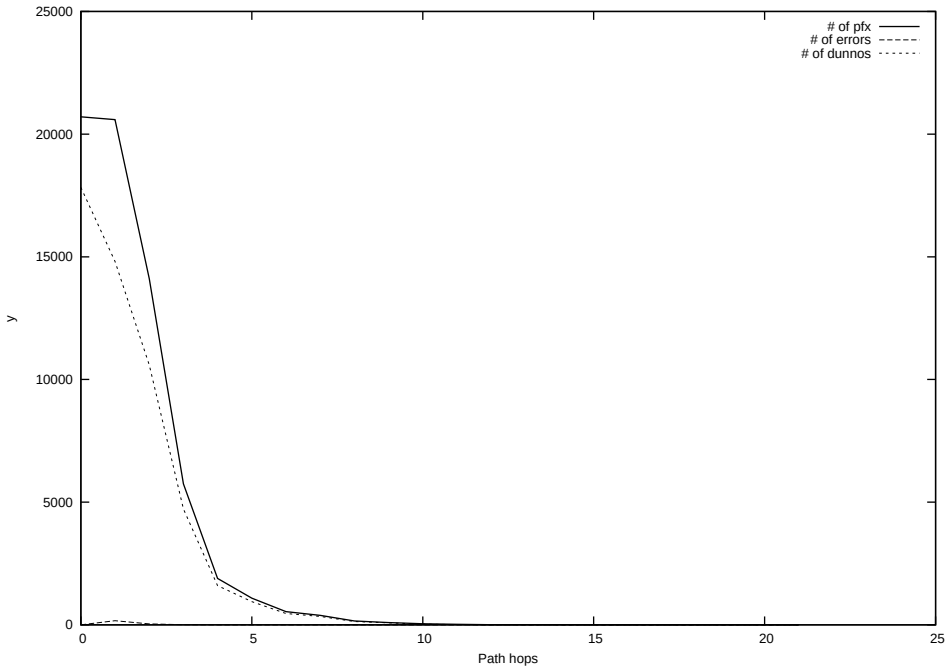
2014-11-09



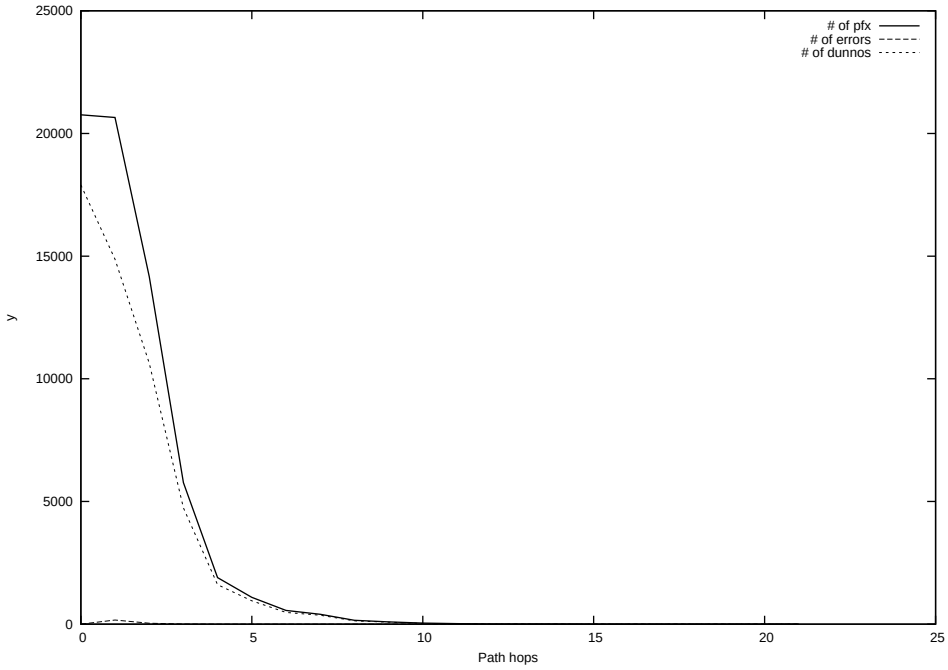
2014-11-10



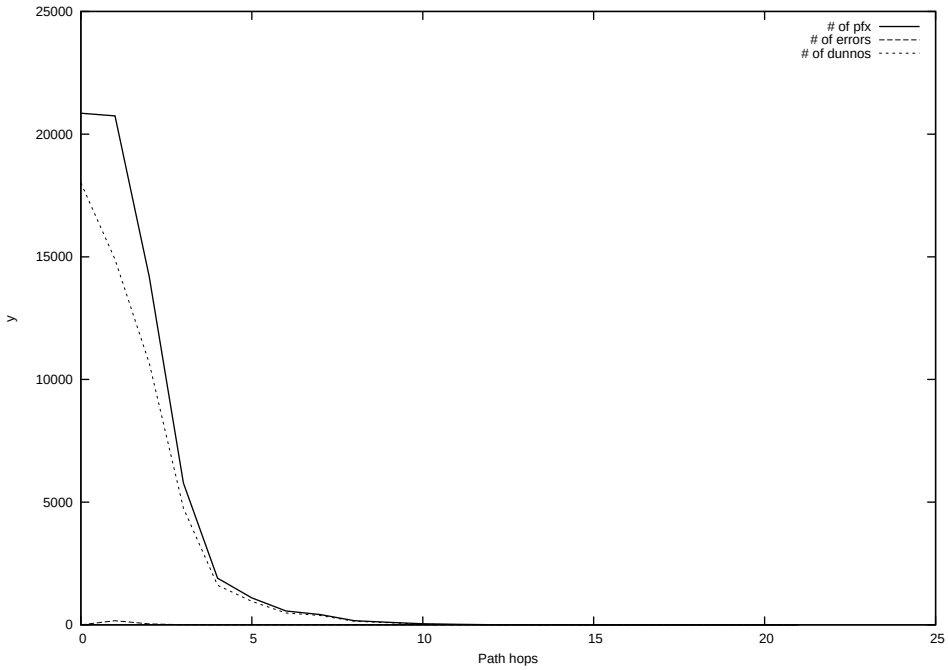
2014-11-11



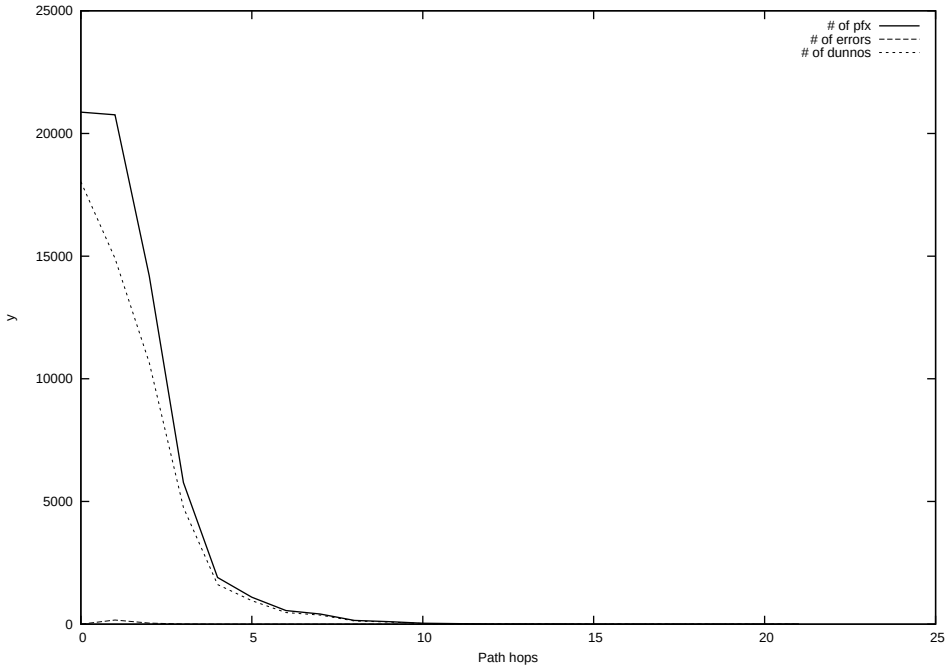
2014-11-12



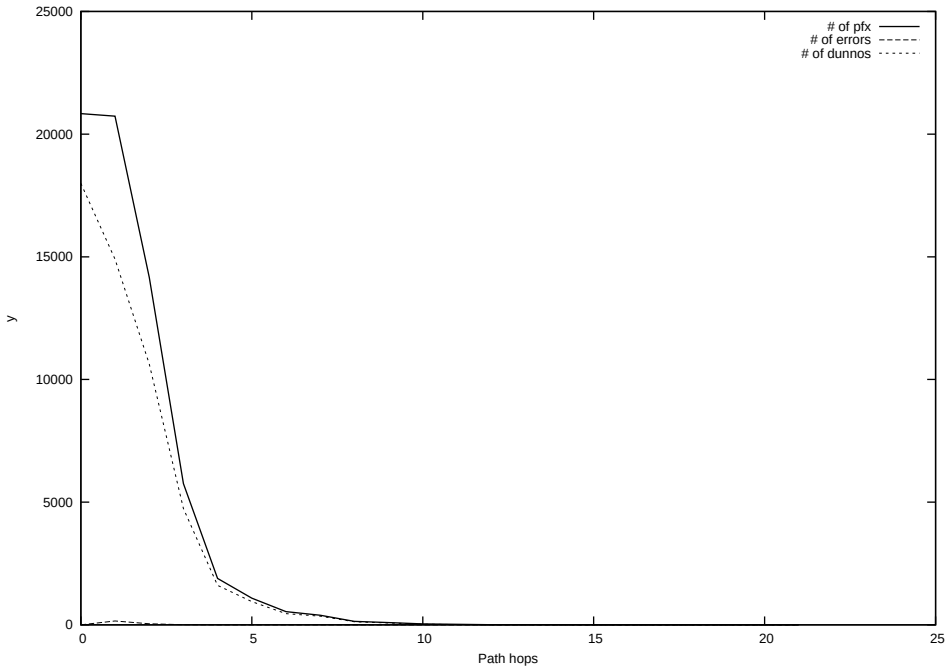
2014-11-13



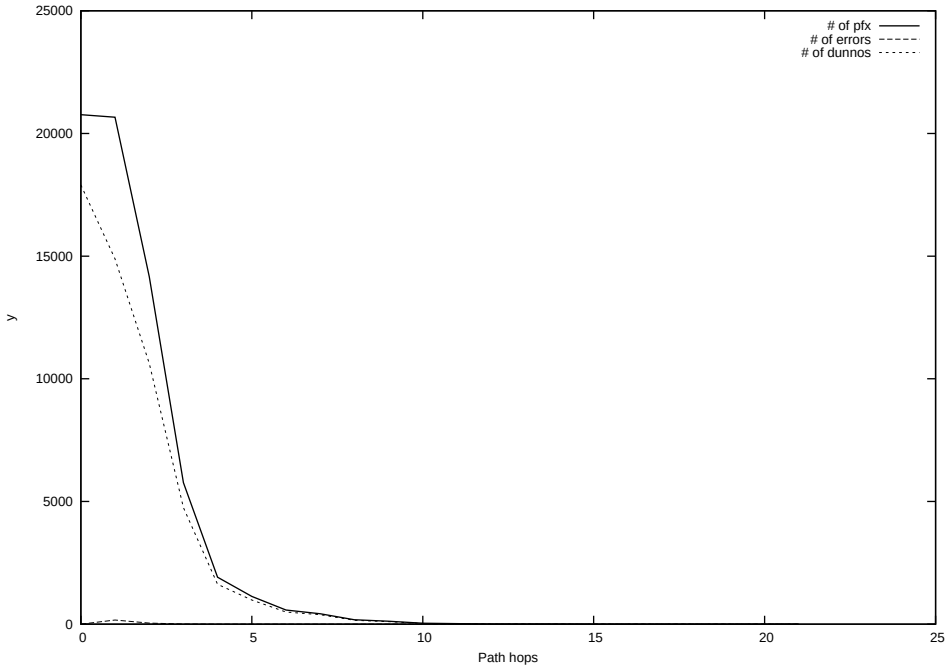
2014-11-14



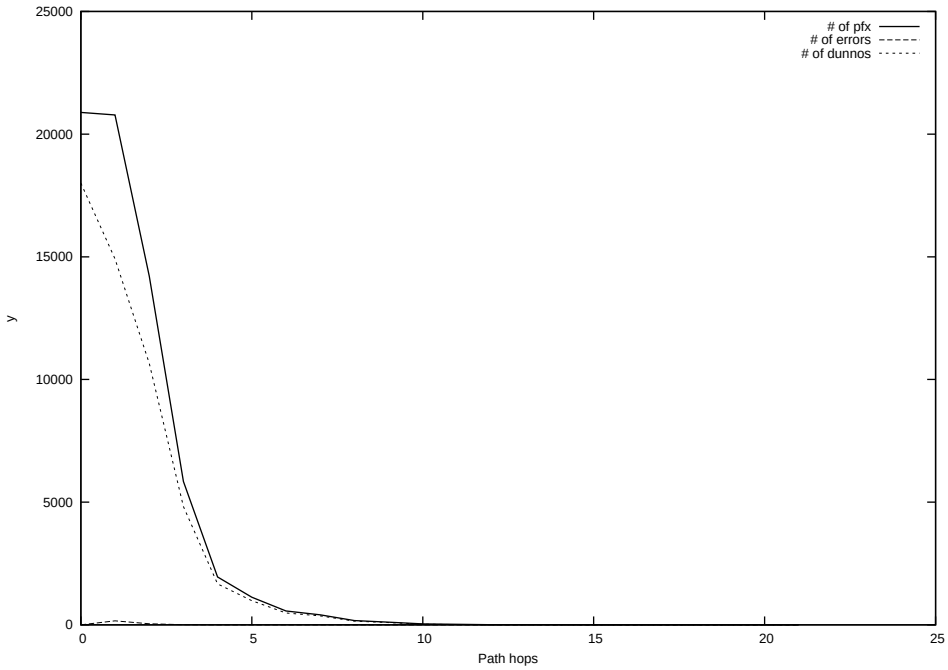
2014-11-15



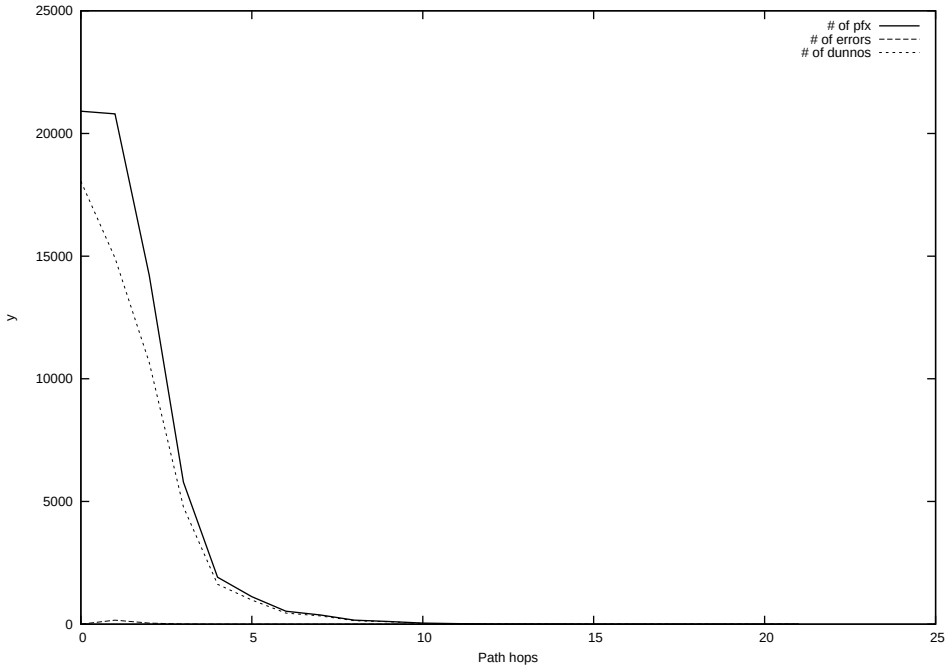
2014-11-16



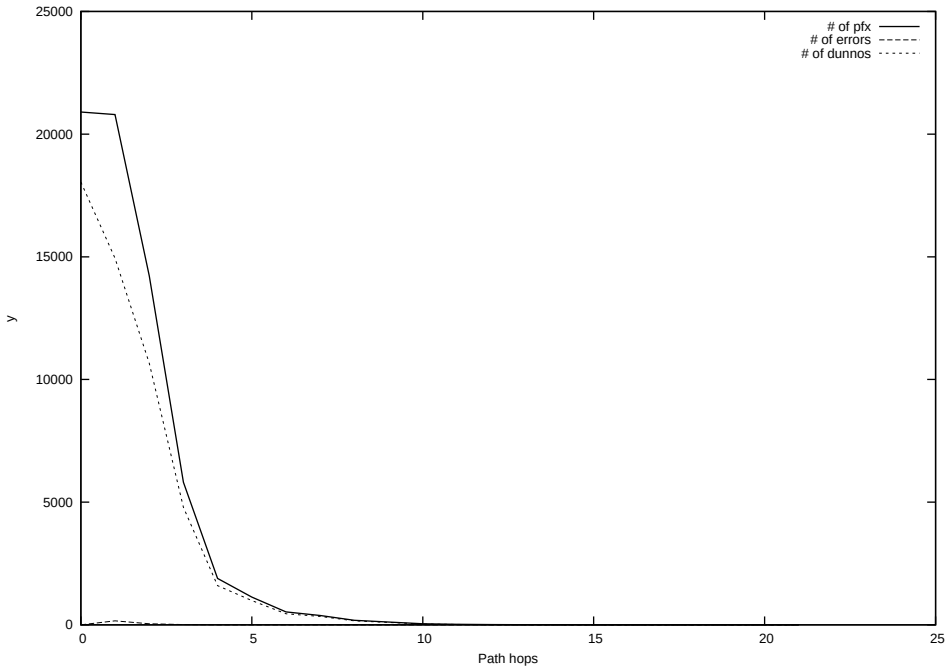
2014-11-17



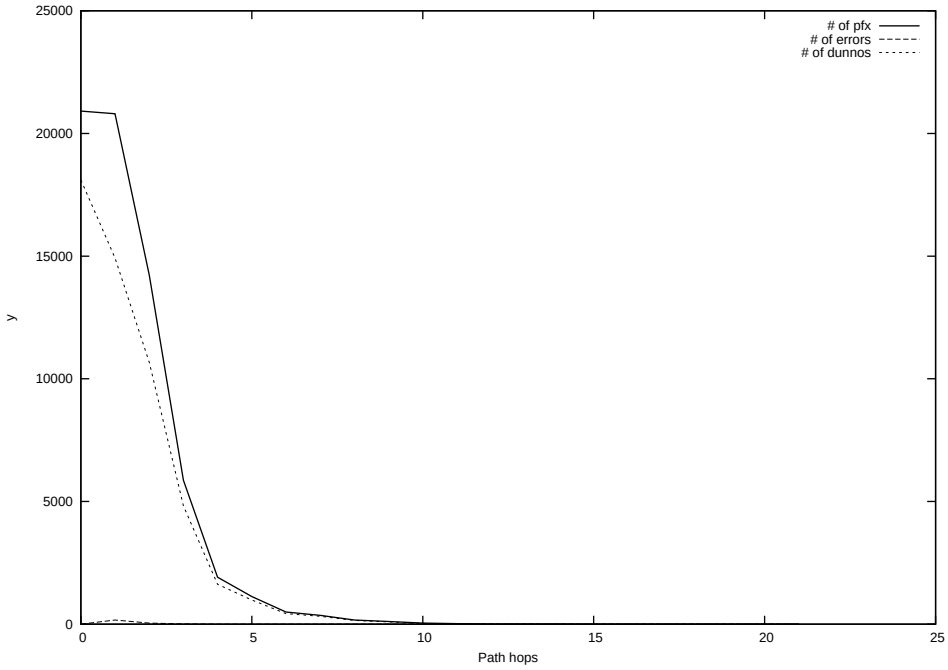
2014-11-18



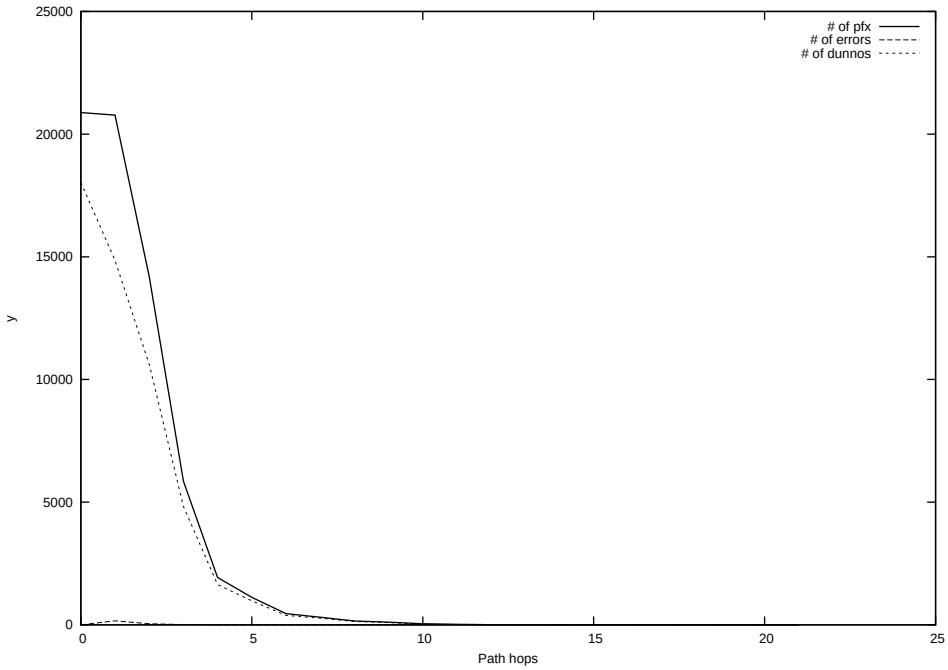
2014-11-19



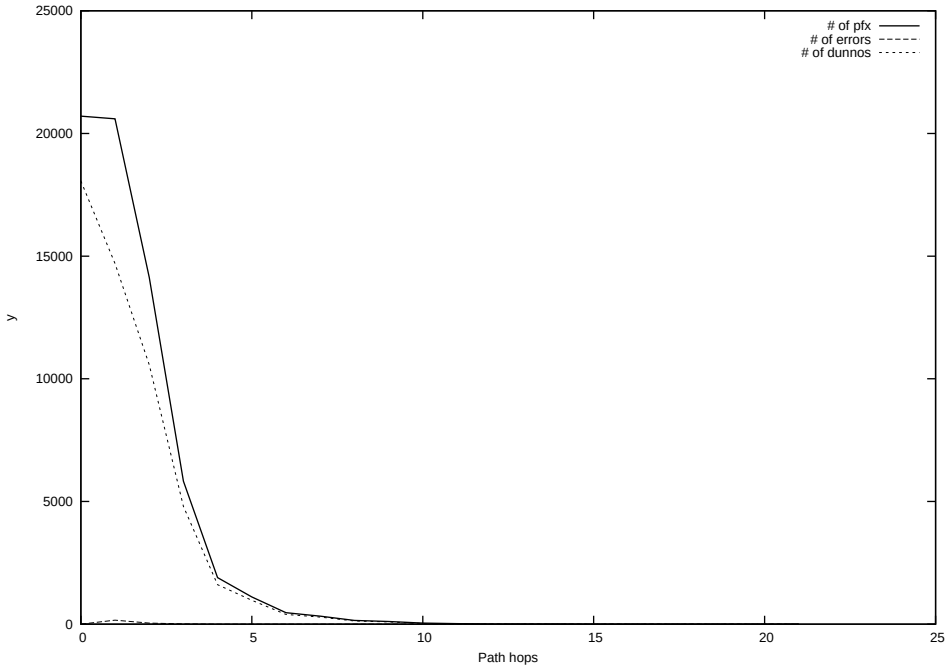
2014-11-20



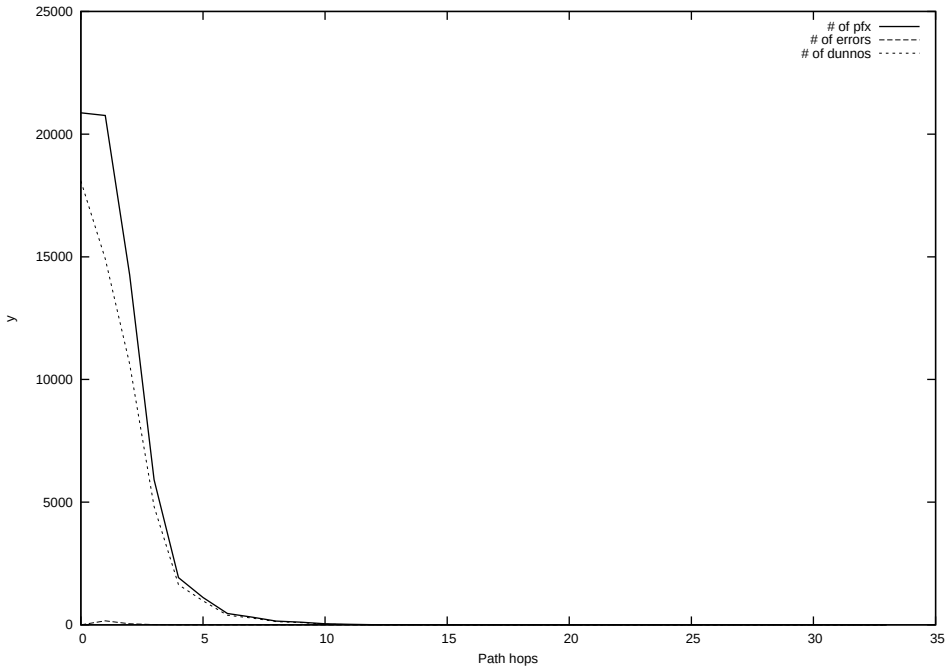
2014-11-21



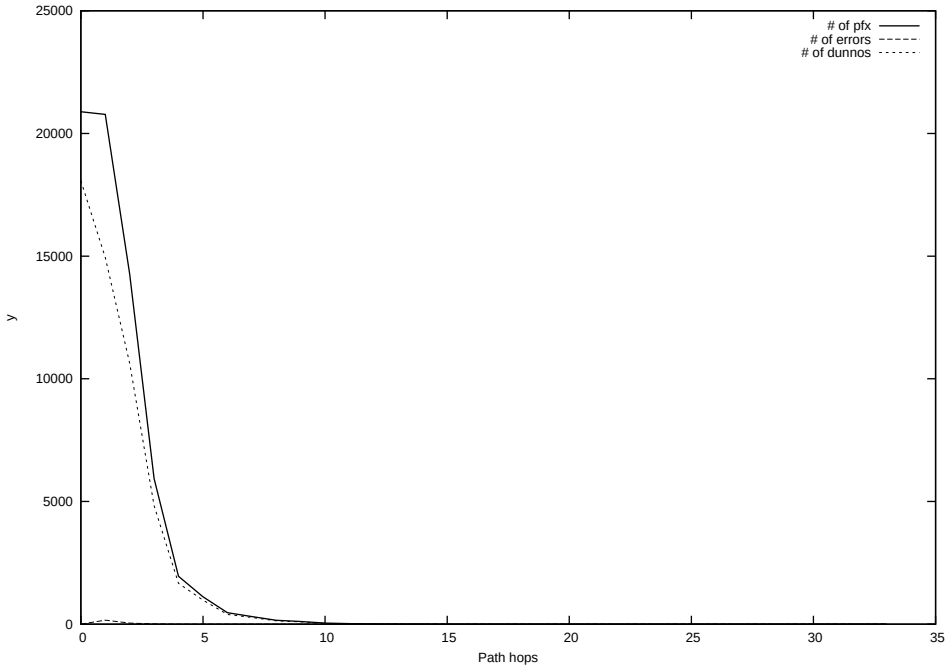
2014-11-22



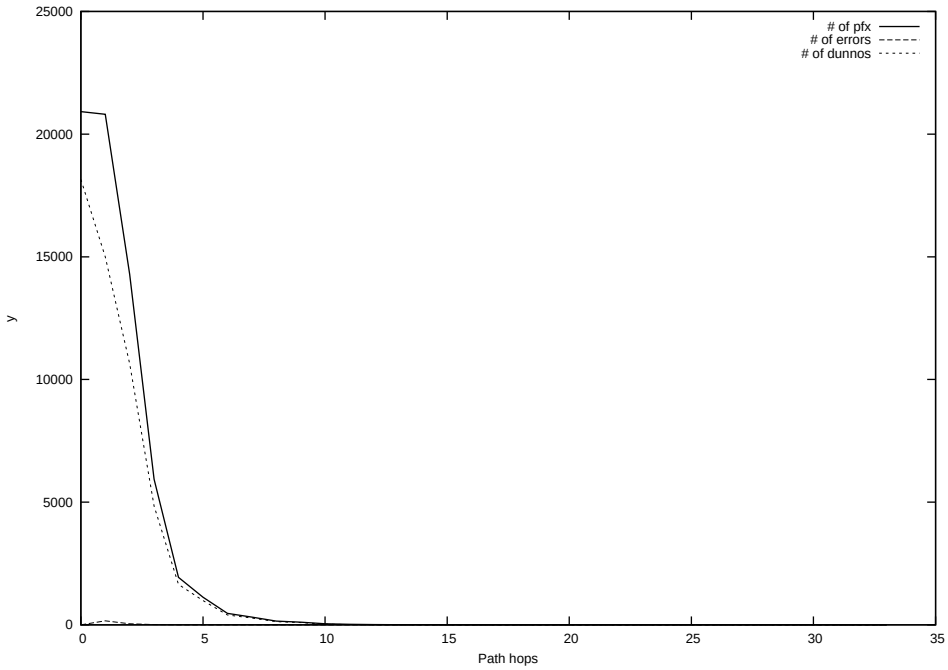
2014-11-23



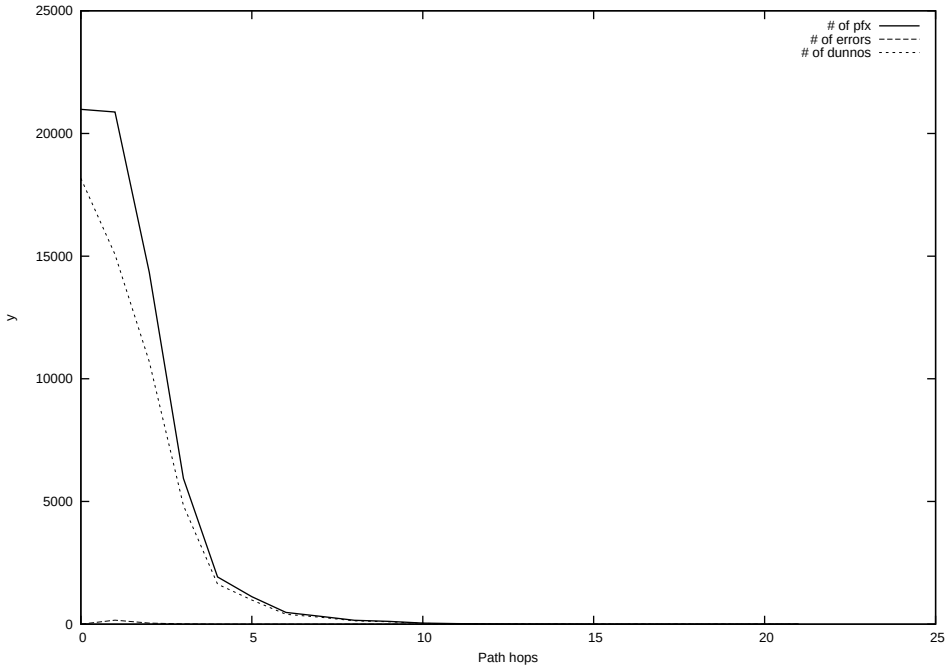
2014-11-24



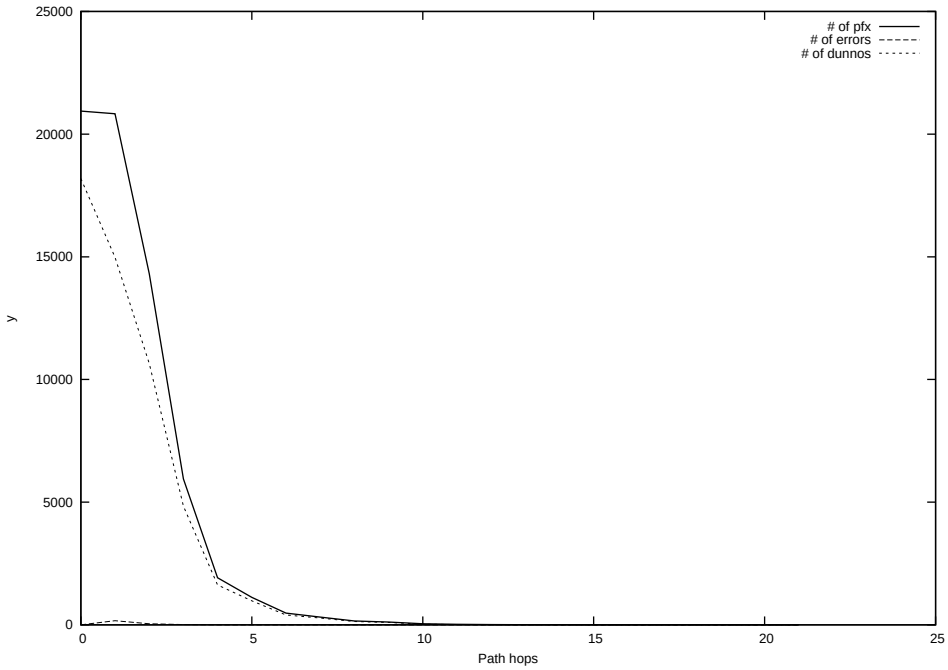
2014-11-25



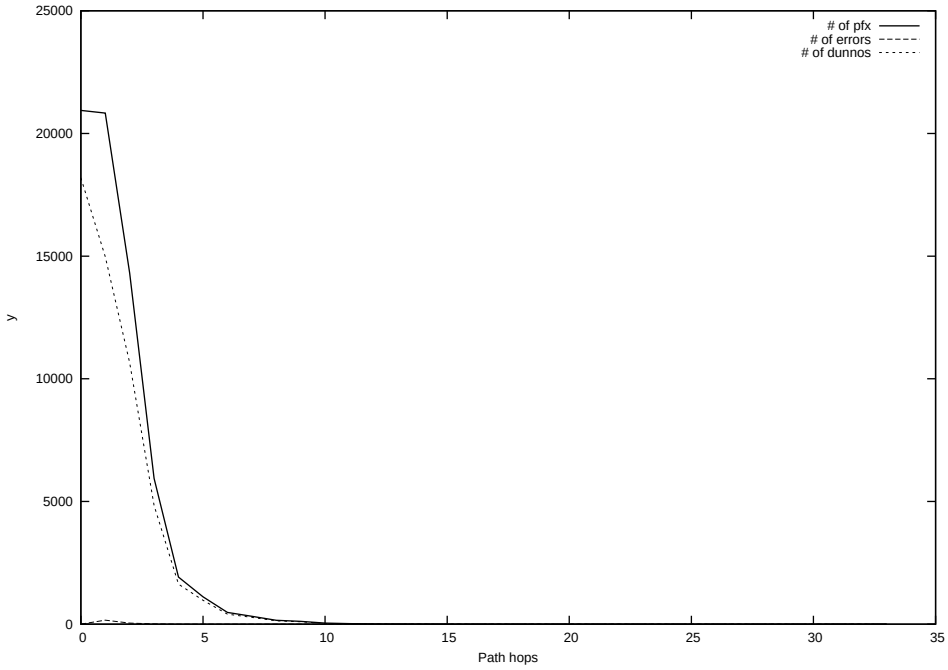
2014-11-26



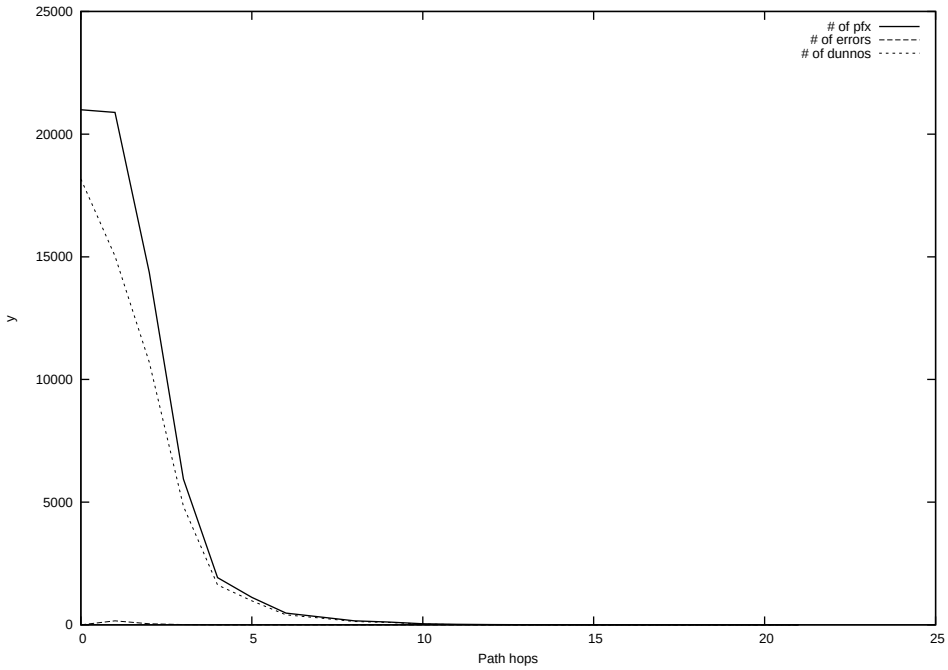
2014-11-27



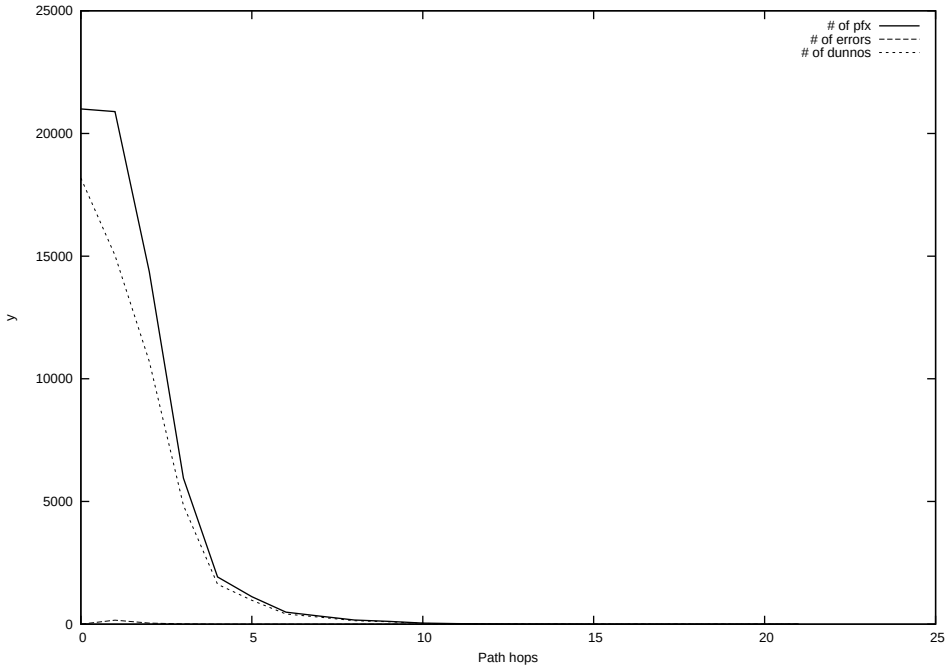
2014-11-28



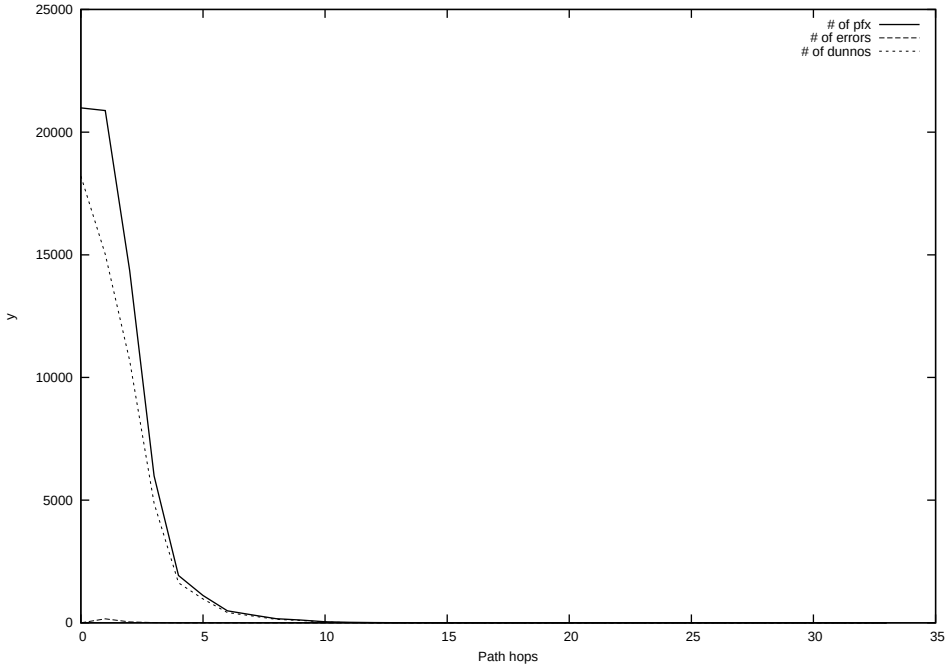
2014-11-29



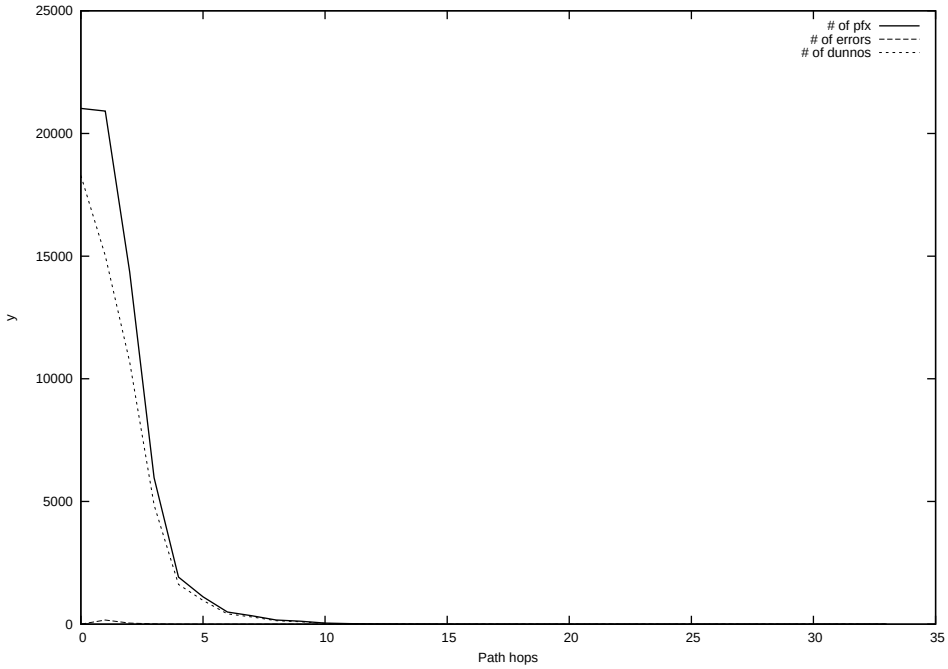
2014-11-30



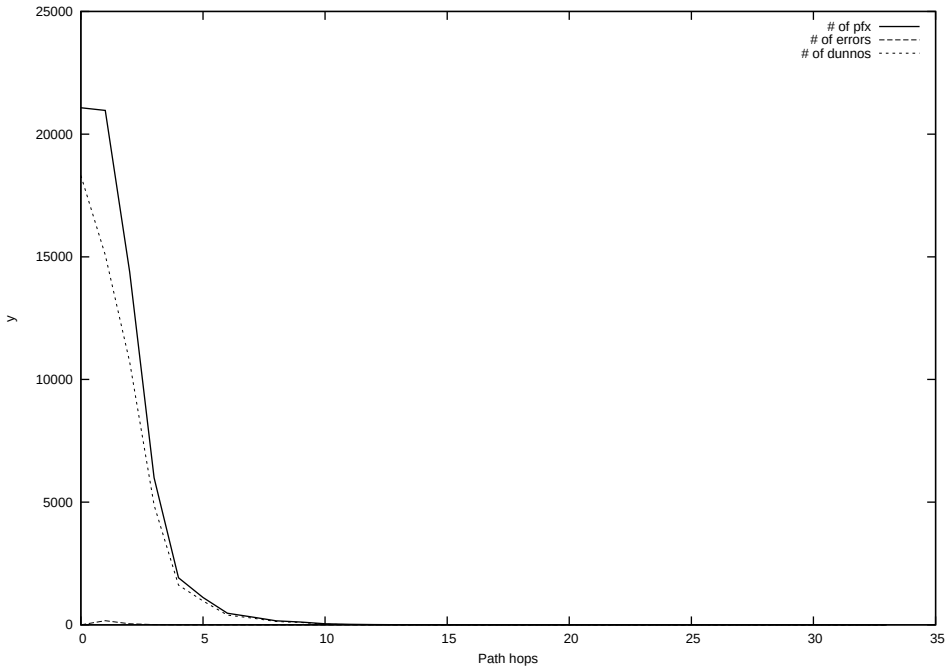
2014-12-01



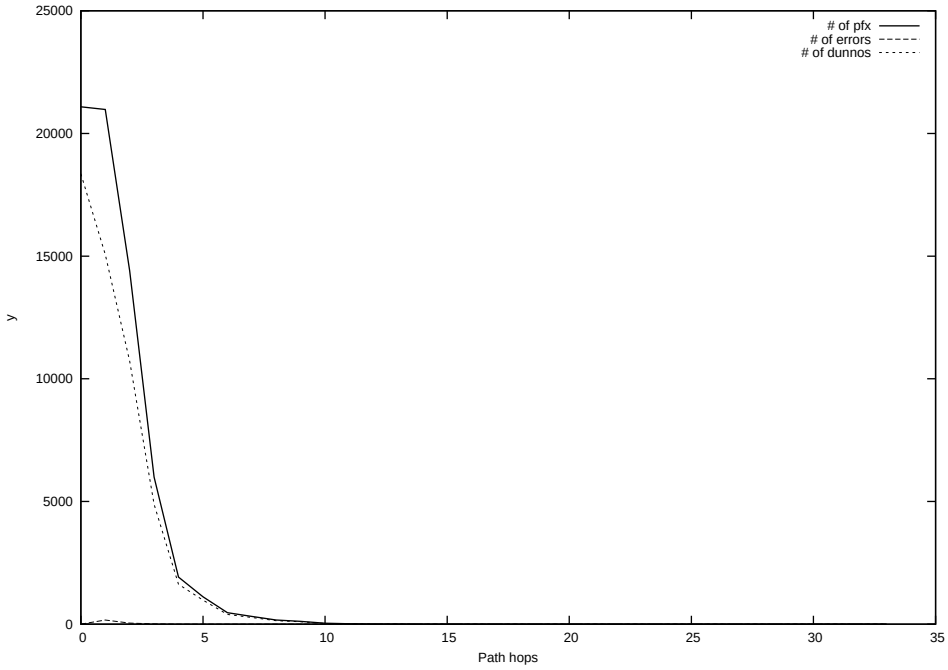
2014-12-02



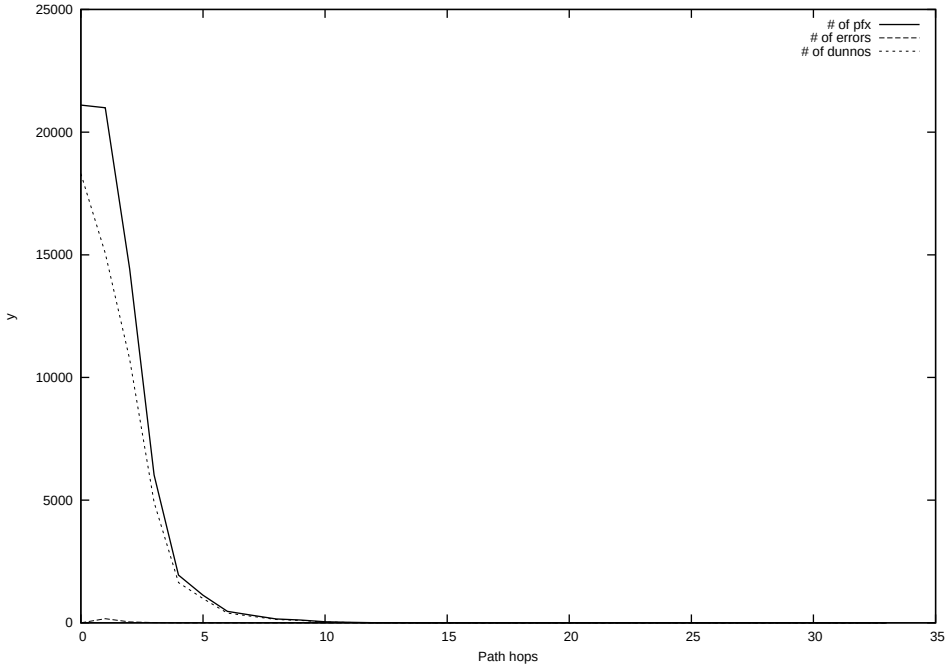
2014-12-03



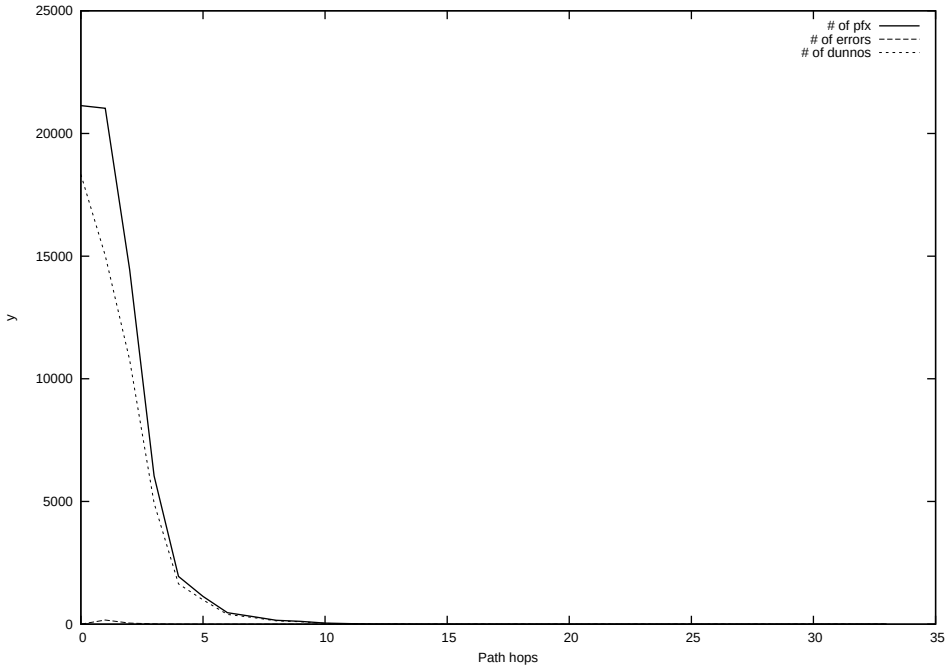
2014-12-04



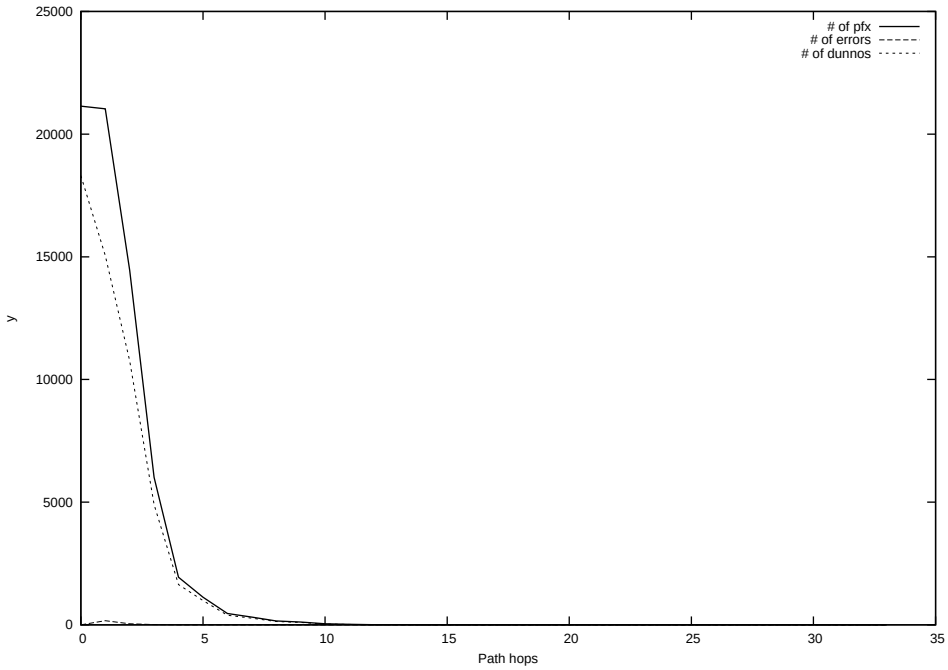
2014-12-05



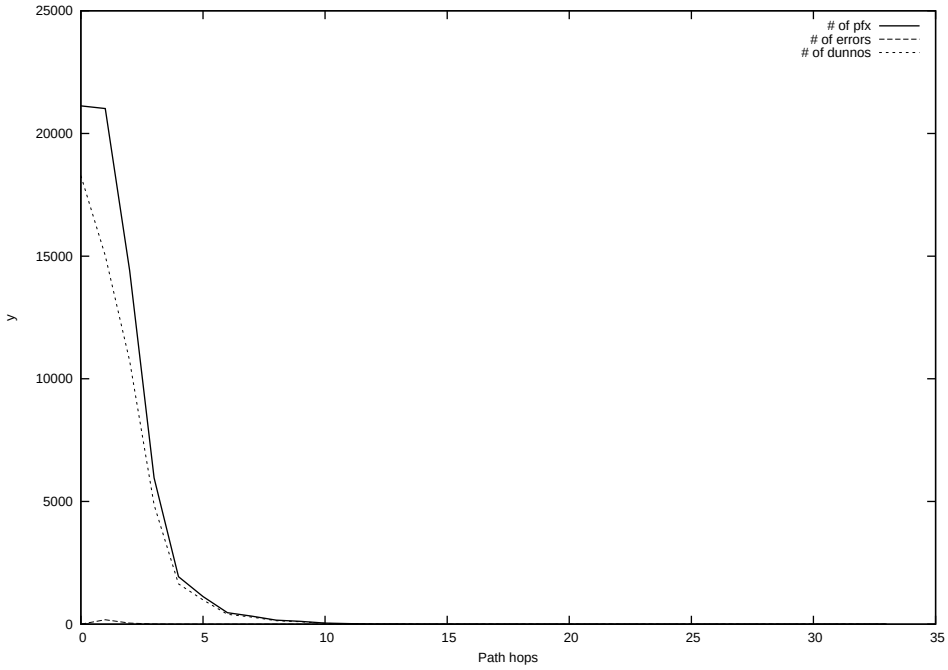
2014-12-06



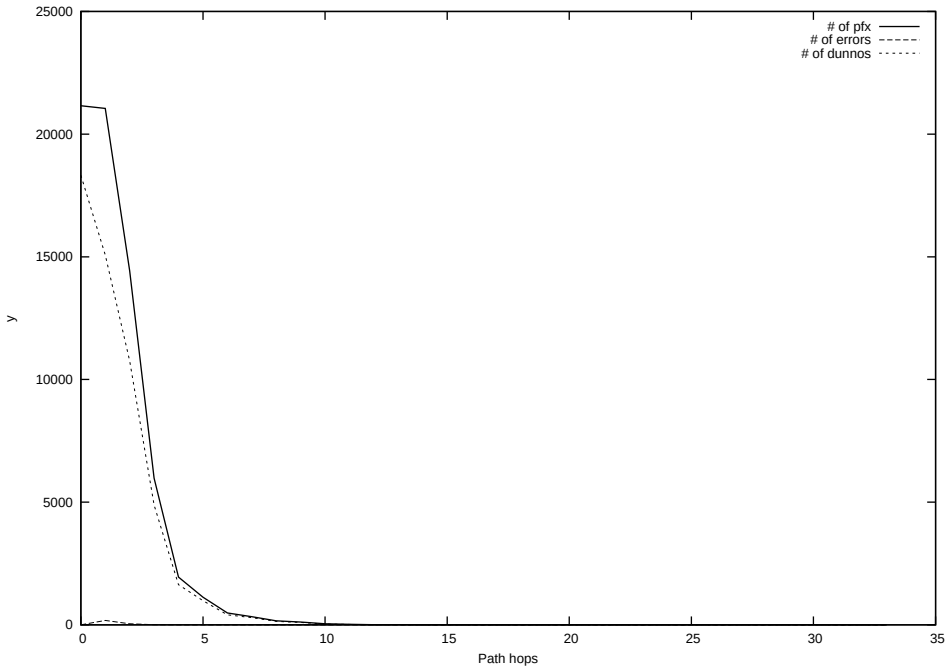
2014-12-07



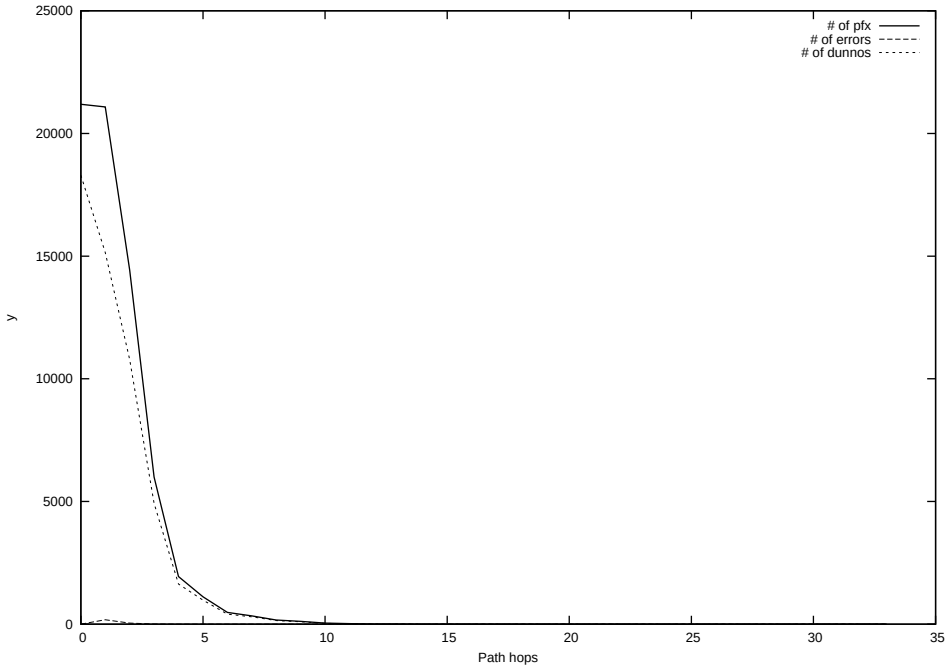
2014-12-08



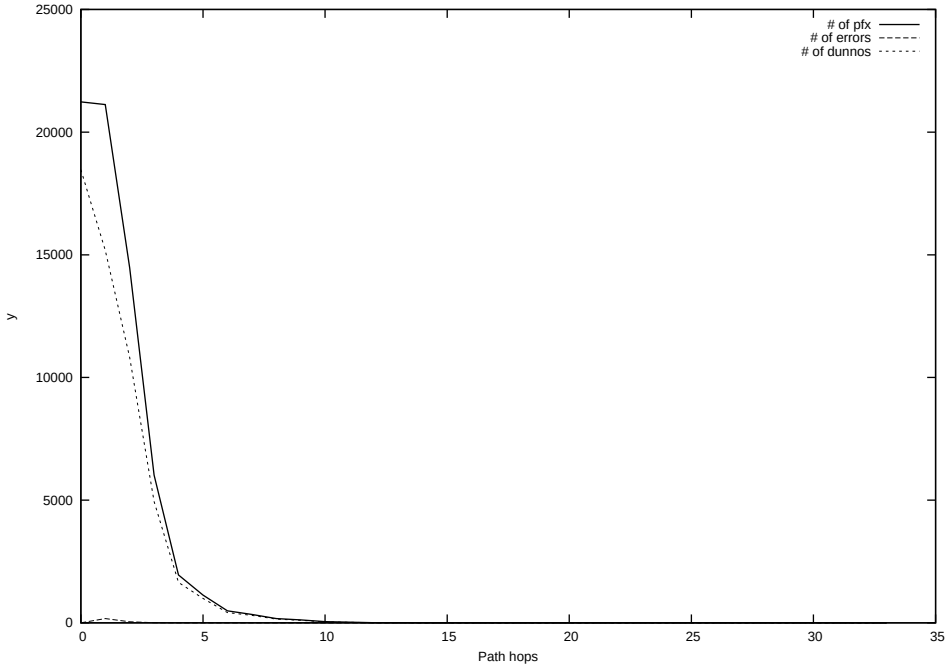
2014-12-09



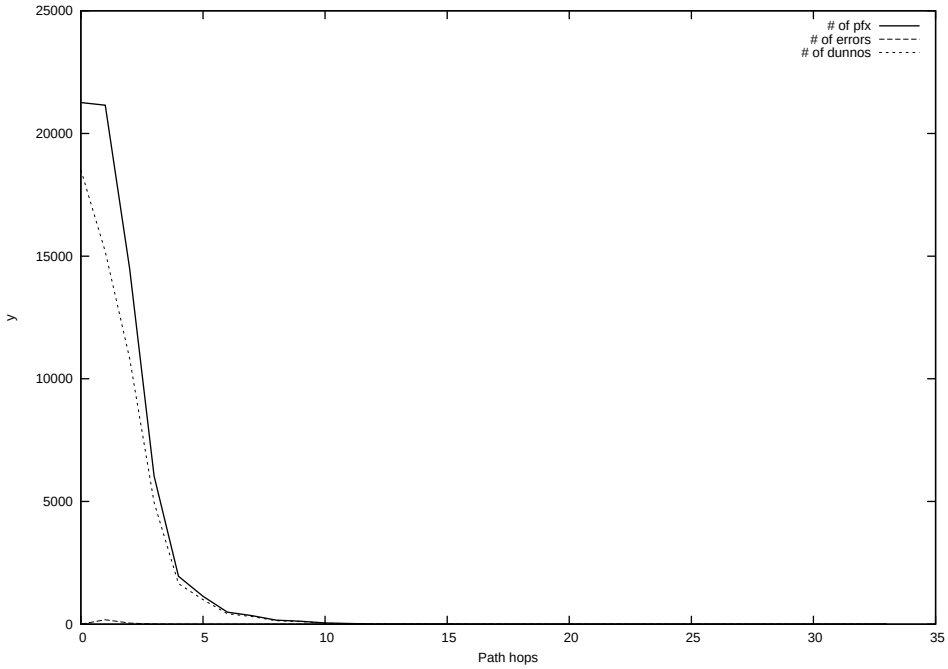
2014-12-10



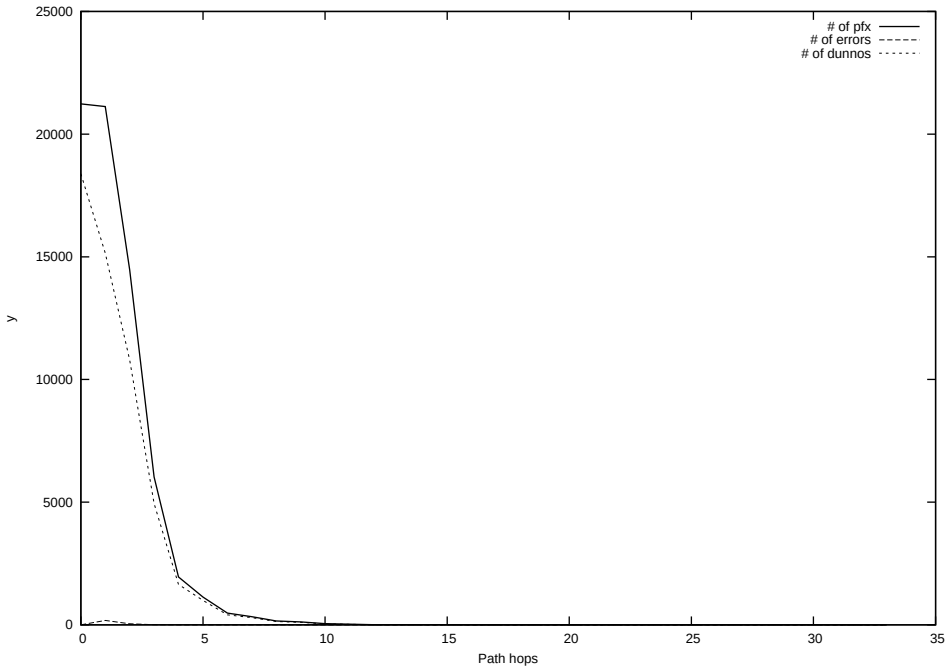
2014-12-11



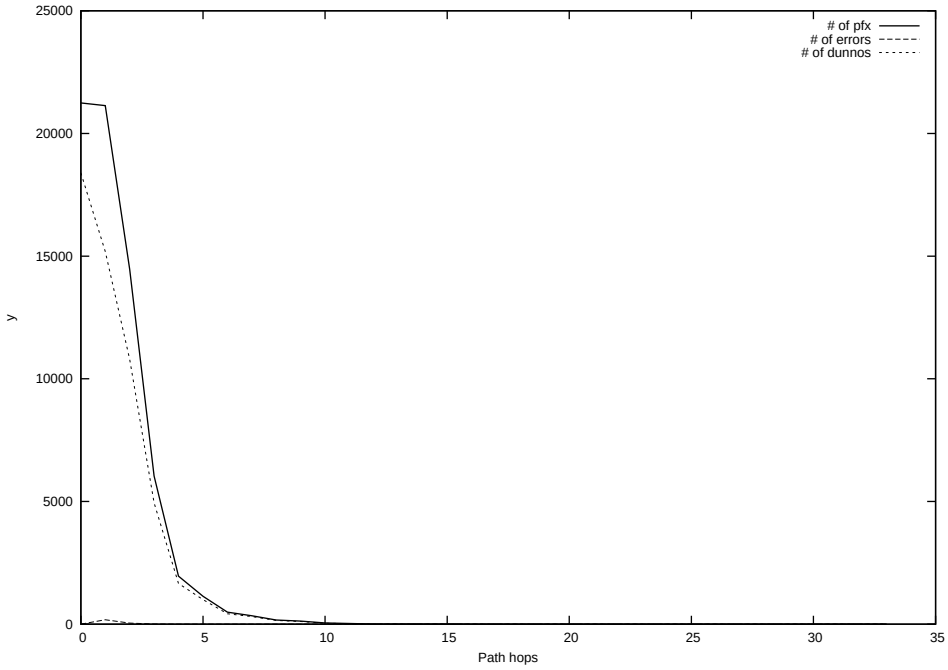
2014-12-12



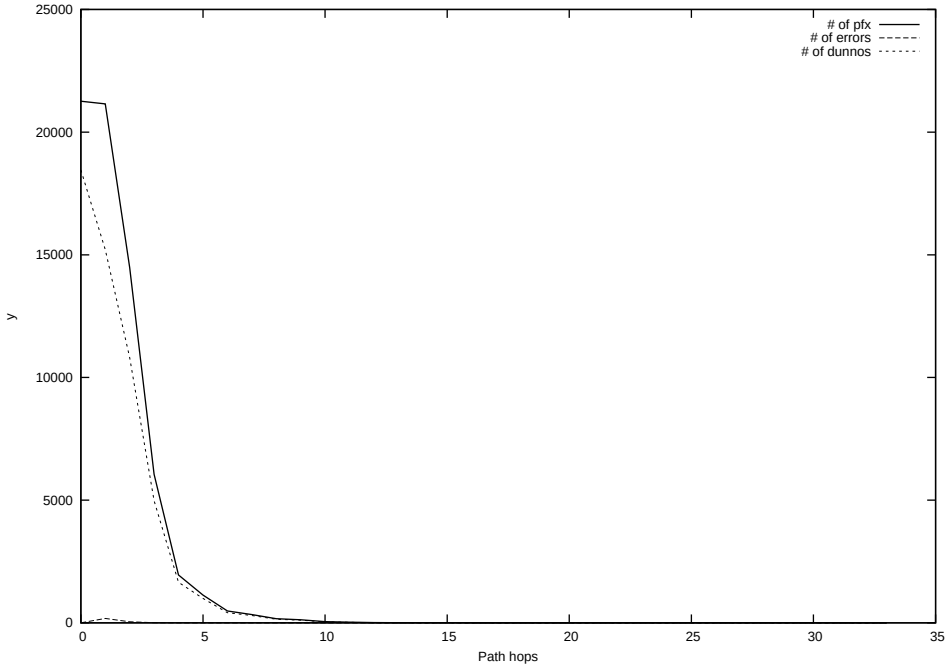
2014-12-13



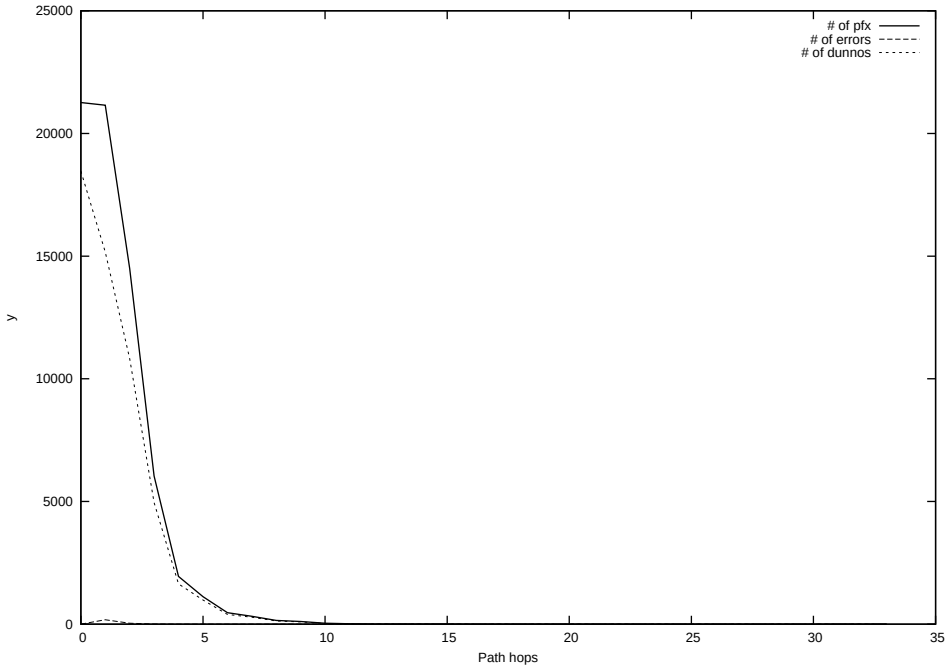
2014-12-14



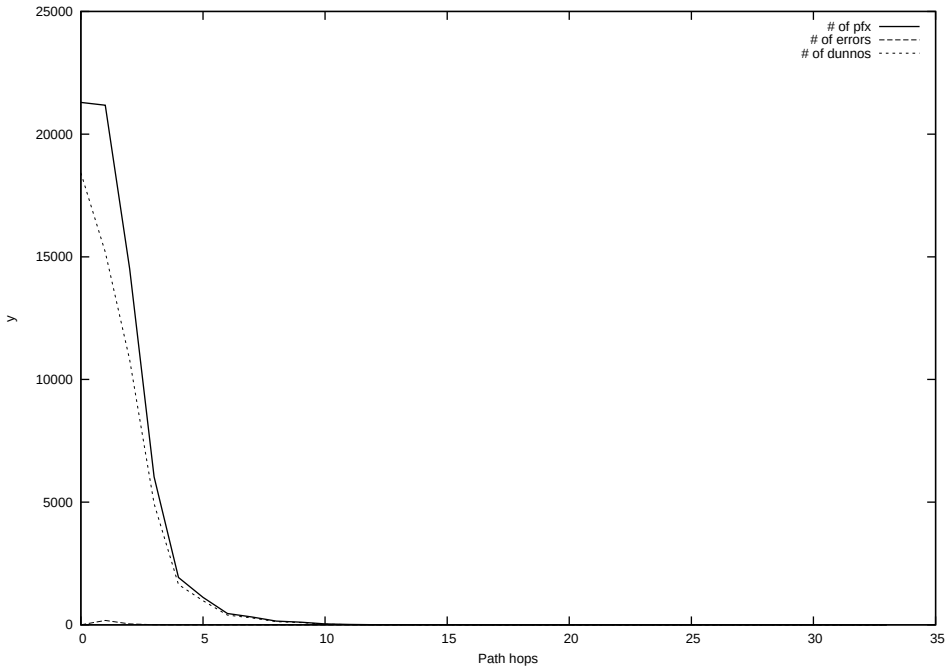
2014-12-15



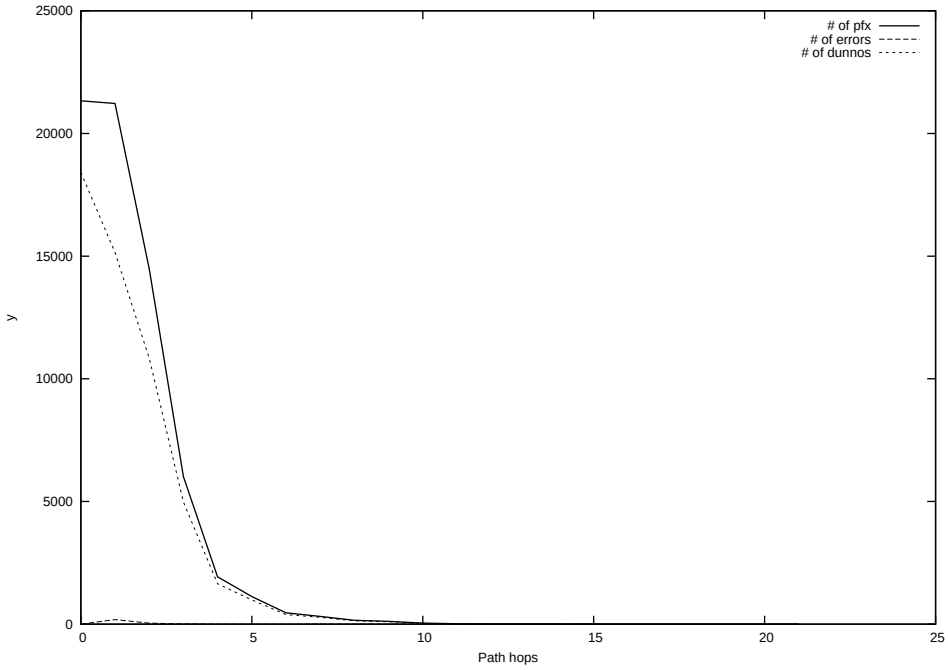
2014-12-16



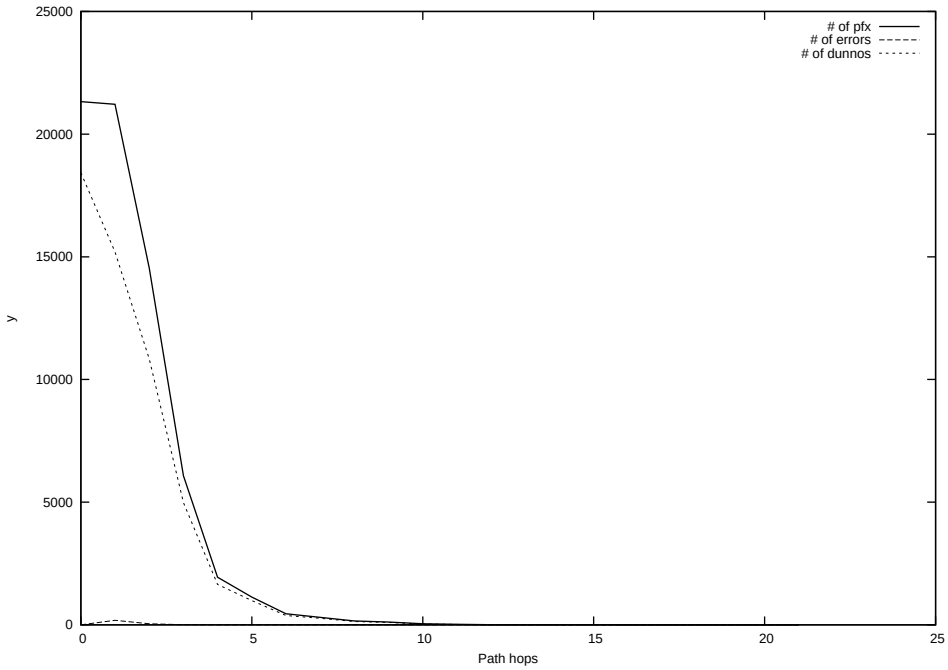
2014-12-17



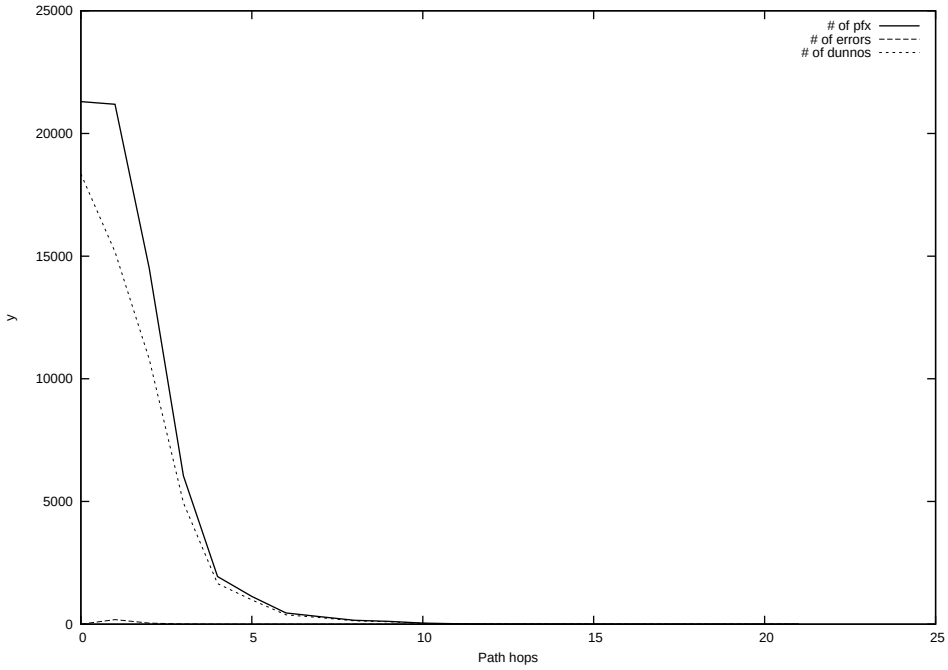
2014-12-18



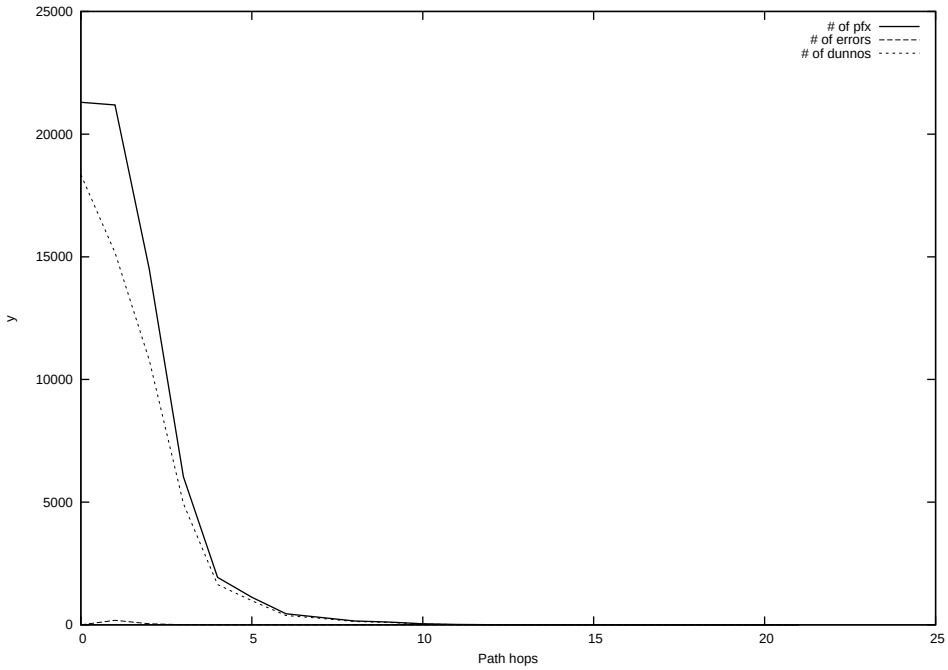
2014-12-19



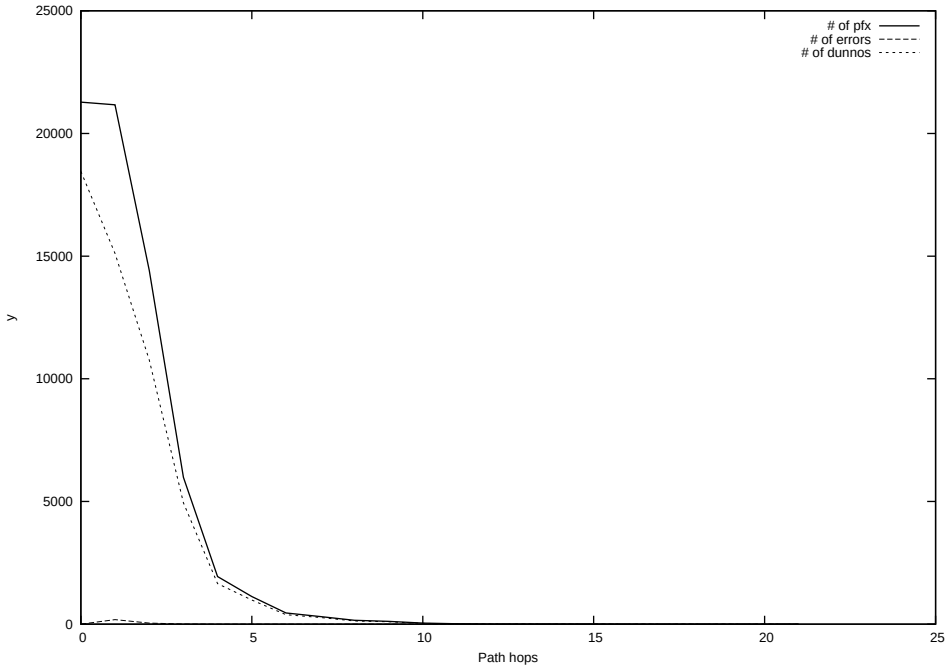
2014-12-20



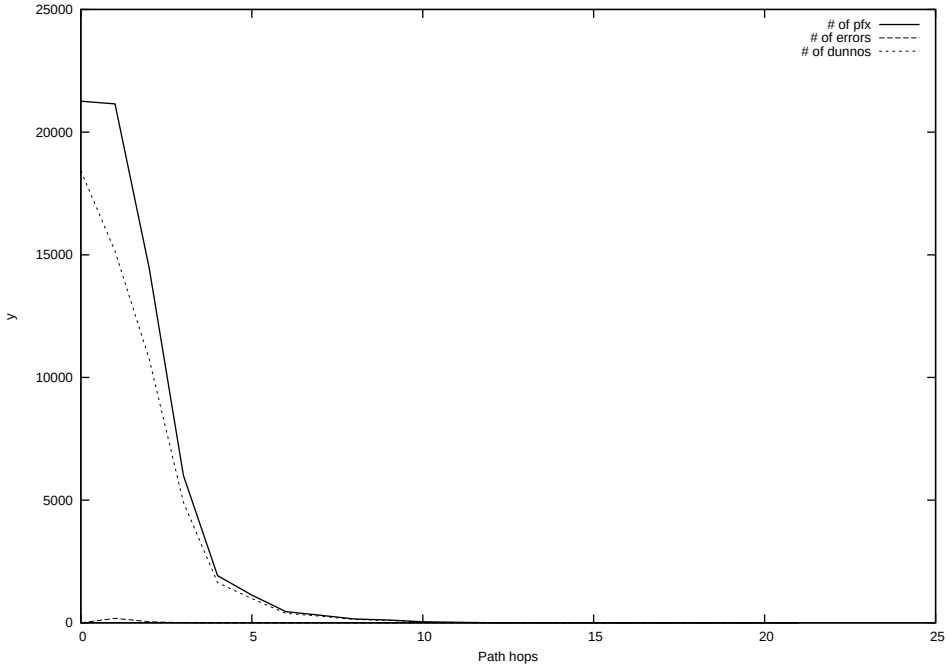
2014-12-21



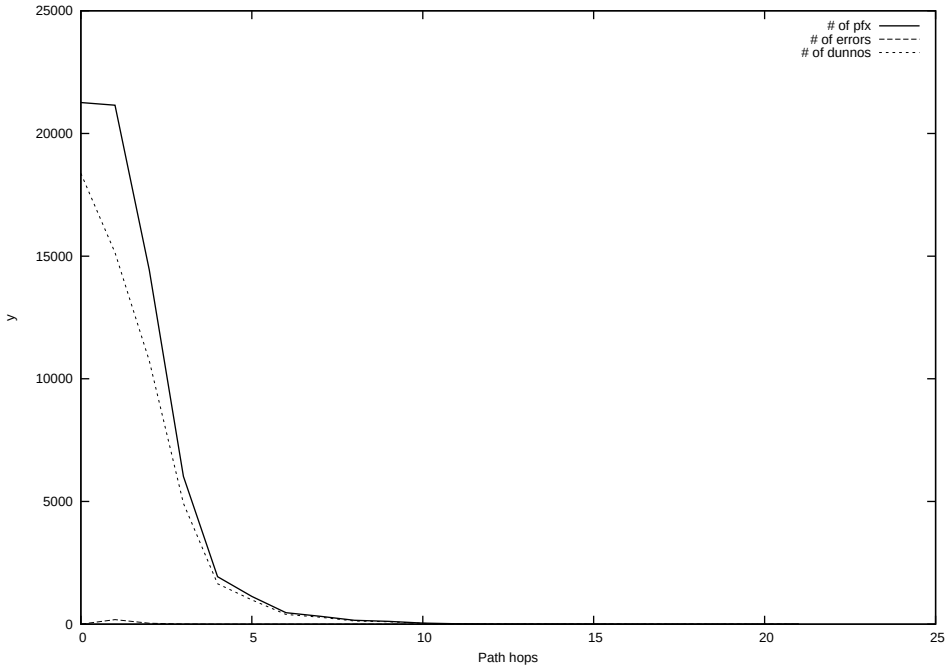
2014-12-22



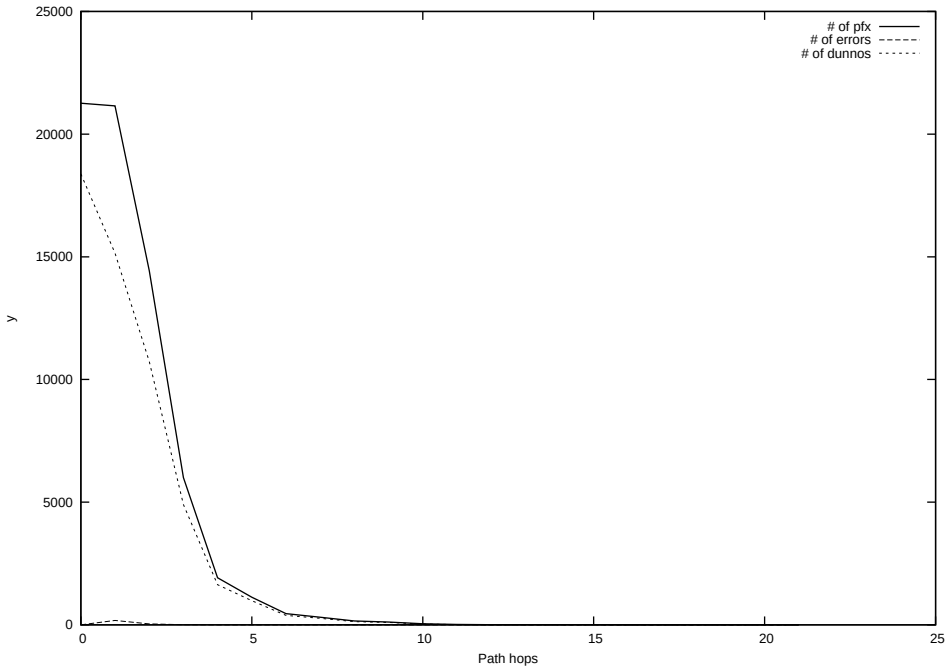
2014-12-23



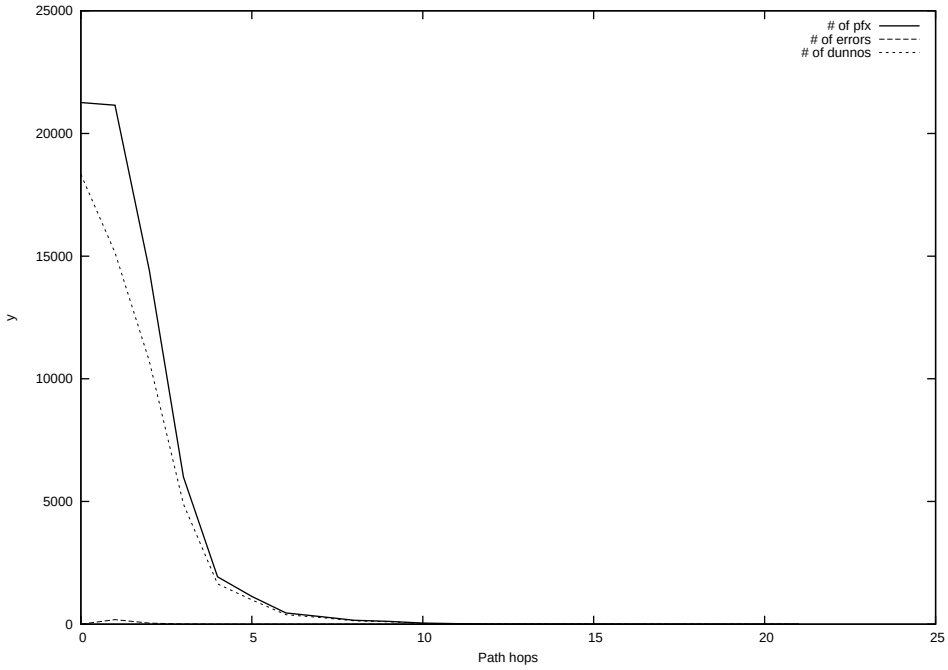
2014-12-24



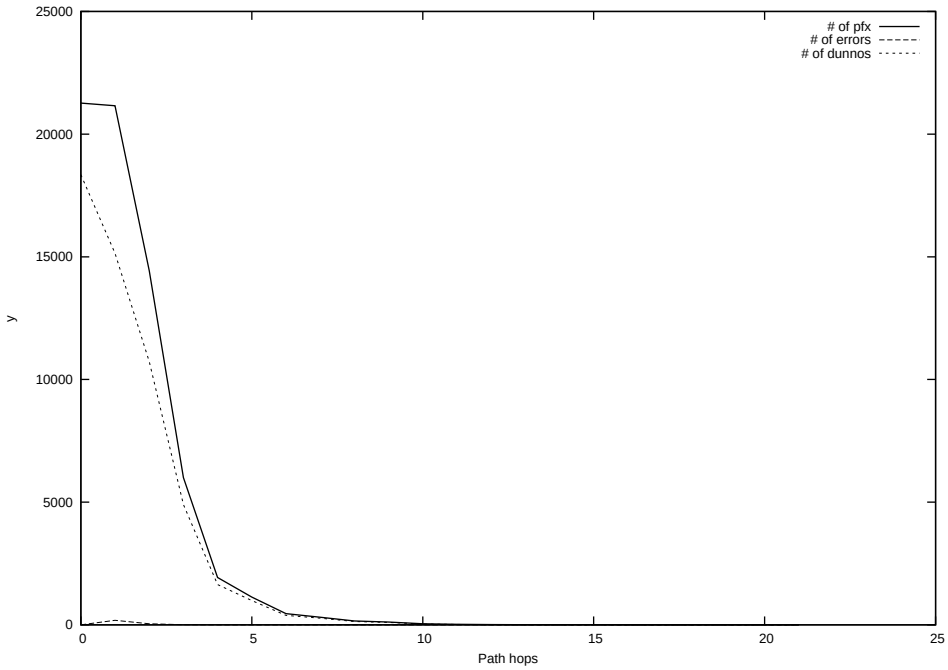
2014-12-25



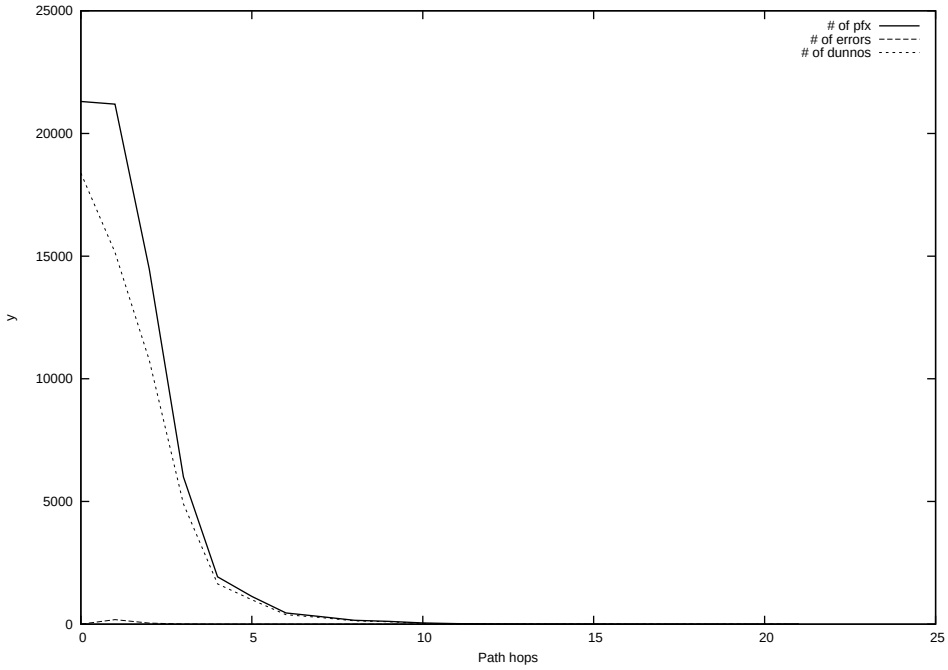
2014-12-26



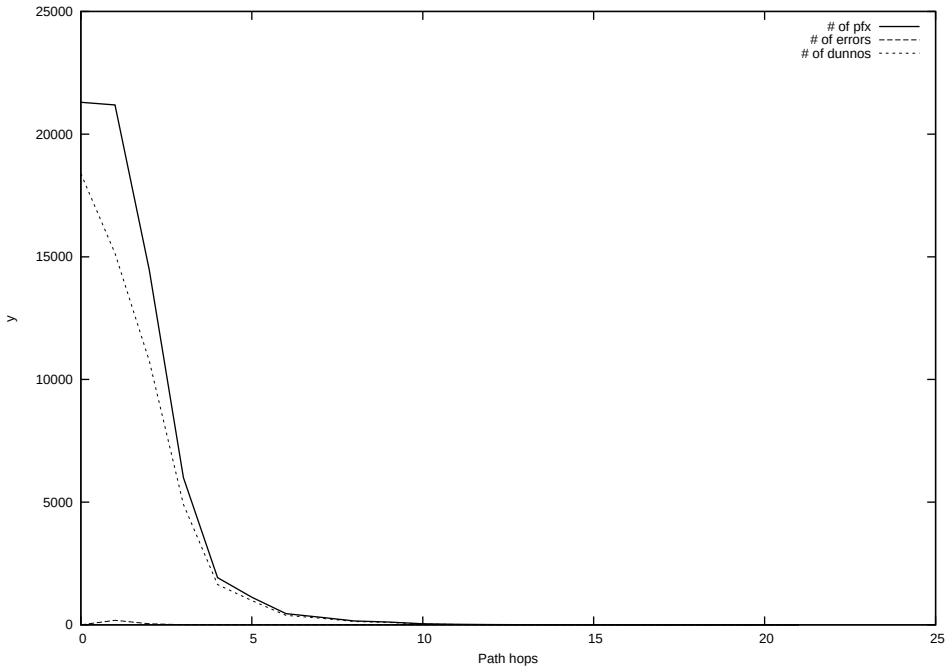
2014-12-27



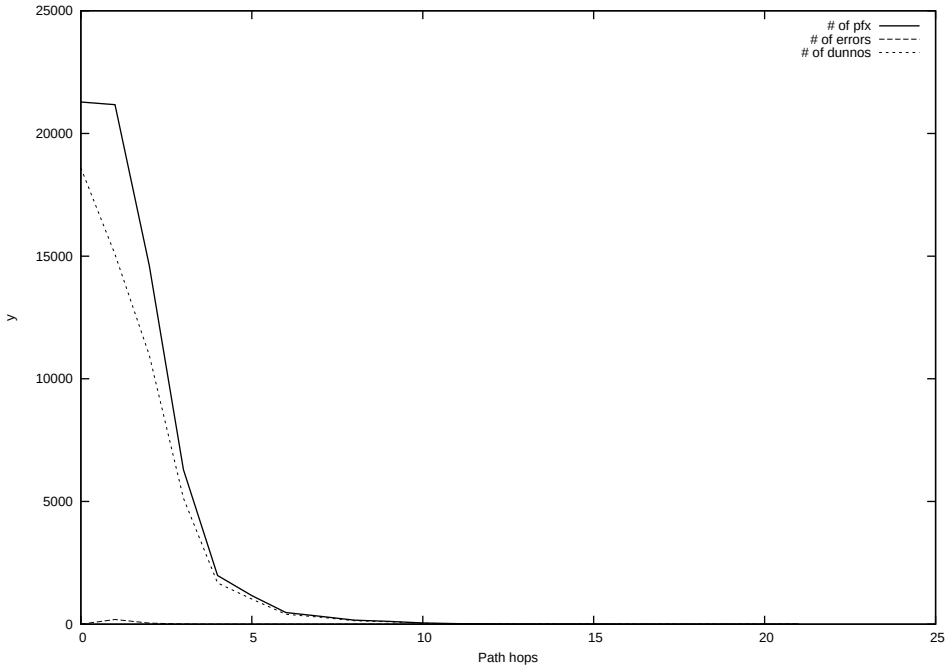
2014-12-28



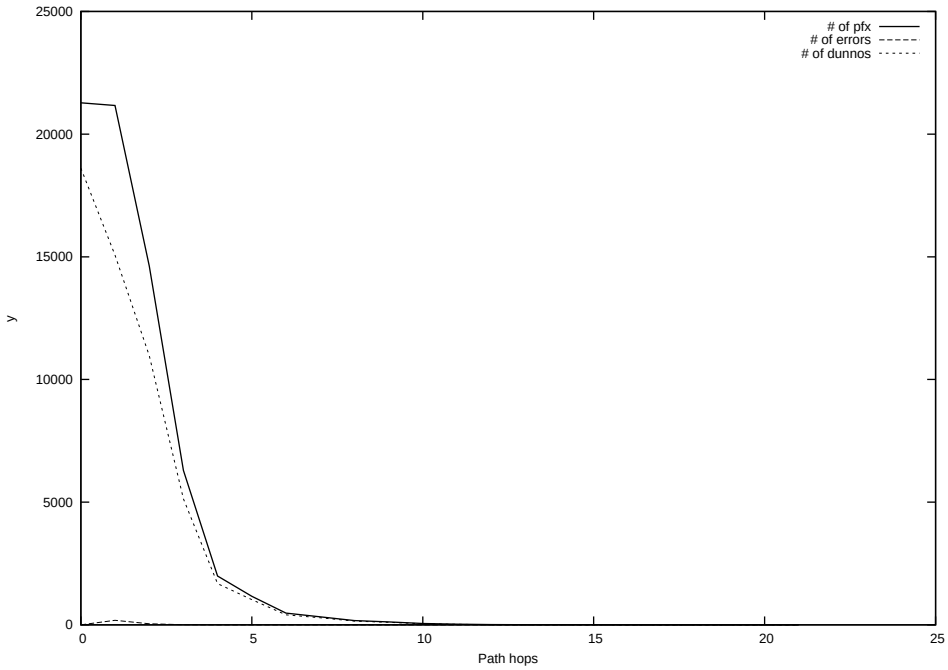
2014-12-29



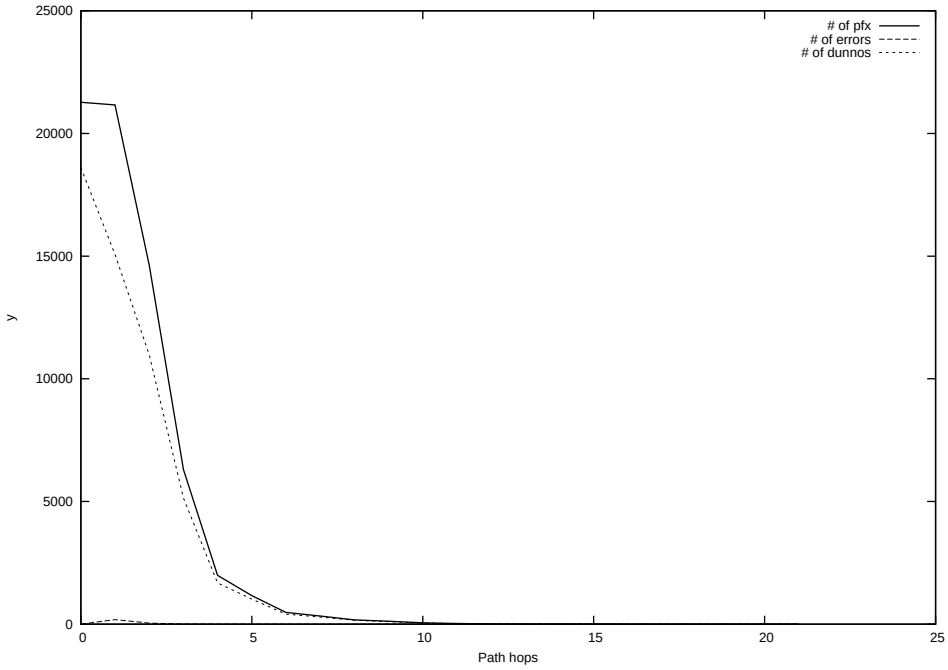
2014-12-30



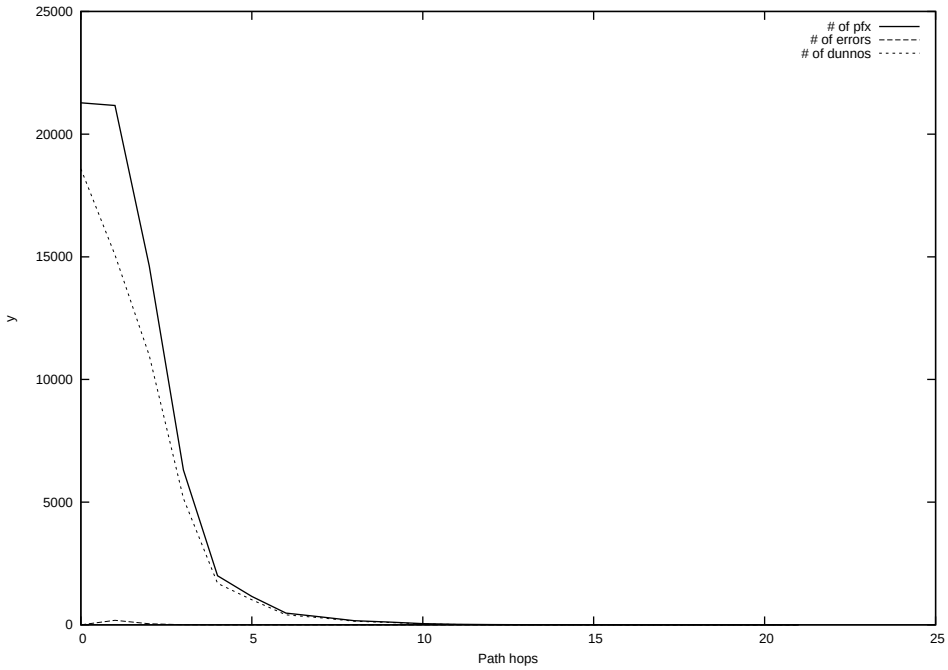
2014-12-31



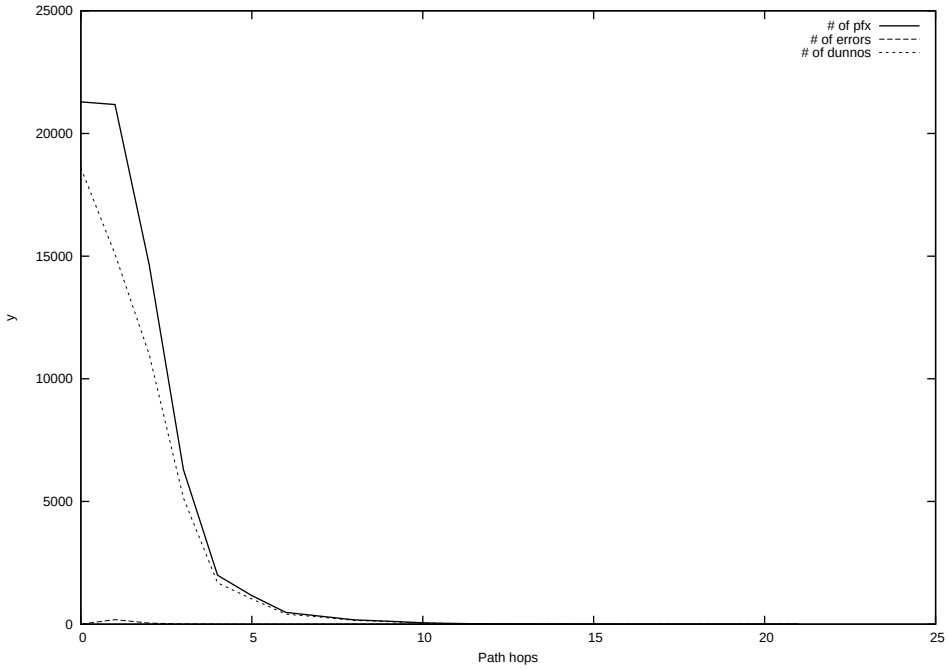
2015-01-01



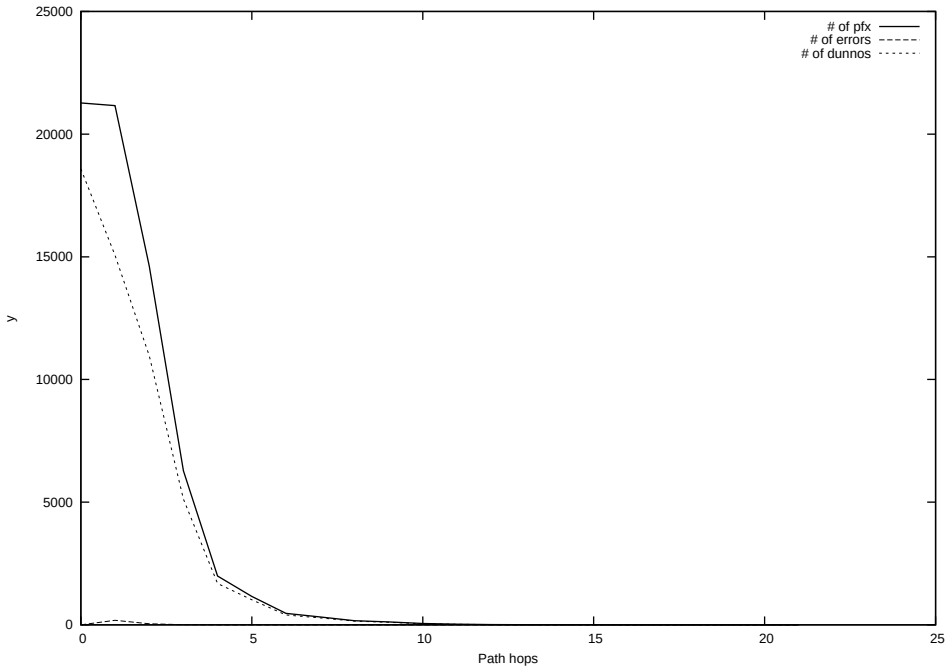
2015-01-02



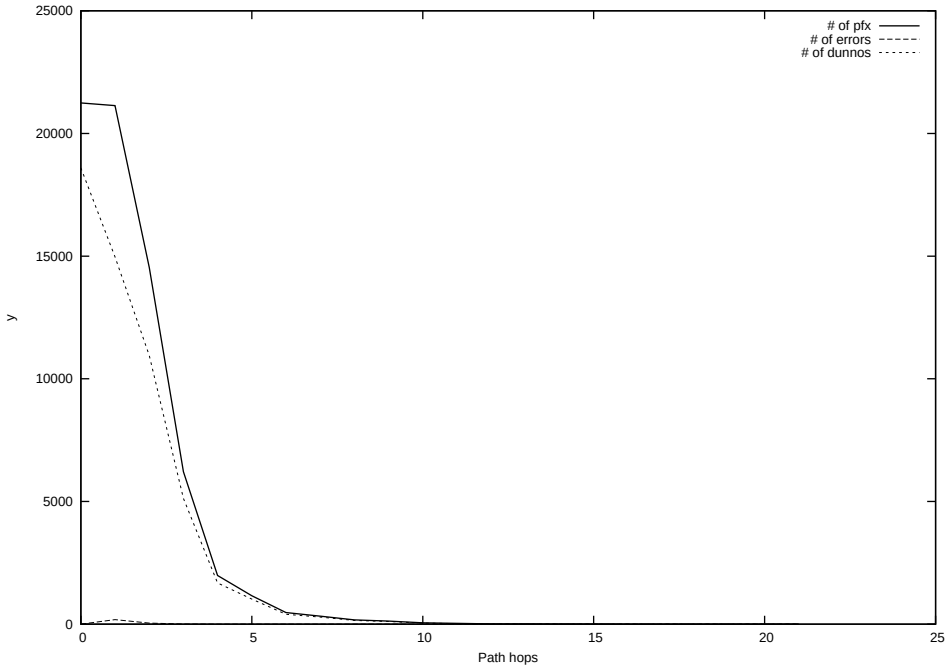
2015-01-03



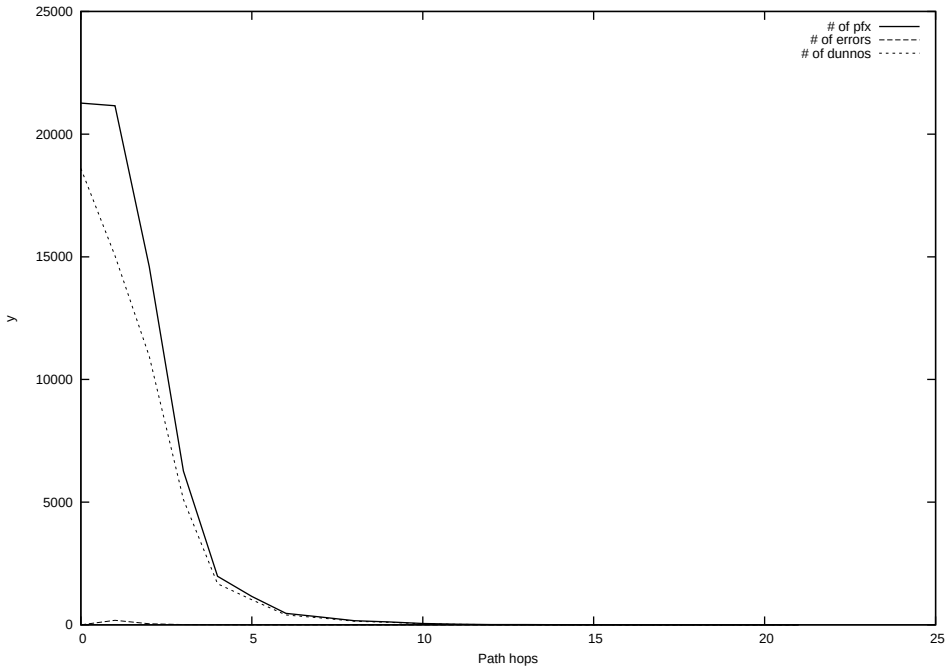
2015-01-04



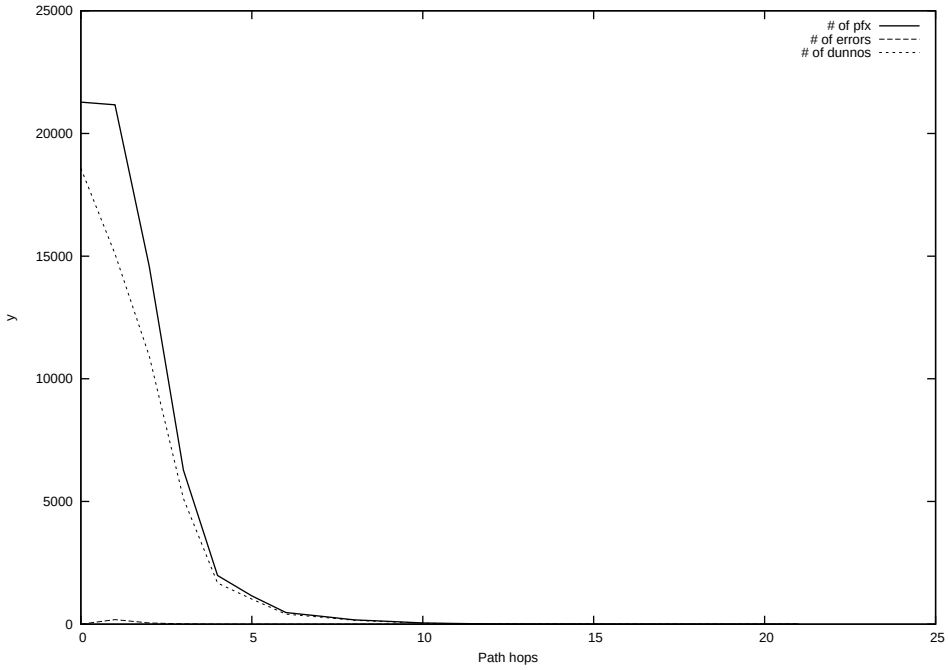
2015-01-05



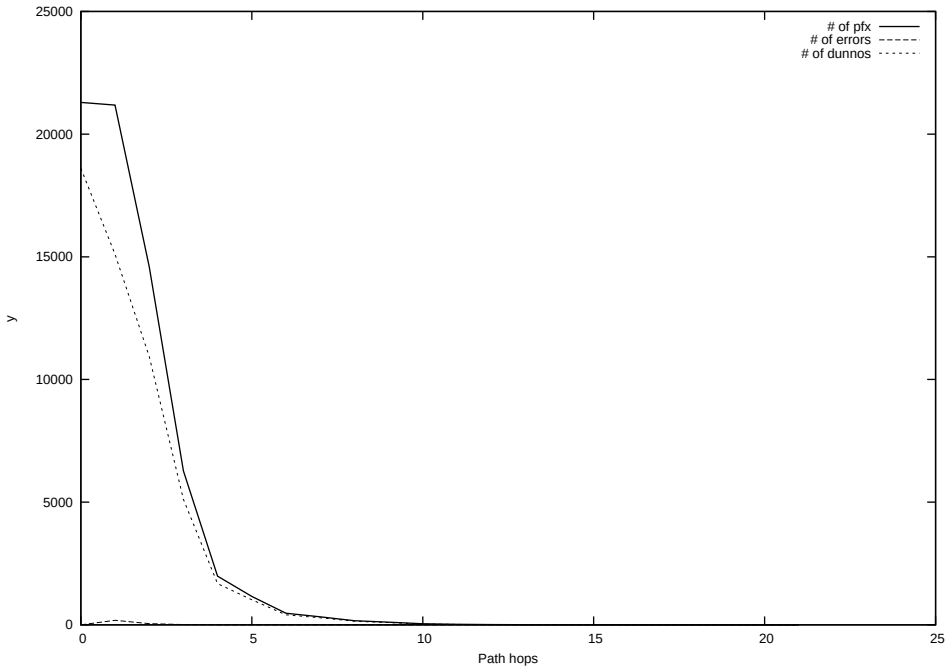
2015-01-06



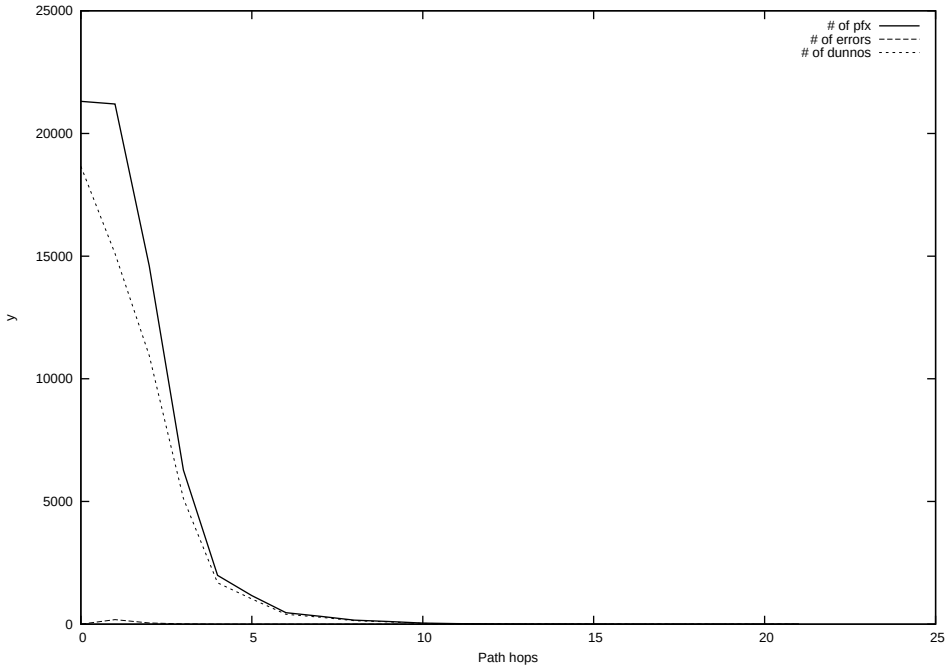
2015-01-07



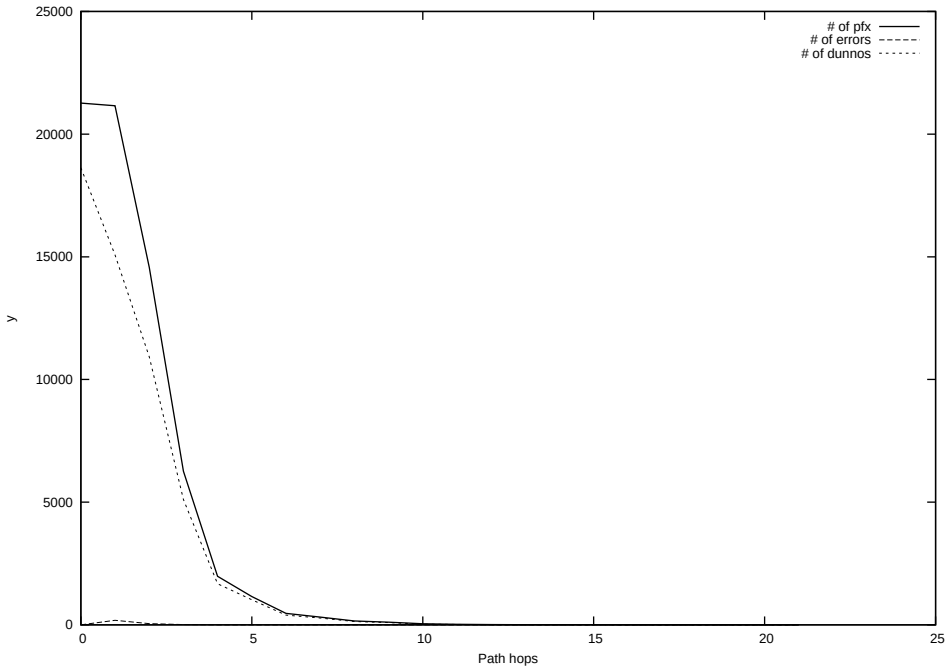
2015-01-08



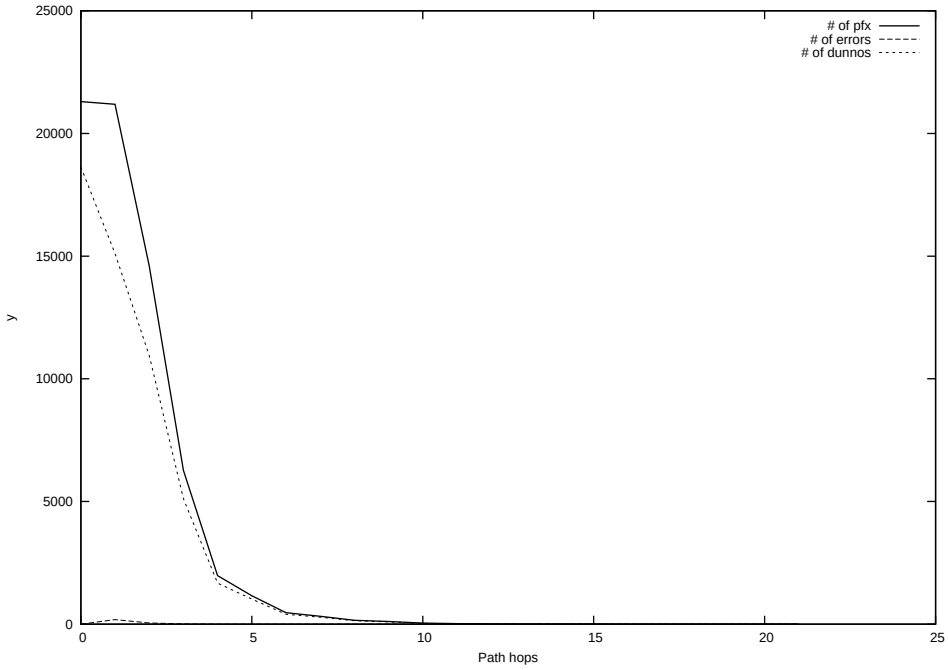
2015-01-09



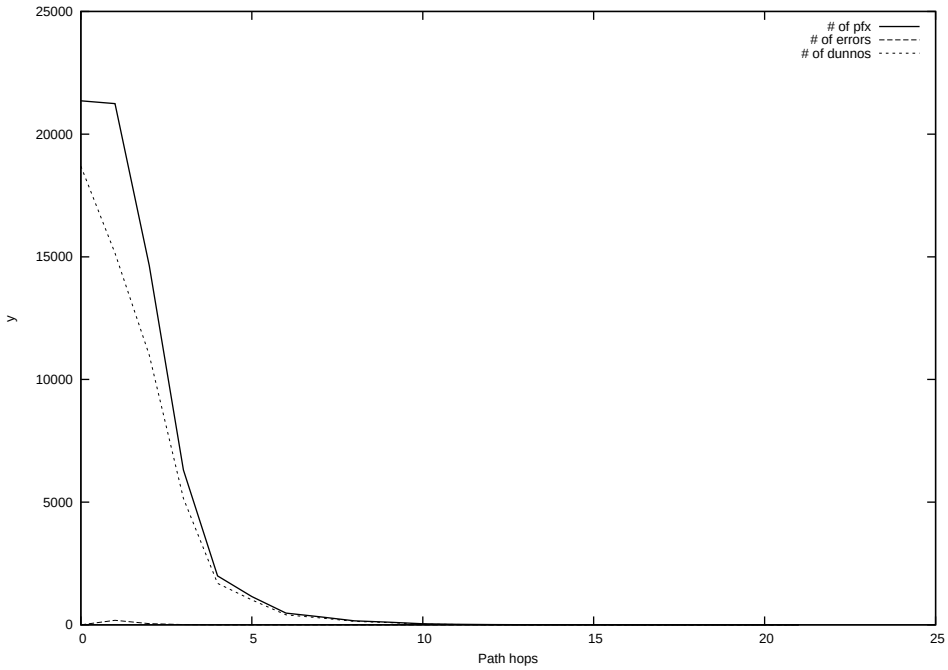
2015-01-10



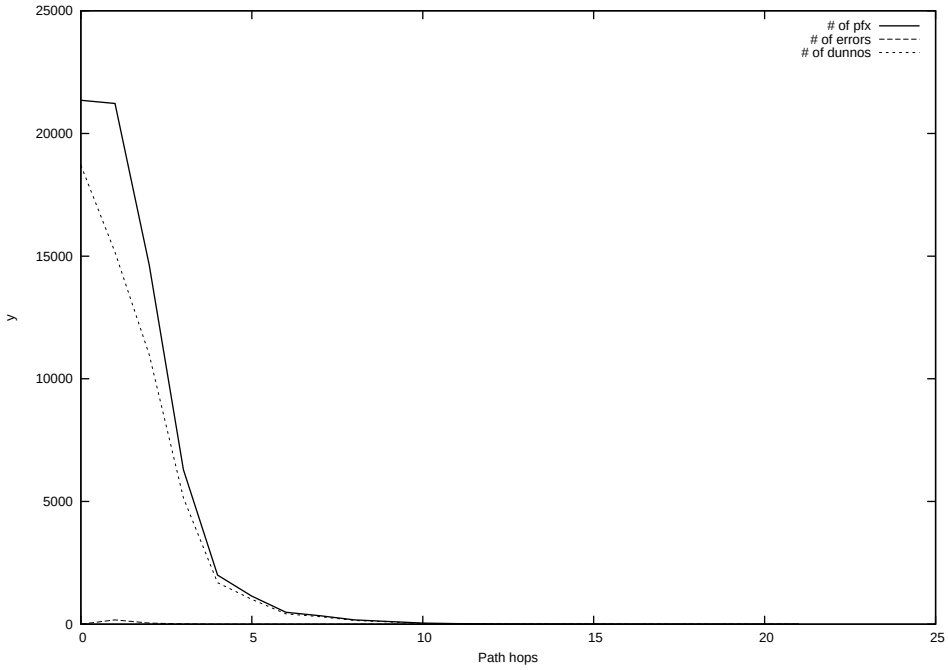
2015-01-11



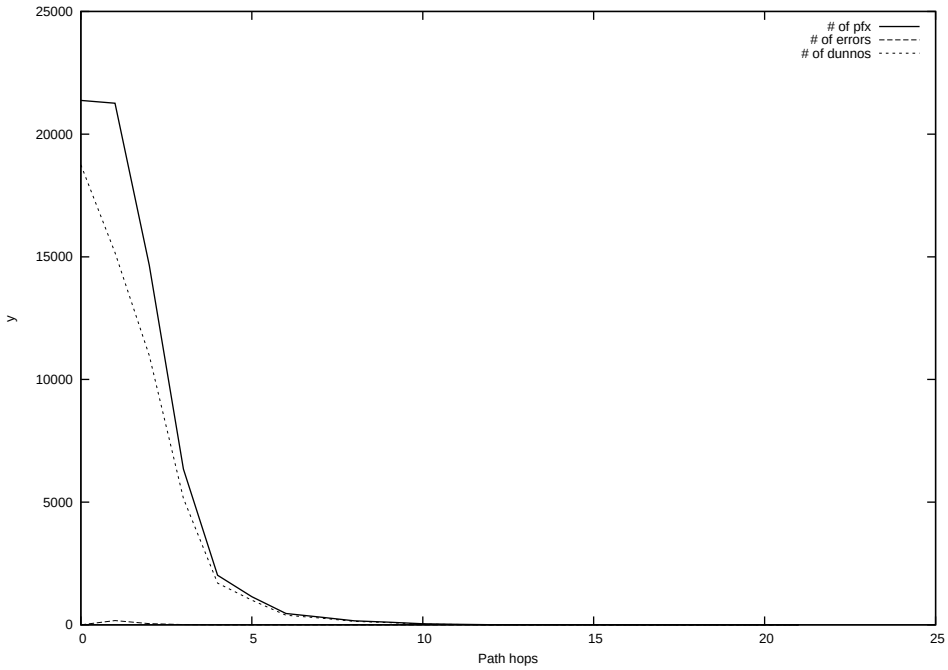
2015-01-12



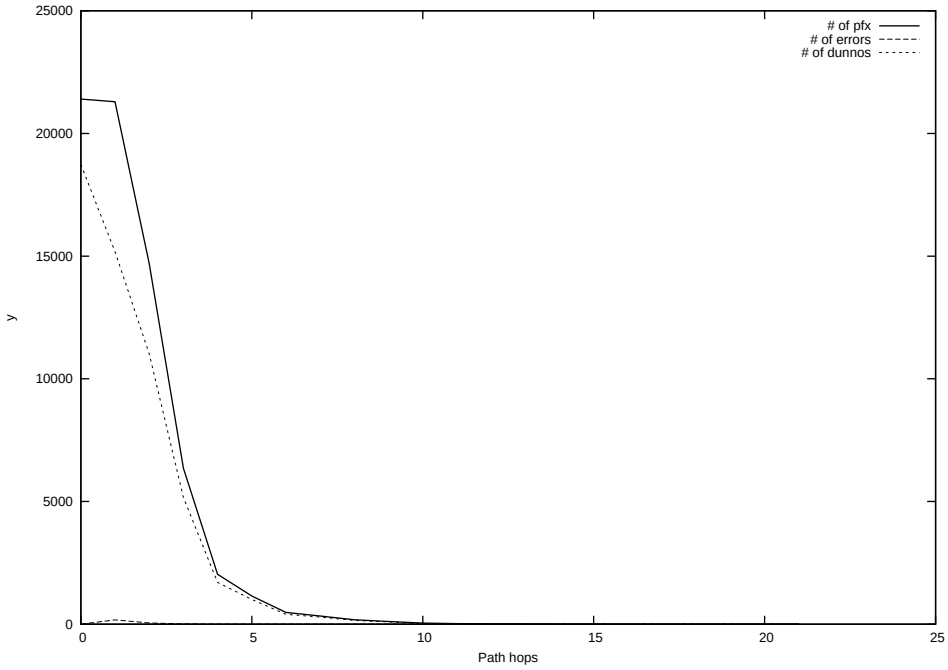
2015-01-13



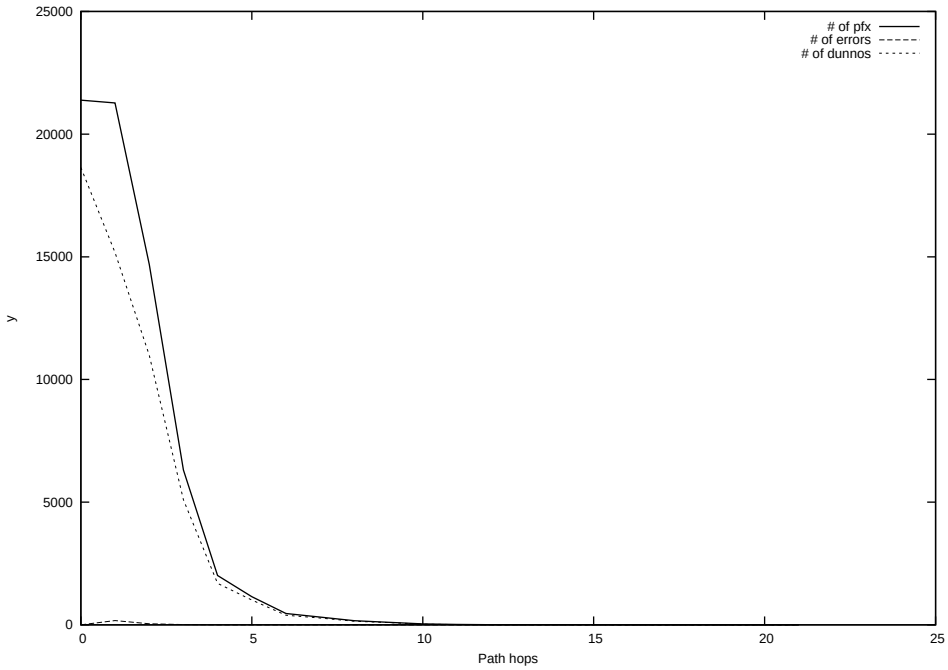
2015-01-14



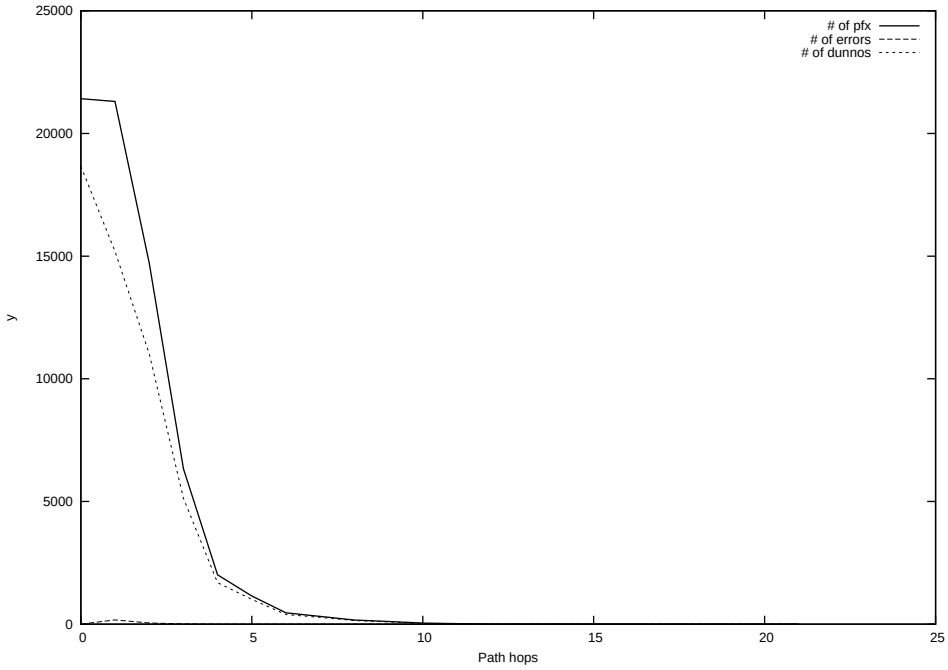
2015-01-15



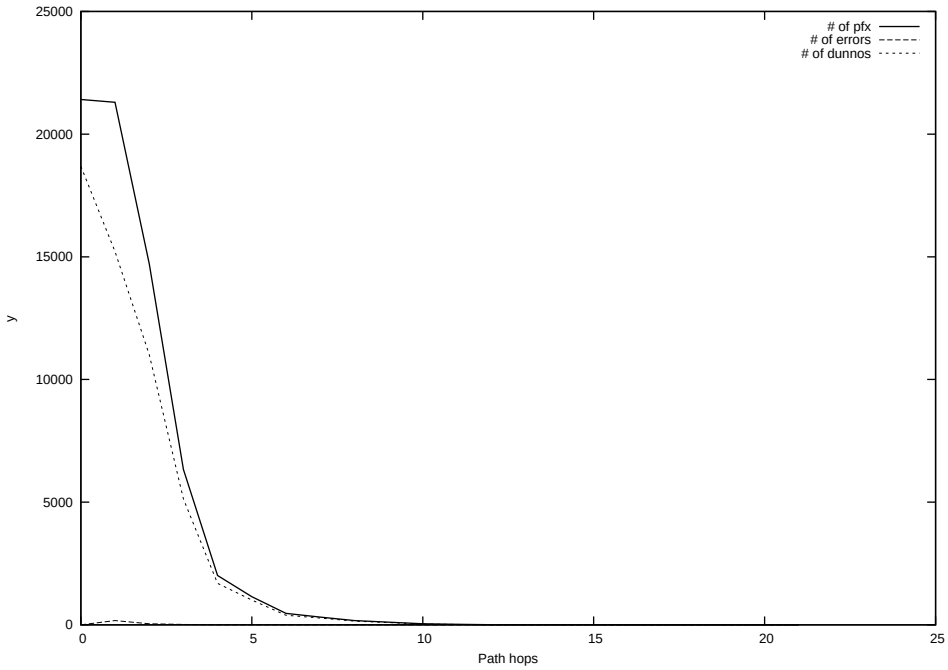
2015-01-16



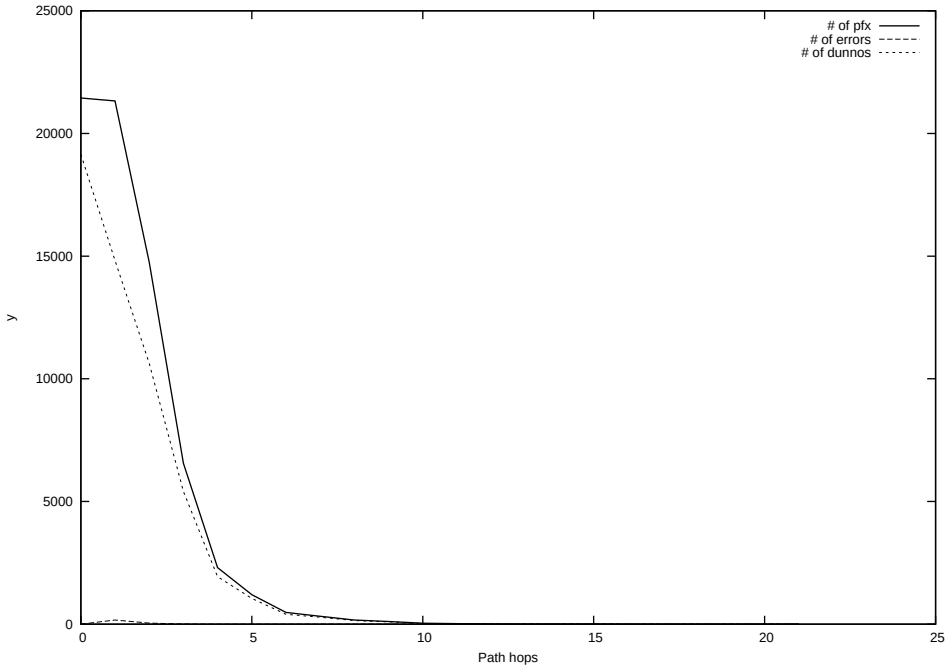
2015-01-17



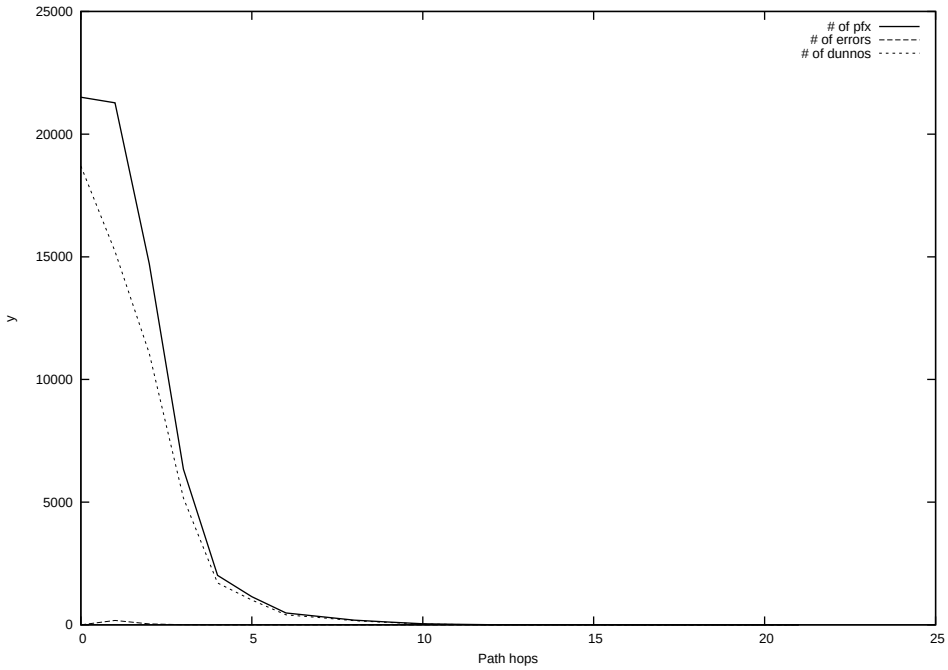
2015-01-18



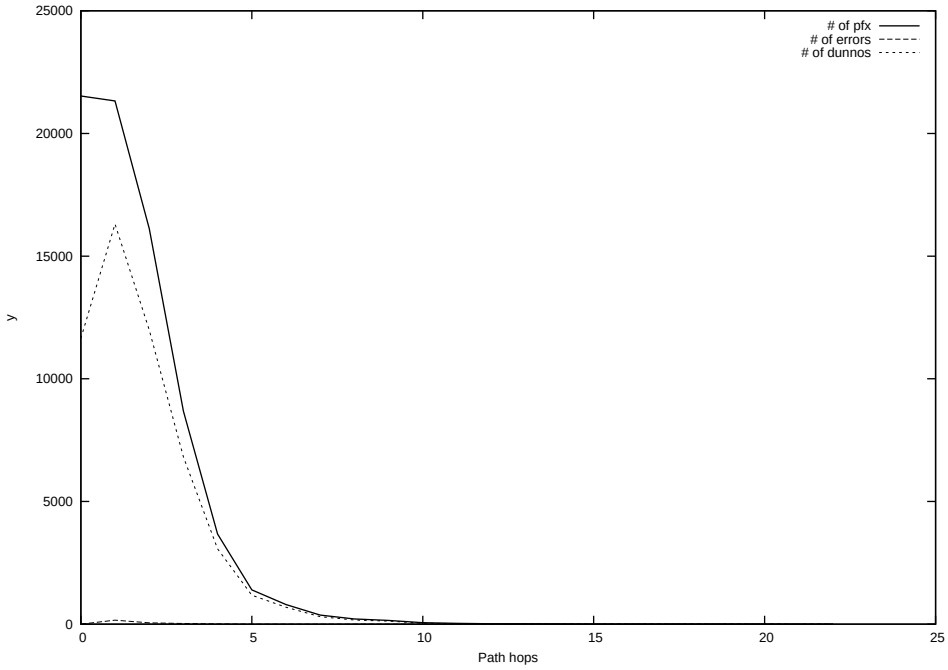
2015-01-19



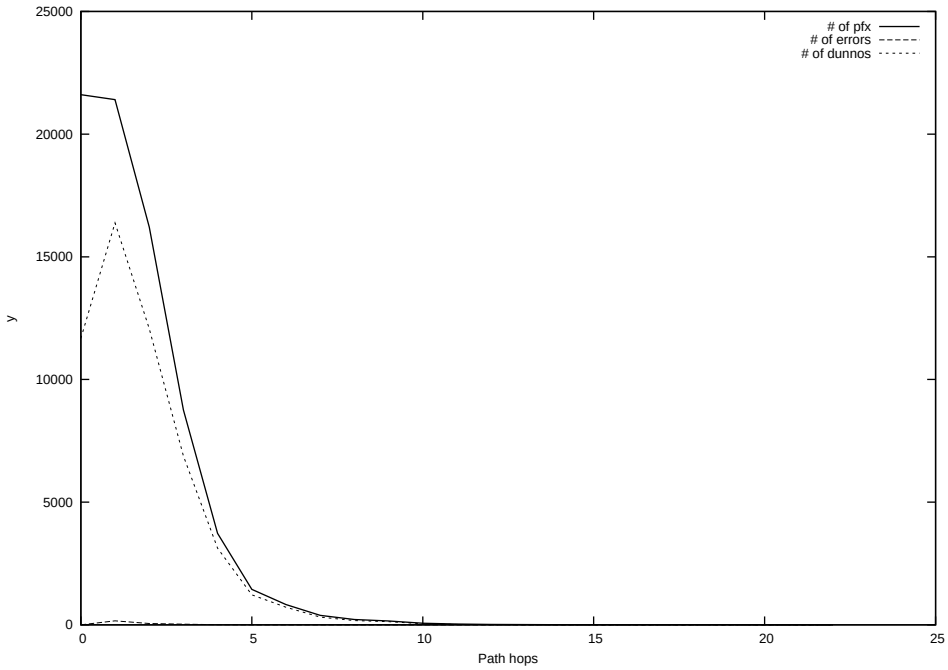
2015-01-20



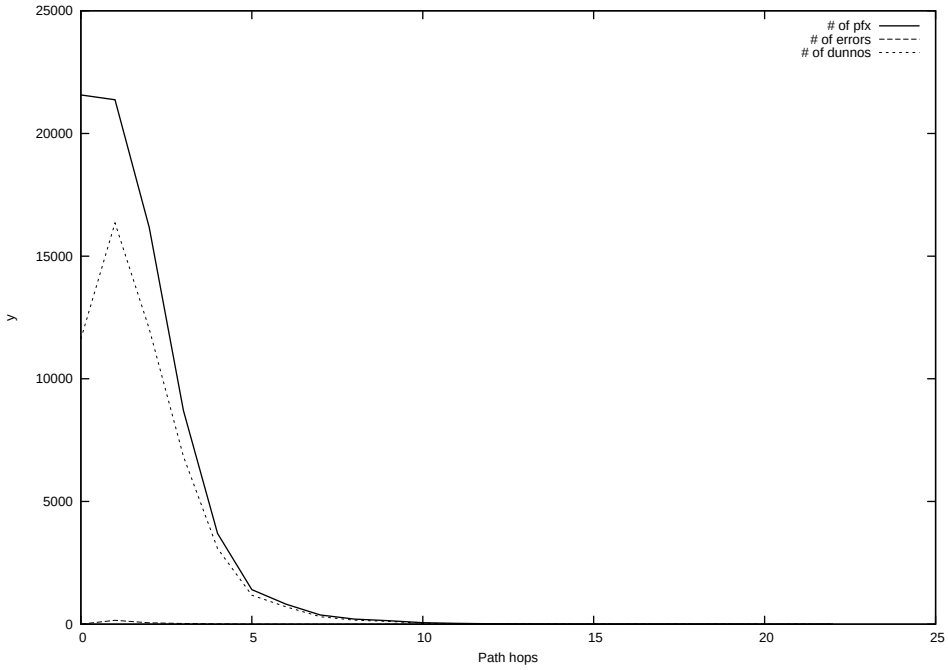
2015-01-21



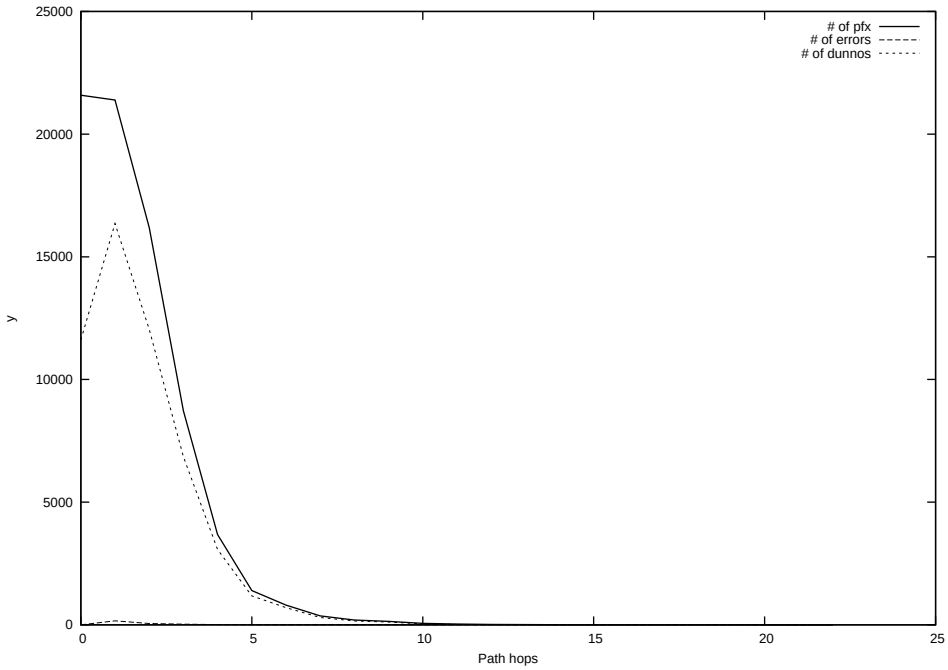
2015-01-22



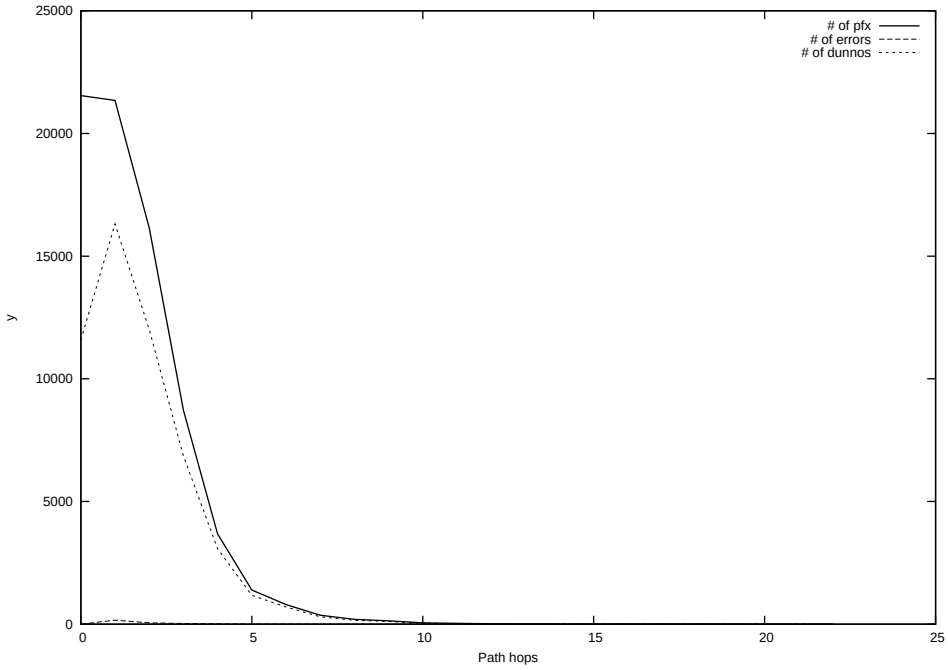
2015-01-23



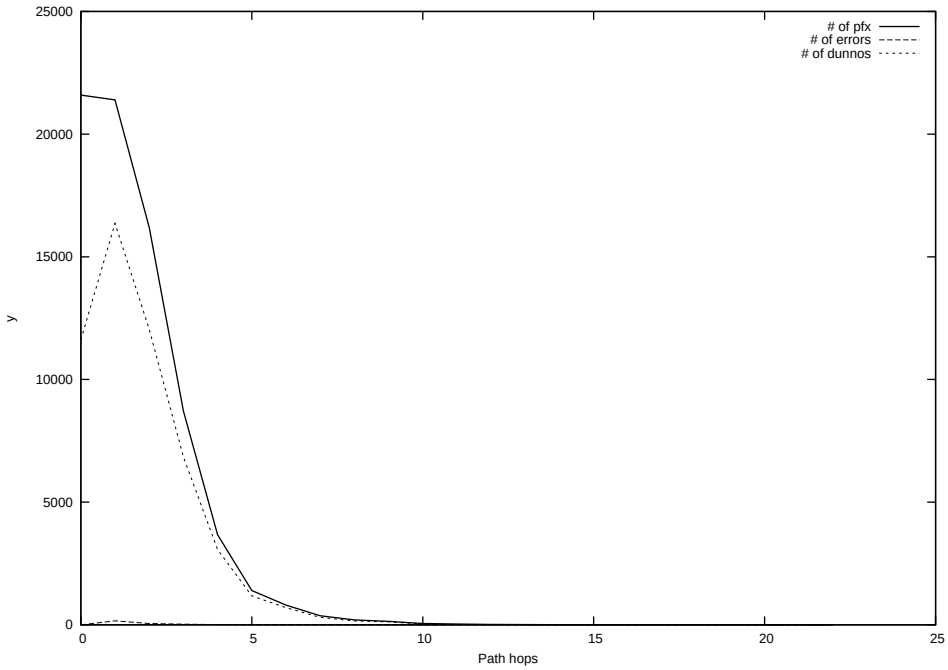
2015-01-24



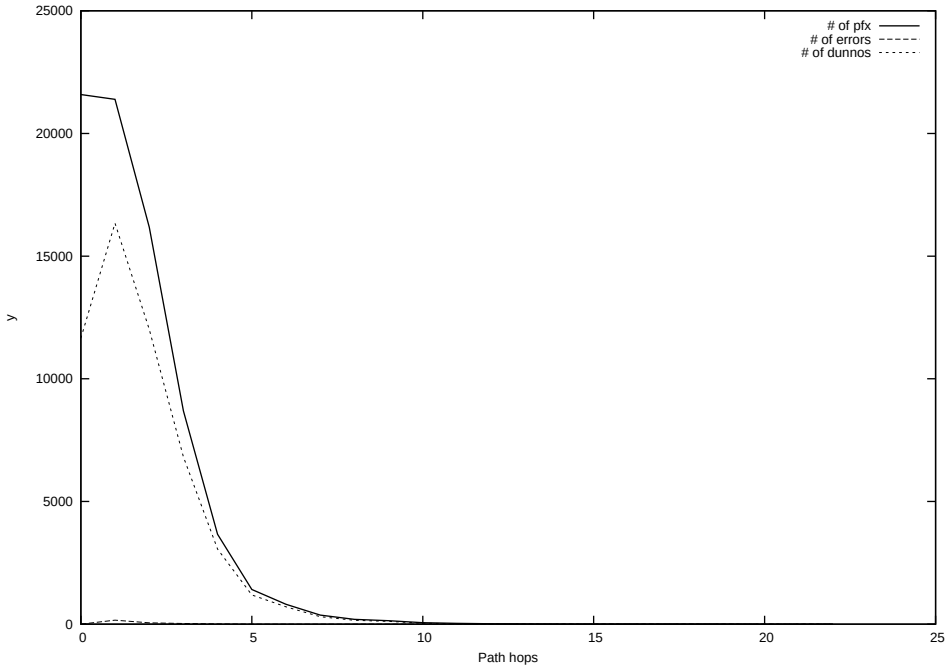
2015-01-25



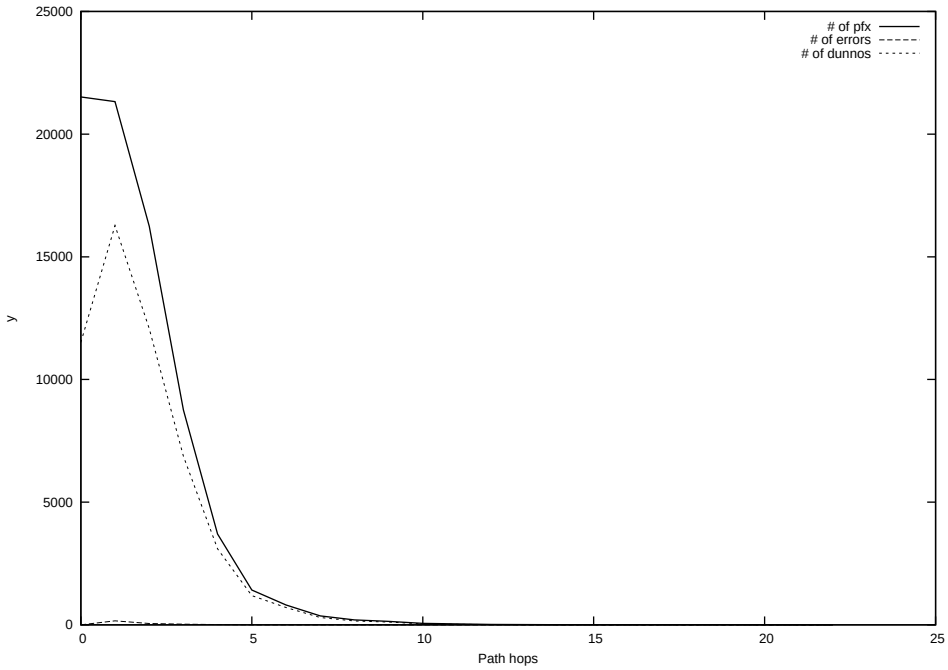
2015-01-26



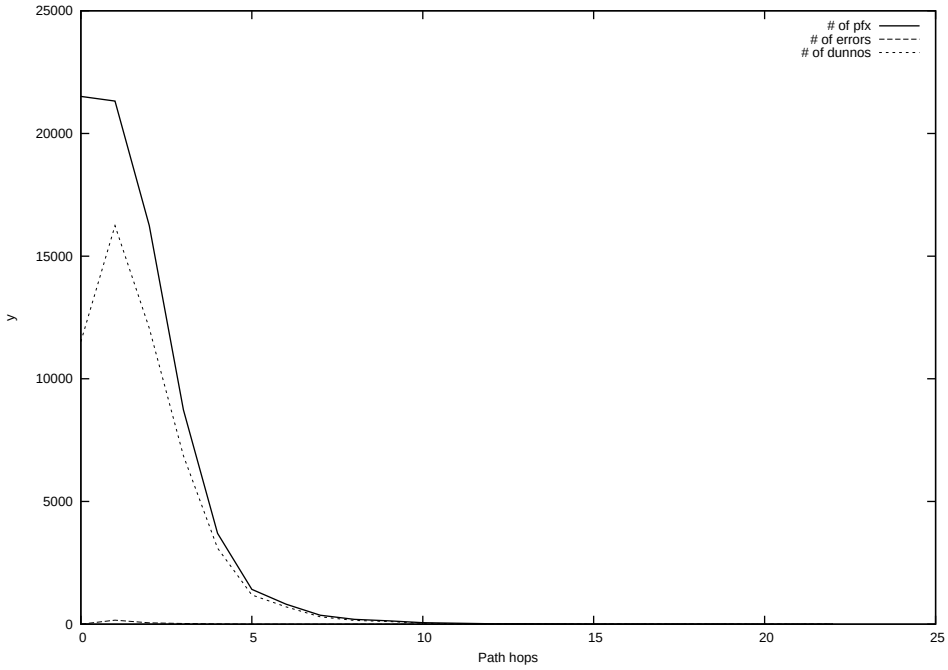
2015-01-27



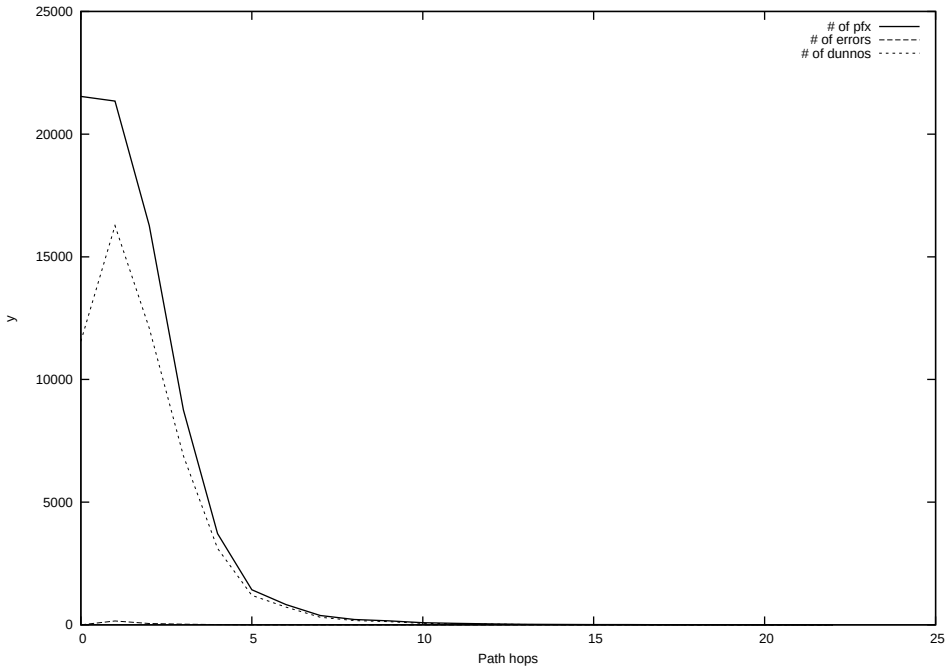
2015-01-28



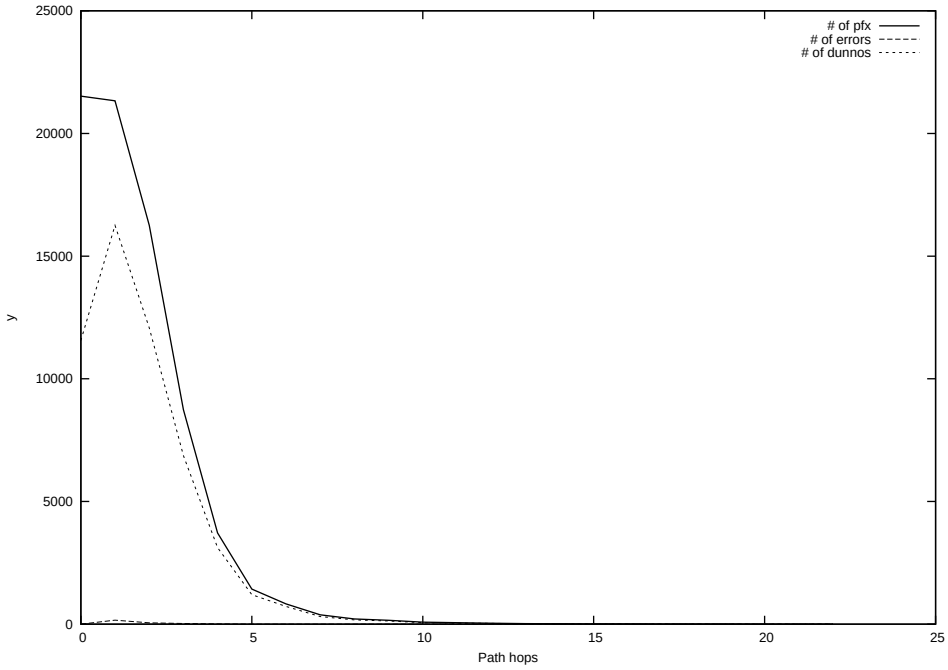
2015-01-29



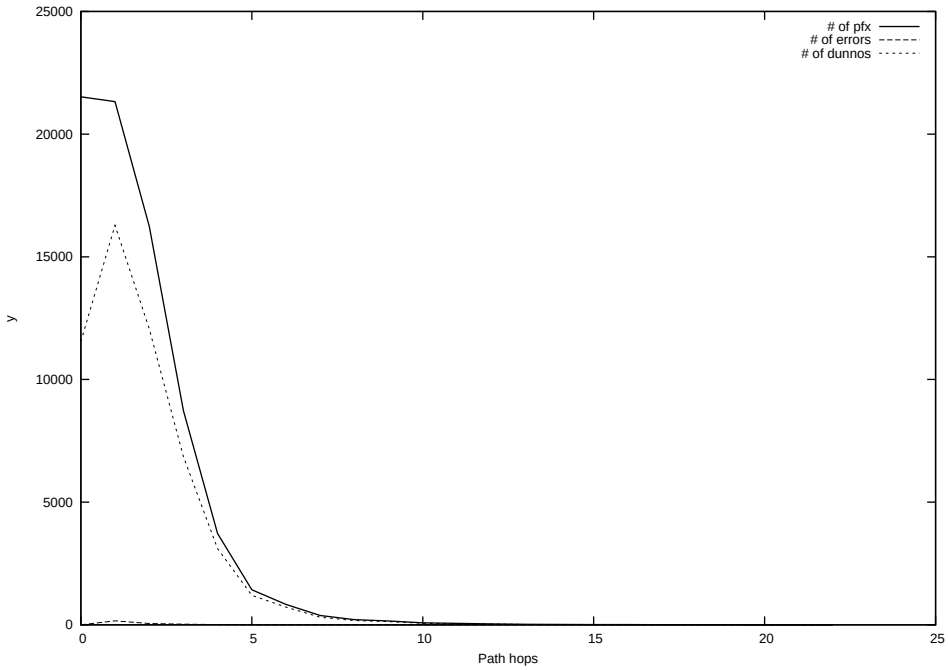
2015-01-30



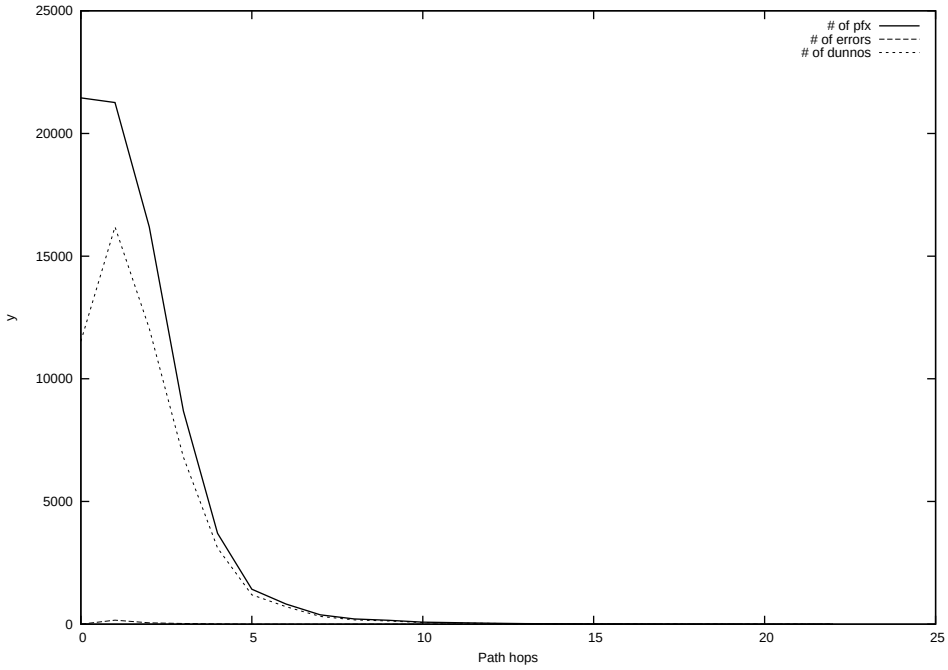
2015-01-31



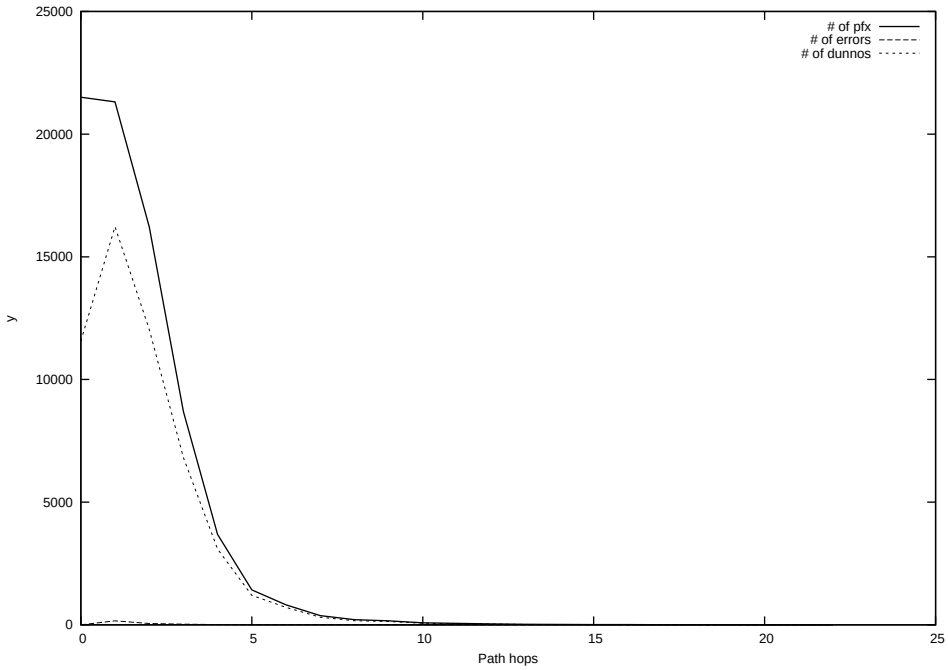
2015-02-01



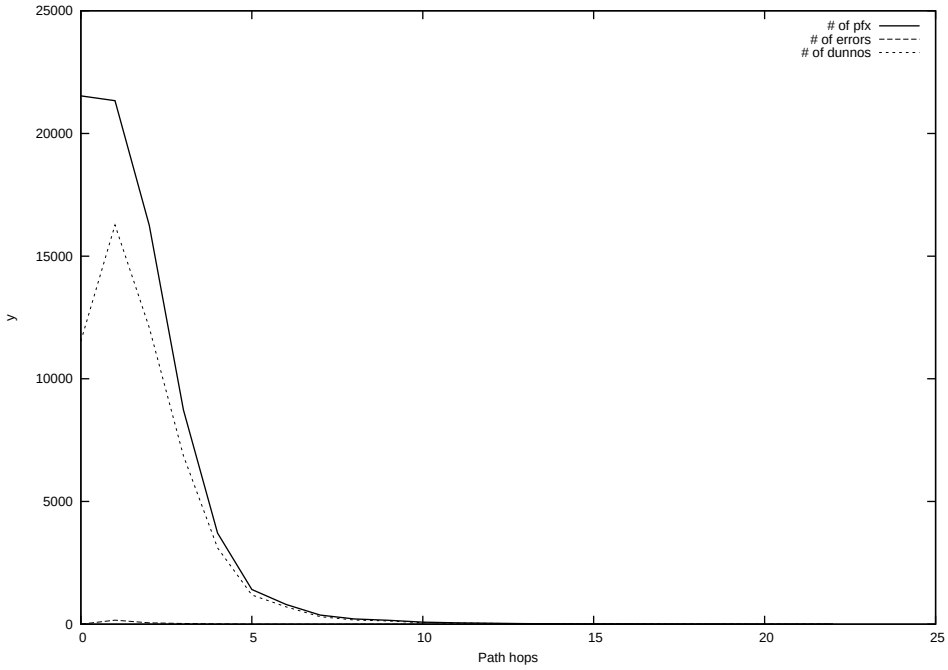
2015-02-02



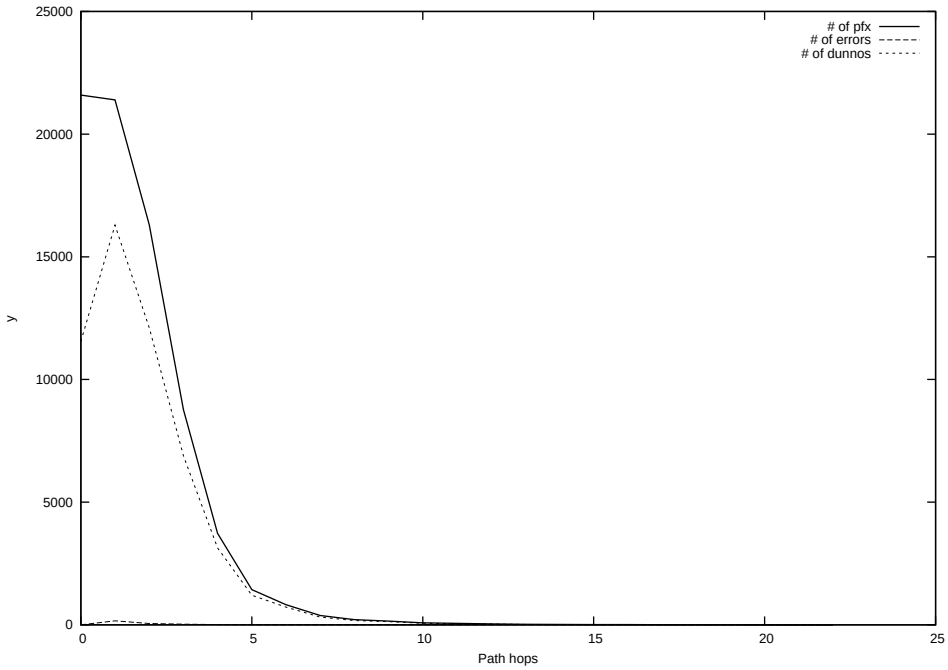
2015-02-03



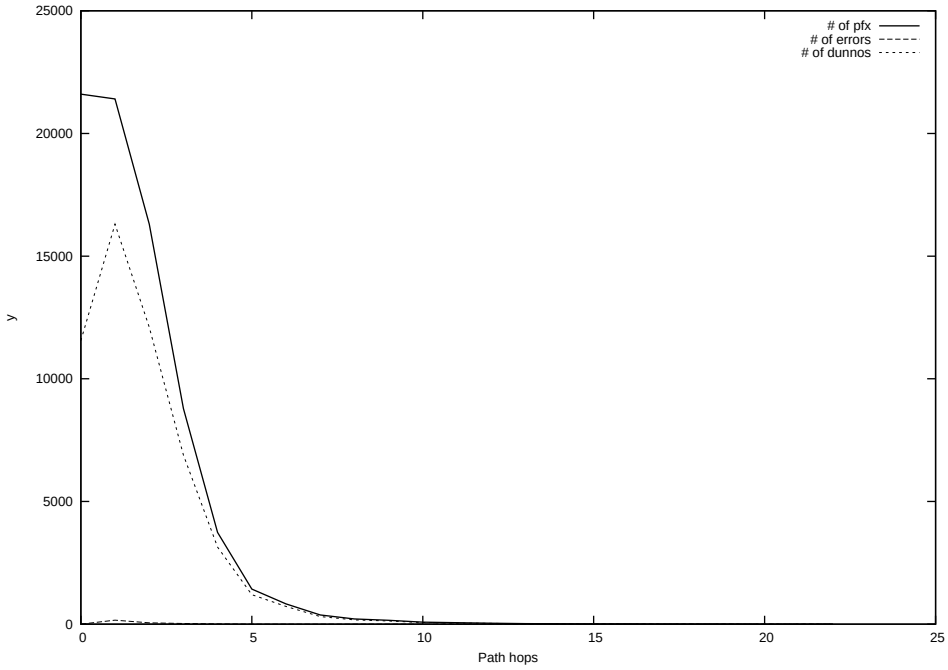
2015-02-04



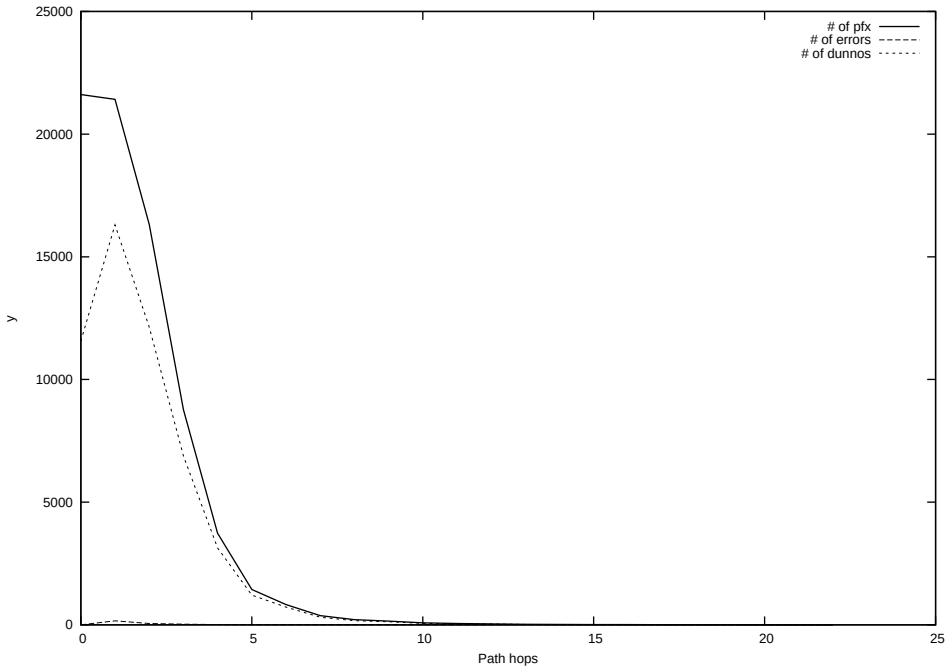
2015-02-05



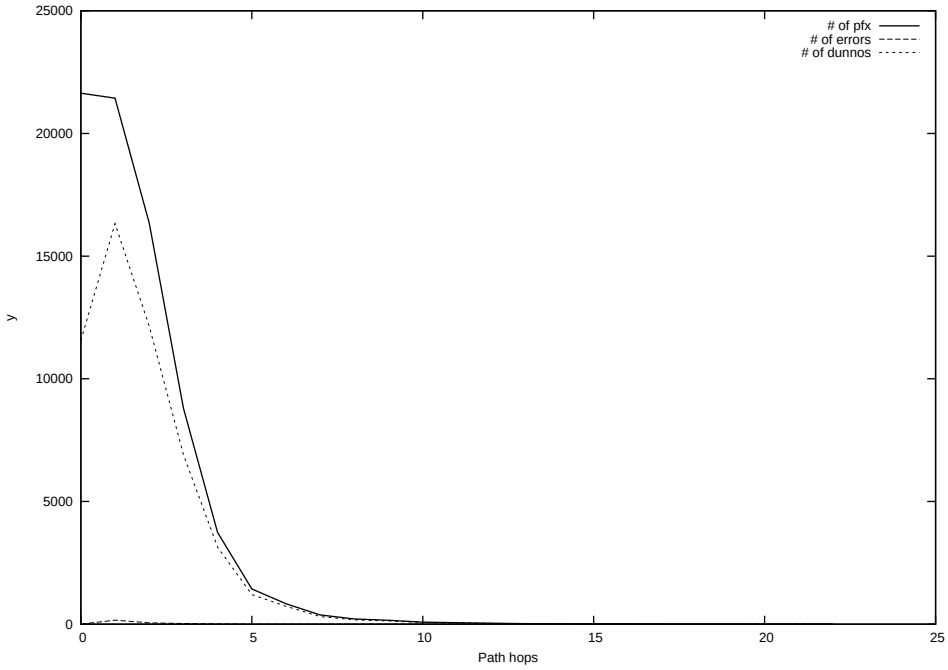
2015-02-06



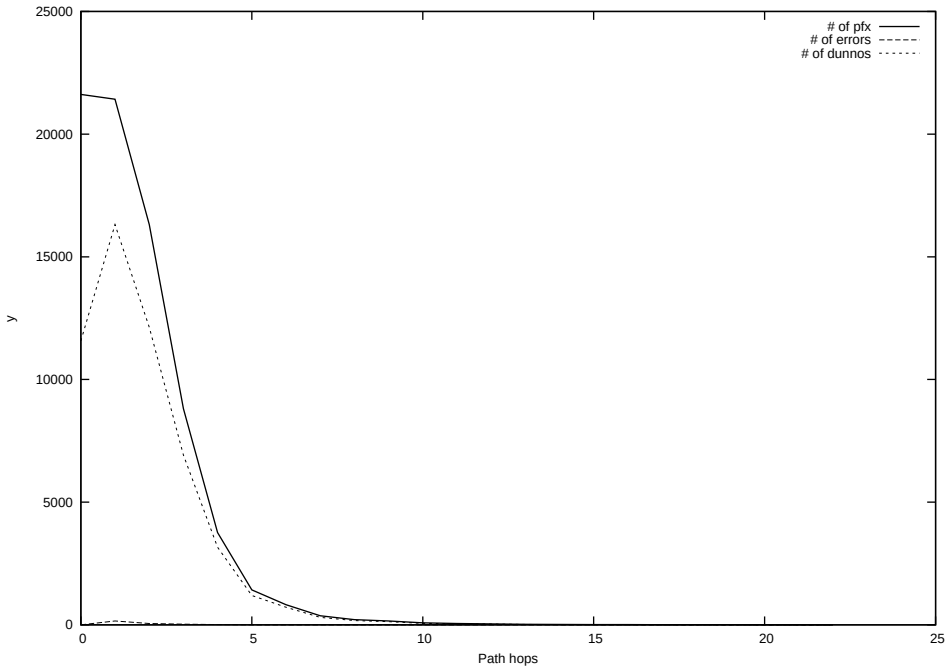
2015-02-07



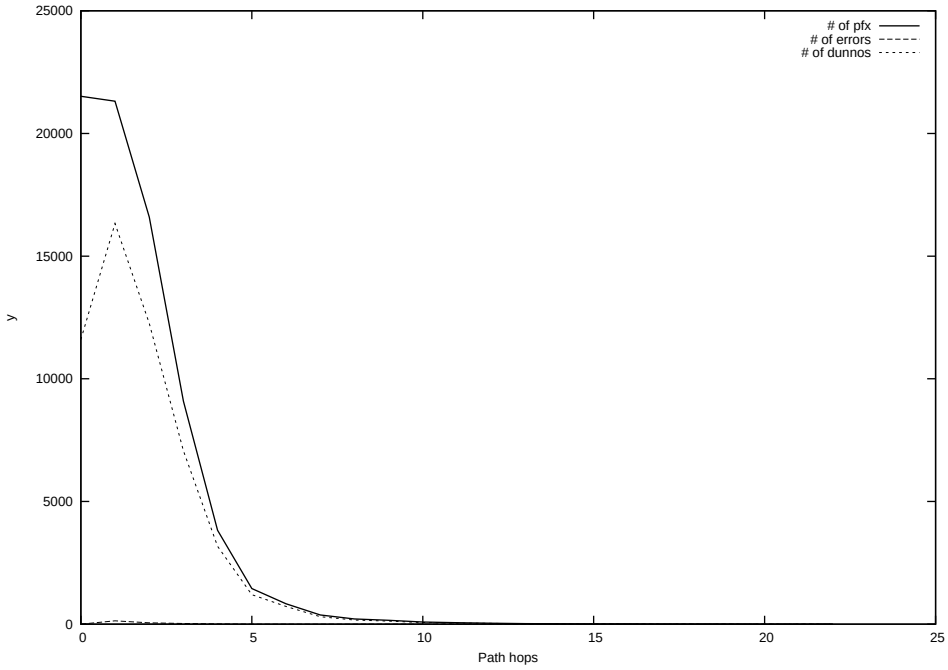
2015-02-08



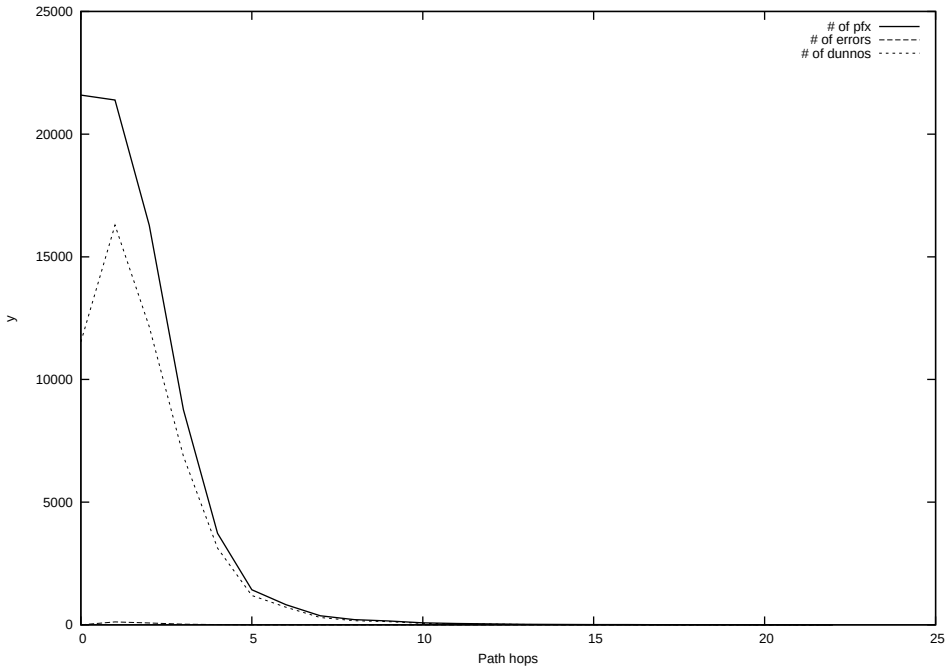
2015-02-09



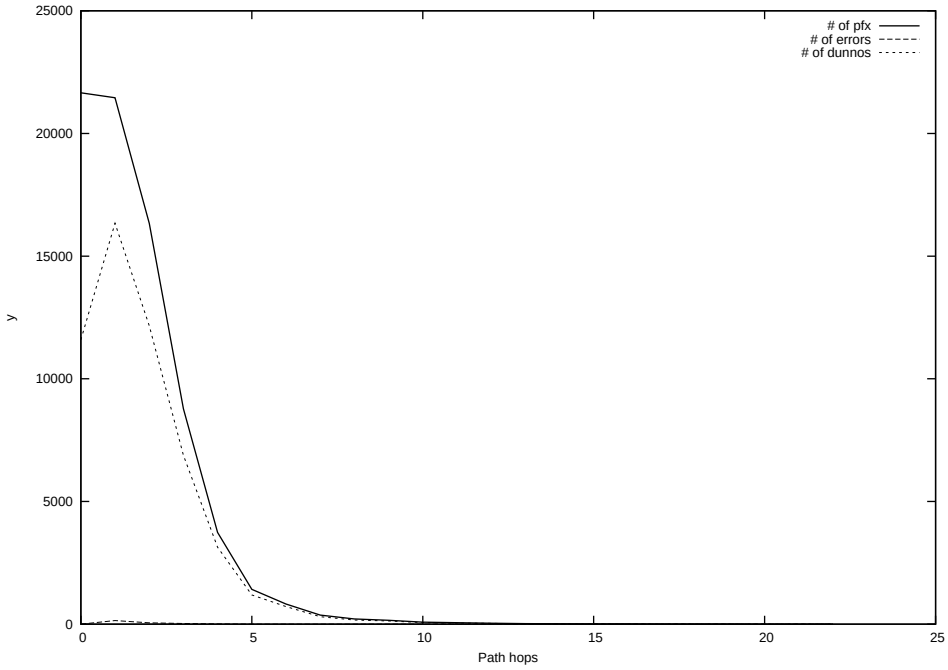
2015-02-10



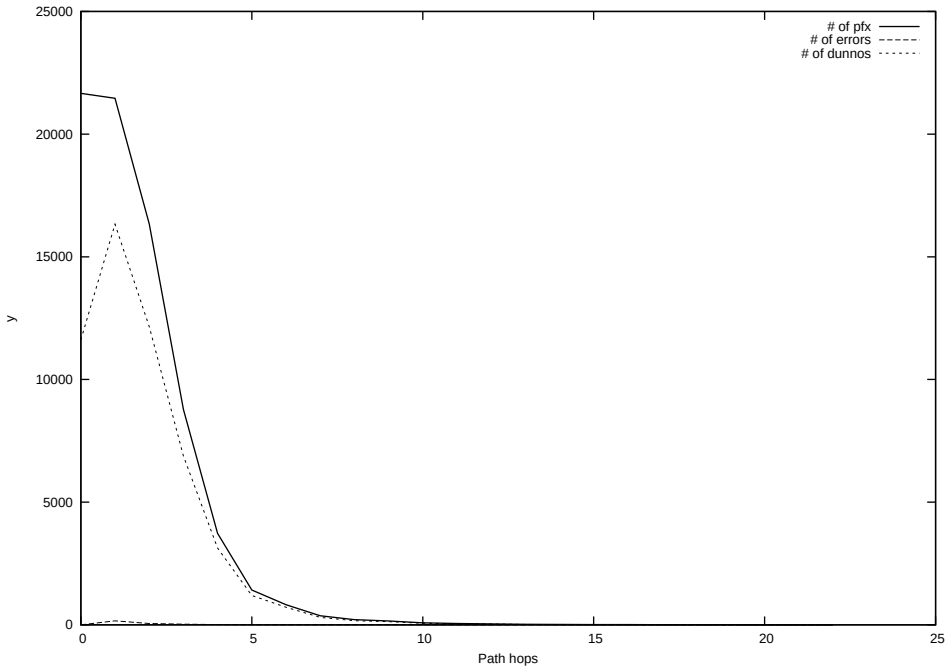
2015-02-11



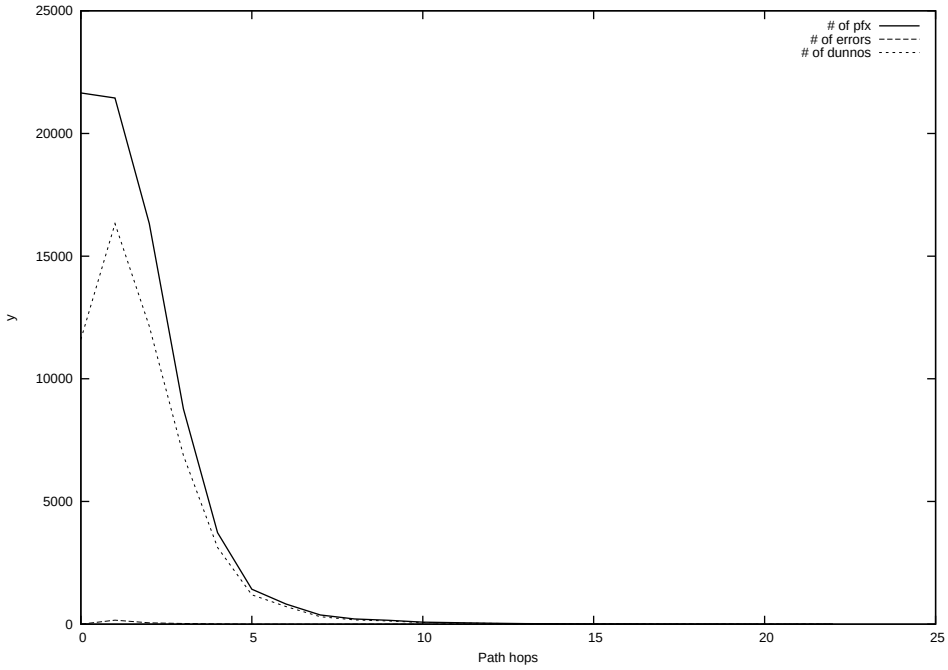
2015-02-12



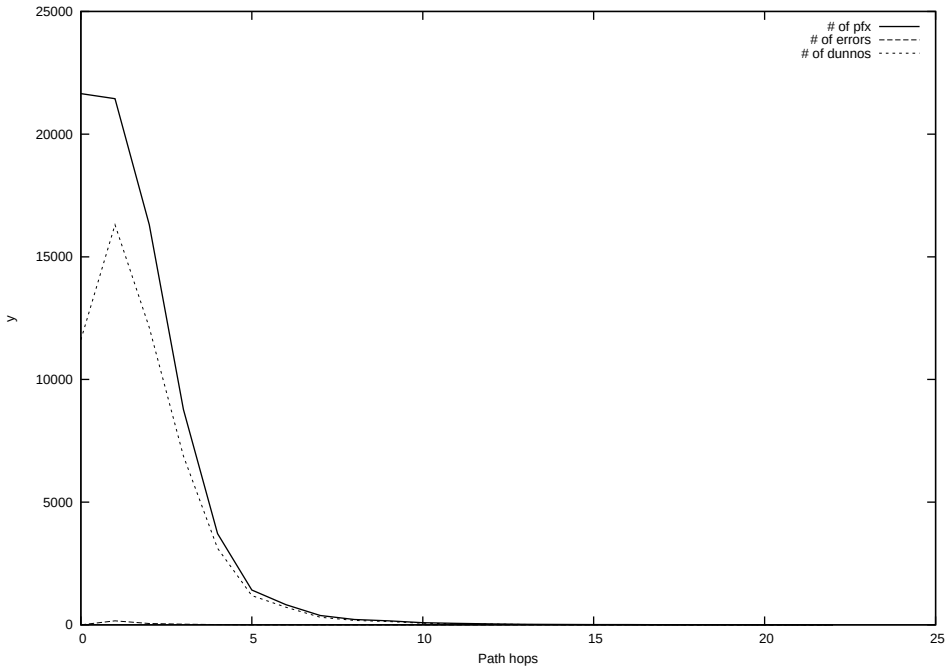
2015-02-13



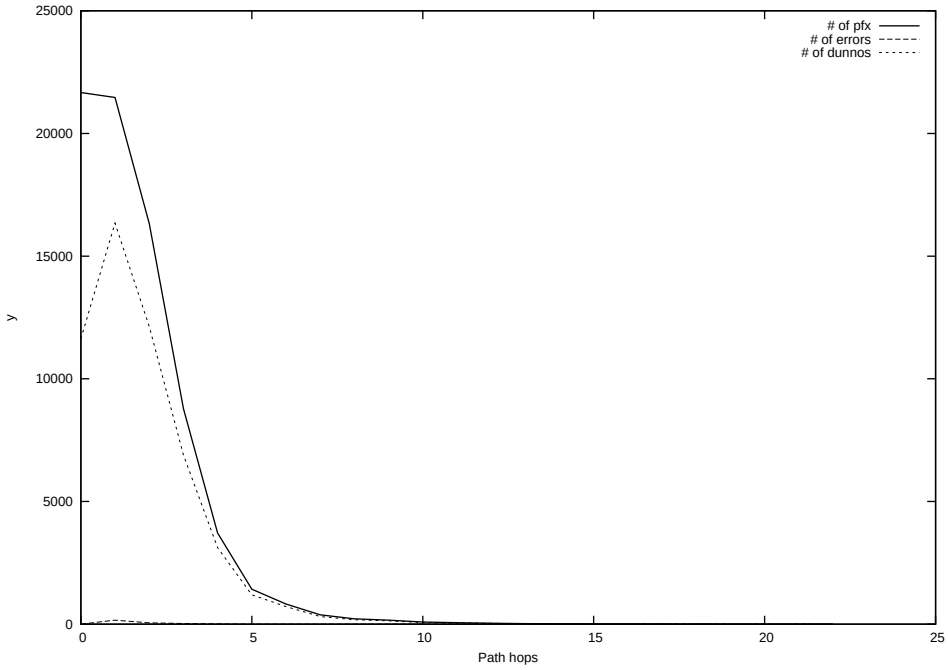
2015-02-14



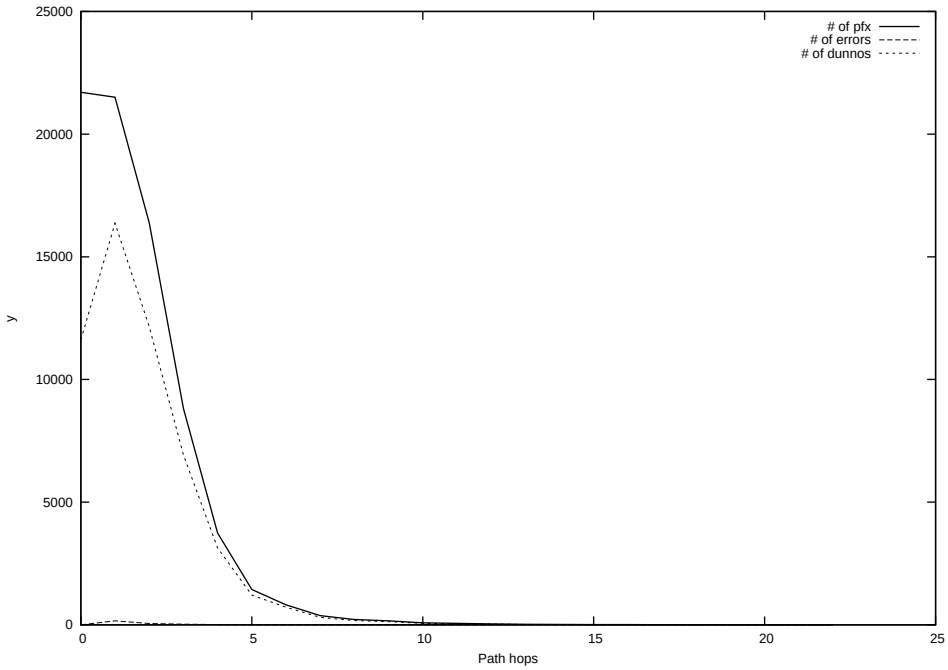
2015-02-15



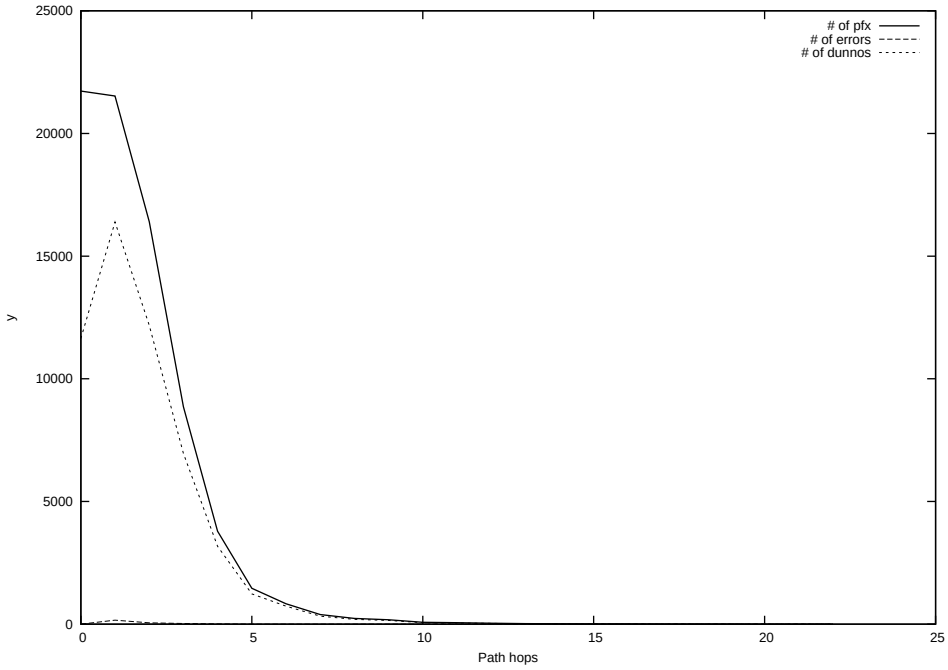
2015-02-16



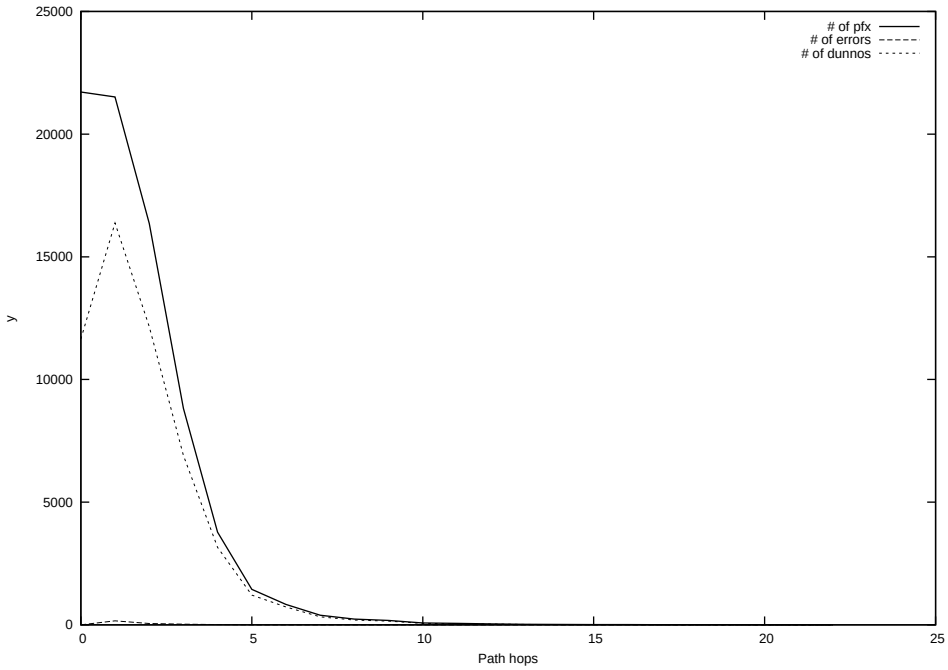
2015-02-17



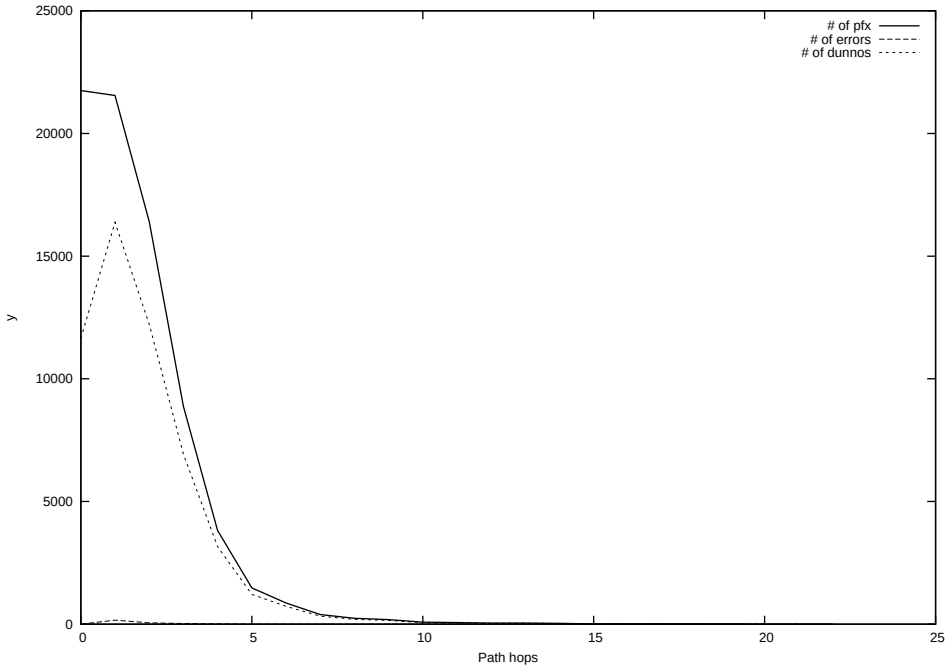
2015-02-18



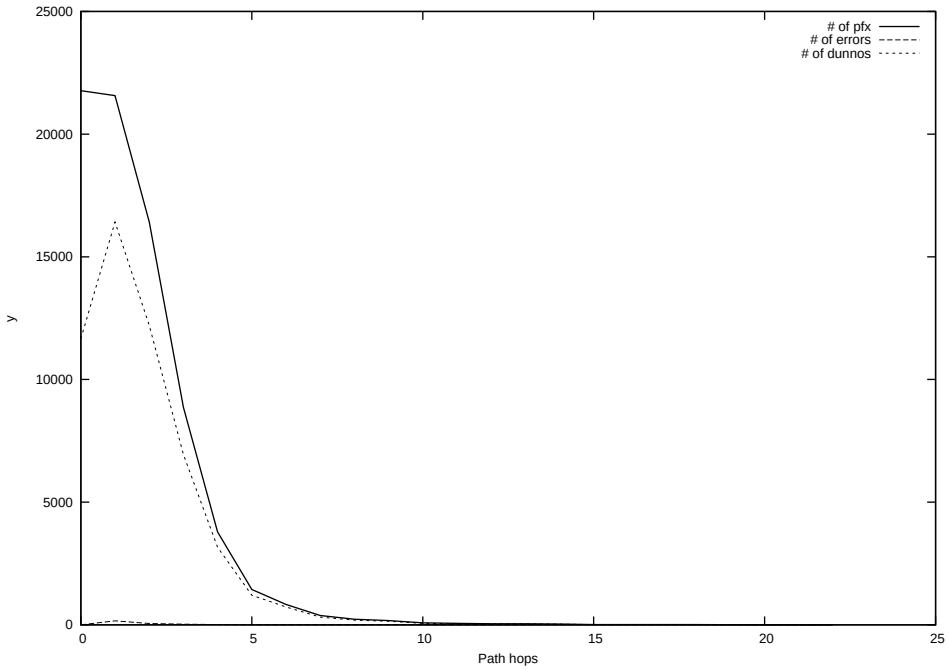
2015-02-19



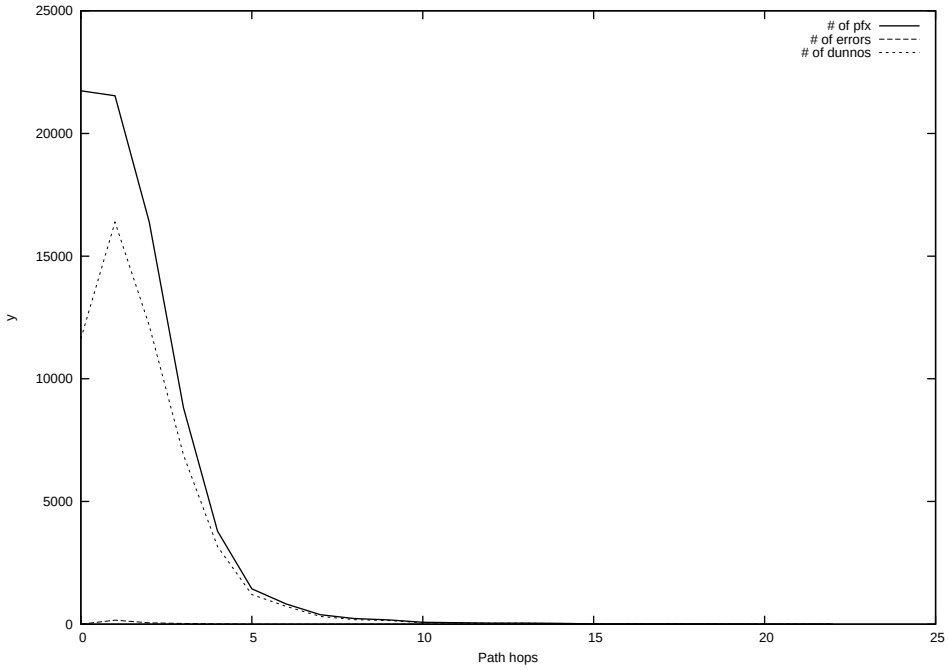
2015-02-20



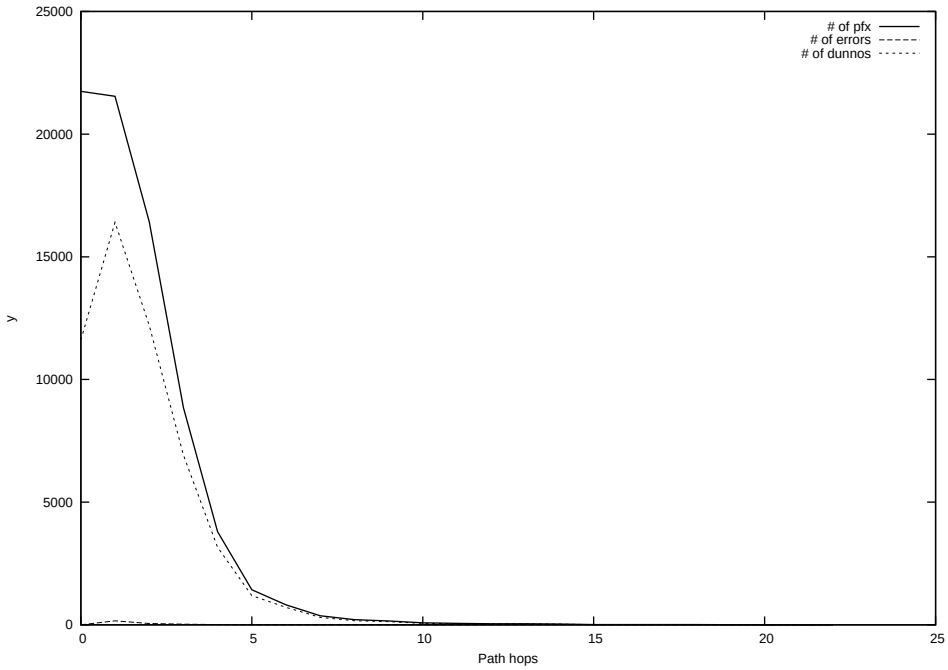
2015-02-21



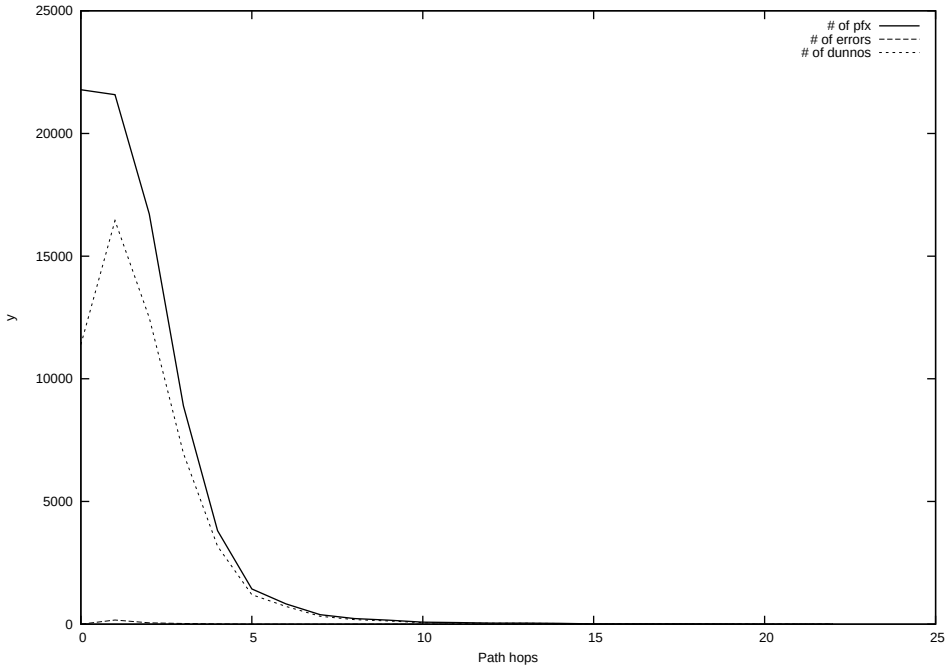
2015-02-22



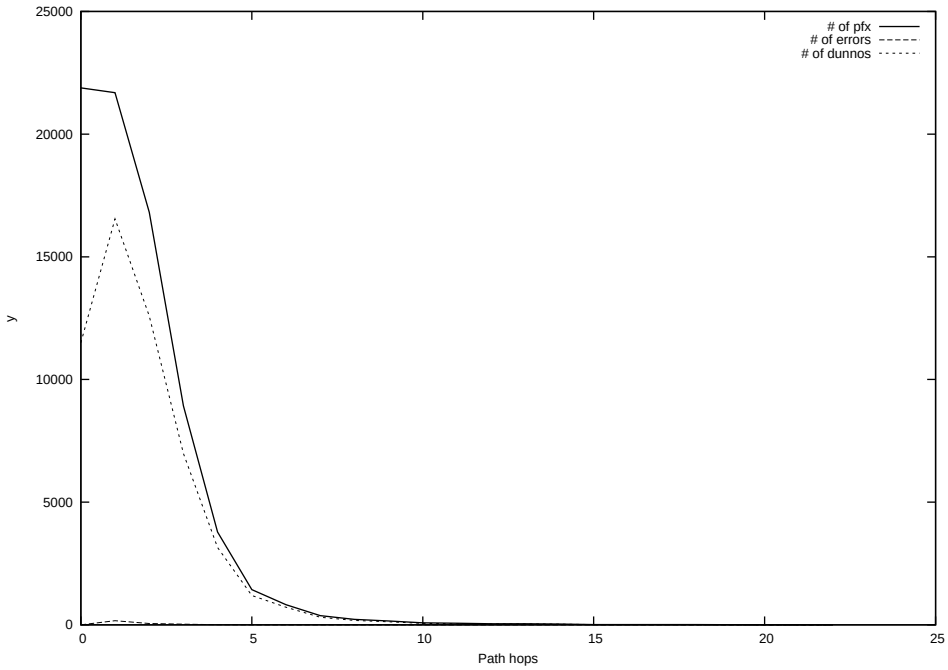
2015-02-23



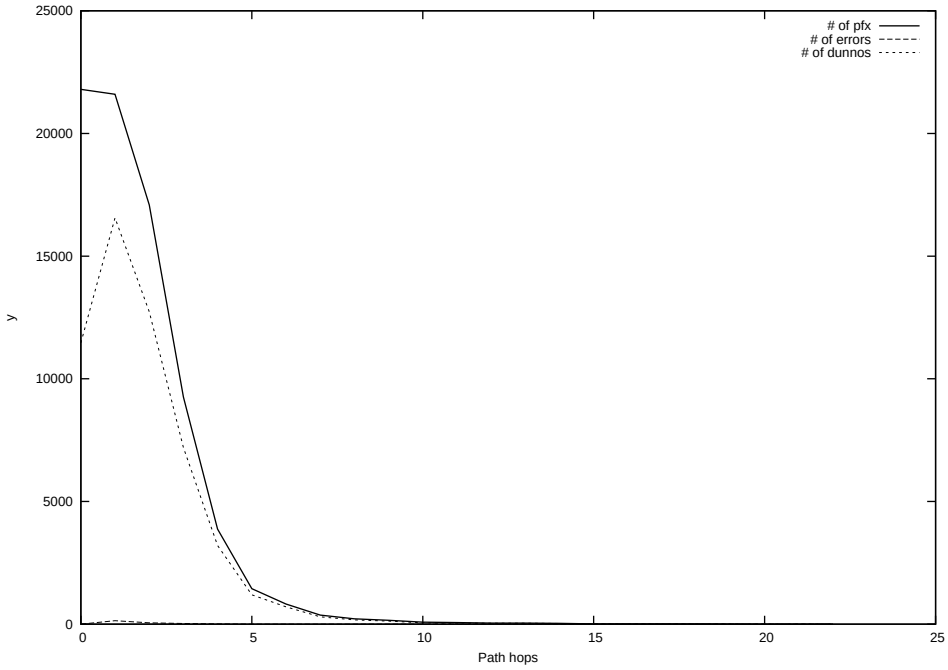
2015-02-24



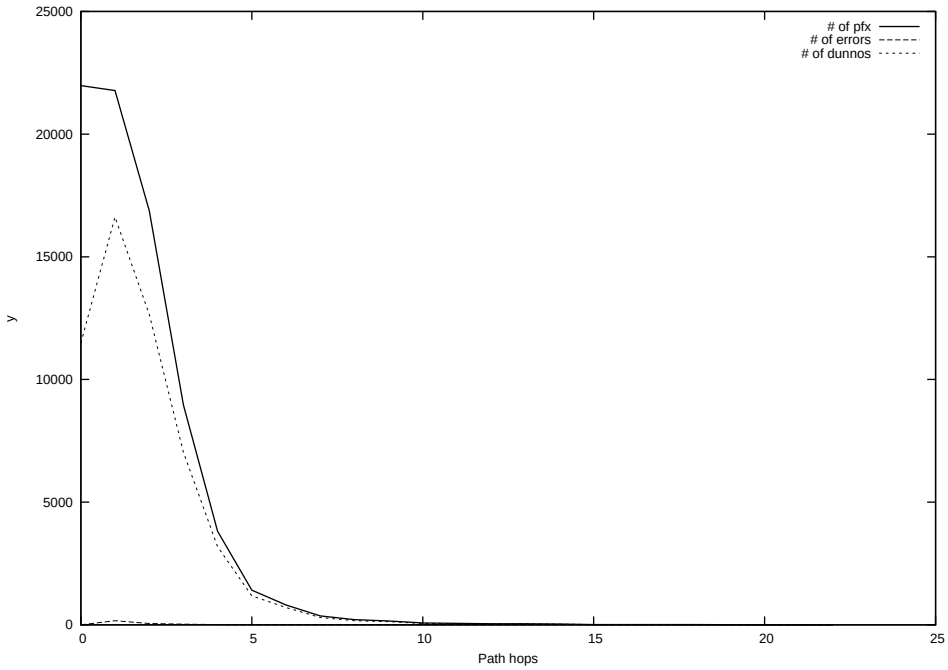
2015-02-25



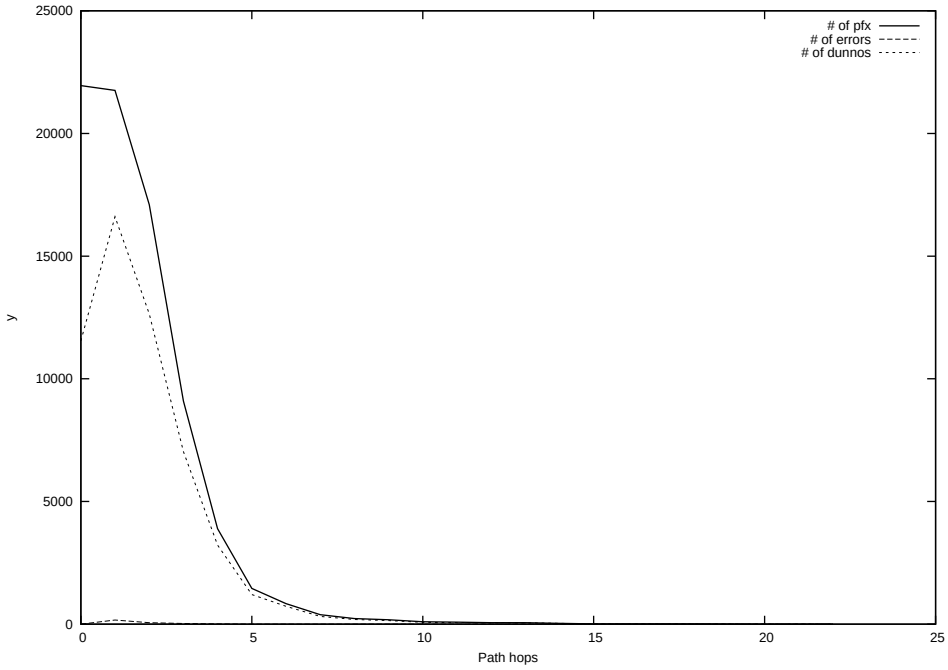
2015-02-26



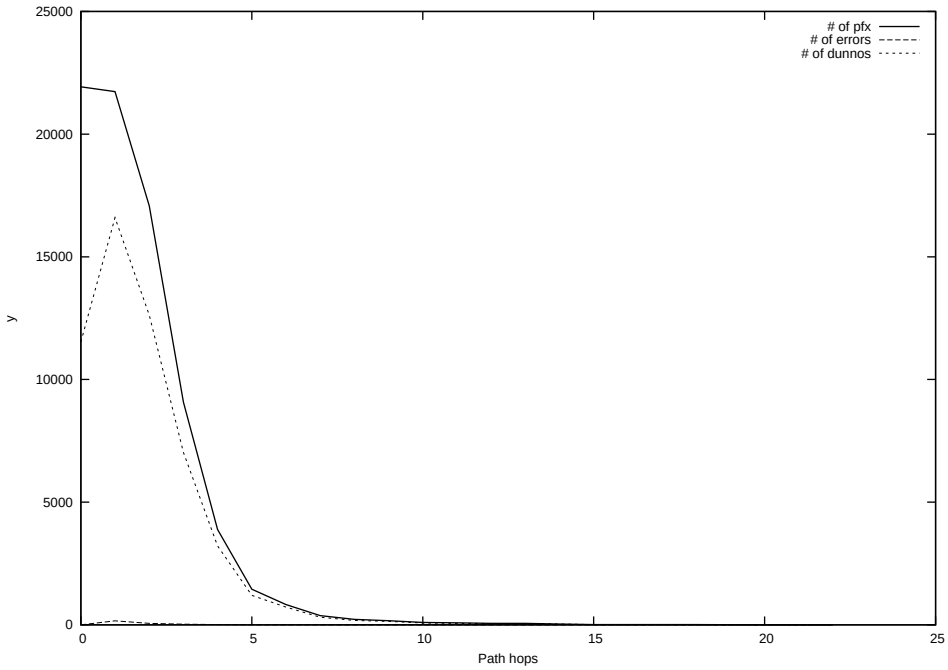
2015-02-27



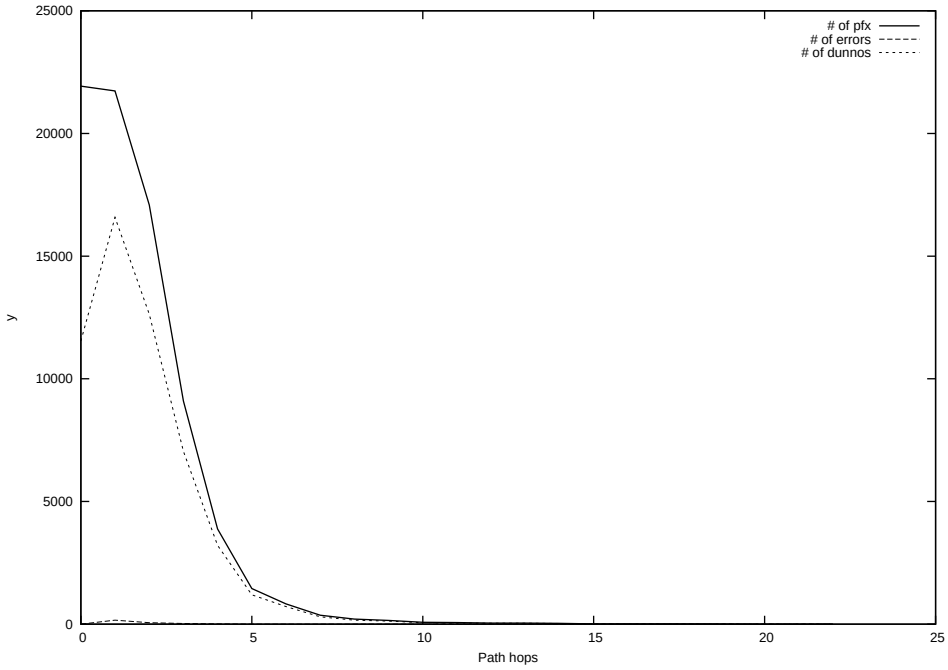
2015-02-28



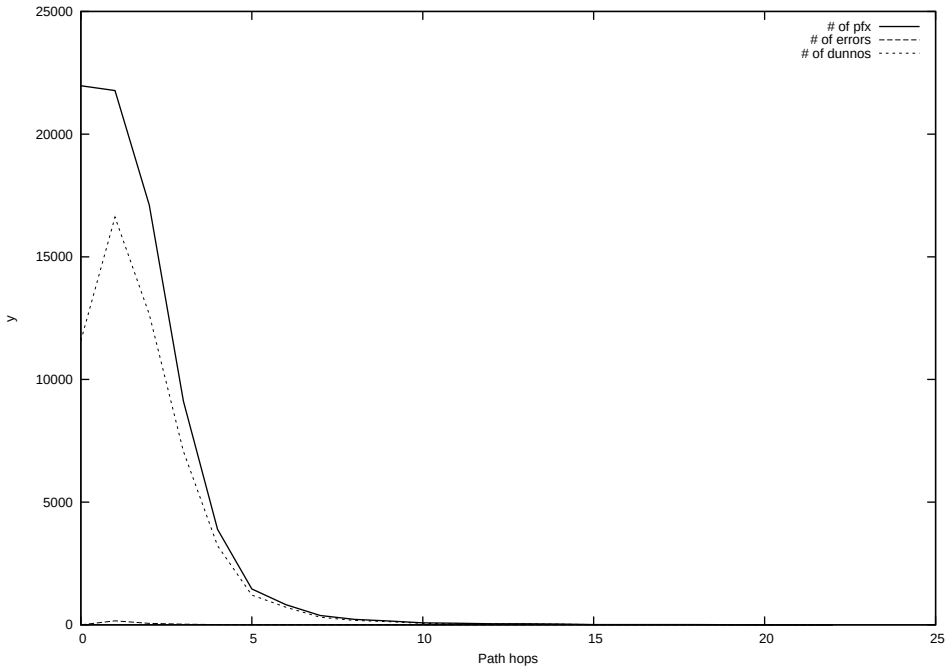
2015-03-01



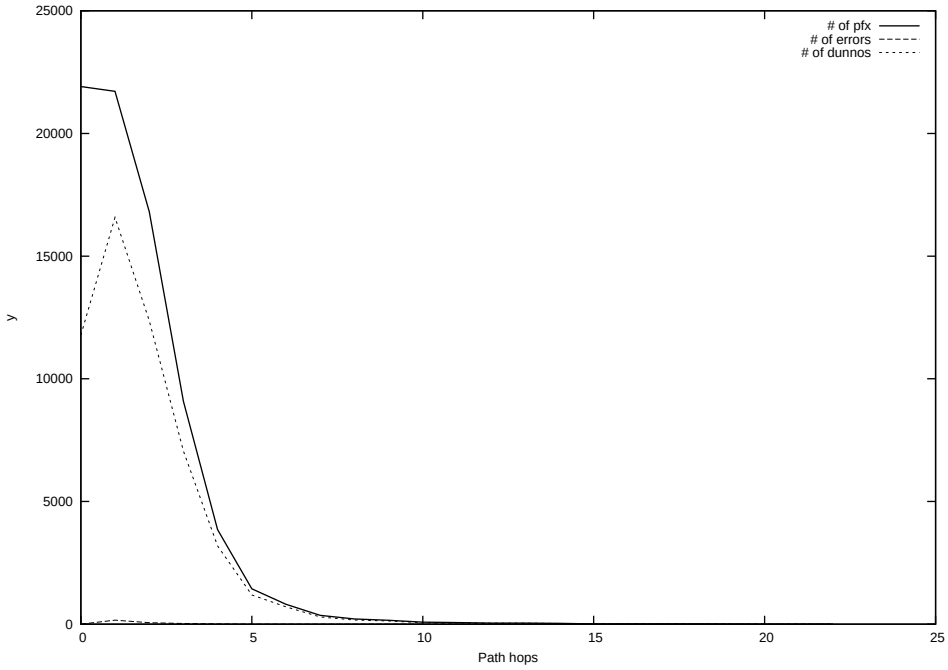
2015-03-02



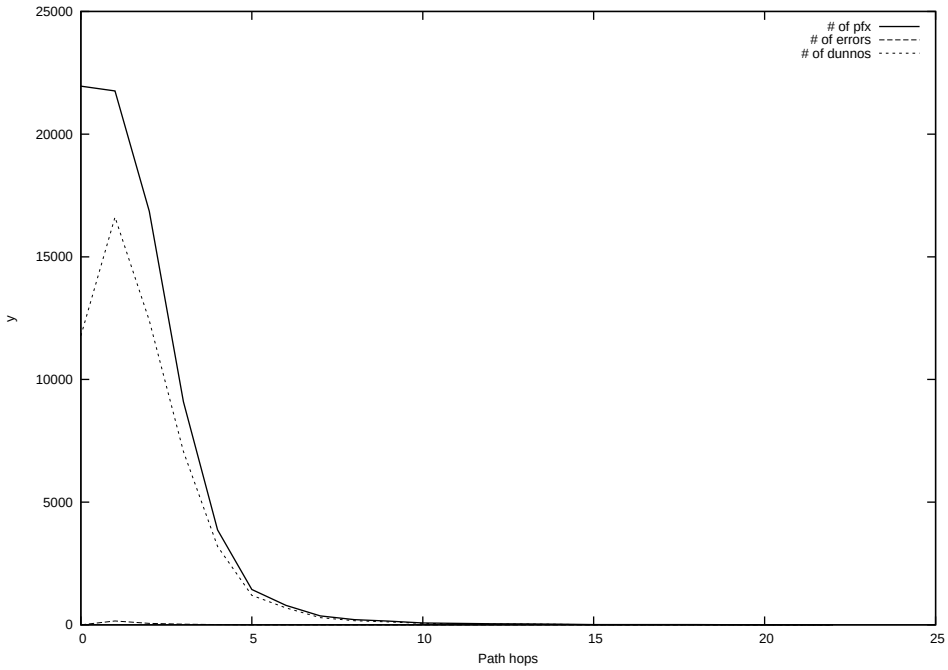
2015-03-03



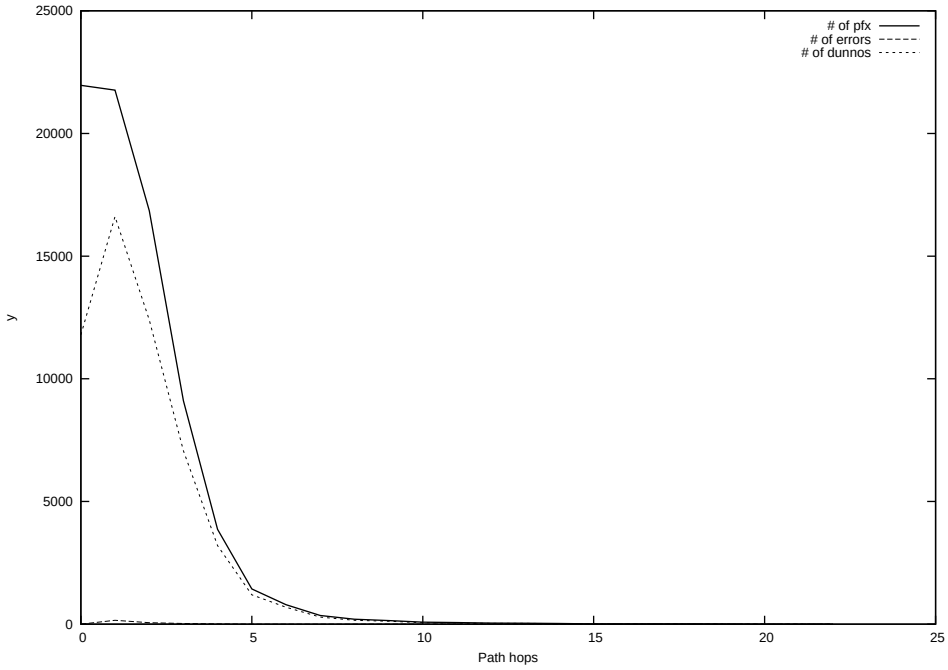
2015-03-04



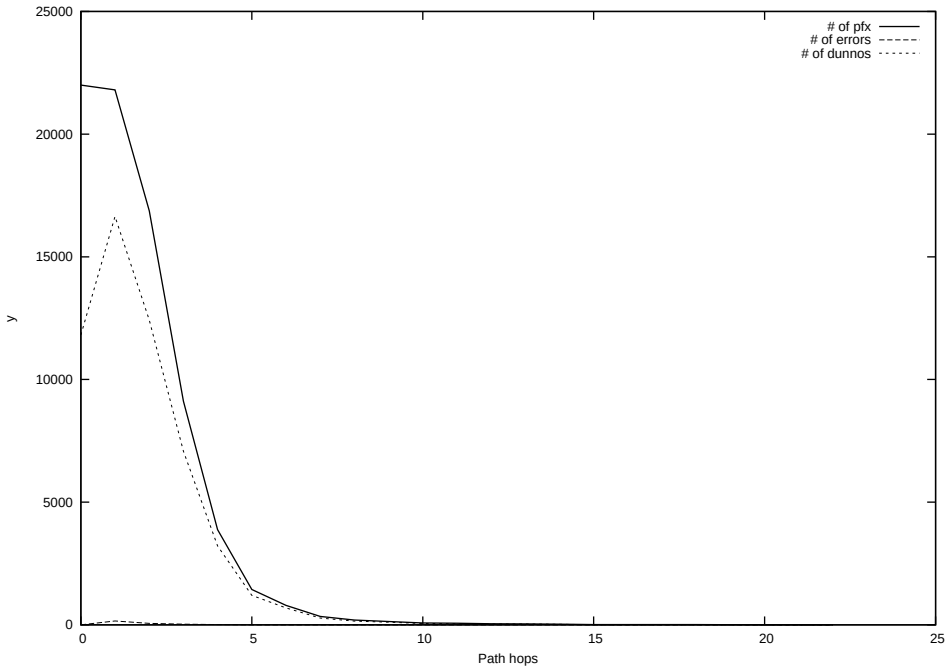
2015-03-05



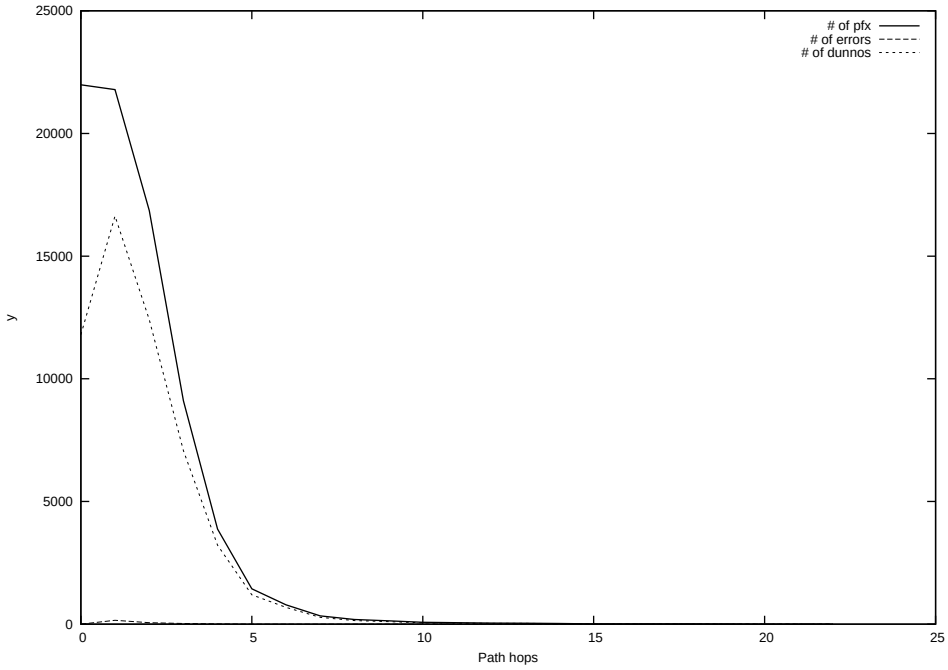
2015-03-06



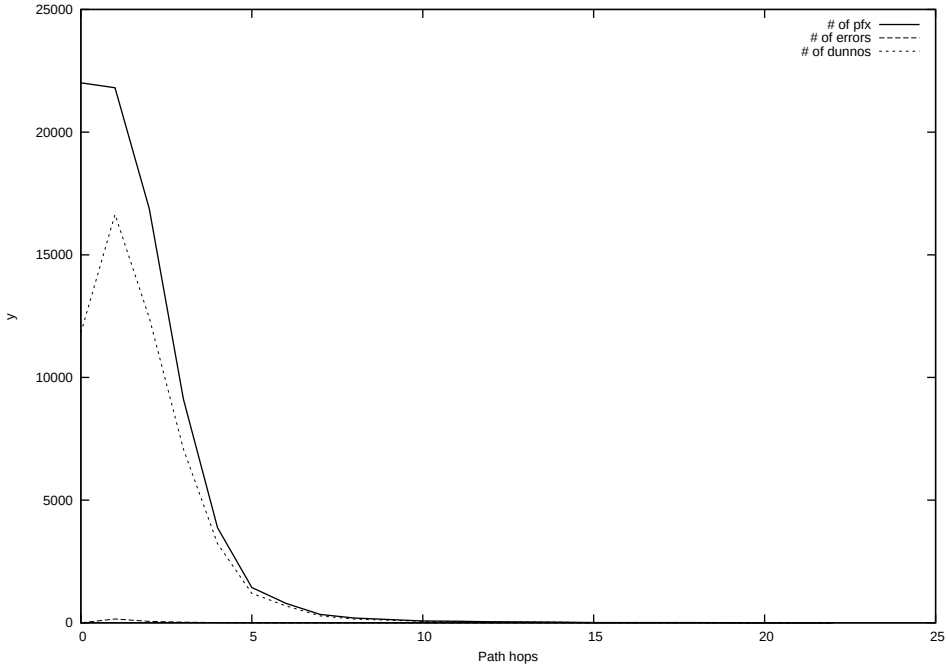
2015-03-07



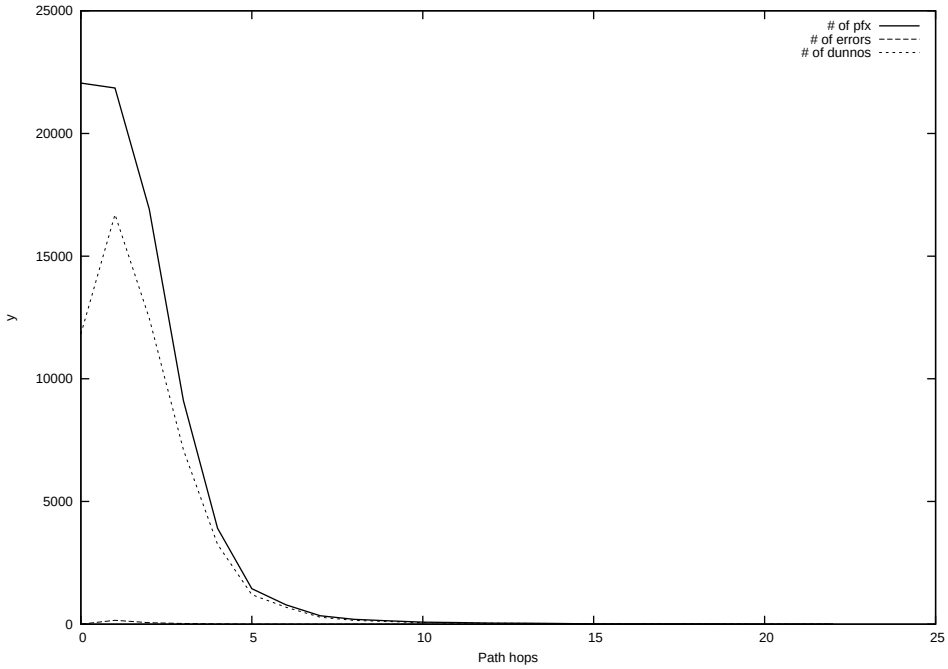
2015-03-08



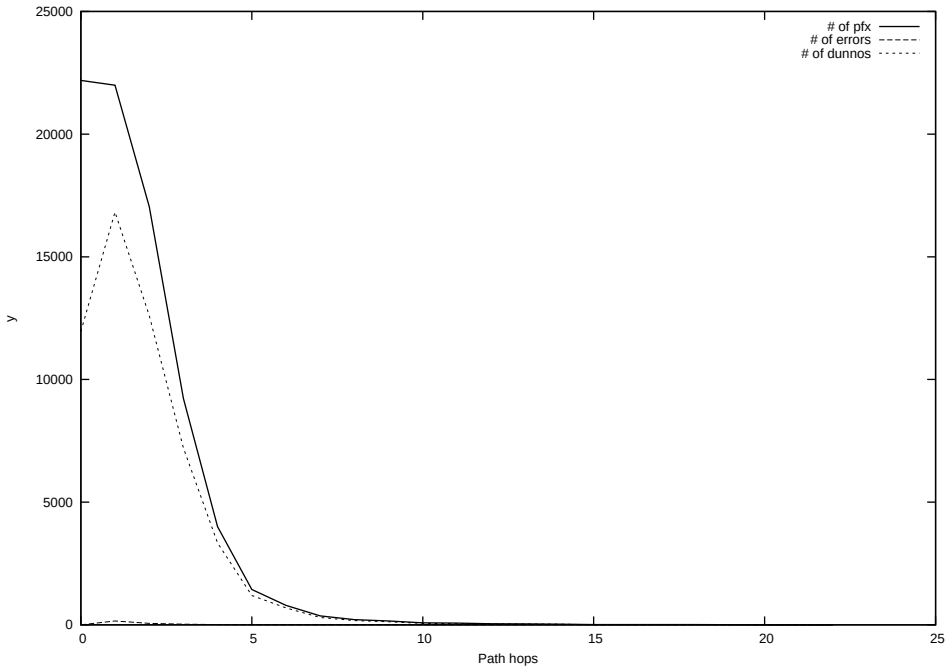
2015-03-09



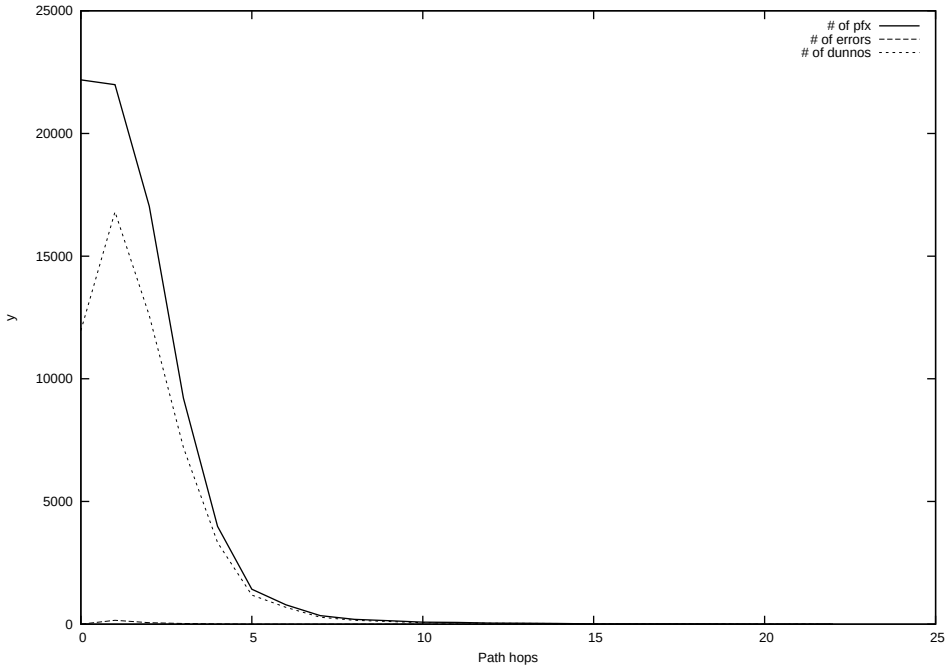
2015-03-10



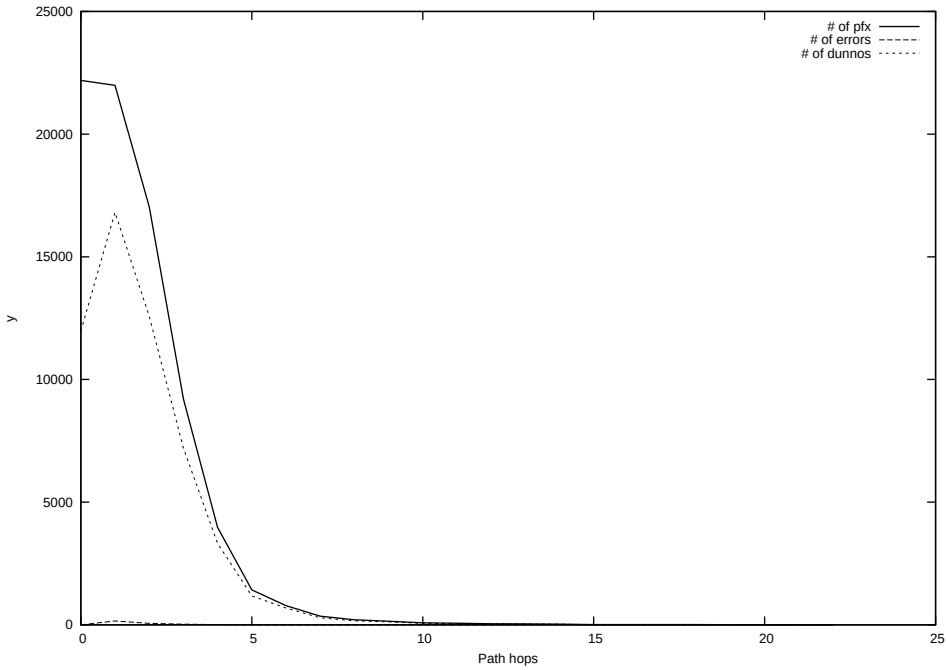
2015-03-11



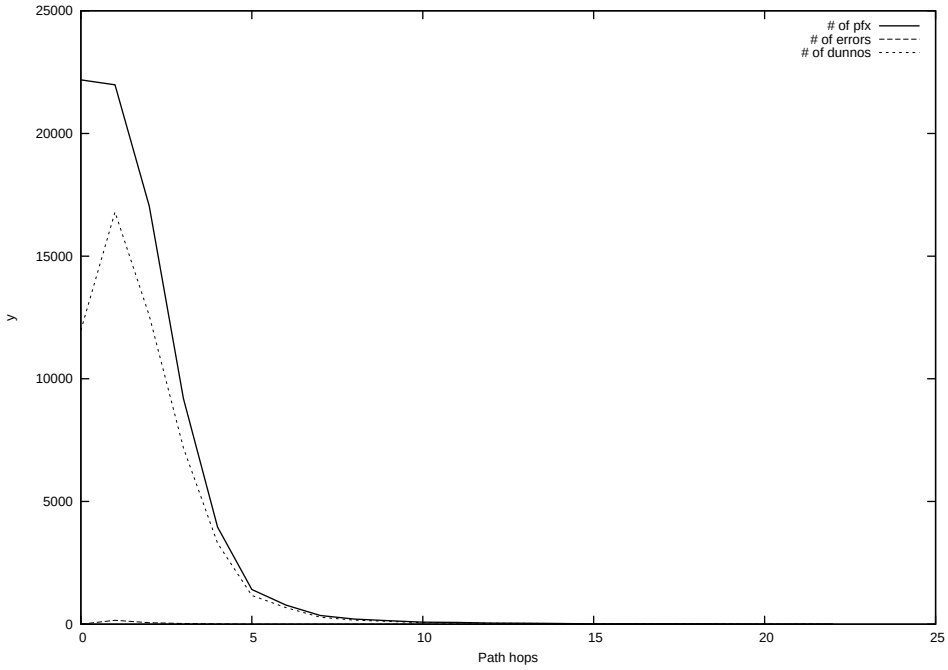
2015-03-12



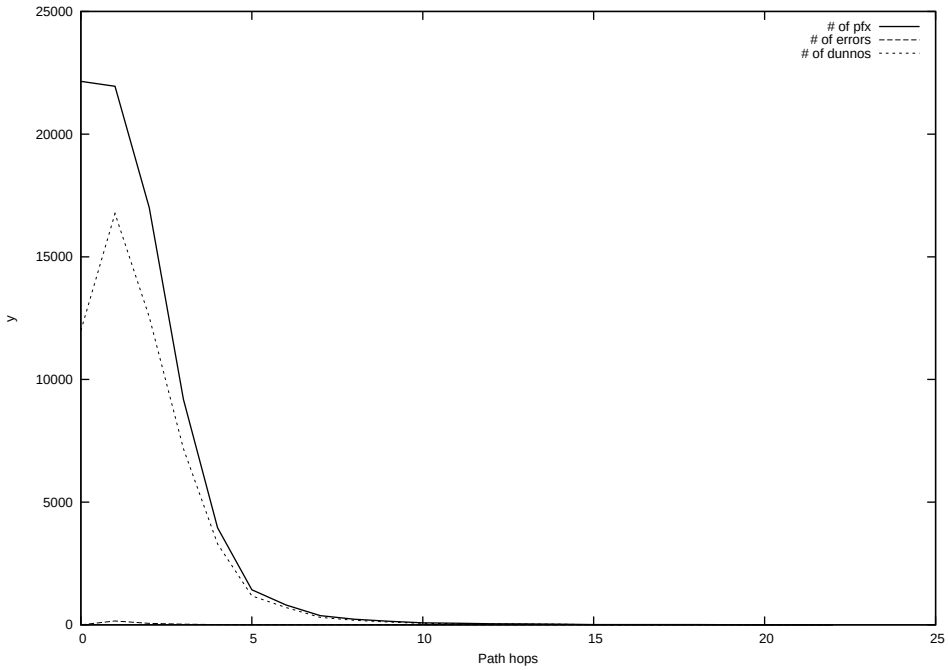
2015-03-13



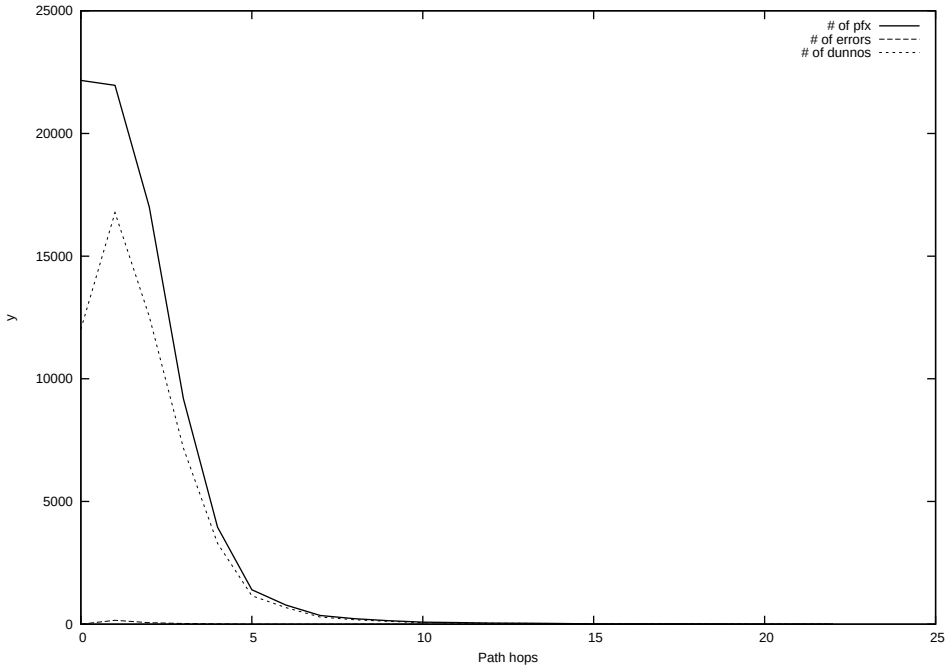
2015-03-14



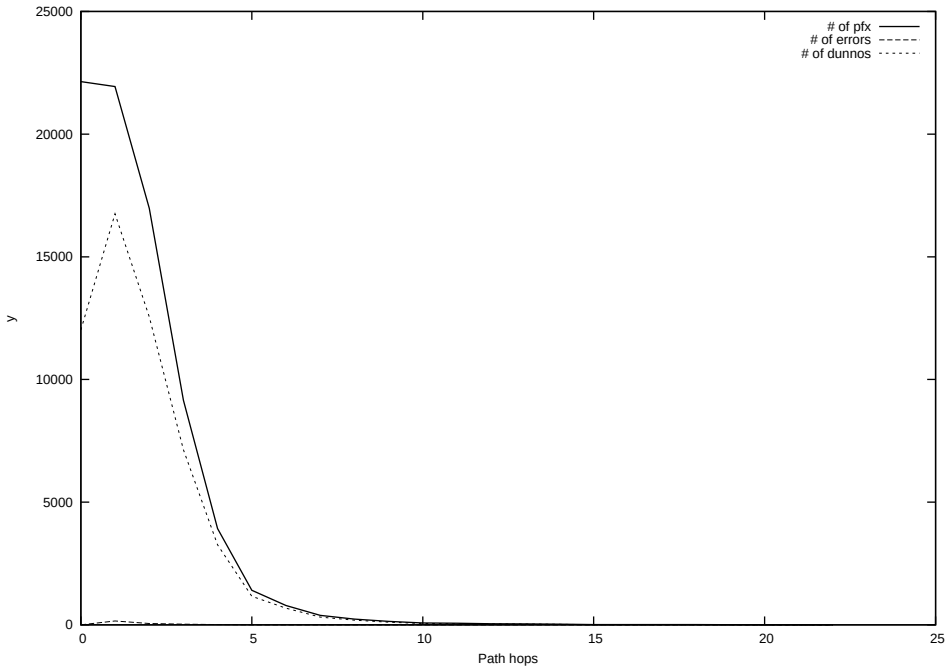
2015-03-15



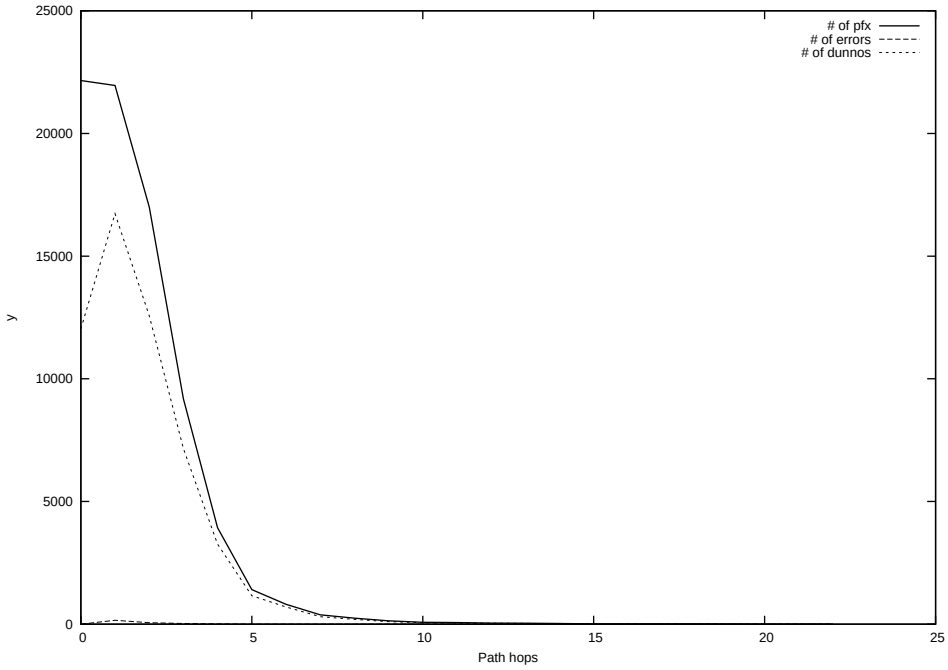
2015-03-16



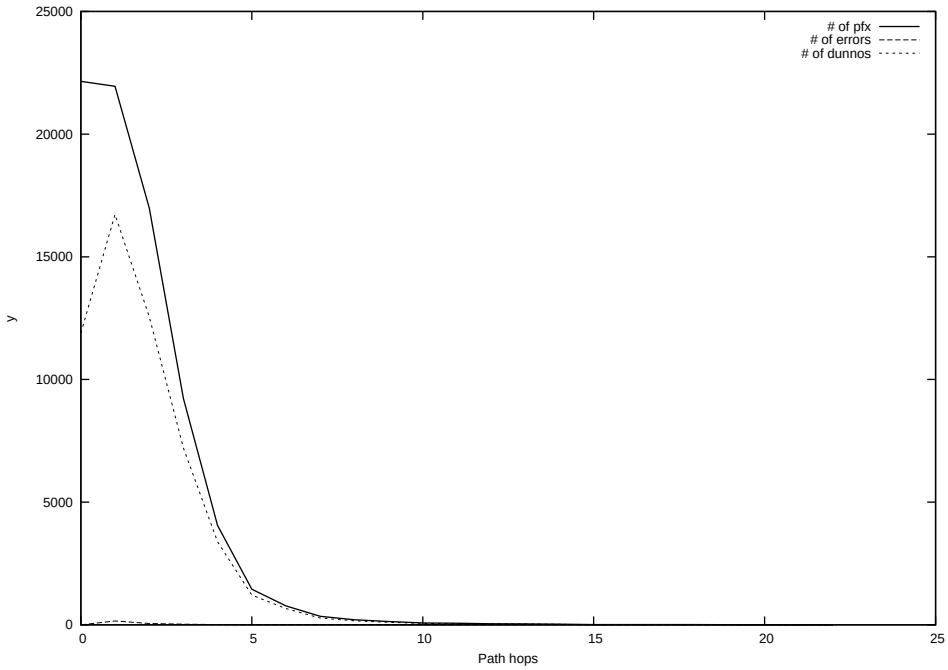
2015-03-17



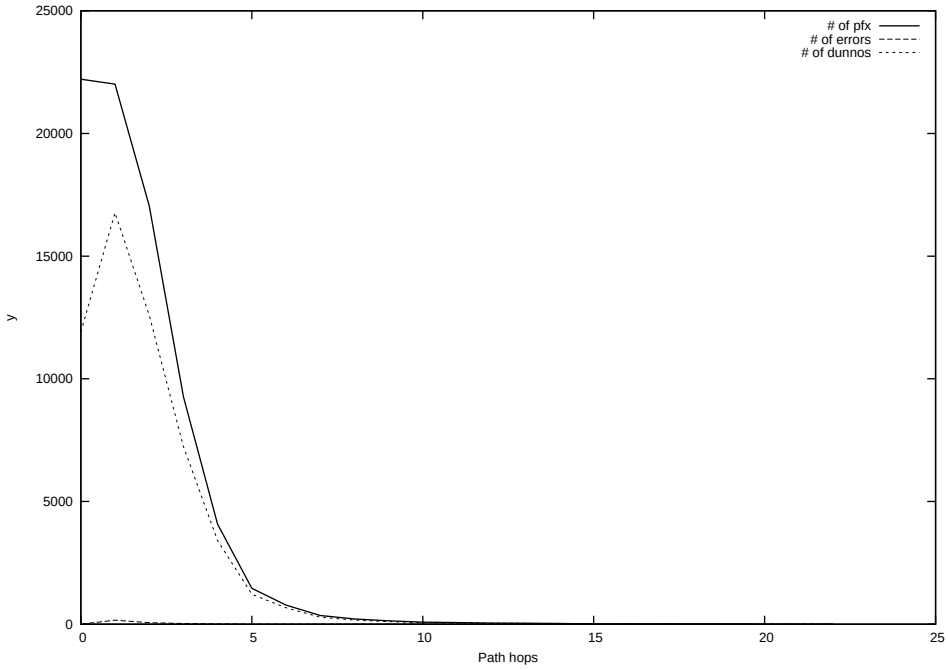
2015-03-18



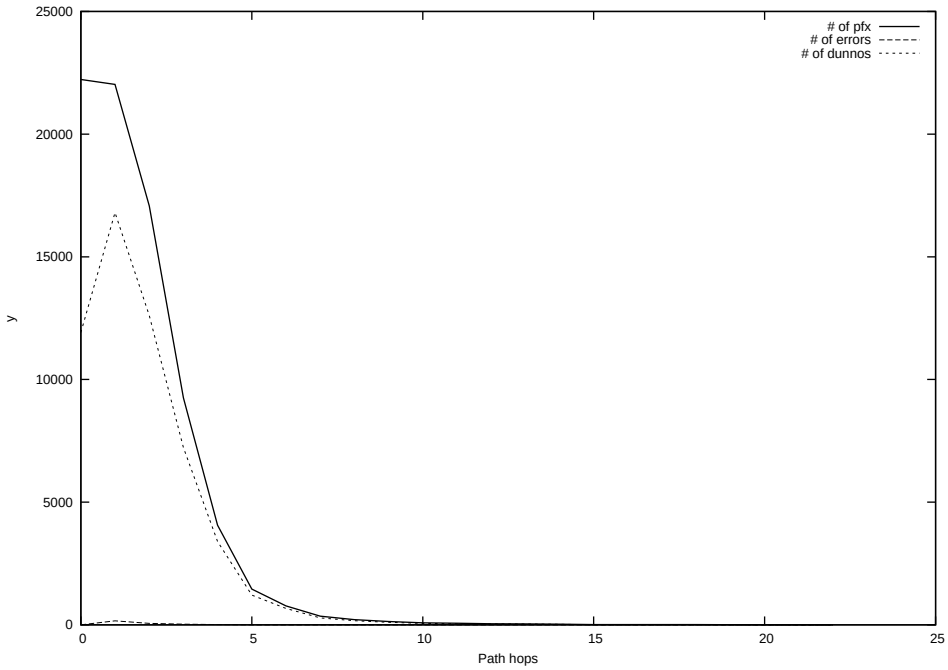
2015-03-19



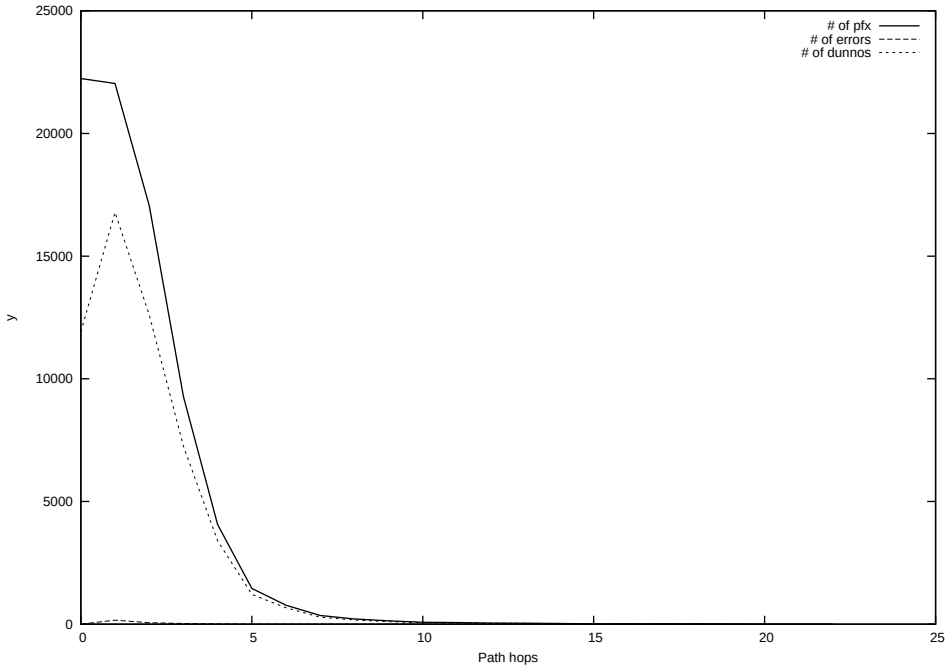
2015-03-20



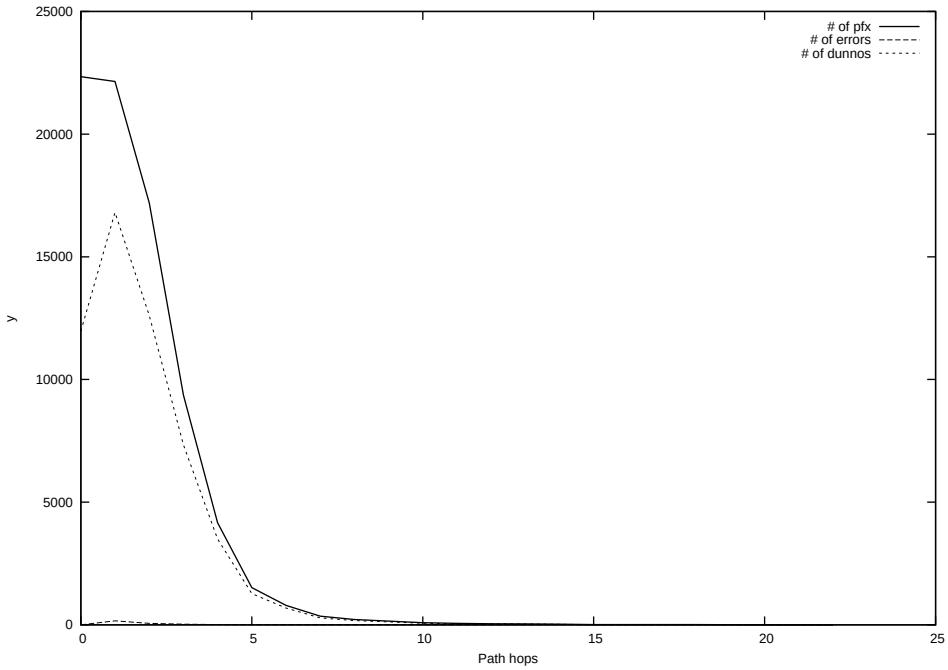
2015-03-21



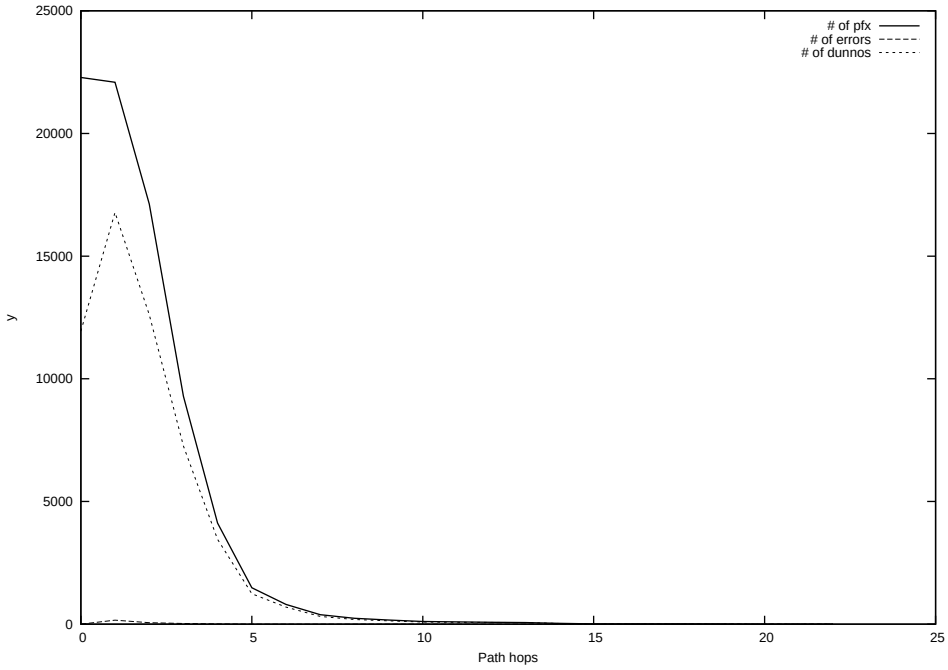
2015-03-22



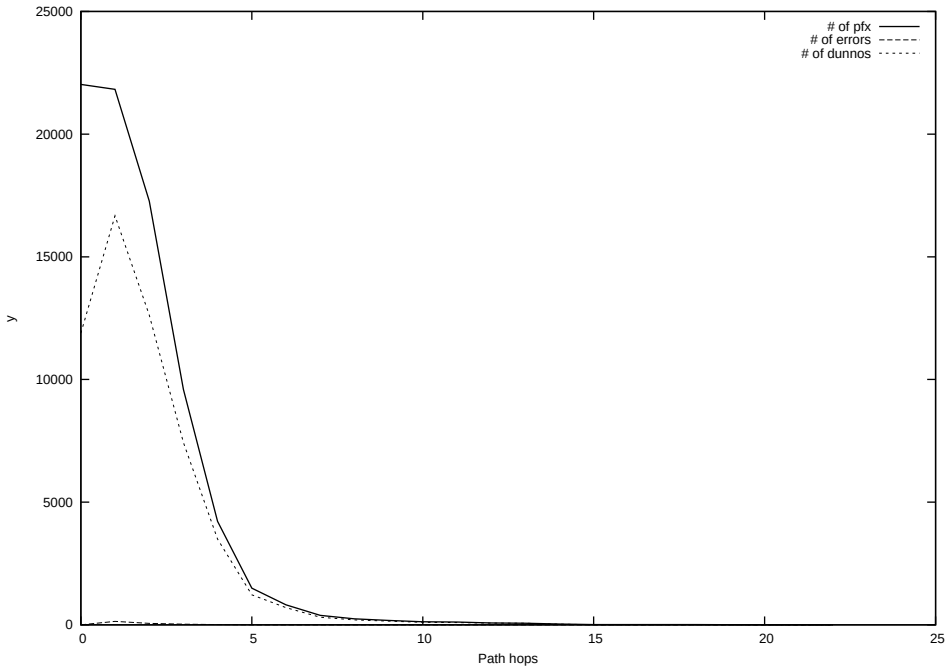
2015-03-23



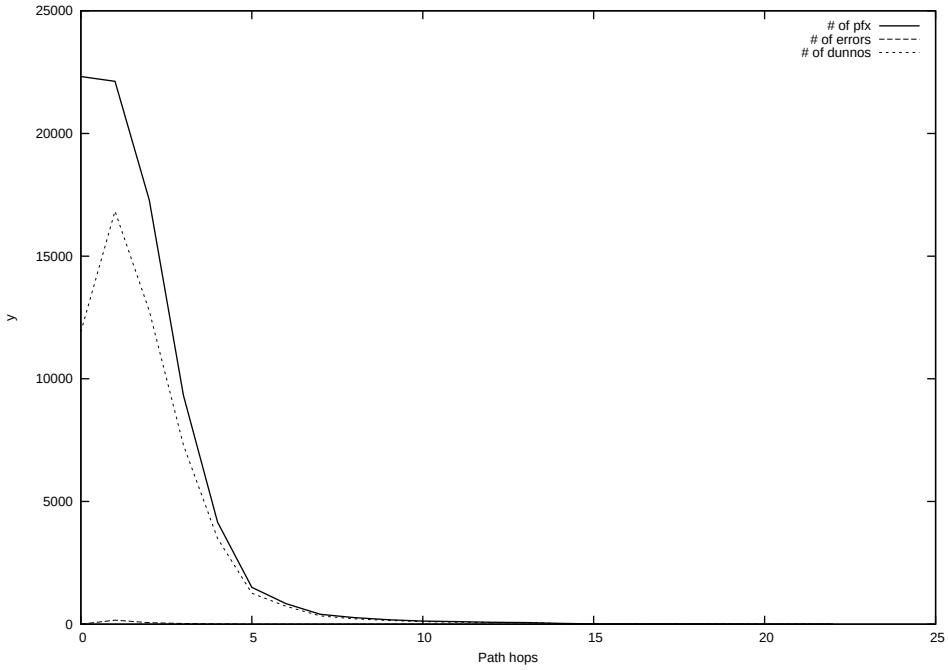
2015-03-24



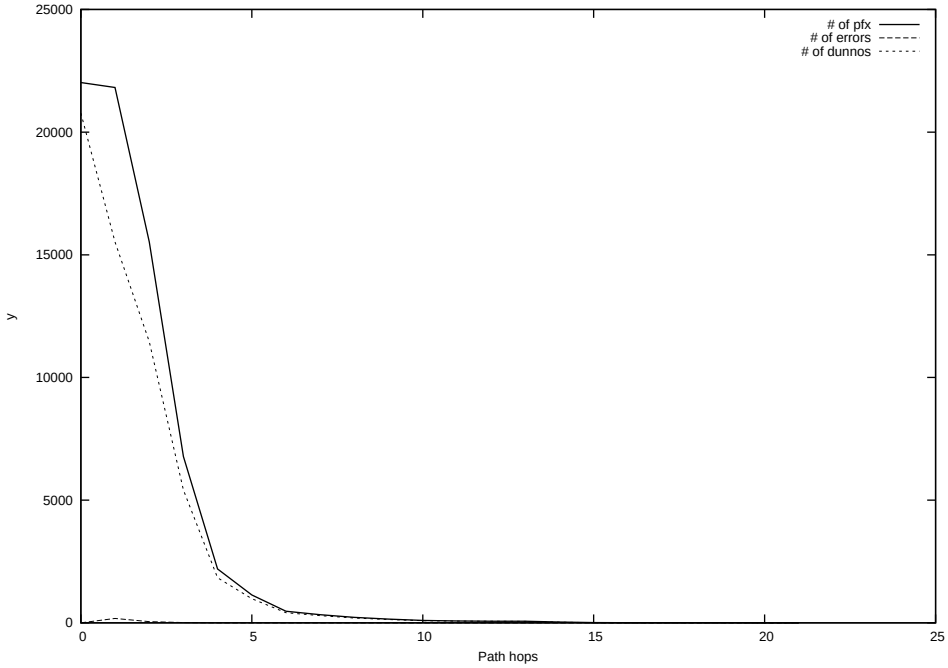
2015-03-25



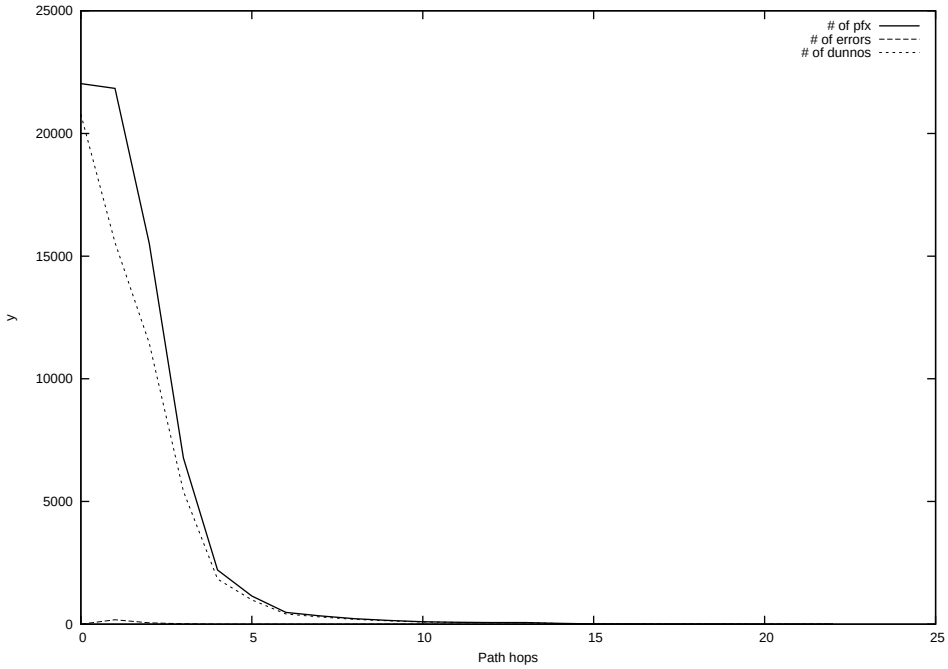
2015-03-26



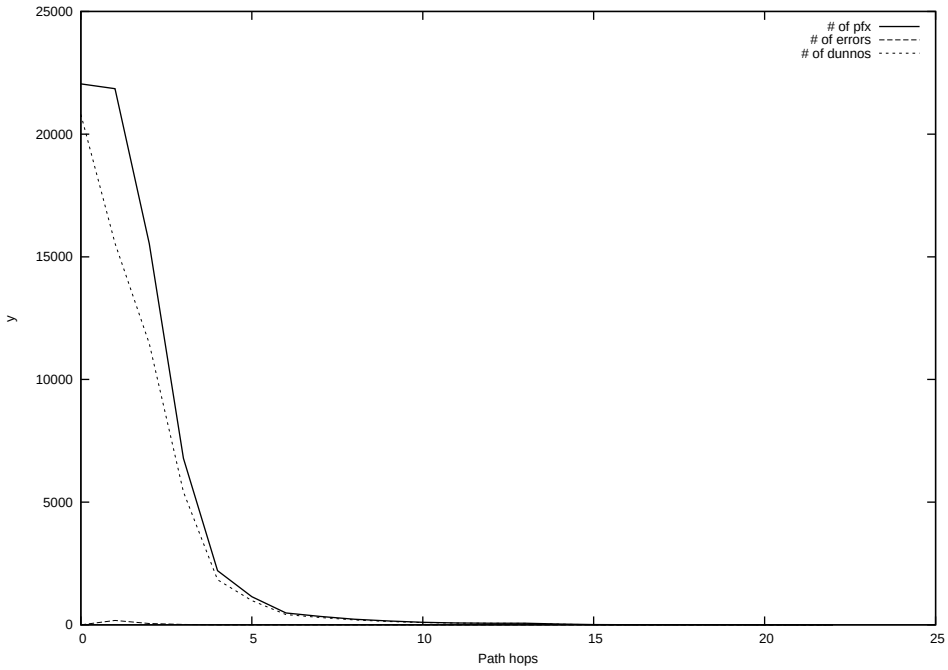
2015-03-27



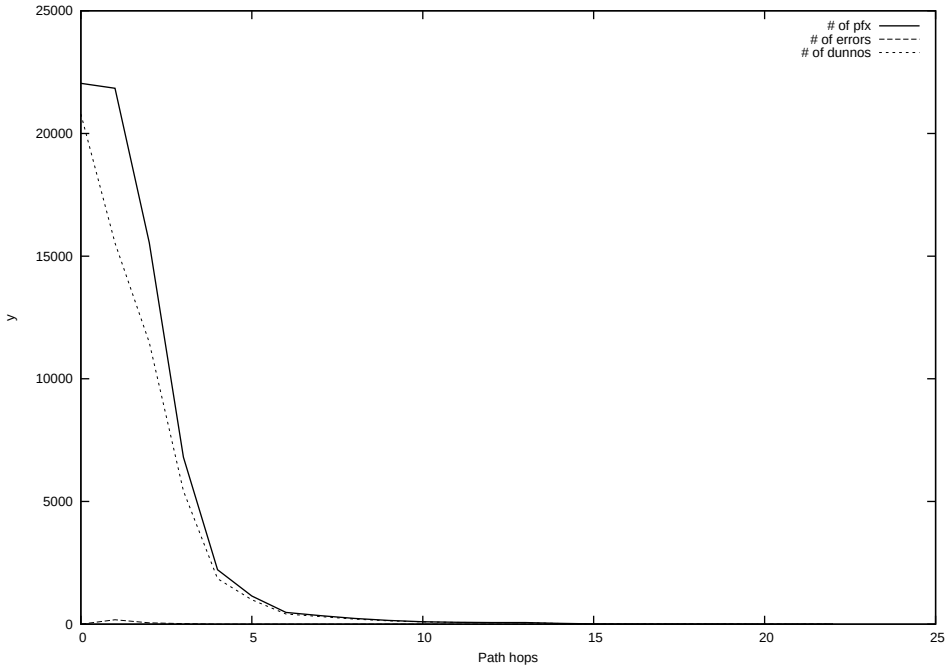
2015-03-28



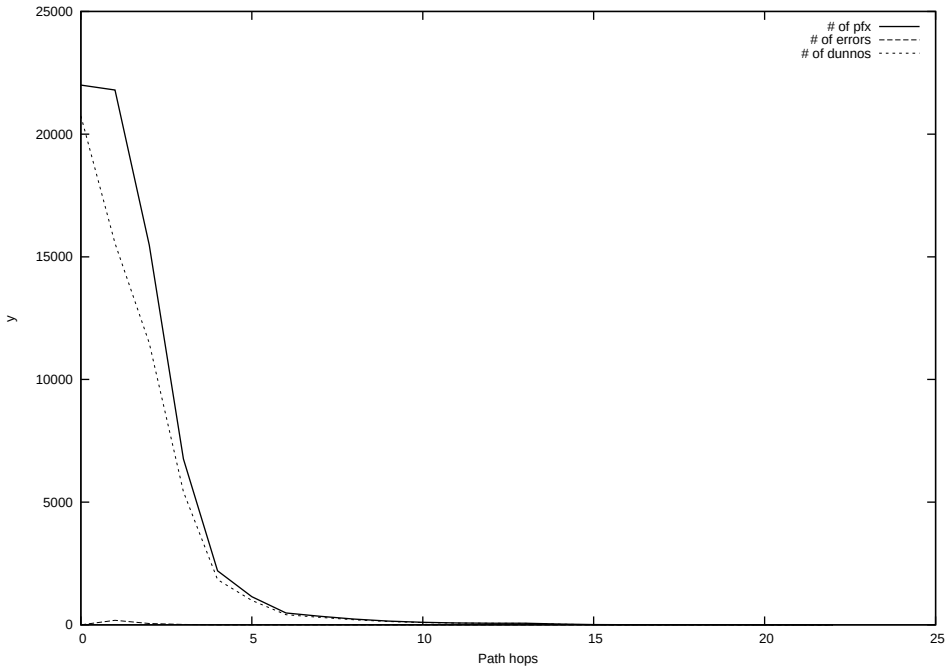
2015-03-29



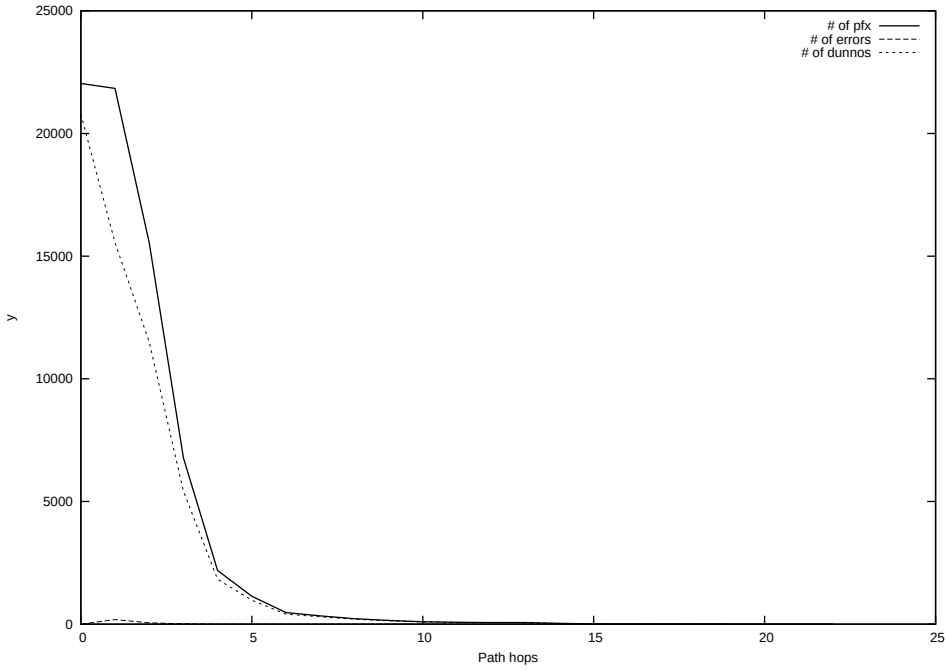
2015-03-30



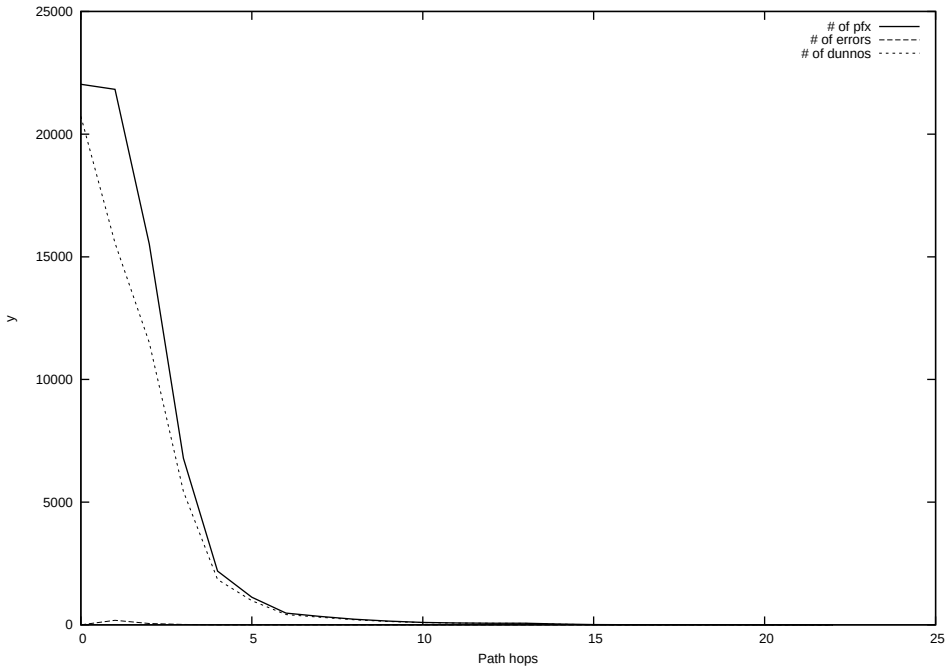
2015-03-31



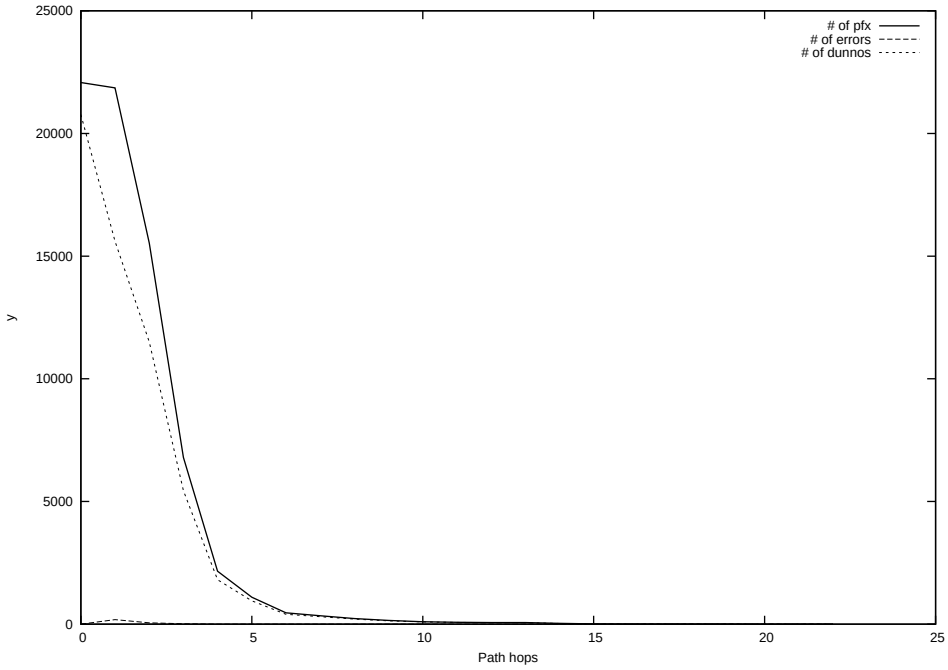
2015-04-01



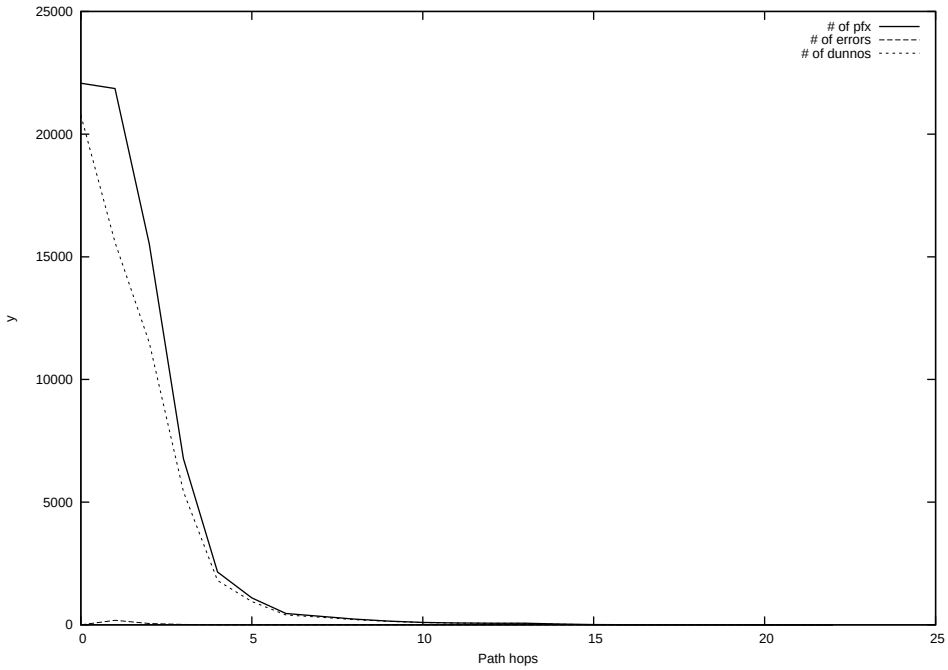
2015-04-02



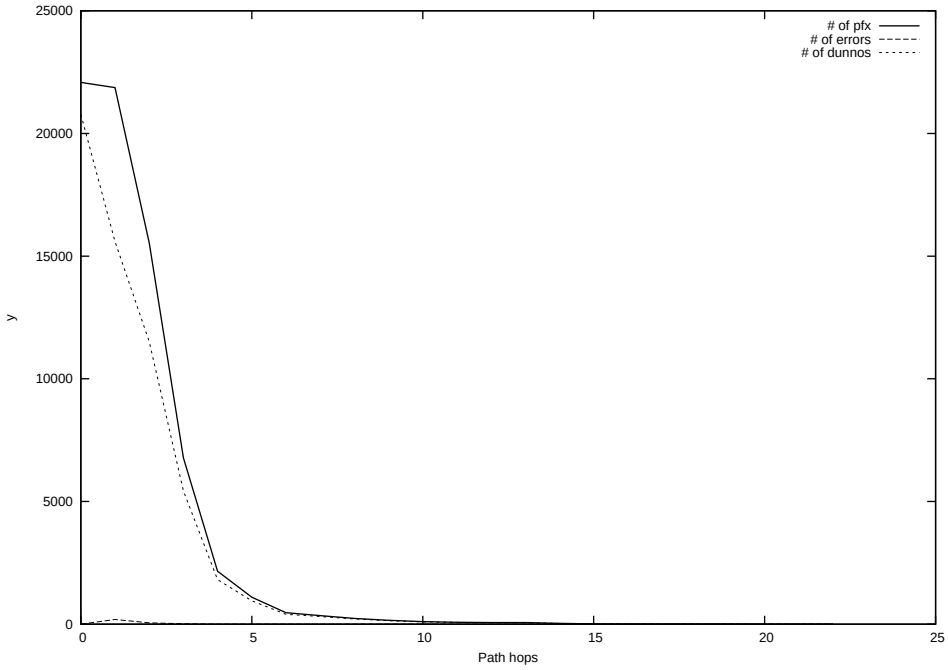
2015-04-03



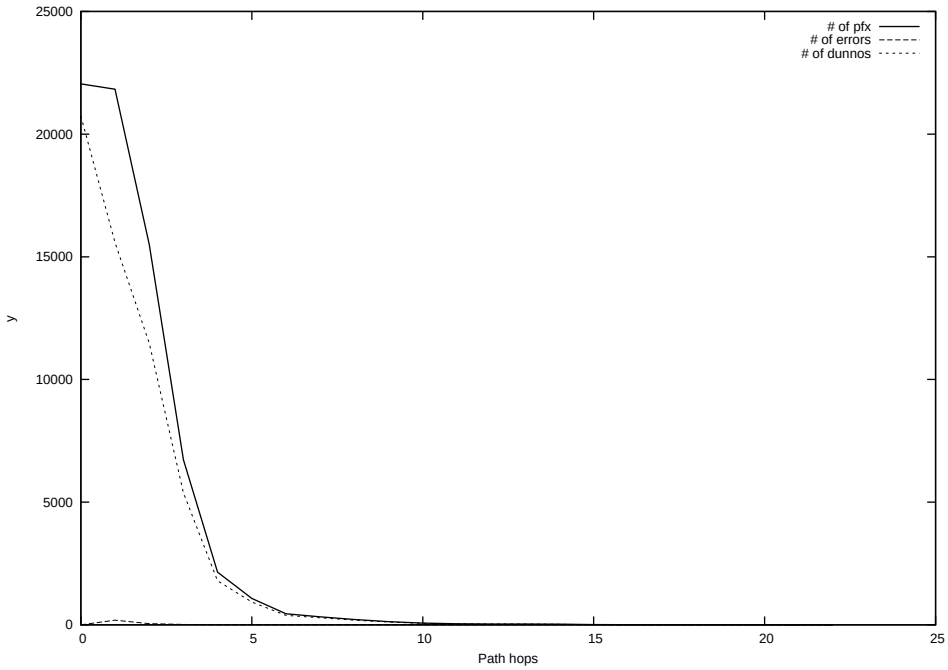
2015-04-04



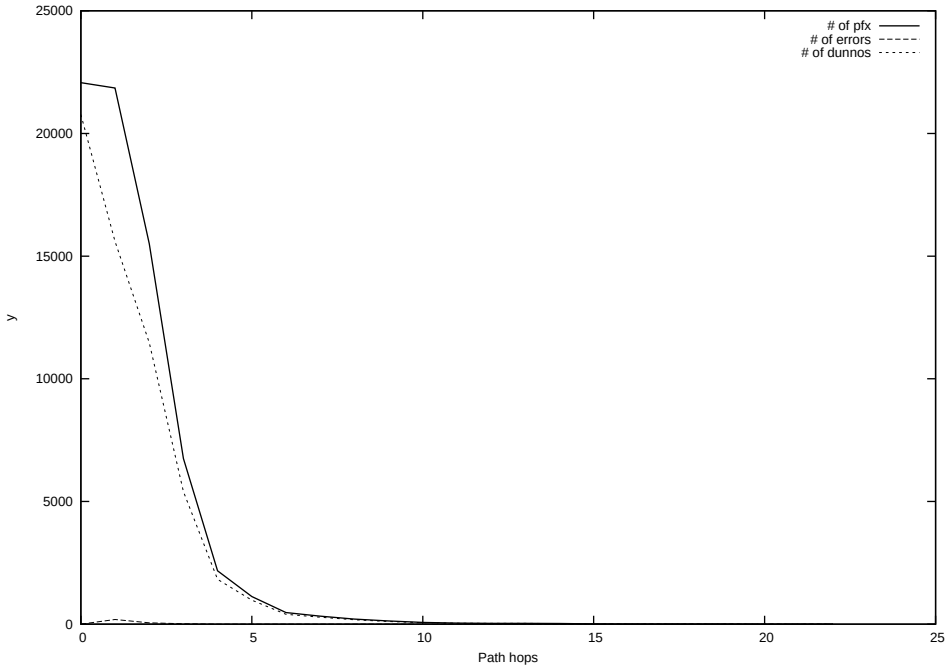
2015-04-05



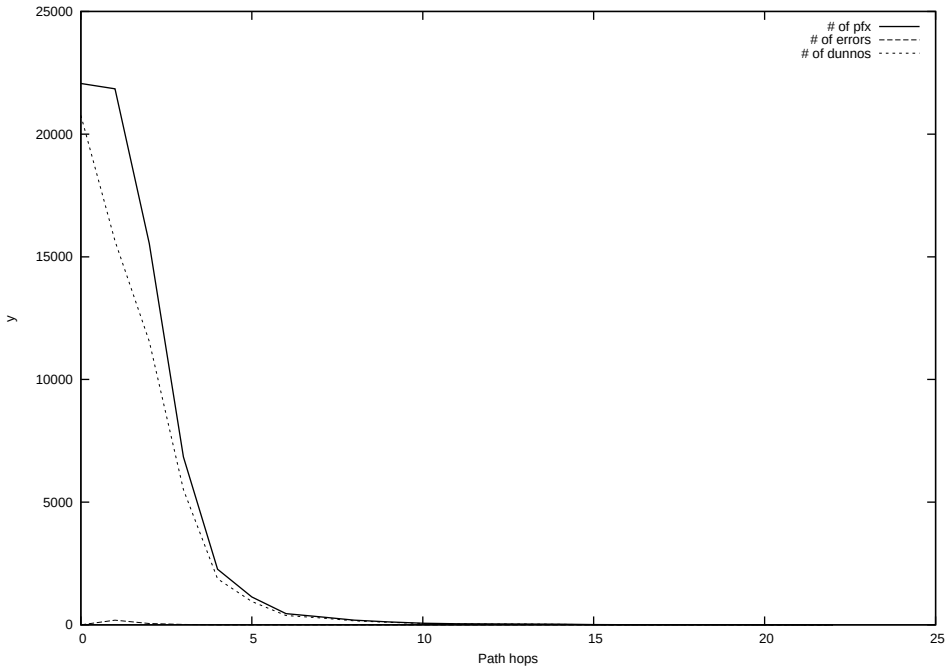
2015-04-06



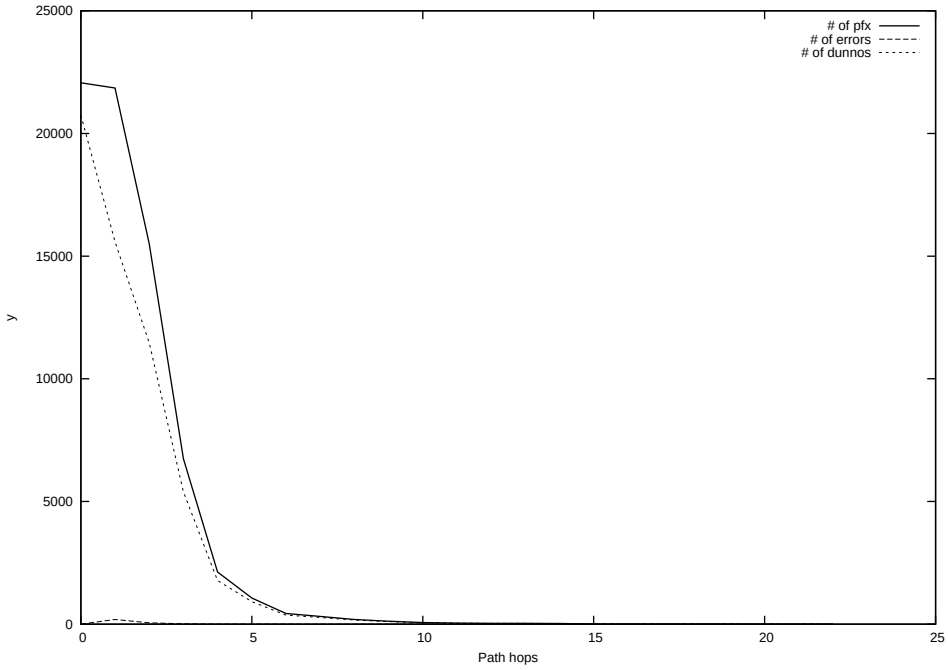
2015-04-07



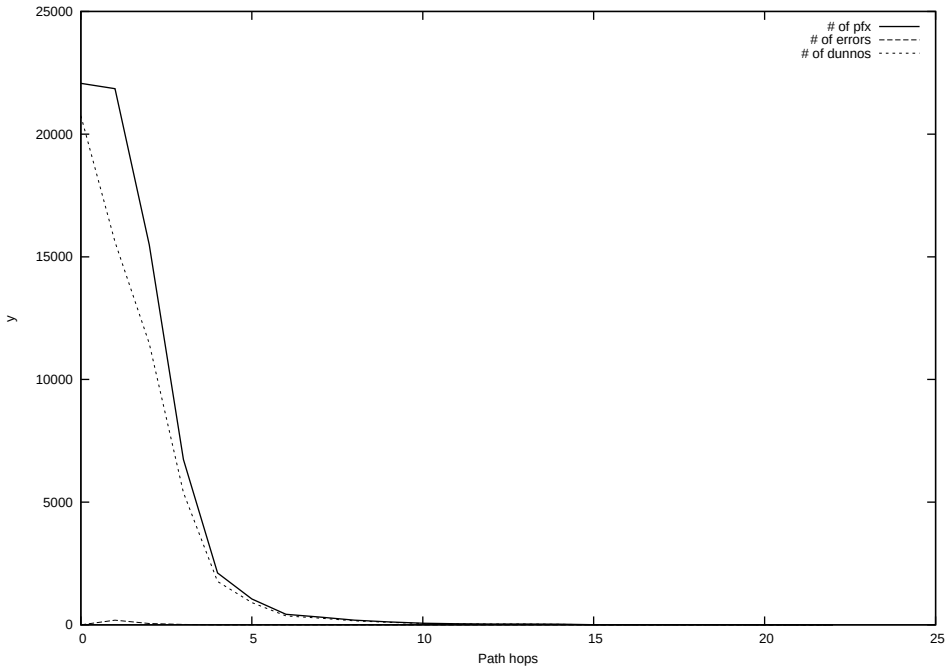
2015-04-08



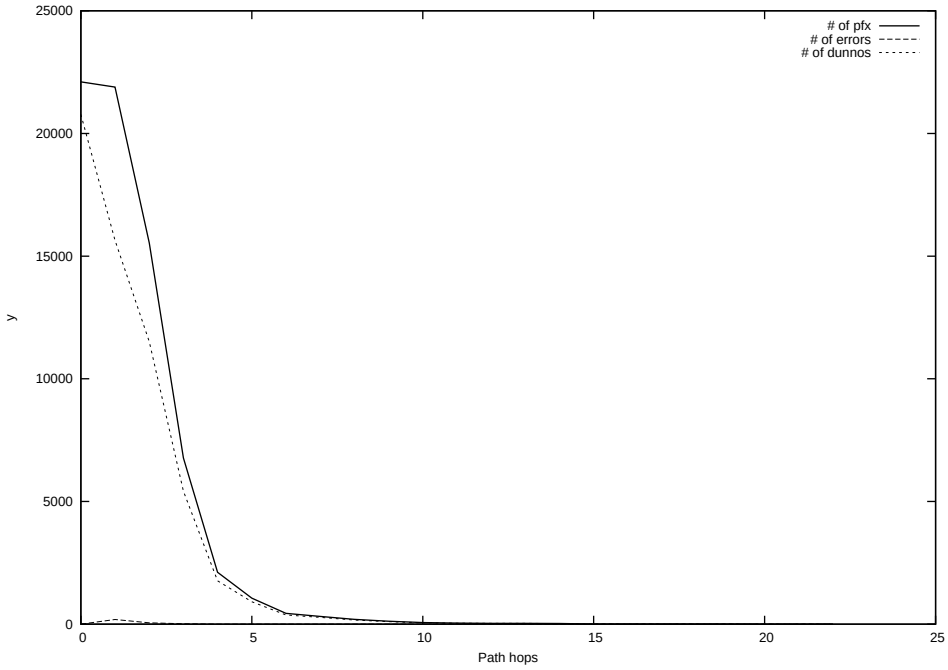
2015-04-09



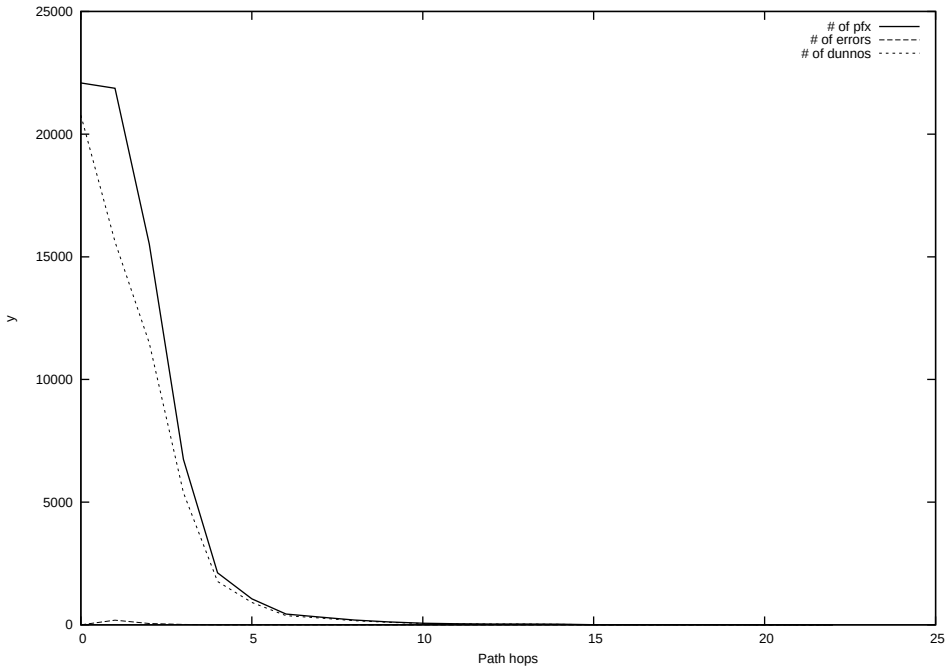
2015-04-10



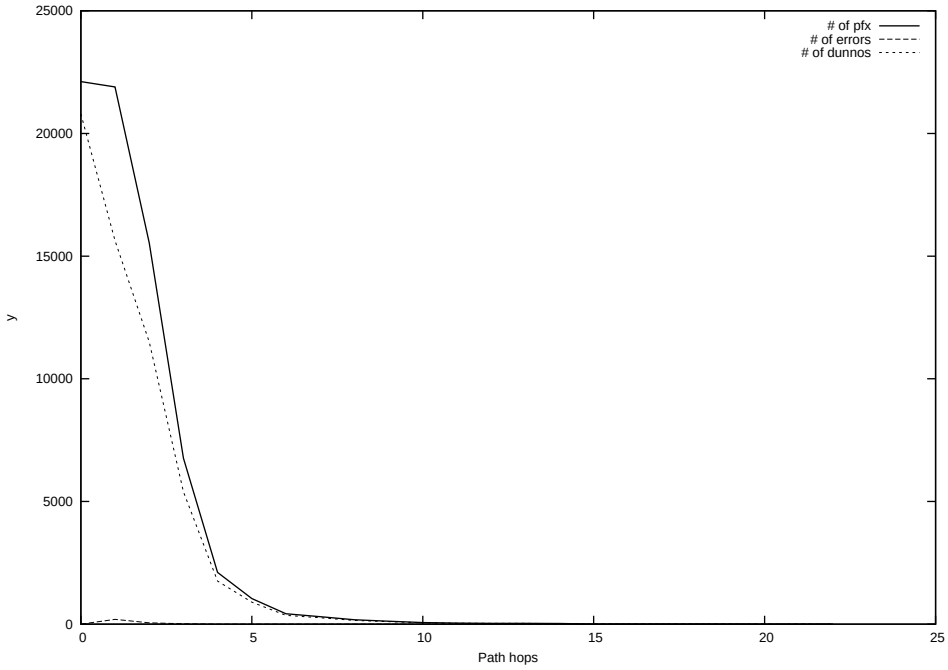
2015-04-11



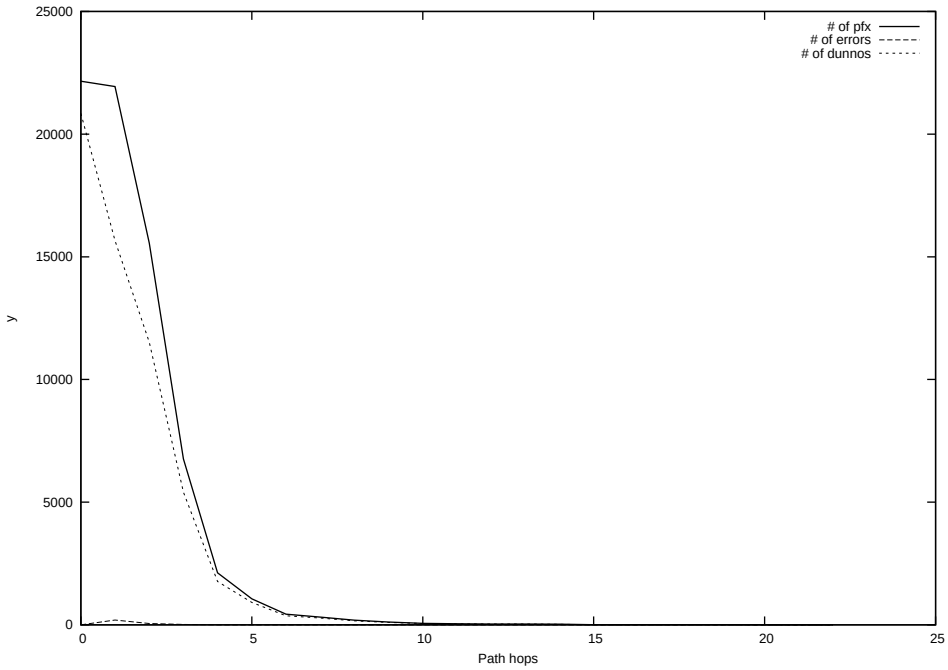
2015-04-12



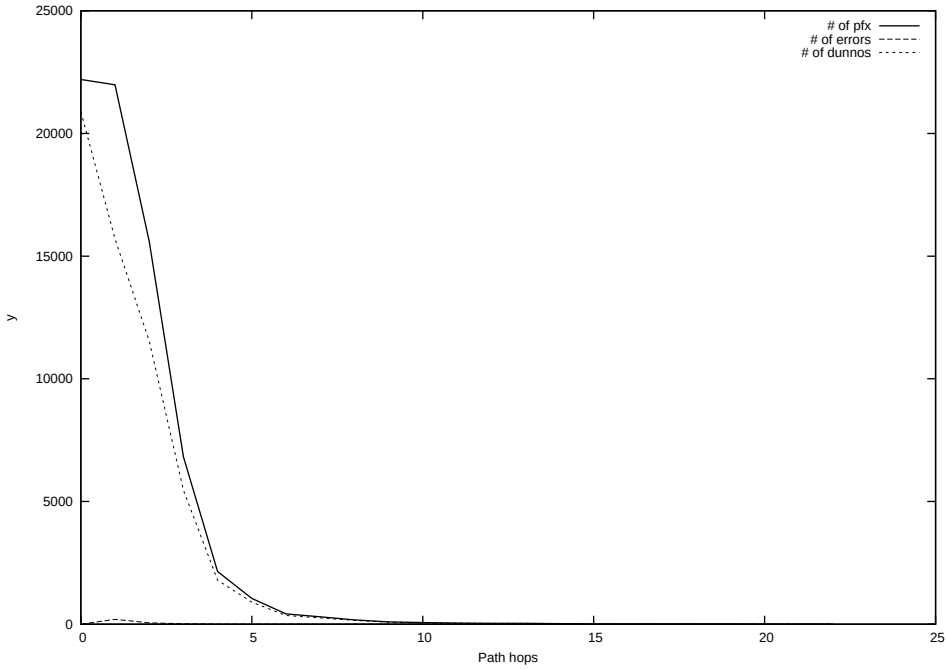
2015-04-13



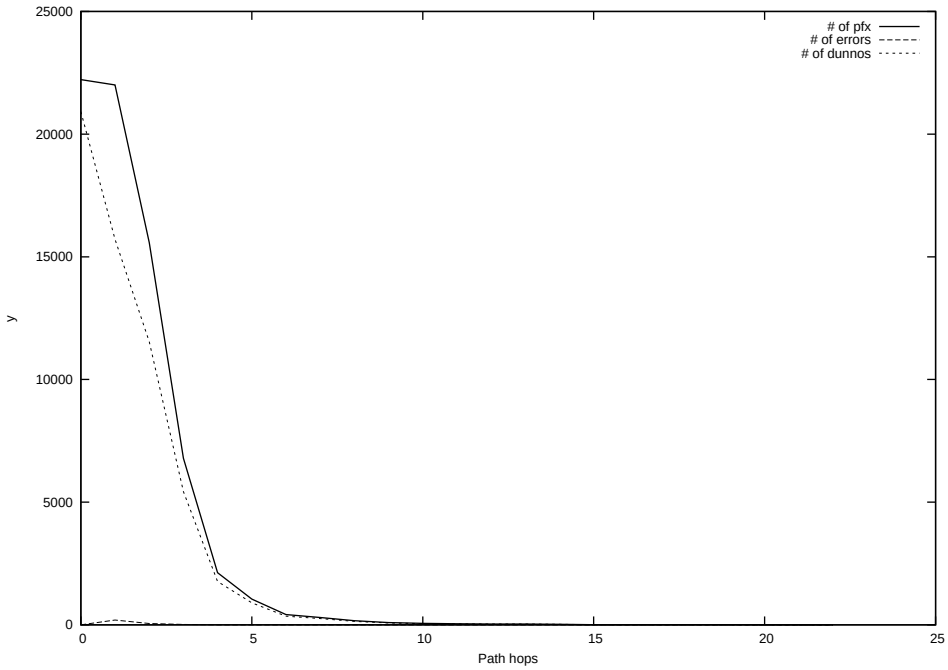
2015-04-14



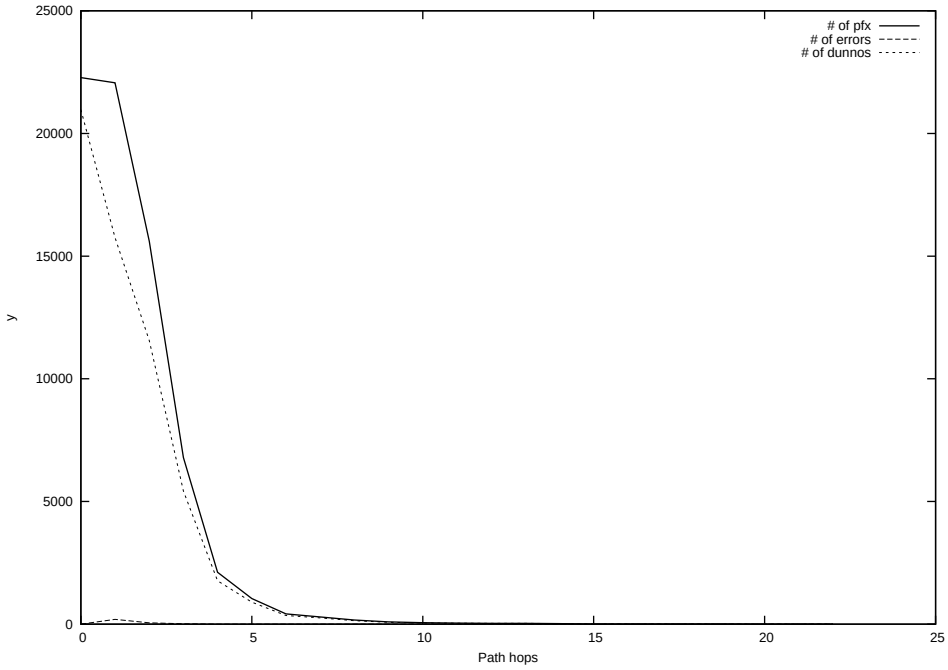
2015-04-15



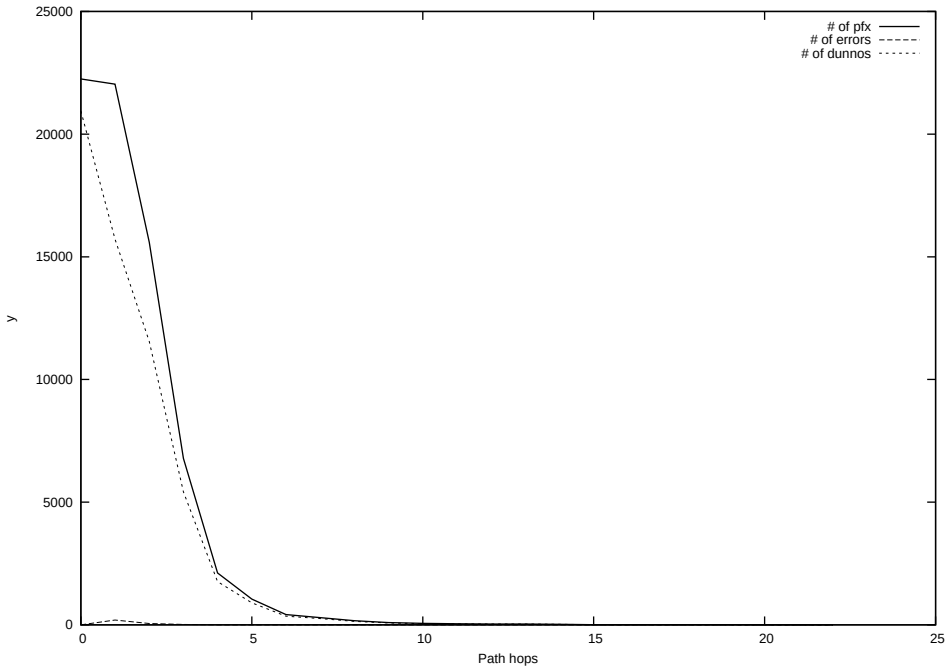
2015-04-16



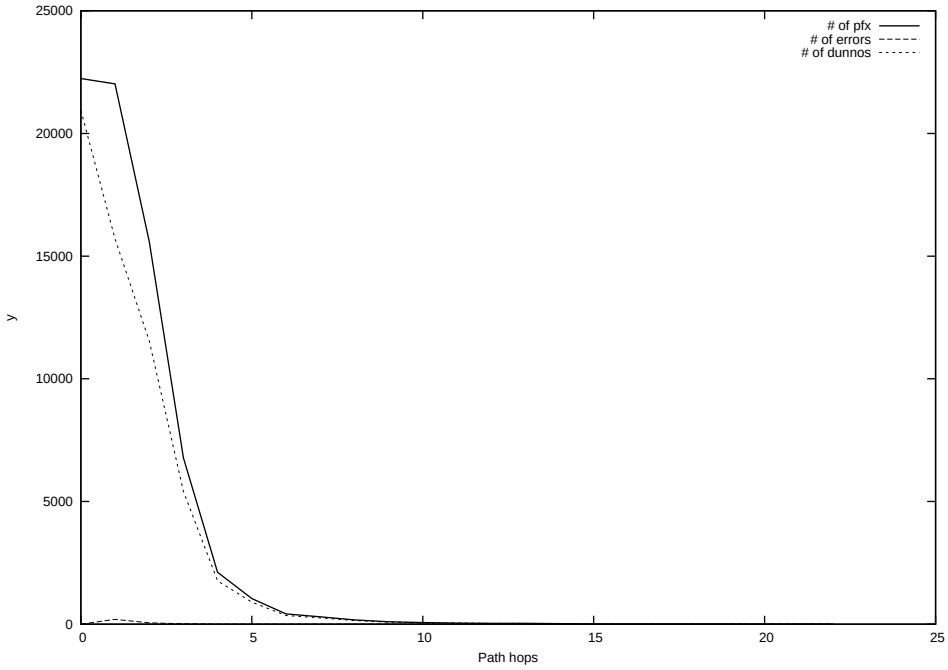
2015-04-17



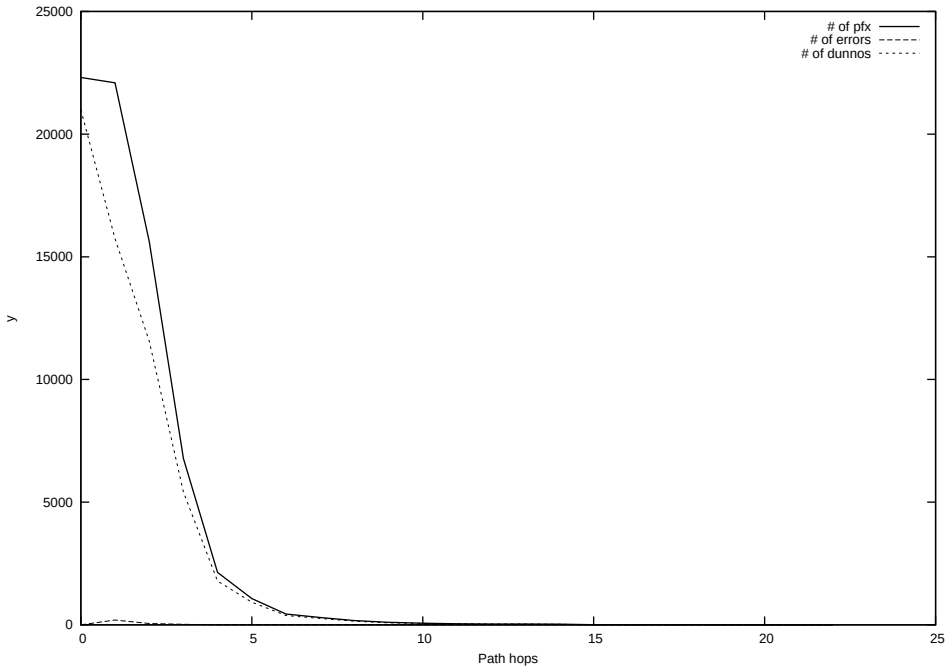
2015-04-18



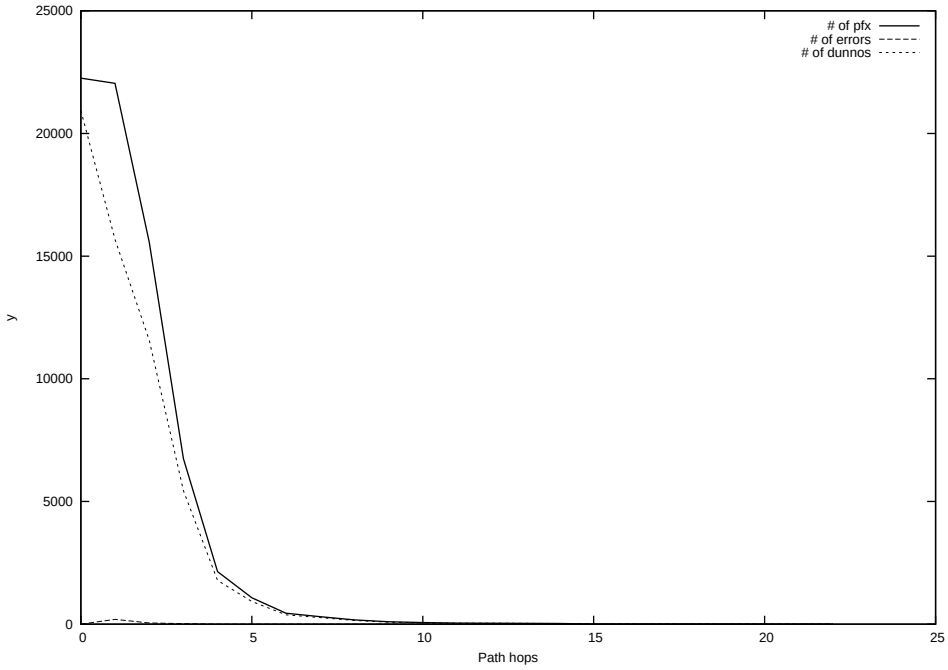
2015-04-19



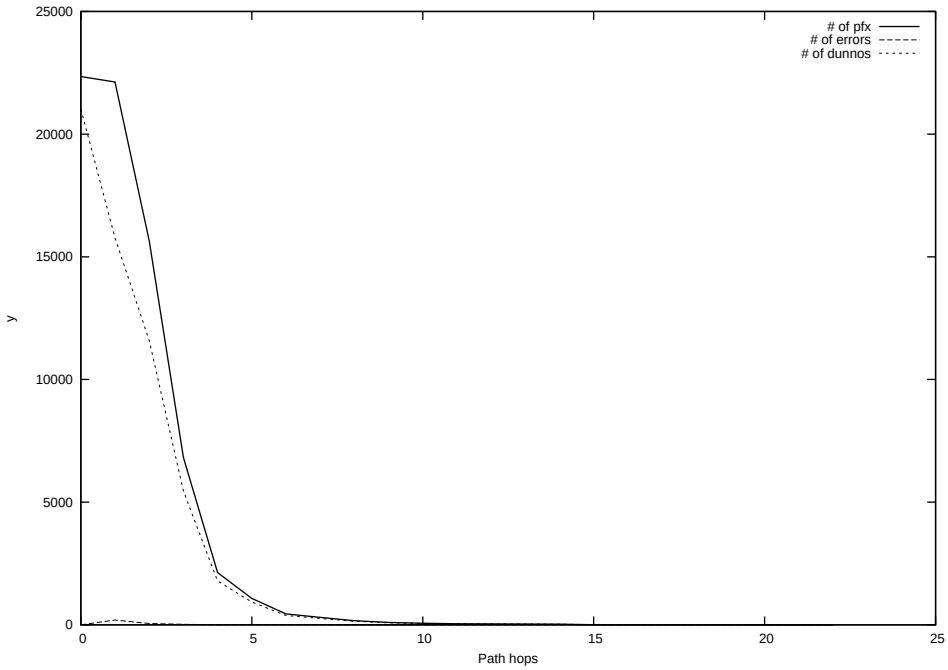
2015-04-20



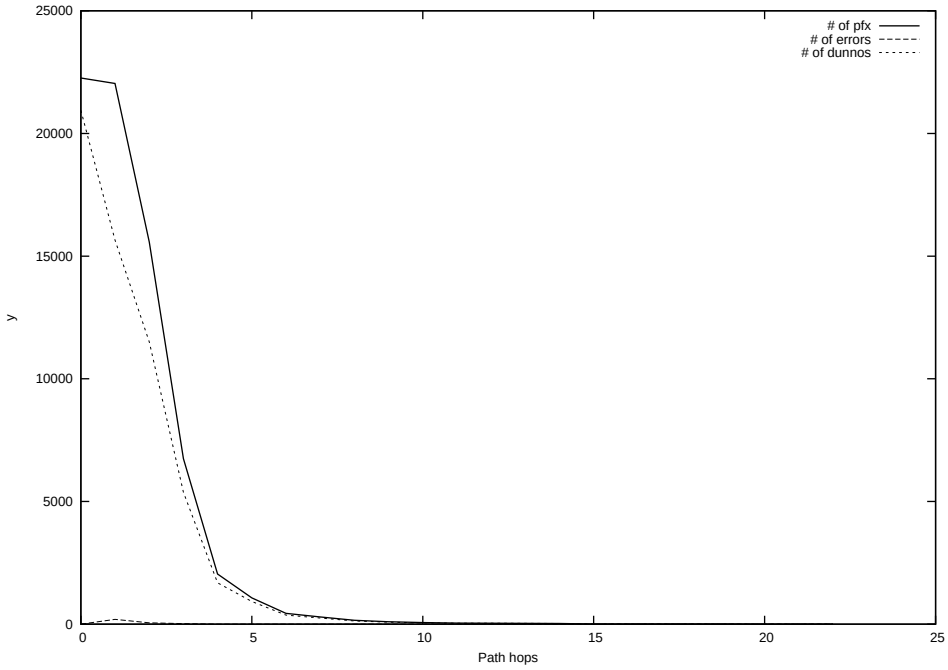
2015-04-21



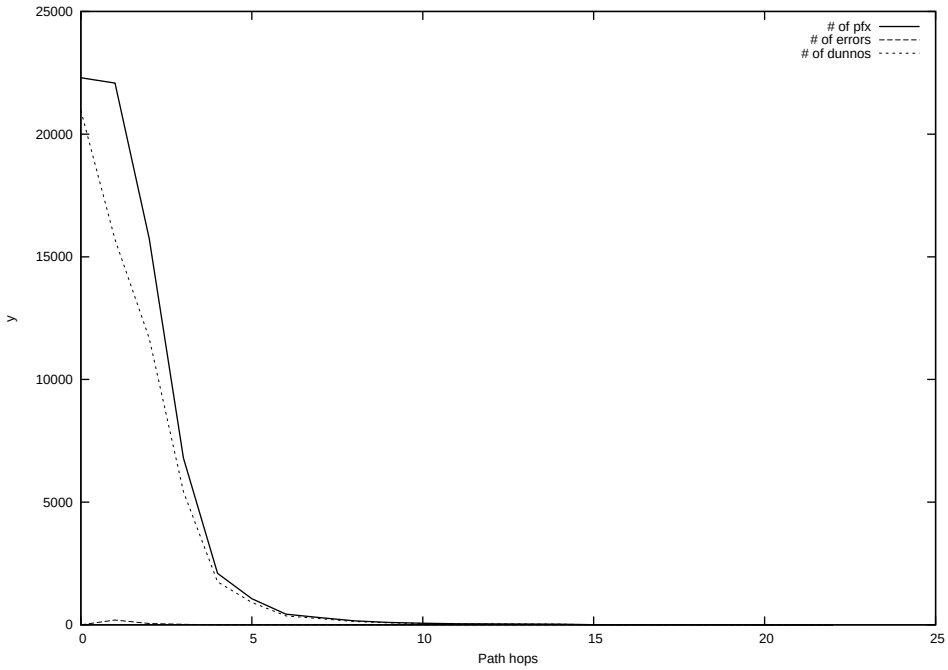
2015-04-22



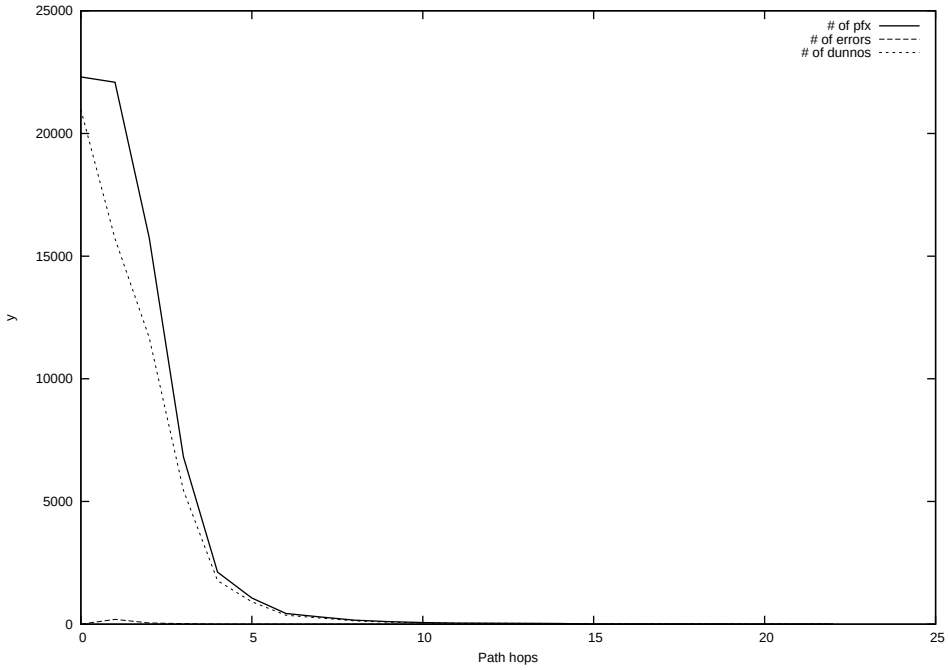
2015-04-23



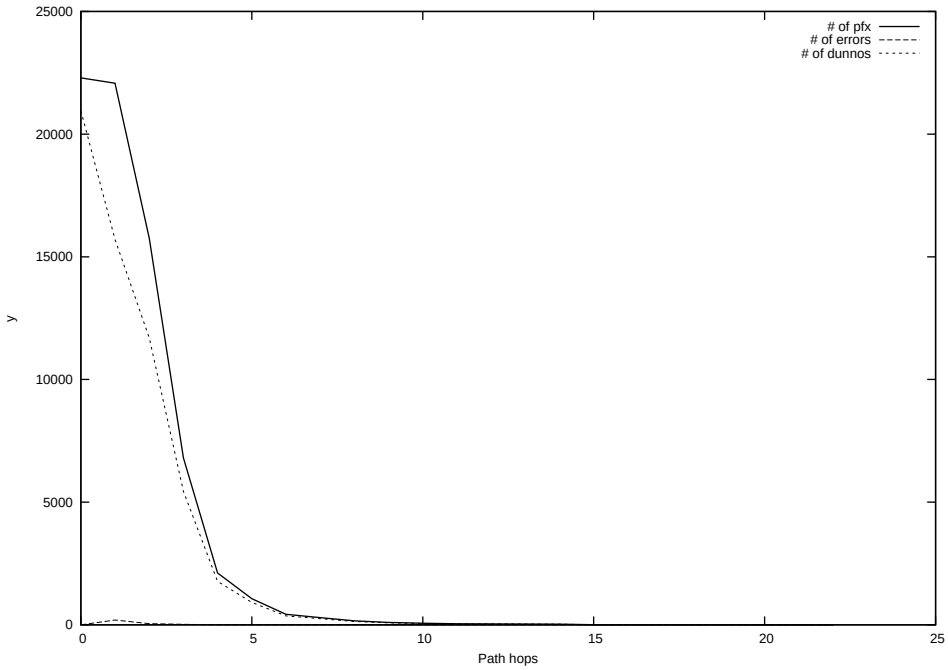
2015-04-24



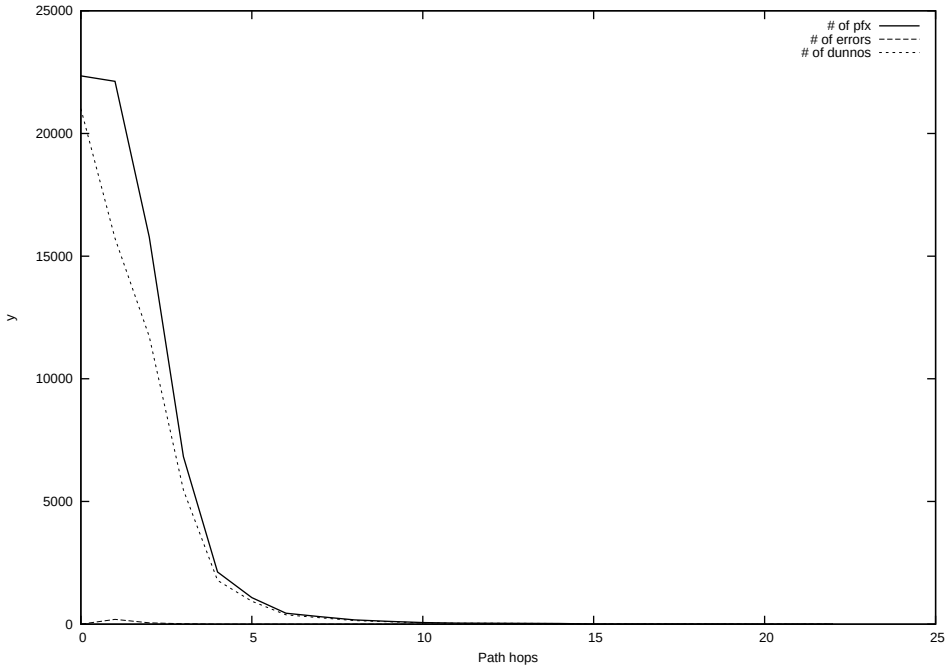
2015-04-25



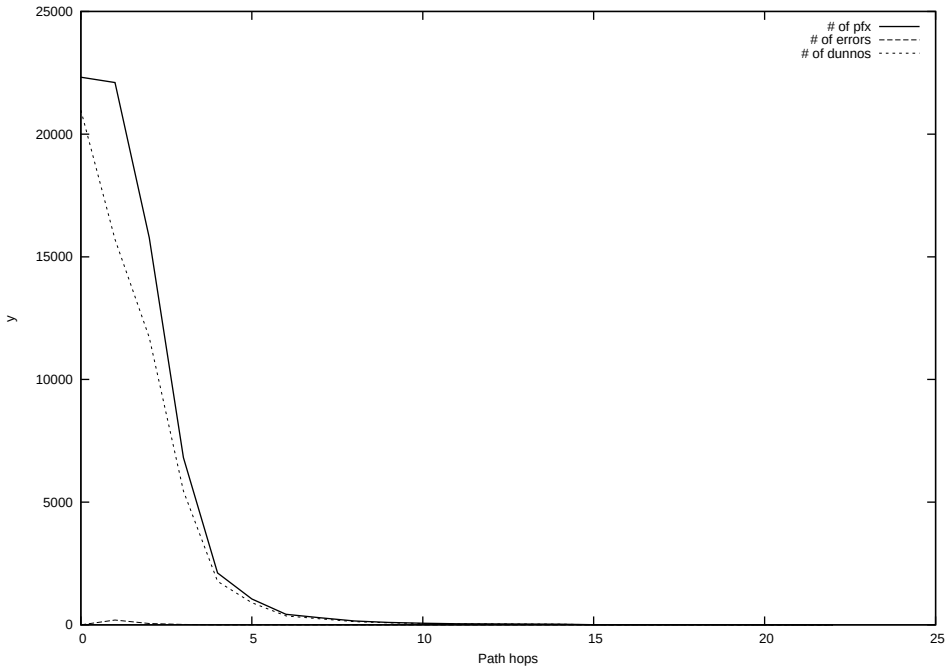
2015-04-26



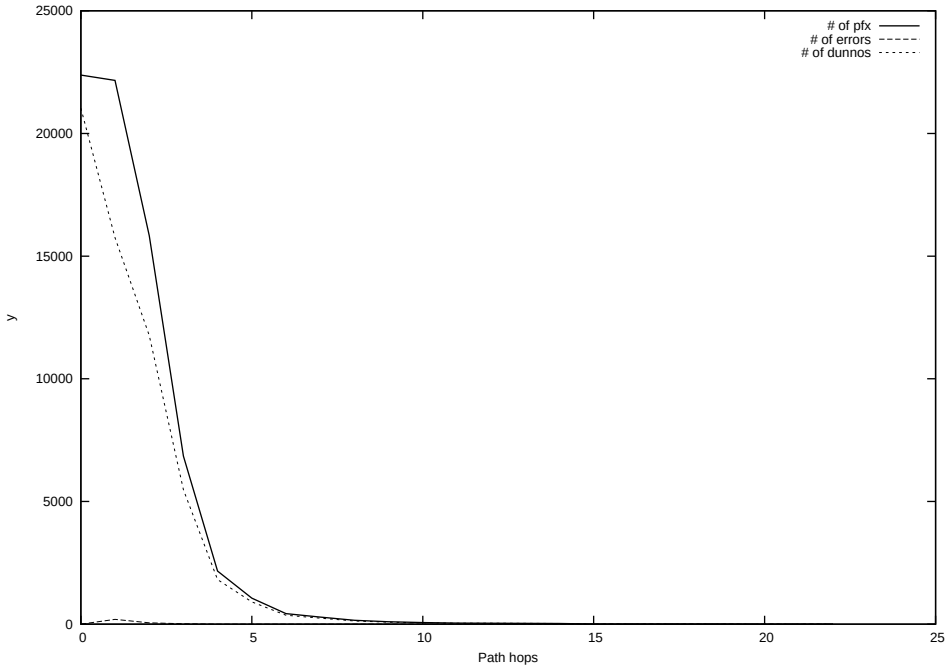
2015-04-27



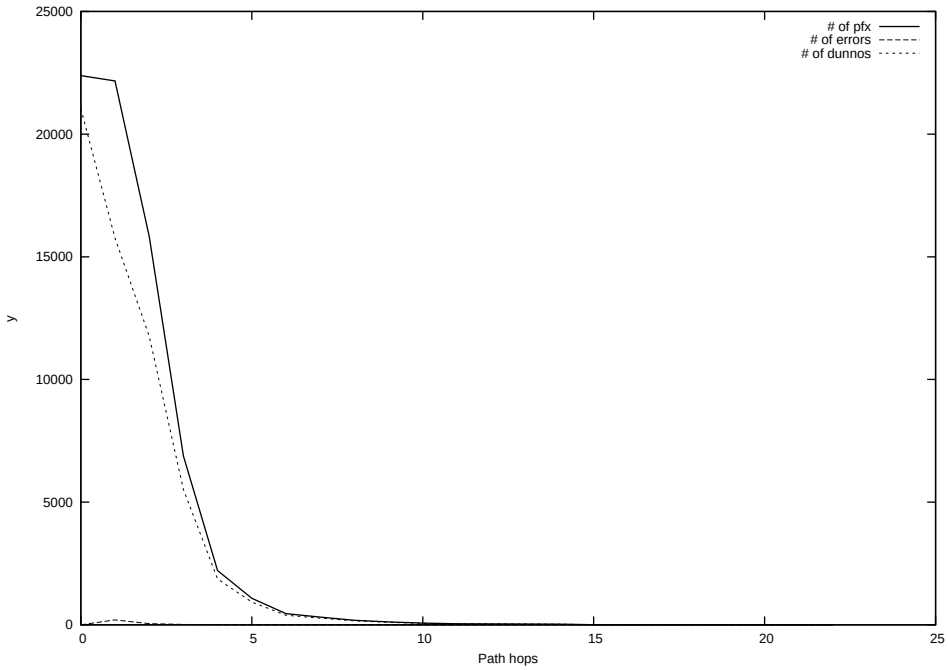
2015-04-28



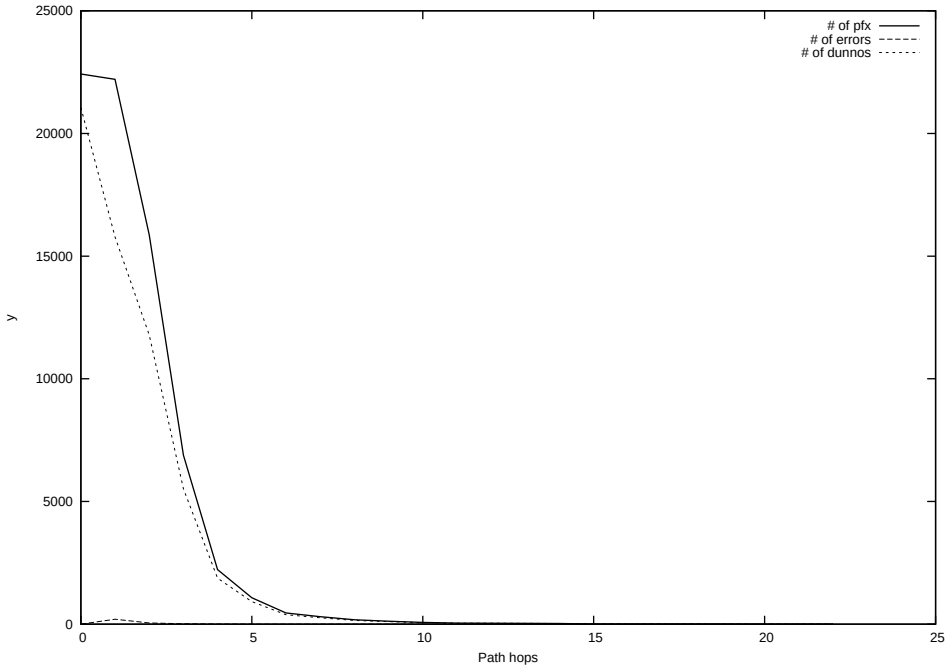
2015-04-29



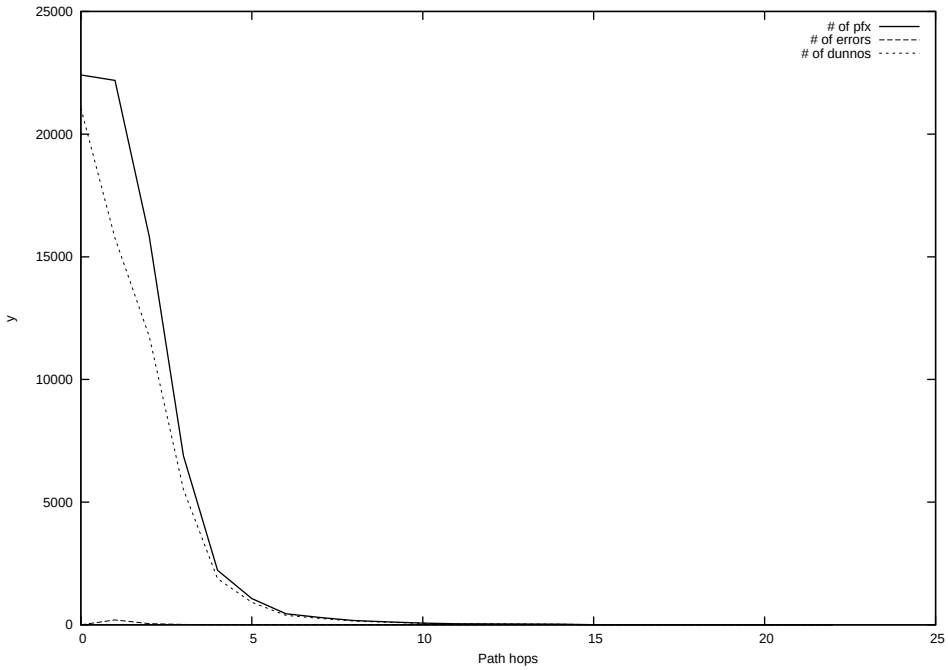
2015-04-30



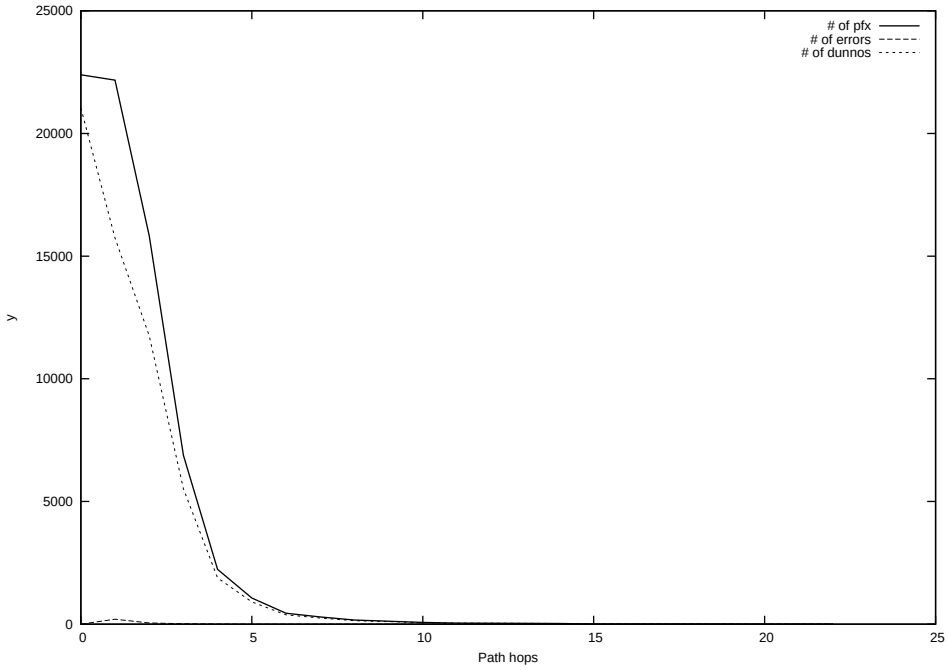
2015-05-01



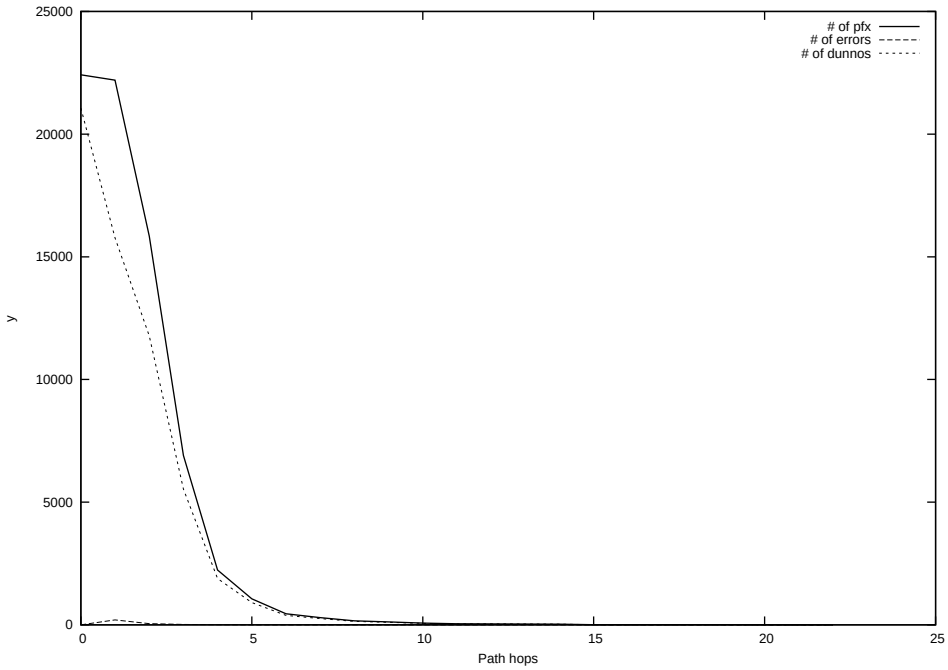
2015-05-02



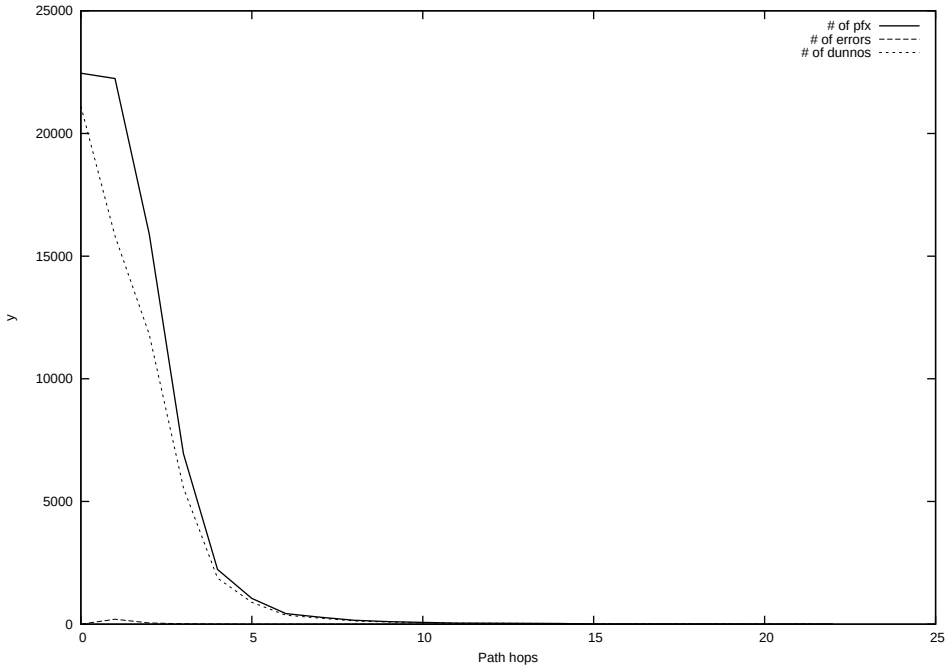
2015-05-03



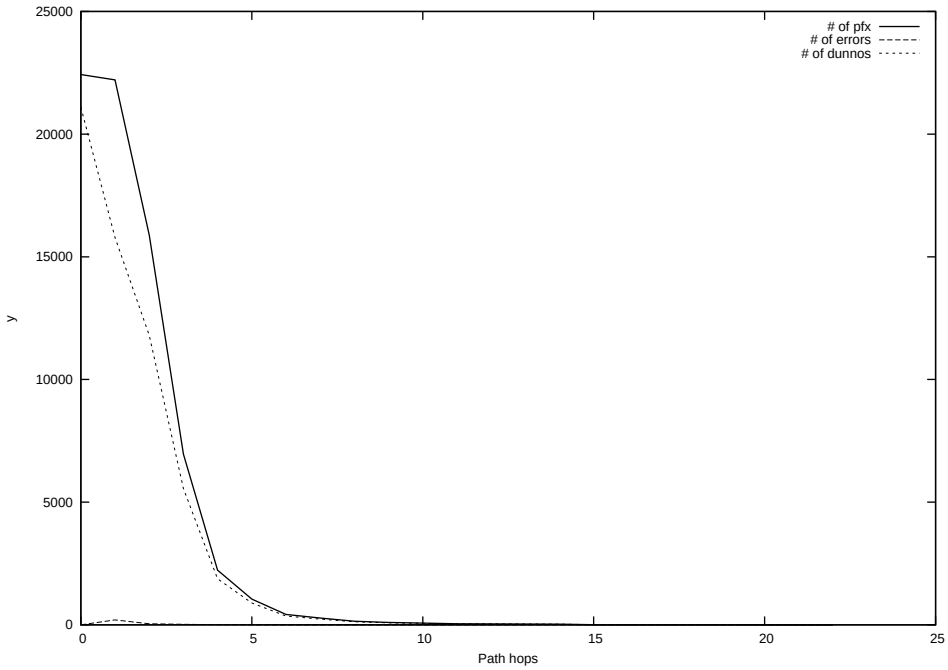
2015-05-04



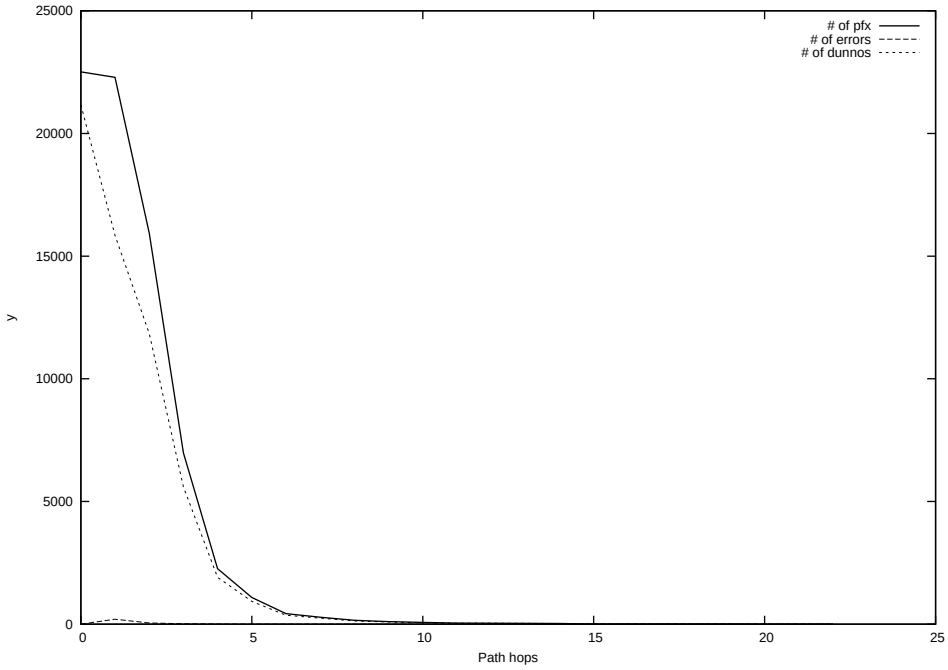
2015-05-05



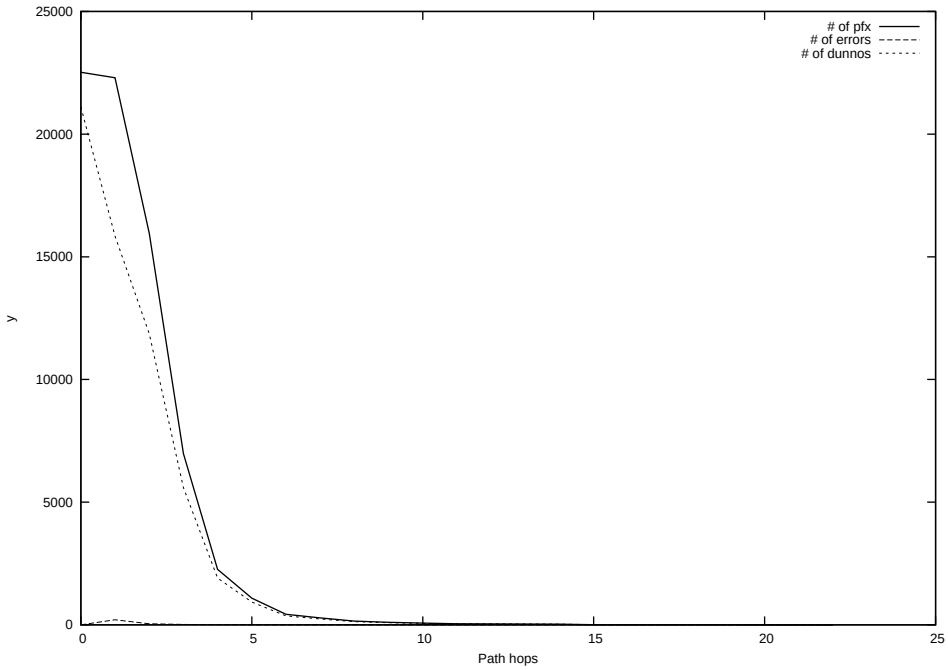
2015-05-06



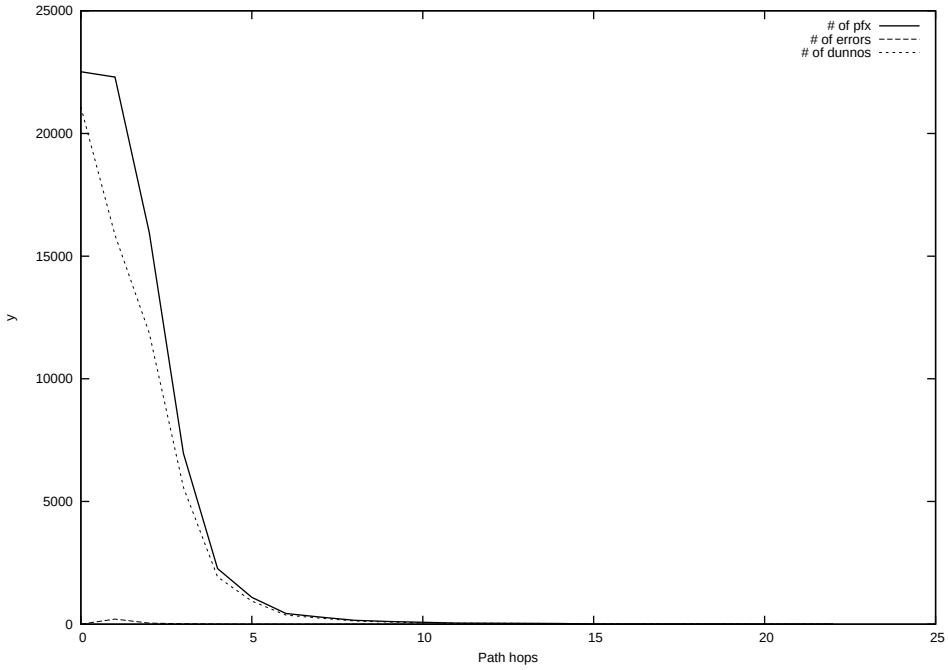
2015-05-07



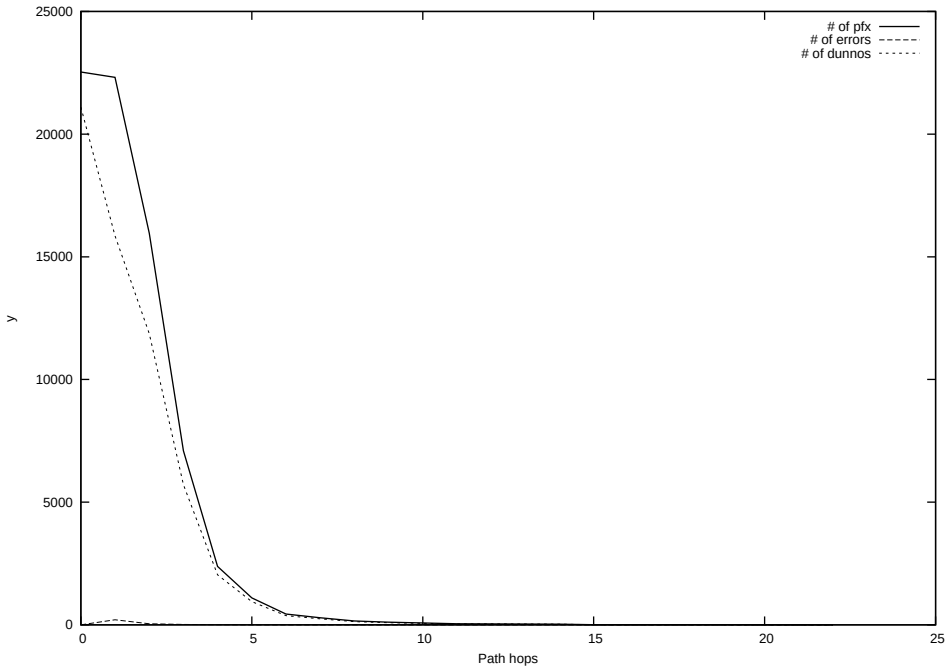
2015-05-08



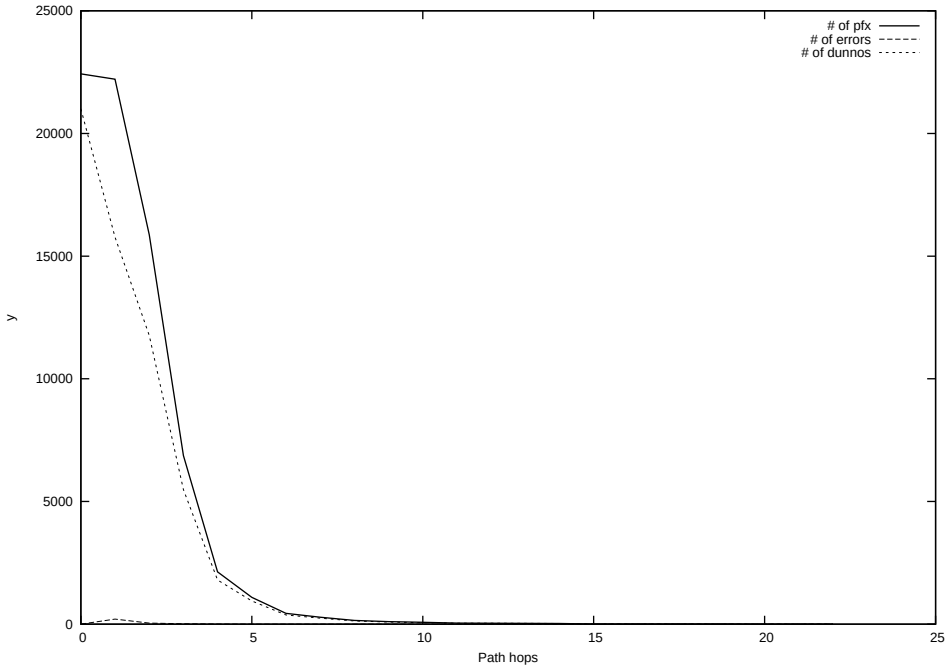
2015-05-09



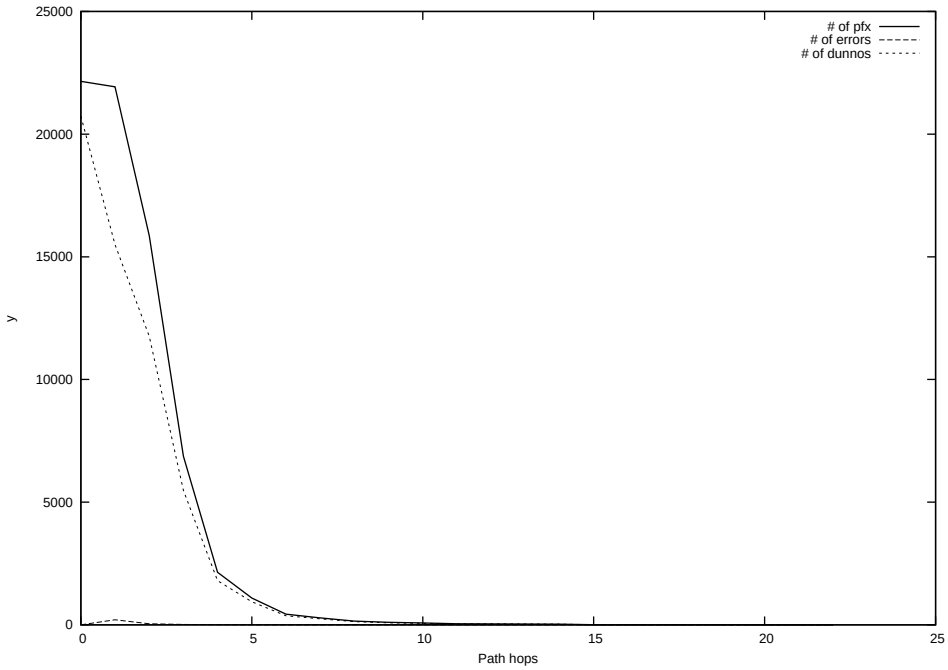
2015-05-10



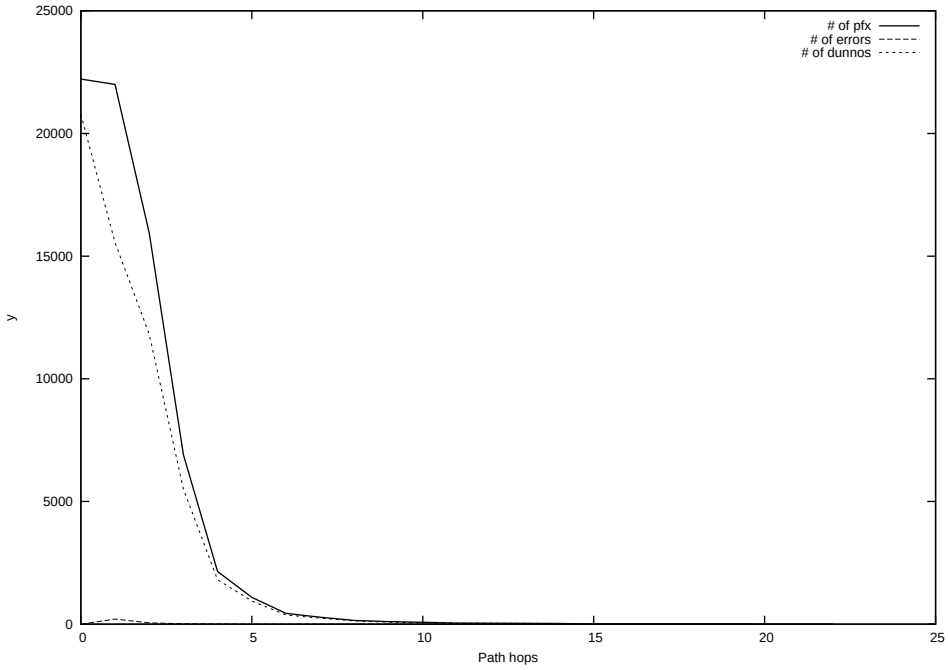
2015-05-11



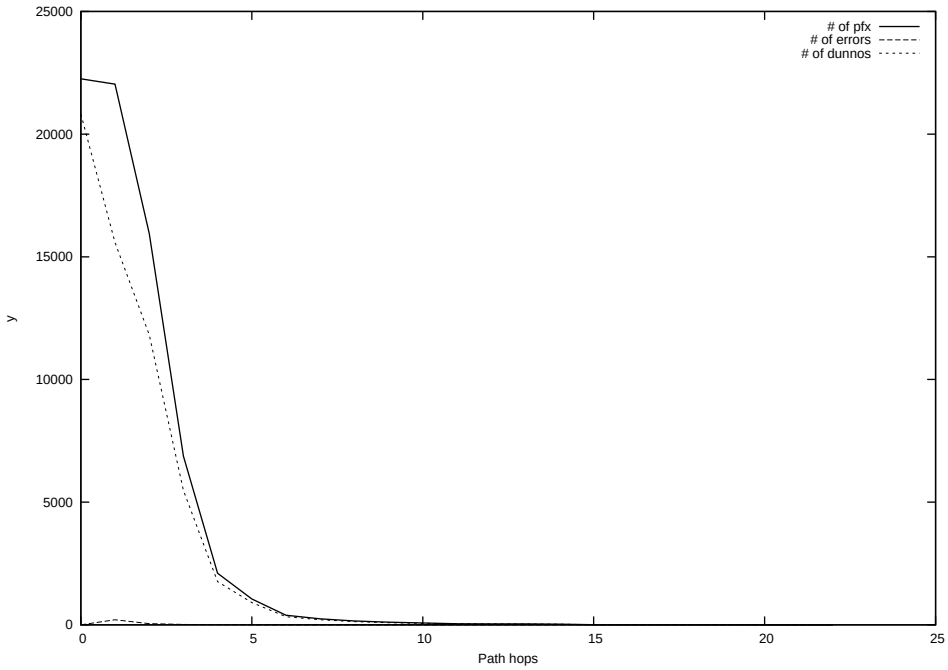
2015-05-12



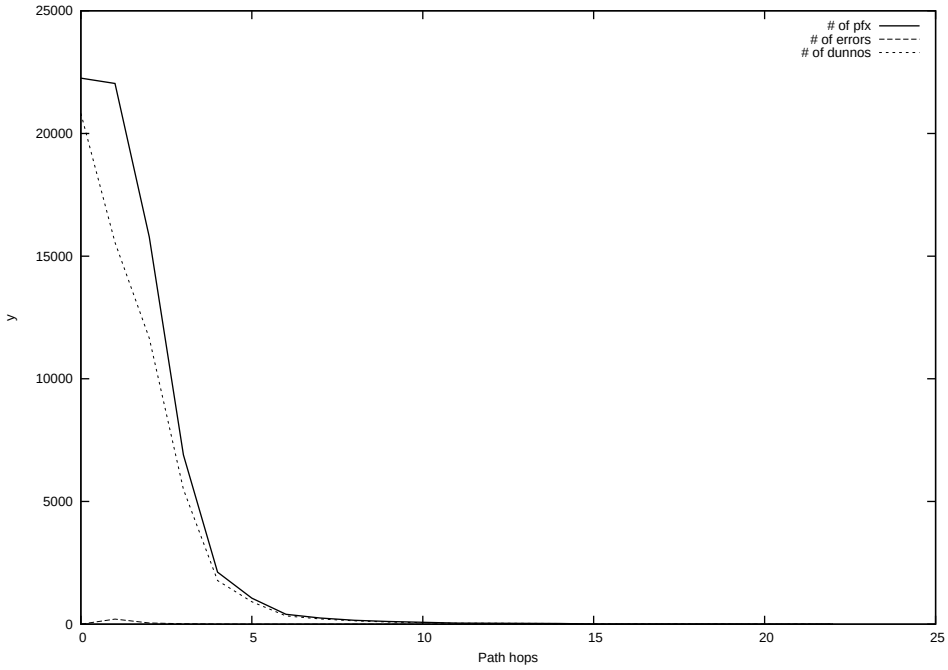
2015-05-13



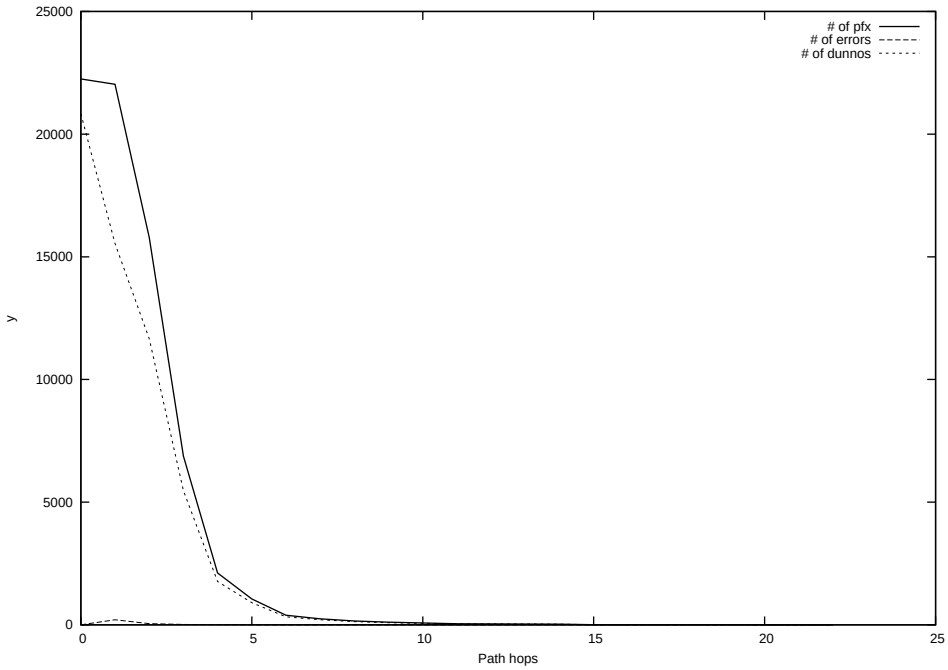
2015-05-14



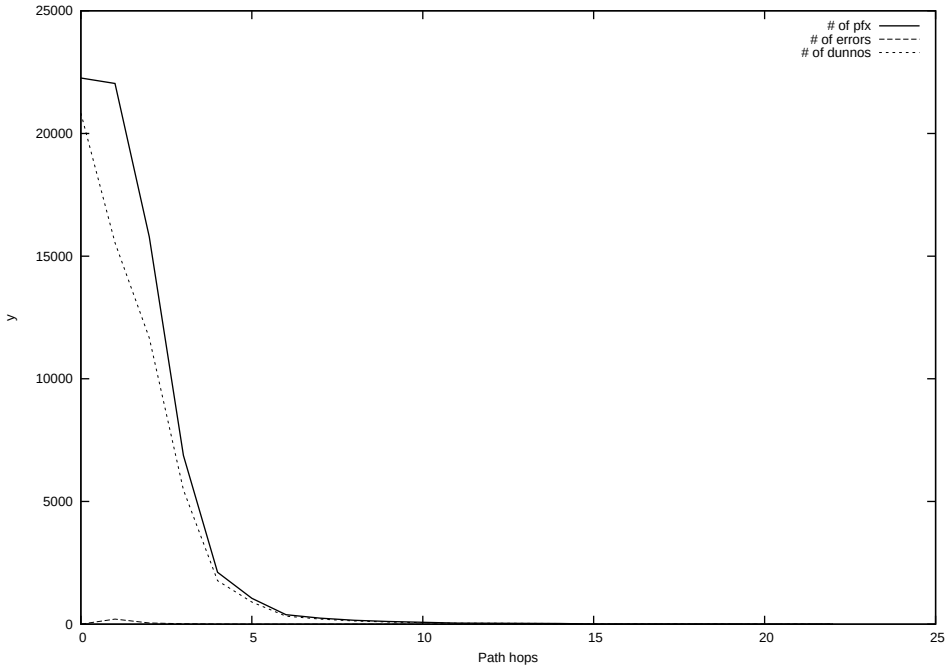
2015-05-15



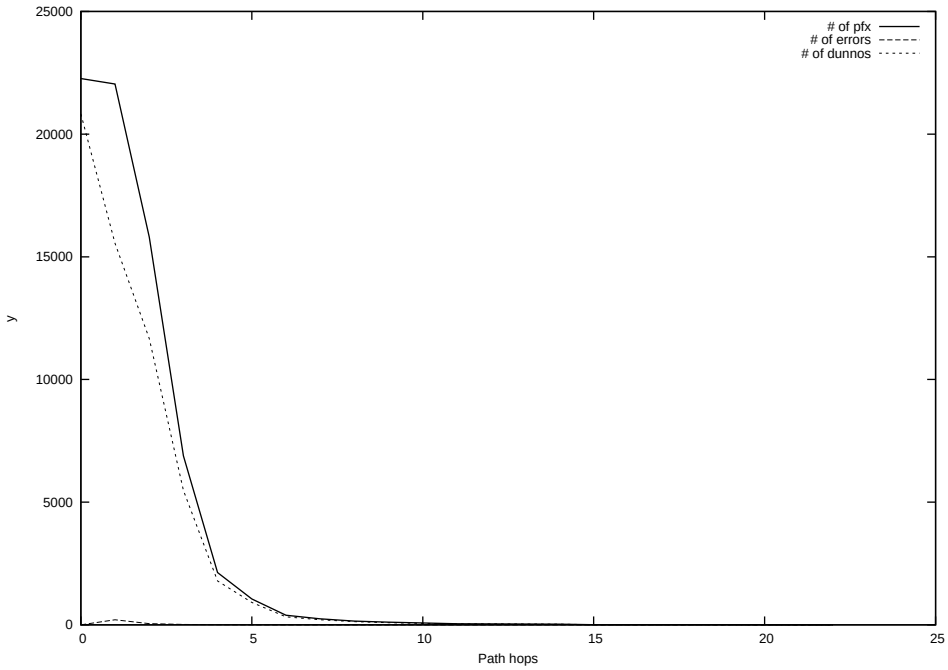
2015-05-16



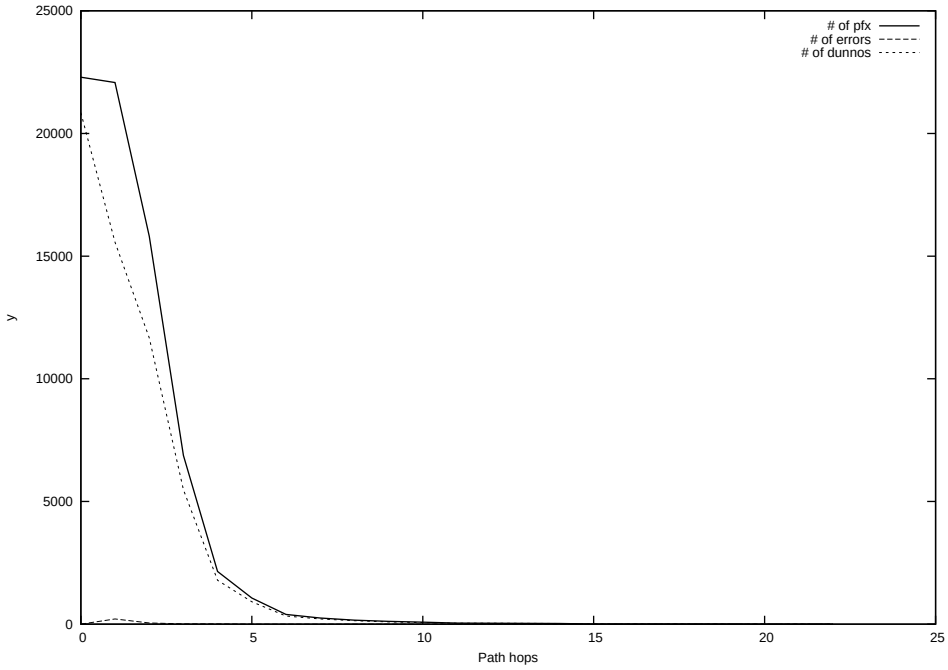
2015-05-17



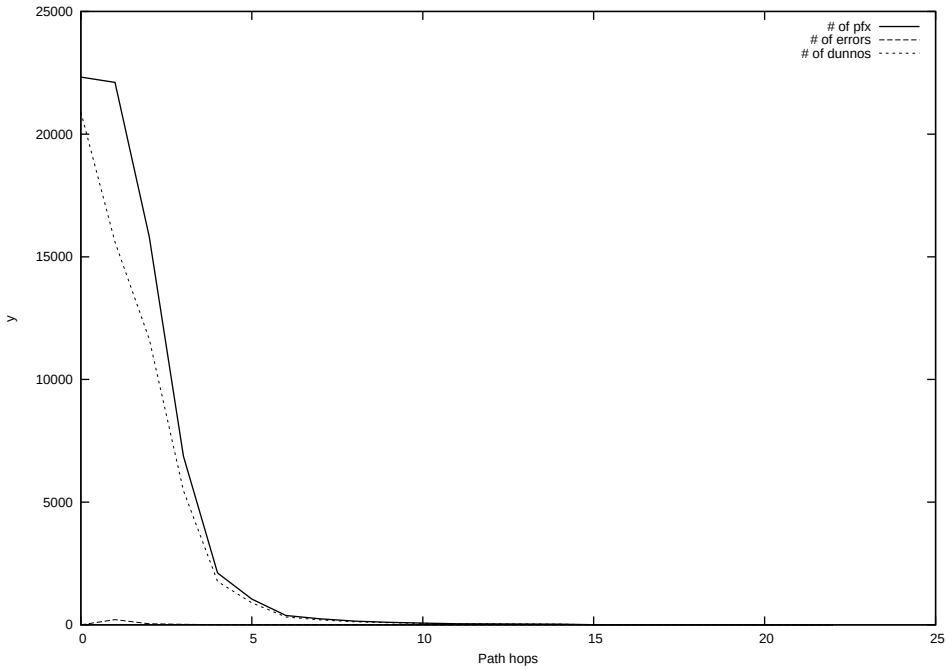
2015-05-18



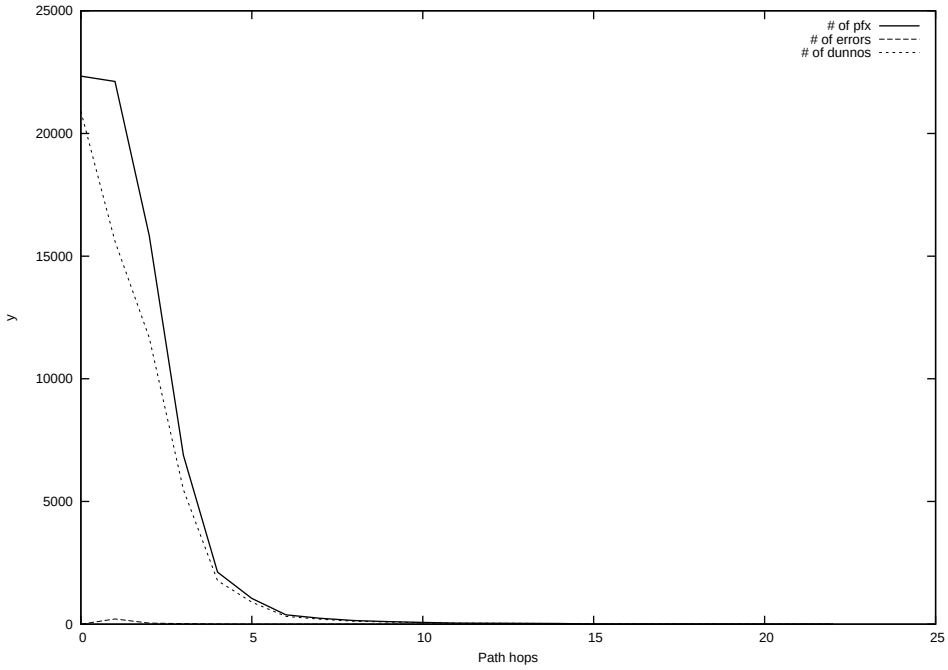
2015-05-19



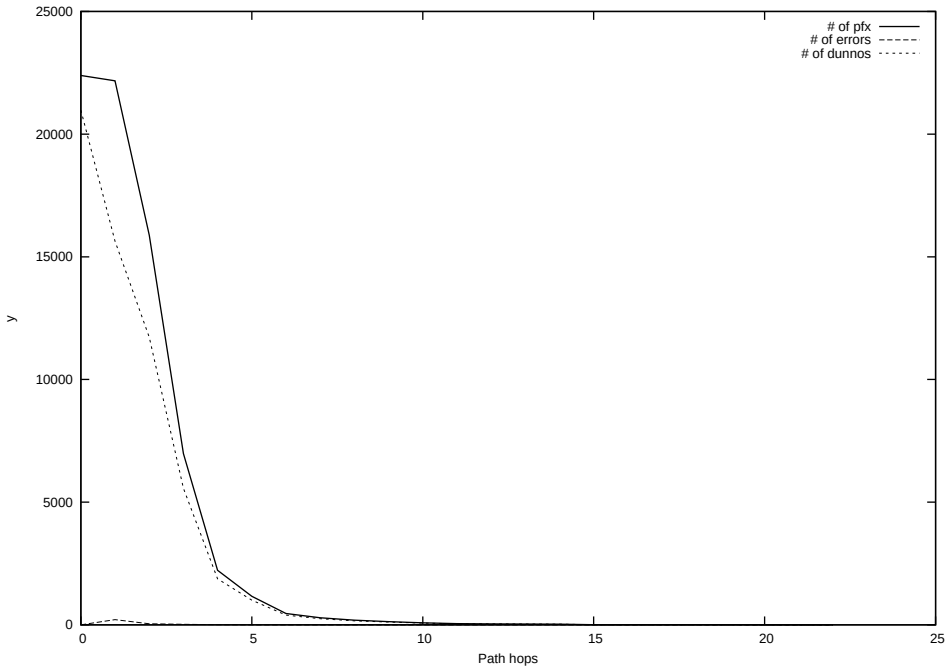
2015-05-20



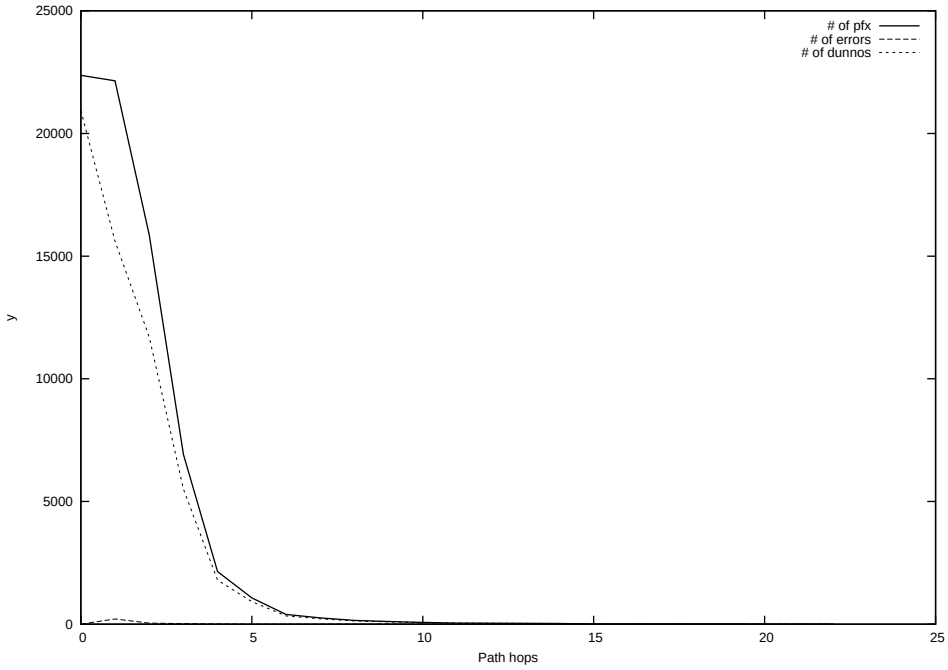
2015-05-21



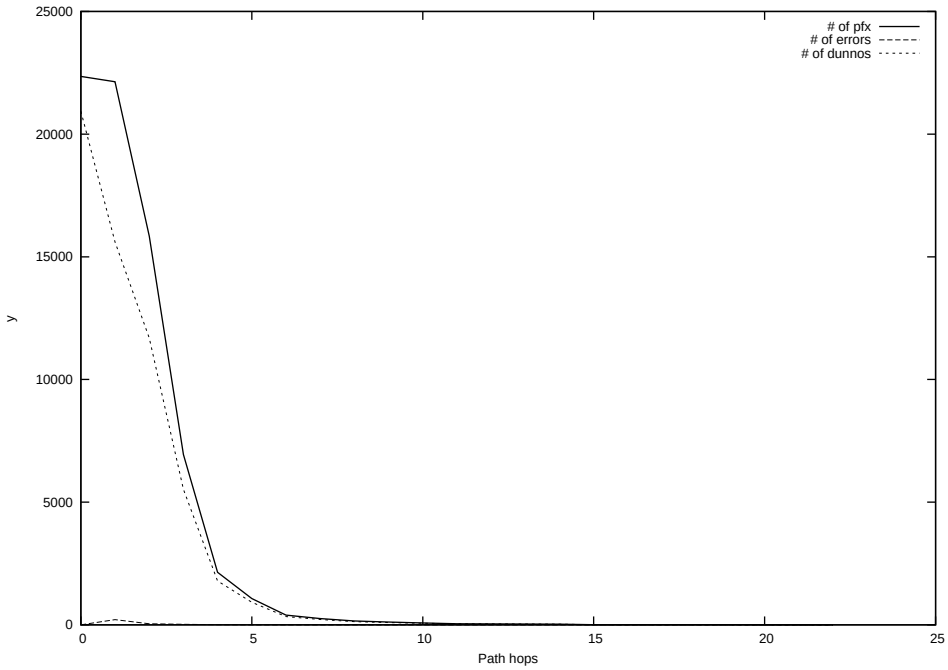
2015-05-22



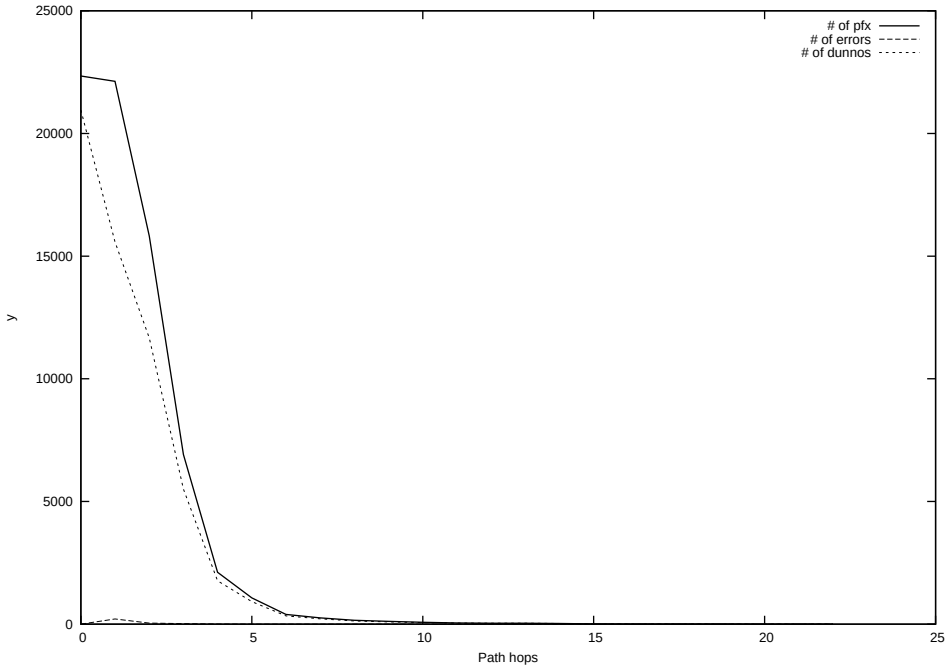
2015-05-23



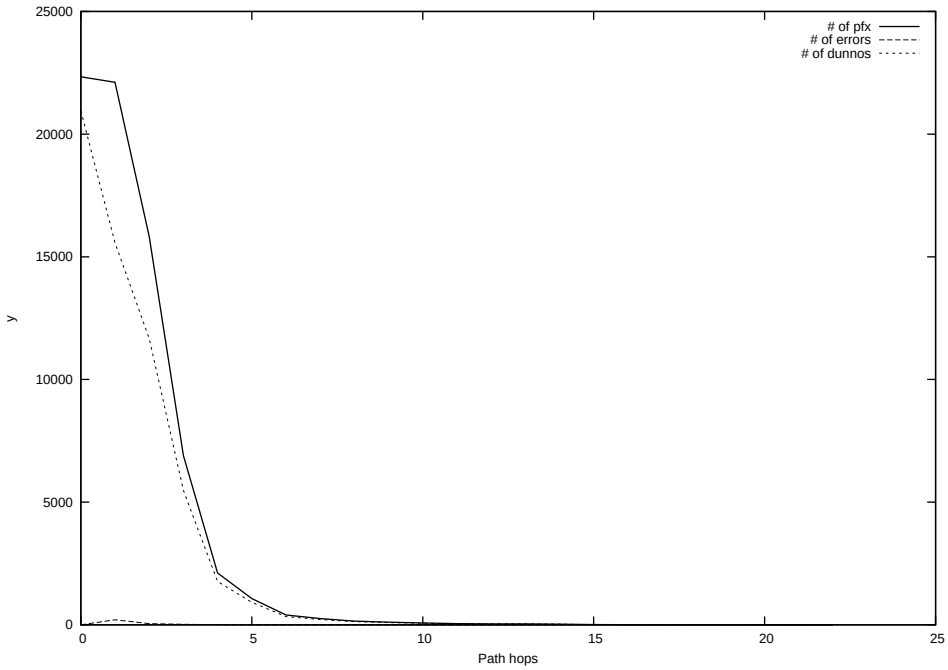
2015-05-24



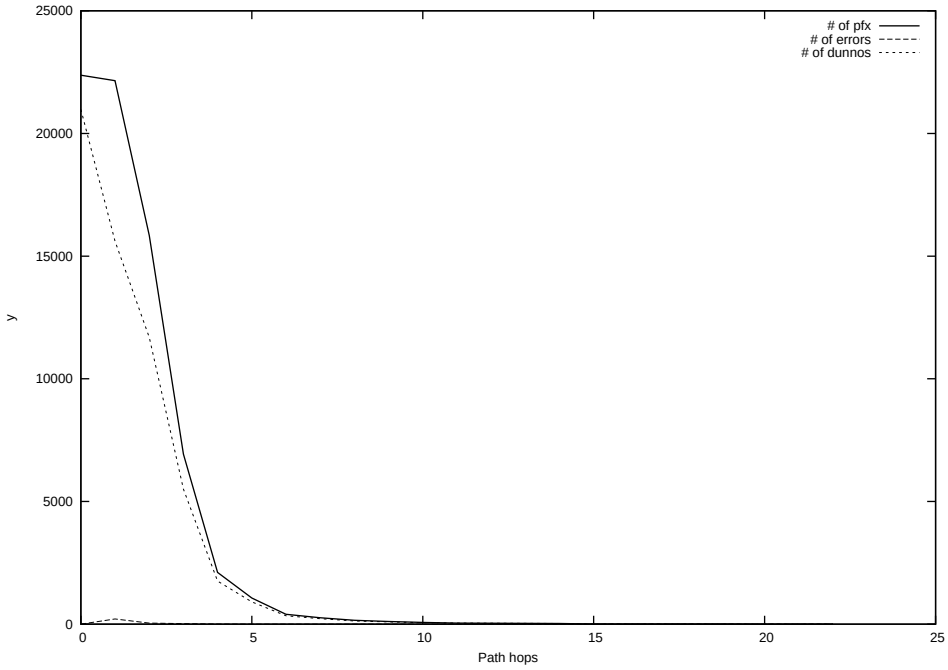
2015-05-25



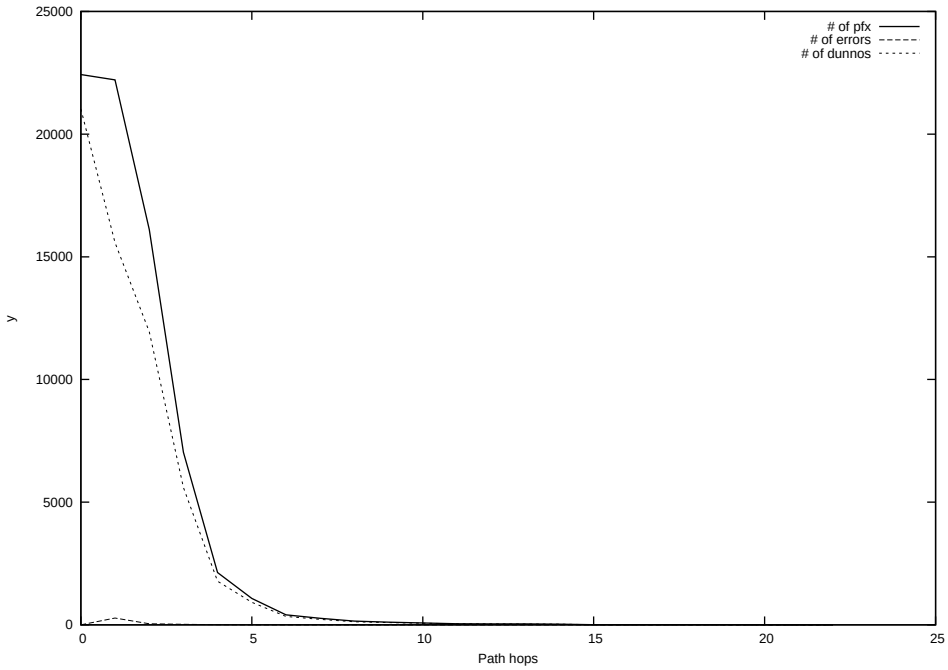
2015-05-26



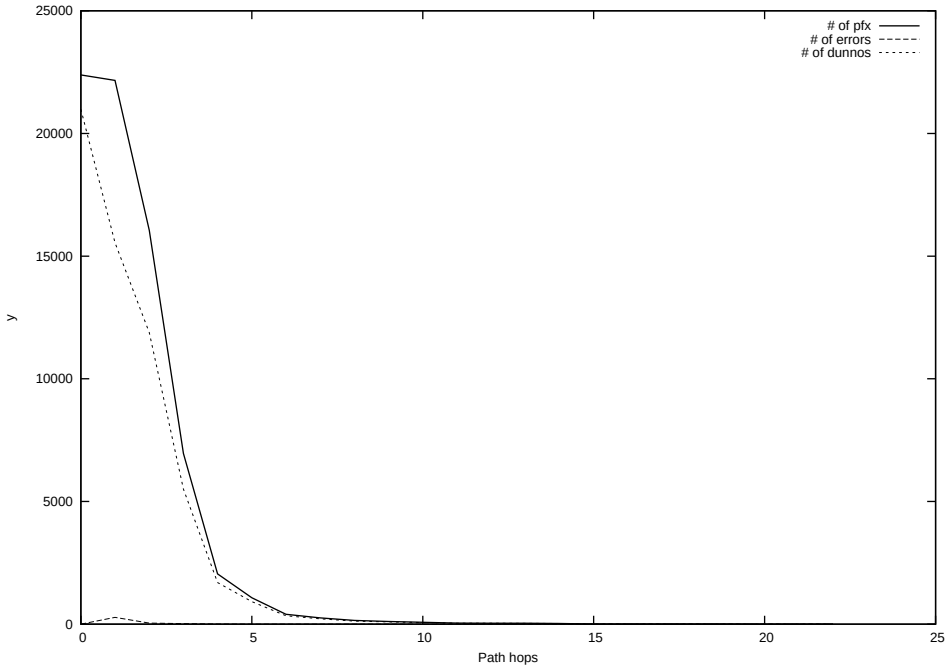
2015-05-27



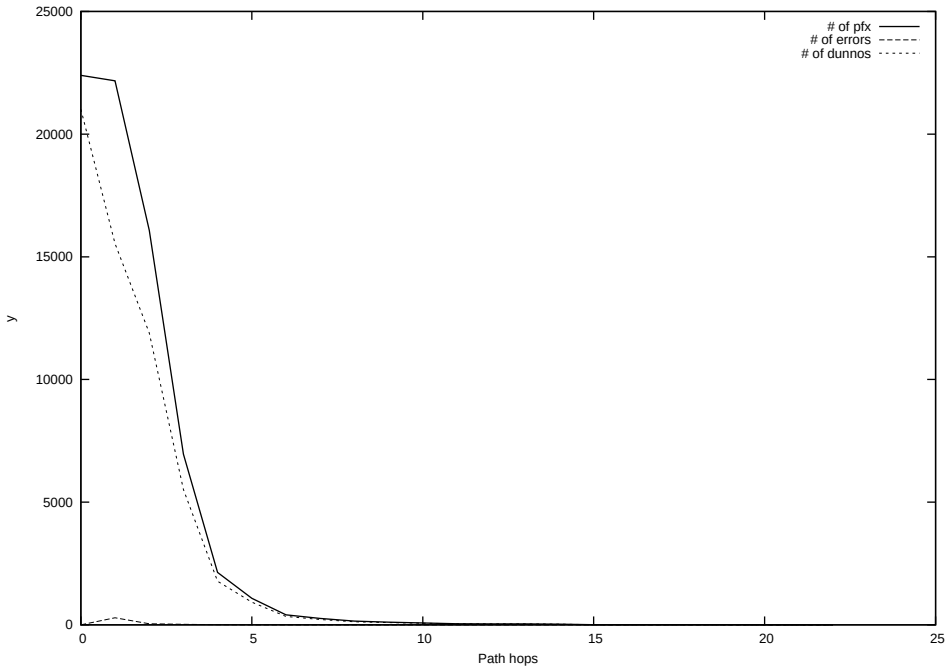
2015-05-28



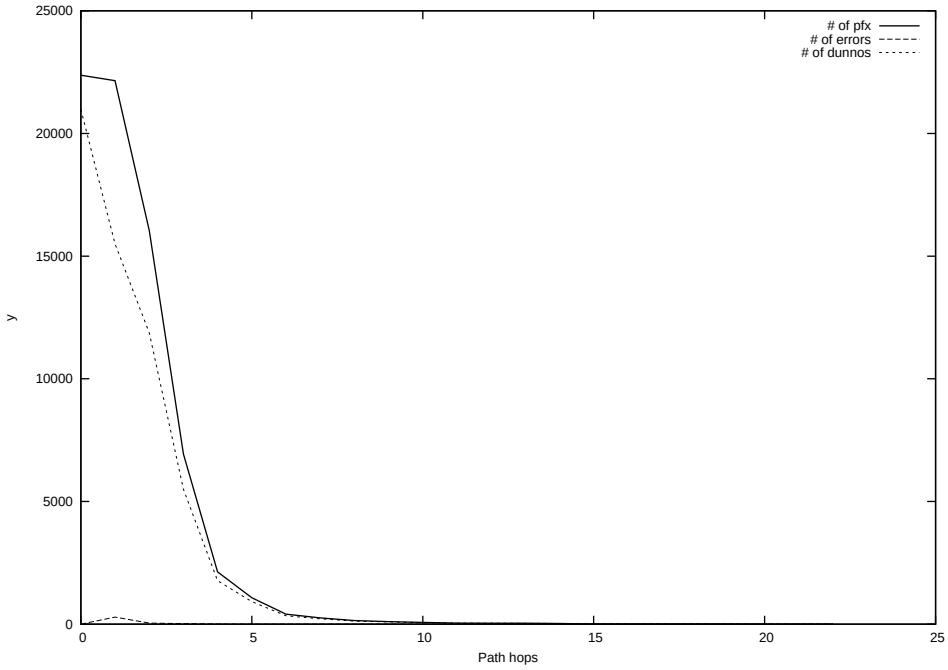
2015-05-29



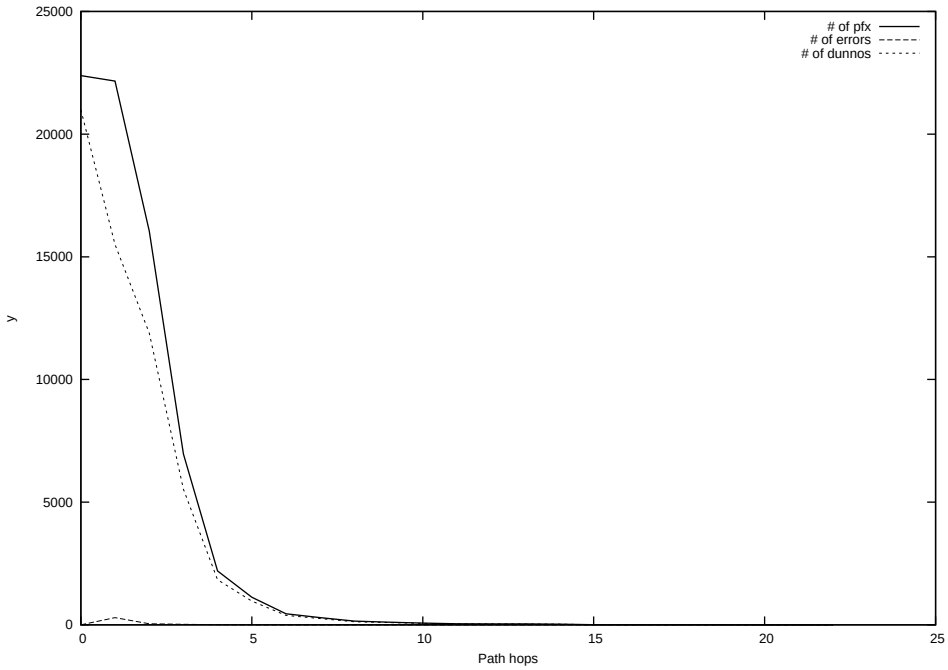
2015-05-30



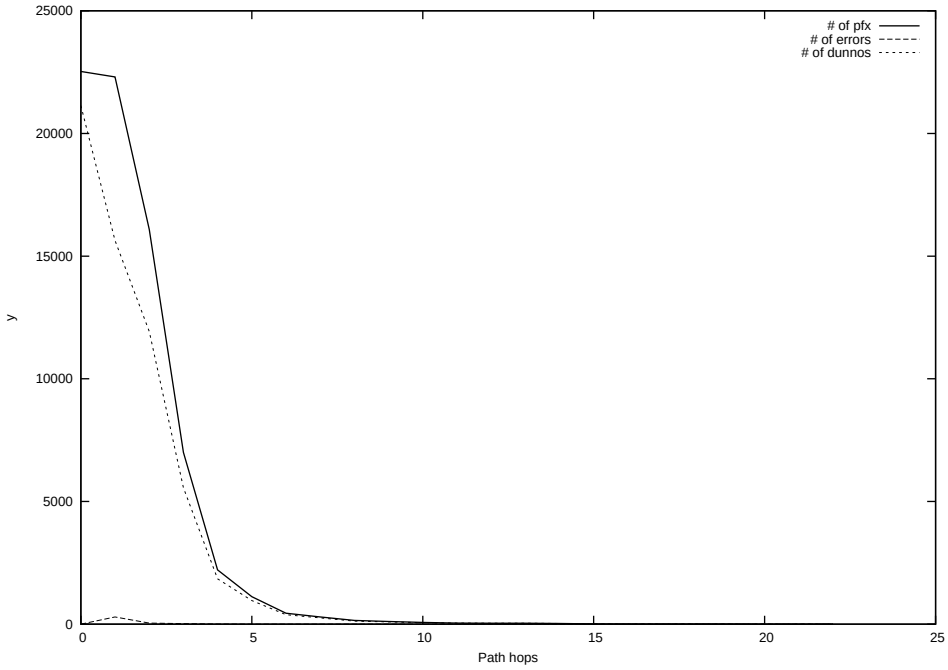
2015-05-31



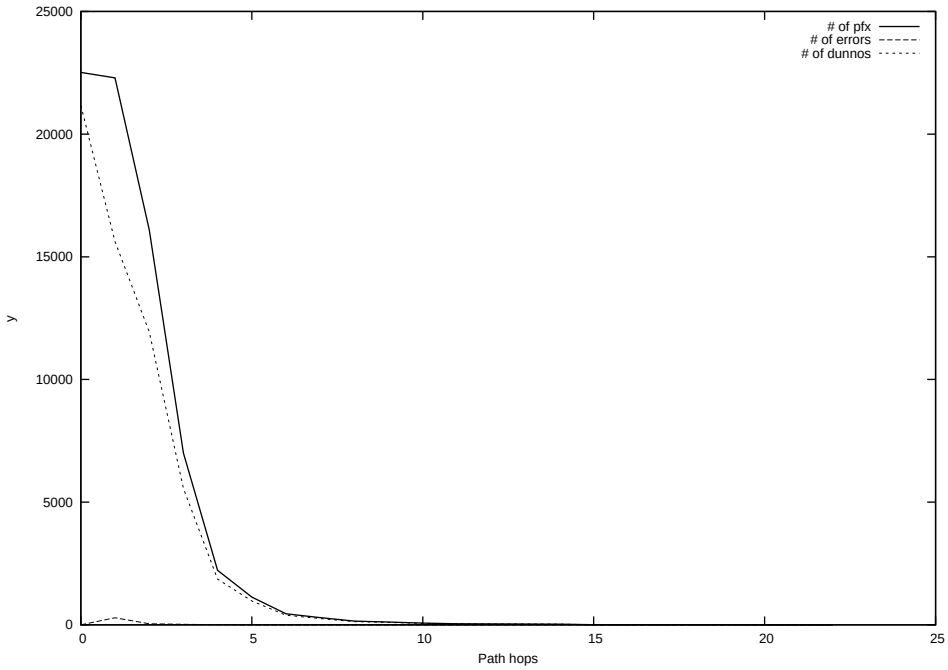
2015-06-01



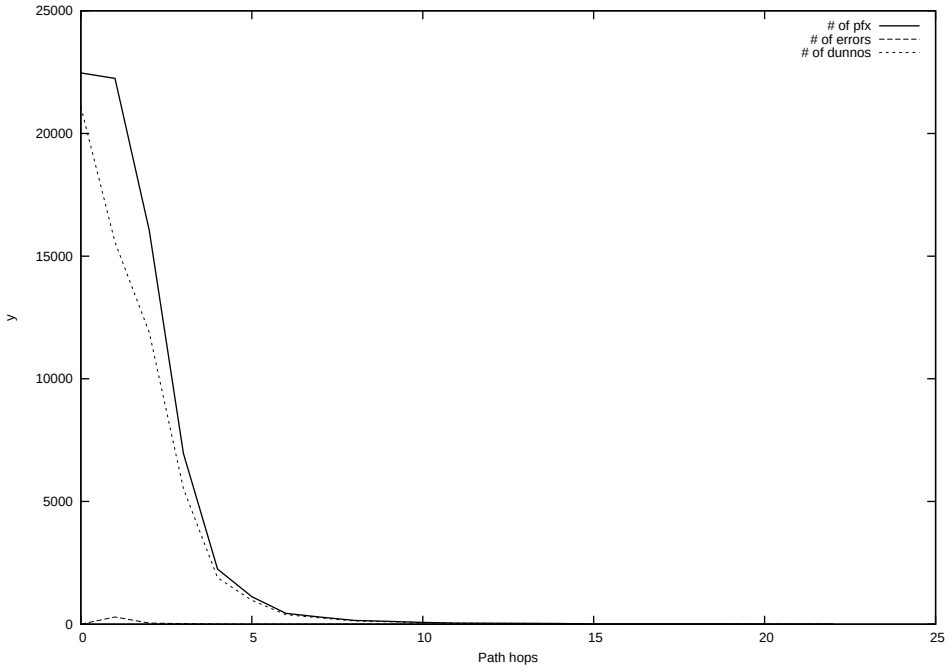
2015-06-02



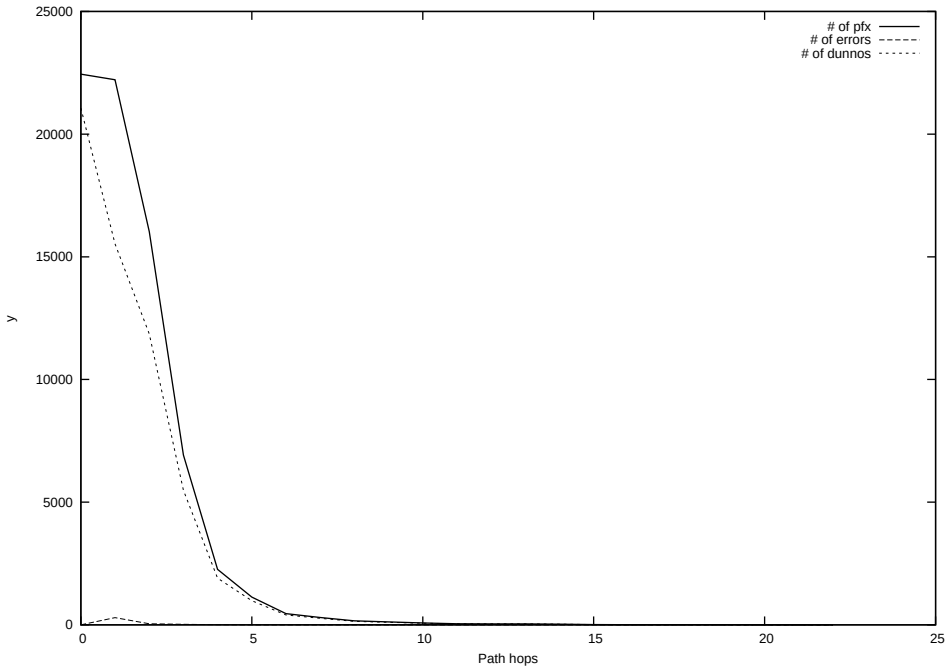
2015-06-03



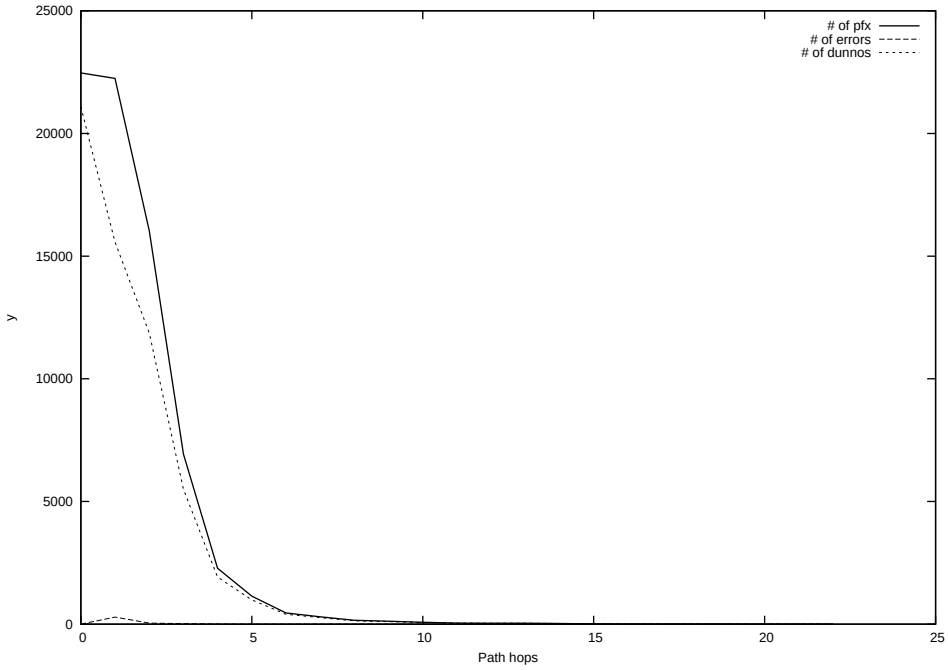
2015-06-04



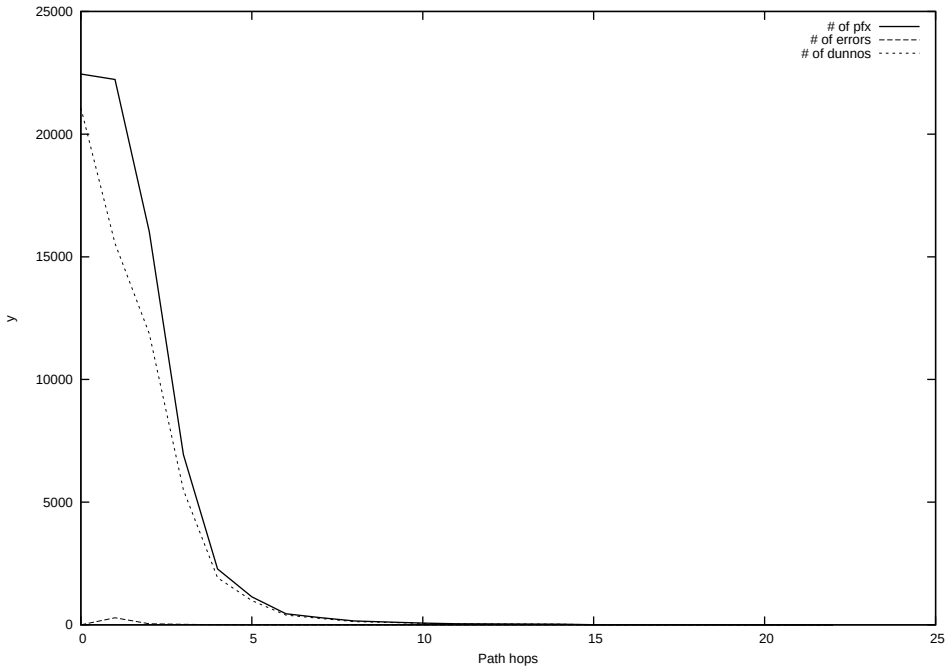
2015-06-05



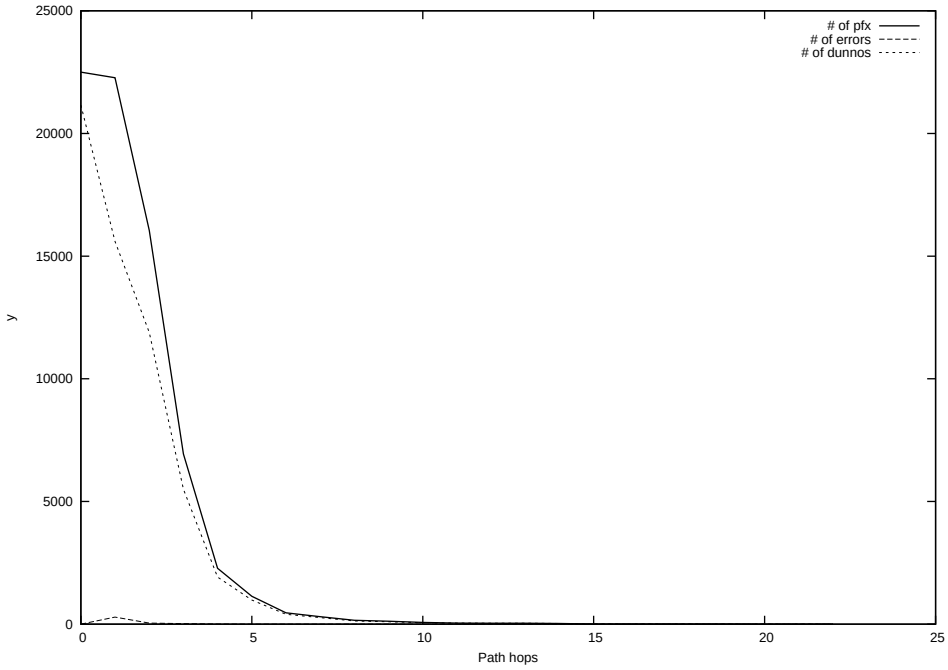
2015-06-06



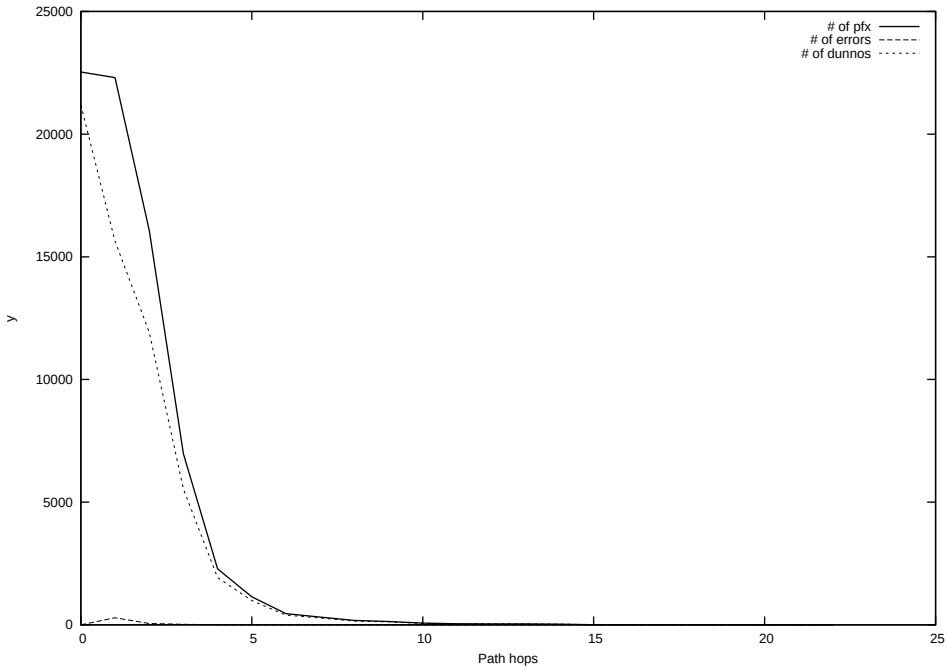
2015-06-07



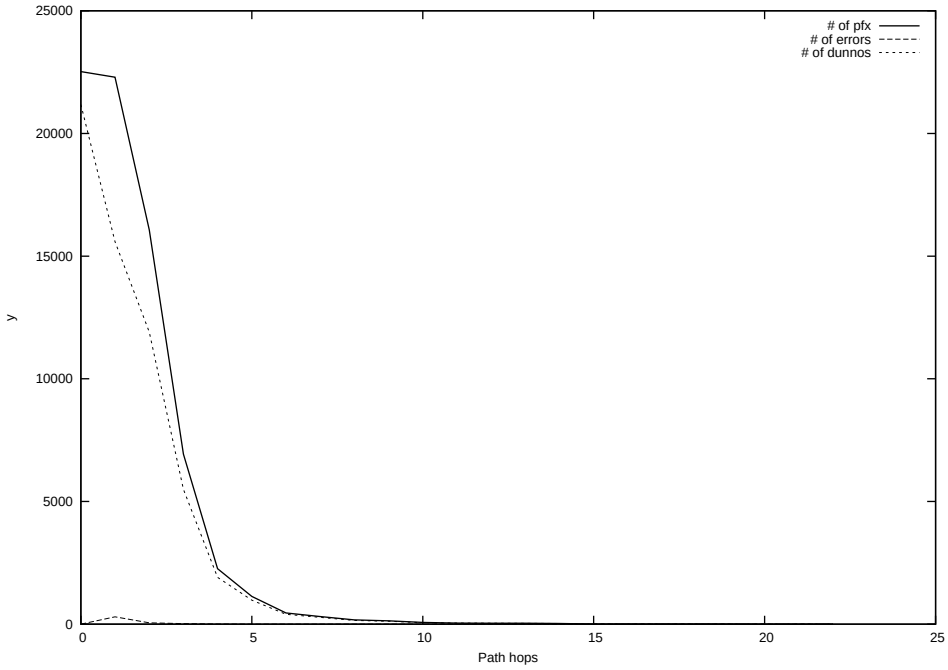
2015-06-08



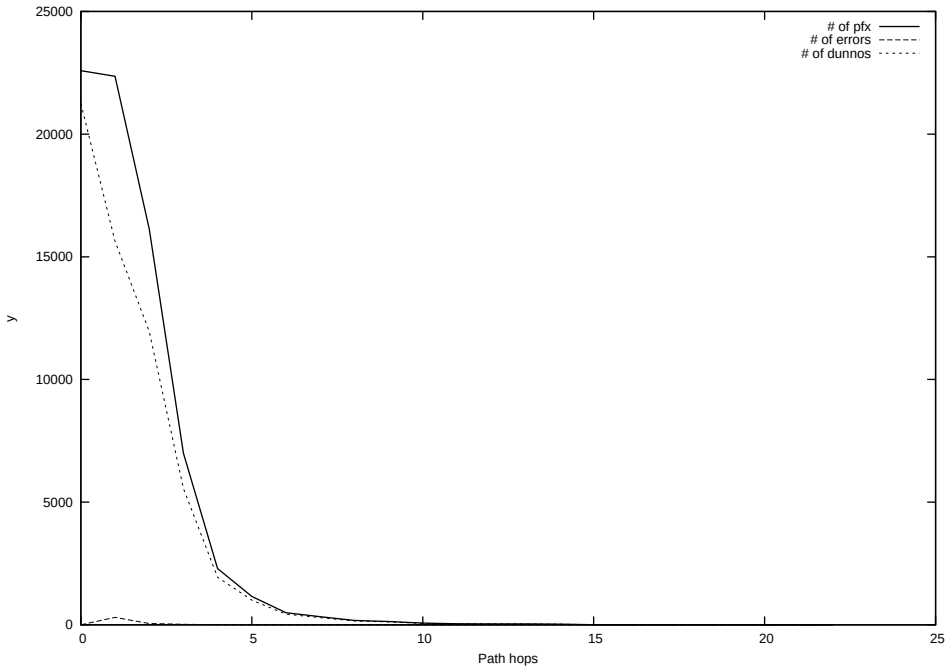
2015-06-09



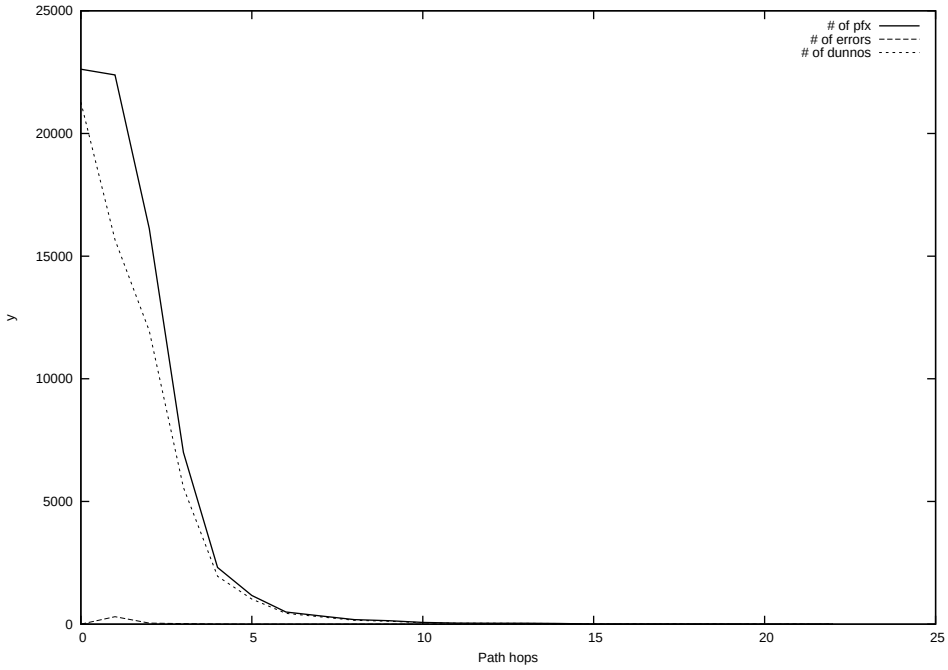
2015-06-10



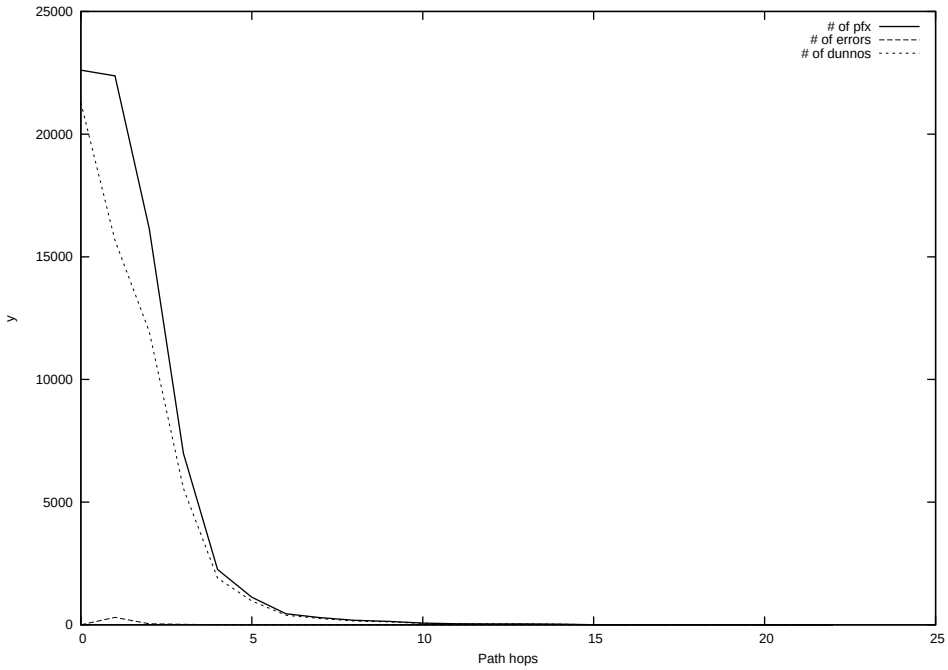
2015-06-11



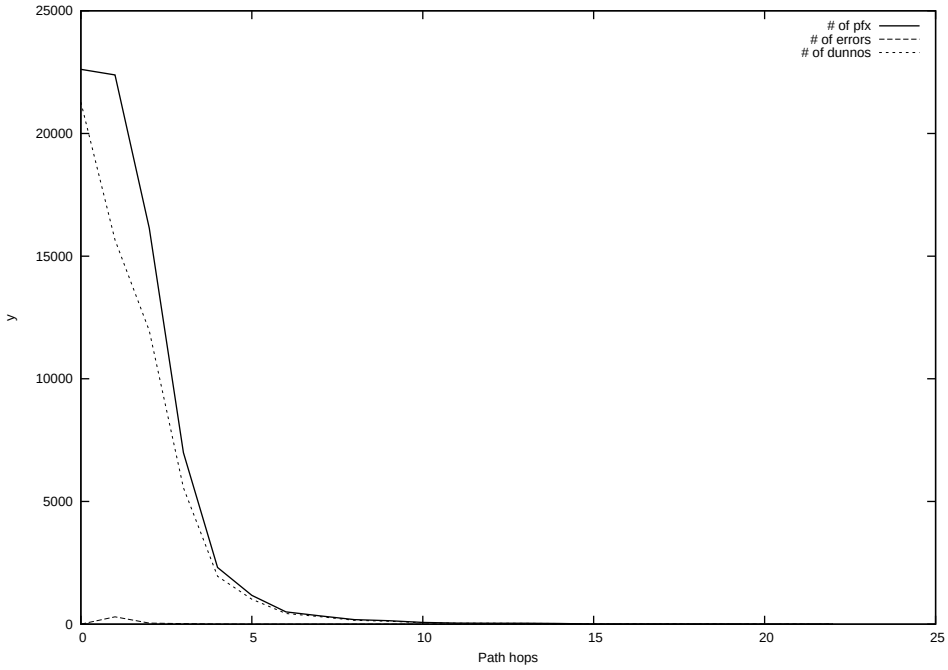
2015-06-12



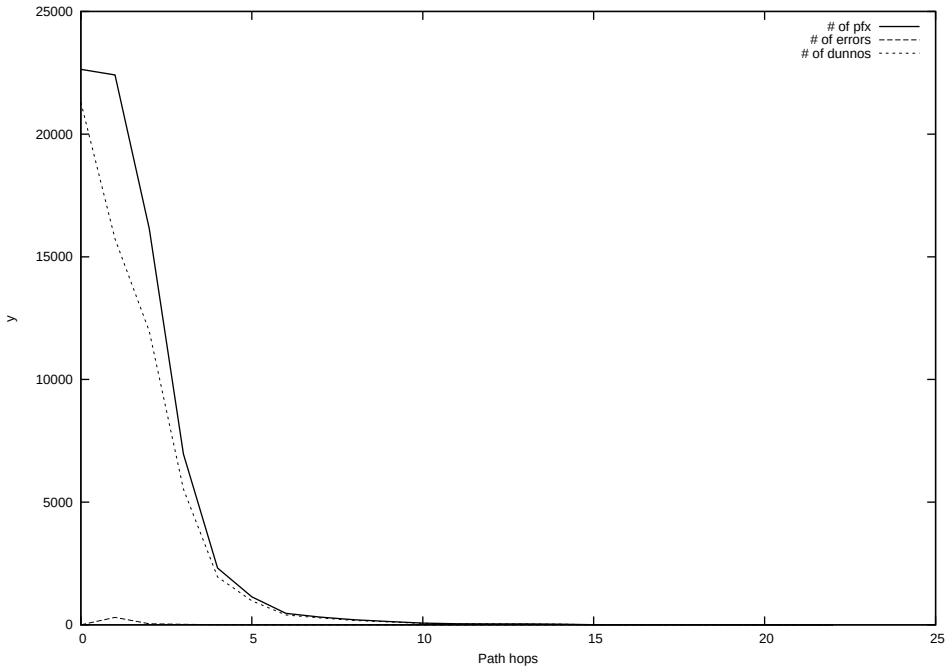
2015-06-13



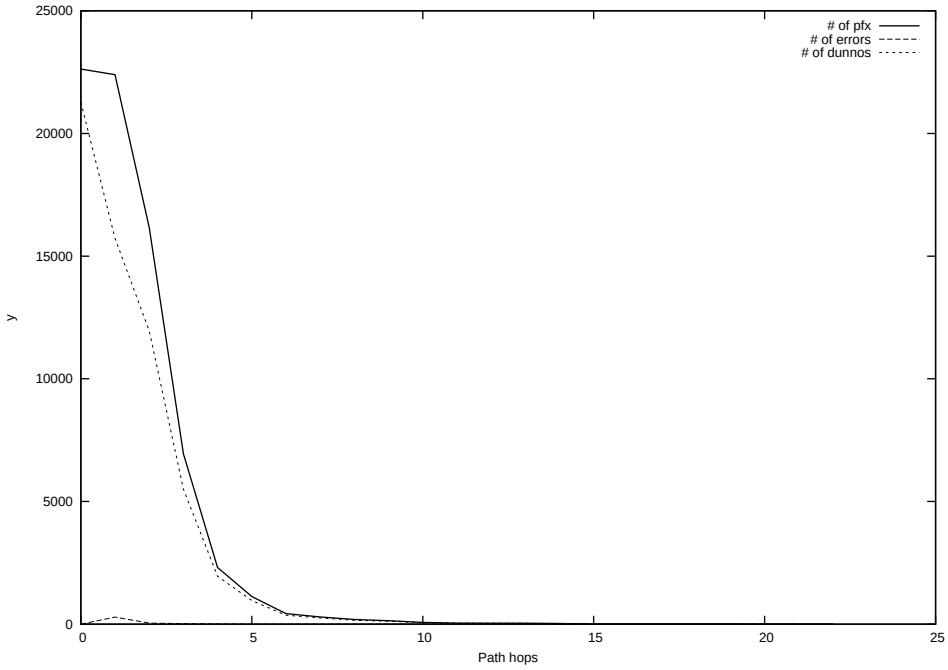
2015-06-14



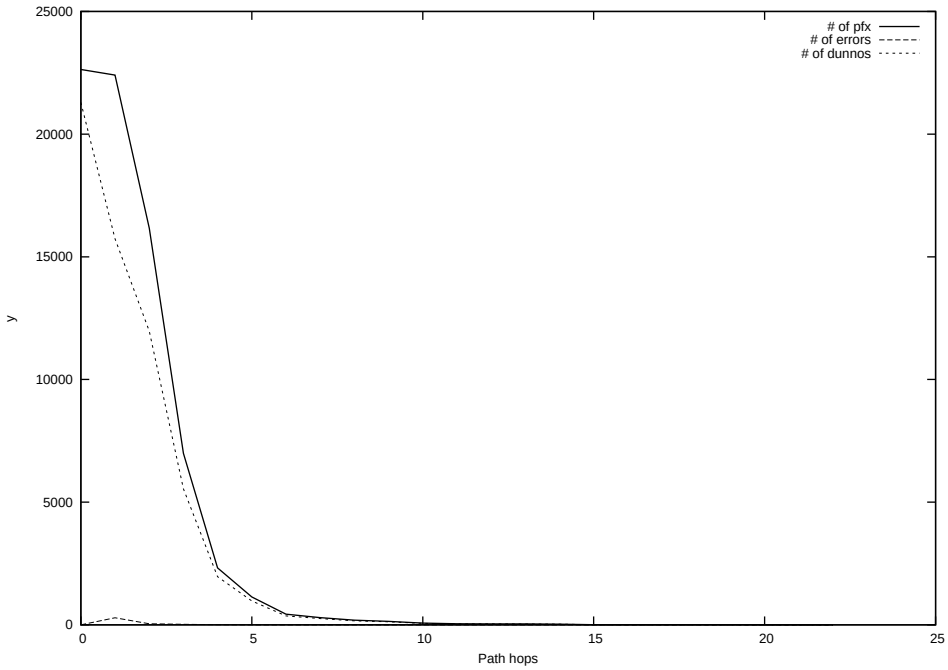
2015-06-15



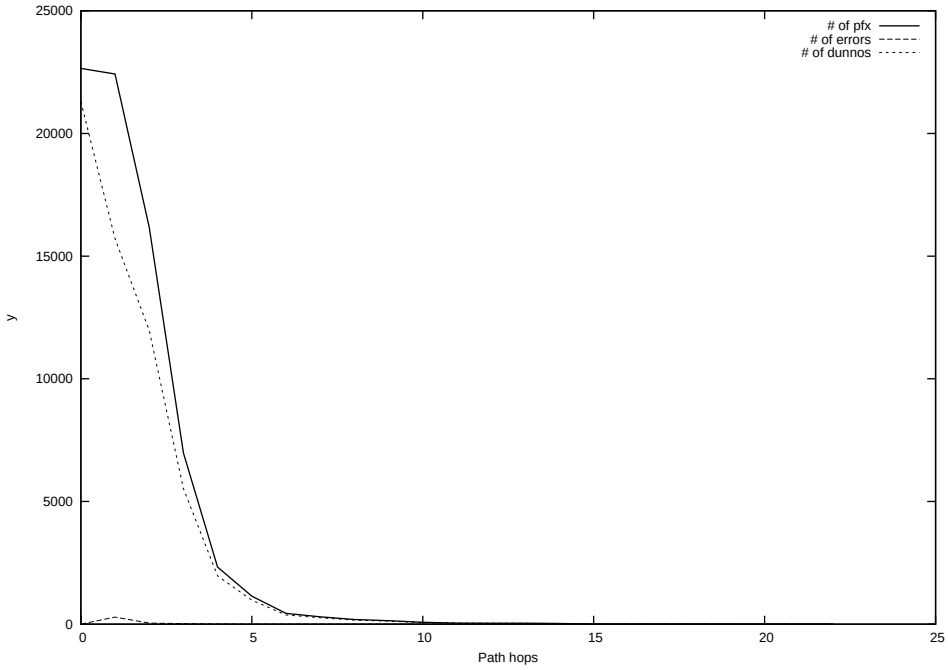
2015-06-16



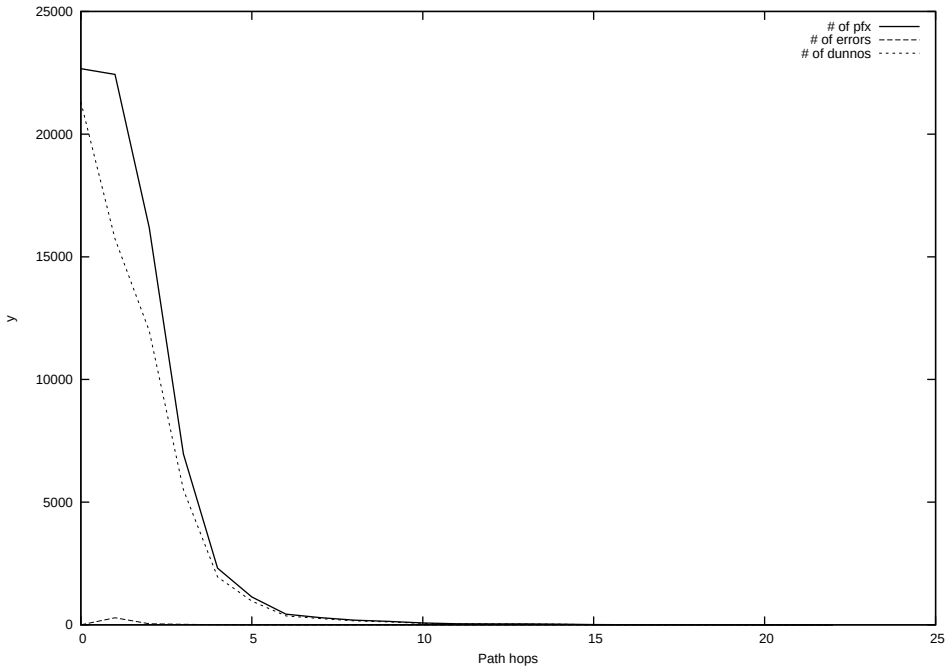
2015-06-17



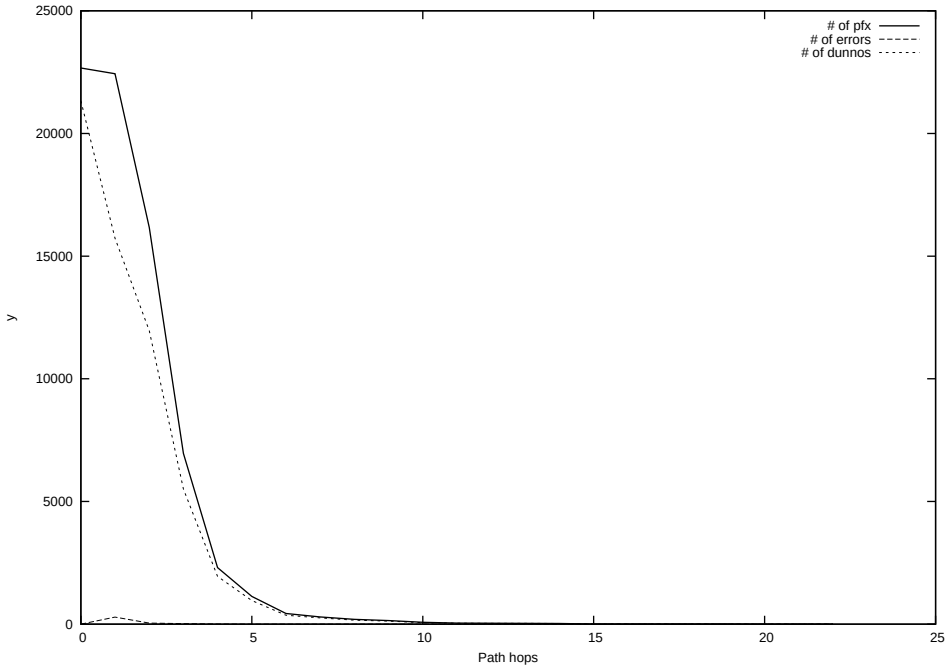
2015-06-18



2015-06-19

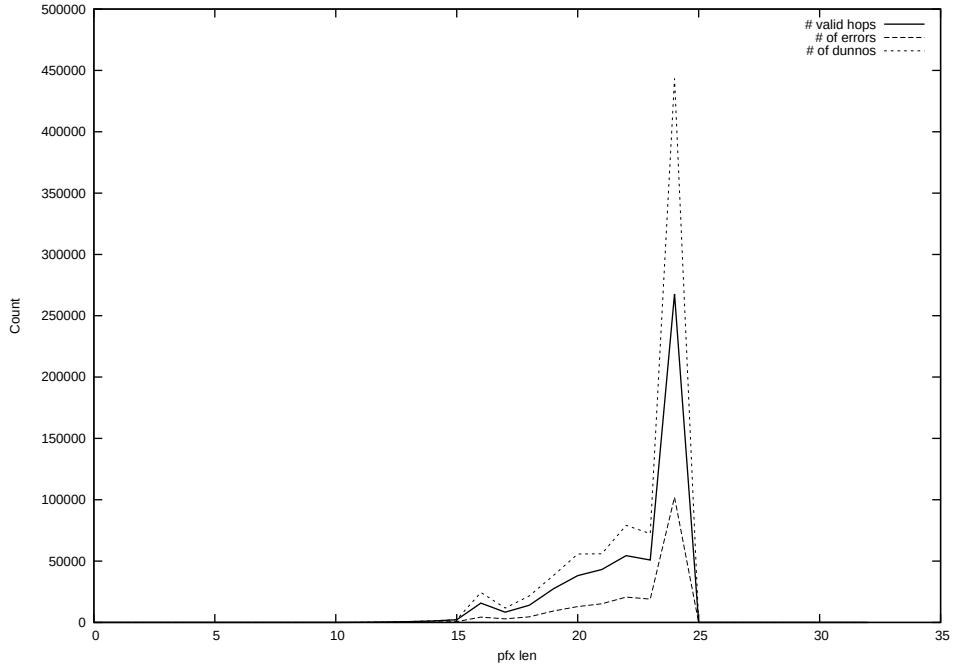


2015-06-20

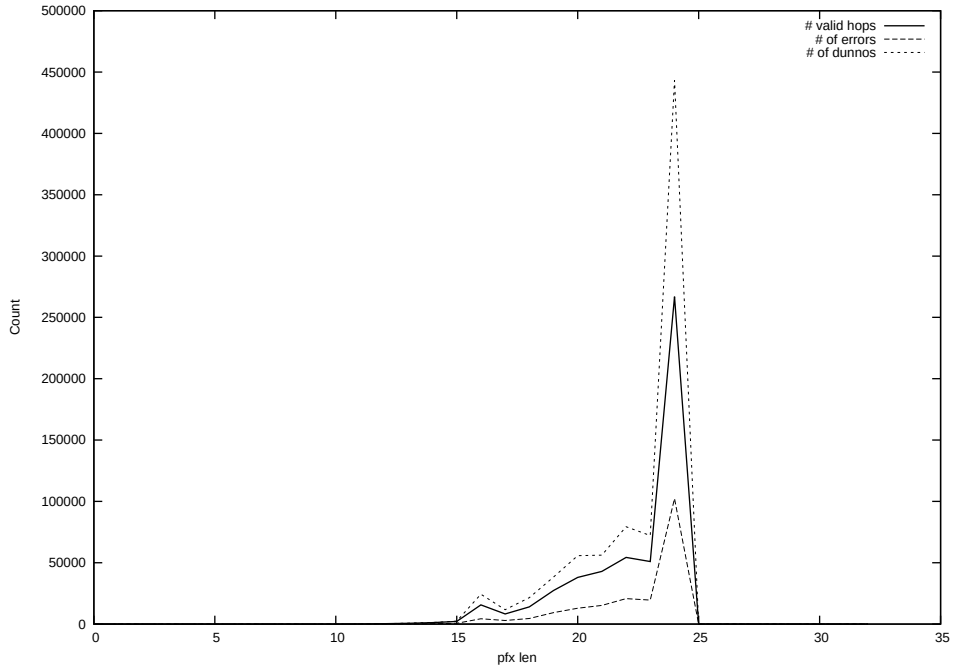


2015-06-21

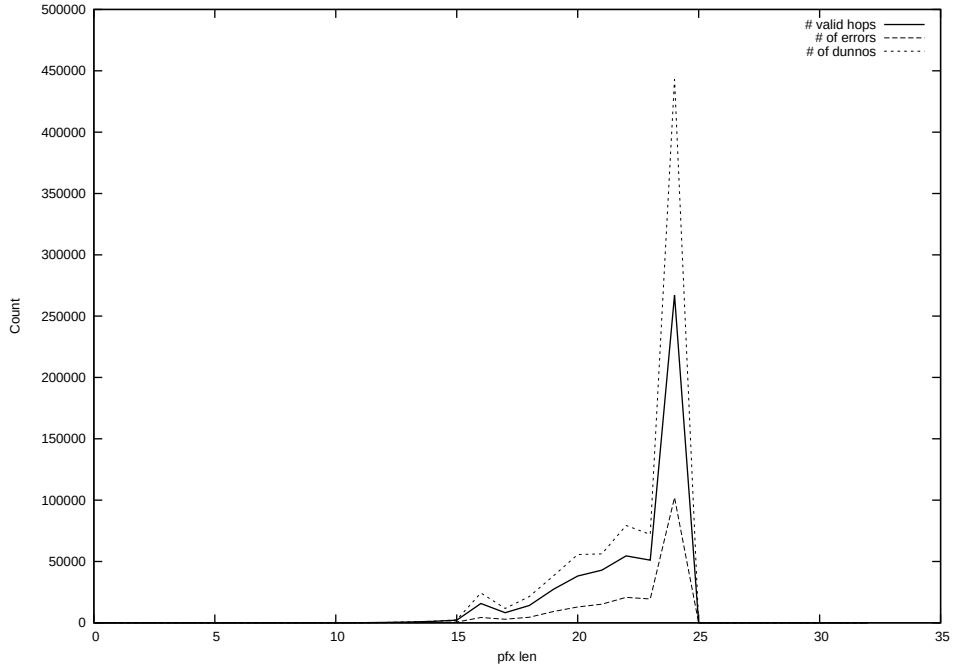
A.13 IPv4 BGP paths against RIPE DB by prefix length



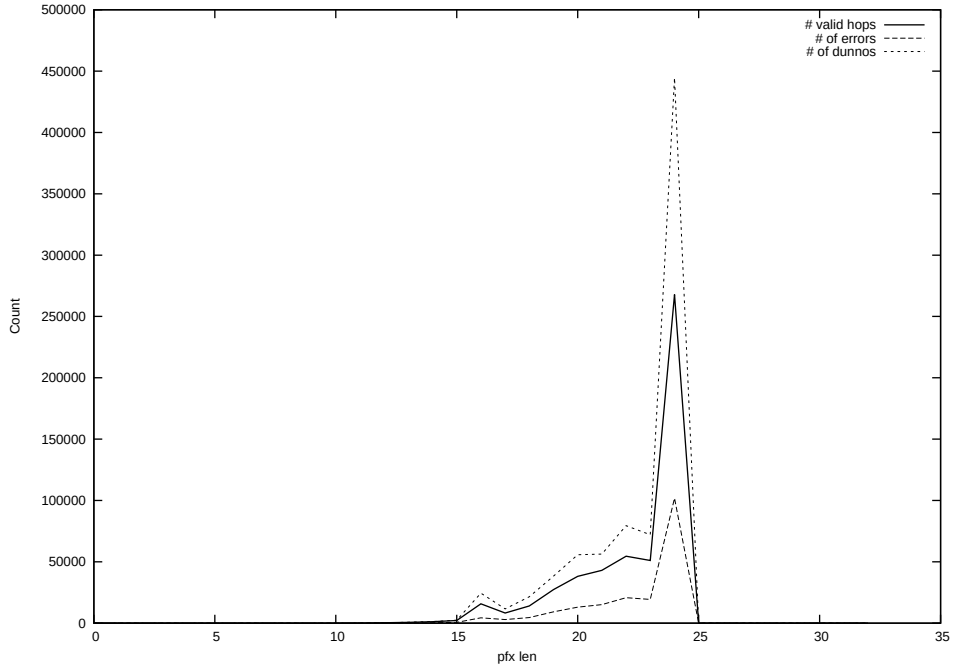
2012-03-22



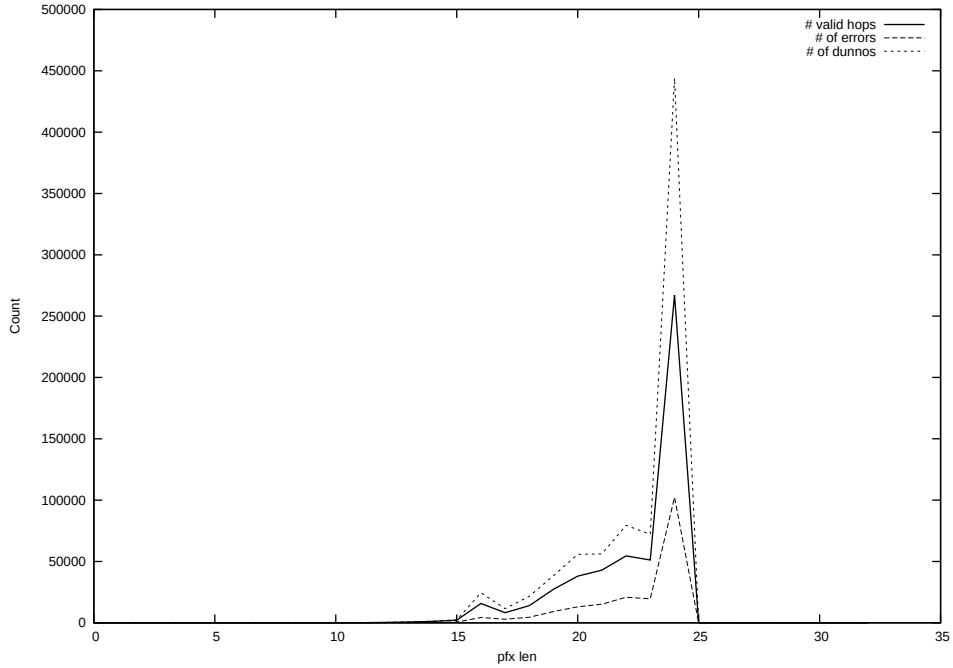
2012-03-23



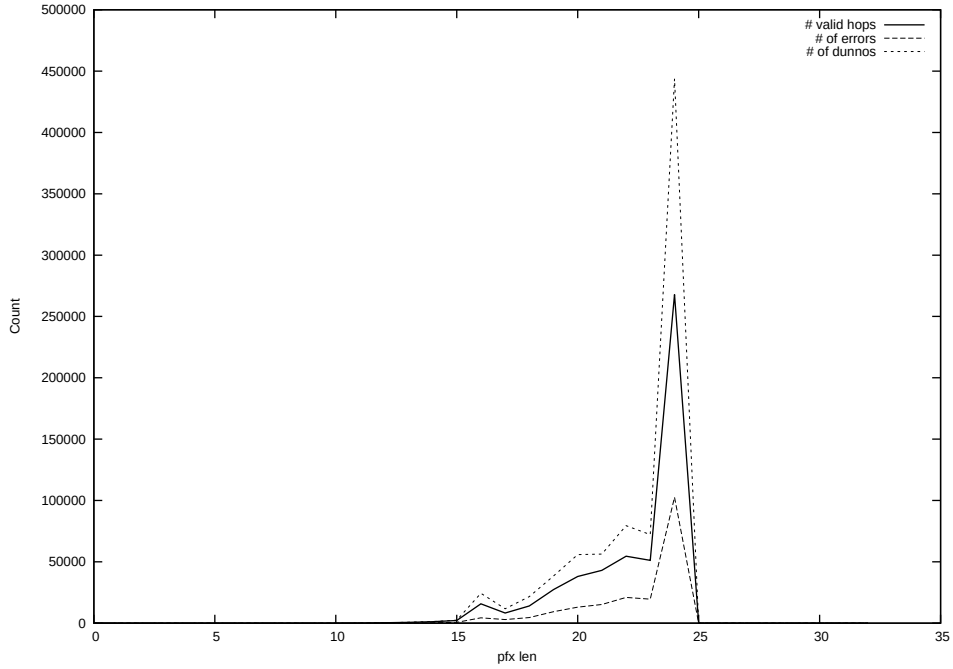
2012-03-24



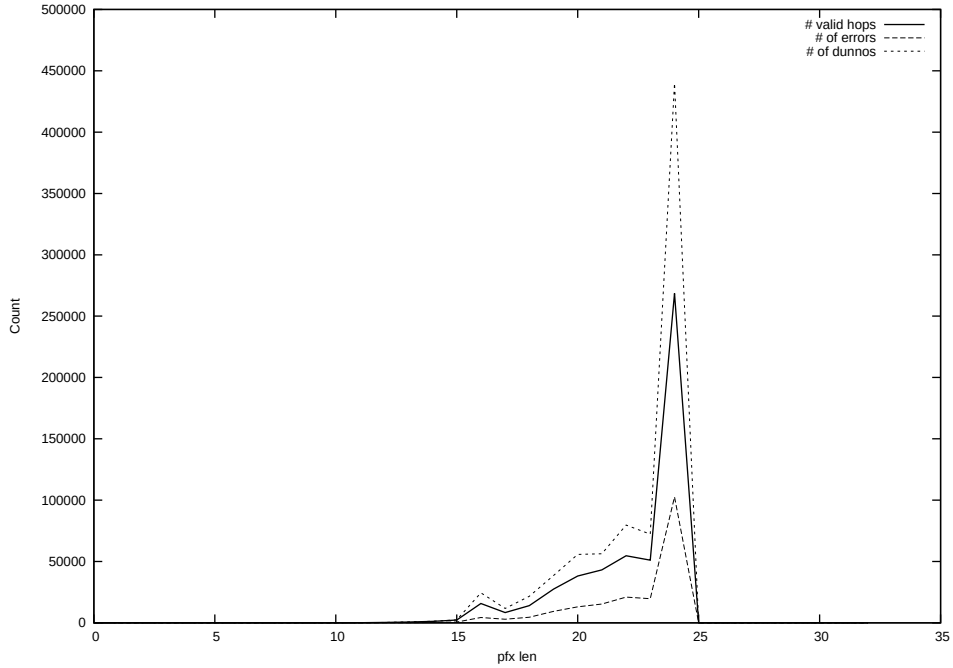
2012-03-25



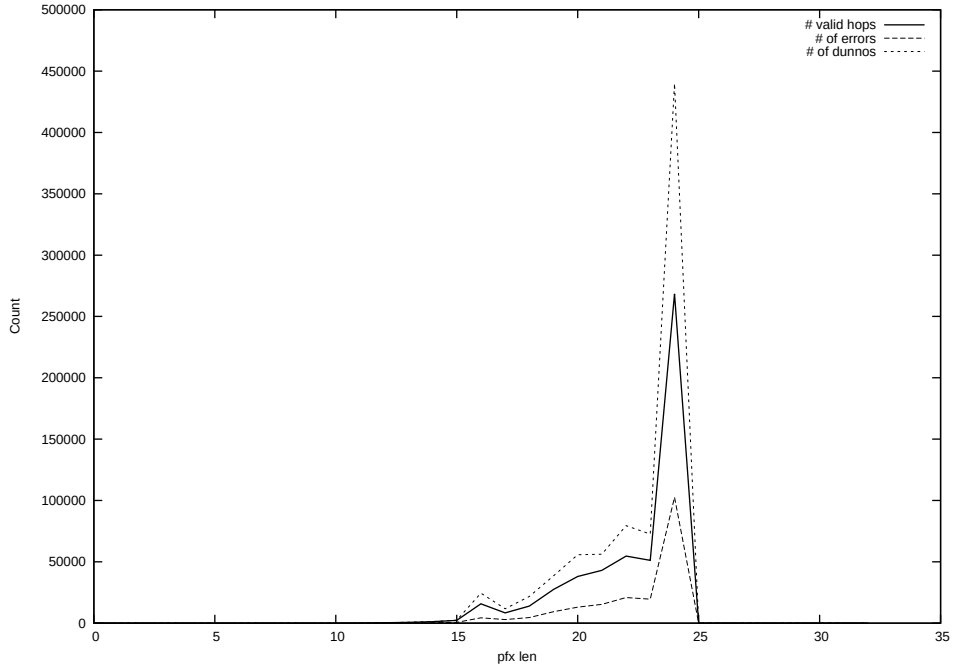
2012-03-26



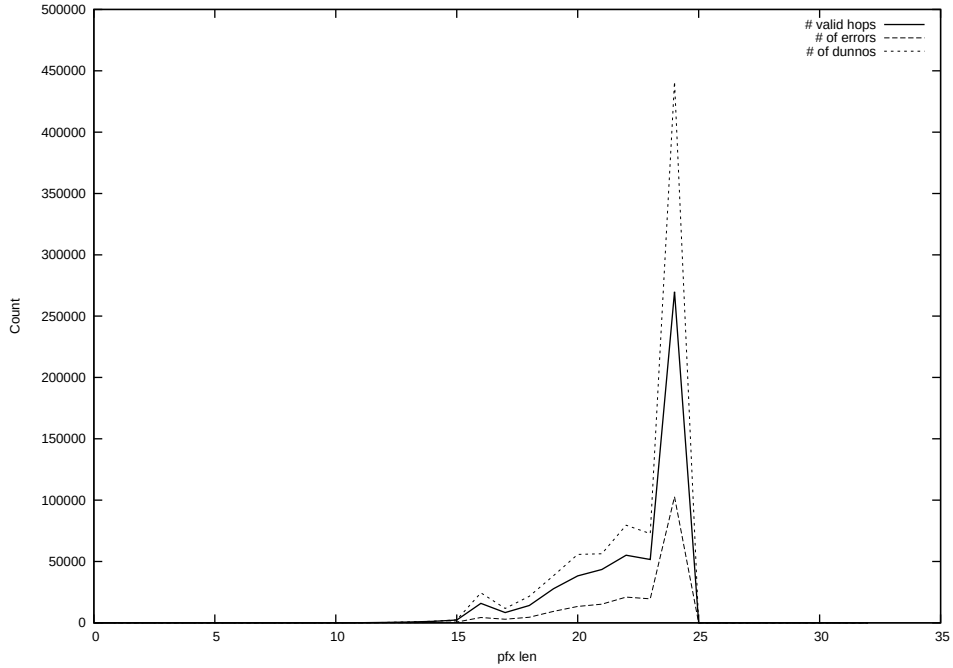
2012-03-27



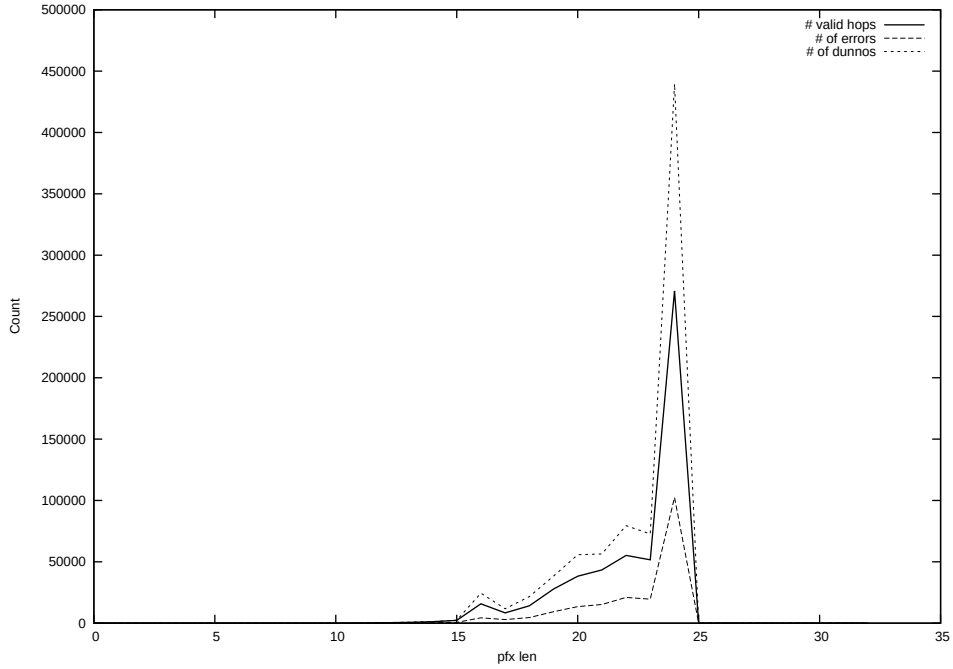
2012-03-28



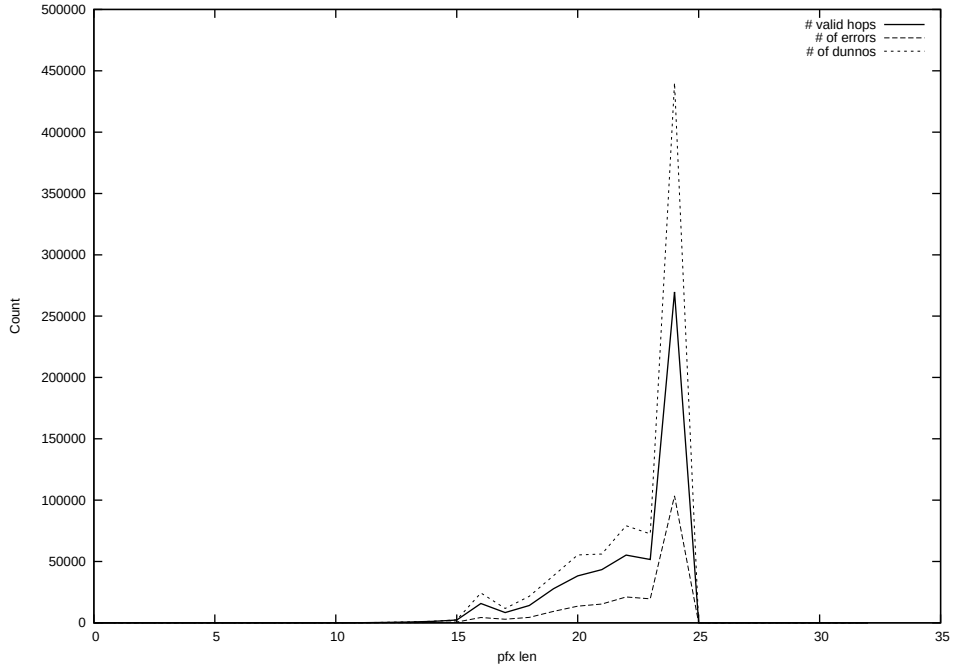
2012-03-29



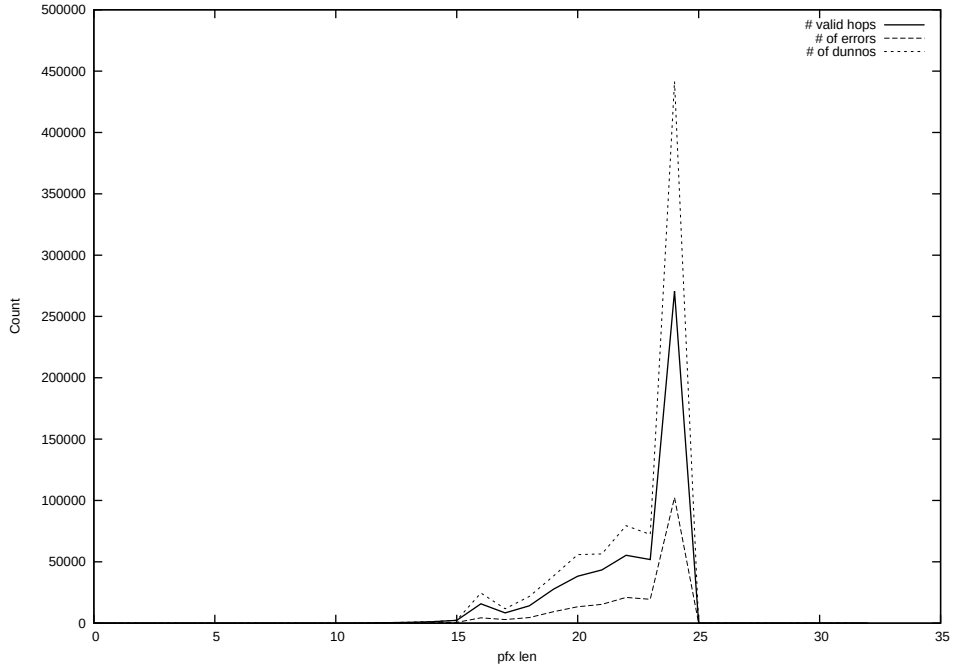
2012-03-30



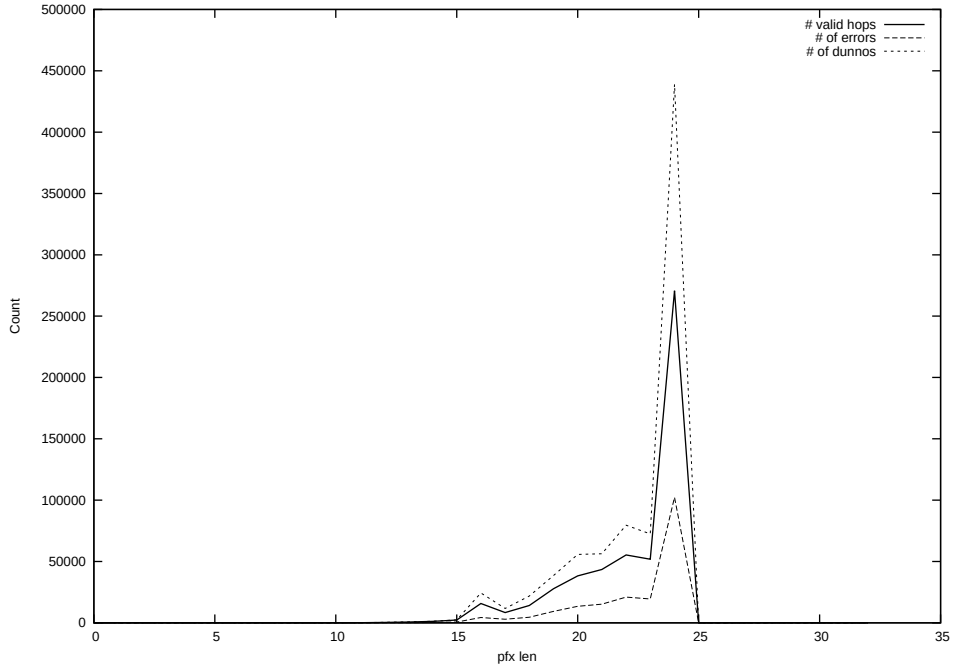
2012-03-31



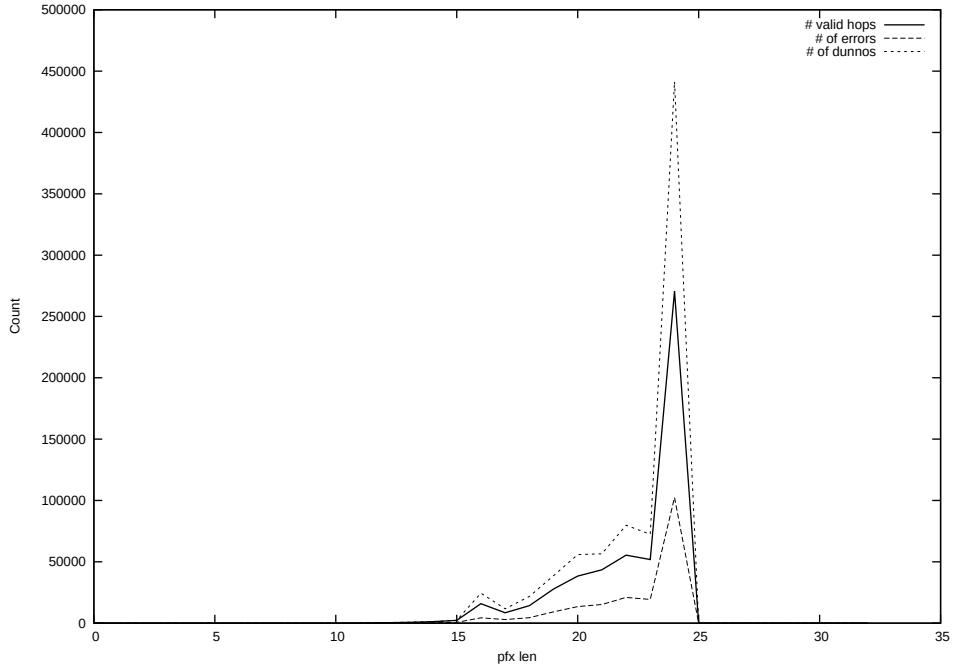
2012-04-01



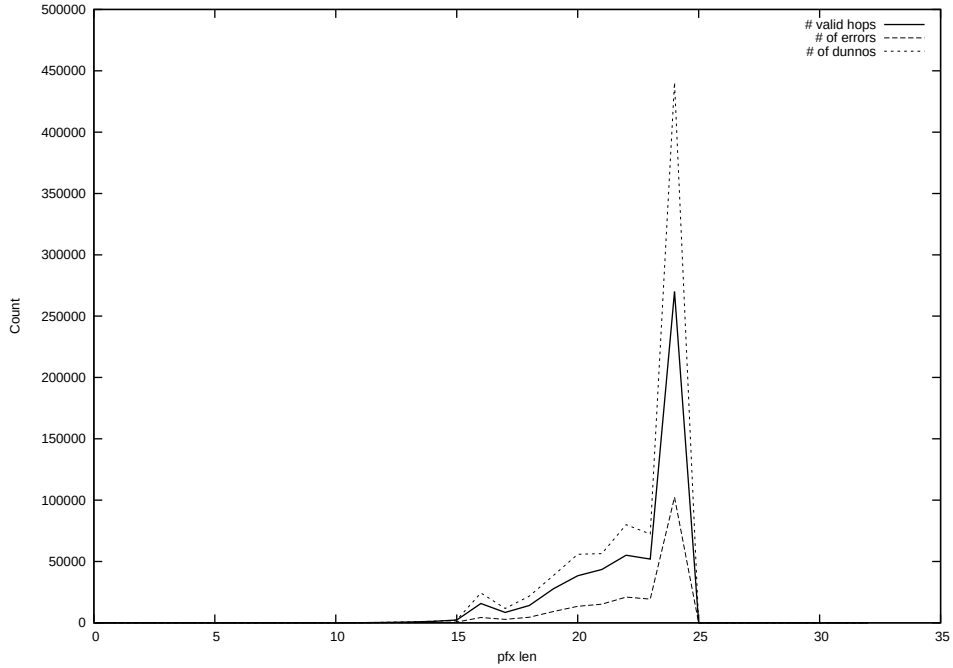
2012-04-02



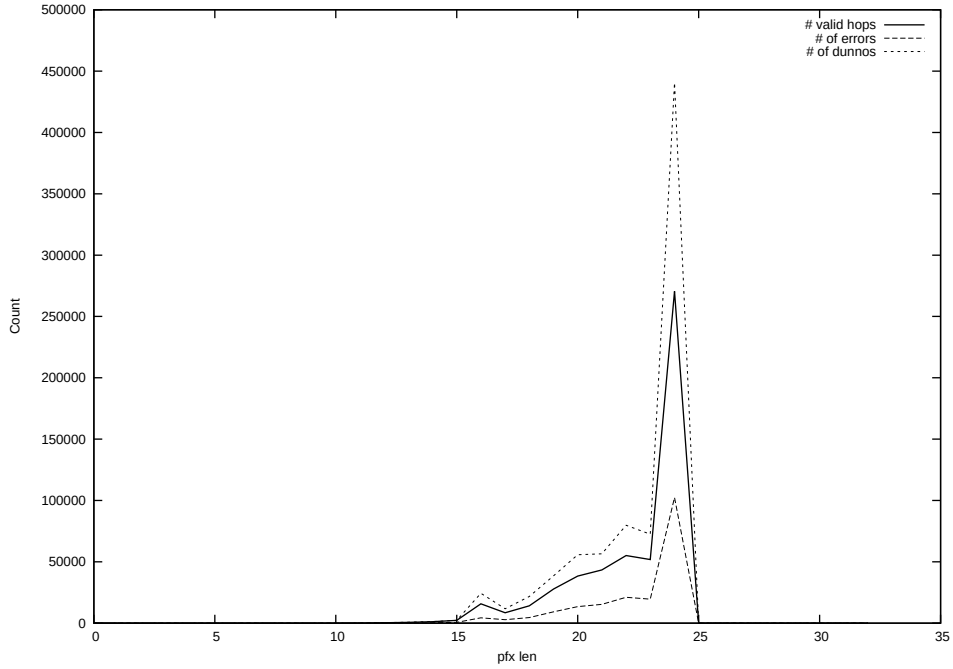
2012-04-03



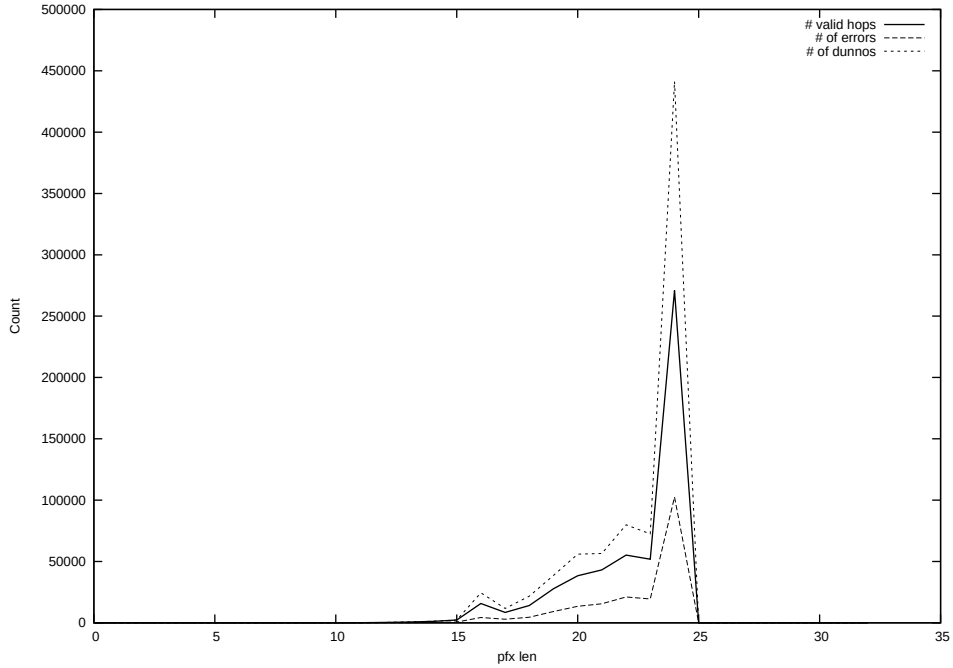
2012-04-04



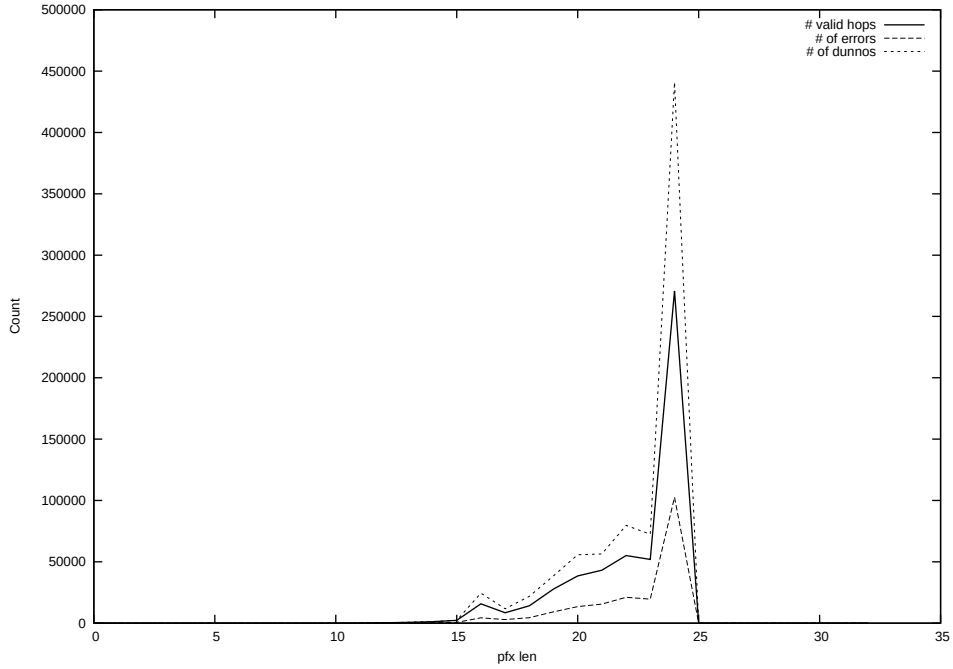
2012-04-05



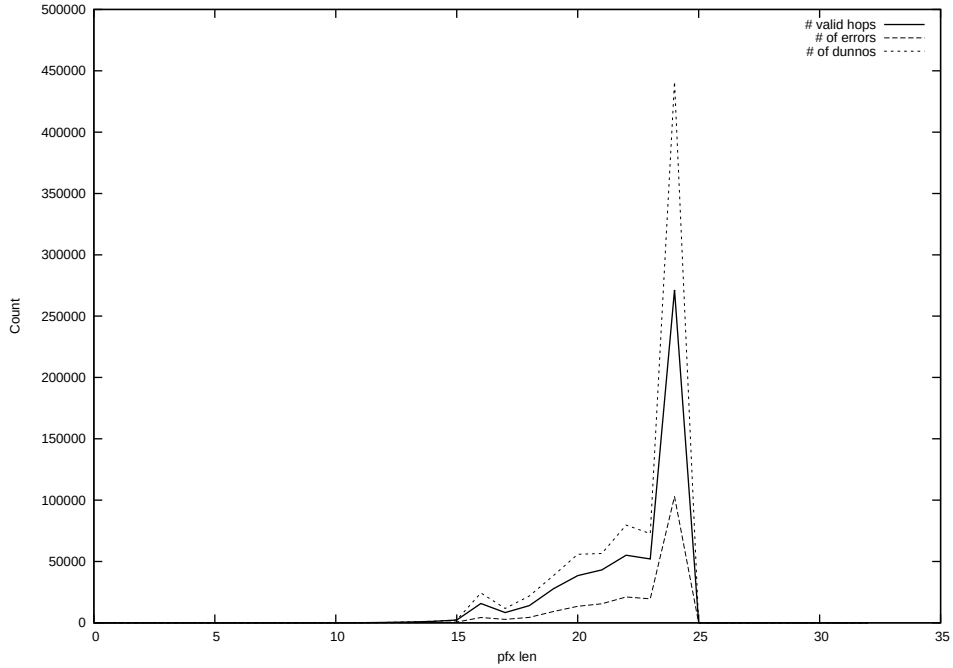
2012-04-06



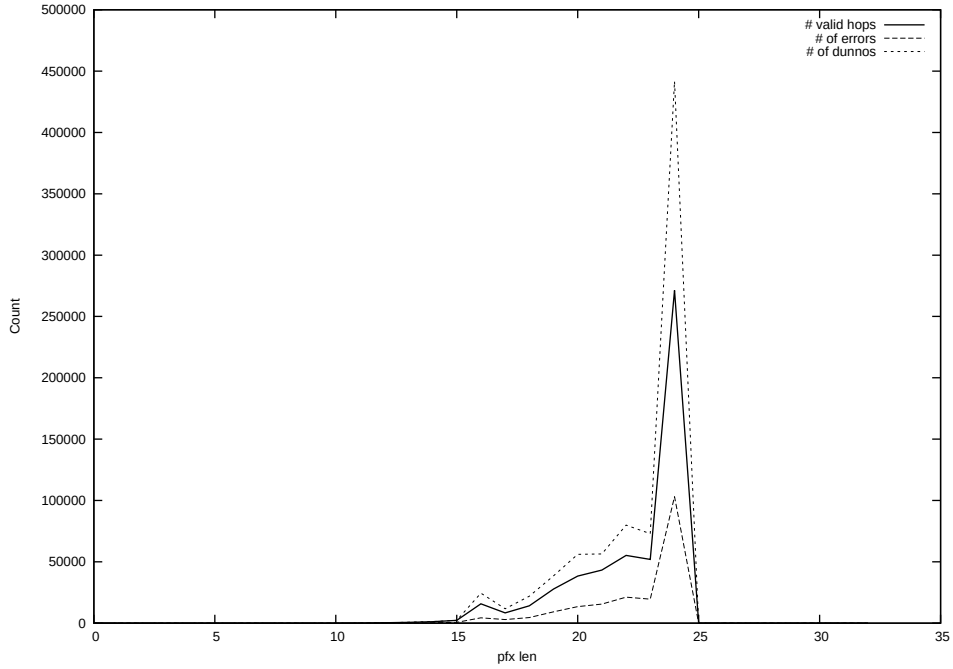
2012-04-07



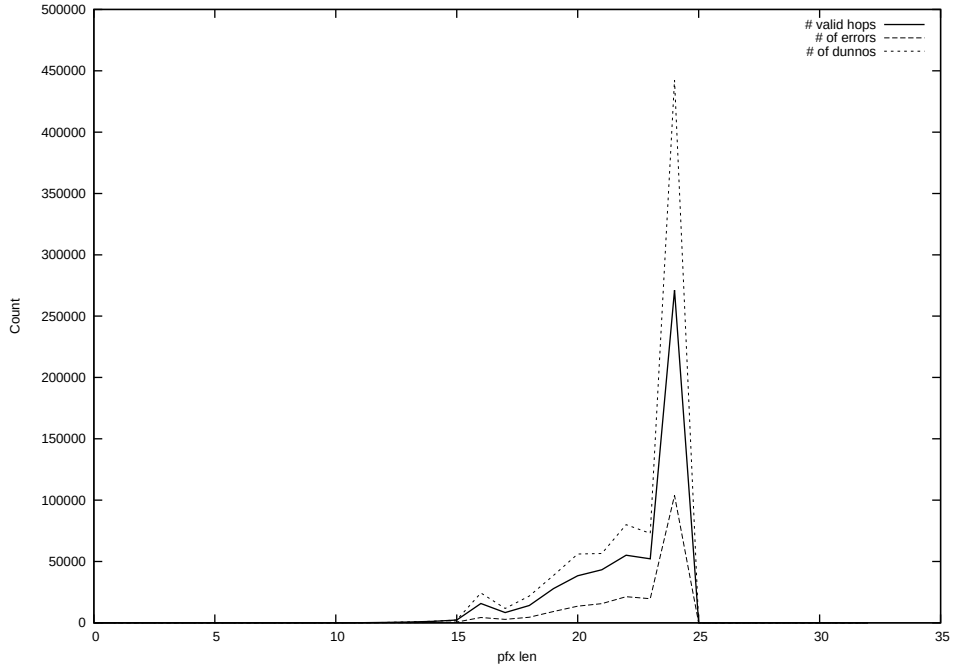
2012-04-08



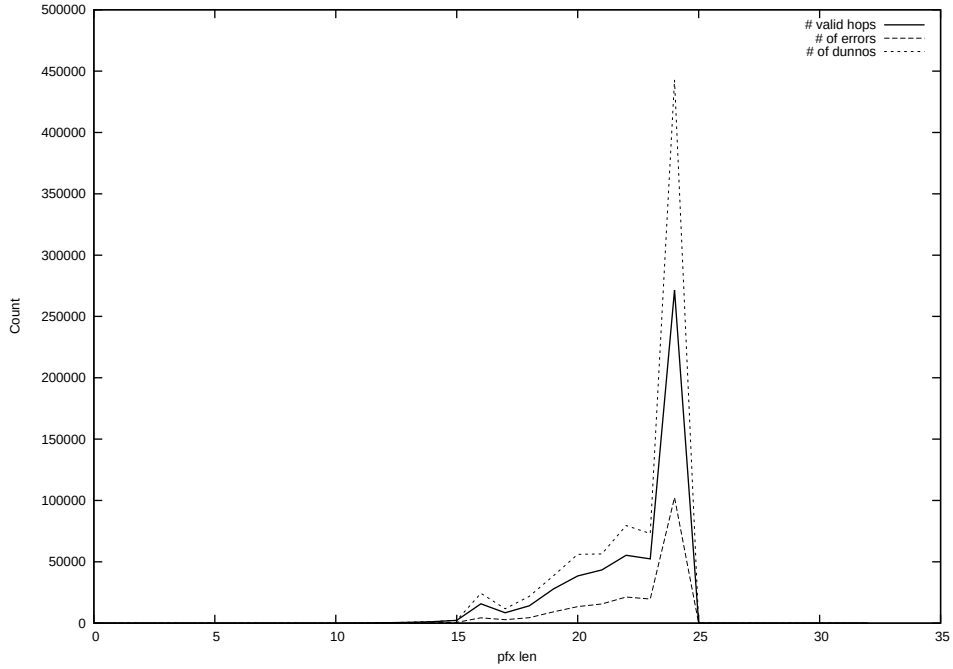
2012-04-09



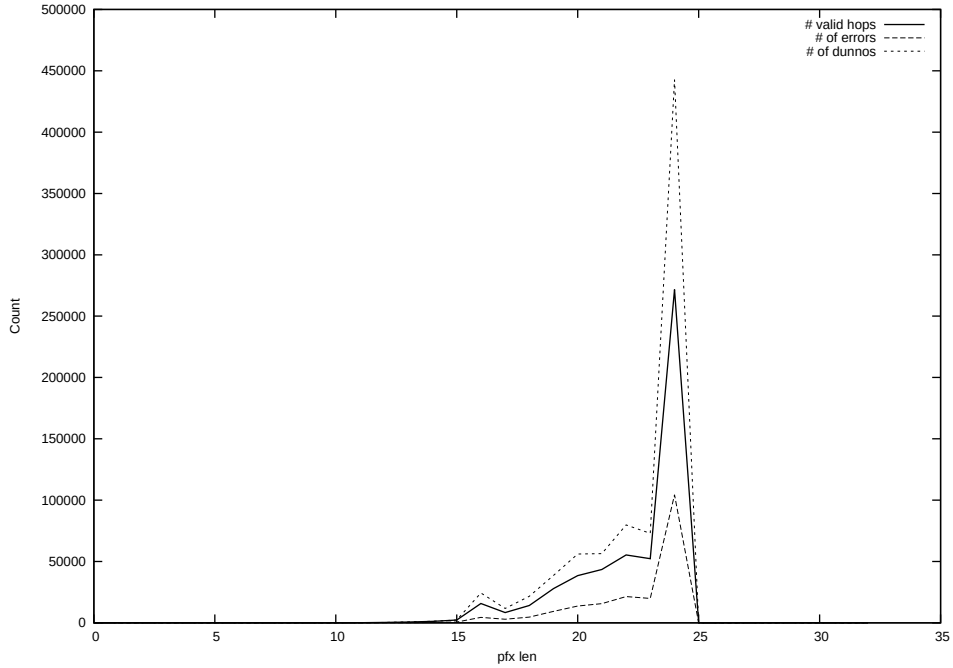
2012-04-10



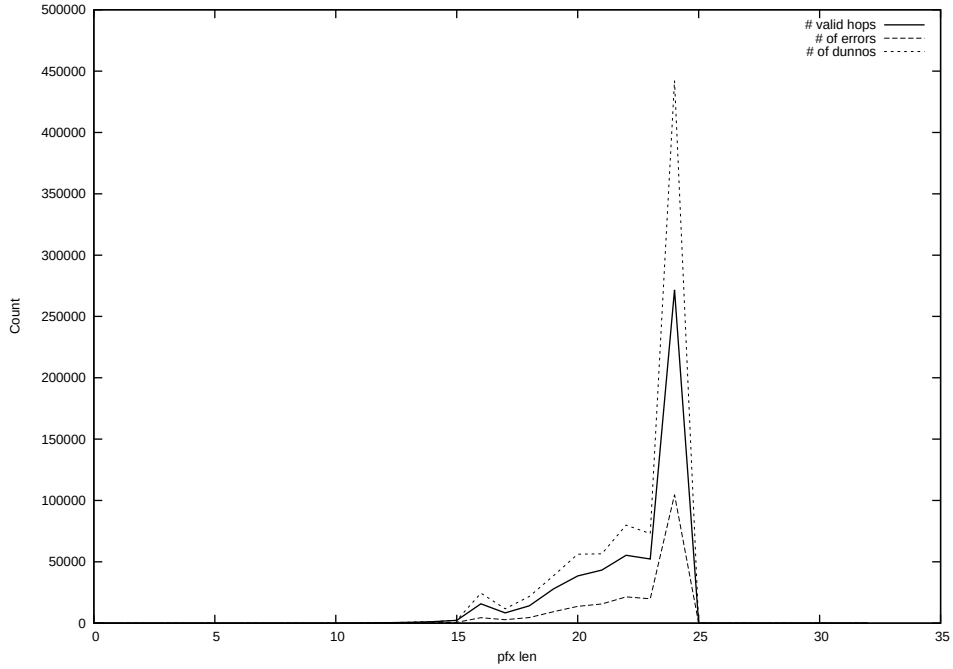
2012-04-11



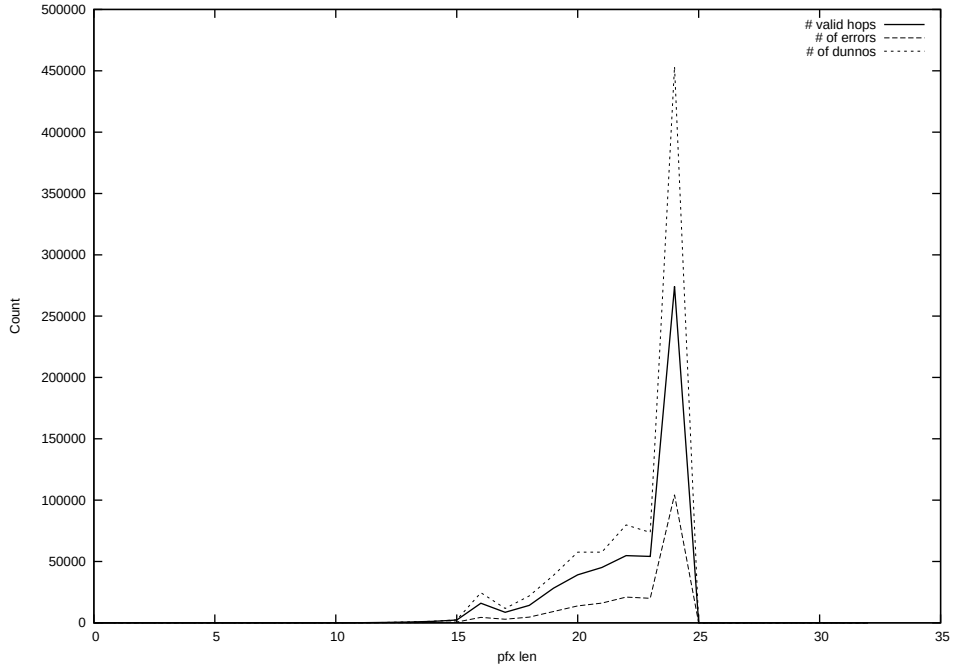
2012-04-12



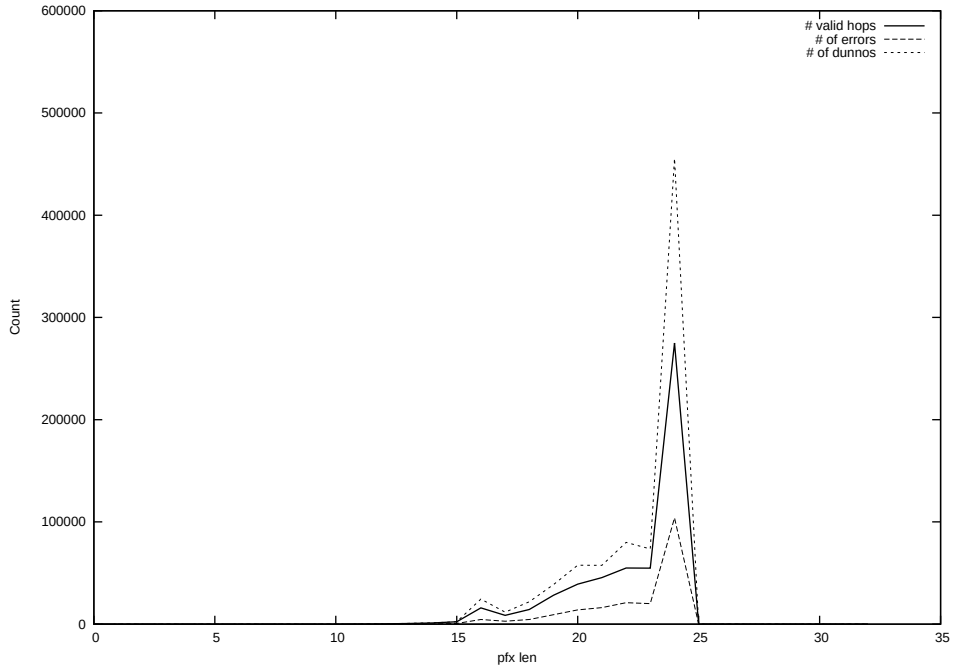
2012-04-13



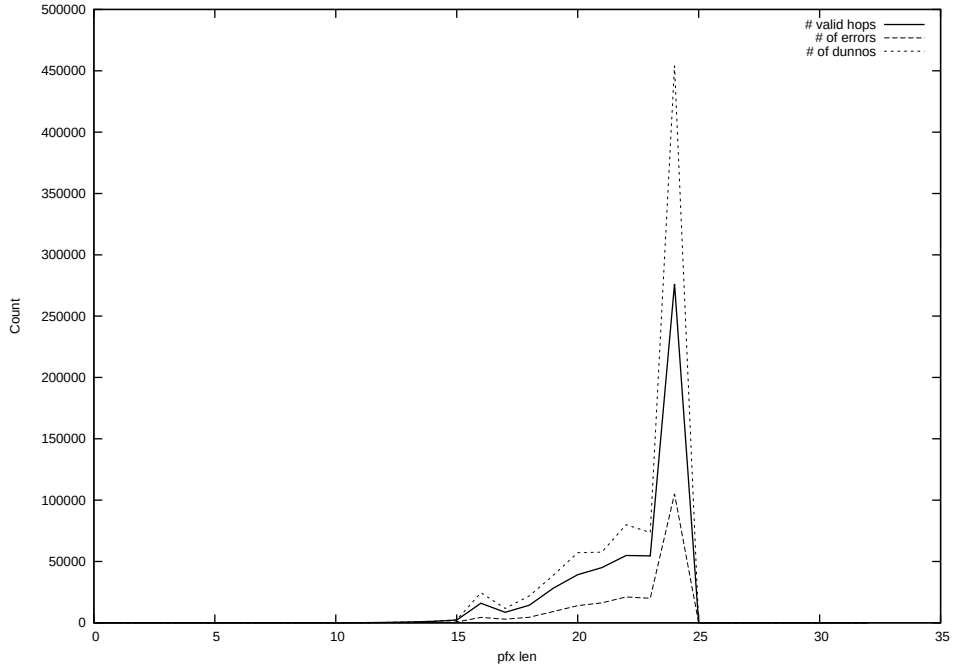
2012-04-14



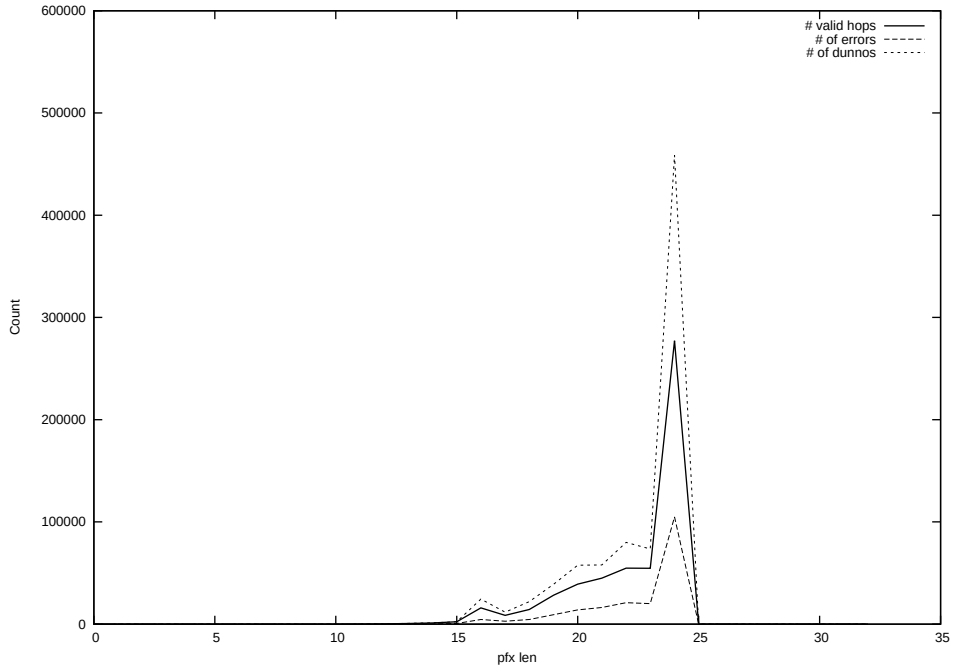
2012-05-16



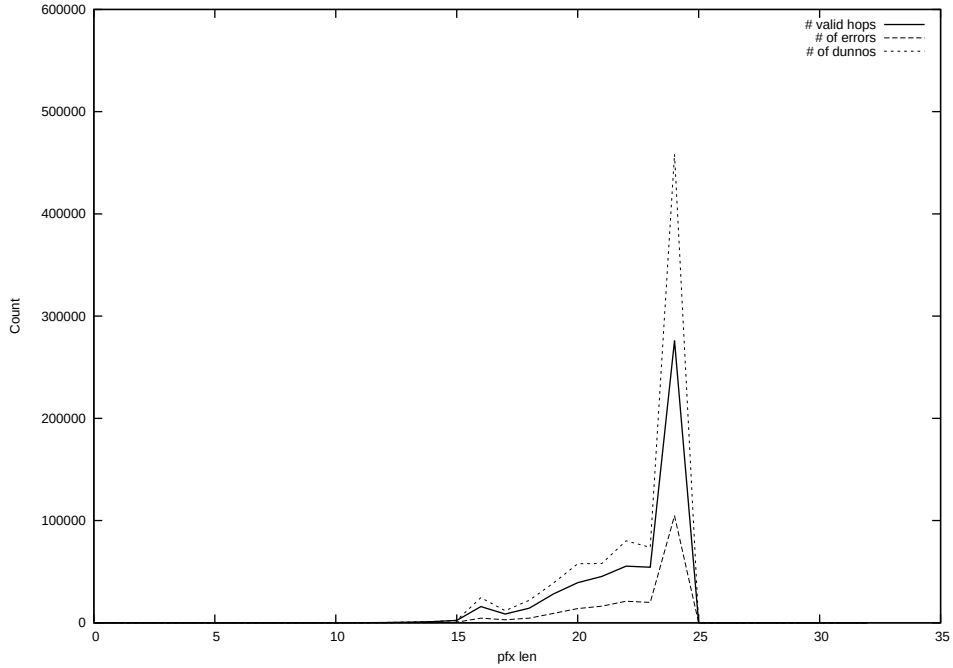
2012-05-17



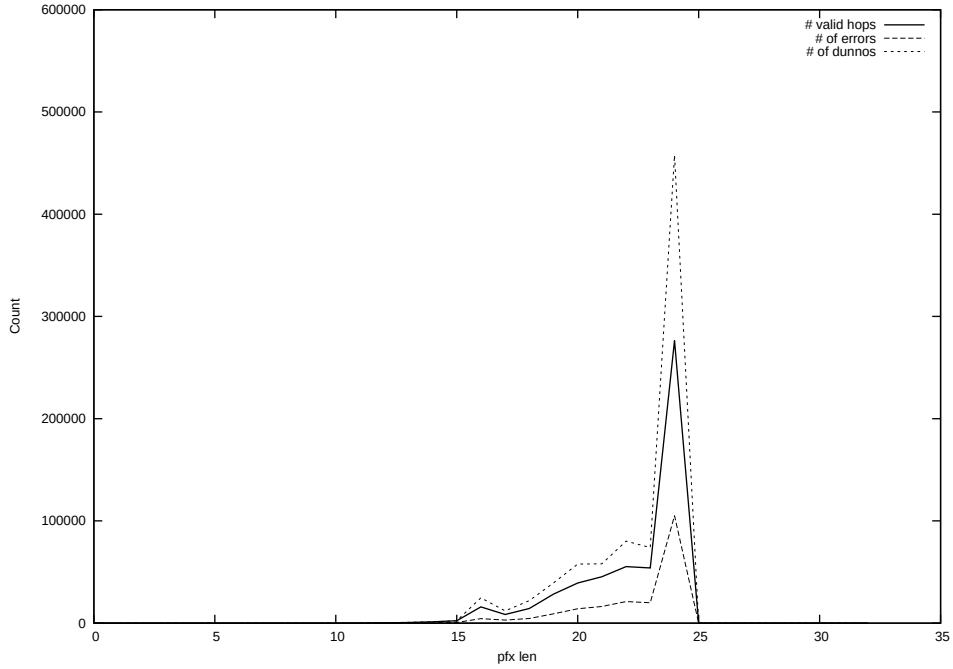
2012-05-19



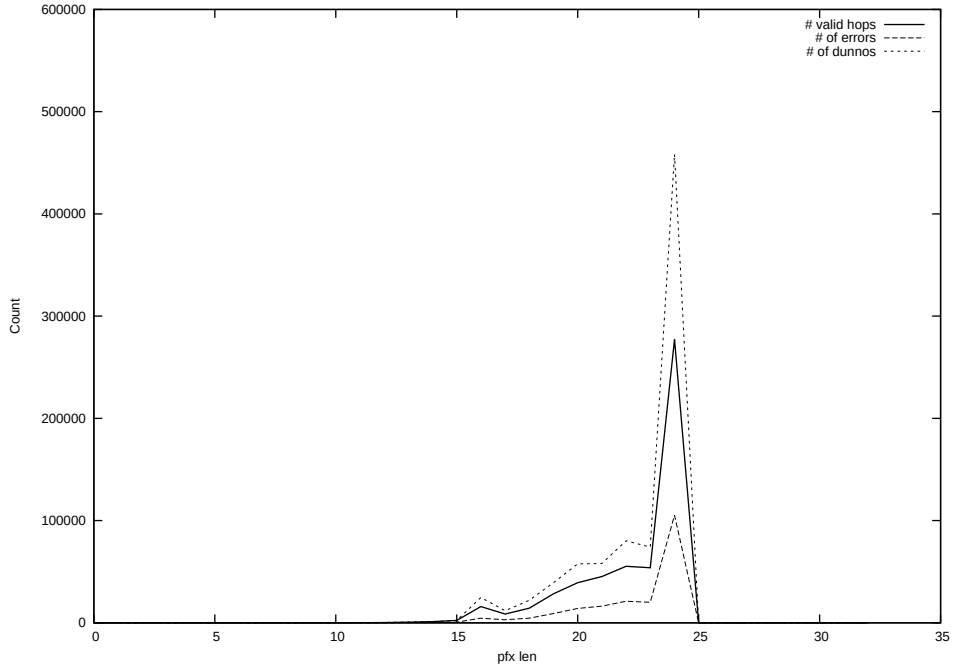
2012-05-20



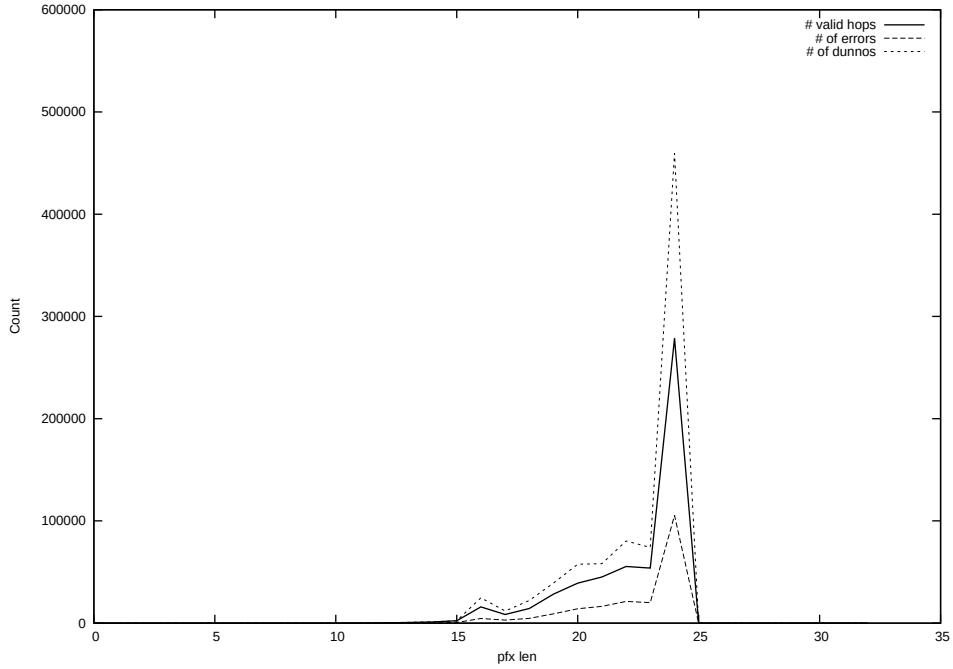
2012-05-21



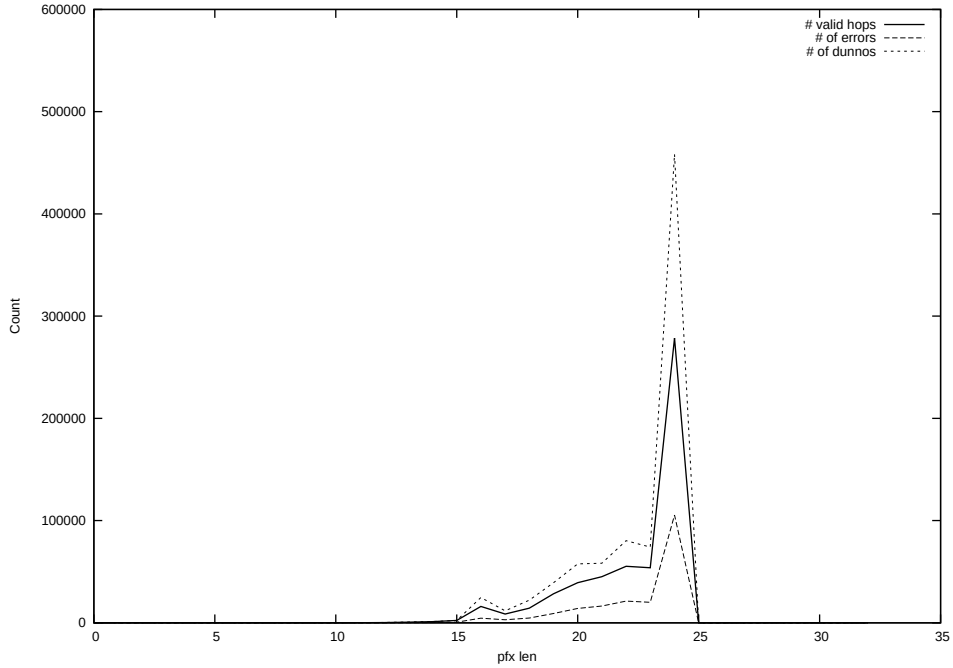
2012-05-22



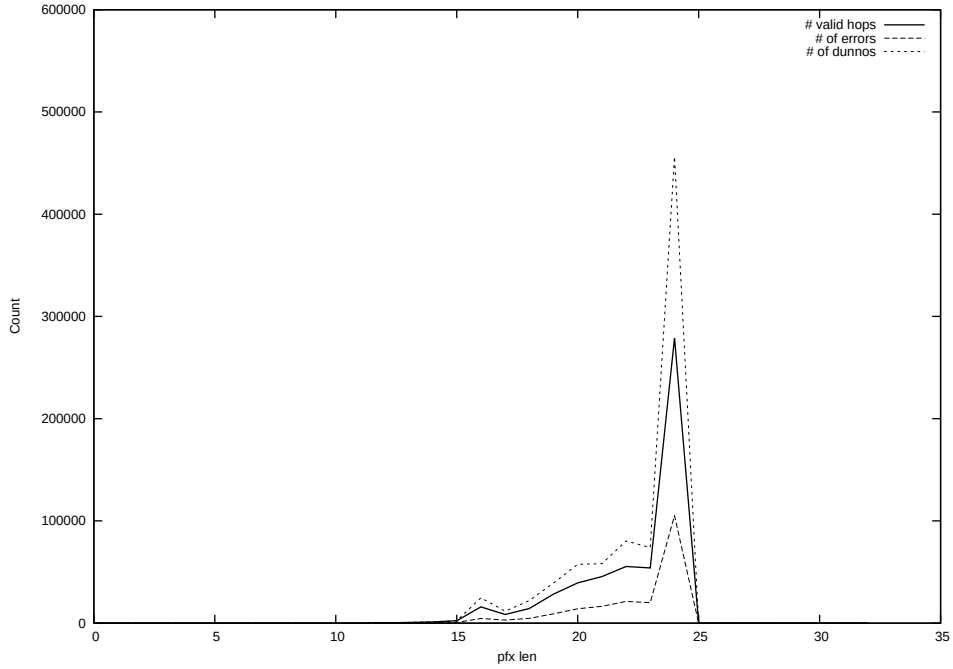
2012-05-23



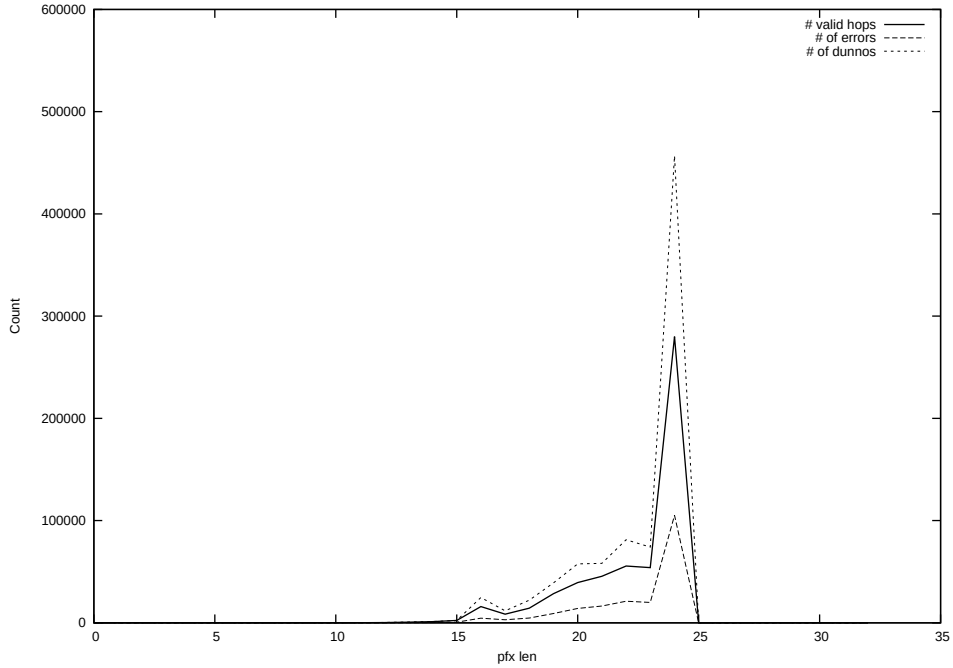
2012-05-24



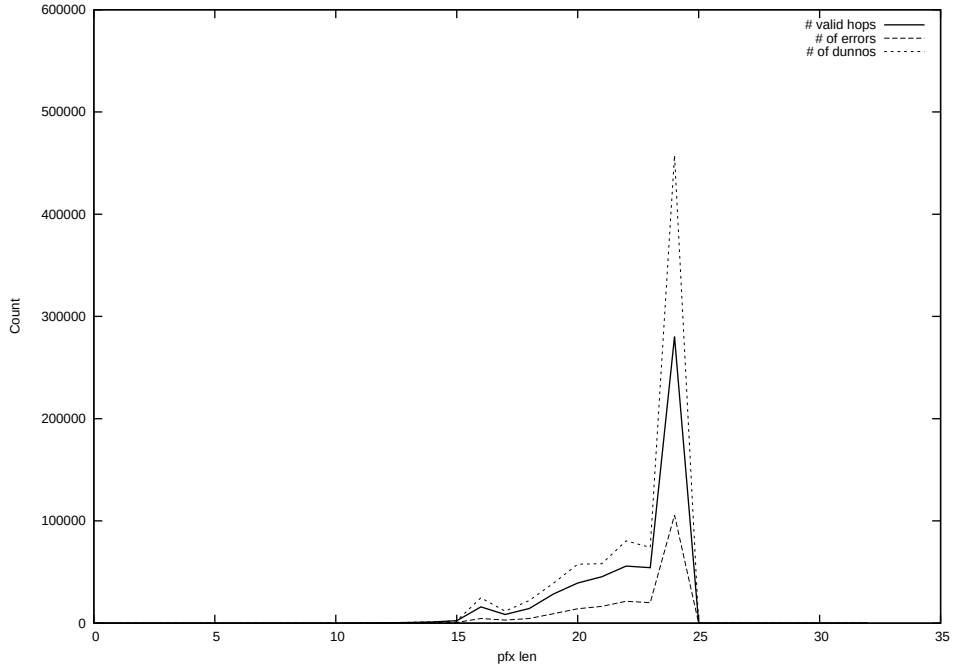
2012-05-25



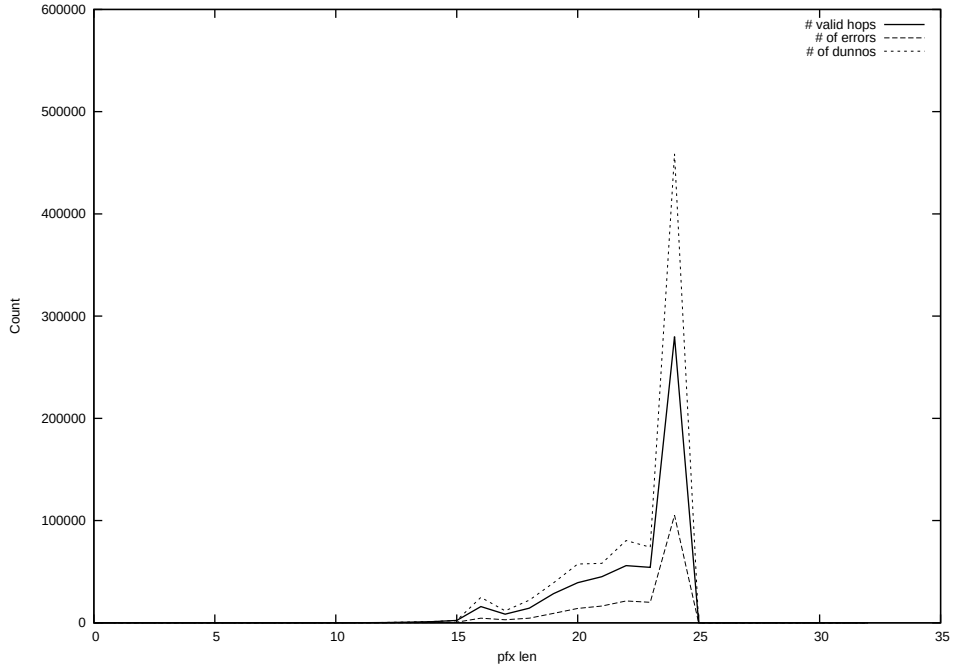
2012-05-26



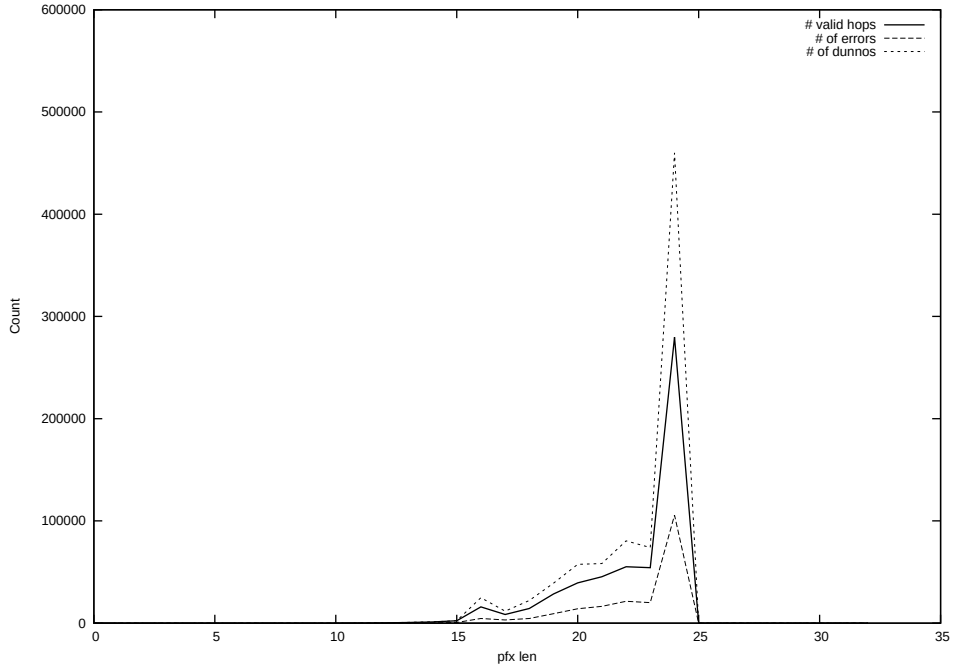
2012-05-27



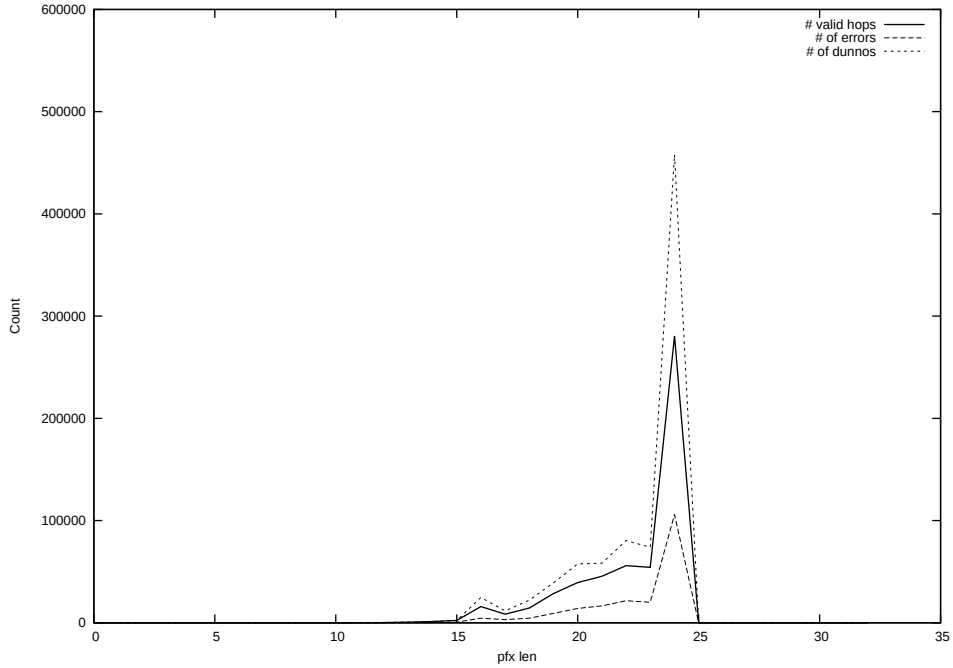
2012-05-28



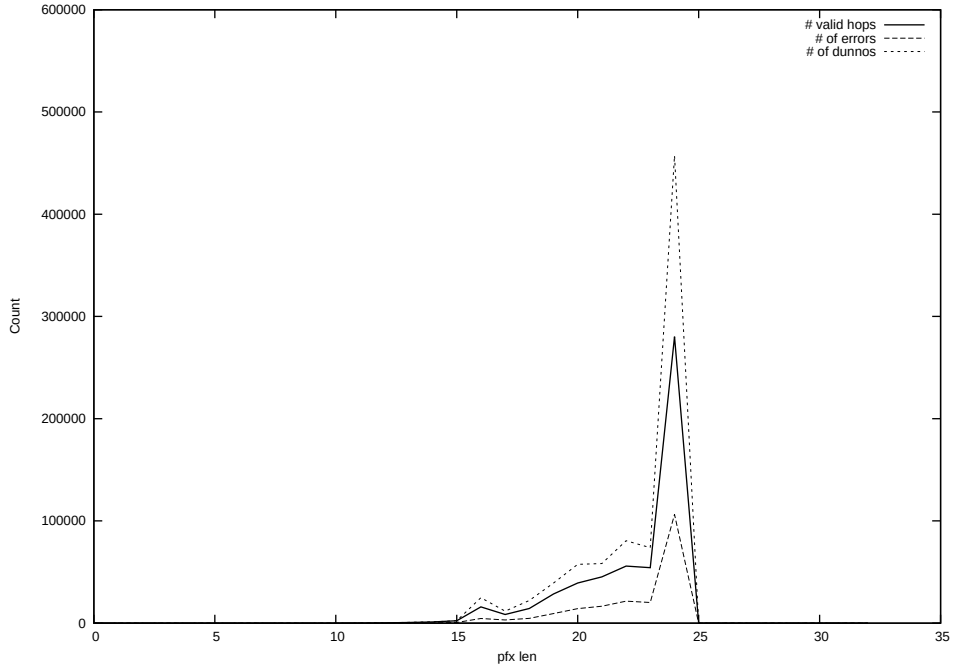
2012-05-29



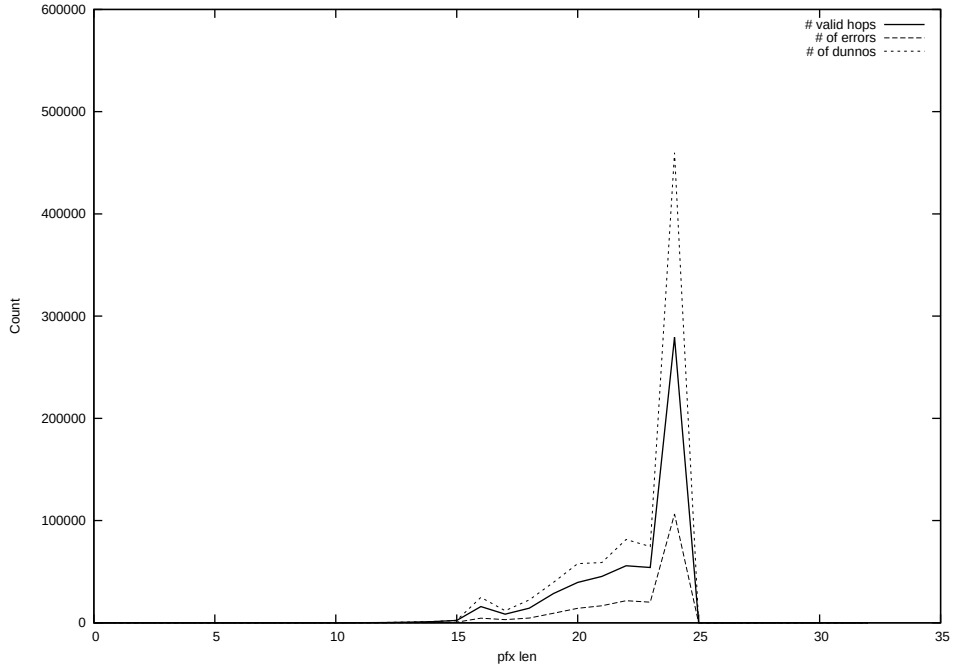
2012-05-30



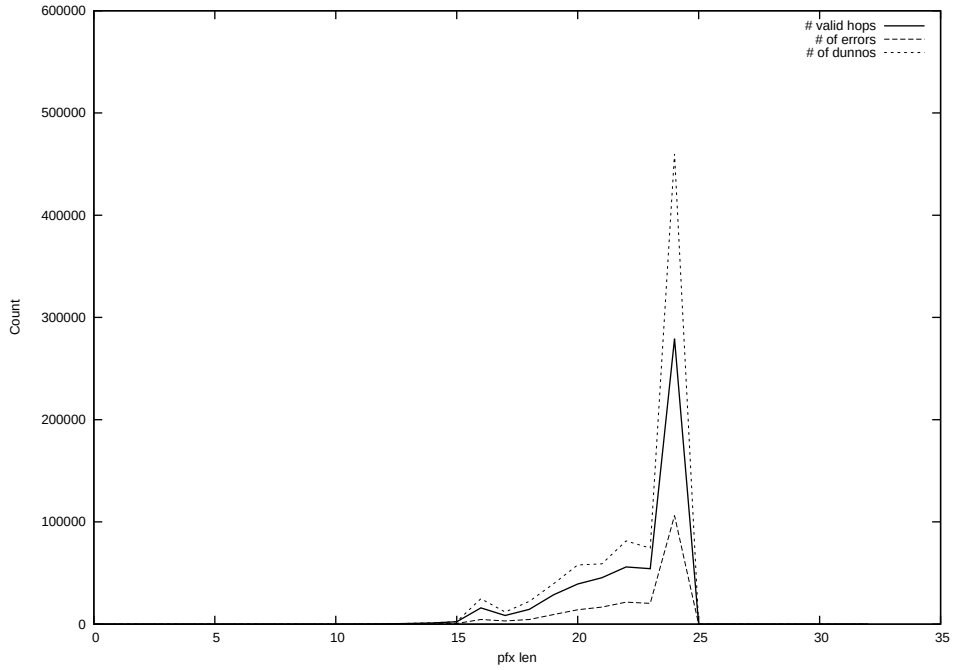
2012-05-31



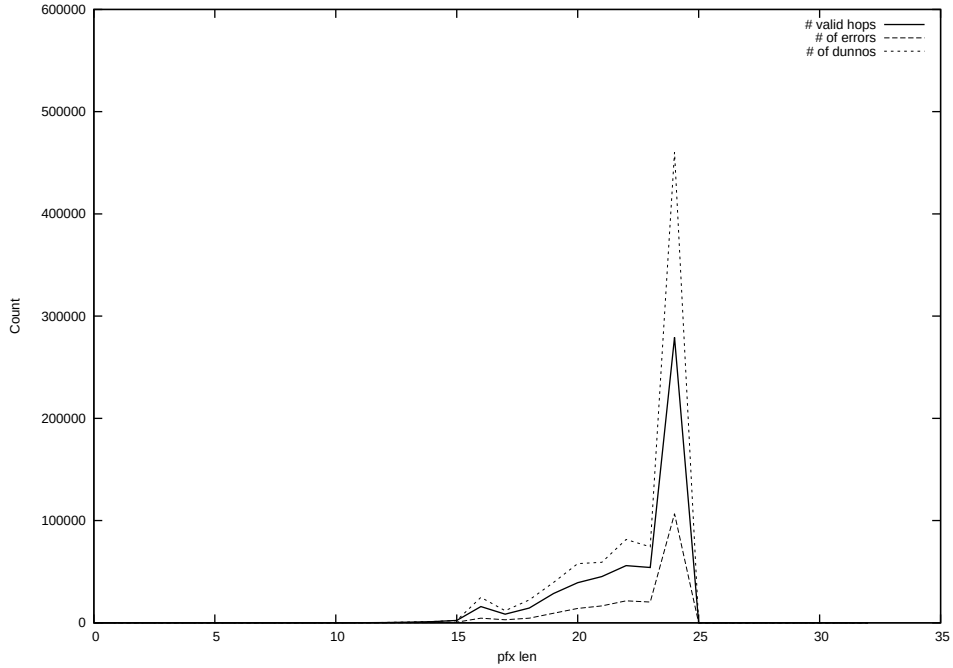
2012-06-01



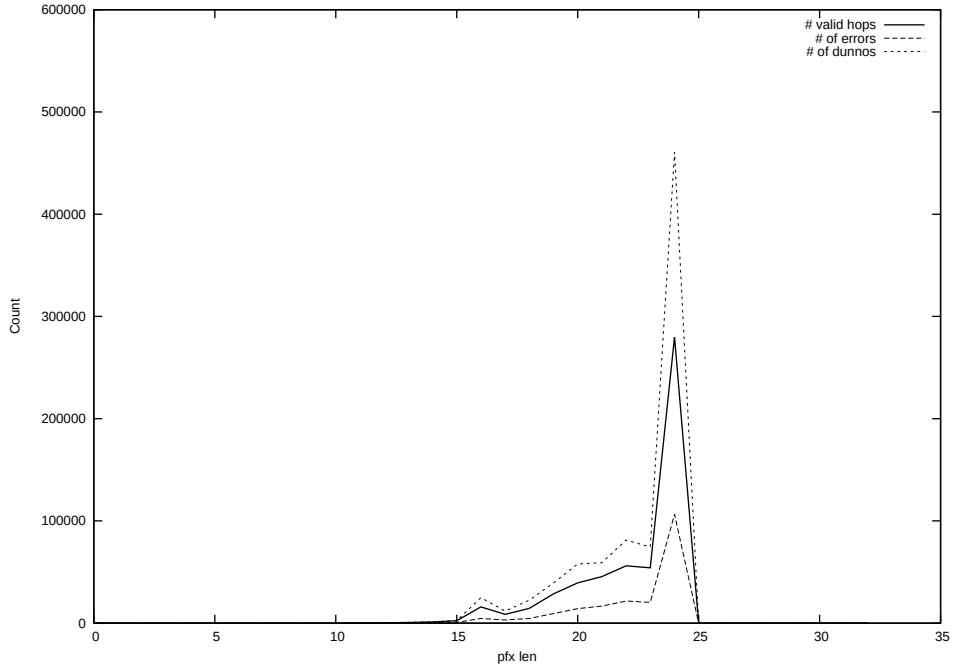
2012-06-02



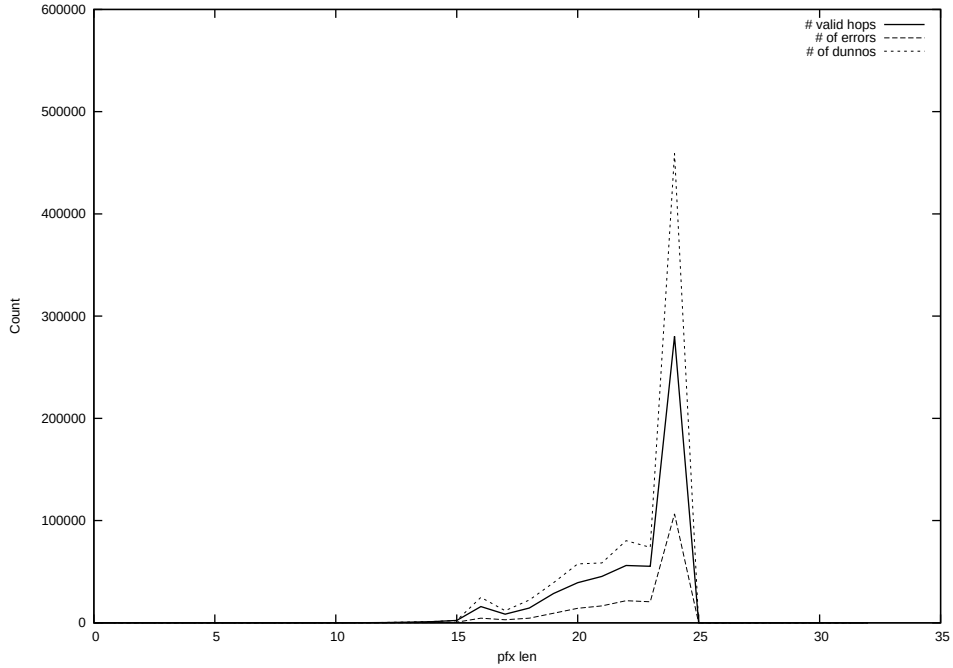
2012-06-03



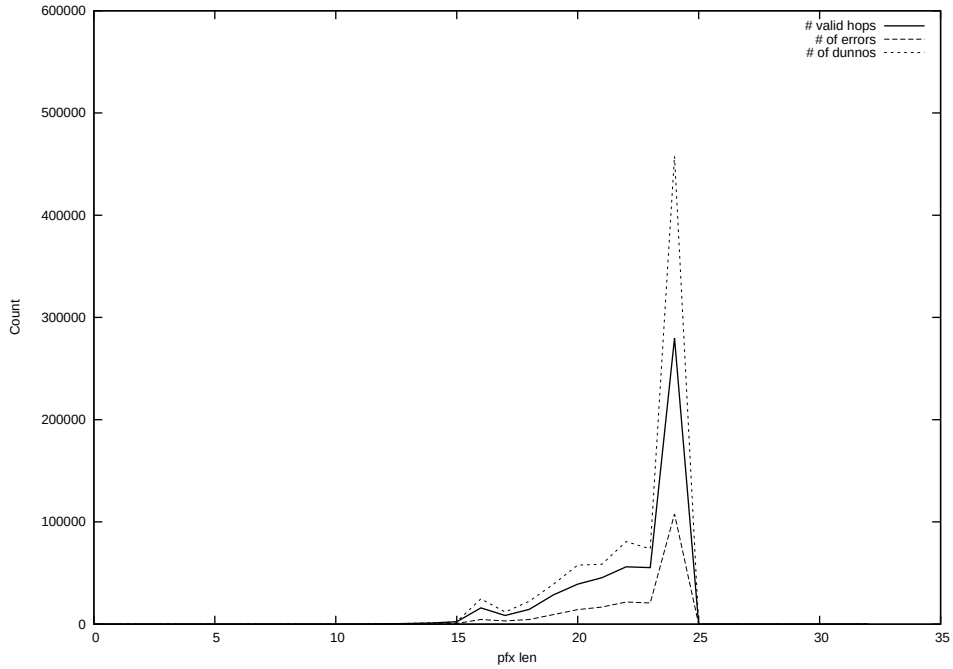
2012-06-04



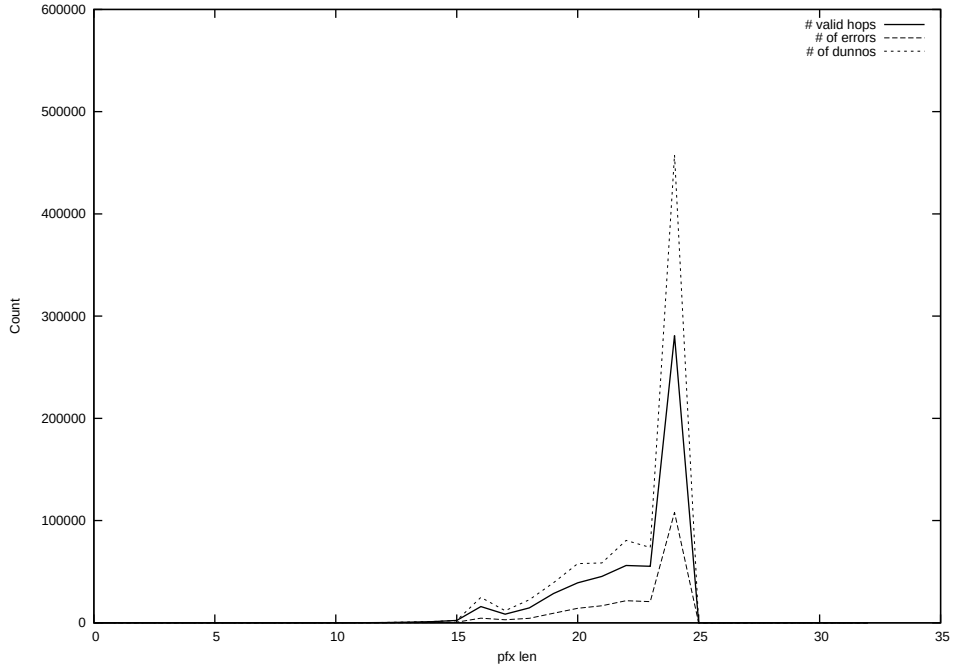
2012-06-05



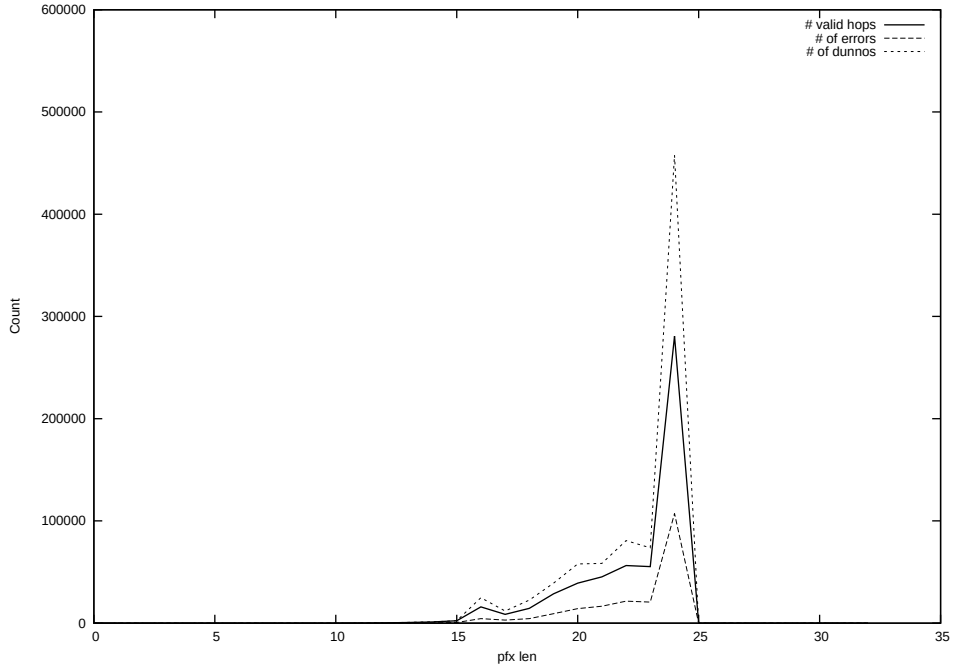
2012-06-06



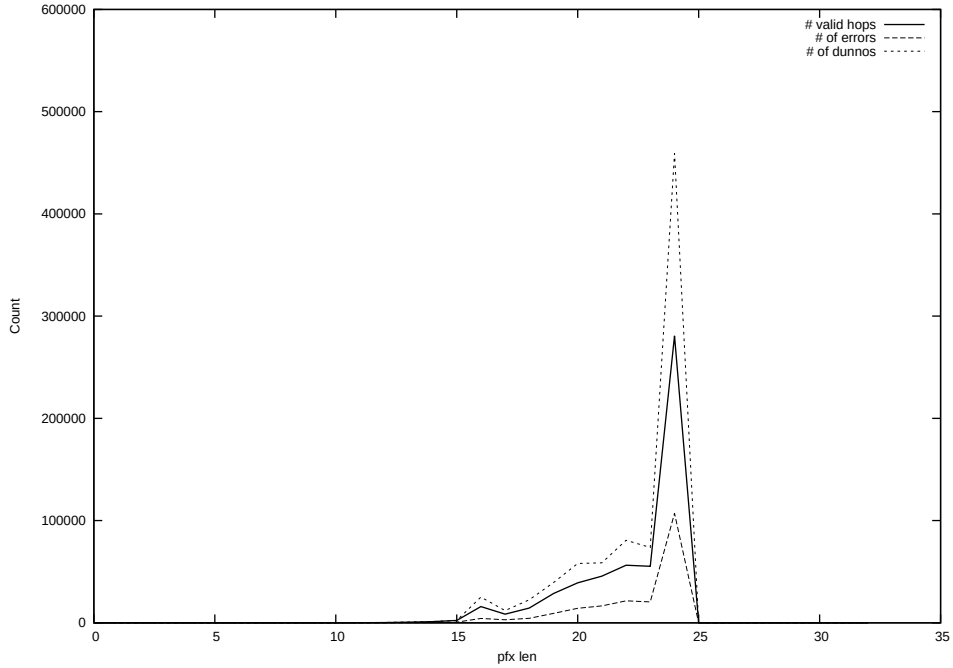
2012-06-07



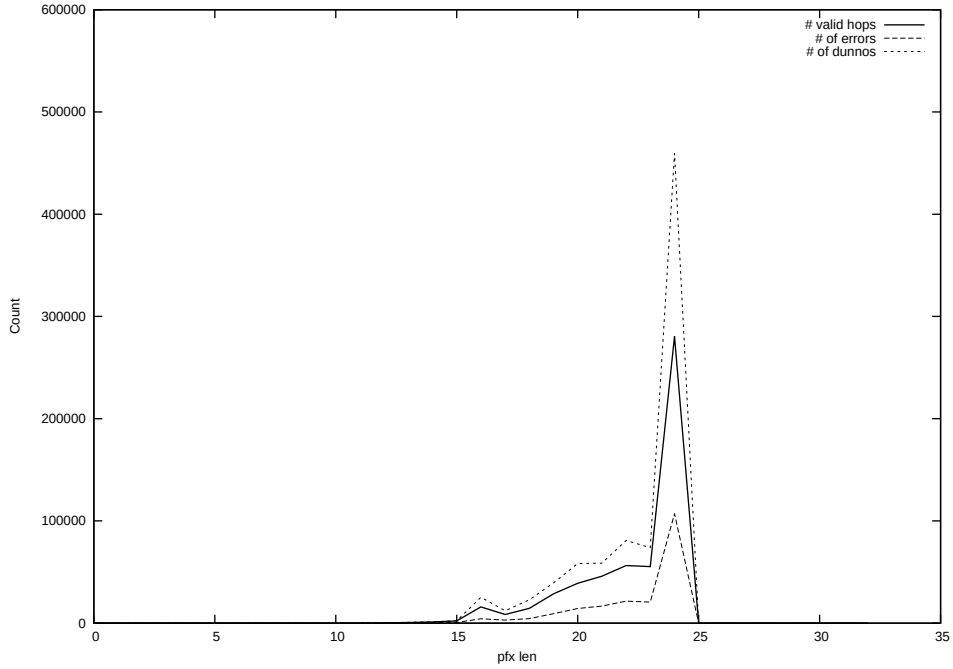
2012-06-08



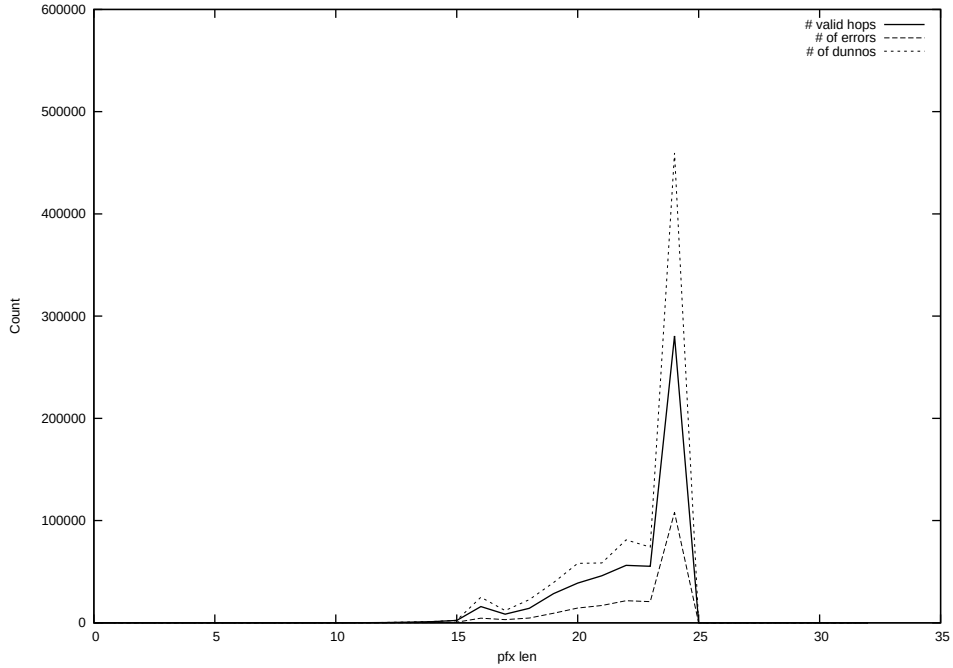
2012-06-10



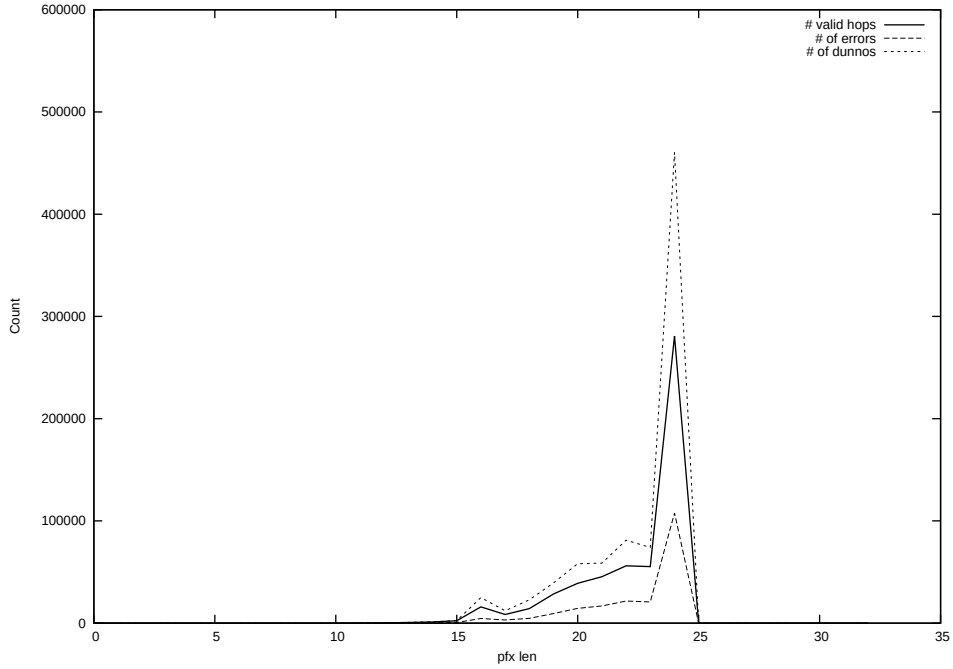
2012-06-11



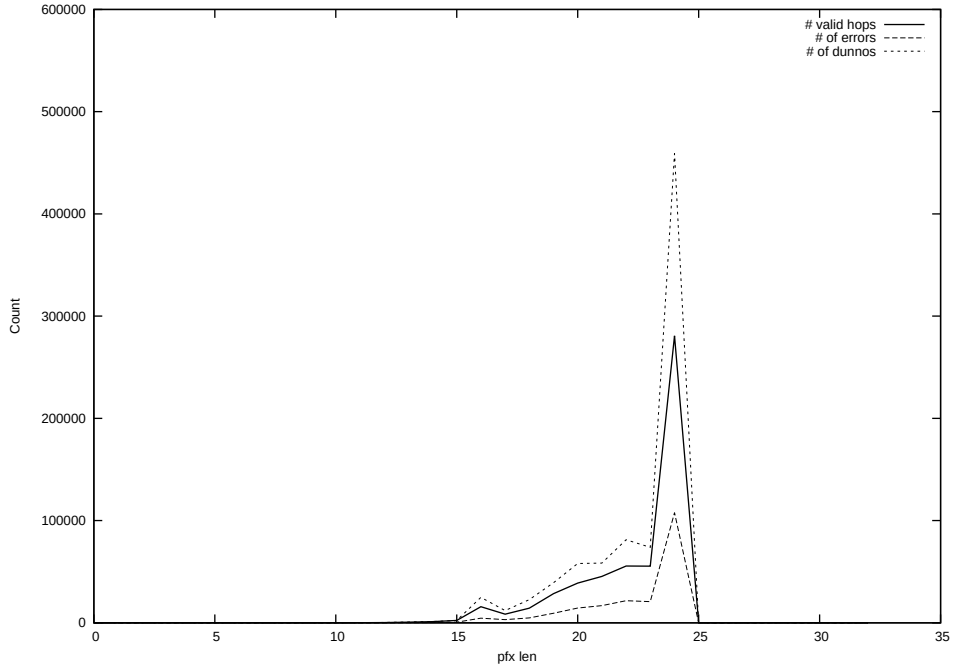
2012-06-12



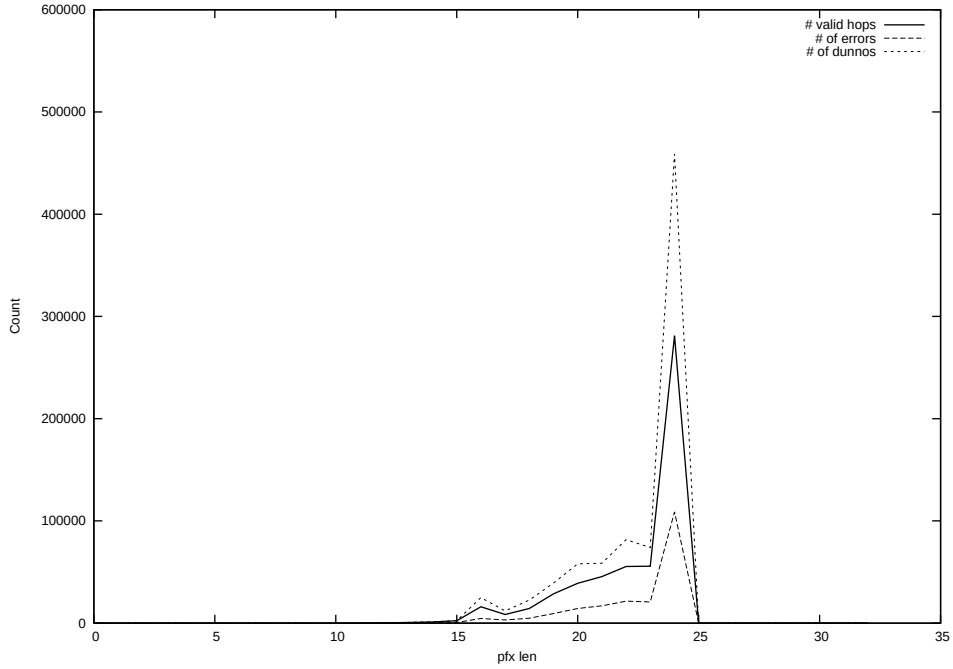
2012-06-13



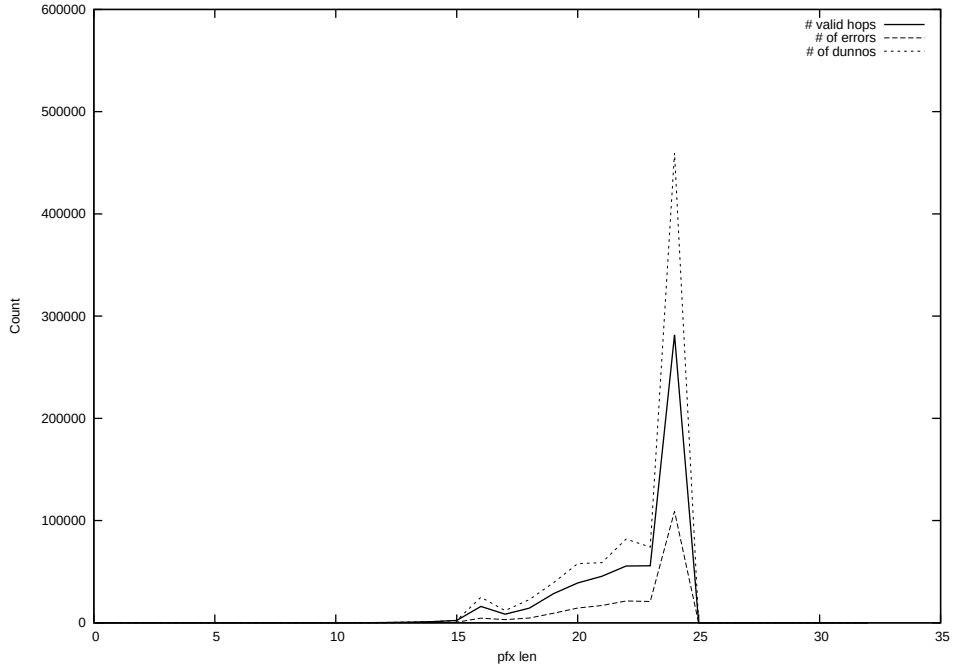
2012-06-14



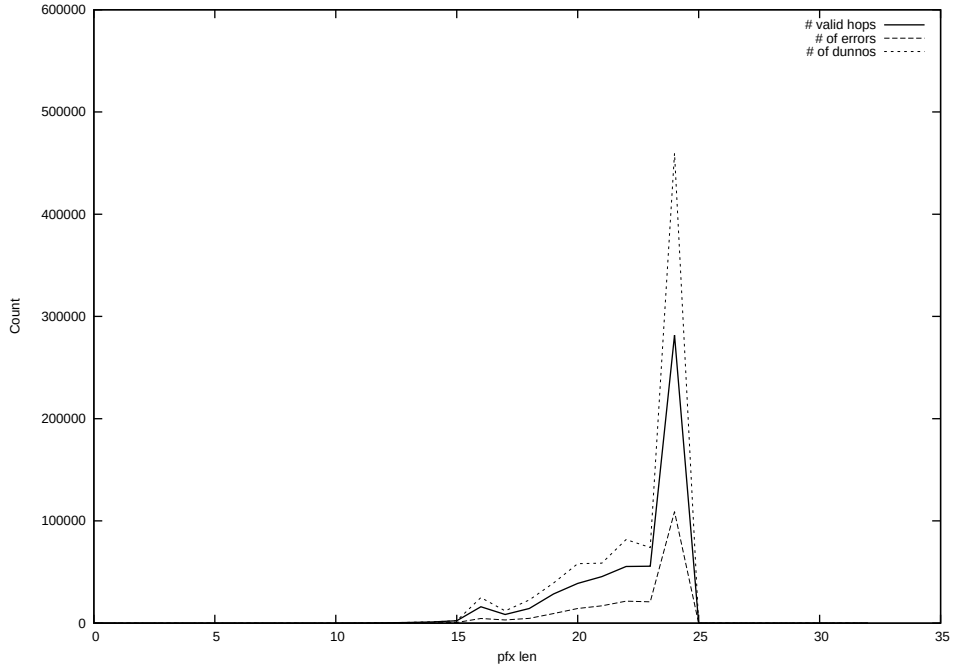
2012-06-15



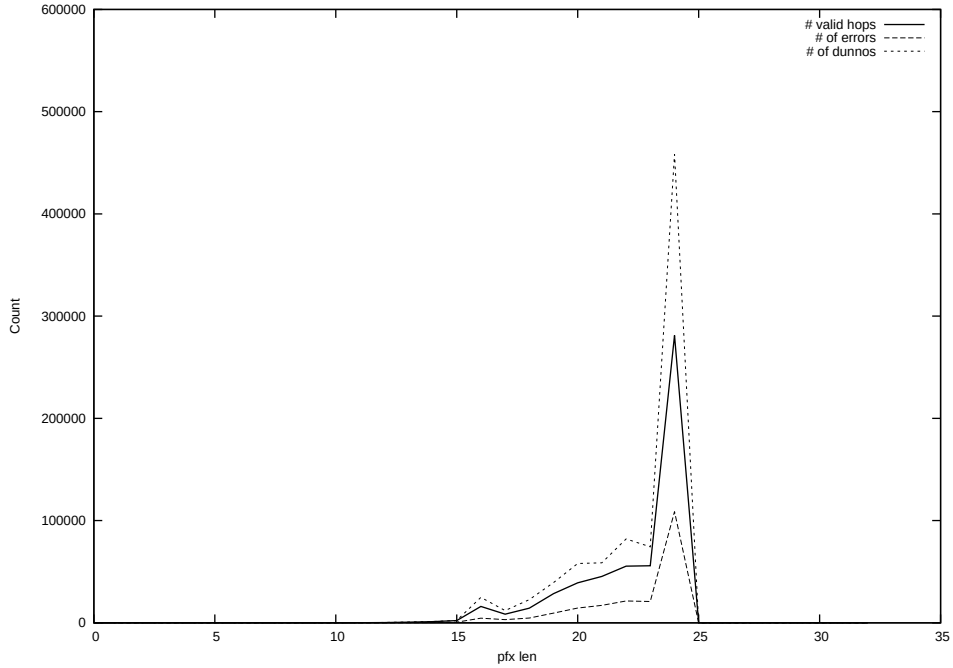
2012-06-16



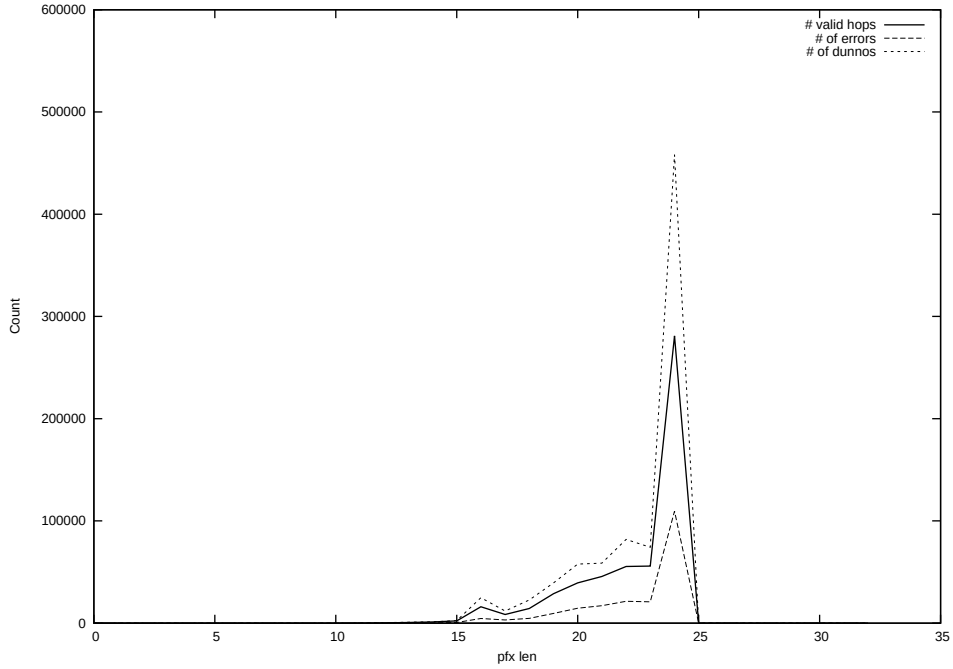
2012-06-17



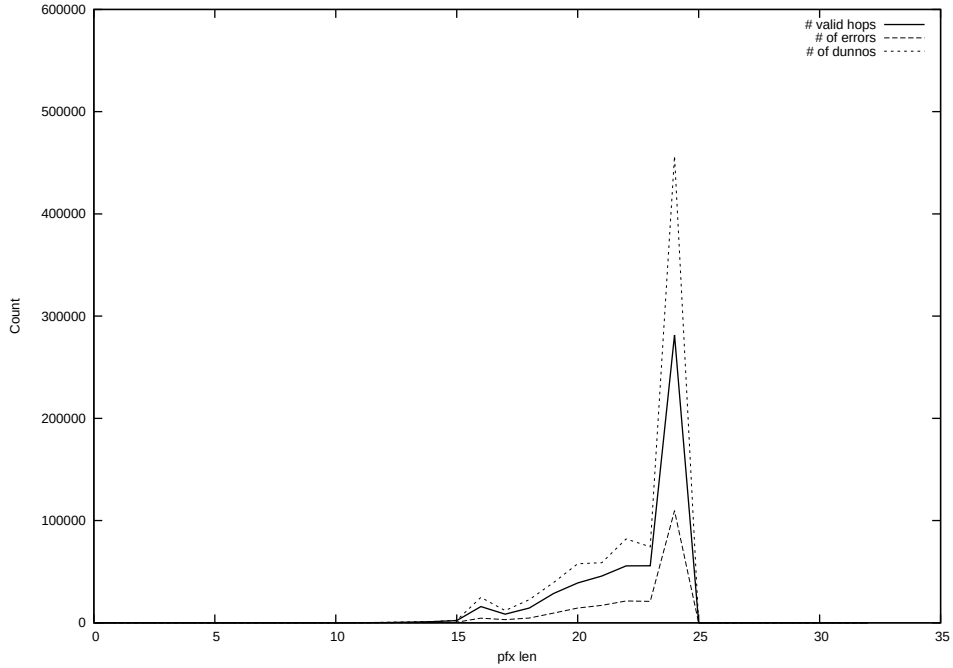
2012-06-18



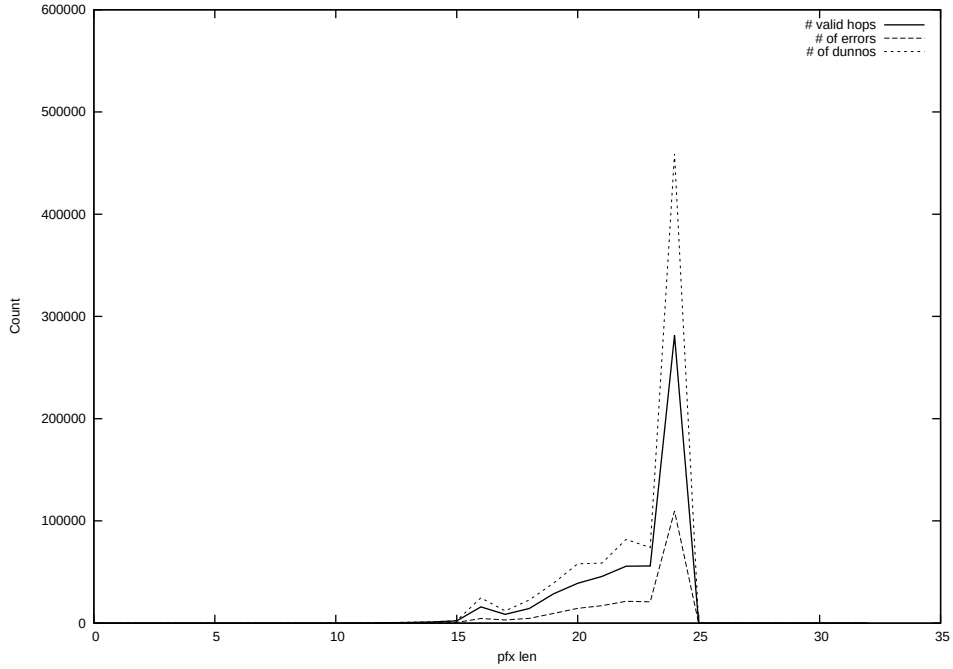
2012-06-19



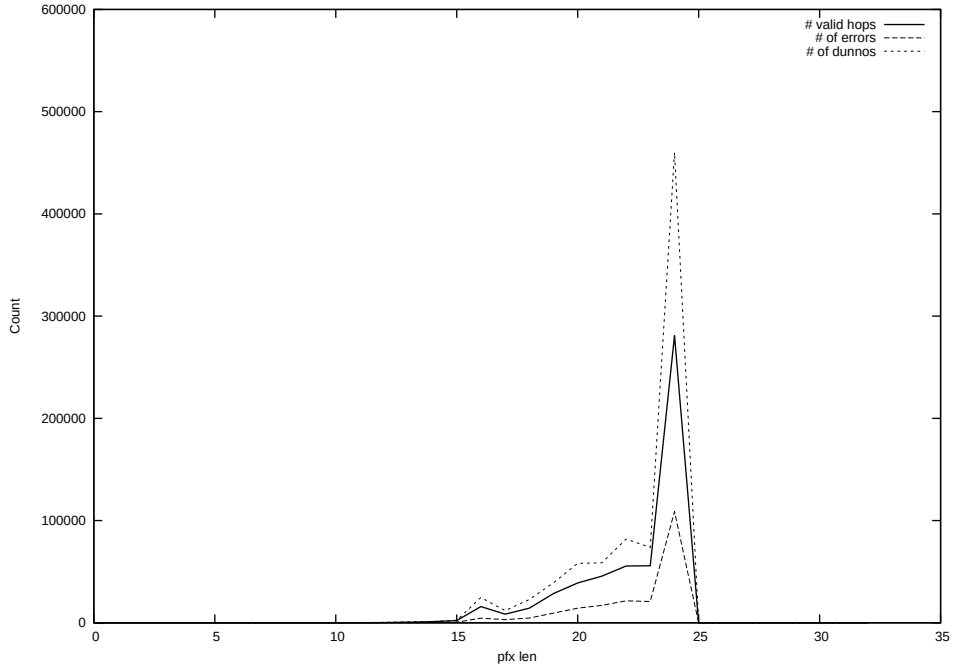
2012-06-20



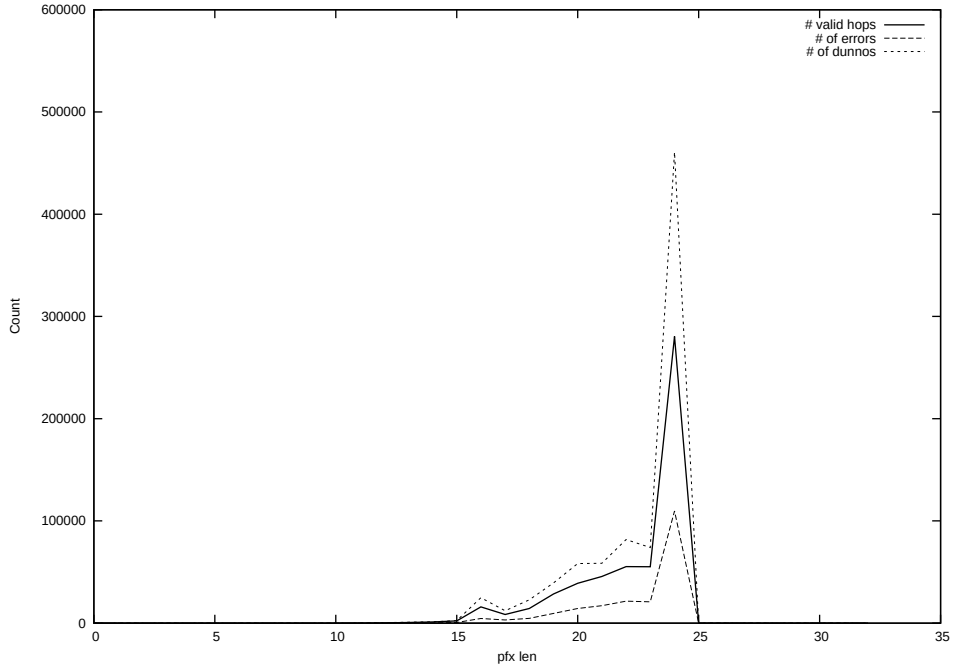
2012-06-21



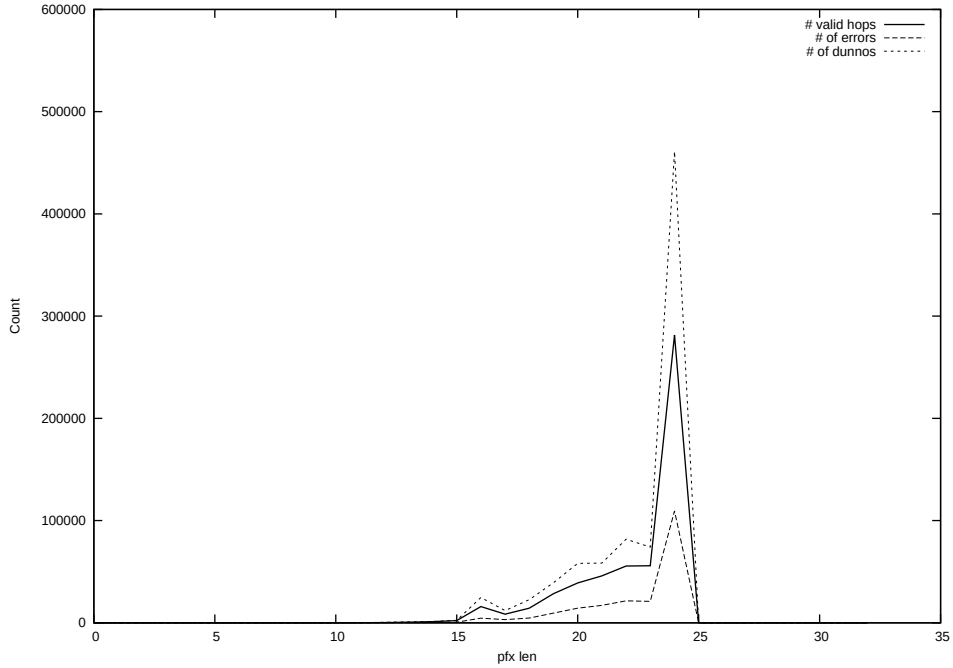
2012-06-22



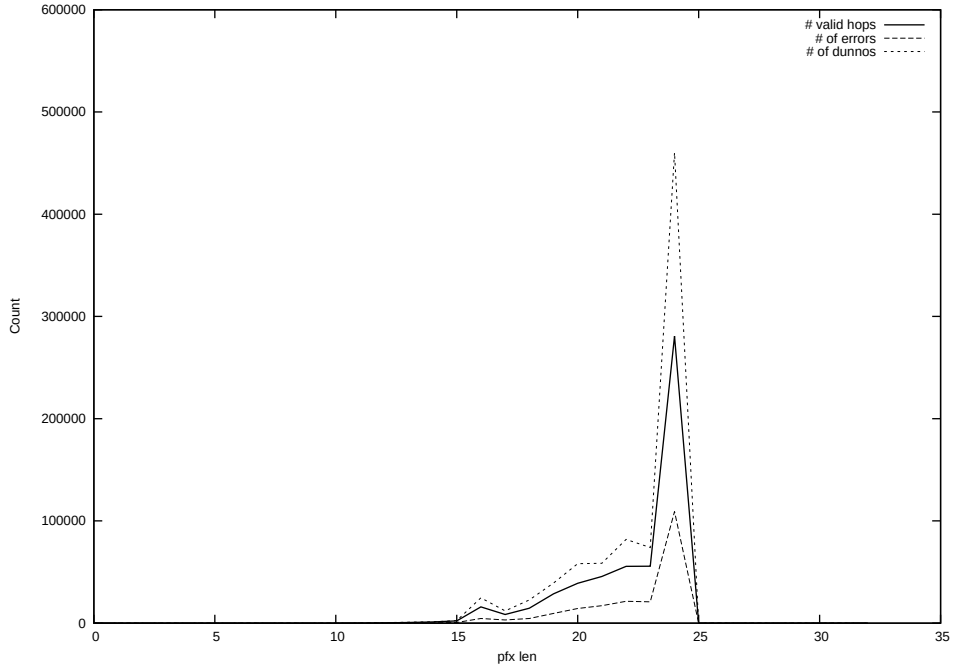
2012-06-23



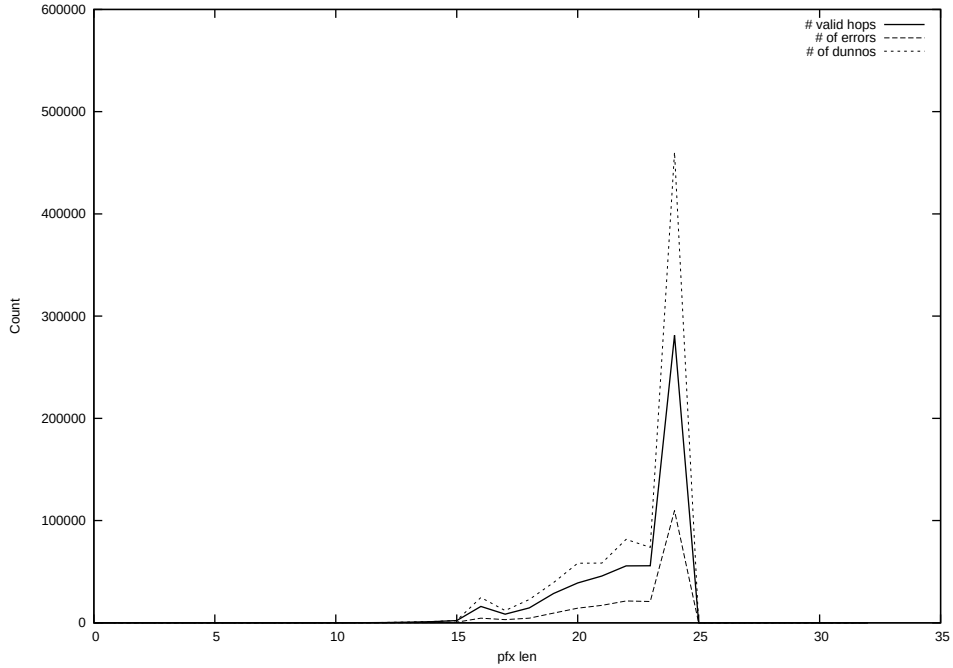
2012-06-24



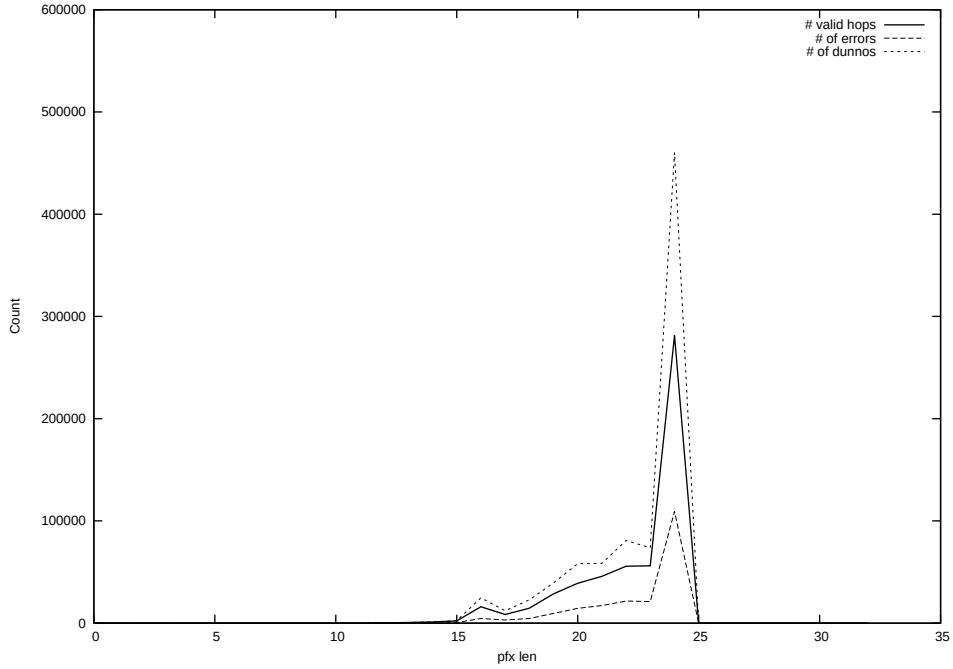
2012-06-25



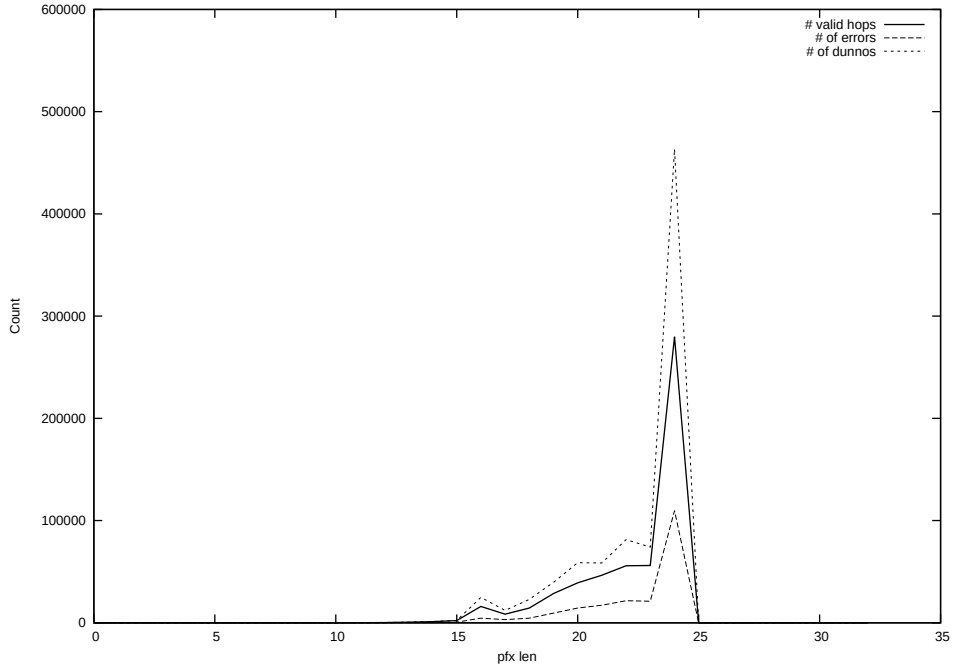
2012-06-26



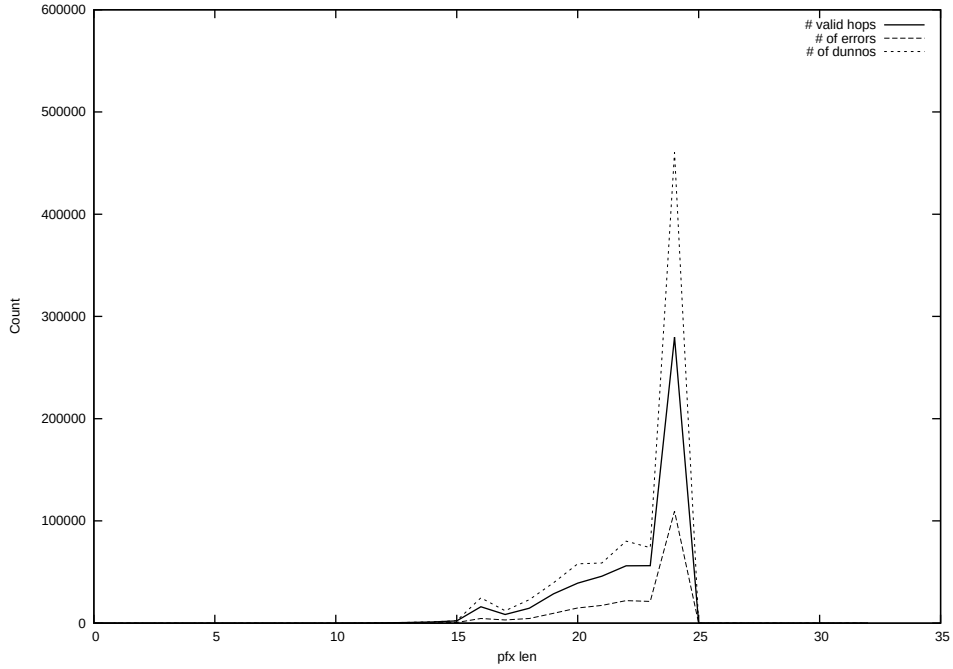
2012-06-27



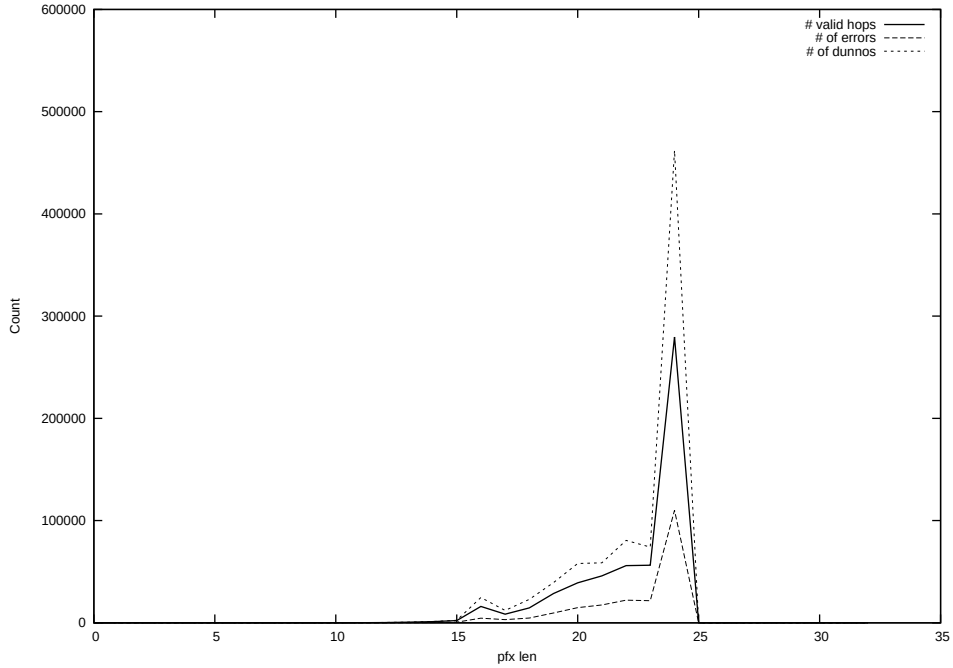
2012-06-28



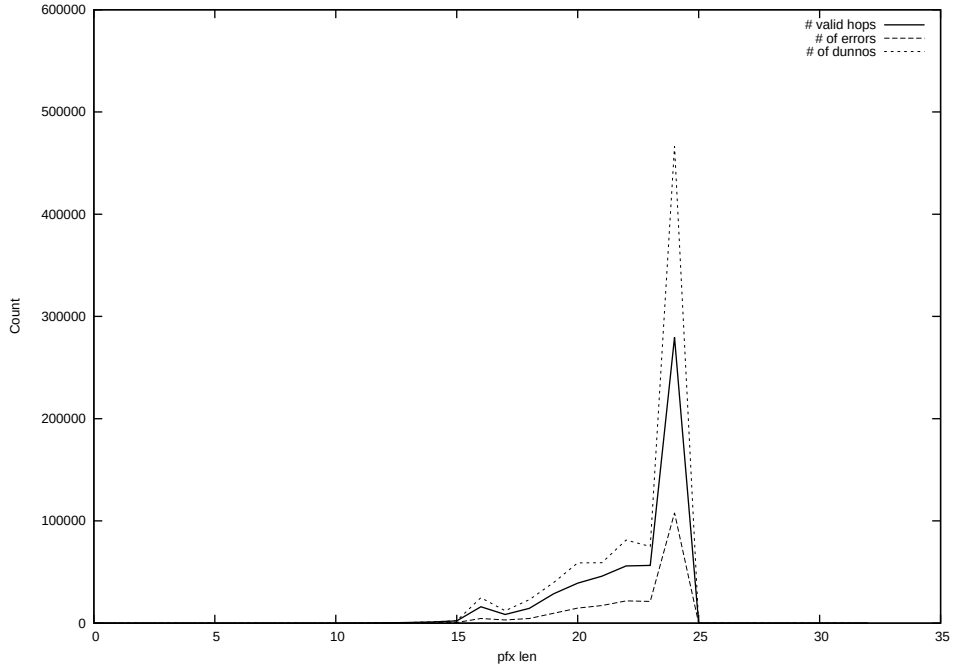
2012-06-29



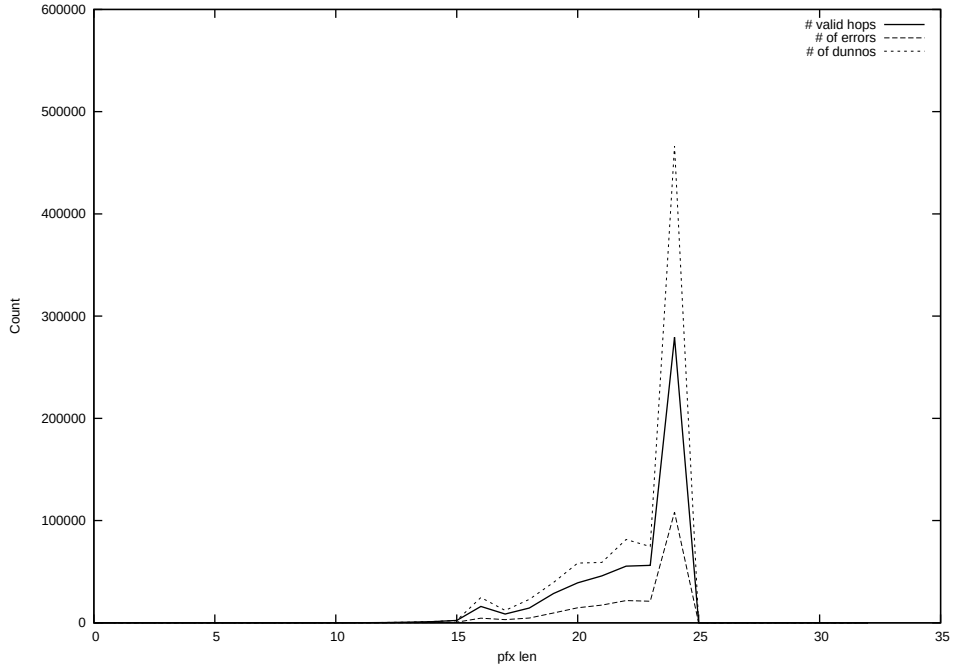
2012-06-30



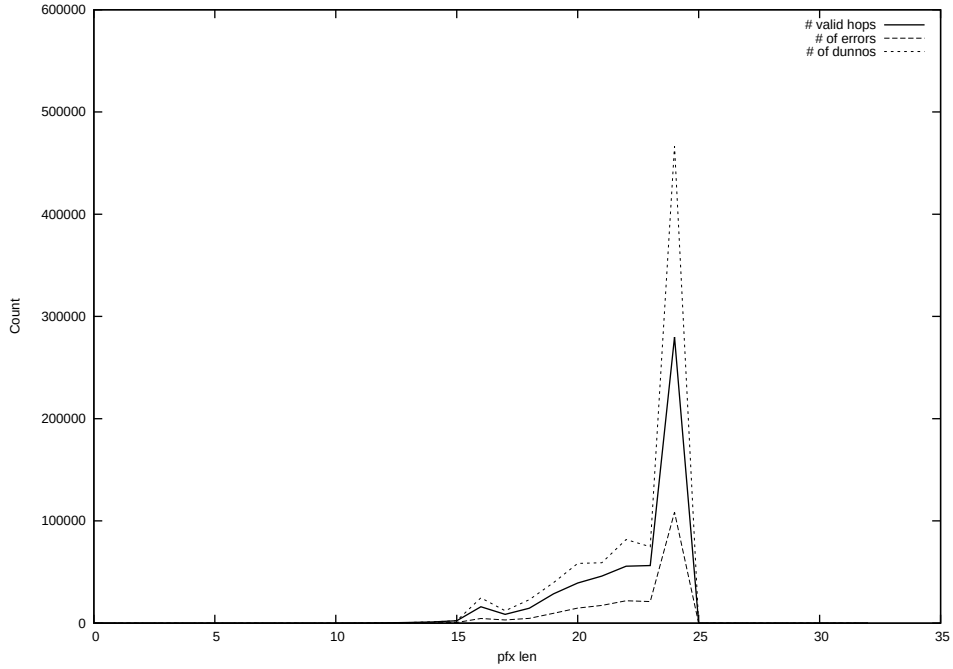
2012-07-01



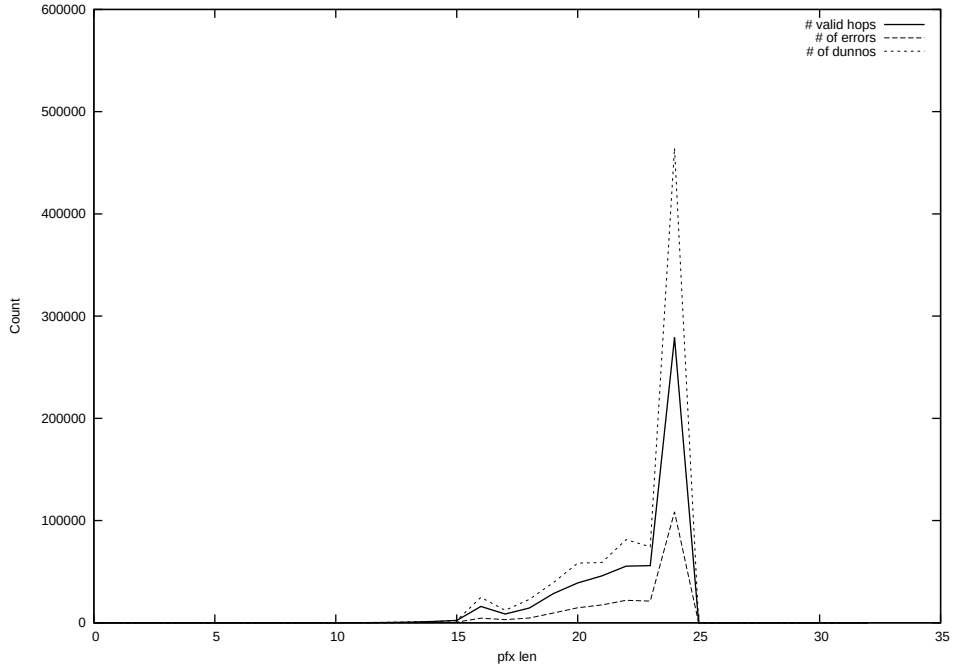
2012-07-02



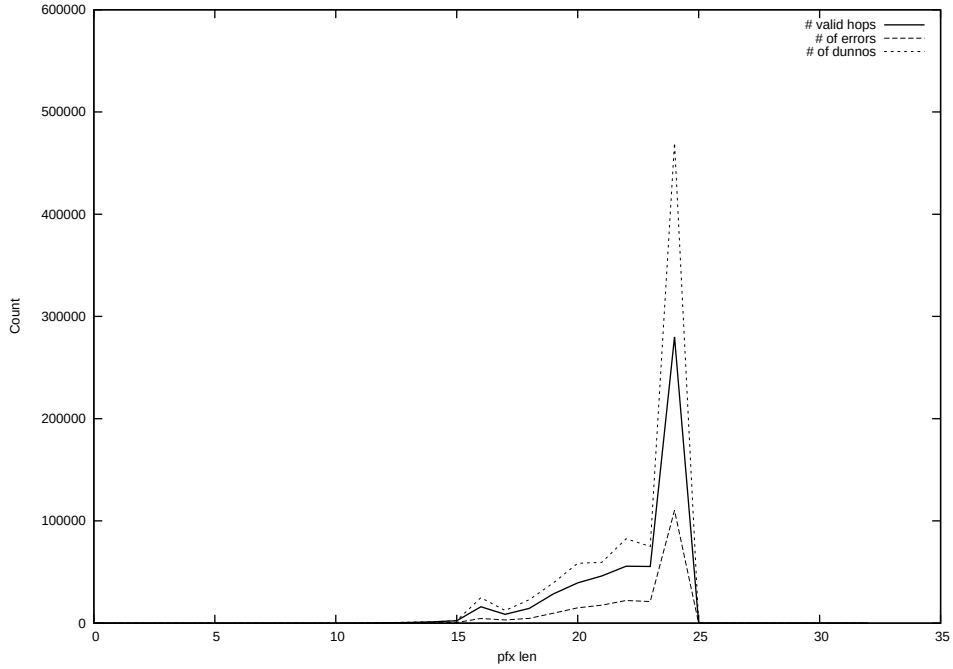
2012-07-03



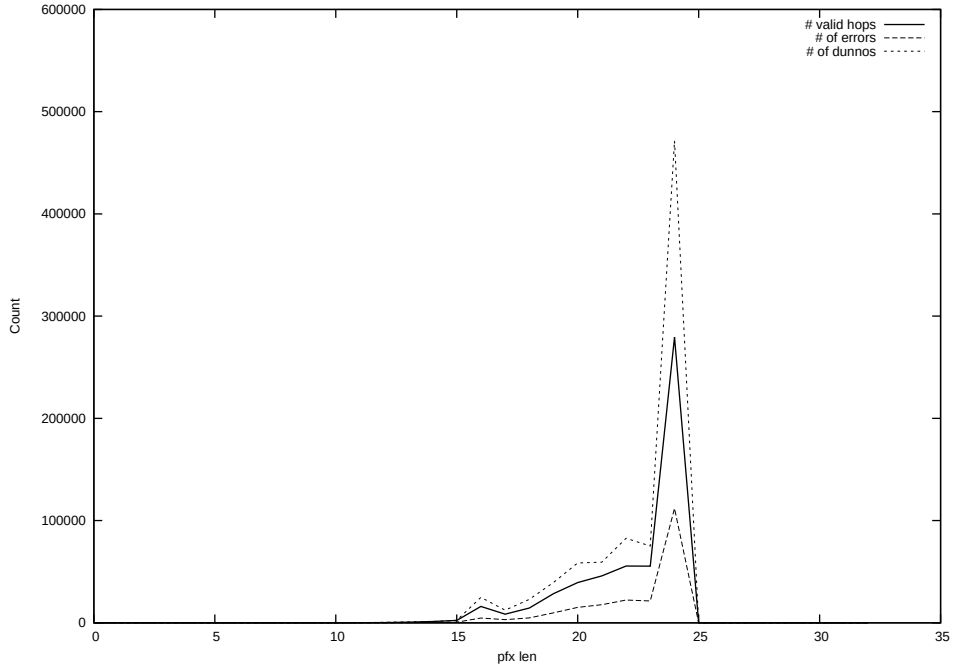
2012-07-04



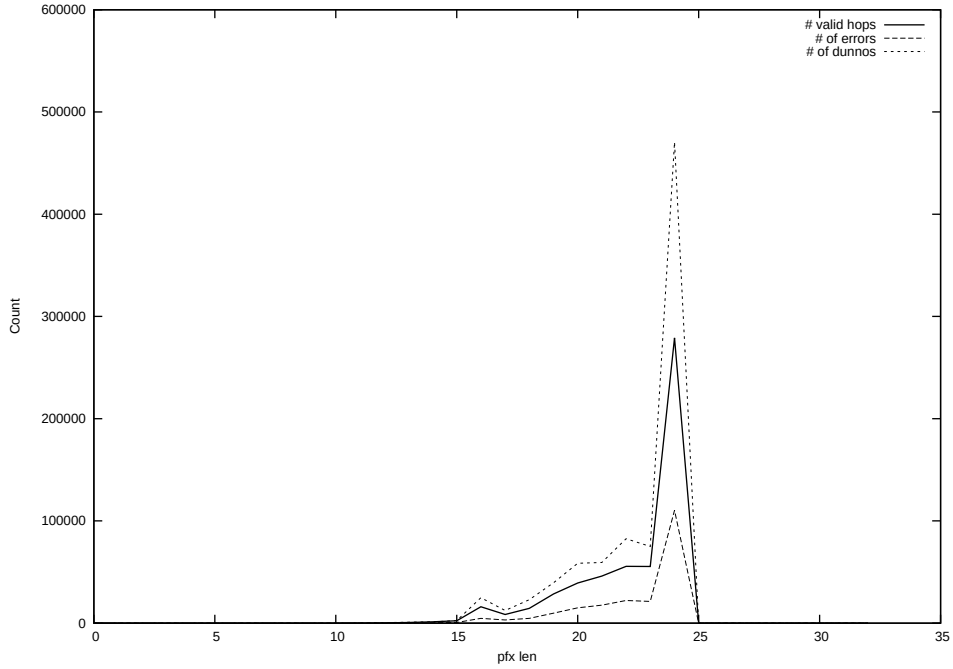
2012-07-05



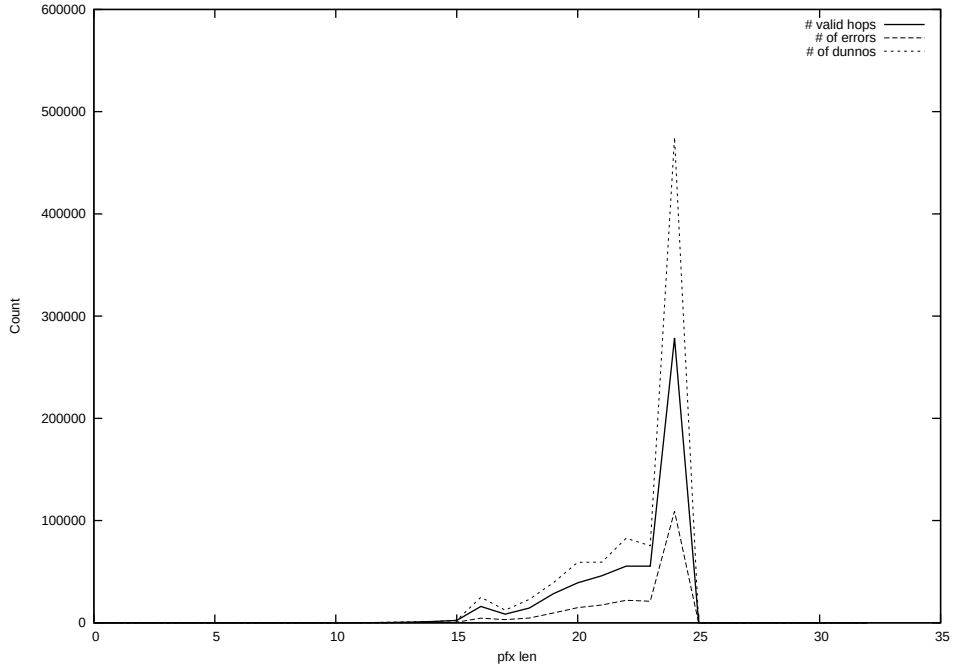
2012-07-06



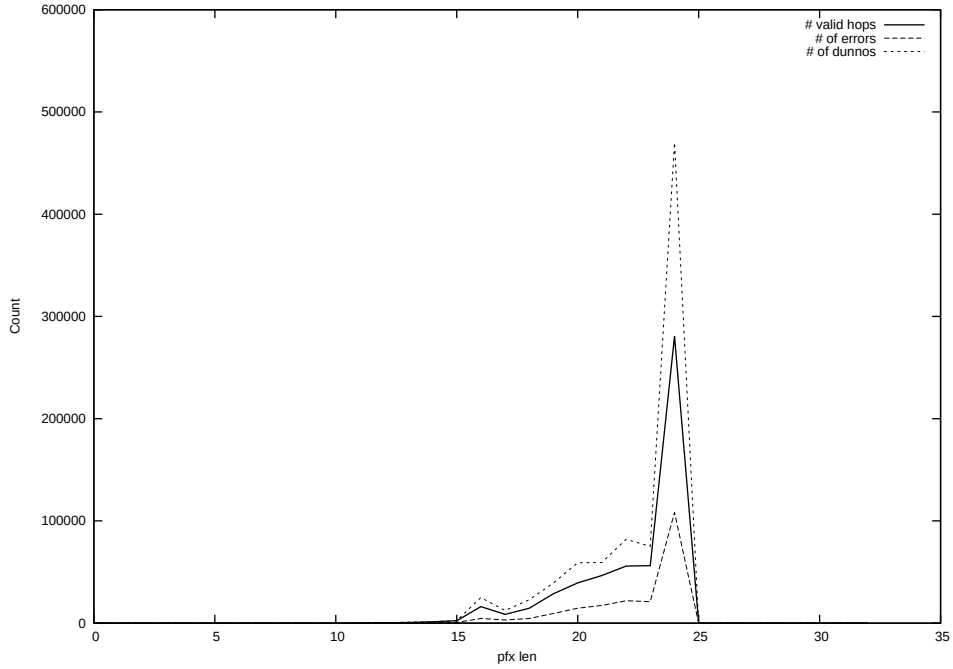
2012-07-07



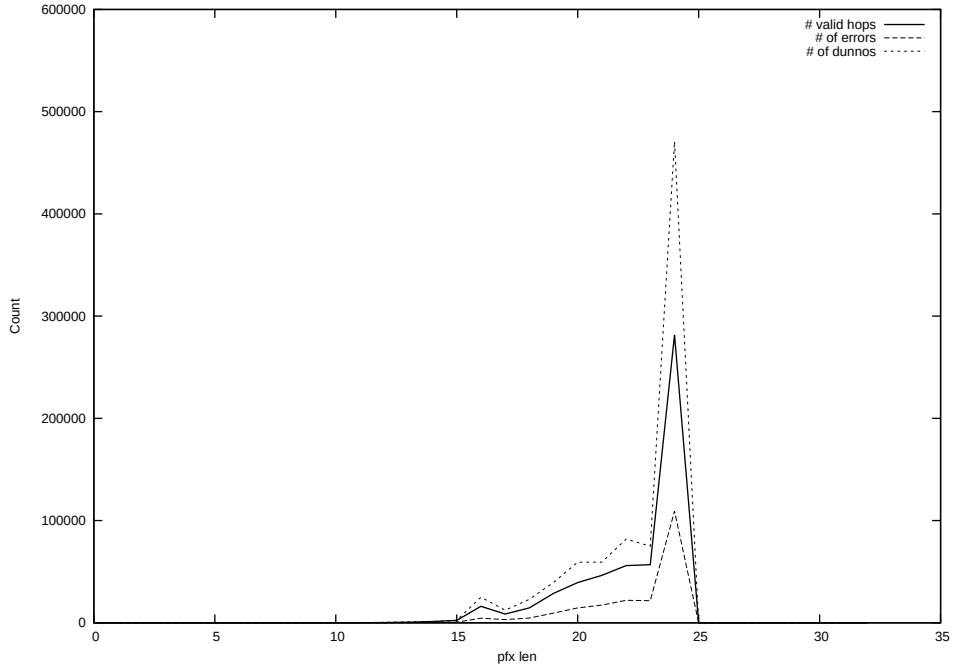
2012-07-08



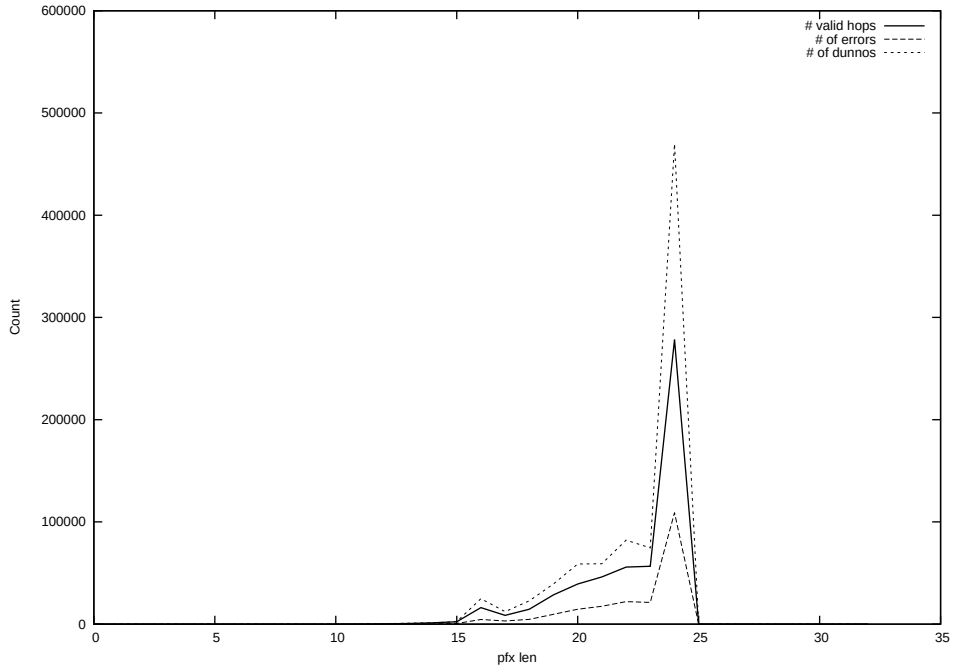
2012-07-09



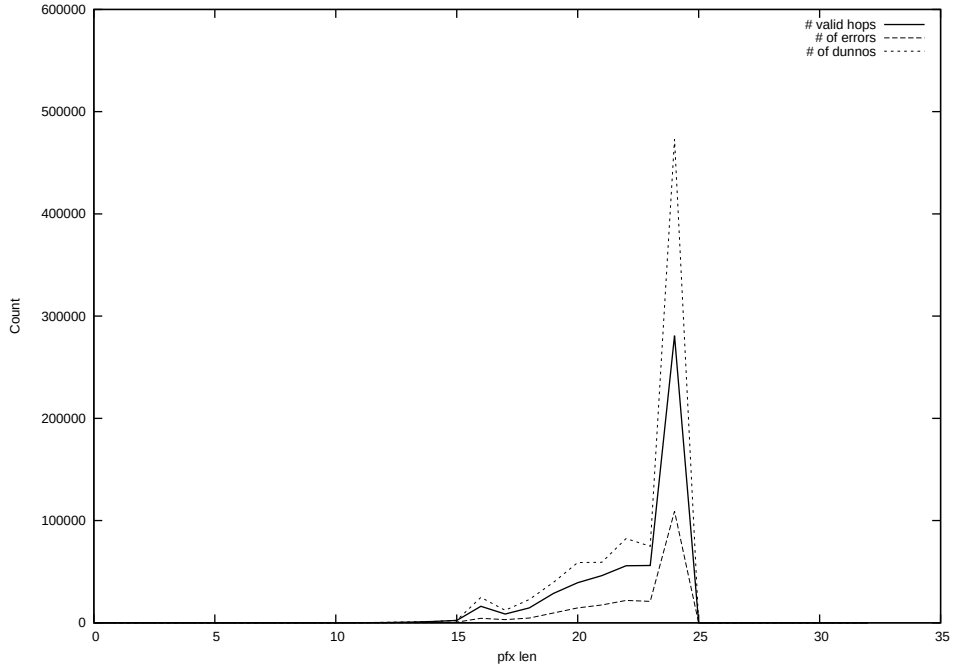
2012-07-10



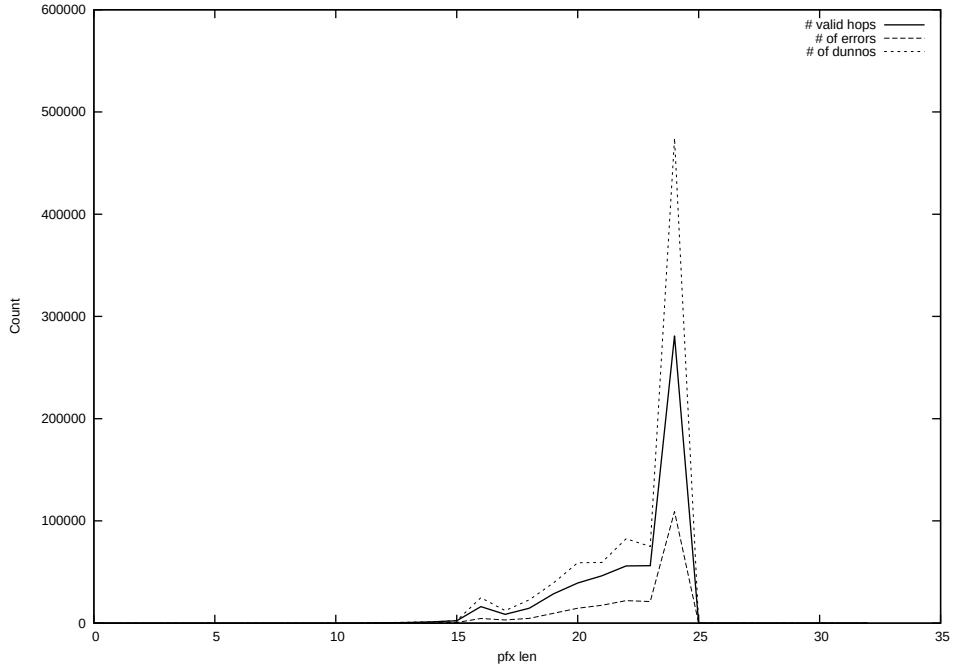
2012-07-11



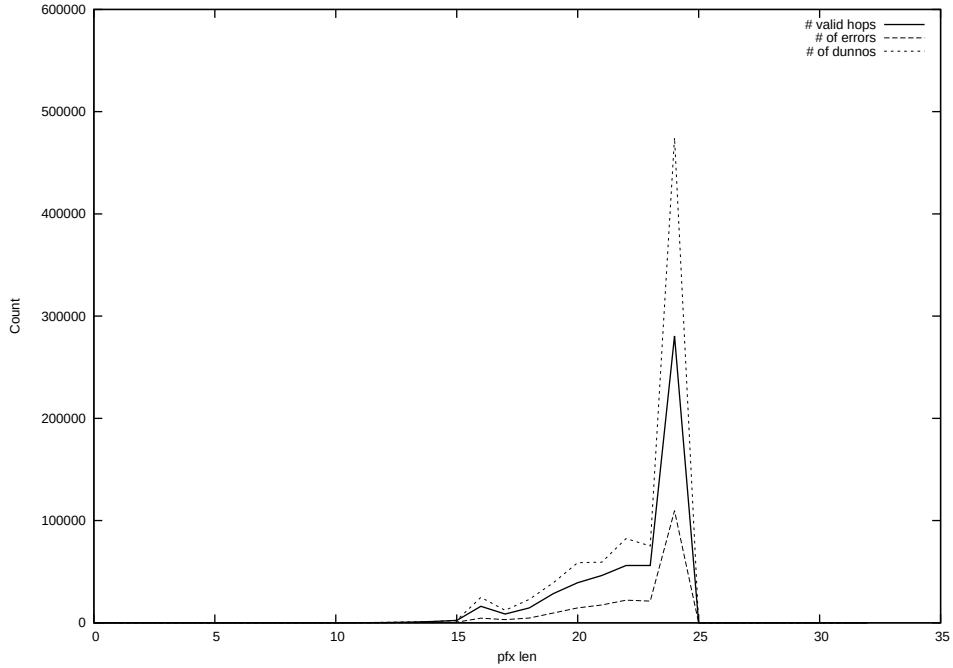
2012-07-12



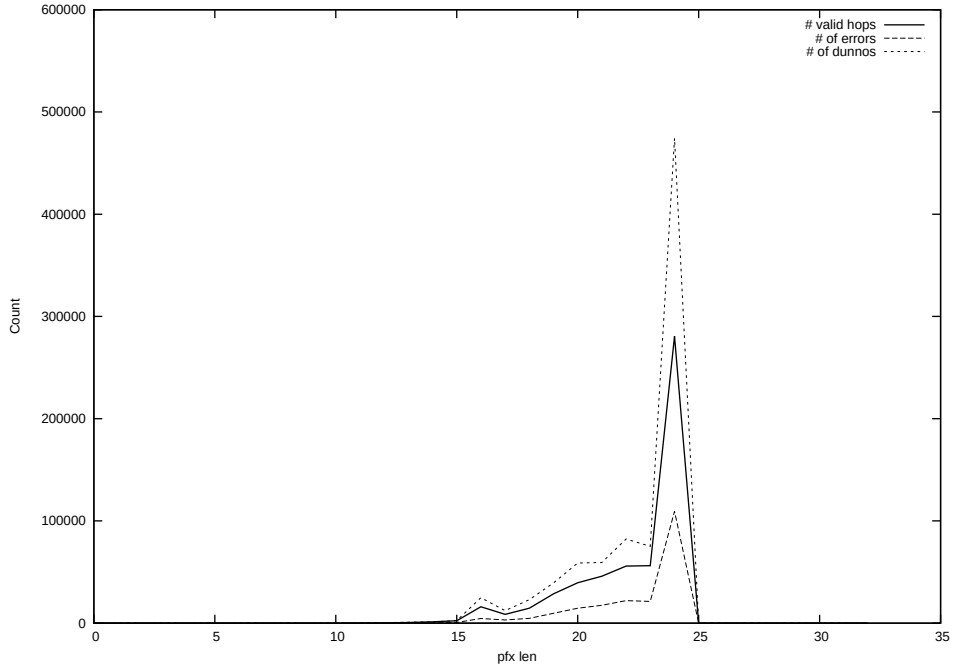
2012-07-13



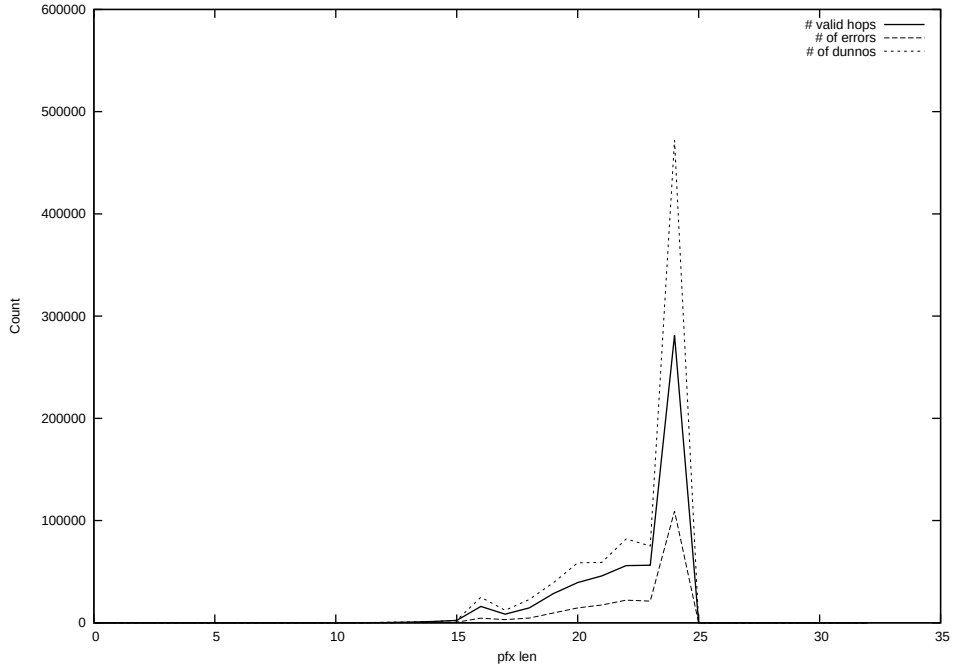
2012-07-14



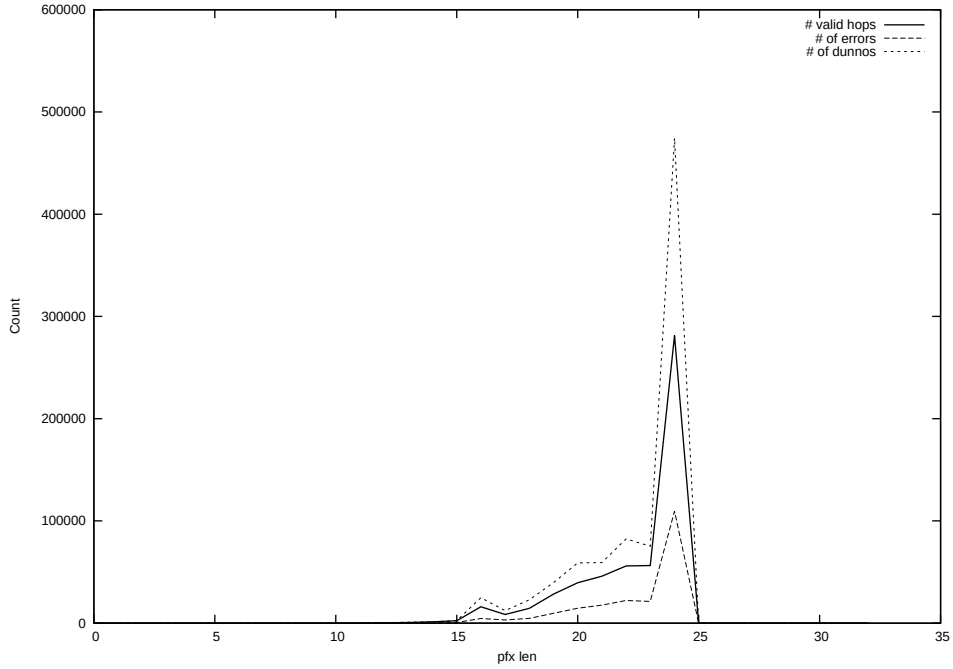
2012-07-15



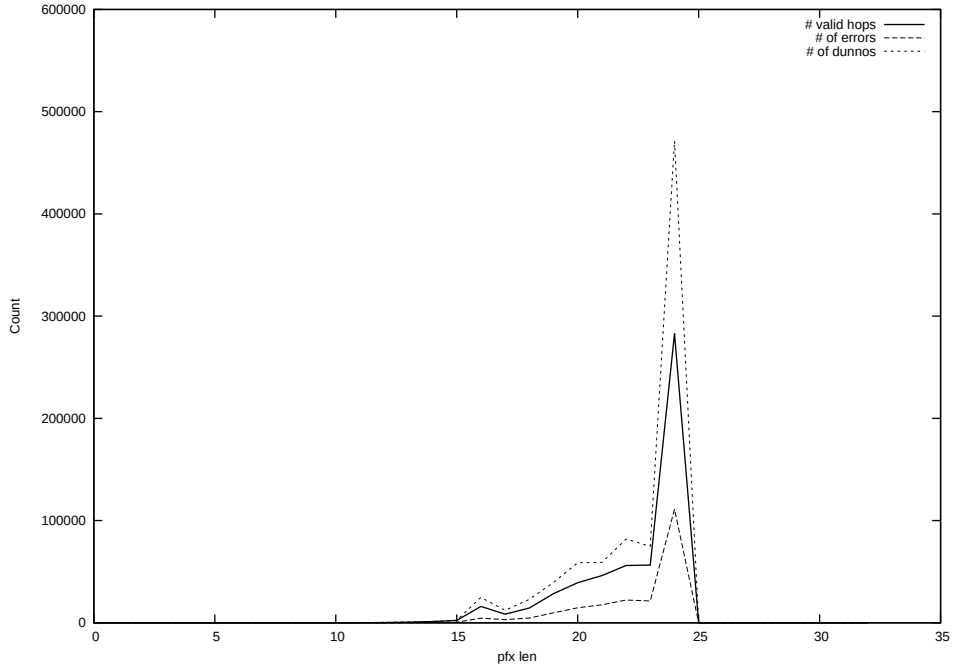
2012-07-16



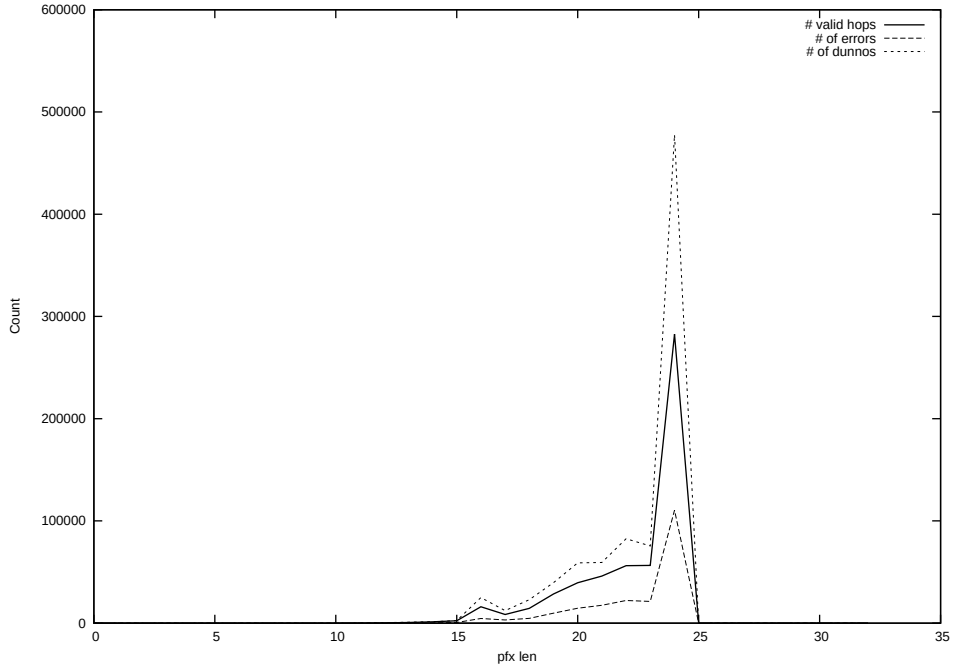
2012-07-17



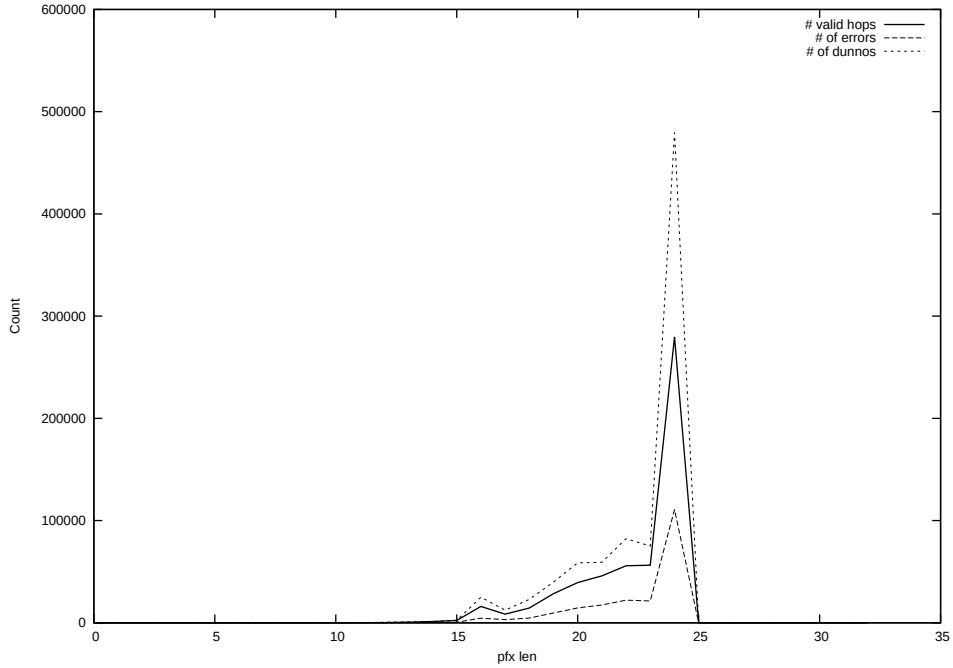
2012-07-18



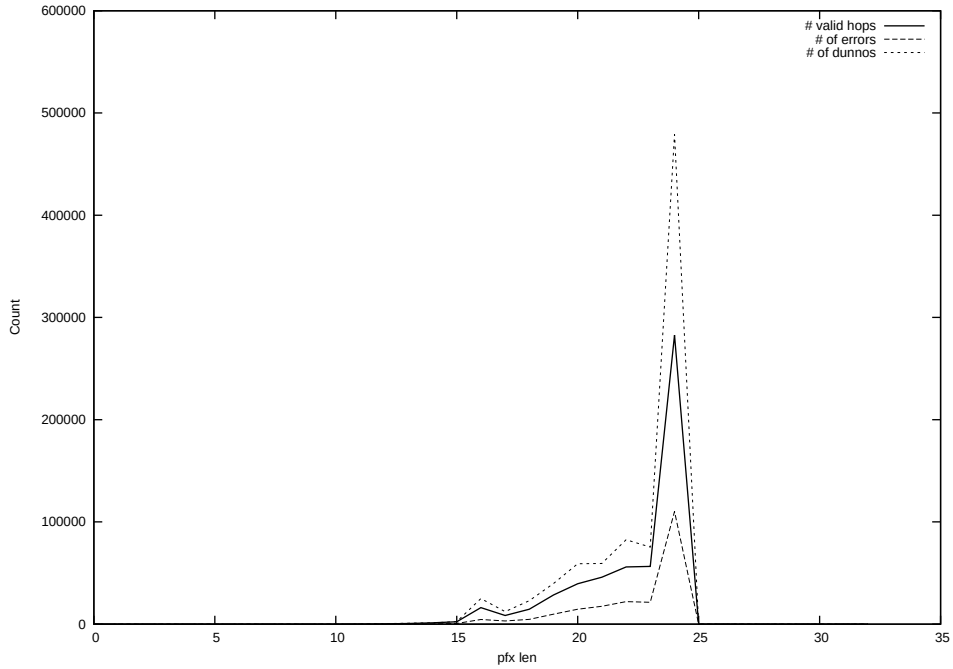
2012-07-19



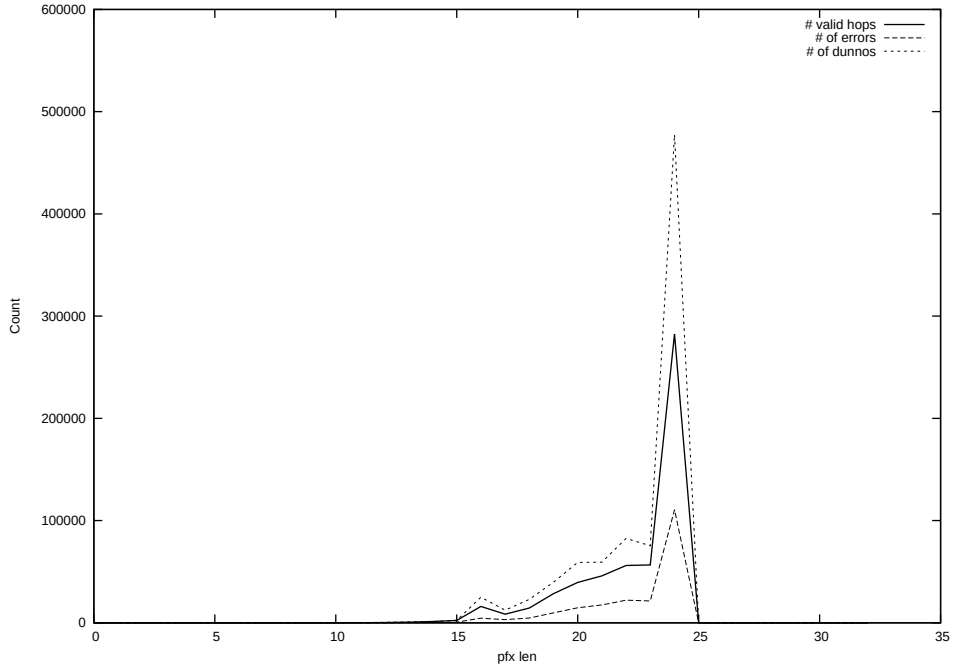
2012-07-20



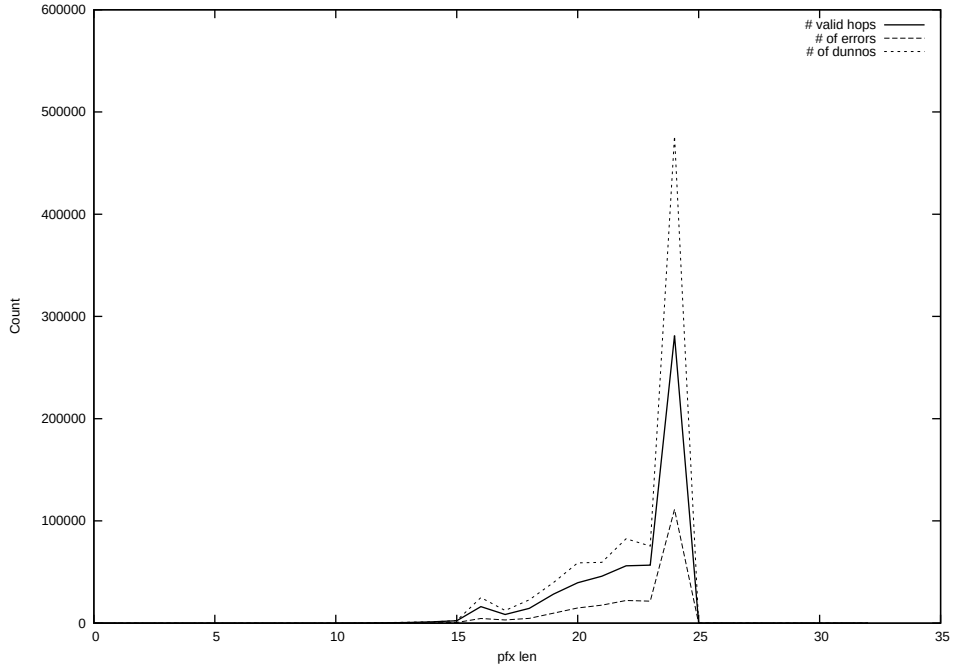
2012-07-21



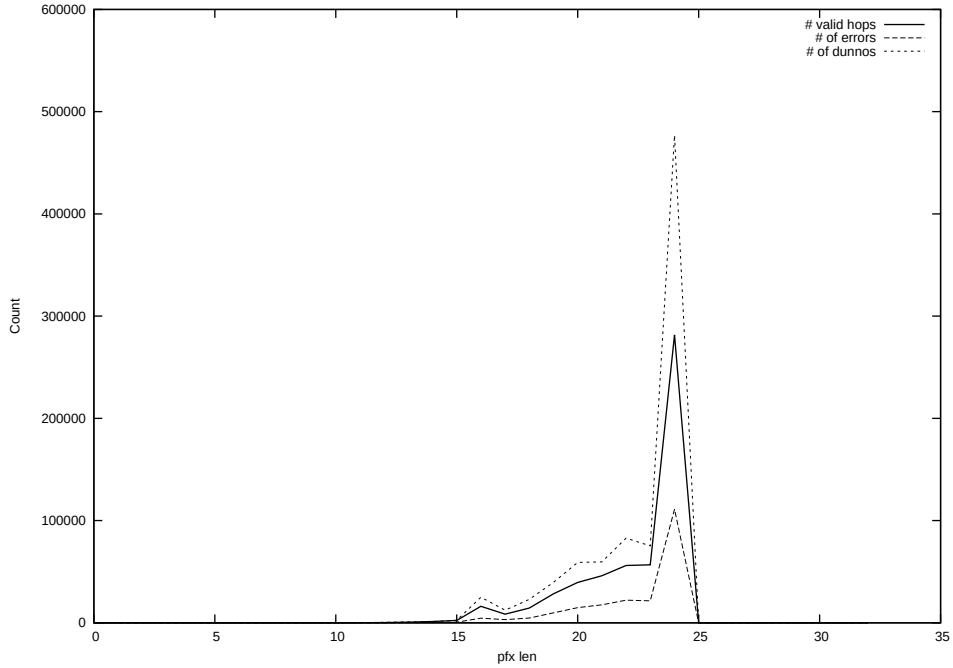
2012-07-22



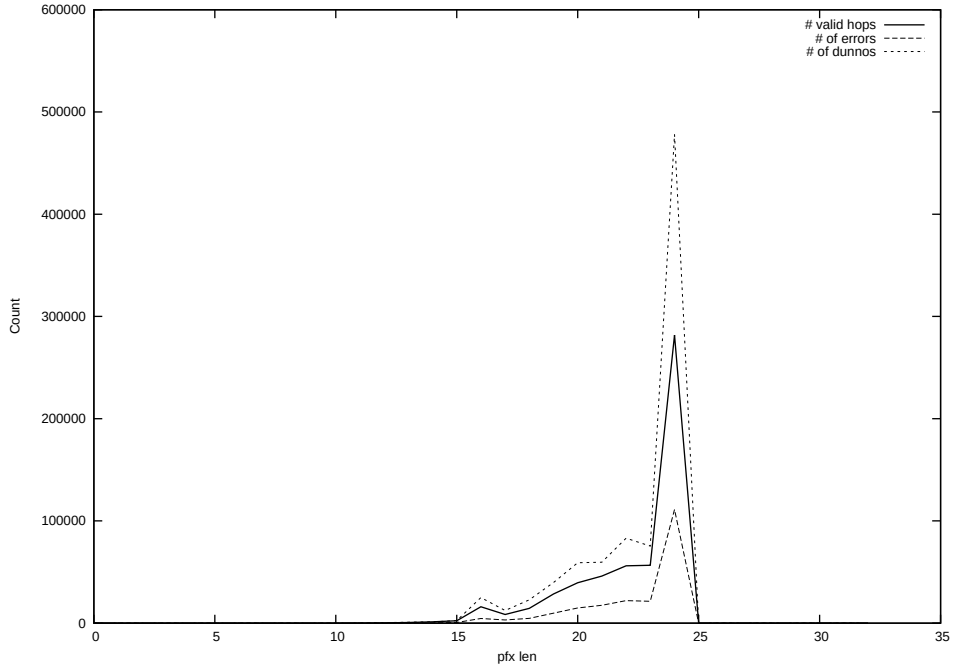
2012-07-23



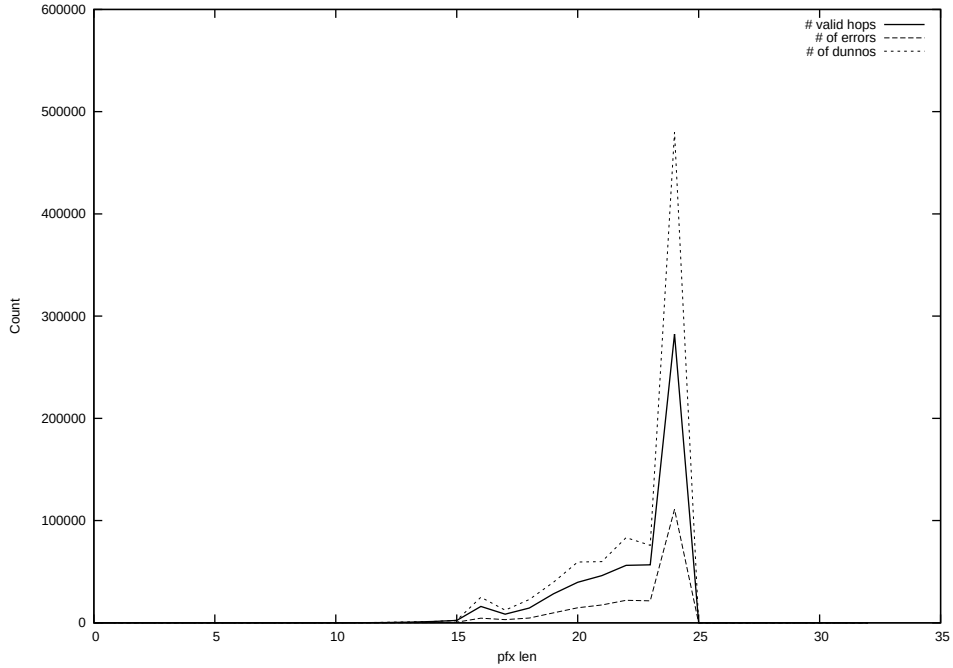
2012-07-24



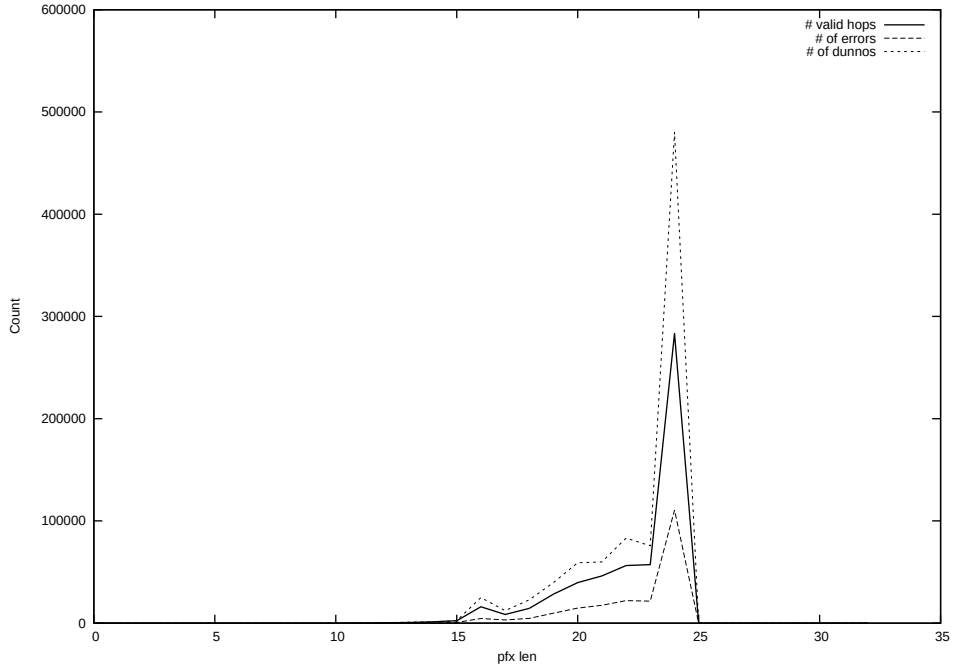
2012-07-25



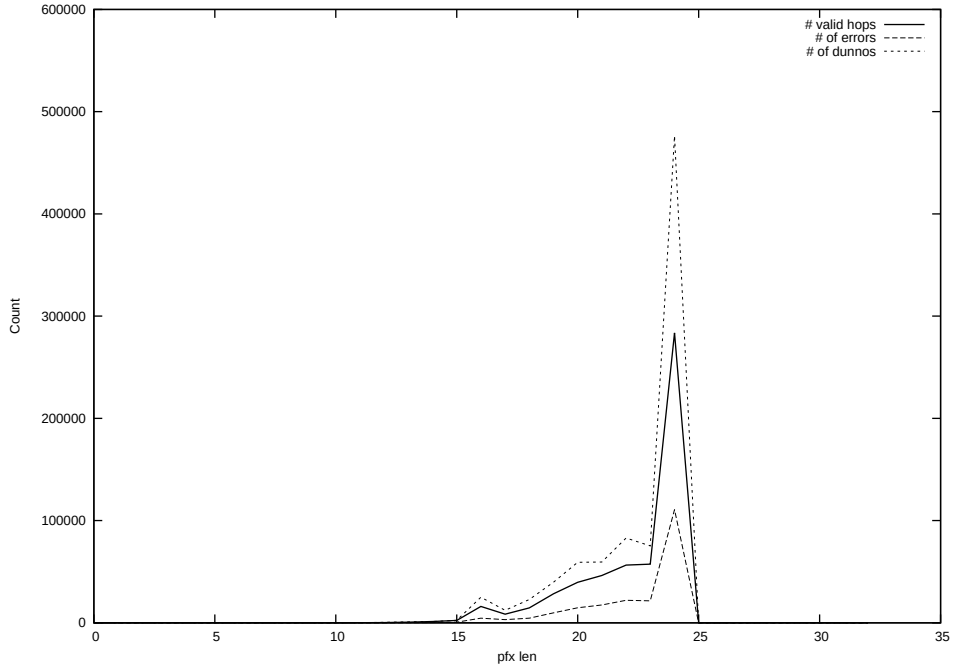
2012-07-26



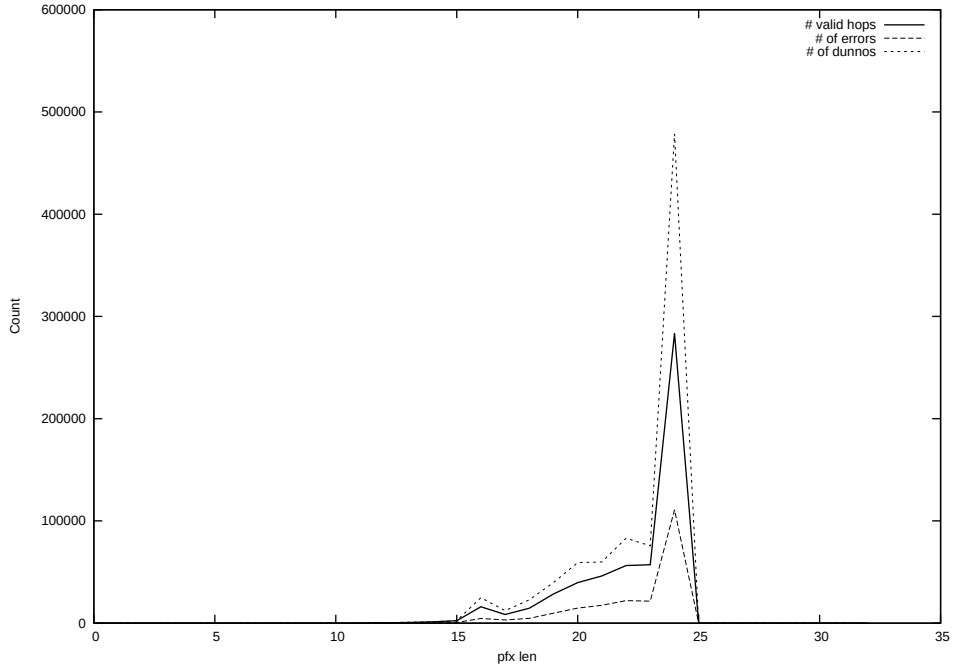
2012-07-27



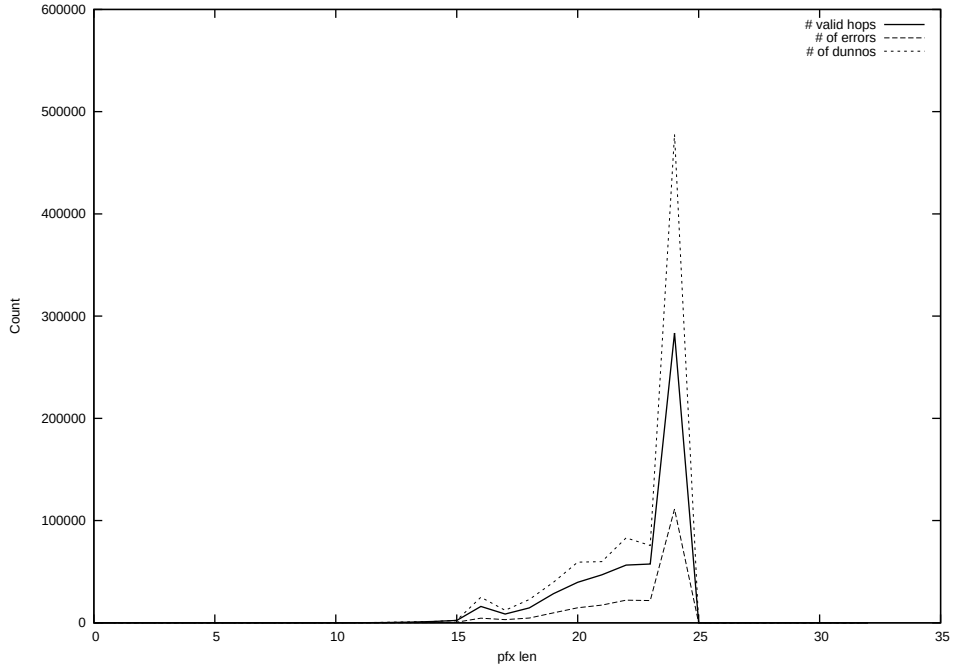
2012-07-28



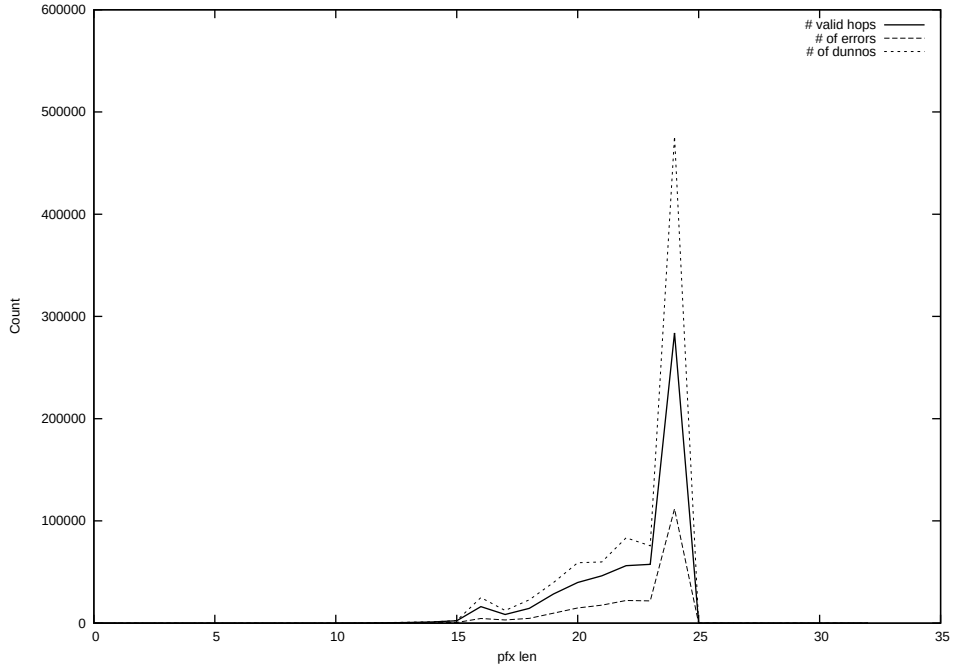
2012-07-29



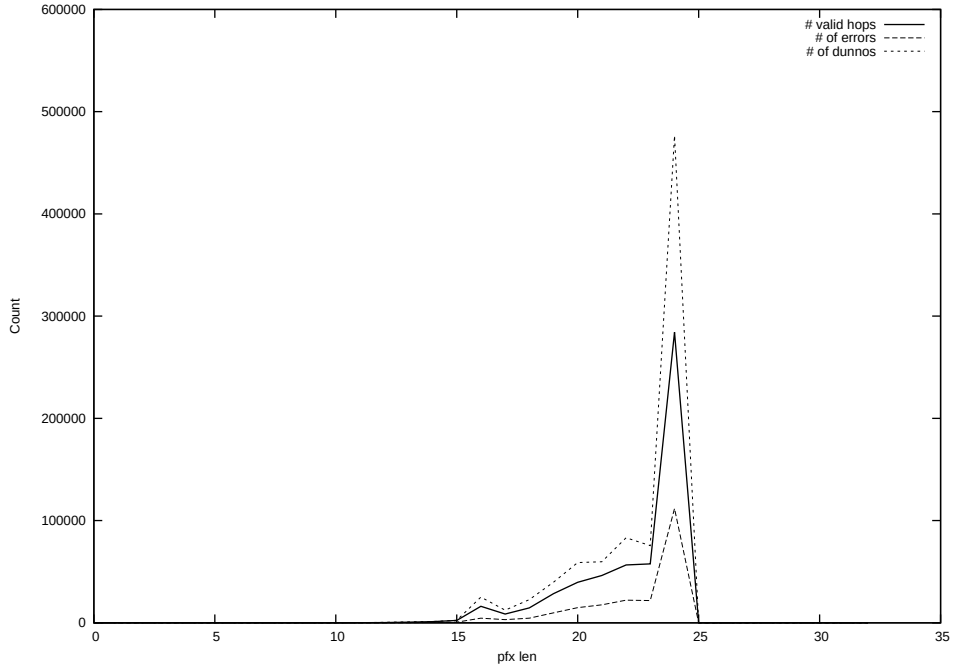
2012-07-30



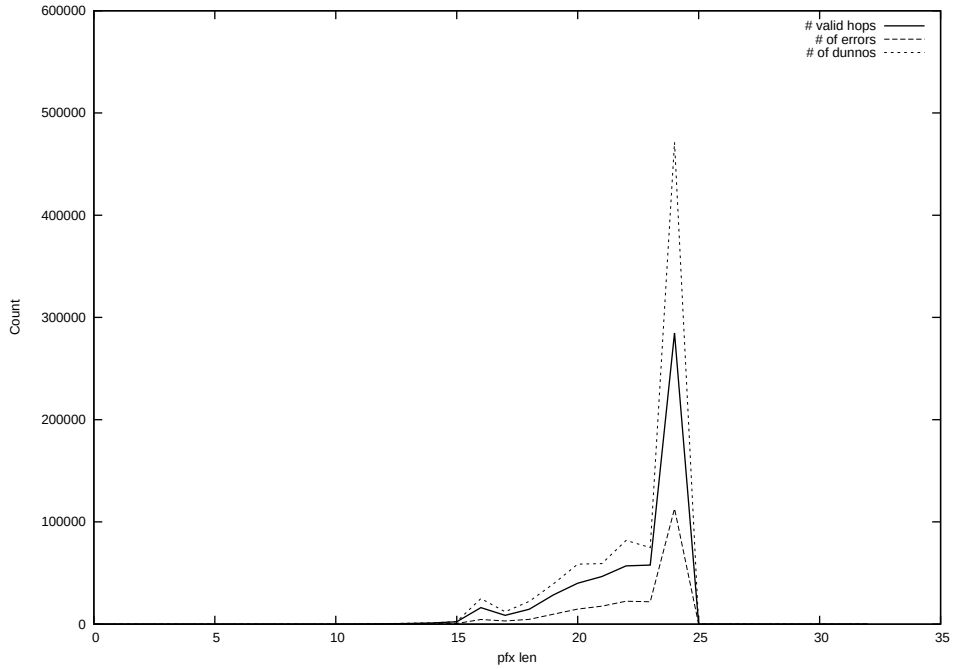
2012-07-31



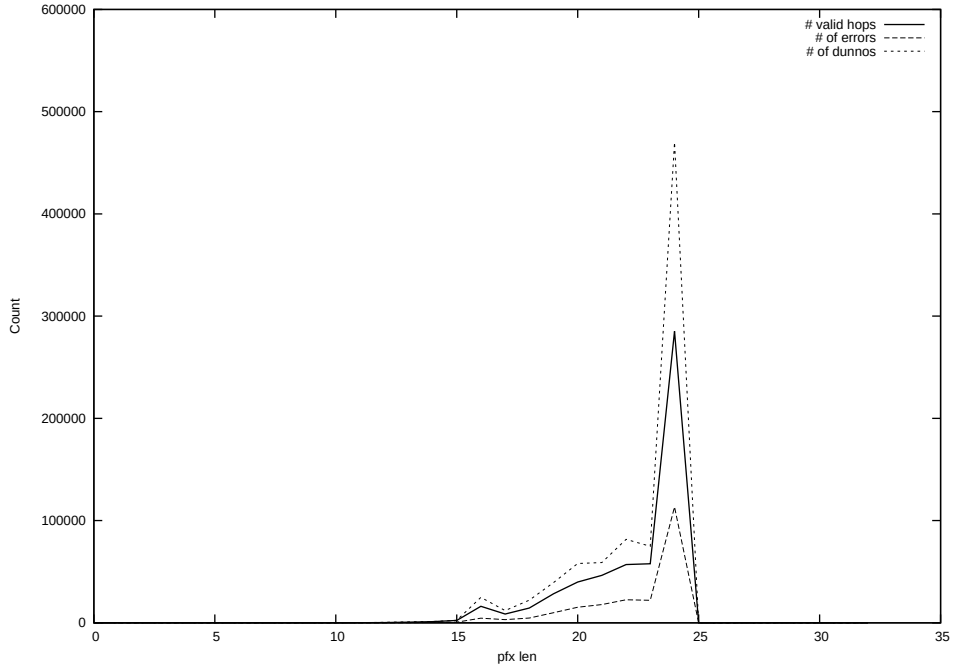
2012-08-01



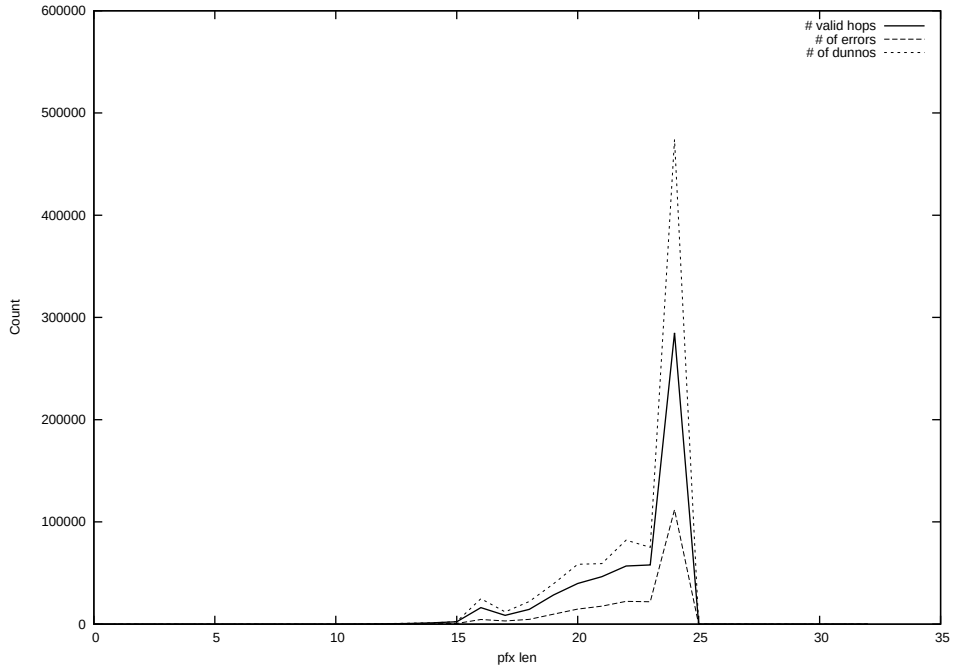
2012-08-02



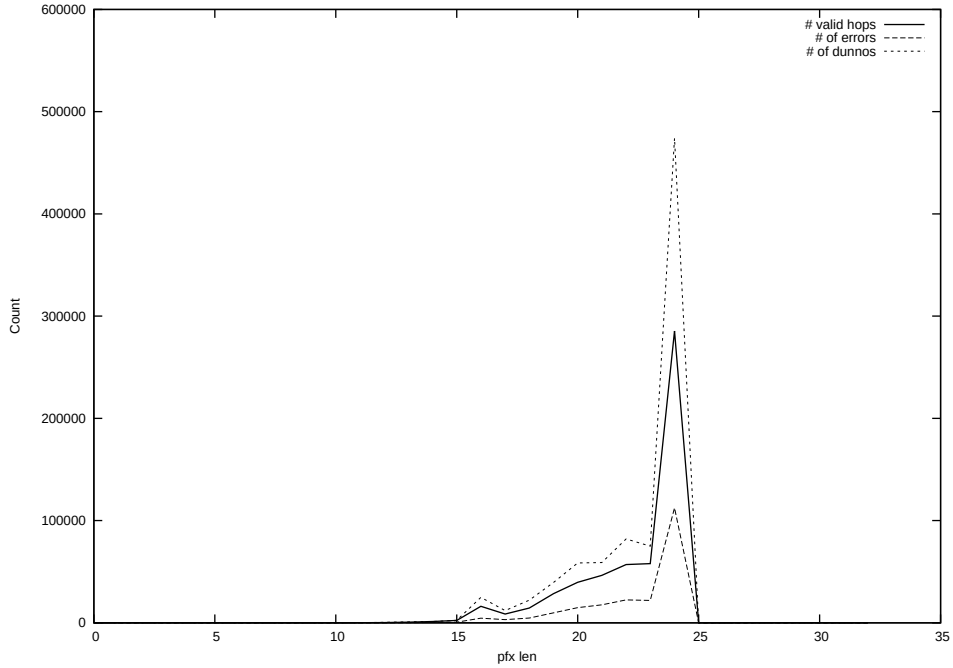
2012-08-03



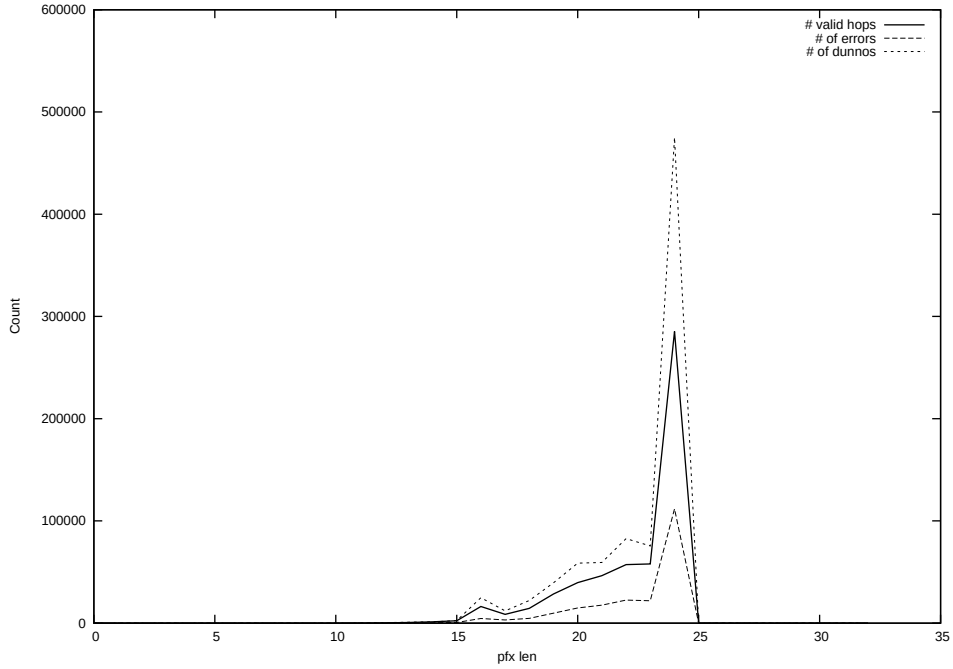
2012-08-04



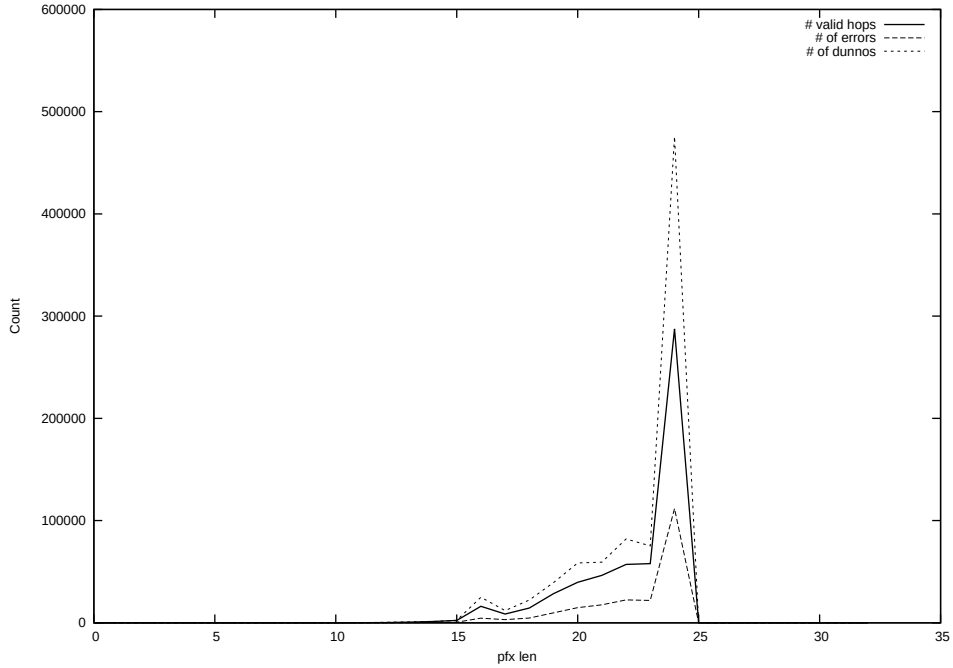
2012-08-05



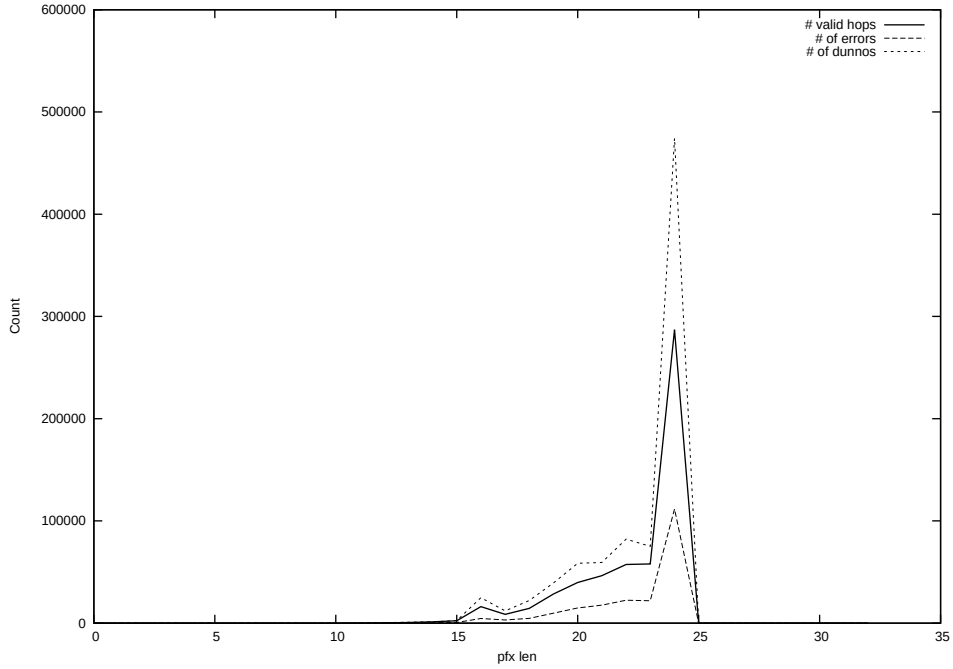
2012-08-06



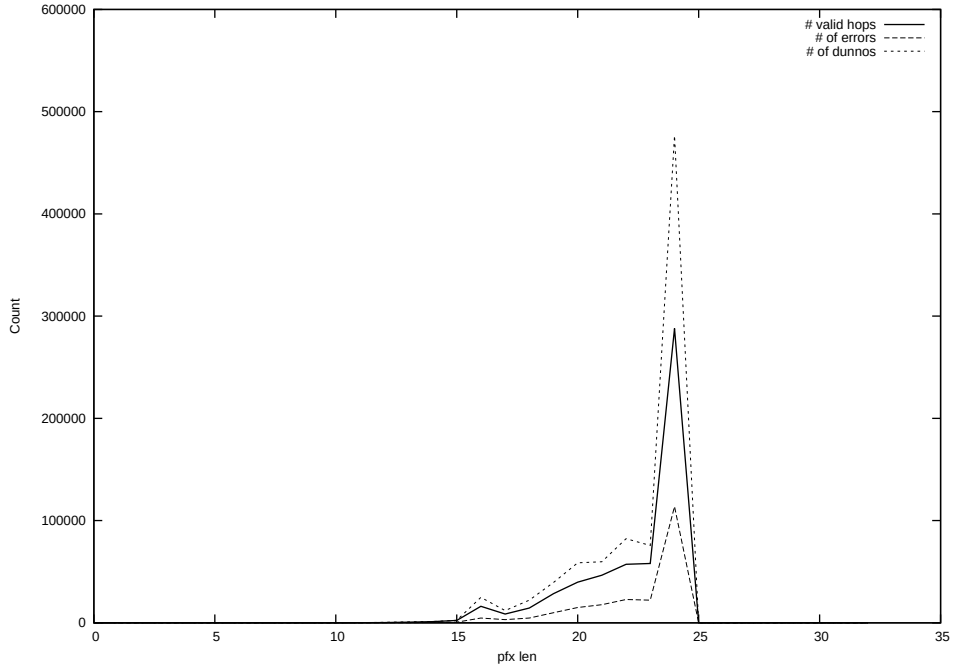
2012-08-07



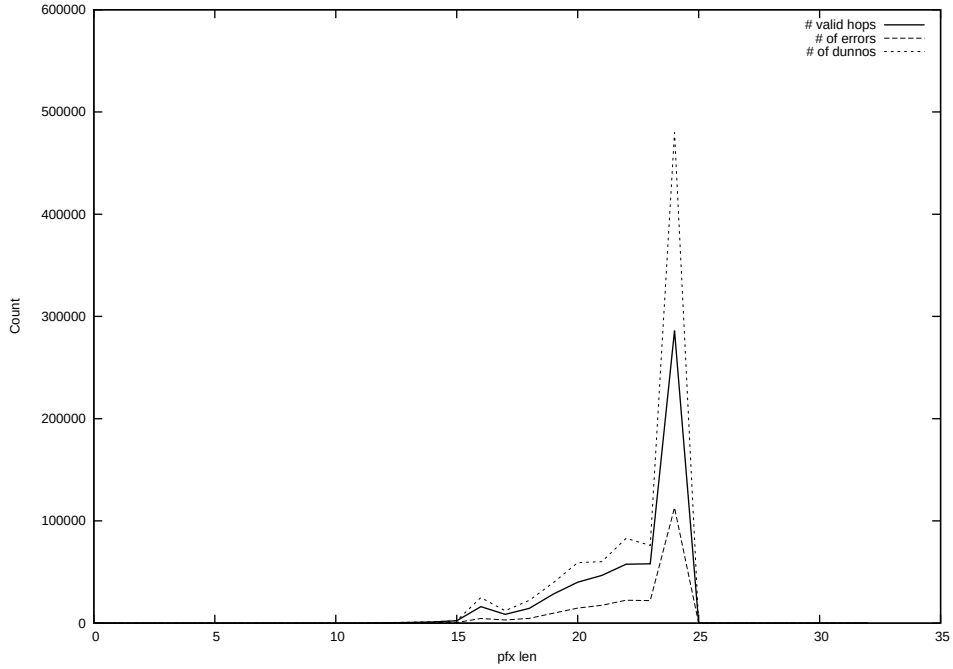
2012-08-08



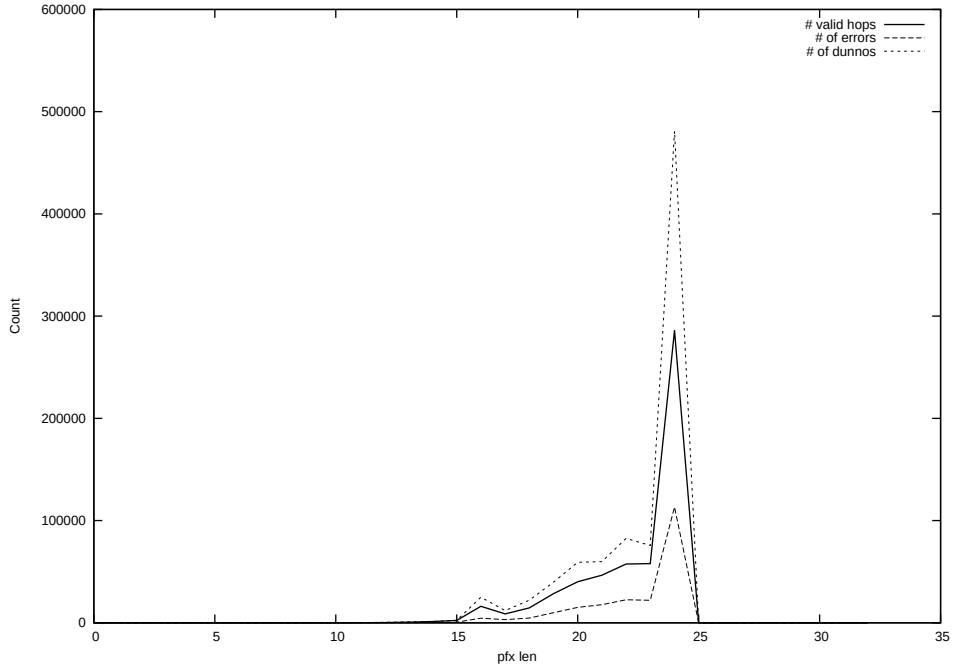
2012-08-09



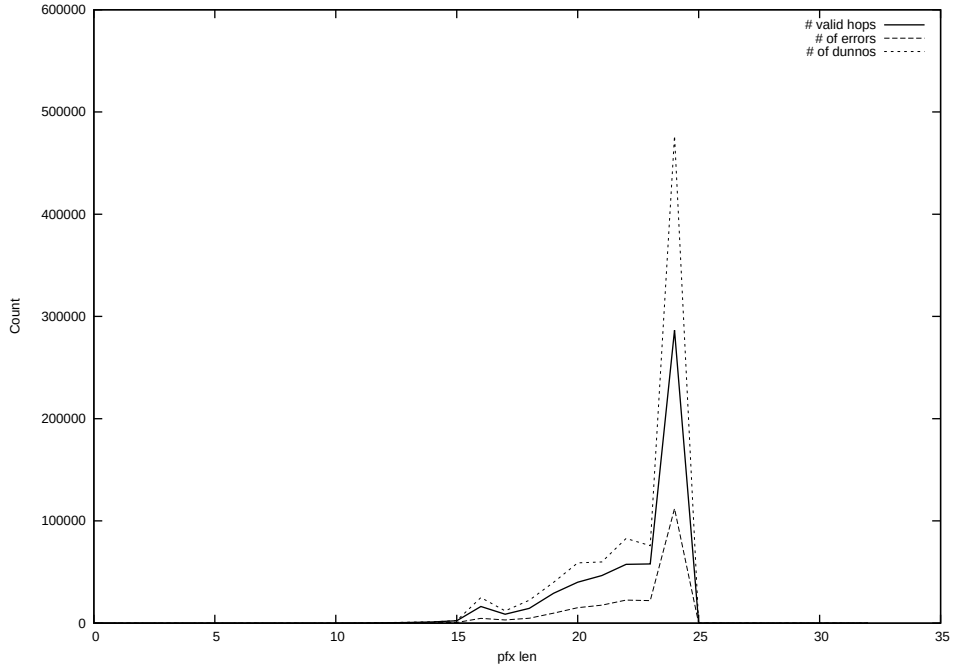
2012-08-10



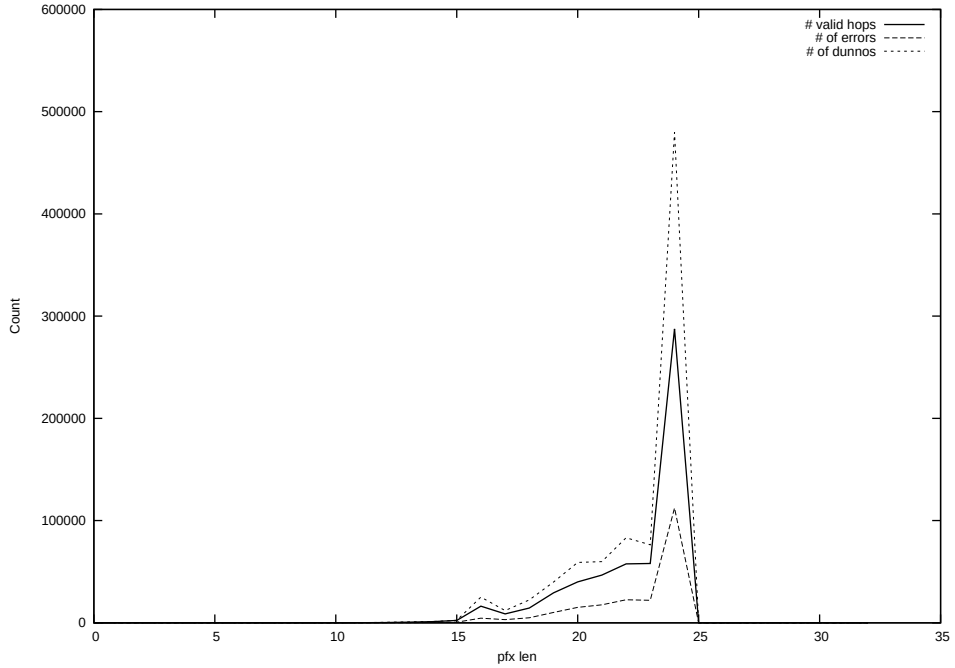
2012-08-11



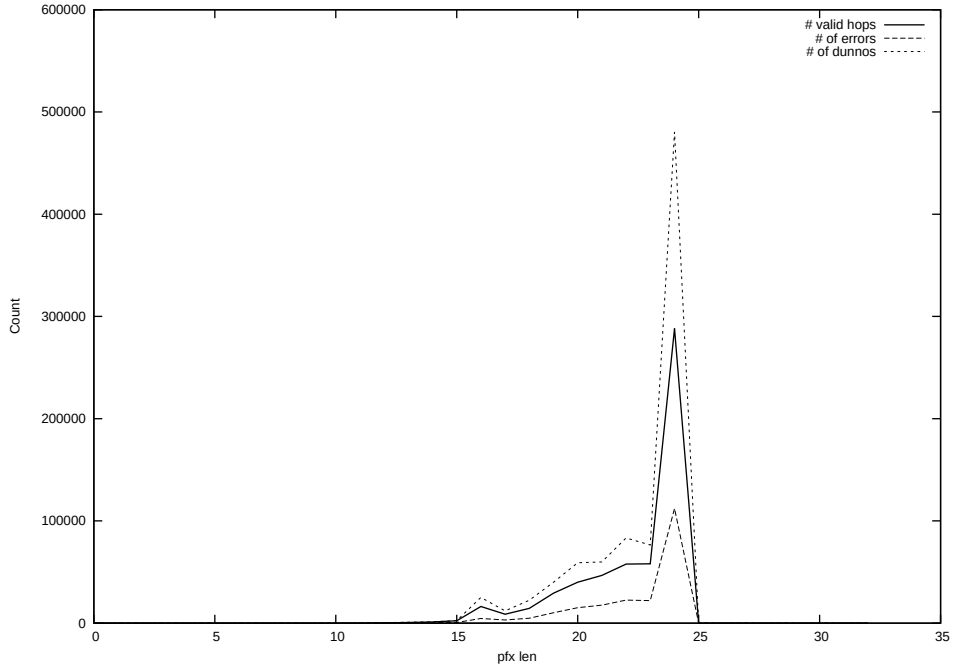
2012-08-12



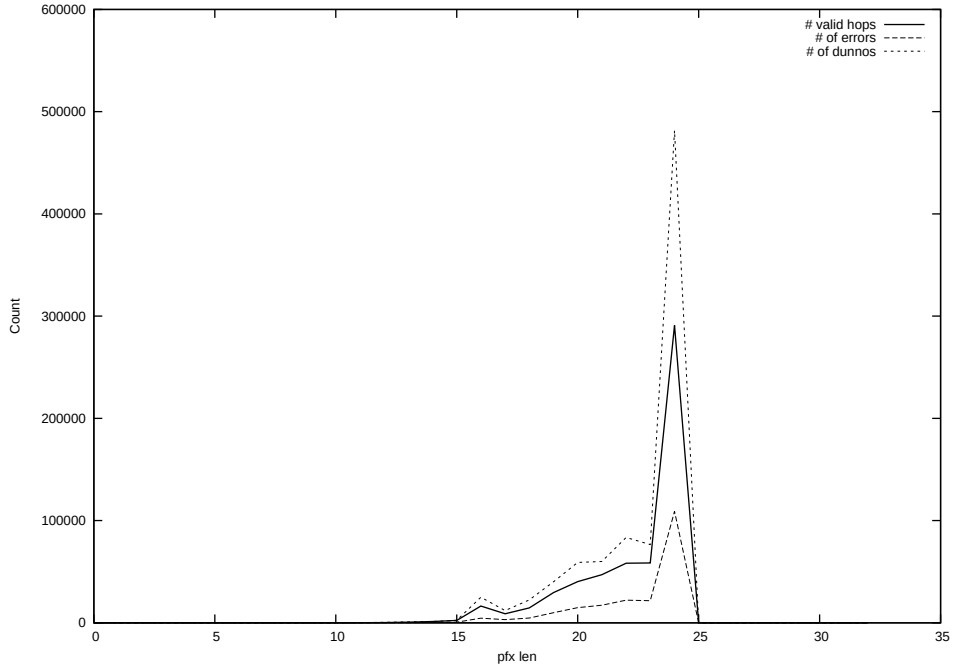
2012-08-13



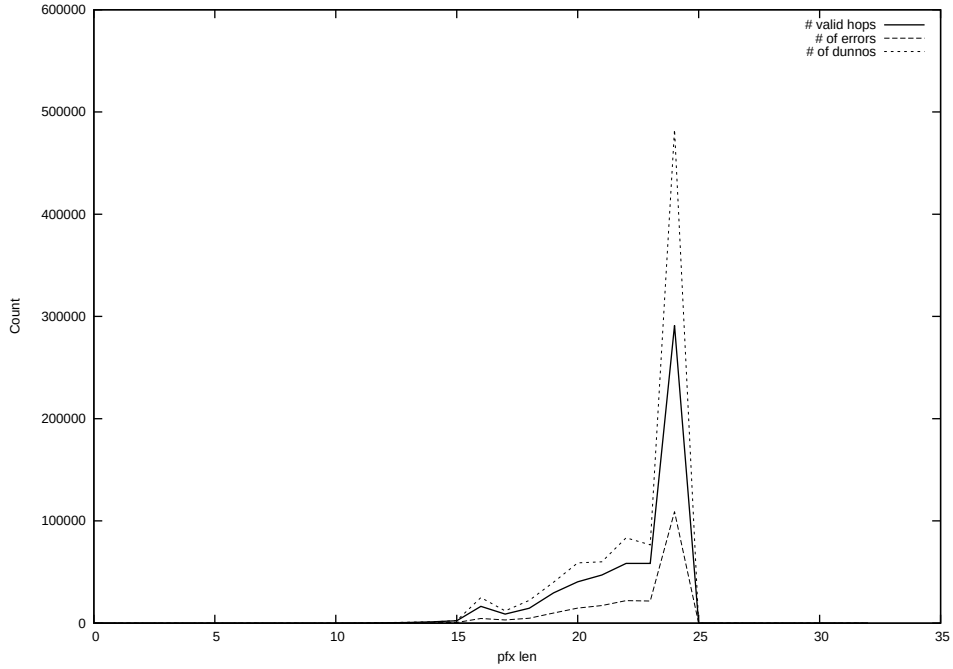
2012-08-14



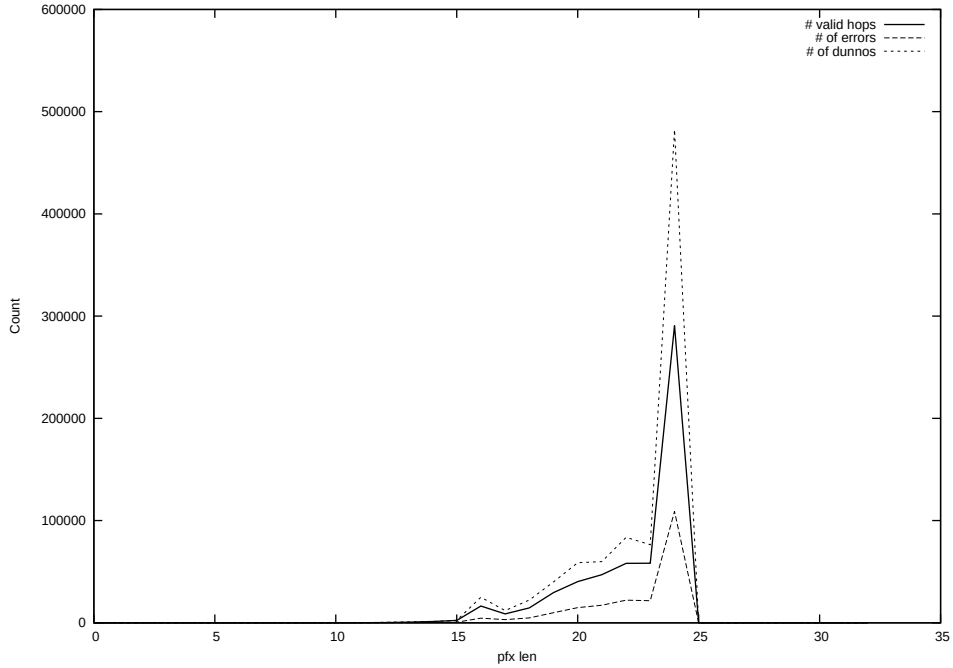
2012-08-15



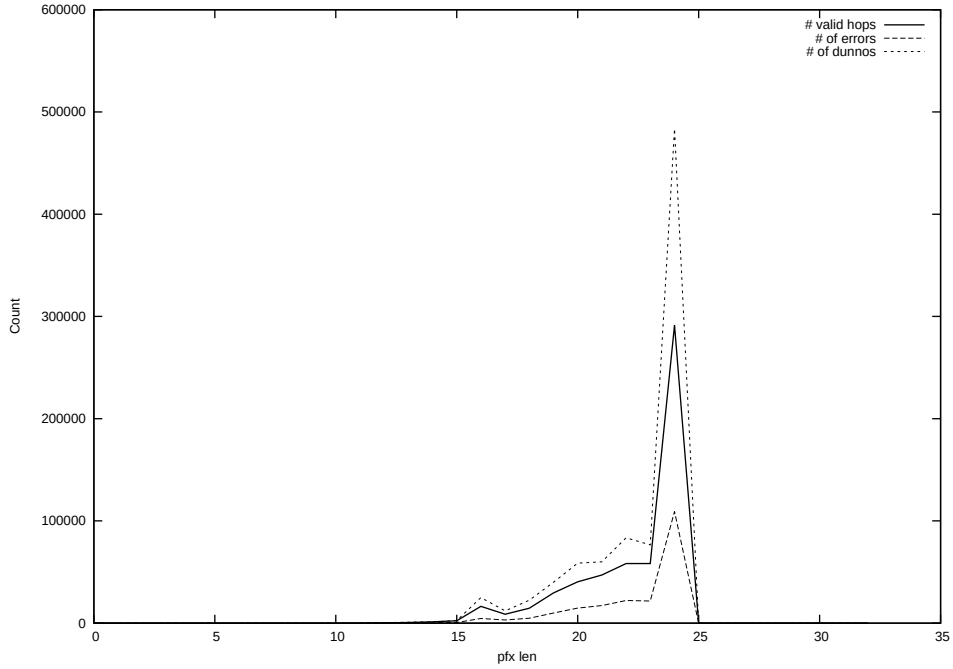
2012-08-16



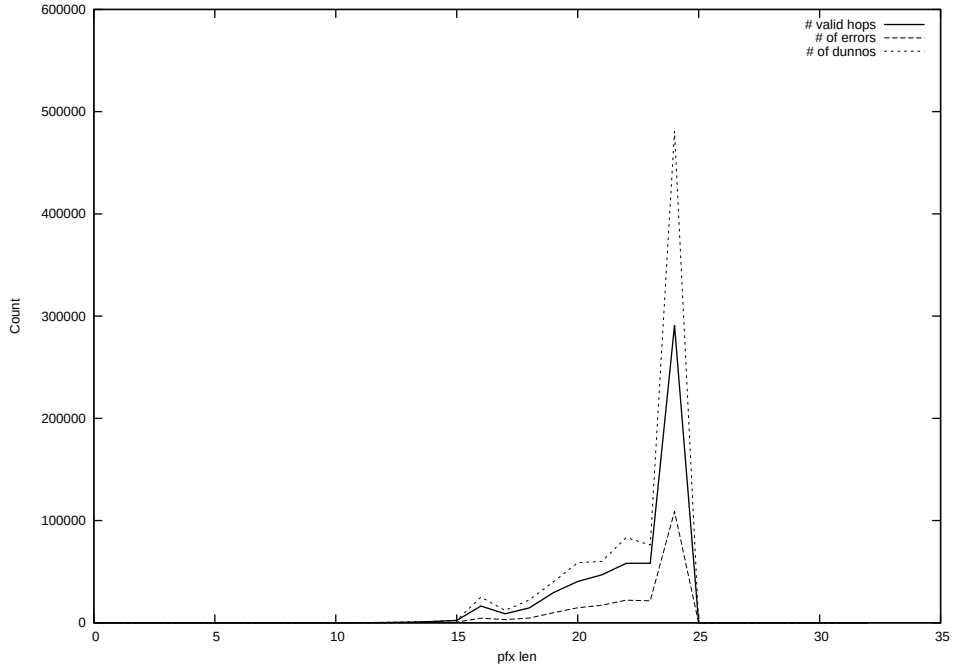
2012-08-17



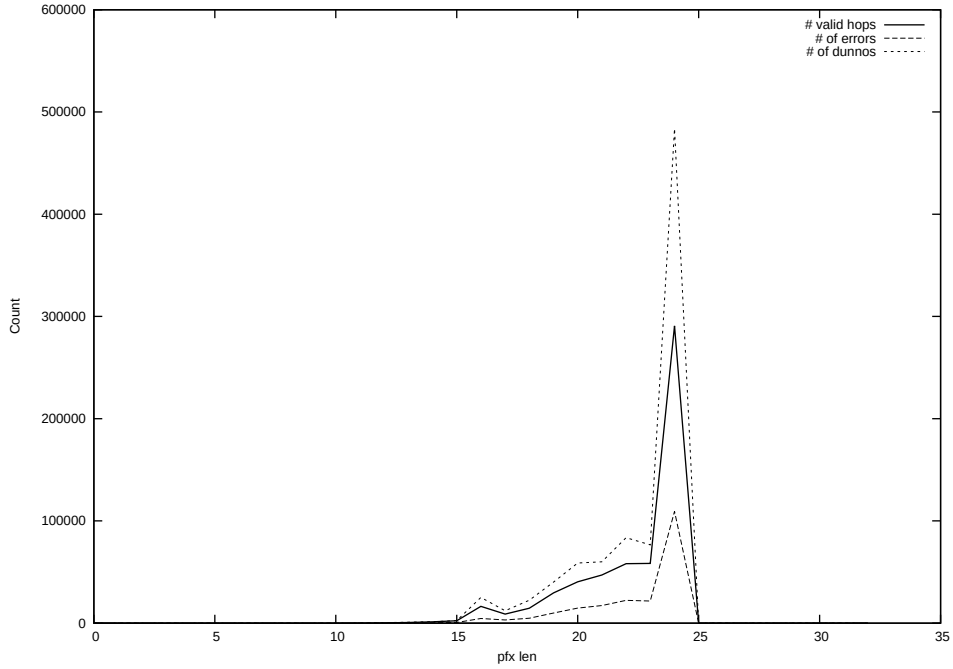
2012-08-18



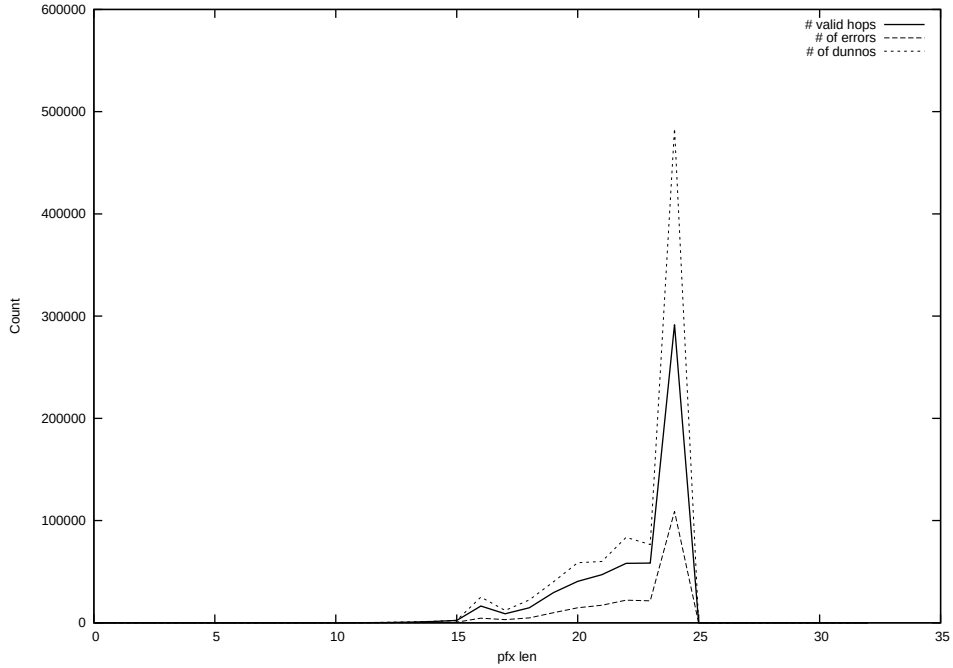
2012-08-19



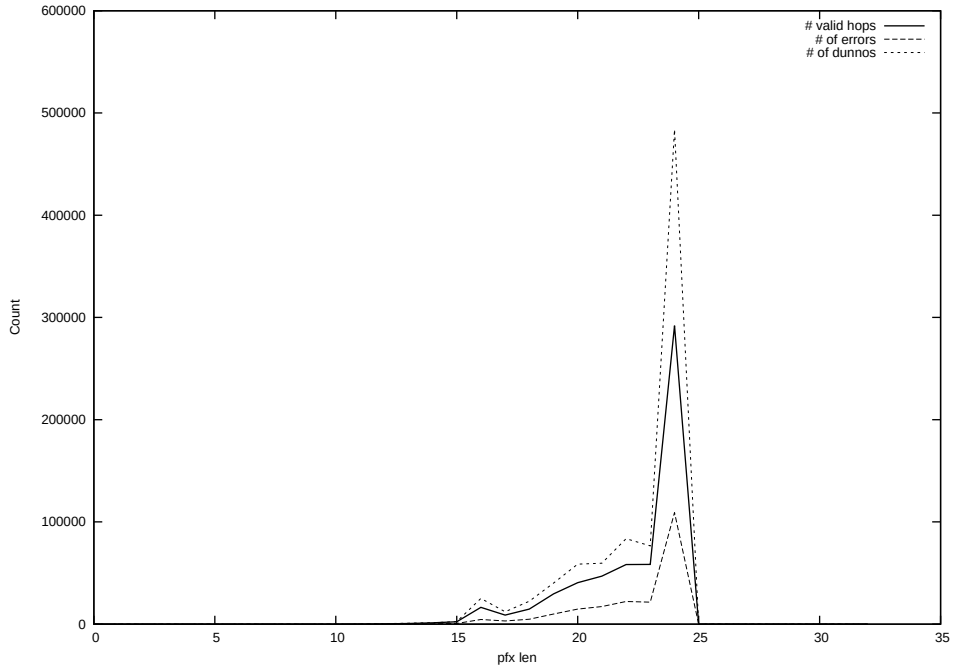
2012-08-20



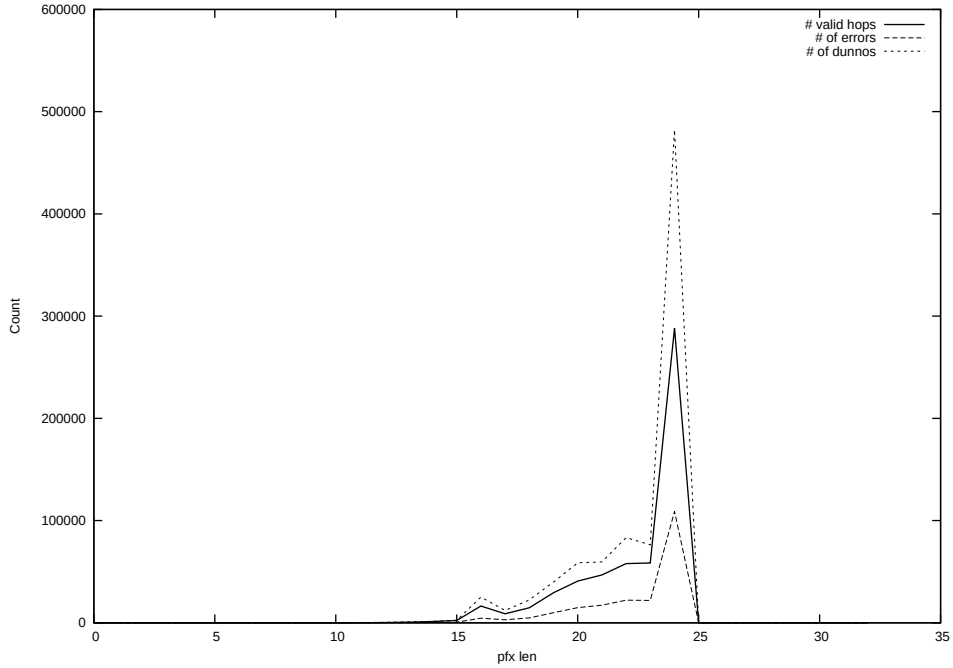
2012-08-21



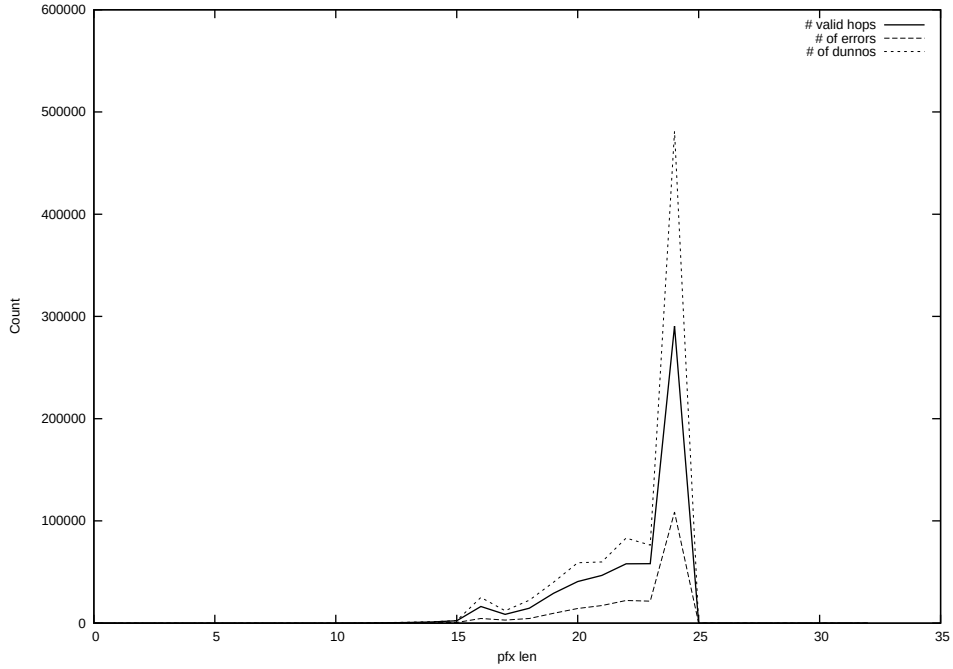
2012-08-22



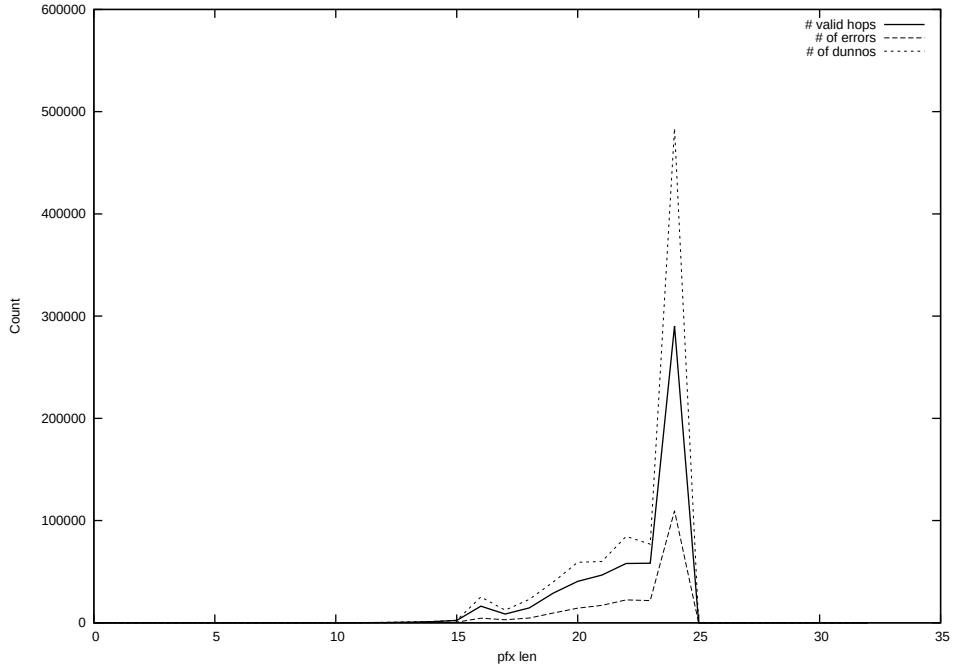
2012-08-23



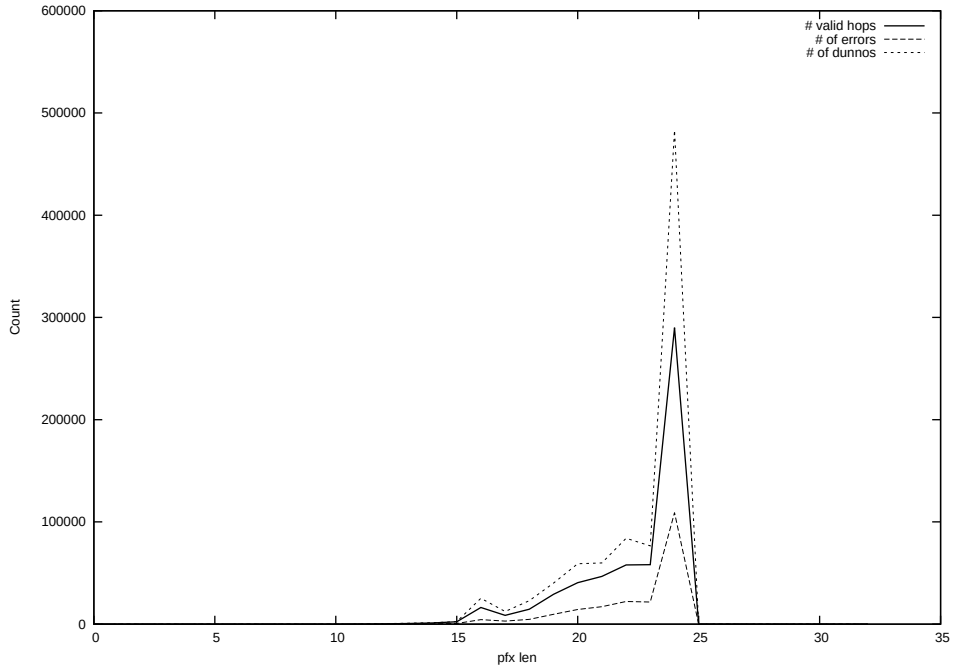
2012-08-24



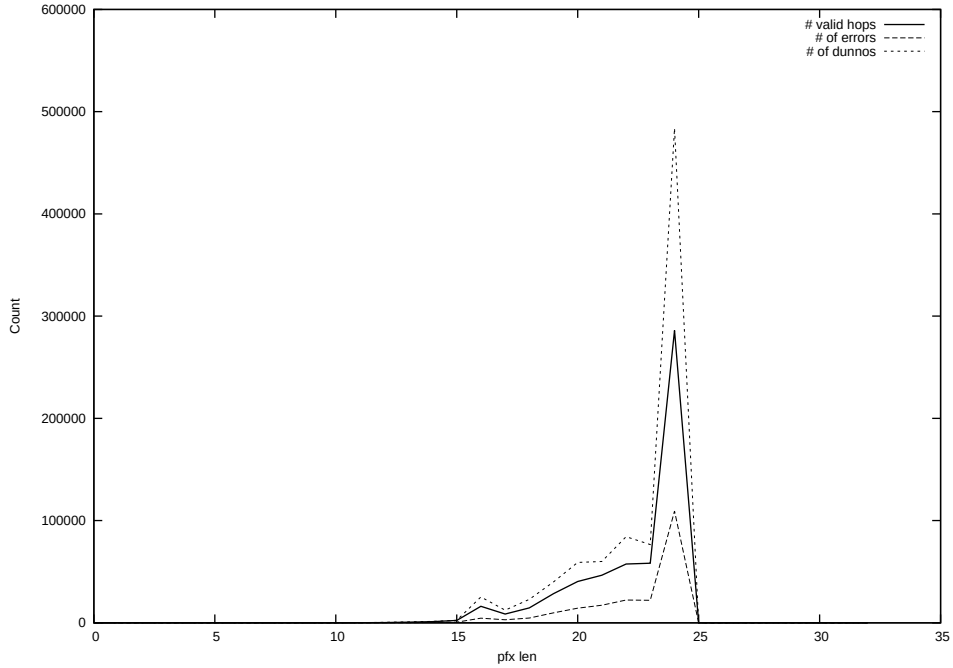
2012-08-25



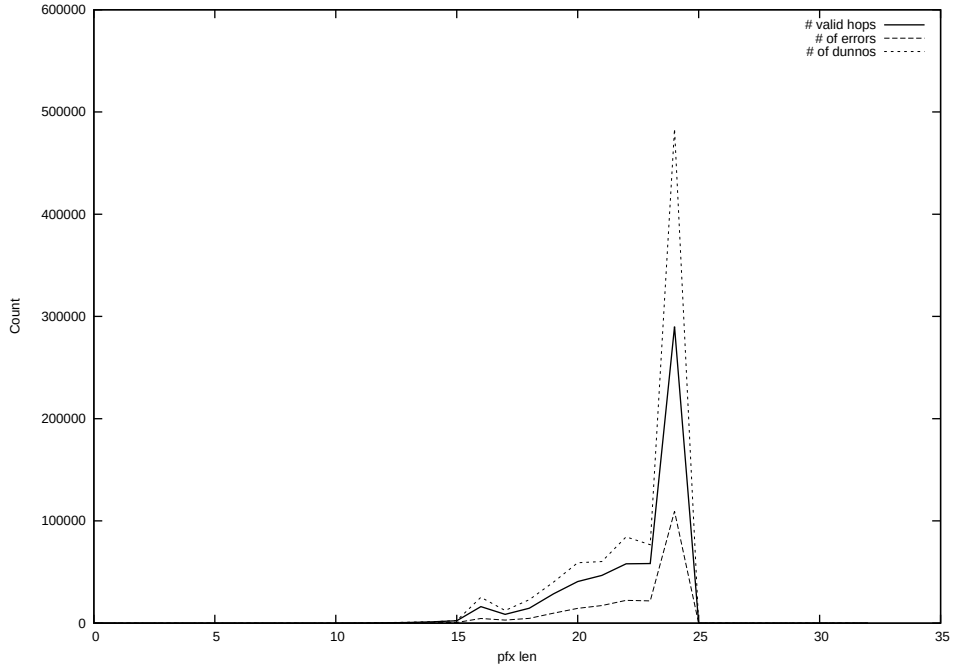
2012-08-26



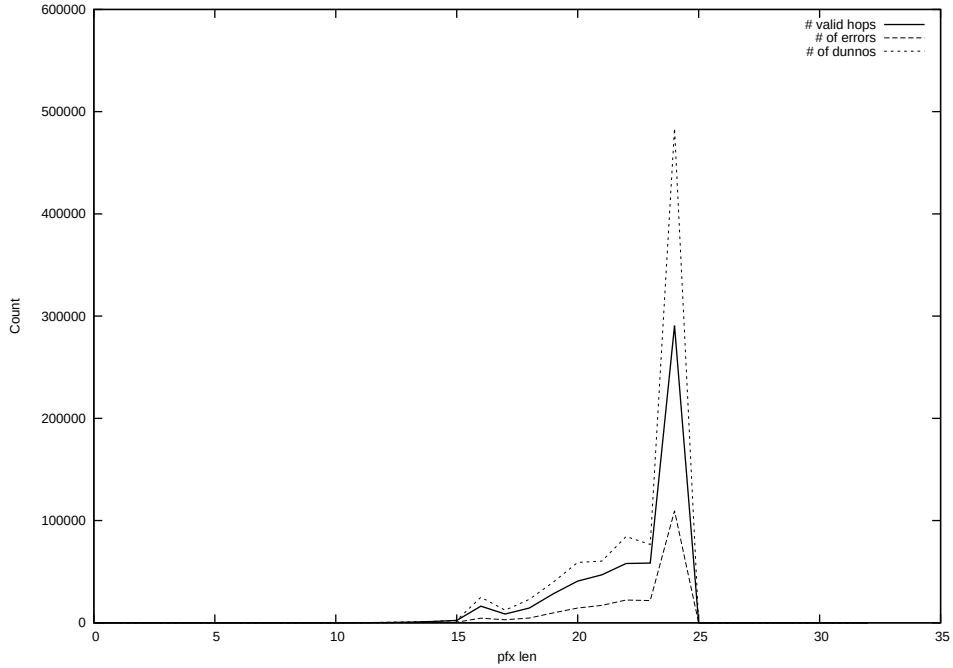
2012-08-27



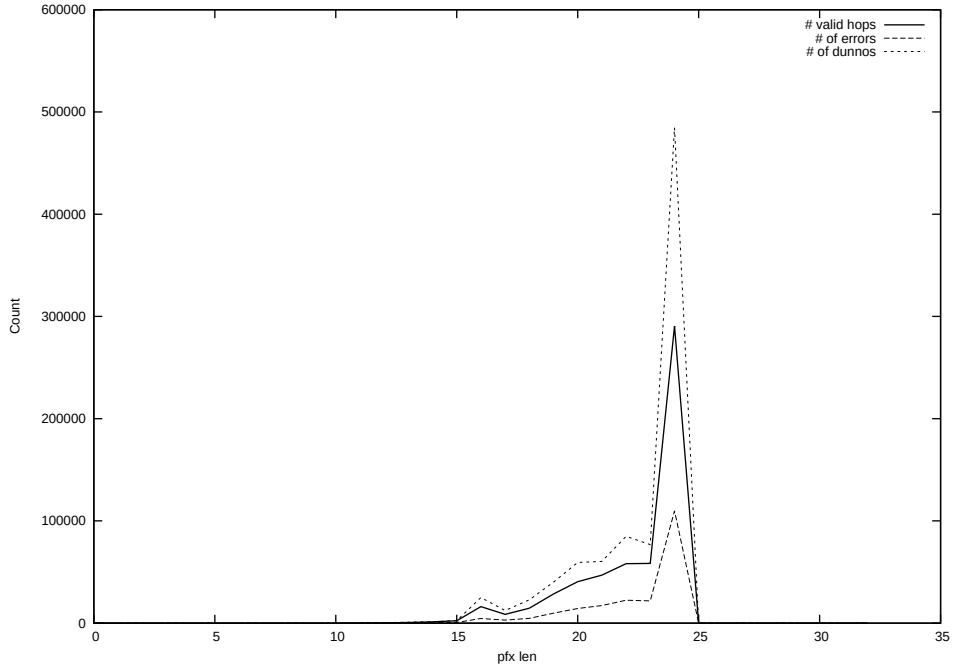
2012-08-28



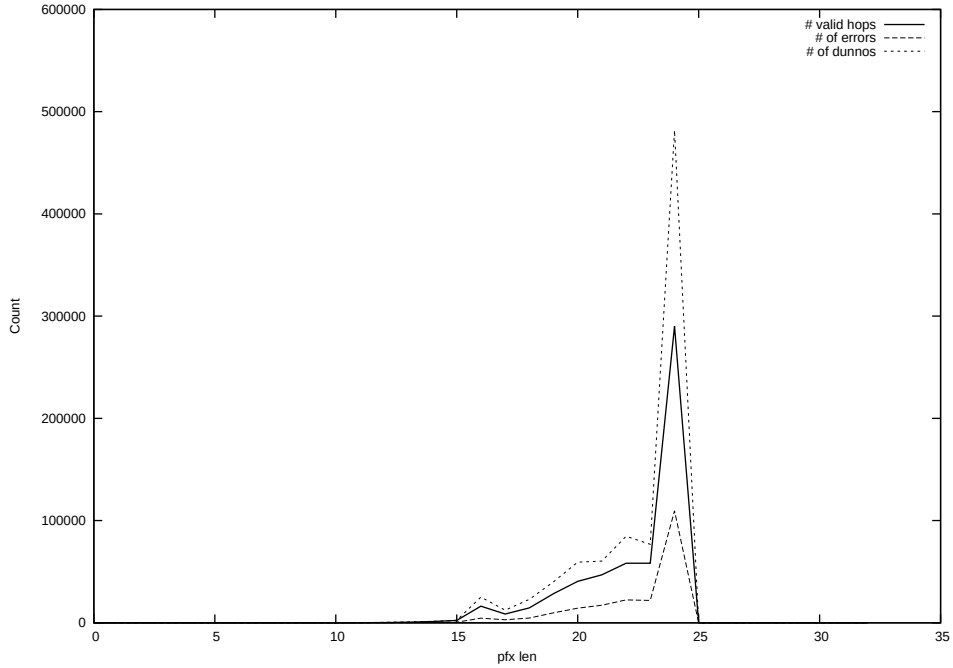
2012-08-29



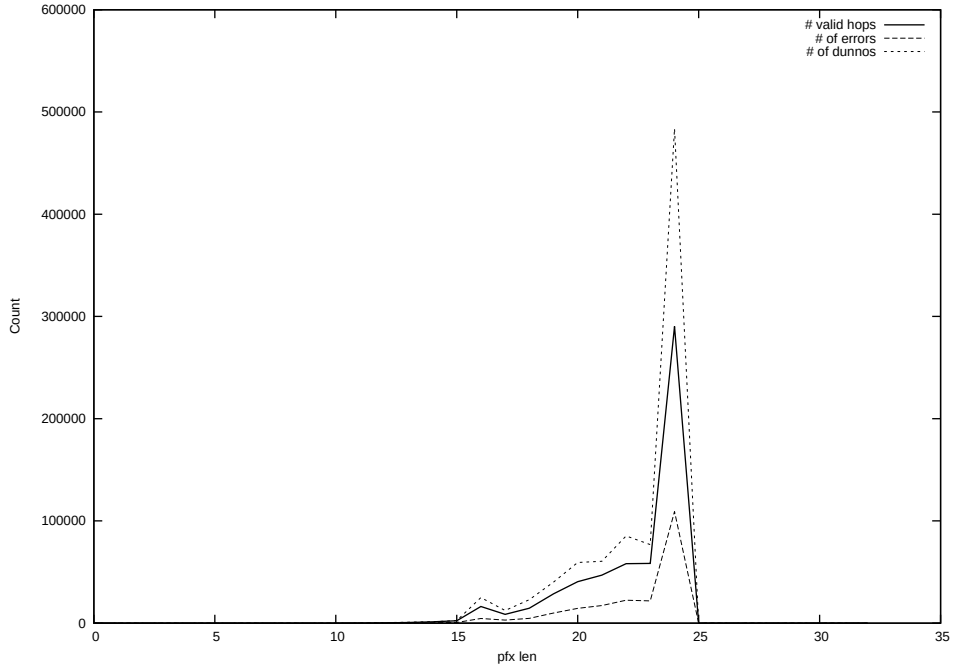
2012-08-30



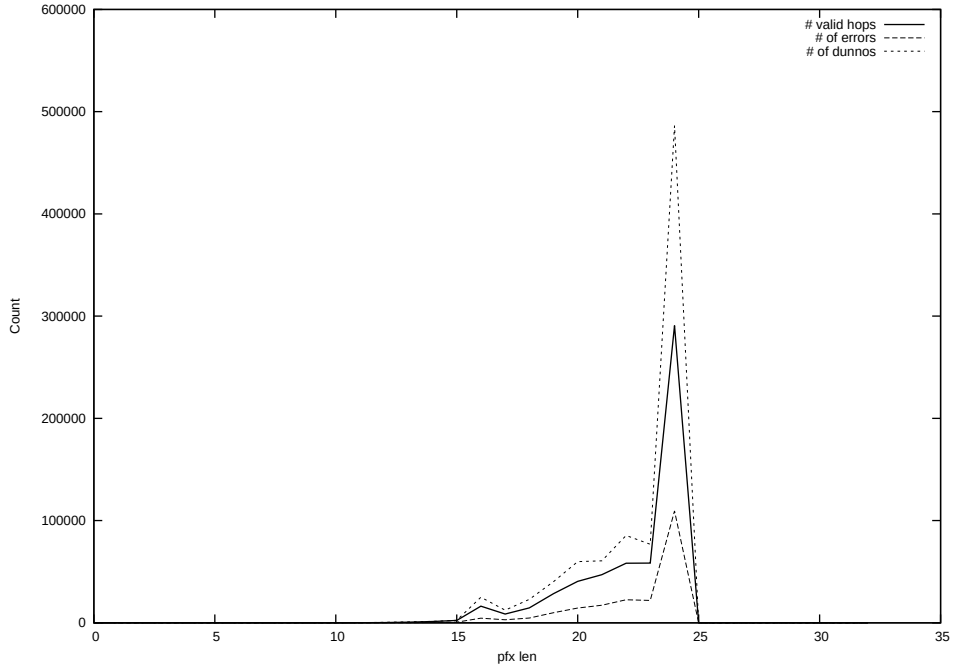
2012-08-31



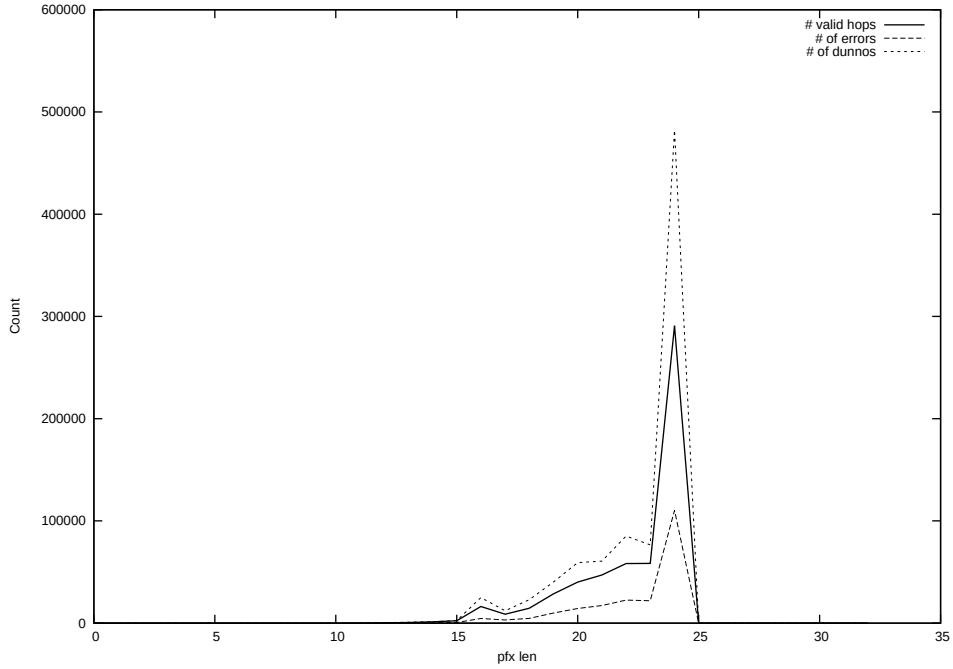
2012-09-01



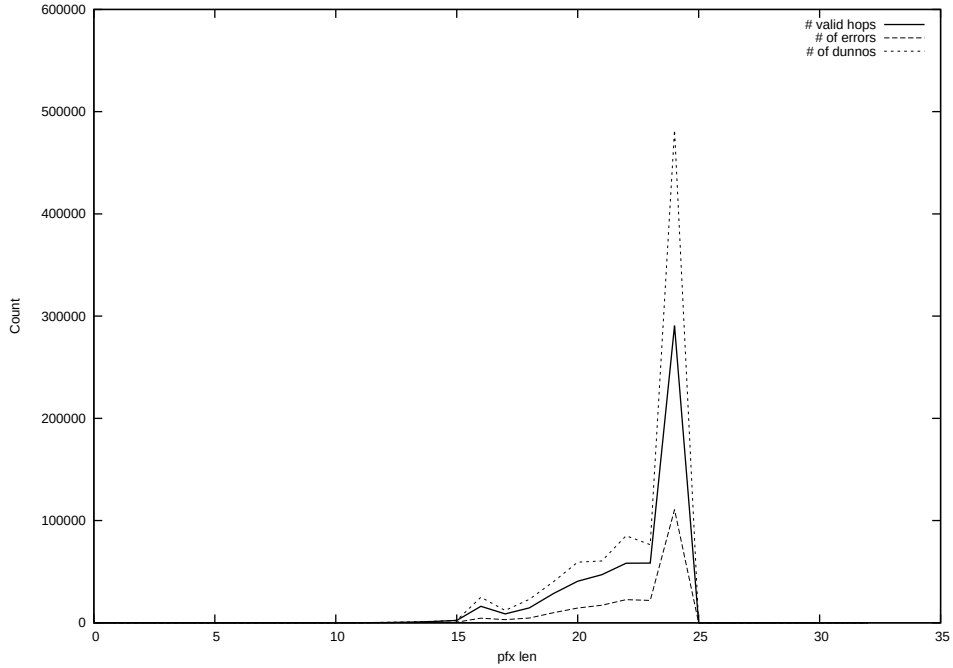
2012-09-02



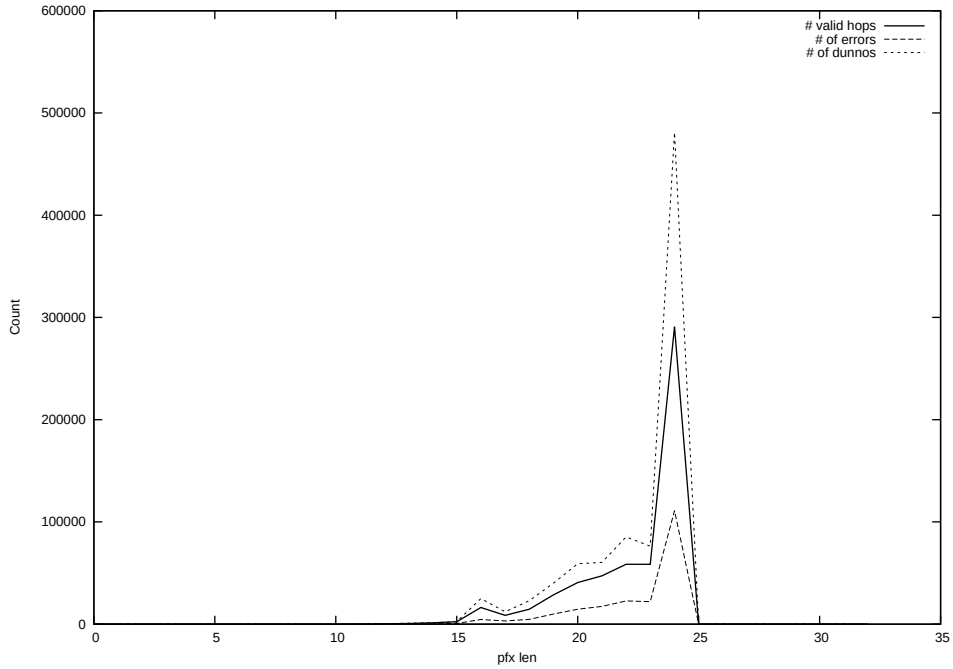
2012-09-03



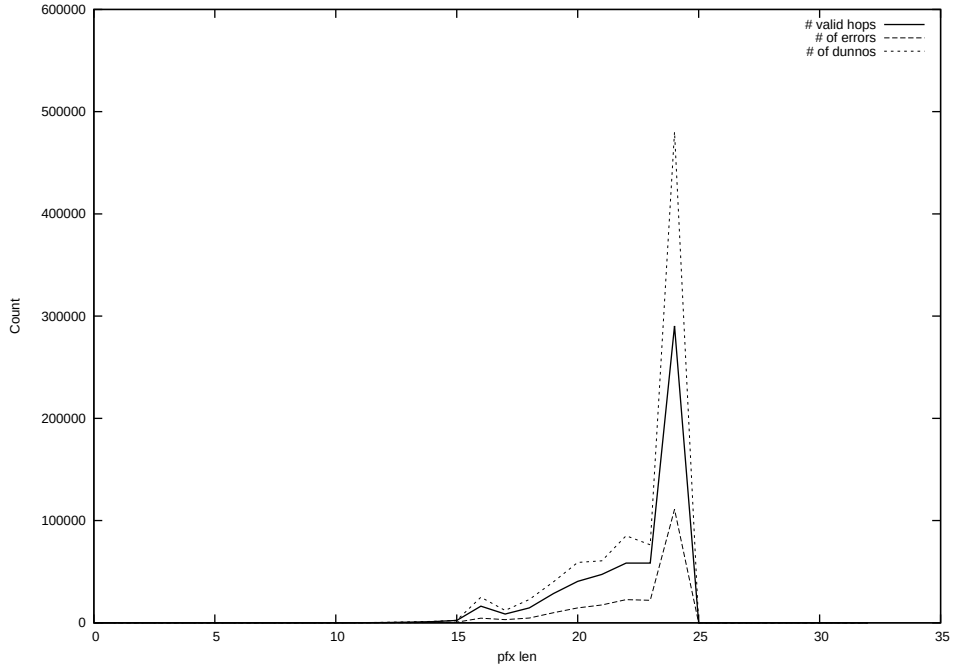
2012-09-04



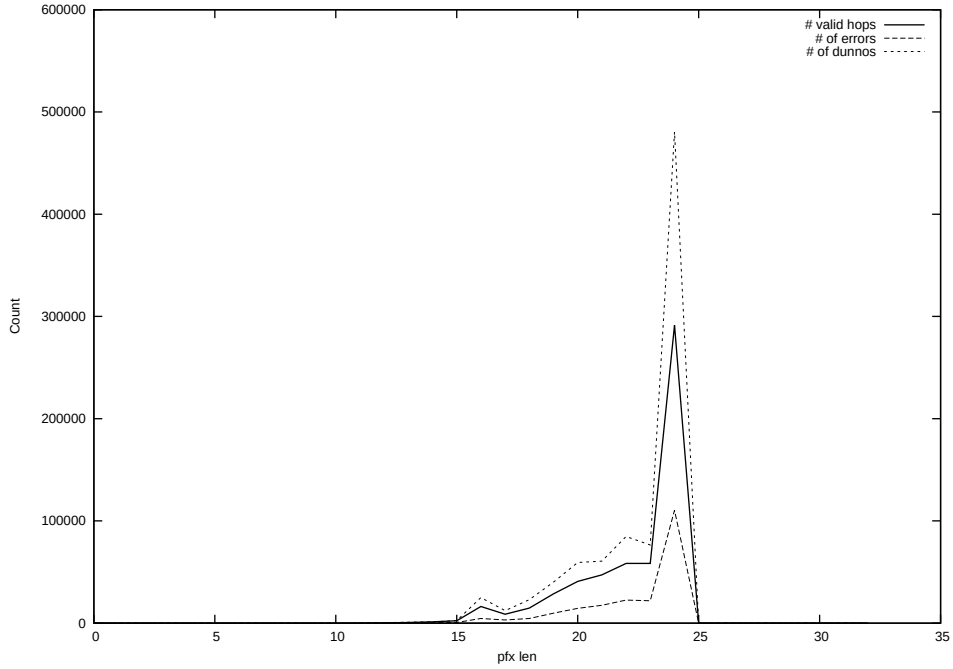
2012-09-05



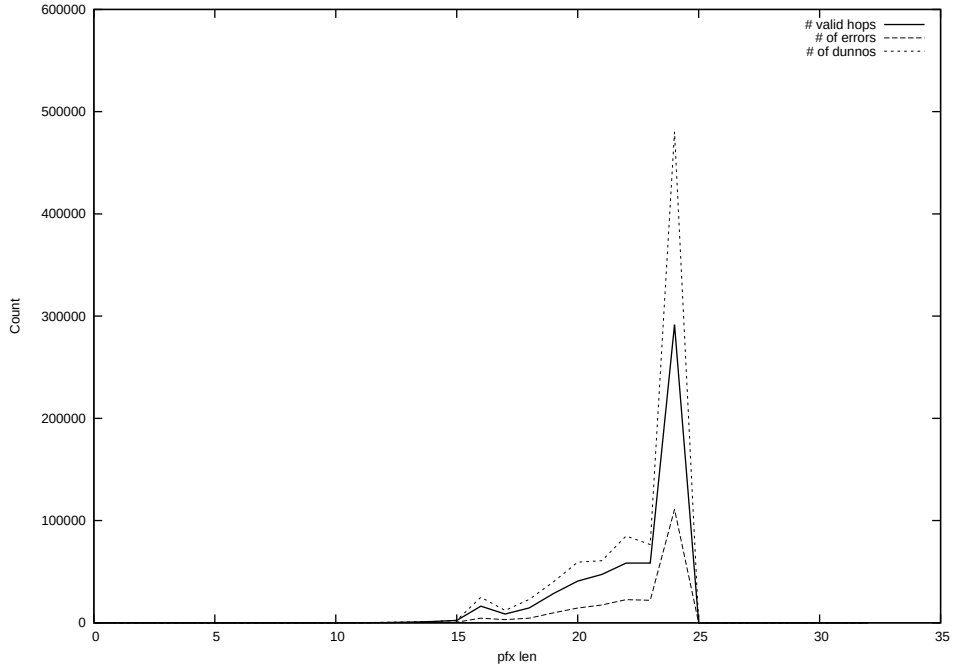
2012-09-06



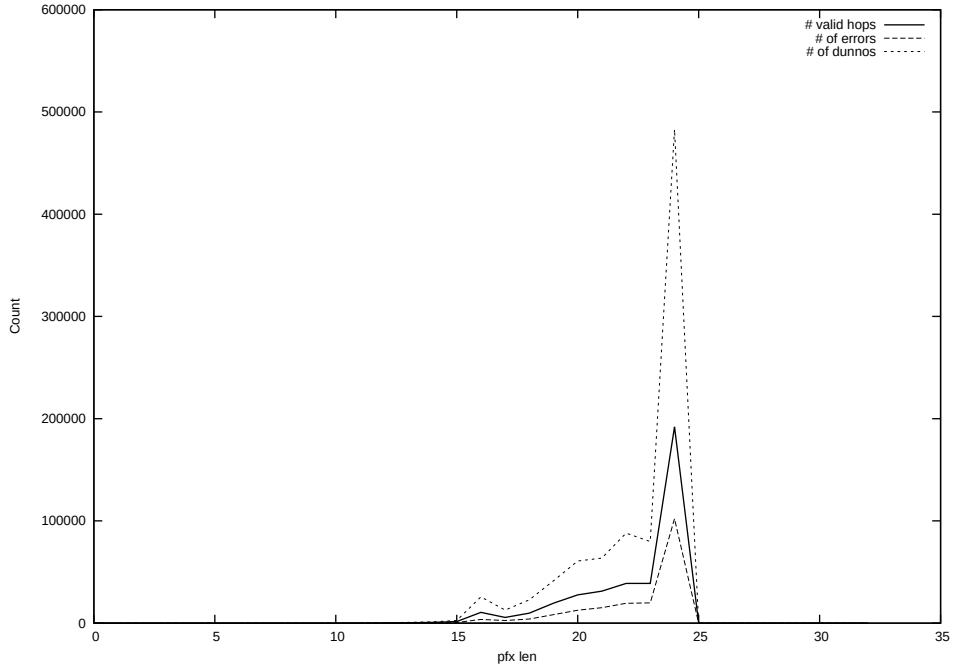
2012-09-07



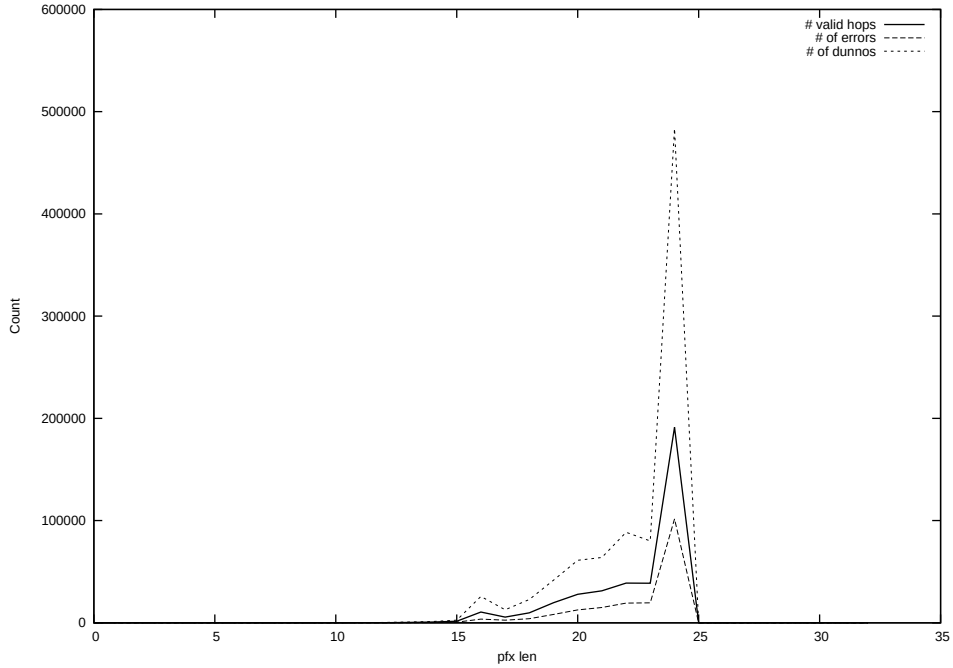
2012-09-08



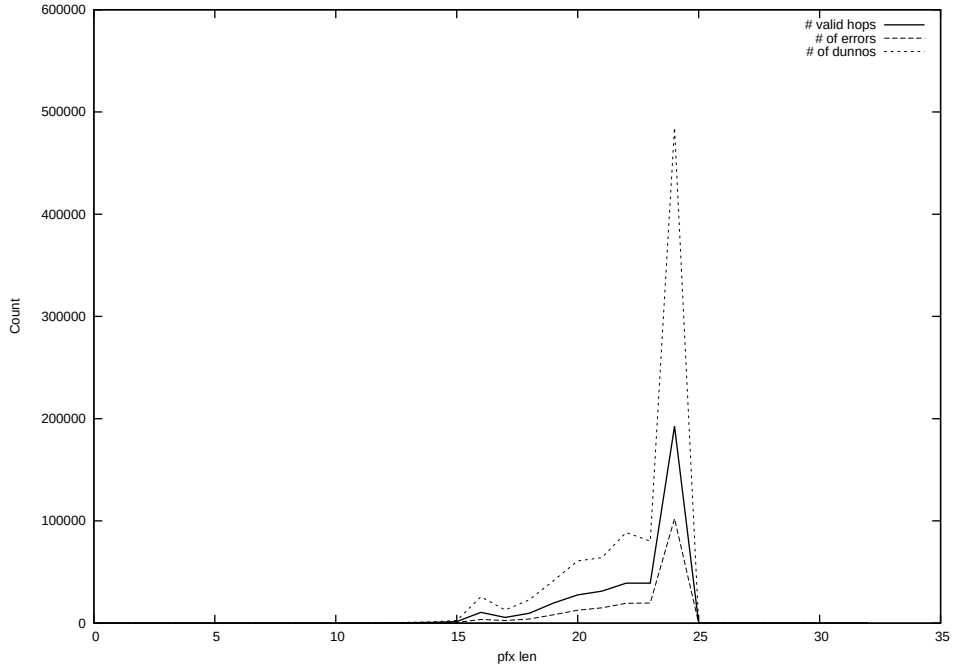
2012-09-09



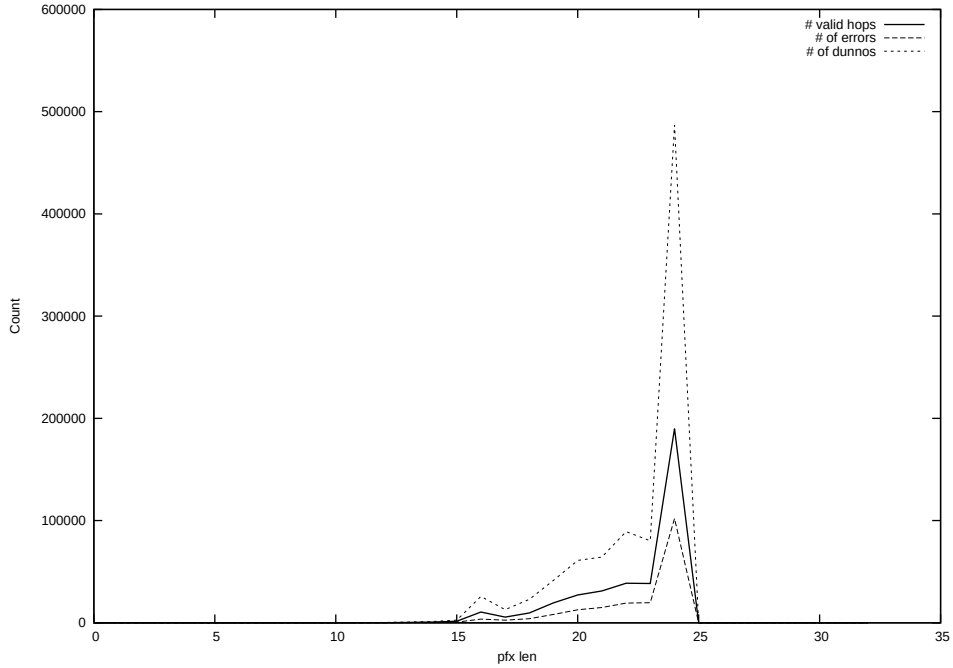
2012-09-10



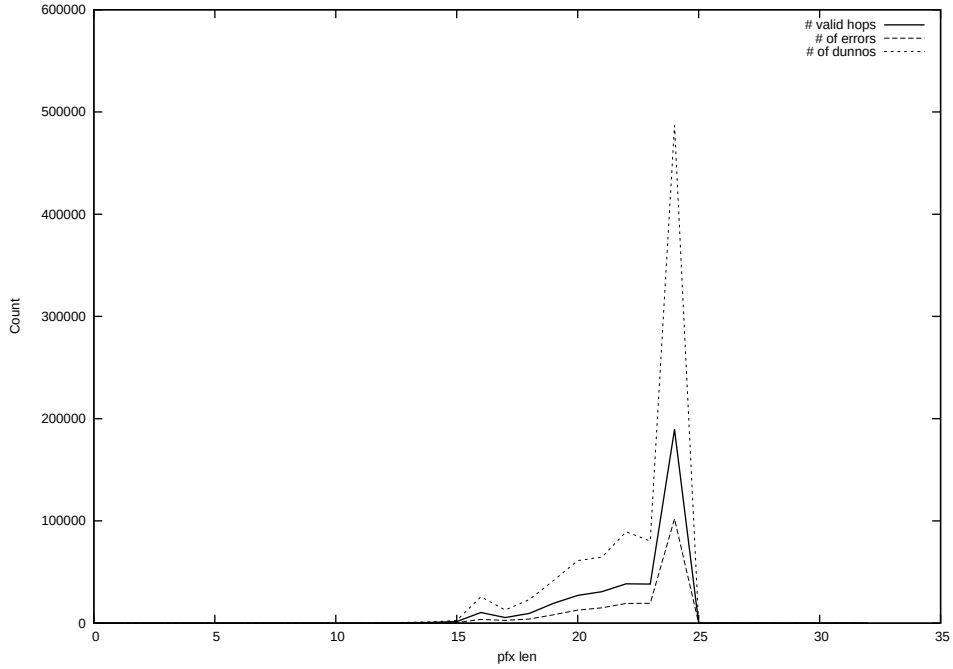
2012-09-11



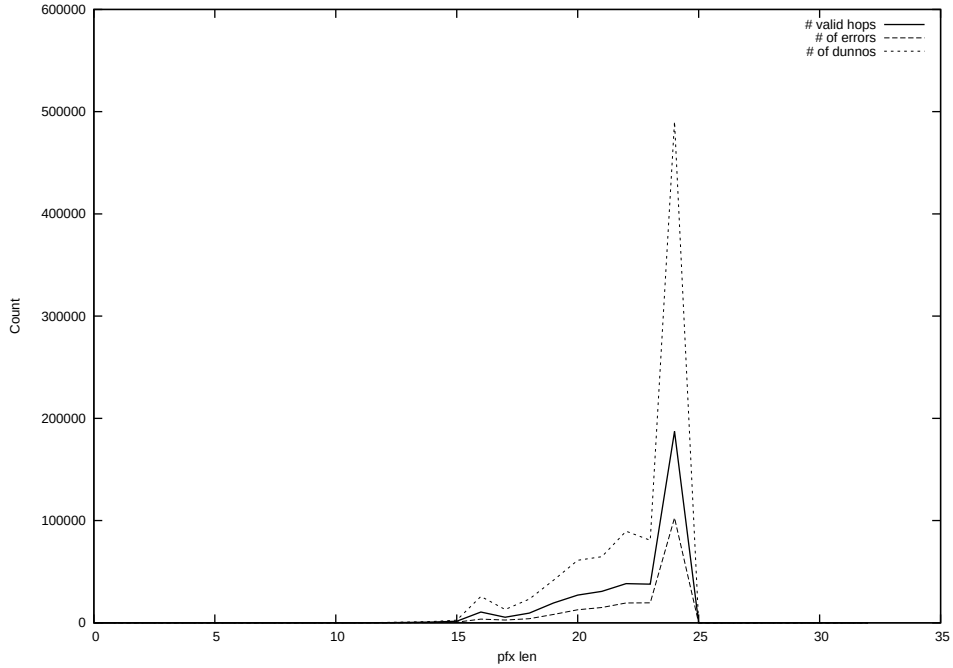
2012-09-12



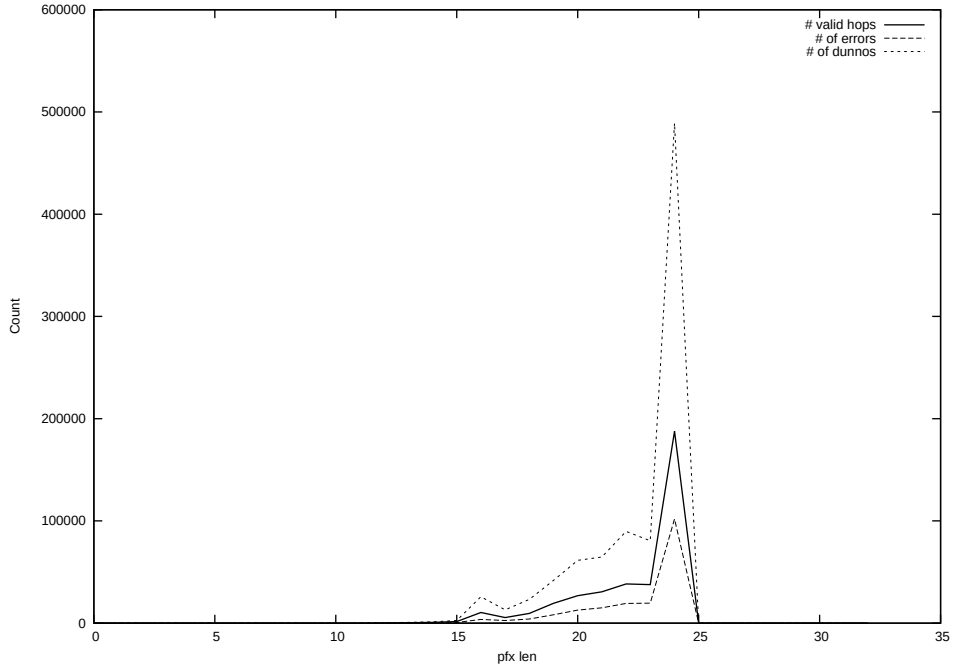
2012-09-13



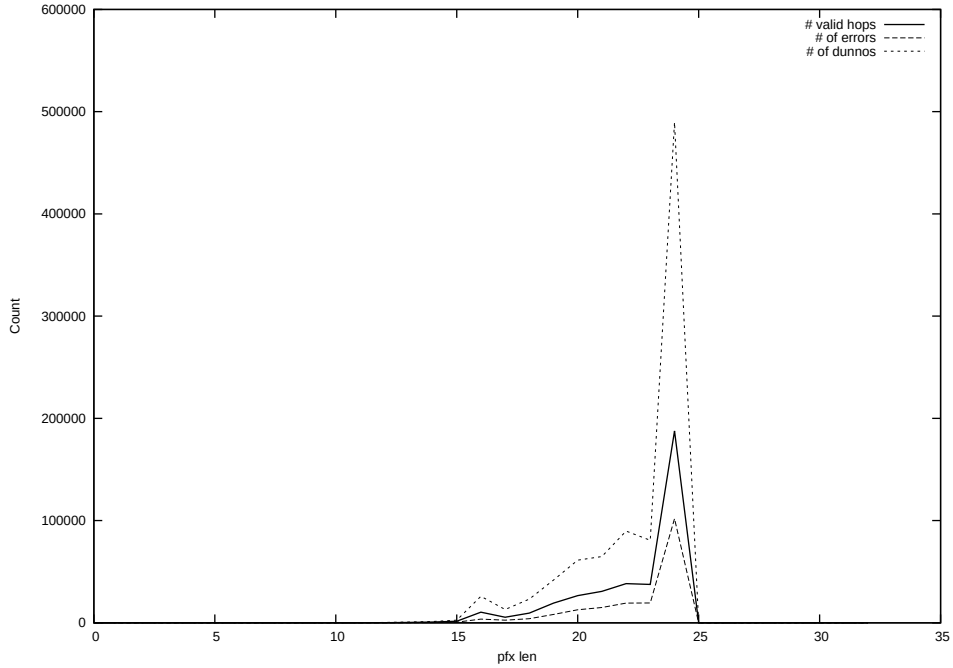
2012-09-14



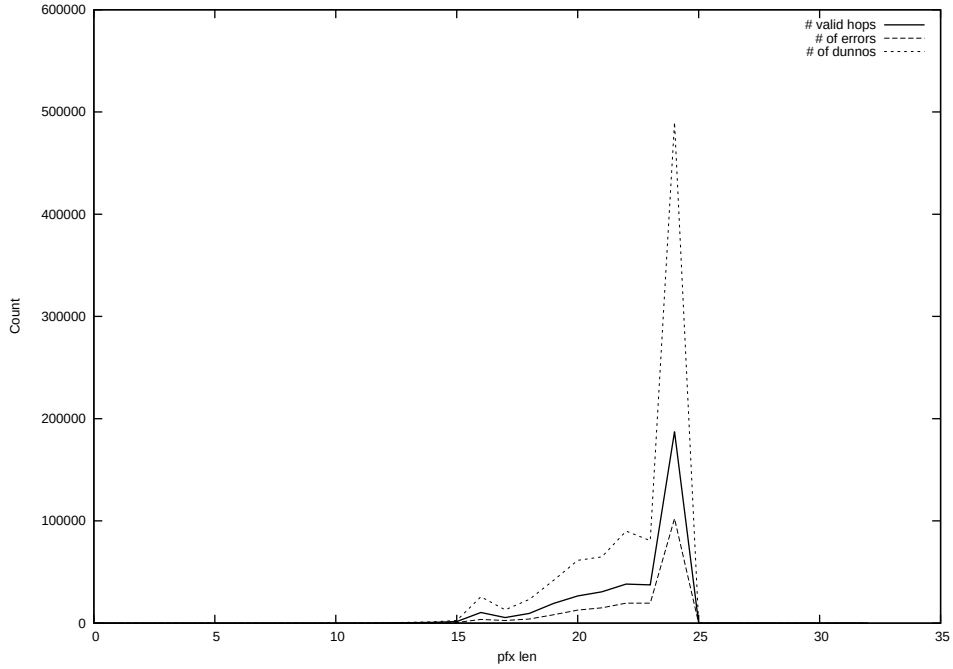
2012-09-15



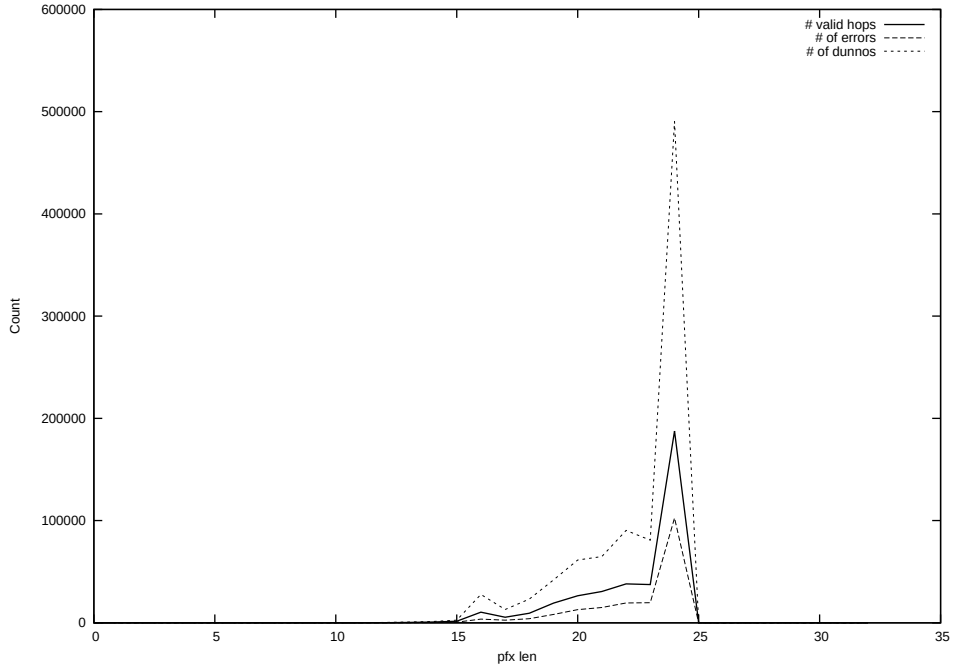
2012-09-16



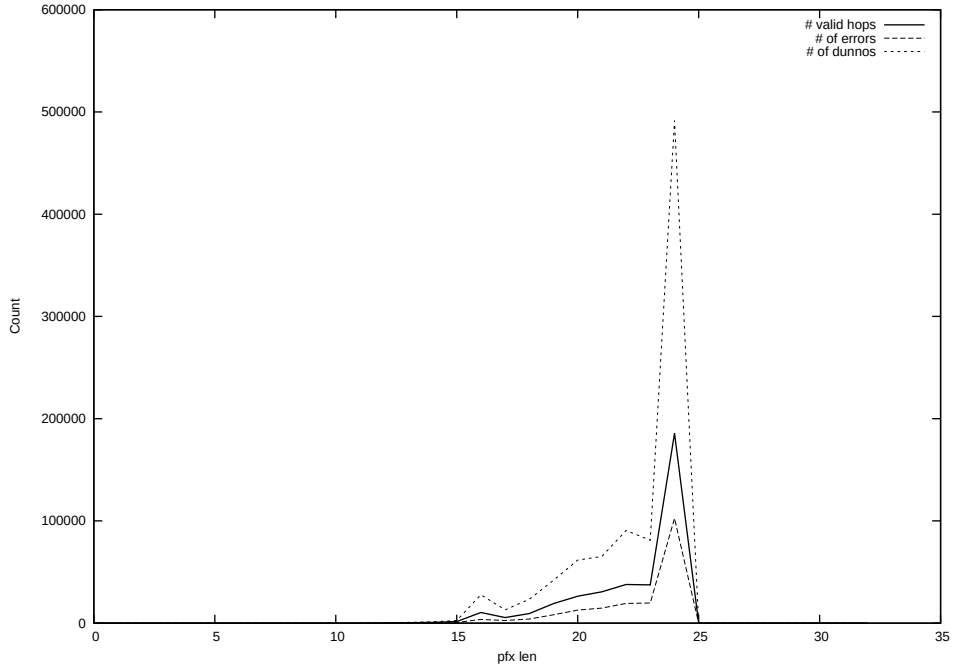
2012-09-17



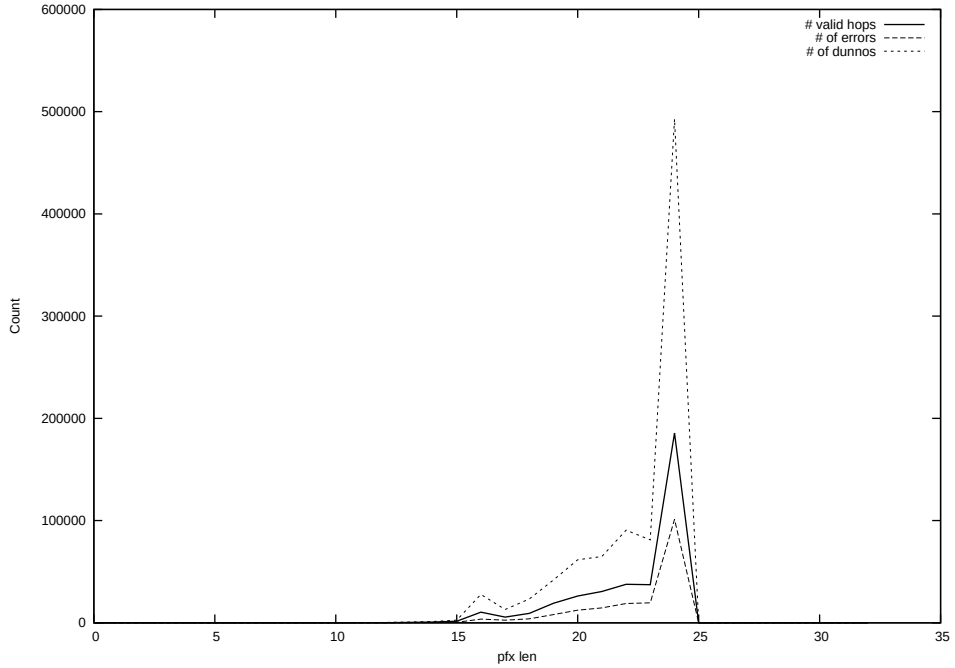
2012-09-18



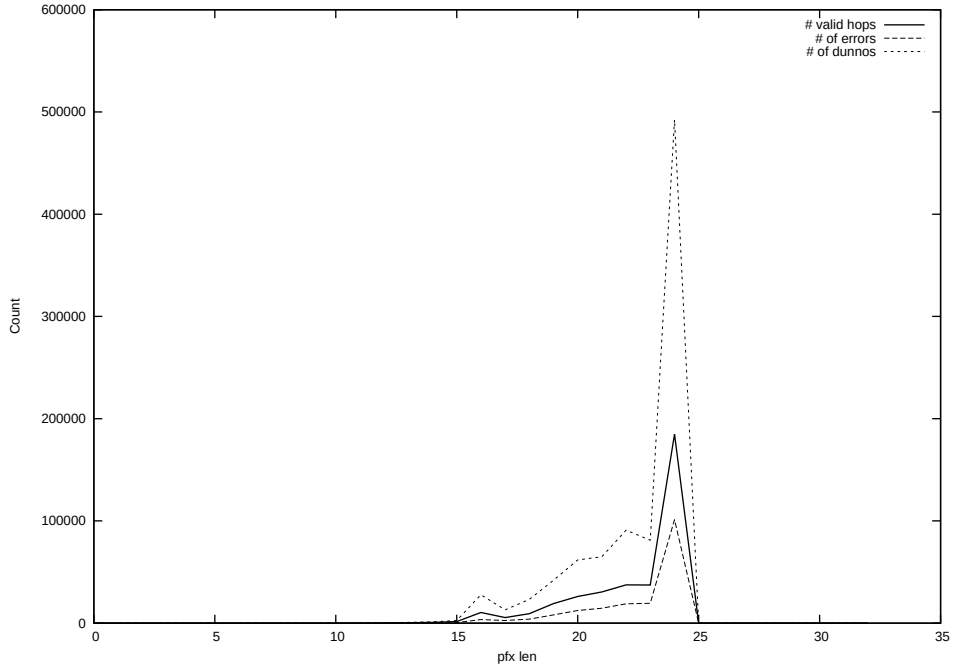
2012-09-19



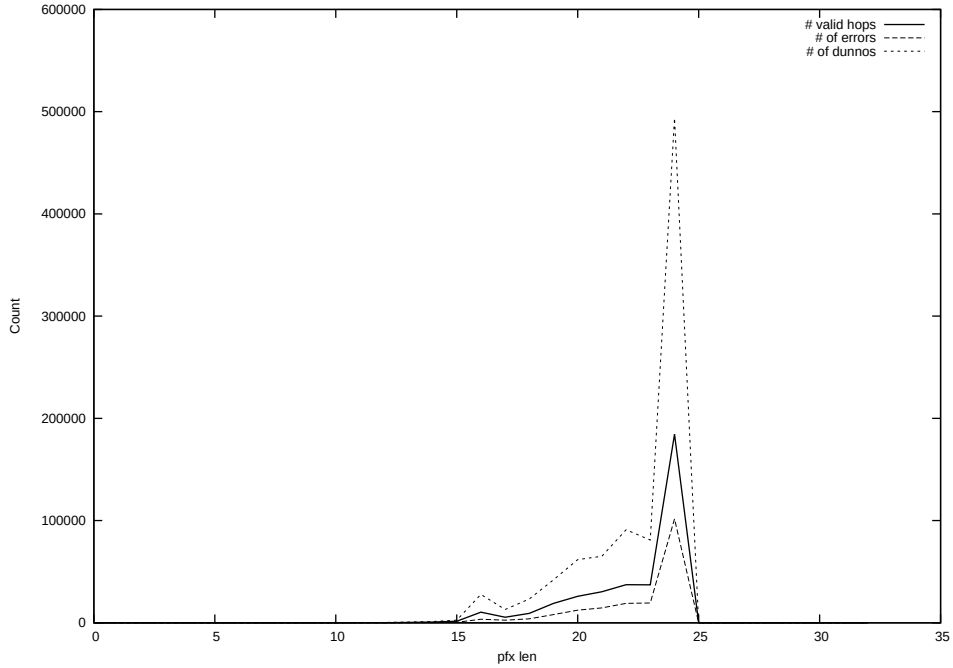
2012-09-20



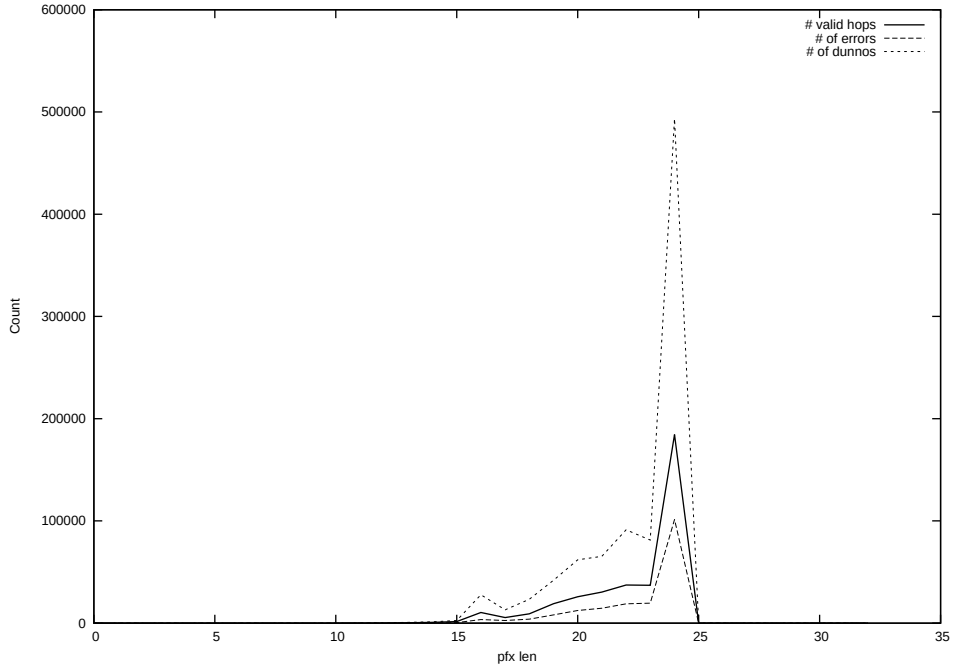
2012-09-21



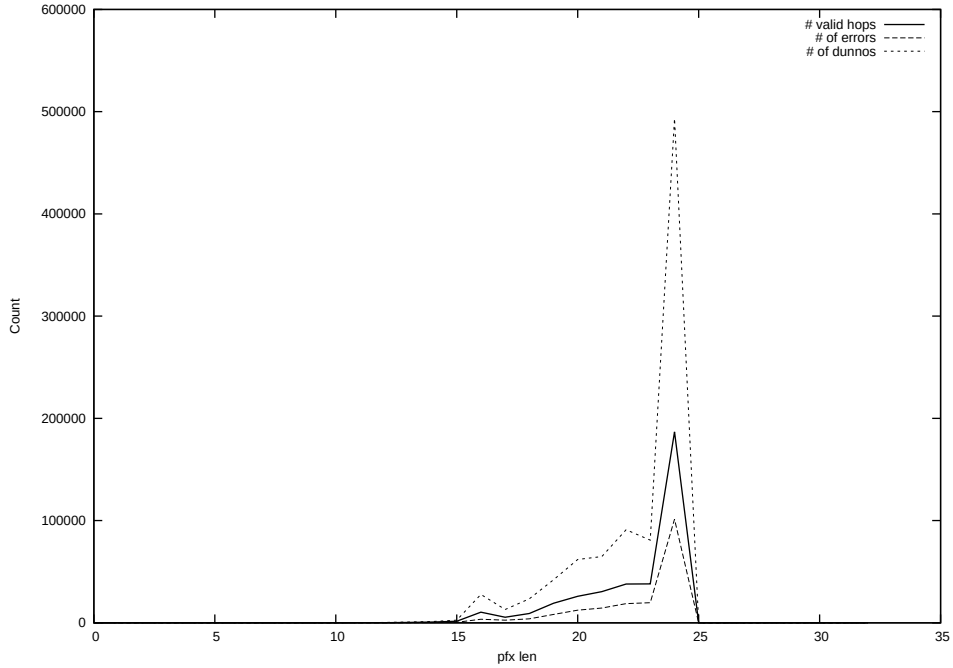
2012-09-22



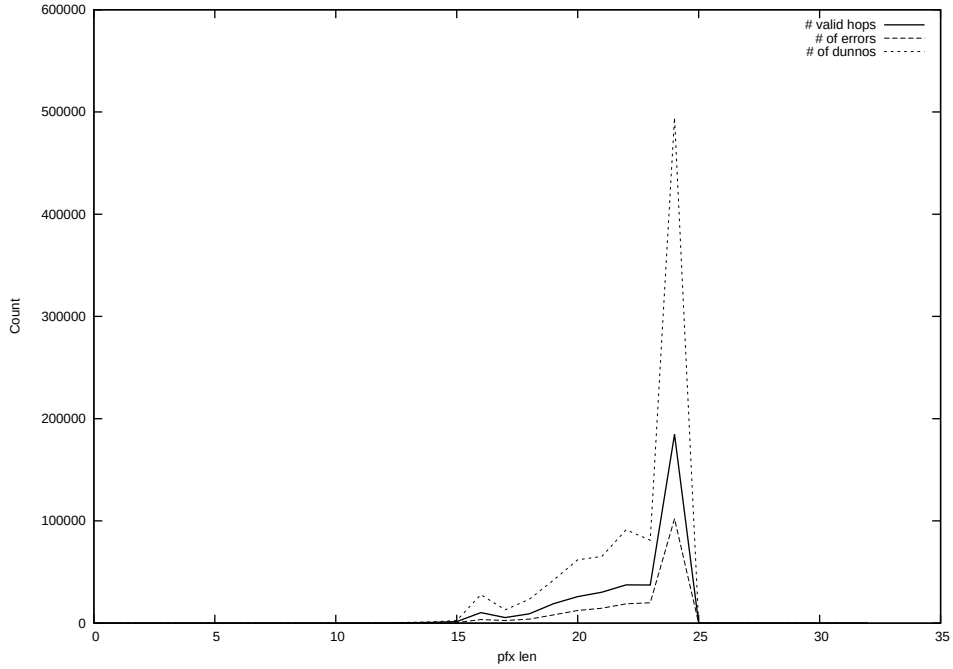
2012-09-23



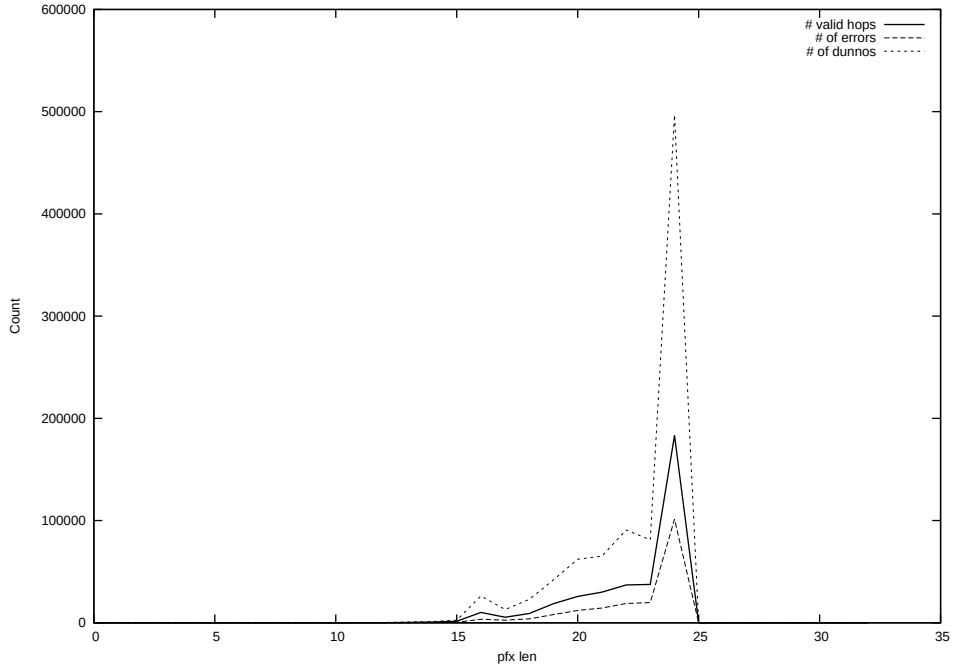
2012-09-24



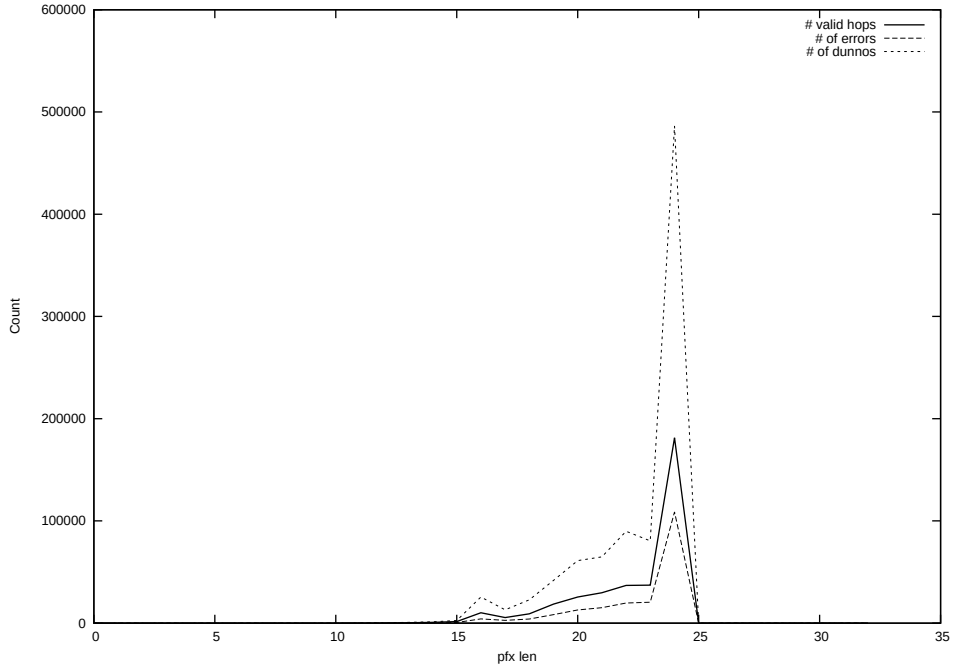
2012-09-25



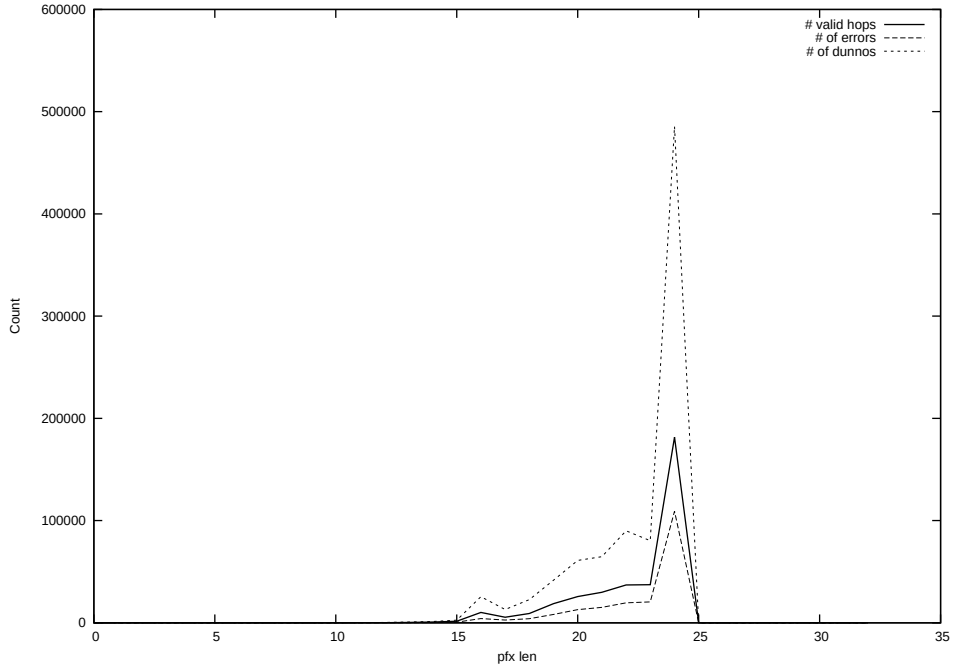
2012-09-26



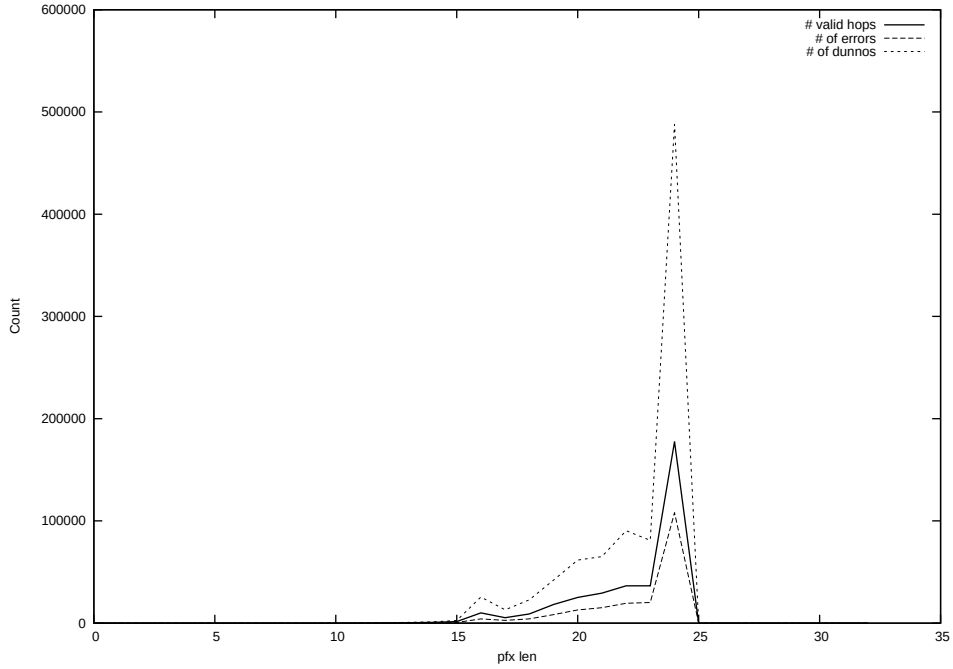
2012-09-27



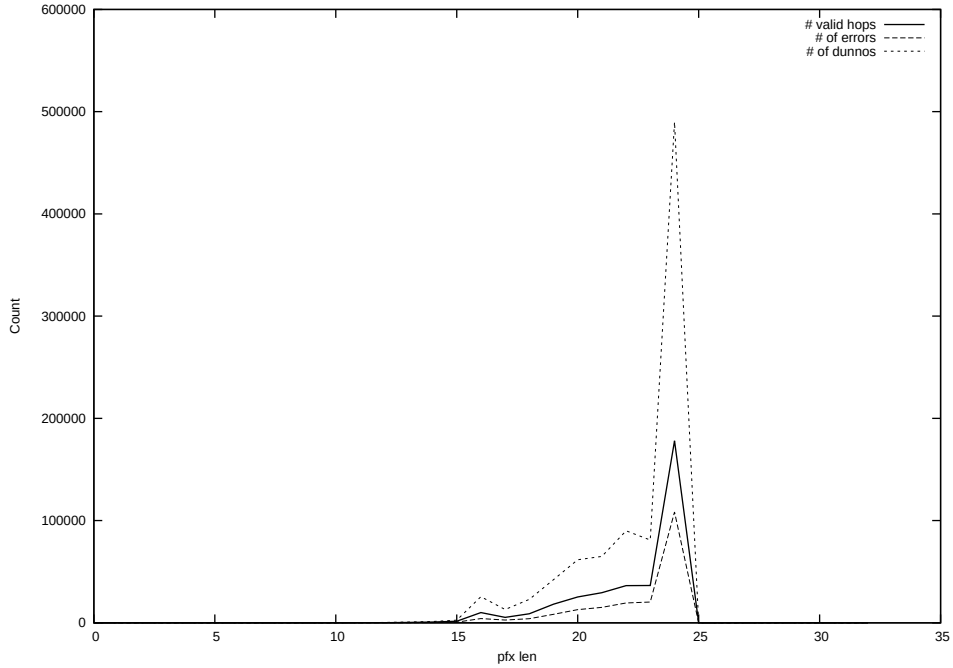
2012-09-28



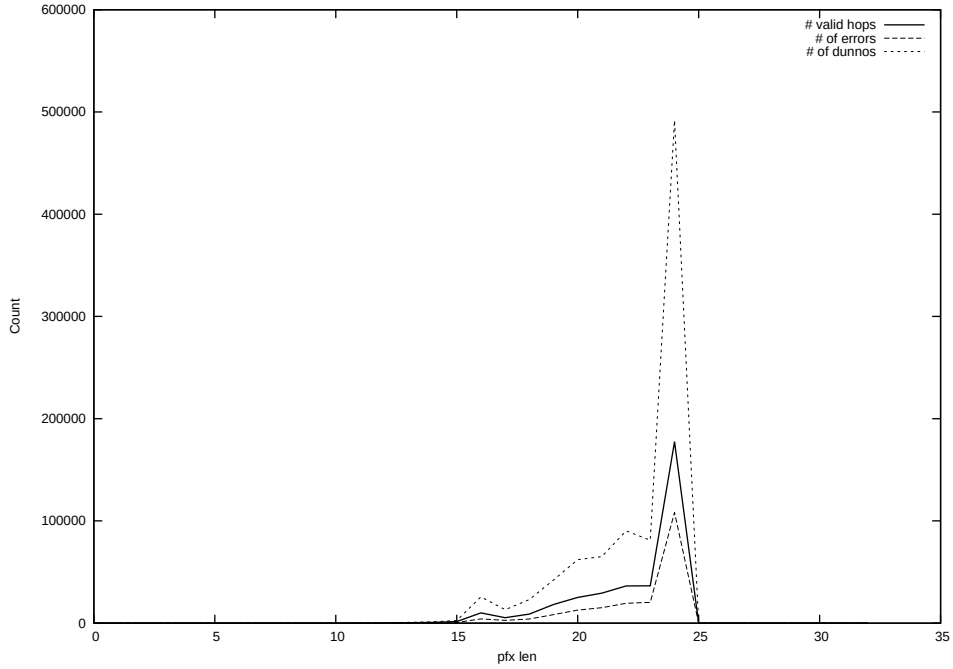
2012-09-29



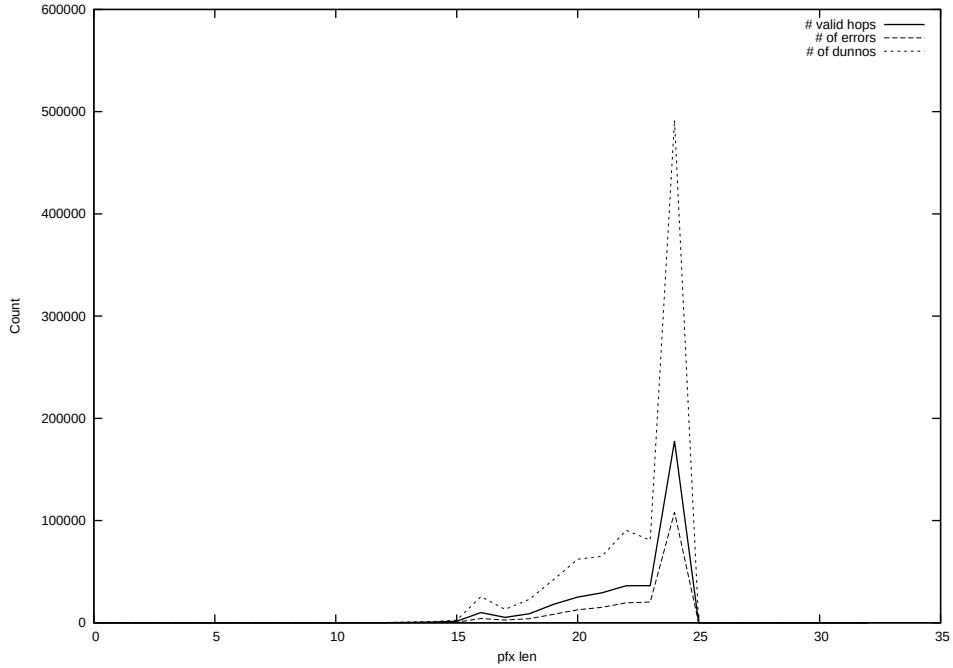
2012-09-30



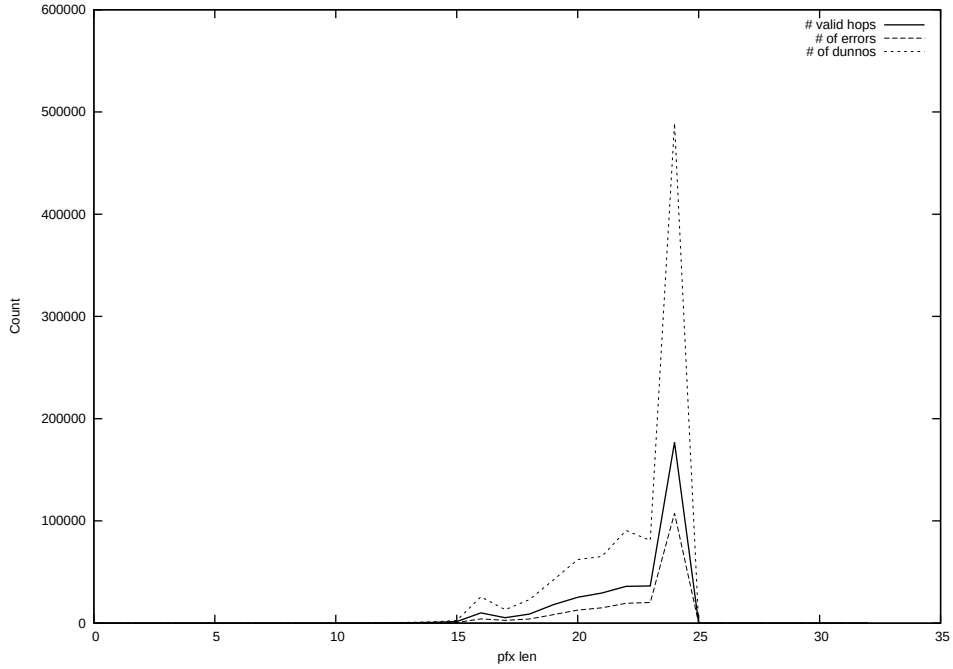
2012-10-01



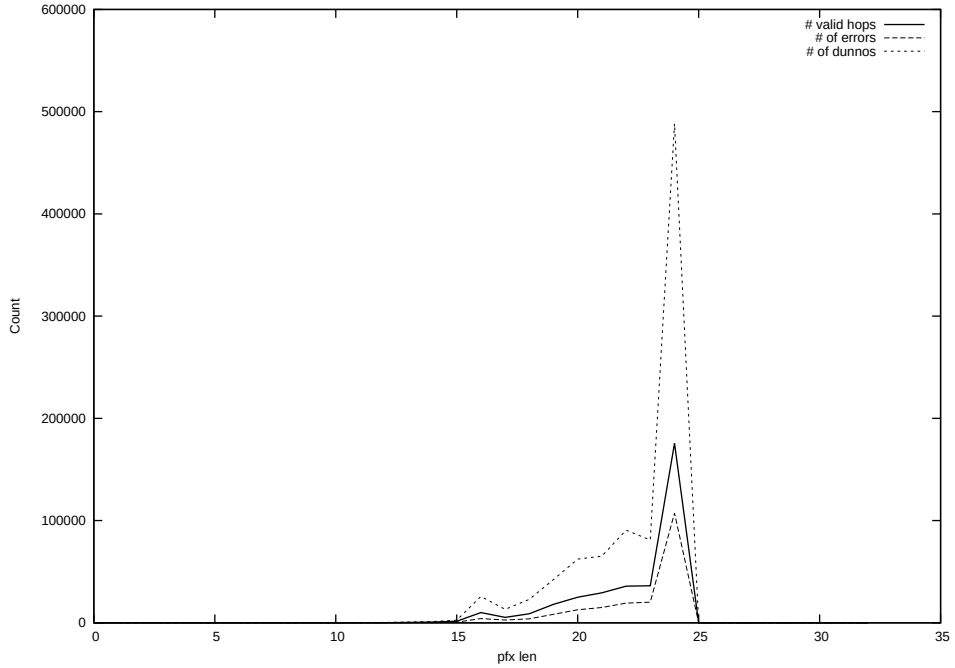
2012-10-02



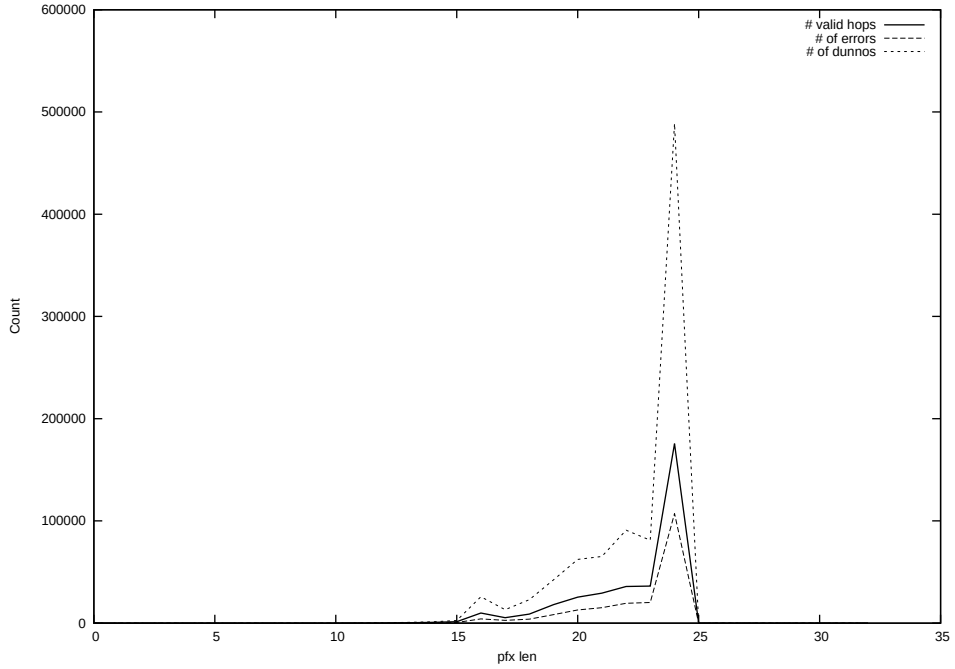
2012-10-03



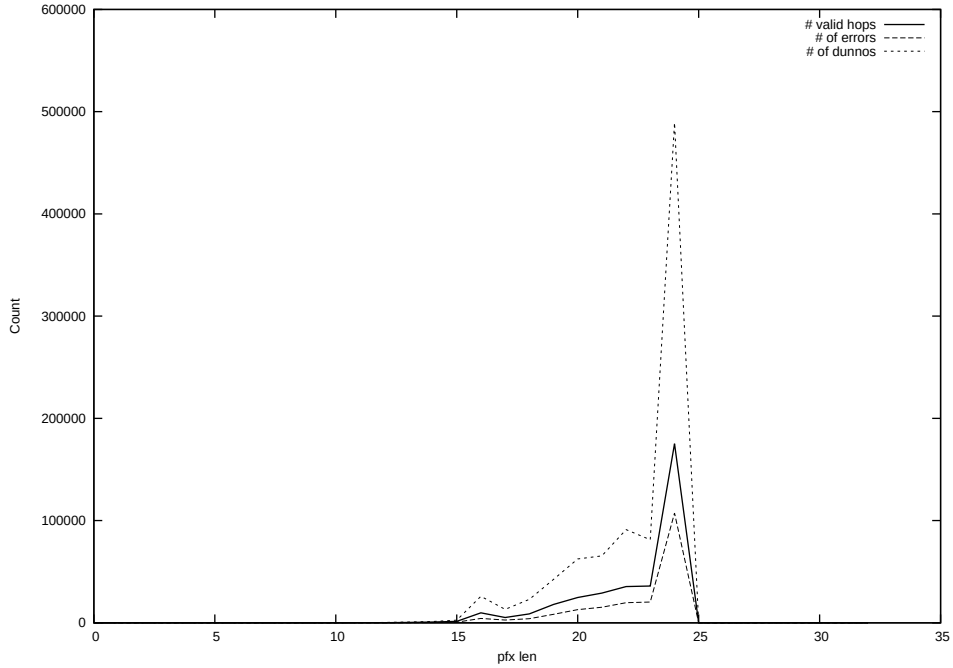
2012-10-04



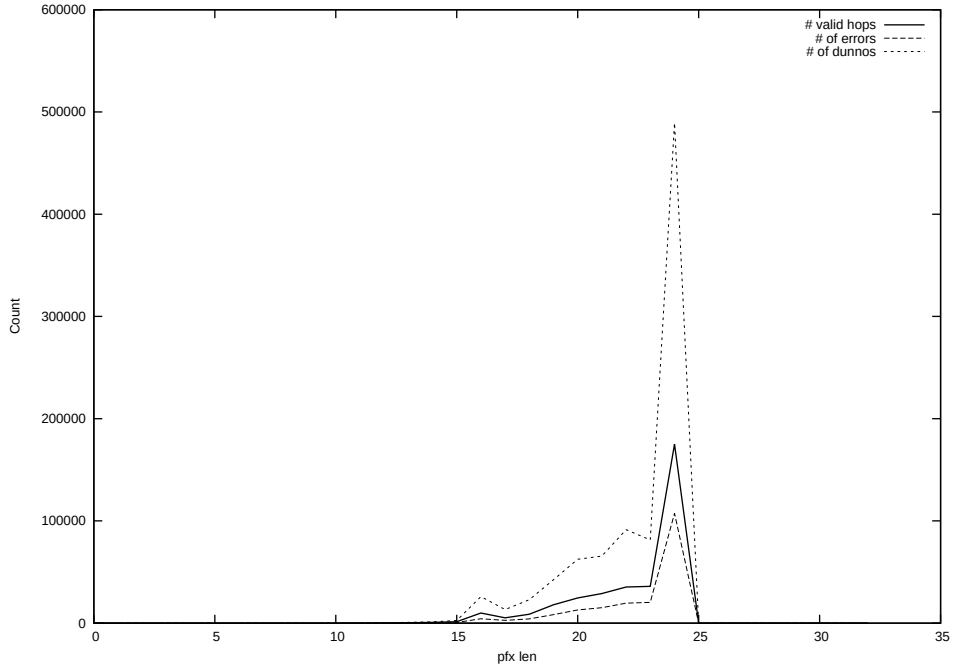
2012-10-05



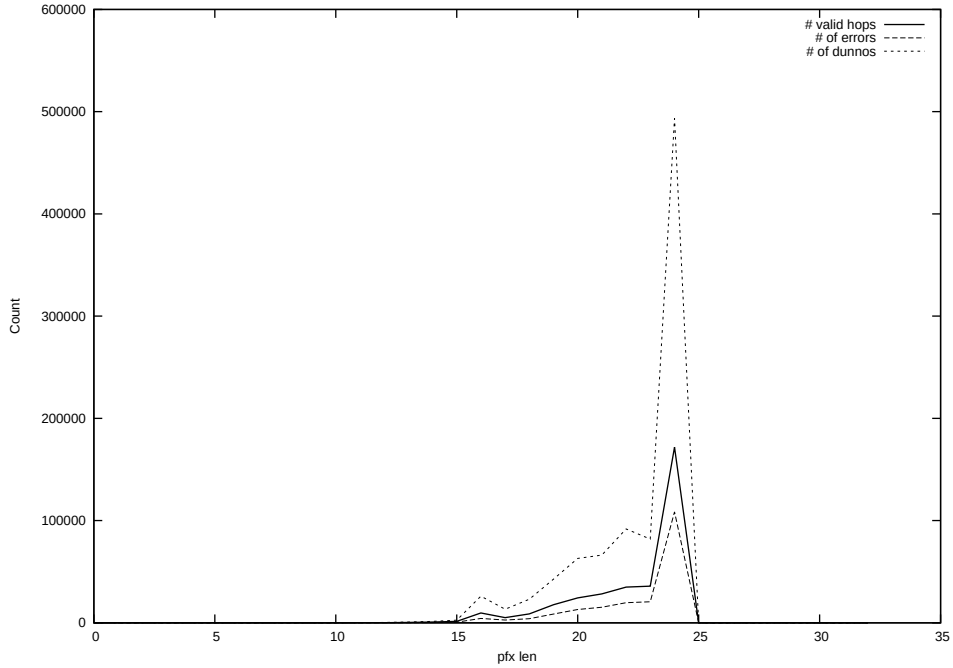
2012-10-06



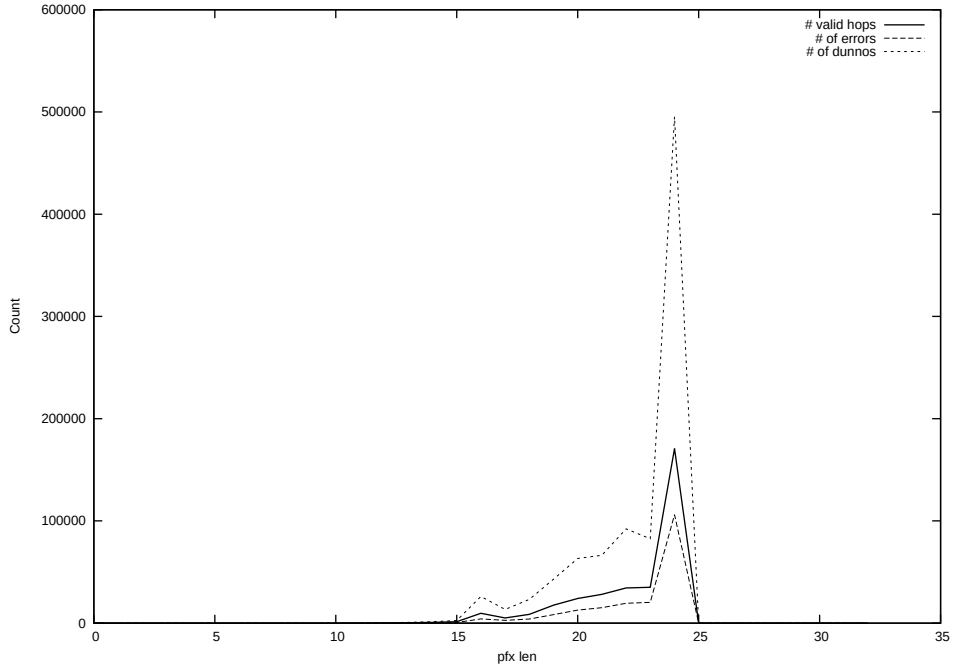
2012-10-07



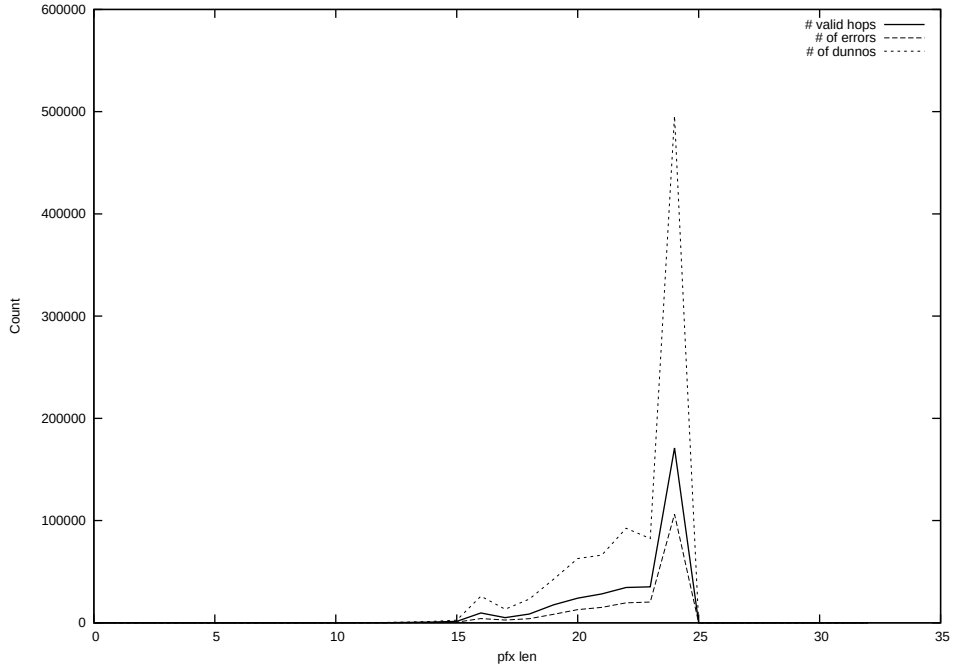
2012-10-08



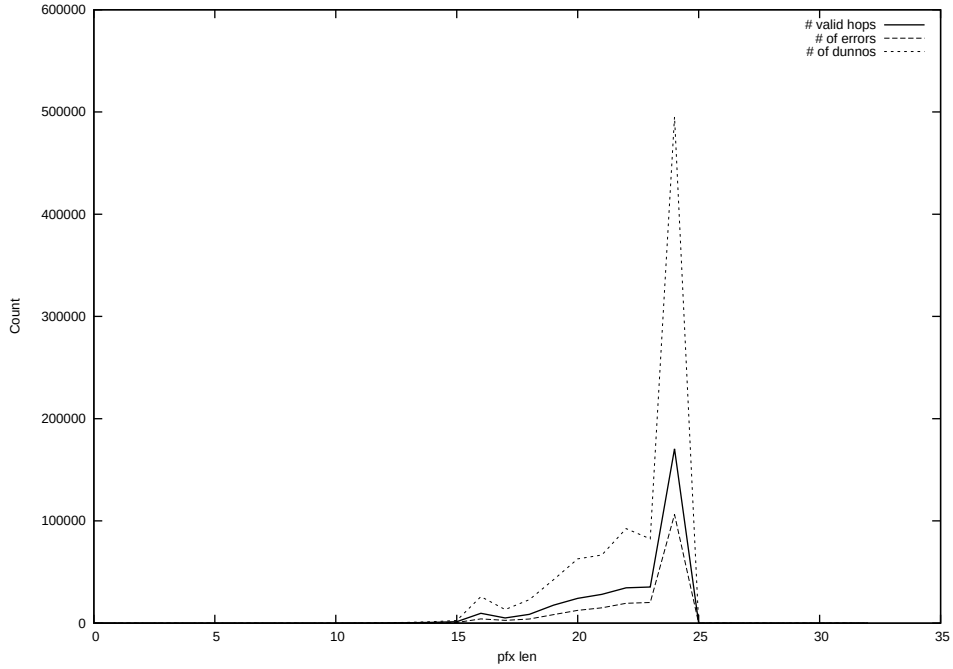
2012-10-09



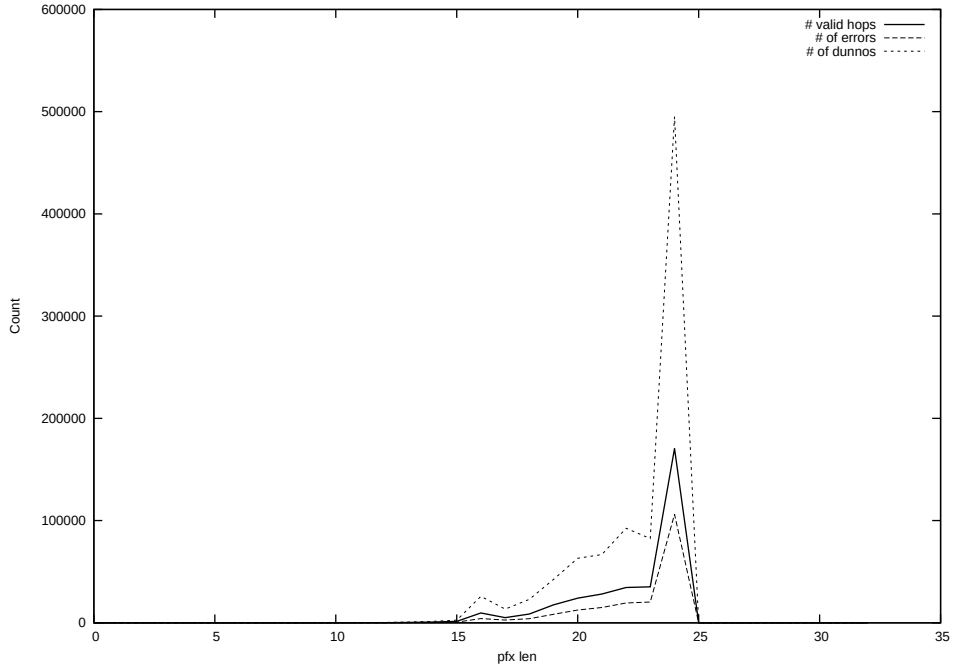
2012-10-10



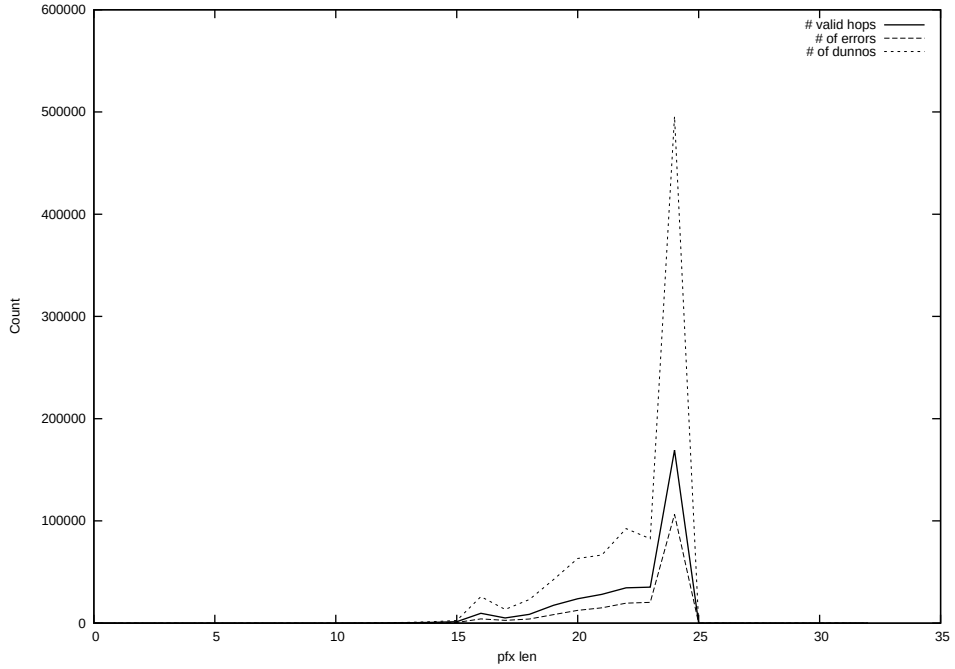
2012-10-11



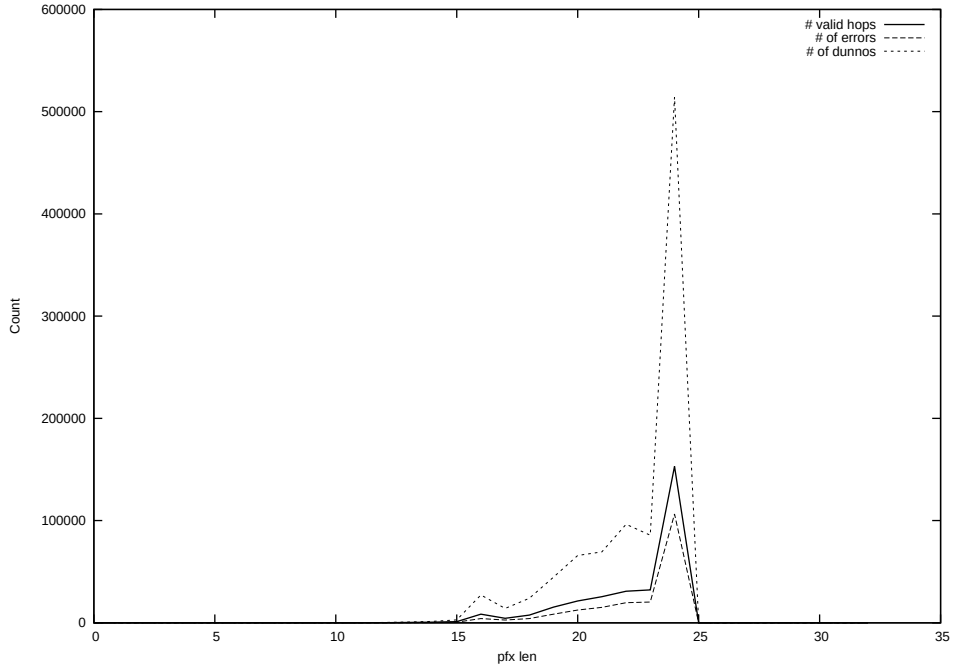
2012-10-12



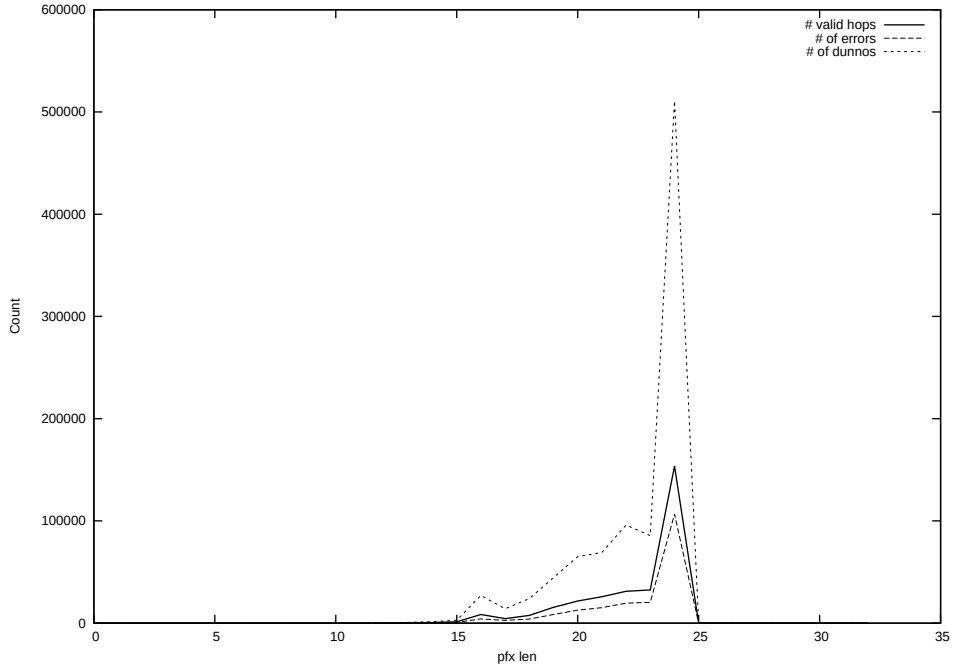
2012-10-13



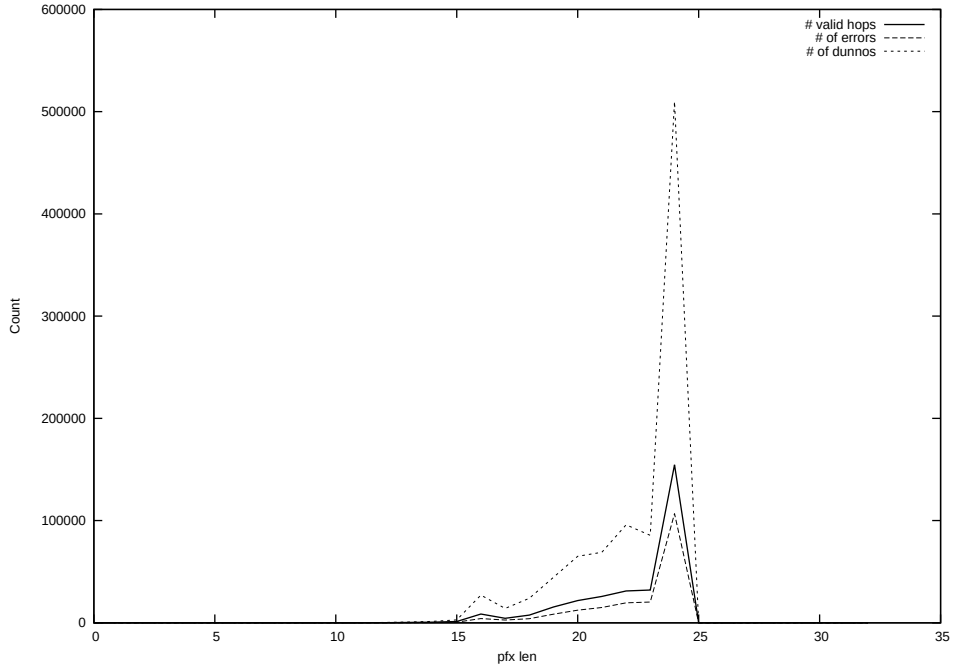
2012-10-14



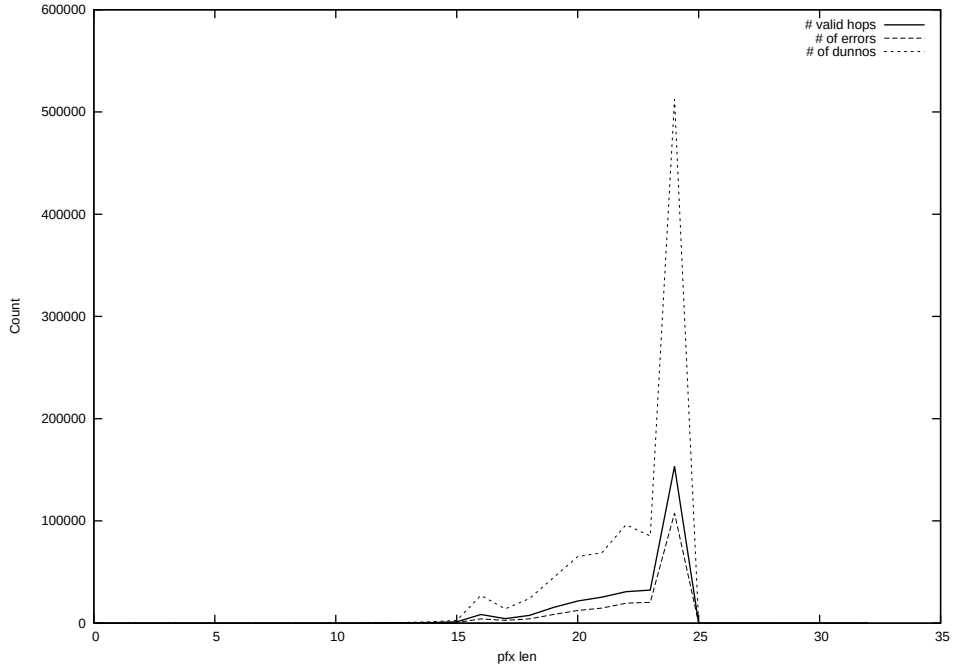
2012-10-15



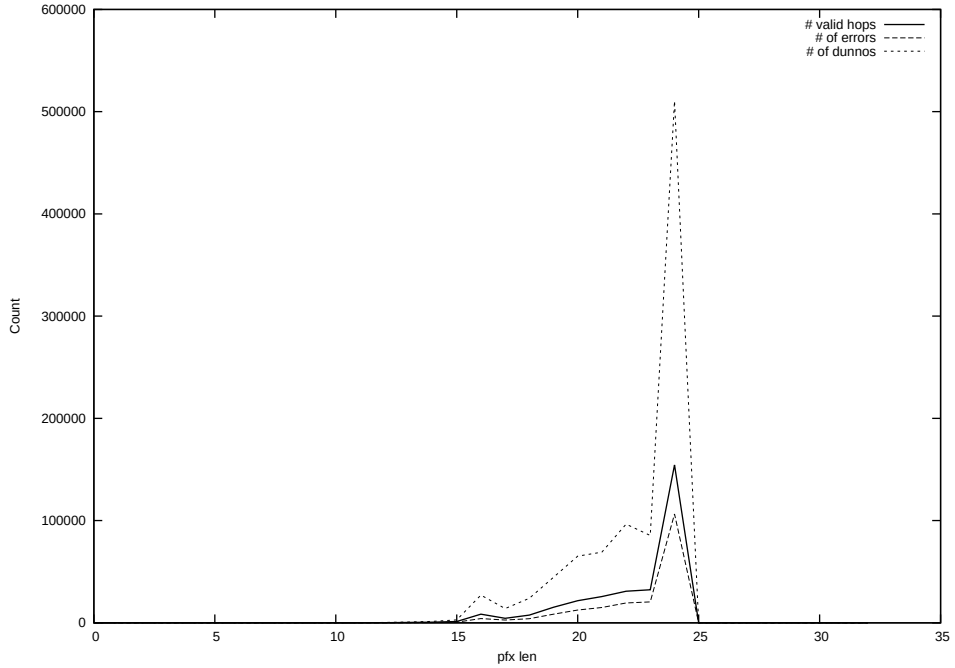
2012-10-16



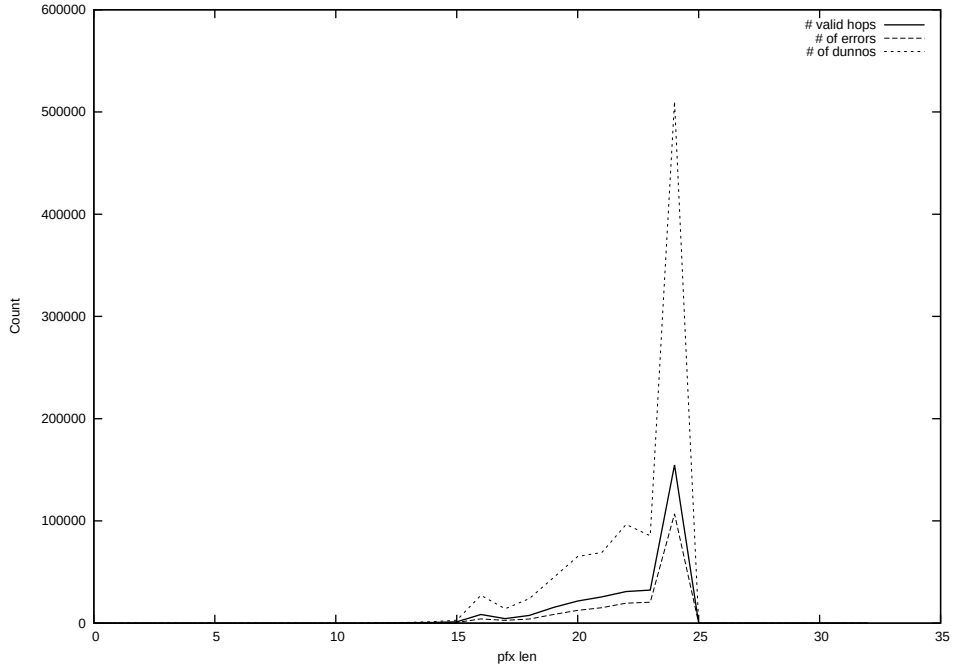
2012-10-18



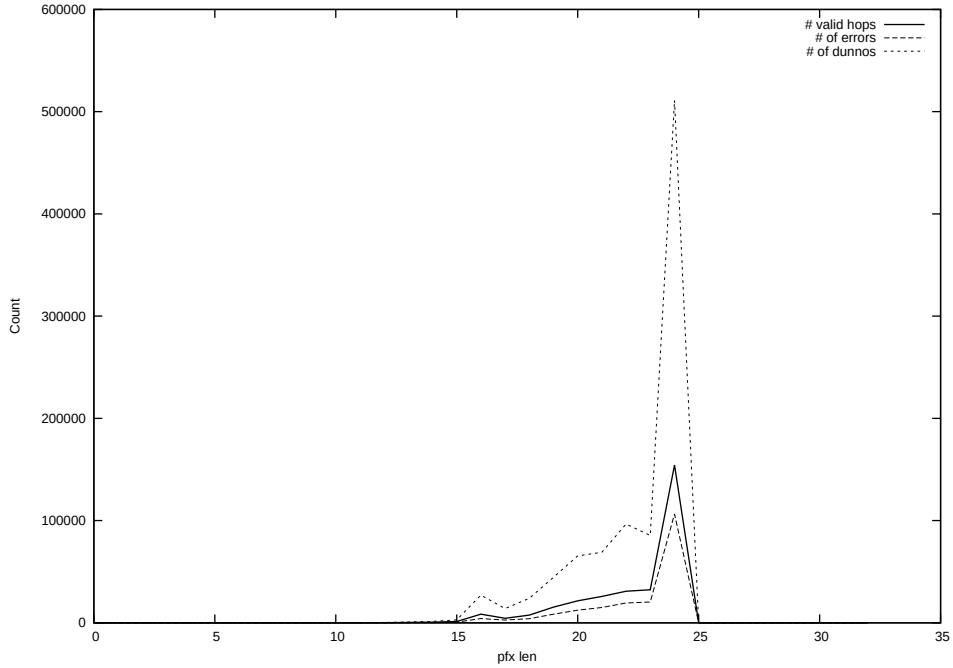
2012-10-19



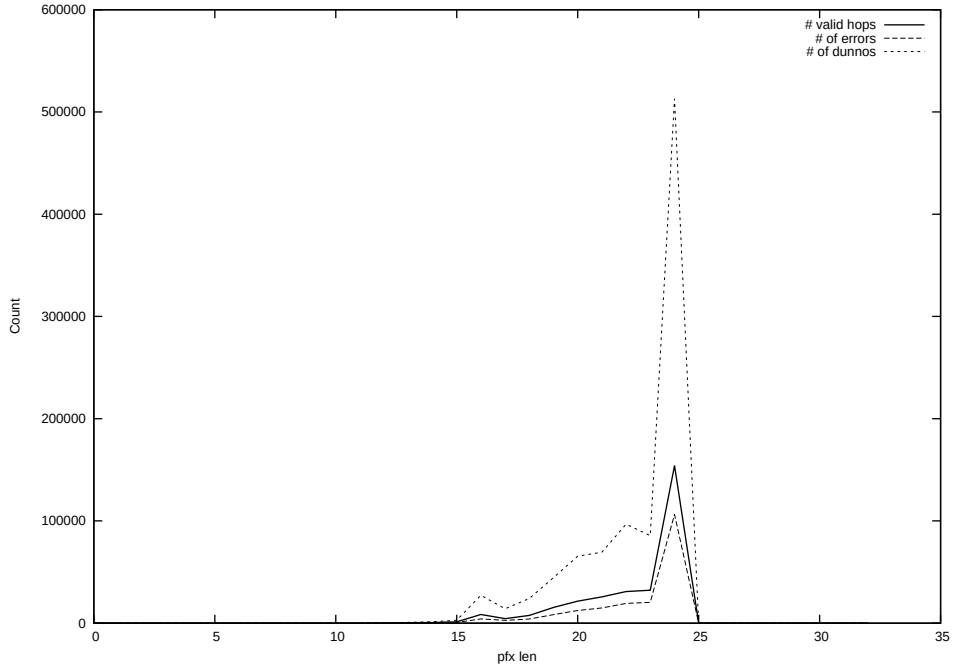
2012-10-20



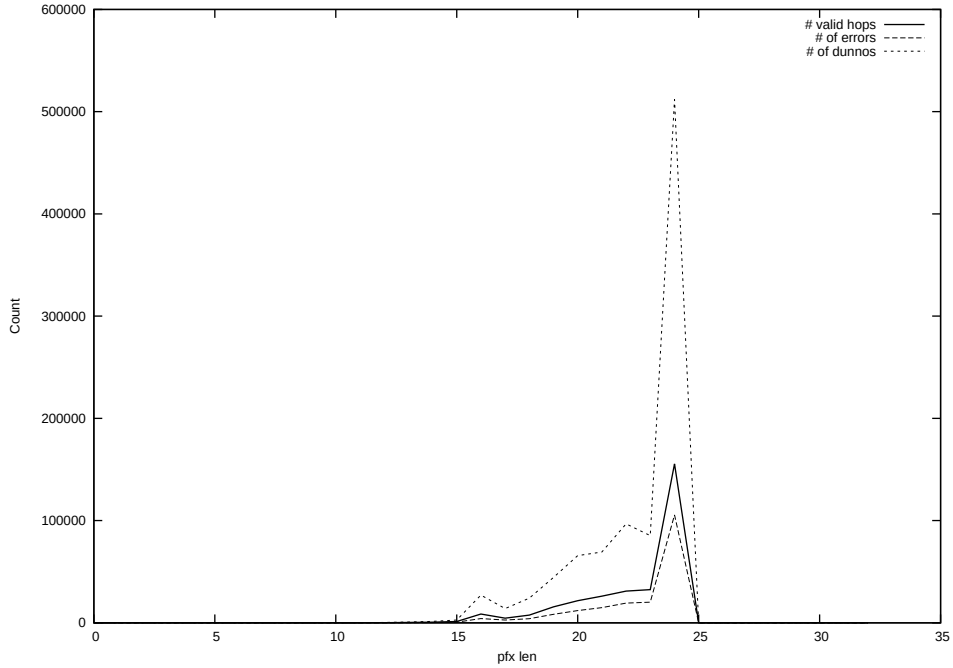
2012-10-21



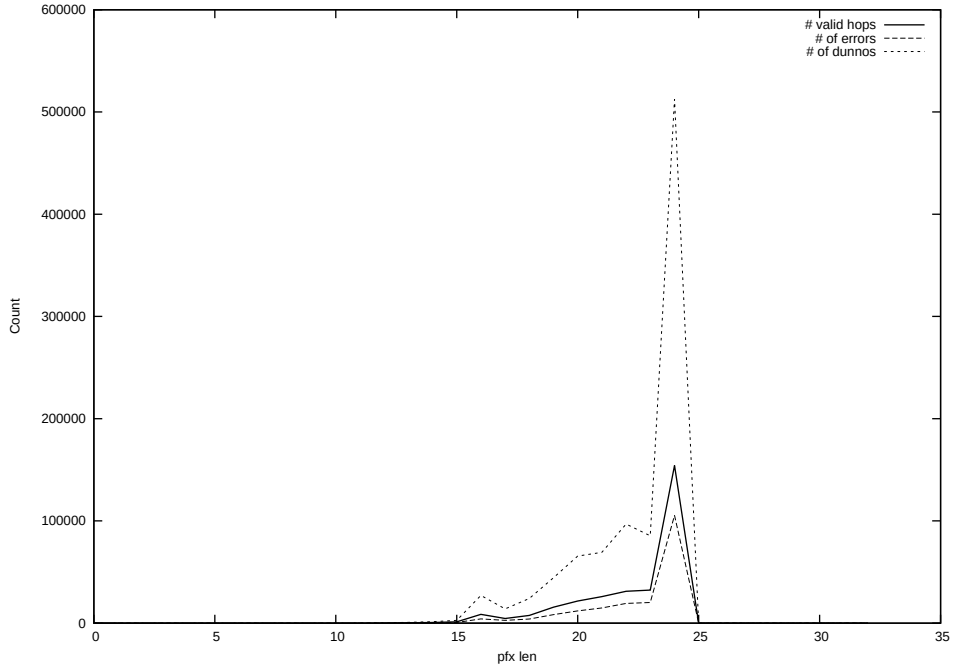
2012-10-22



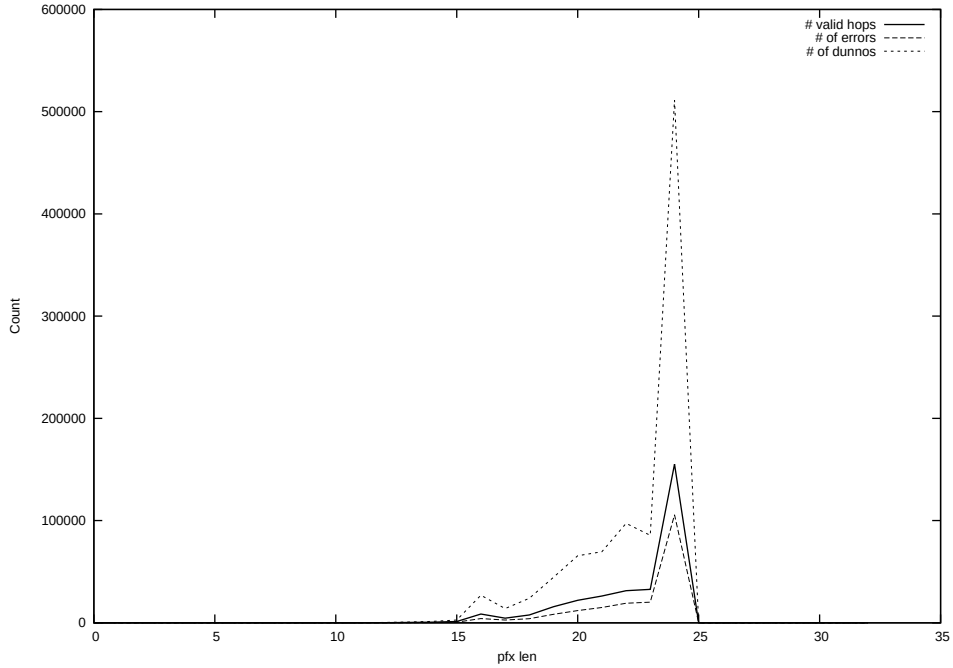
2012-10-23



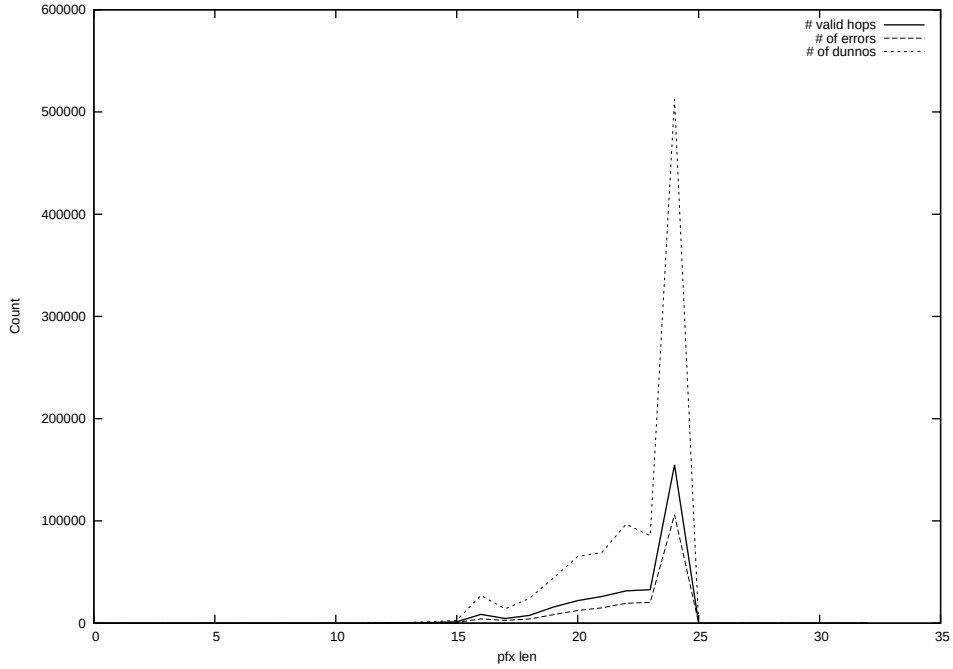
2012-10-24



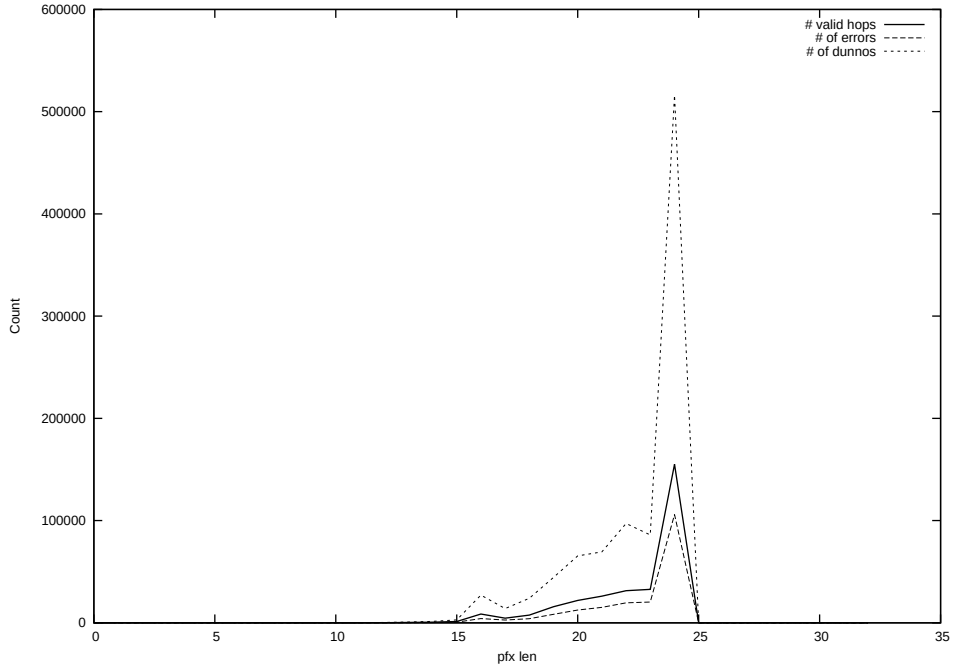
2012-10-25



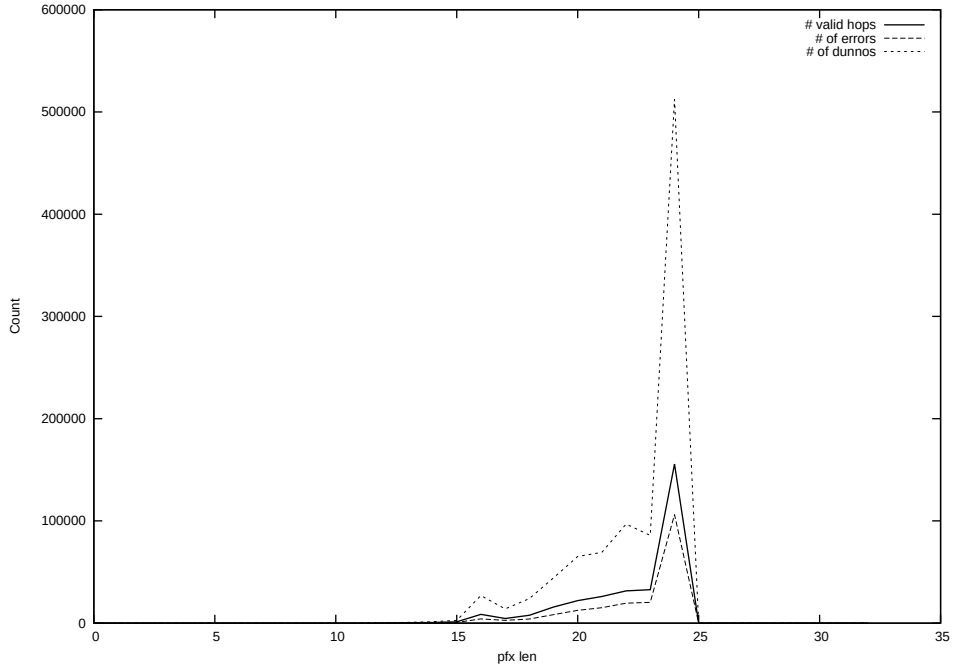
2012-10-26



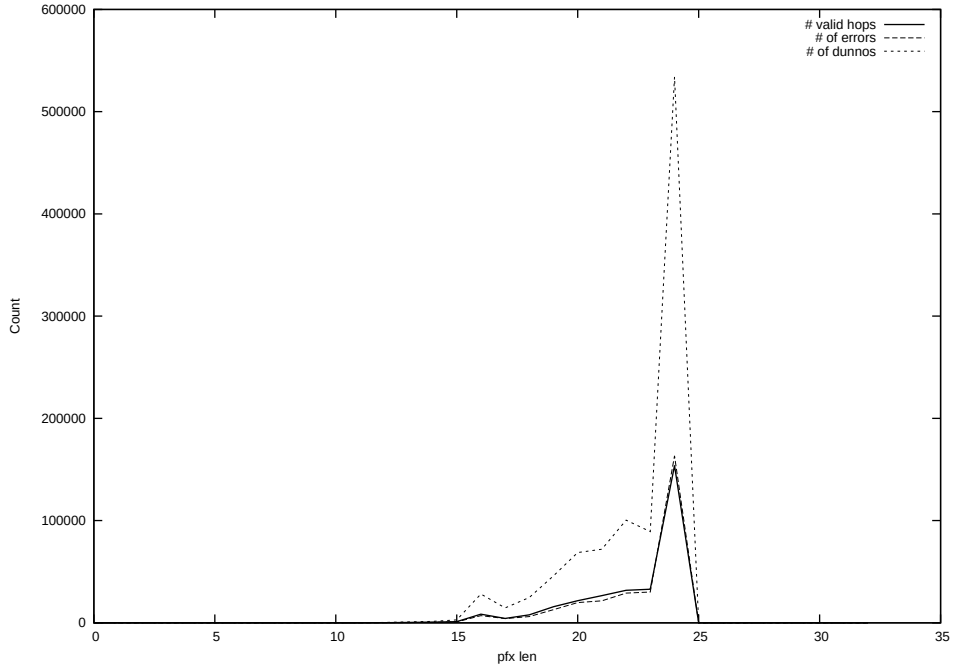
2012-10-27



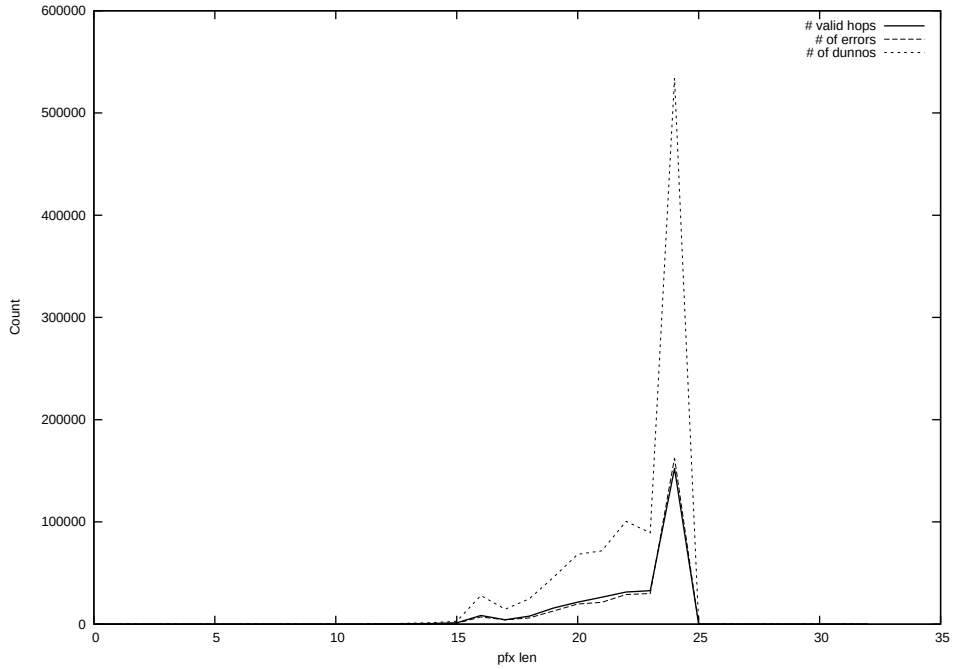
2012-10-28



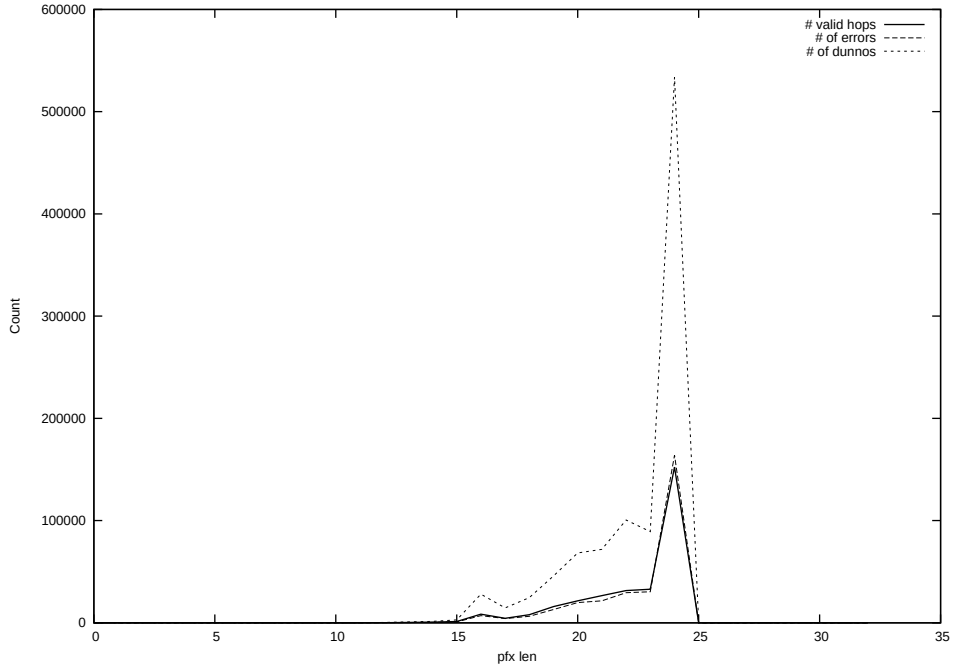
2012-10-29



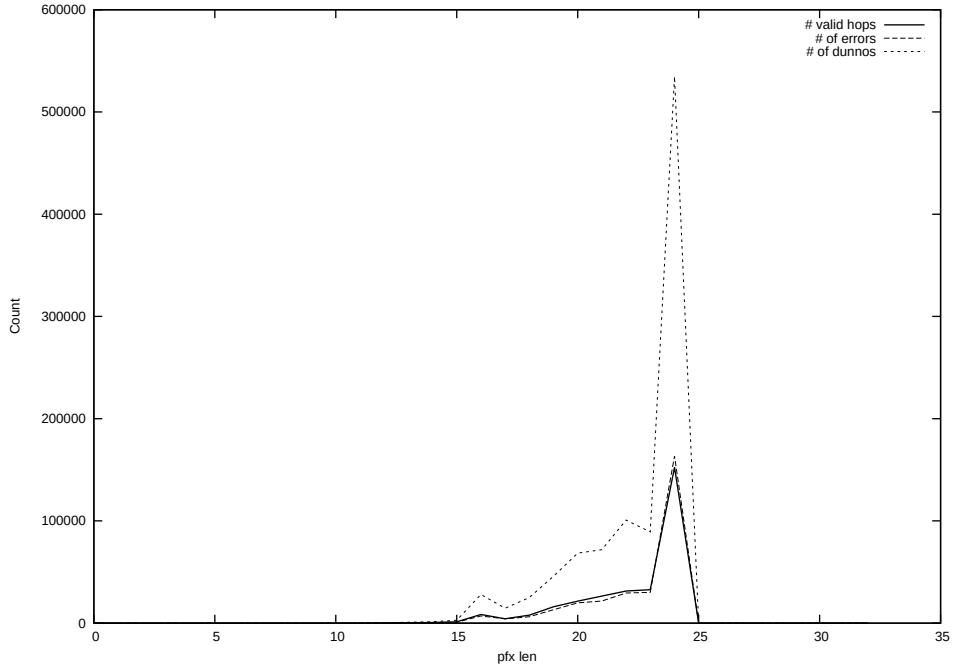
2012-10-30



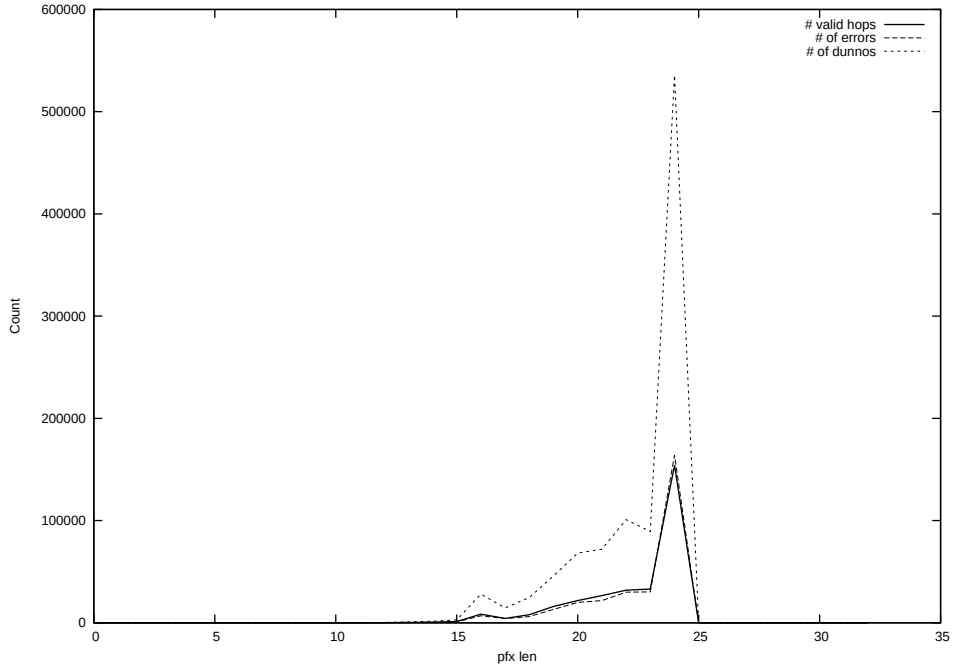
2012-10-31



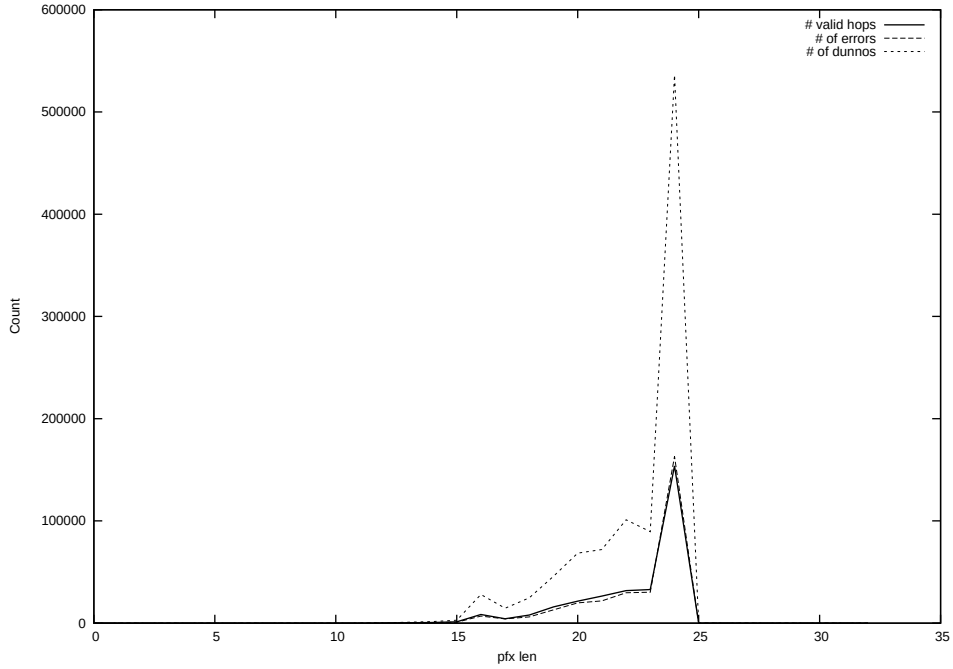
2012-11-01



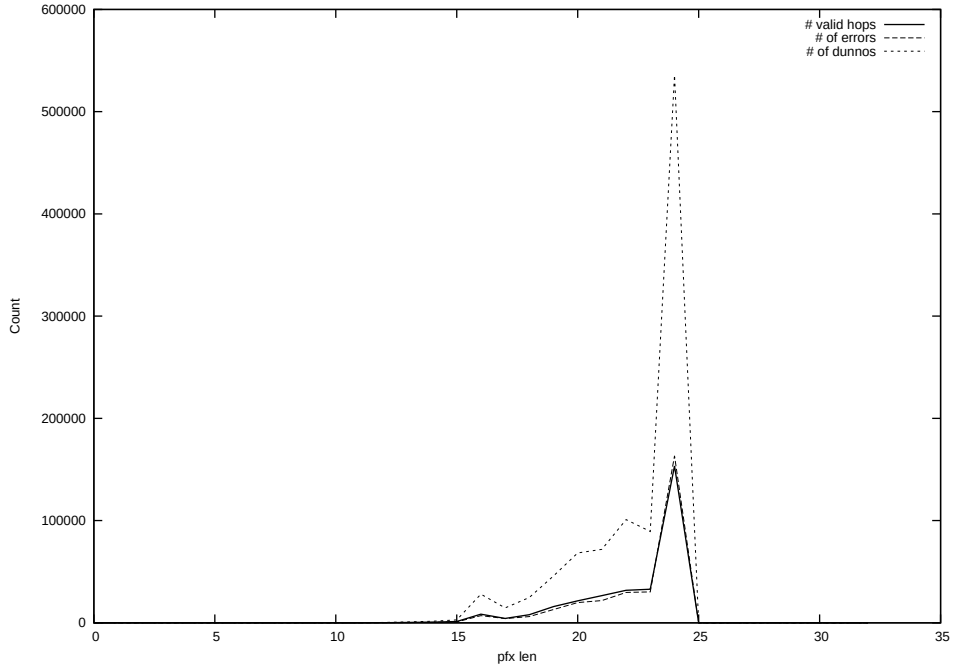
2012-11-02



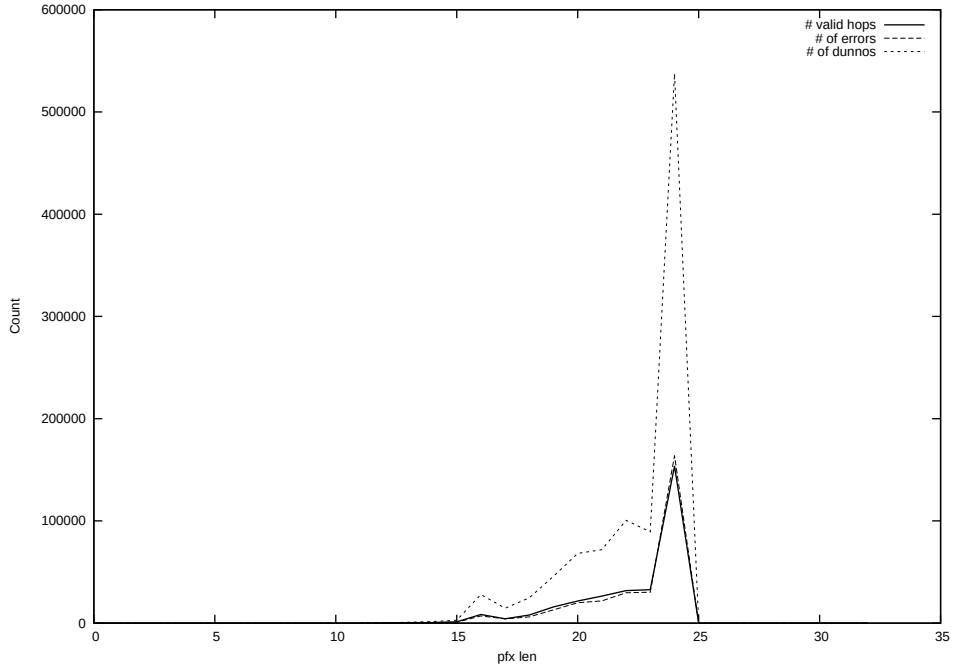
2012-11-03



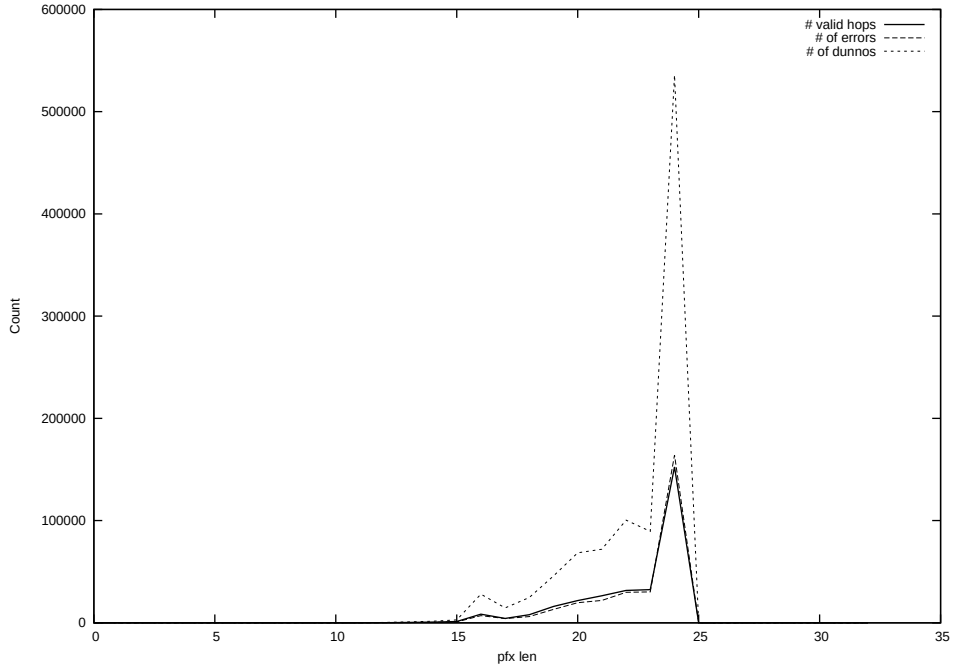
2012-11-04



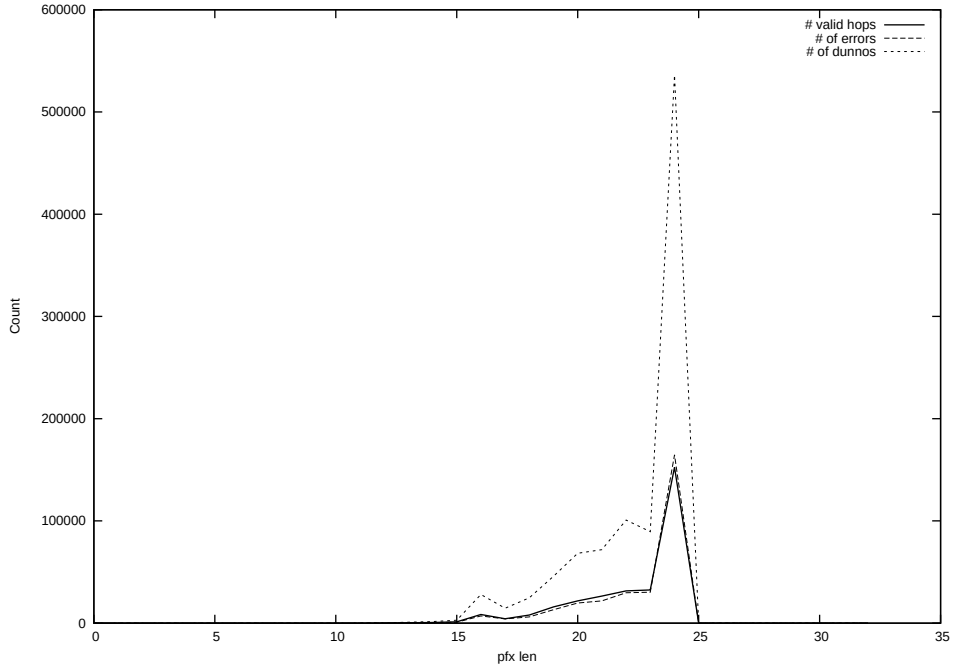
2012-11-05



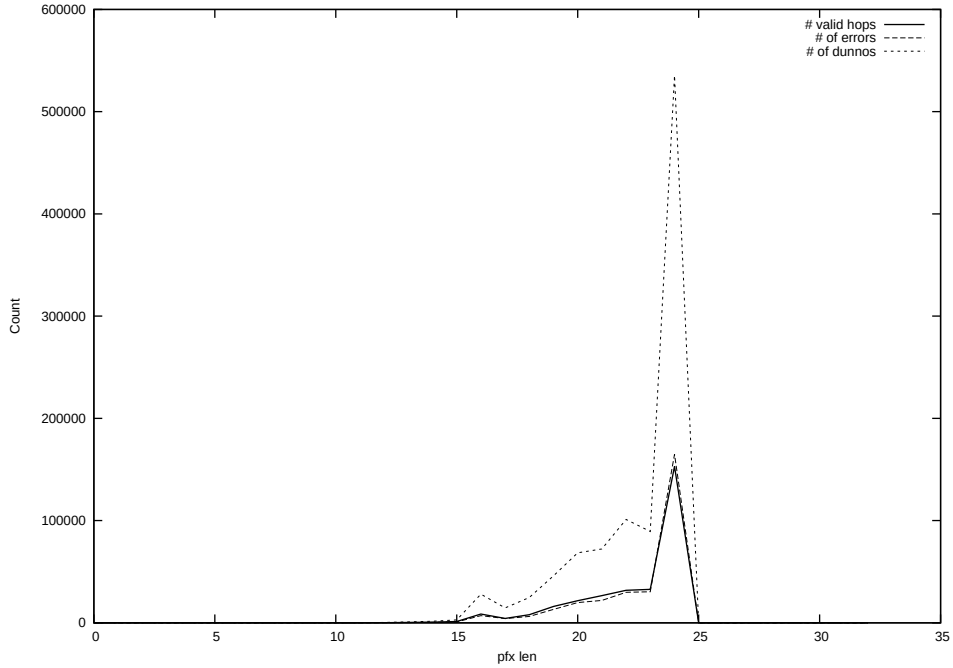
2012-11-06



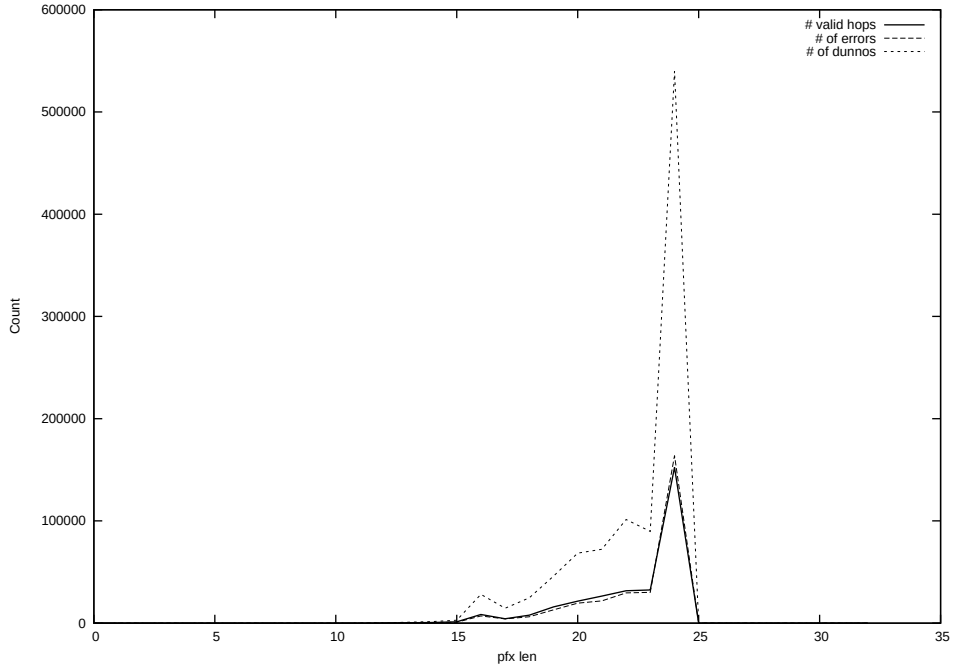
2012-11-07



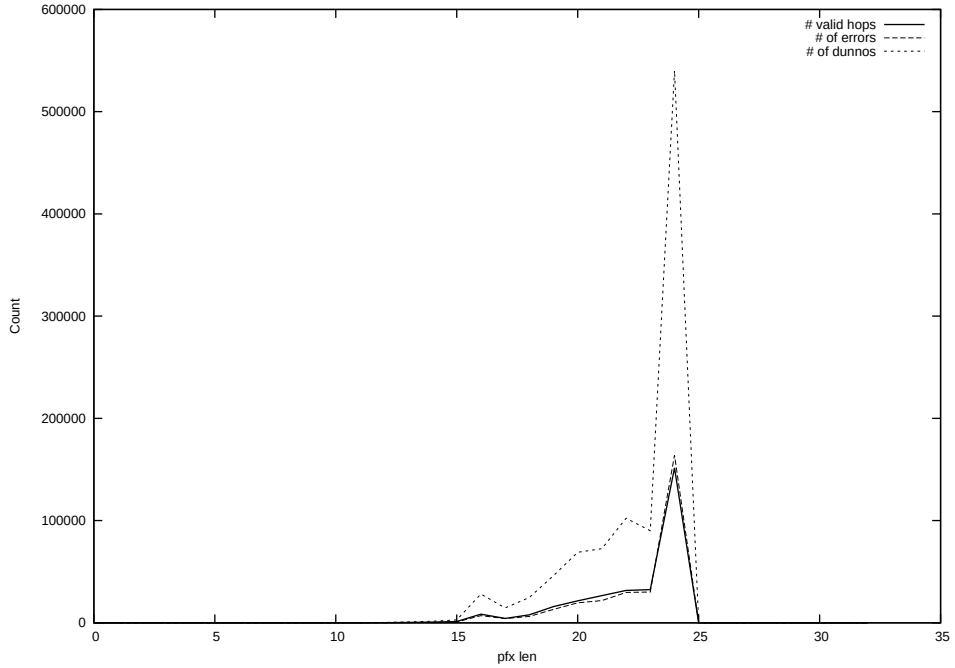
2012-11-08



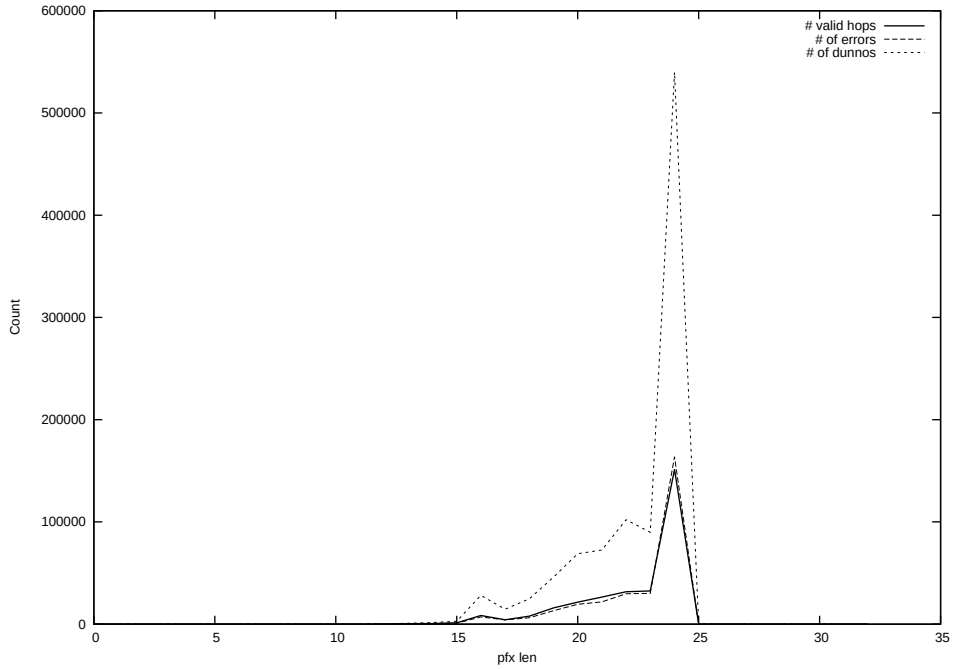
2012-11-09



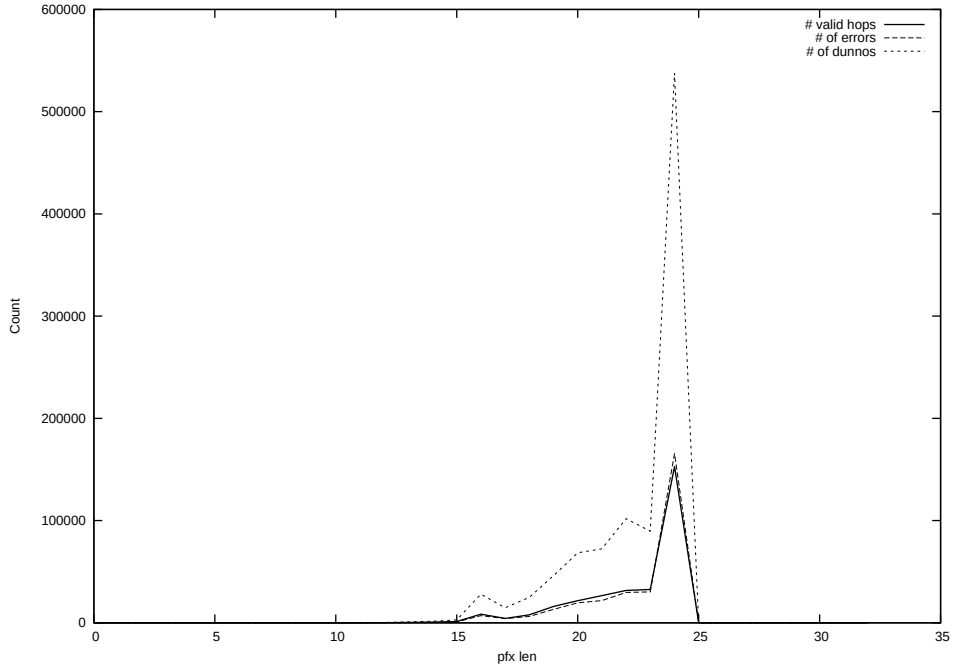
2012-11-10



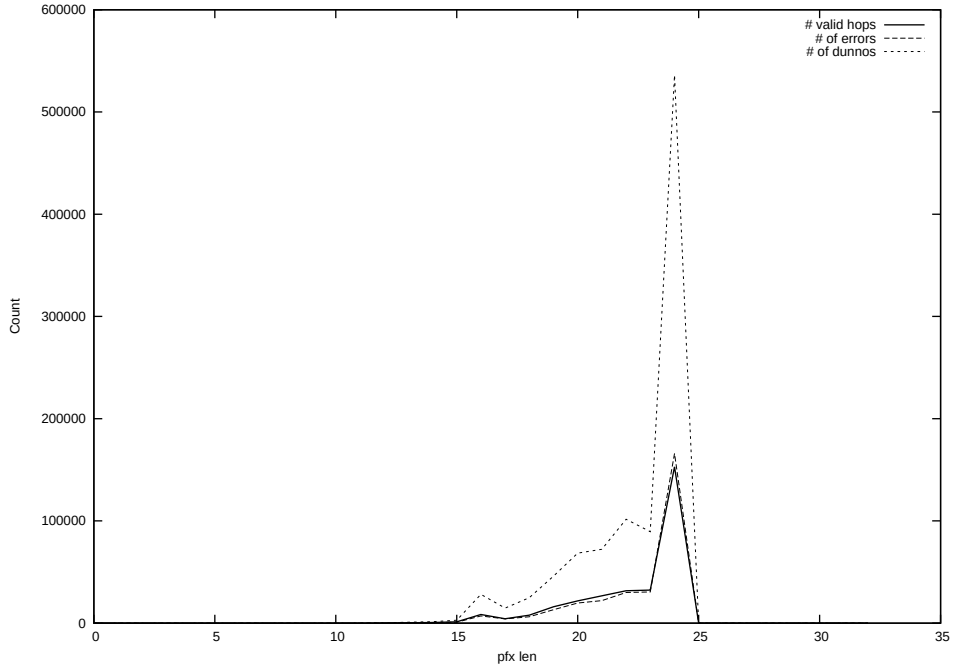
2012-11-11



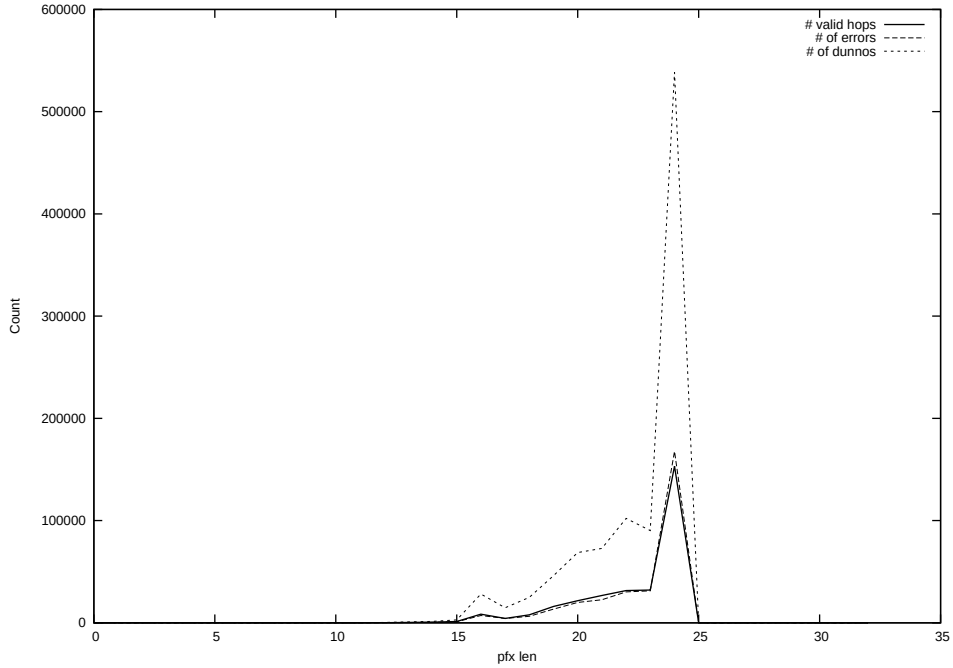
2012-11-12



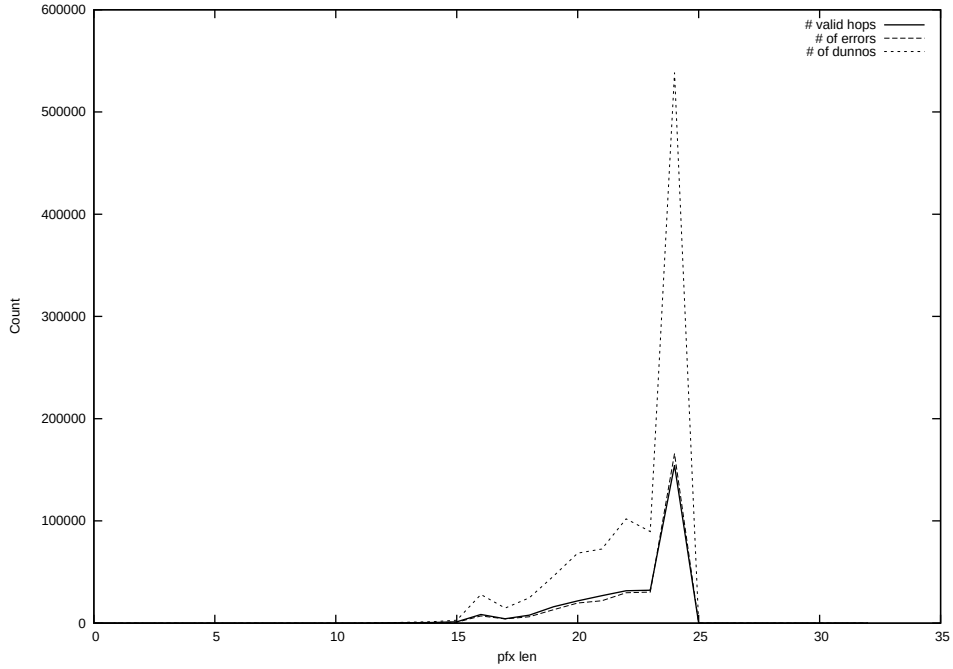
2012-11-13



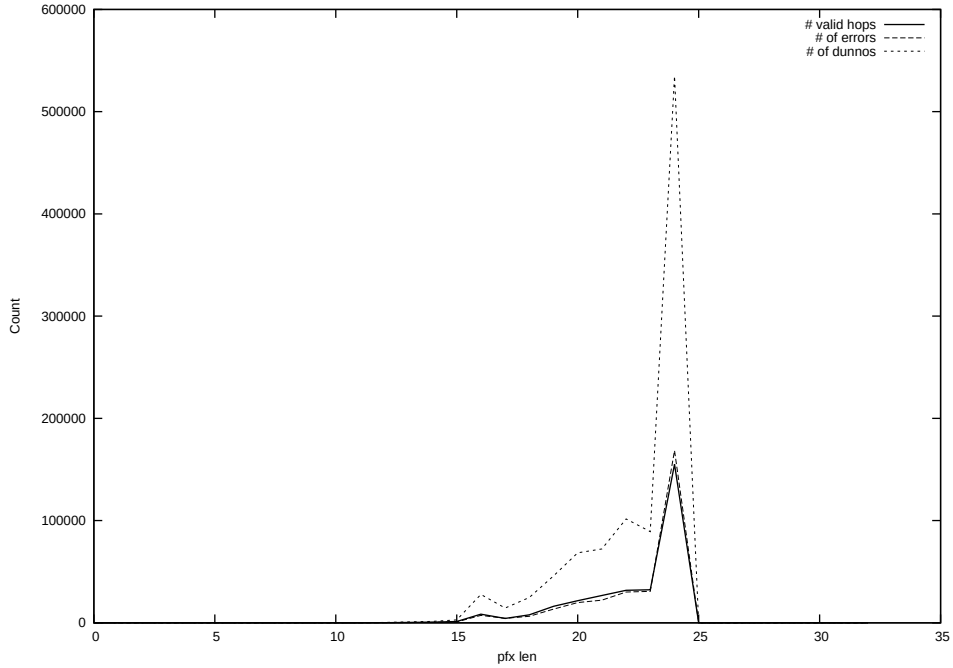
2012-11-14



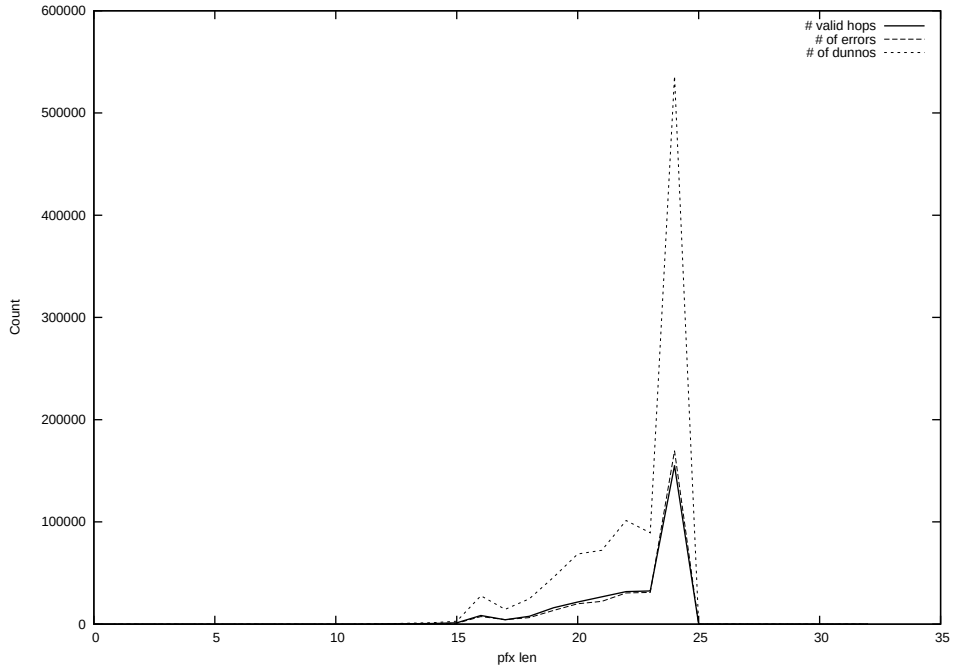
2012-11-15



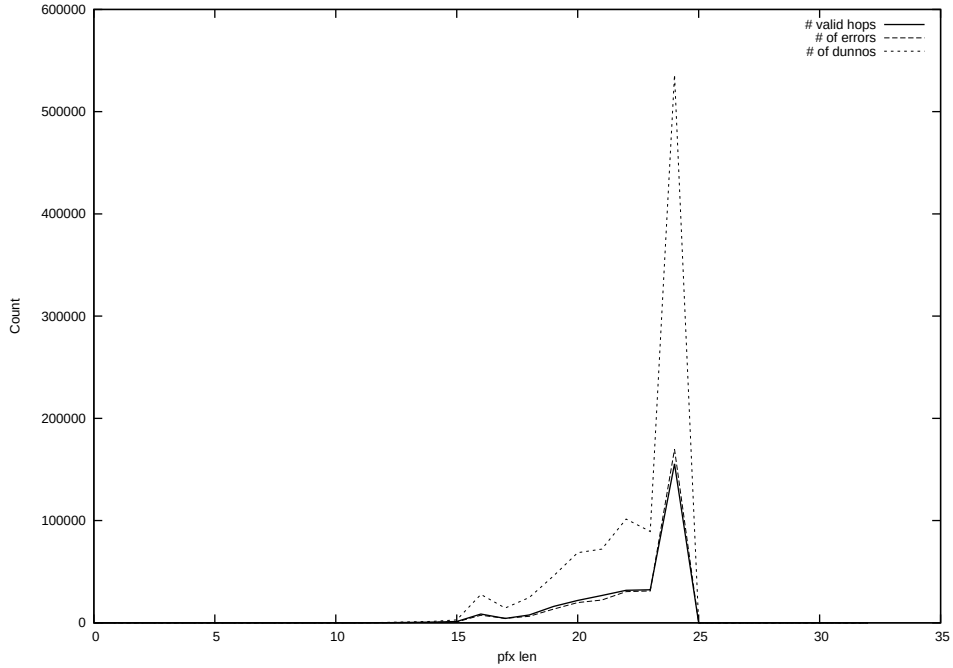
2012-11-16



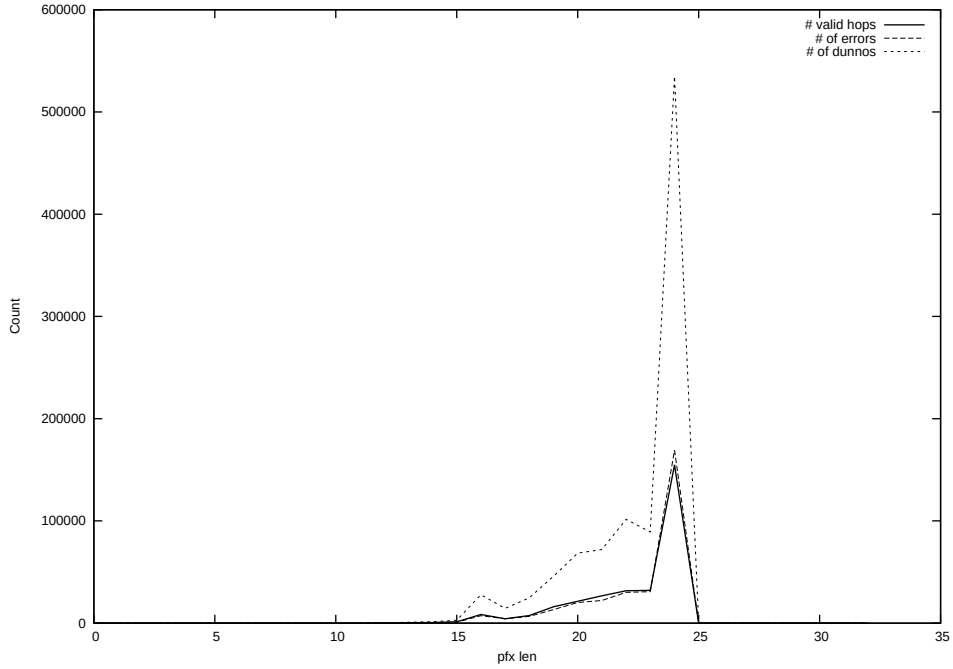
2012-11-17



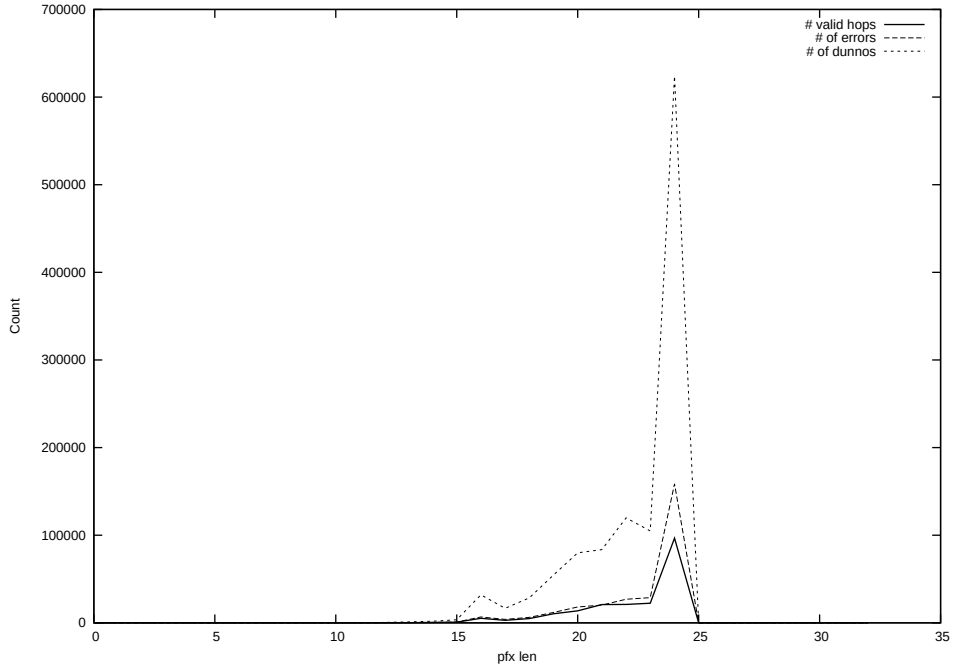
2012-11-18



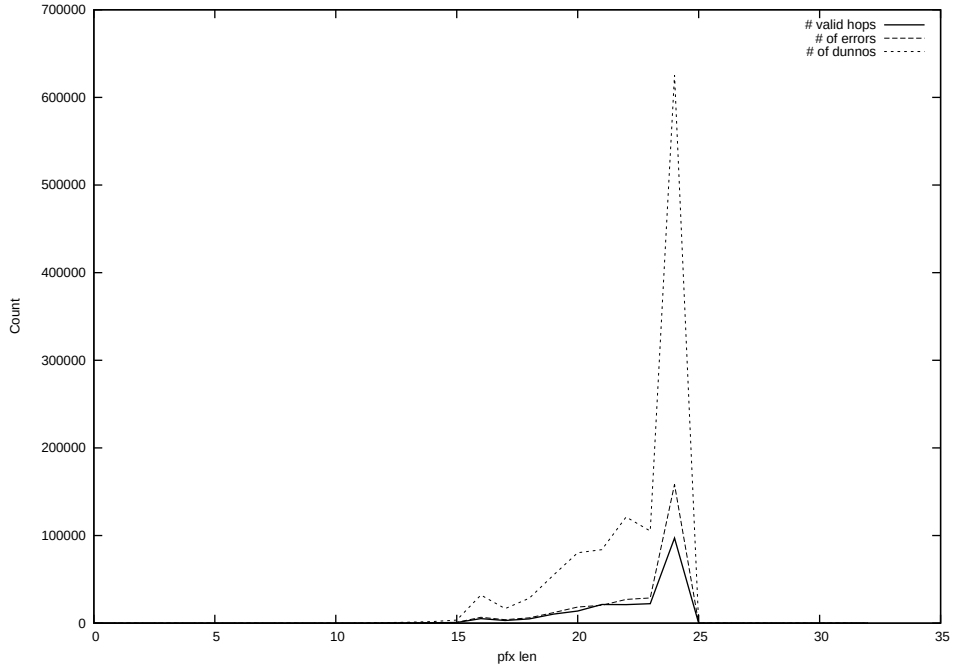
2012-11-19



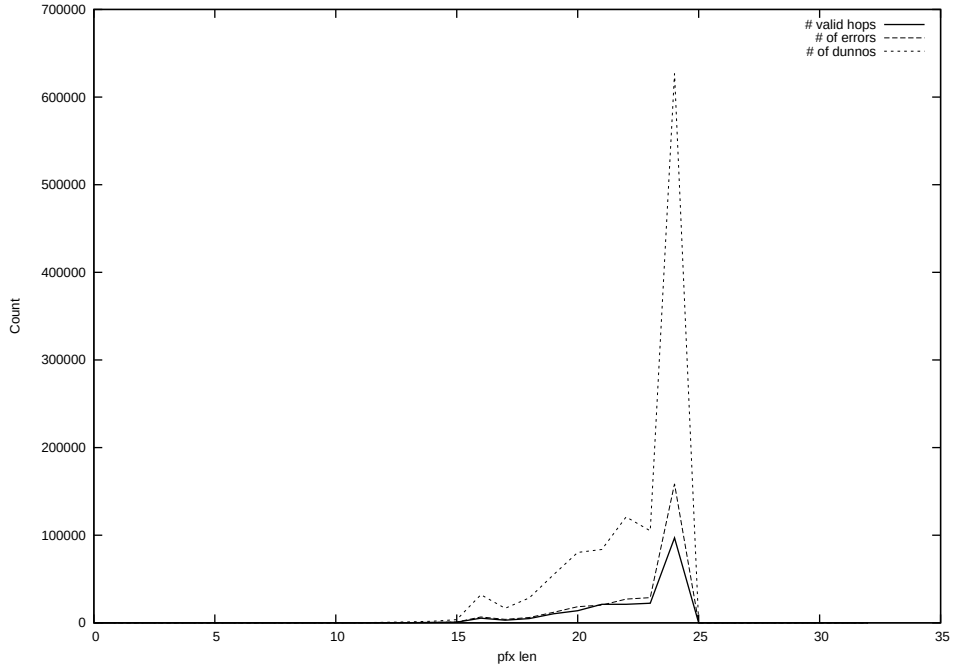
2012-11-20



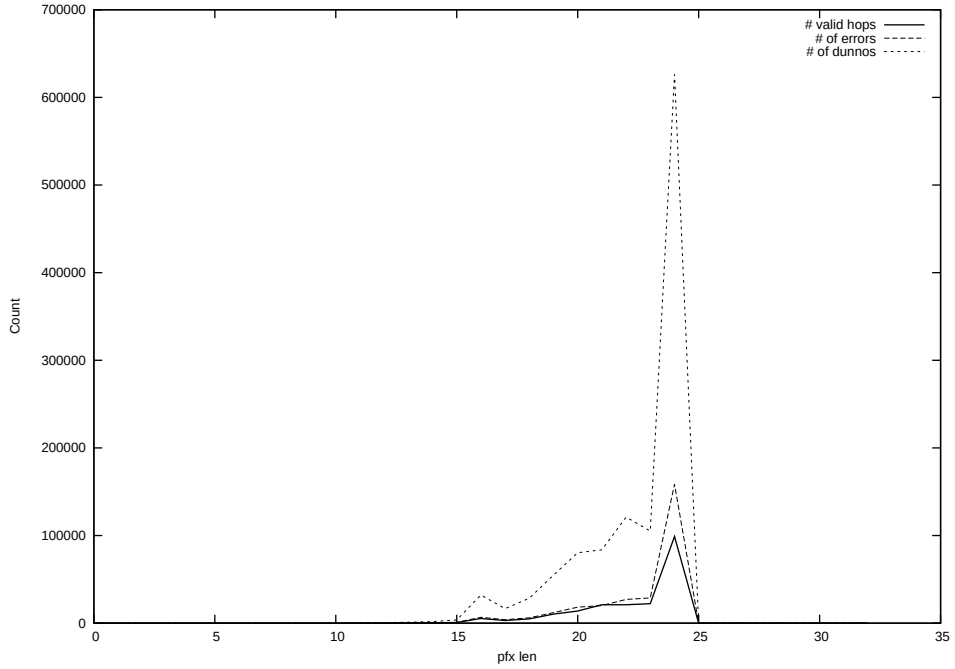
2012-11-21



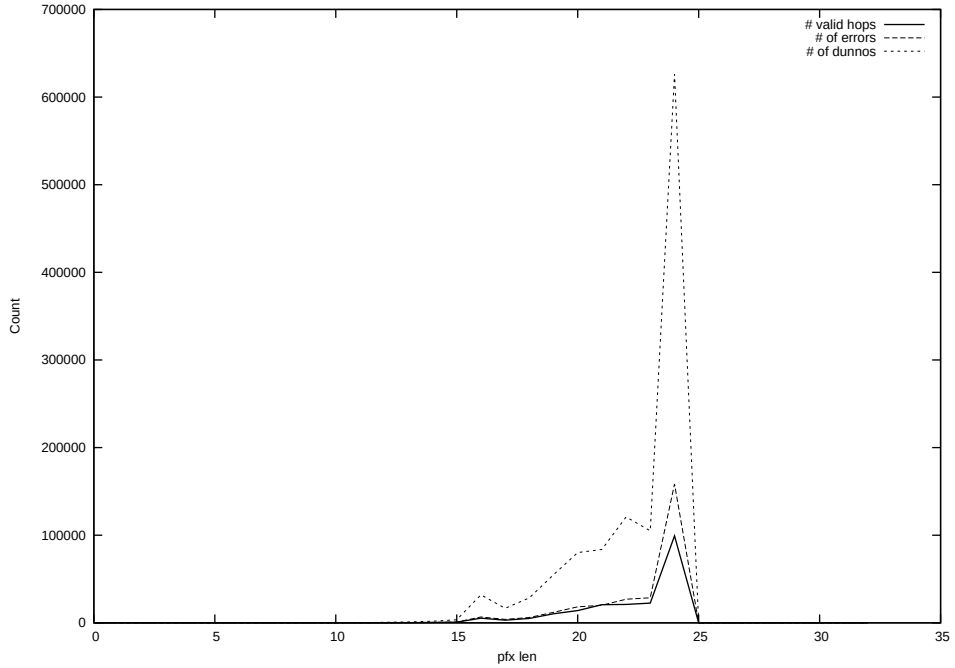
2012-11-22



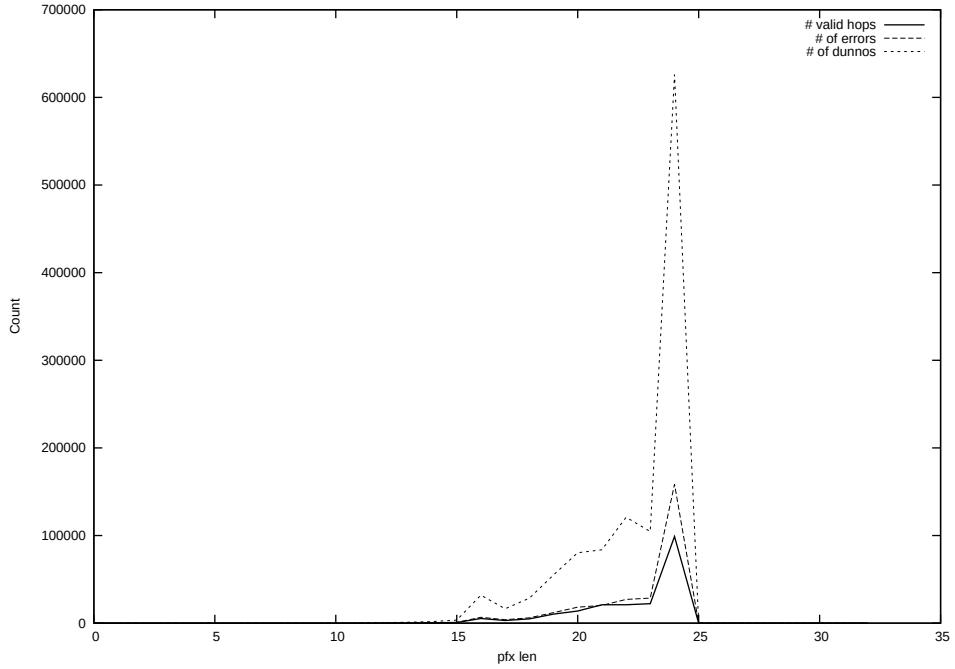
2012-11-23



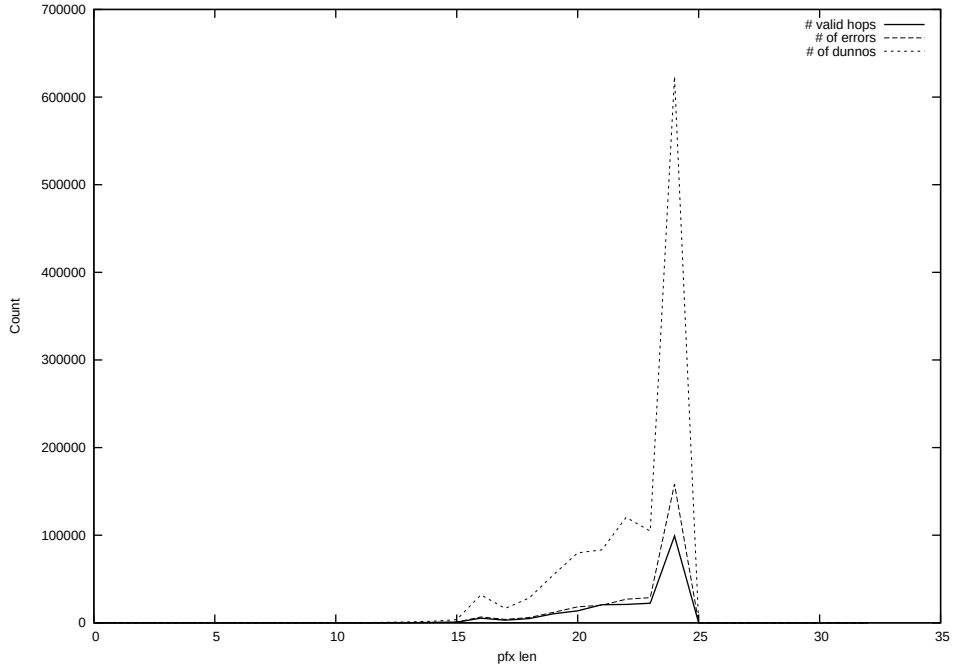
2012-11-24



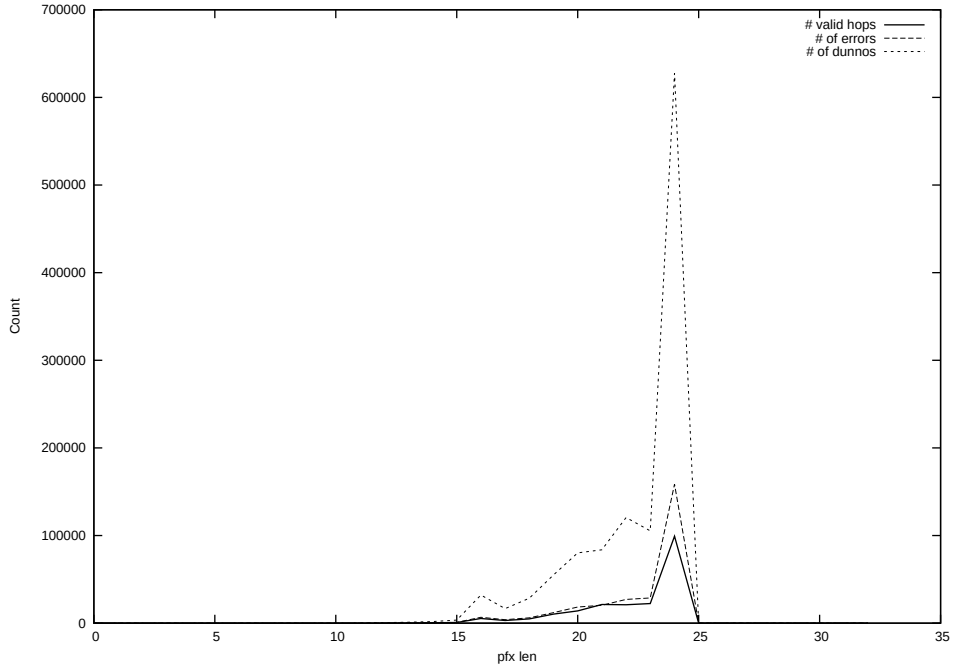
2012-11-25



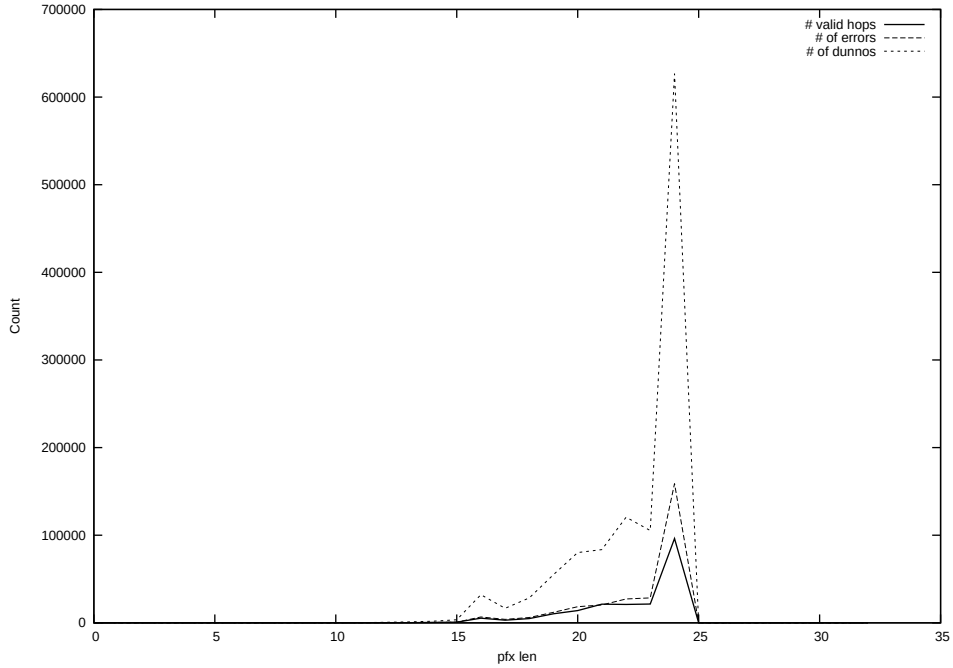
2012-11-26



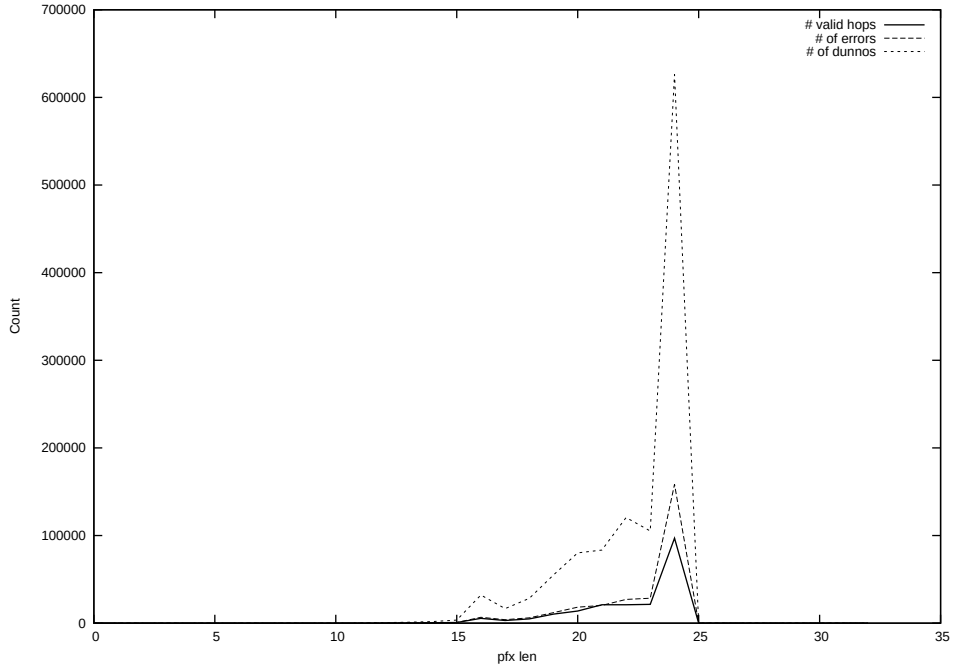
2012-11-27



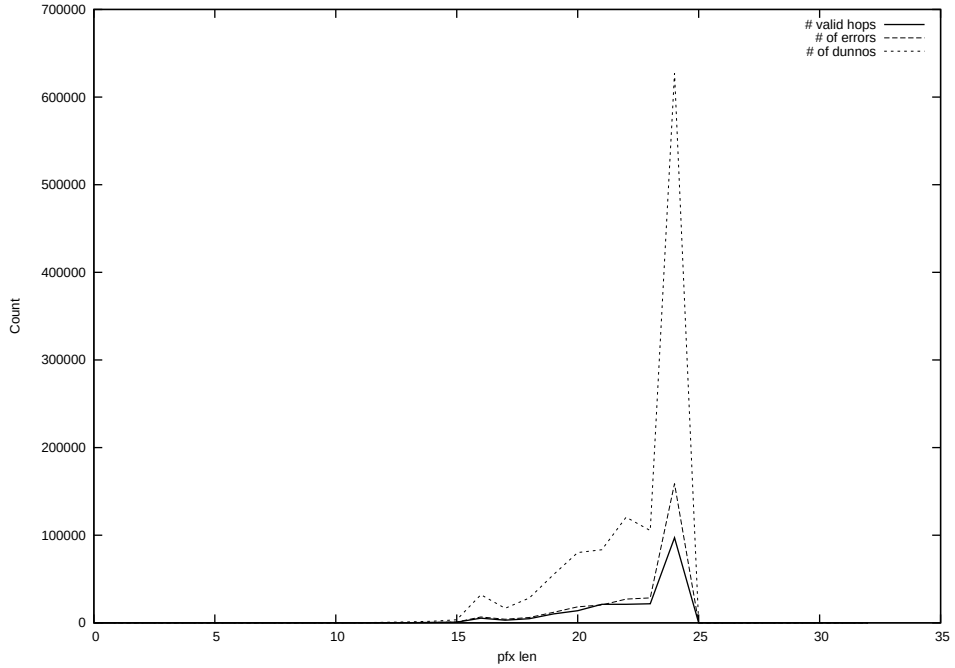
2012-11-28



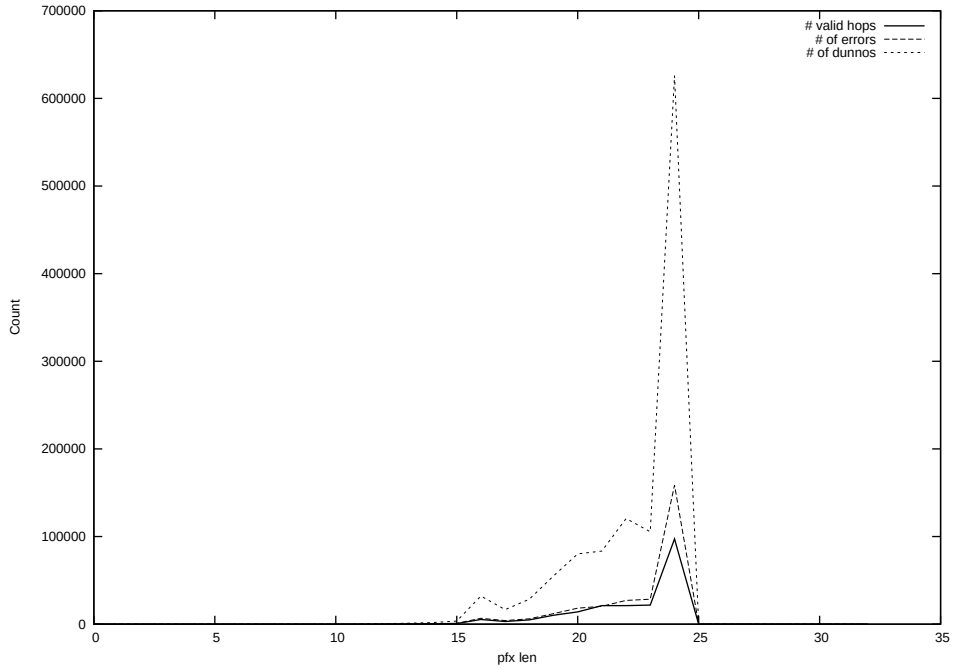
2012-11-29



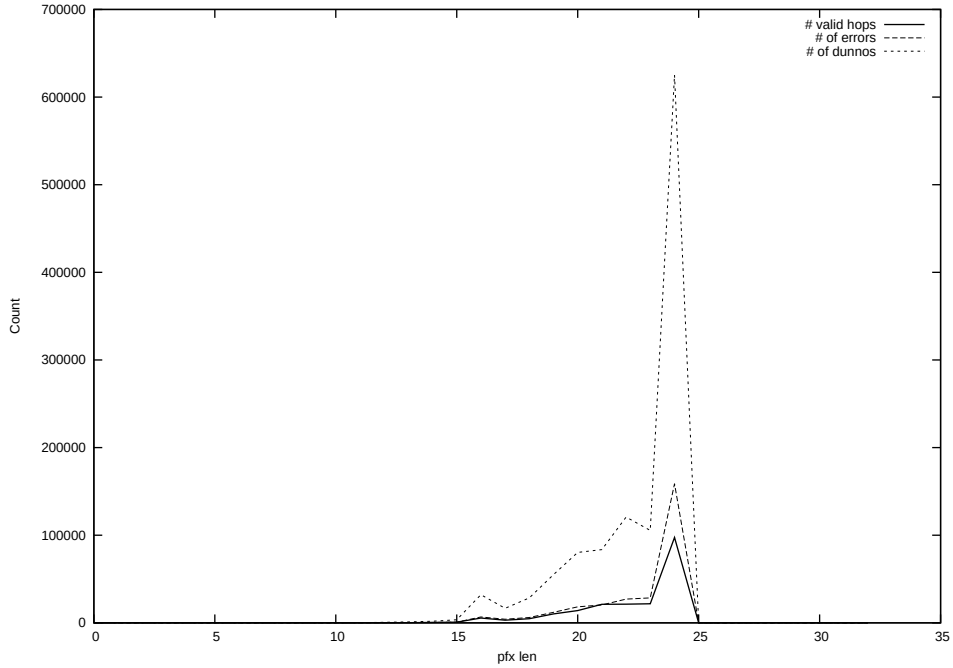
2012-11-30



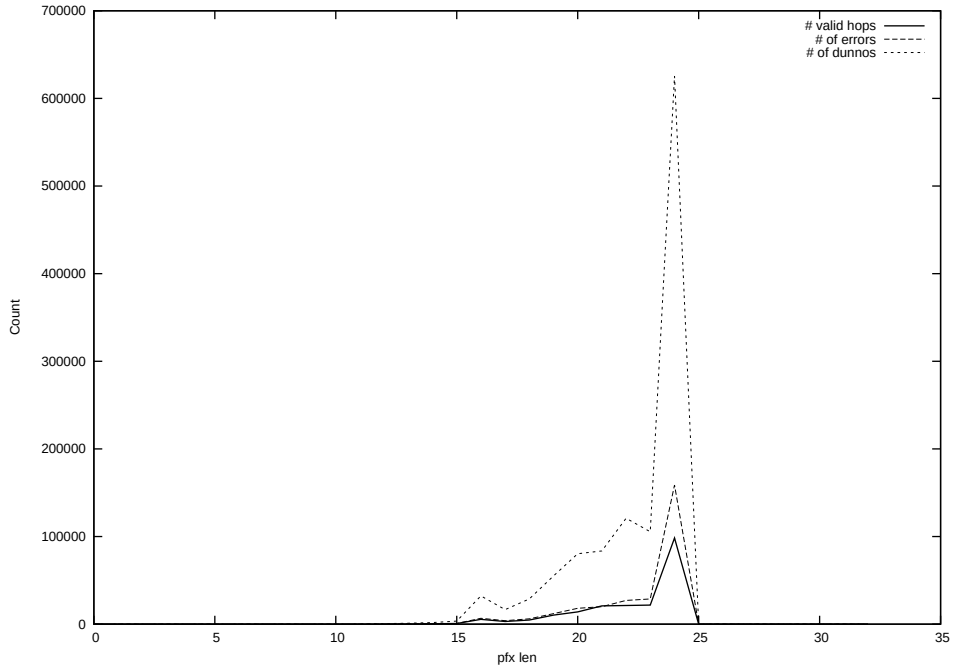
2012-12-01



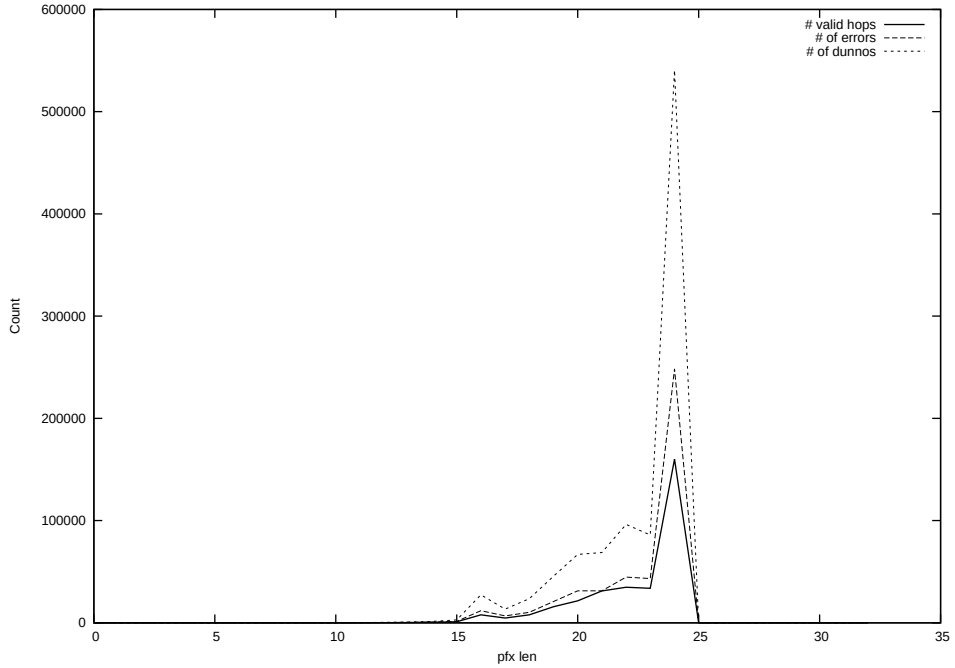
2012-12-02



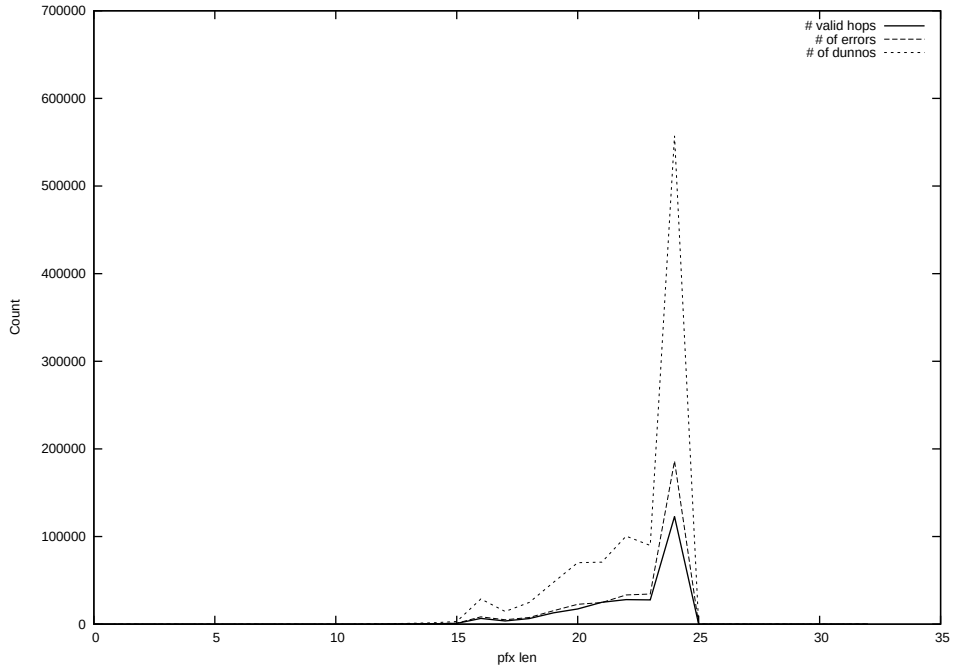
2012-12-03



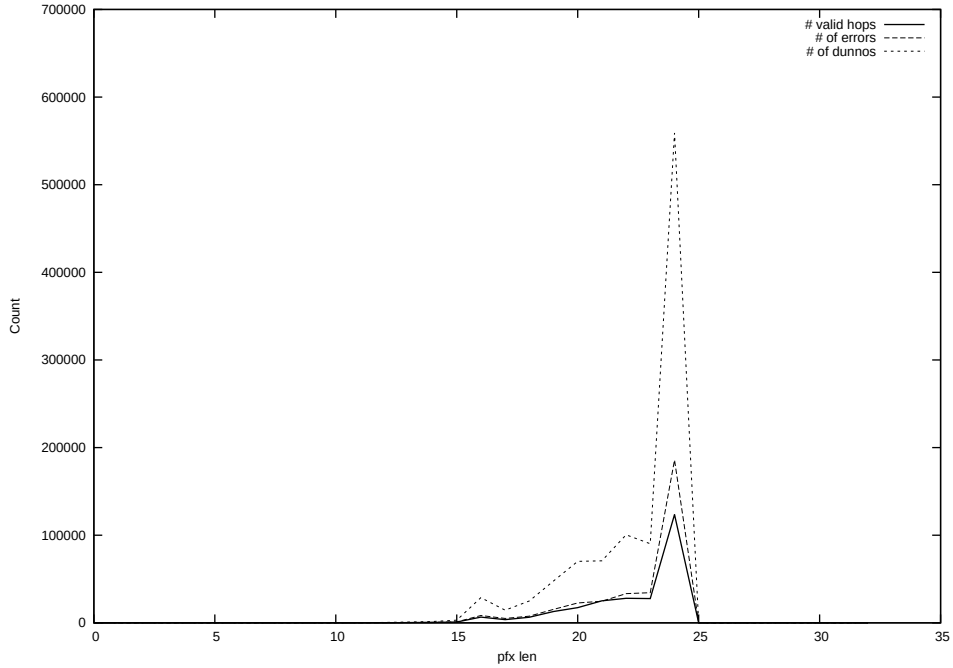
2012-12-04



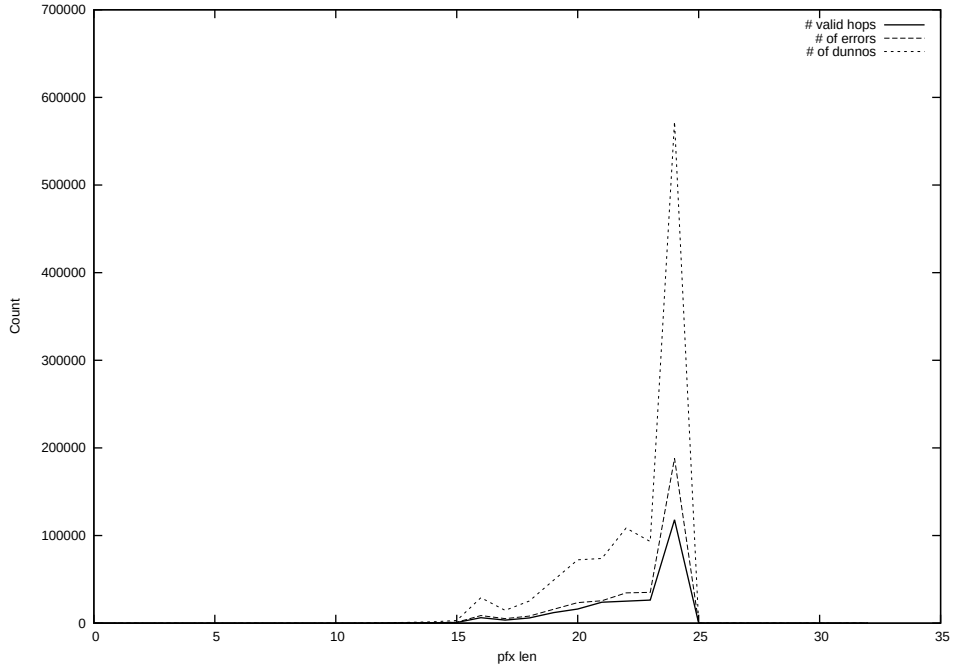
2012-12-05



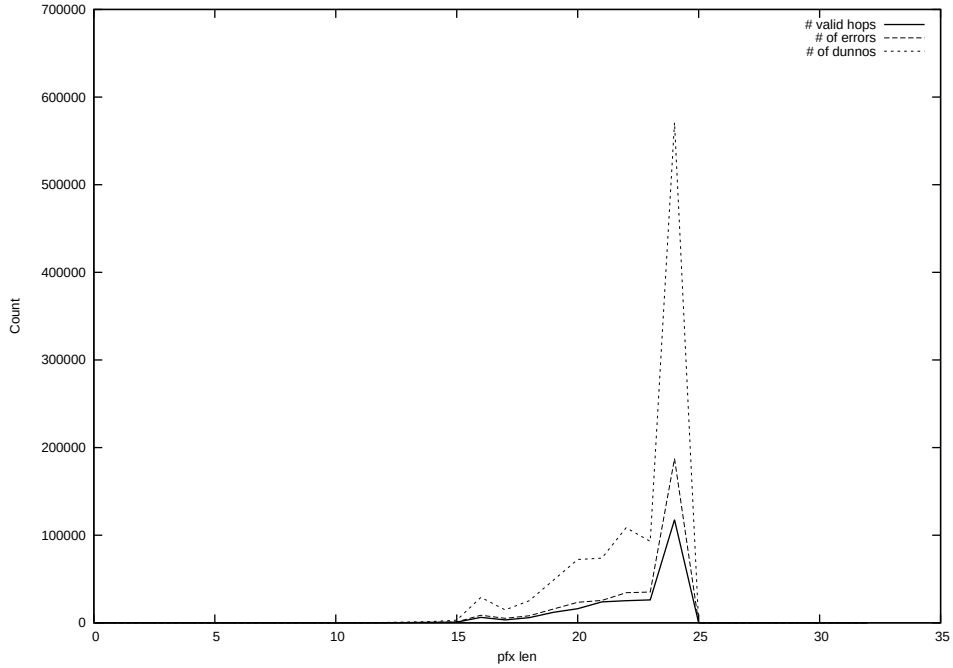
2012-12-06



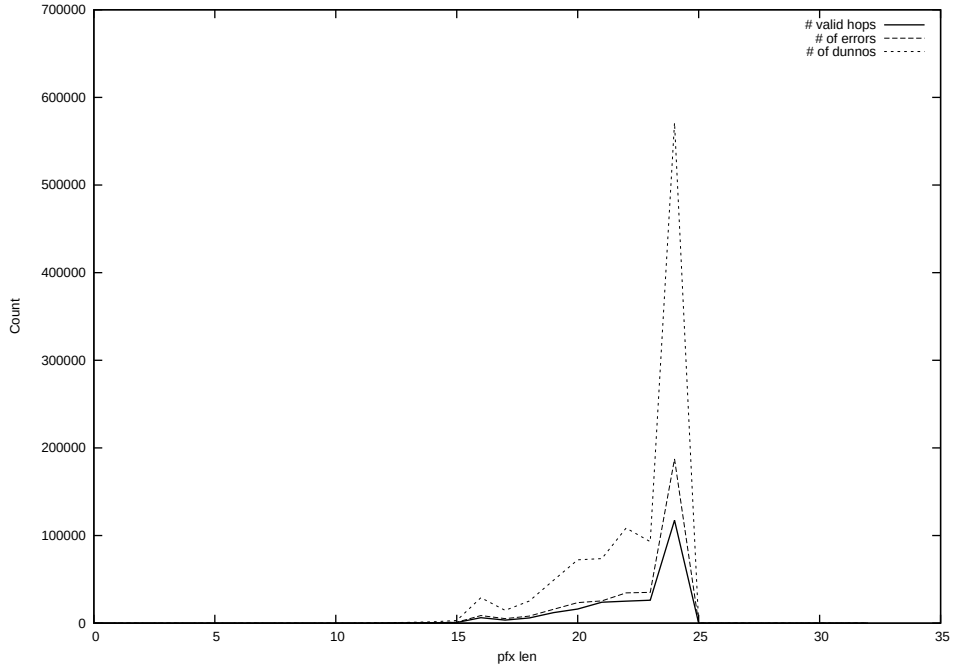
2012-12-07



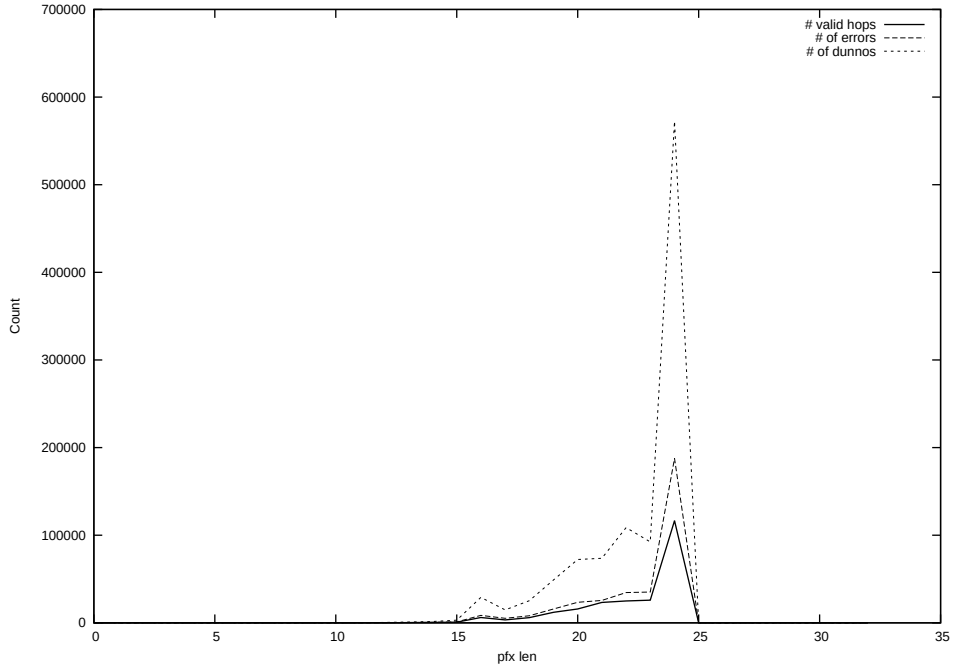
2012-12-08



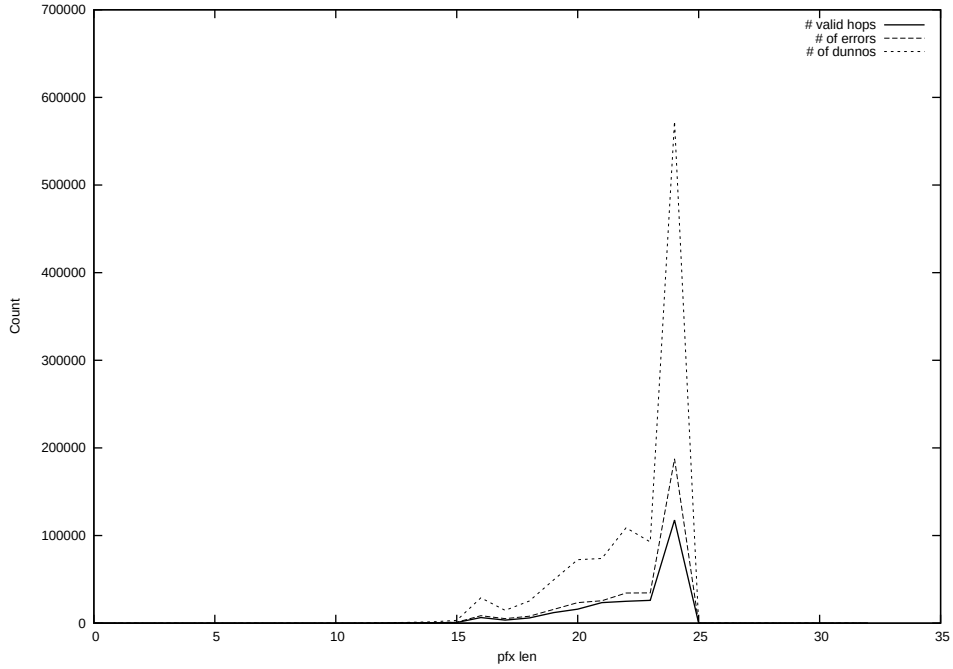
2012-12-09



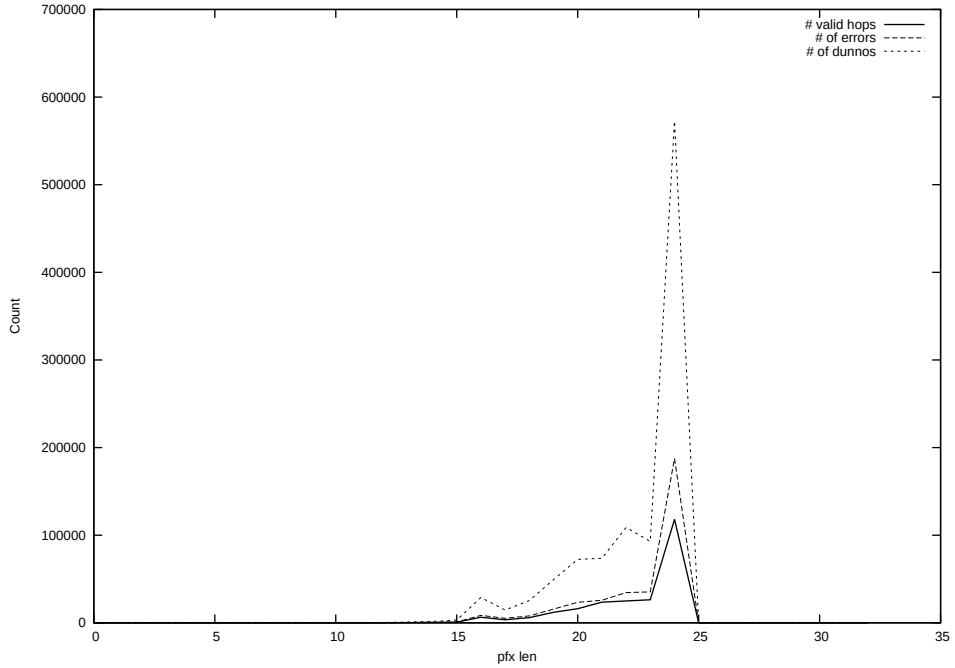
2012-12-10



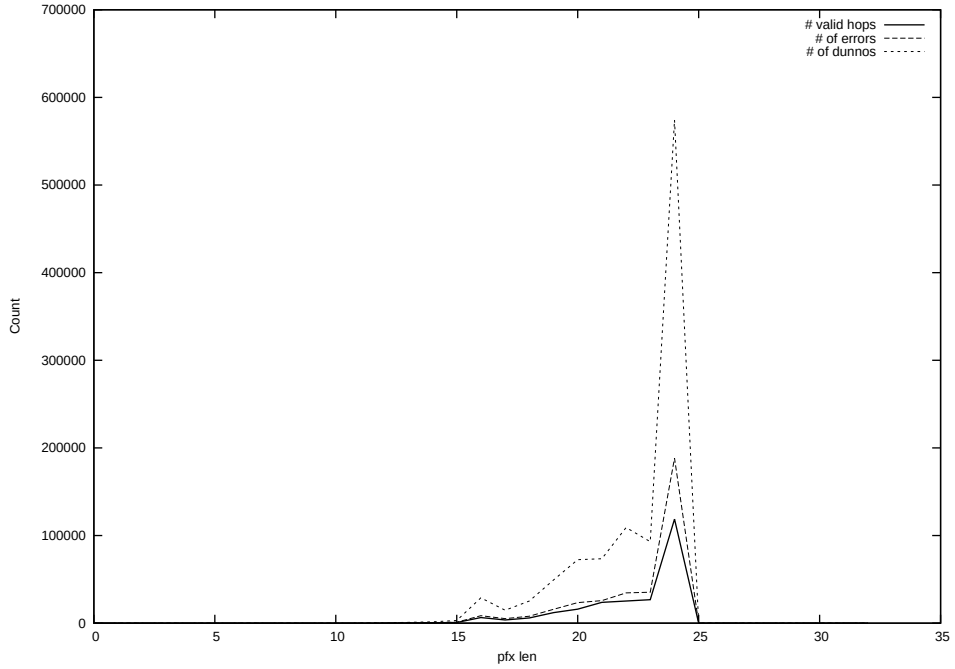
2012-12-11



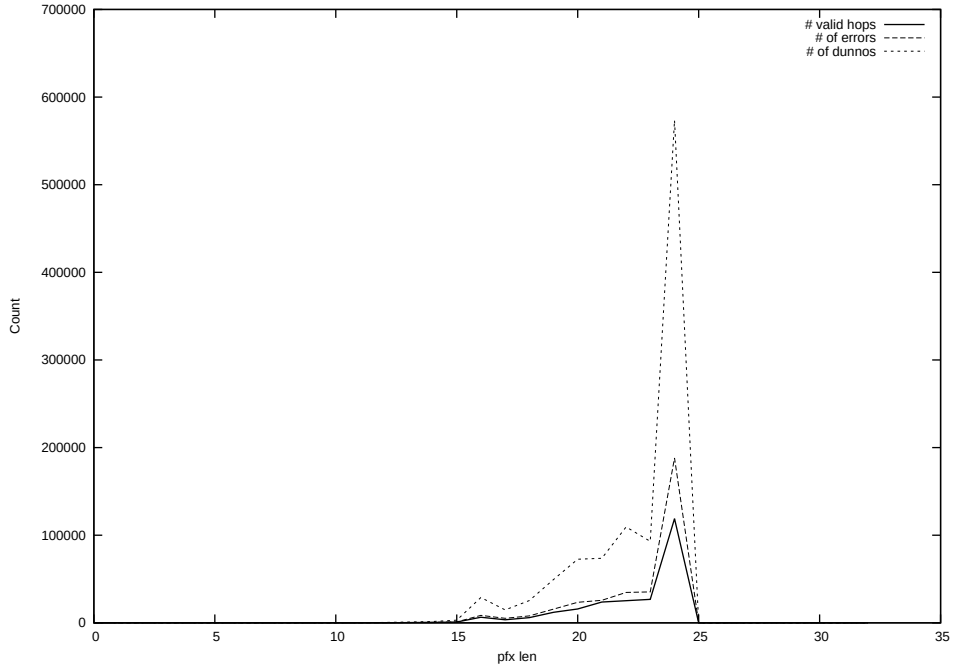
2012-12-12



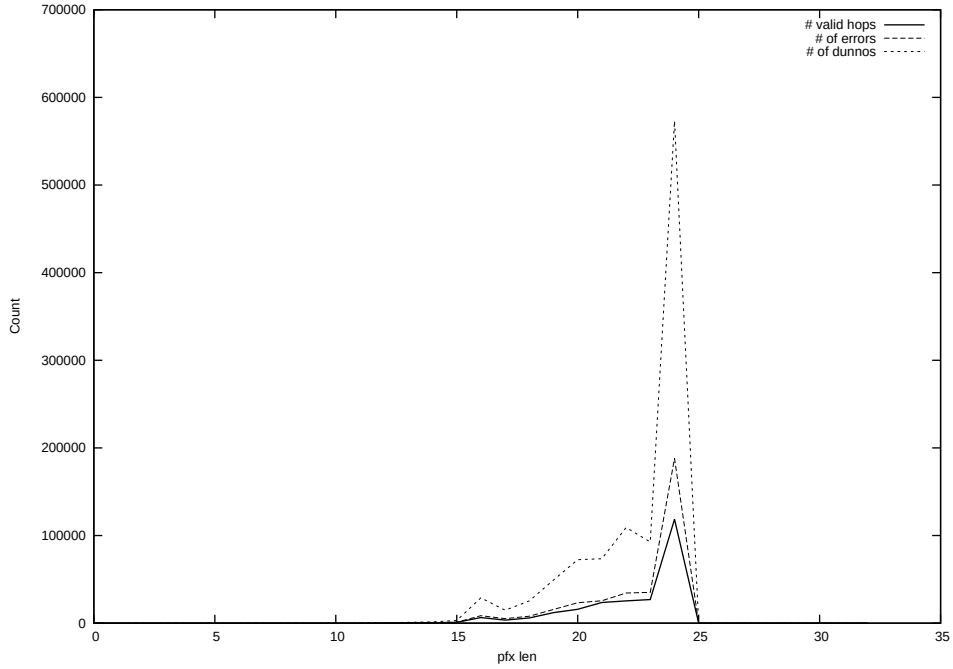
2012-12-13



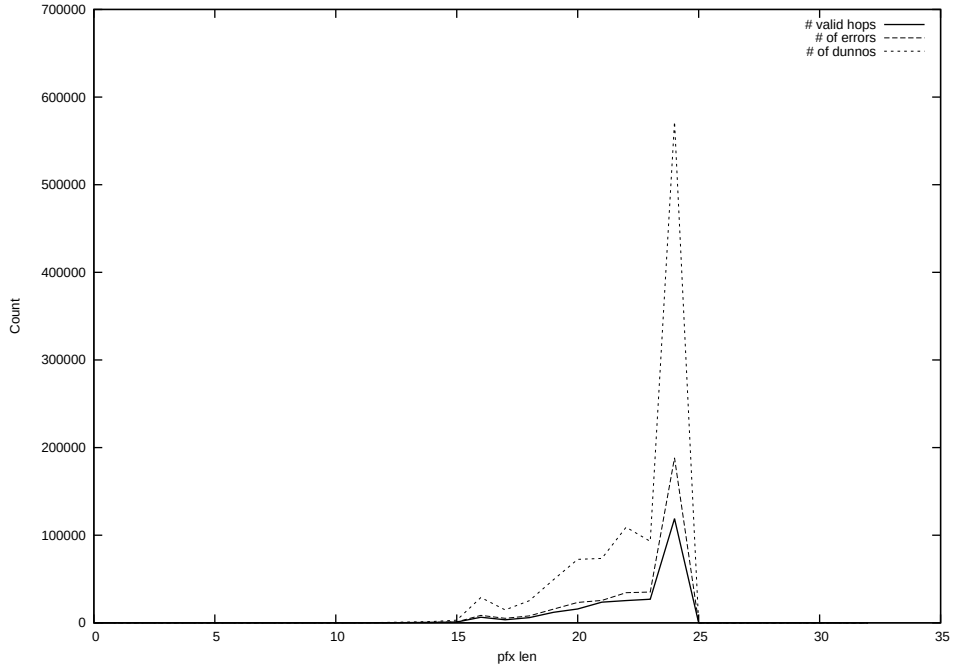
2012-12-14



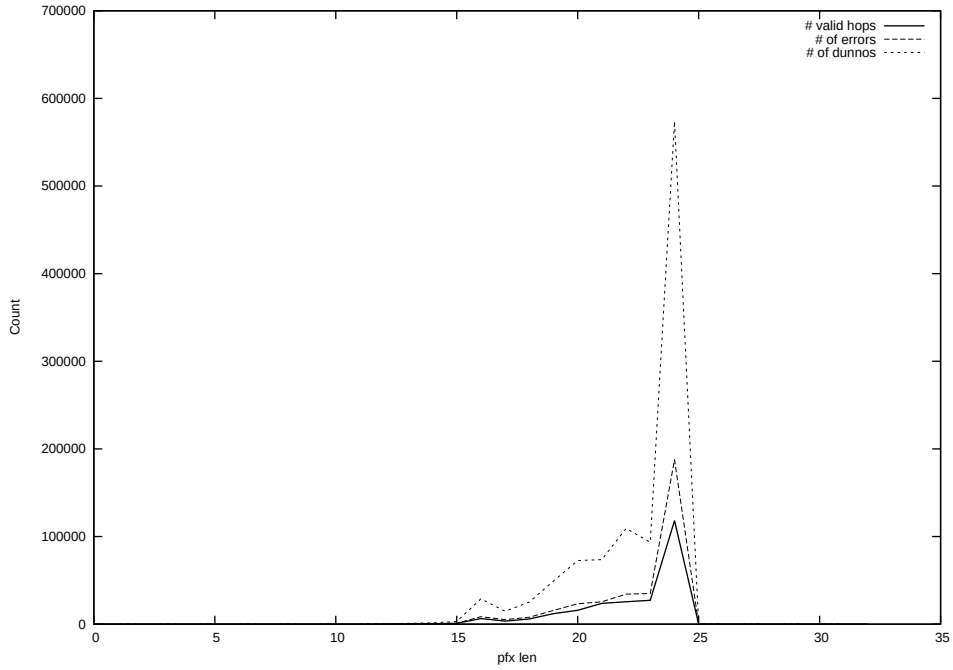
2012-12-15



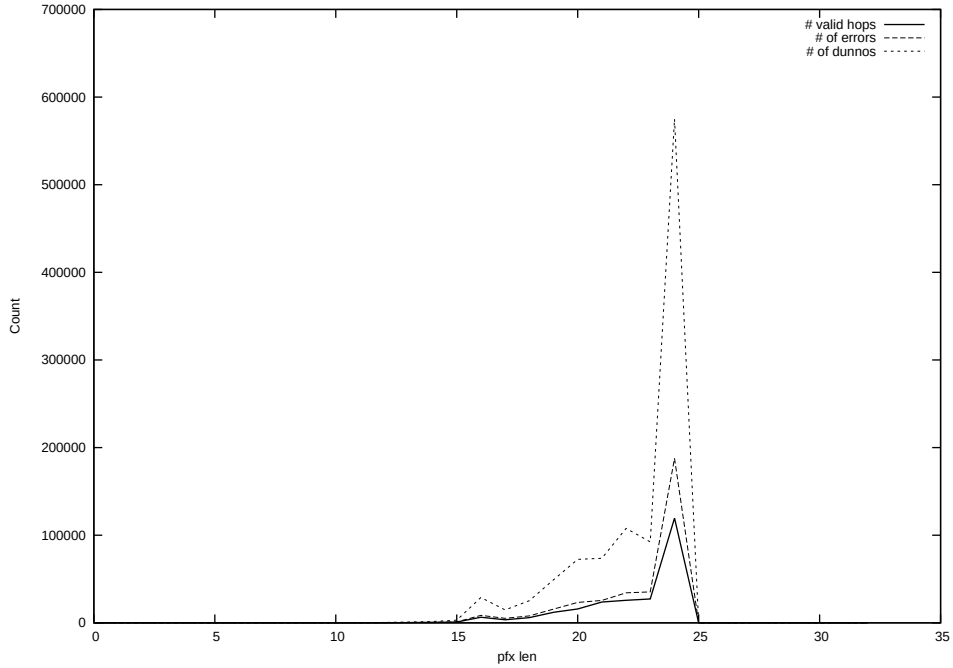
2012-12-16



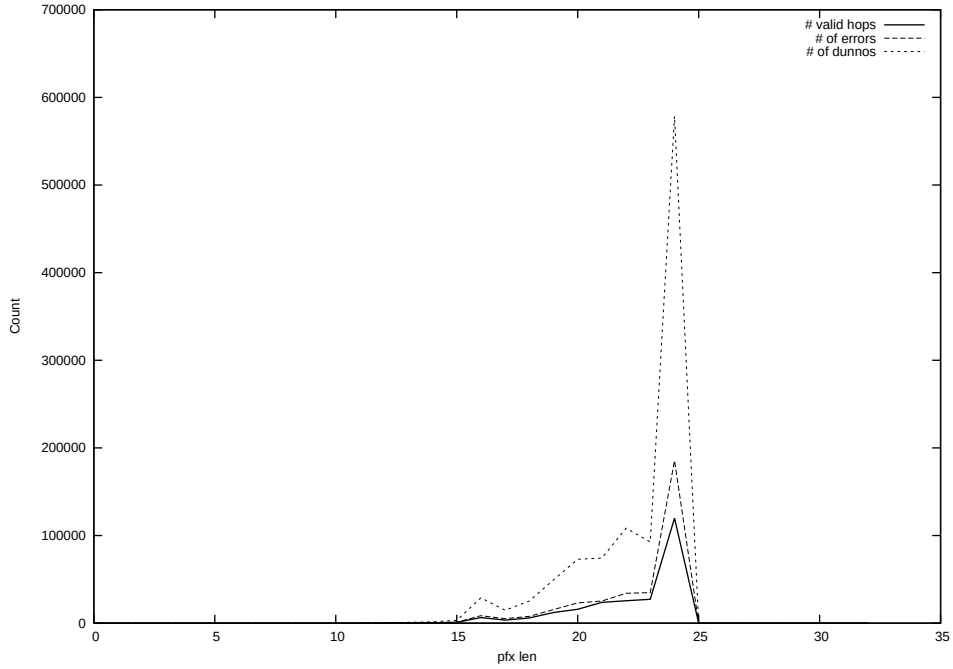
2012-12-17



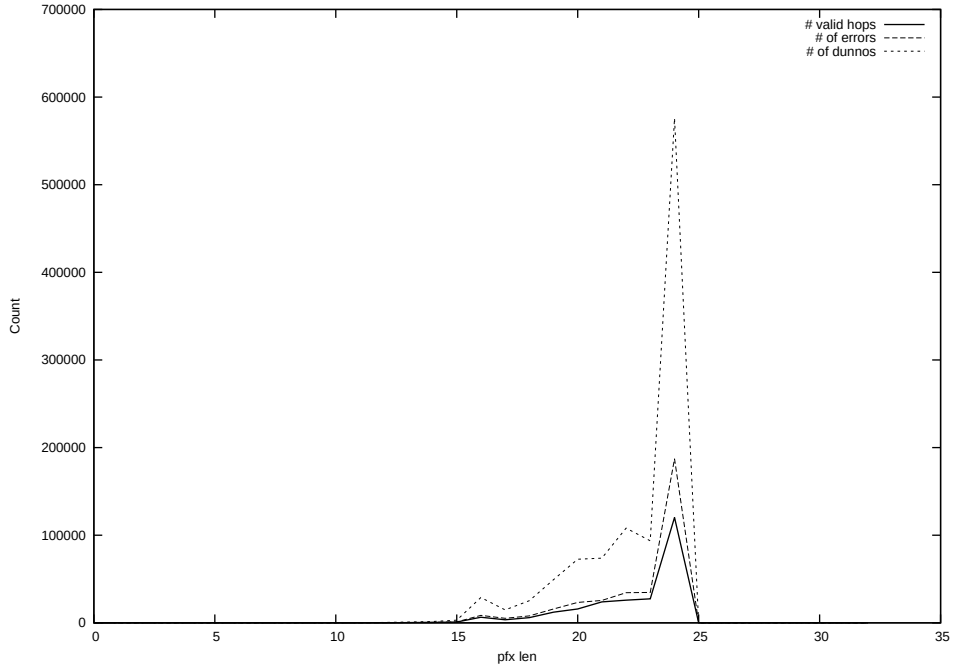
2012-12-18



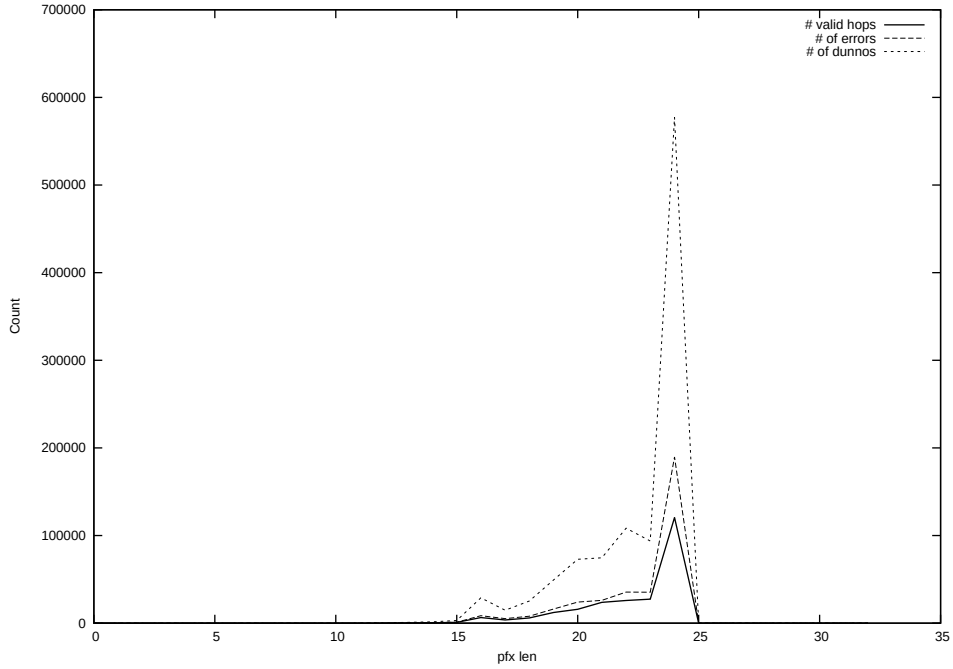
2012-12-19



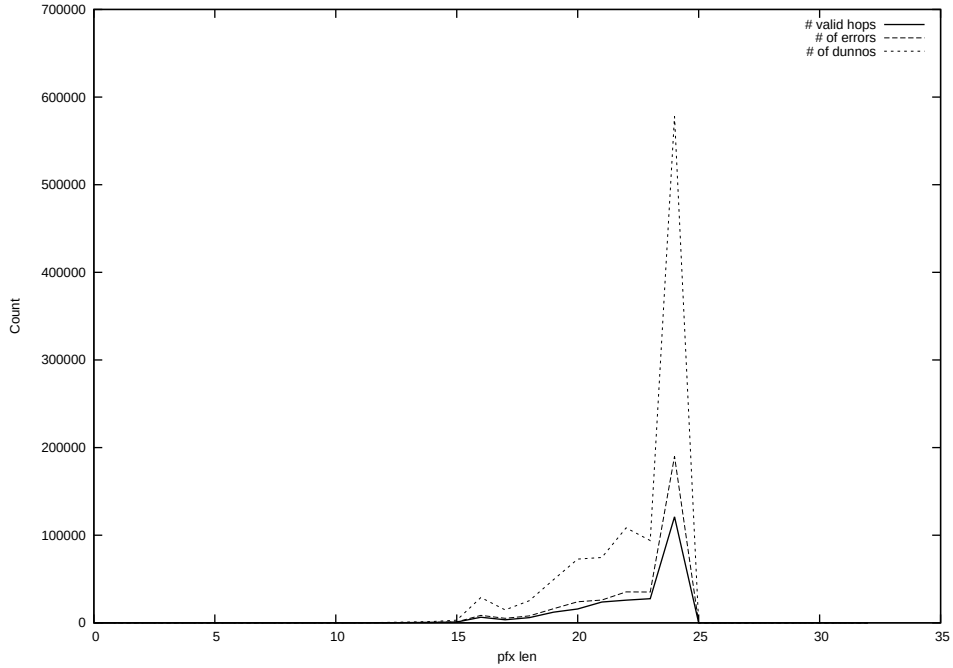
2012-12-20



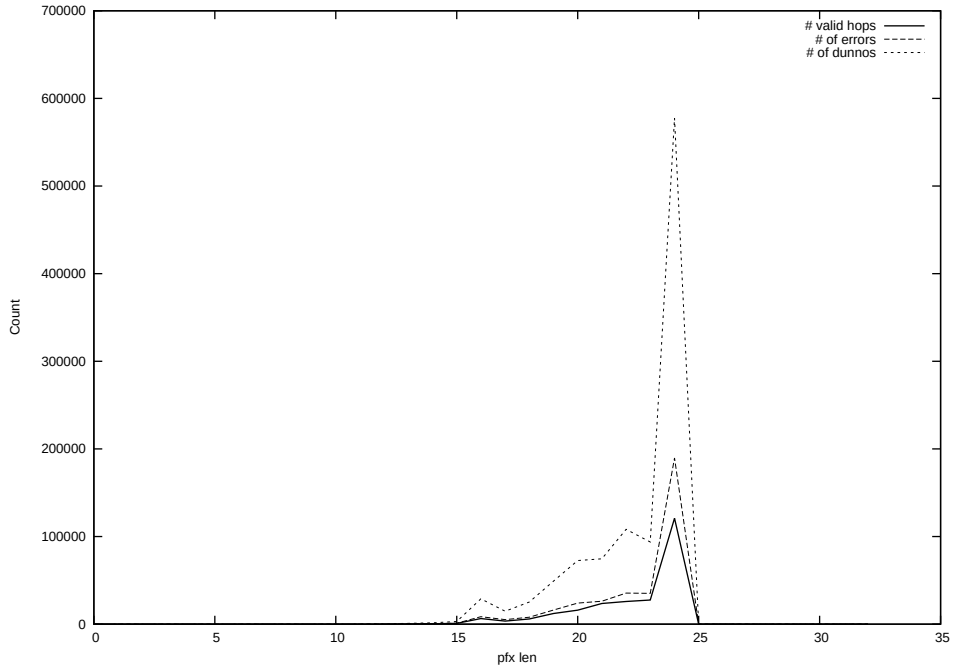
2012-12-21



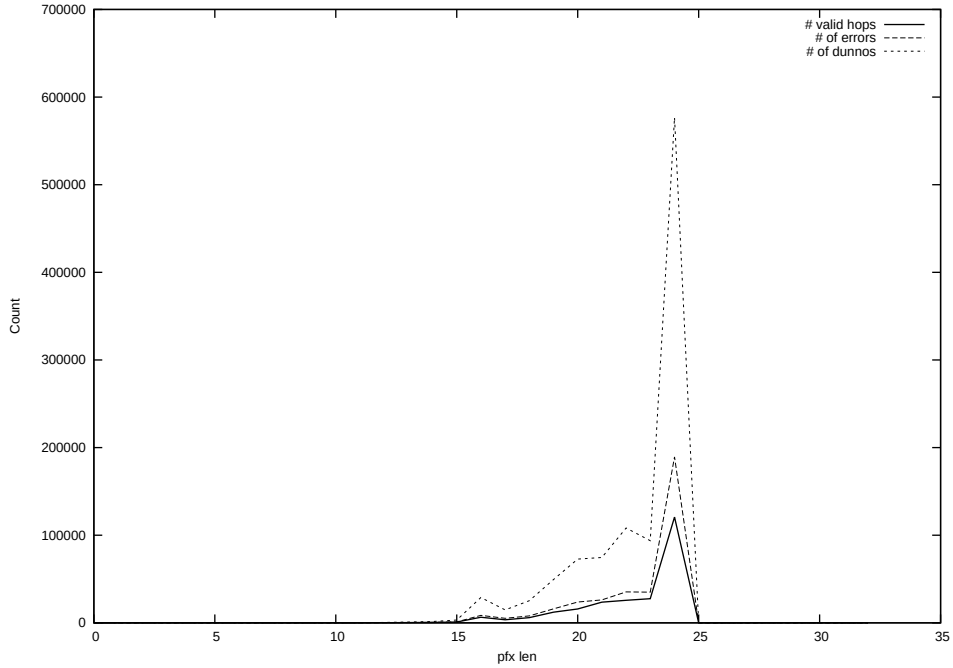
2012-12-22



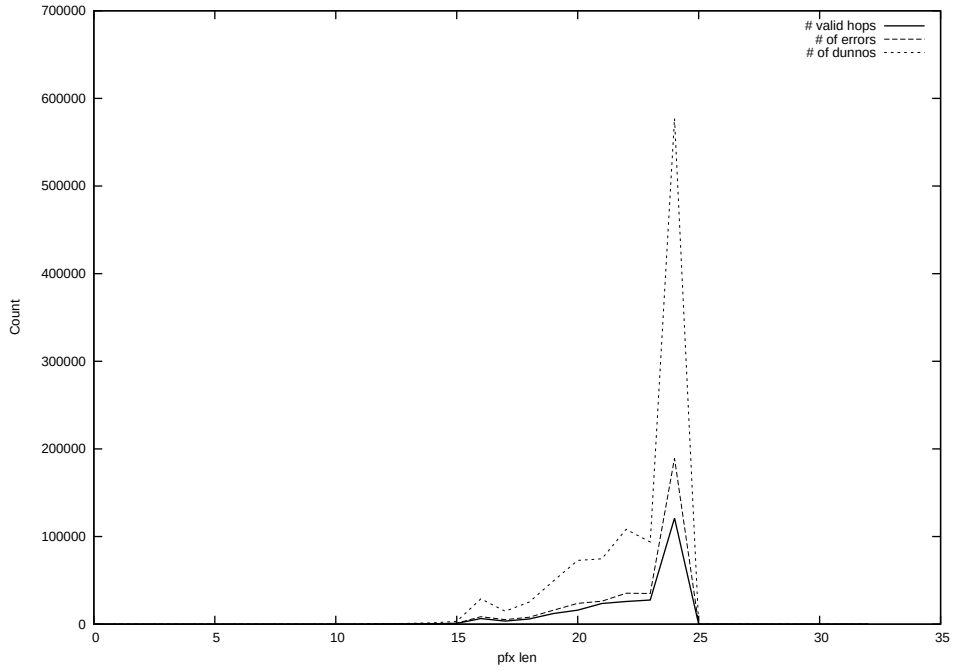
2012-12-23



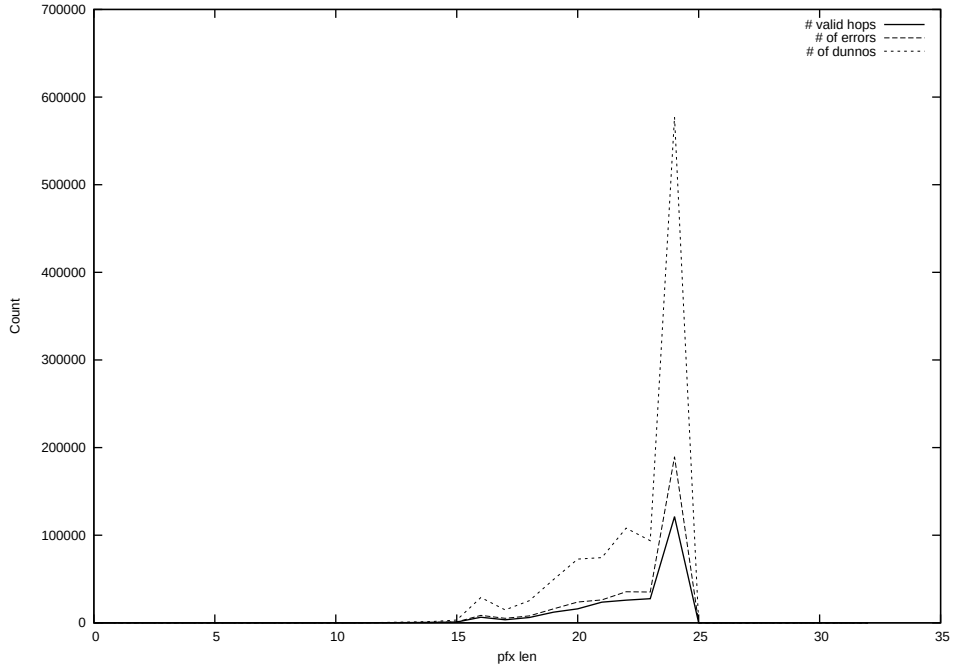
2012-12-24



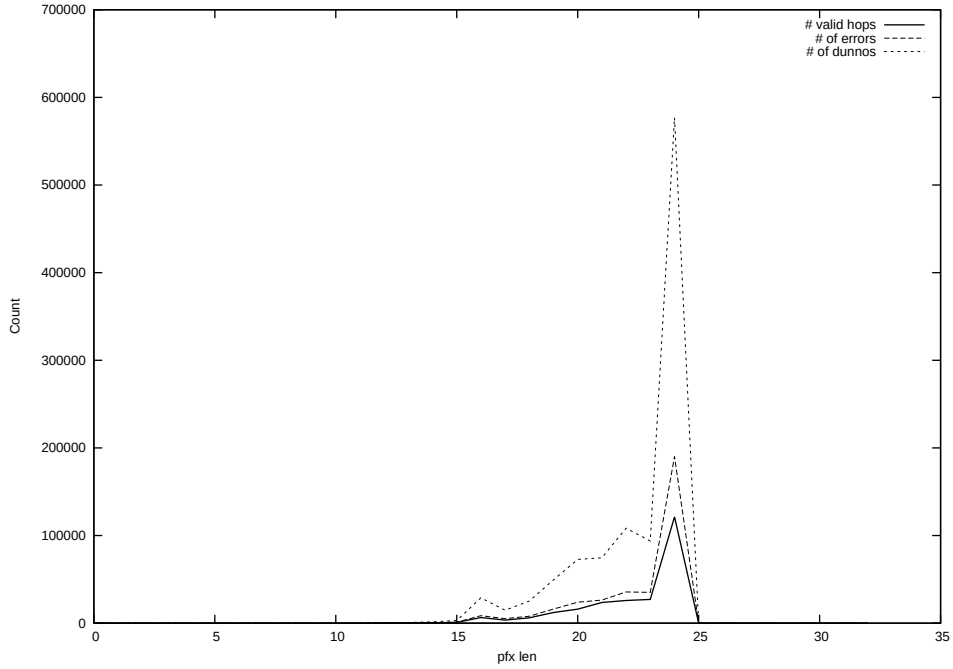
2012-12-25



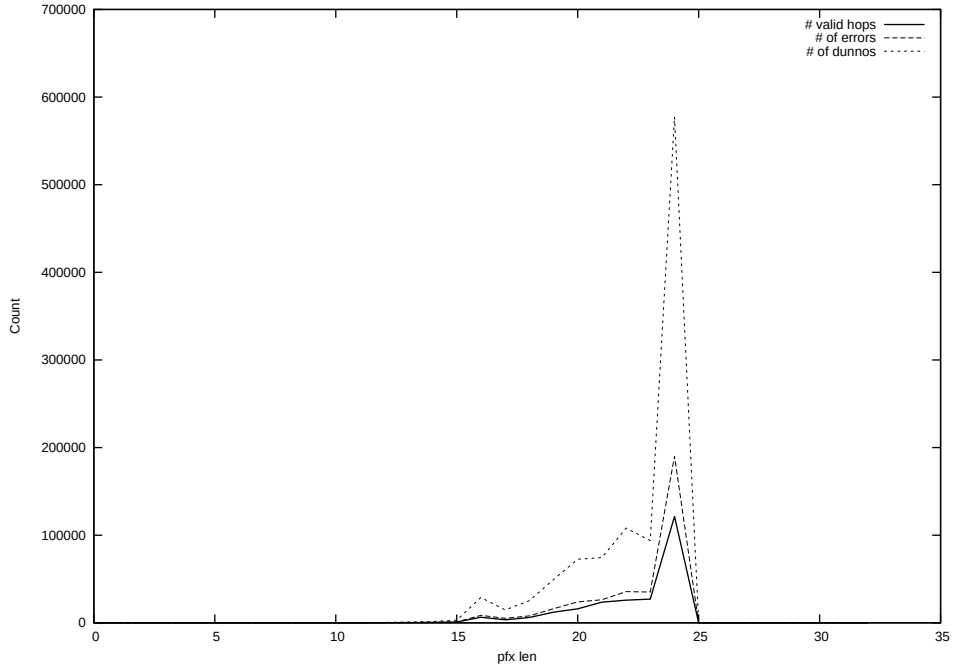
2012-12-26



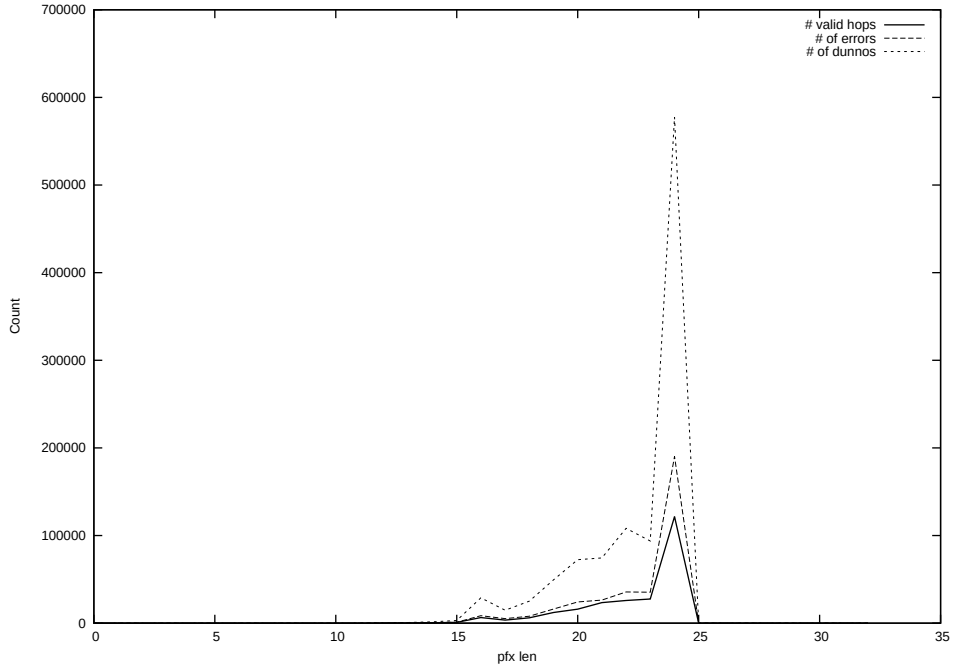
2012-12-27



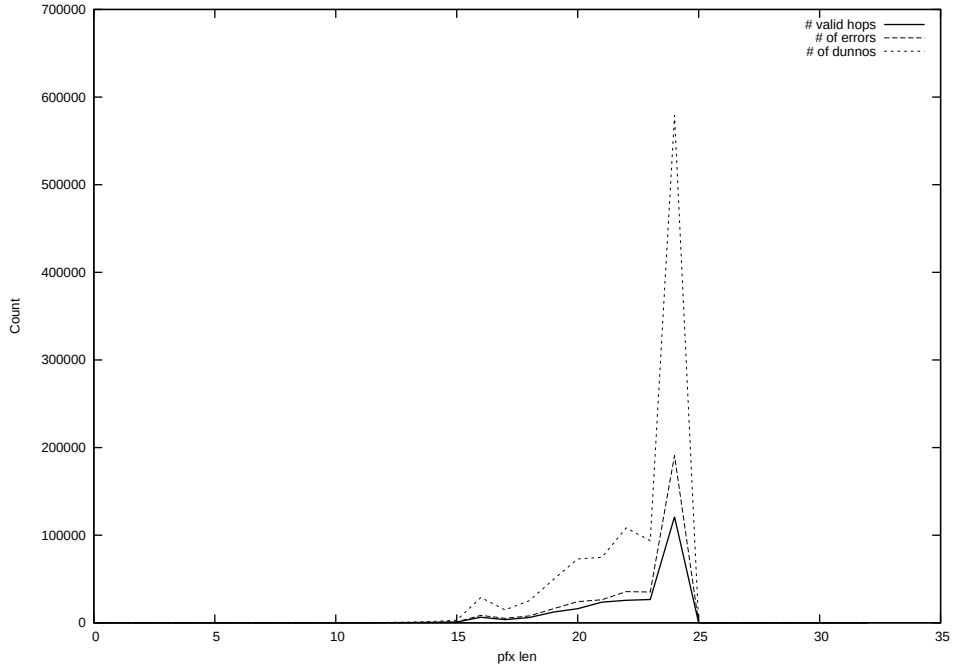
2012-12-28



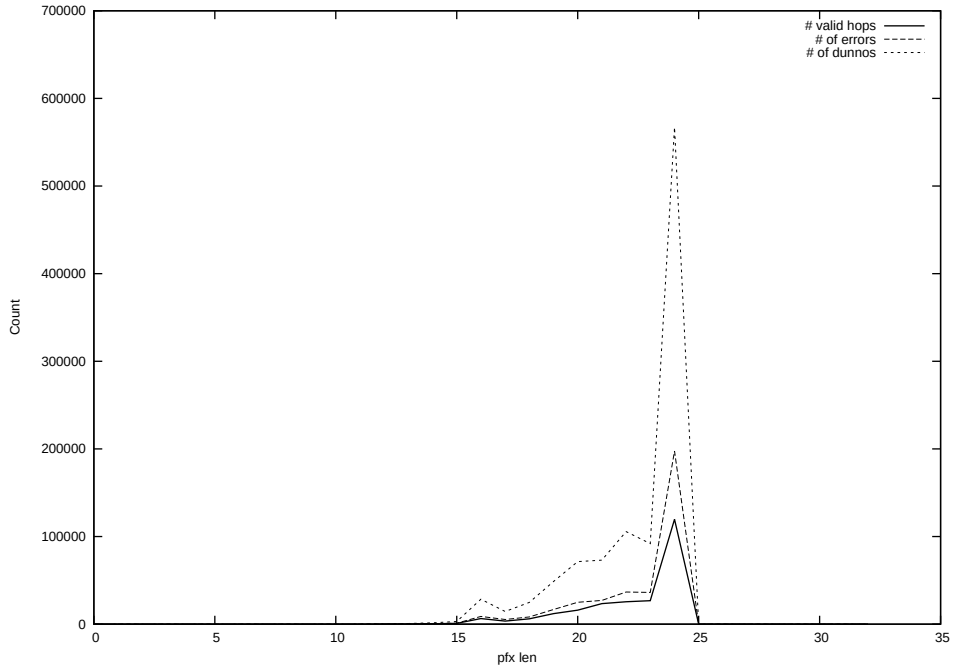
2012-12-29



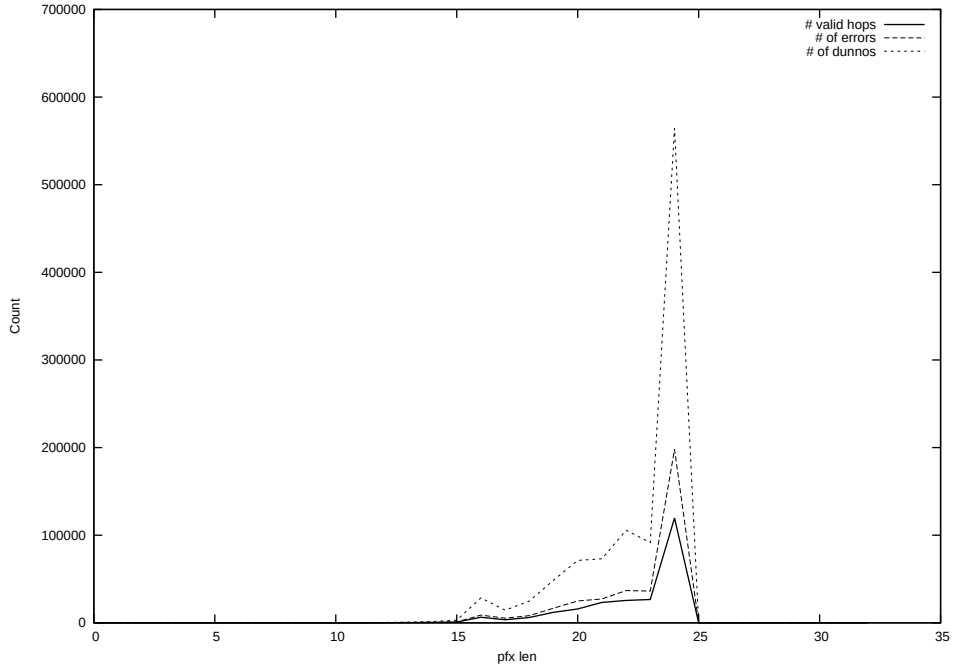
2012-12-30



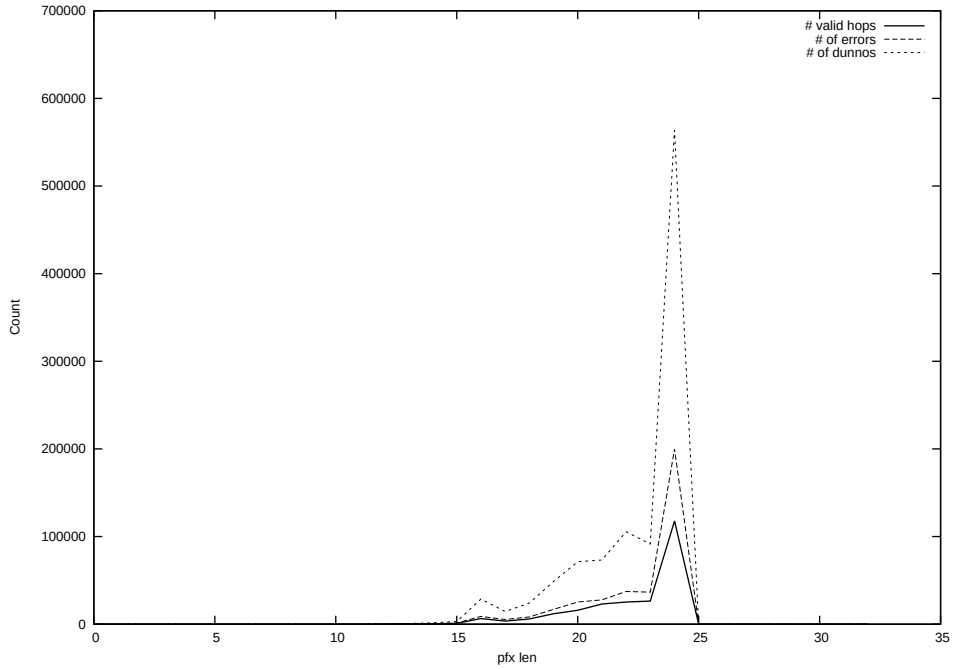
2012-12-31



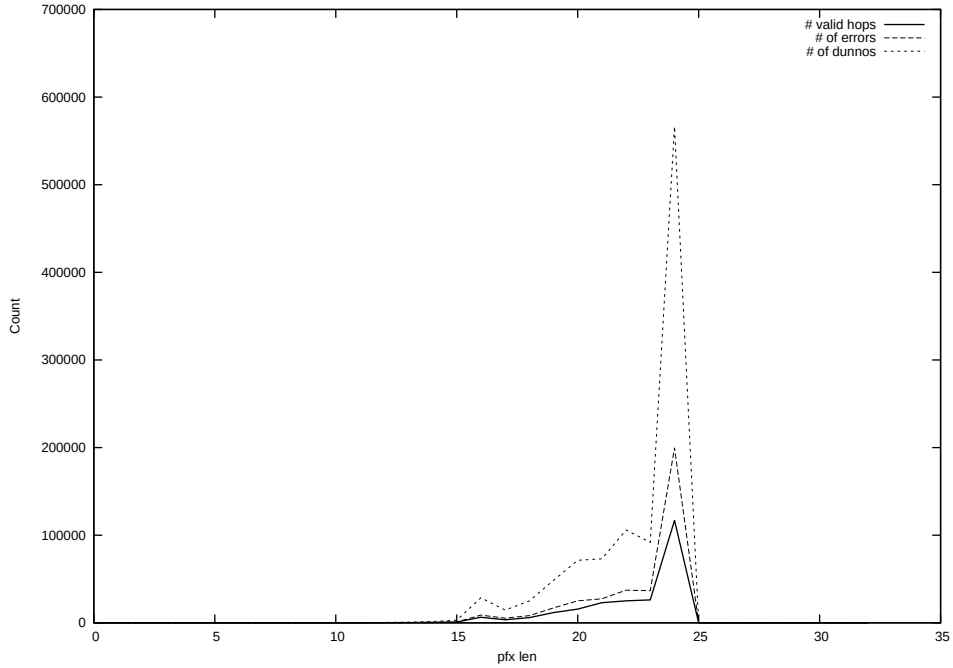
2013-01-01



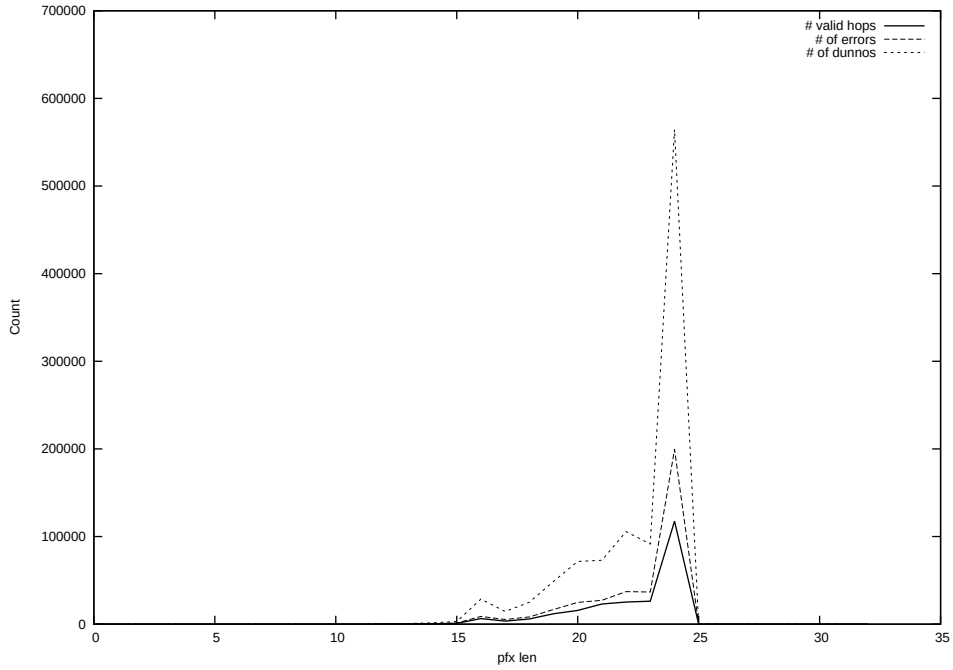
2013-01-02



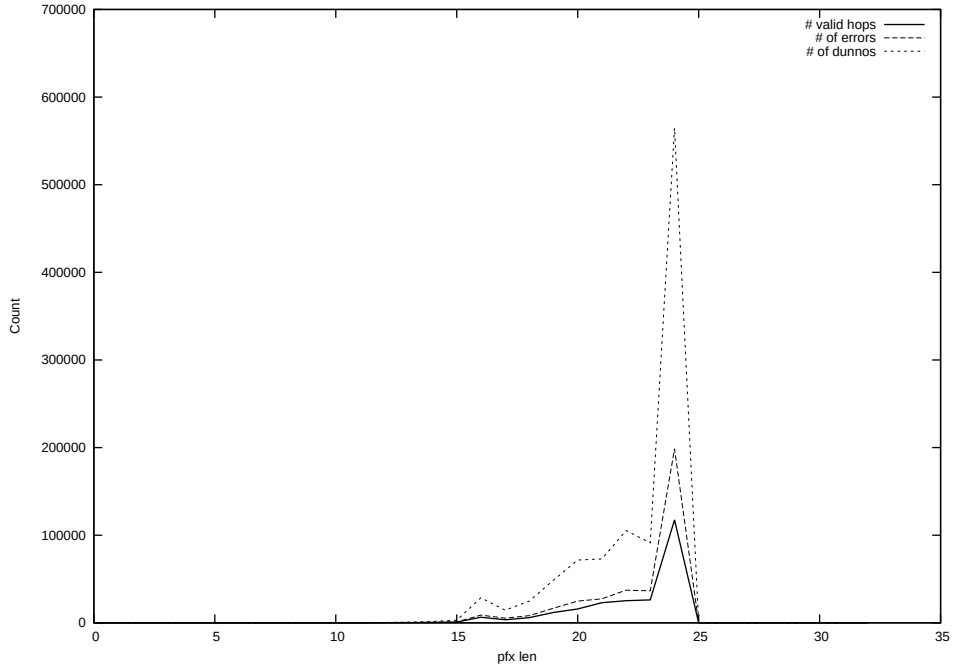
2013-01-03



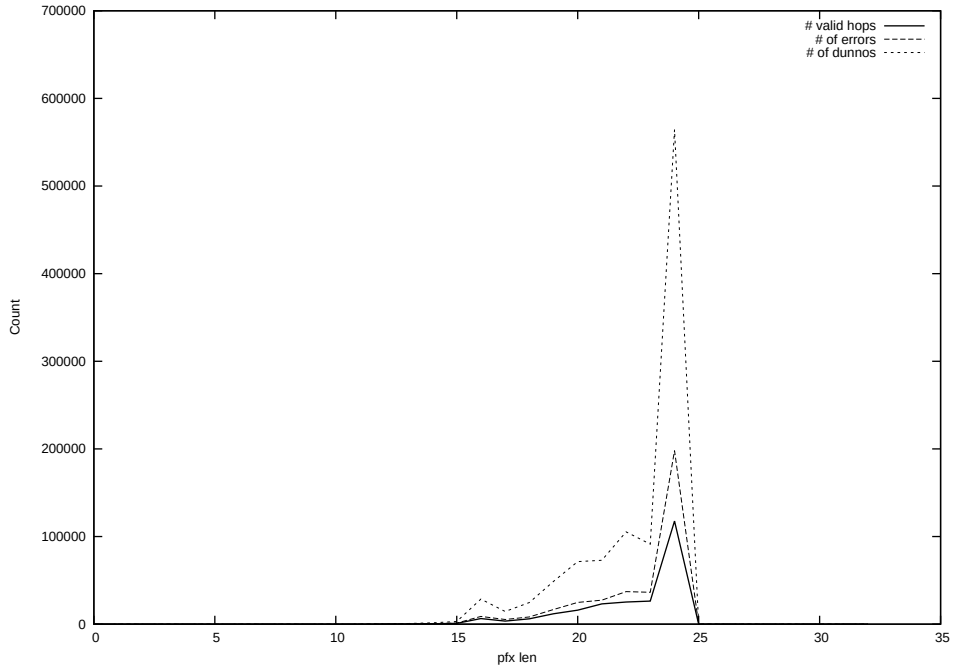
2013-01-04



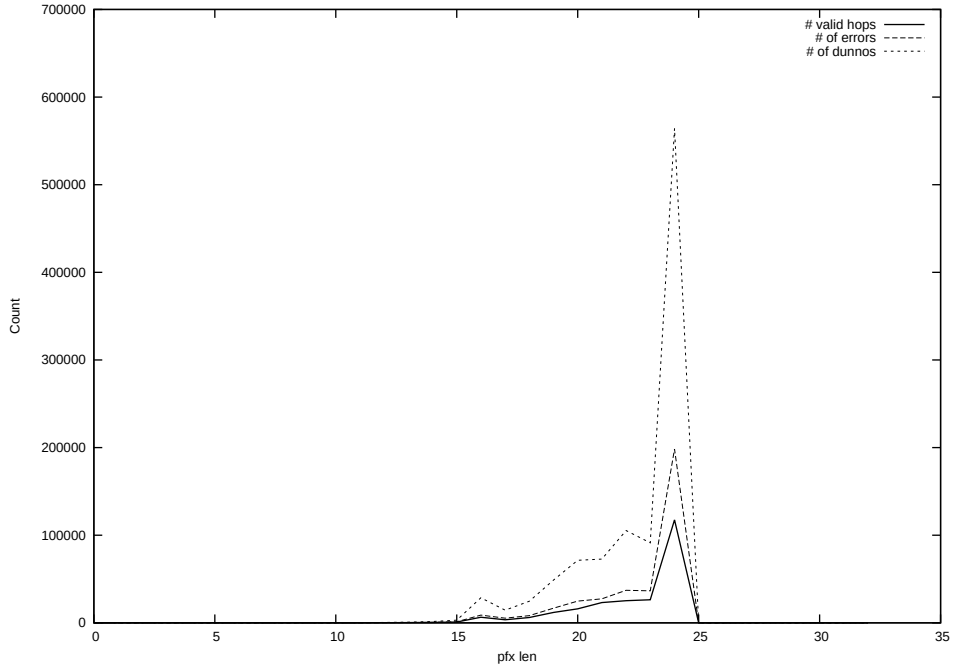
2013-01-05



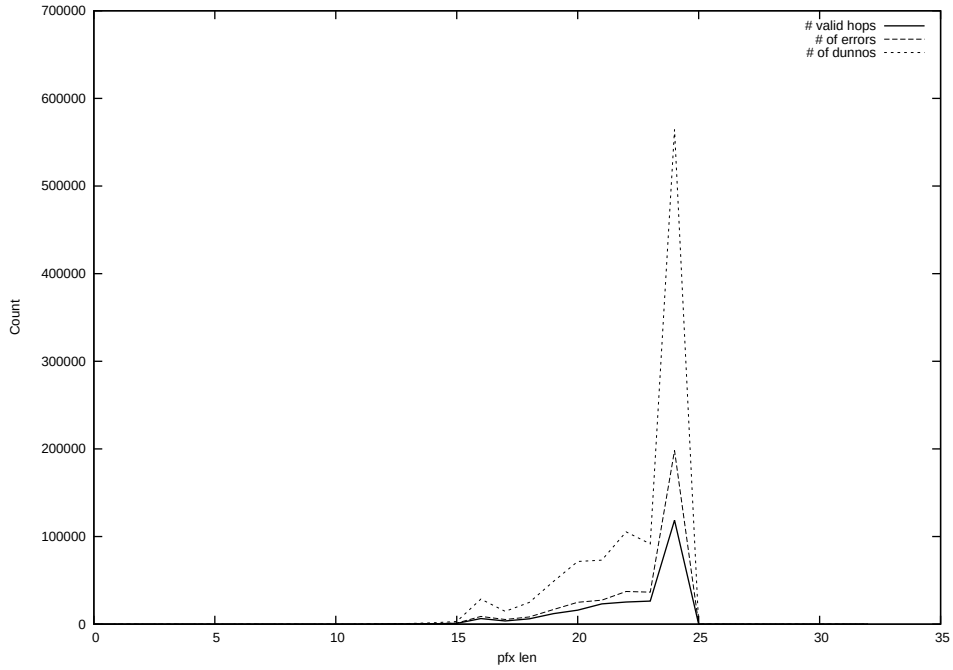
2013-01-06



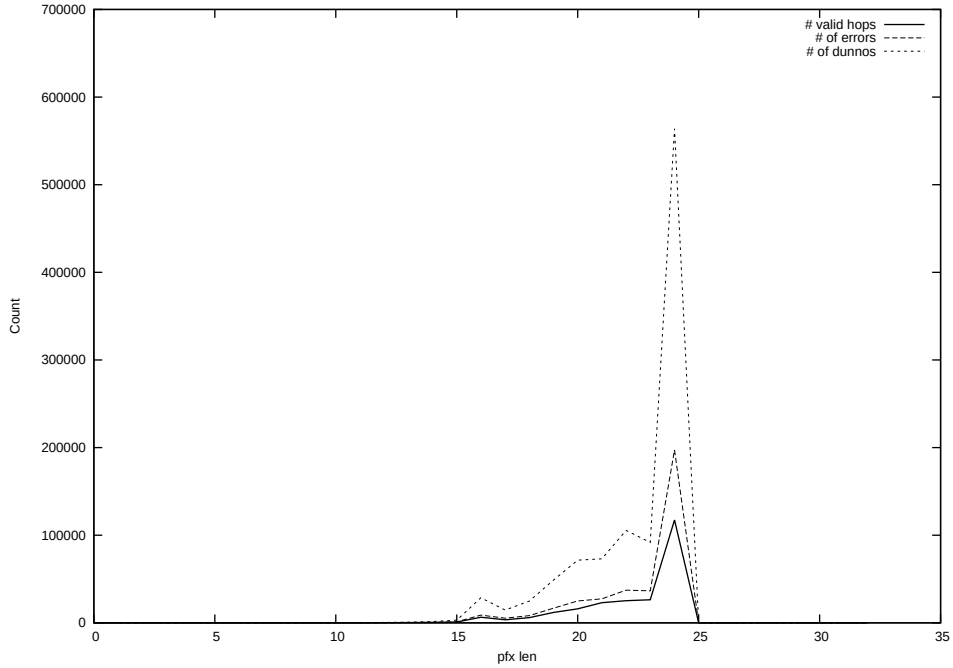
2013-01-07



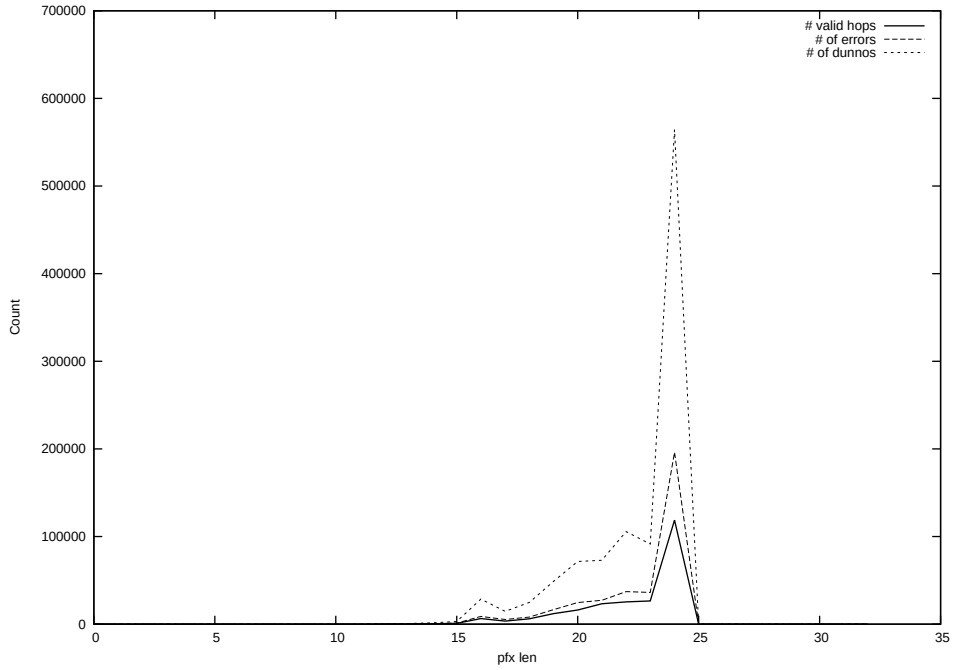
2013-01-08



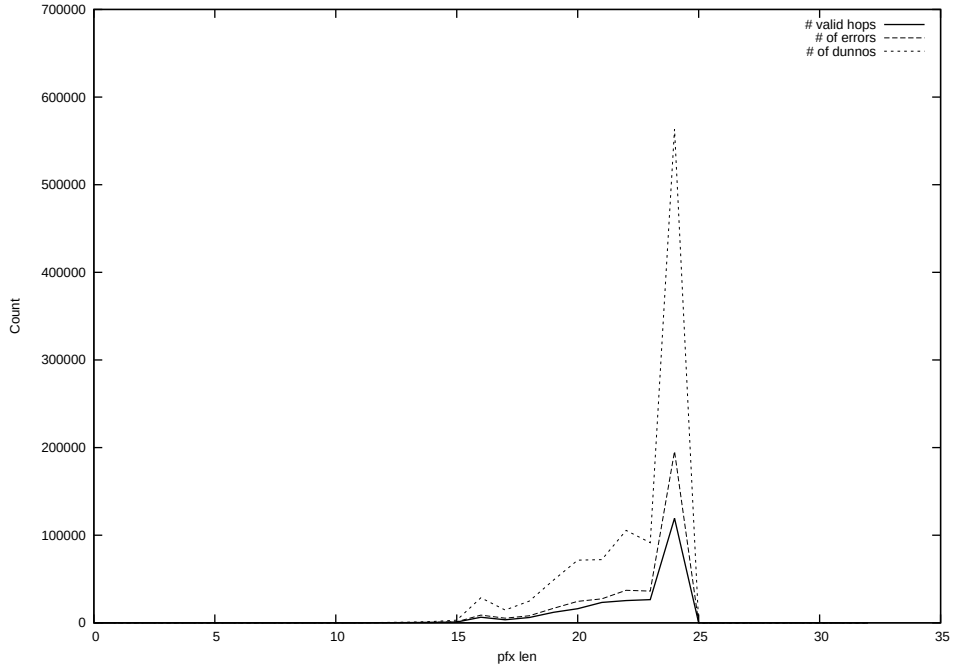
2013-01-09



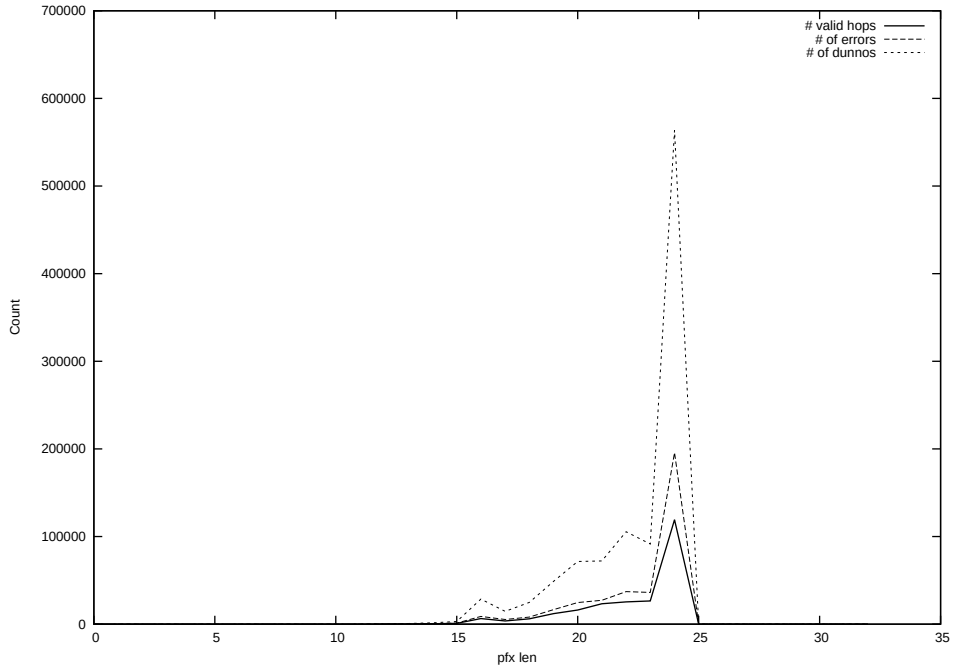
2013-01-10



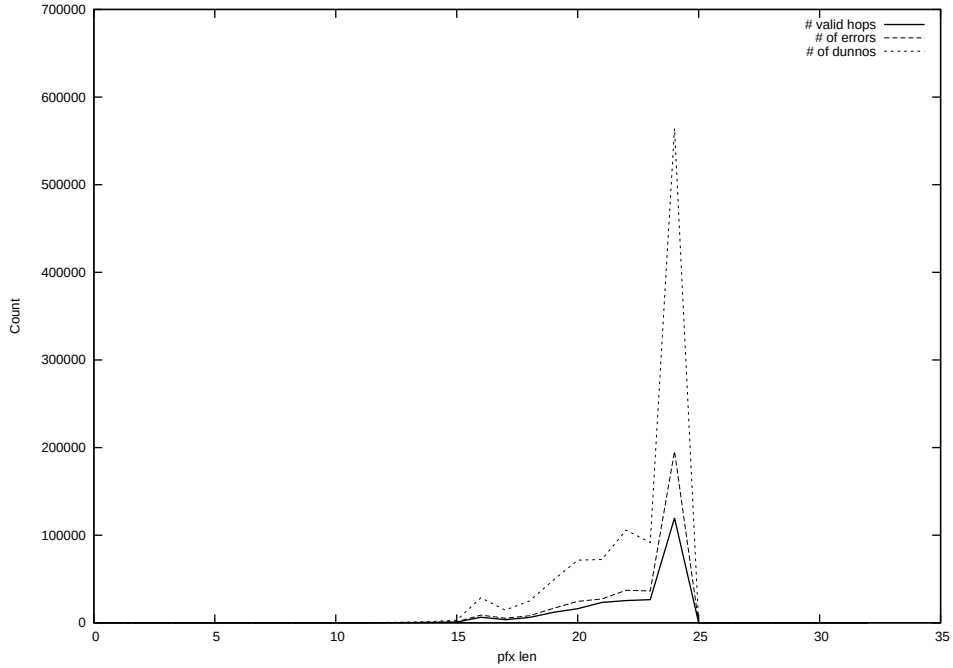
2013-01-11



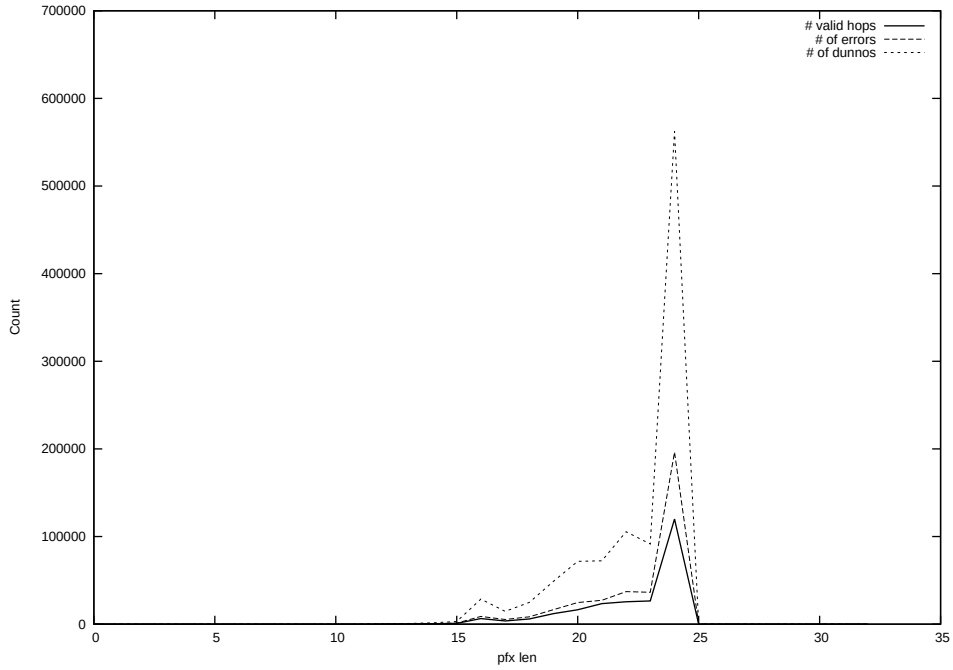
2013-01-12



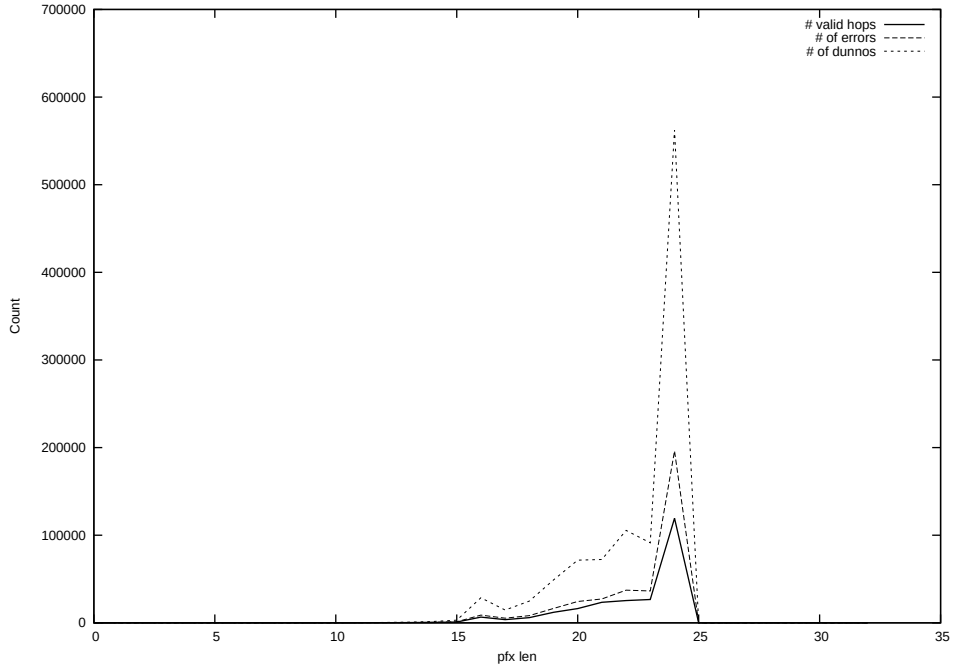
2013-01-13



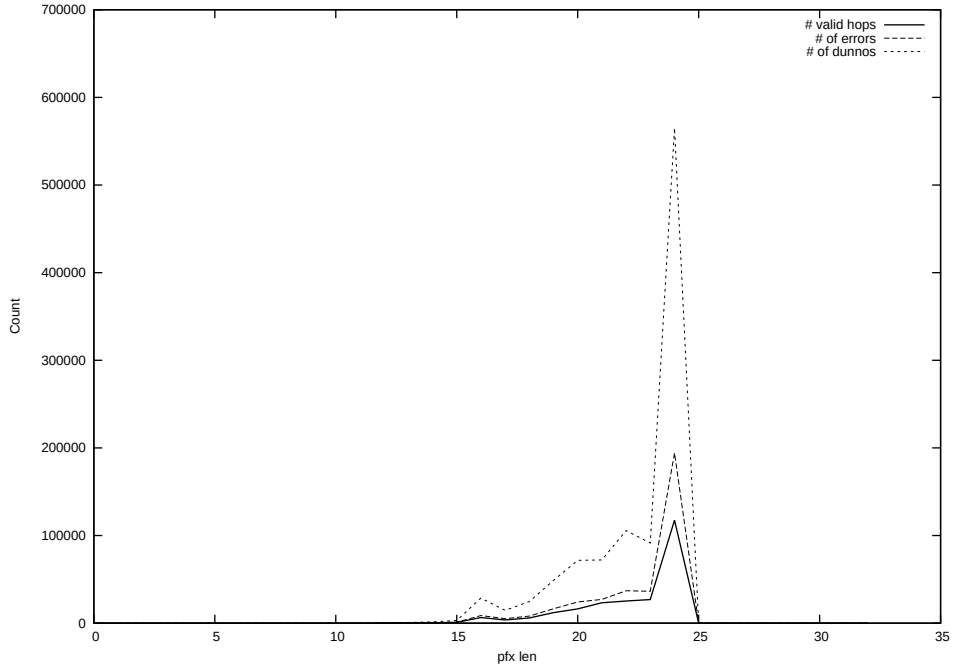
2013-01-14



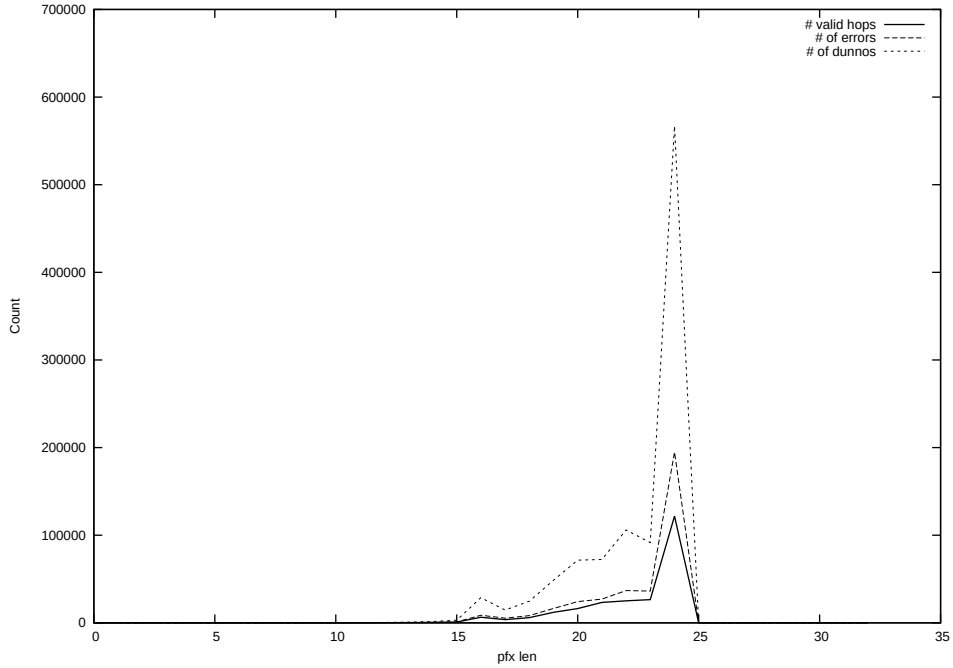
2013-01-15



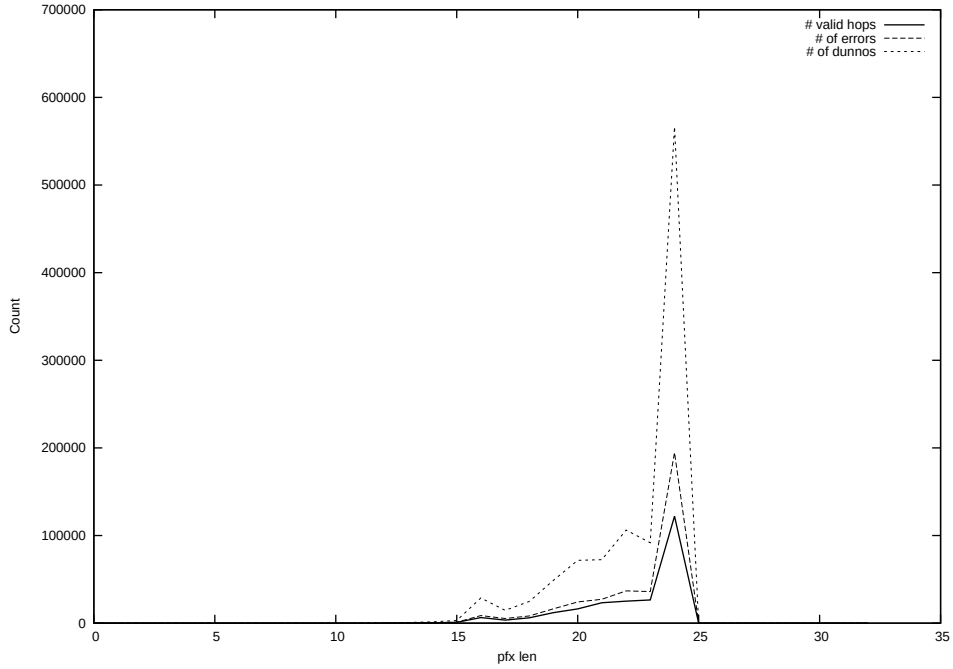
2013-01-16



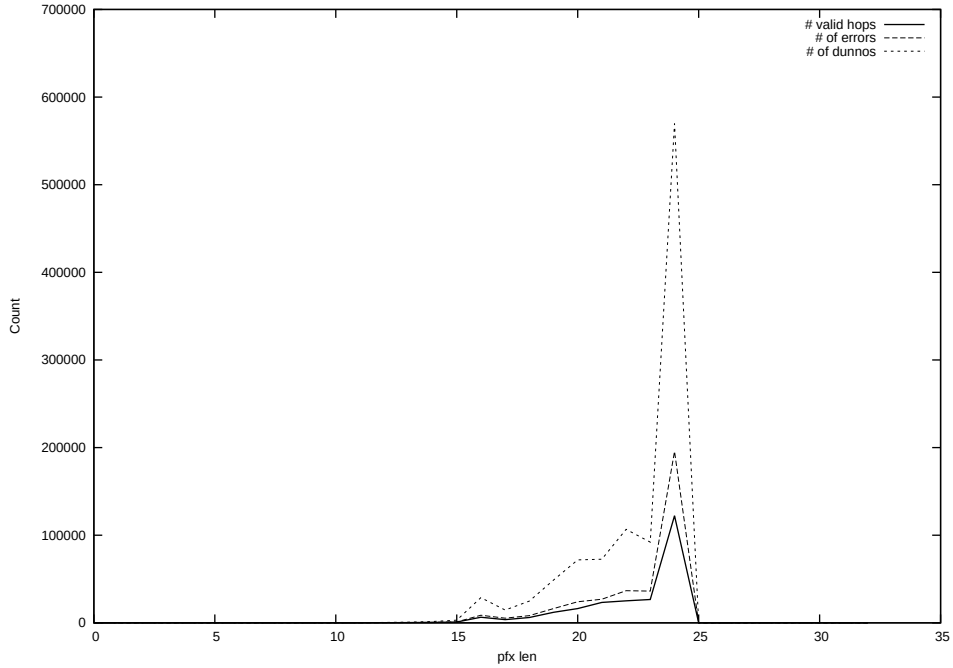
2013-01-17



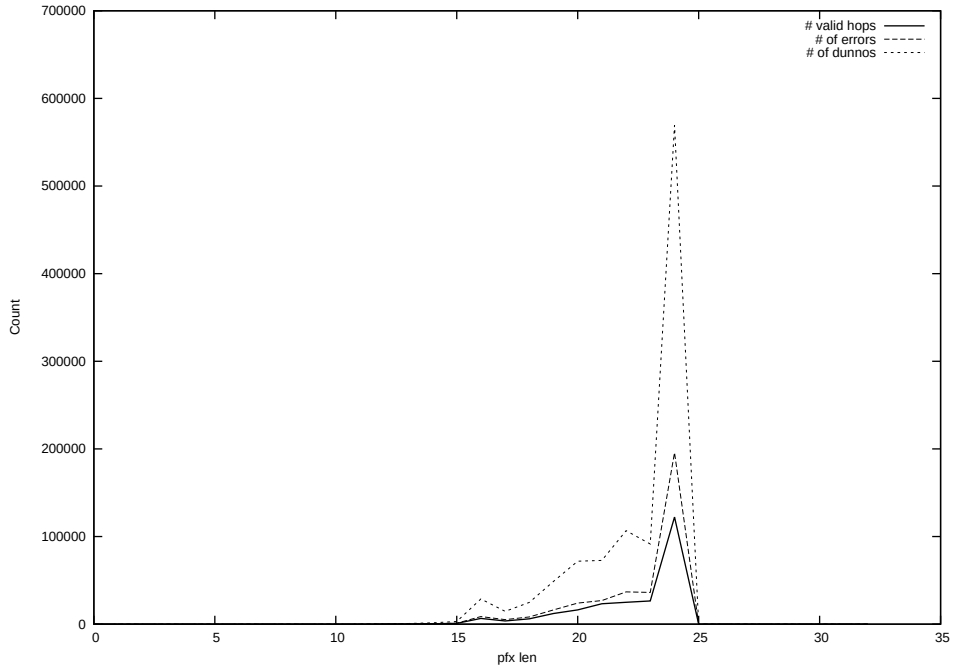
2013-01-18



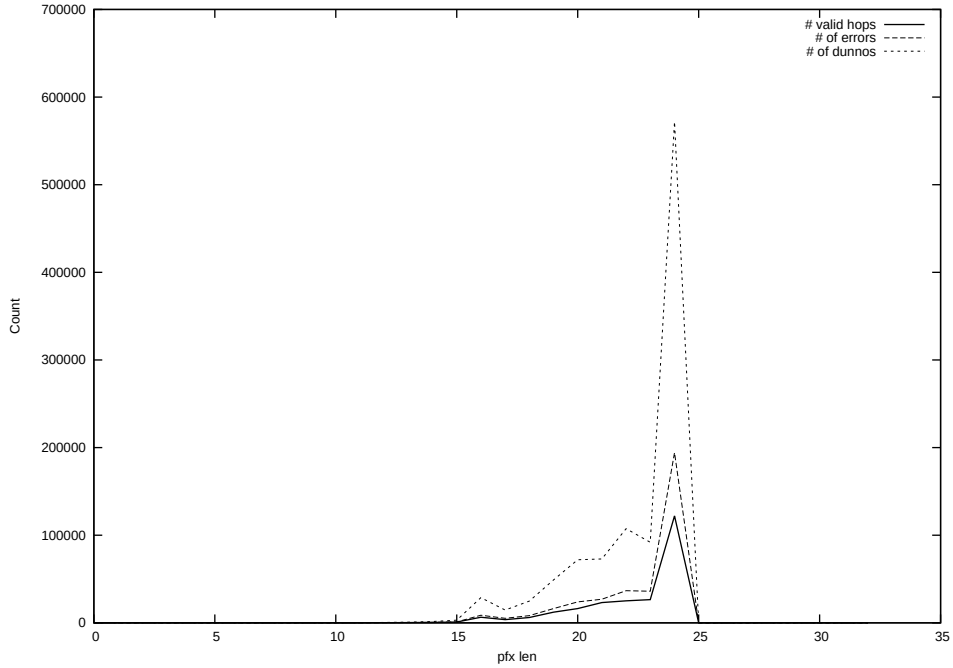
2013-01-19



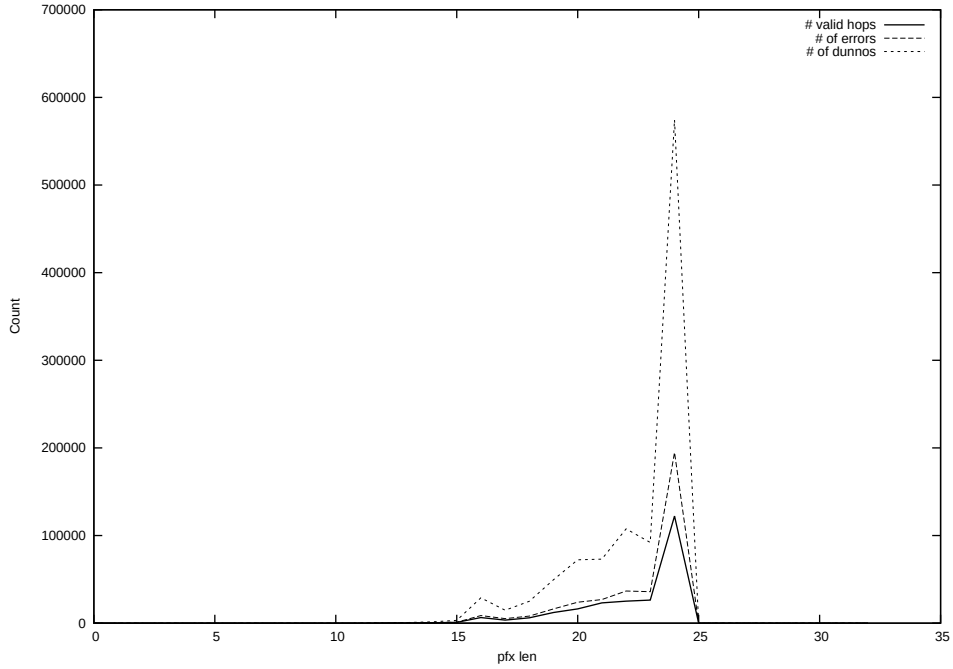
2013-01-20



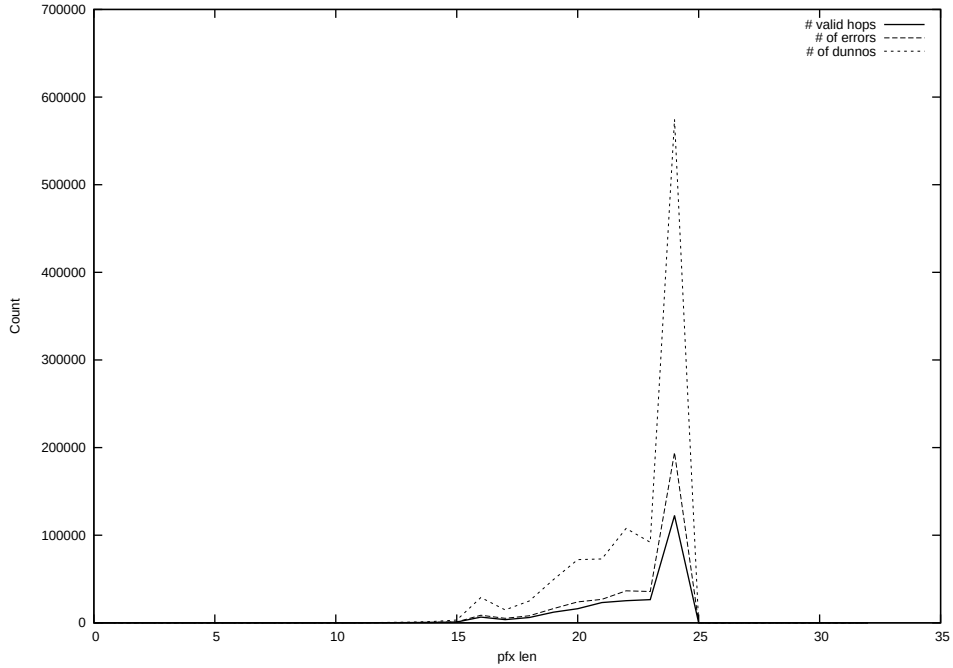
2013-01-21



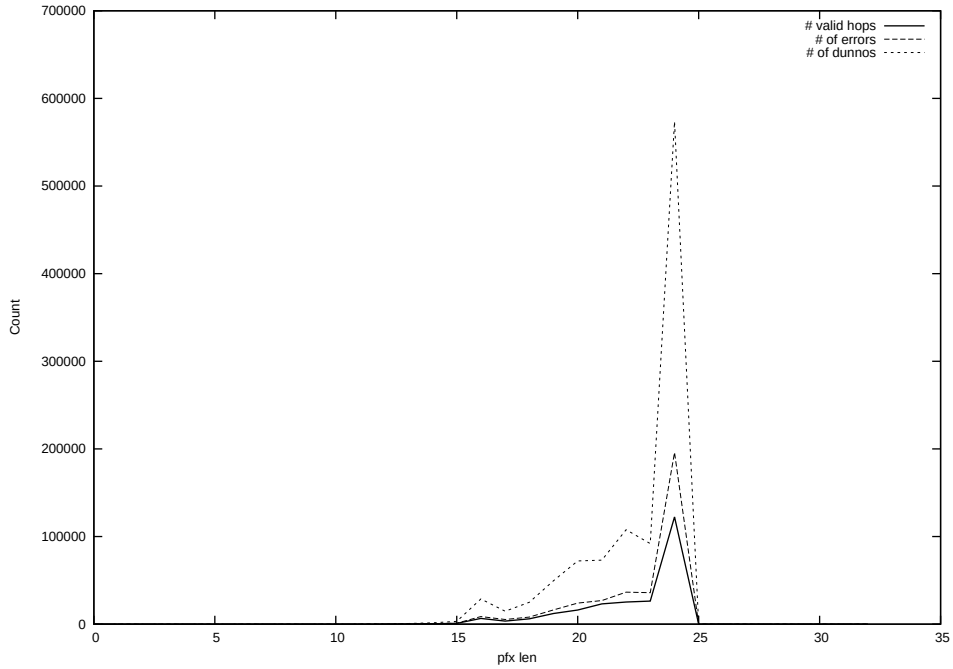
2013-01-22



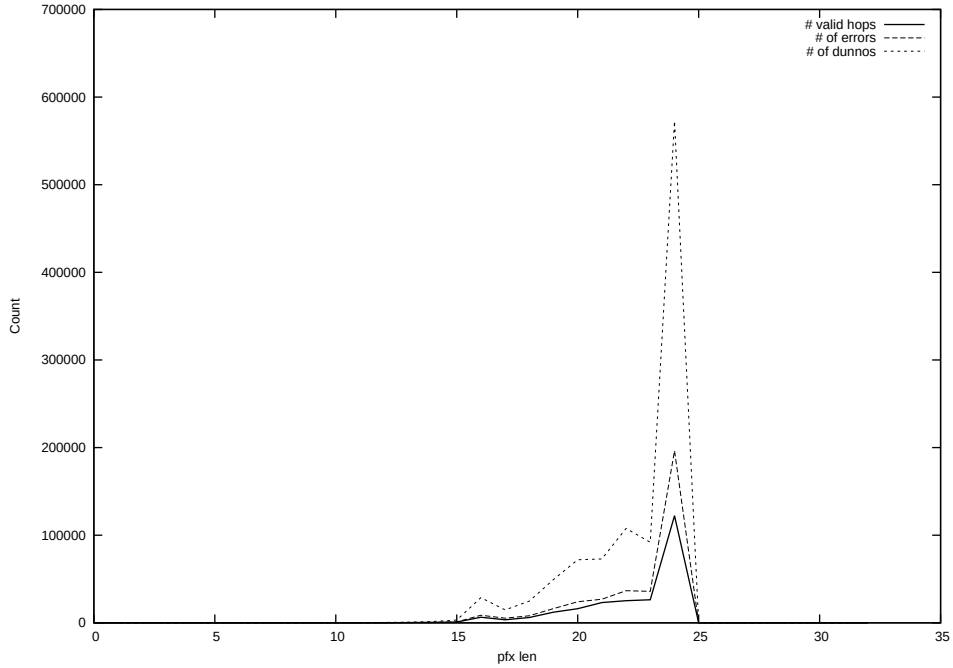
2013-01-23



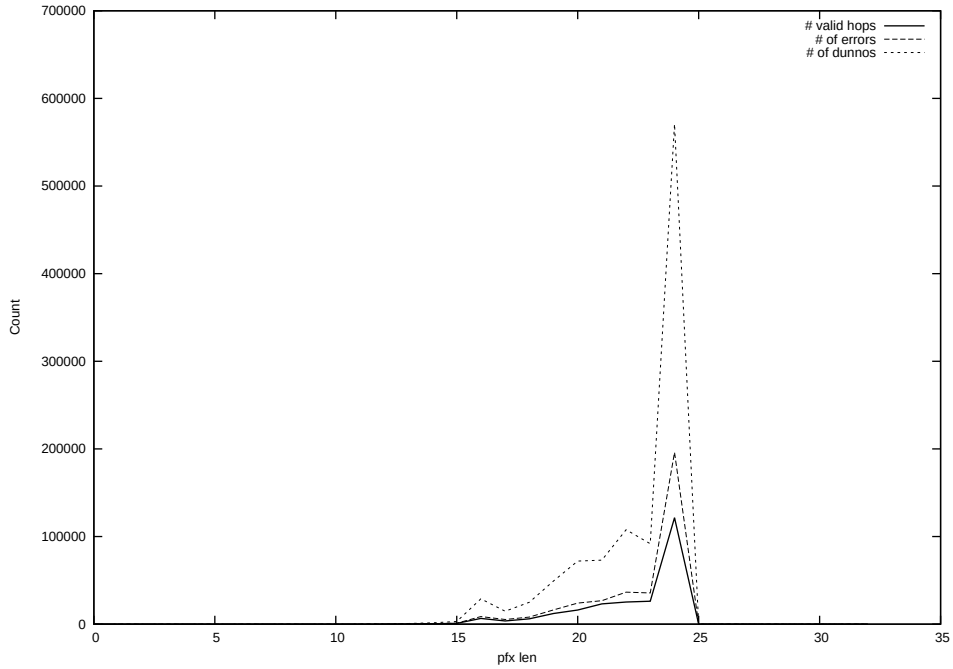
2013-01-24



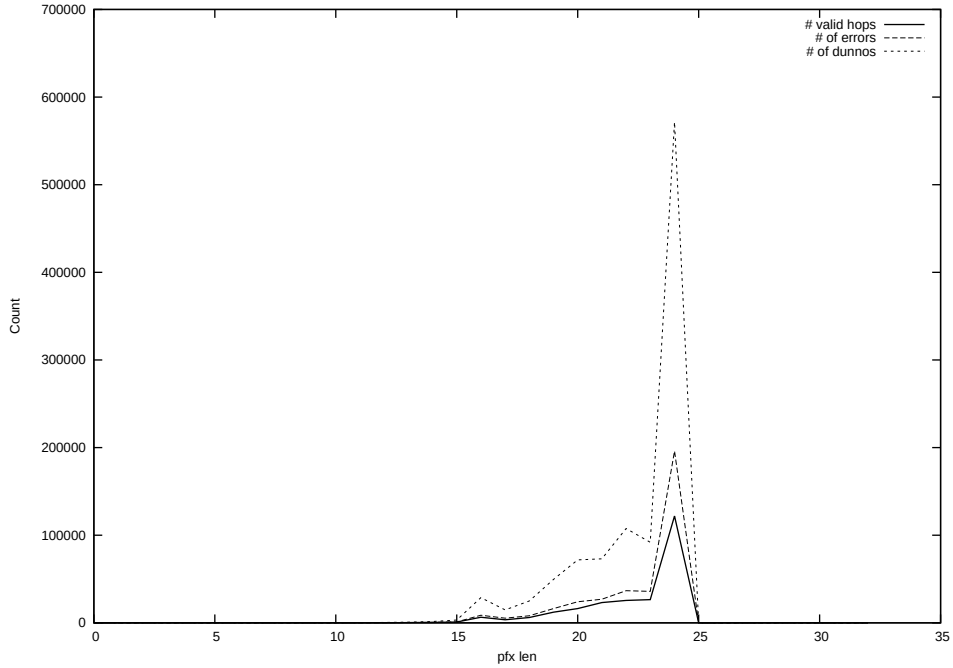
2013-01-25



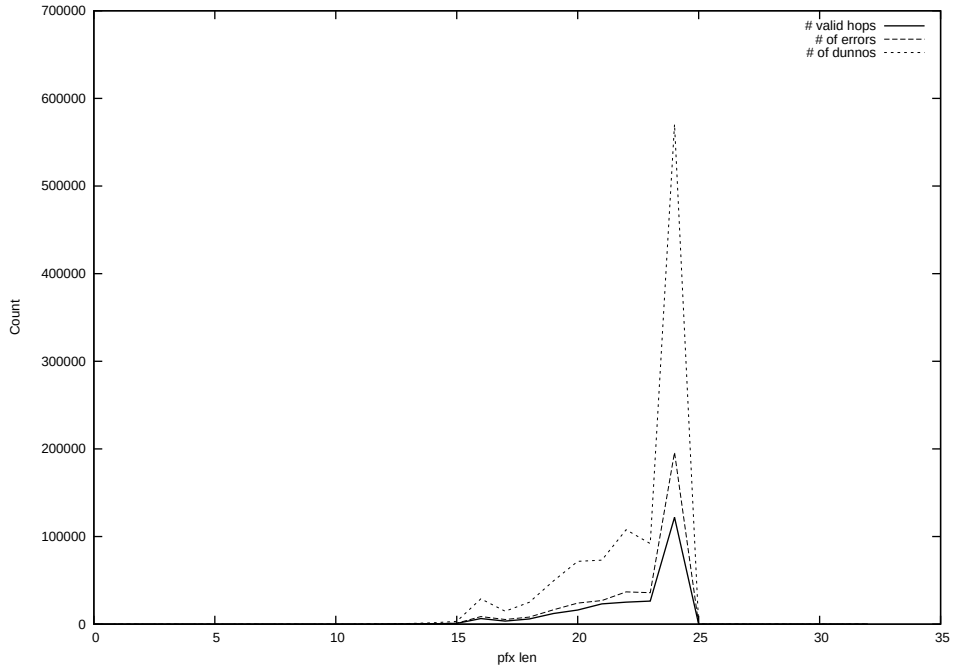
2013-01-26



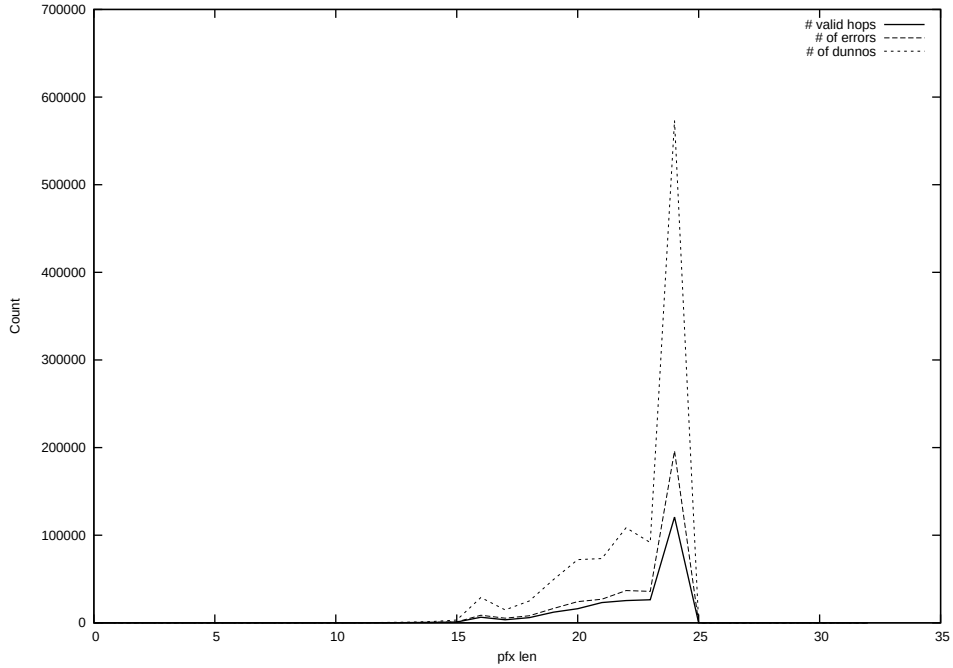
2013-01-27



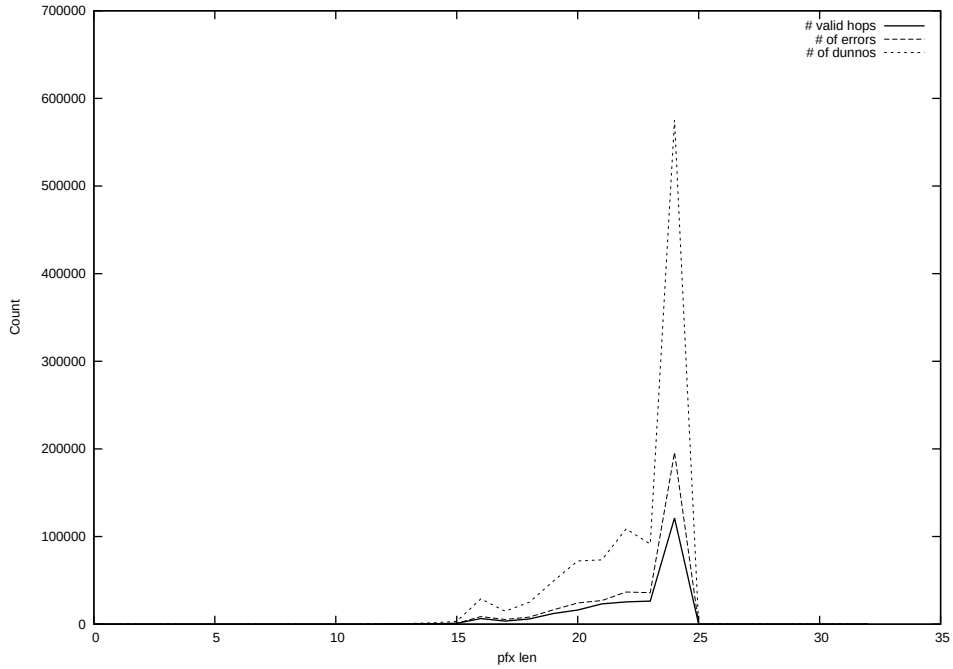
2013-01-28



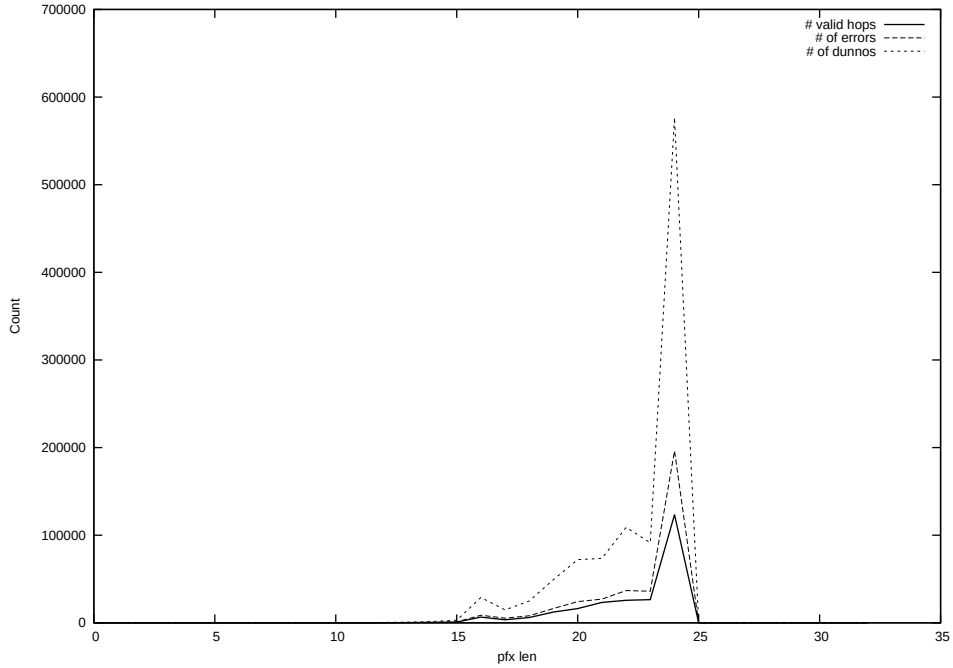
2013-01-29



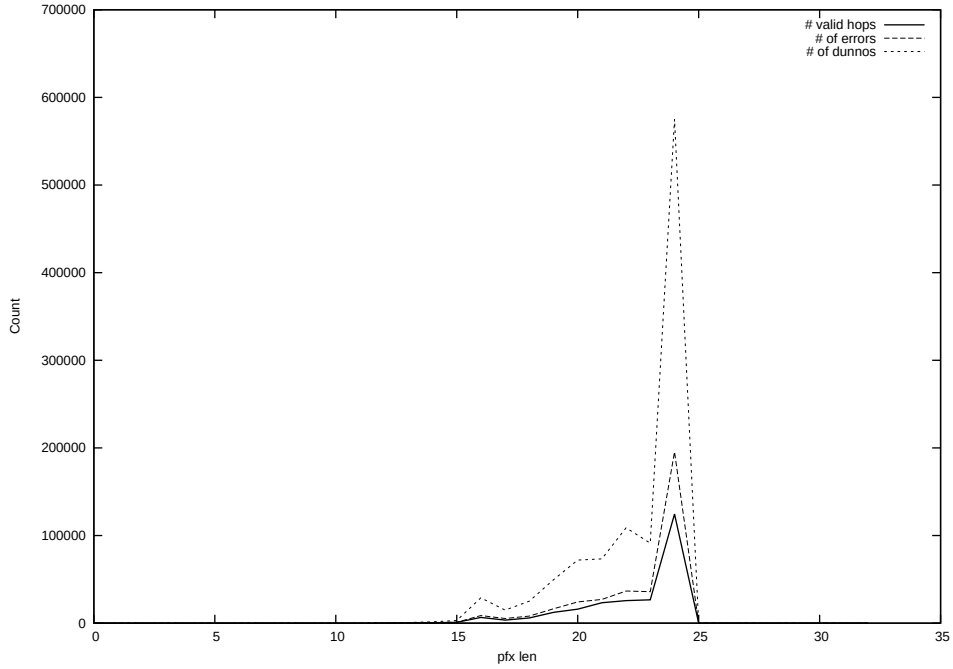
2013-01-30



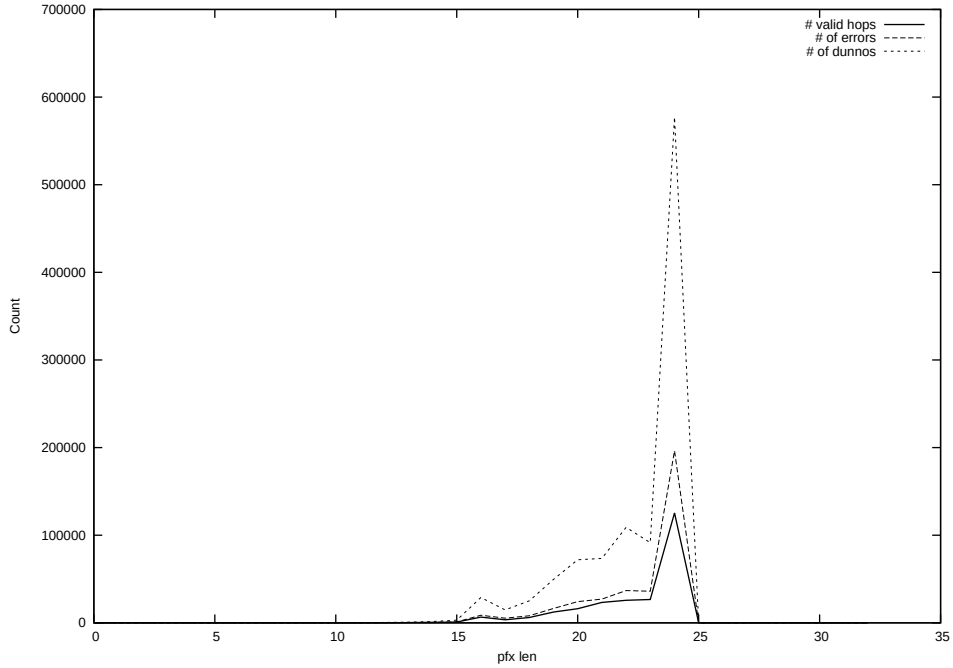
2013-01-31



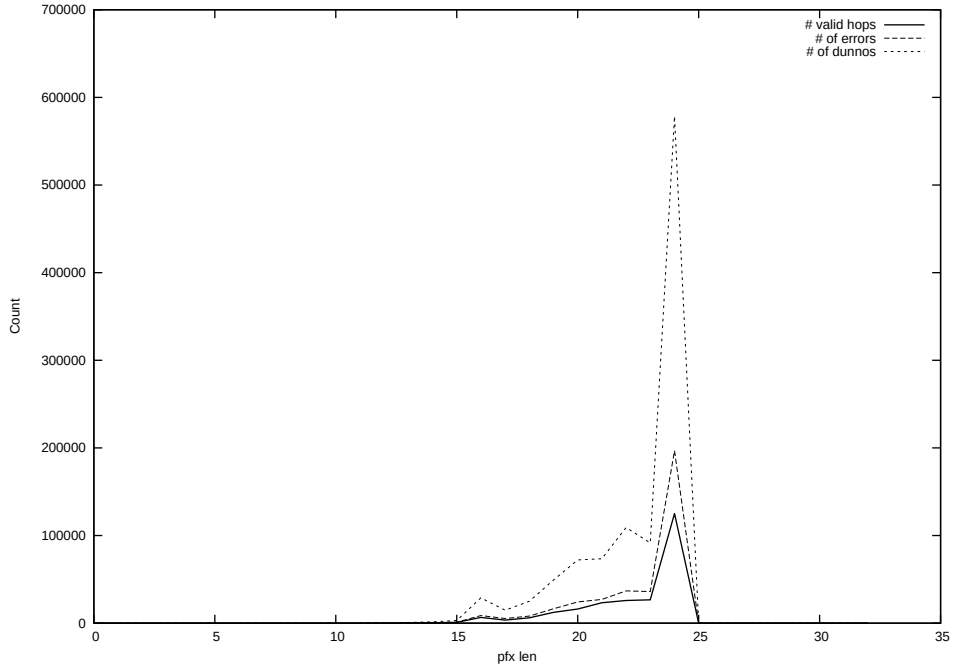
2013-02-01



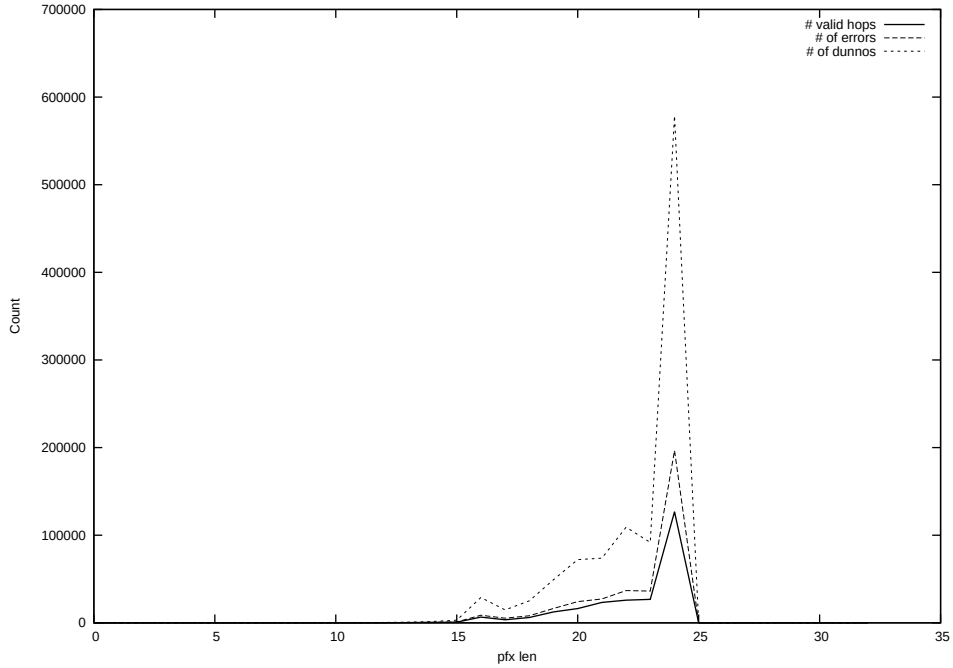
2013-02-02



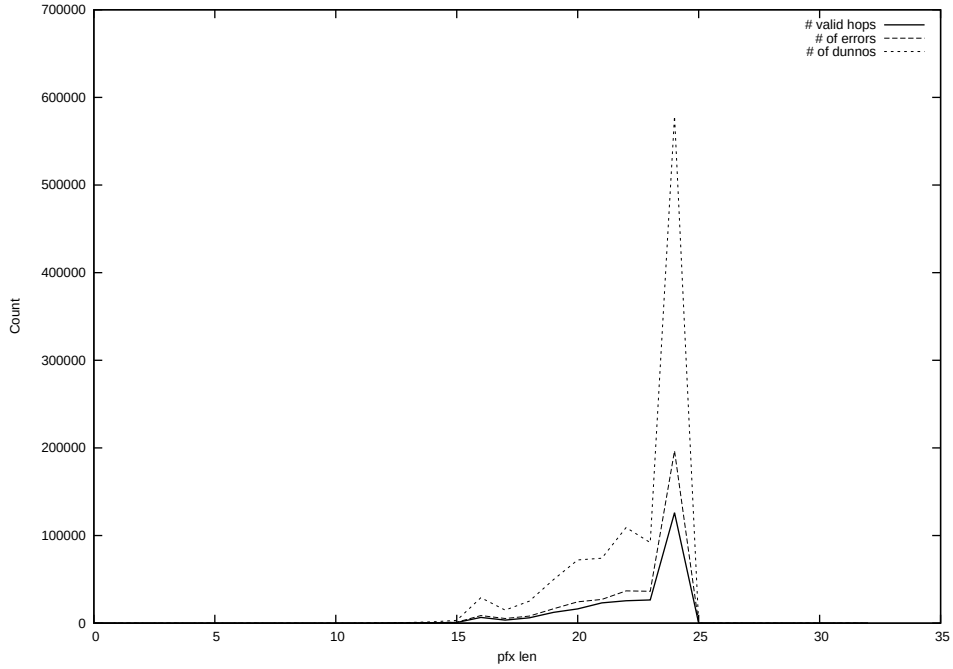
2013-02-03



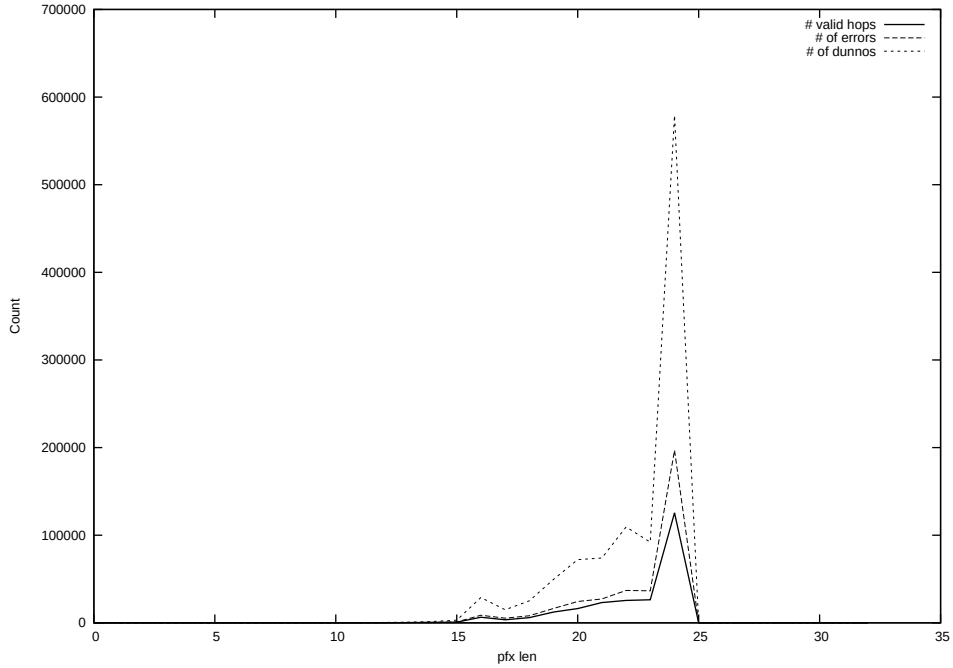
2013-02-04



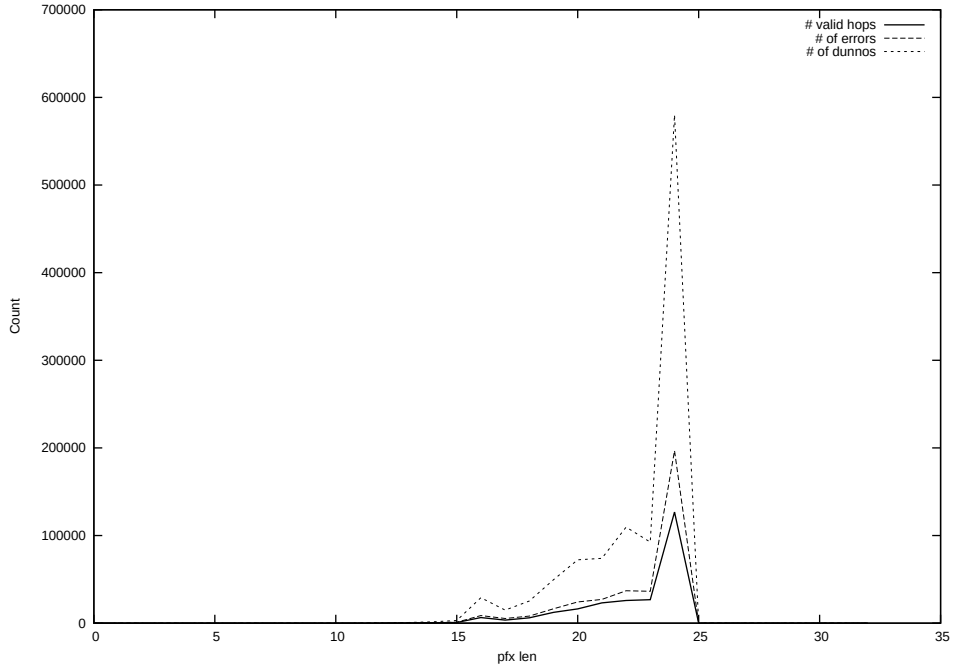
2013-02-05



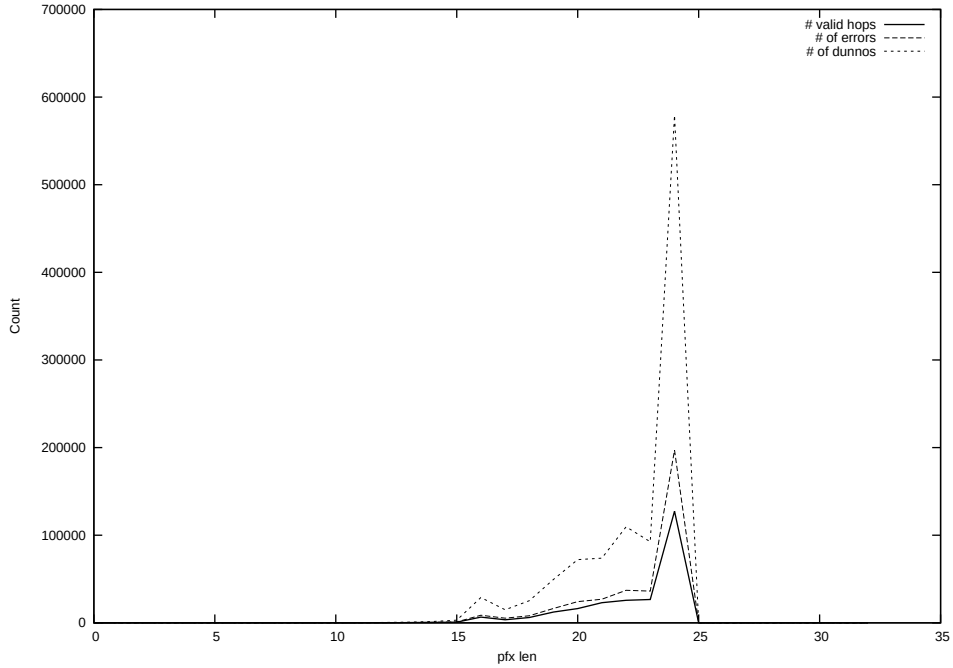
2013-02-06



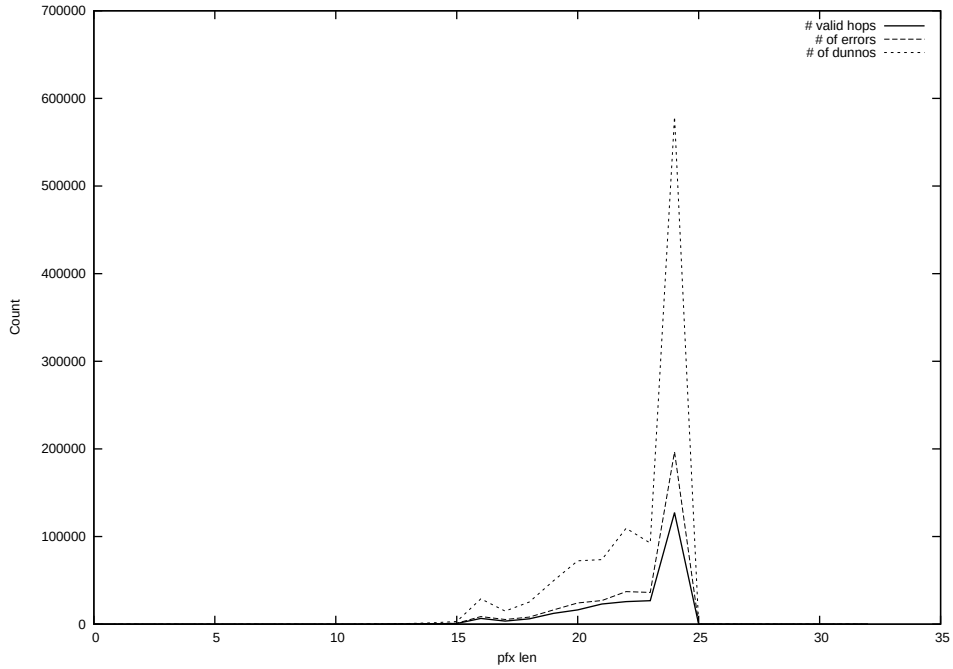
2013-02-07



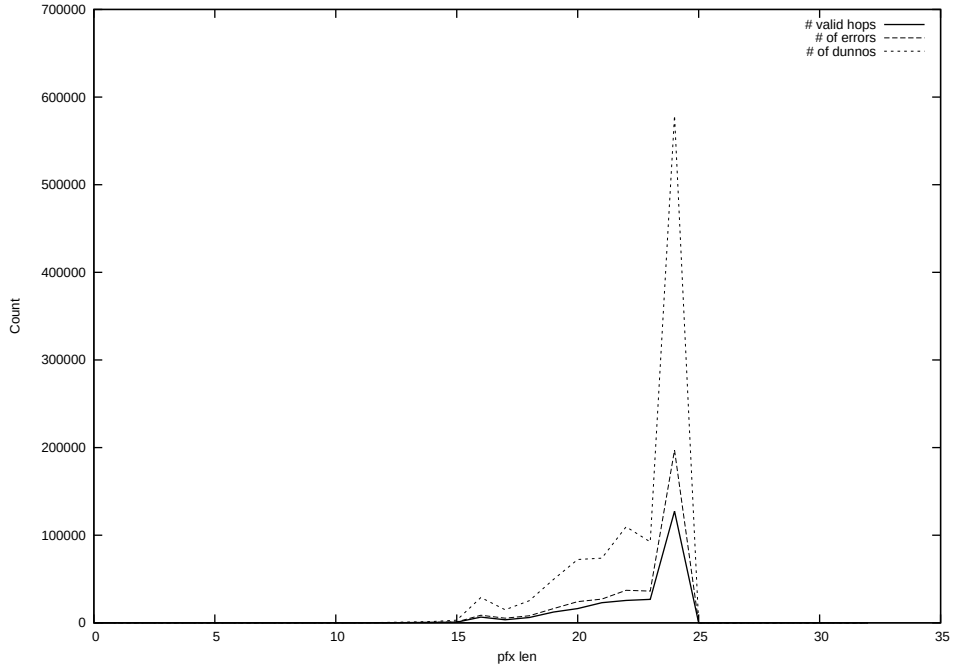
2013-02-08



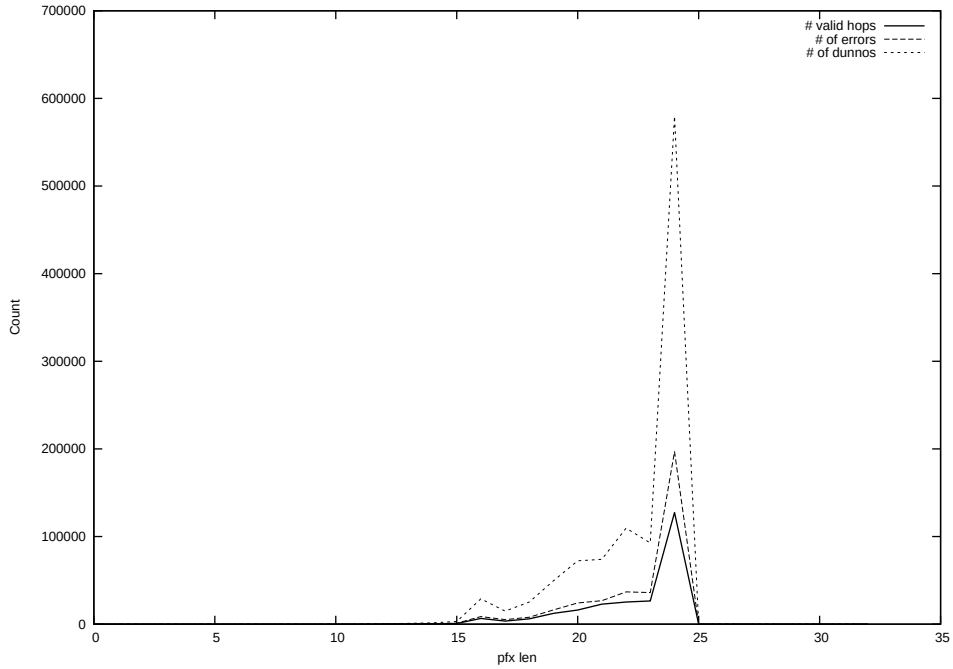
2013-02-09



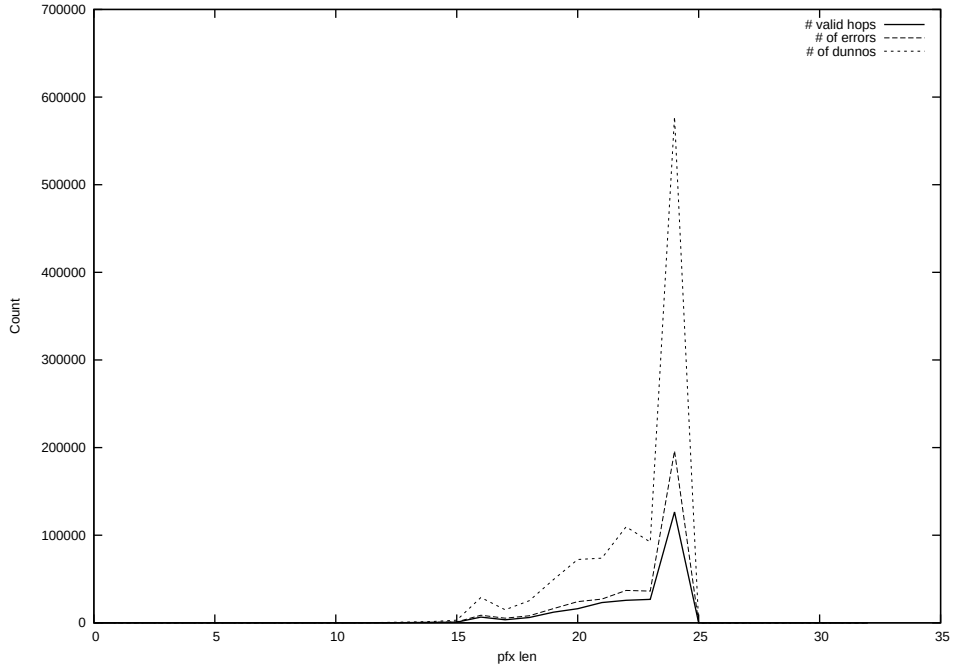
2013-02-10



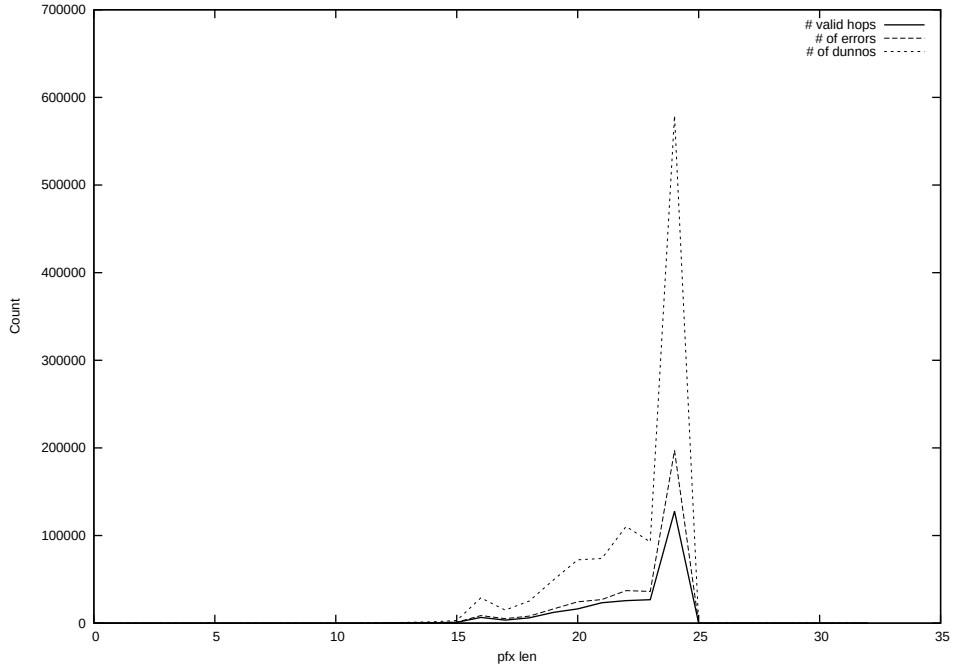
2013-02-11



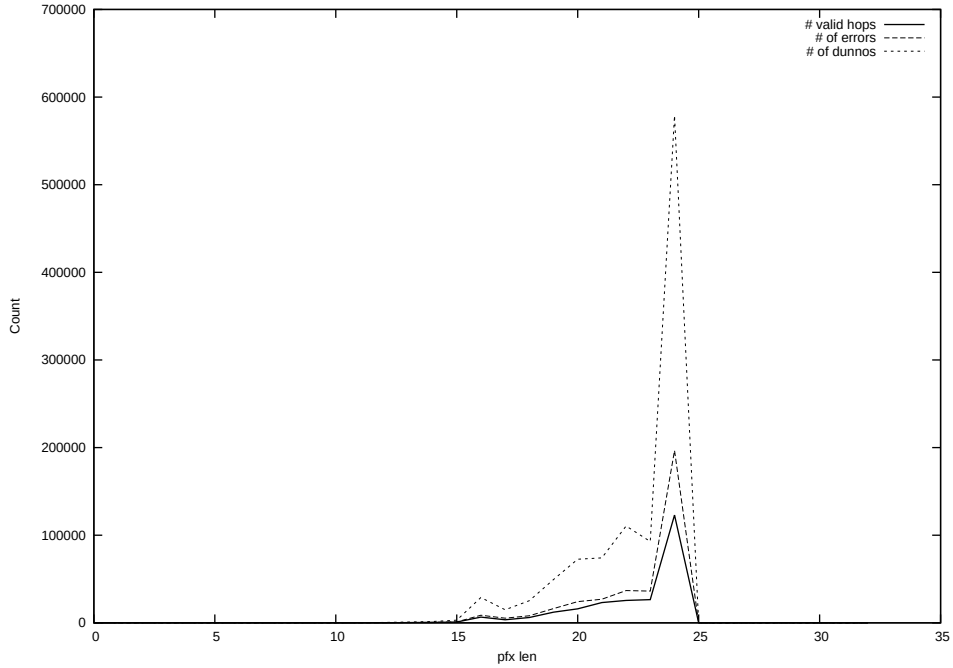
2013-02-12



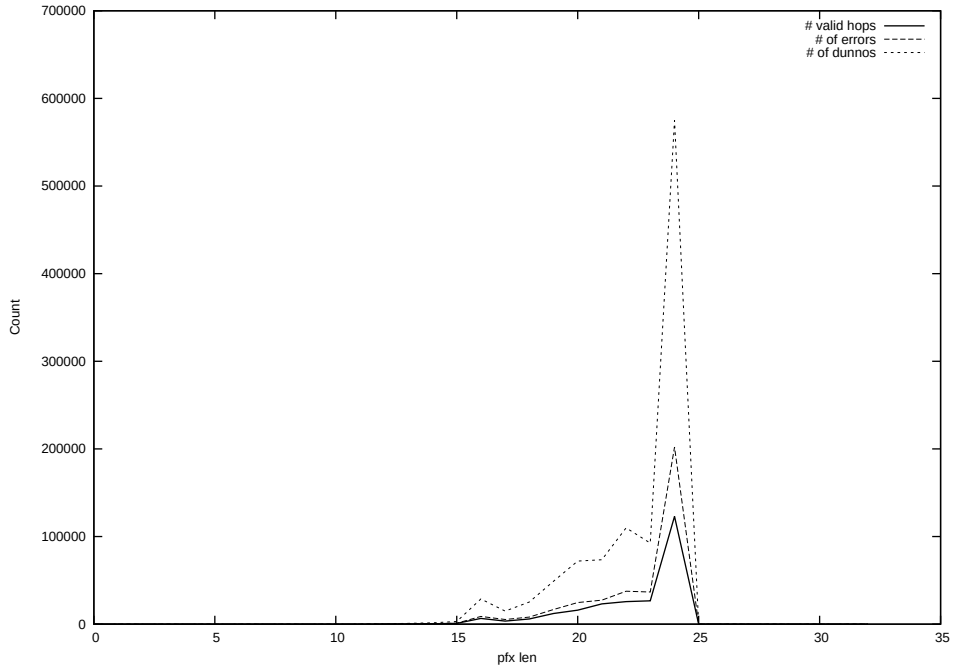
2013-02-13



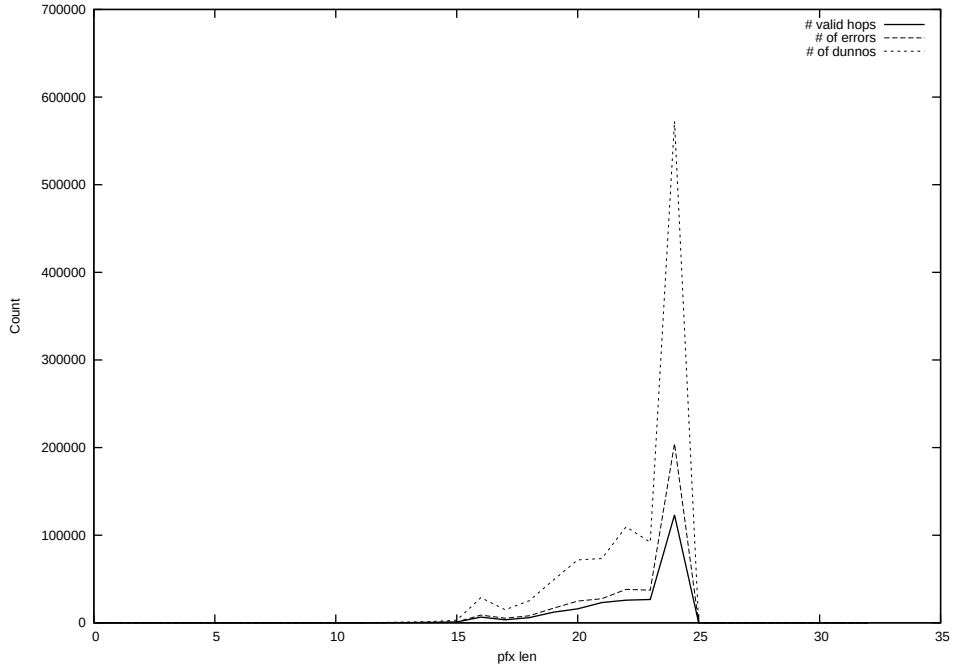
2013-02-14



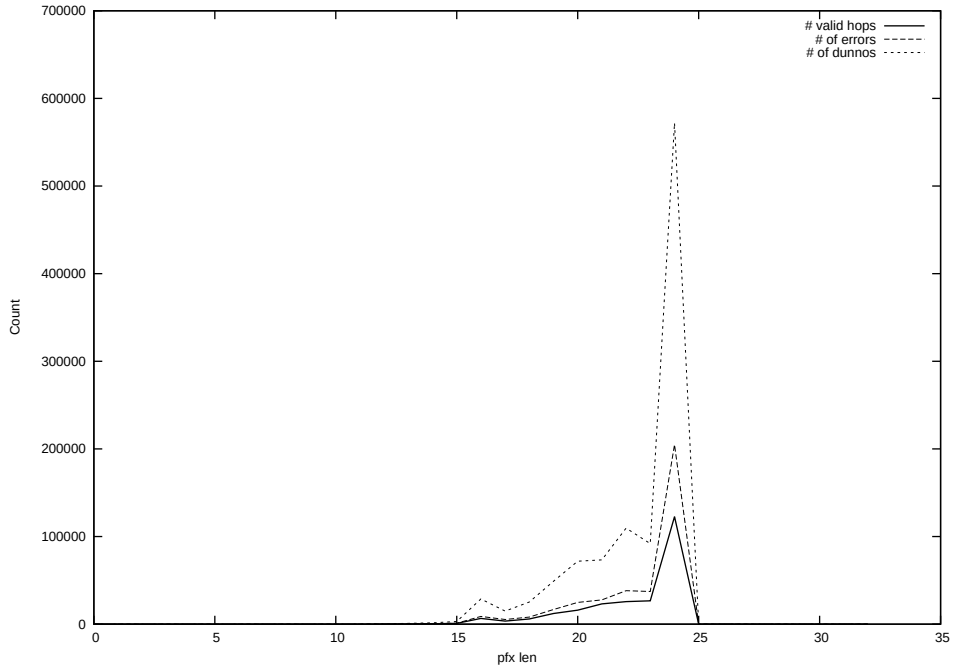
2013-02-15



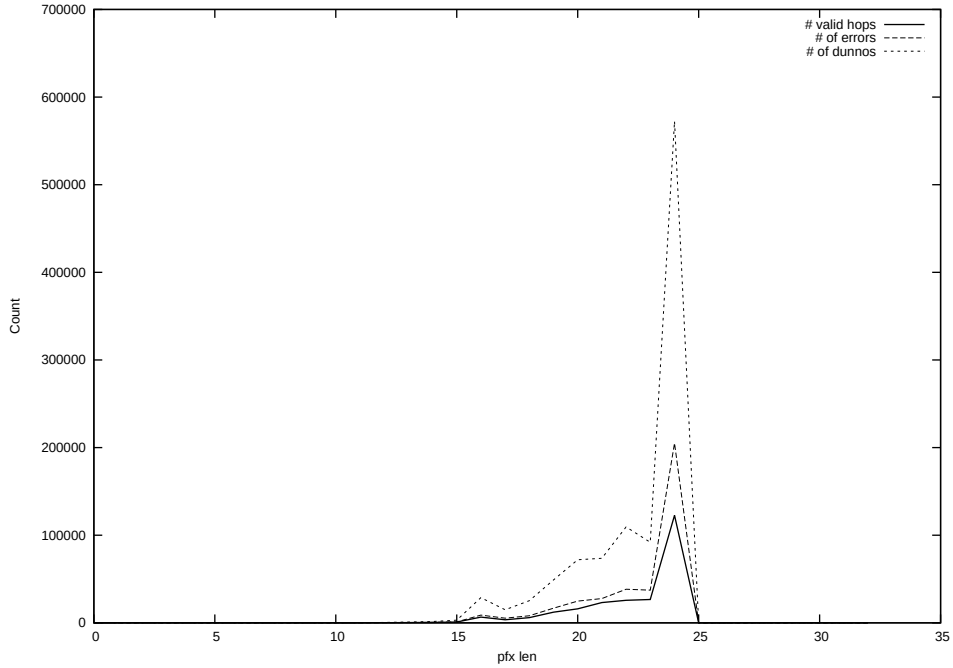
2013-02-16



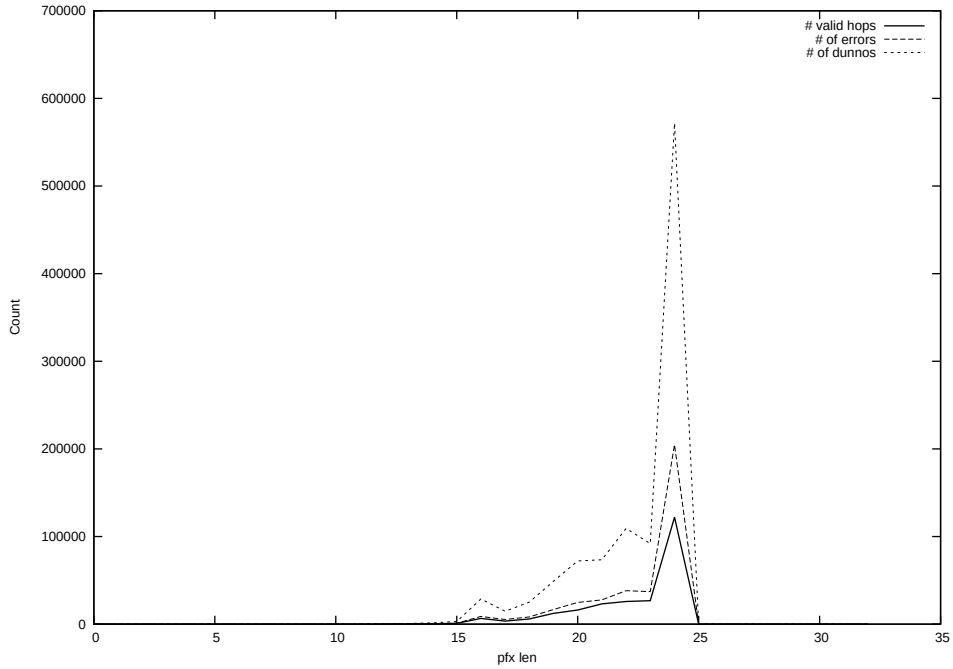
2013-02-17



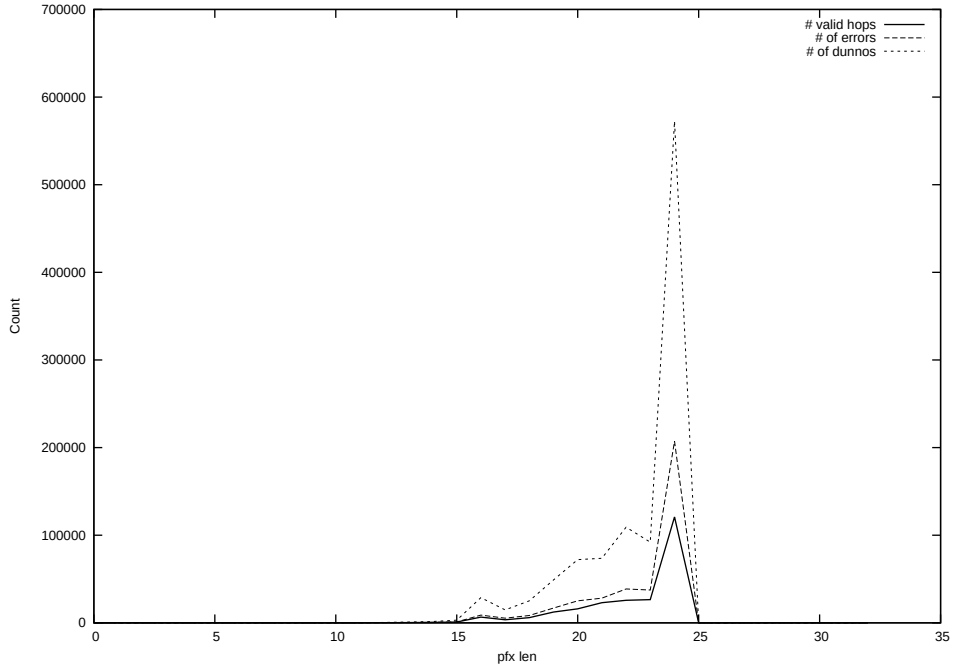
2013-02-18



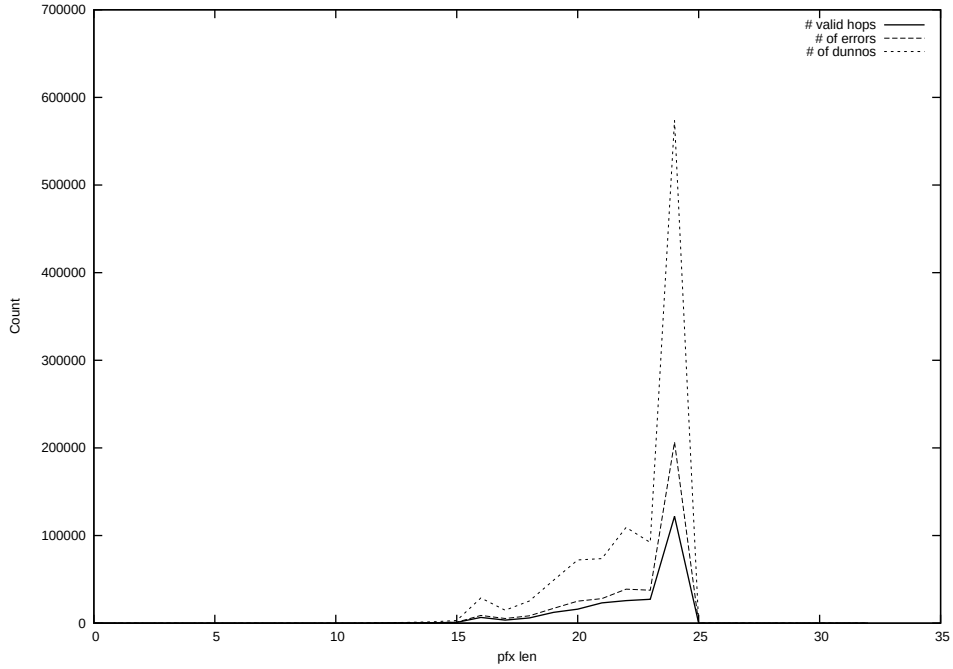
2013-02-19



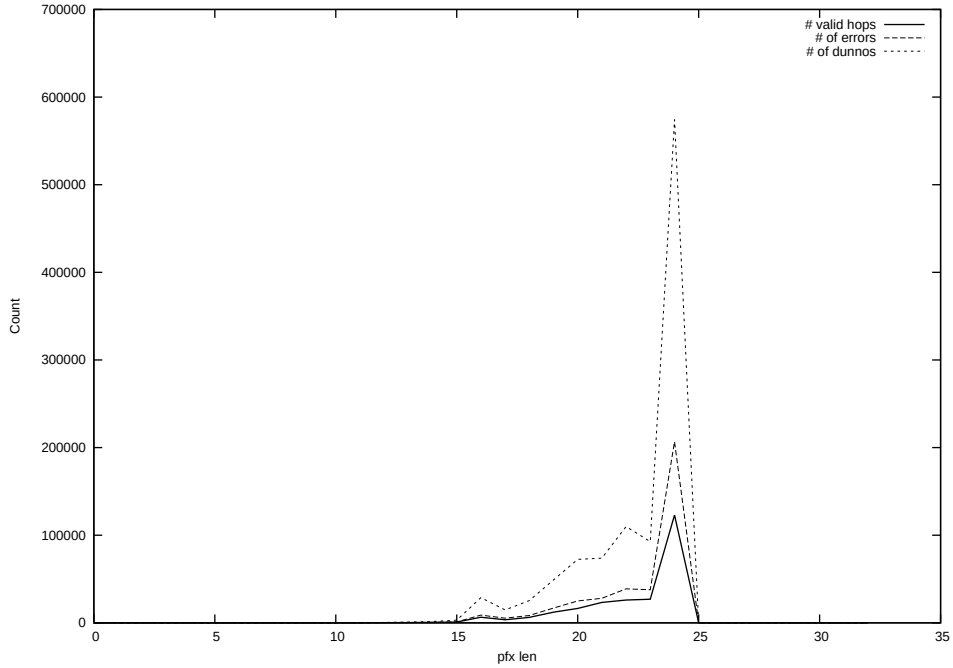
2013-02-20



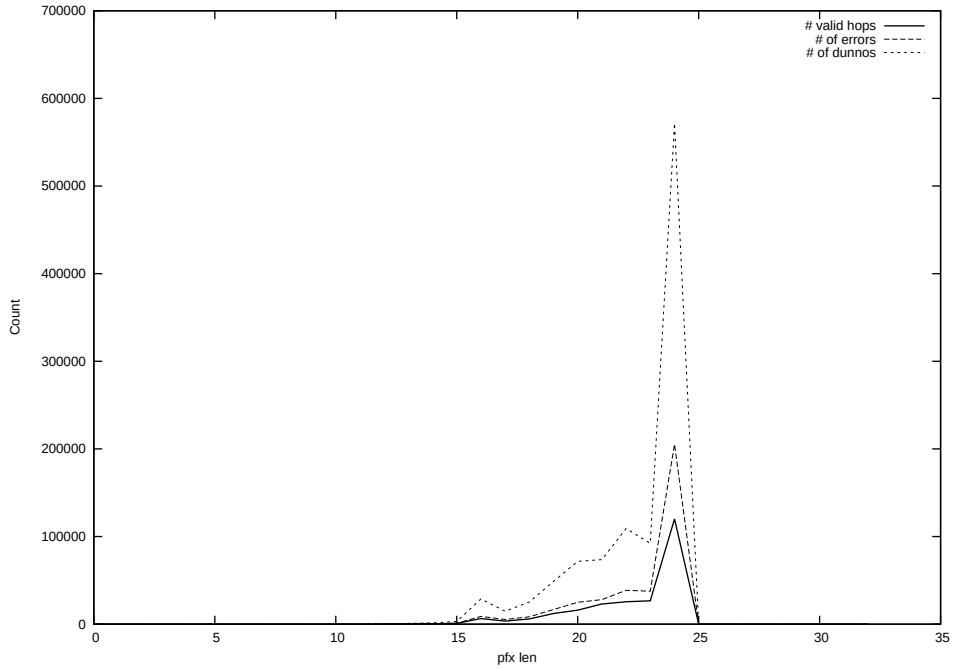
2013-02-21



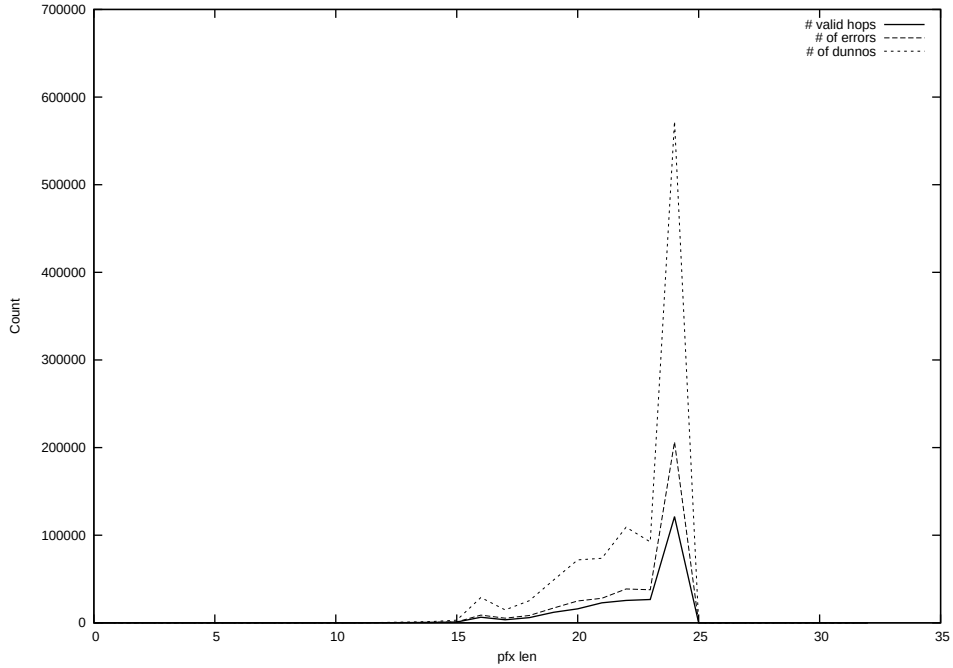
2013-02-22



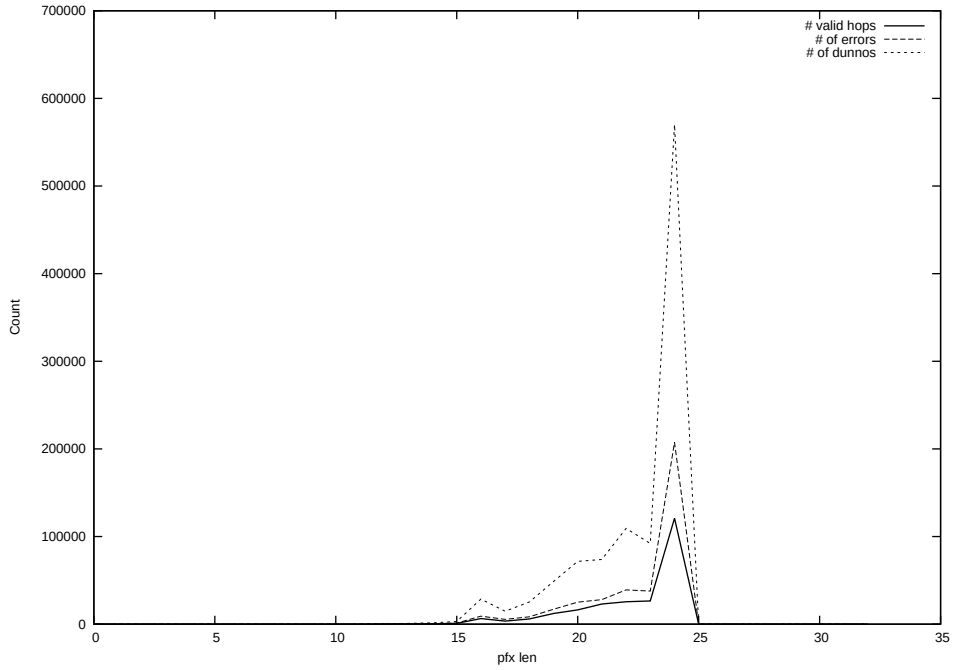
2013-02-23



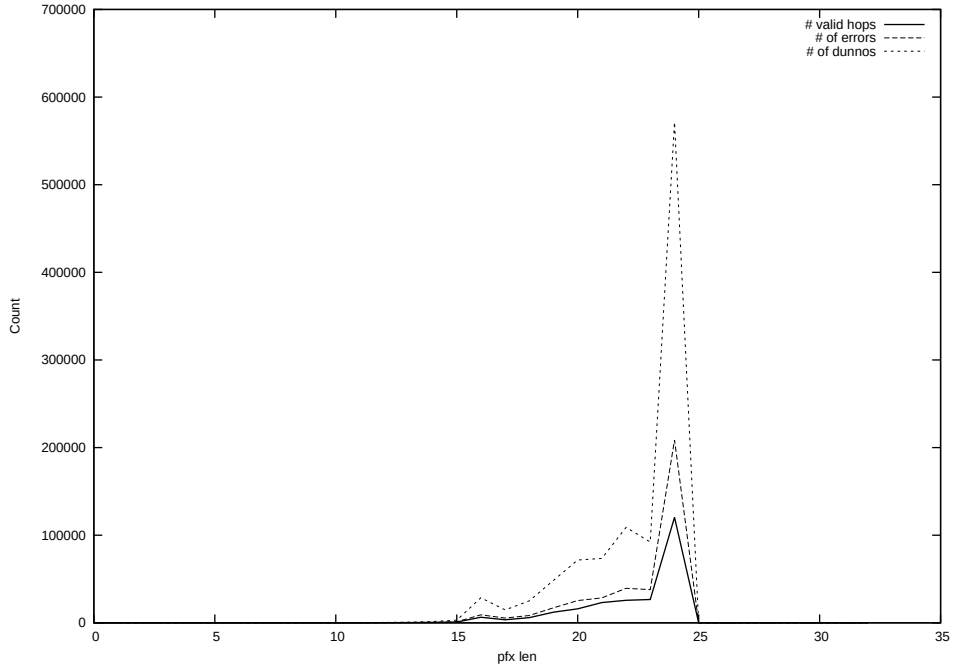
2013-02-24



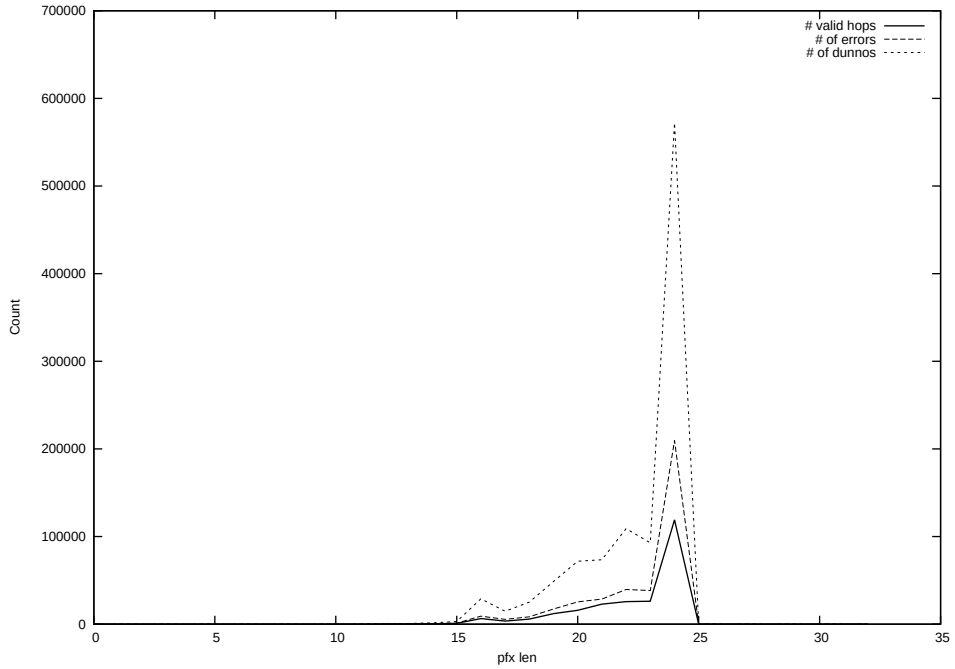
2013-02-25



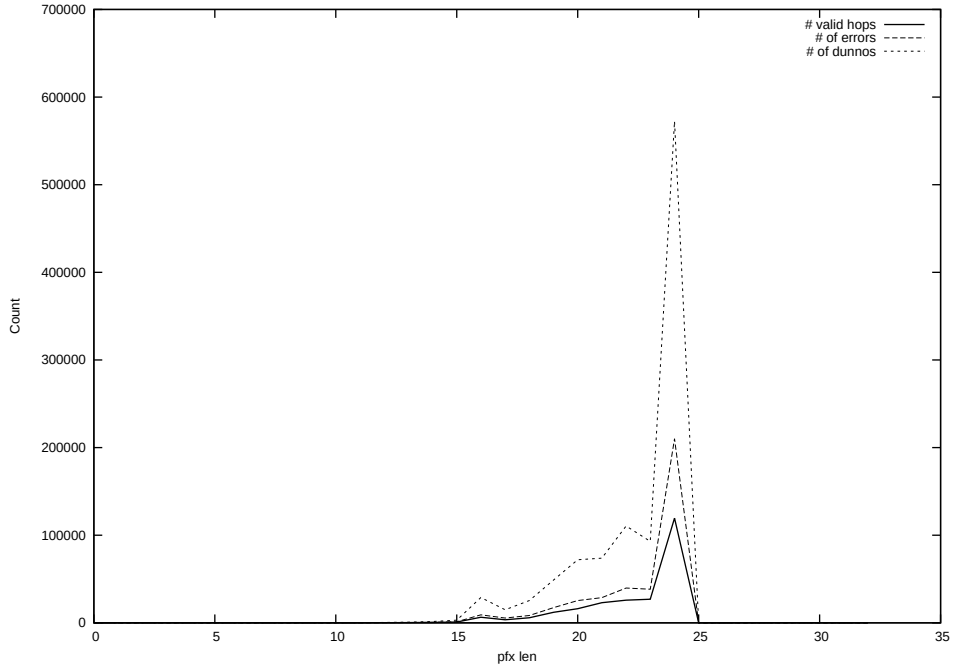
2013-02-26



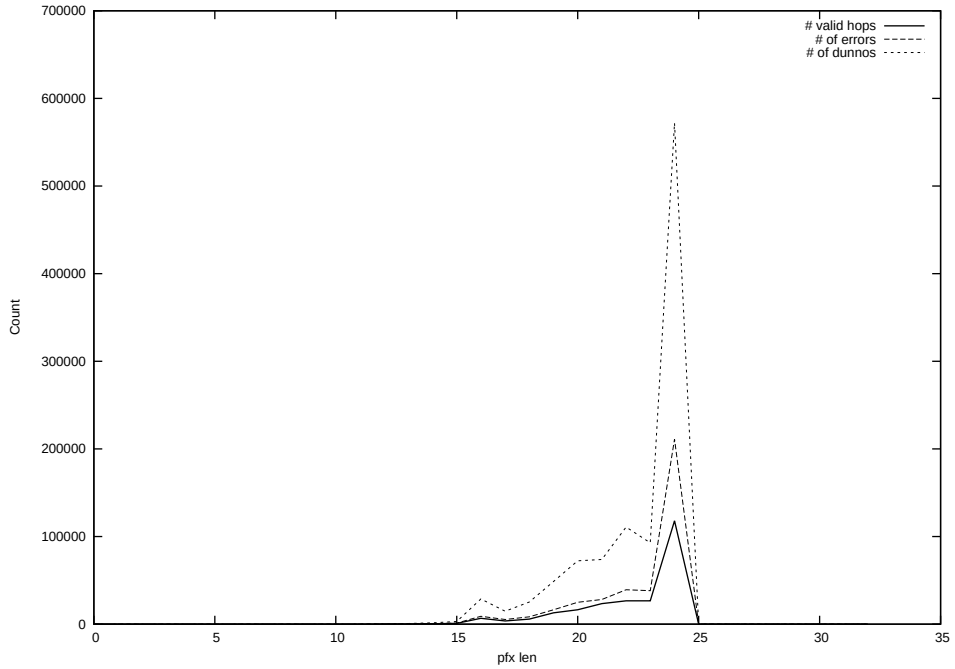
2013-02-27



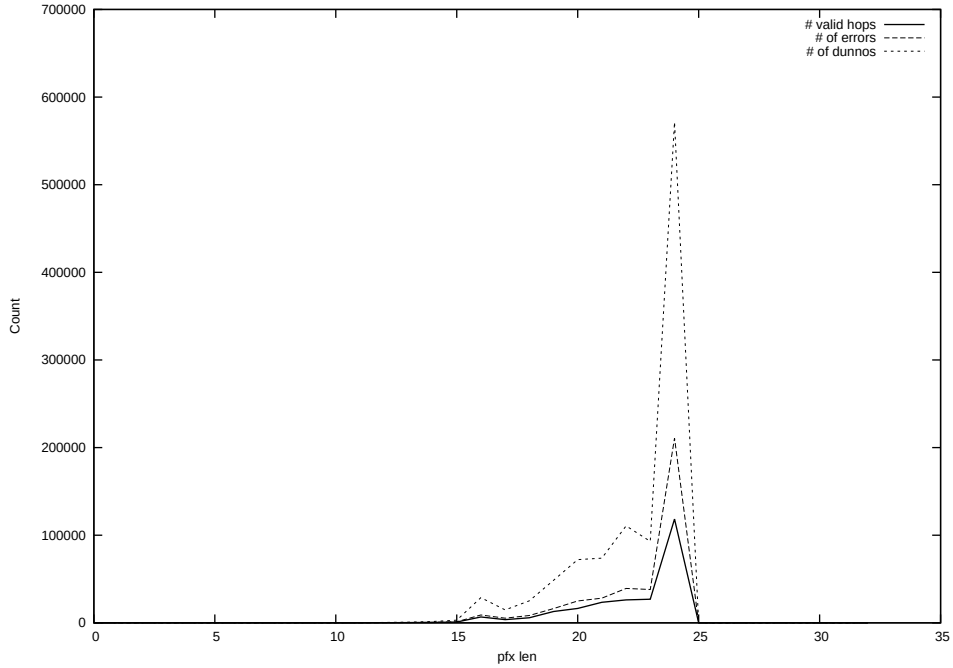
2013-02-28



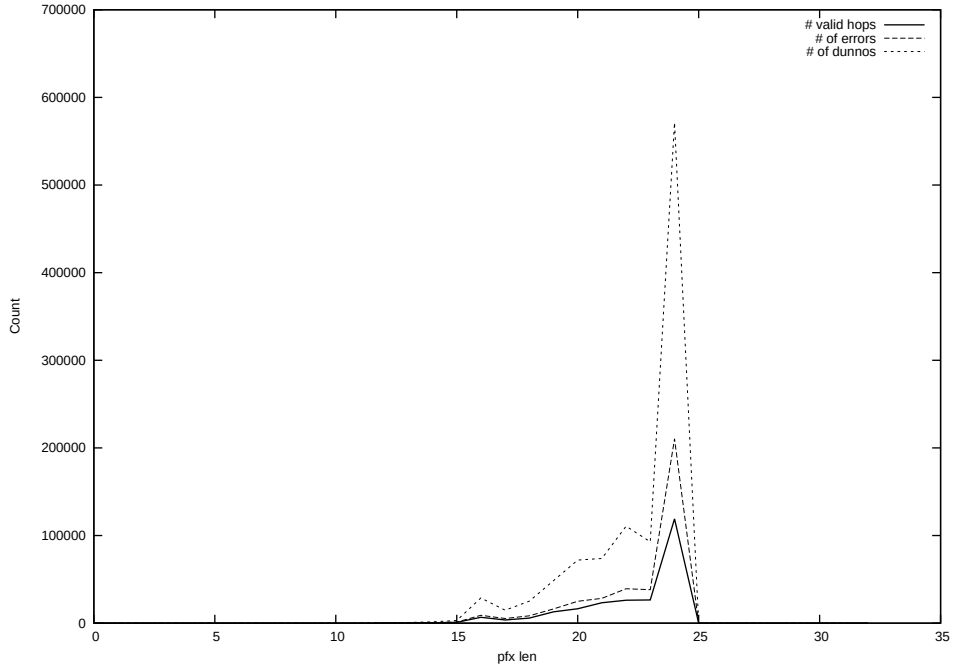
2013-03-01



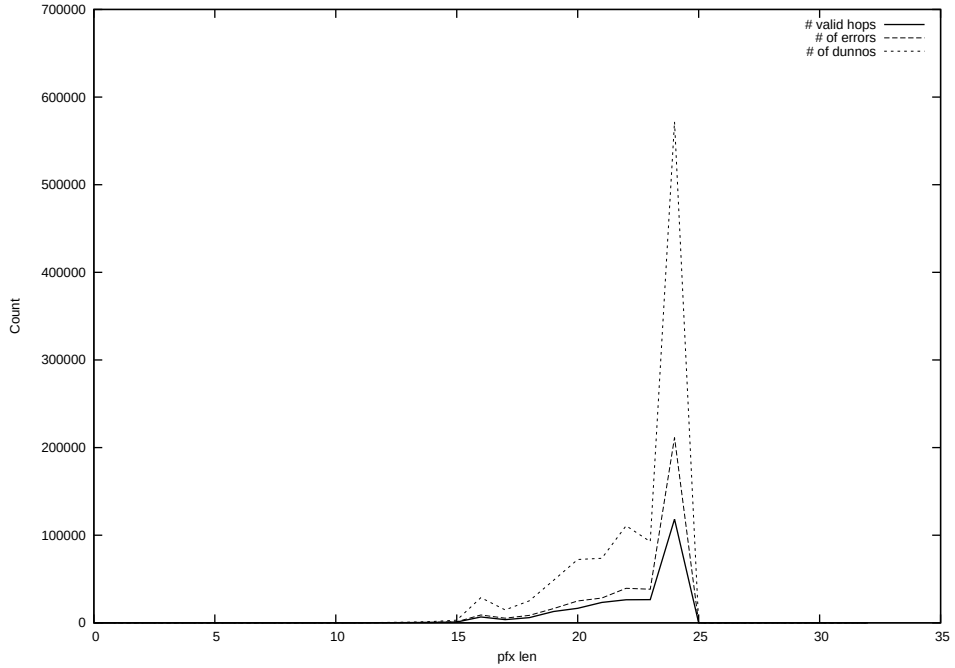
2013-03-02



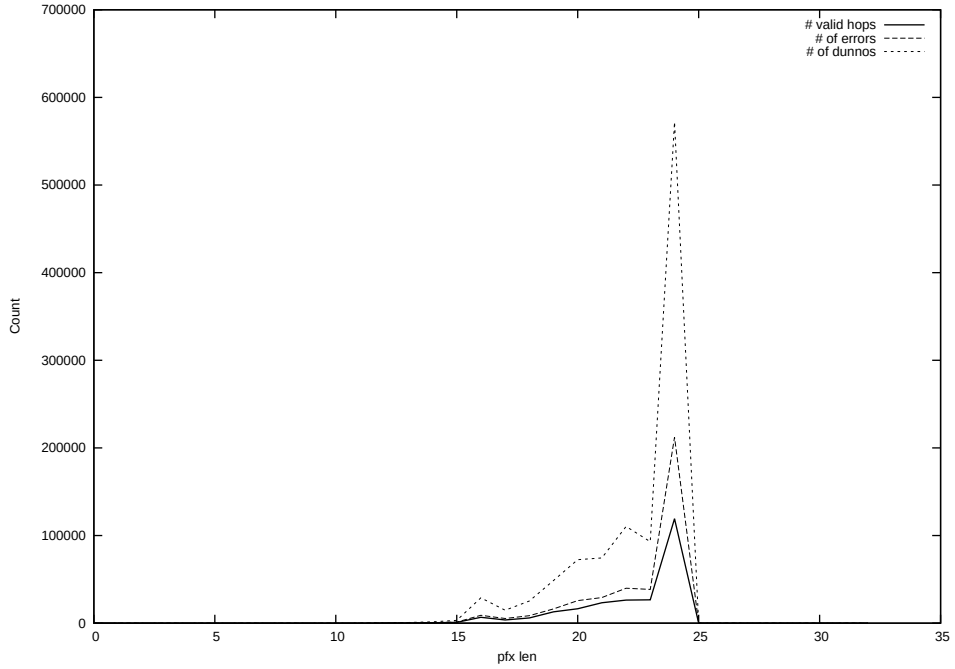
2013-03-03



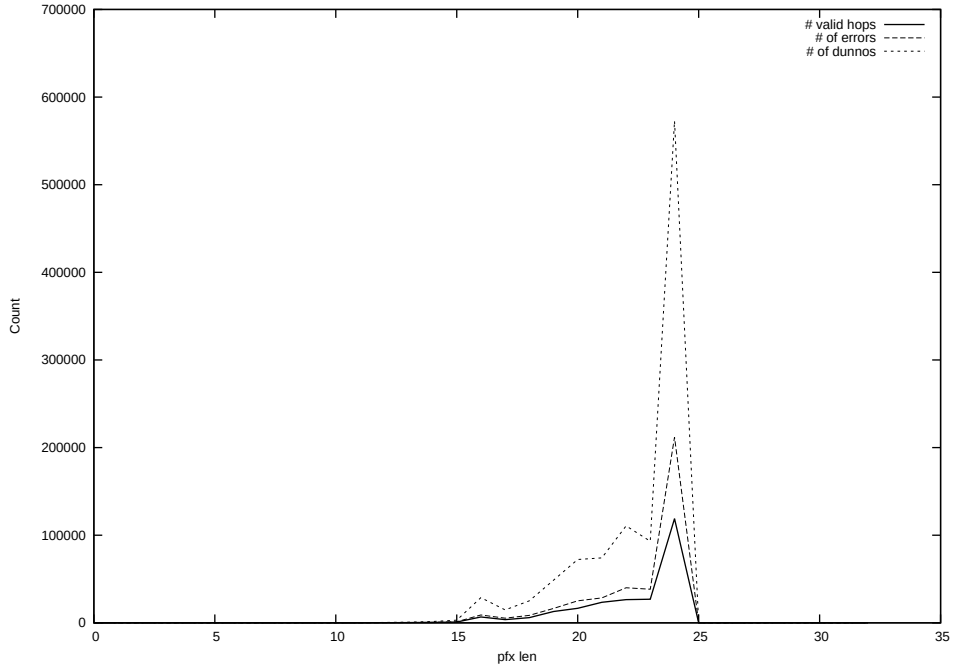
2013-03-04



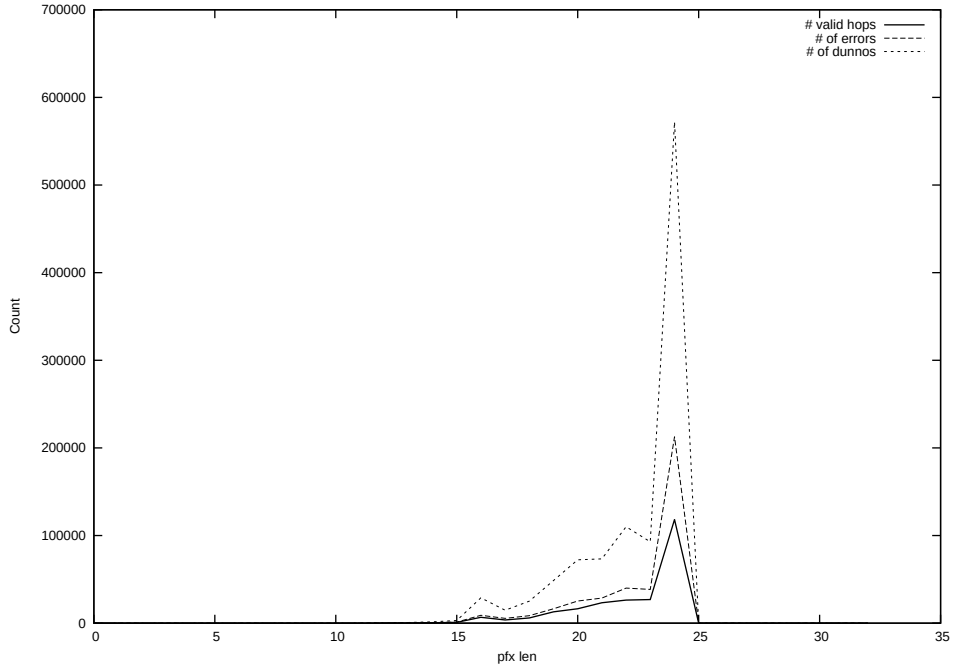
2013-03-05



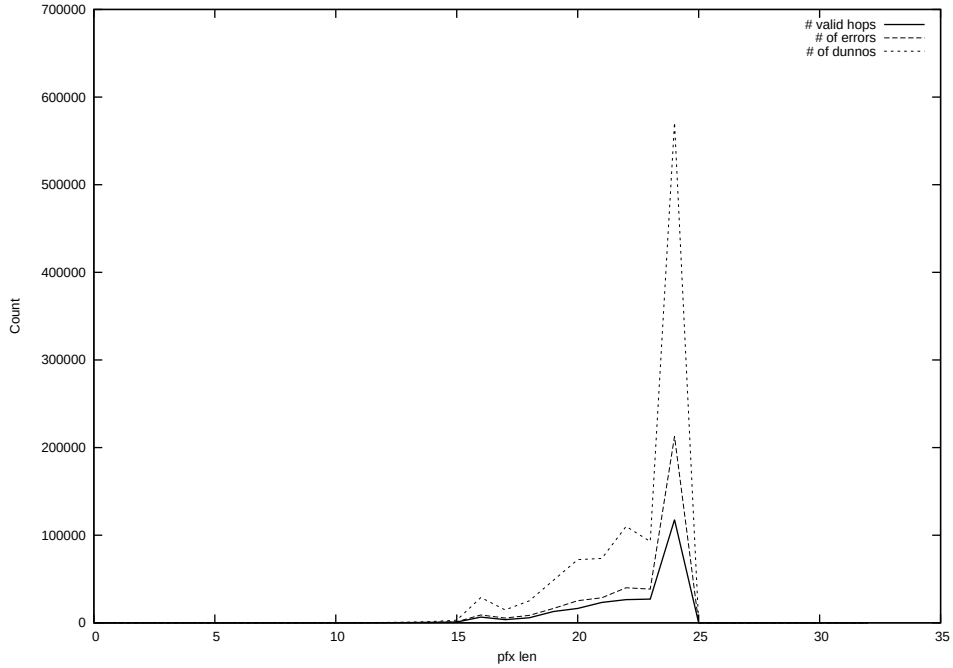
2013-03-06



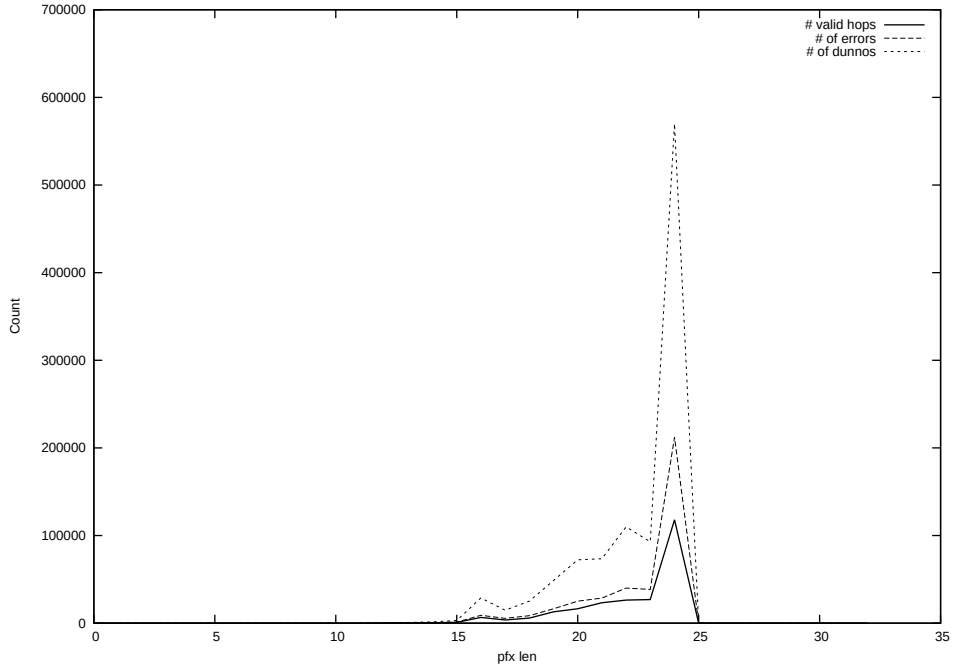
2013-03-07



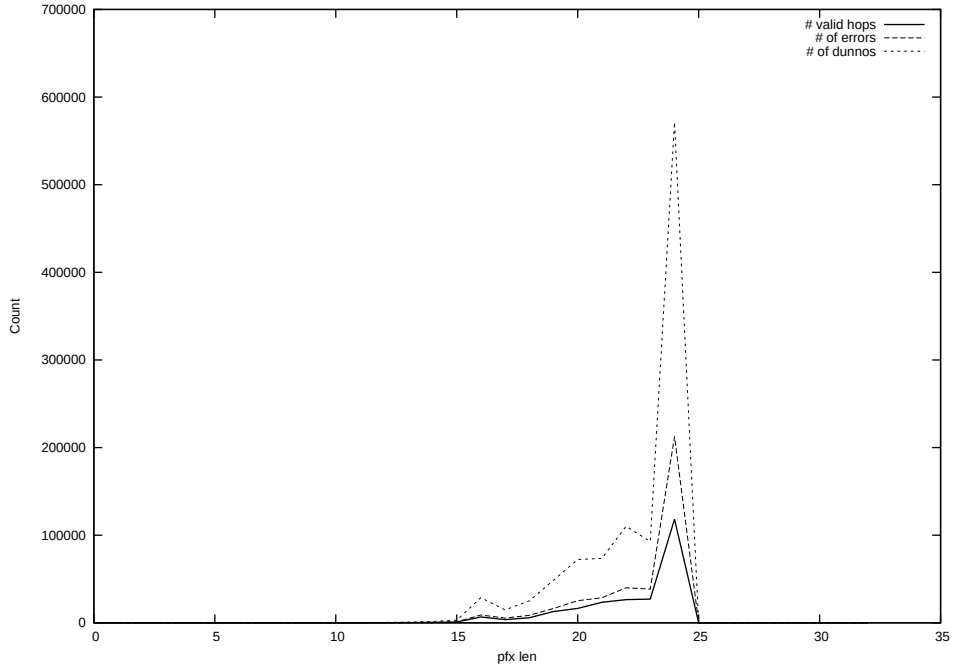
2013-03-08



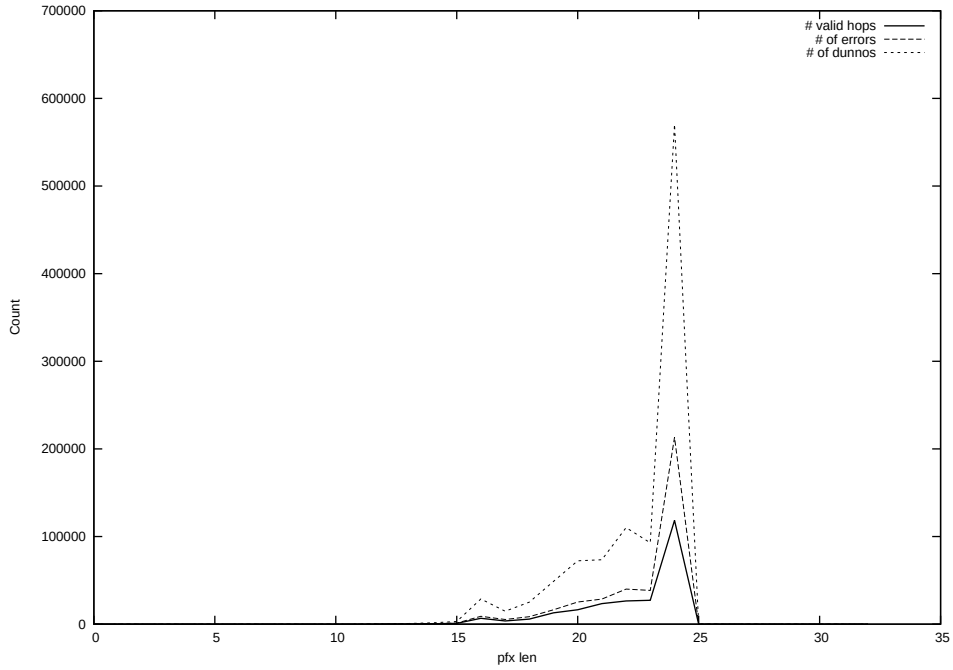
2013-03-09



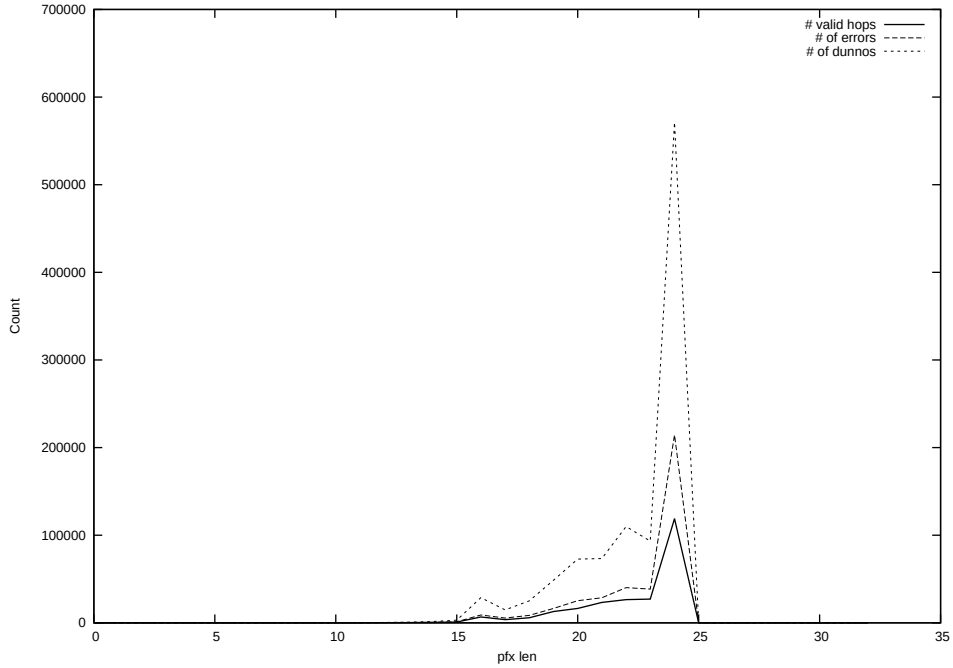
2013-03-10



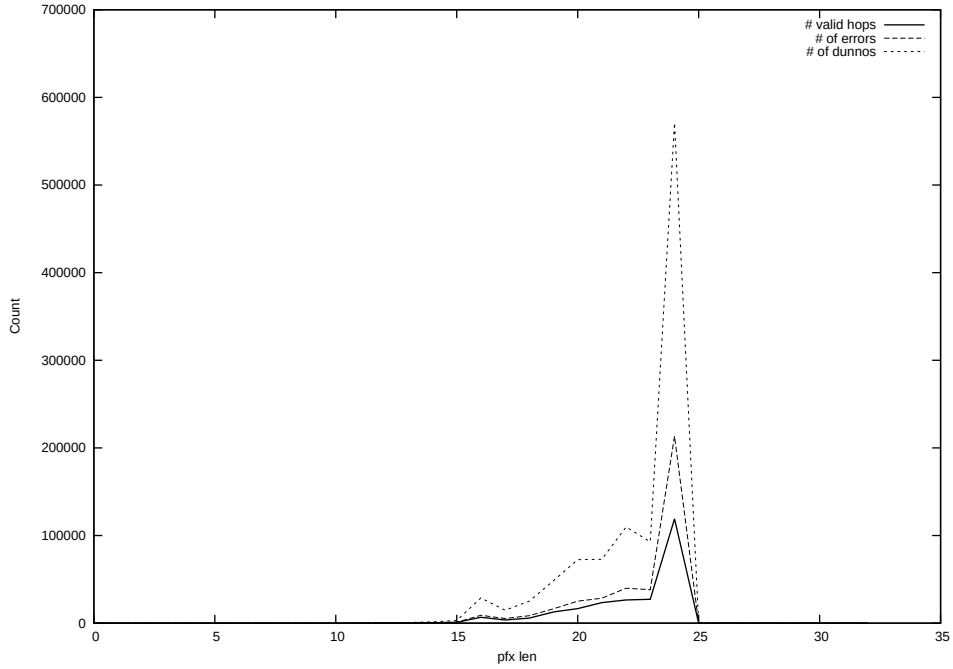
2013-03-11



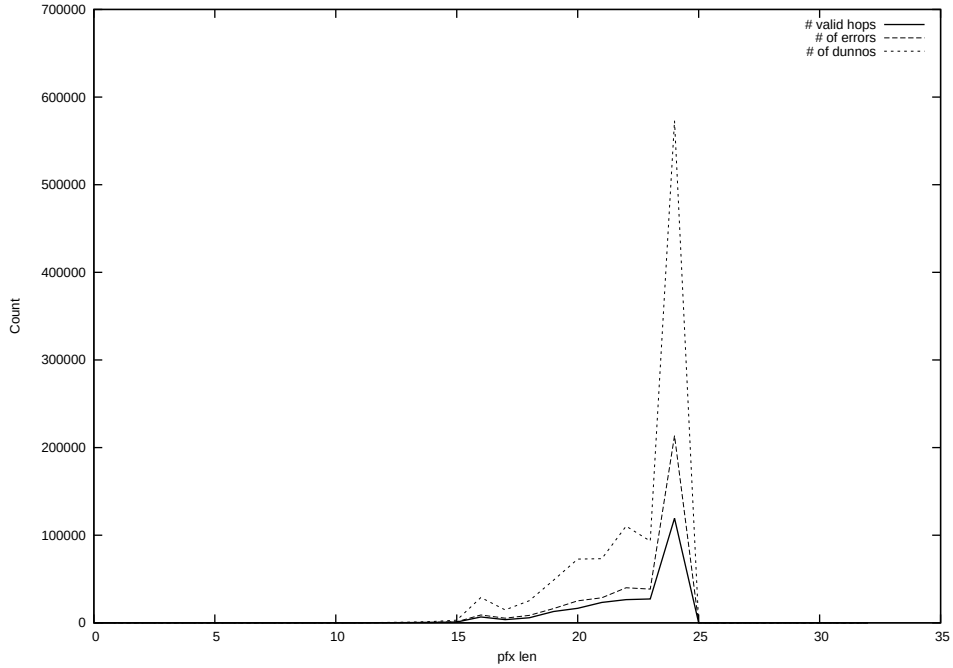
2013-03-12



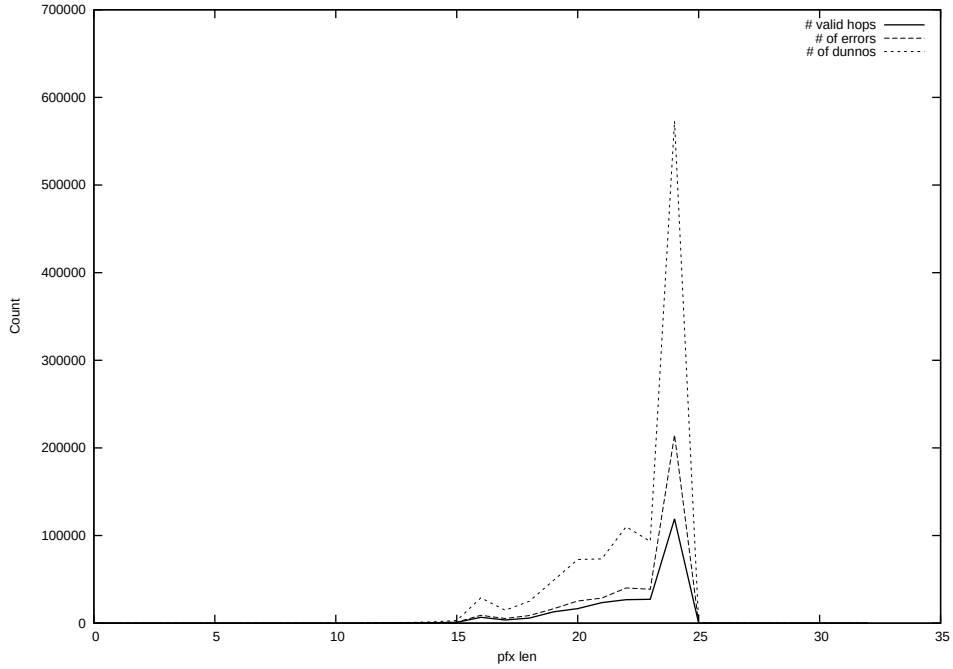
2013-03-13



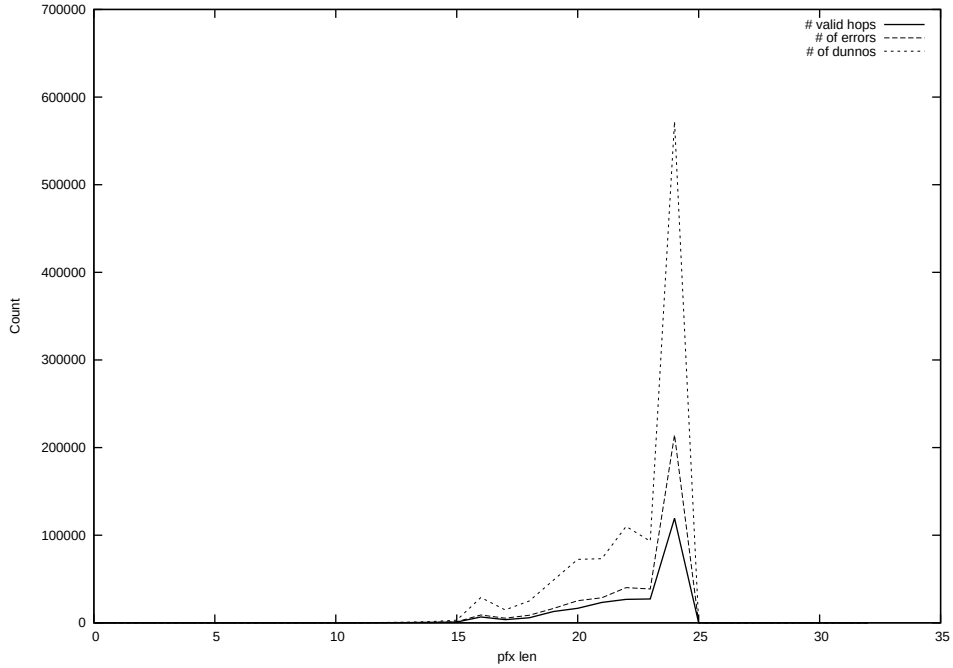
2013-03-14



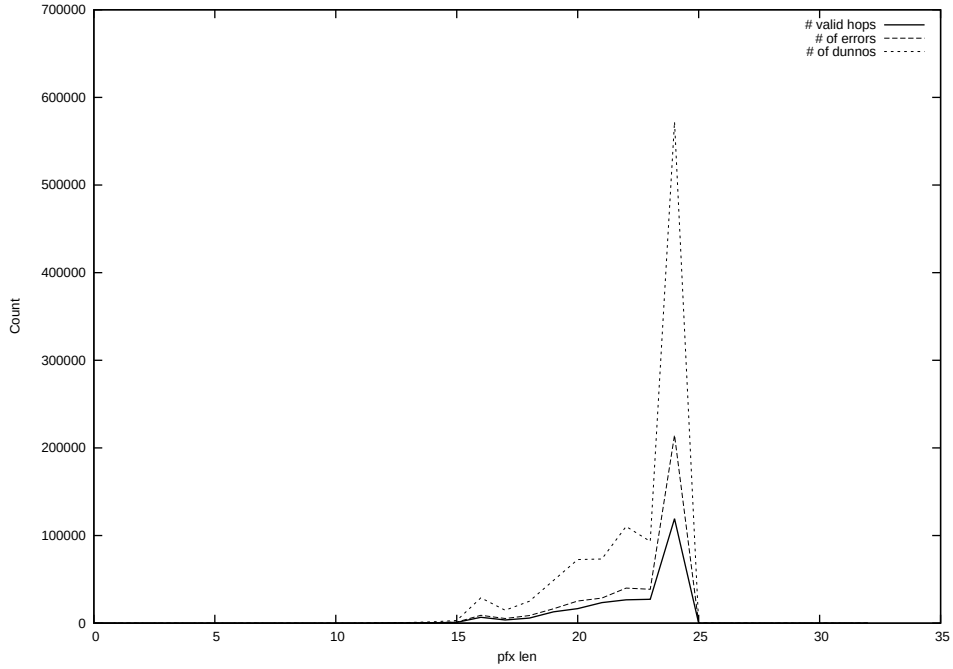
2013-03-15



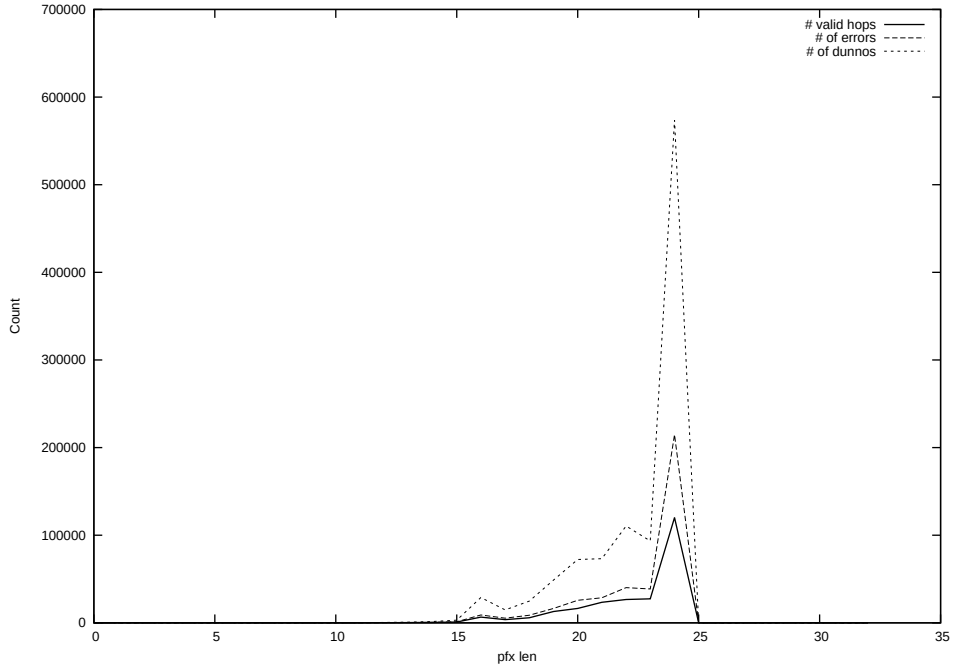
2013-03-16



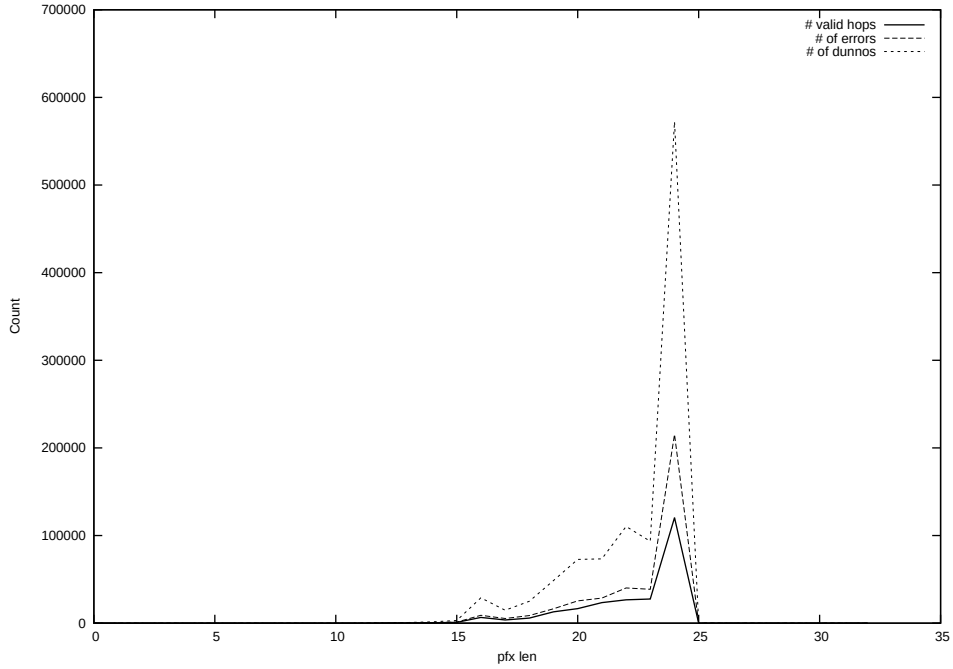
2013-03-17



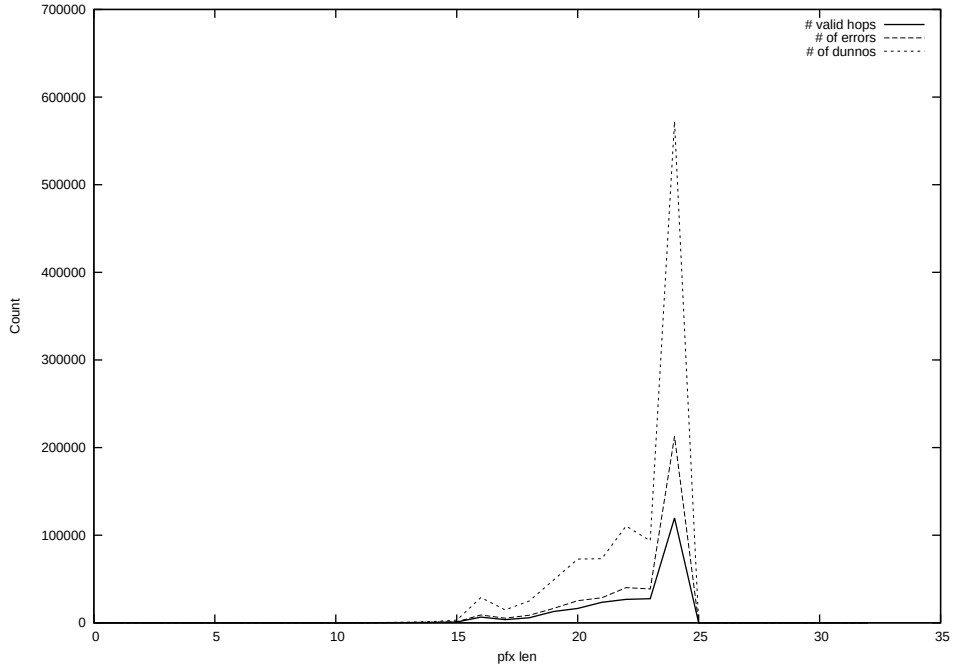
2013-03-18



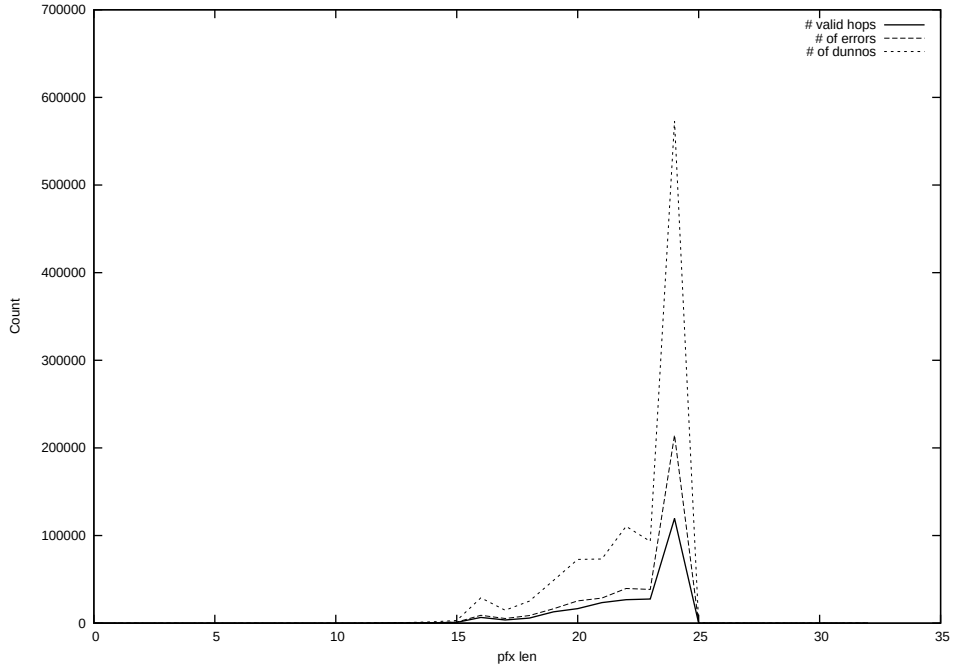
2013-03-19



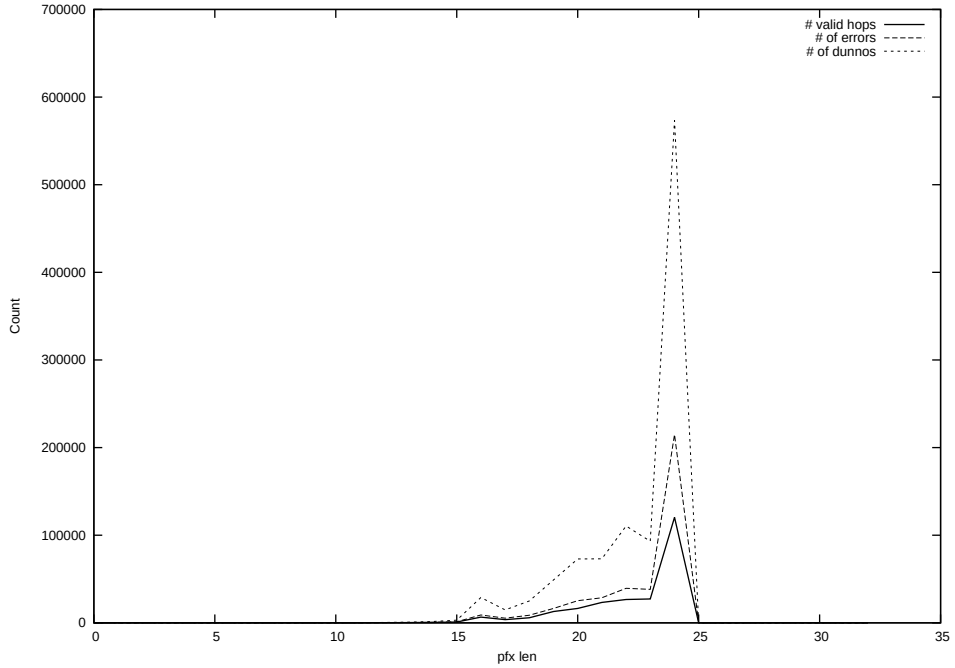
2013-03-20



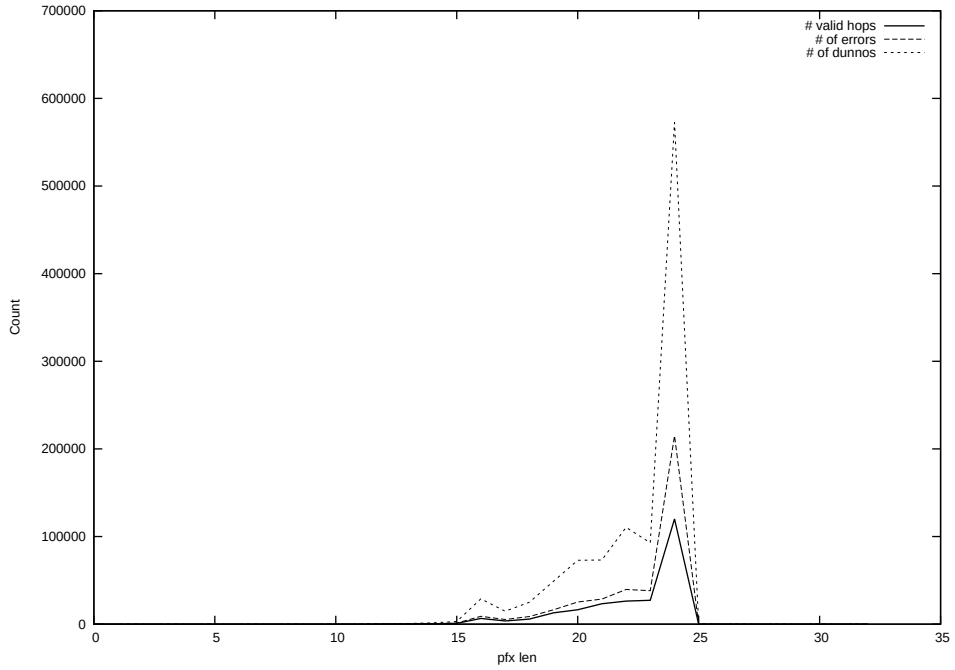
2013-03-21



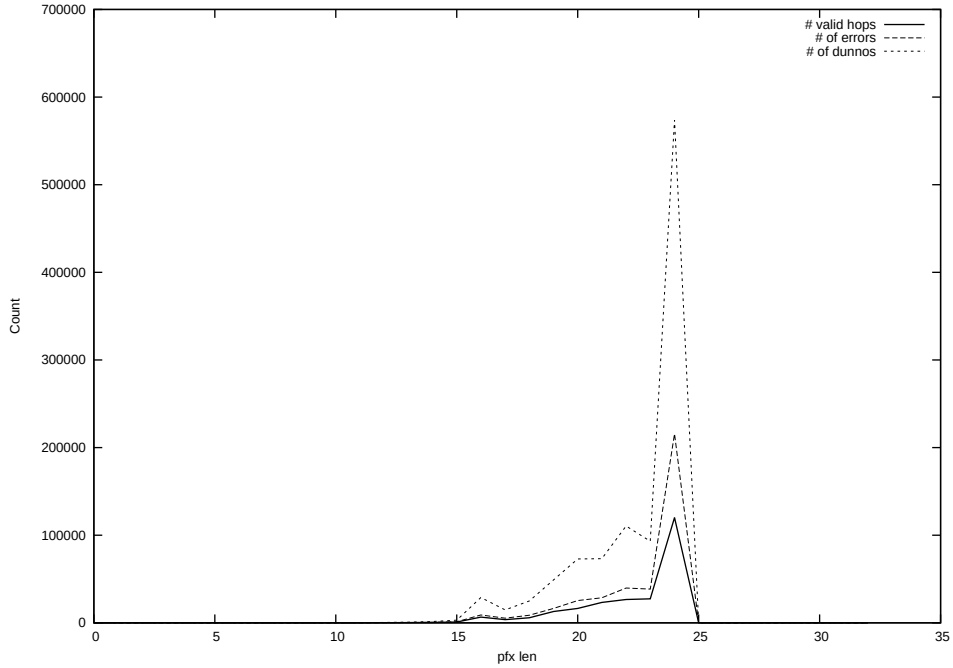
2013-03-22



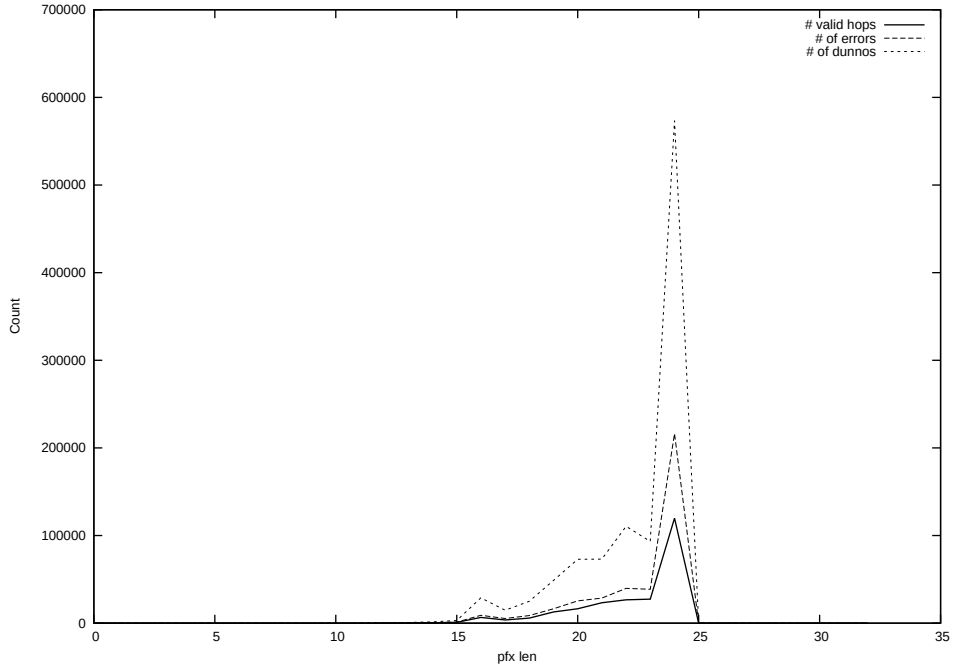
2013-03-23



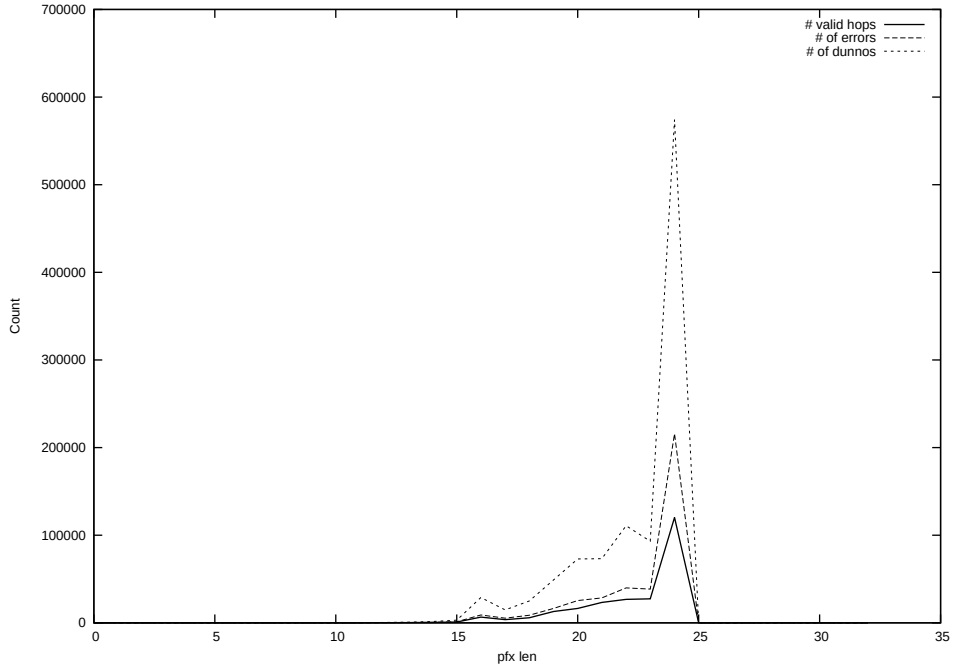
2013-03-24



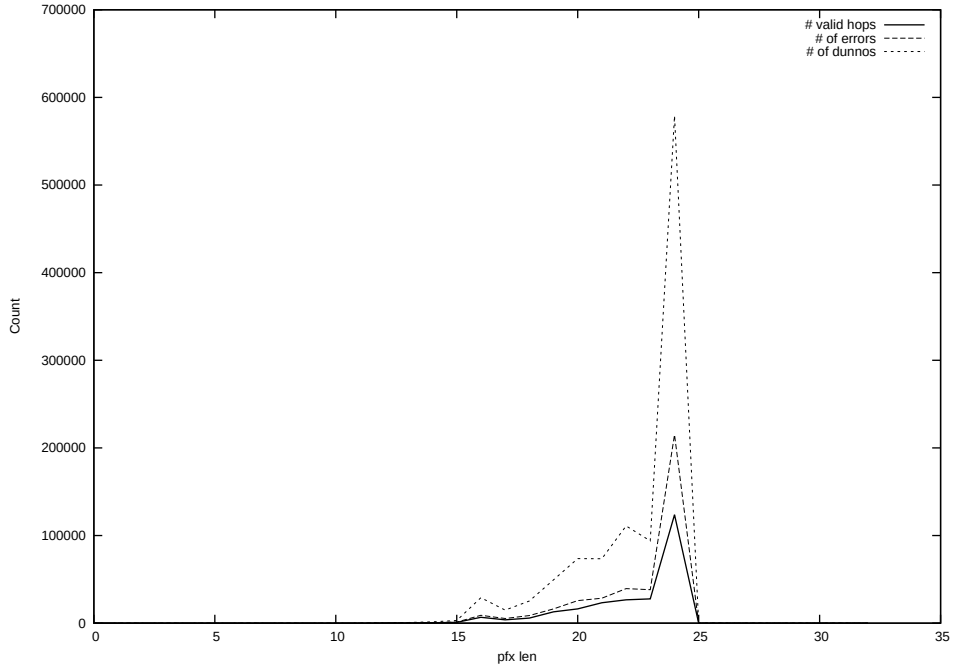
2013-03-25



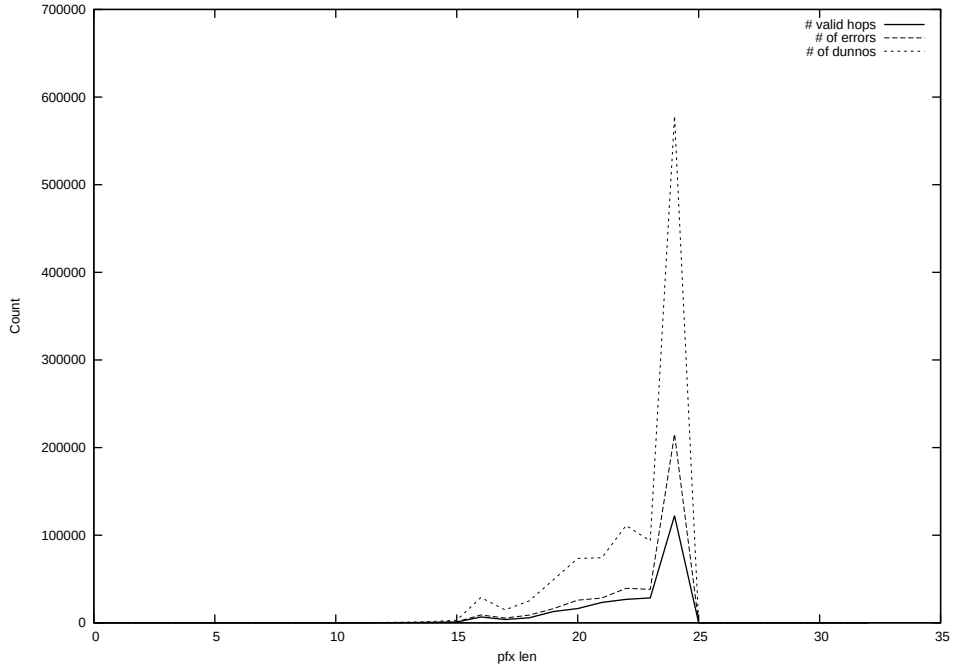
2013-03-26



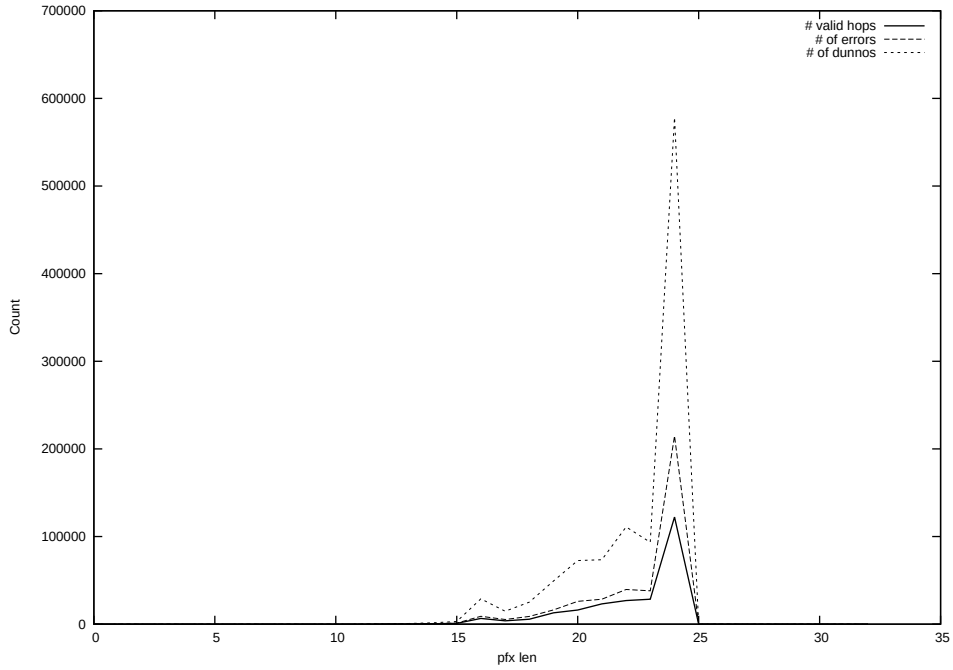
2013-03-27



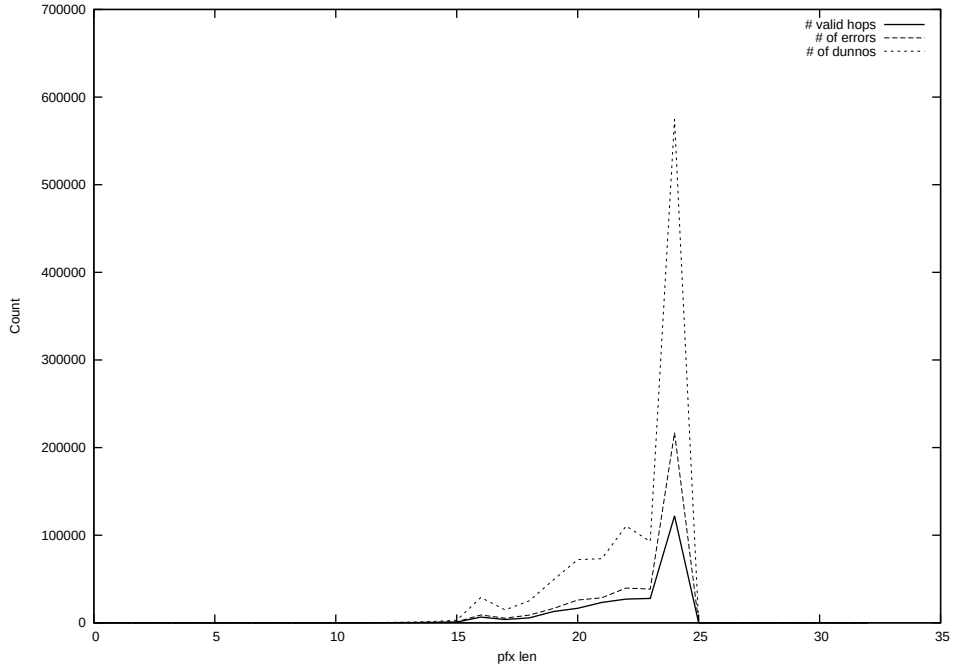
2013-03-28



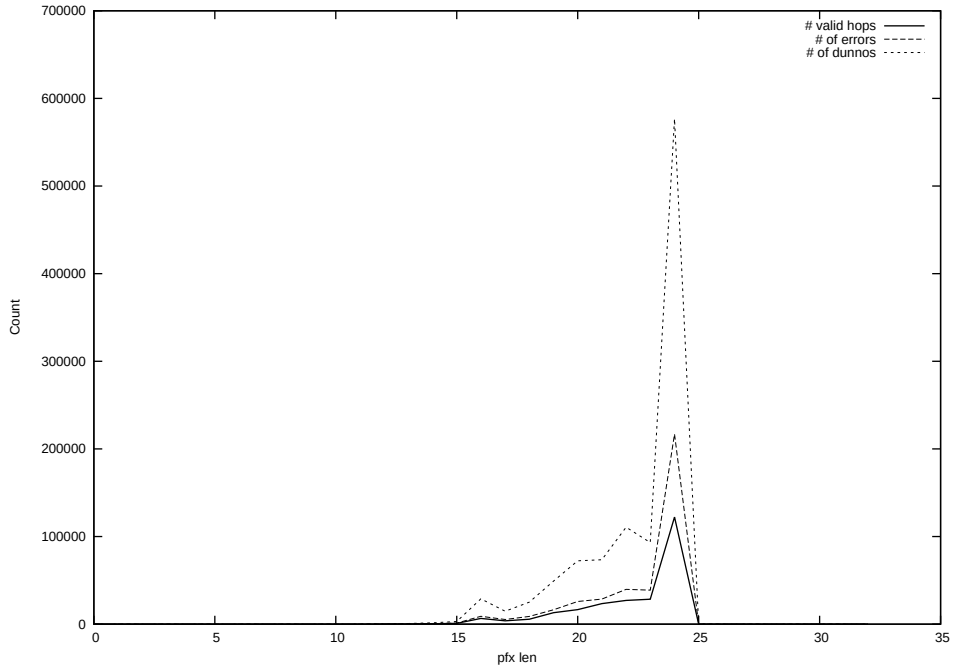
2013-03-29



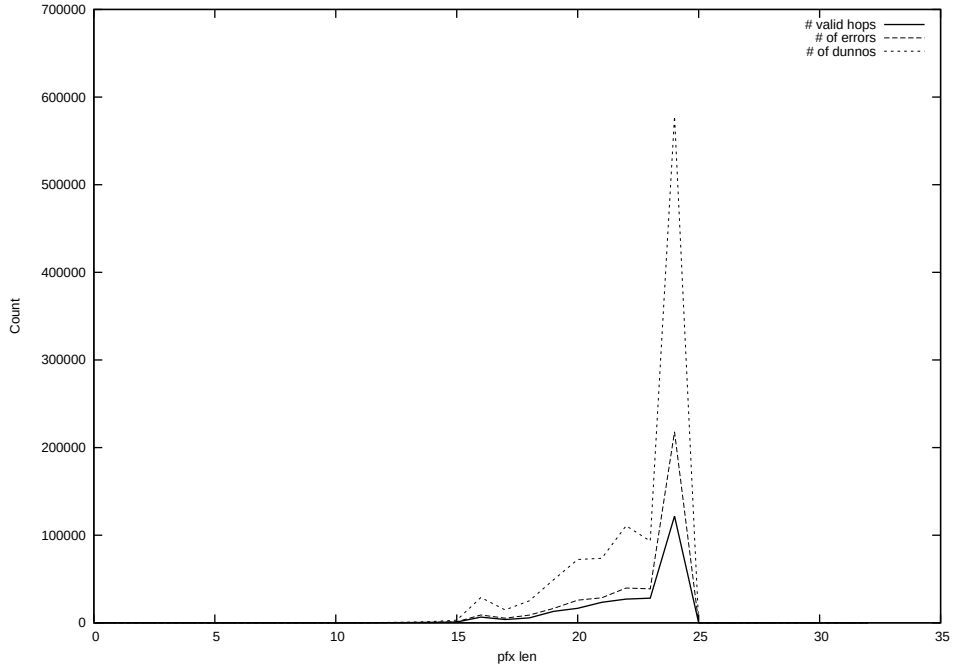
2013-03-30



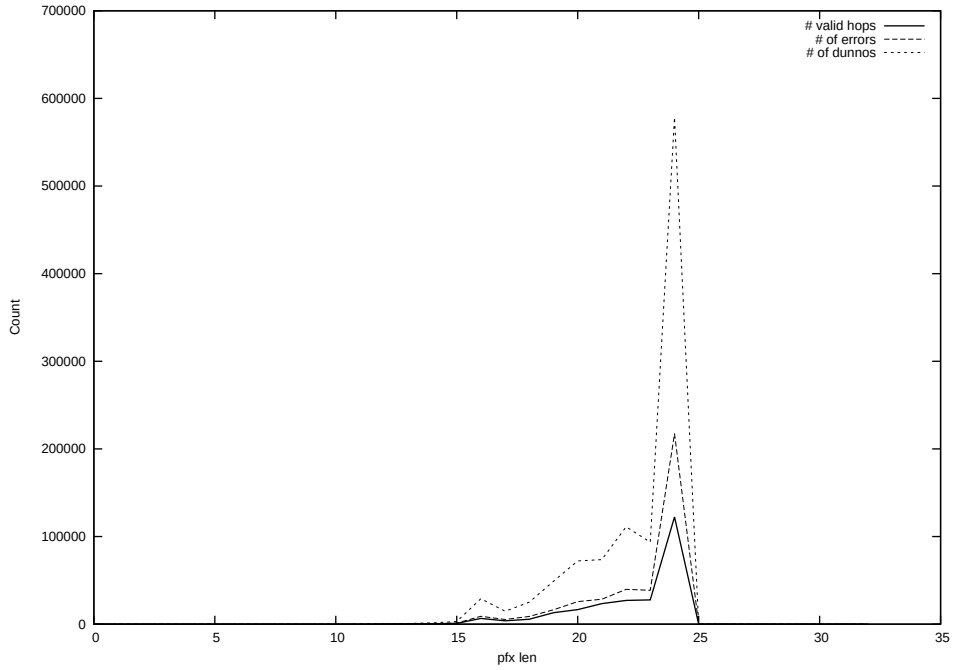
2013-03-31



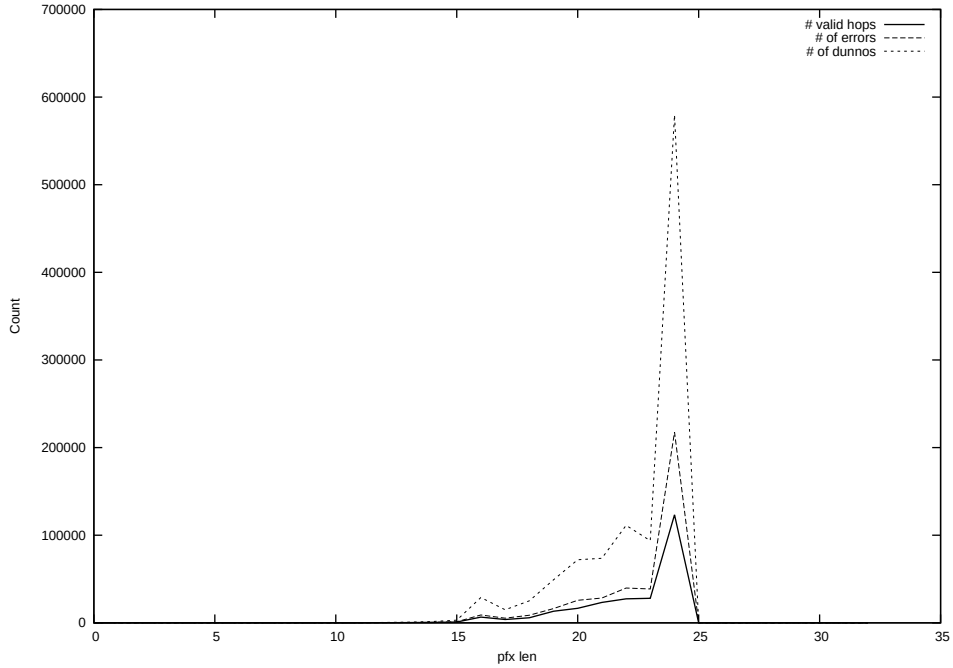
2013-04-01



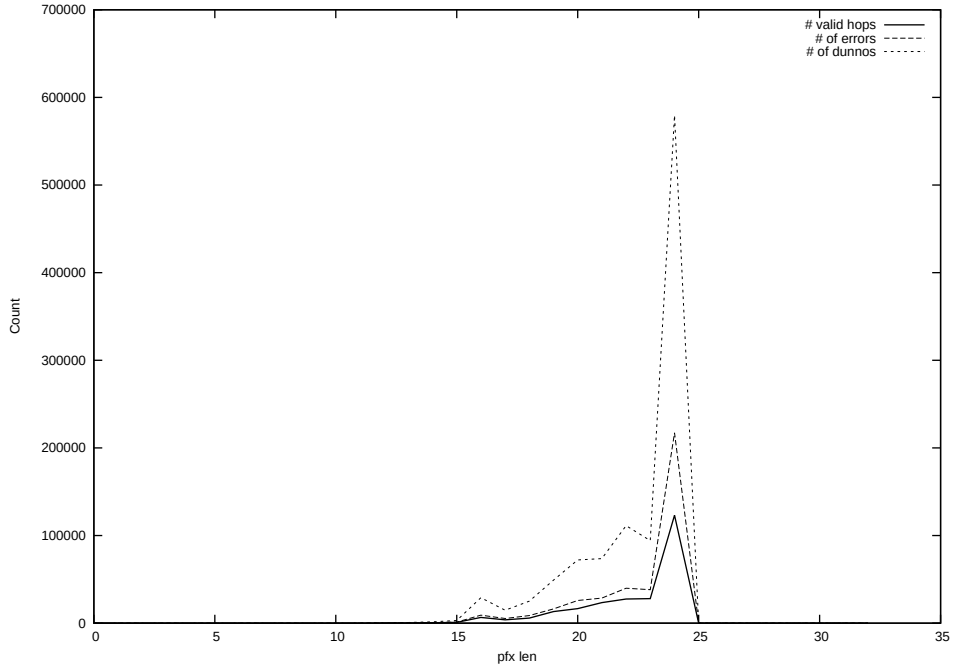
2013-04-02



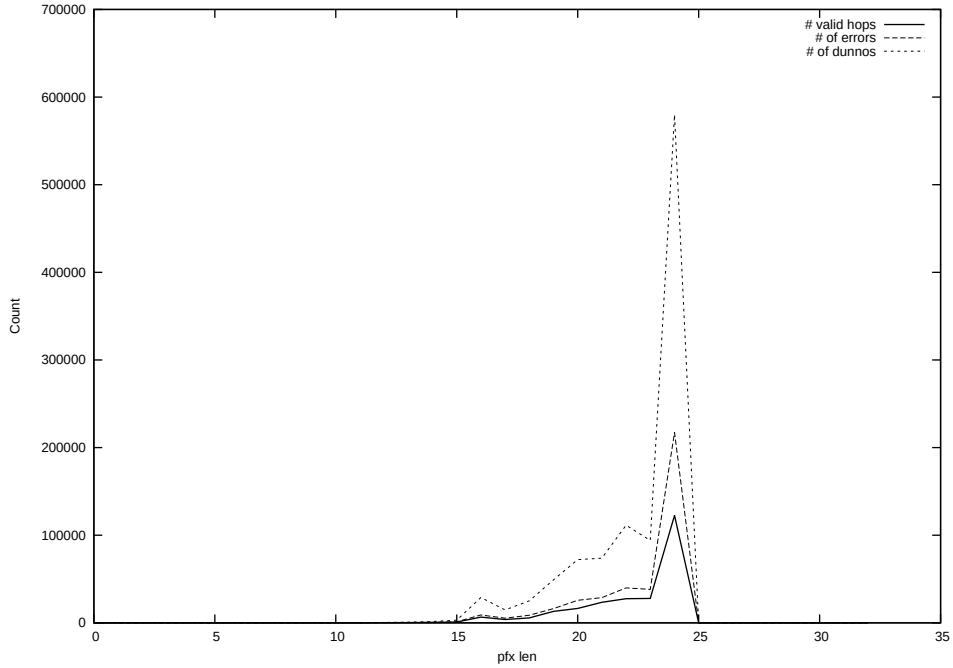
2013-04-03



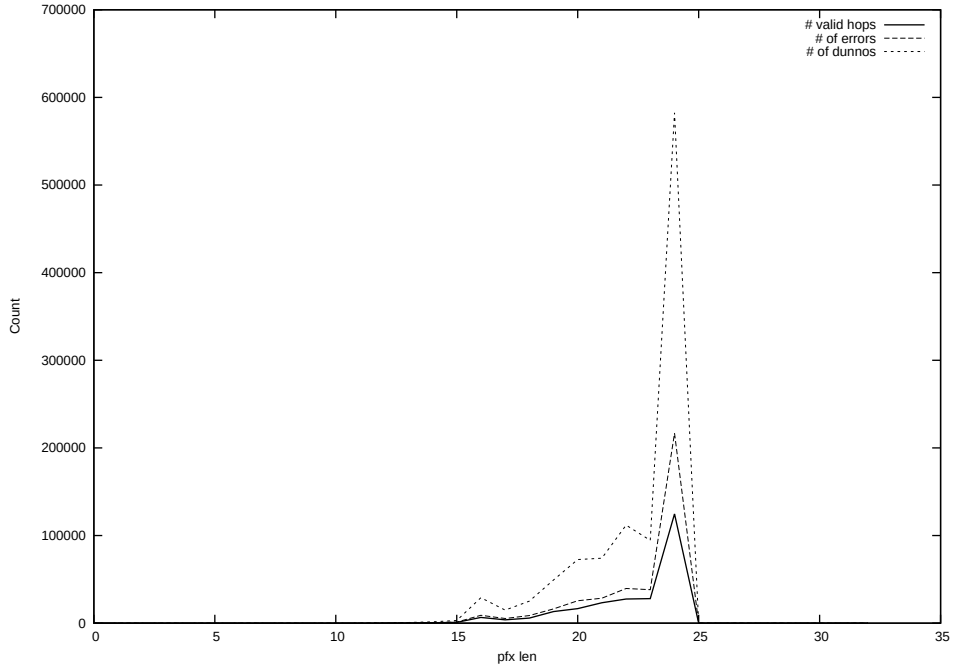
2013-04-04



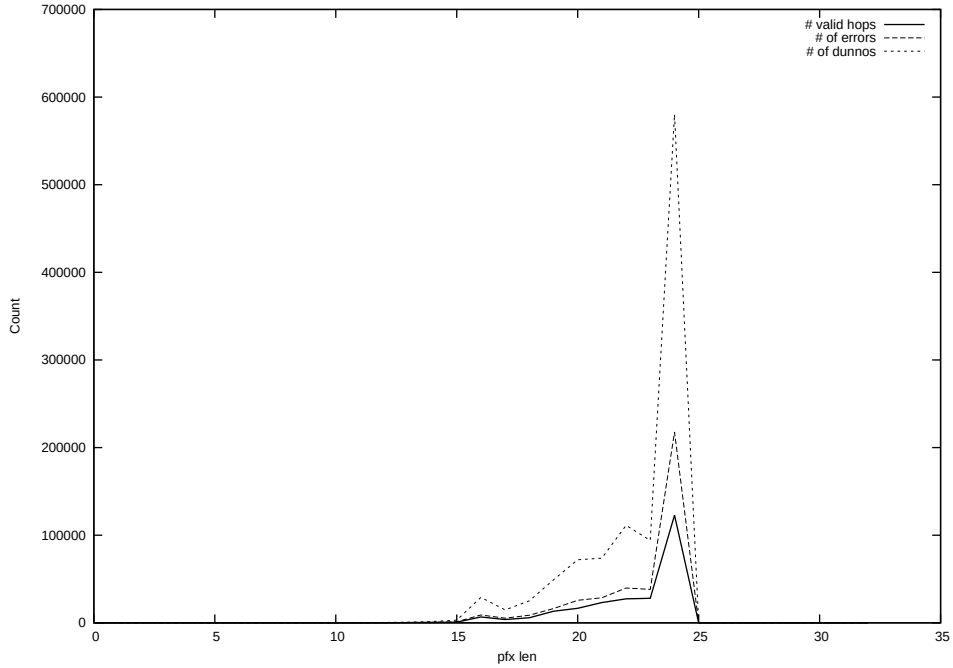
2013-04-05



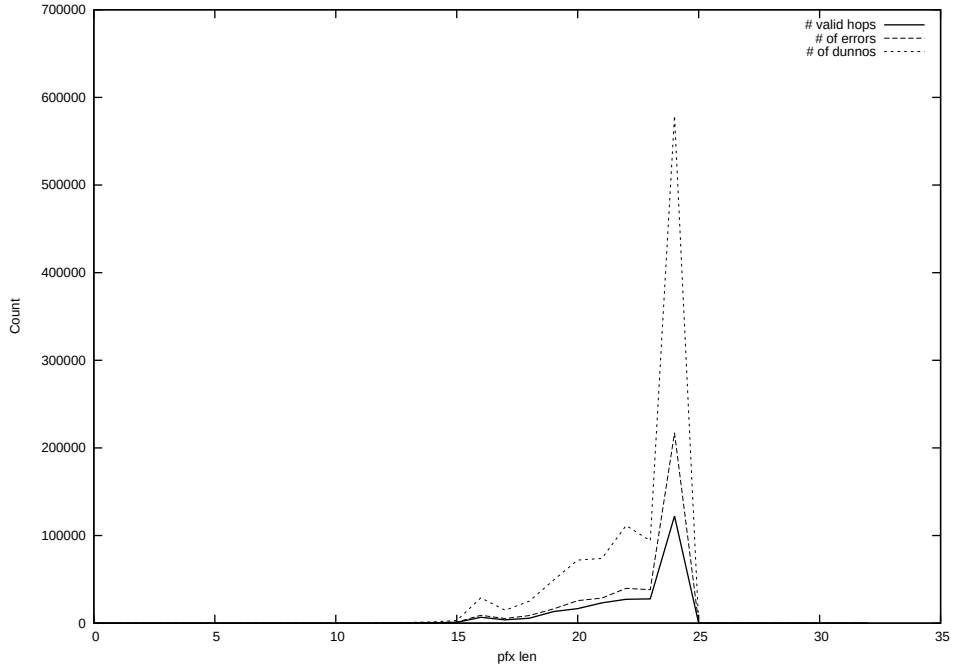
2013-04-06



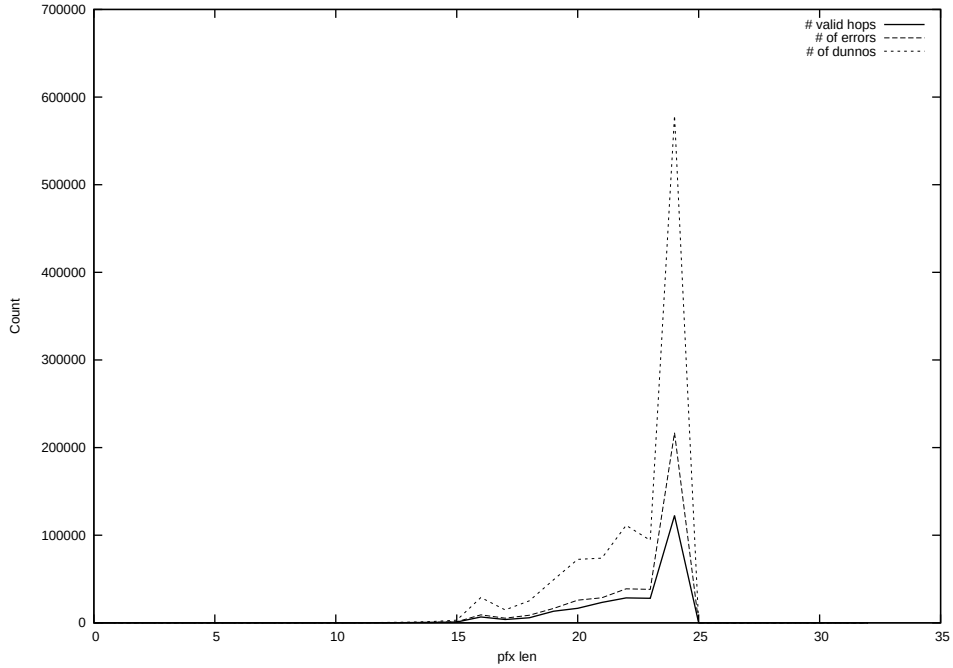
2013-04-07



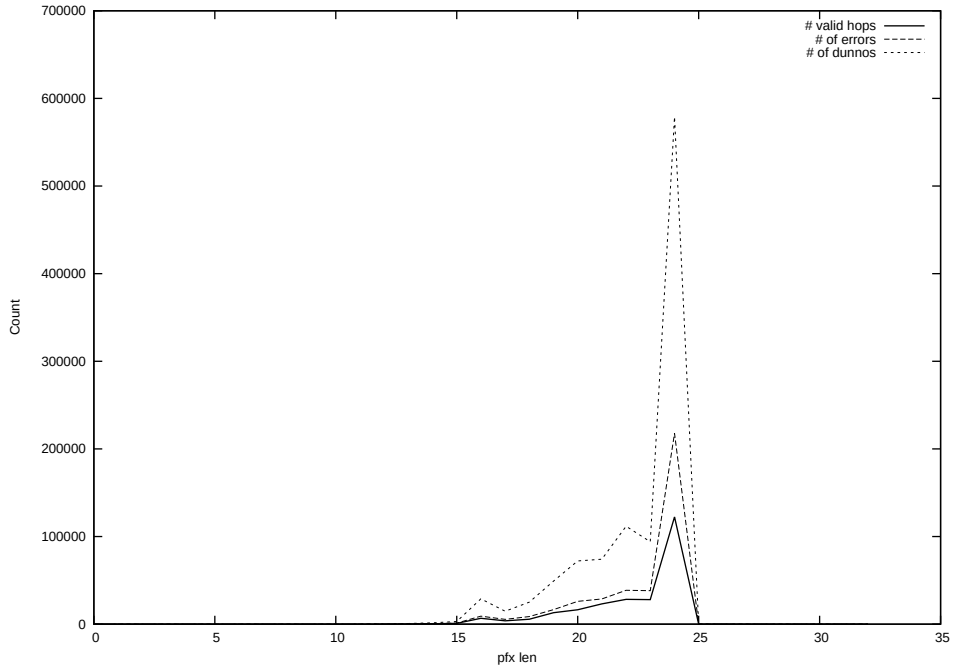
2013-04-08



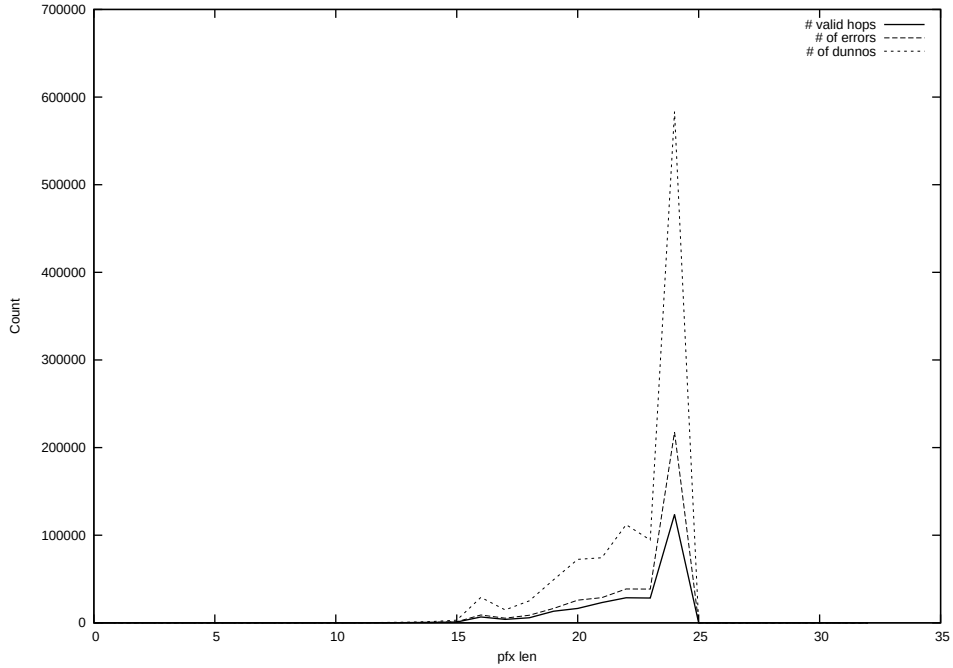
2013-04-09



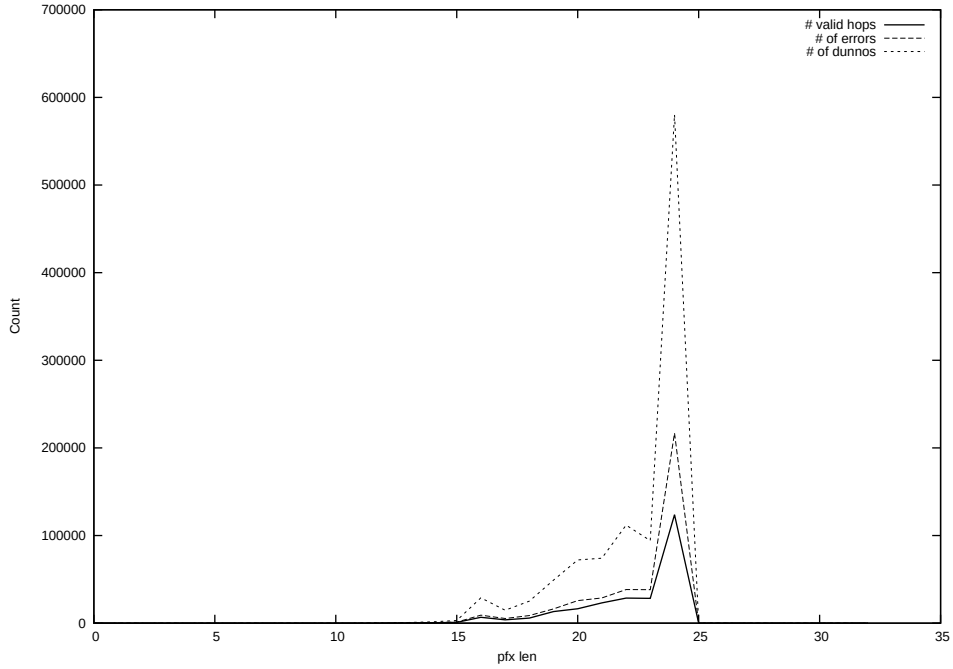
2013-04-10



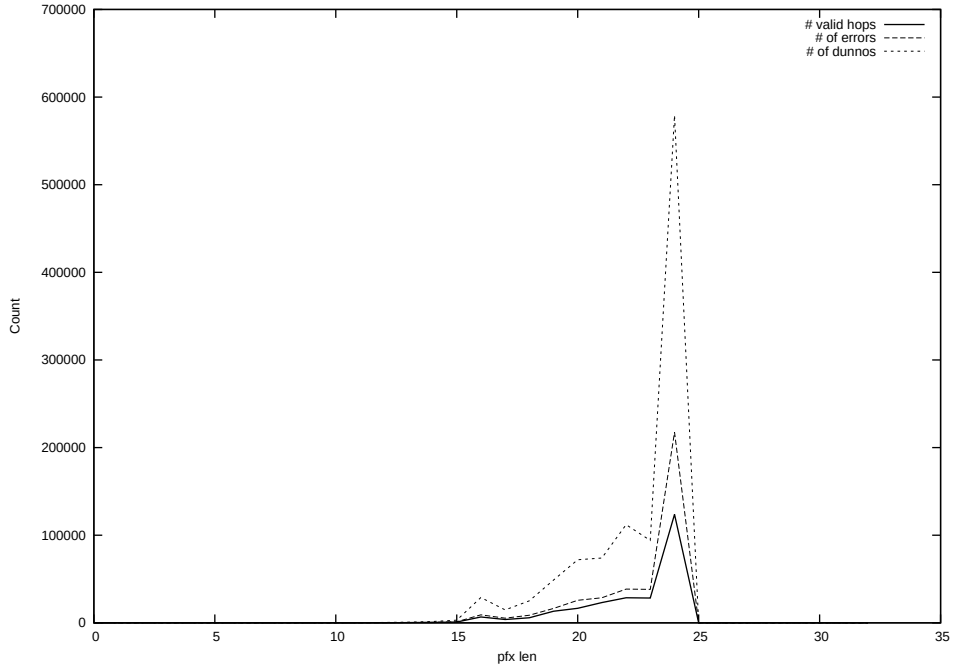
2013-04-11



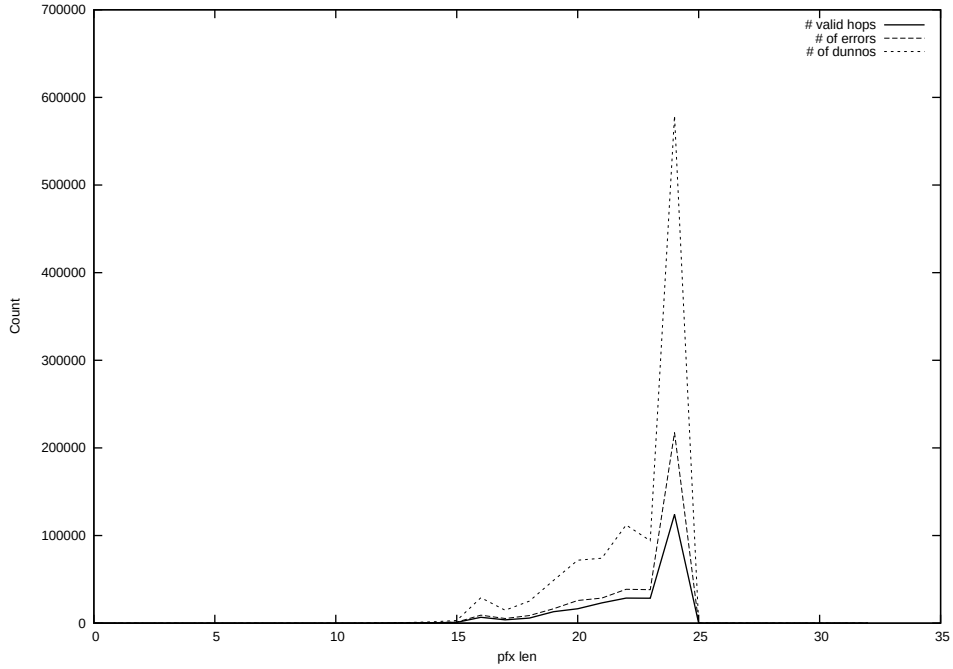
2013-04-12



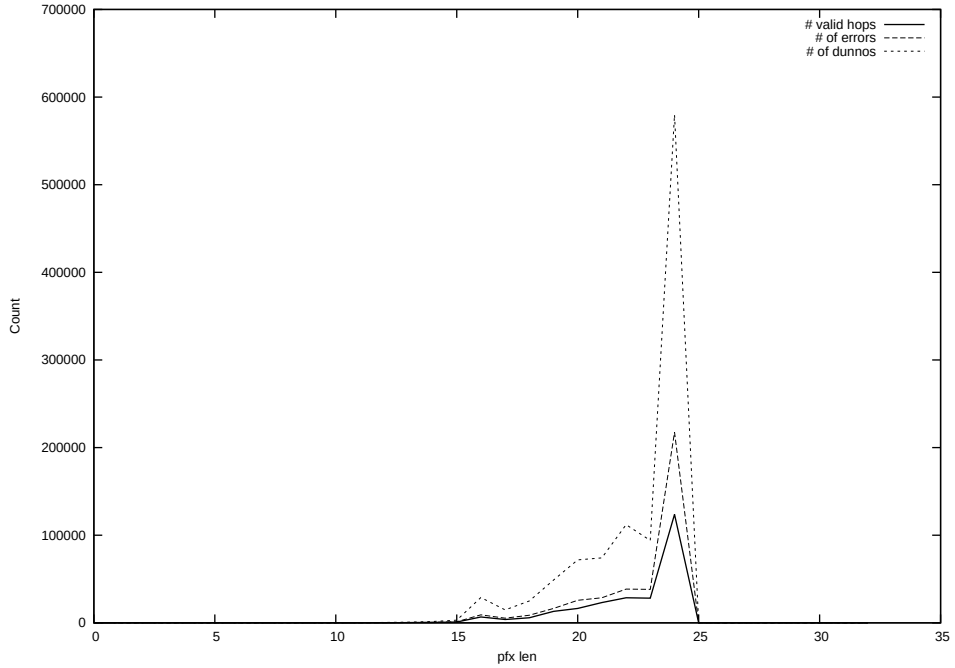
2013-04-13



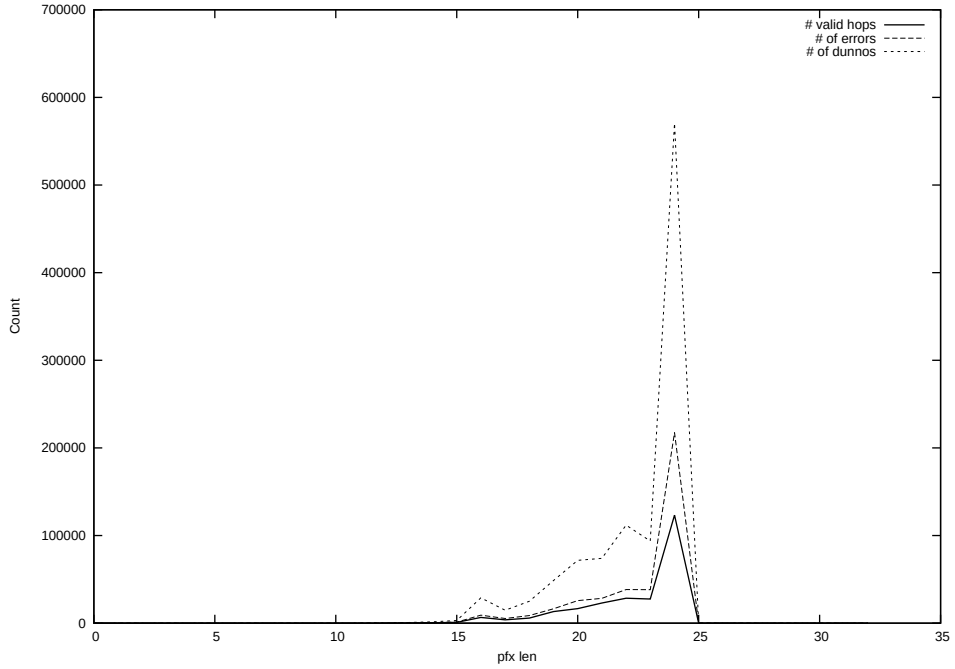
2013-04-14



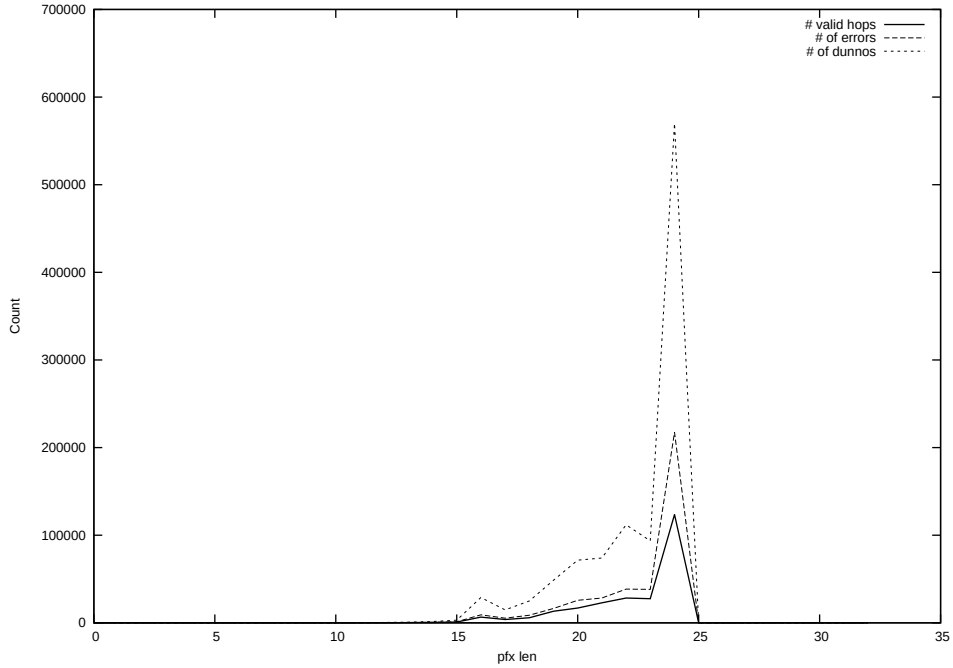
2013-04-15



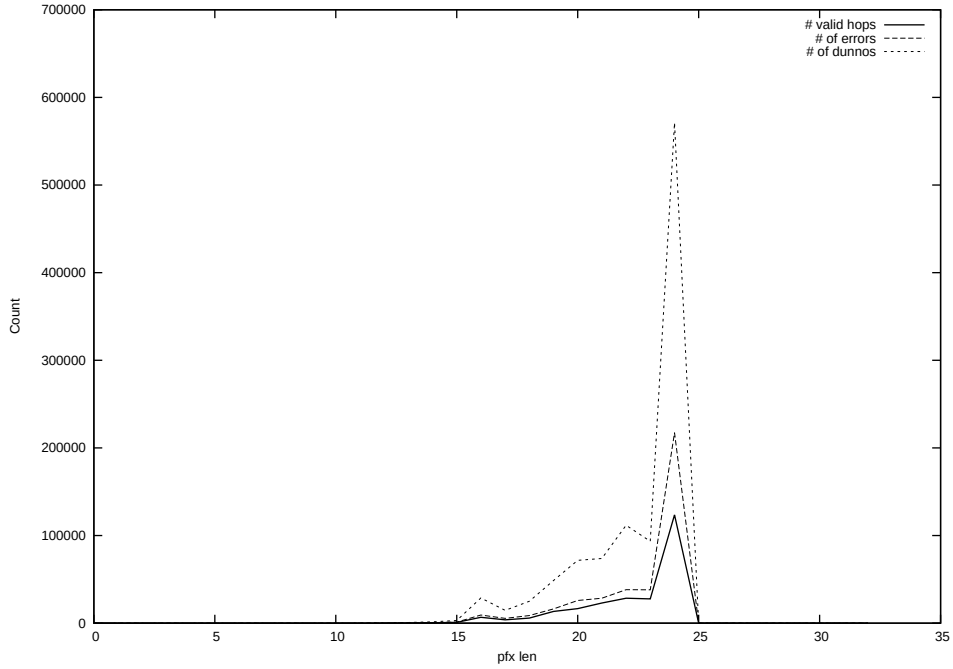
2013-04-16



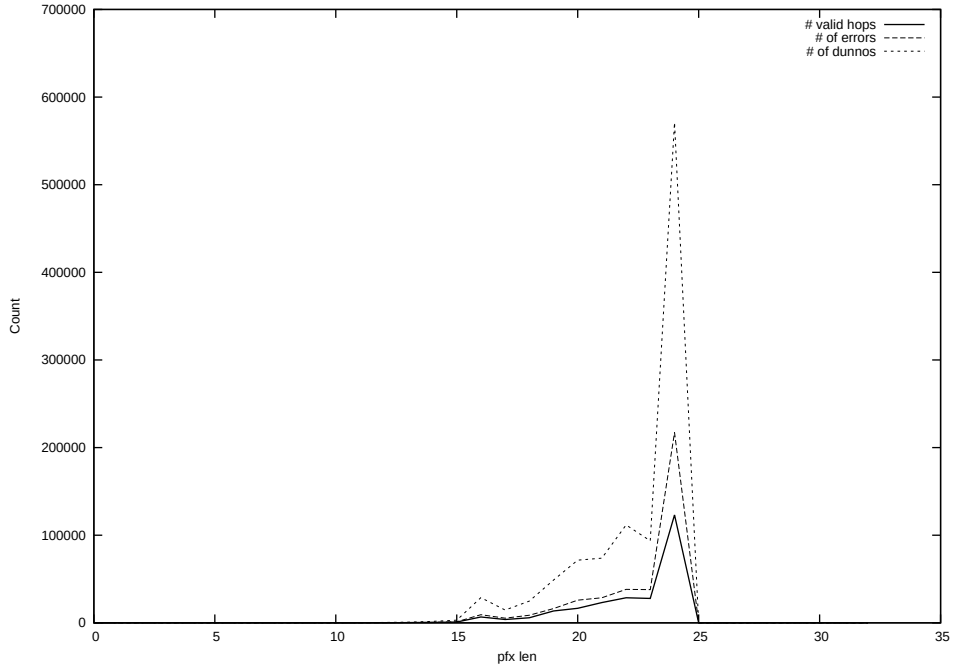
2013-04-17



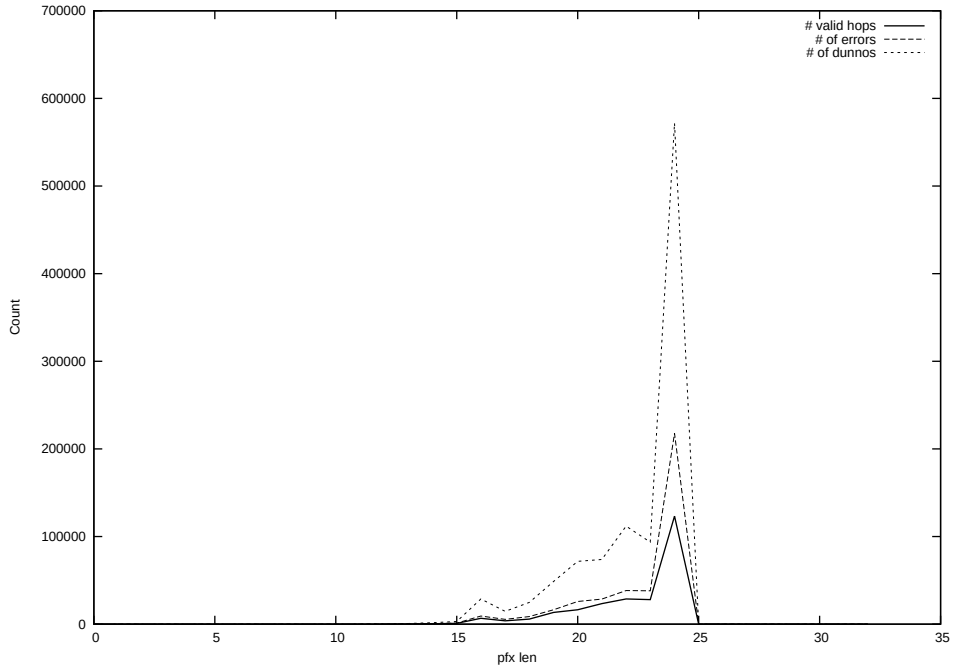
2013-04-18



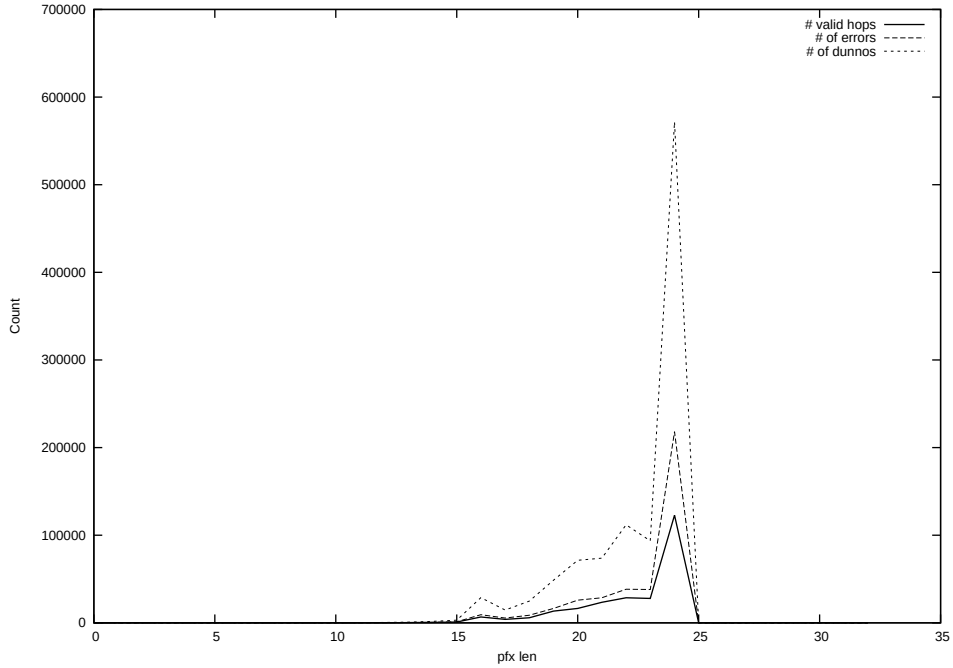
2013-04-19



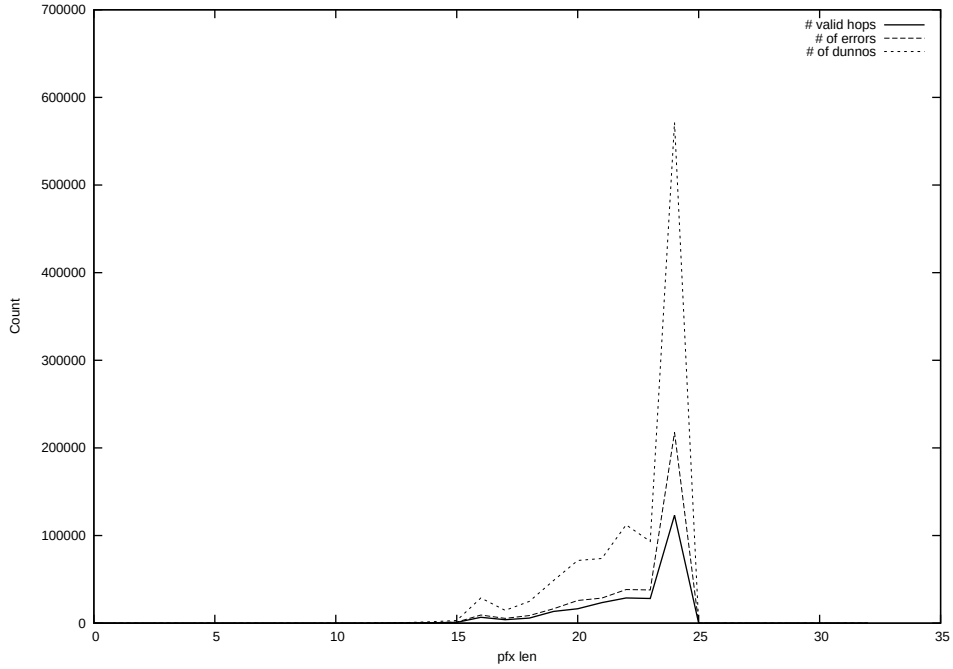
2013-04-20



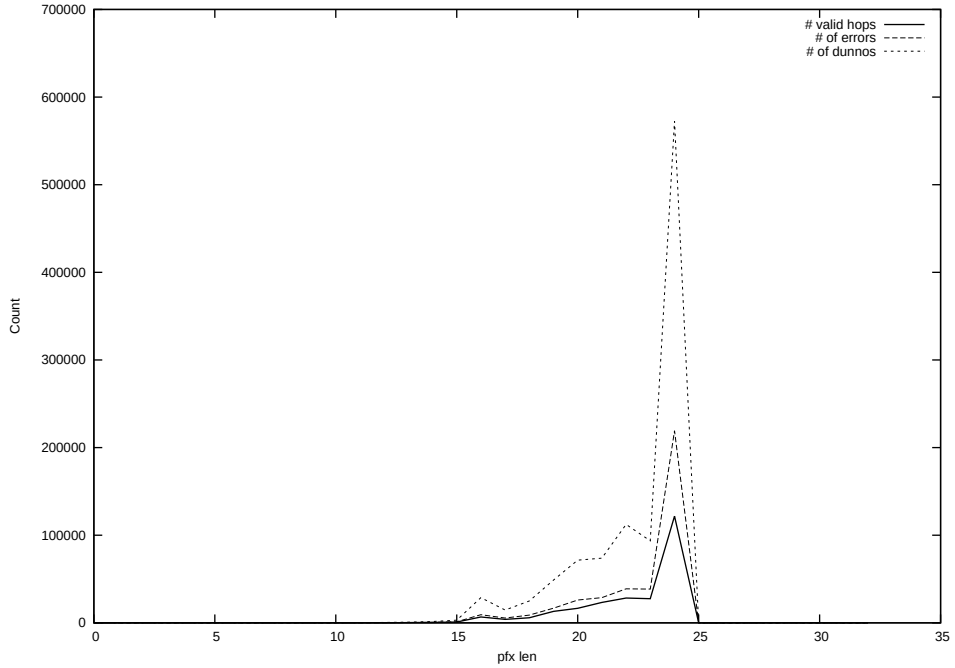
2013-04-21



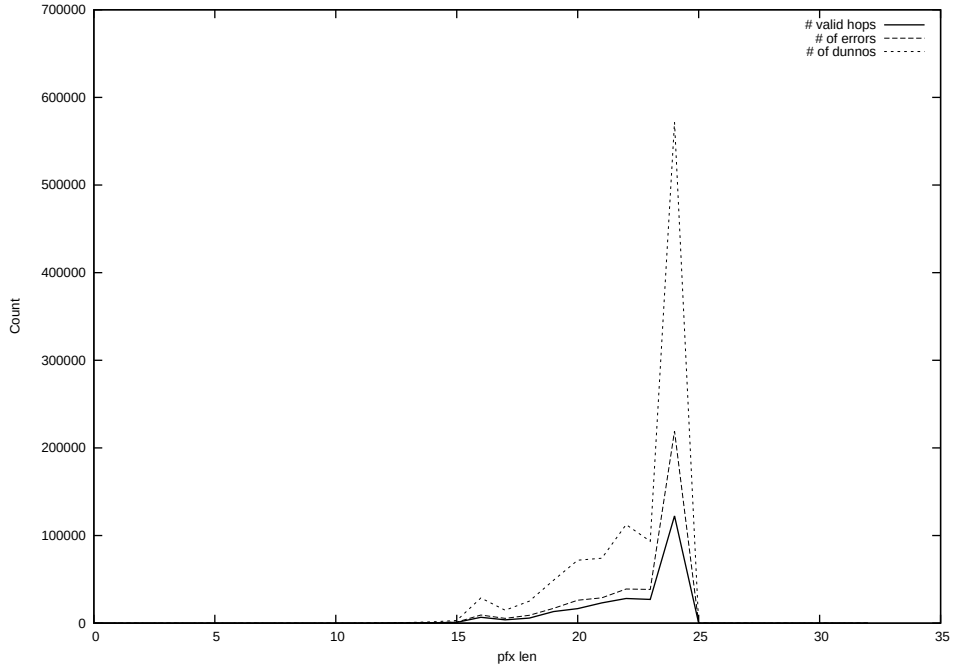
2013-04-22



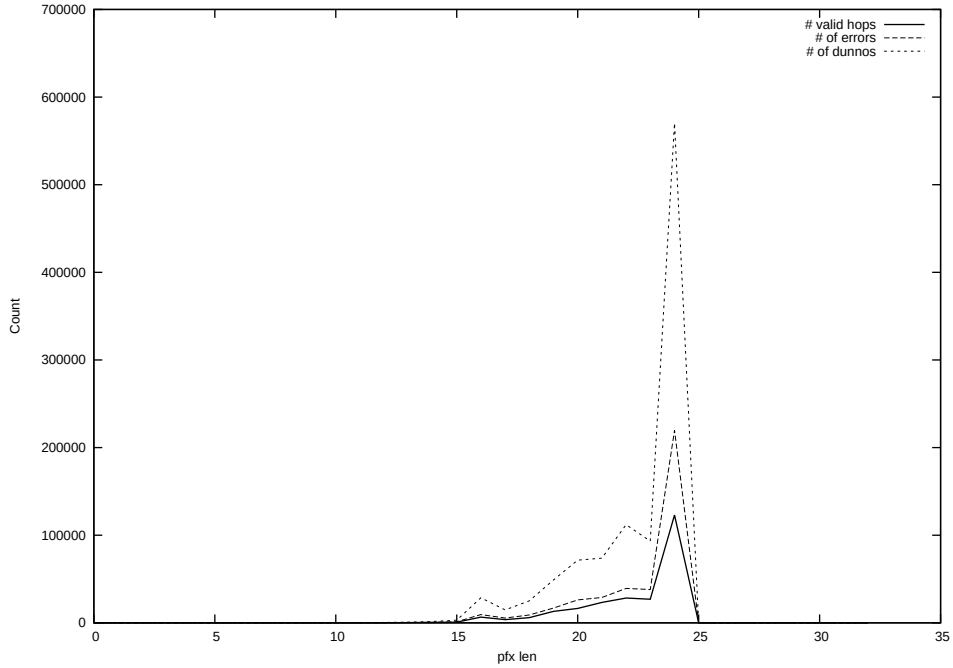
2013-04-23



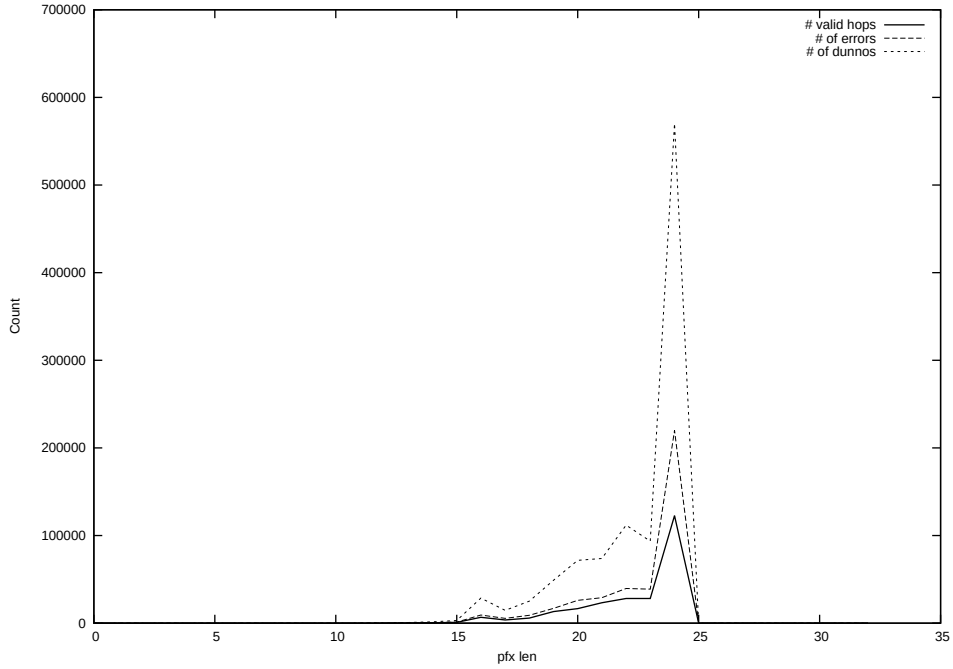
2013-04-24



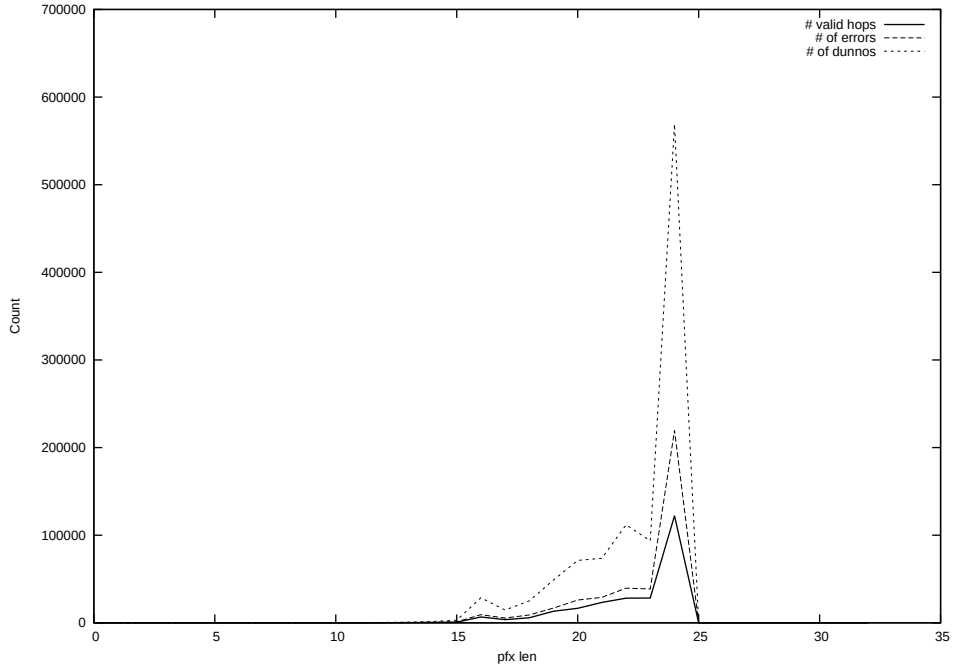
2013-04-25



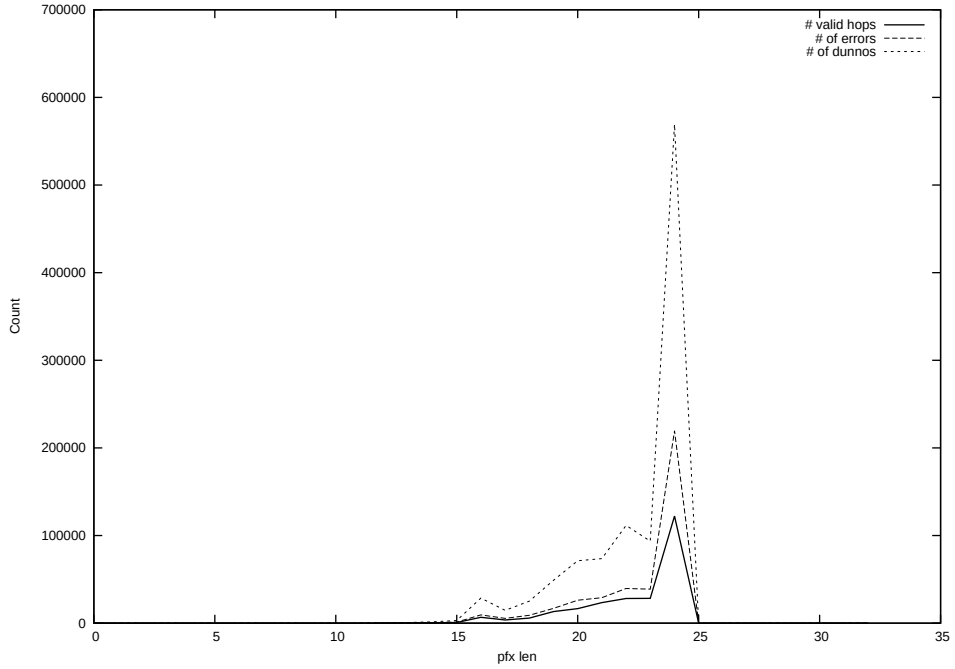
2013-04-26



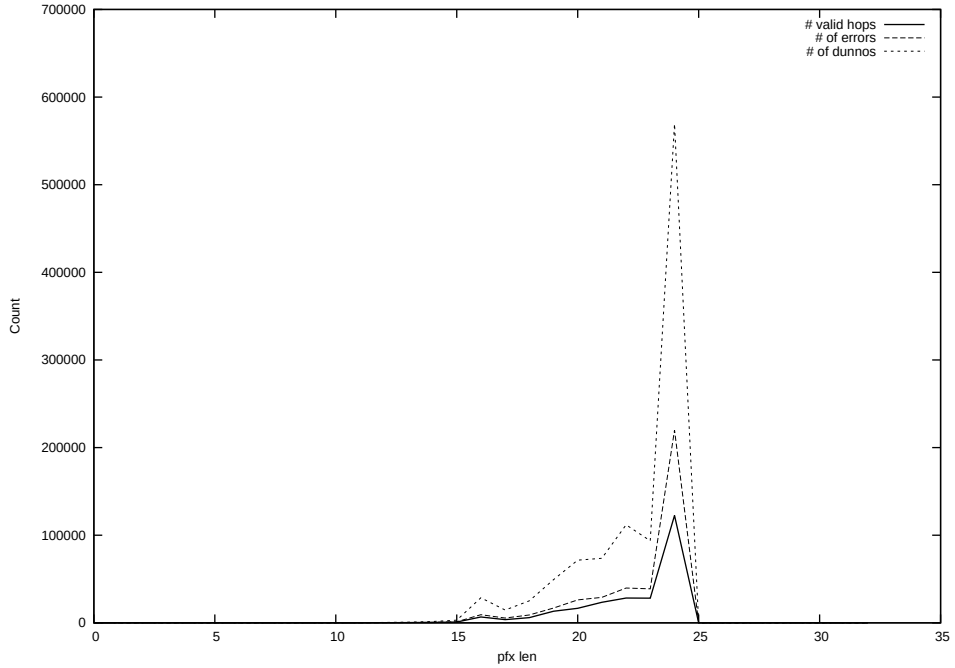
2013-04-27



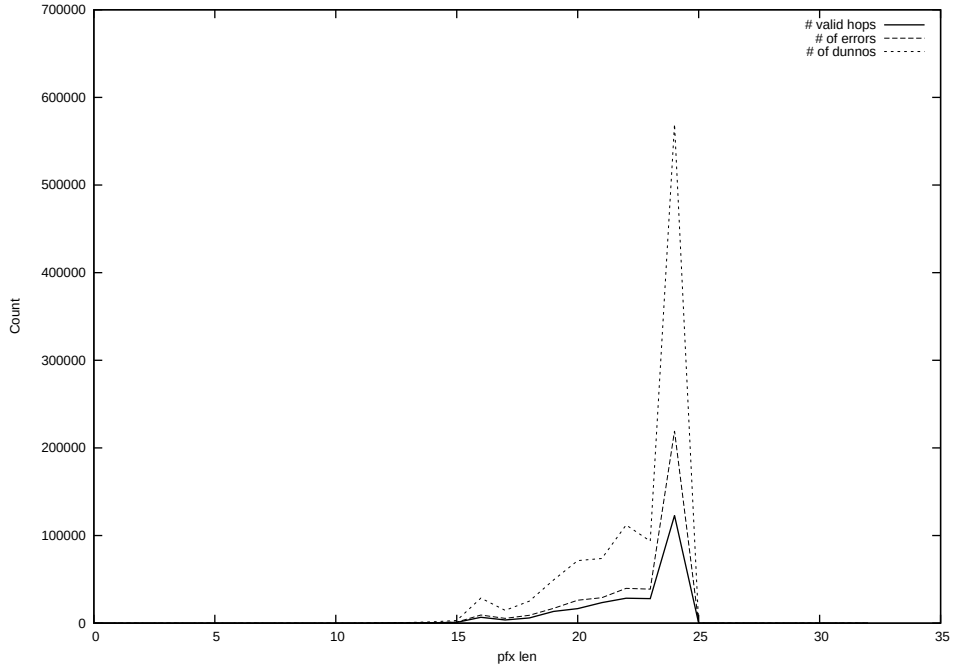
2013-04-28



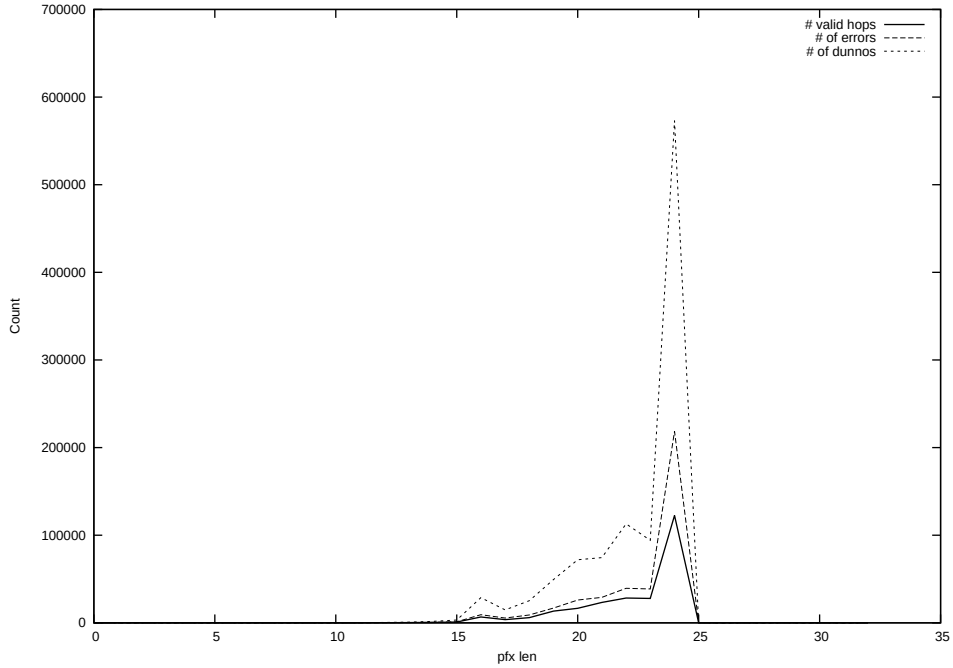
2013-04-29



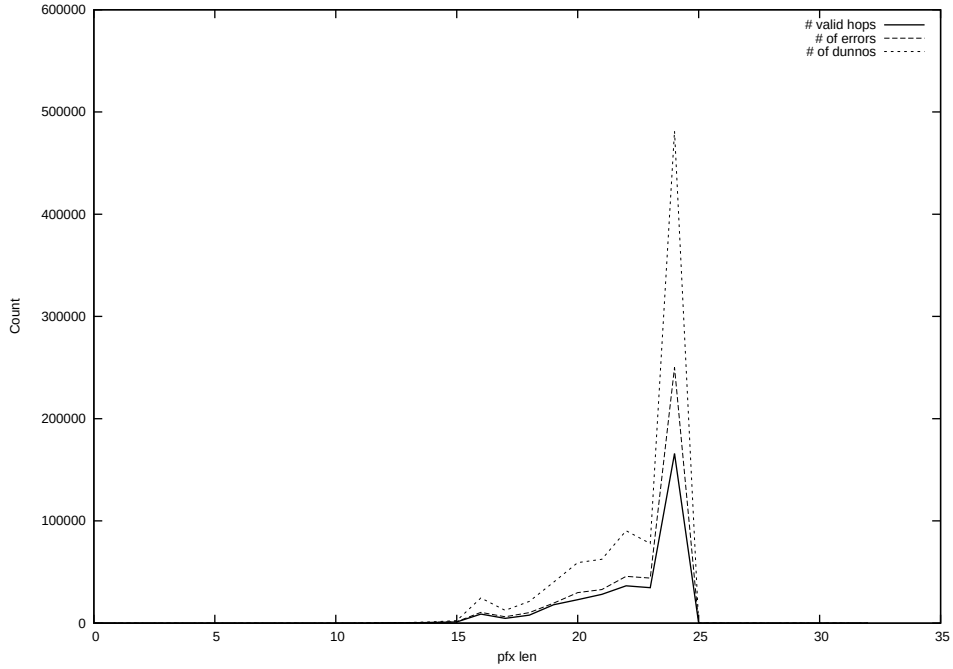
2013-04-30



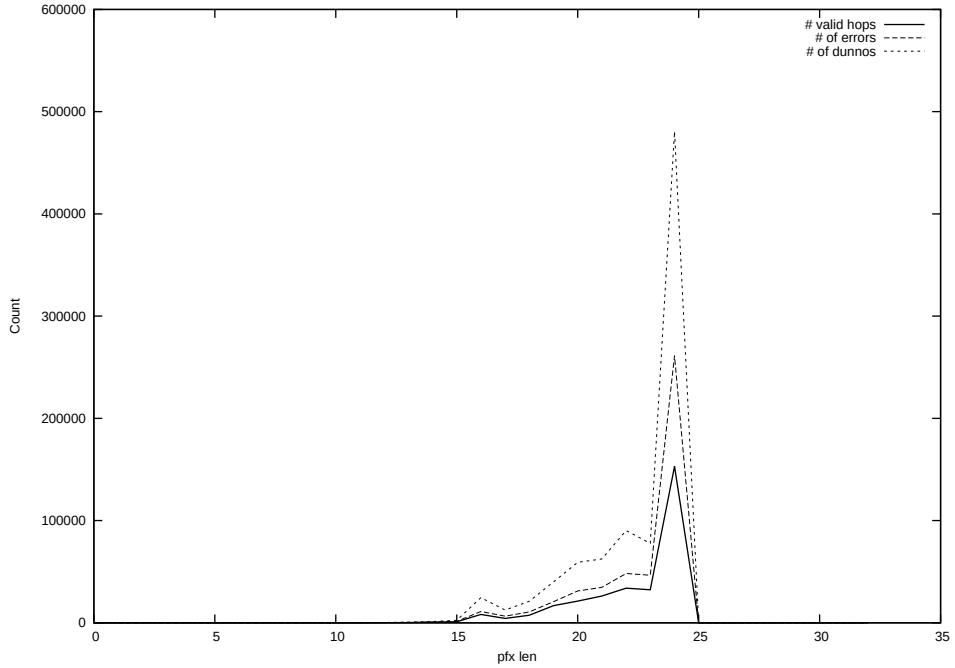
2013-05-01



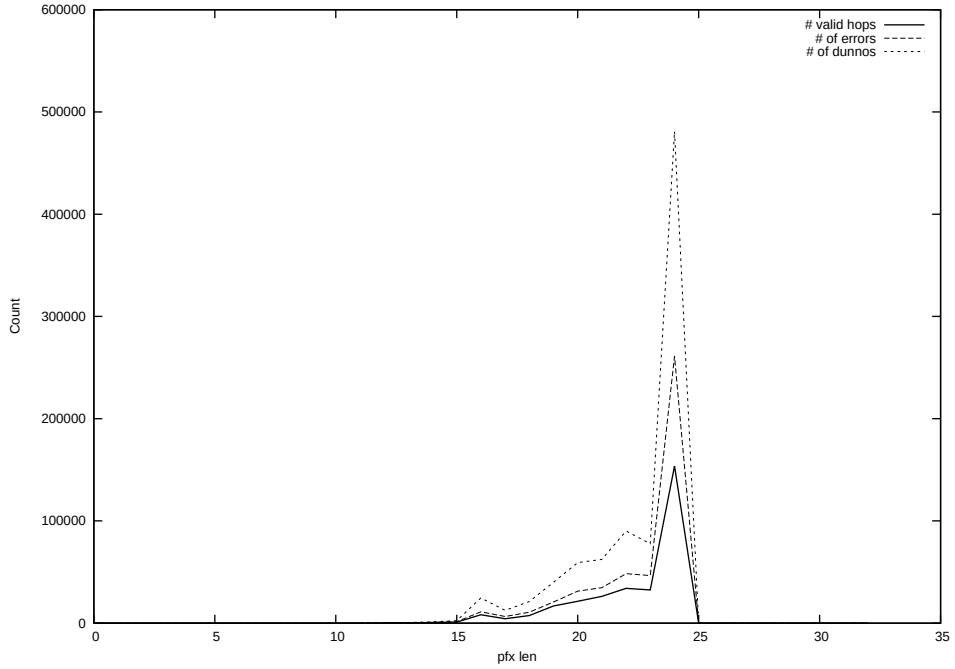
2013-05-02



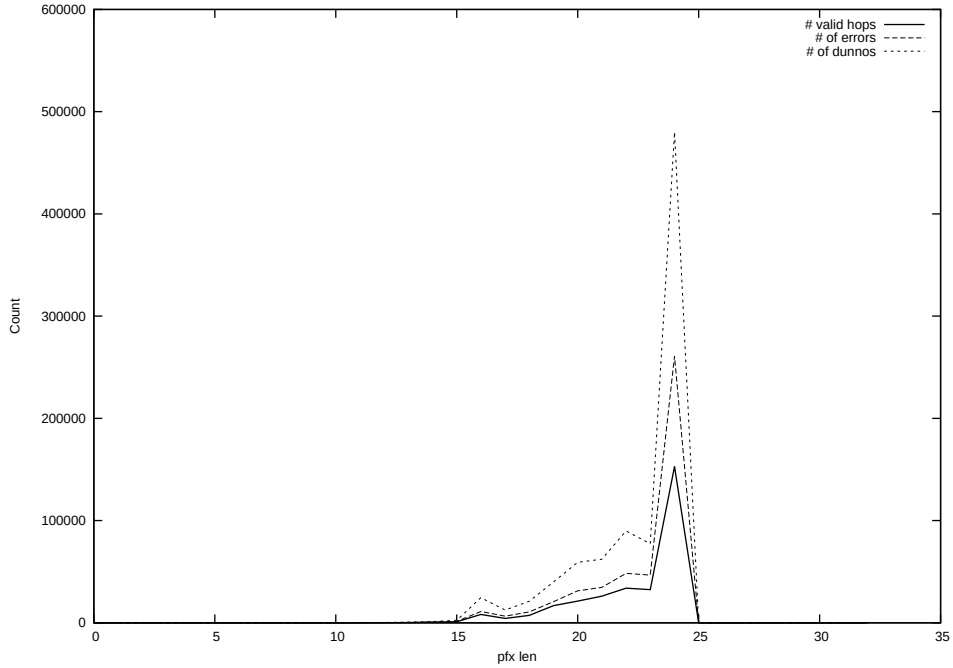
2013-05-03



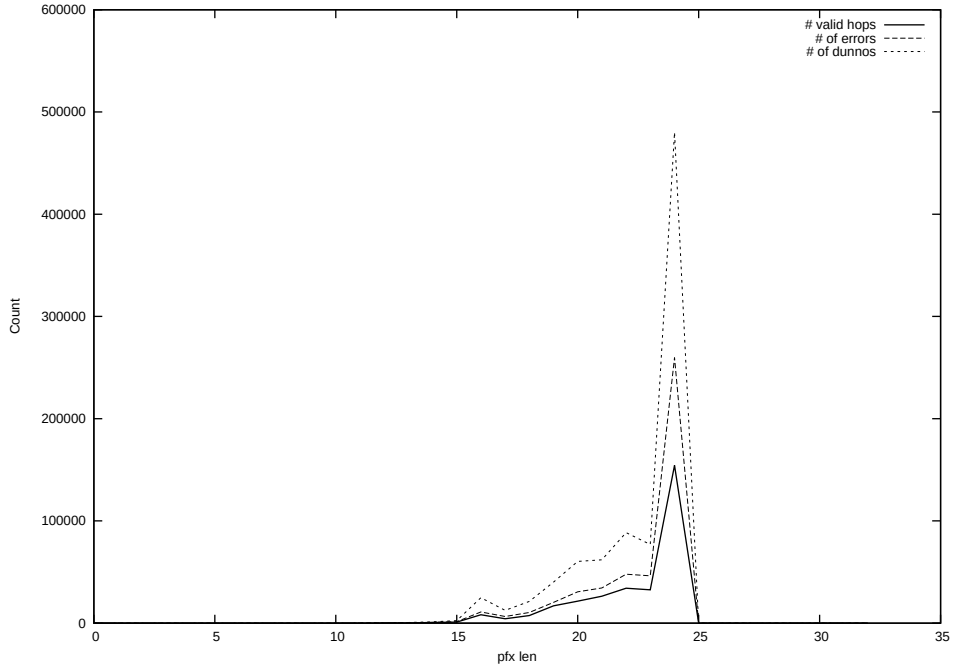
2013-05-04



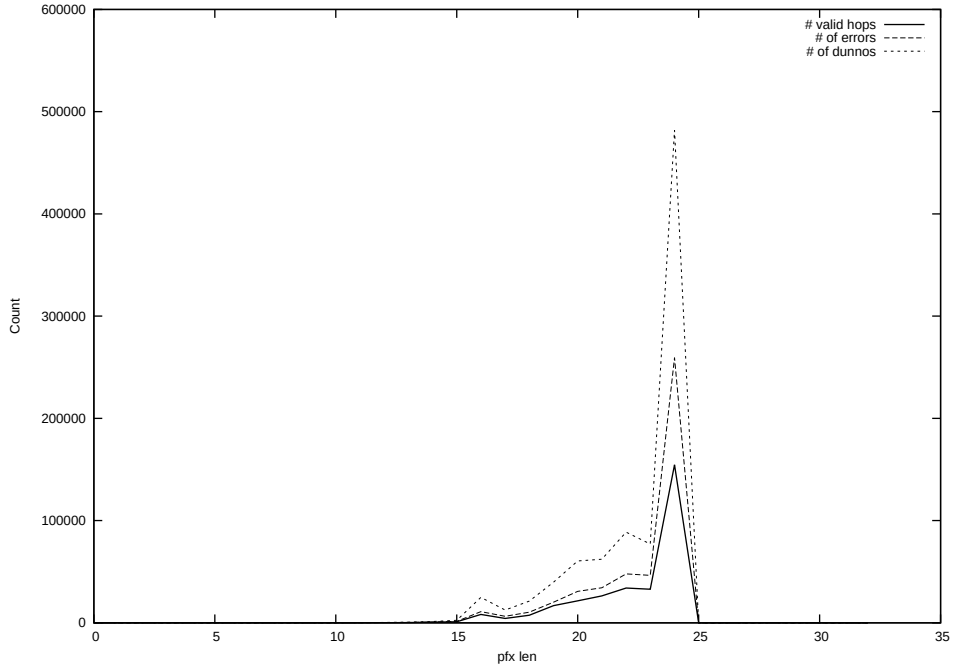
2013-05-05



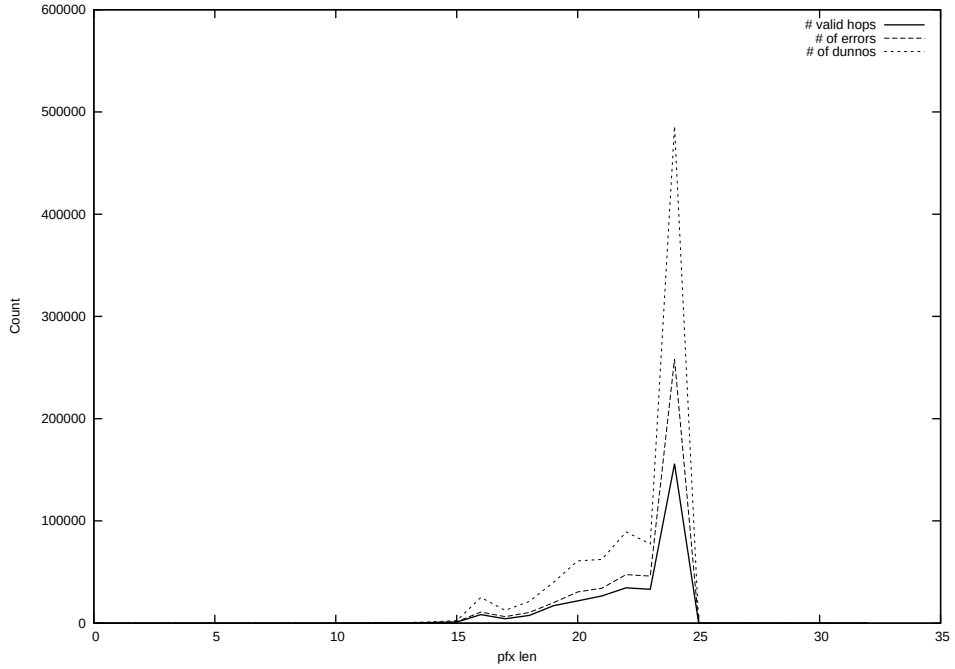
2013-05-06



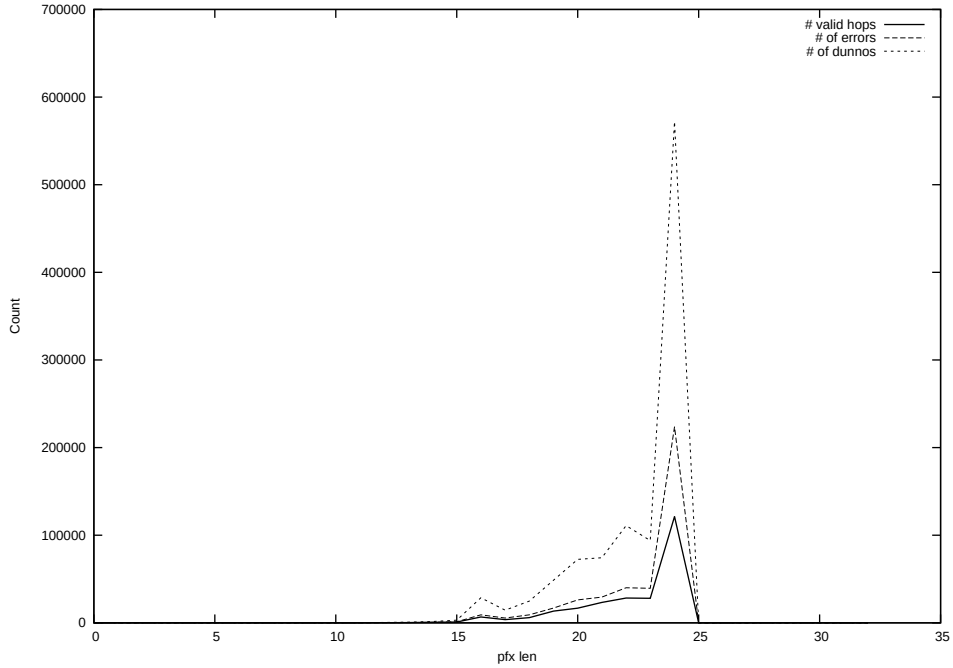
2013-05-07



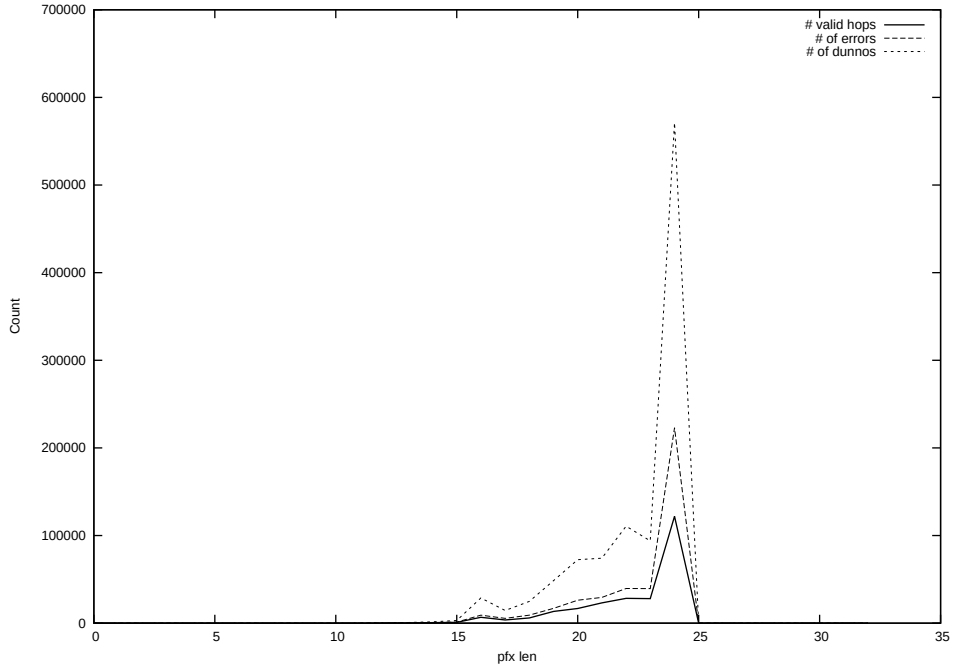
2013-05-08



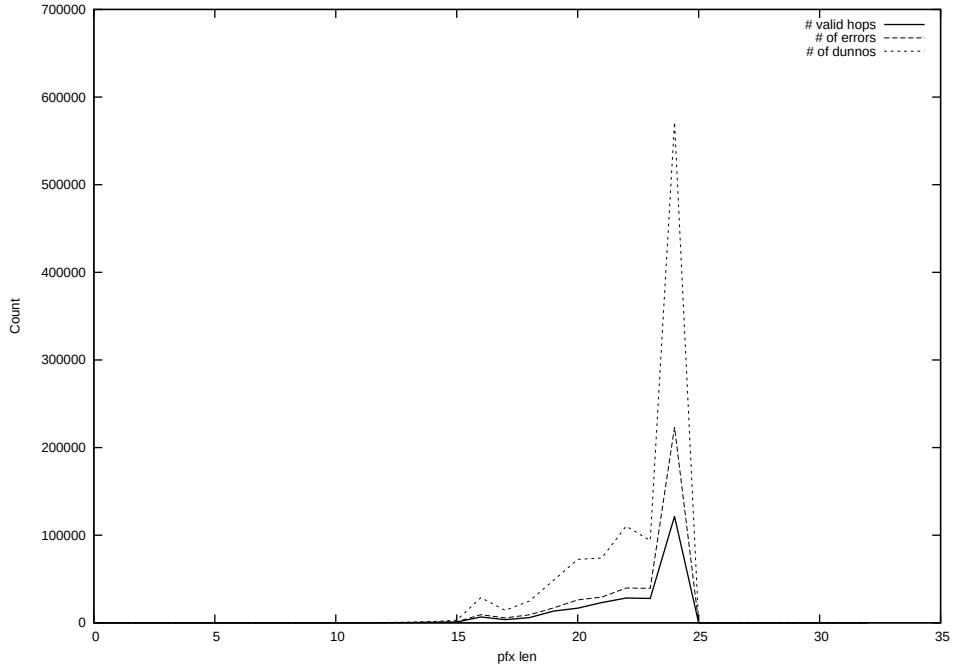
2013-05-09



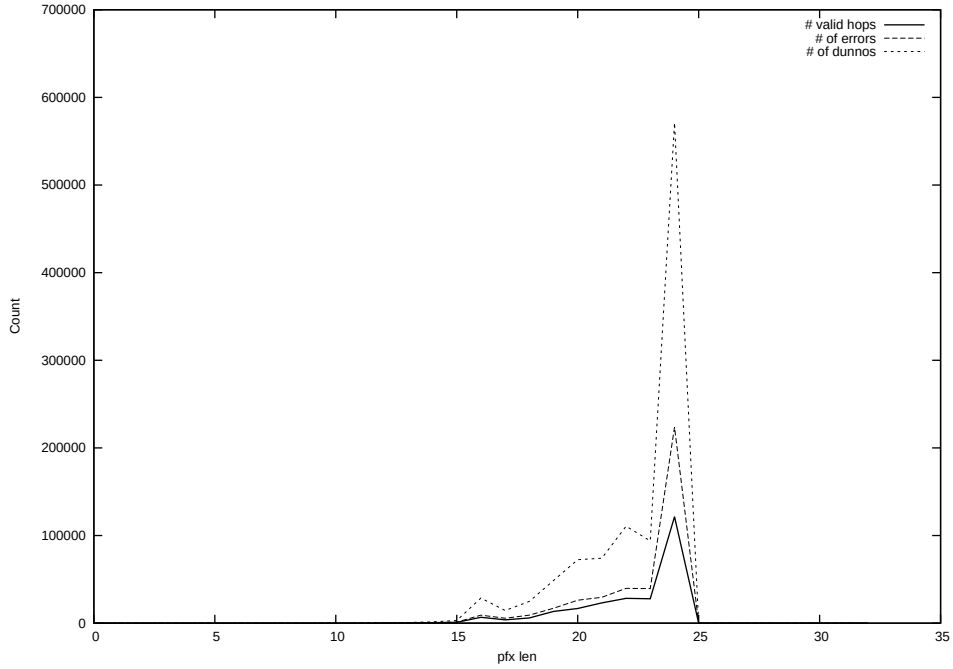
2013-05-10



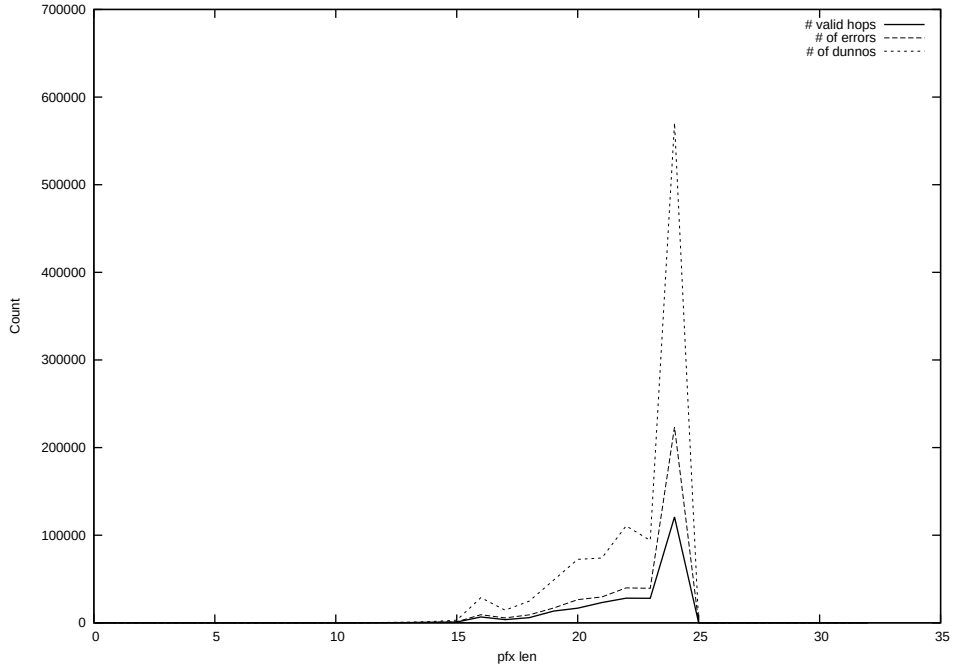
2013-05-11



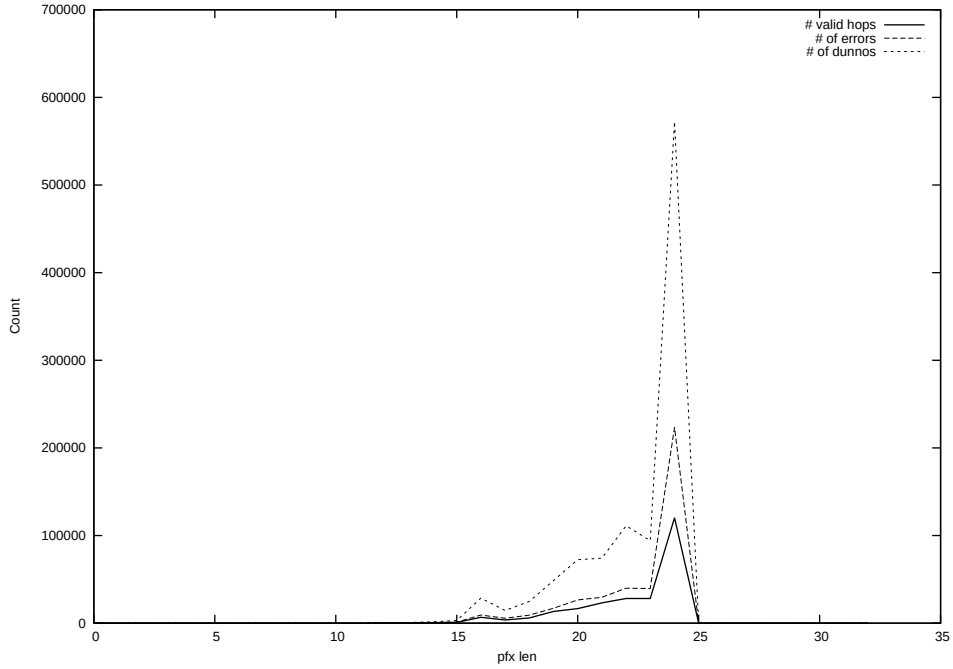
2013-05-12



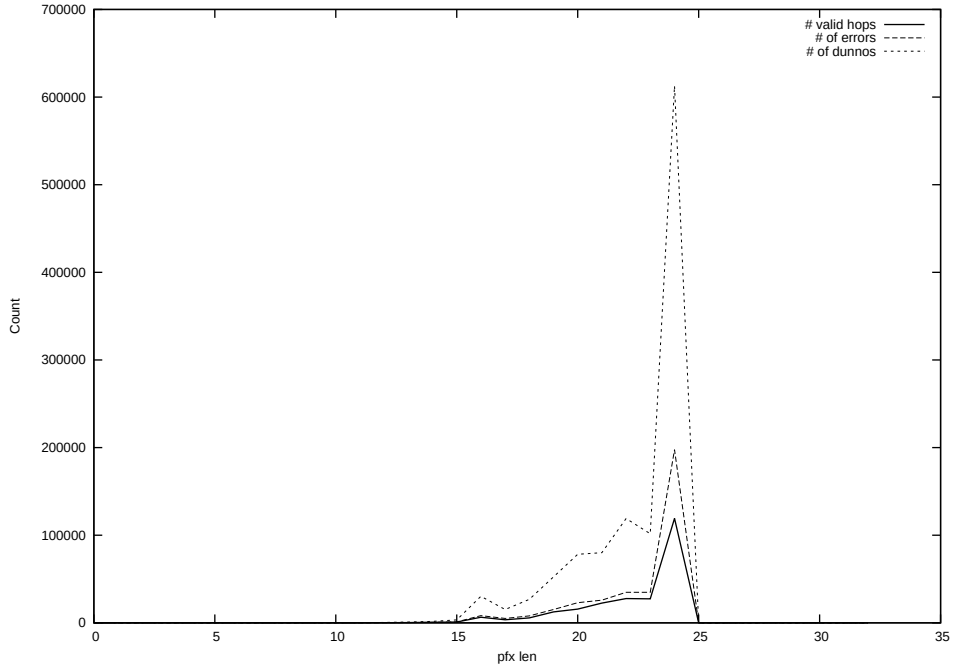
2013-05-13



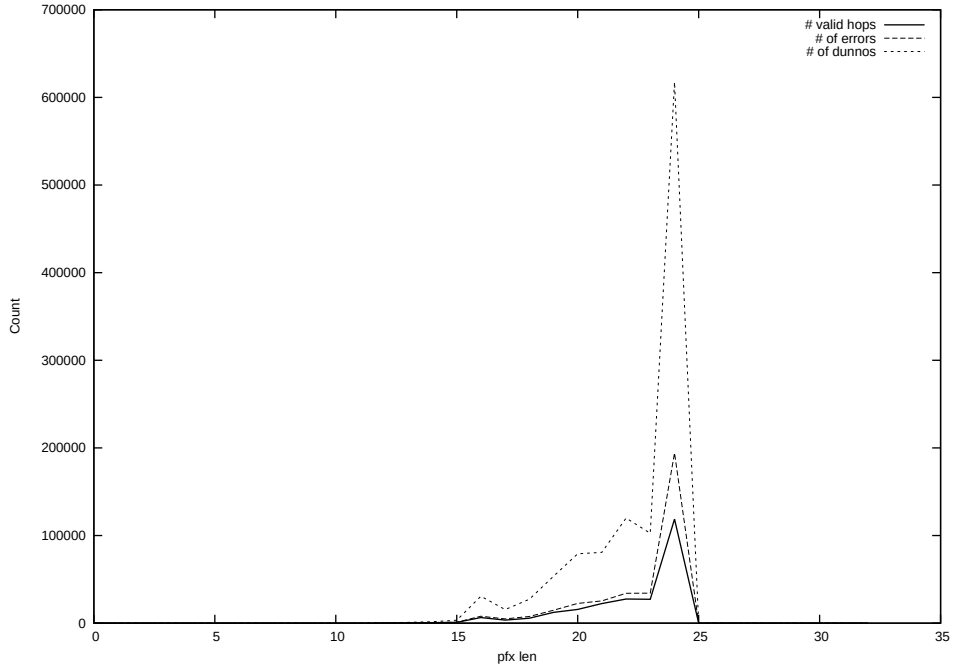
2013-05-14



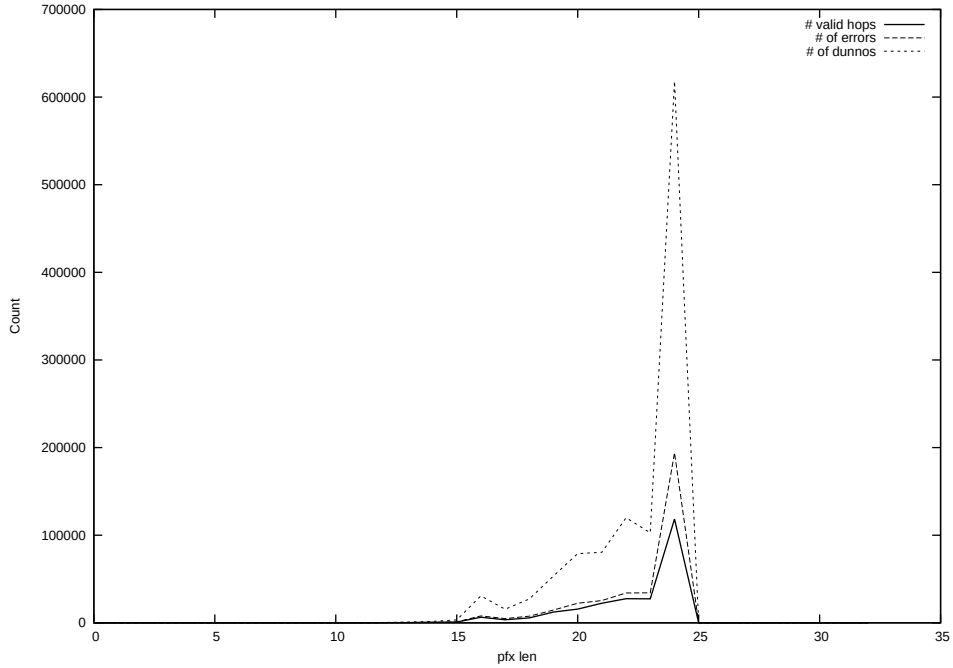
2013-05-15



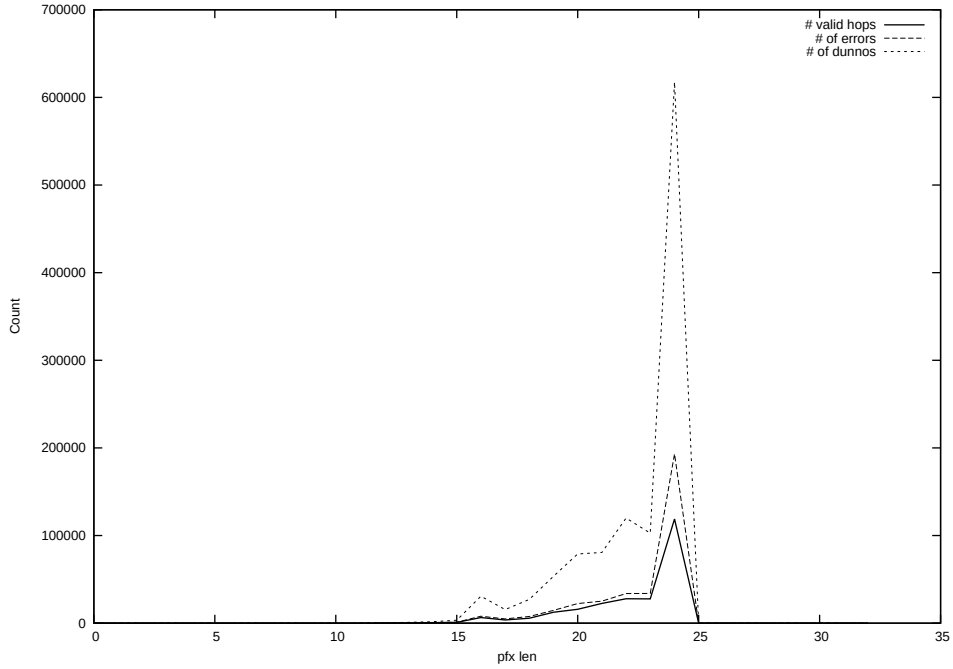
2013-05-16



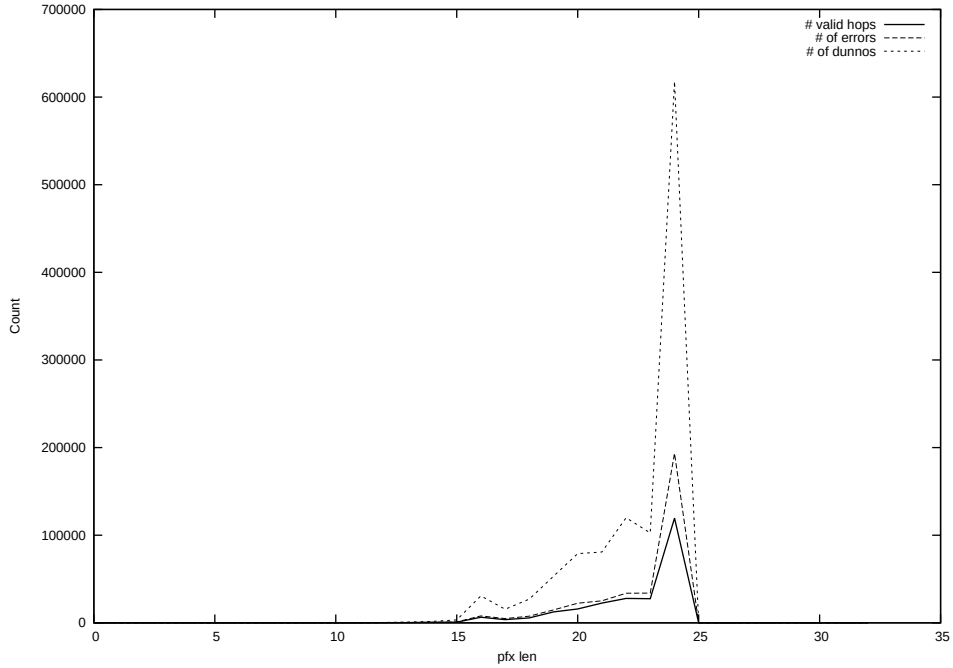
2013-05-17



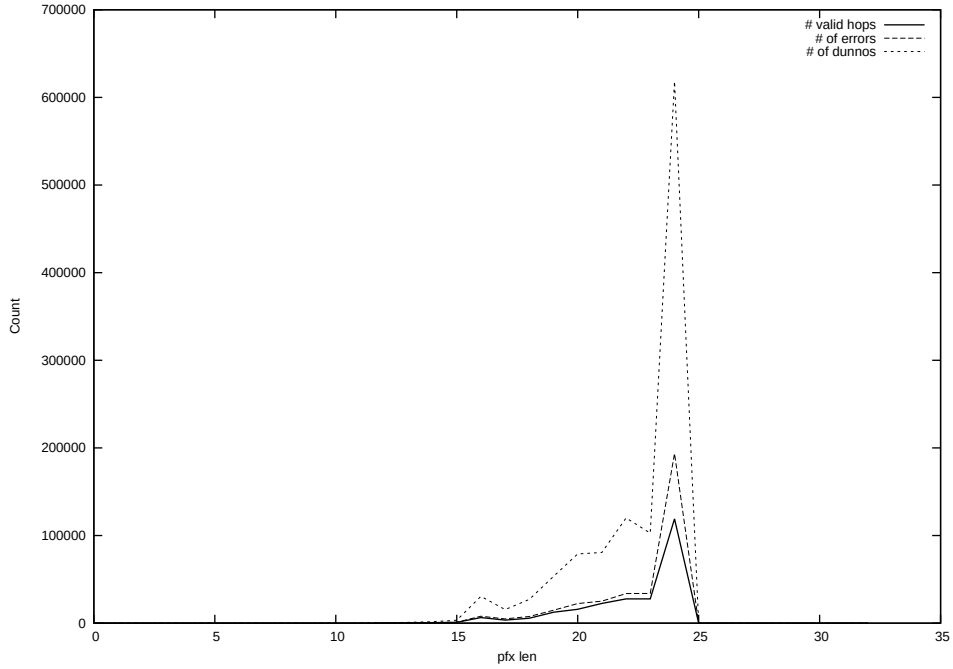
2013-05-18



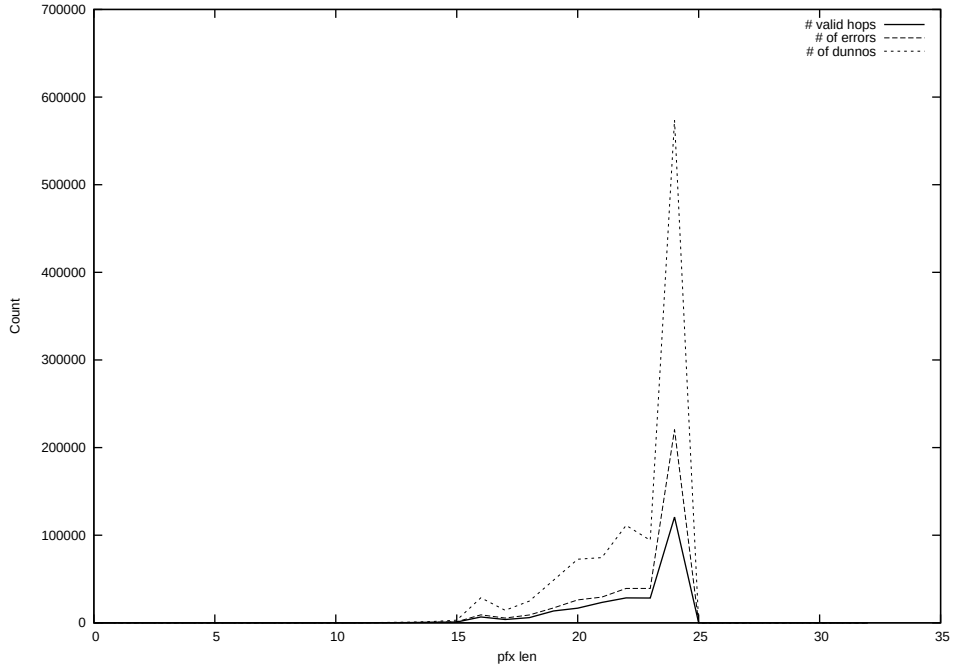
2013-05-19



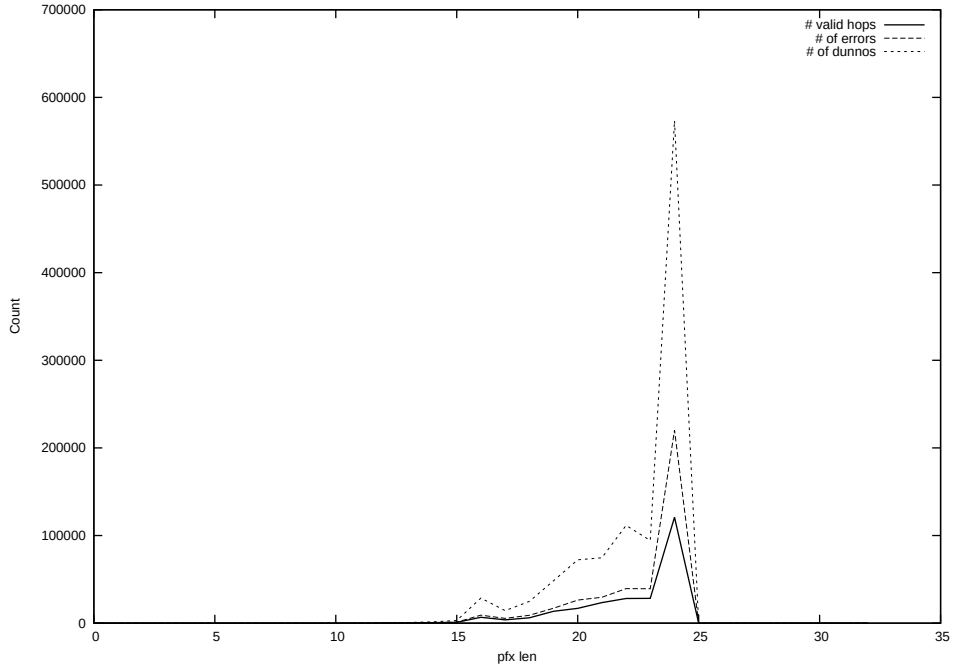
2013-05-20



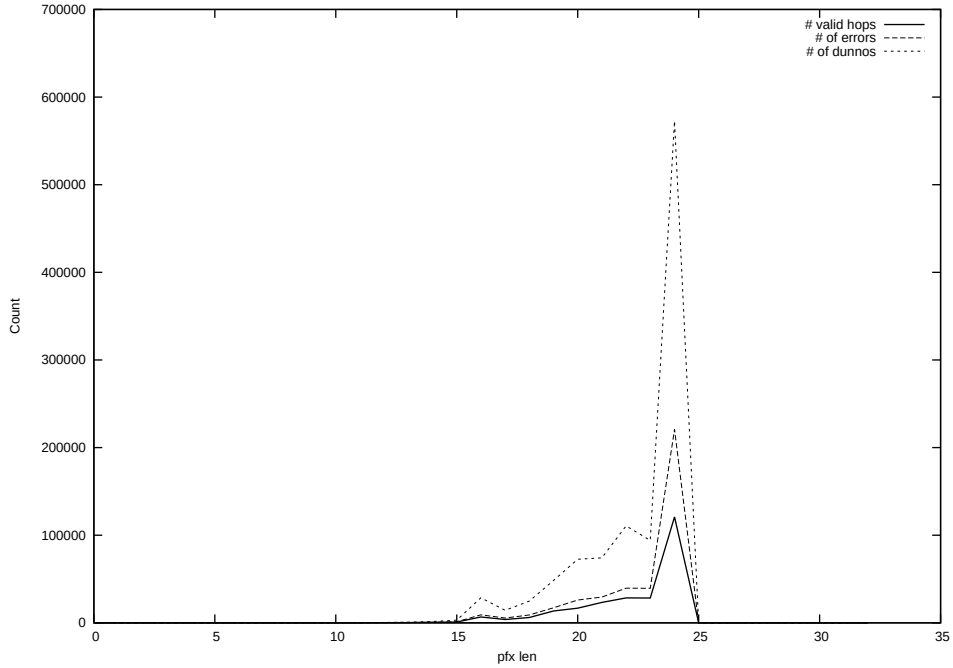
2013-05-21



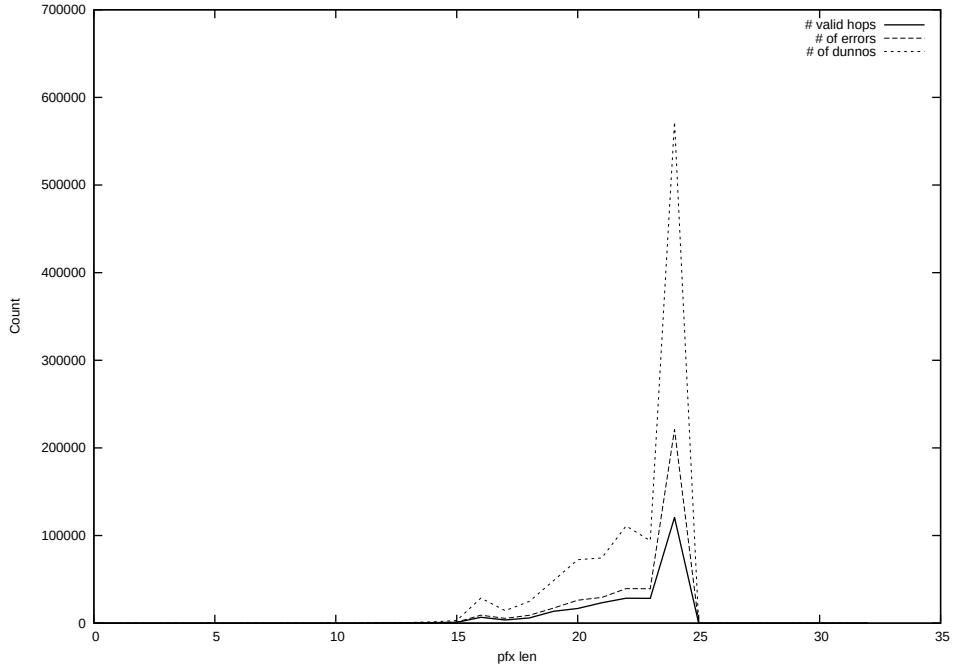
2013-05-22



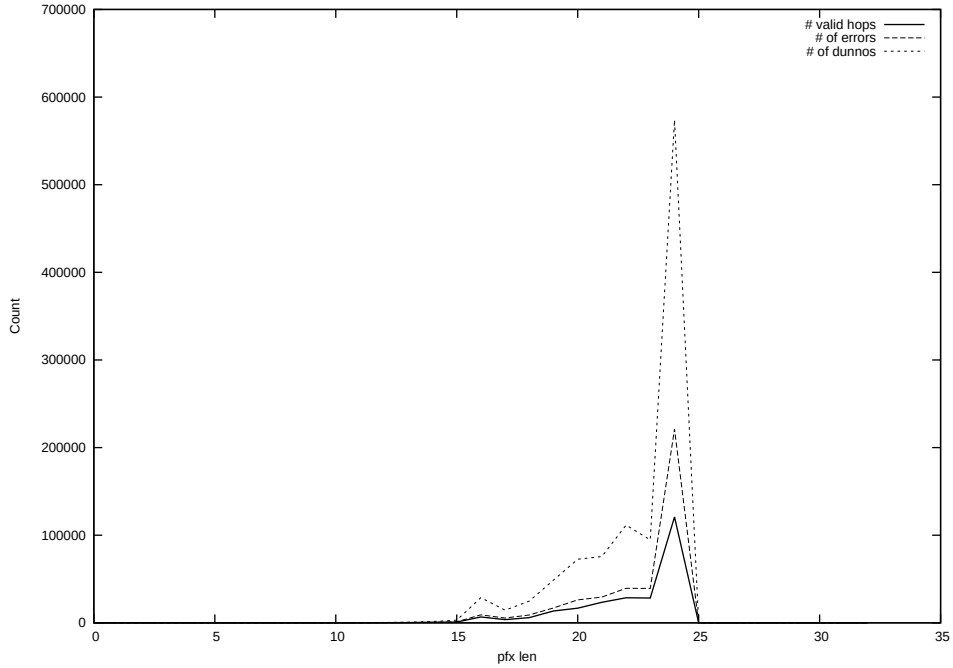
2013-05-23



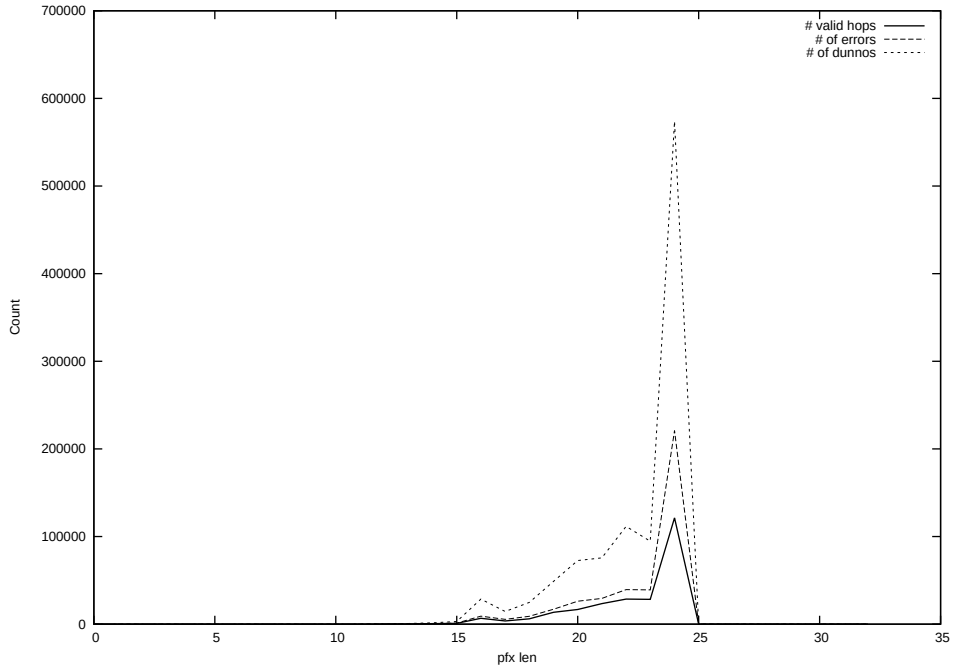
2013-05-24



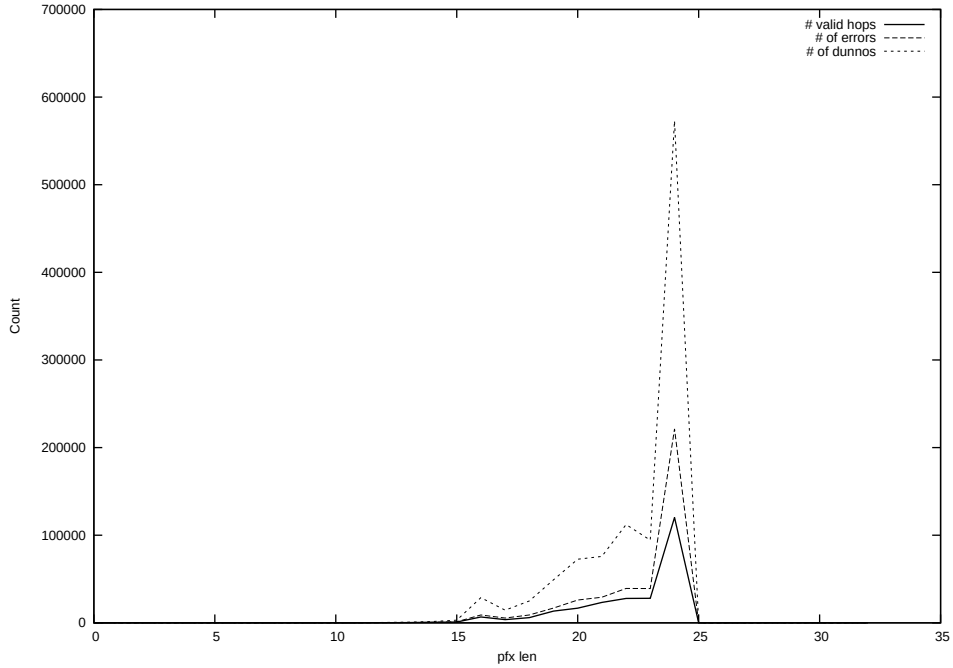
2013-05-25



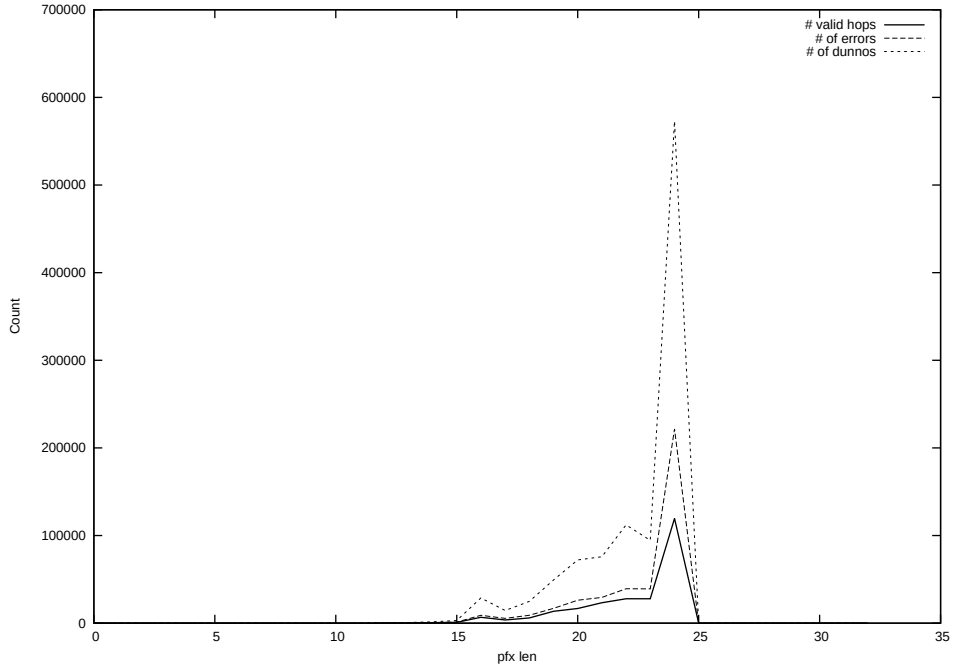
2013-05-26



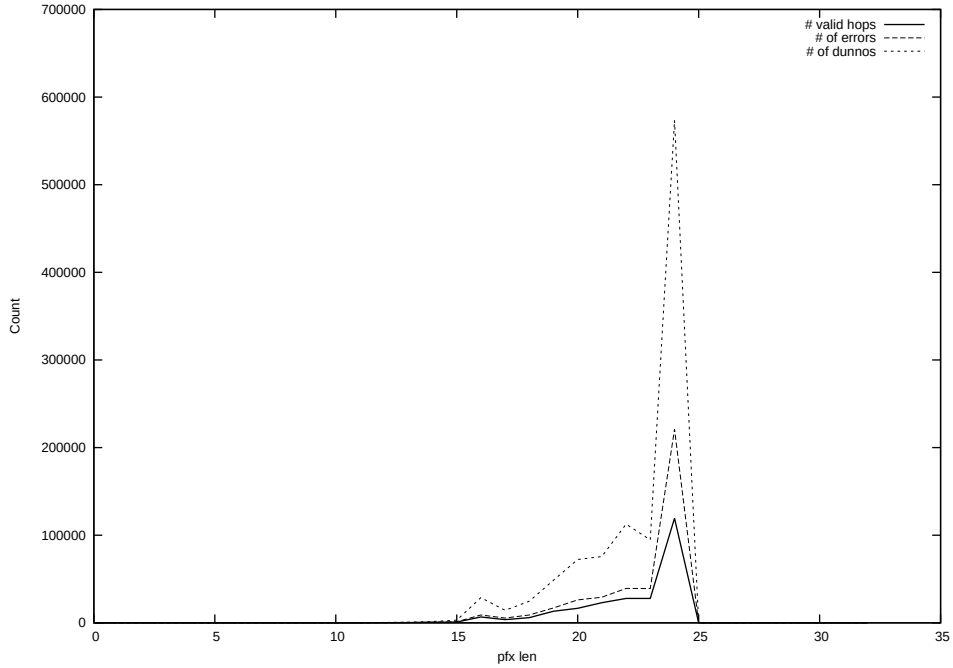
2013-05-27



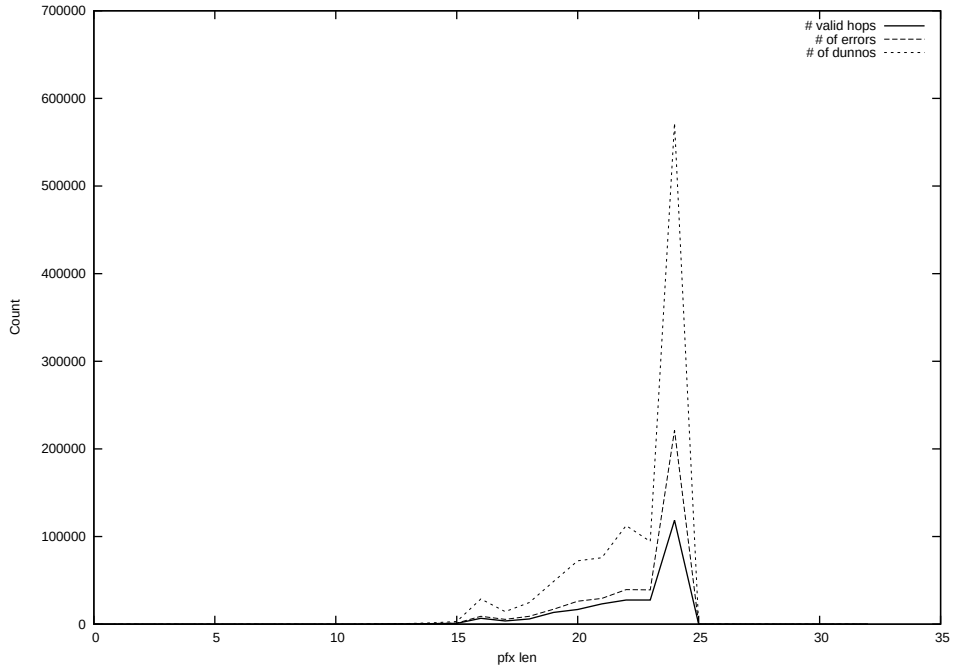
2013-05-28



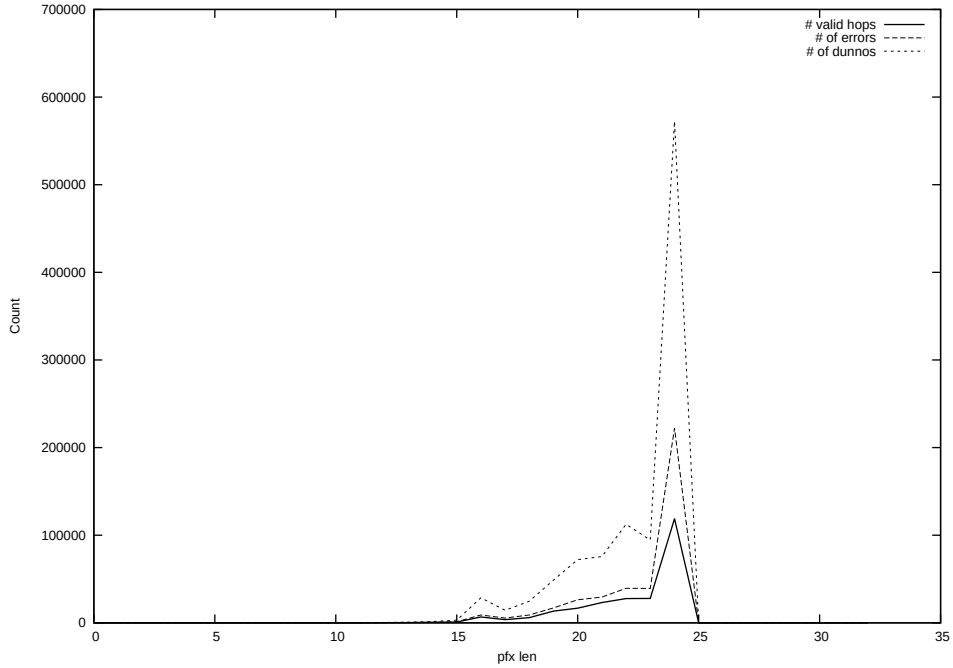
2013-05-29



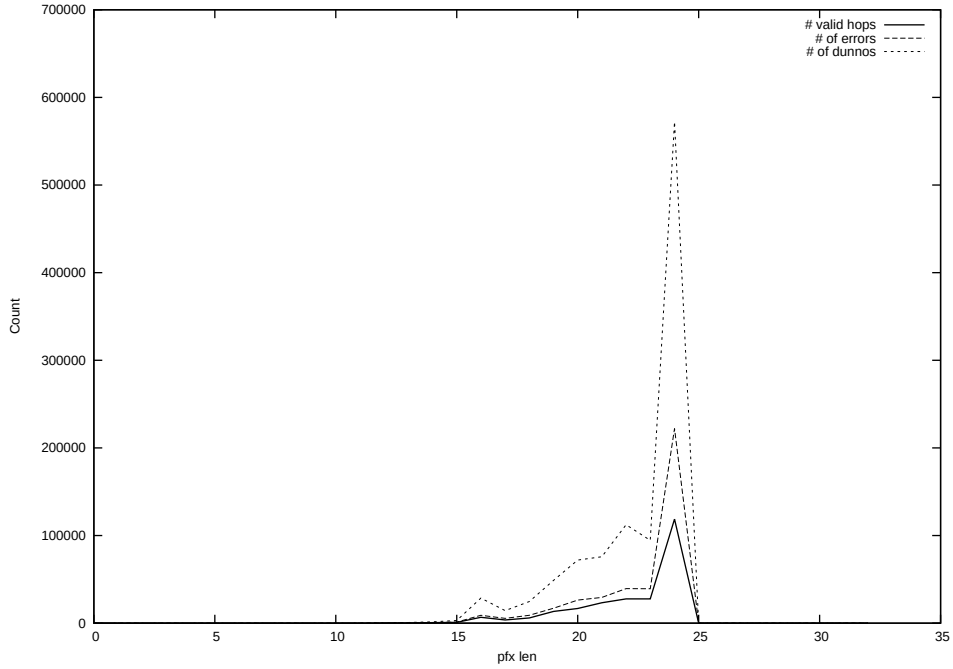
2013-05-30



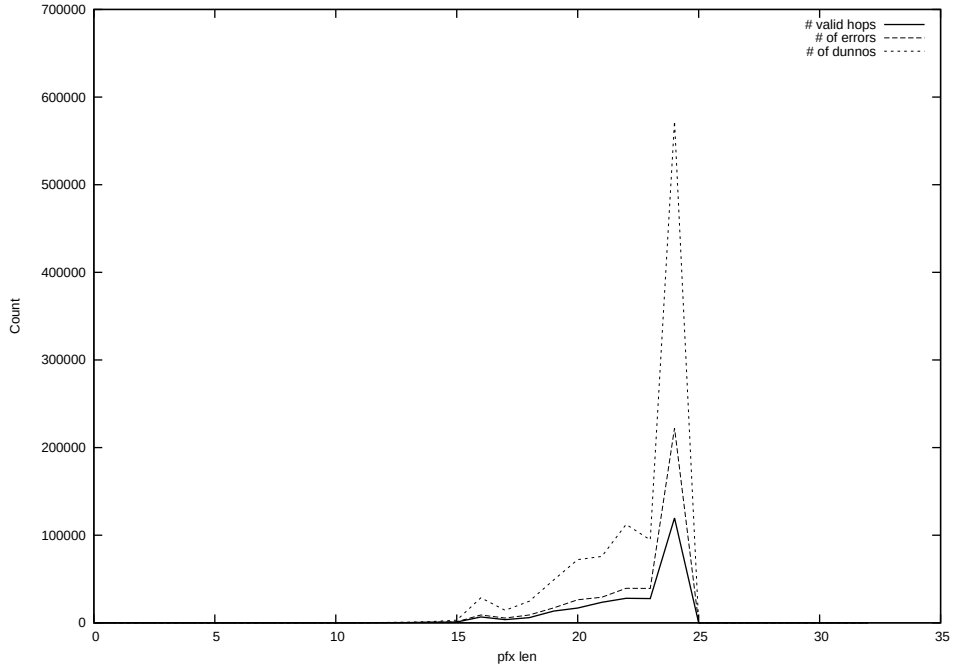
2013-05-31



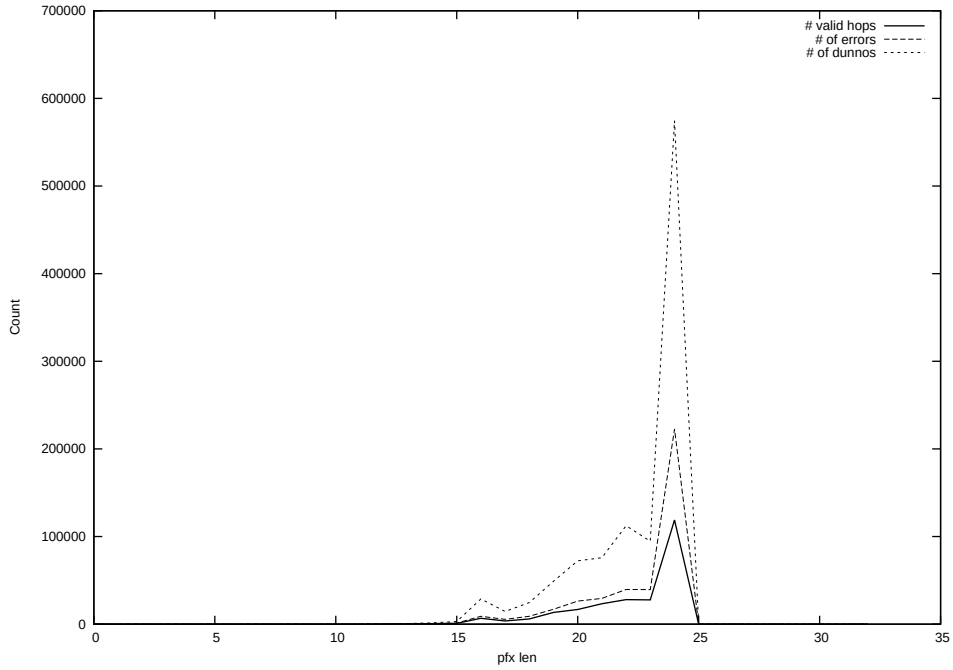
2013-06-01



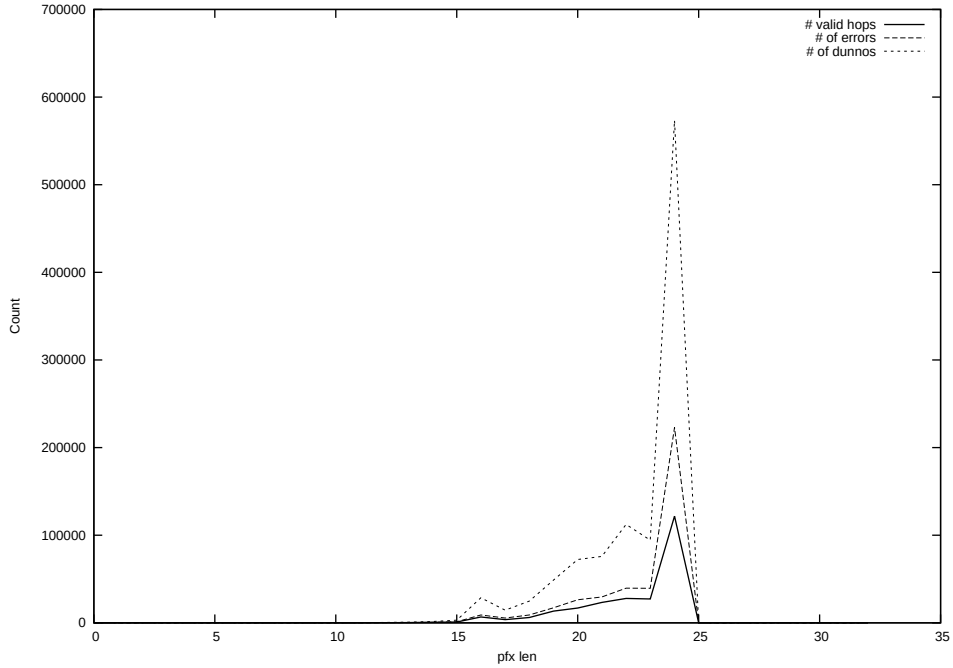
2013-06-02



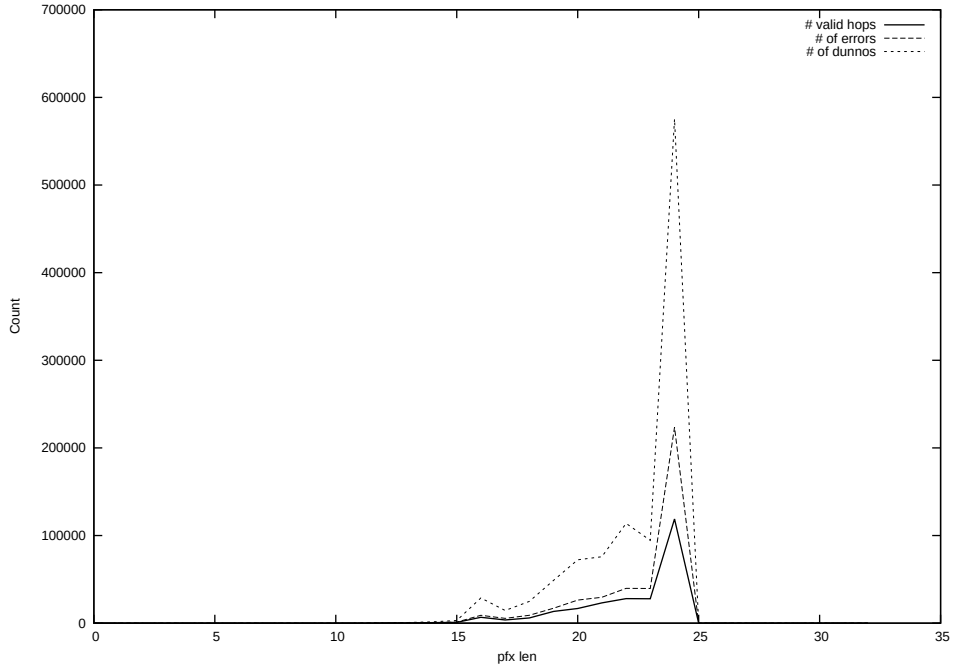
2013-06-03



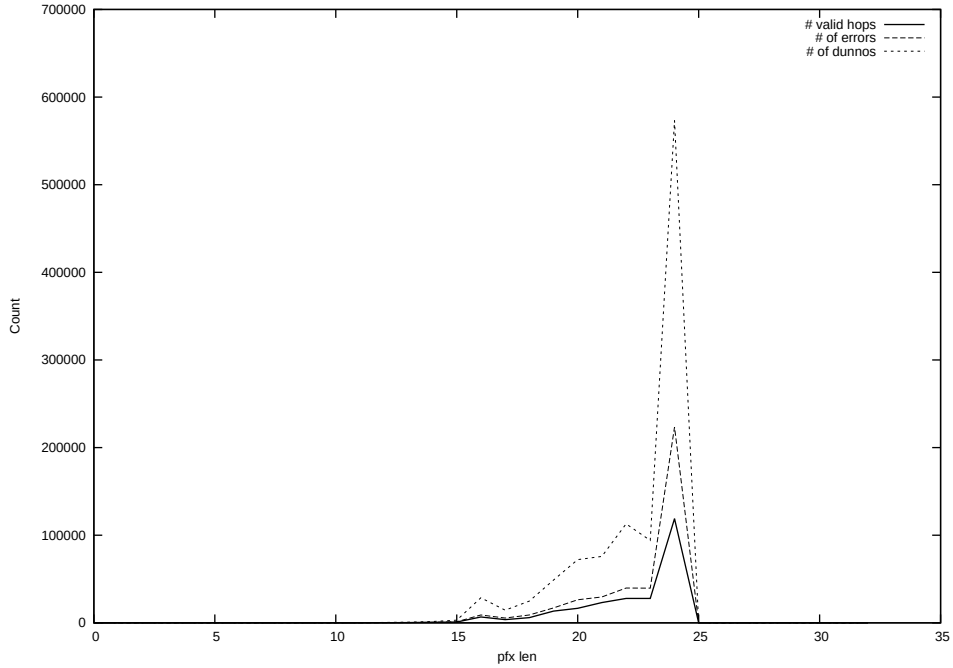
2013-06-04



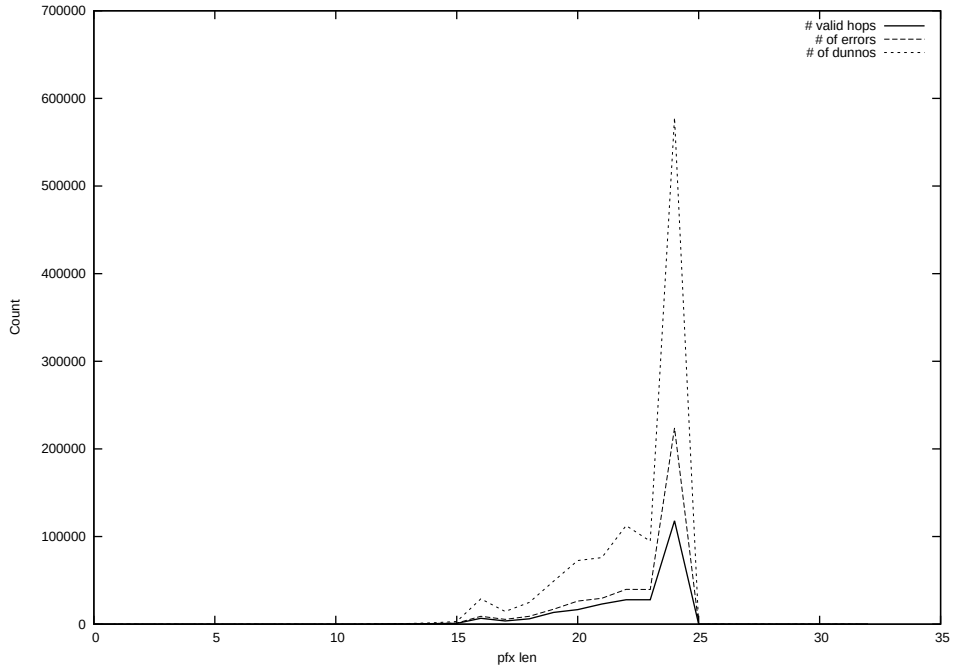
2013-06-05



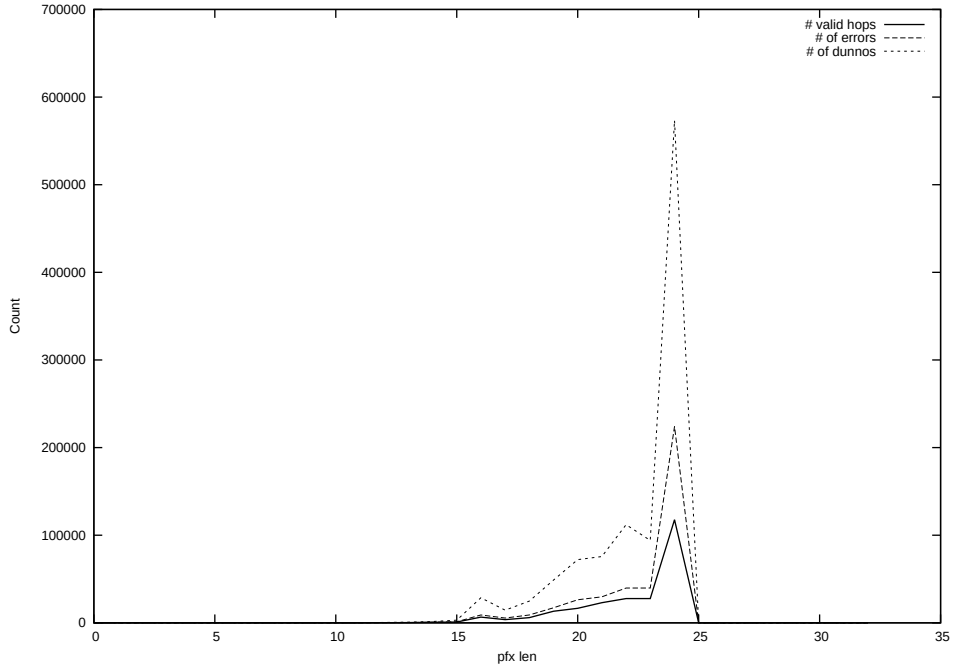
2013-06-06



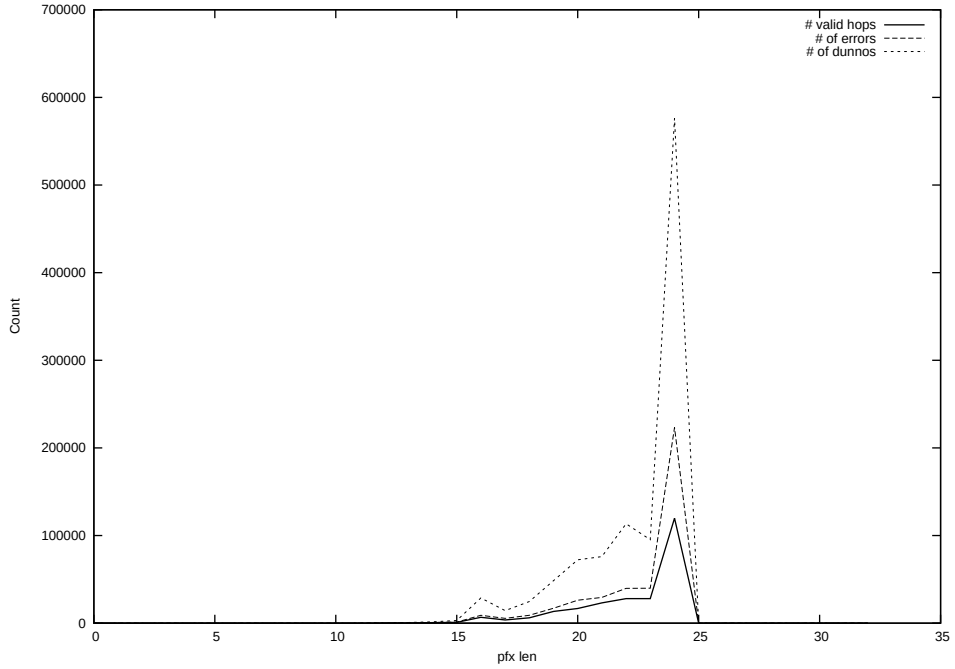
2013-06-07



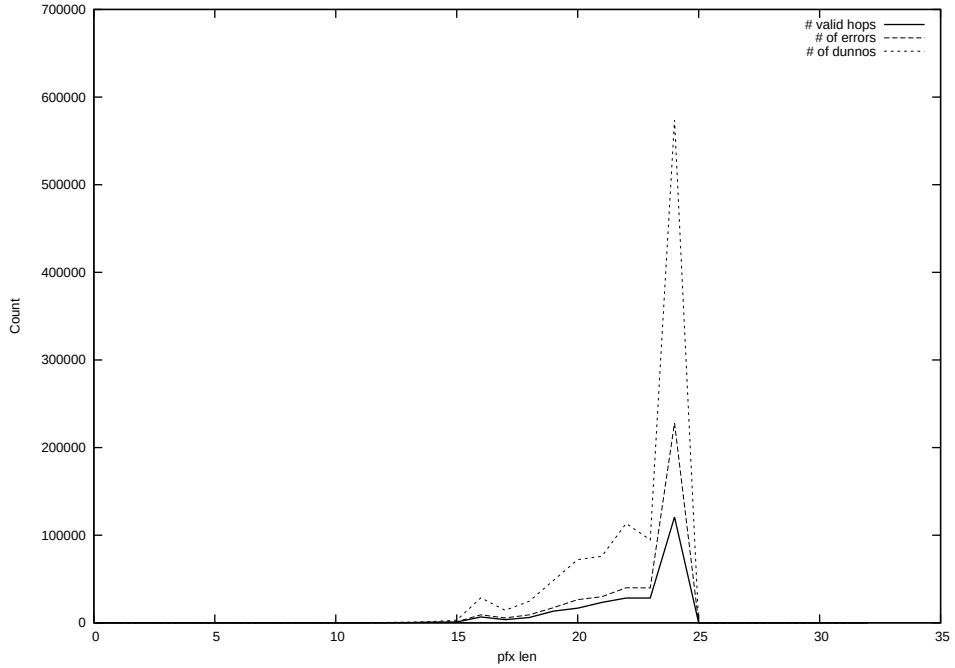
2013-06-09



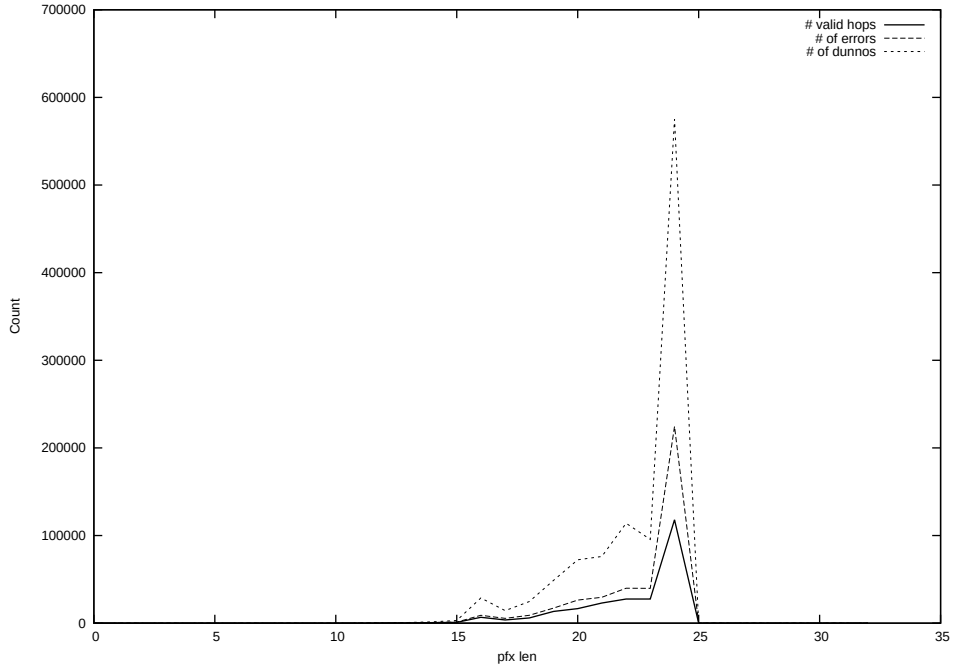
2013-06-10



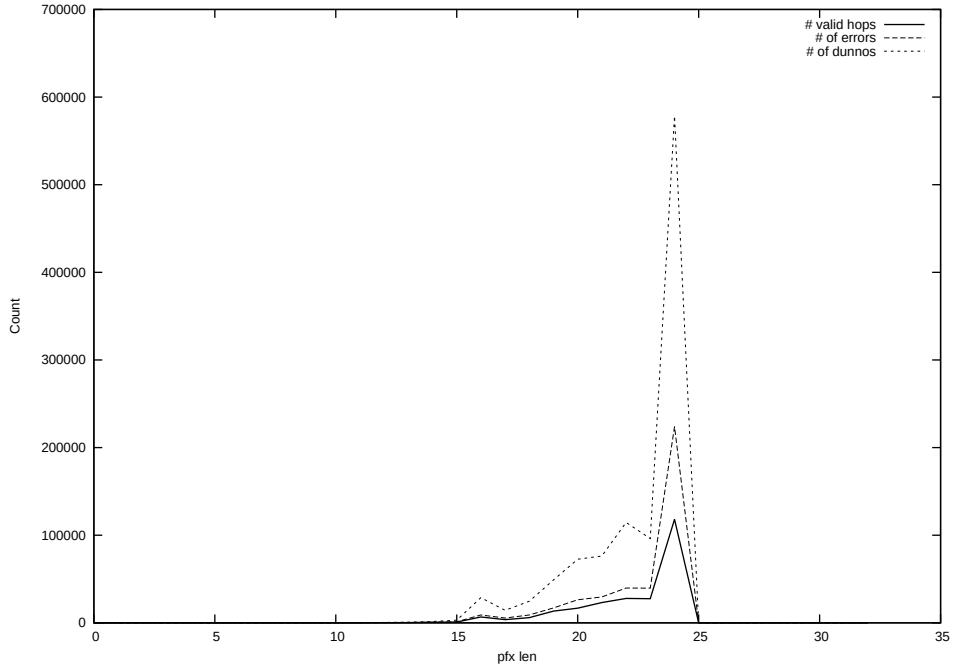
2013-06-12



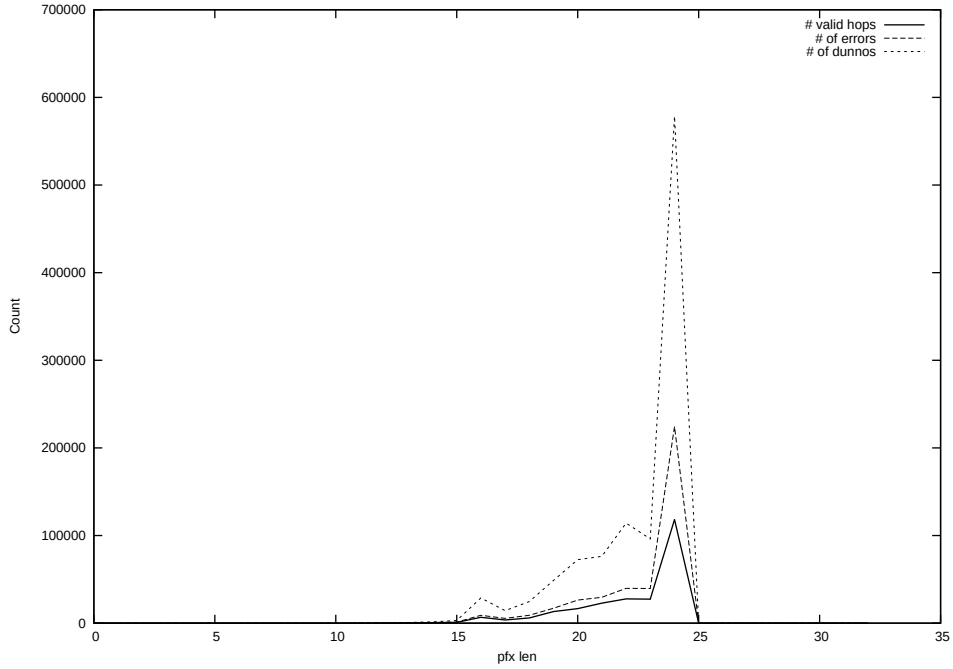
2013-06-13



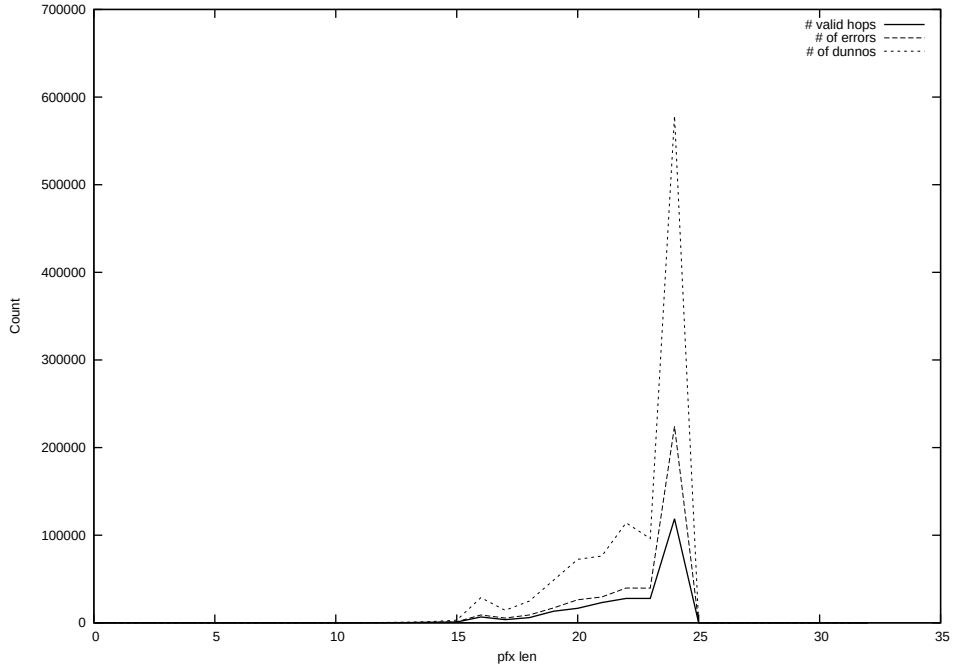
2013-06-14



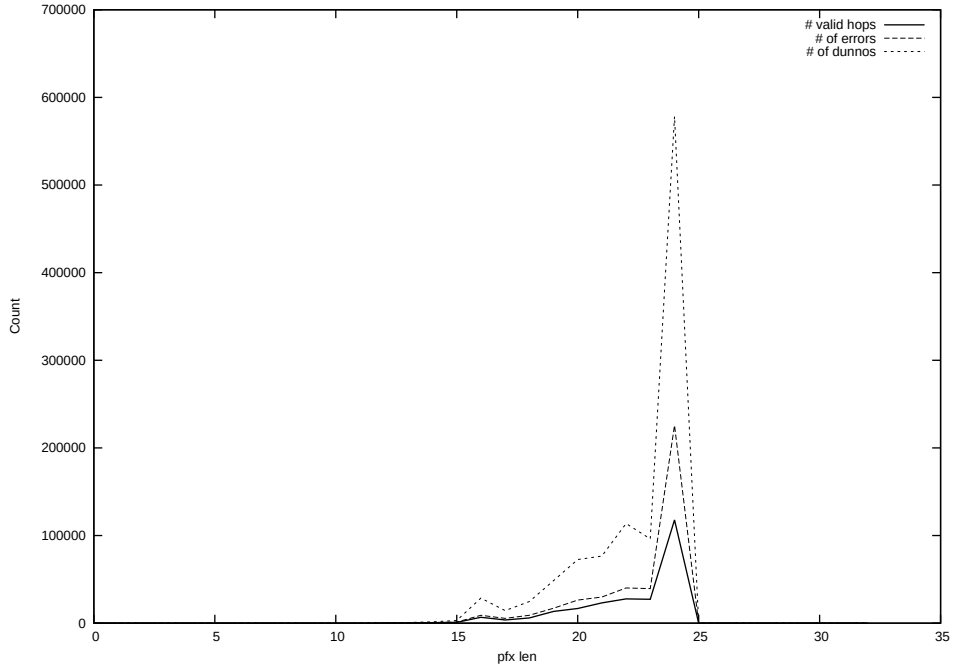
2013-06-15



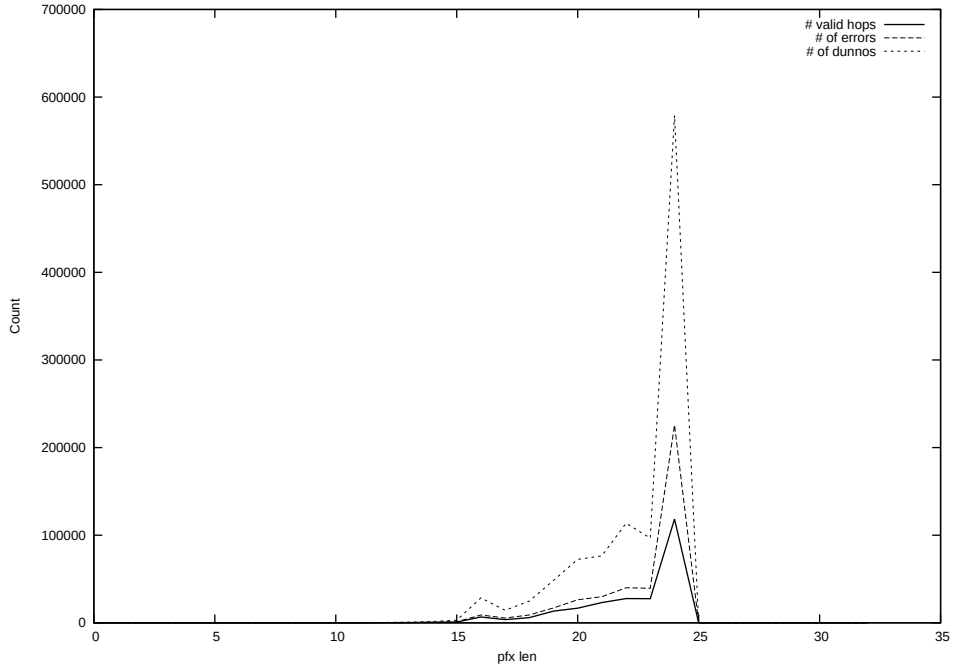
2013-06-16



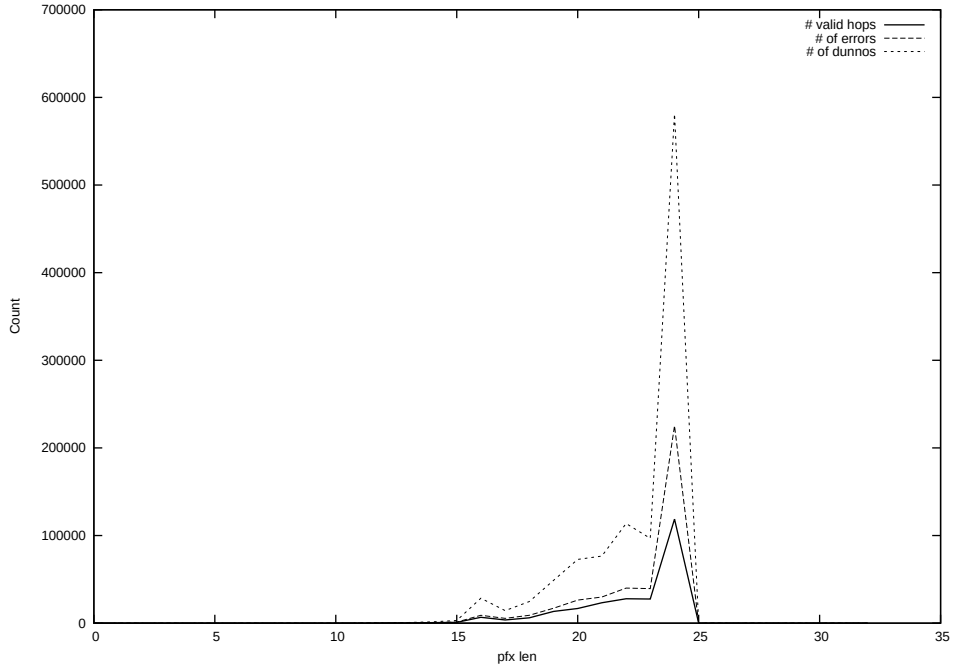
2013-06-17



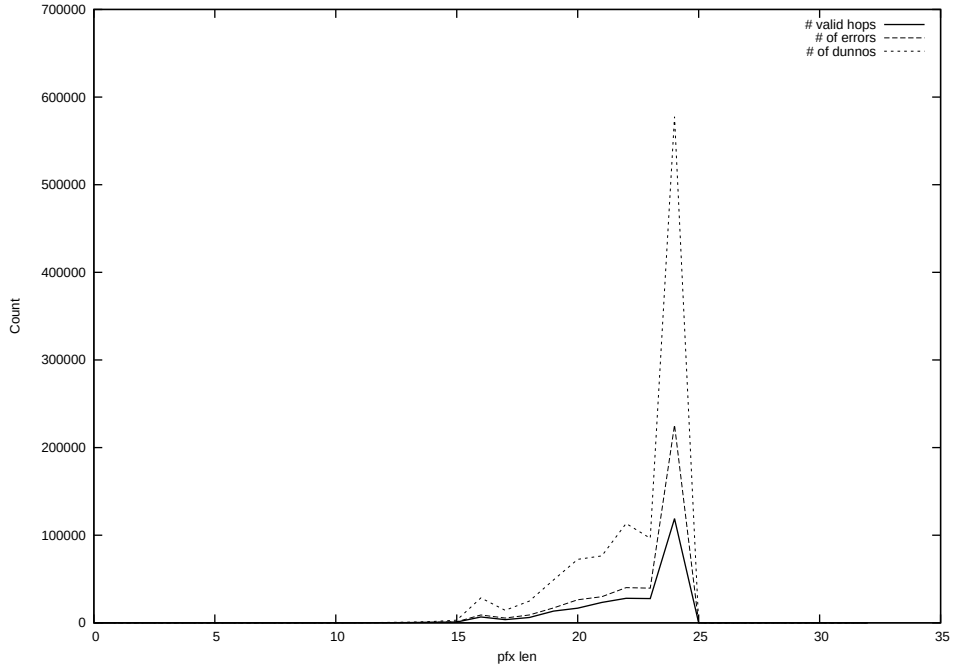
2013-06-18



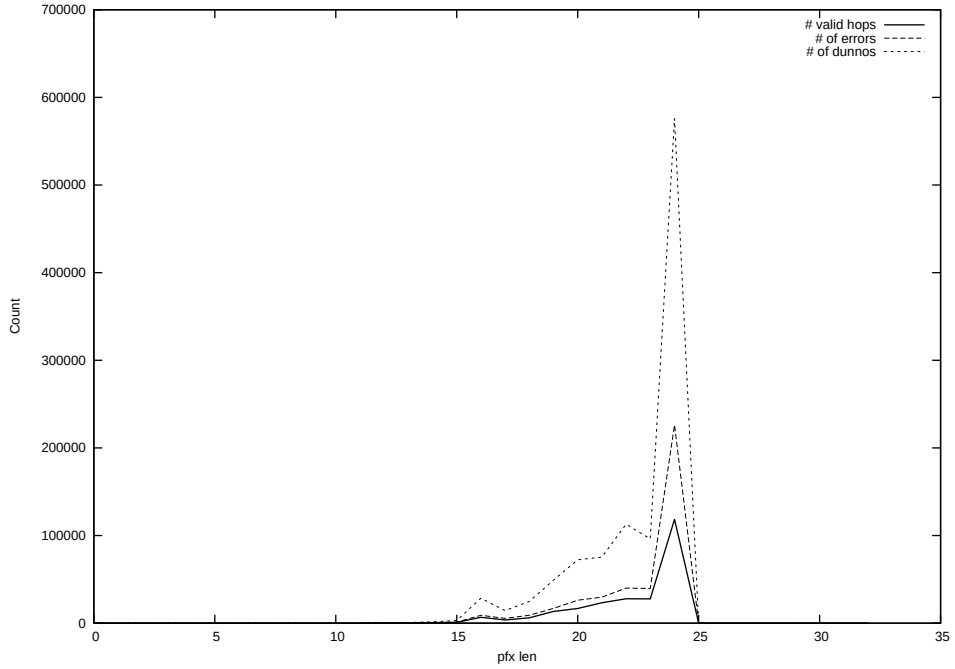
2013-06-19



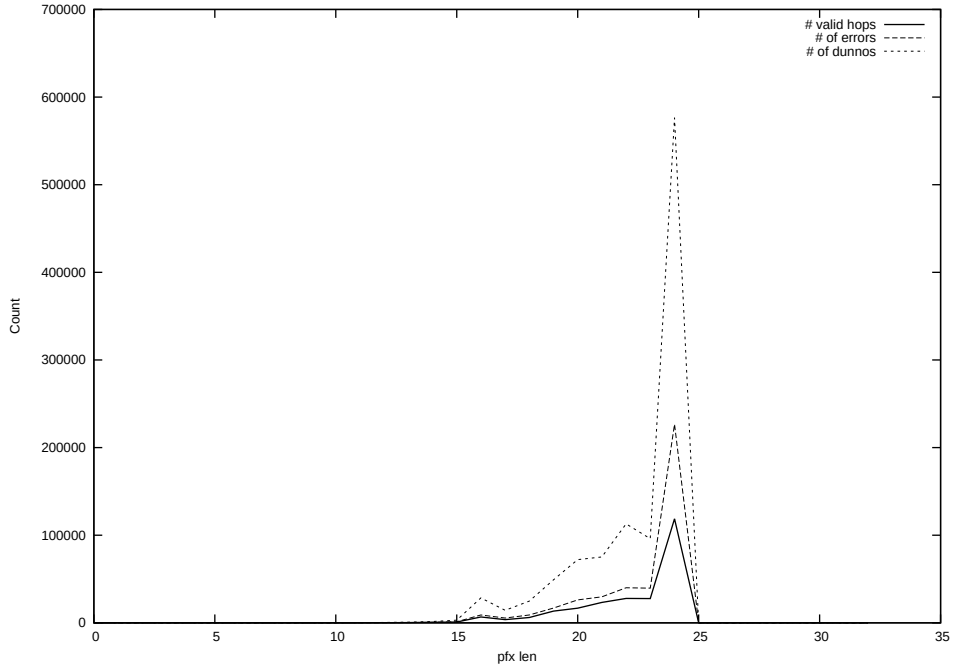
2013-06-20



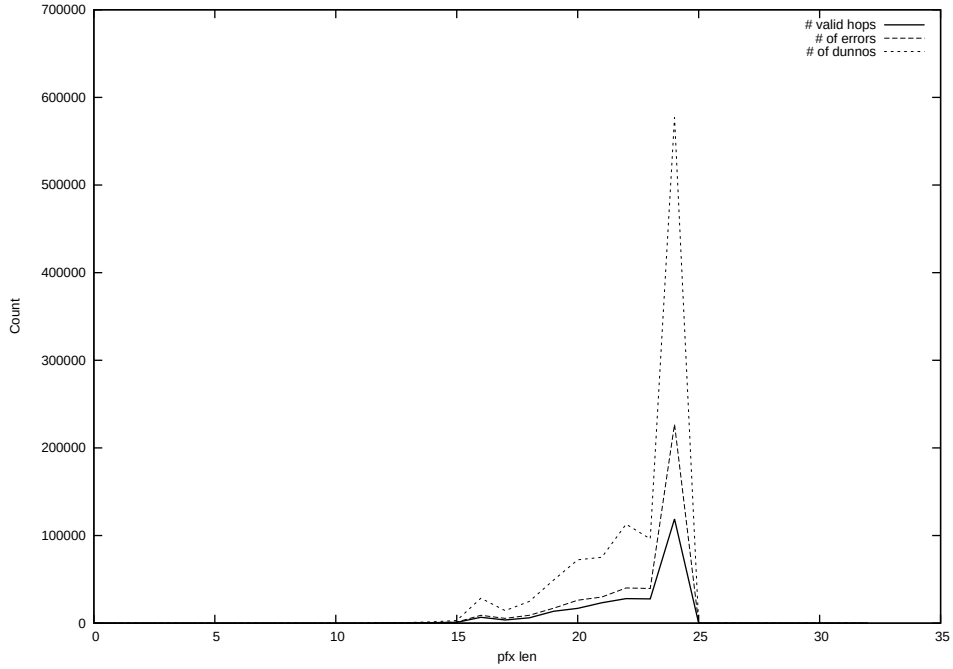
2013-06-21



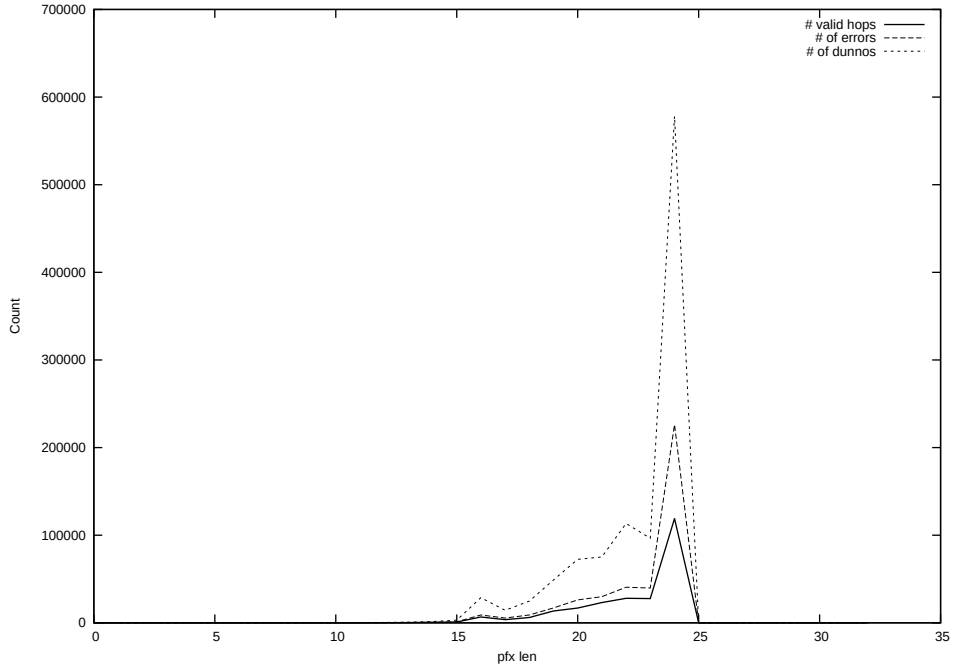
2013-06-22



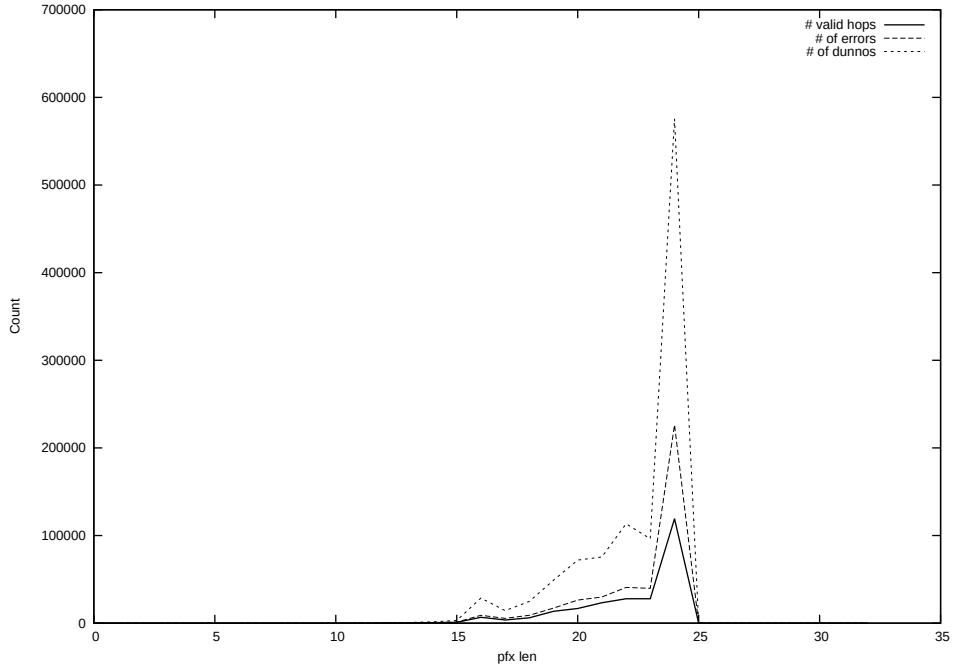
2013-06-23



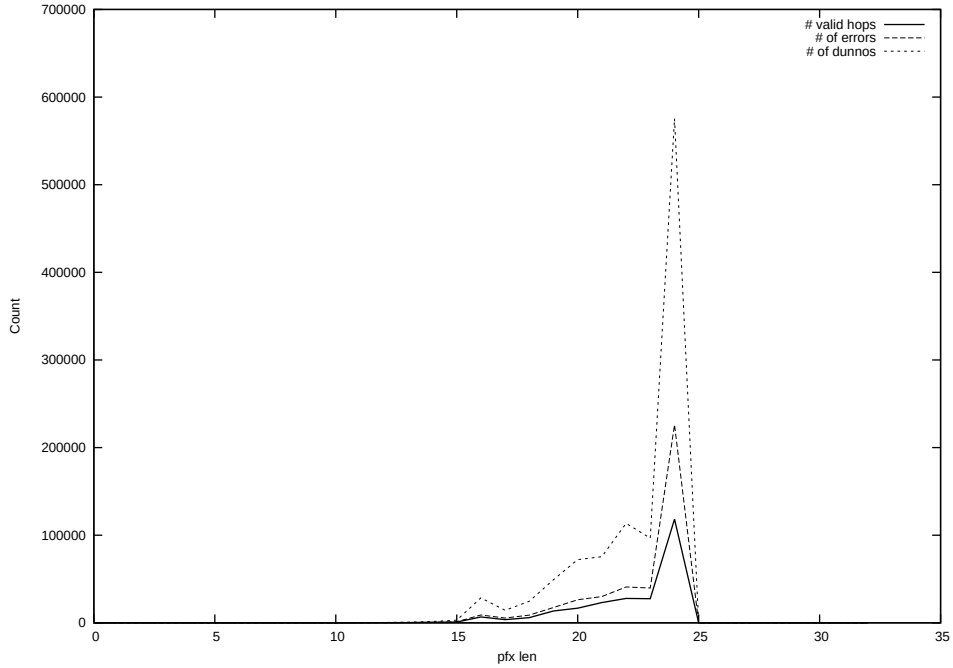
2013-06-24



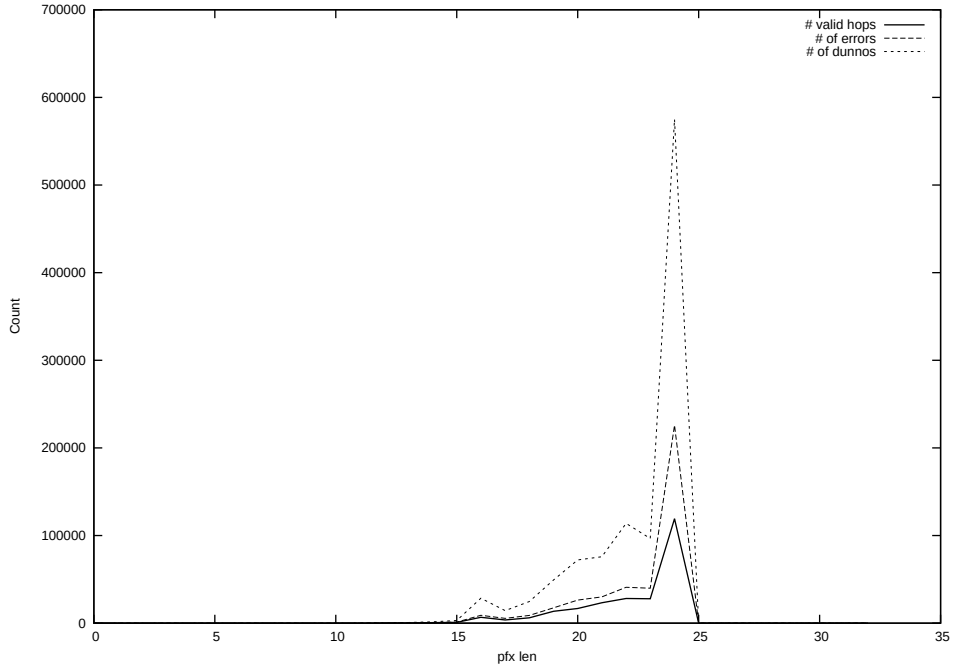
2013-06-25



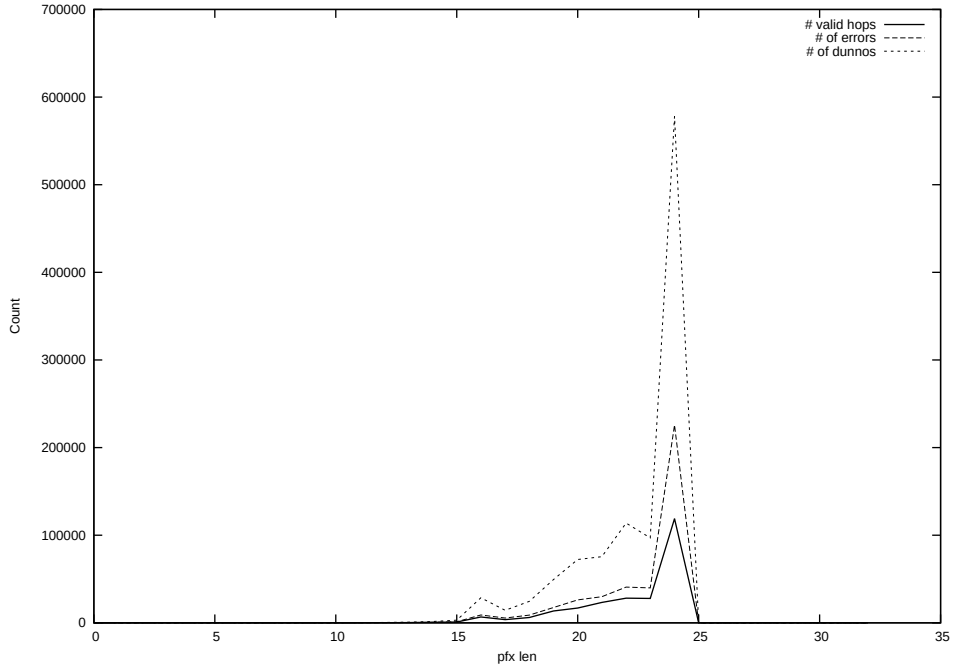
2013-06-26



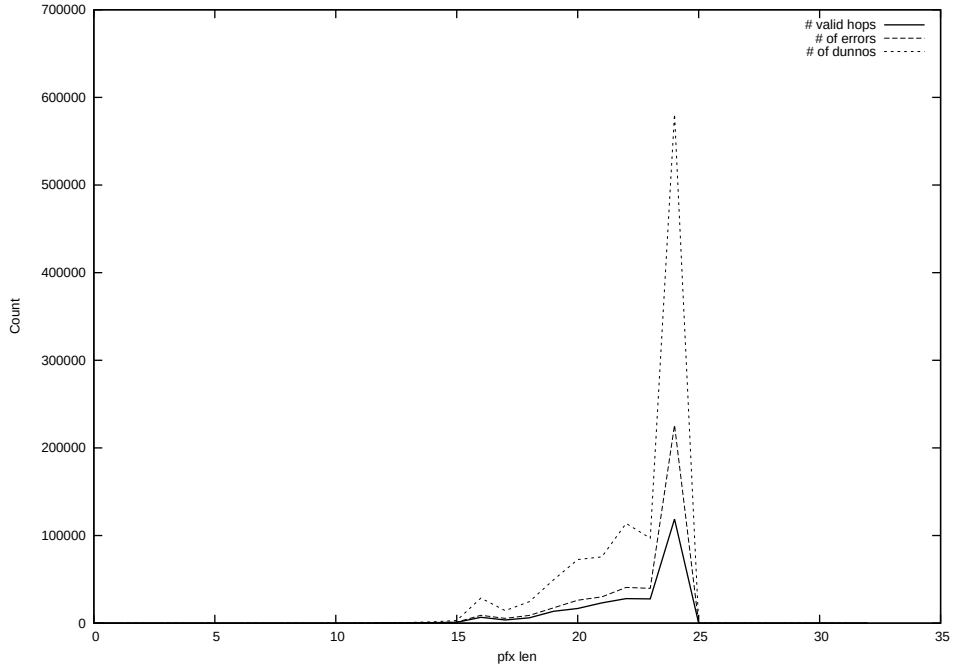
2013-06-27



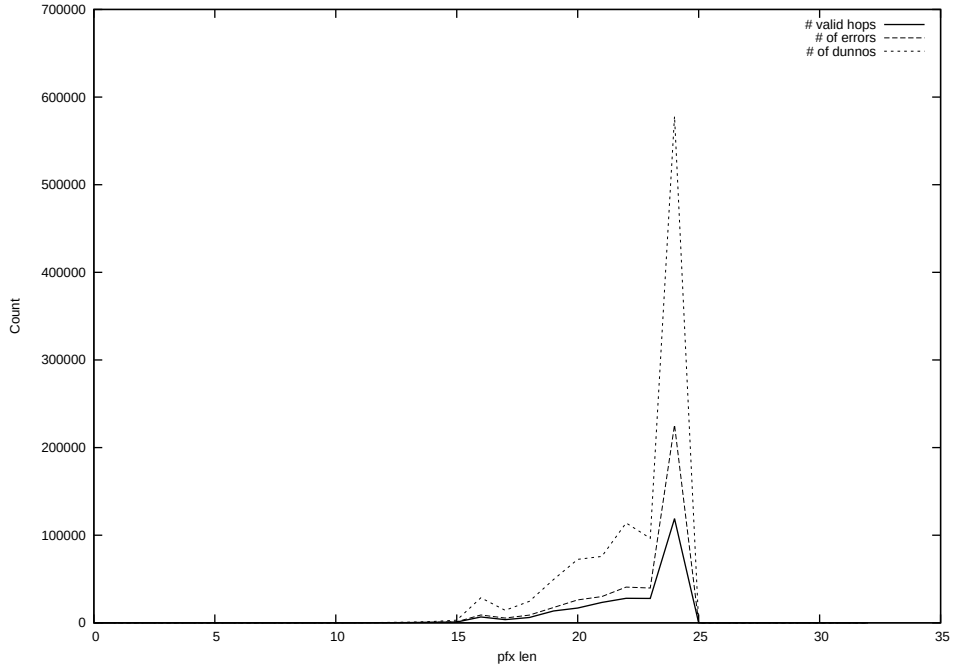
2013-06-28



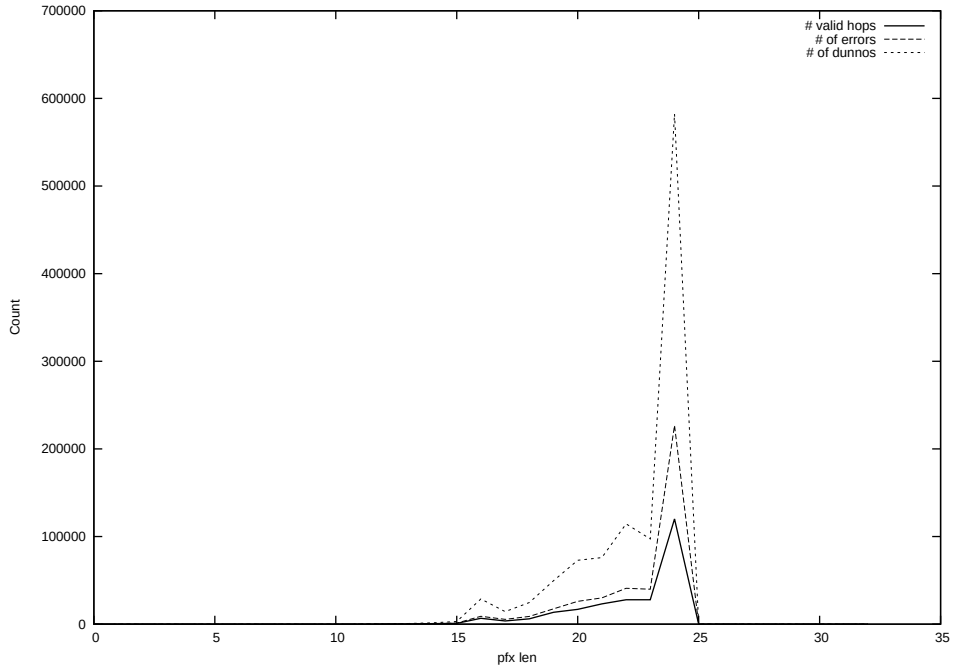
2013-06-29



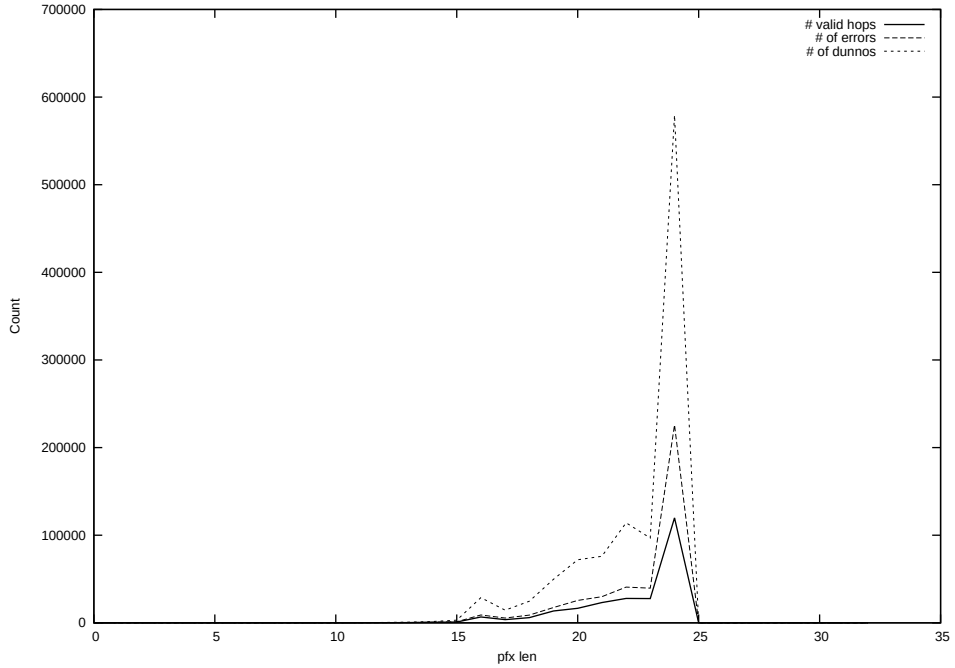
2013-06-30



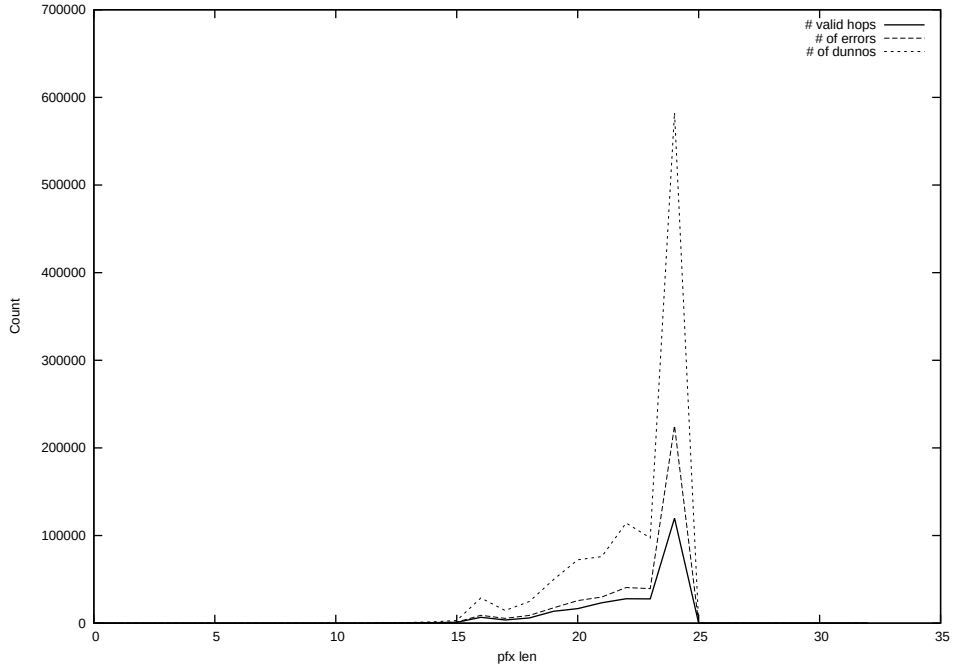
2013-07-01



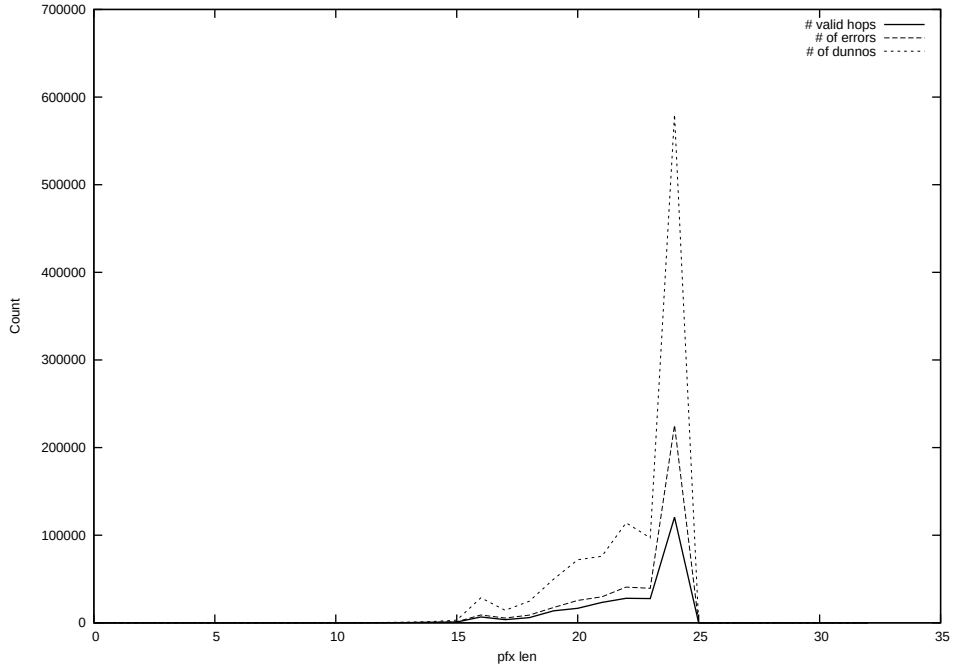
2013-07-02



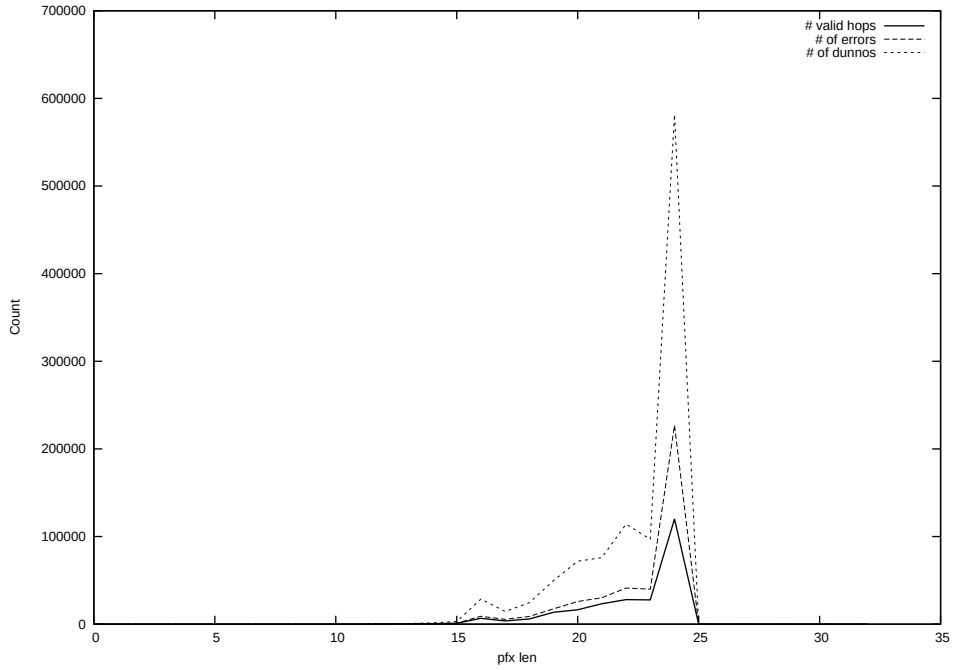
2013-07-03



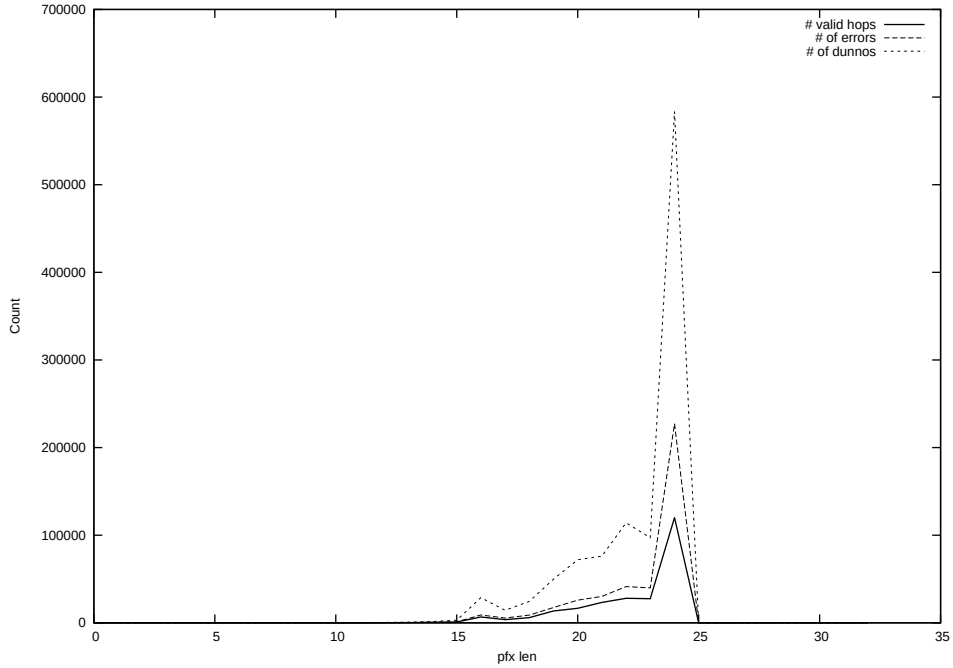
2013-07-04



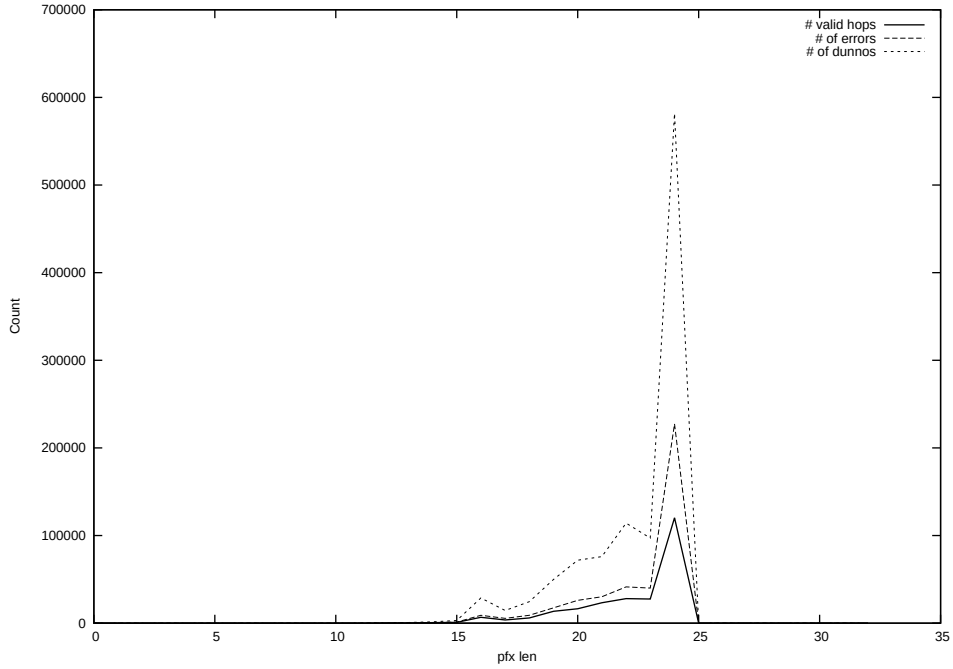
2013-07-05



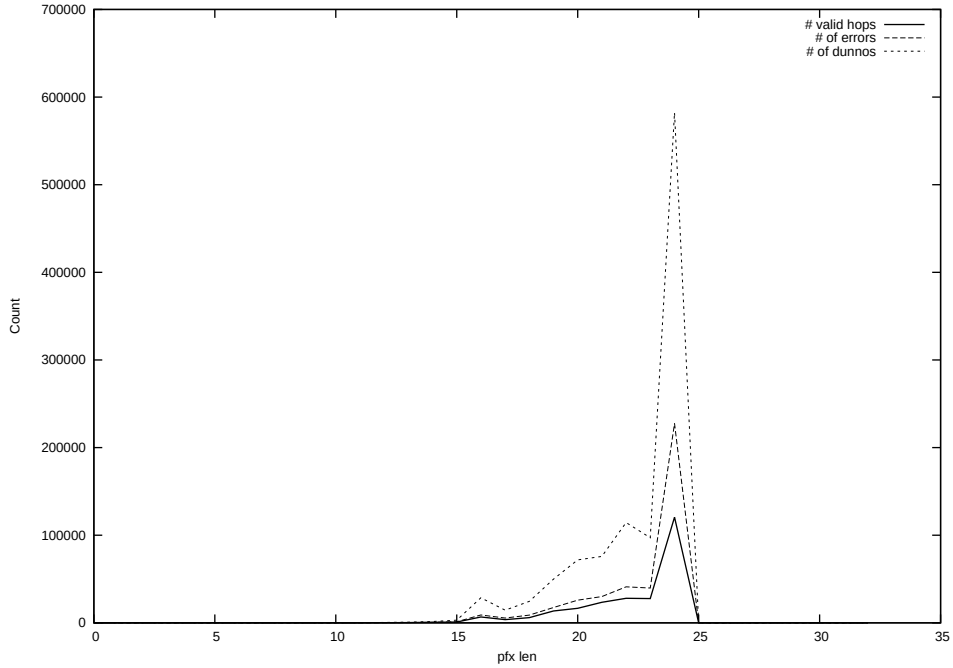
2013-07-06



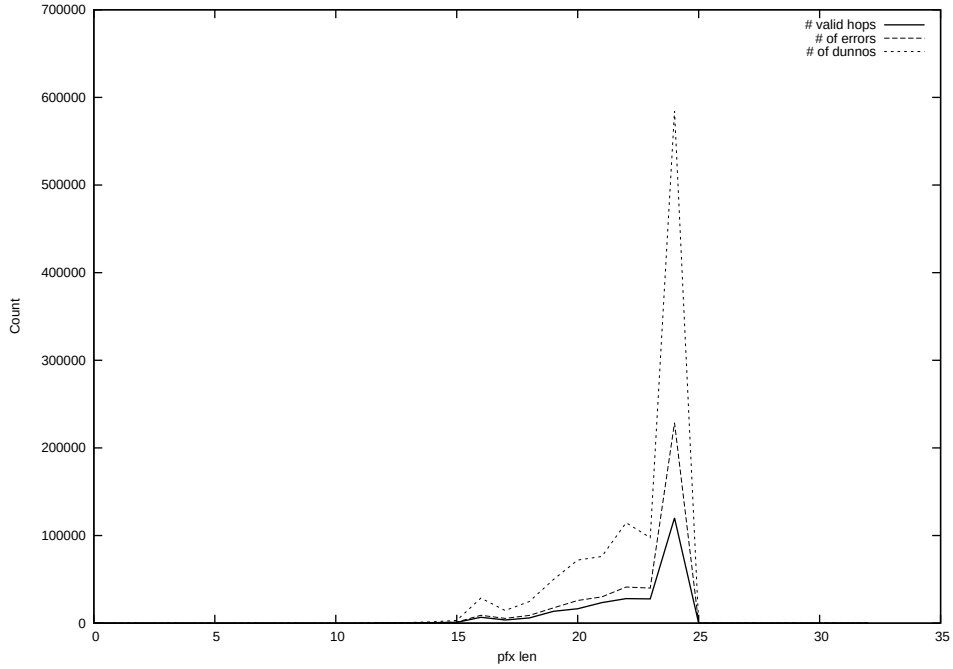
2013-07-07



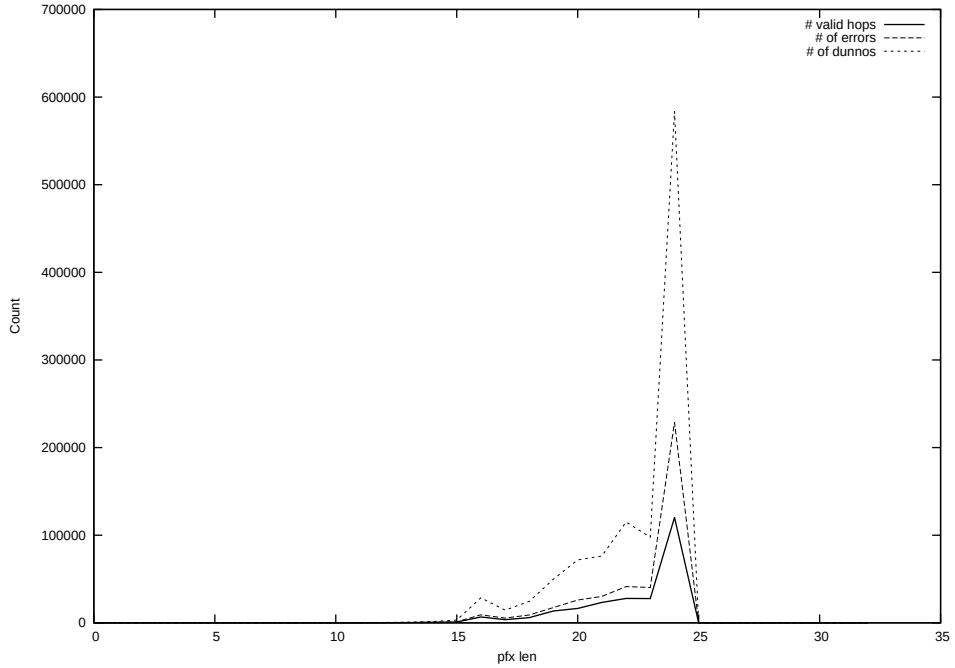
2013-07-08



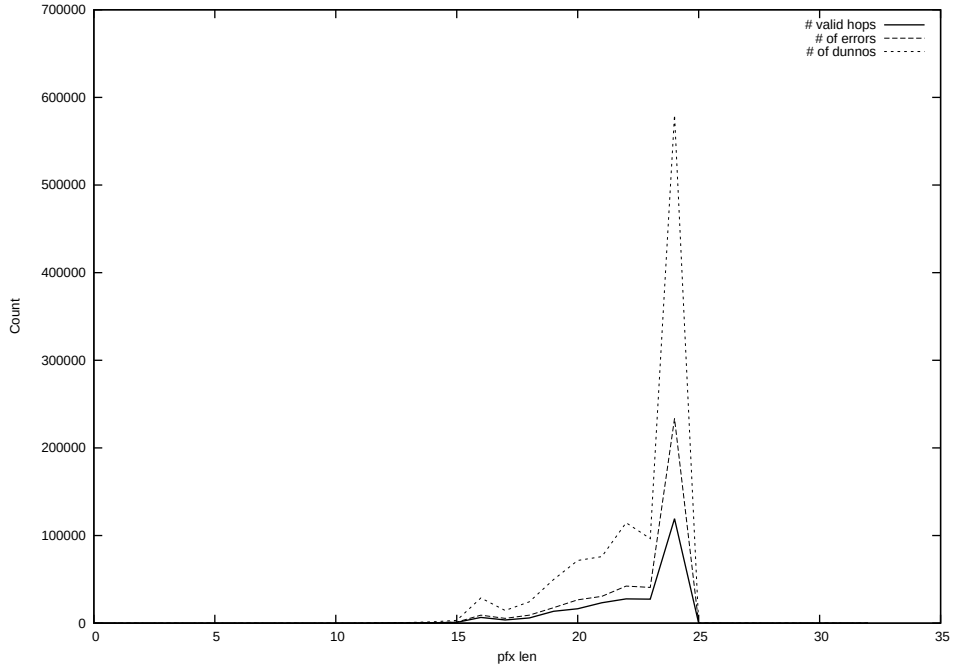
2013-07-09



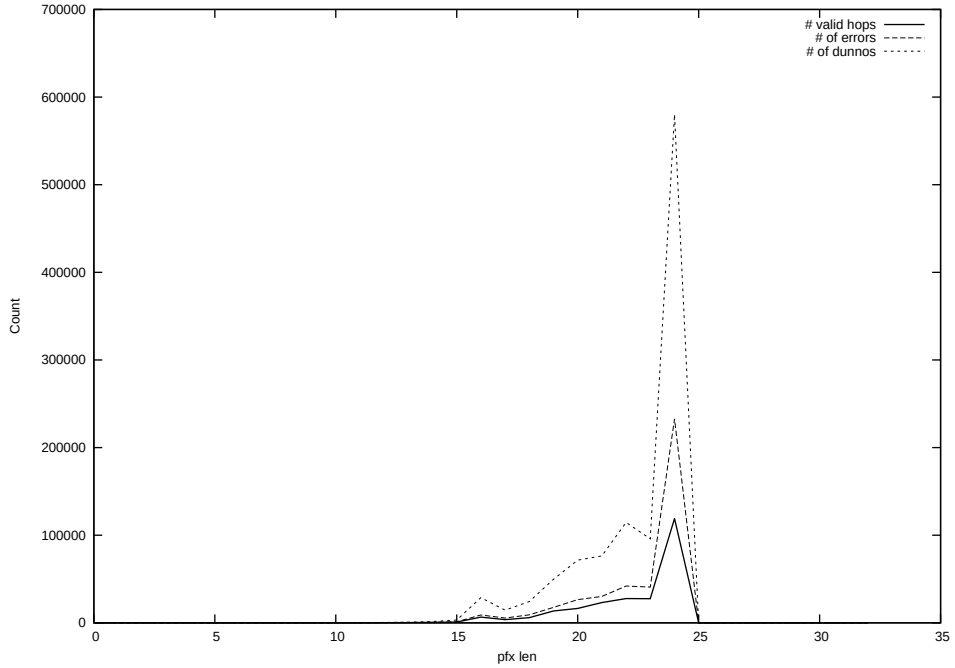
2013-07-10



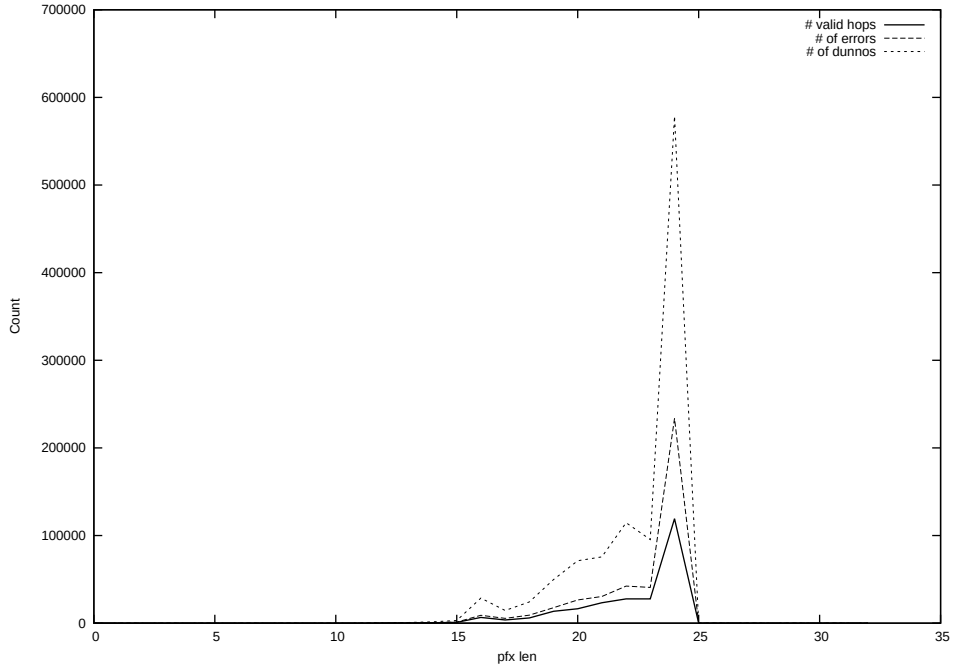
2013-07-11



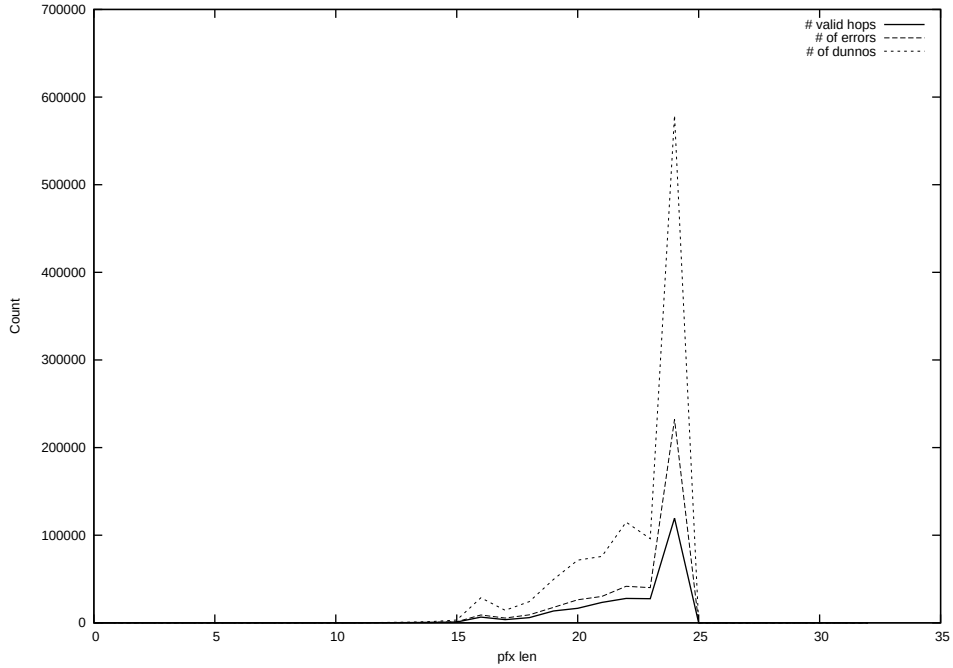
2013-07-12



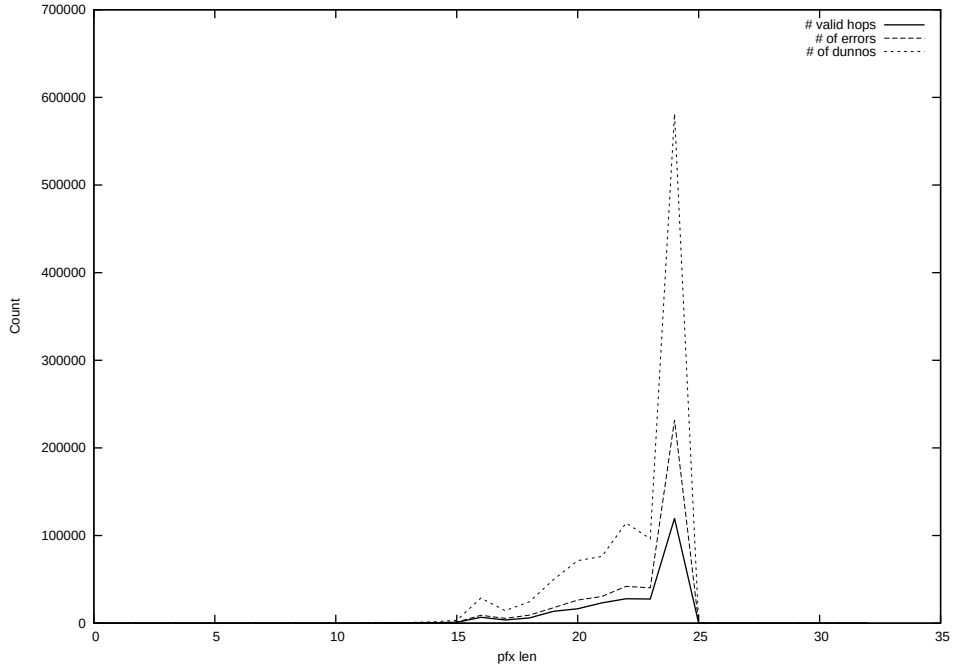
2013-07-13



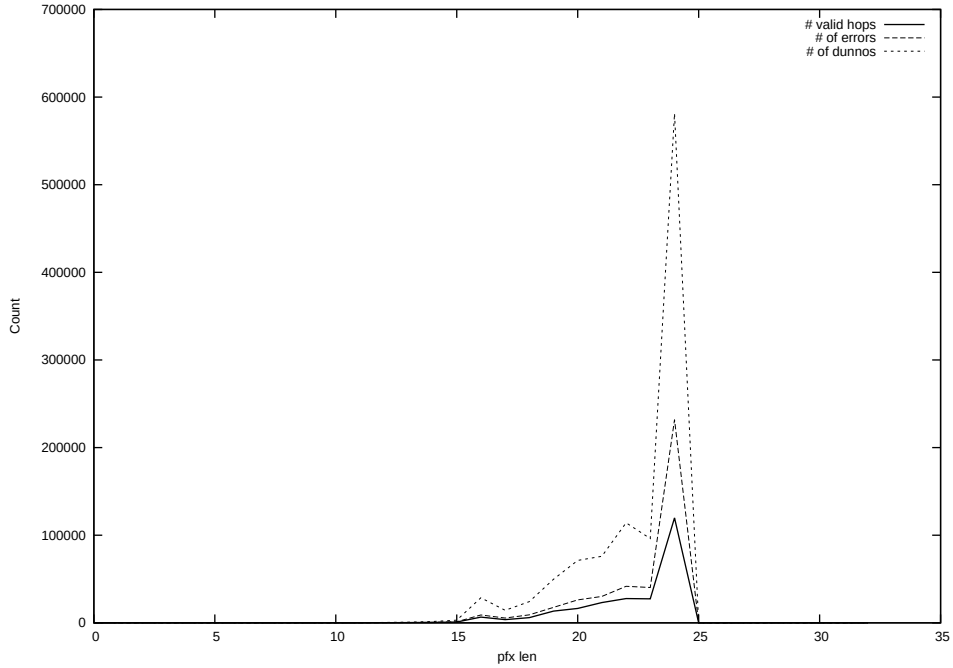
2013-07-14



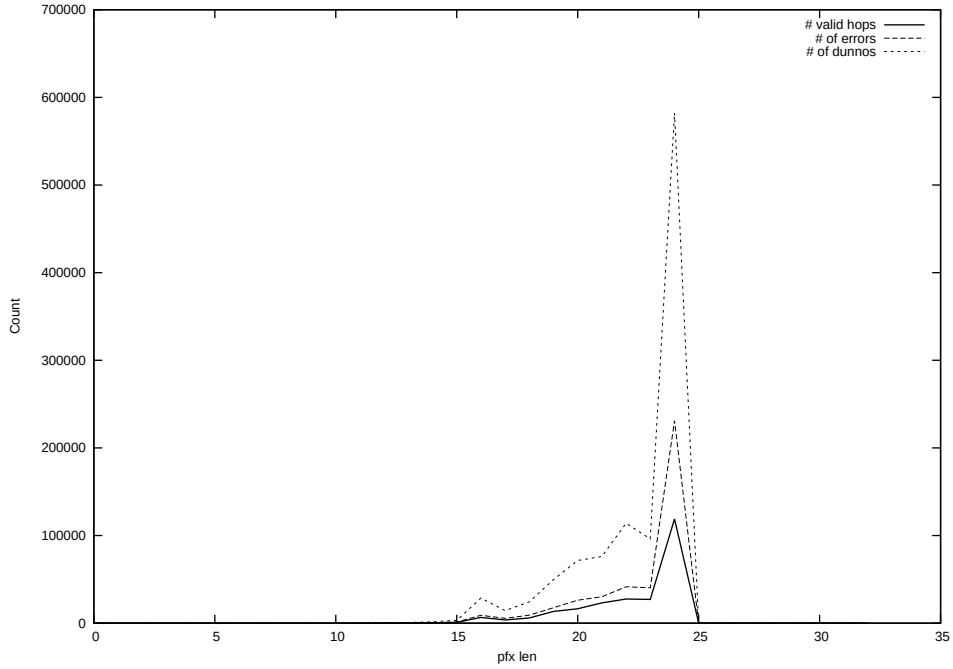
2013-07-15



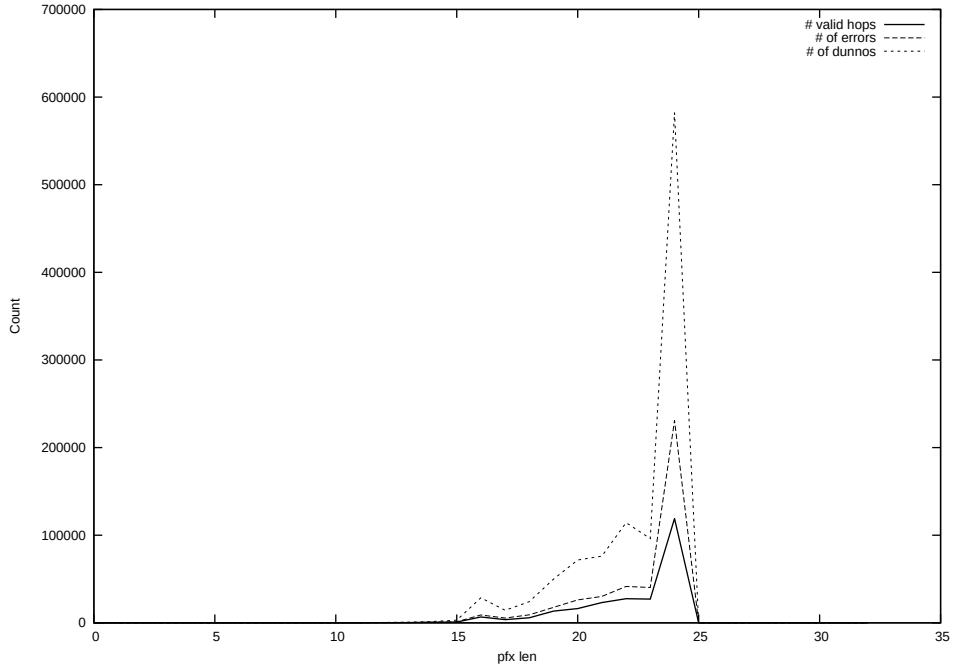
2013-07-16



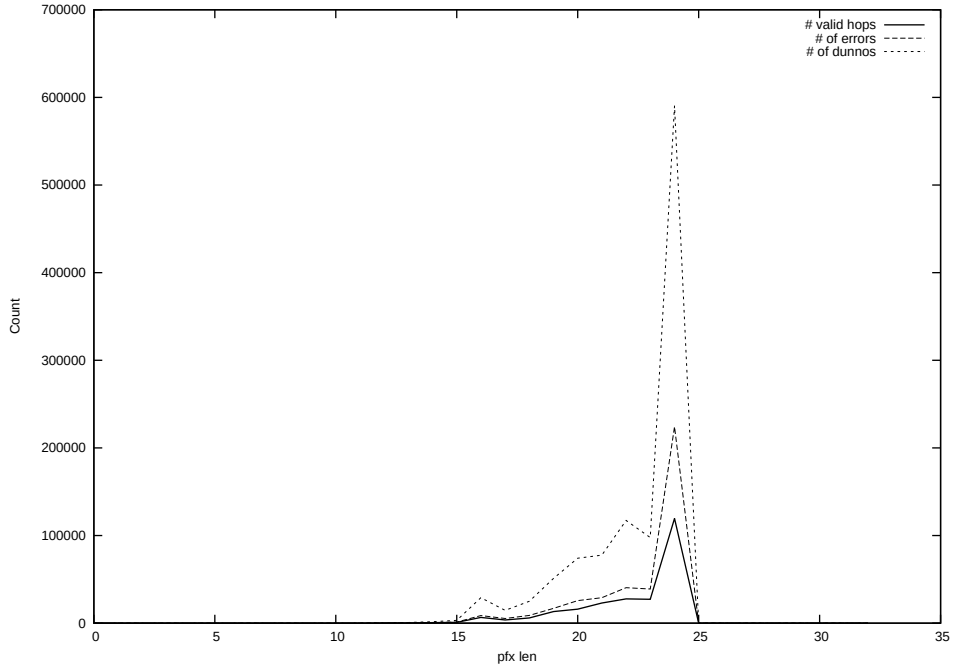
2013-07-17



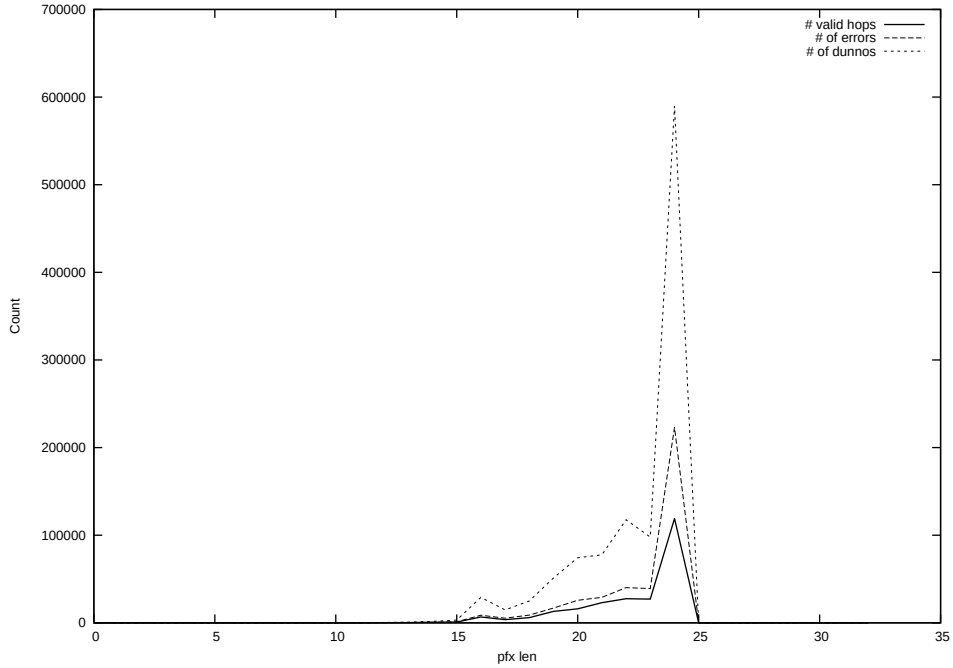
2013-07-18



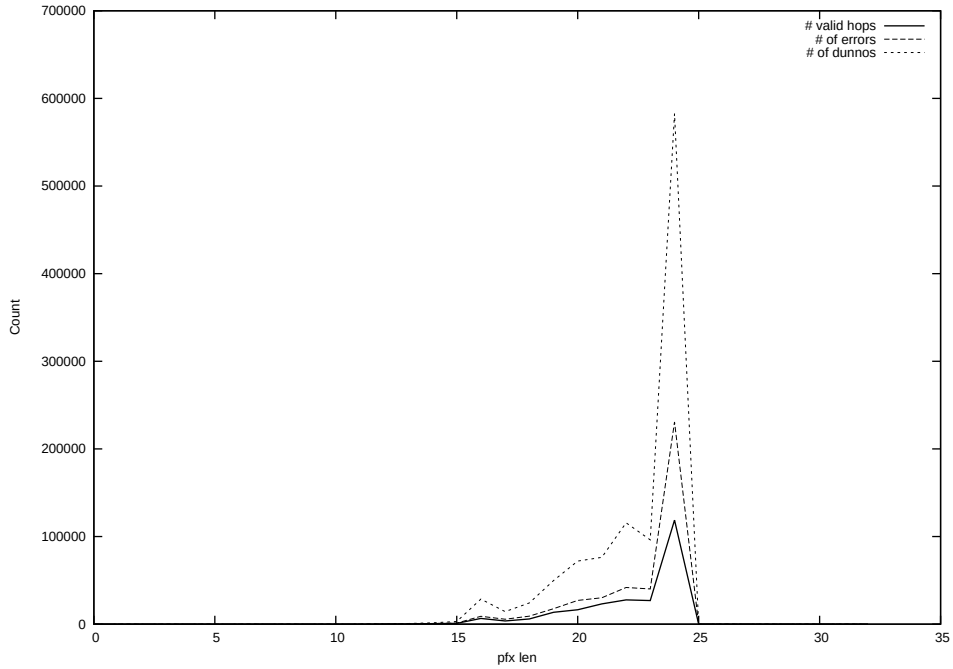
2013-07-19



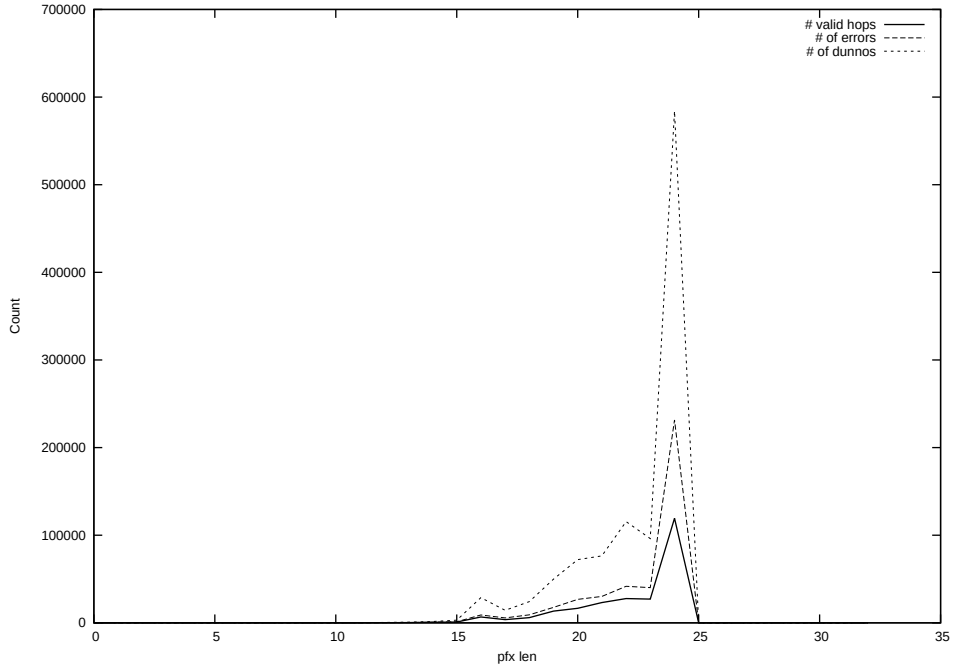
2013-07-20



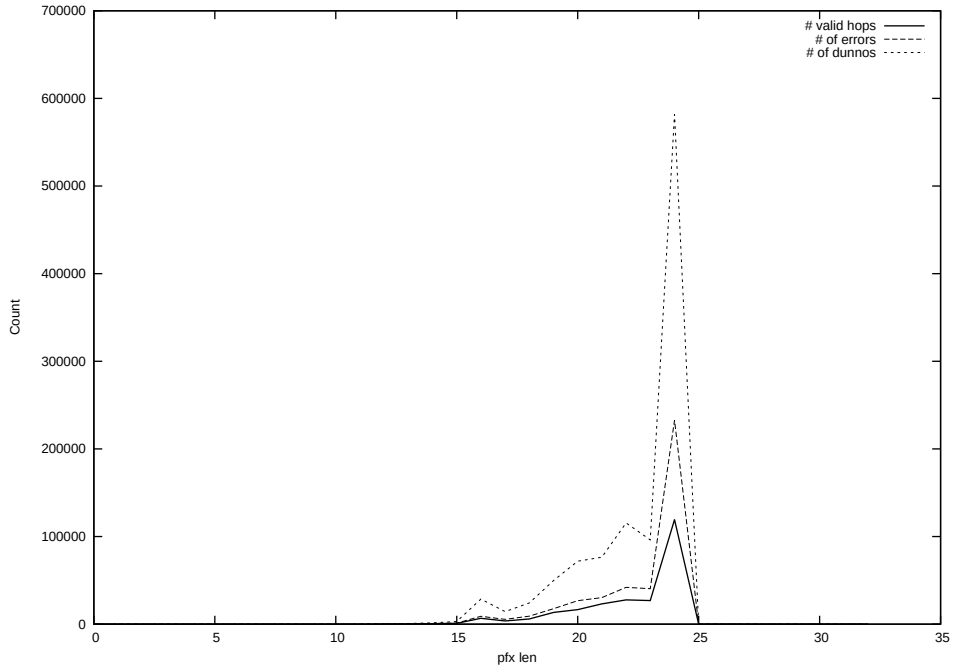
2013-07-21



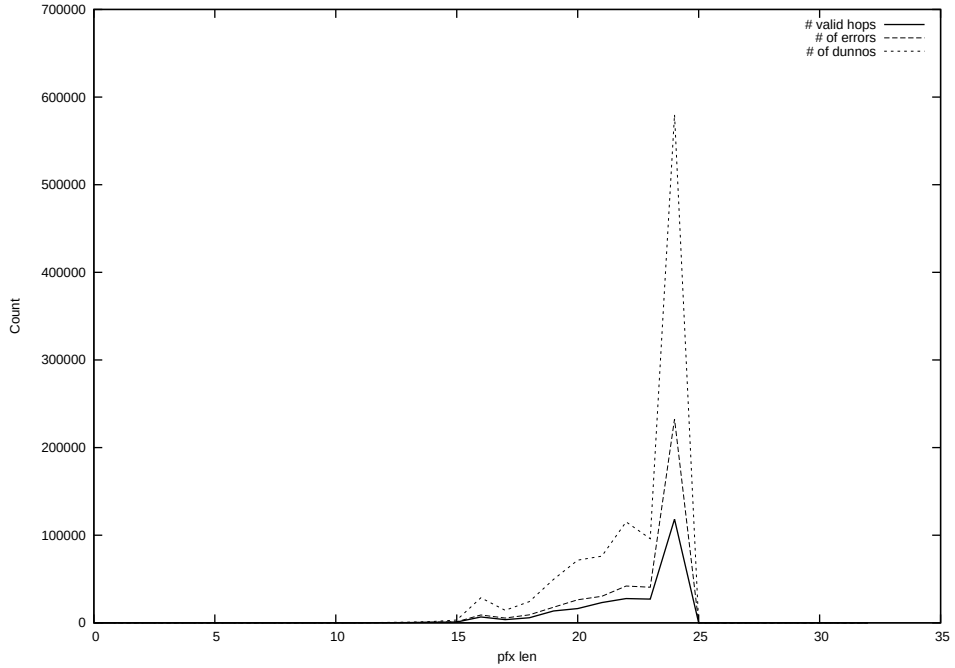
2013-07-22



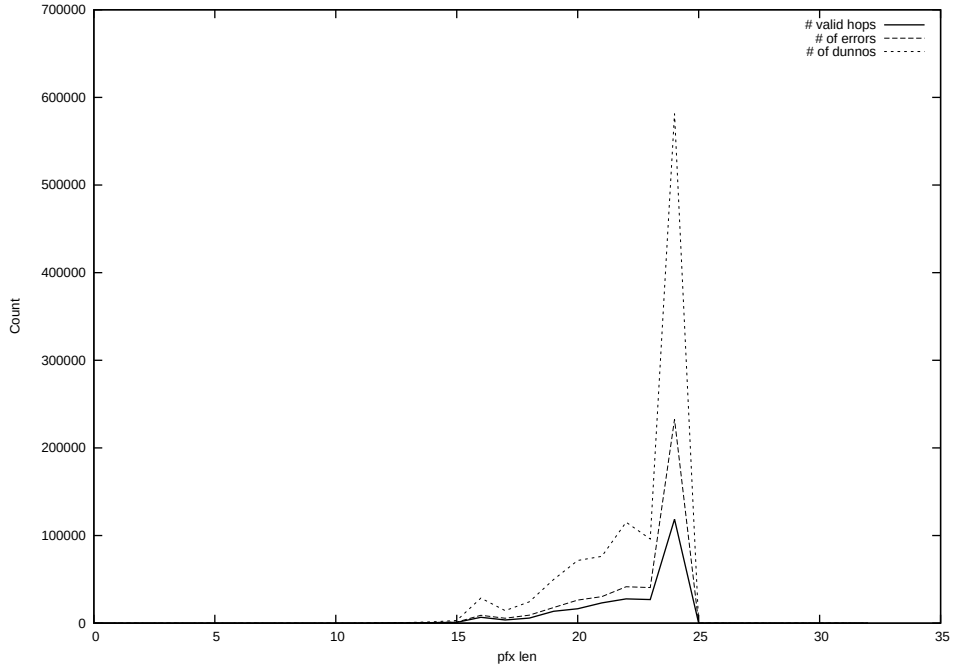
2013-07-23



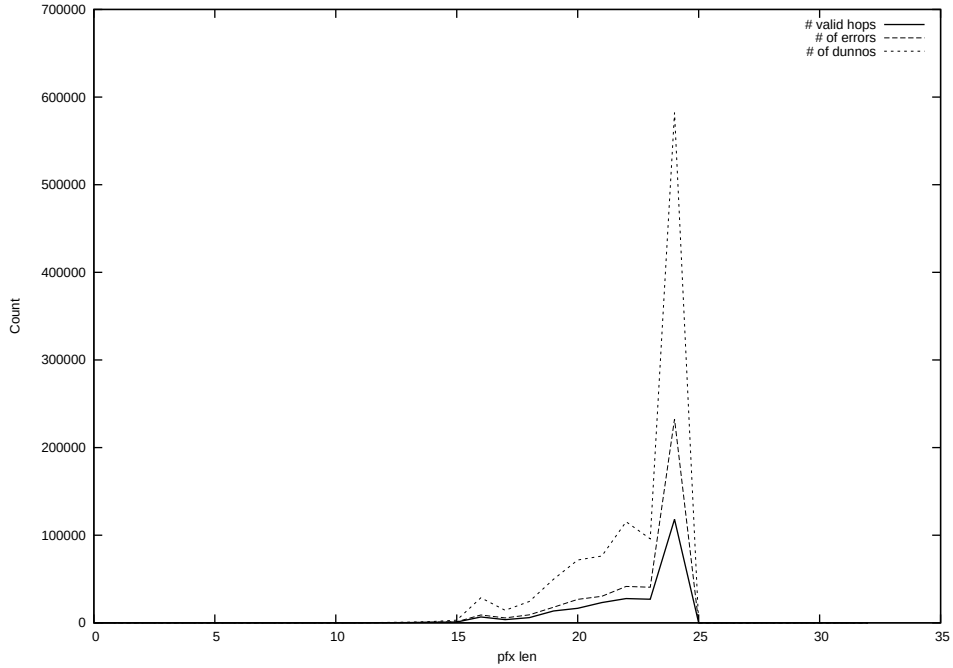
2013-07-24



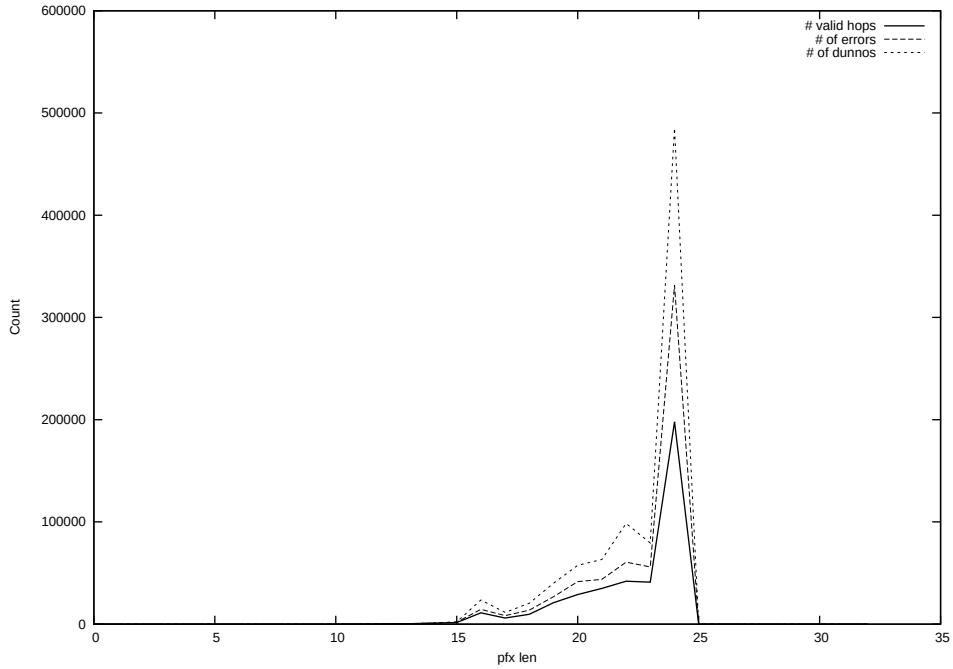
2013-07-25



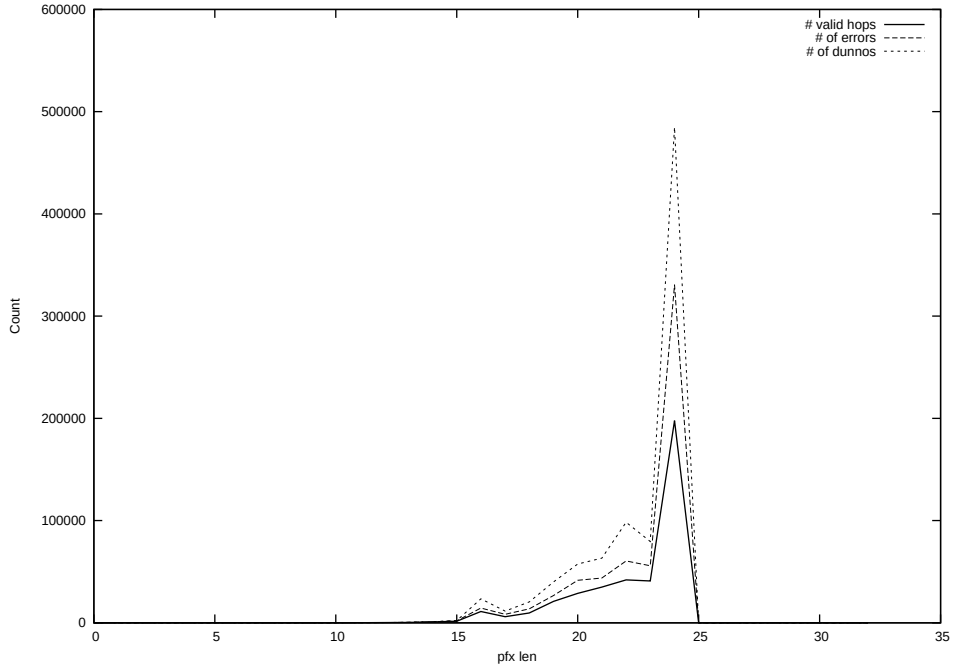
2013-07-26



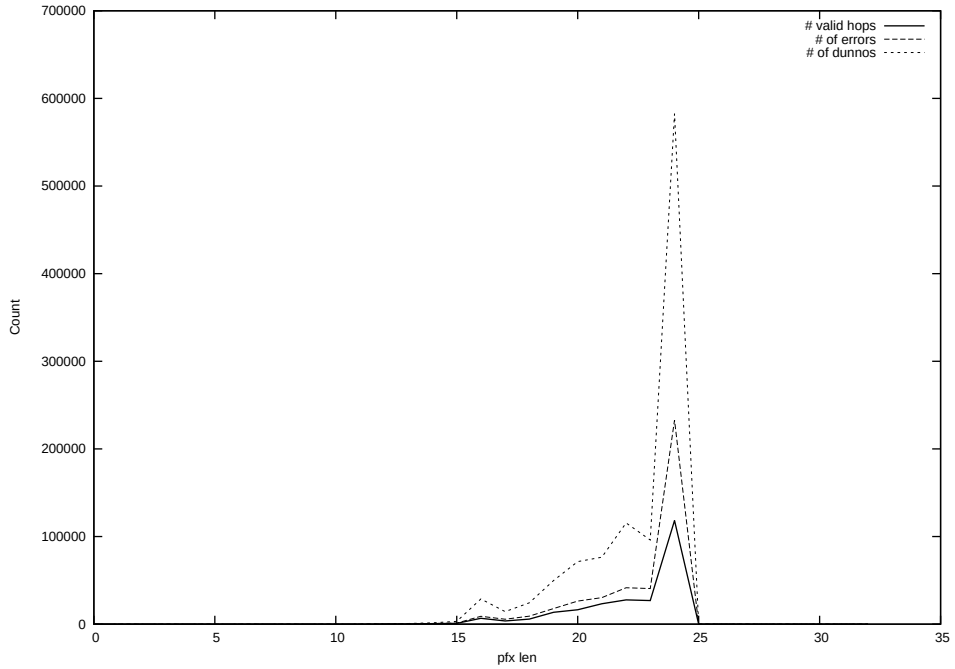
2013-07-27



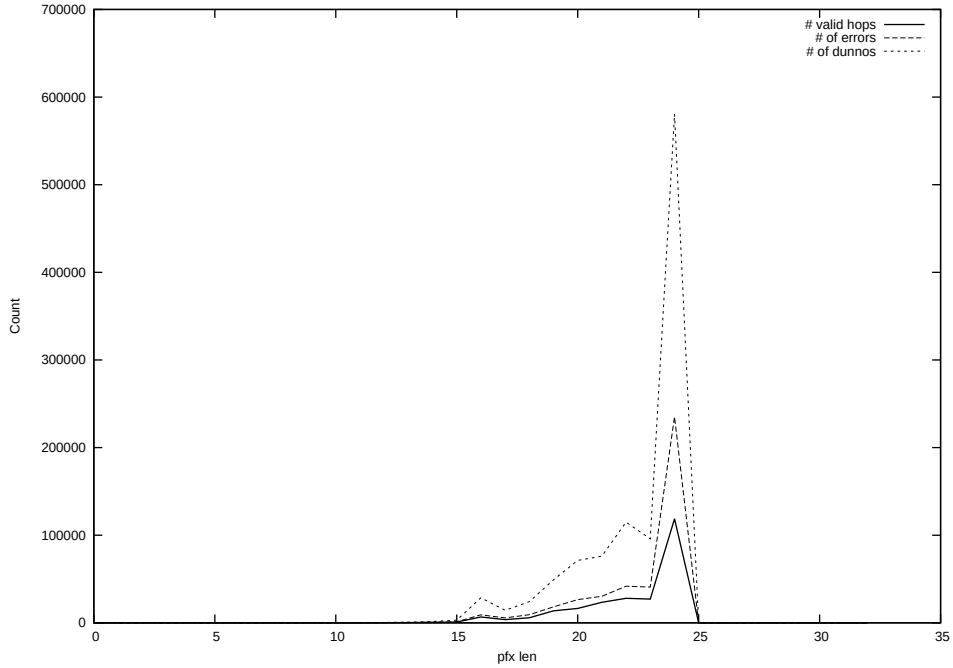
2013-07-28



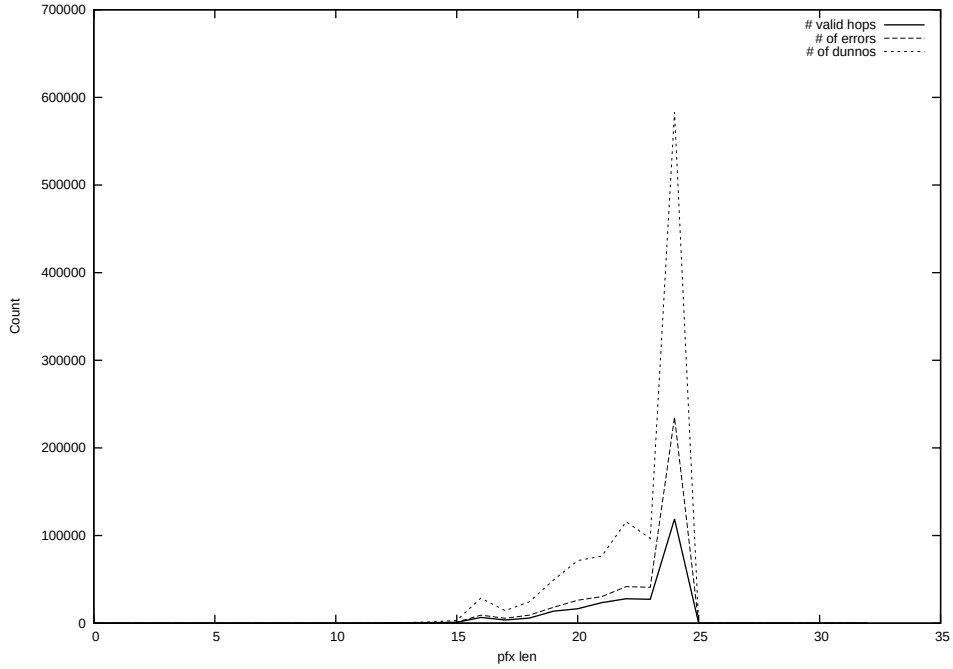
2013-07-29



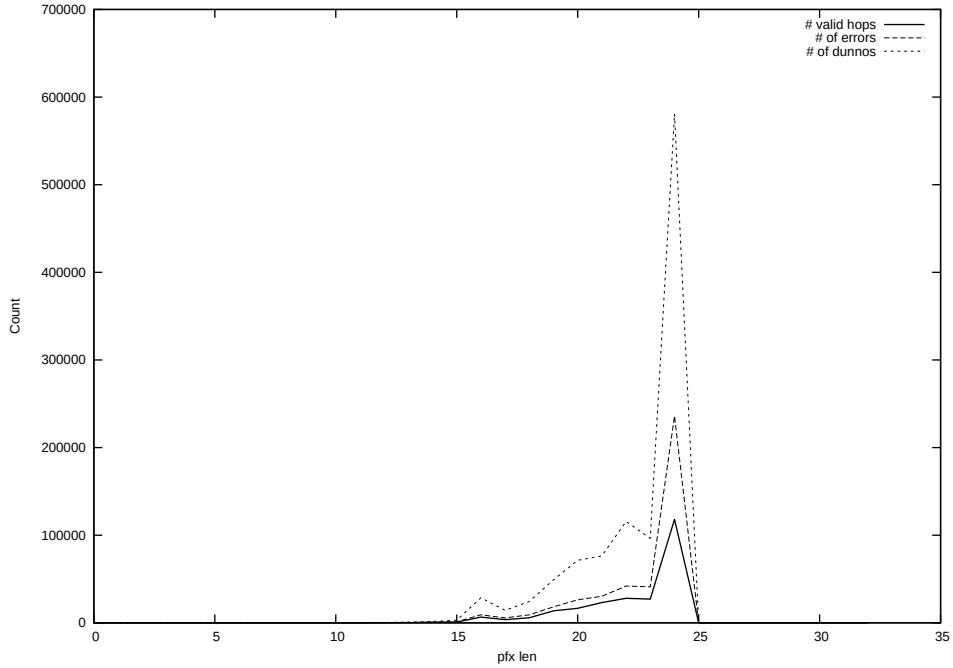
2013-07-30



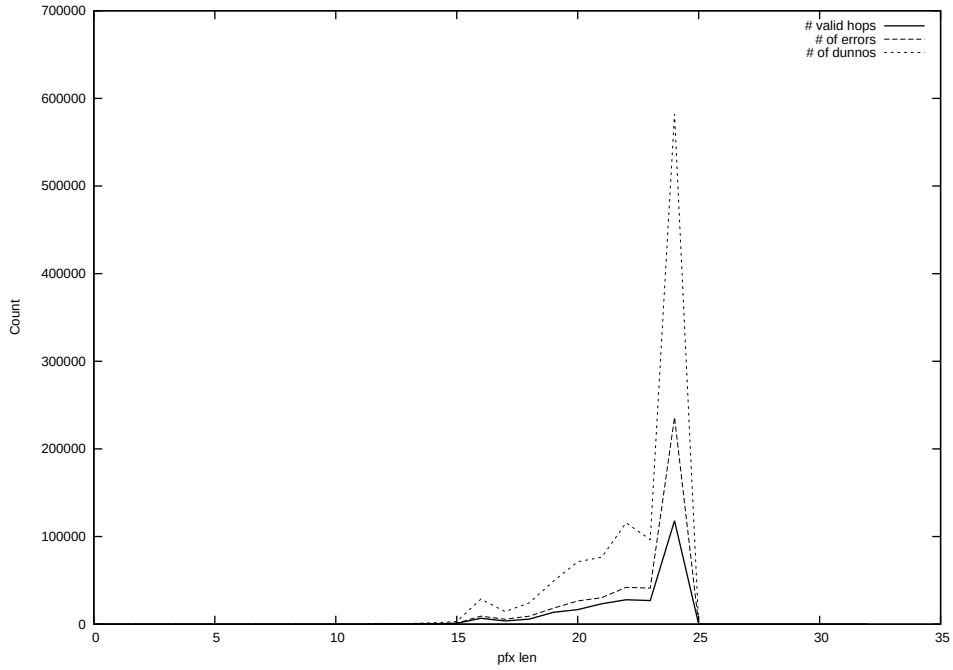
2013-07-31



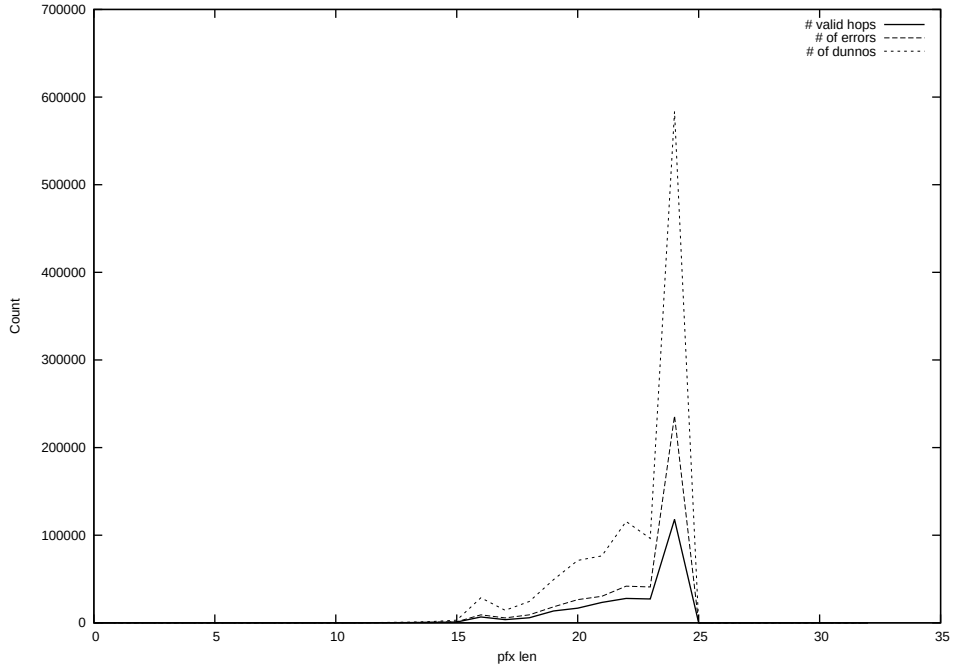
2013-08-01



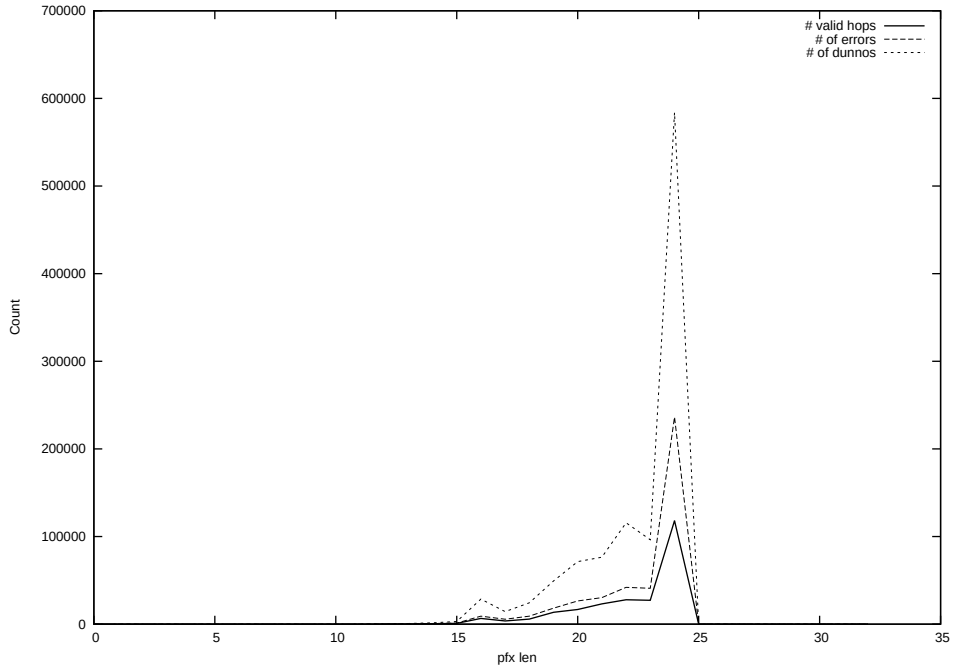
2013-08-02



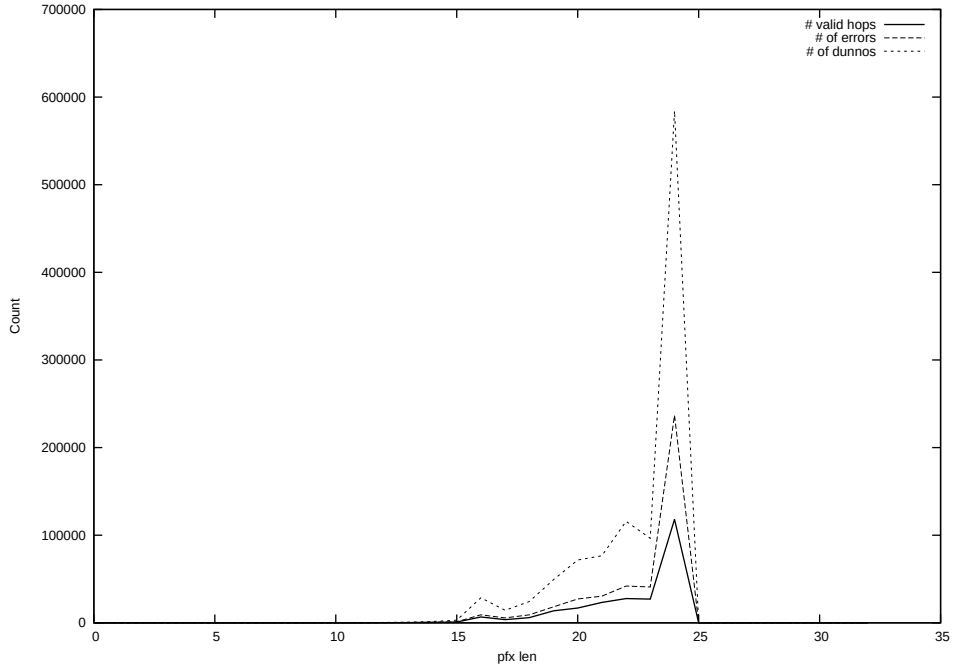
2013-08-03



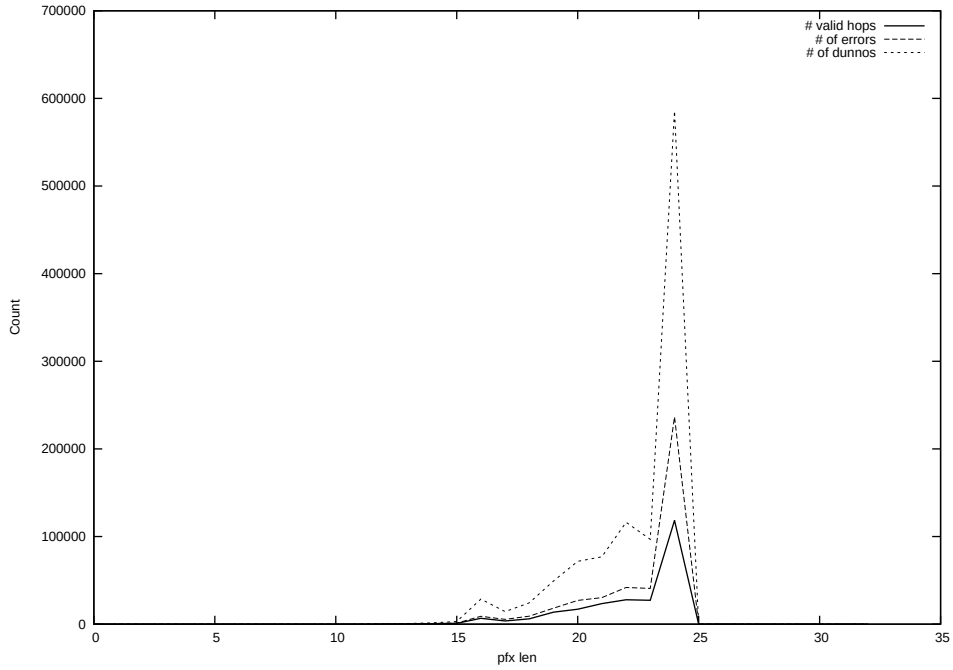
2013-08-04



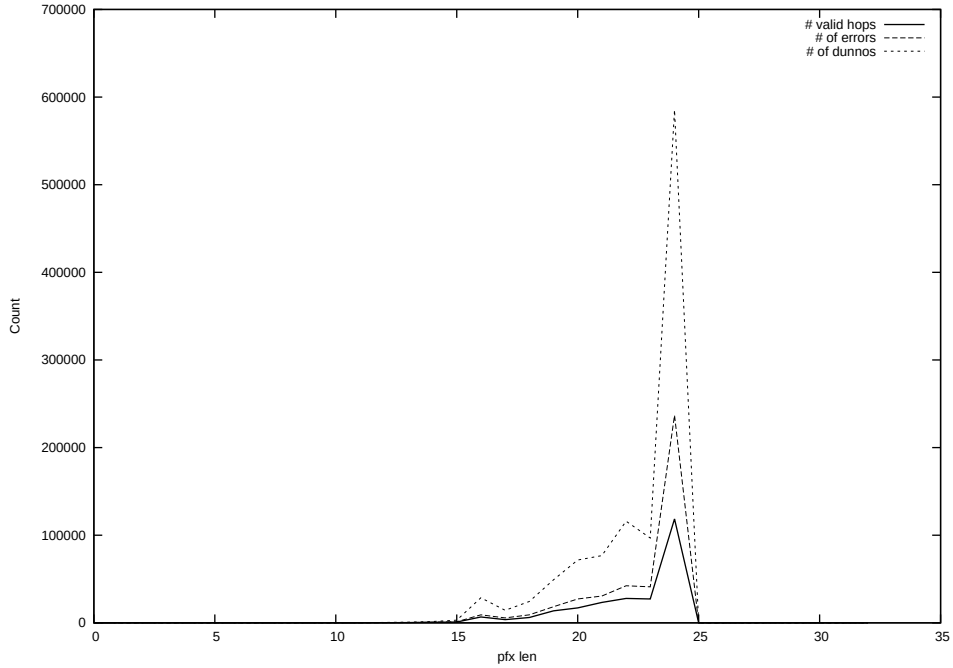
2013-08-05



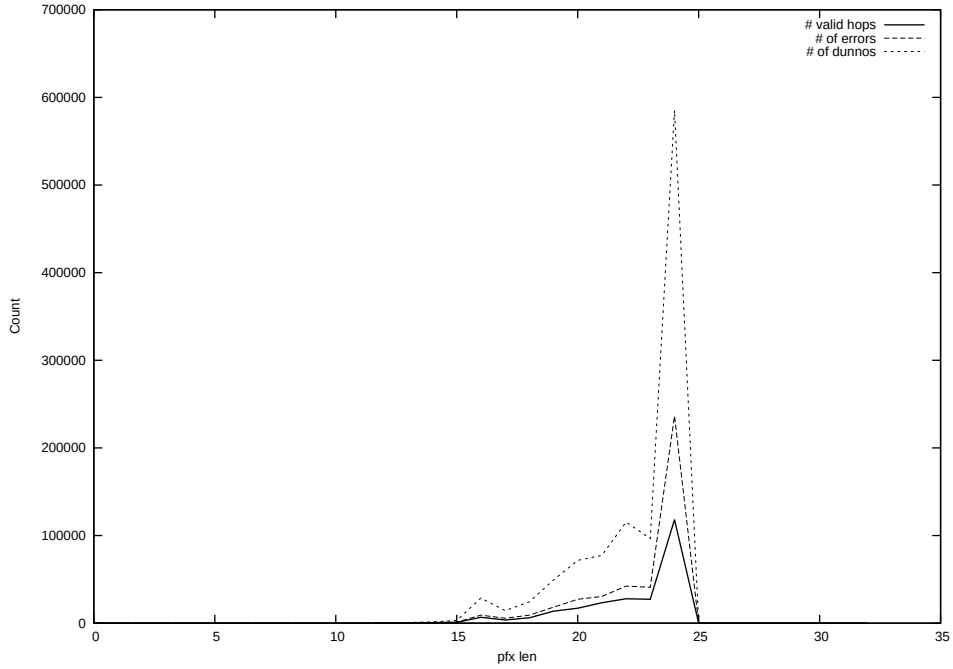
2013-08-06



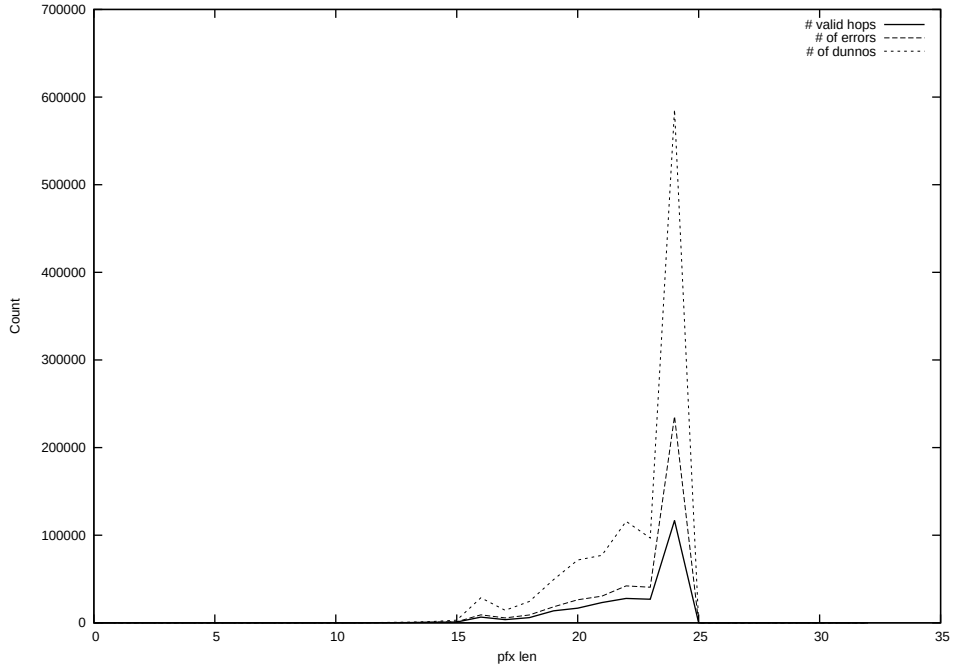
2013-08-07



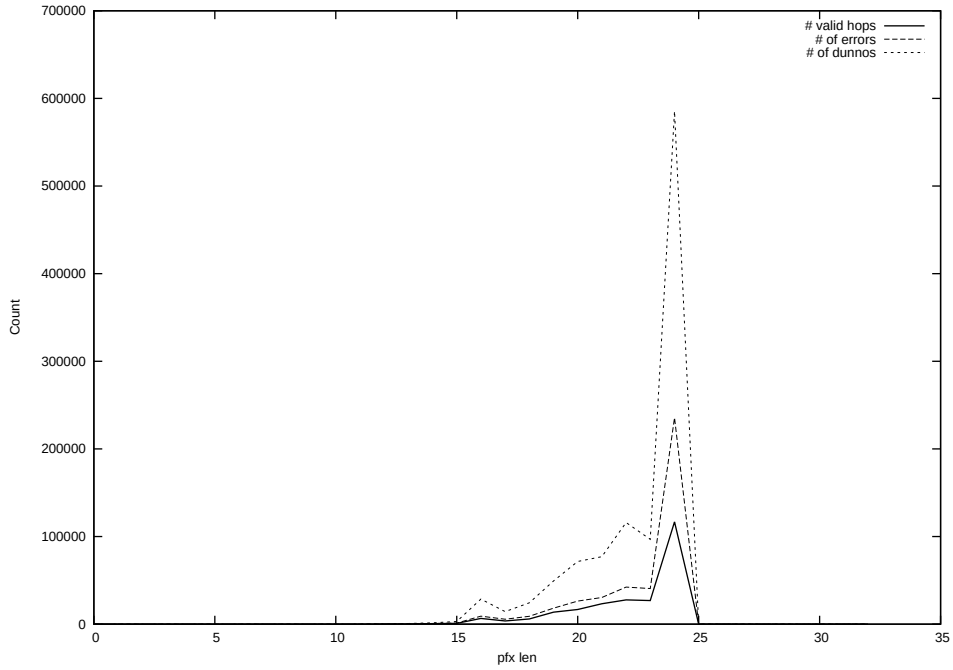
2013-08-08



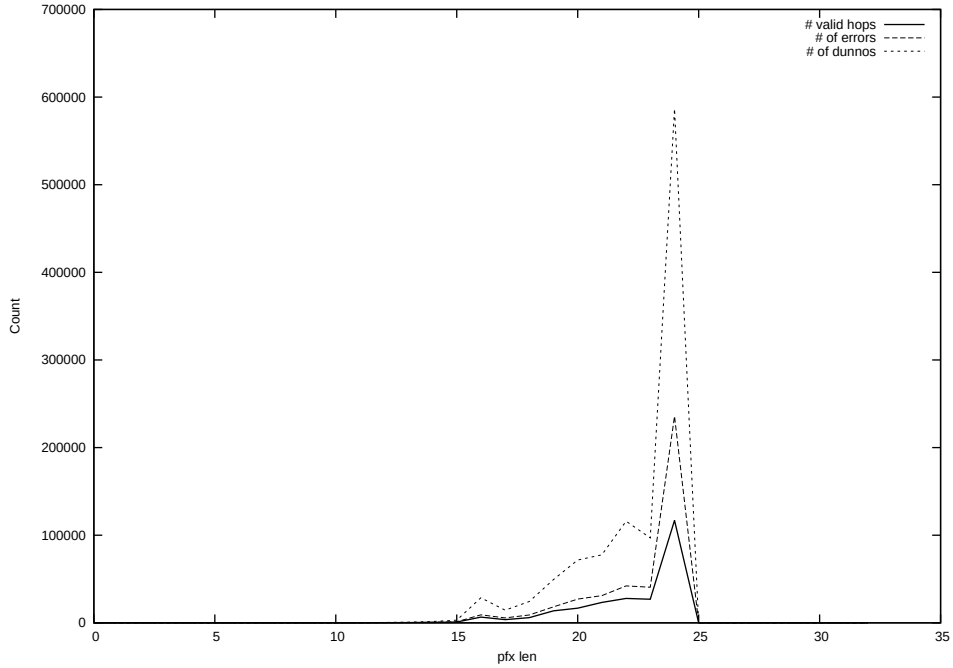
2013-08-09



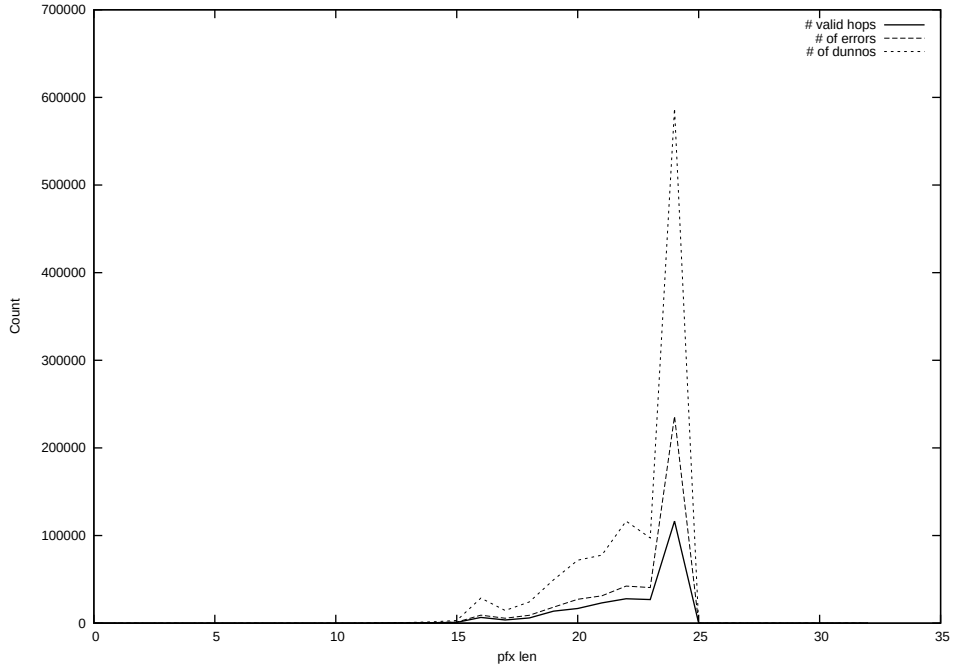
2013-08-10



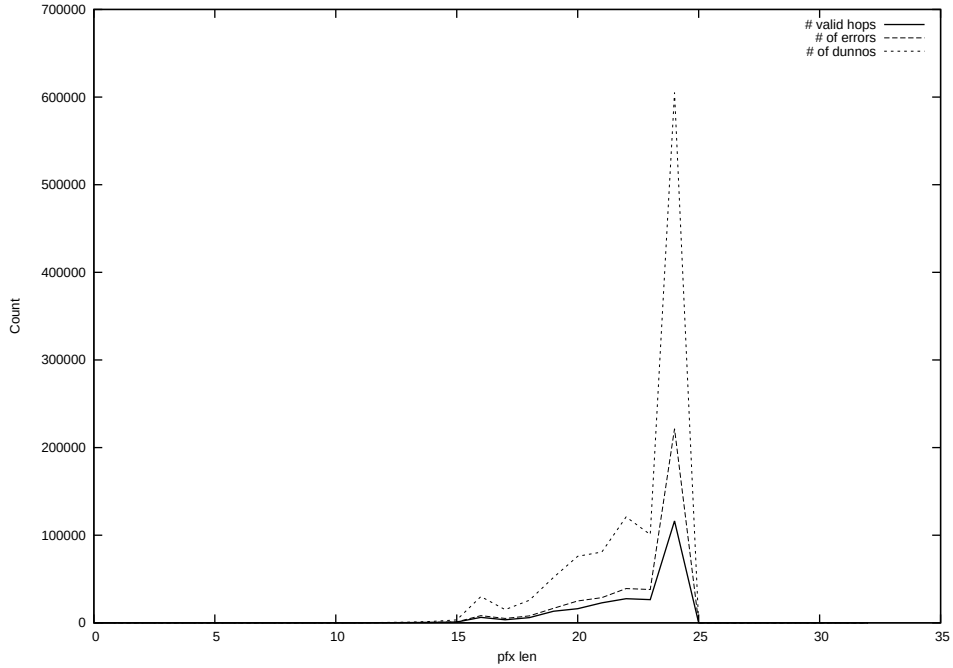
2013-08-11



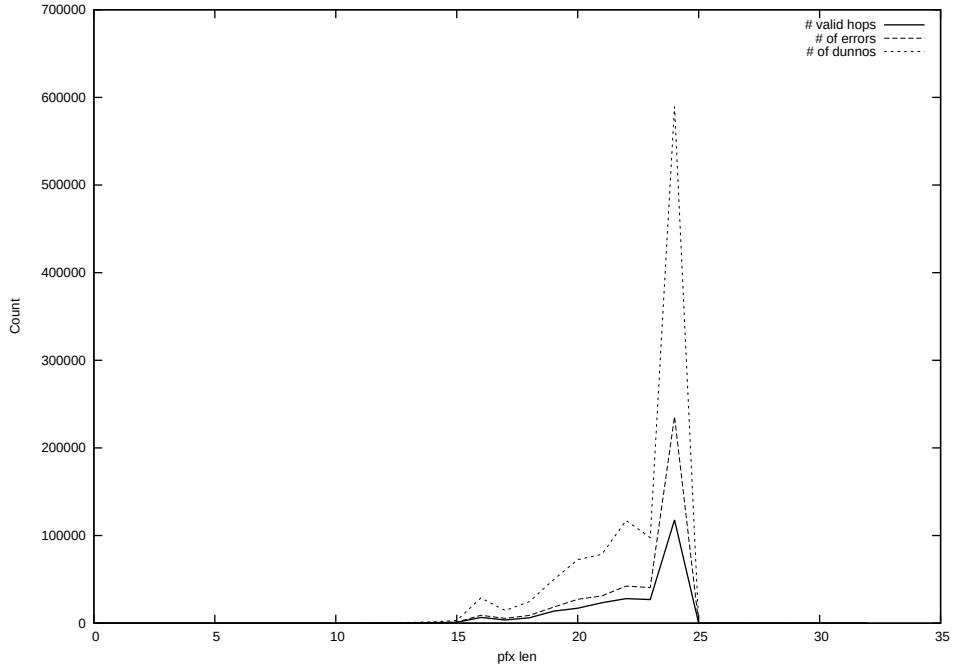
2013-08-12



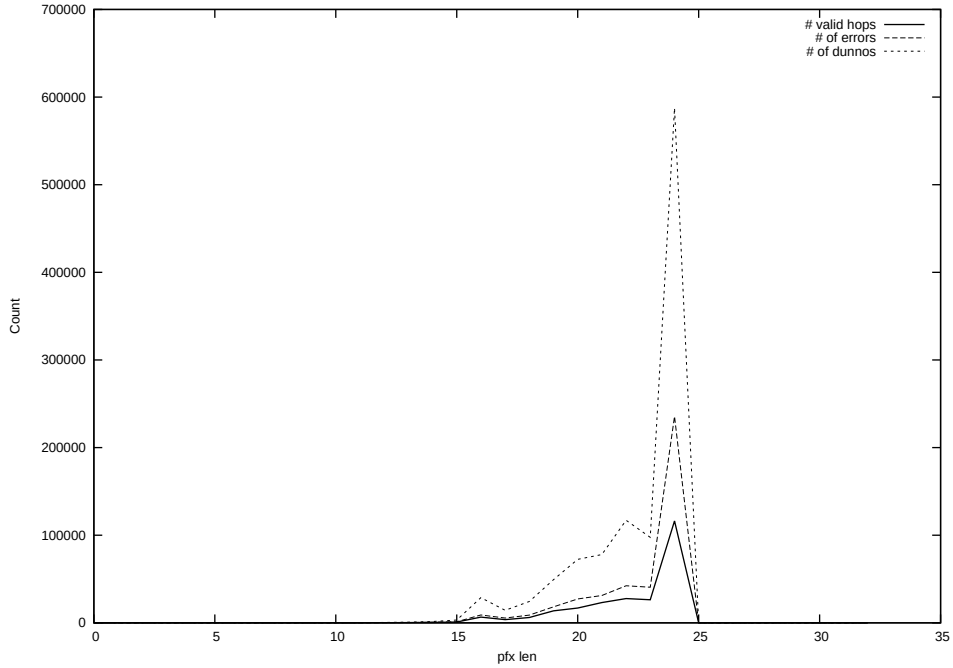
2013-08-13



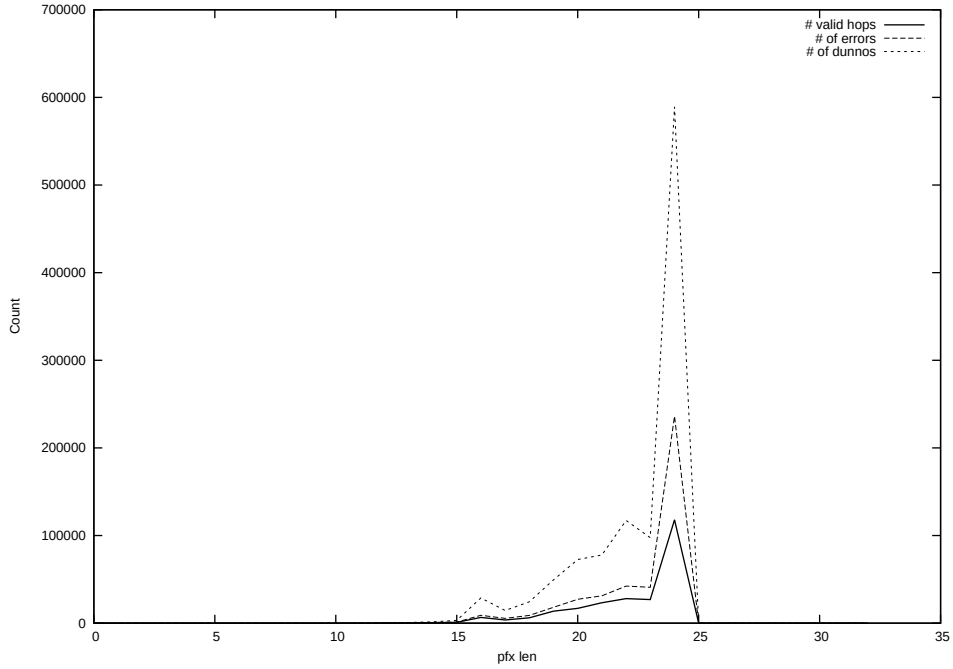
2013-08-15



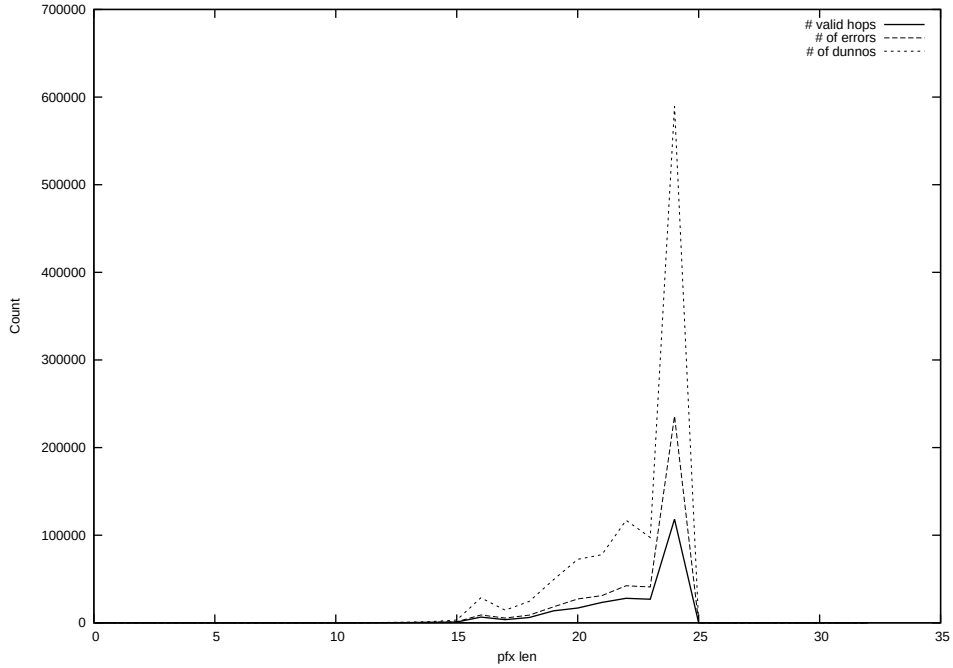
2013-08-16



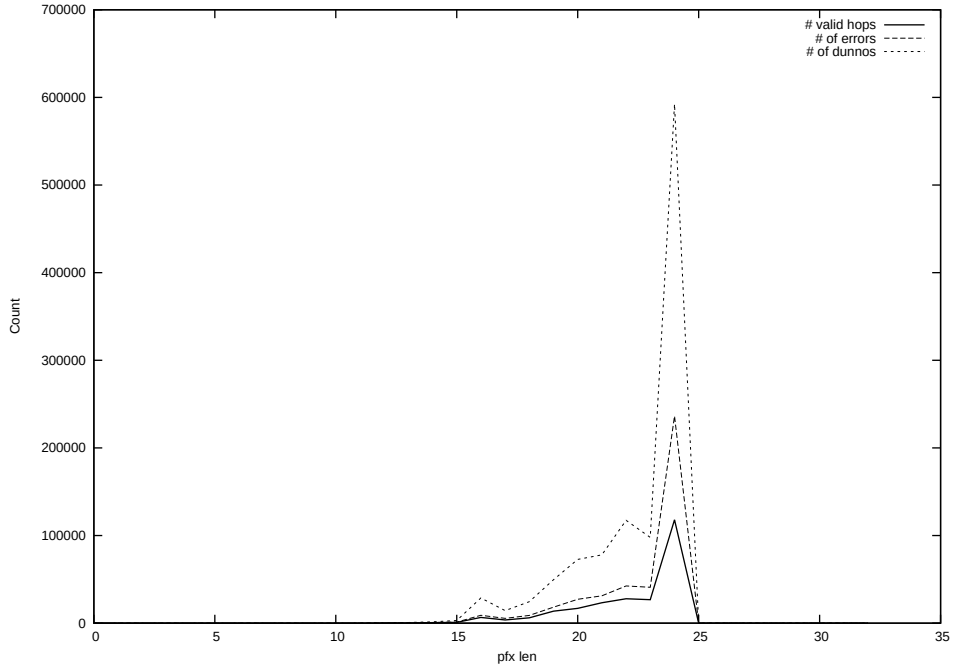
2013-08-17



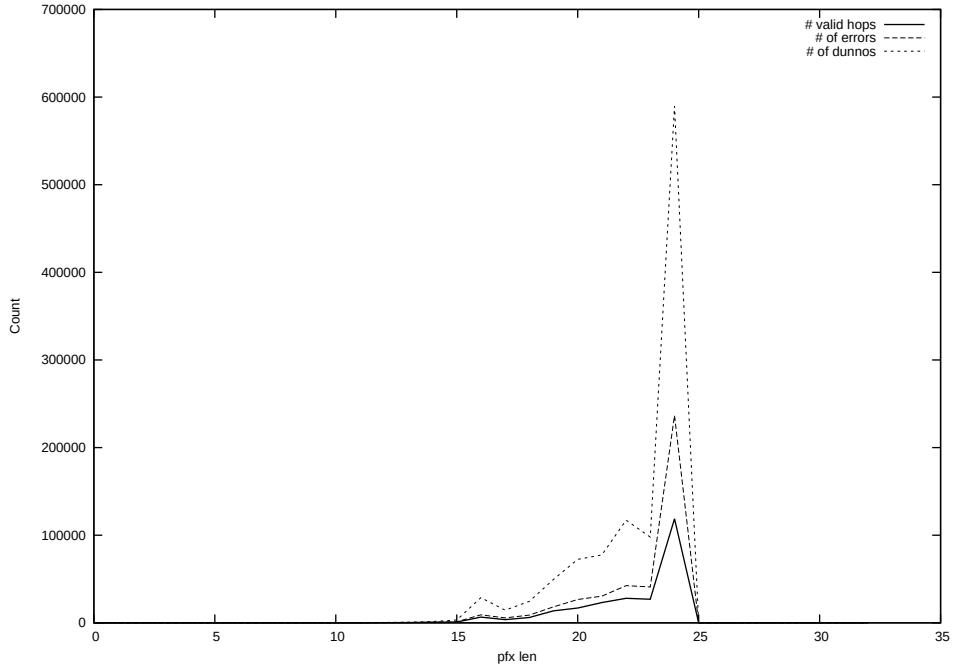
2013-08-18



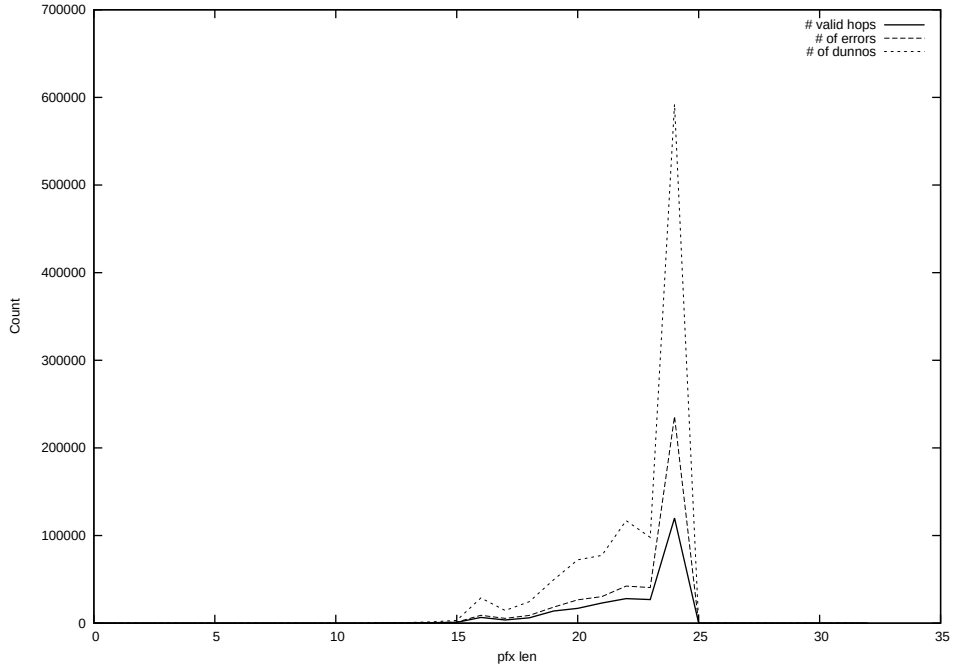
2013-08-19



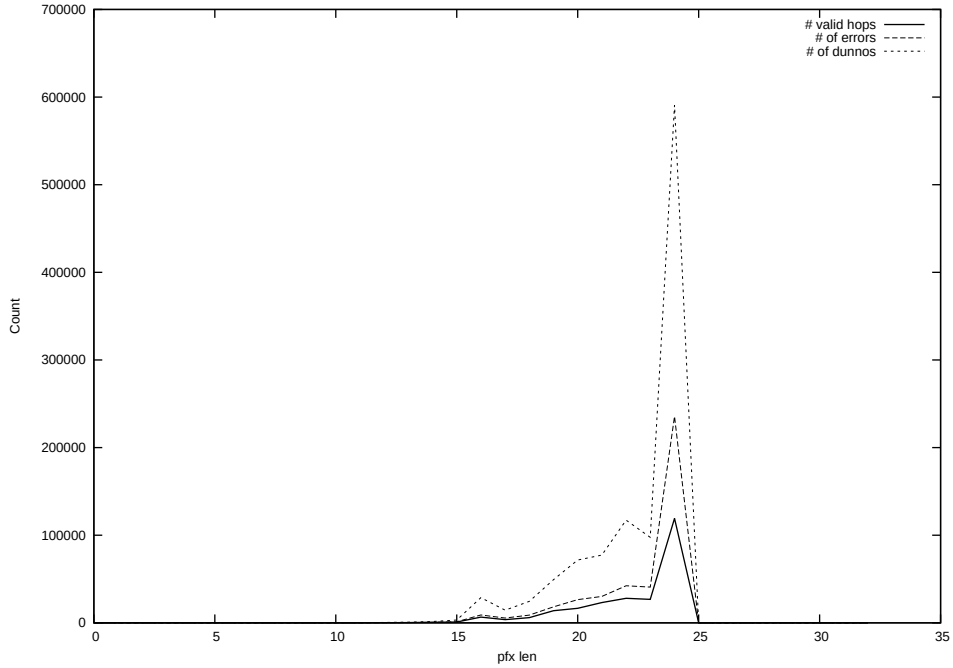
2013-08-20



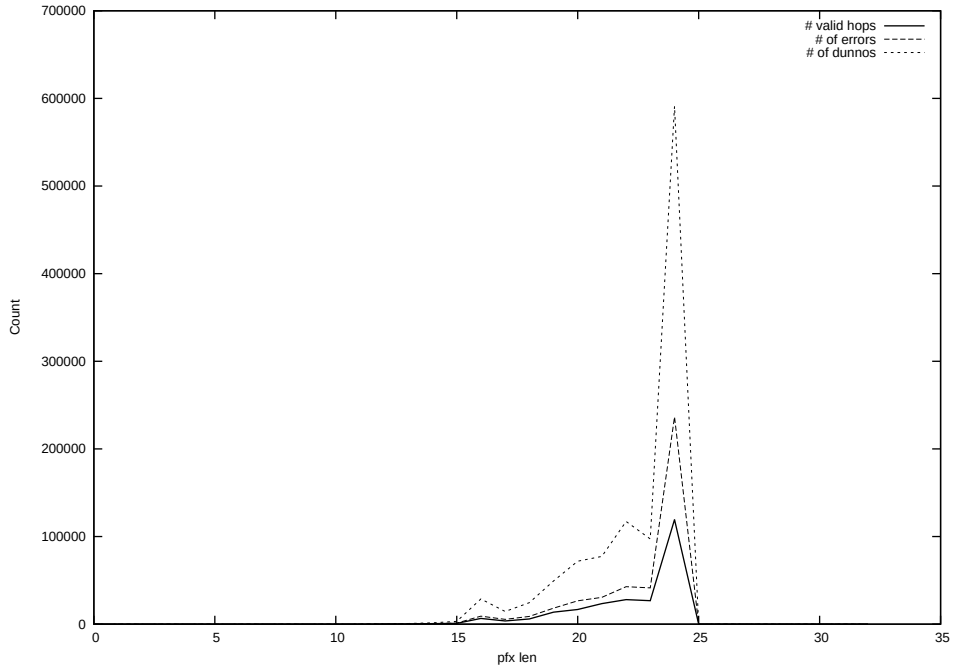
2013-08-21



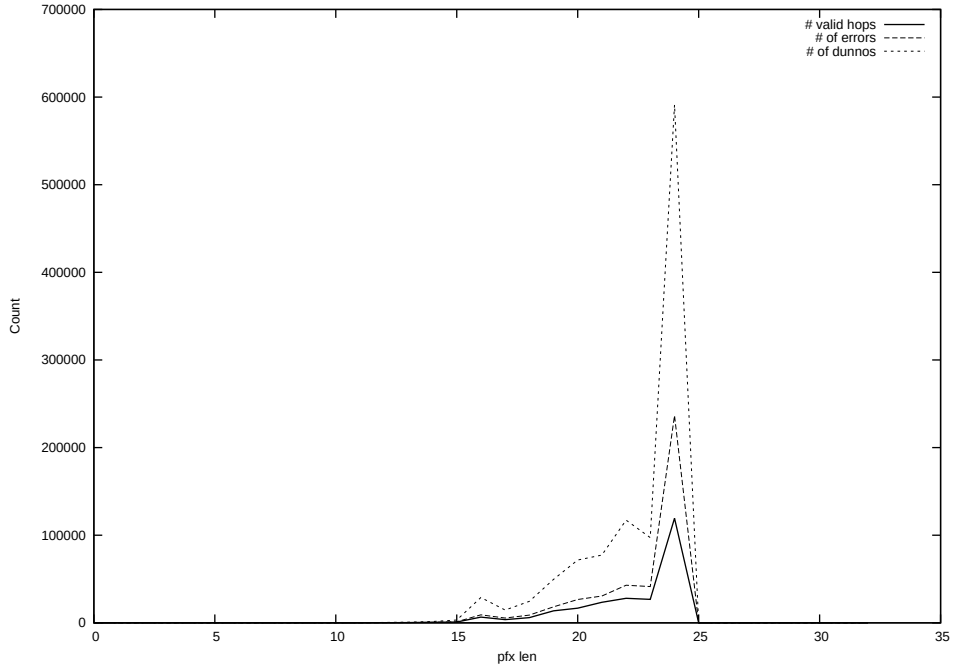
2013-08-22



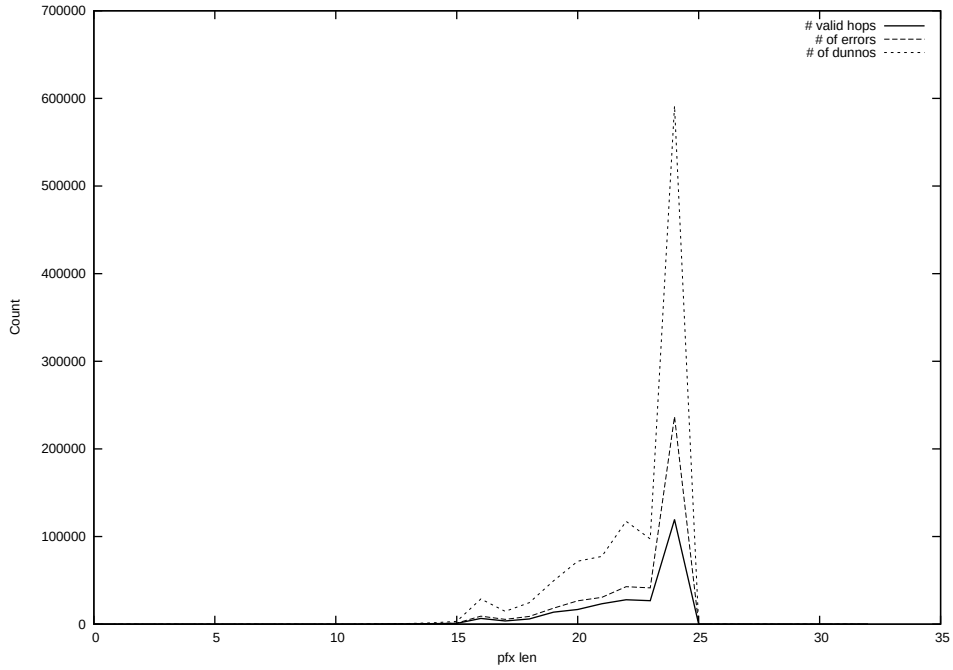
2013-08-23



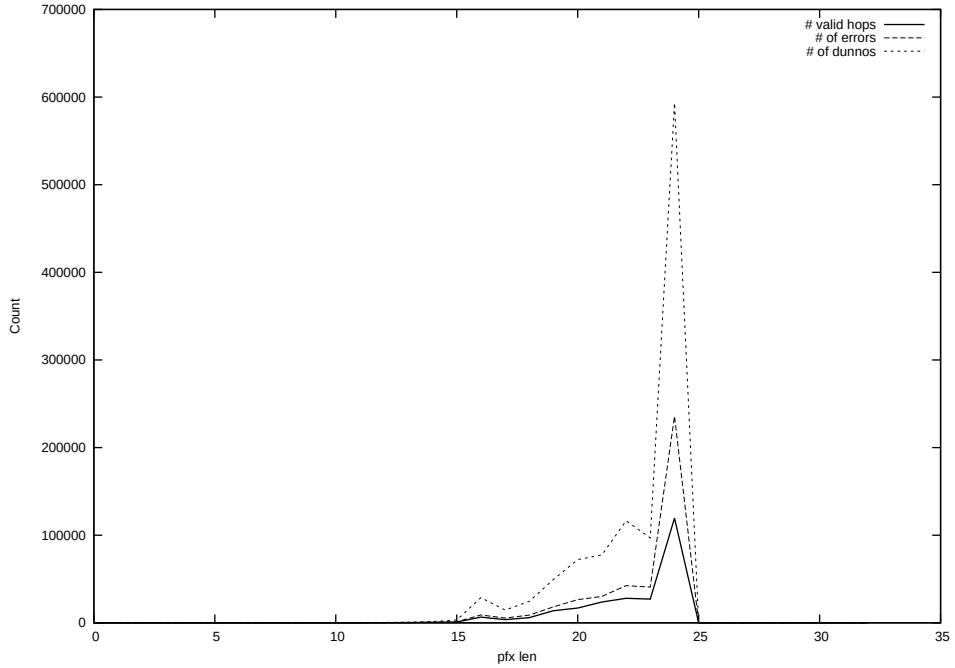
2013-08-24



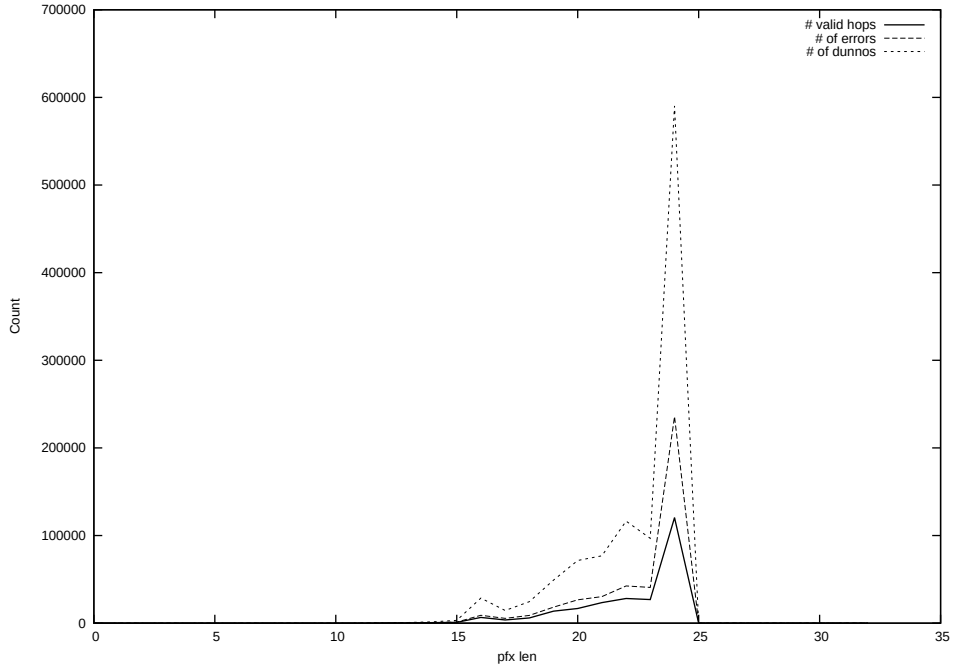
2013-08-25



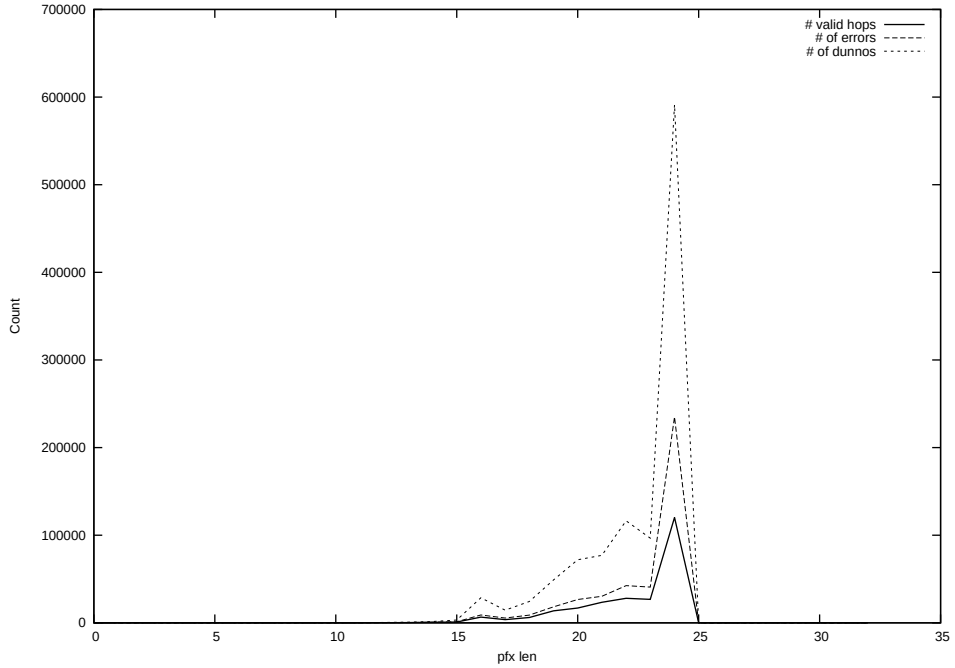
2013-08-26



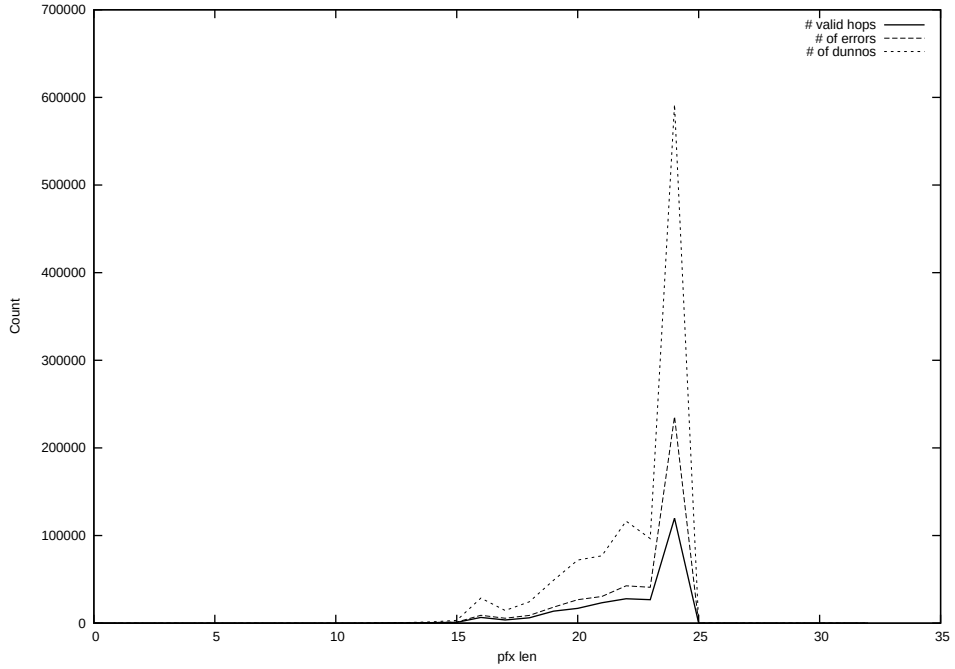
2013-08-27



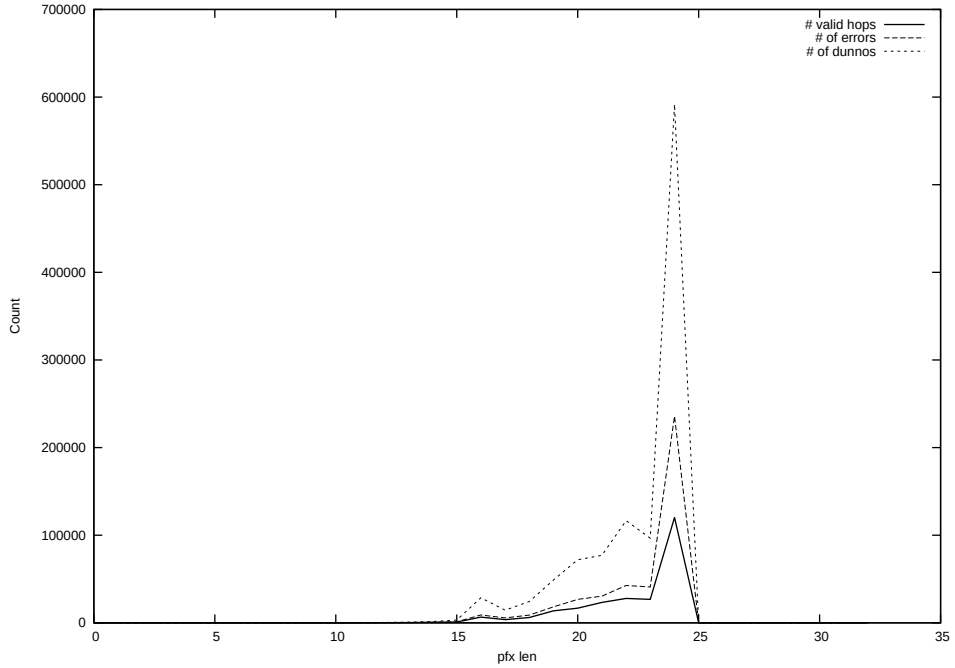
2013-08-28



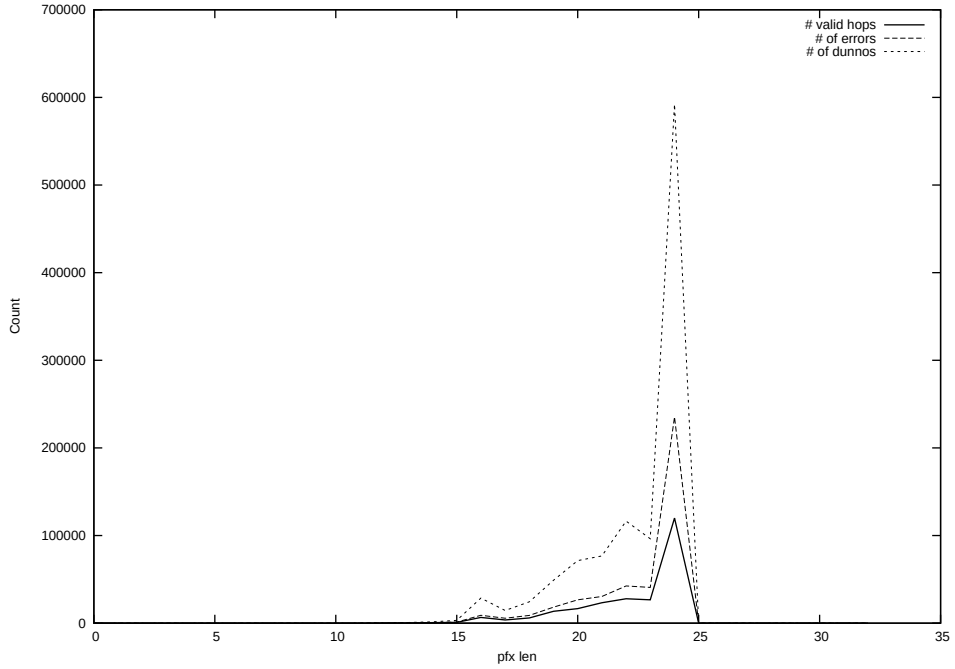
2013-08-29



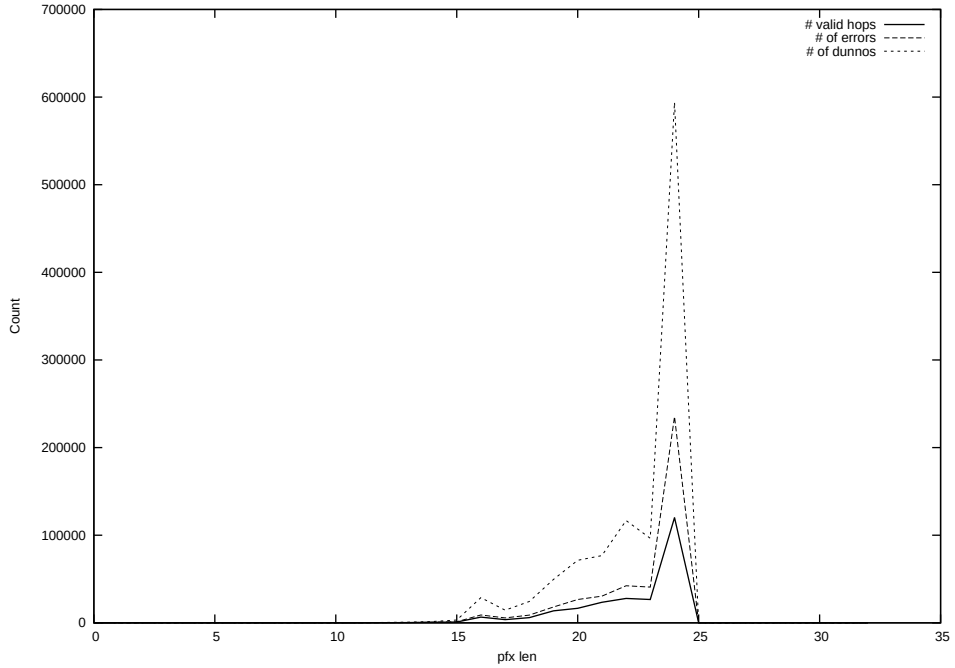
2013-08-30



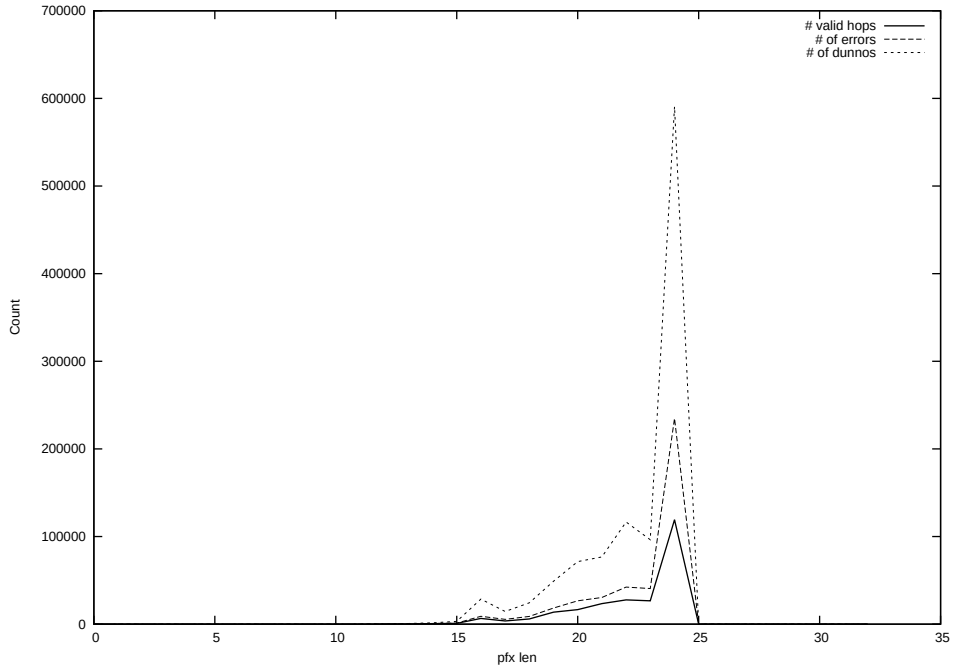
2013-08-31



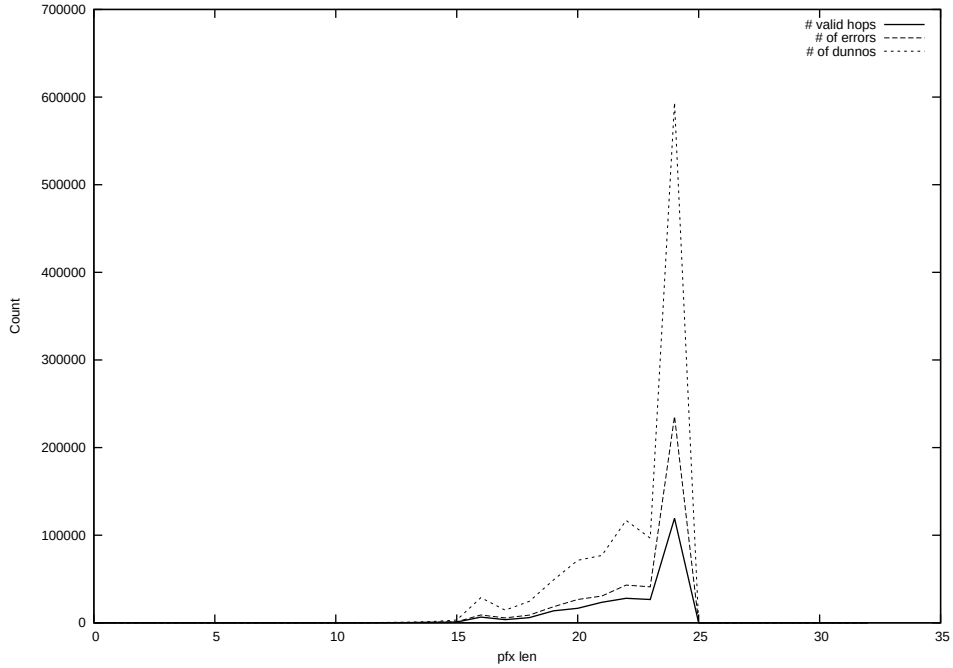
2013-09-01



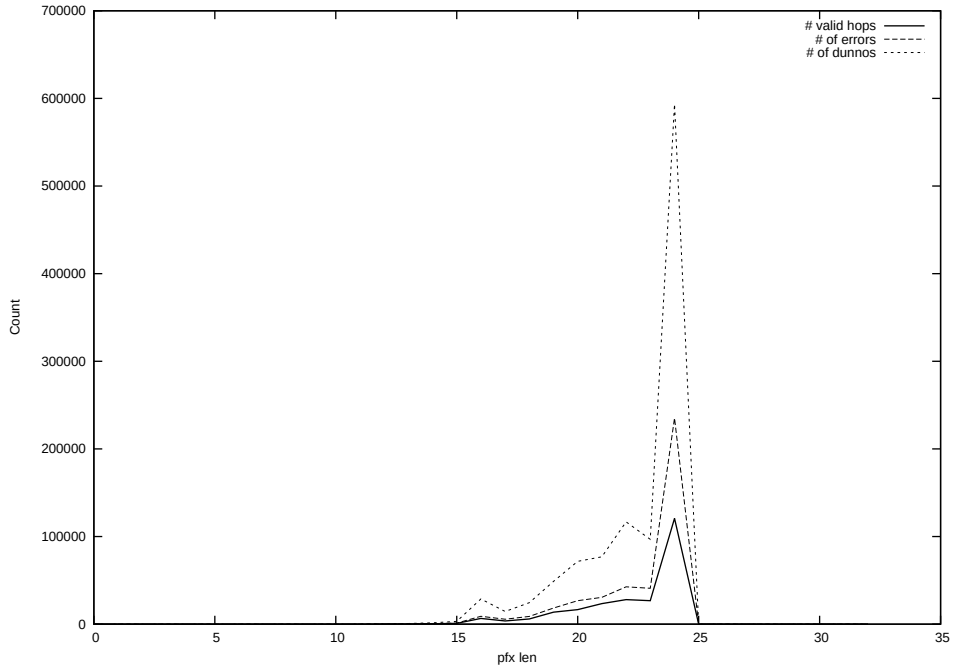
2013-09-02



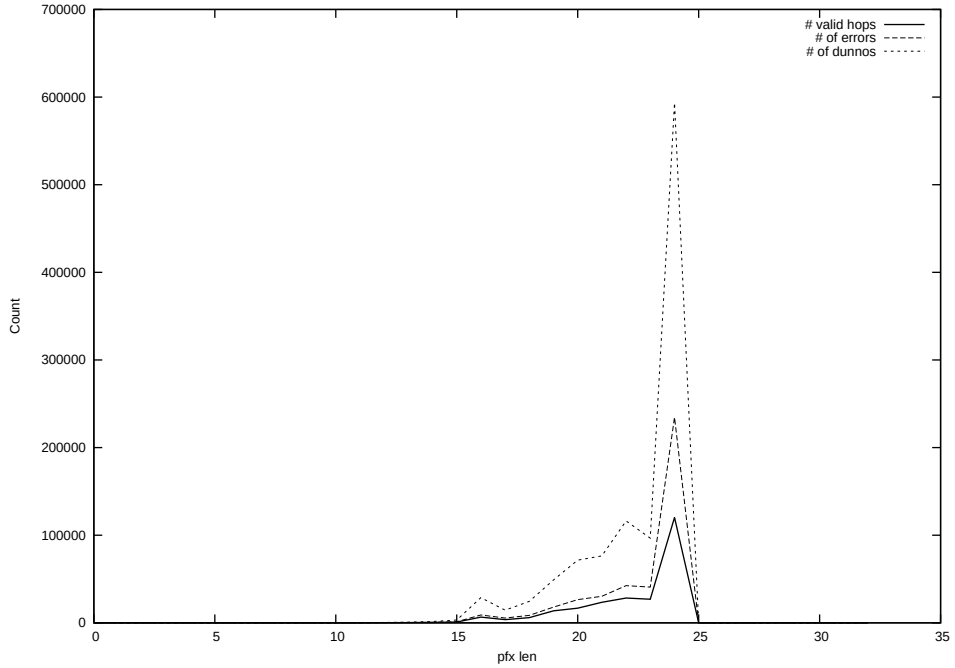
2013-09-03



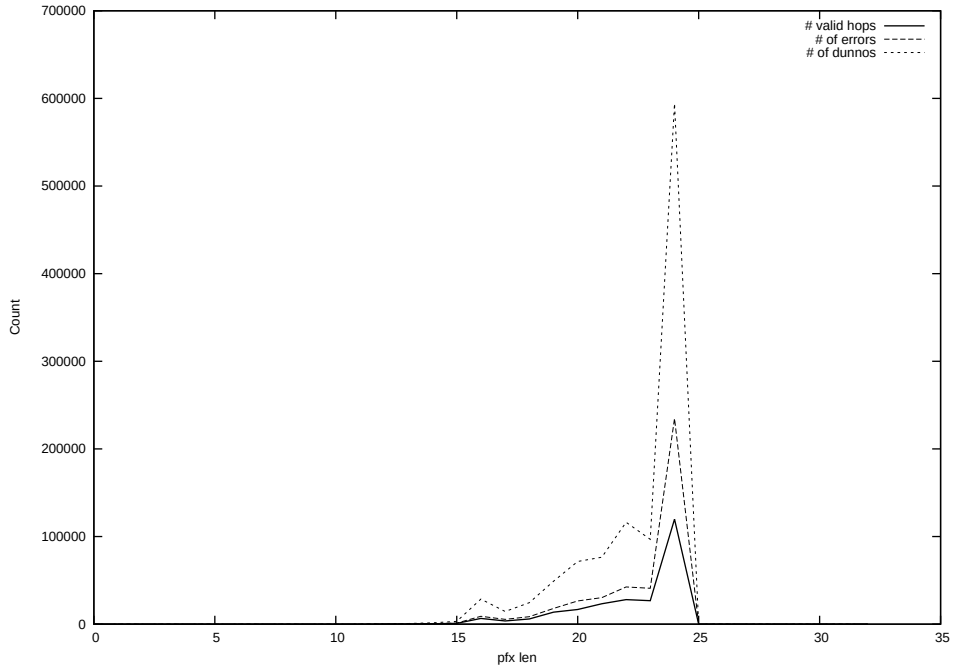
2013-09-04



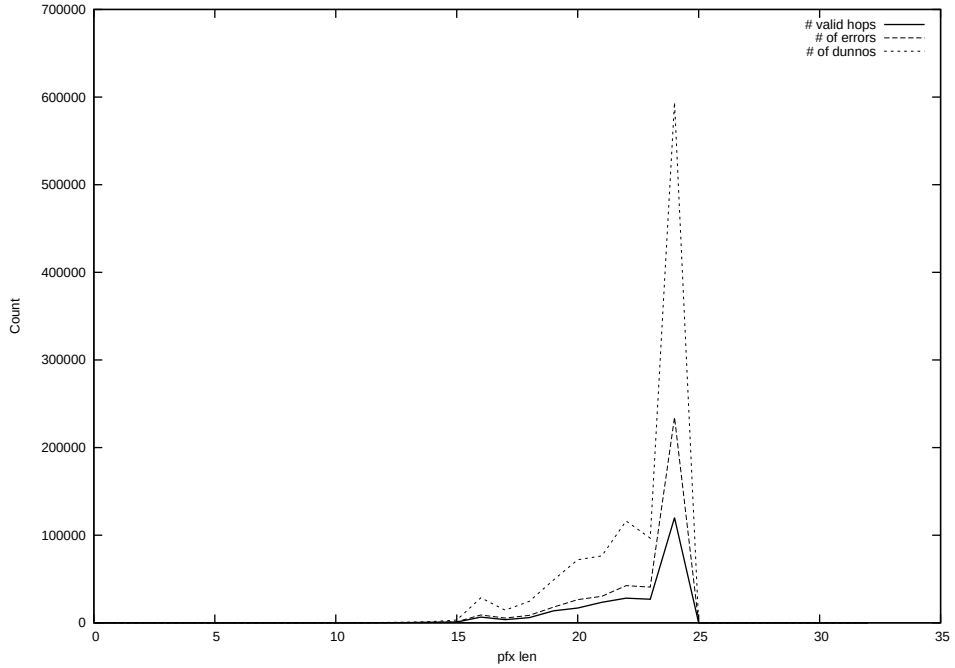
2013-09-05



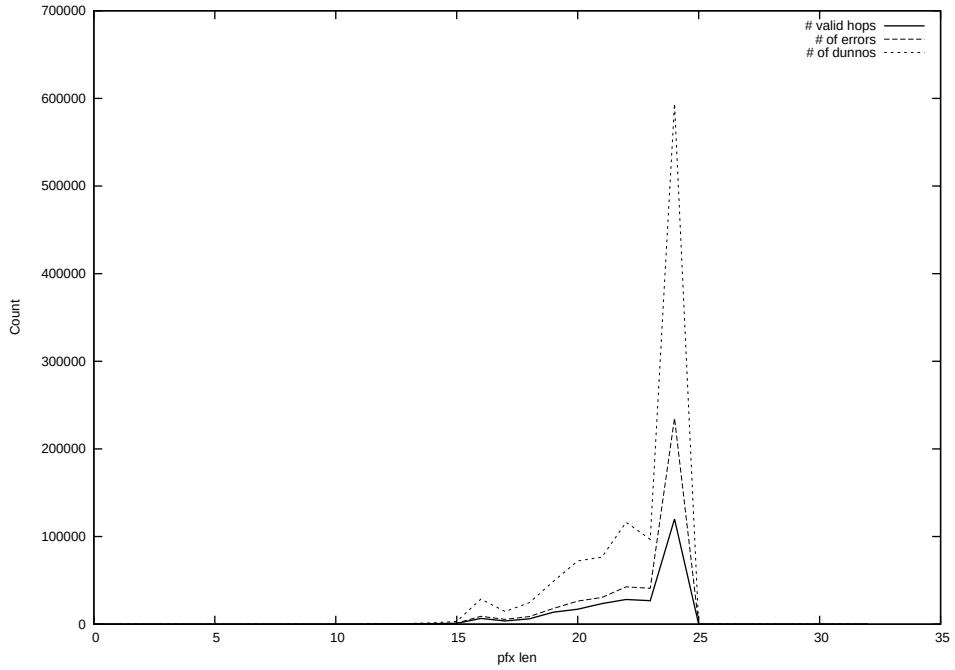
2013-09-06



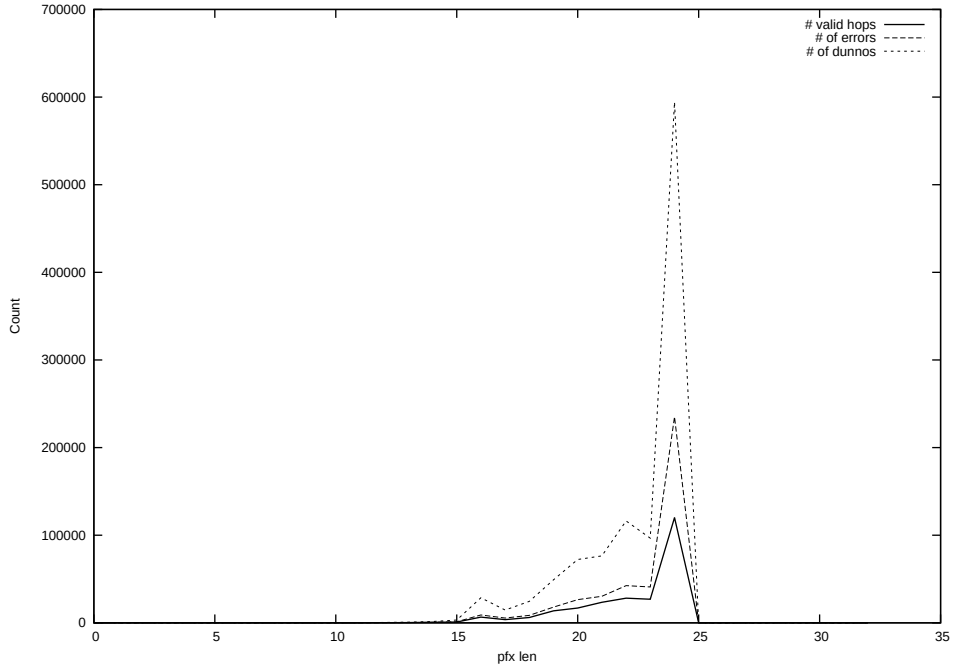
2013-09-07



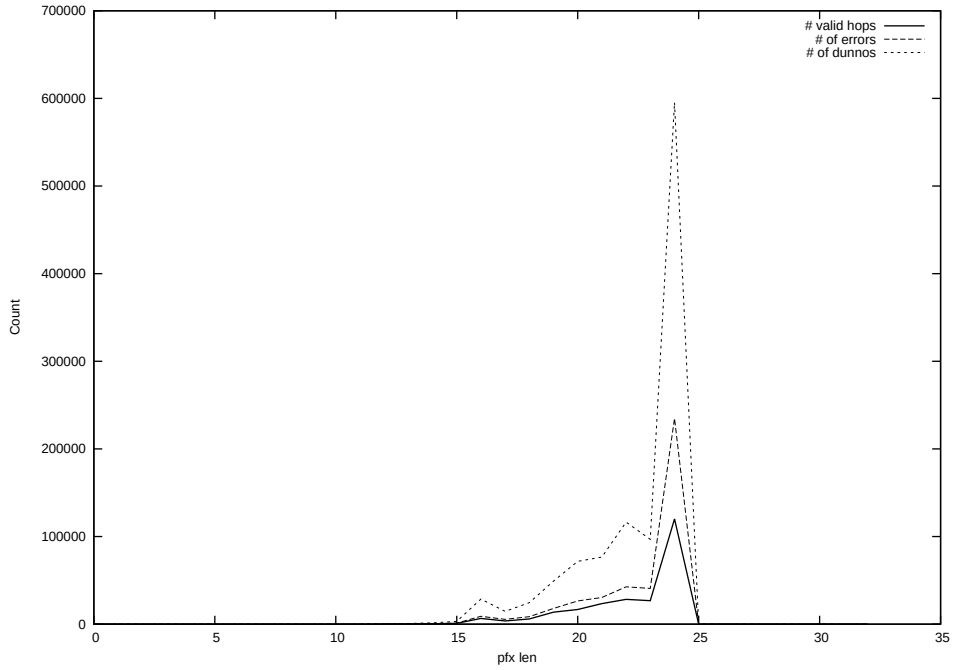
2013-09-08



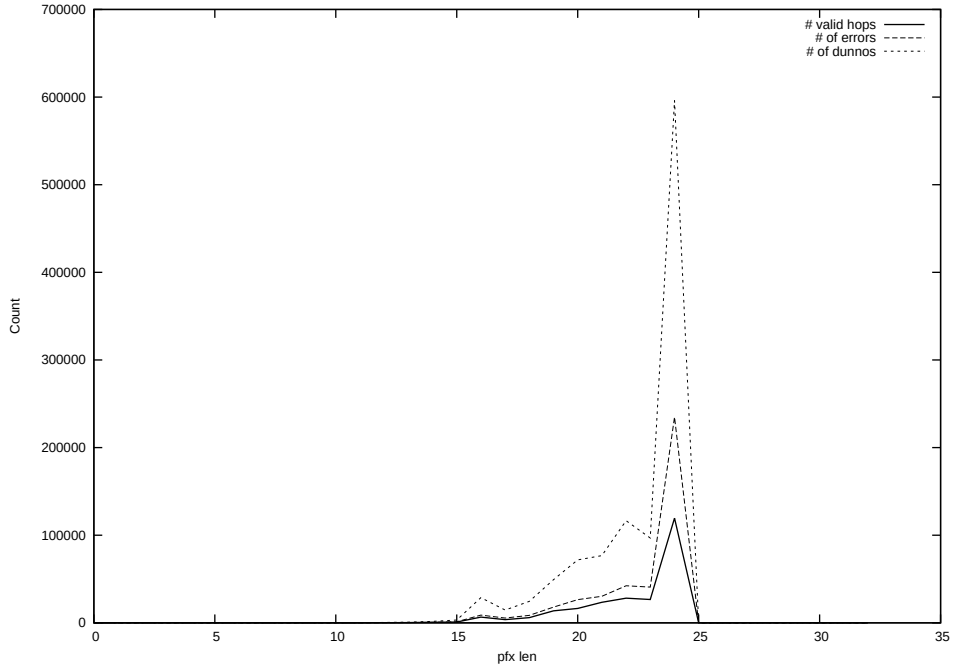
2013-09-09



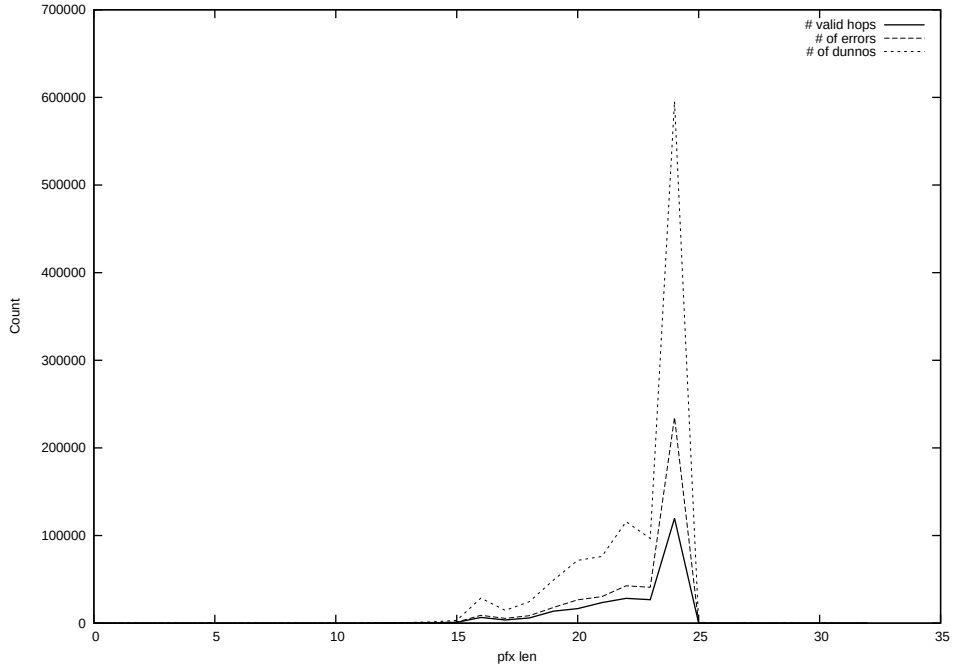
2013-09-10



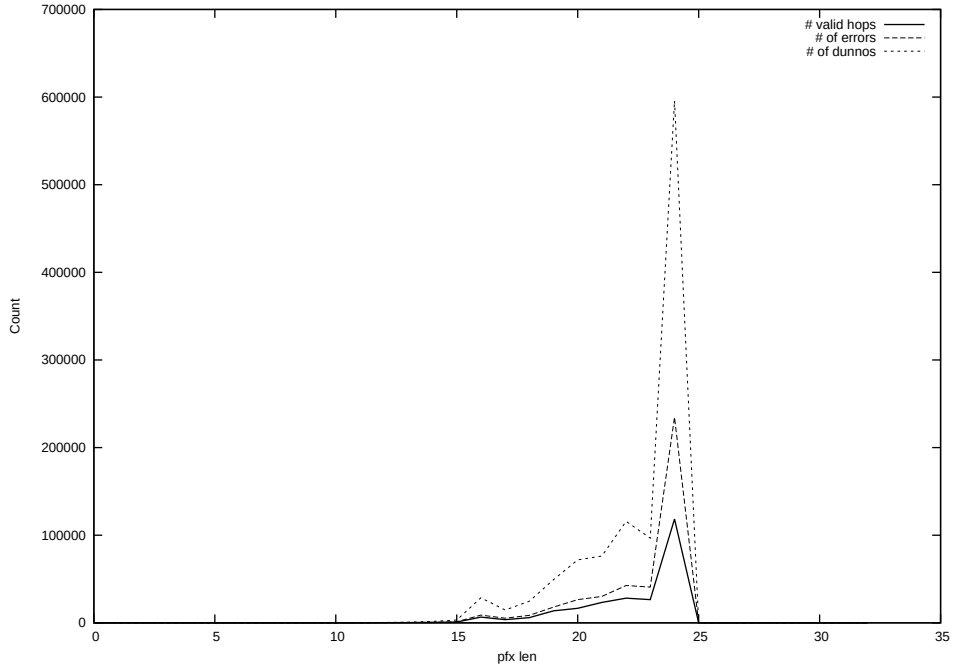
2013-09-11



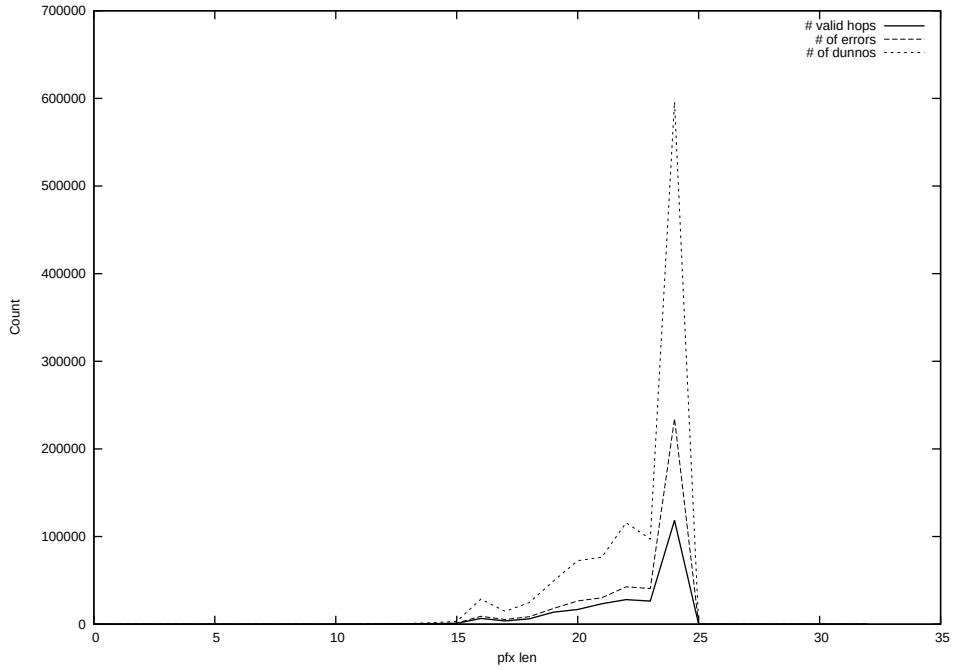
2013-09-12



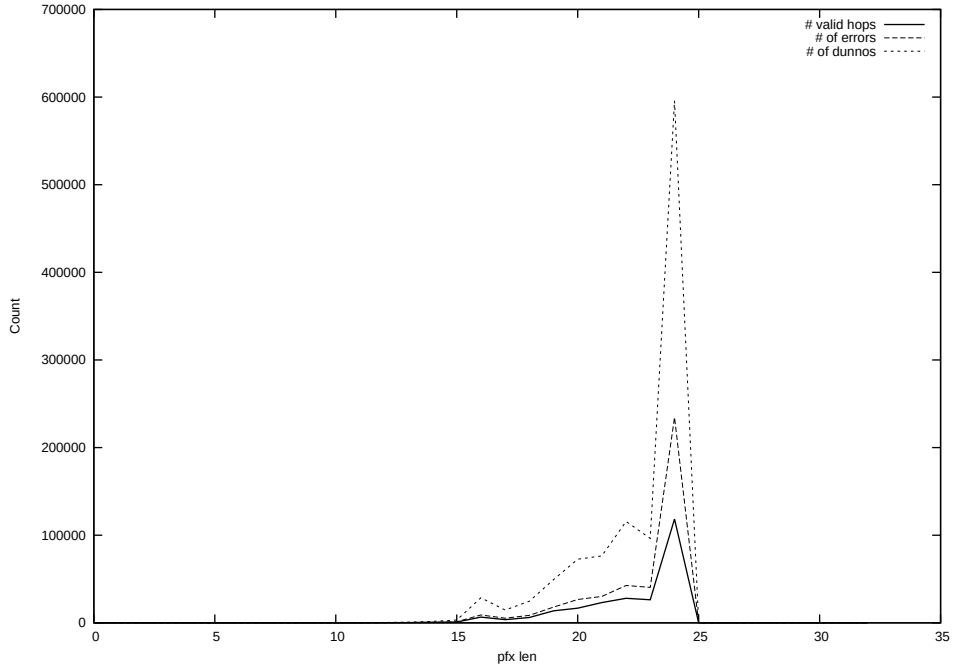
2013-09-13



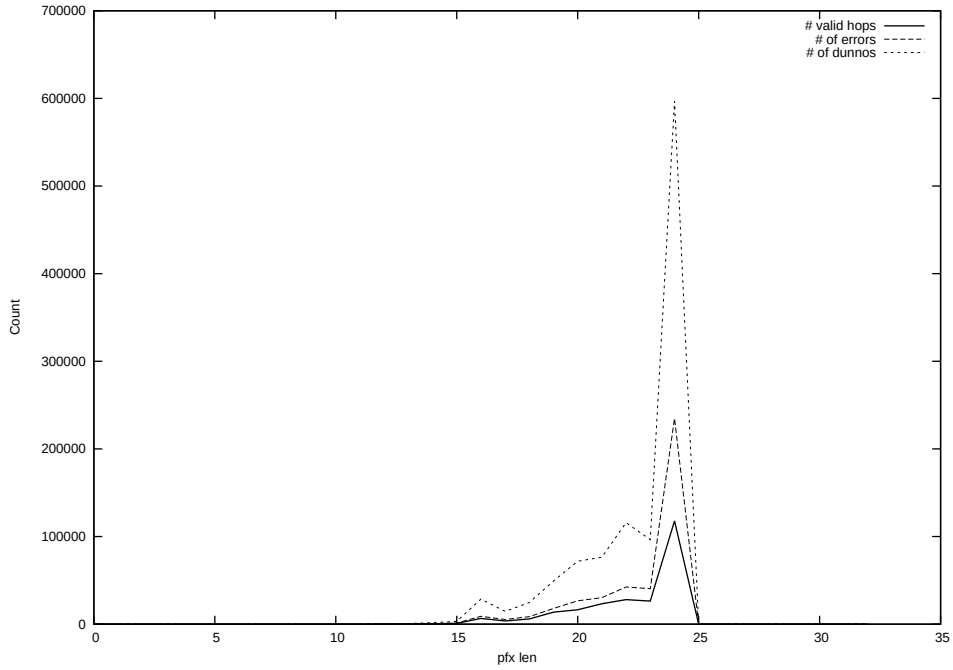
2013-09-14



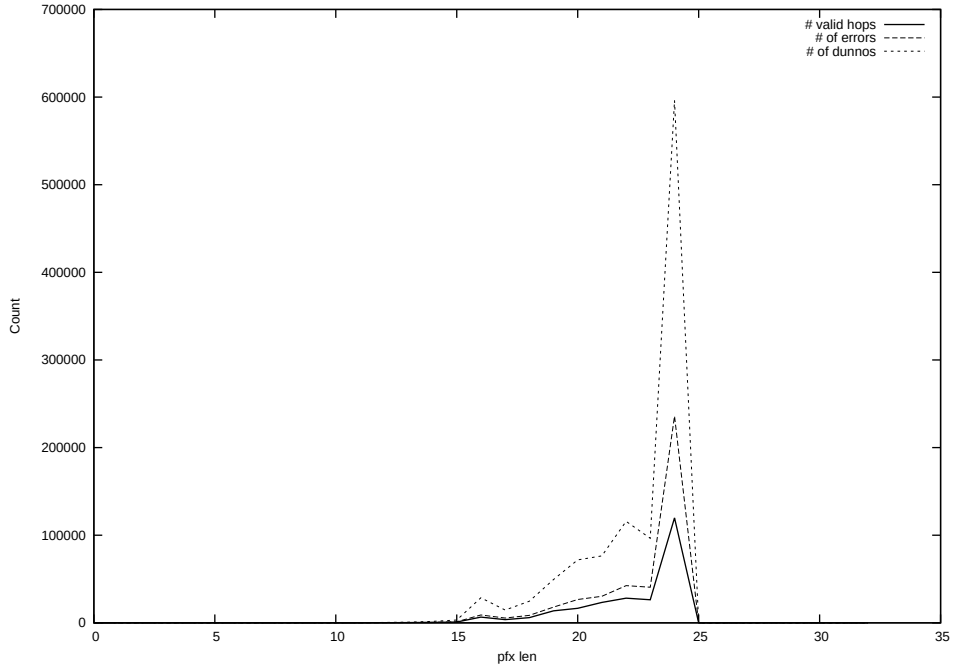
2013-09-15



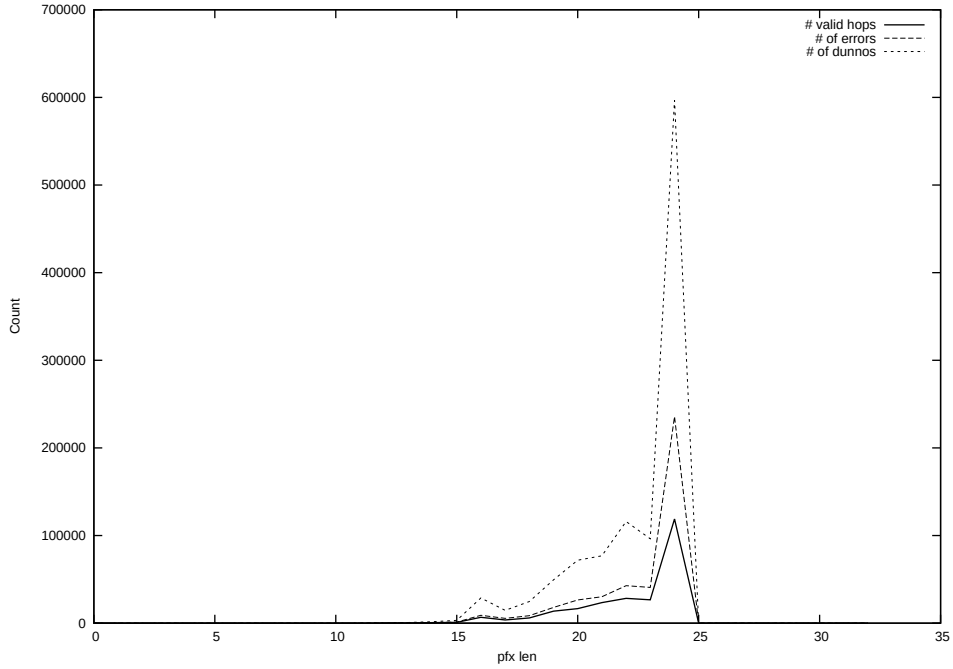
2013-09-16



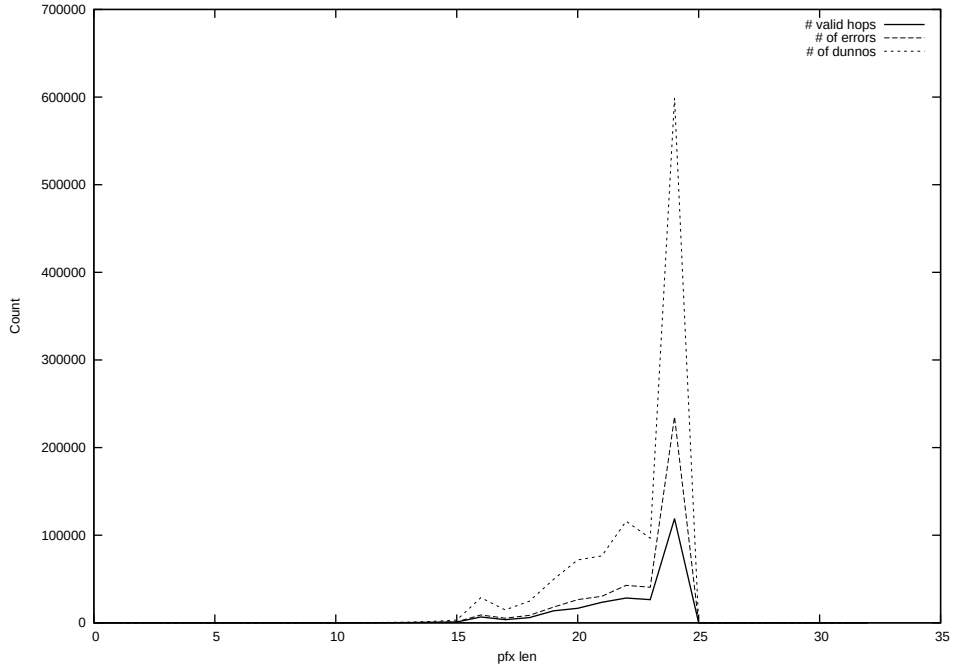
2013-09-17



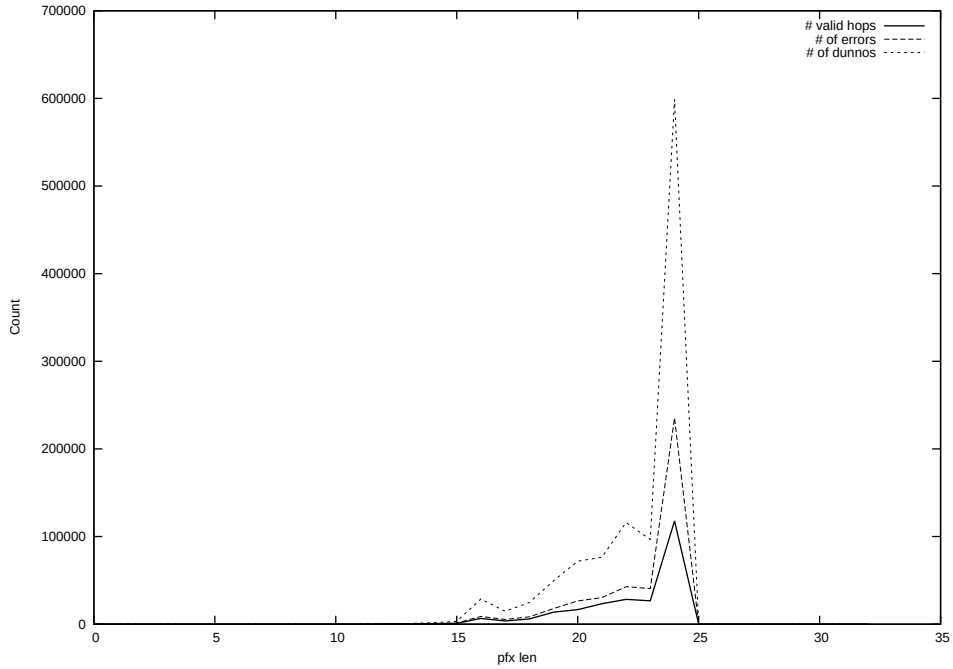
2013-09-18



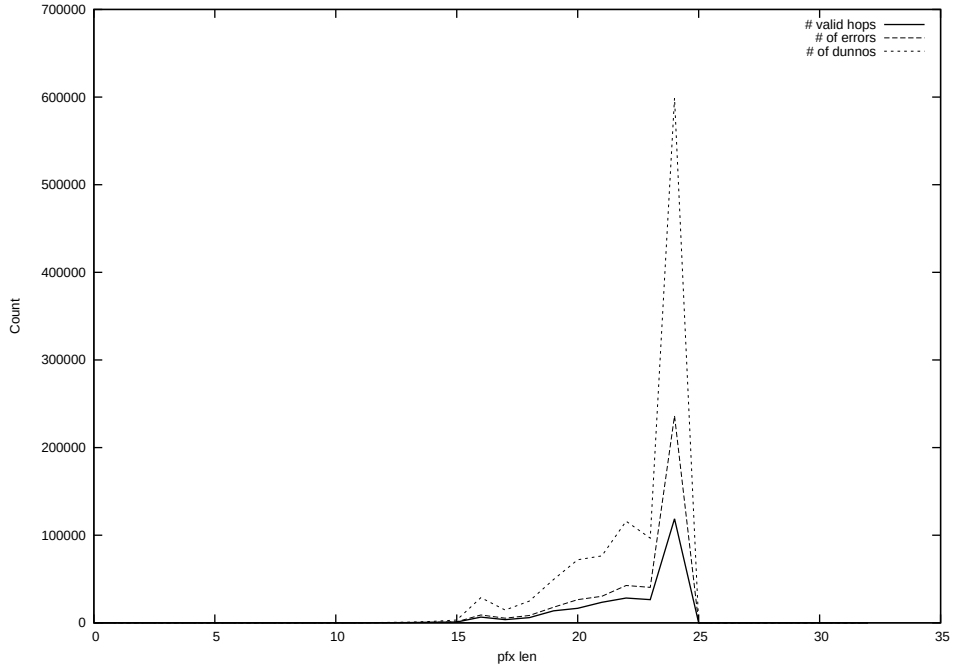
2013-09-19



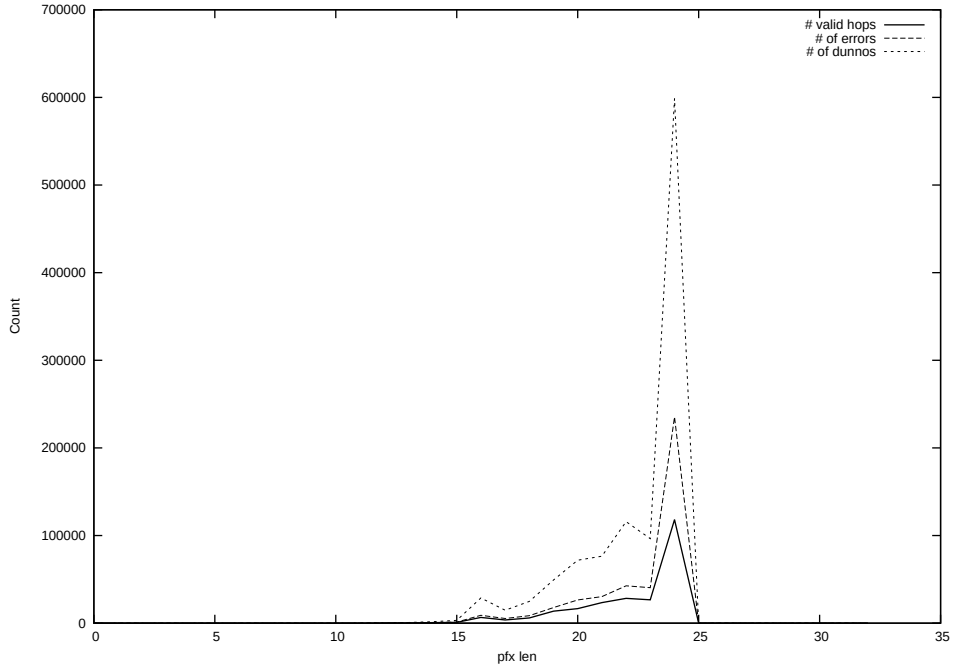
2013-09-20



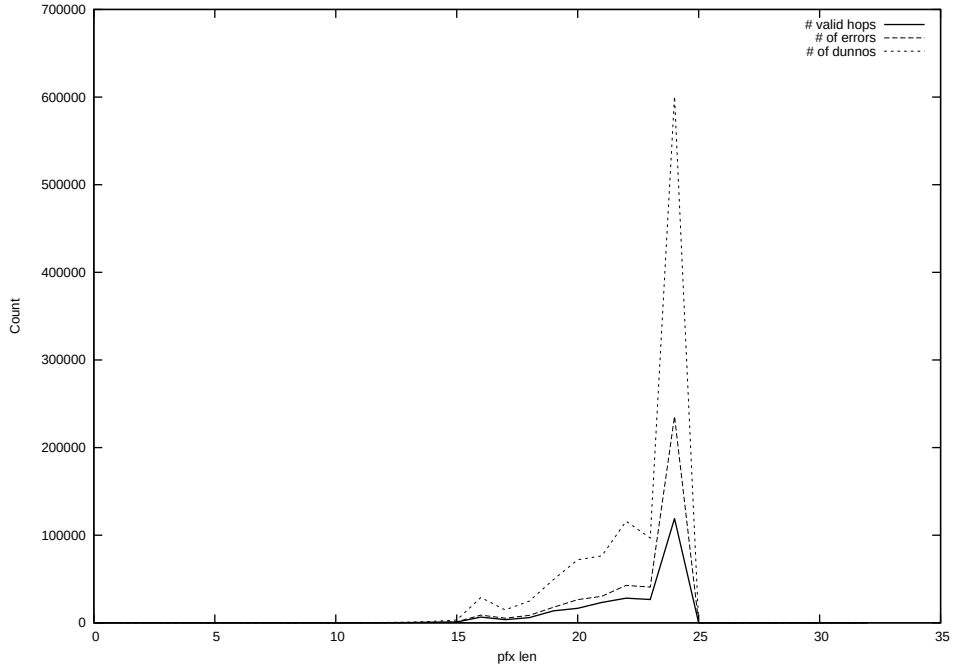
2013-09-21



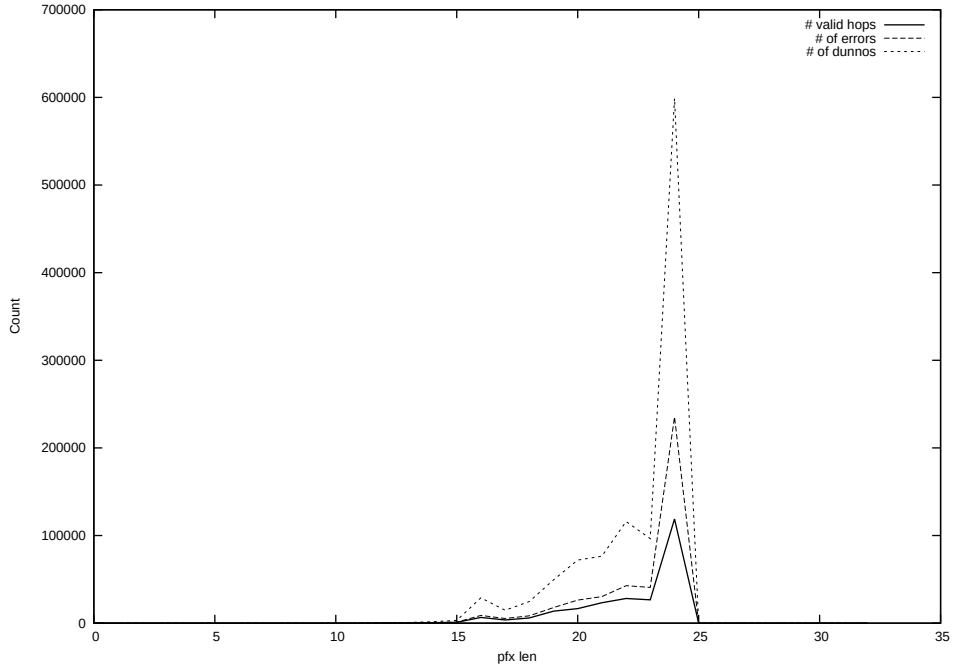
2013-09-22



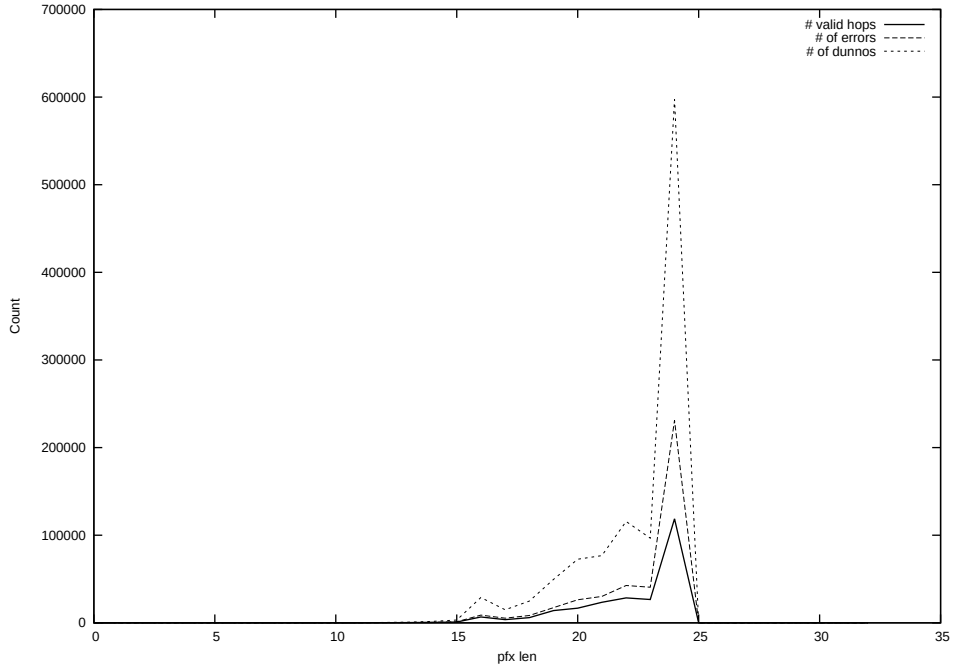
2013-09-23



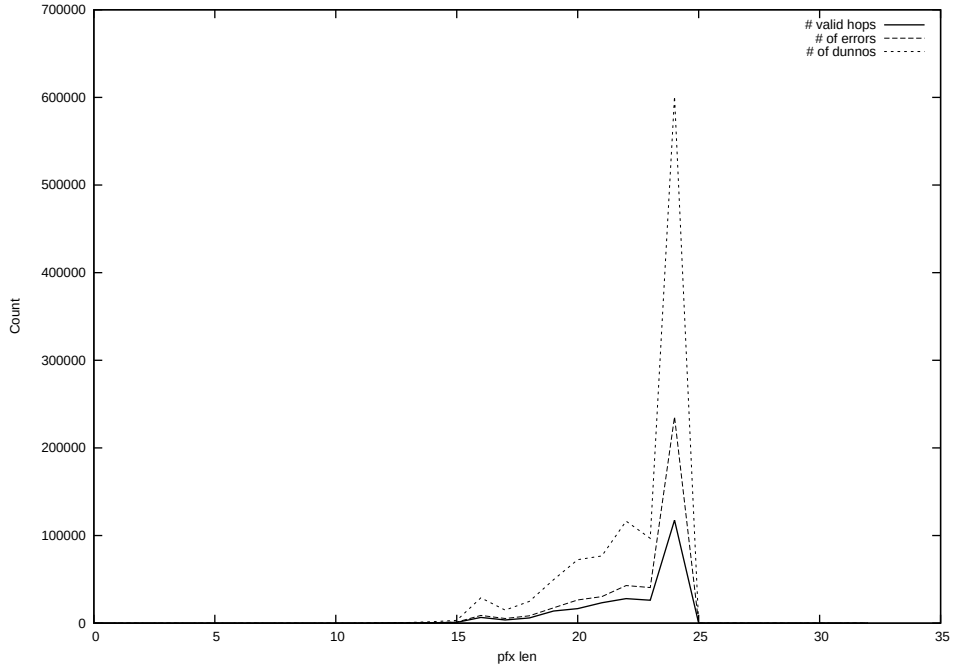
2013-09-24



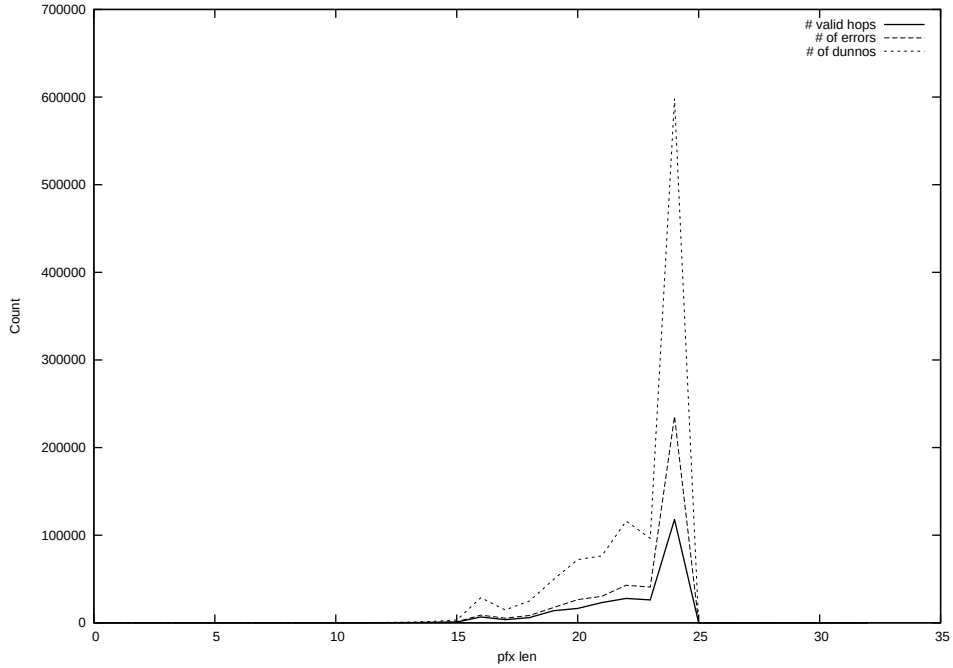
2013-09-25



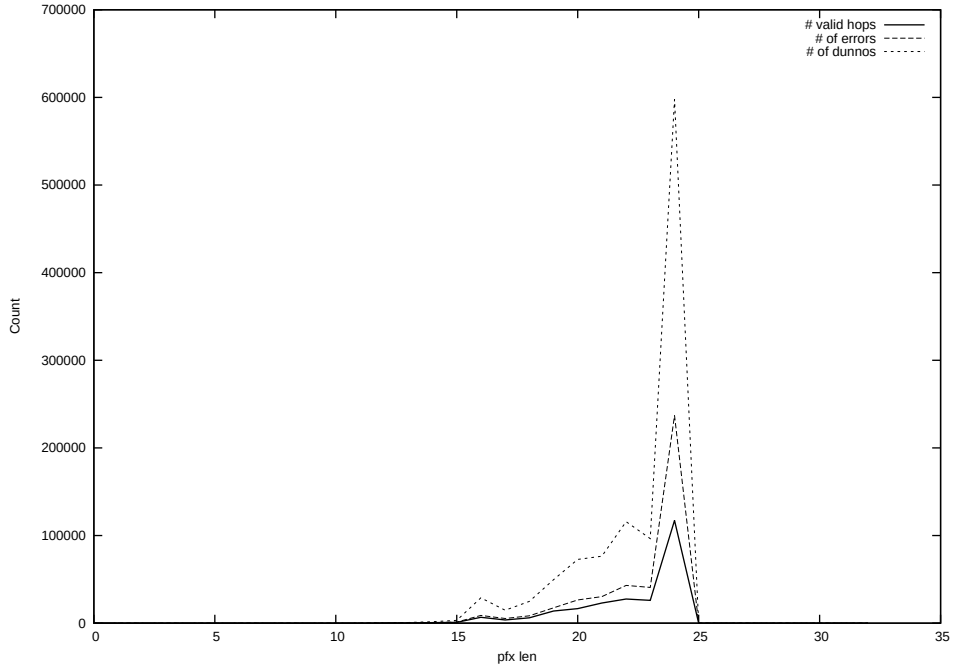
2013-09-26



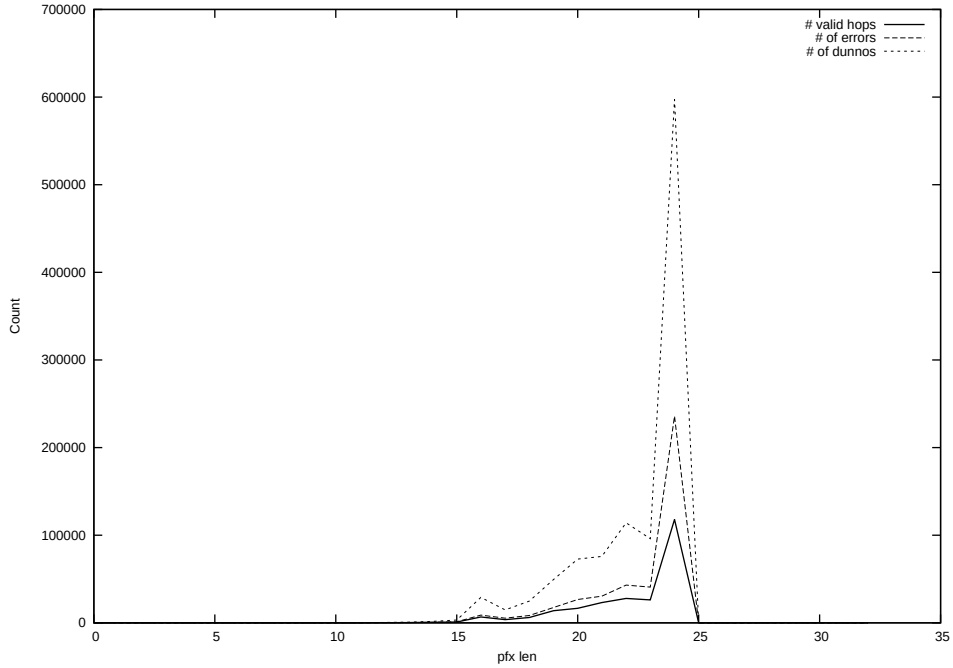
2013-09-27



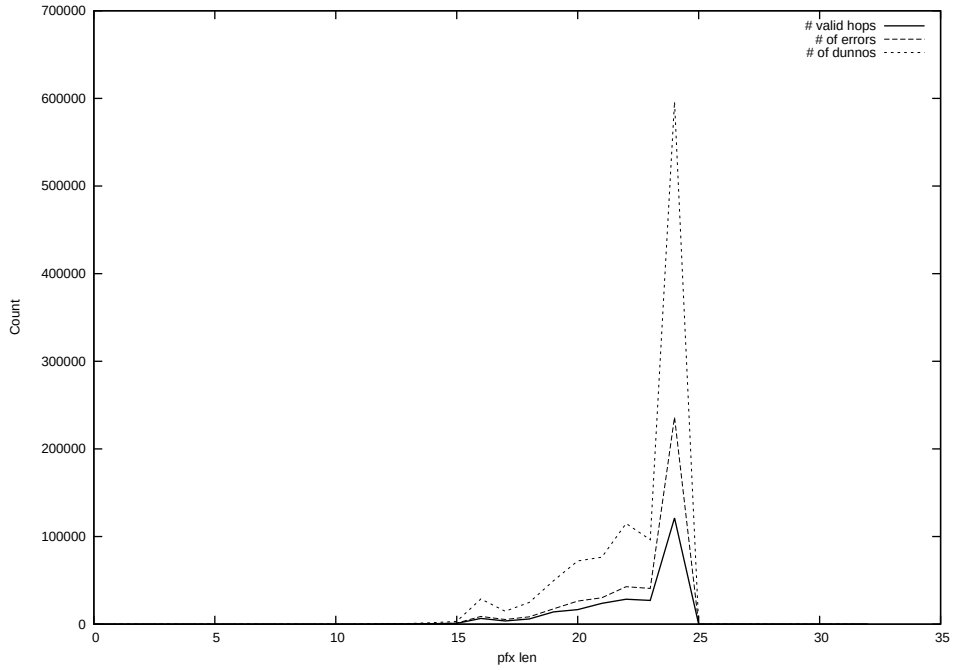
2013-09-28



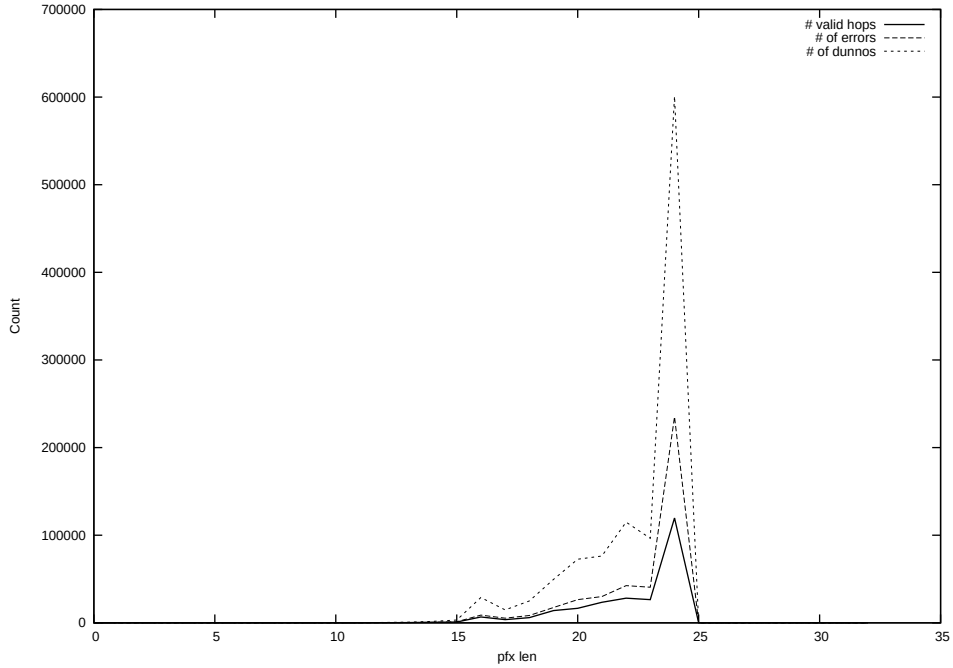
2013-09-29



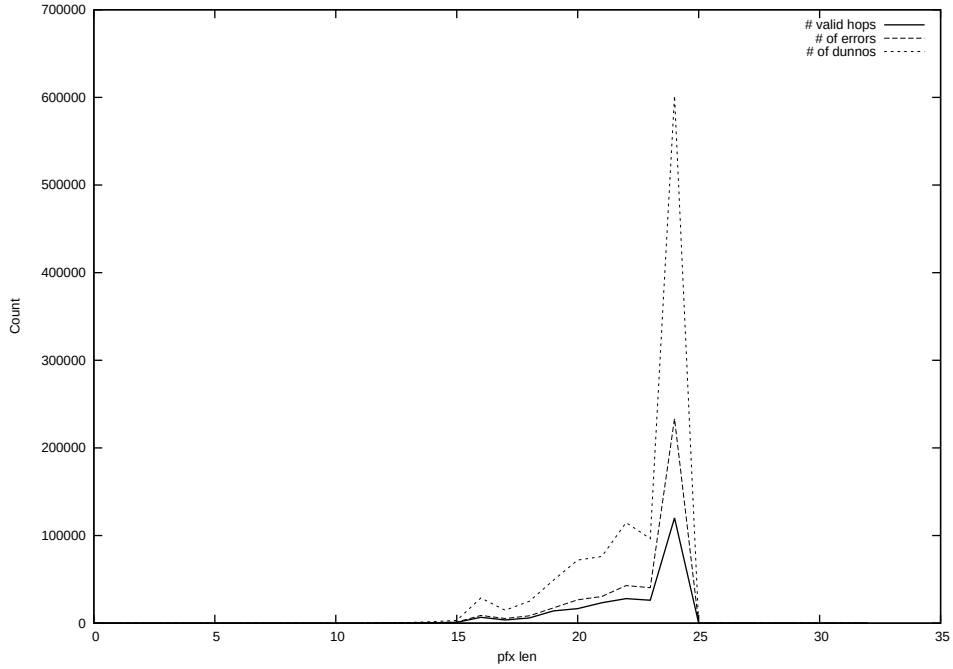
2013-09-30



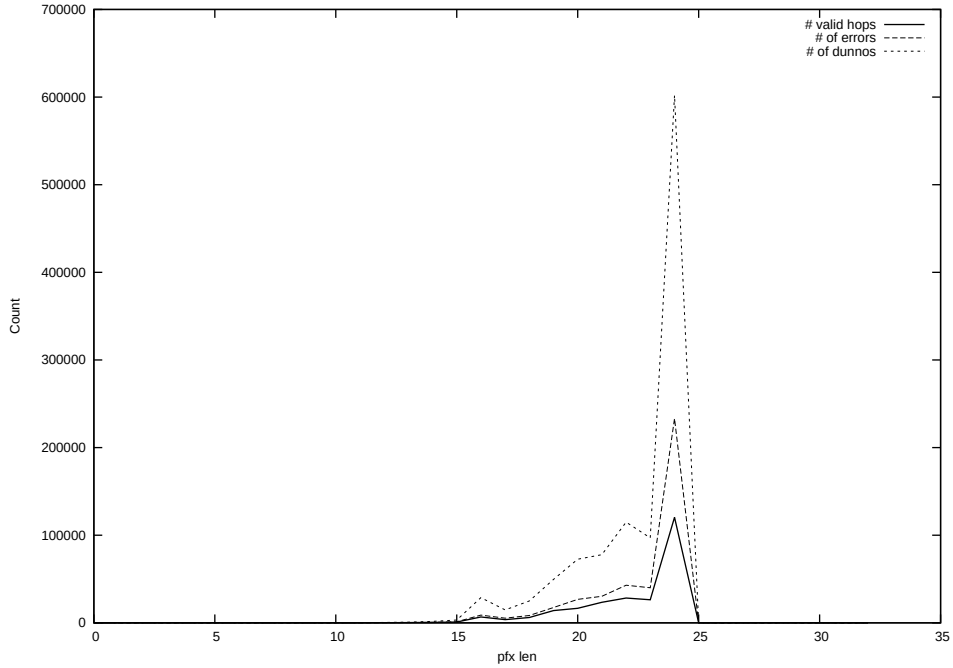
2013-10-01



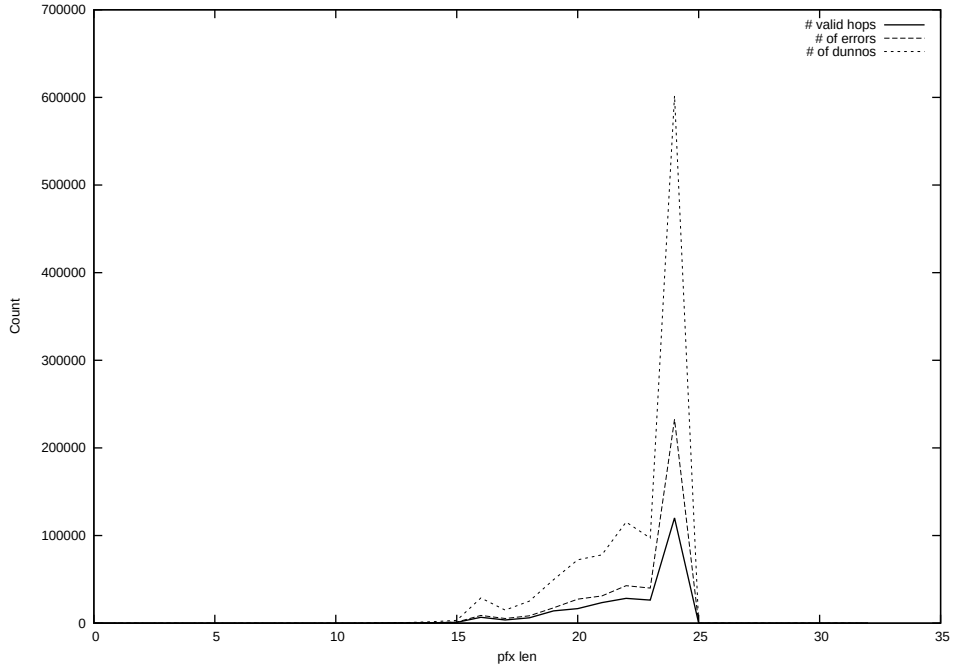
2013-10-02



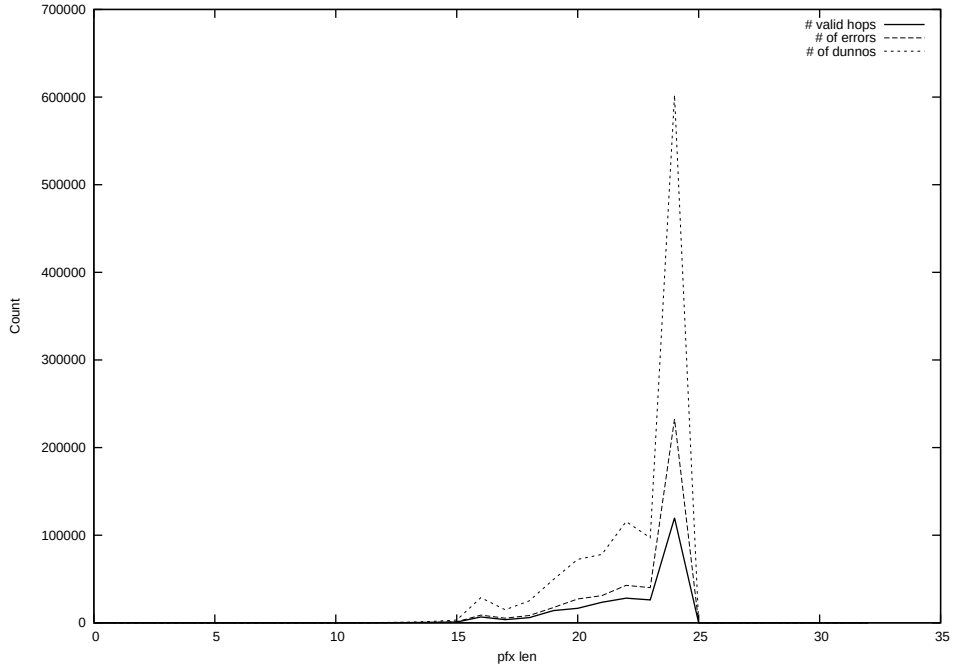
2013-10-03



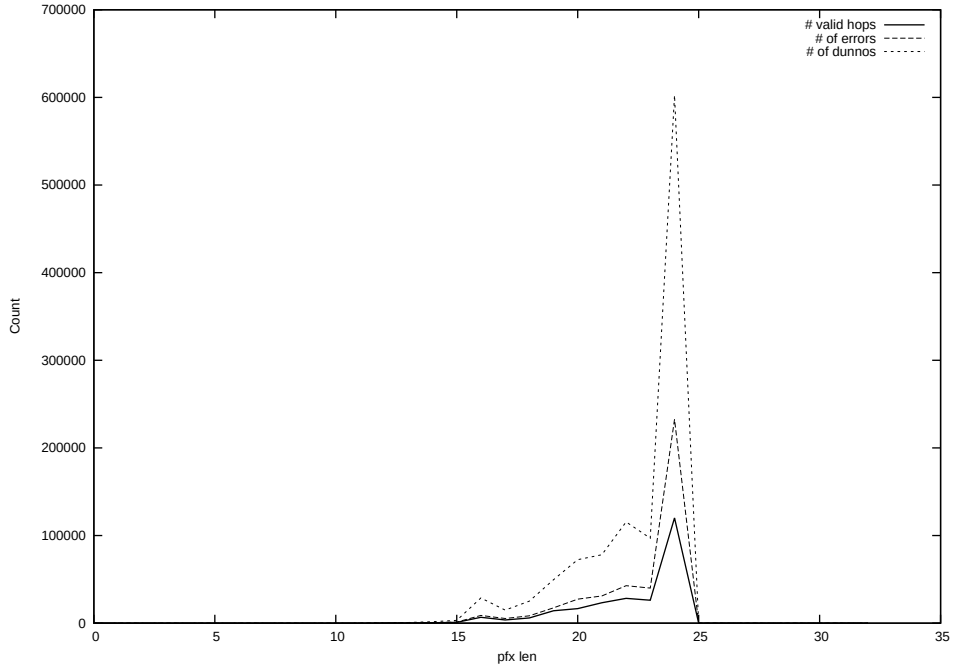
2013-10-04



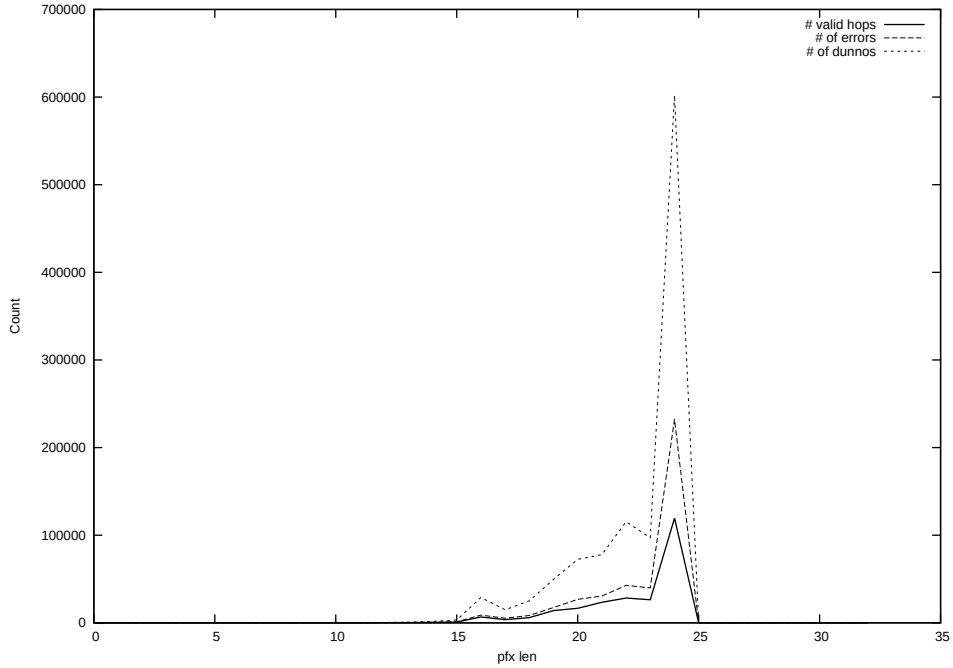
2013-10-05



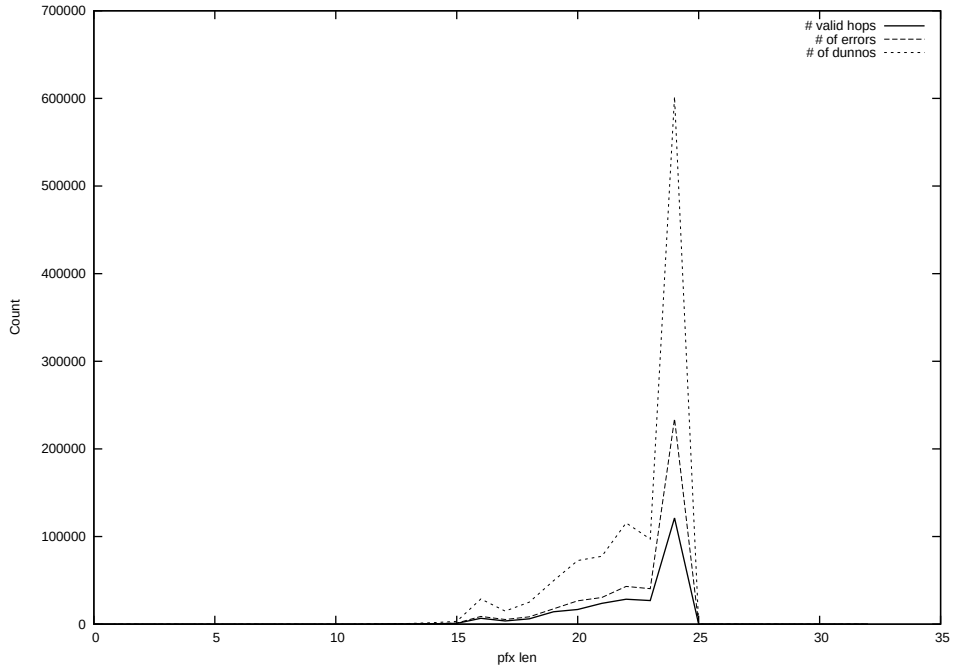
2013-10-06



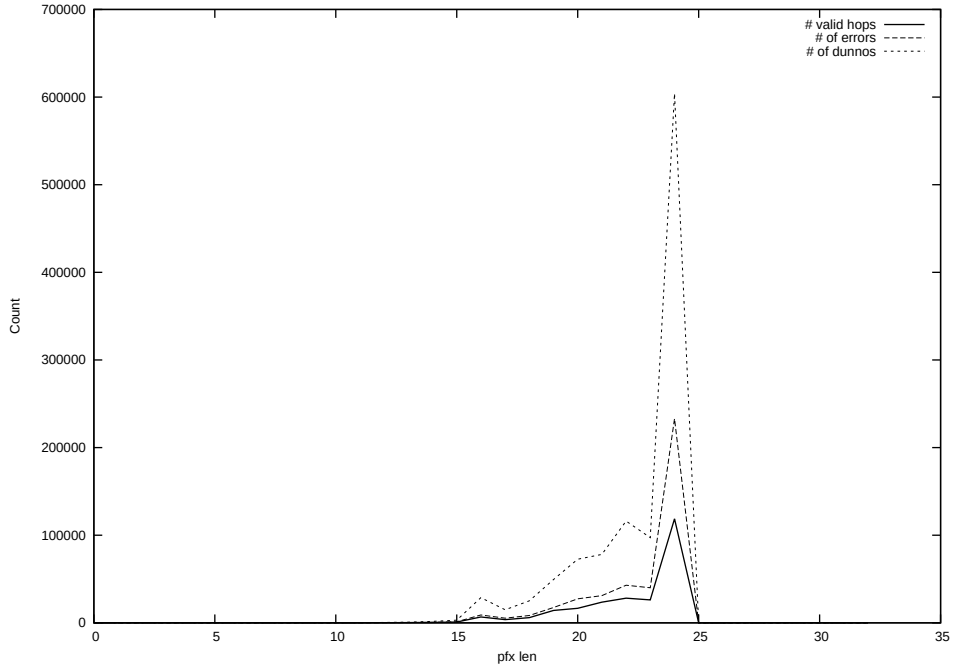
2013-10-07



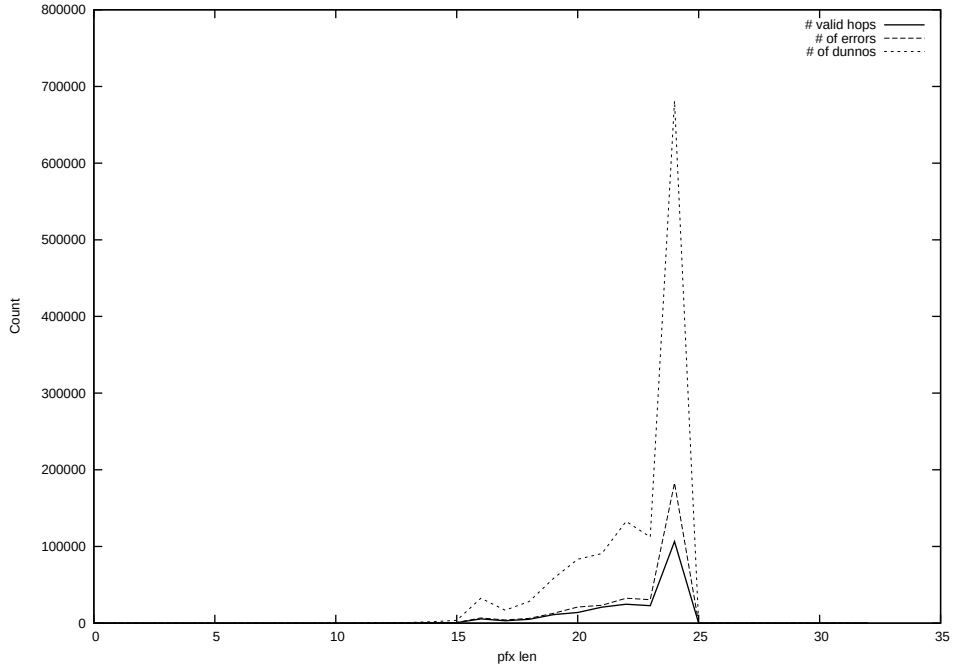
2013-10-08



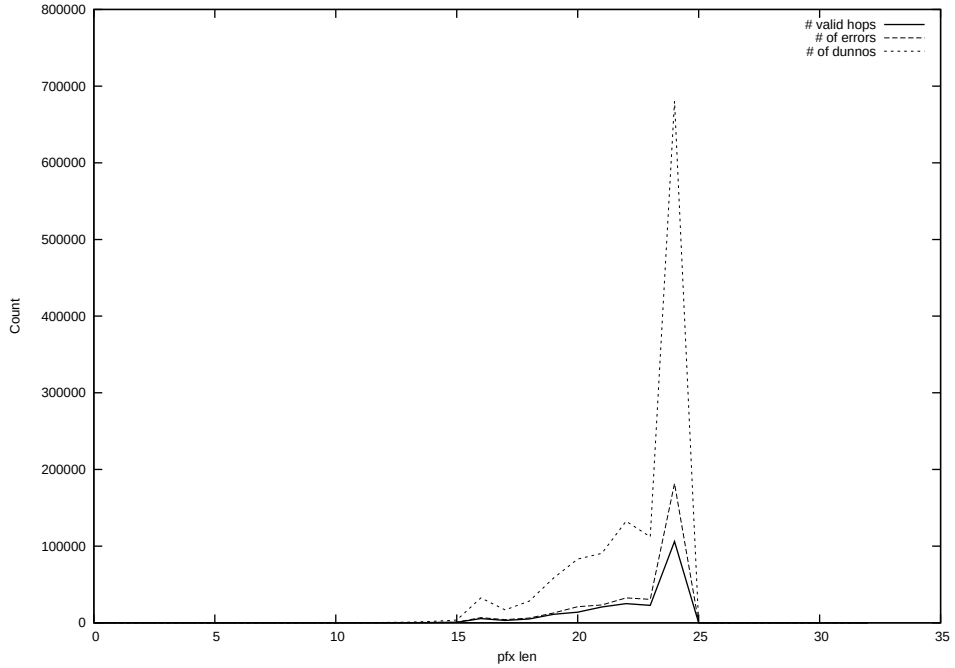
2013-10-09



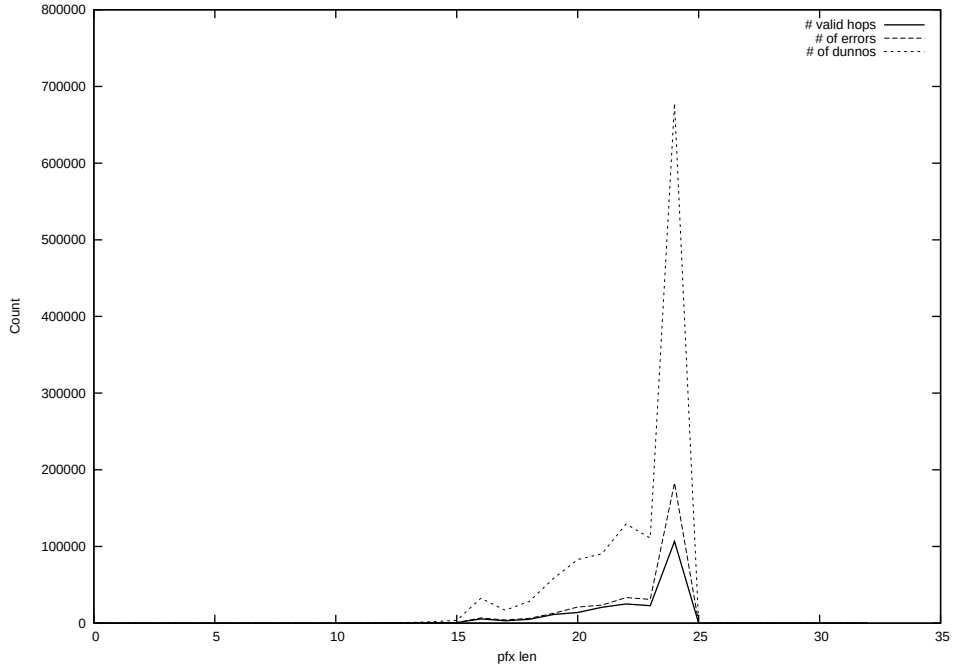
2013-10-10



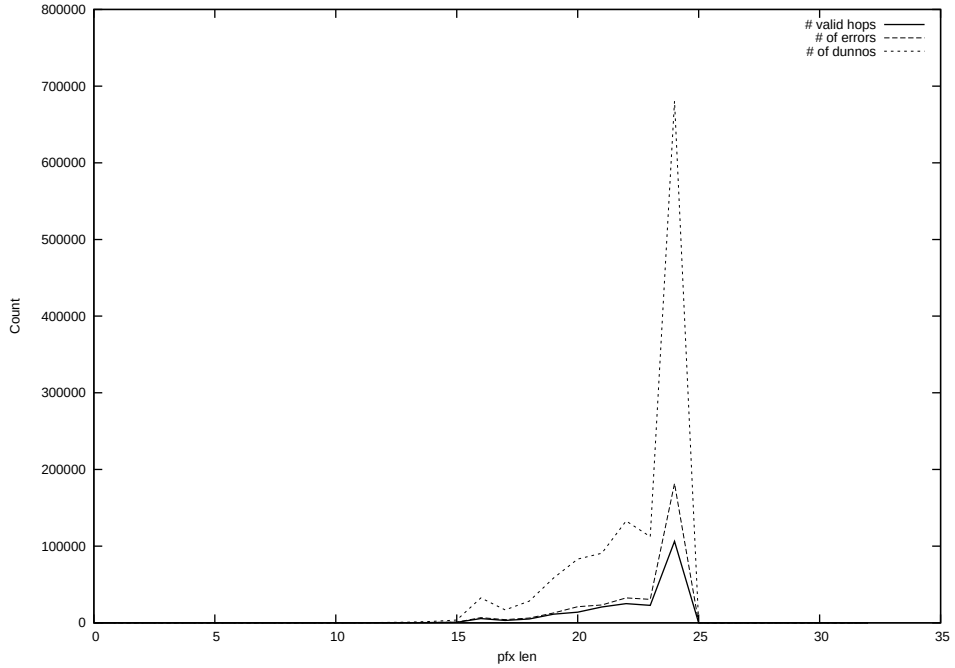
2013-10-11



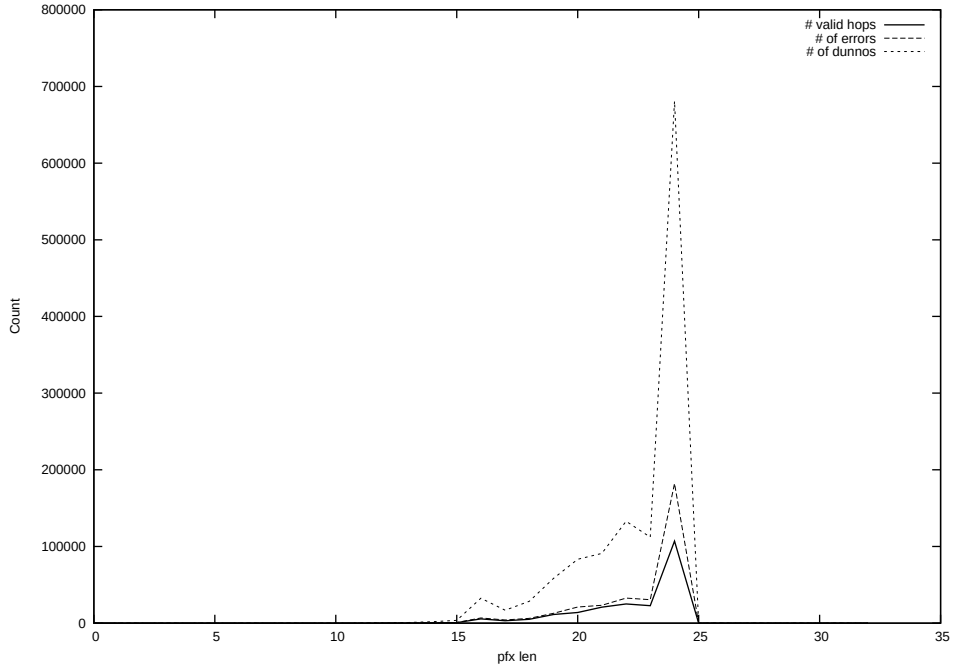
2013-10-12



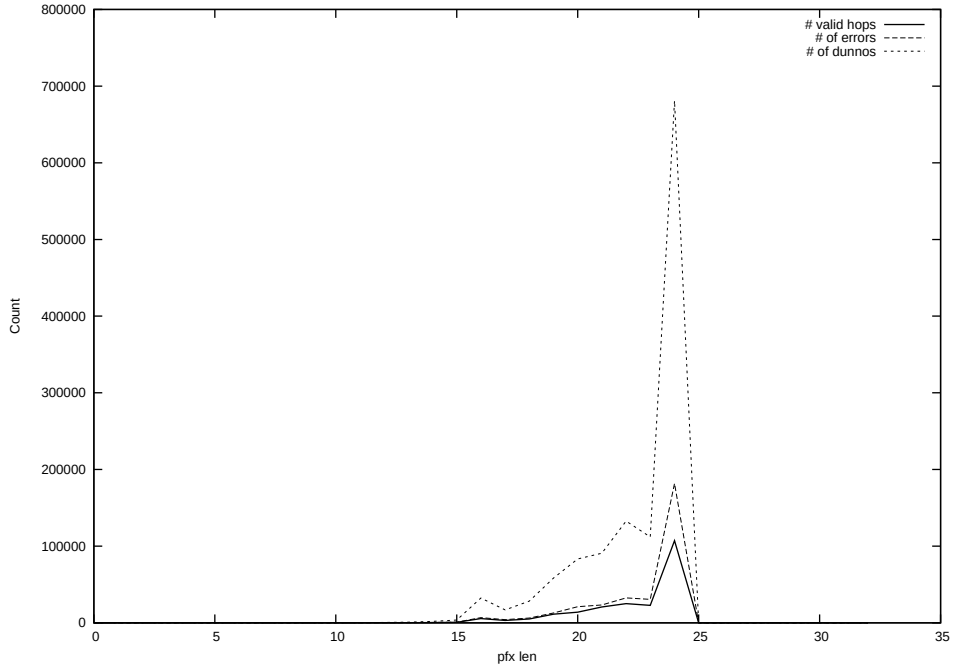
2013-10-13



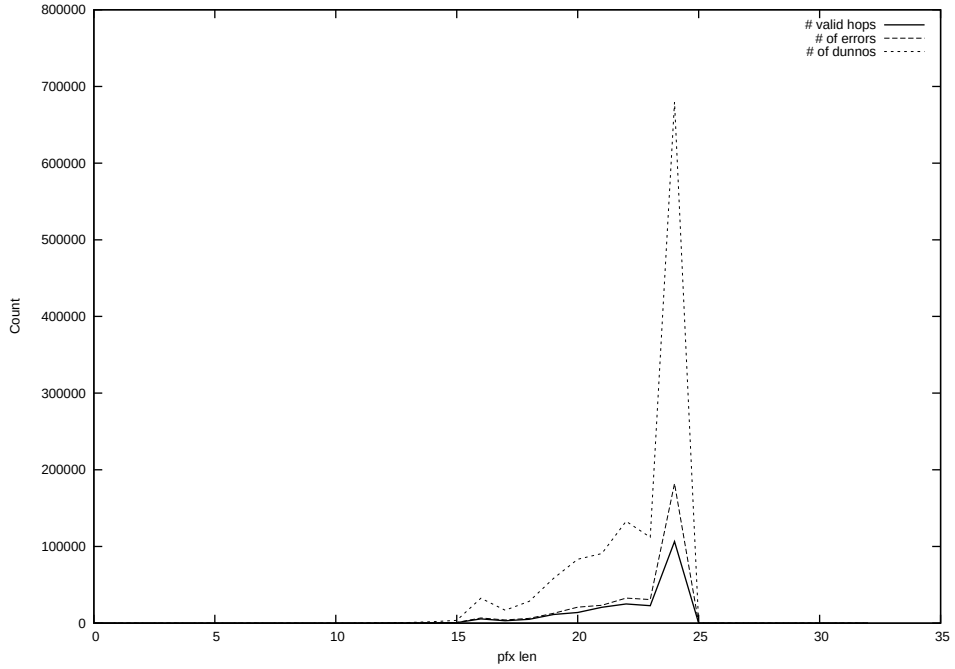
2013-10-14



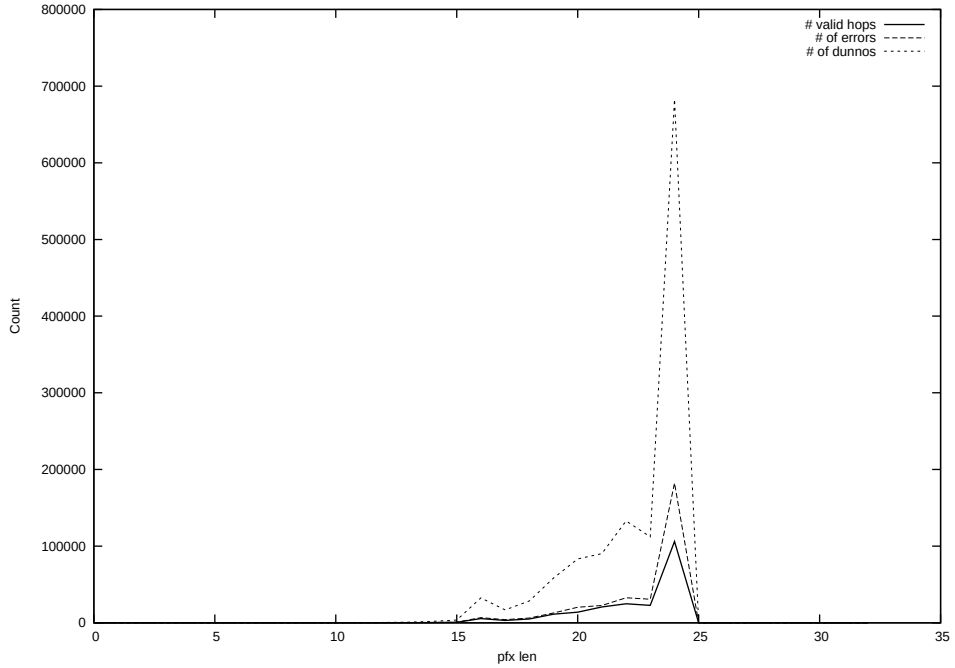
2013-10-15



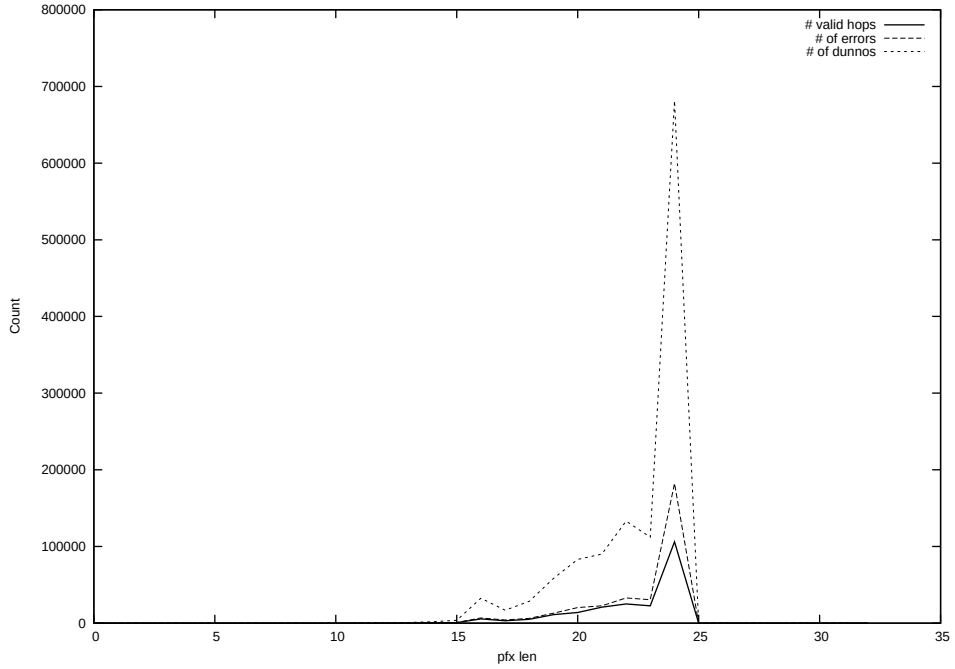
2013-10-16



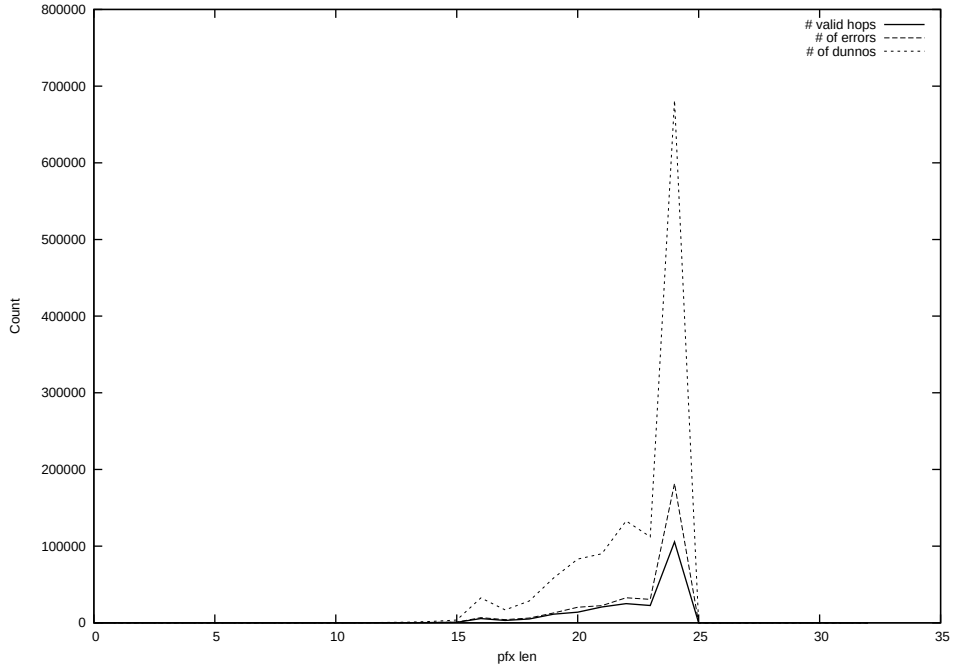
2013-10-17



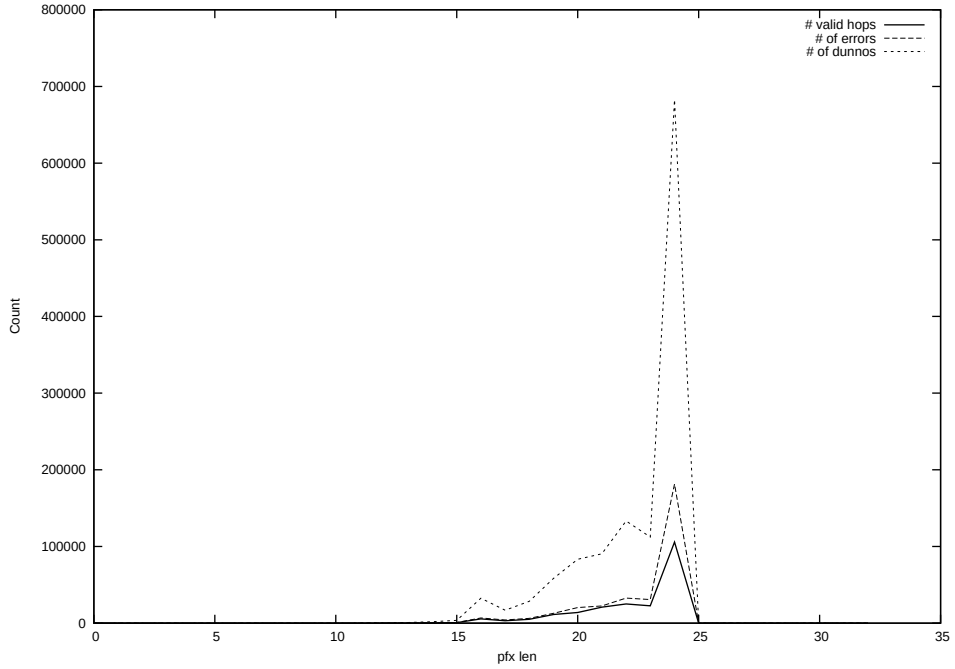
2013-10-18



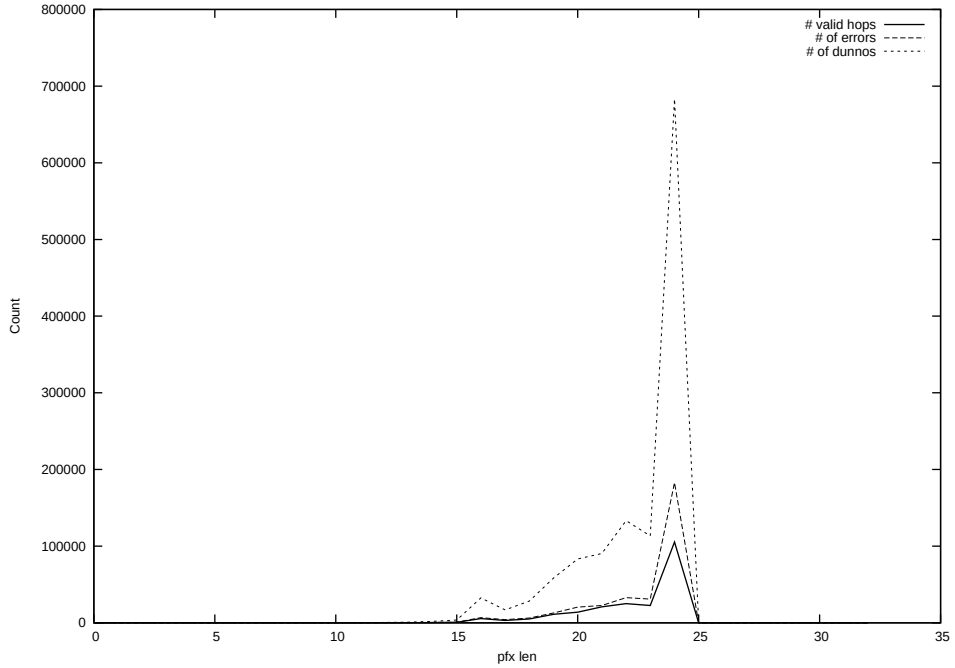
2013-10-19



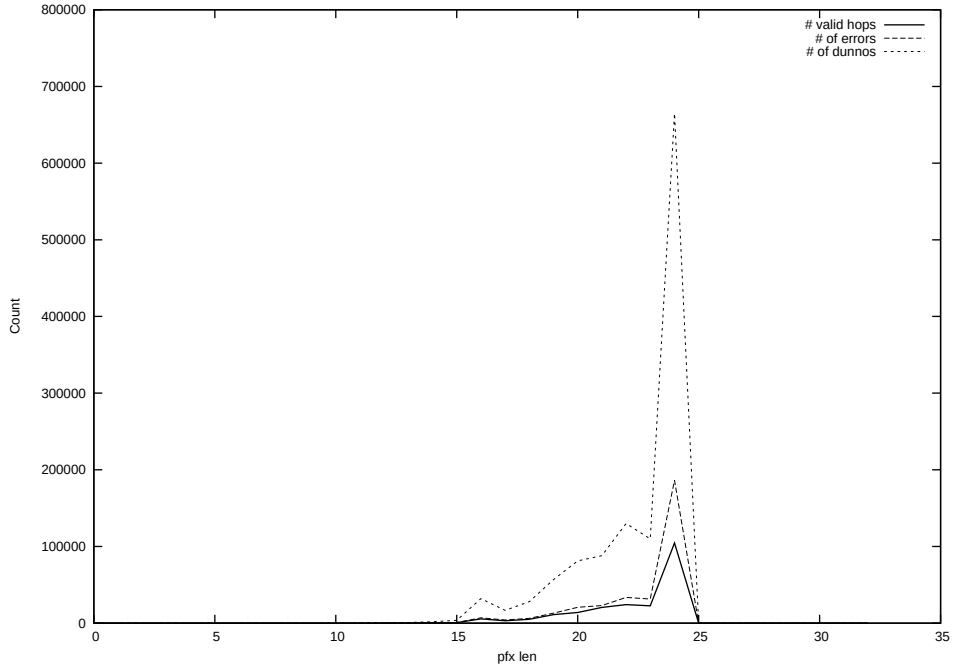
2013-10-20



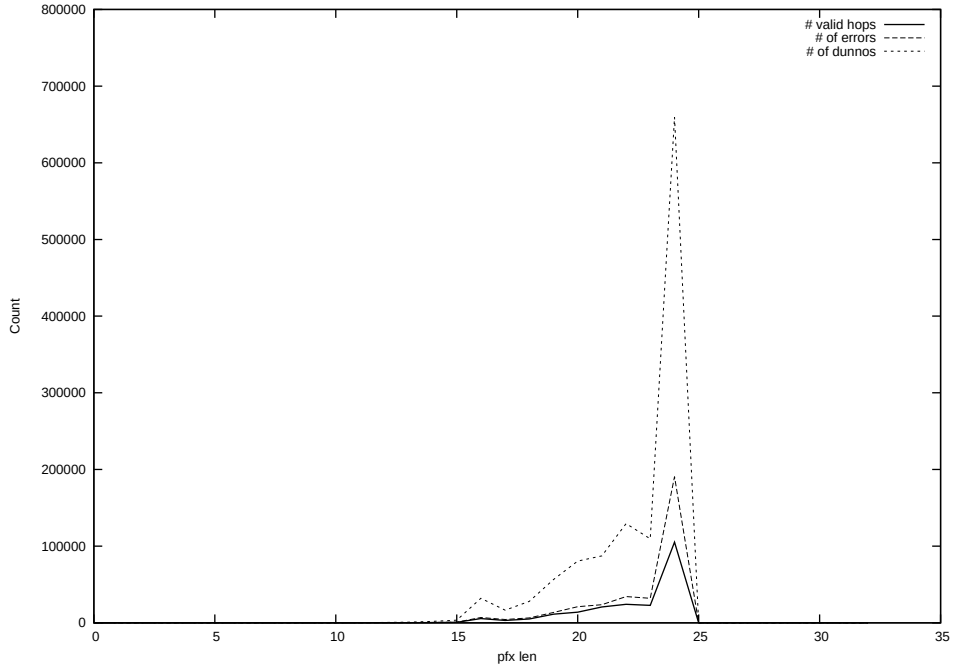
2013-10-21



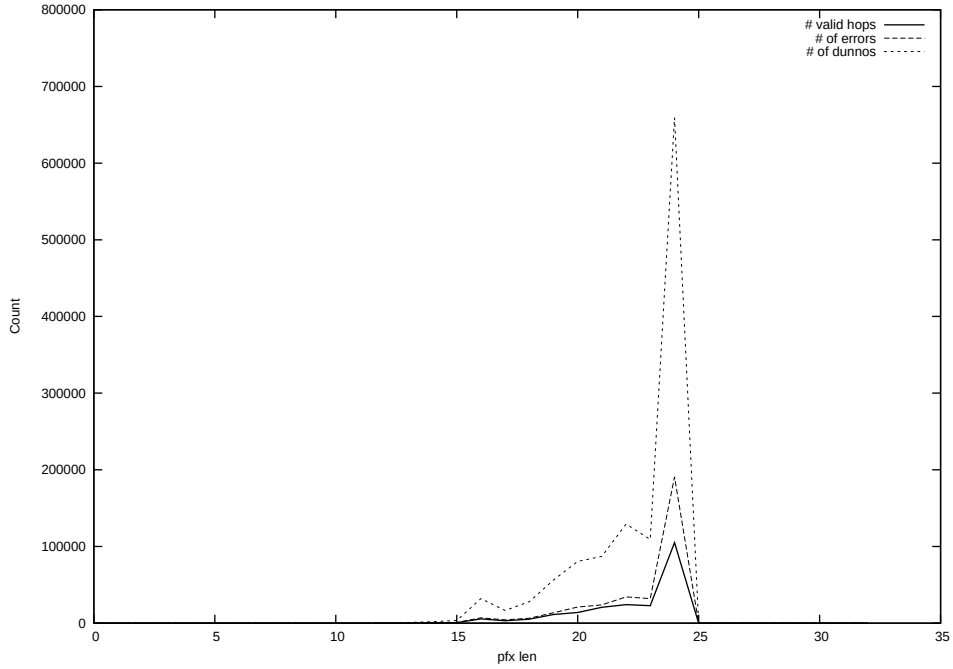
2013-10-22



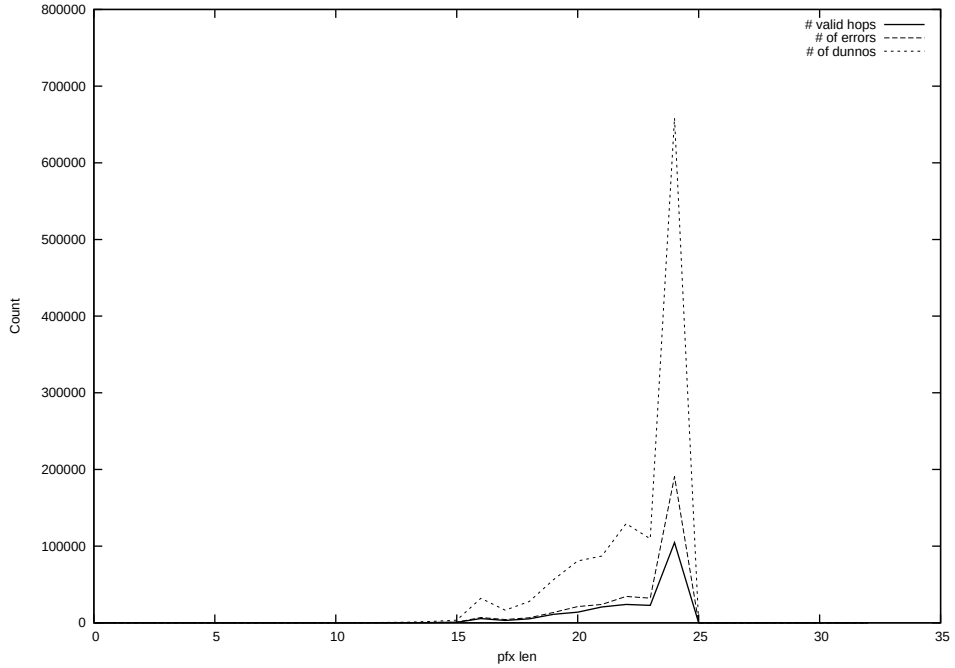
2013-10-23



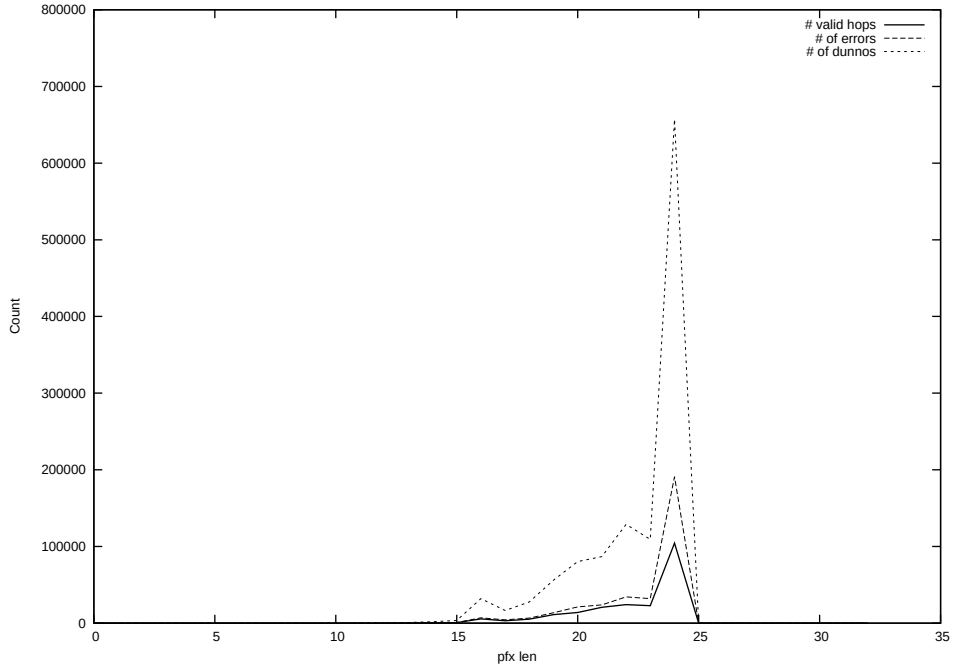
2013-10-24



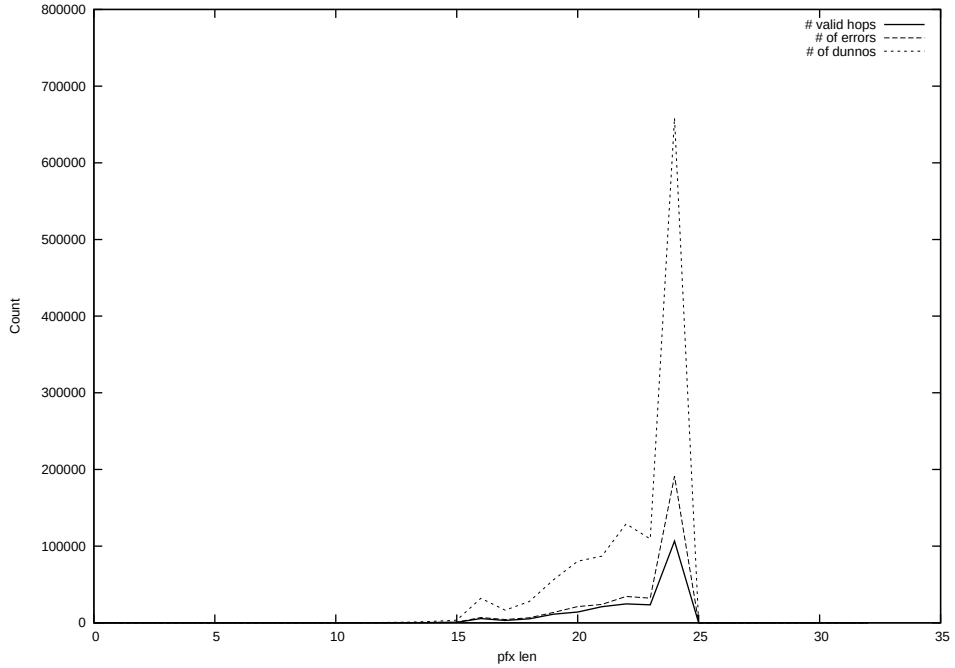
2013-10-25



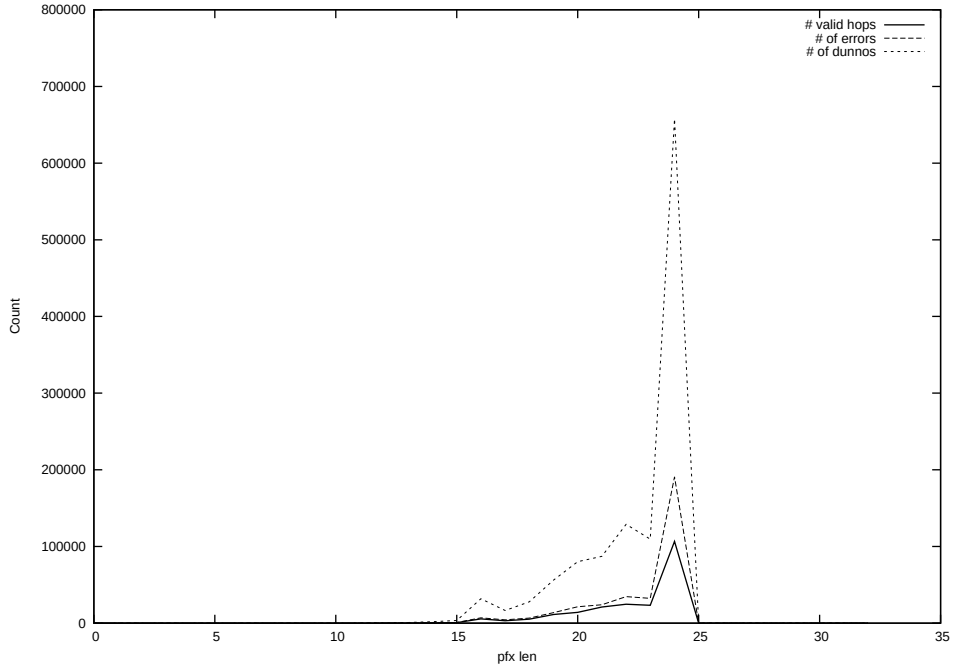
2013-10-26



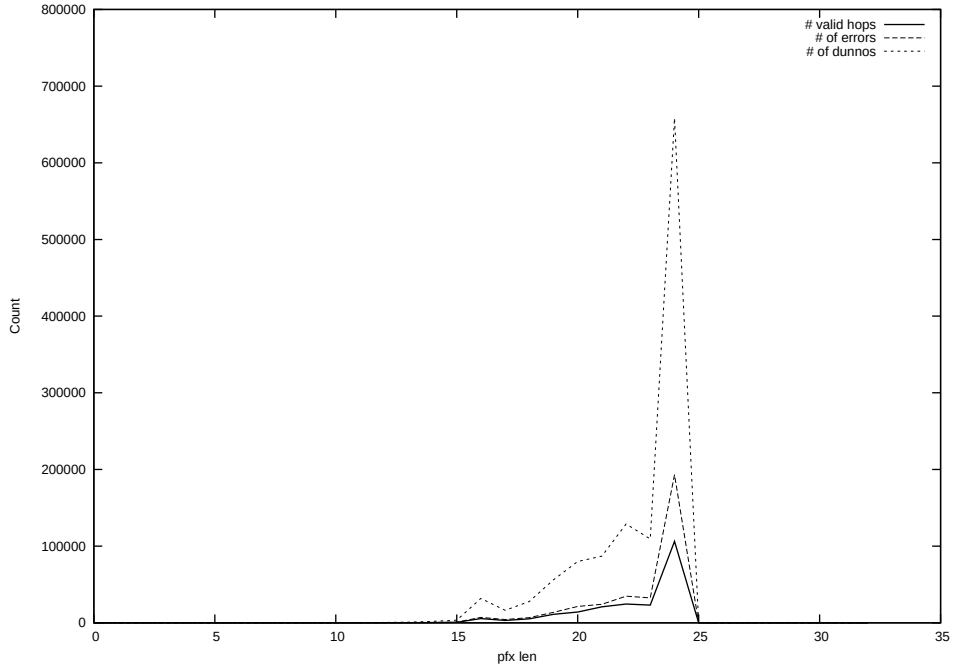
2013-10-27



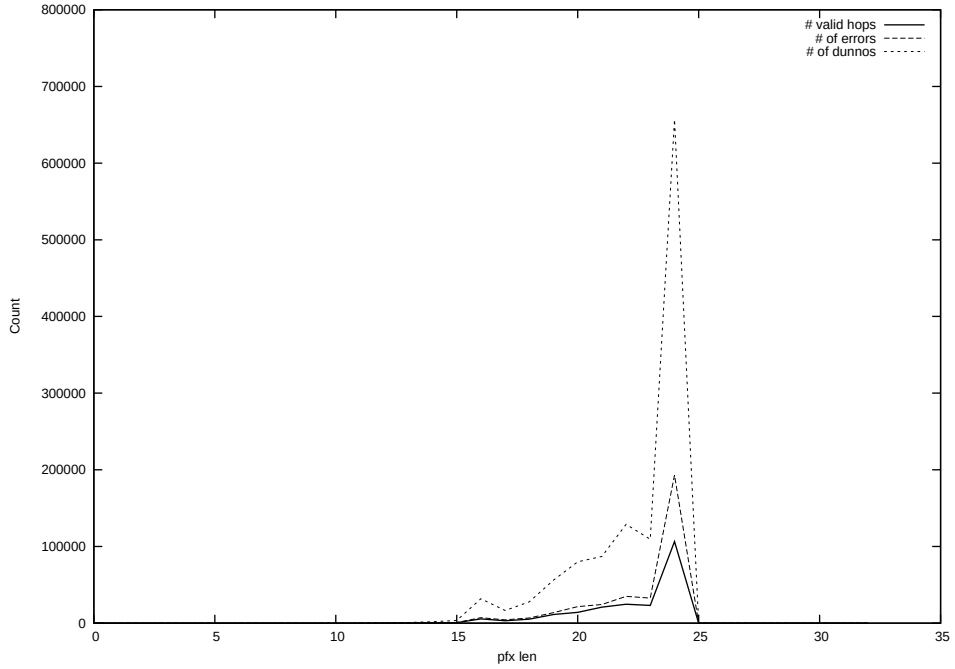
2013-10-28



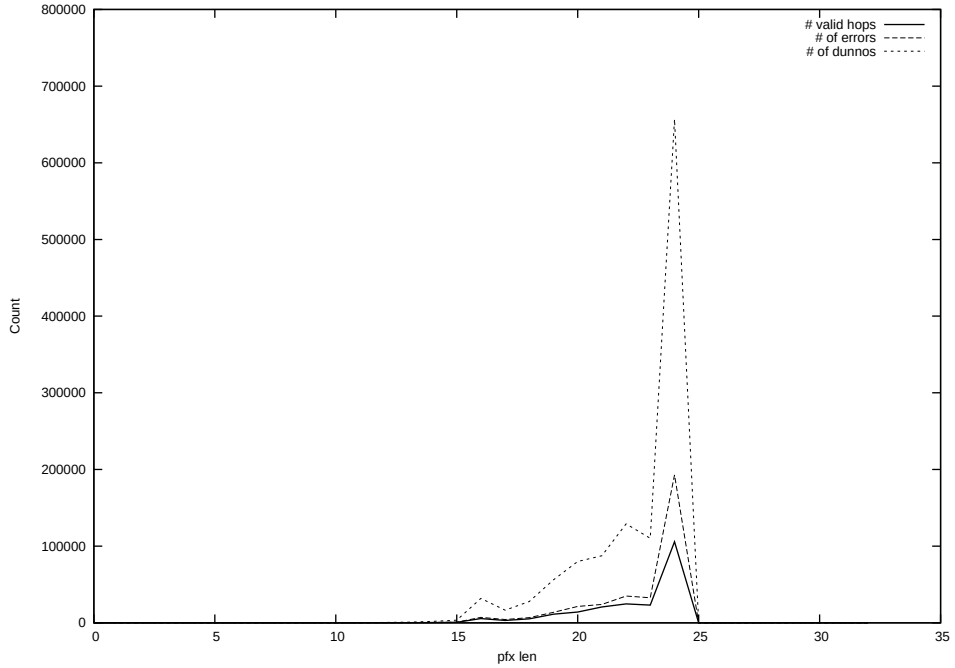
2013-10-29



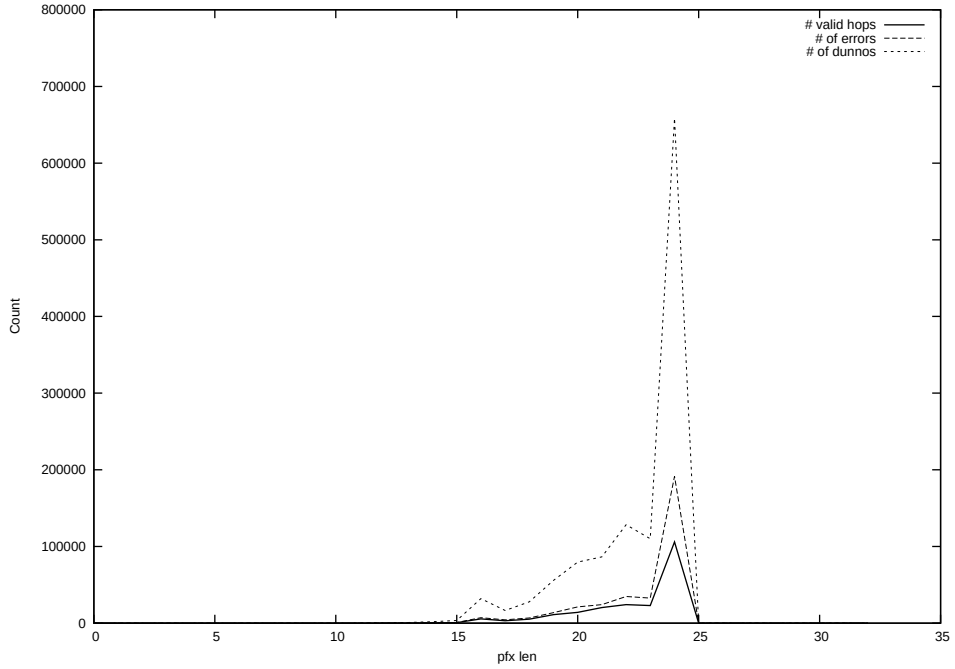
2013-10-30



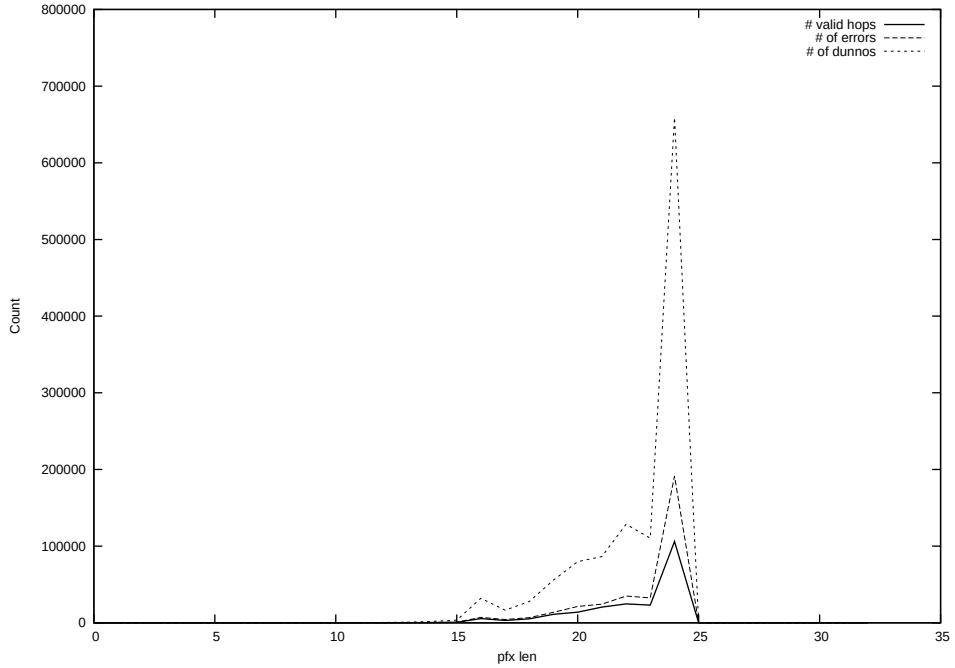
2013-10-31



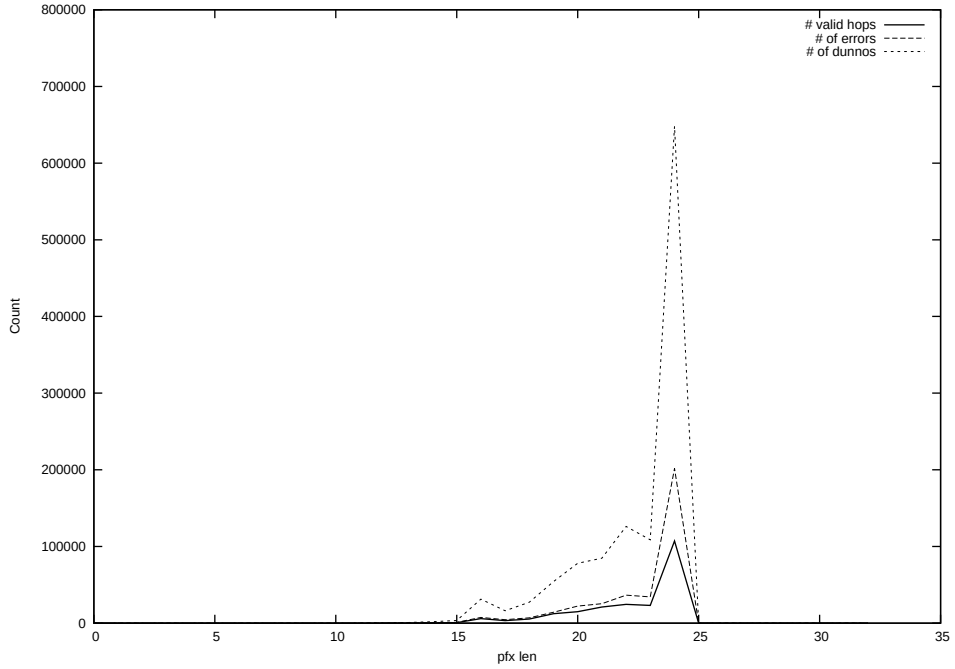
2013-11-01



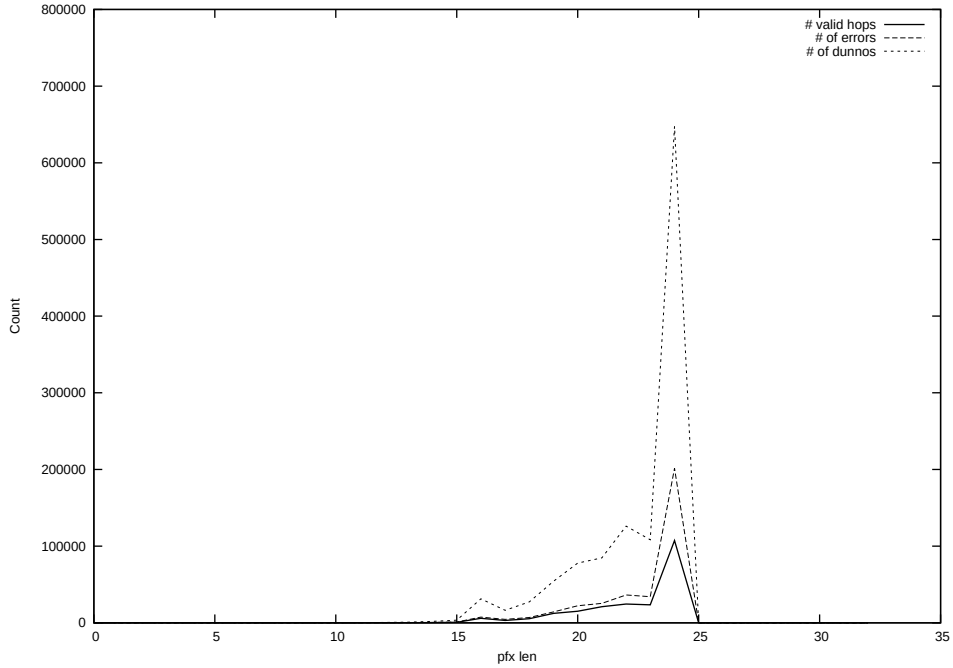
2013-11-02



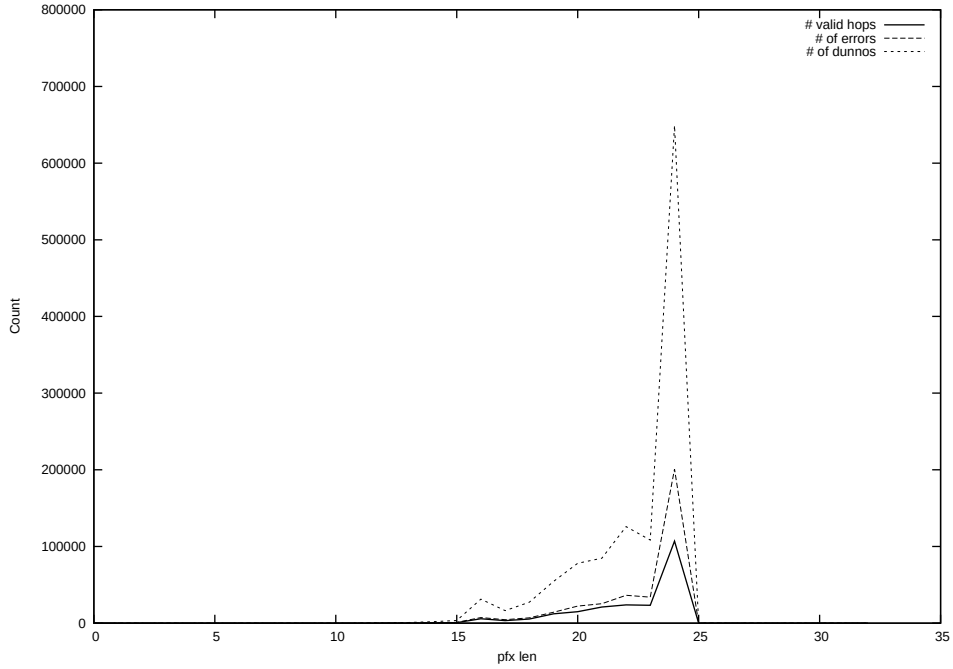
2013-11-03



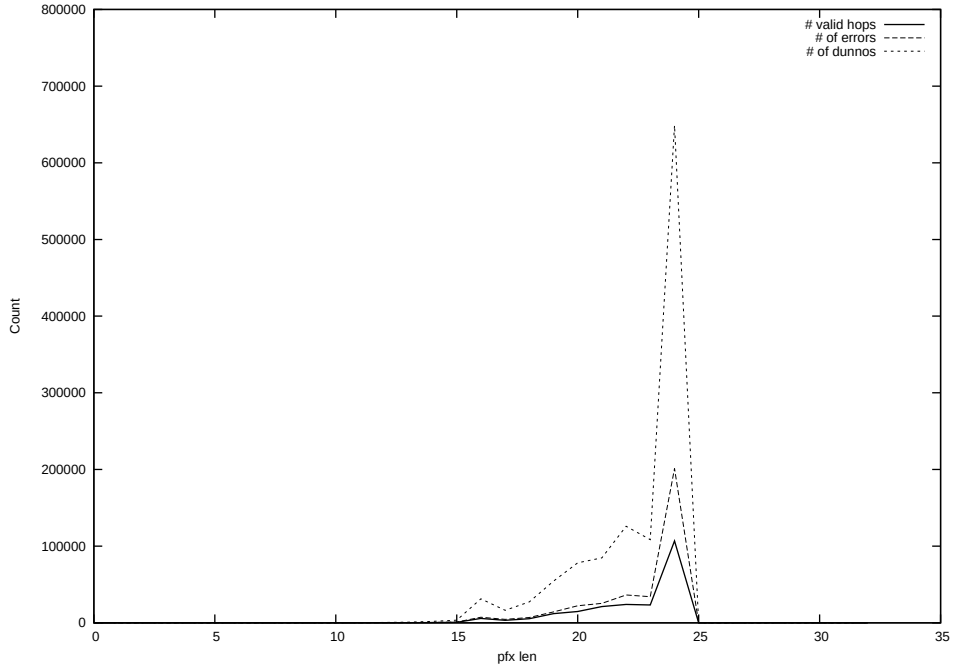
2013-11-04



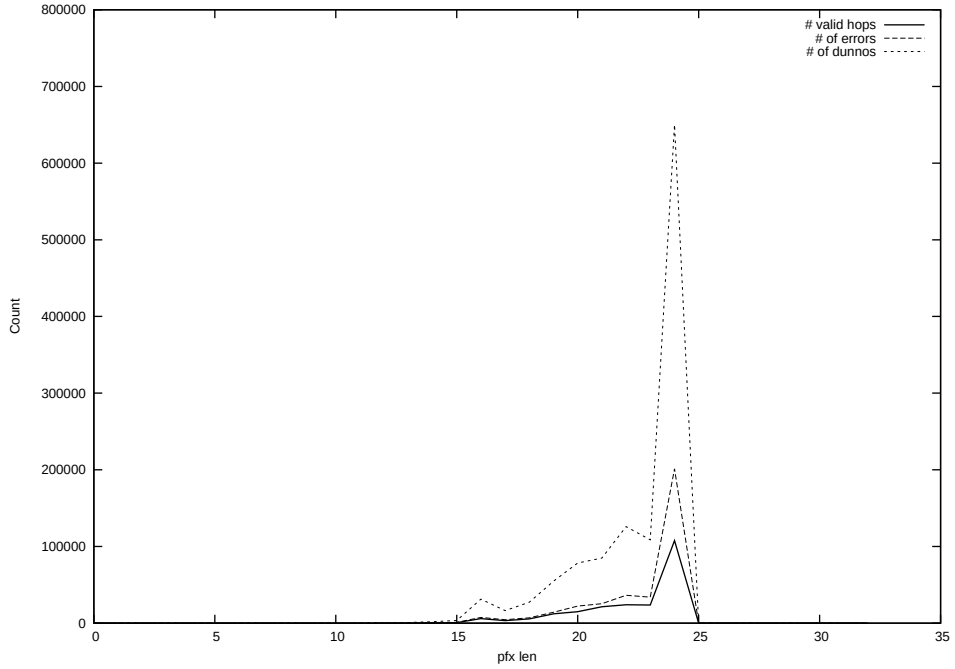
2013-11-05



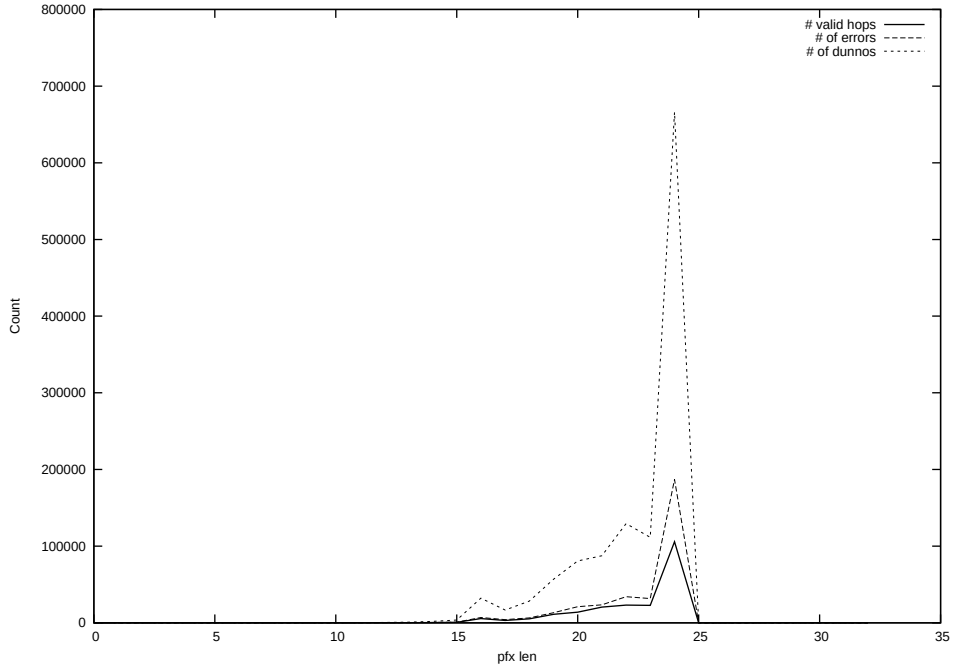
2013-11-06



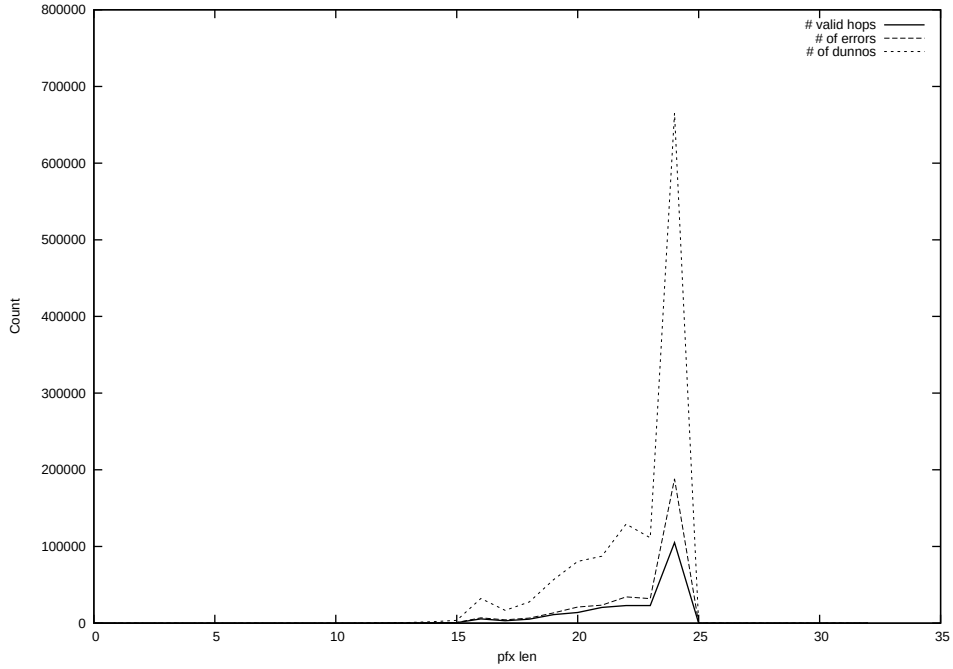
2013-11-07



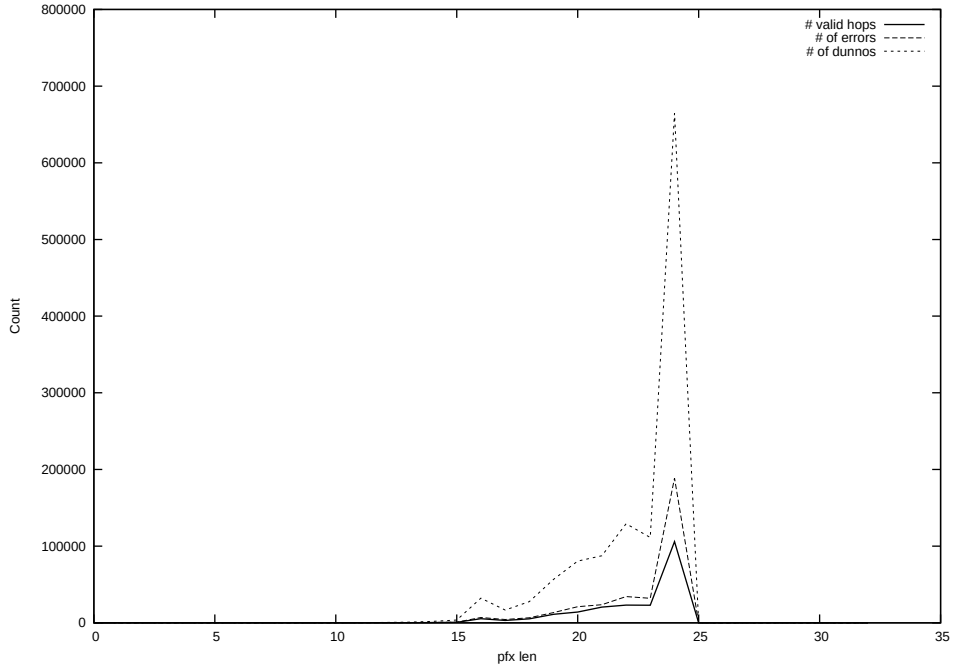
2013-11-08



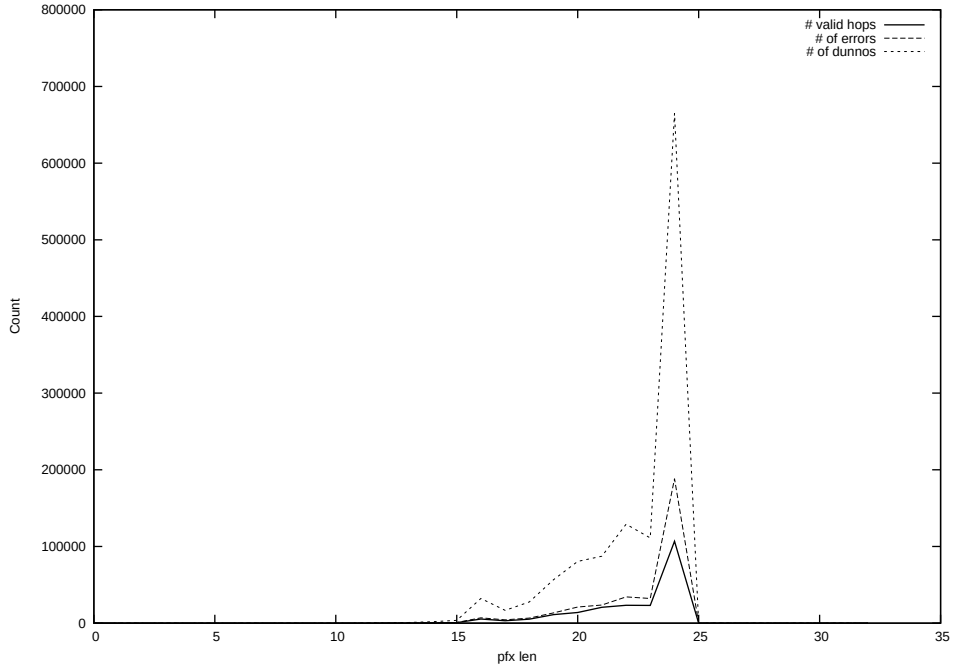
2013-11-09



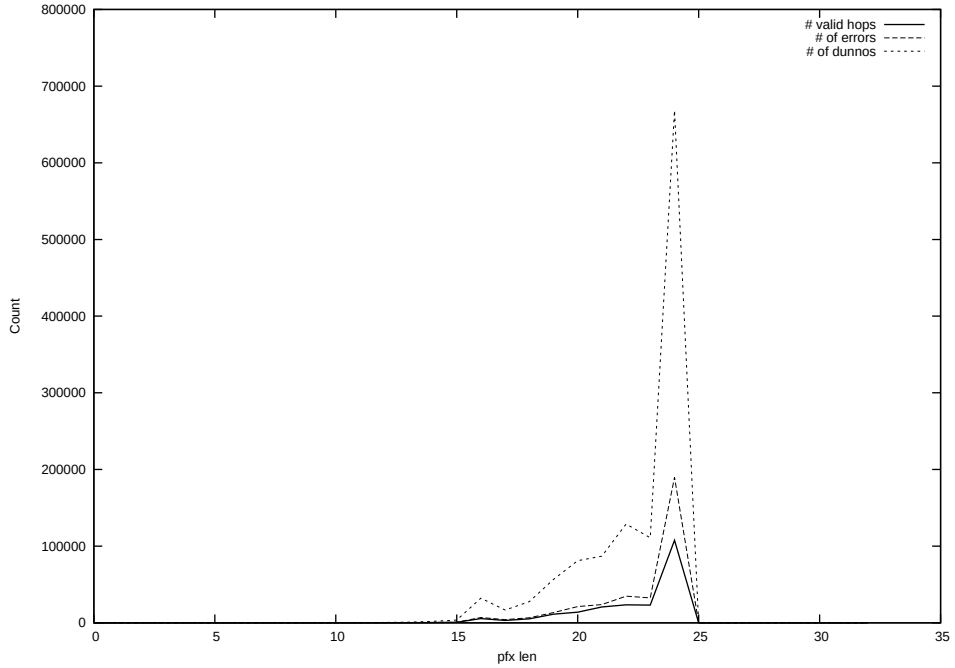
2013-11-10



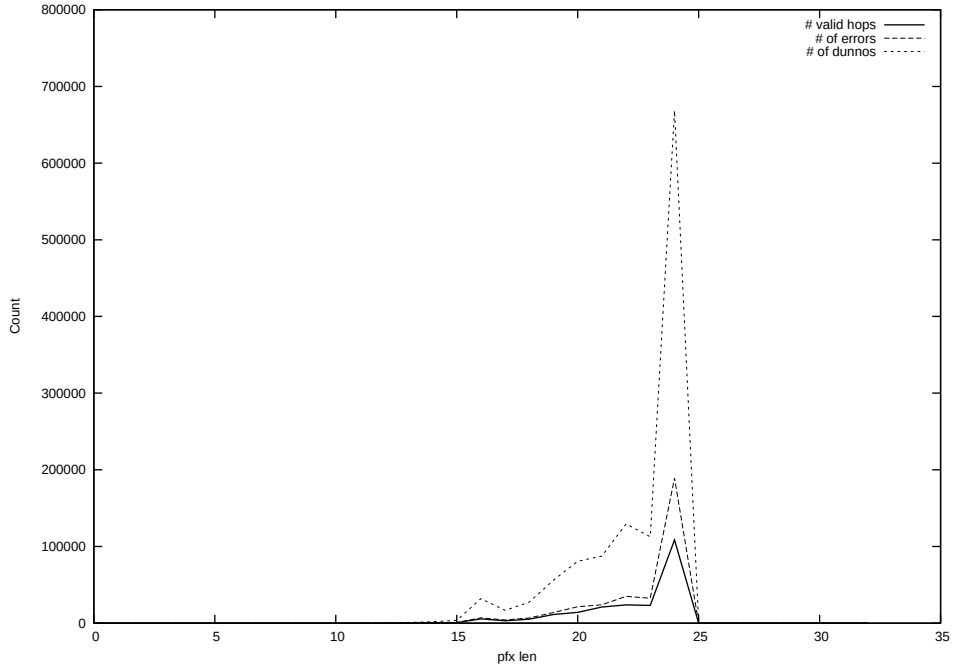
2013-11-11



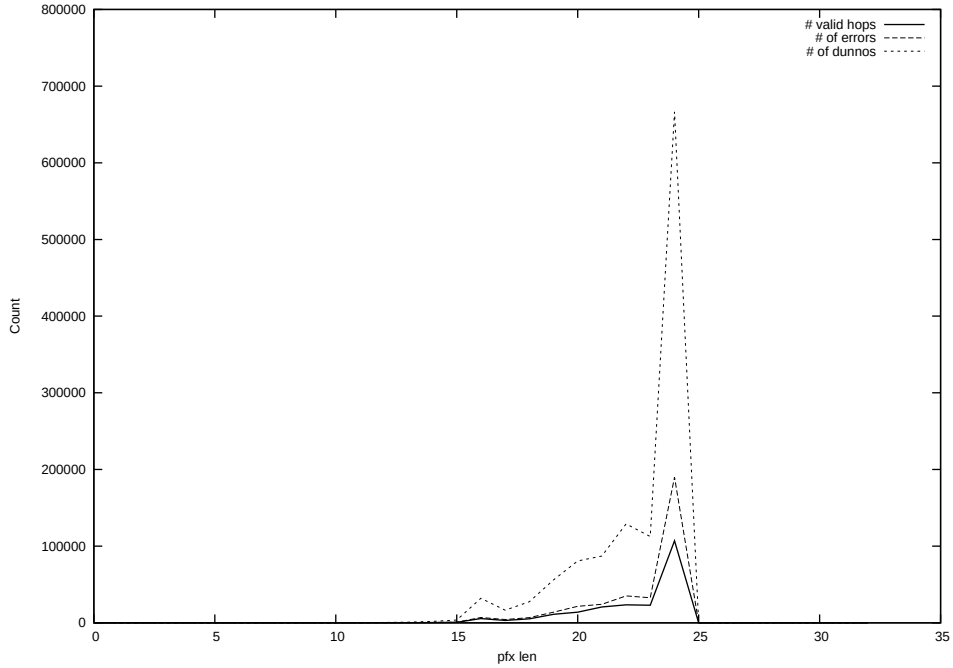
2013-11-12



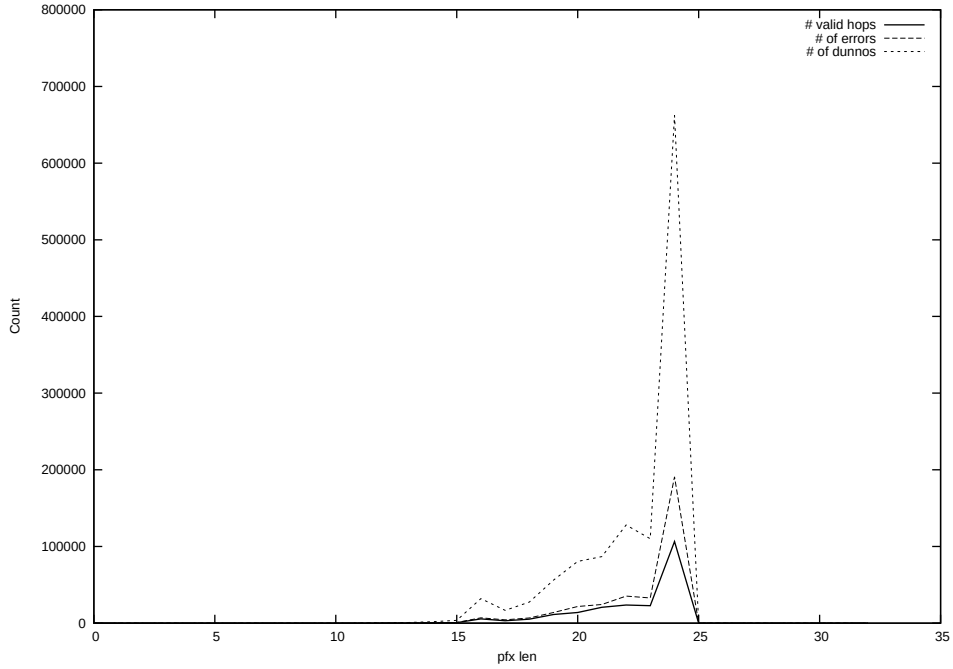
2013-11-13



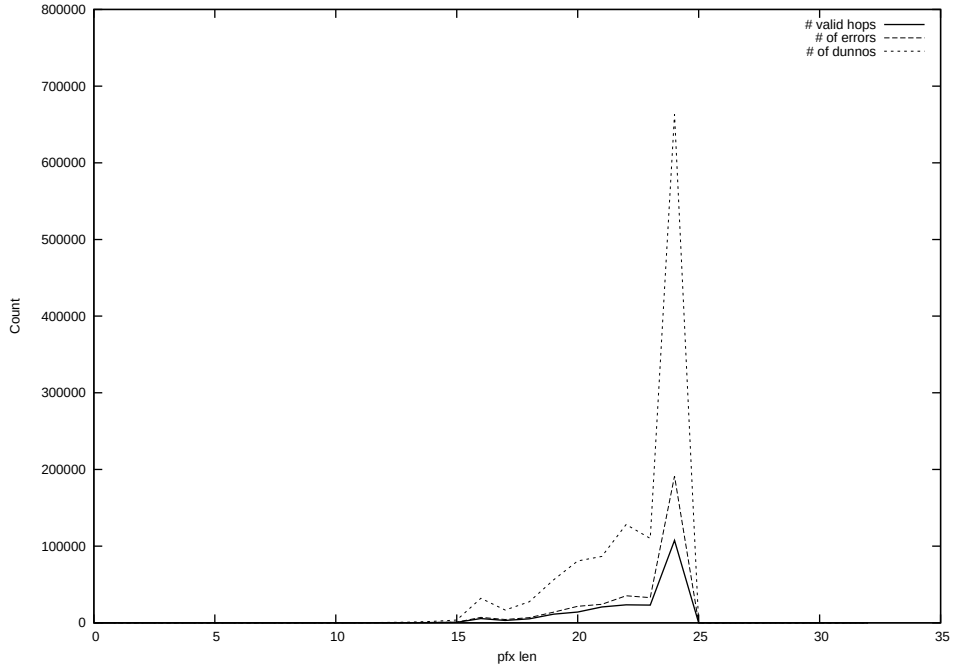
2013-11-14



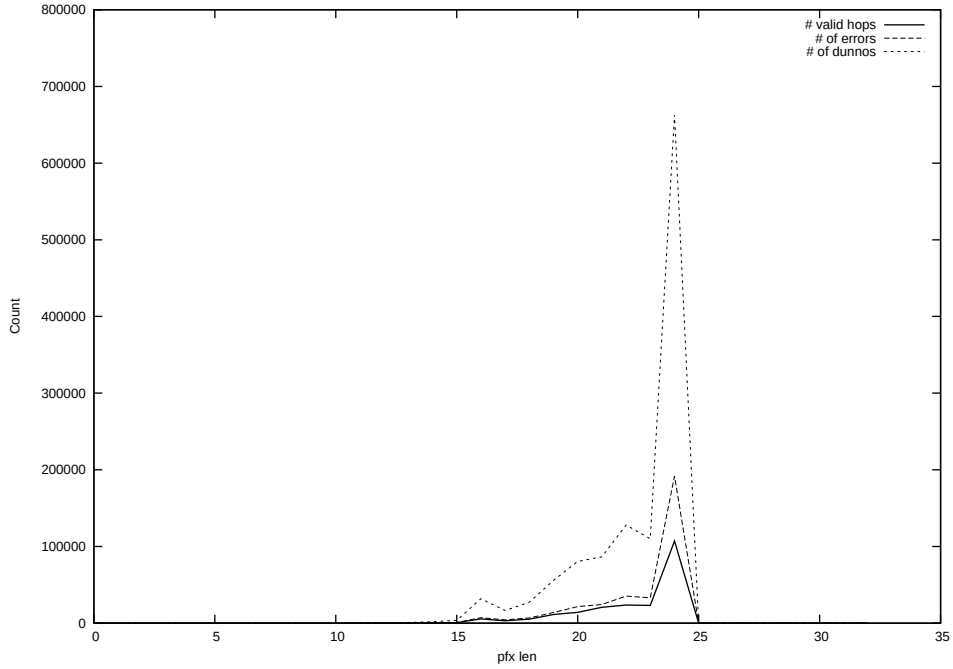
2013-11-15



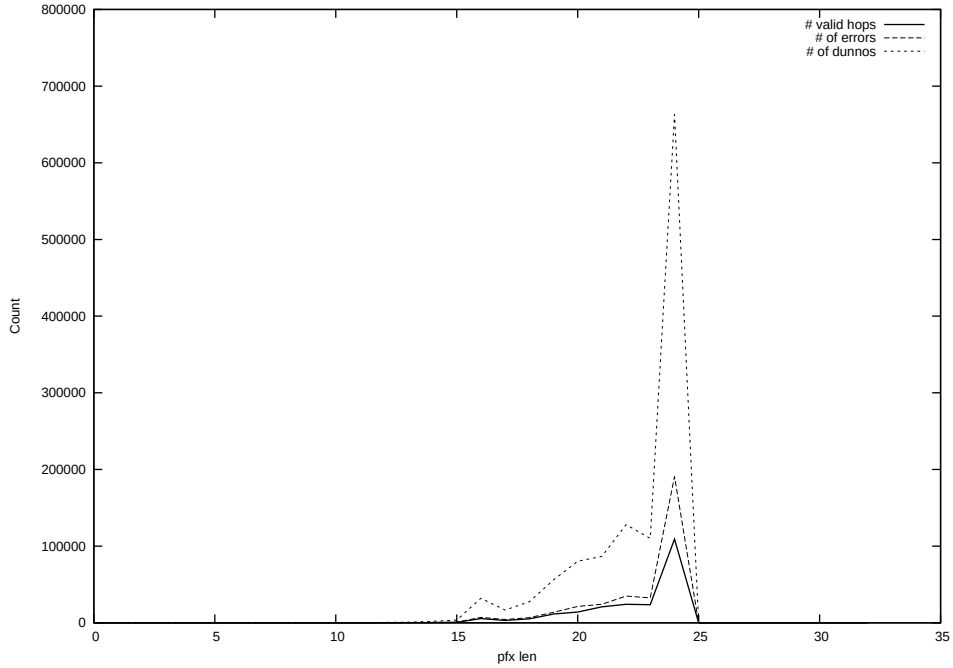
2013-11-16



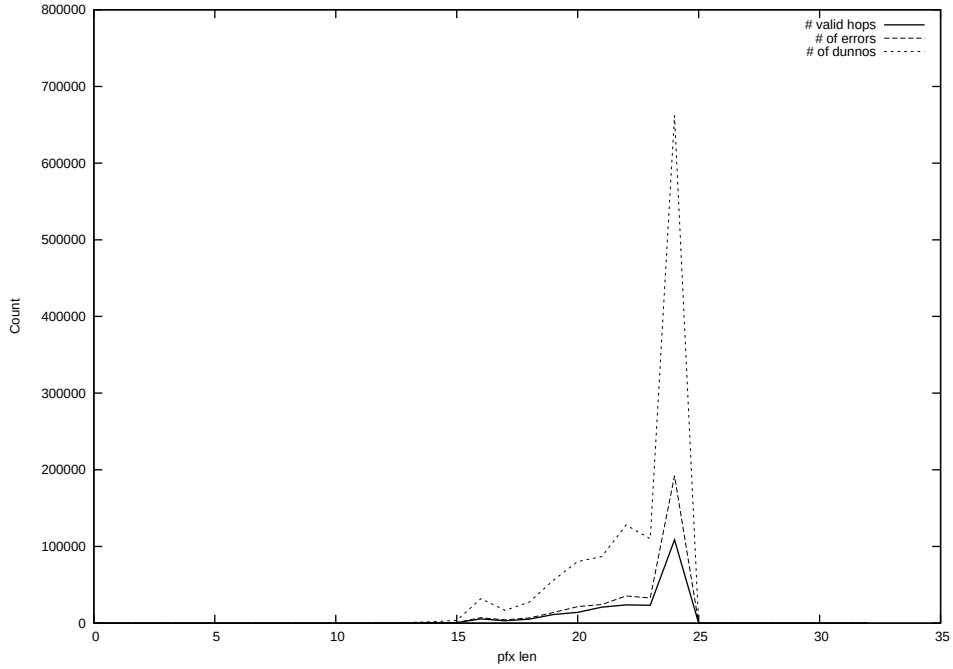
2013-11-17



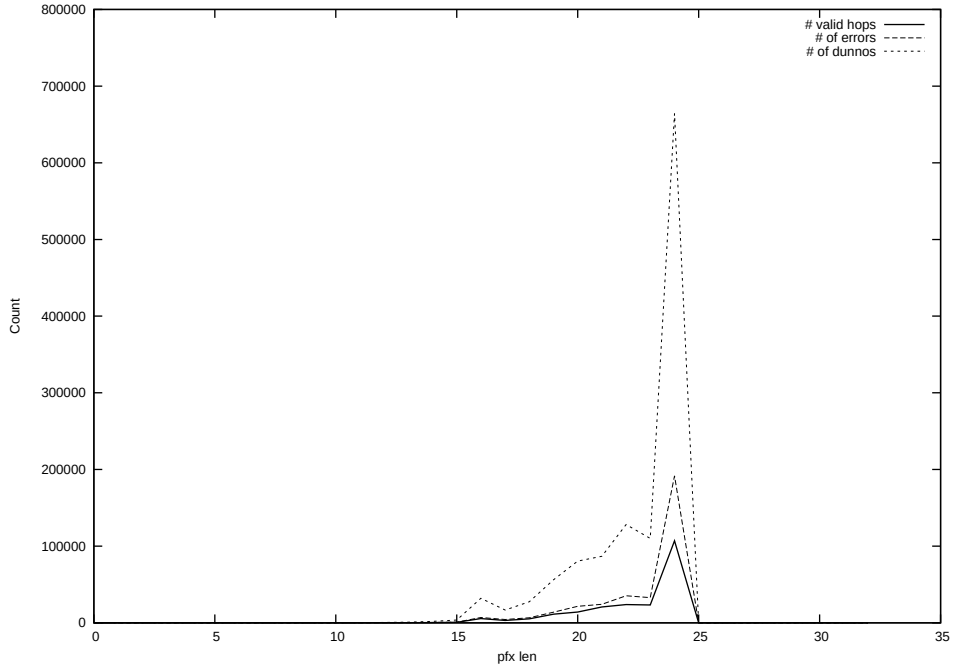
2013-11-18



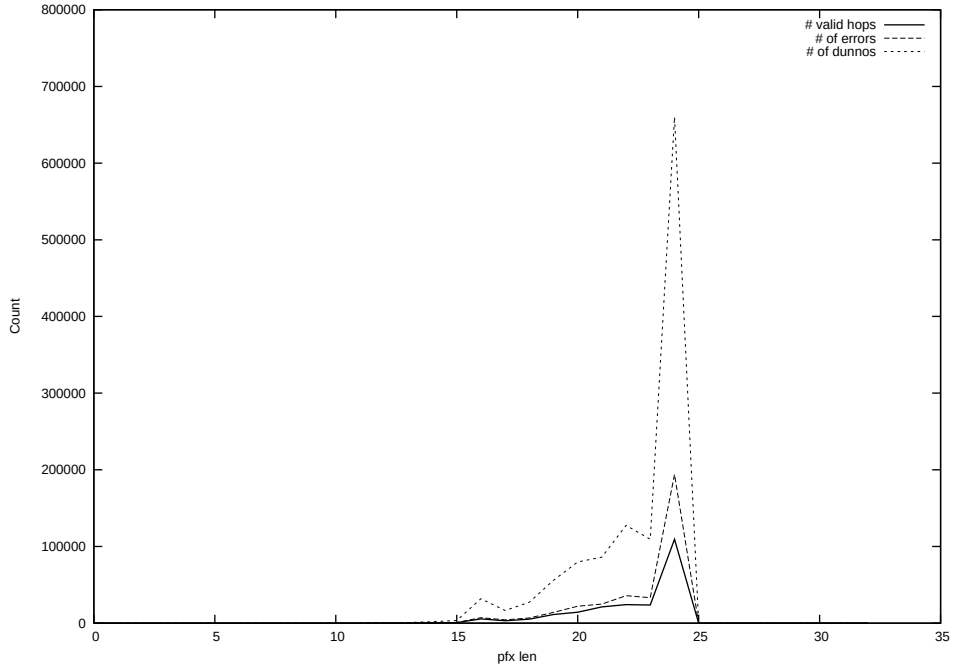
2013-11-19



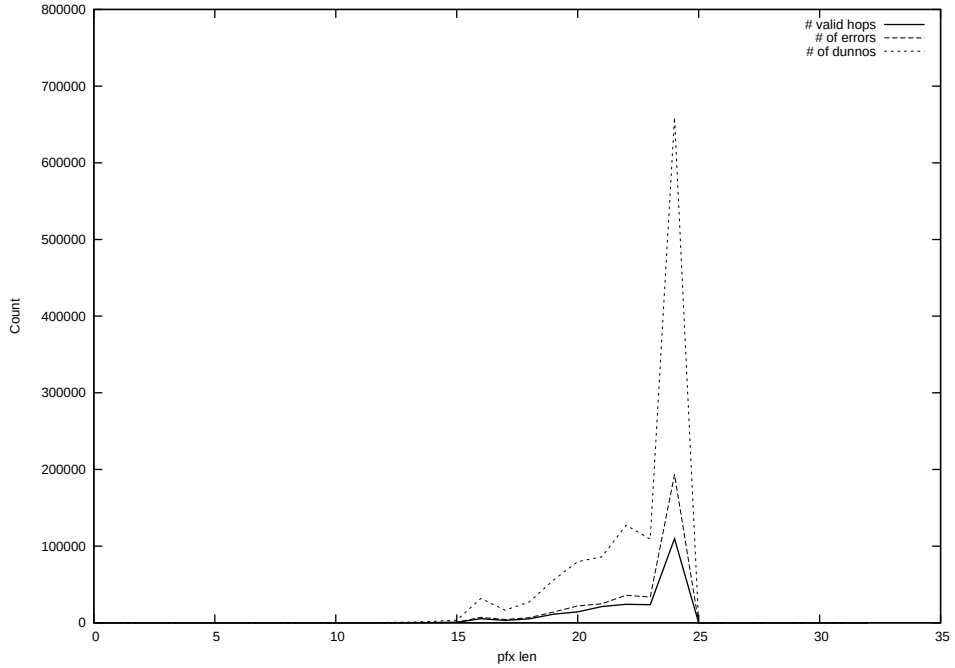
2013-11-20



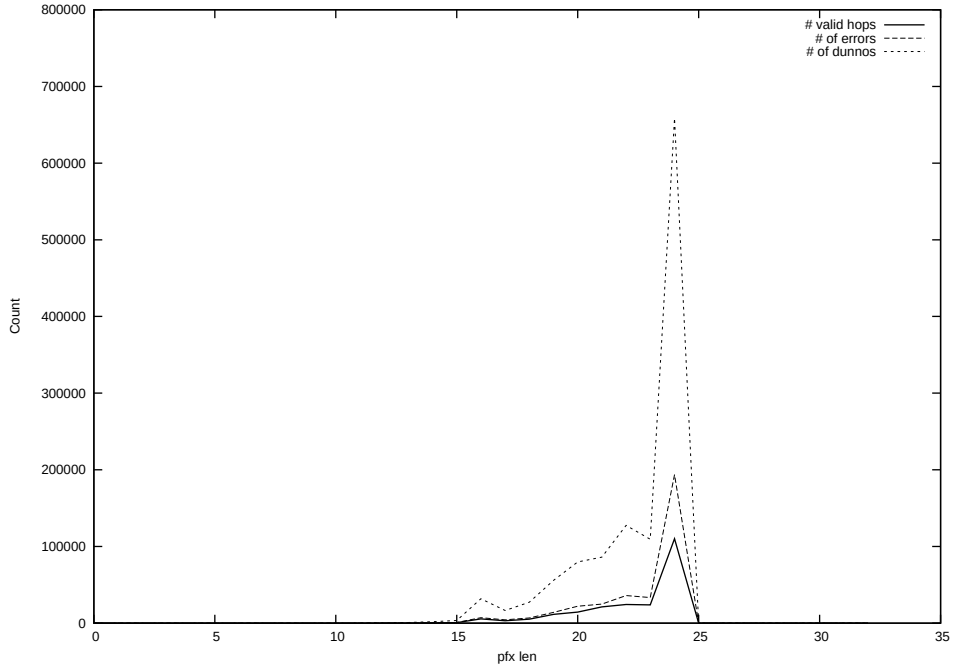
2013-11-21



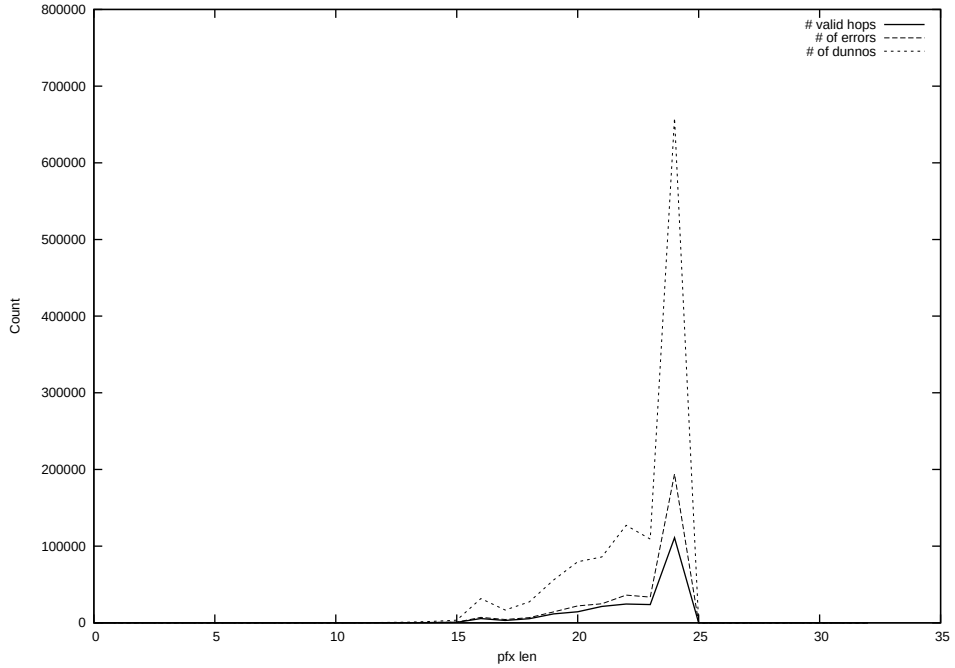
2013-11-22



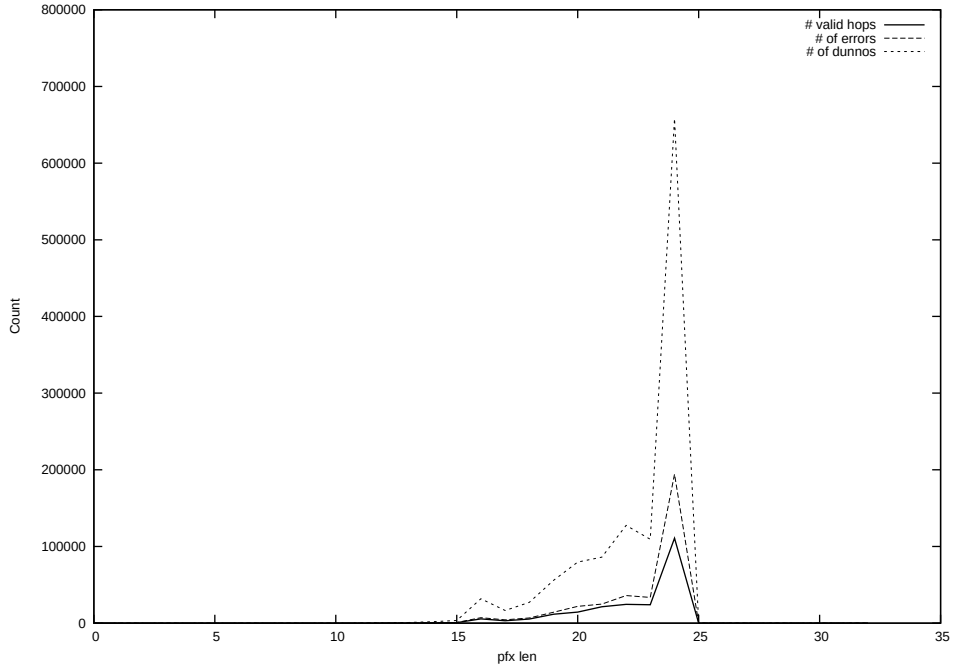
2013-11-23



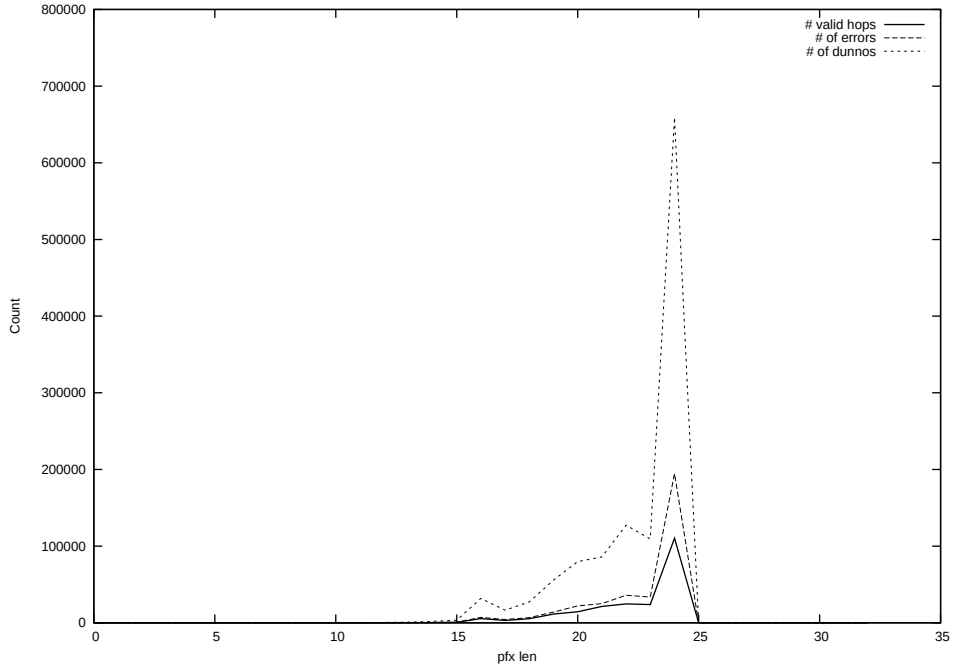
2013-11-24



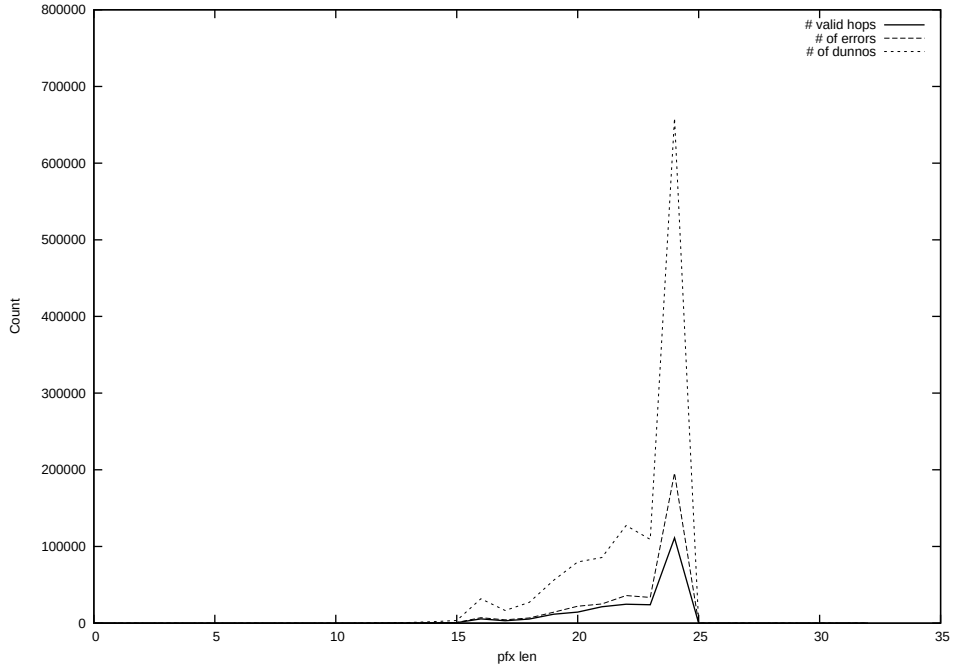
2013-11-25



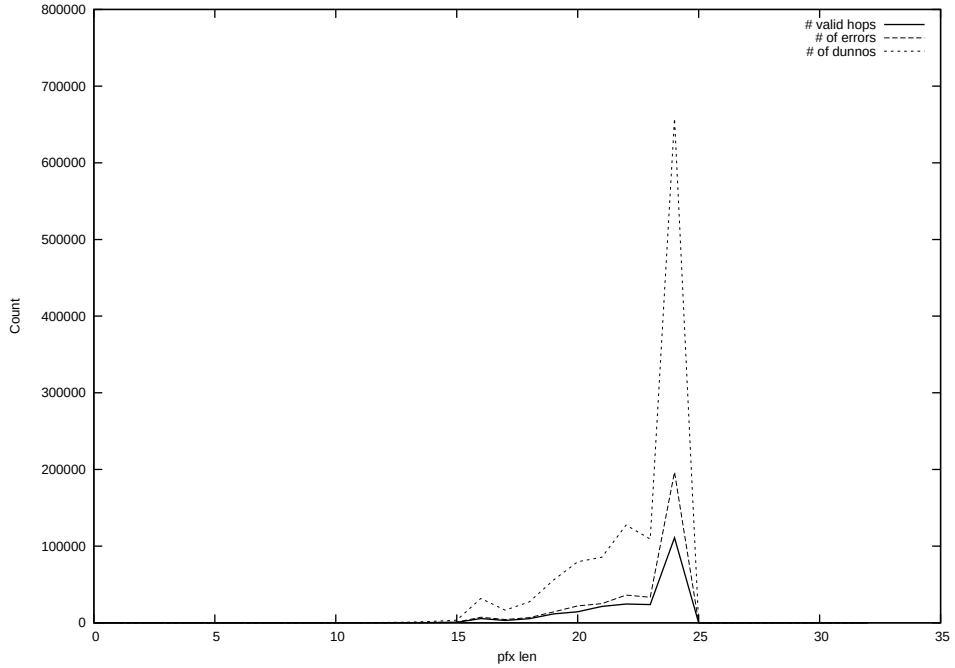
2013-11-26



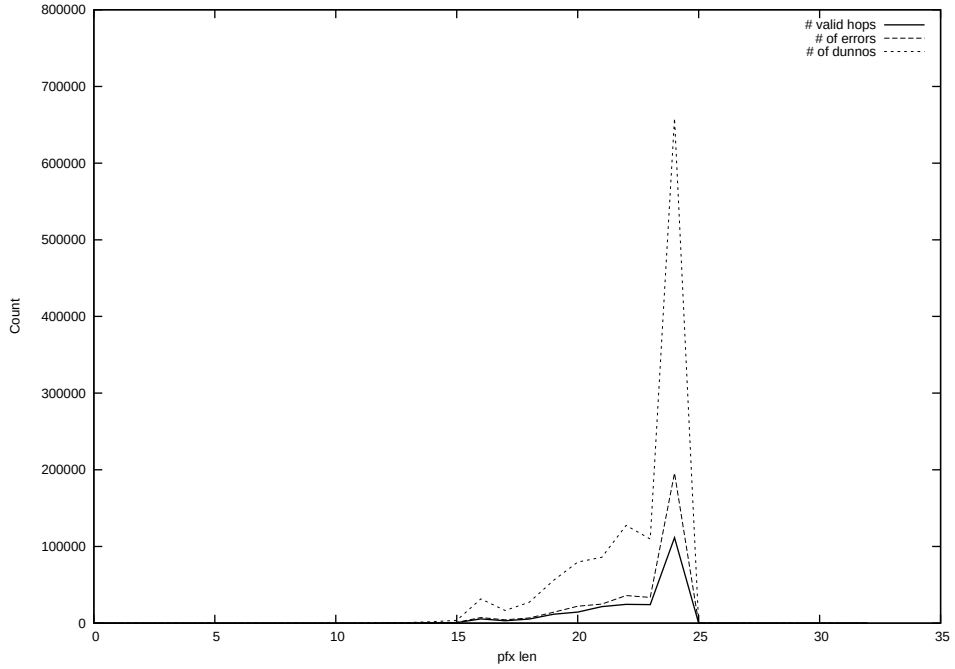
2013-11-27



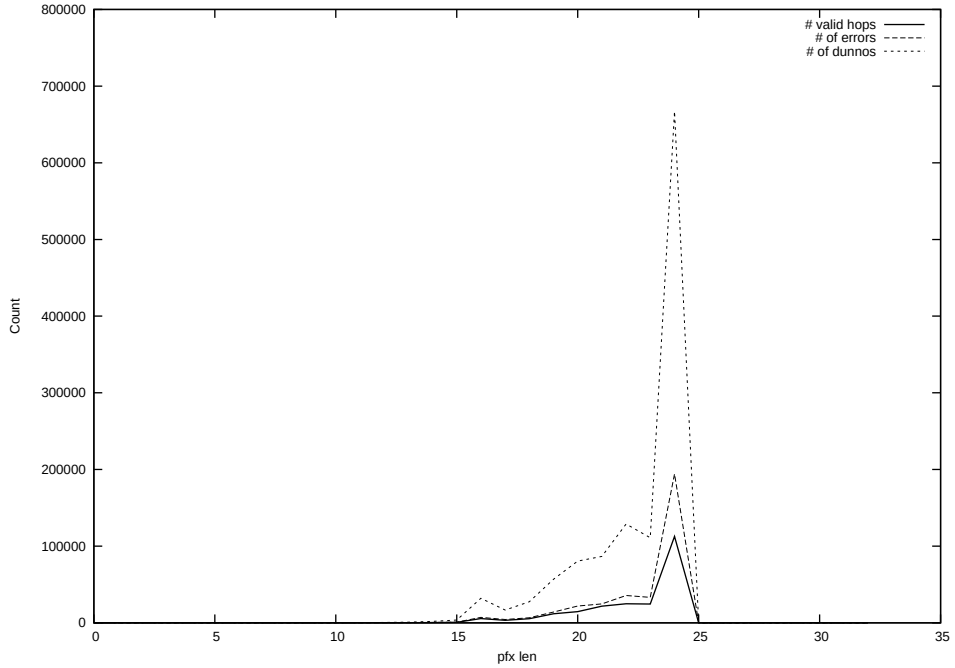
2013-11-28



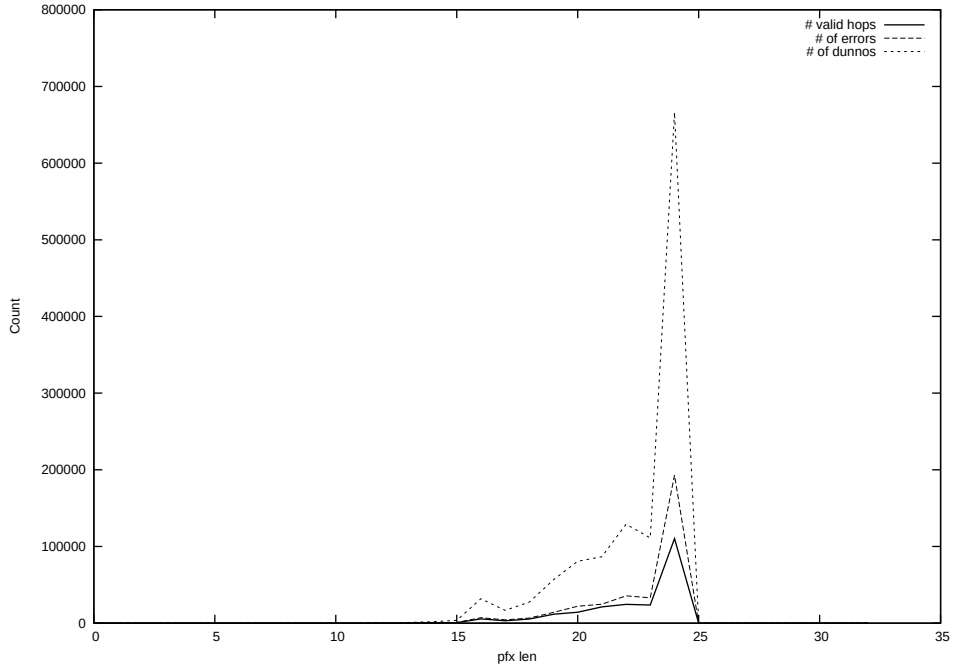
2013-11-29



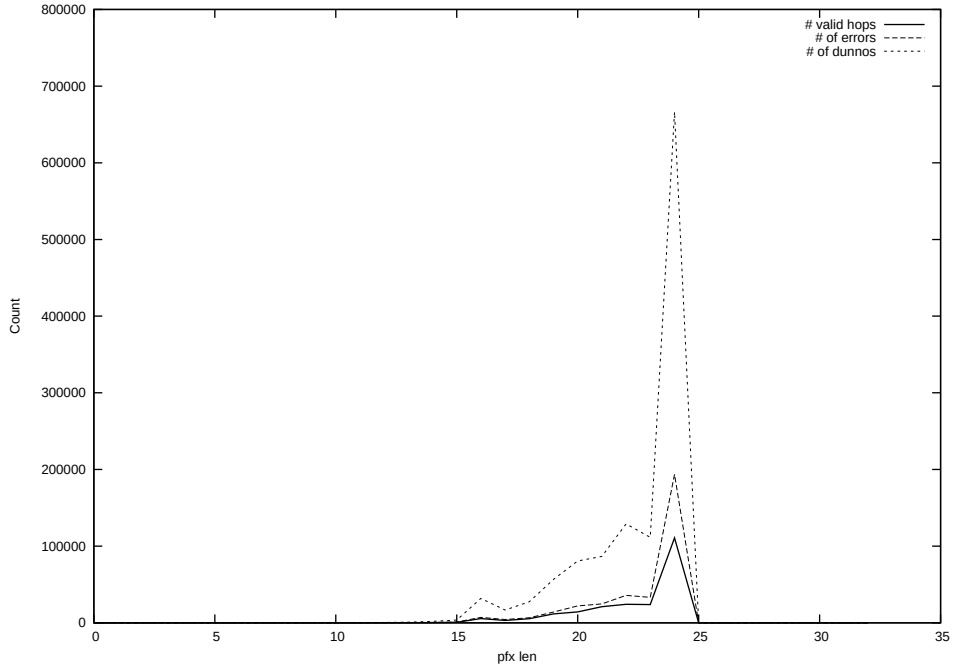
2013-11-30



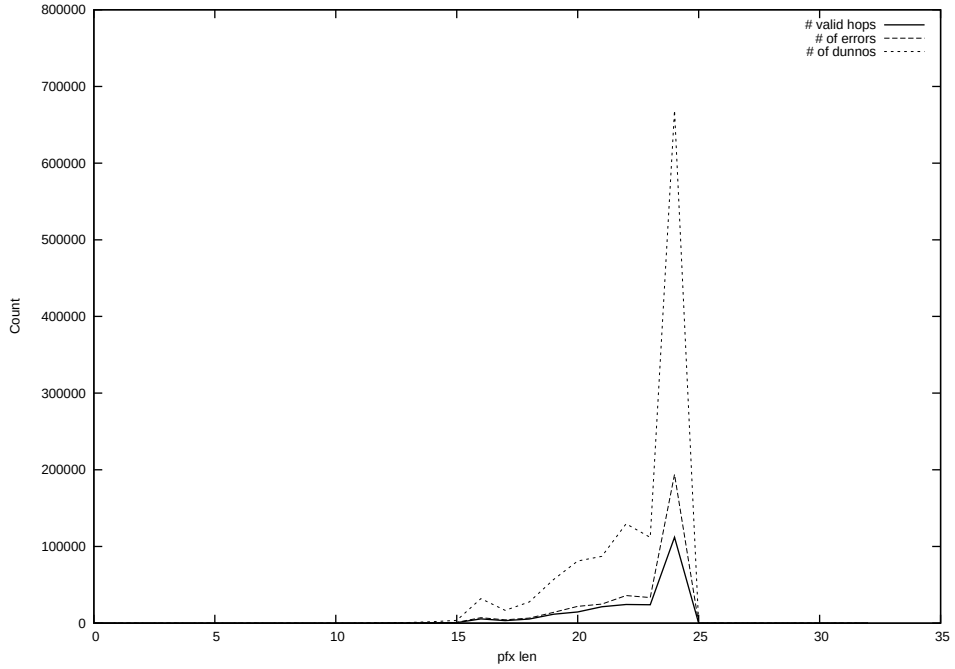
2013-12-01



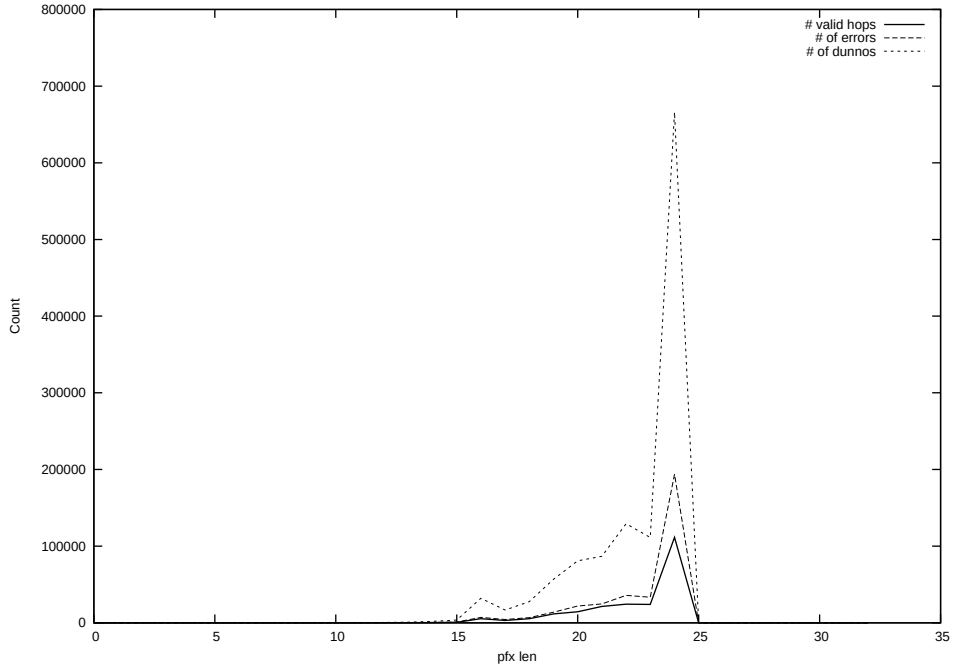
2013-12-02



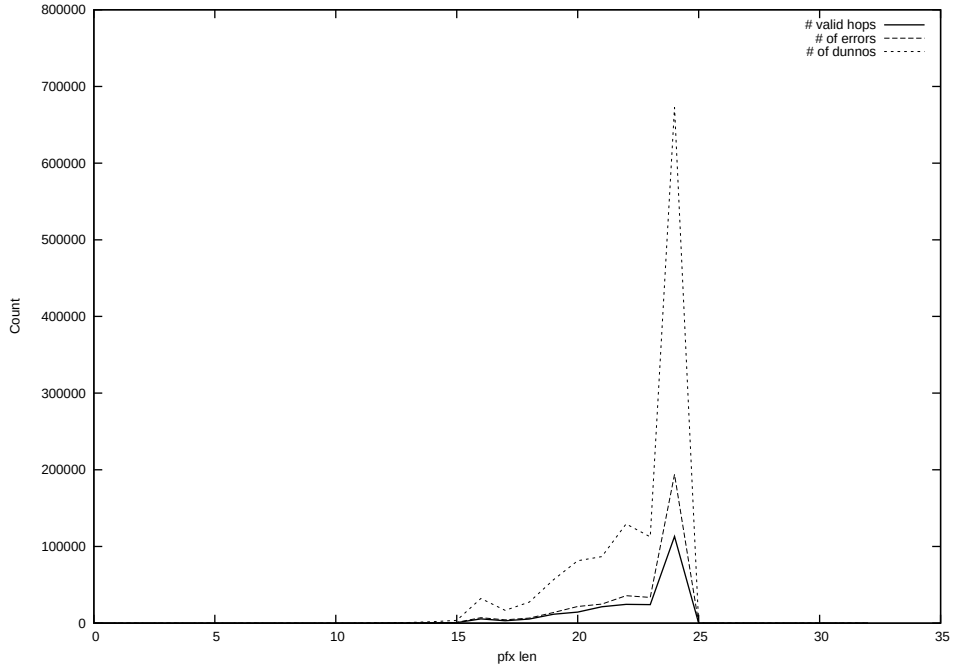
2013-12-03



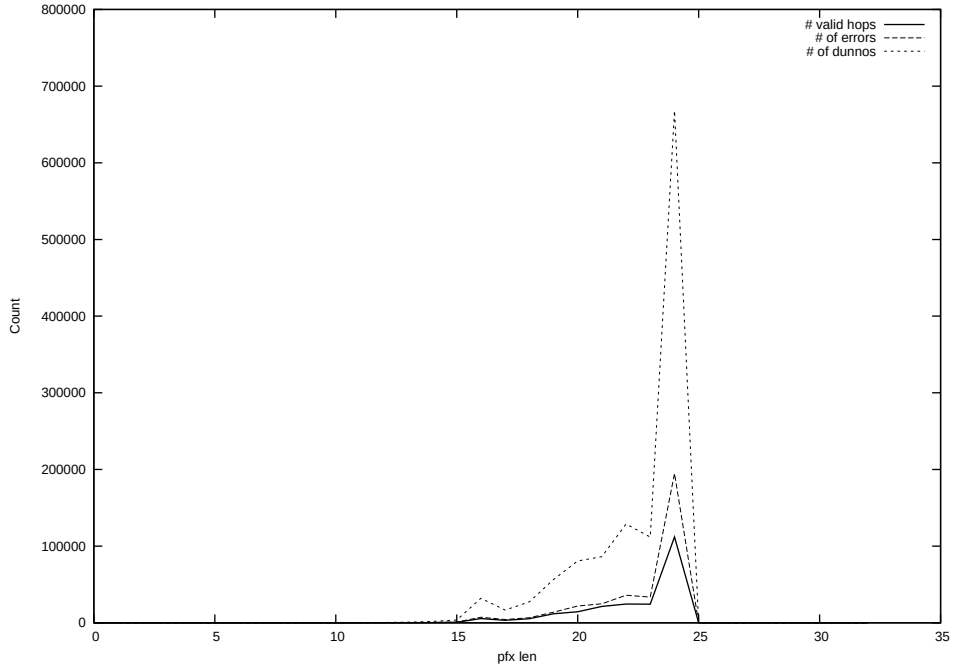
2013-12-04



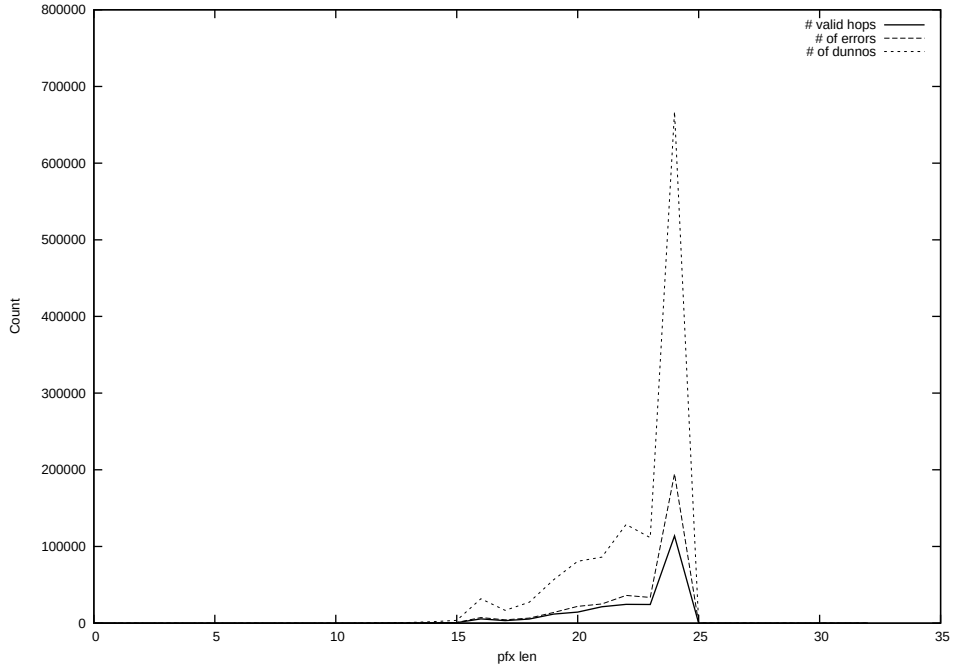
2013-12-05



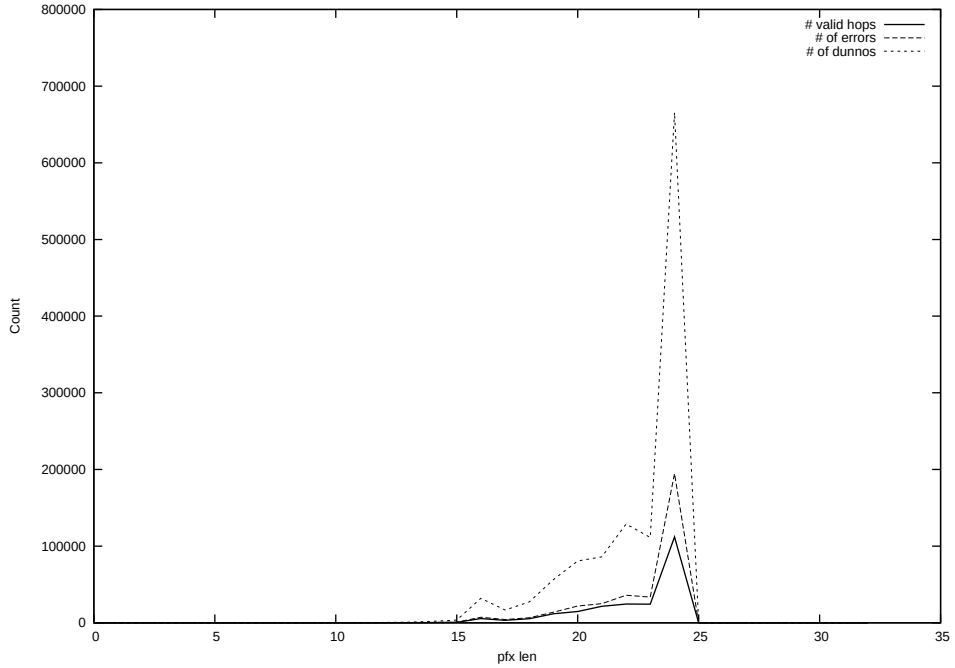
2013-12-06



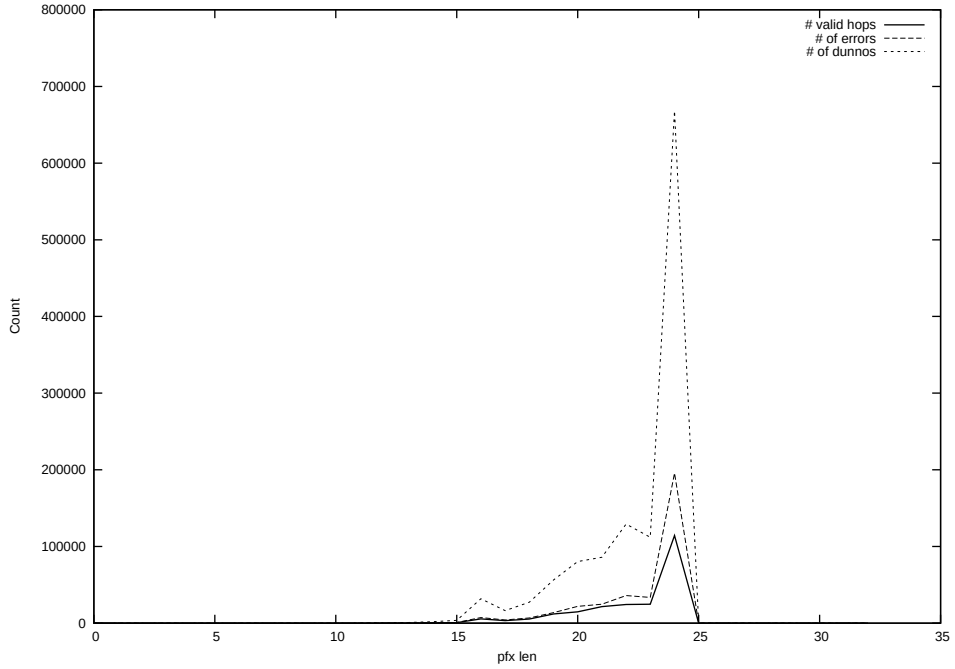
2013-12-07



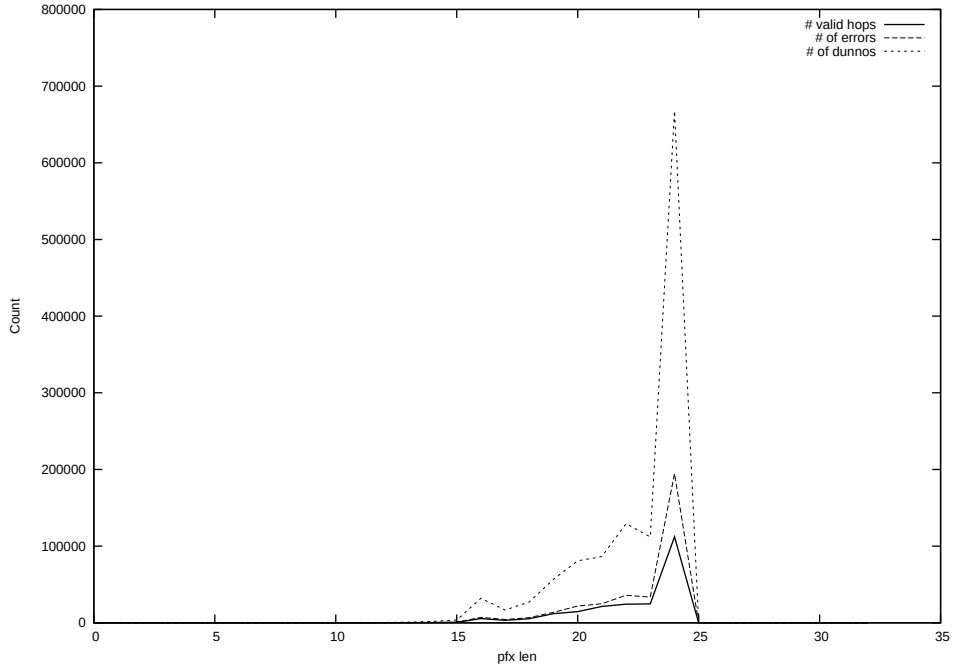
2013-12-08



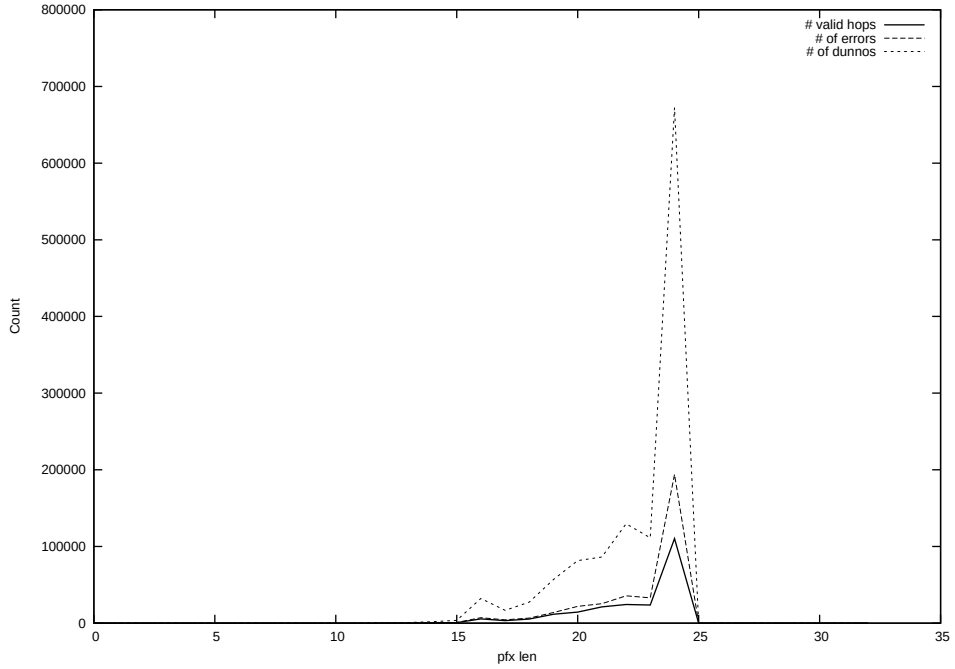
2013-12-09



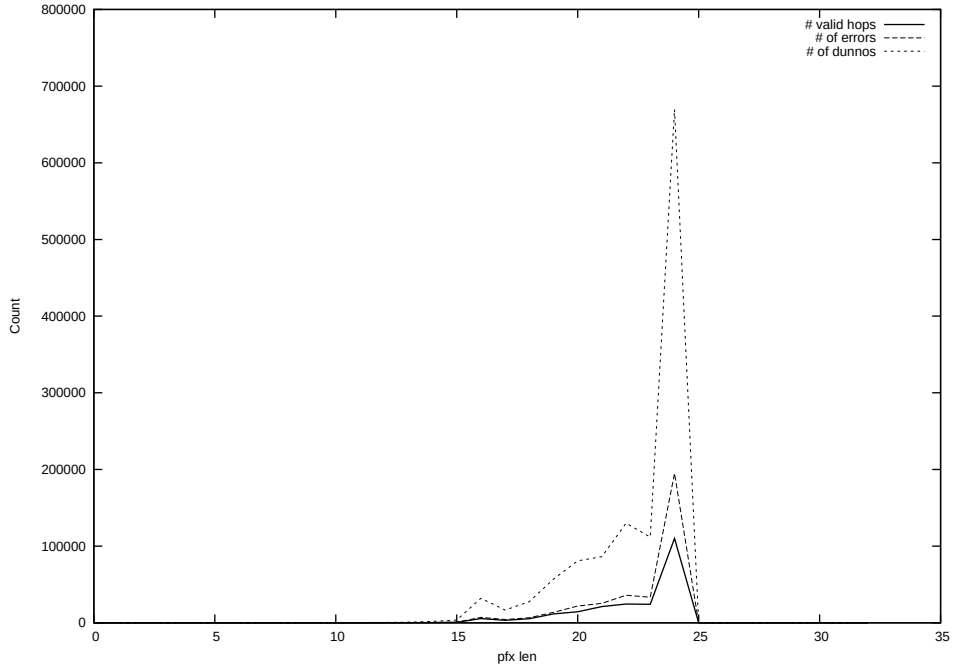
2013-12-10



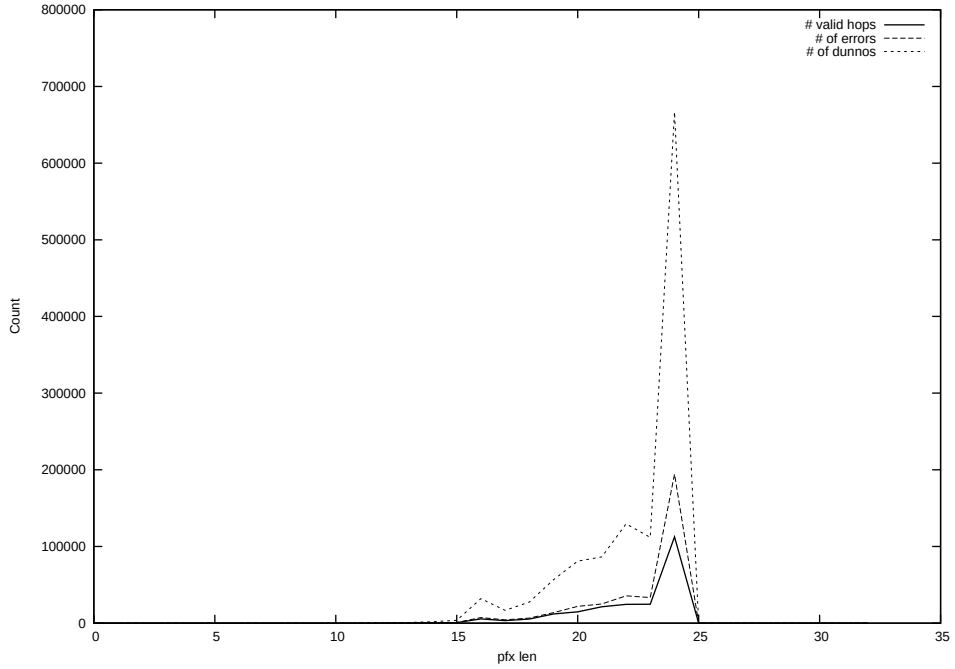
2013-12-11



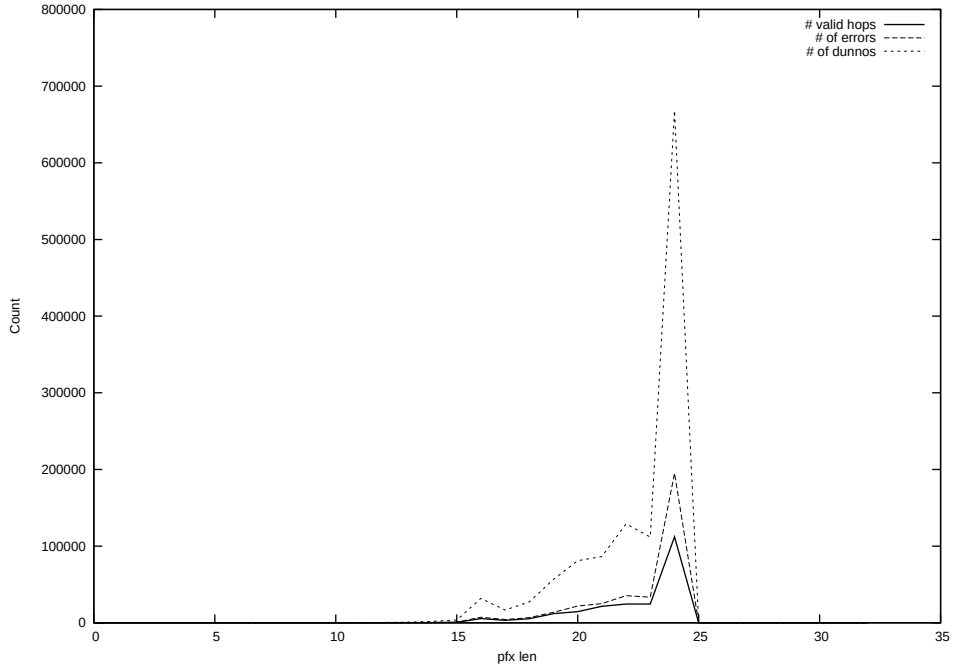
2013-12-12



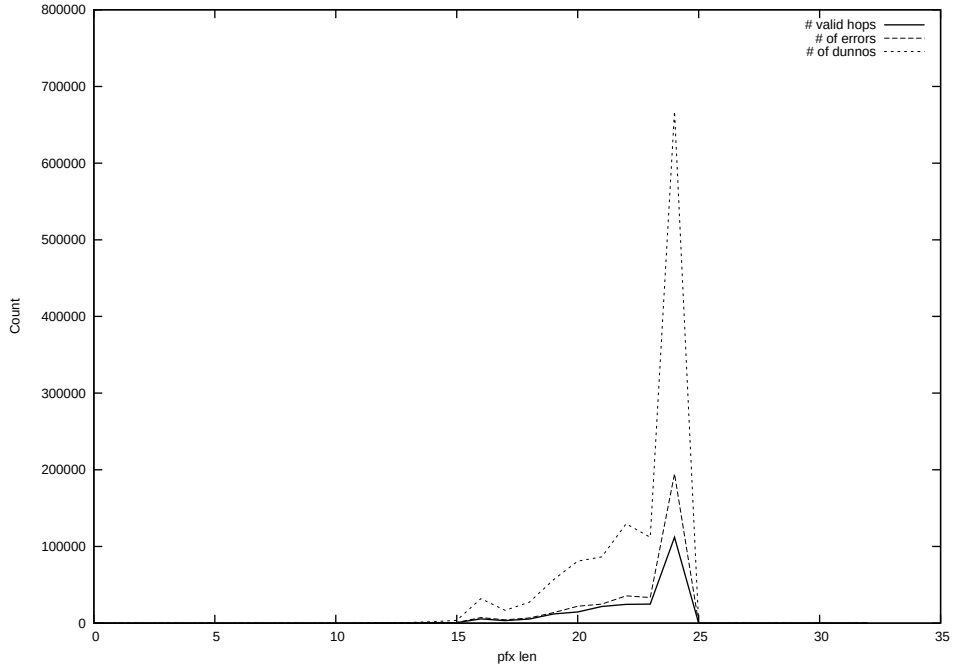
2013-12-13



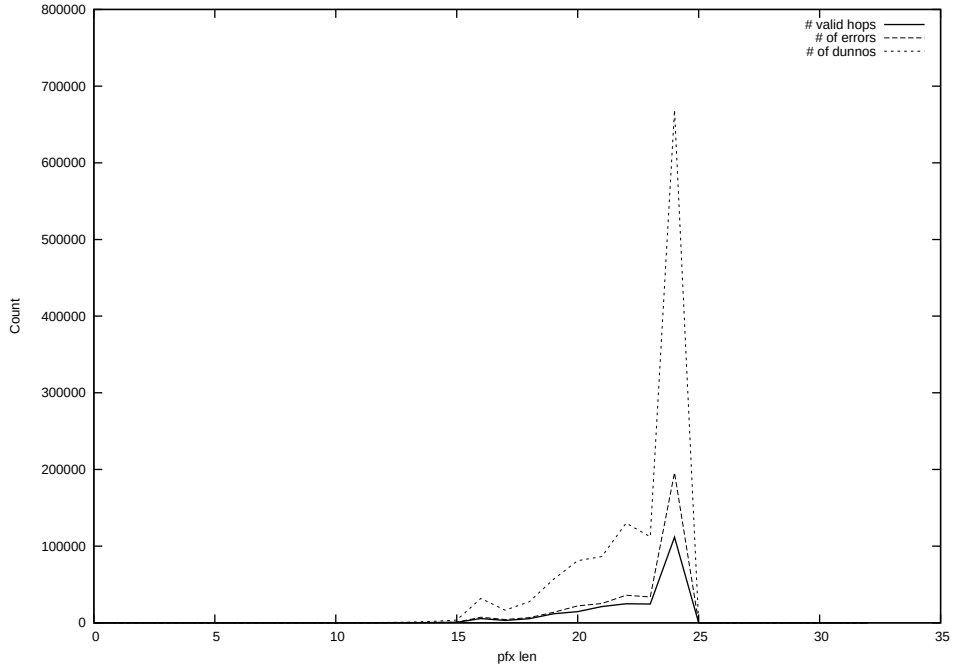
2013-12-14



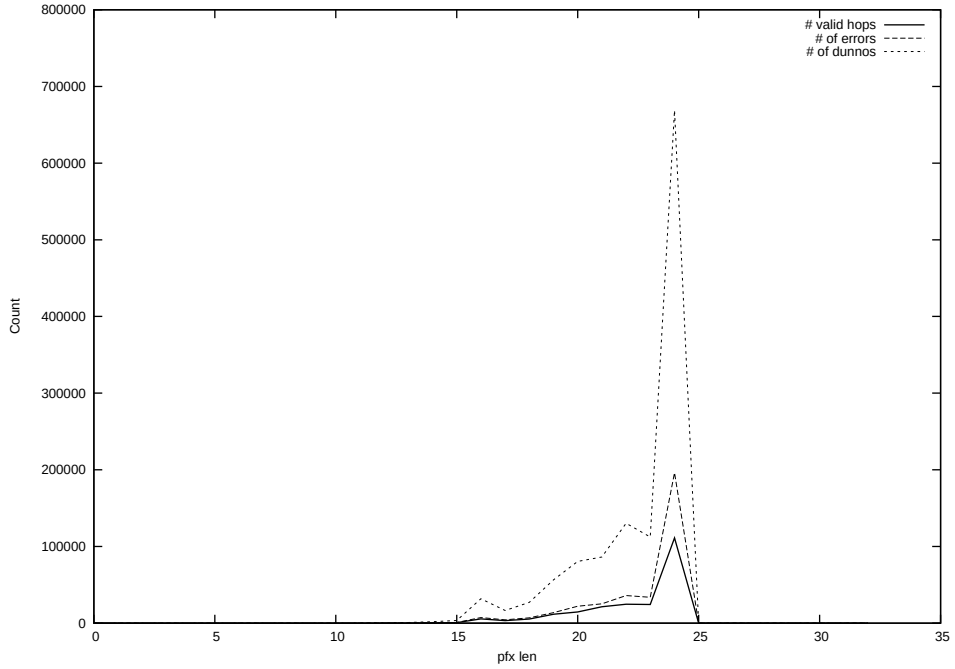
2013-12-15



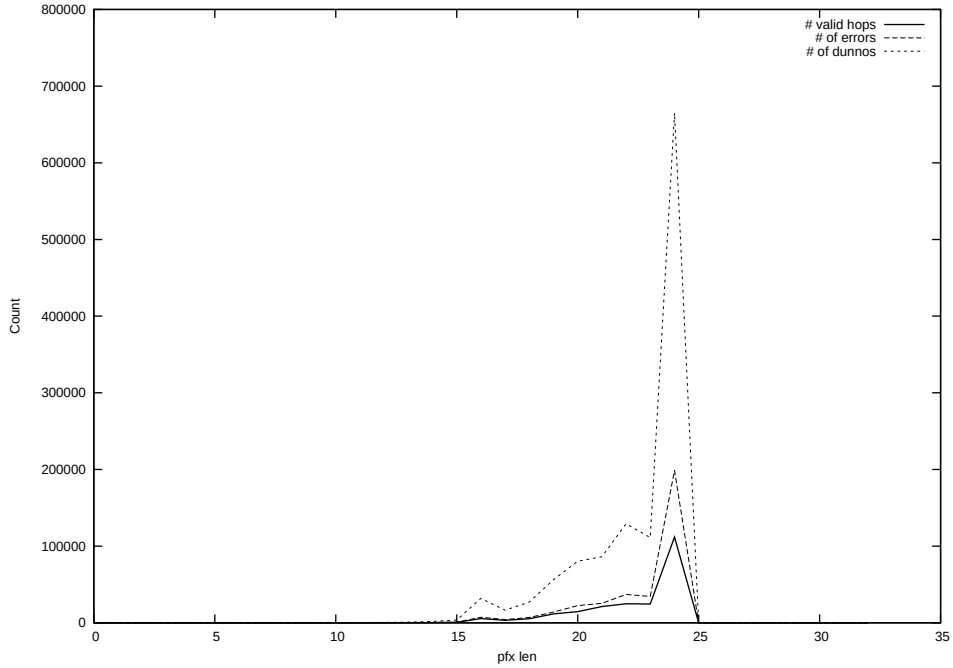
2013-12-16



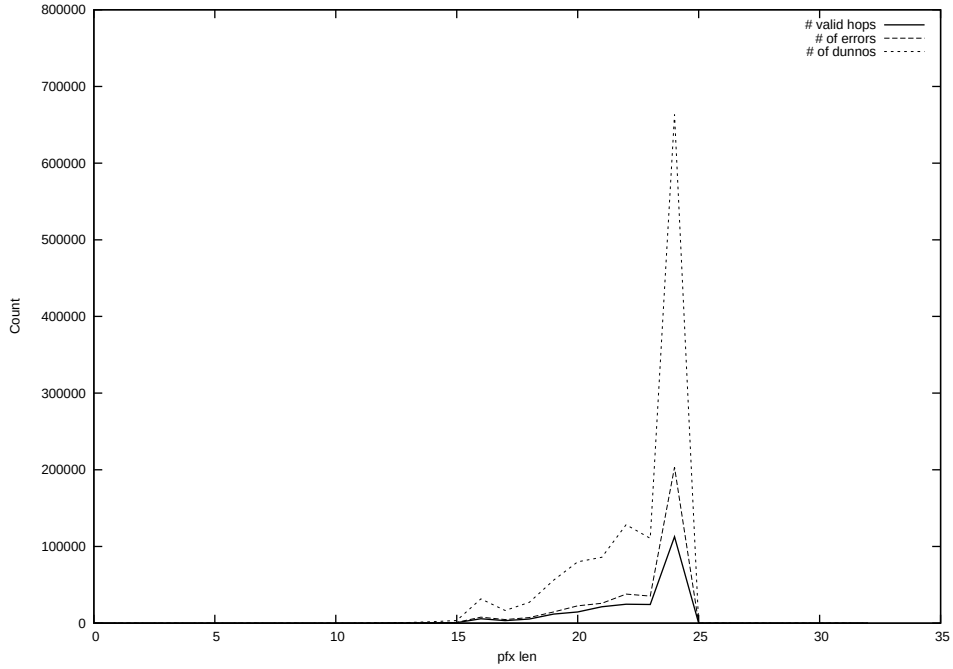
2013-12-17



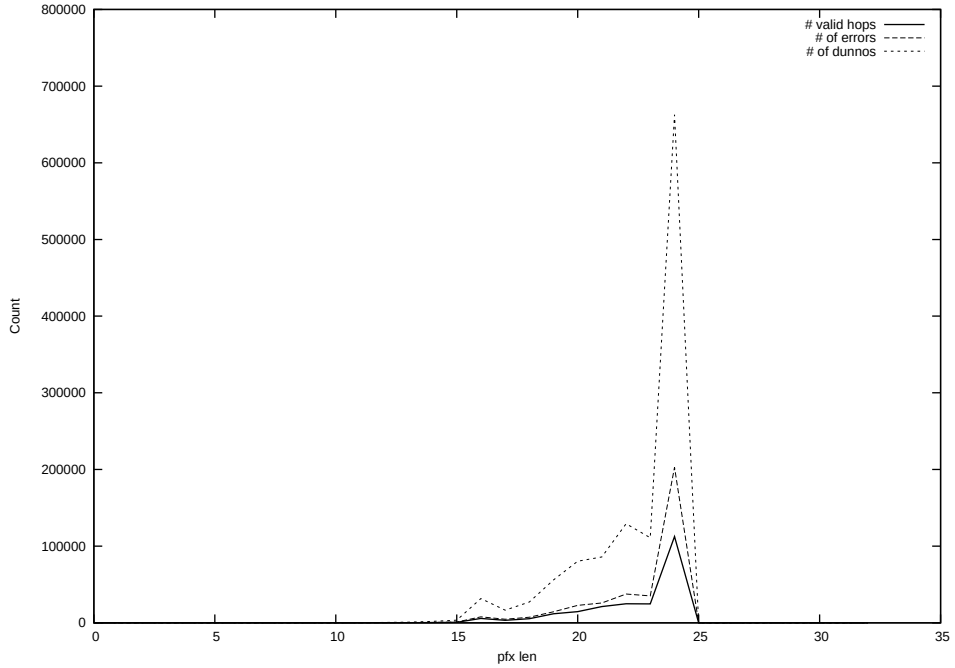
2013-12-18



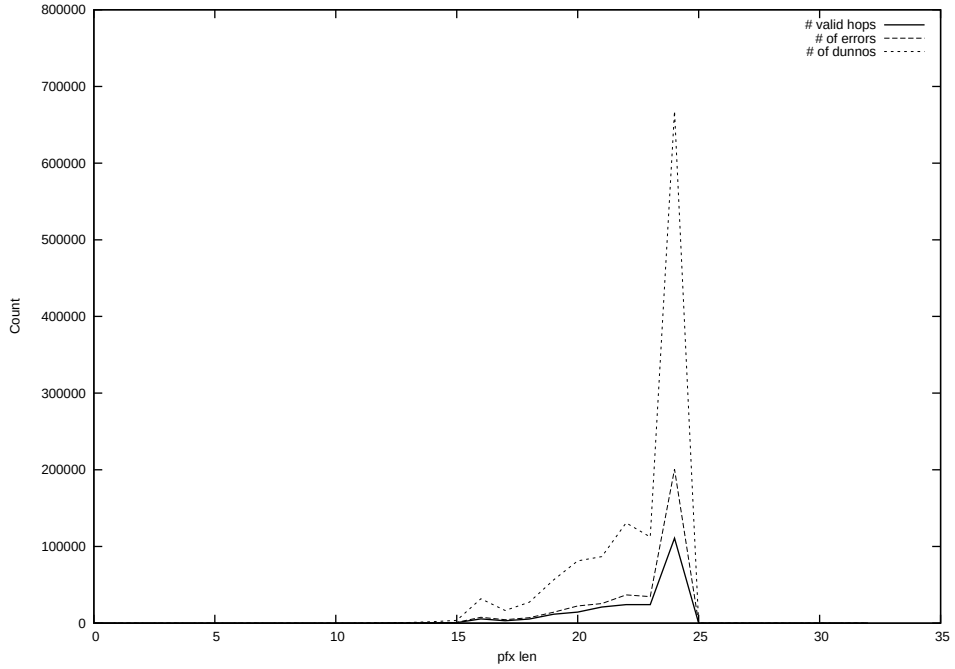
2013-12-19



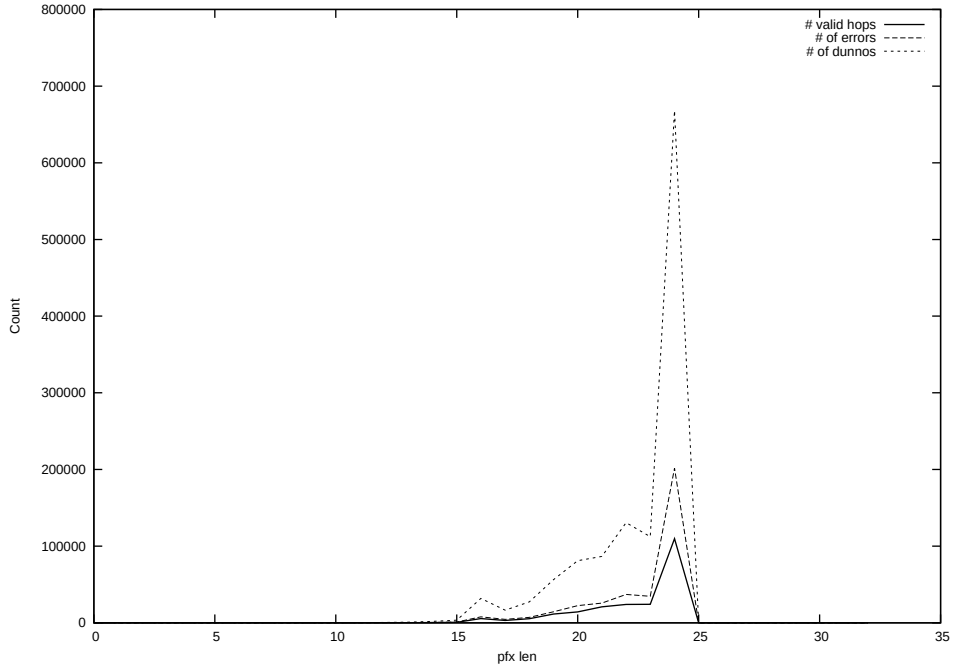
2013-12-20



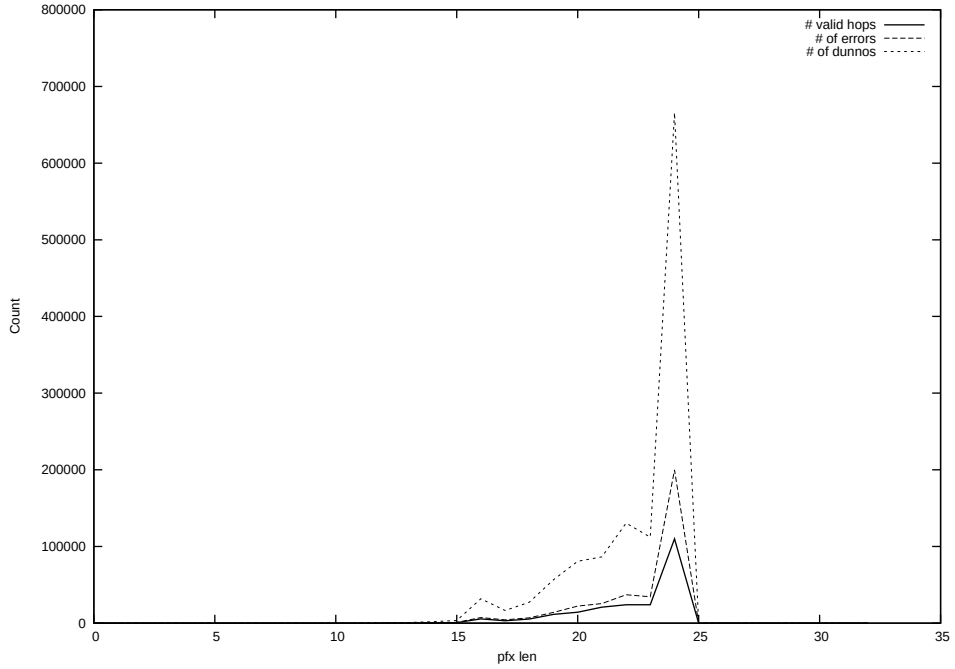
2013-12-21



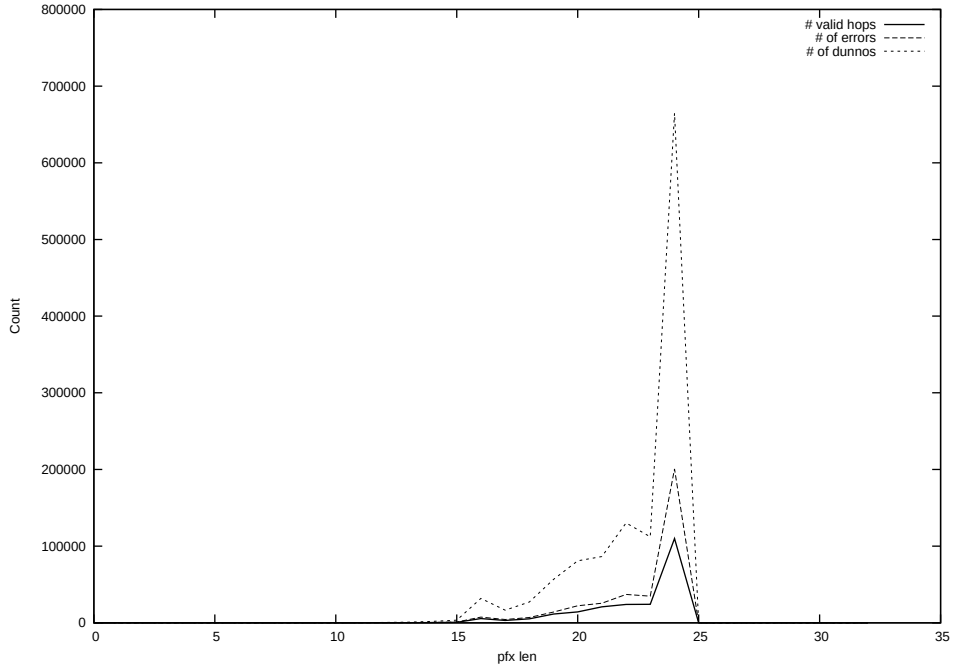
2013-12-22



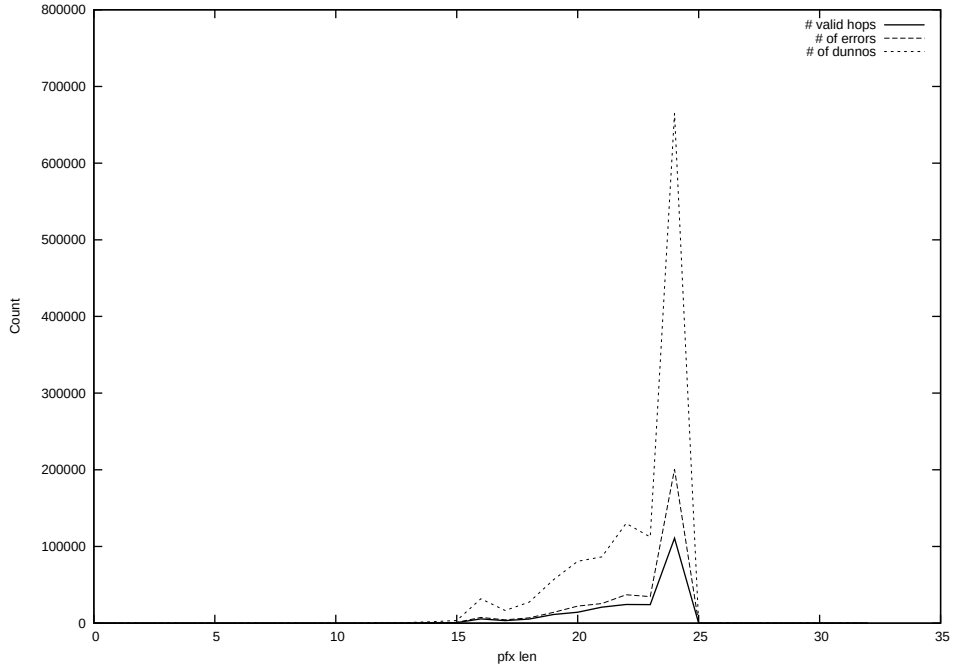
2013-12-23



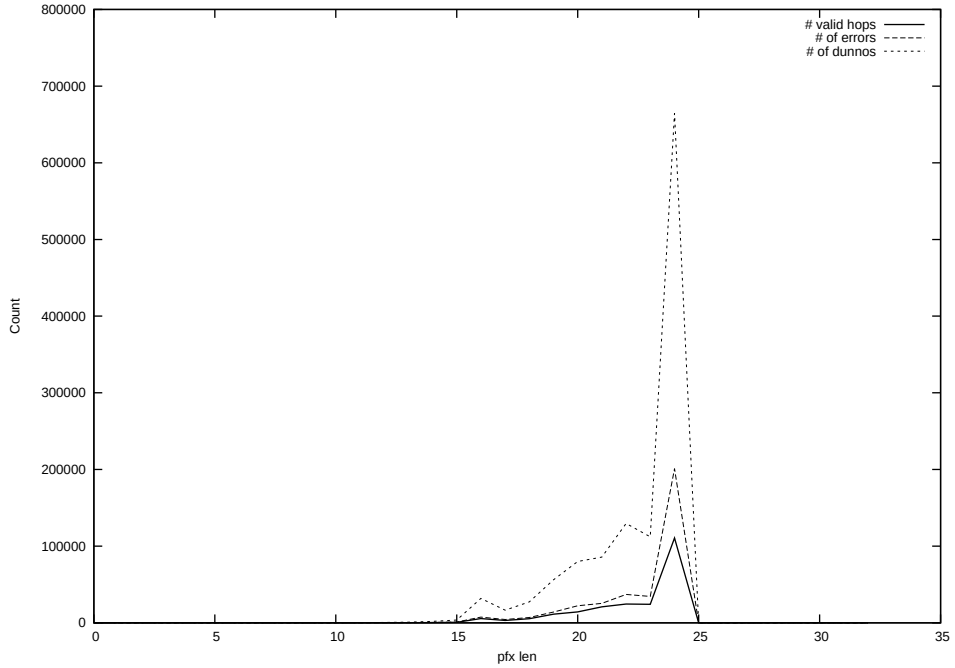
2013-12-24



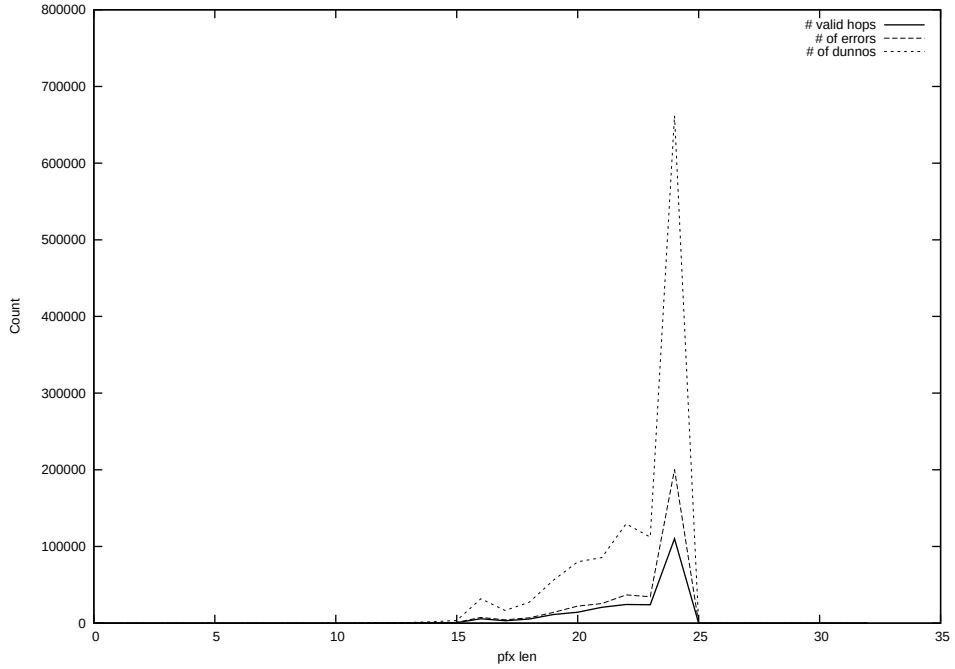
2013-12-25



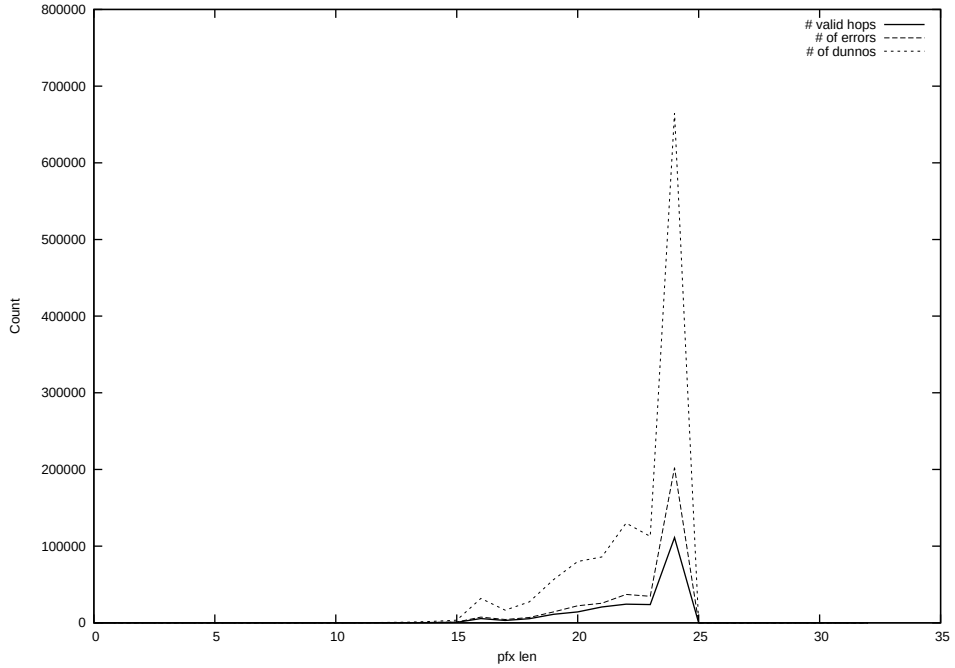
2013-12-26



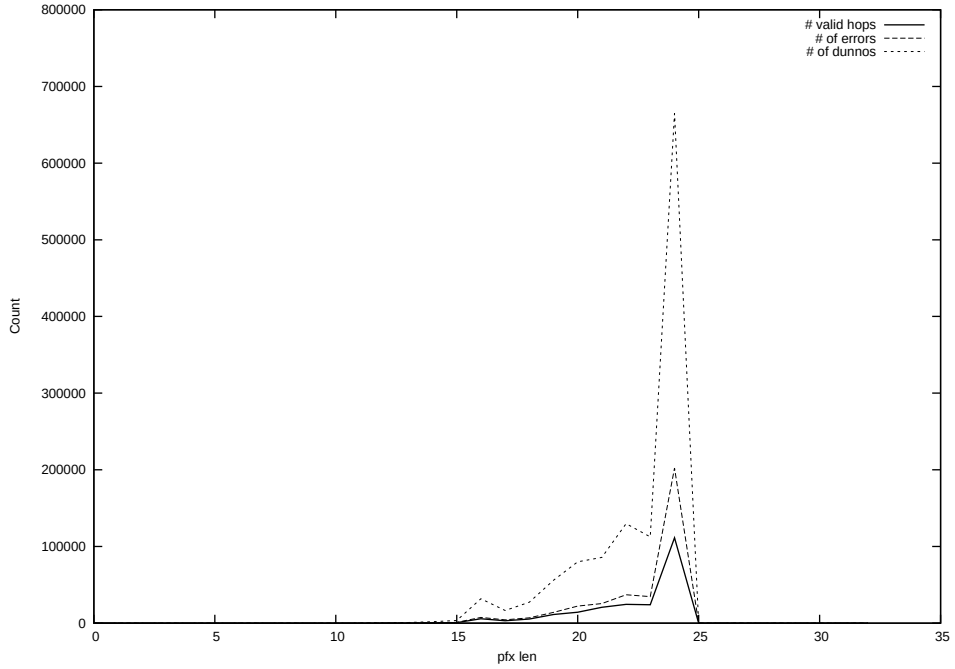
2013-12-27



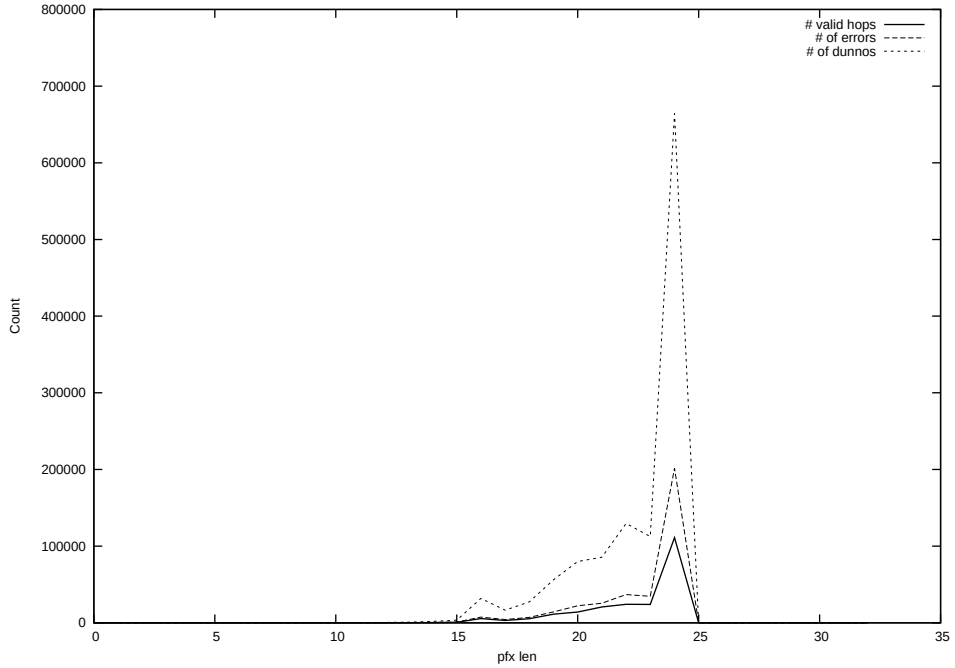
2013-12-28



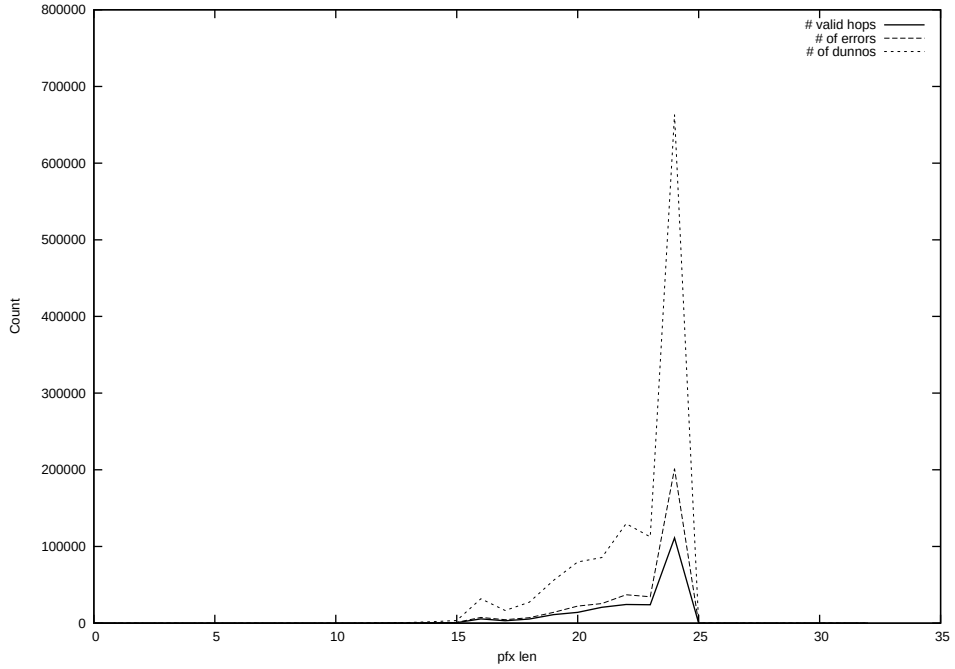
2013-12-29



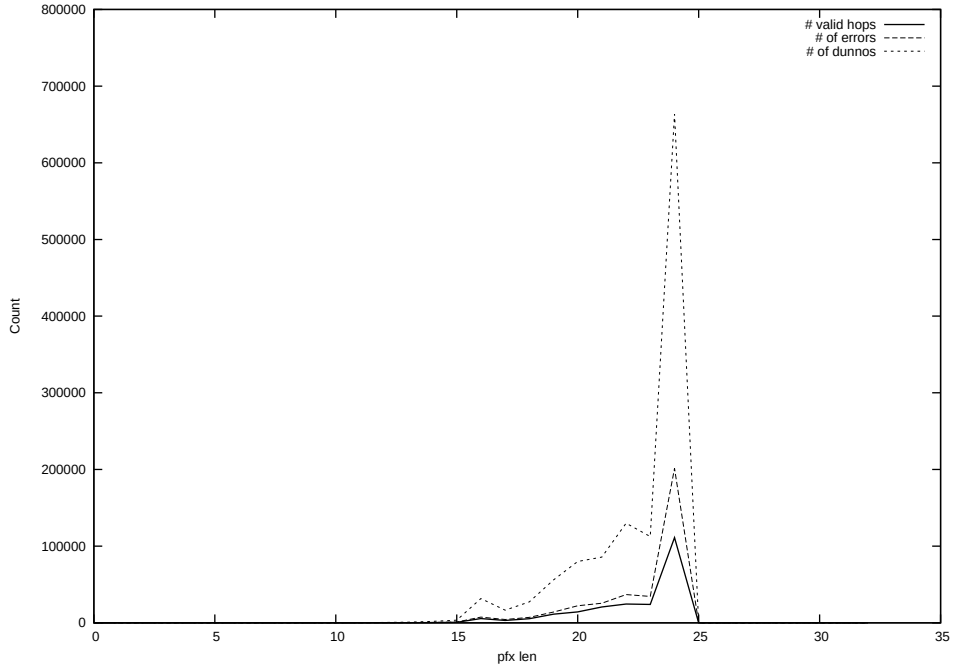
2013-12-30



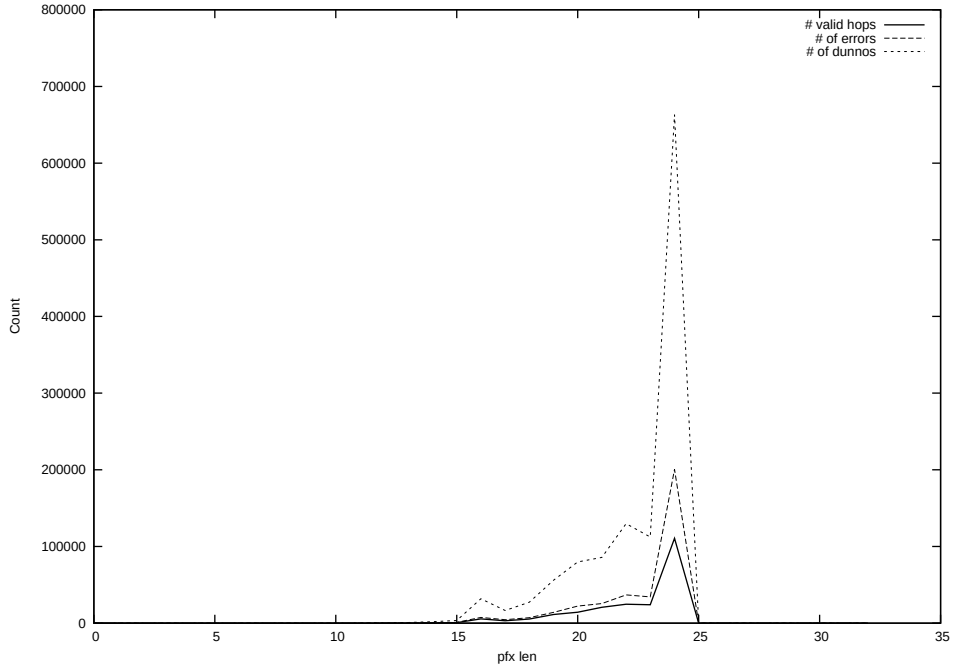
2013-12-31



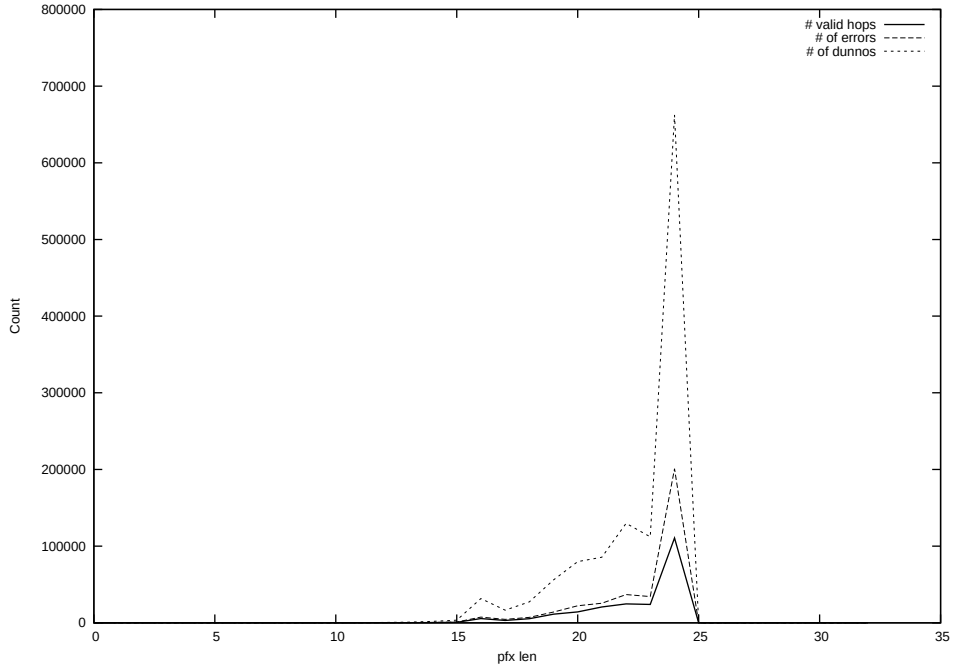
2014-01-01



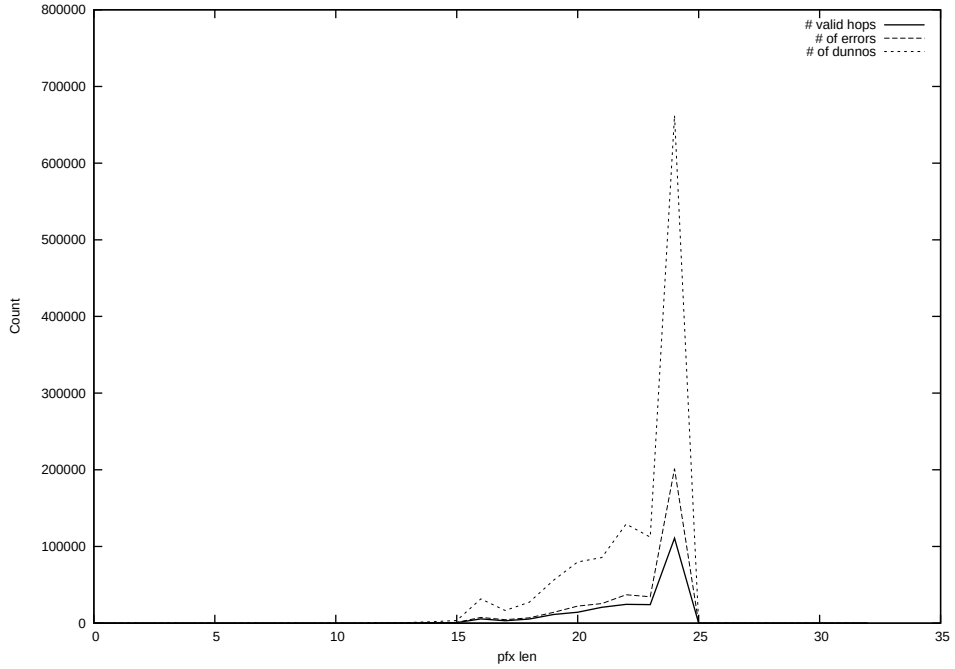
2014-01-02



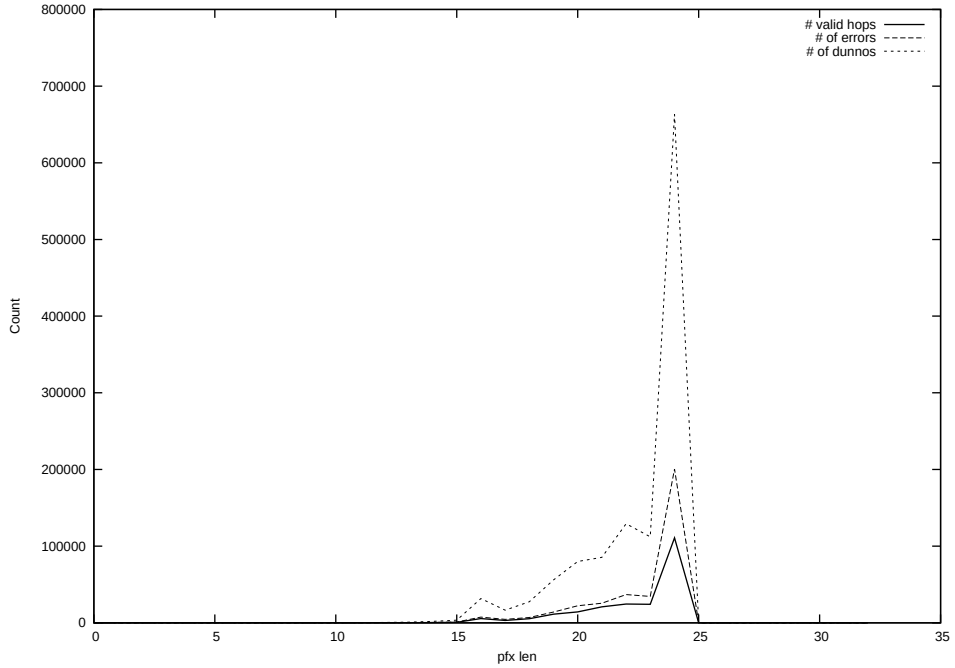
2014-01-03



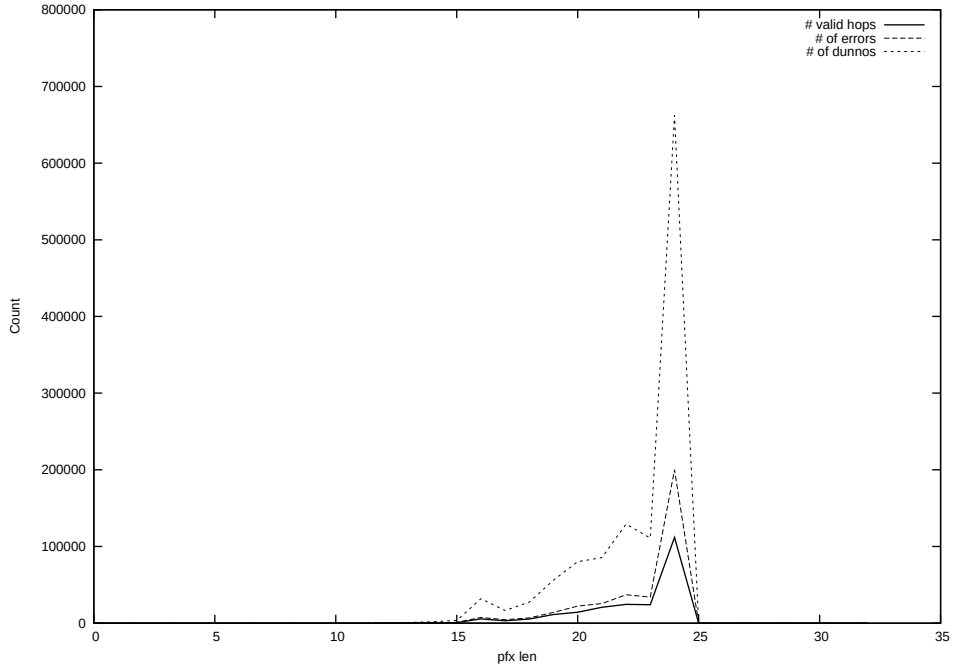
2014-01-04



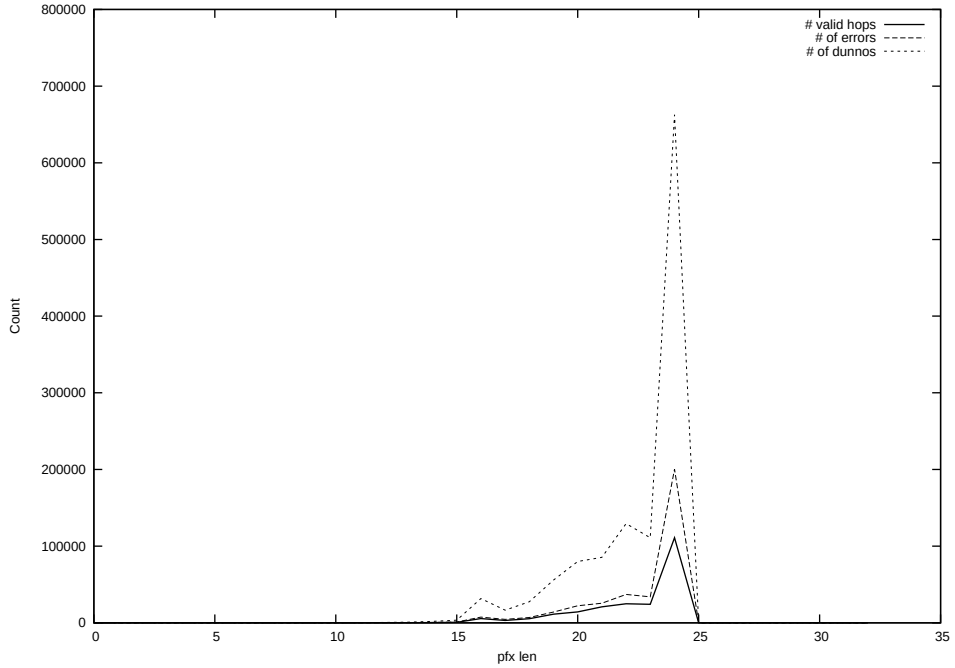
2014-01-05



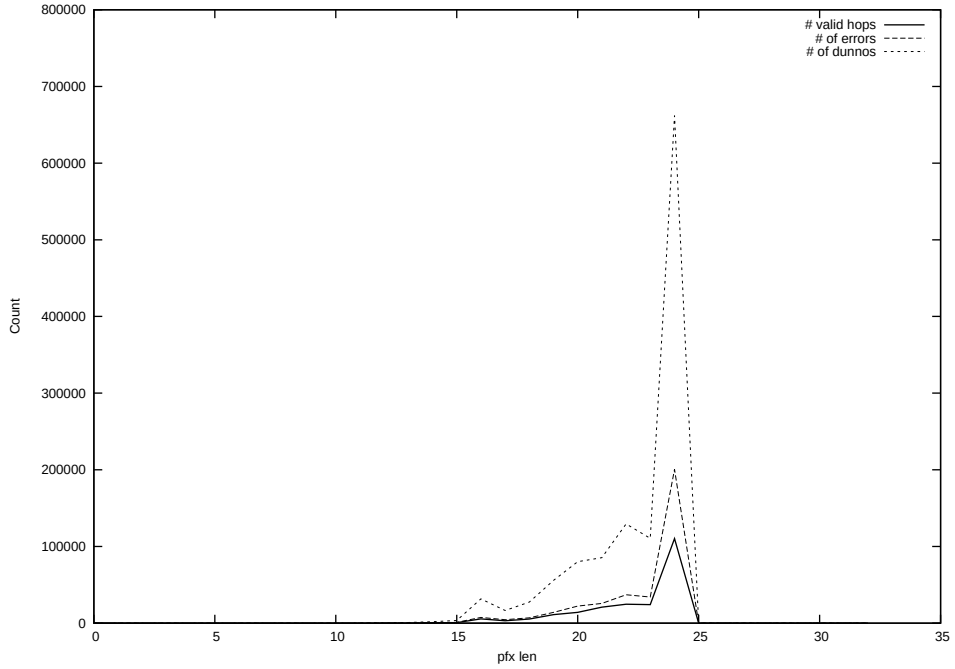
2014-01-06



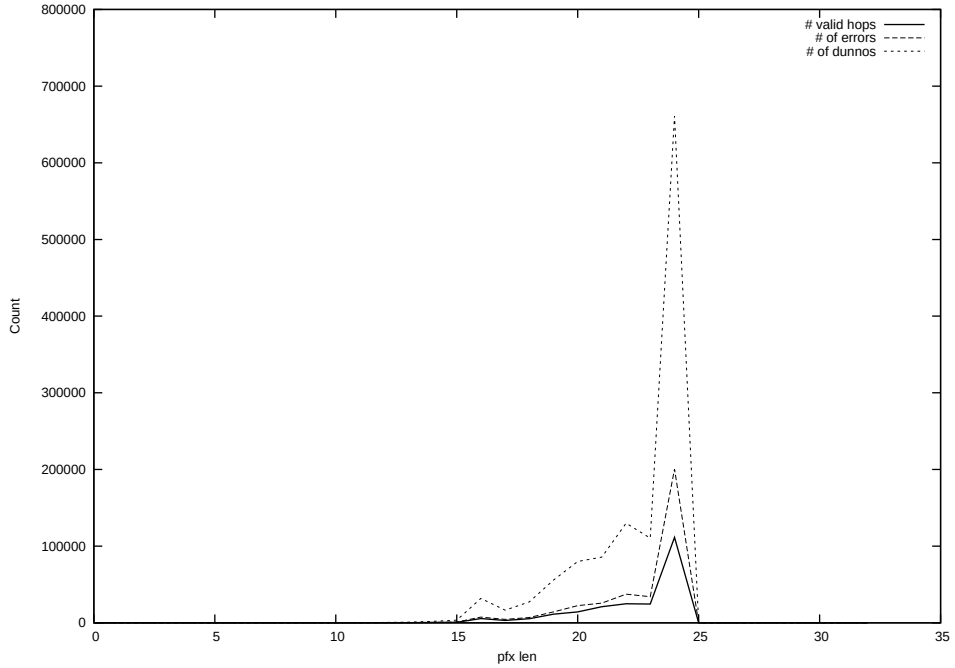
2014-01-07



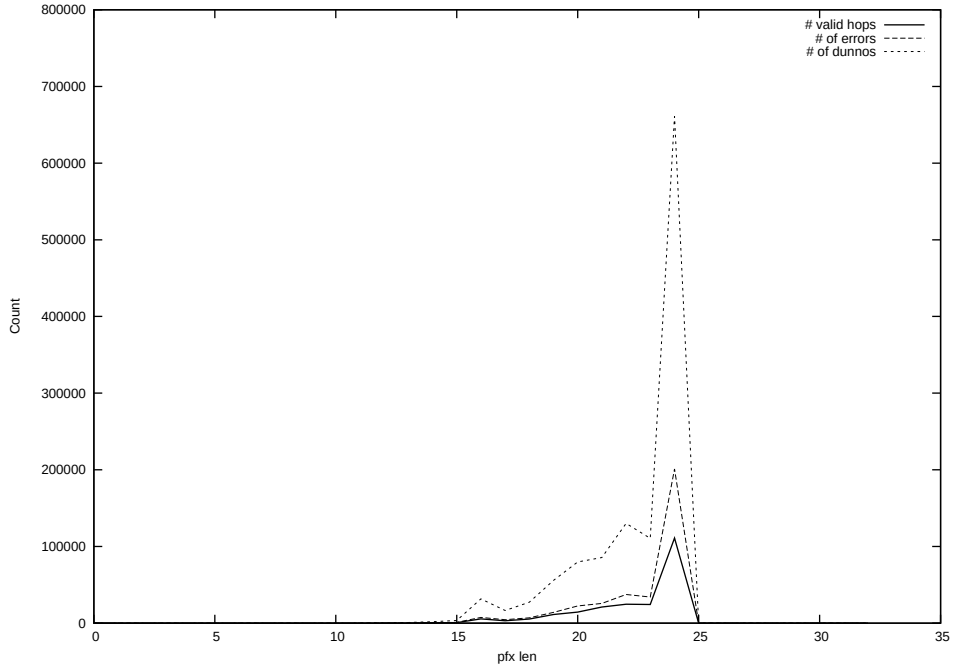
2014-01-08



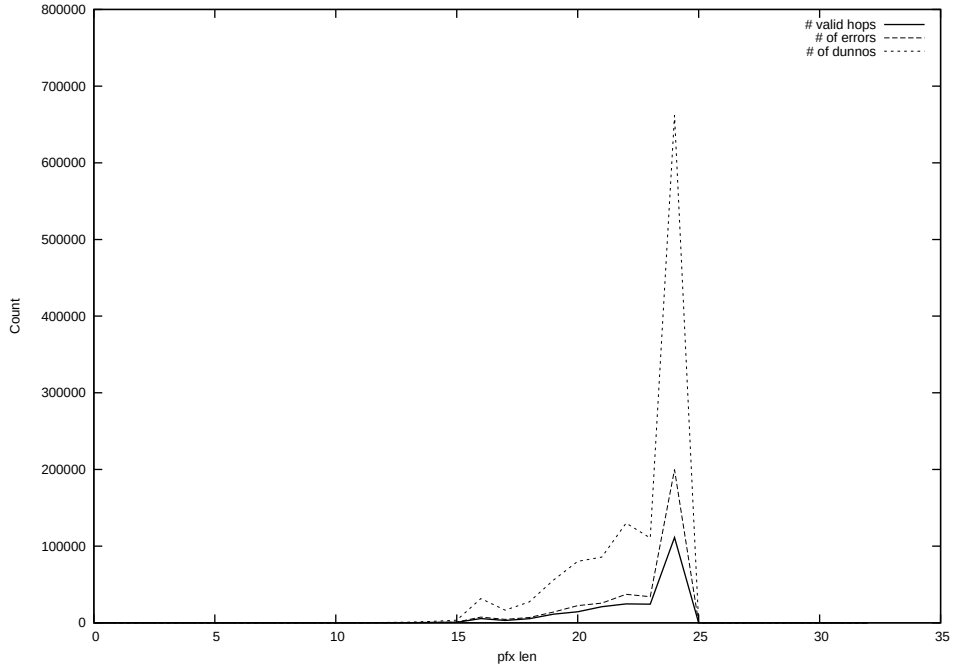
2014-01-09



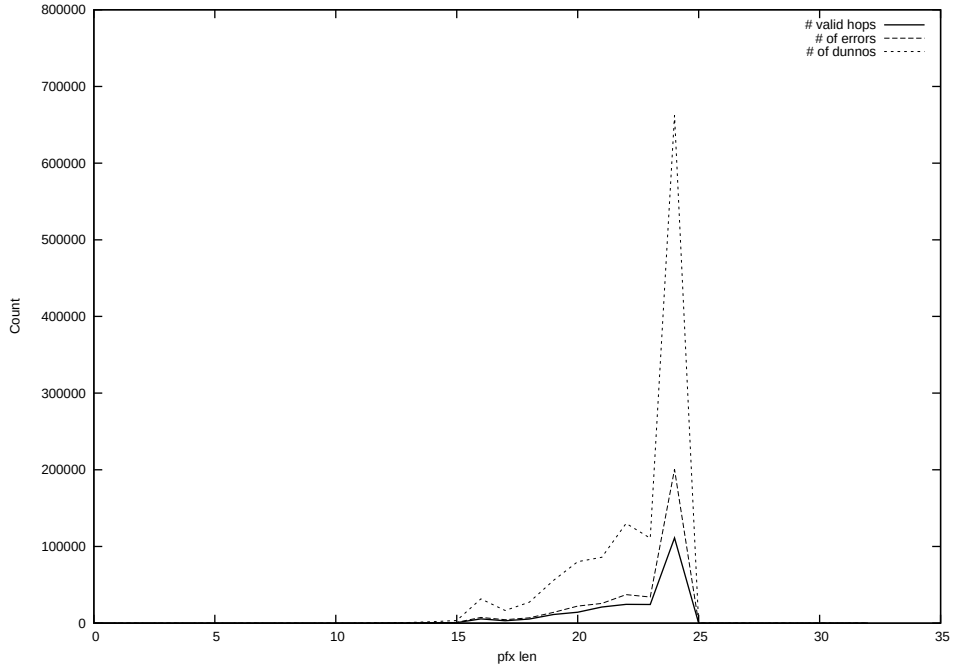
2014-01-10



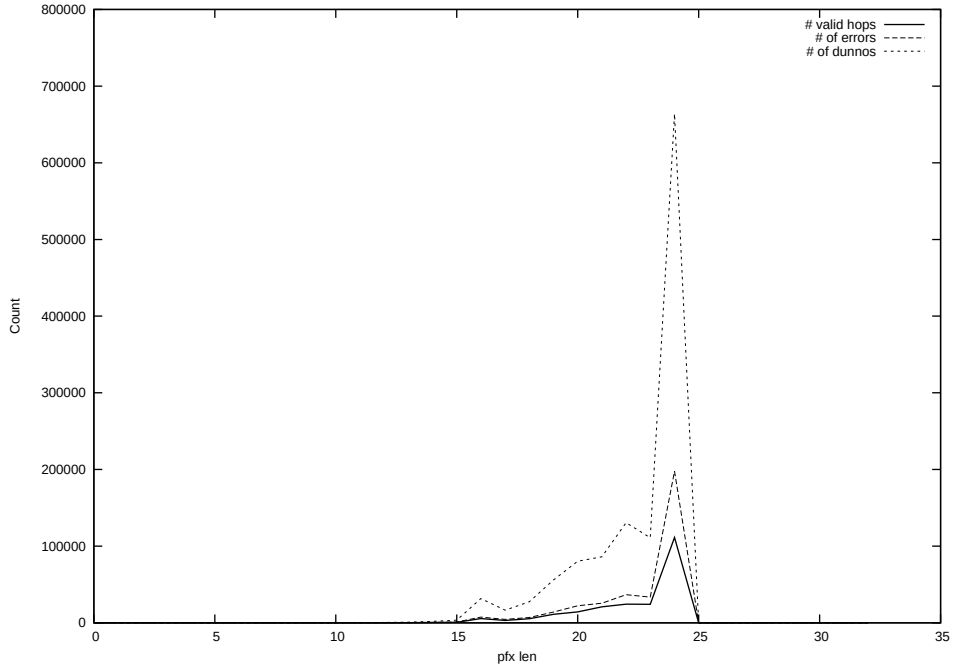
2014-01-11



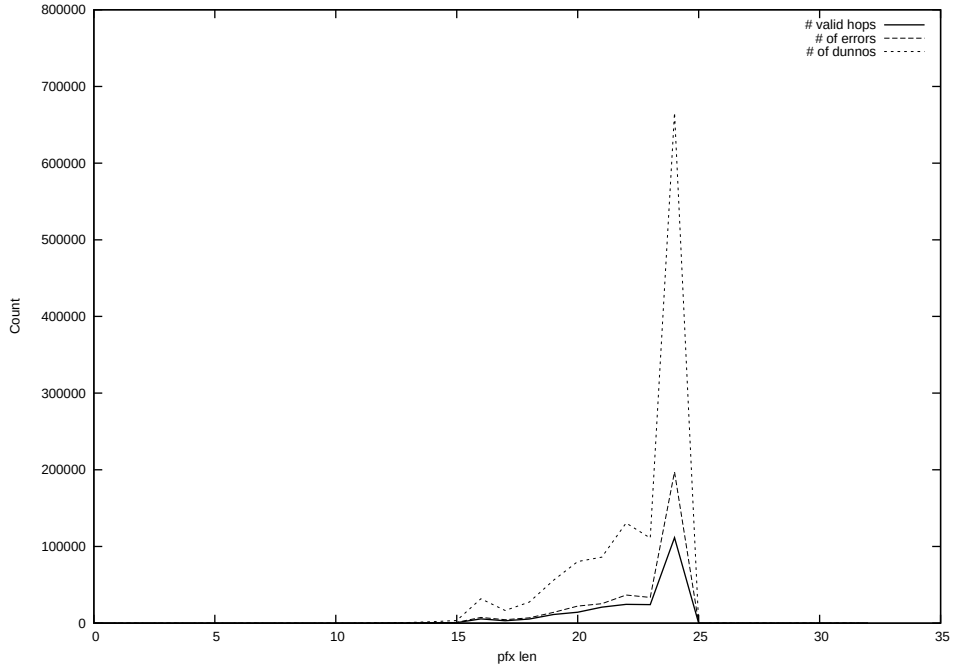
2014-01-12



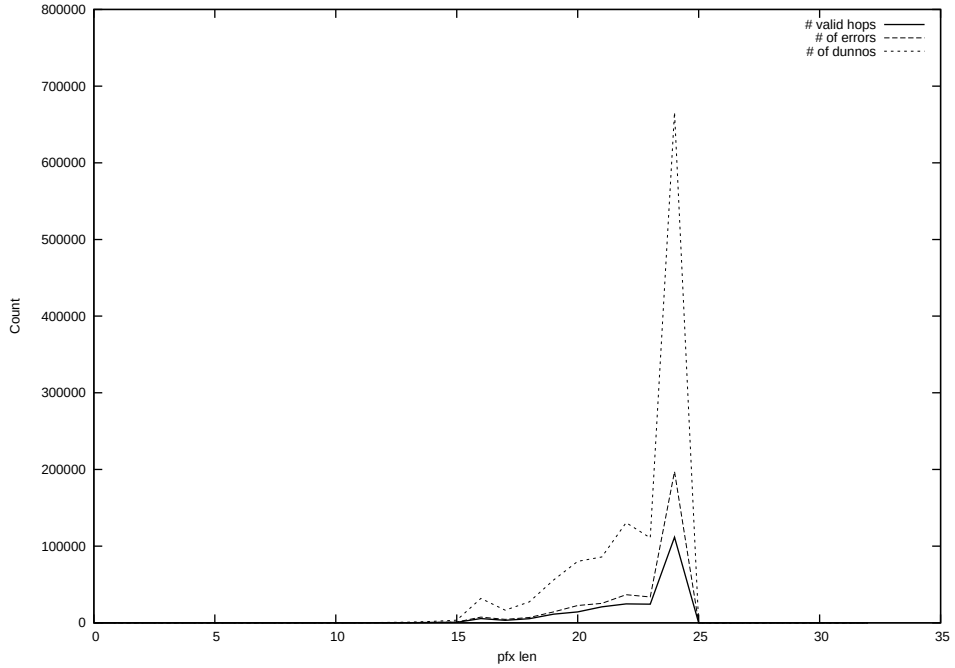
2014-01-13



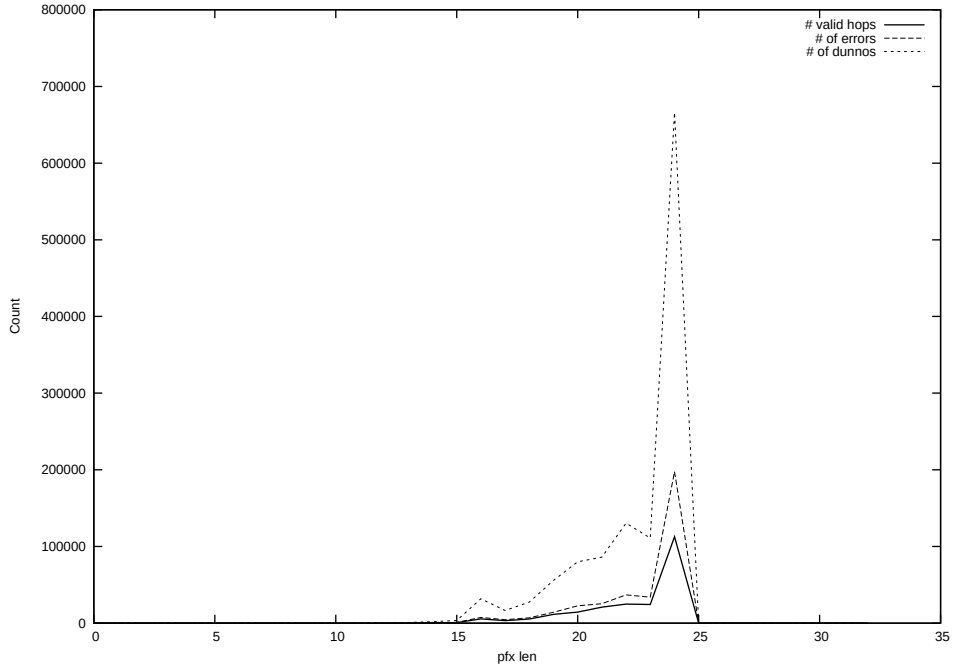
2014-01-14



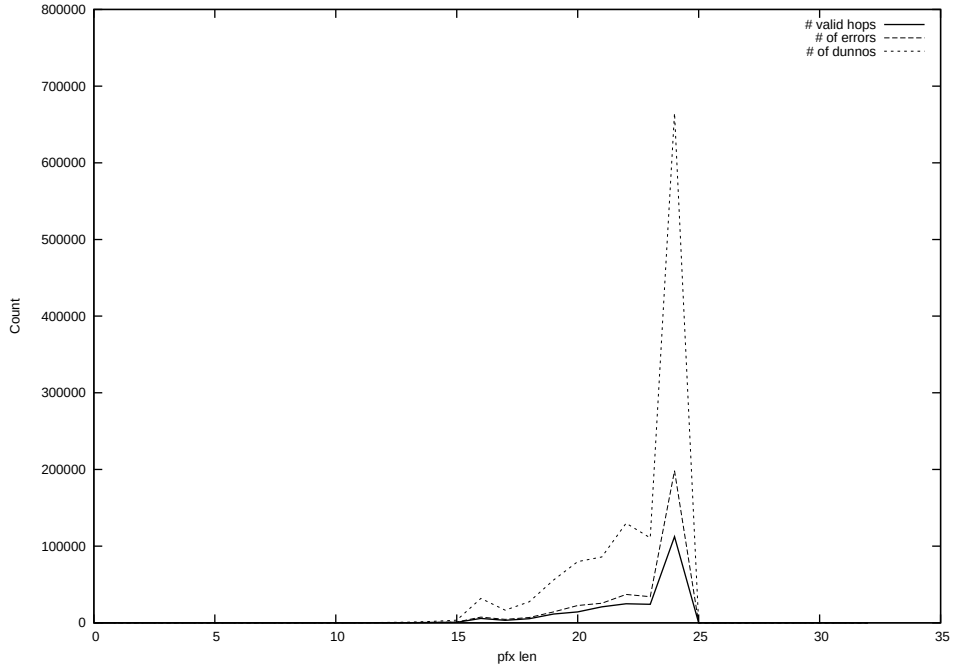
2014-01-15



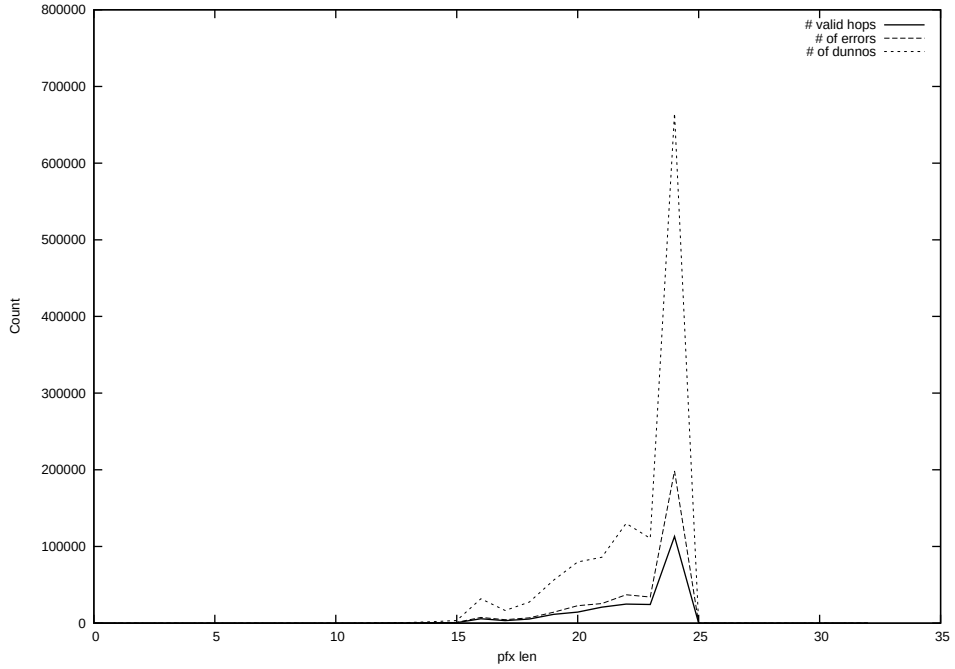
2014-01-16



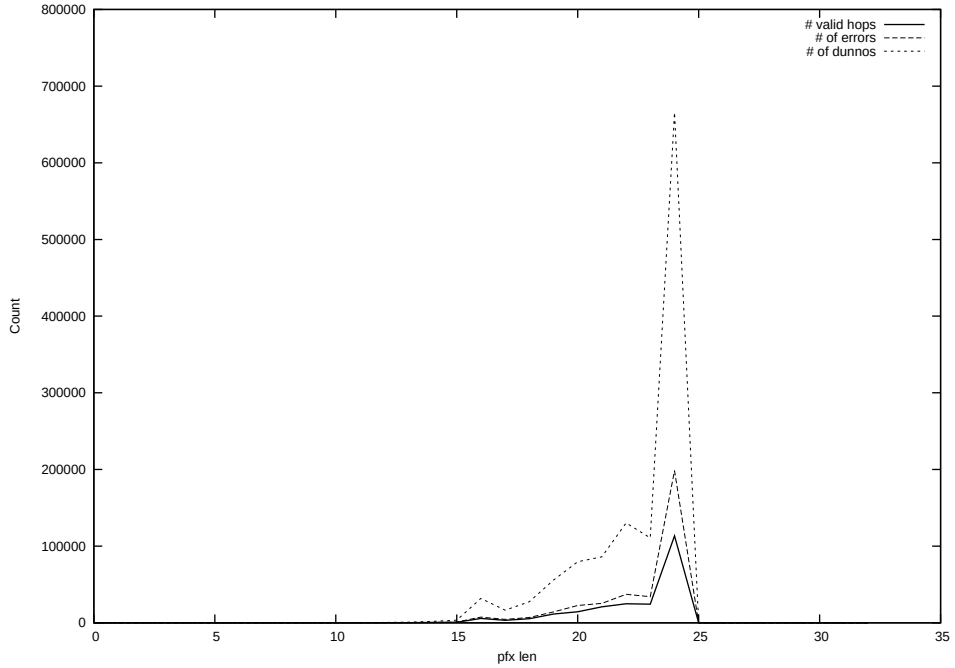
2014-01-17



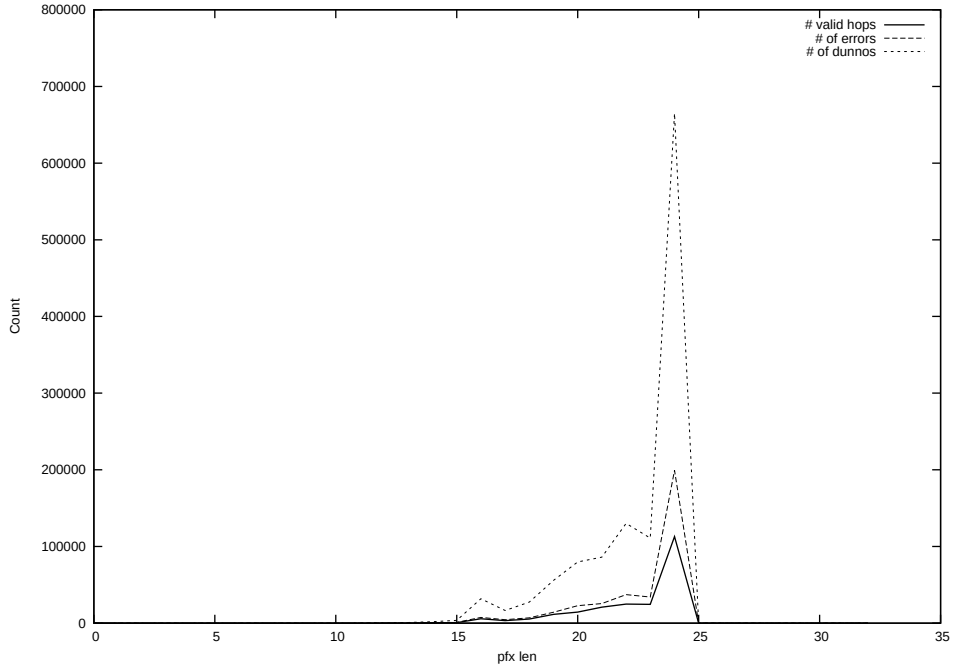
2014-01-18



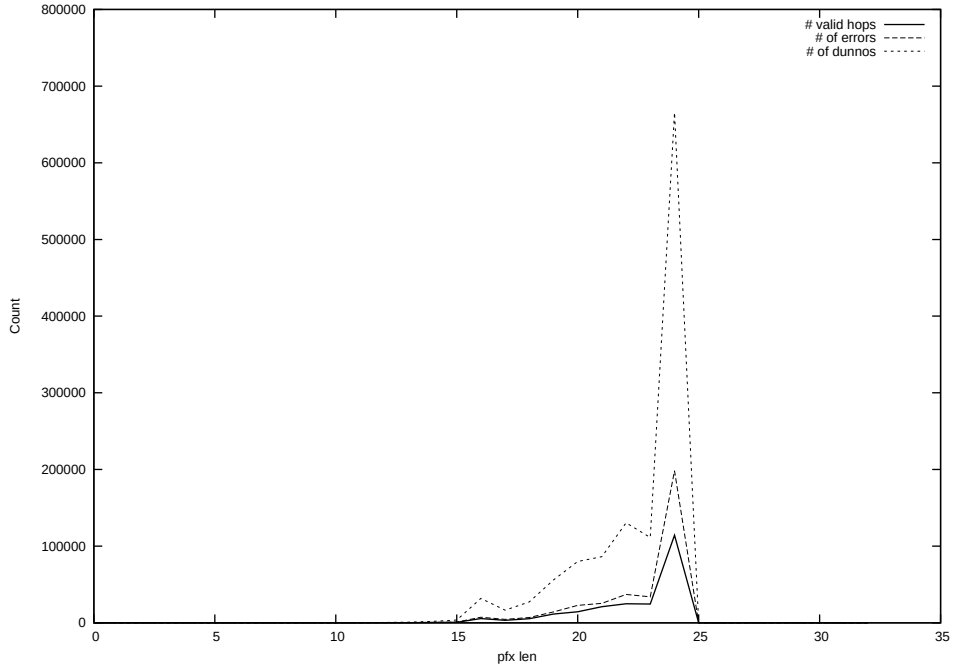
2014-01-19



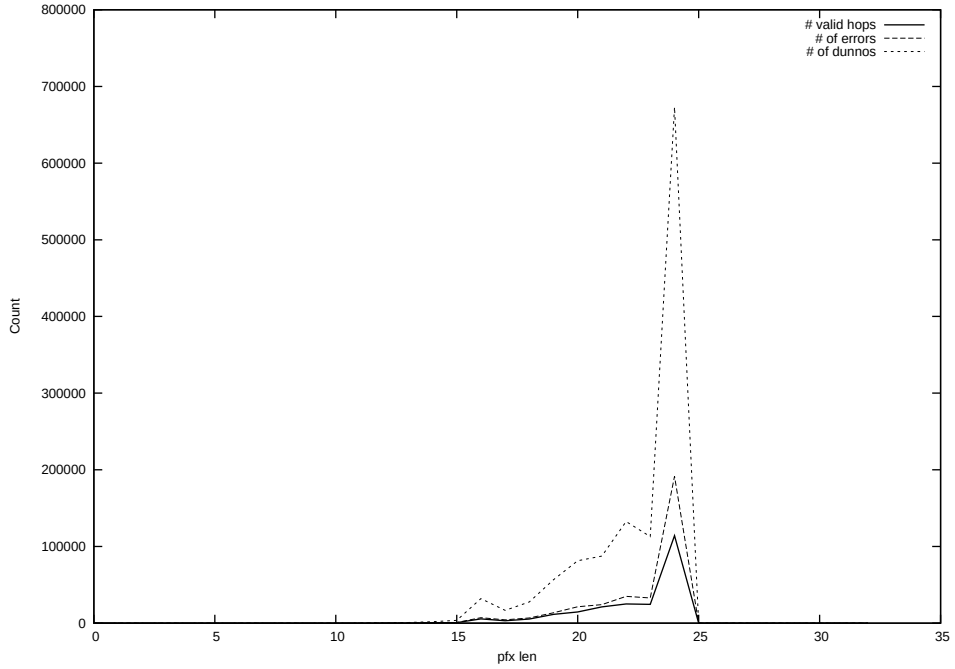
2014-01-20



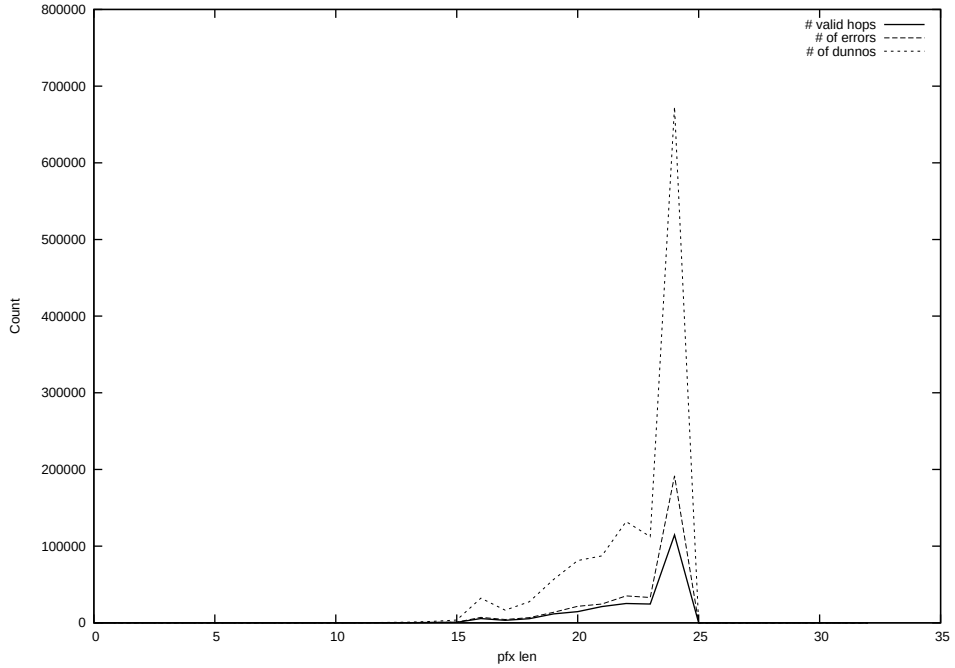
2014-01-21



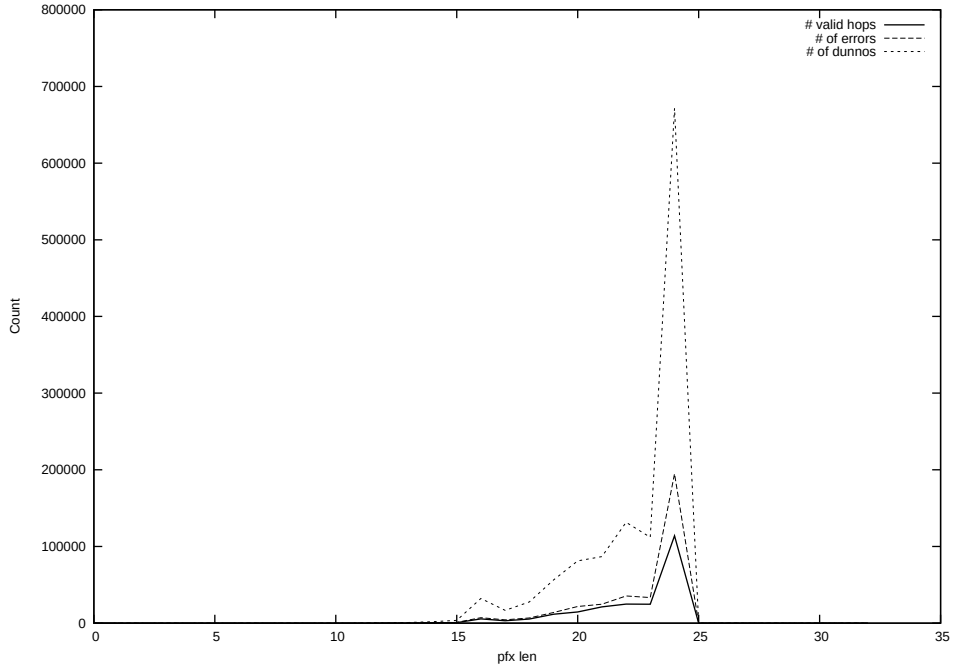
2014-01-22



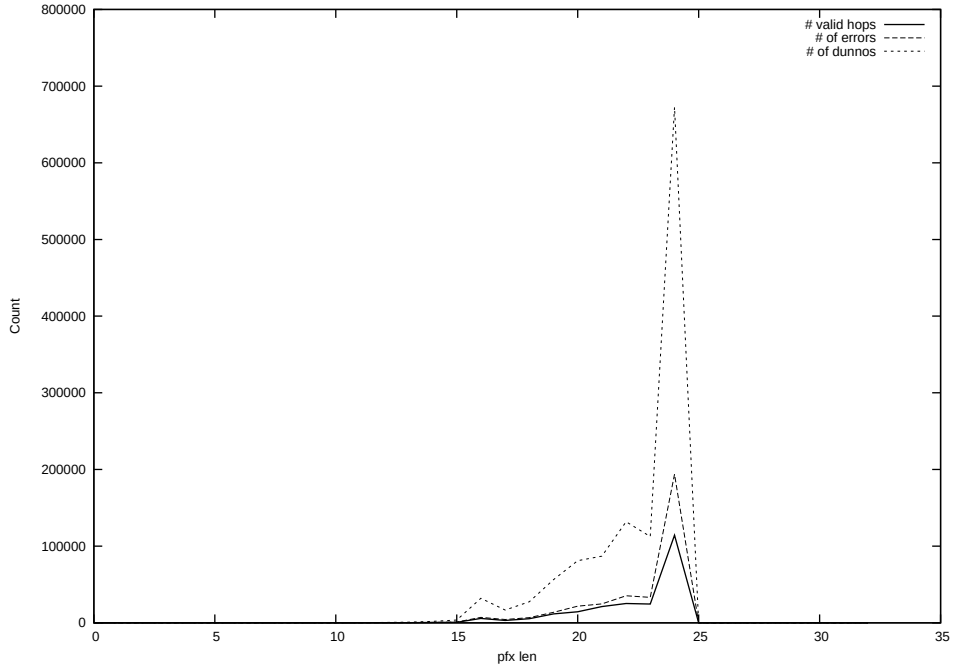
2014-01-23



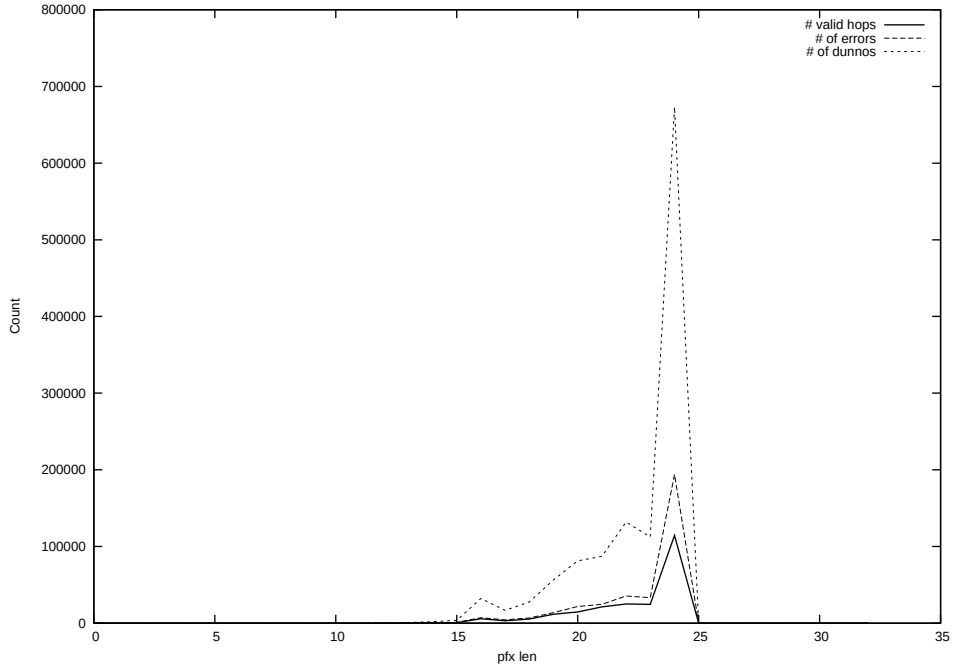
2014-01-24



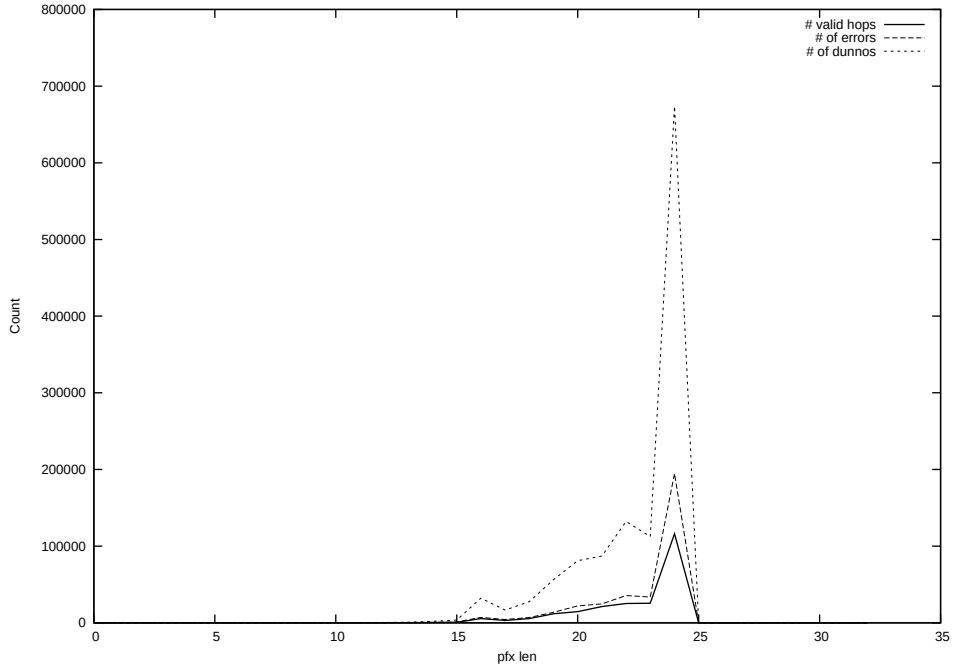
2014-01-25



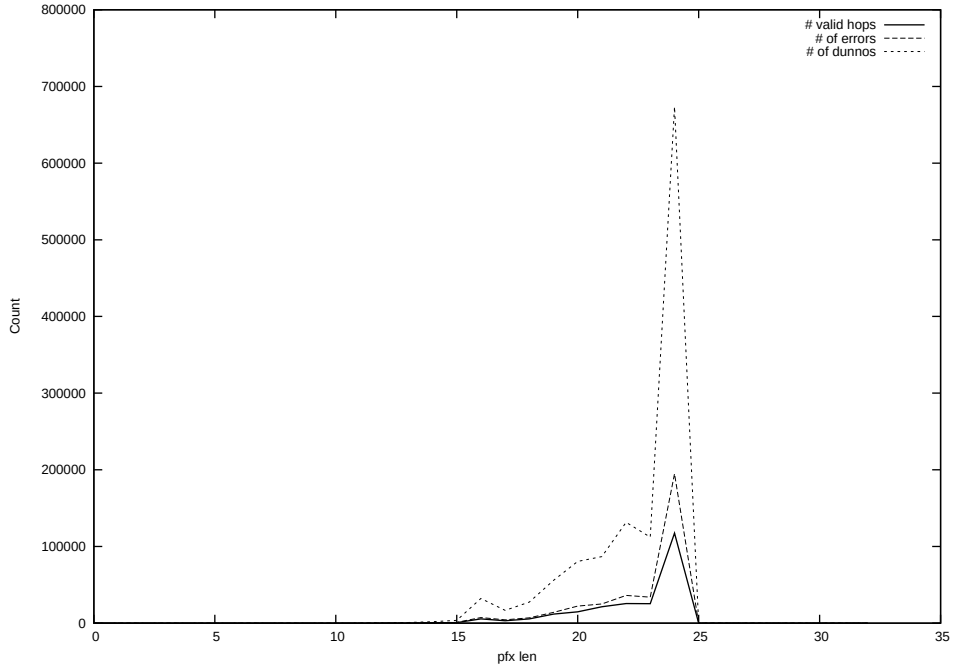
2014-01-26



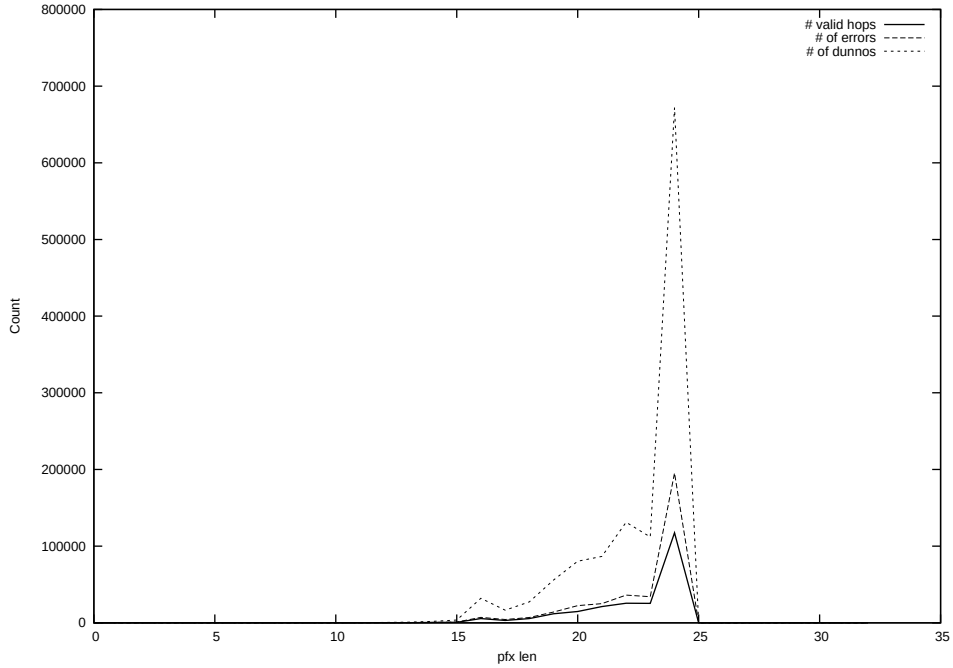
2014-01-27



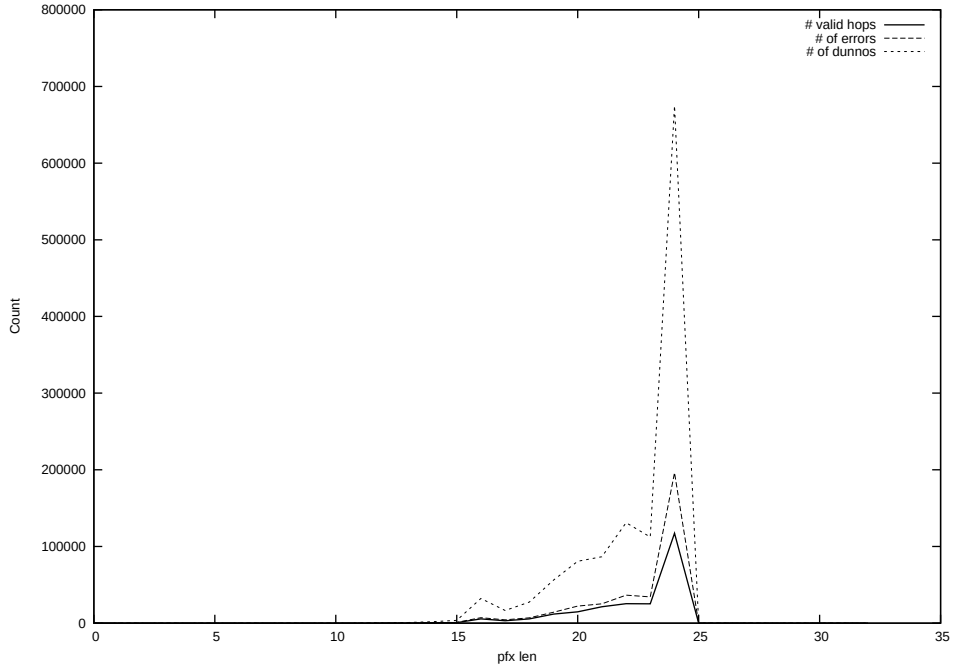
2014-01-28



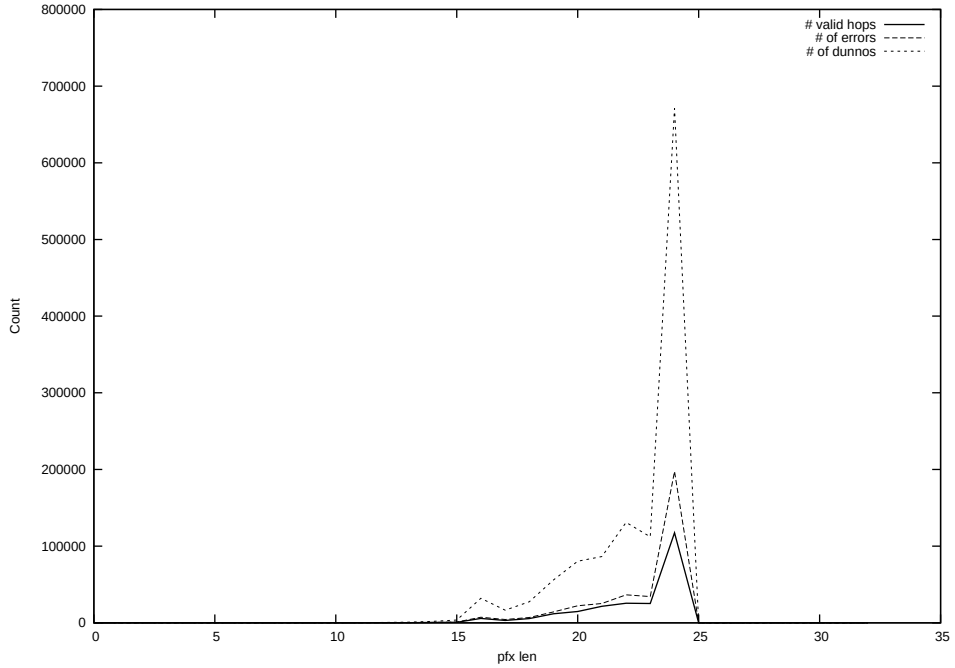
2014-01-29



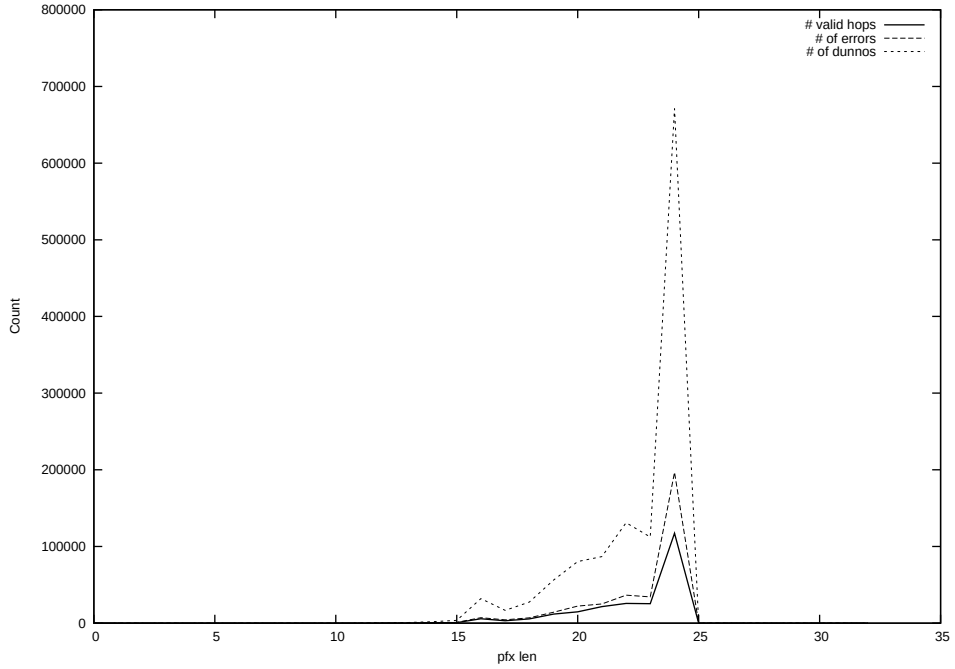
2014-01-30



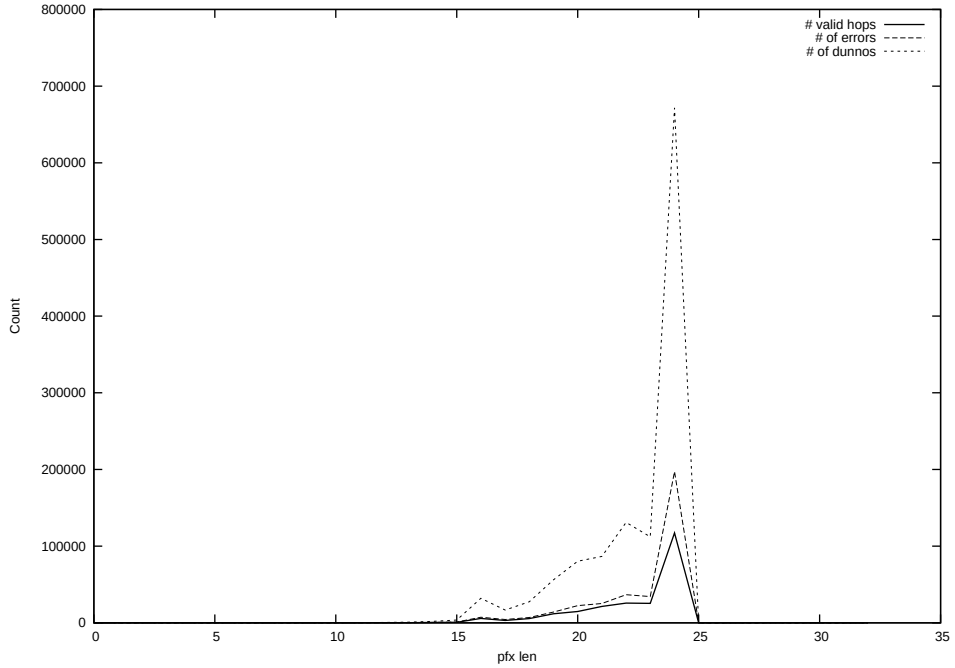
2014-01-31



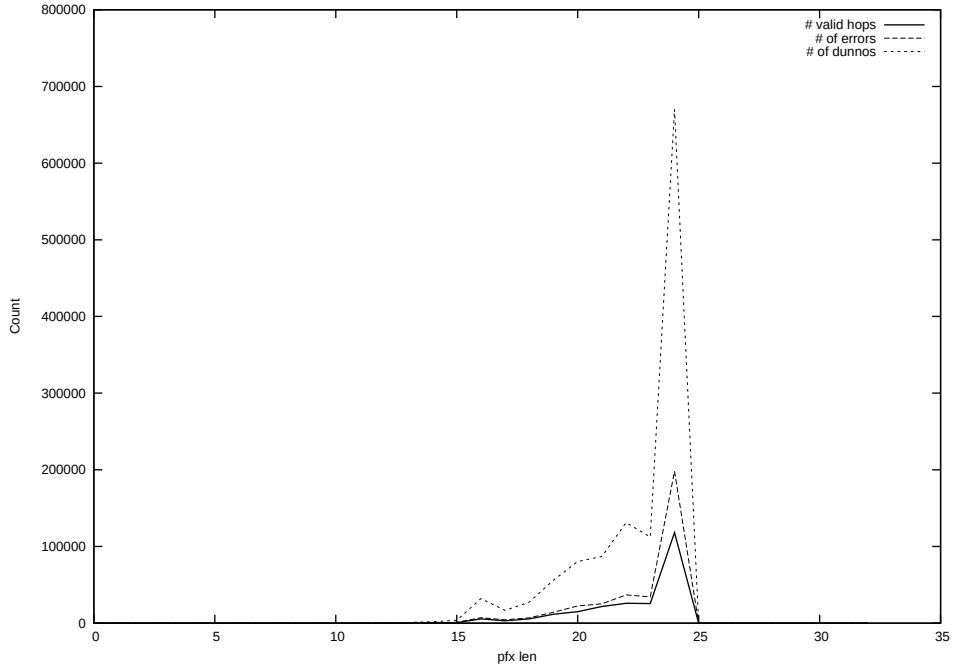
2014-02-01



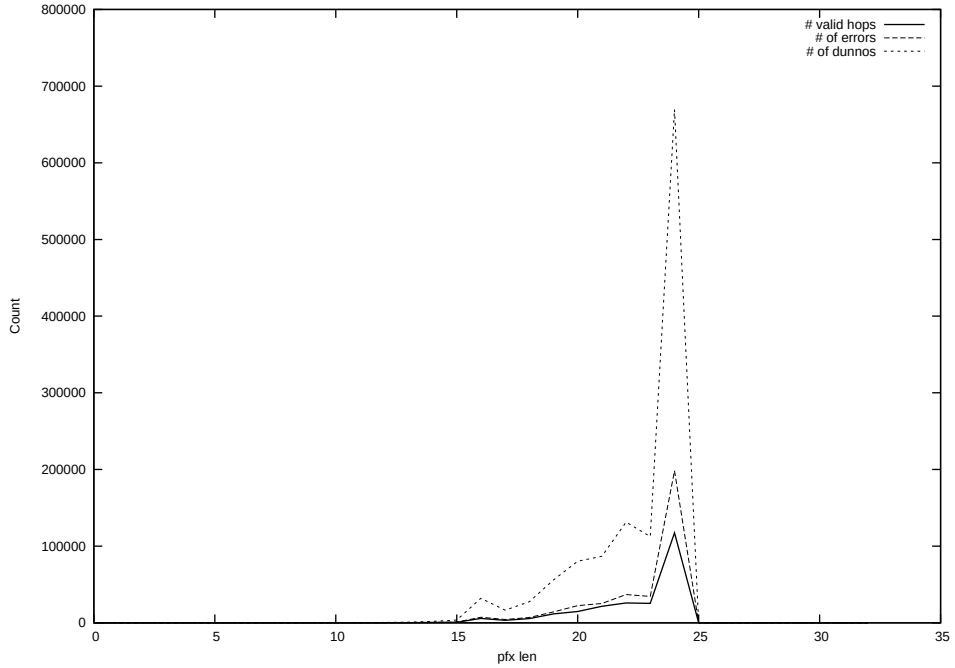
2014-02-02



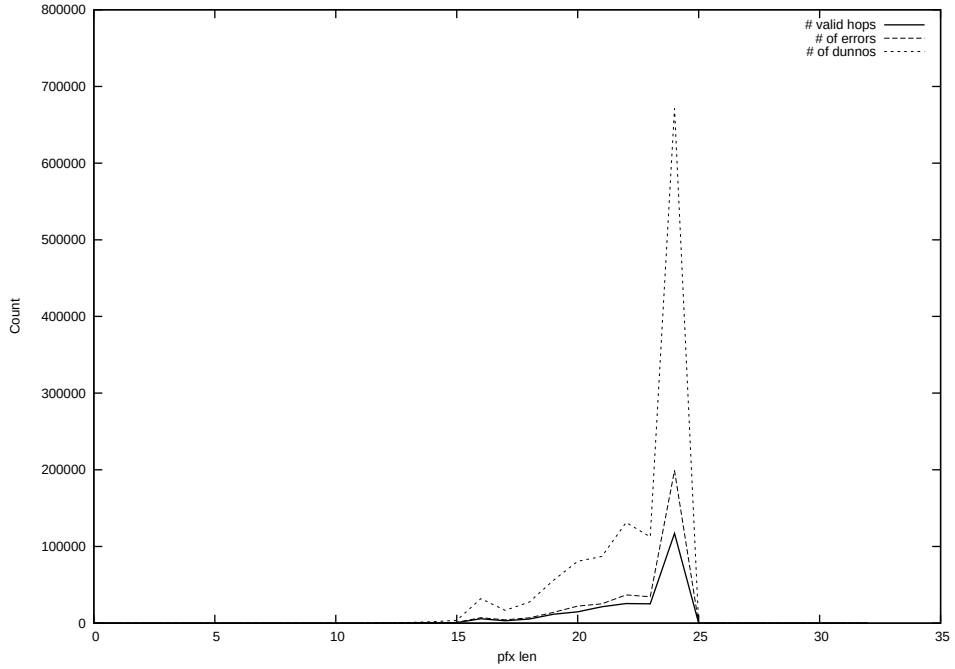
2014-02-03



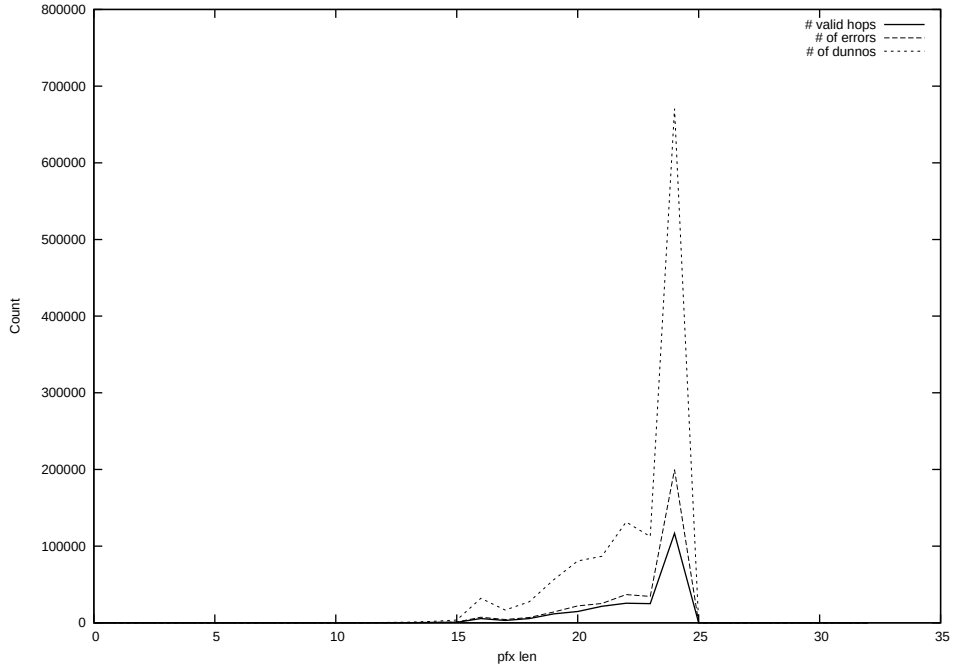
2014-02-04



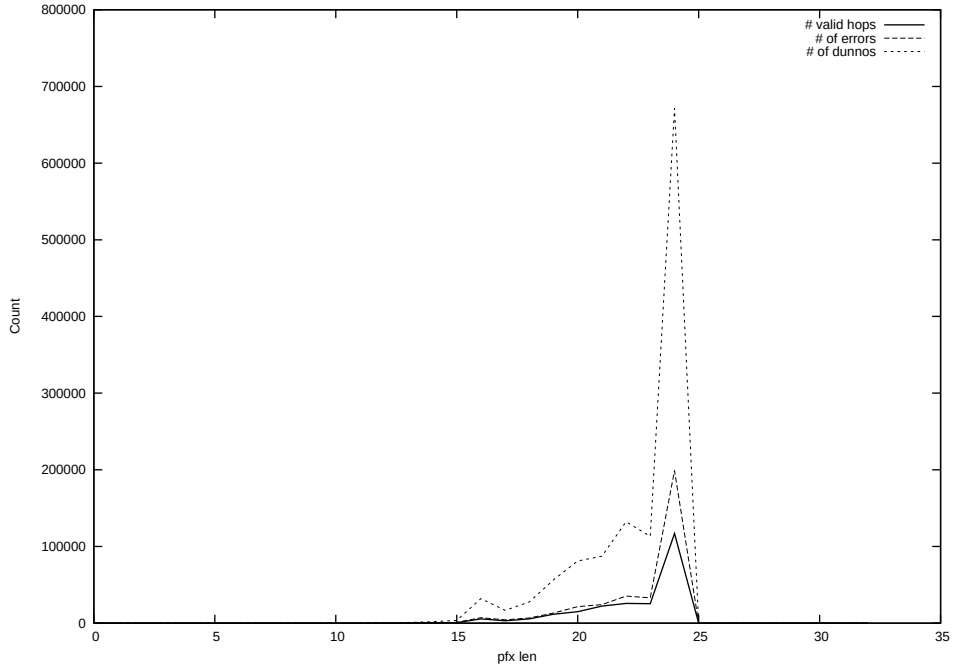
2014-02-05



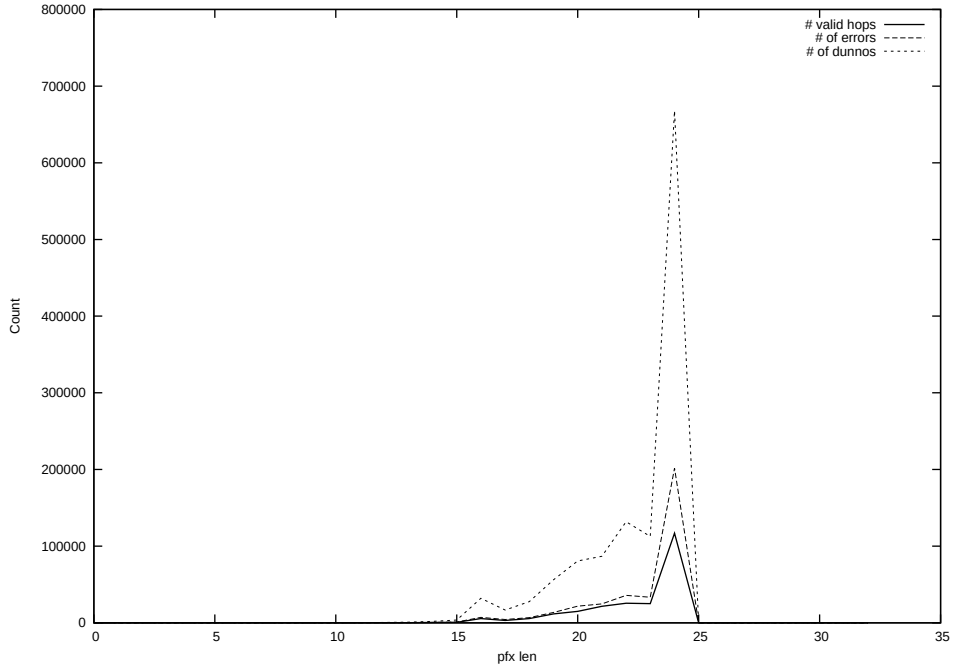
2014-02-06



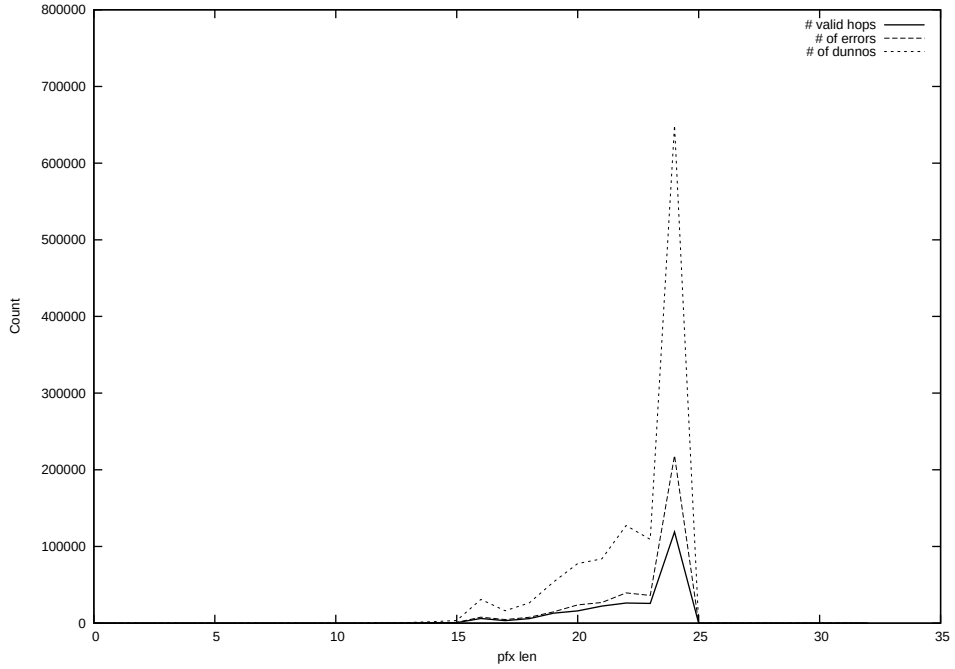
2014-02-07



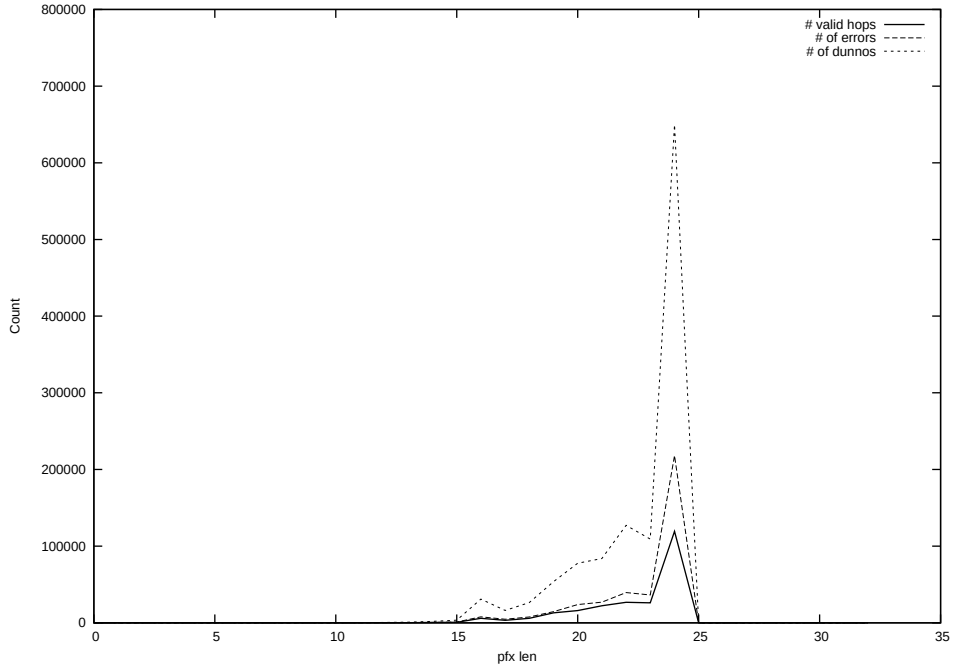
2014-02-08



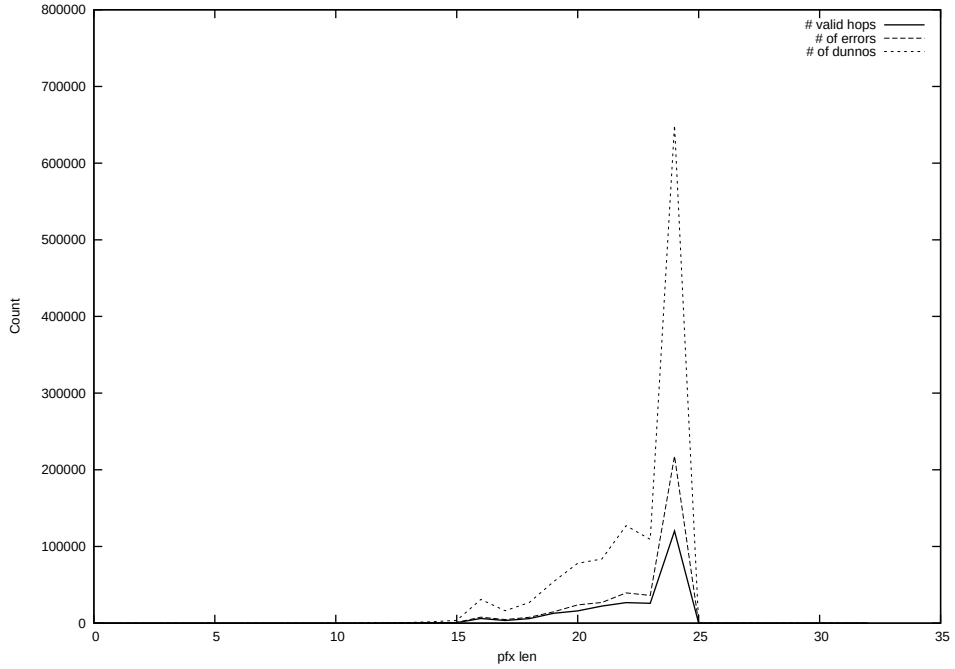
2014-02-09



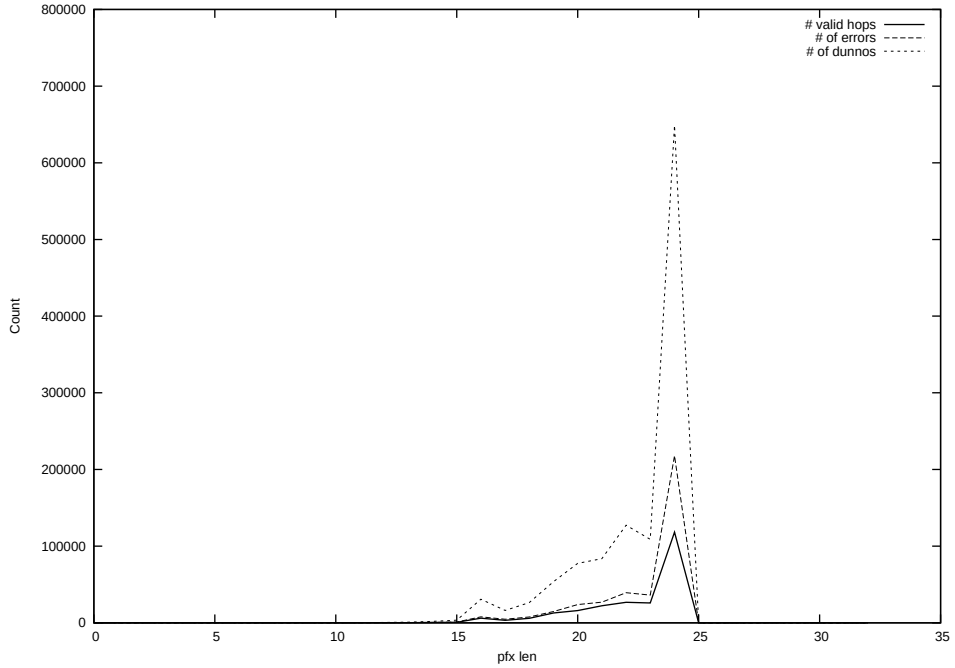
2014-02-10



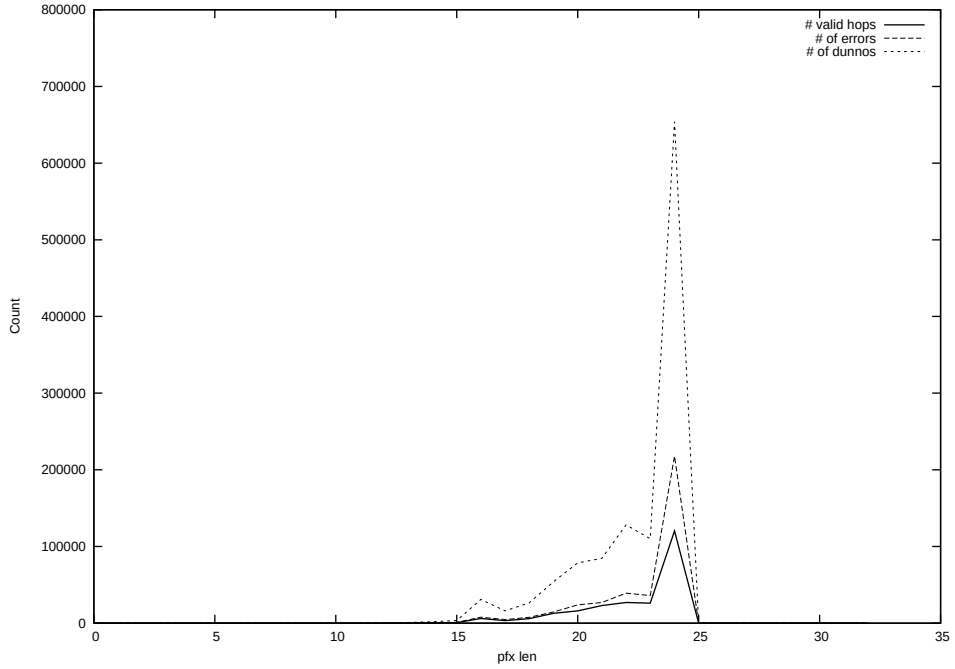
2014-02-11



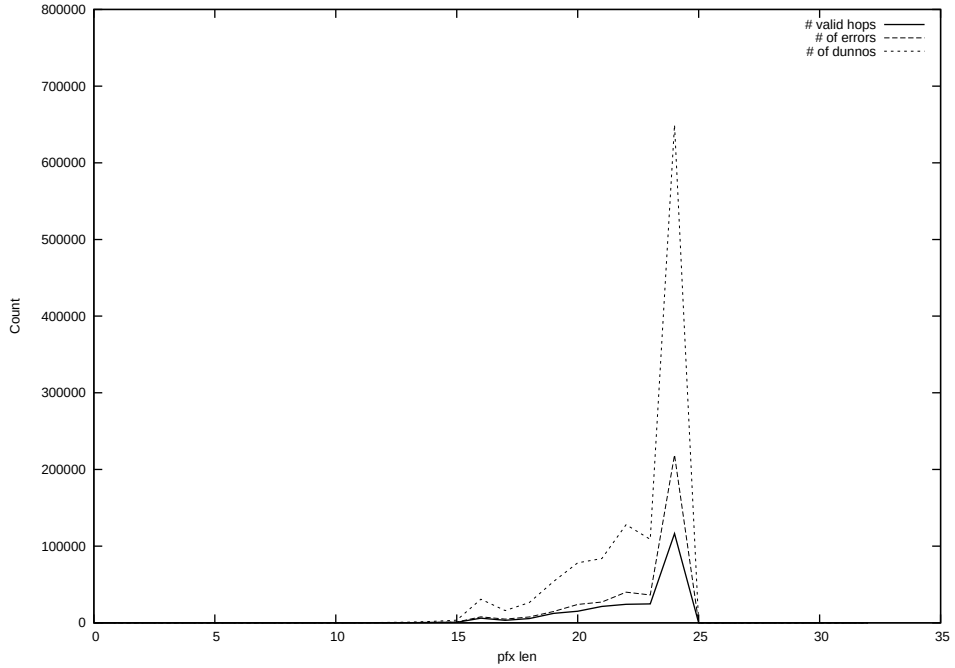
2014-02-12



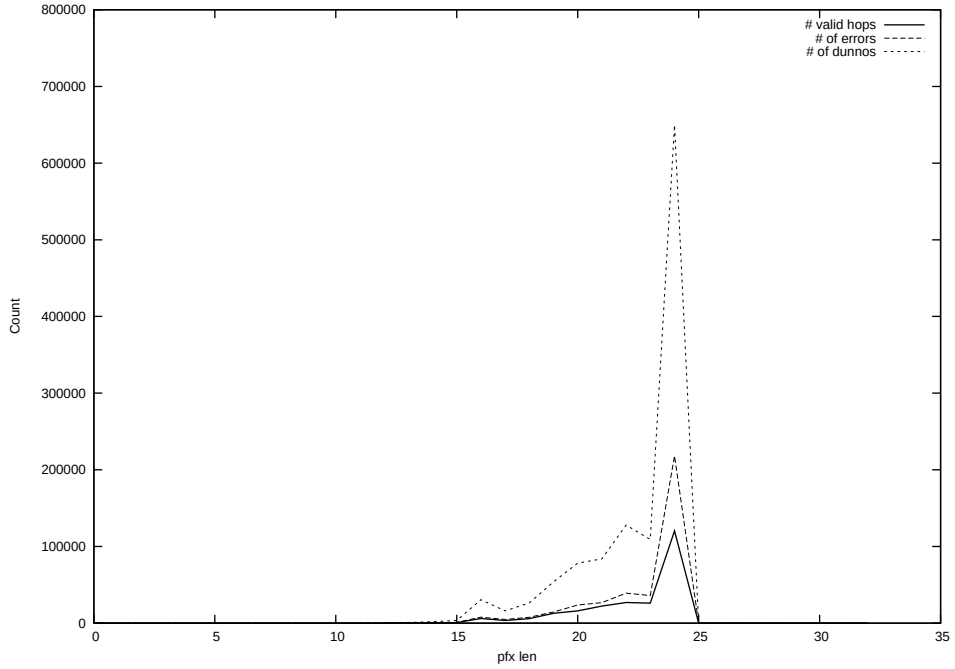
2014-02-13



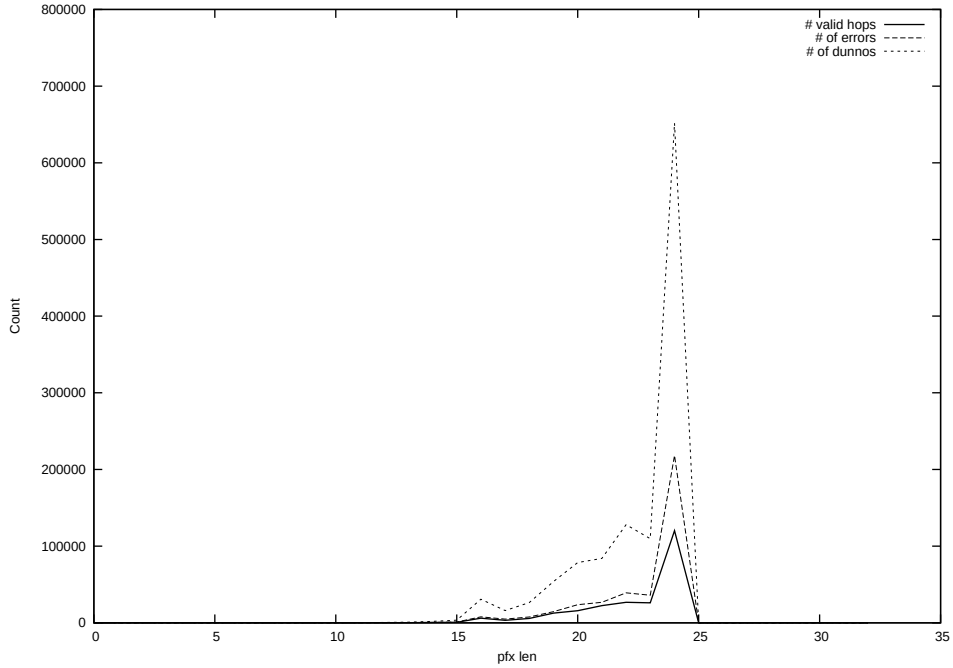
2014-02-14



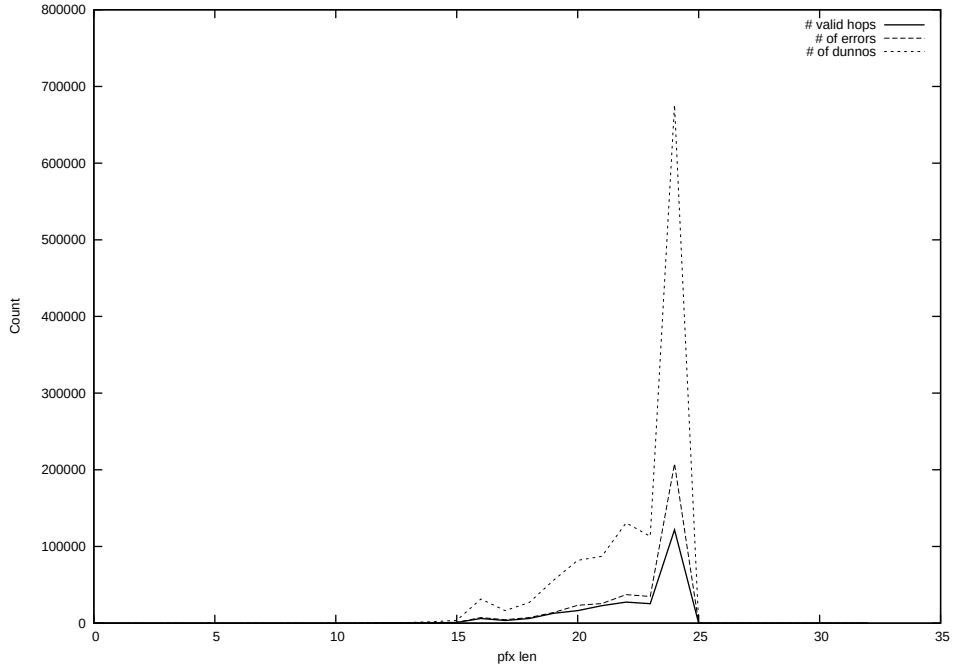
2014-02-15



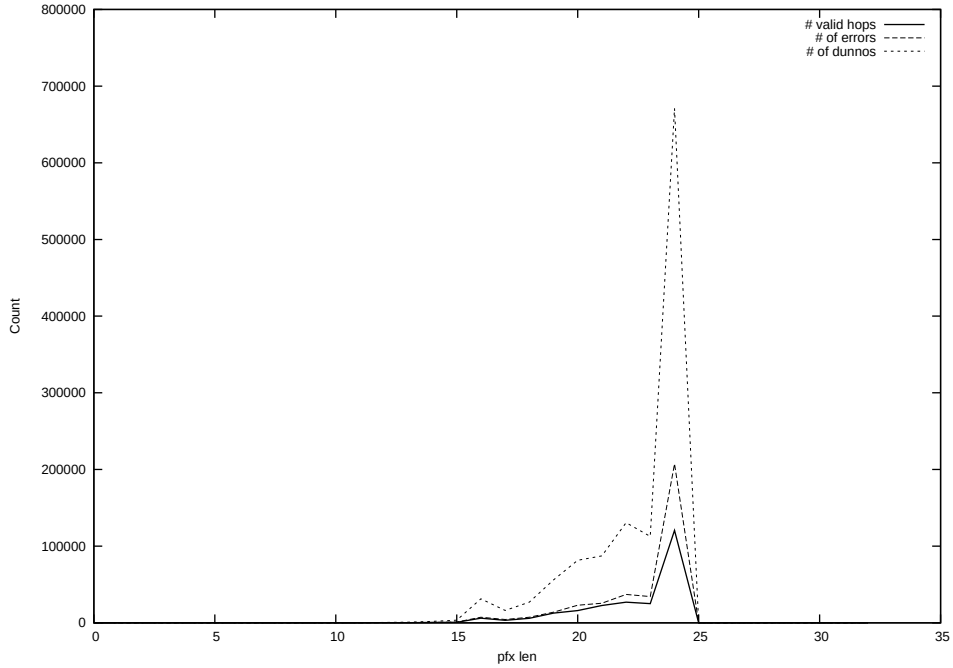
2014-02-16



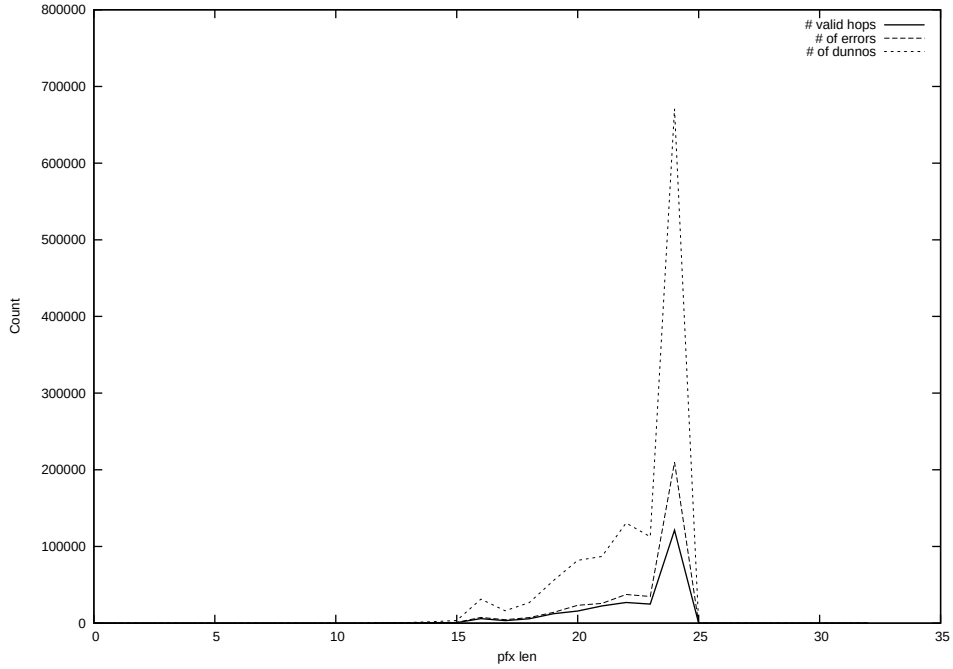
2014-02-17



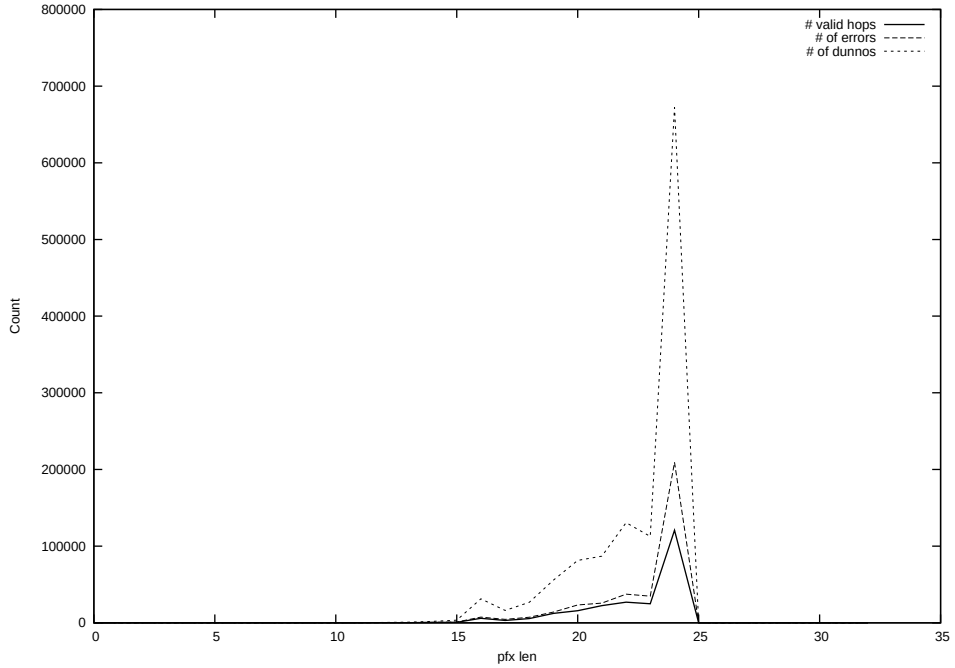
2014-03-20



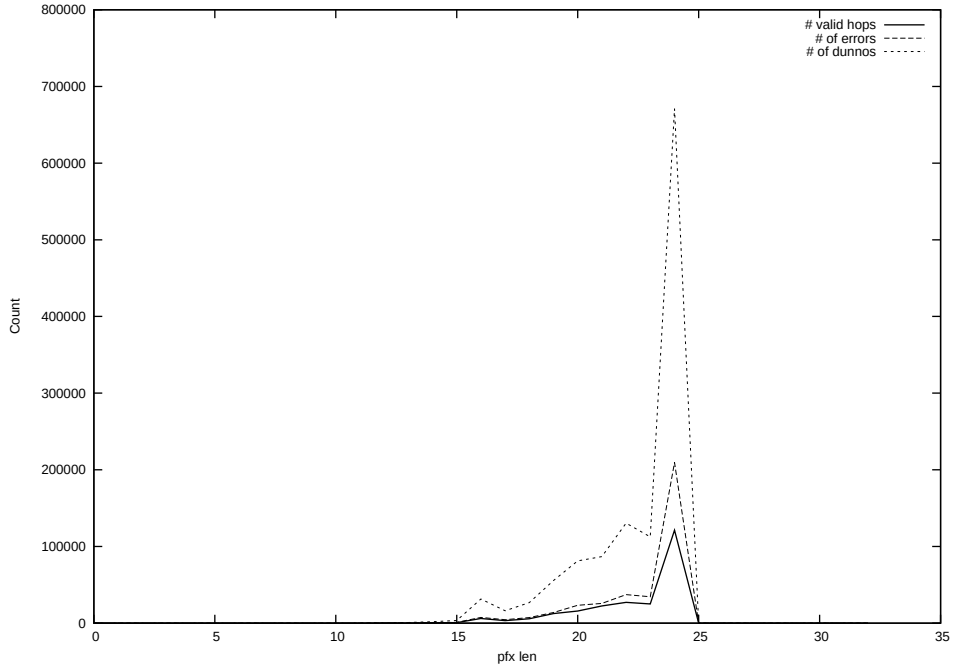
2014-03-21



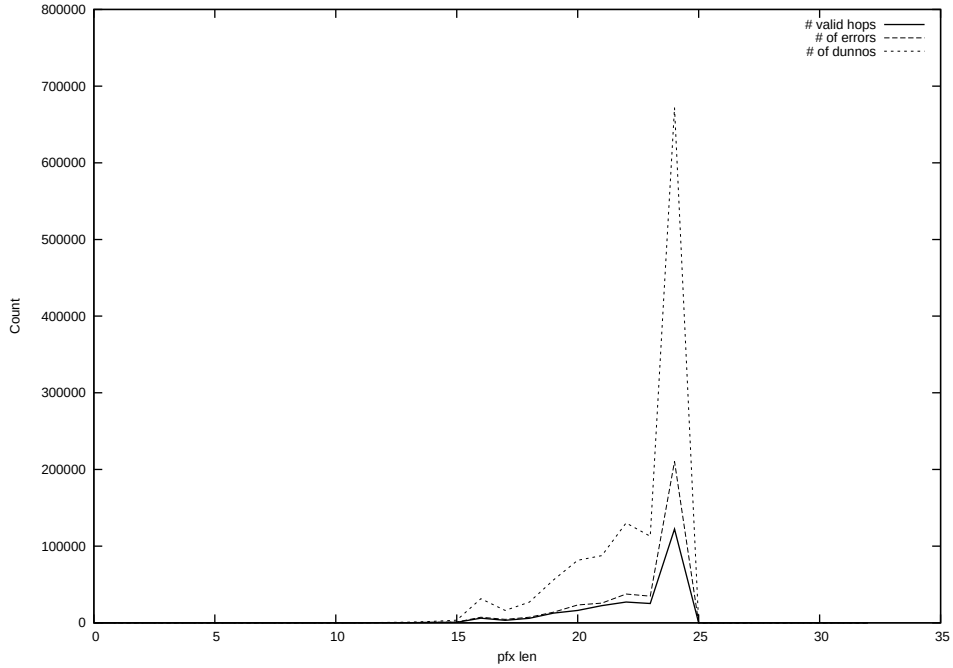
2014-03-22



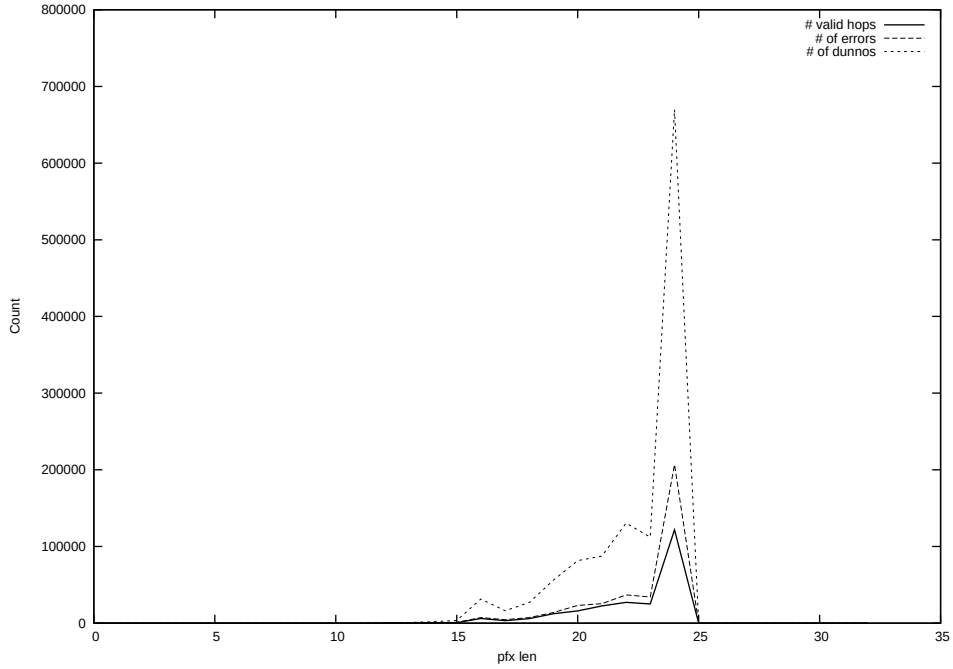
2014-03-23



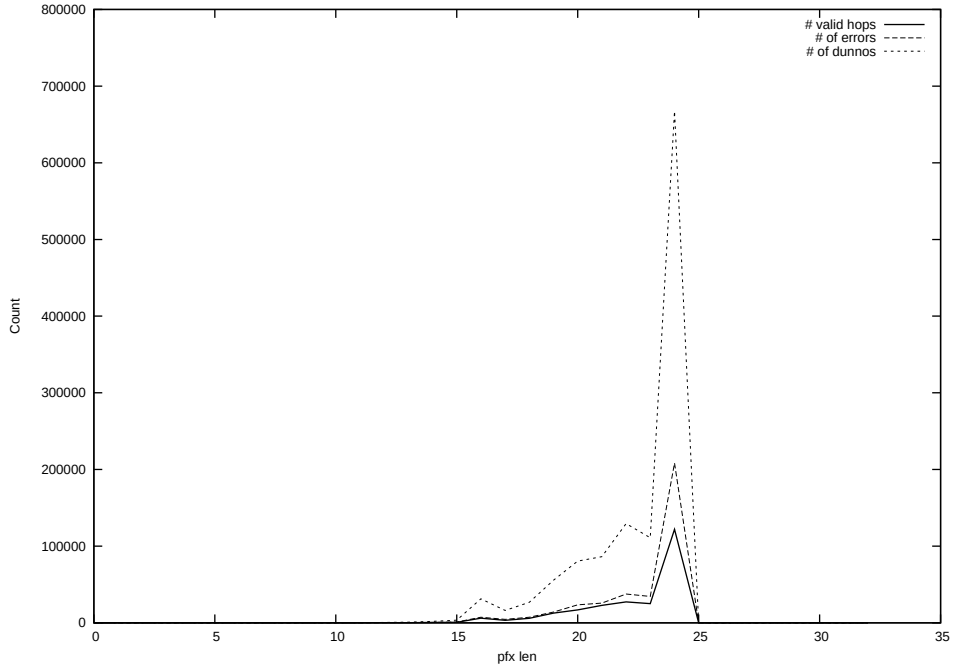
2014-03-24



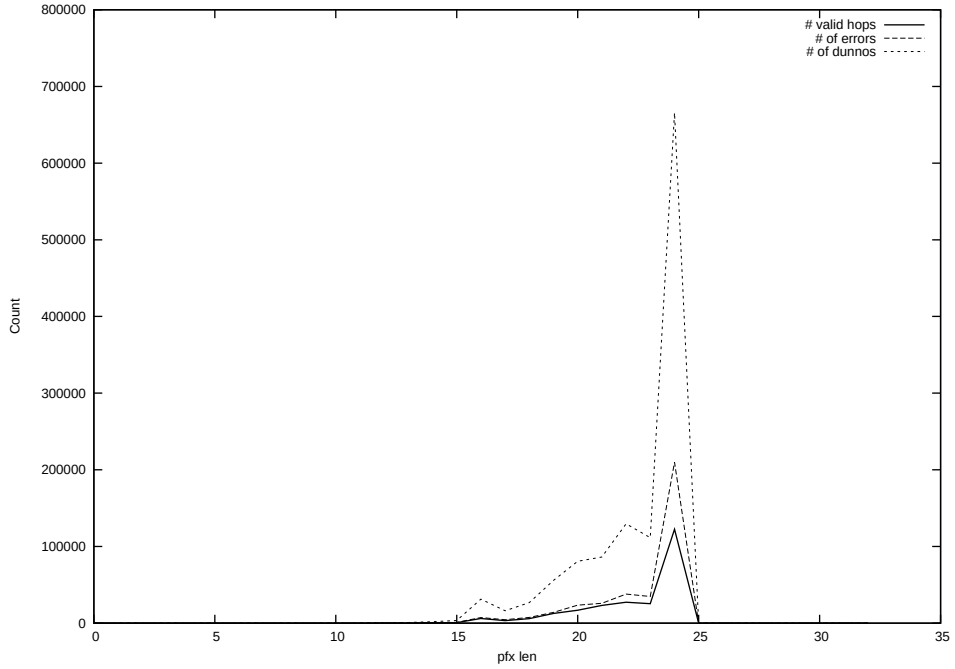
2014-03-25



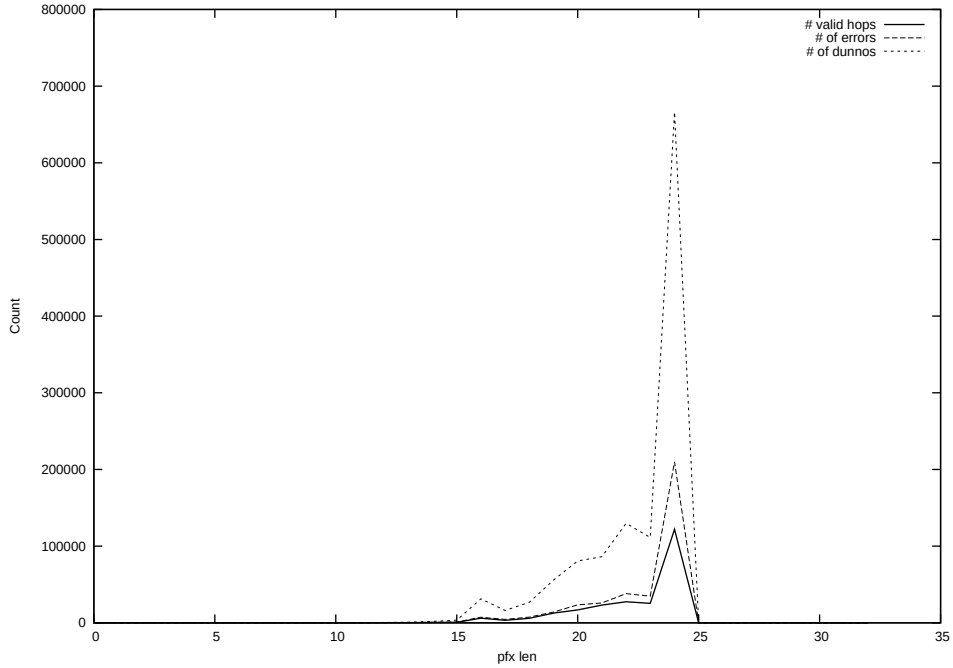
2014-03-26



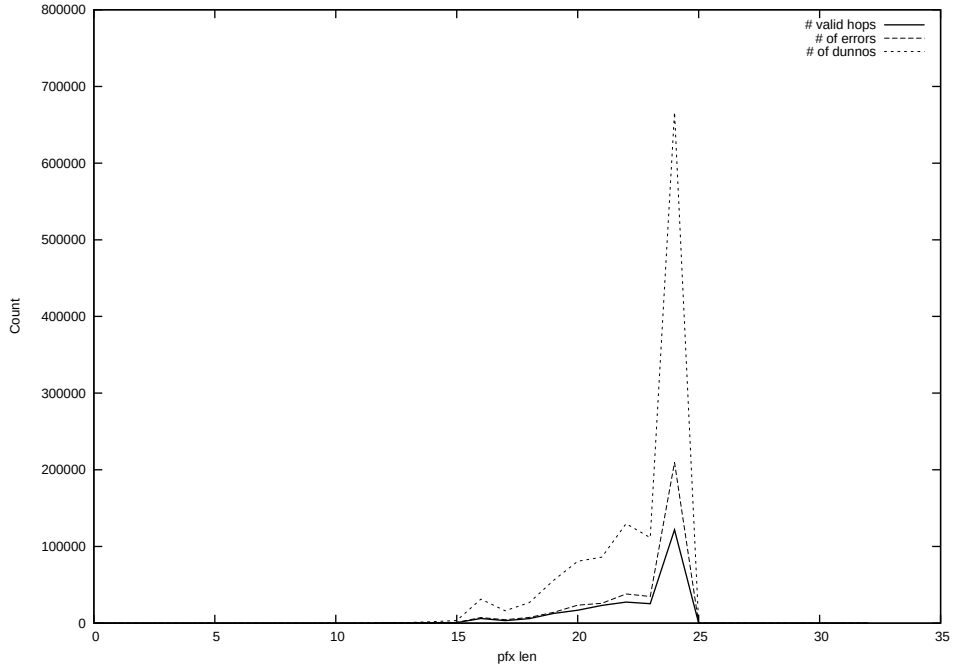
2014-03-27



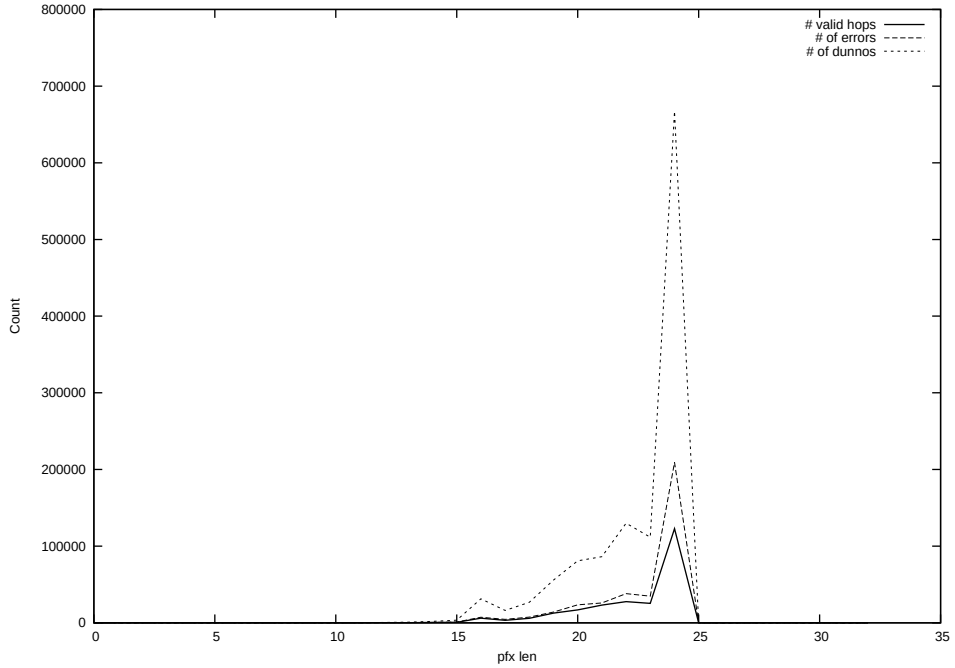
2014-03-28



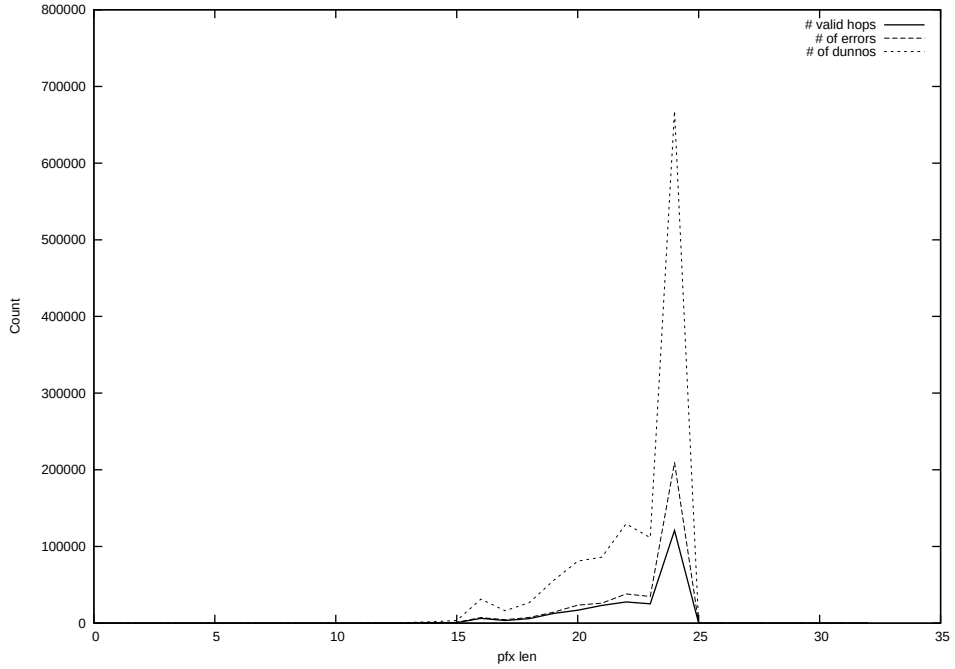
2014-03-29



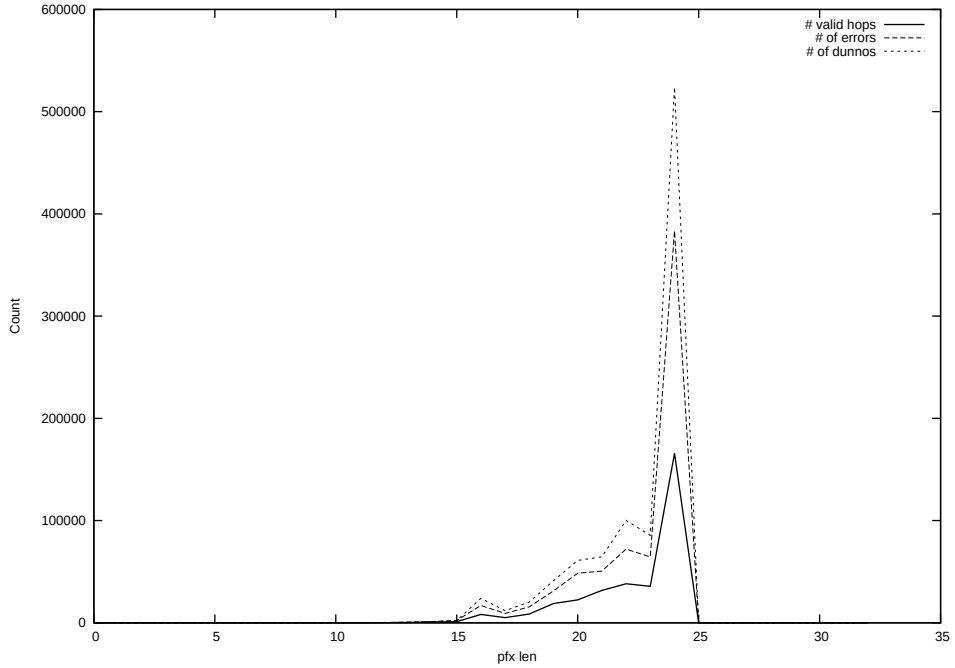
2014-03-30



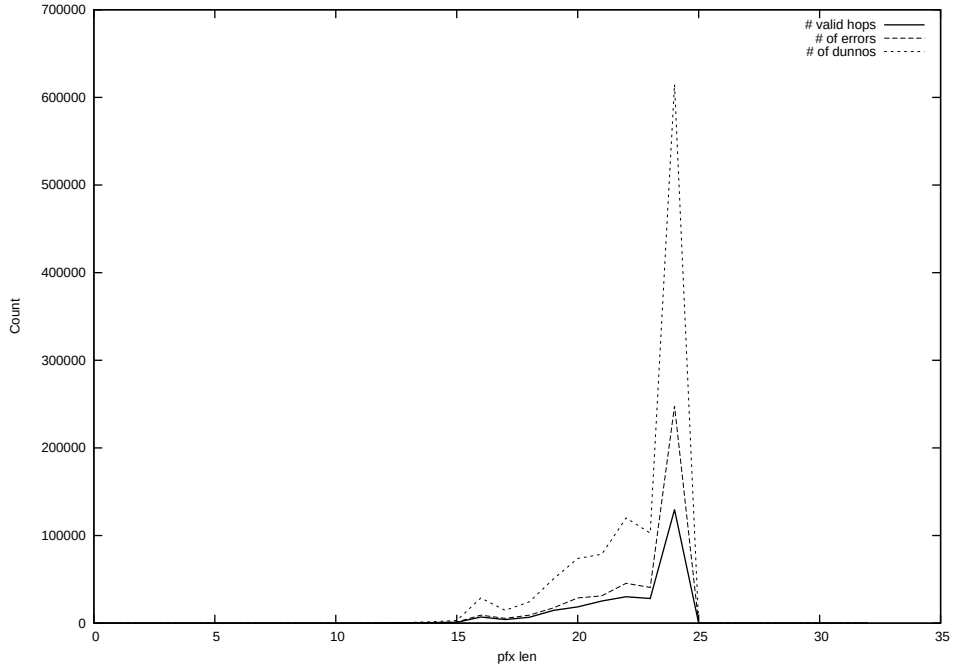
2014-03-31



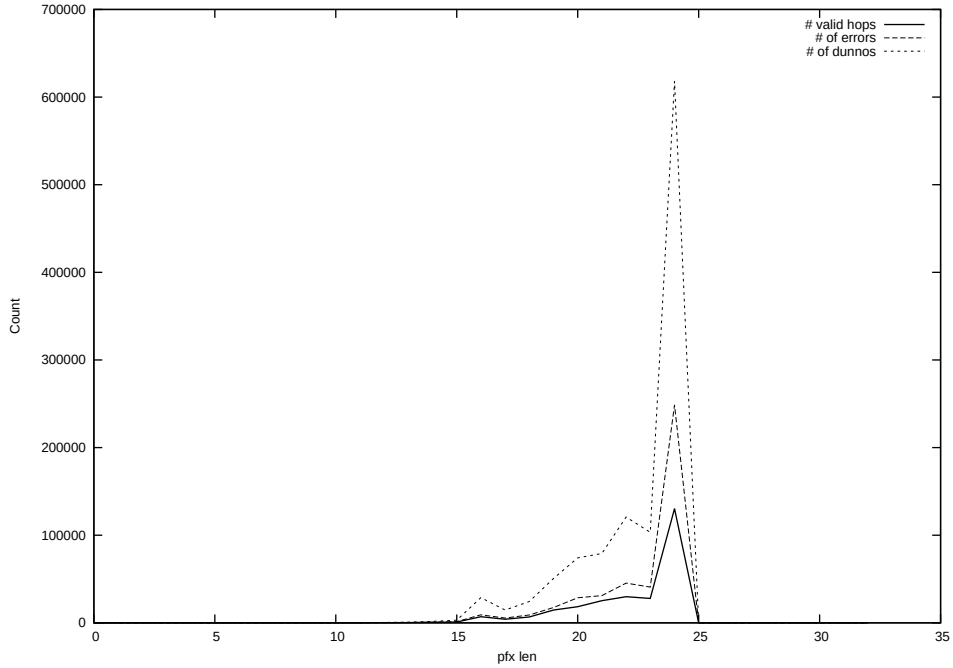
2014-04-01



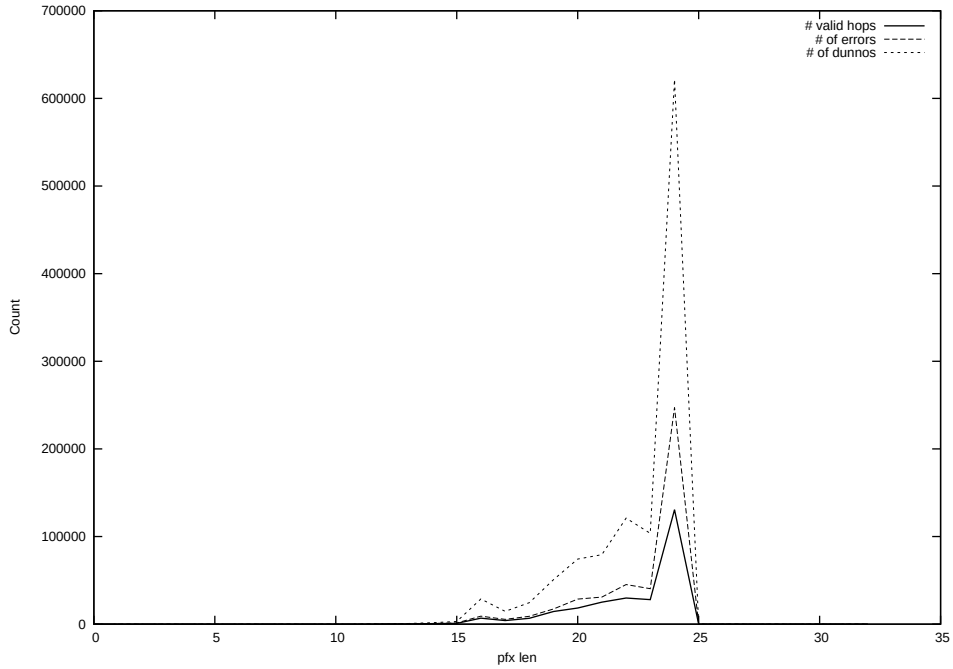
2014-04-02



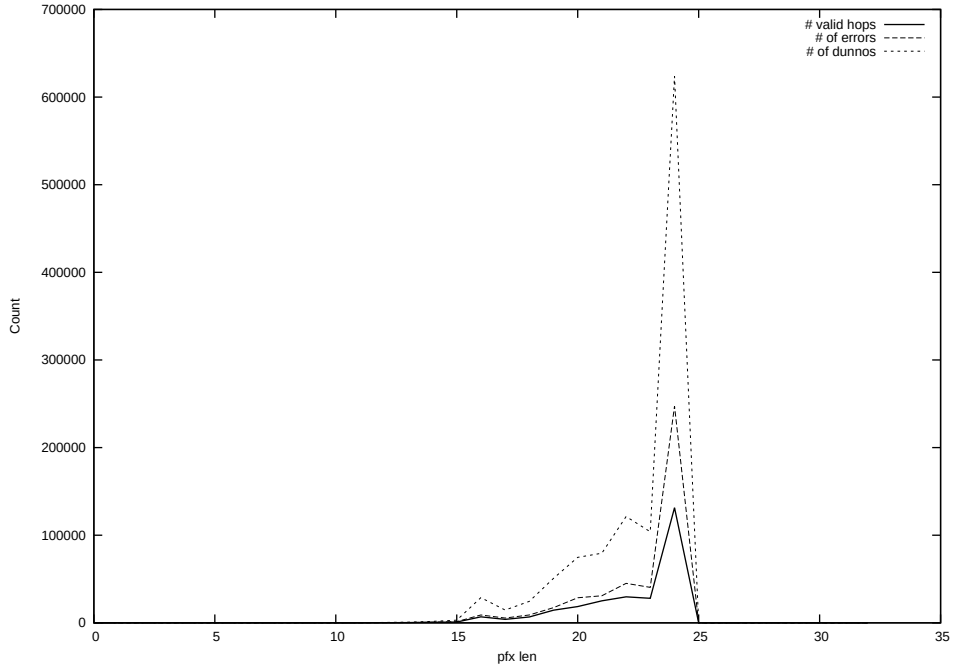
2014-04-03



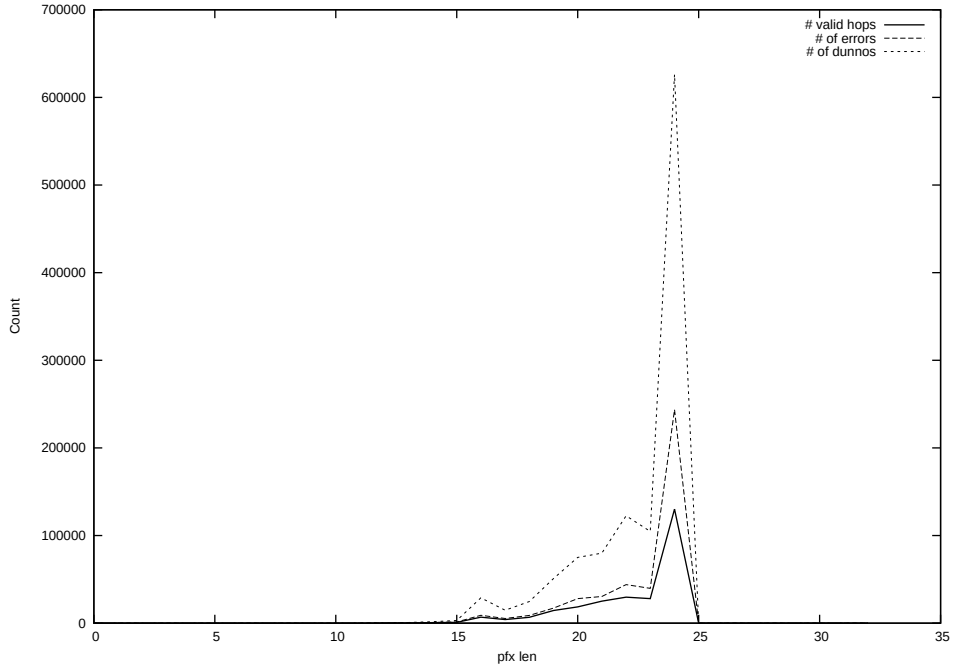
2014-04-04



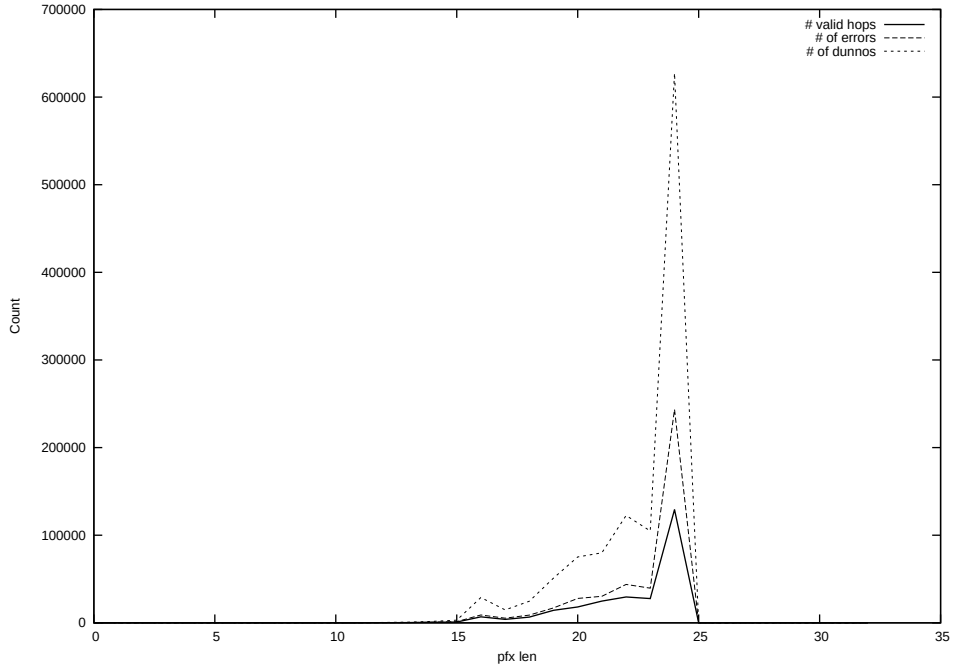
2014-04-05



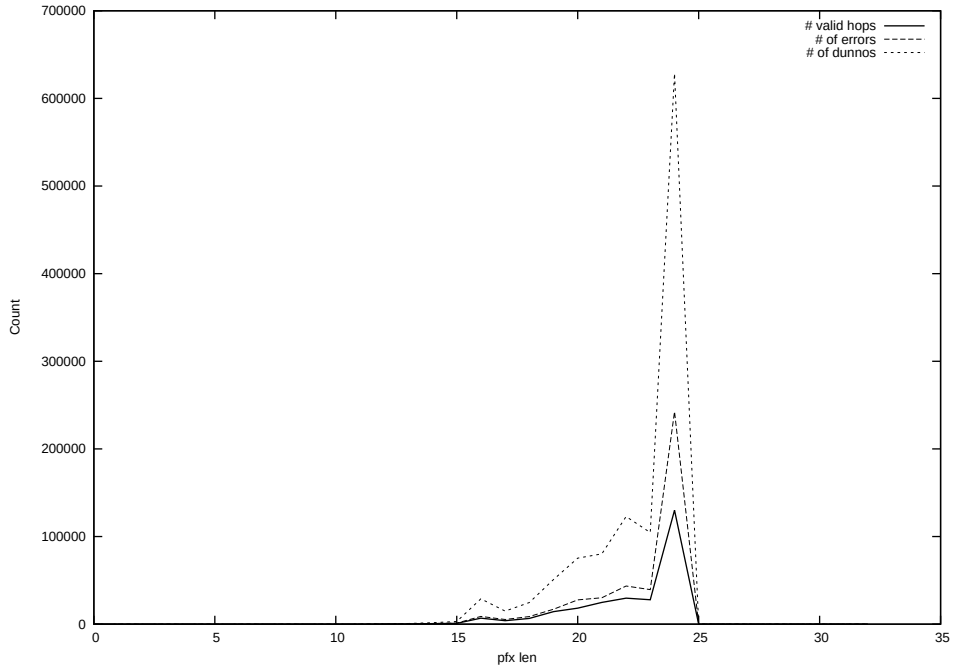
2014-04-06



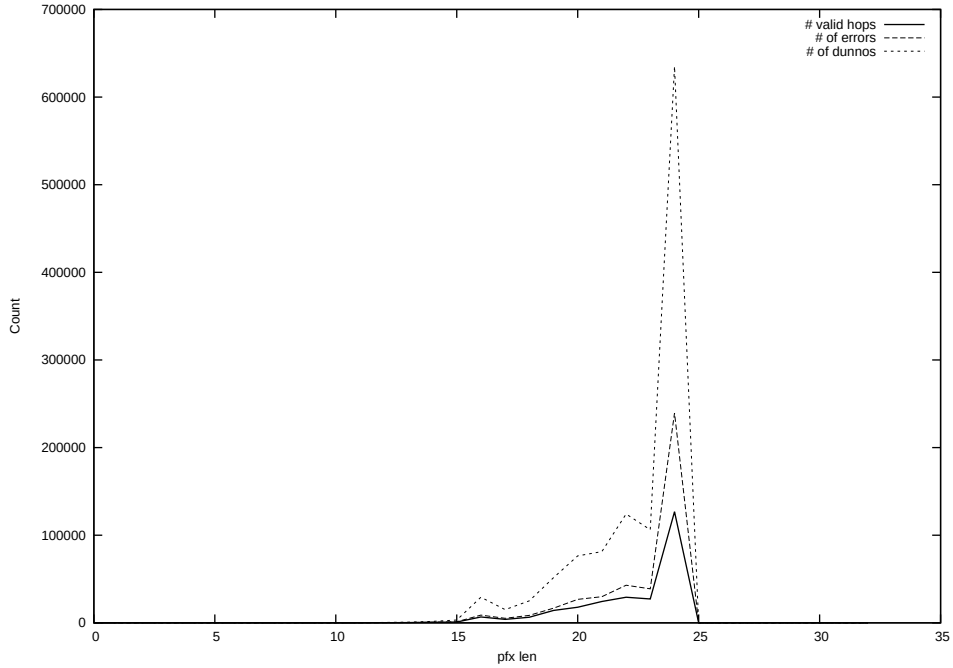
2014-04-07



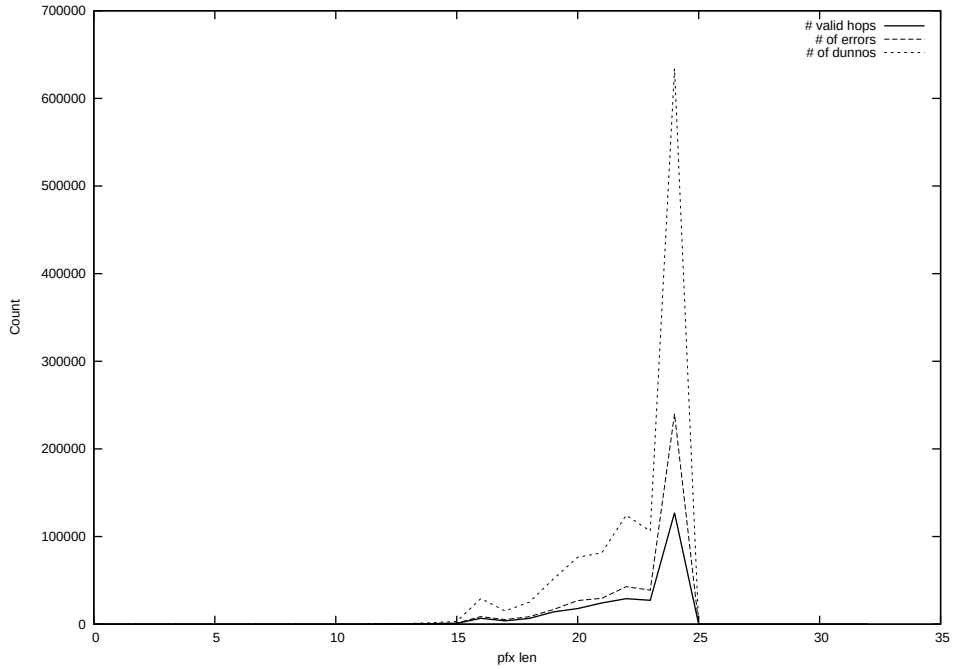
2014-04-08



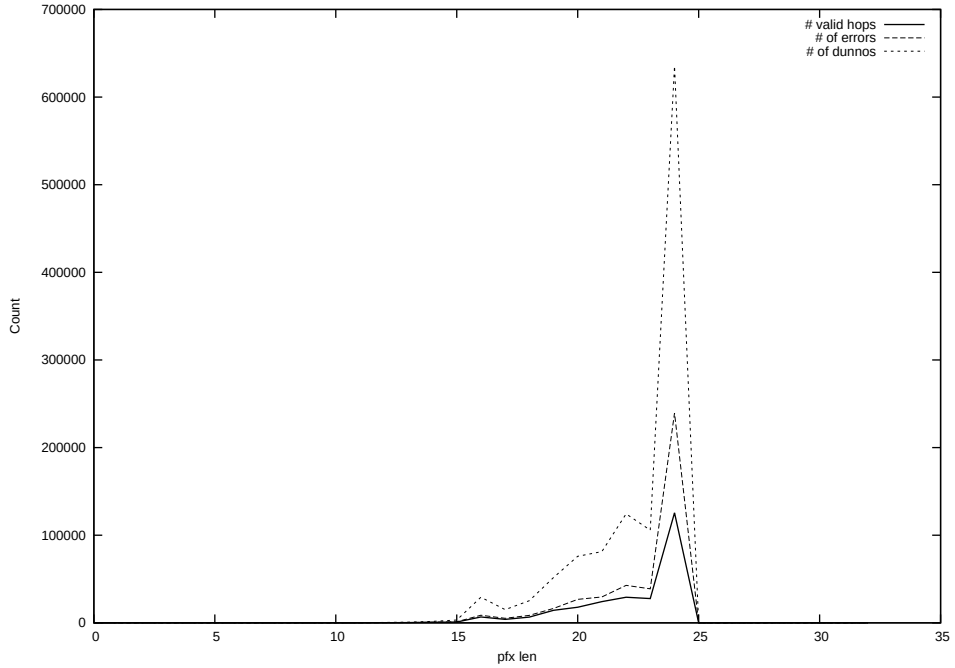
2014-04-09



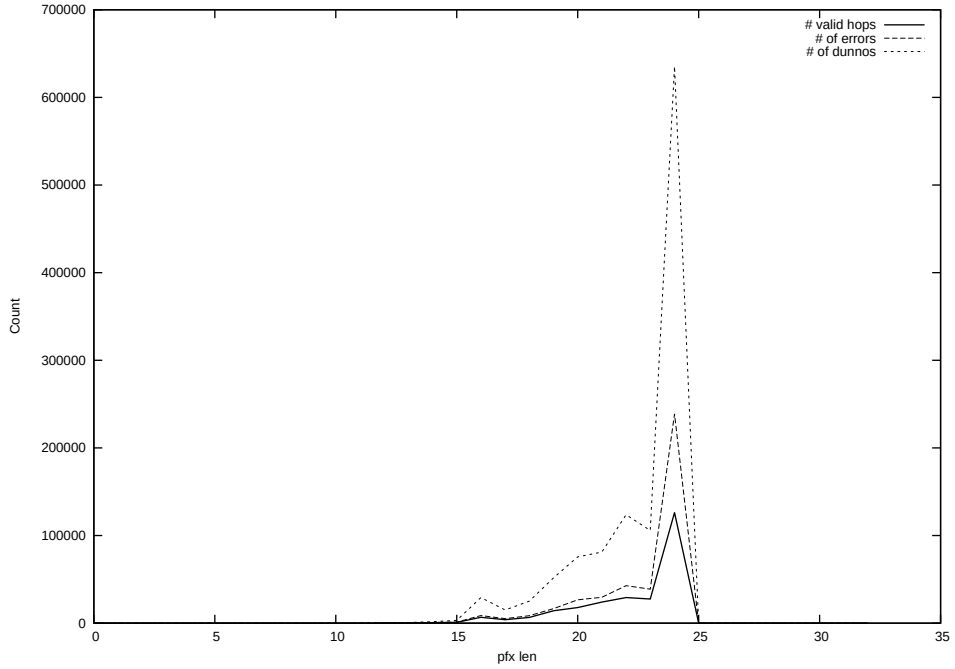
2014-04-10



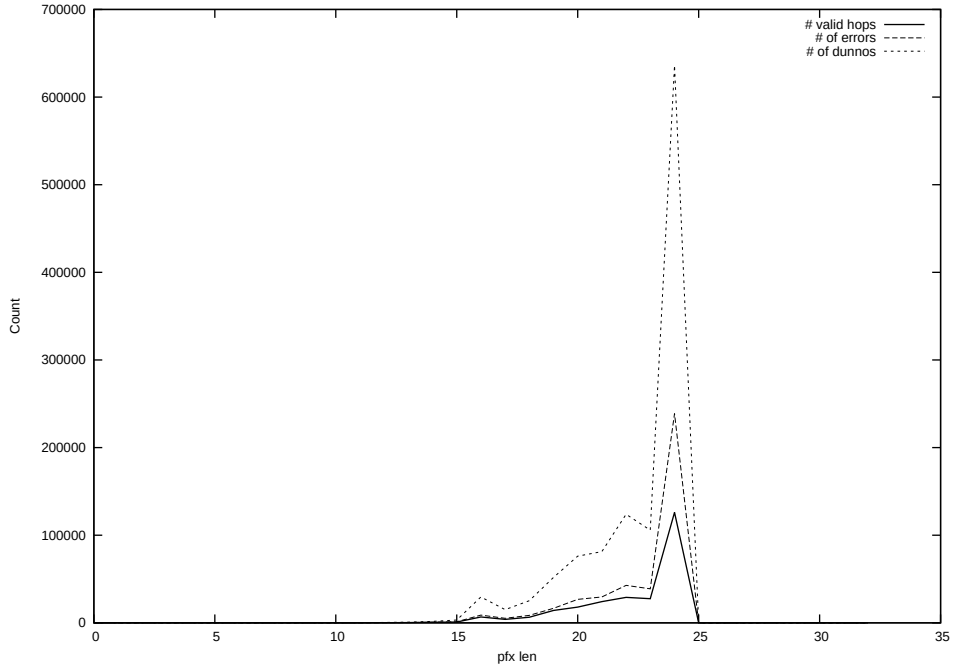
2014-04-11



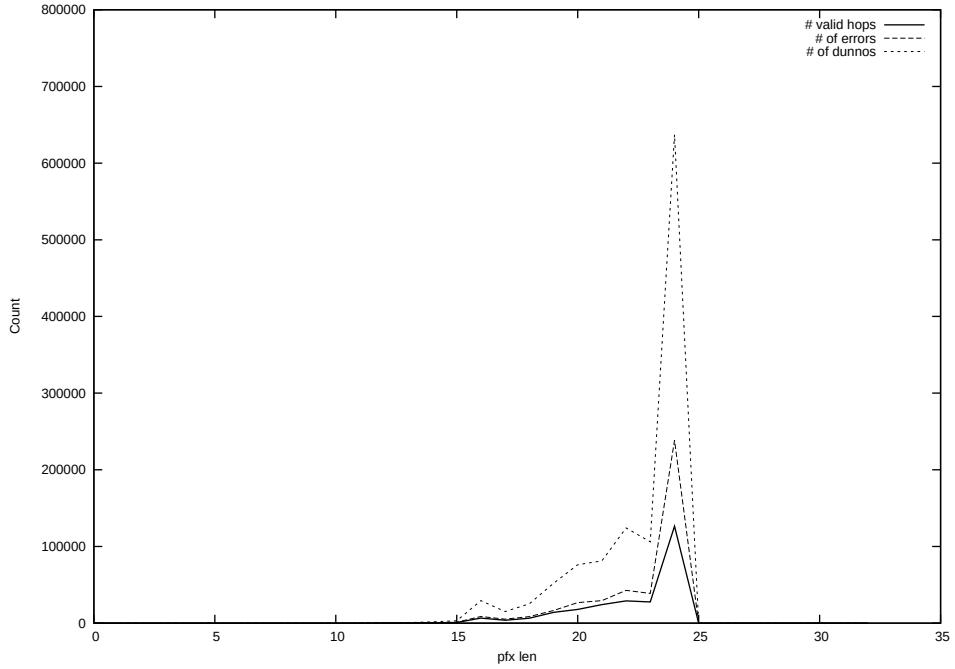
2014-04-12



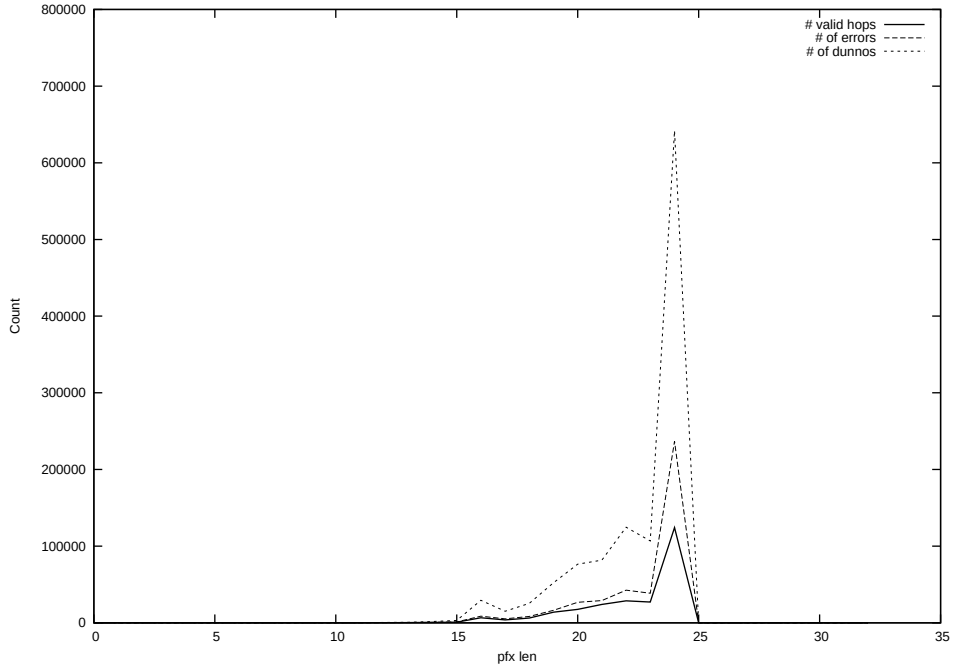
2014-04-13



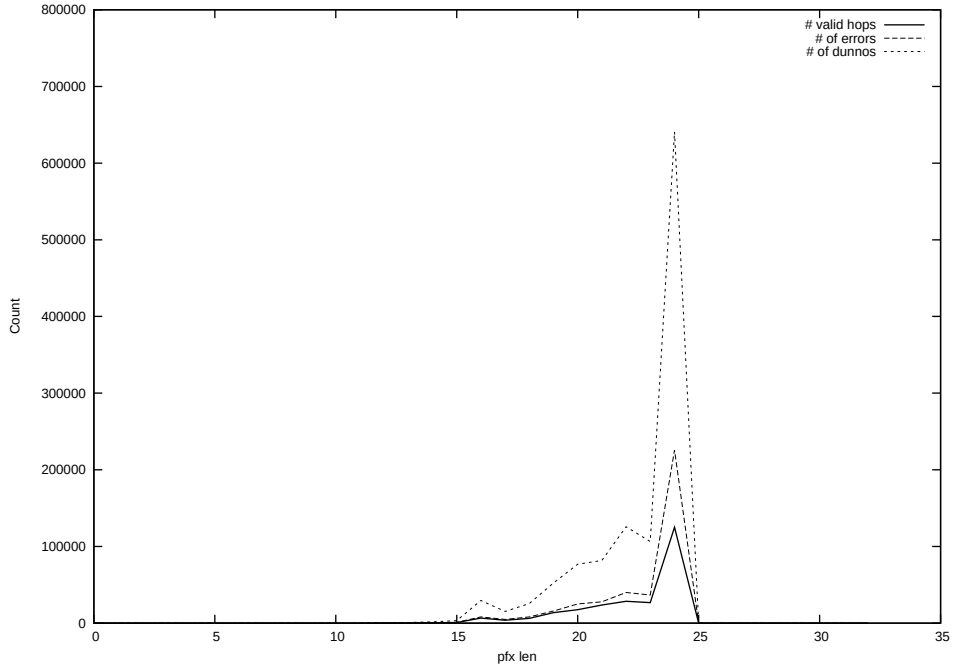
2014-04-14



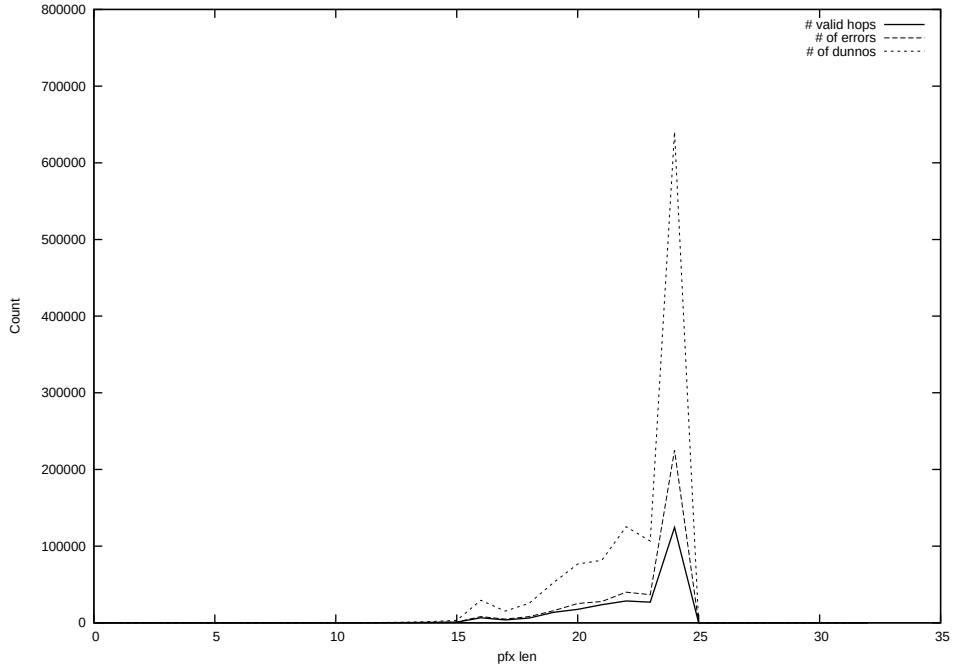
2014-04-15



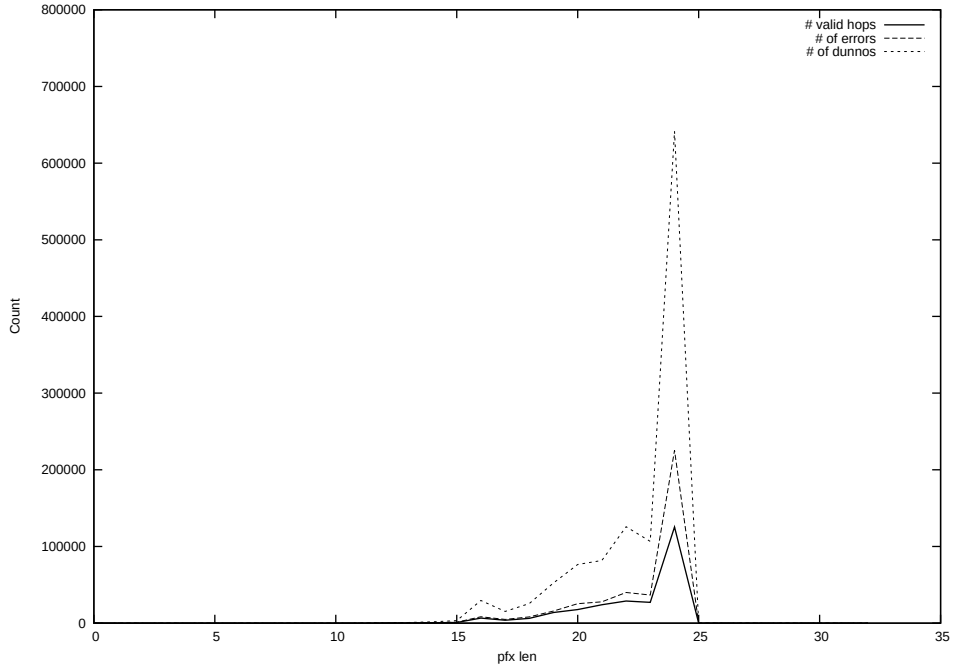
2014-04-16



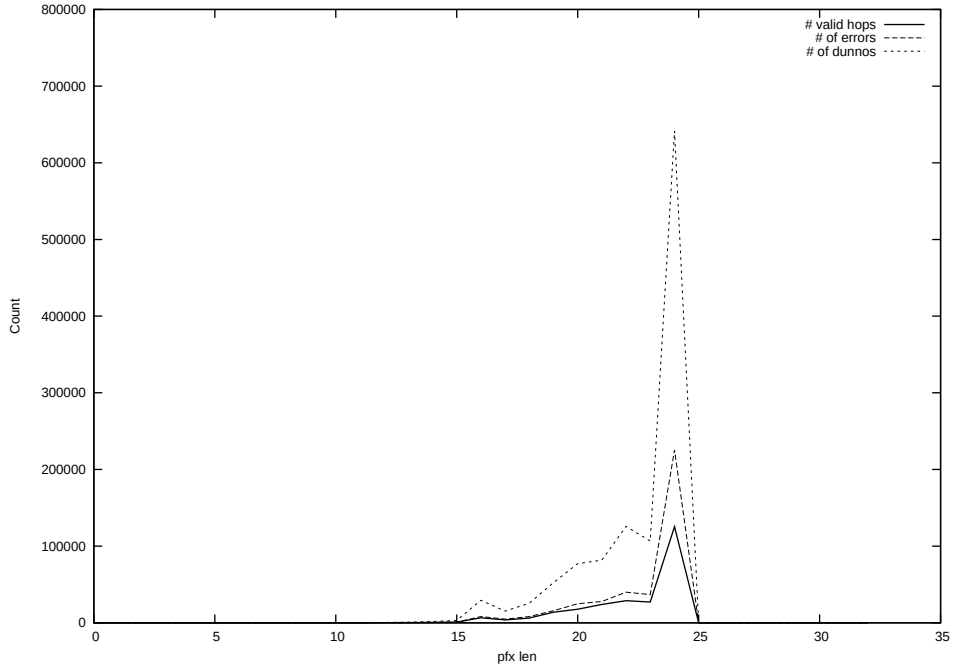
2014-04-17



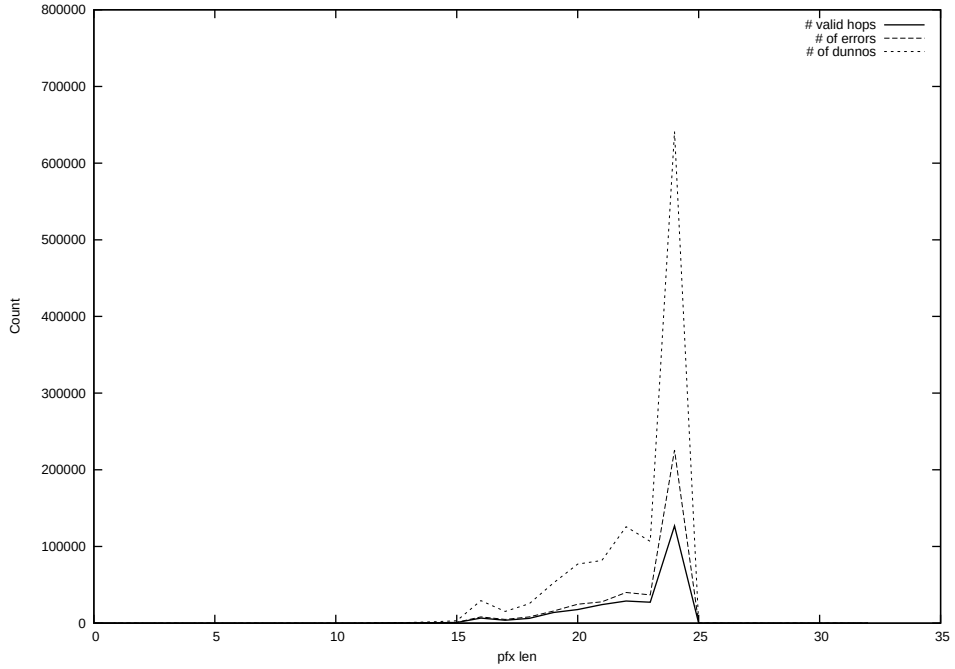
2014-04-18



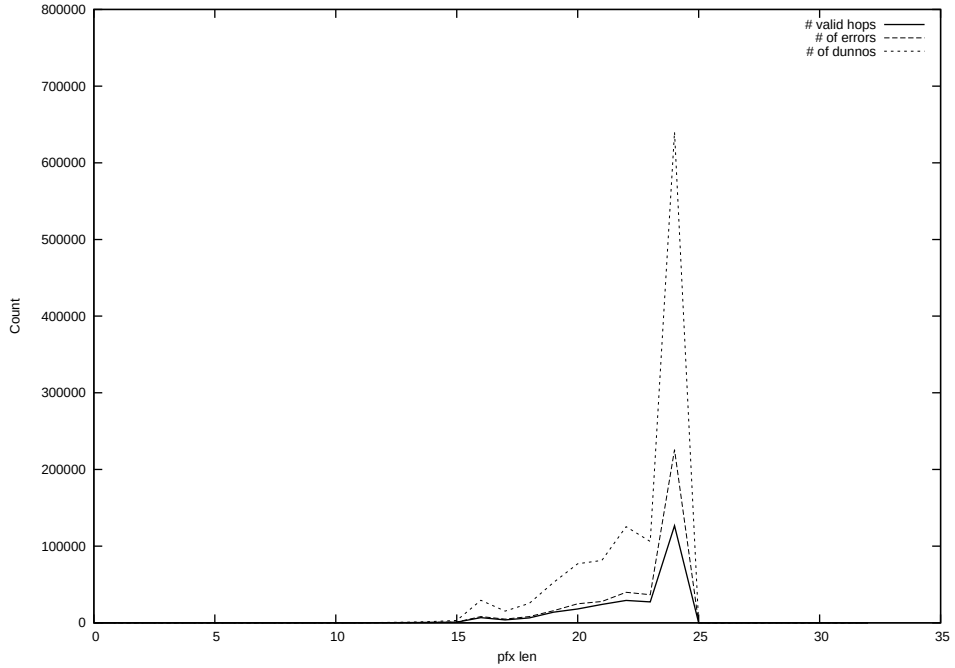
2014-04-19



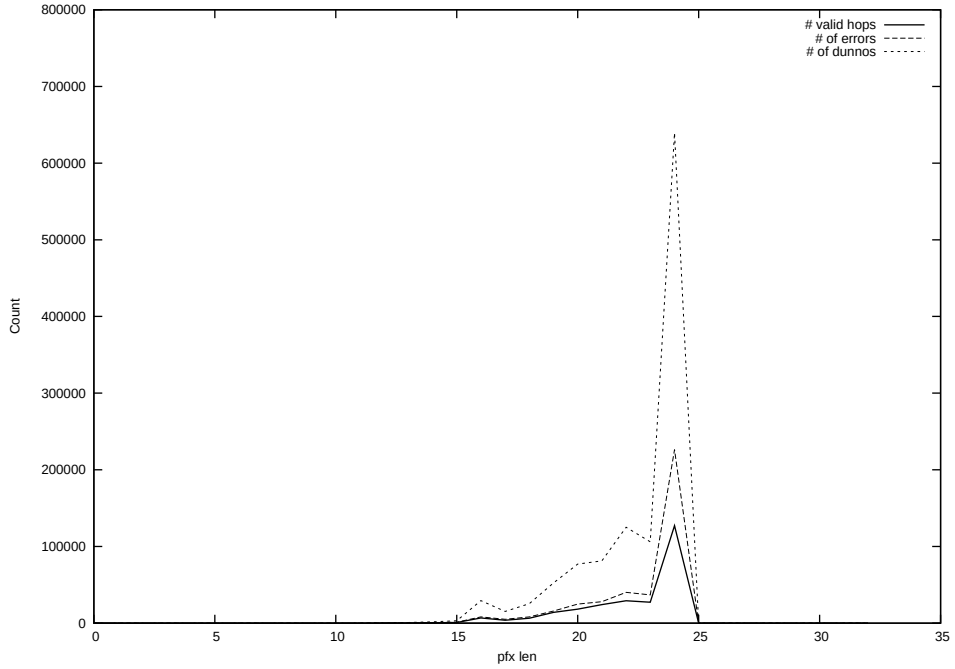
2014-04-20



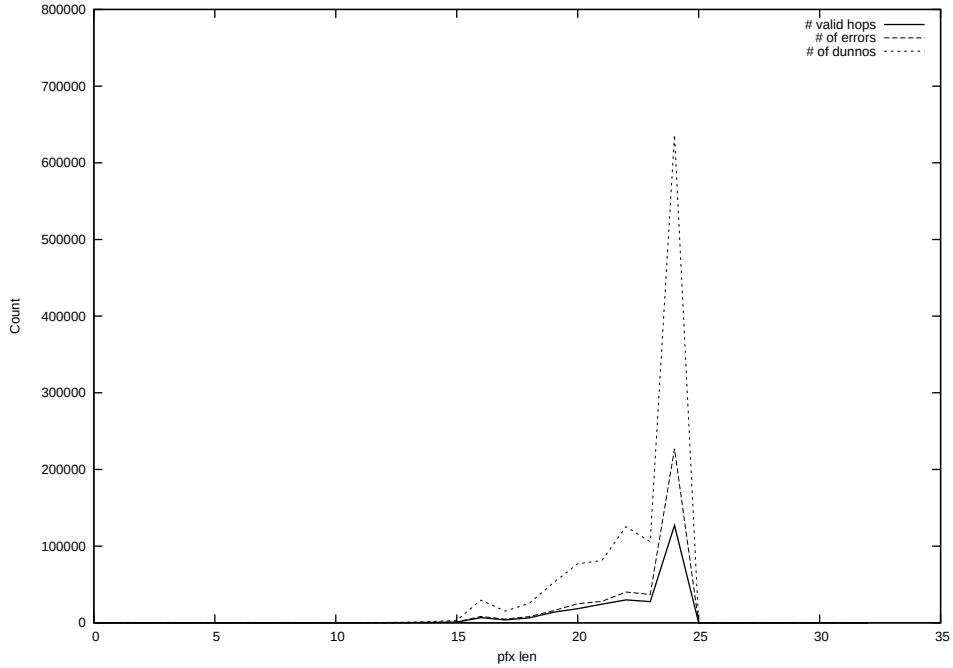
2014-04-21



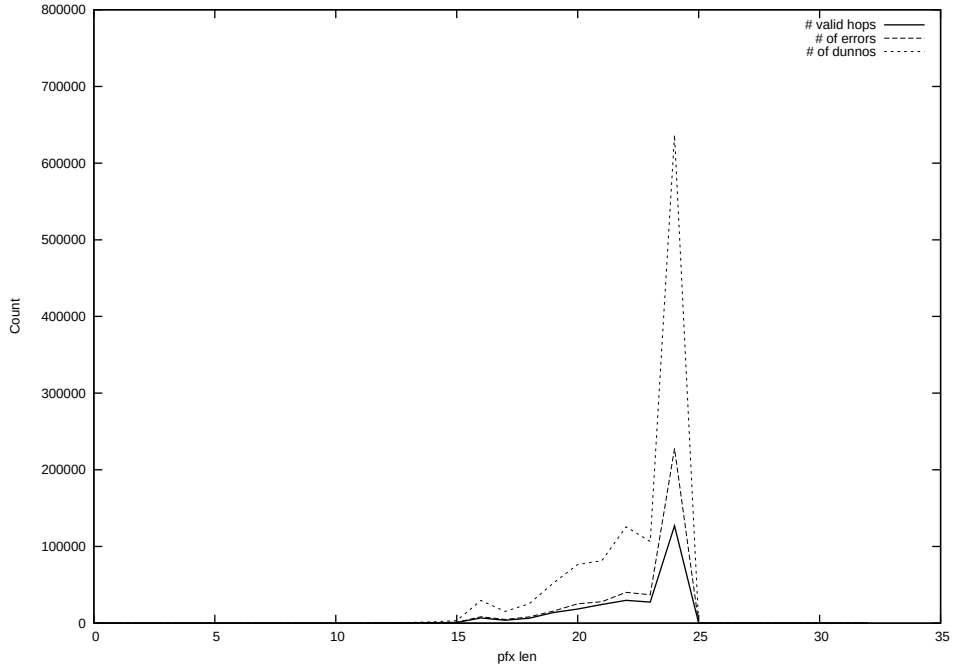
2014-04-22



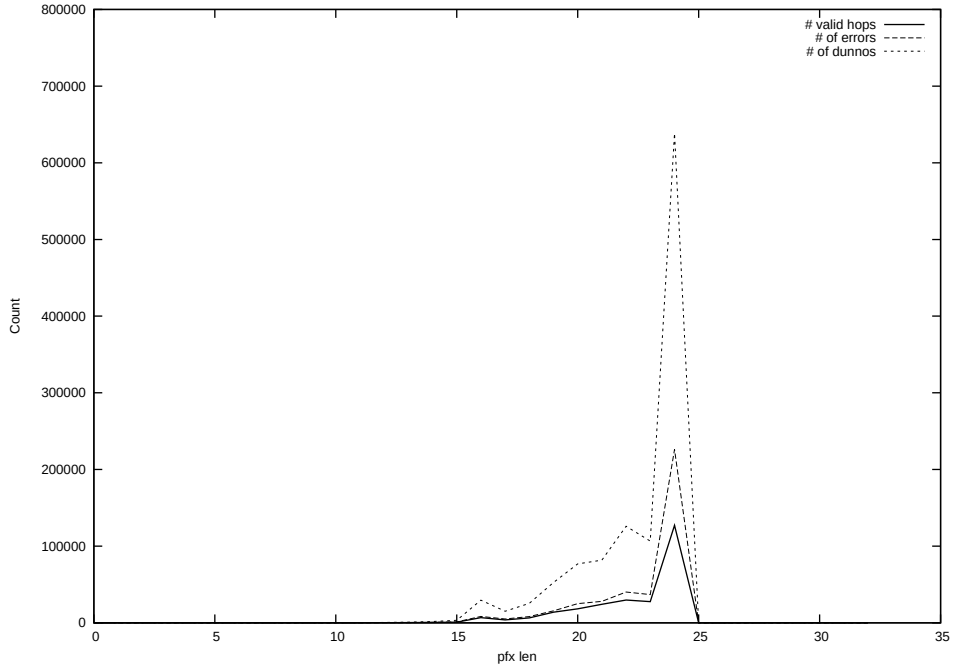
2014-04-23



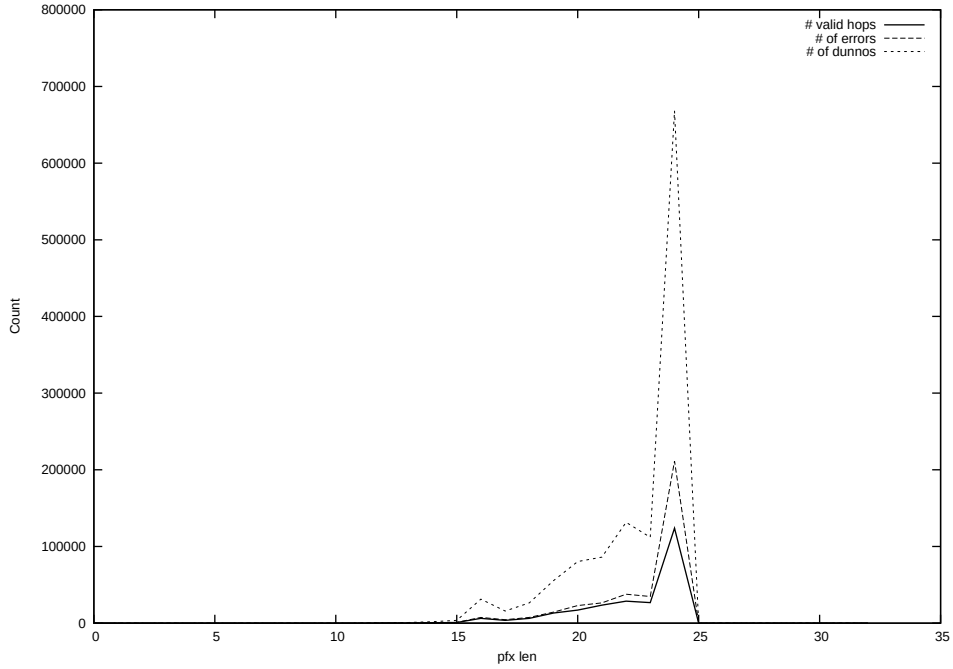
2014-04-24



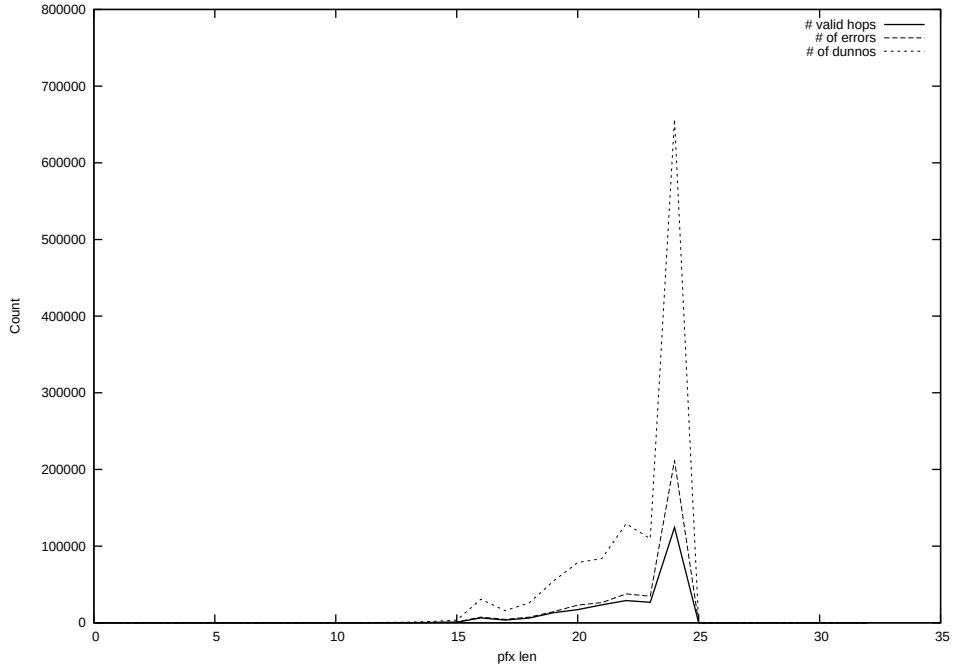
2014-04-25



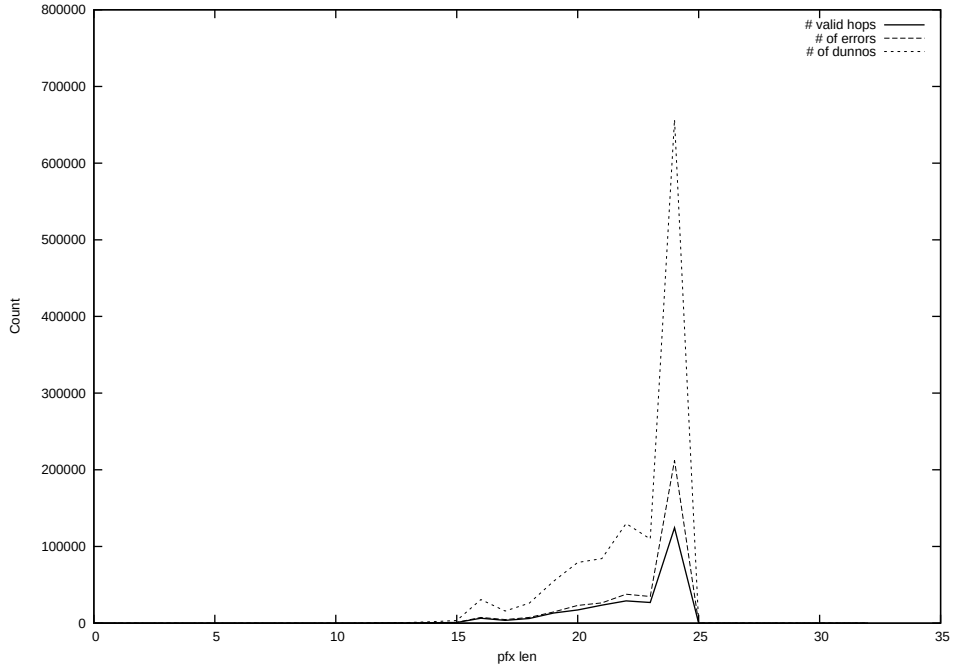
2014-04-26



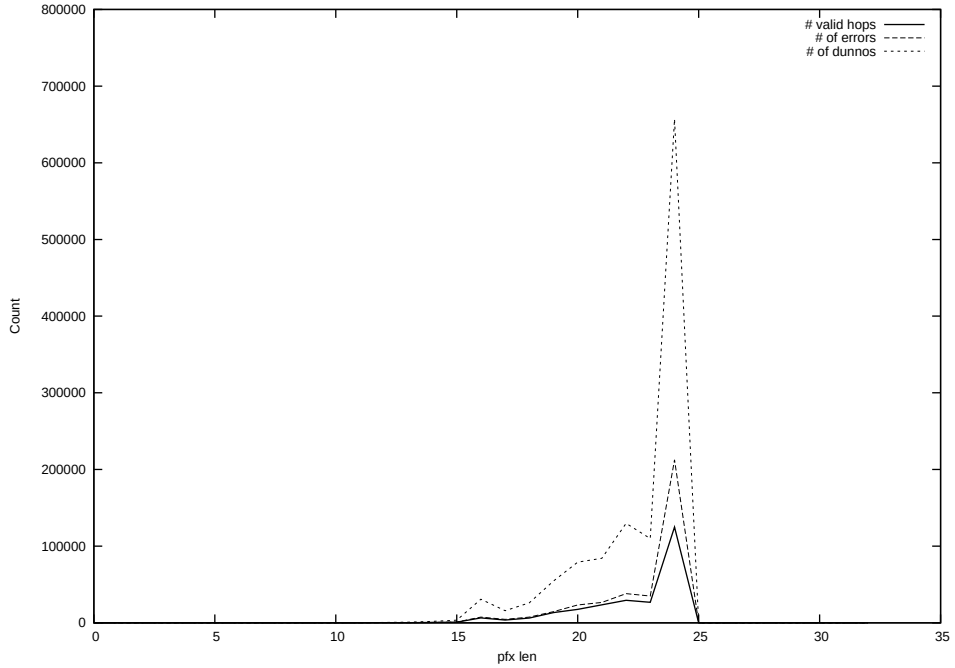
2014-04-27



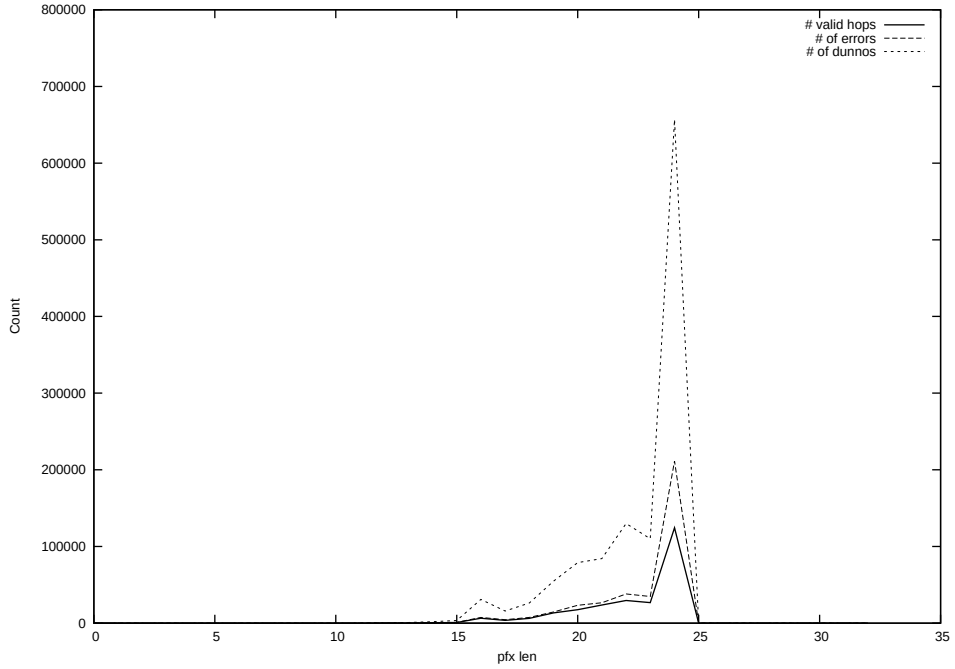
2014-04-28



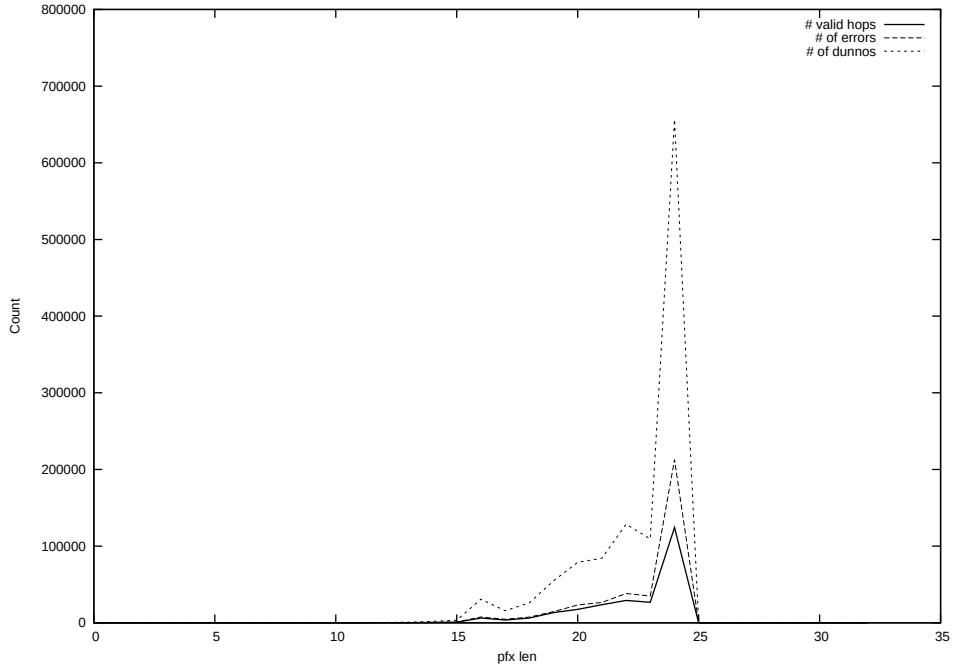
2014-04-29



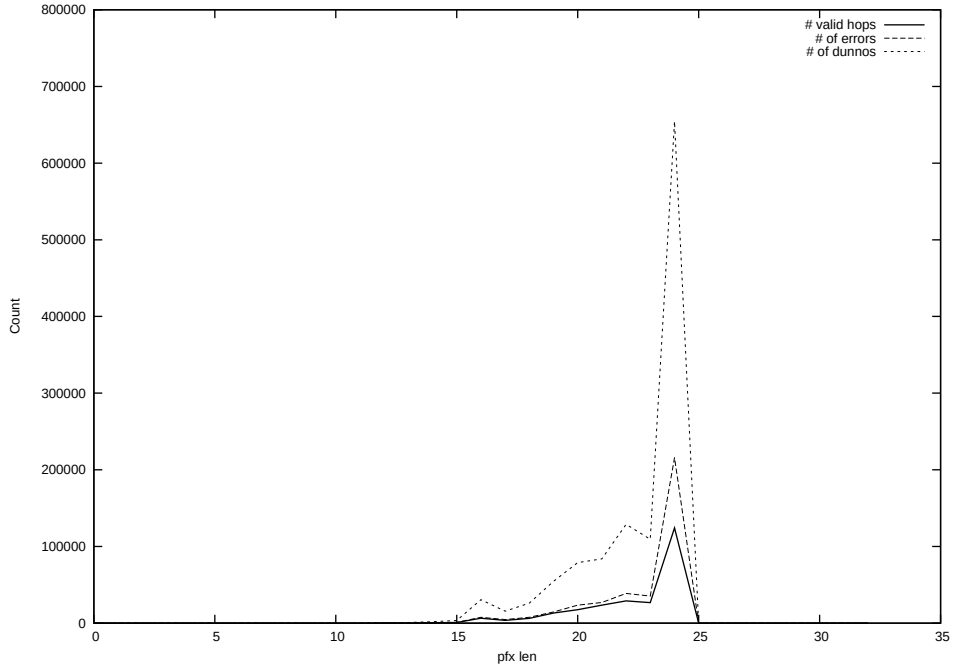
2014-04-30



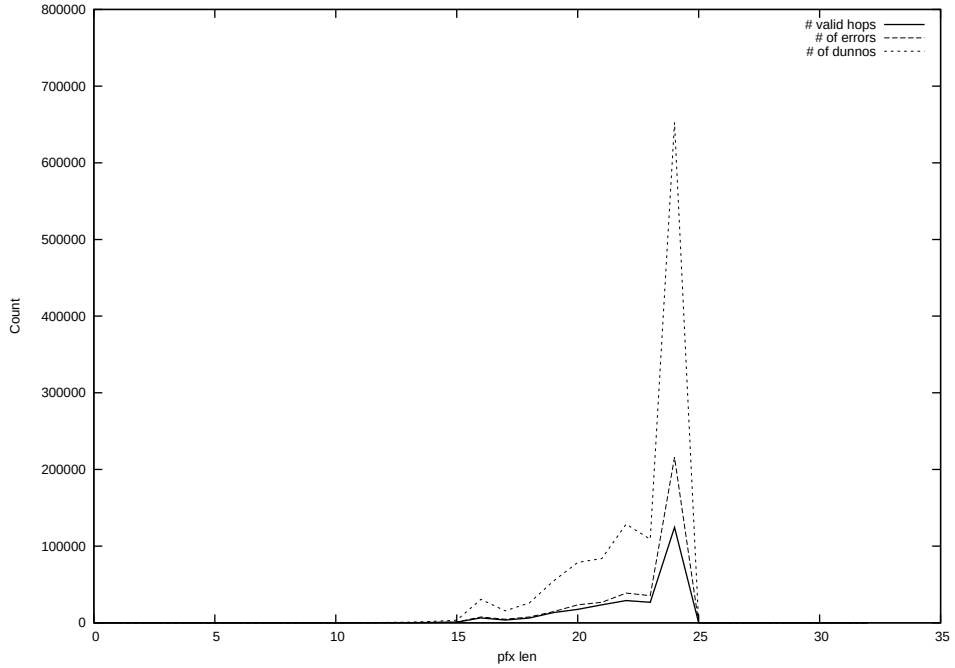
2014-05-01



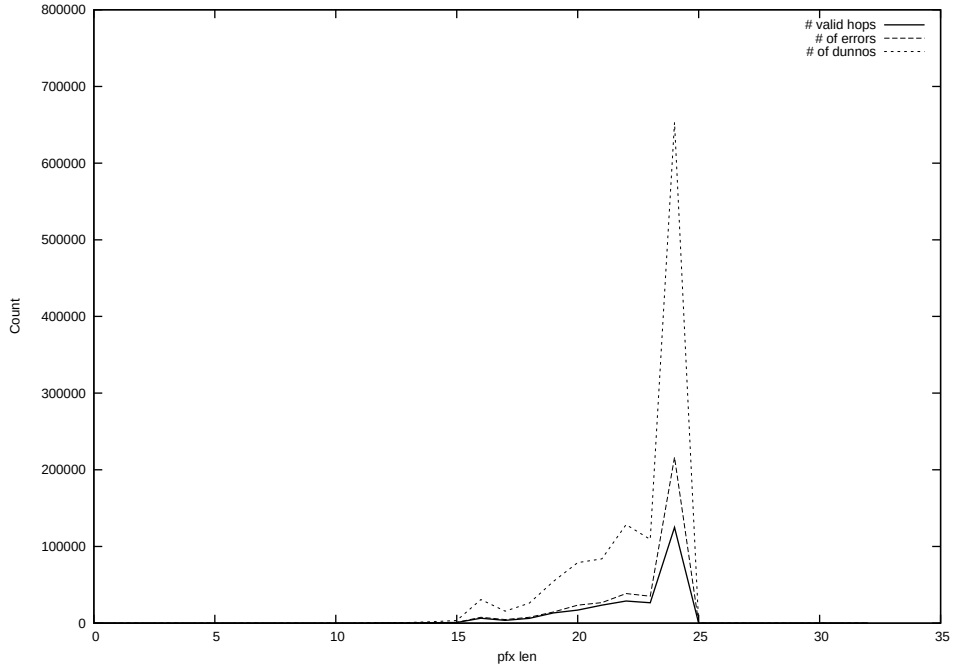
2014-05-02



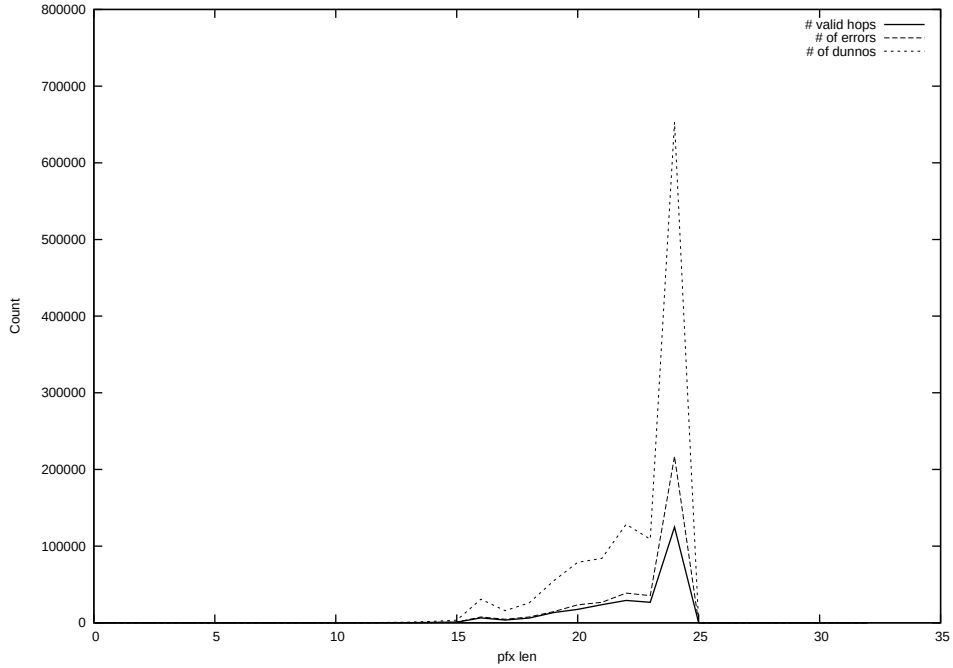
2014-05-03



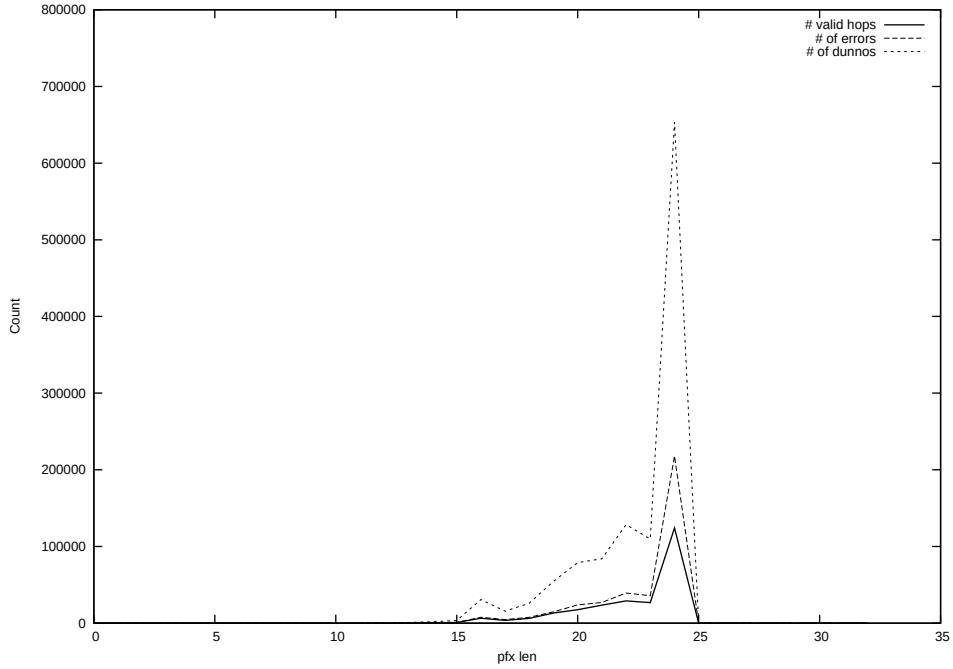
2014-05-04



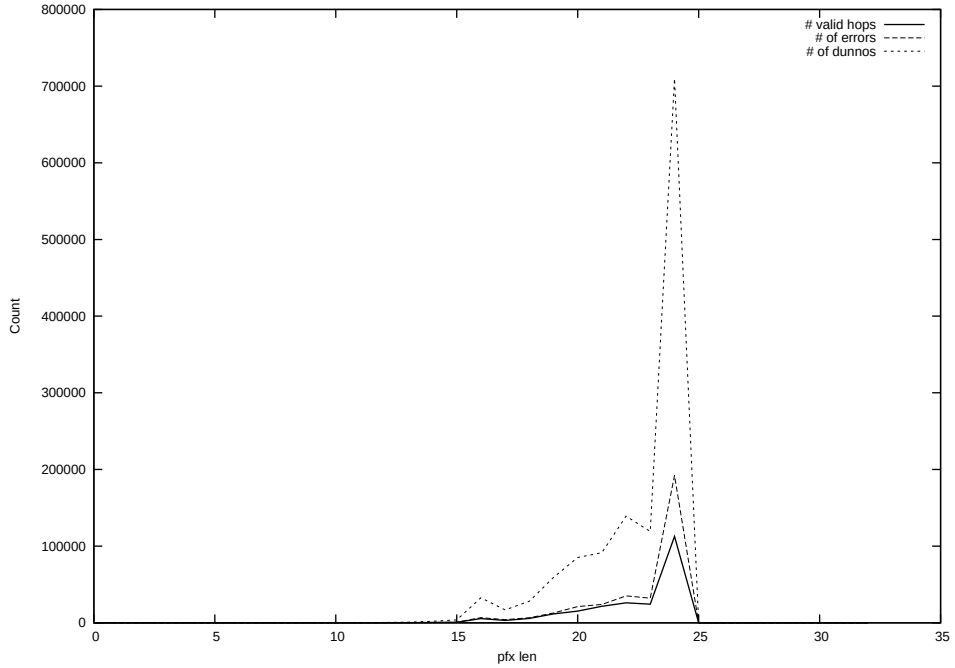
2014-05-05



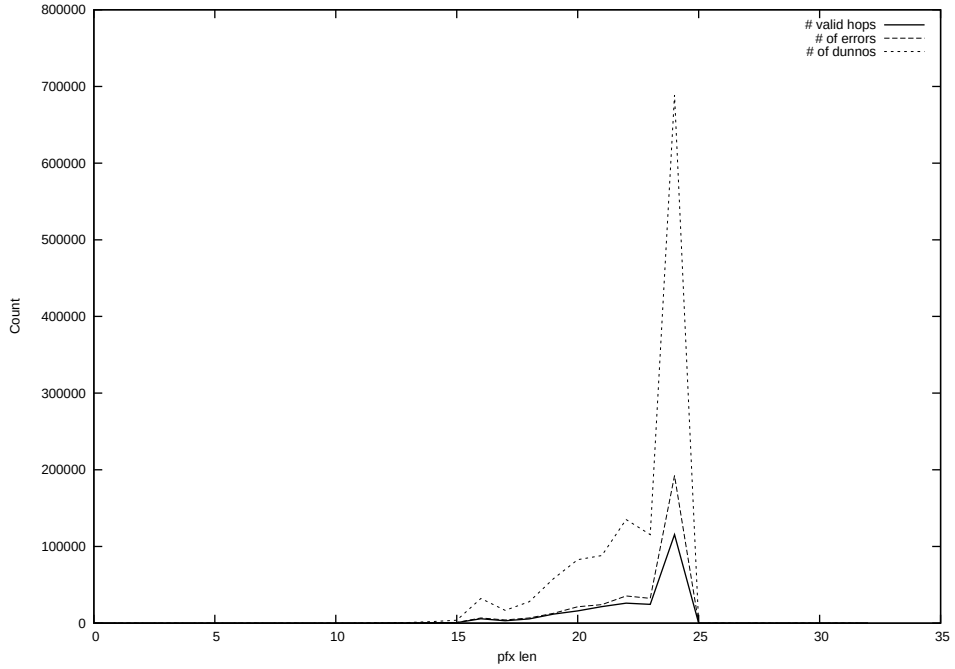
2014-05-06



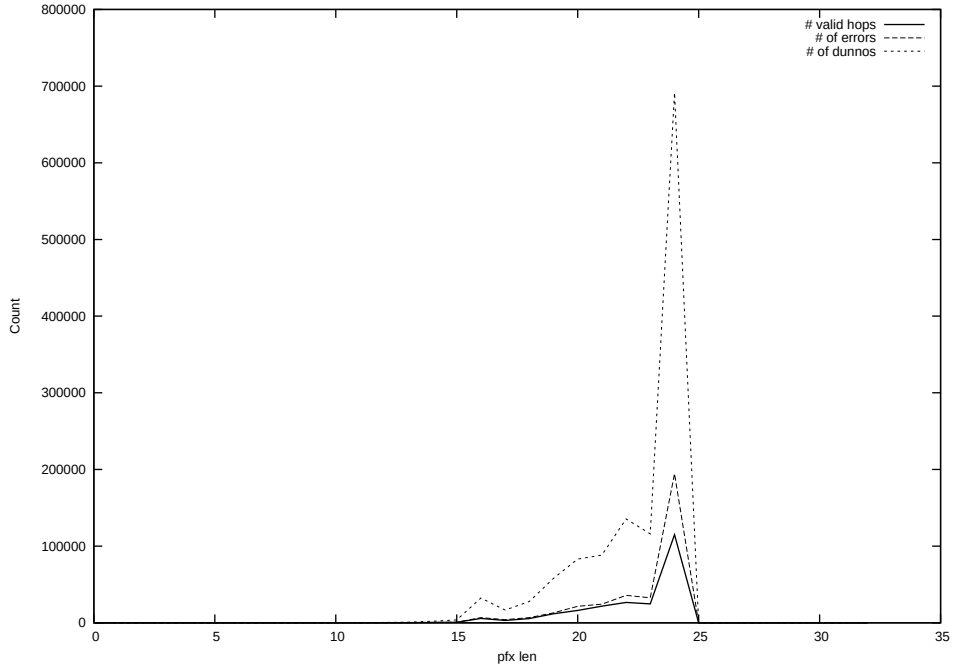
2014-05-07



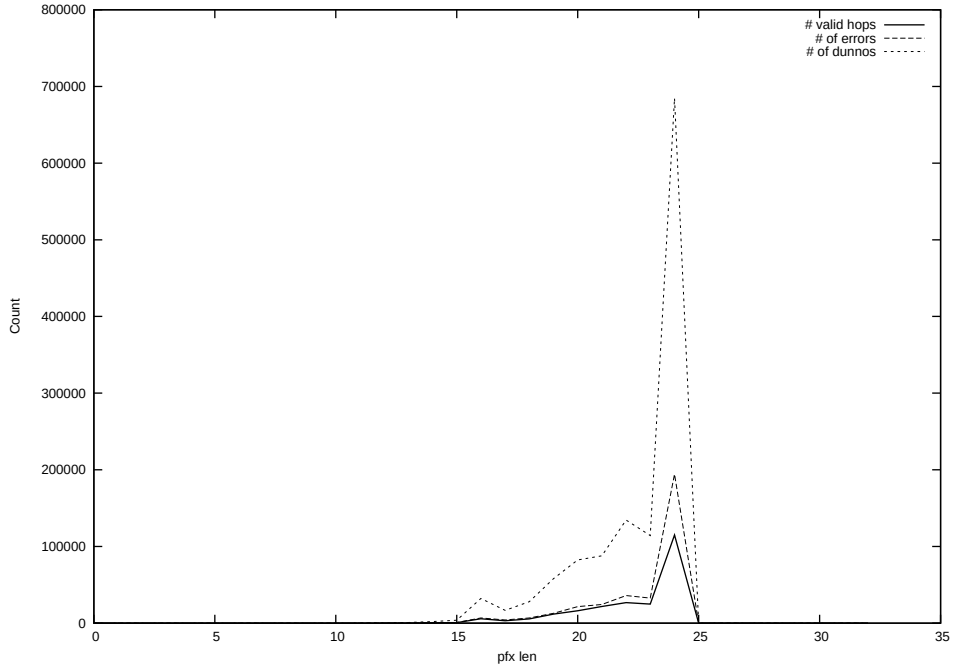
2014-05-08



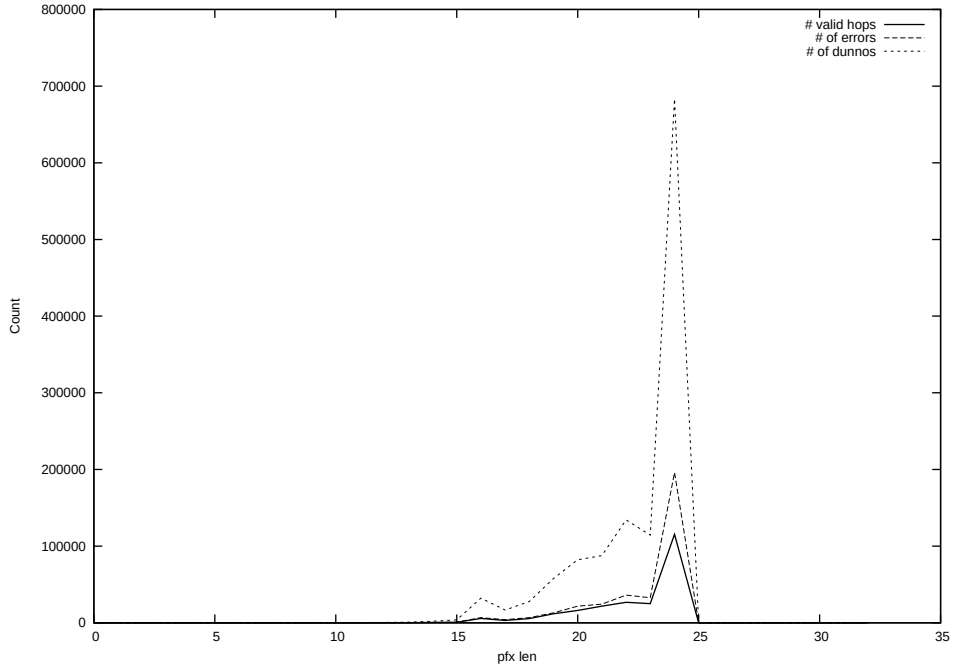
2014-05-09



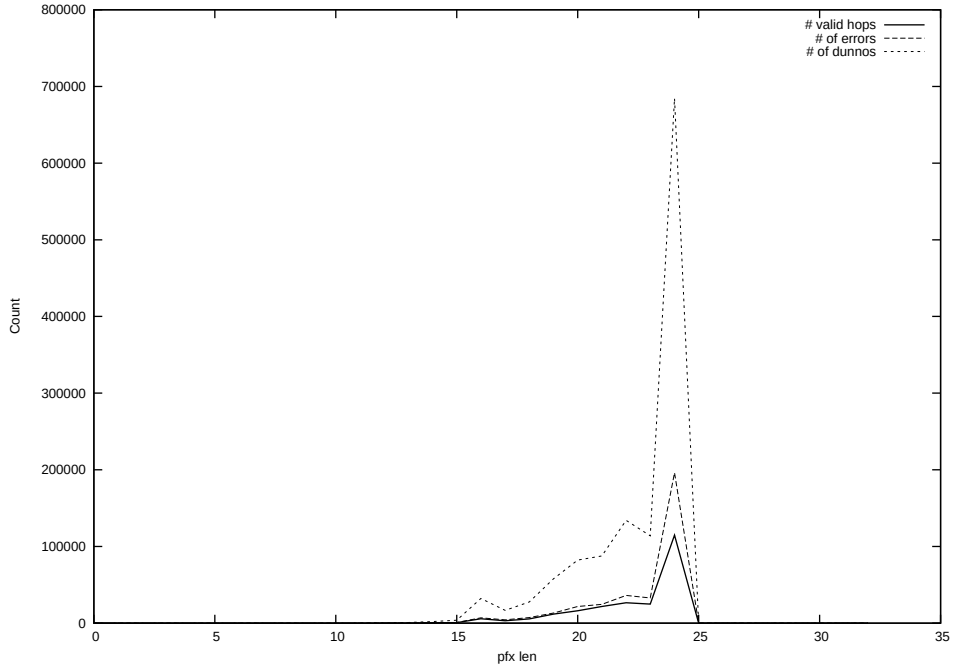
2014-05-10



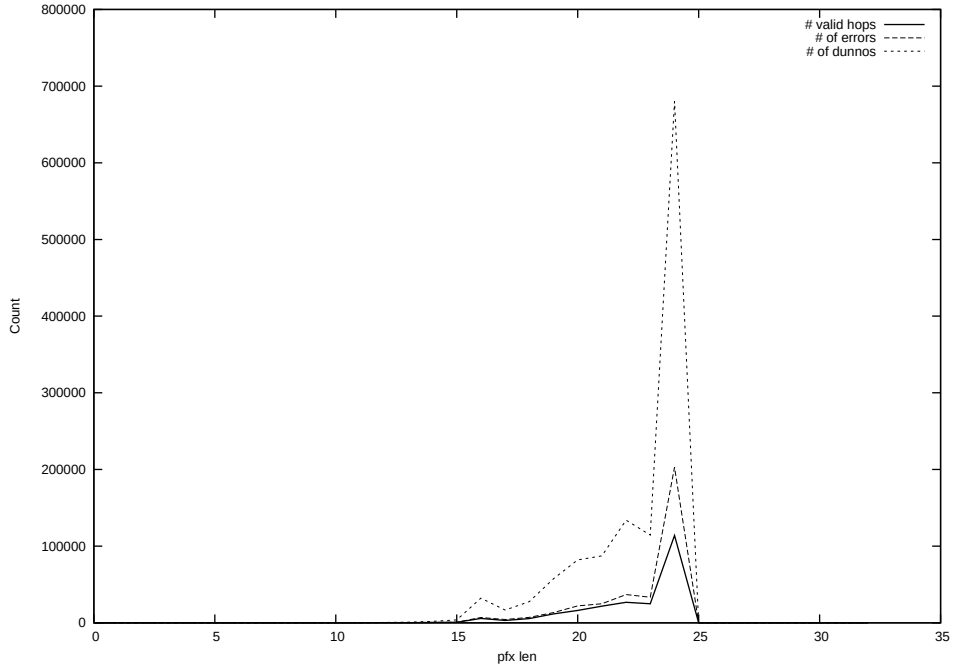
2014-05-11



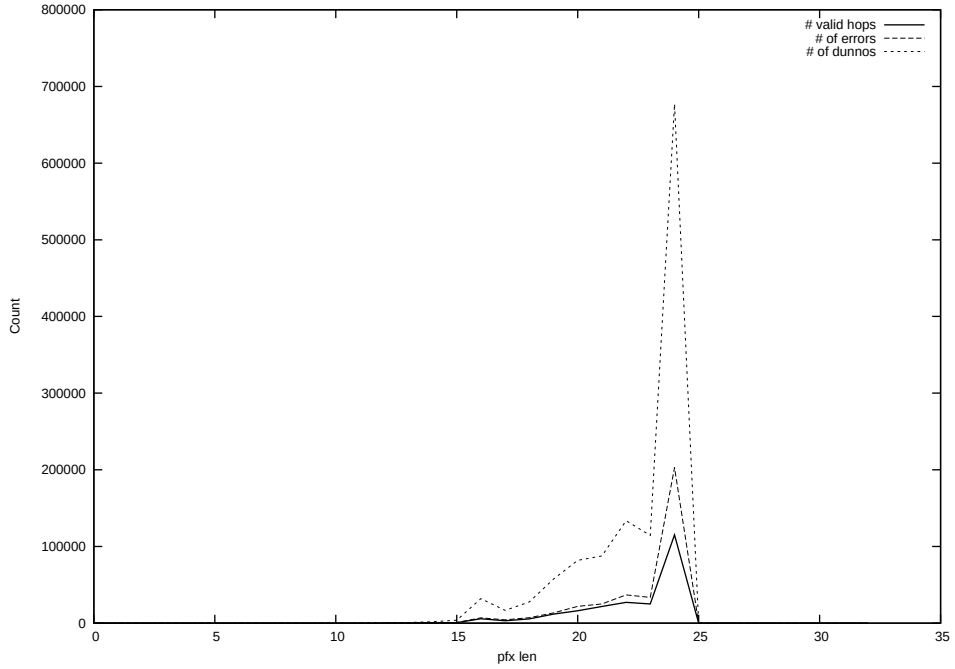
2014-05-12



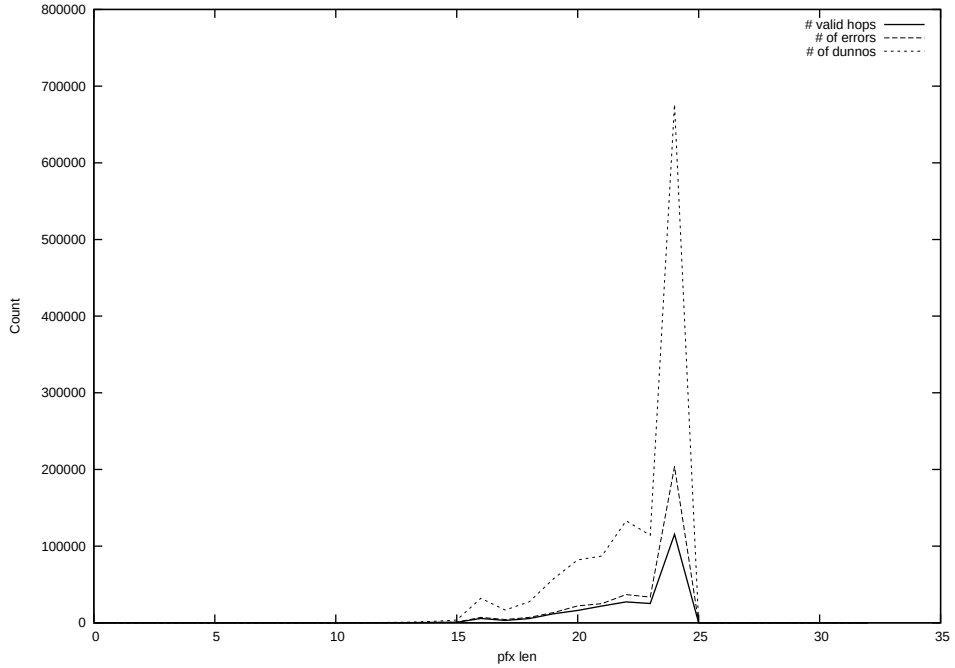
2014-05-13



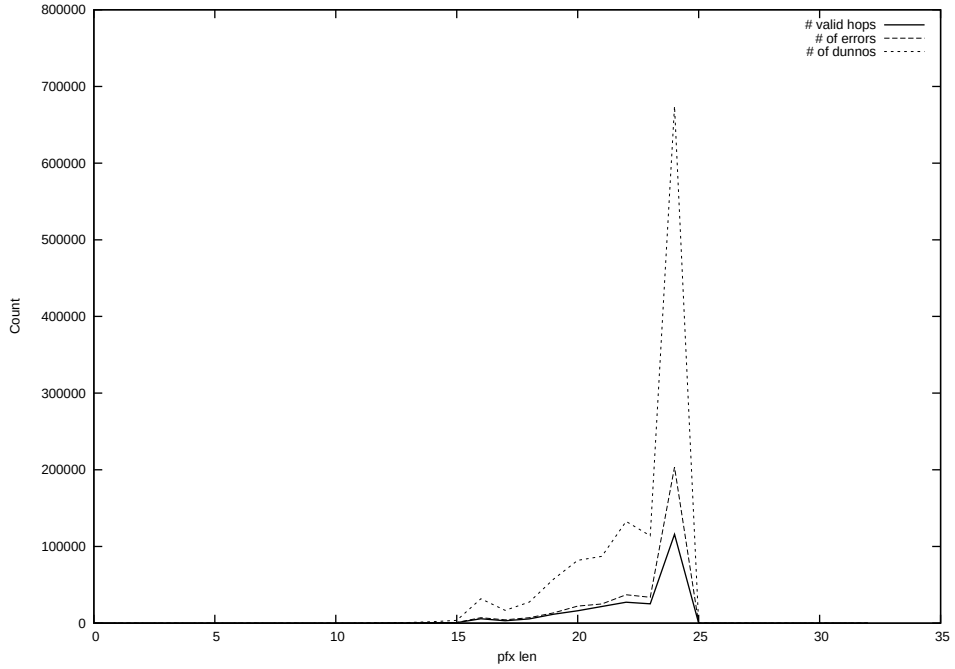
2014-05-14



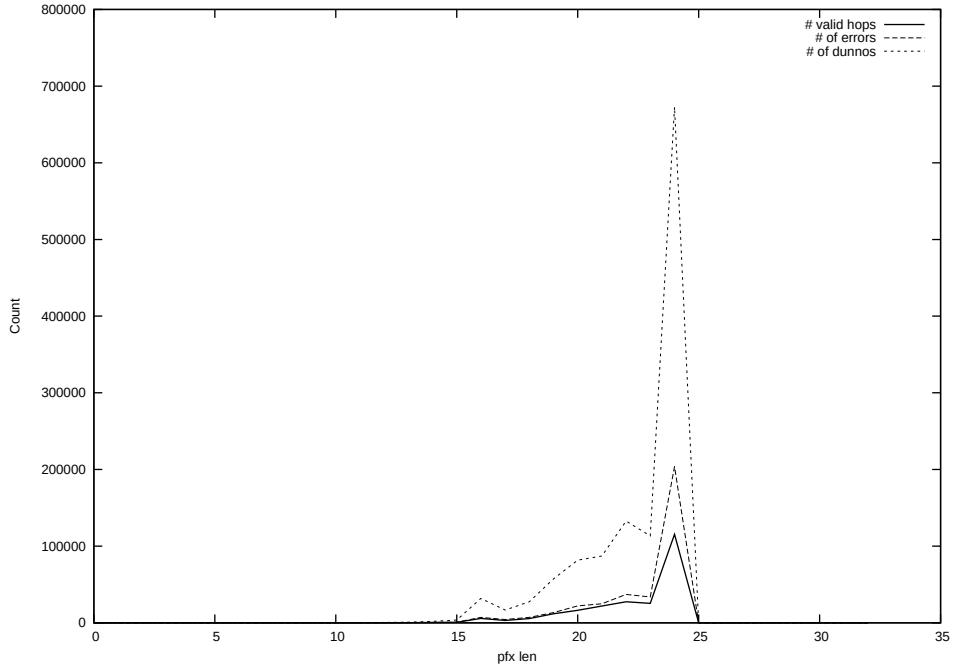
2014-05-15



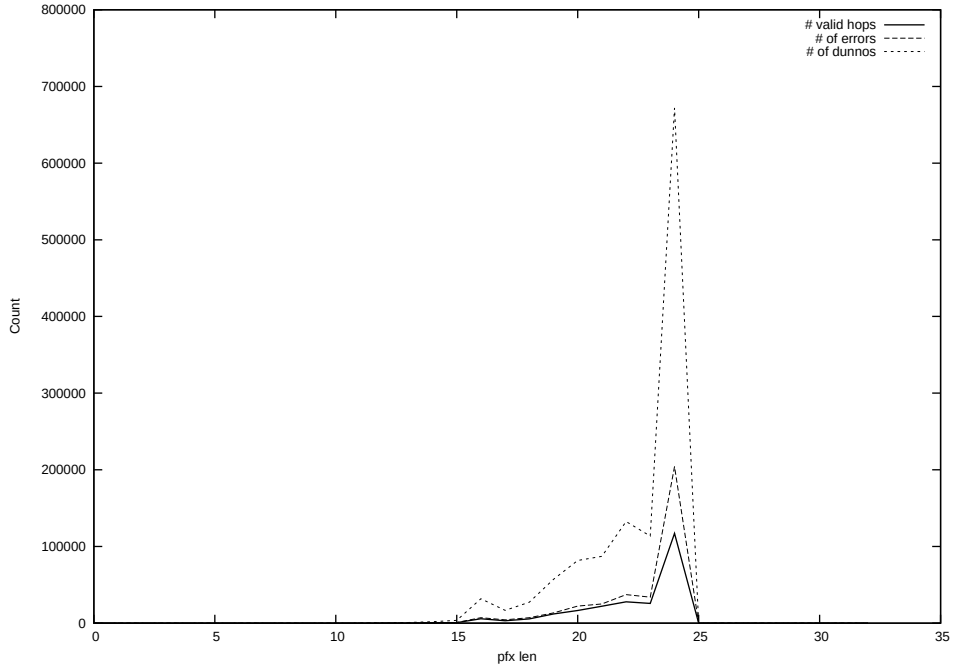
2014-05-16



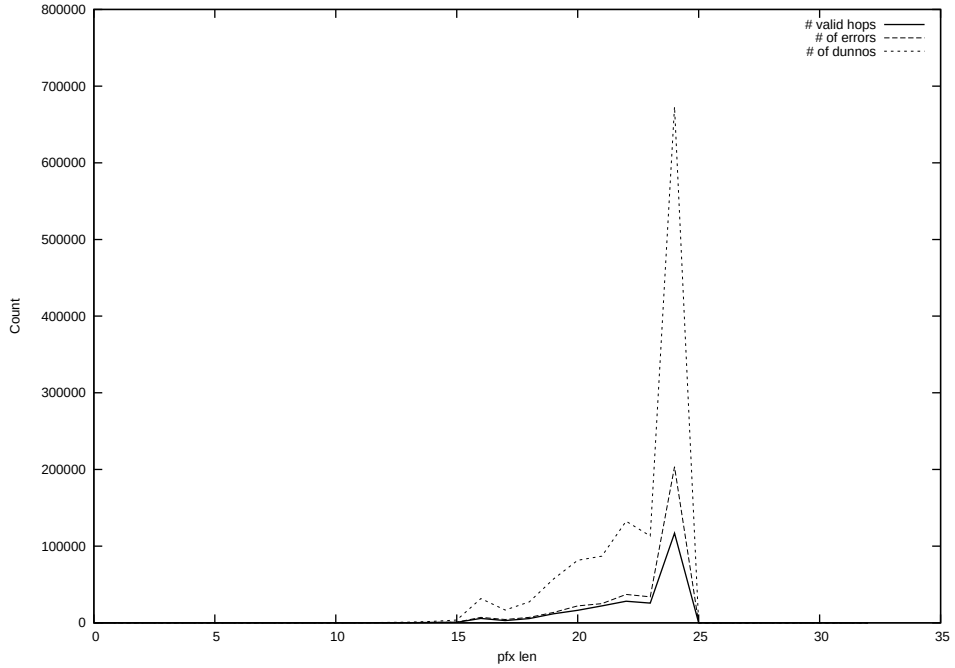
2014-05-17



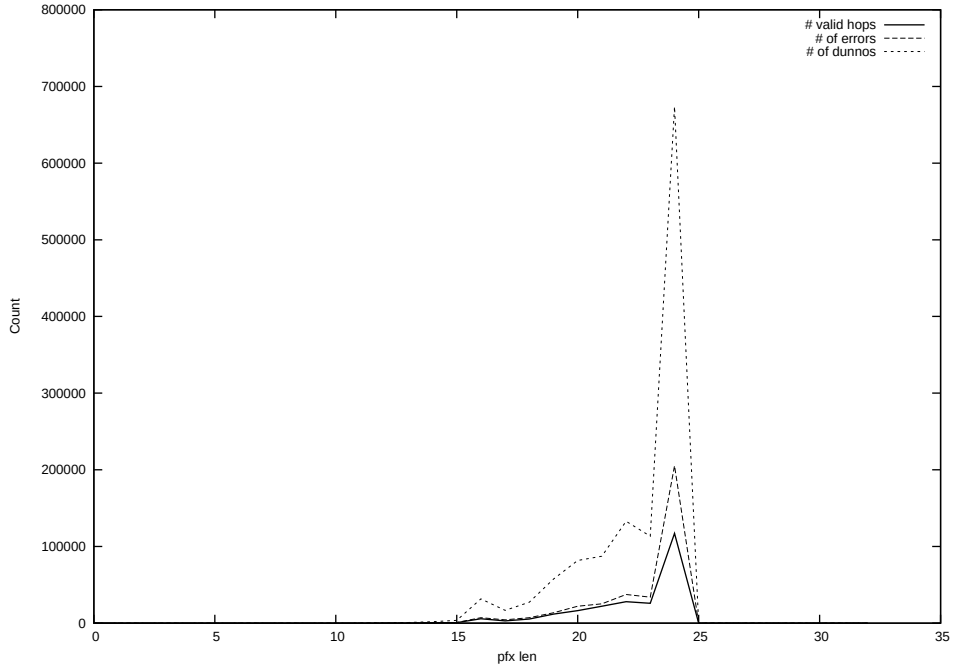
2014-05-18



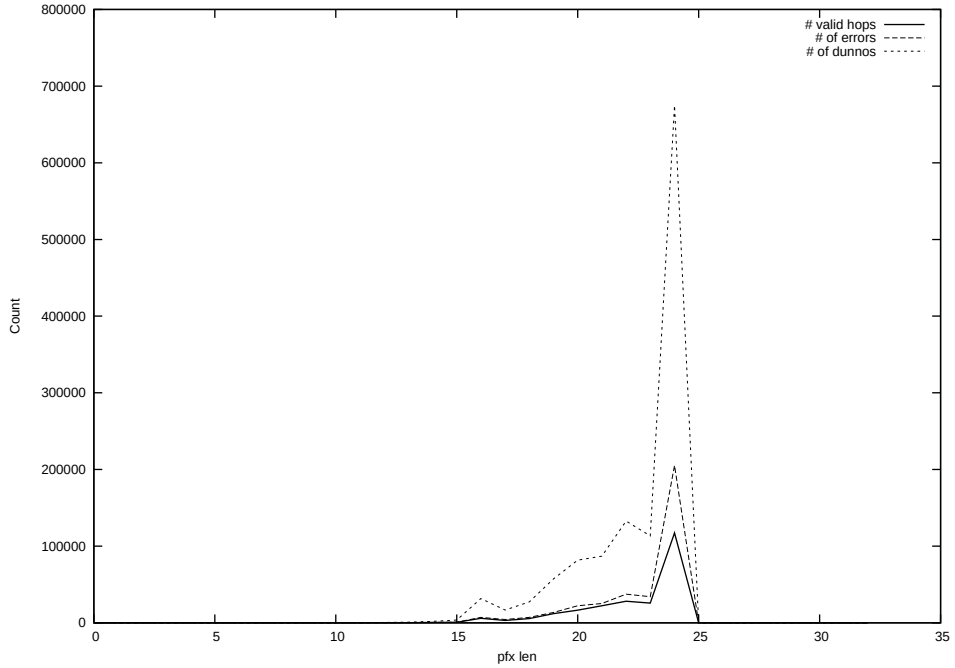
2014-05-19



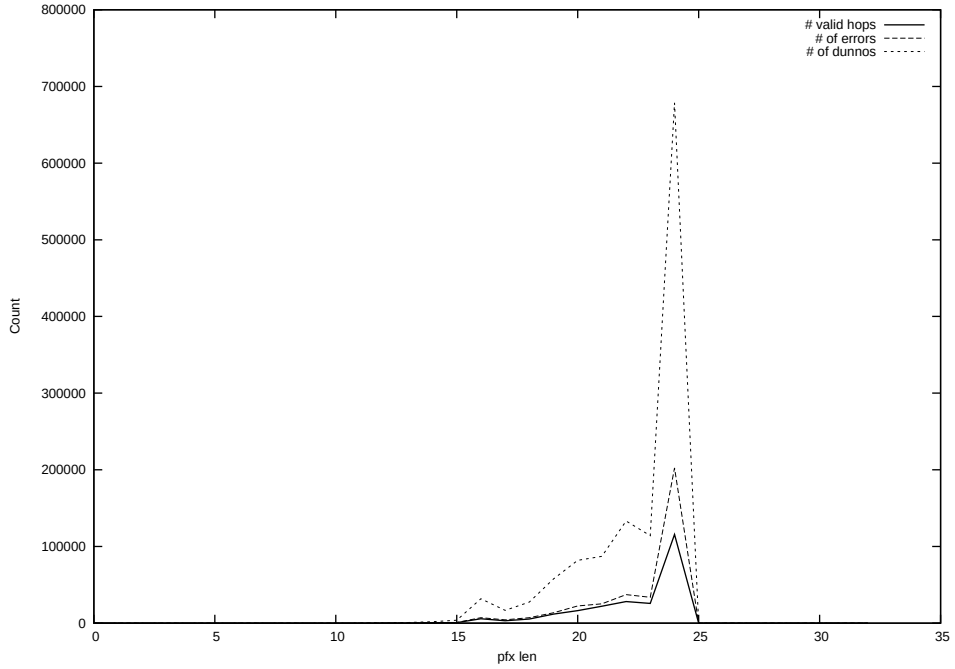
2014-05-20



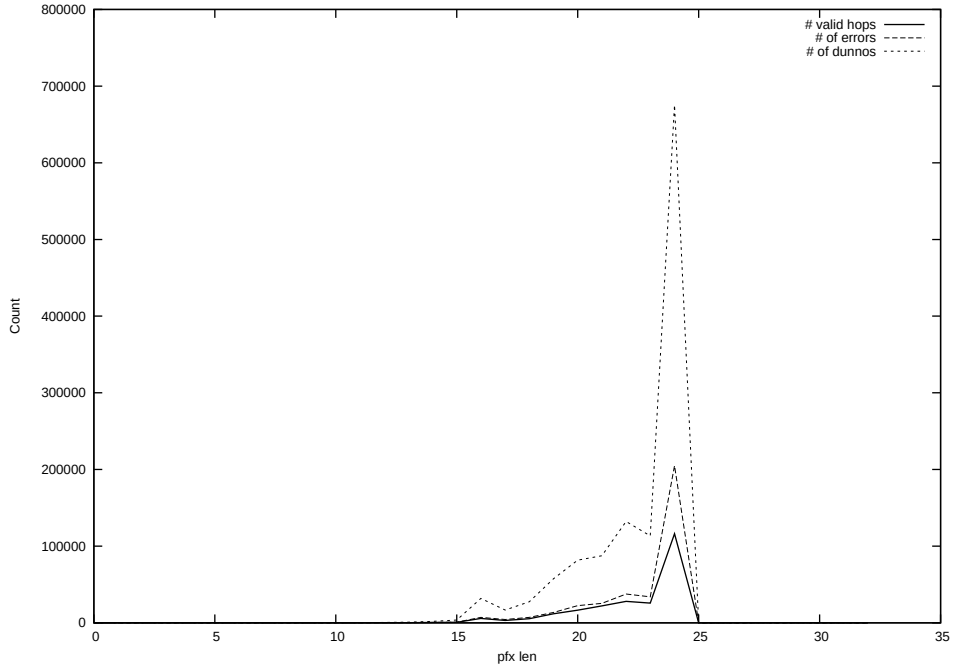
2014-05-21



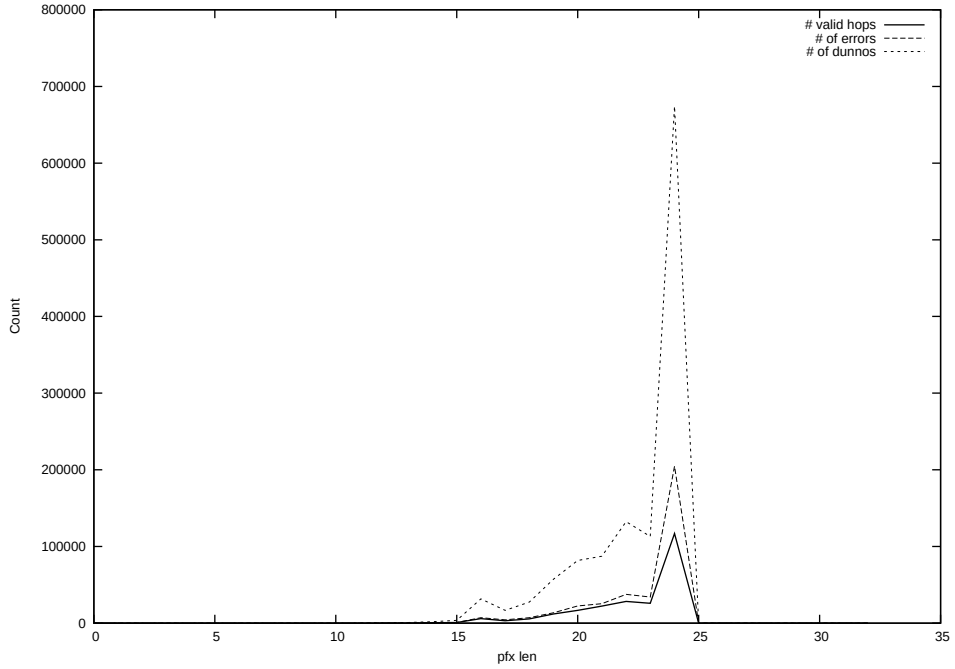
2014-05-22



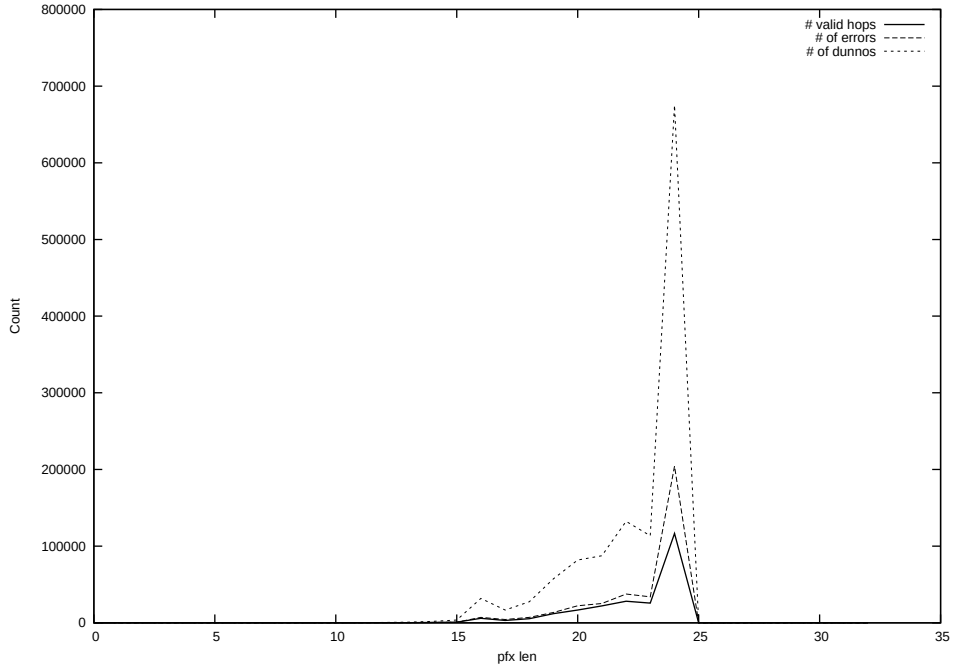
2014-05-23



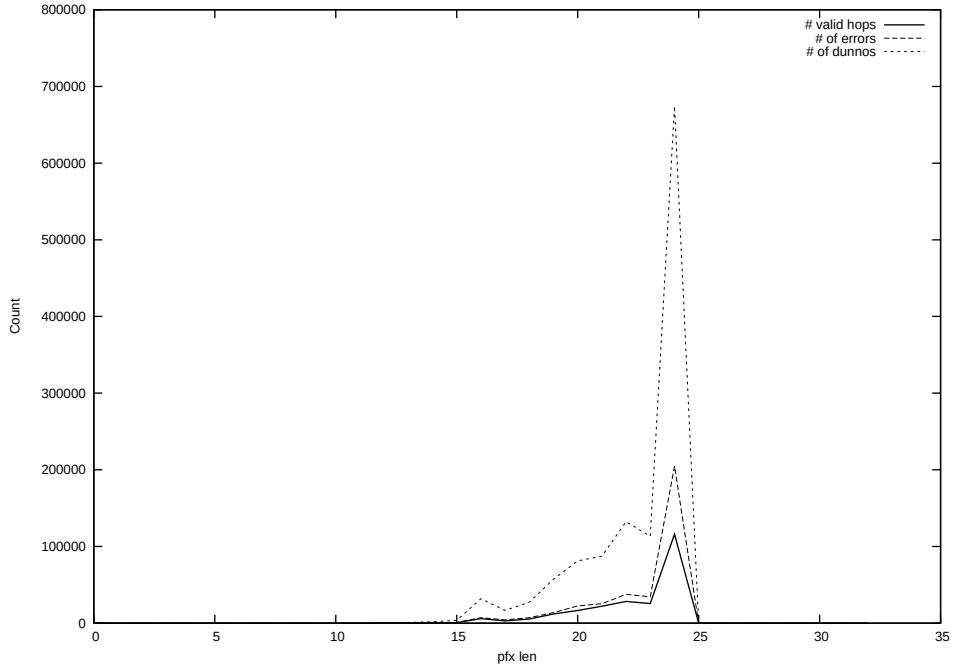
2014-05-24



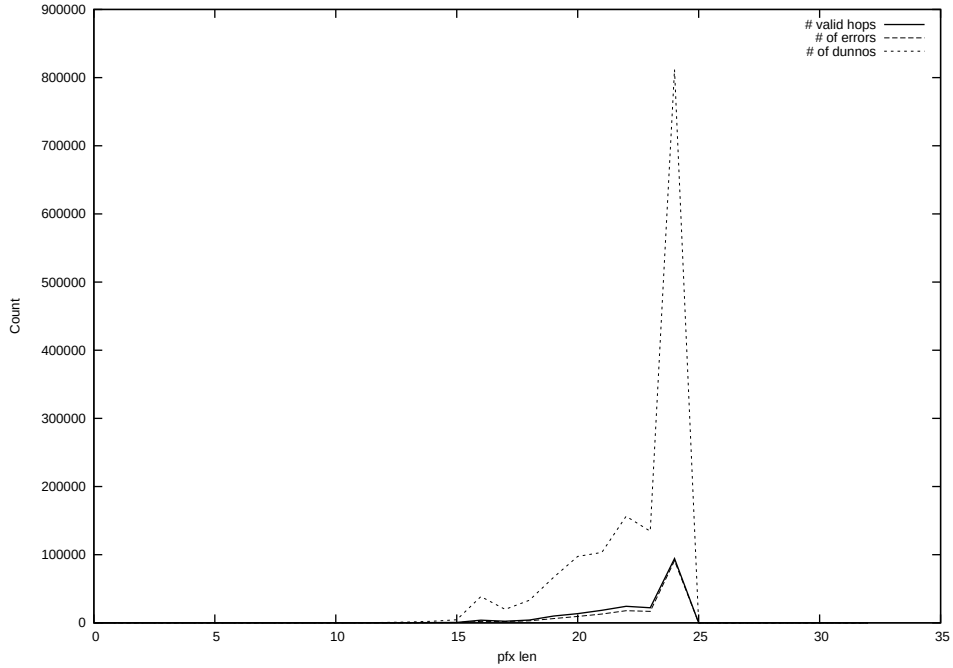
2014-05-25



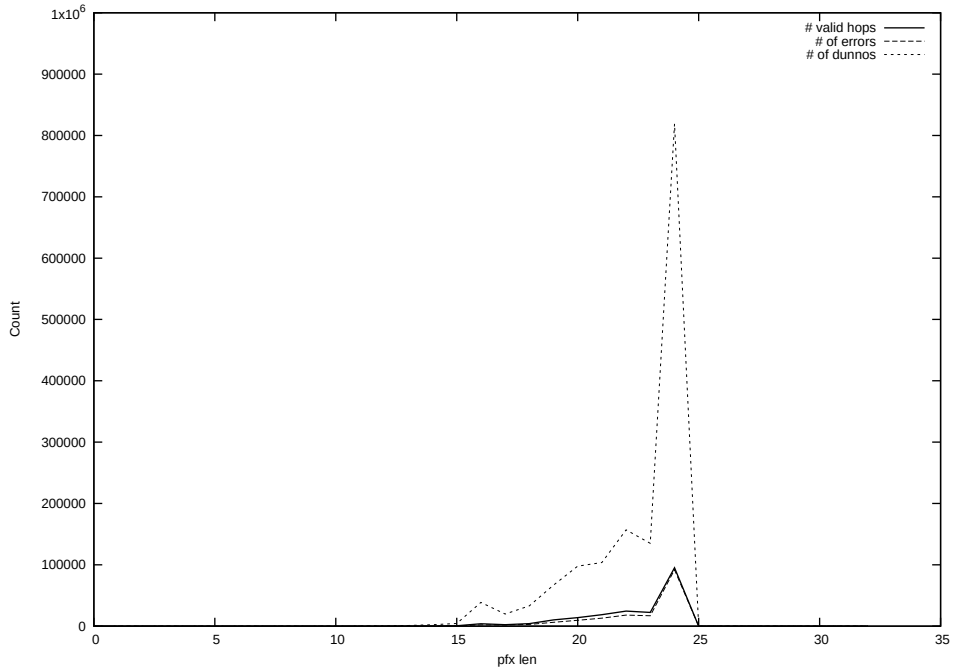
2014-05-26



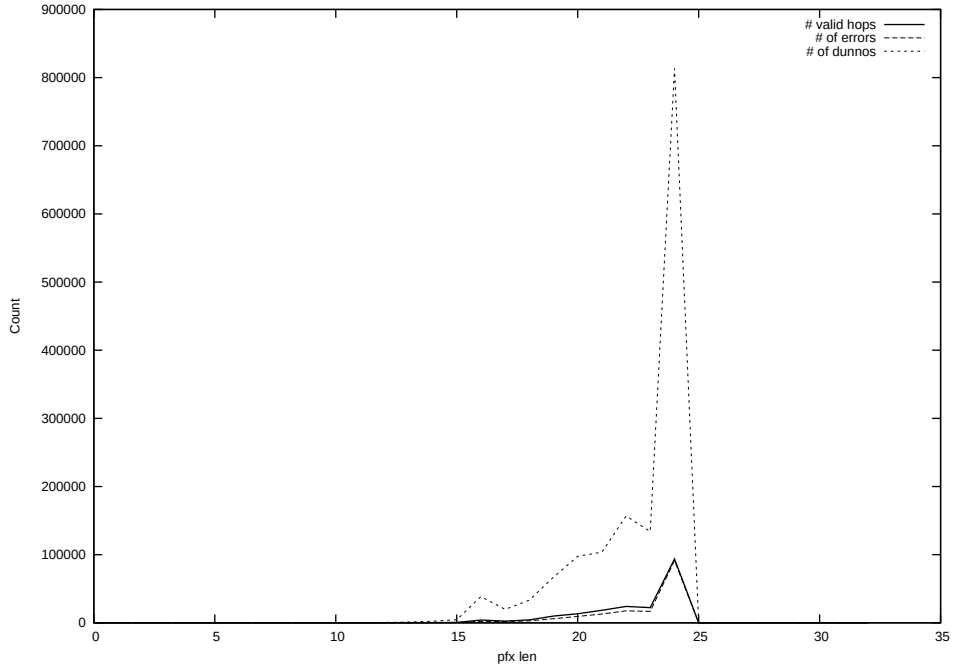
2014-05-27



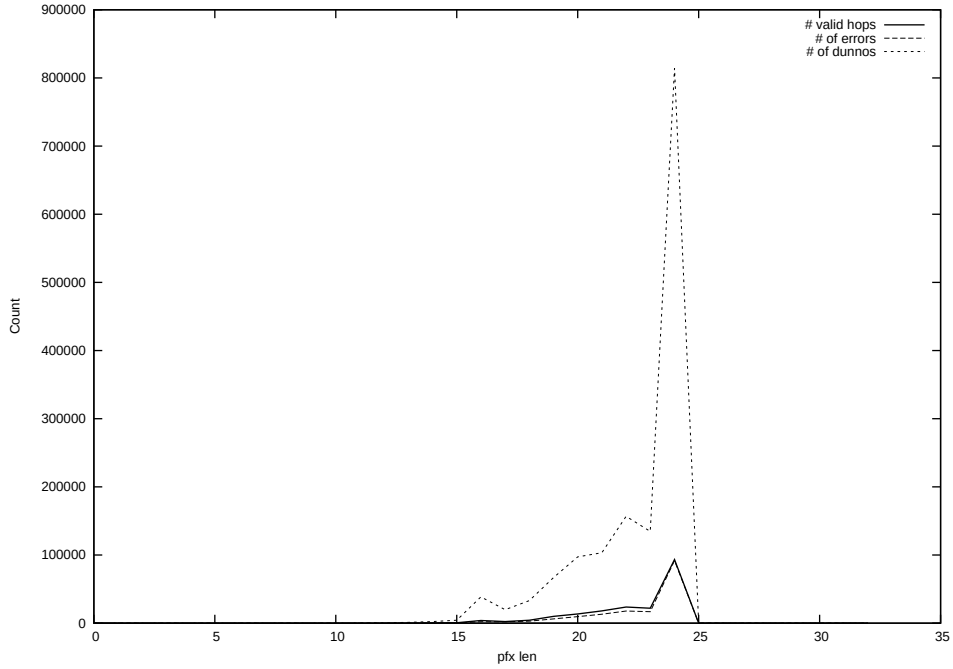
2014-05-28



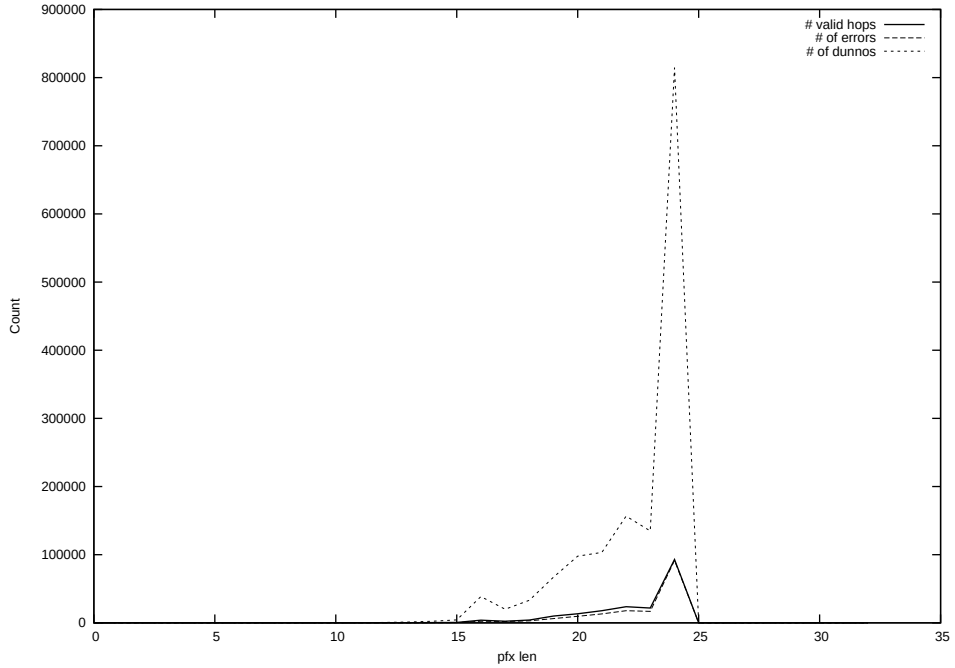
2014-05-29



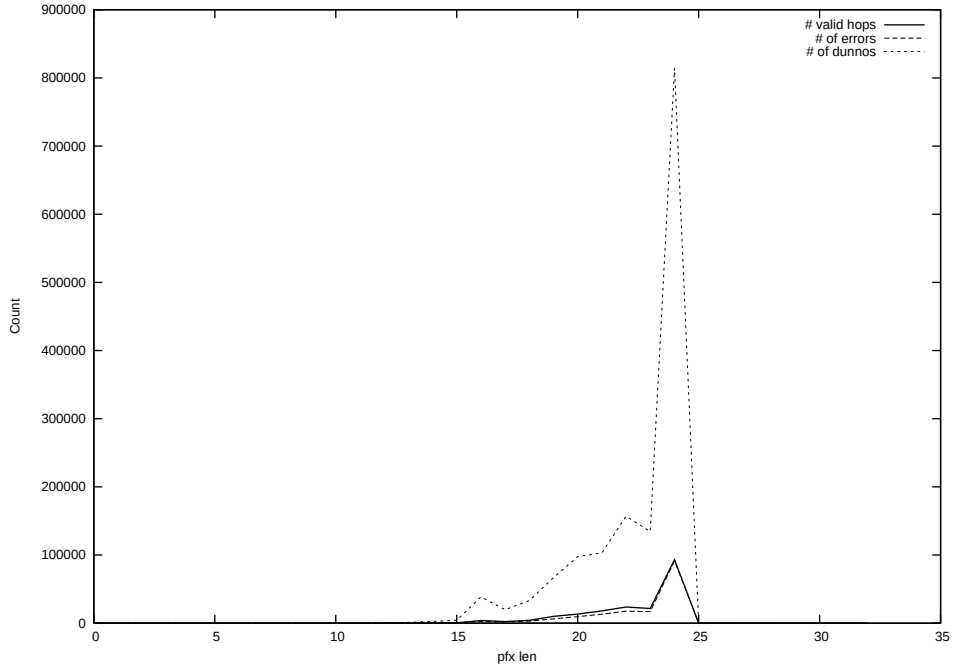
2014-05-30



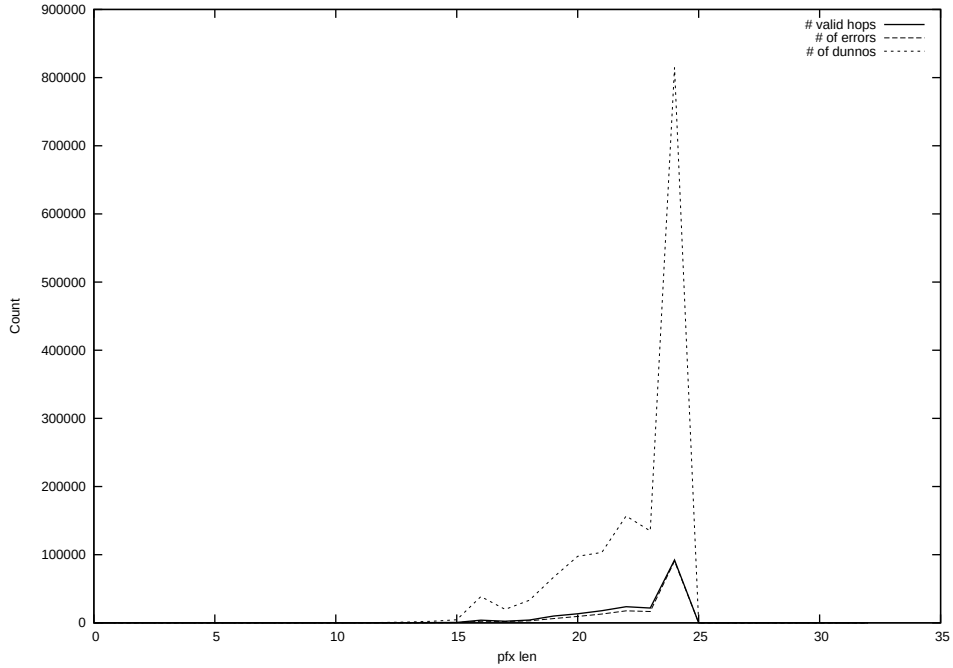
2014-05-31



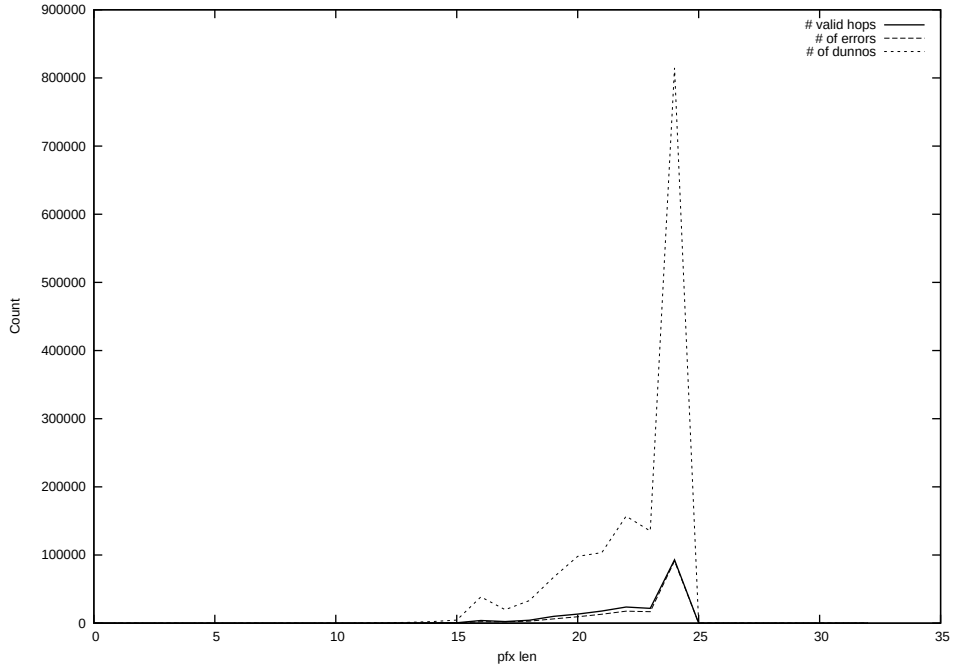
2014-06-01



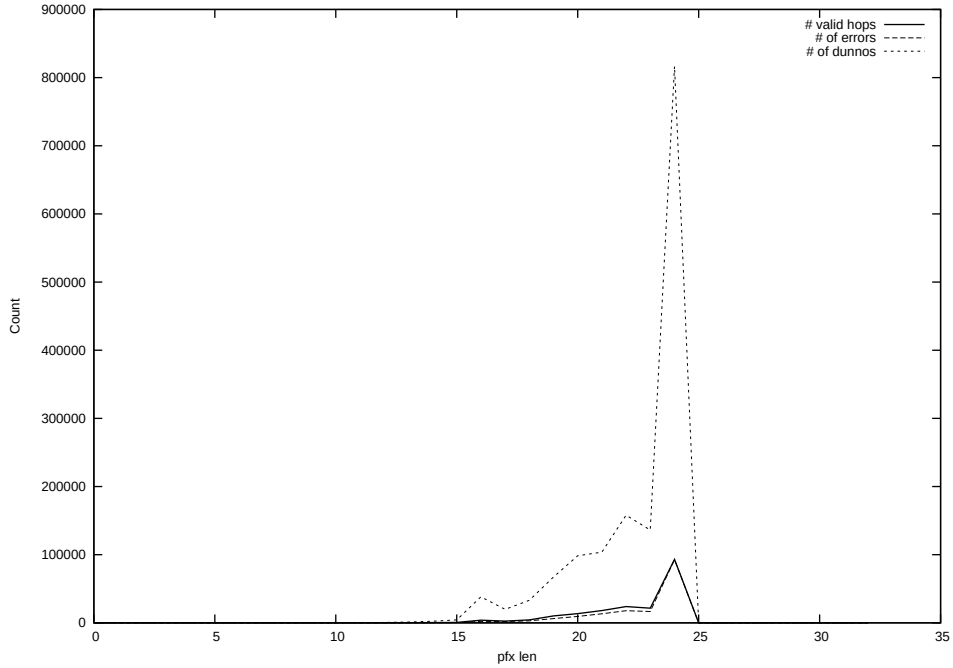
2014-06-02



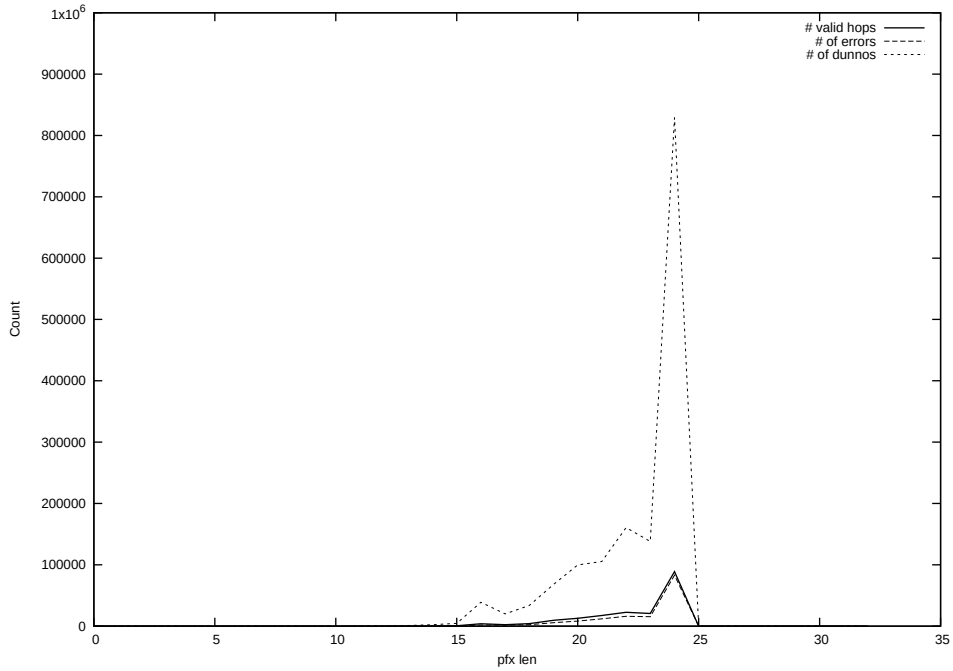
2014-06-03



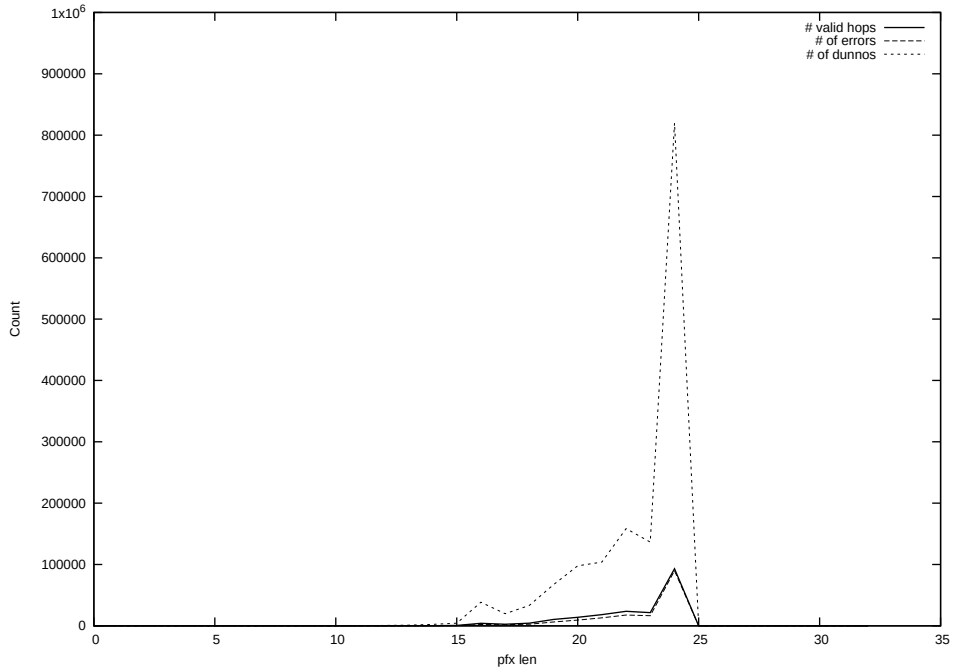
2014-06-04



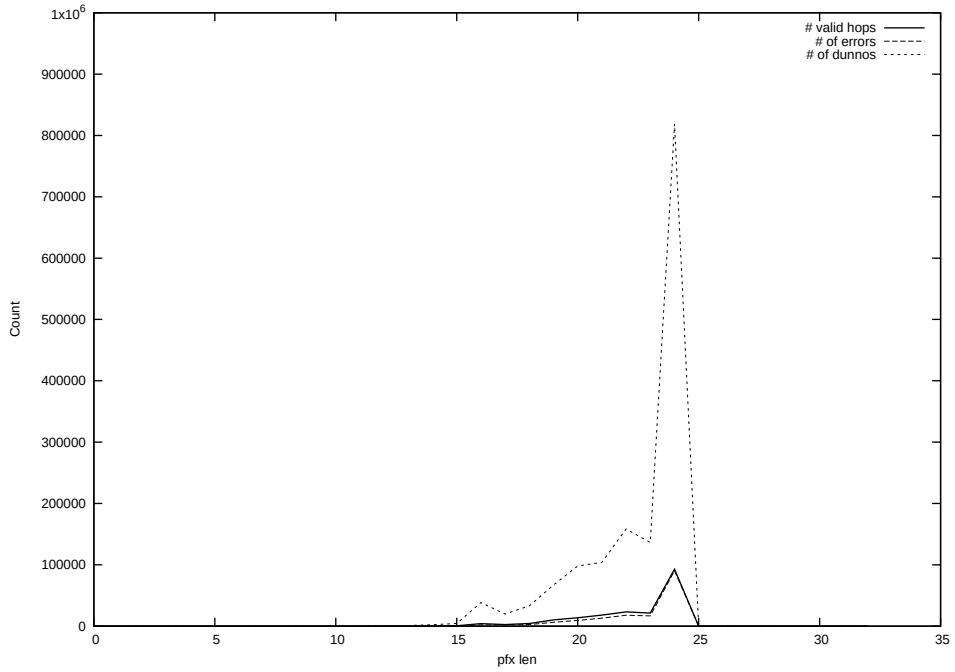
2014-06-05



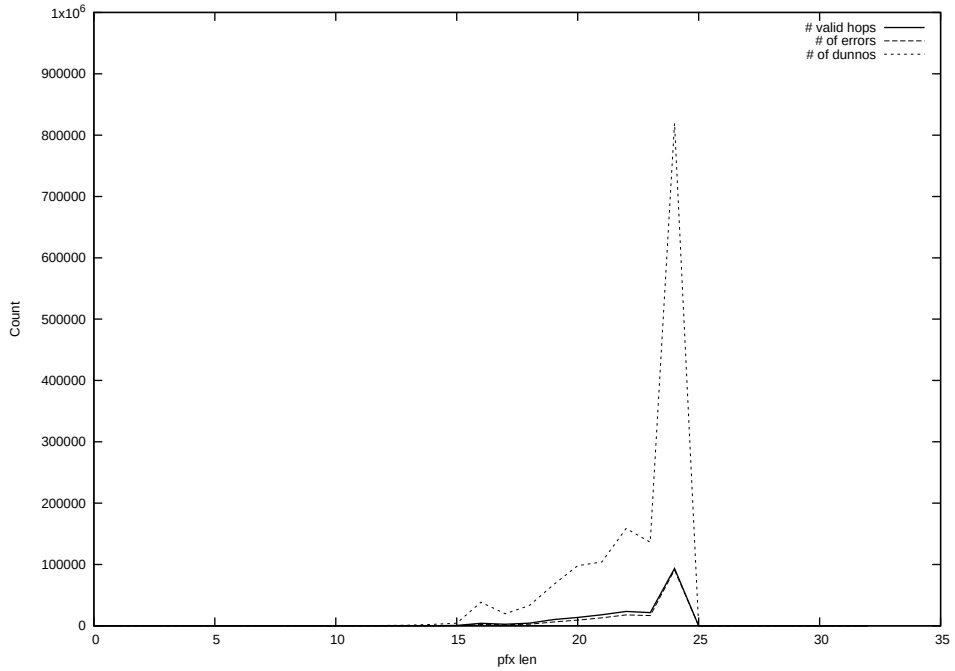
2014-06-06



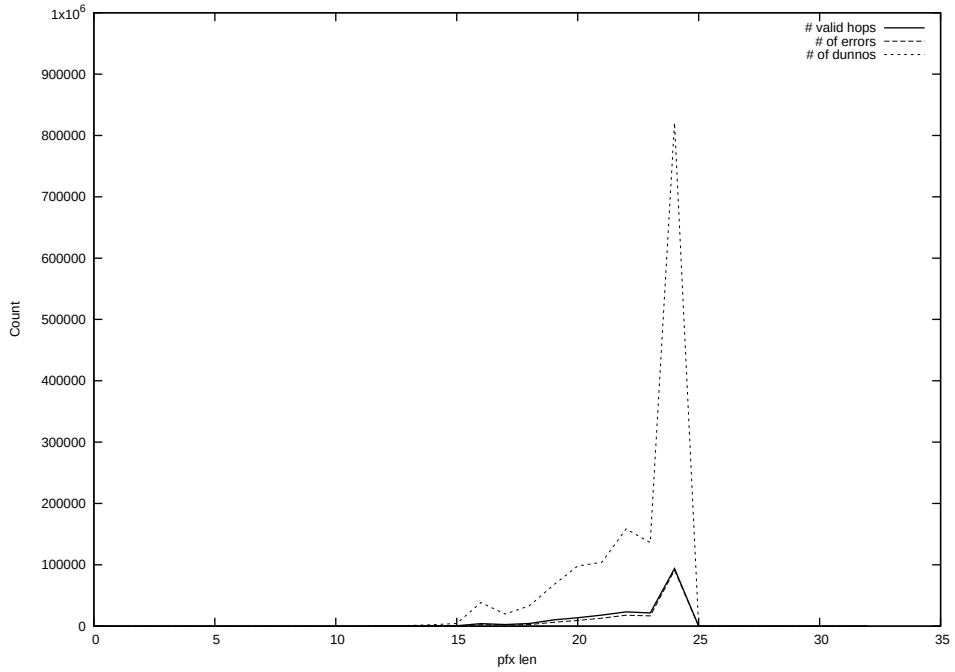
2014-06-07



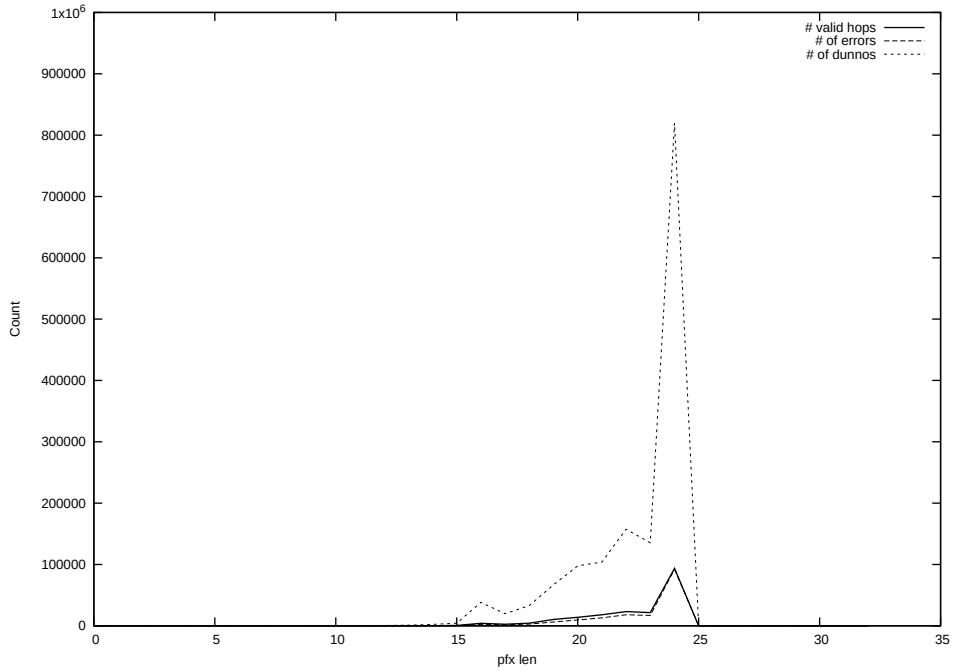
2014-06-08



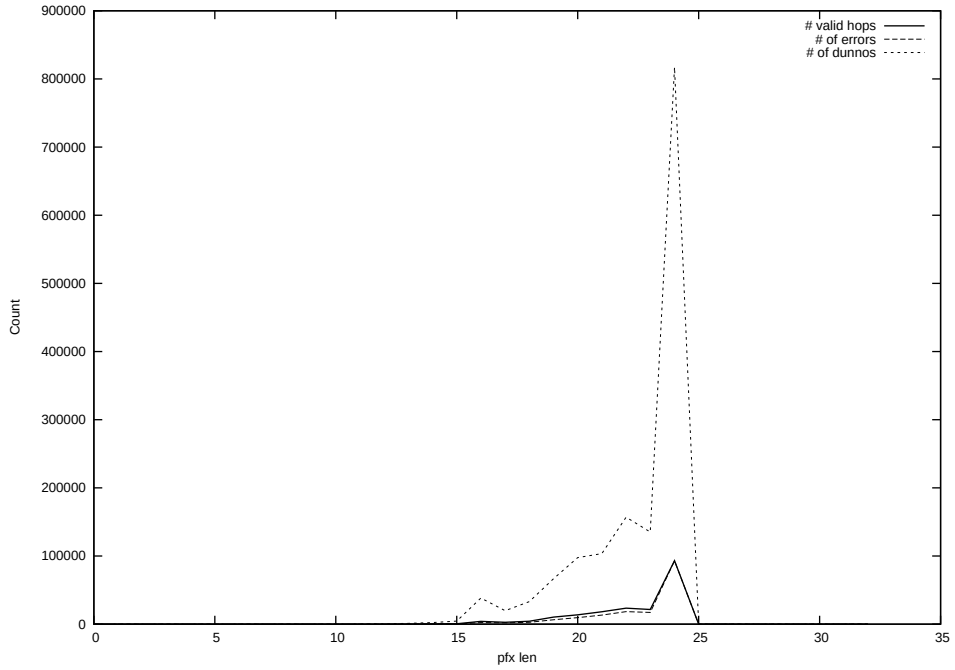
2014-06-09



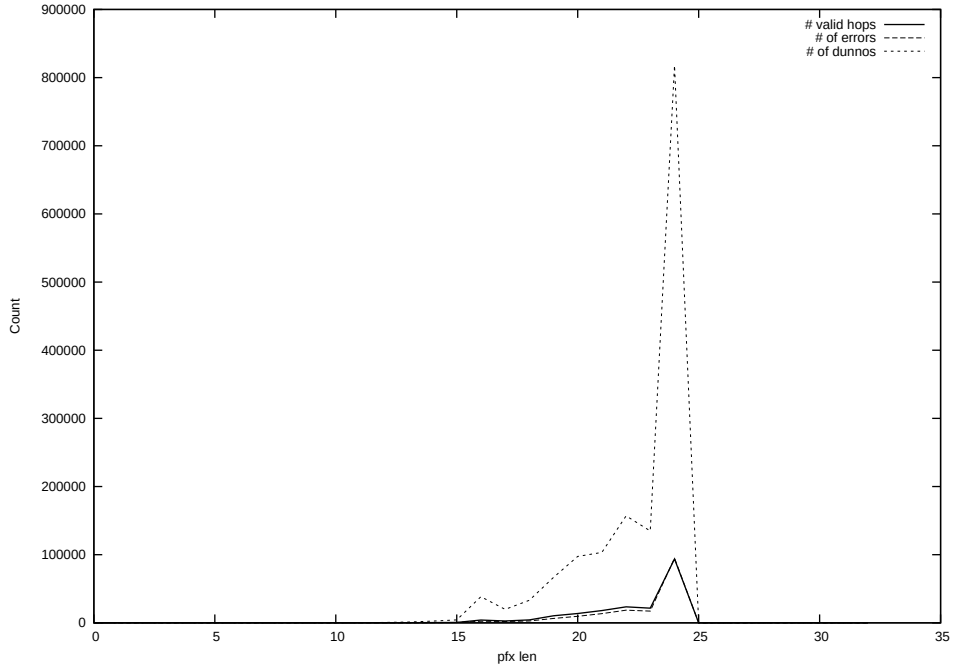
2014-06-10



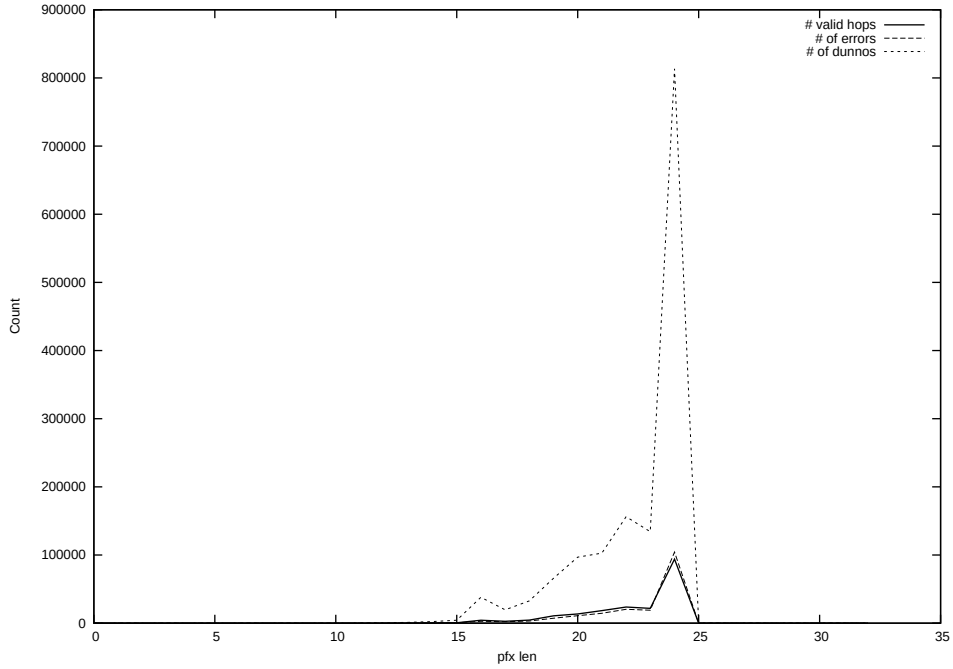
2014-06-11



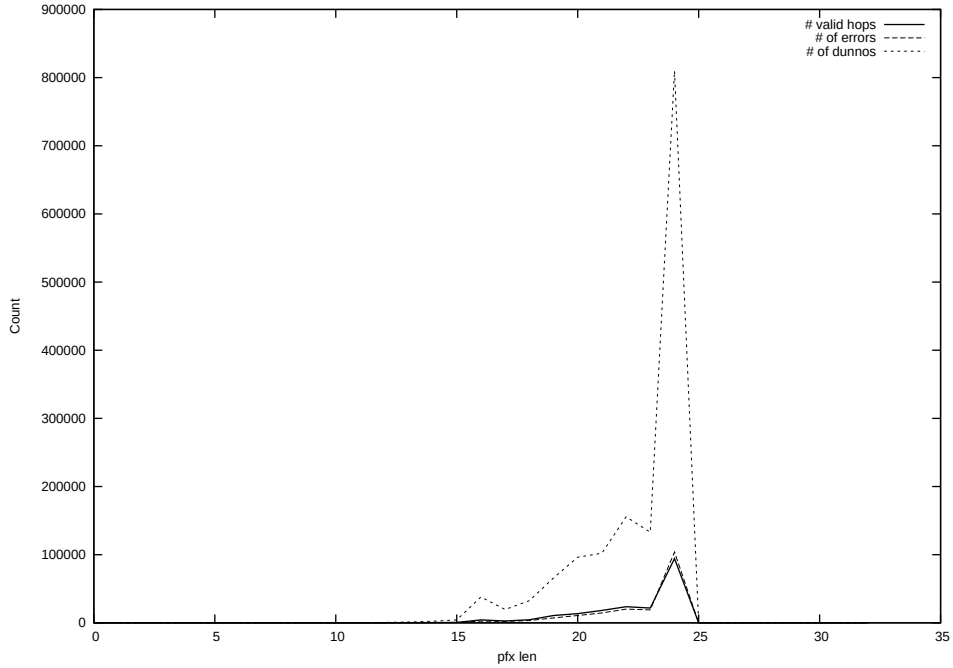
2014-06-12



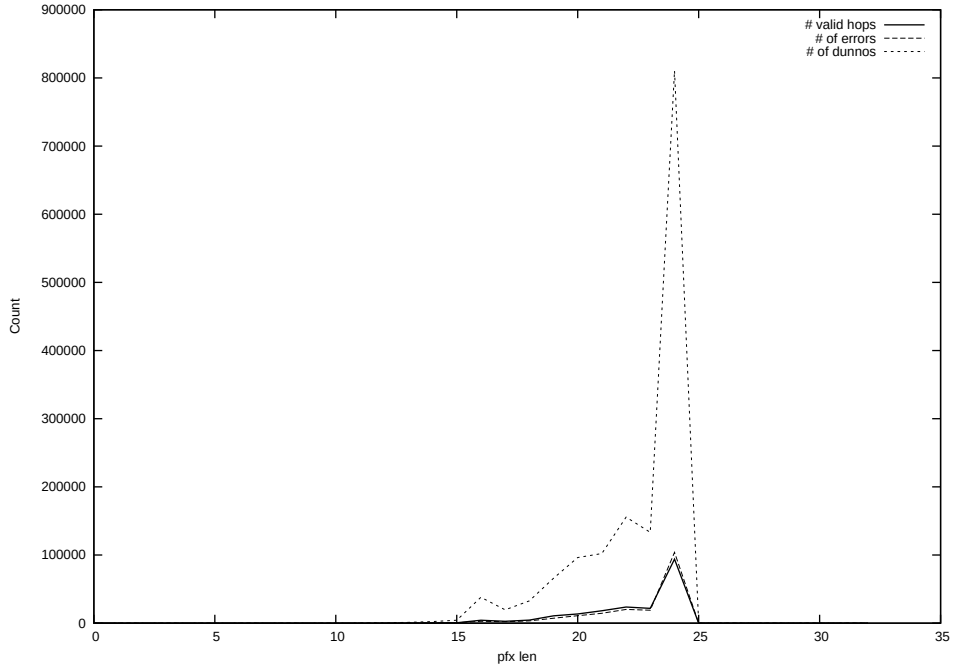
2014-06-13



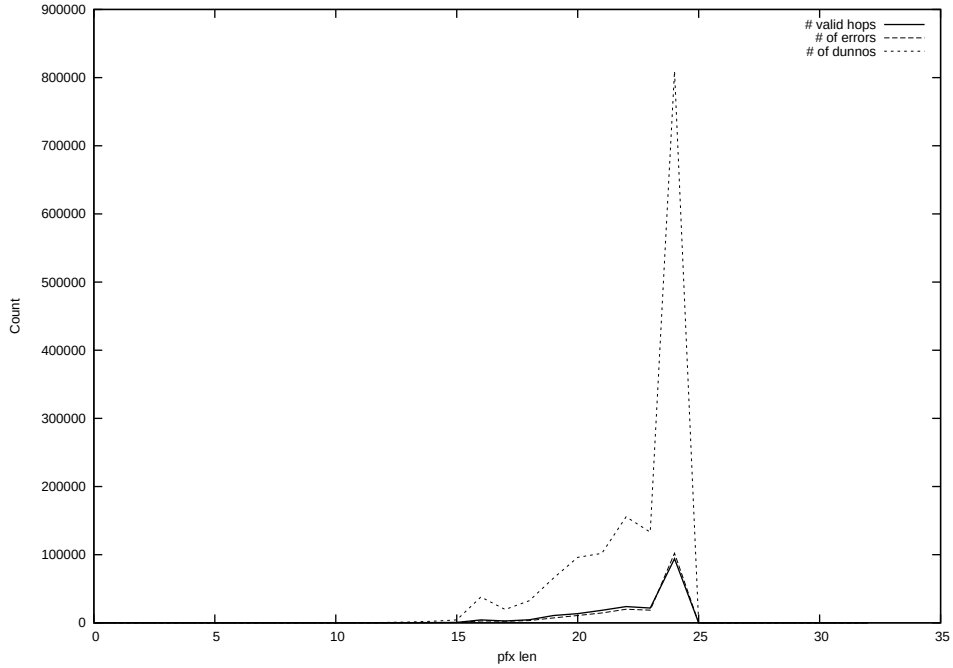
2014-06-14



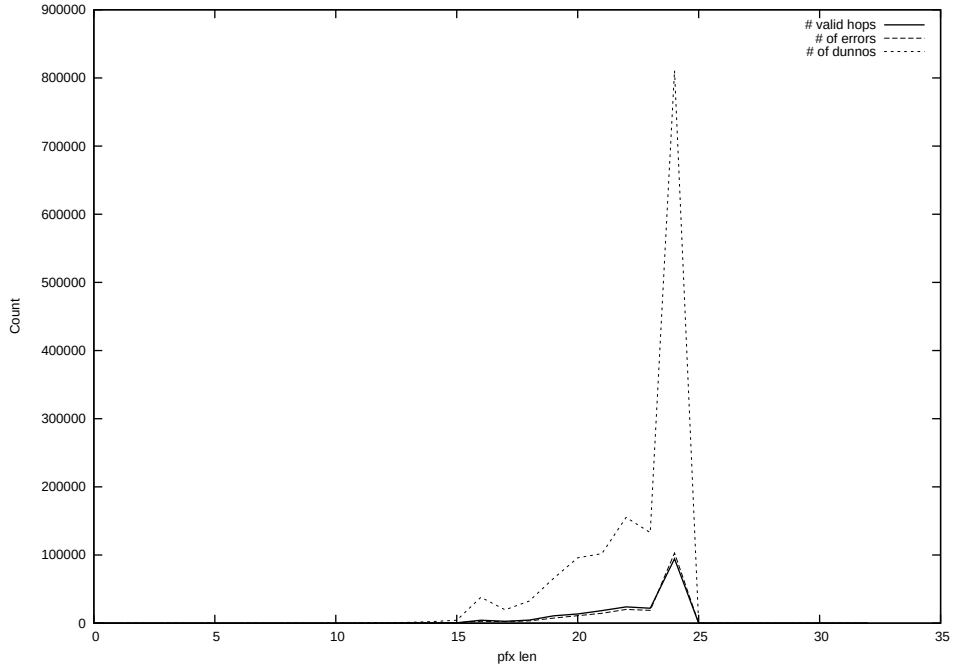
2014-06-15



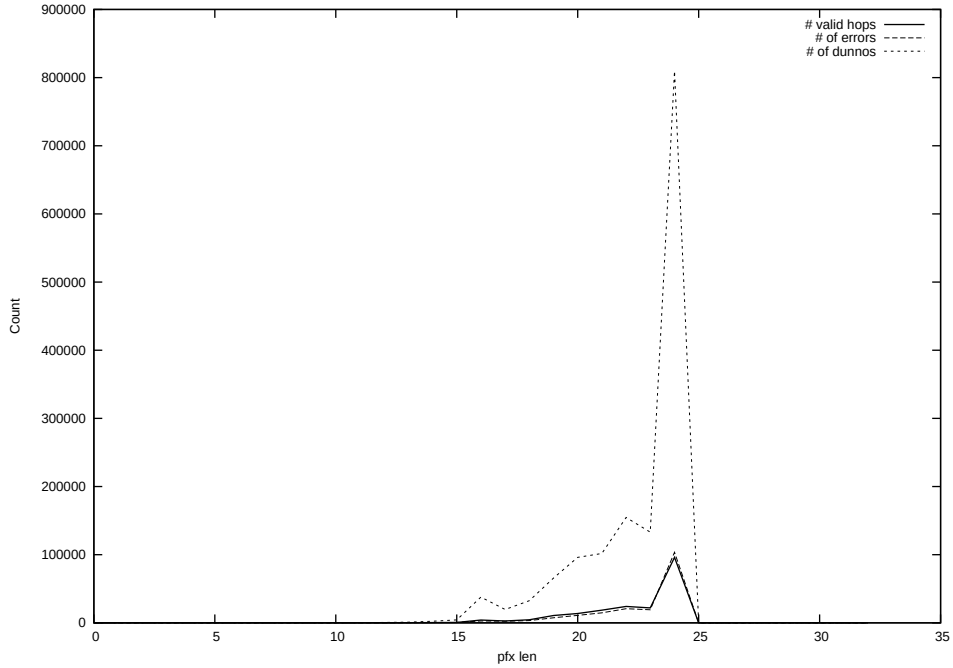
2014-06-16



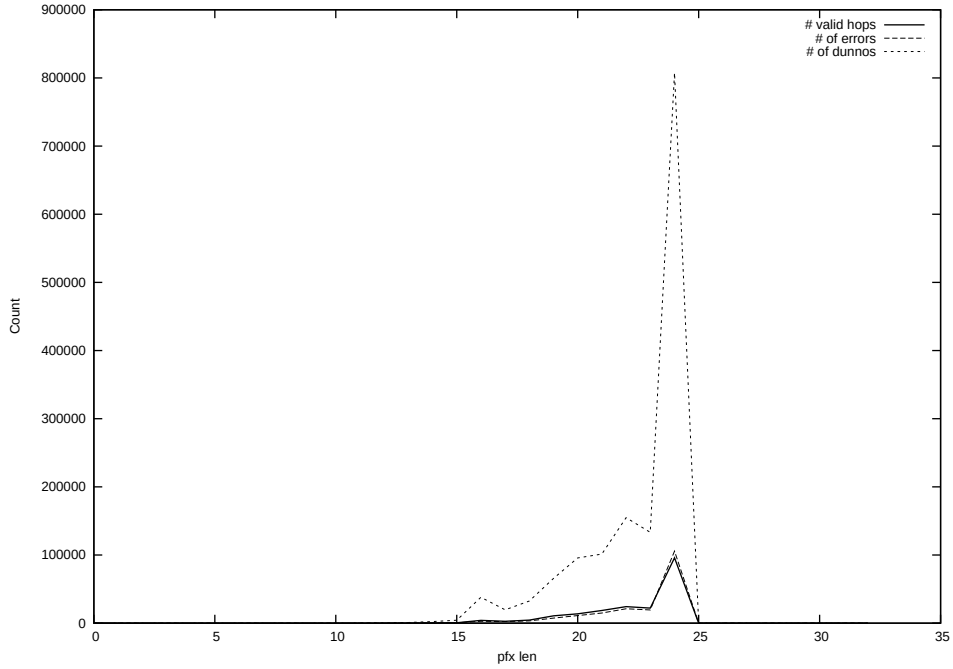
2014-06-17



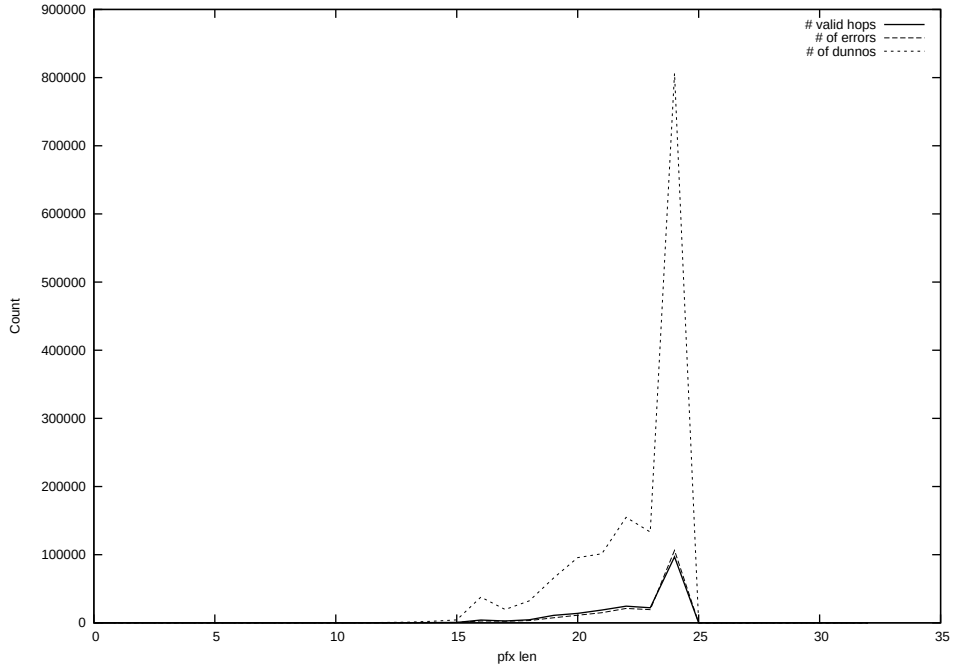
2014-06-18



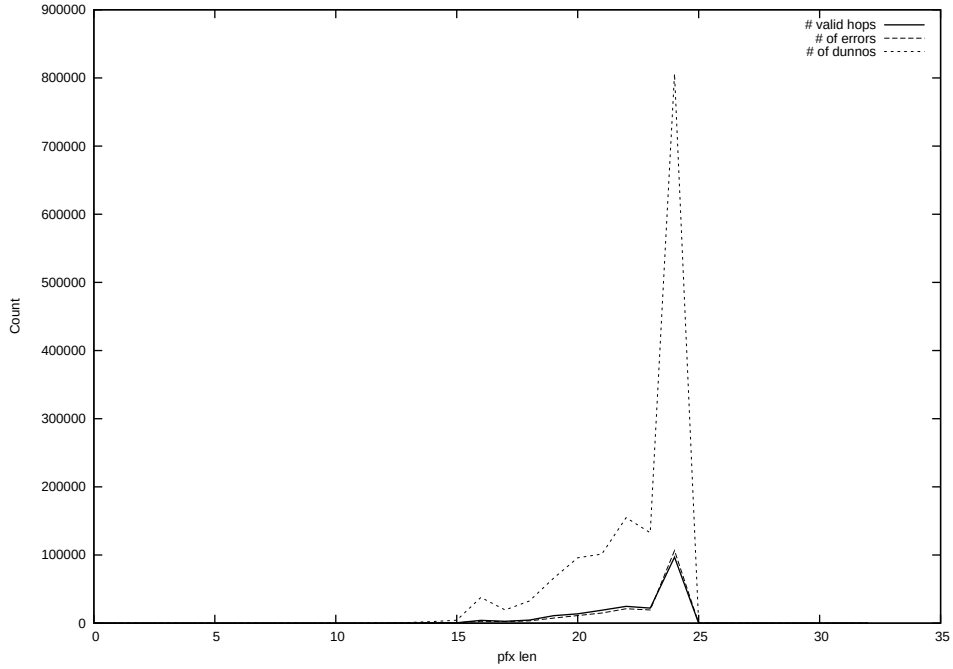
2014-06-19



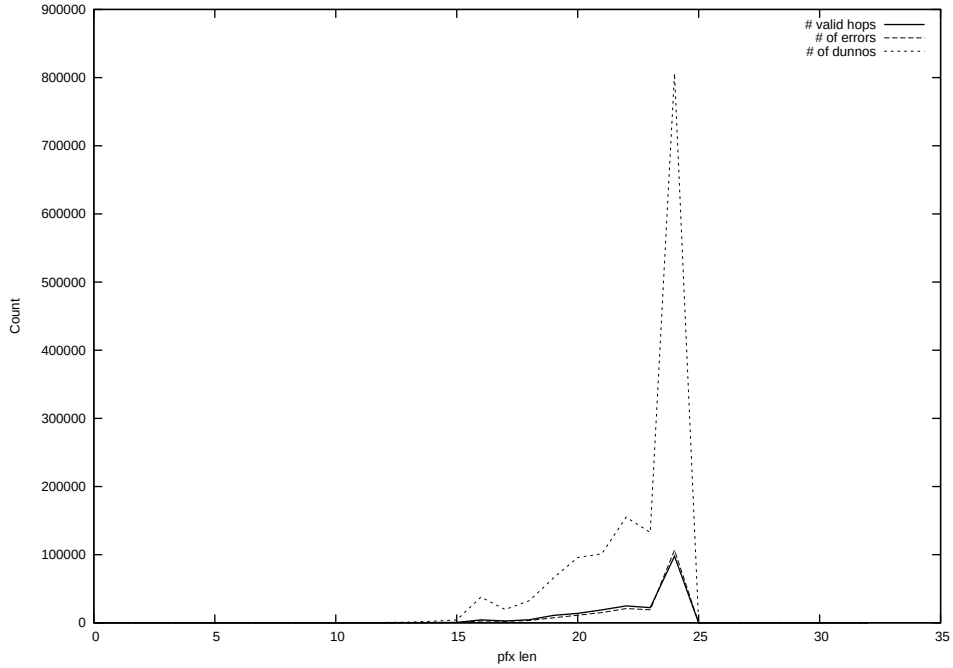
2014-06-20



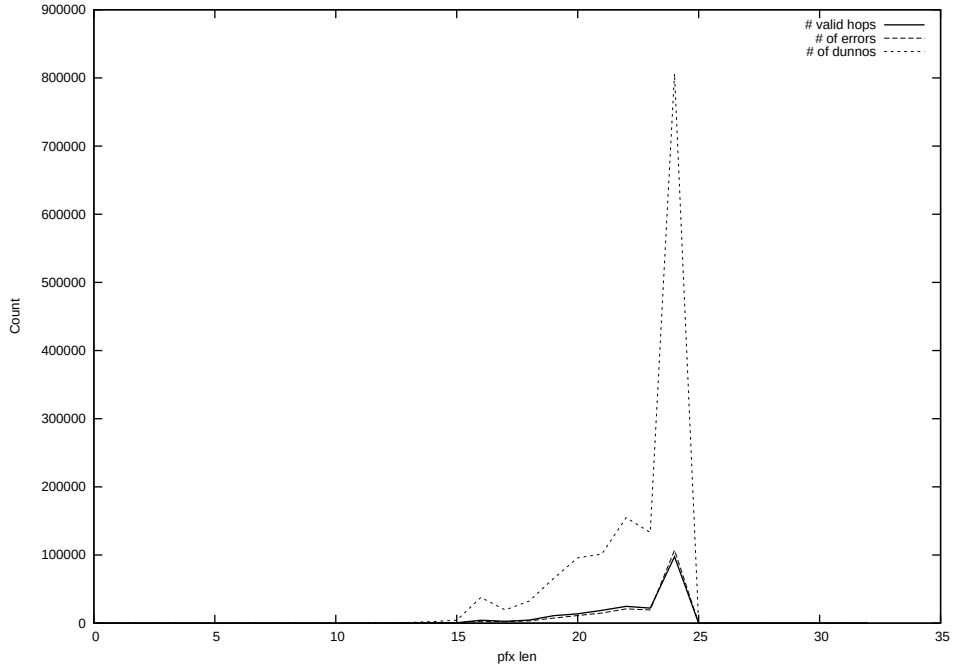
2014-06-21



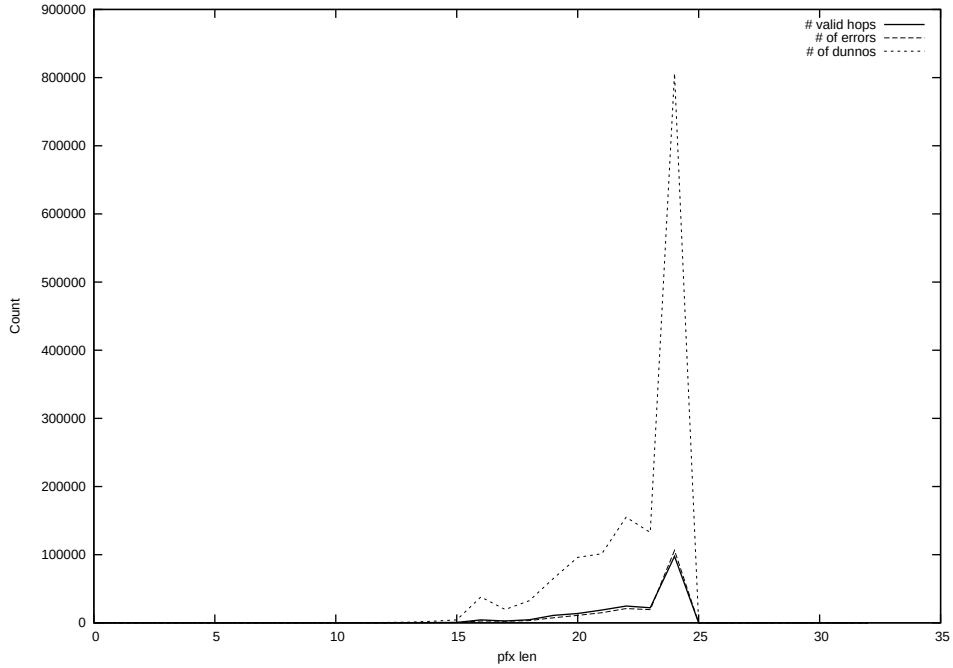
2014-06-22



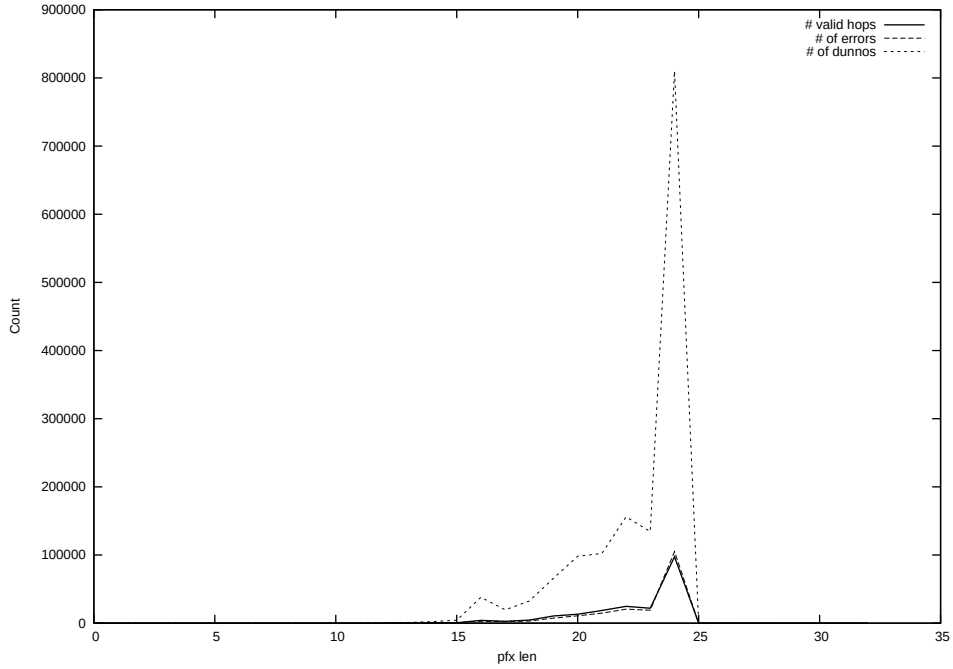
2014-06-23



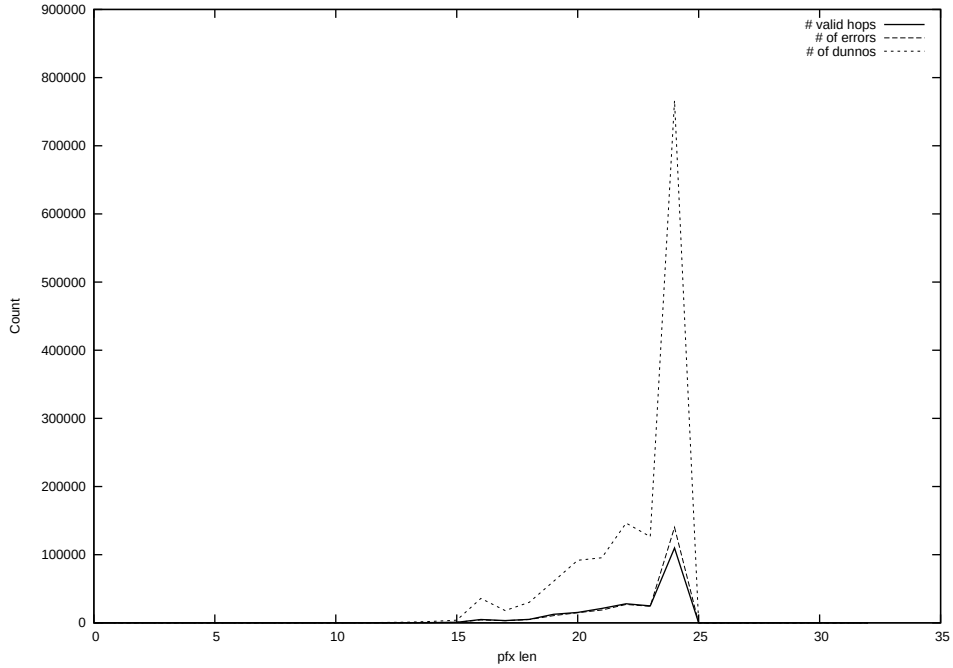
2014-06-24



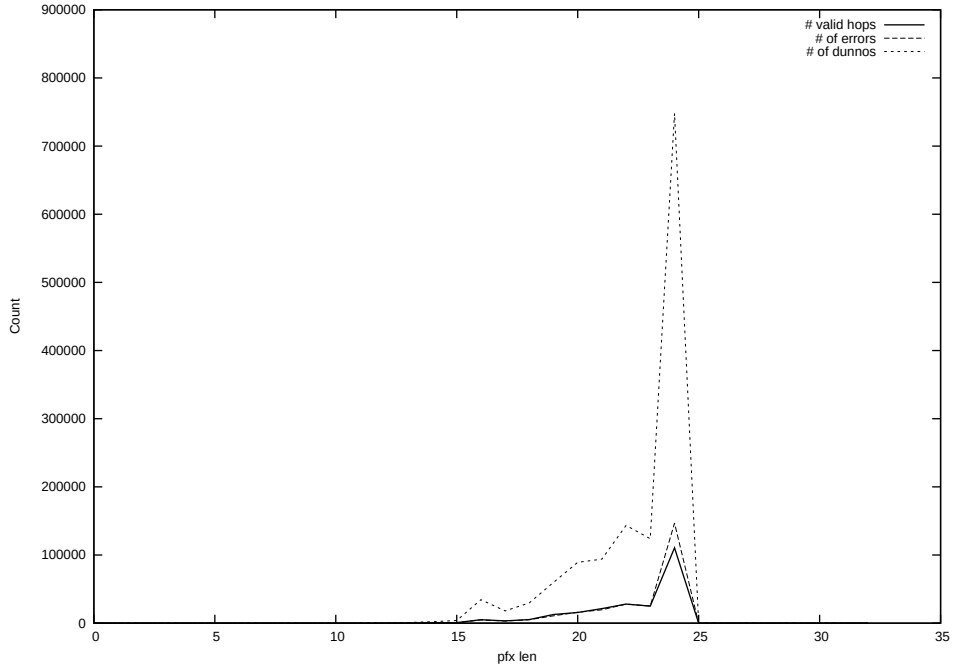
2014-06-25



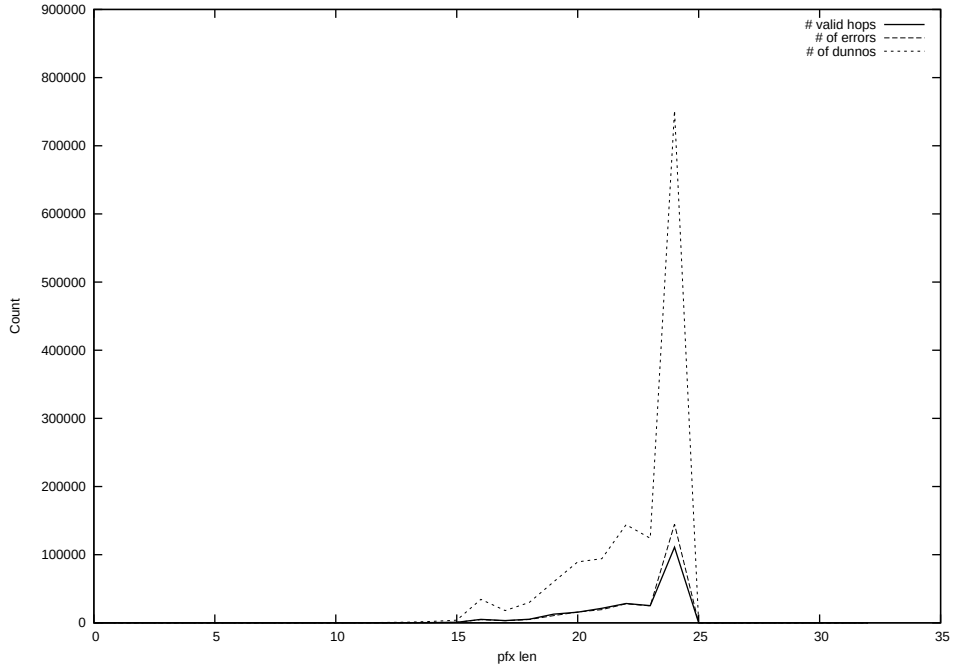
2014-06-26



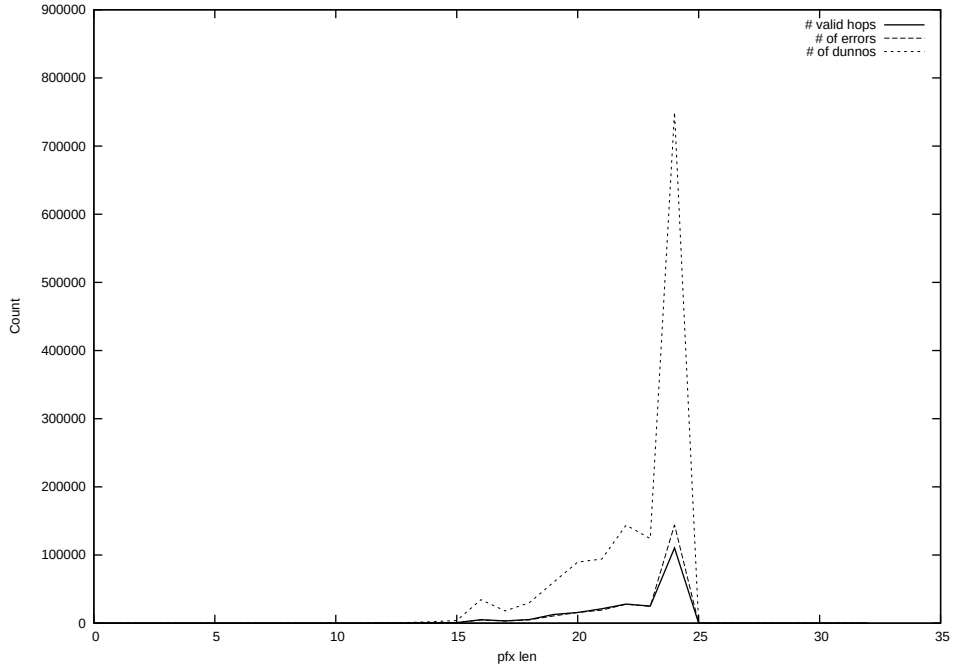
2014-06-27



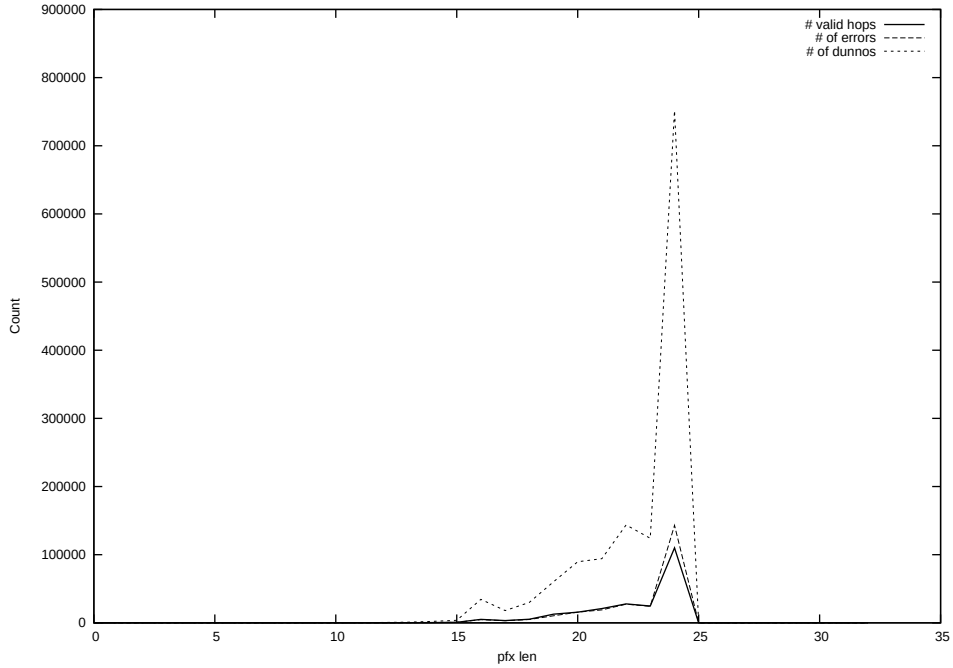
2014-06-28



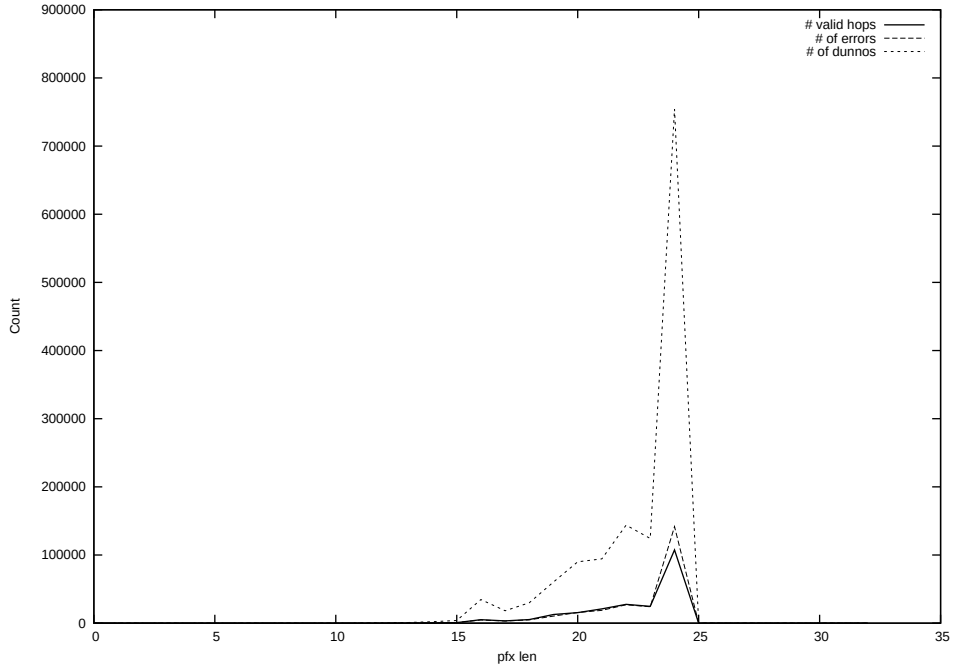
2014-06-29



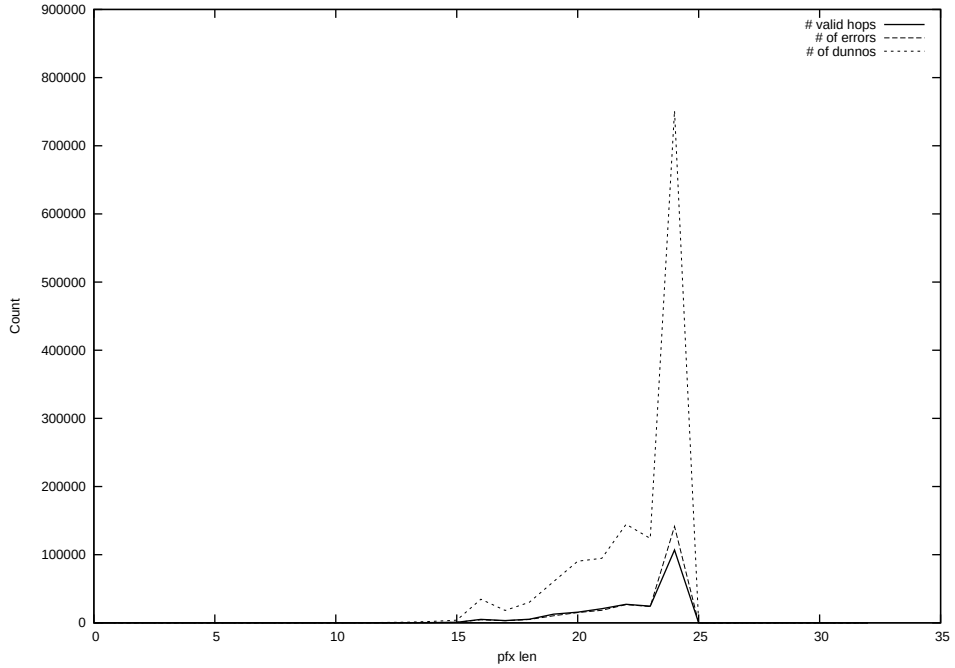
2014-06-30



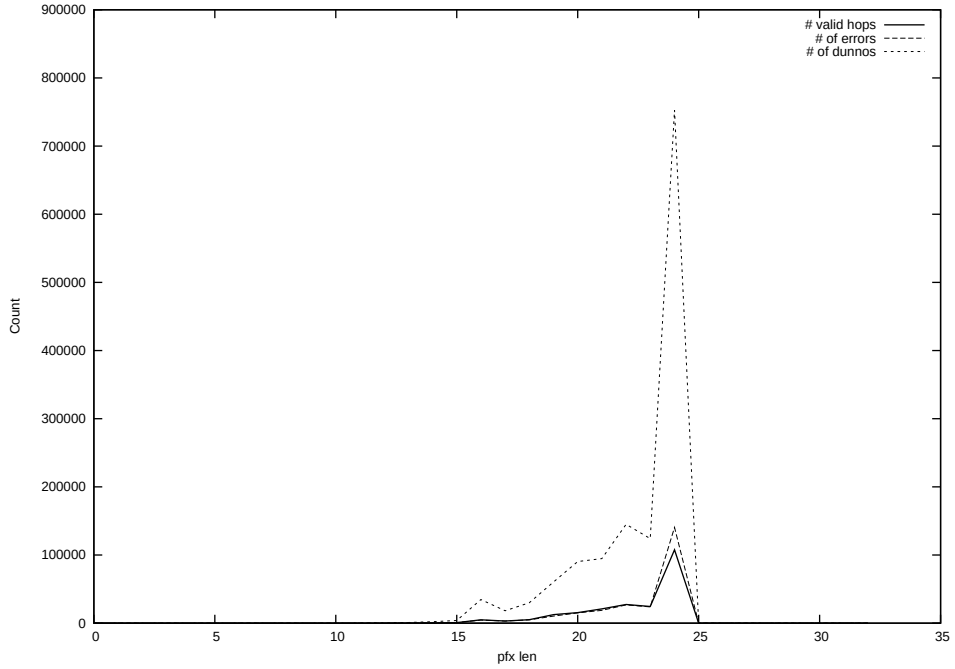
2014-07-01



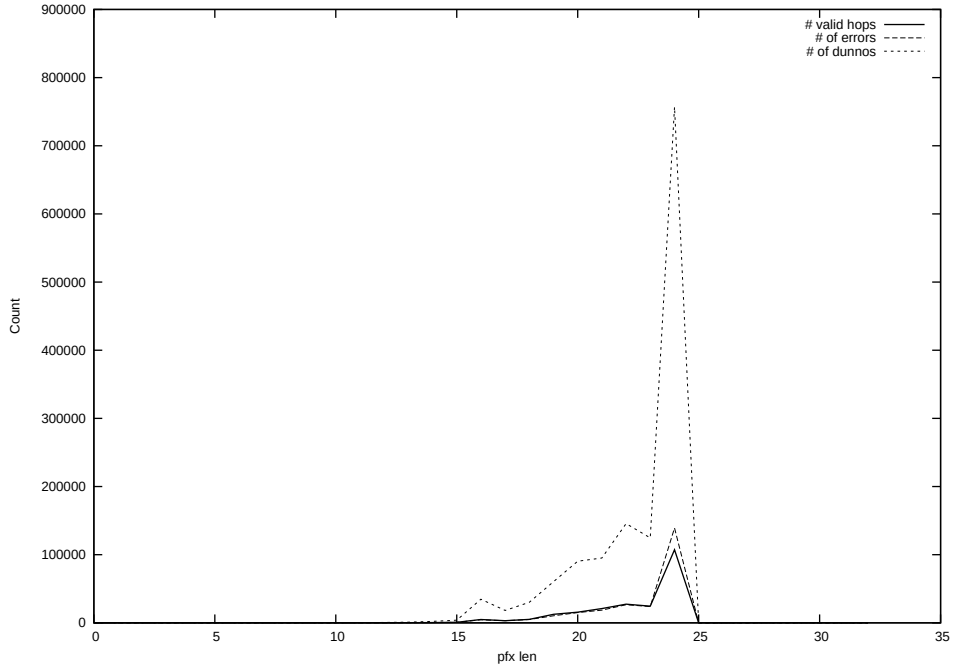
2014-07-02



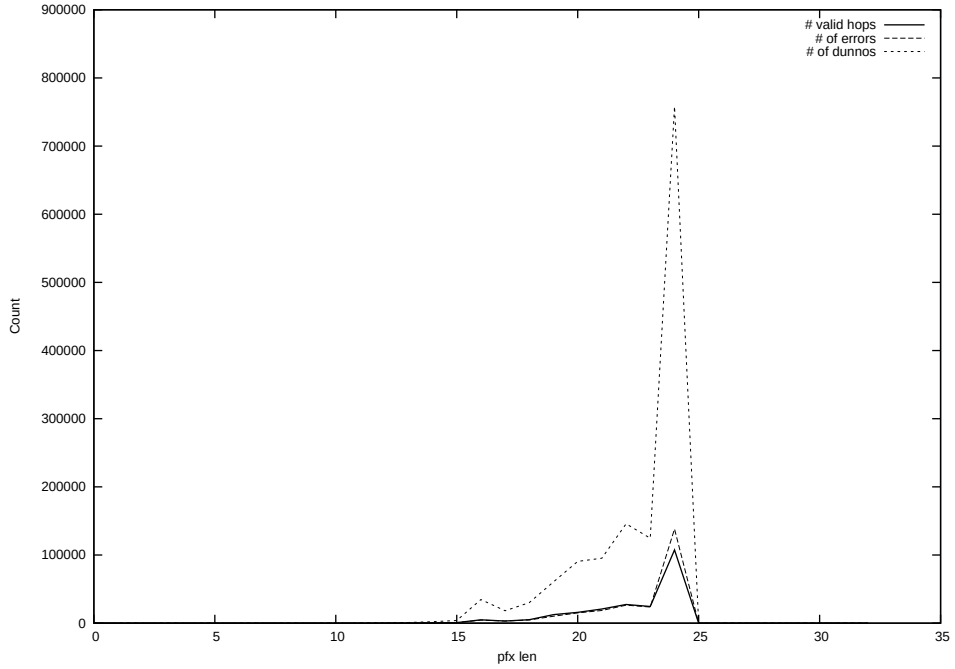
2014-07-03



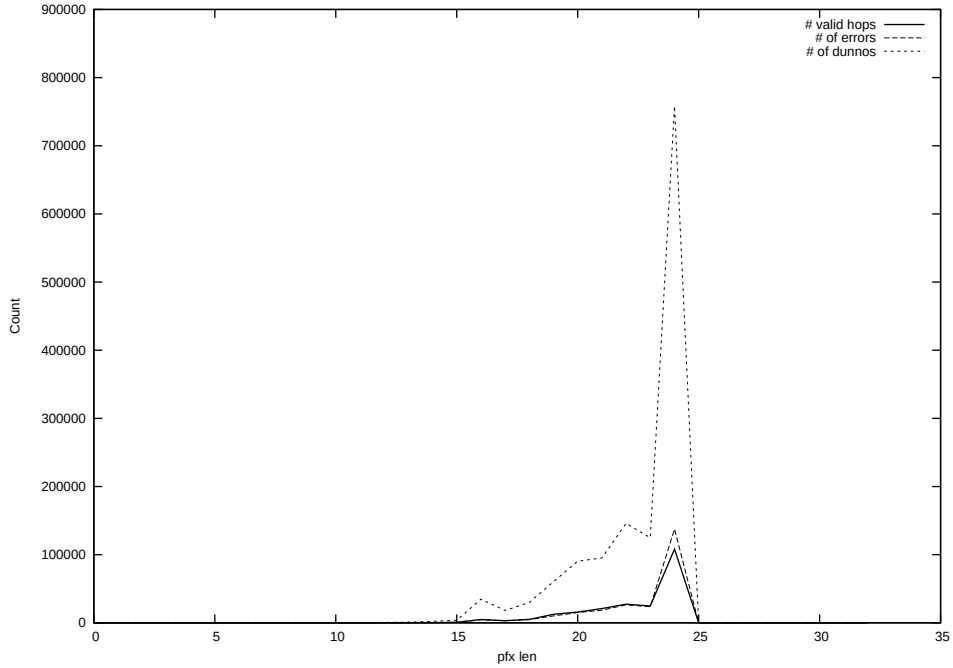
2014-07-04



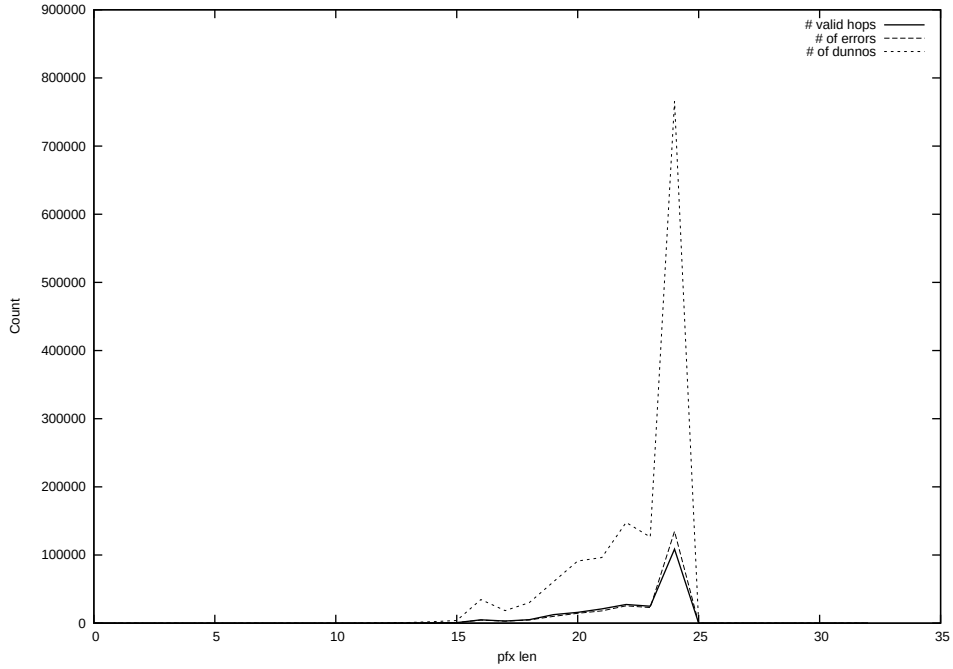
2014-07-05



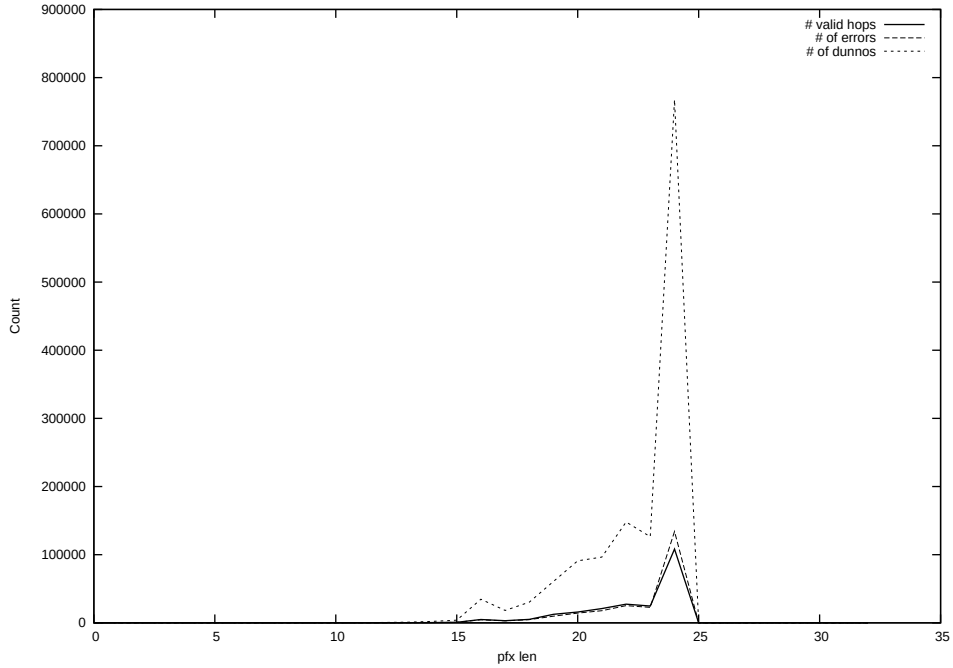
2014-07-06



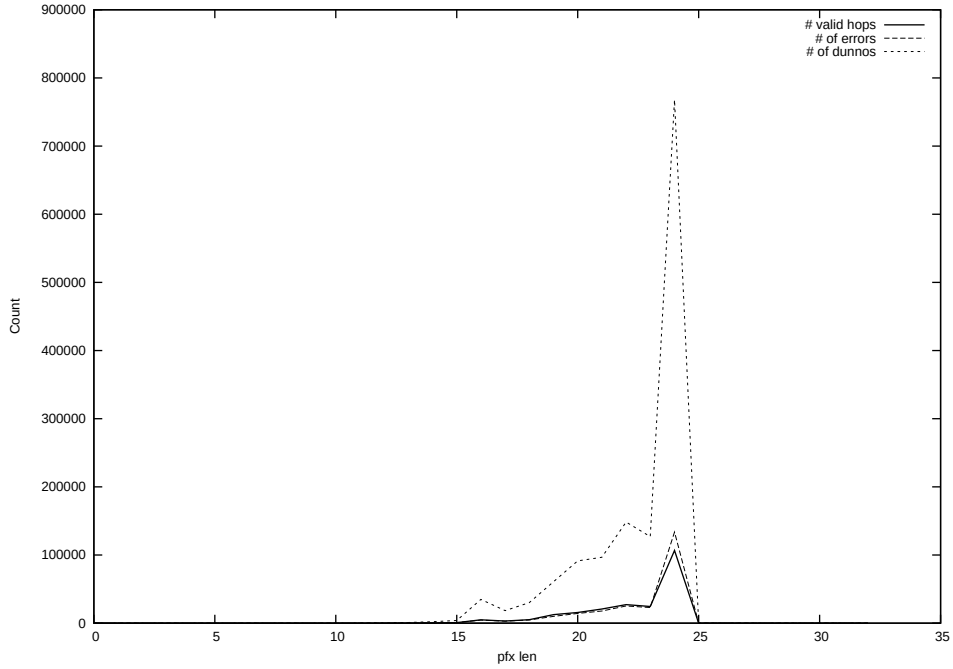
2014-07-07



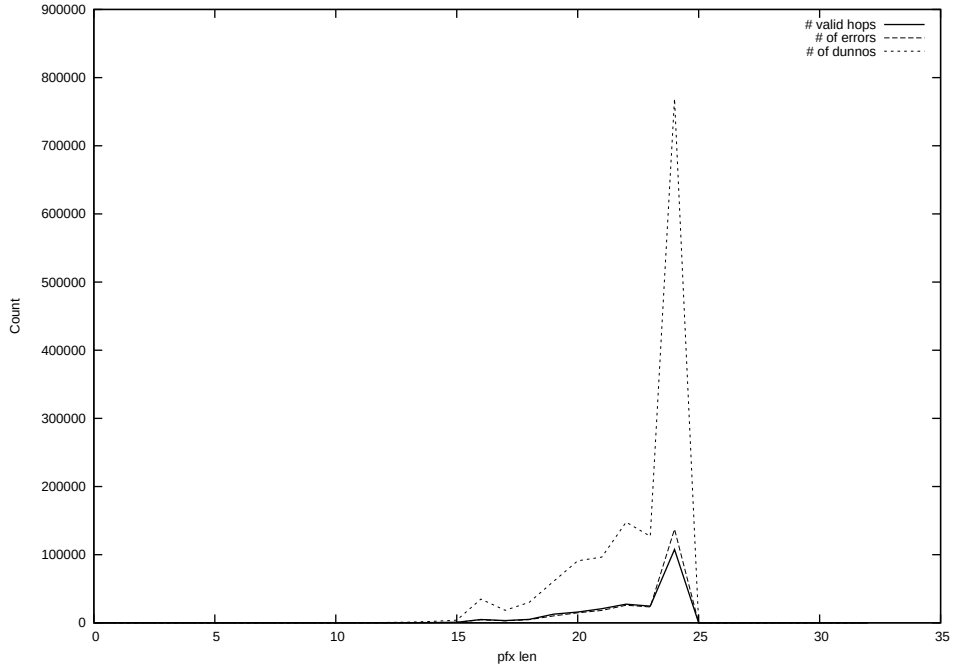
2014-07-08



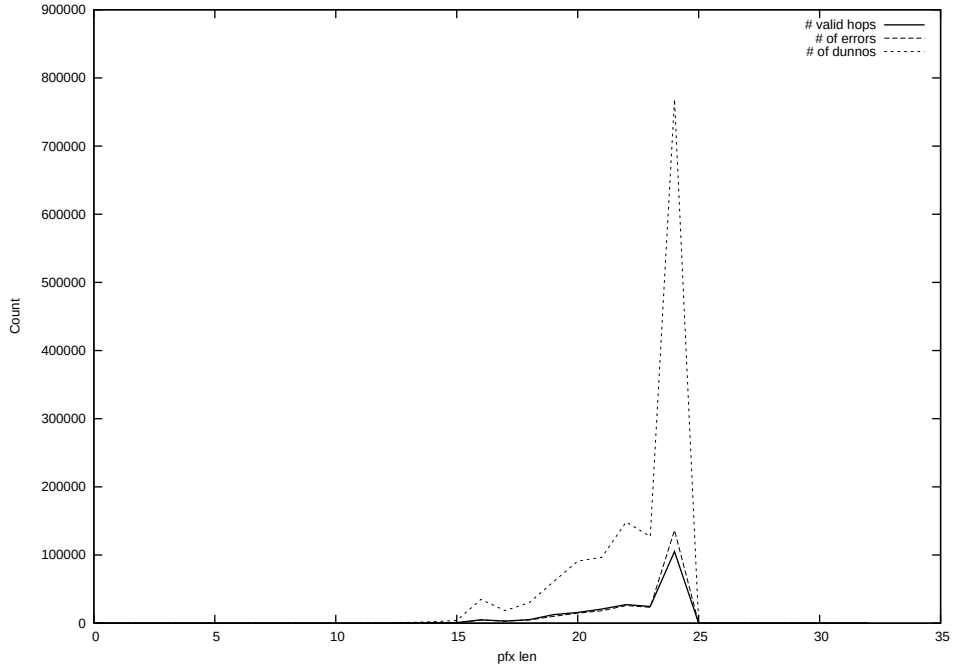
2014-07-09



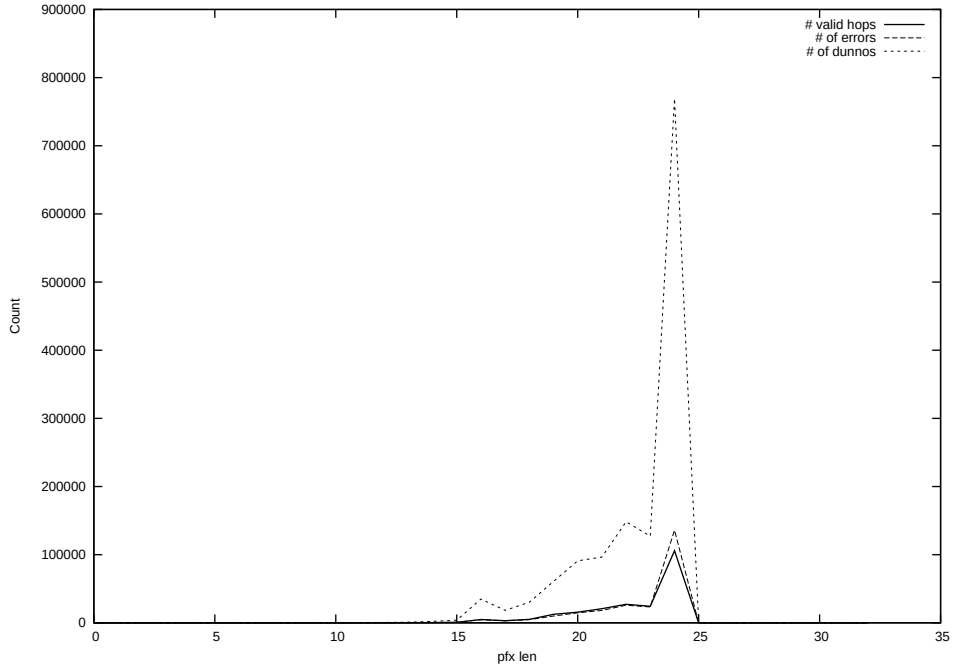
2014-07-10



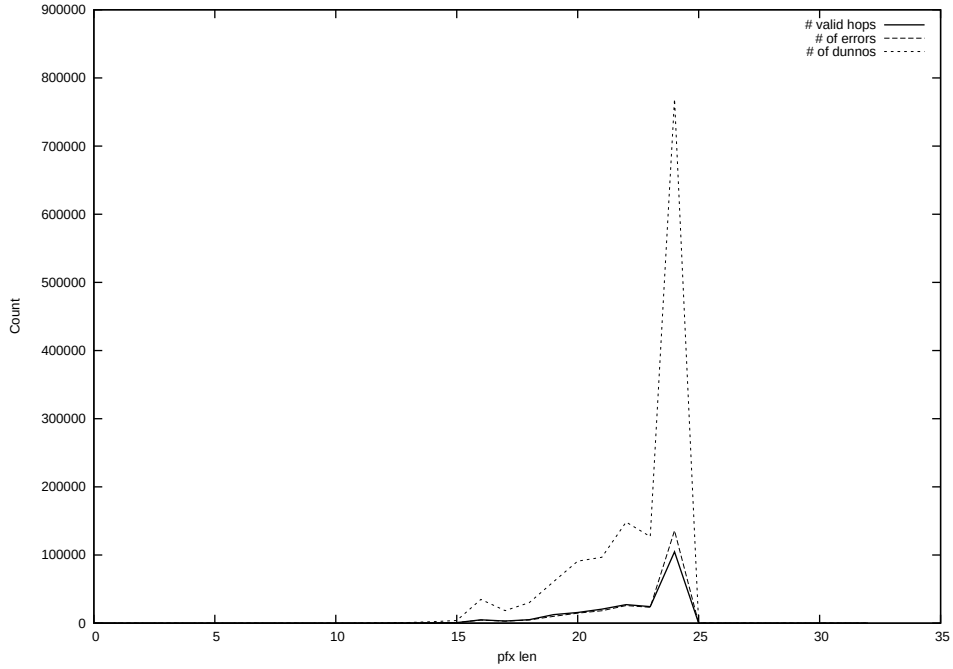
2014-07-11



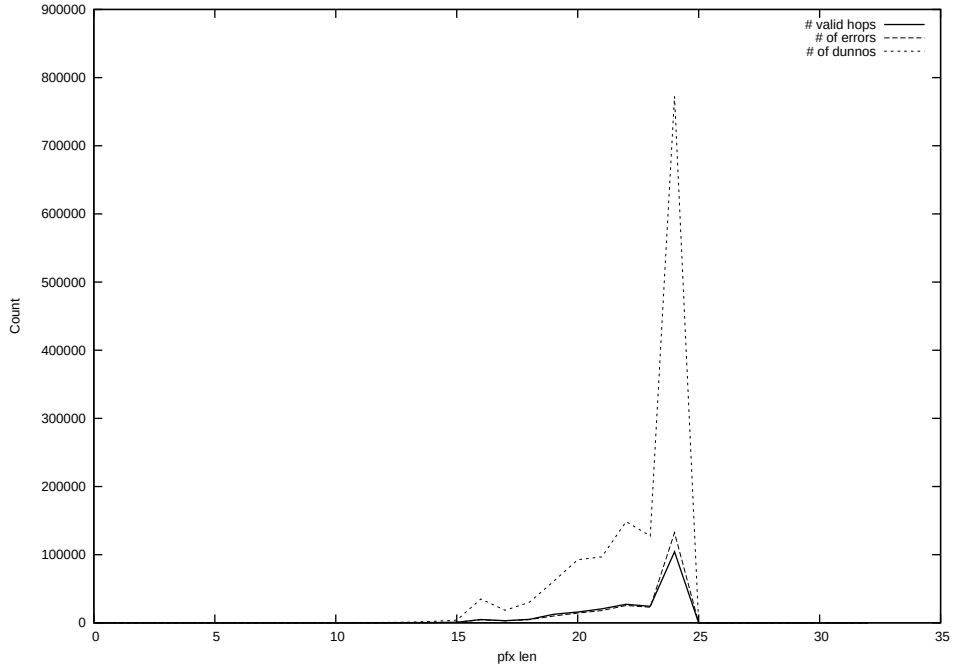
2014-07-12



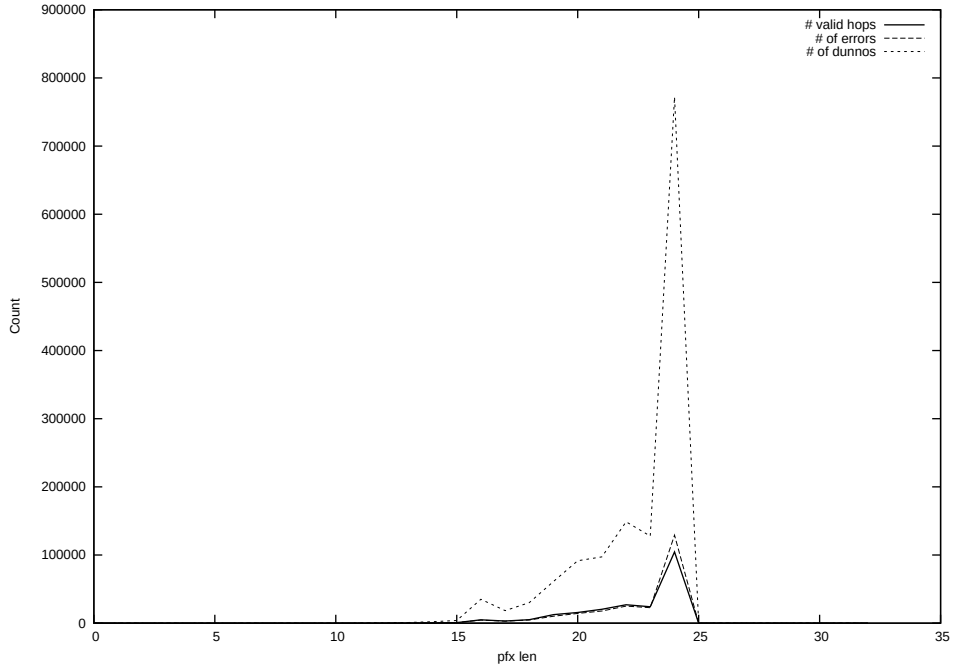
2014-07-13



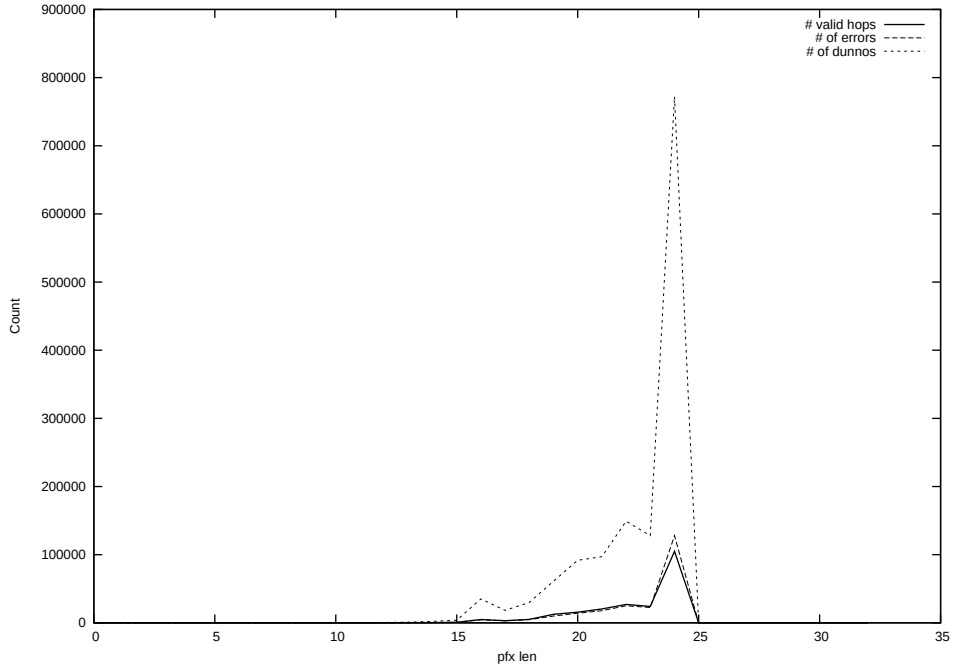
2014-07-14



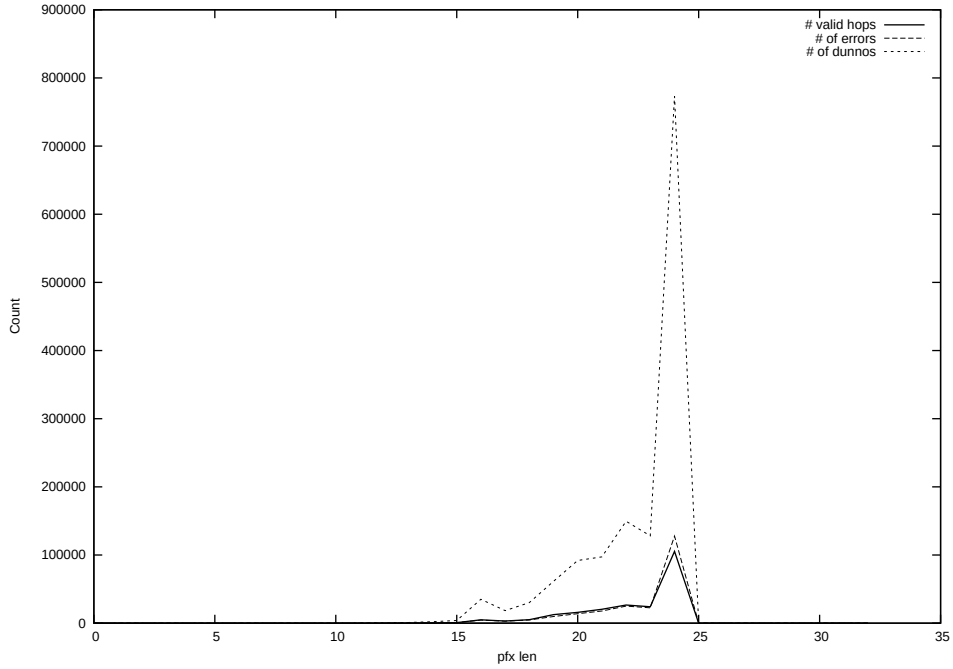
2014-07-15



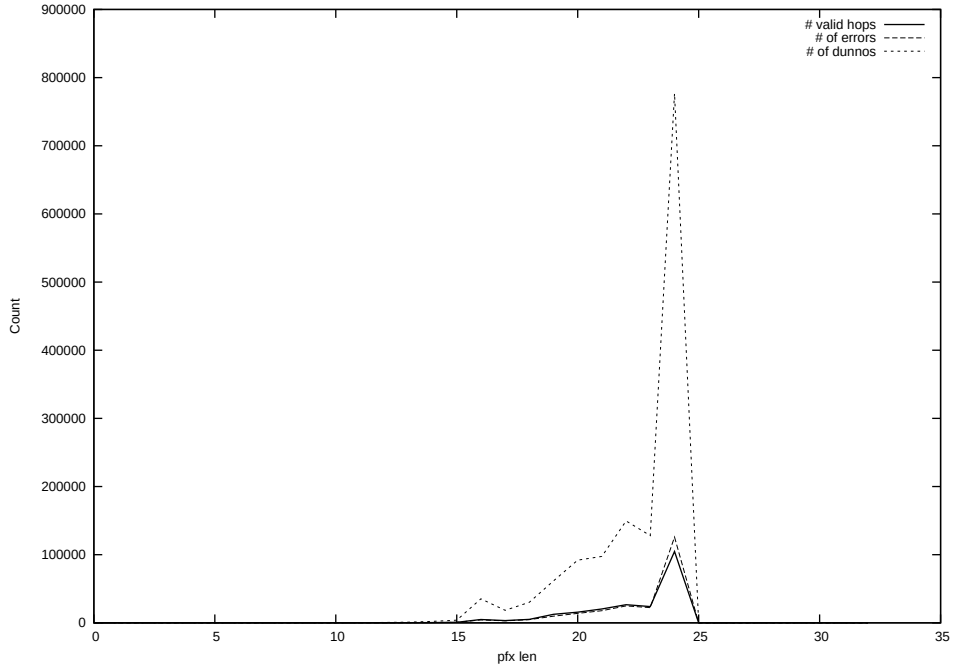
2014-07-16



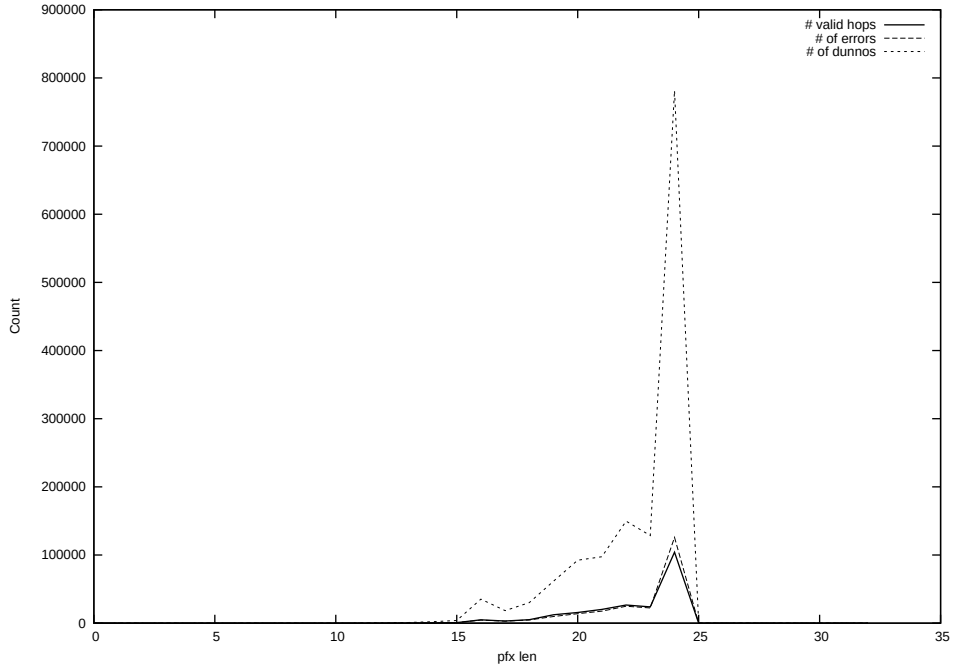
2014-07-17



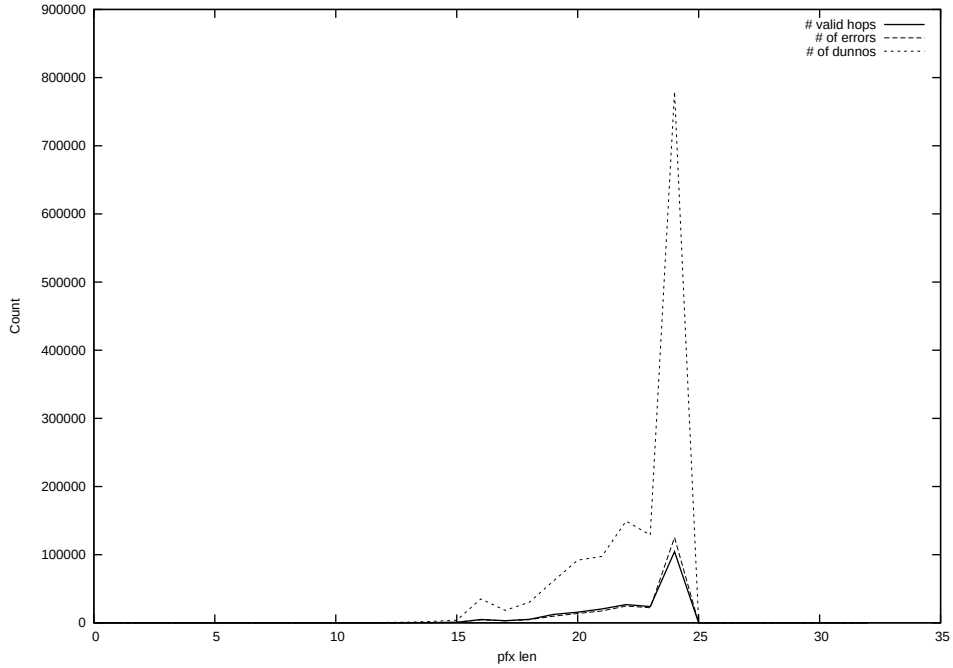
2014-07-18



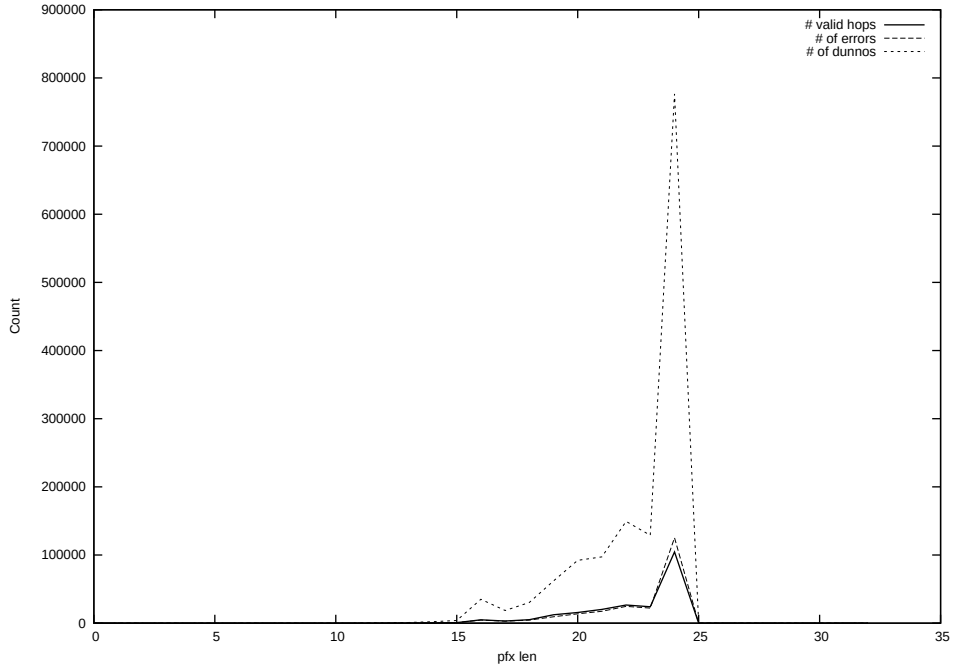
2014-07-19



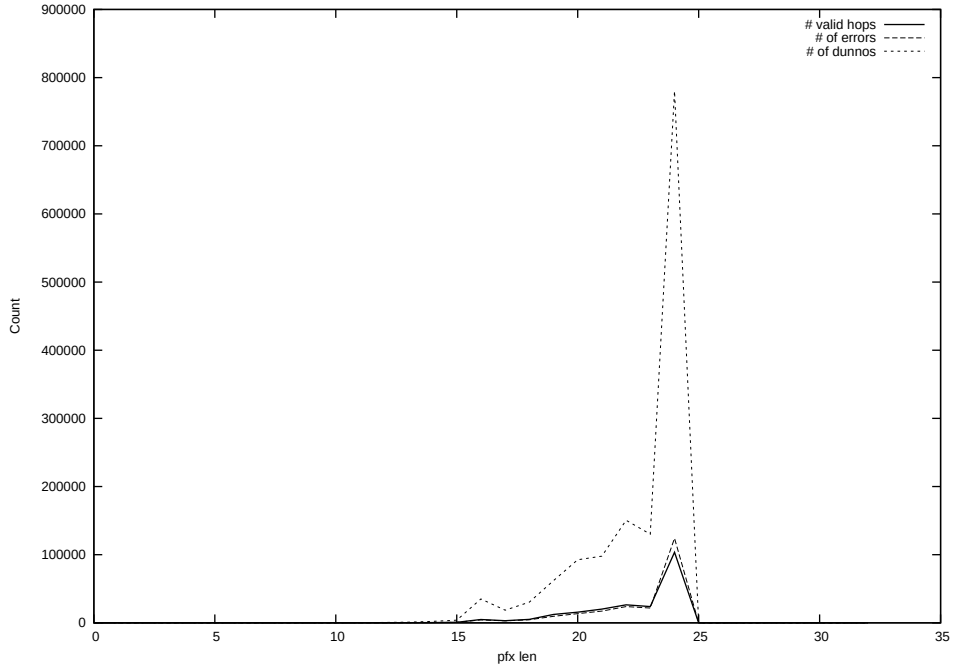
2014-07-20



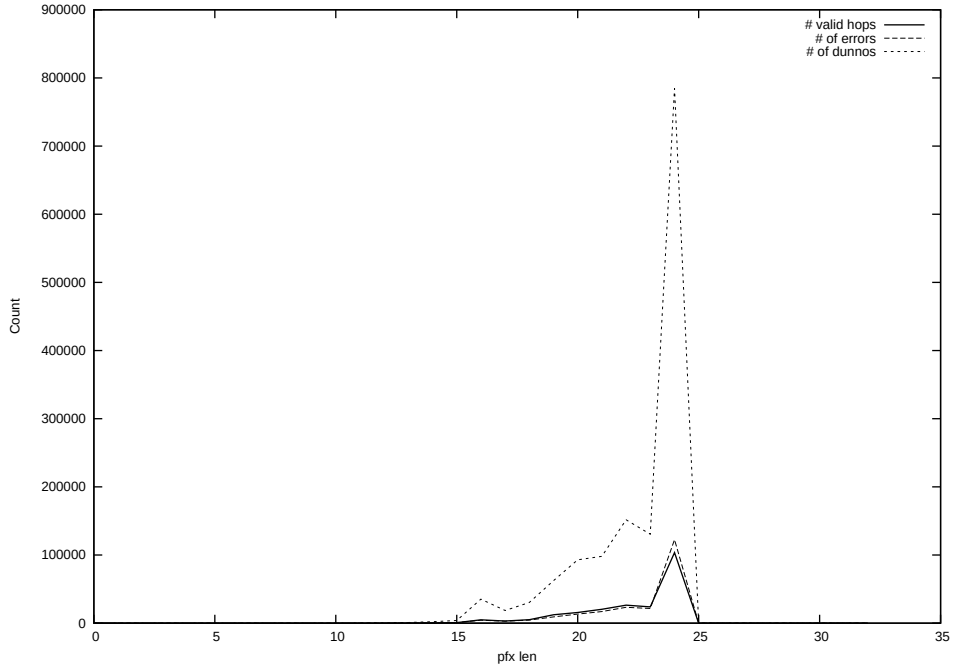
2014-07-21



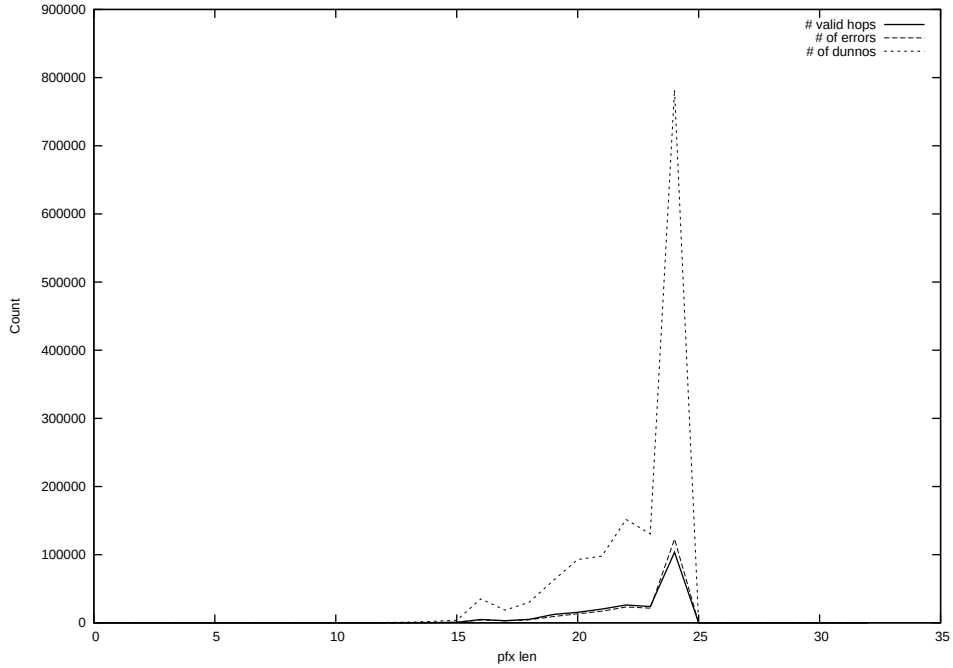
2014-07-22



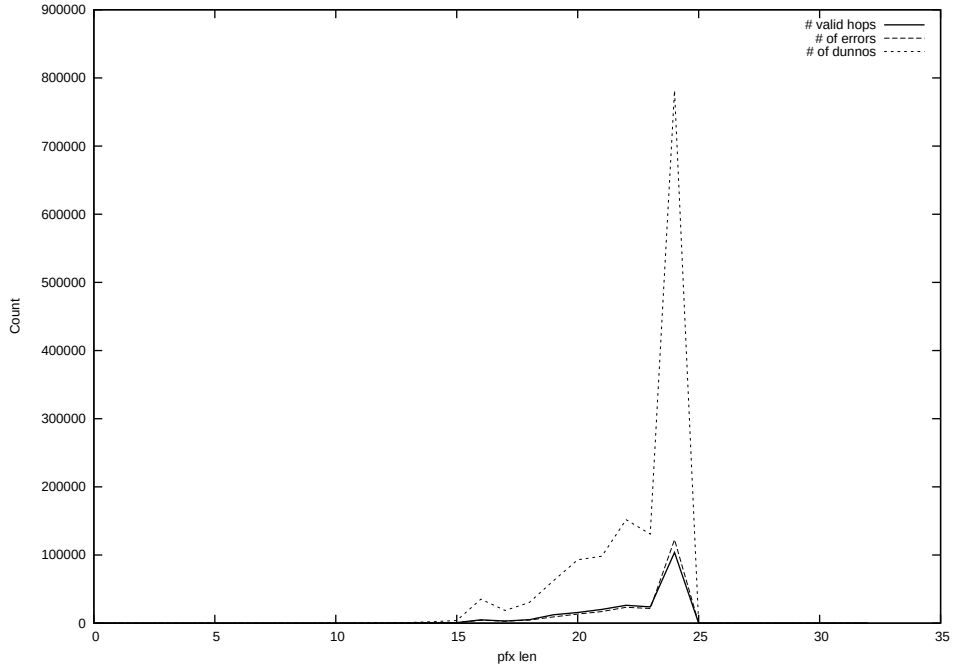
2014-07-23



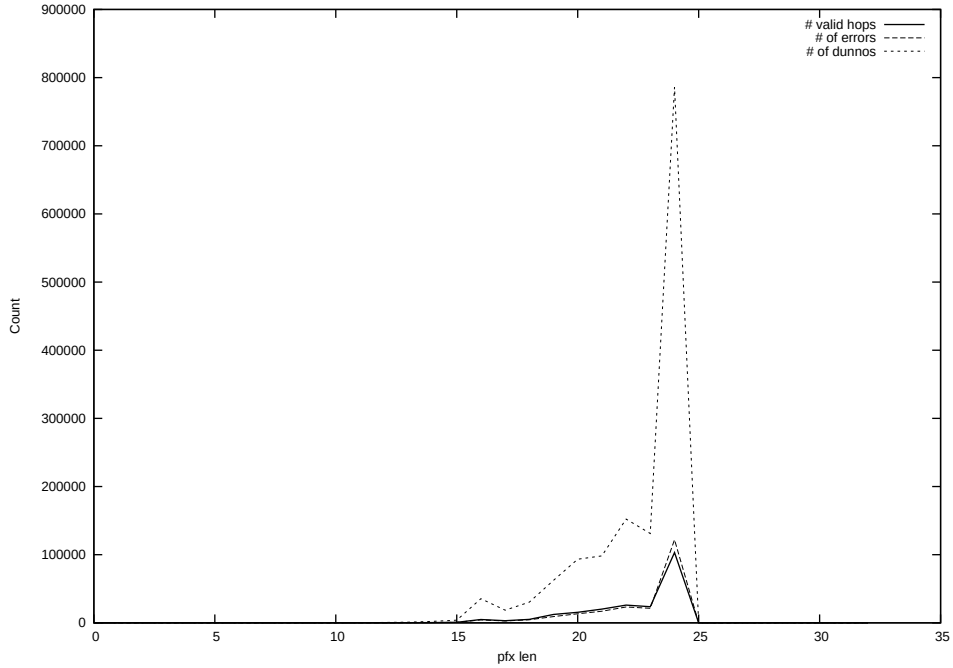
2014-07-24



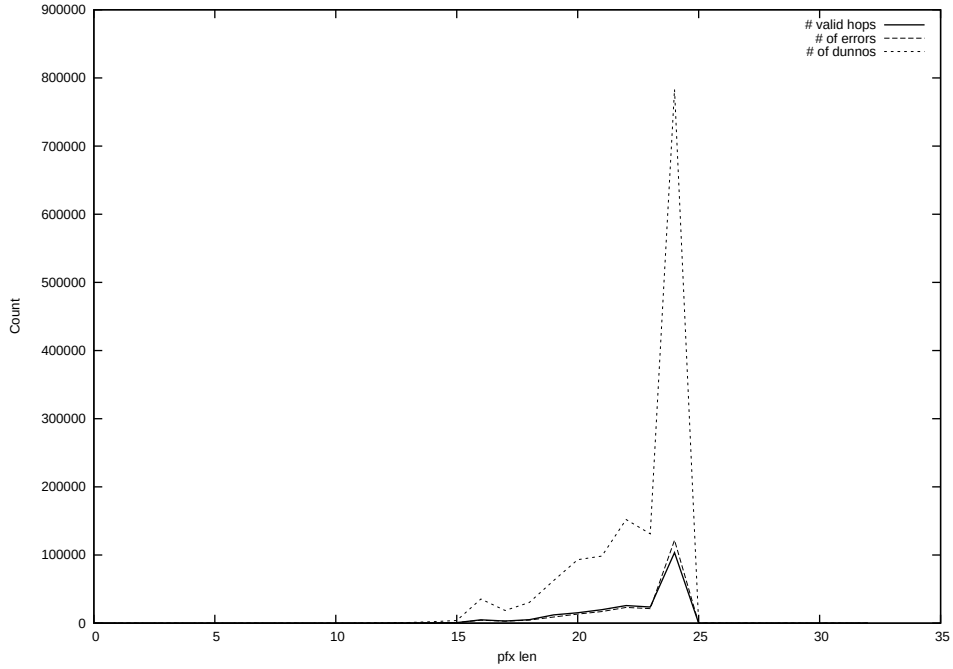
2014-07-25



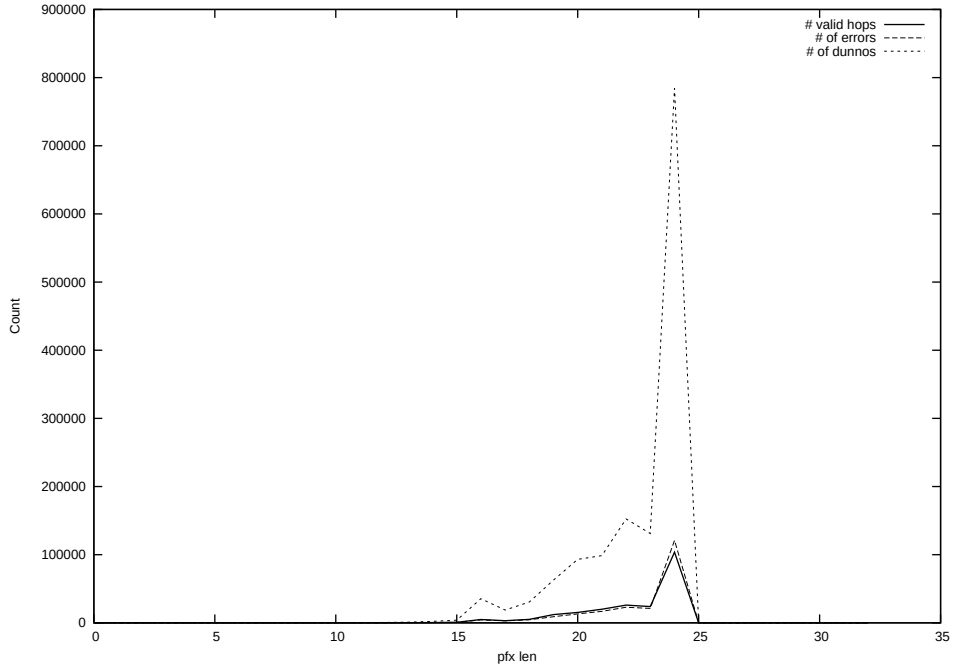
2014-07-26



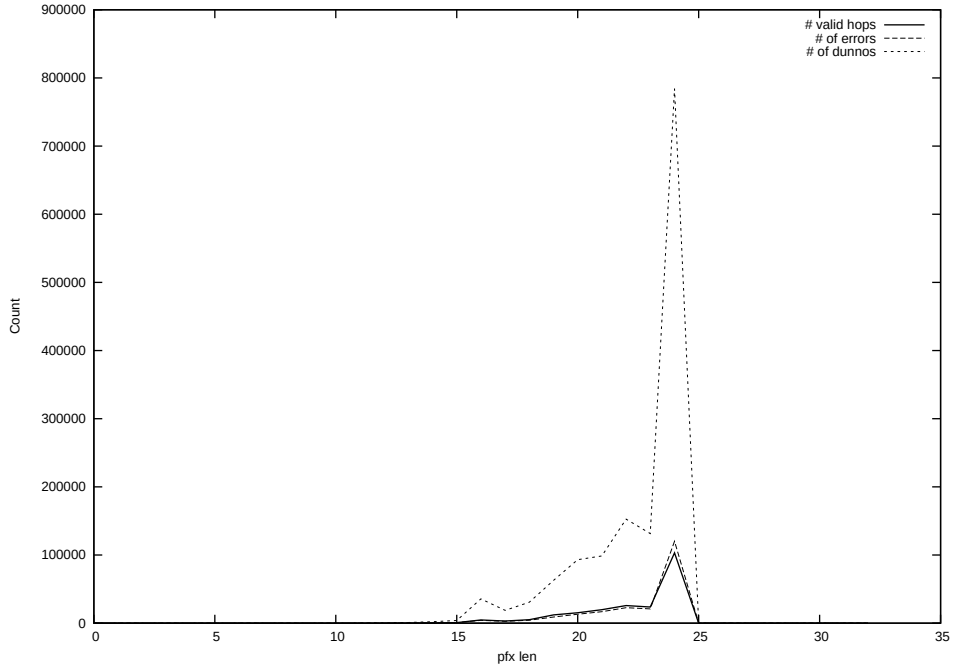
2014-07-27



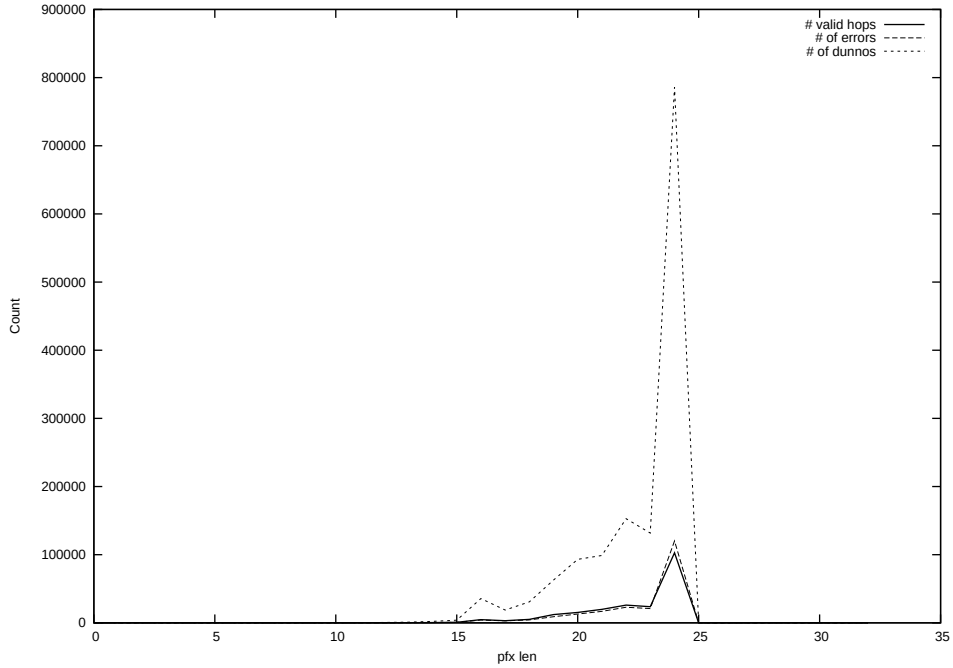
2014-07-28



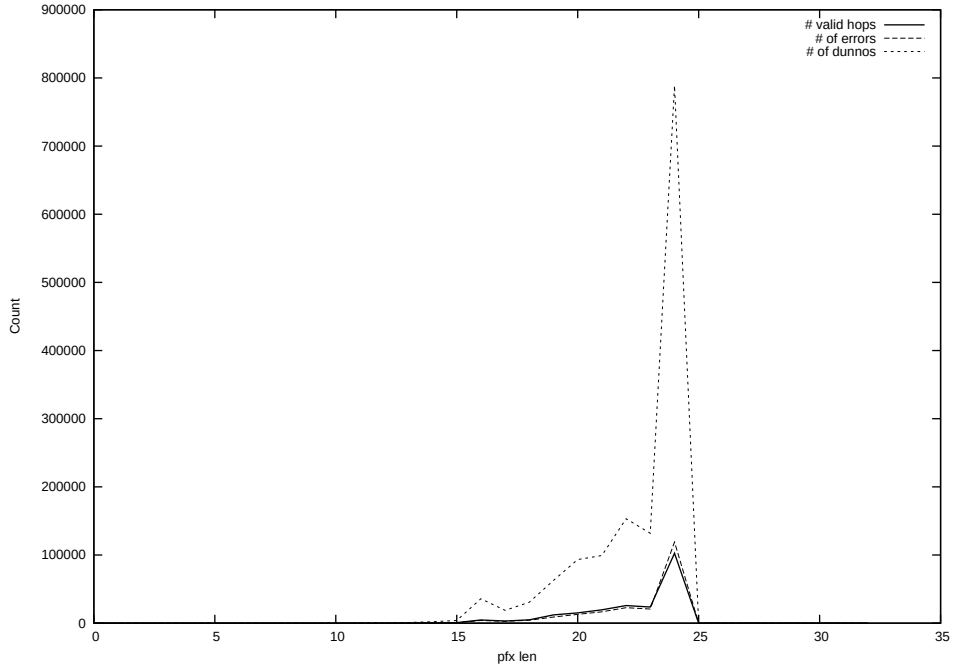
2014-07-29



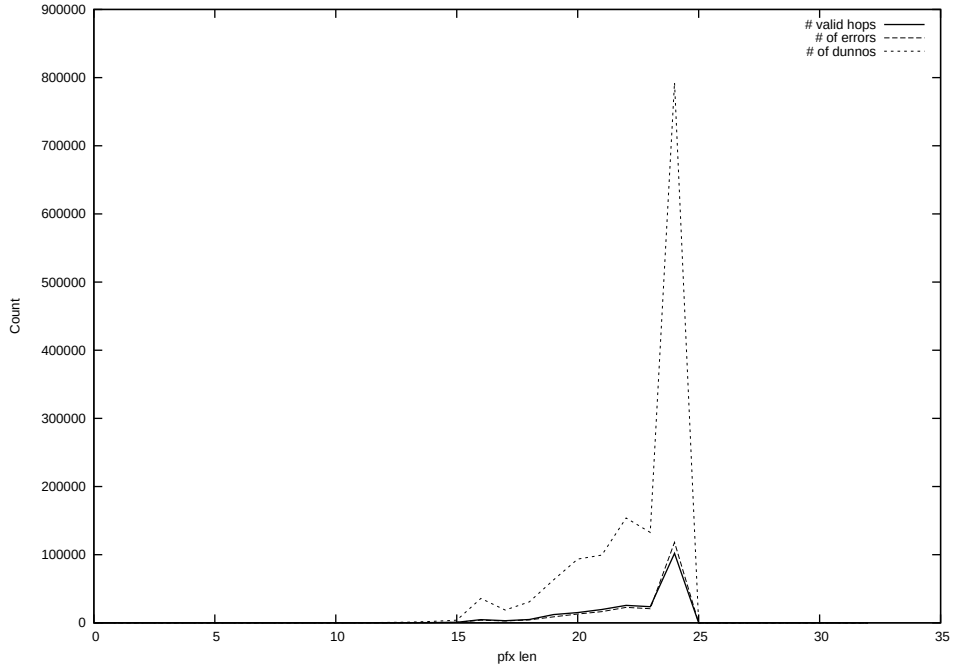
2014-07-30



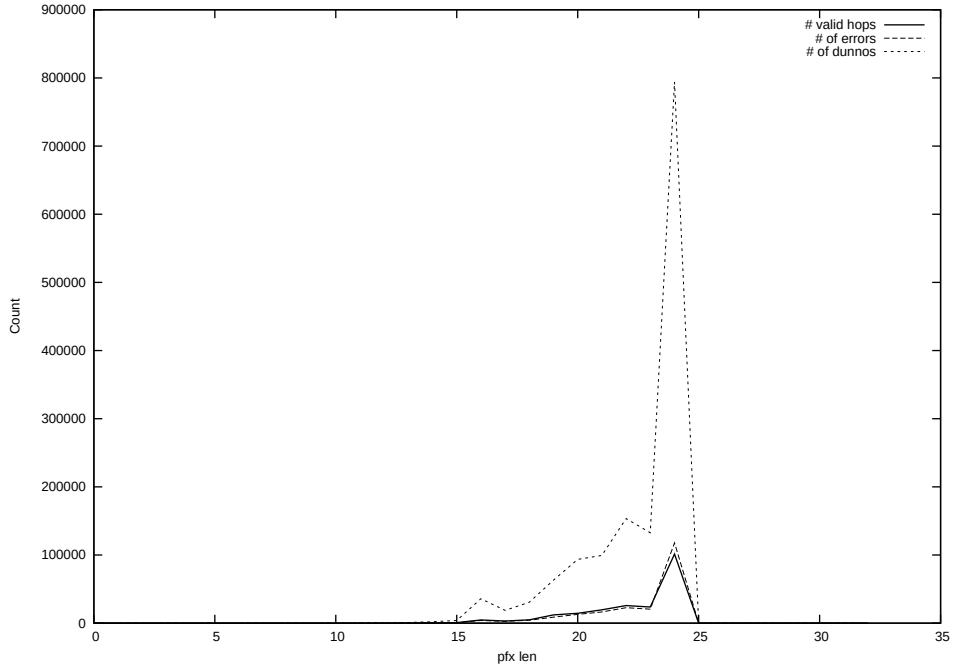
2014-07-31



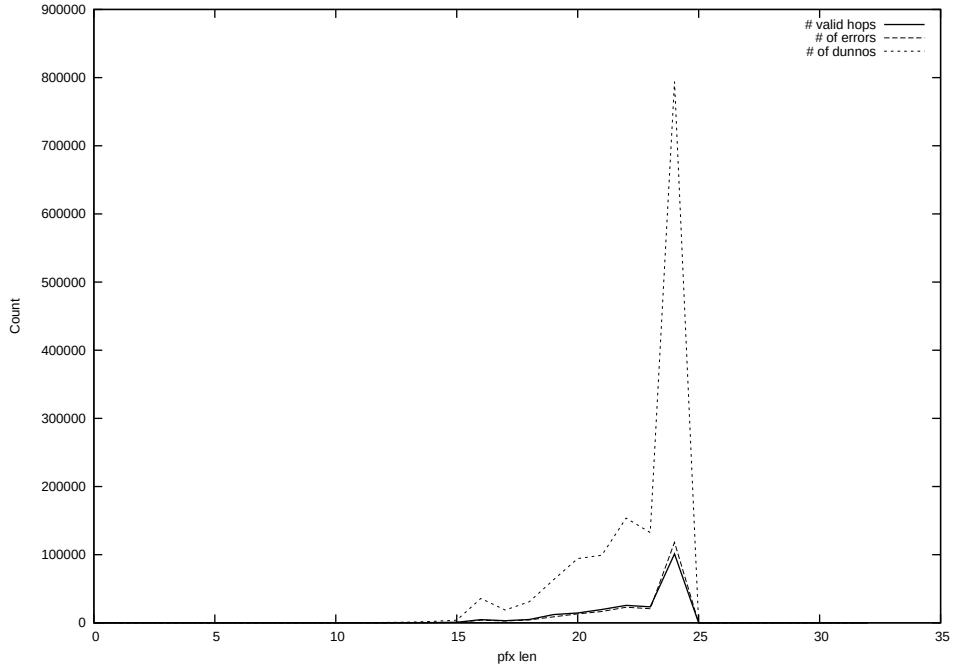
2014-08-01



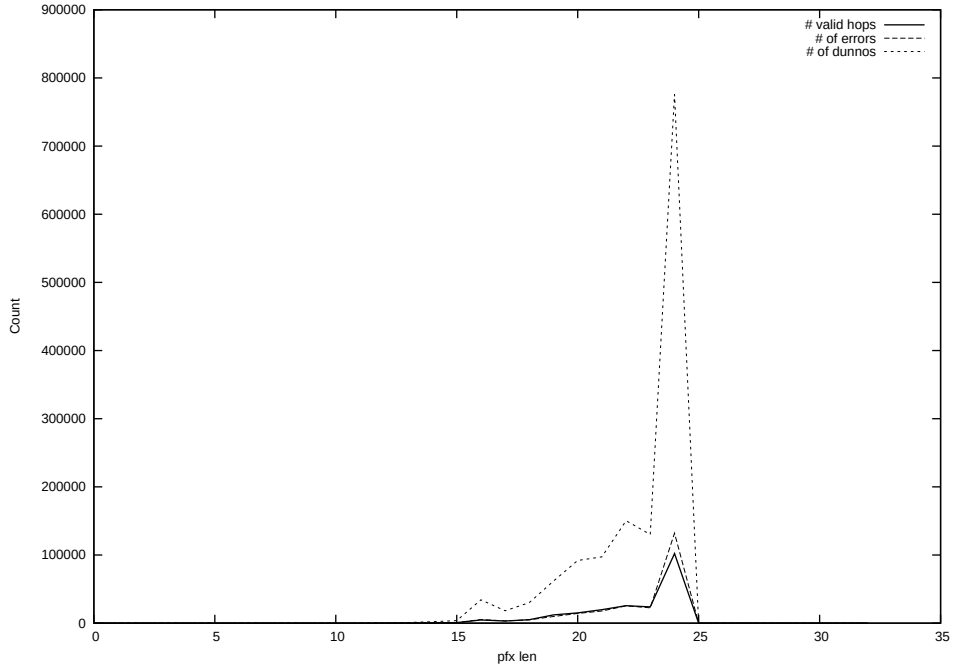
2014-08-02



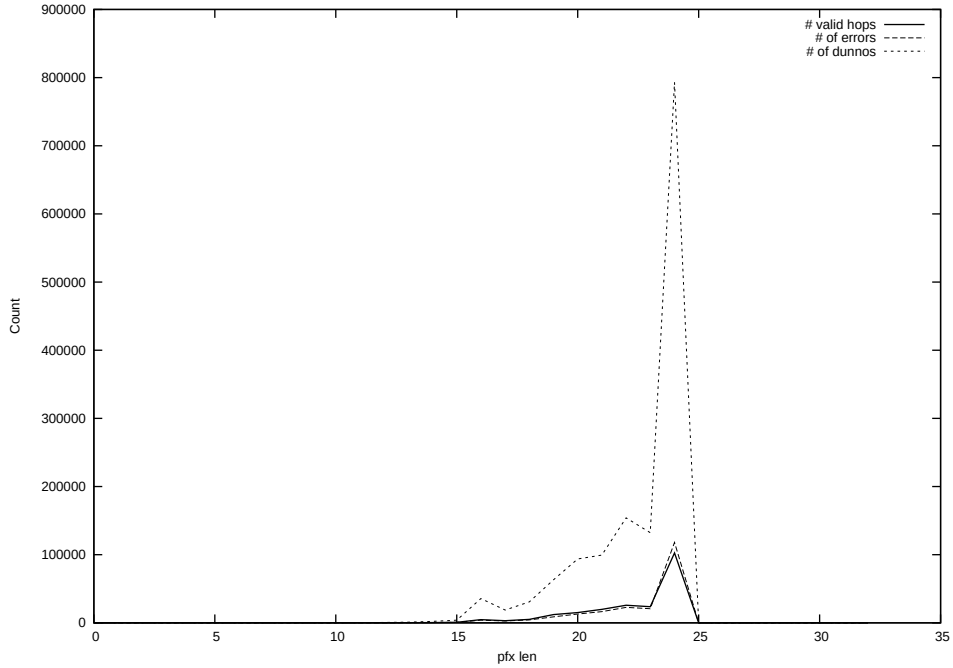
2014-08-03



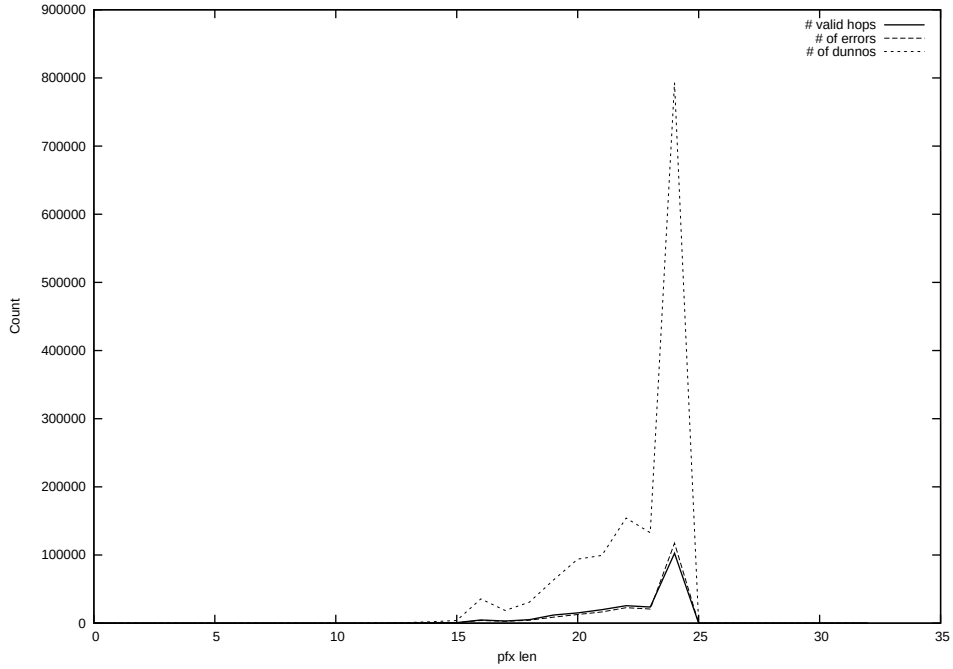
2014-08-04



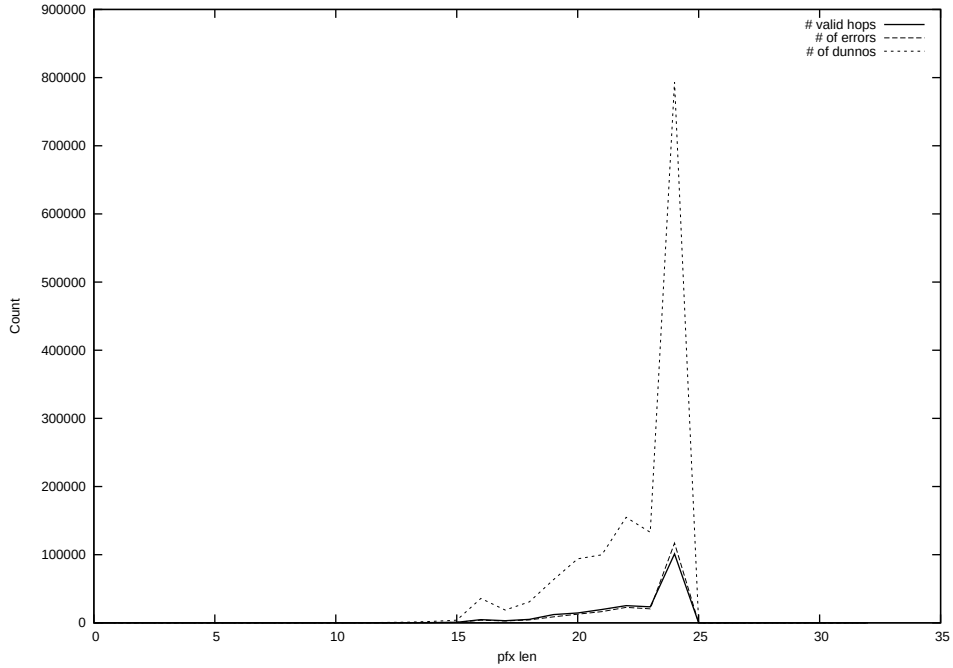
2014-08-05



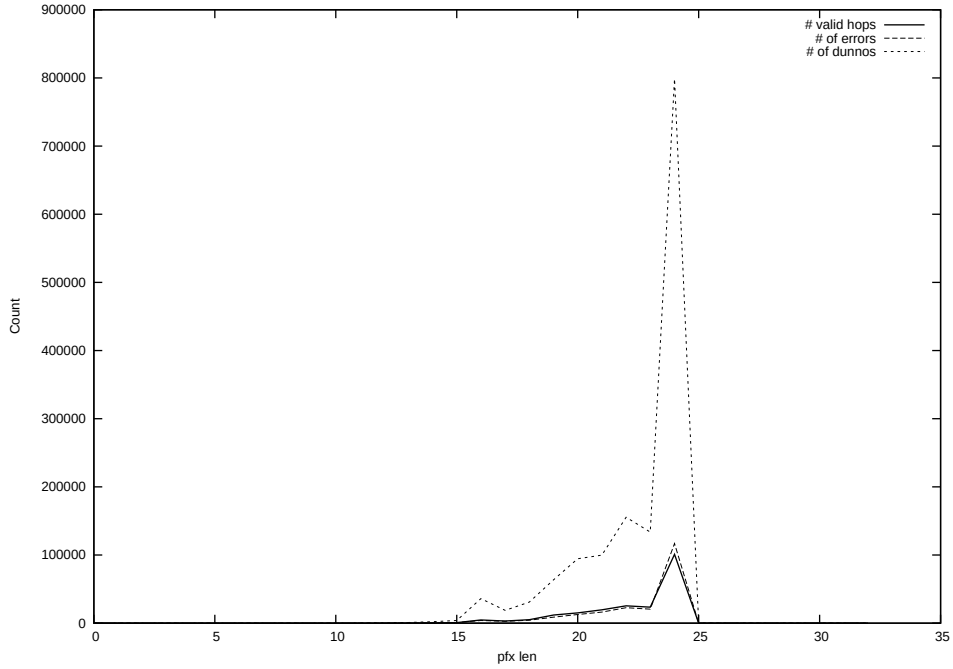
2014-08-06



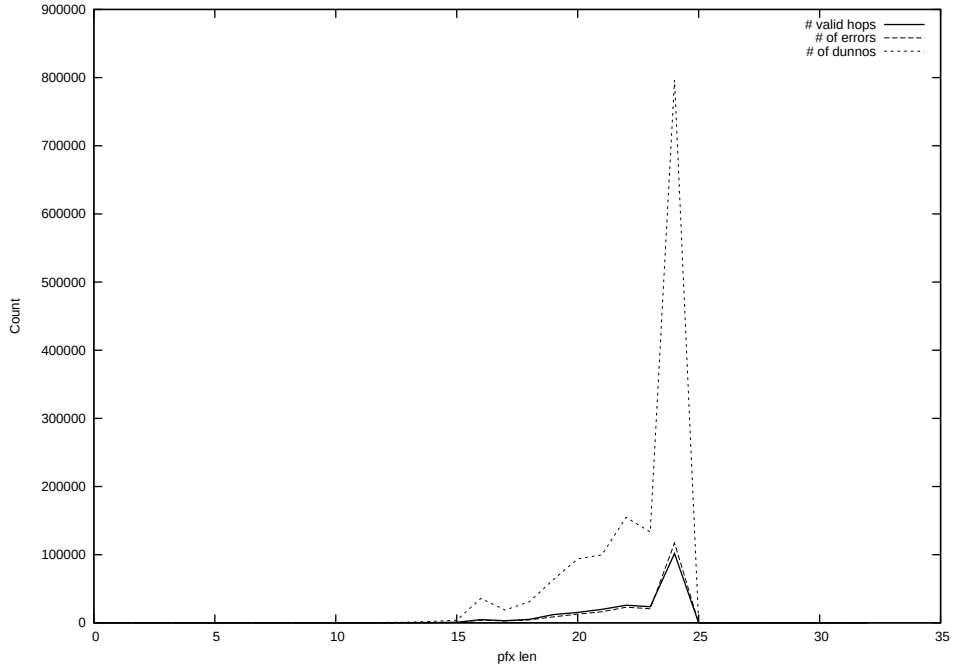
2014-08-07



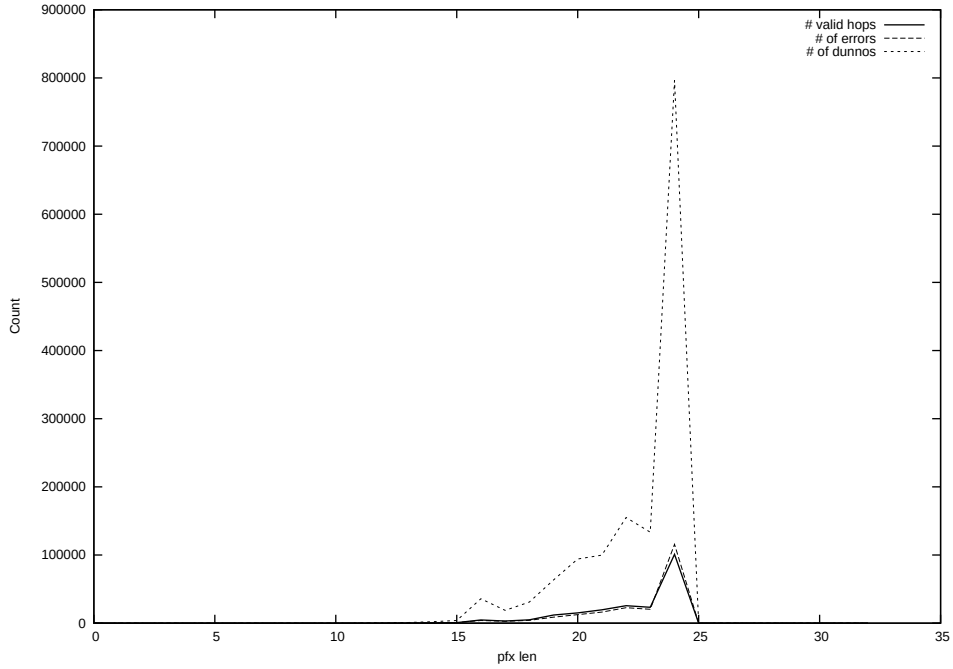
2014-08-08



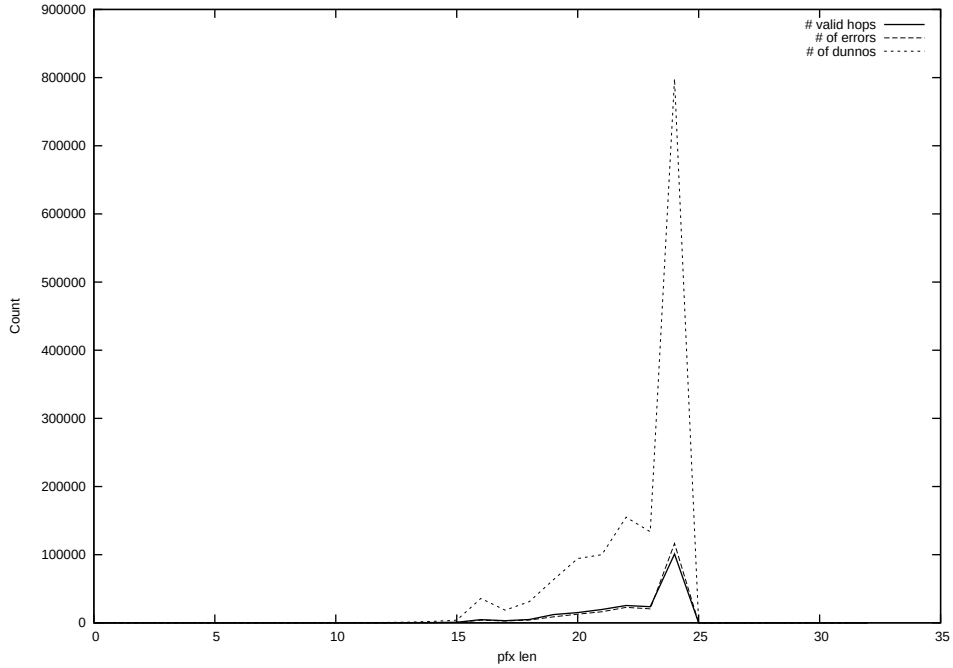
2014-08-09



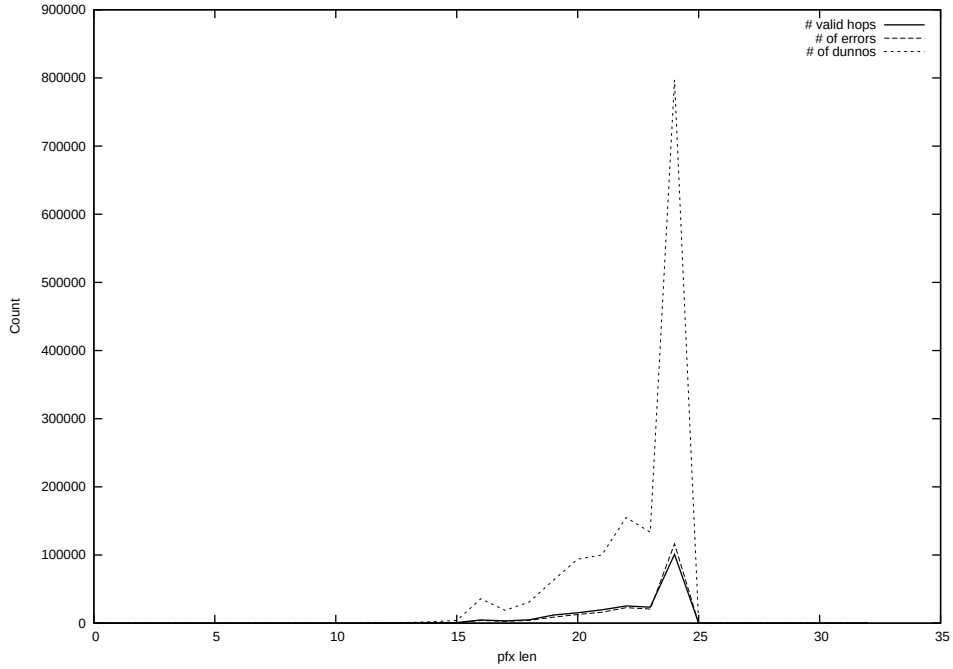
2014-08-10



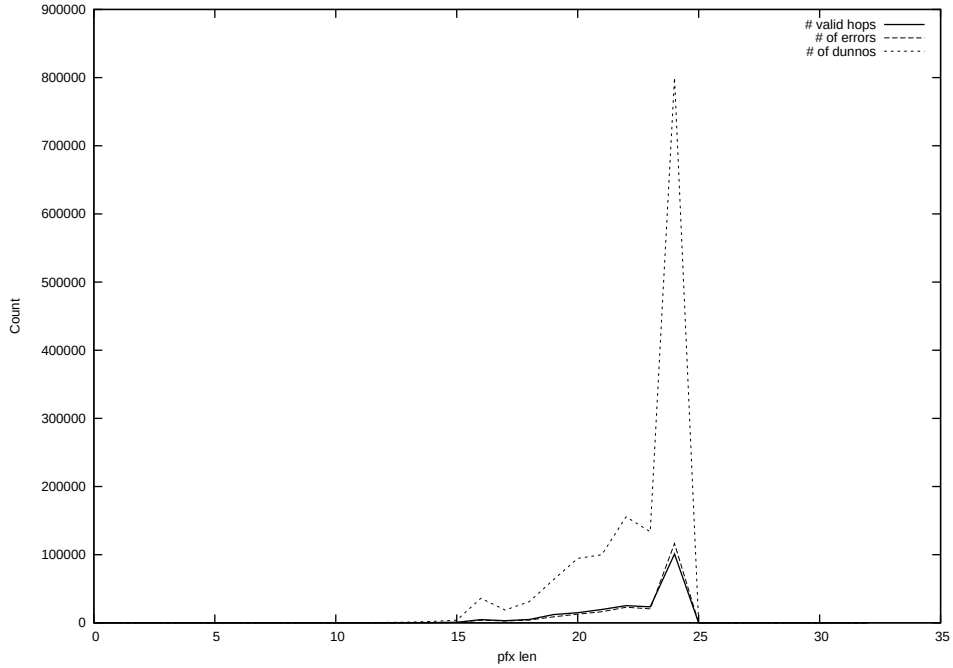
2014-08-11



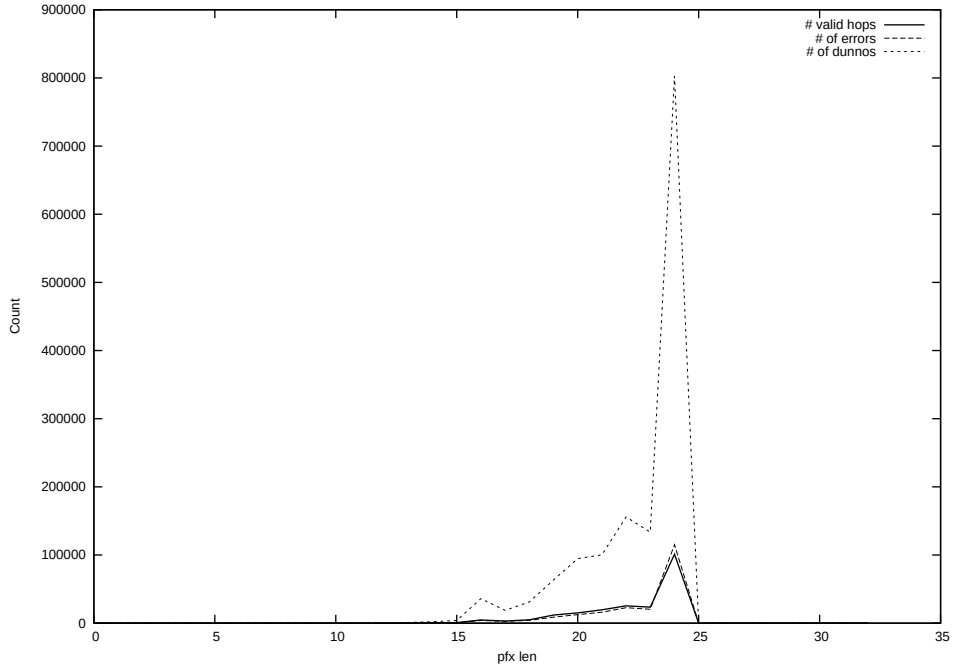
2014-08-12



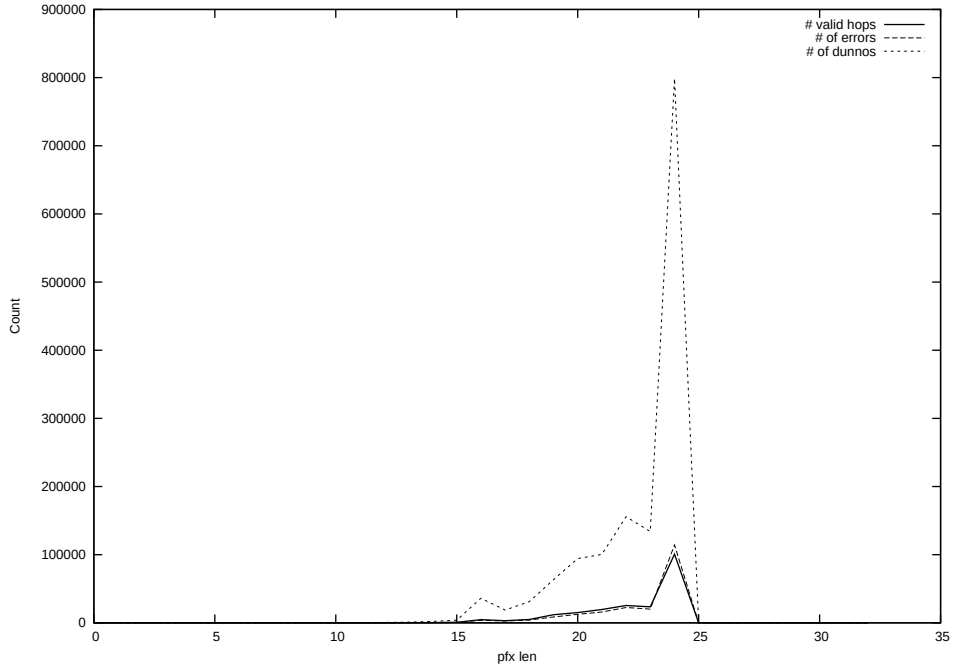
2014-08-13



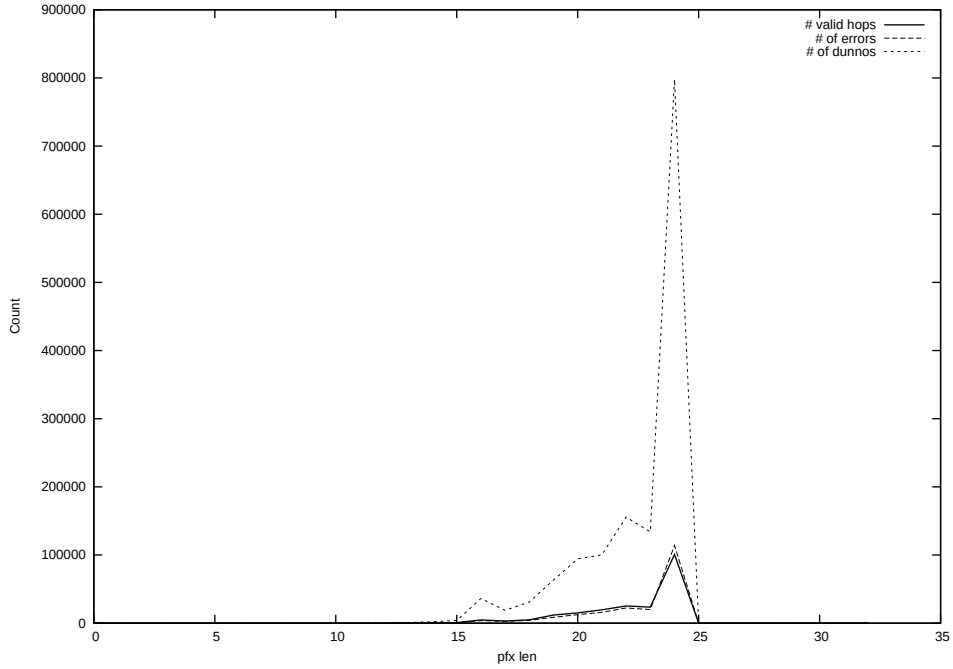
2014-08-14



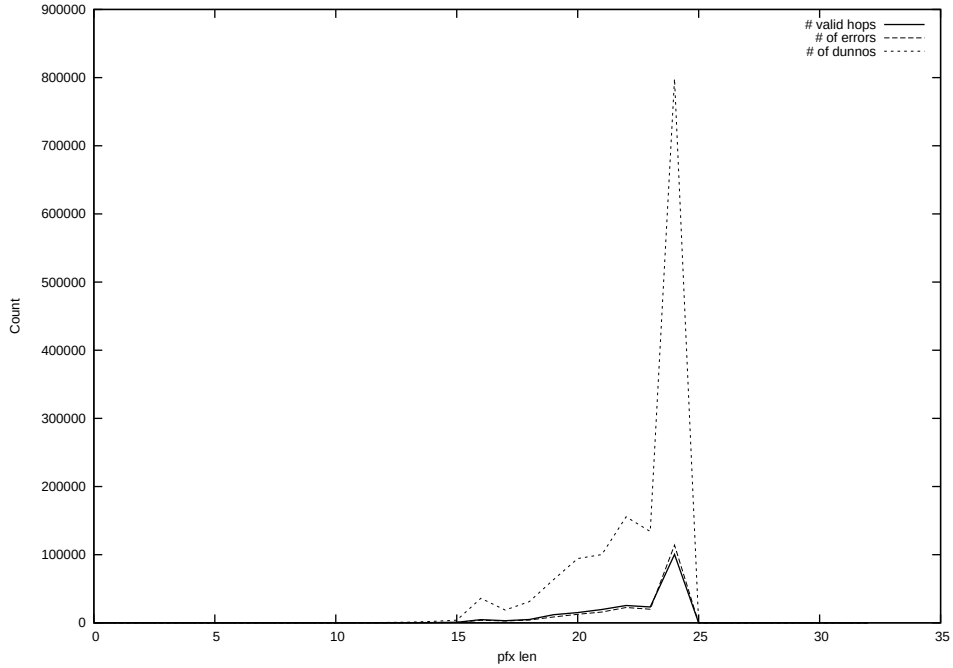
2014-08-15



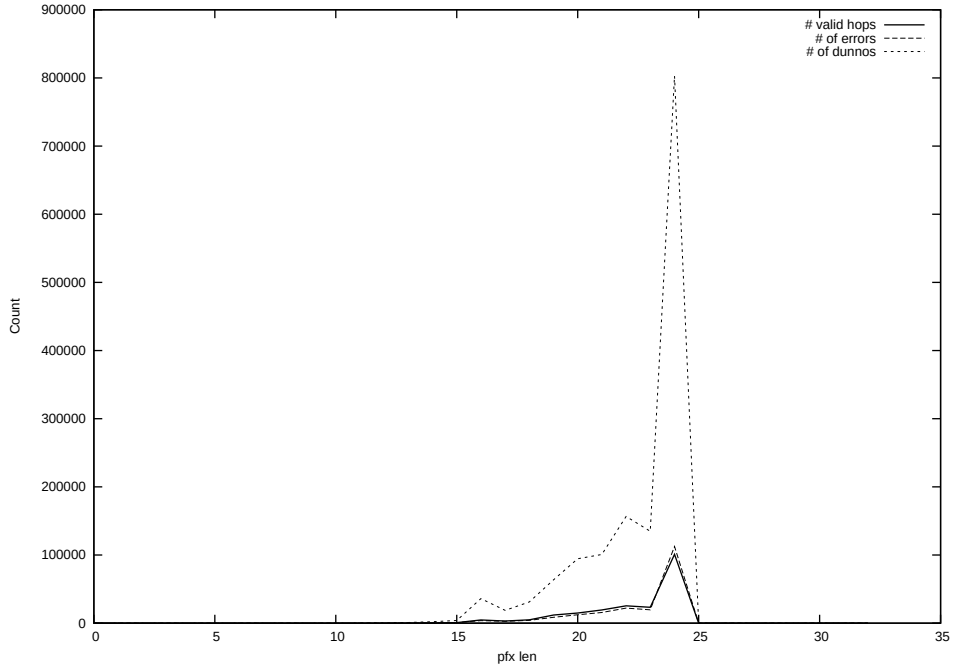
2014-08-16



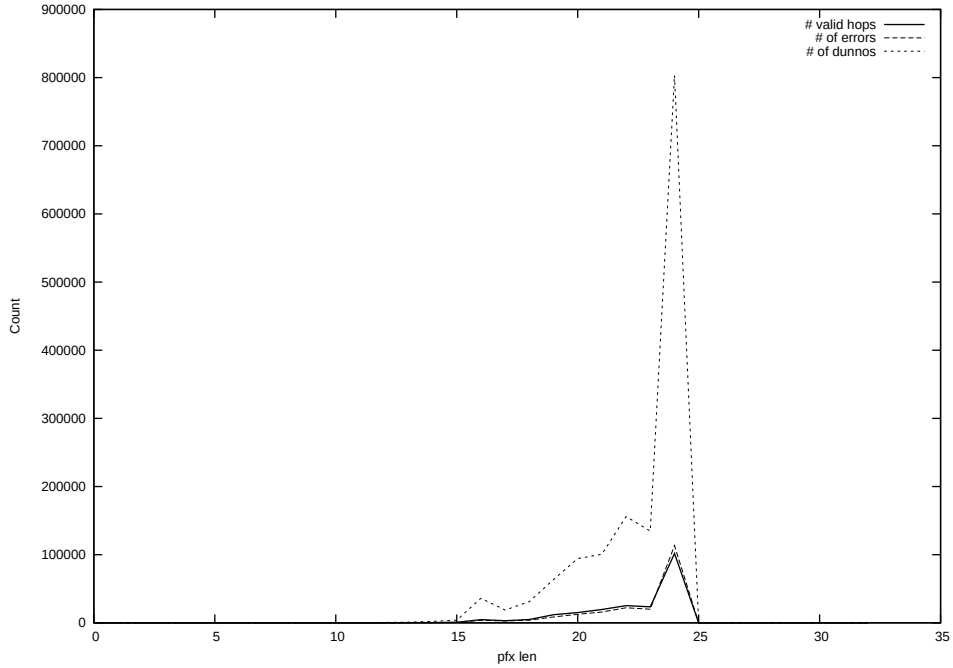
2014-08-17



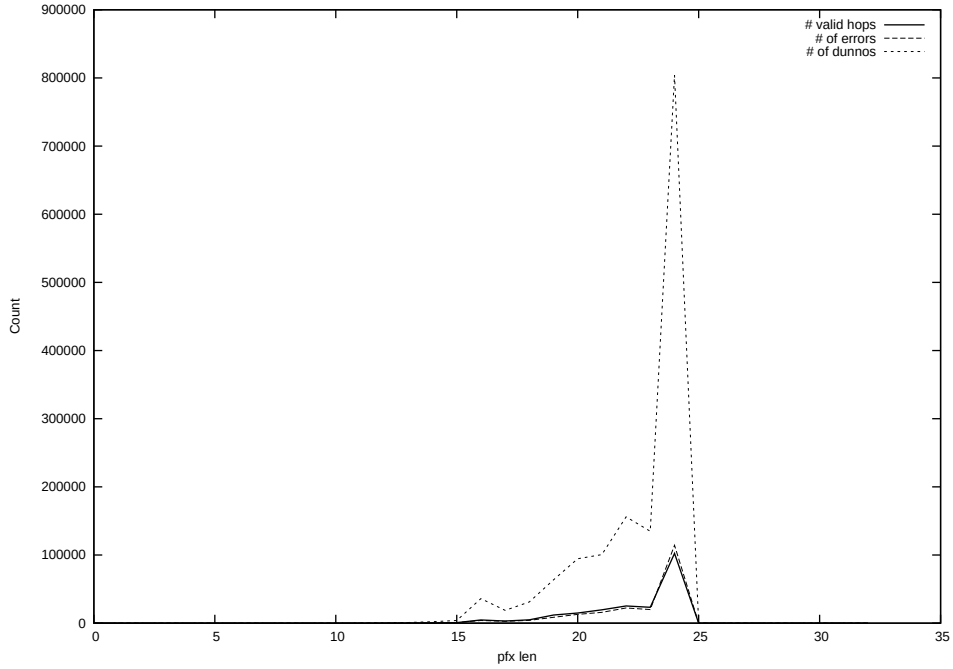
2014-08-18



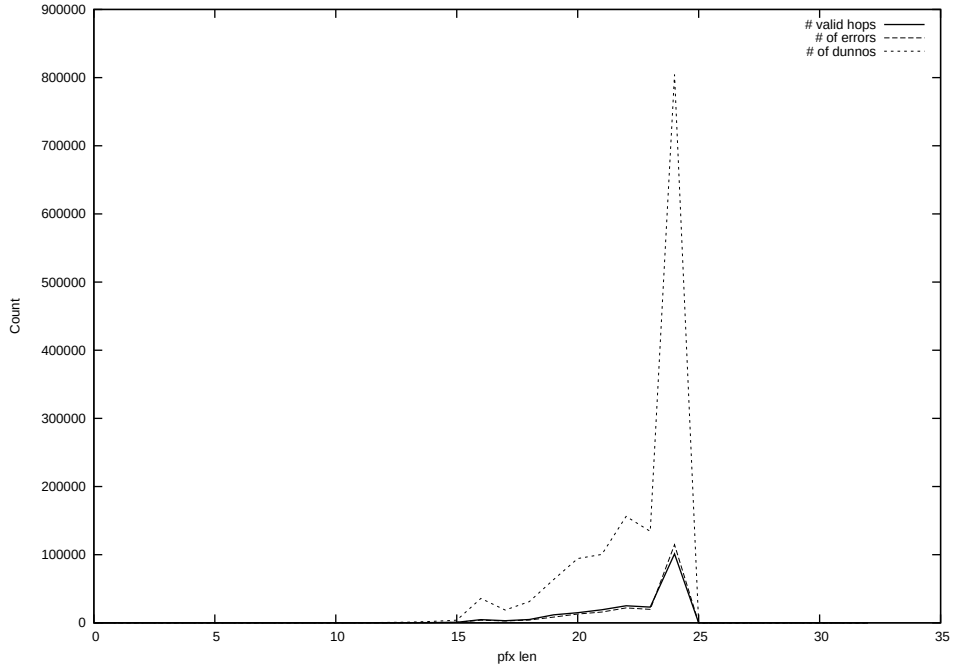
2014-08-19



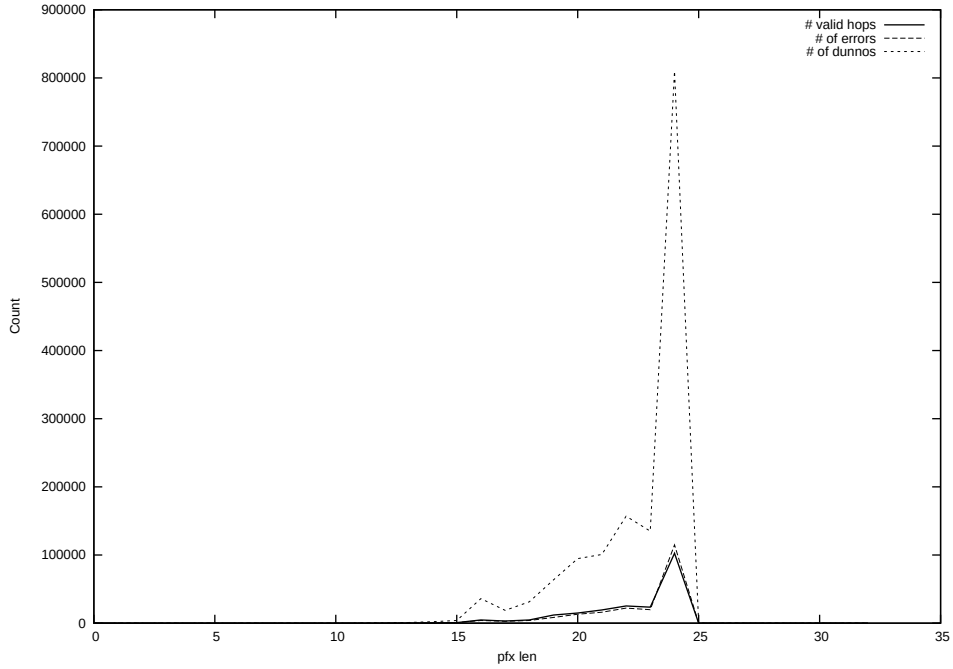
2014-08-20



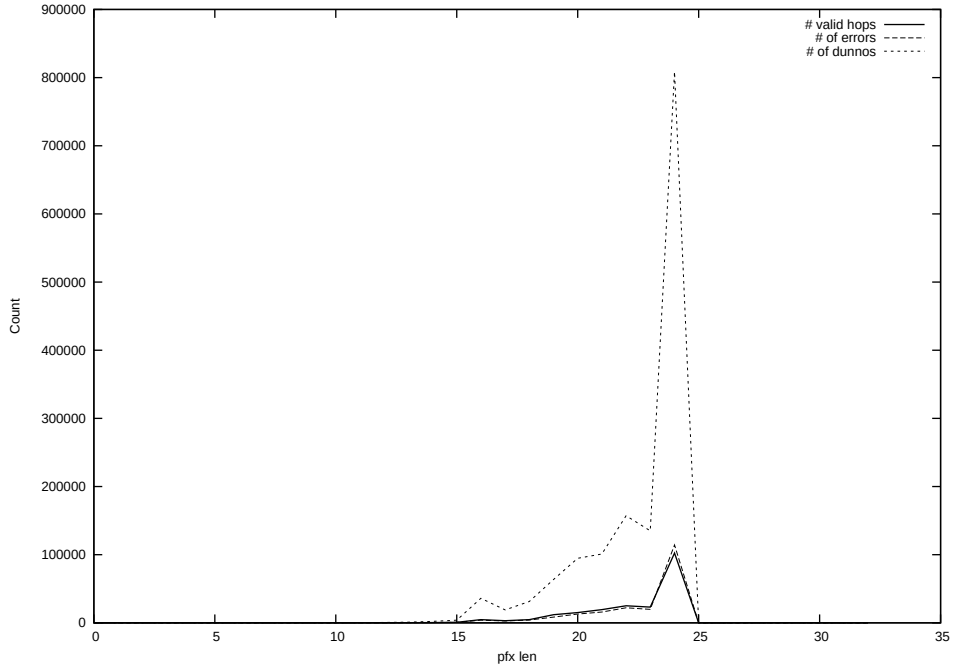
2014-08-21



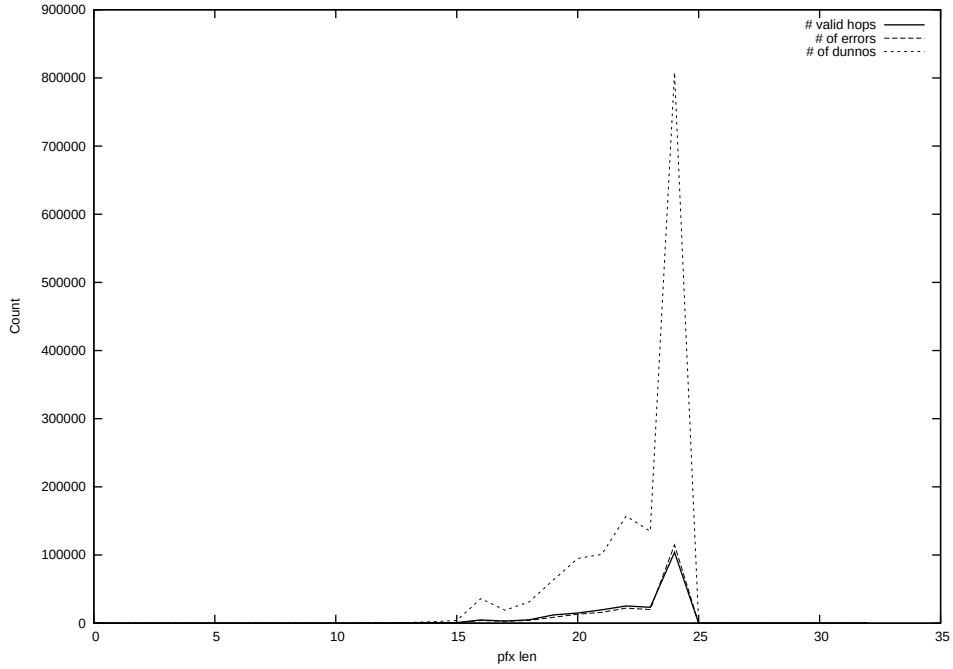
2014-08-22



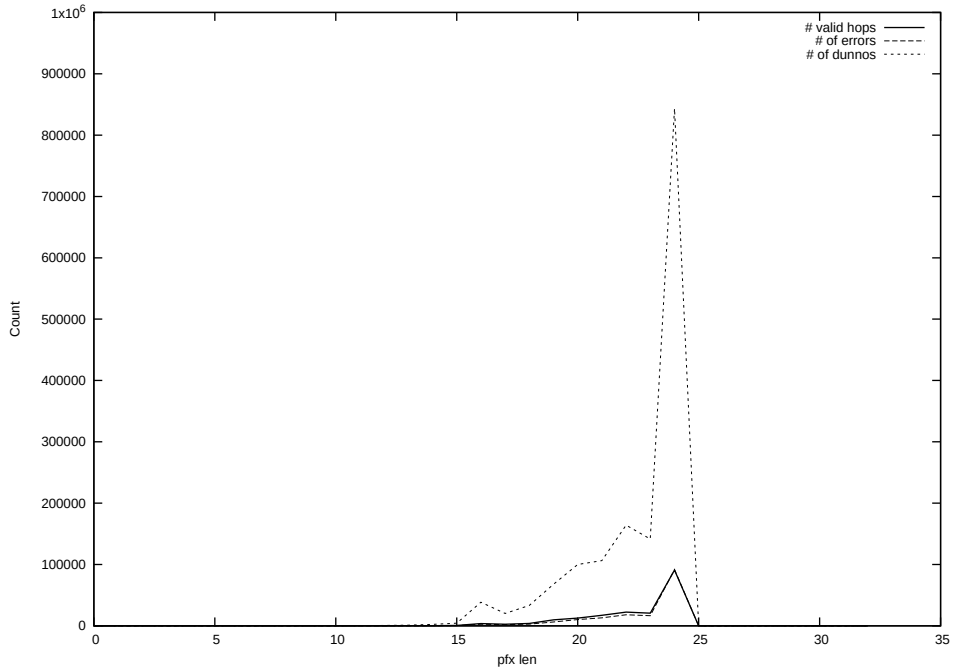
2014-08-23



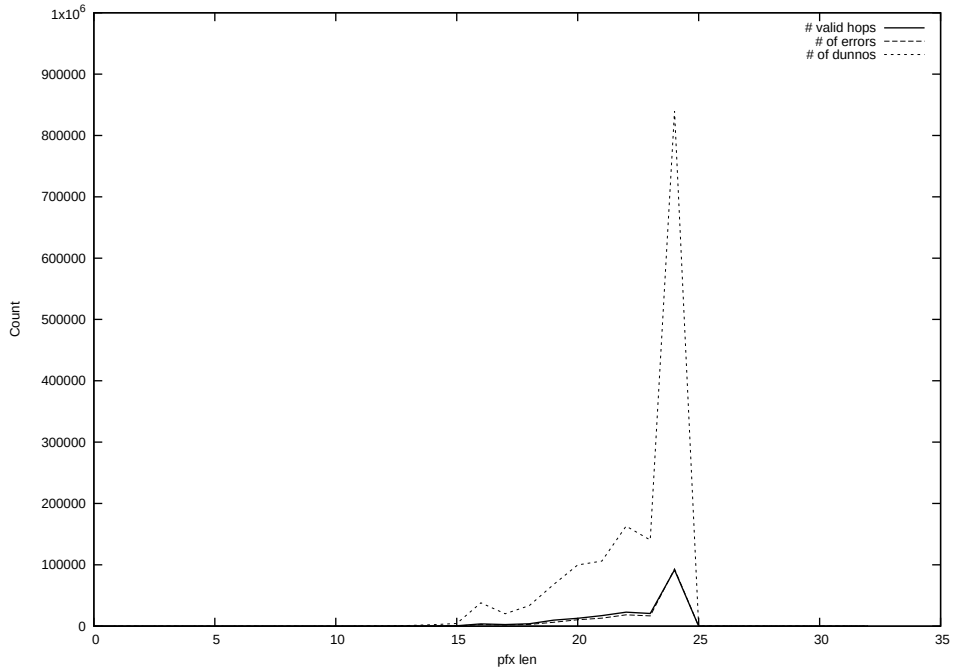
2014-08-24



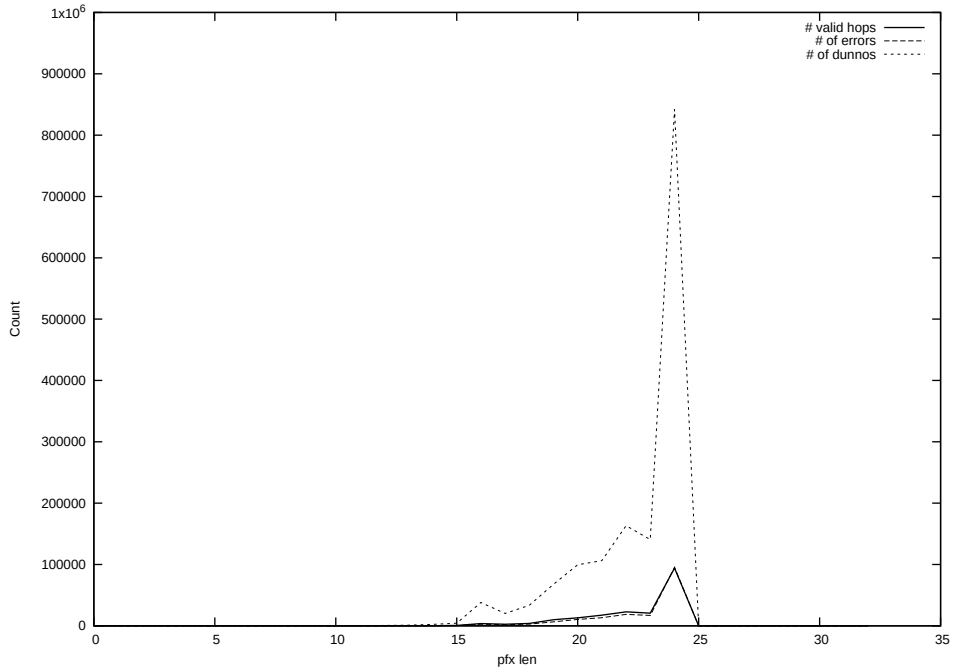
2014-08-25



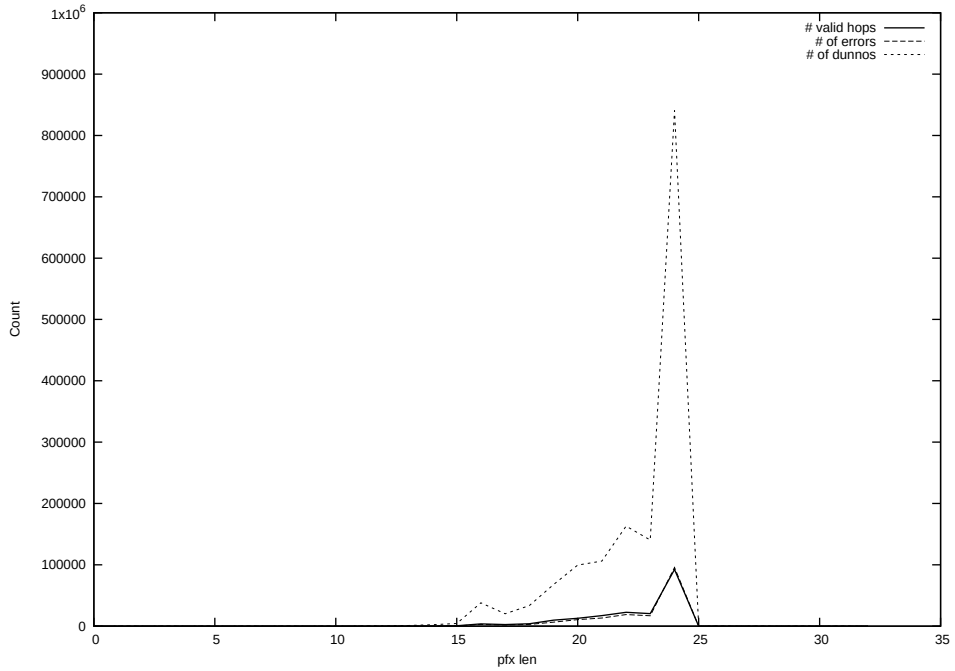
2014-08-26



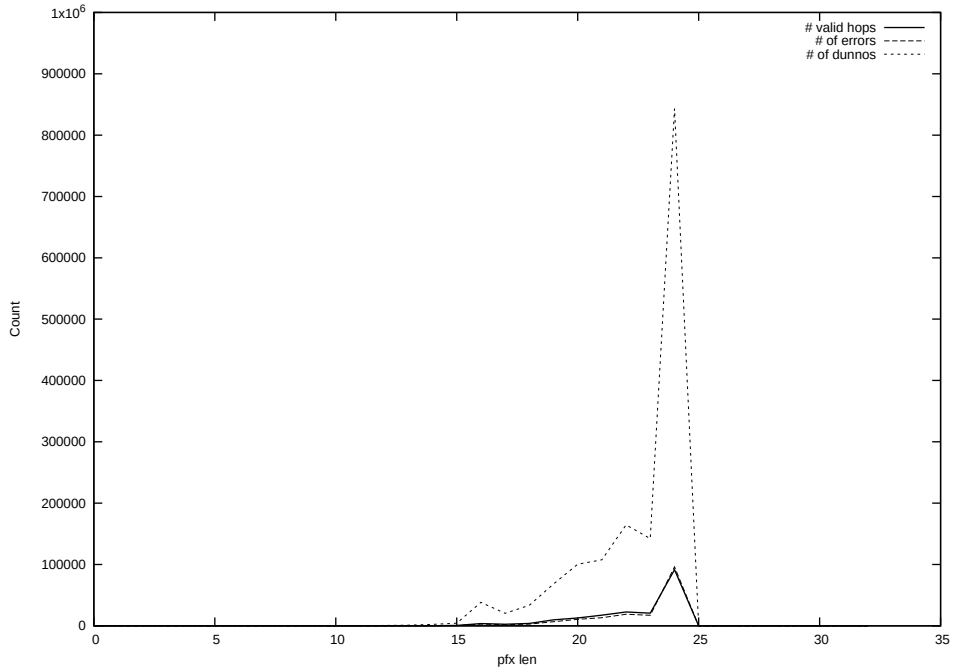
2014-08-27



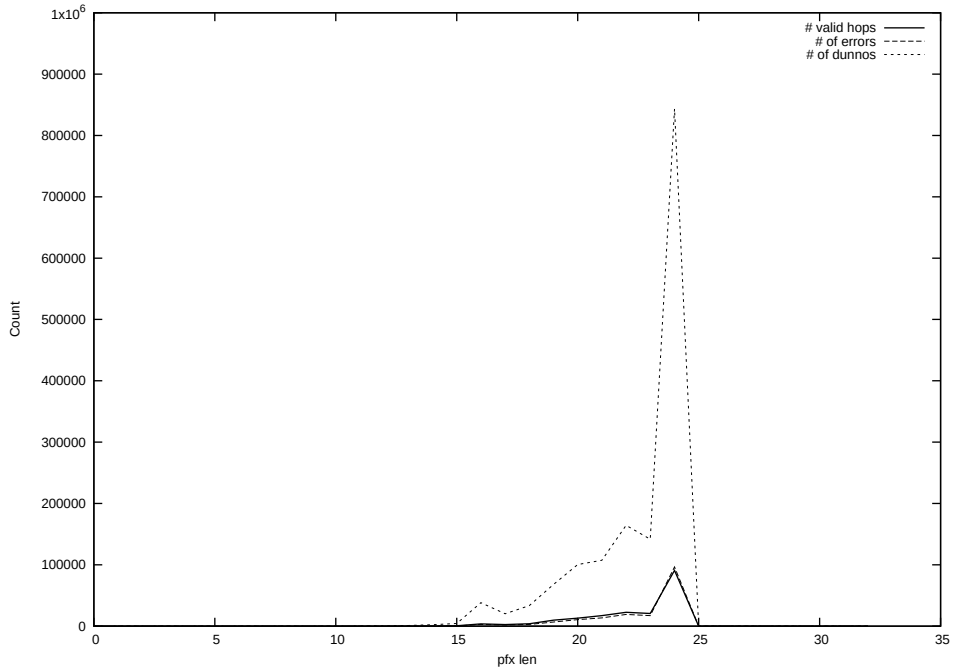
2014-08-28



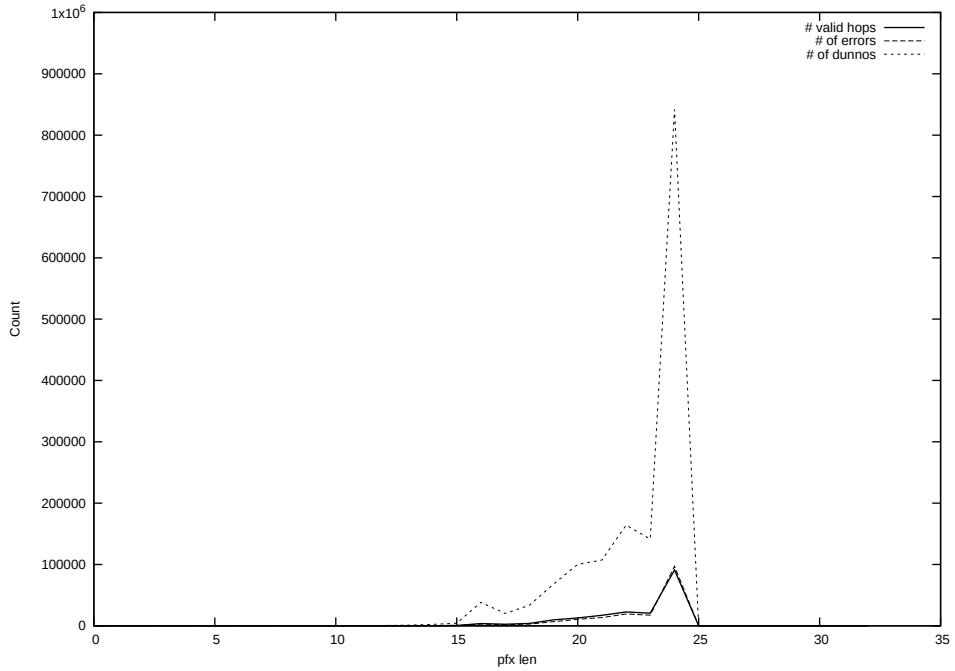
2014-08-29



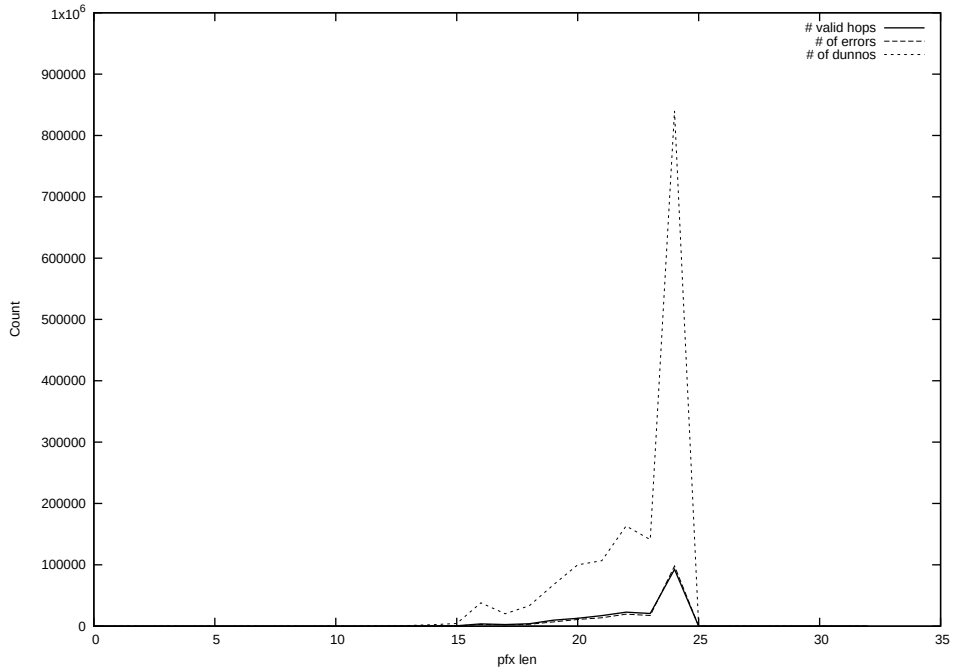
2014-08-30



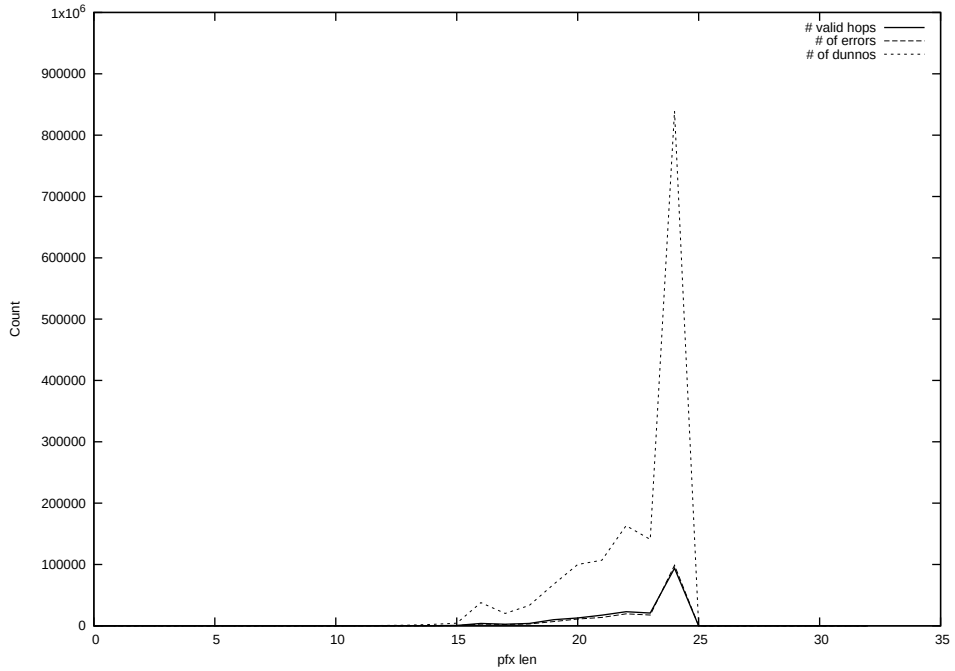
2014-08-31



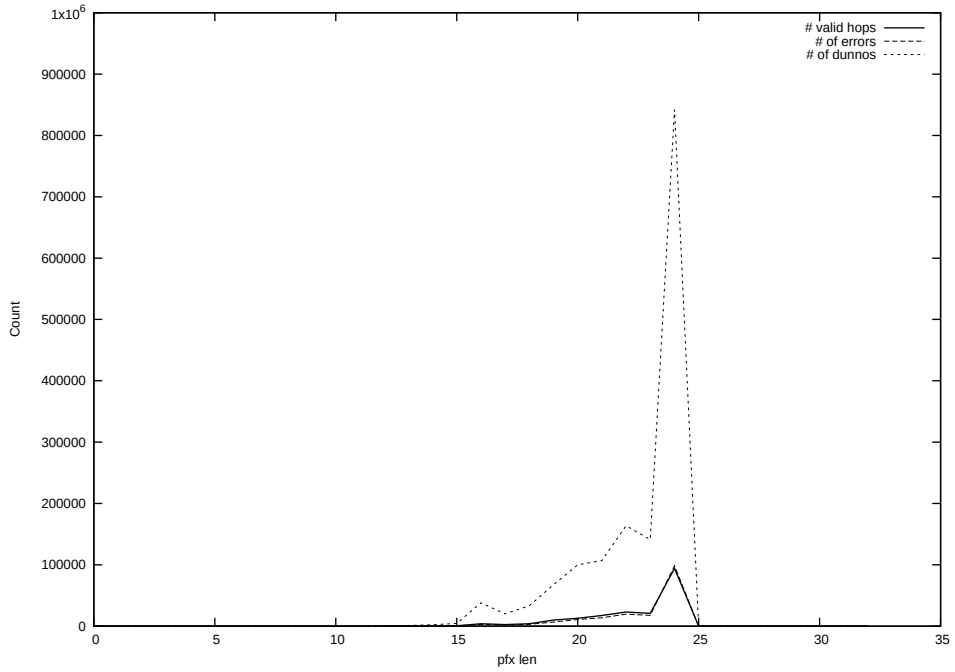
2014-09-01



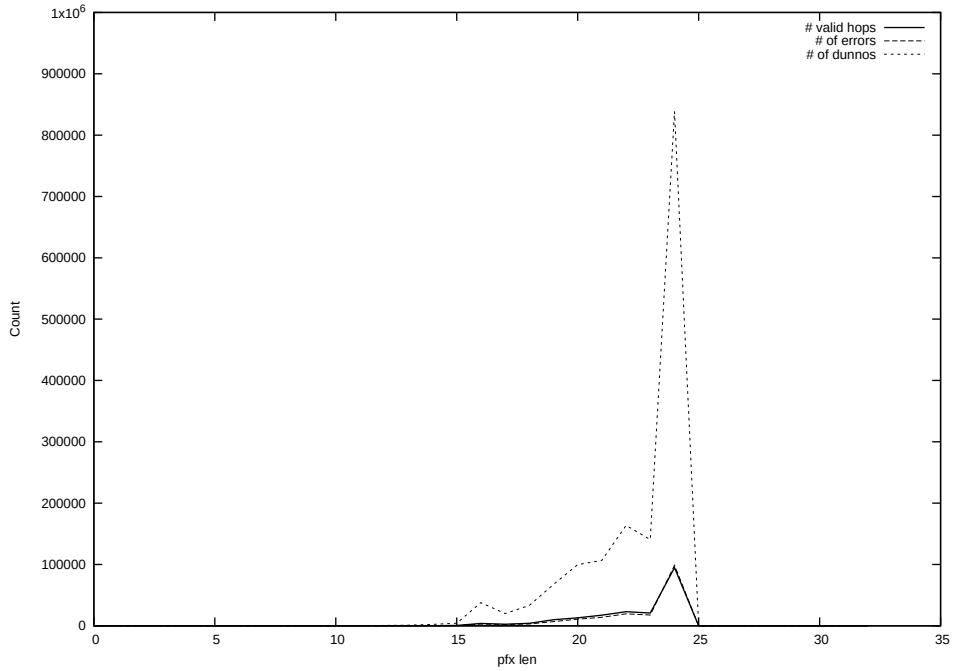
2014-09-02



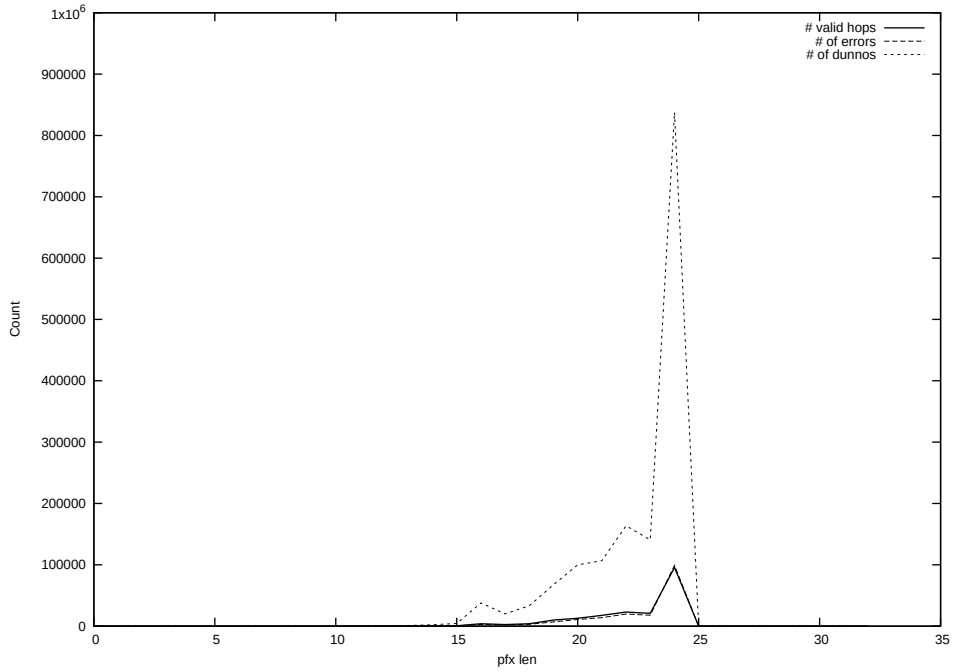
2014-09-03



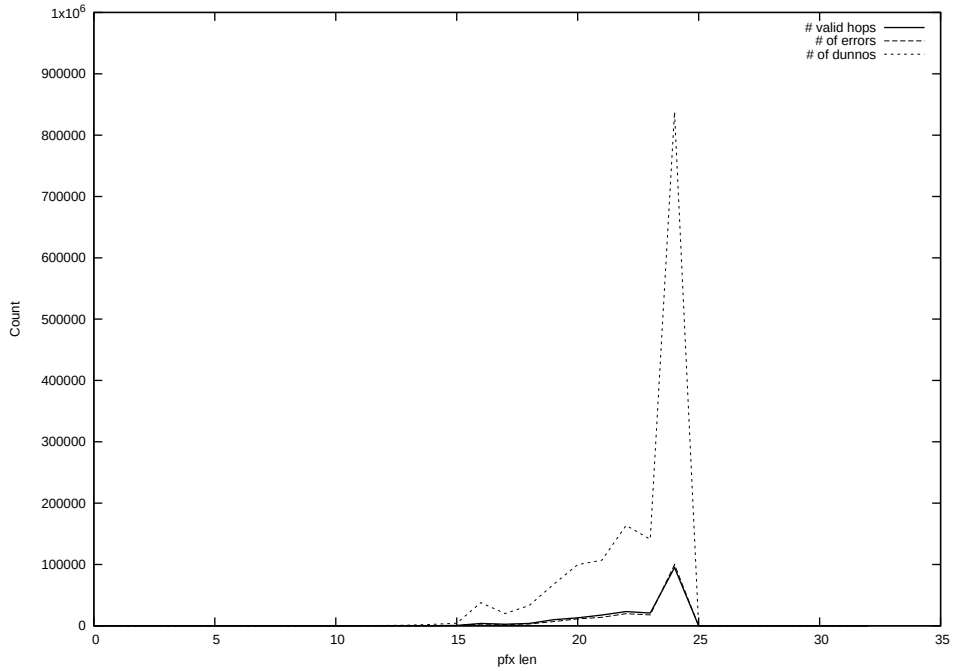
2014-09-04



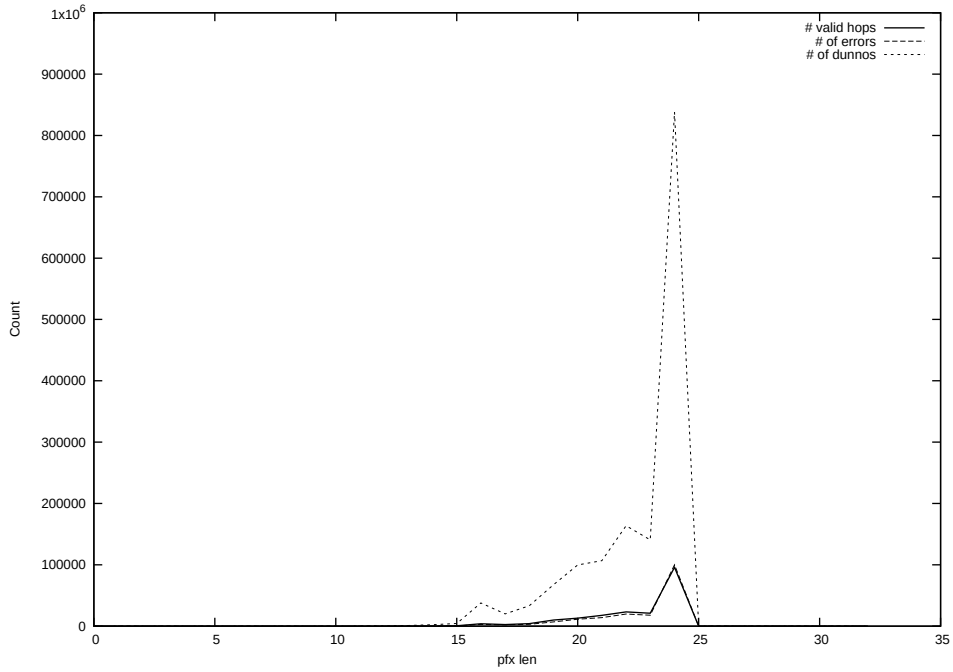
2014-09-05



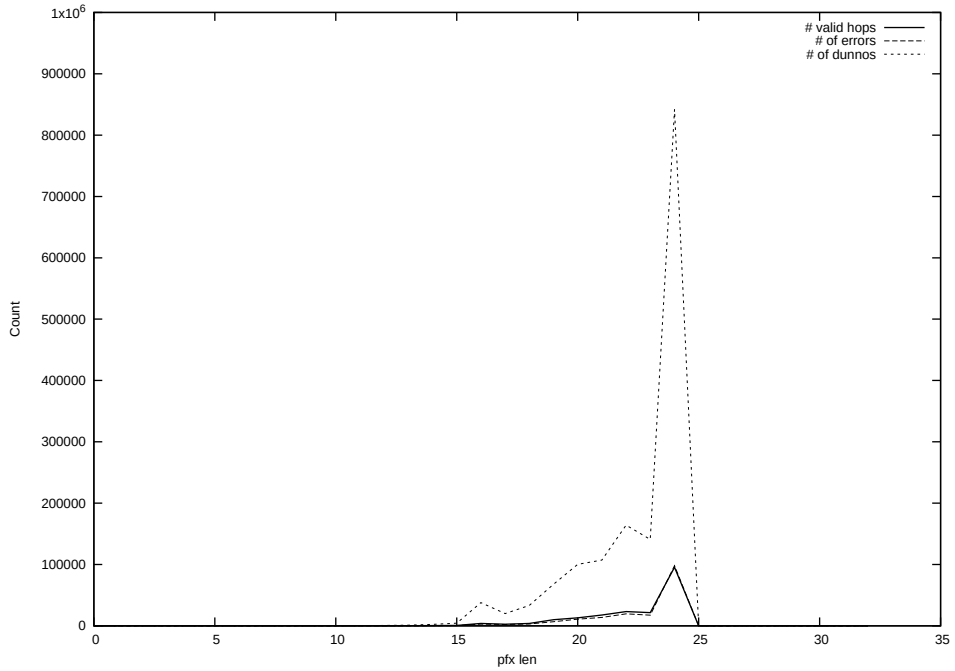
2014-09-06



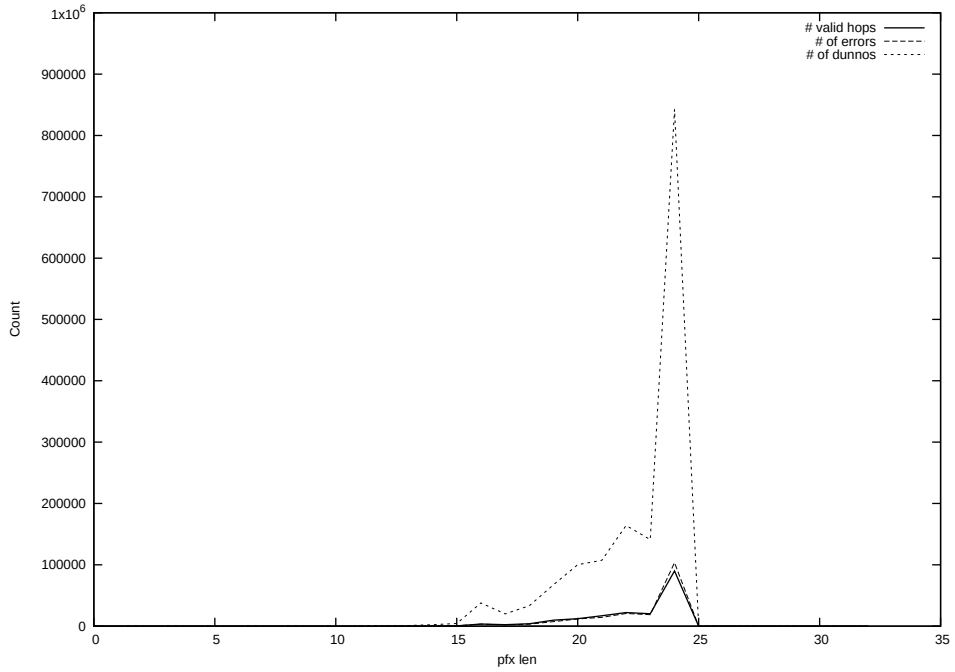
2014-09-07



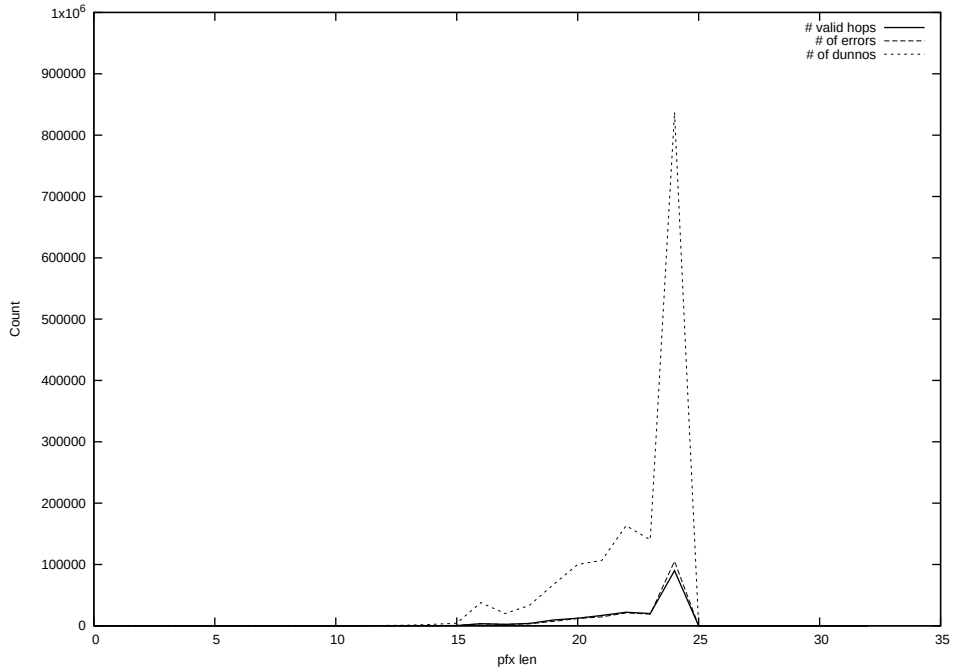
2014-09-08



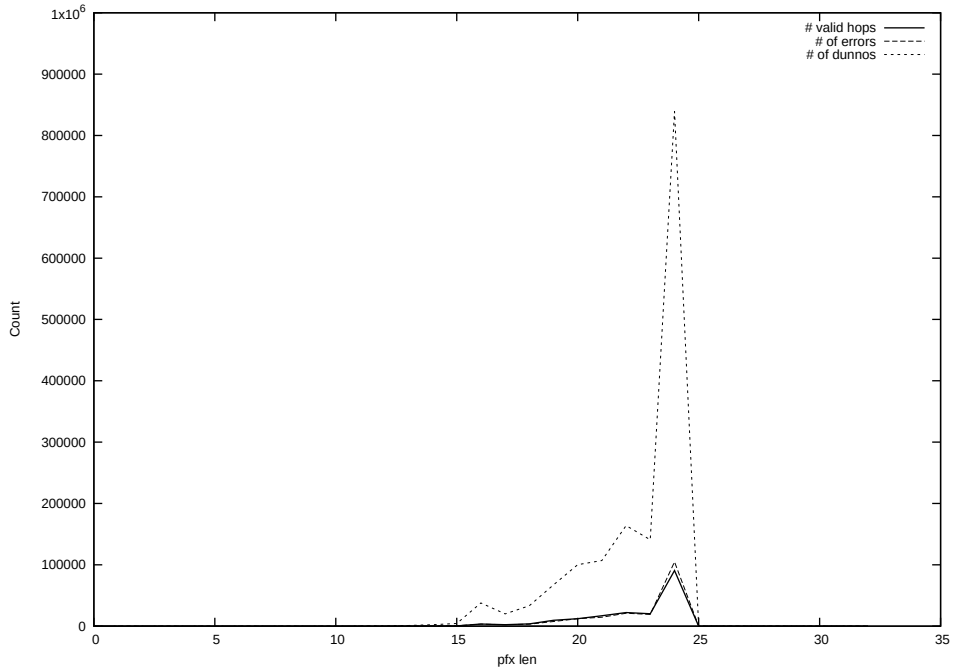
2014-09-09



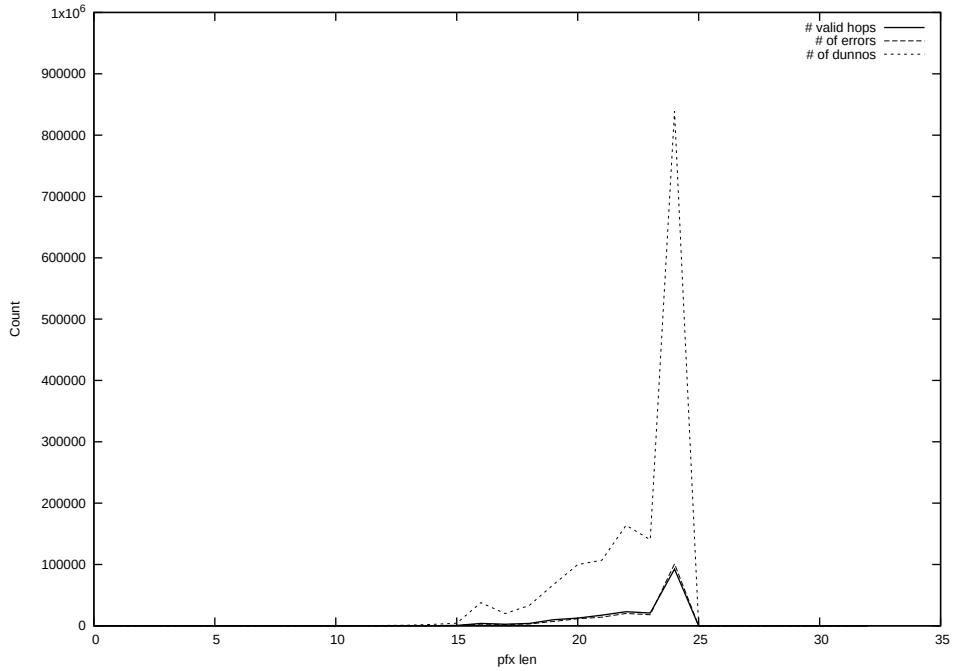
2014-09-10



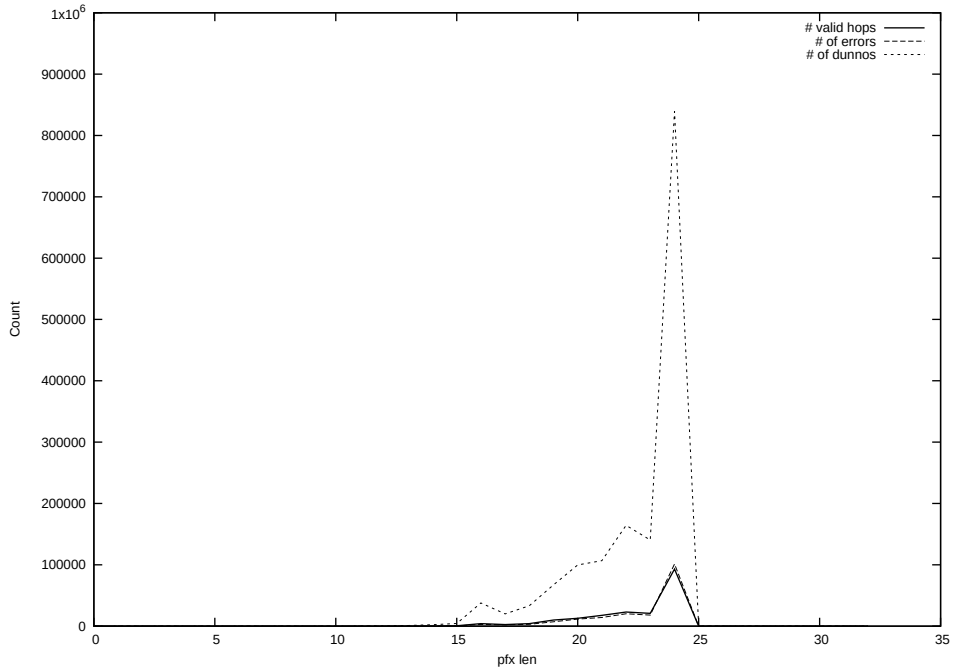
2014-09-11



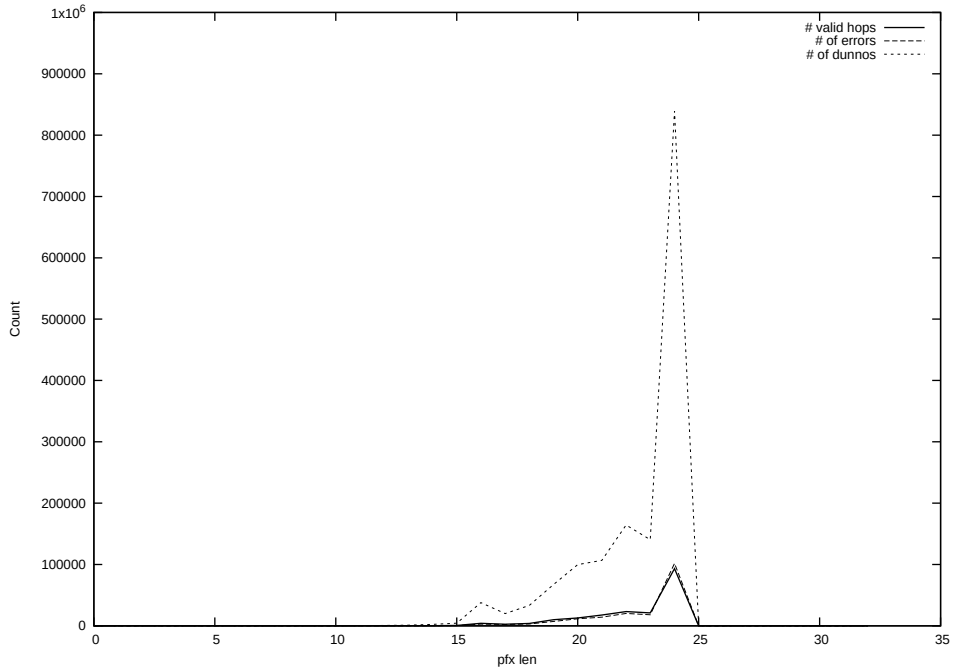
2014-09-12



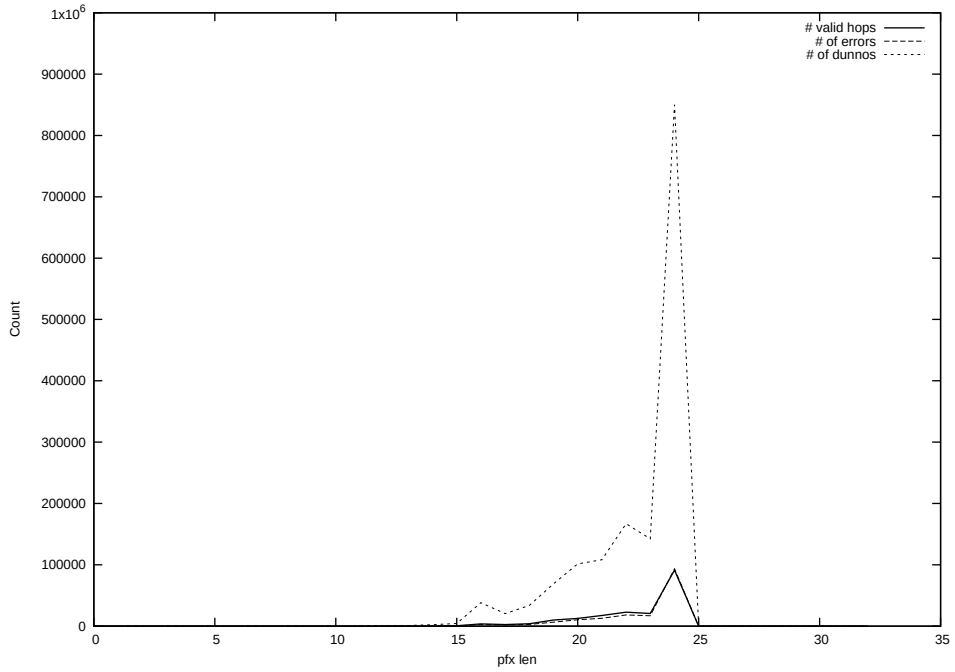
2014-09-13



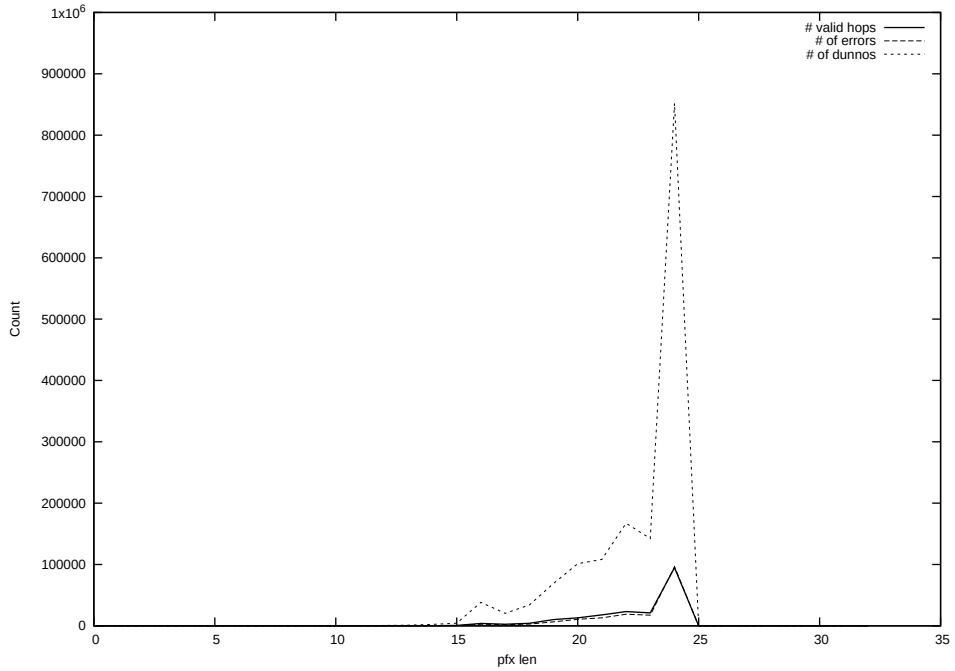
2014-09-14



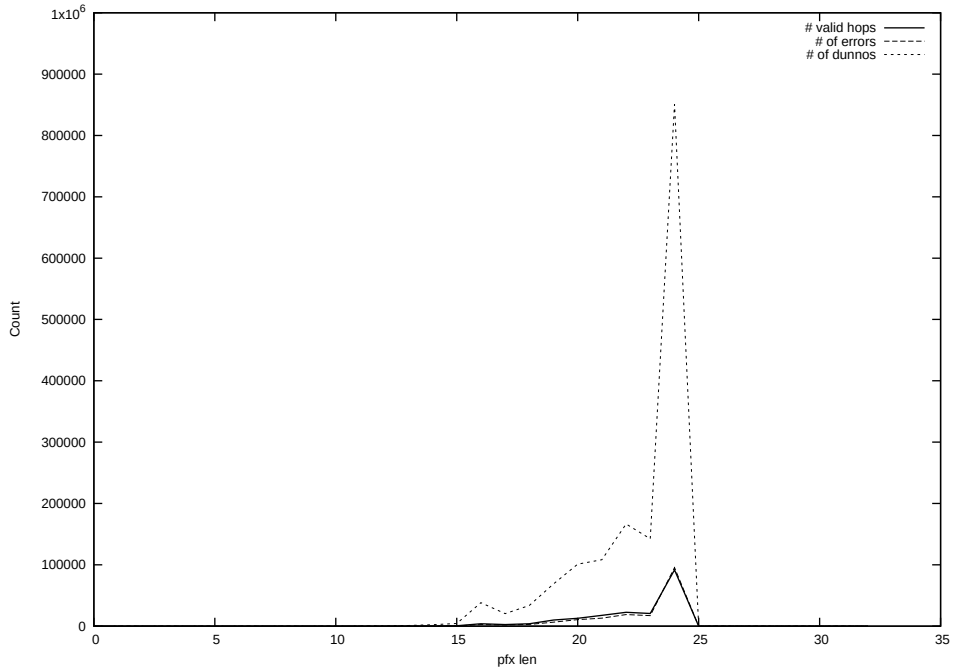
2014-09-15



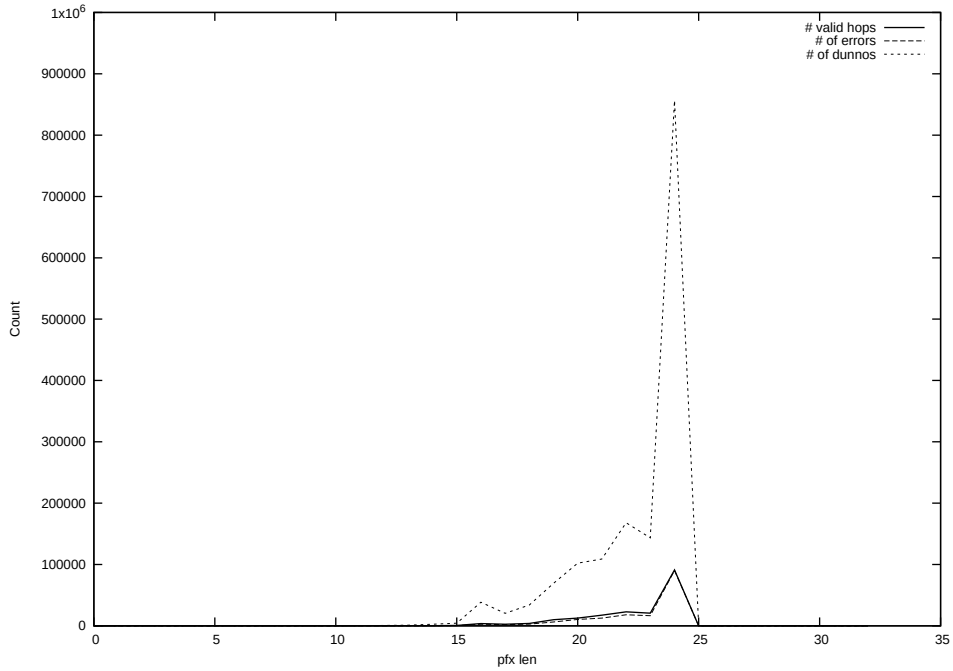
2014-09-16



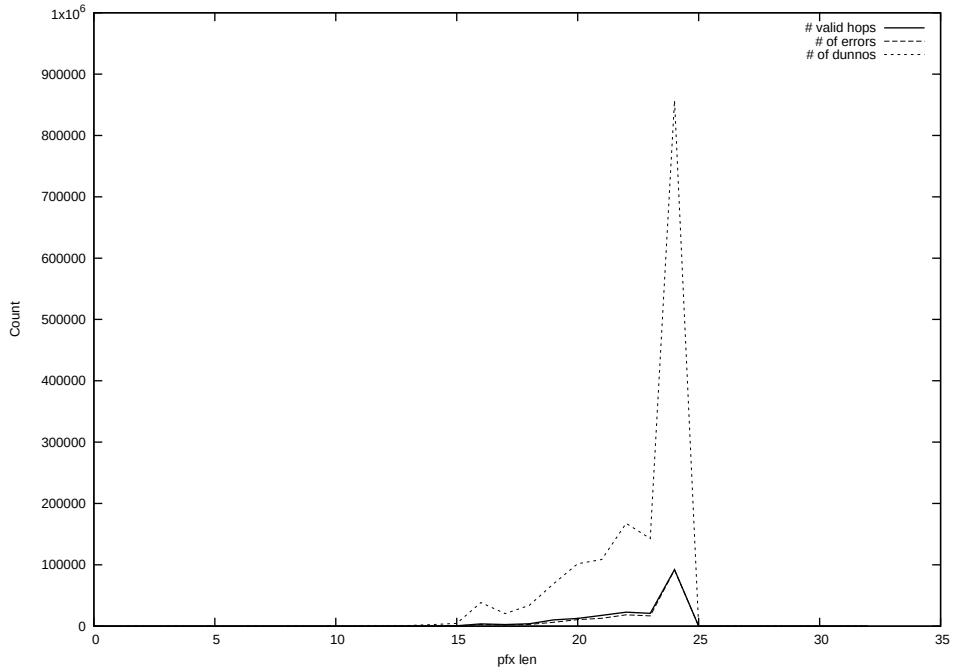
2014-09-17



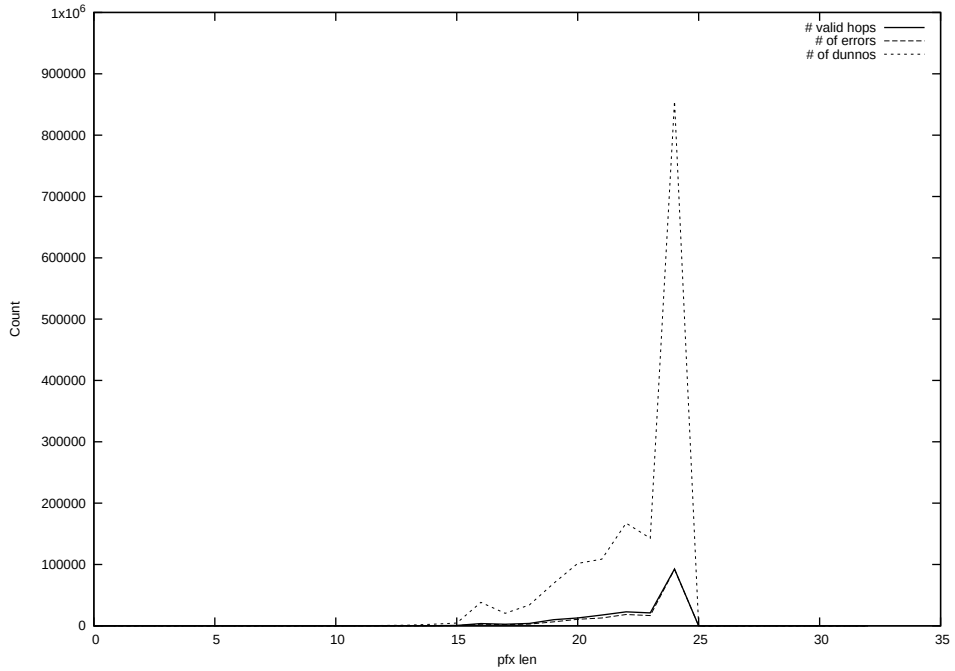
2014-09-18



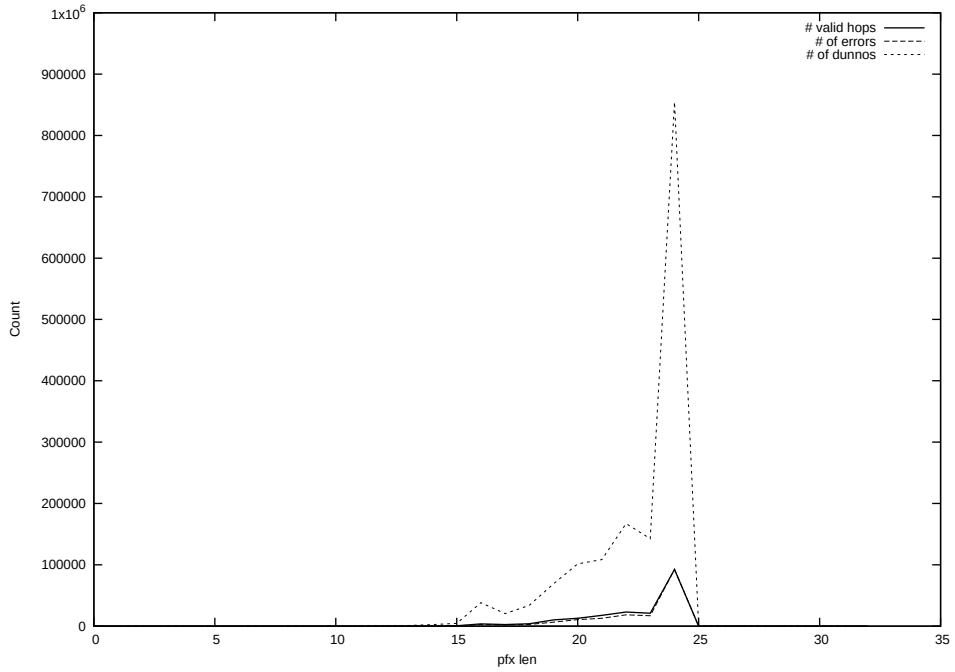
2014-09-19



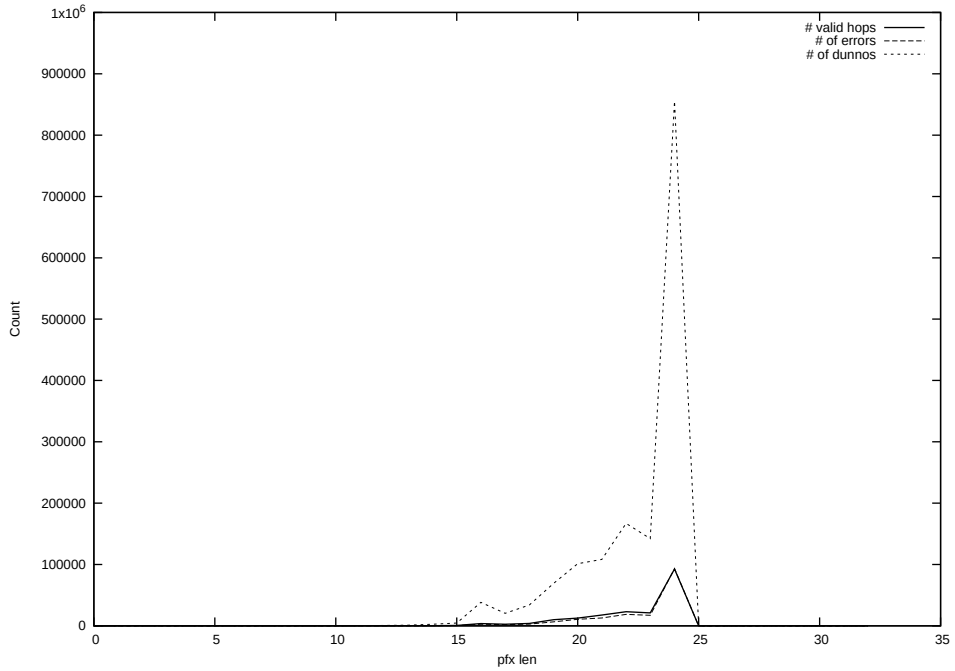
2014-09-20



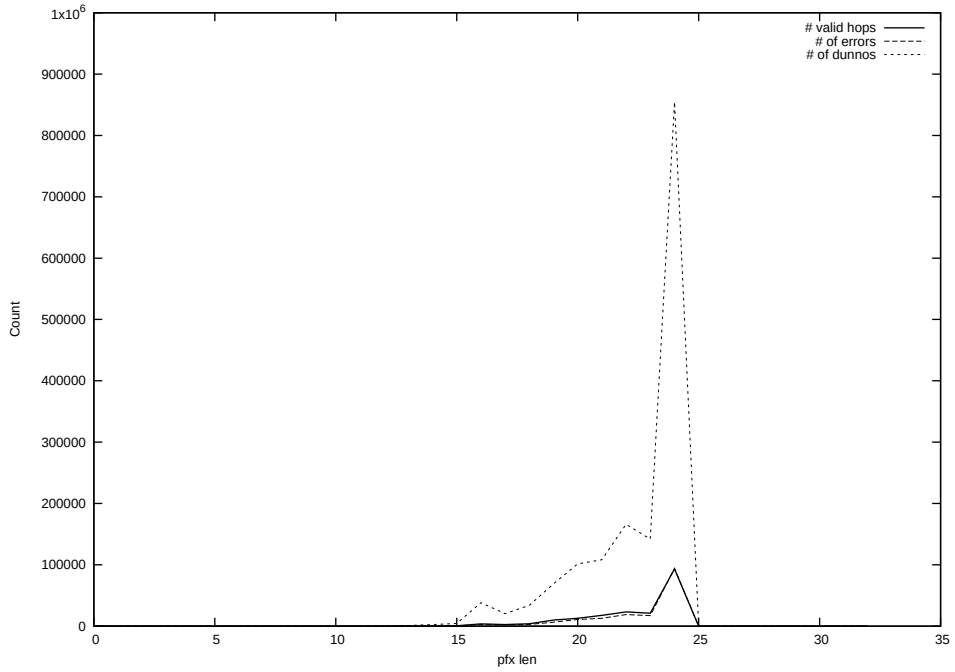
2014-09-21



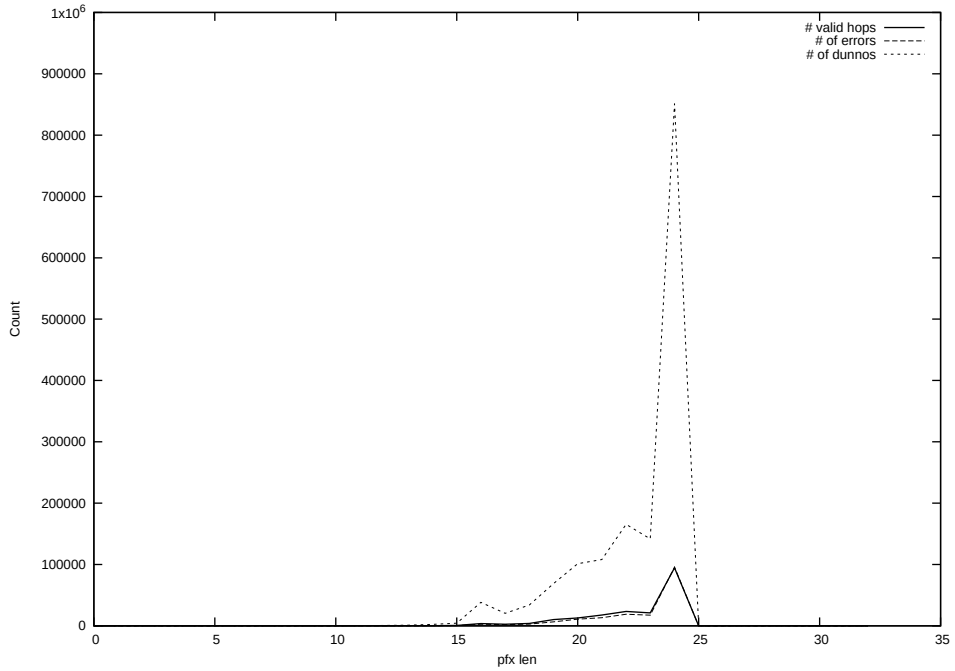
2014-09-22



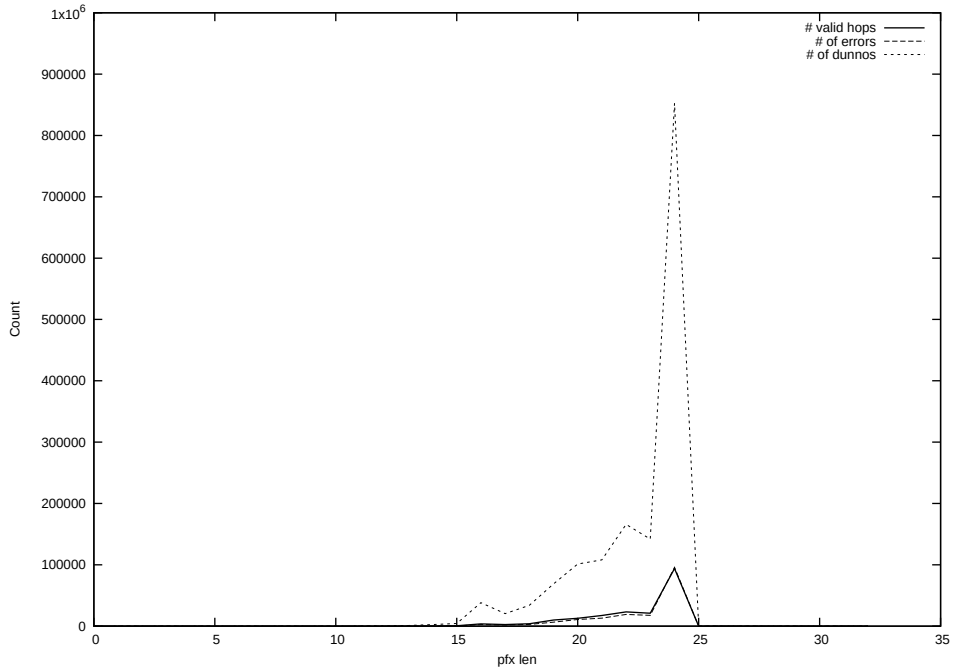
2014-09-23



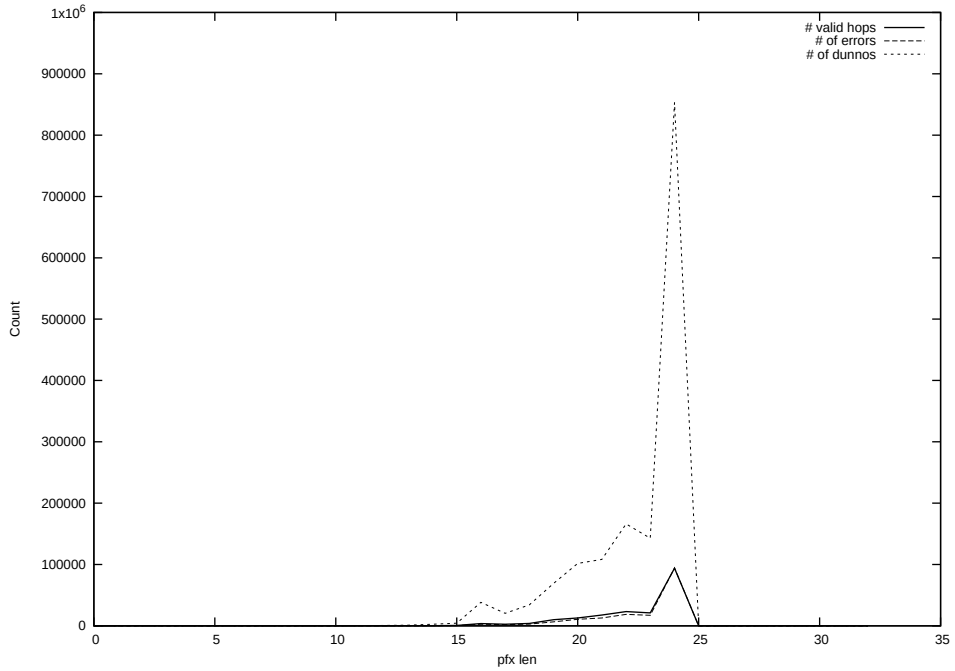
2014-09-24



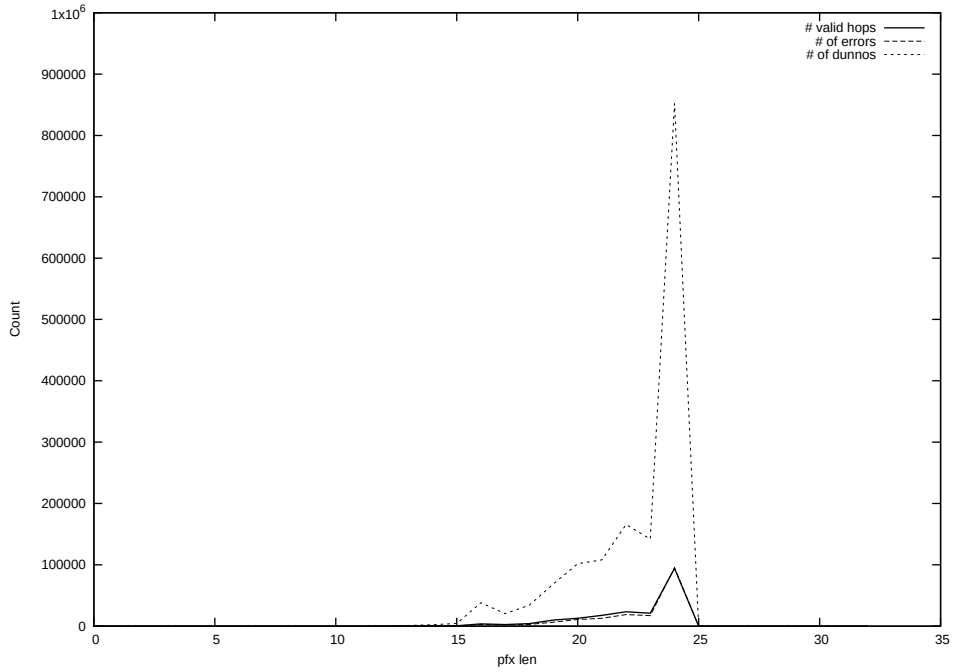
2014-09-25



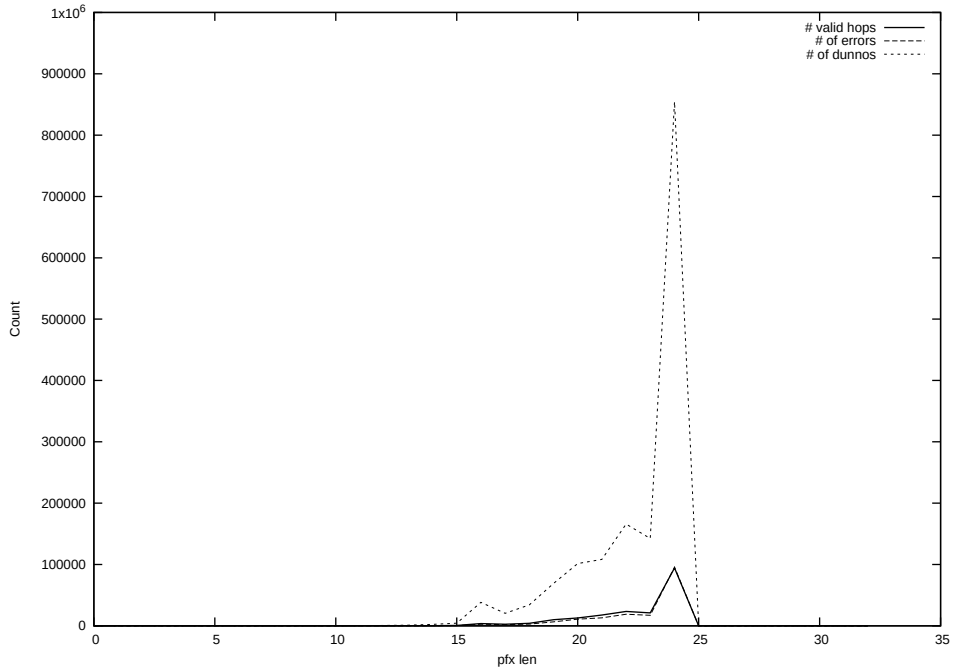
2014-09-26



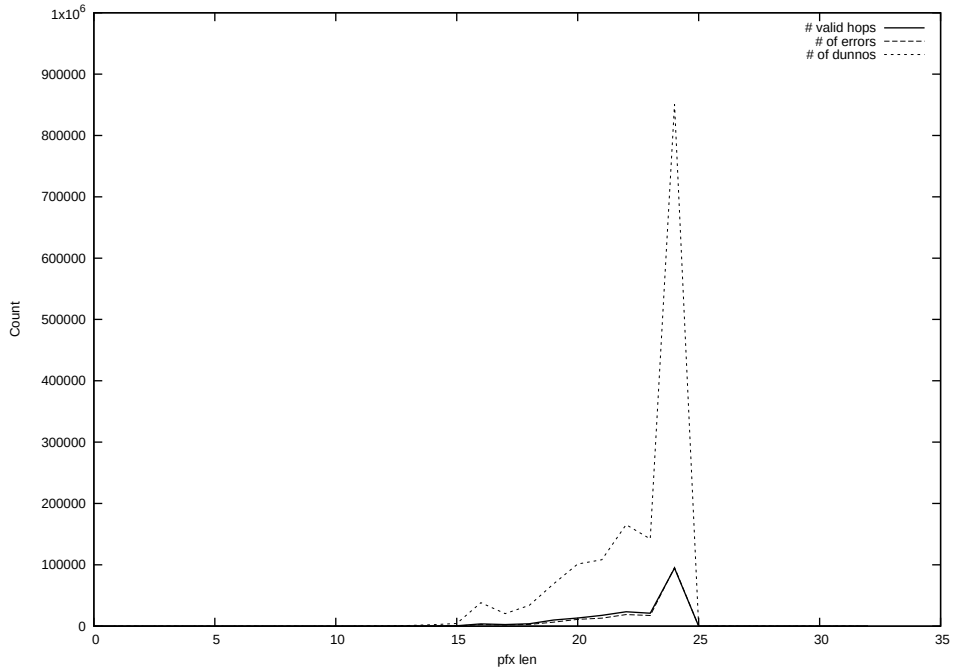
2014-09-27



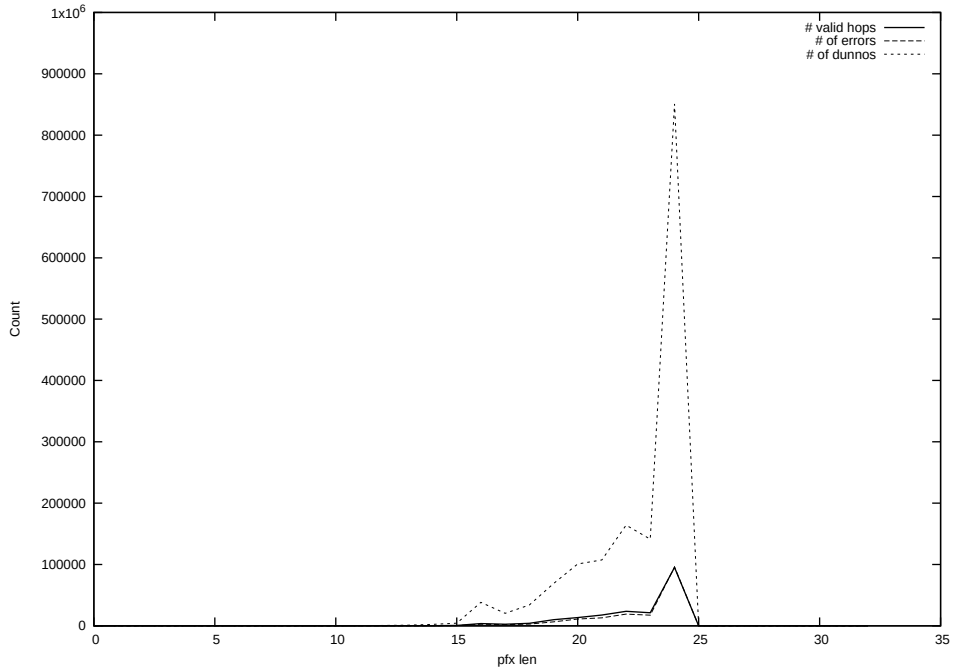
2014-09-28



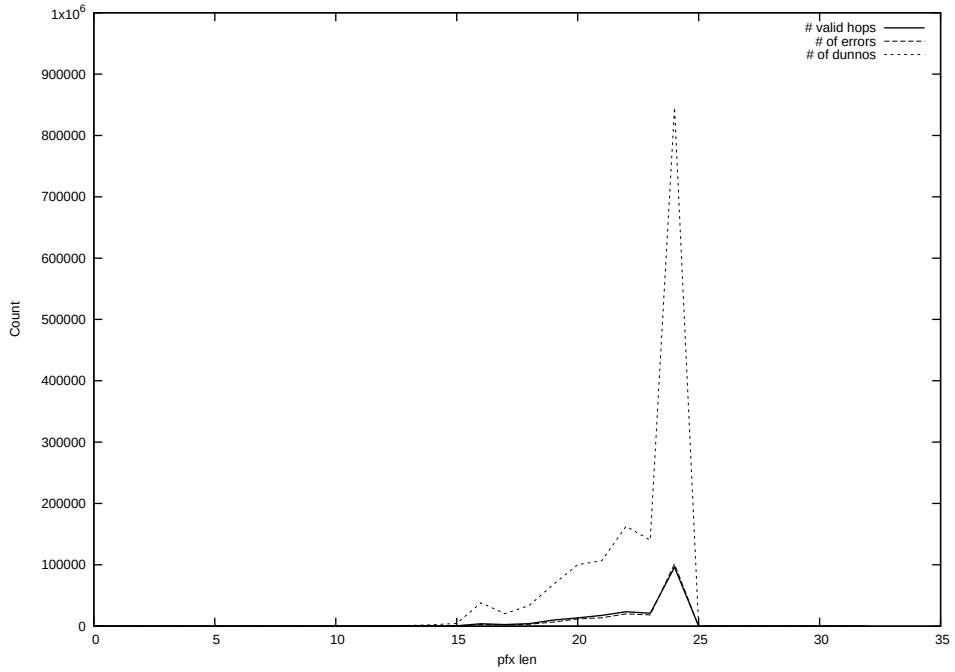
2014-09-29



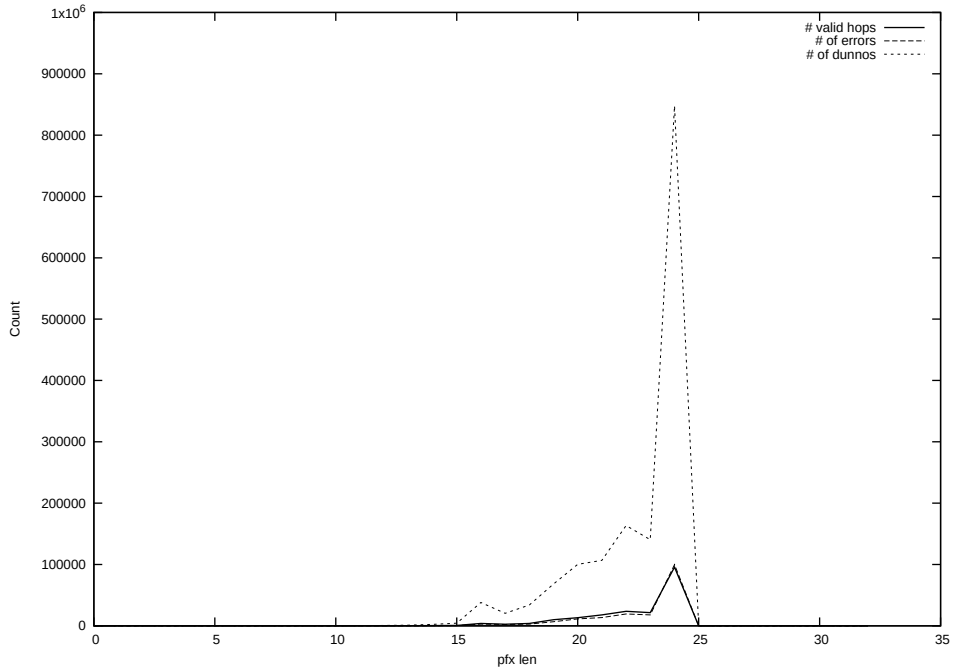
2014-09-30



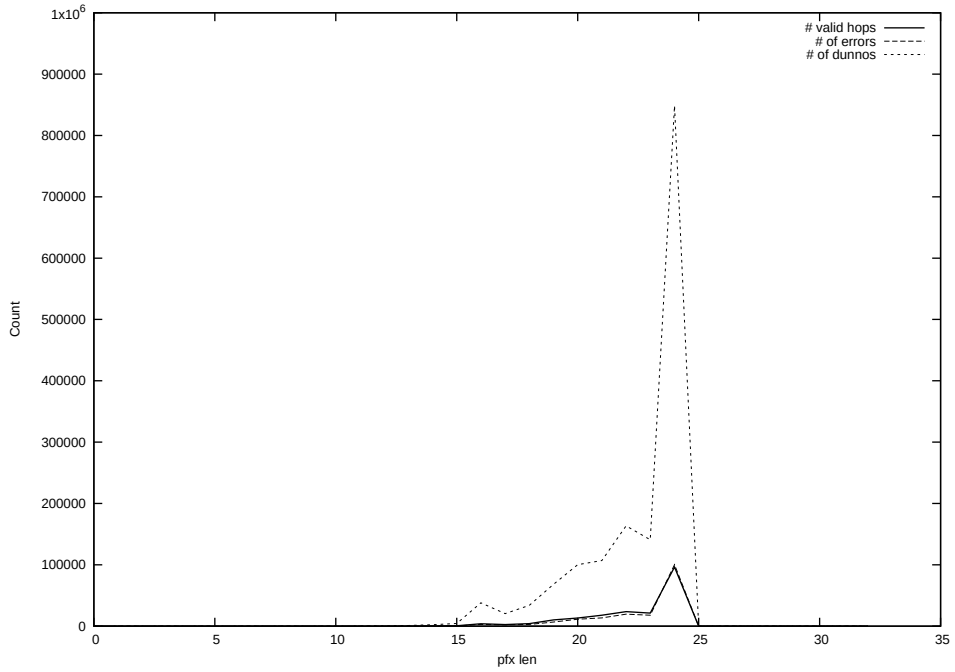
2014-10-01



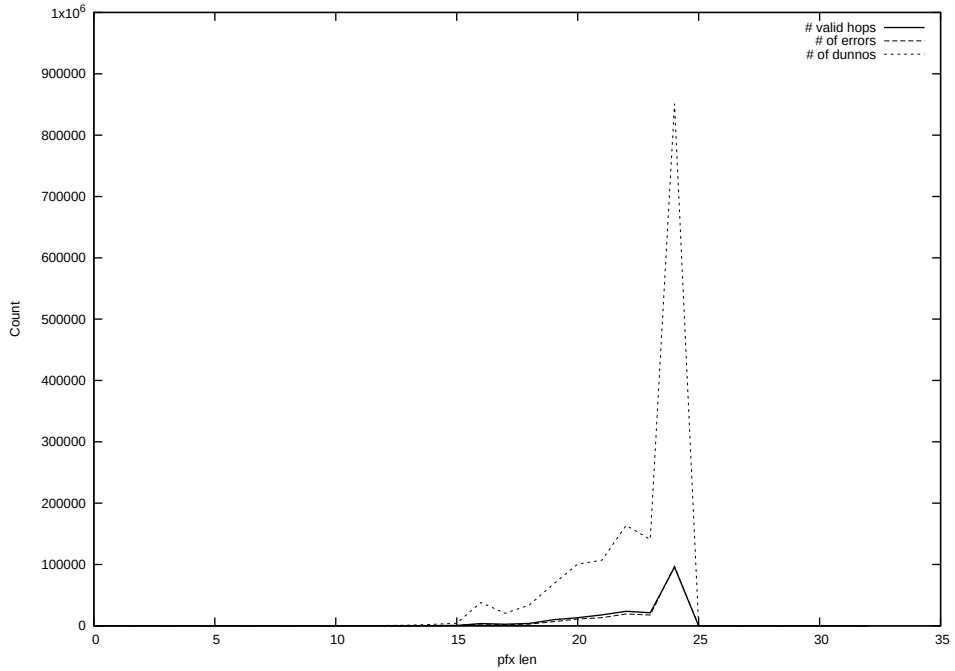
2014-10-02



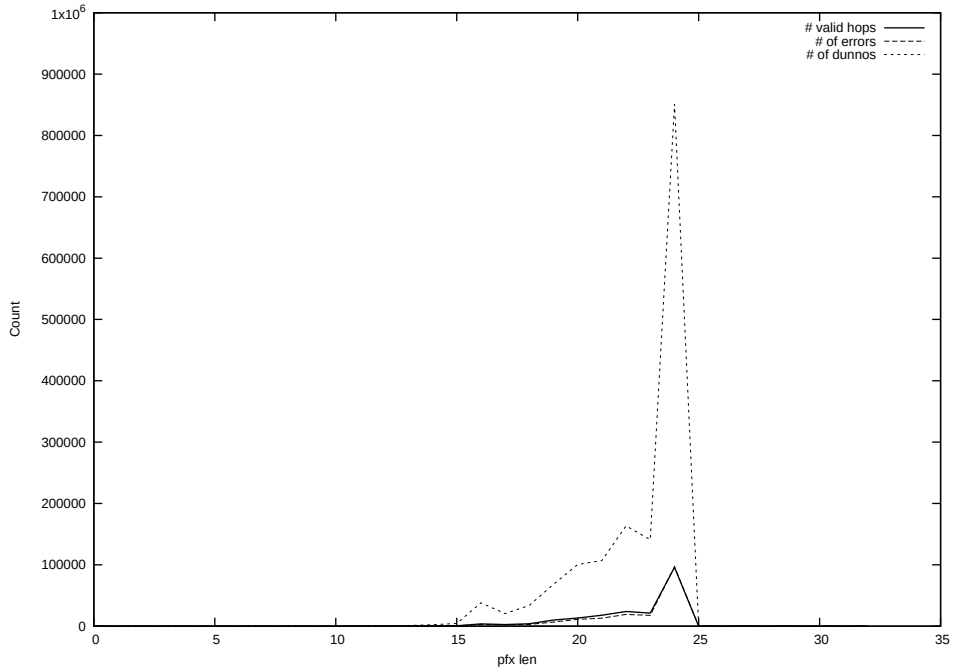
2014-10-03



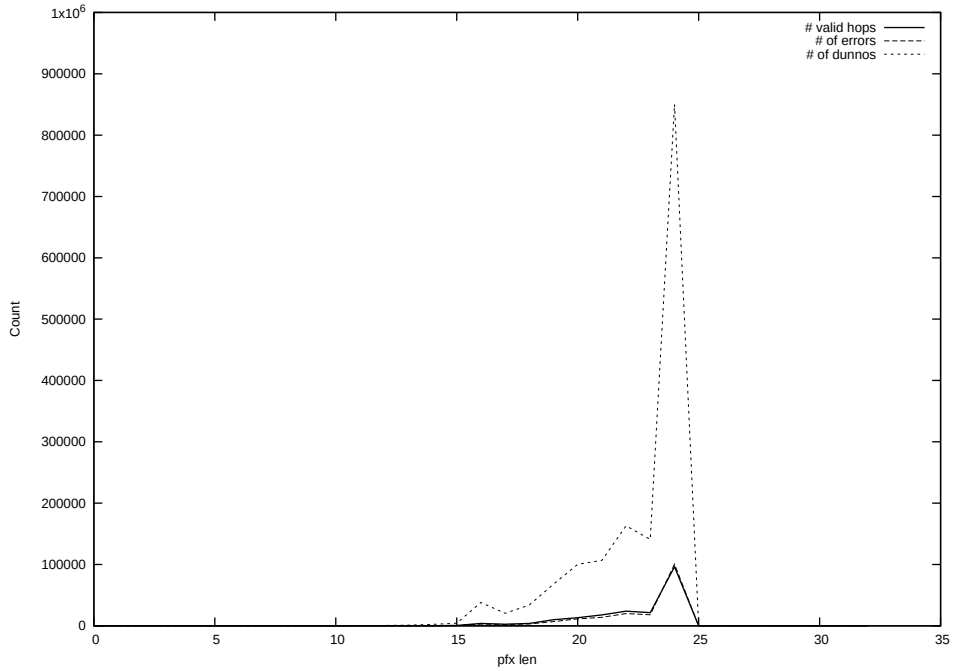
2014-10-04



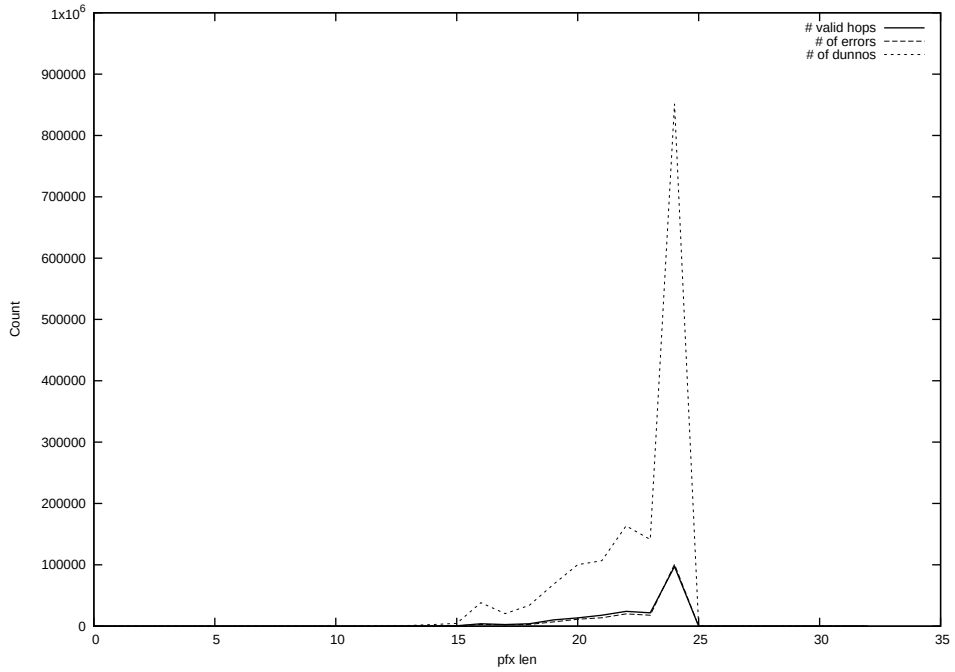
2014-10-05



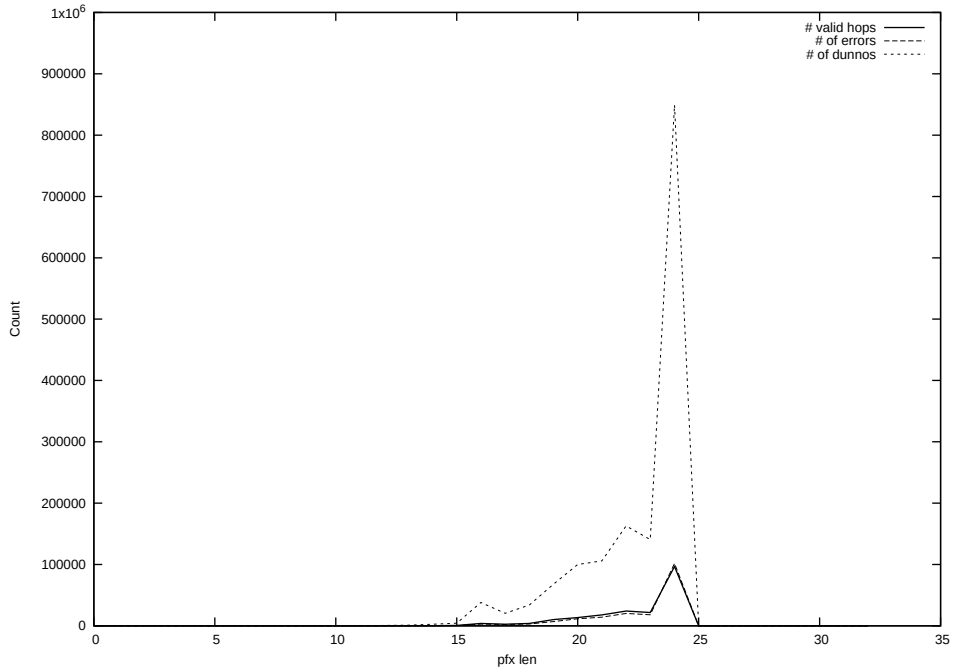
2014-10-06



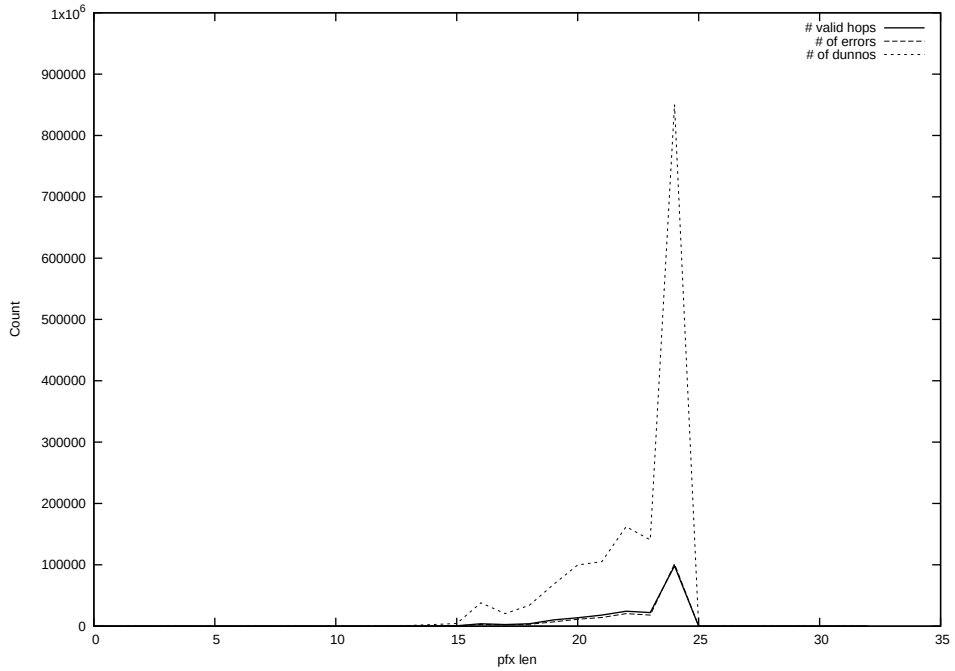
2014-10-07



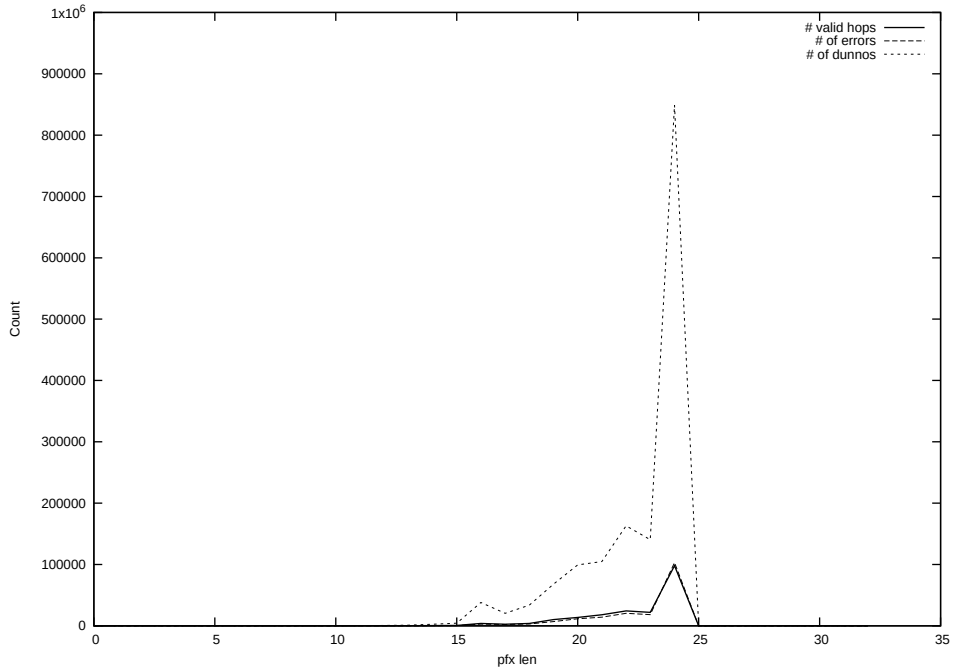
2014-10-08



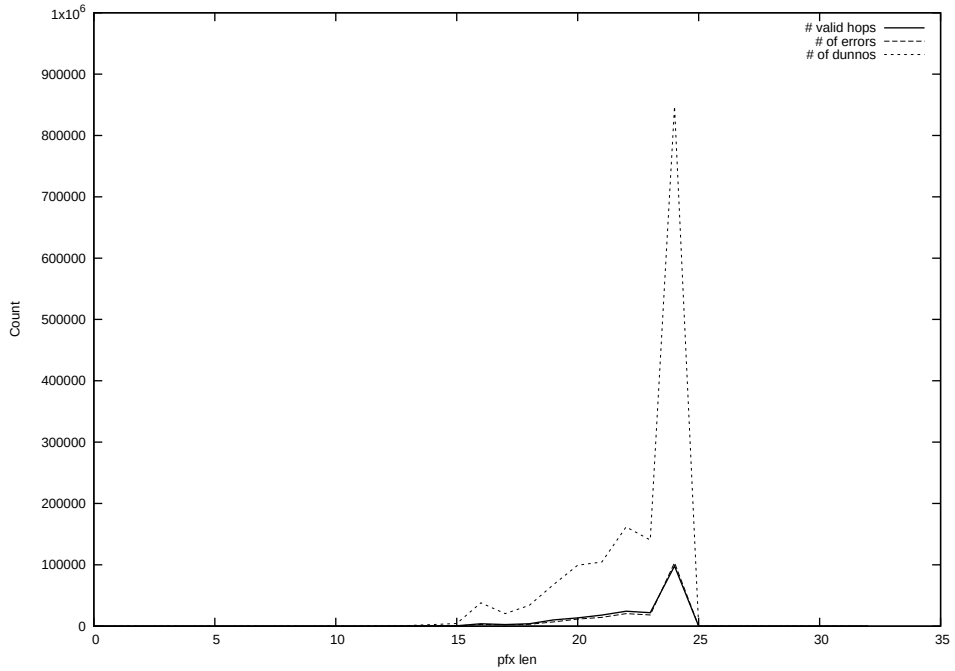
2014-10-09



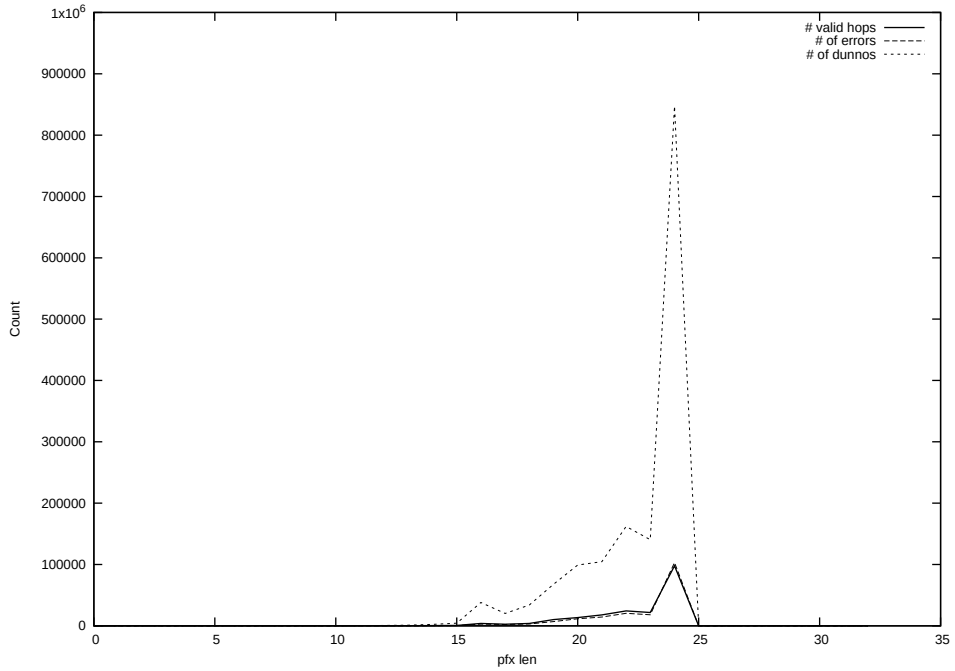
2014-10-10



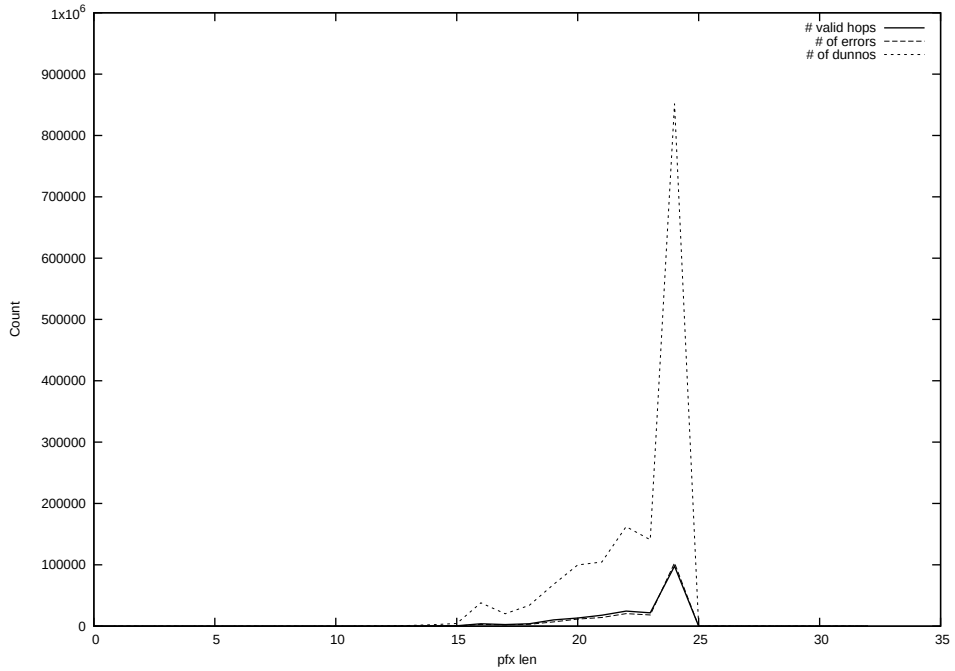
2014-10-11



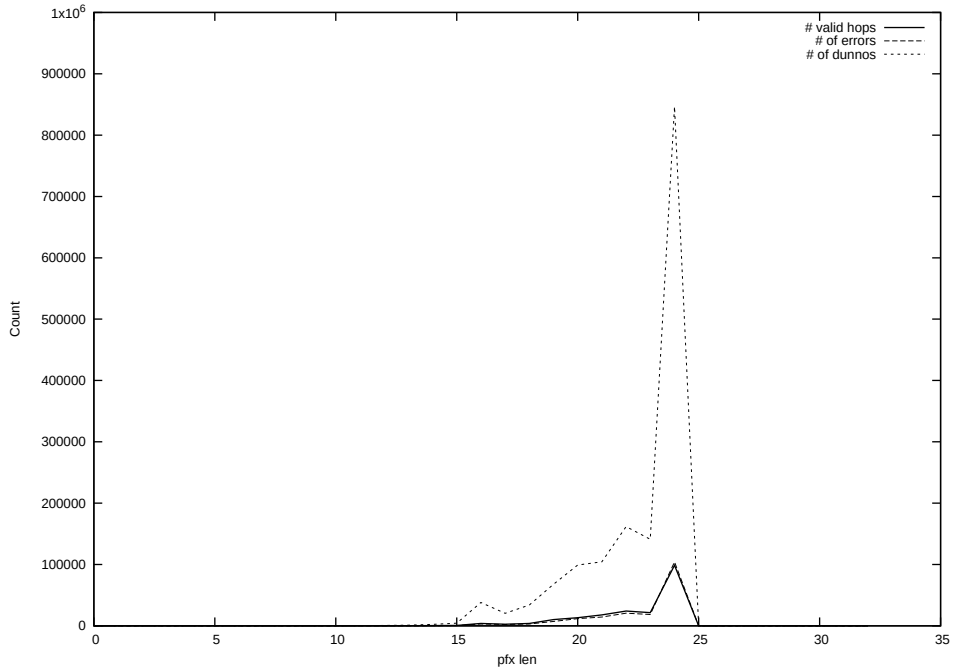
2014-10-12



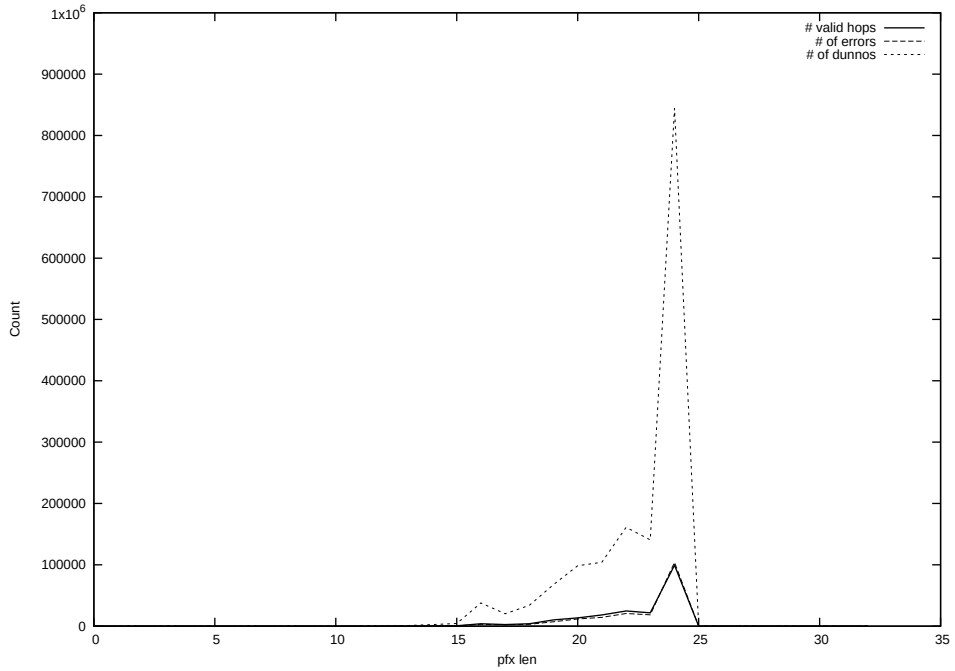
2014-10-13



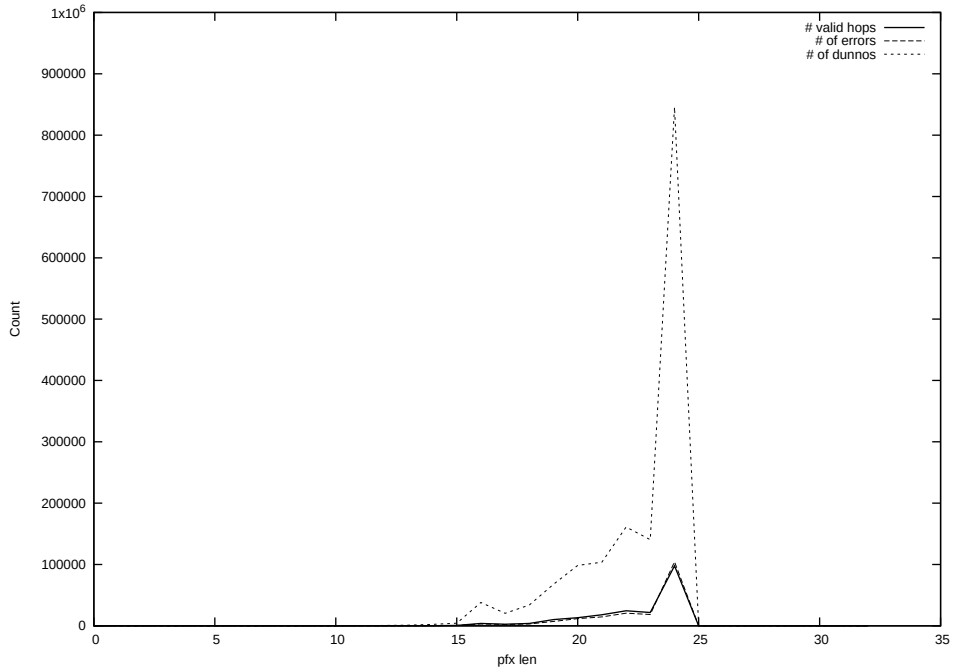
2014-10-14



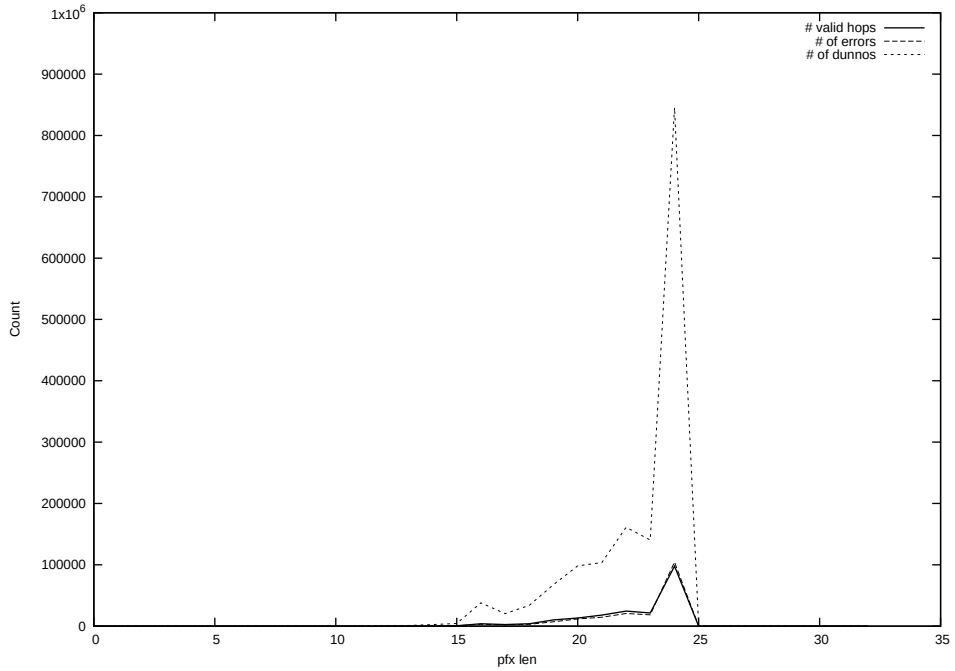
2014-10-15



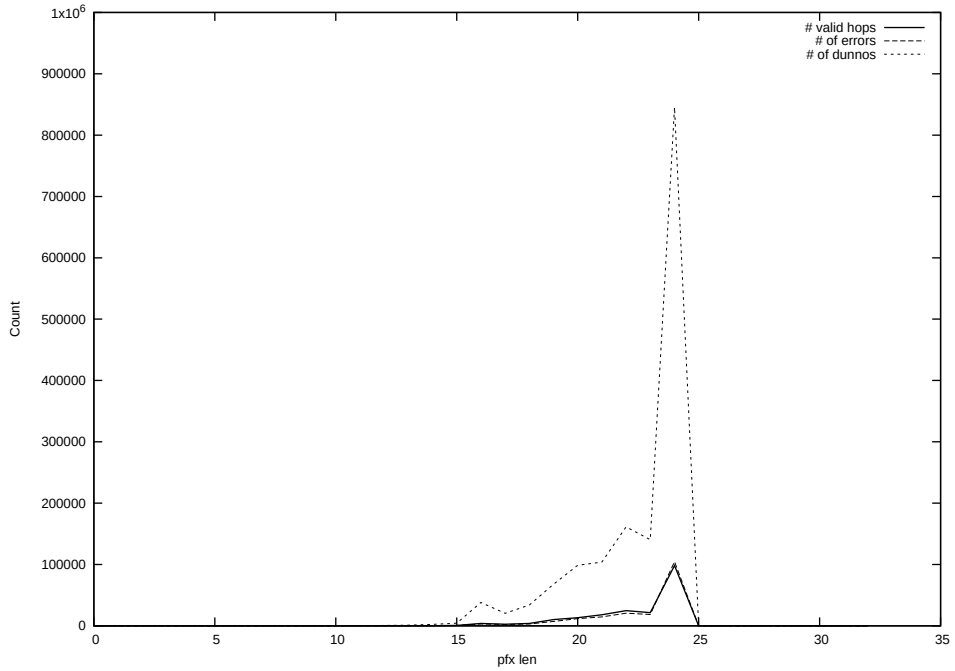
2014-10-16



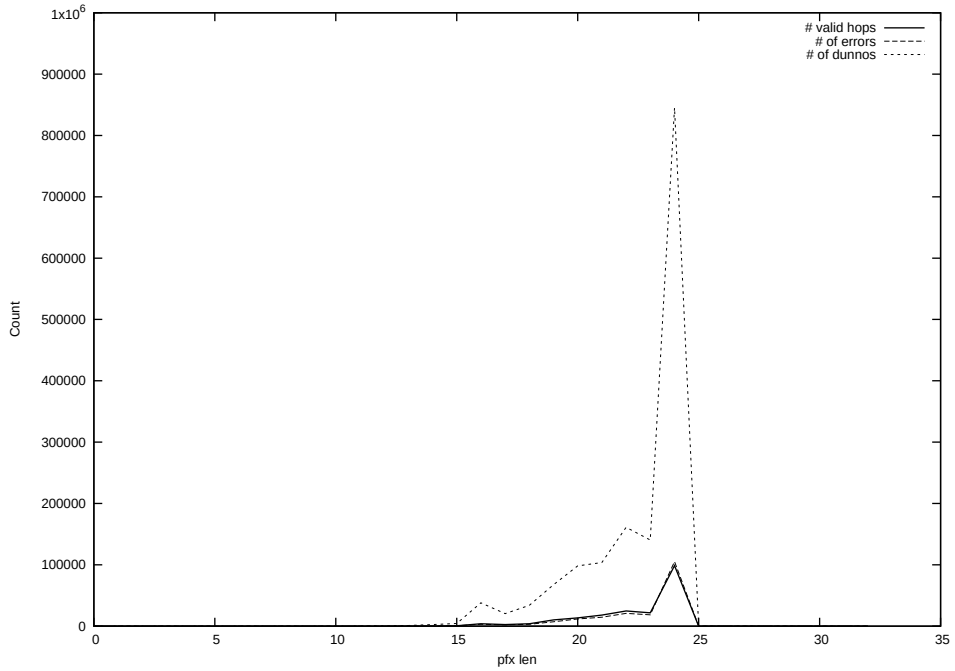
2014-10-17



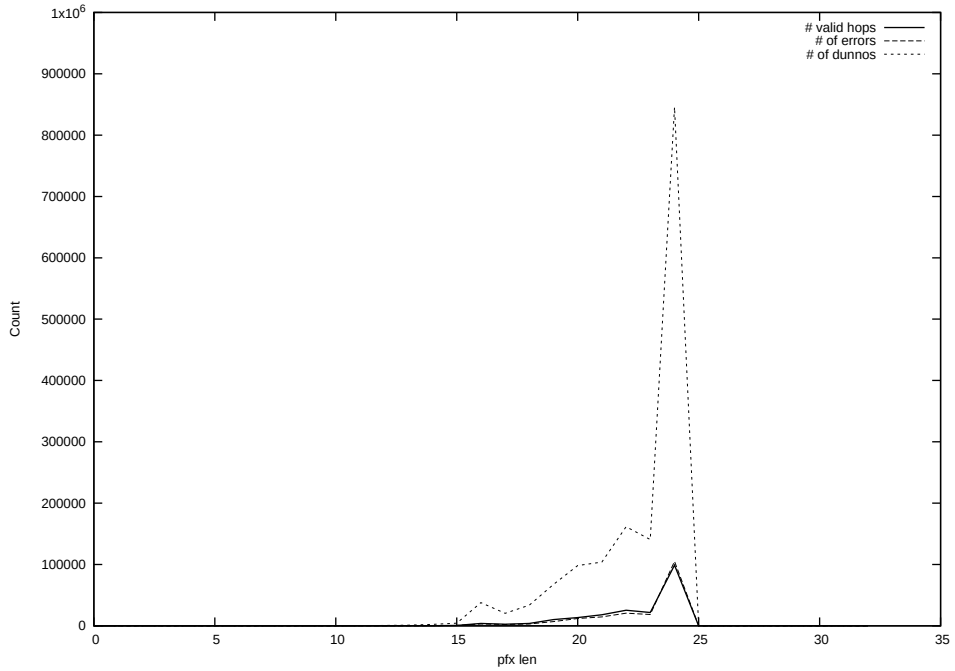
2014-10-18



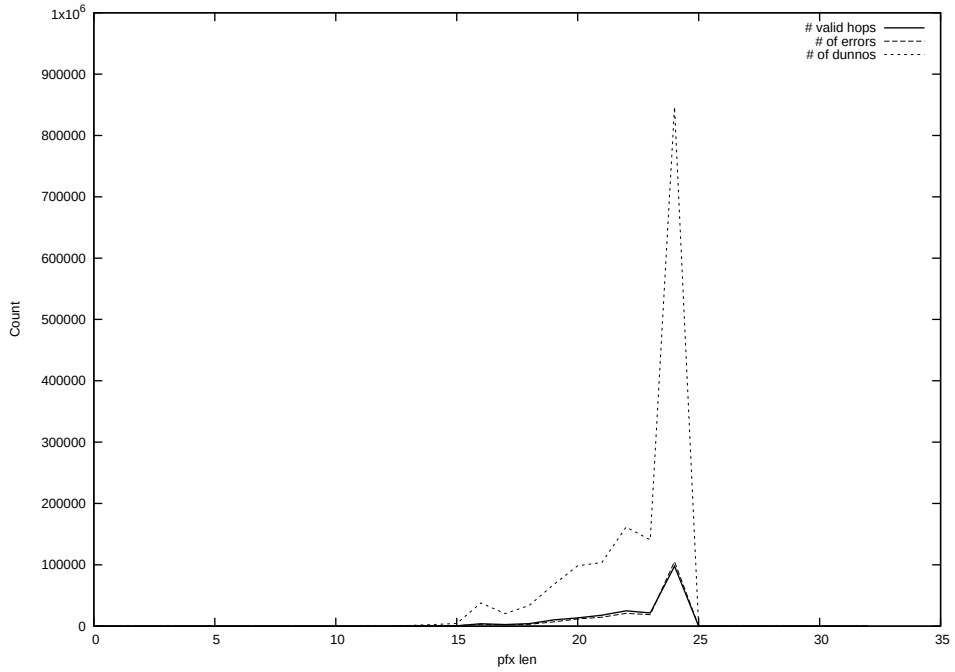
2014-10-19



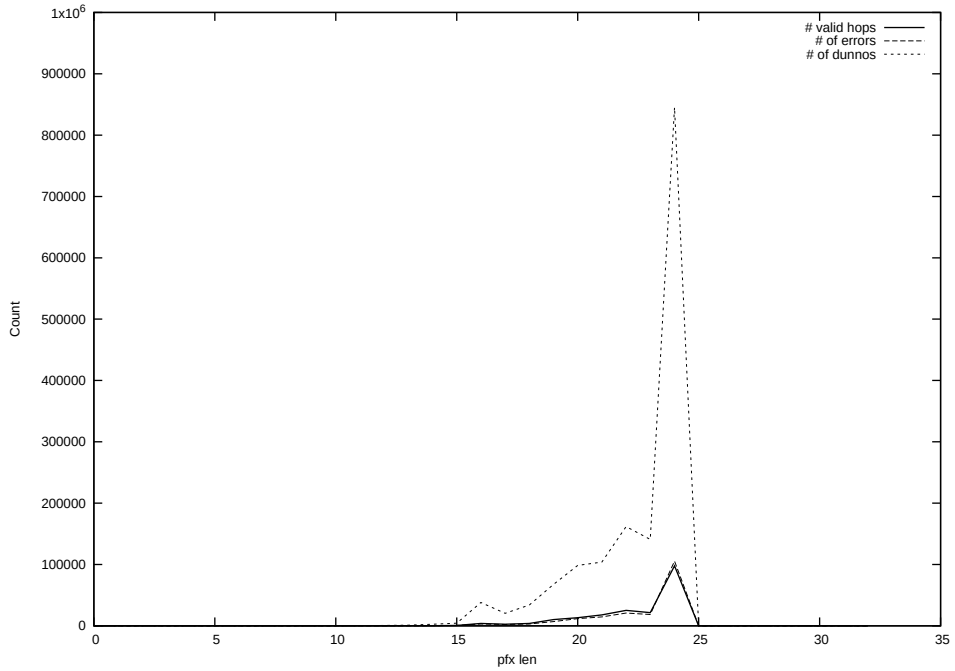
2014-10-20



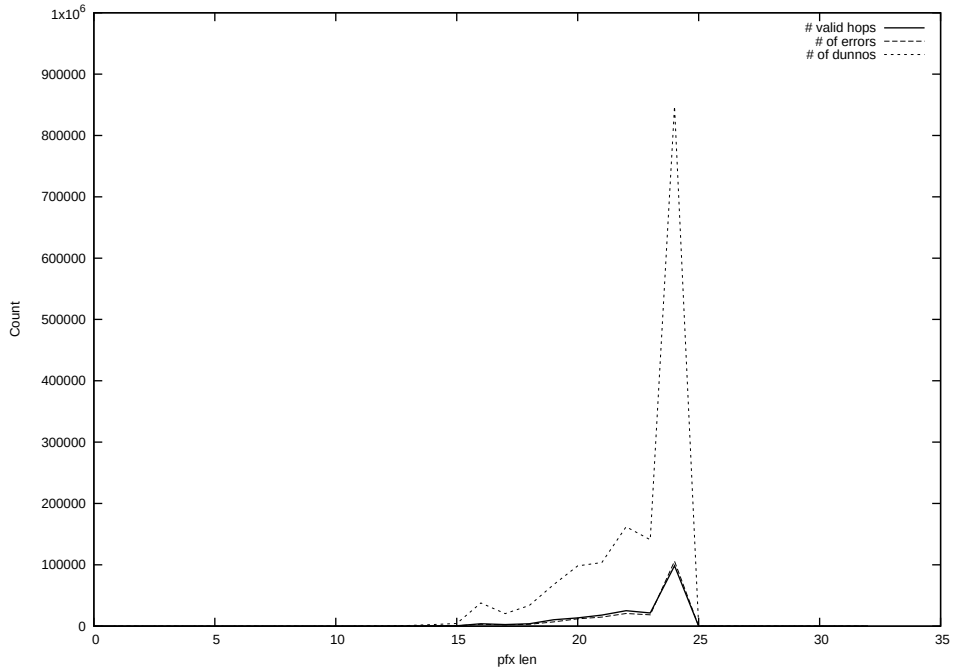
2014-10-21



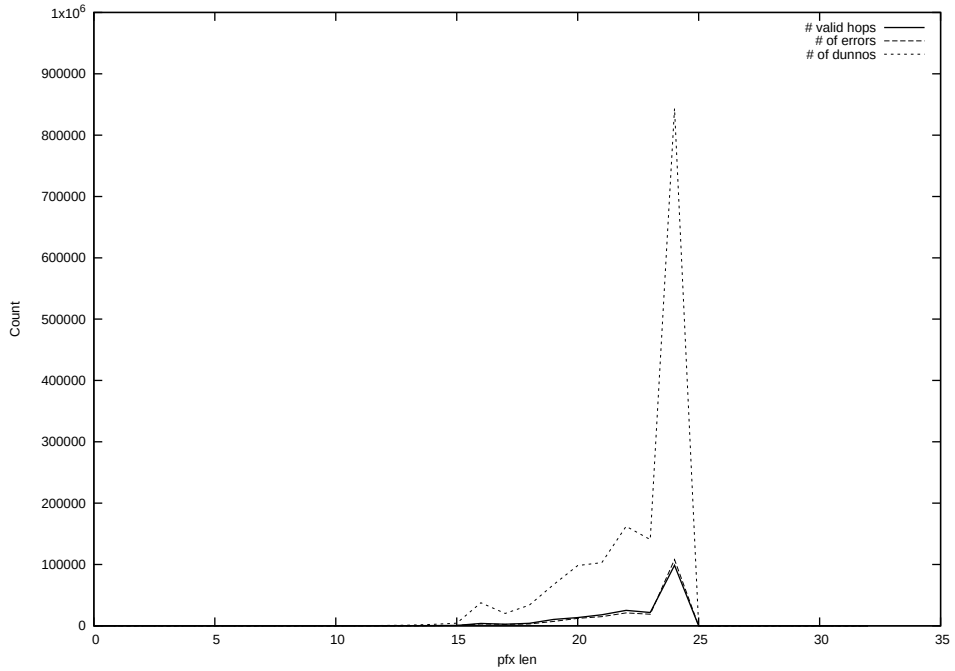
2014-10-22



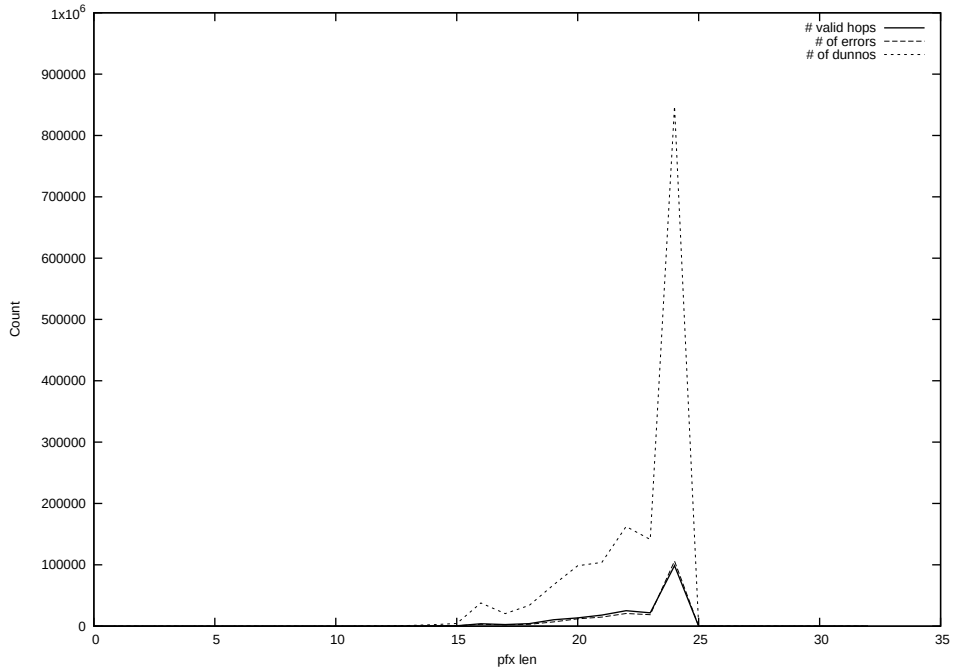
2014-10-23



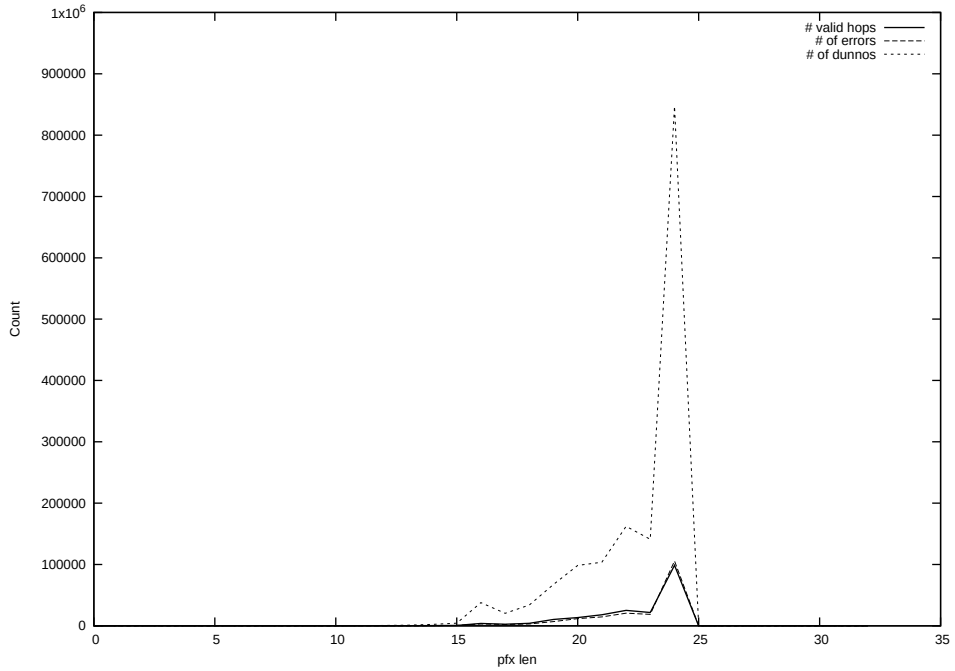
2014-10-24



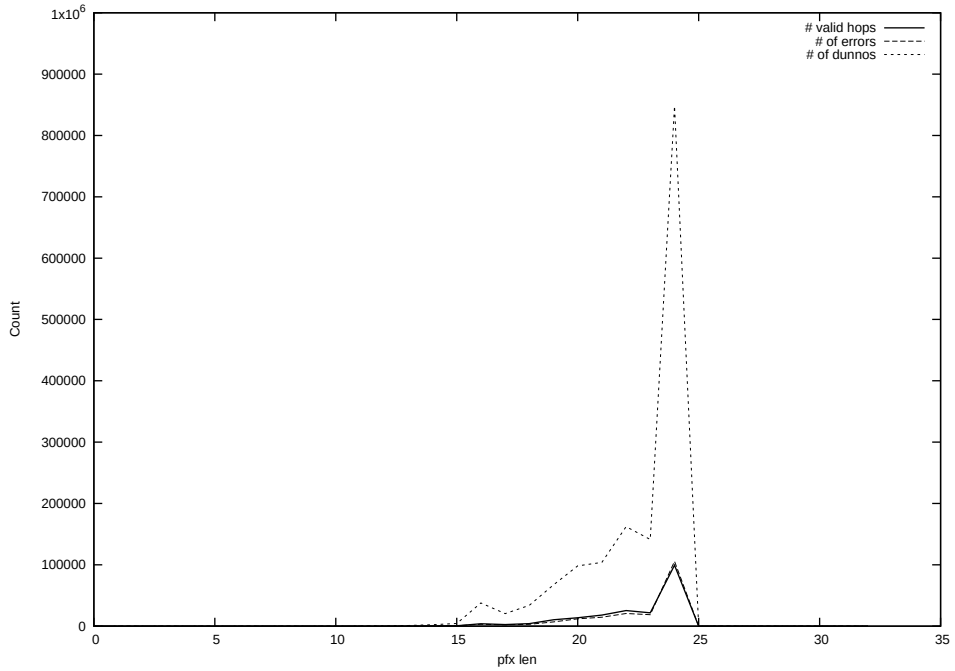
2014-10-25



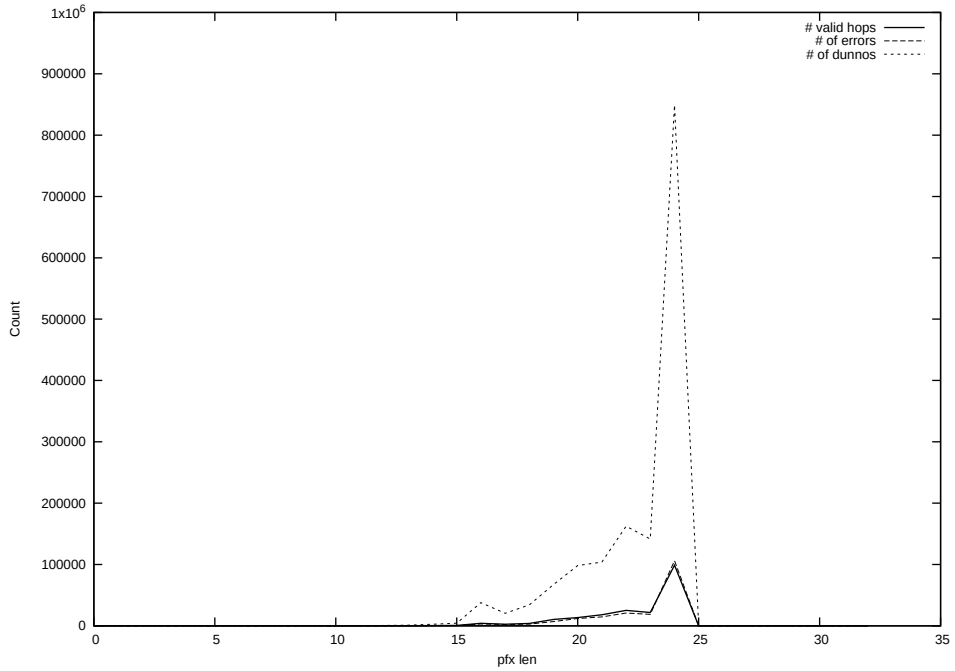
2014-10-26



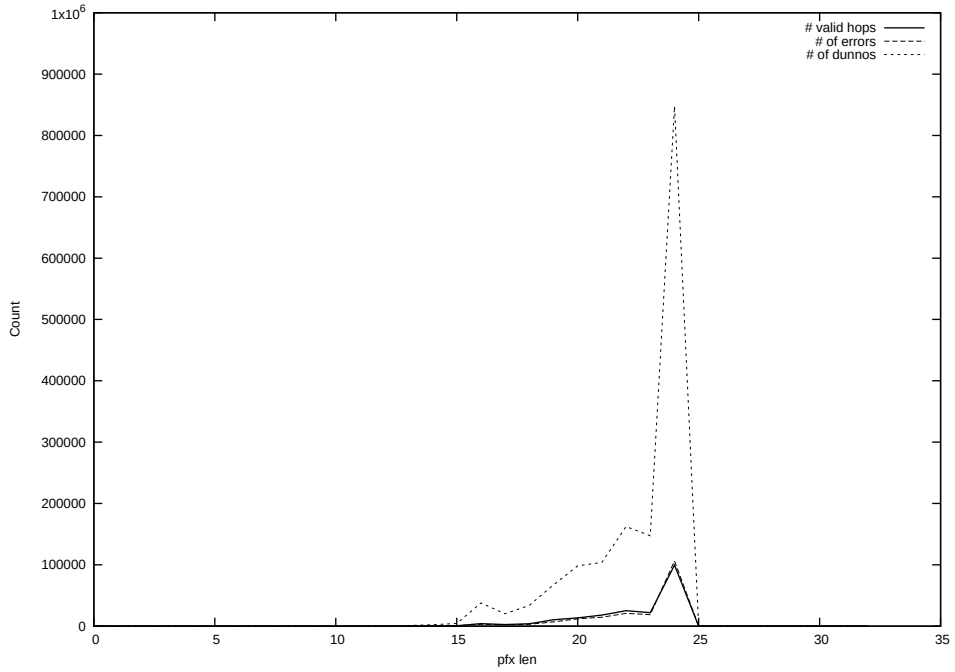
2014-10-27



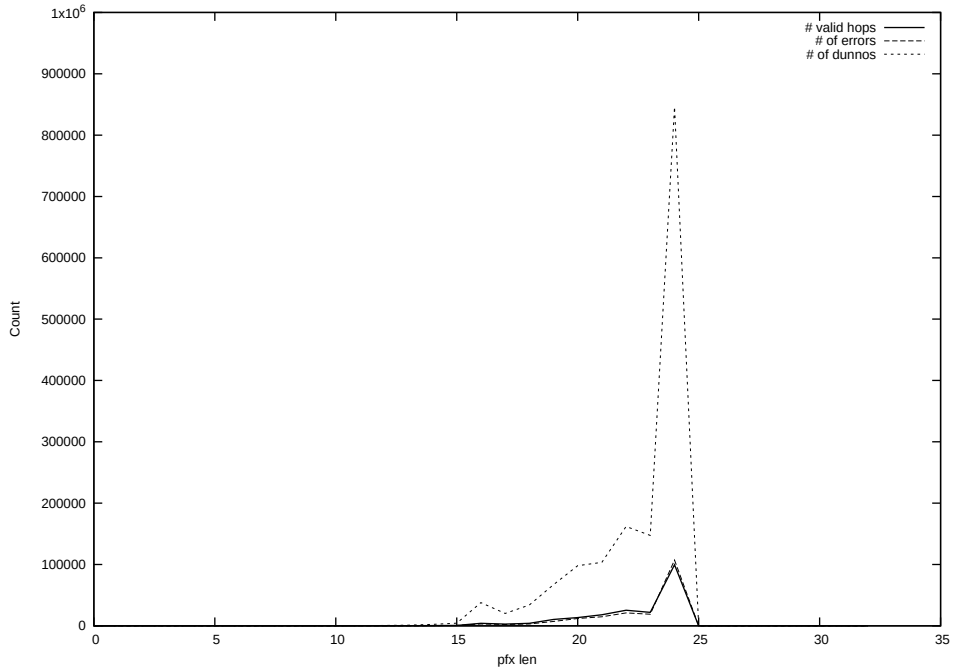
2014-10-28



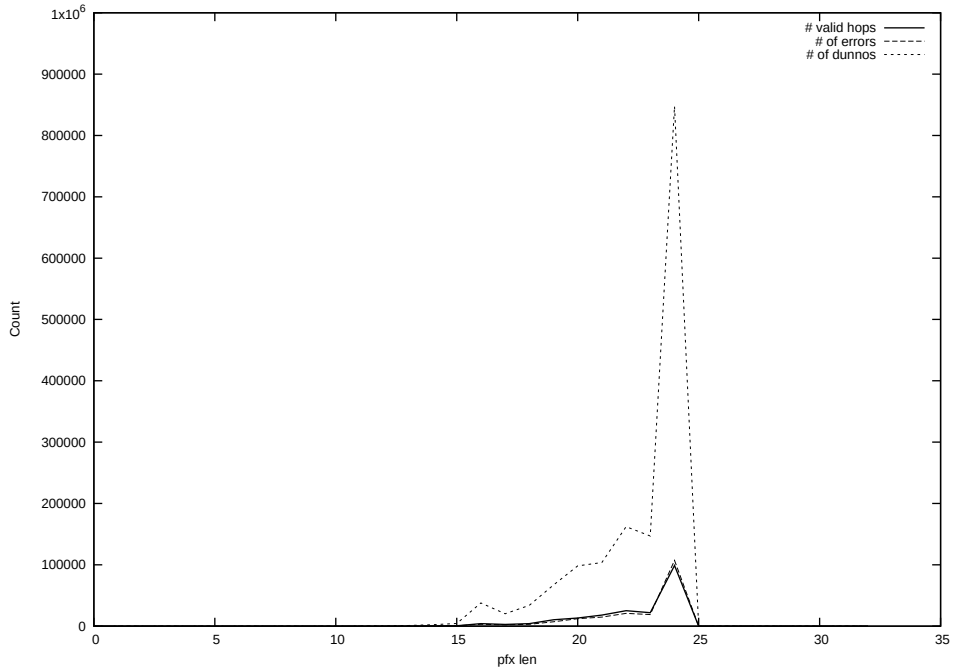
2014-10-29



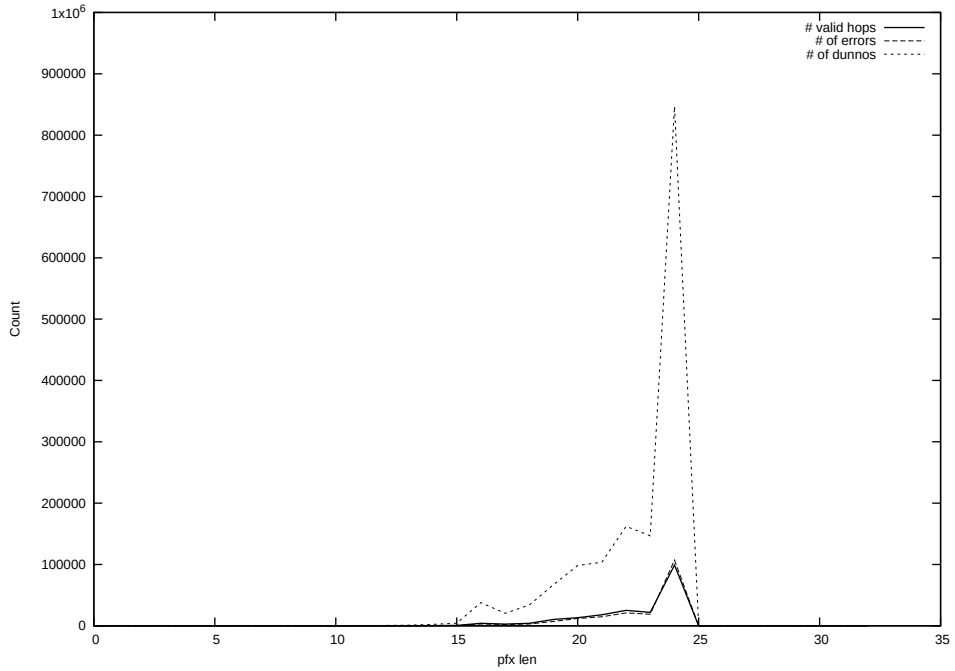
2014-10-30



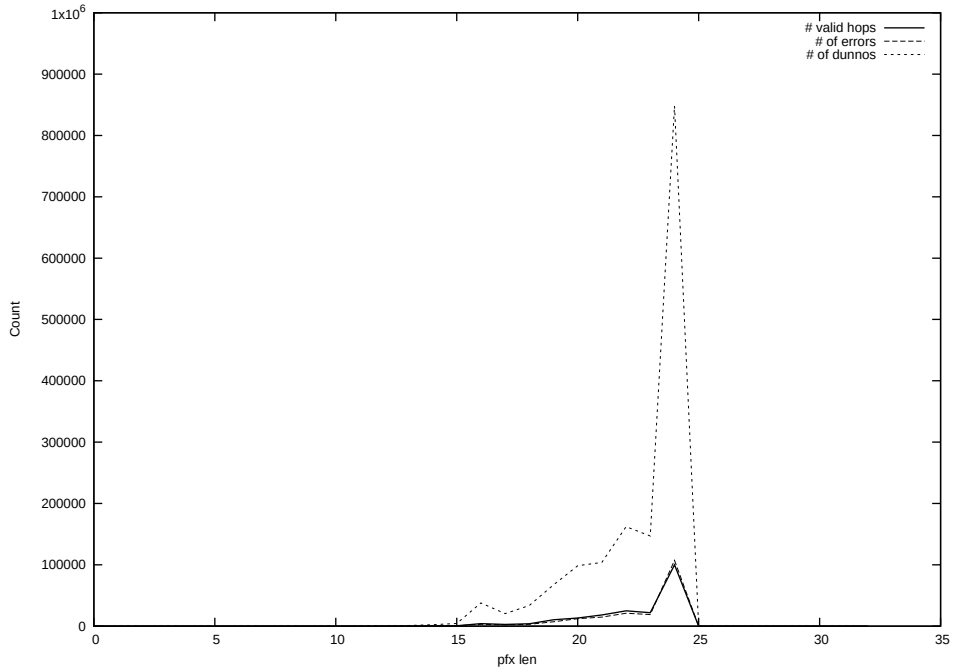
2014-10-31



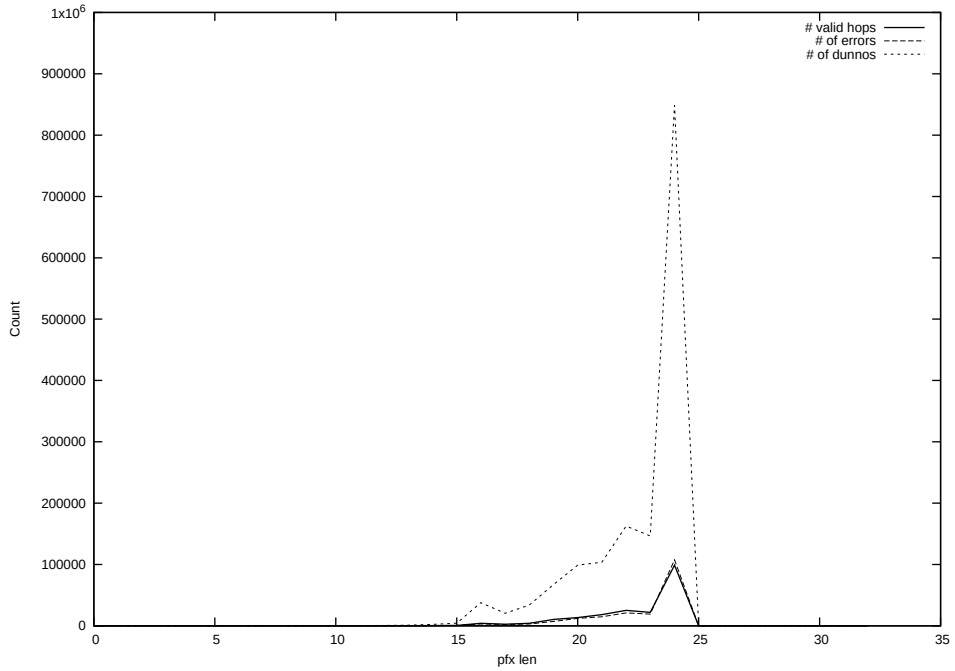
2014-11-02



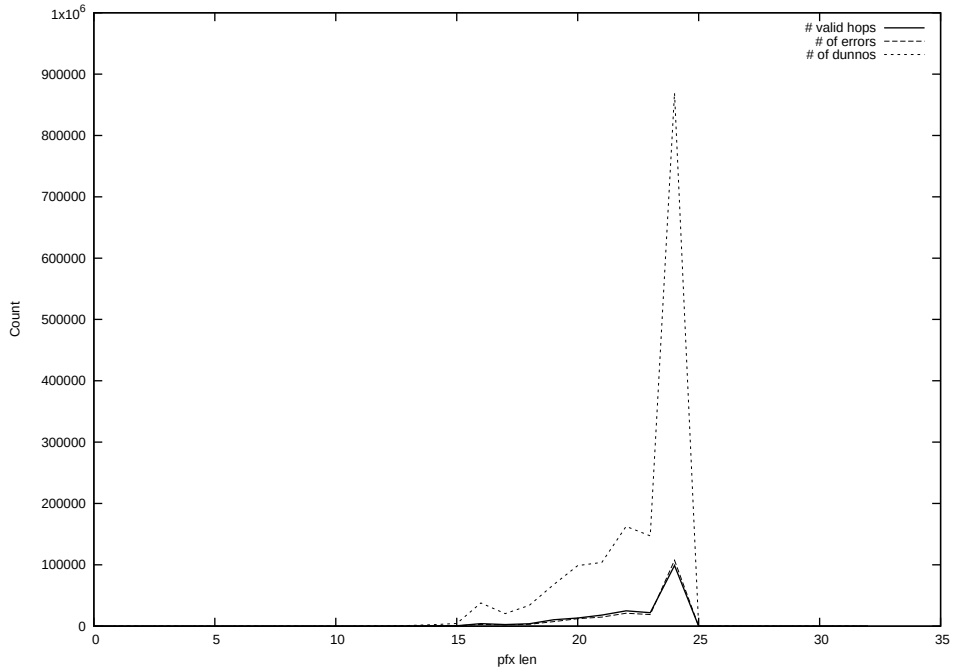
2014-11-03



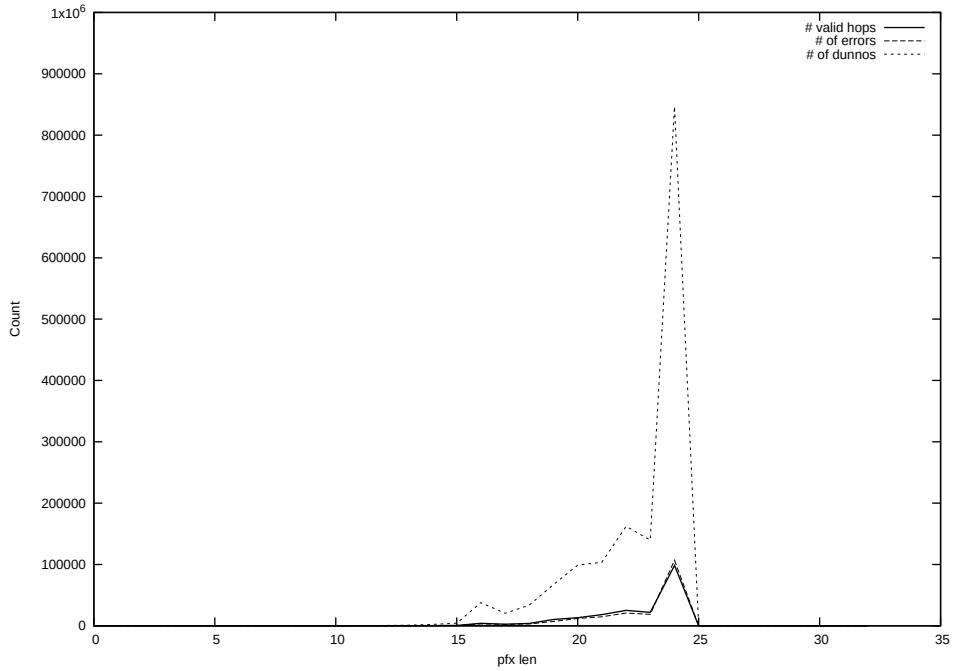
2014-11-04



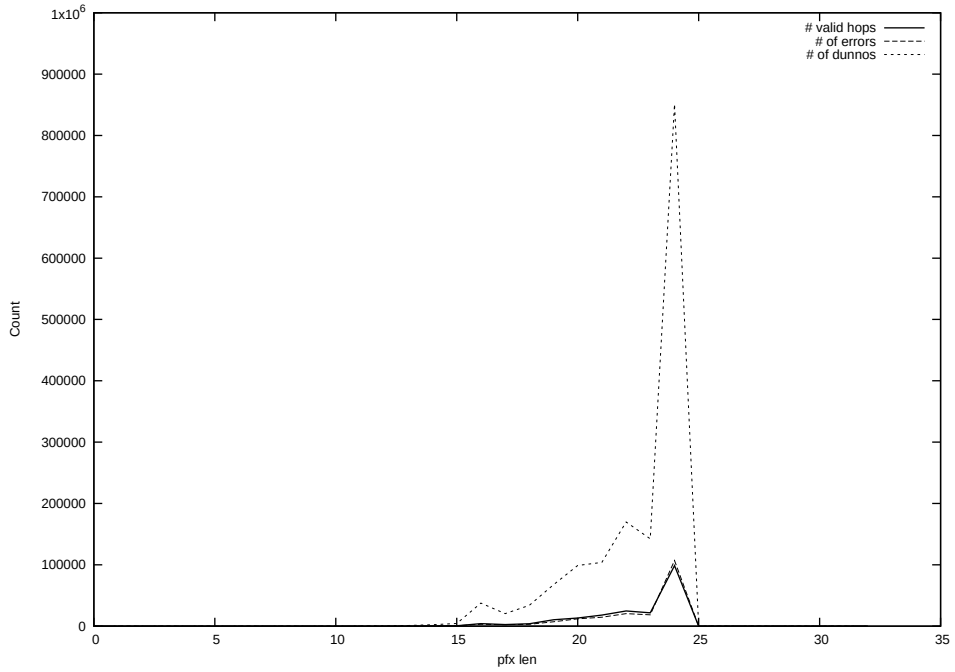
2014-11-05



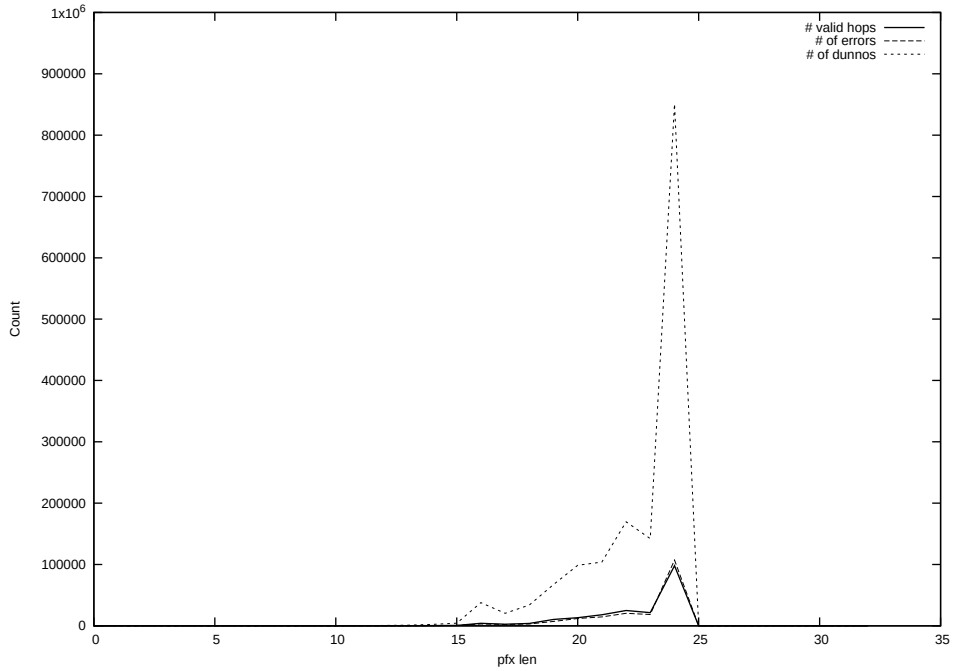
2014-11-06



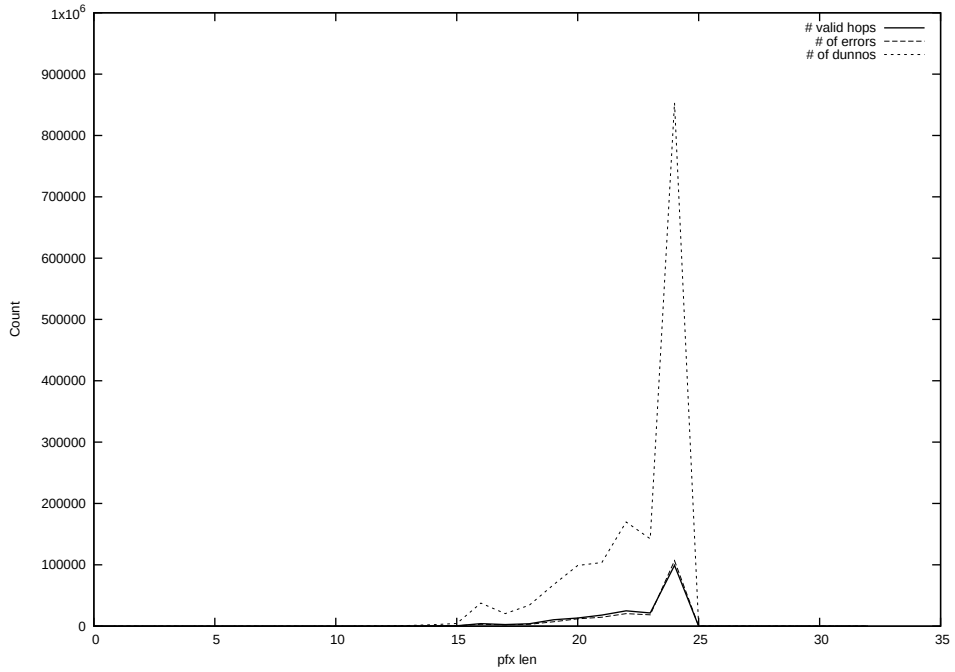
2014-11-07



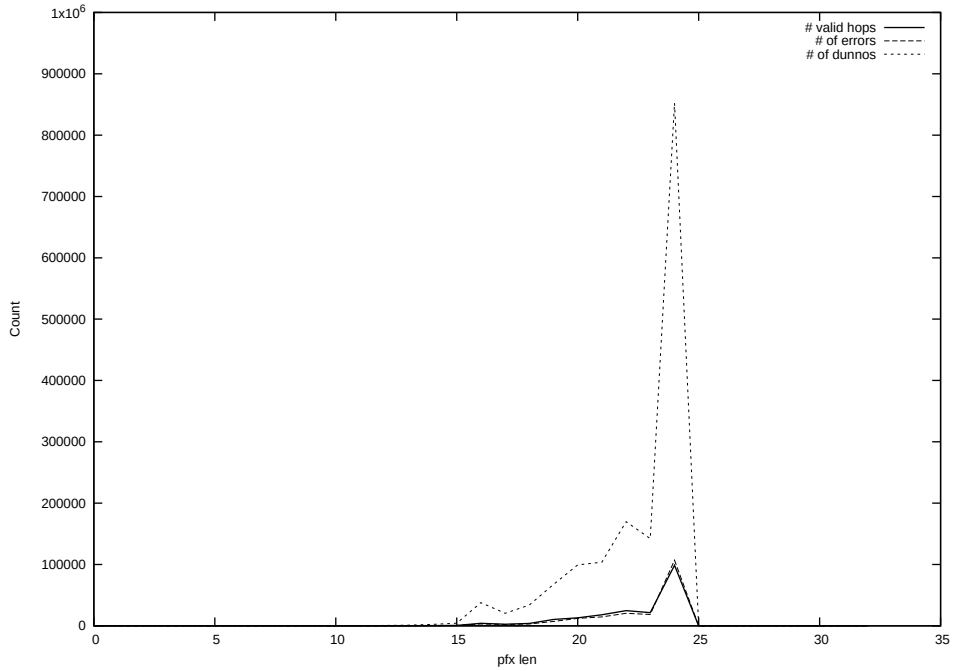
2014-11-08



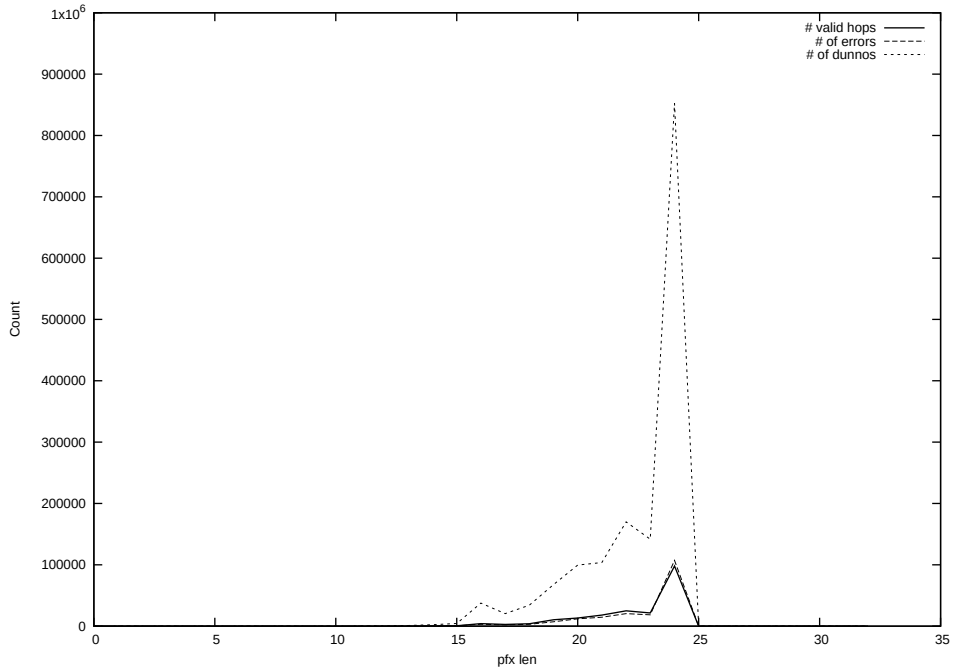
2014-11-09



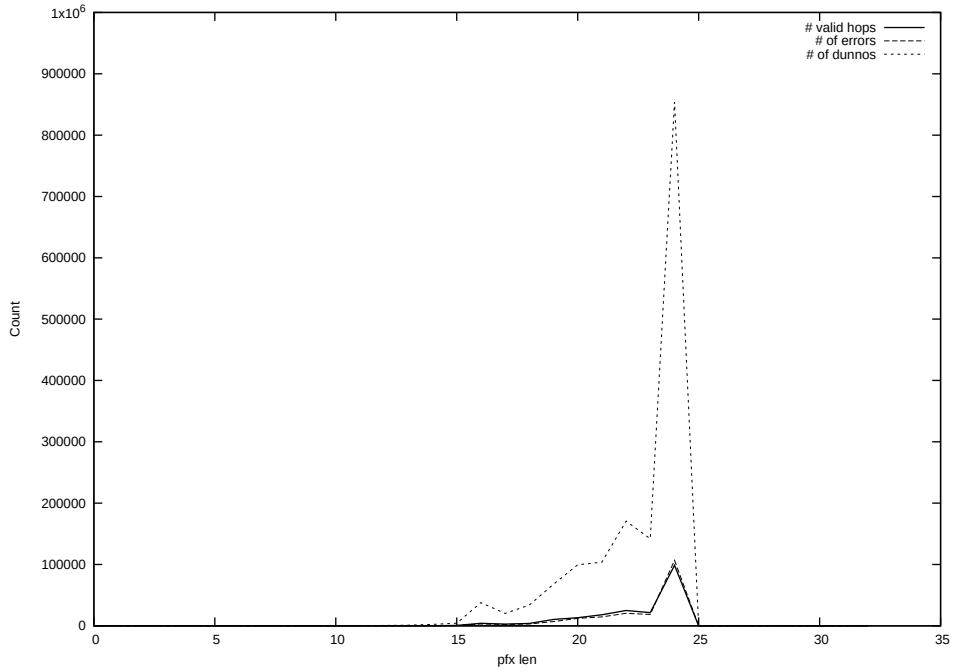
2014-11-10



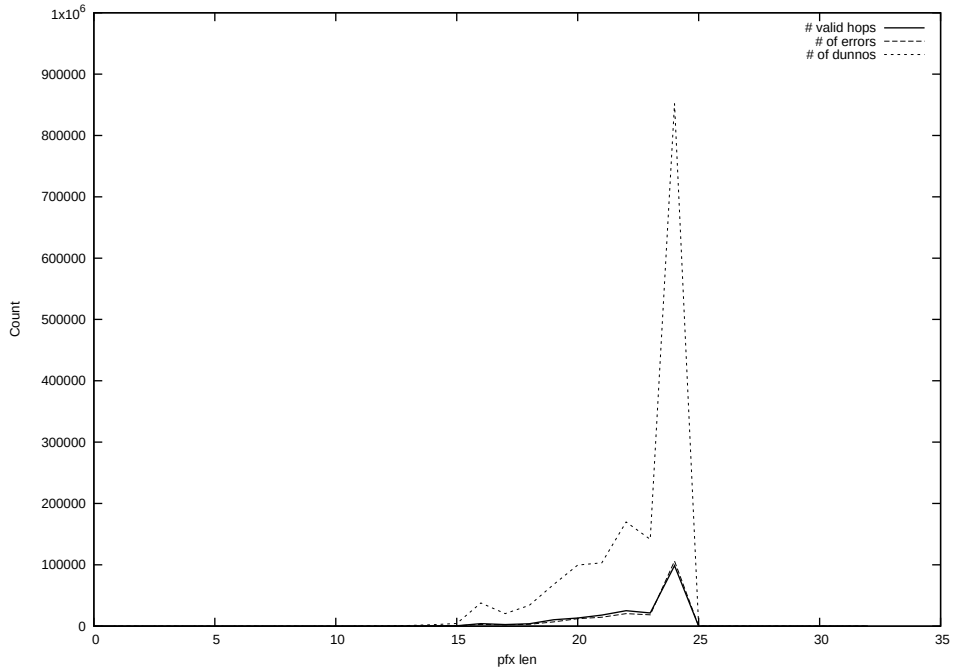
2014-11-11



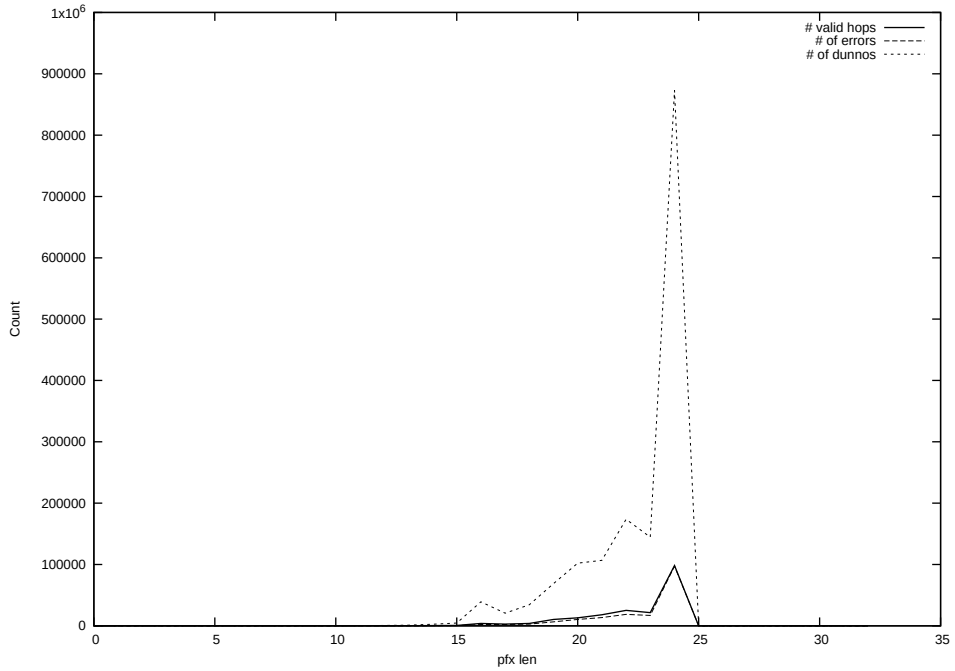
2014-11-12



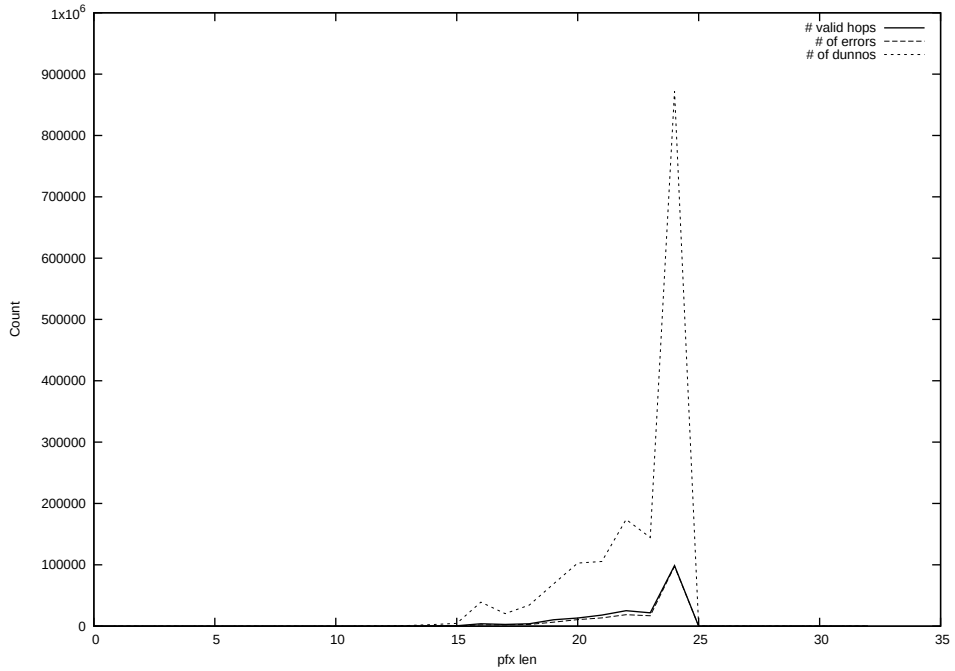
2014-11-13



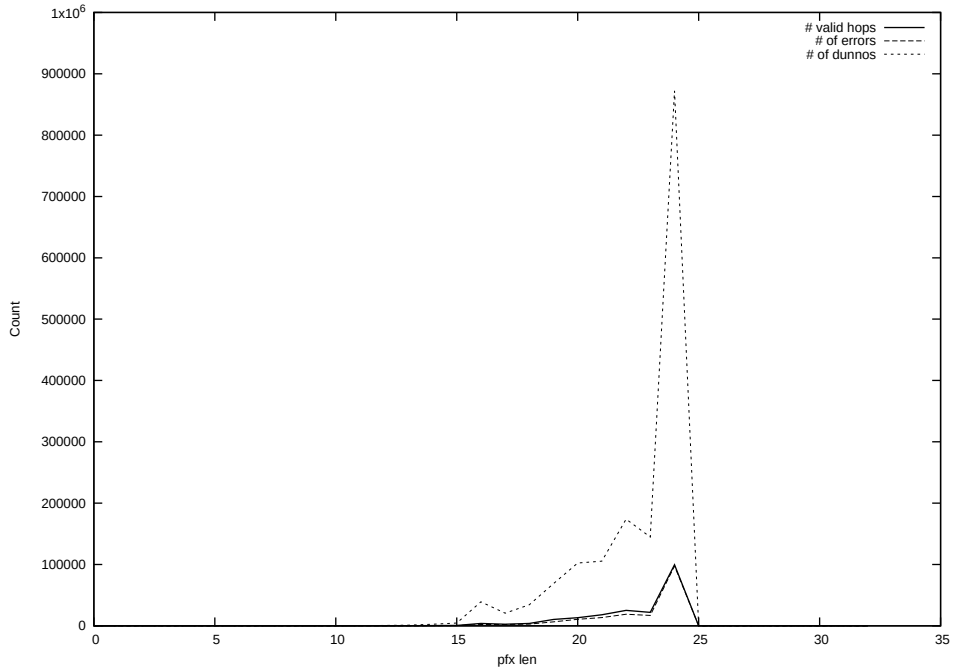
2014-11-14



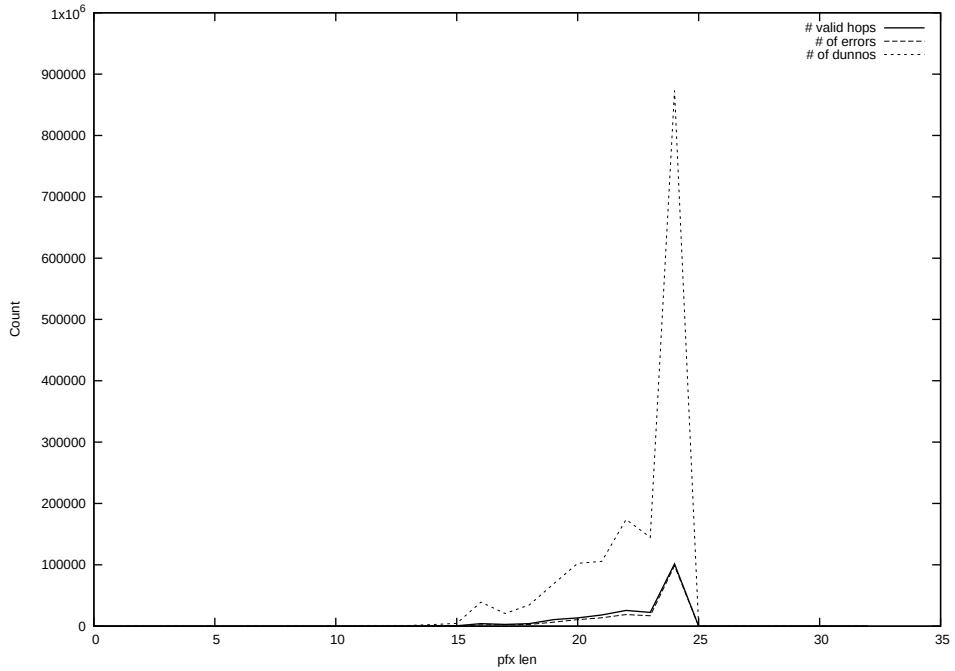
2014-11-15



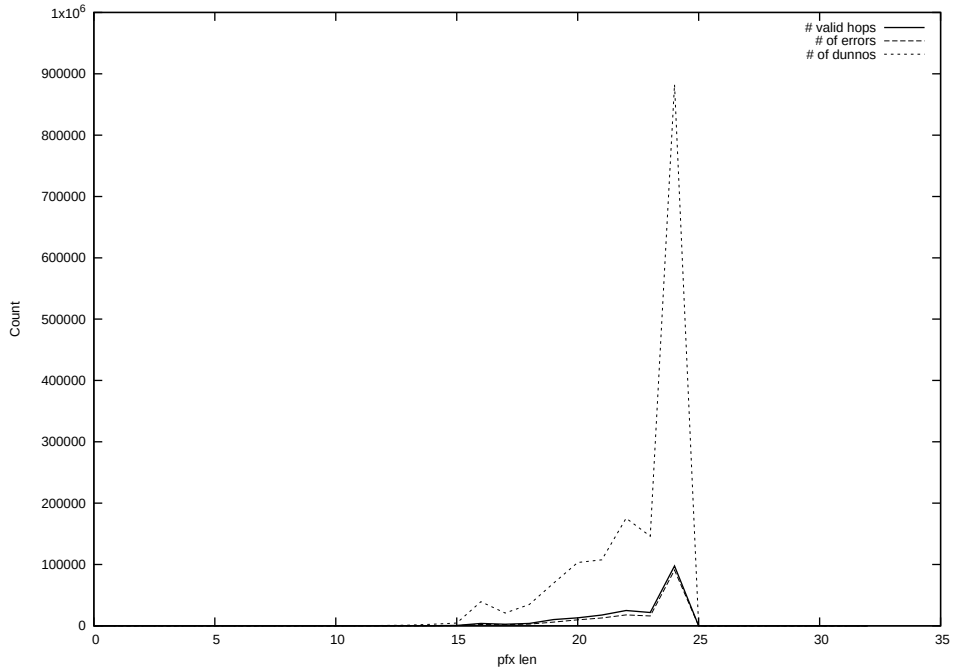
2014-11-16



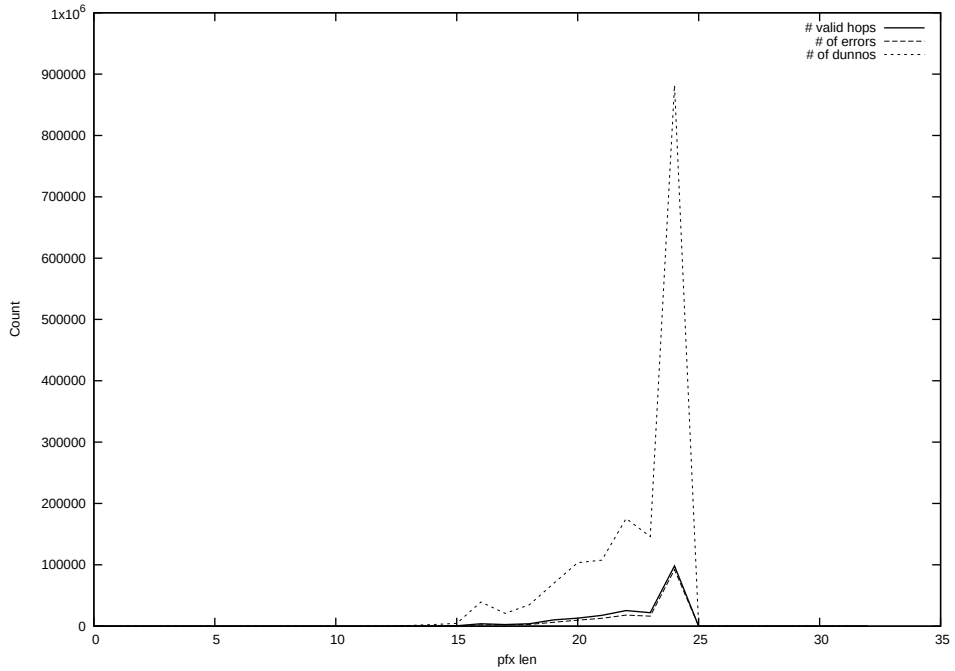
2014-11-17



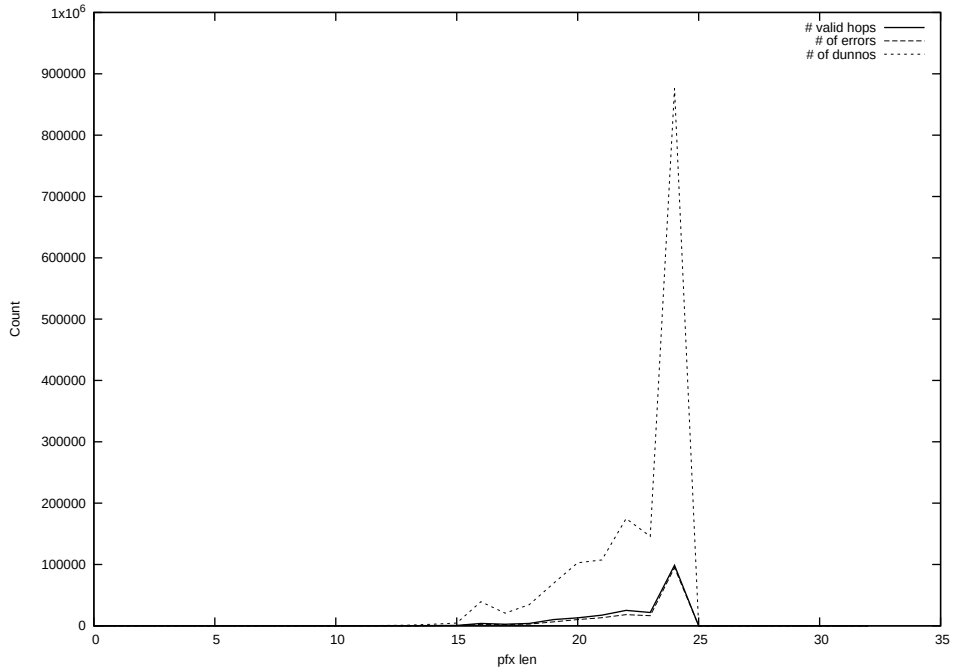
2014-11-18



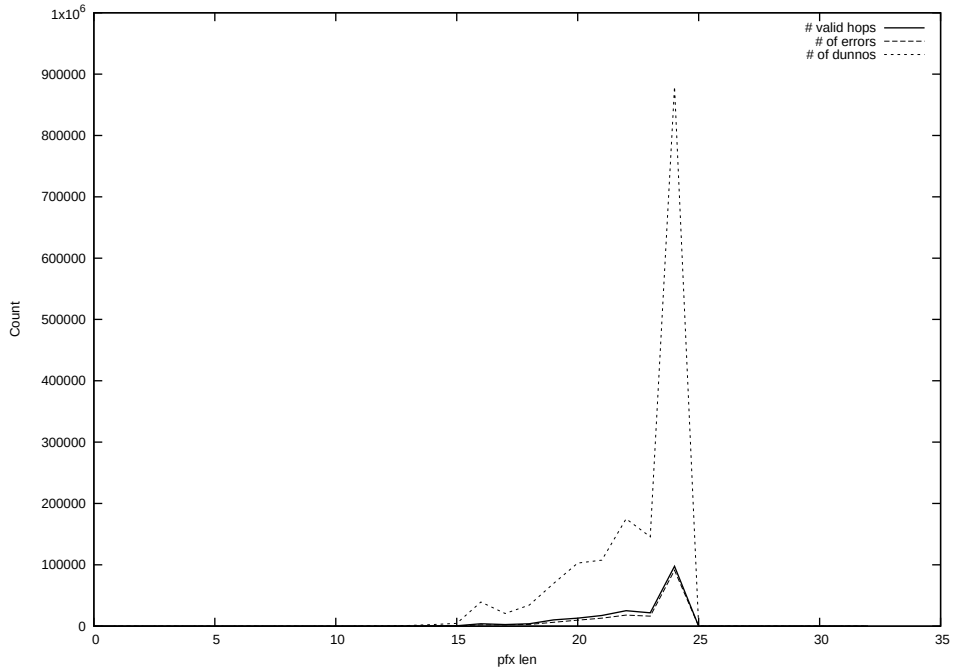
2014-11-19



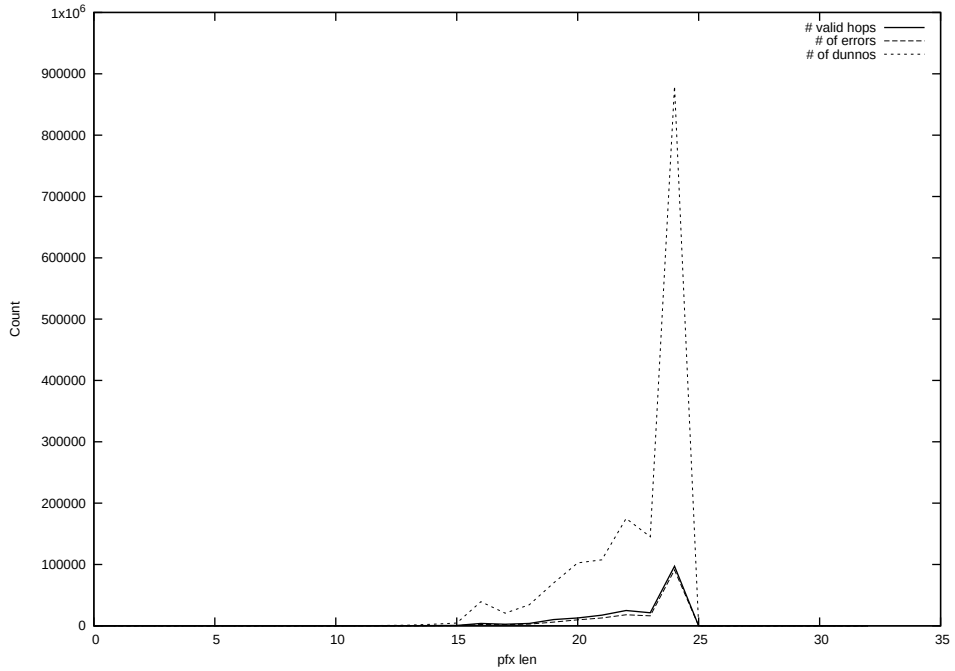
2014-11-20



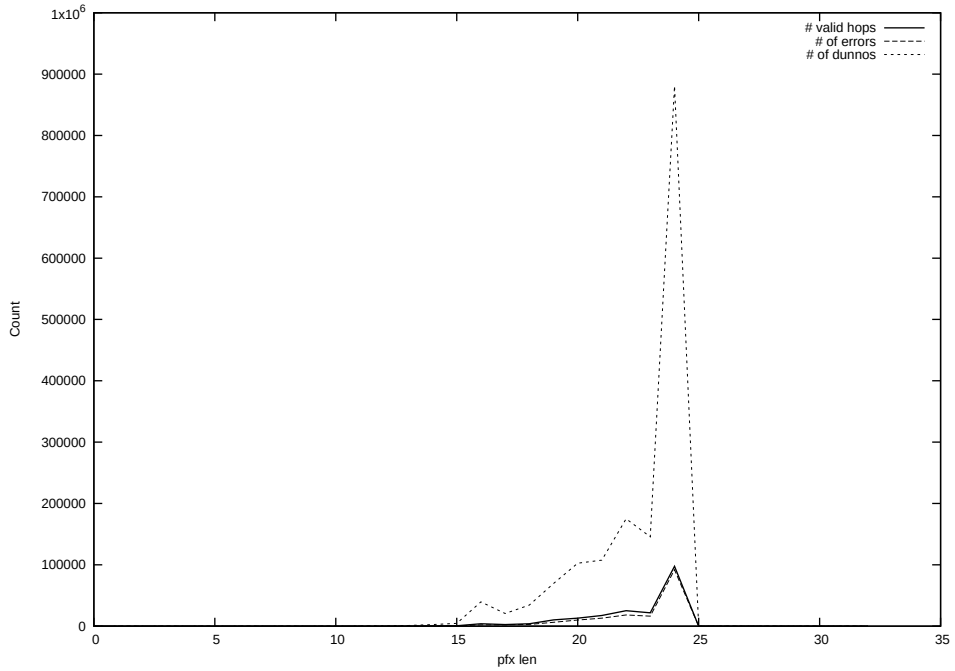
2014-11-21



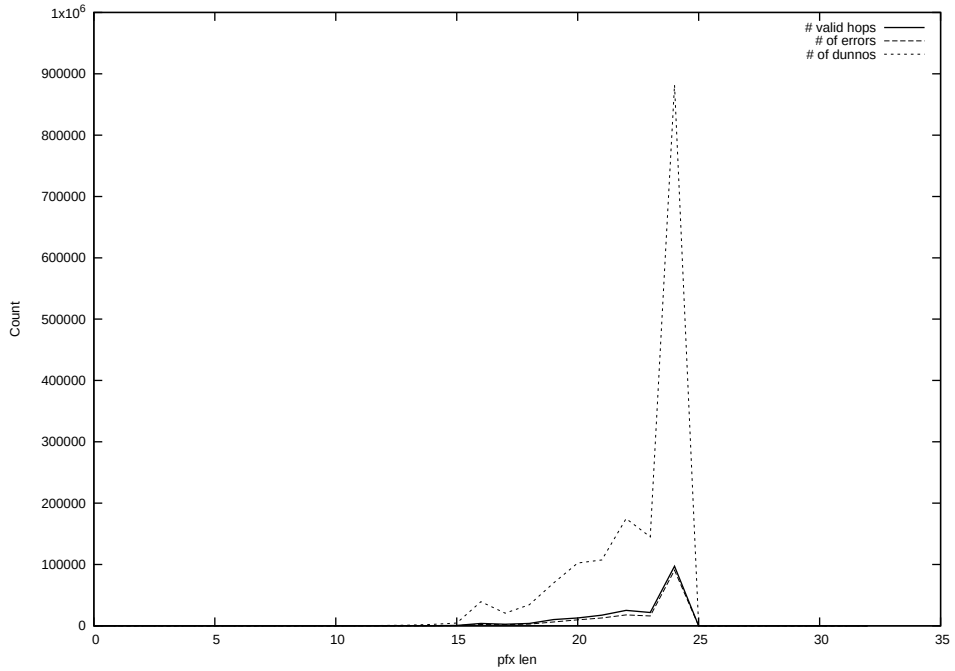
2014-11-22



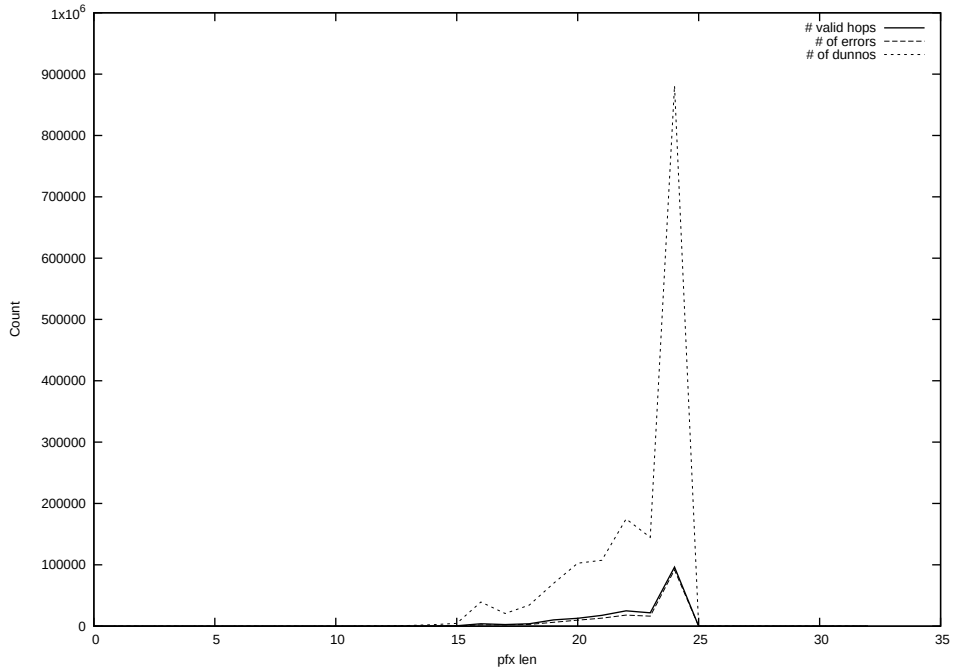
2014-11-23



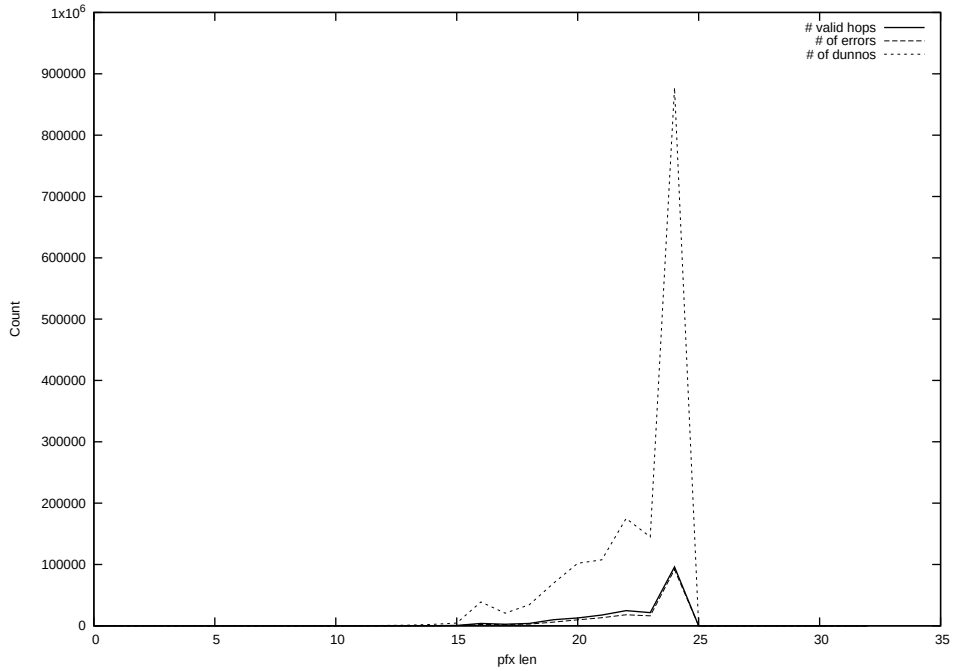
2014-11-24



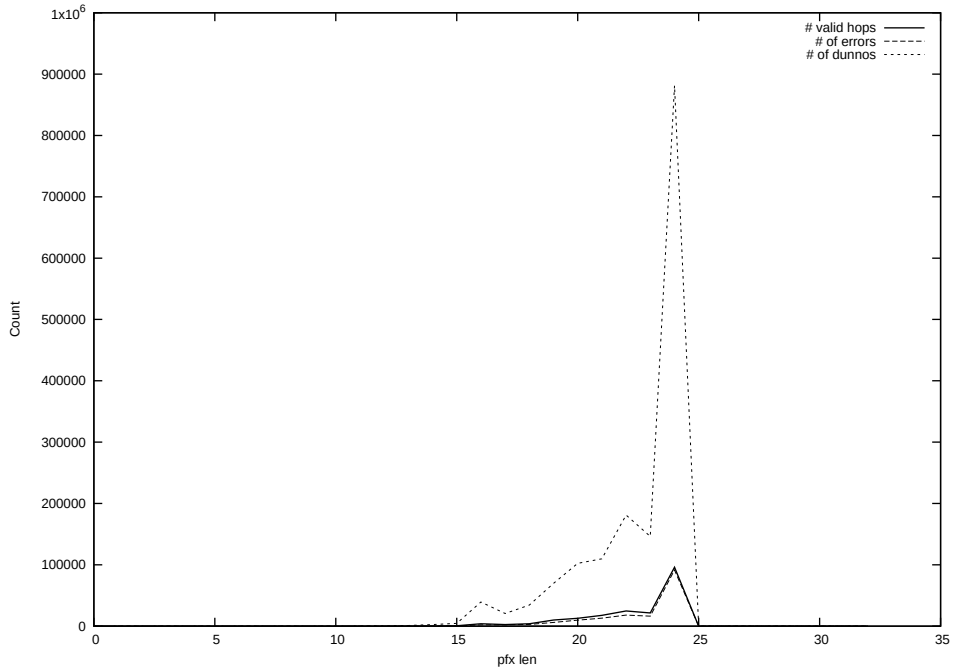
2014-11-25



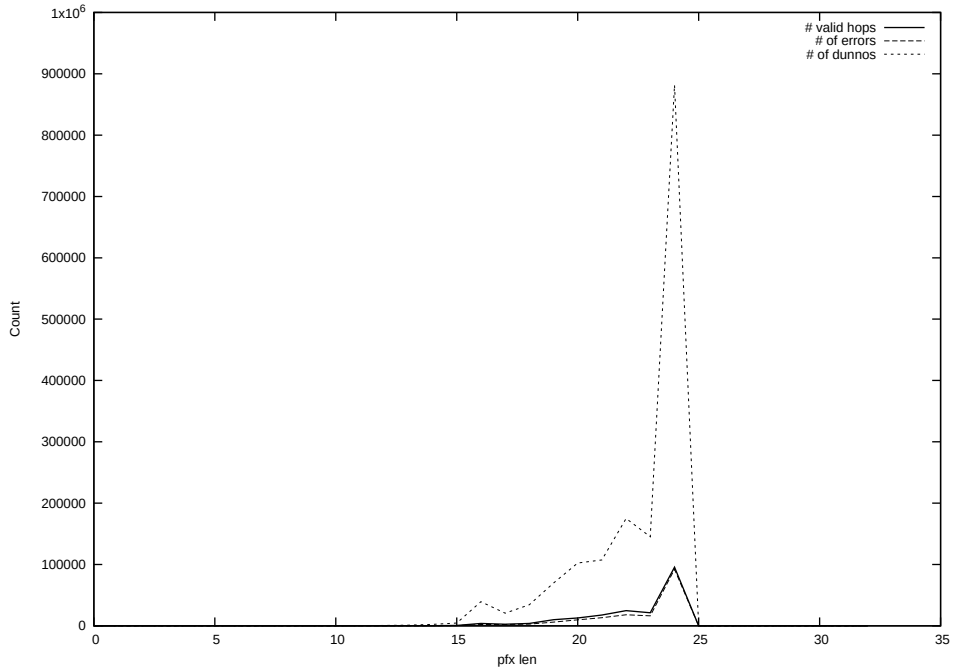
2014-11-26



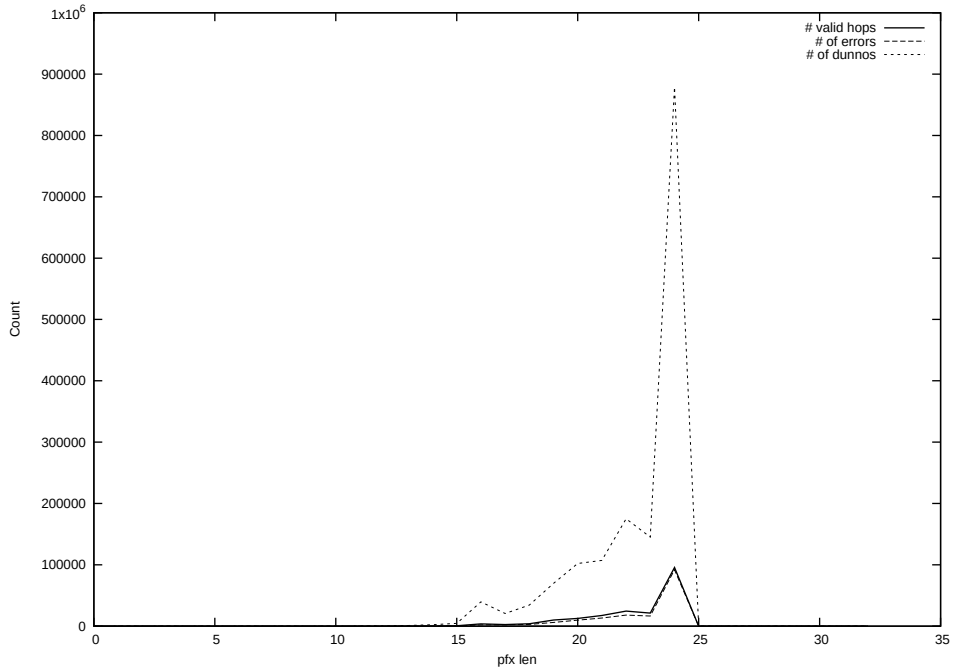
2014-11-27



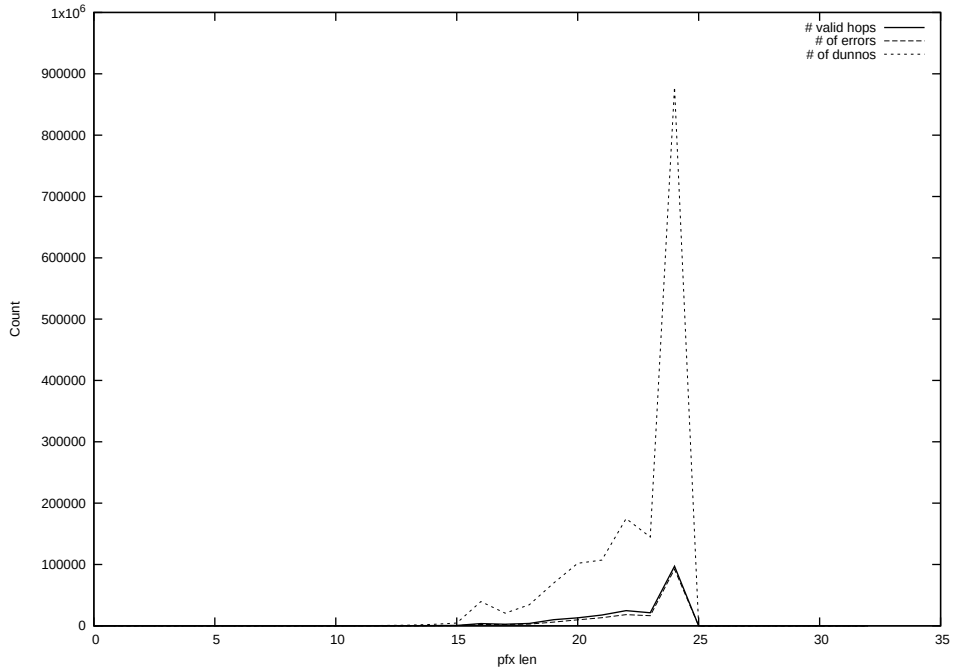
2014-11-28



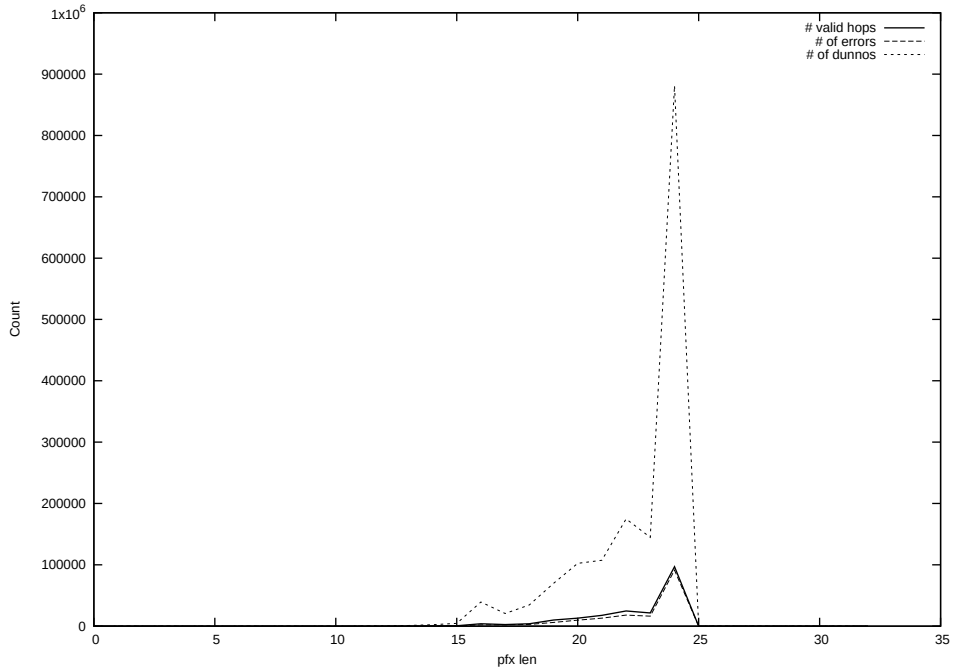
2014-11-29



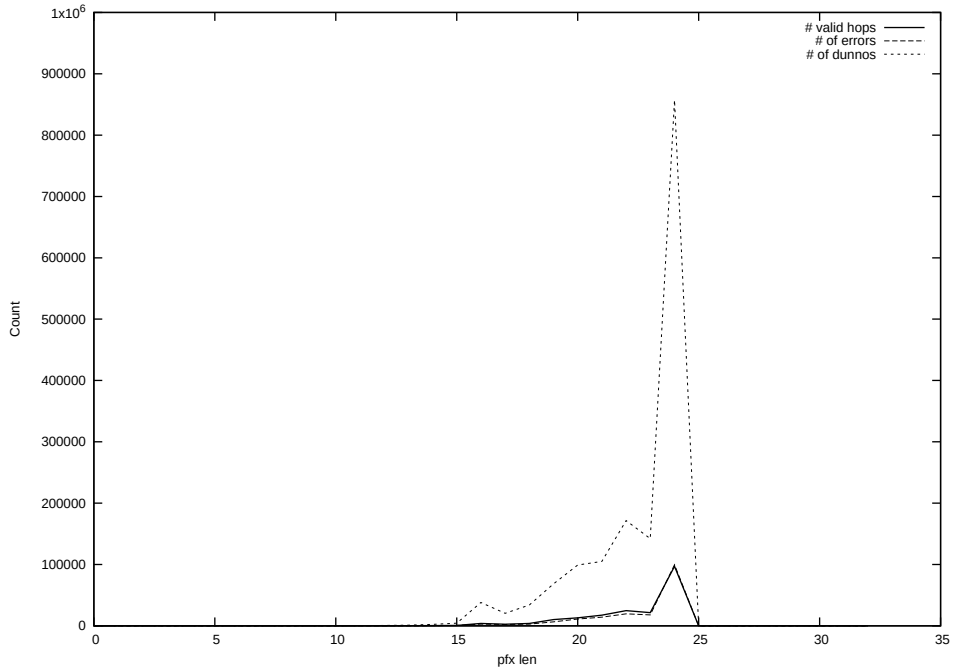
2014-11-30



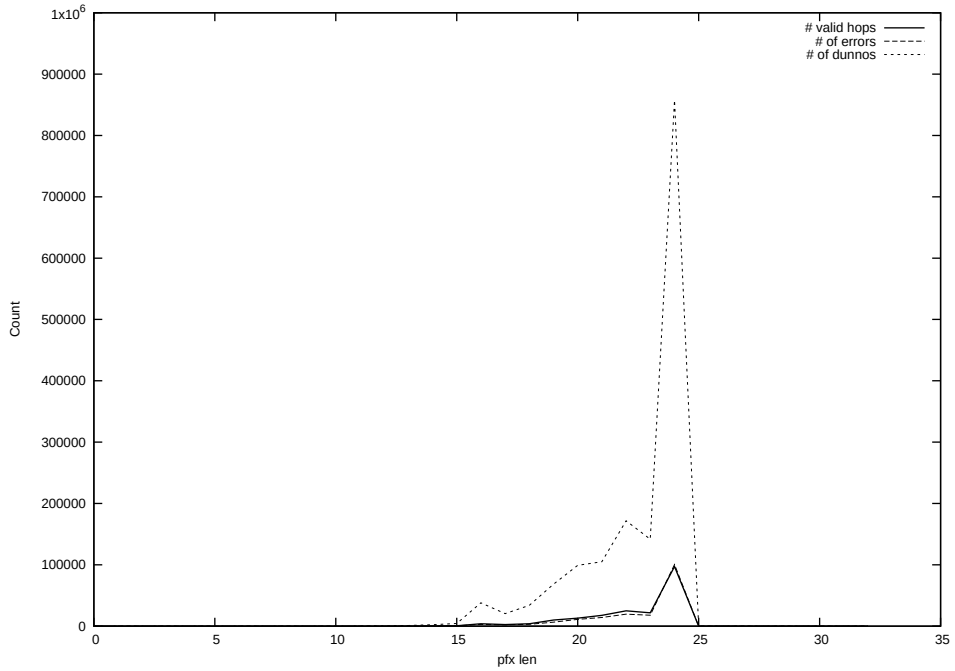
2014-12-01



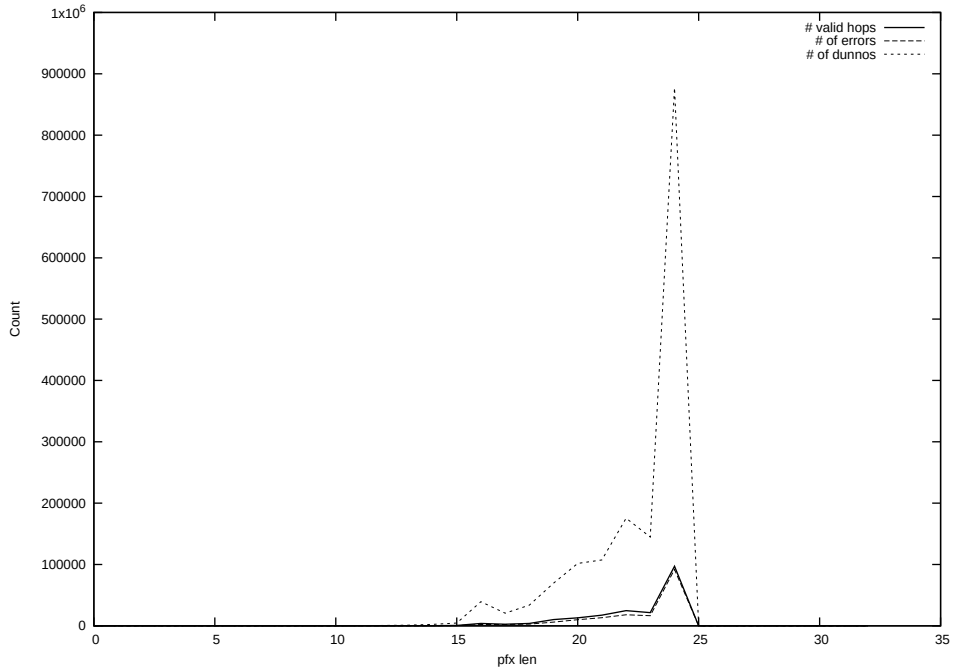
2014-12-02



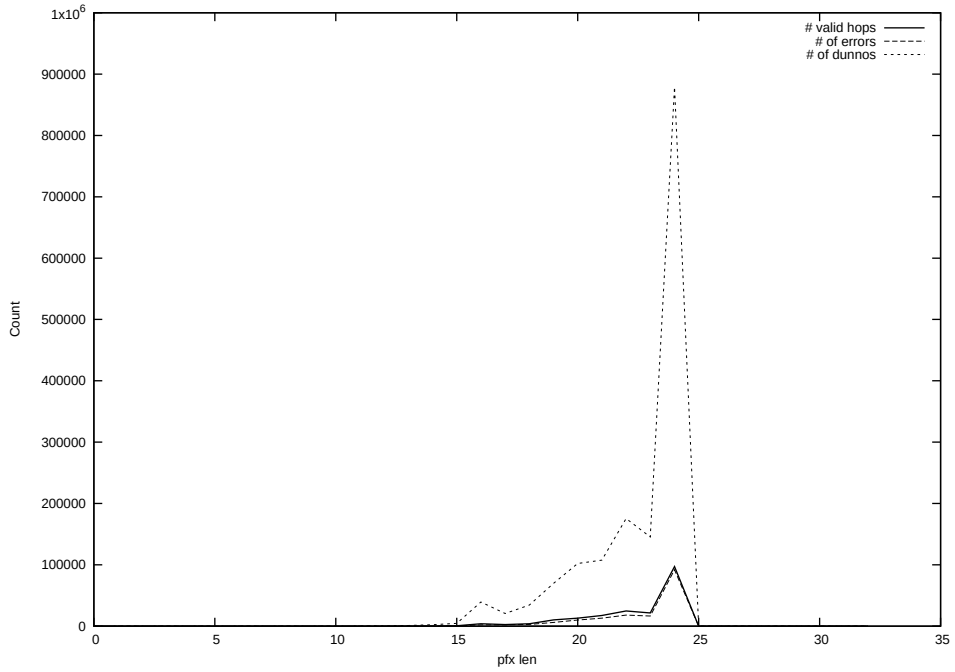
2014-12-03



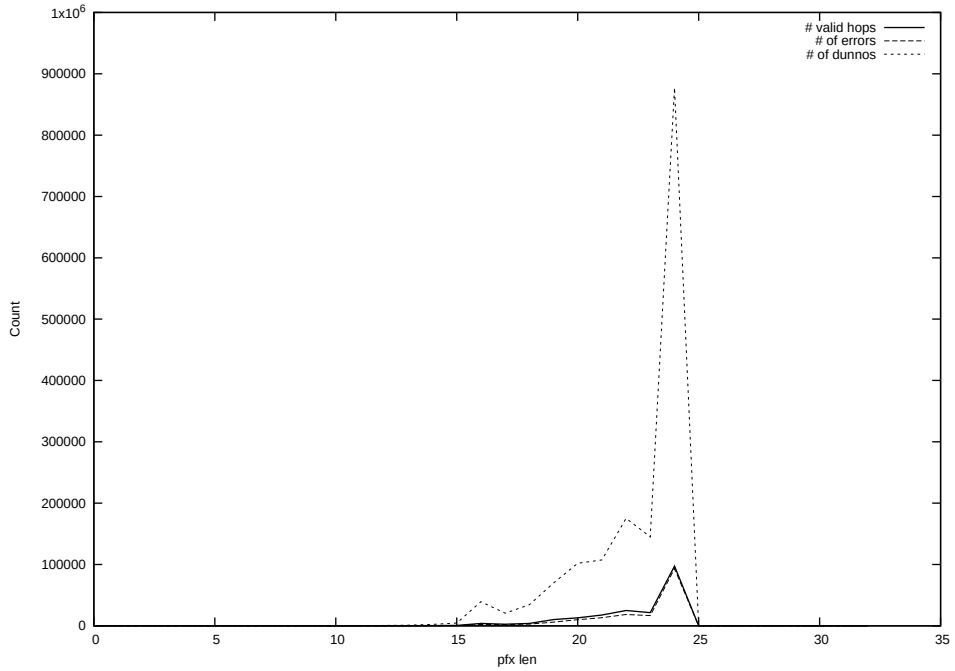
2014-12-04



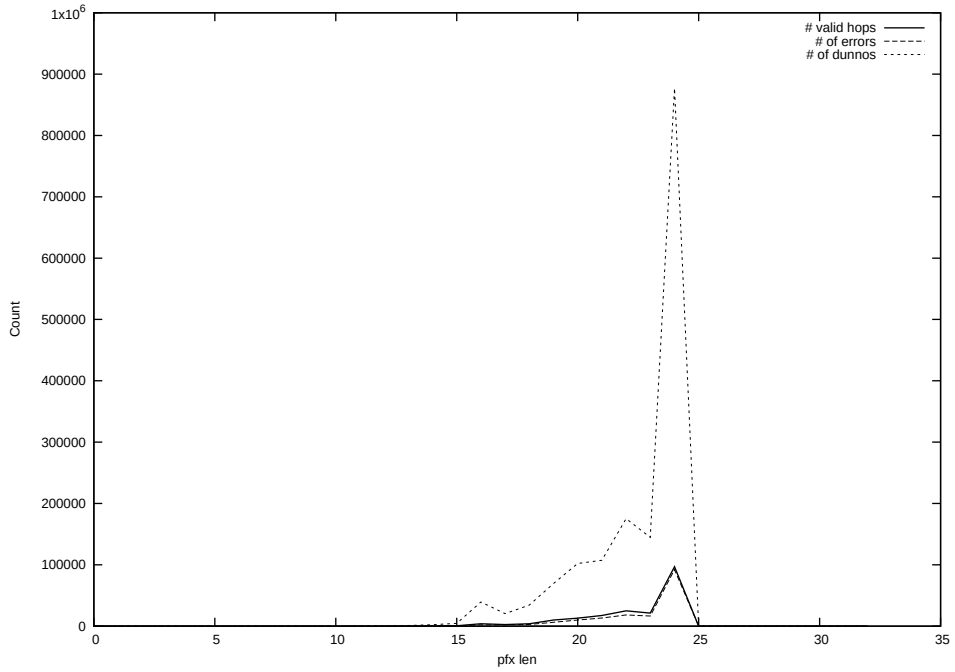
2014-12-05



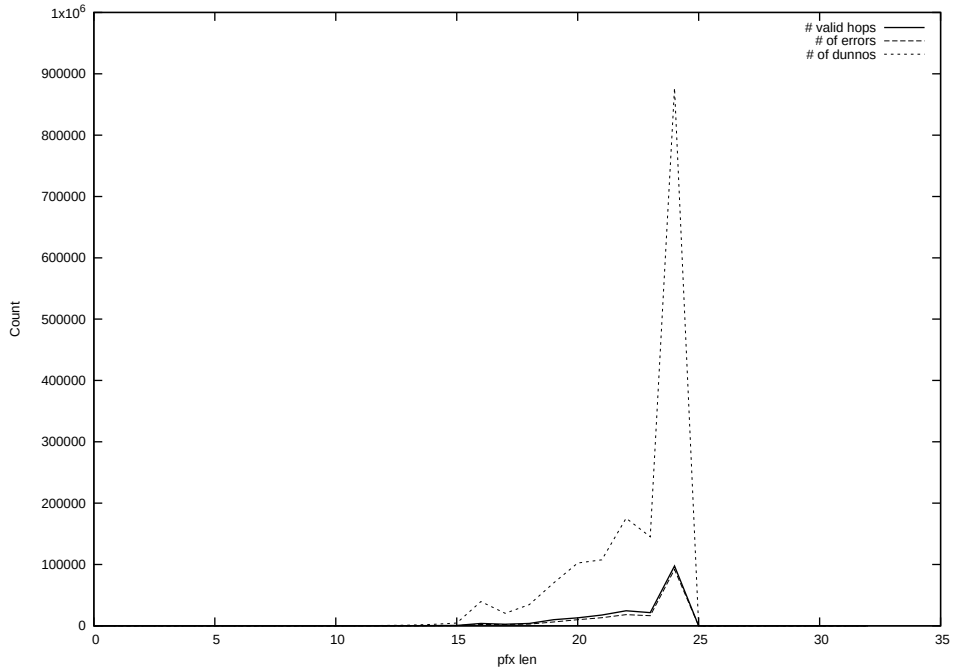
2014-12-06



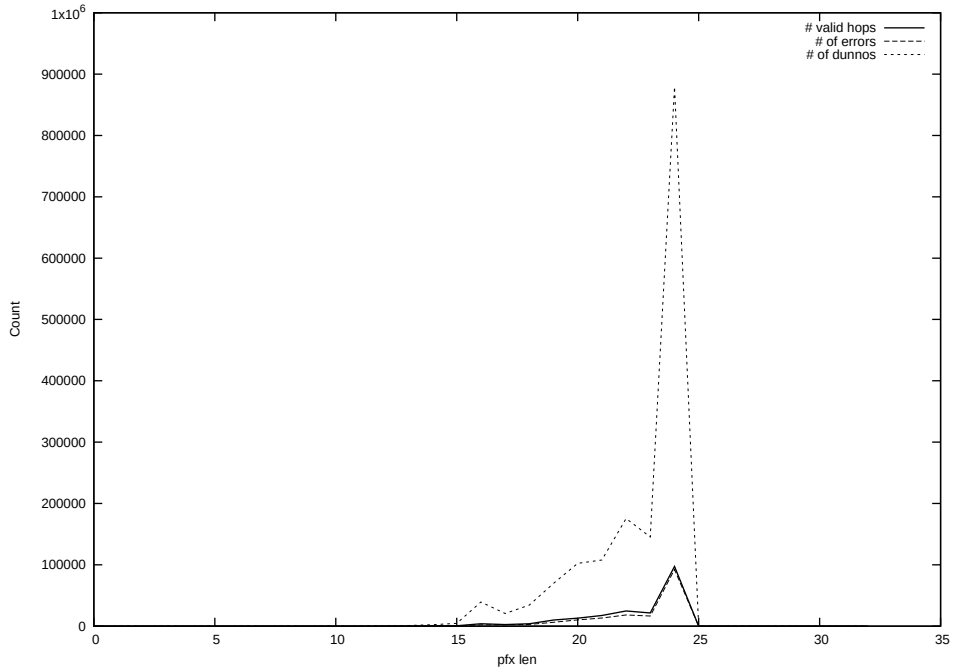
2014-12-07



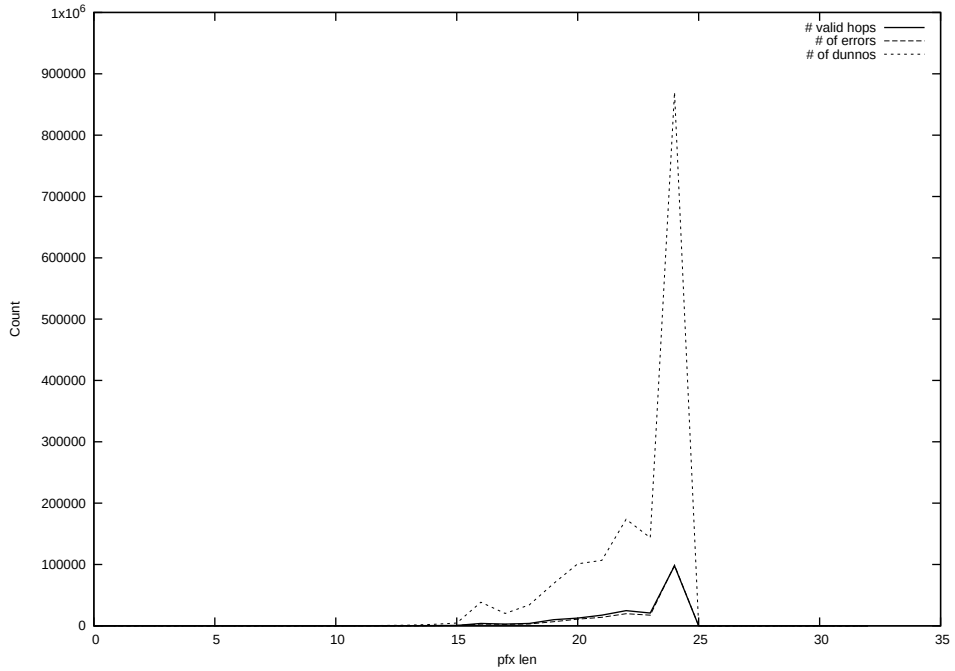
2014-12-08



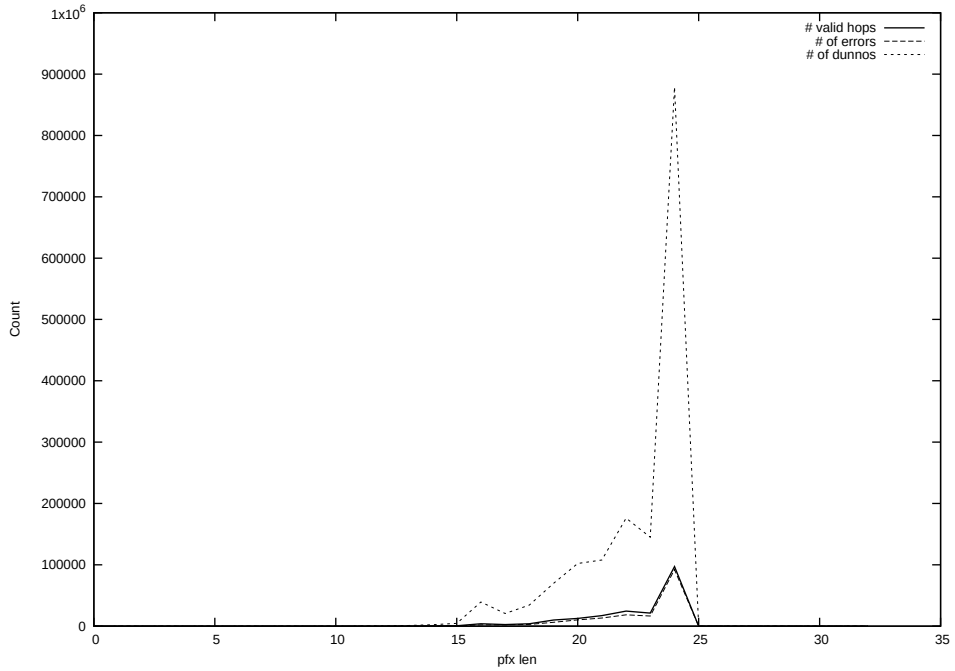
2014-12-09



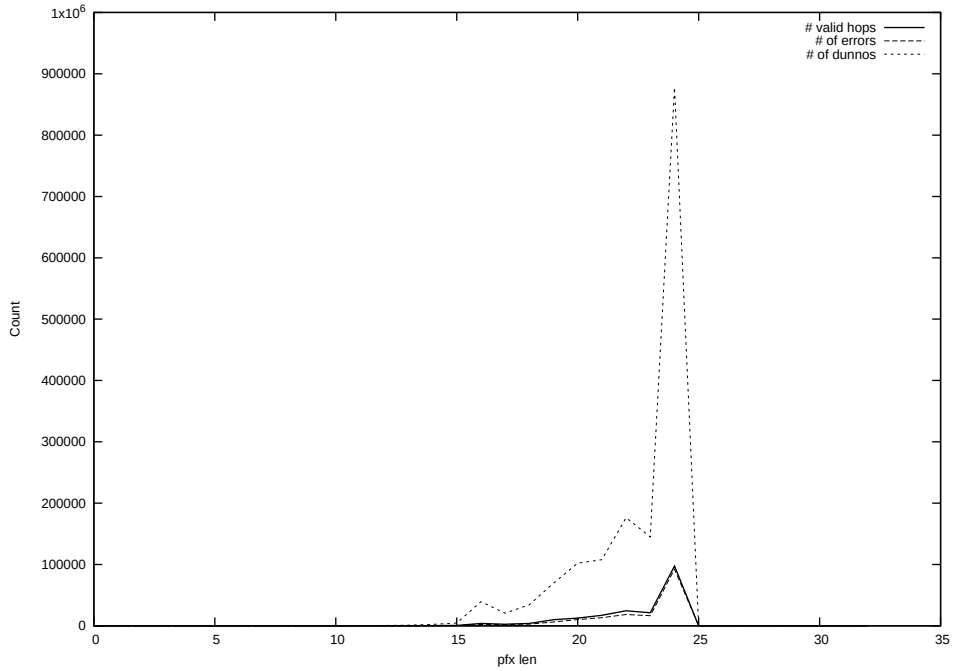
2014-12-10



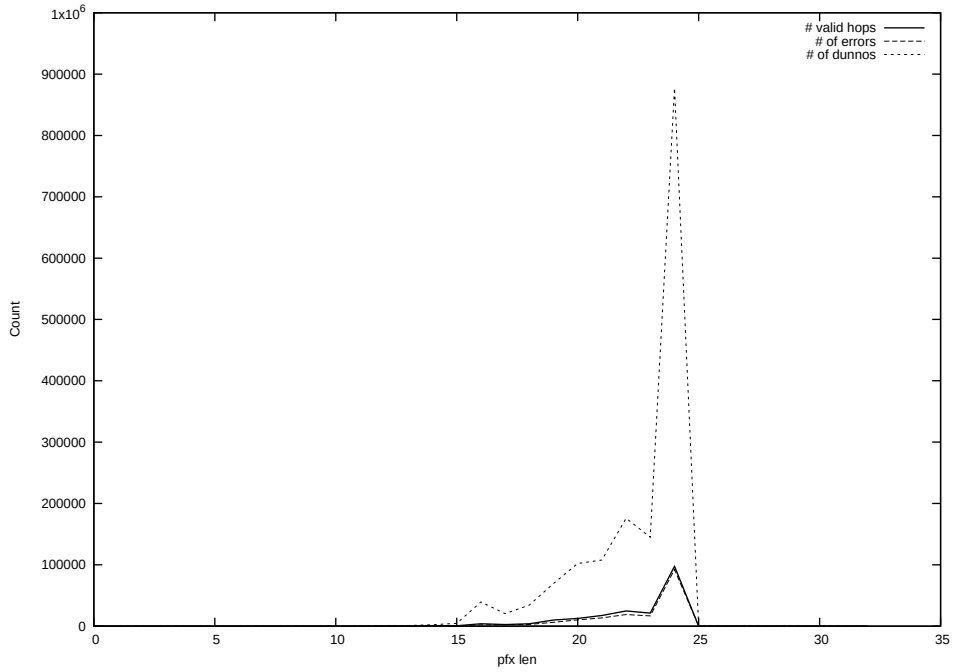
2014-12-11



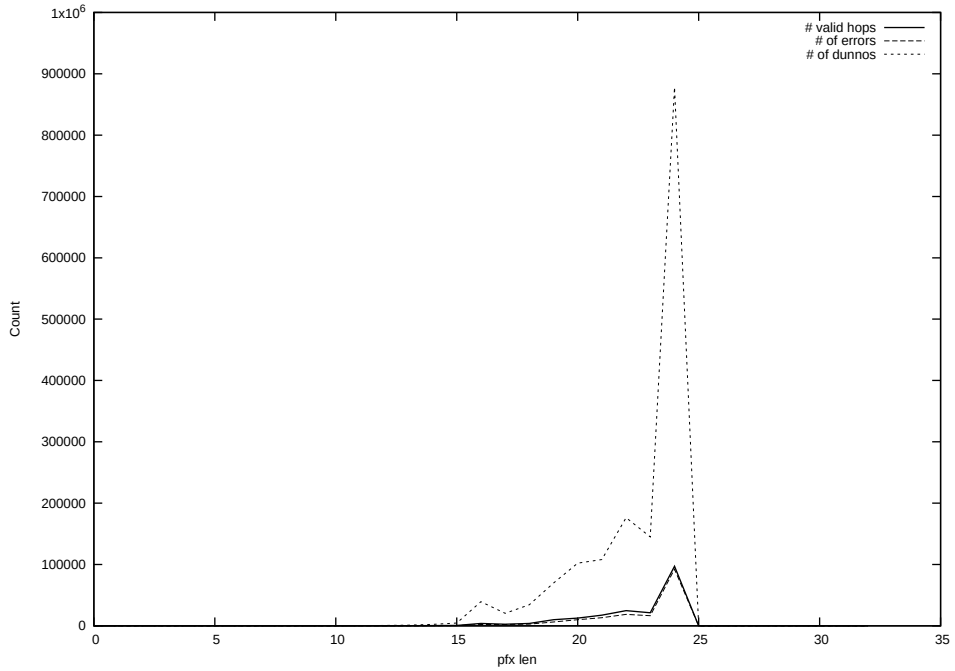
2014-12-12



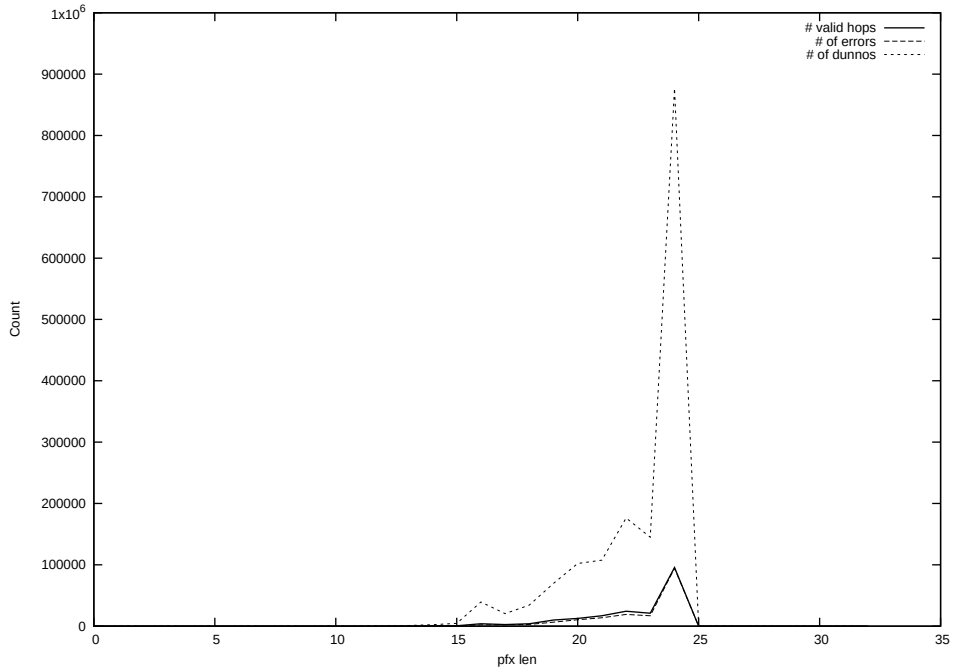
2014-12-13



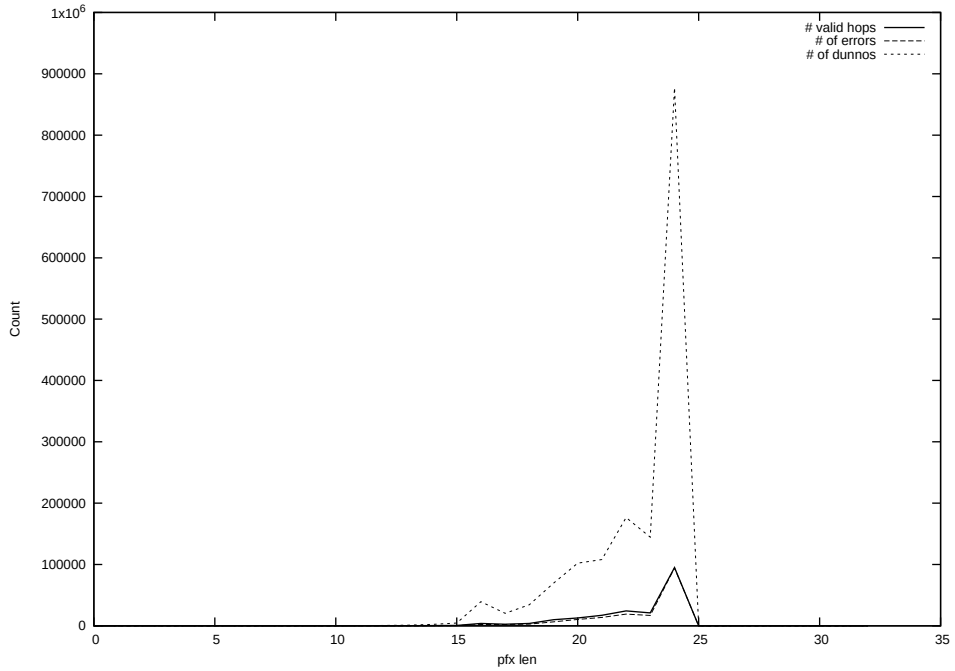
2014-12-14



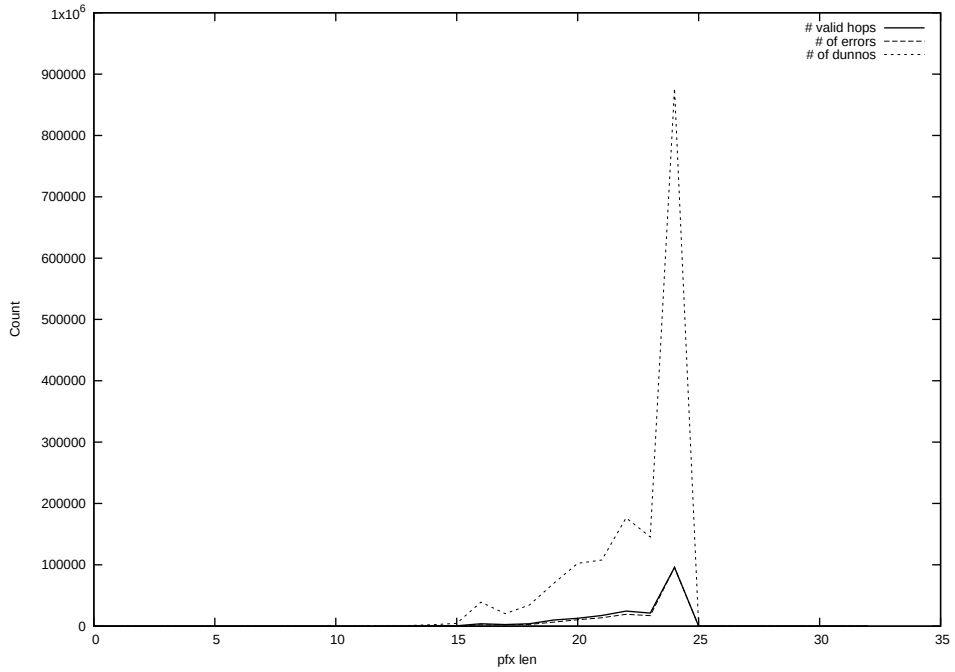
2014-12-15



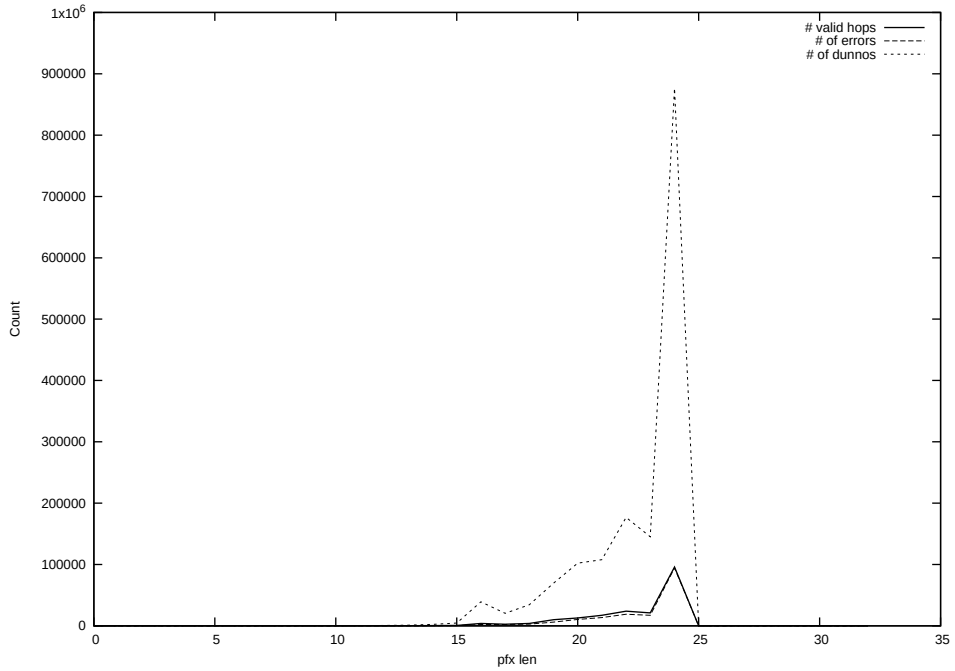
2014-12-16



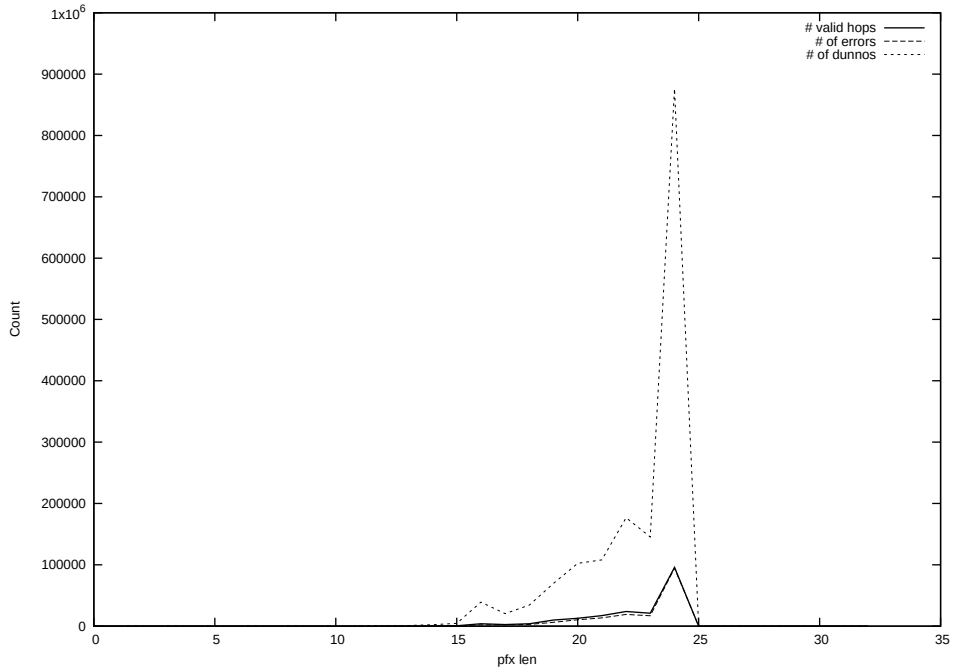
2014-12-17



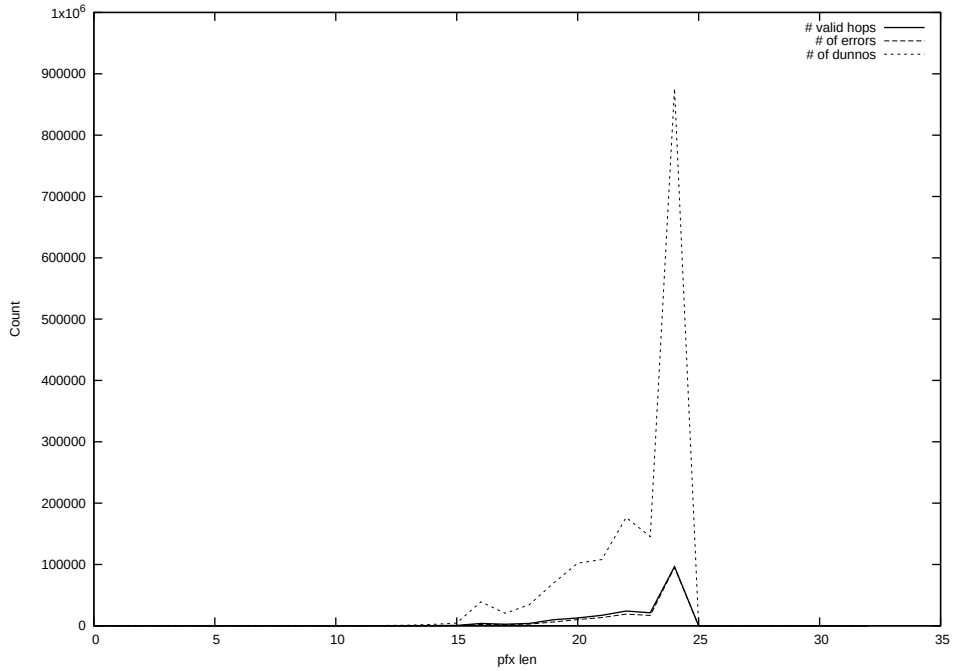
2014-12-18



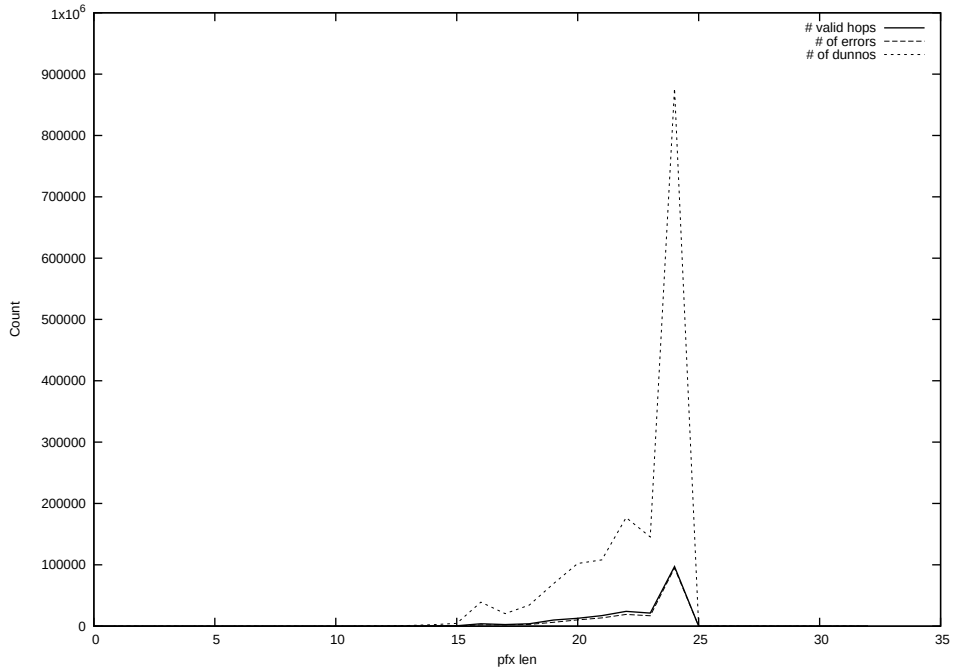
2014-12-19



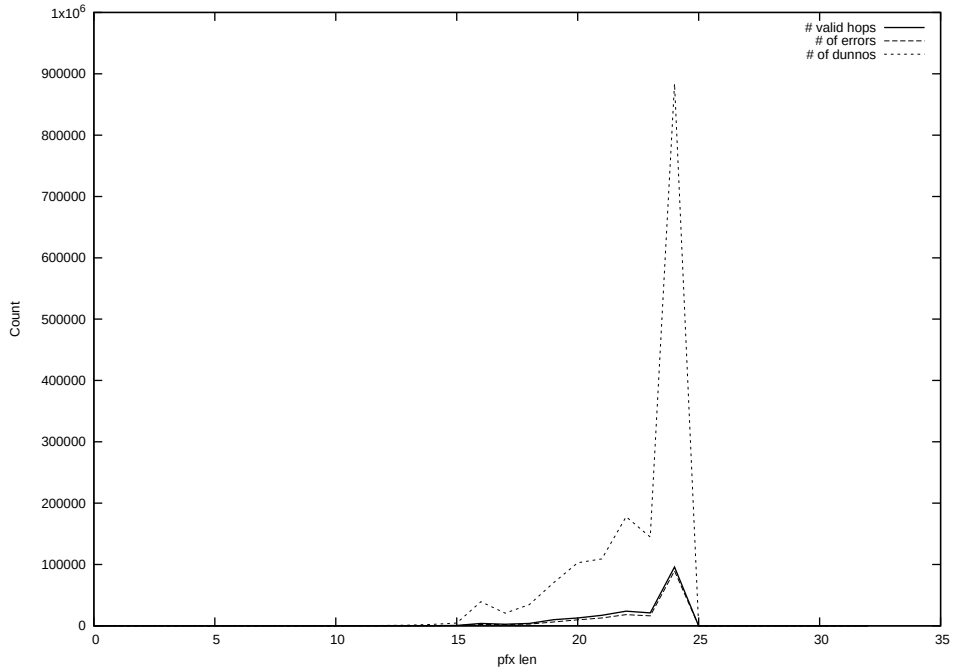
2014-12-20



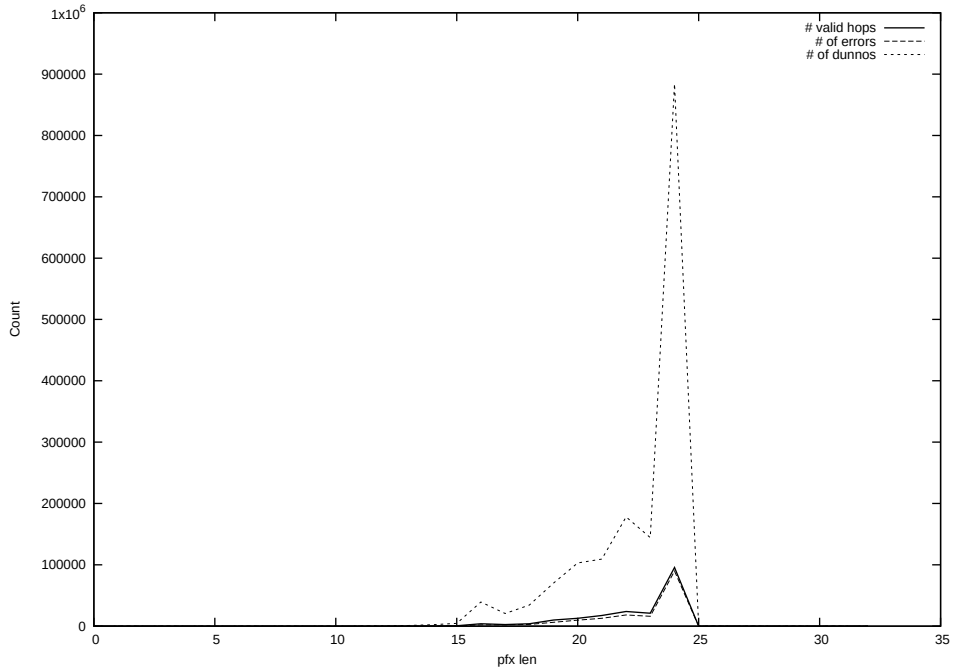
2014-12-21



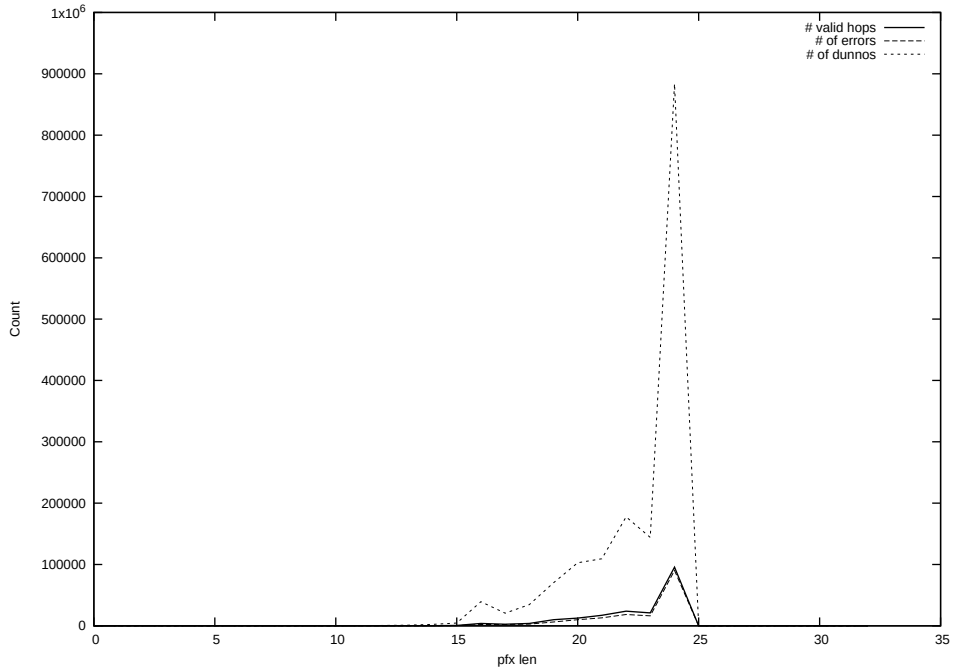
2014-12-22



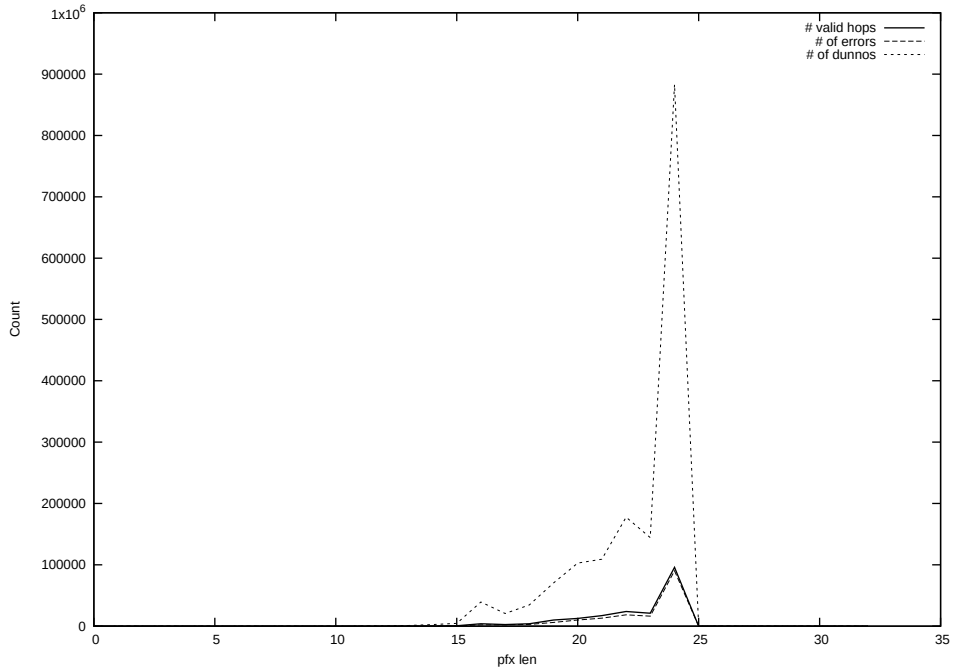
2014-12-23



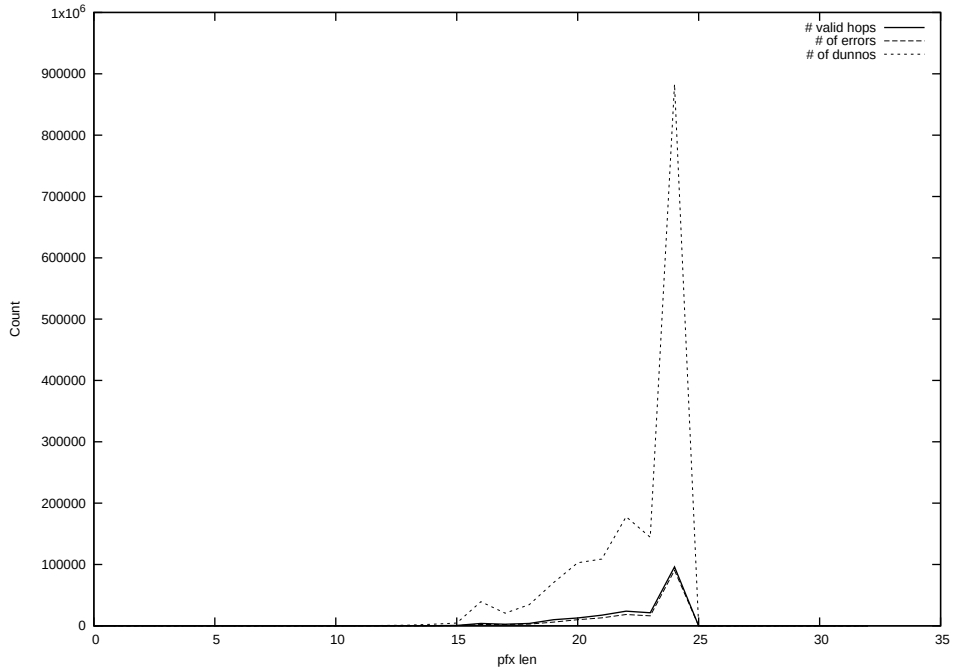
2014-12-24



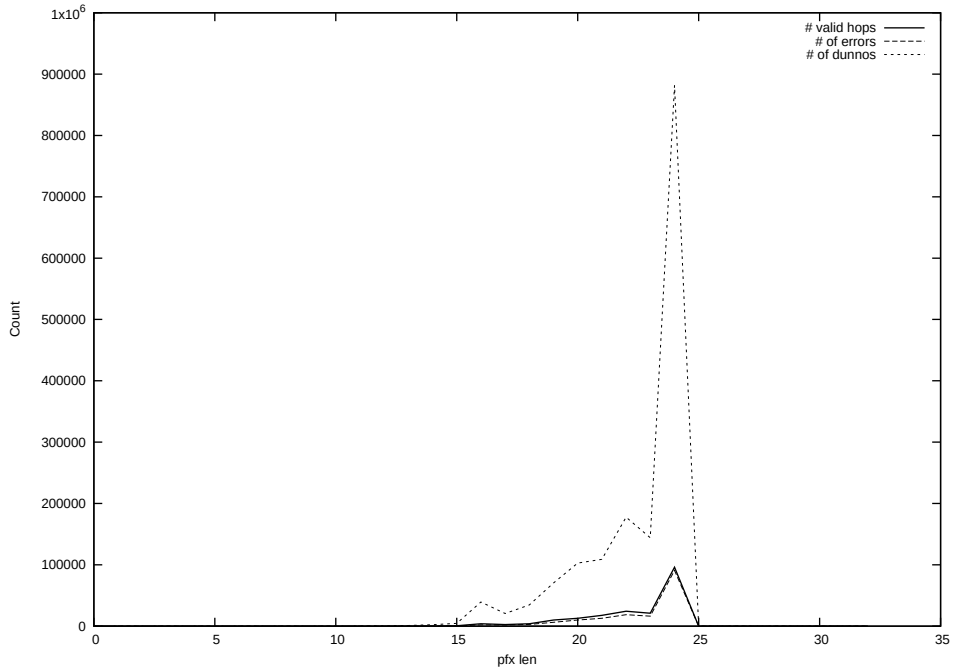
2014-12-25



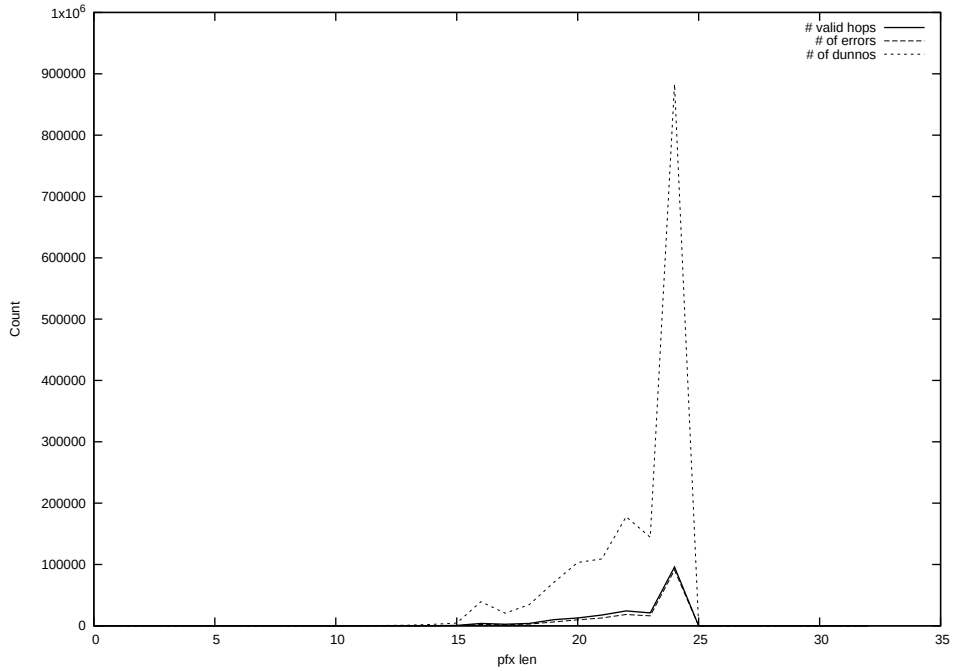
2014-12-26



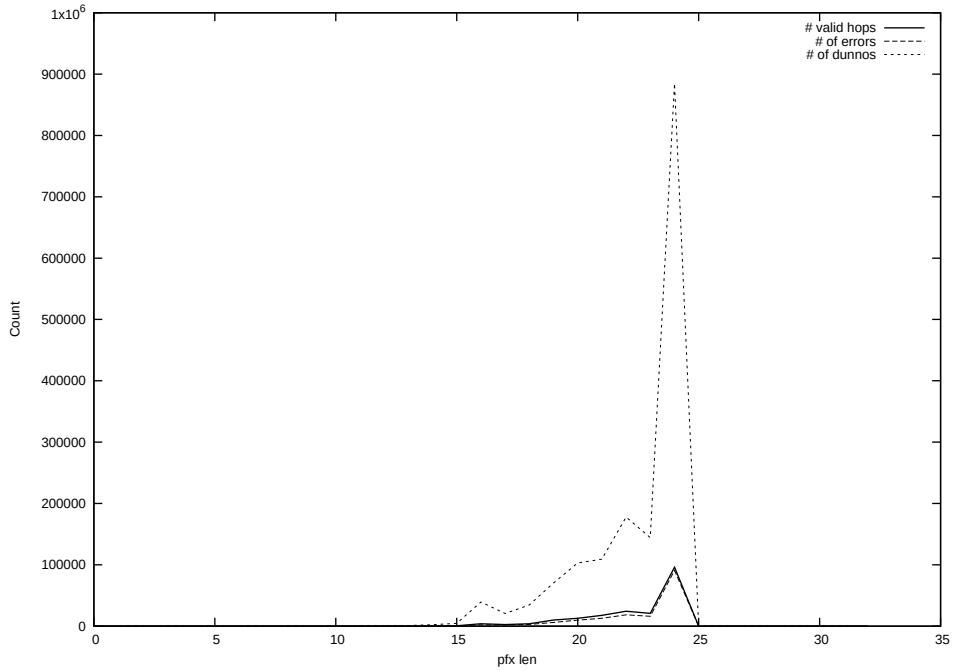
2014-12-27



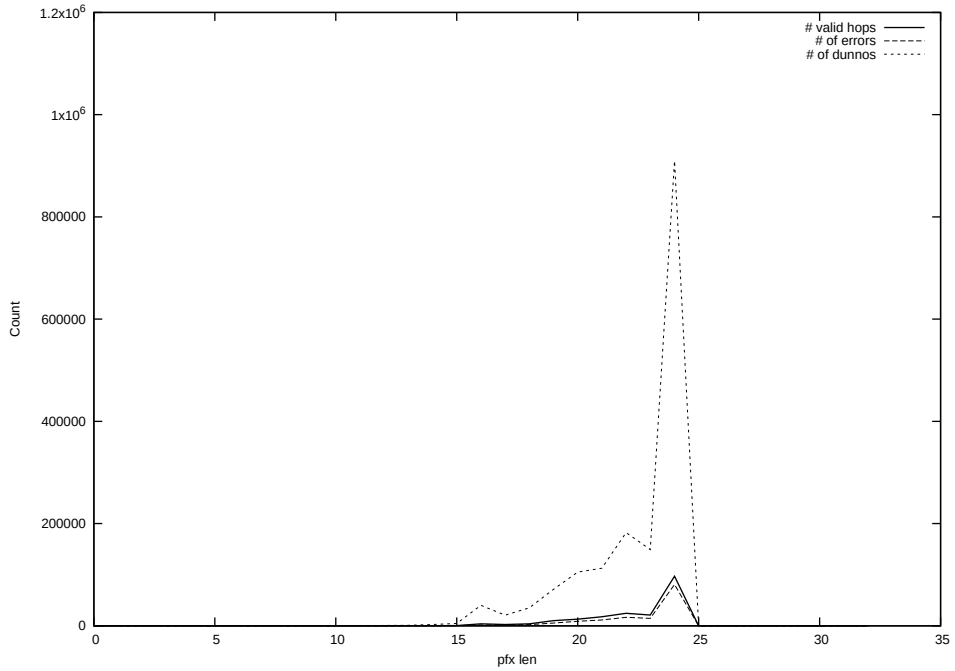
2014-12-28



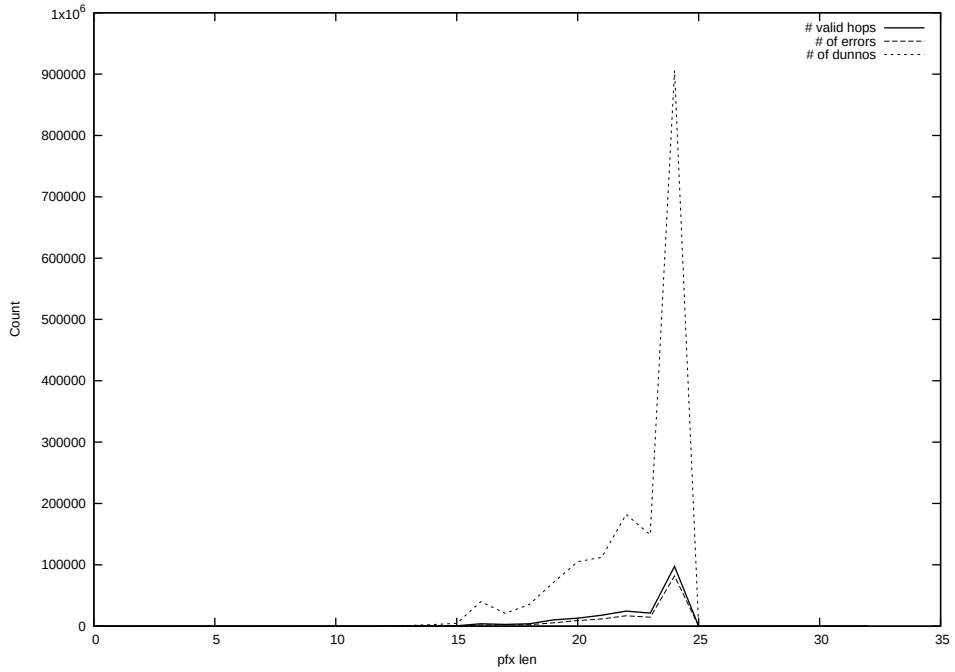
2014-12-29



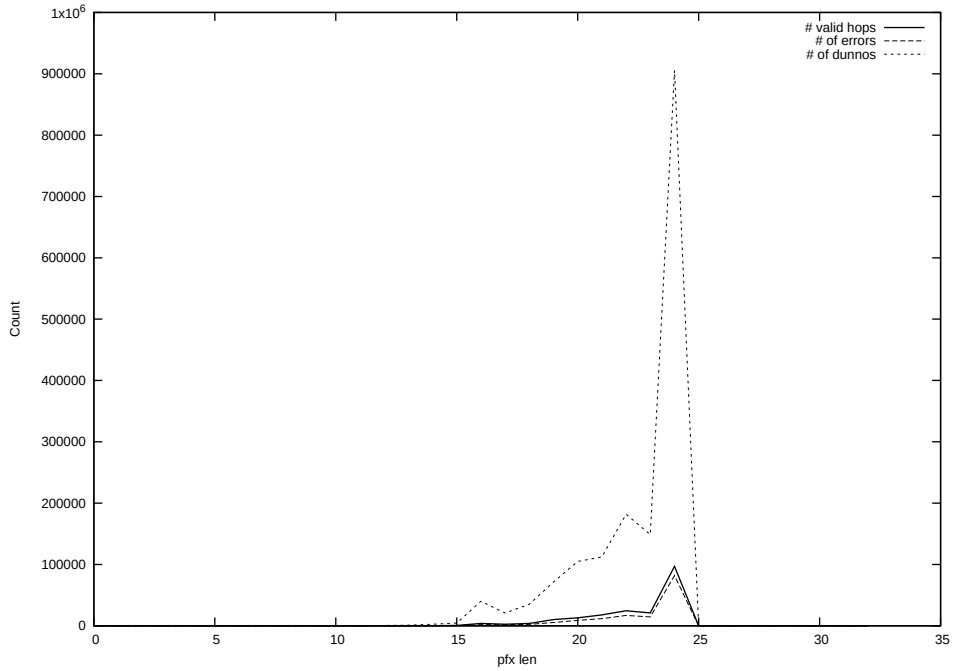
2014-12-30



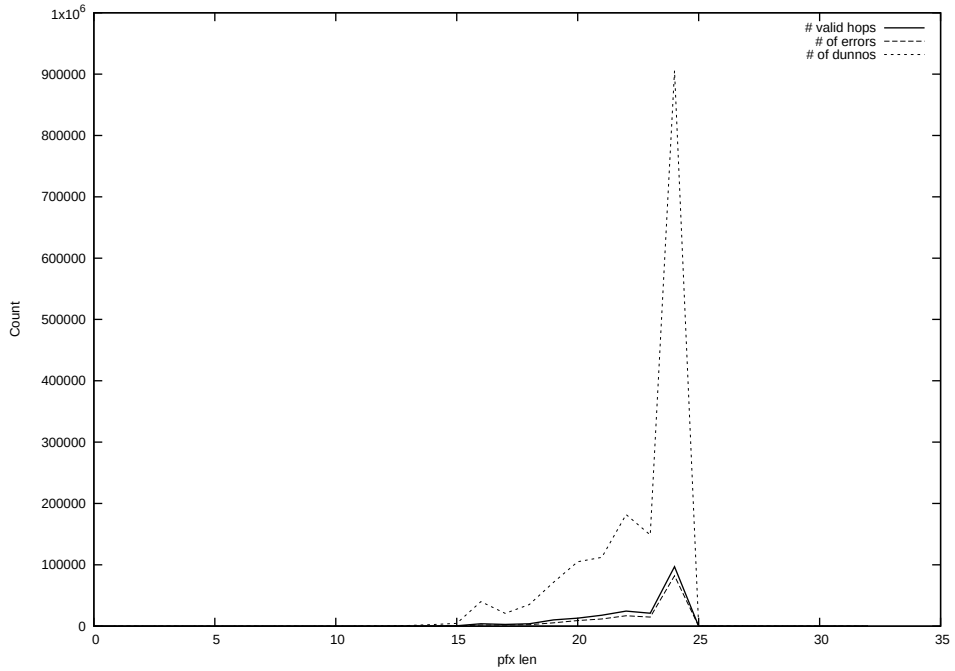
2014-12-31



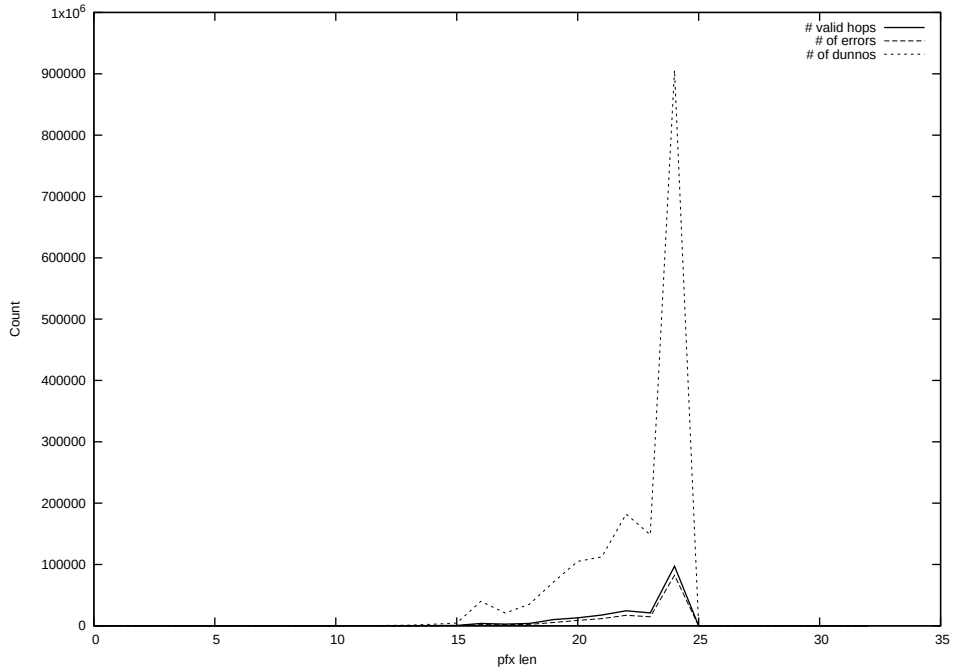
2015-01-01



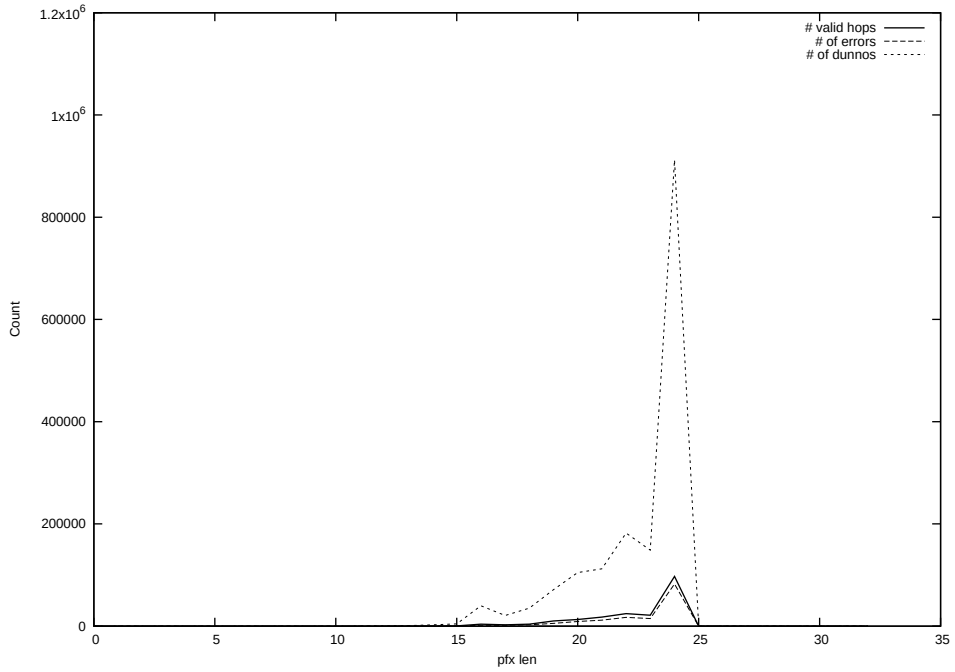
2015-01-02



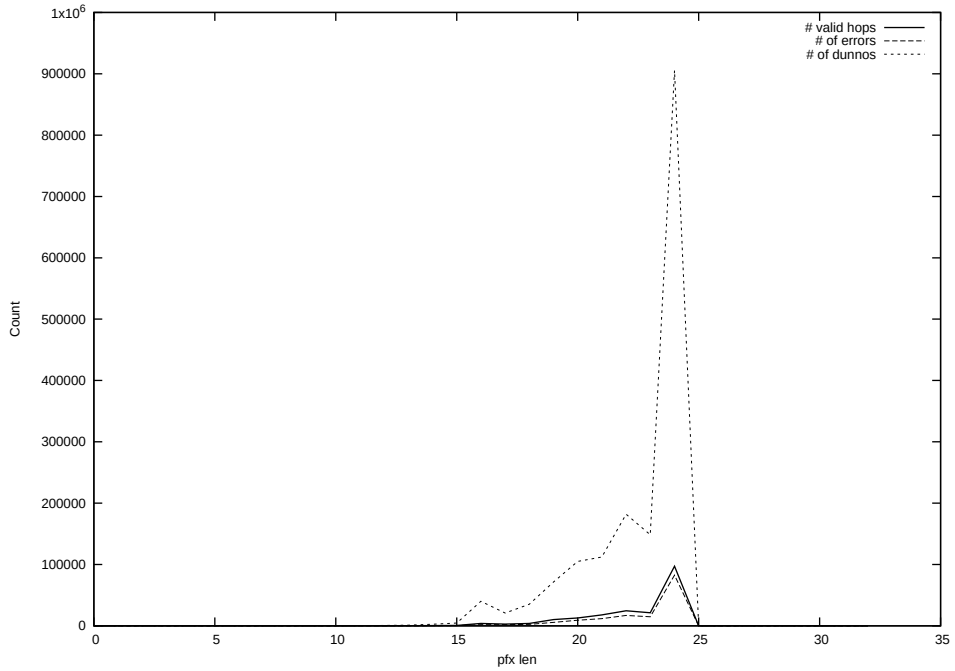
2015-01-03



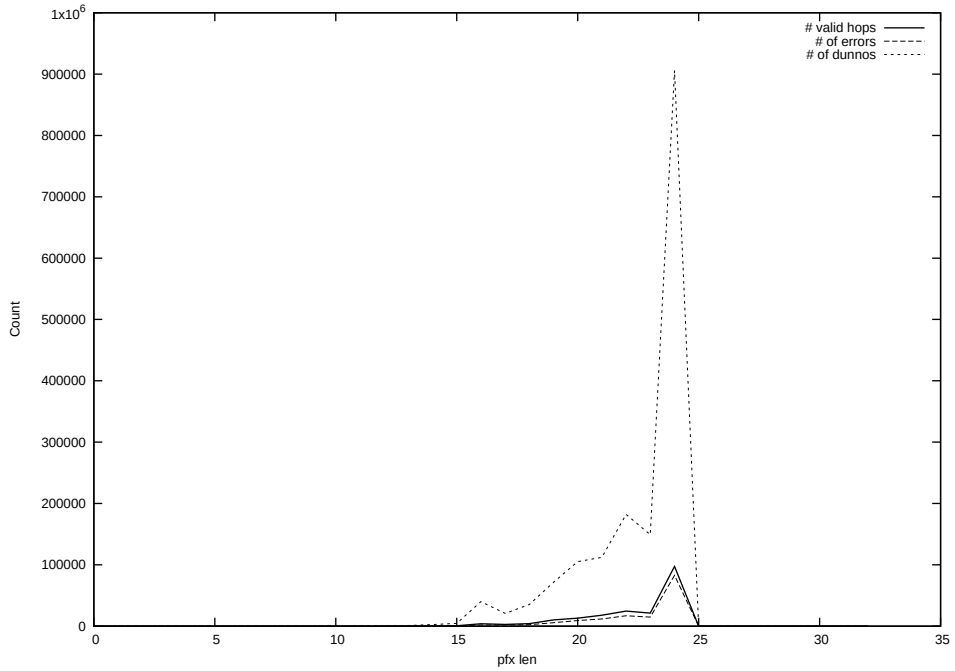
2015-01-04



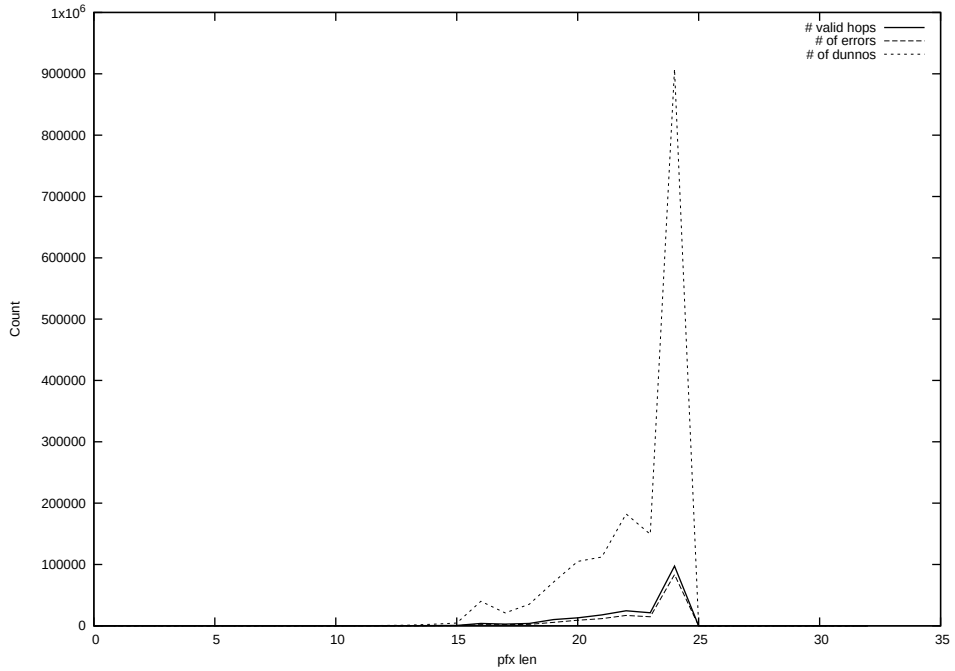
2015-01-05



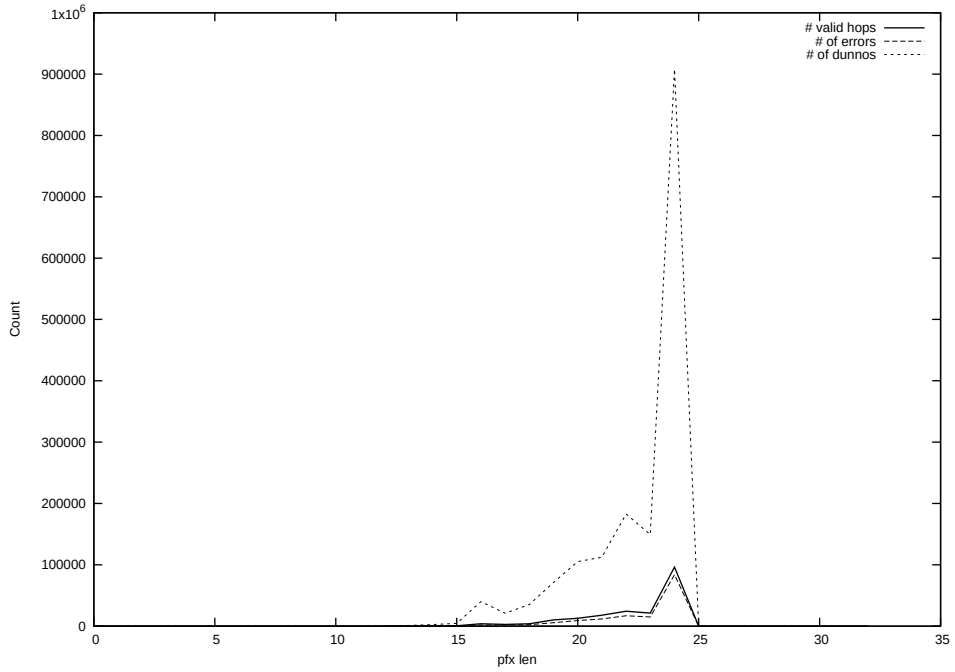
2015-01-06



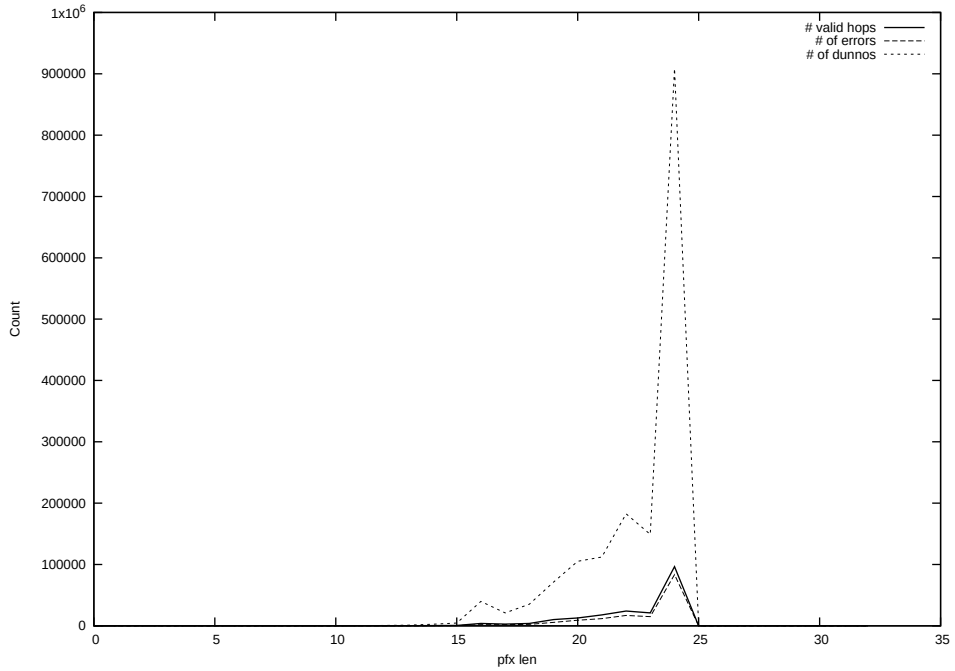
2015-01-07



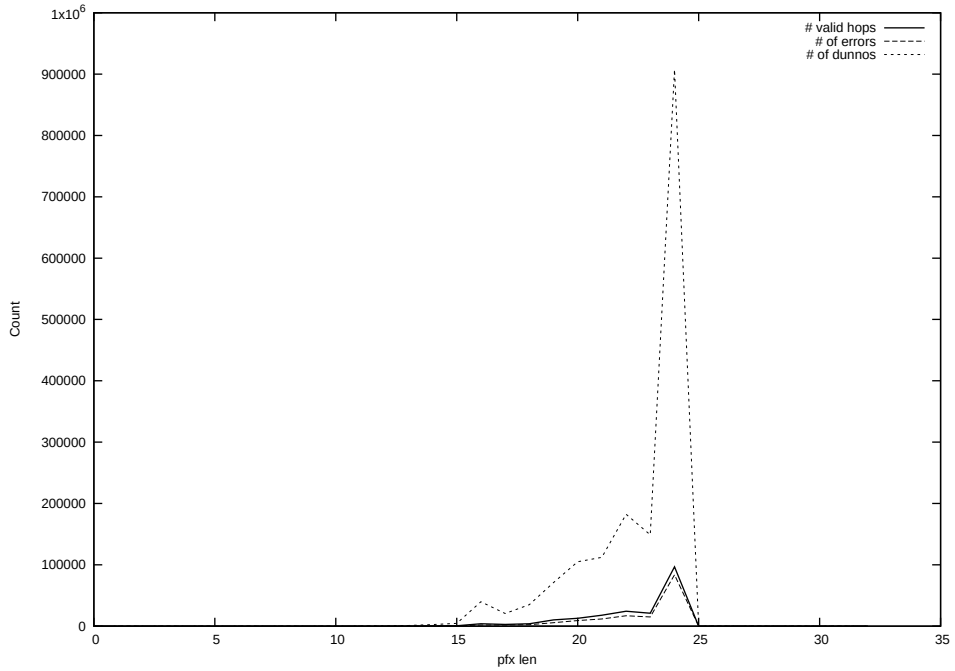
2015-01-08



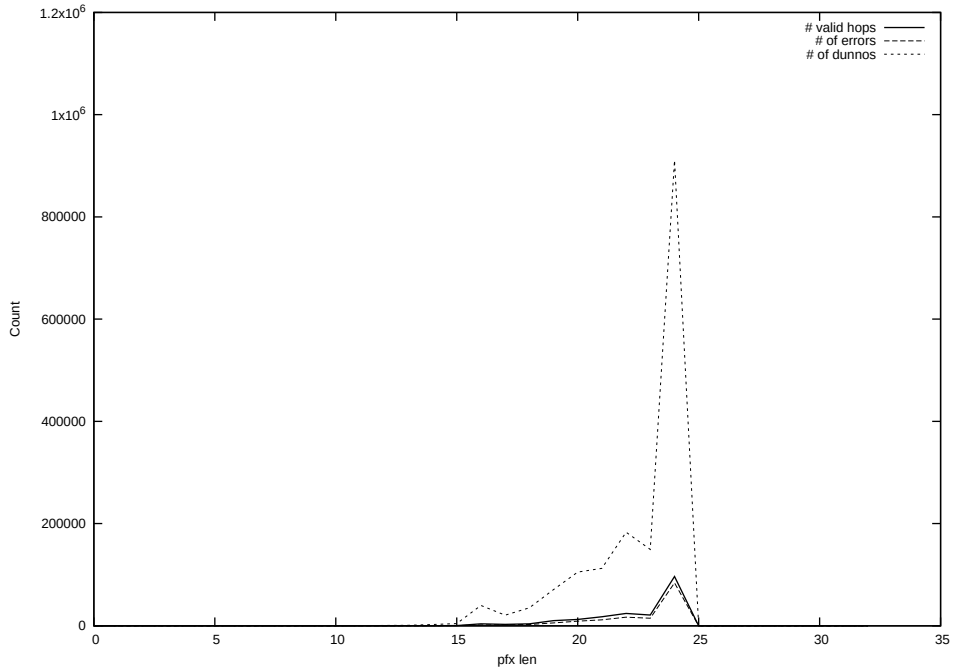
2015-01-09



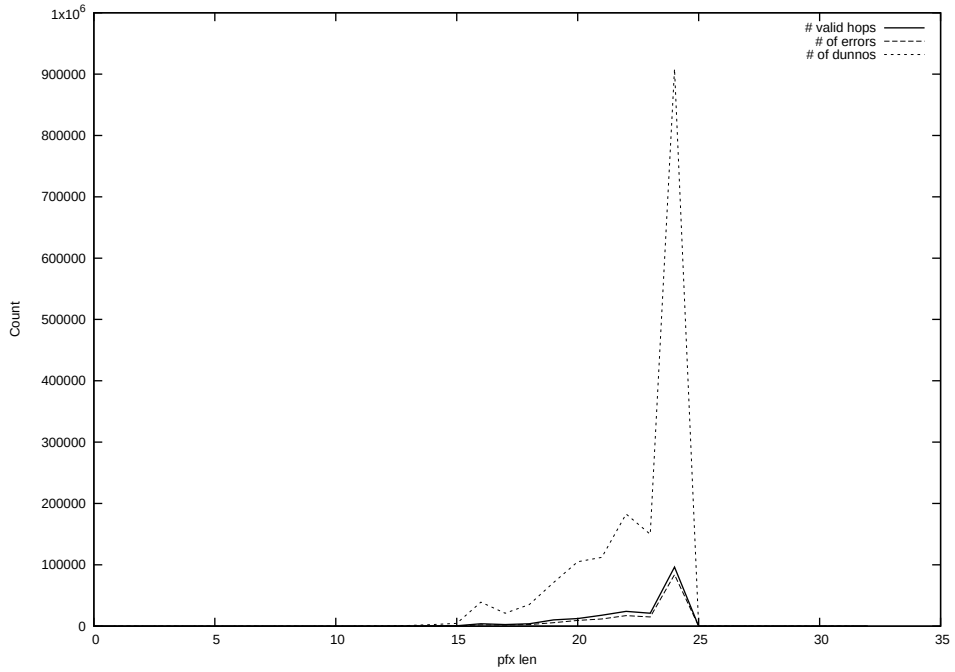
2015-01-10



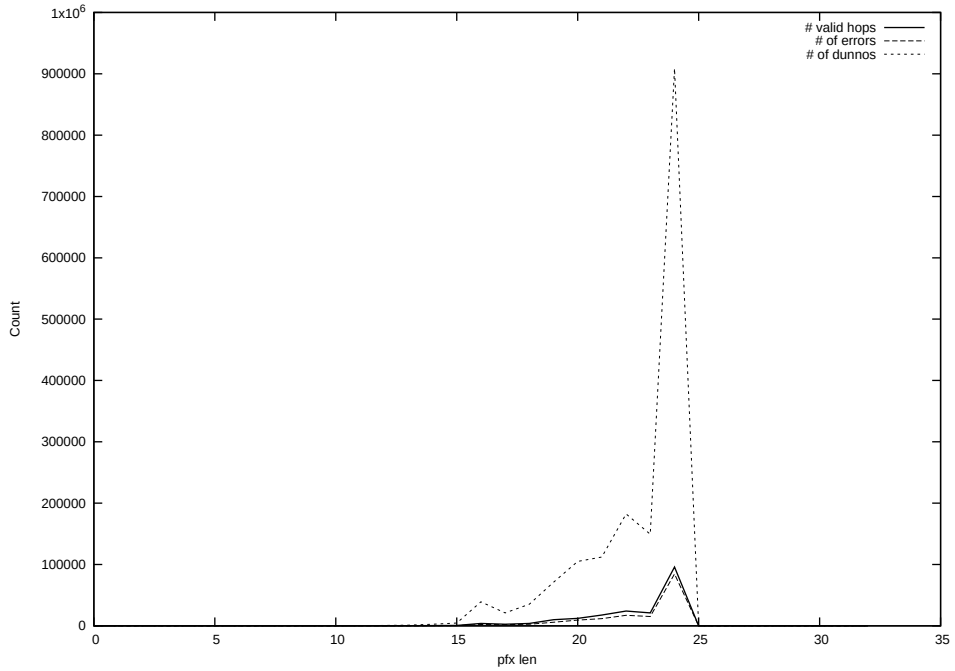
2015-01-11



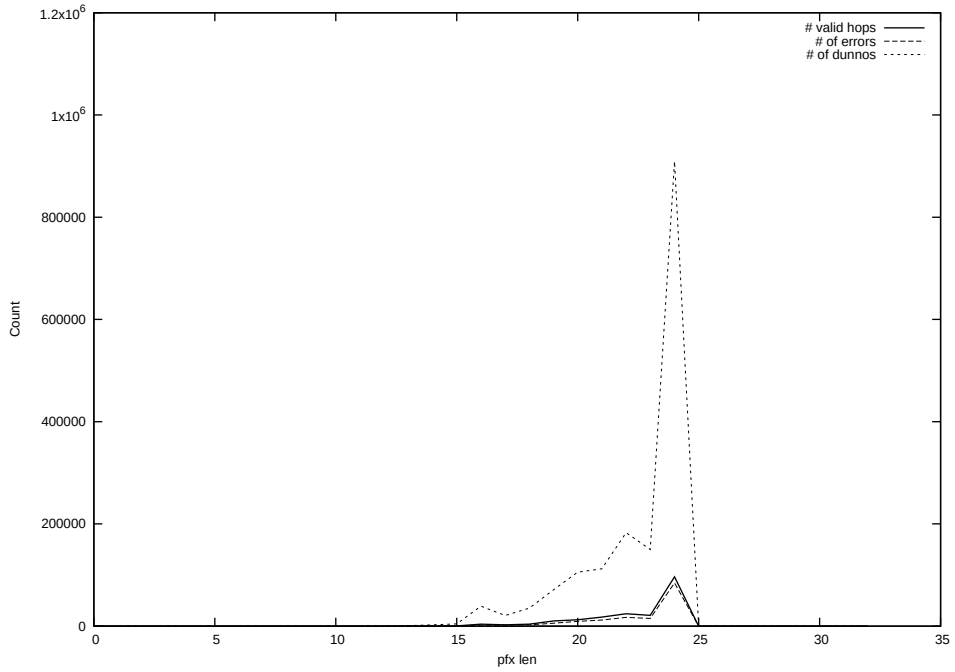
2015-01-12



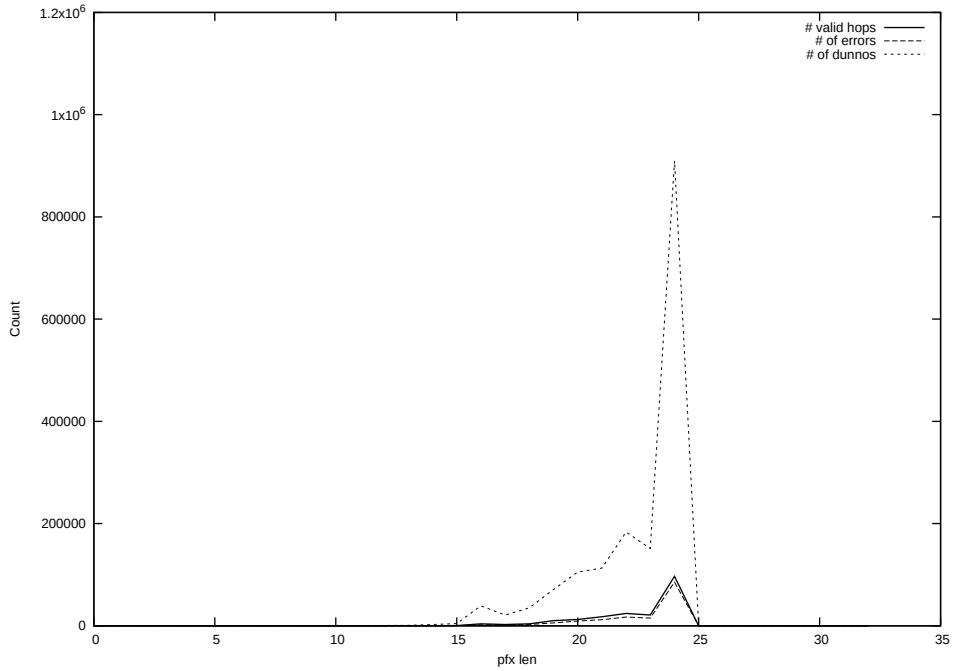
2015-01-13



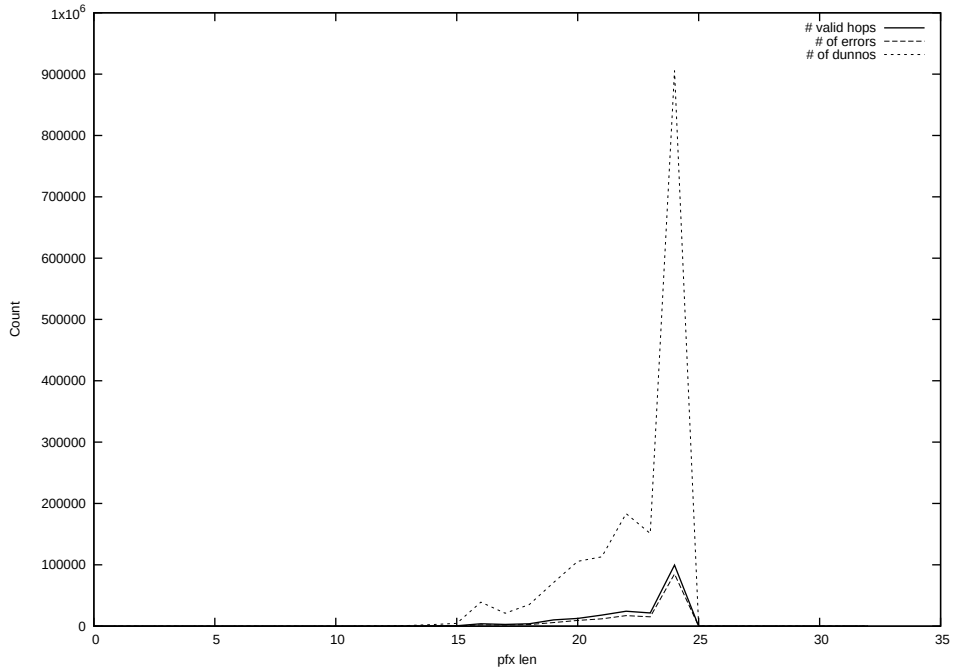
2015-01-14



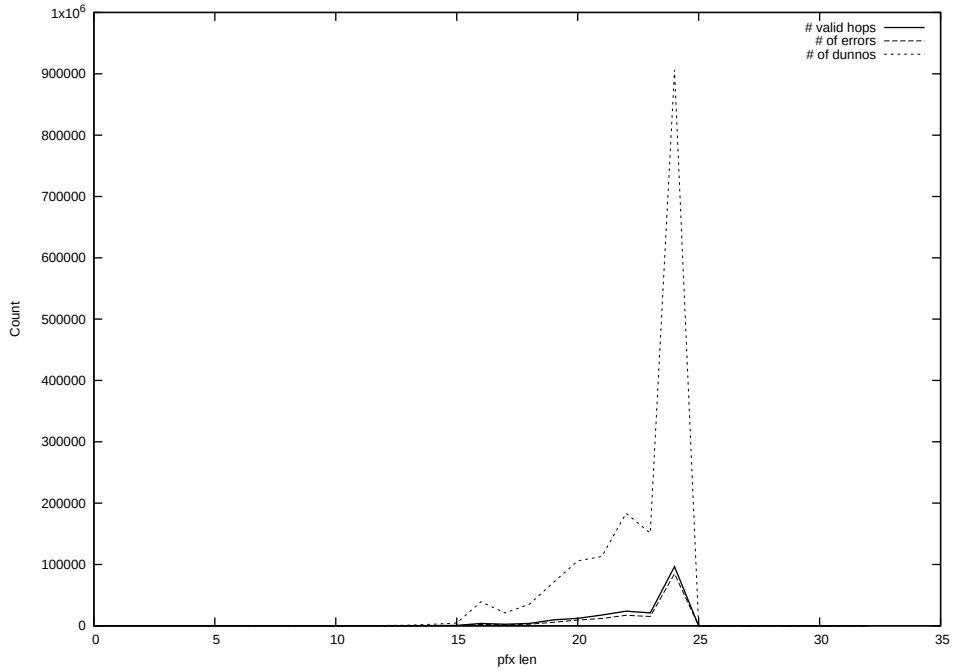
2015-01-15



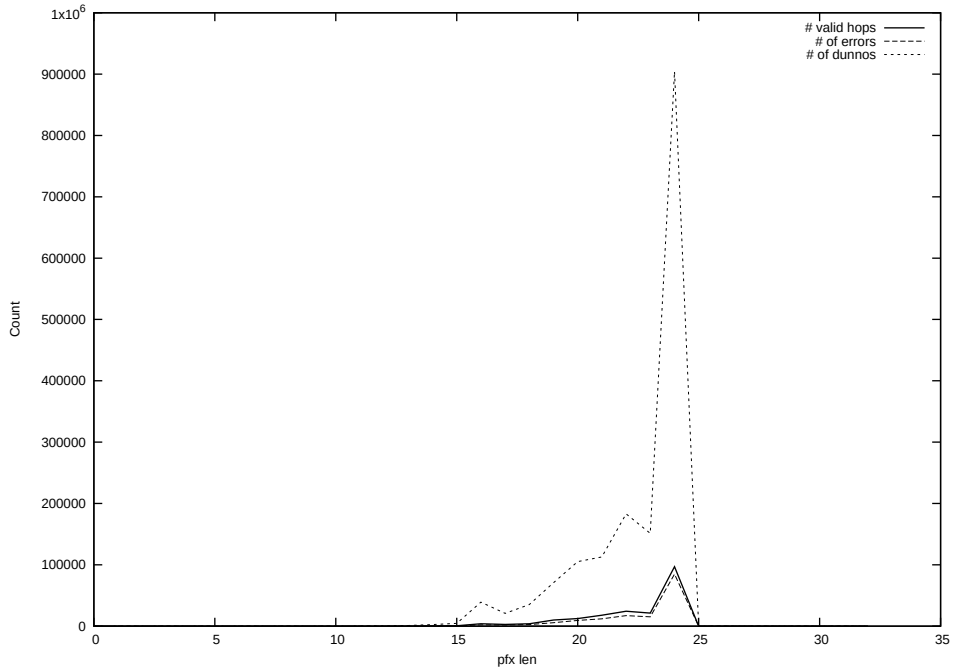
2015-01-16



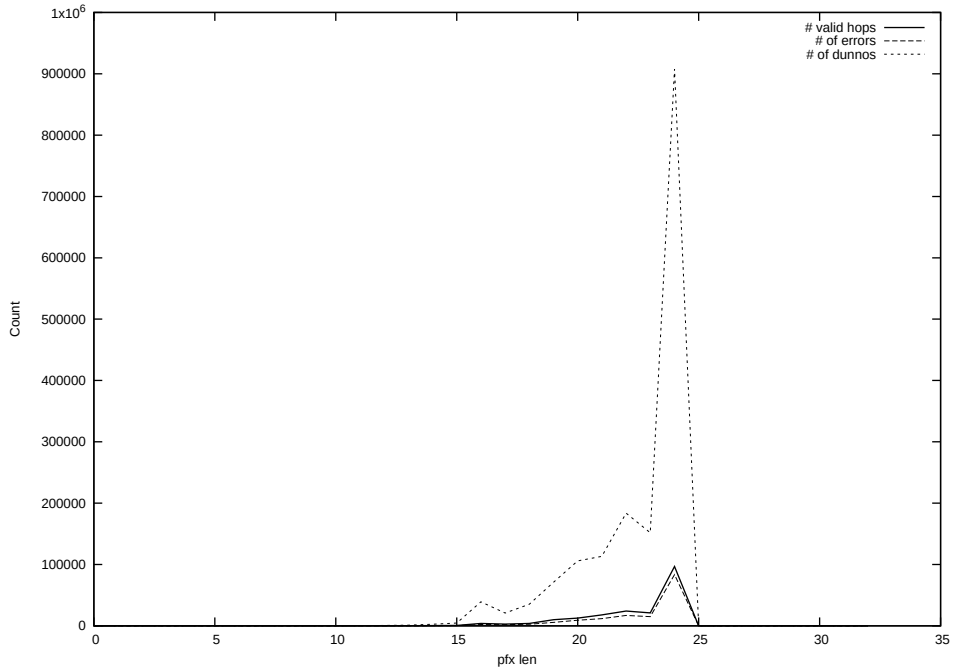
2015-01-17



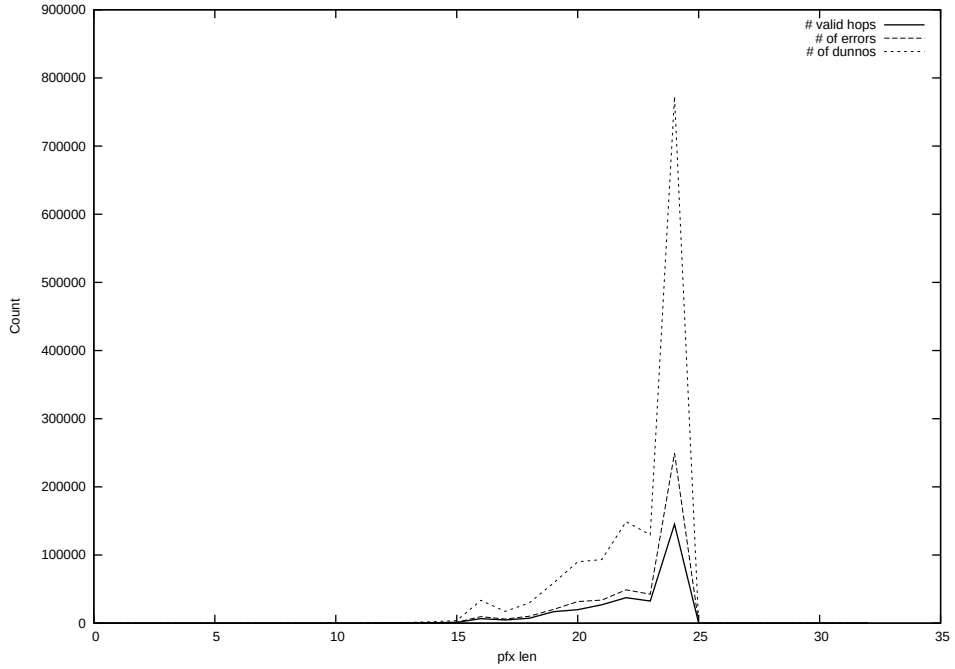
2015-01-18



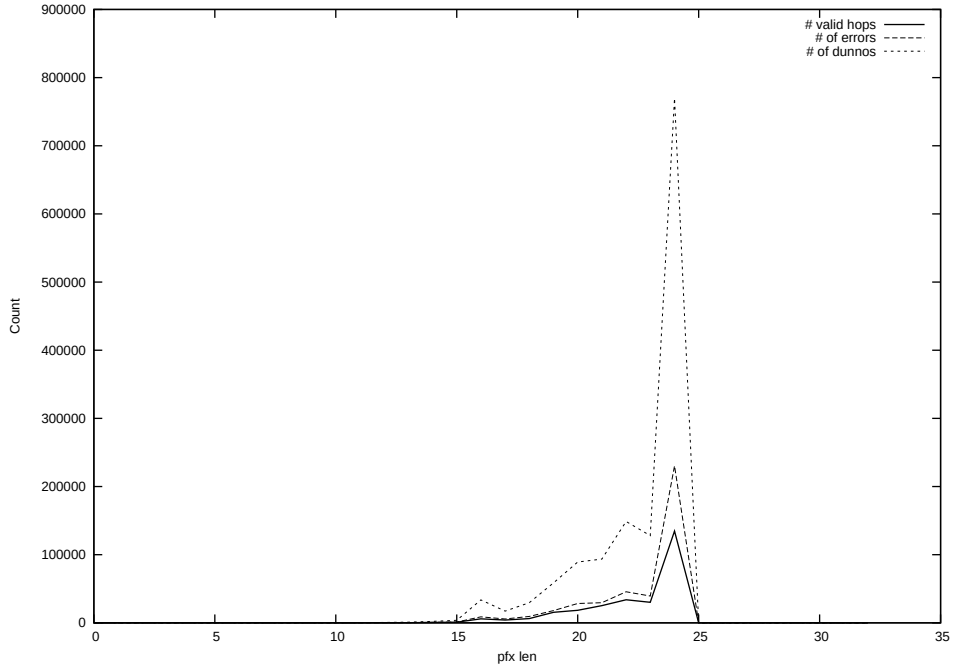
2015-01-19



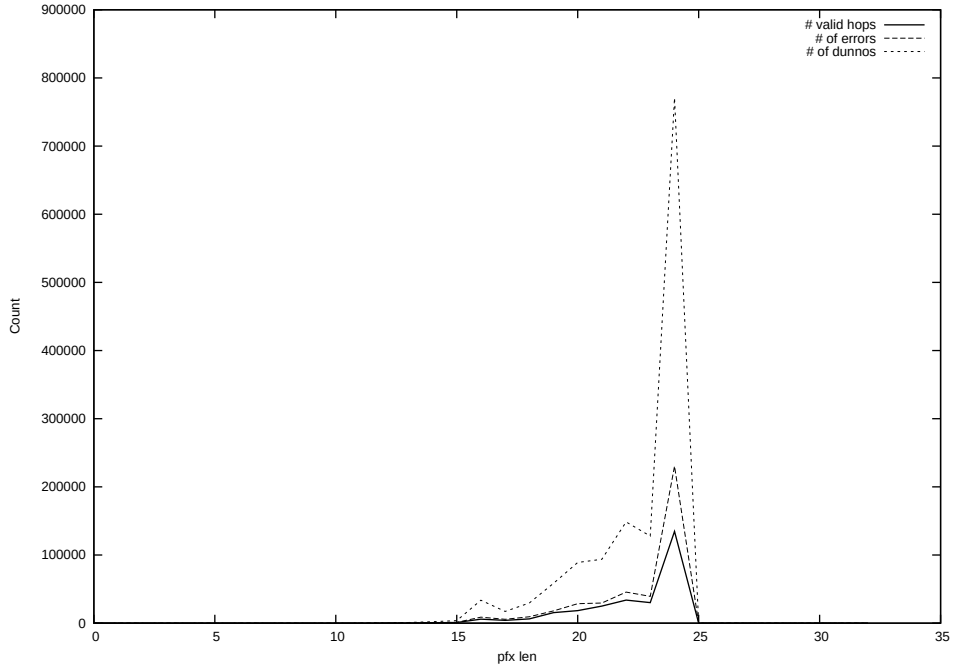
2015-01-20



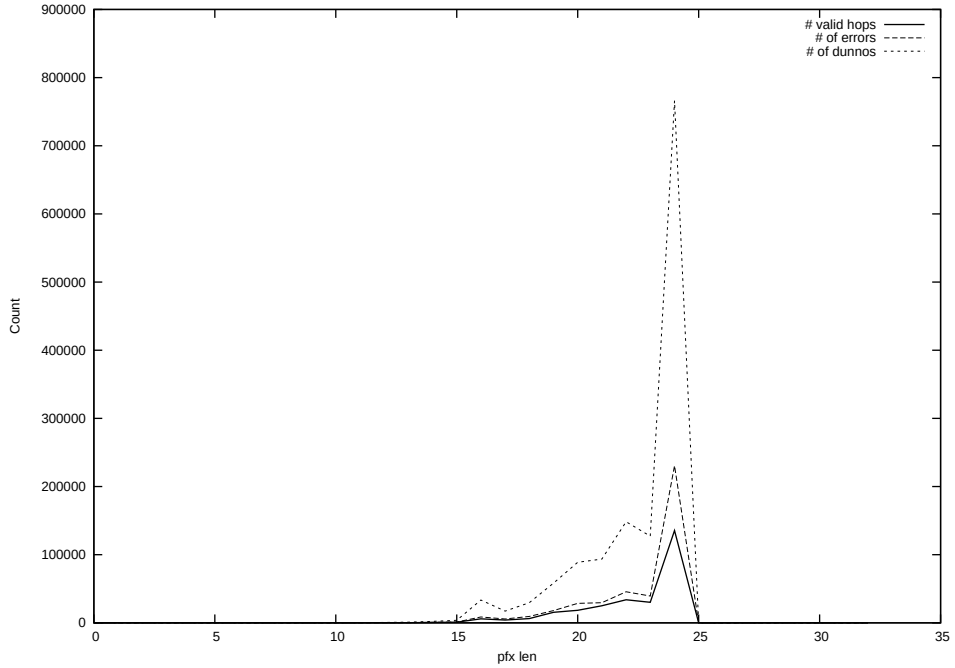
2015-01-21



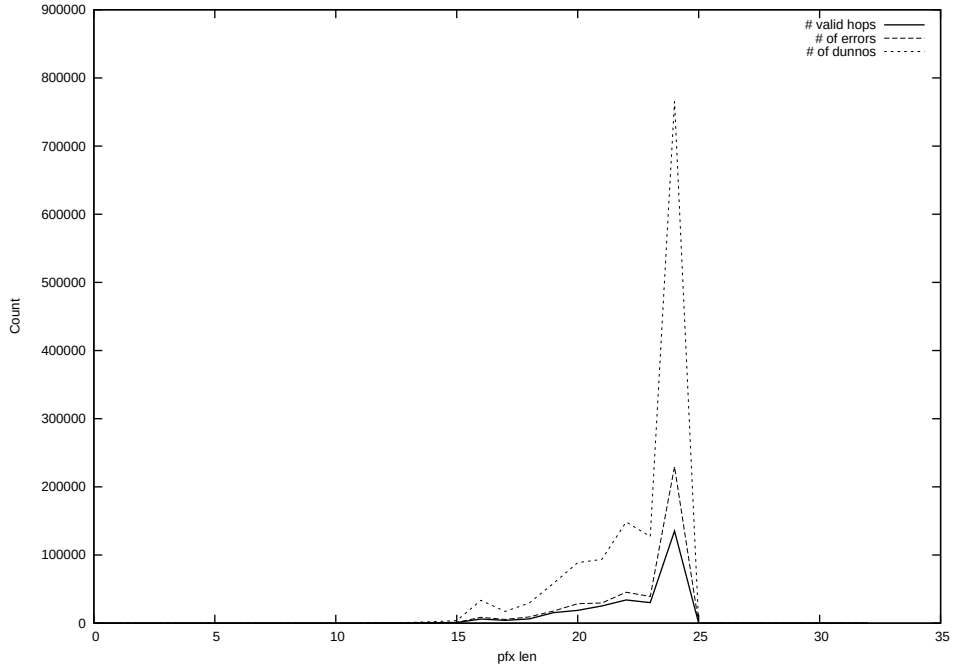
2015-01-22



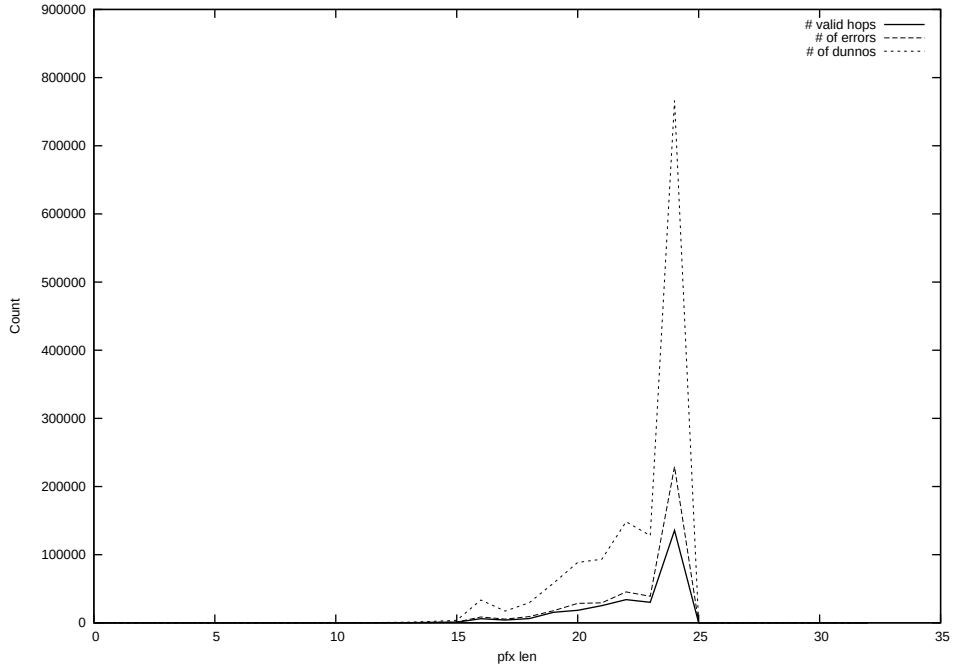
2015-01-23



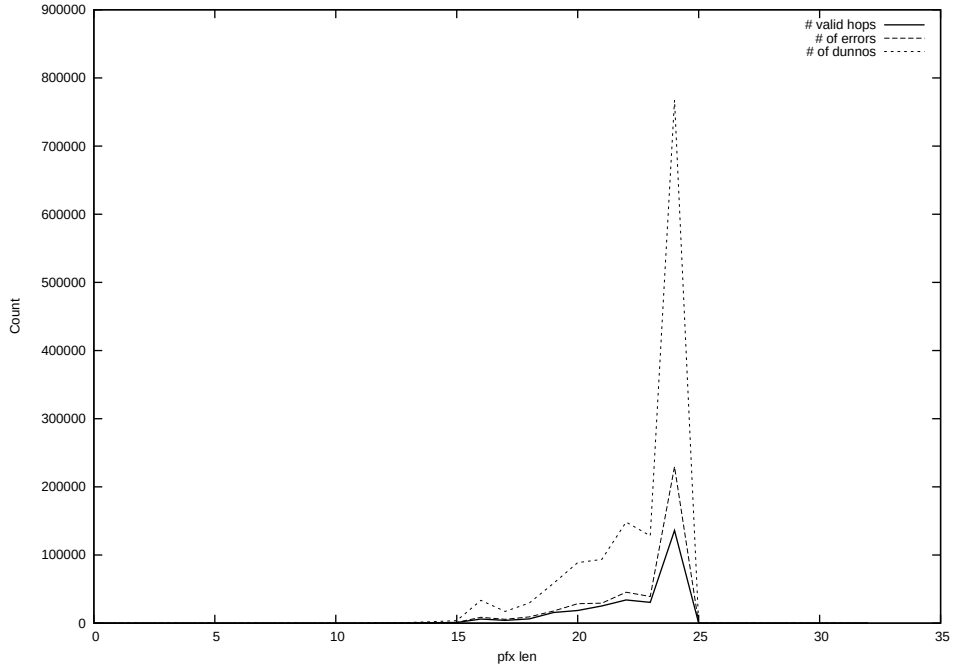
2015-01-24



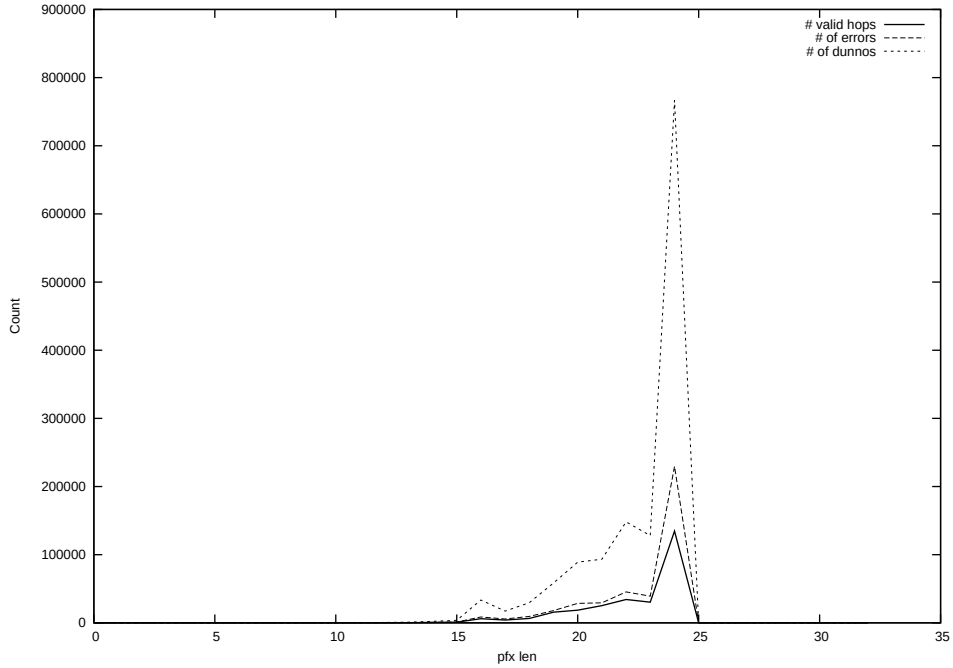
2015-01-25



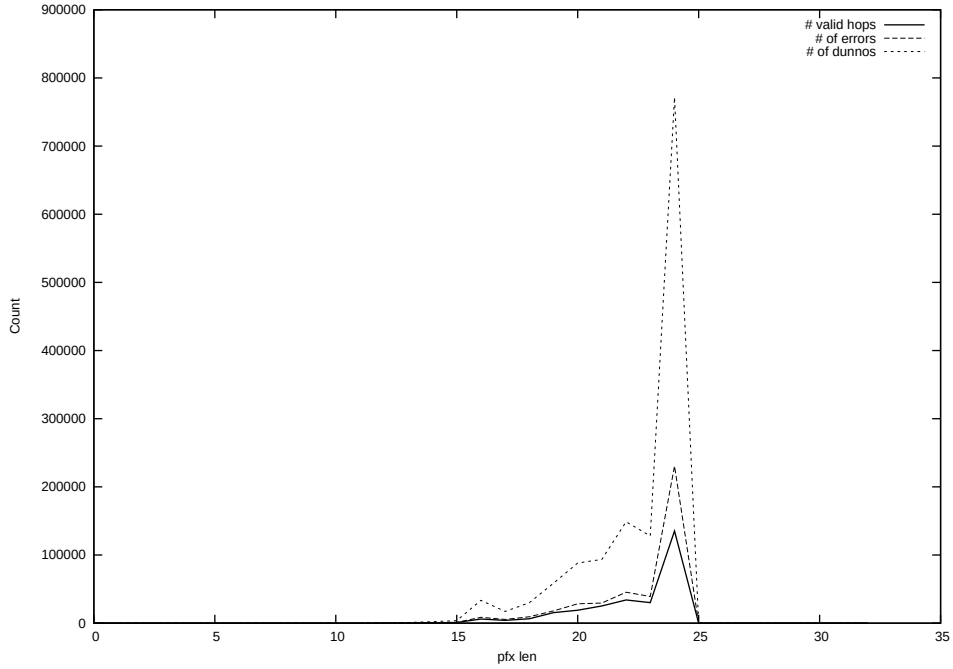
2015-01-26



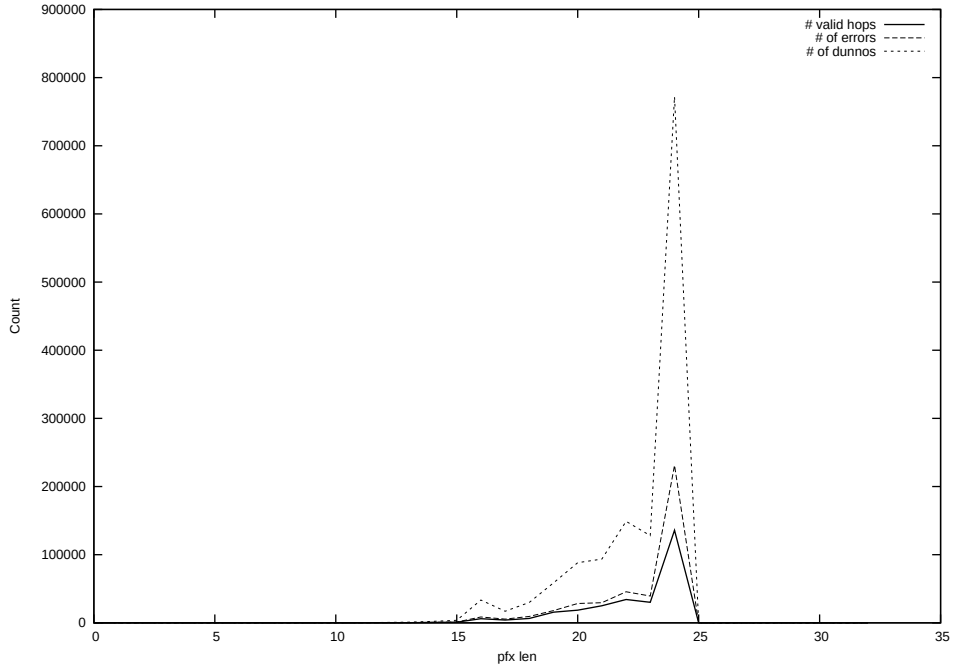
2015-01-27



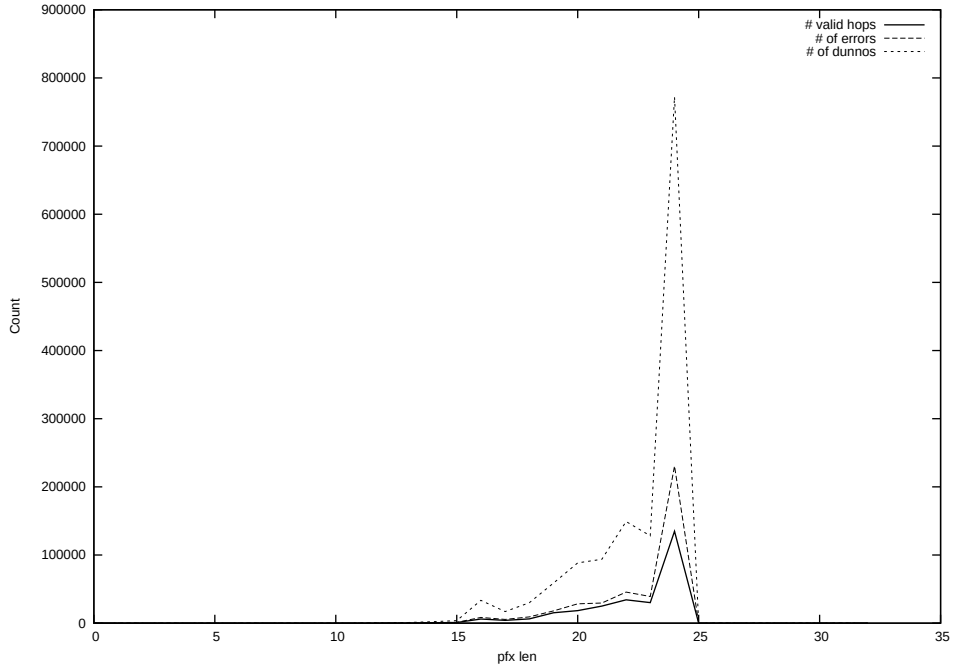
2015-01-28



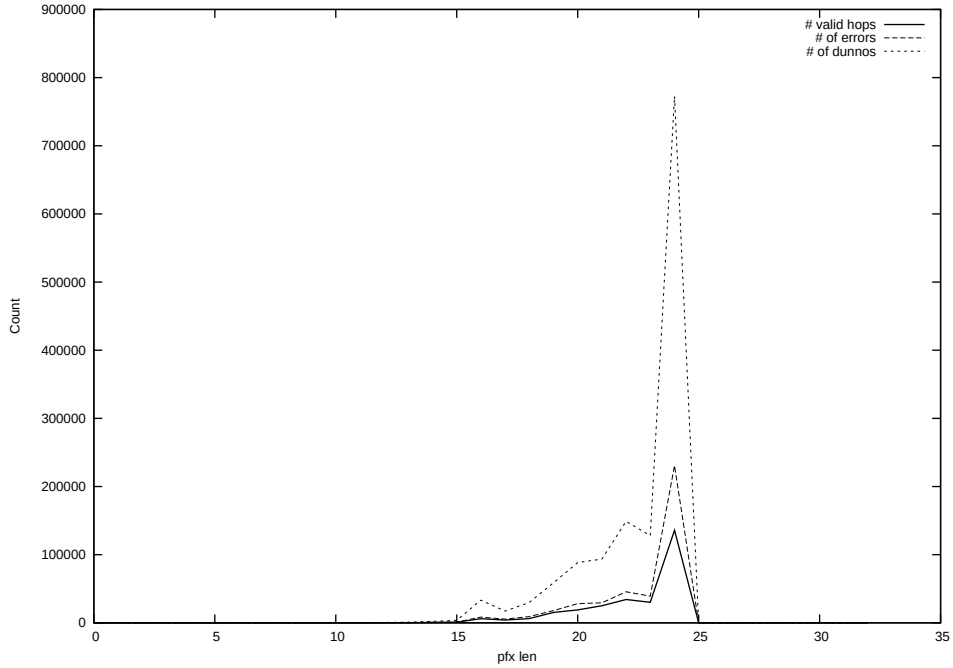
2015-01-29



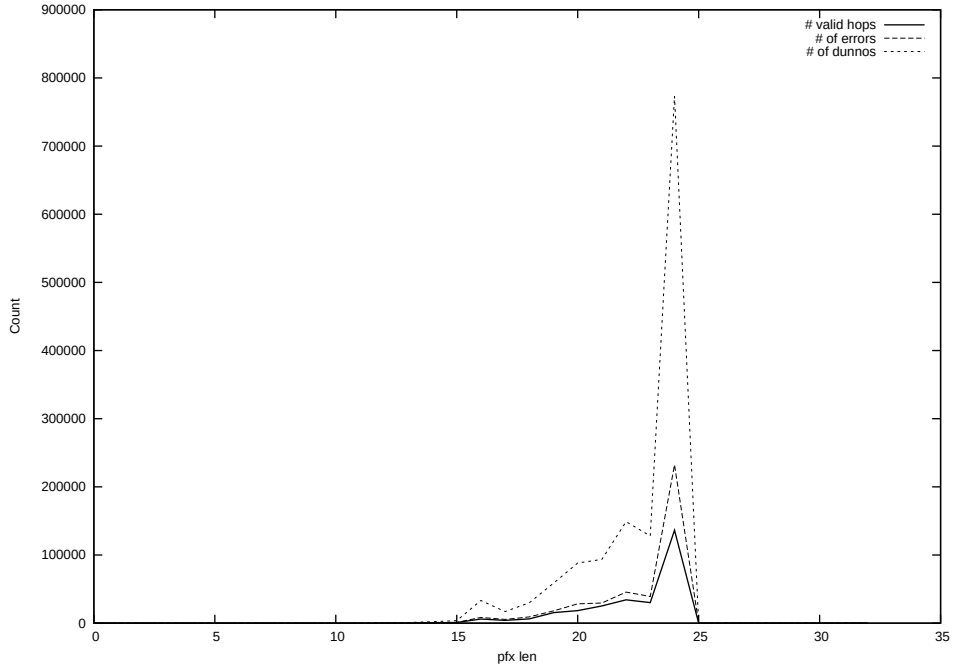
2015-01-30



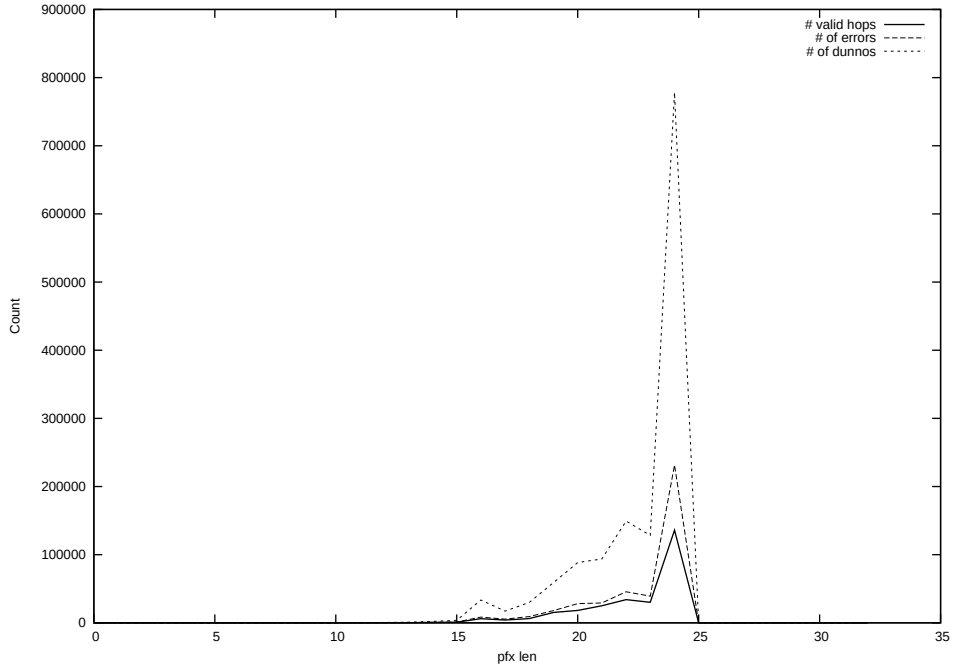
2015-01-31



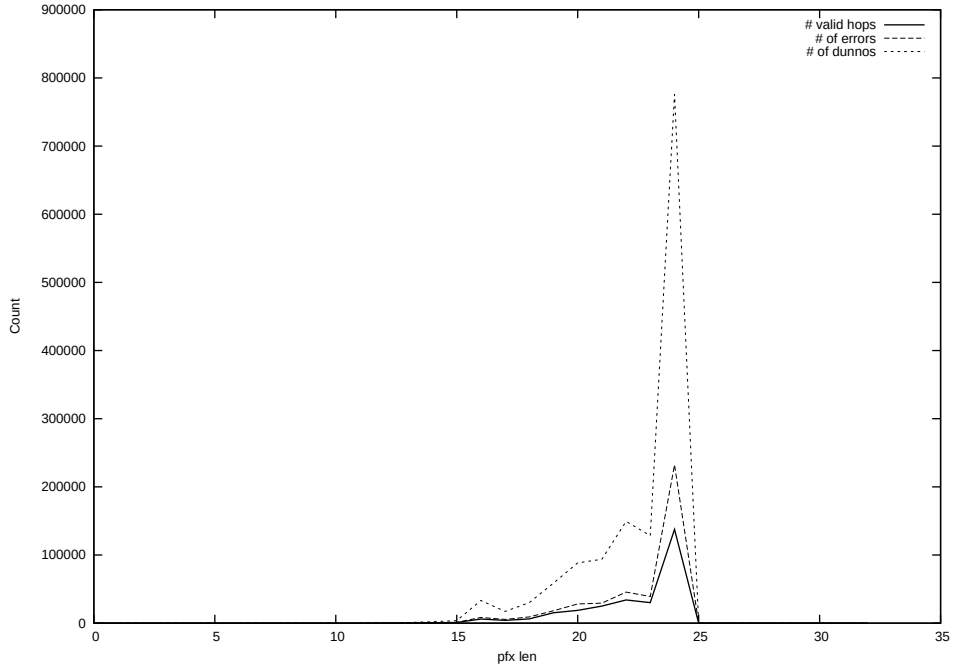
2015-02-01



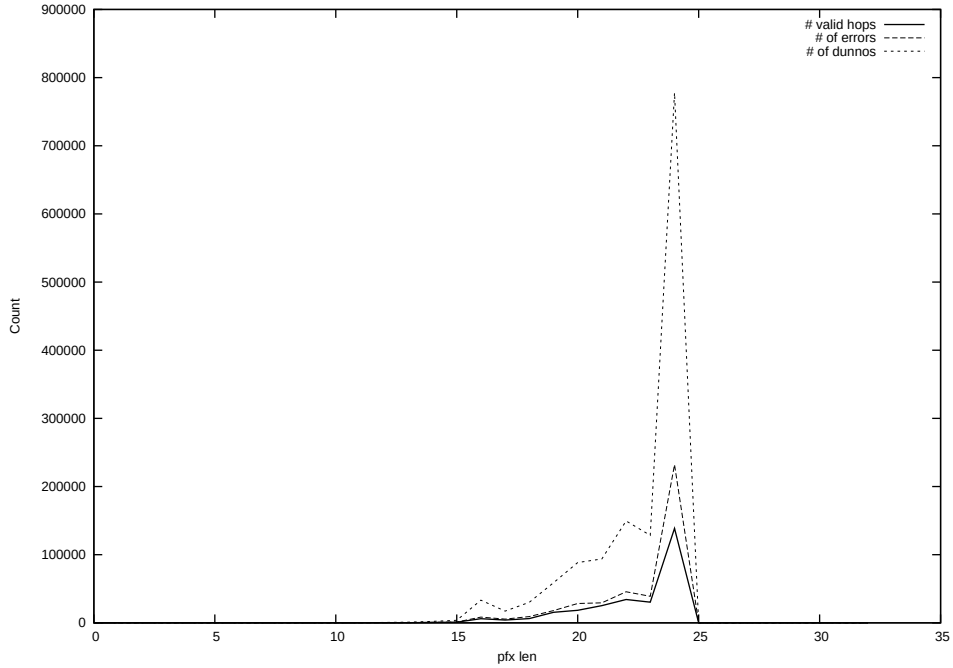
2015-02-02



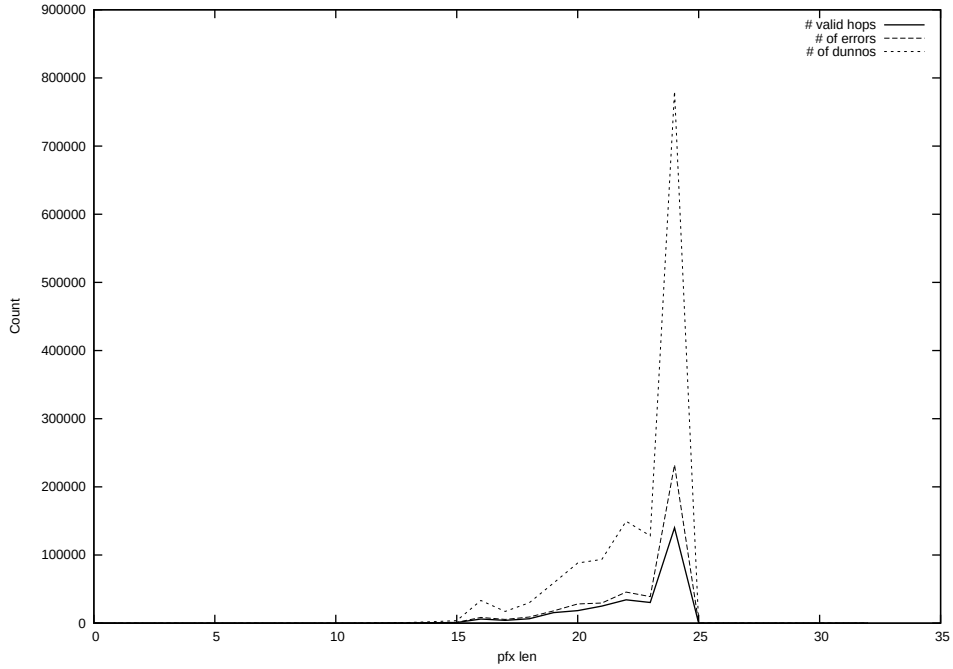
2015-02-03



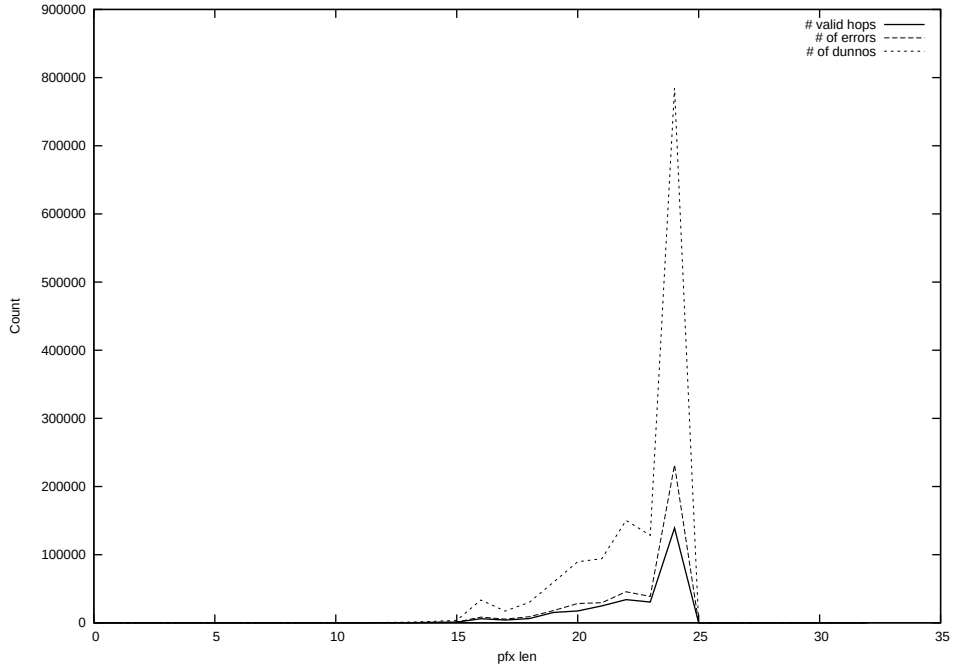
2015-02-04



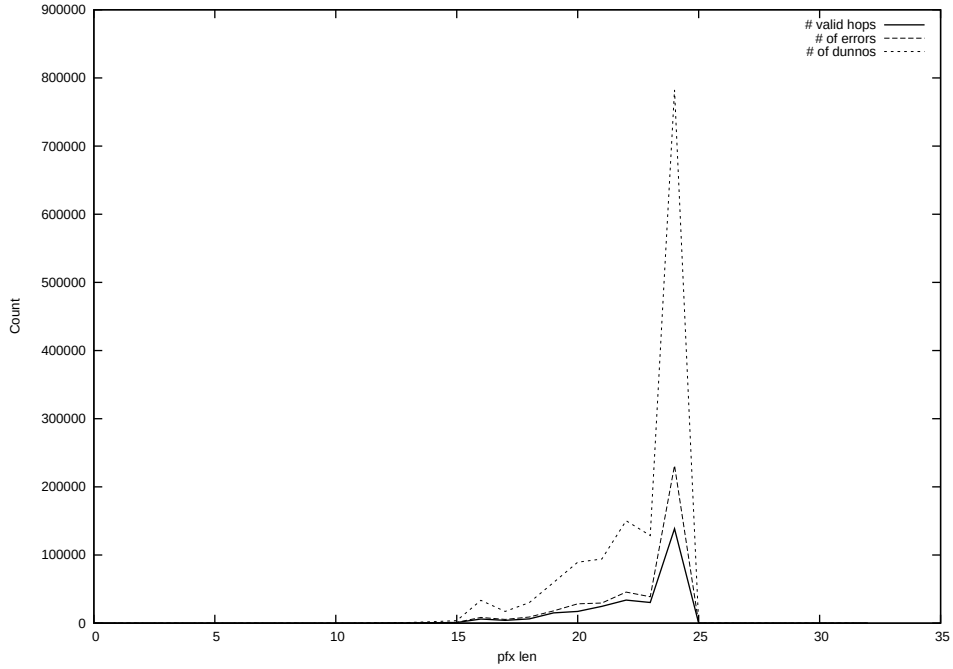
2015-02-05



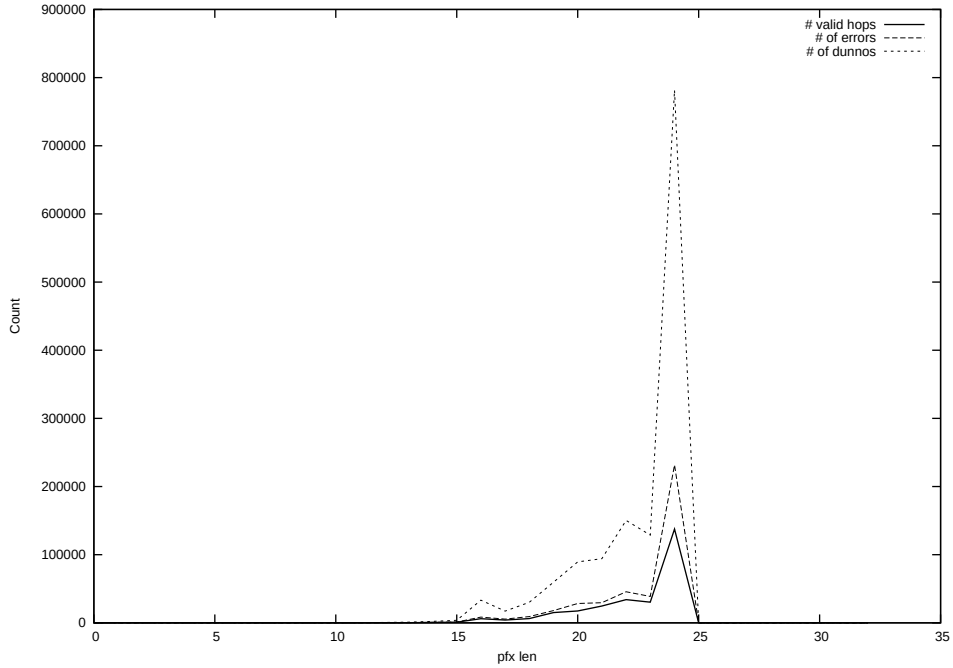
2015-02-06



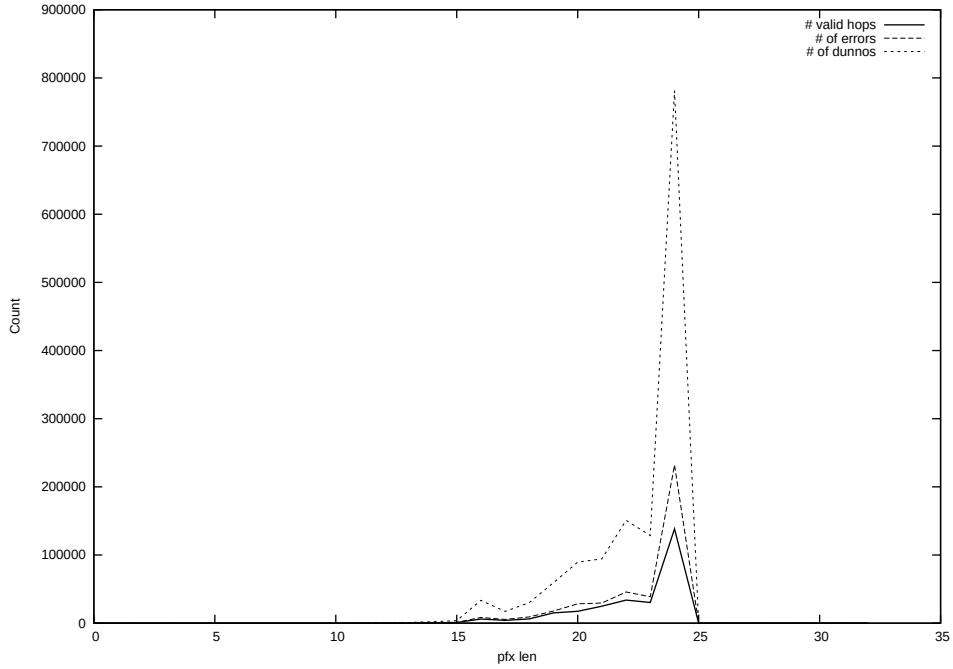
2015-02-07



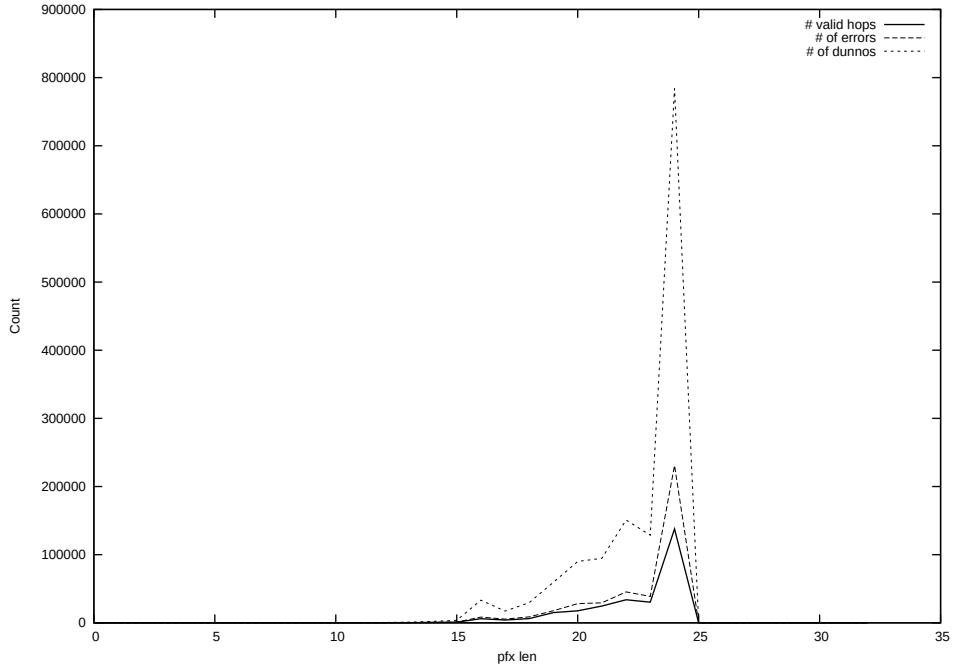
2015-02-08



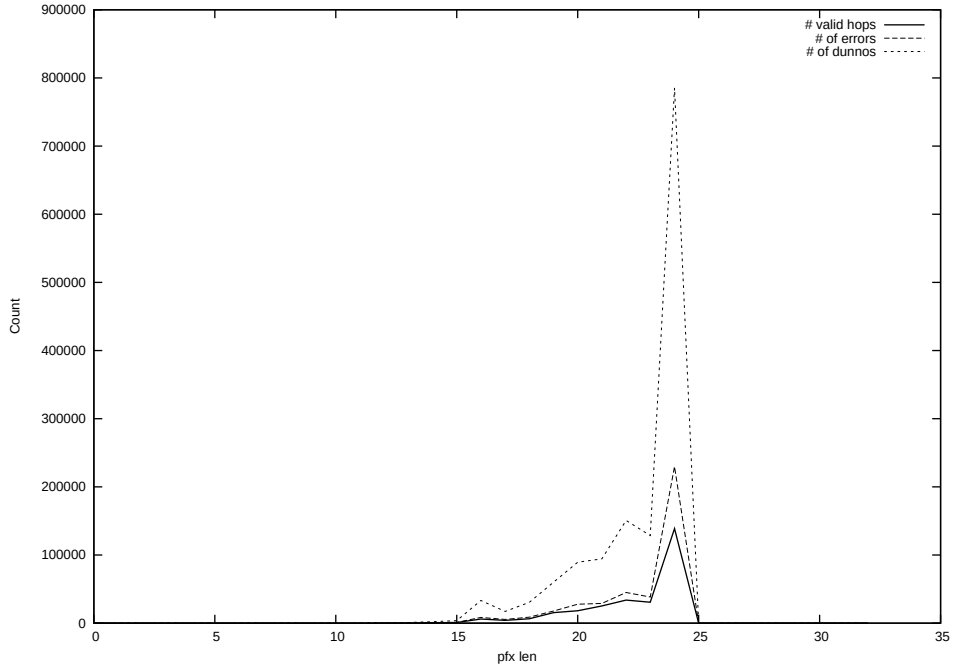
2015-02-09



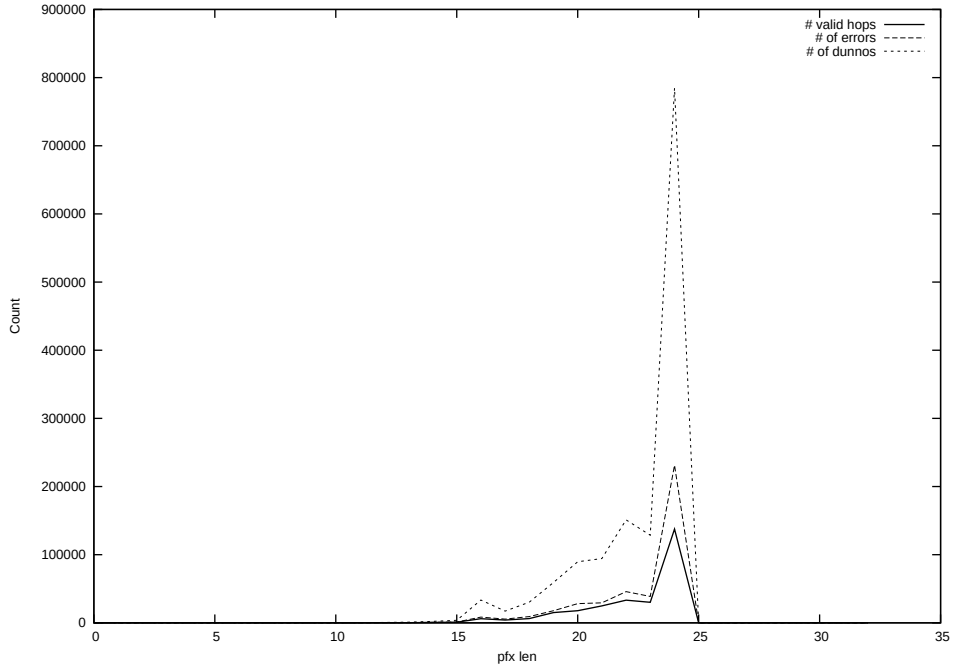
2015-02-10



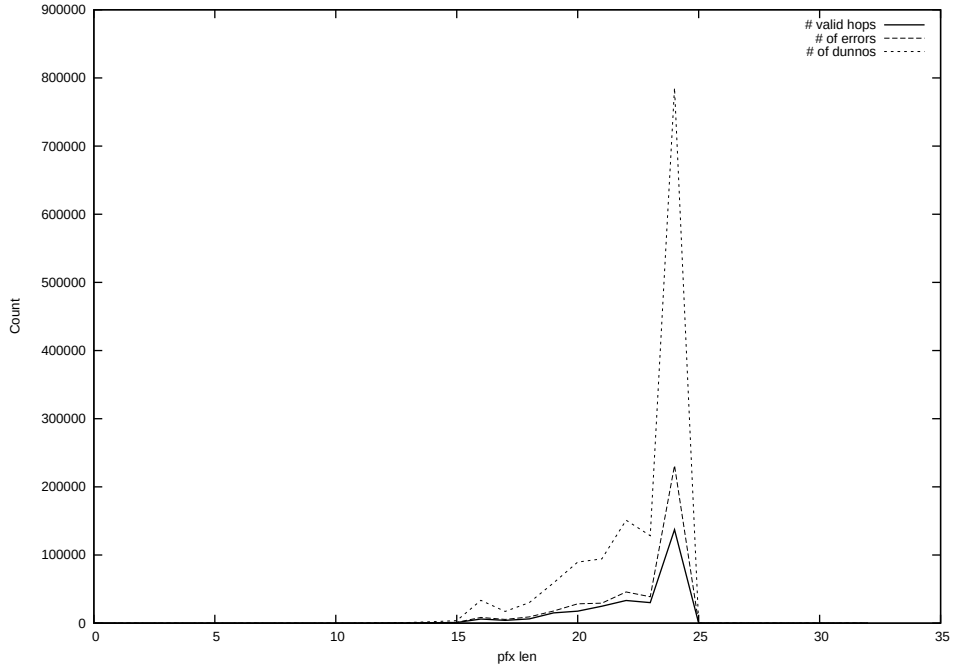
2015-02-11



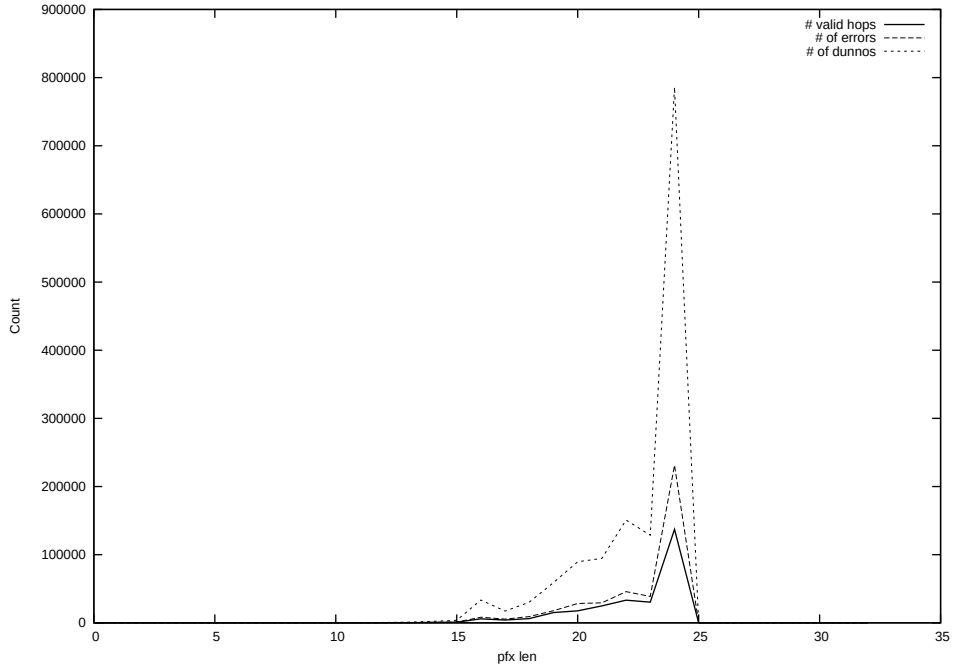
2015-02-12



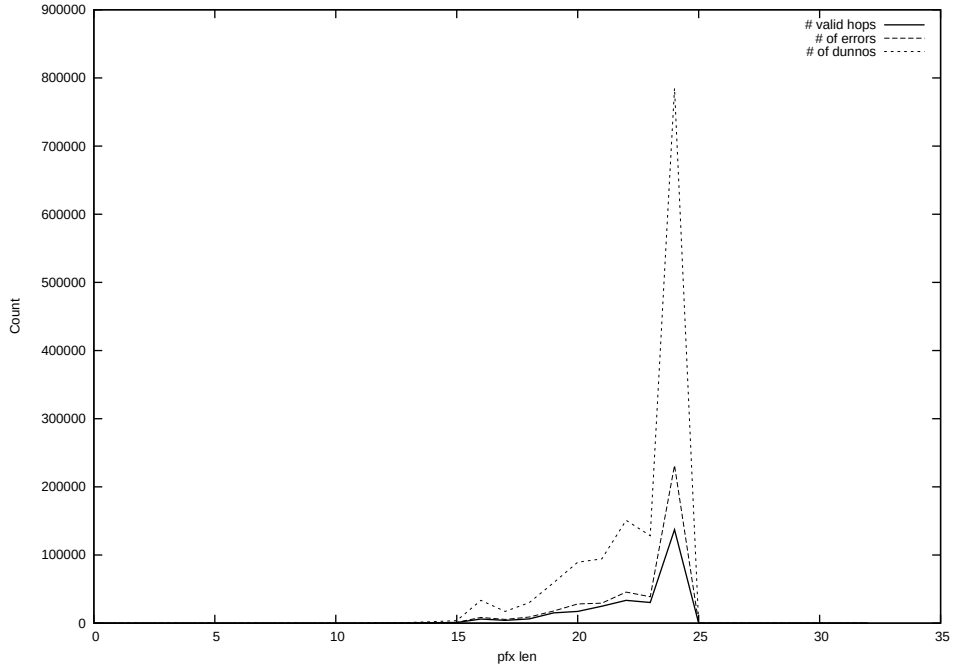
2015-02-13



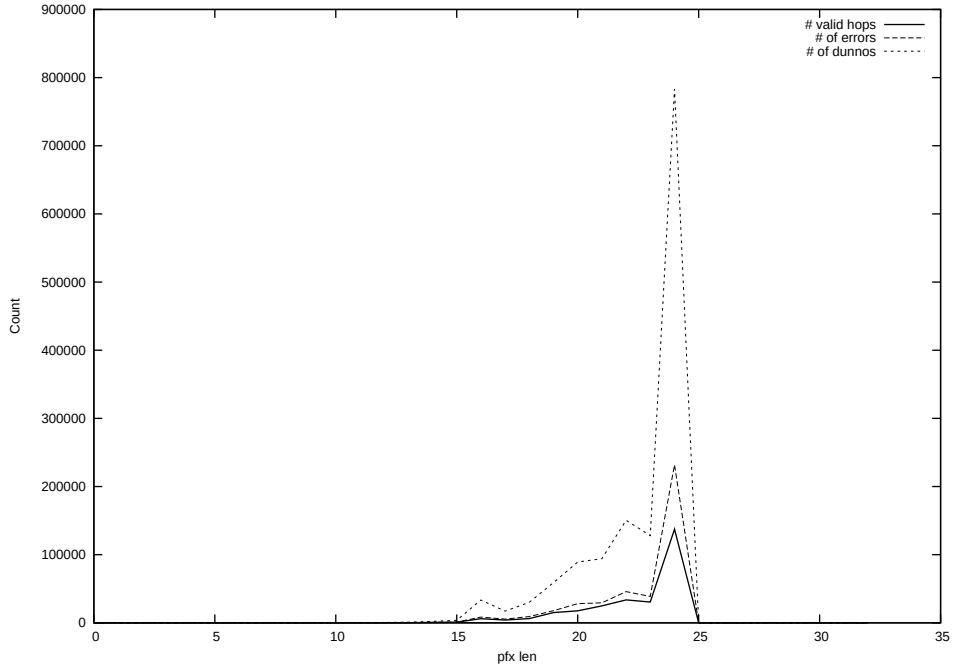
2015-02-14



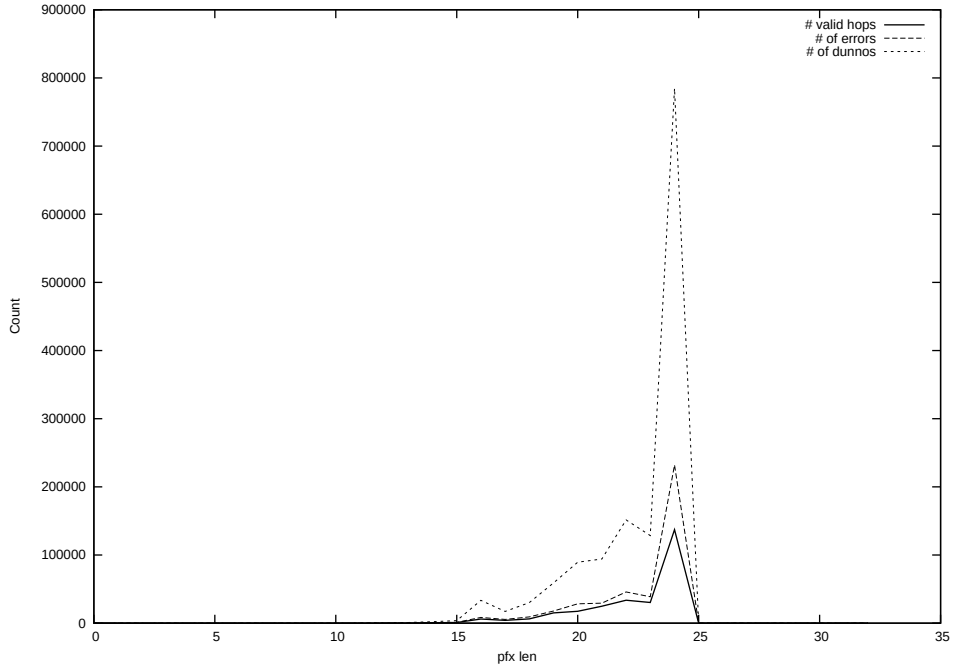
2015-02-15



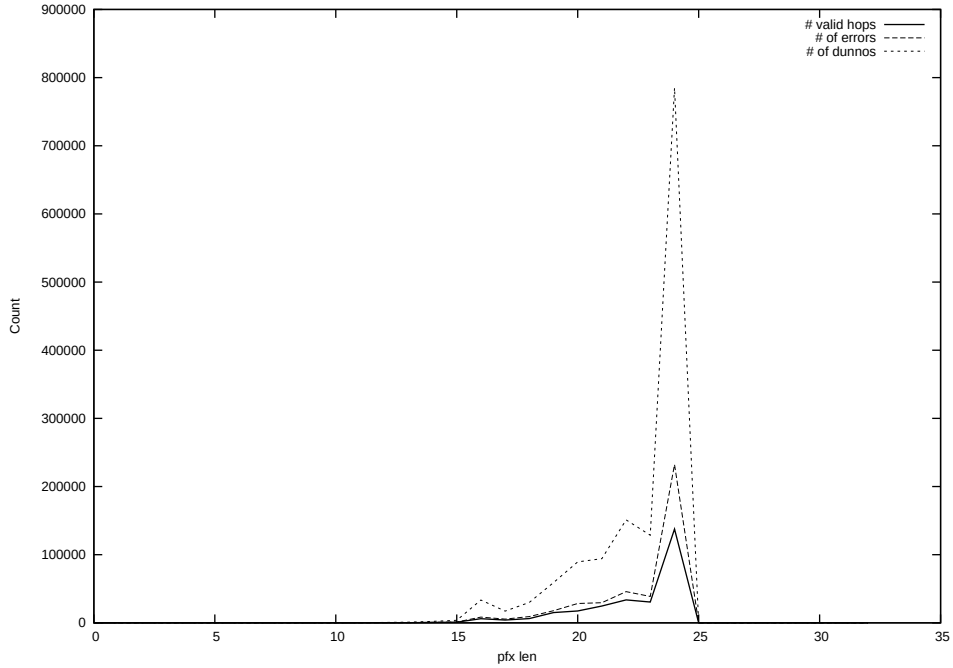
2015-02-16



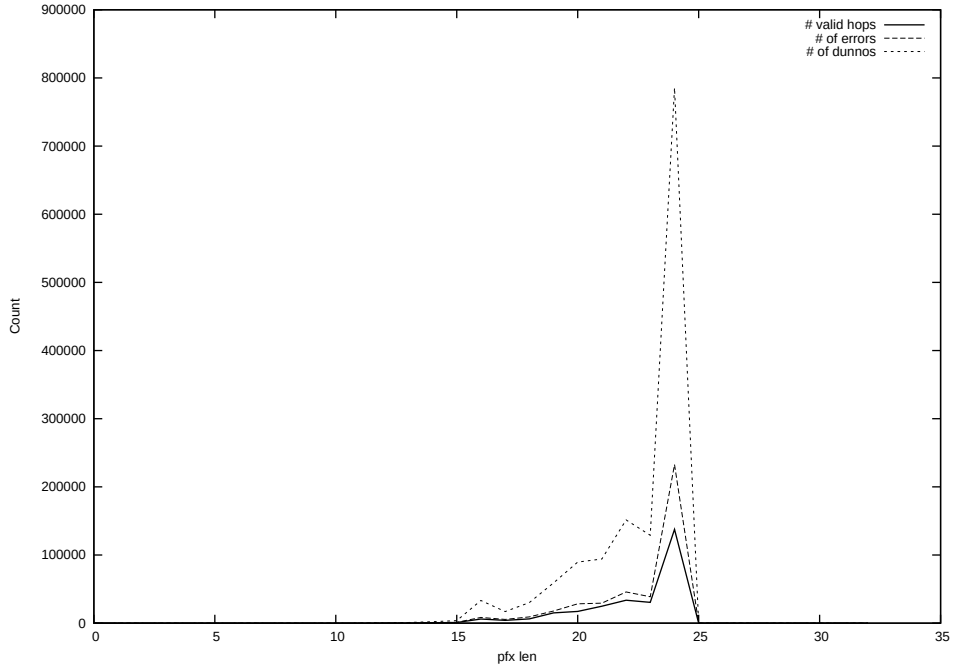
2015-02-17



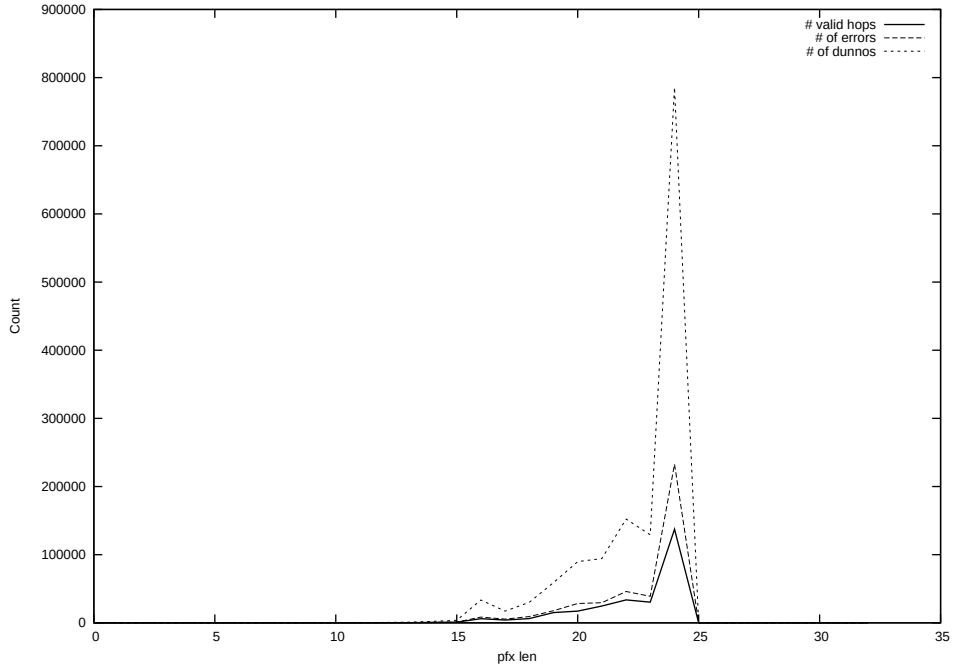
2015-02-18



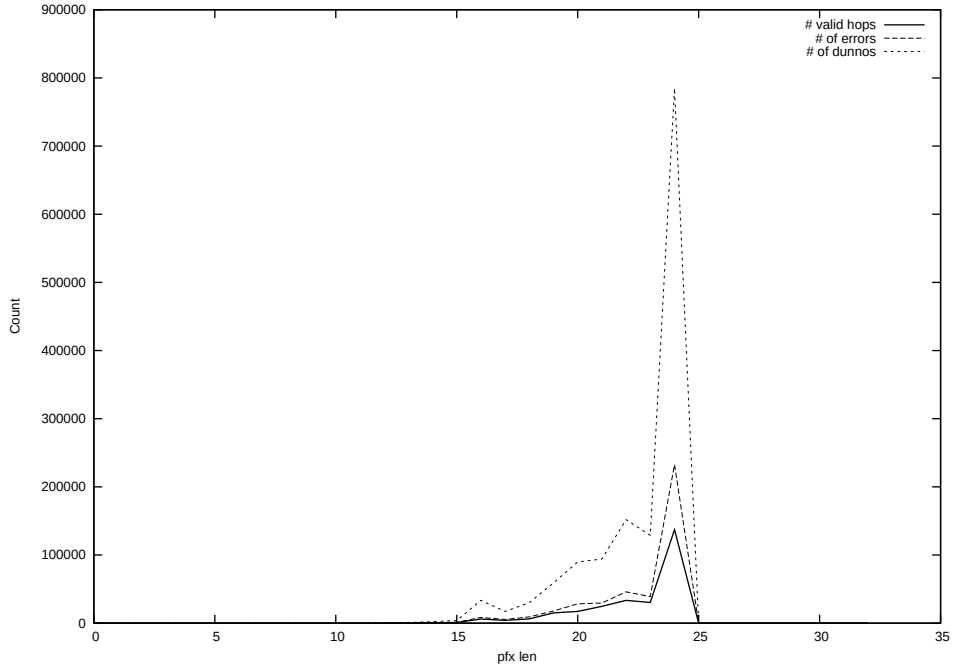
2015-02-19



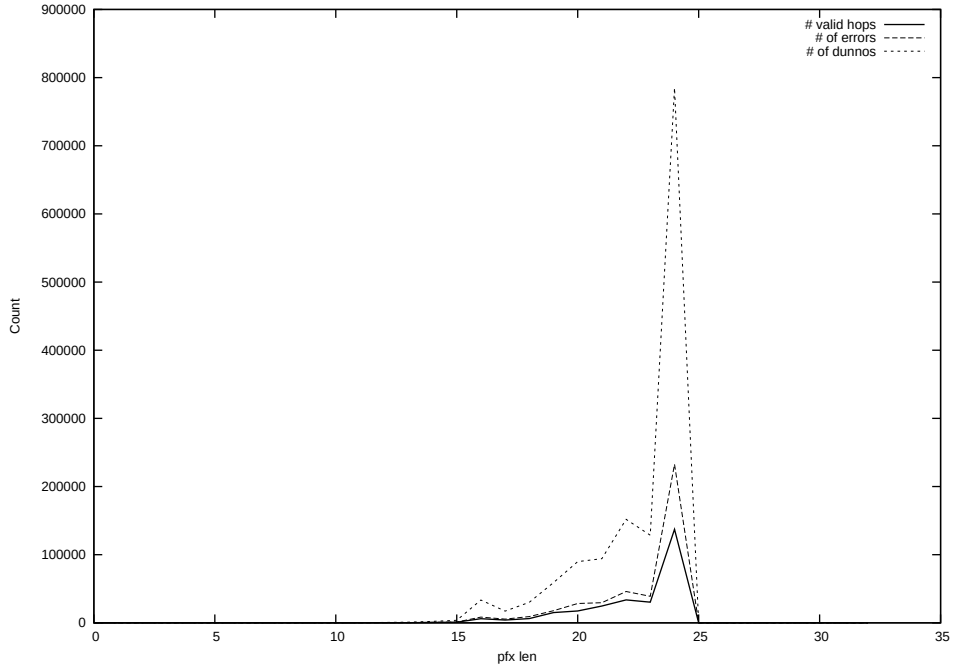
2015-02-20



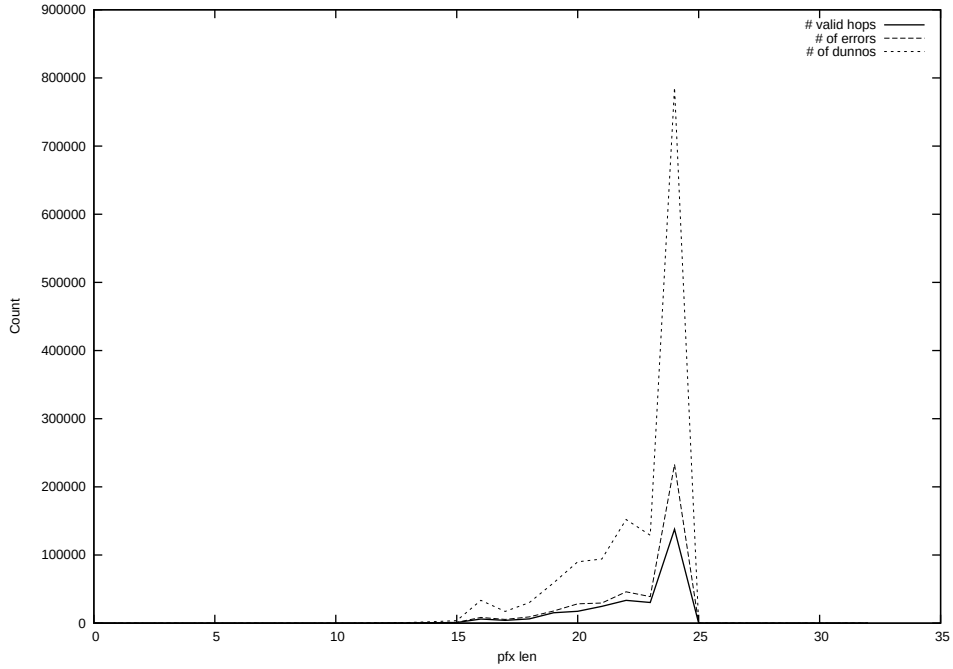
2015-02-21



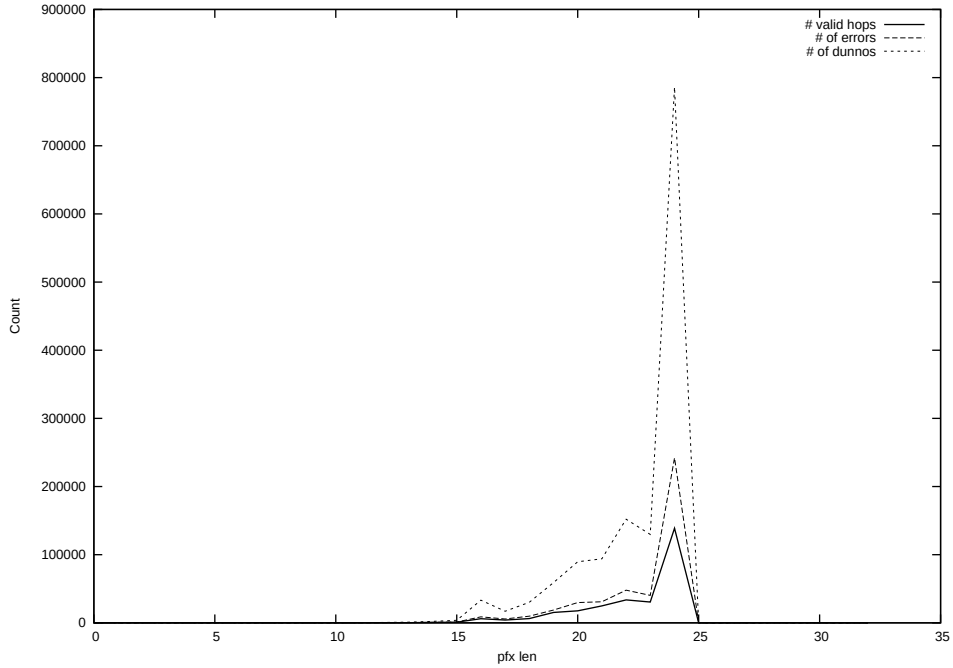
2015-02-22



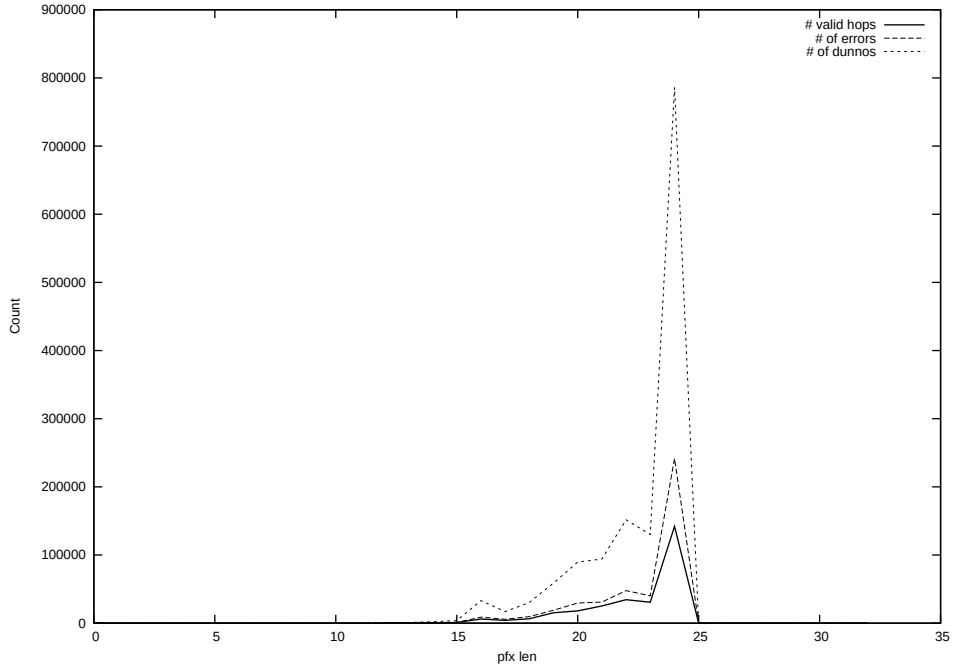
2015-02-23



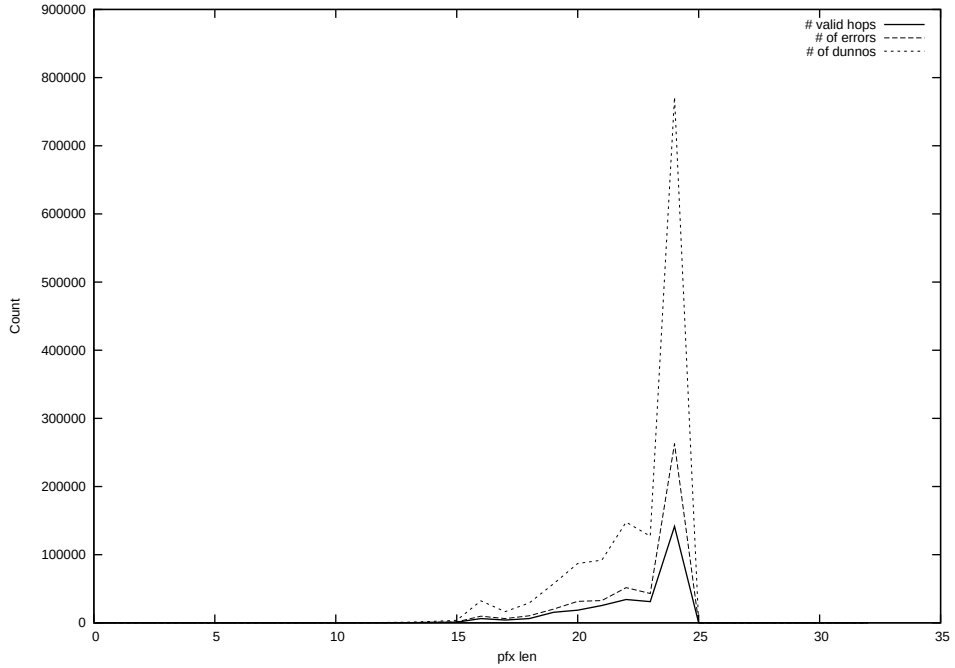
2015-02-24



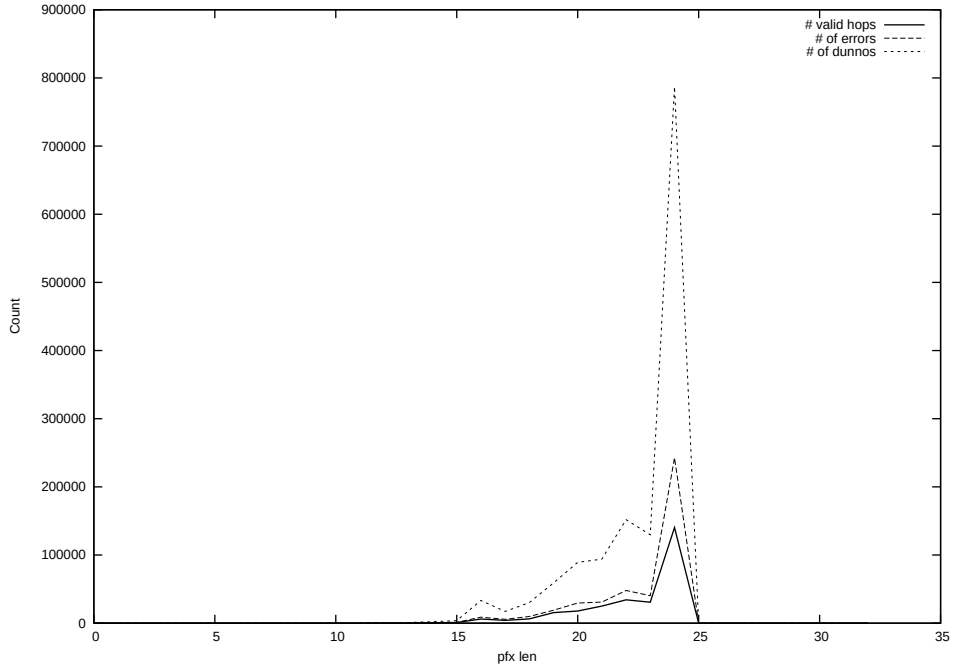
2015-02-25



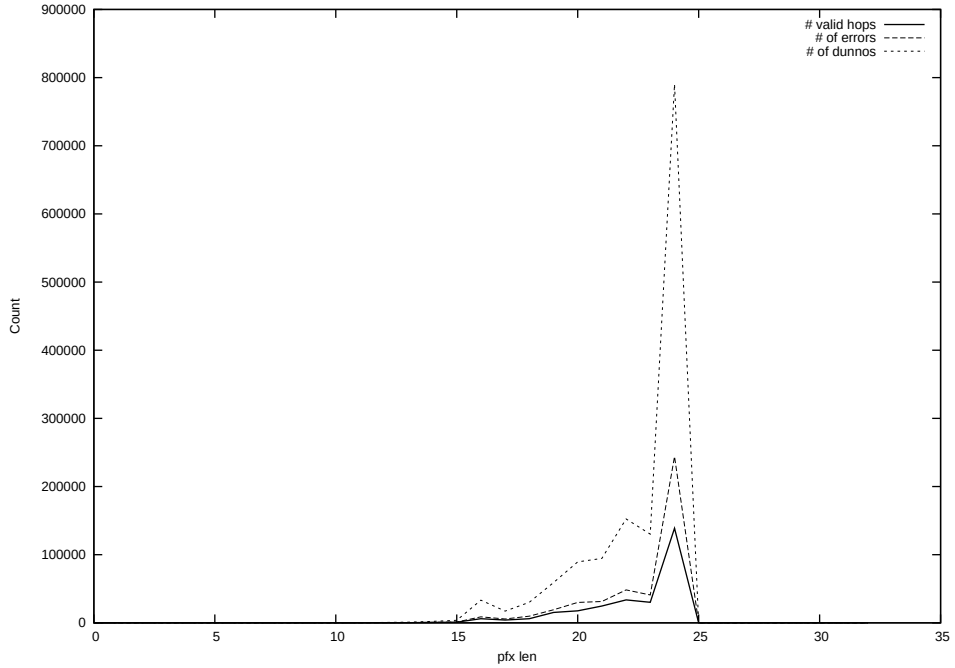
2015-02-26



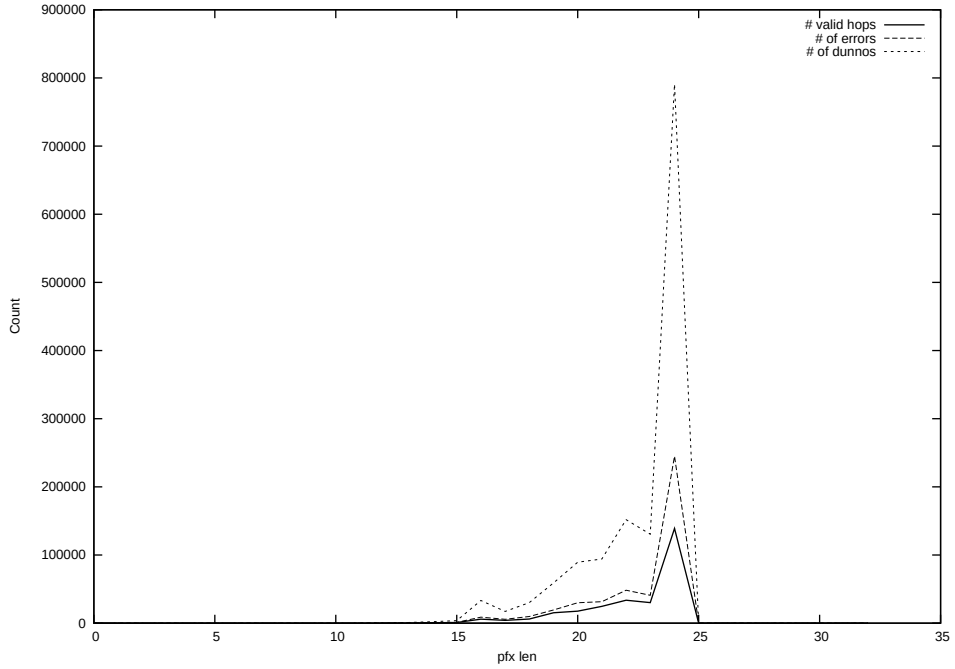
2015-02-27



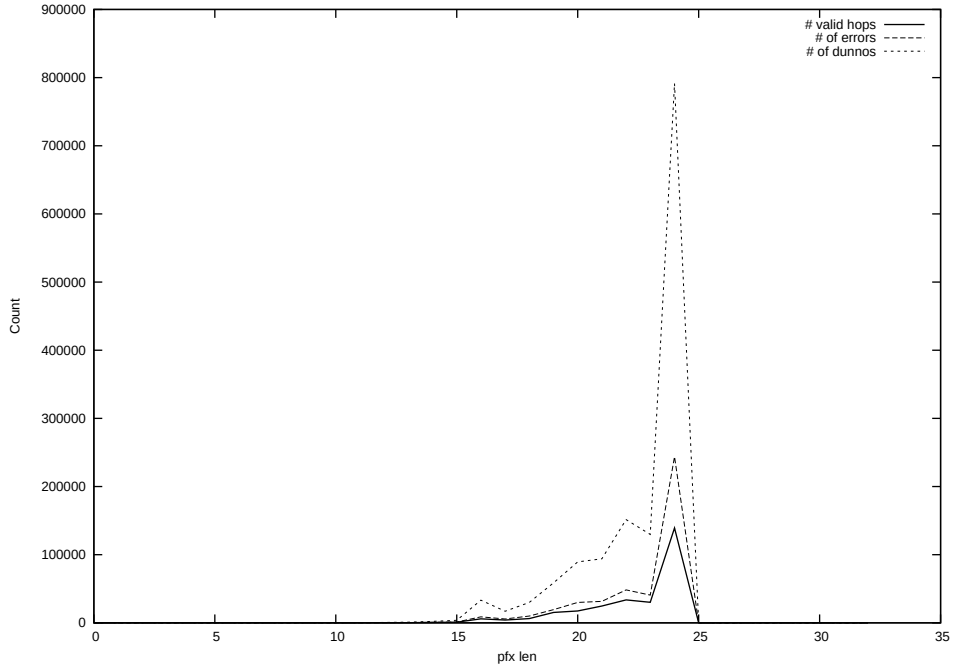
2015-02-28



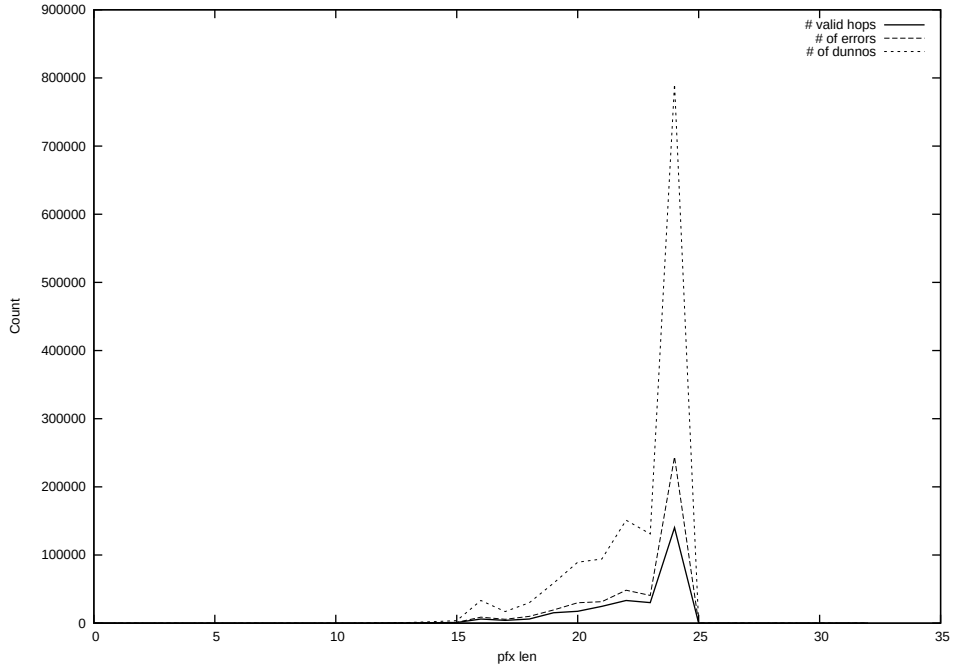
2015-03-01



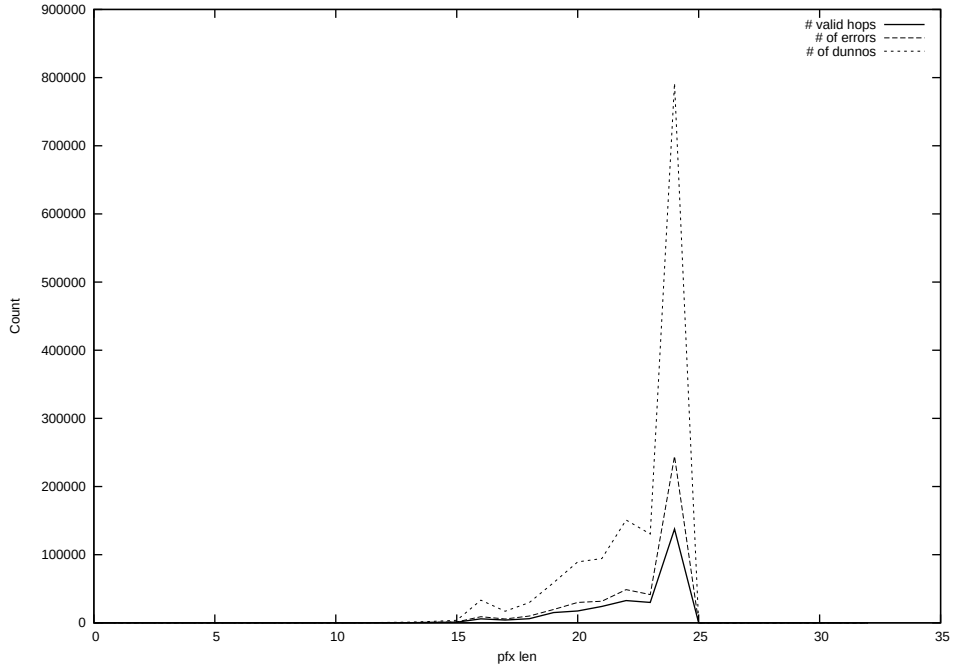
2015-03-02



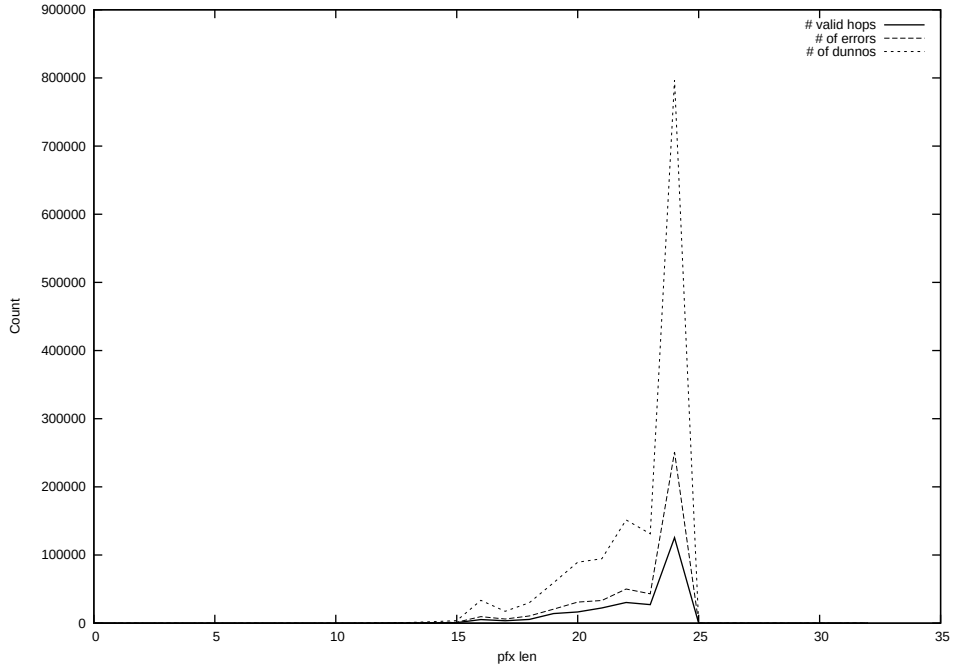
2015-03-03



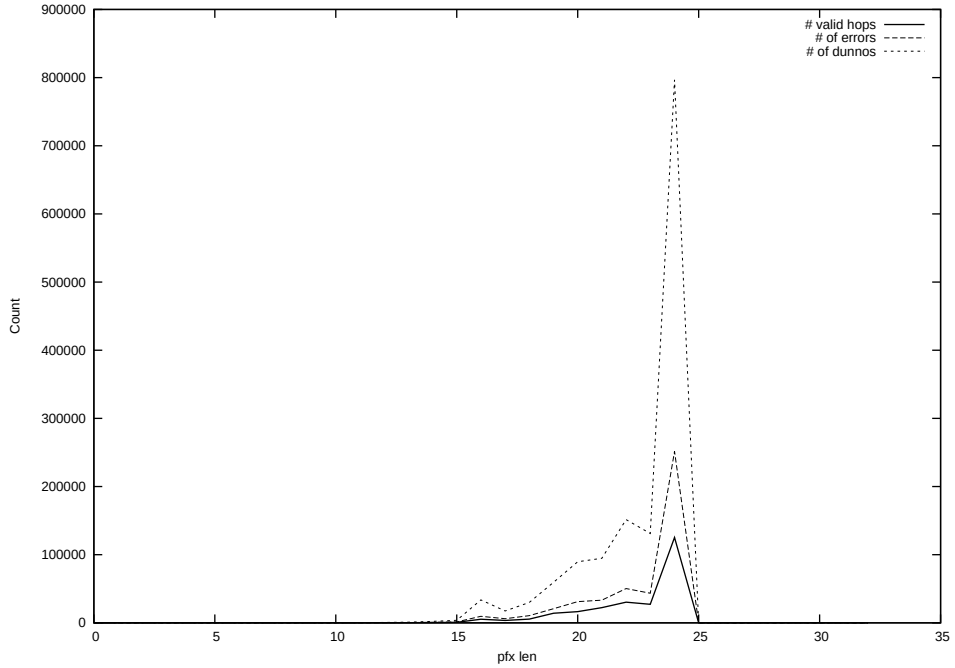
2015-03-04



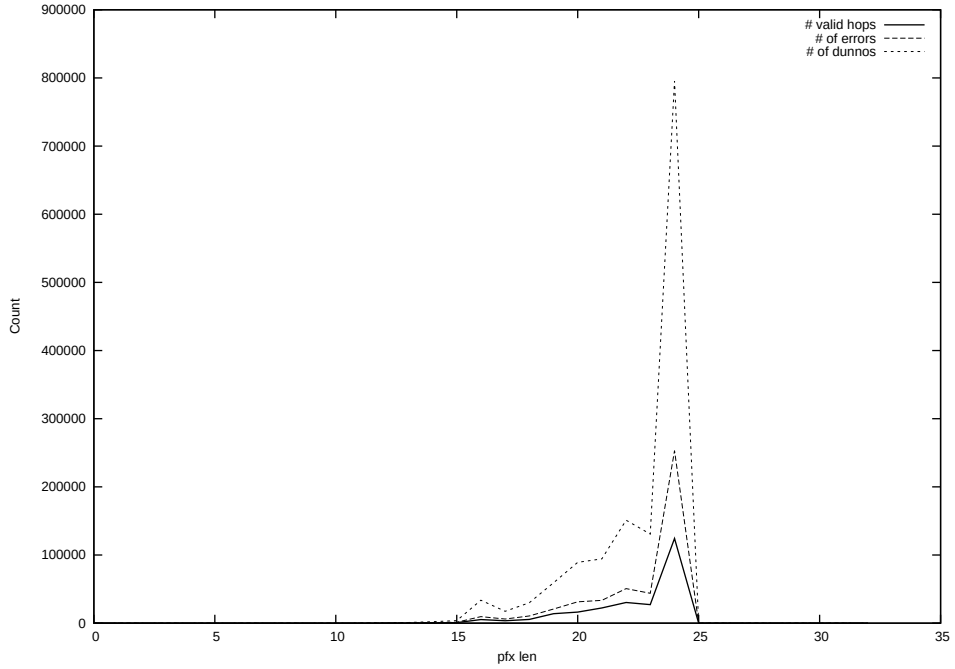
2015-03-05



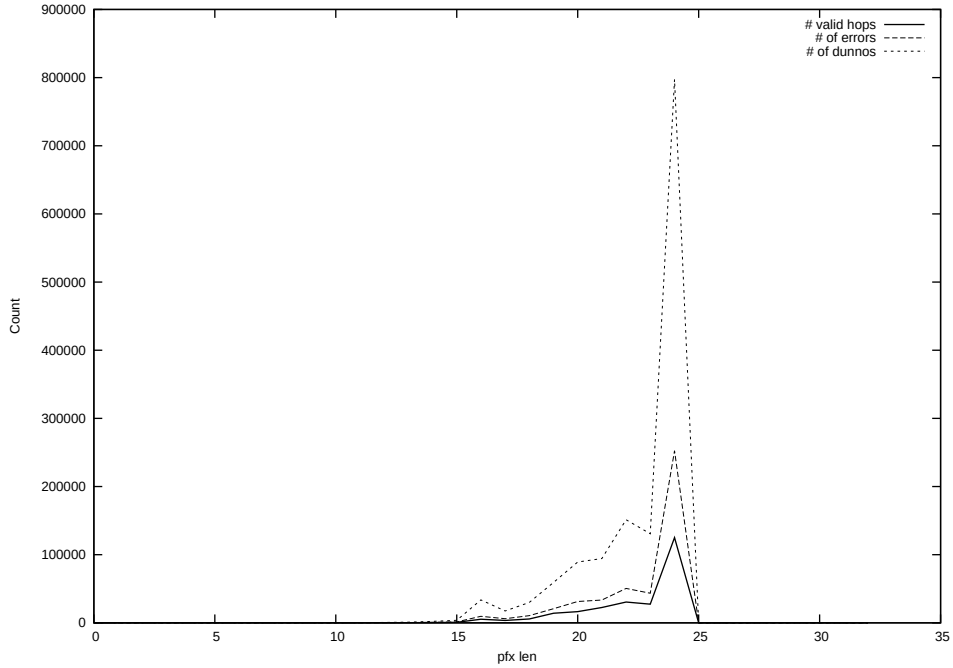
2015-03-06



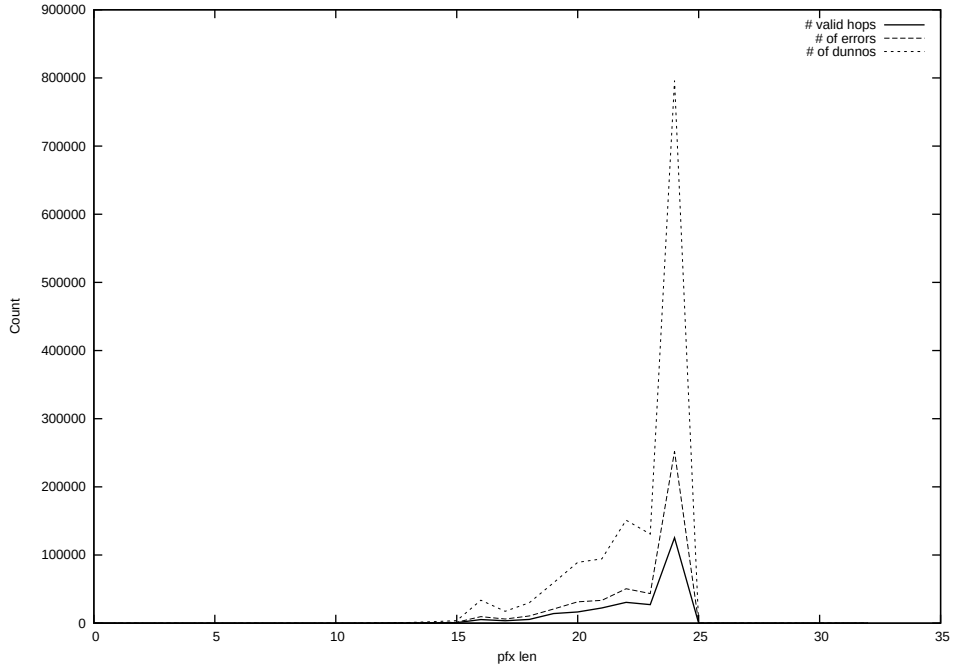
2015-03-07



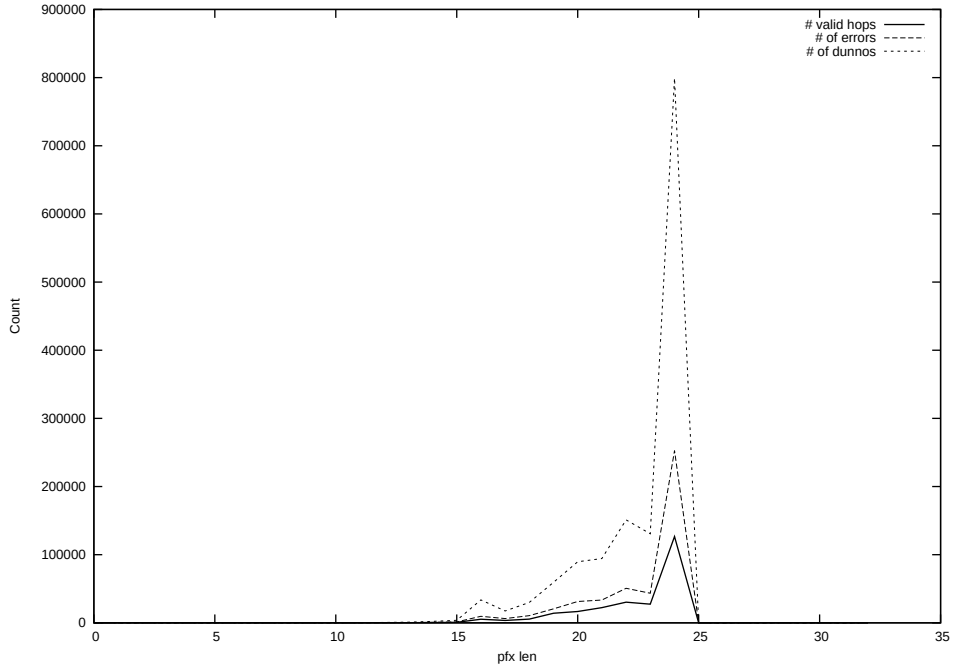
2015-03-08



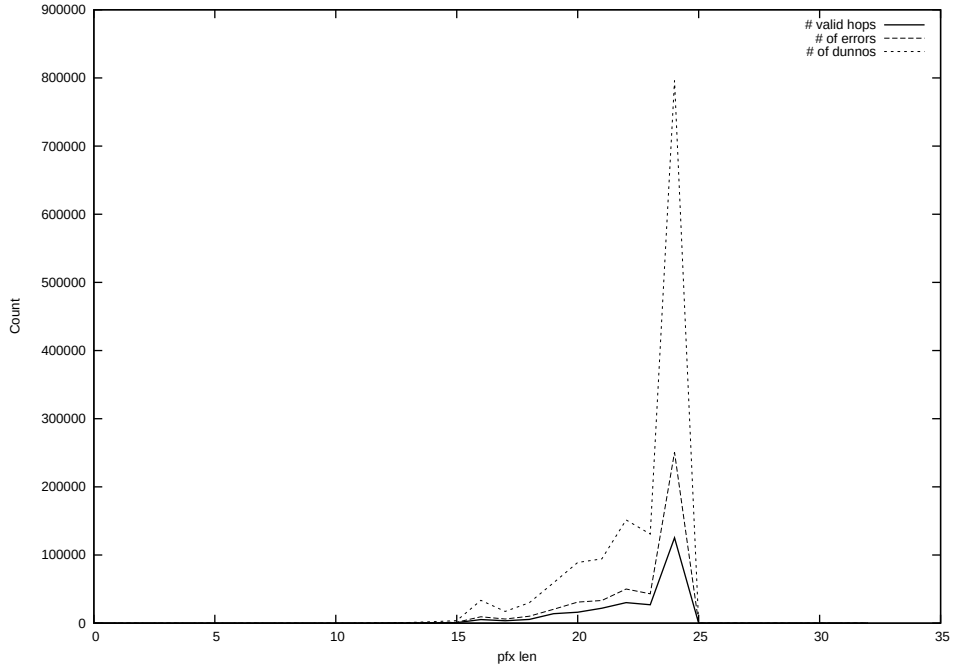
2015-03-09



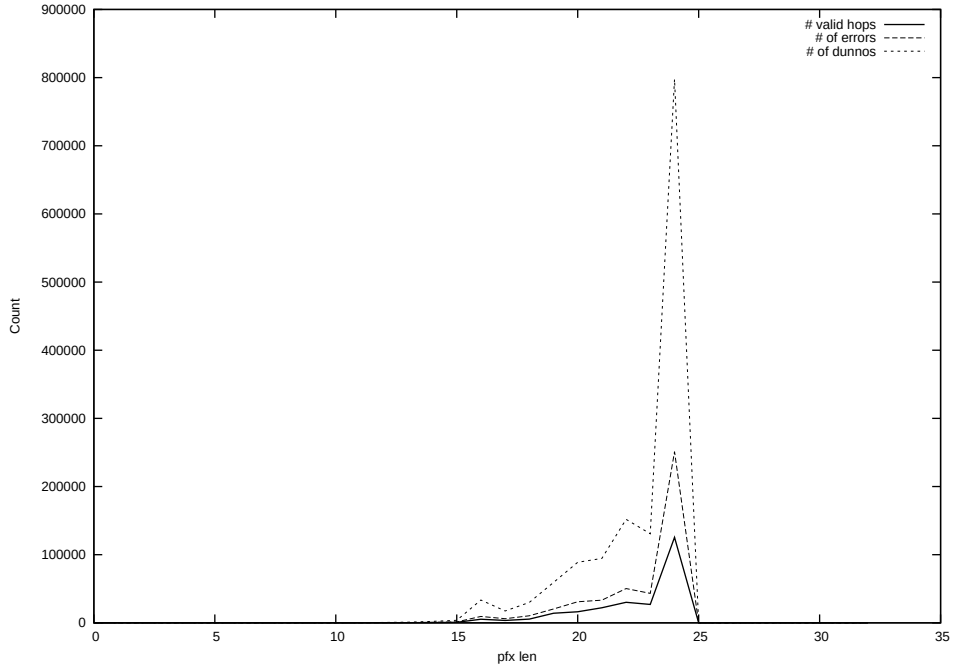
2015-03-10



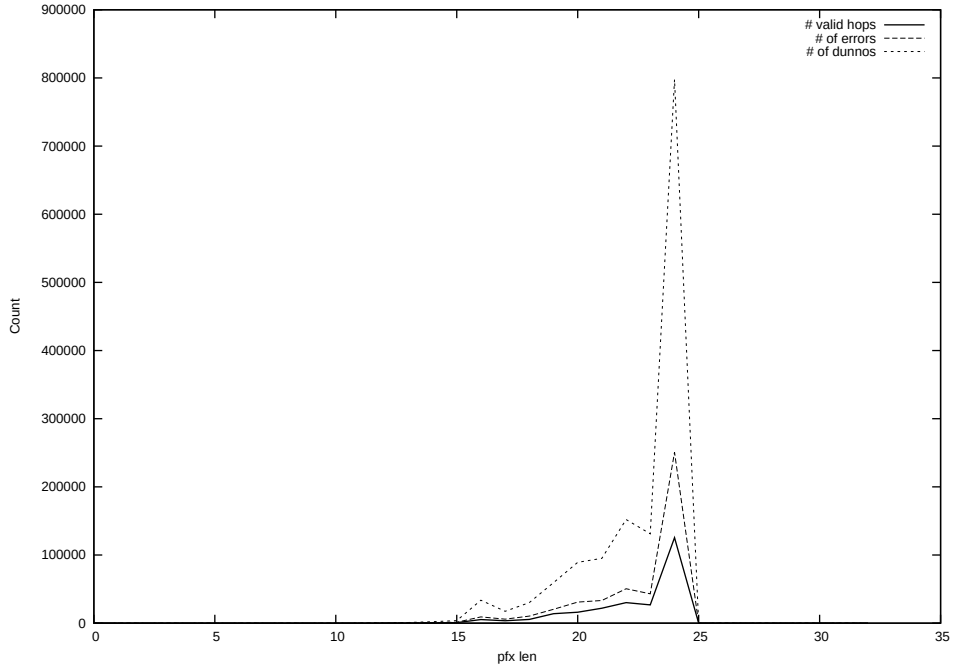
2015-03-11



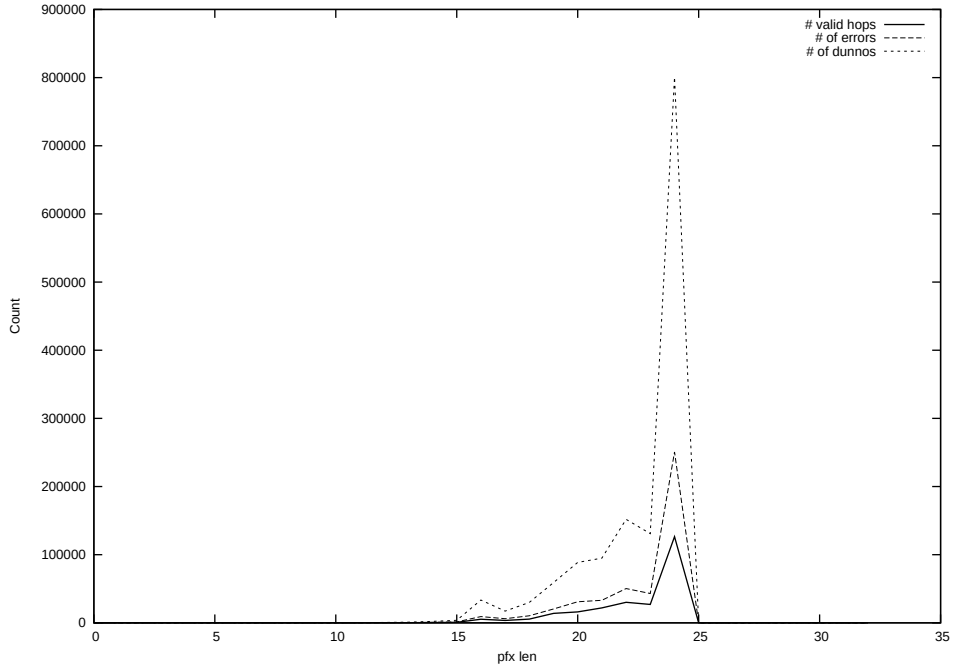
2015-03-12



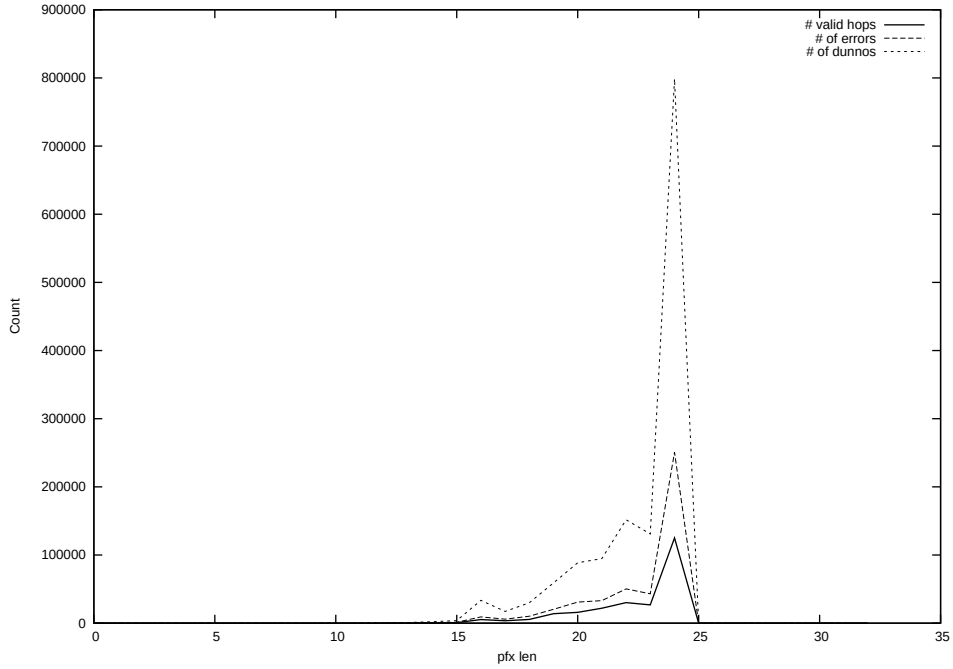
2015-03-13



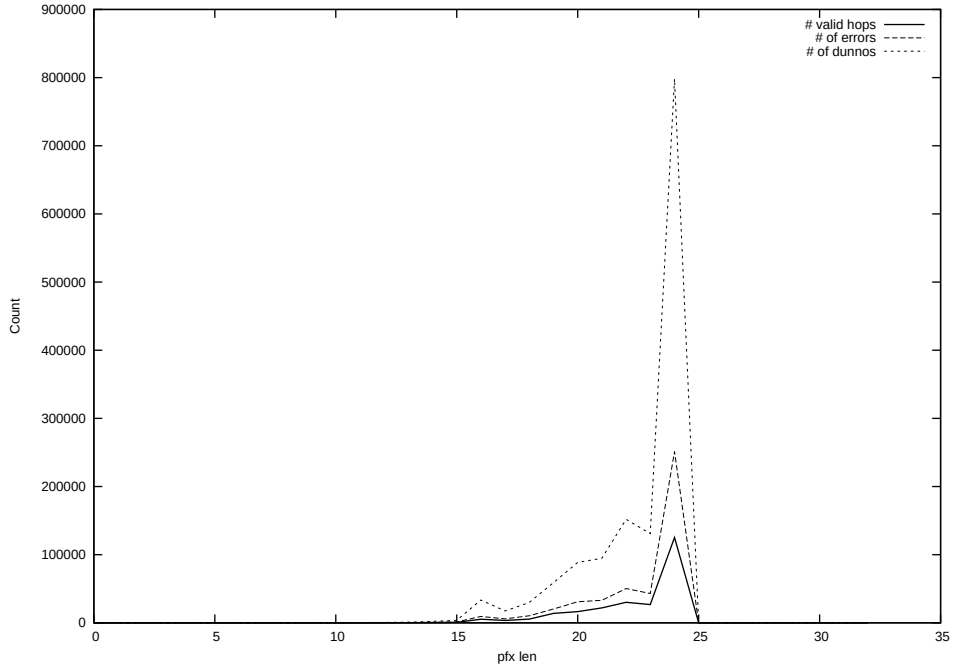
2015-03-14



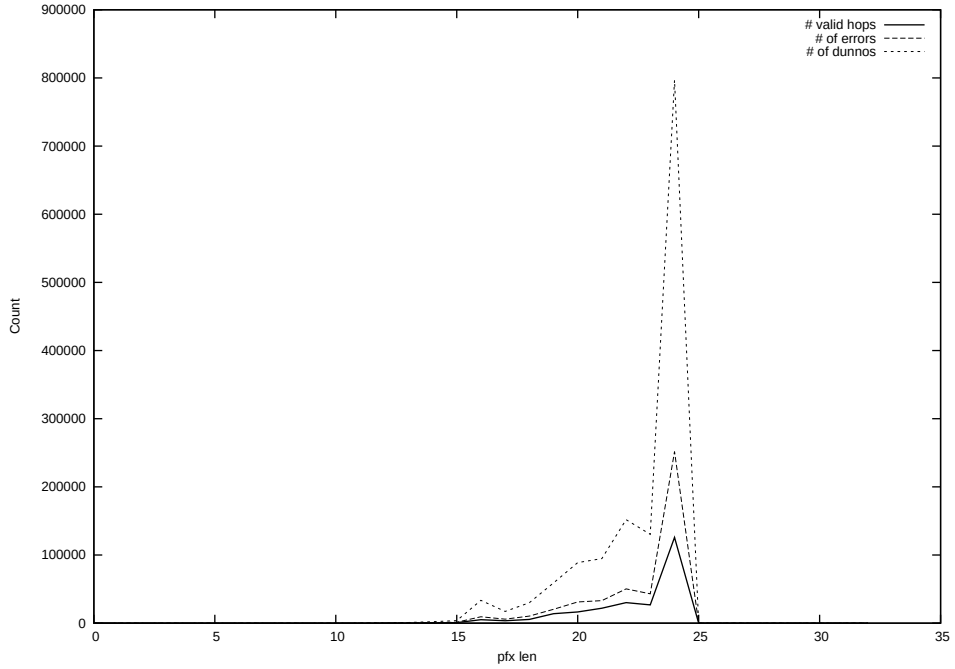
2015-03-15



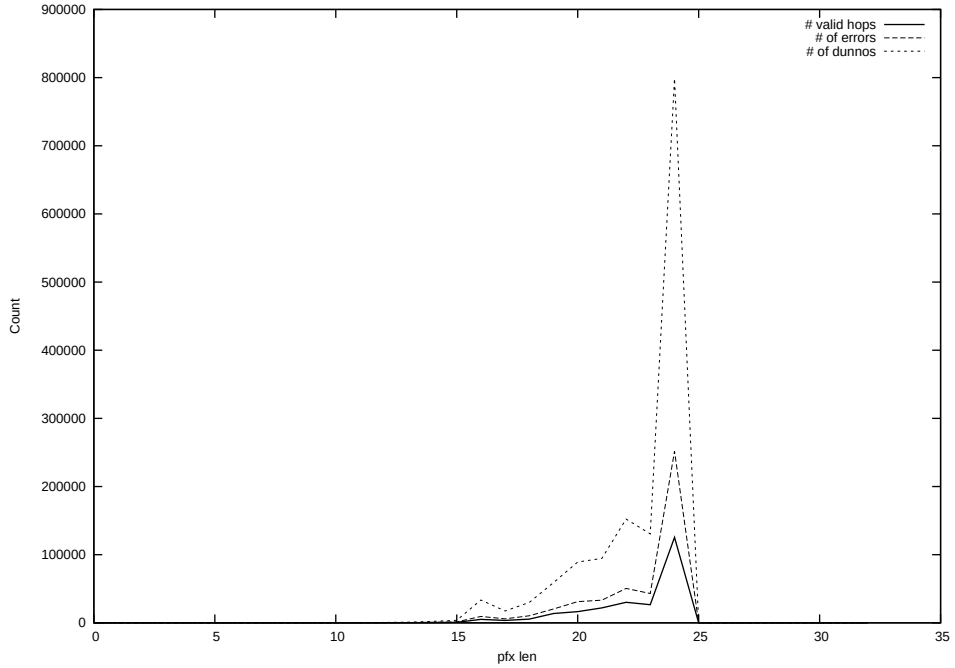
2015-03-16



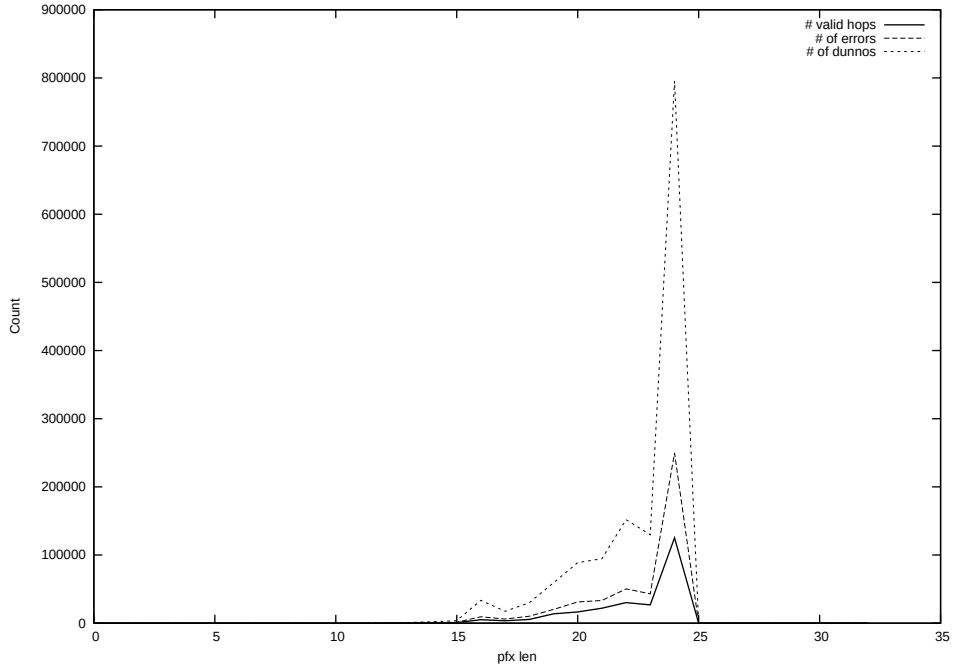
2015-03-17



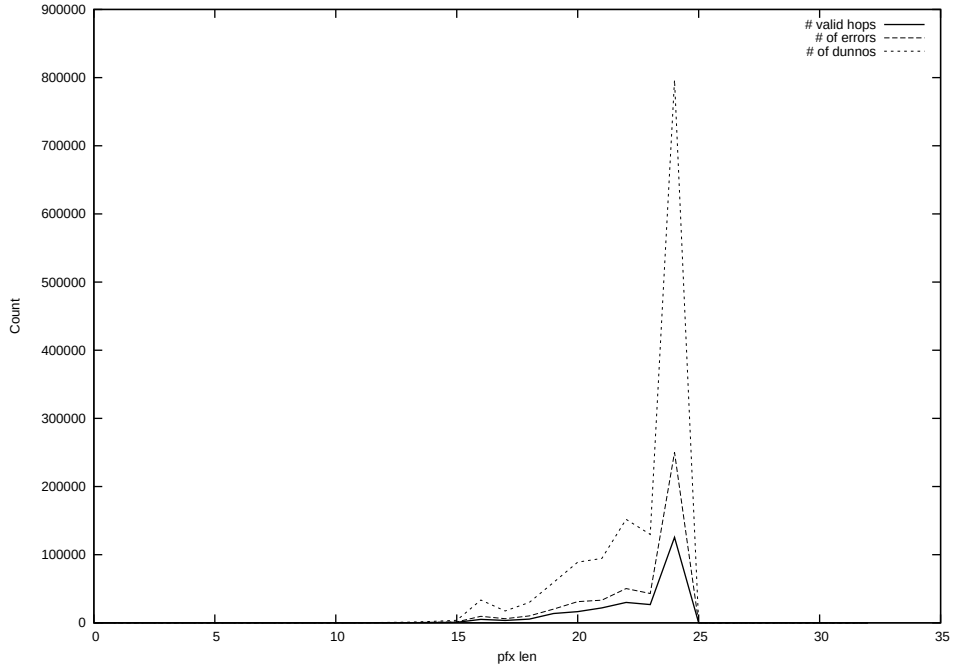
2015-03-18



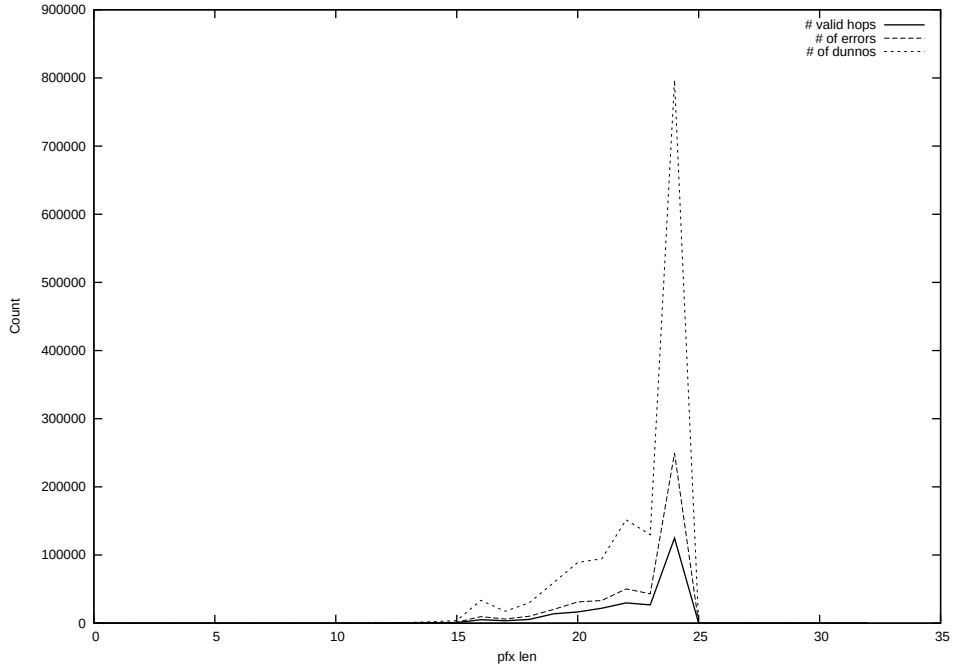
2015-03-19



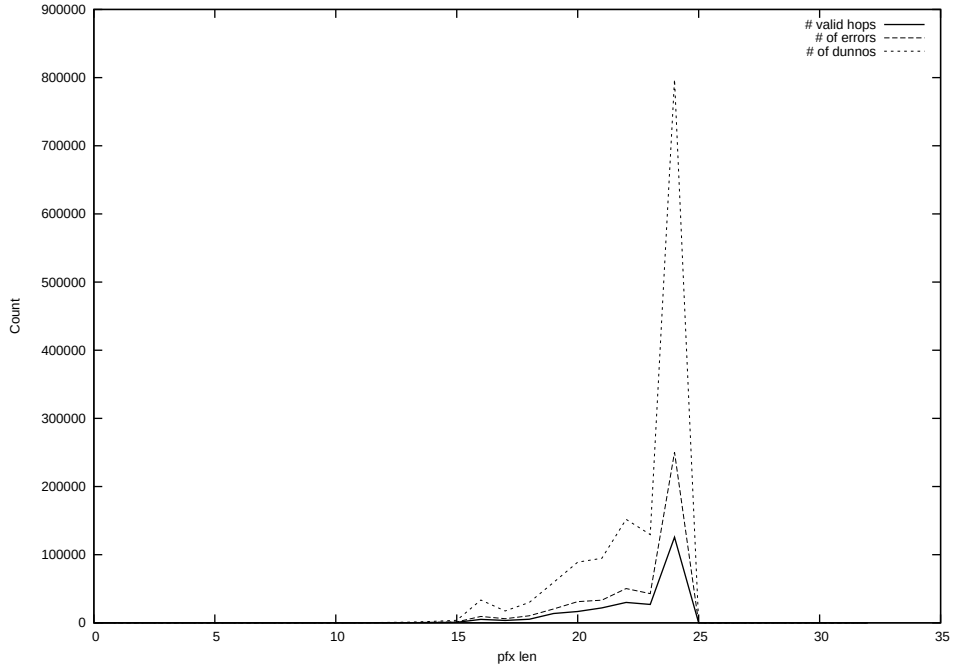
2015-03-20



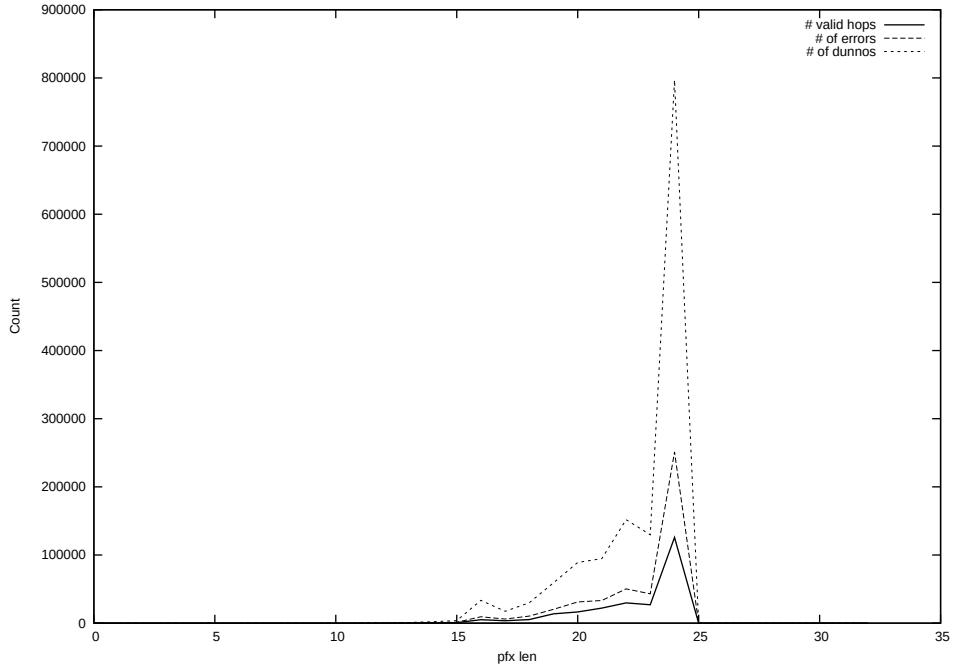
2015-03-21



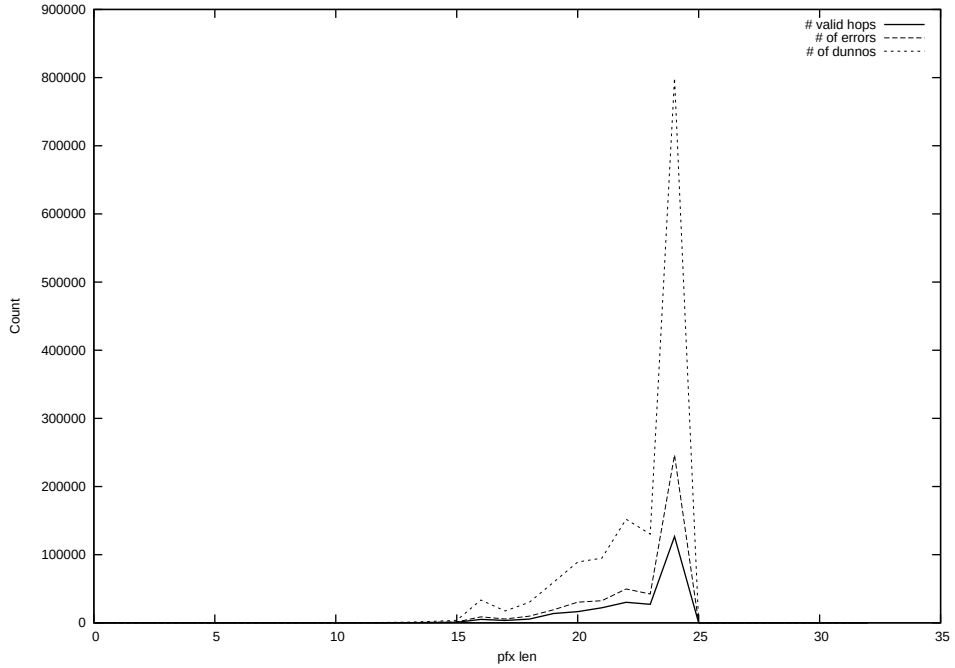
2015-03-22



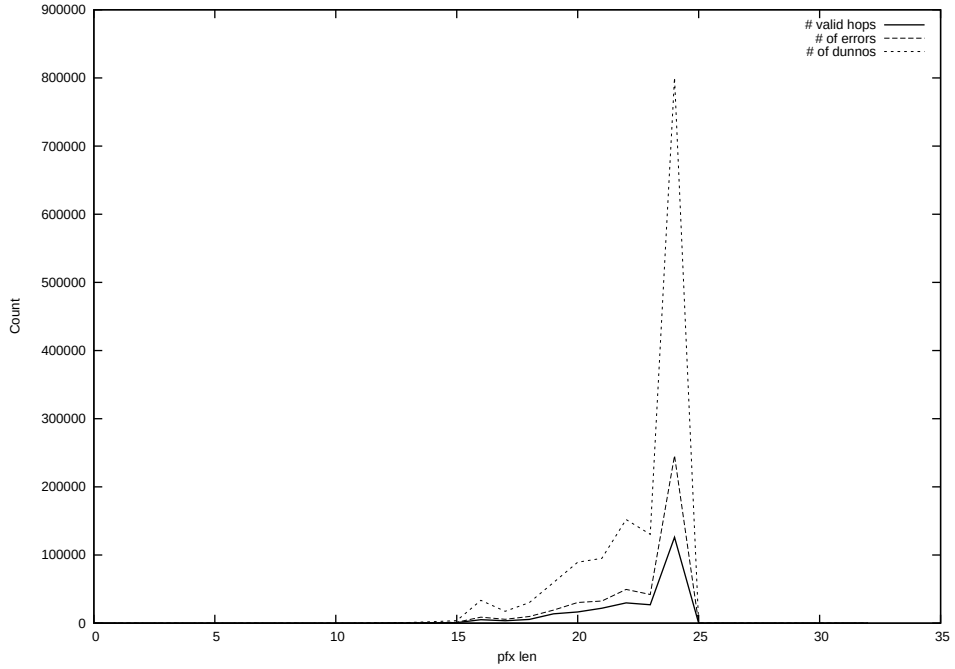
2015-03-23



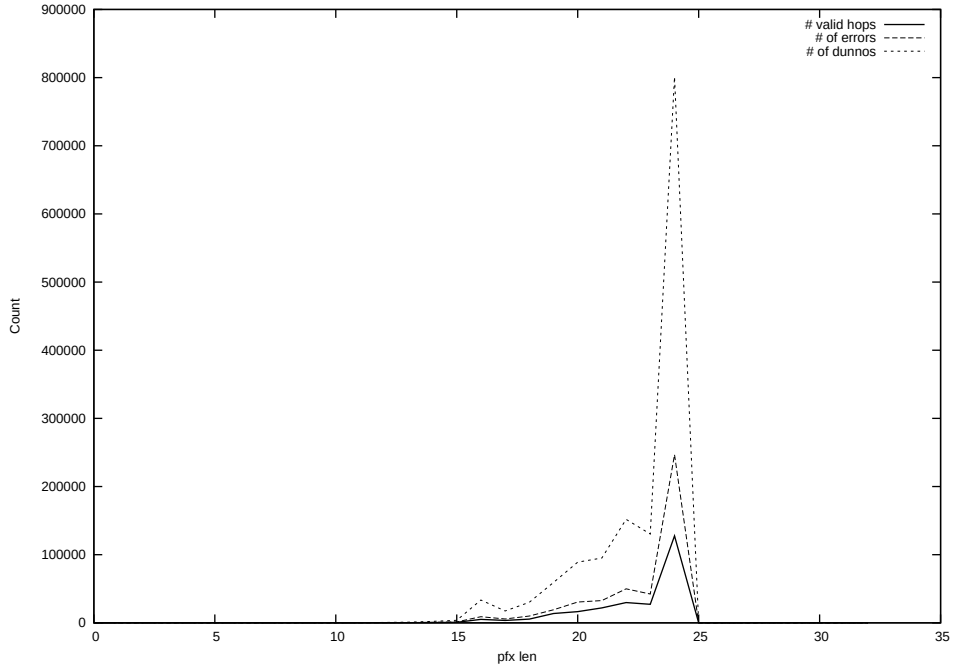
2015-03-24



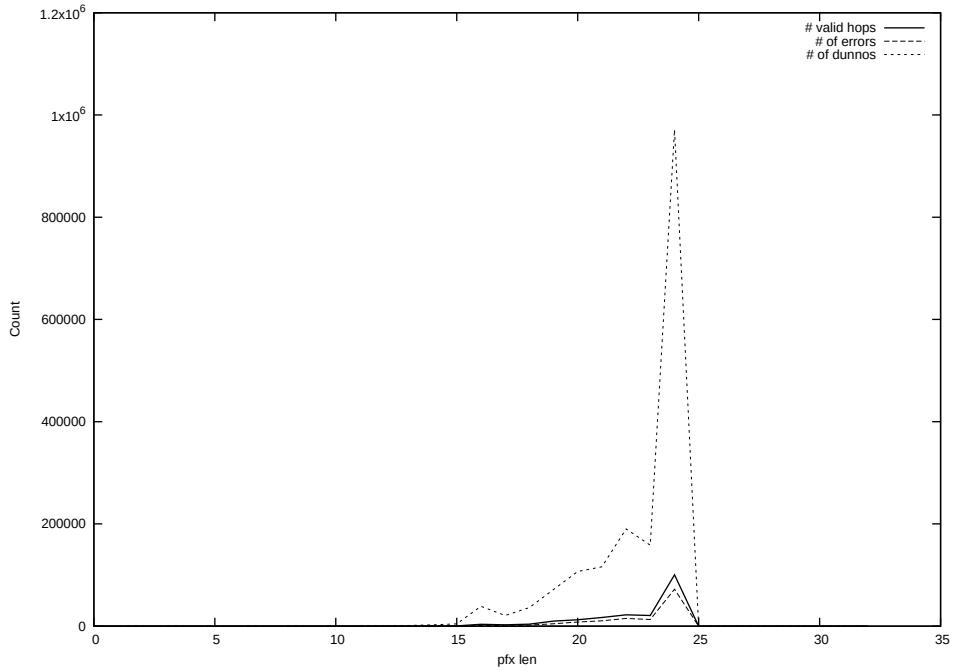
2015-03-25



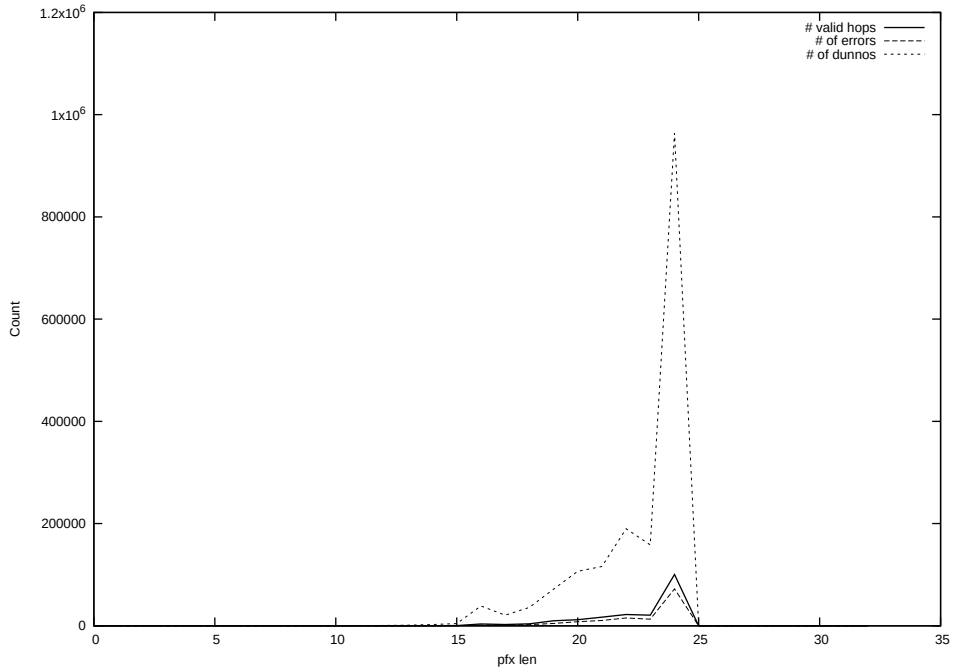
2015-03-26



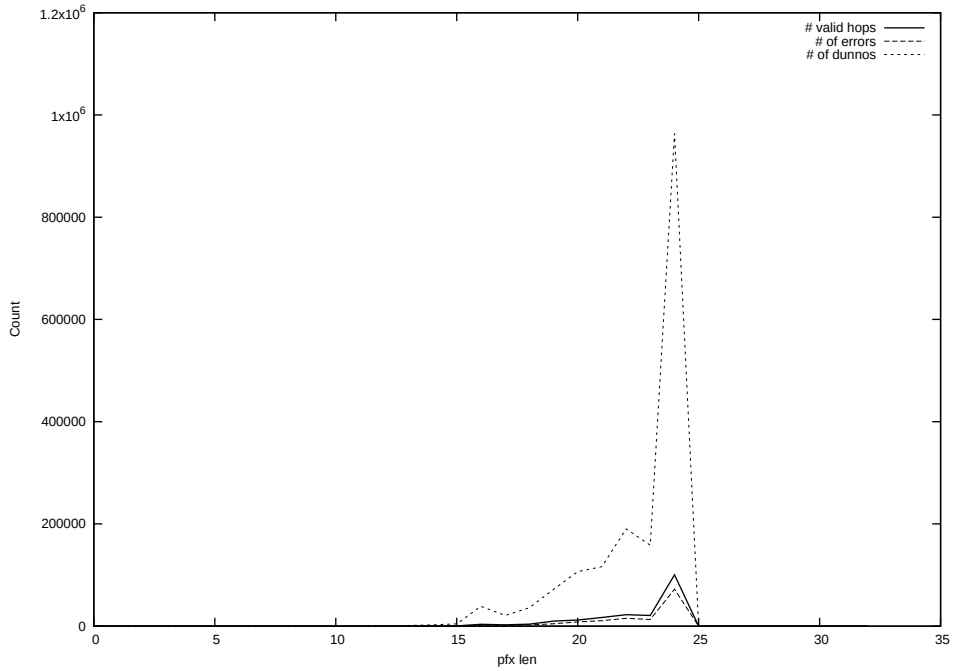
2015-03-27



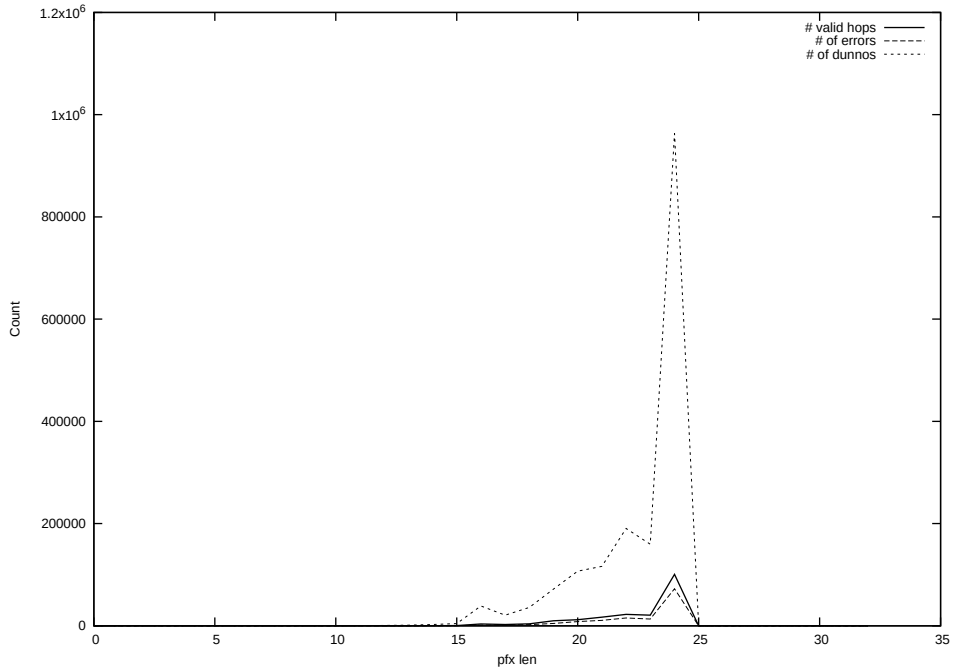
2015-03-28



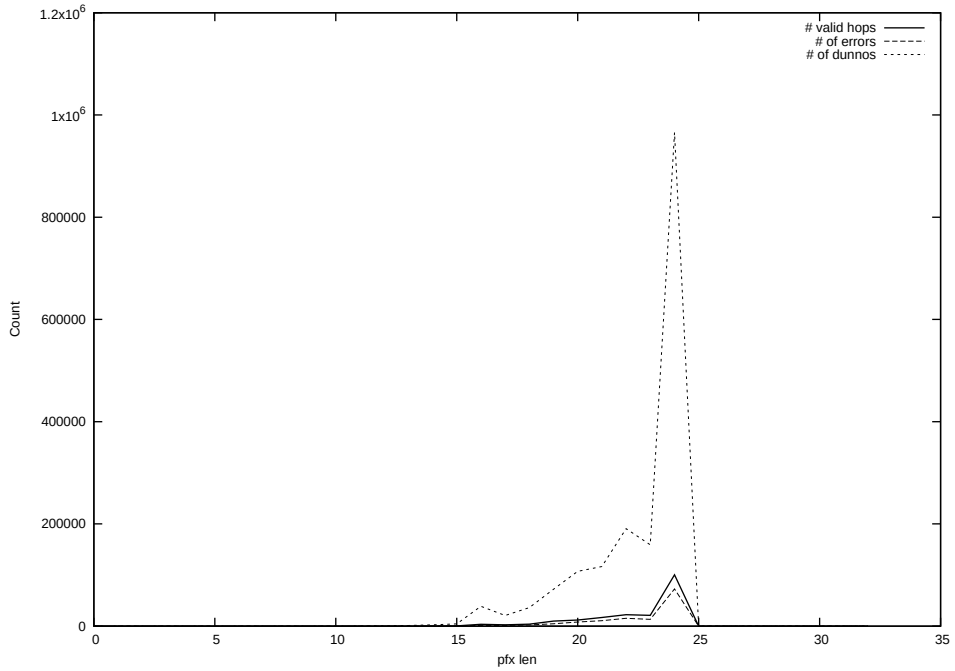
2015-03-29



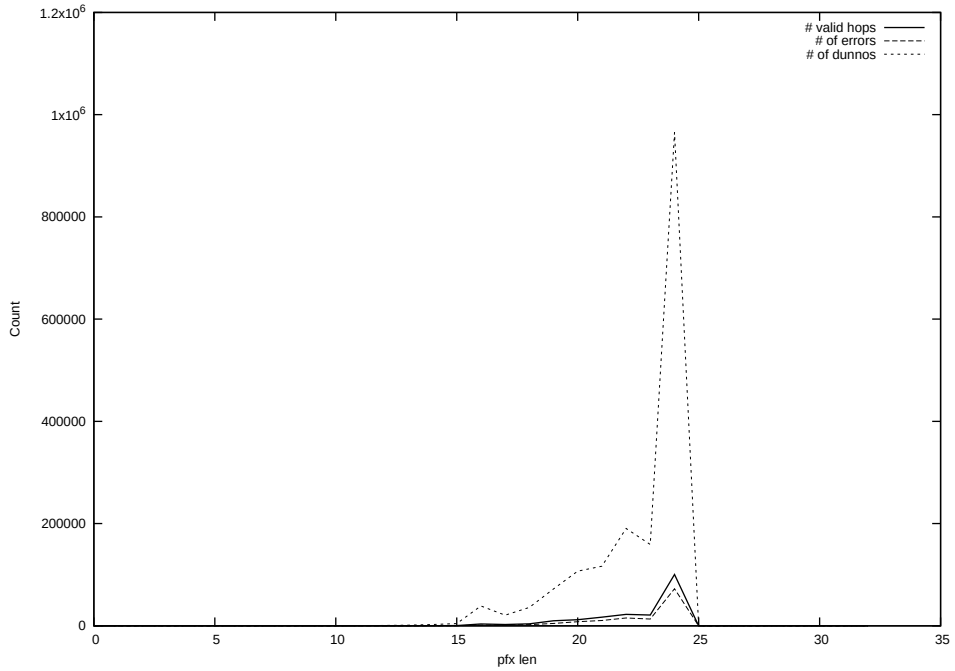
2015-03-30



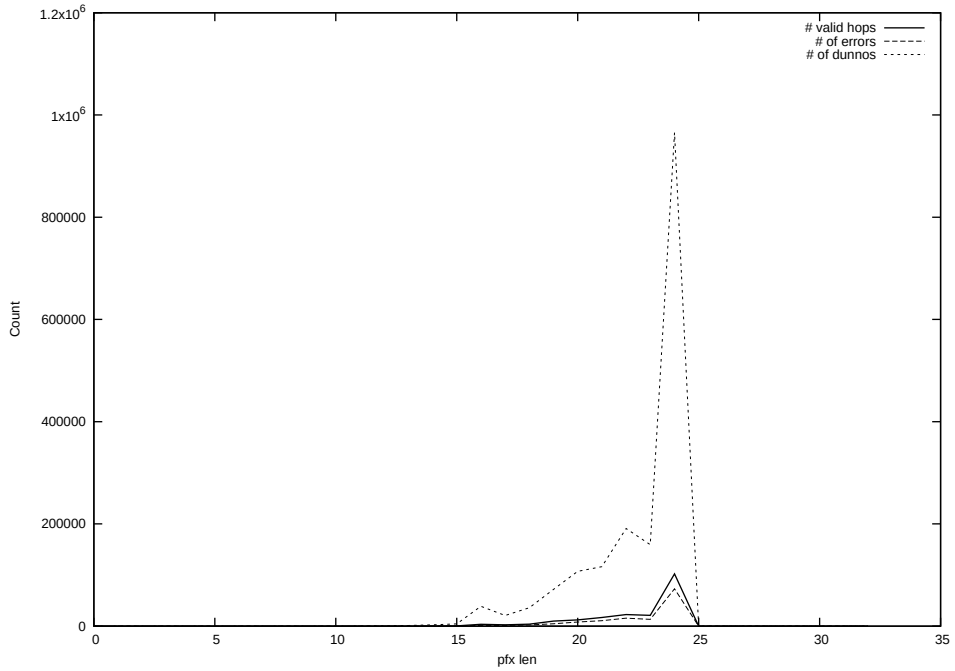
2015-03-31



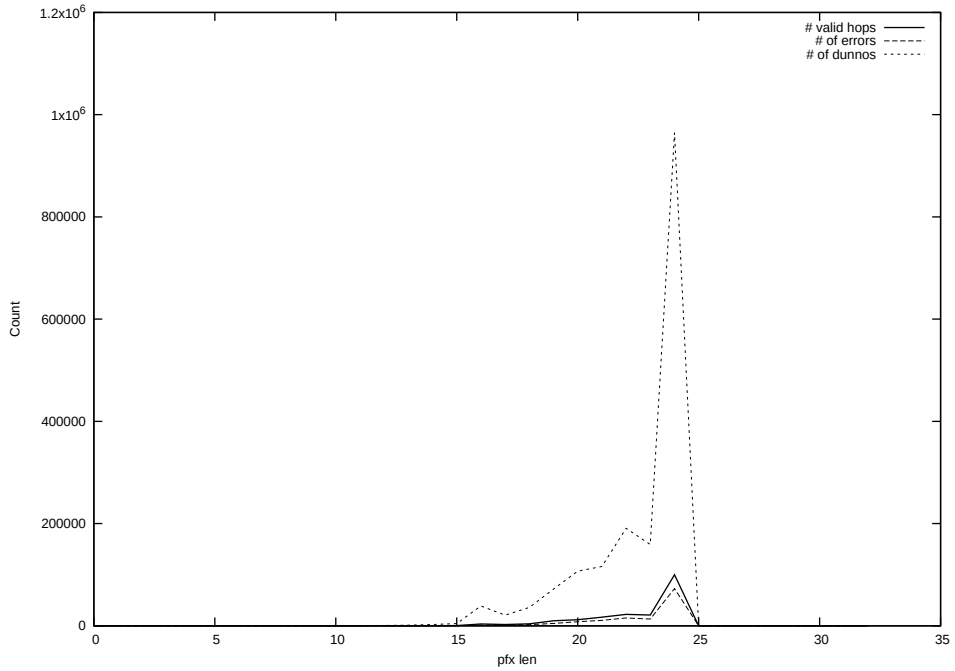
2015-04-01



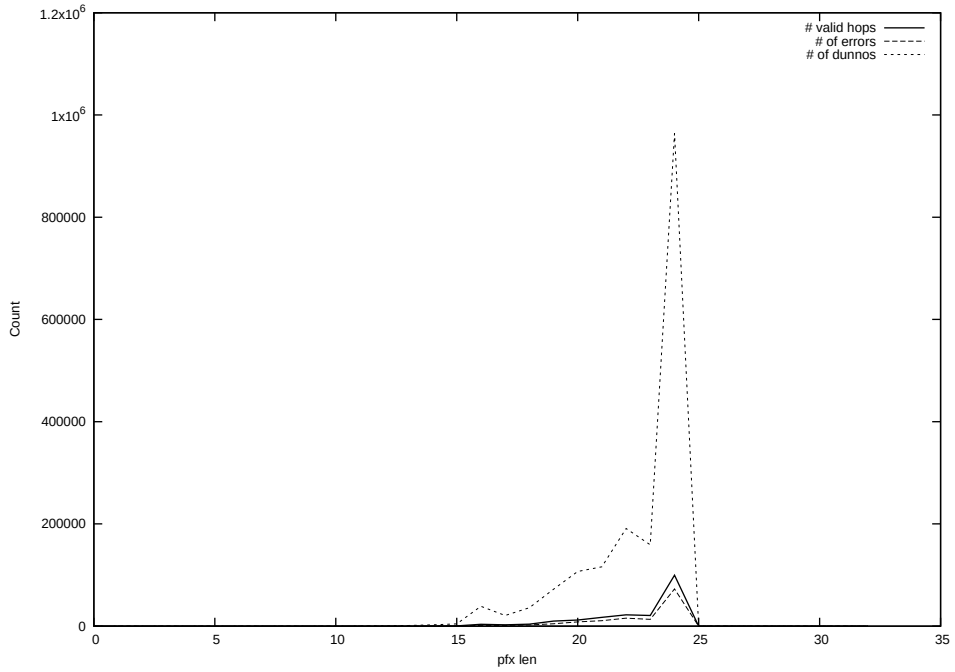
2015-04-02



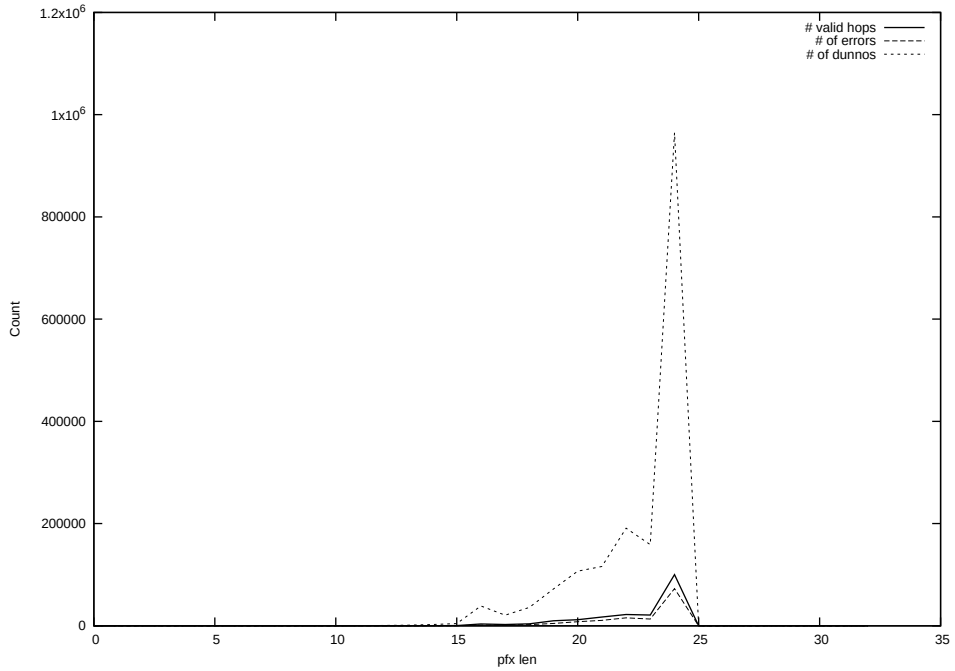
2015-04-03



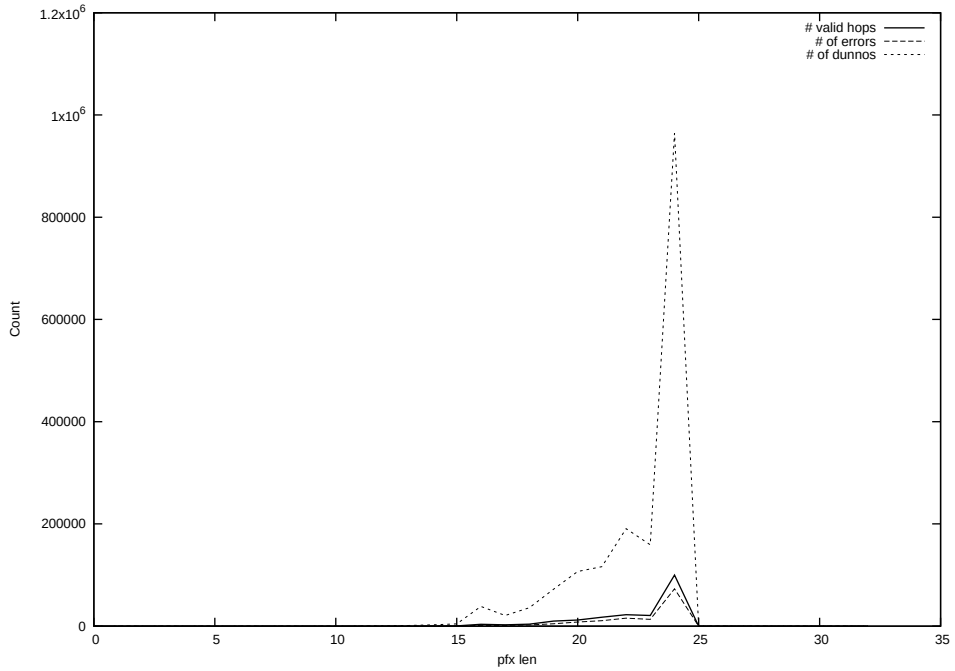
2015-04-04



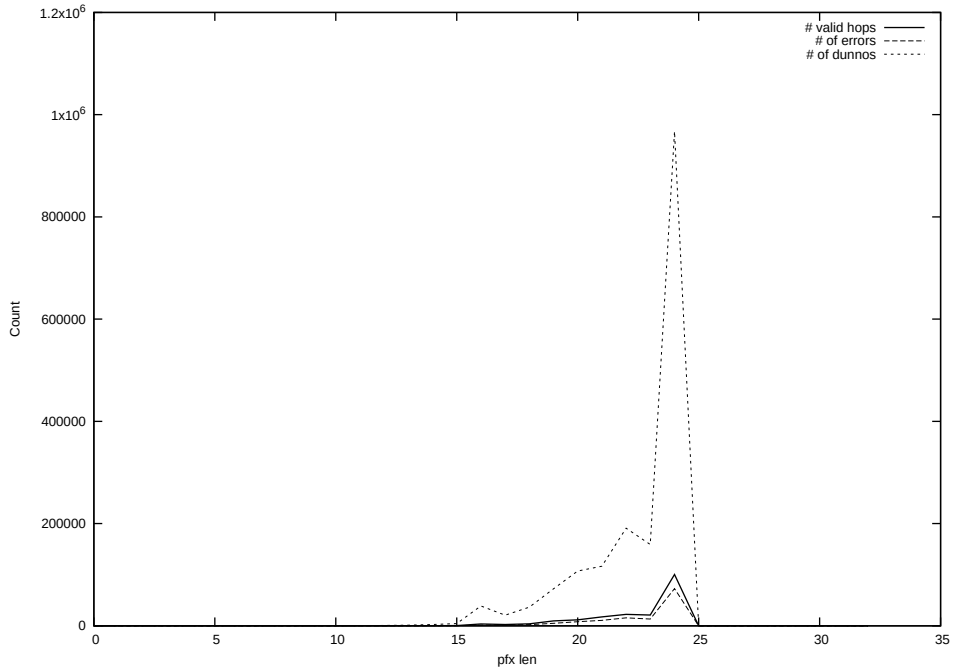
2015-04-05



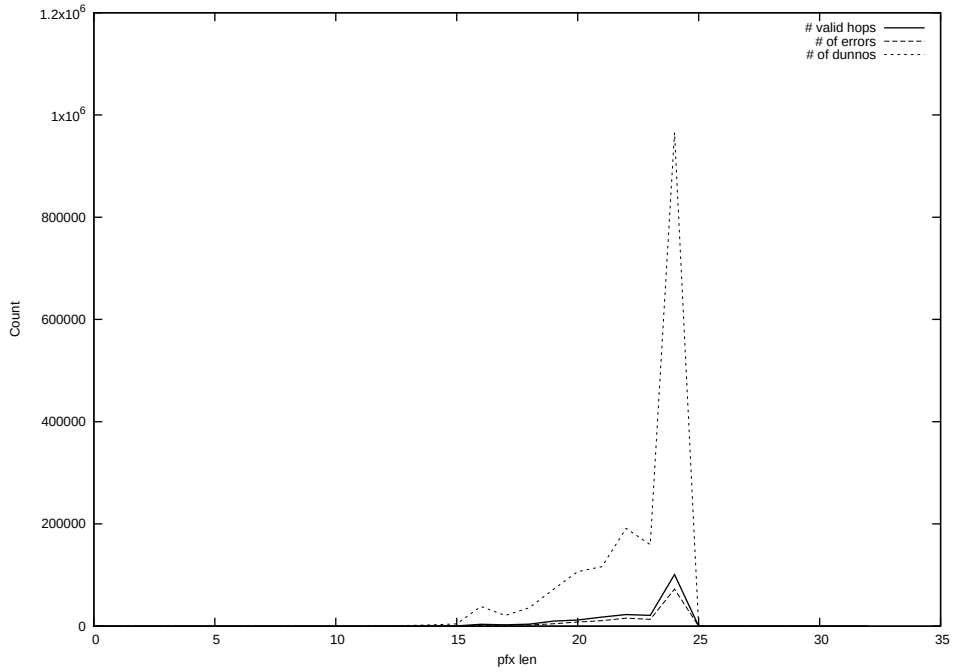
2015-04-06



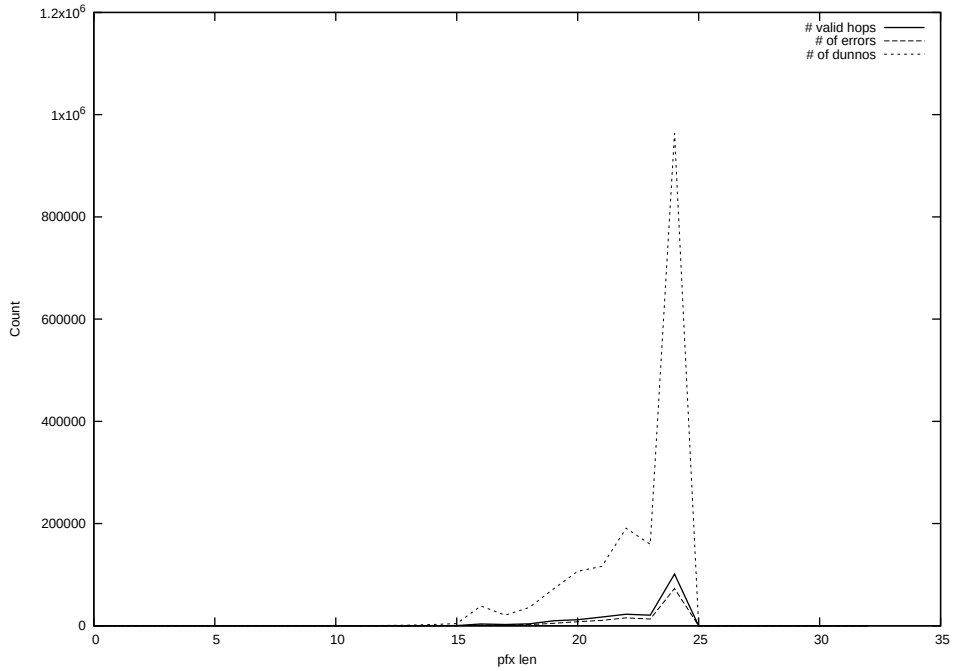
2015-04-07



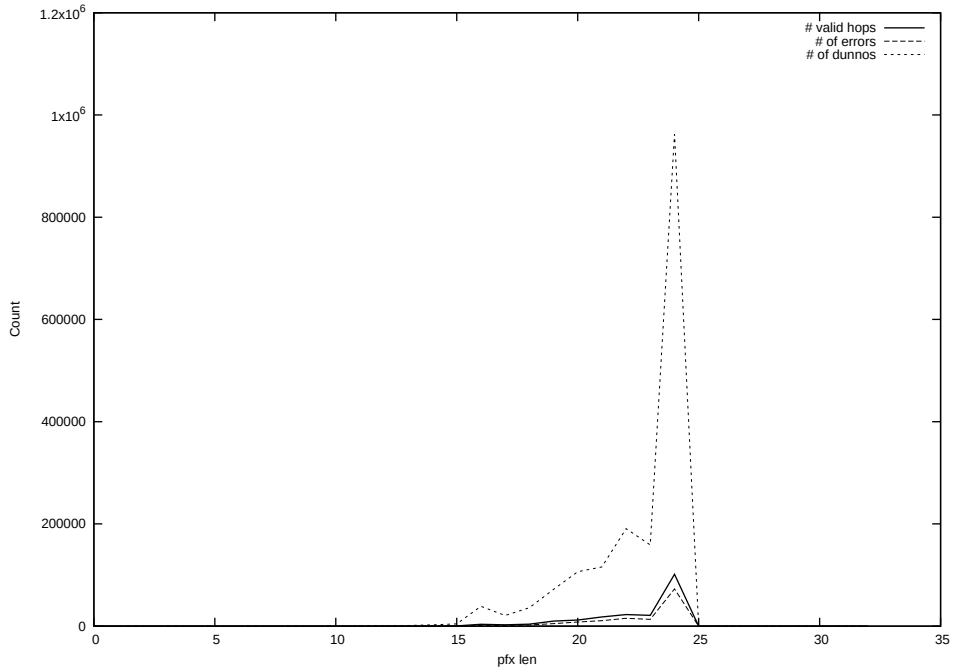
2015-04-08



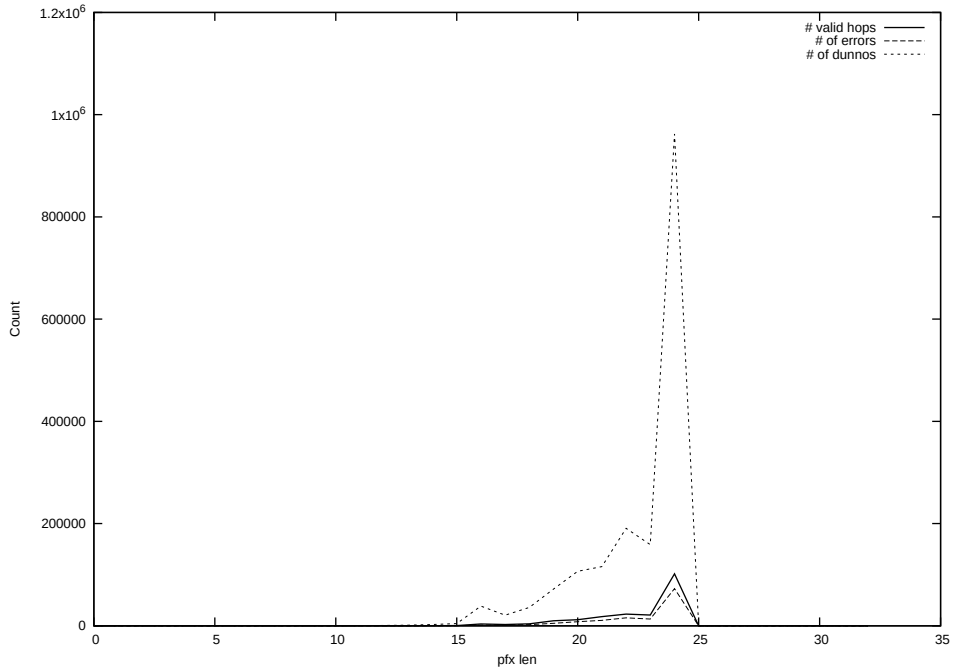
2015-04-09



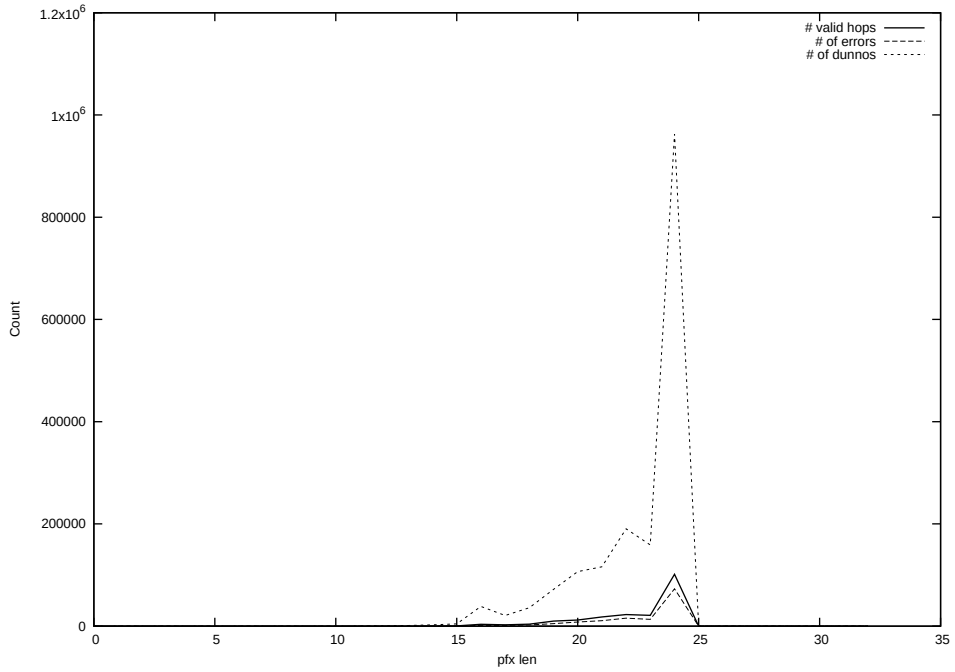
2015-04-10



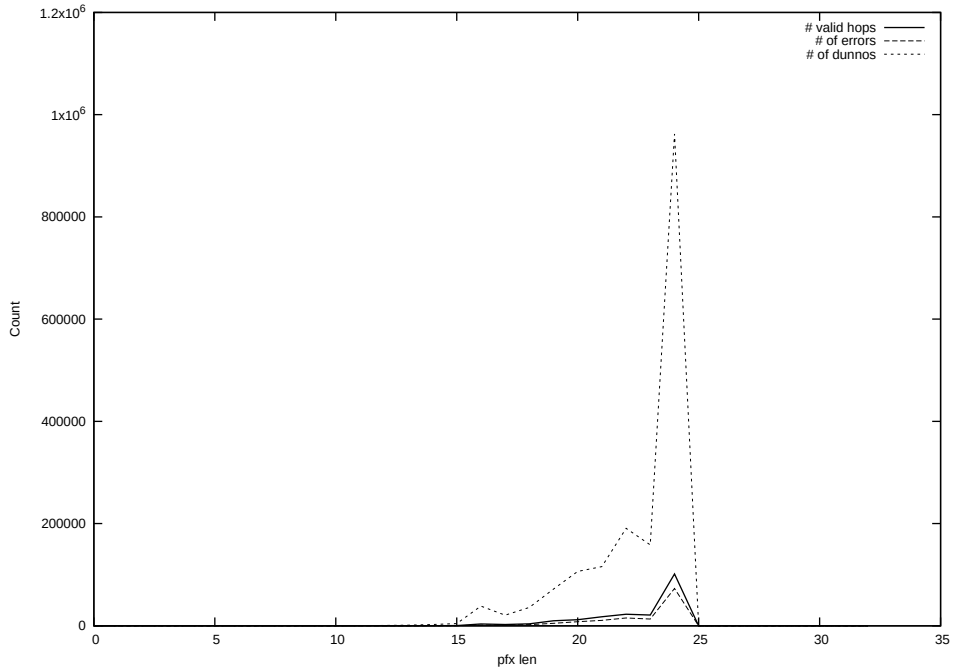
2015-04-11



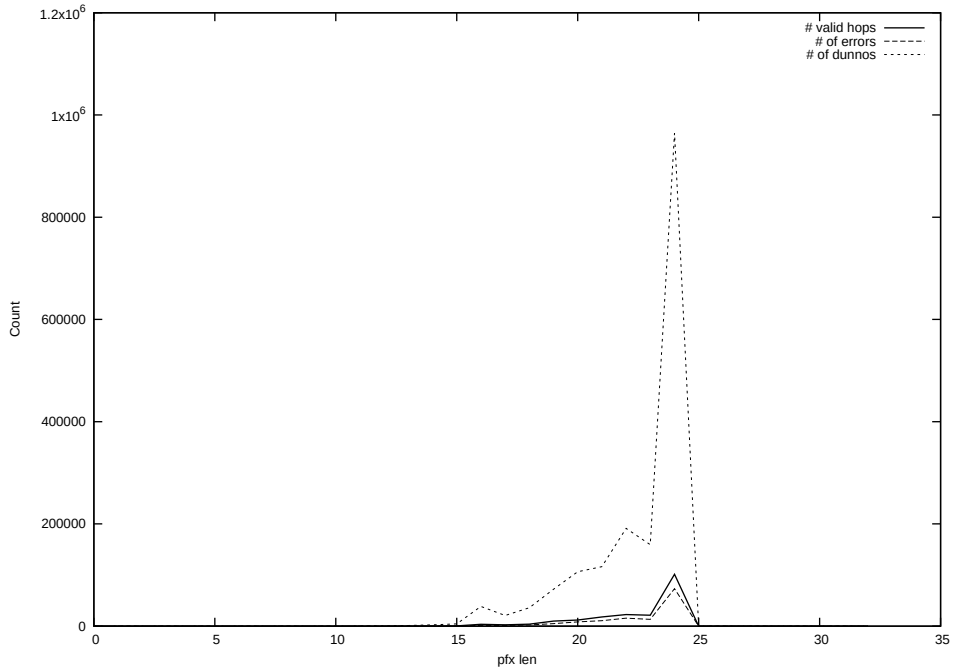
2015-04-12



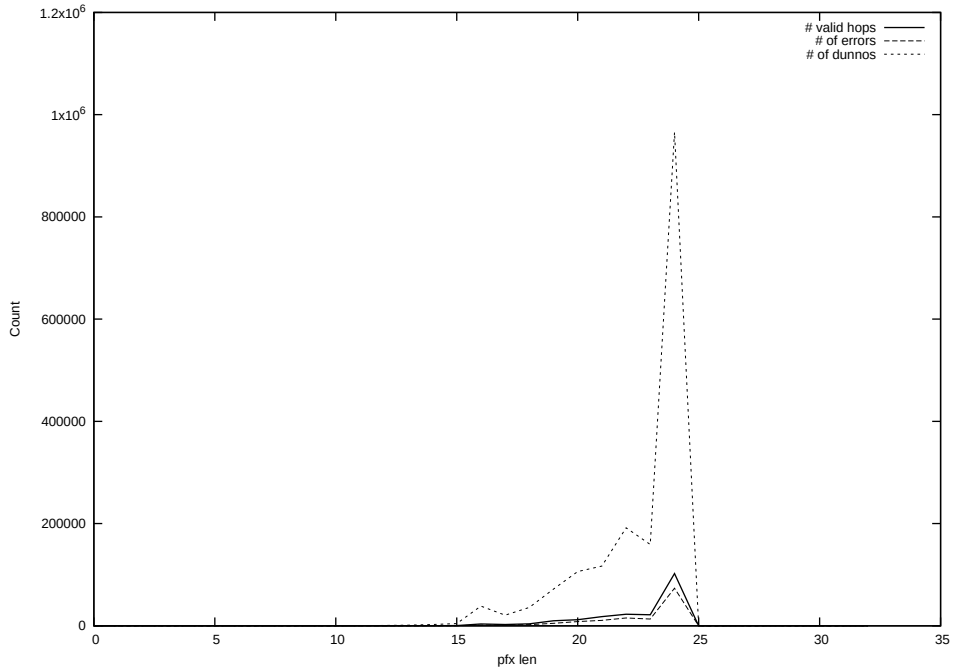
2015-04-13



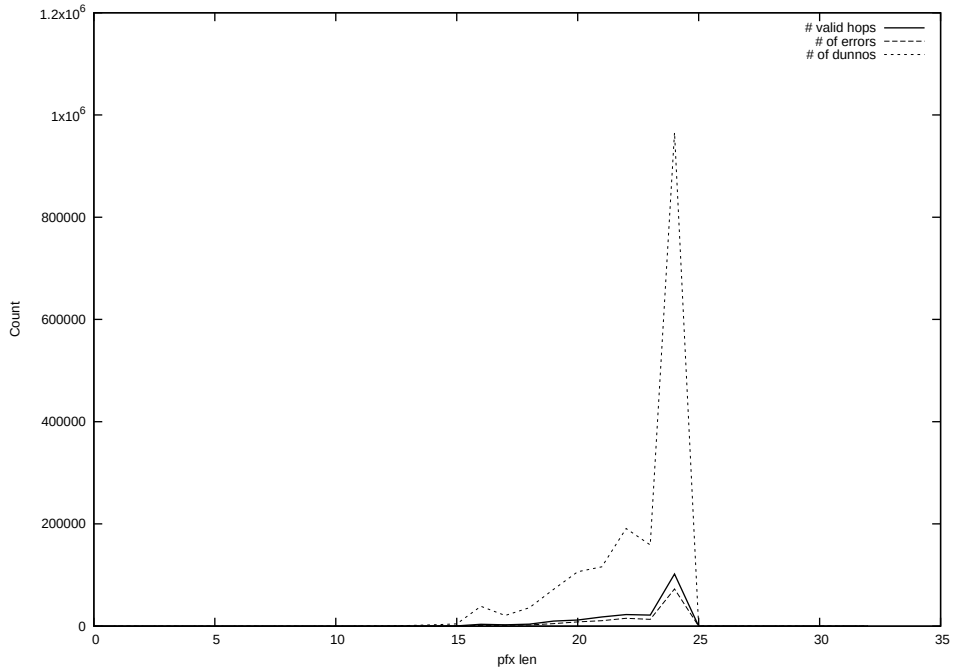
2015-04-14



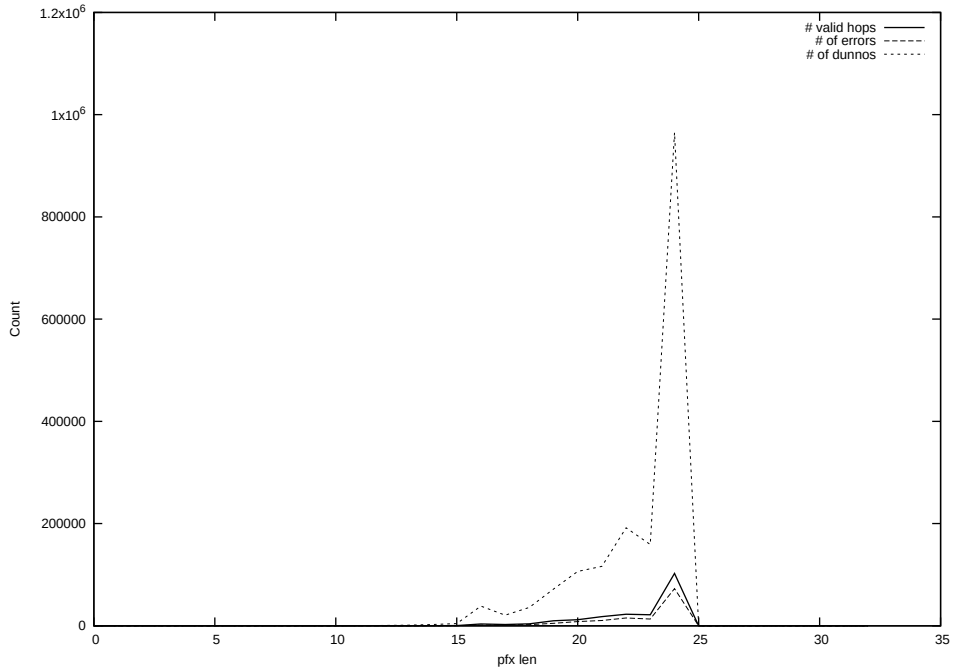
2015-04-15



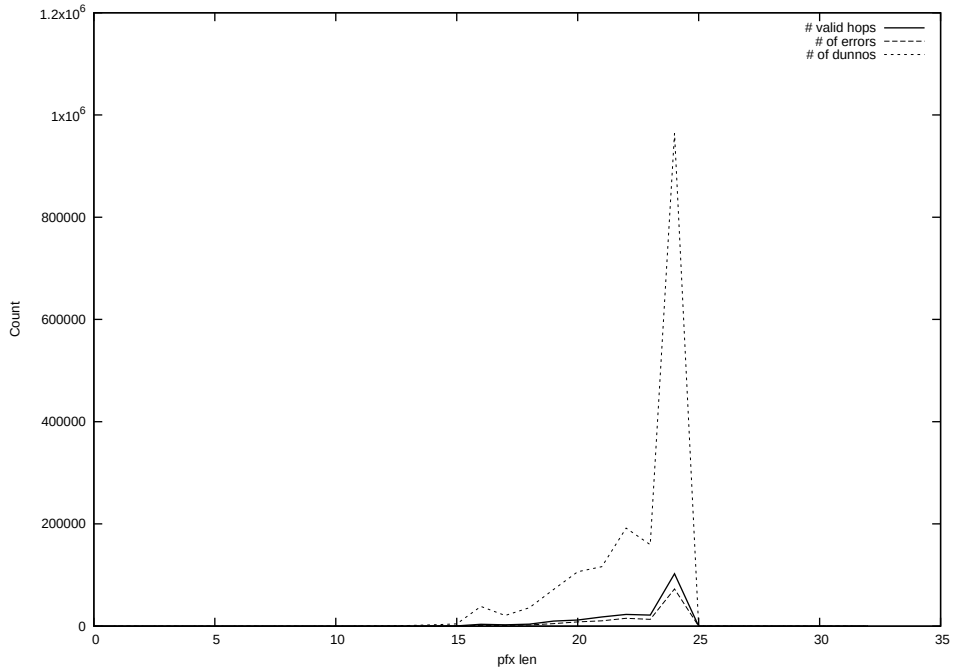
2015-04-16



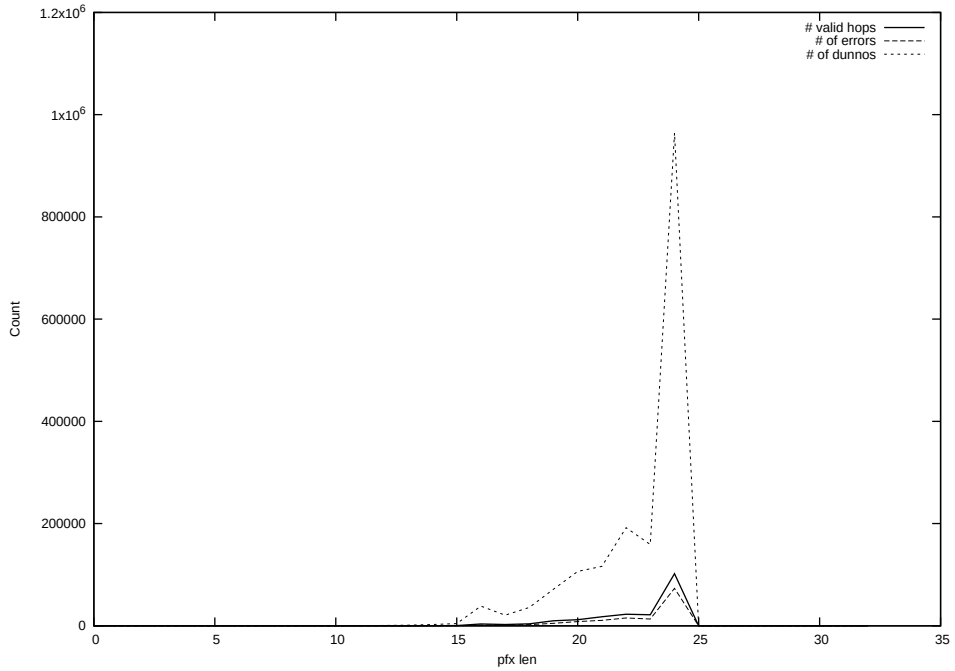
2015-04-17



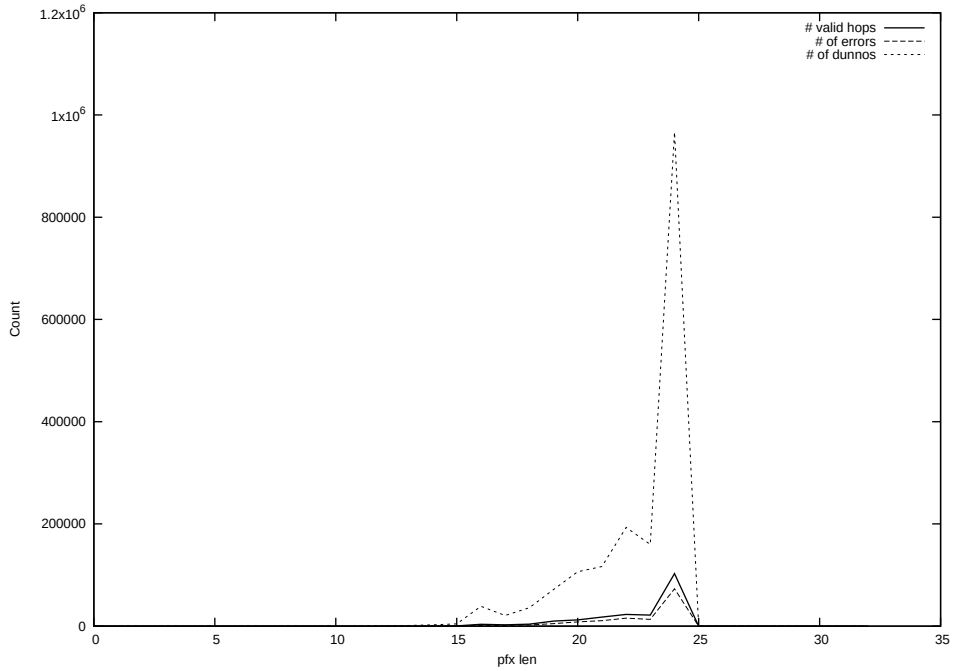
2015-04-18



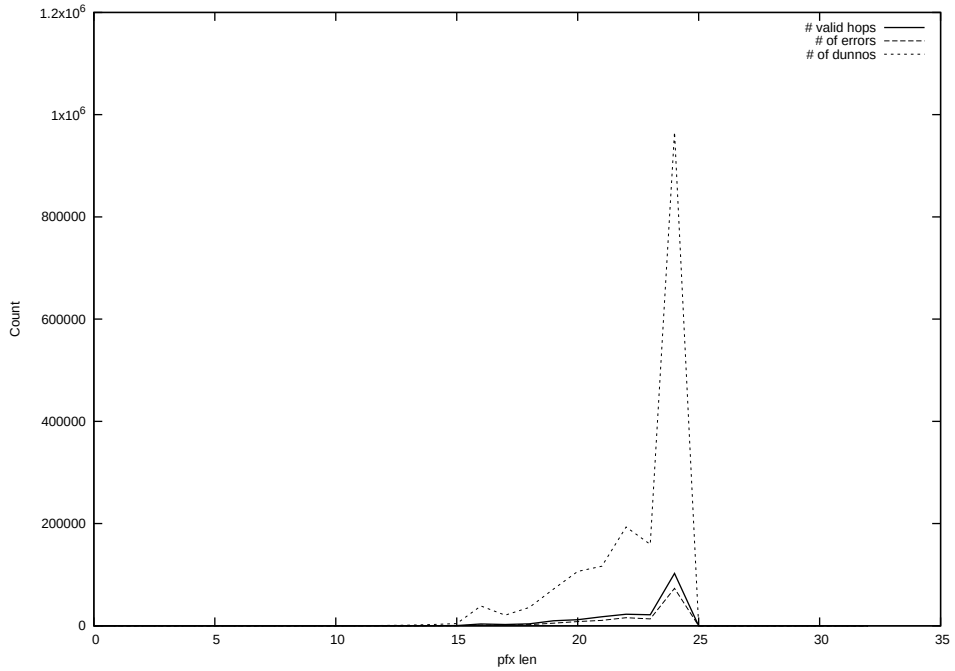
2015-04-19



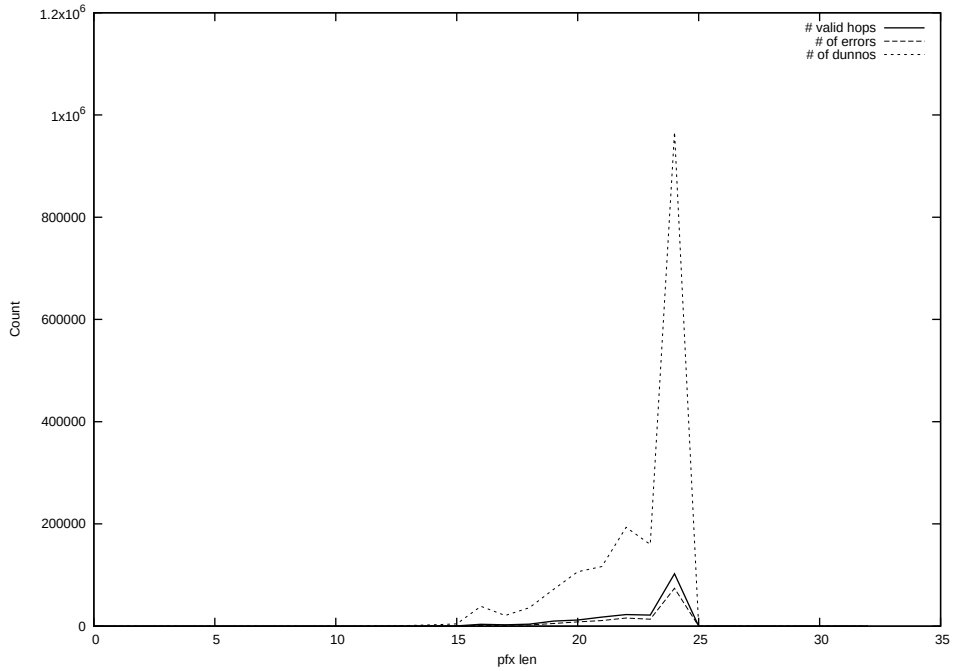
2015-04-20



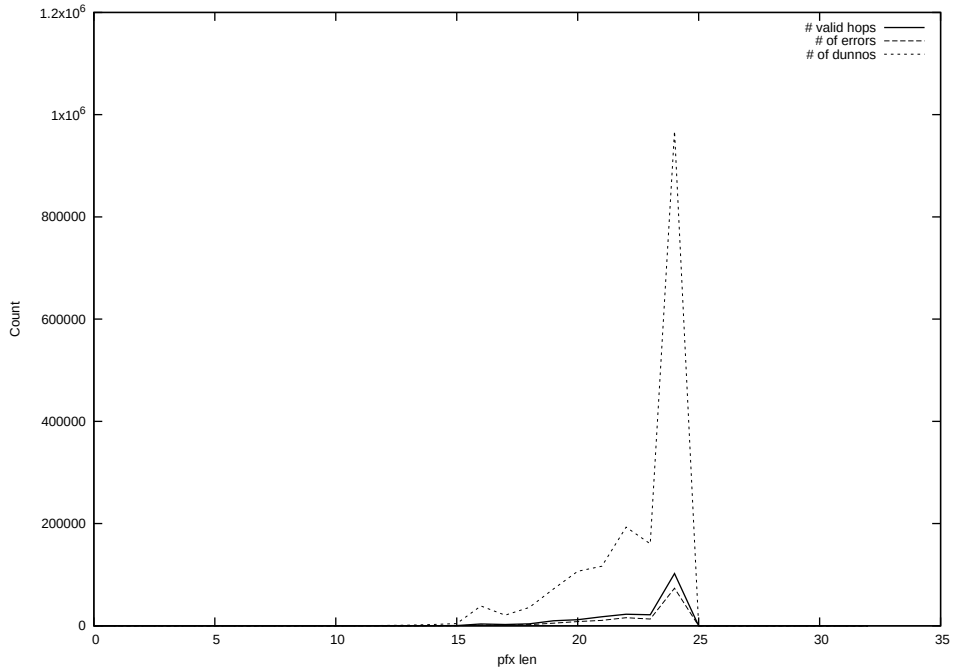
2015-04-21



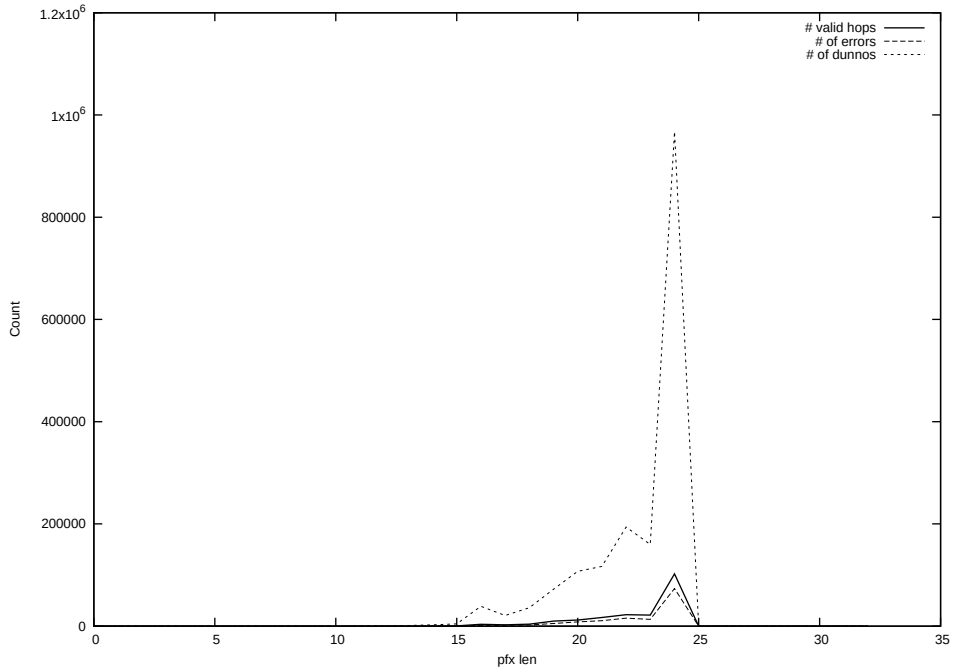
2015-04-22



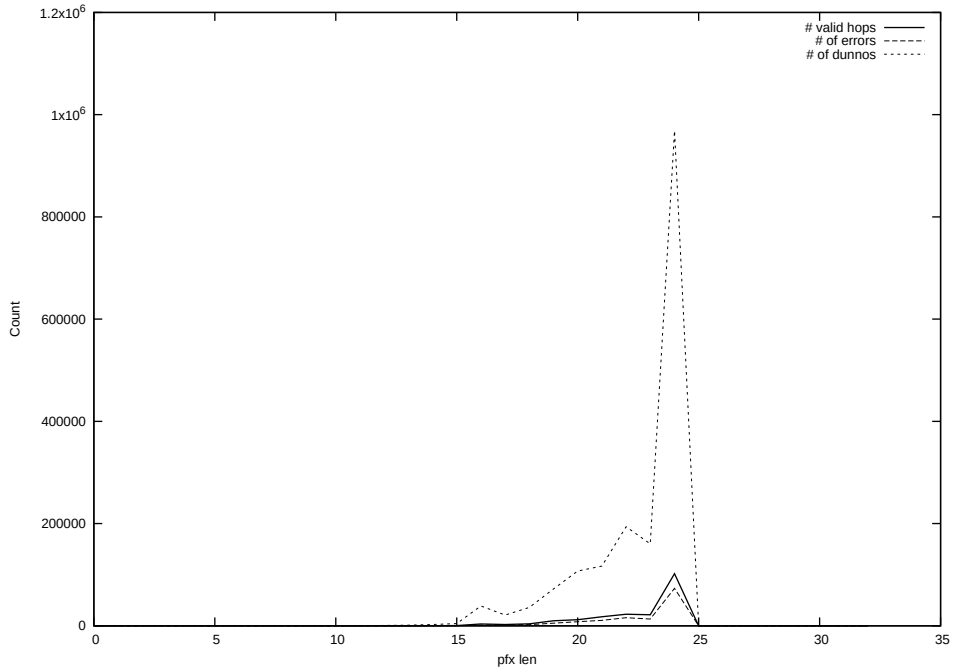
2015-04-23



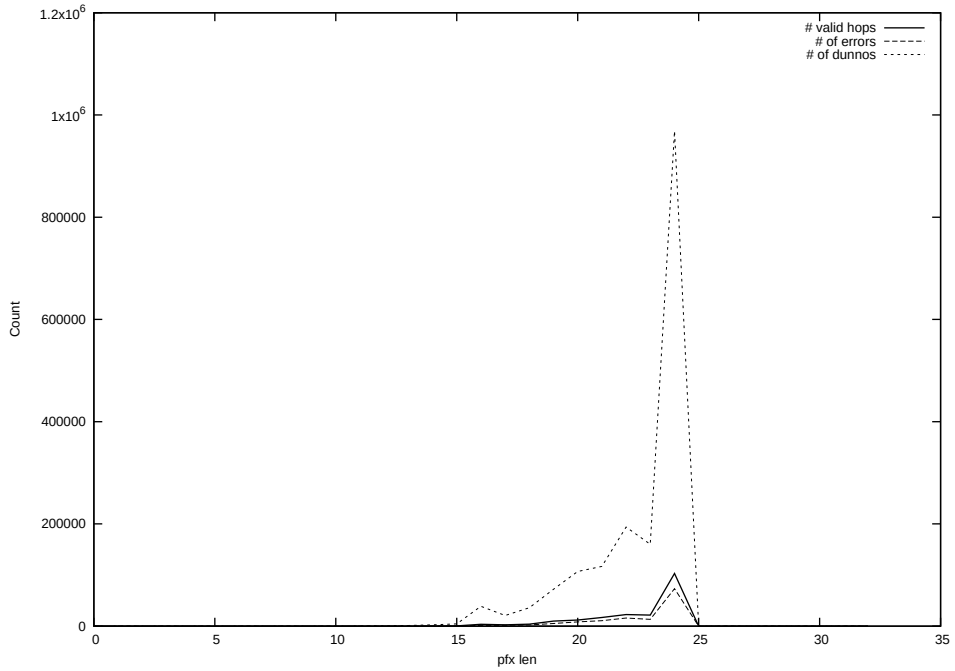
2015-04-24



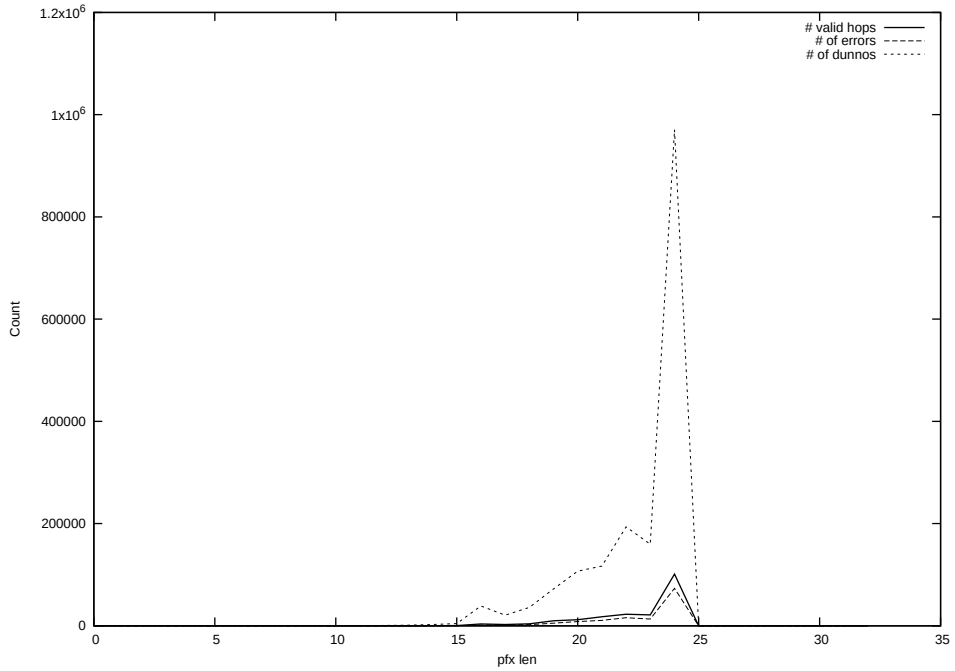
2015-04-25



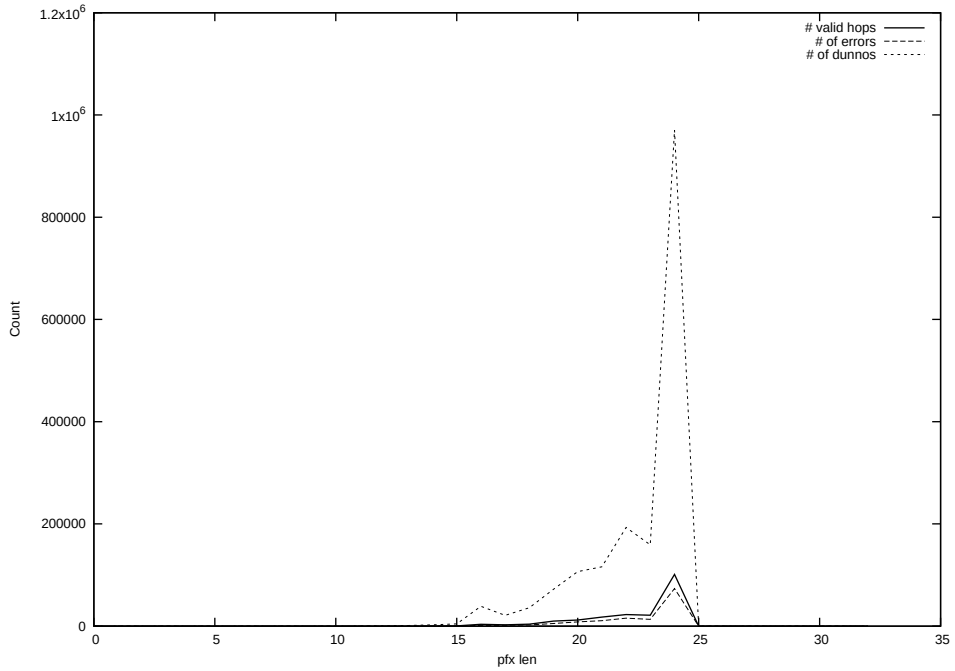
2015-04-26



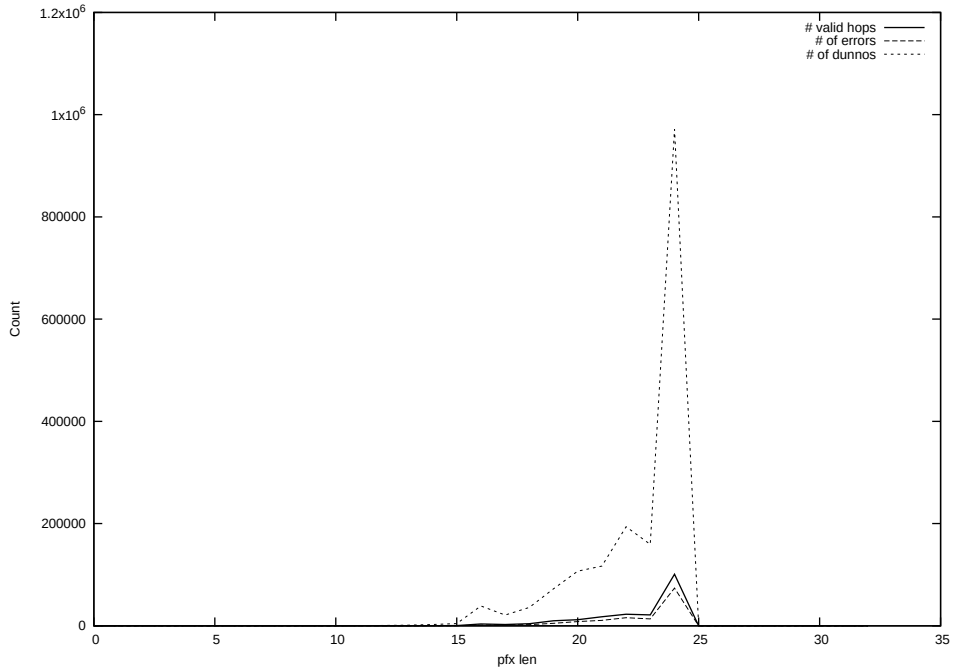
2015-04-27



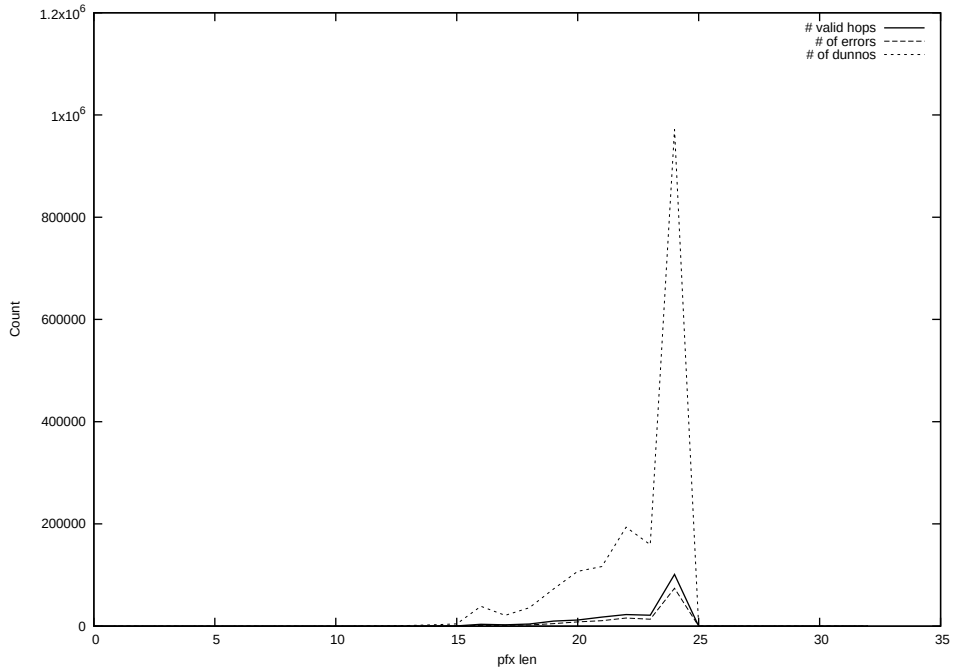
2015-04-28



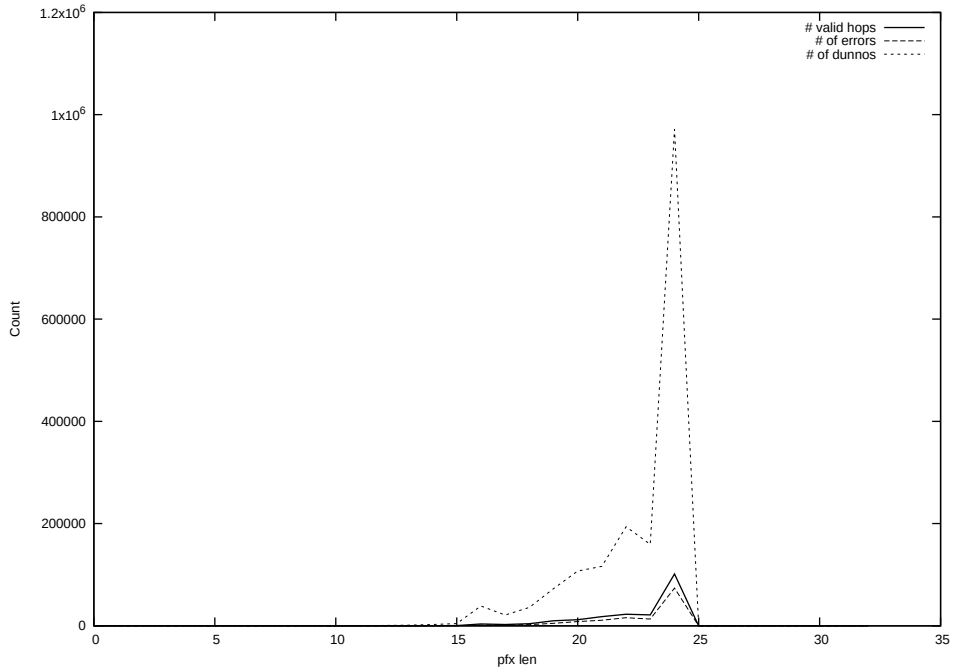
2015-04-29



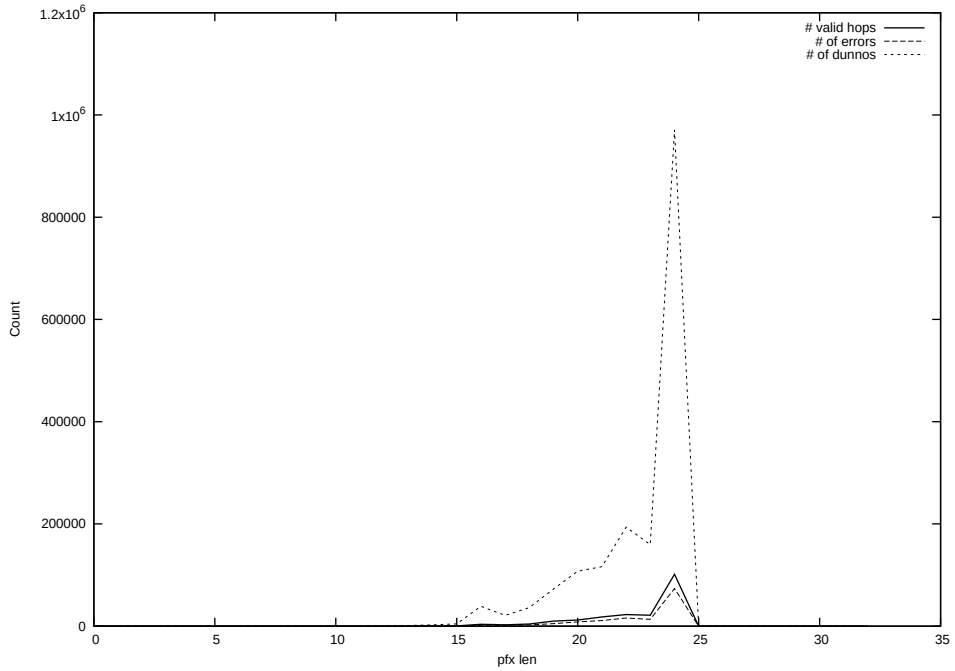
2015-04-30



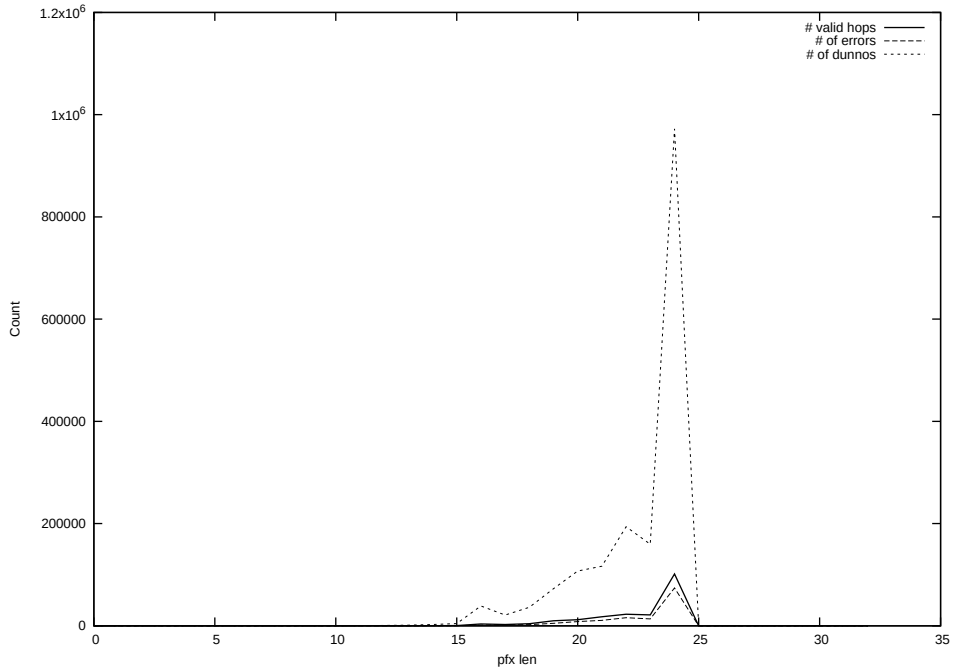
2015-05-01



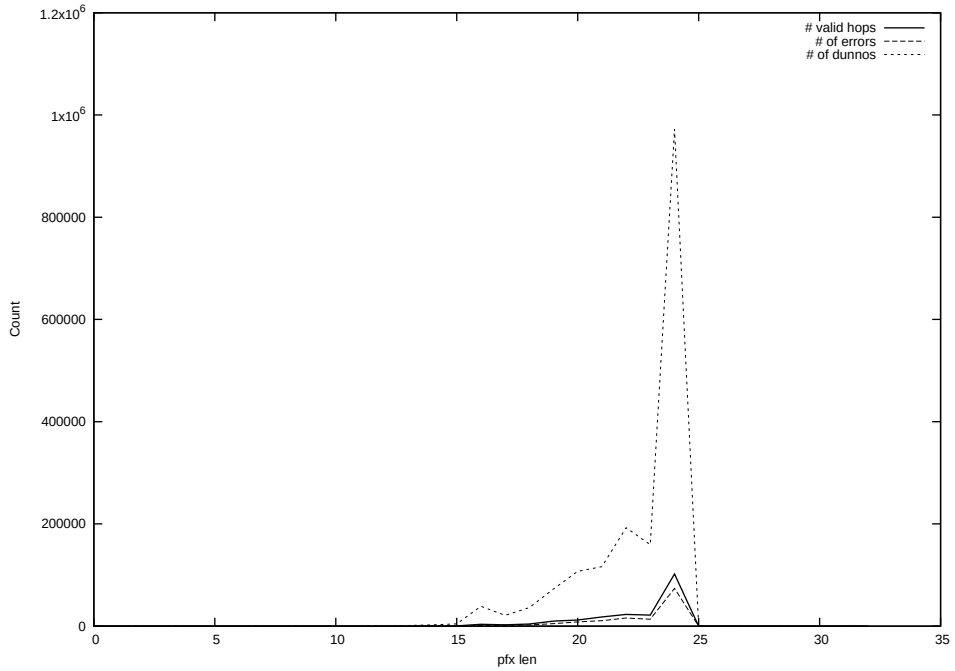
2015-05-02



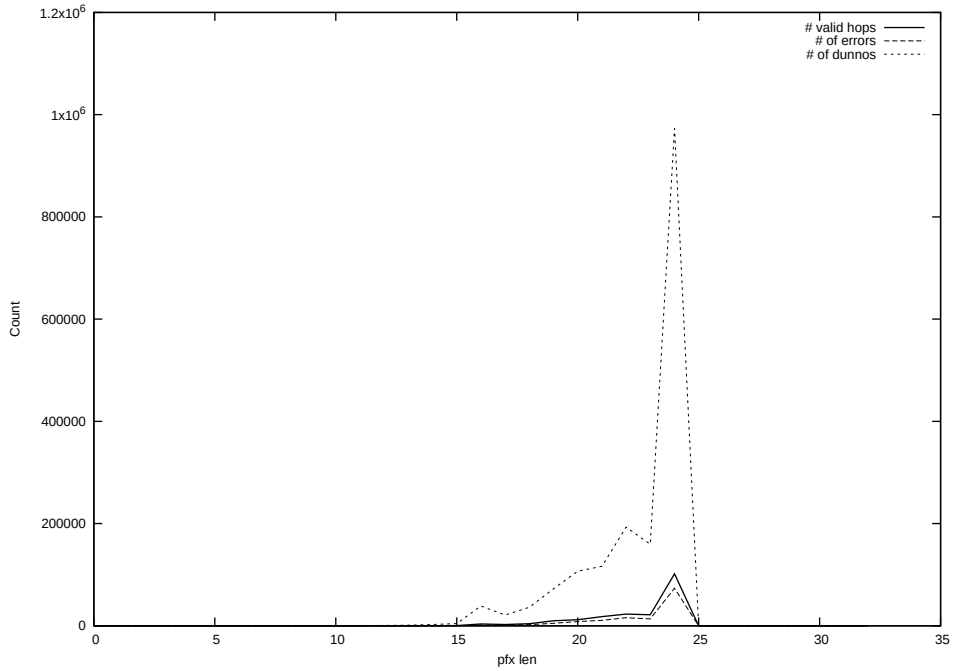
2015-05-03



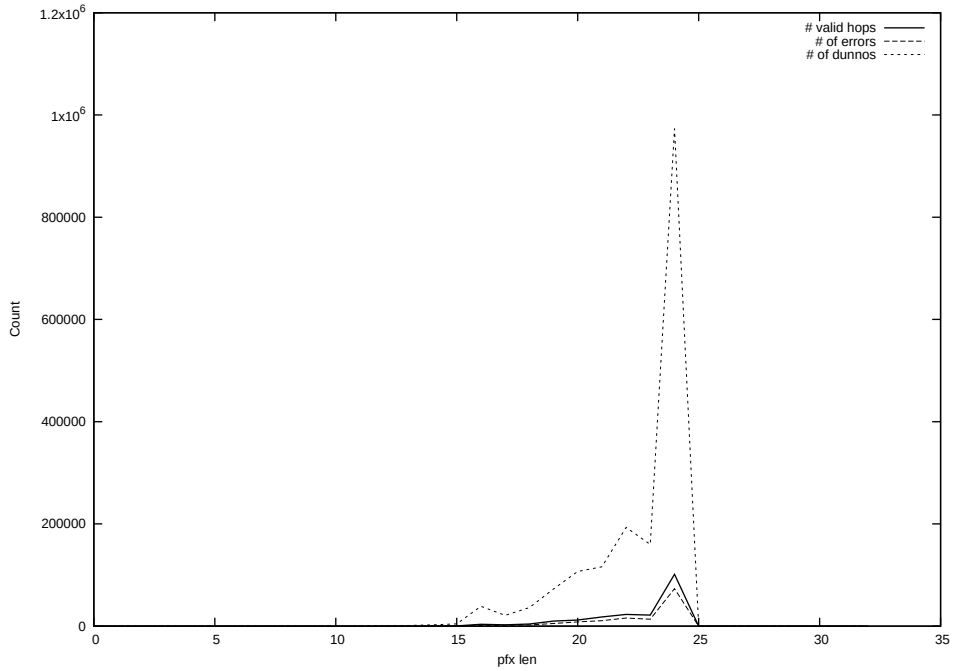
2015-05-04



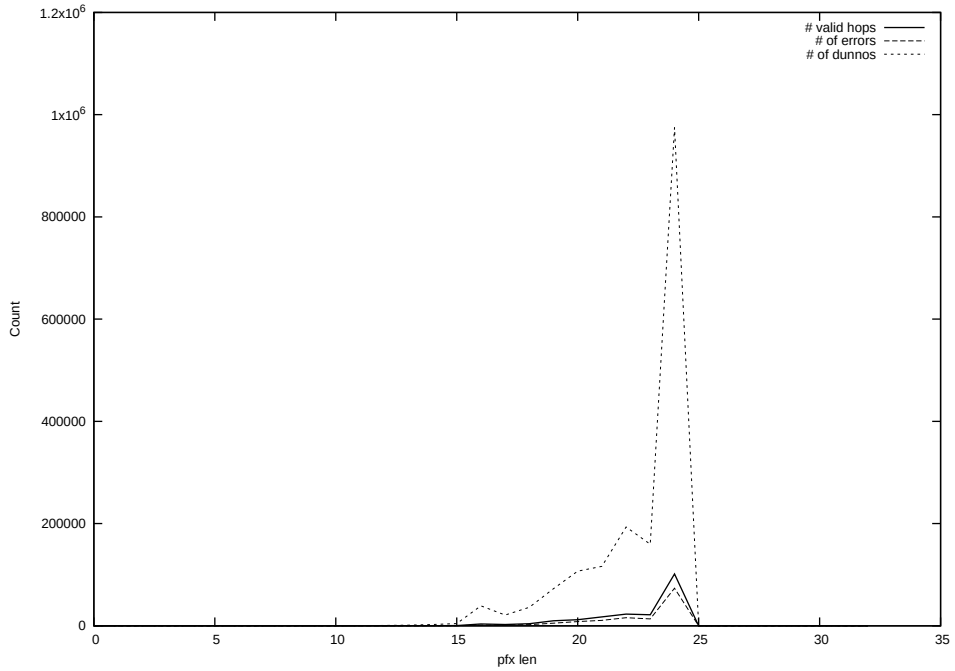
2015-05-05



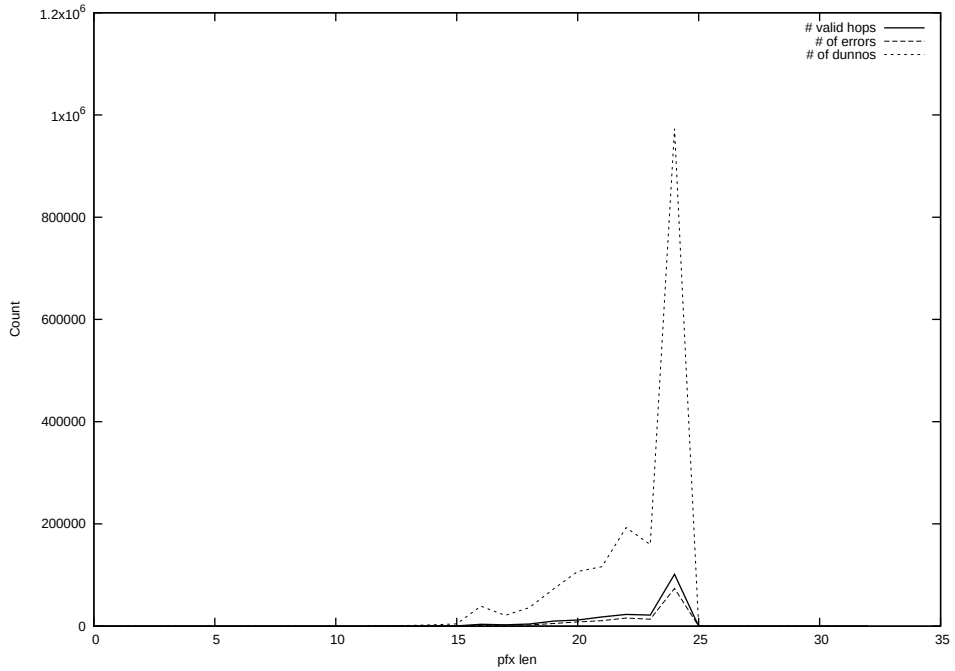
2015-05-06



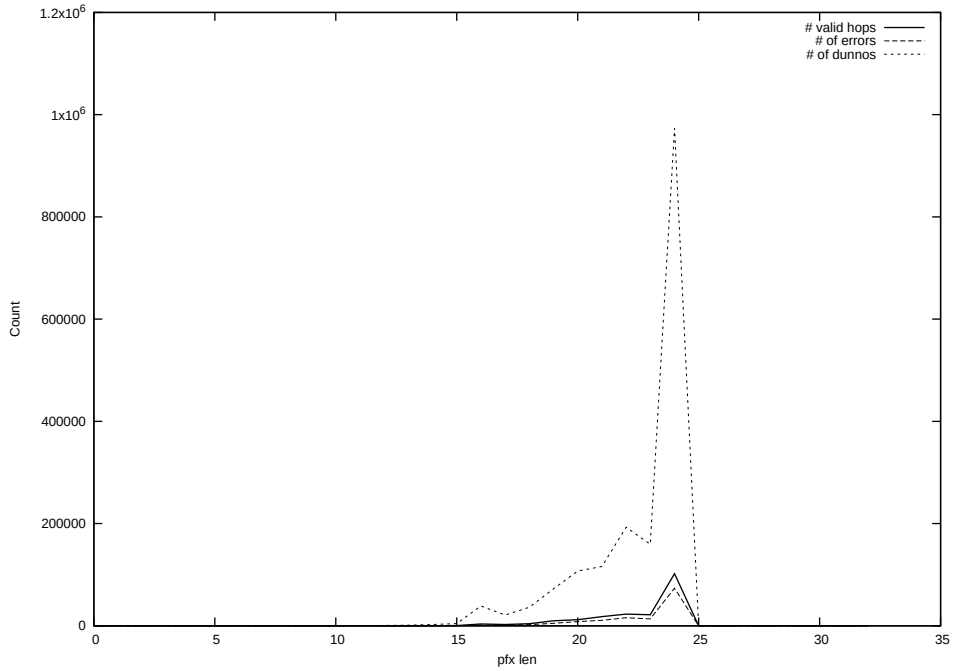
2015-05-07



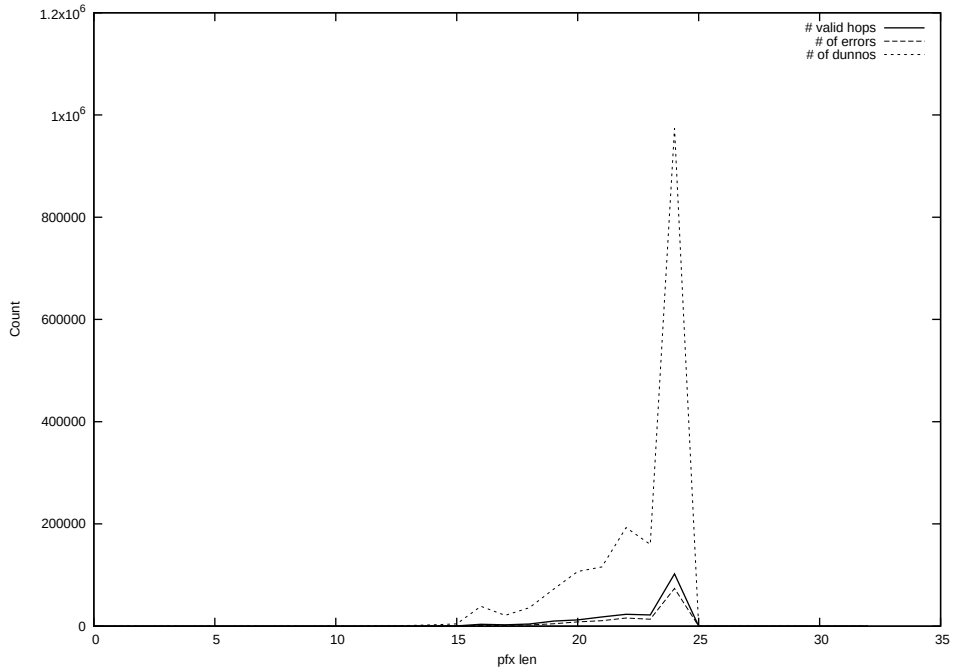
2015-05-08



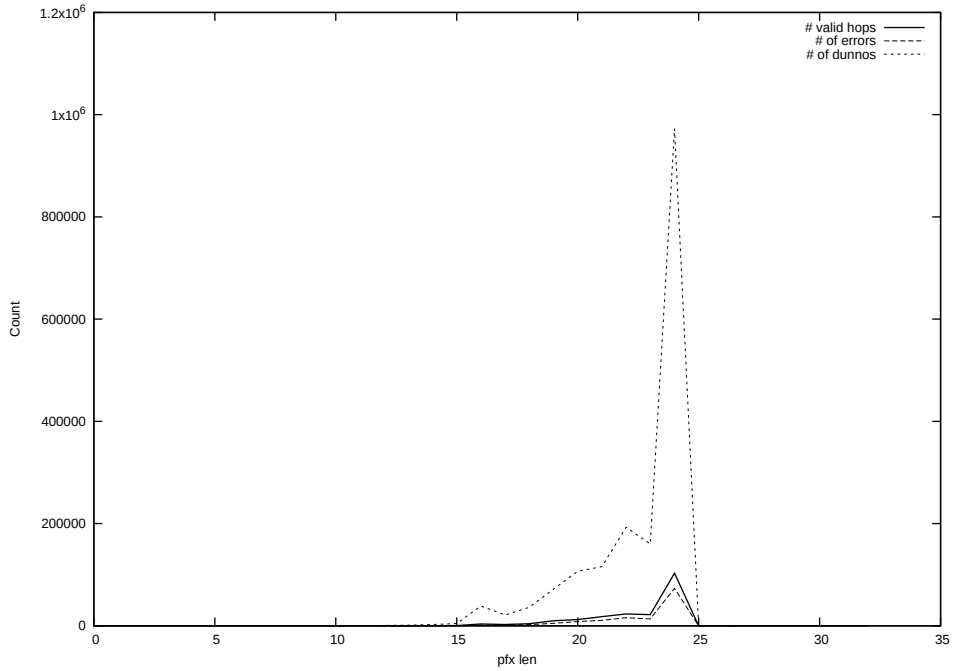
2015-05-09



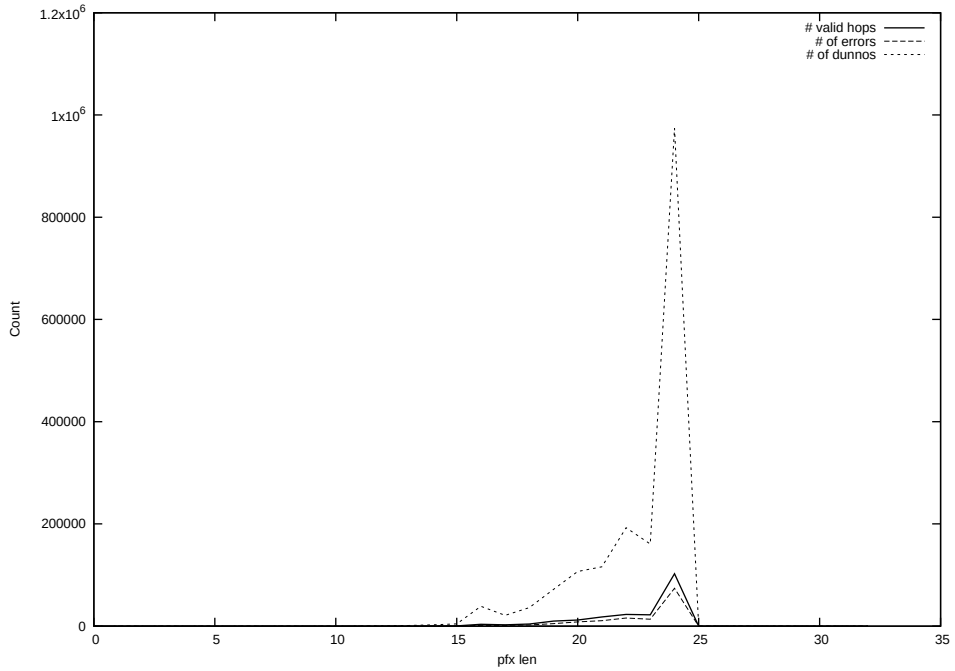
2015-05-10



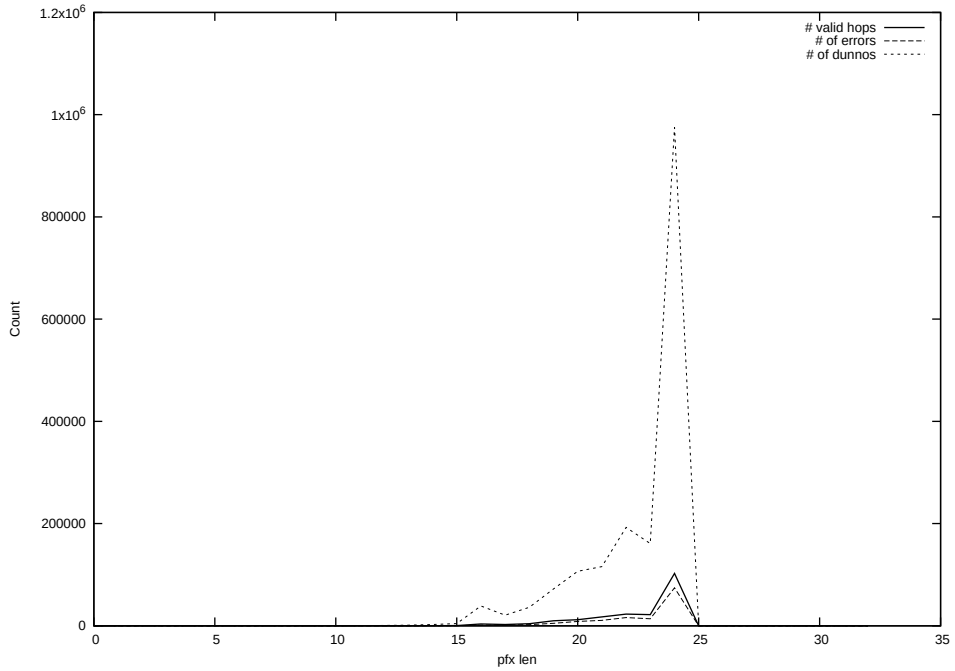
2015-05-11



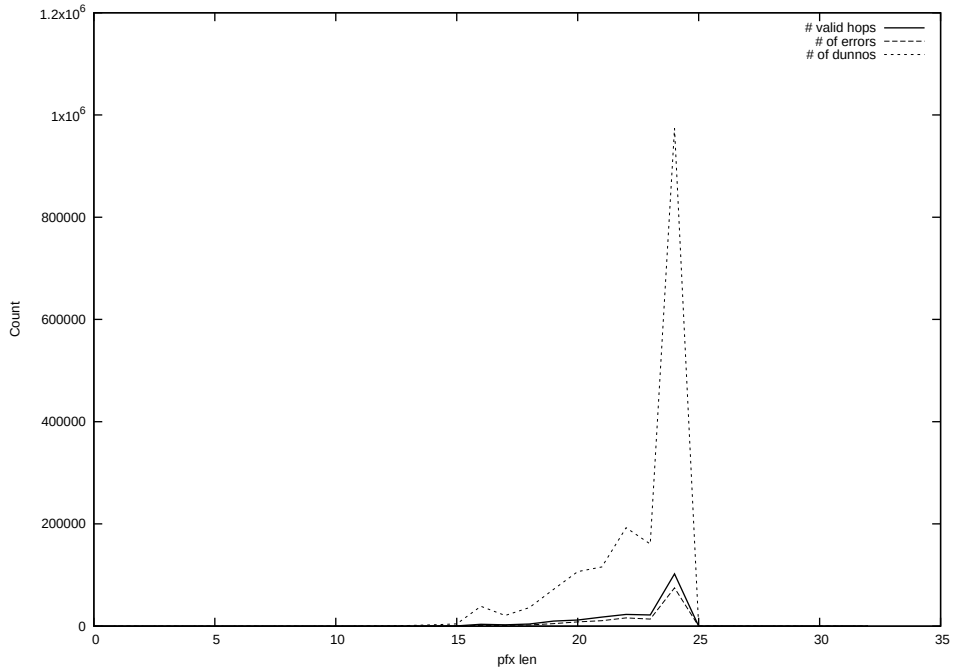
2015-05-12



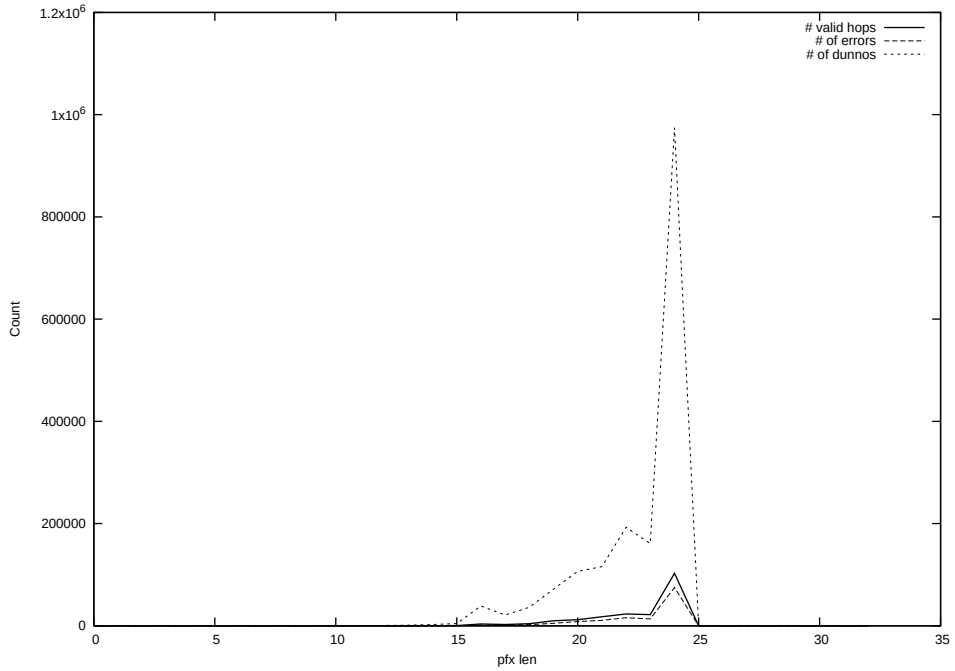
2015-05-13



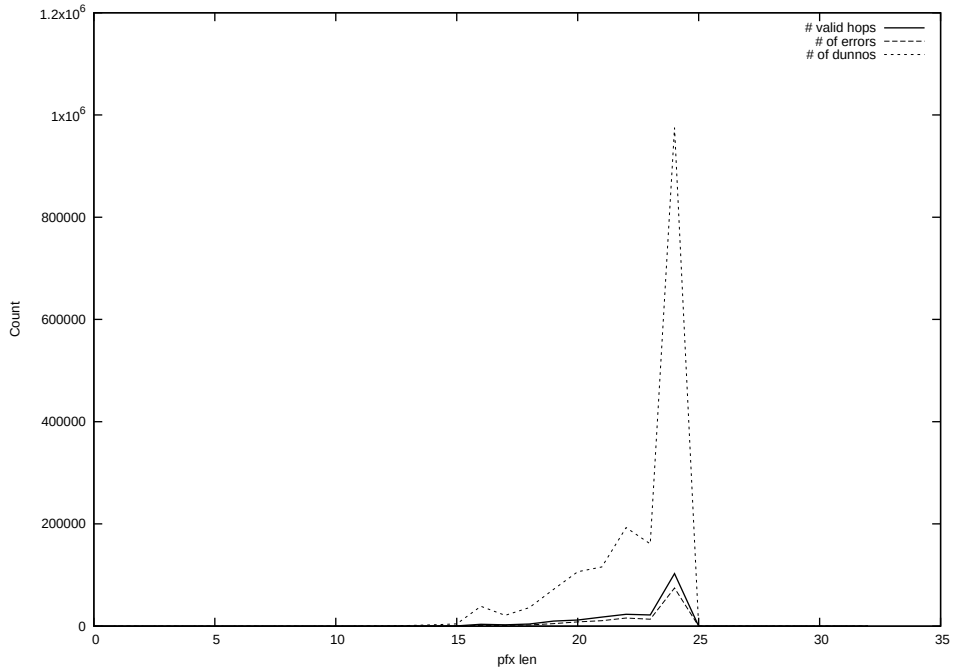
2015-05-14



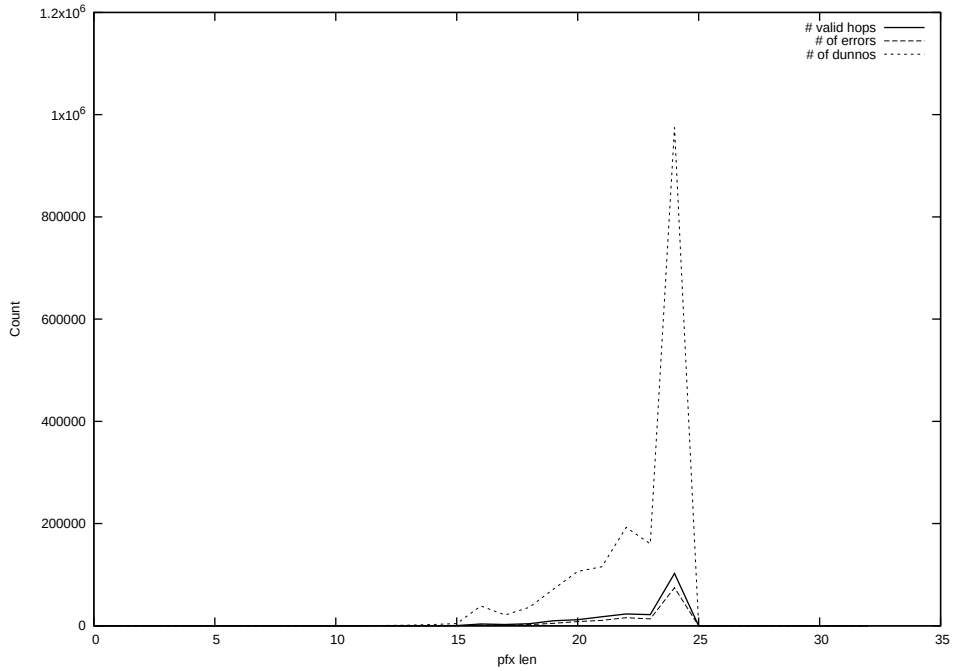
2015-05-15



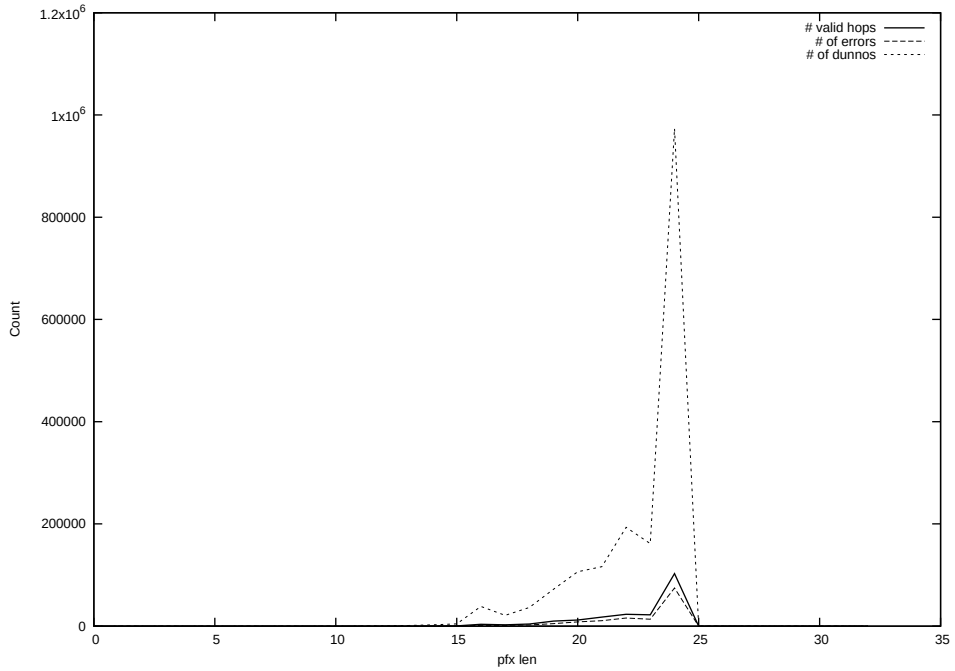
2015-05-16



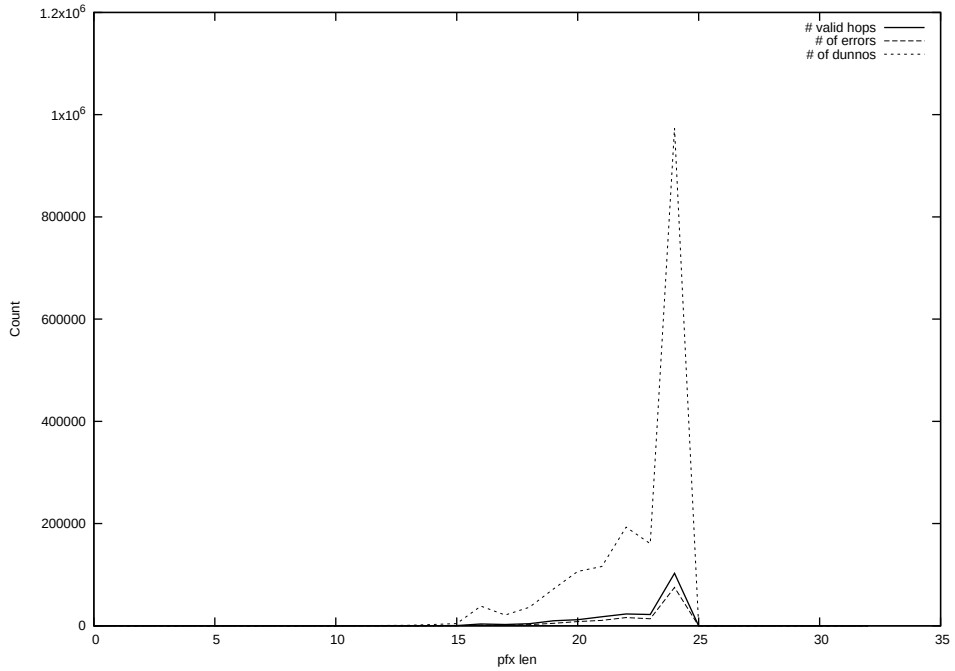
2015-05-17



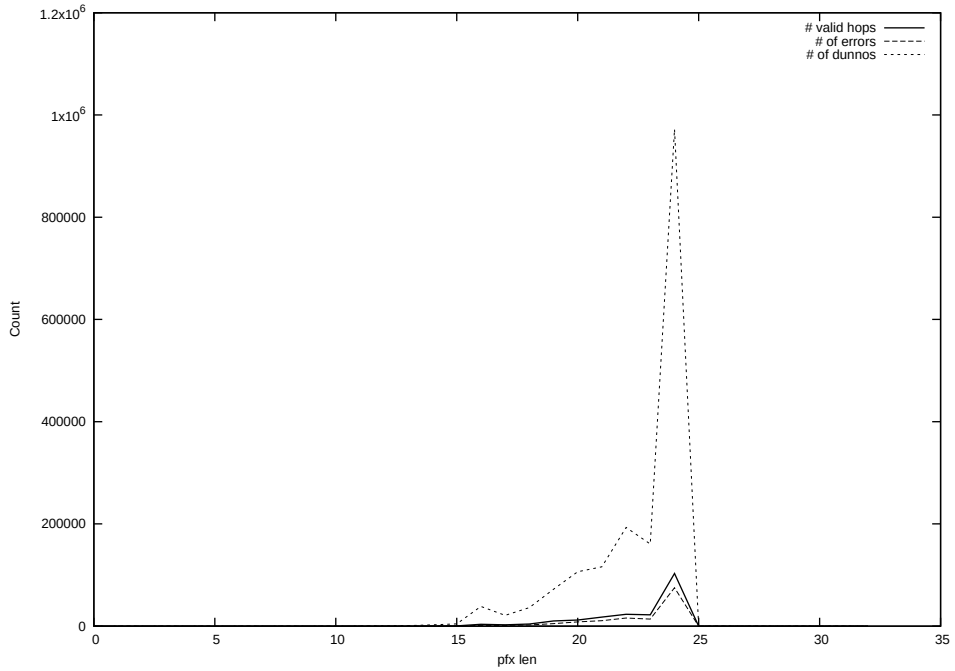
2015-05-18



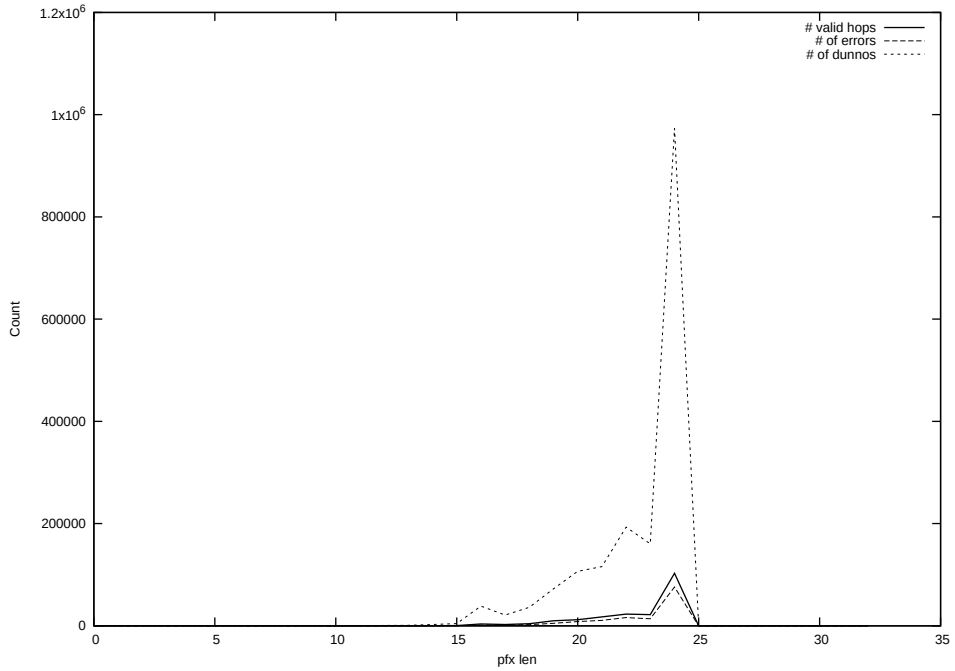
2015-05-19



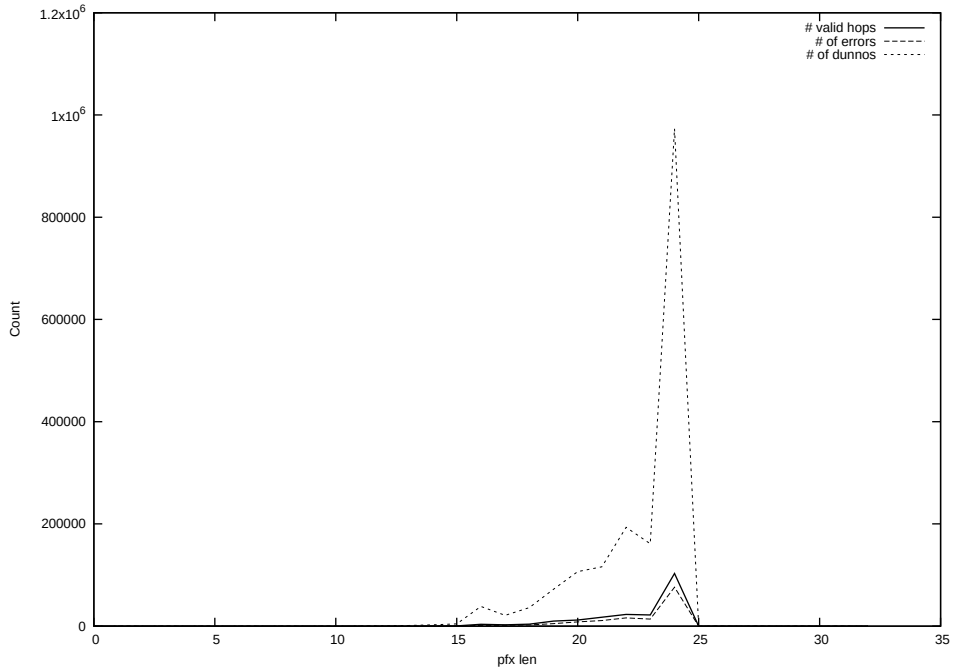
2015-05-20



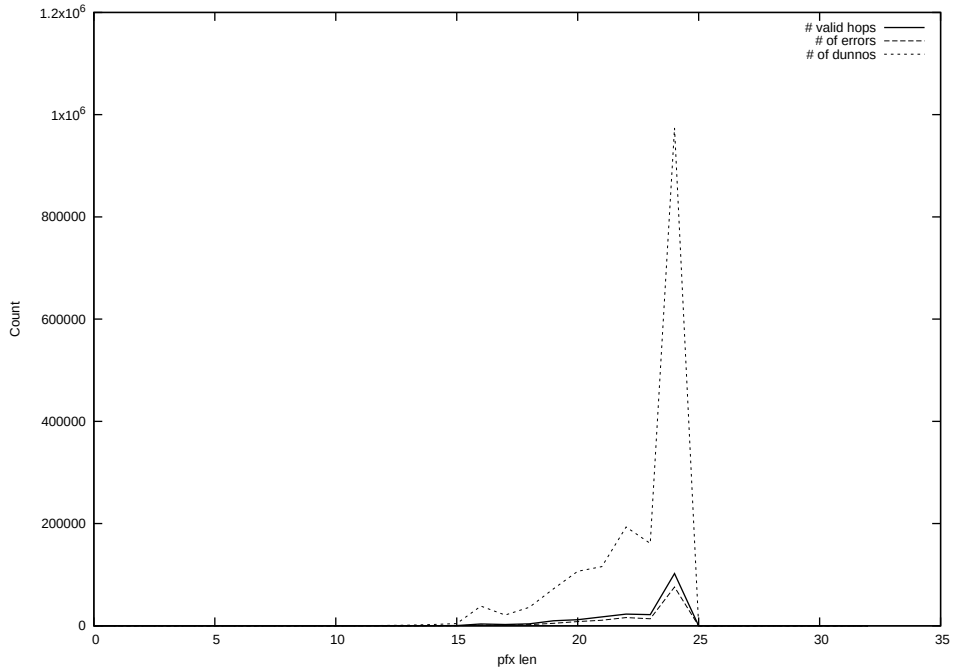
2015-05-21



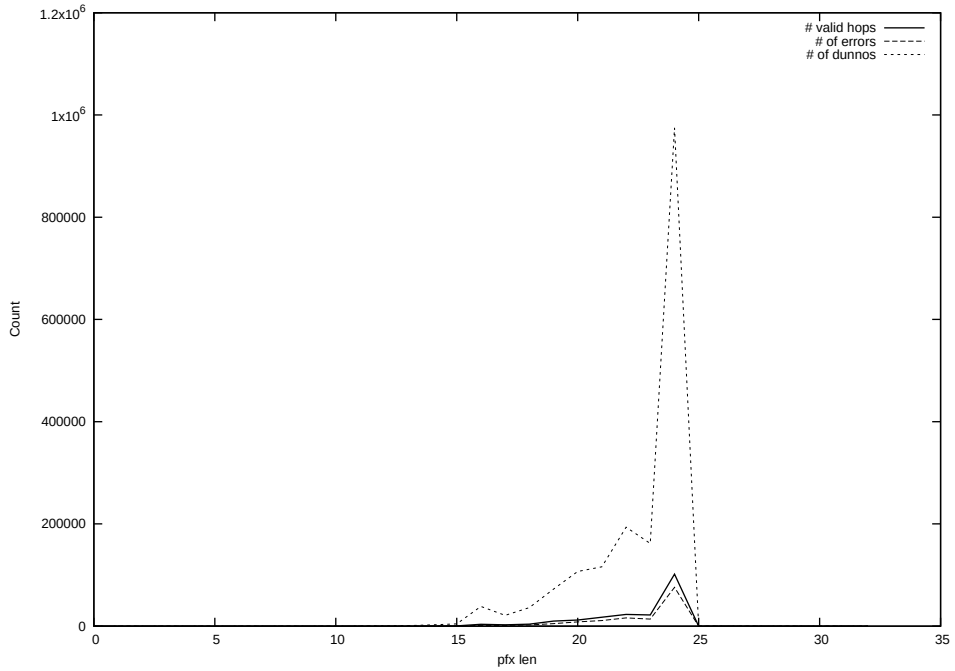
2015-05-22



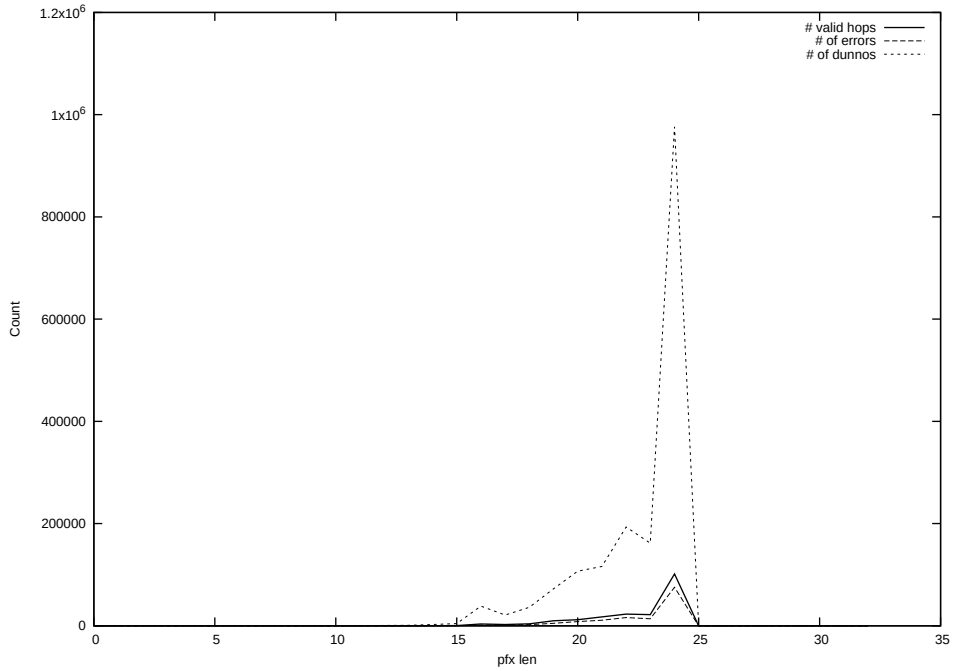
2015-05-23



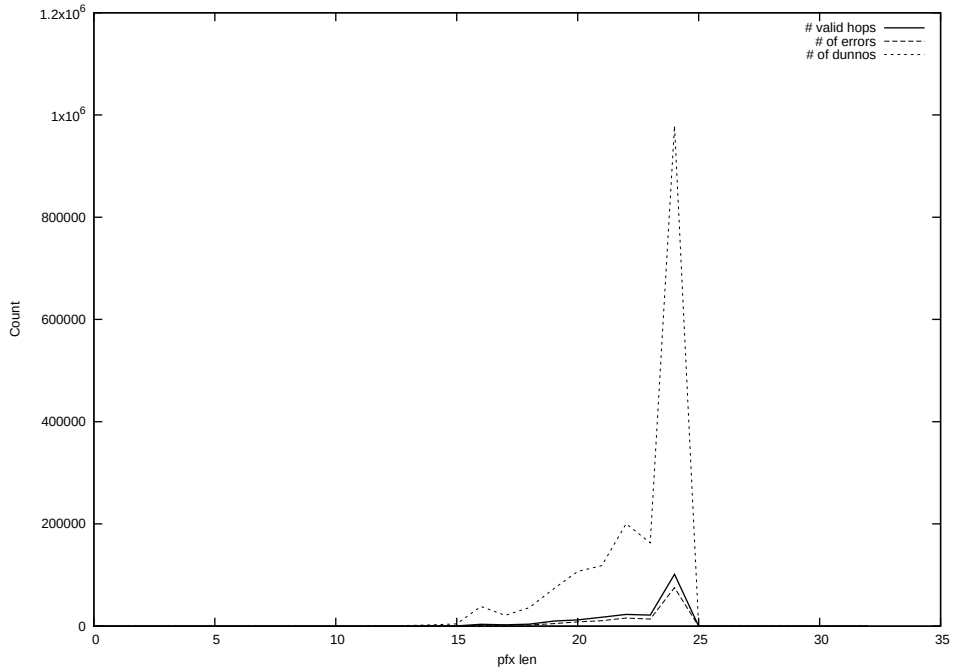
2015-05-24



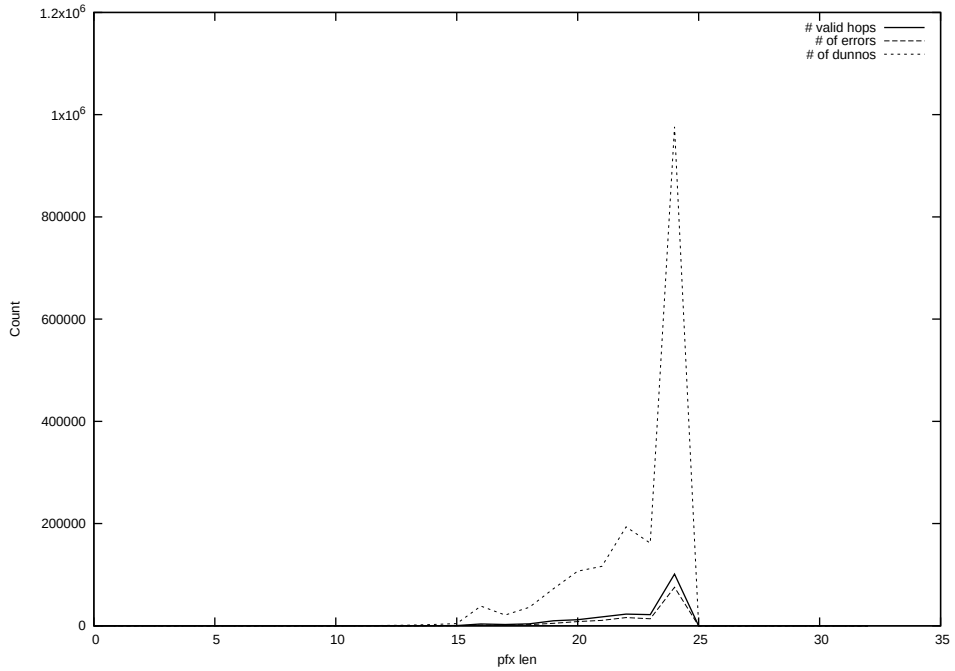
2015-05-25



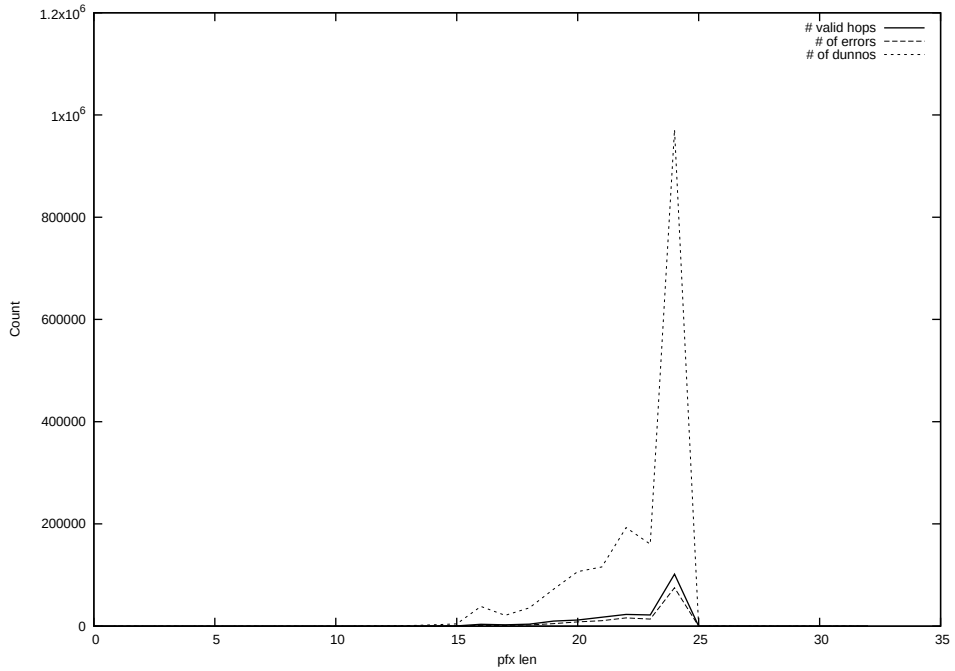
2015-05-26



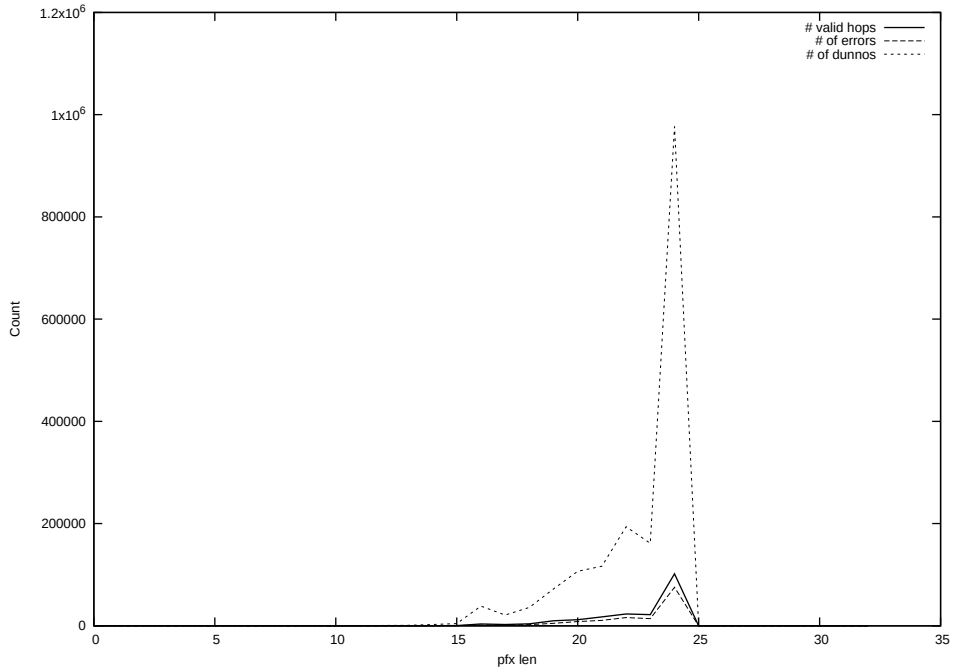
2015-05-27



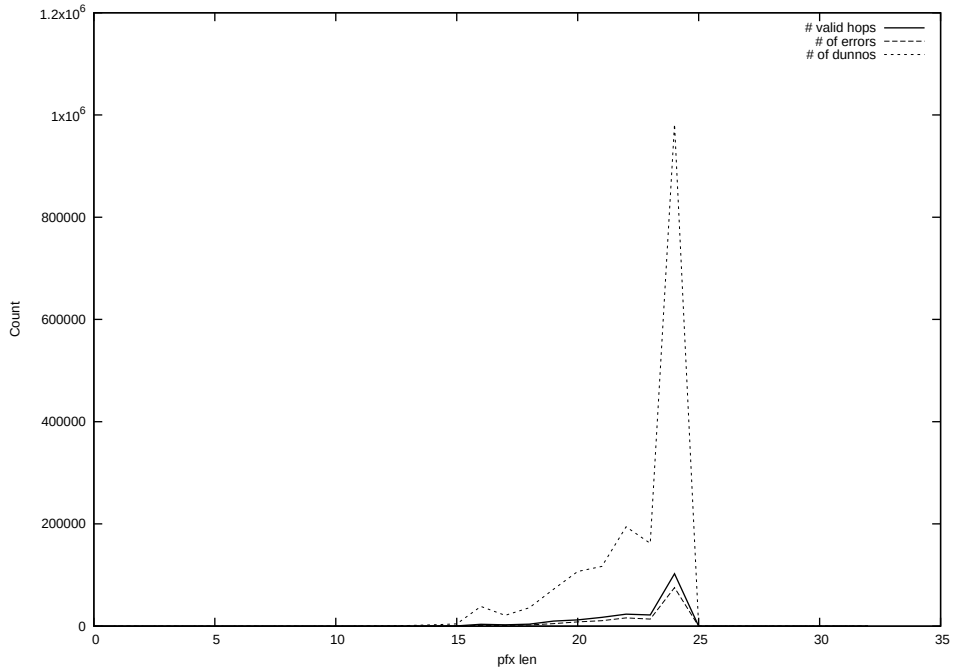
2015-05-28



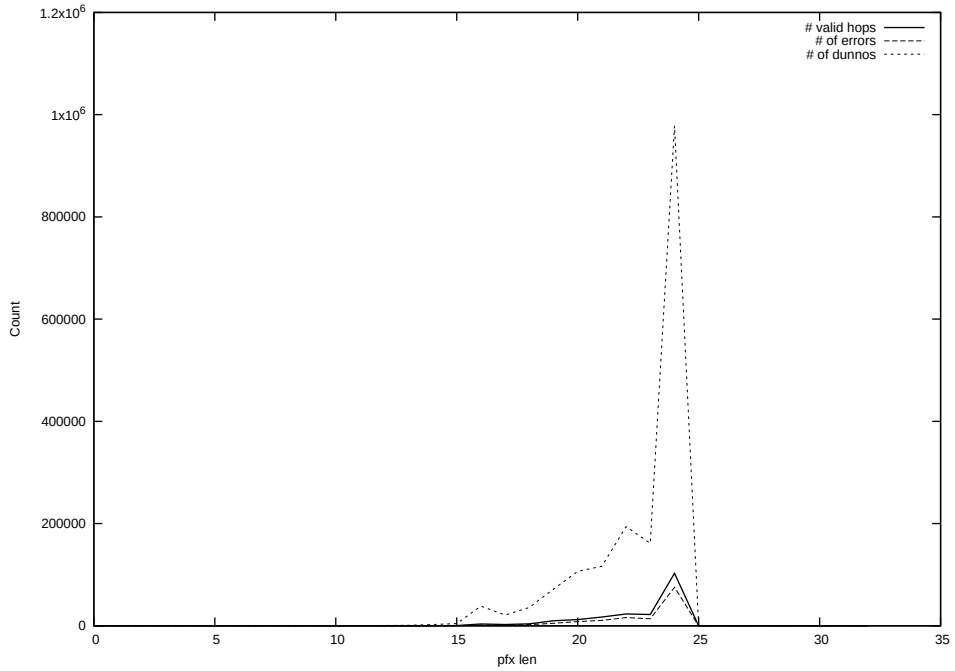
2015-05-29



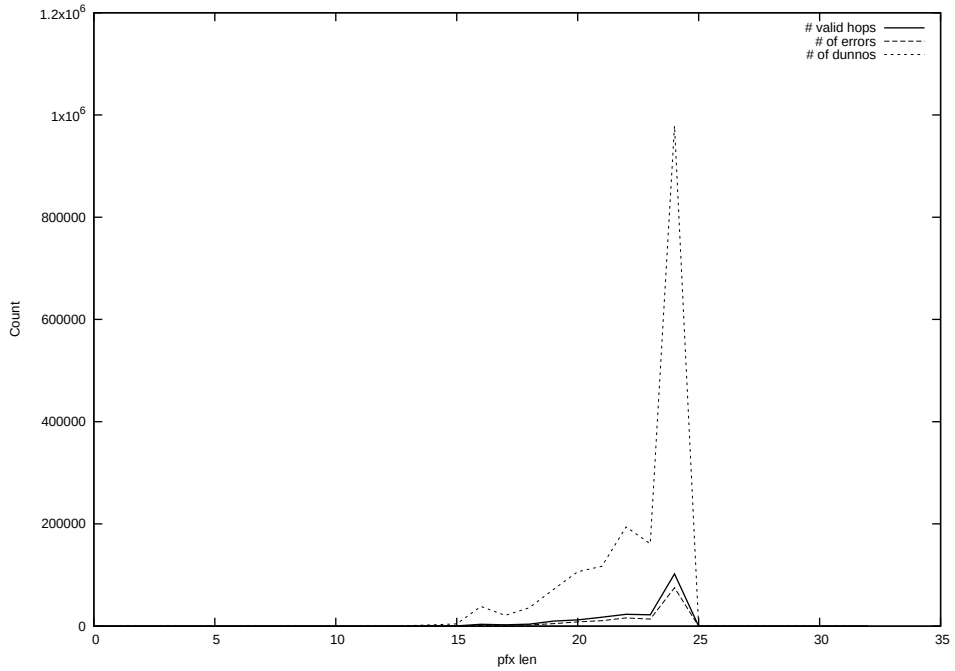
2015-05-30



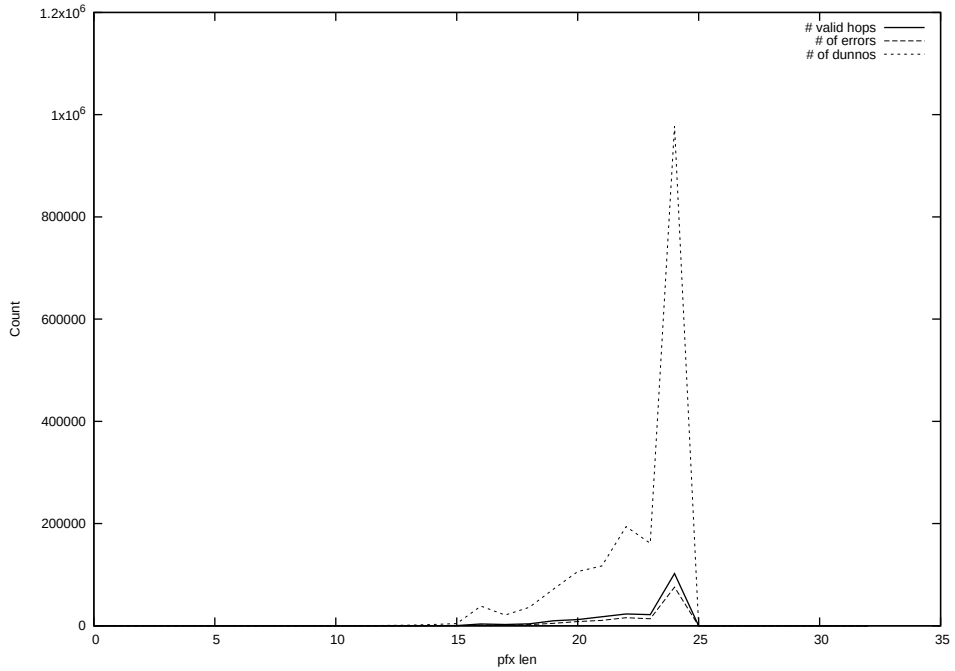
2015-05-31



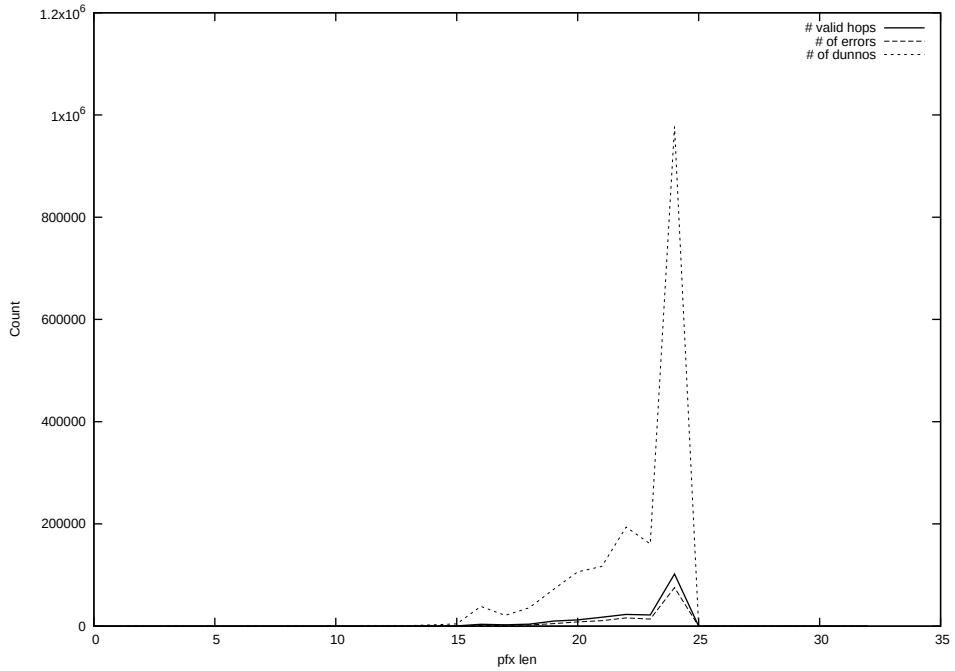
2015-06-01



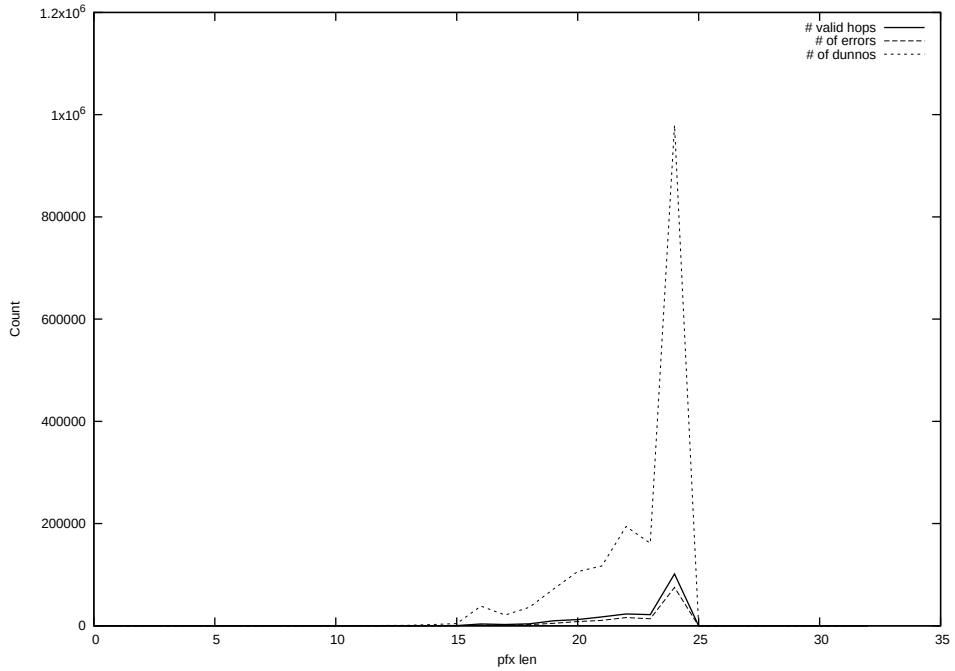
2015-06-02



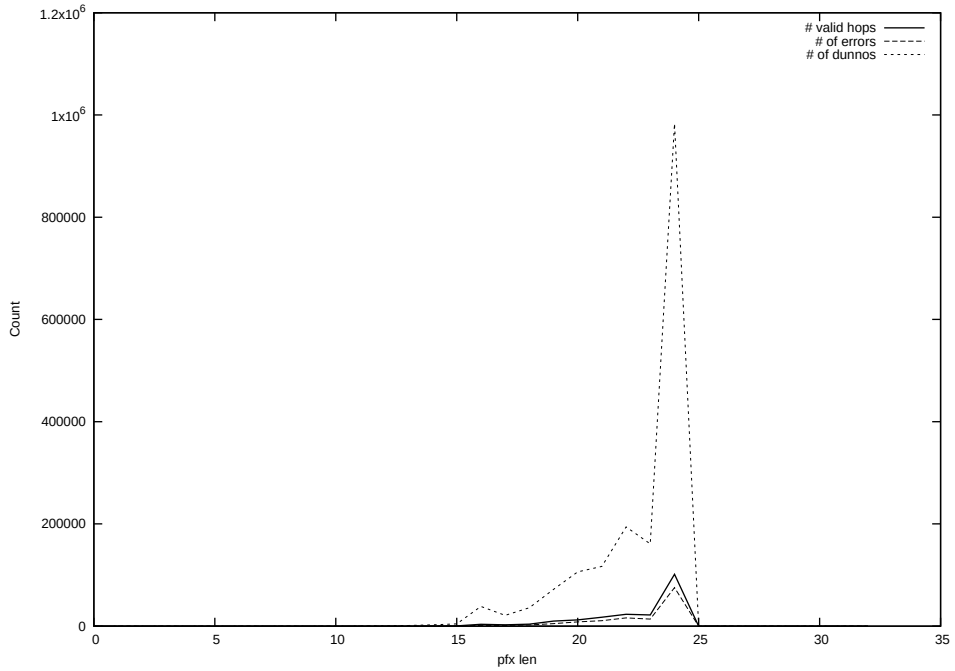
2015-06-03



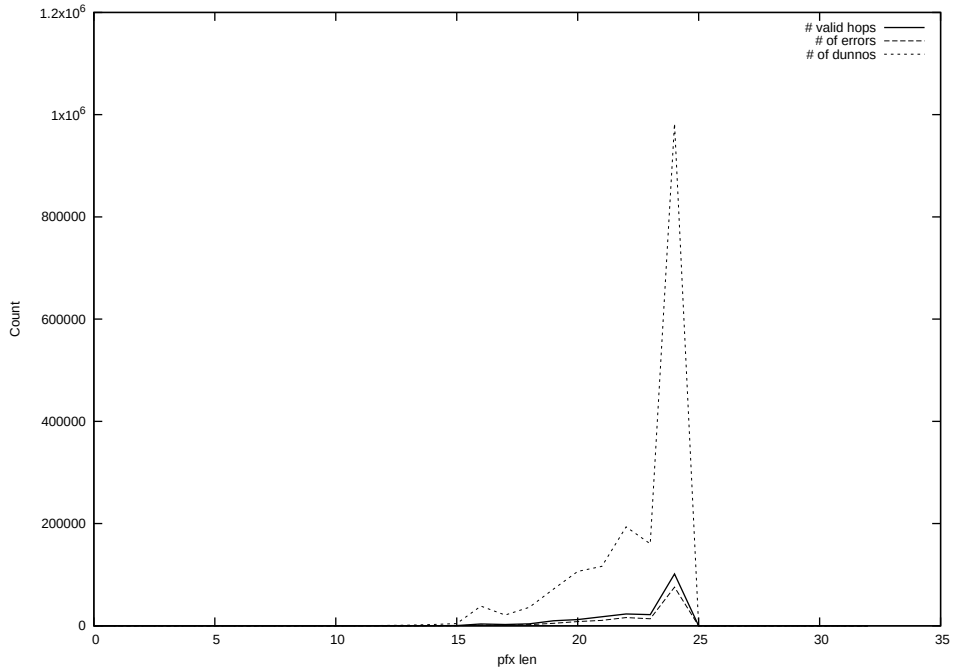
2015-06-04



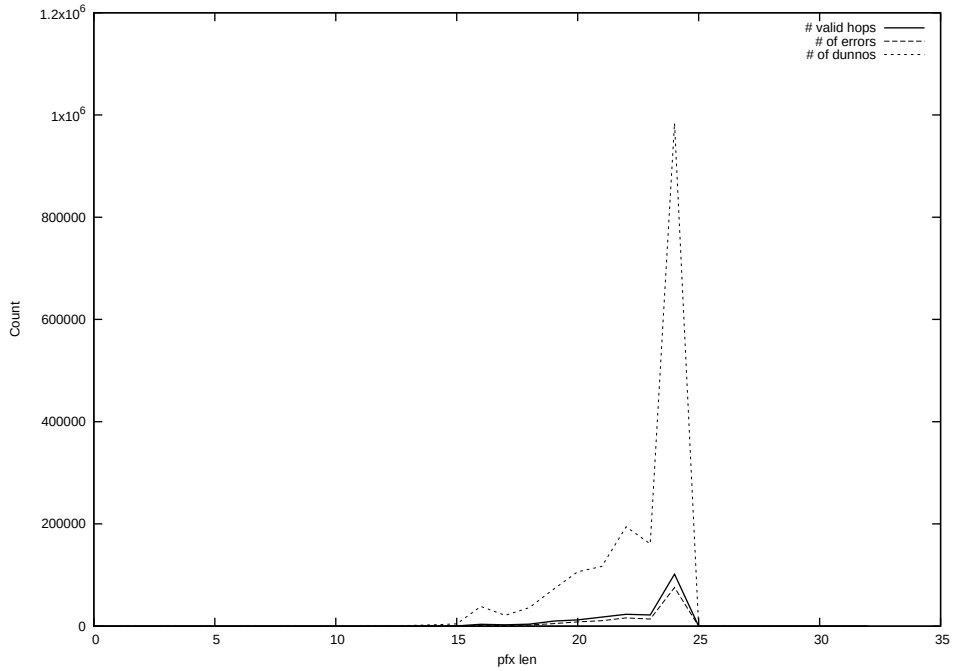
2015-06-05



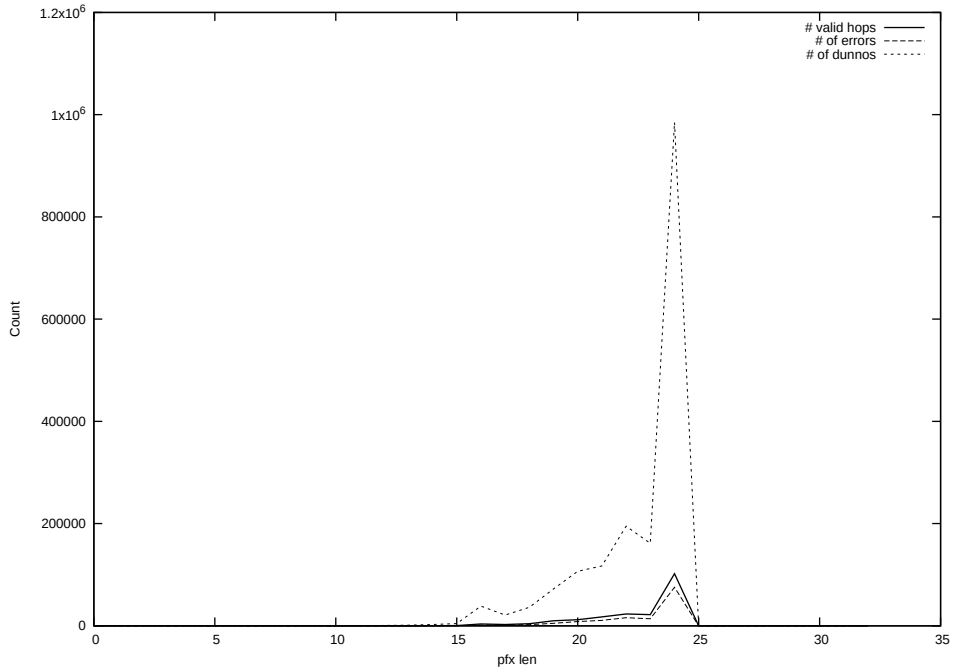
2015-06-06



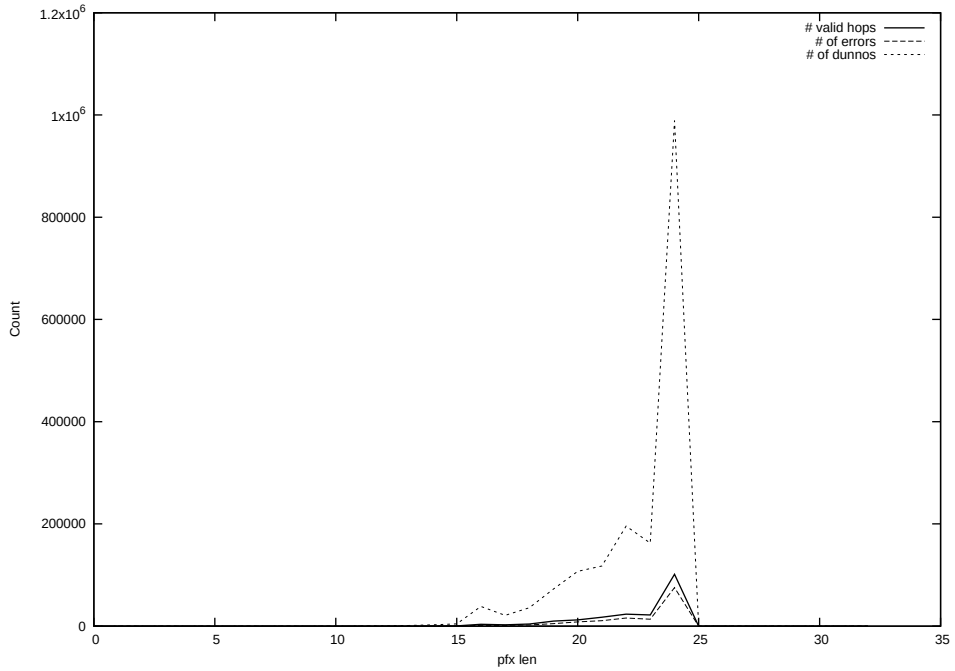
2015-06-07



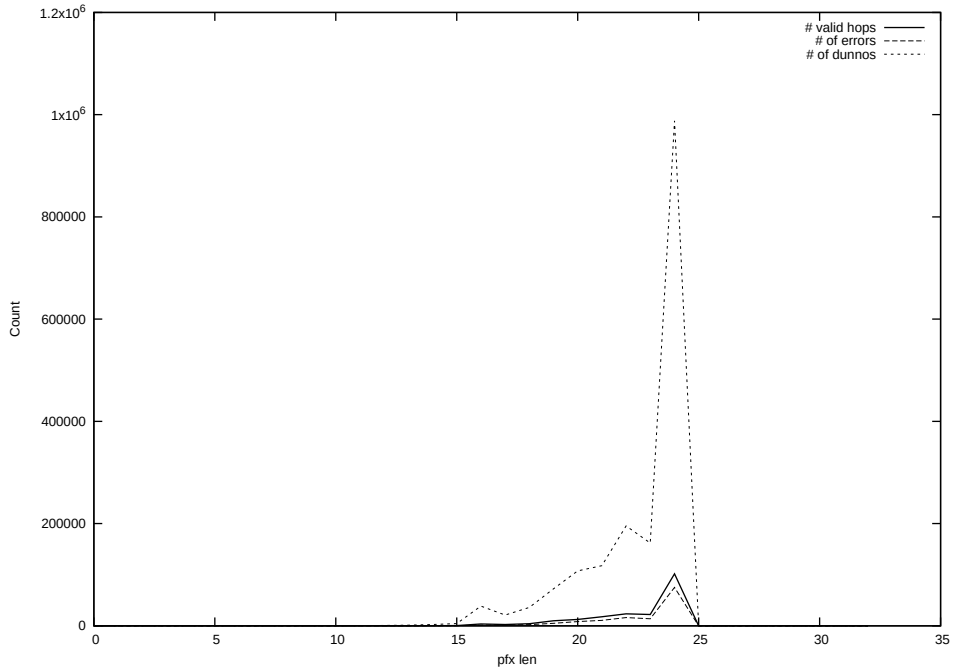
2015-06-08



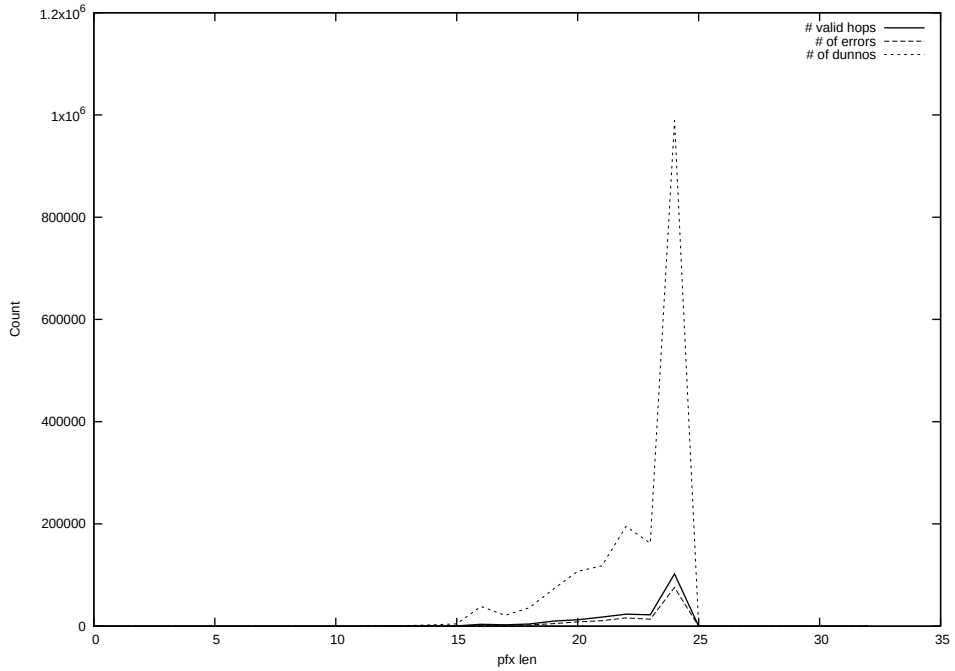
2015-06-09



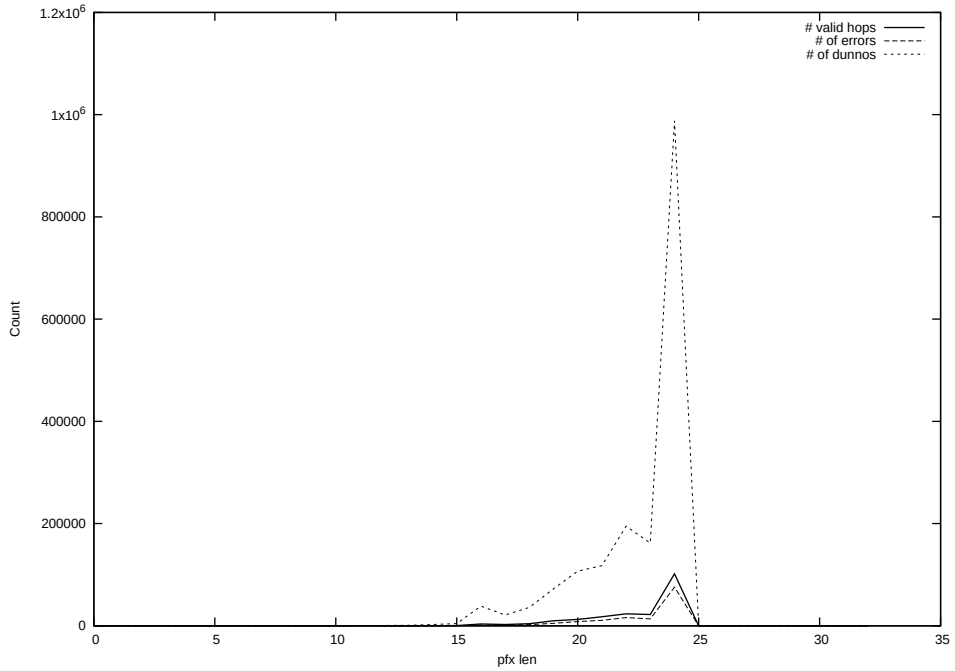
2015-06-10



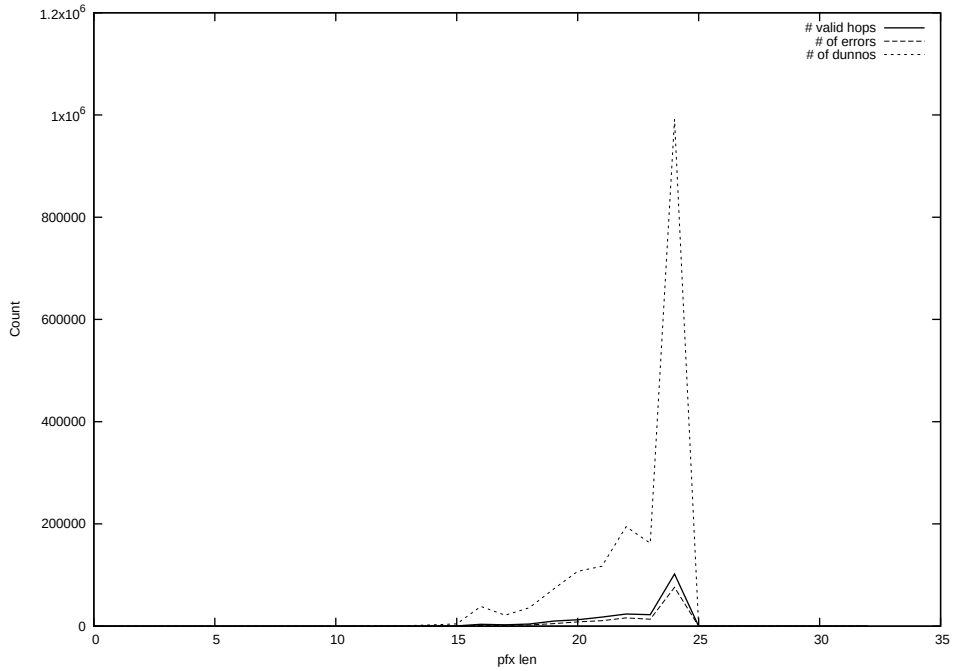
2015-06-11



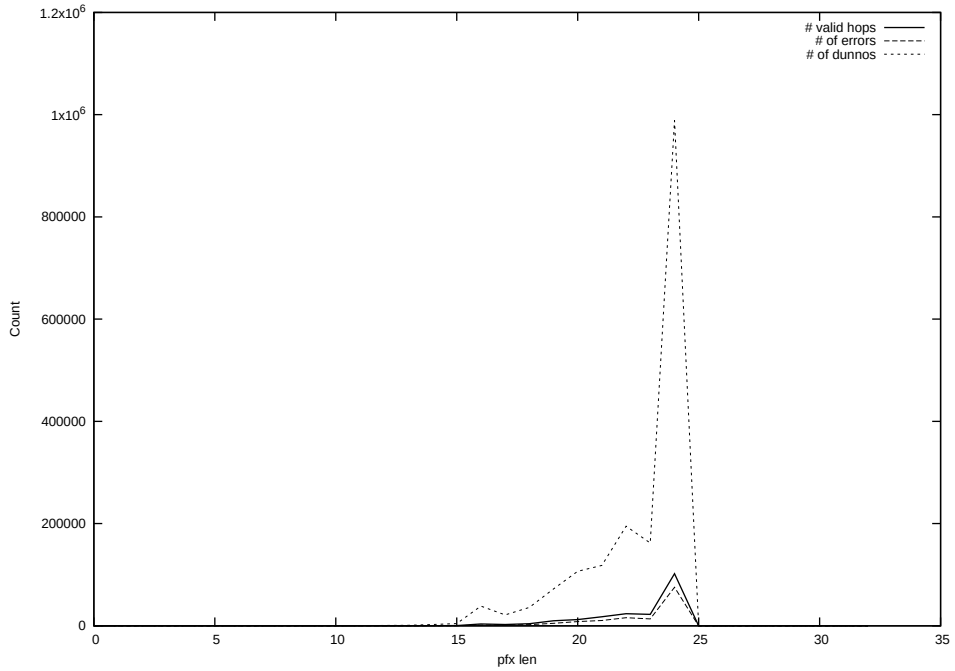
2015-06-12



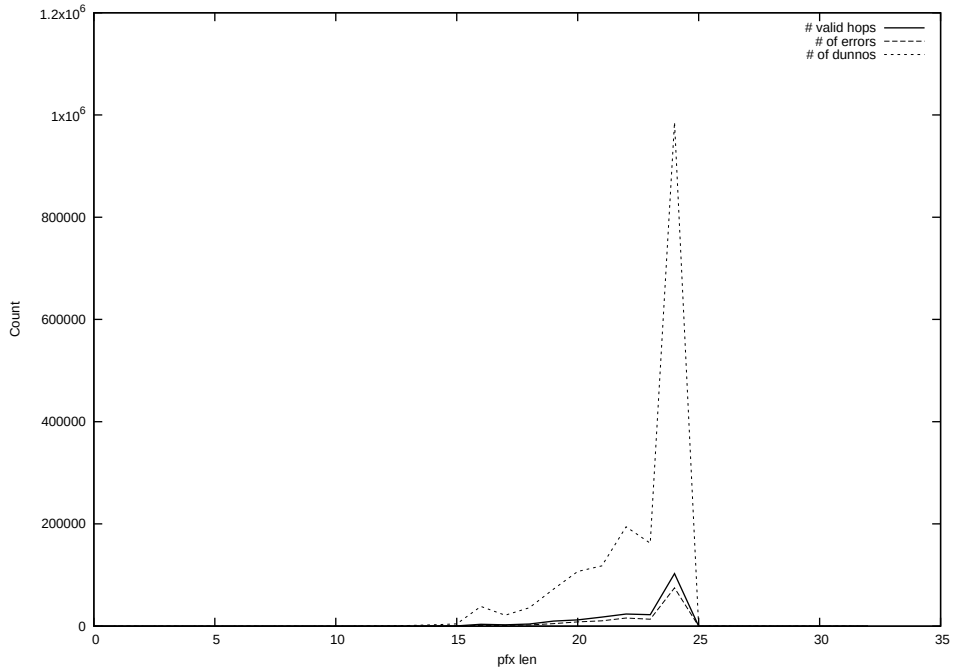
2015-06-13



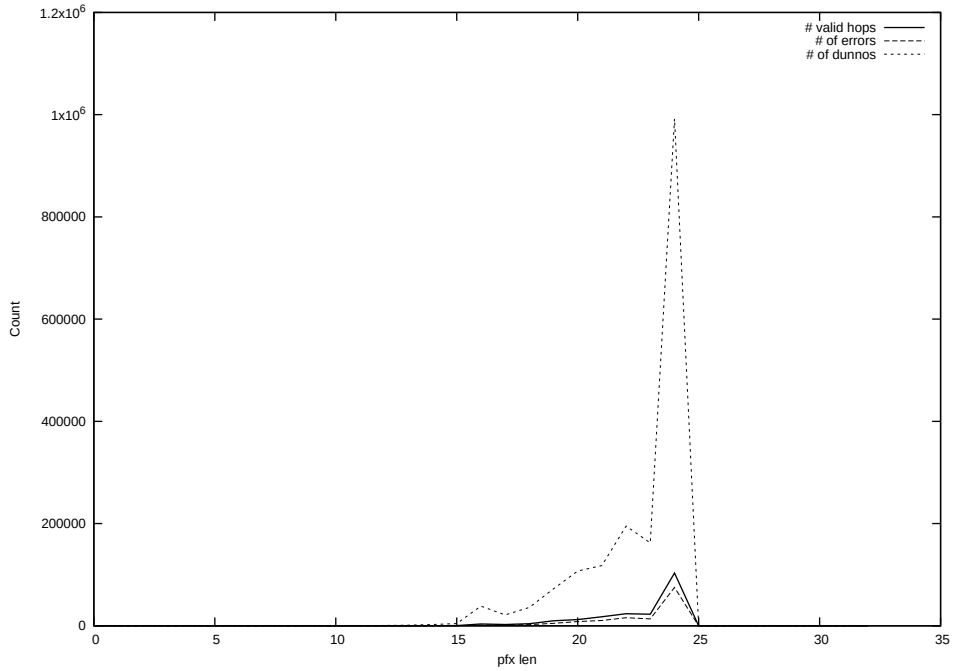
2015-06-14



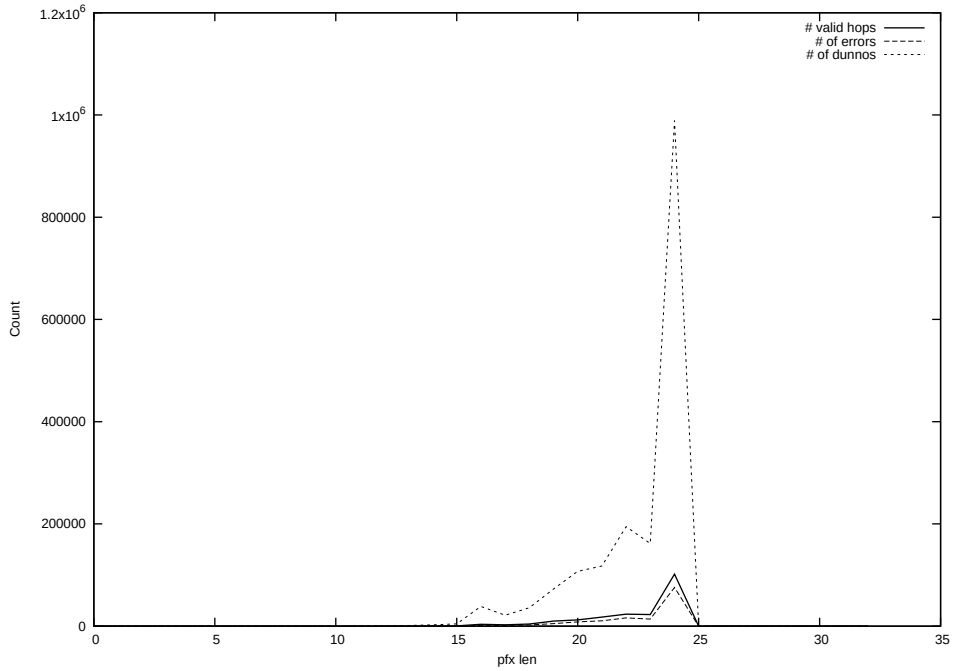
2015-06-15



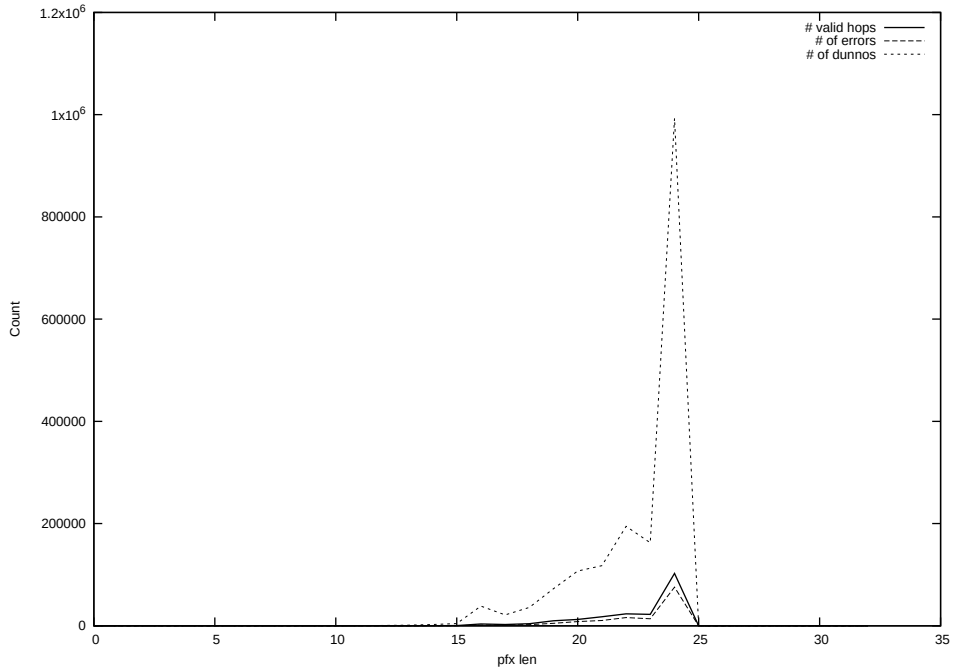
2015-06-16



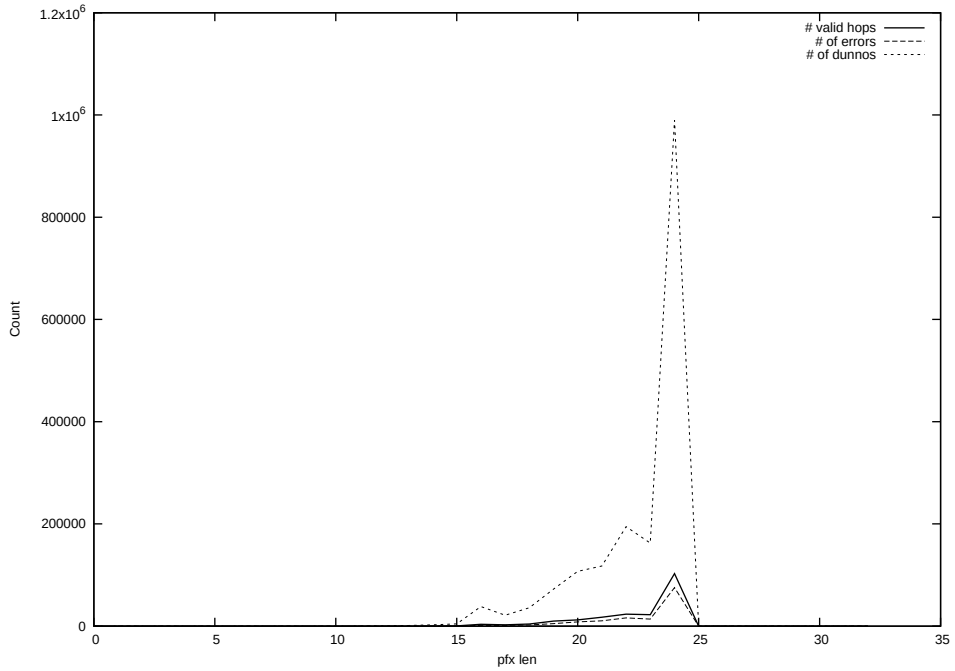
2015-06-17



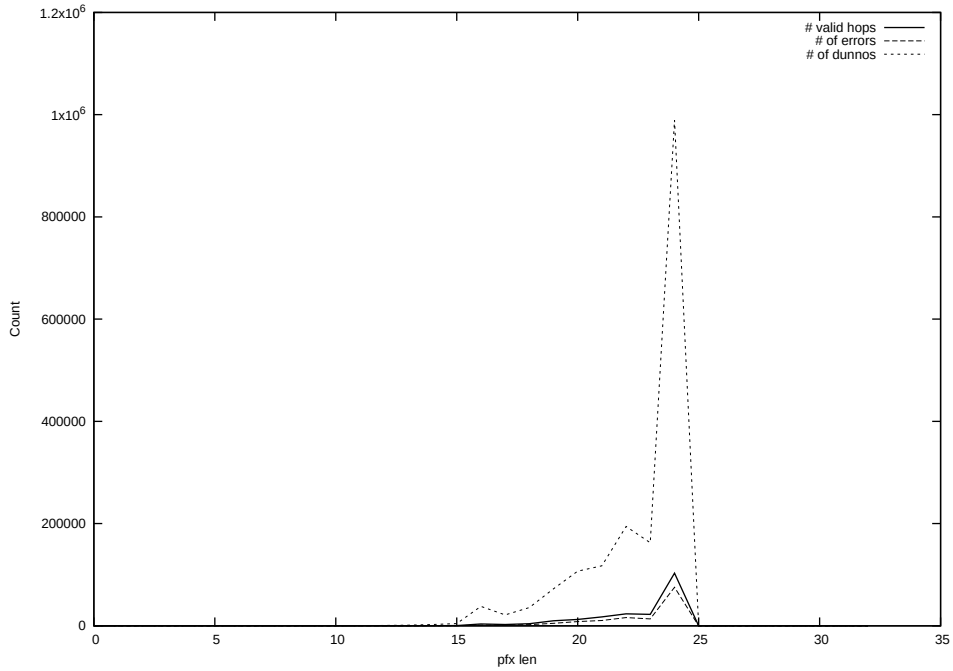
2015-06-18



2015-06-19

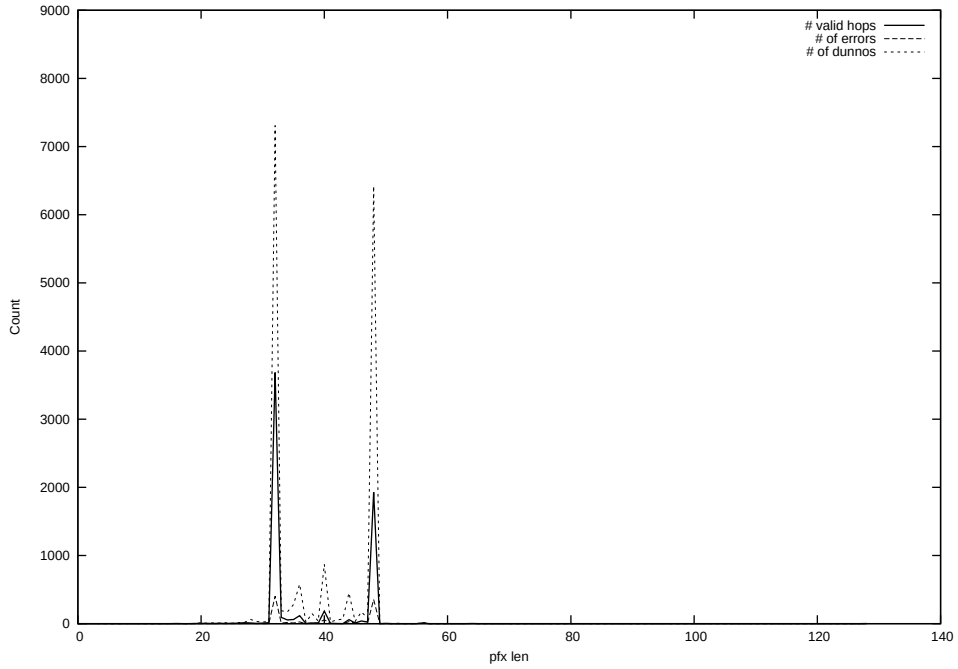


2015-06-20

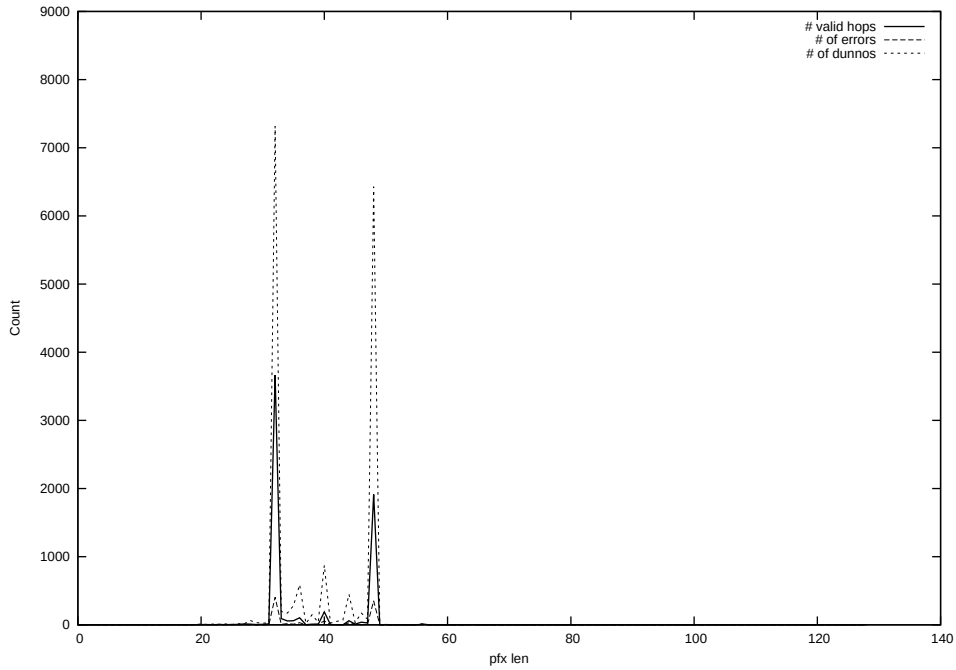


2015-06-21

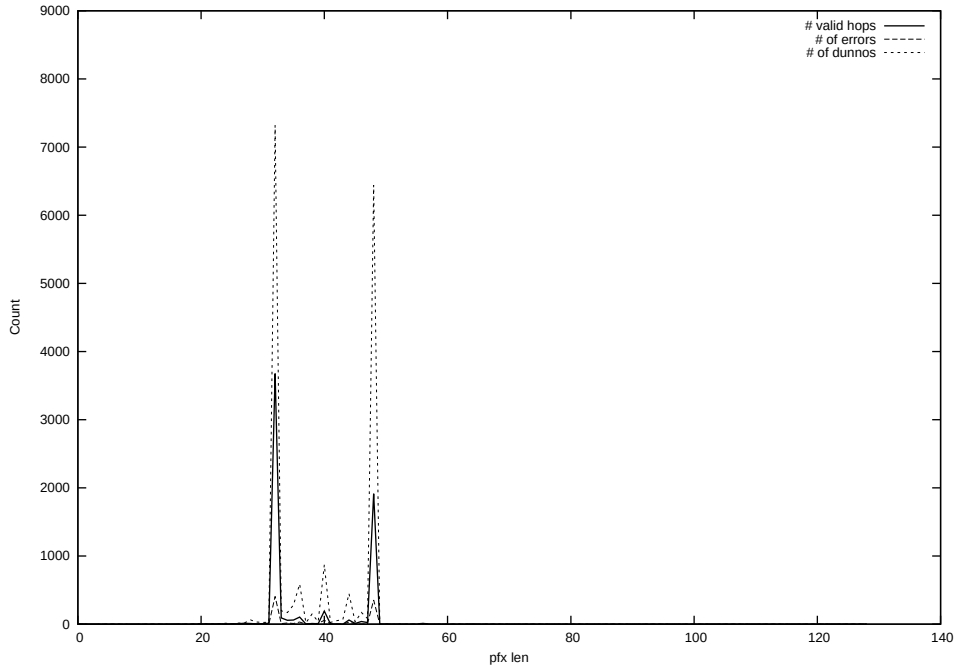
A.14 IPv6 BGP paths against RIPE DB by prefix length



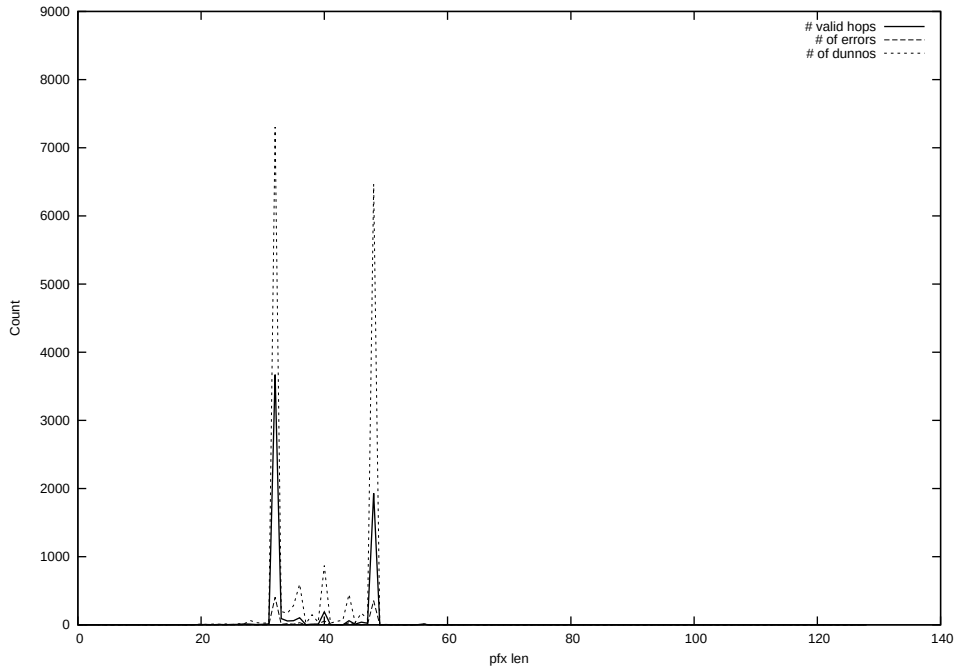
2012-03-22



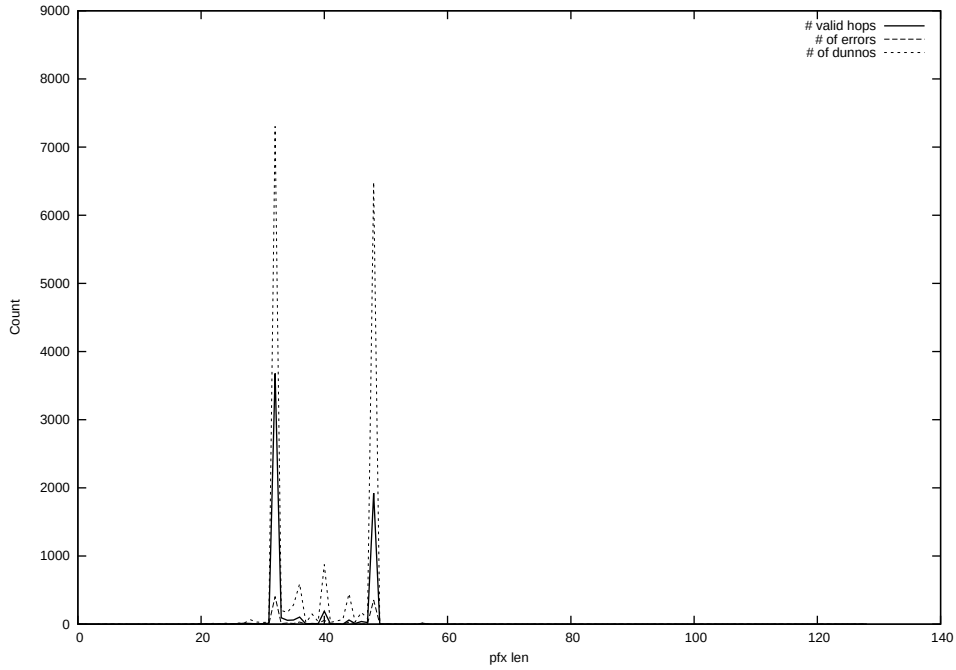
2012-03-23



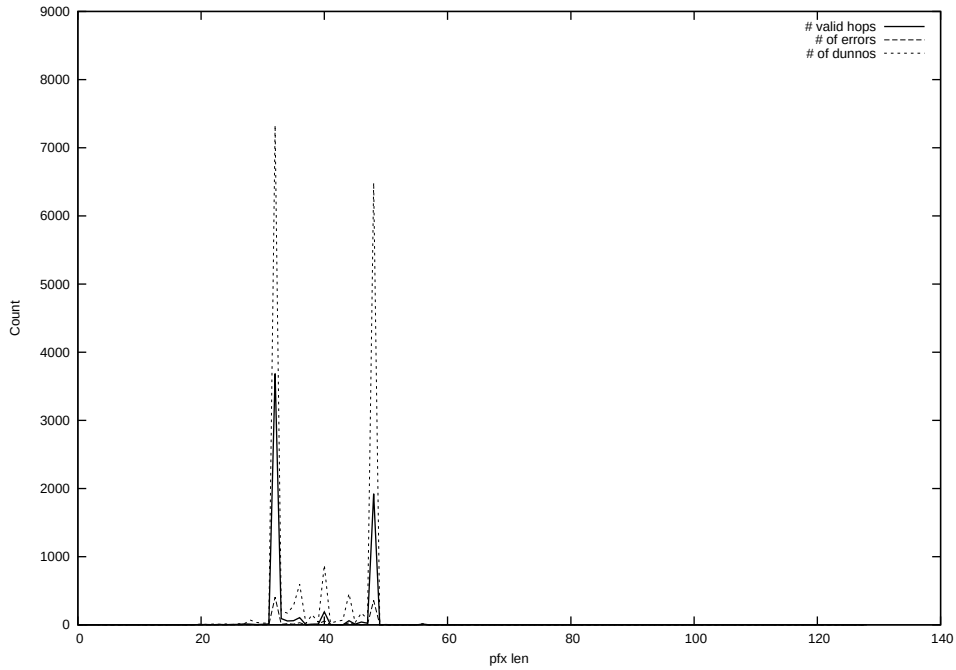
2012-03-24



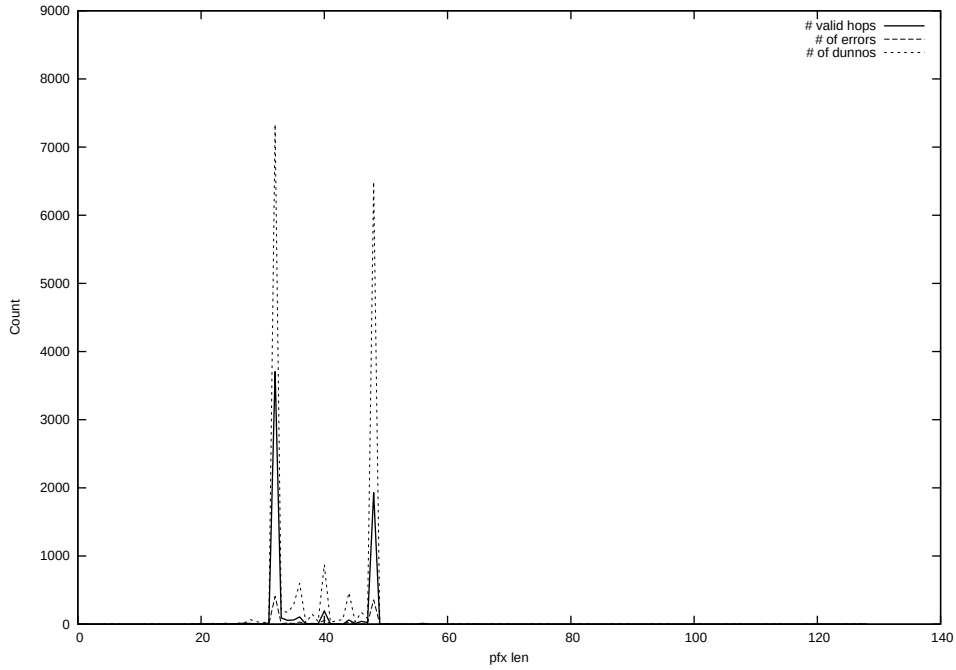
2012-03-25



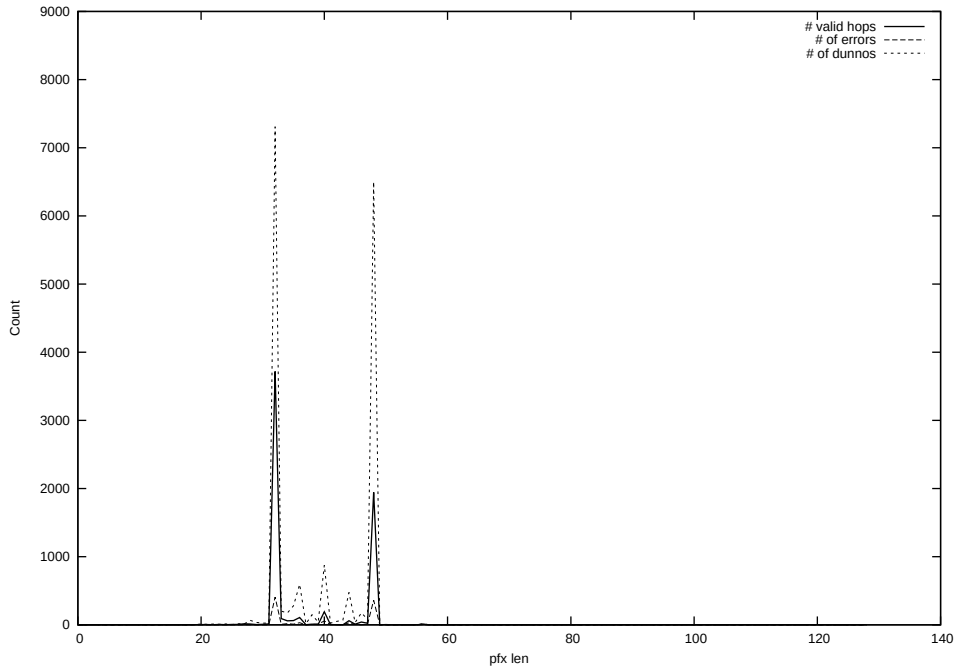
2012-03-26



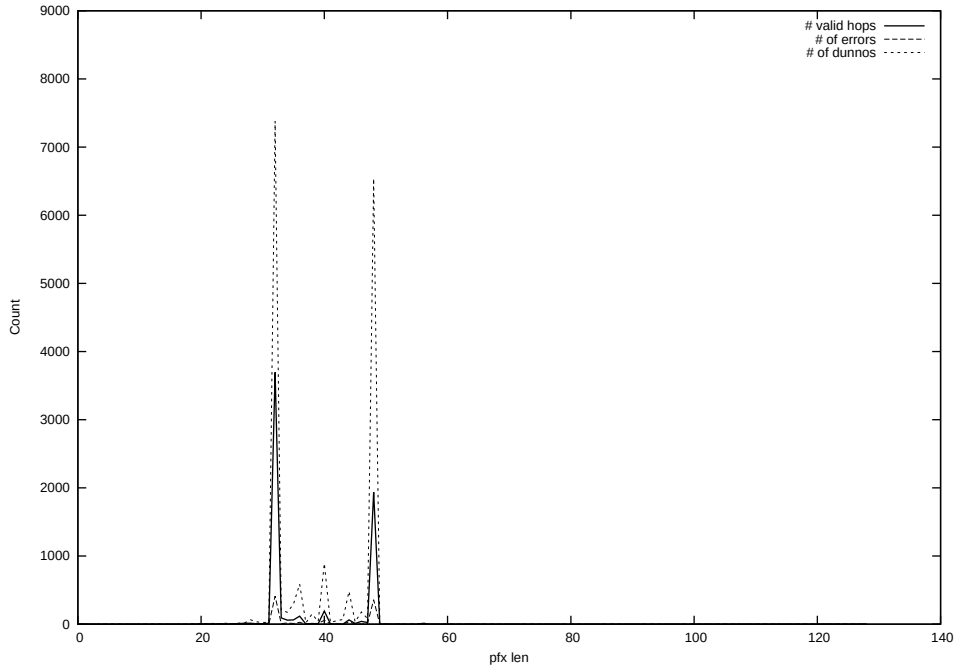
2012-03-27



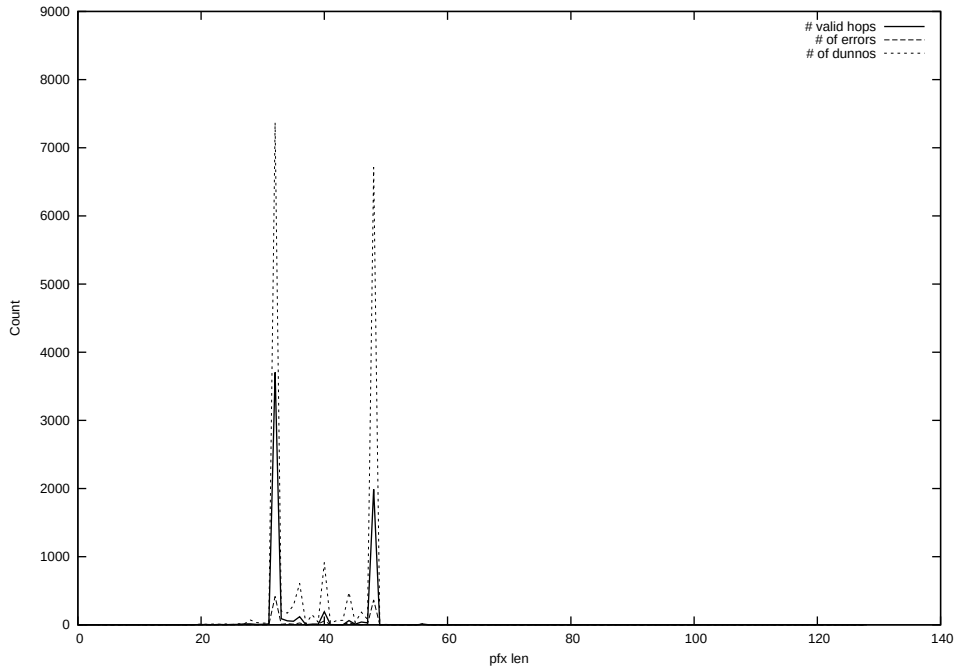
2012-03-28



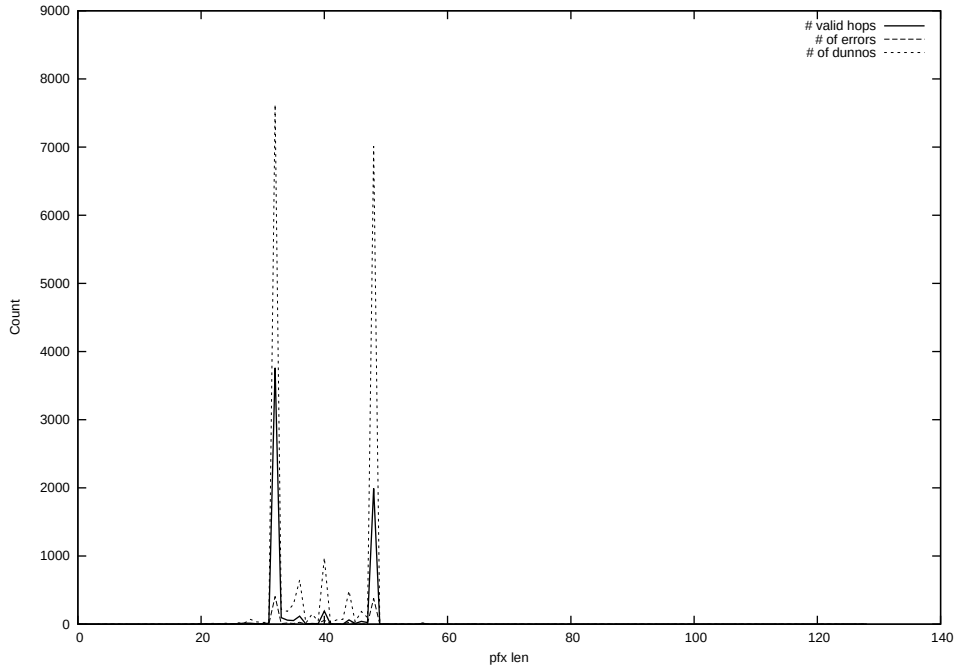
2012-03-29



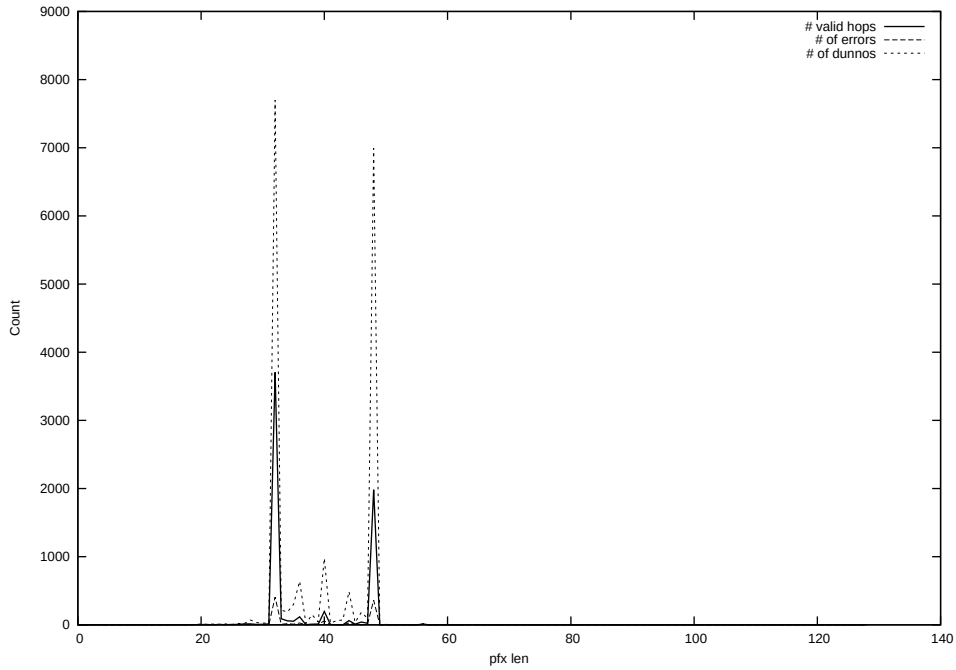
2012-03-30



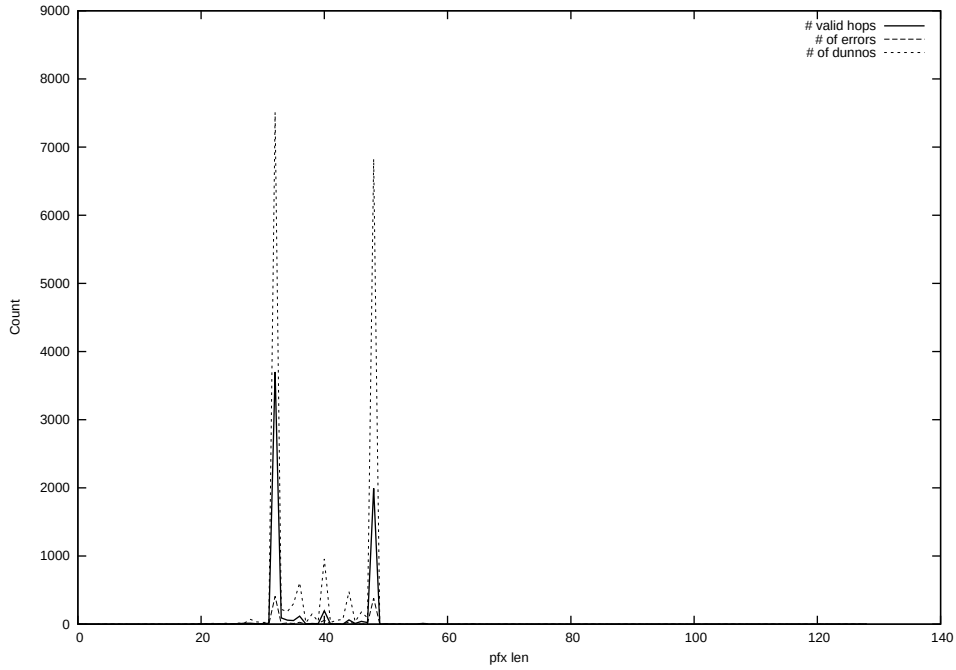
2012-03-31



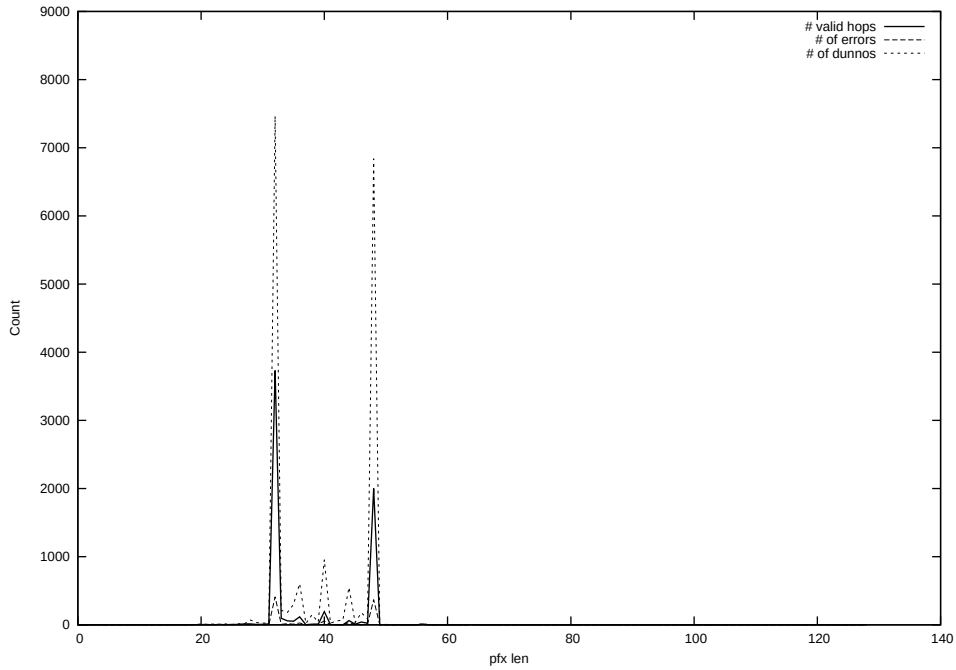
2012-04-01



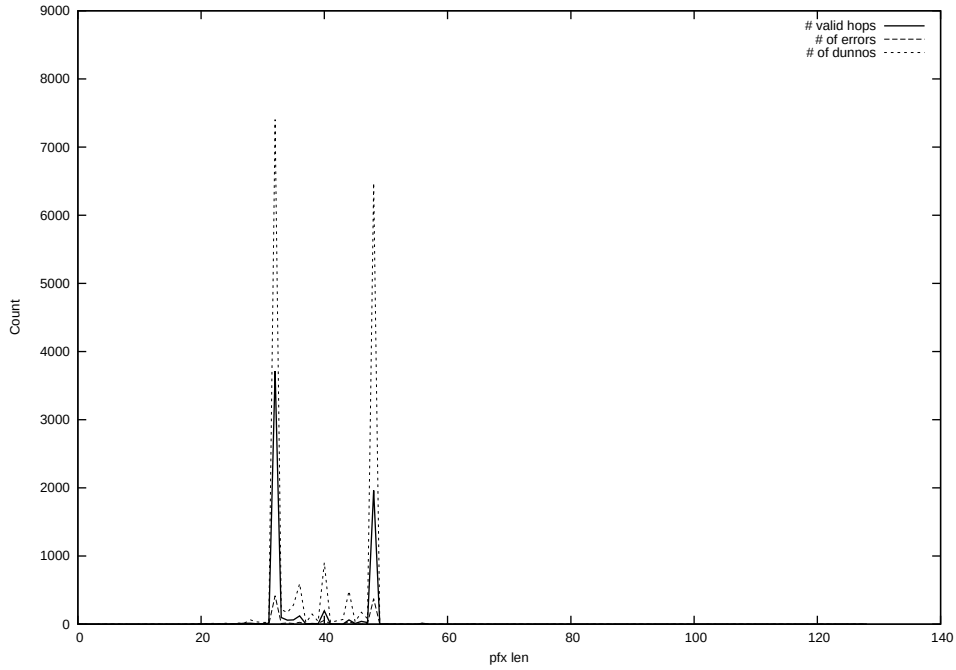
2012-04-02



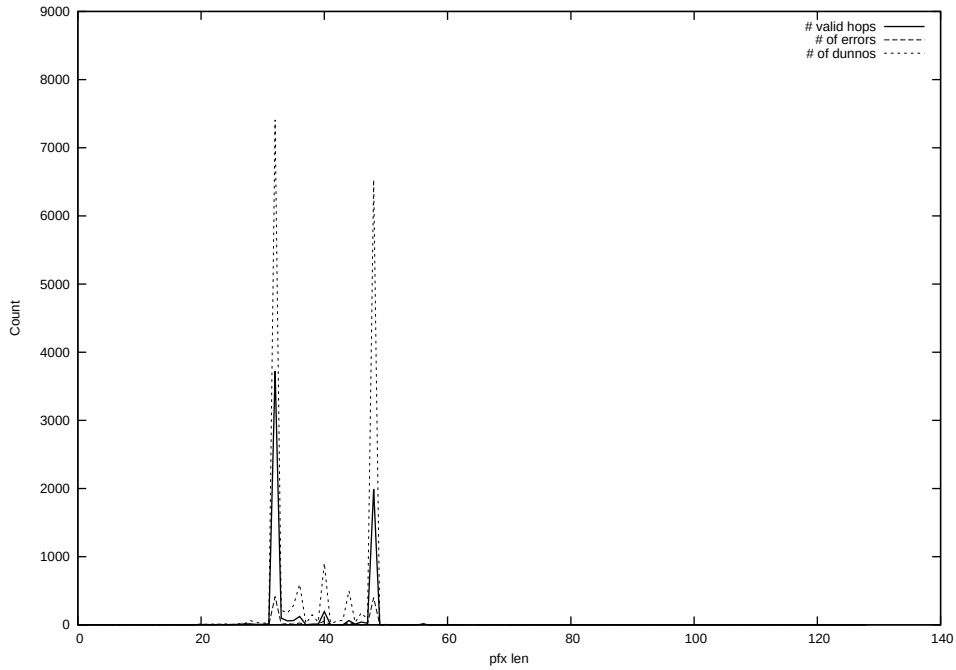
2012-04-03



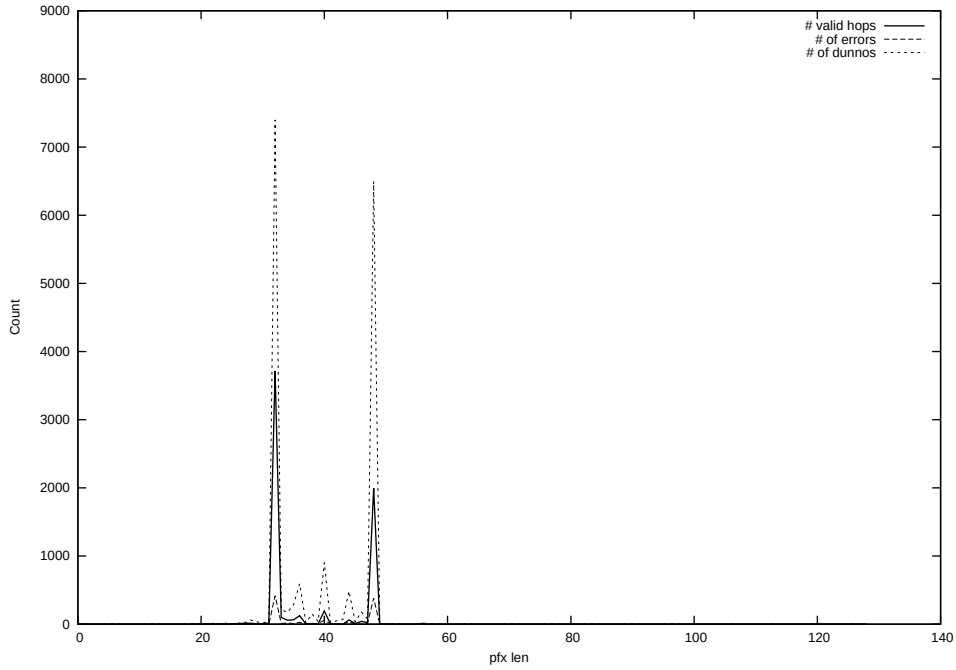
2012-04-04



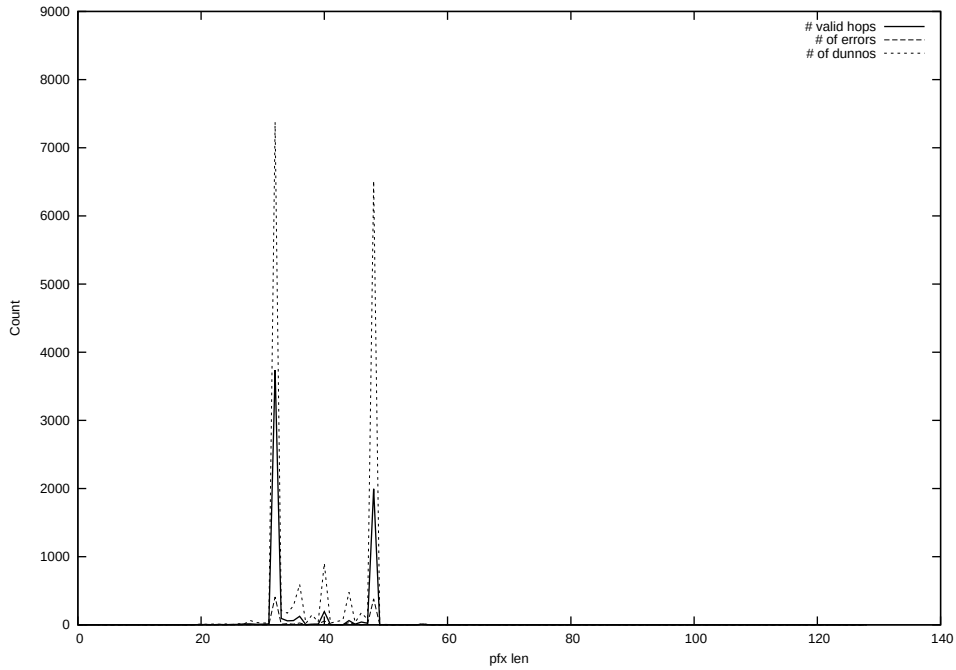
2012-04-05



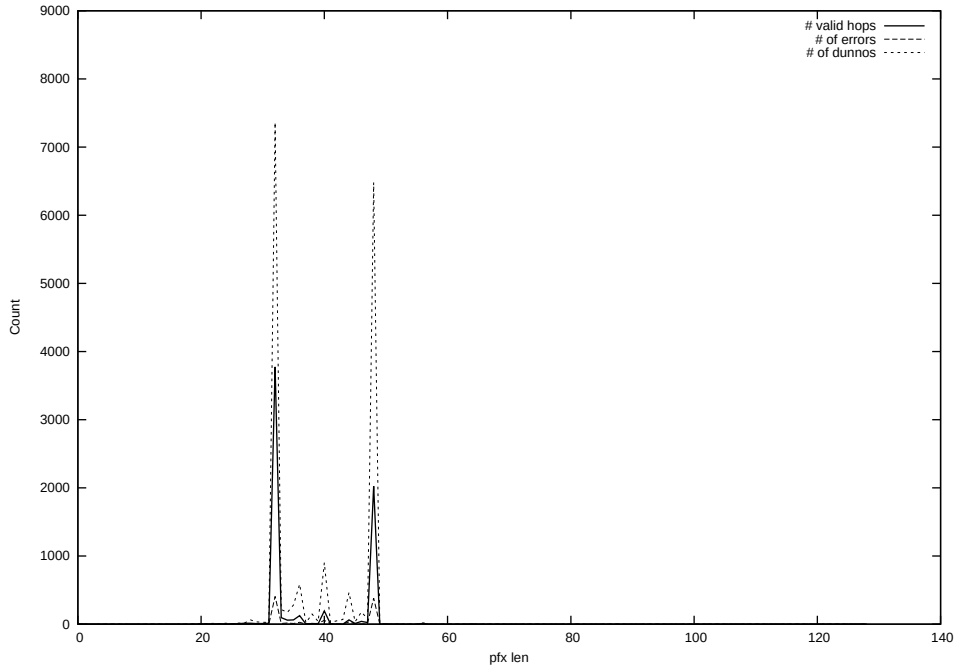
2012-04-06



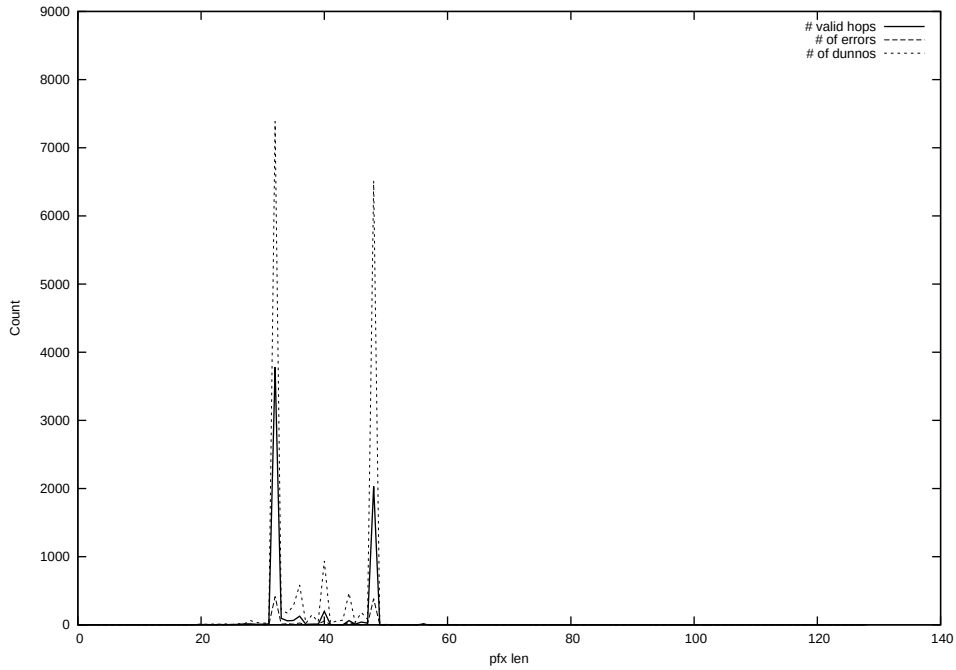
2012-04-07



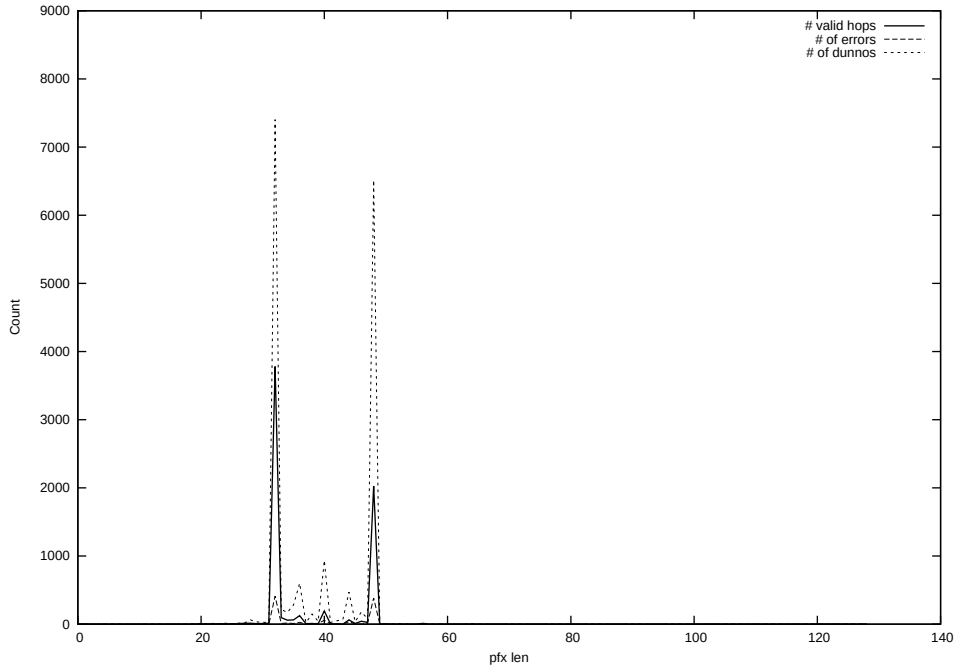
2012-04-08



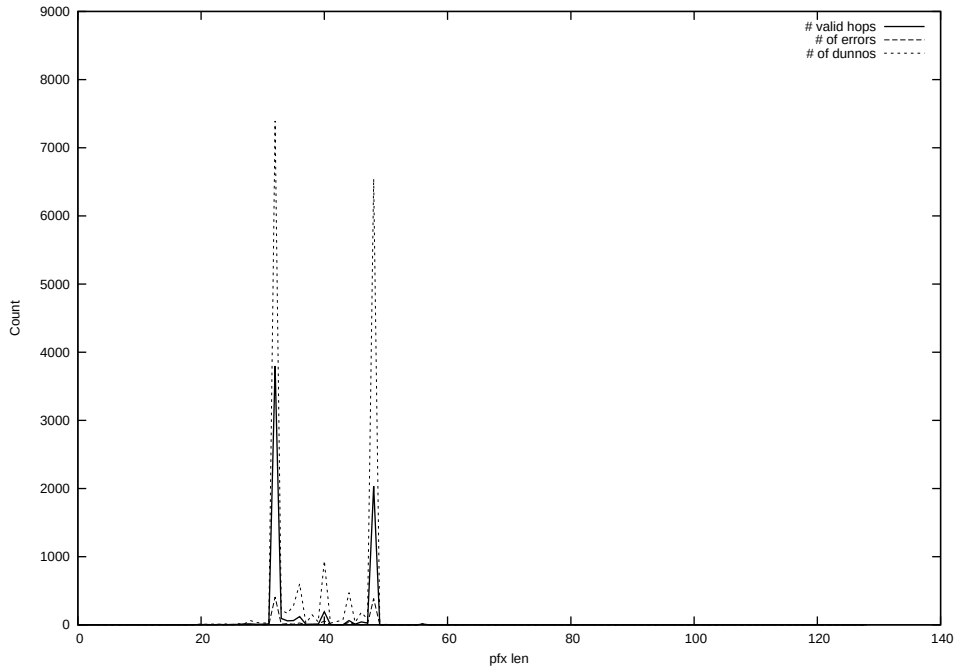
2012-04-09



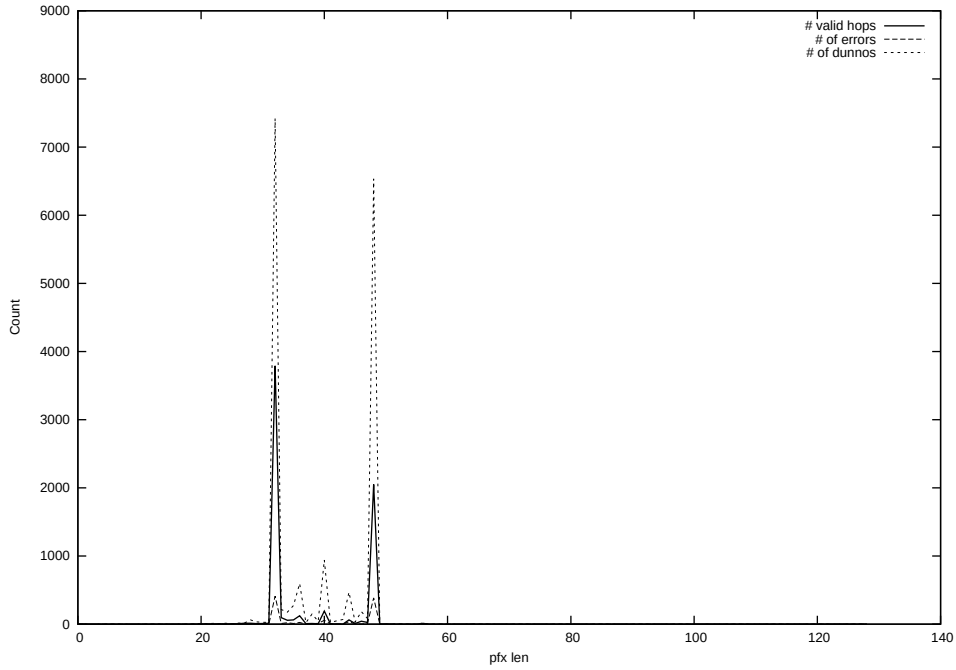
2012-04-10



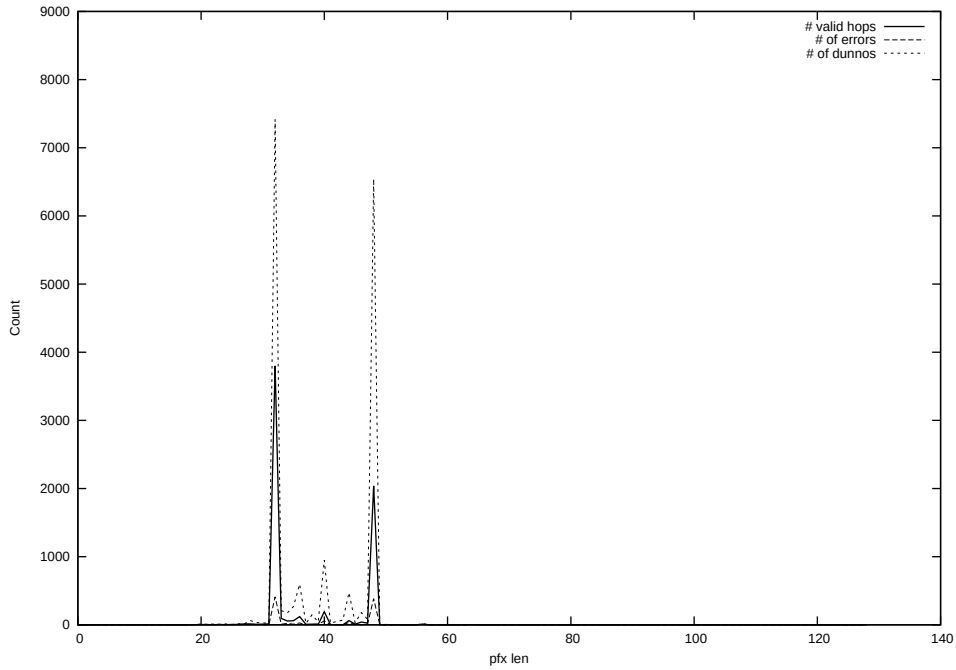
2012-04-11



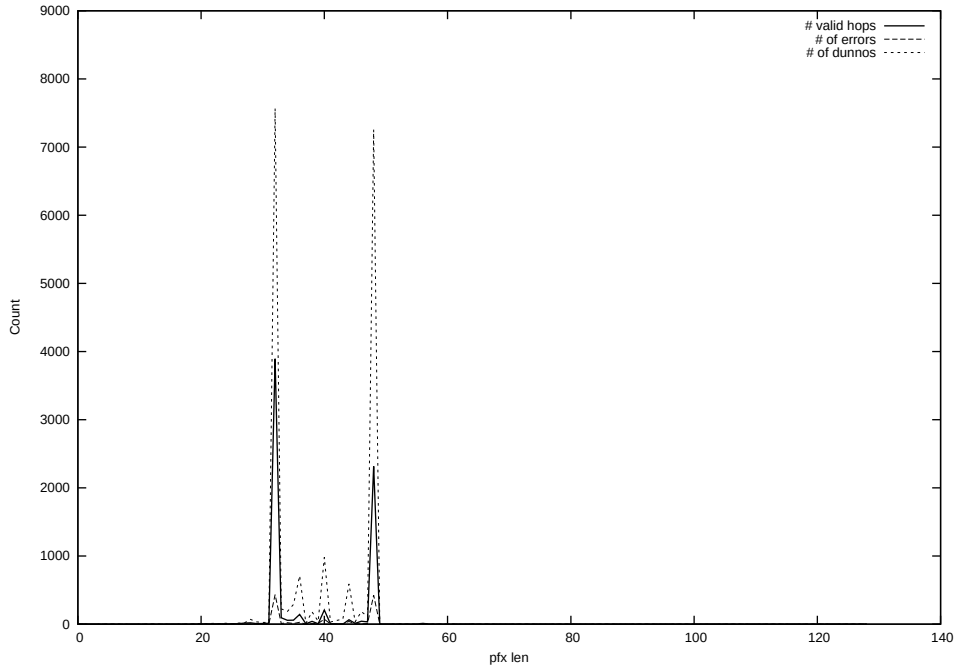
2012-04-12



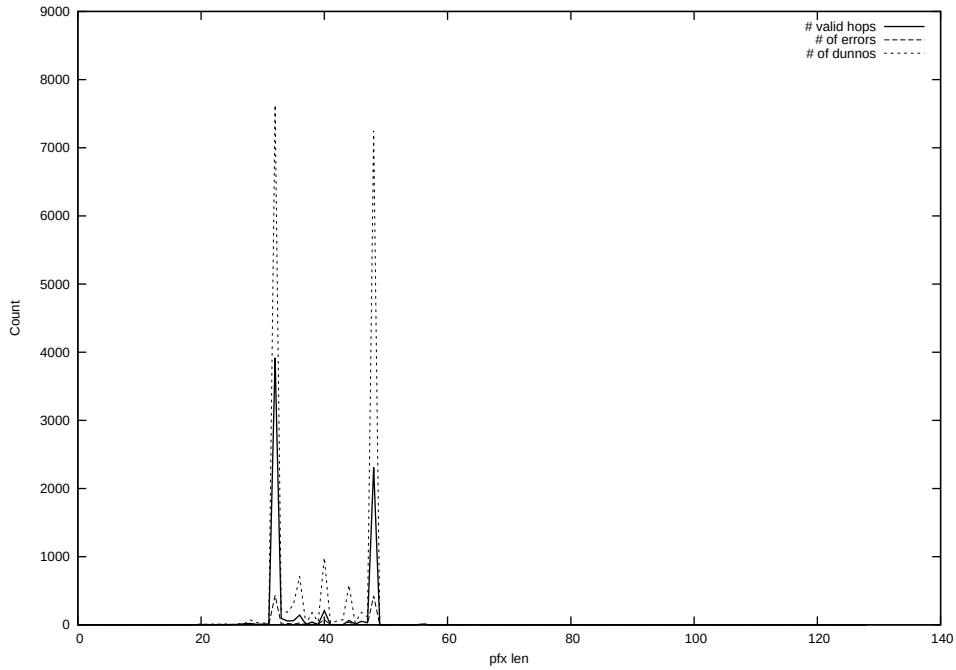
2012-04-13



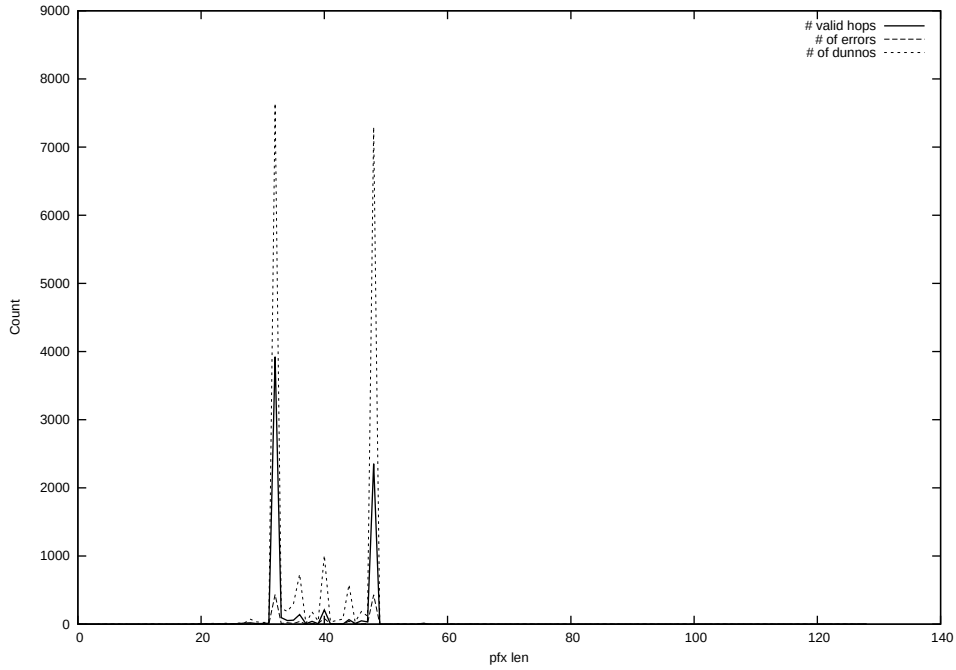
2012-04-14



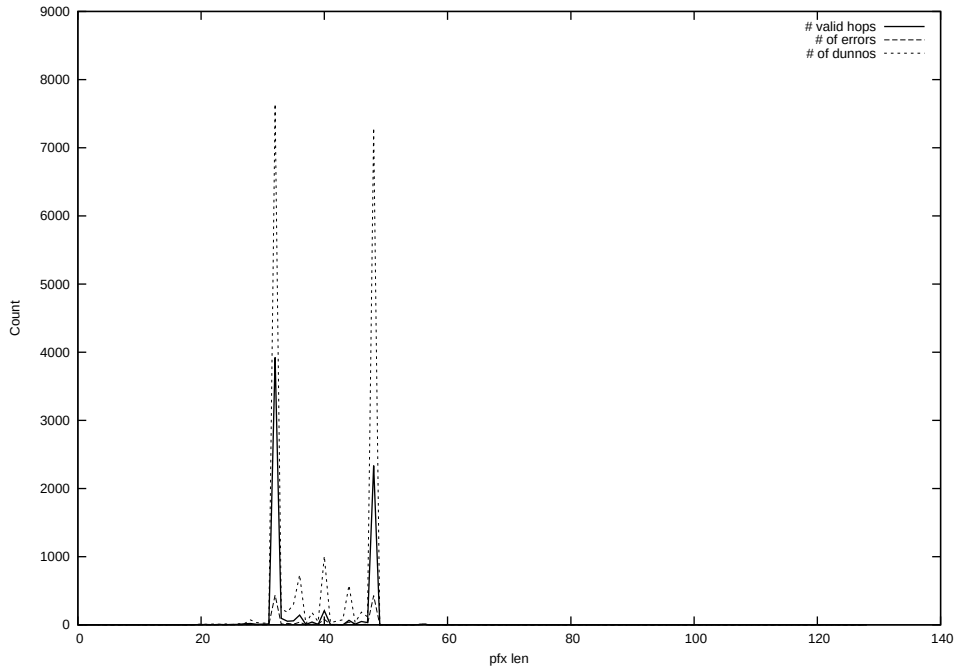
2012-05-16



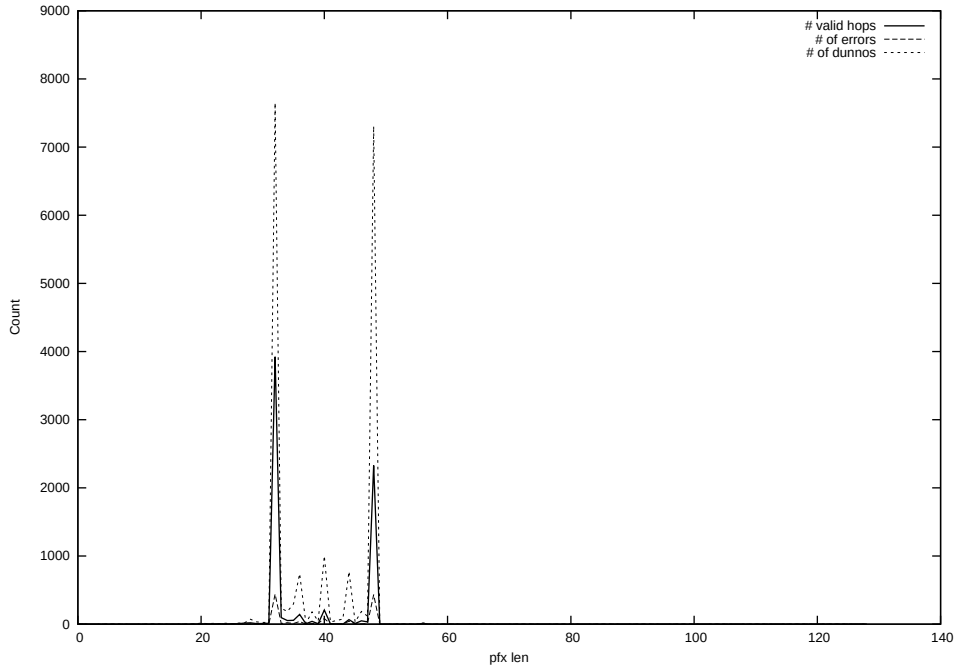
2012-05-17



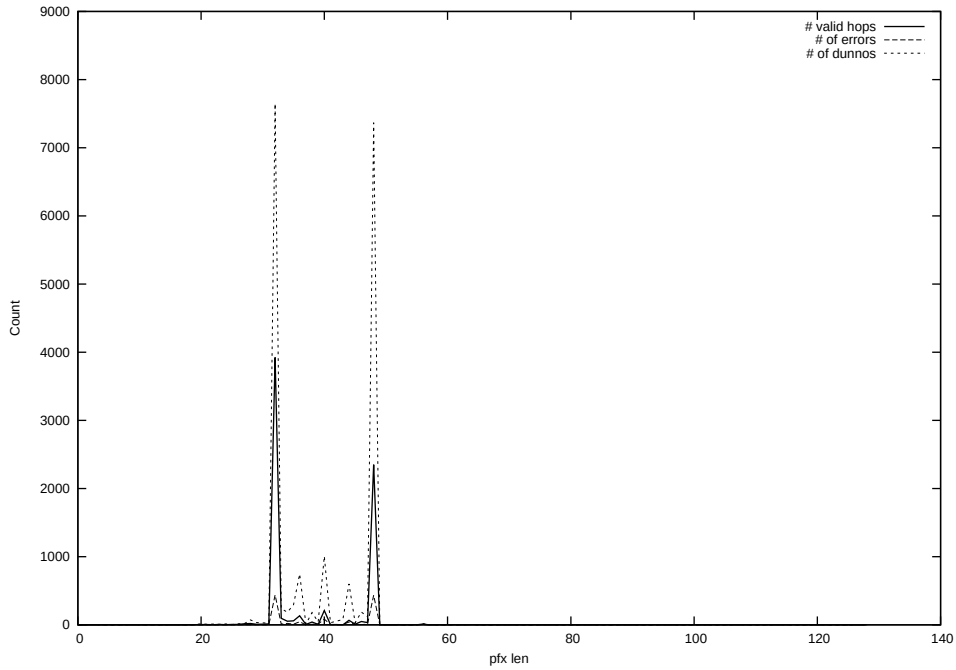
2012-05-19



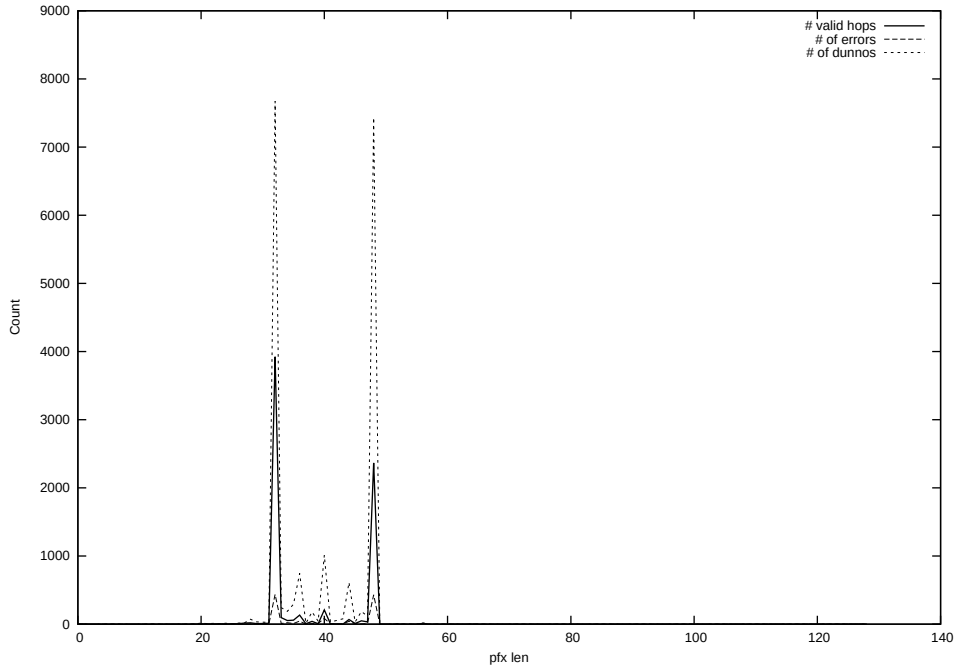
2012-05-20



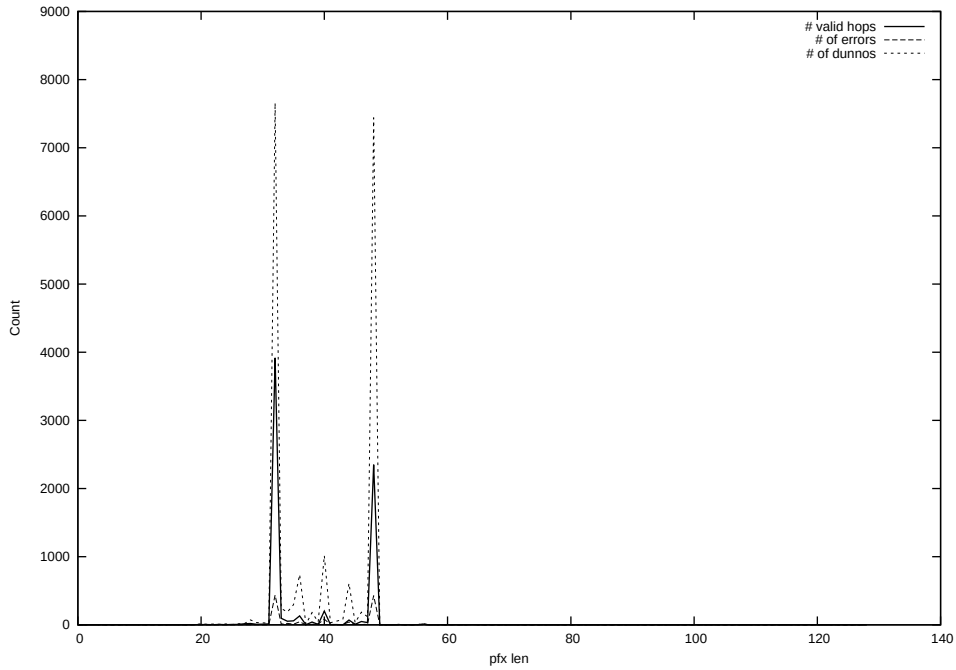
2012-05-21



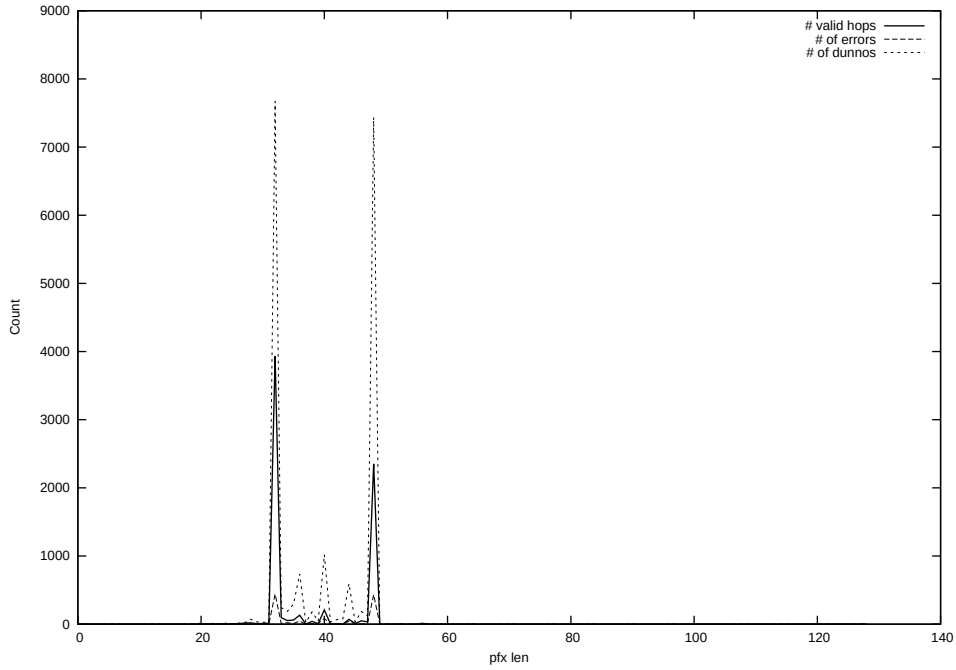
2012-05-22



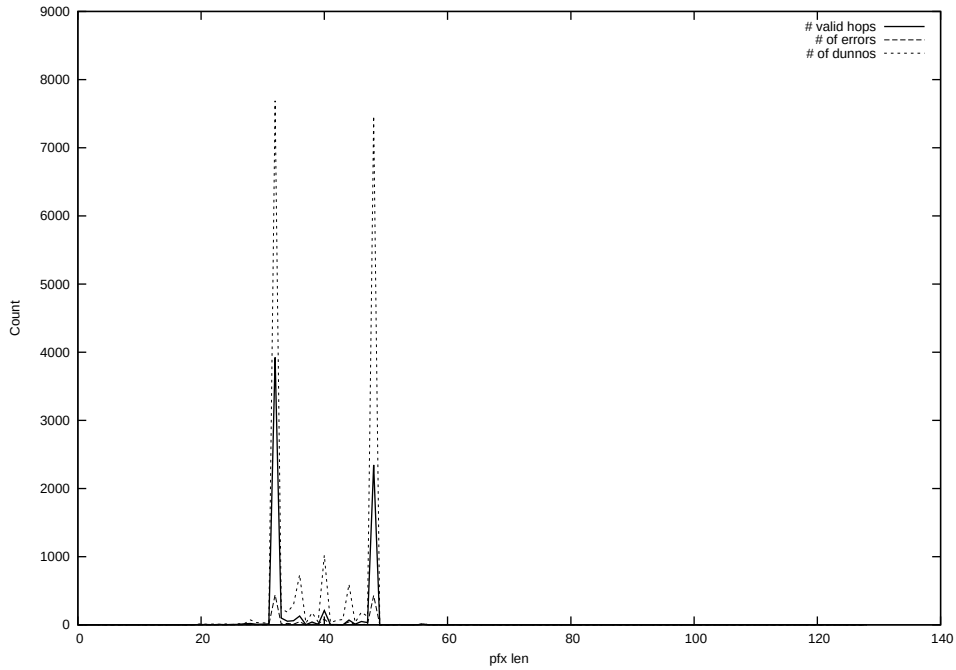
2012-05-23



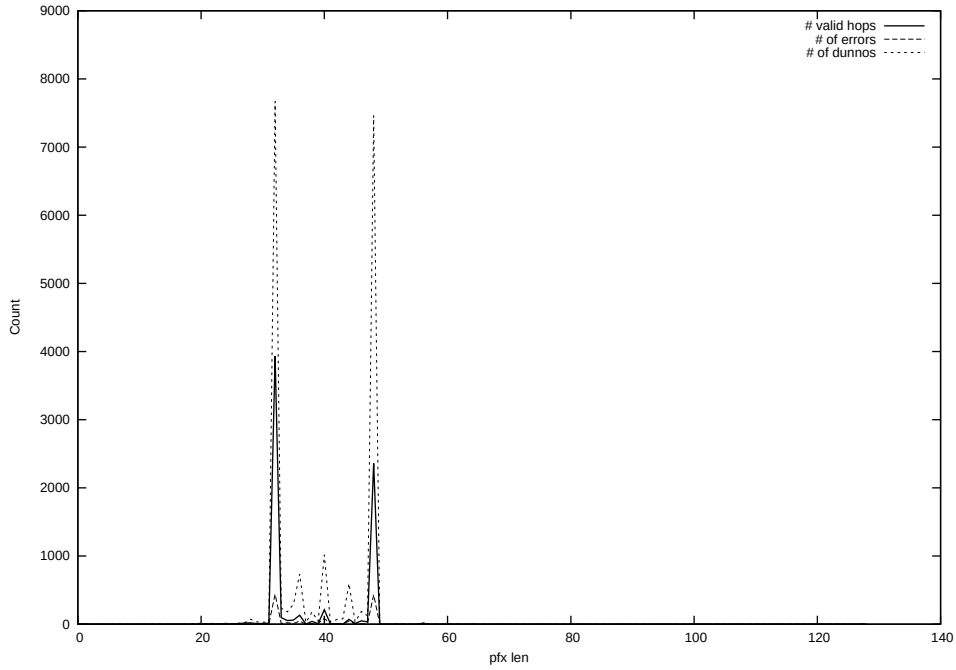
2012-05-24



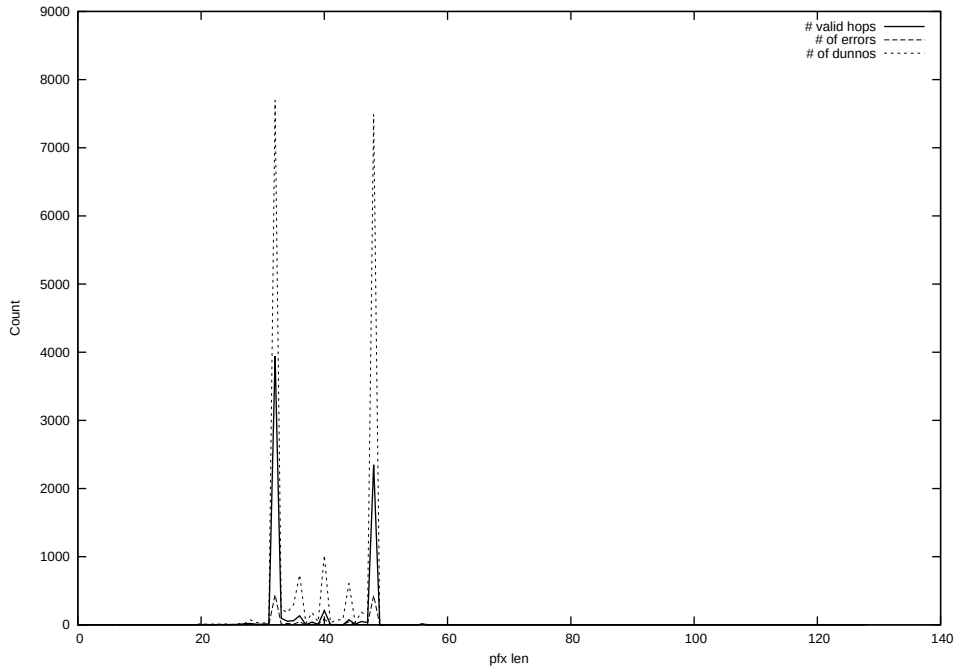
2012-05-25



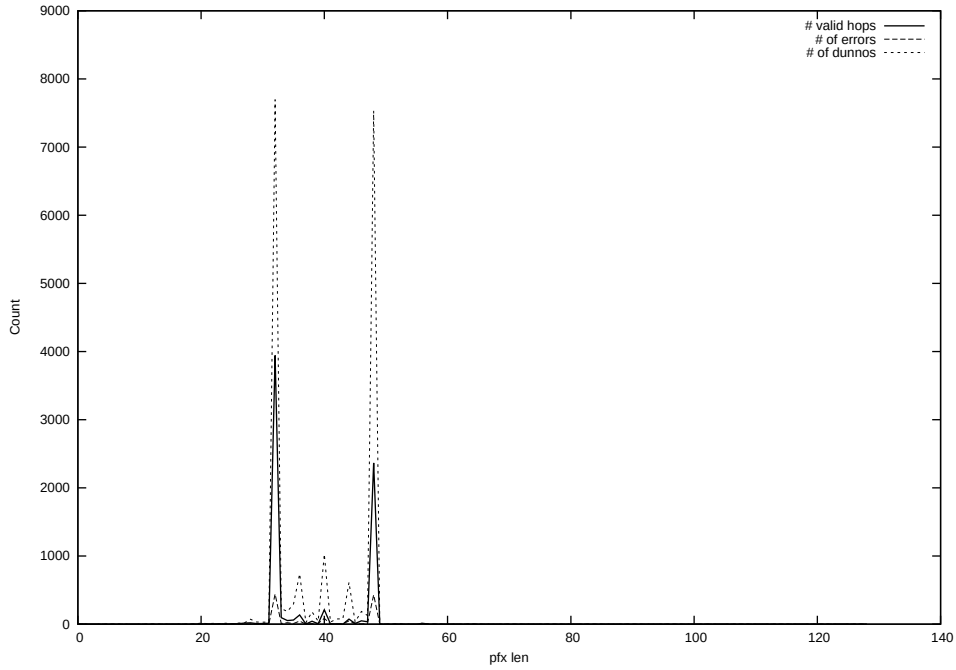
2012-05-26



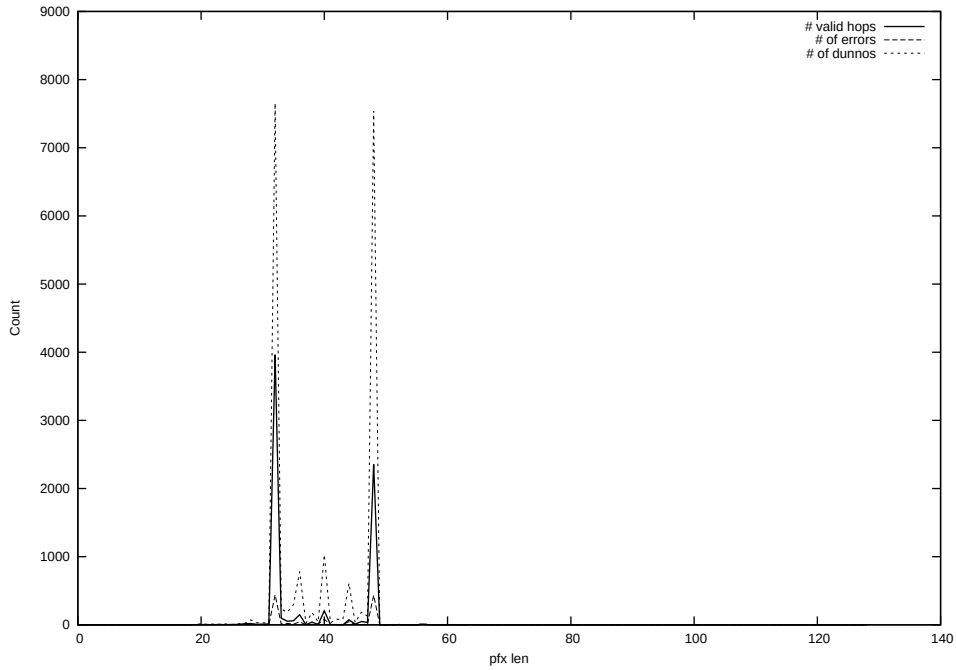
2012-05-27



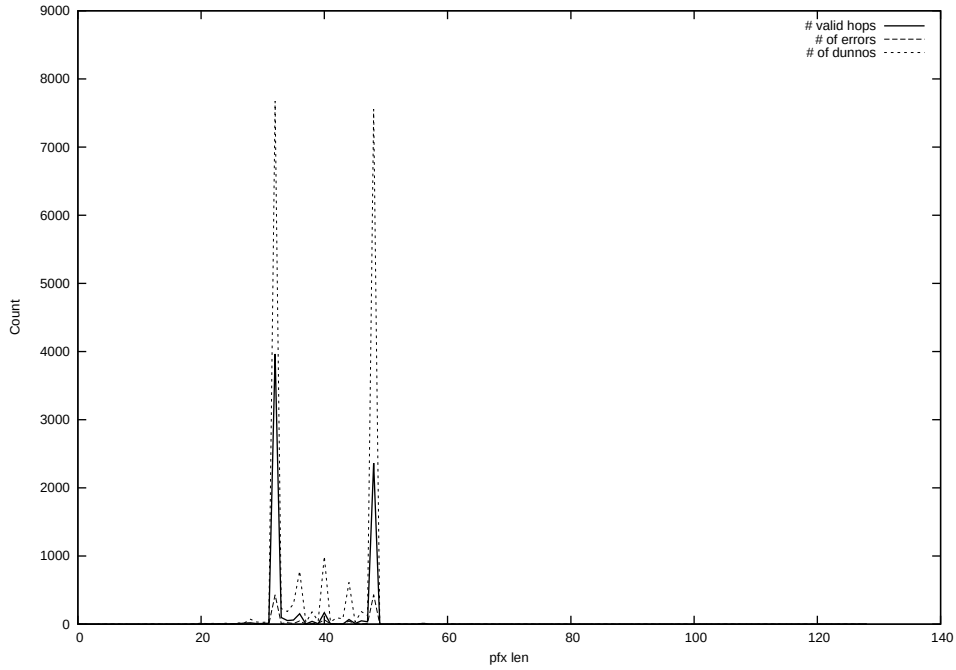
2012-05-28



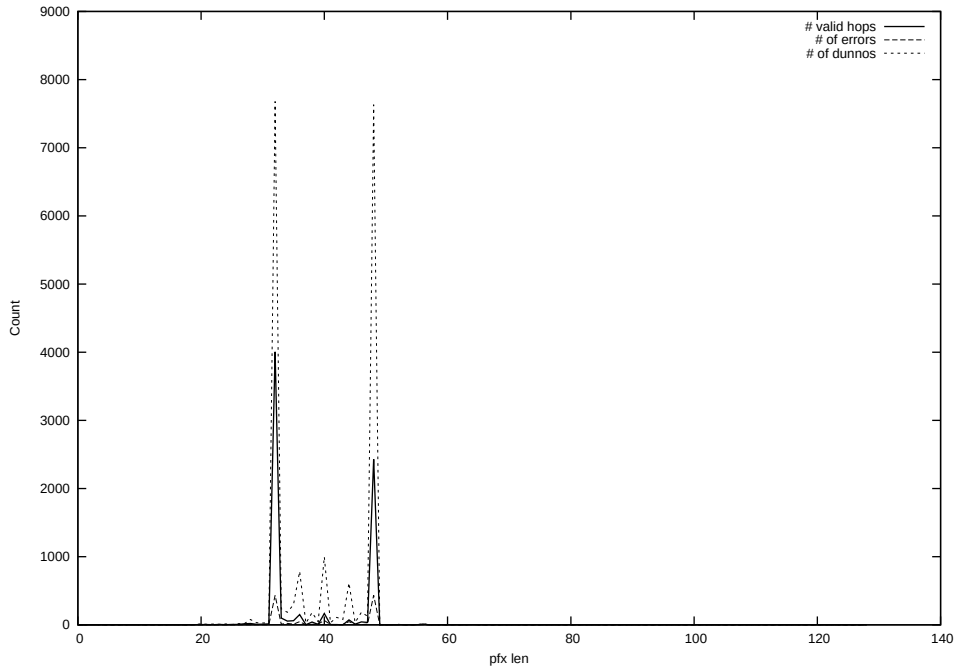
2012-05-29



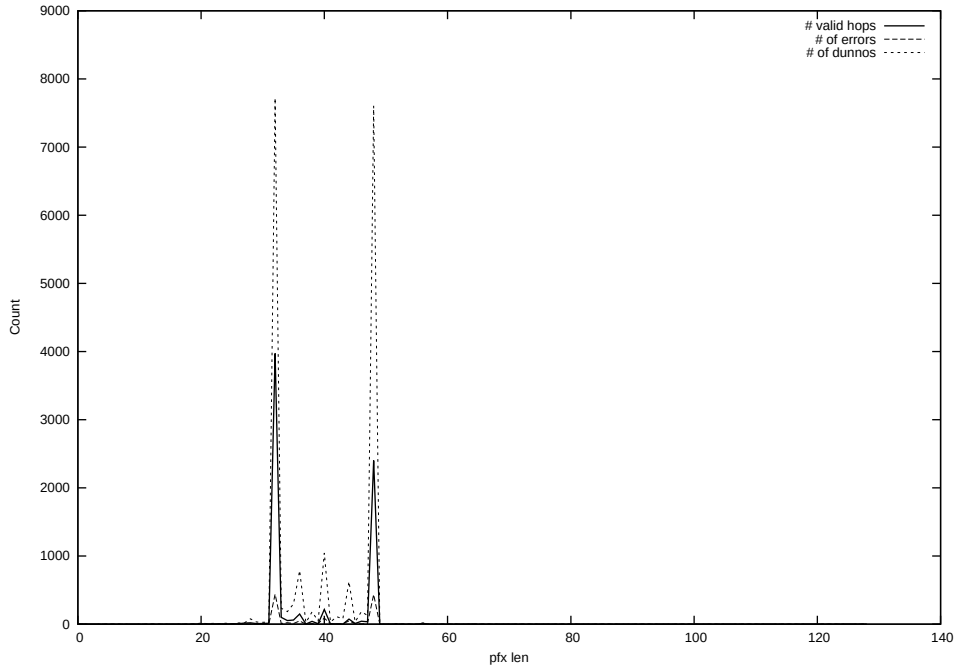
2012-05-30



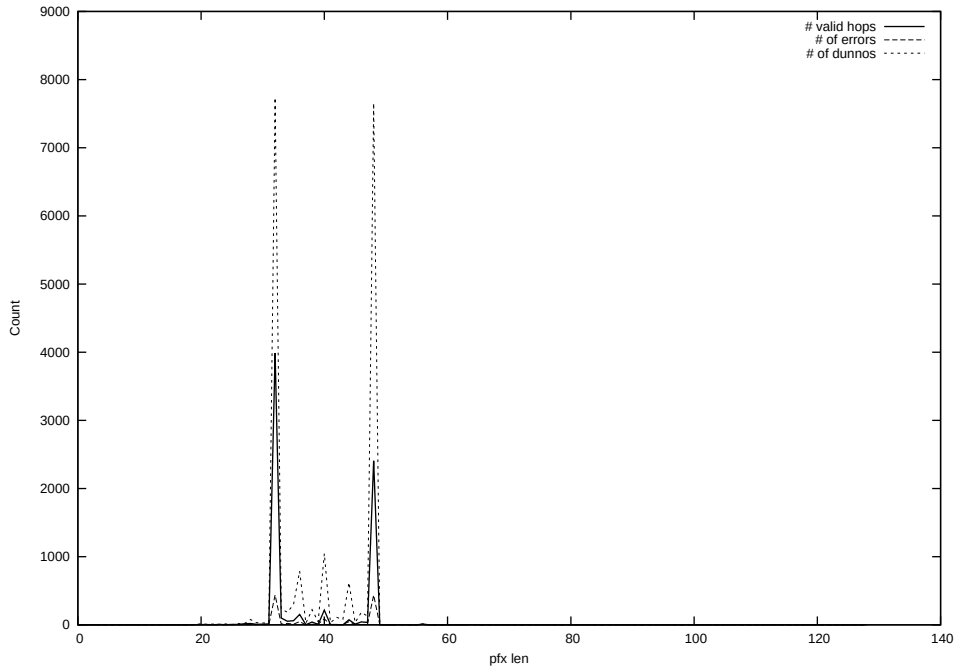
2012-05-31



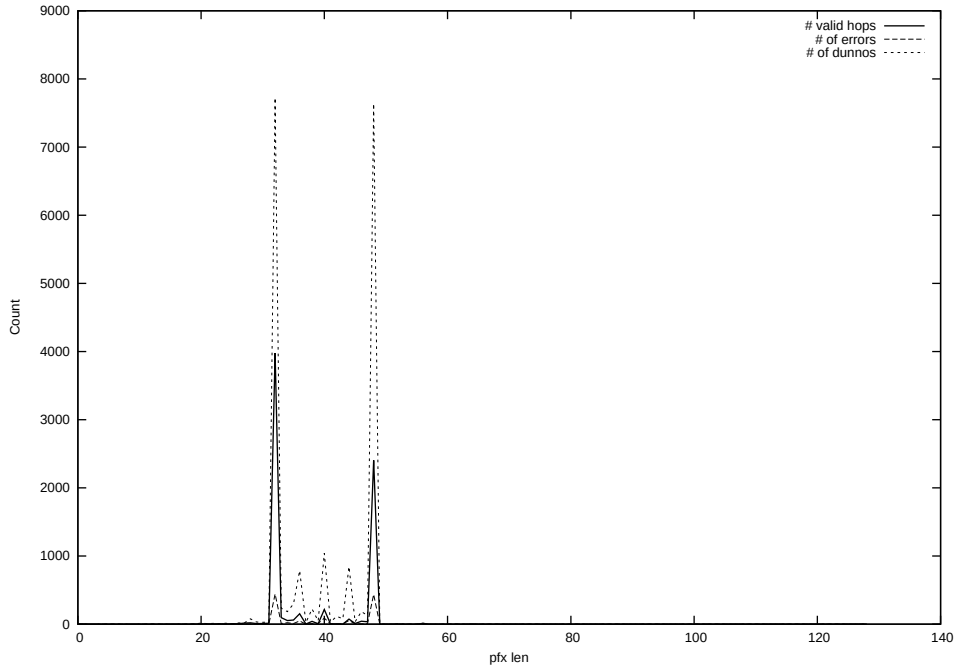
2012-06-01



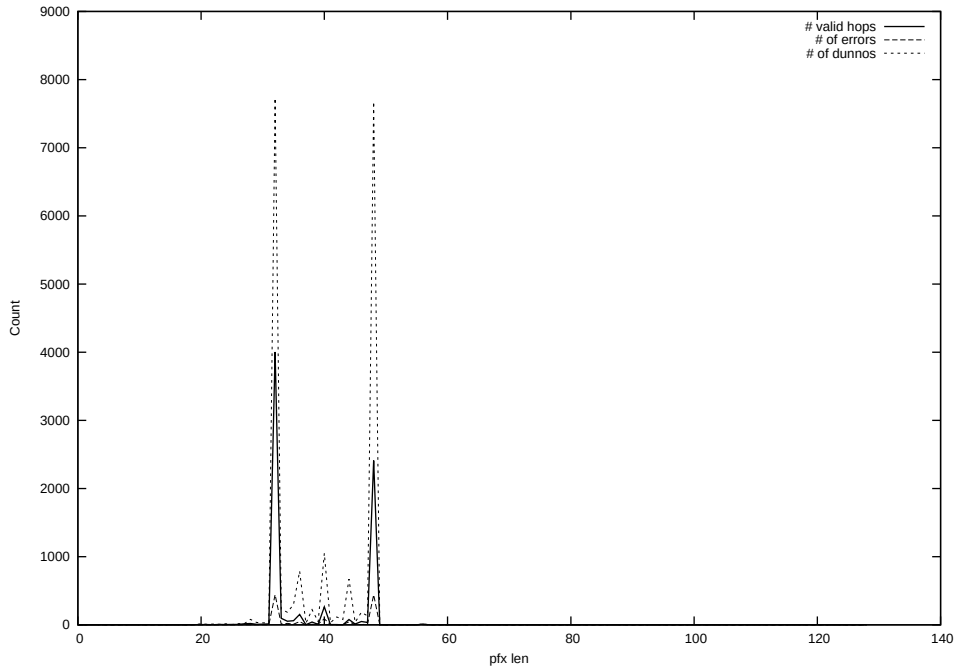
2012-06-02



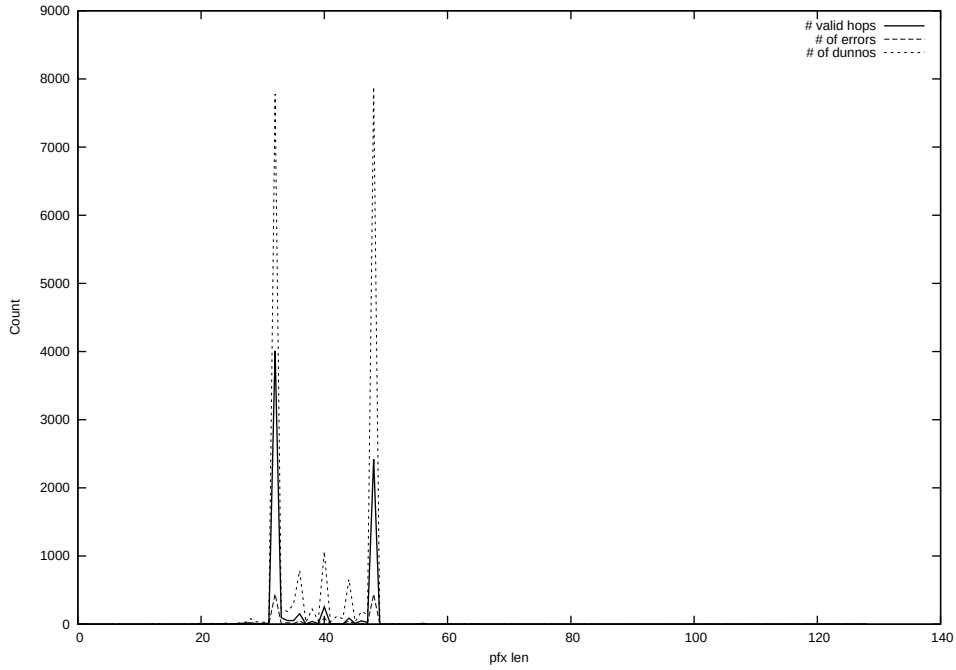
2012-06-03



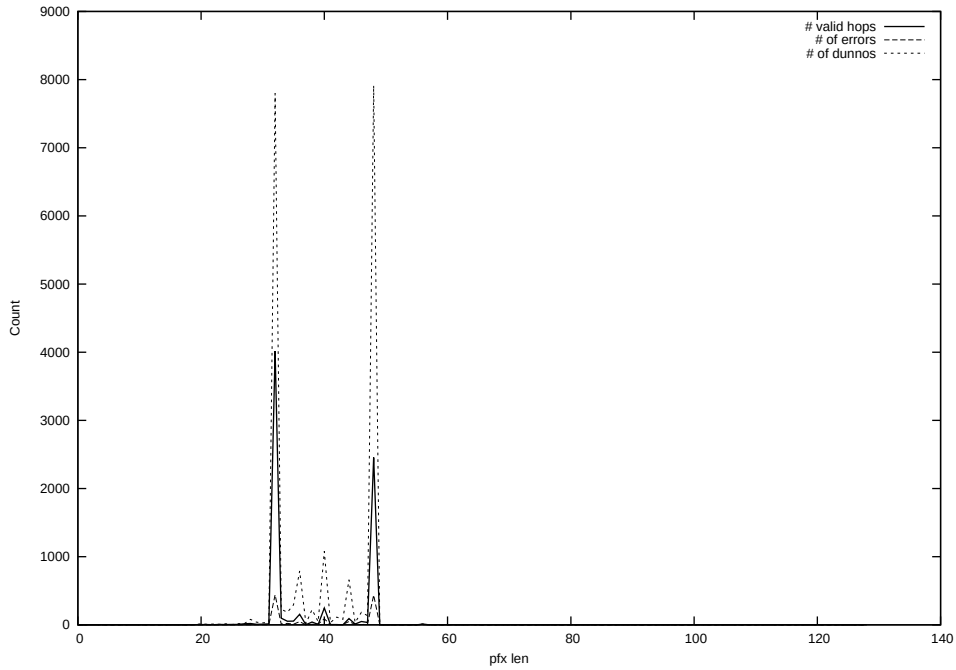
2012-06-04



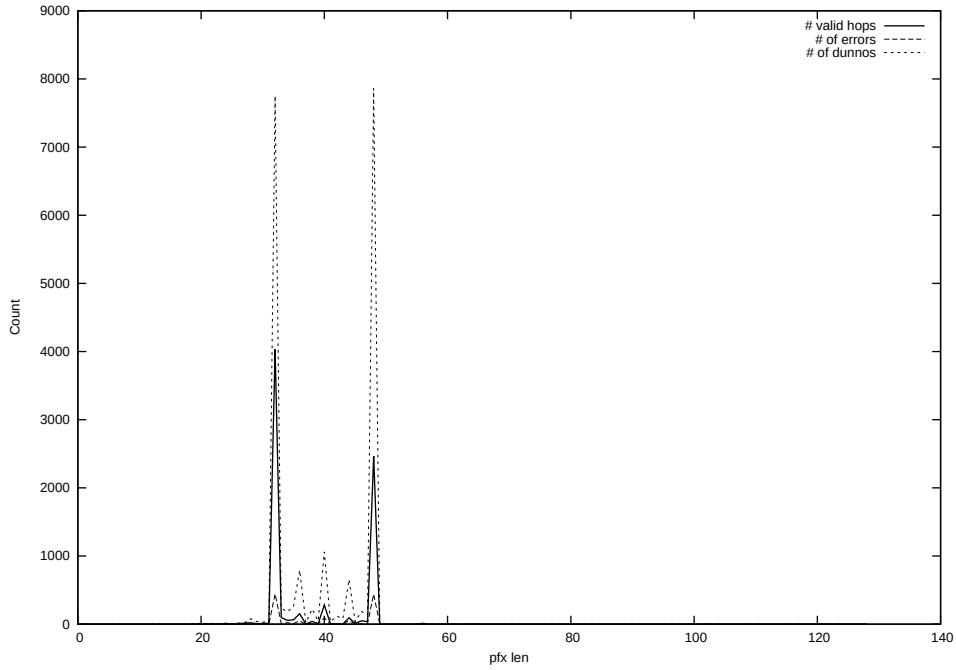
2012-06-05



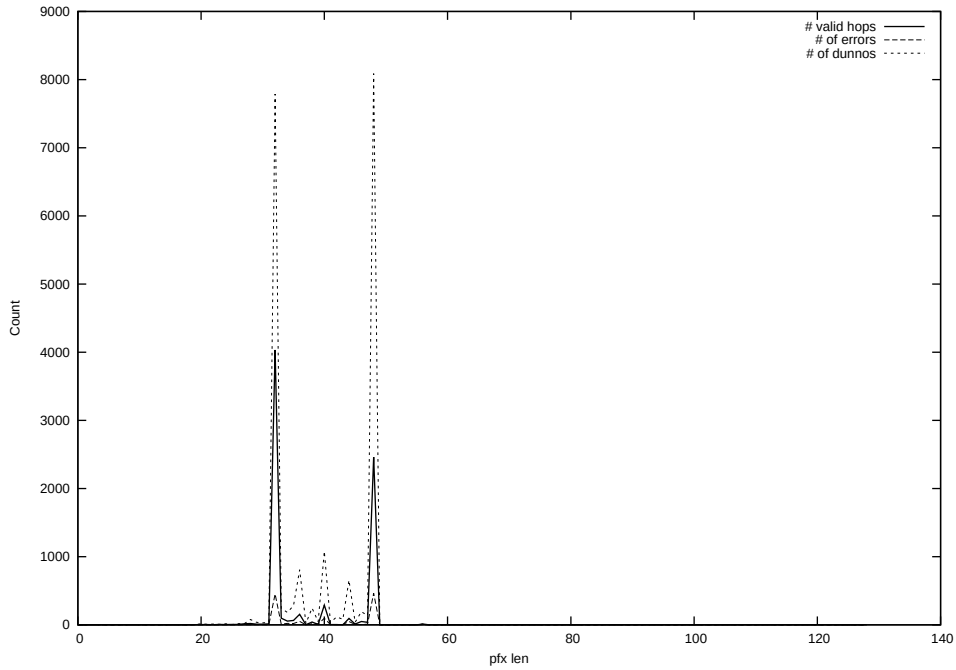
2012-06-06



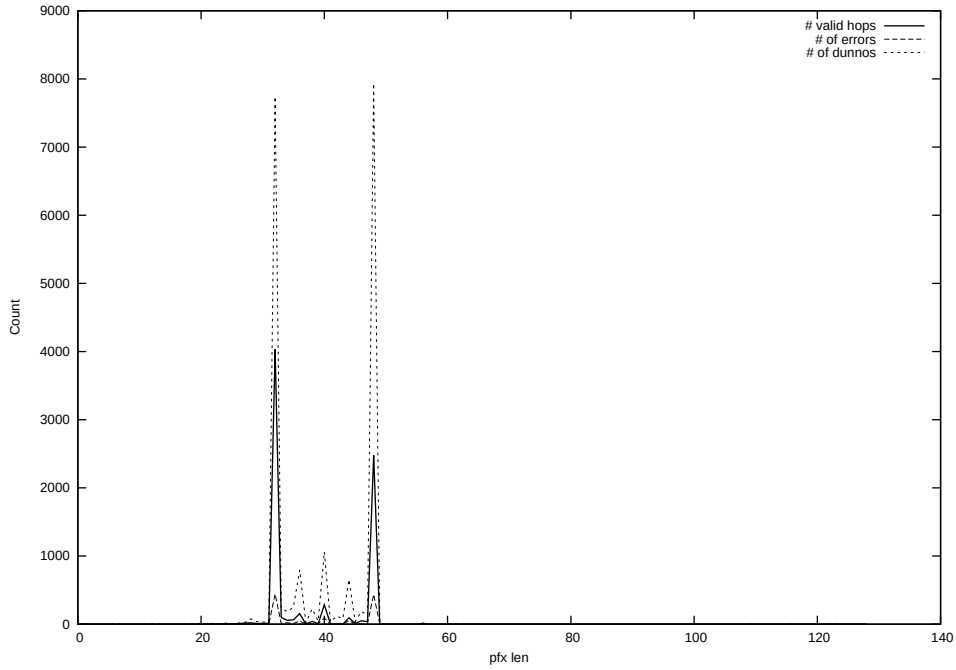
2012-06-07



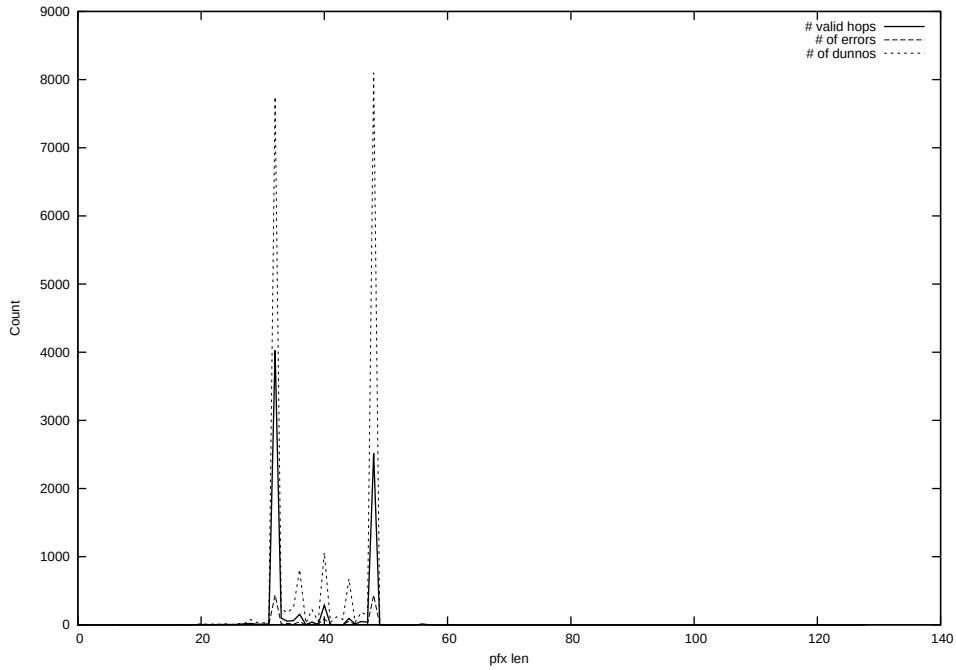
2012-06-08



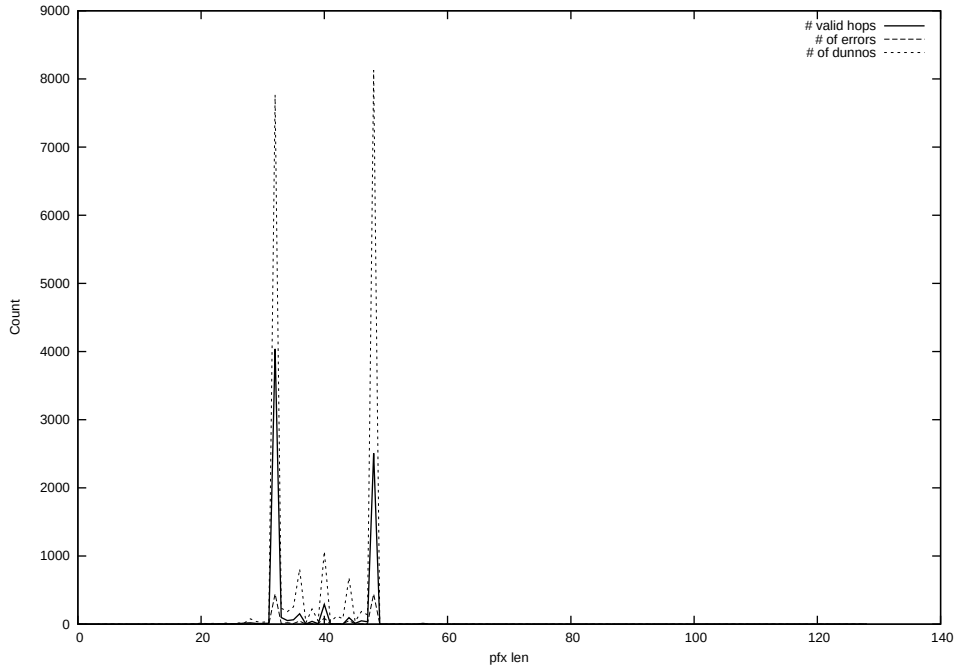
2012-06-10



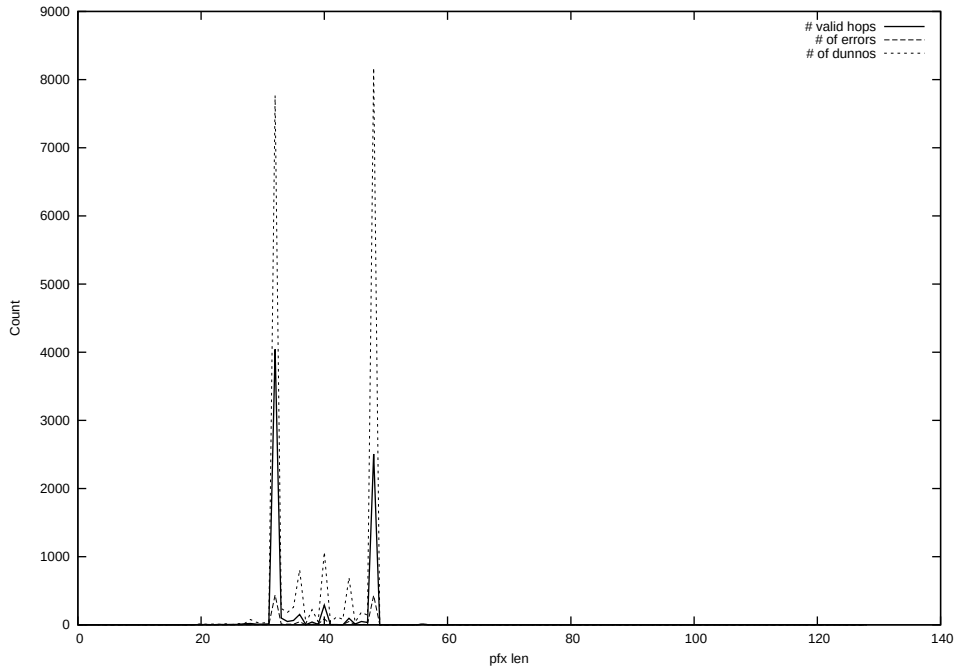
2012-06-11



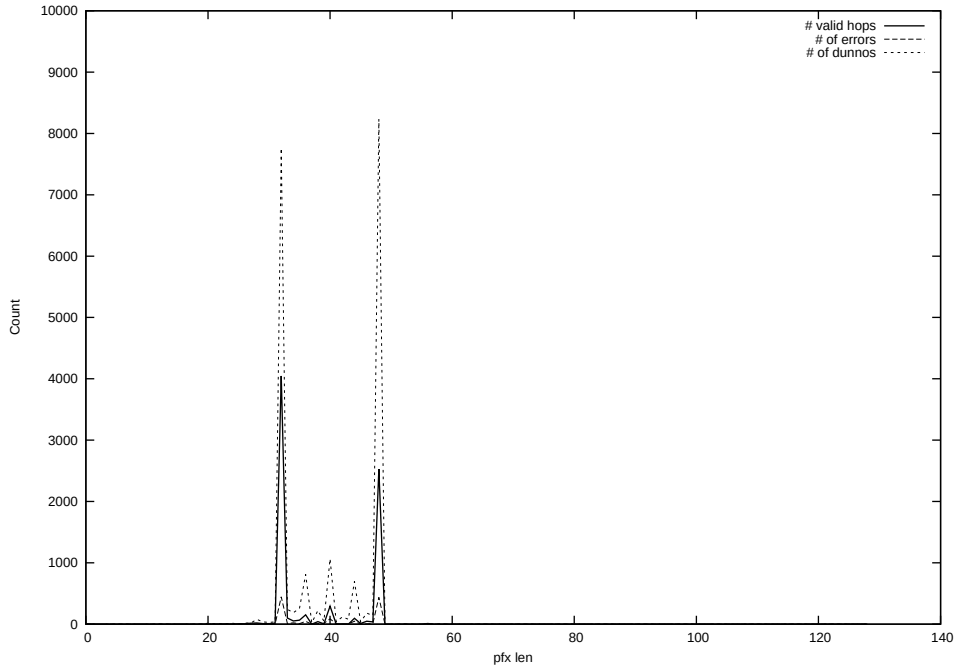
2012-06-12



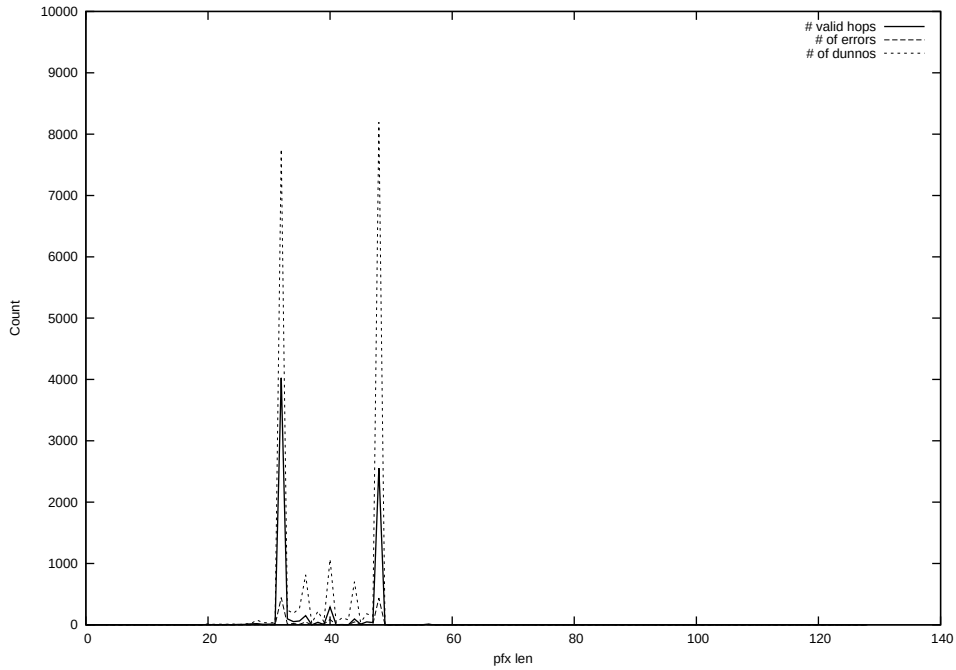
2012-06-13



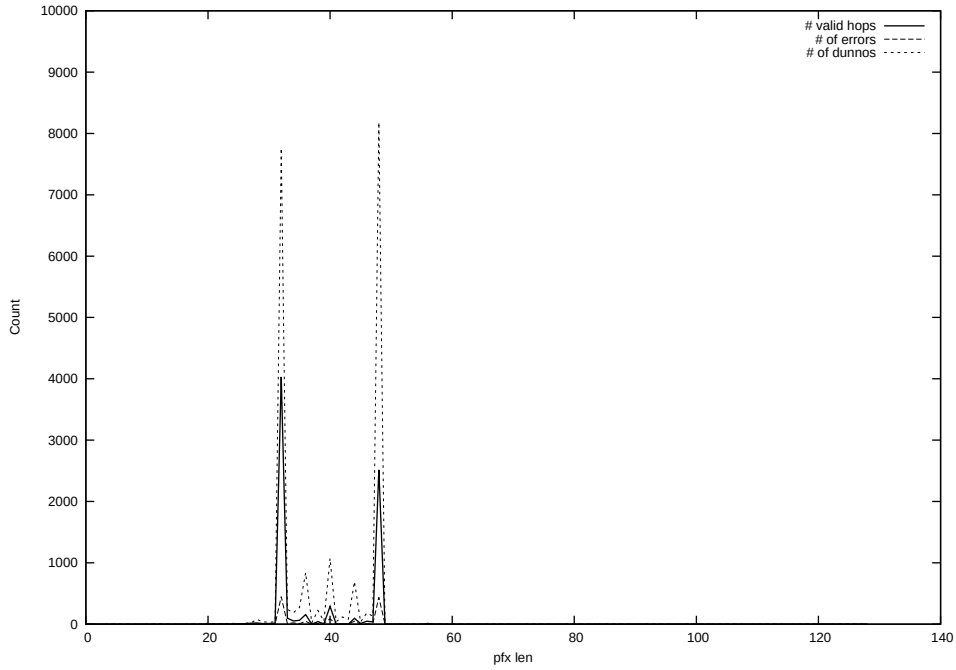
2012-06-14



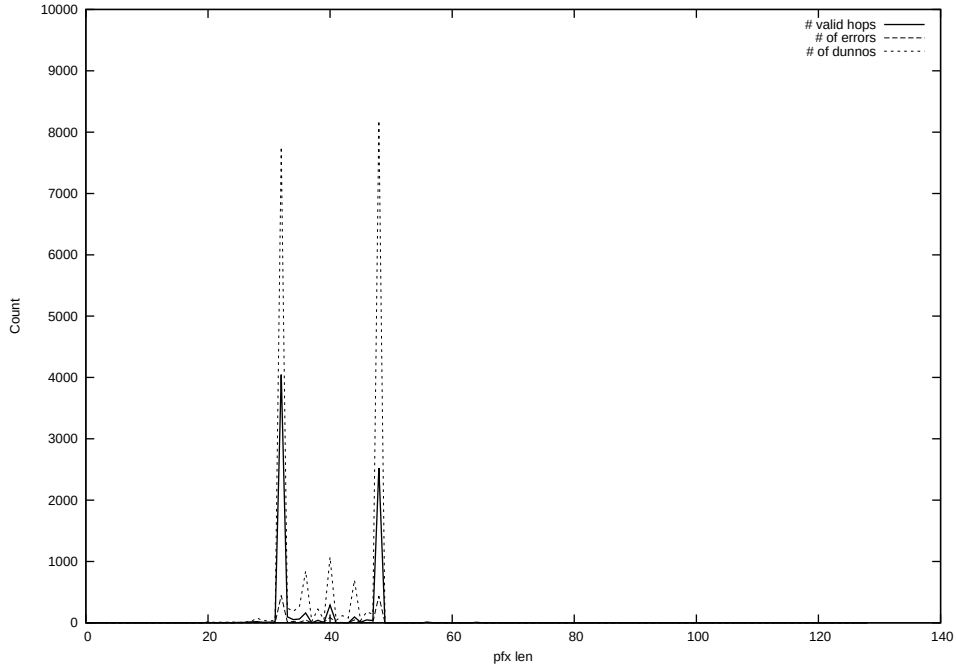
2012-06-15



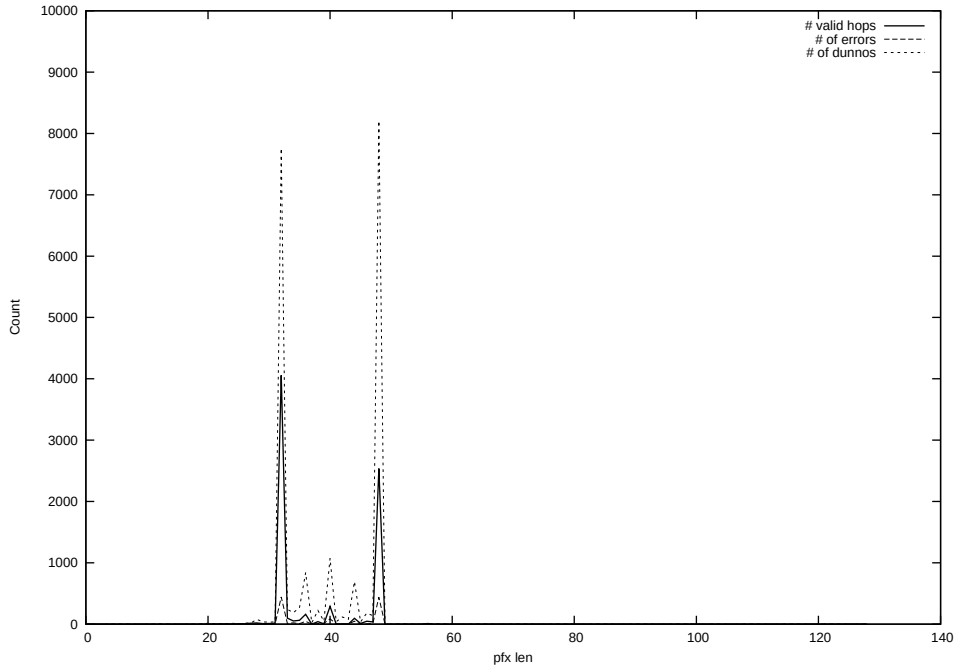
2012-06-16



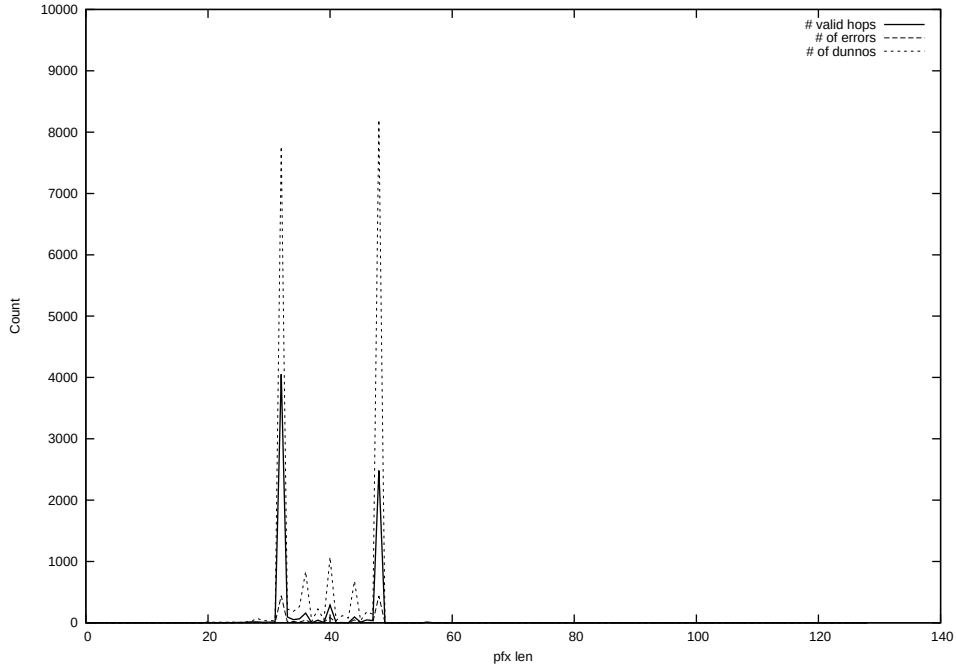
2012-06-17



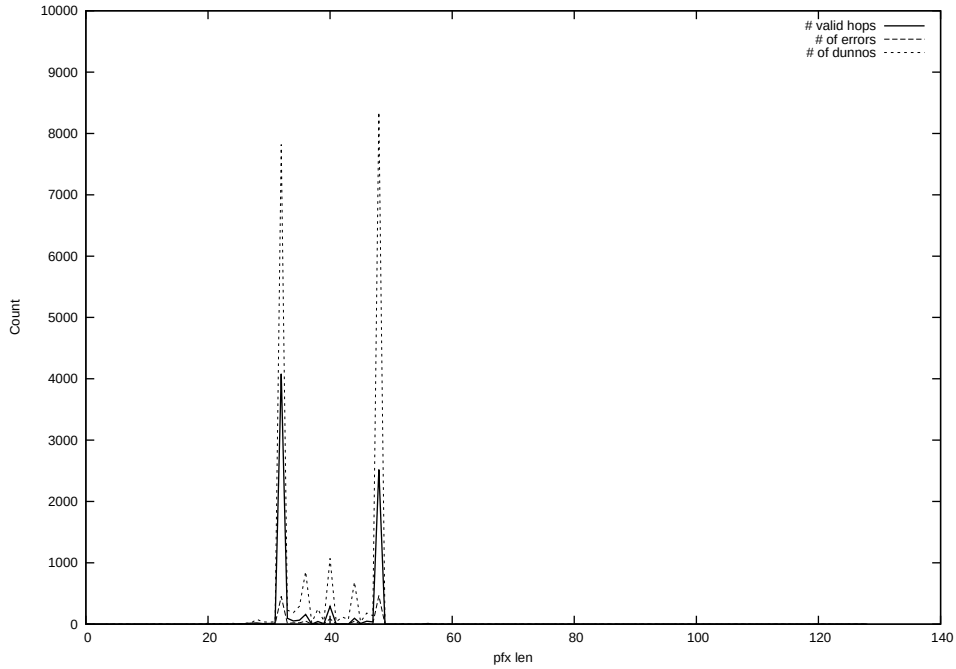
2012-06-18



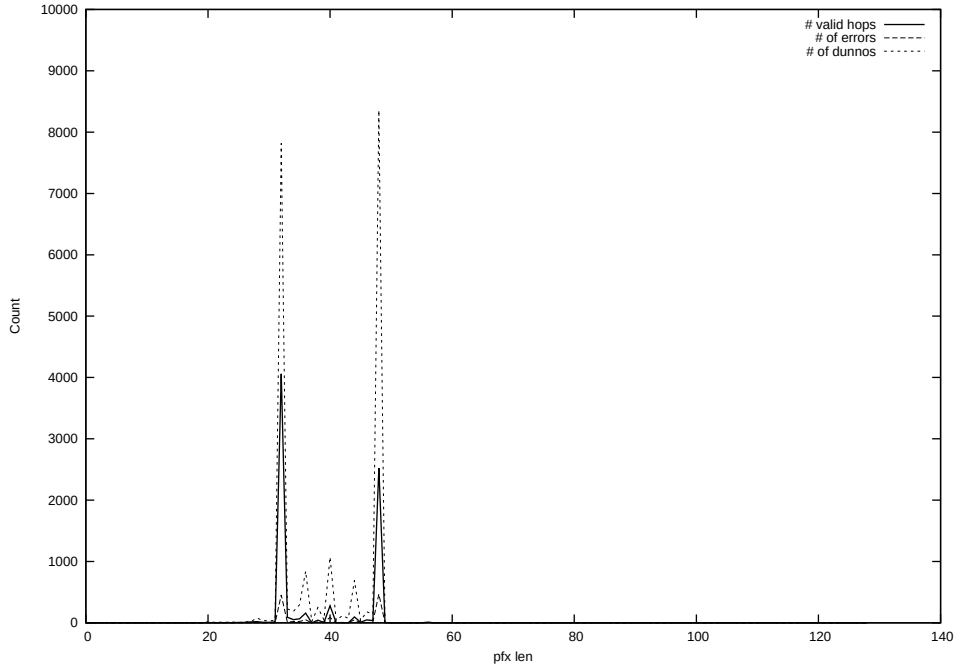
2012-06-19



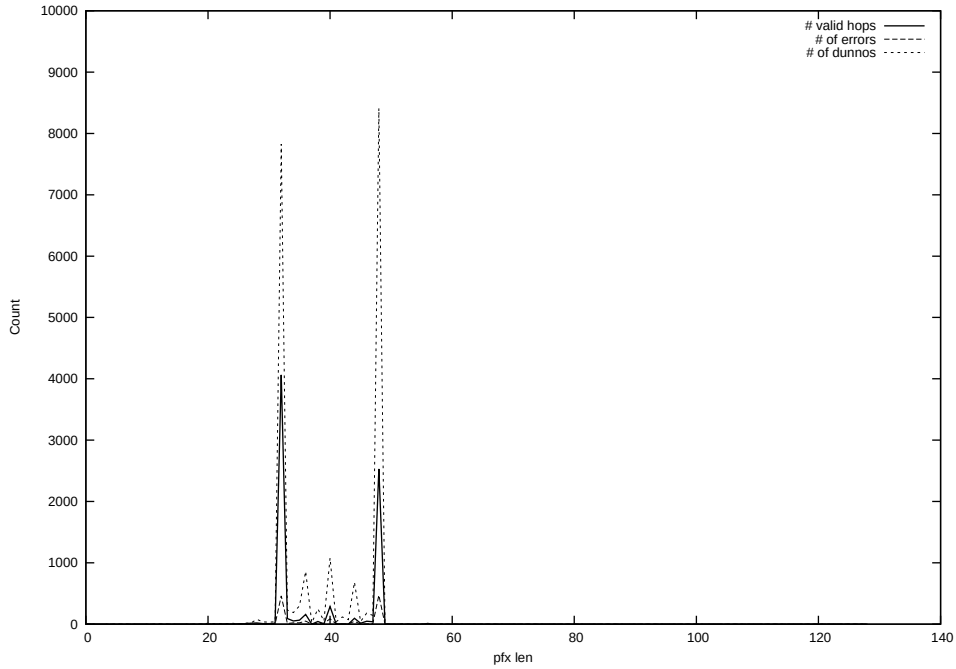
2012-06-20



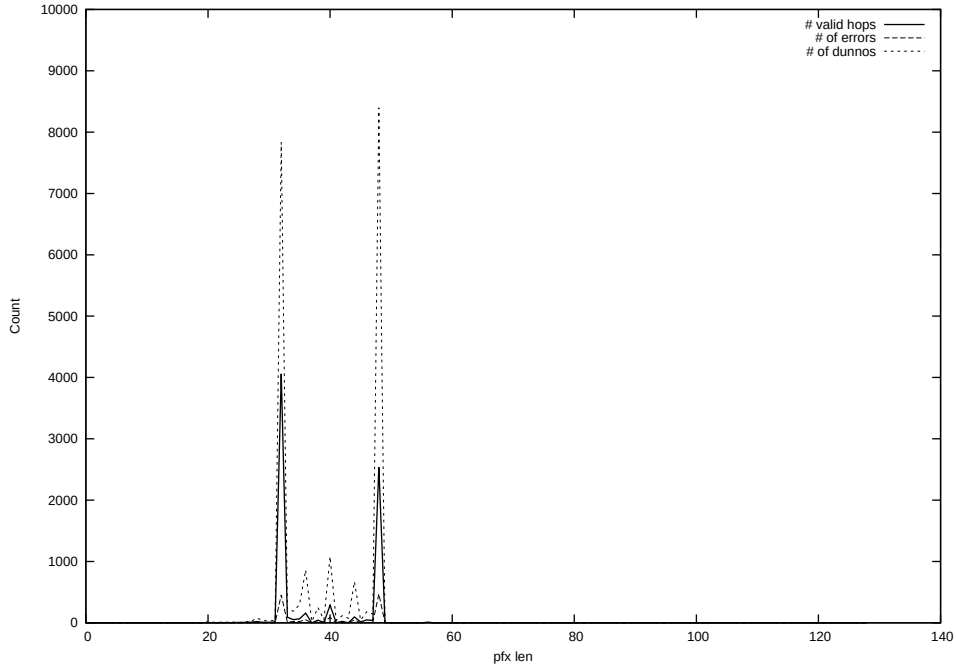
2012-06-21



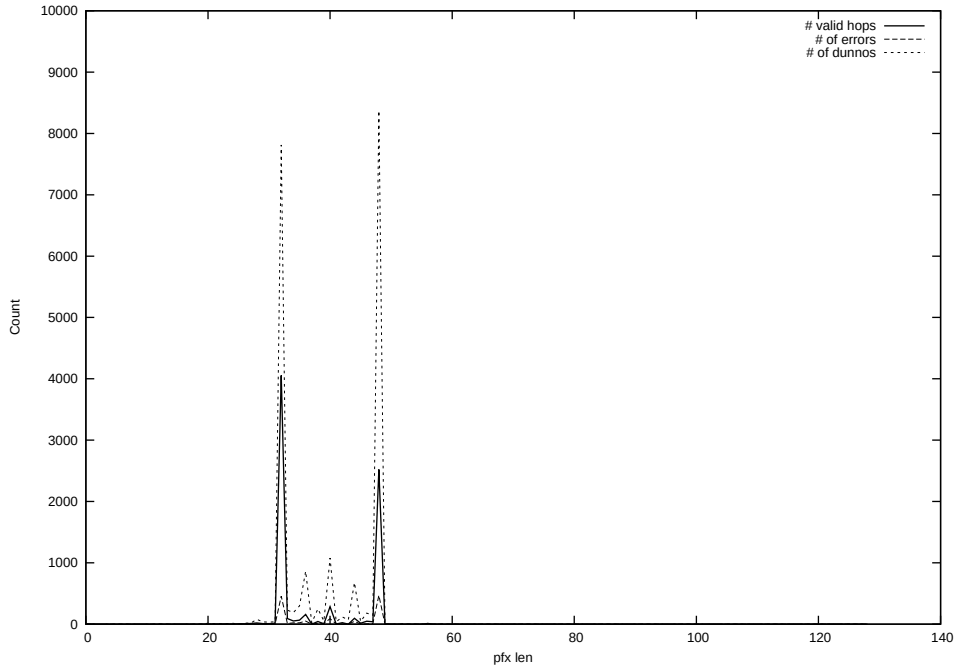
2012-06-22



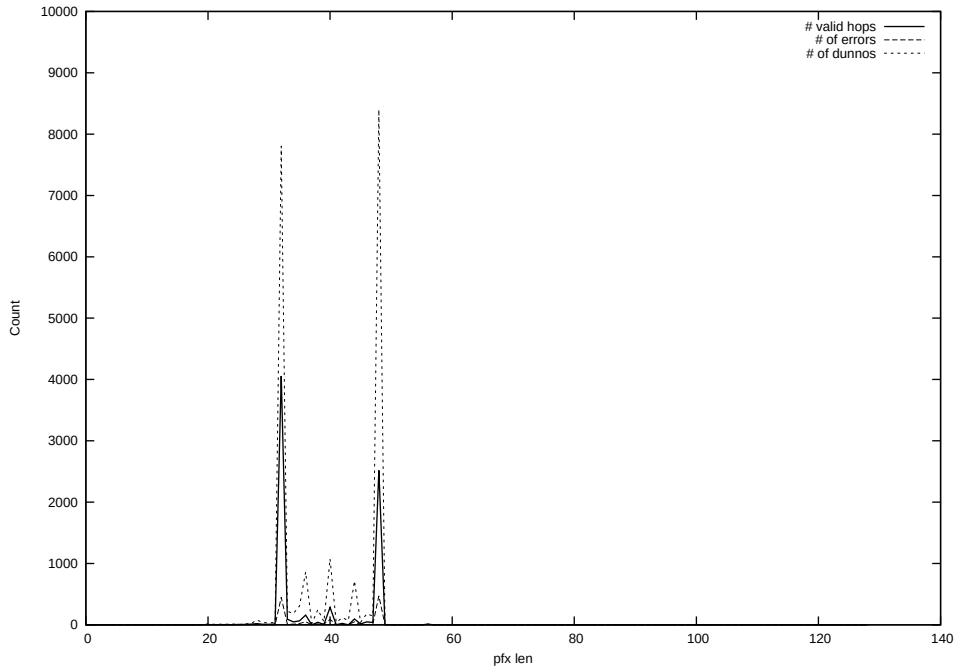
2012-06-23



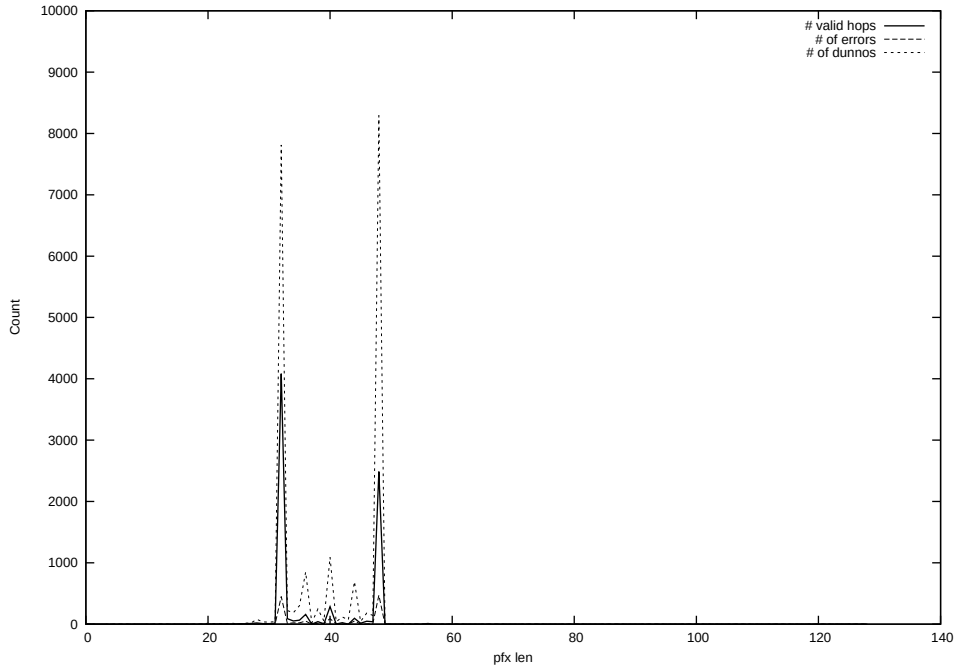
2012-06-24



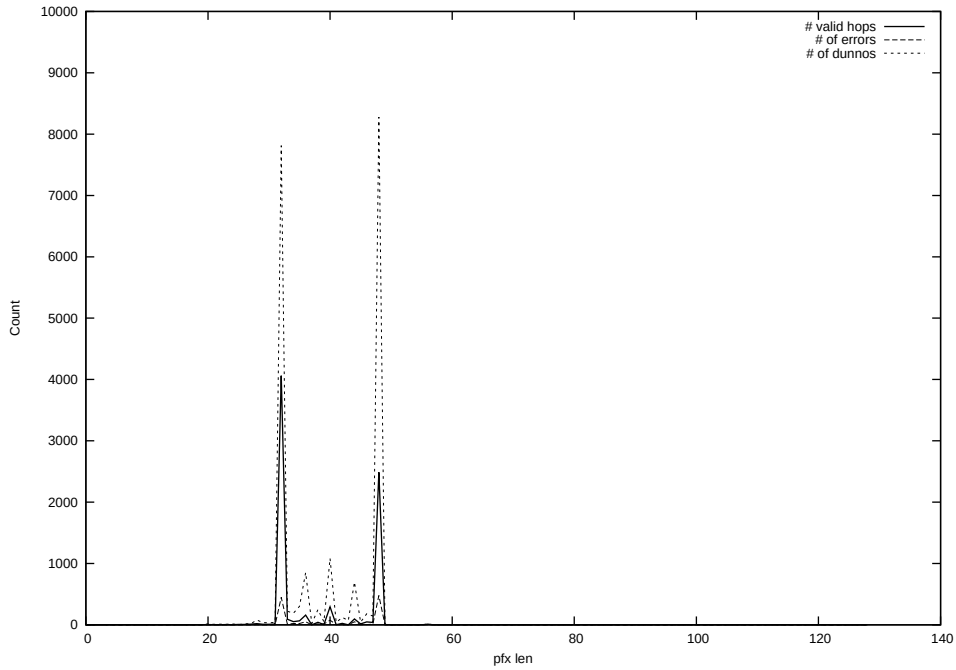
2012-06-25



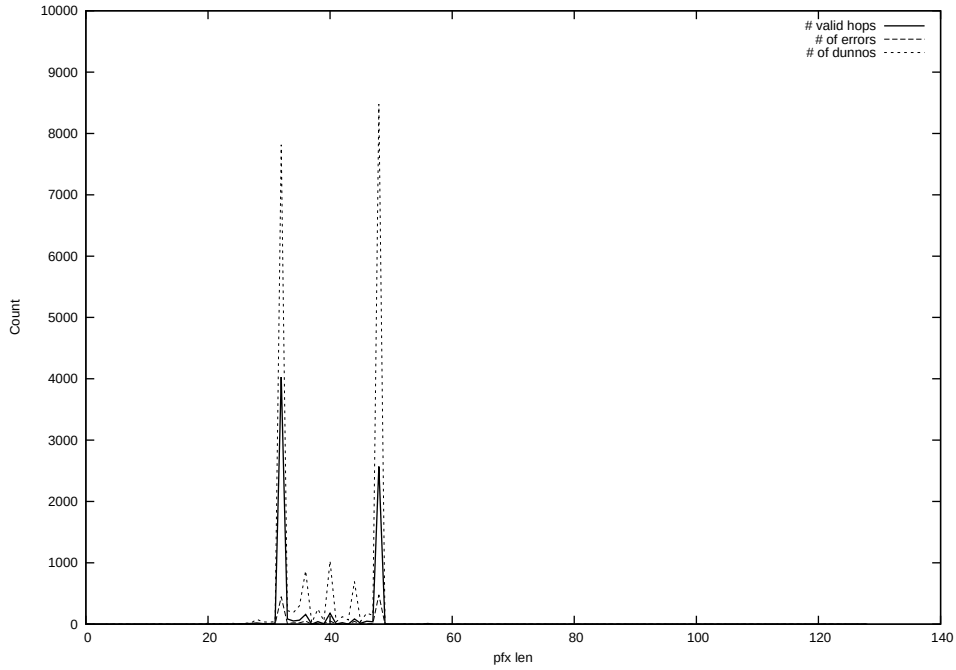
2012-06-26



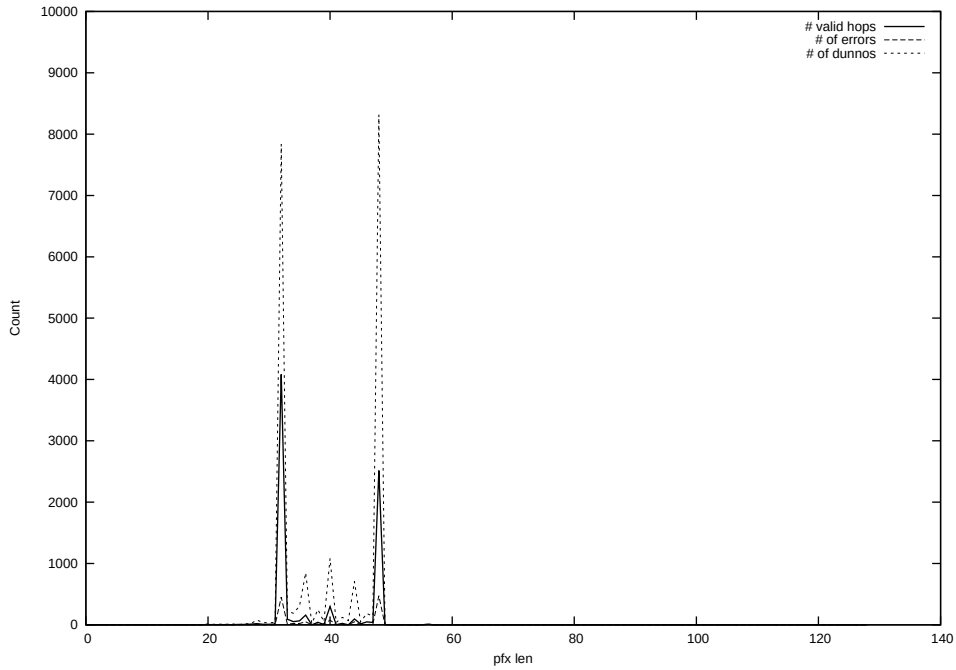
2012-06-27



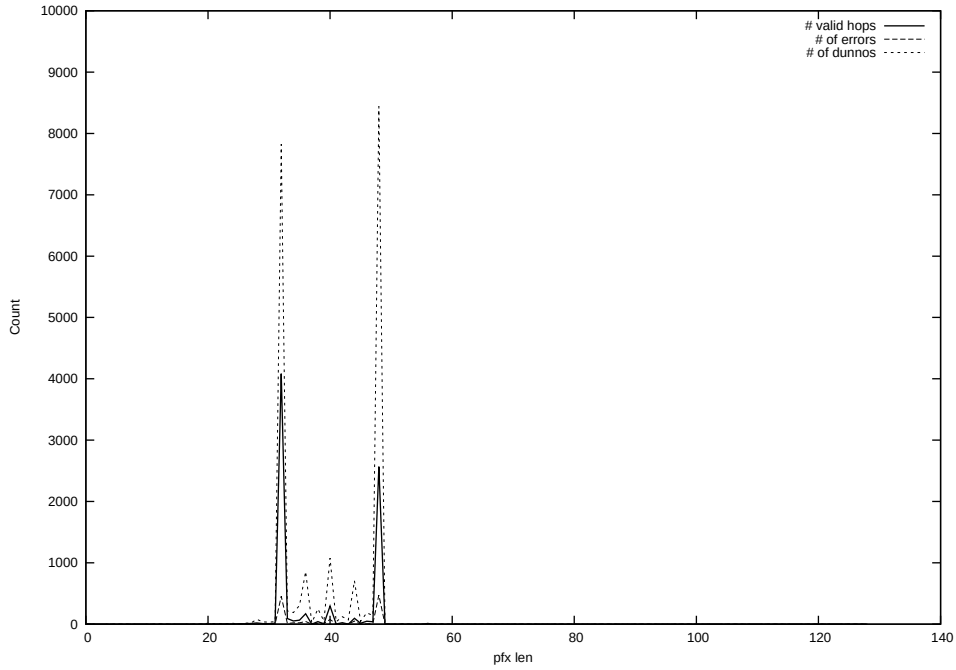
2012-06-28



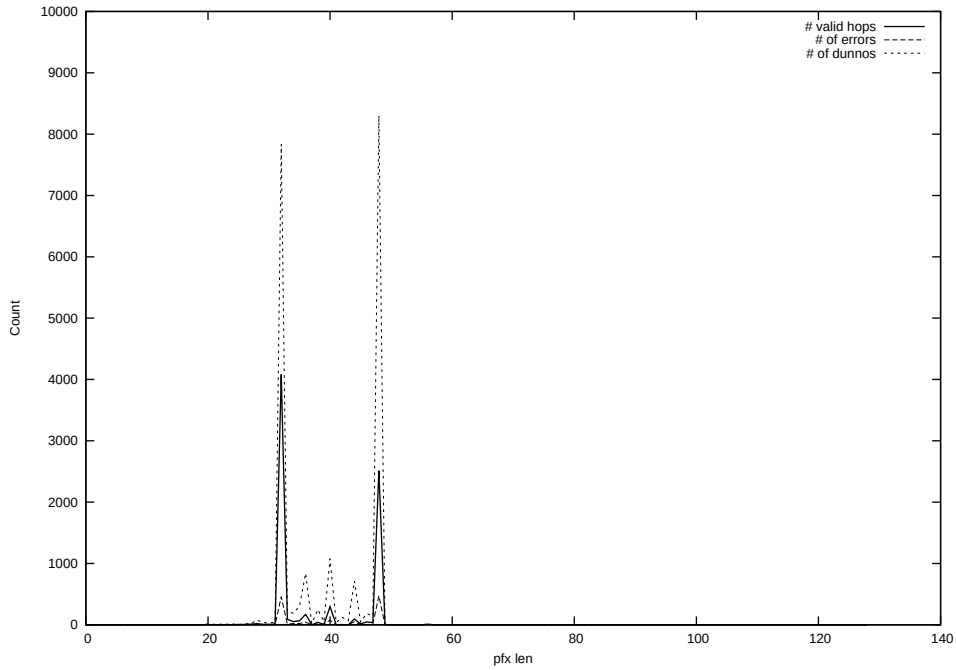
2012-06-29



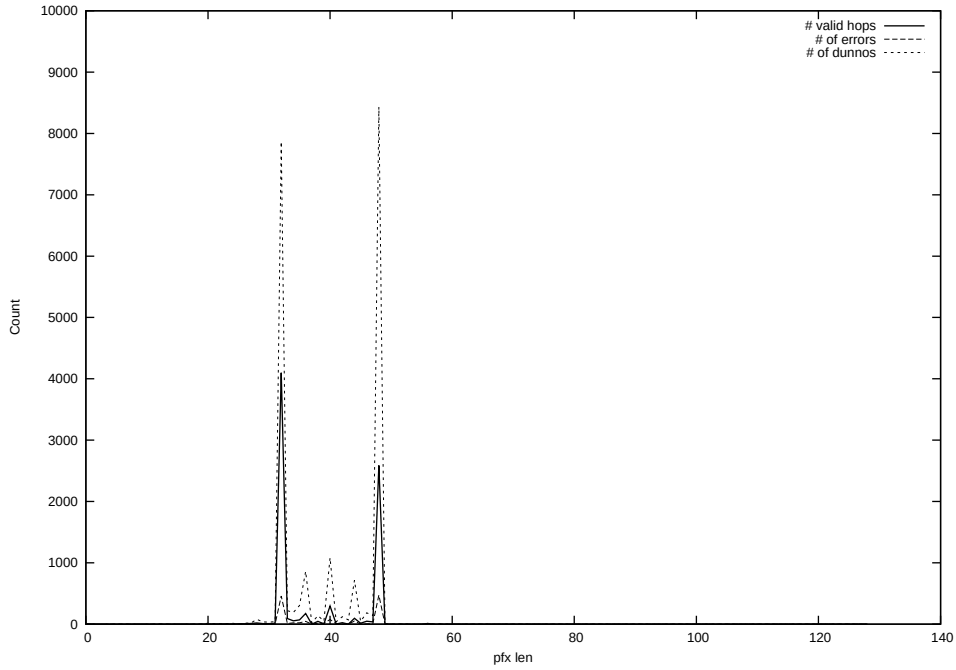
2012-06-30



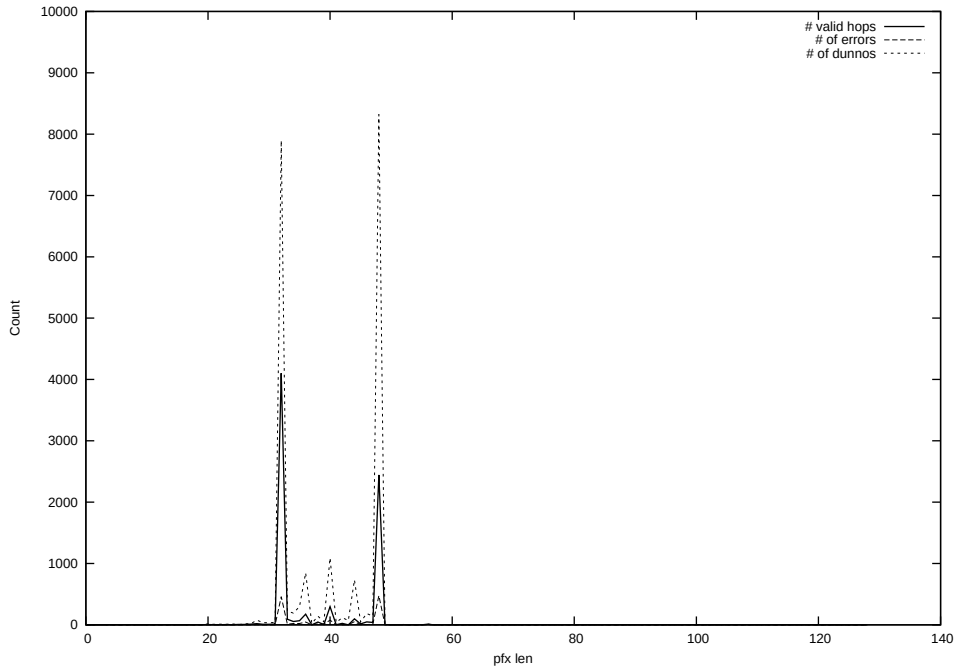
2012-07-01



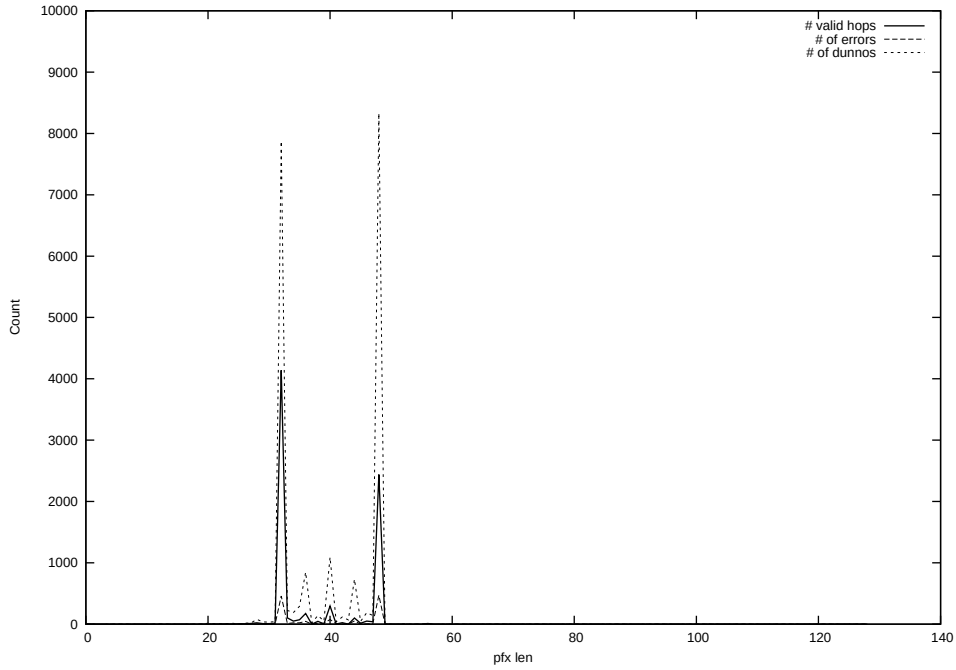
2012-07-02



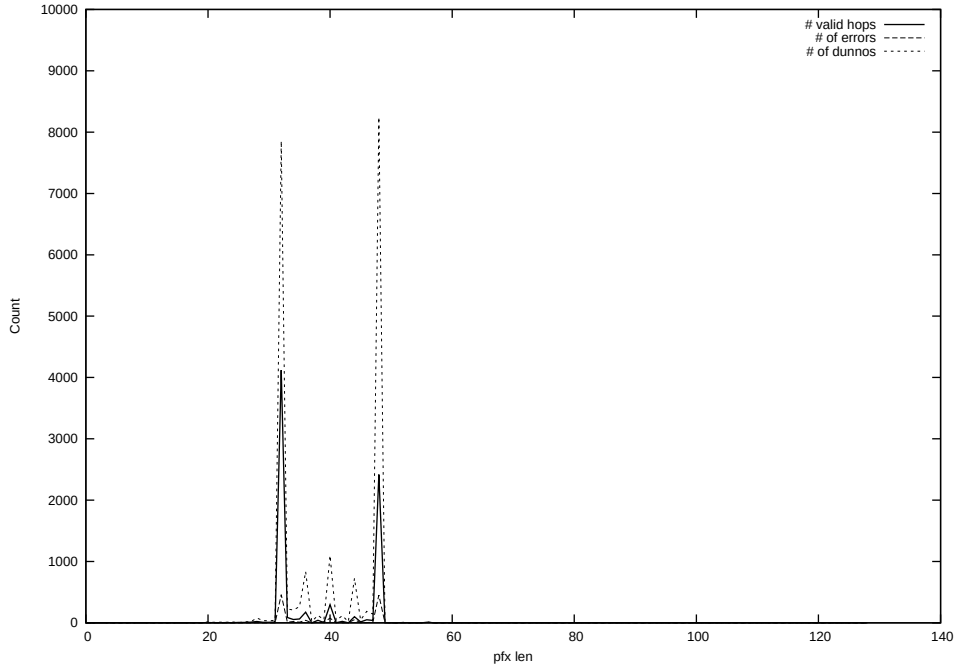
2012-07-03



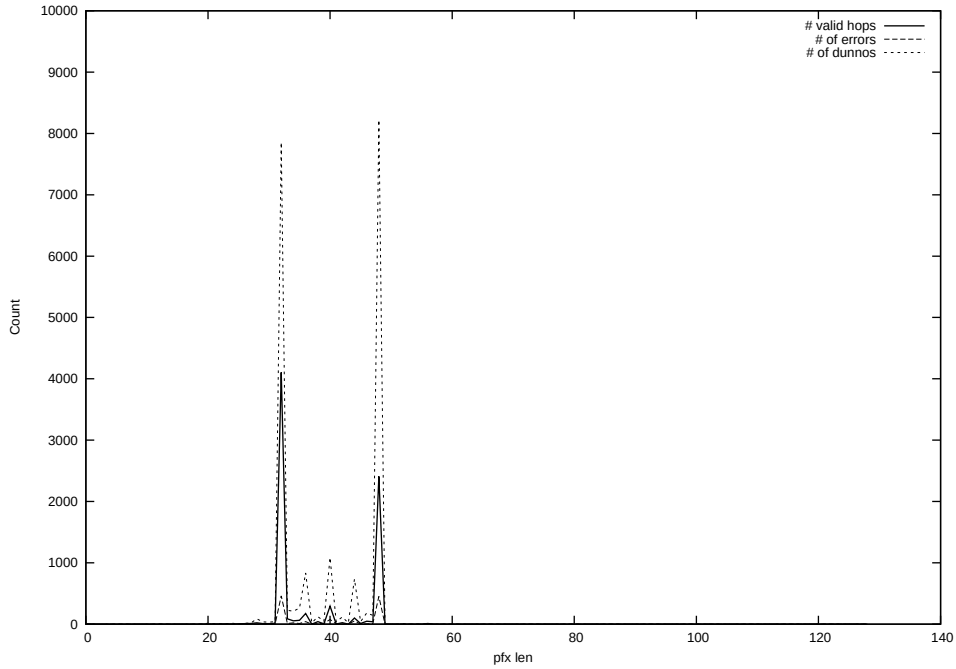
2012-07-04



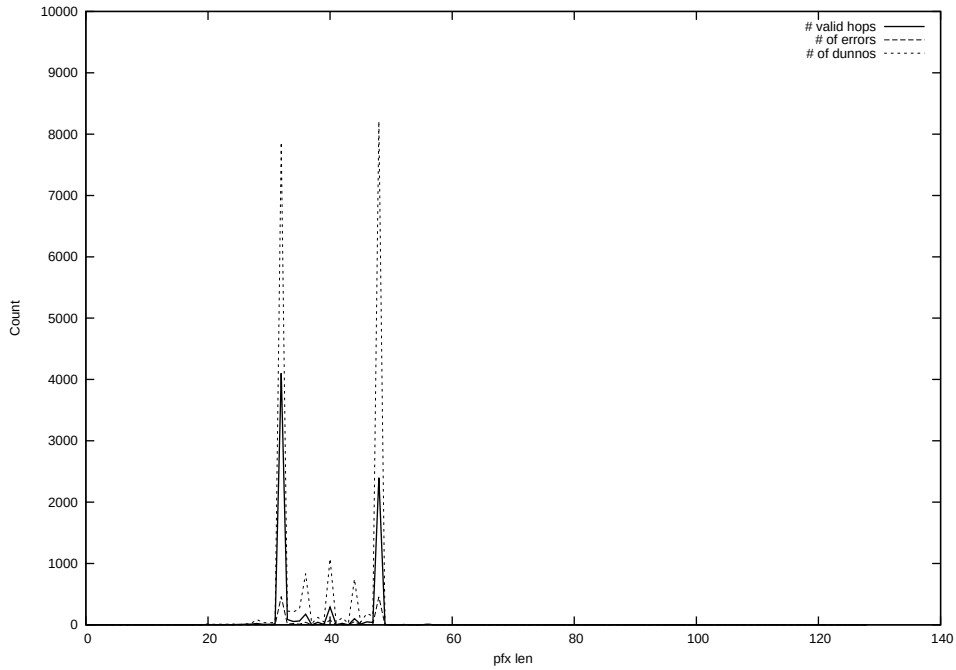
2012-07-05



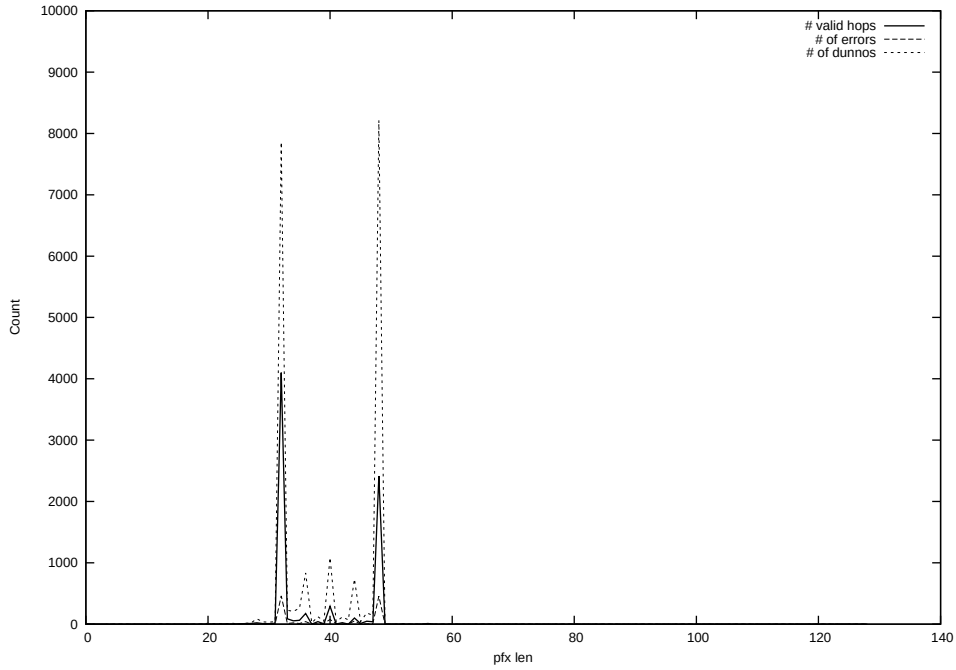
2012-07-06



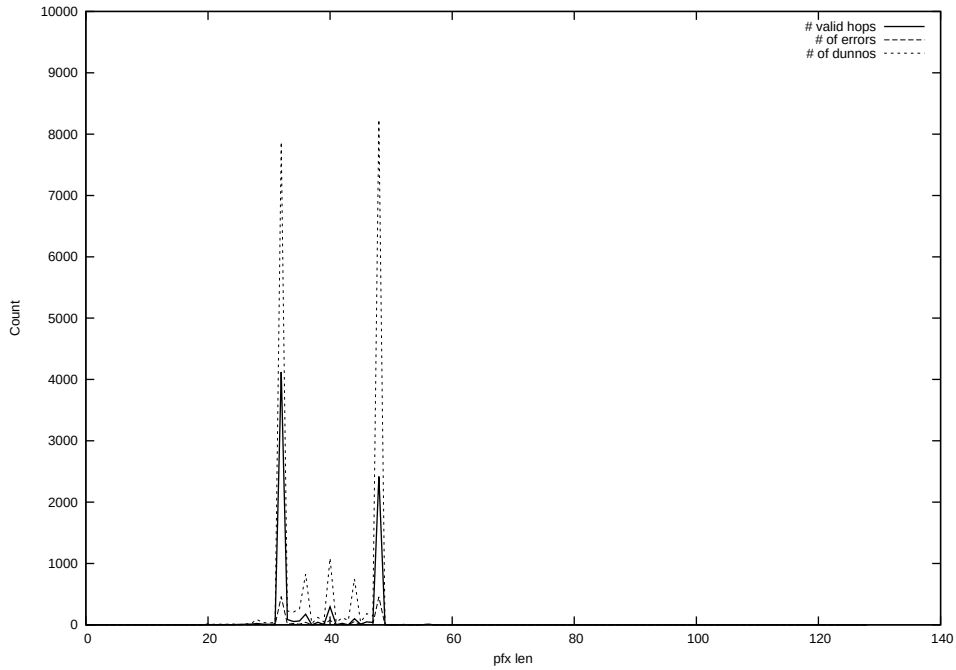
2012-07-07



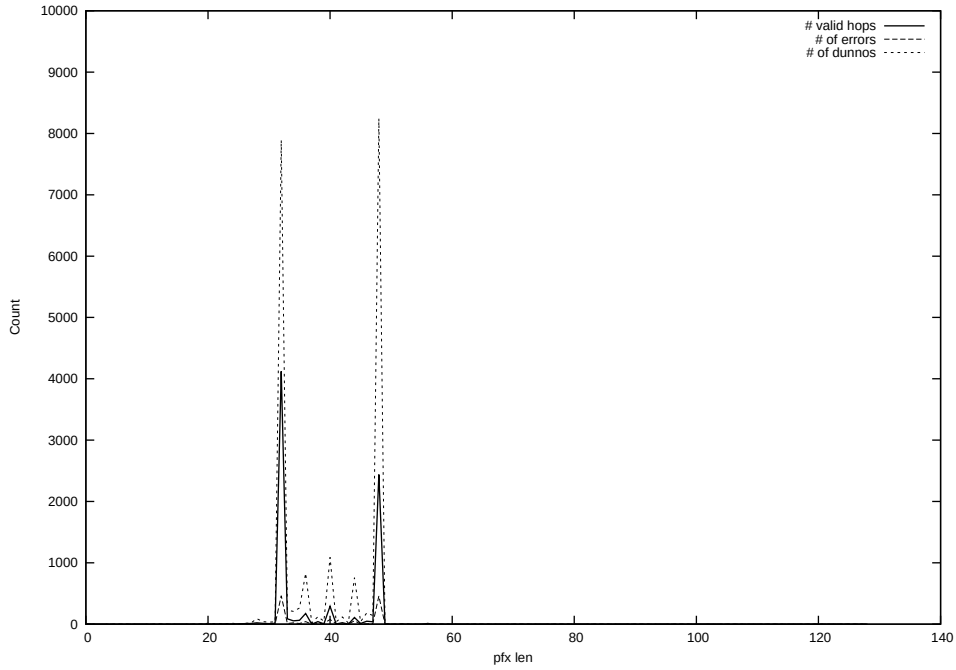
2012-07-08



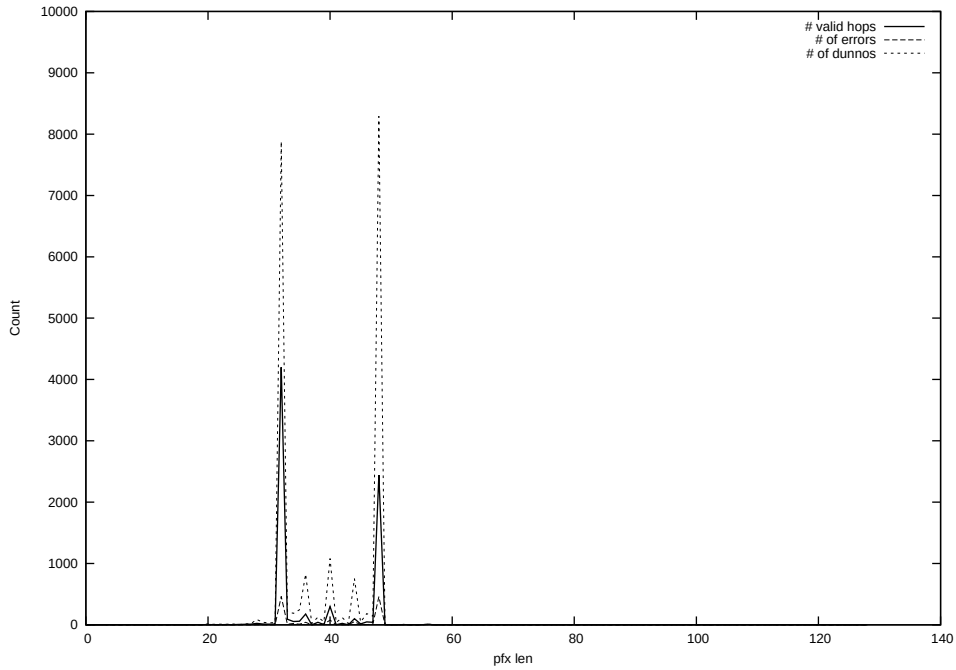
2012-07-09



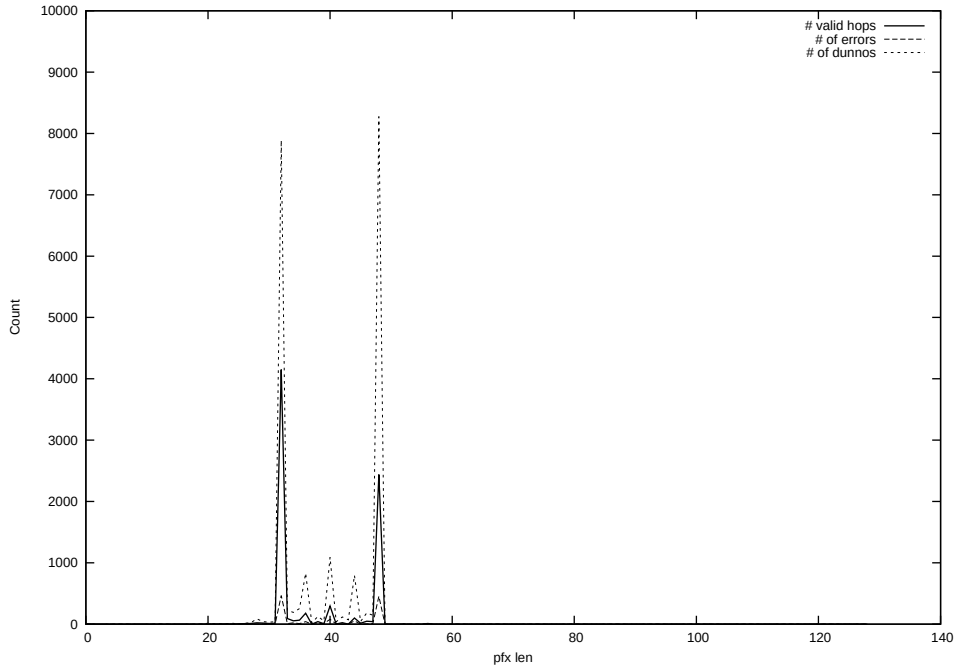
2012-07-10



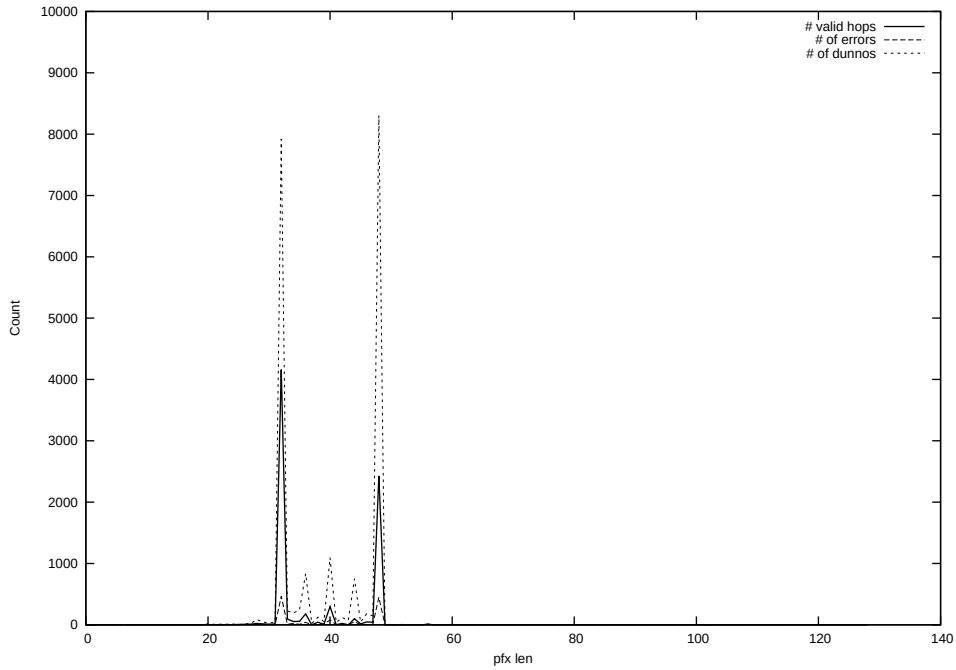
2012-07-11



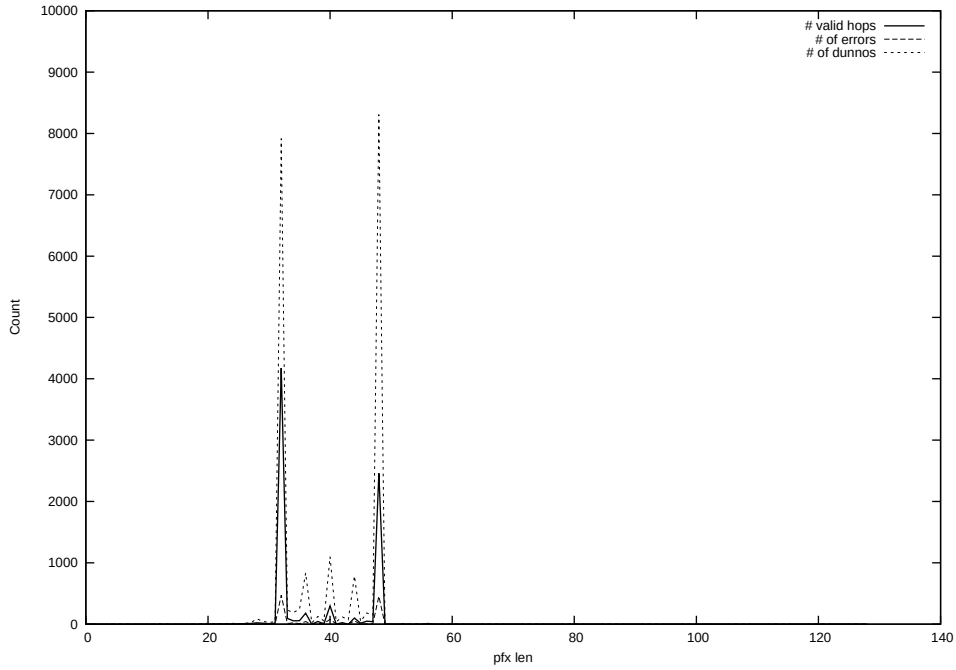
2012-07-12



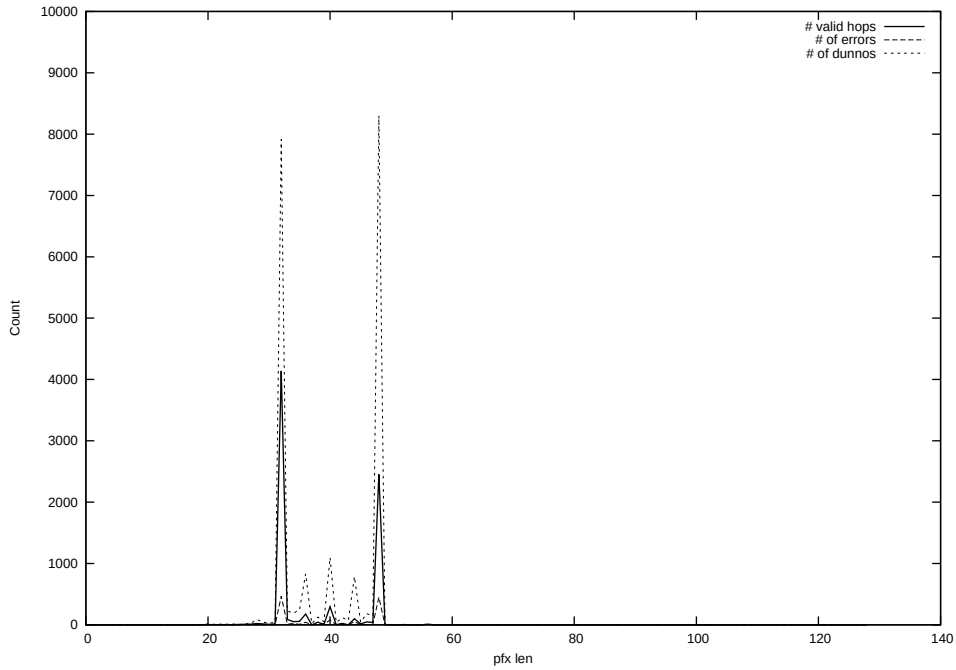
2012-07-13



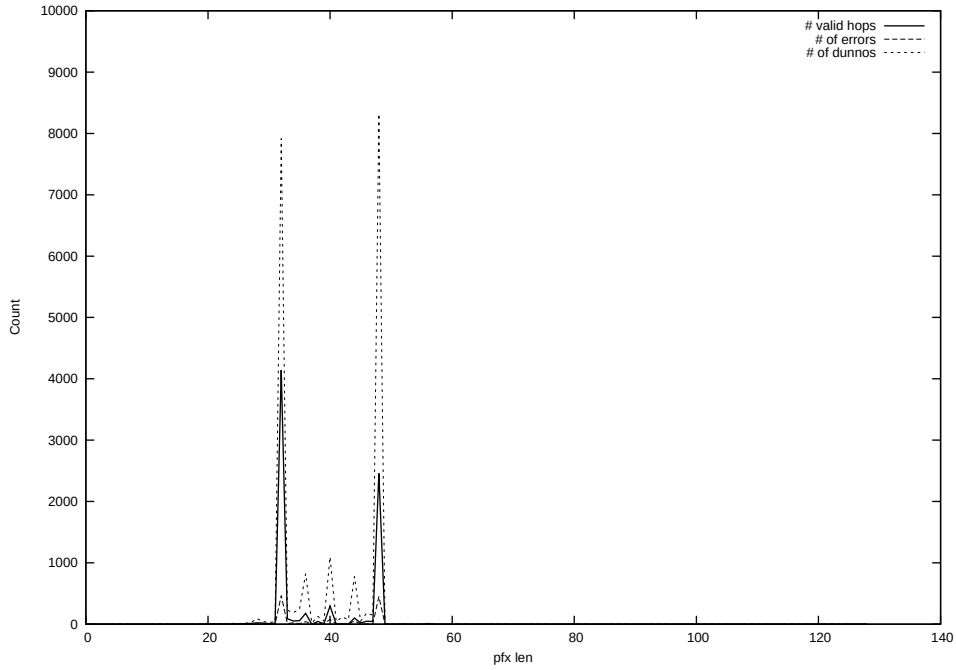
2012-07-14



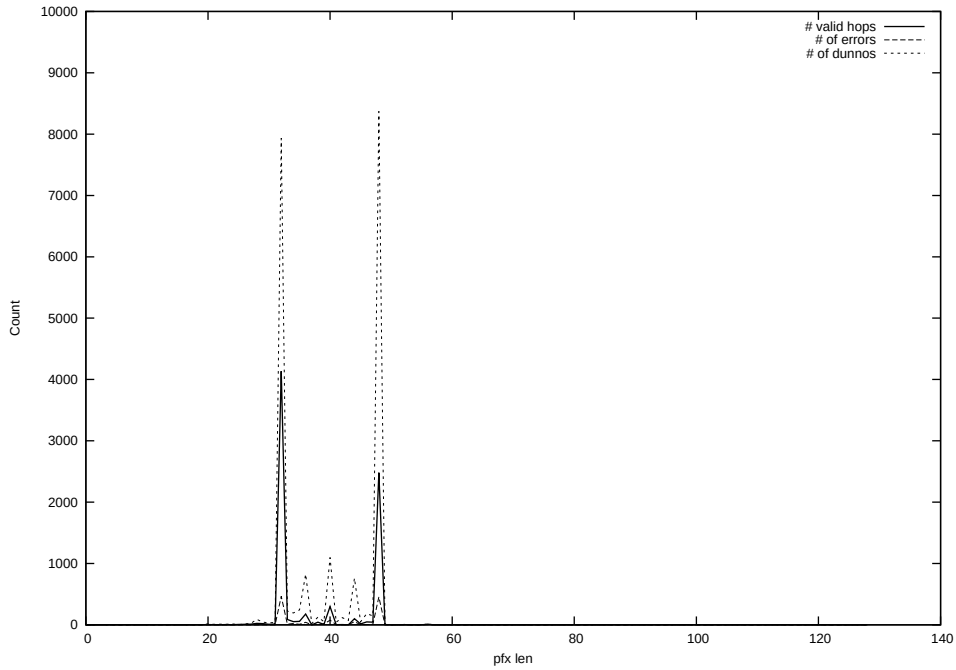
2012-07-15



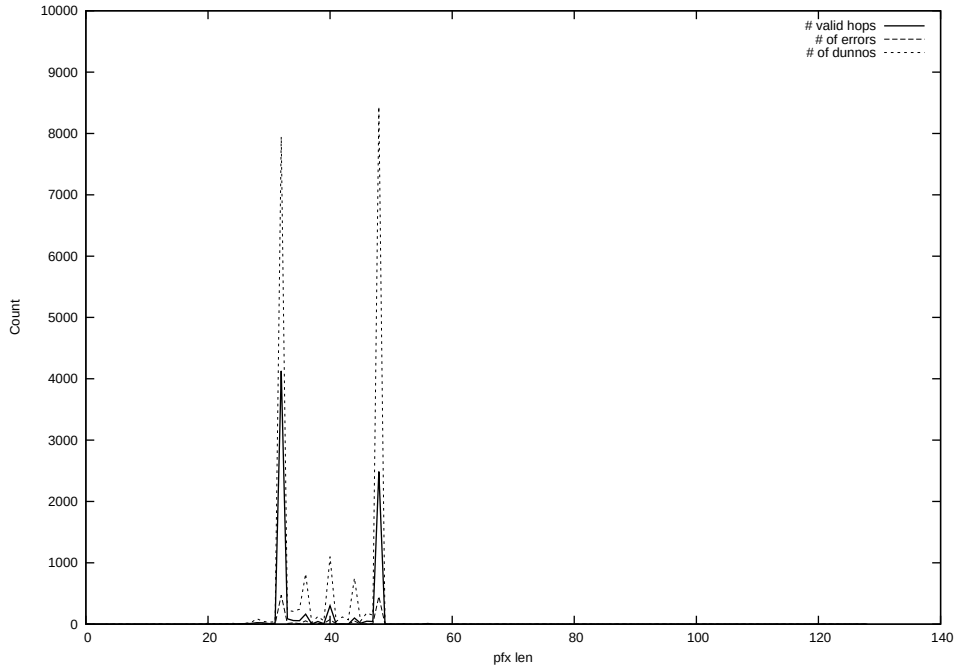
2012-07-16



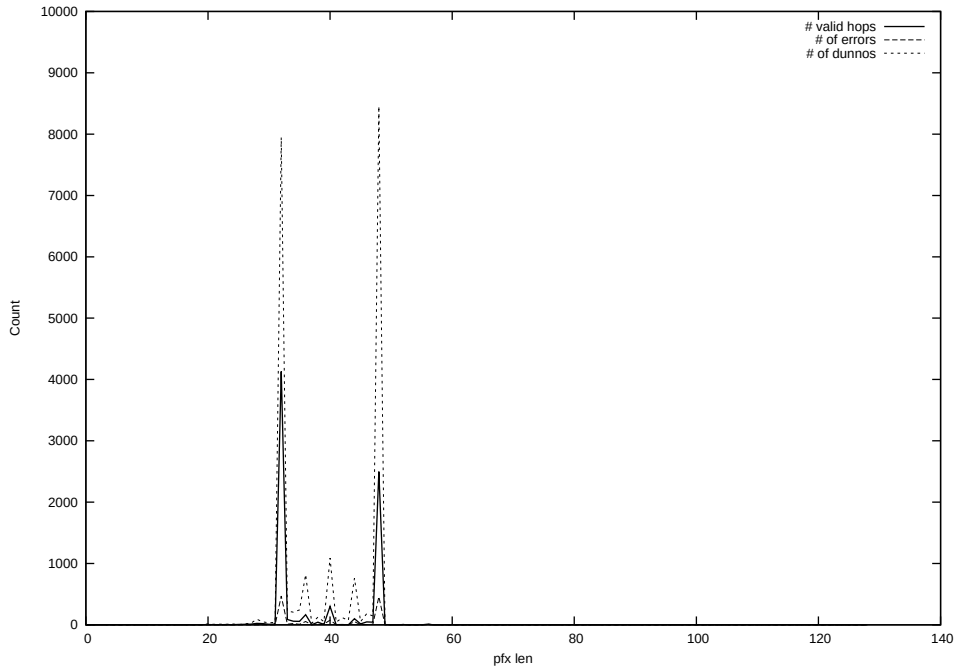
2012-07-17



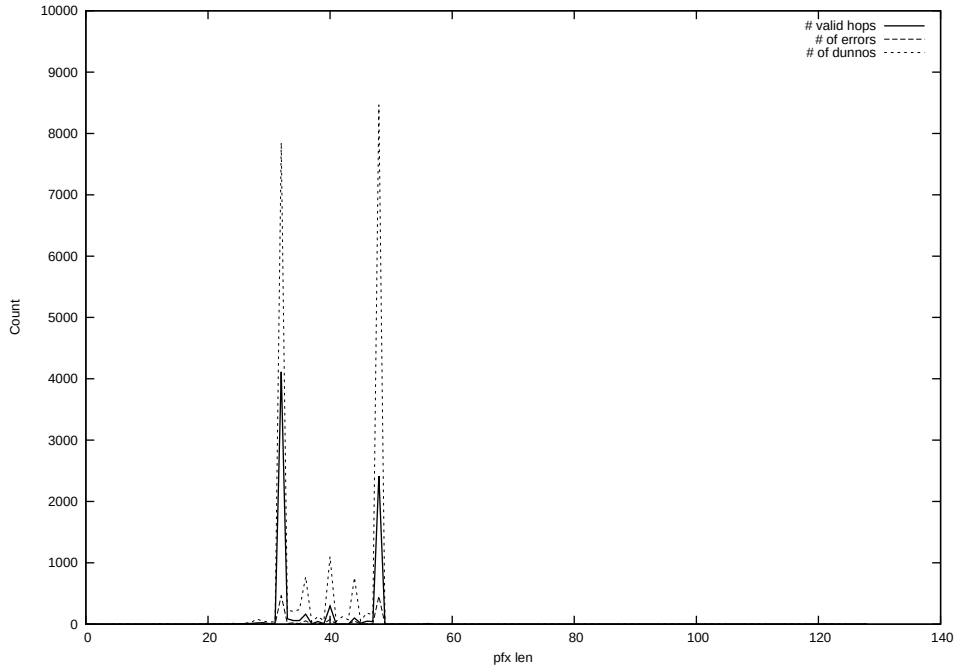
2012-07-18



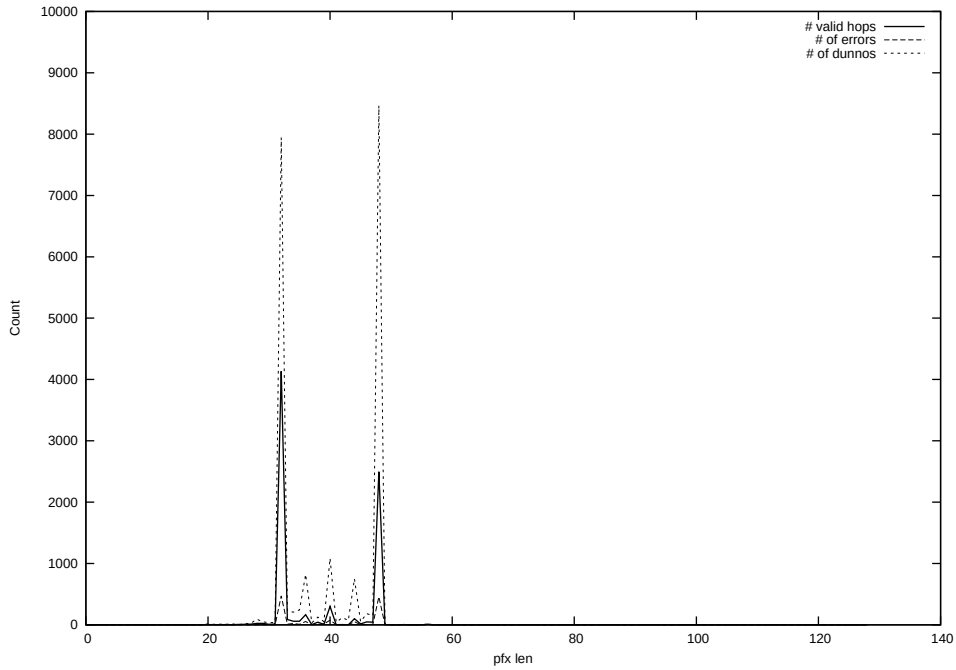
2012-07-19



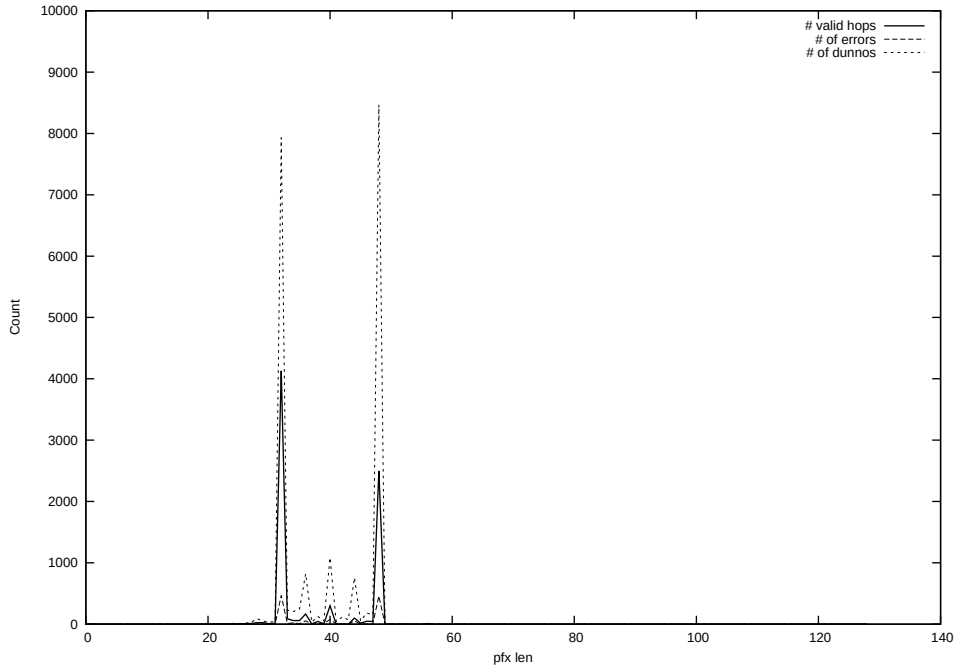
2012-07-20



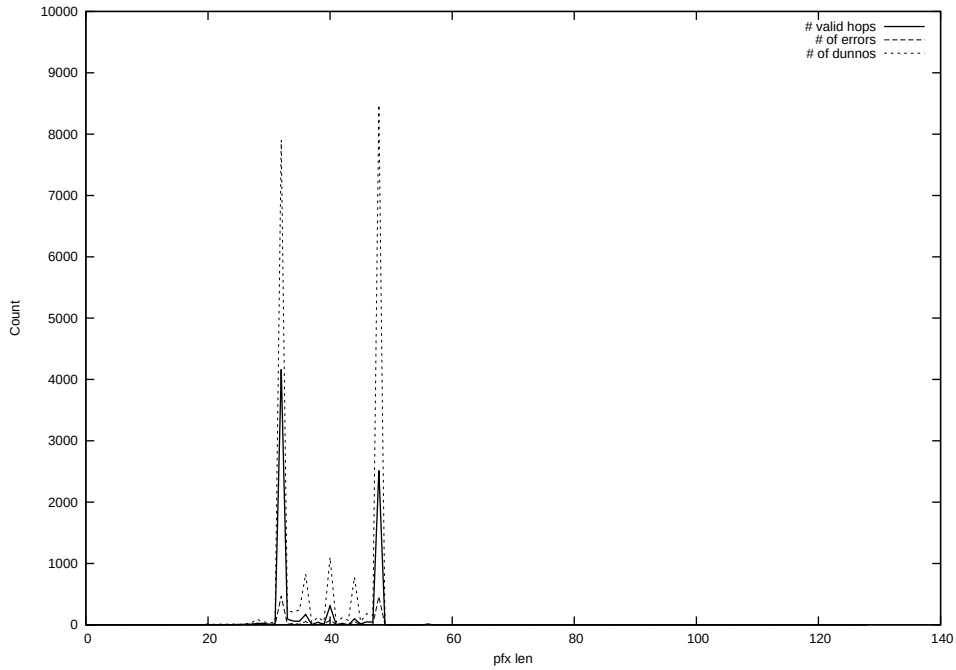
2012-07-21



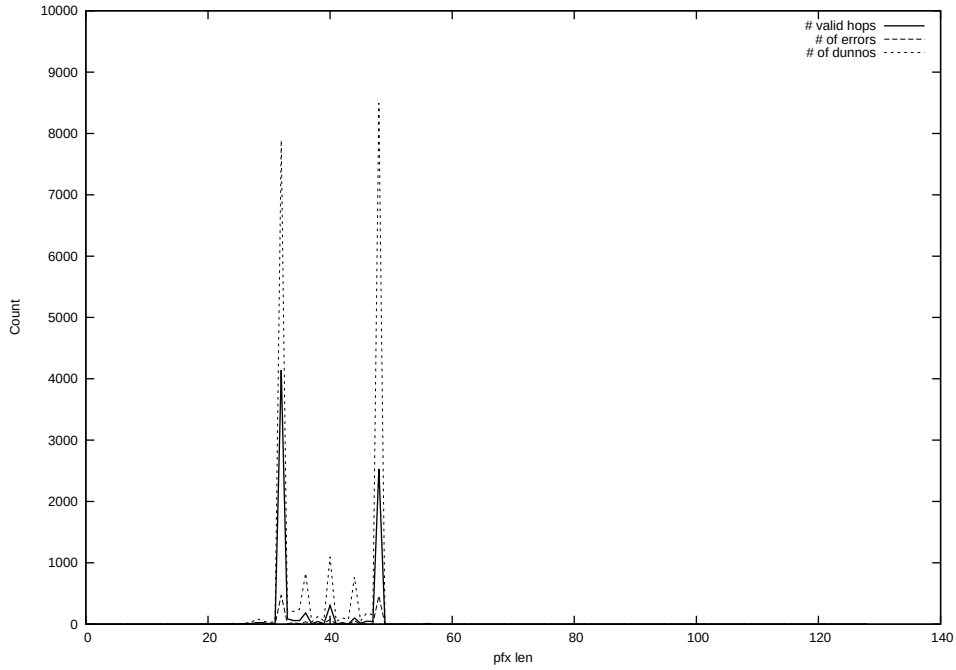
2012-07-22



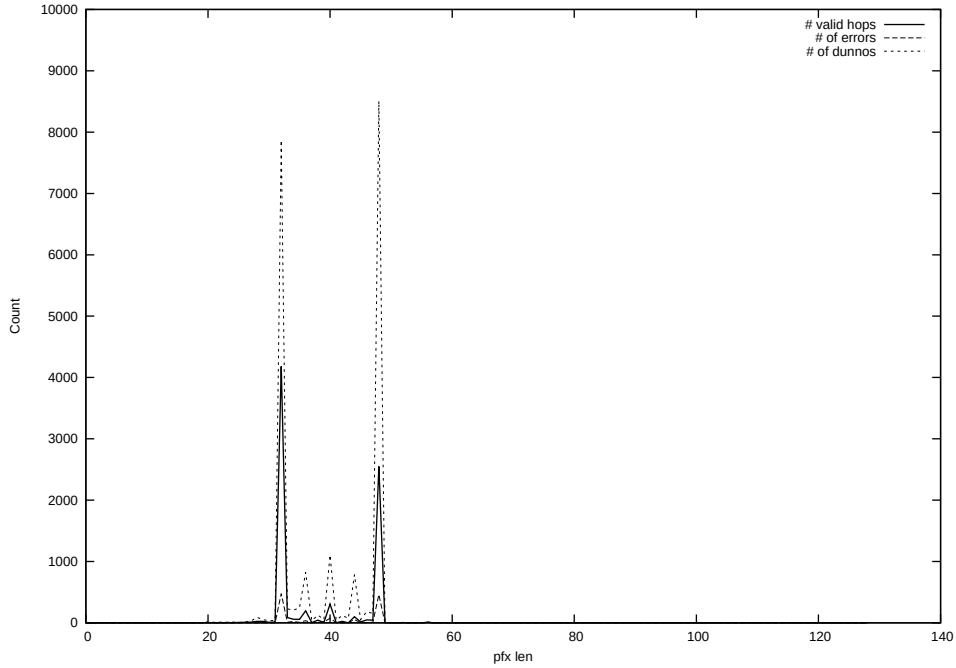
2012-07-23



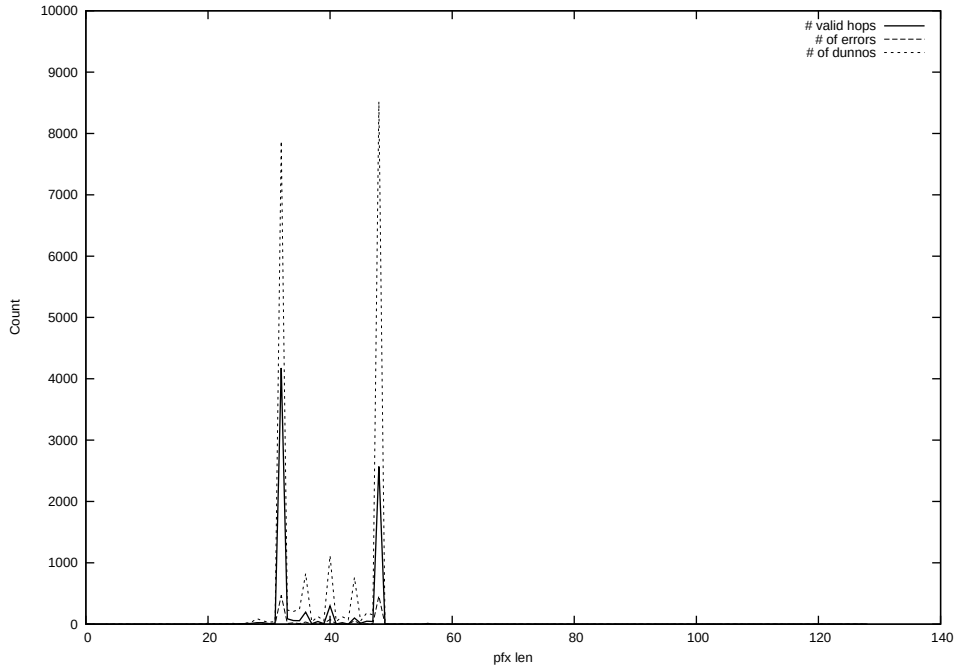
2012-07-24



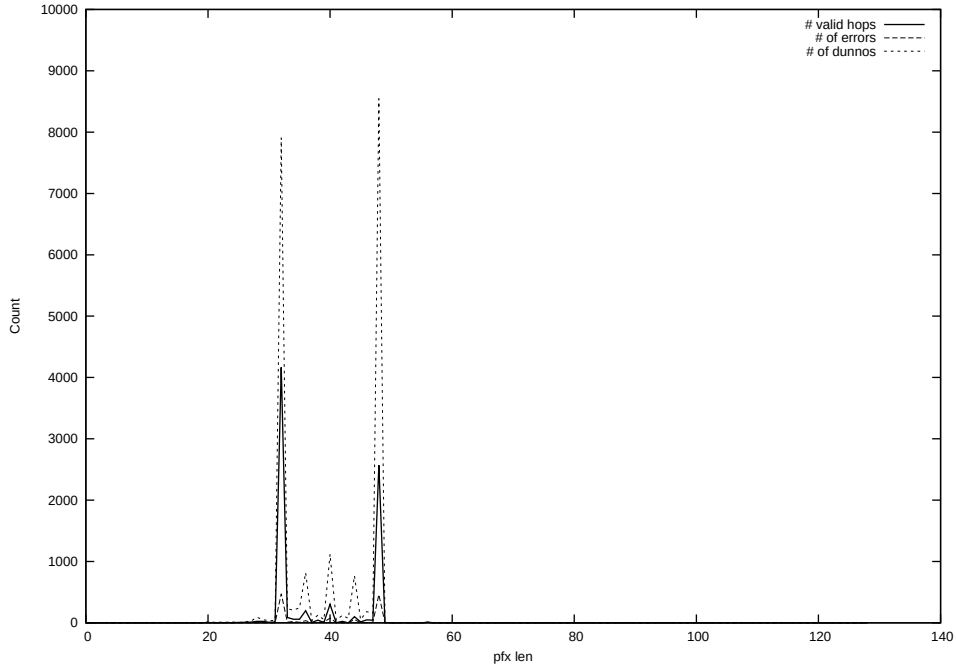
2012-07-25



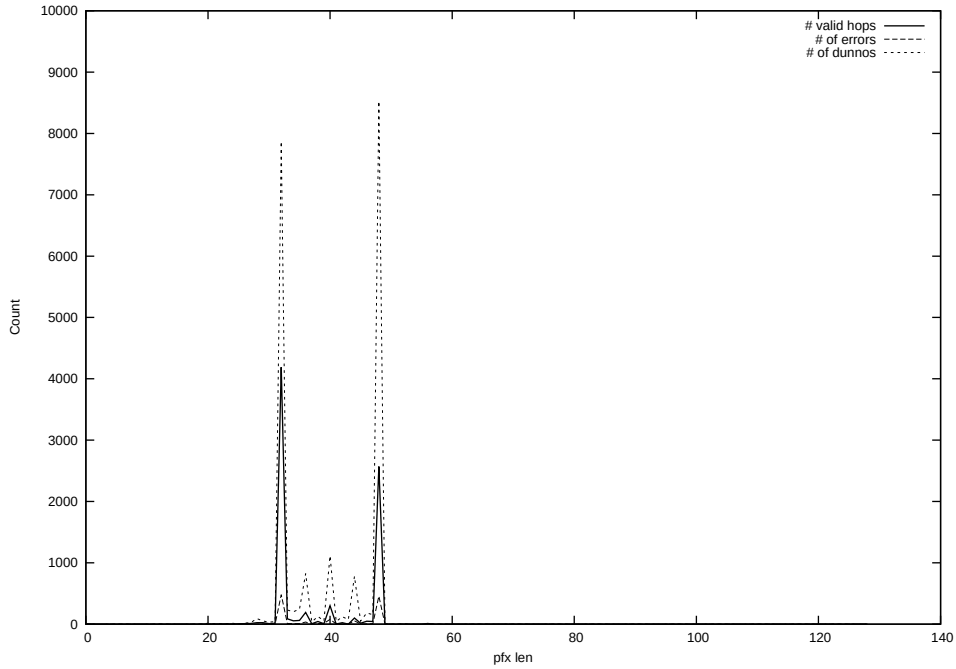
2012-07-26



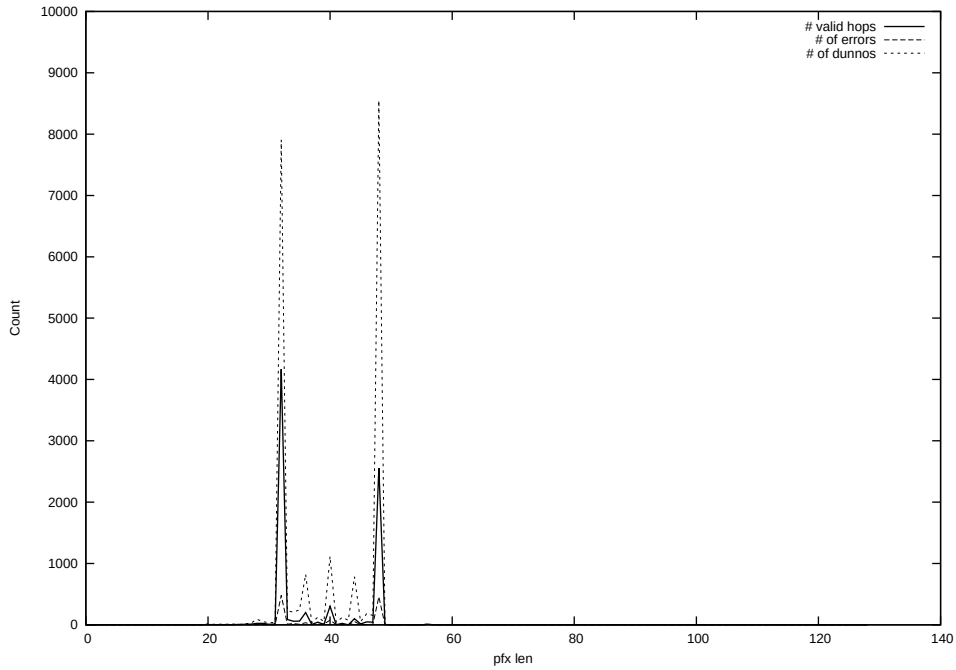
2012-07-27



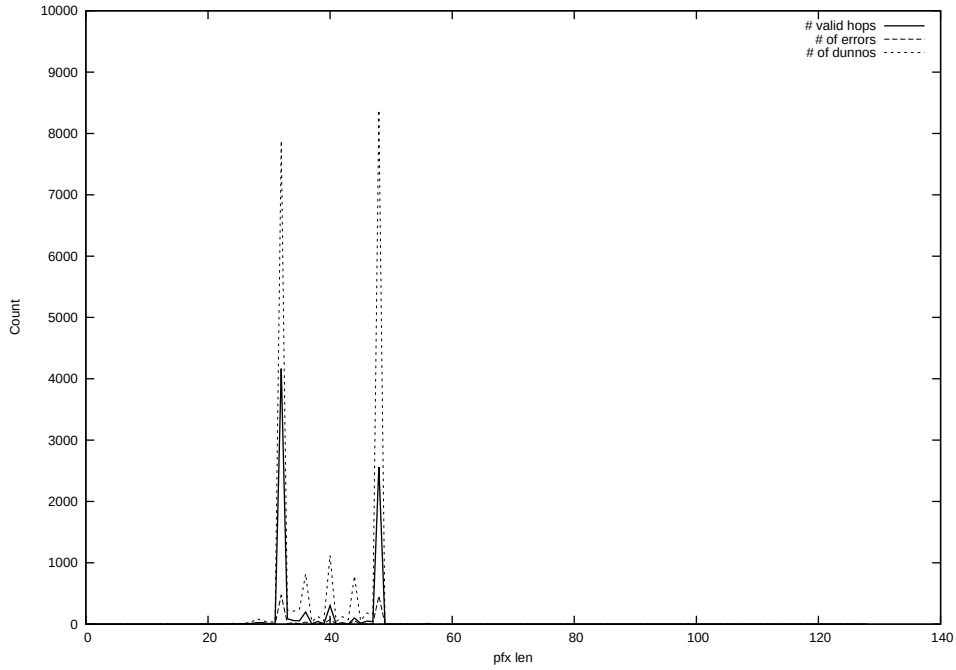
2012-07-28



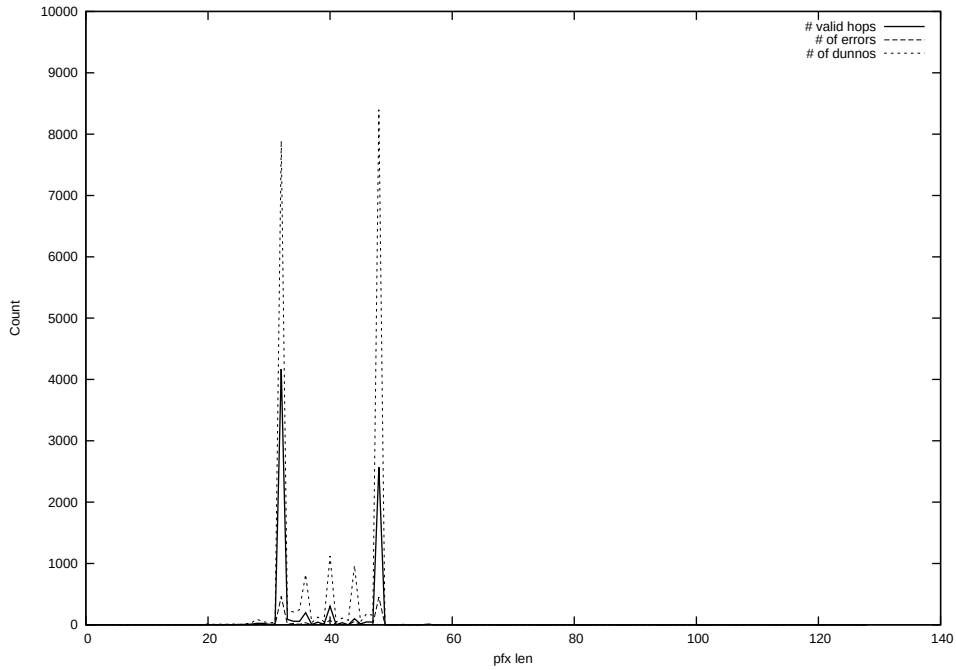
2012-07-29



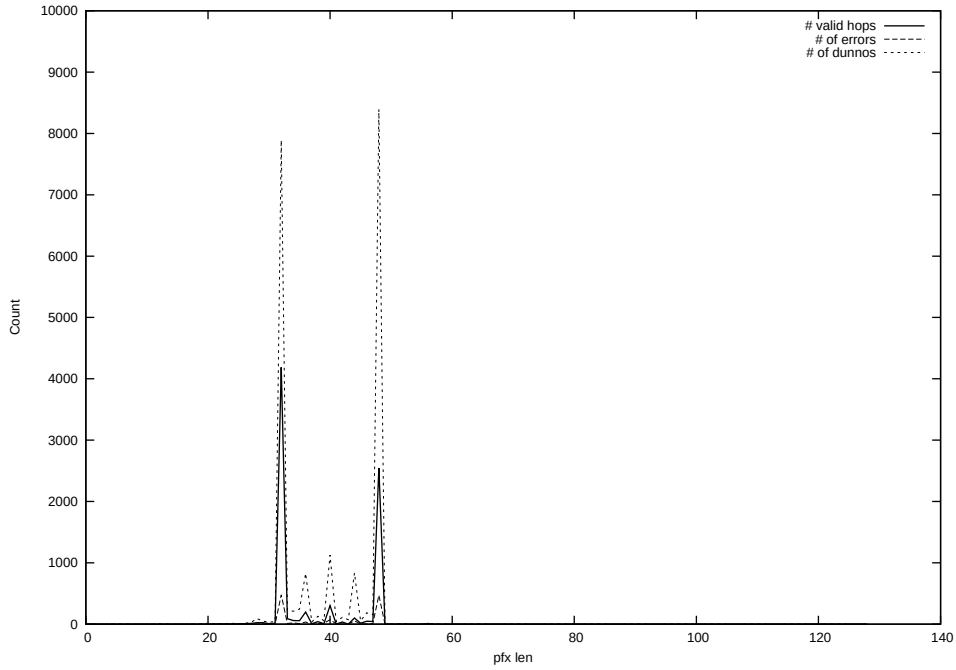
2012-07-30



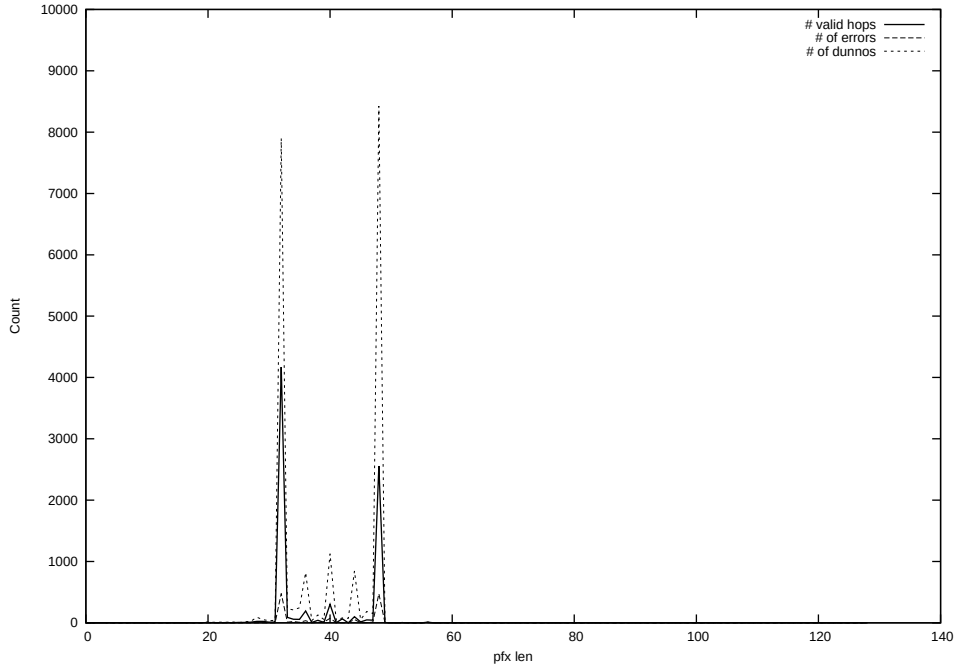
2012-07-31



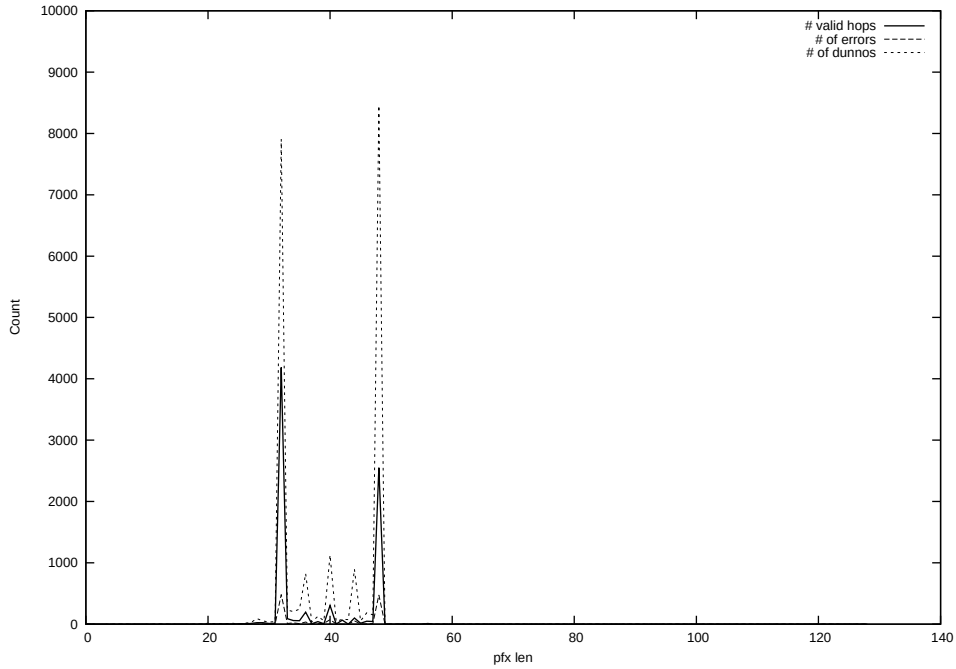
2012-08-01



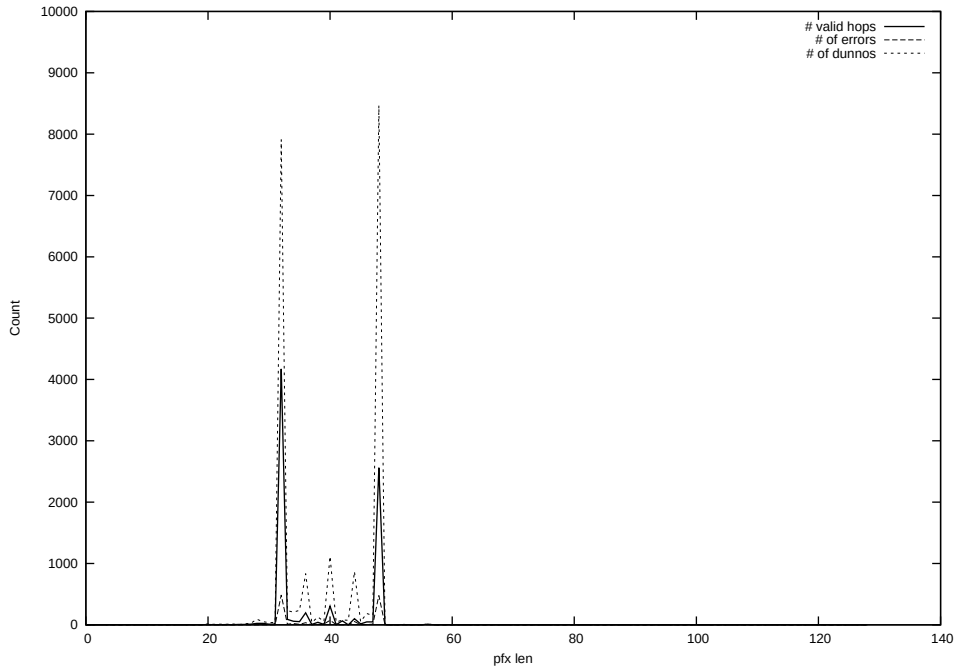
2012-08-02



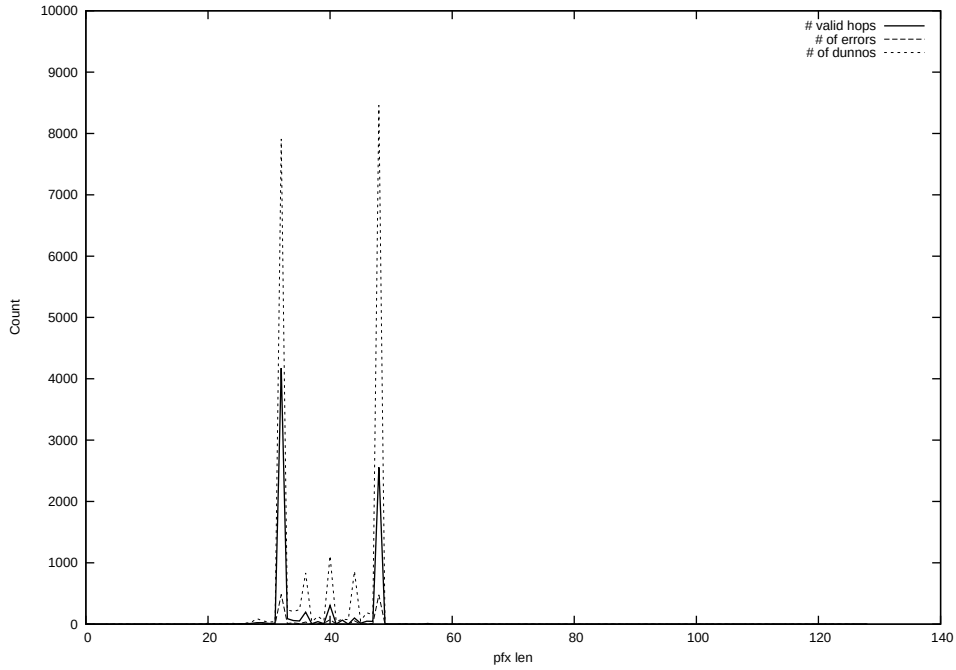
2012-08-03



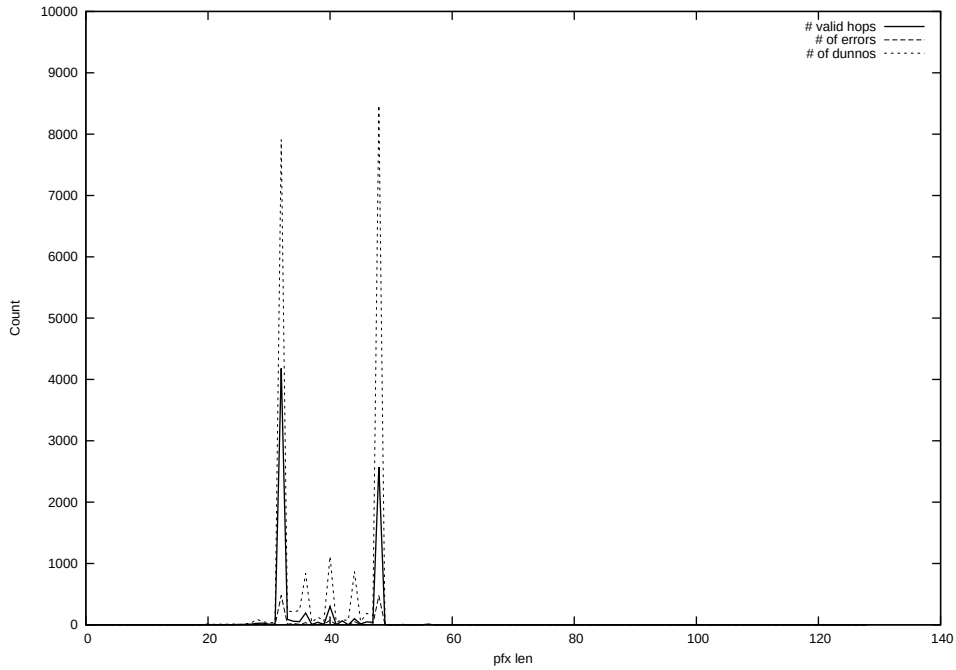
2012-08-04



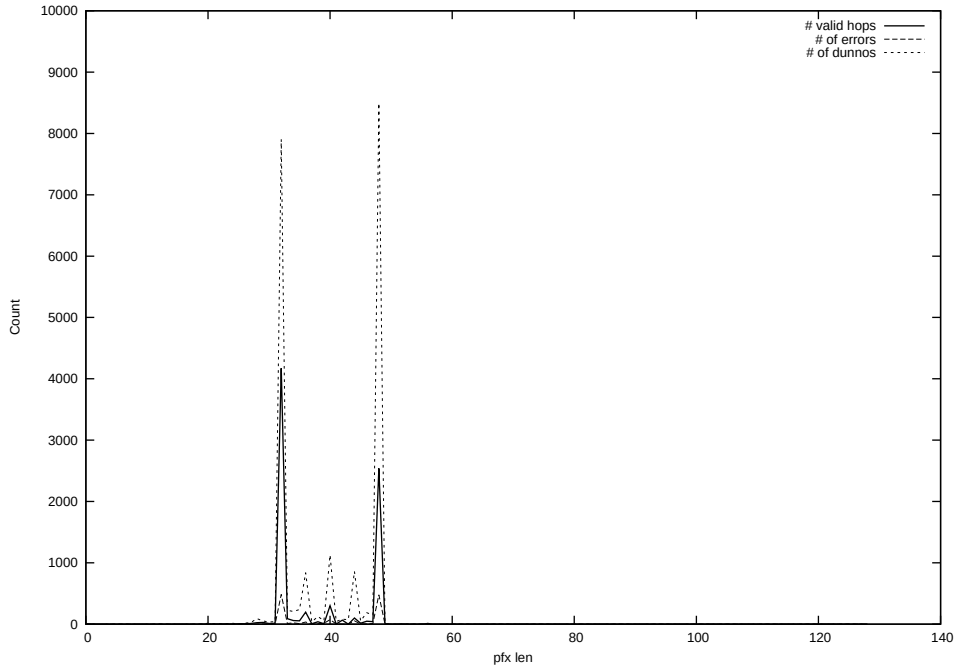
2012-08-05



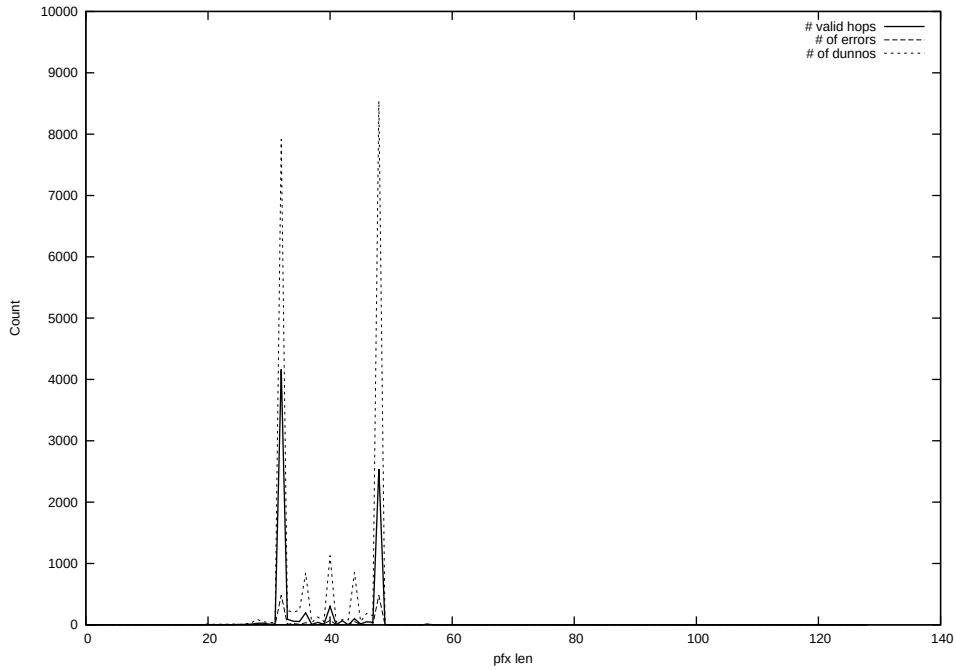
2012-08-06



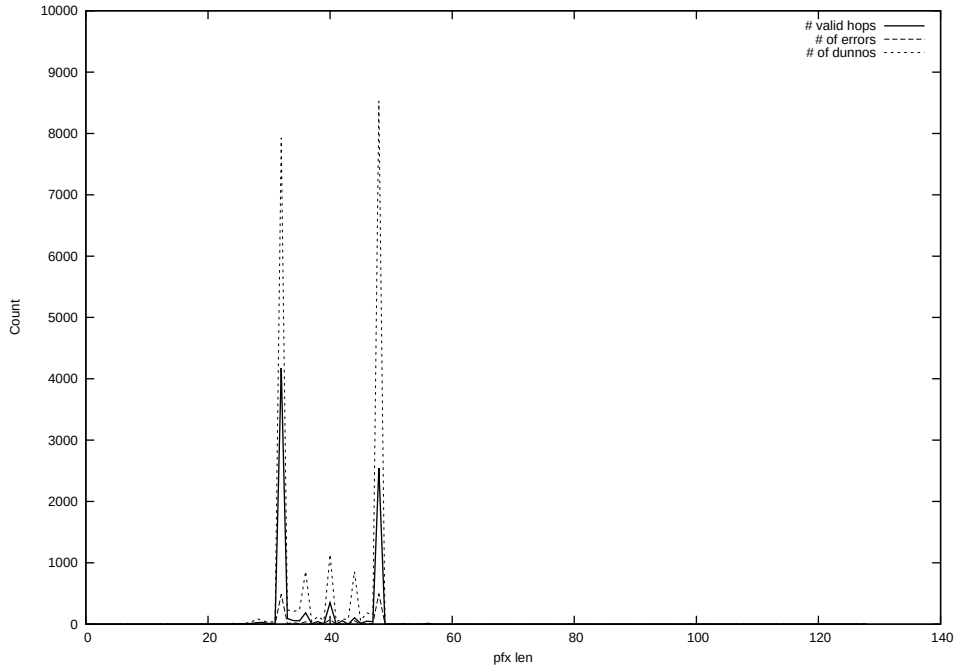
2012-08-07



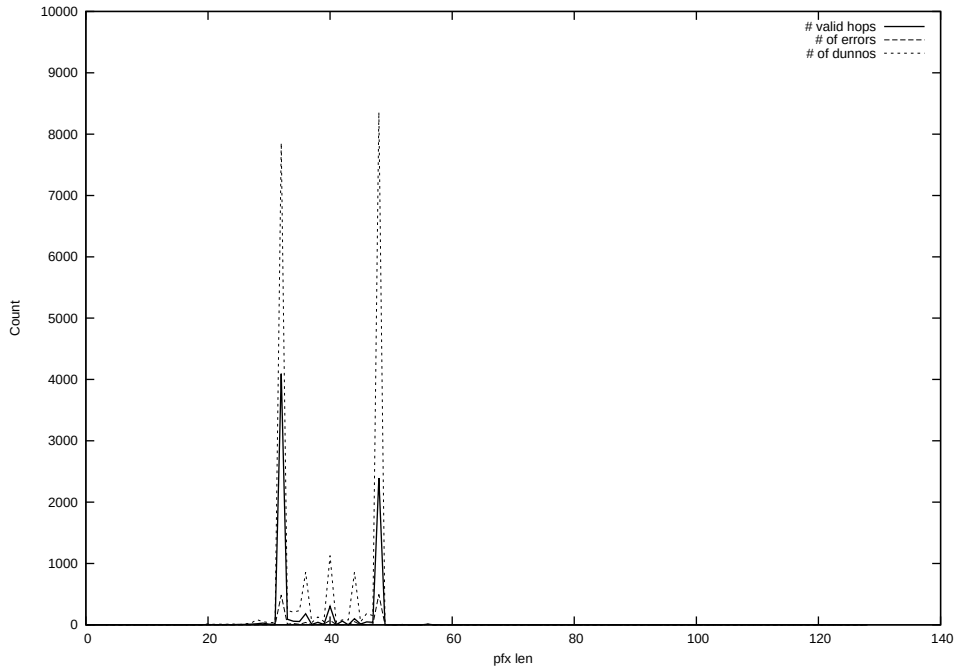
2012-08-08



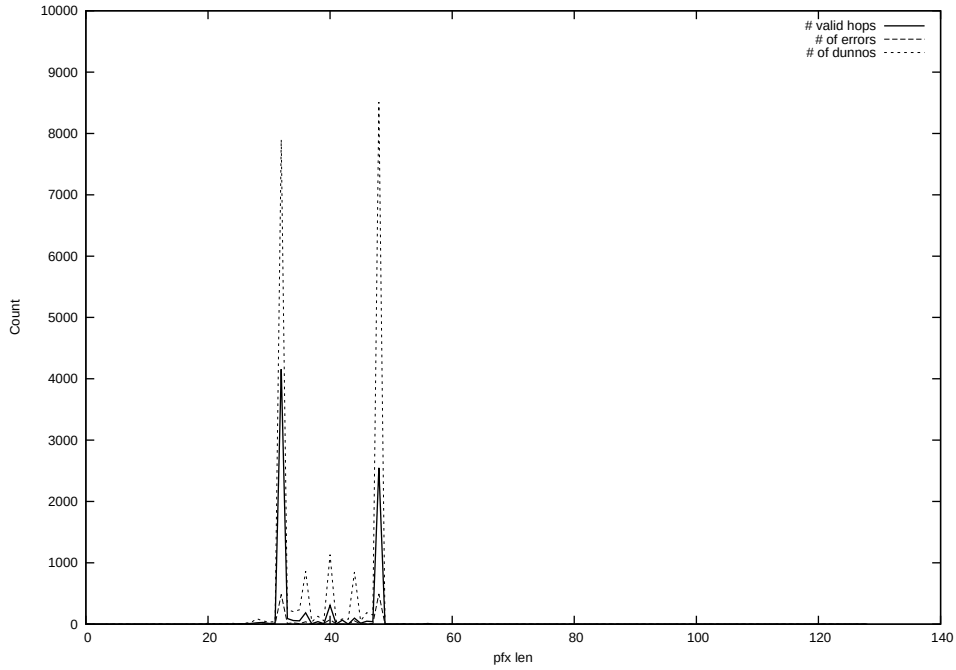
2012-08-09



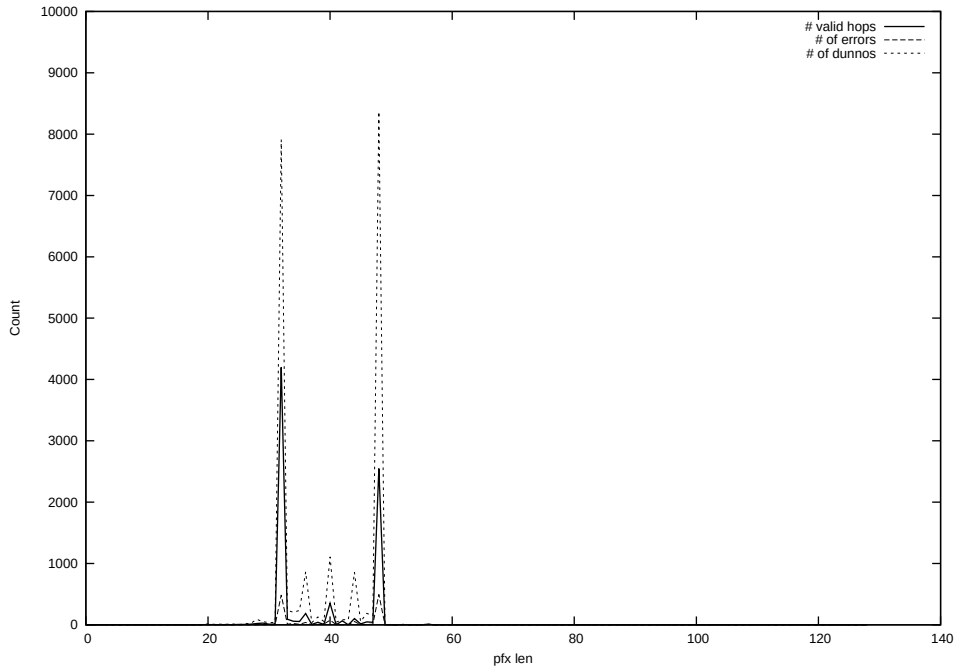
2012-08-10



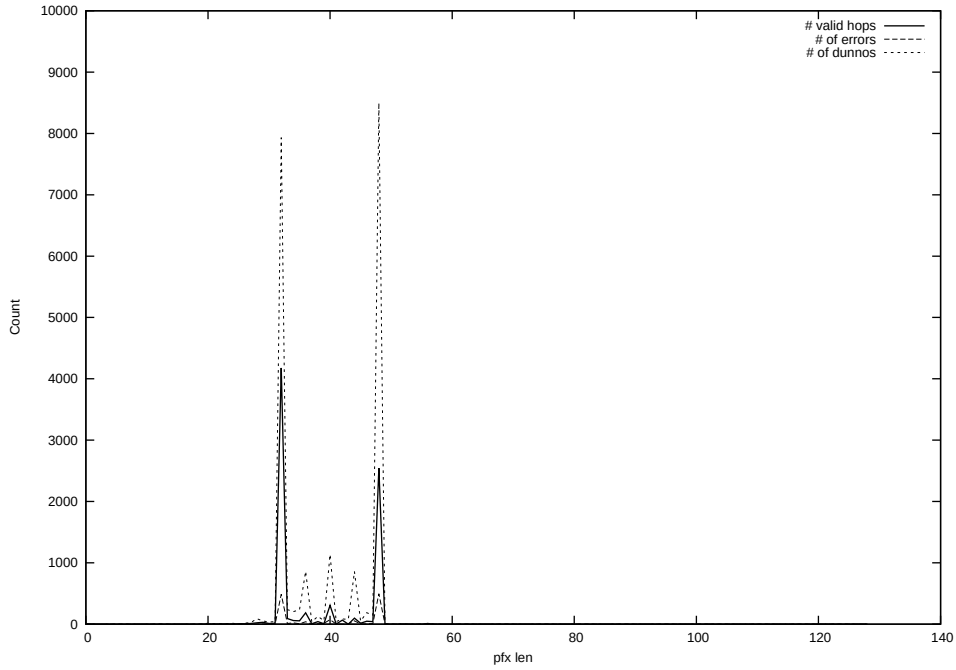
2012-08-11



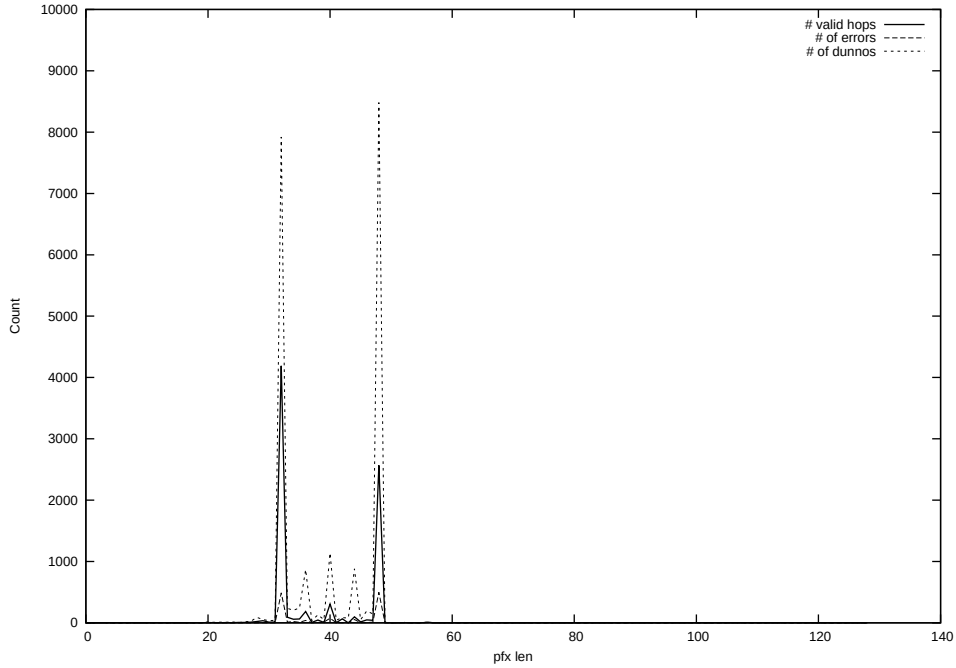
2012-08-12



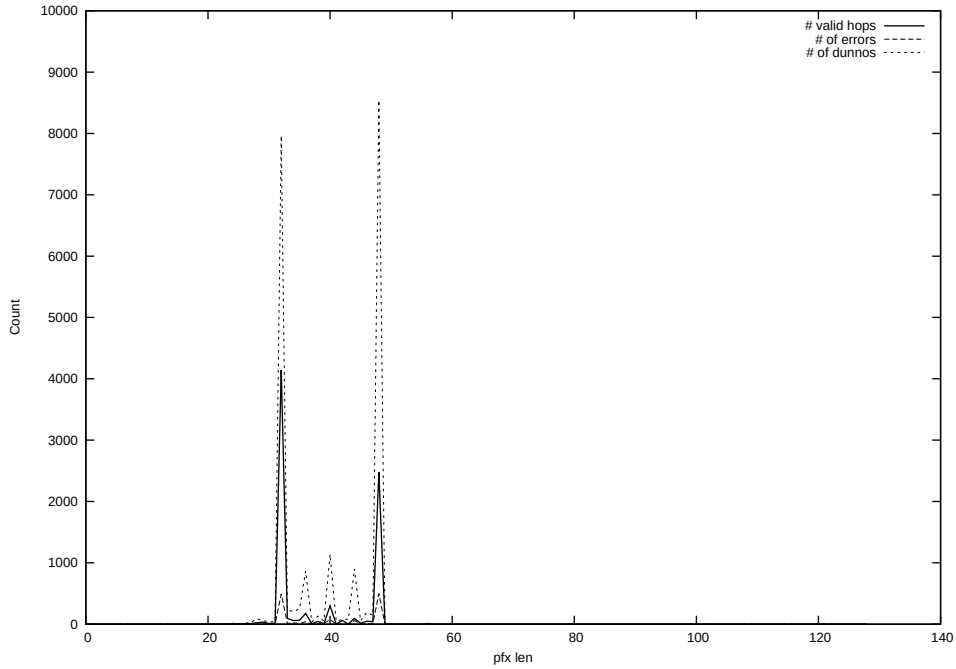
2012-08-13



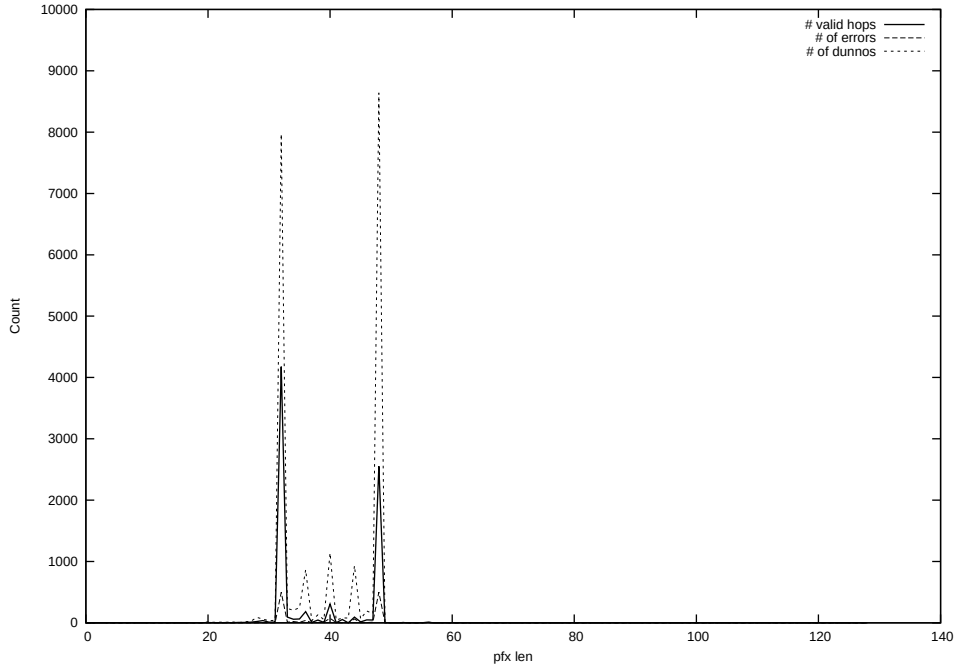
2012-08-14



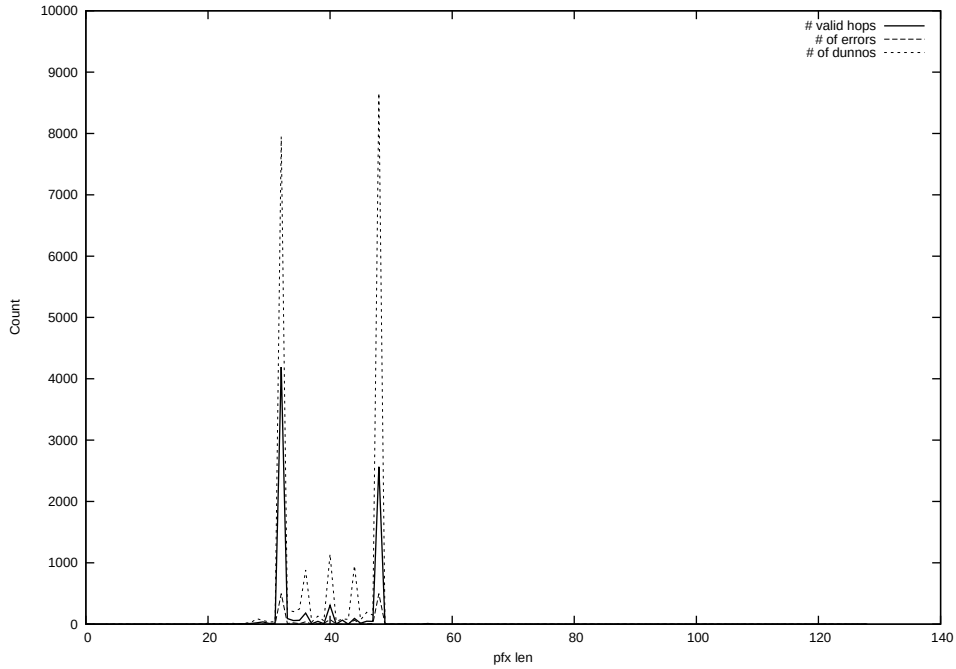
2012-08-15



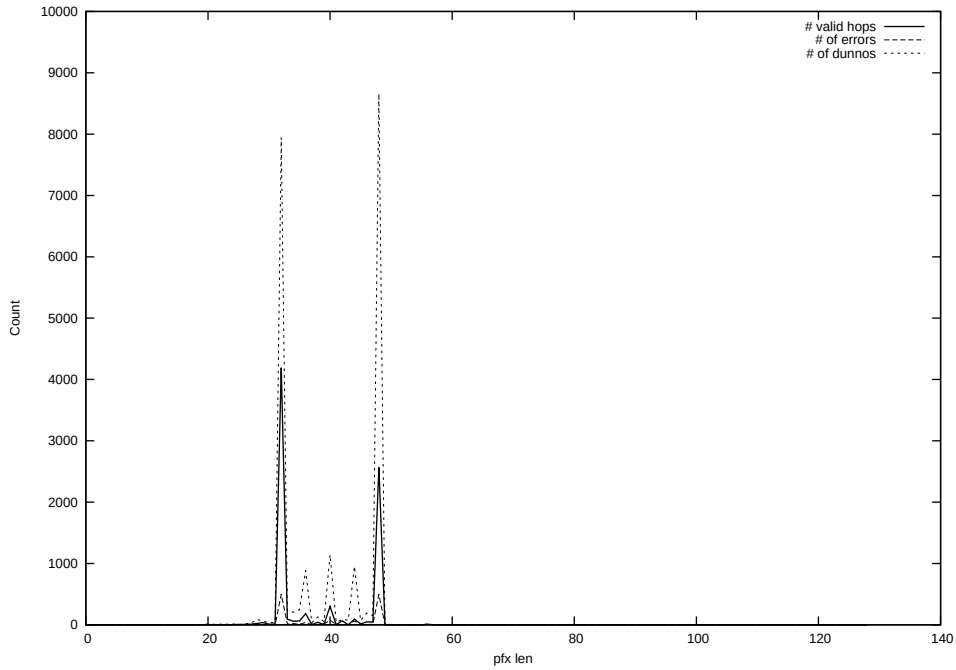
2012-08-16



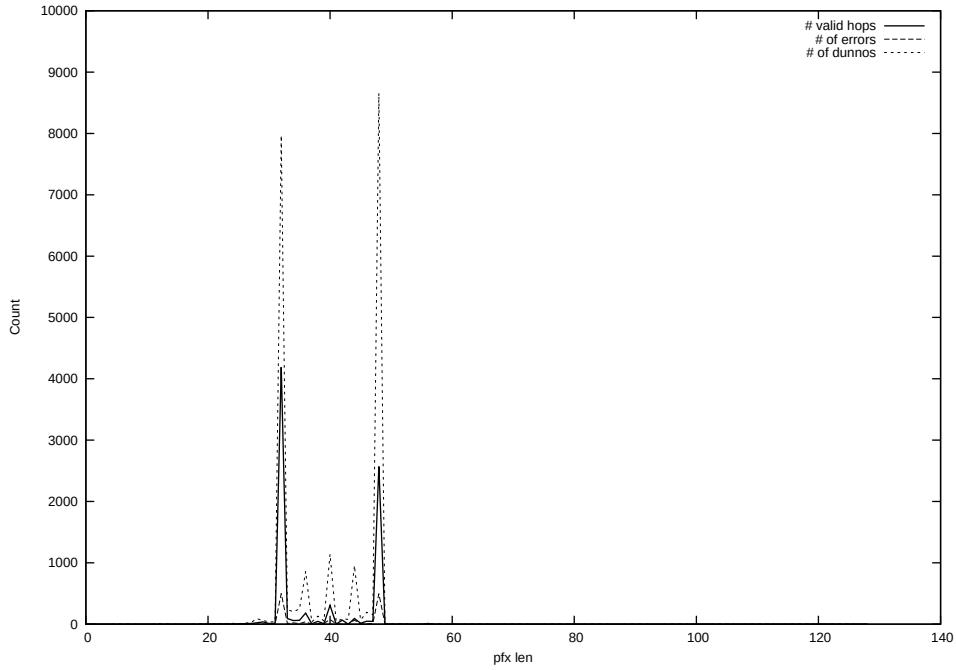
2012-08-17



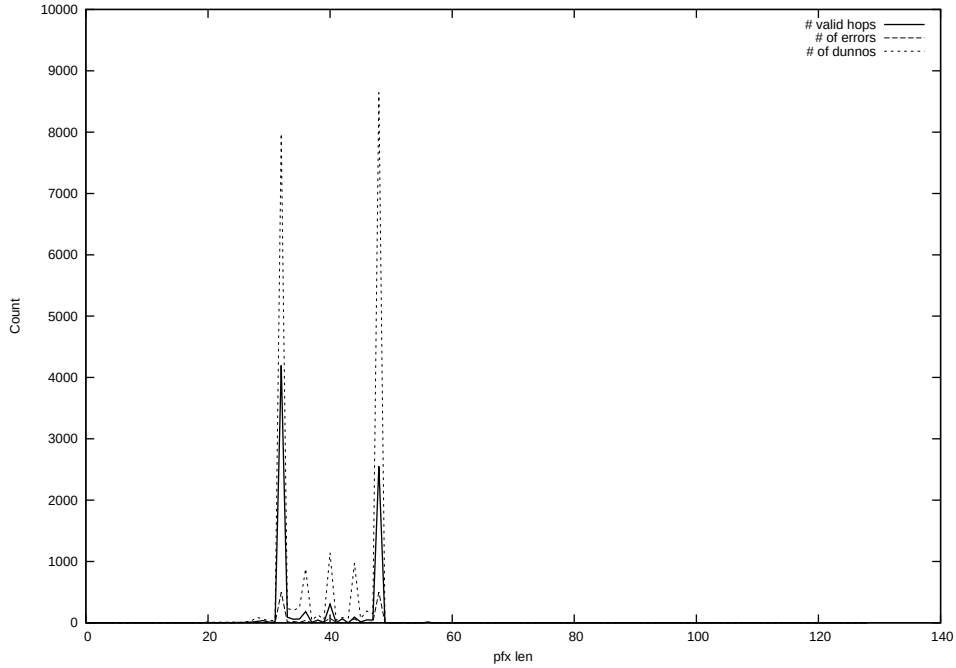
2012-08-18



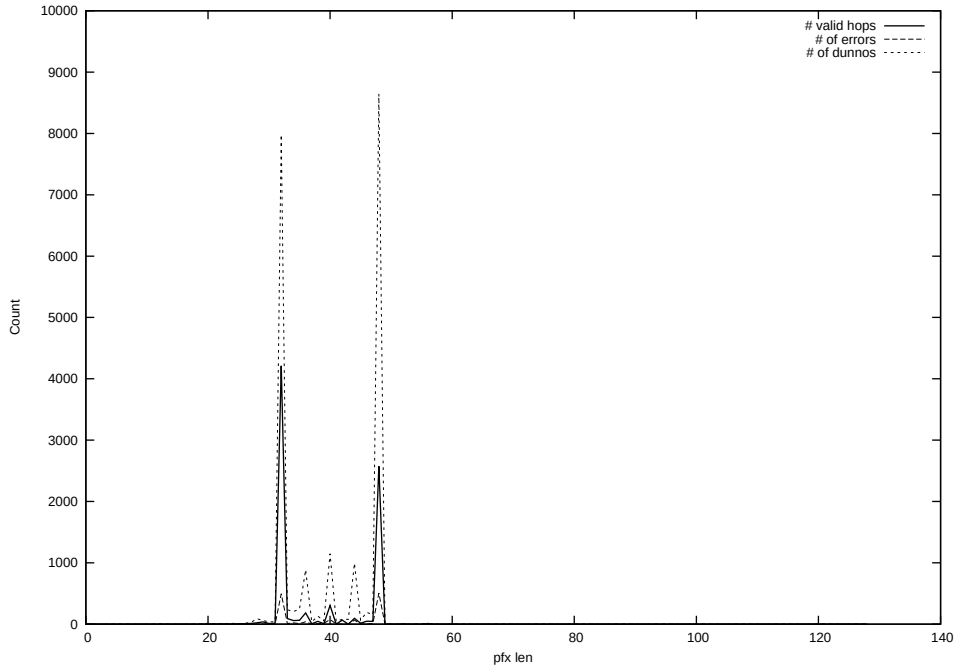
2012-08-19



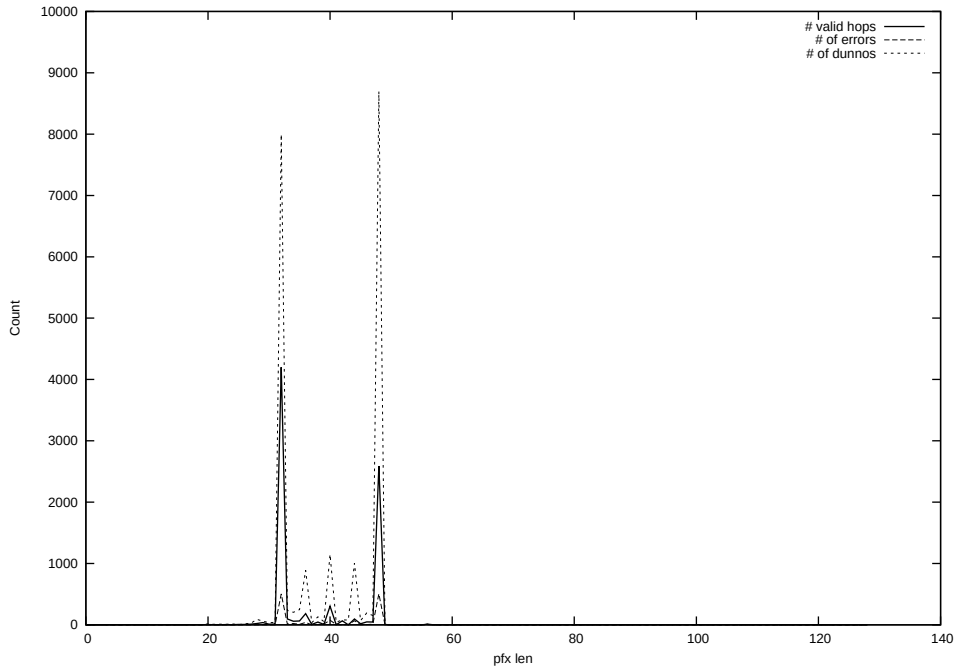
2012-08-20



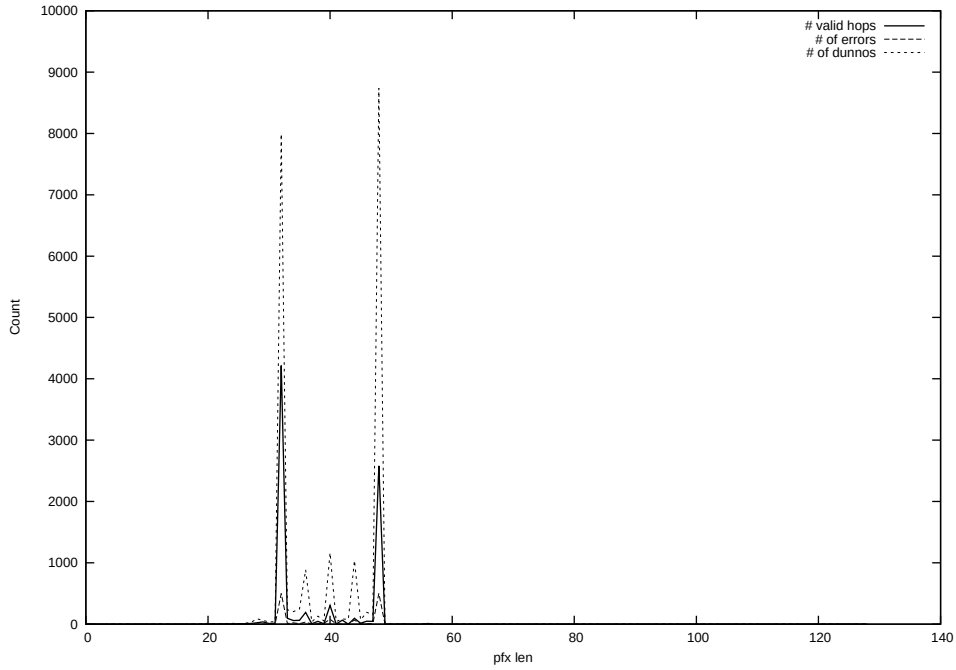
2012-08-21



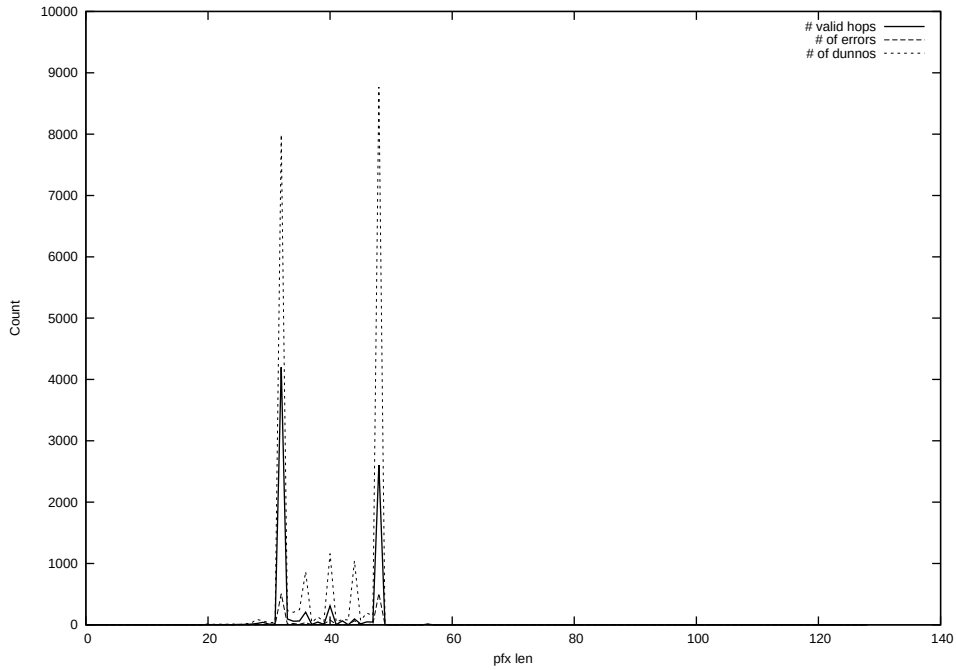
2012-08-22



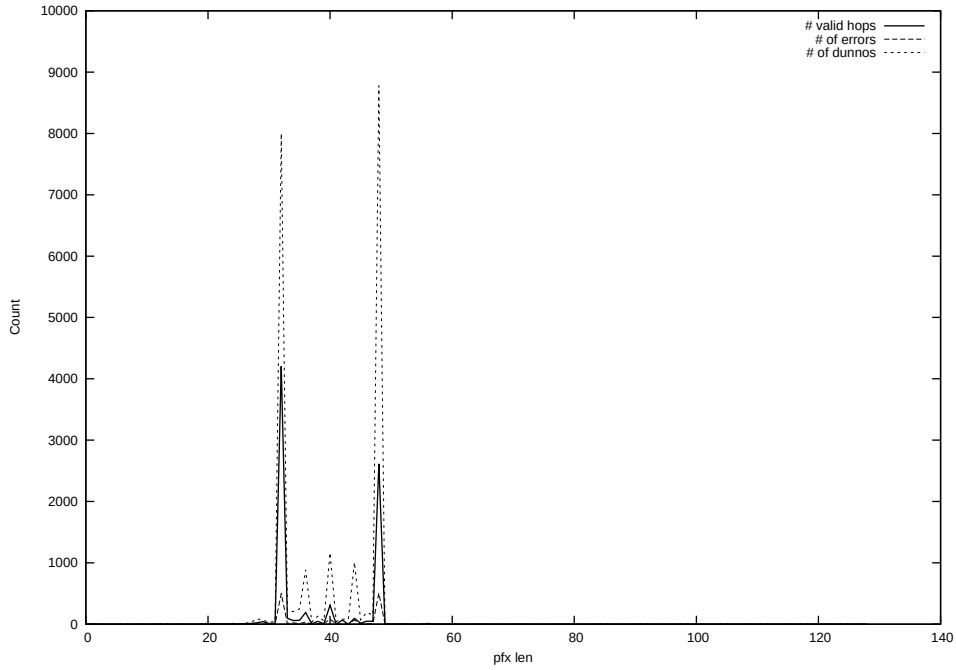
2012-08-23



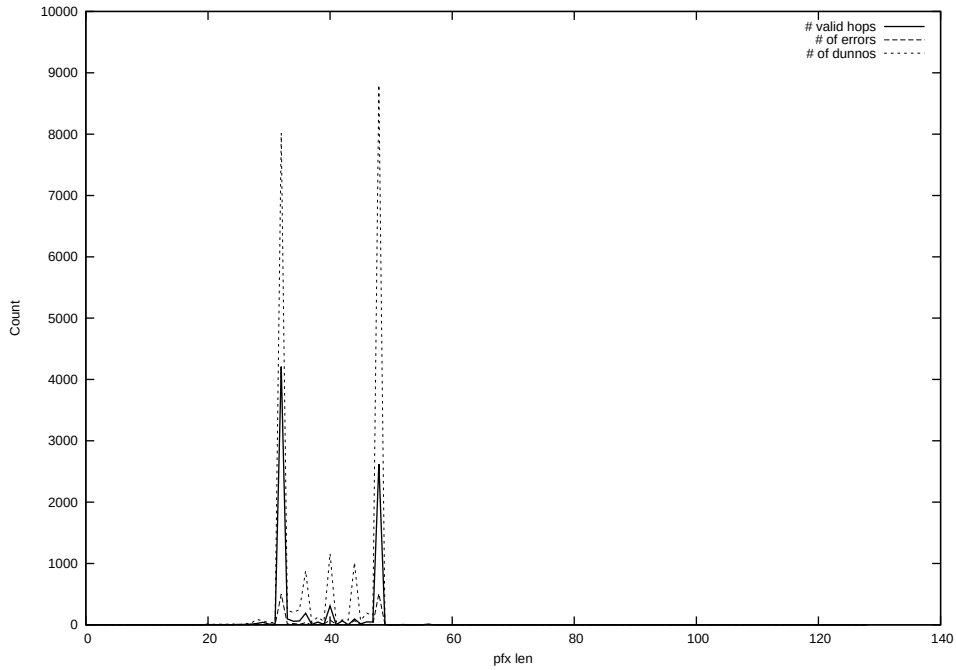
2012-08-24



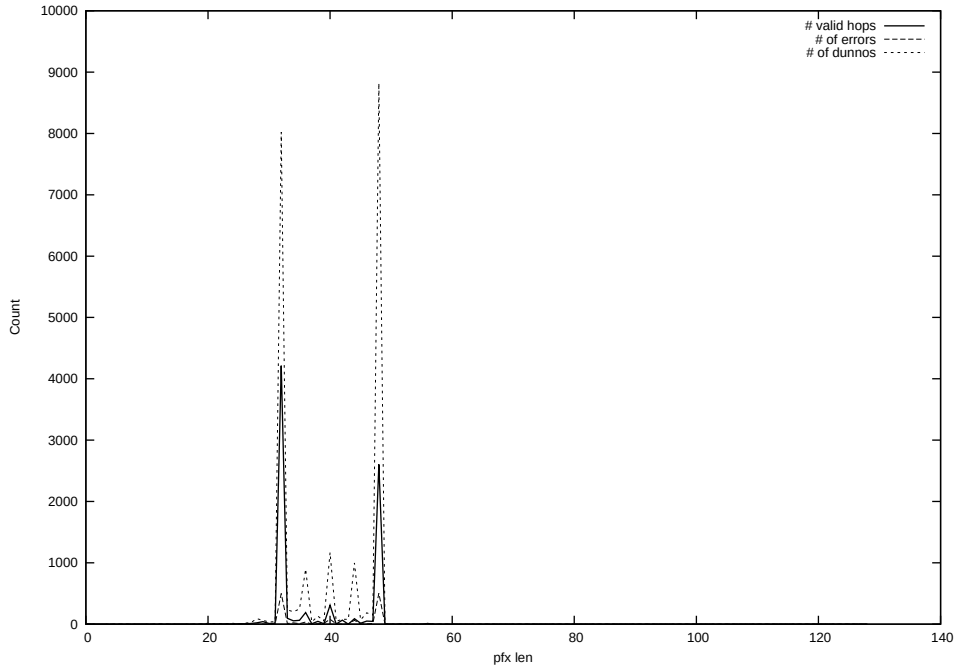
2012-08-25



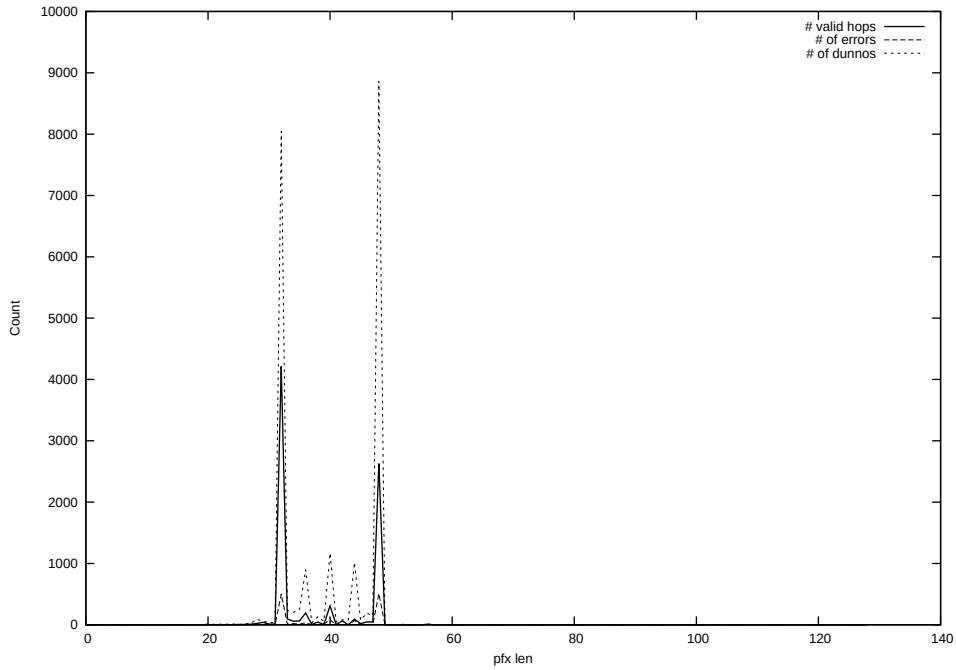
2012-08-26



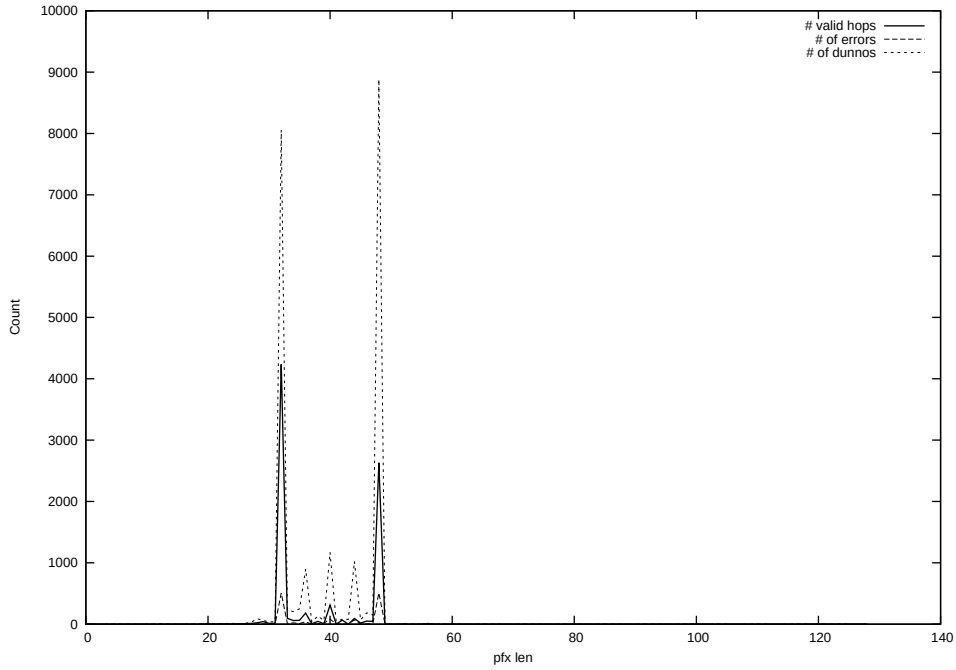
2012-08-27



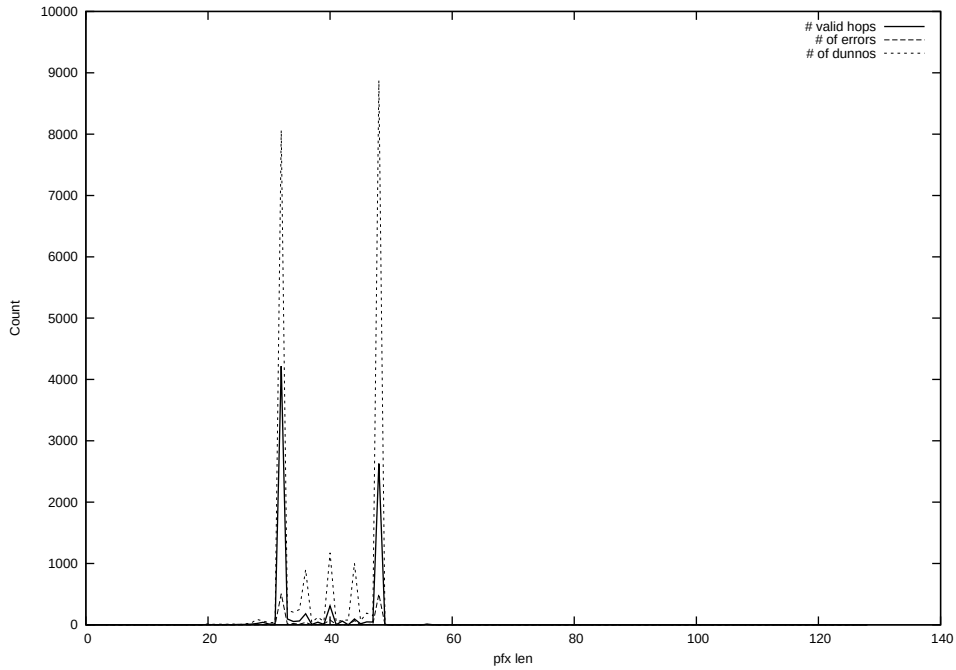
2012-08-28



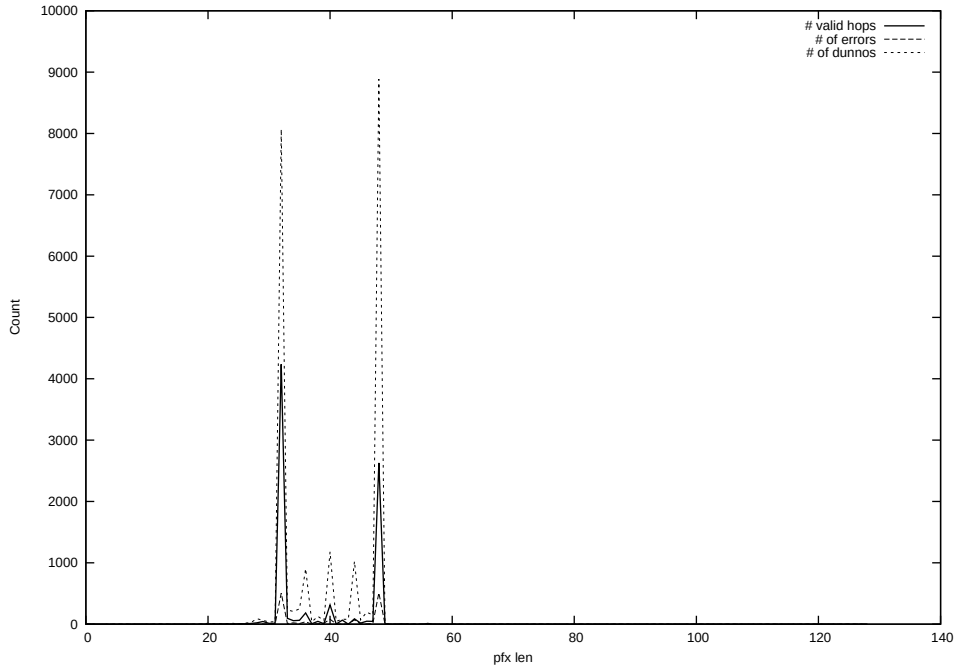
2012-08-29



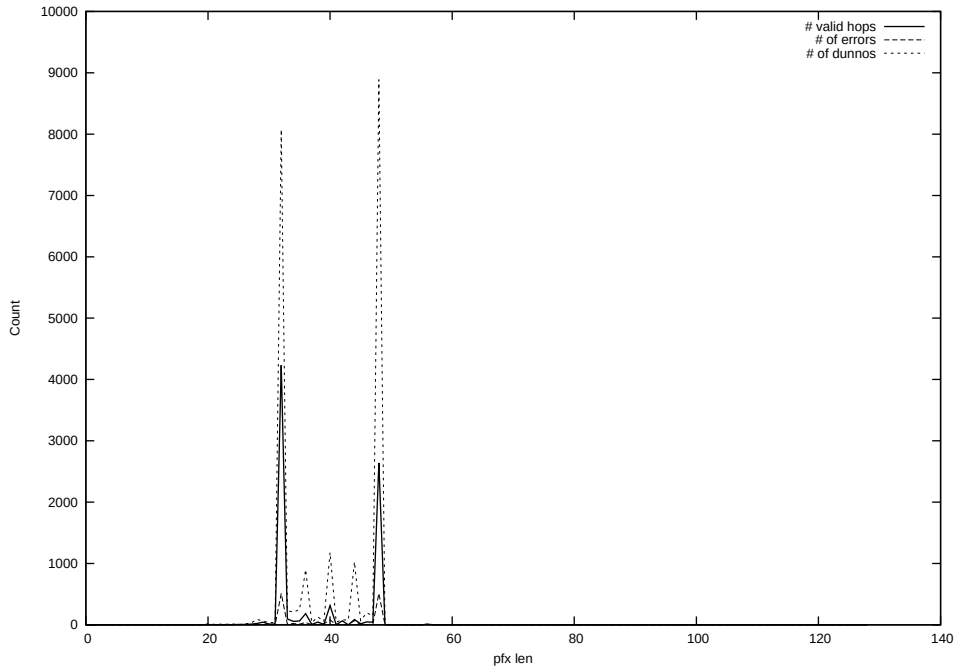
2012-08-30



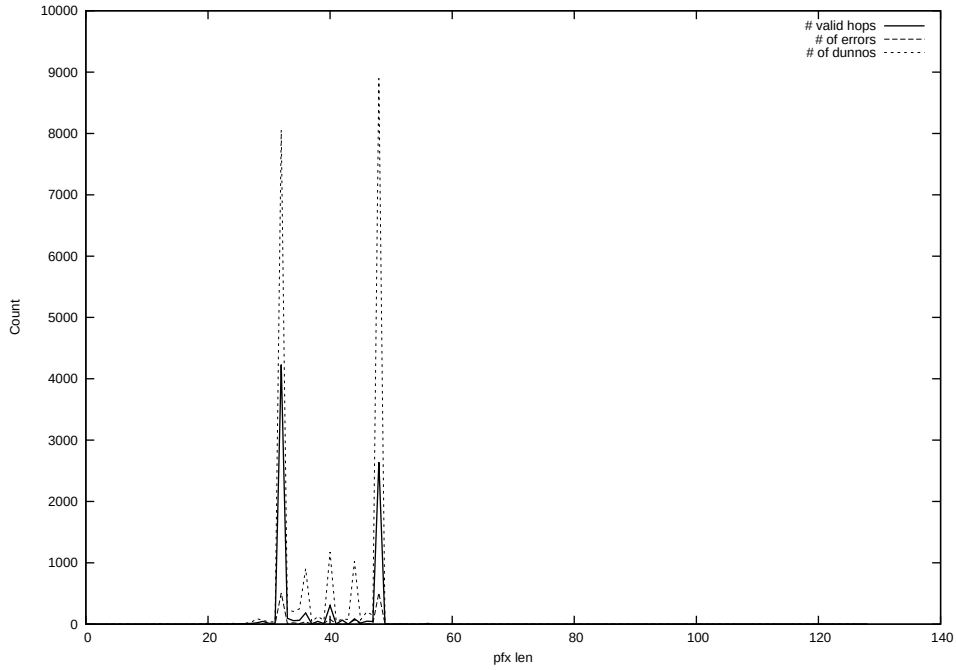
2012-08-31



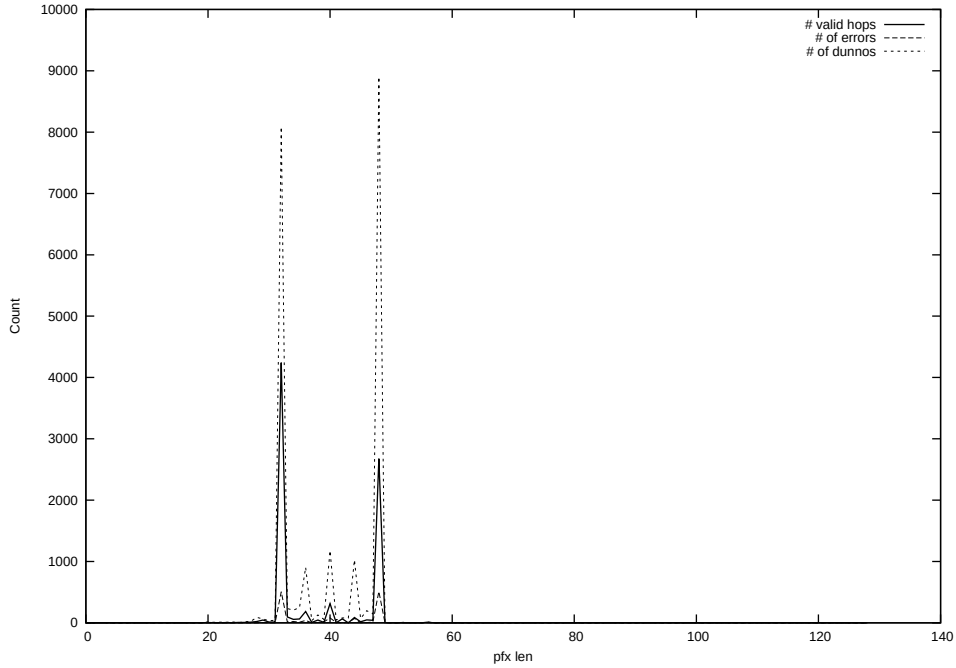
2012-09-01



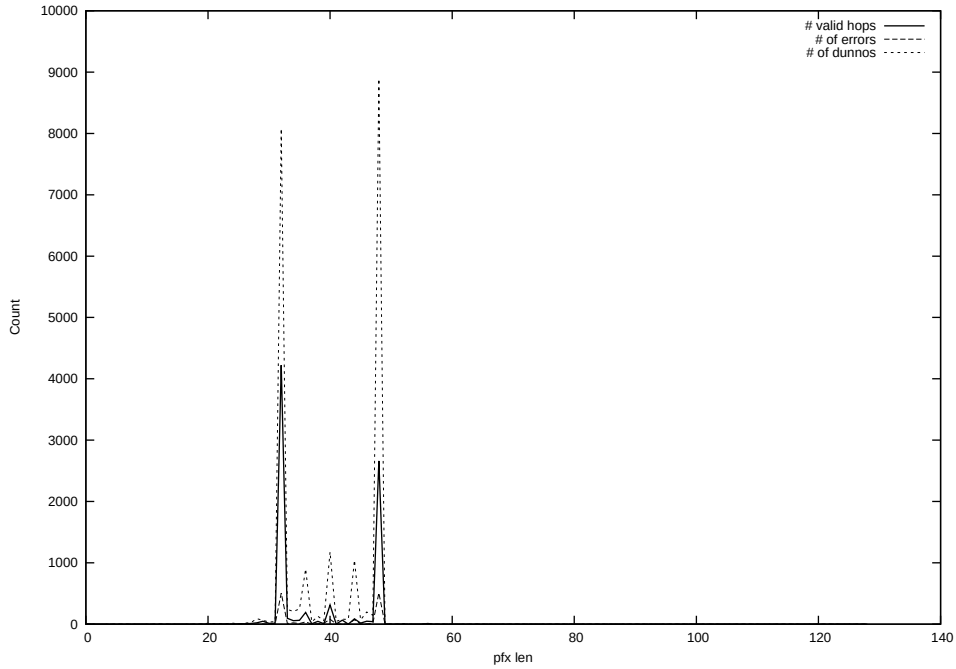
2012-09-02



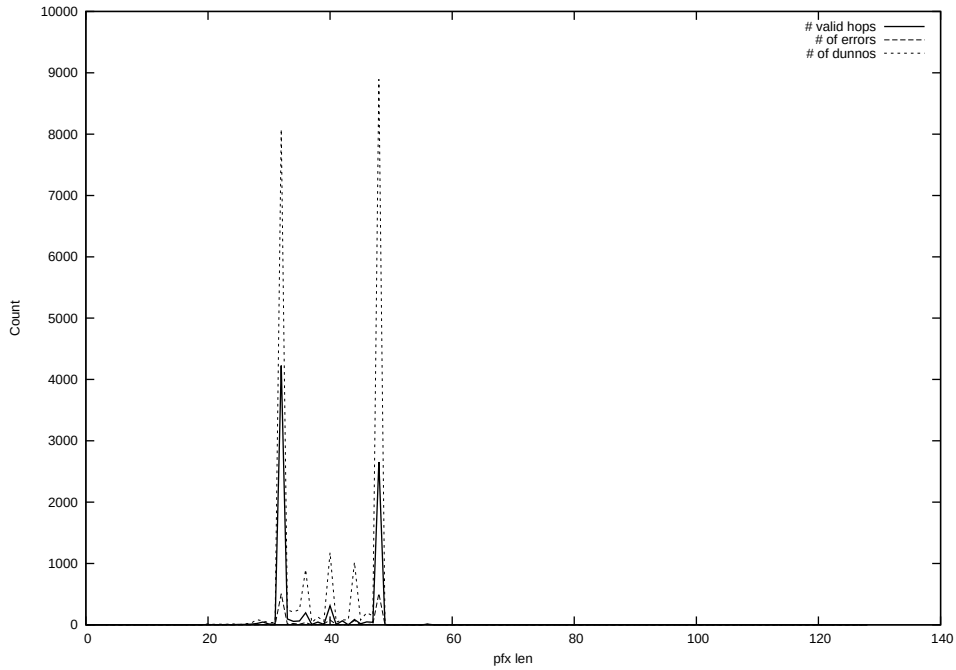
2012-09-03



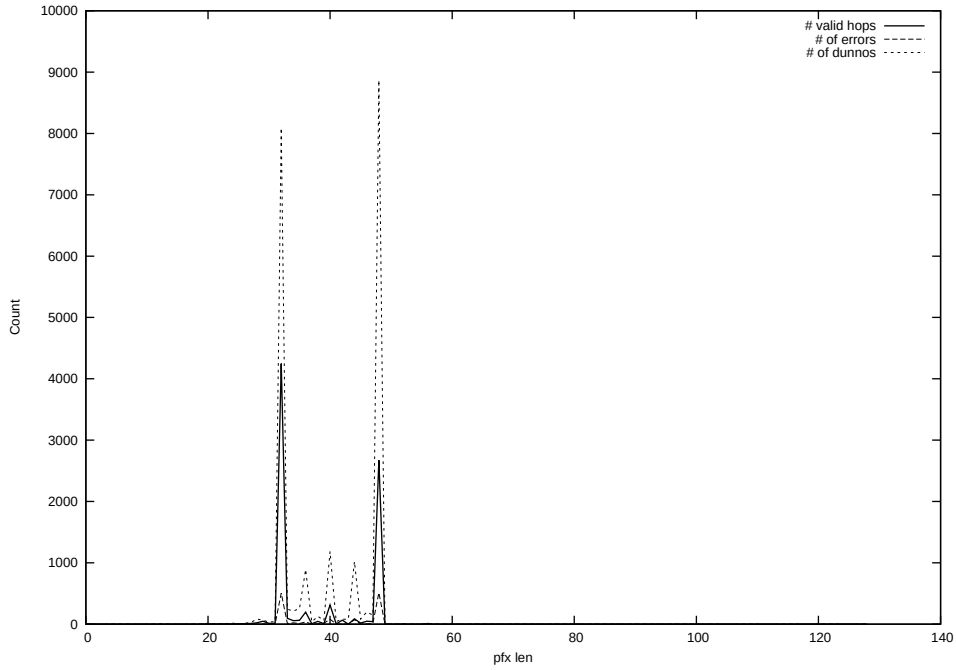
2012-09-04



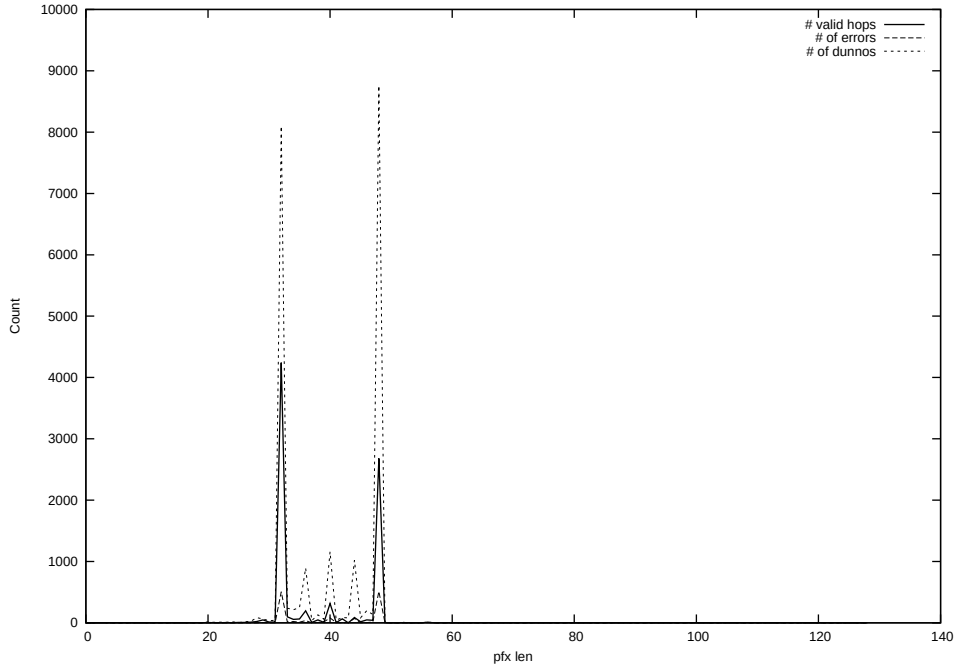
2012-09-05



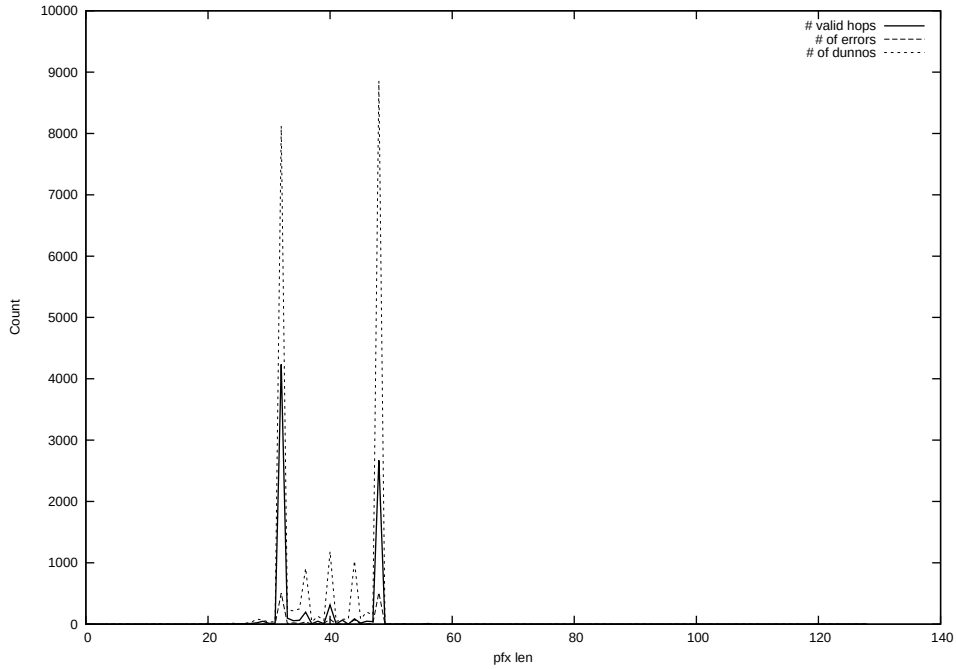
2012-09-06



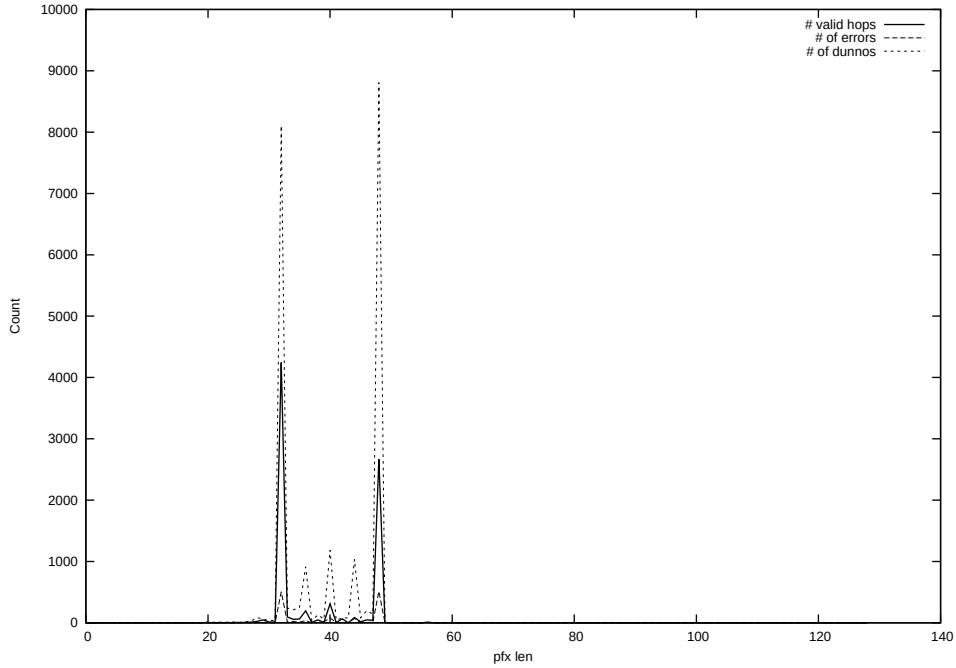
2012-09-07



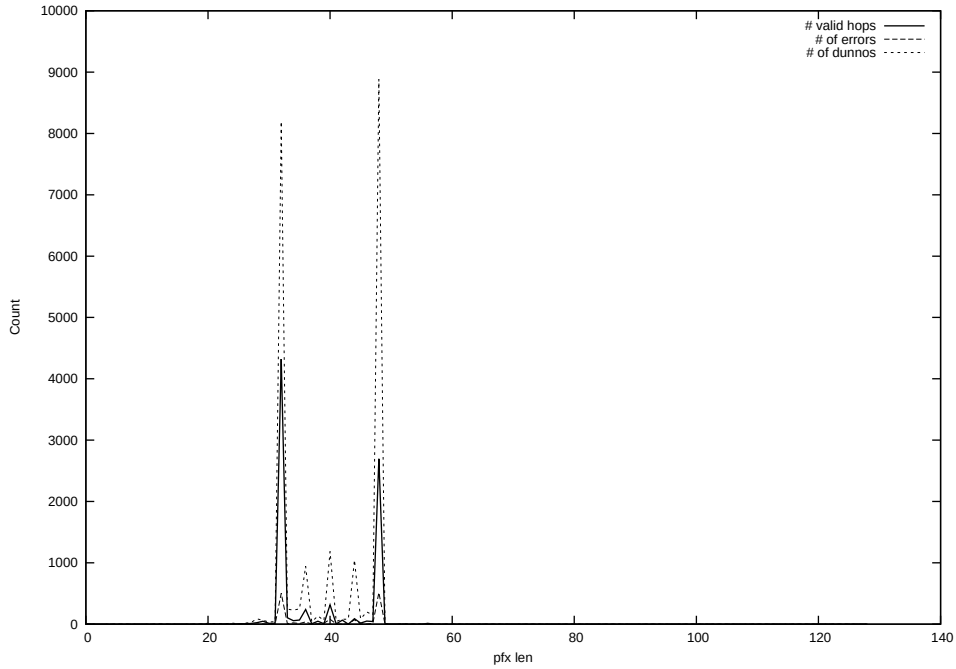
2012-09-08



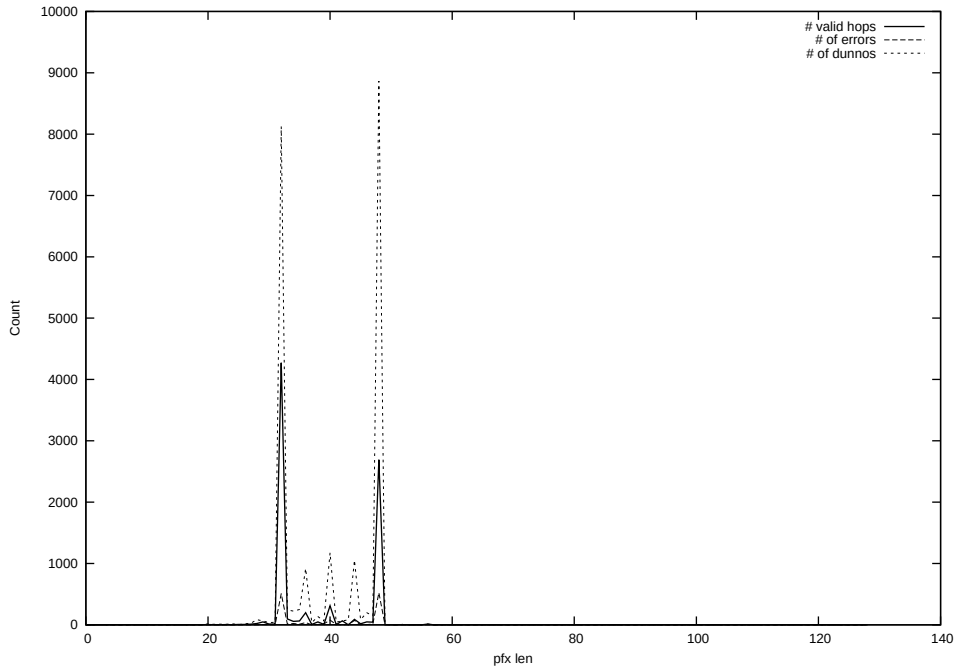
2012-09-09



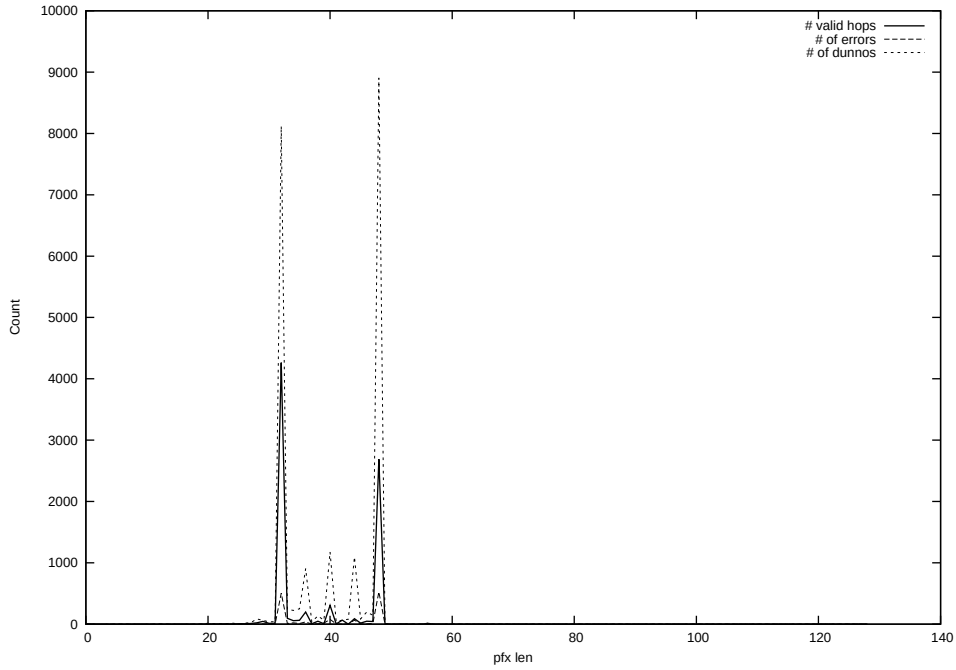
2012-09-10



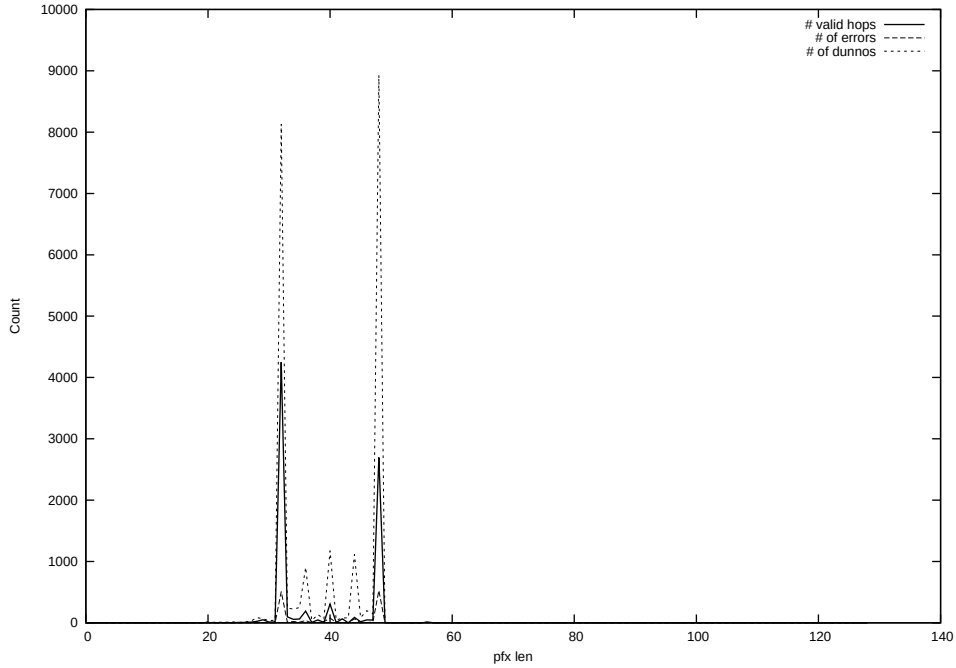
2012-09-11



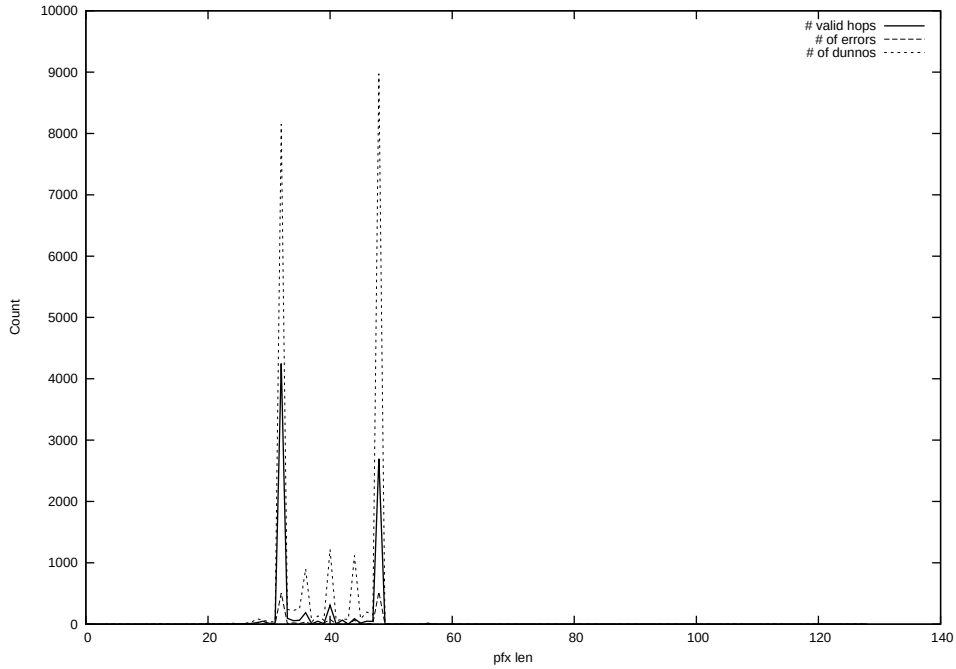
2012-09-12



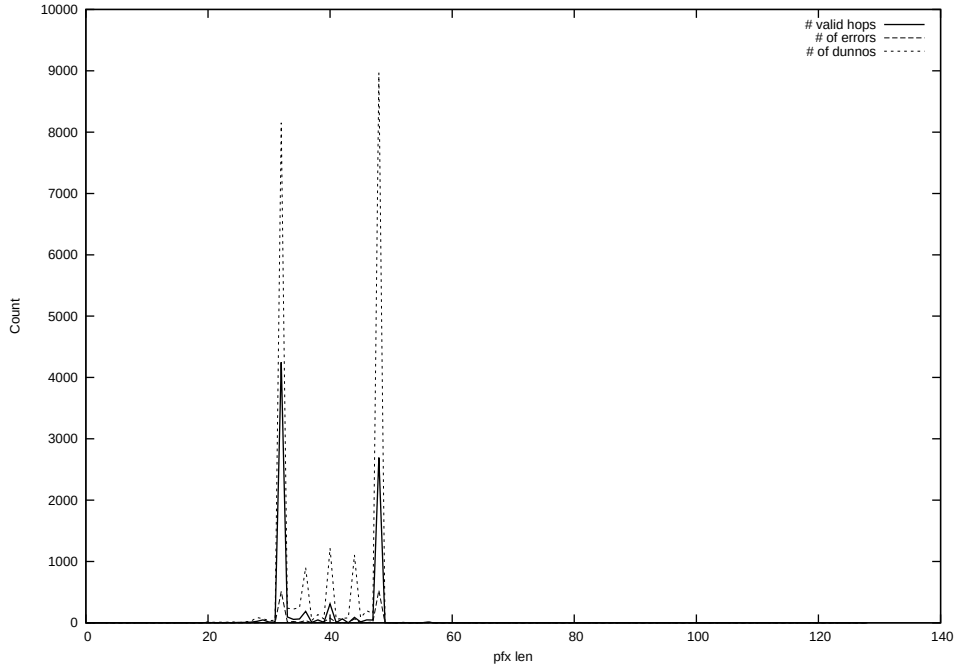
2012-09-13



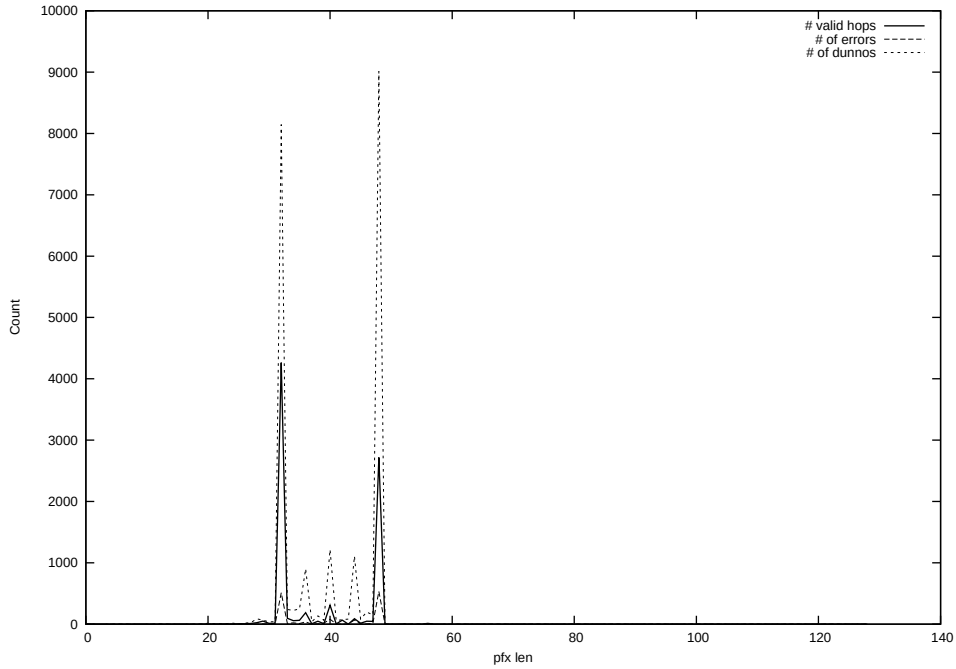
2012-09-14



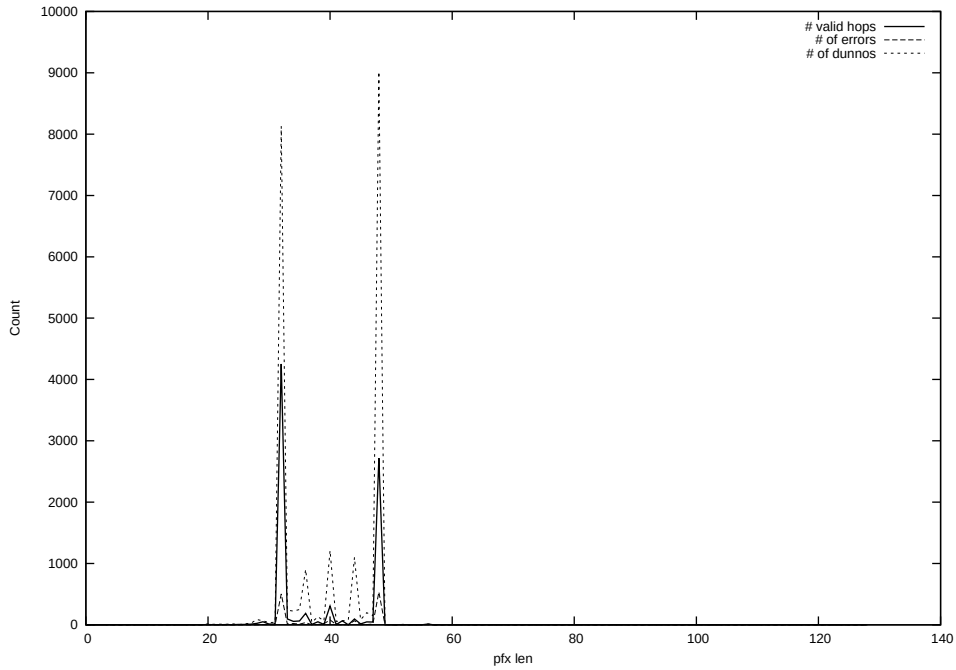
2012-09-15



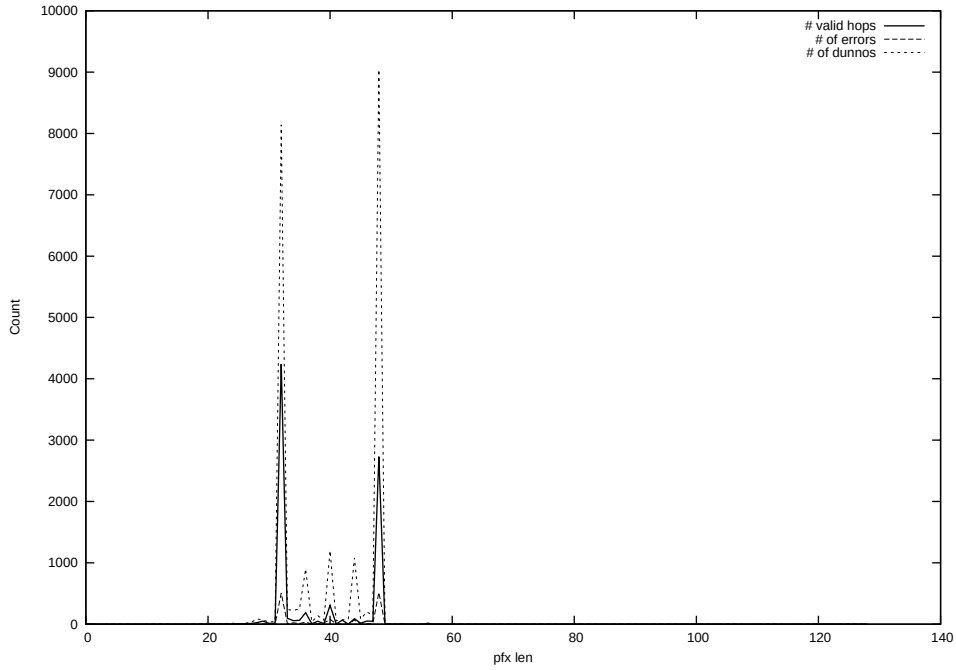
2012-09-16



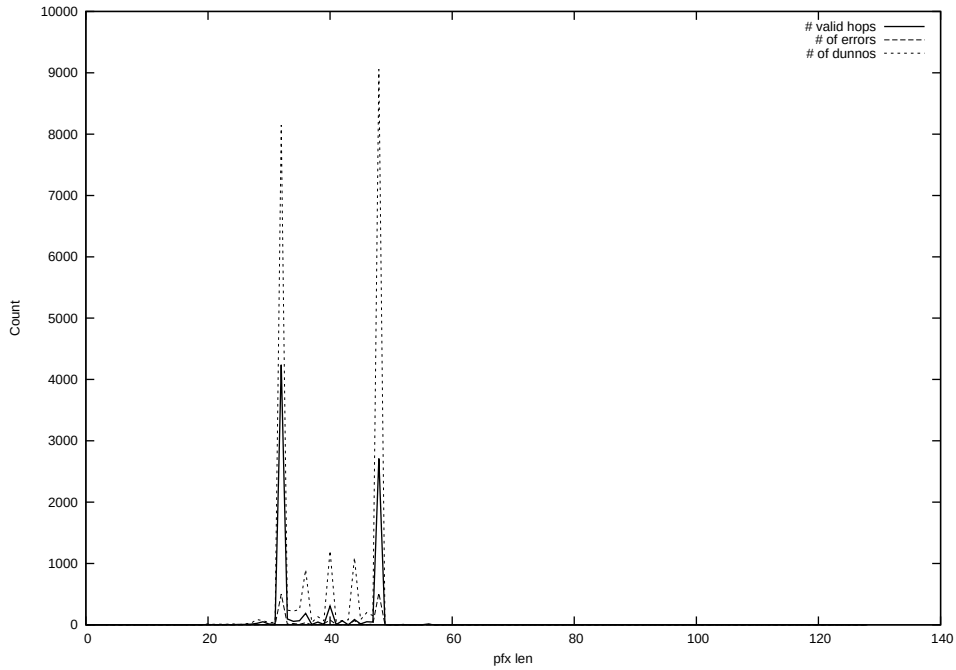
2012-09-17



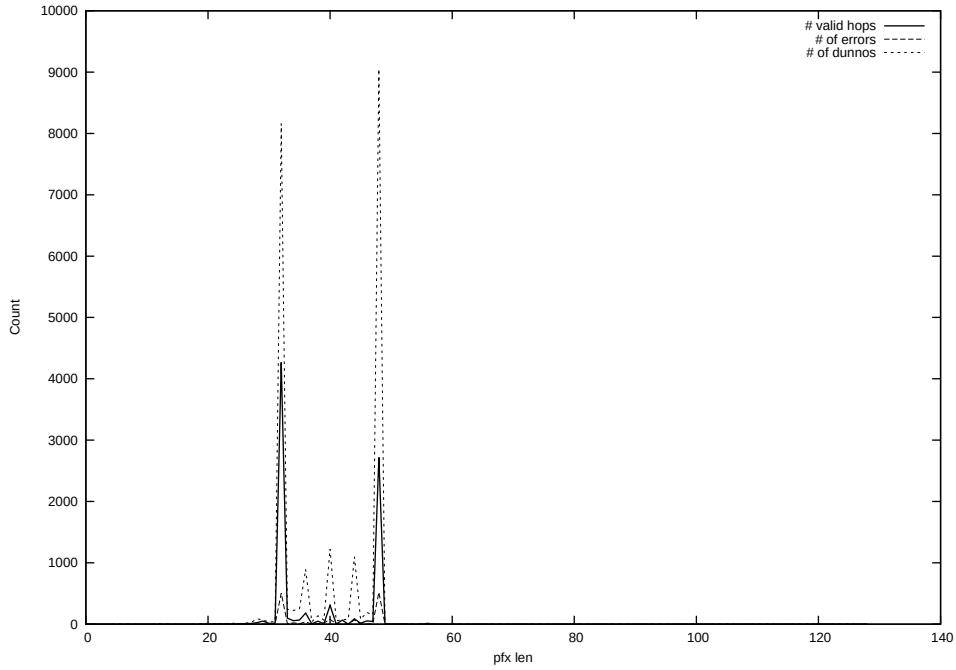
2012-09-18



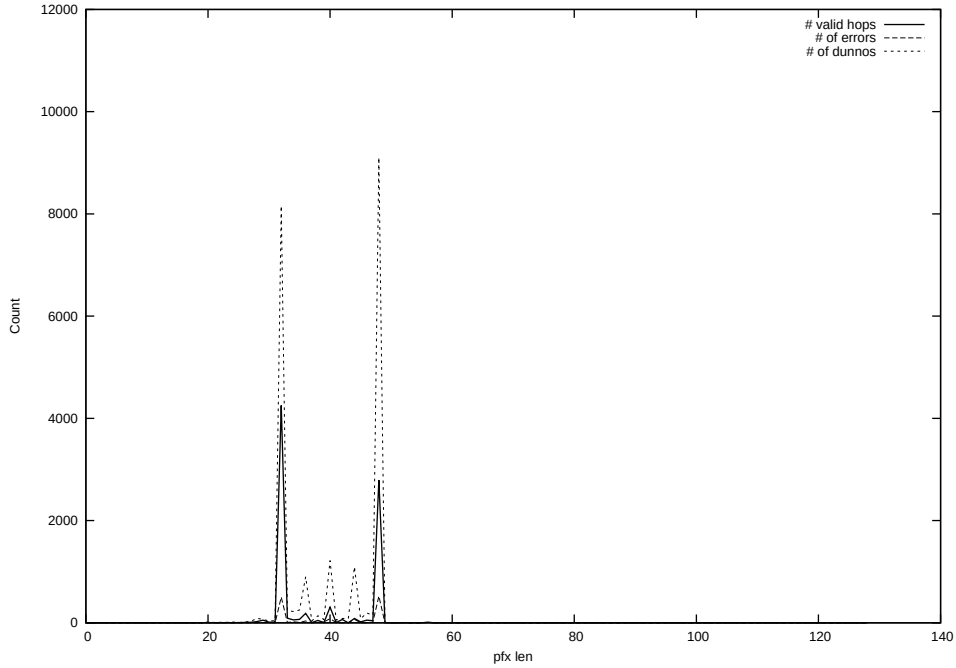
2012-09-19



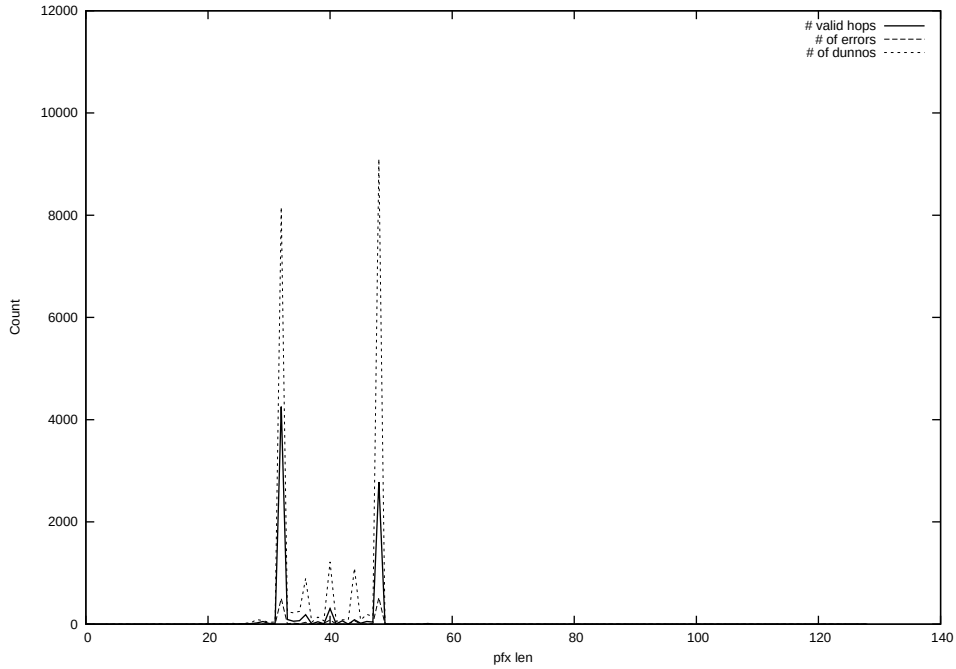
2012-09-20



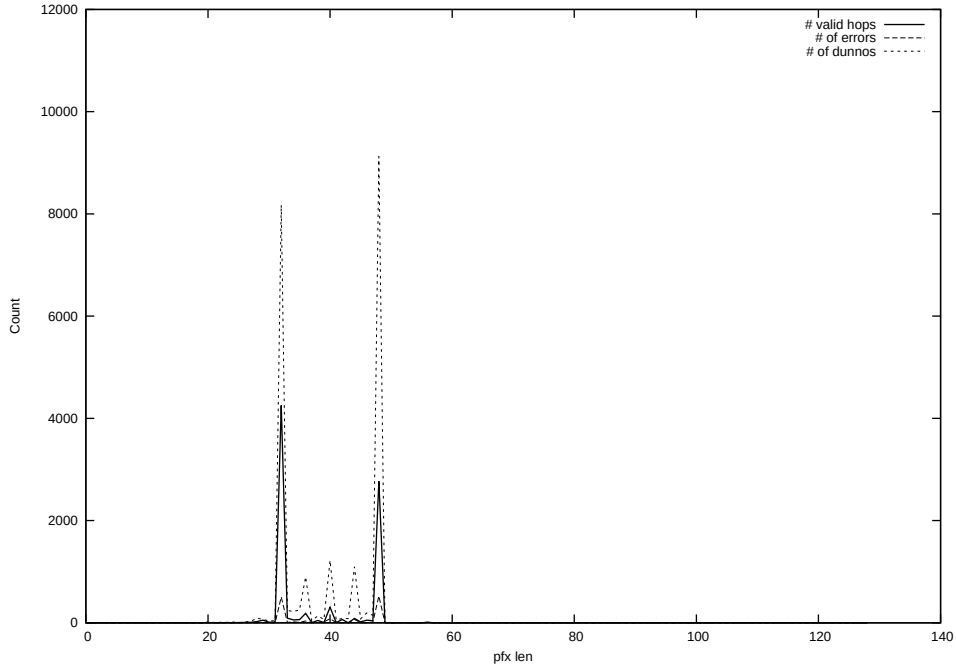
2012-09-21



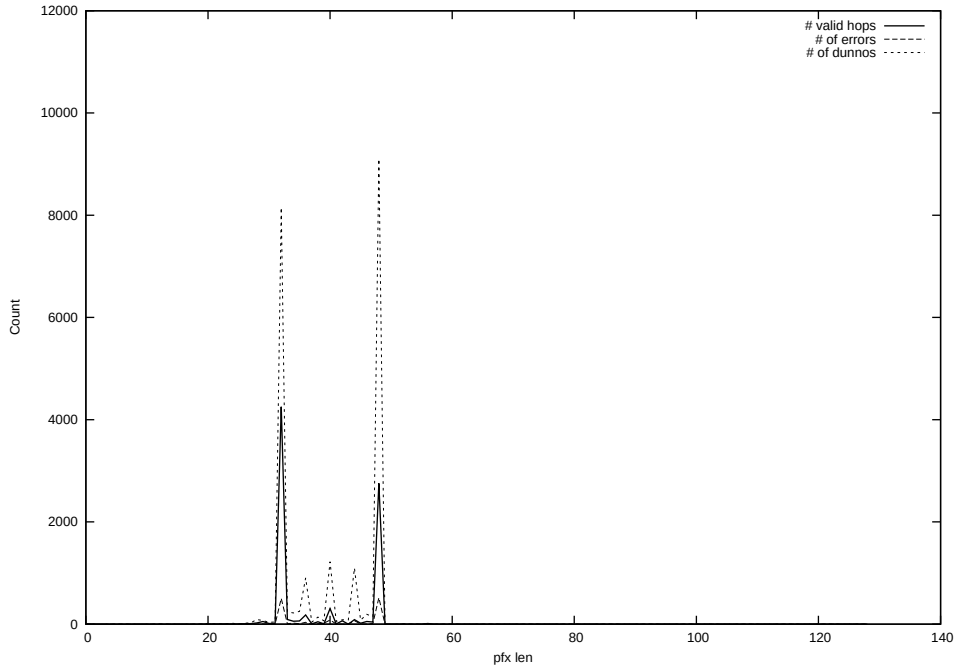
2012-09-22



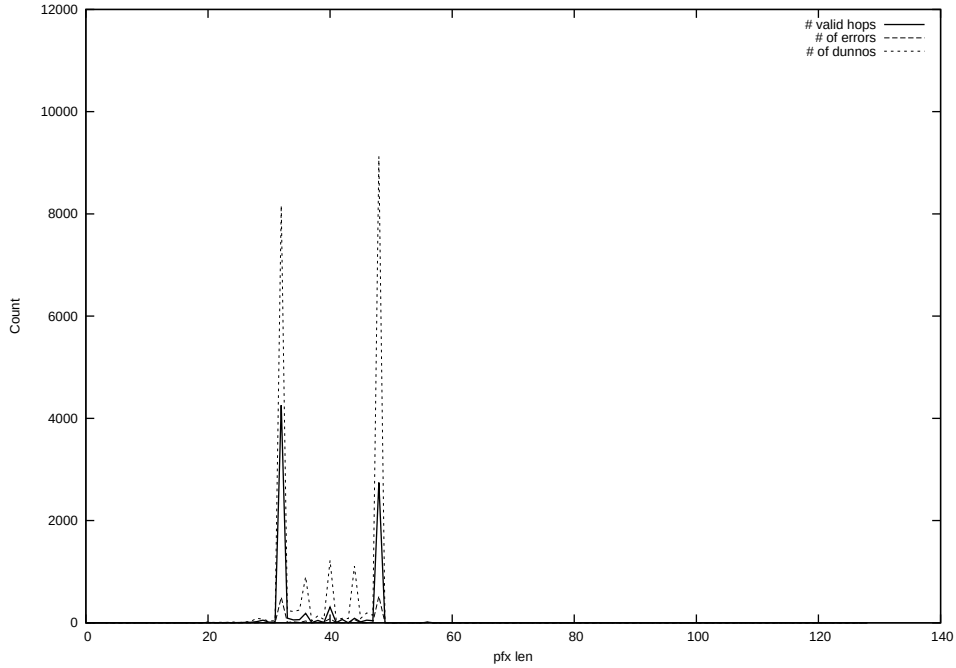
2012-09-23



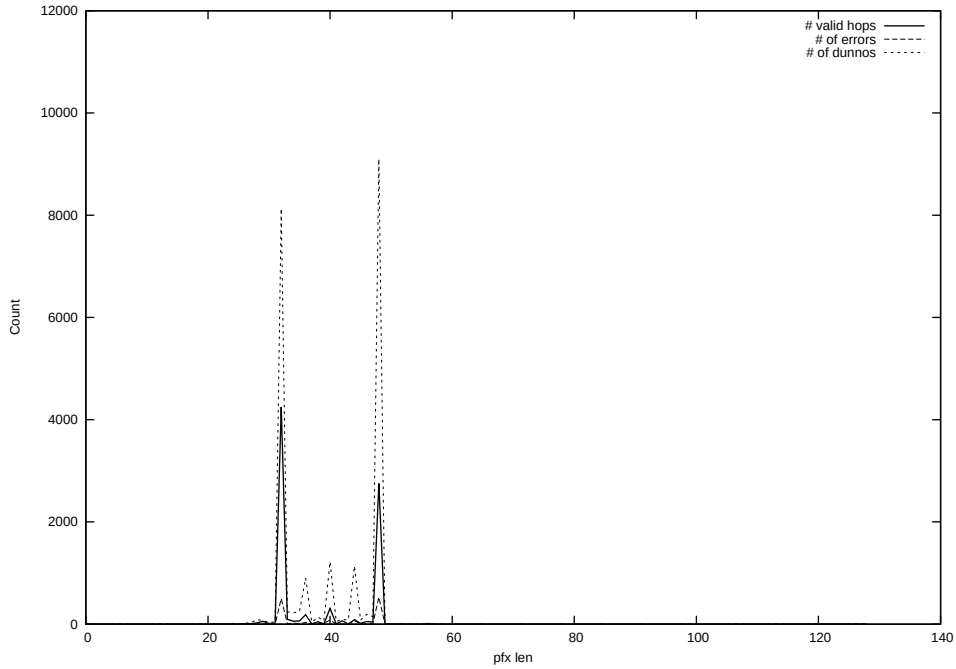
2012-09-24



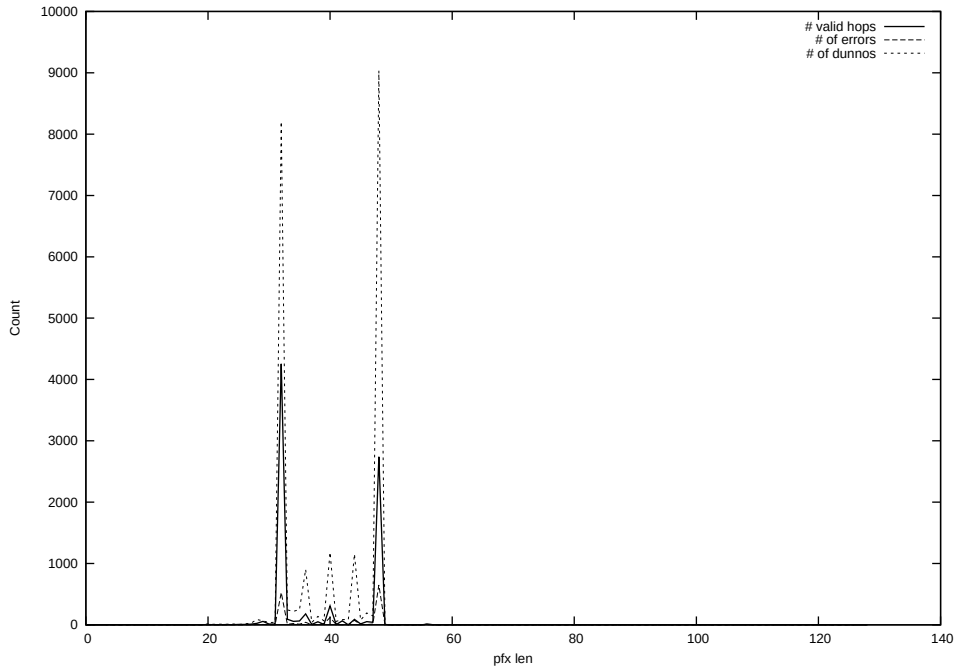
2012-09-25



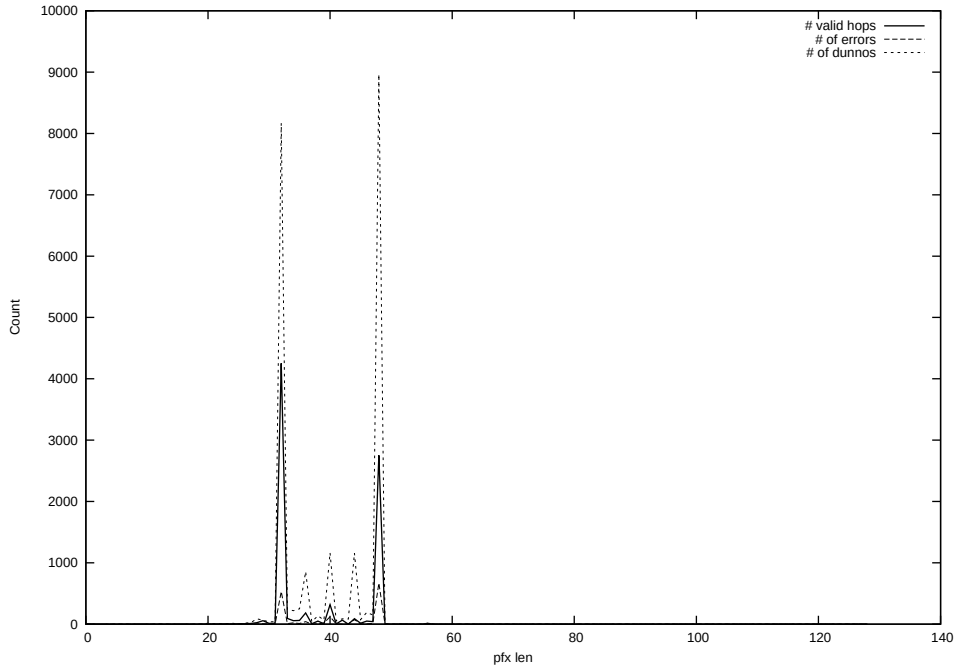
2012-09-26



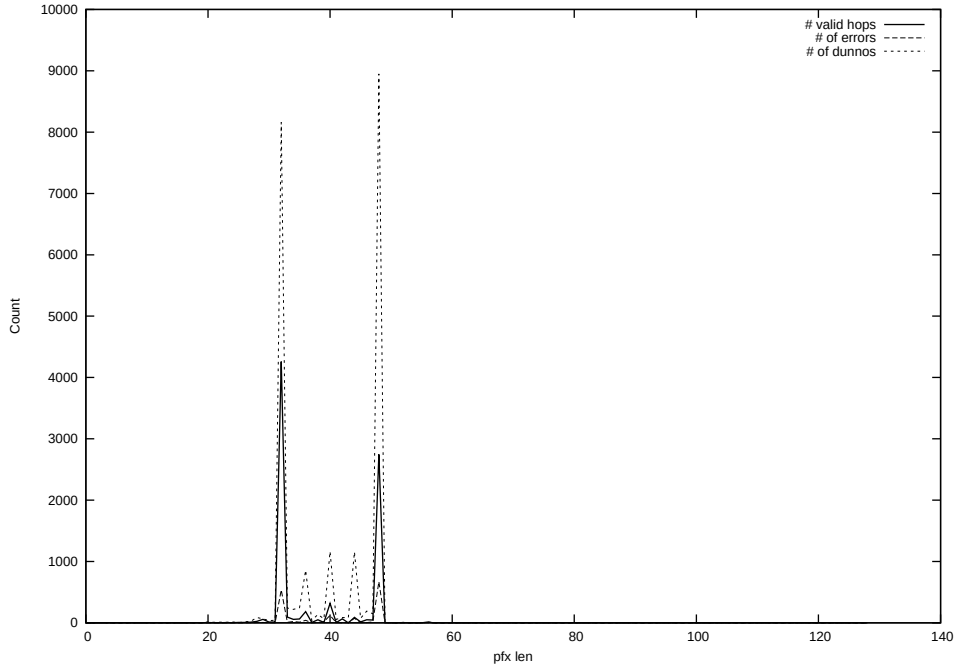
2012-09-27



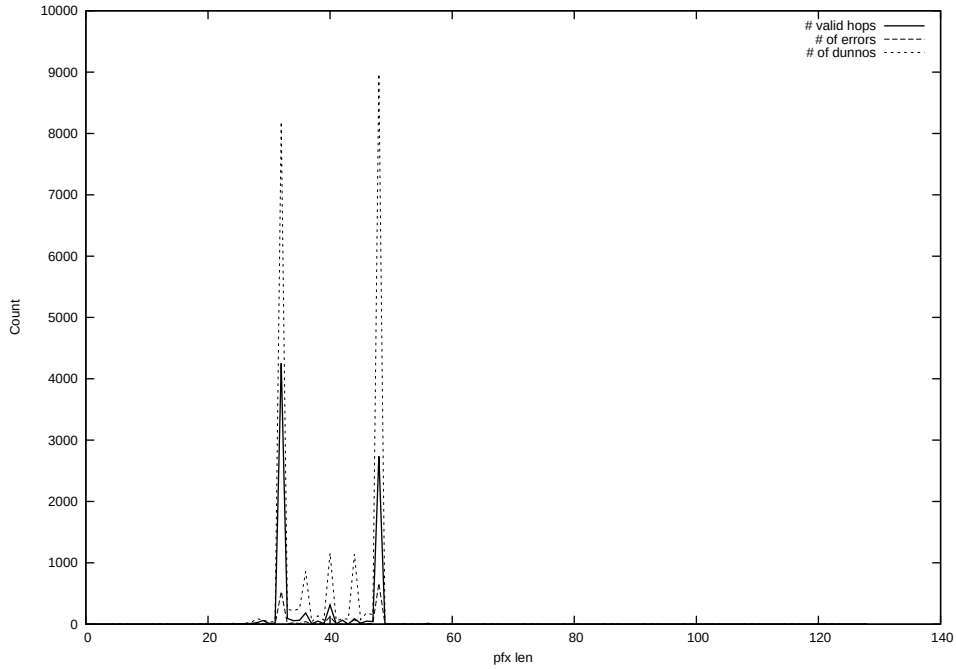
2012-09-28



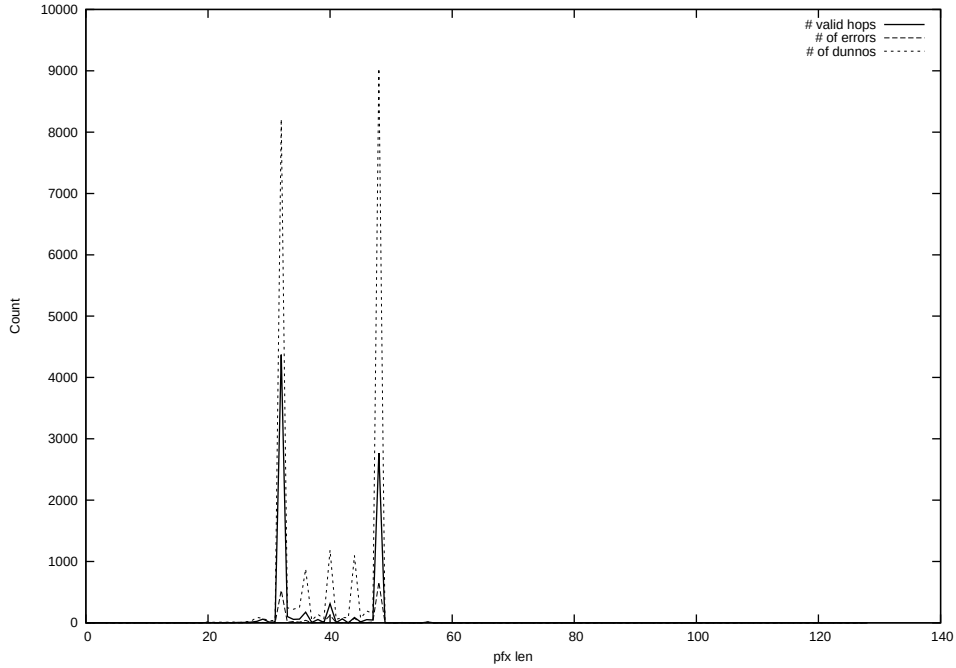
2012-09-29



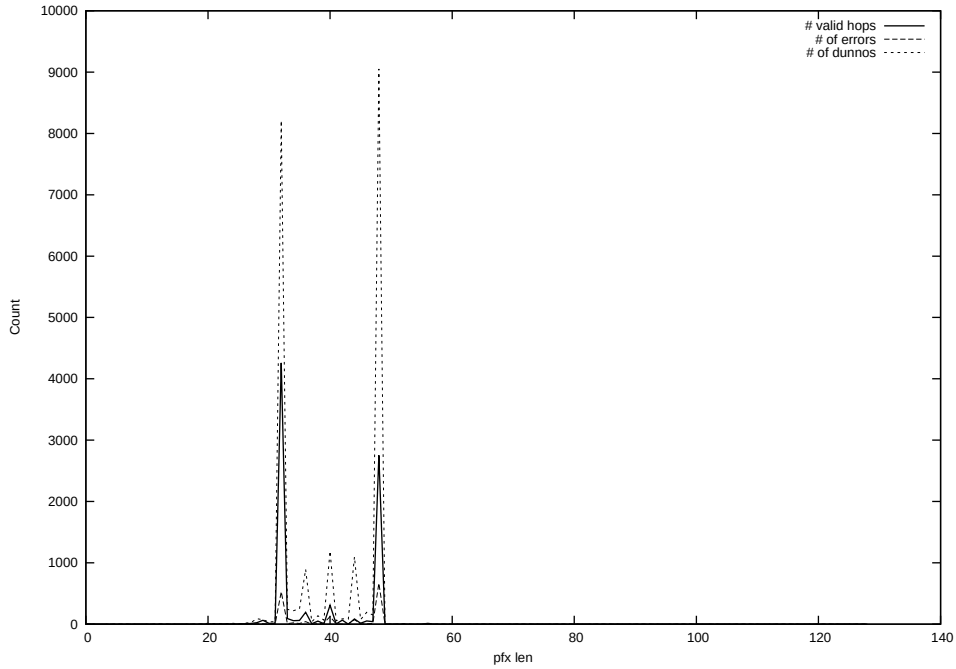
2012-09-30



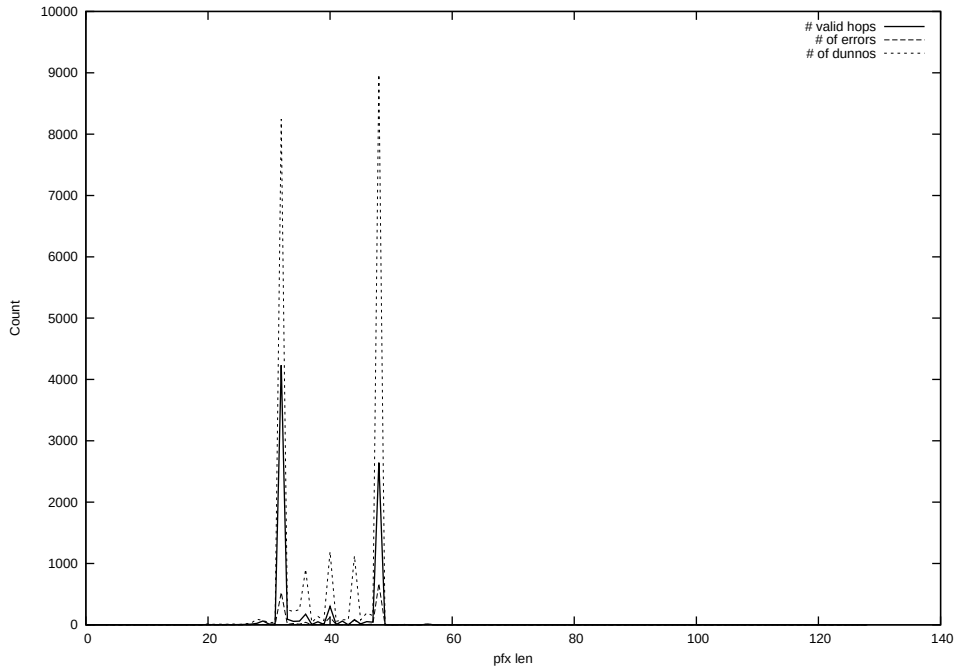
2012-10-01



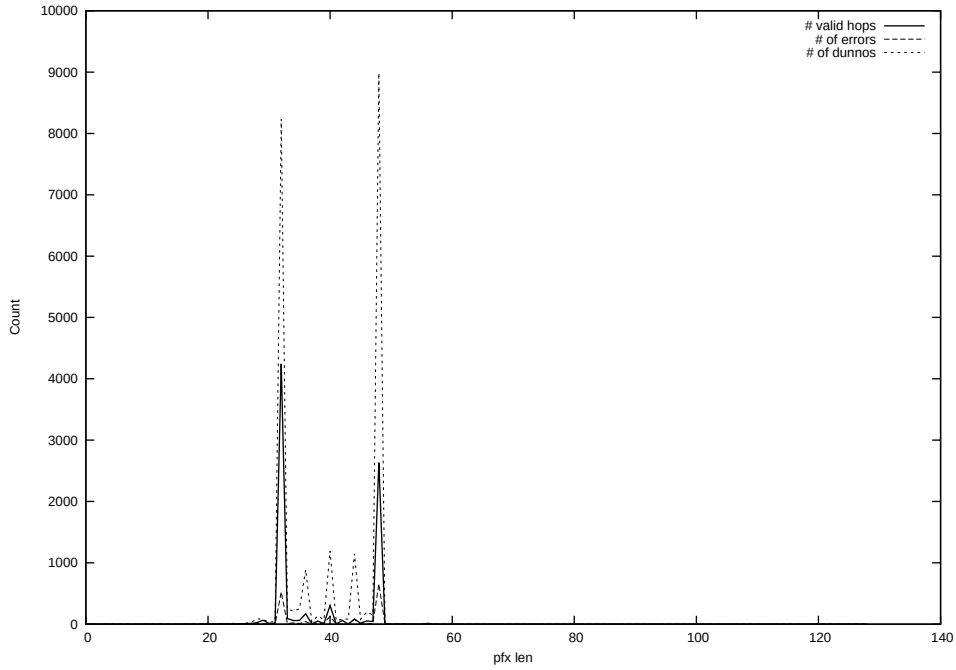
2012-10-02



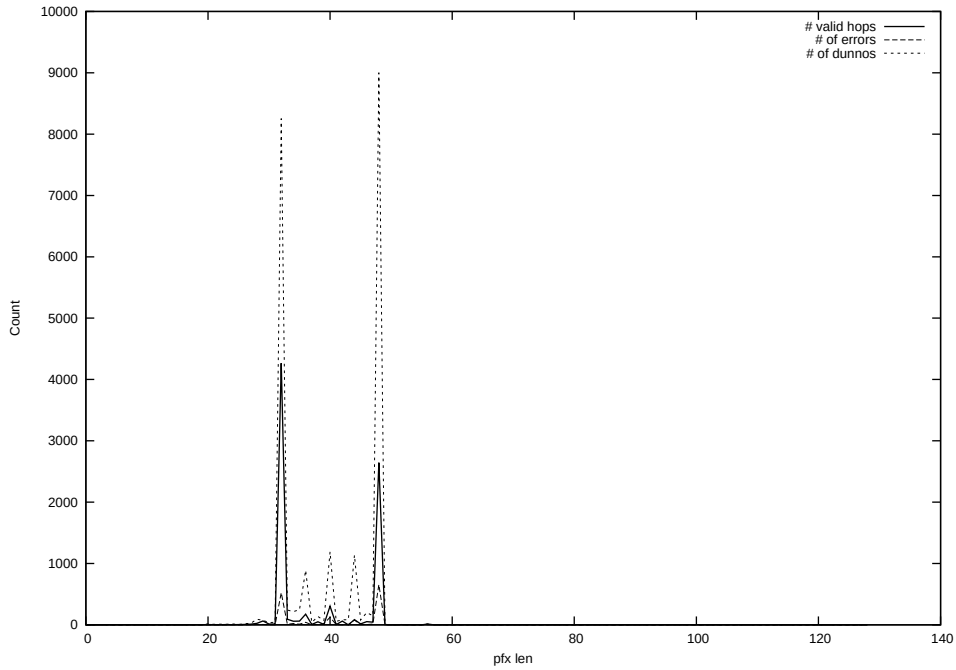
2012-10-03



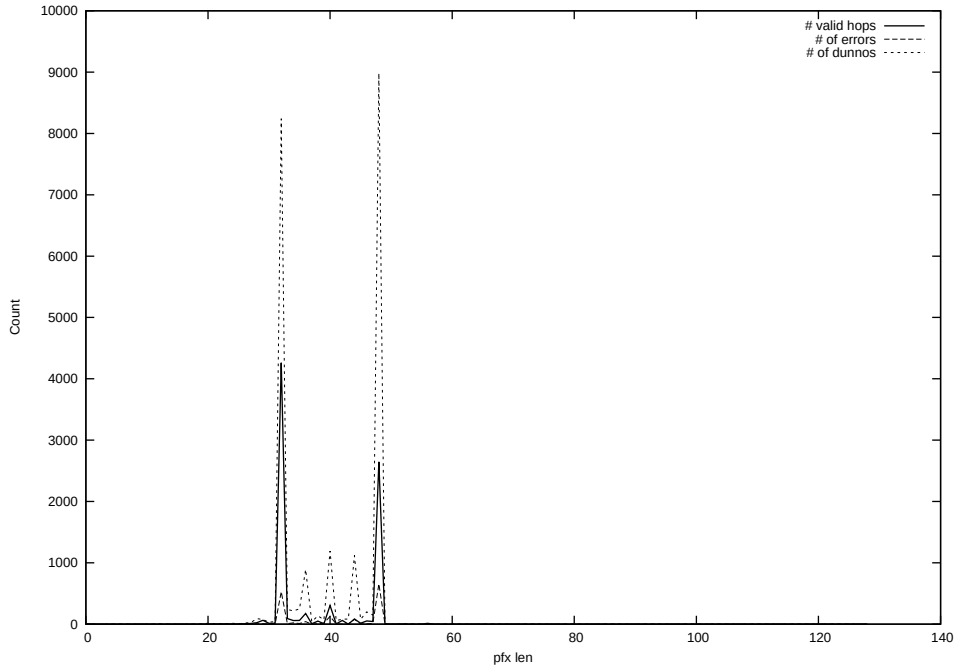
2012-10-04



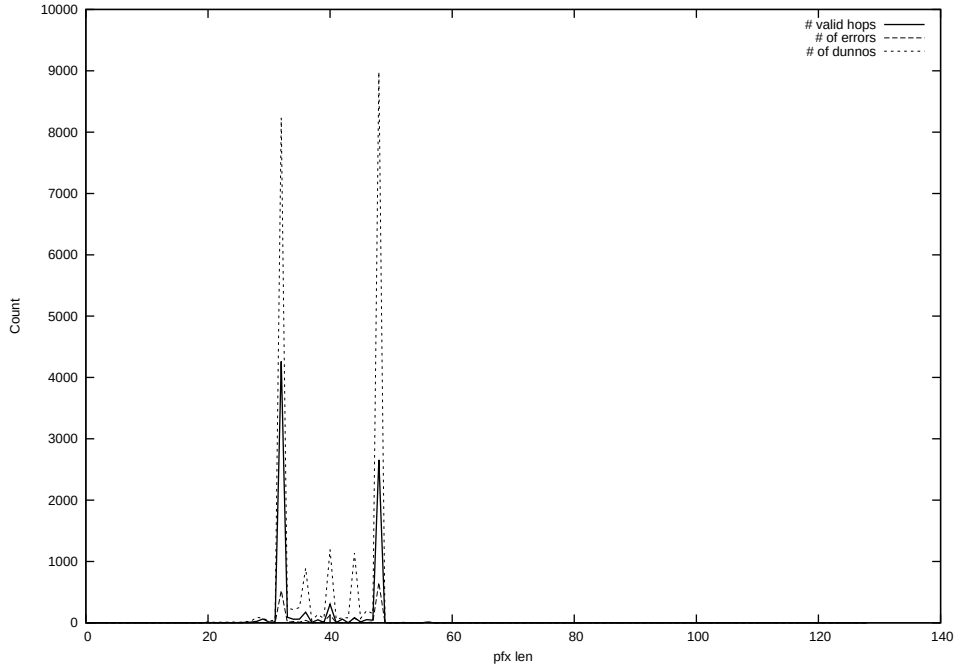
2012-10-05



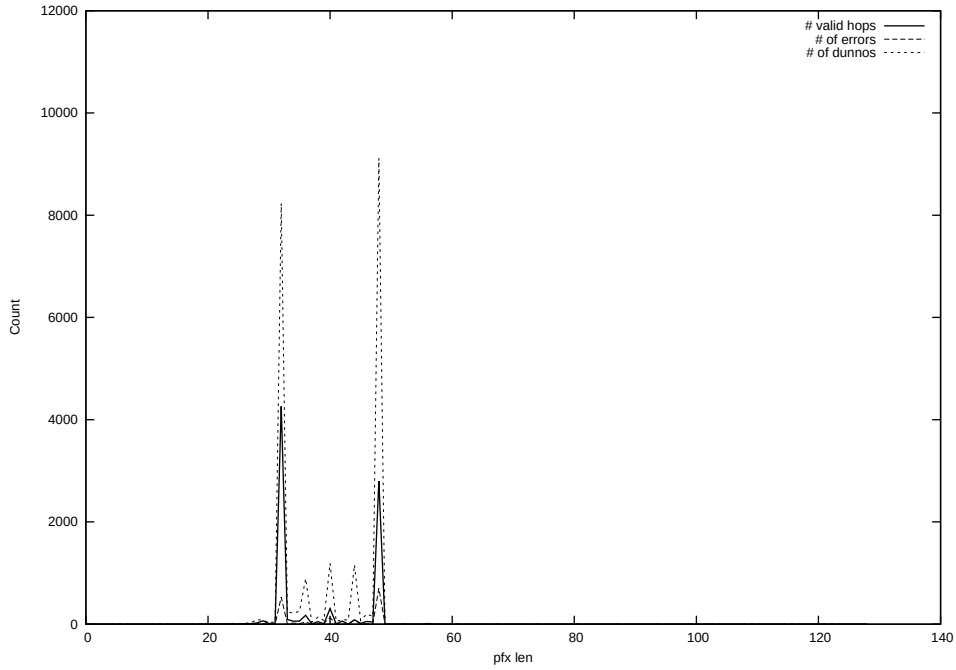
2012-10-06



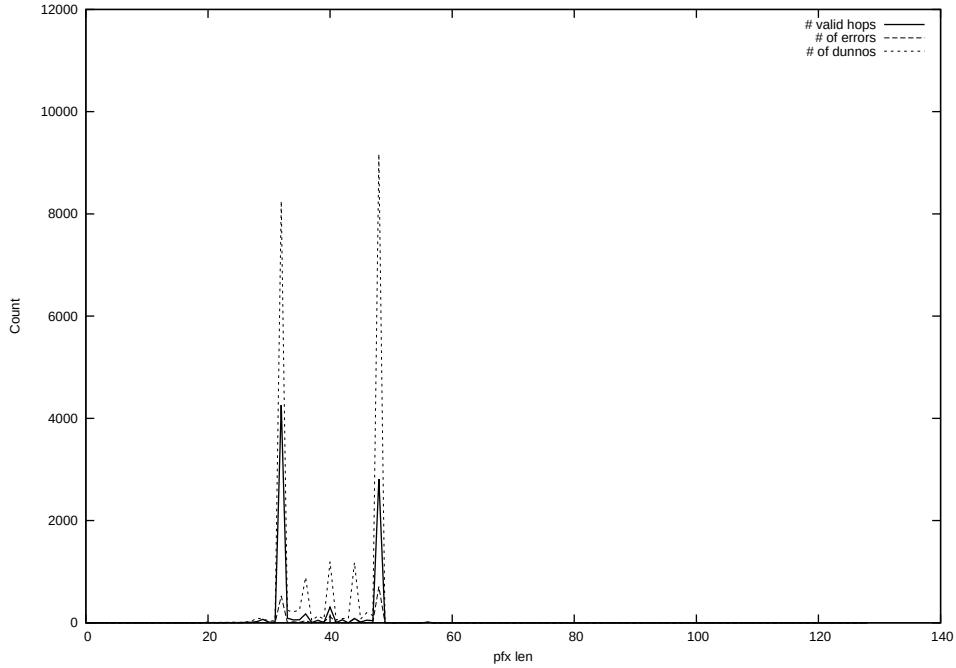
2012-10-07



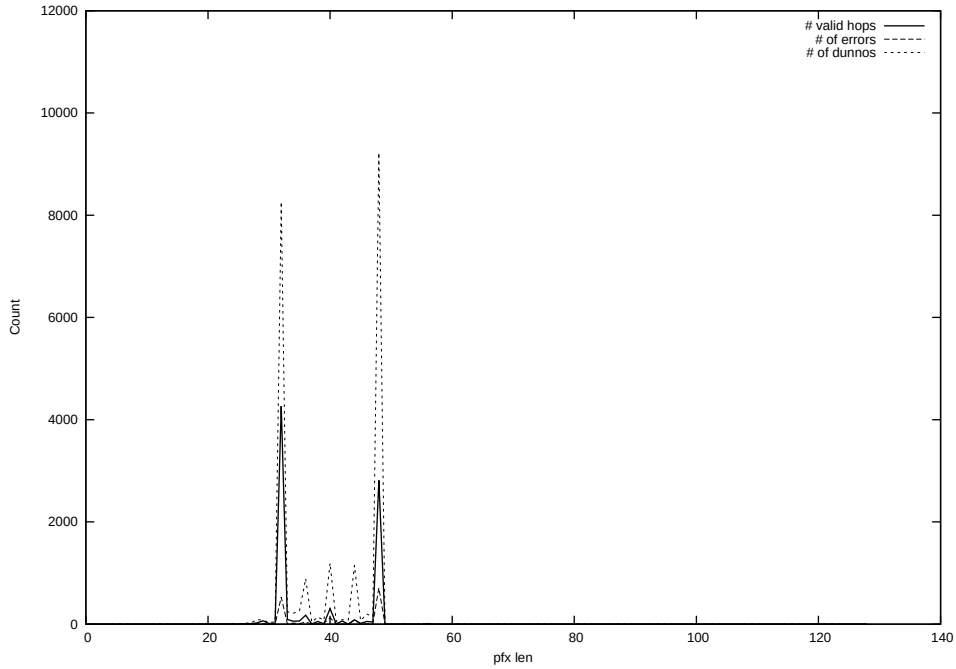
2012-10-08



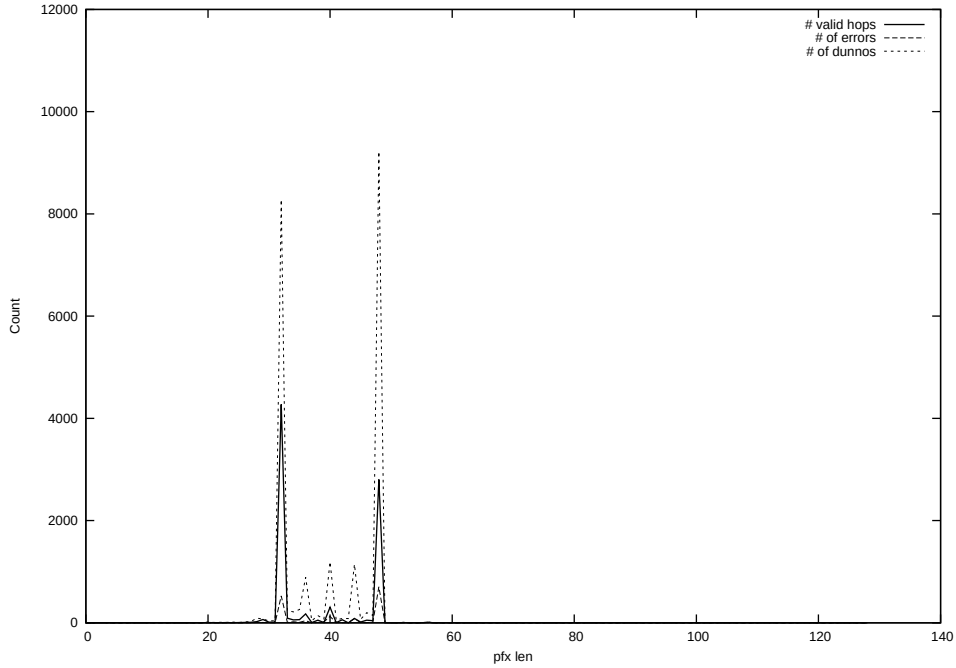
2012-10-09



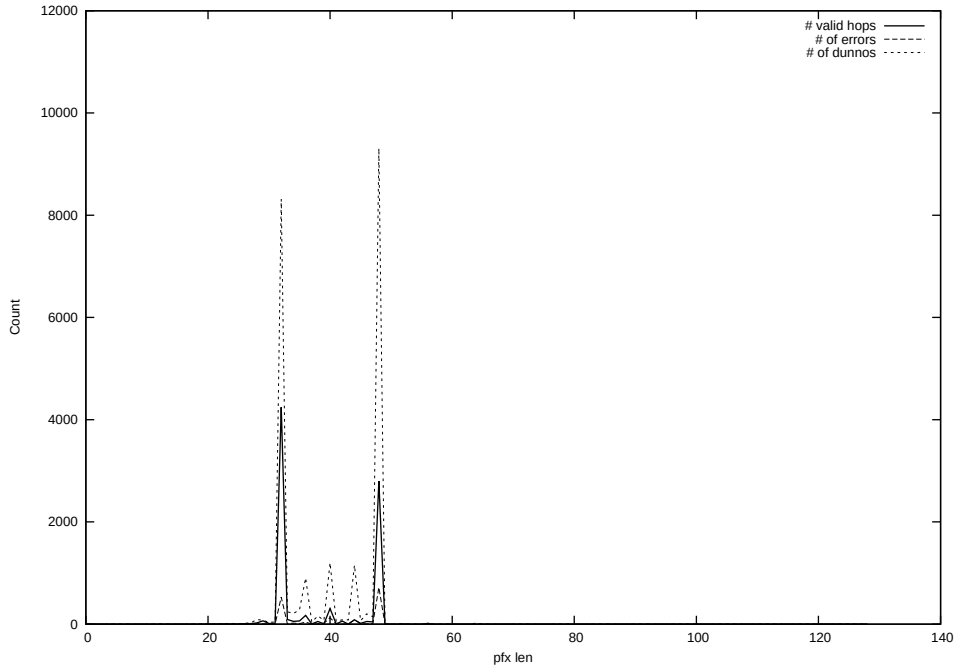
2012-10-10



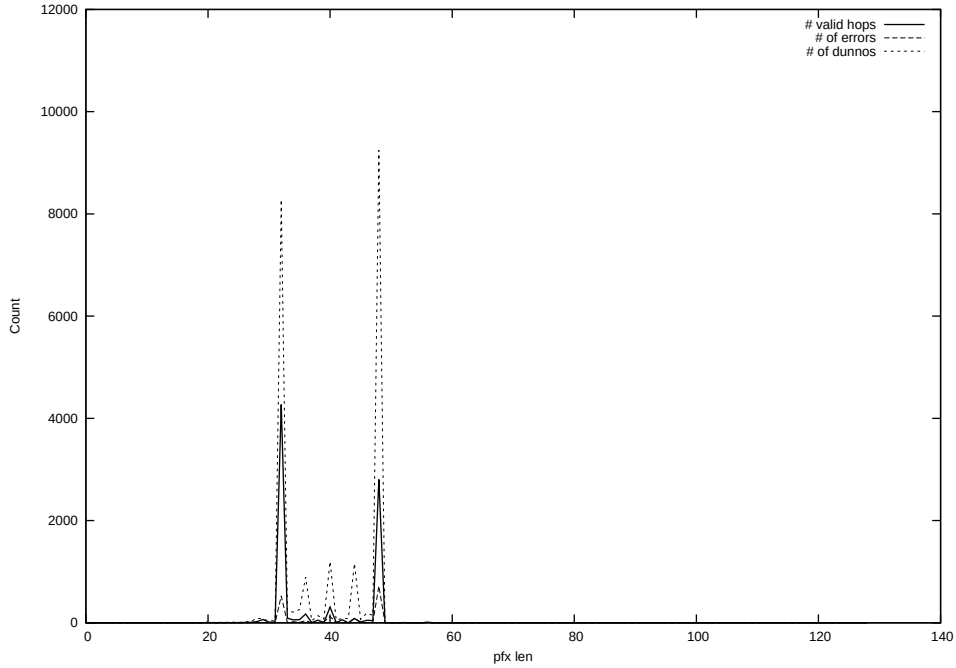
2012-10-11



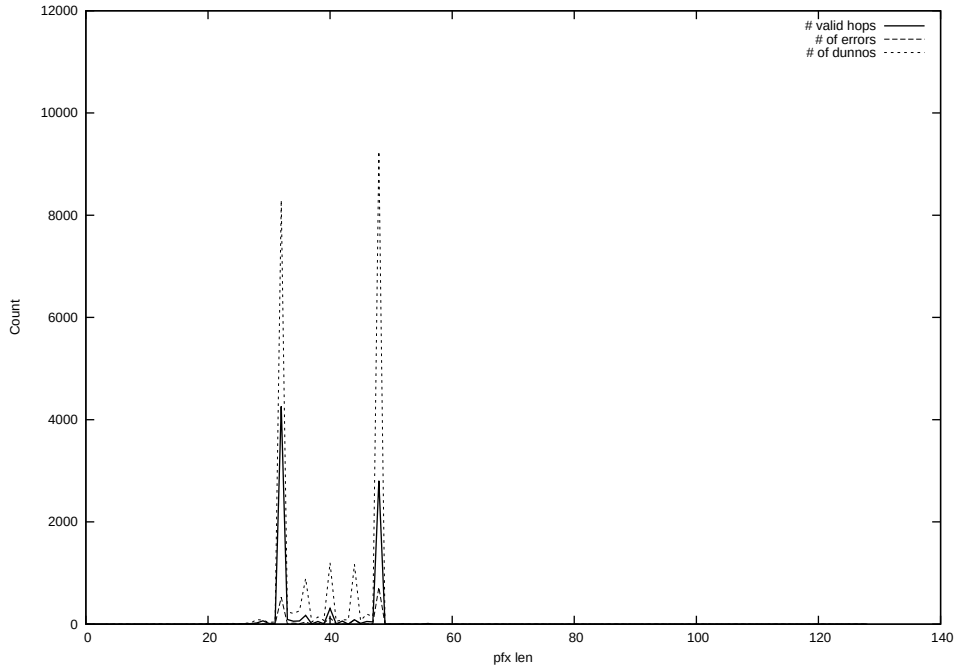
2012-10-12



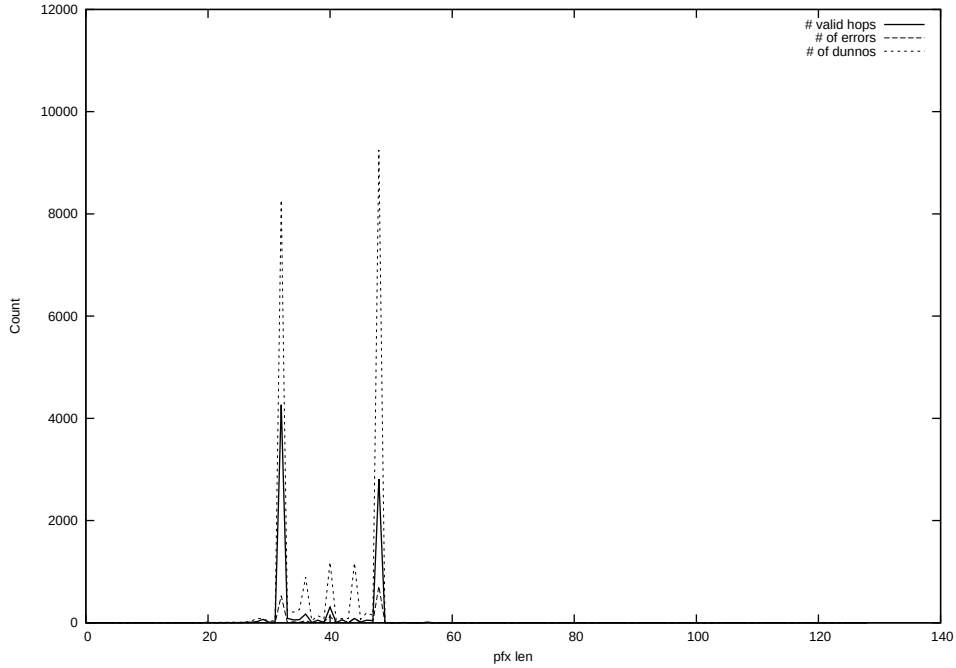
2012-10-13



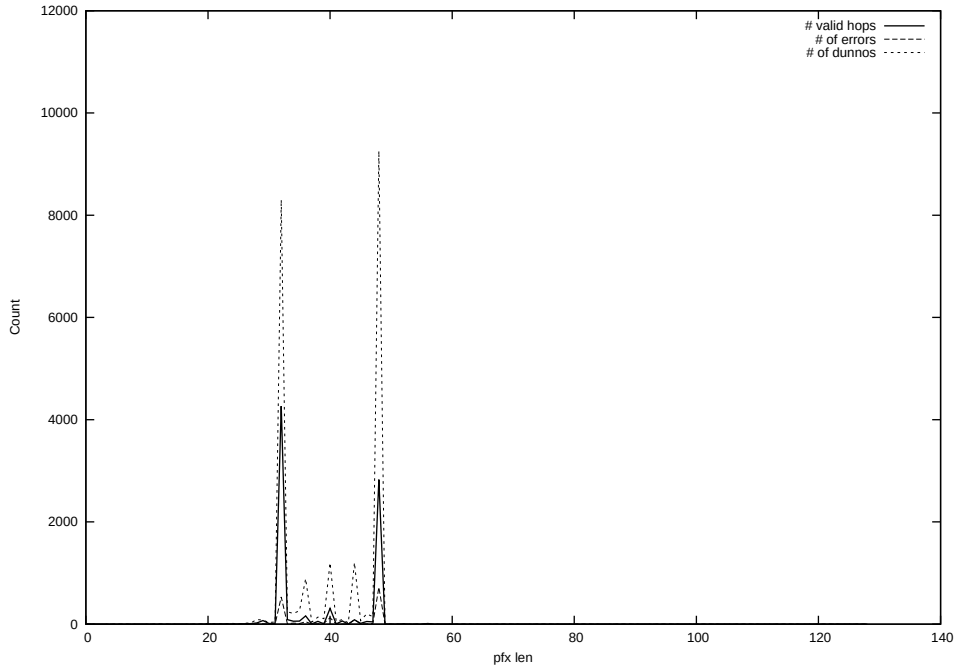
2012-10-14



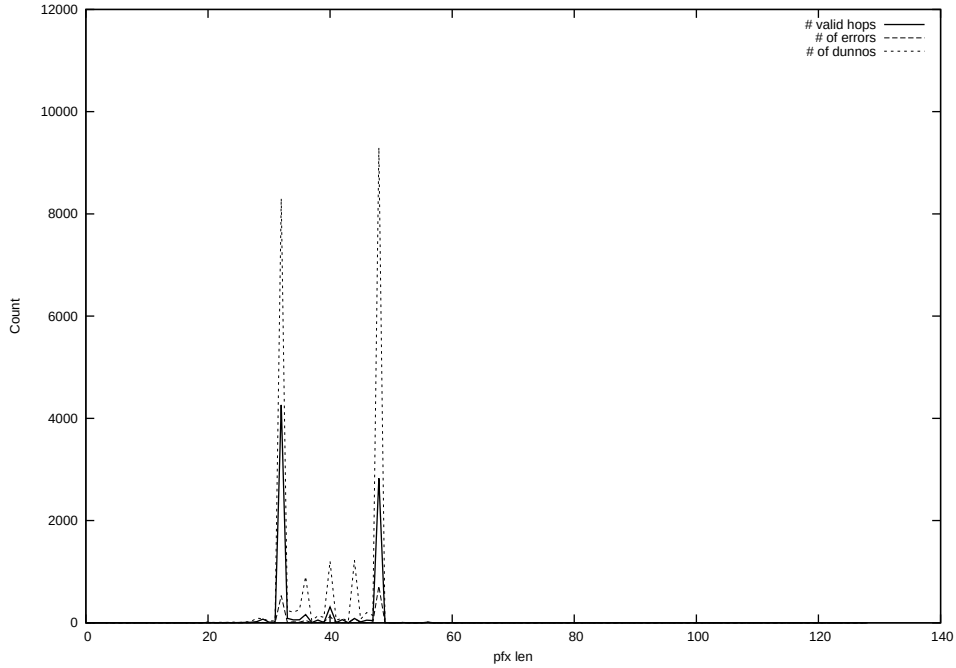
2012-10-15



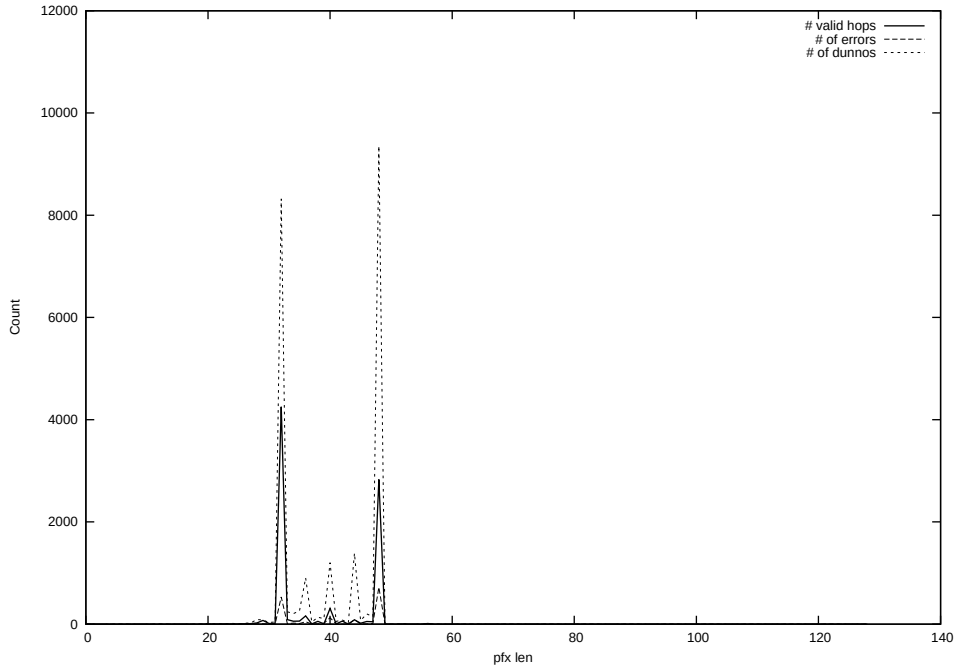
2012-10-16



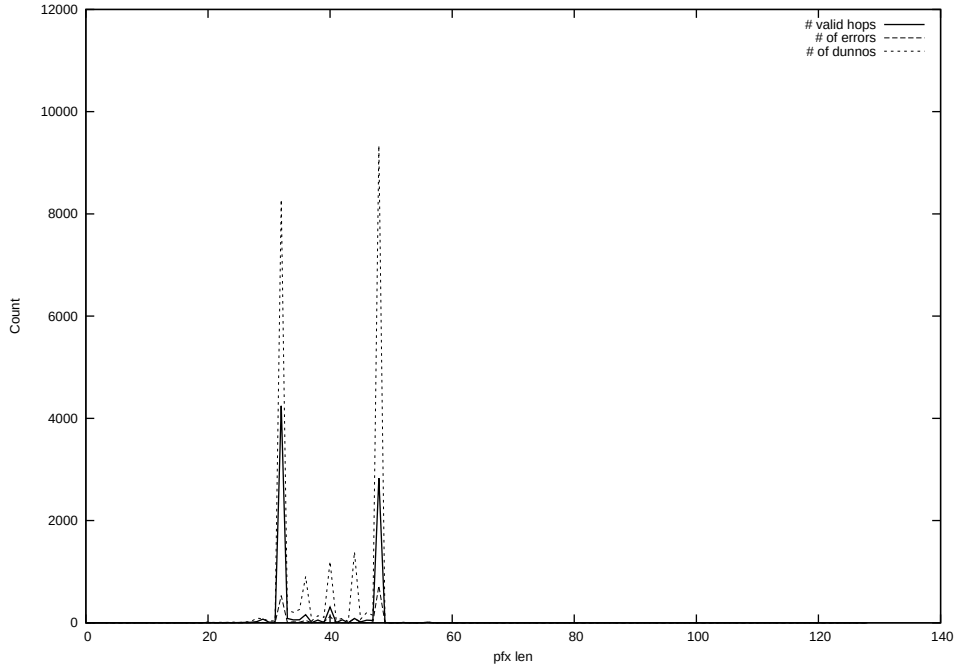
2012-10-18



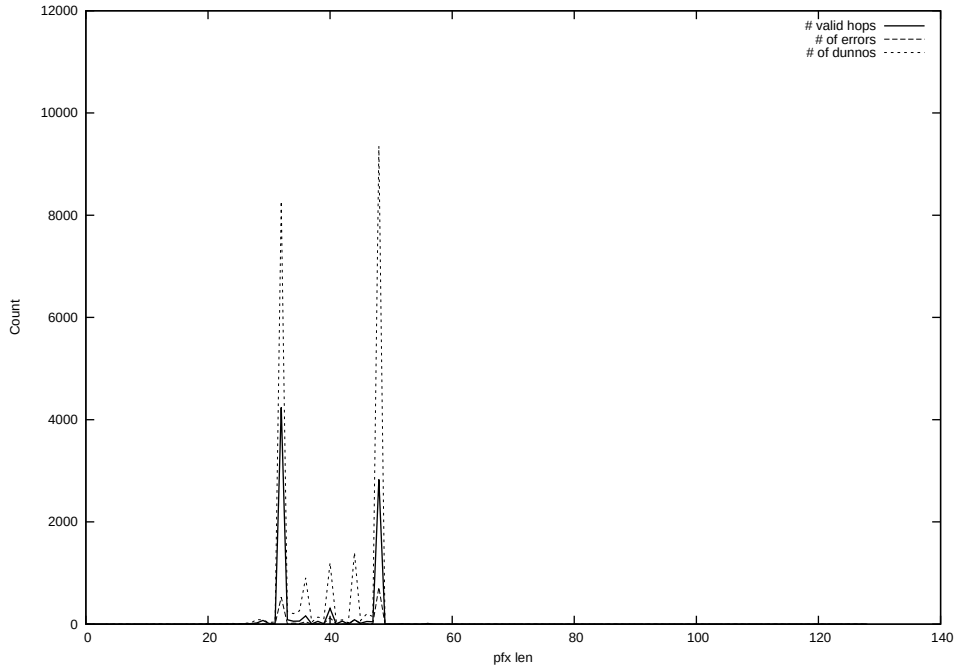
2012-10-19



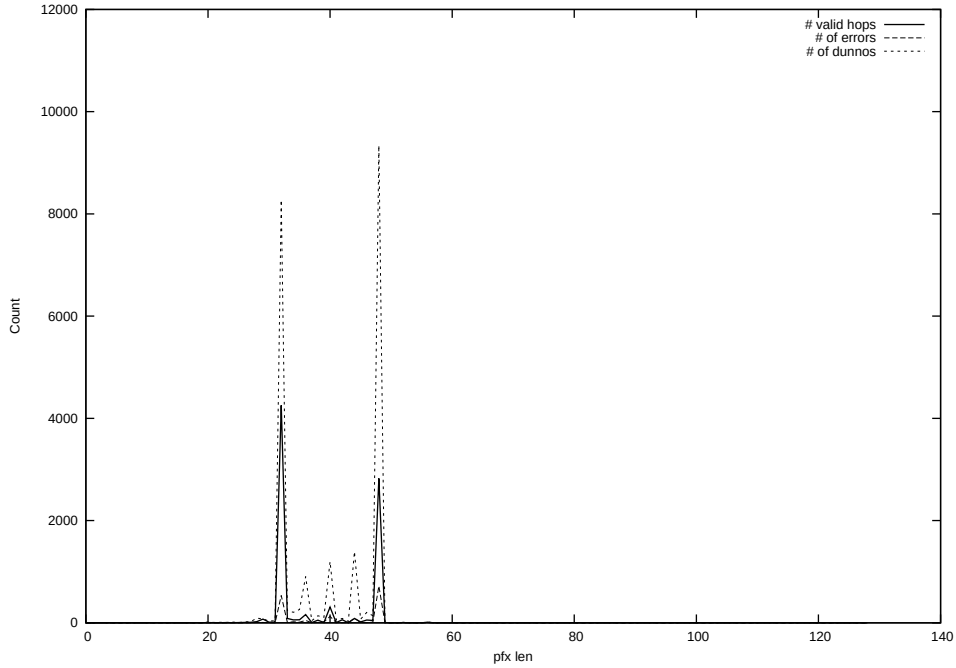
2012-10-20



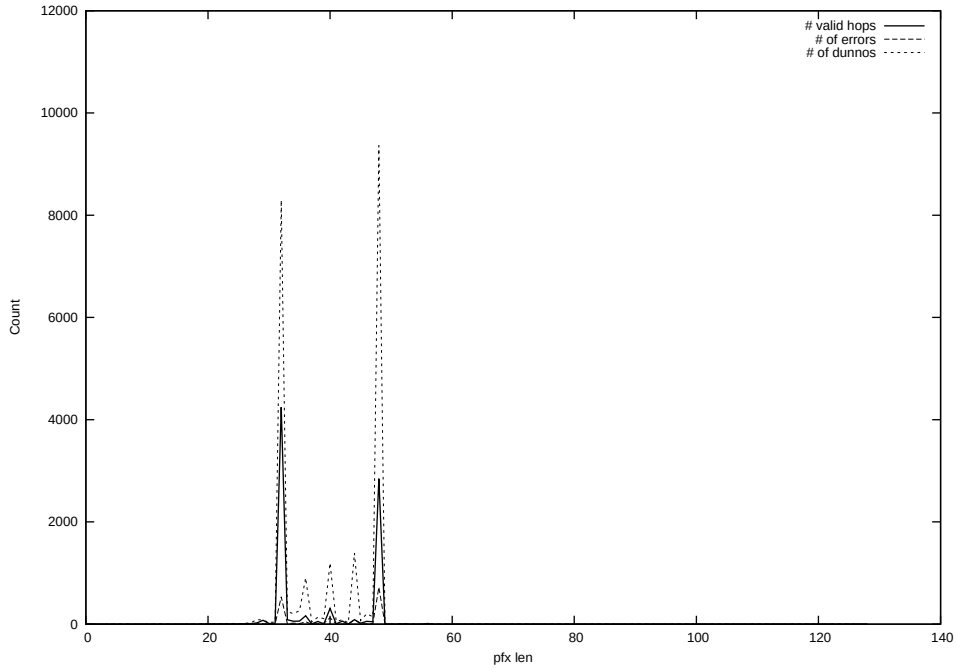
2012-10-21



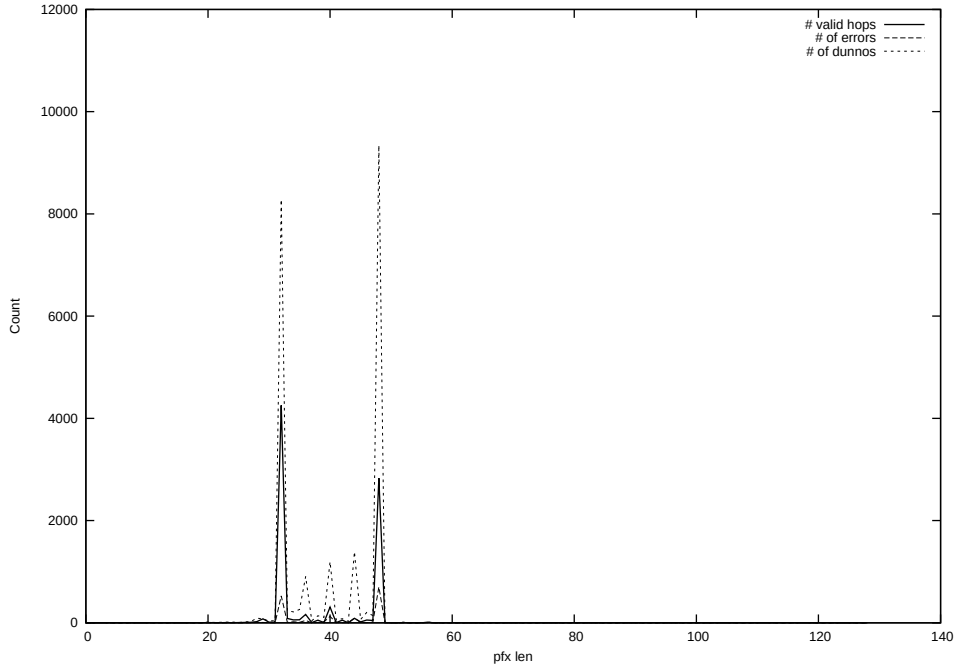
2012-10-22



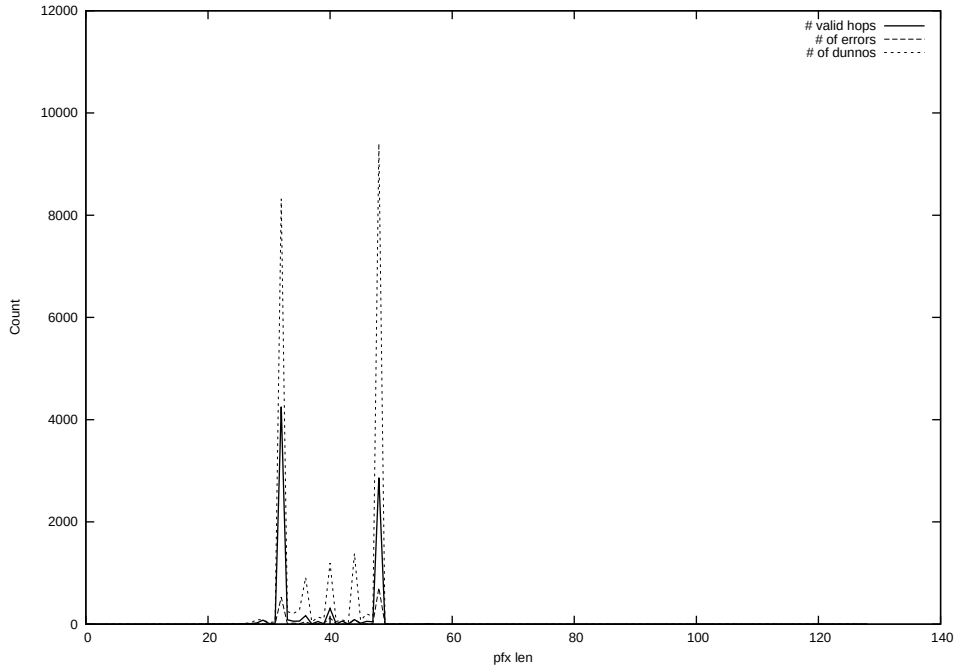
2012-10-23



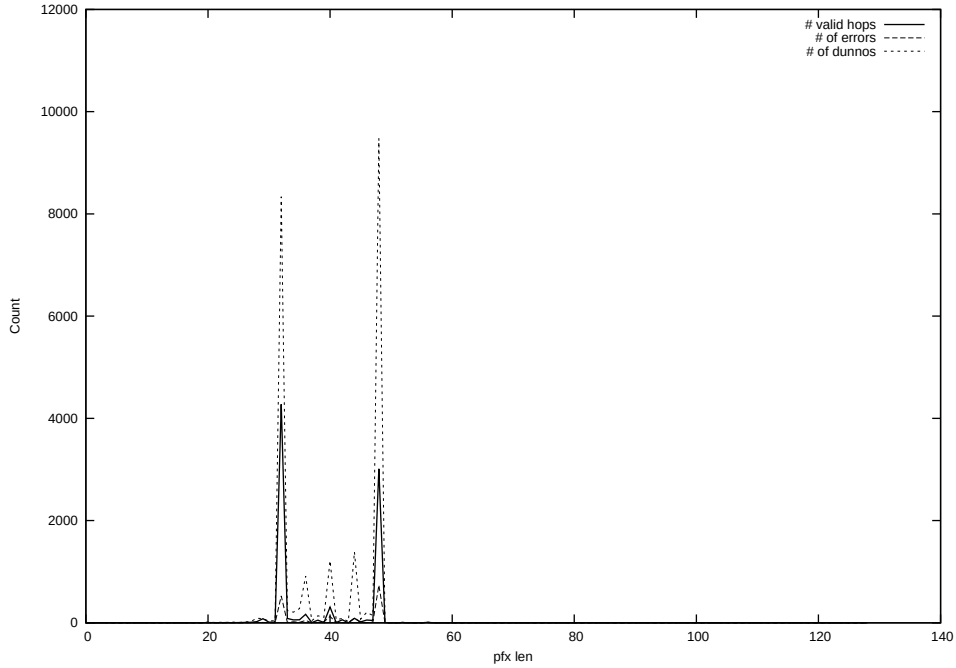
2012-10-24



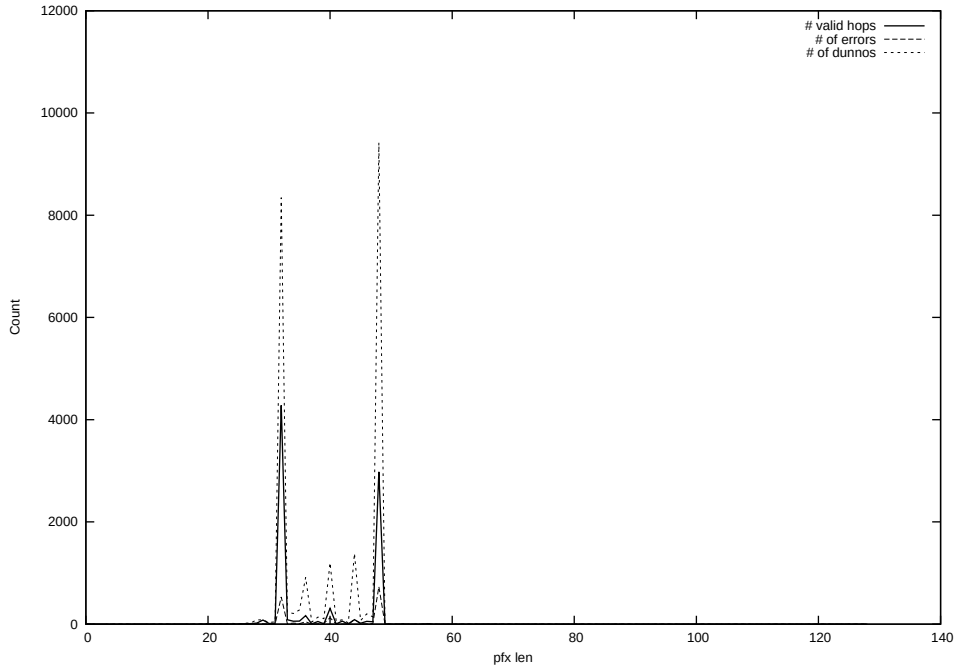
2012-10-25



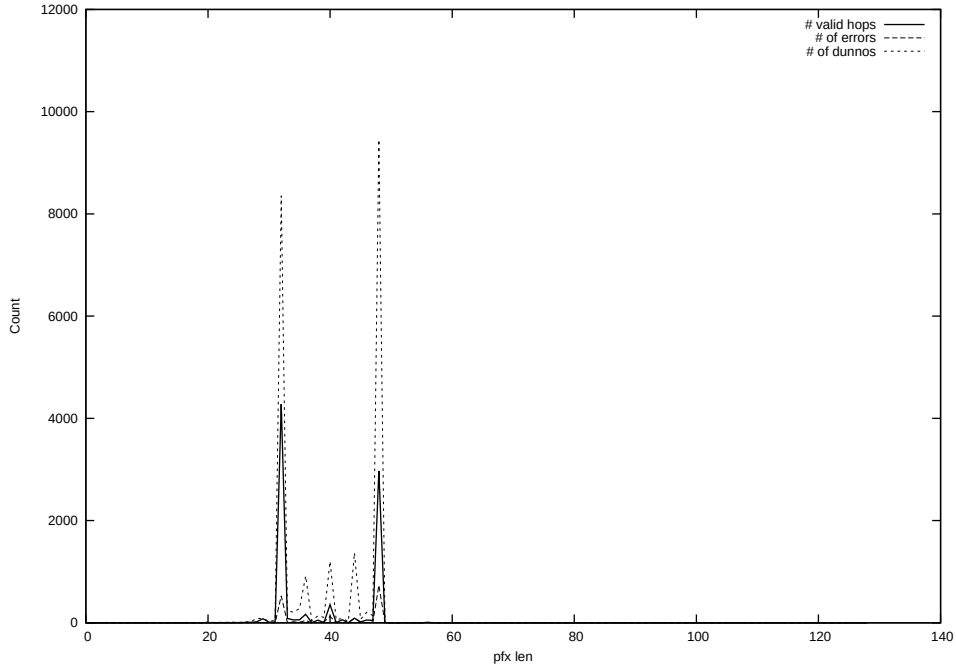
2012-10-26



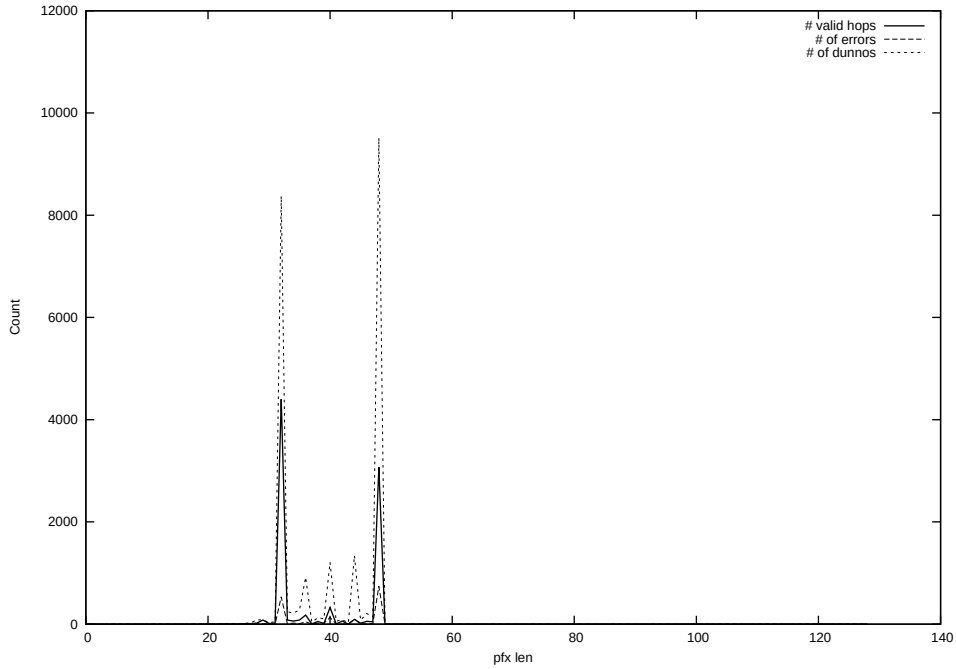
2012-10-27



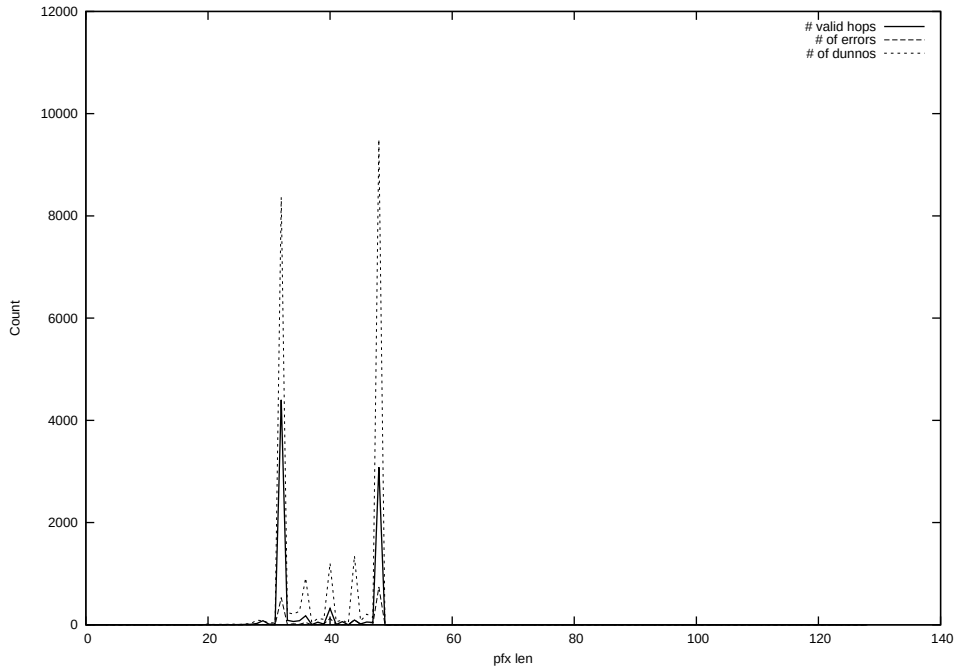
2012-10-28



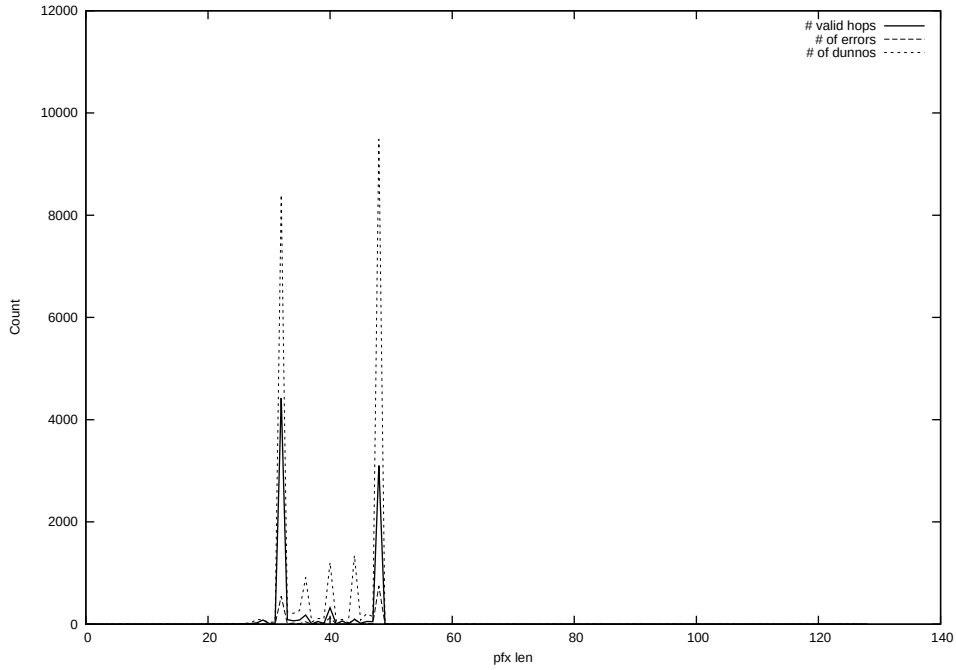
2012-10-29



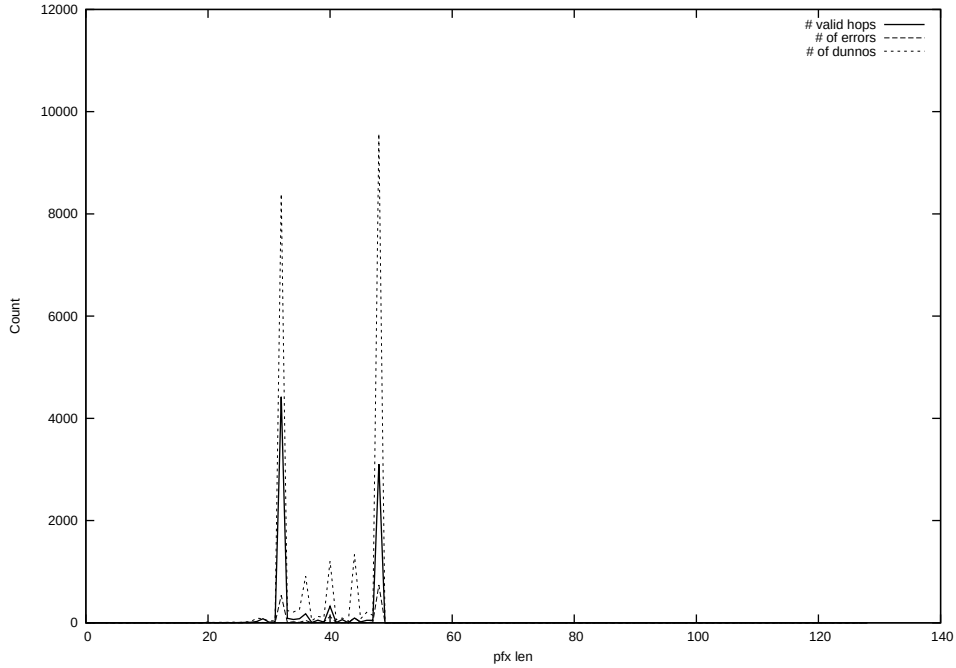
2012-10-30



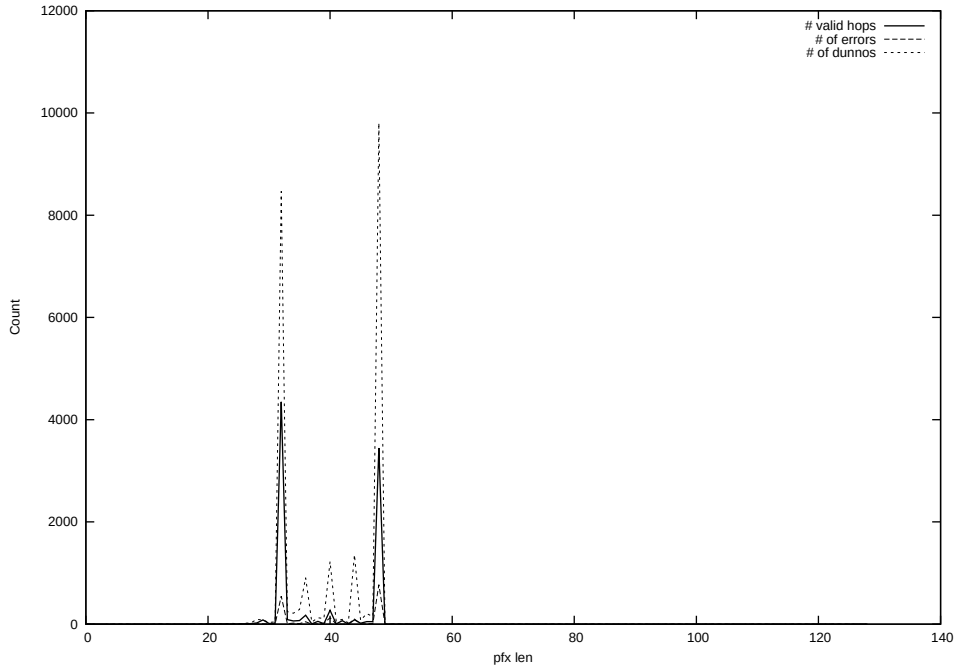
2012-10-31



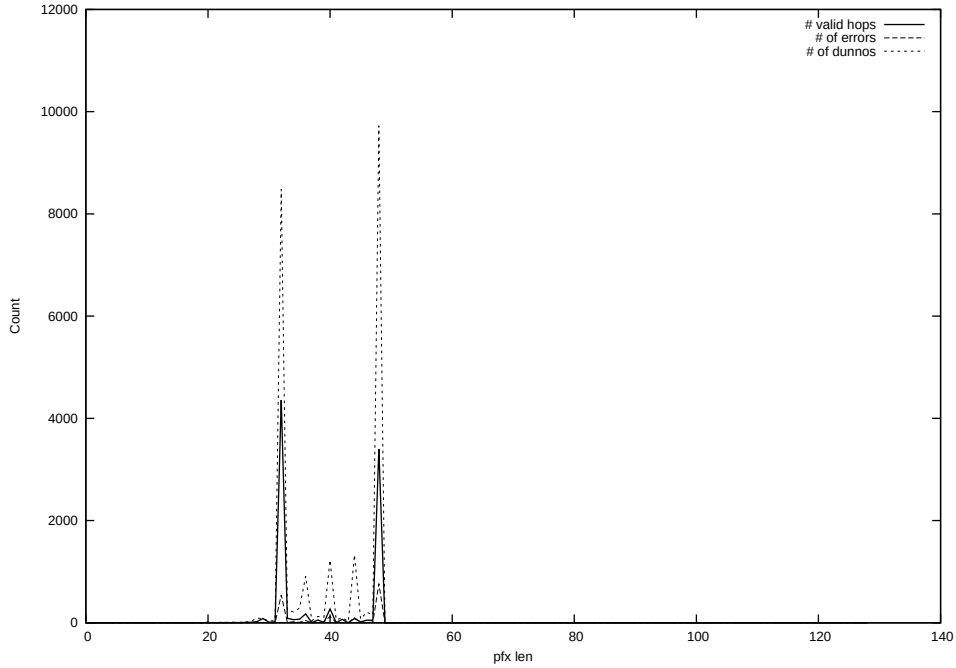
2012-11-01



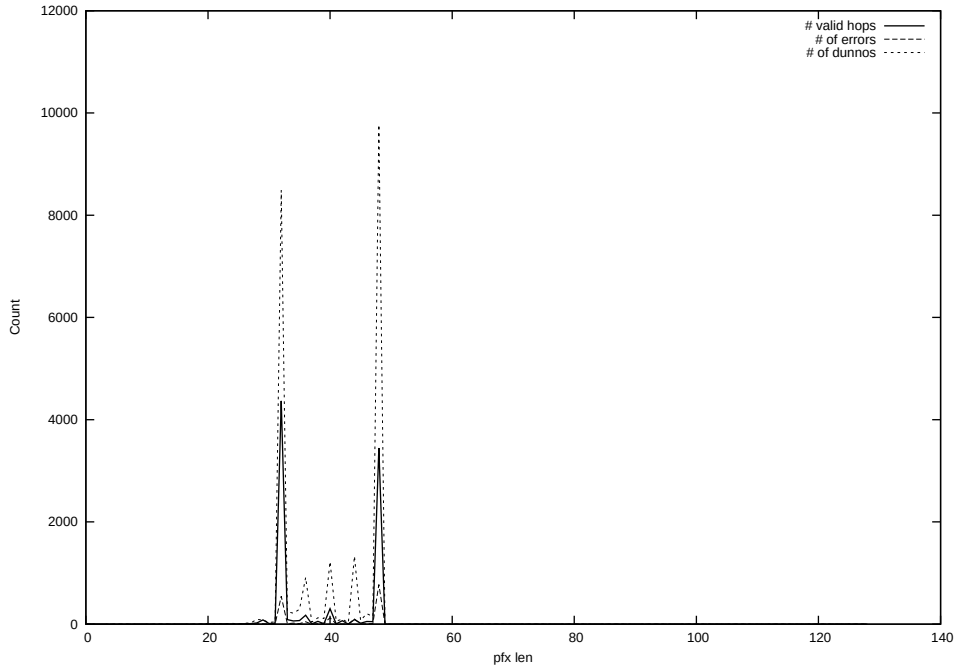
2012-11-02



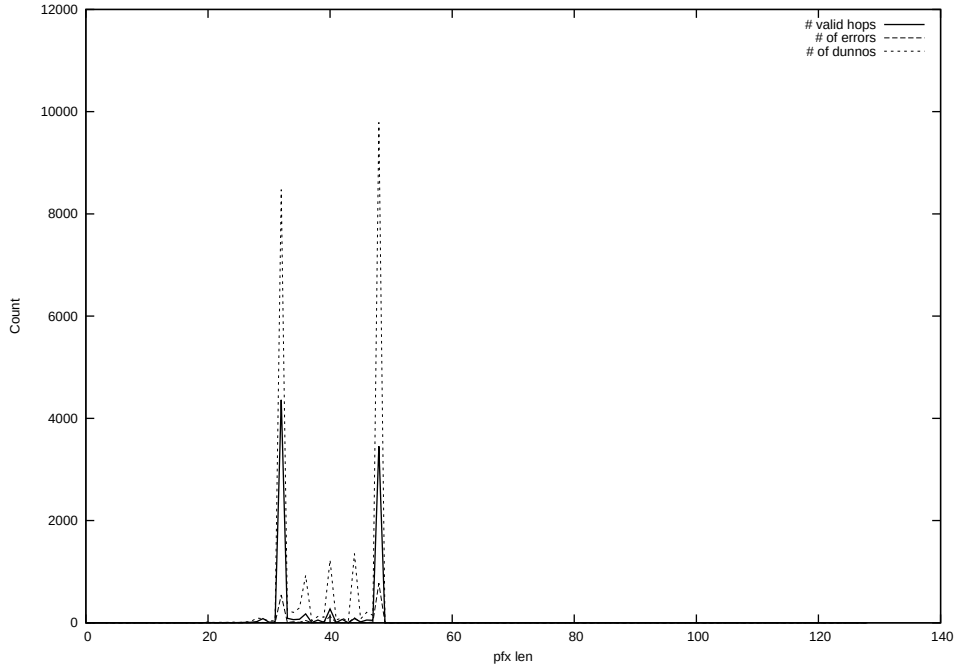
2012-11-03



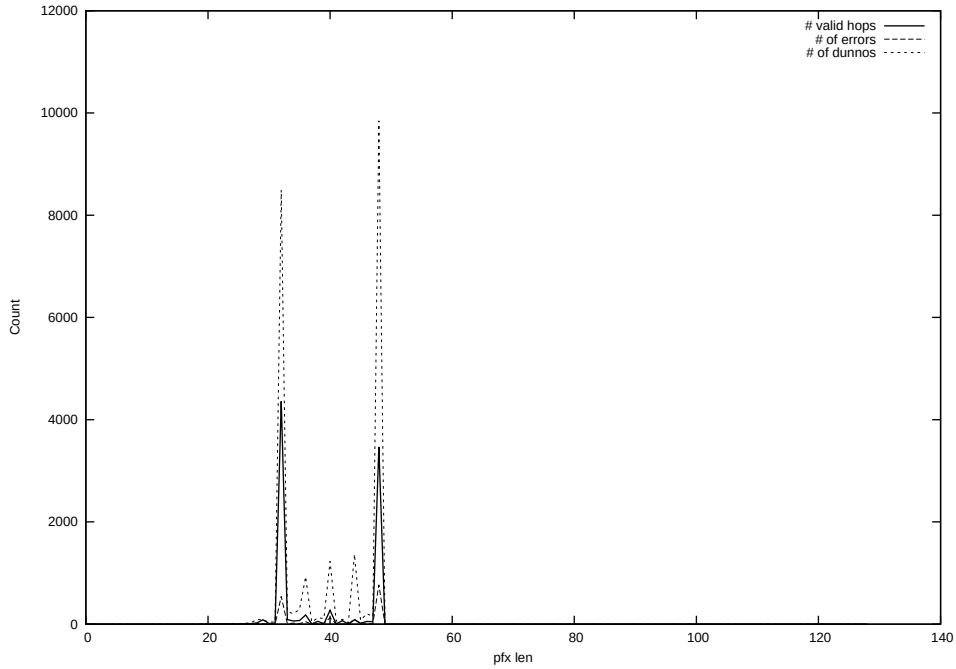
2012-11-04



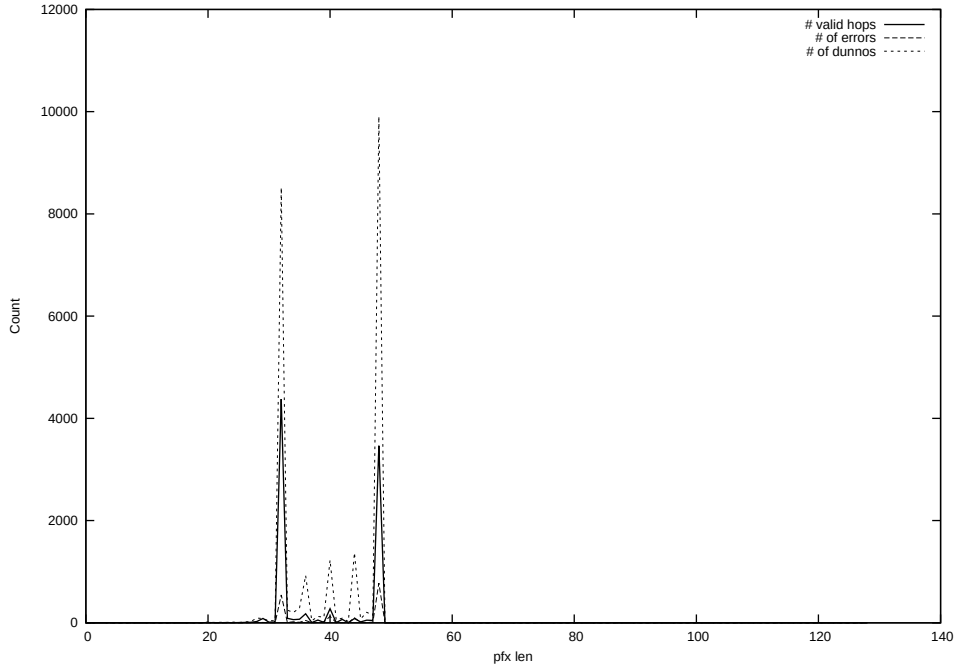
2012-11-05



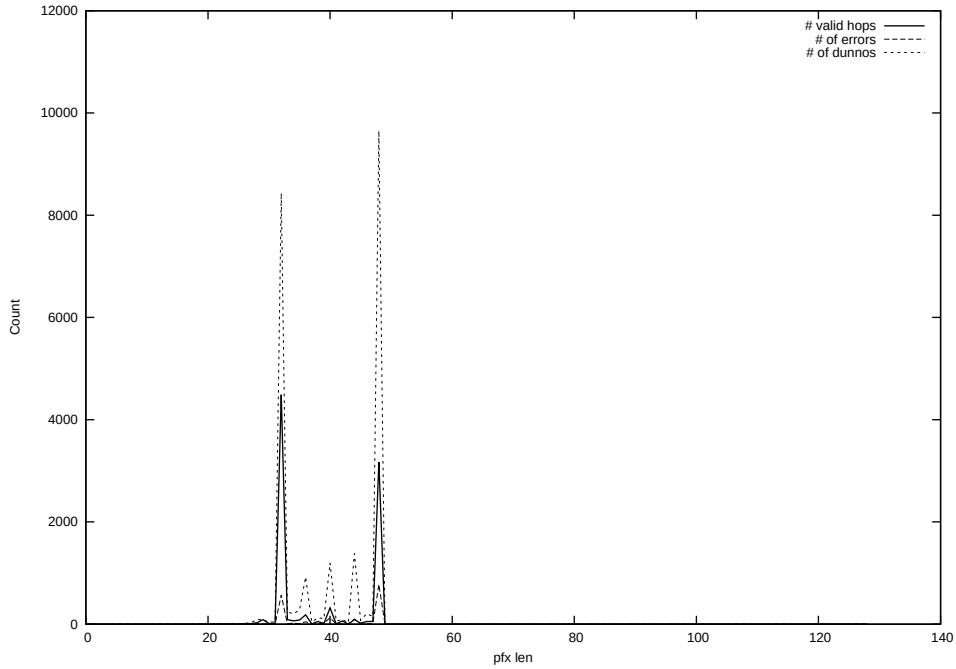
2012-11-06



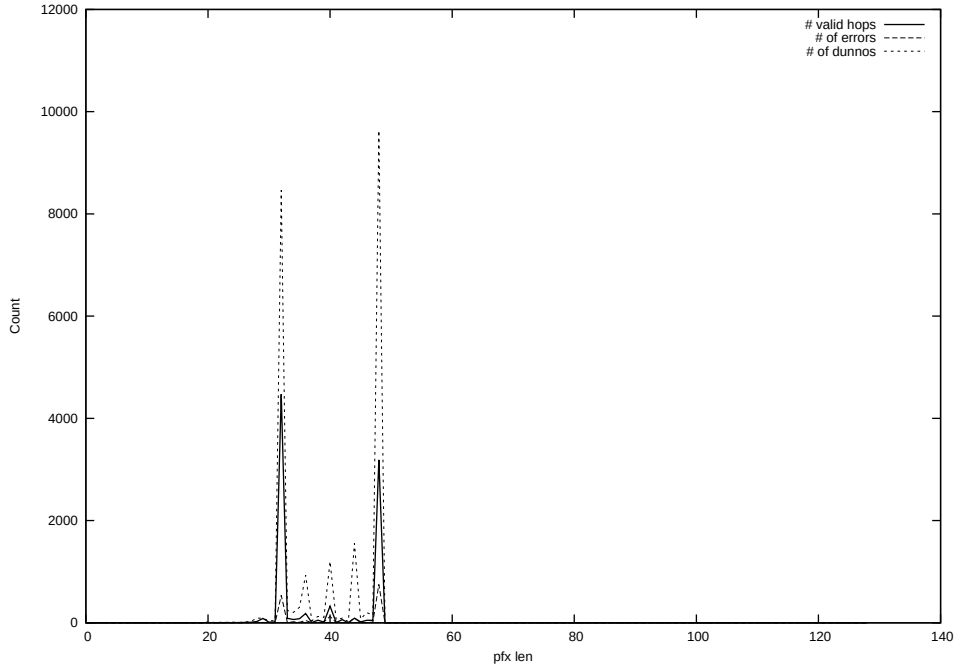
2012-11-07



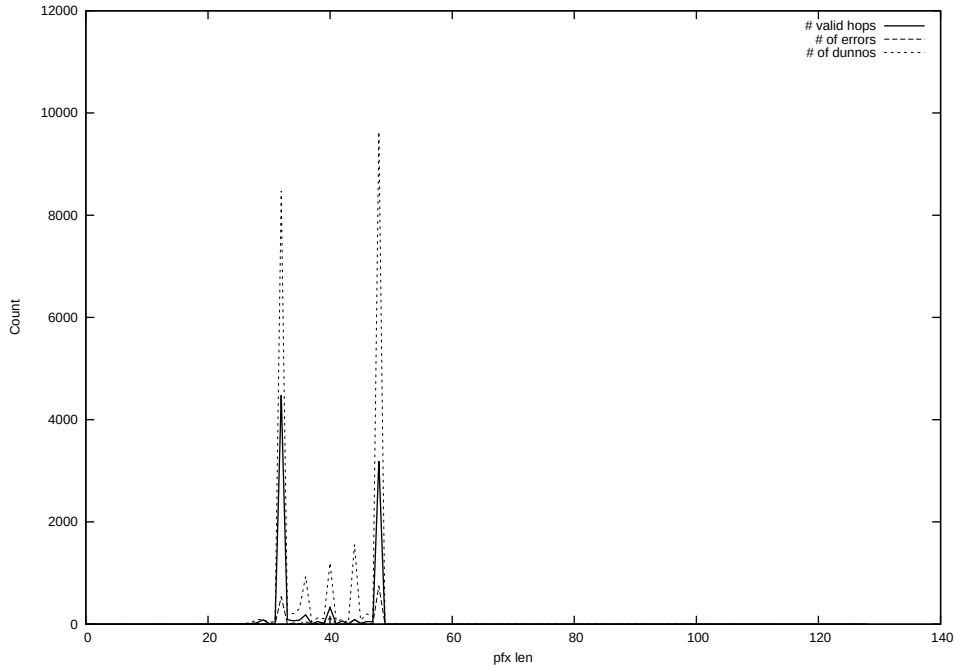
2012-11-08



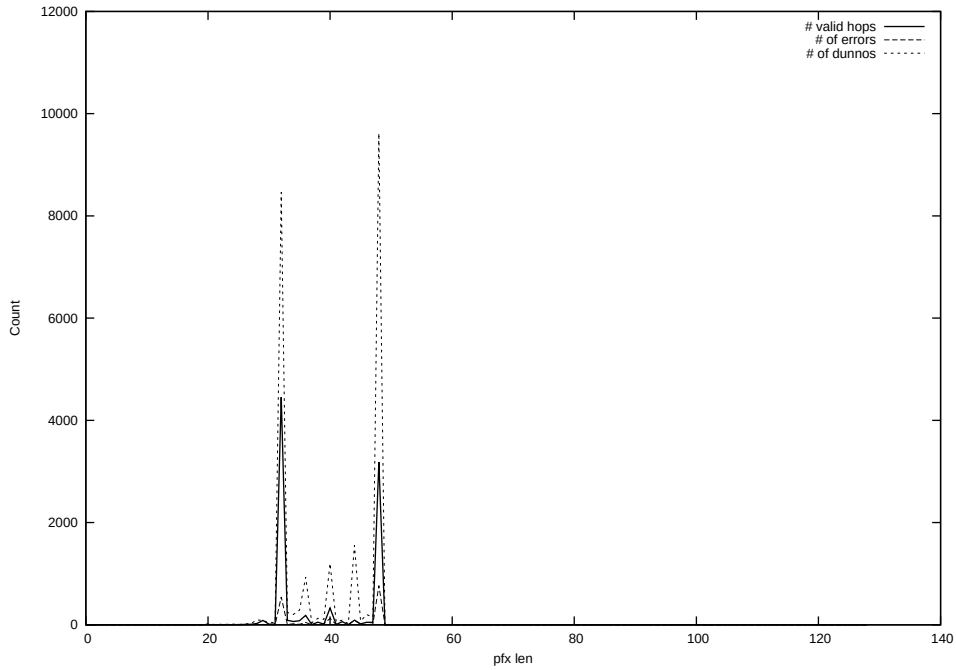
2012-11-09



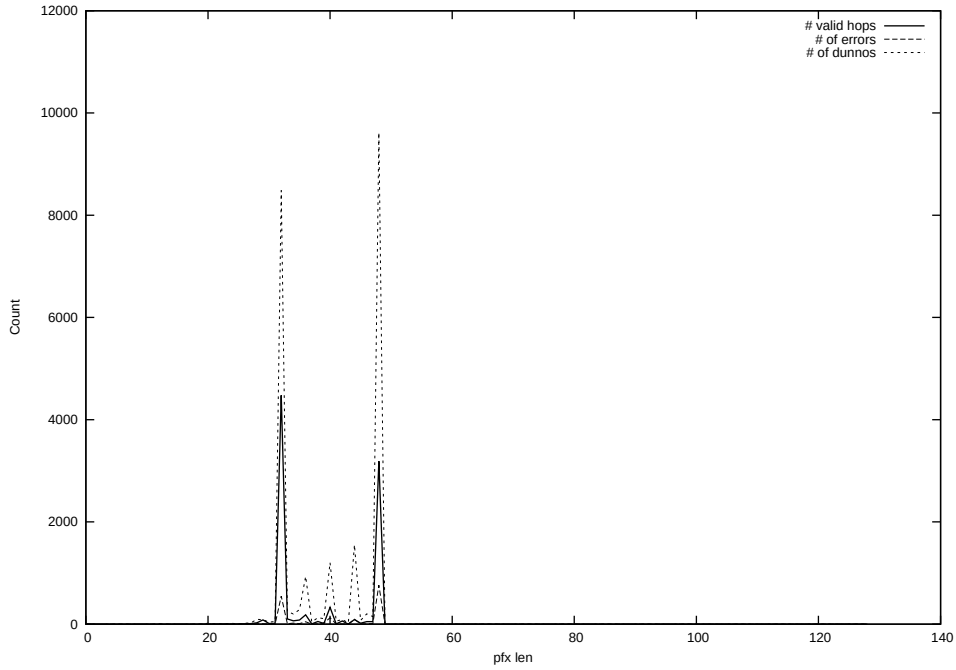
2012-11-10



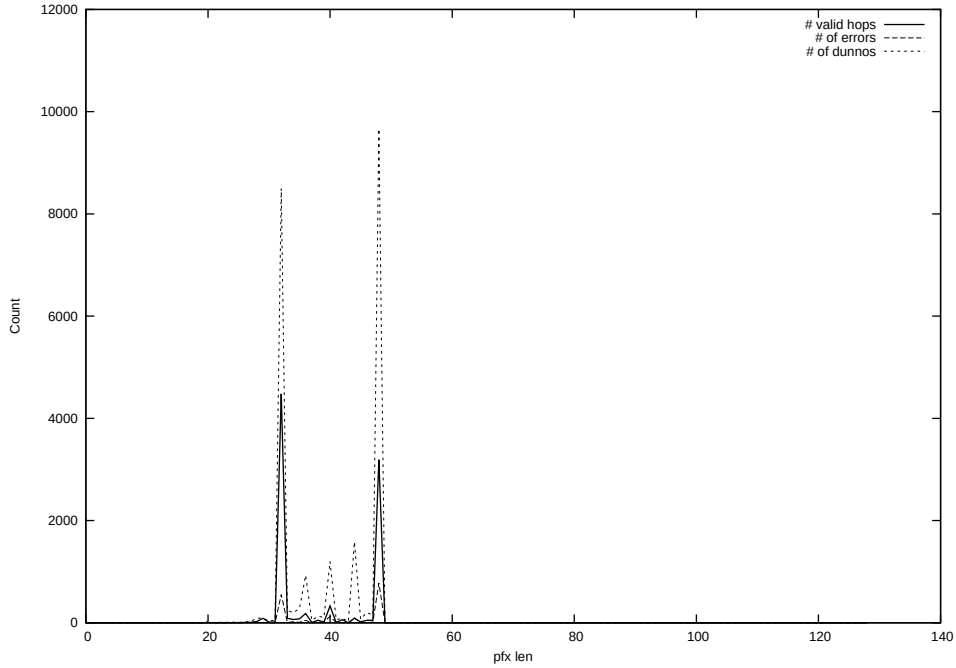
2012-11-11



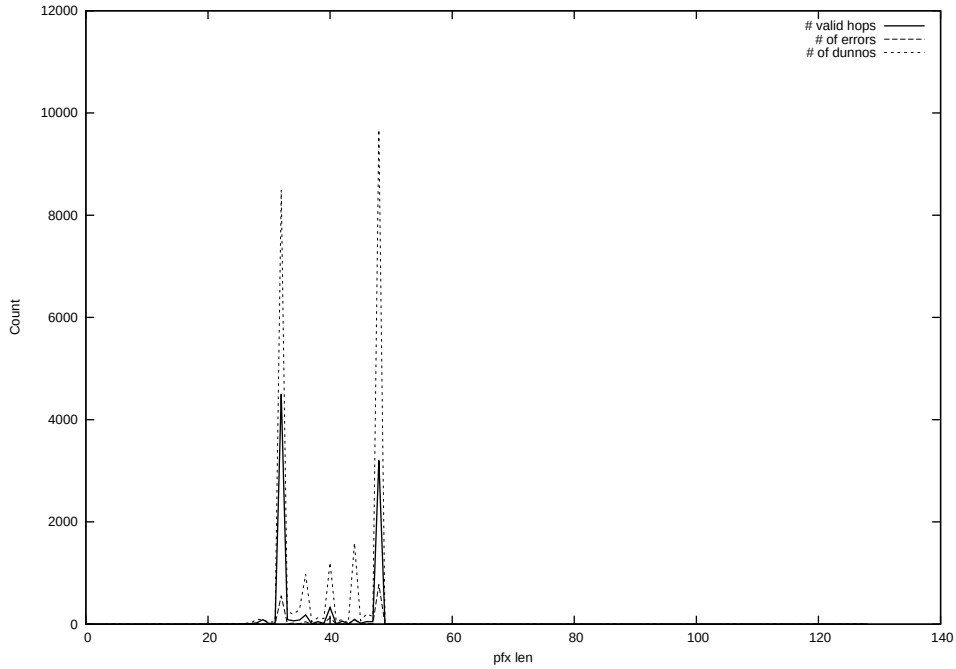
2012-11-12



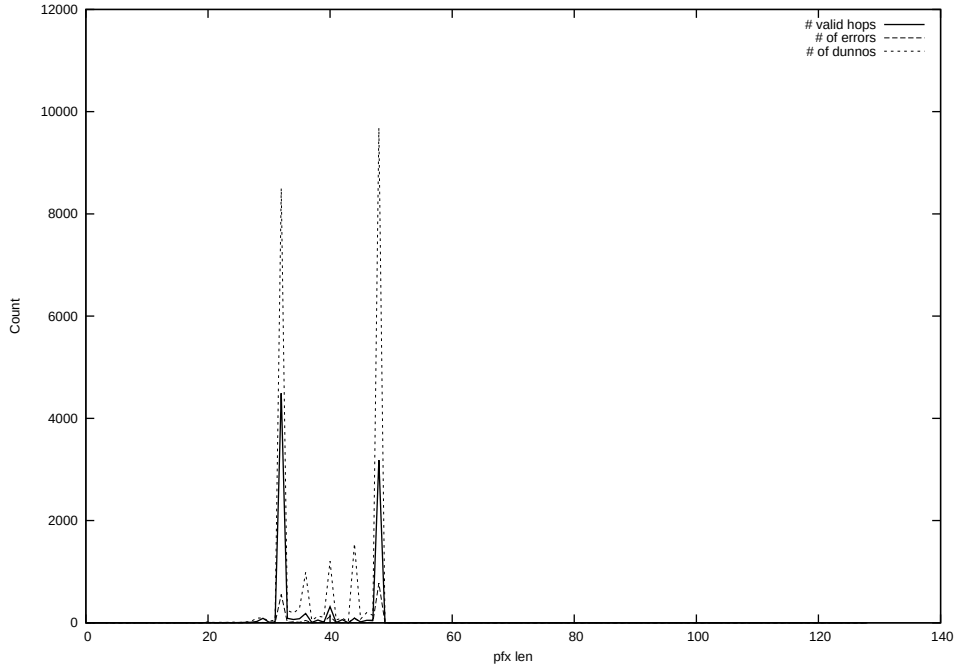
2012-11-13



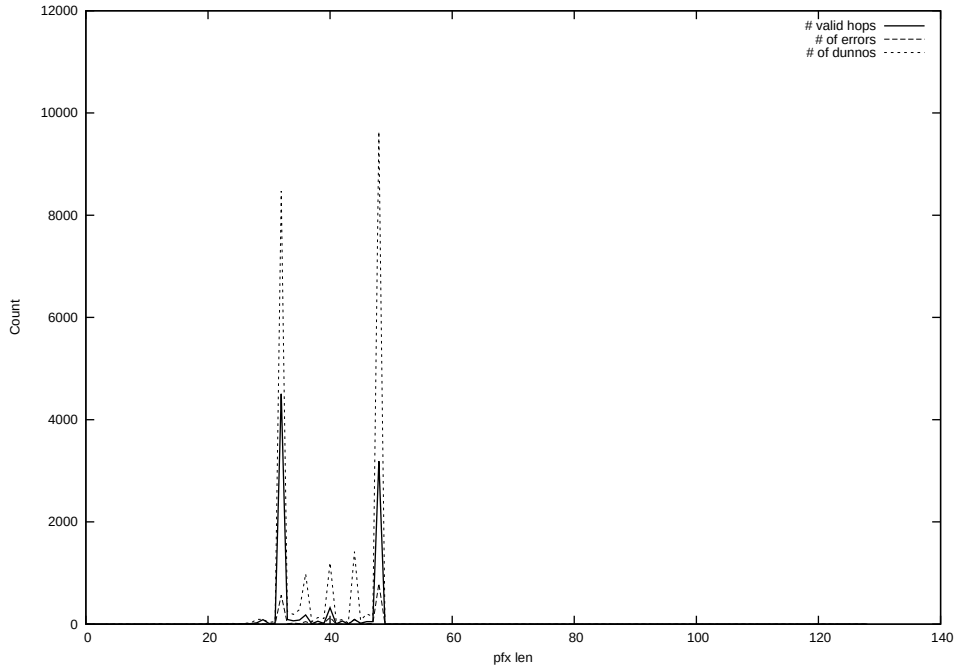
2012-11-14



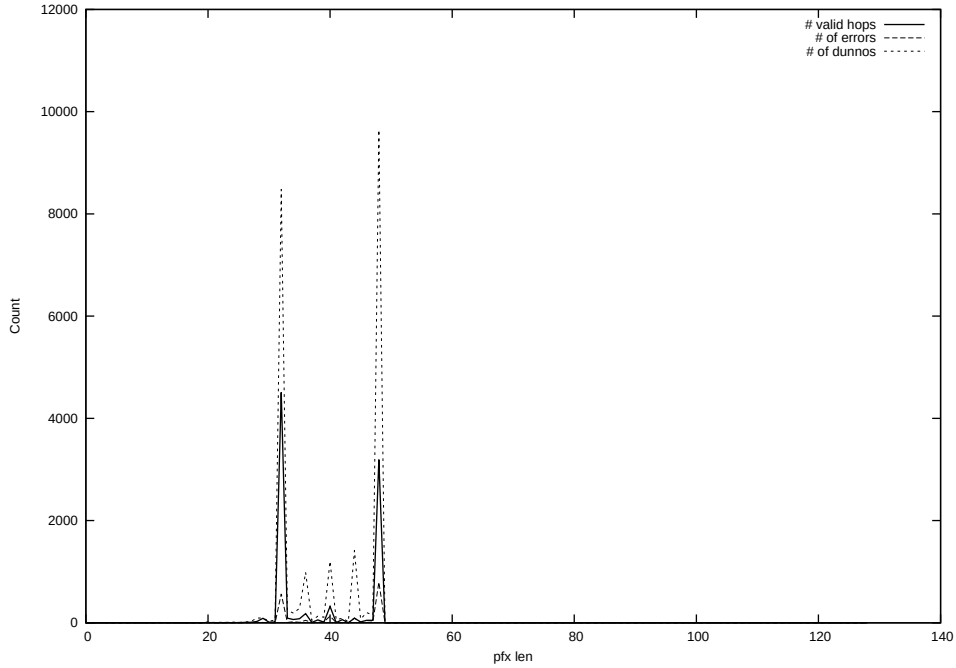
2012-11-15



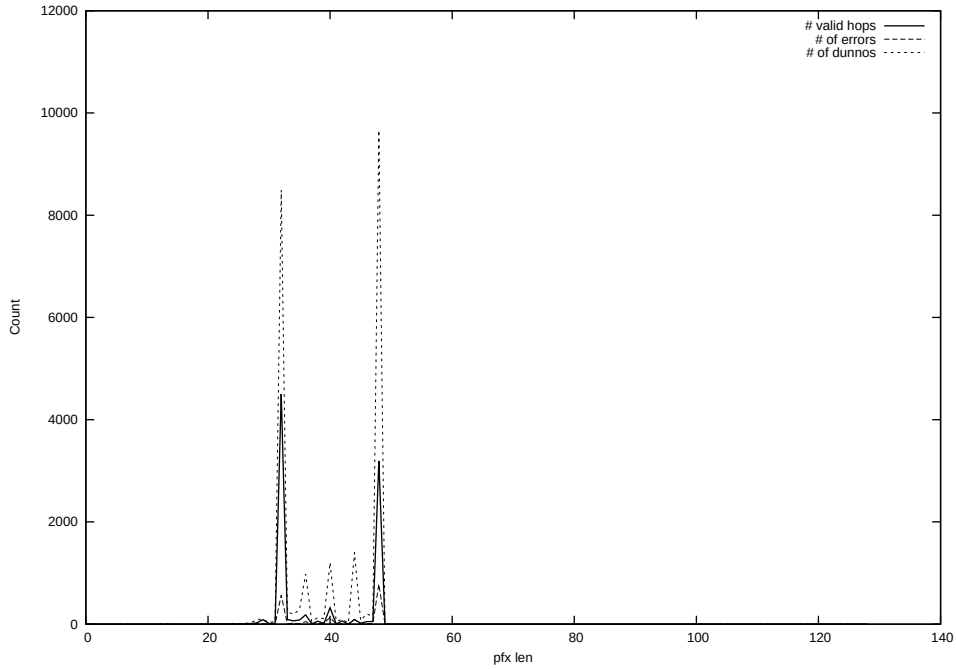
2012-11-16



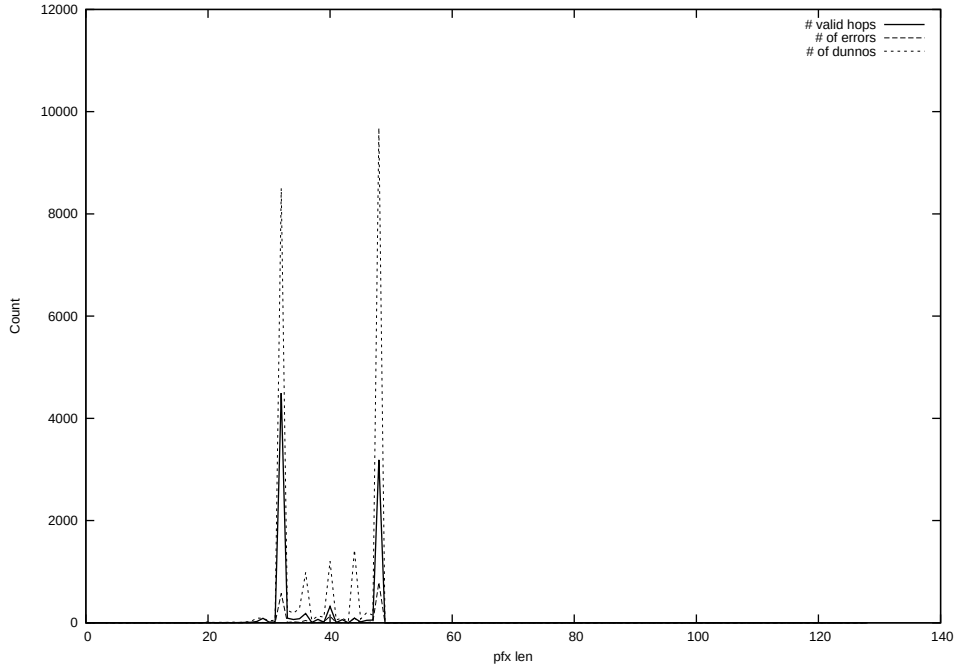
2012-11-17



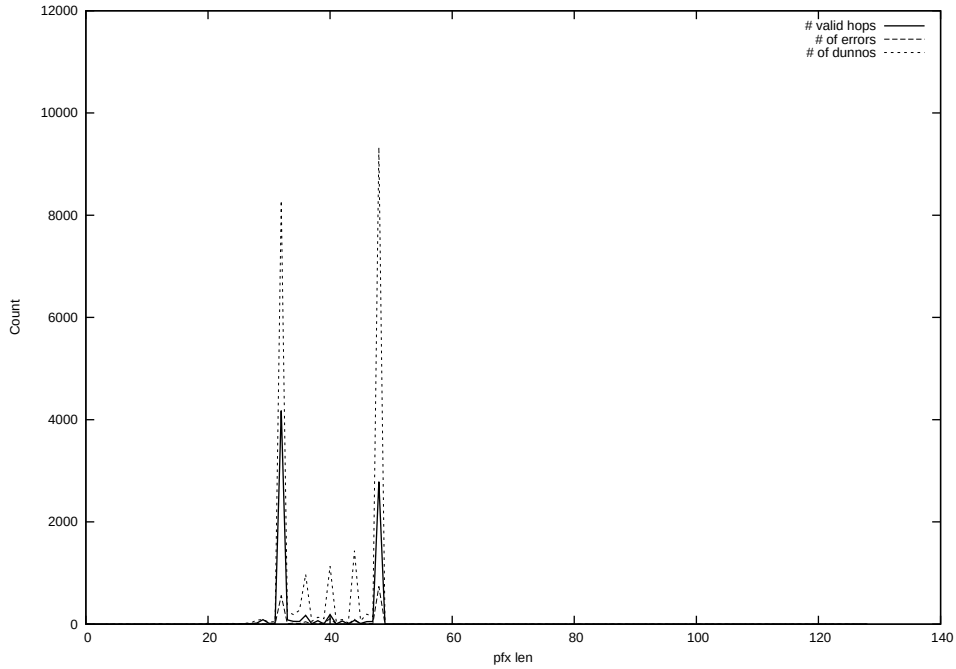
2012-11-18



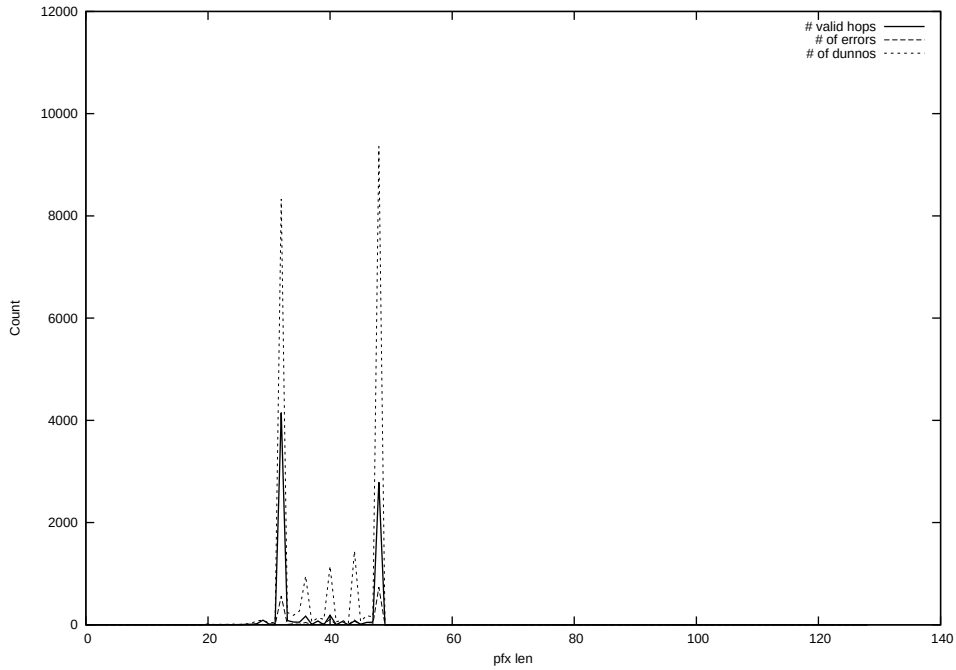
2012-11-19



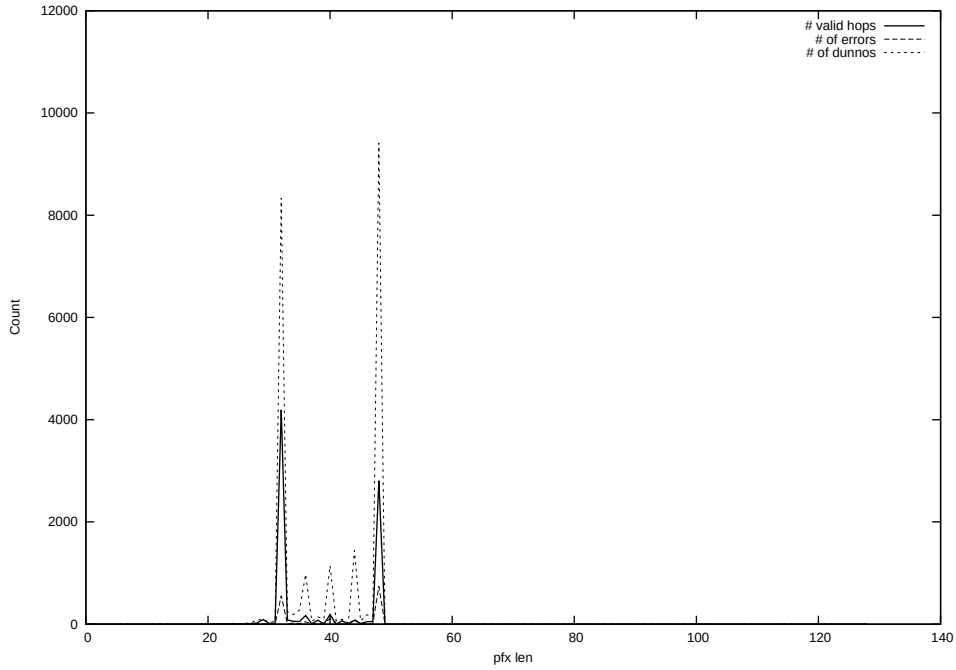
2012-11-20



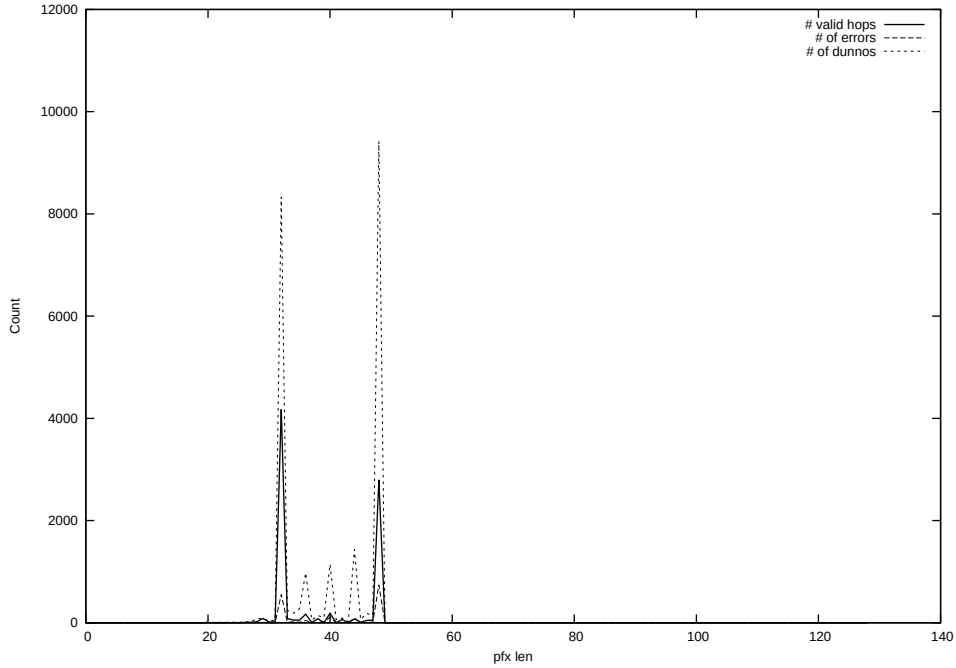
2012-11-21



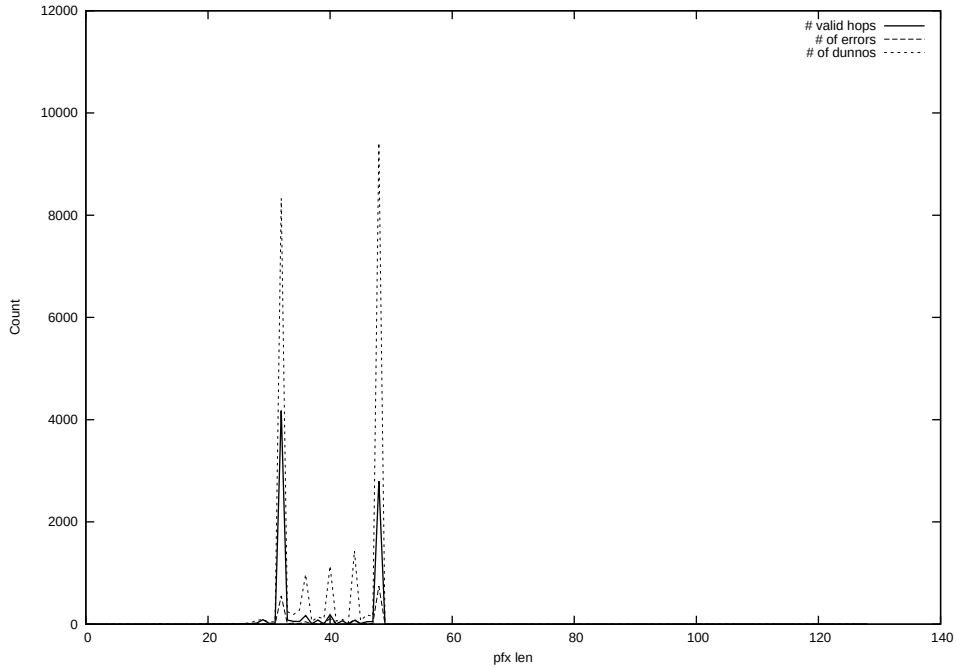
2012-11-22



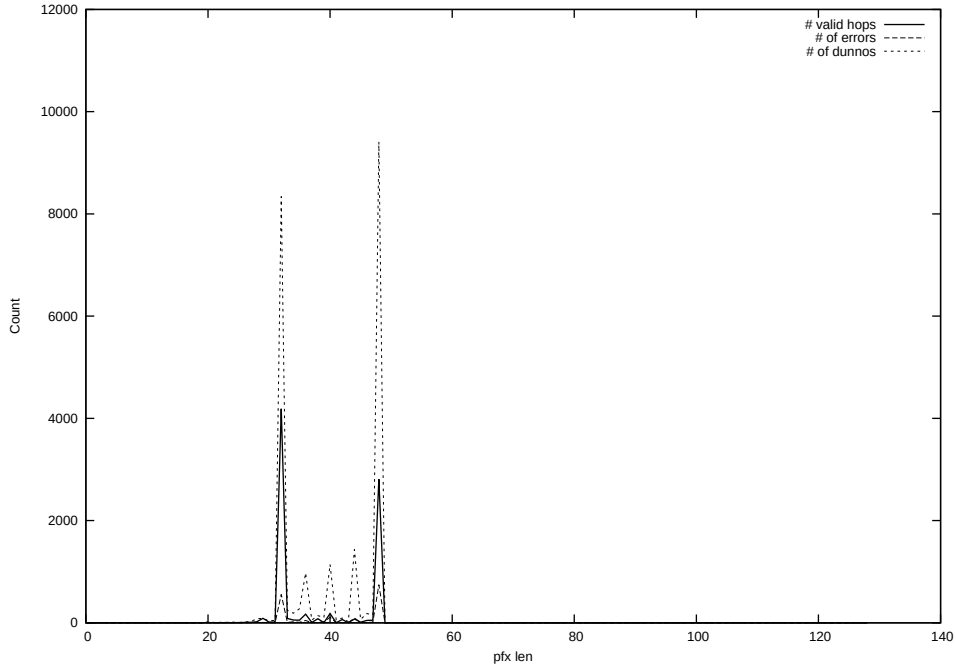
2012-11-23



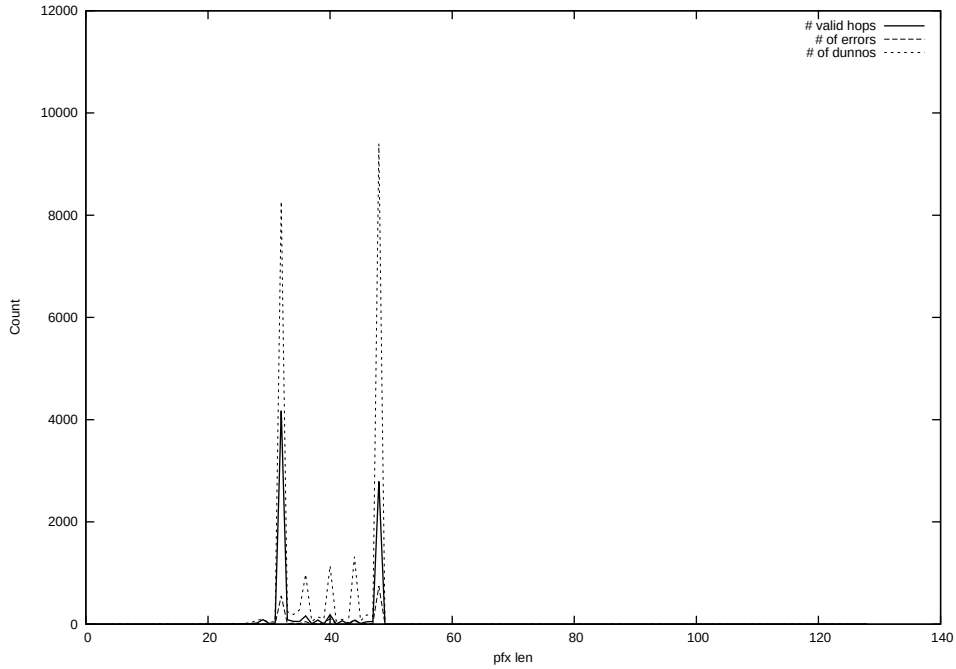
2012-11-24



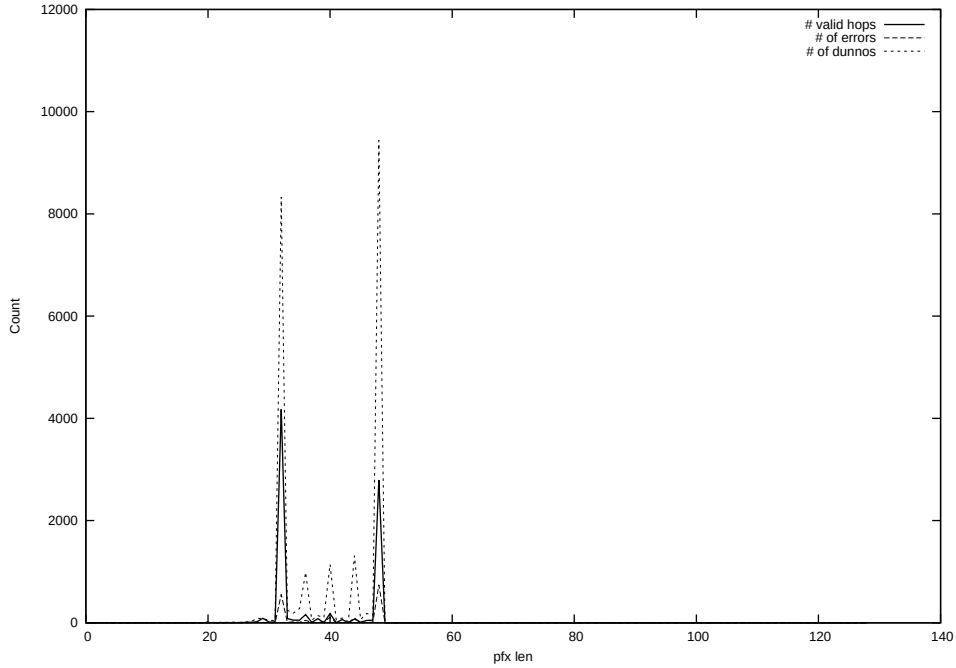
2012-11-25



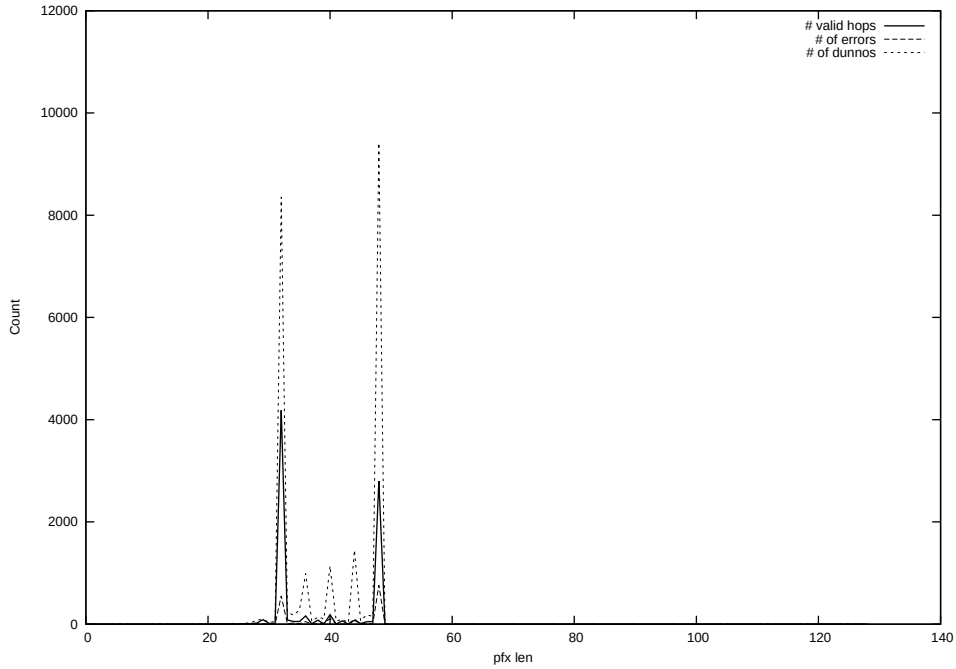
2012-11-26



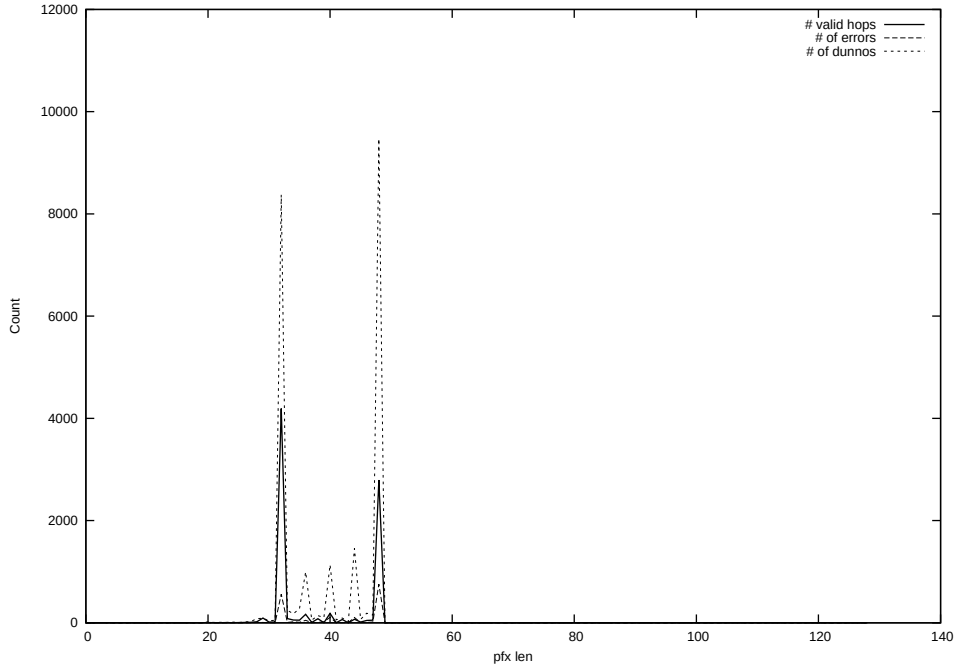
2012-11-27



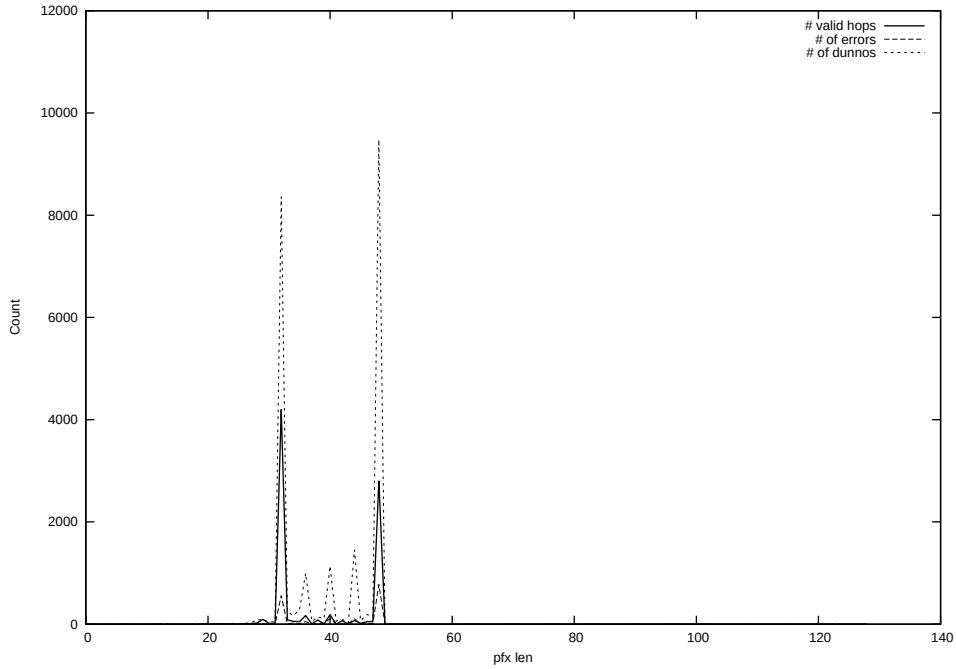
2012-11-28



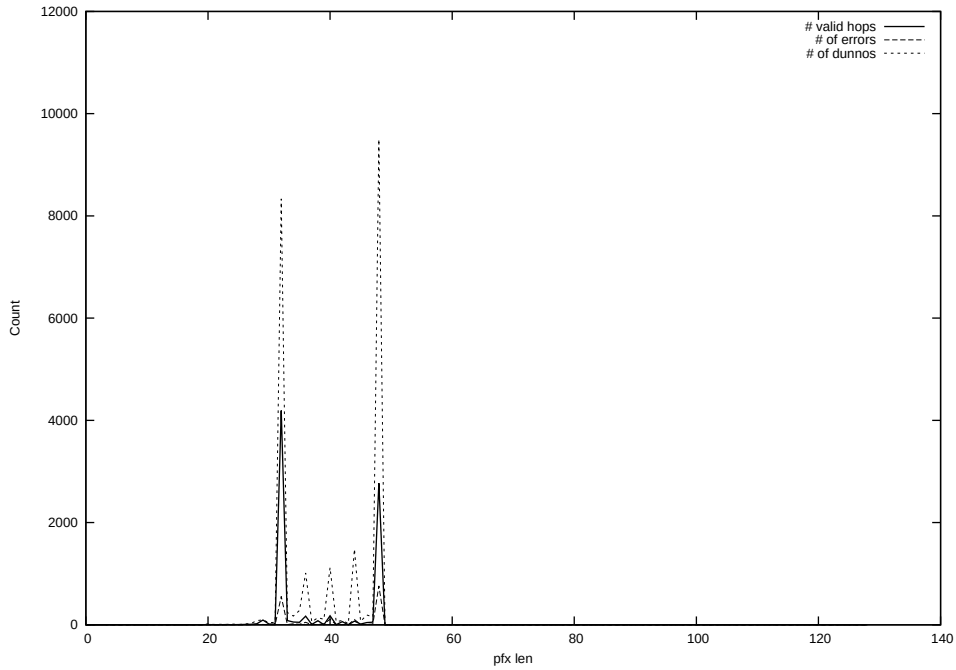
2012-11-29



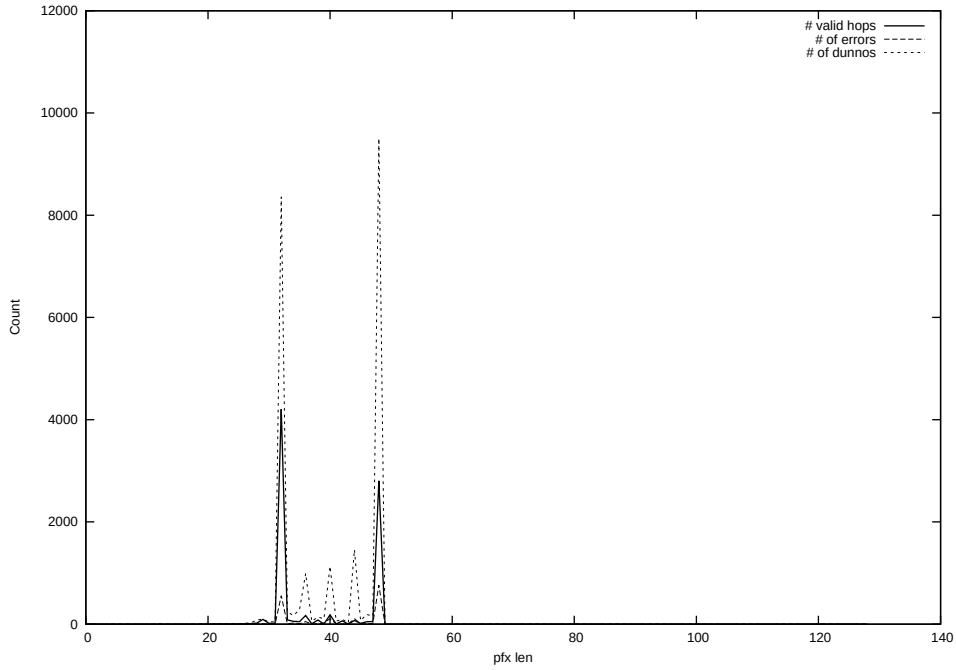
2012-11-30



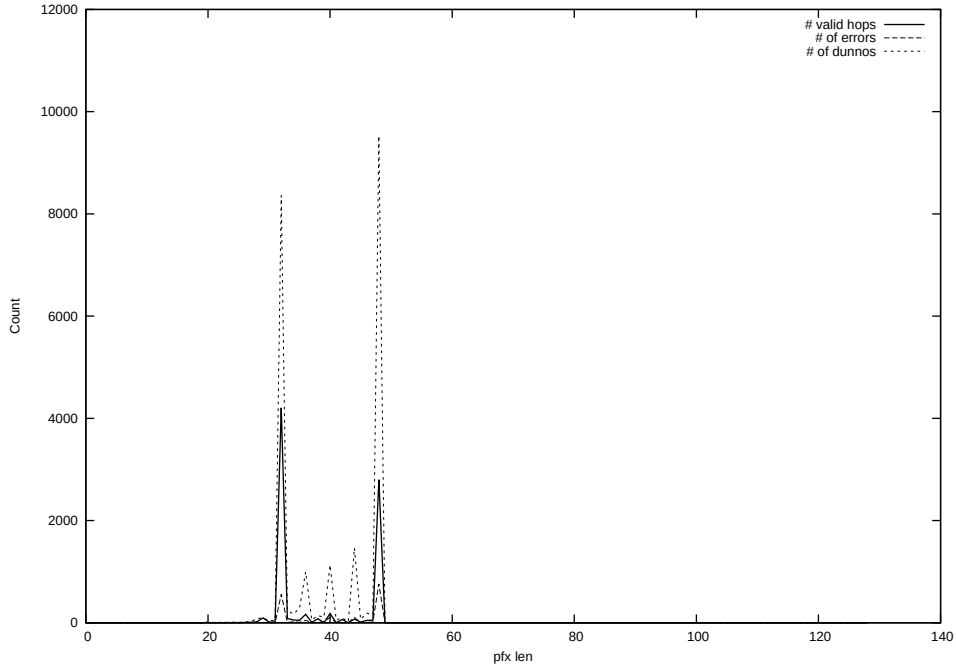
2012-12-01



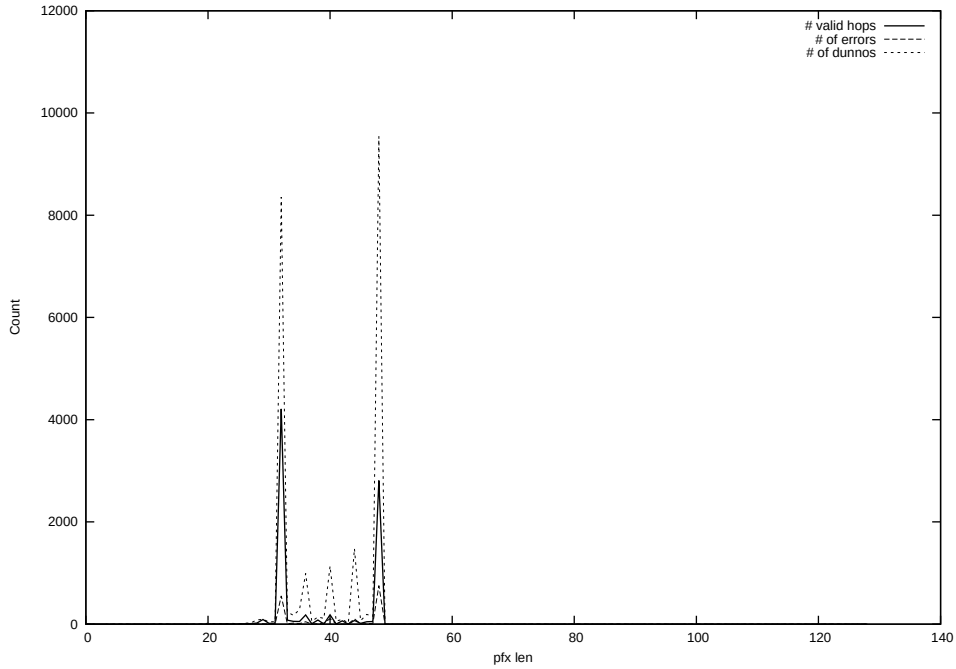
2012-12-02



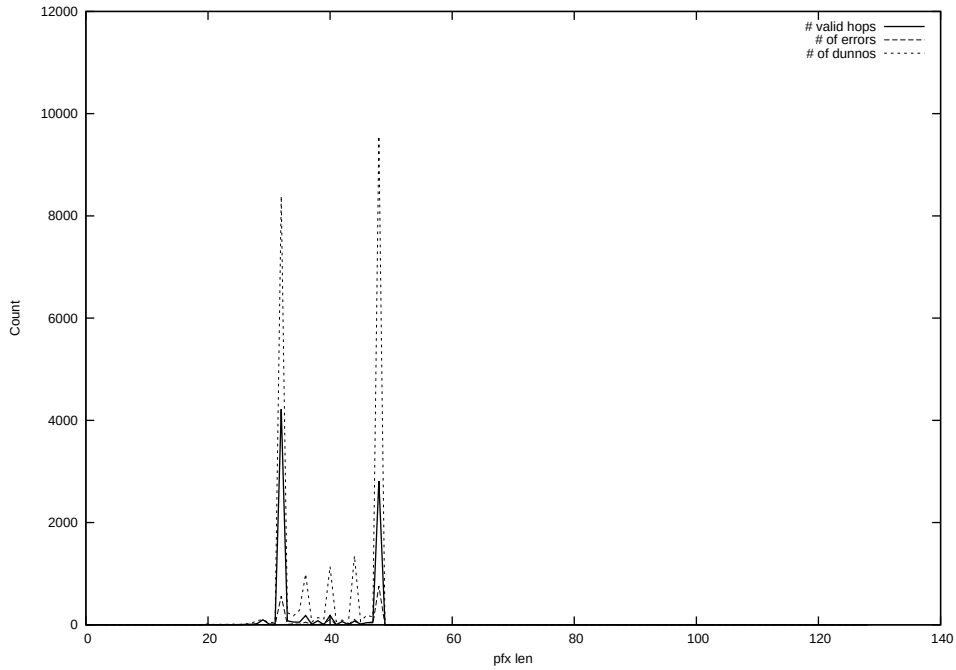
2012-12-03



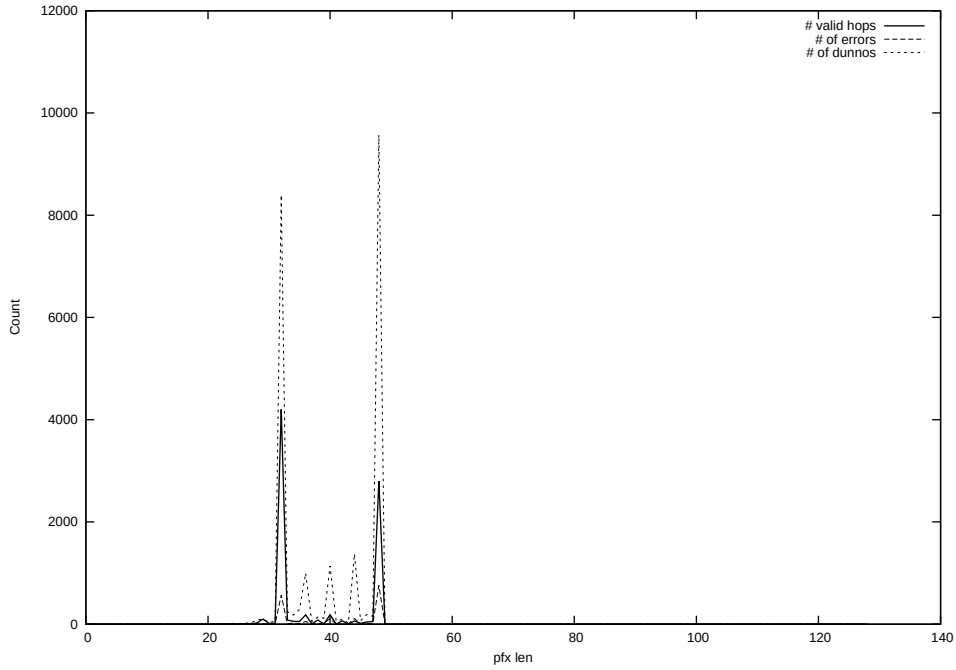
2012-12-04



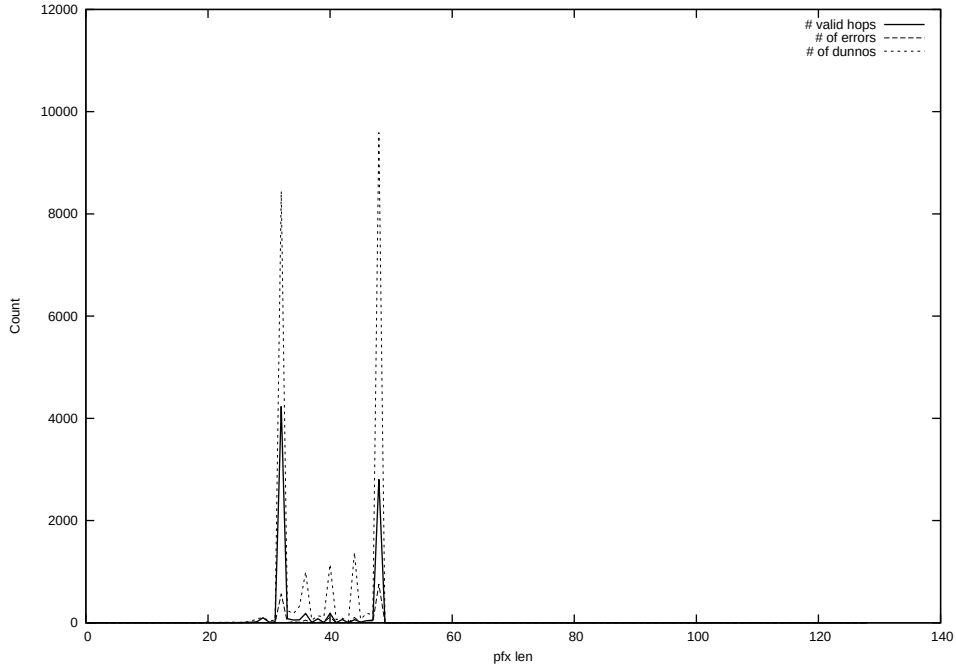
2012-12-05



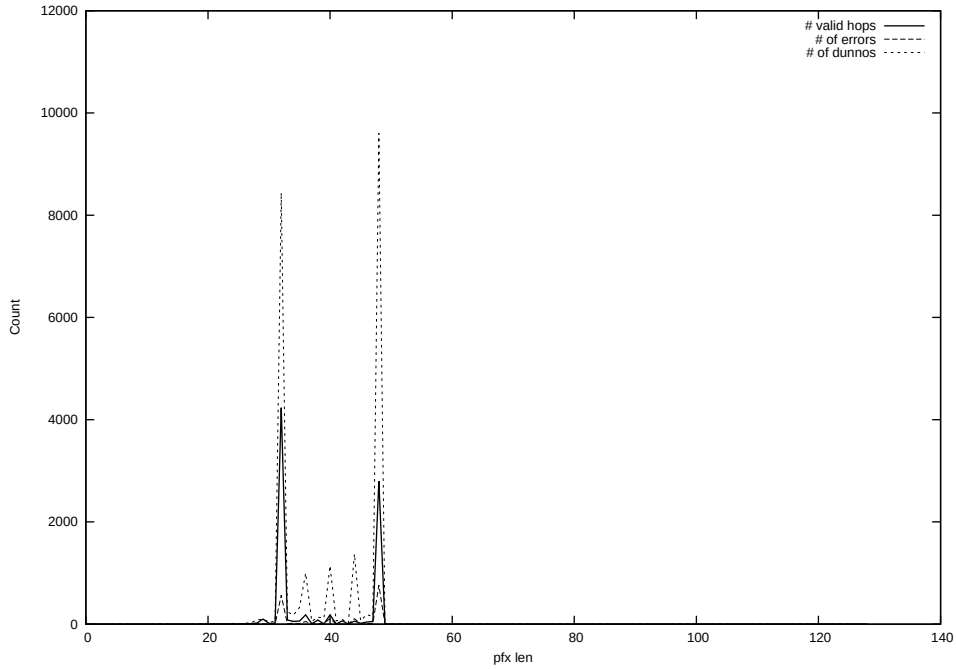
2012-12-06



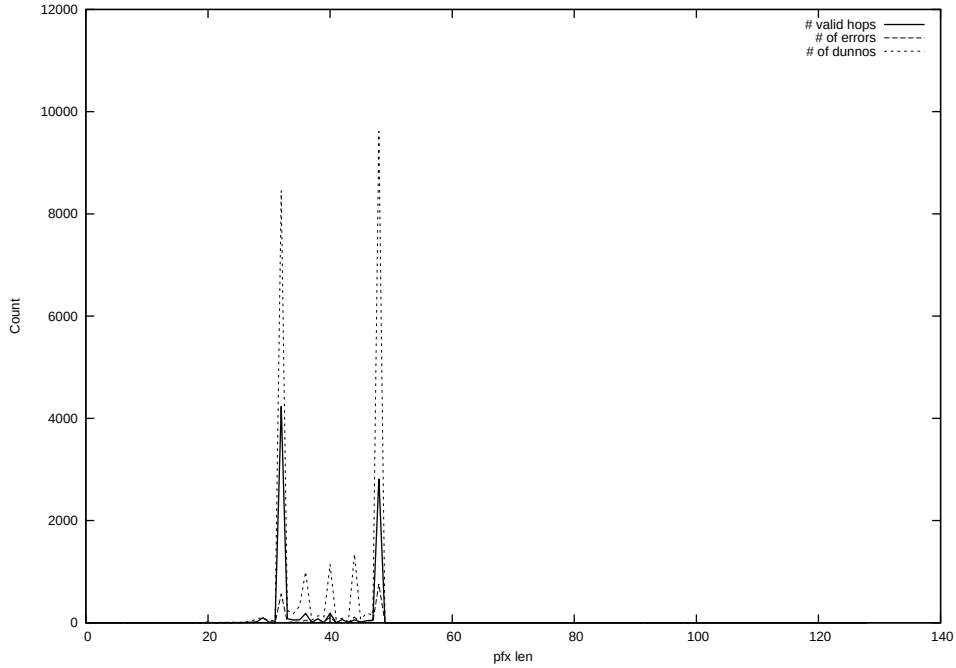
2012-12-07



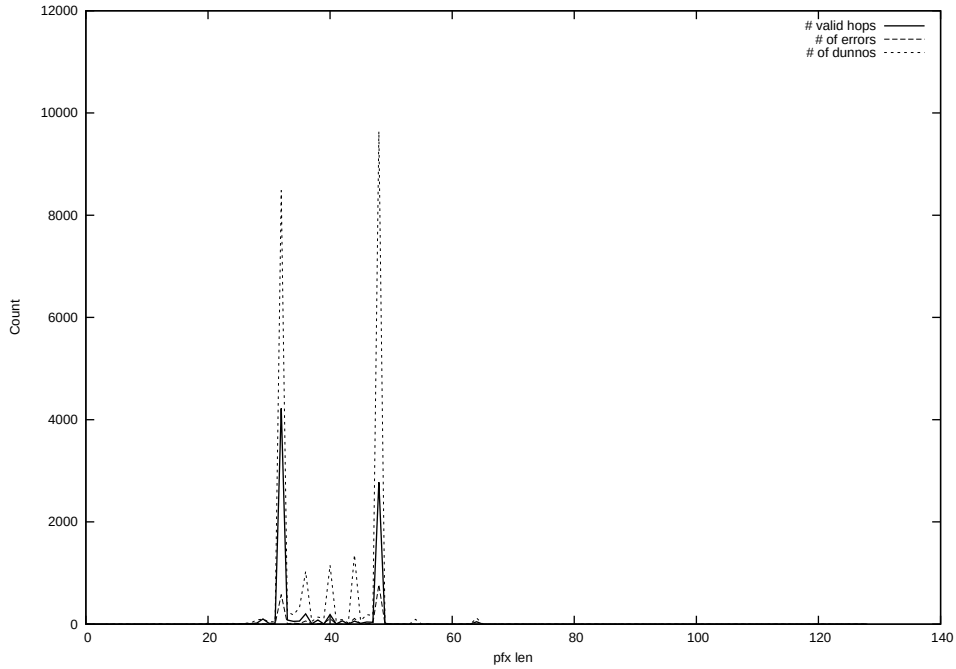
2012-12-08



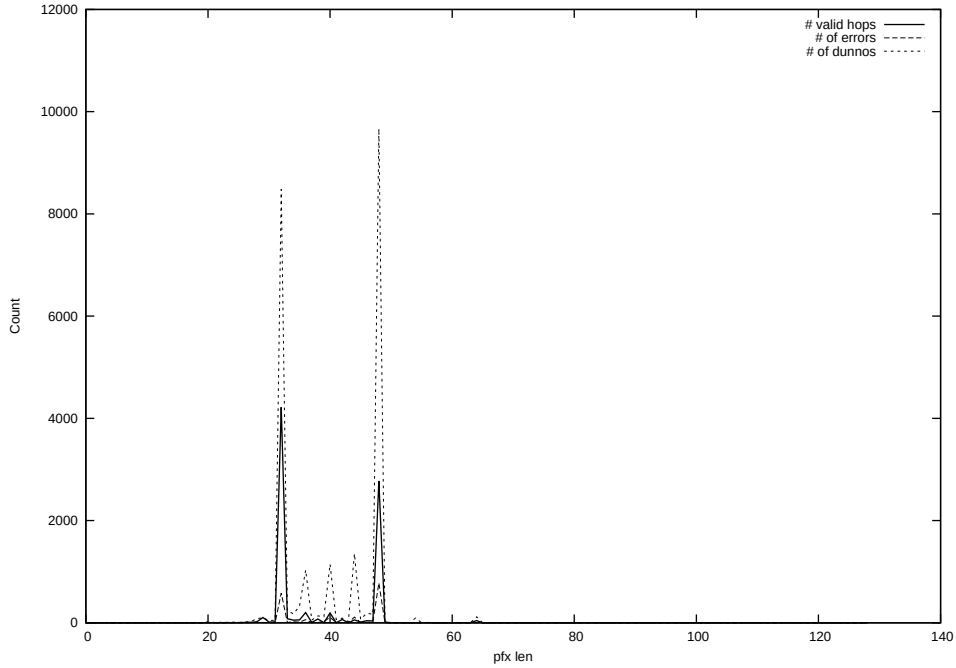
2012-12-09



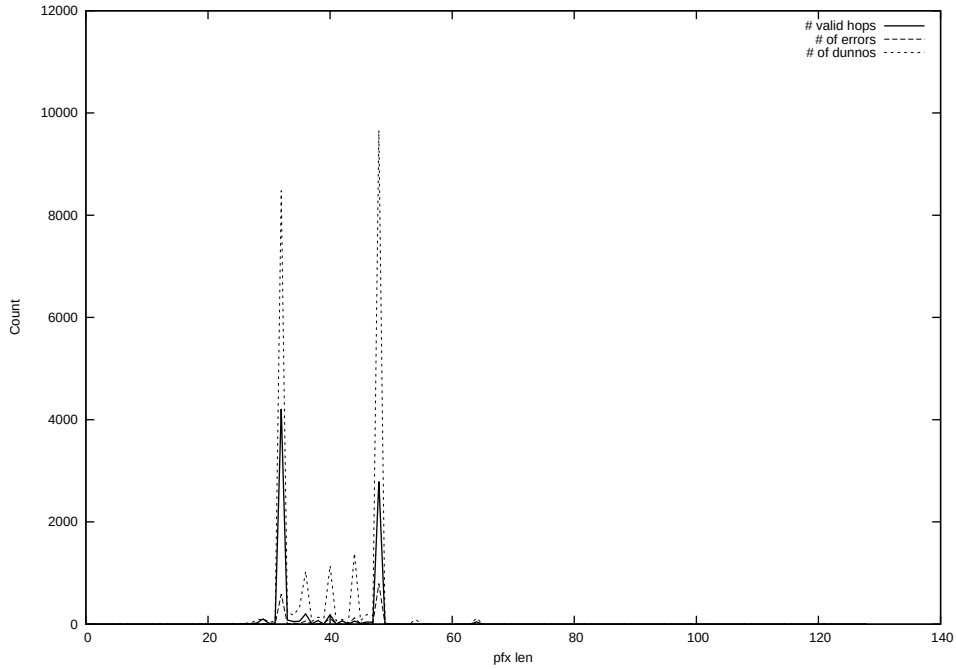
2012-12-10



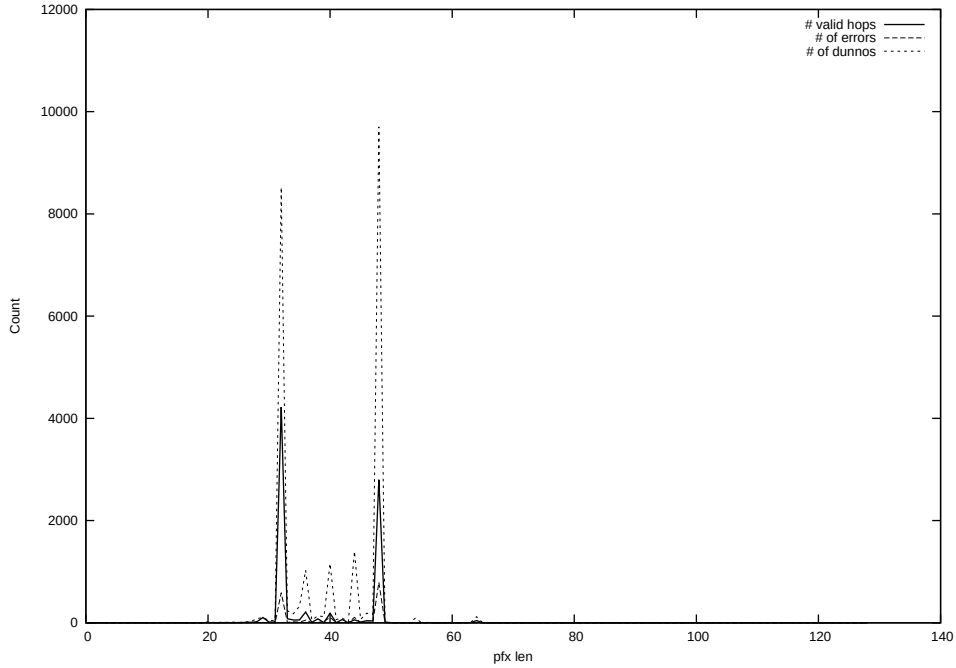
2012-12-11



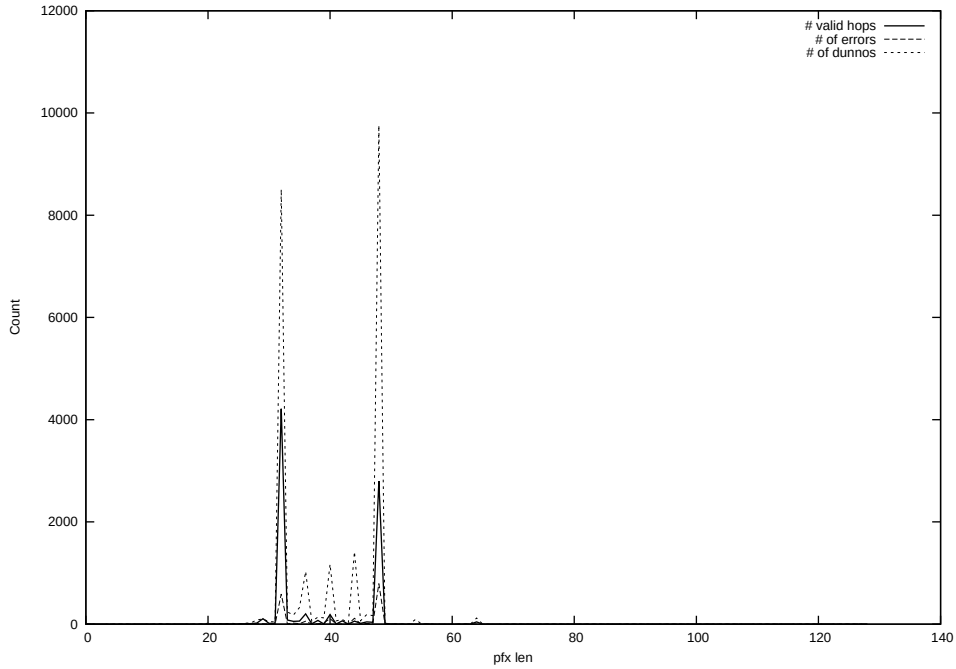
2012-12-12



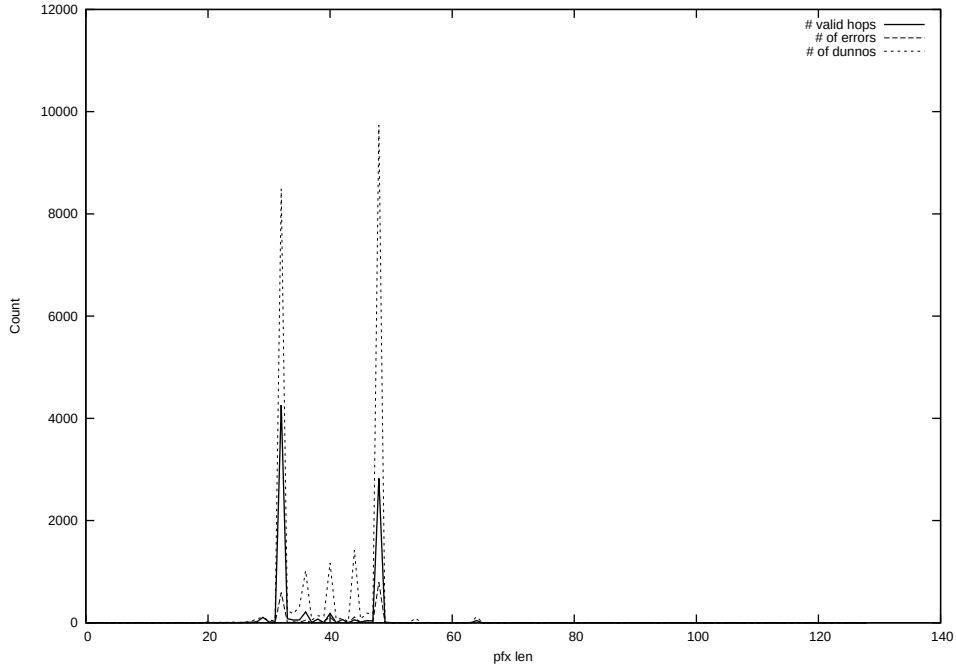
2012-12-13



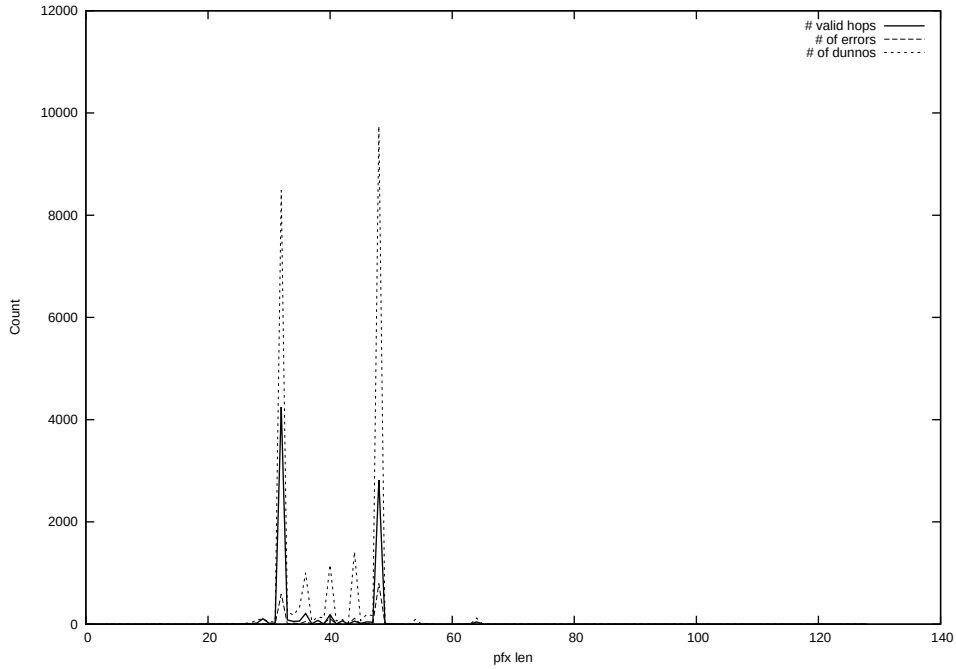
2012-12-14



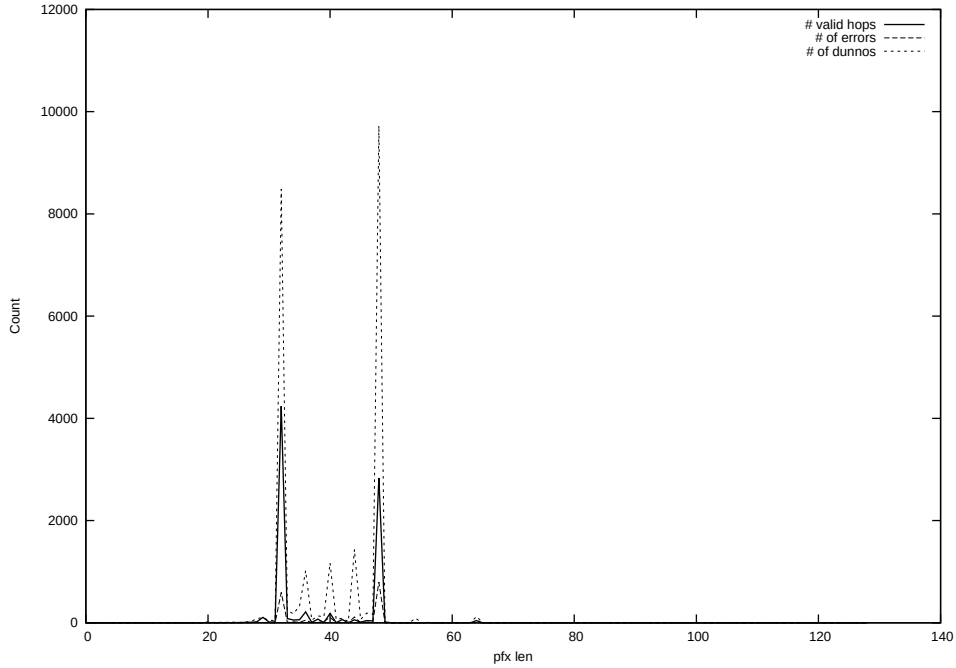
2012-12-15



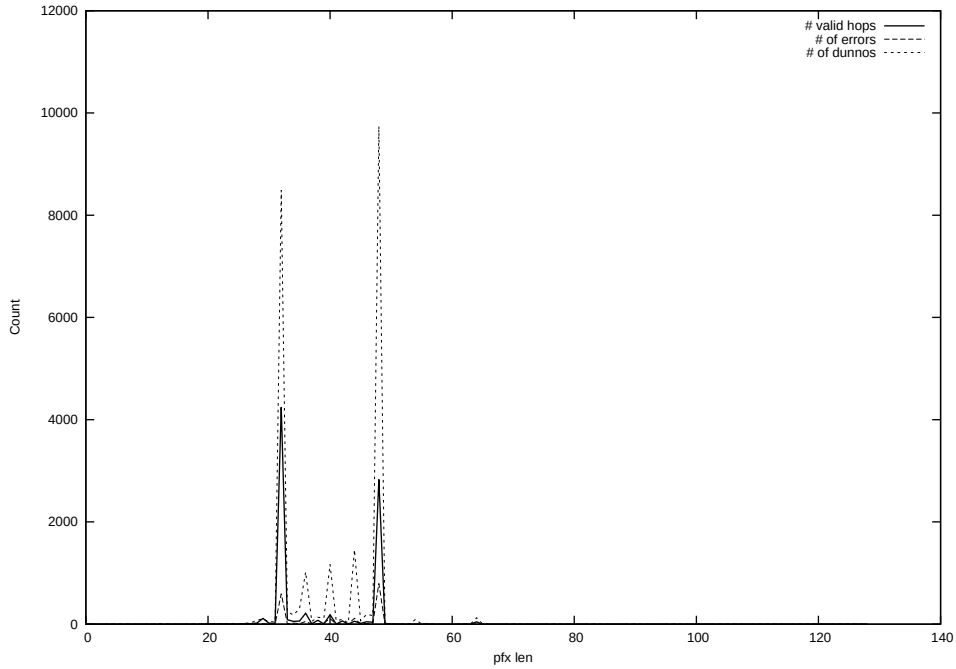
2012-12-16



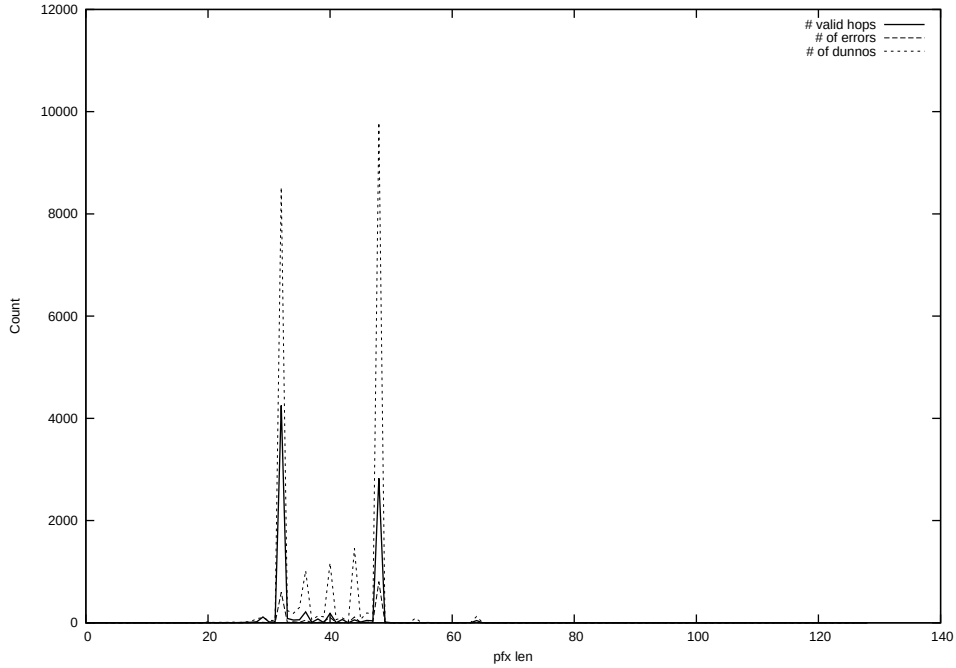
2012-12-17



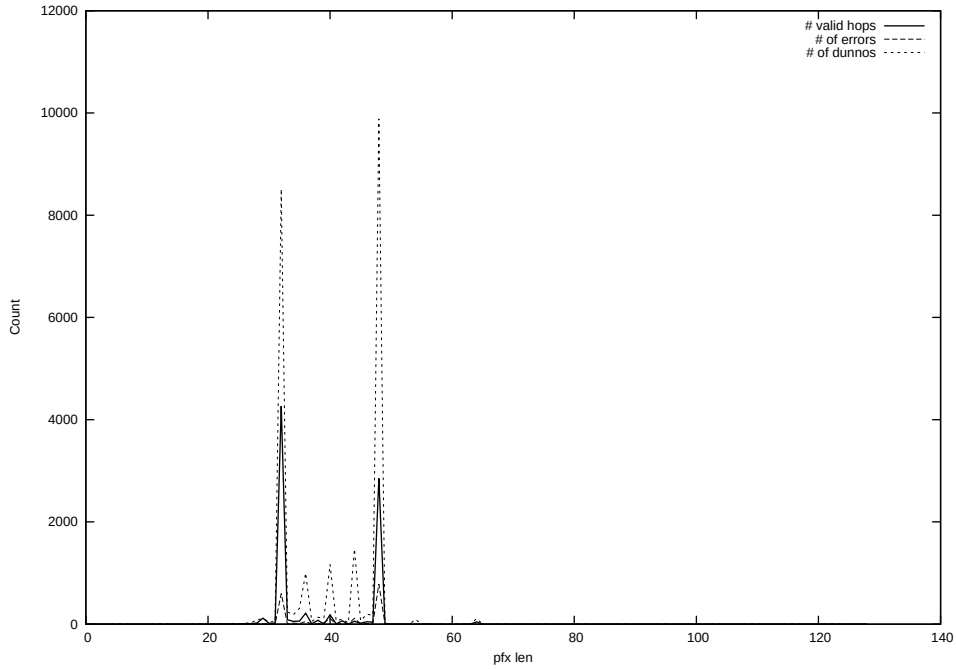
2012-12-18



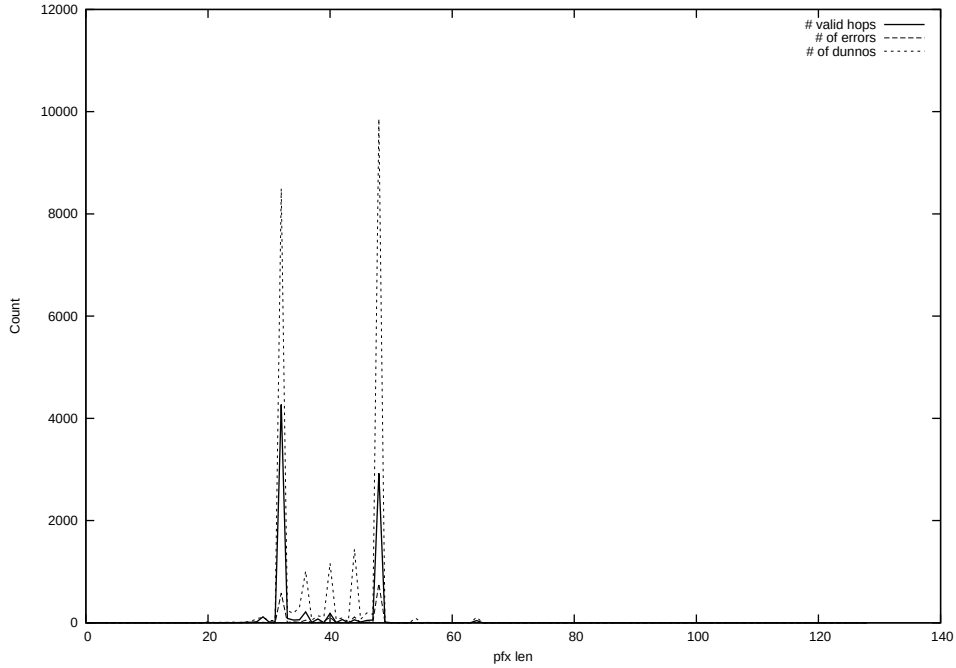
2012-12-19



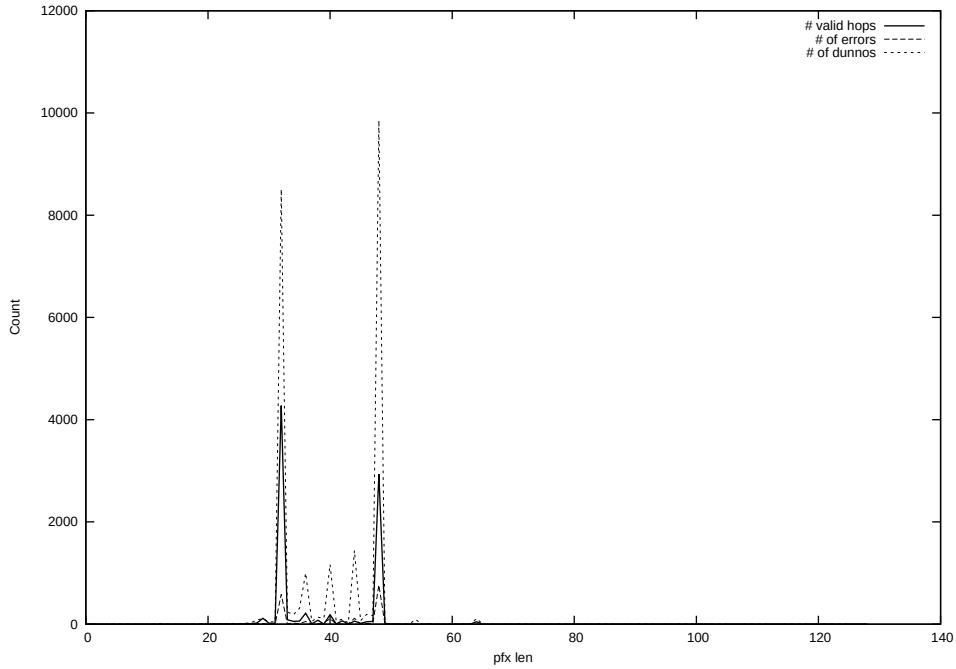
2012-12-20



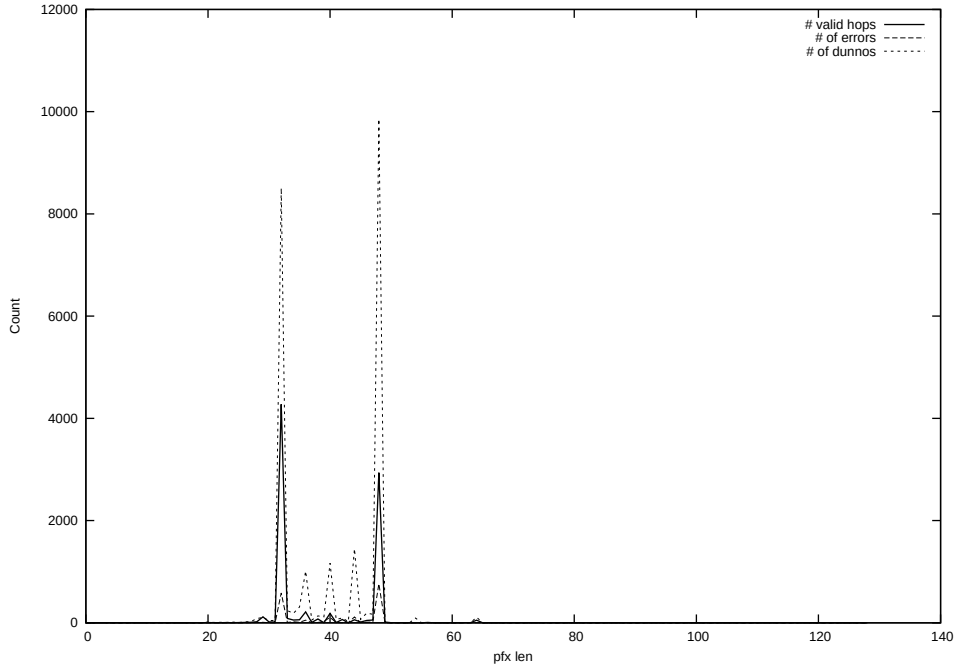
2012-12-21



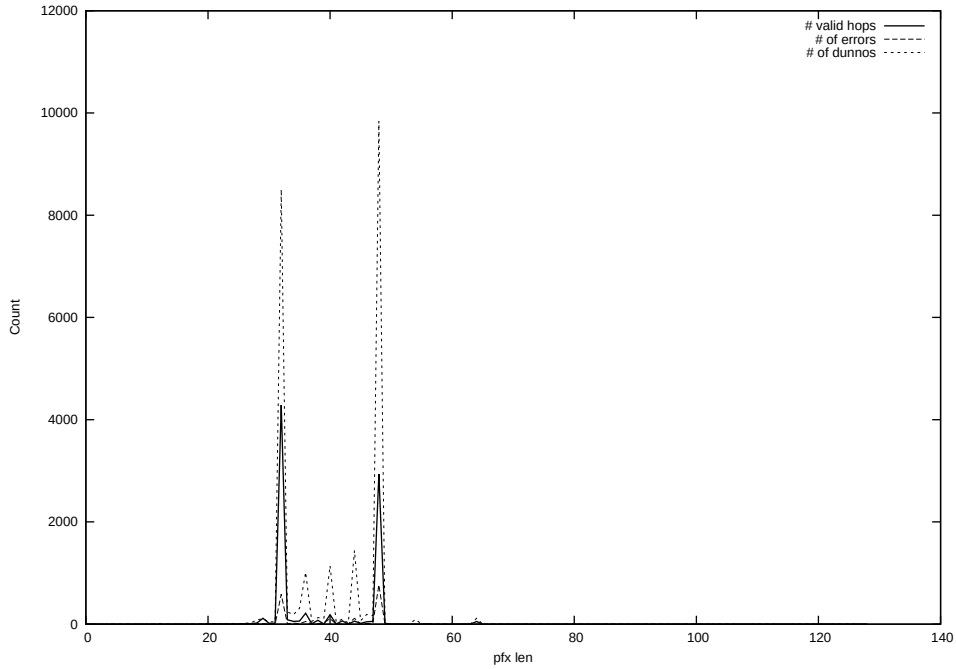
2012-12-22



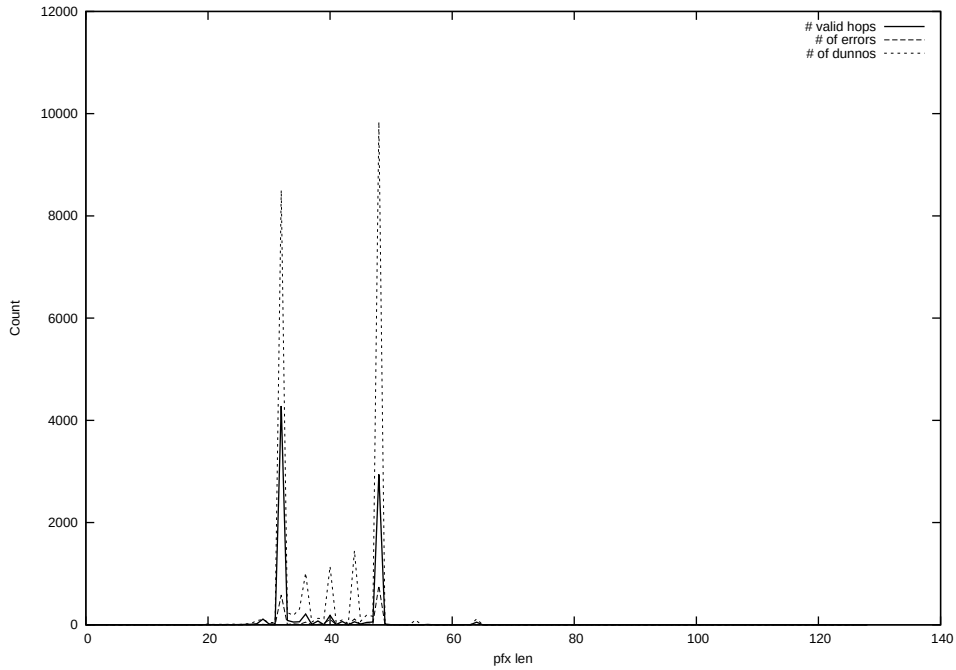
2012-12-23



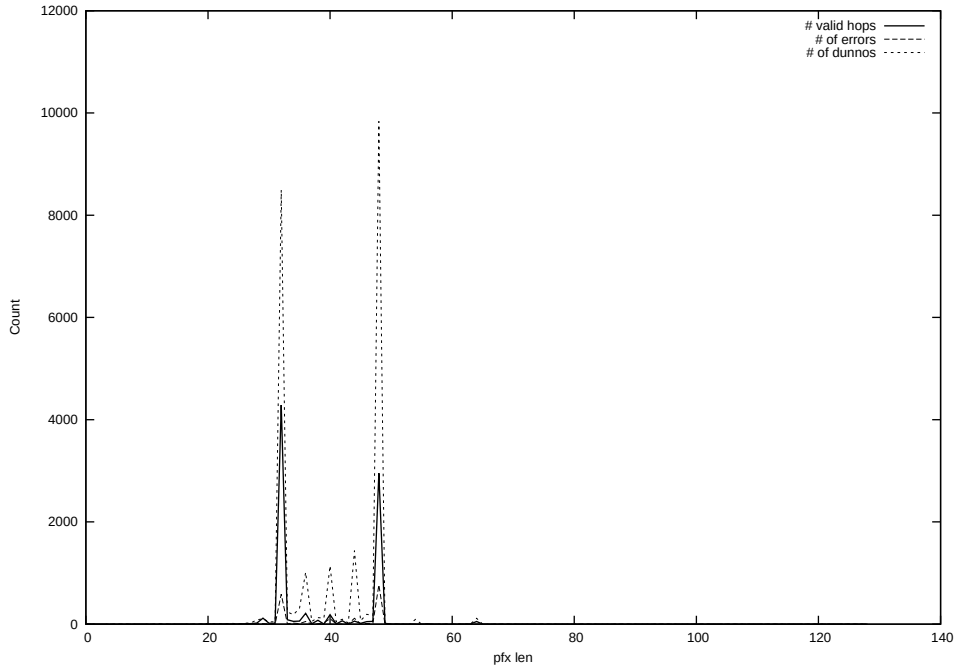
2012-12-24



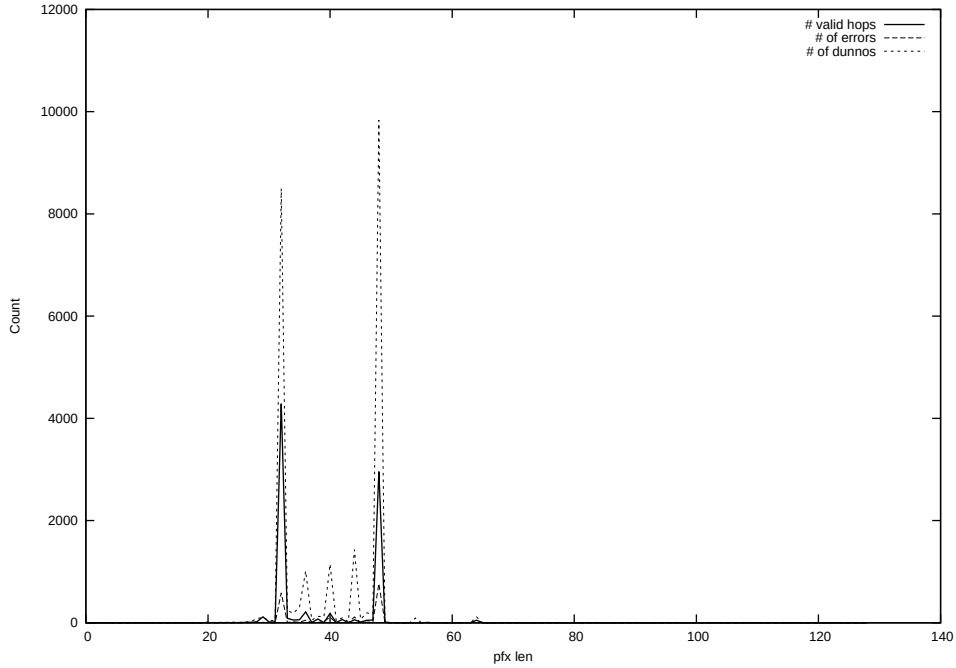
2012-12-25



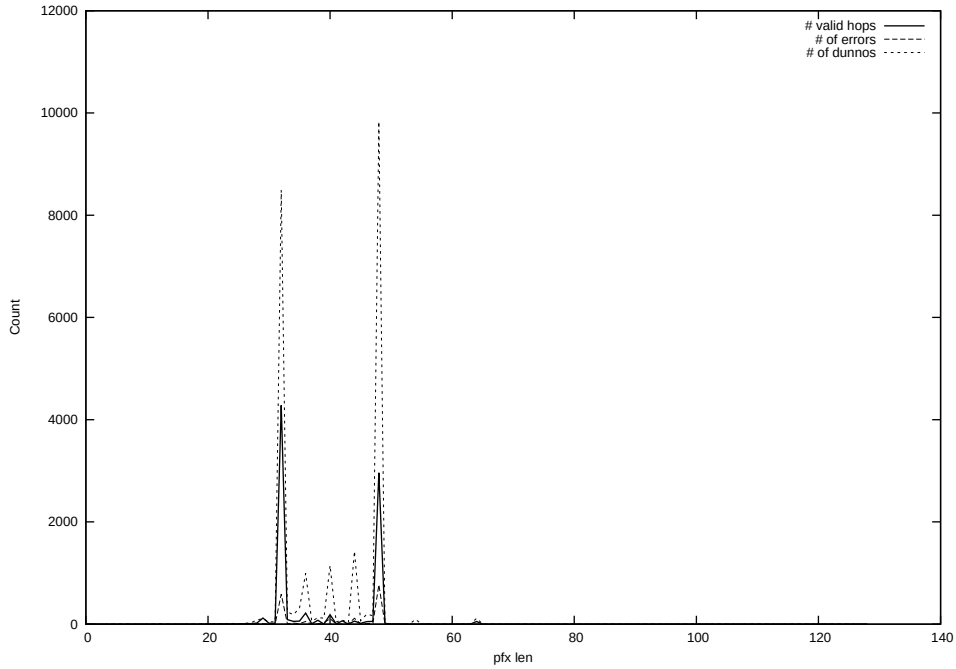
2012-12-26



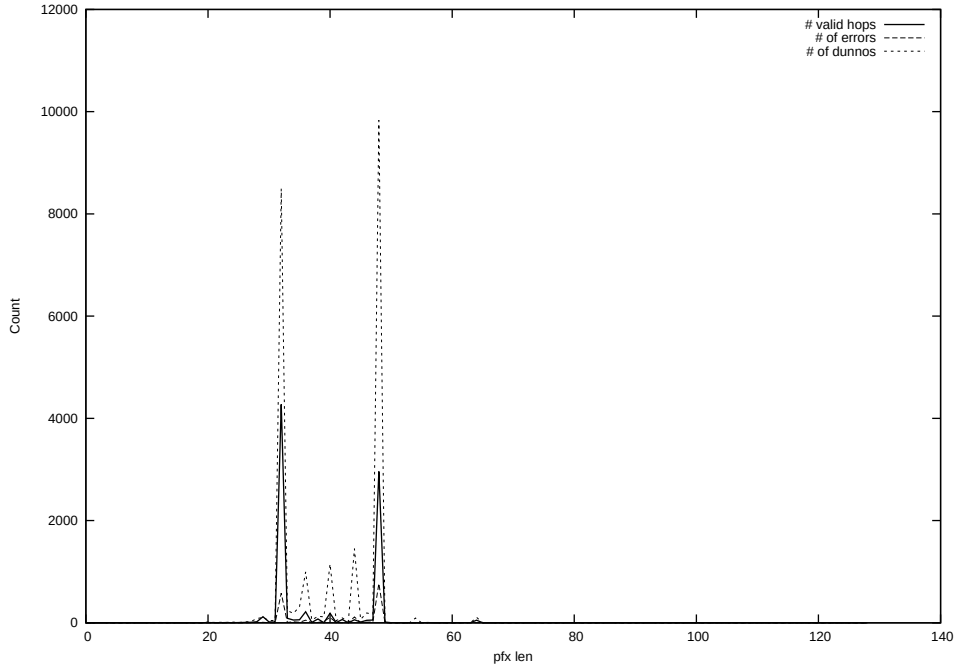
2012-12-27



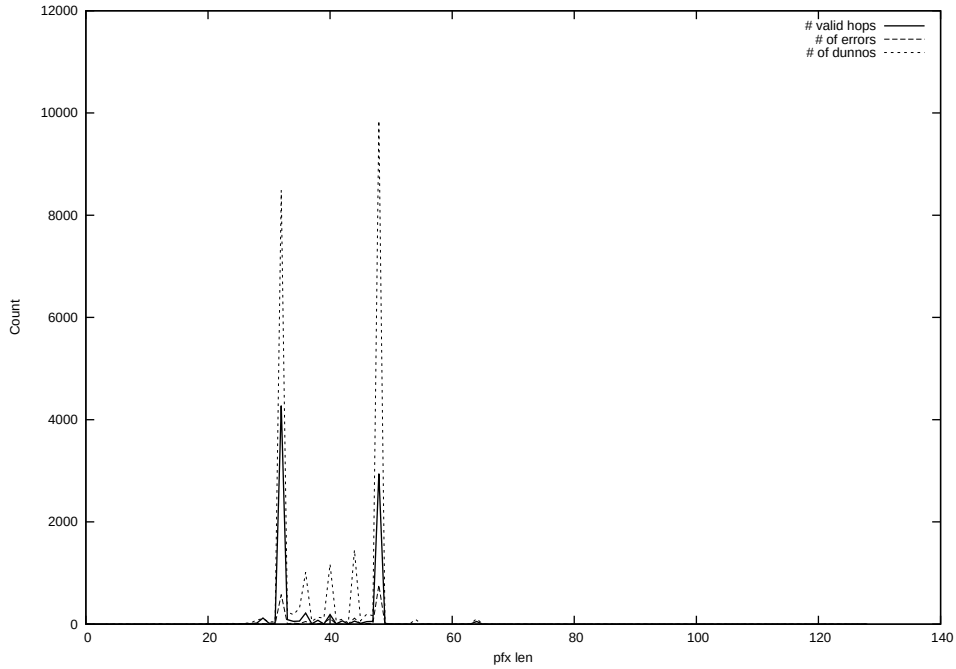
2012-12-28



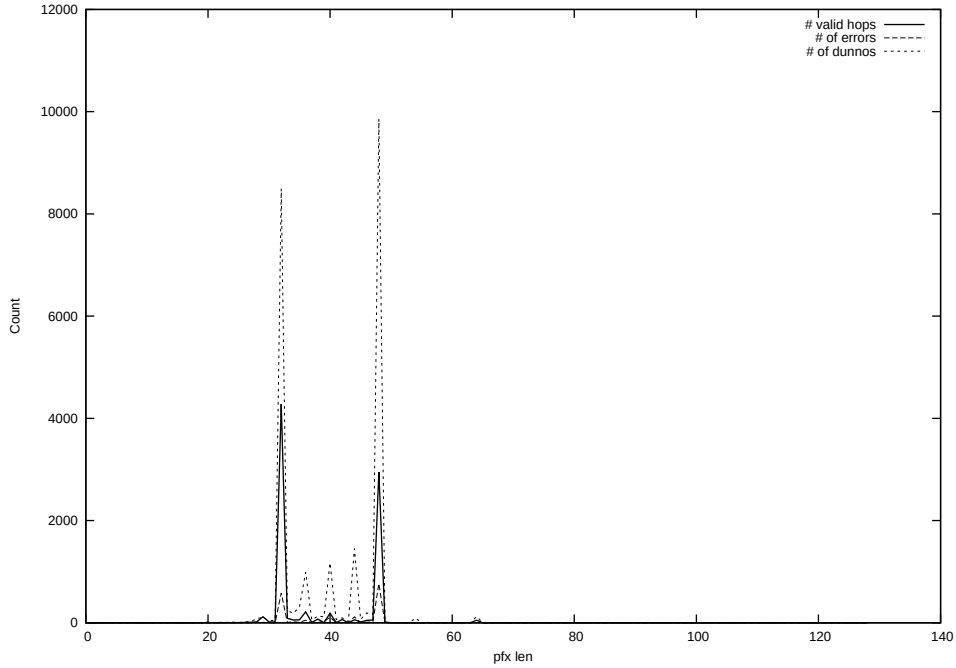
2012-12-29



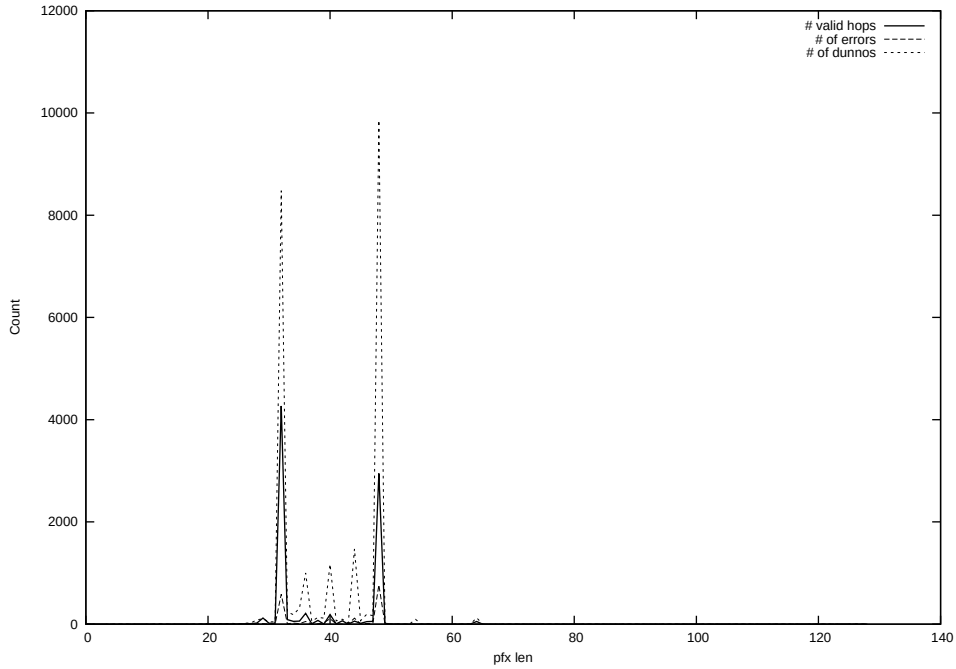
2012-12-30



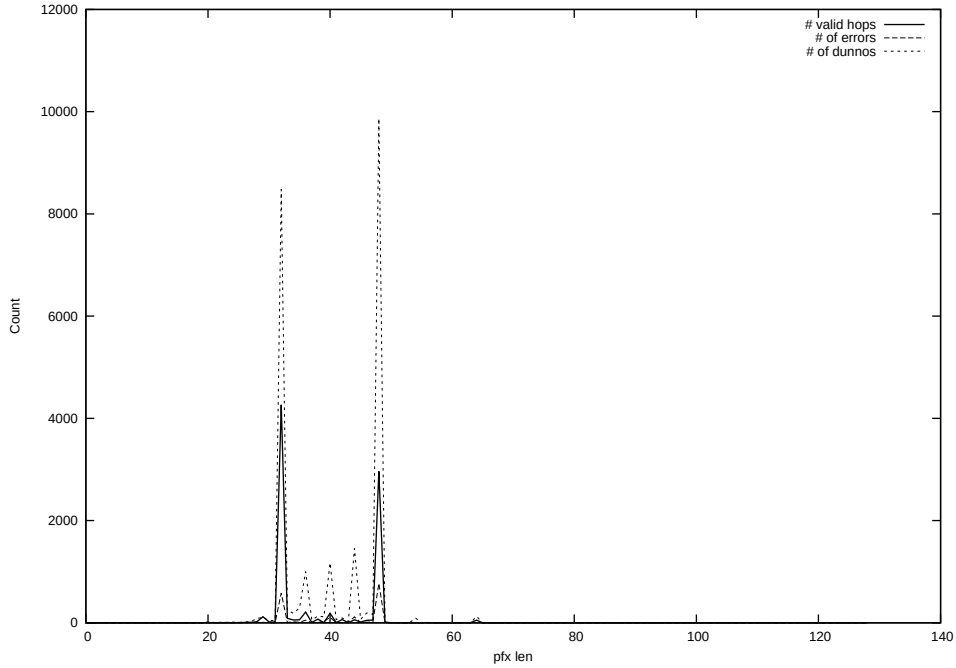
2012-12-31



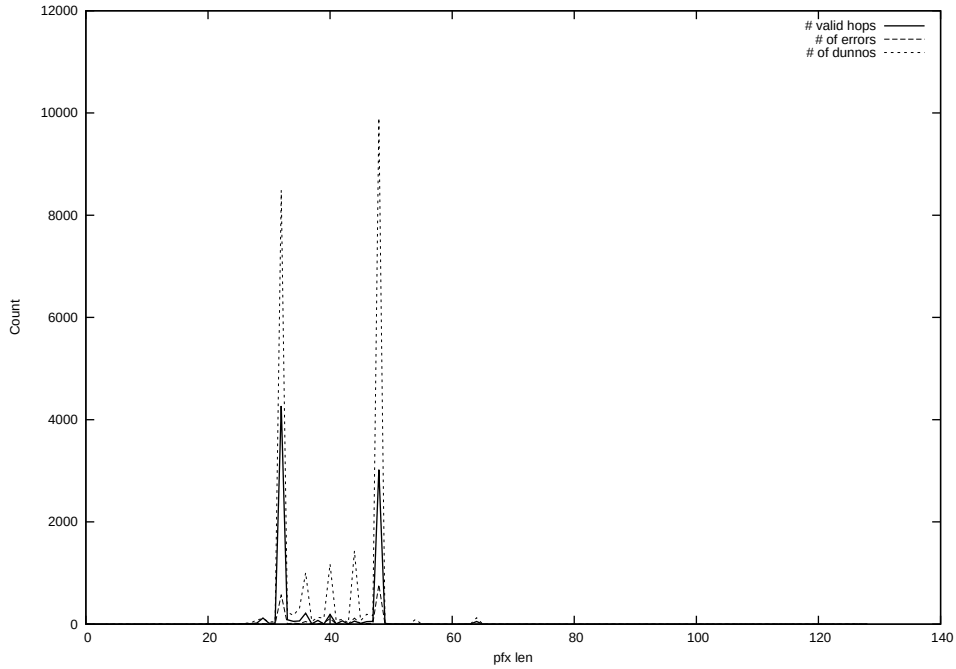
2013-01-01



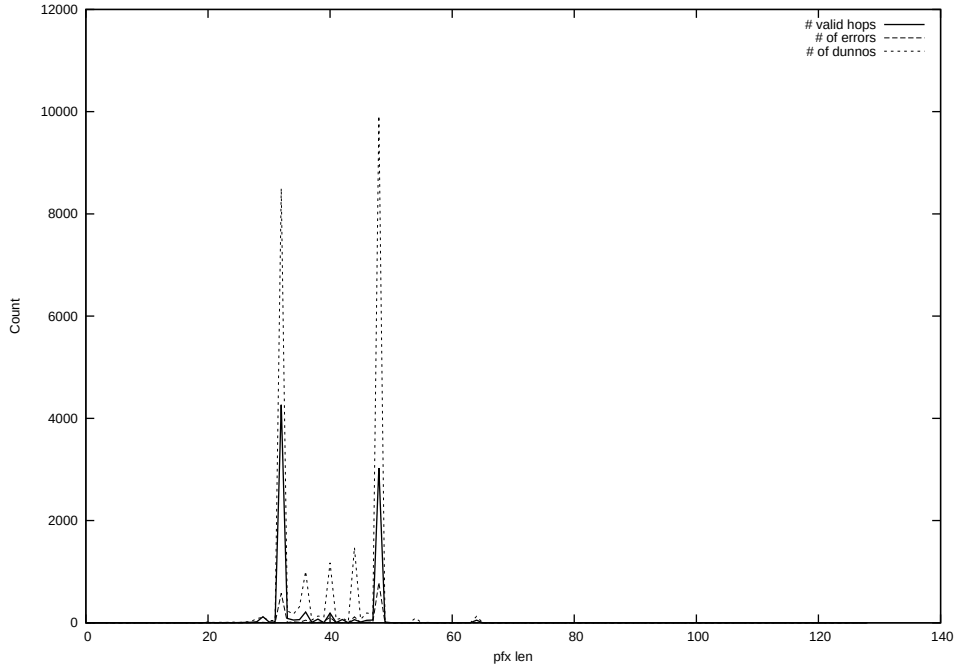
2013-01-02



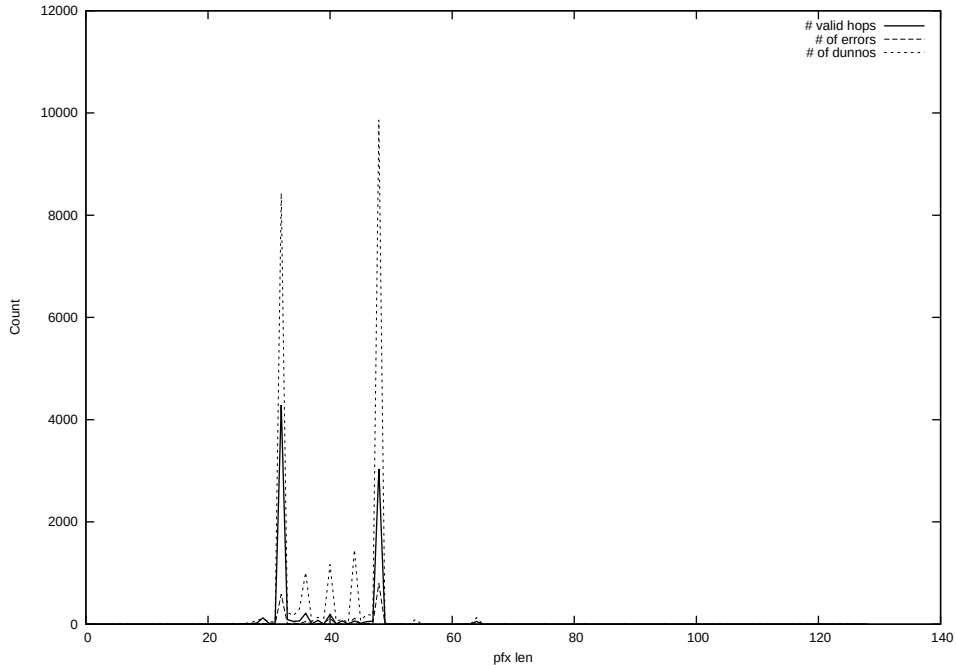
2013-01-03



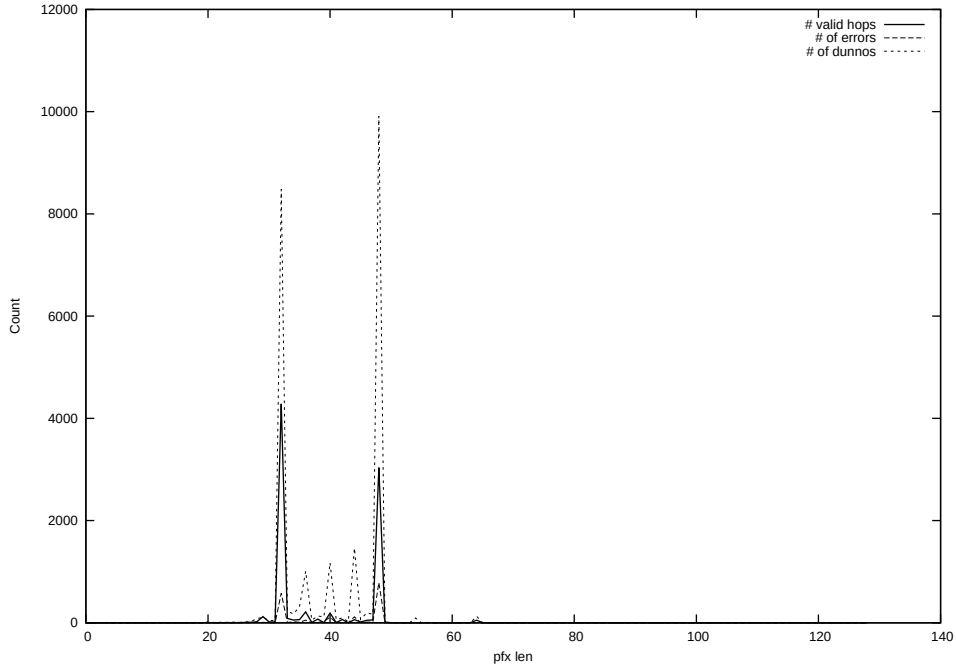
2013-01-04



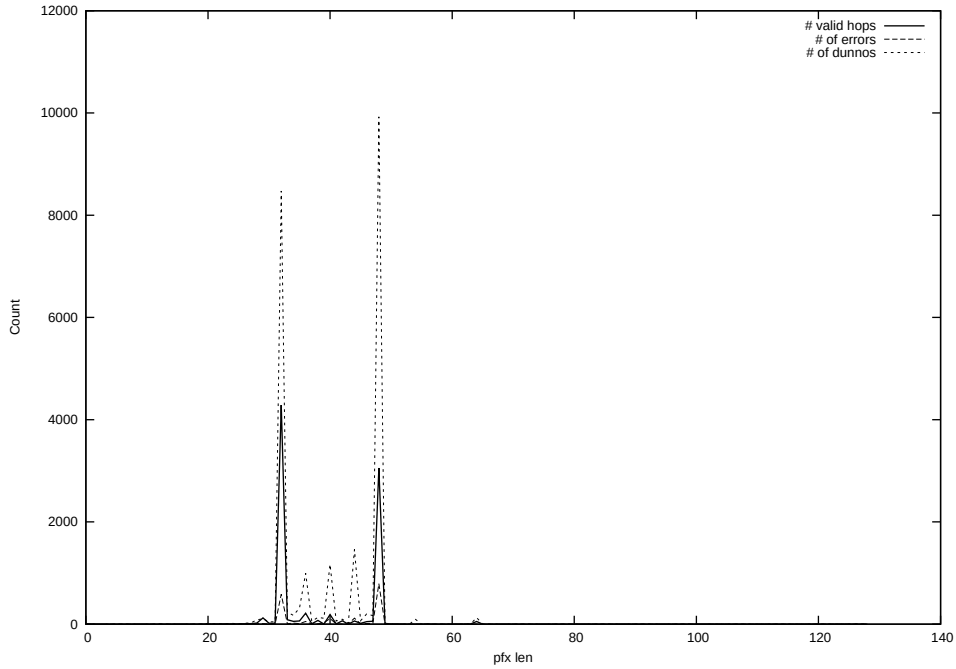
2013-01-05



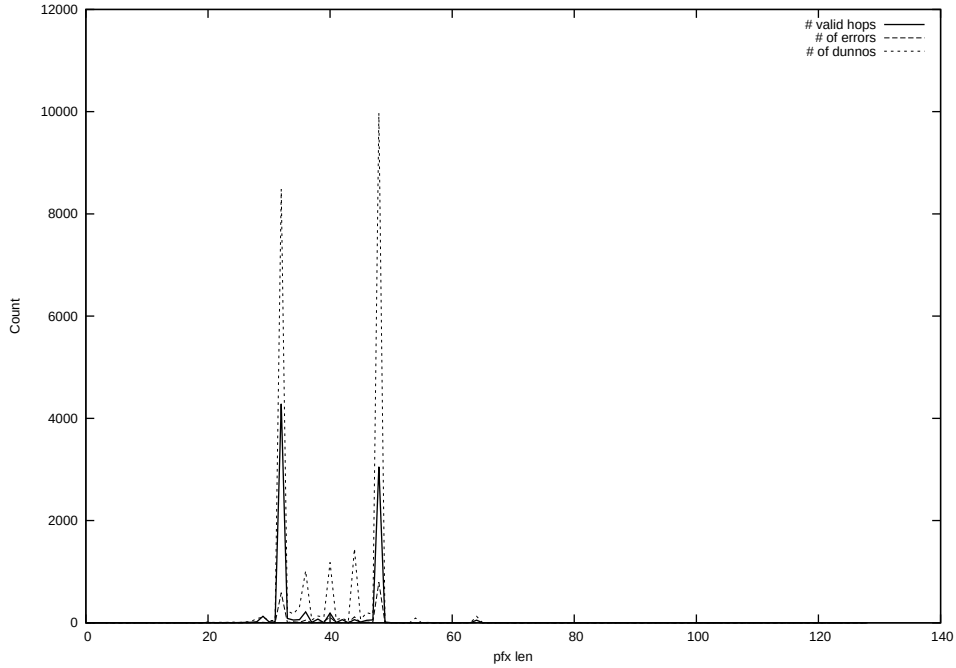
2013-01-06



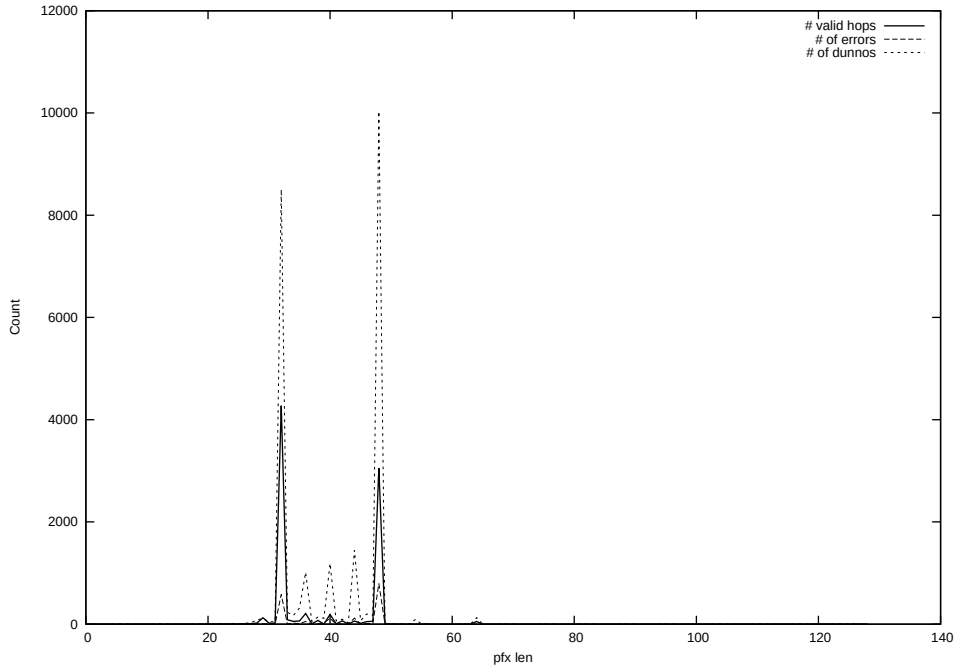
2013-01-07



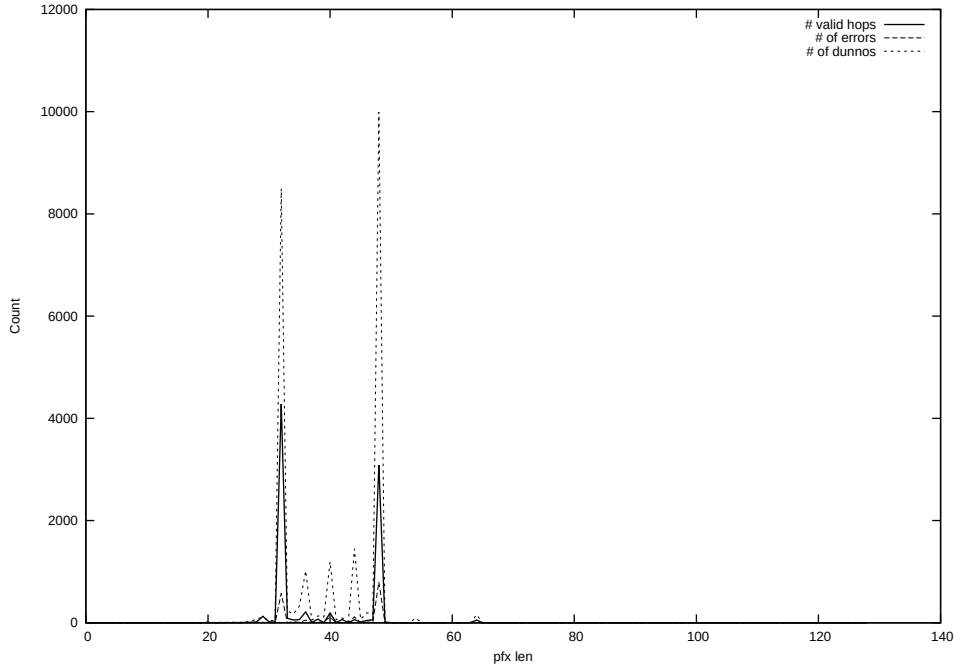
2013-01-08



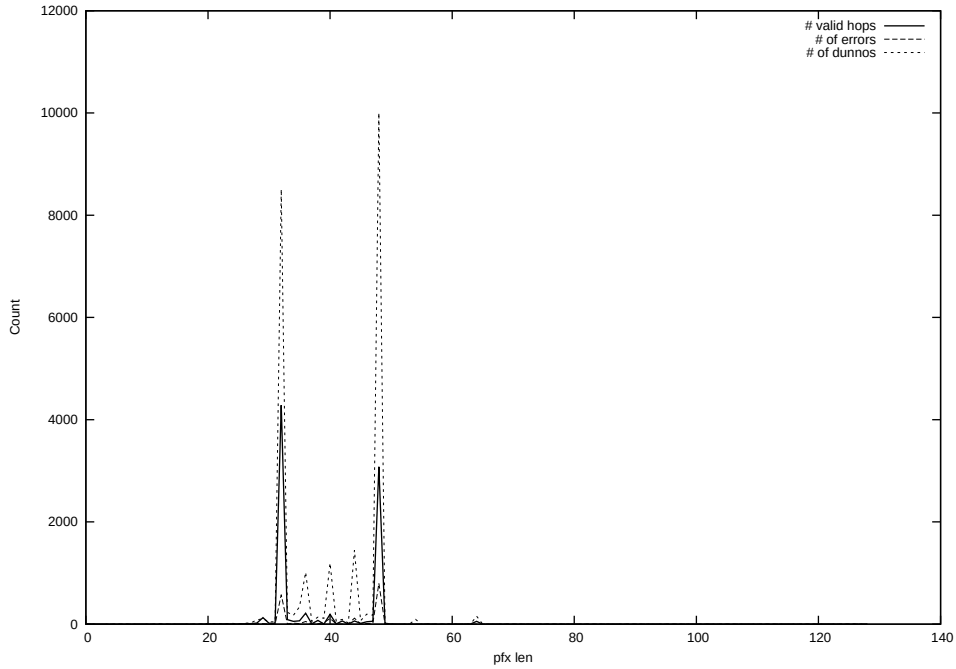
2013-01-09



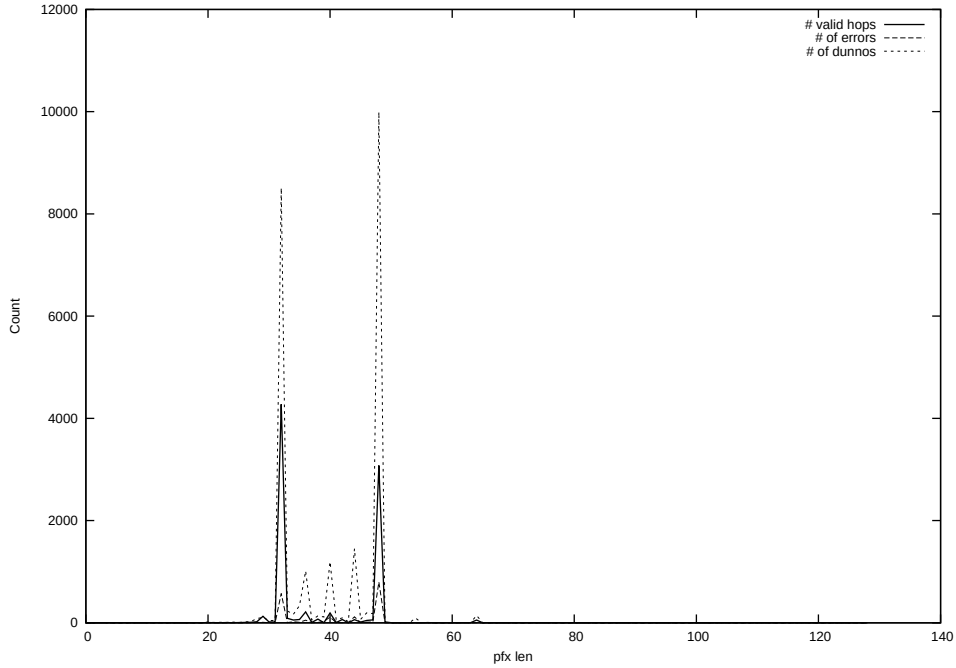
2013-01-10



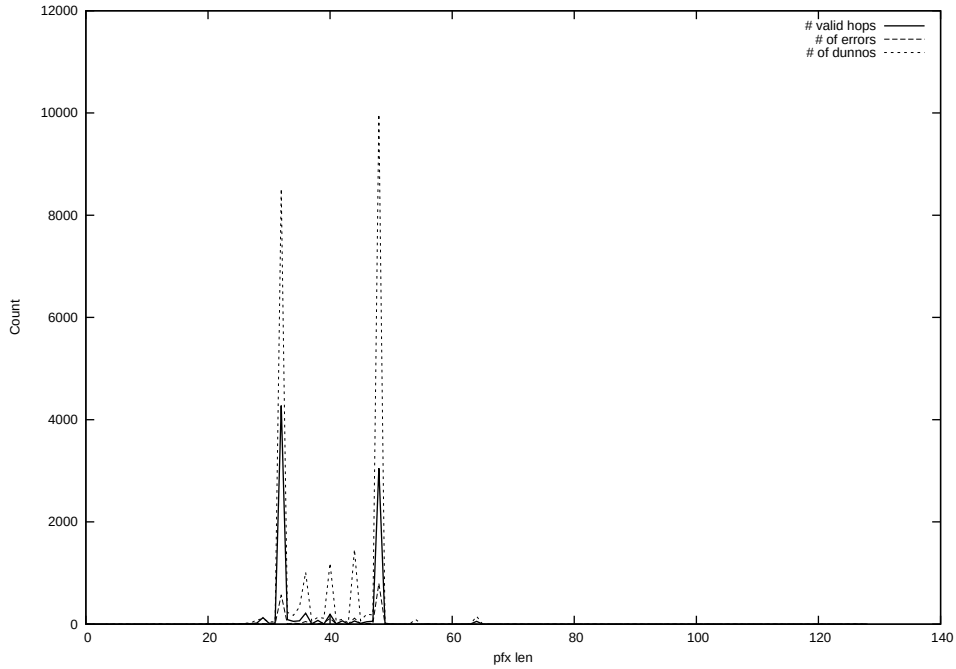
2013-01-11



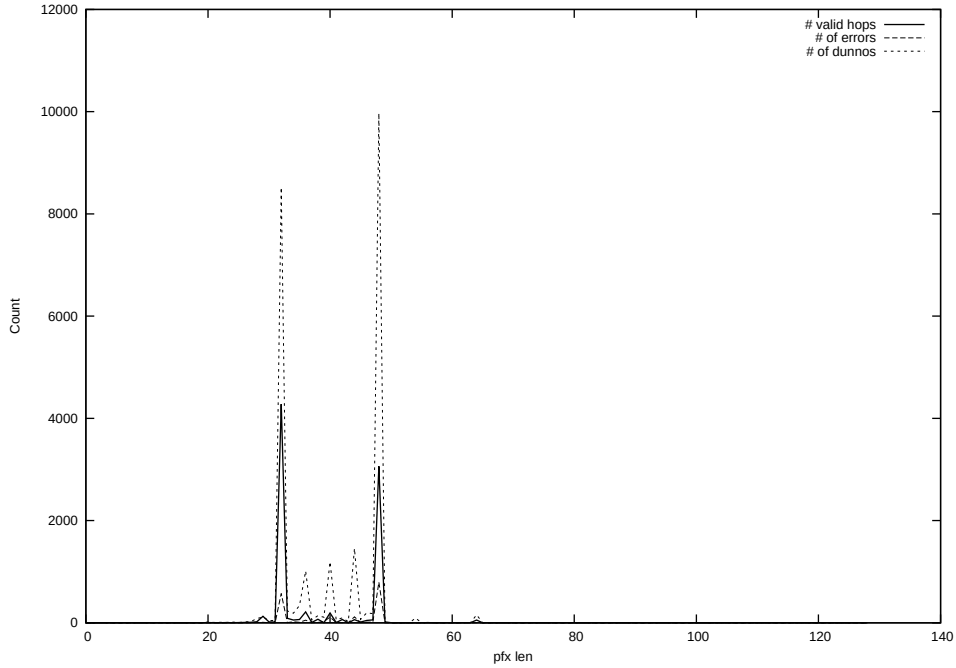
2013-01-12



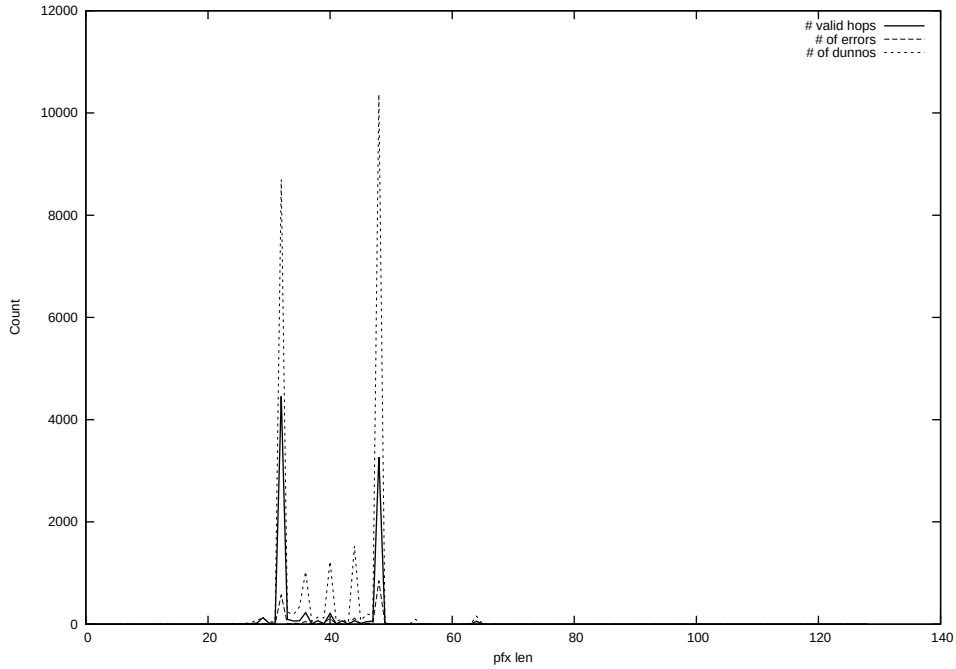
2013-01-13



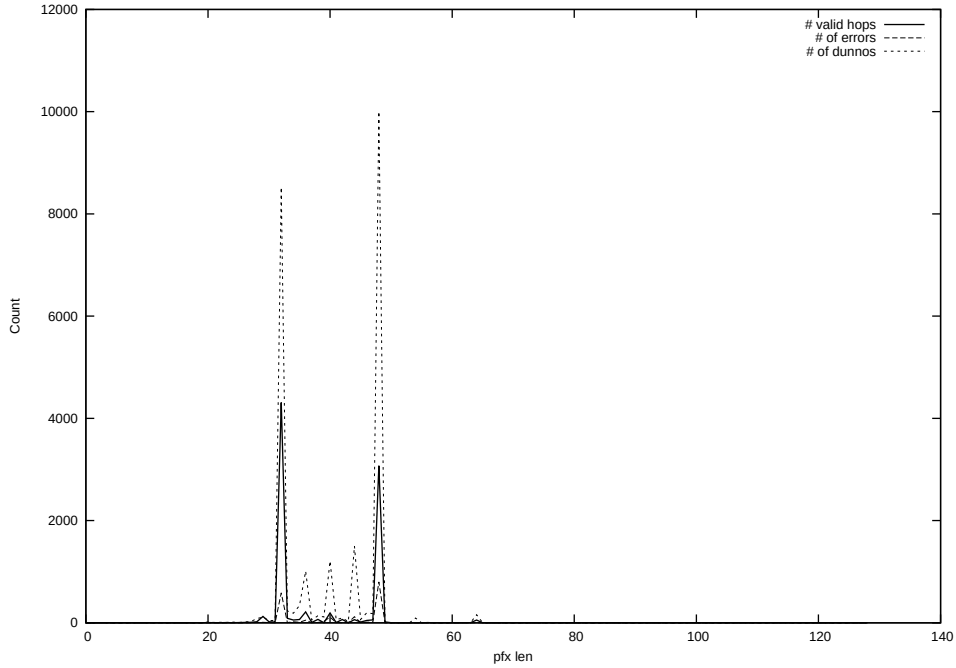
2013-01-14



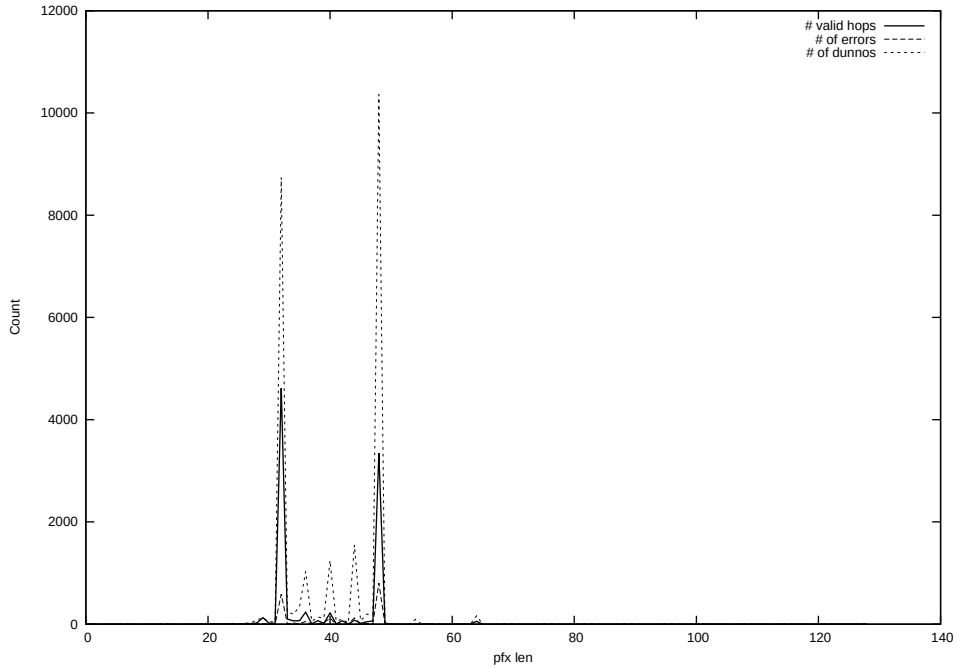
2013-01-15



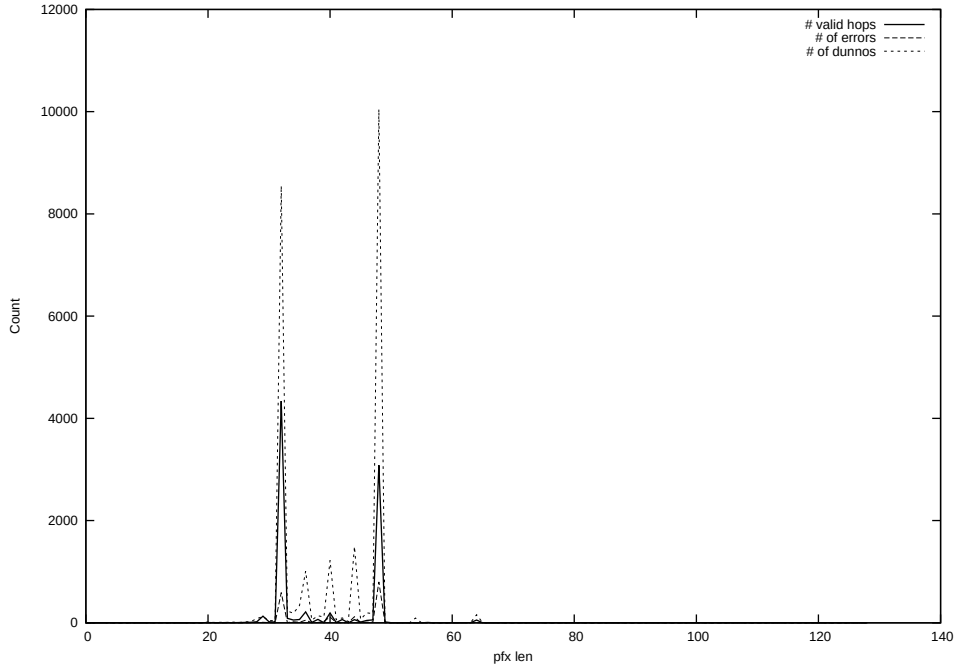
2013-01-16



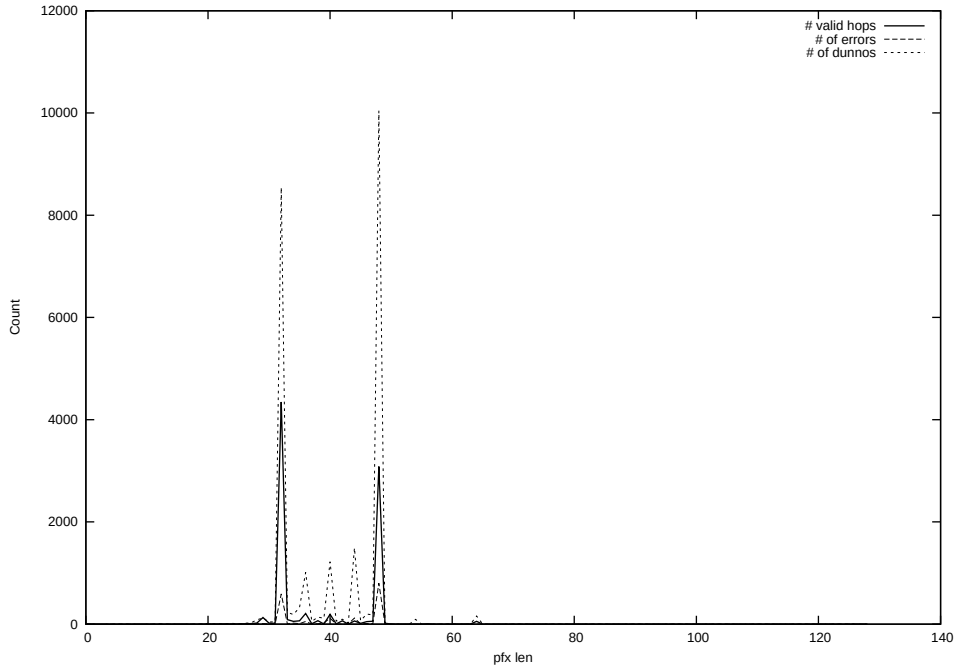
2013-01-17



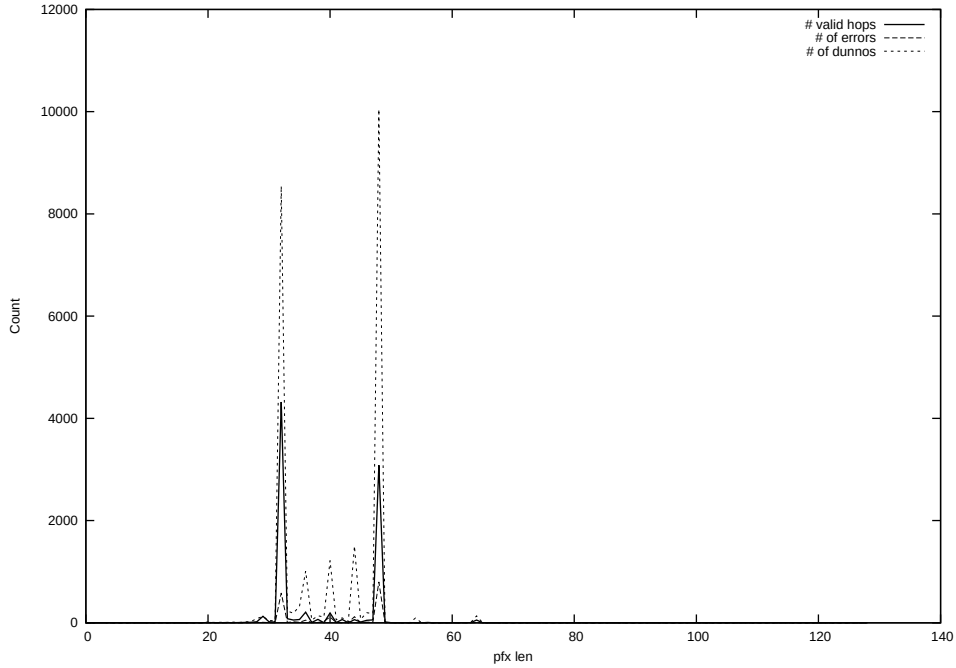
2013-01-18



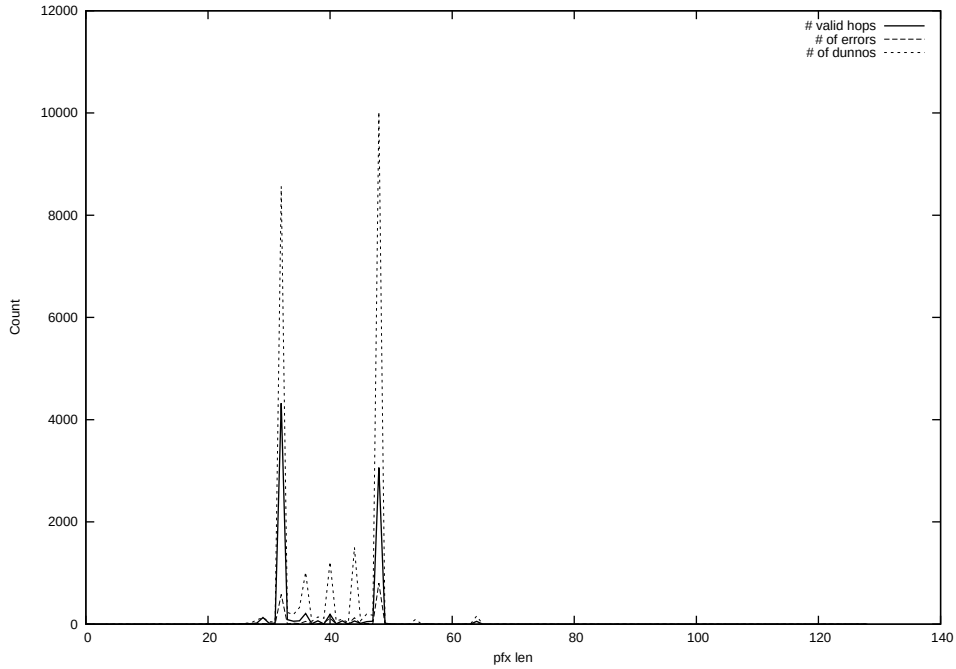
2013-01-19



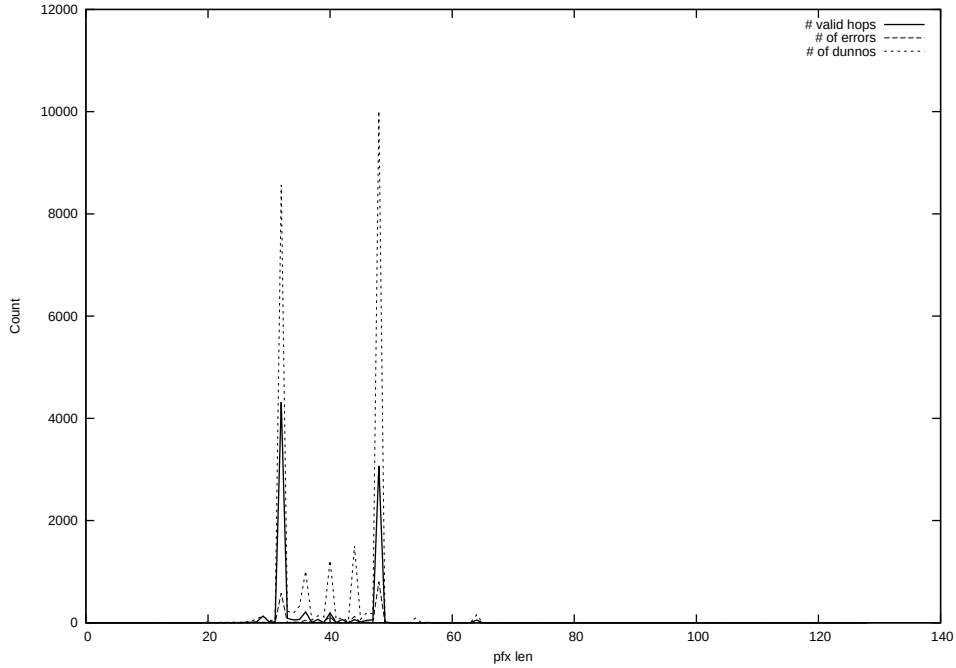
2013-01-20



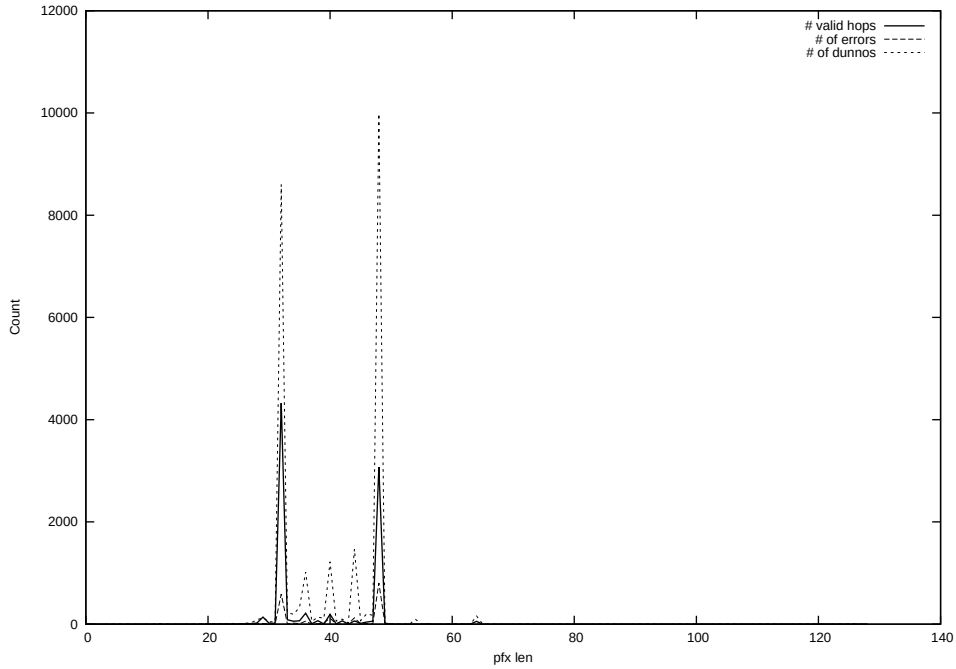
2013-01-21



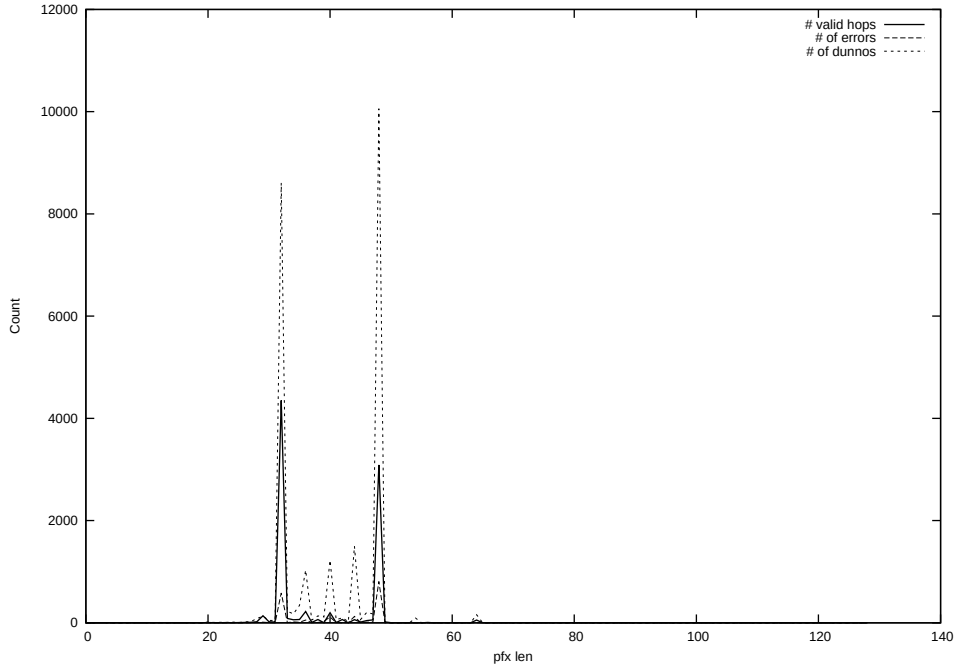
2013-01-22



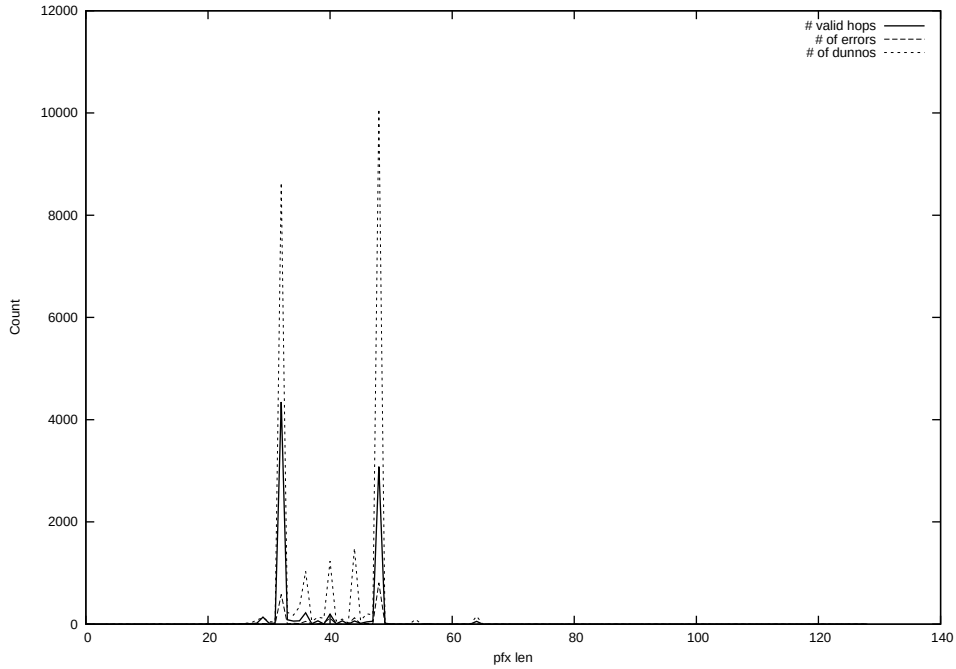
2013-01-23



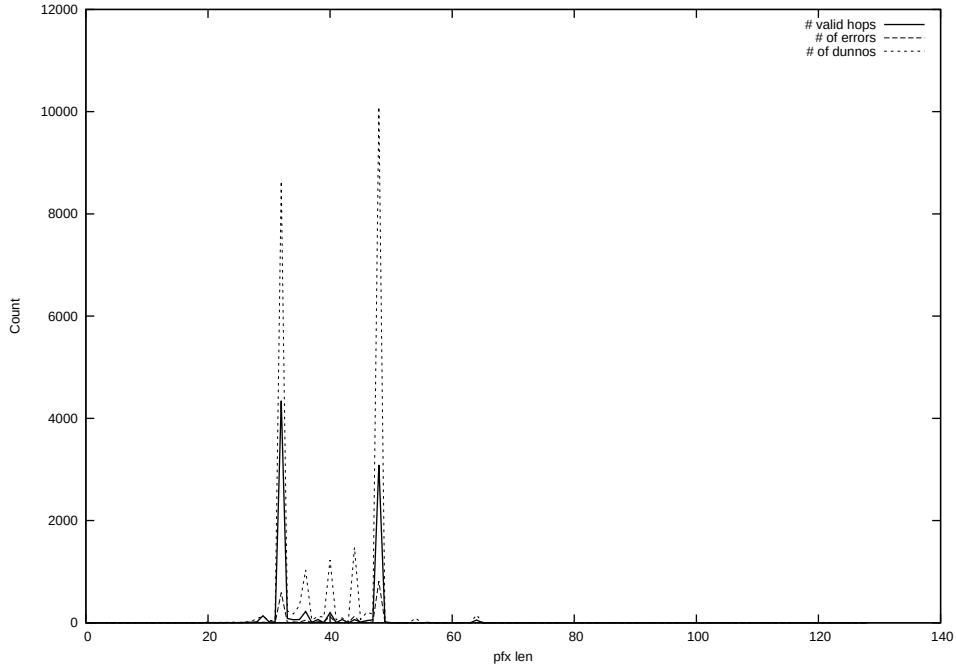
2013-01-24



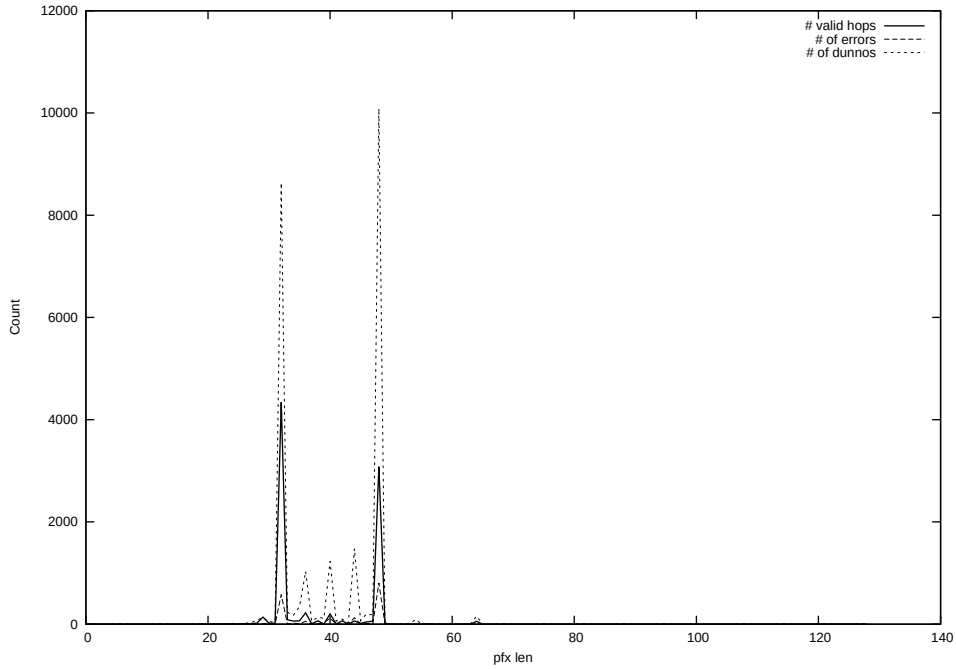
2013-01-25



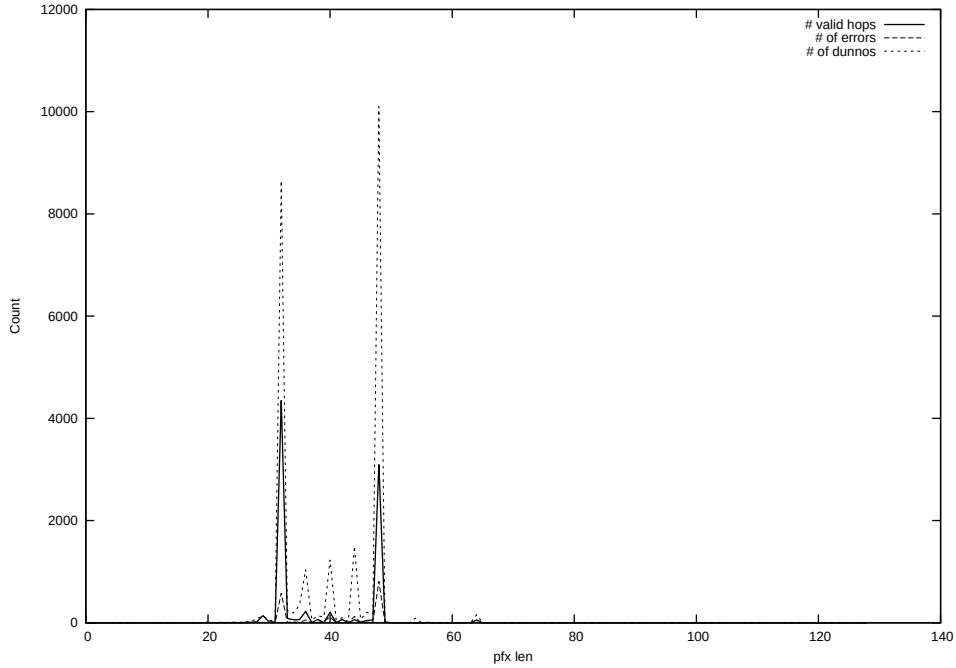
2013-01-26



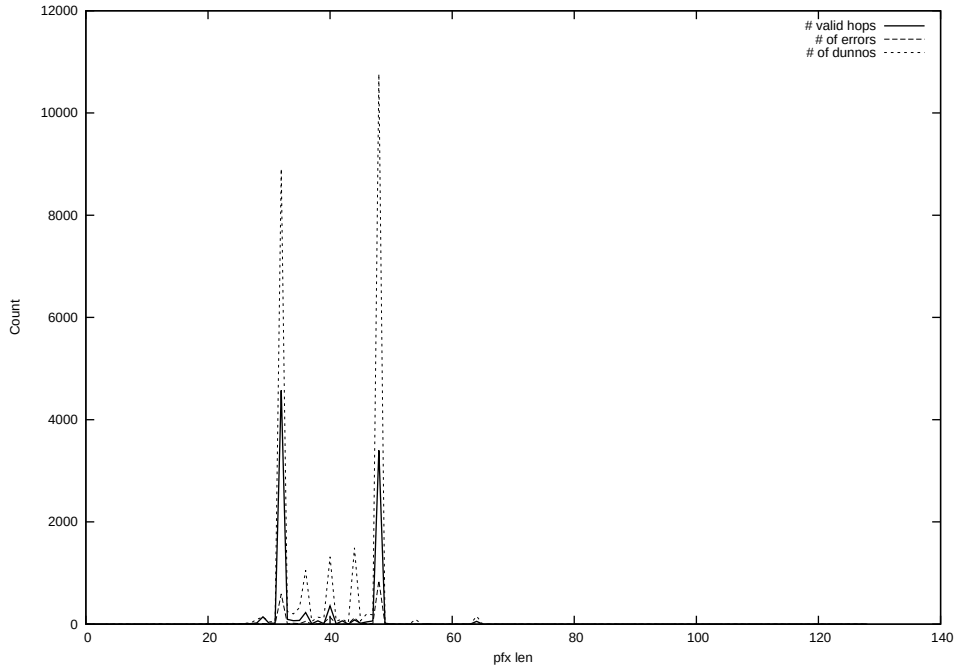
2013-01-27



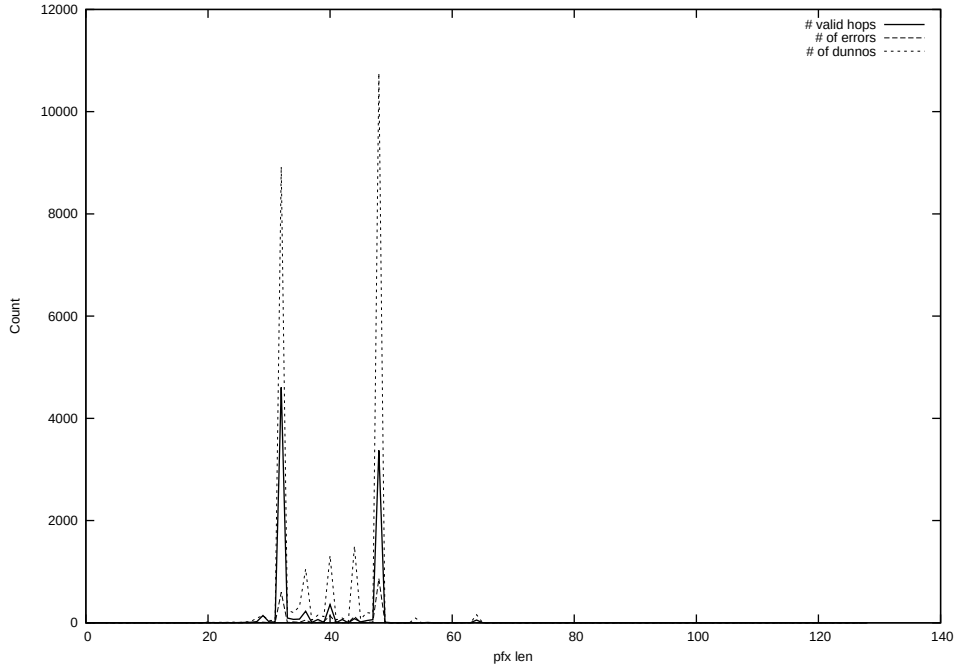
2013-01-28



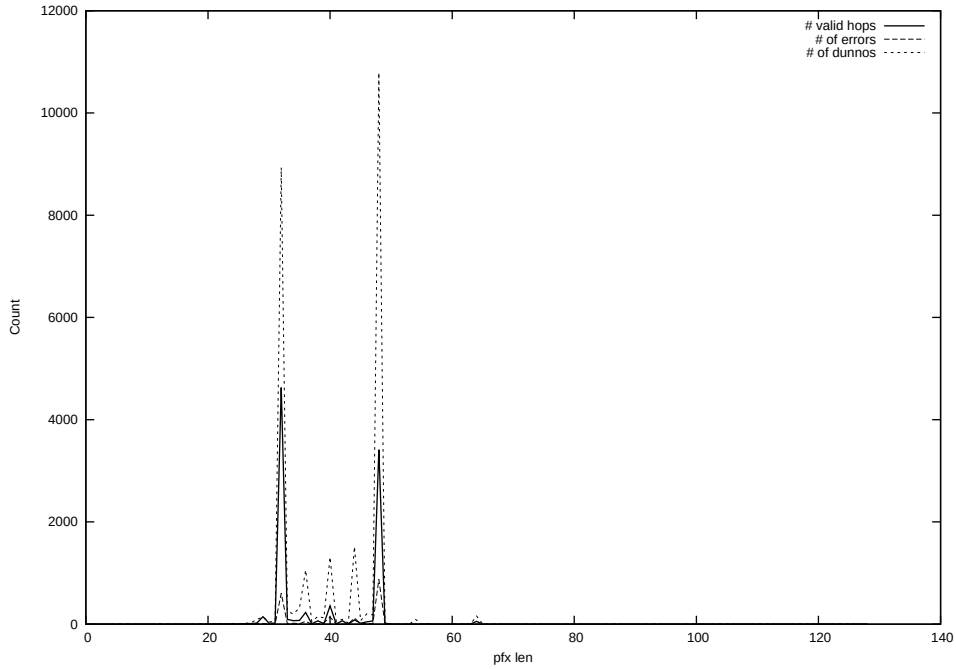
2013-01-29



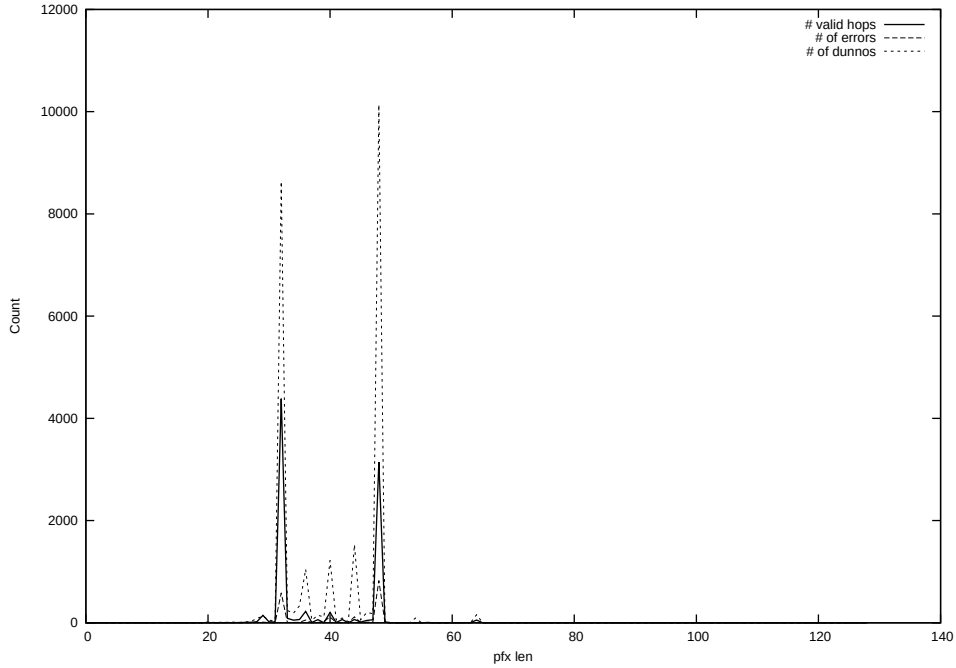
2013-01-30



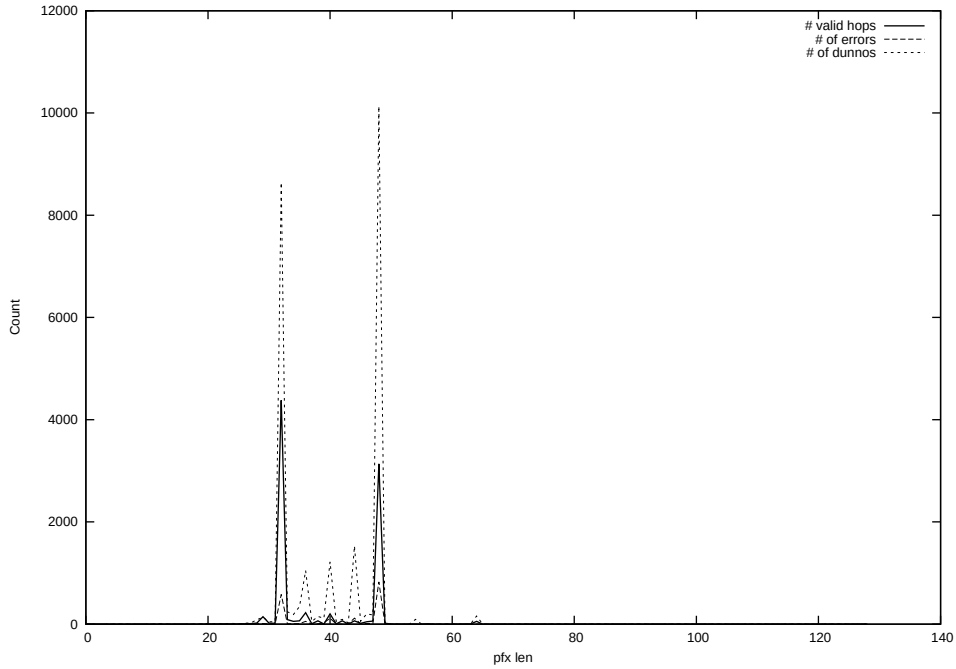
2013-01-31



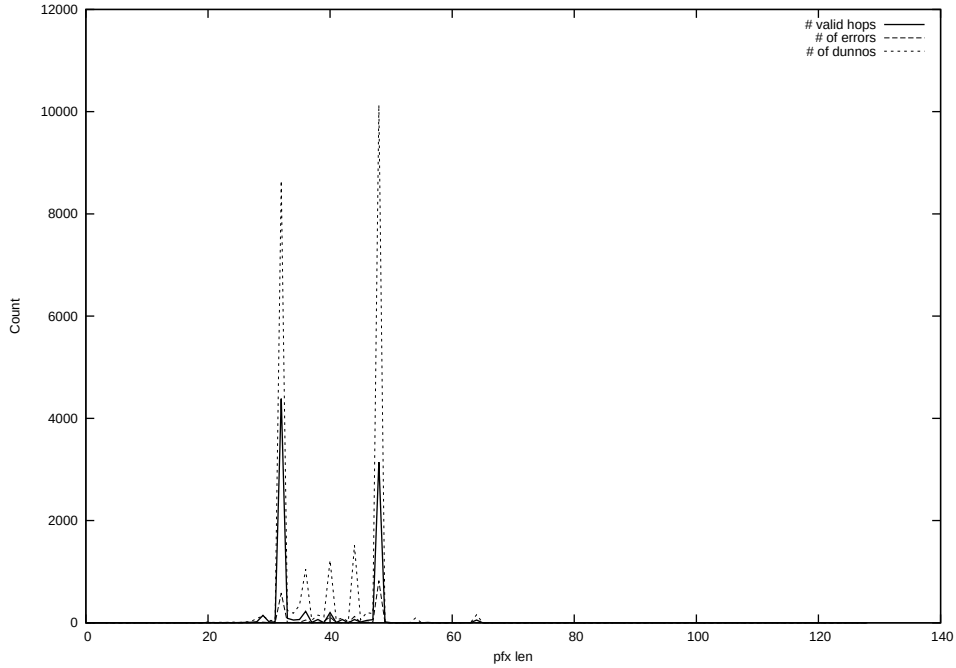
2013-02-01



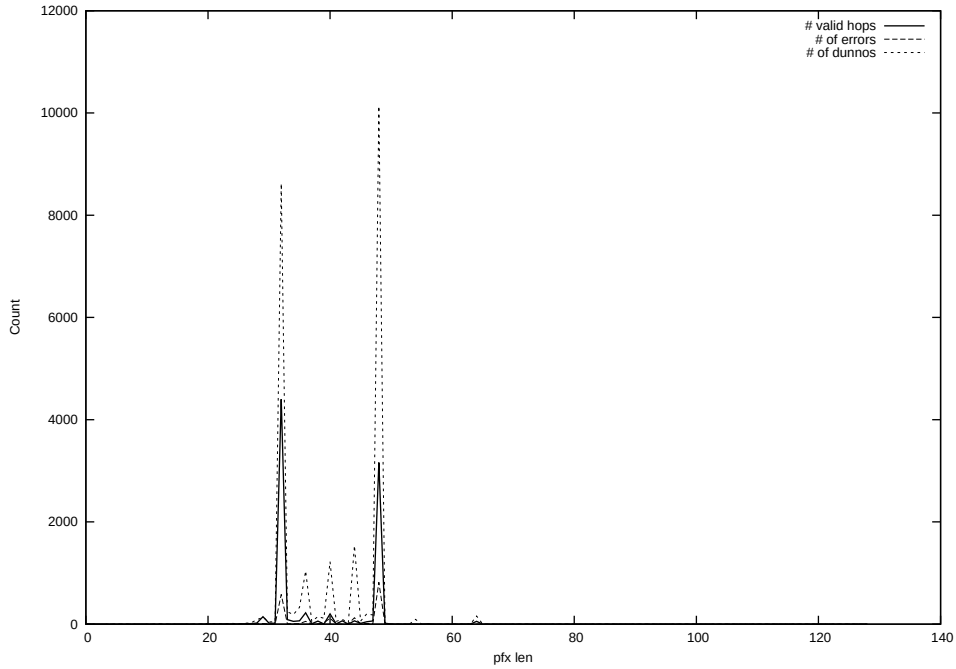
2013-02-02



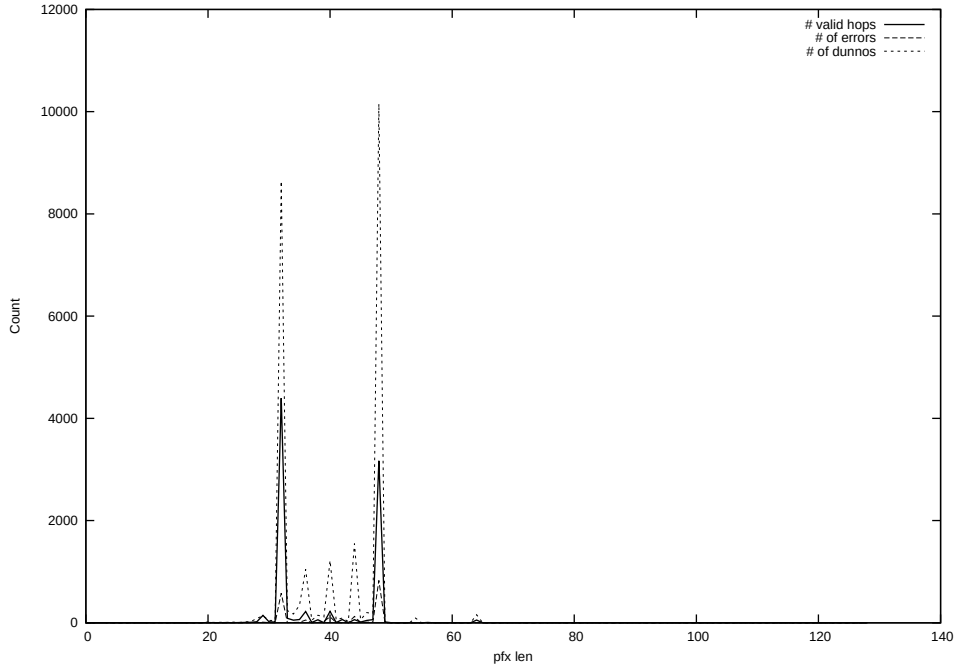
2013-02-03



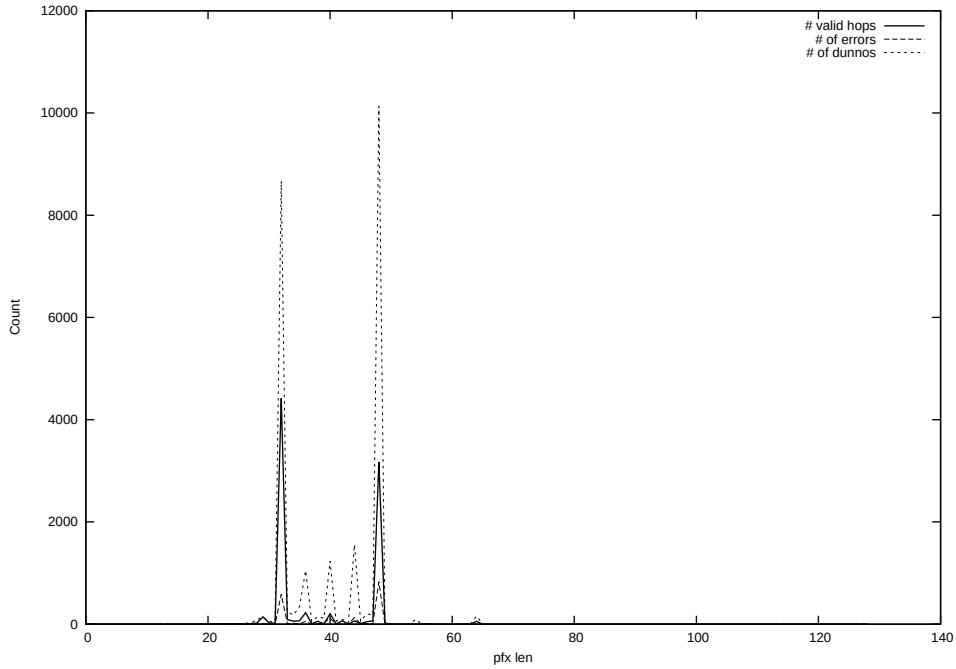
2013-02-04



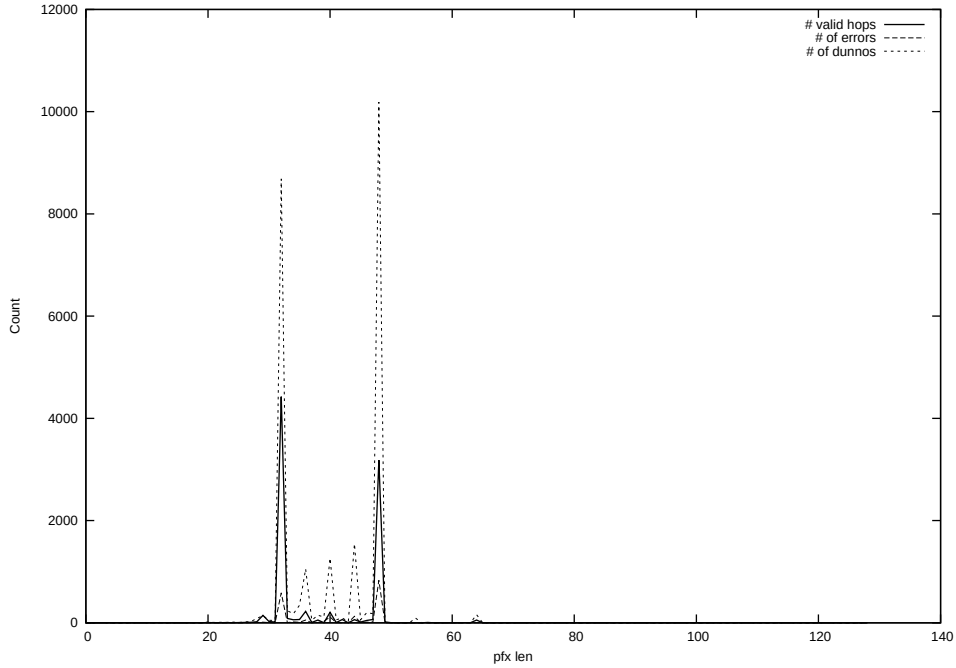
2013-02-05



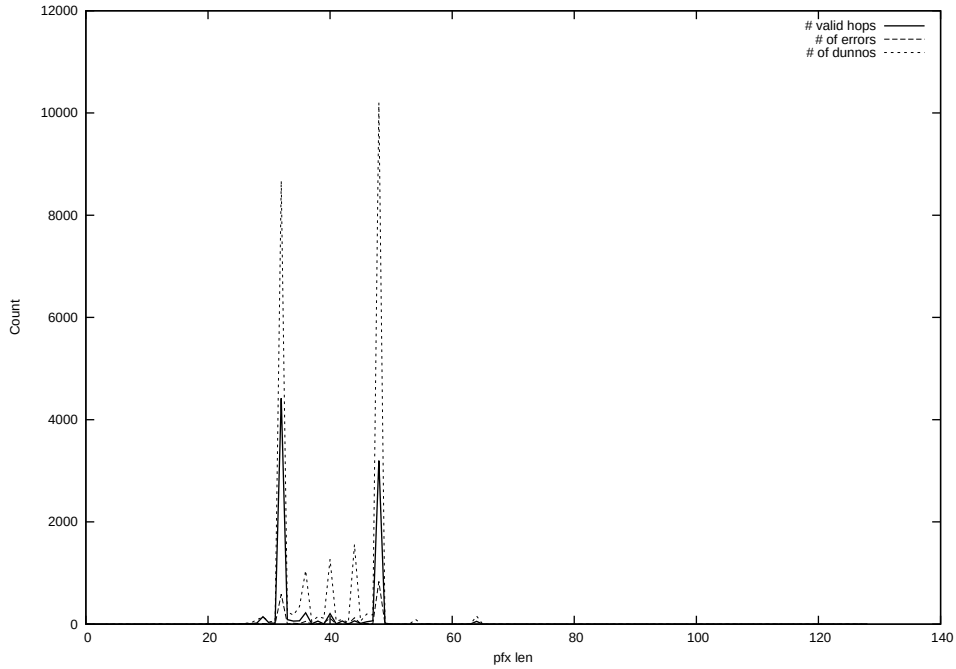
2013-02-06



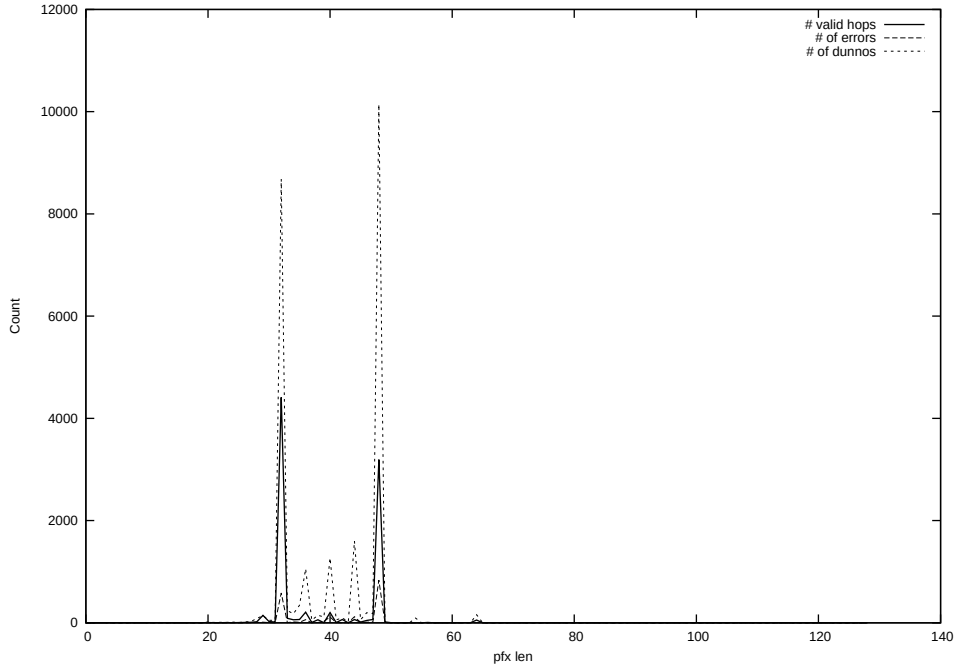
2013-02-07



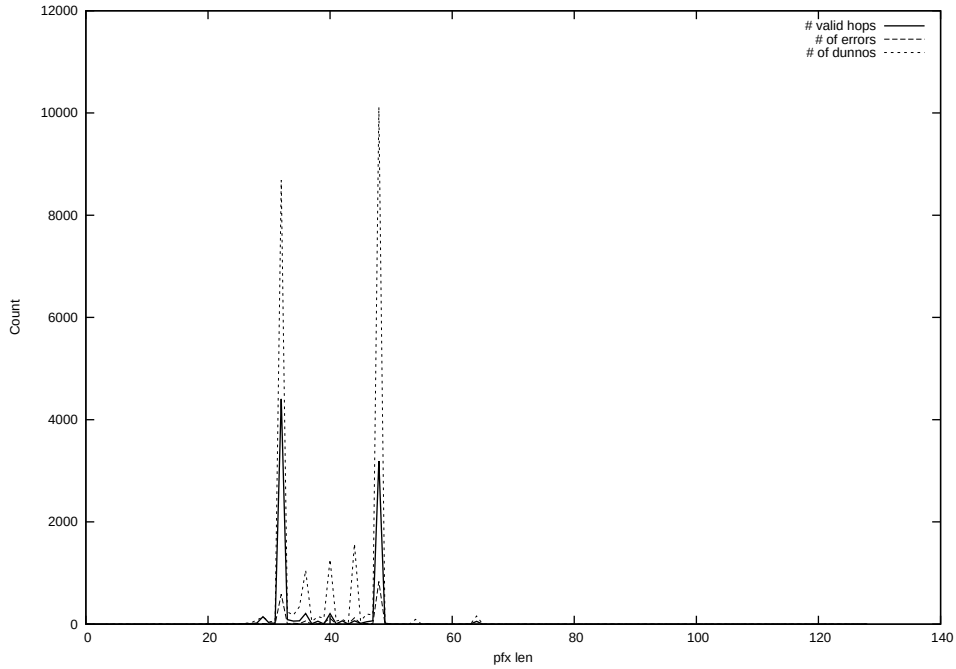
2013-02-08



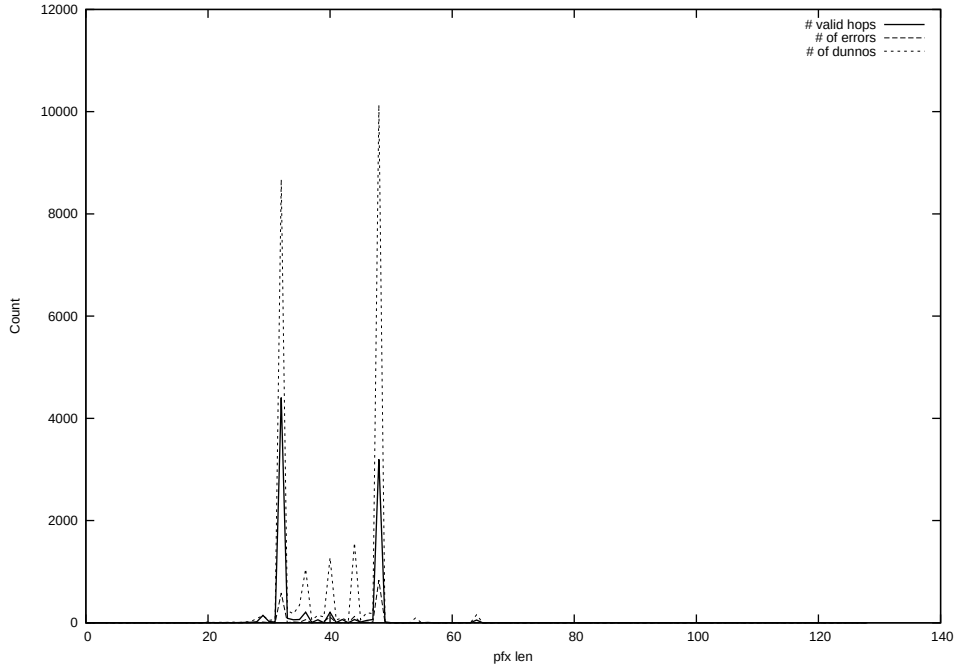
2013-02-09



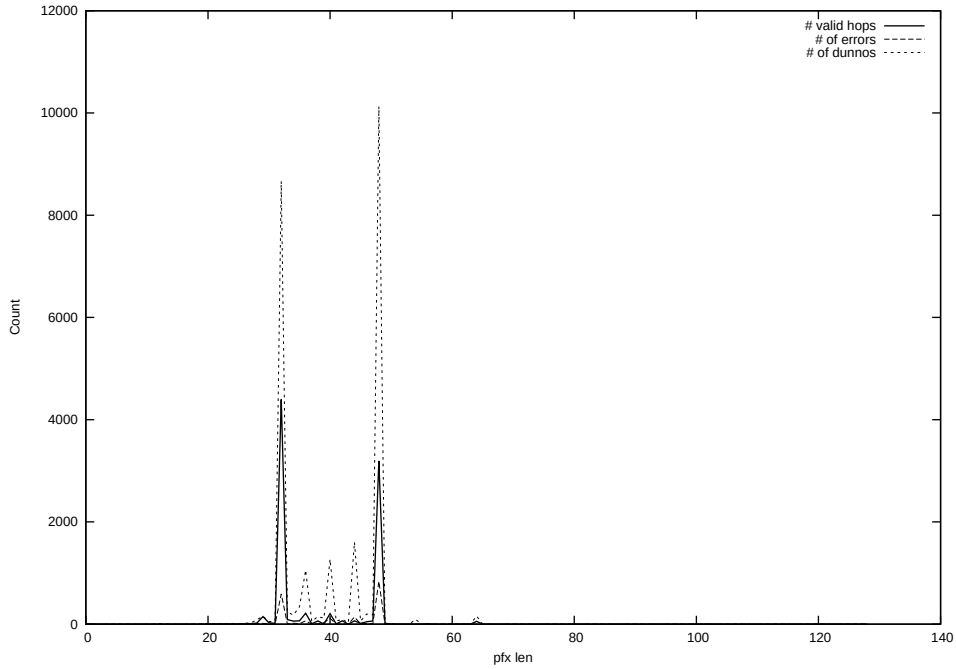
2013-02-10



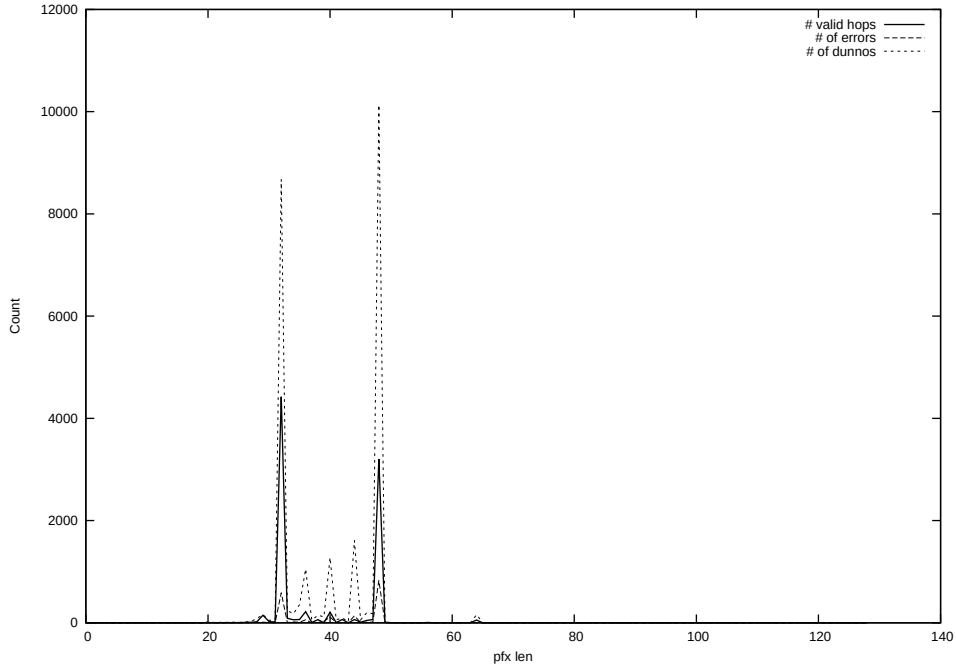
2013-02-11



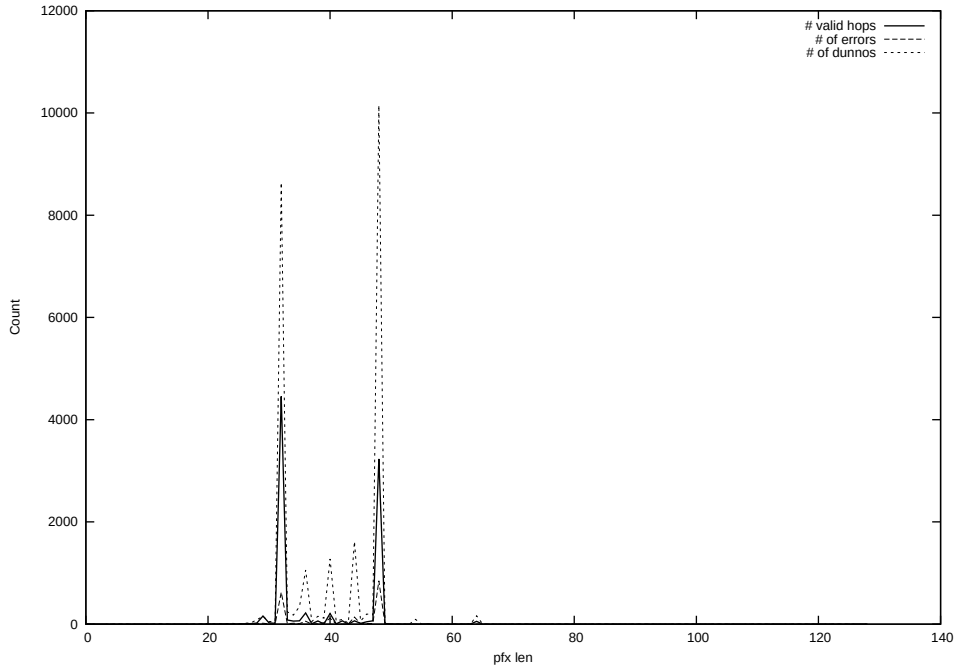
2013-02-12



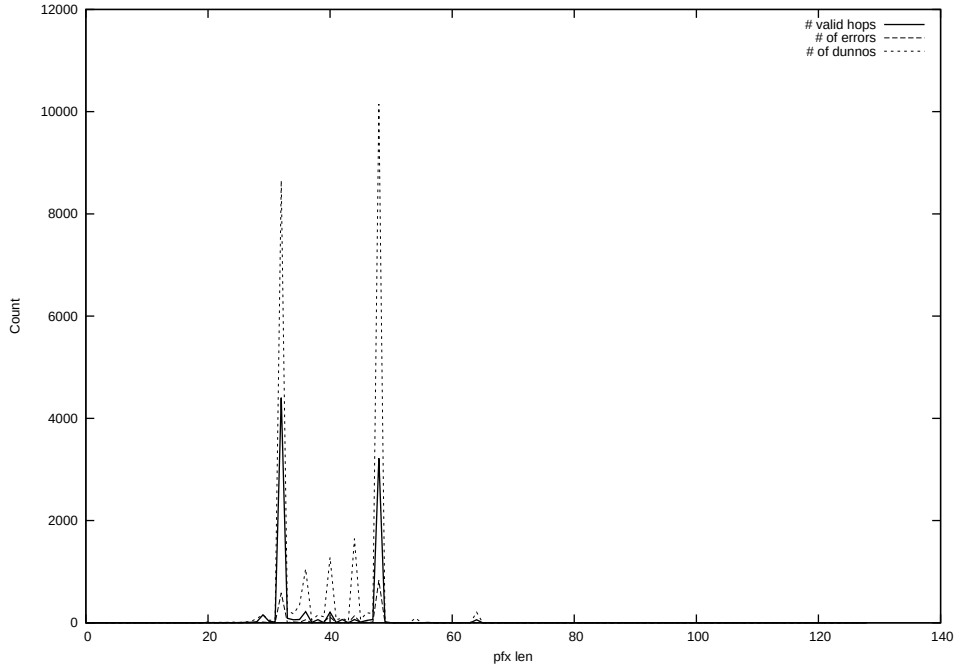
2013-02-13



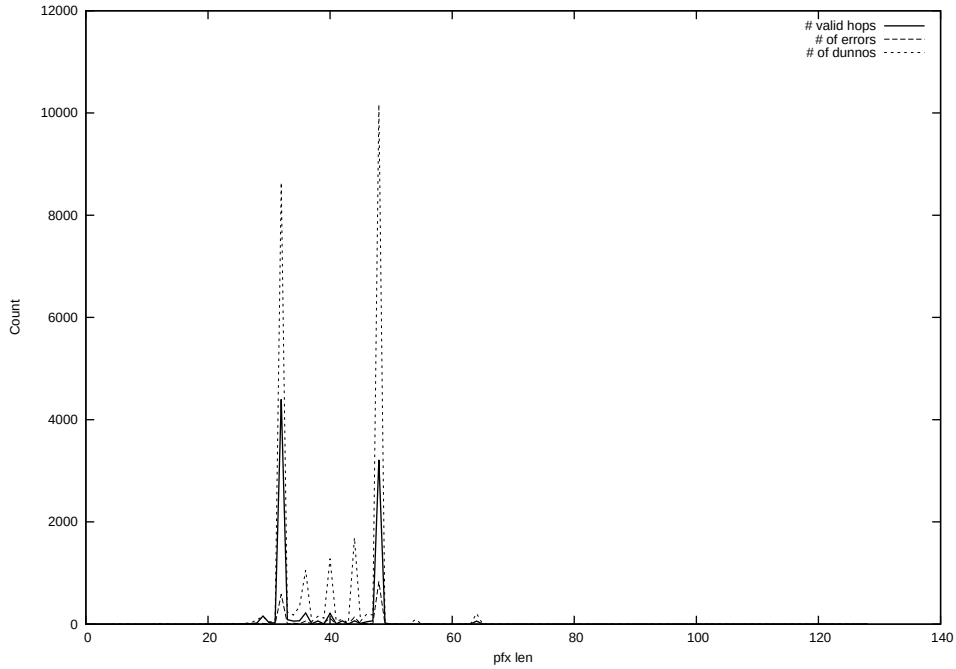
2013-02-14



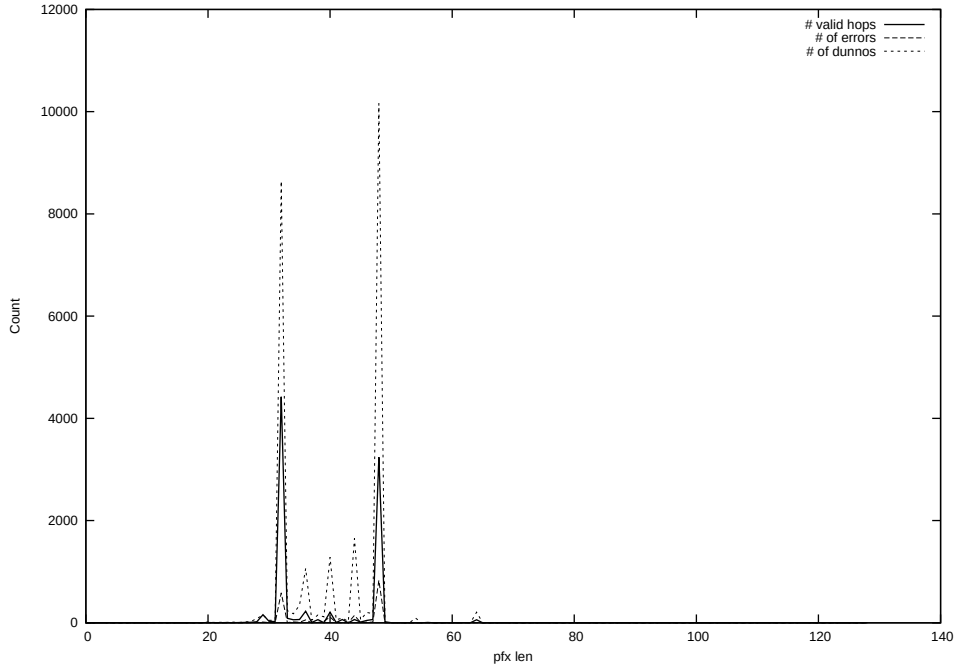
2013-02-15



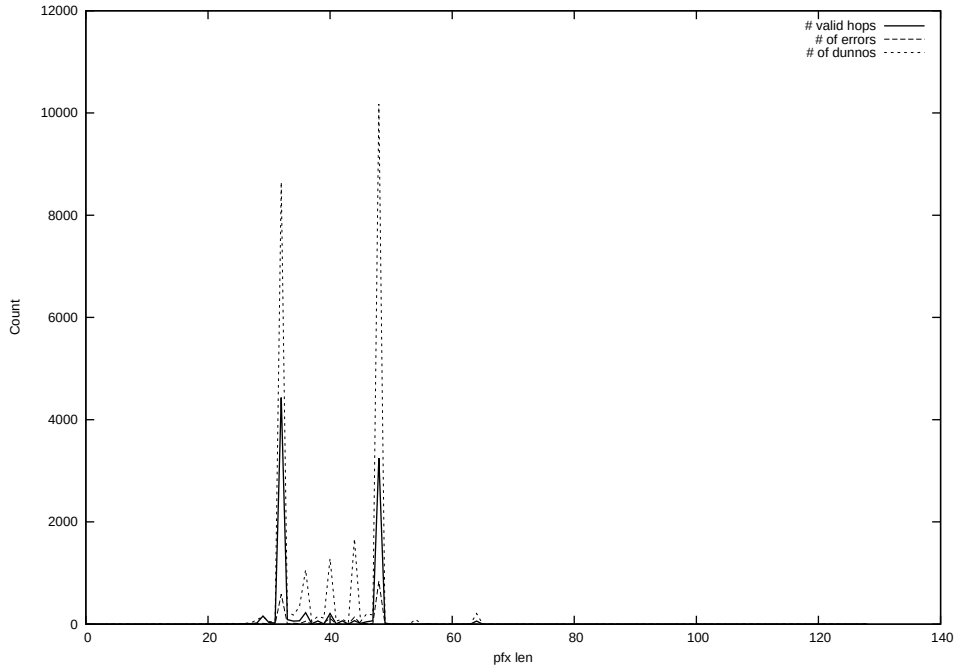
2013-02-16



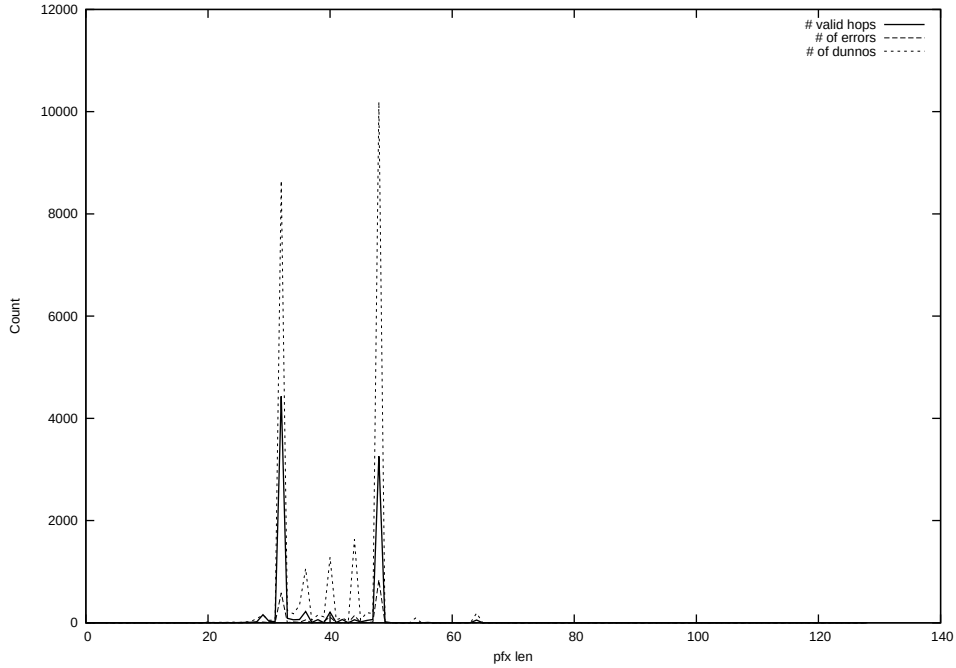
2013-02-17



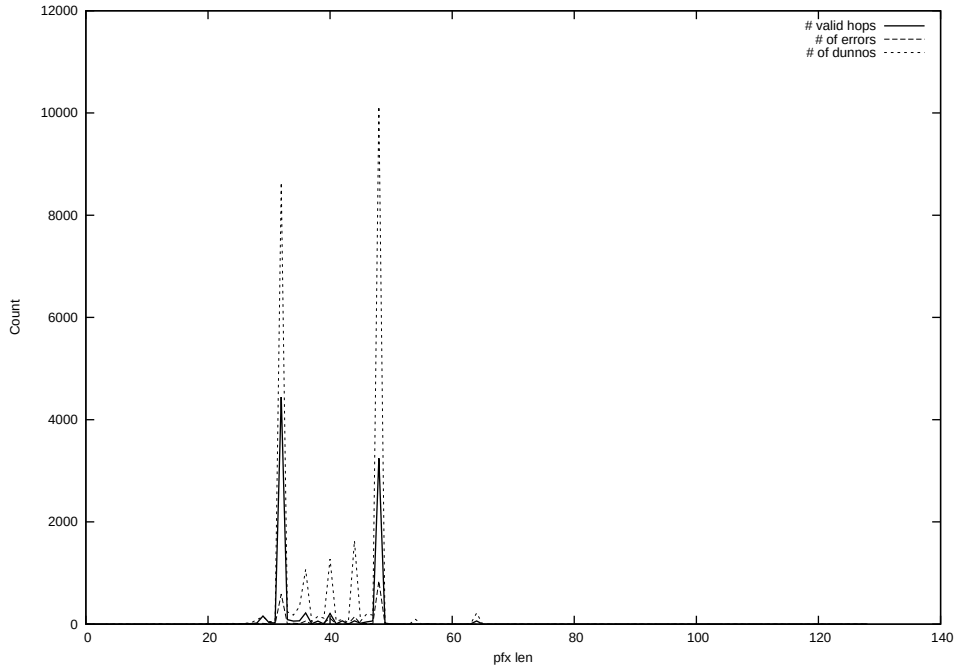
2013-02-18



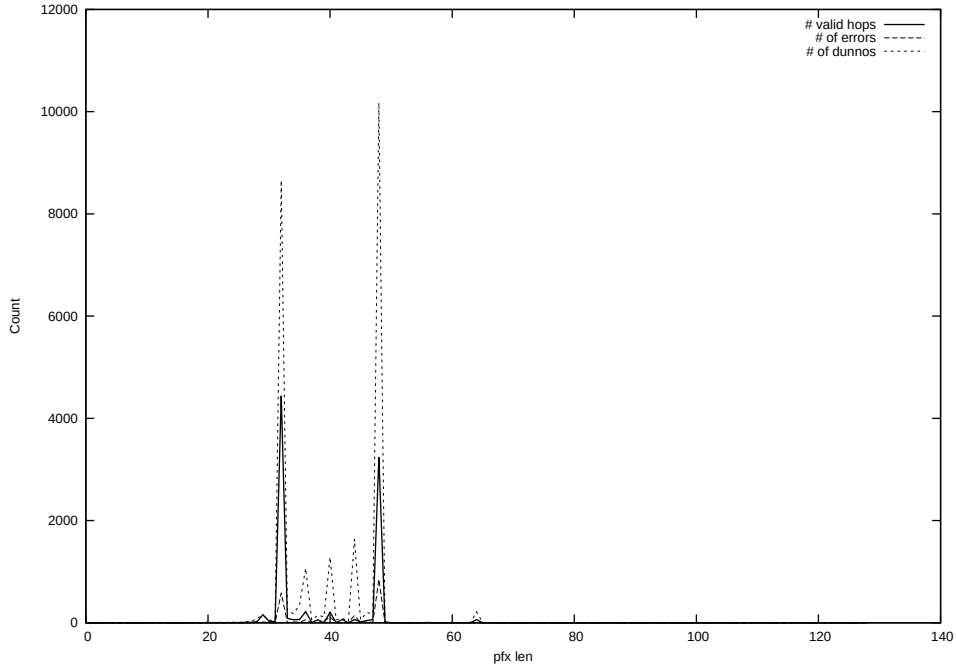
2013-02-19



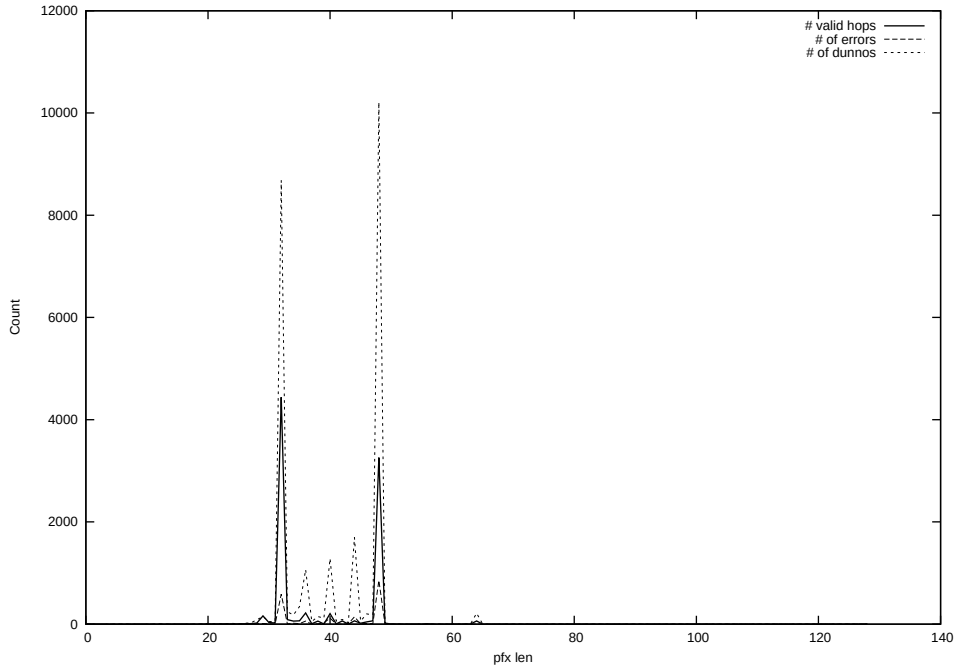
2013-02-20



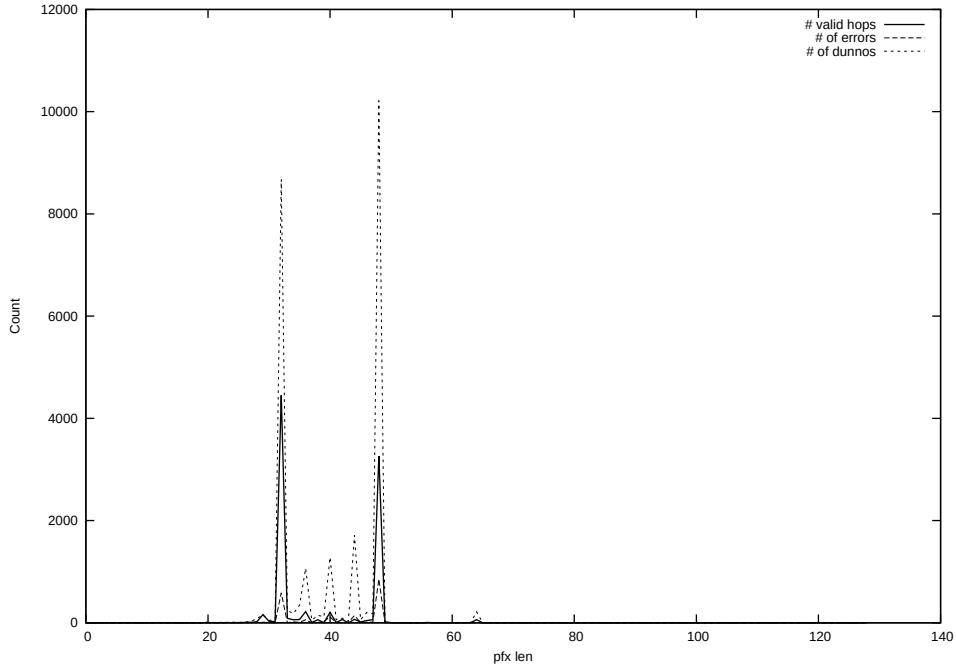
2013-02-21



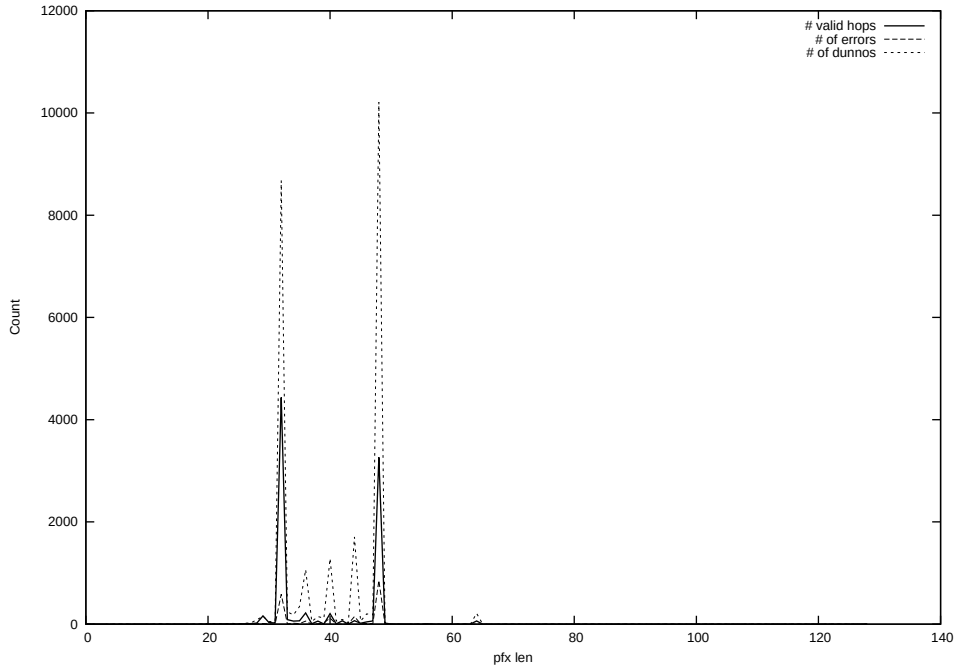
2013-02-22



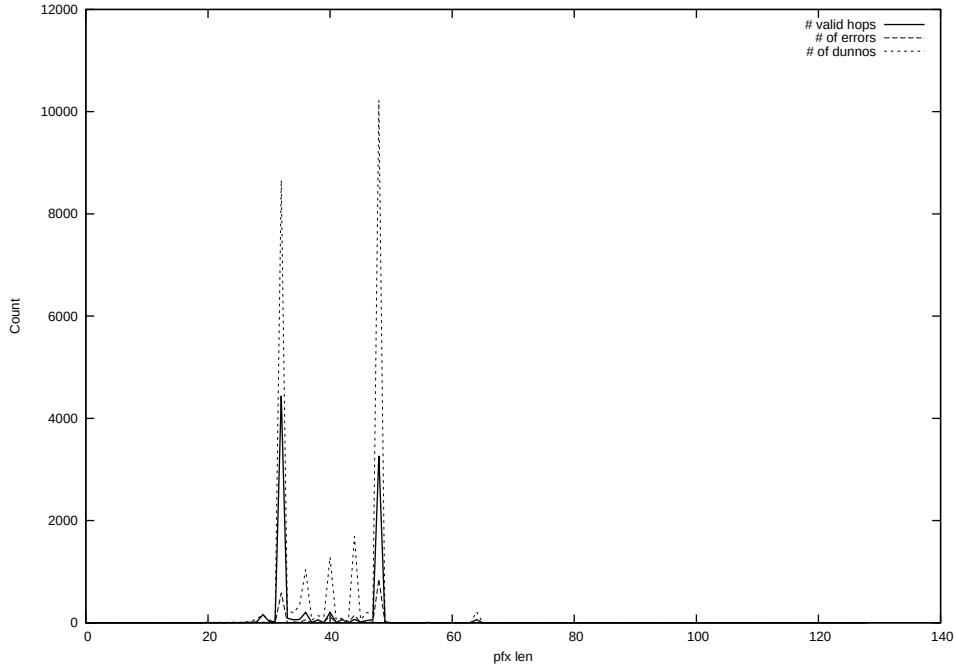
2013-02-23



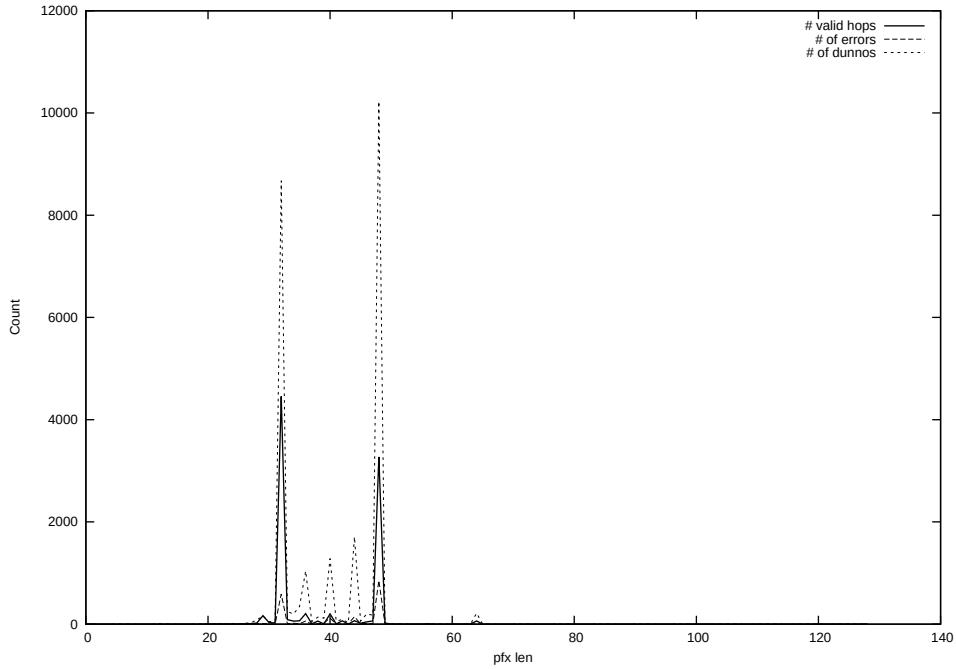
2013-02-24



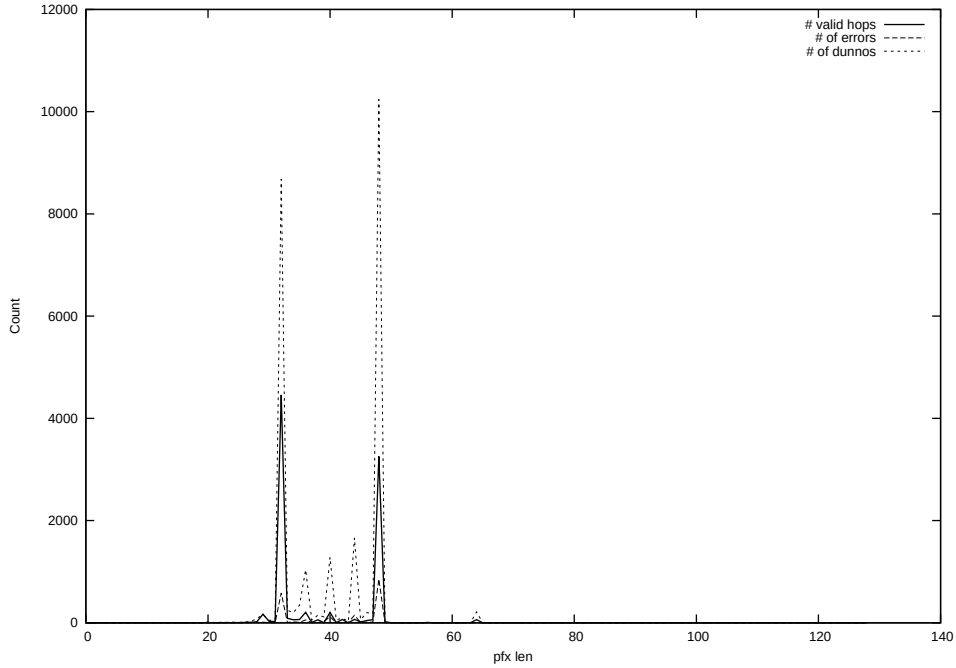
2013-02-25



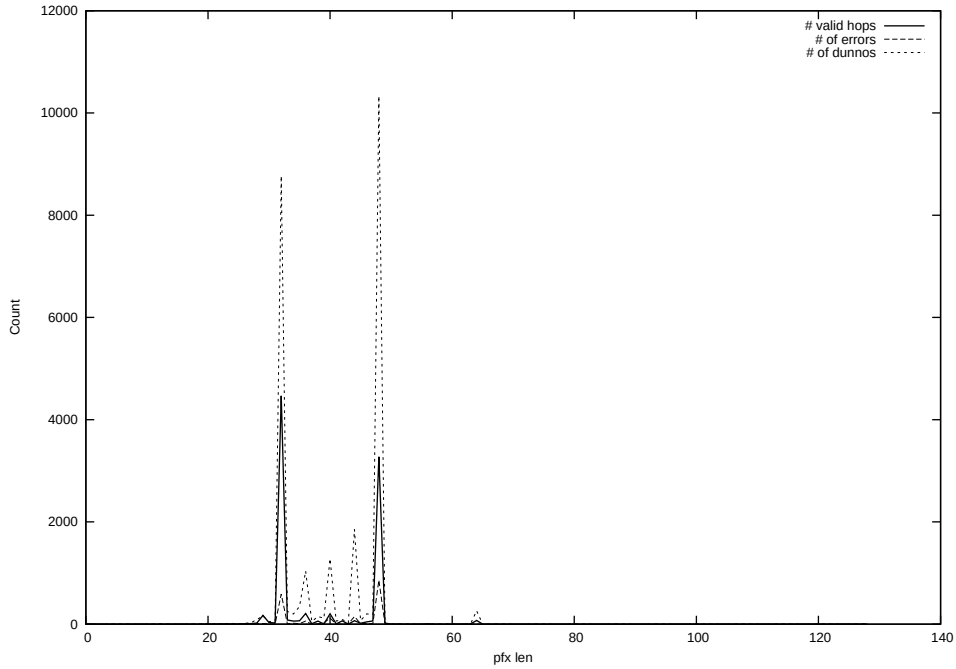
2013-02-26



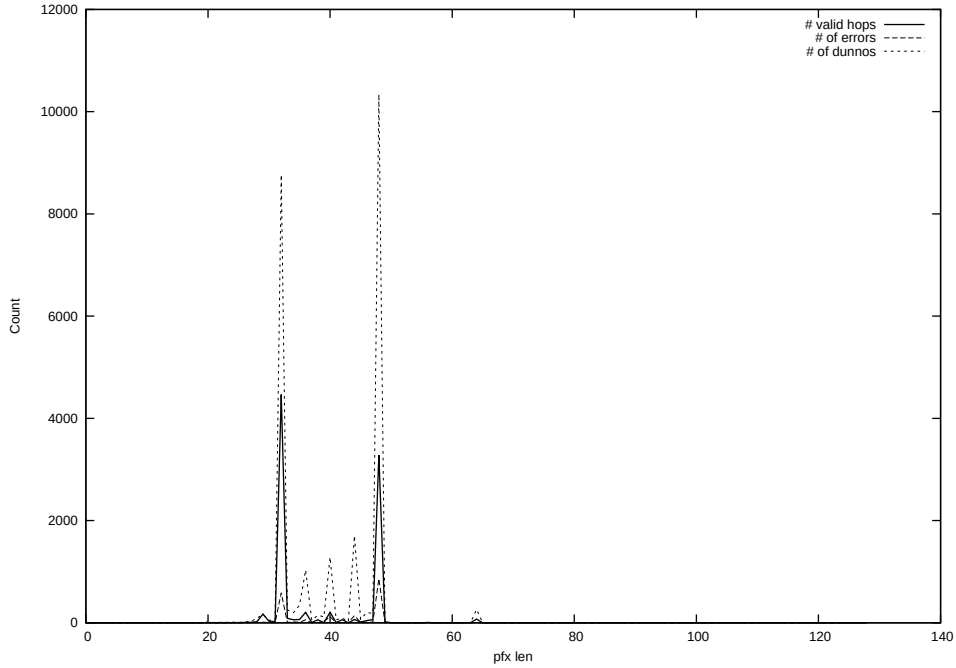
2013-02-27



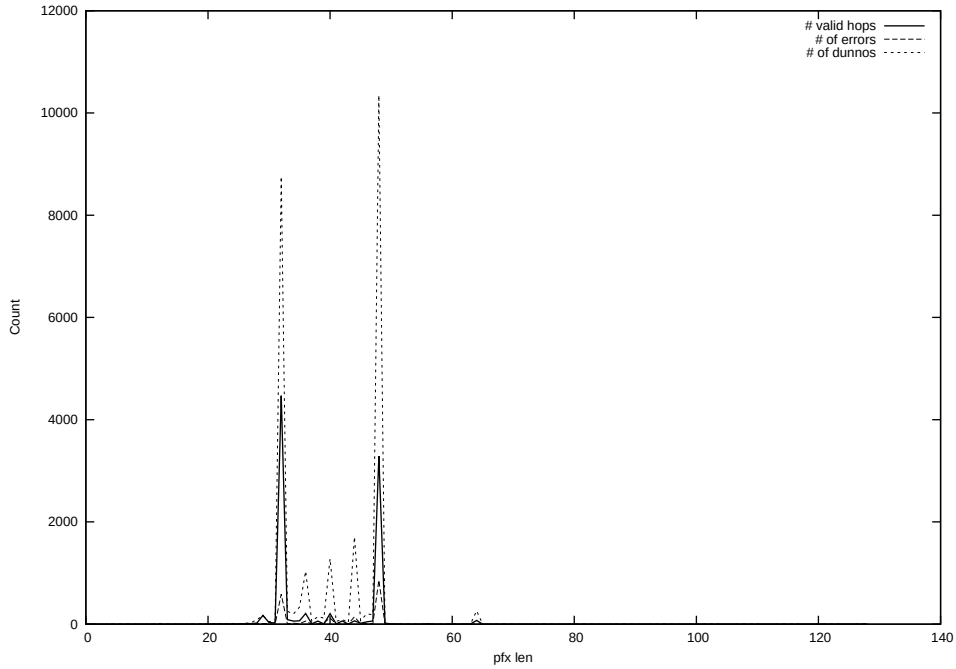
2013-02-28



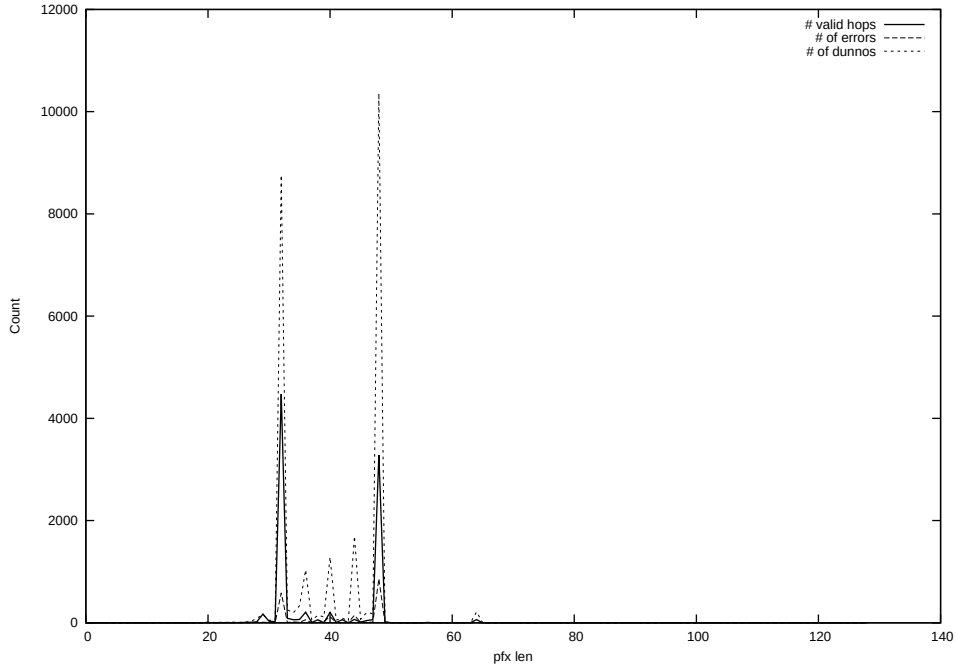
2013-03-01



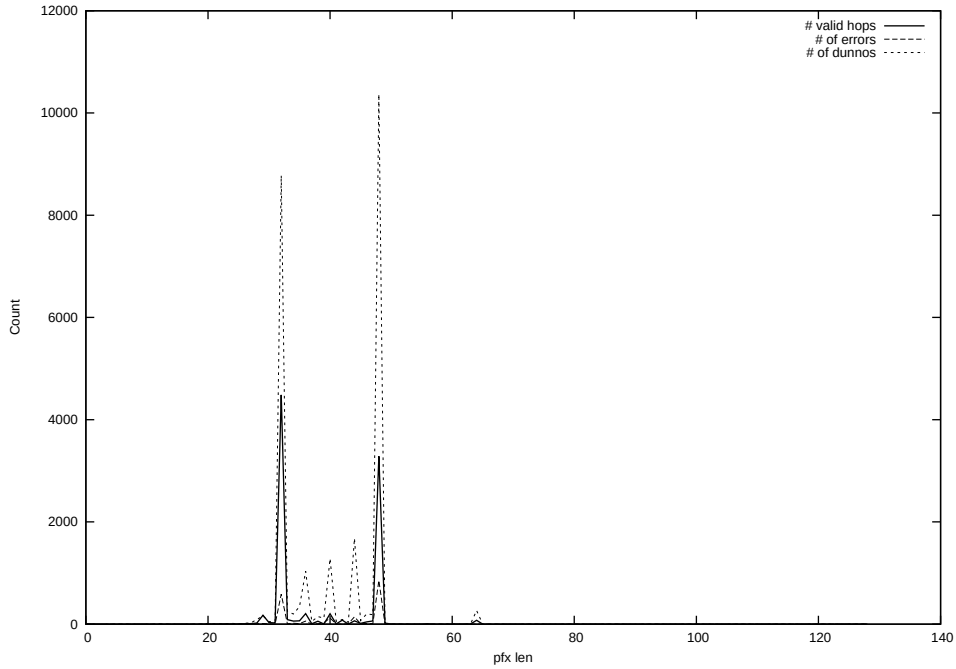
2013-03-02



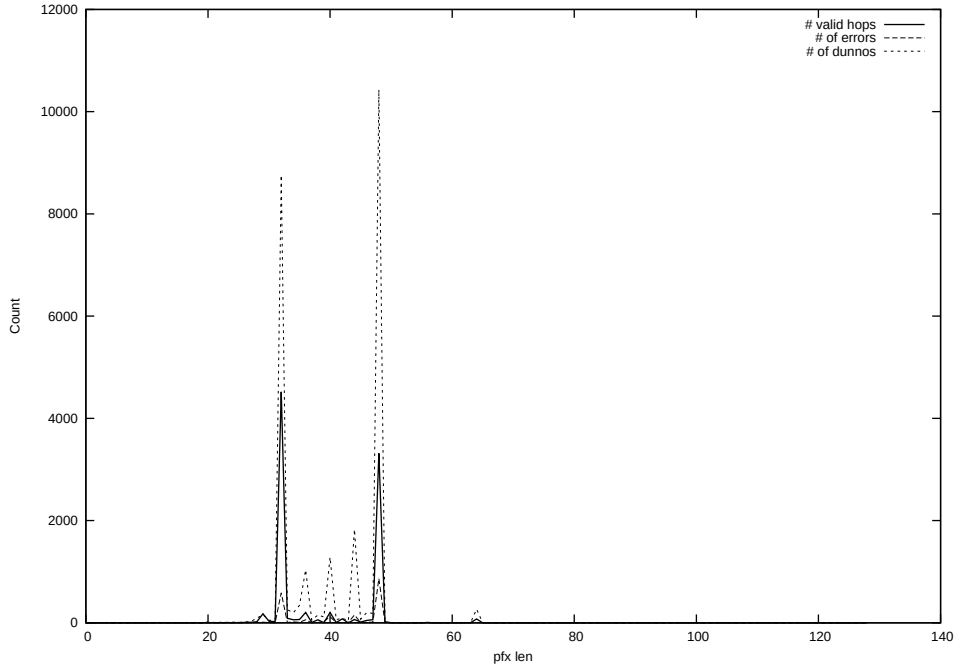
2013-03-03



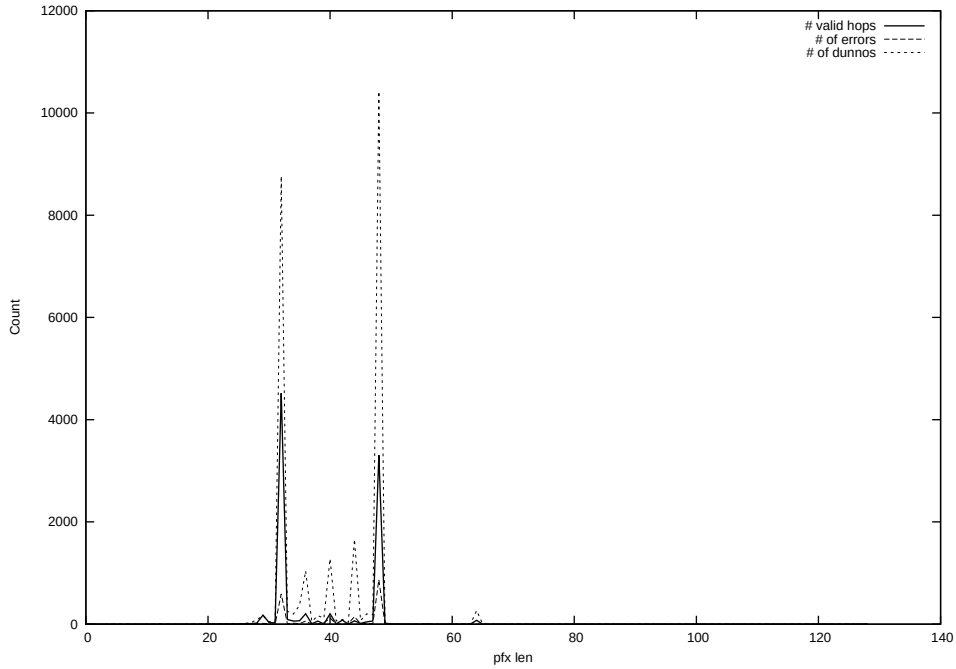
2013-03-04



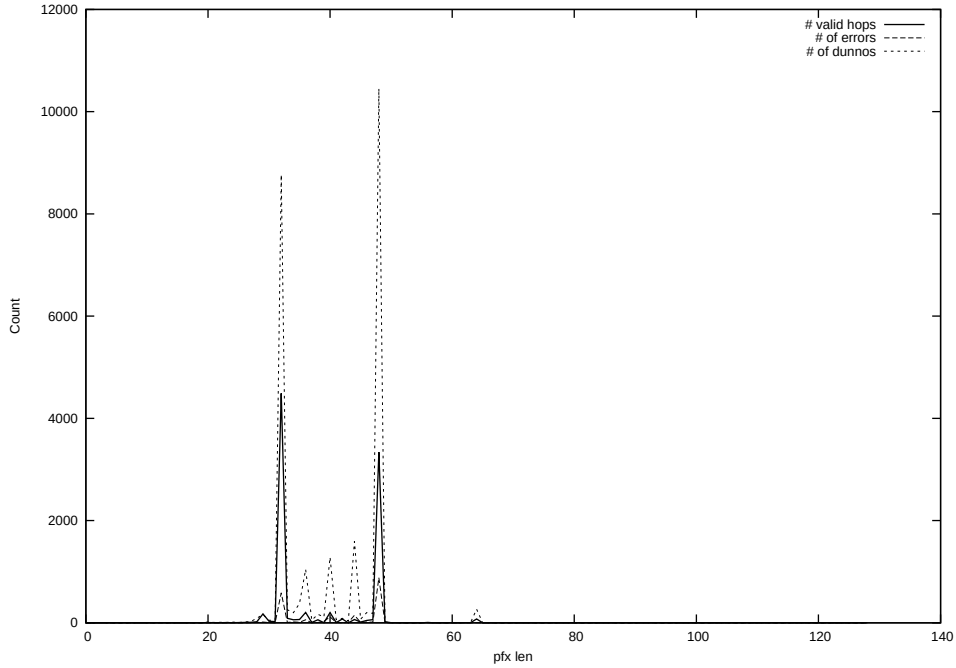
2013-03-05



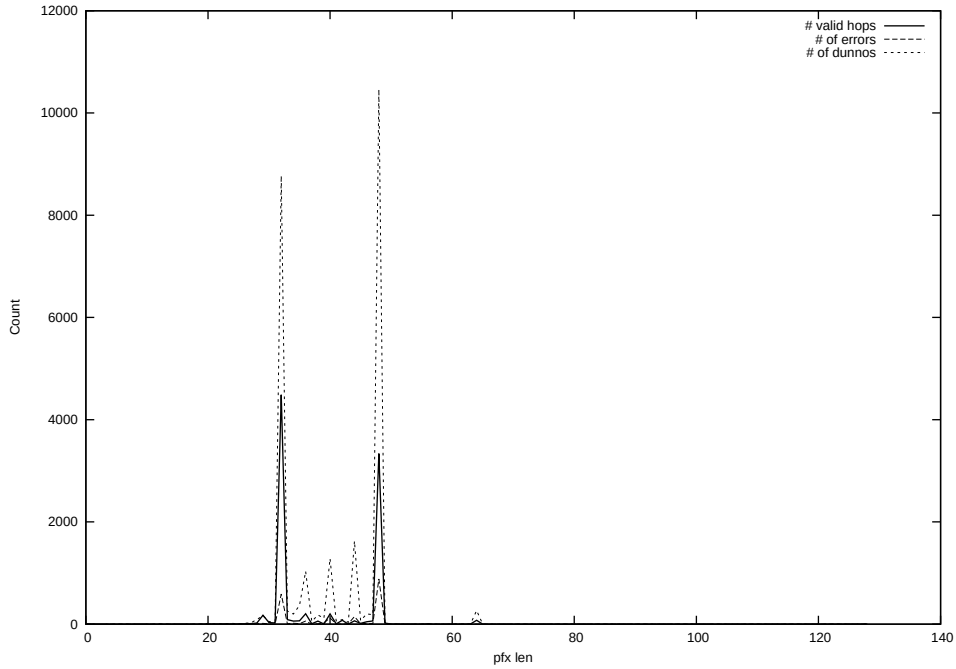
2013-03-06



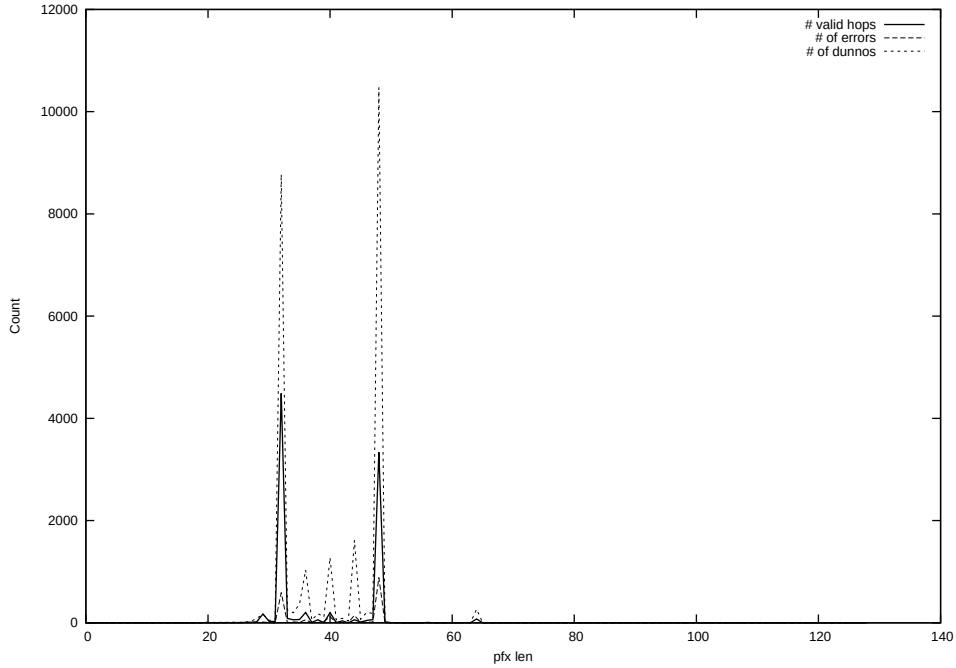
2013-03-07



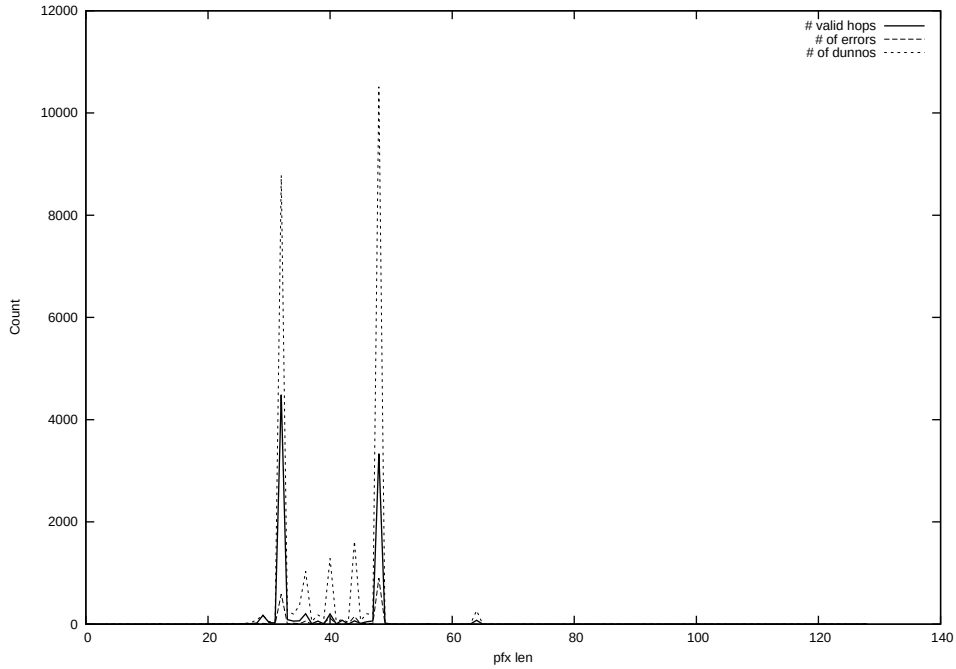
2013-03-08



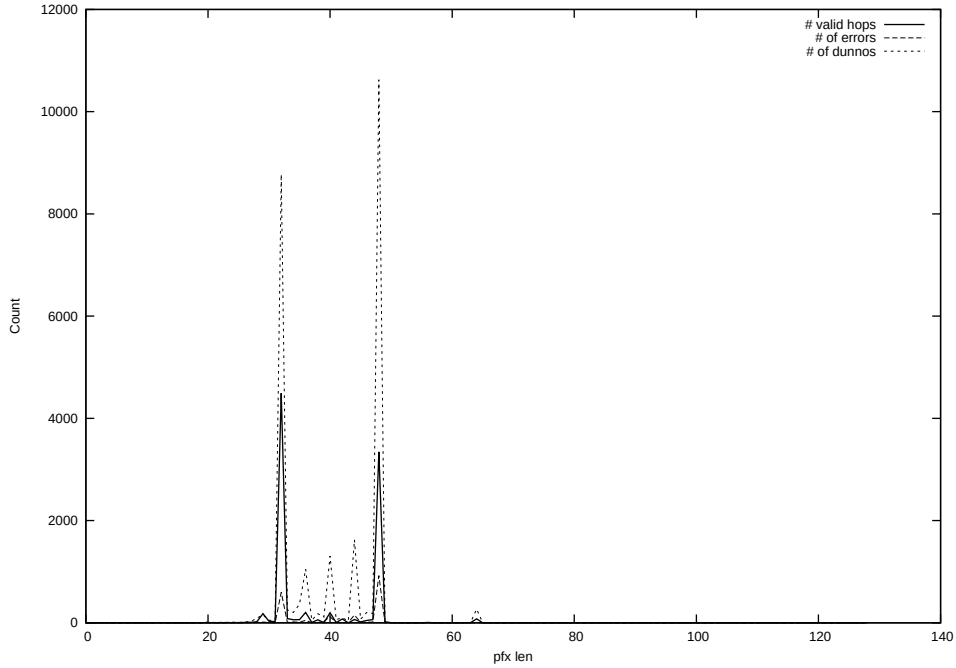
2013-03-09



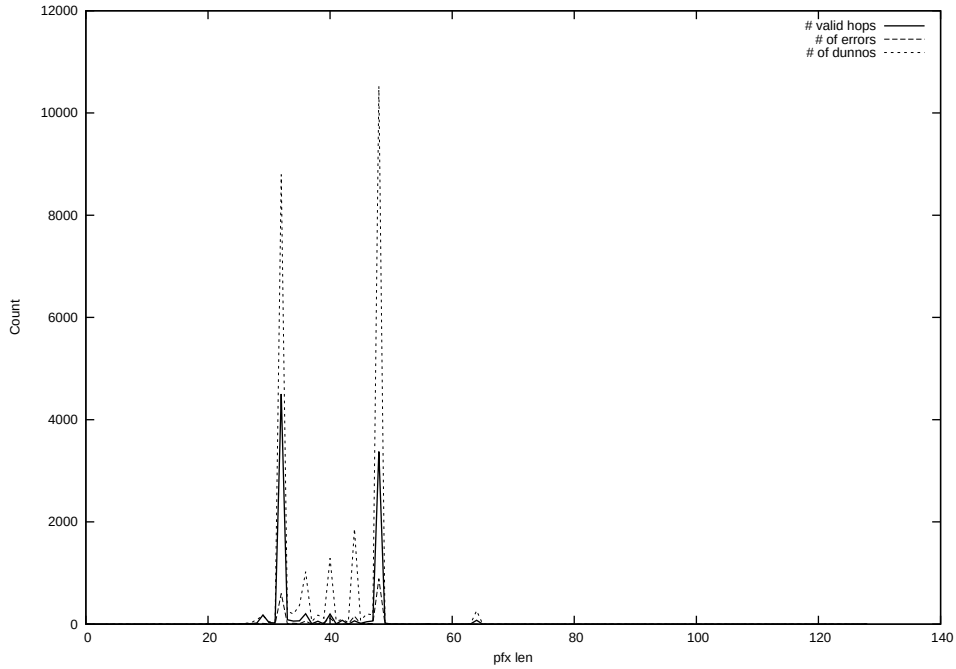
2013-03-10



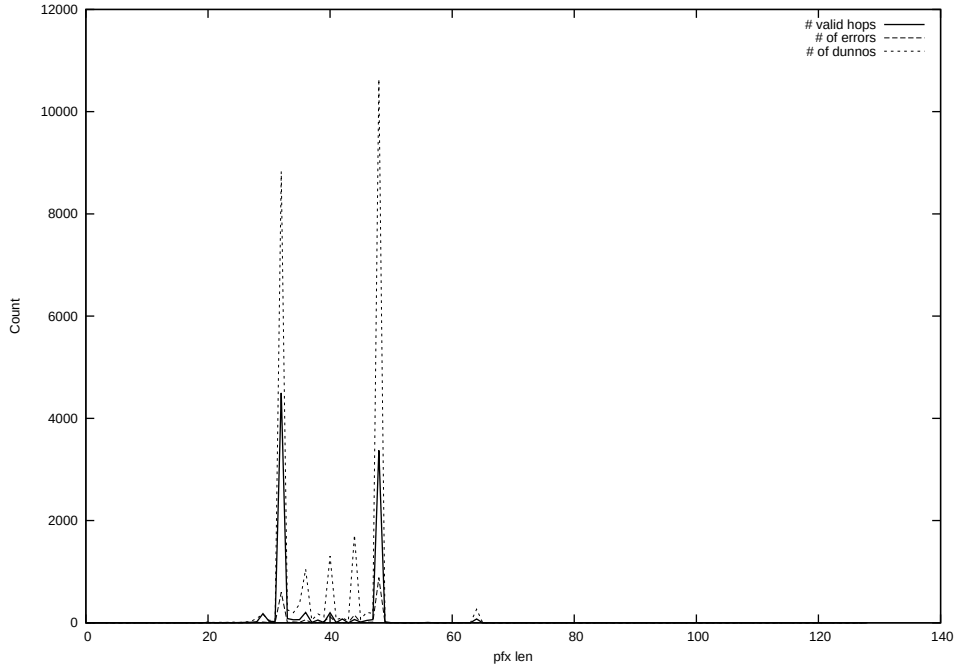
2013-03-11



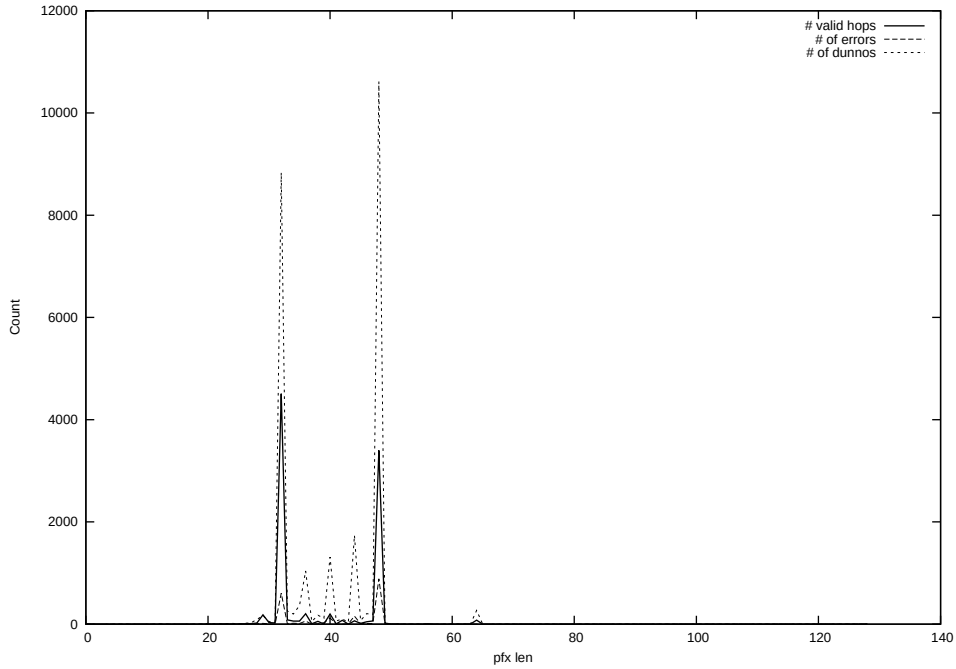
2013-03-12



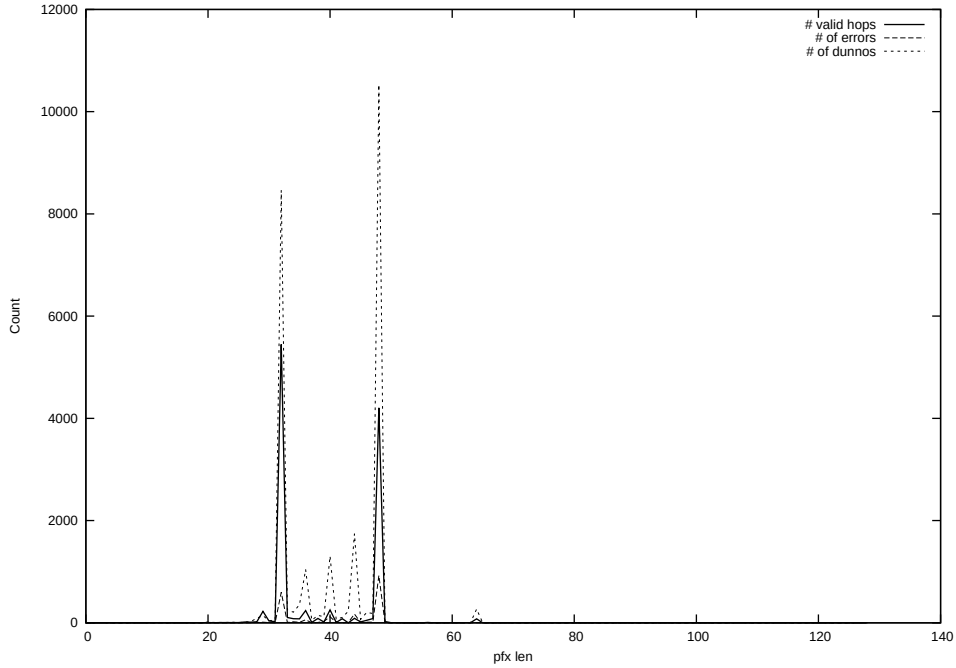
2013-03-13



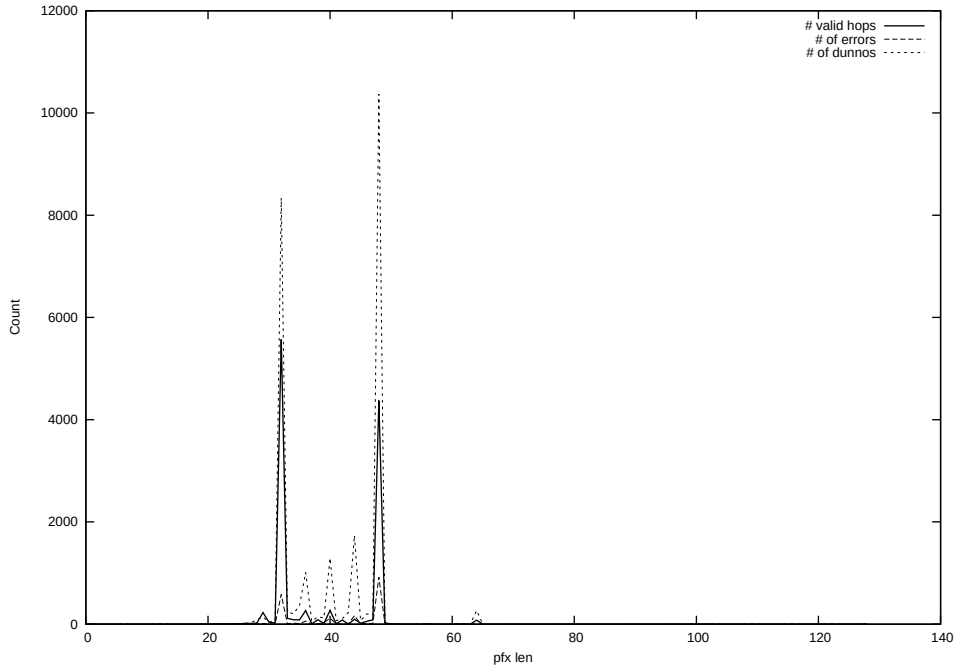
2013-03-14



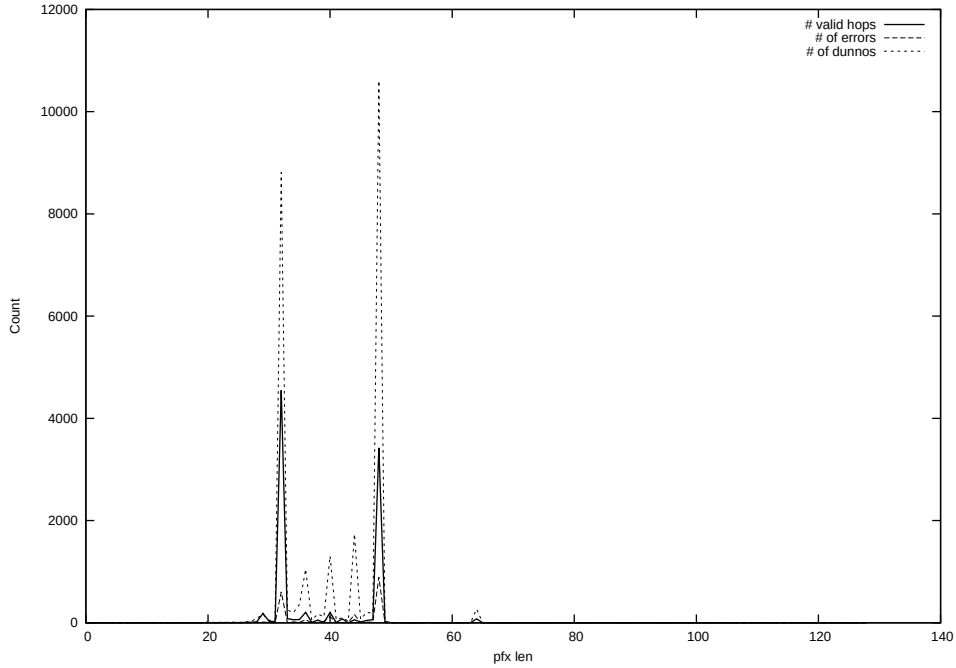
2013-03-15



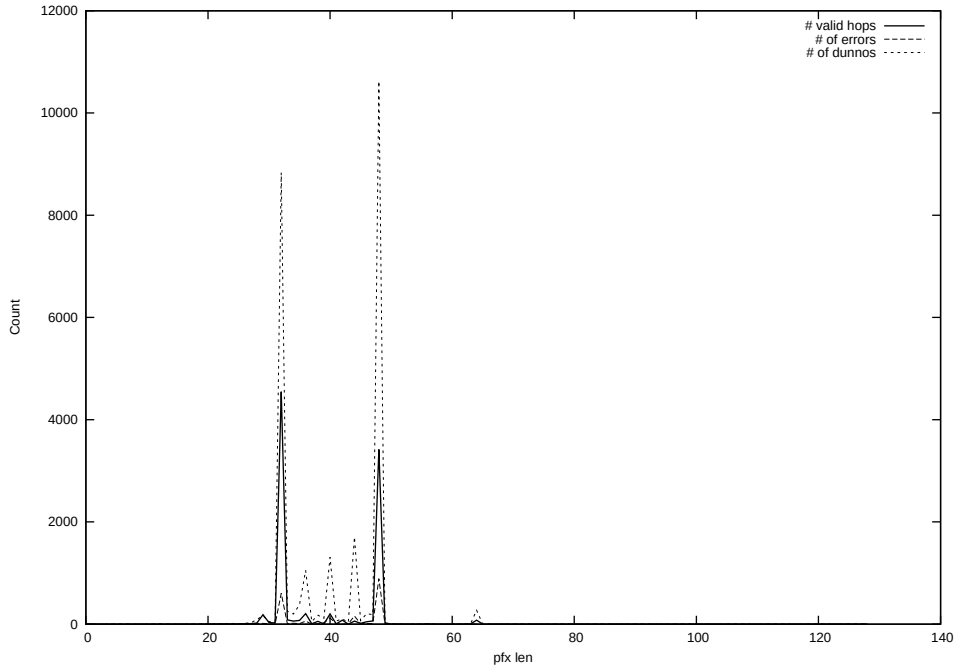
2013-03-16



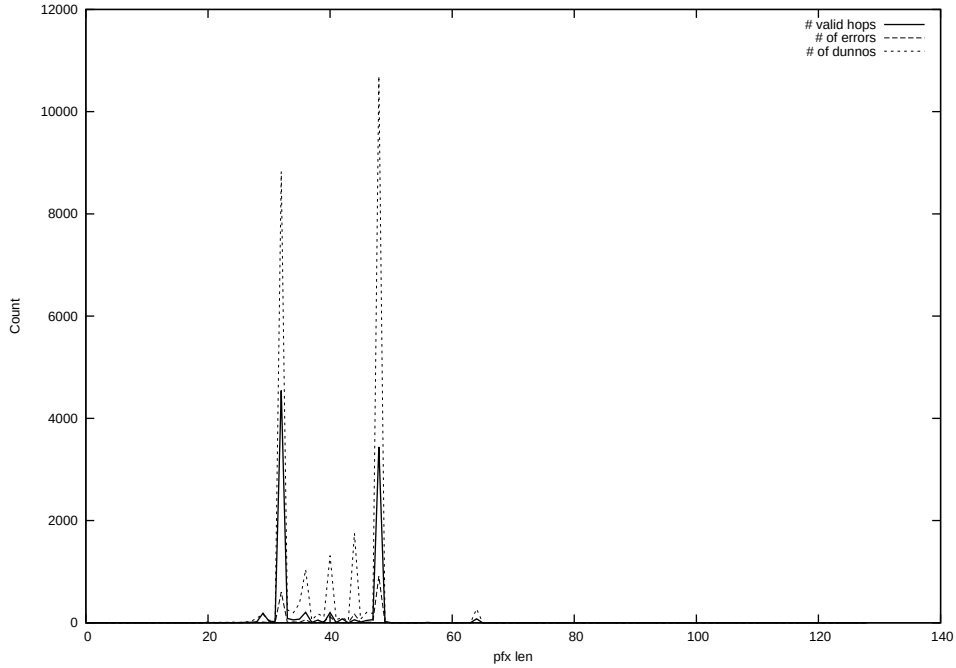
2013-03-17



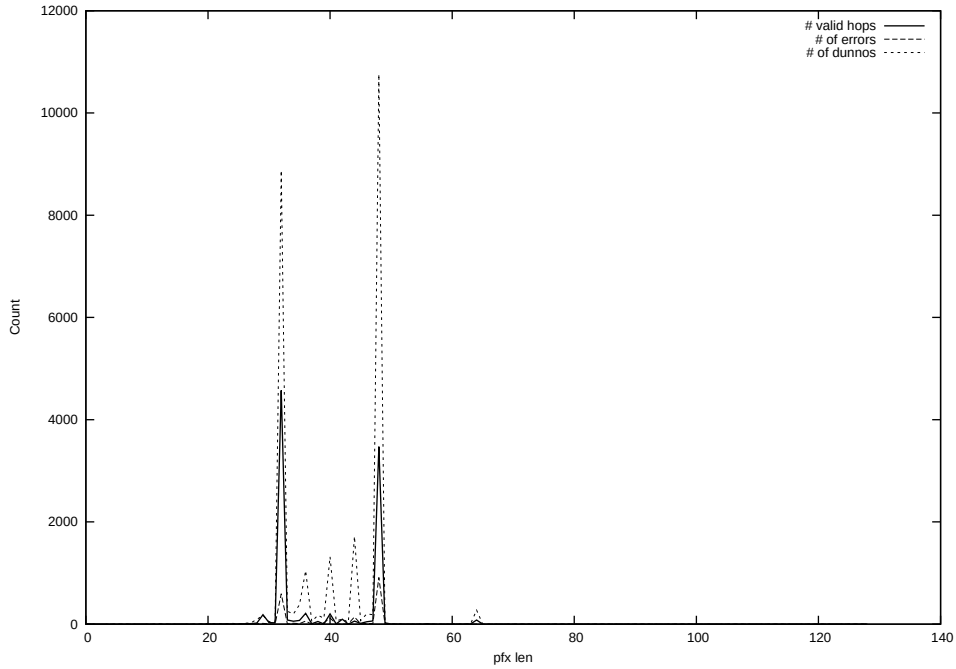
2013-03-18



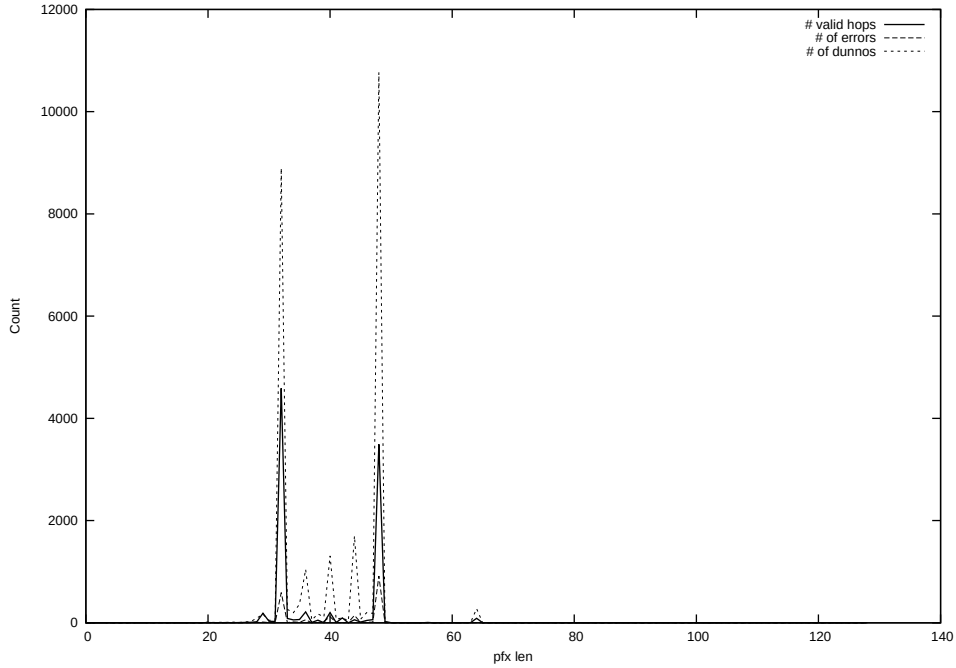
2013-03-19



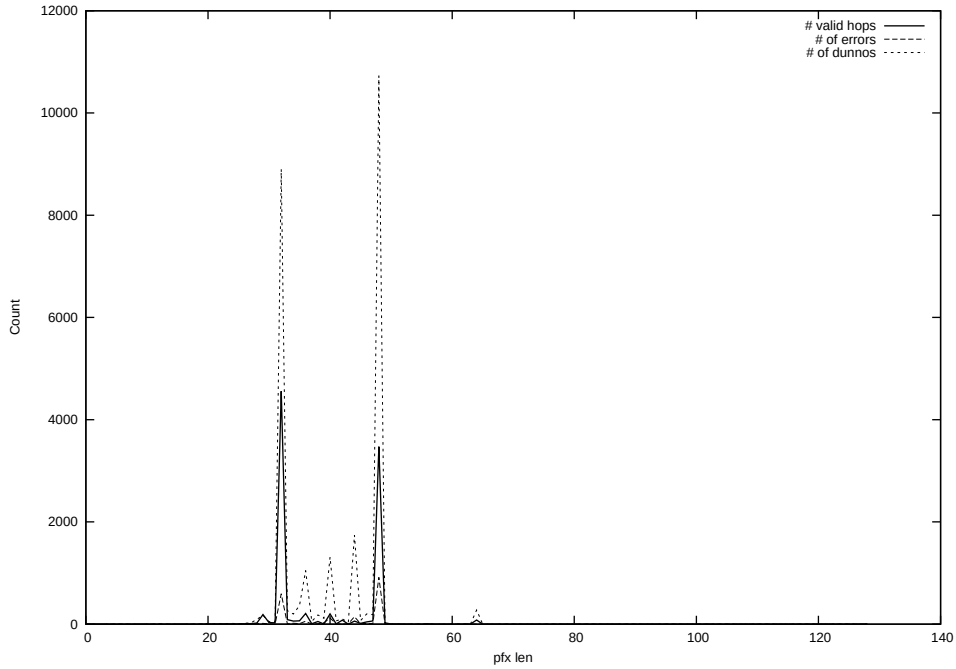
2013-03-20



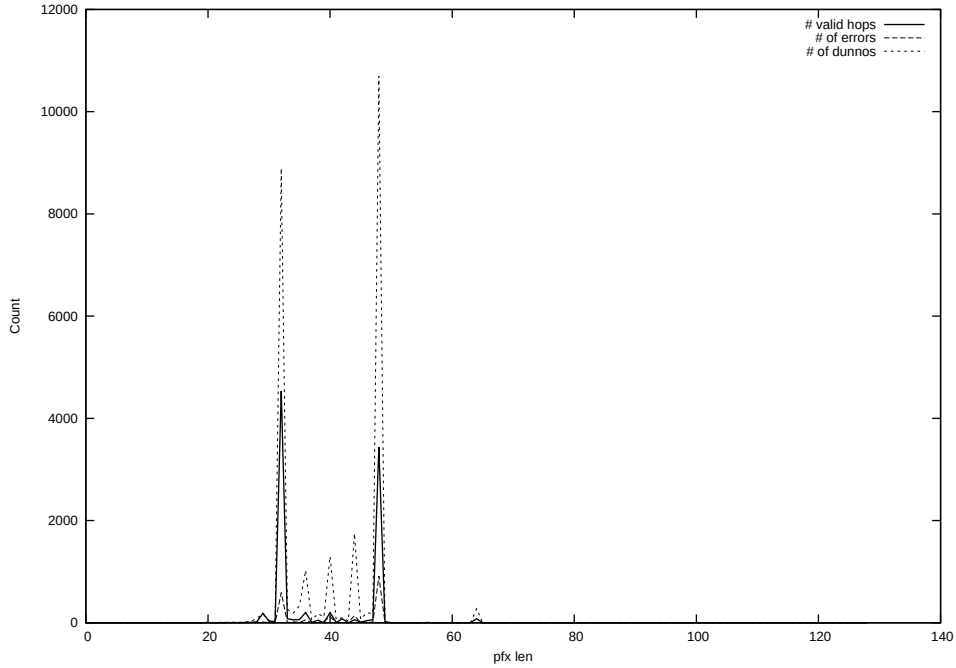
2013-03-21



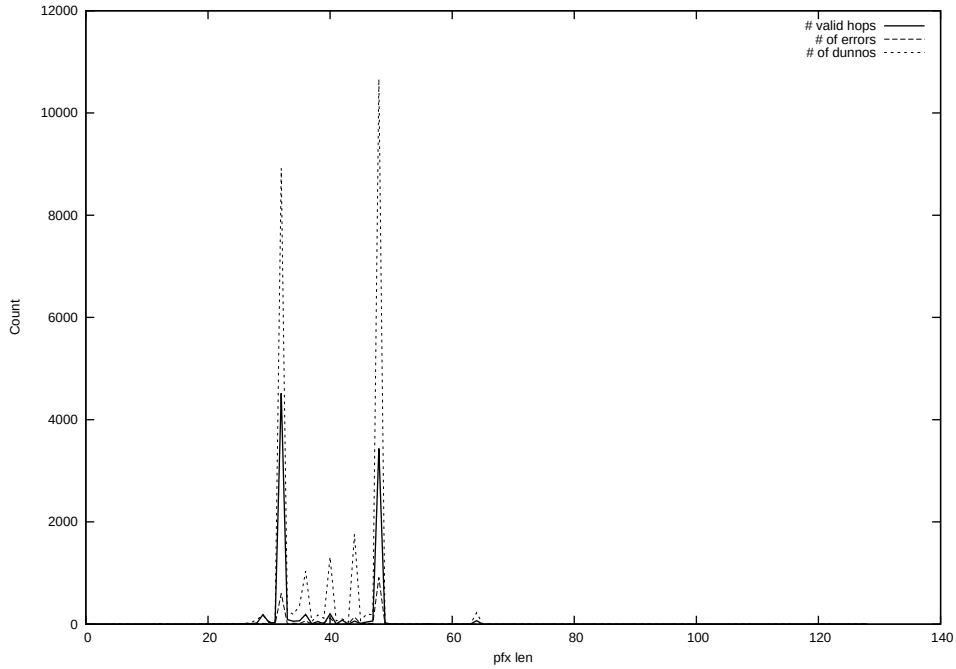
2013-03-22



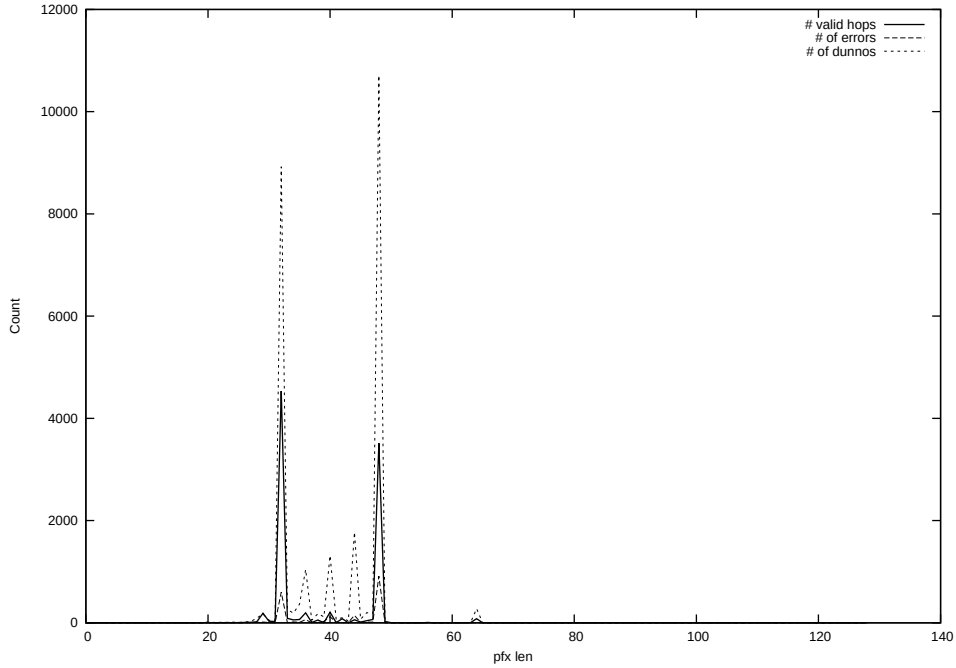
2013-03-23



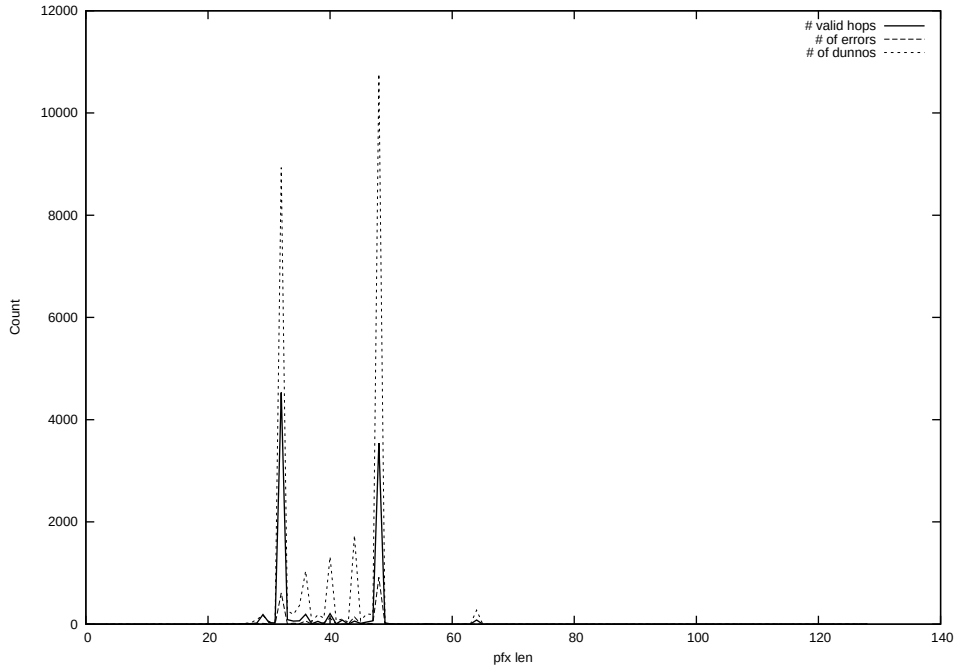
2013-03-24



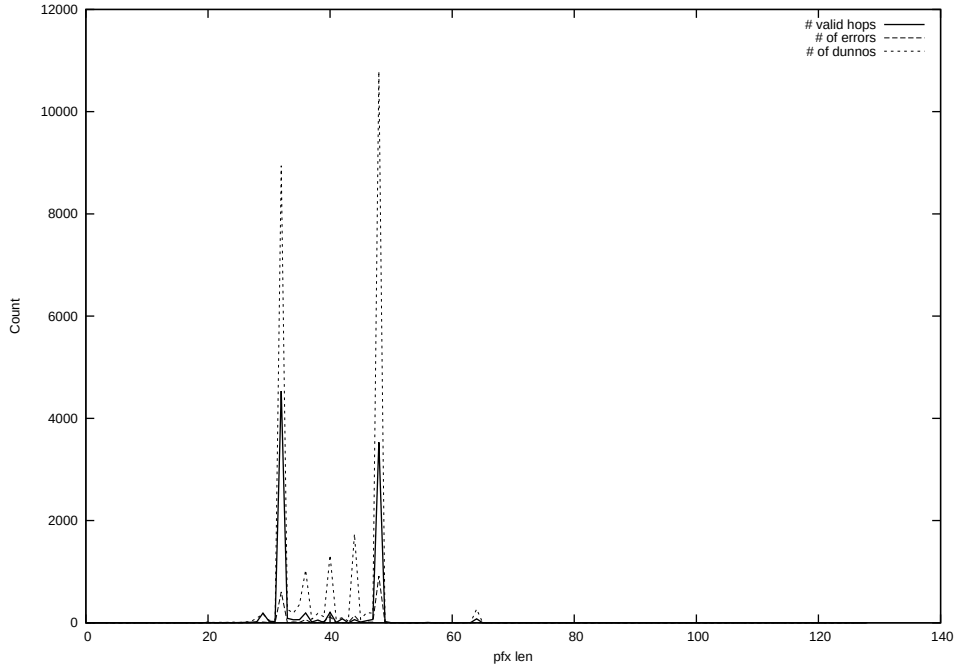
2013-03-25



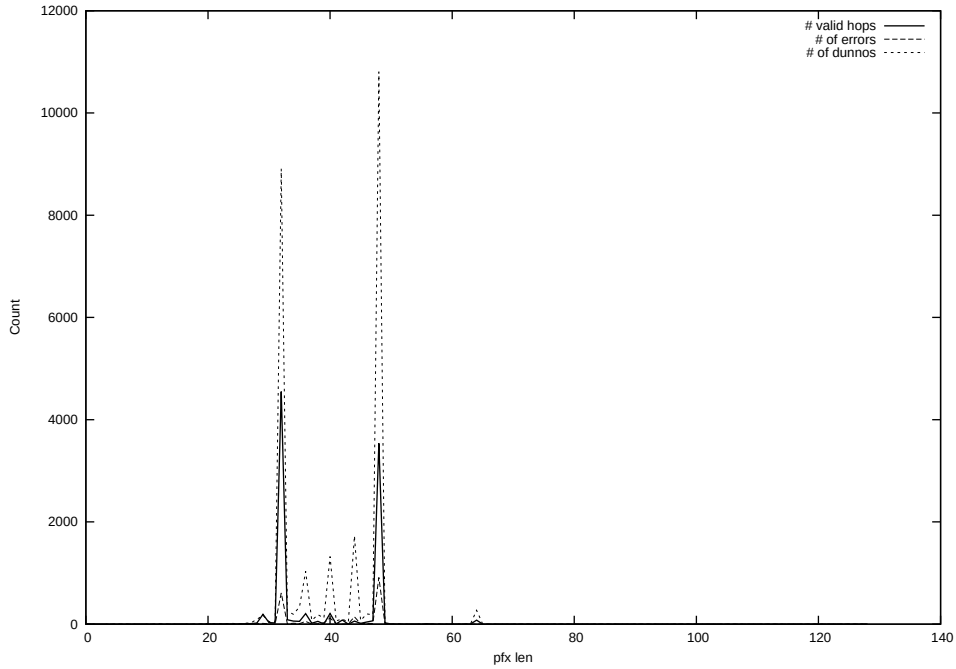
2013-03-26



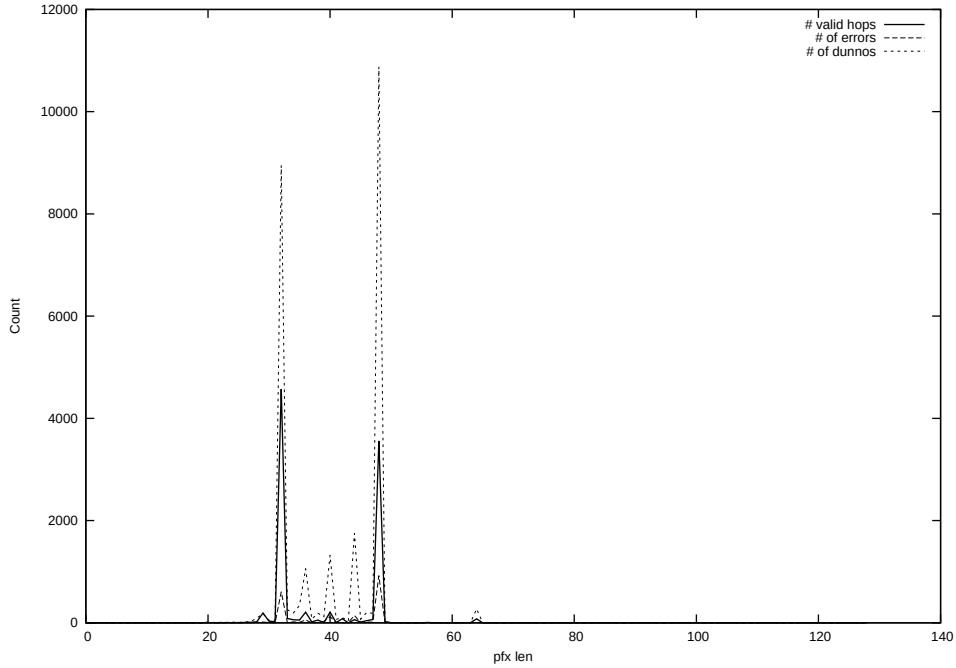
2013-03-27



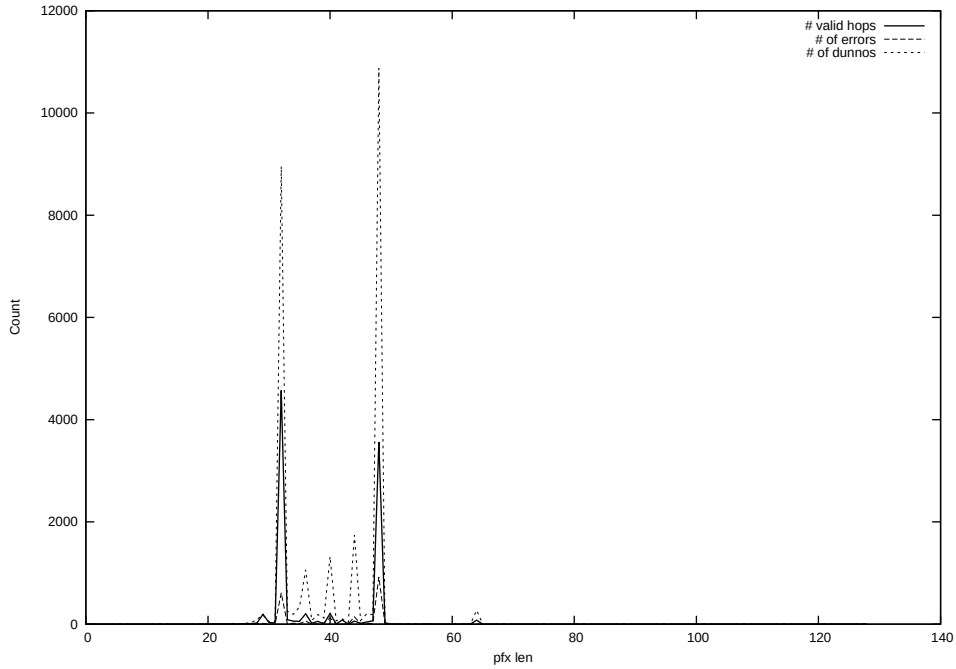
2013-03-28



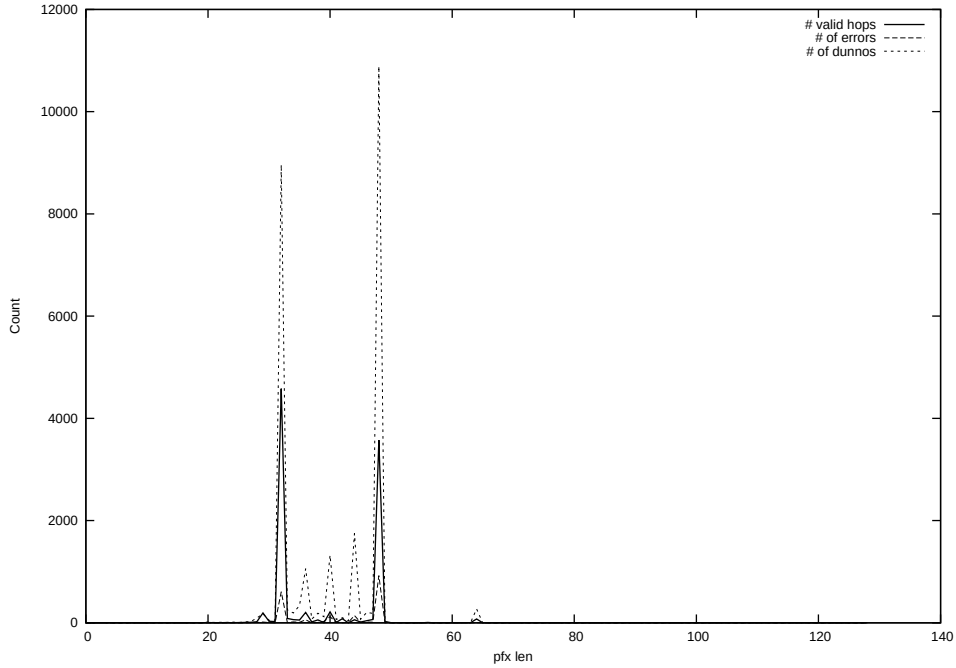
2013-03-29



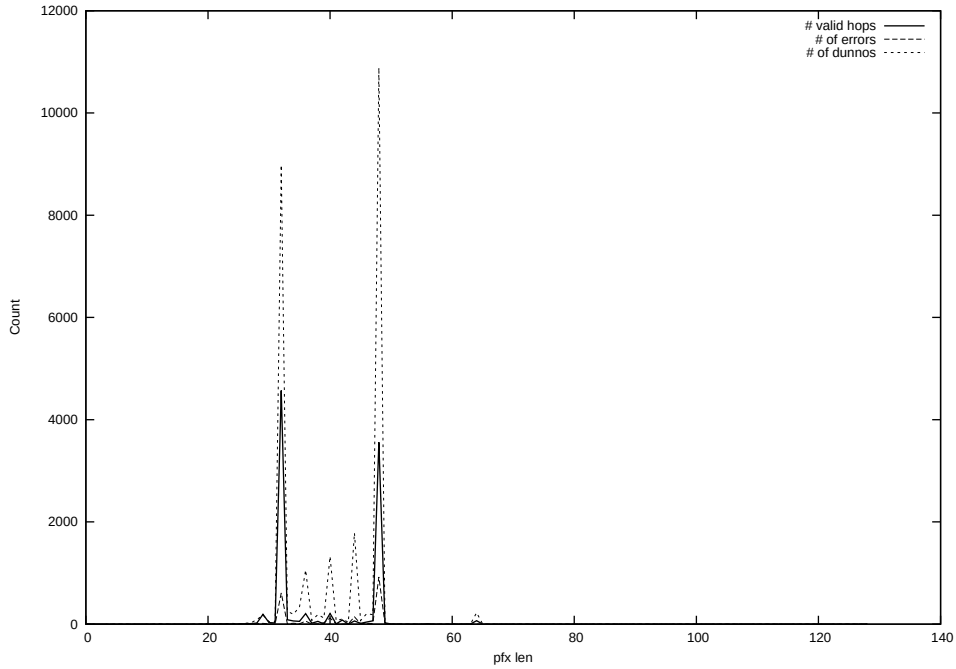
2013-03-30



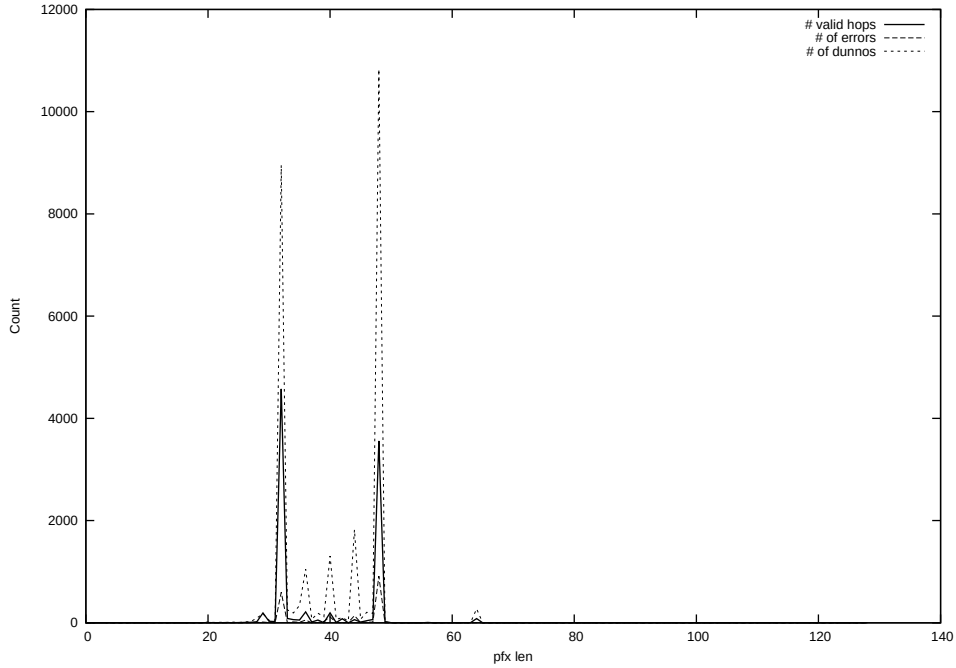
2013-03-31



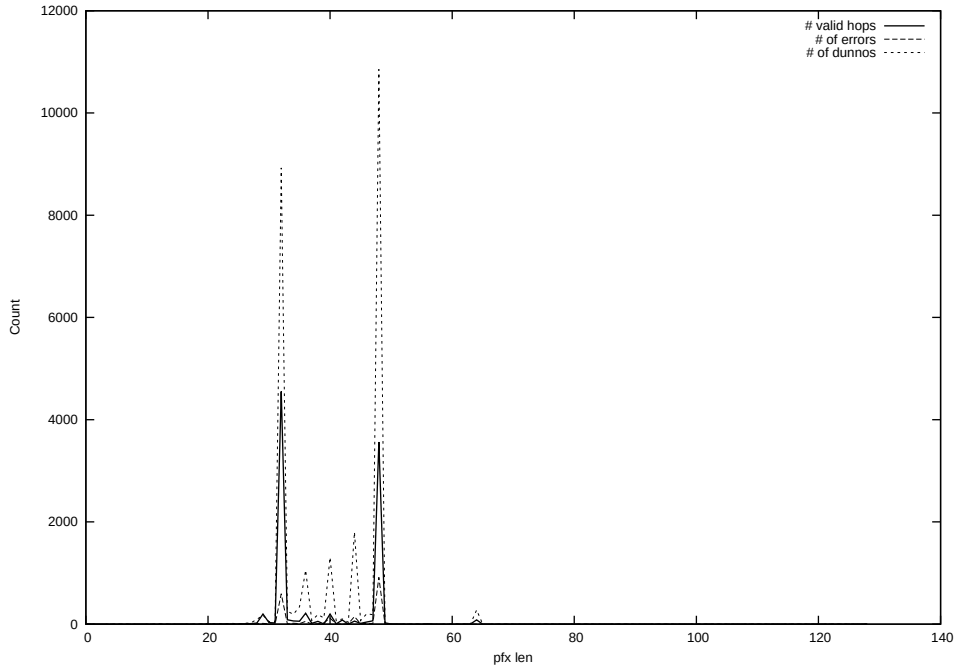
2013-04-01



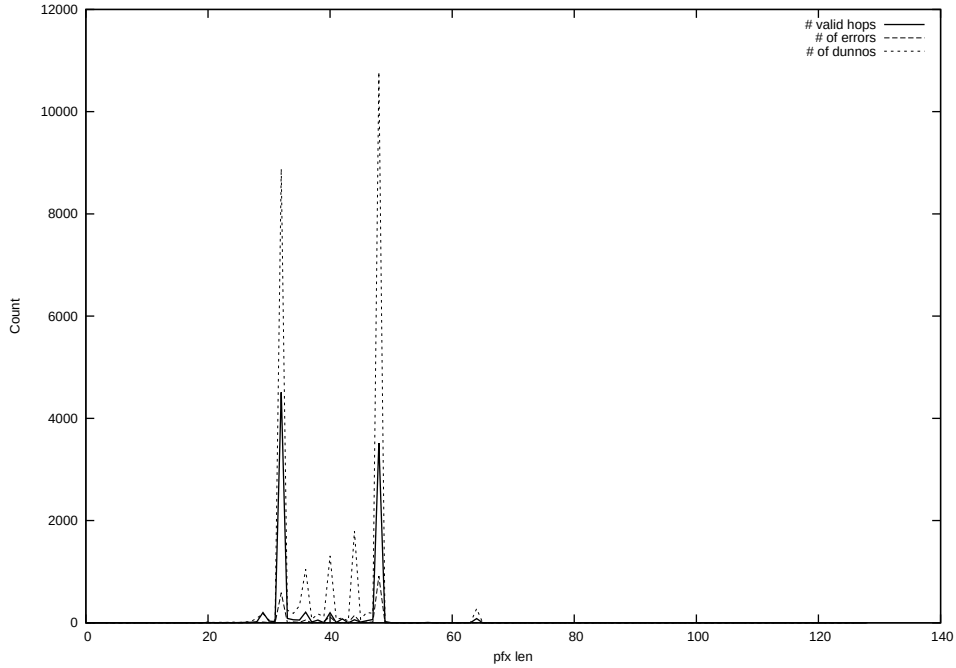
2013-04-02



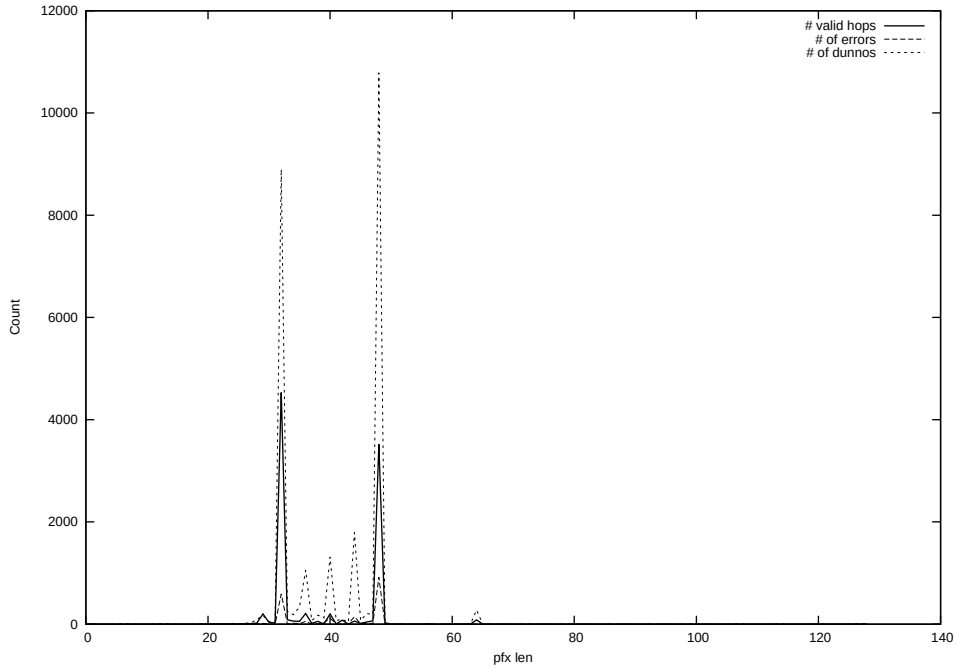
2013-04-03



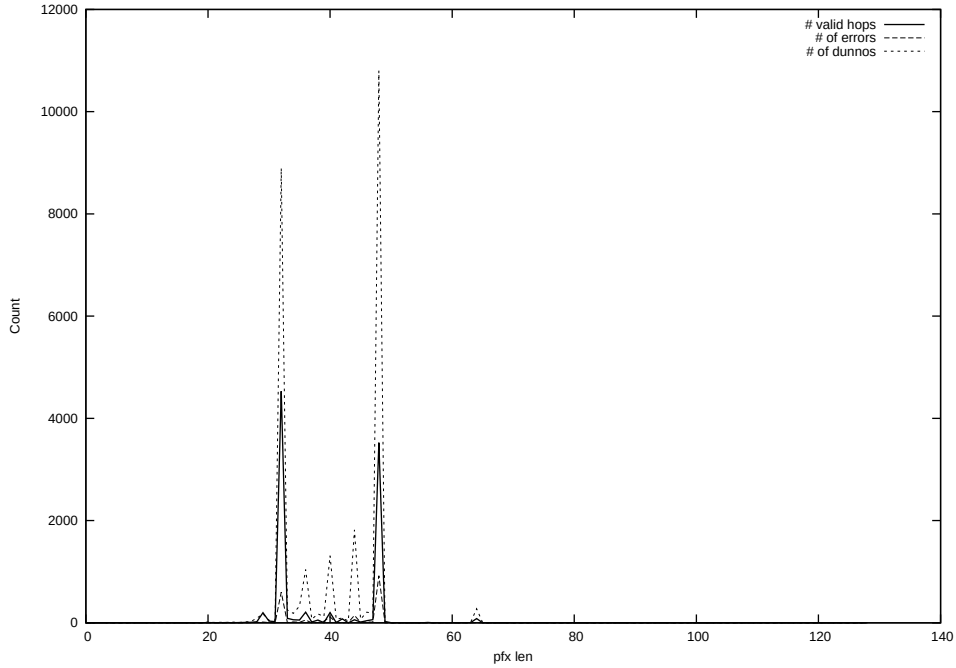
2013-04-04



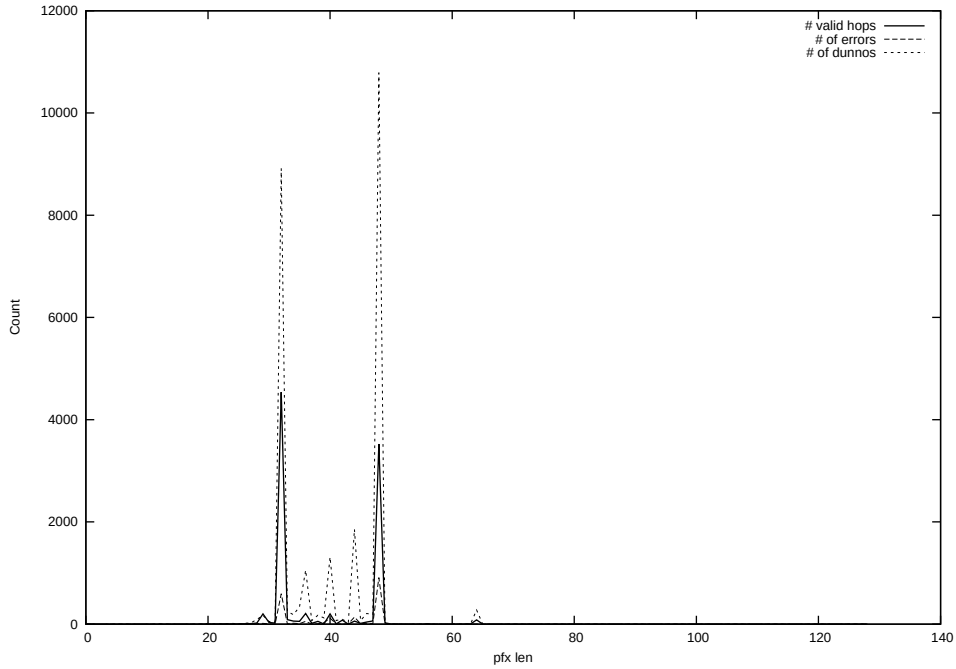
2013-04-05



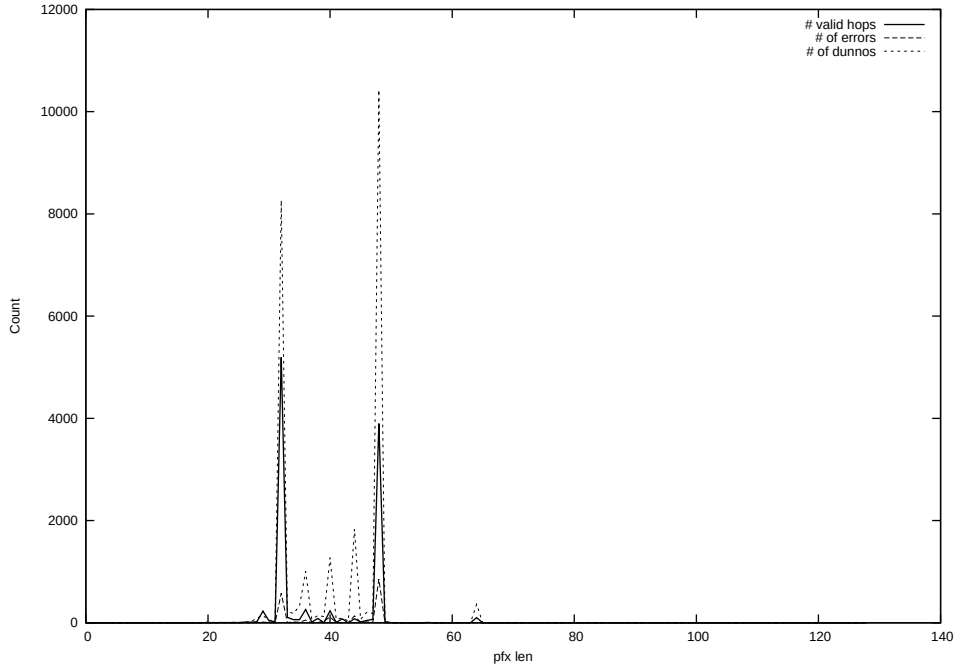
2013-04-06



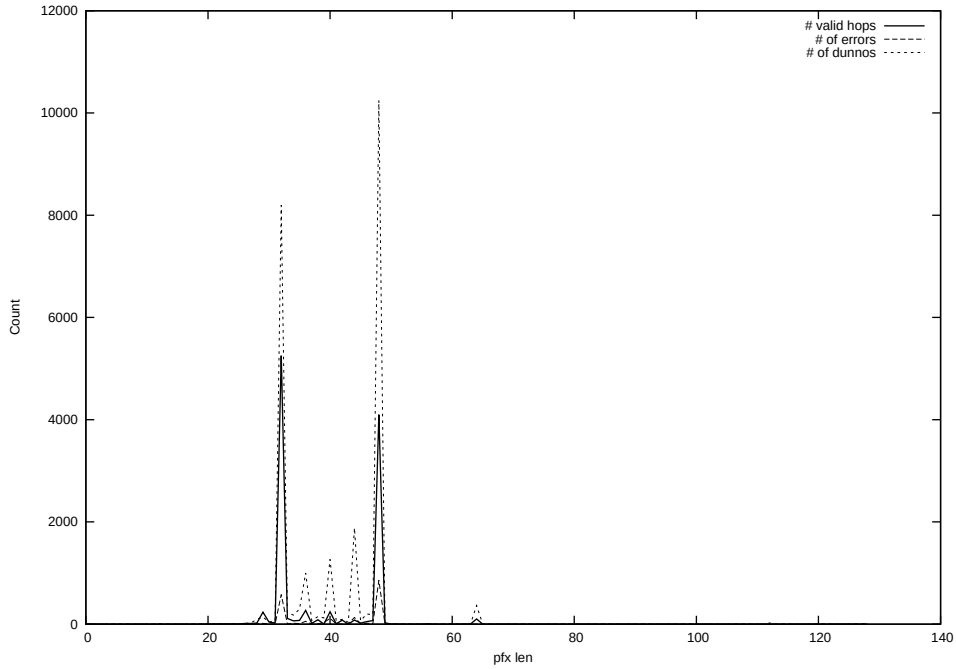
2013-04-07



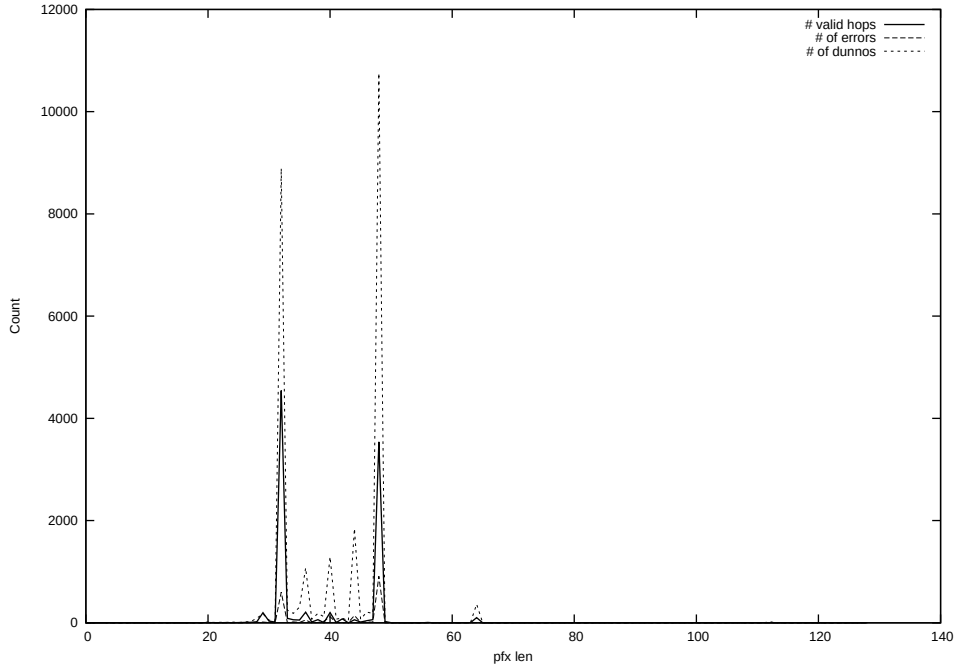
2013-04-08



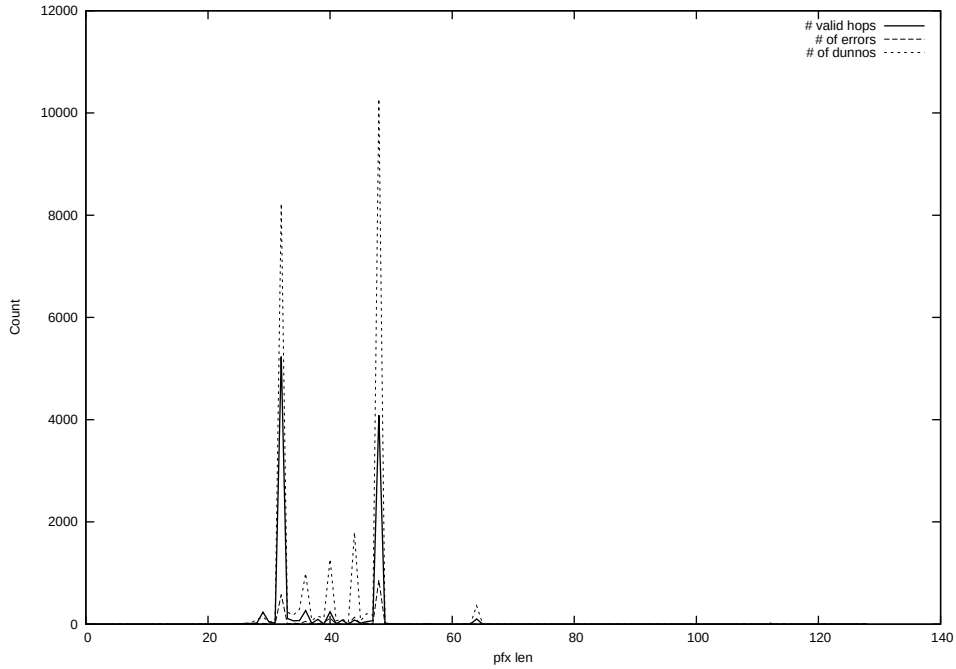
2013-04-09



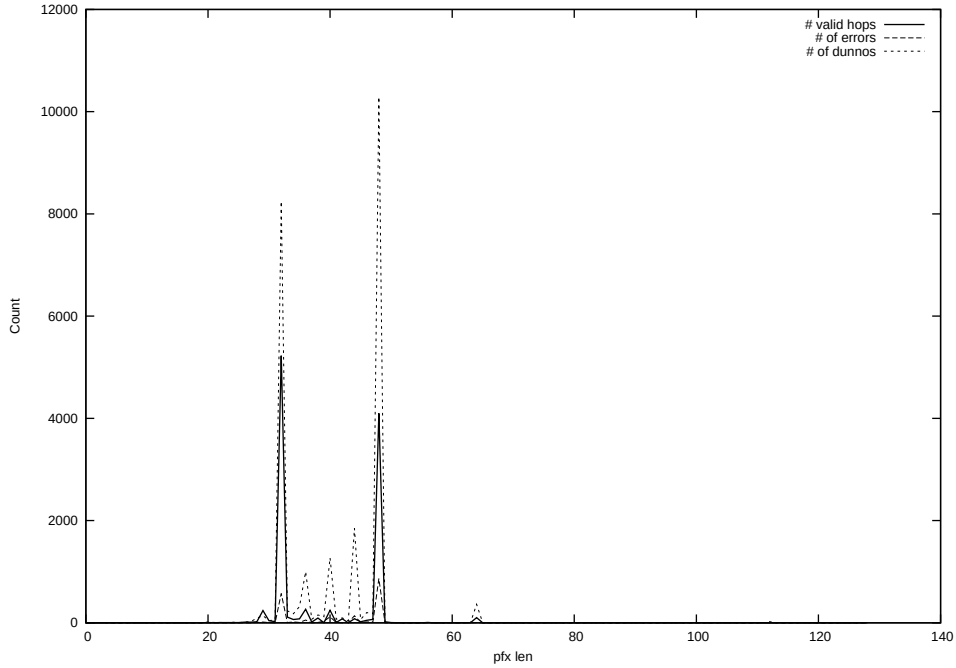
2013-04-10



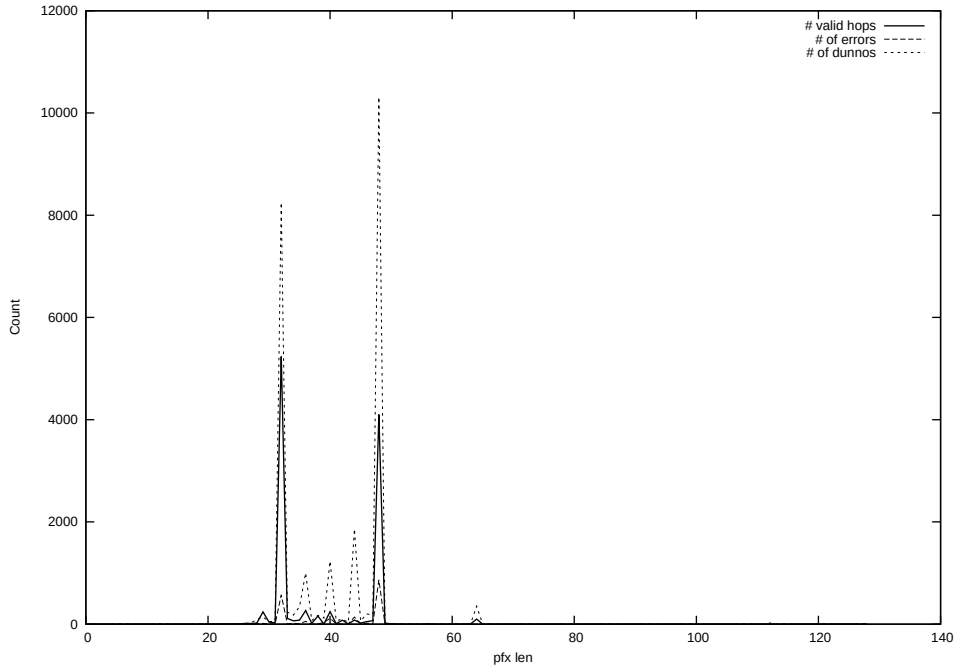
2013-04-11



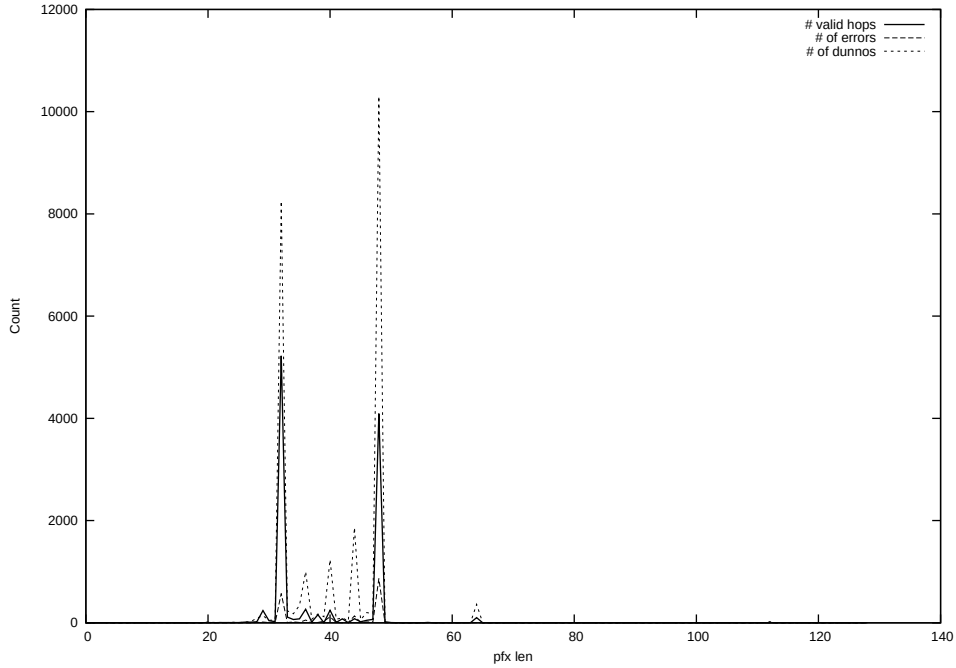
2013-04-12



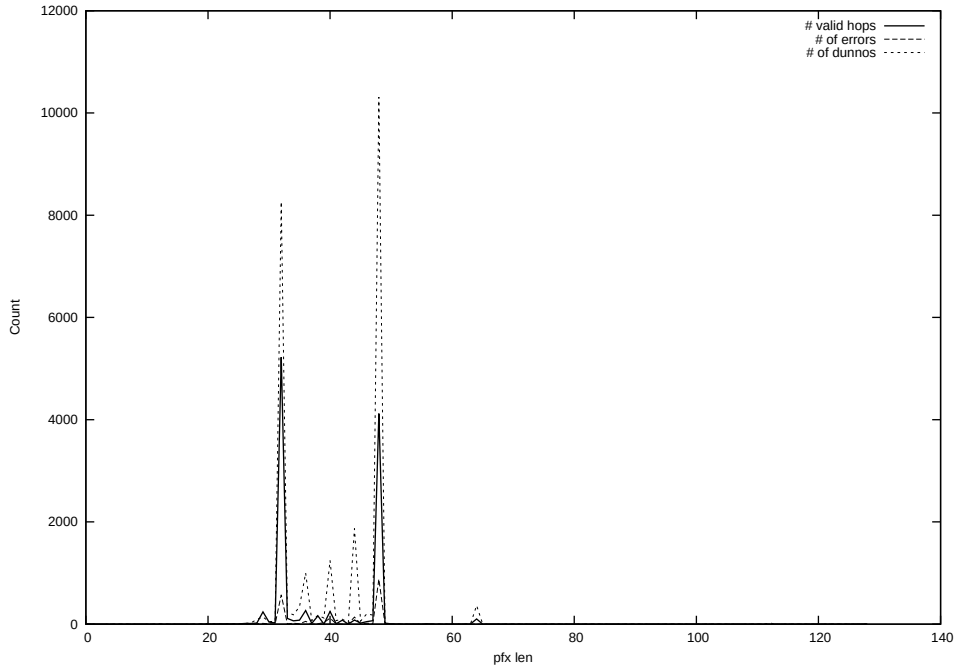
2013-04-13



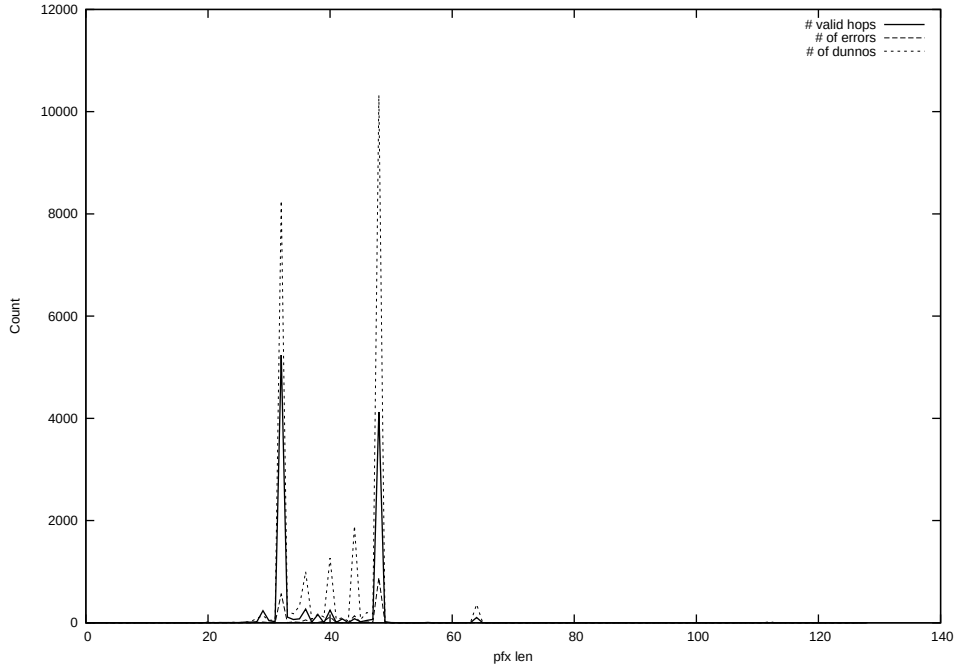
2013-04-14



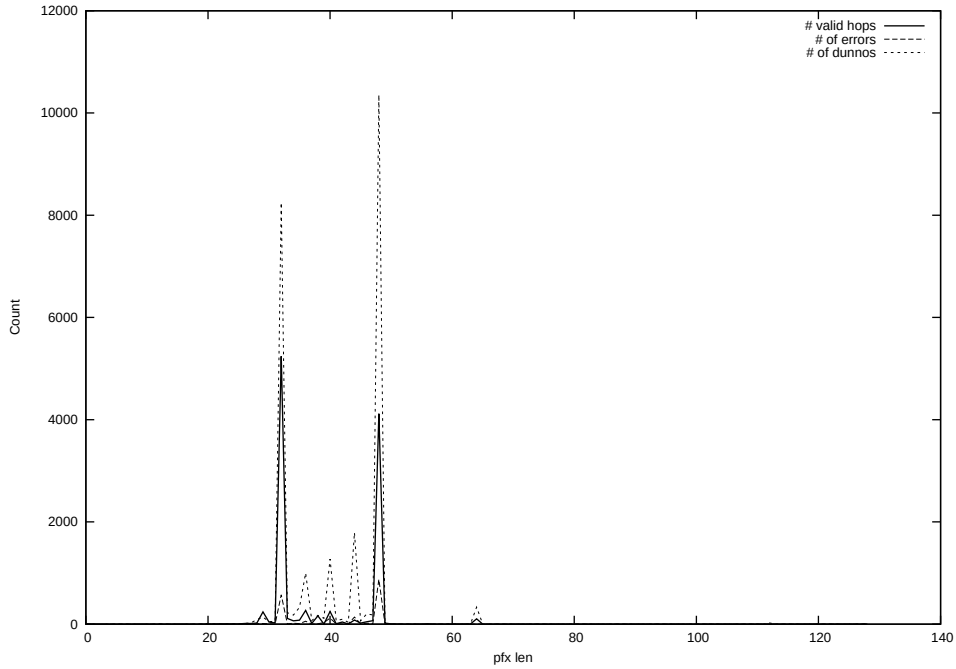
2013-04-15



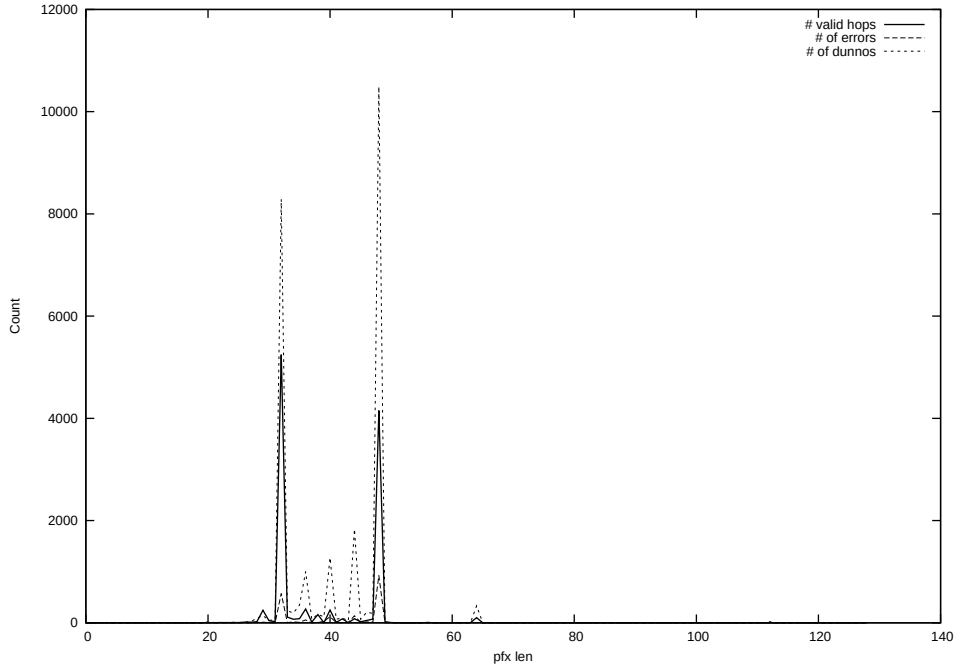
2013-04-16



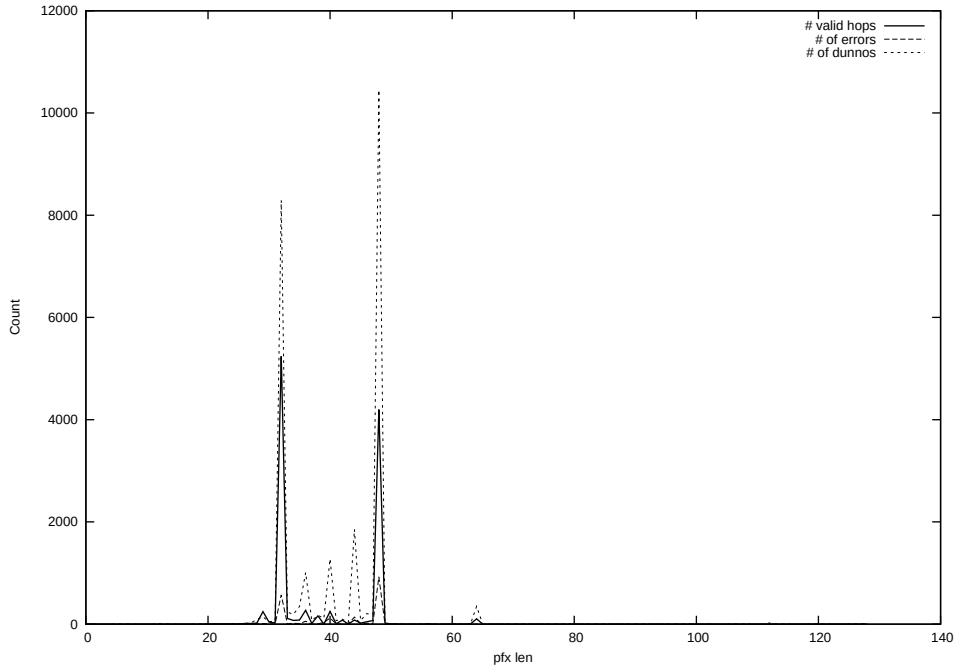
2013-04-17



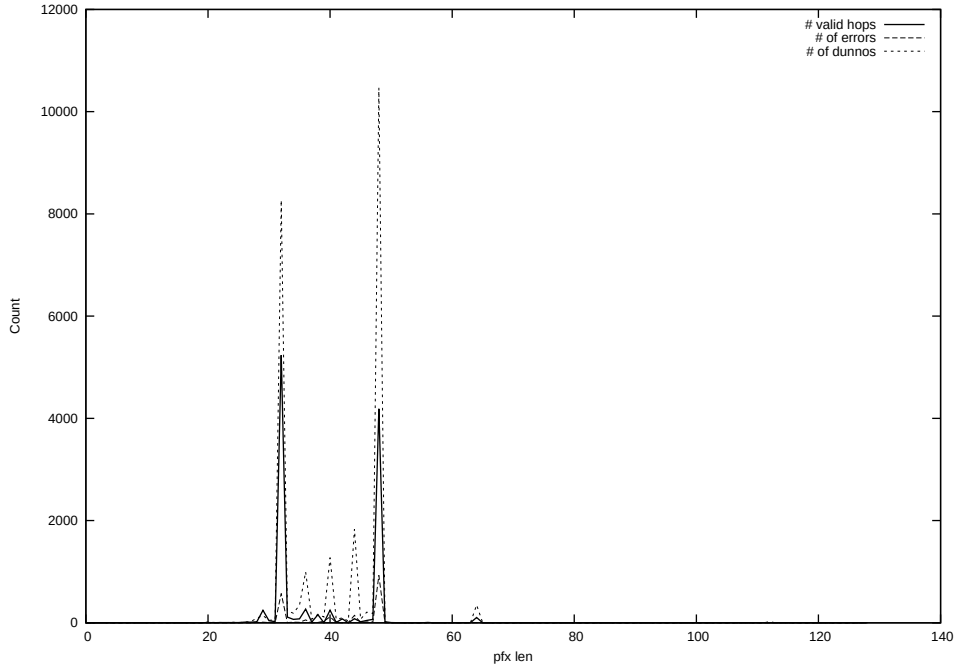
2013-04-18



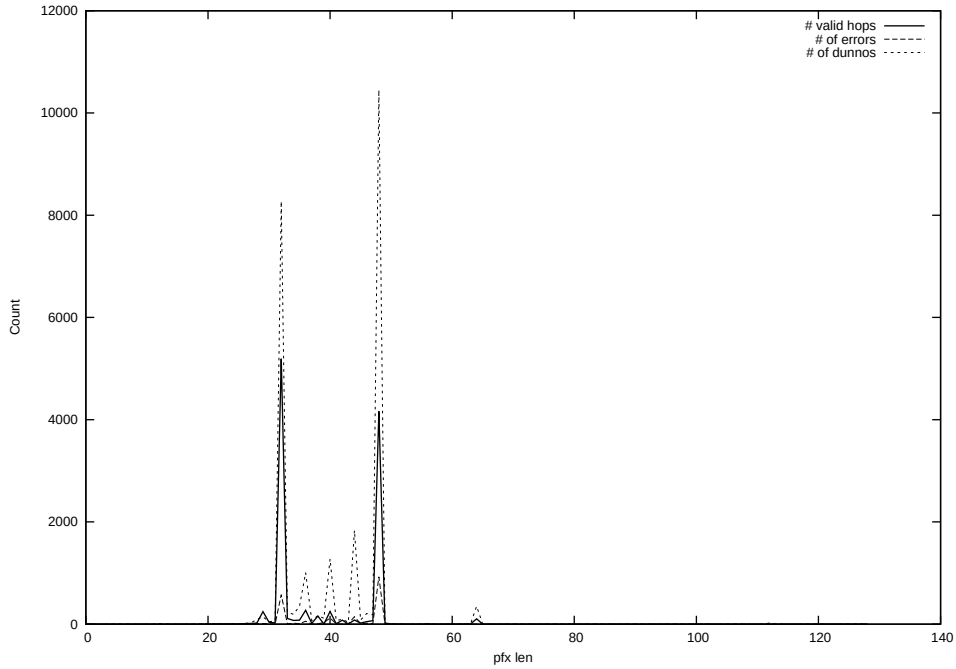
2013-04-19



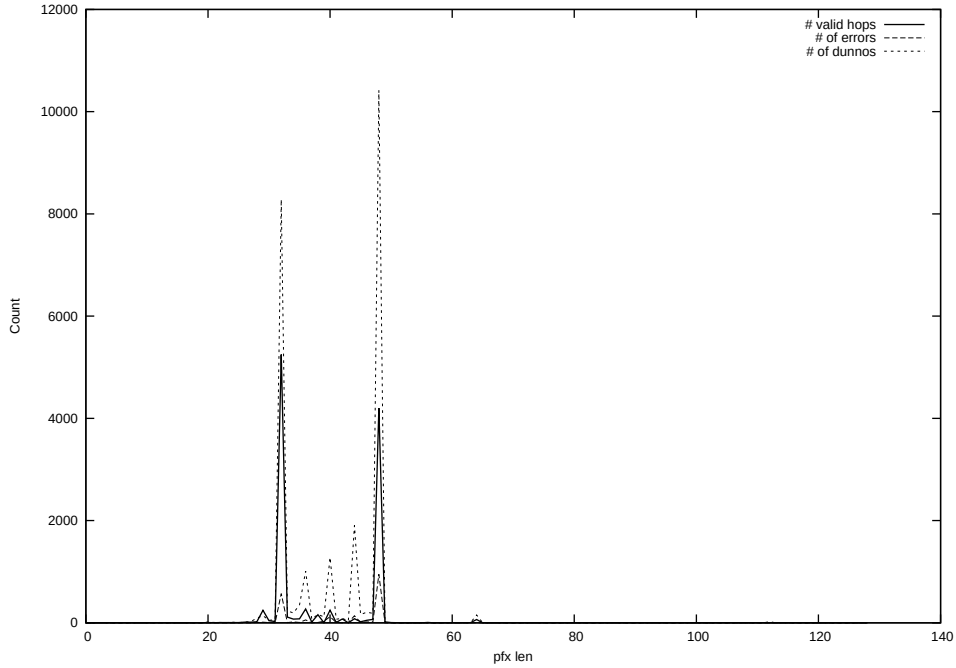
2013-04-20



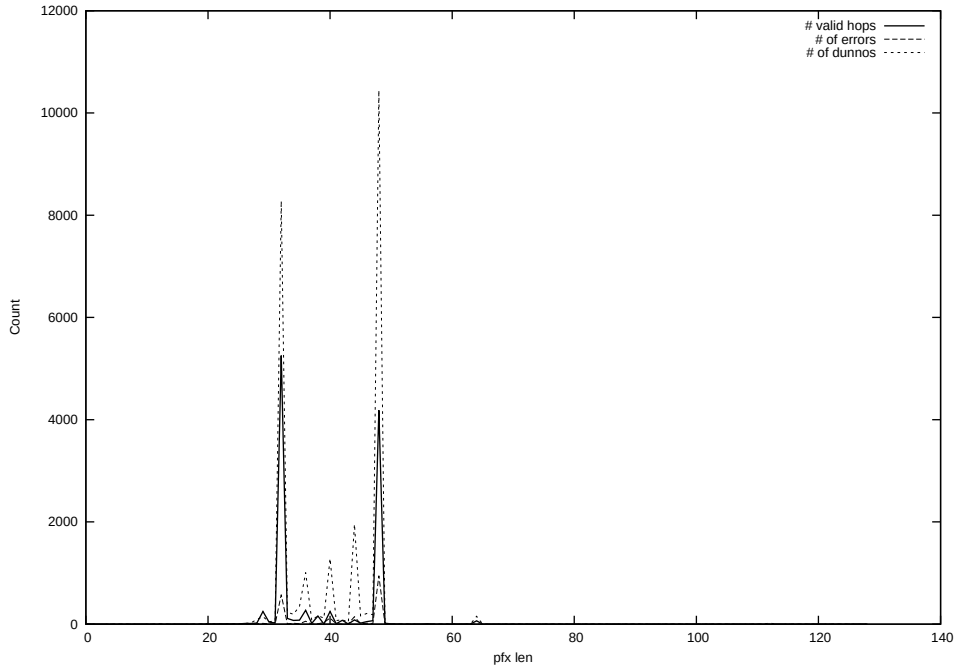
2013-04-21



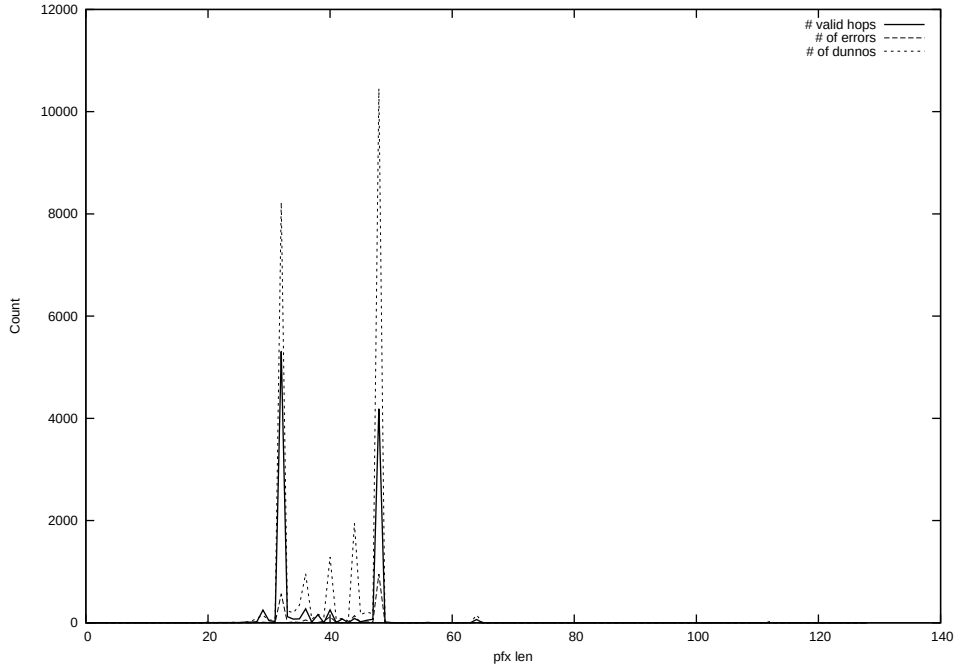
2013-04-22



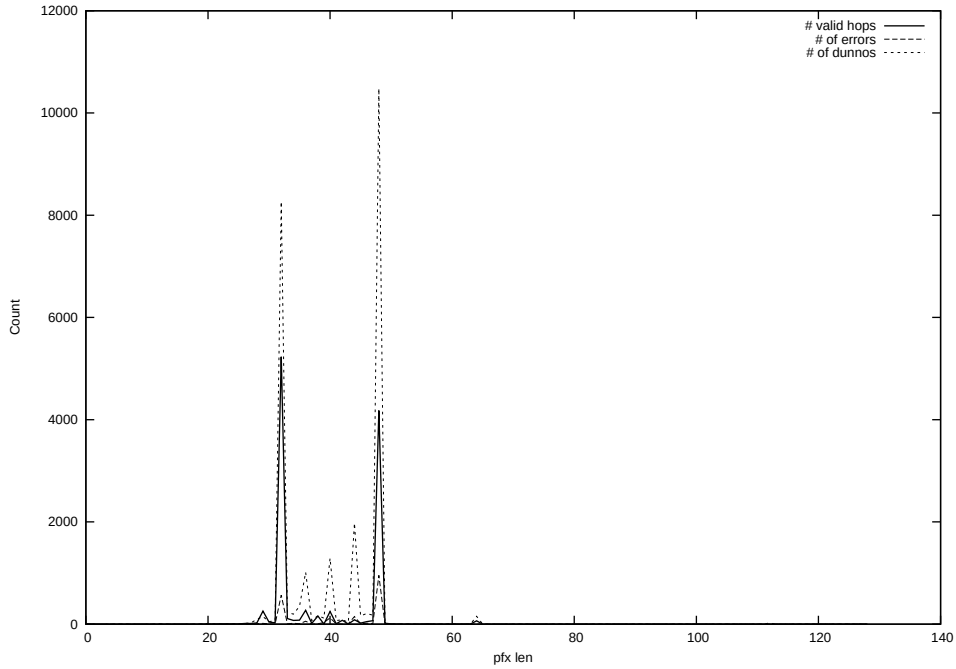
2013-04-23



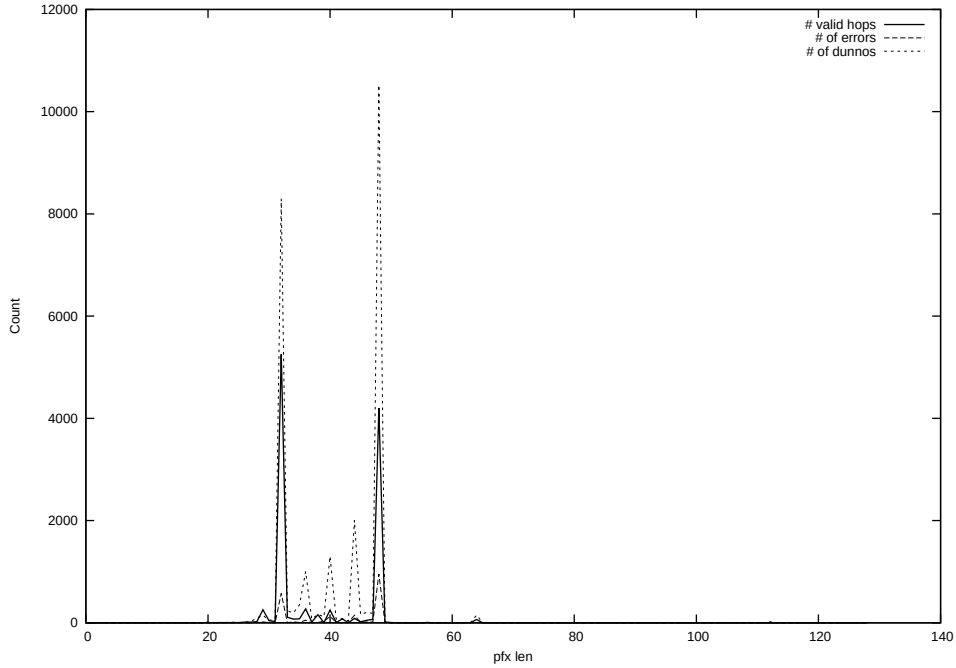
2013-04-24



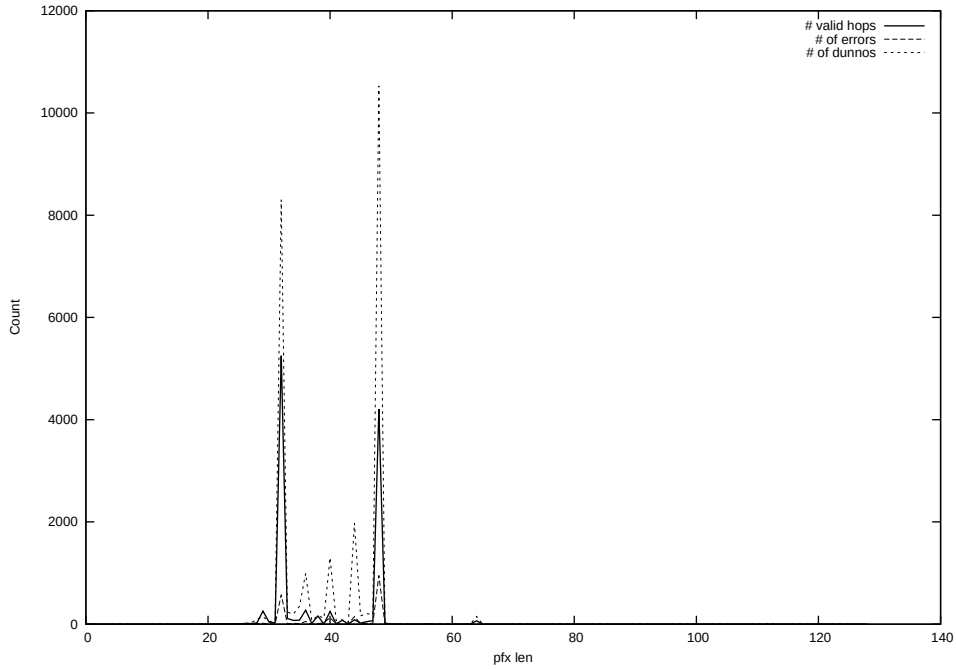
2013-04-25



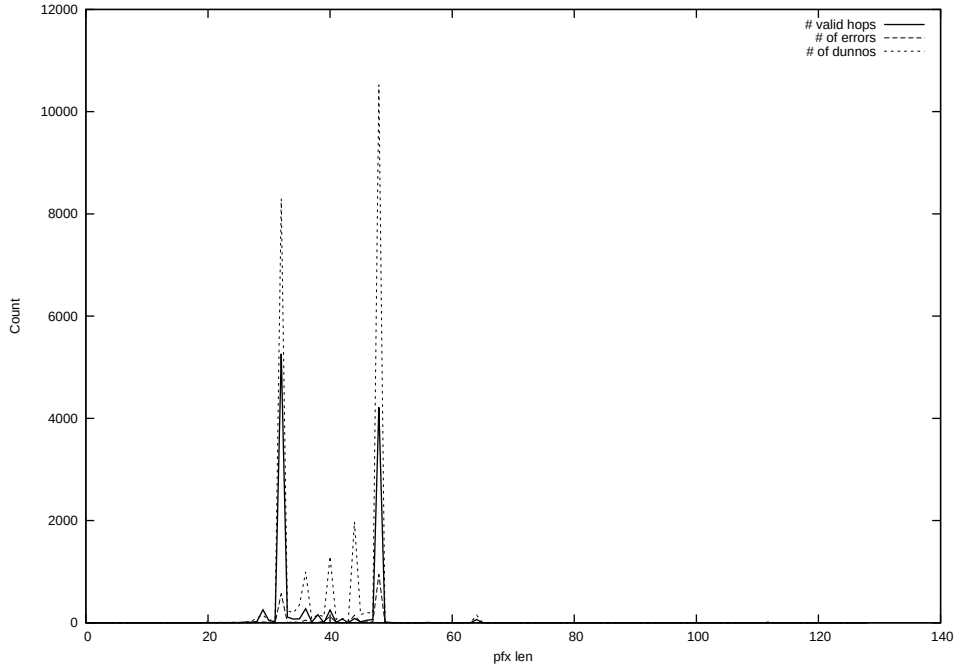
2013-04-26



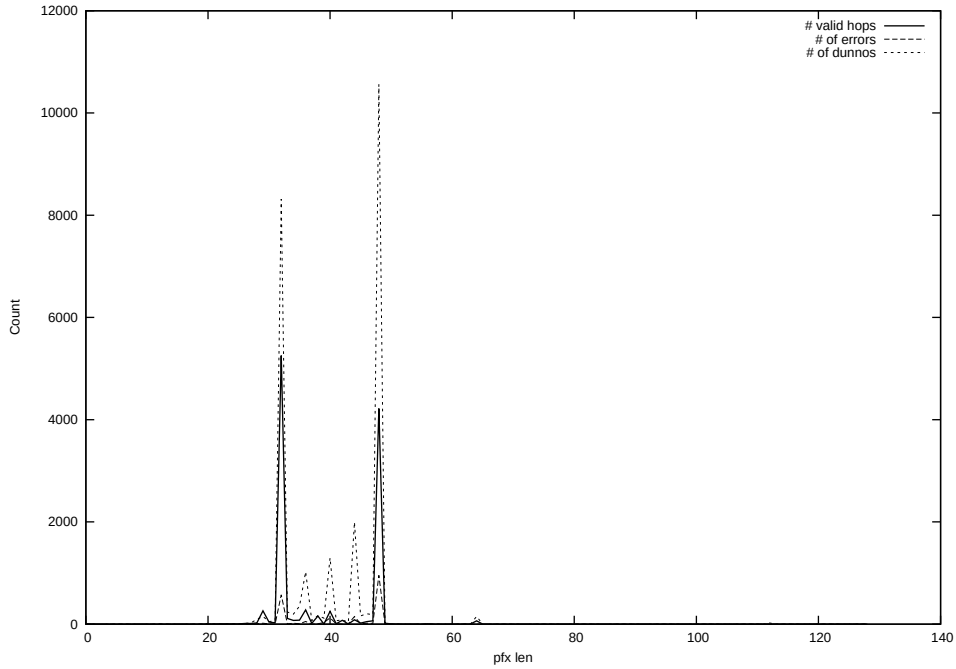
2013-04-27



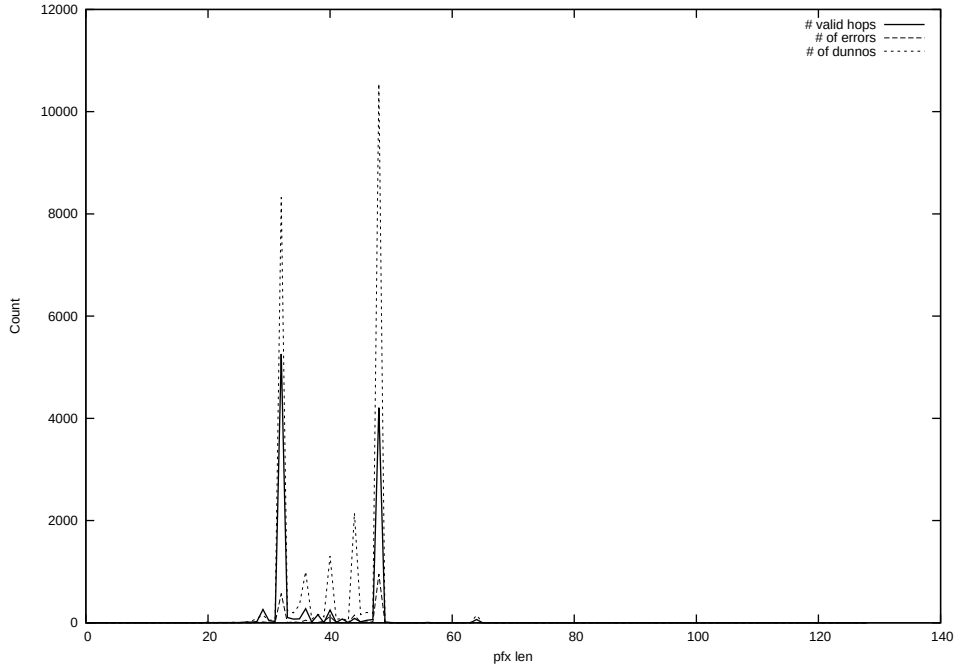
2013-04-28



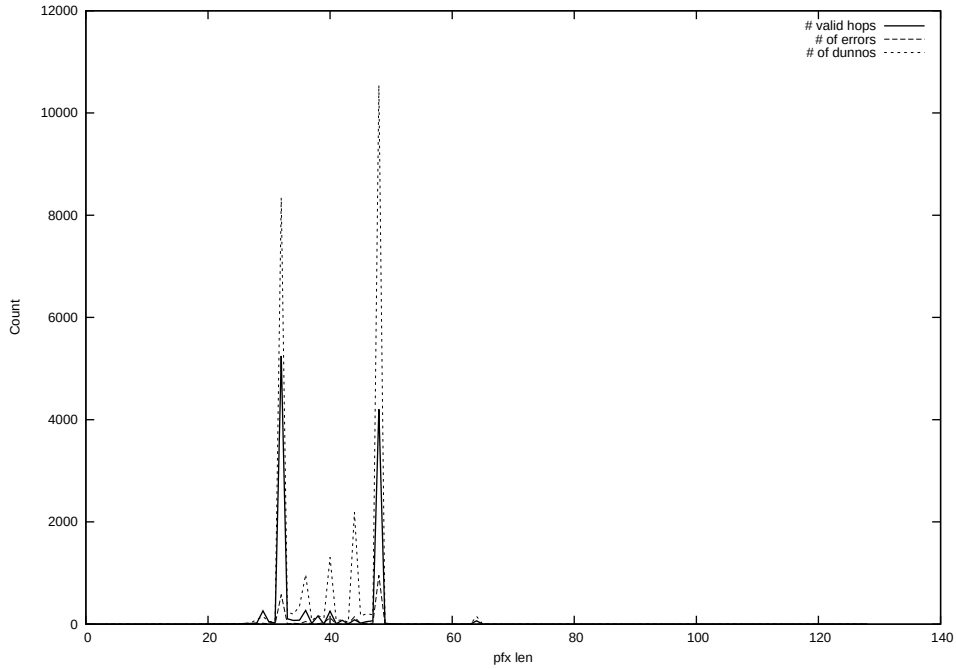
2013-04-29



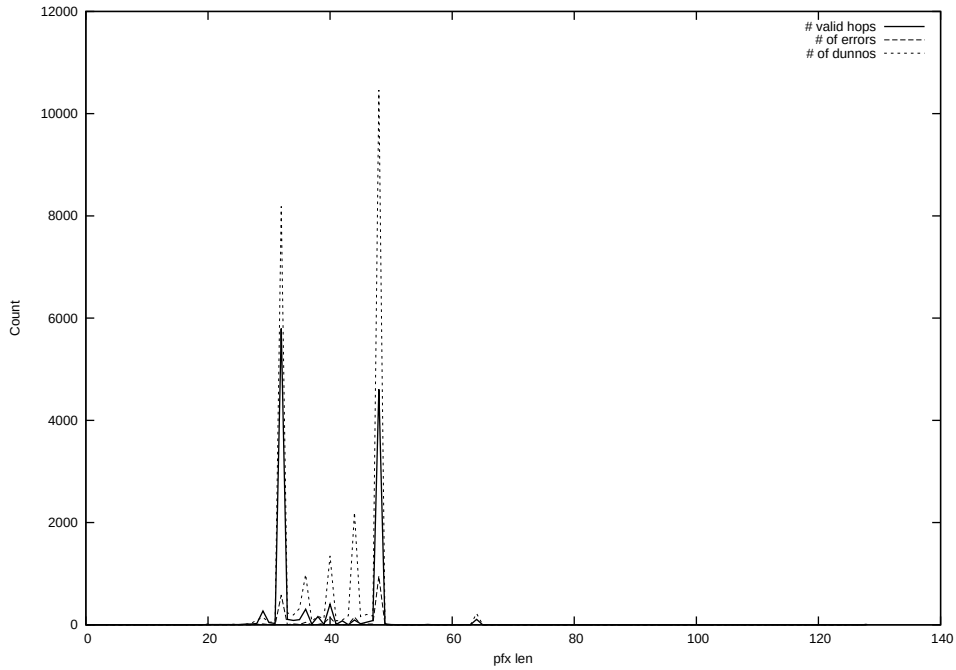
2013-04-30



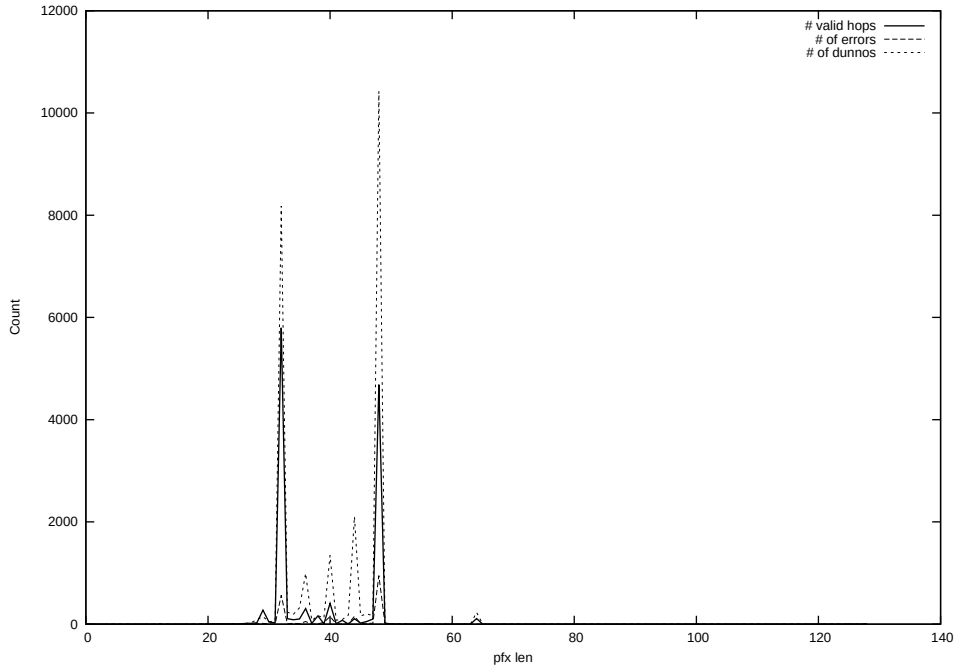
2013-05-01



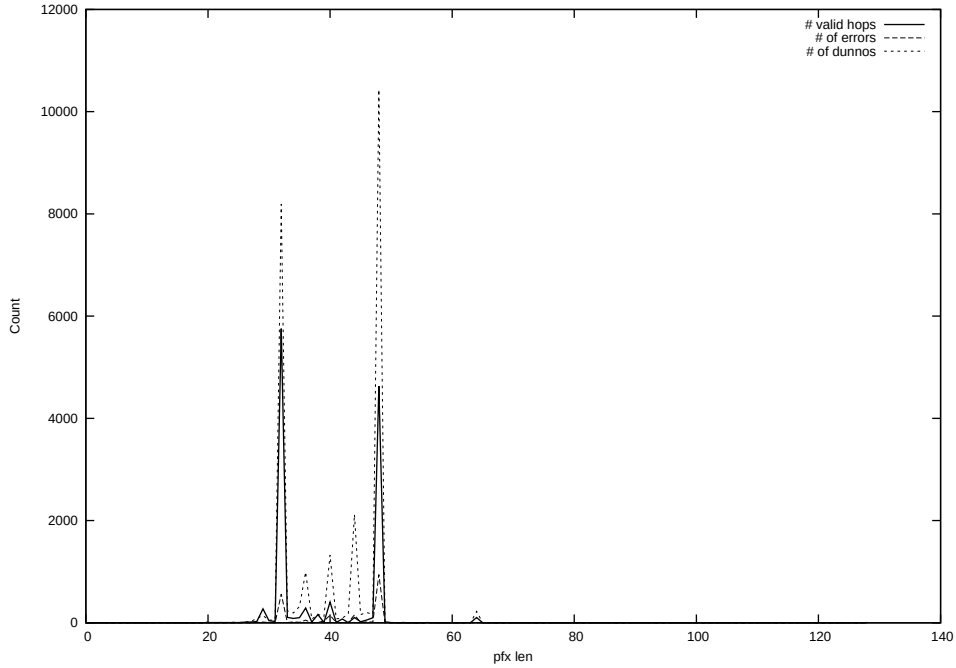
2013-05-02



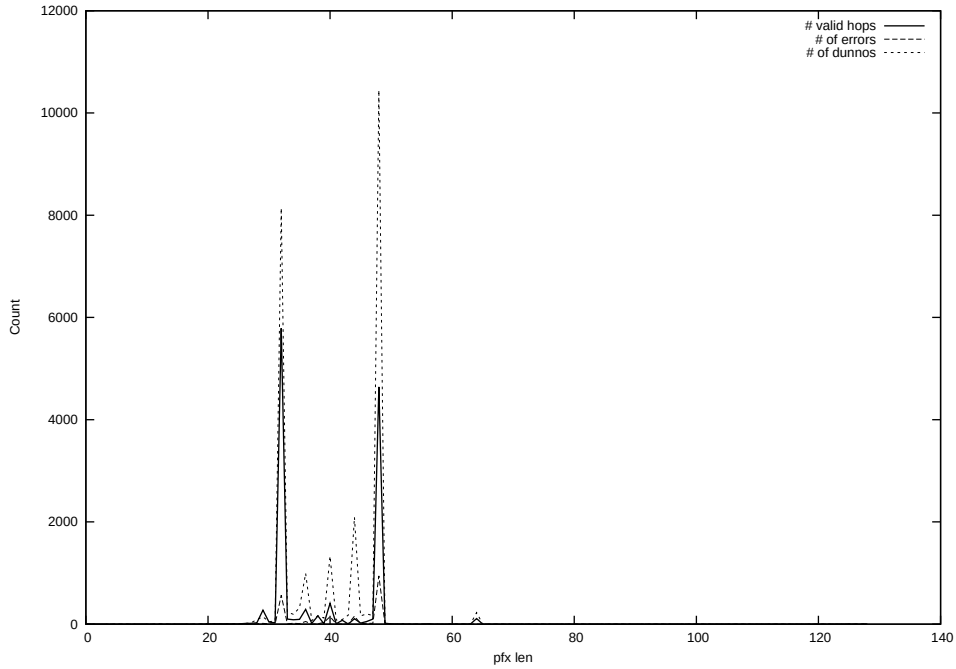
2013-05-03



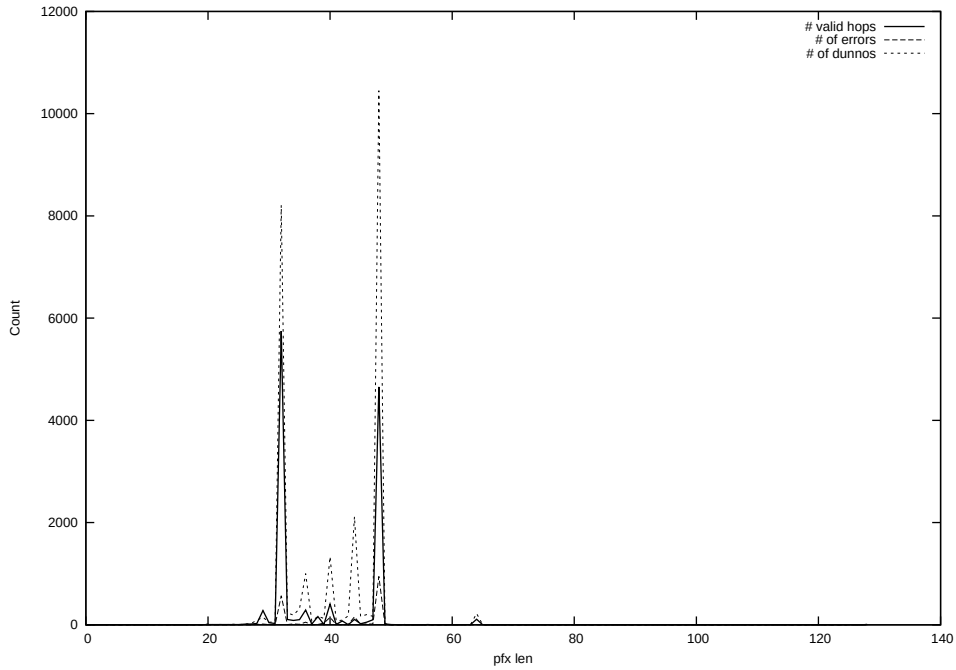
2013-05-04



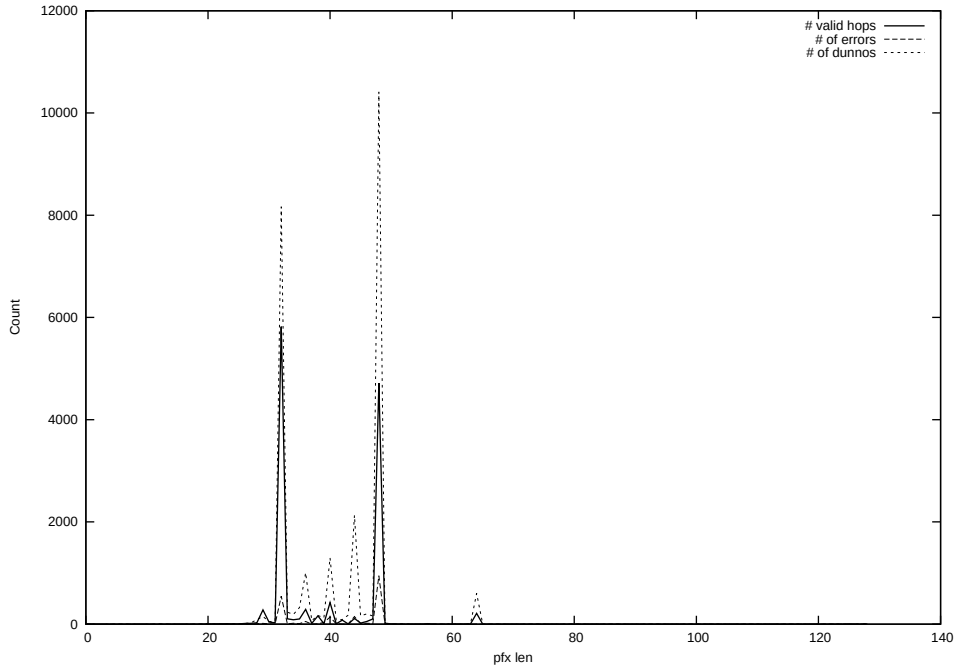
2013-05-05



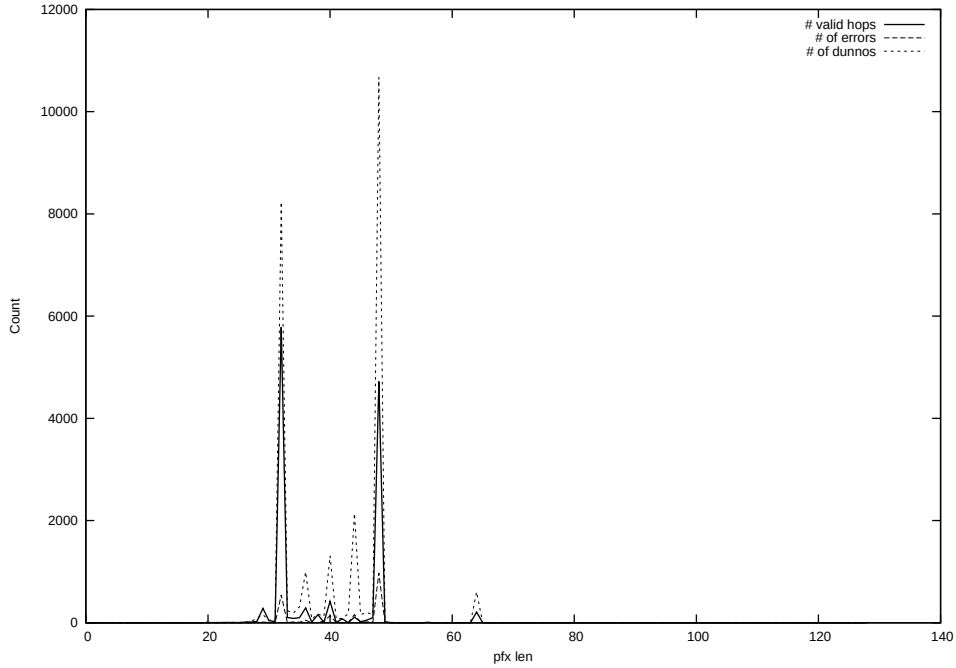
2013-05-06



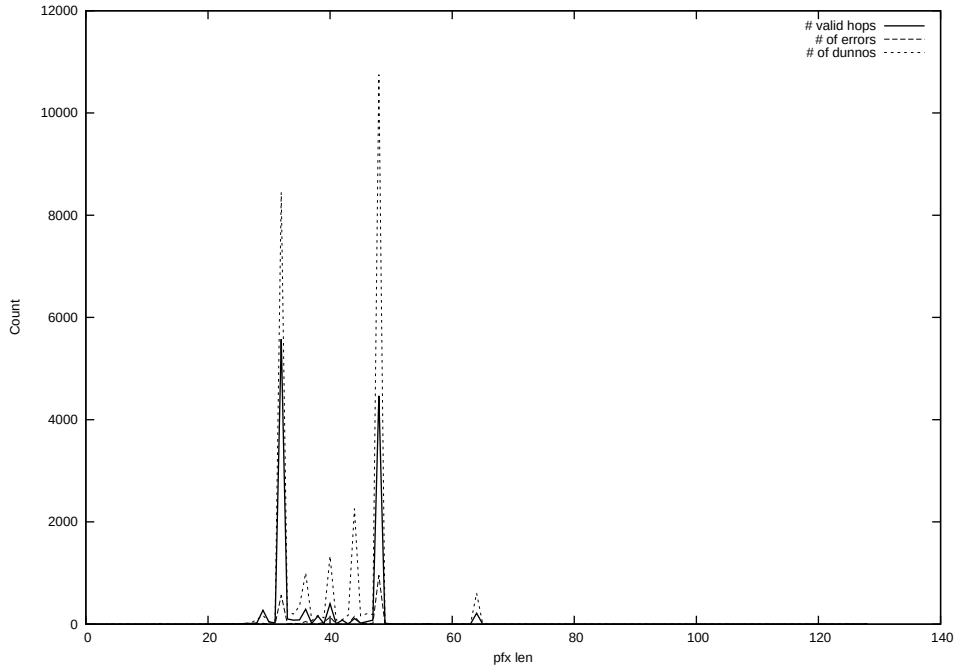
2013-05-07



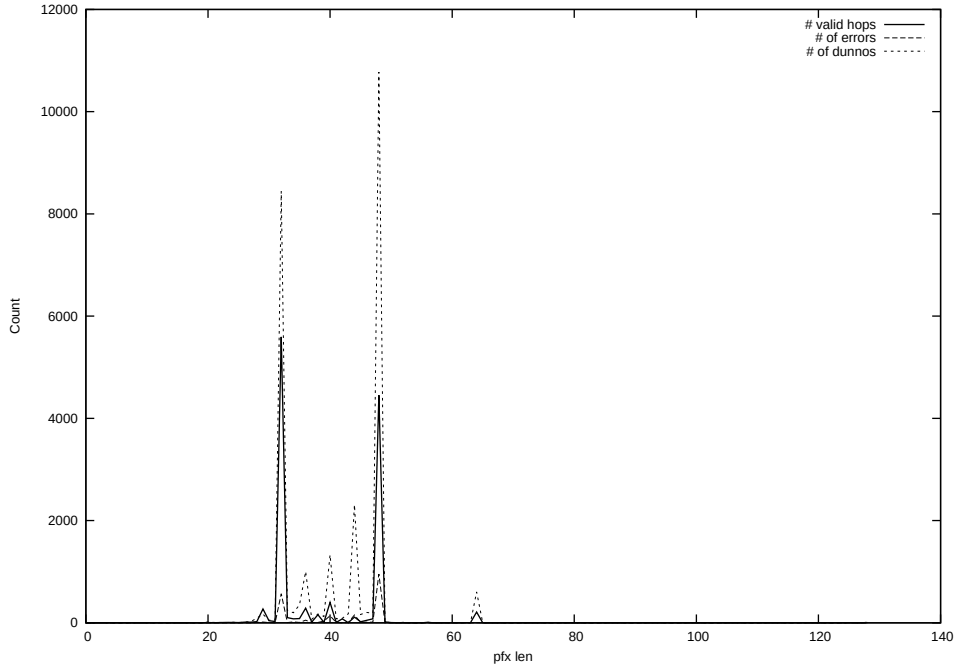
2013-05-08



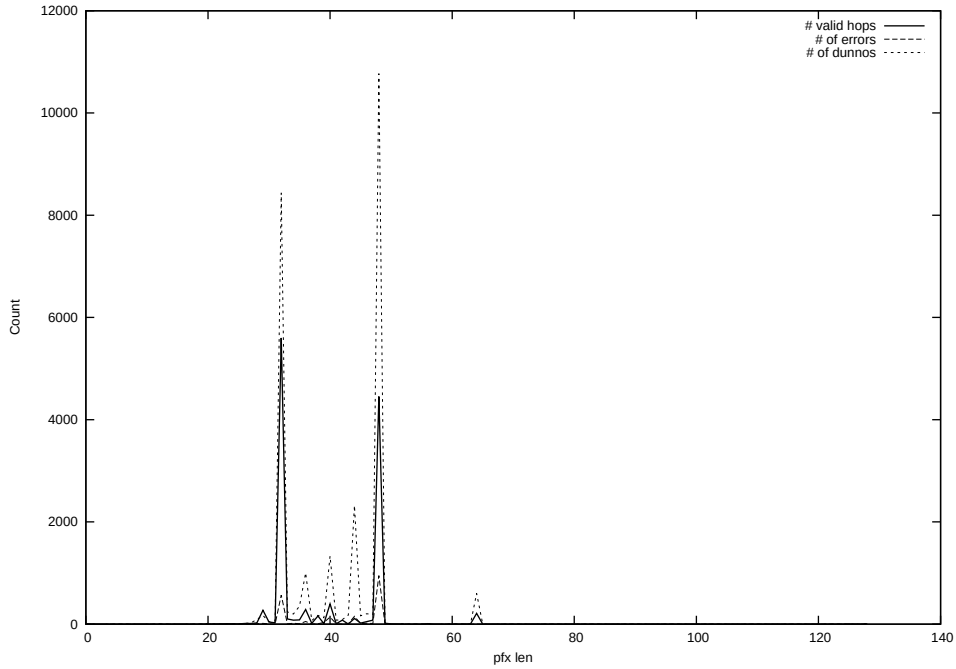
2013-05-09



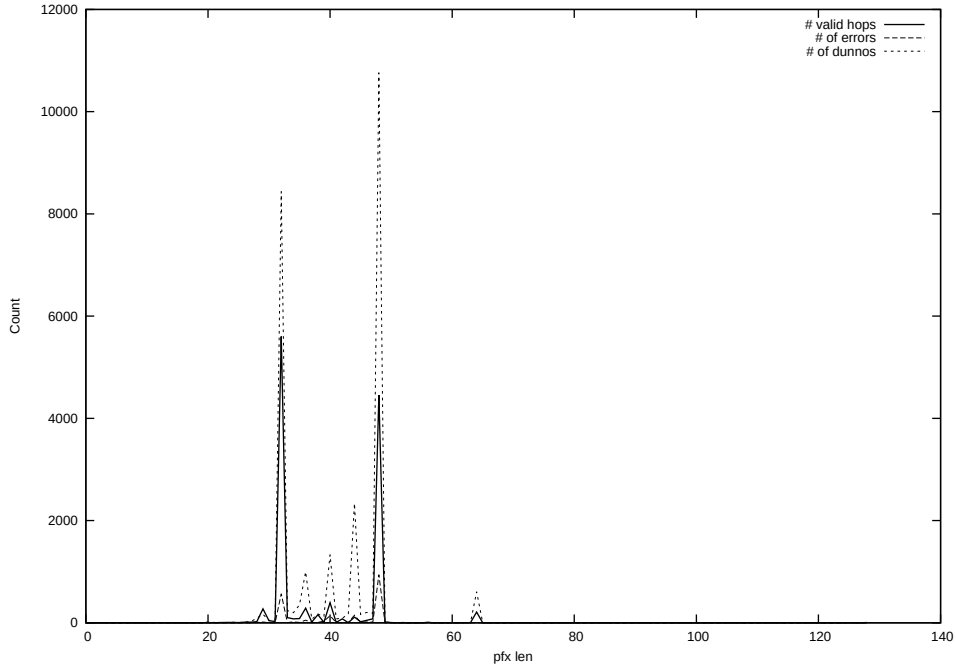
2013-05-10



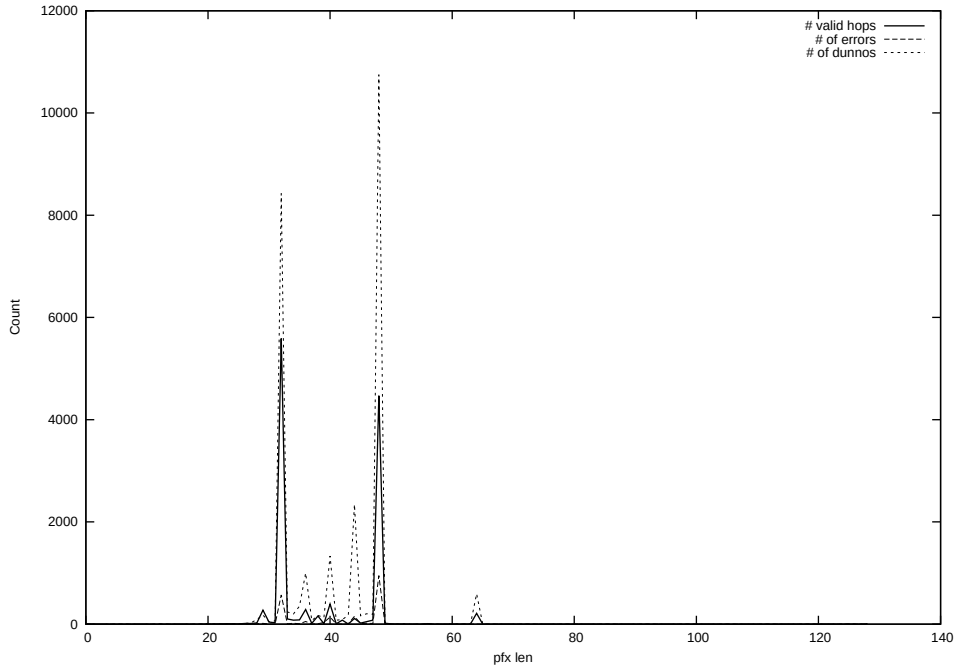
2013-05-11



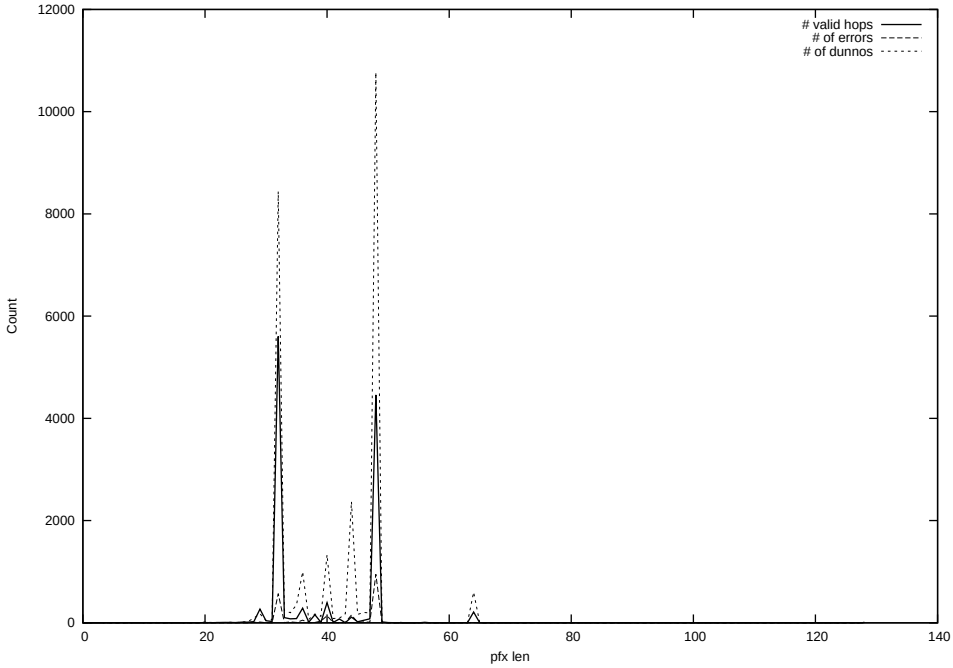
2013-05-12



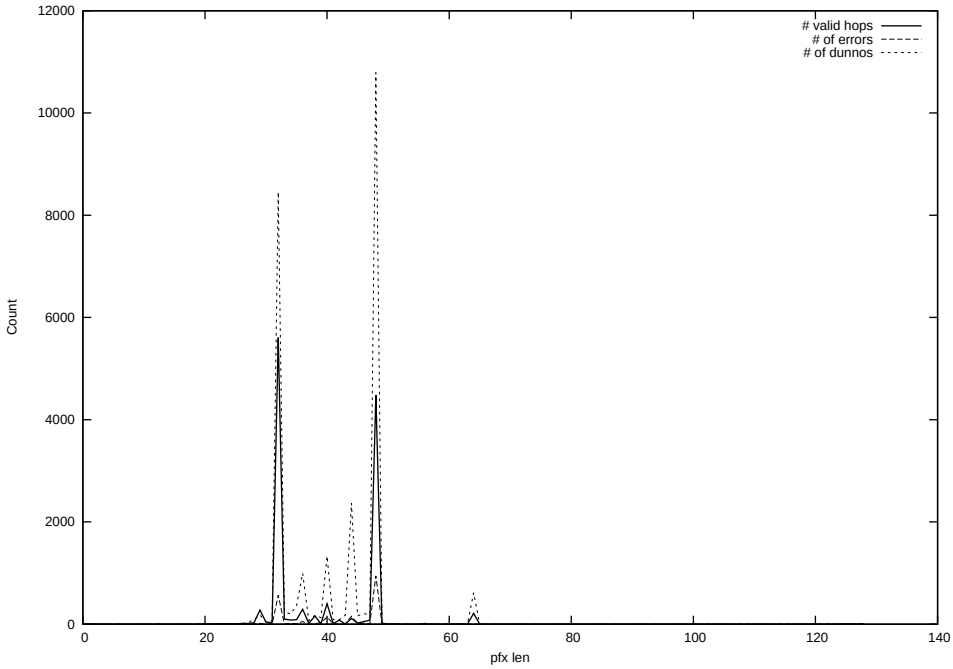
2013-05-13



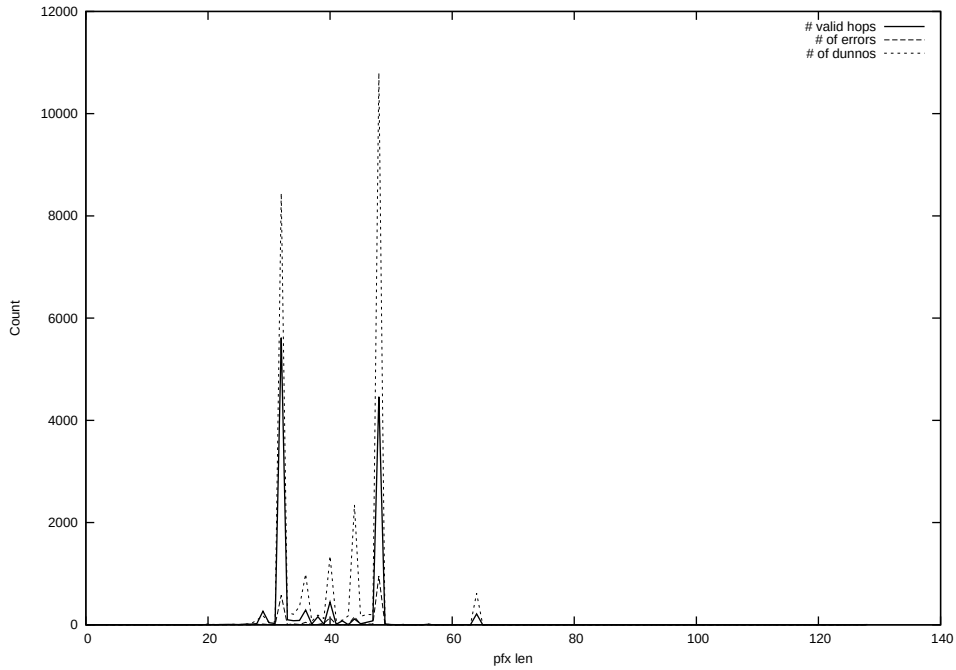
2013-05-14



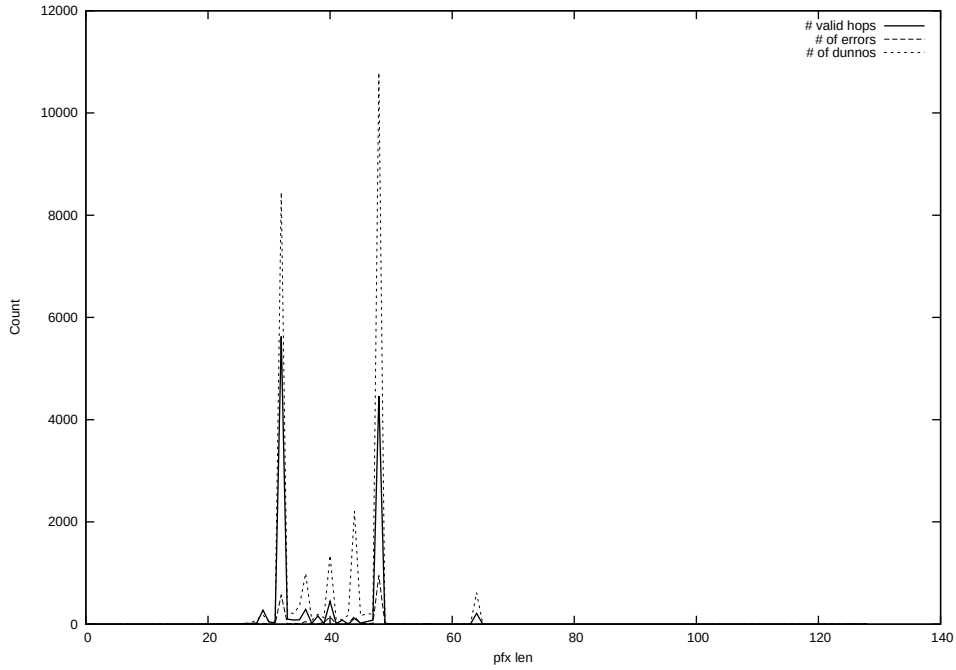
2013-05-15



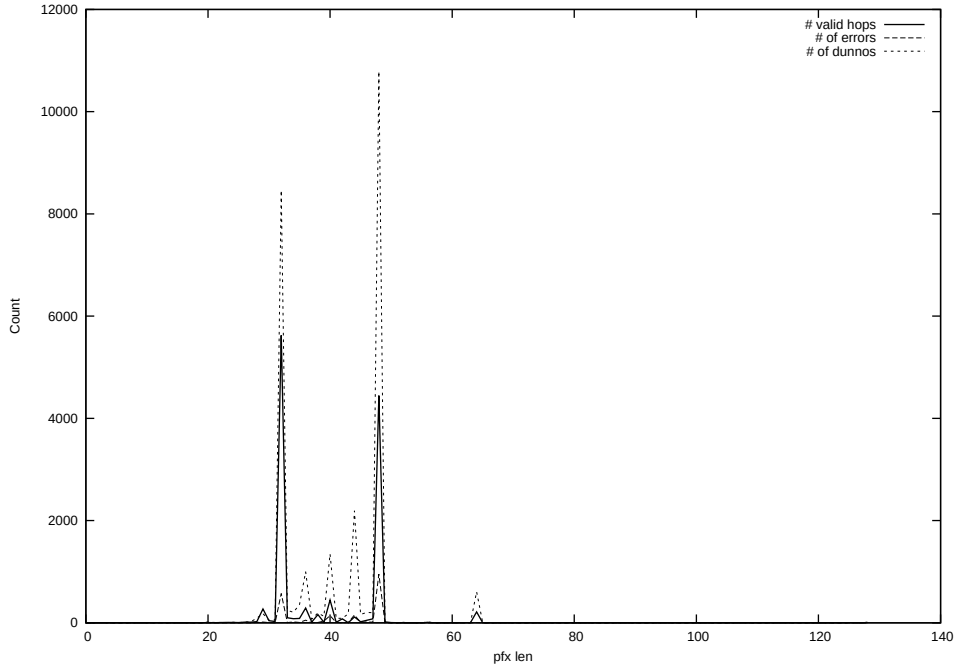
2013-05-16



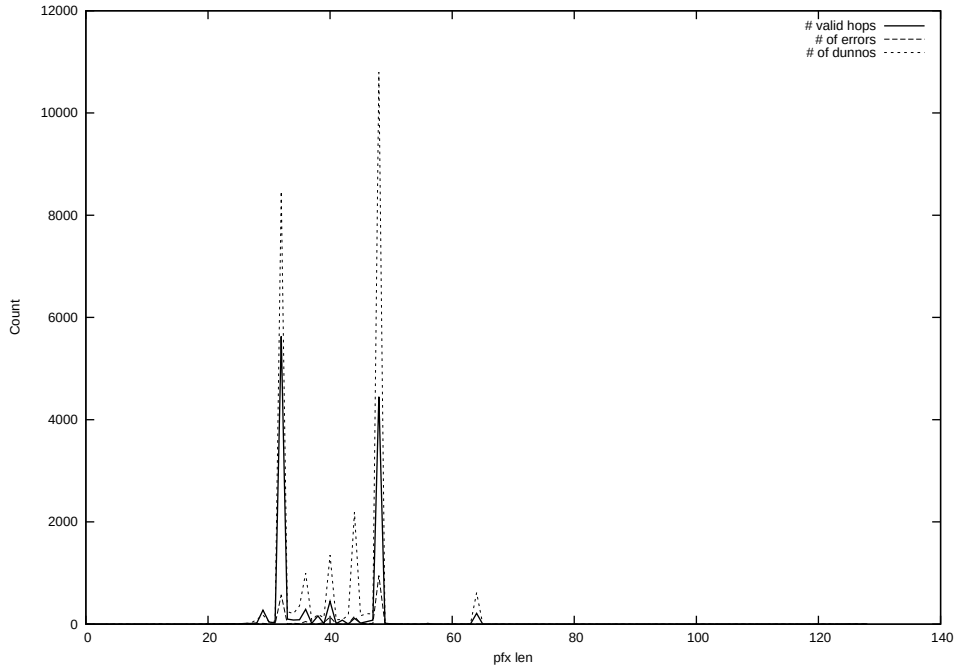
2013-05-17



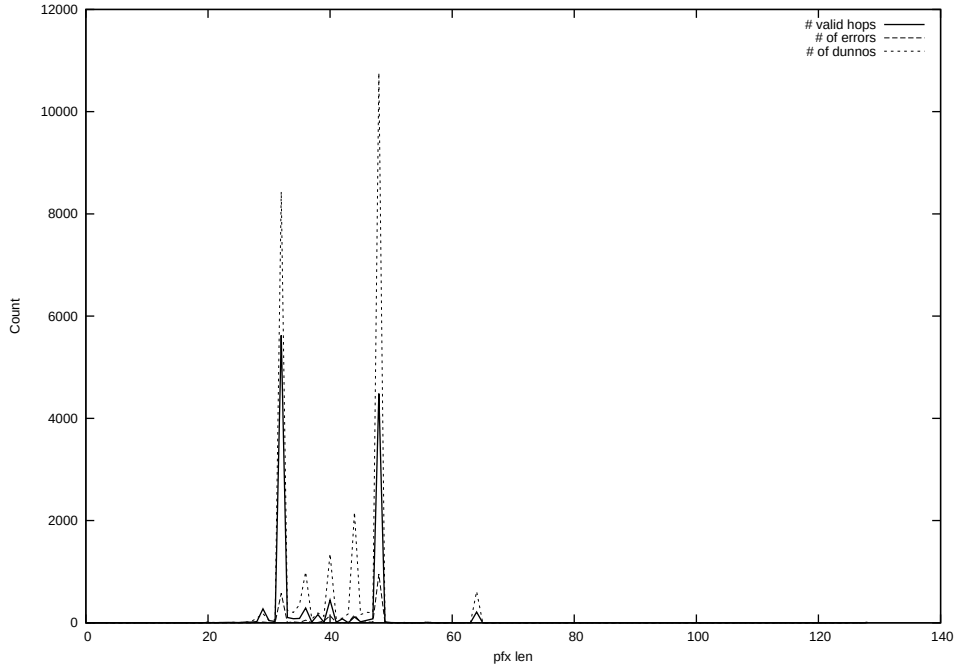
2013-05-18



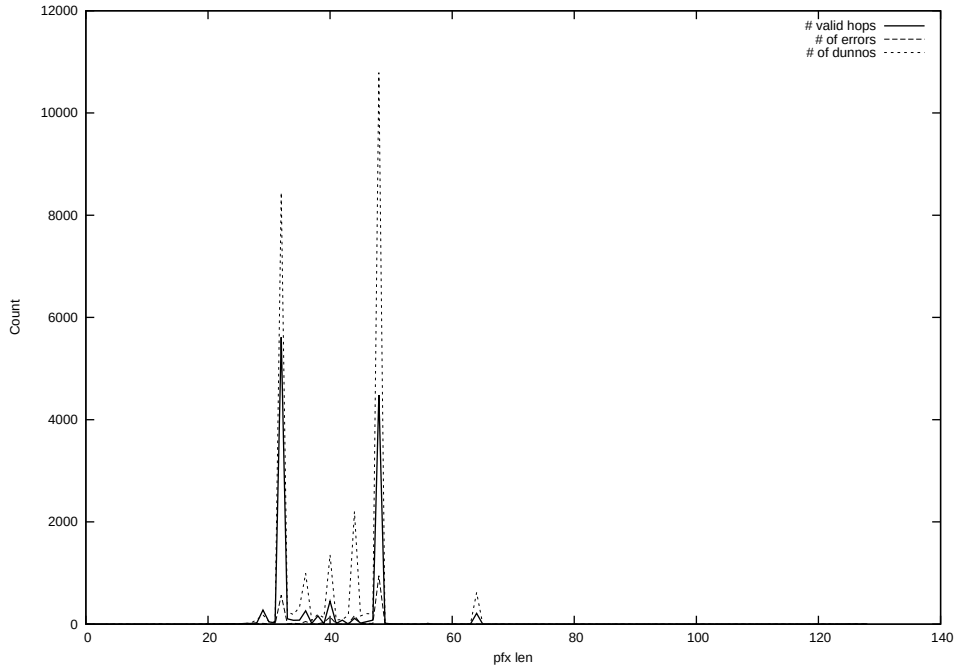
2013-05-19



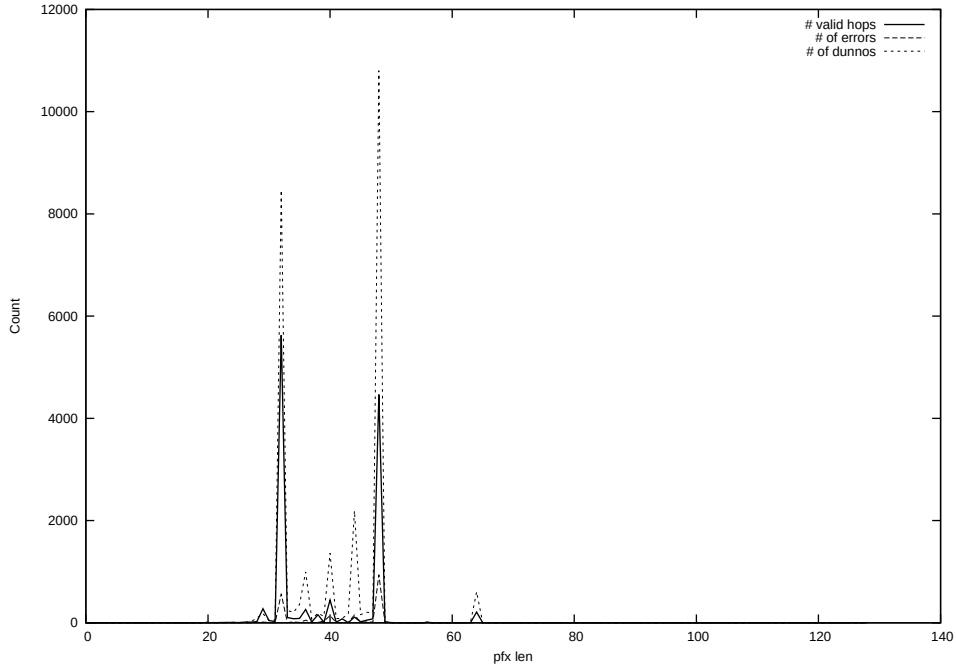
2013-05-20



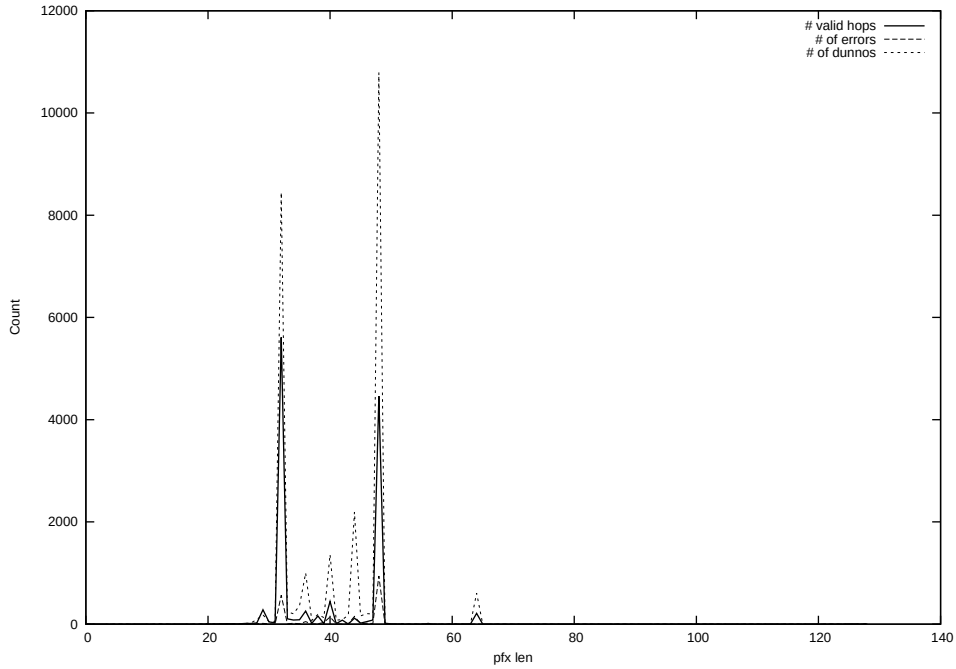
2013-05-21



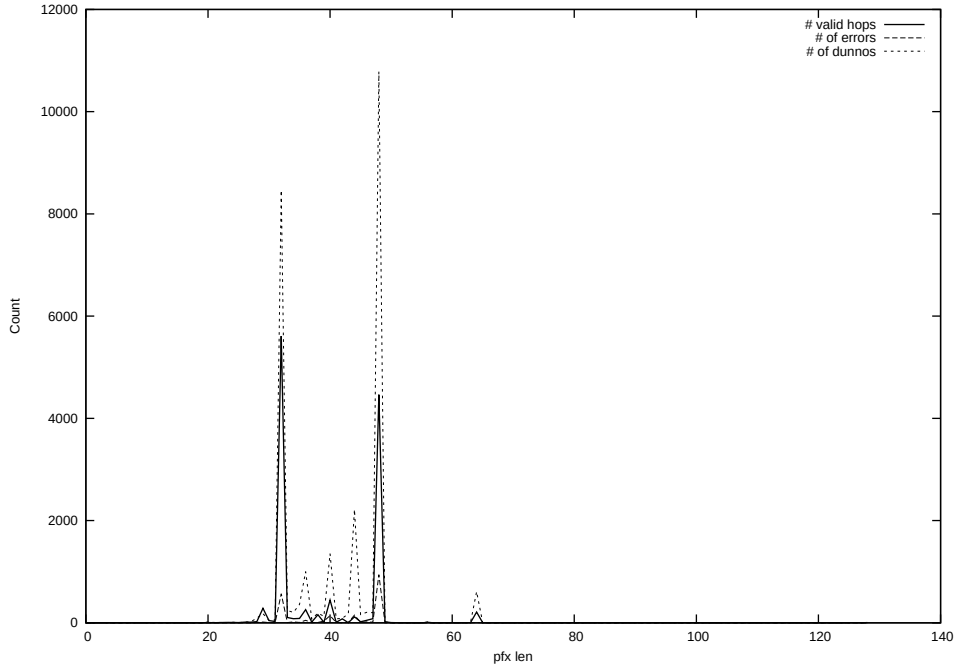
2013-05-22



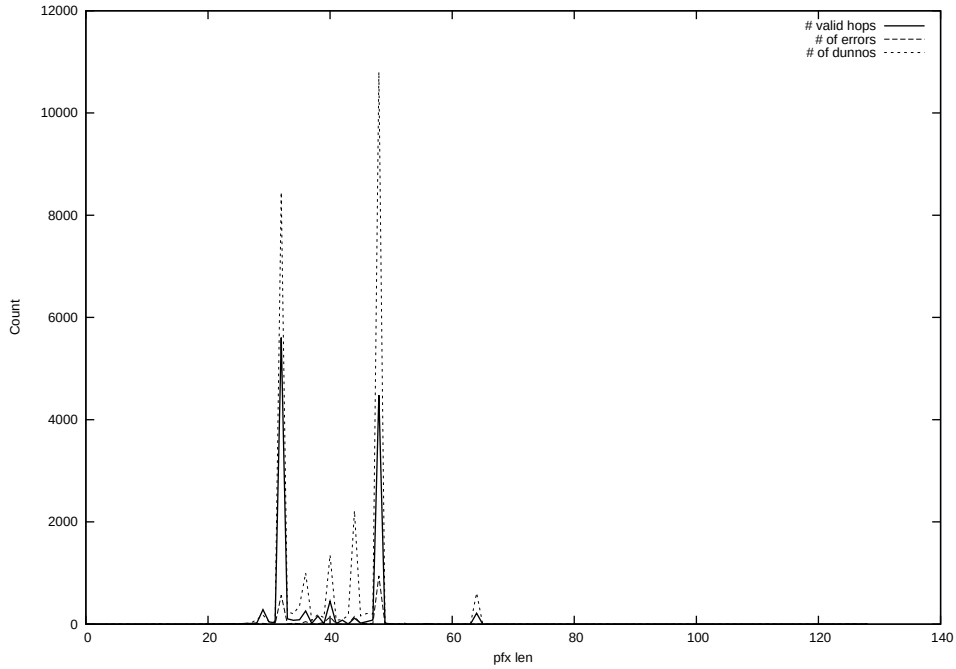
2013-05-23



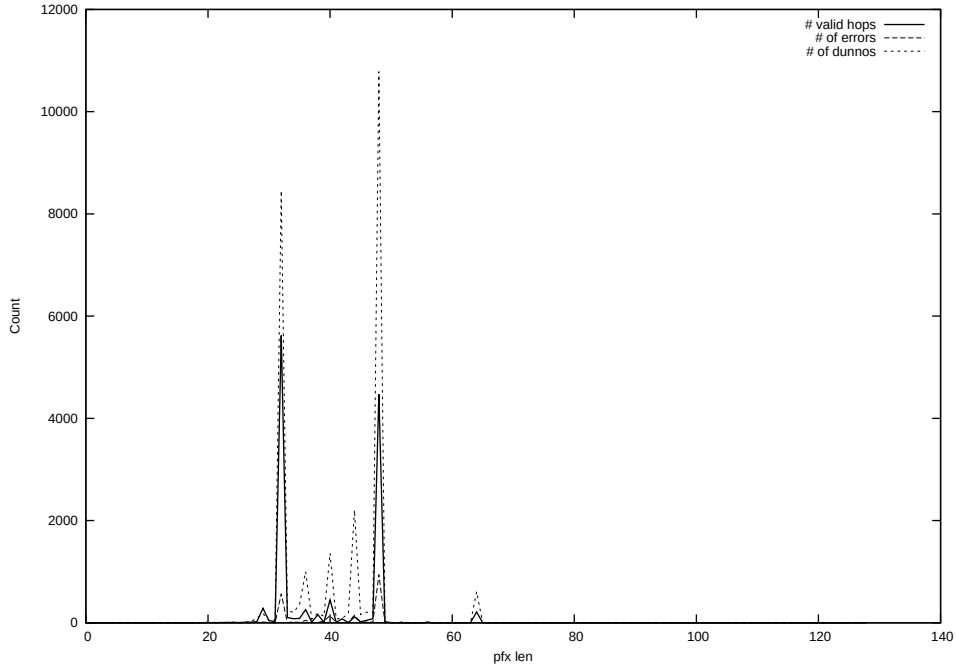
2013-05-24



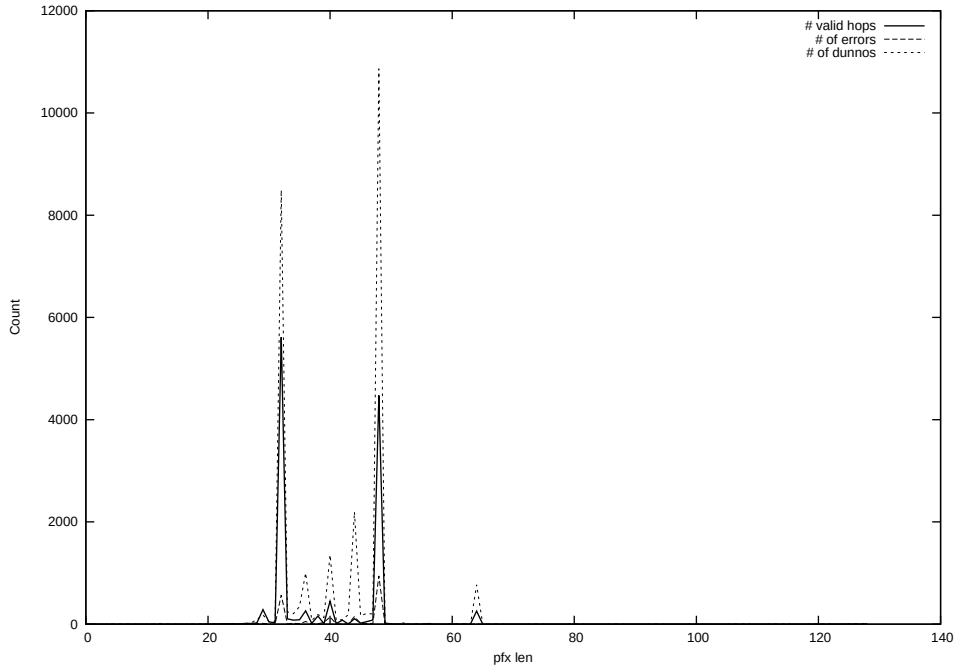
2013-05-25



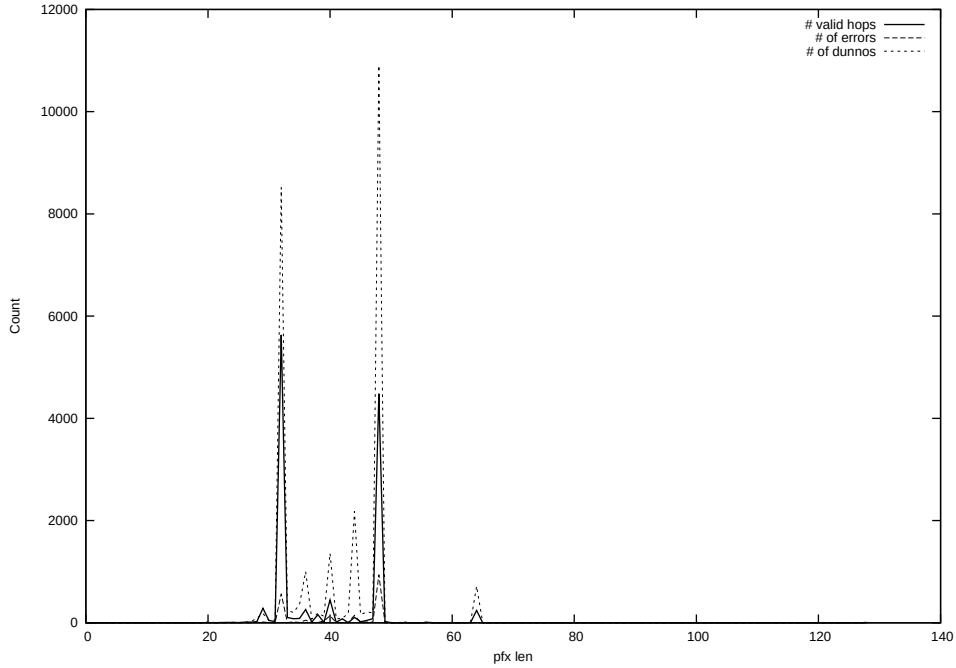
2013-05-26



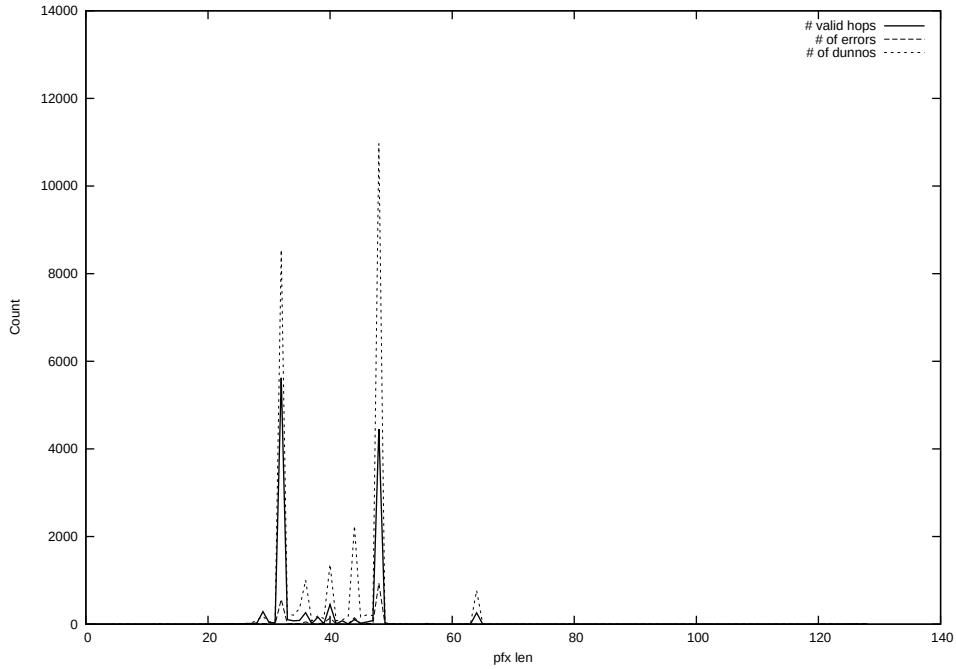
2013-05-27



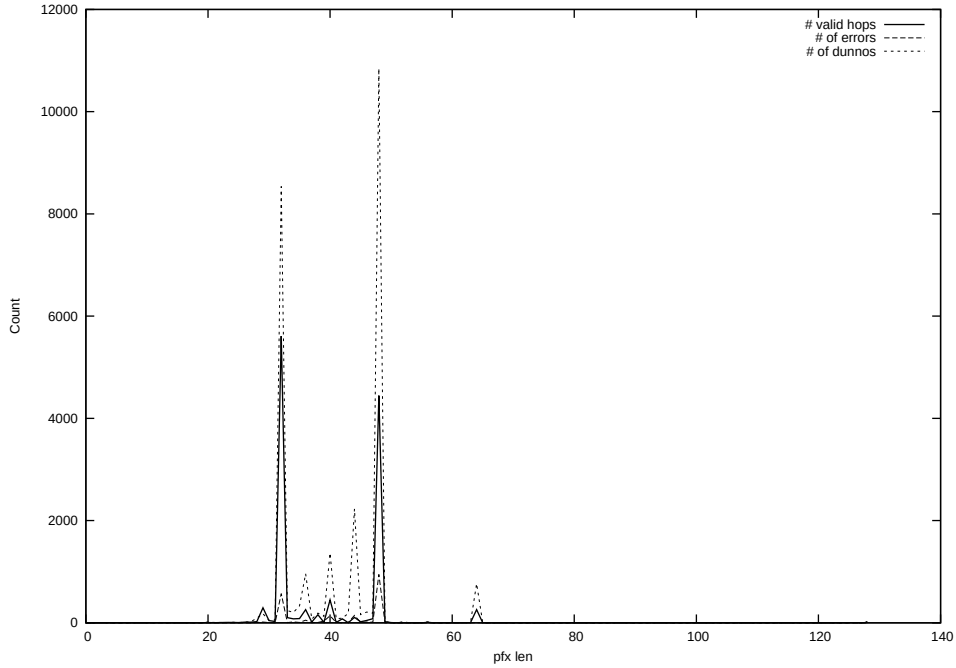
2013-05-28



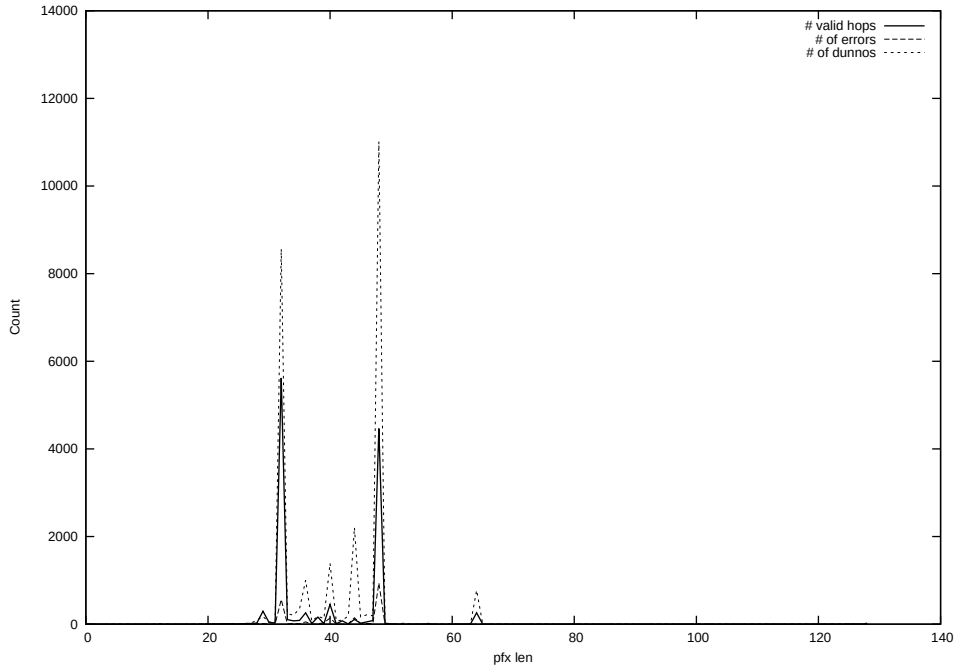
2013-05-29



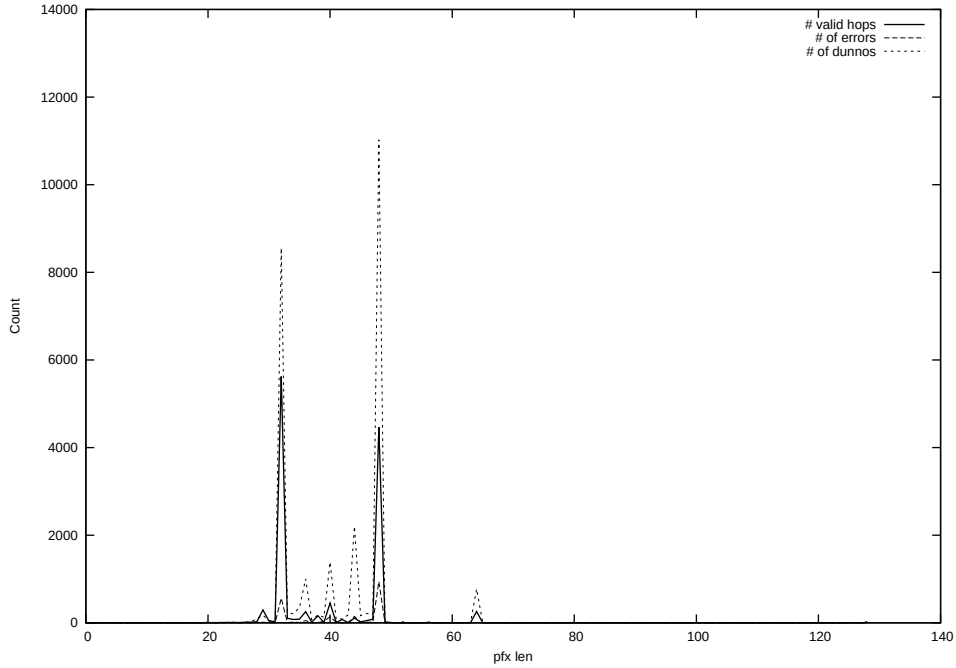
2013-05-30



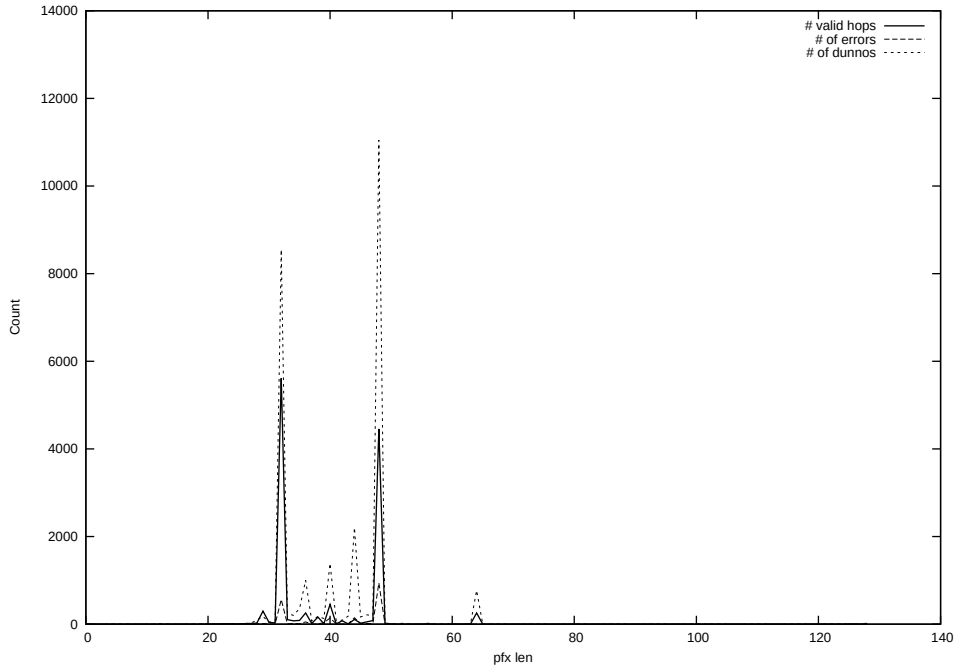
2013-05-31



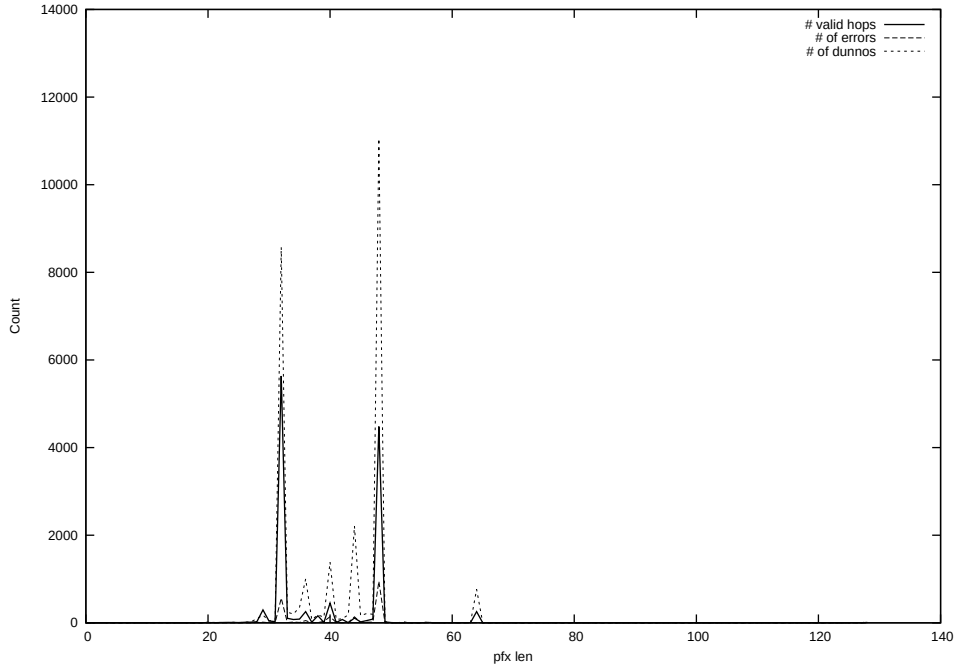
2013-06-01



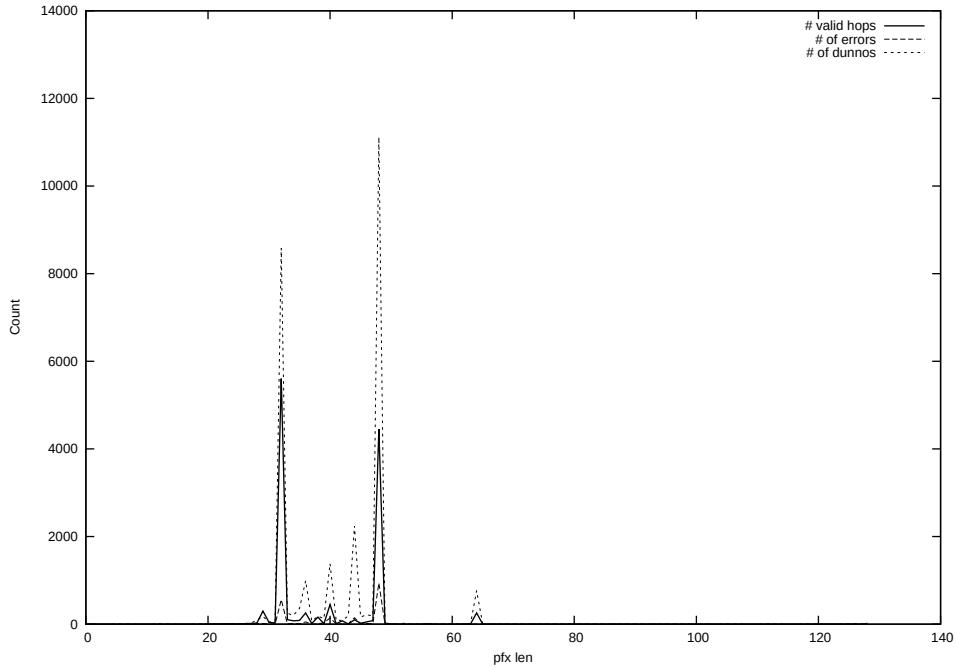
2013-06-02



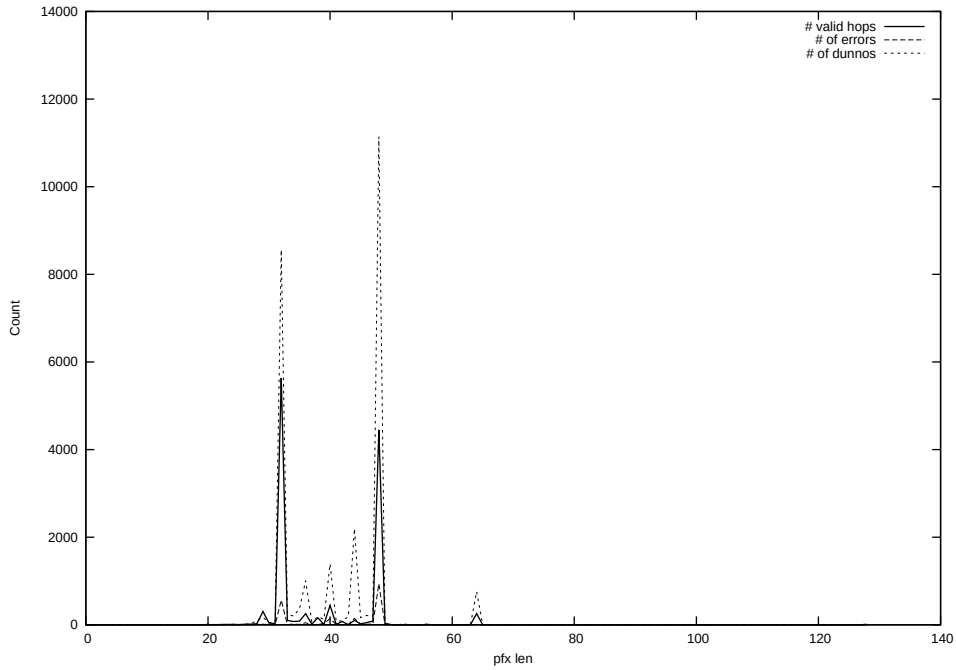
2013-06-03



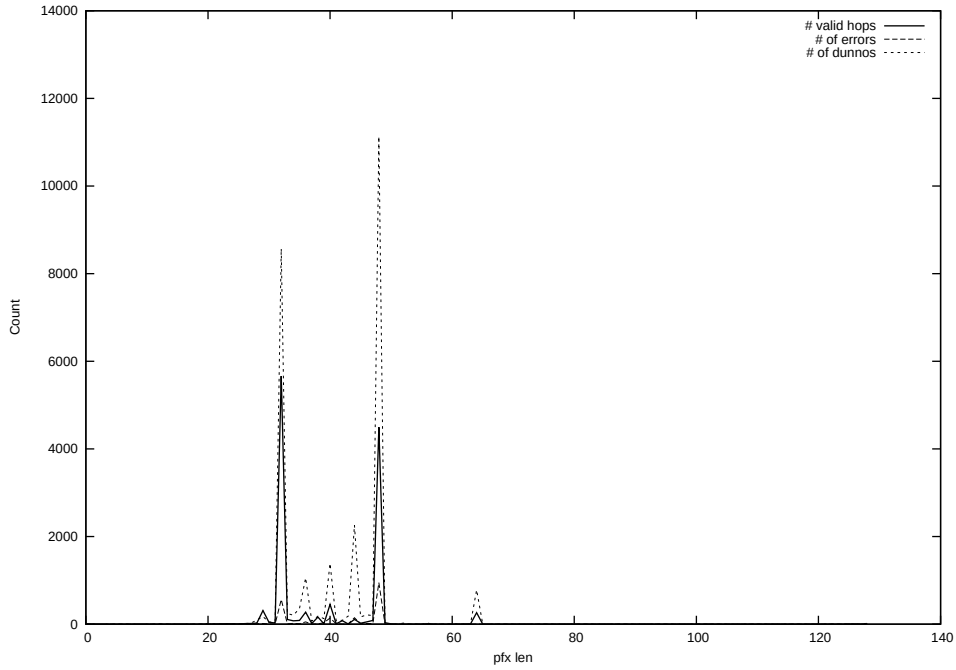
2013-06-04



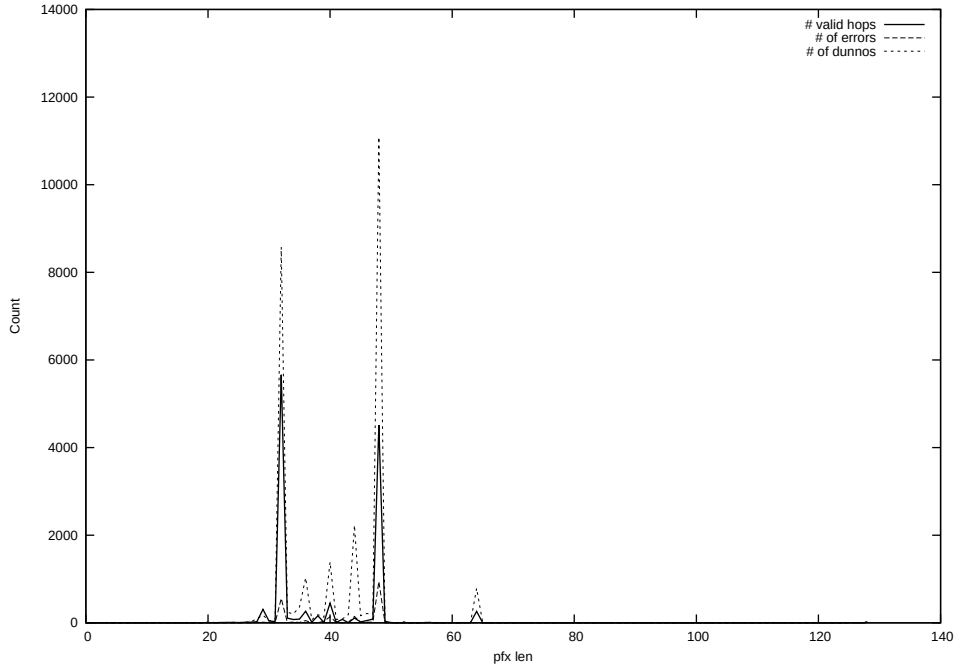
2013-06-05



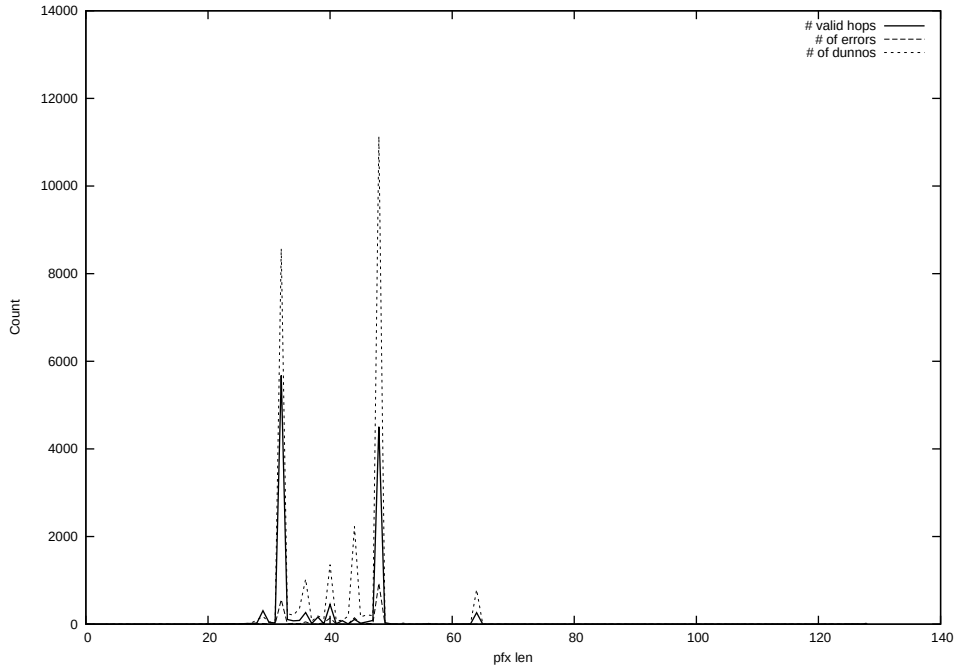
2013-06-06



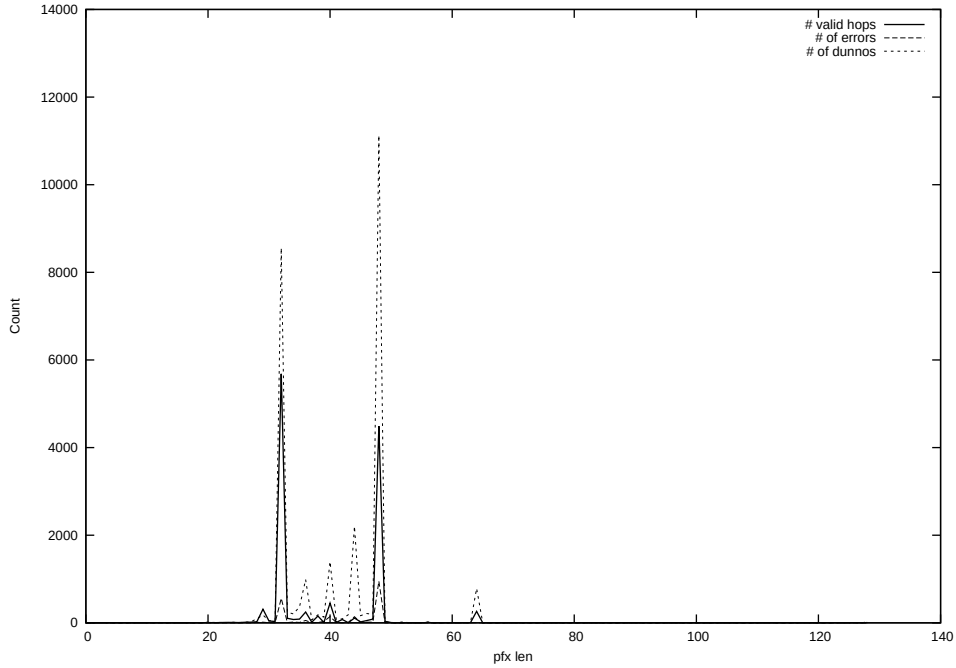
2013-06-07



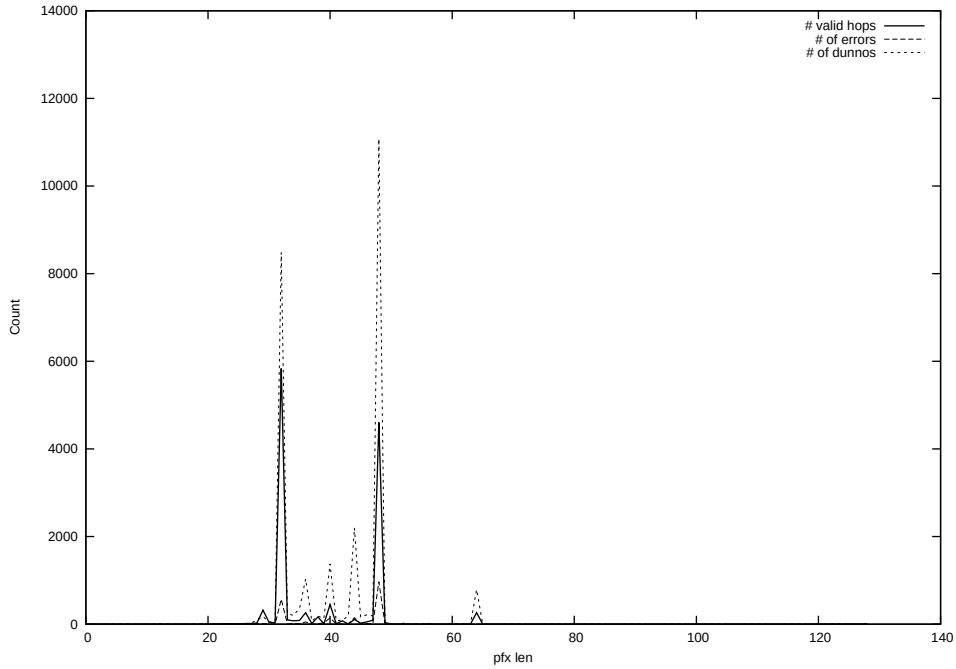
2013-06-09



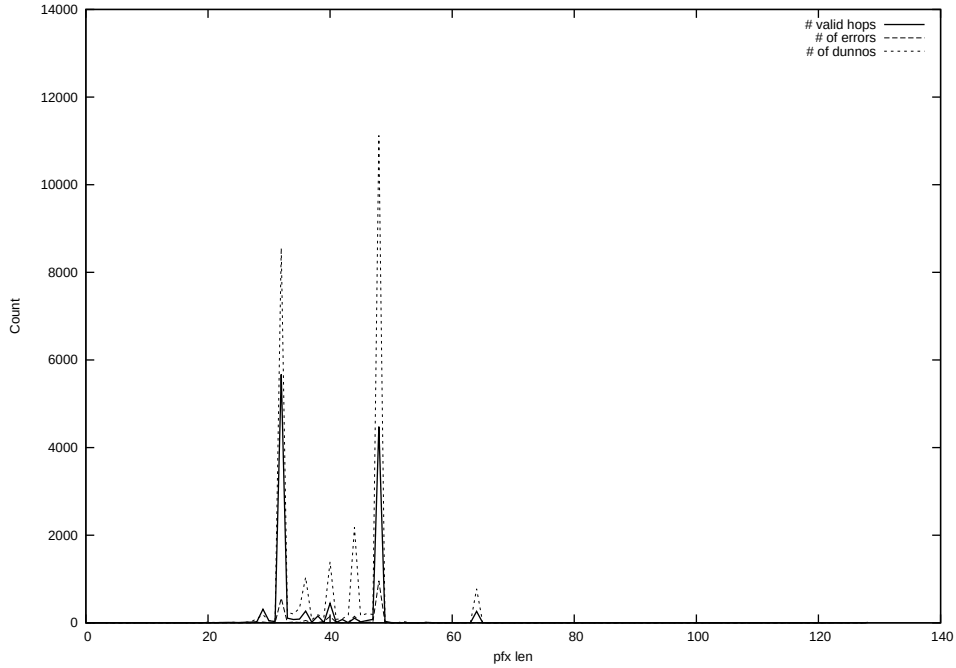
2013-06-10



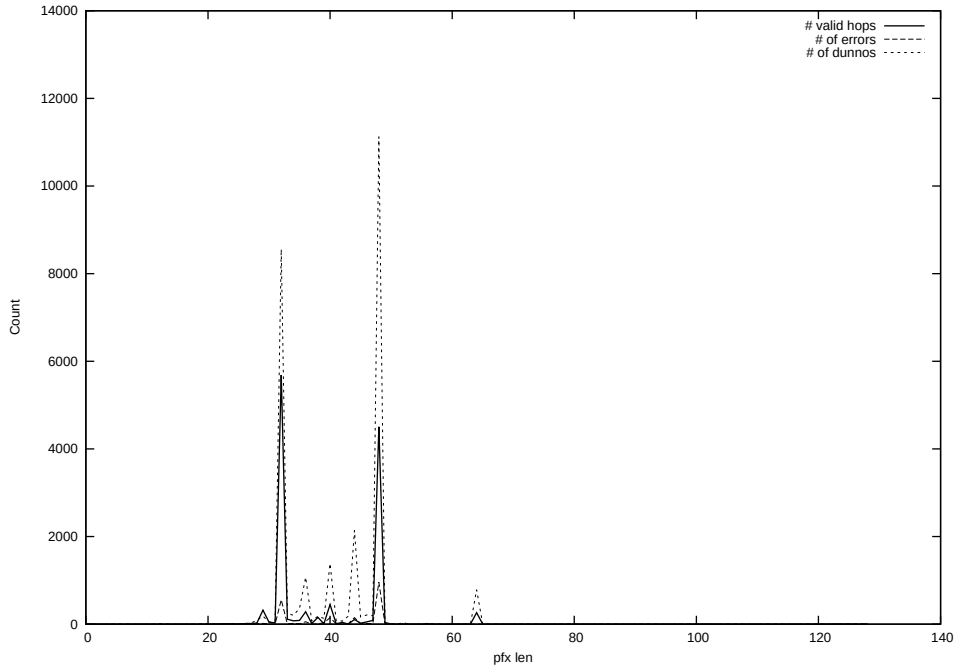
2013-06-12



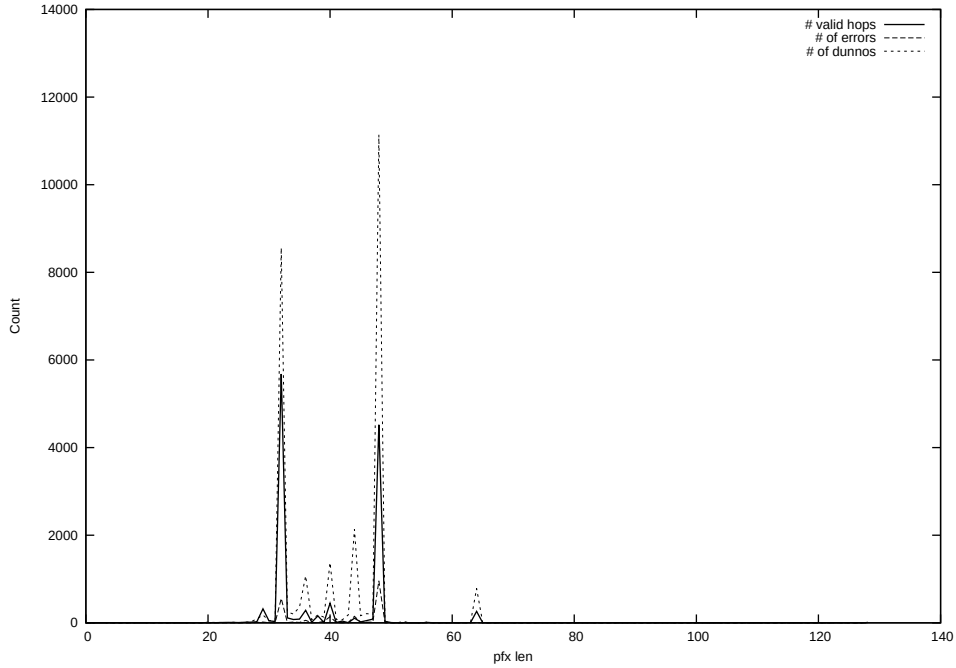
2013-06-13



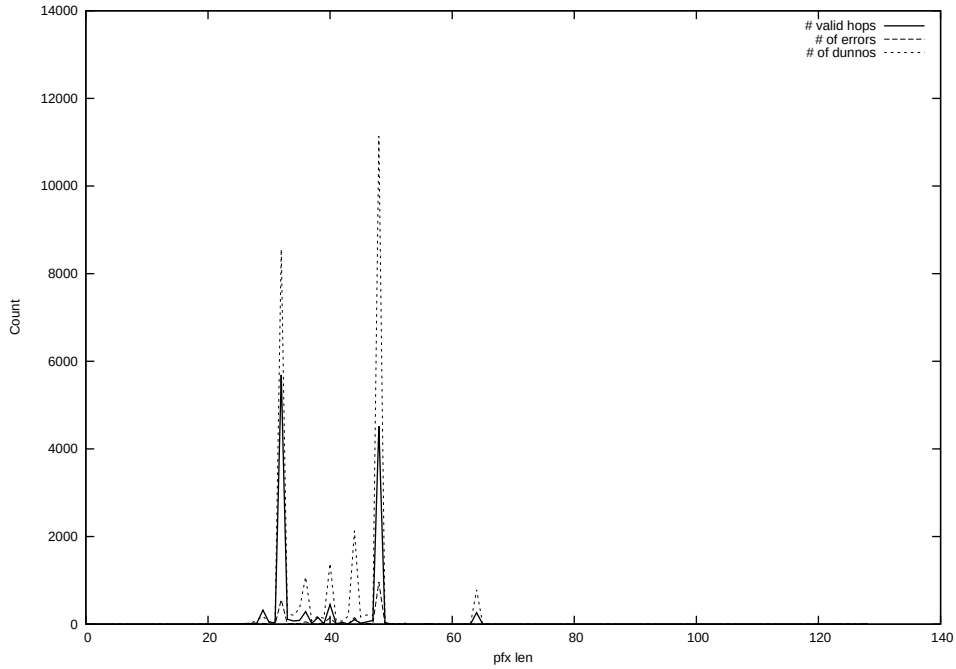
2013-06-14



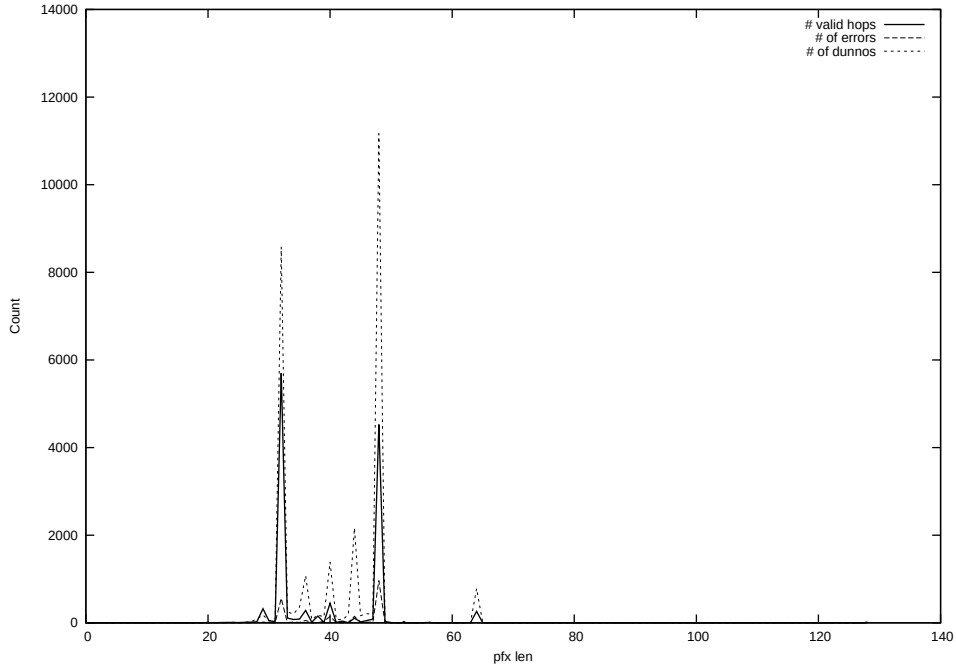
2013-06-15



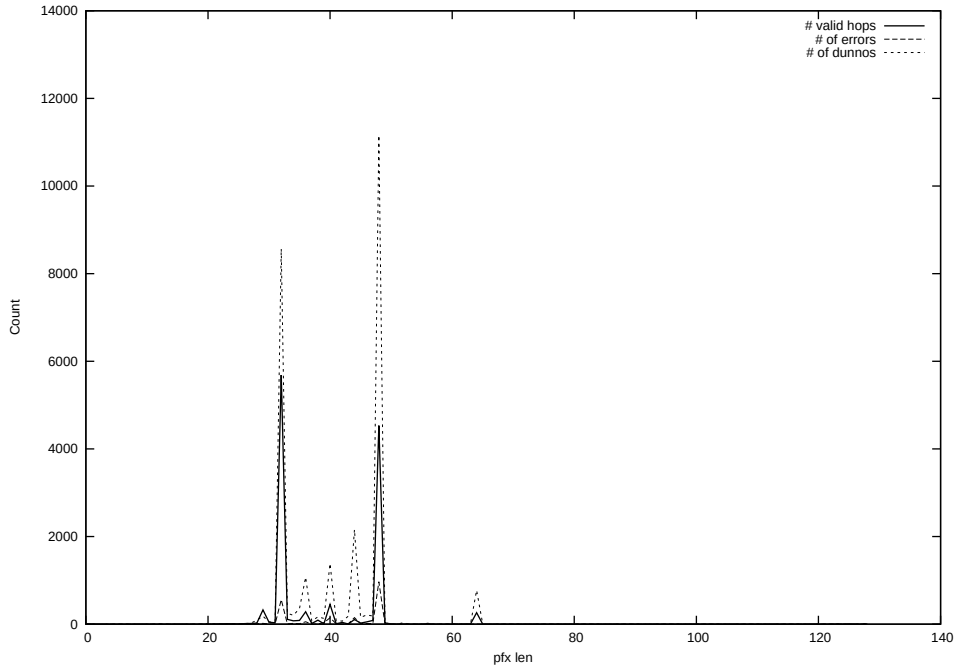
2013-06-16



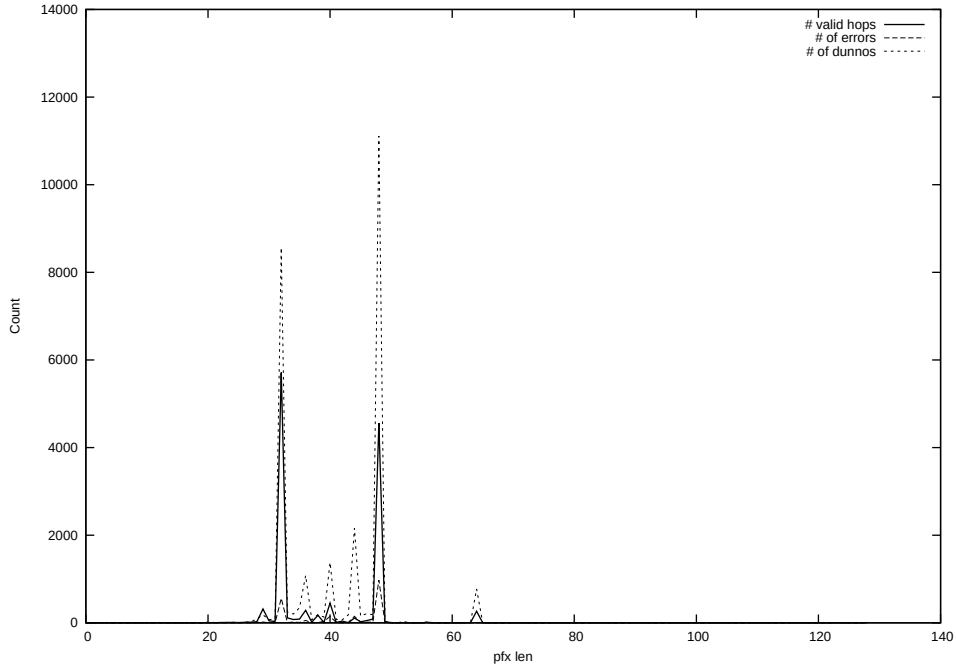
2013-06-17



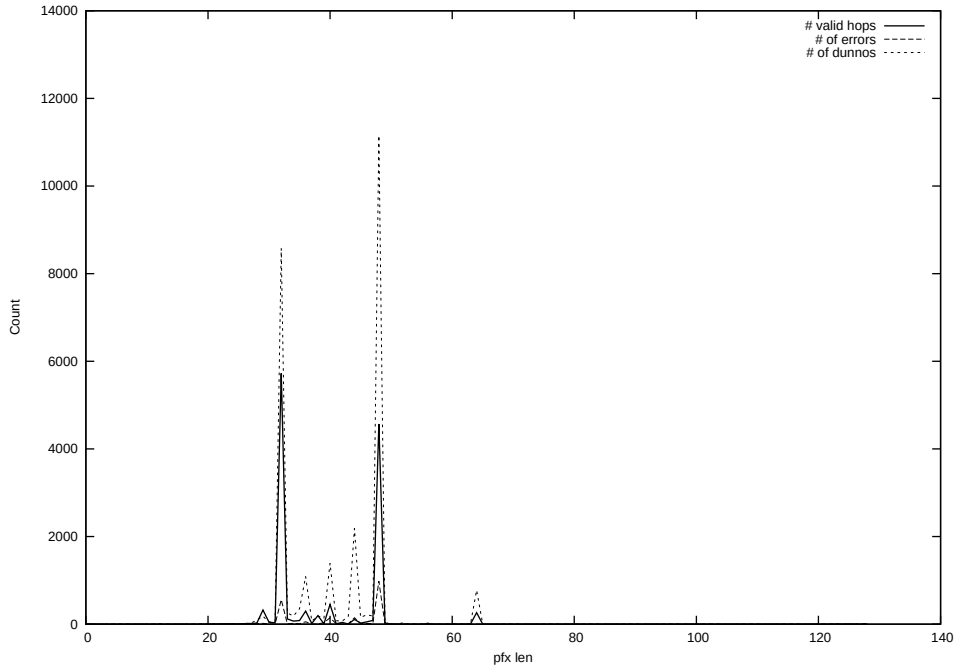
2013-06-18



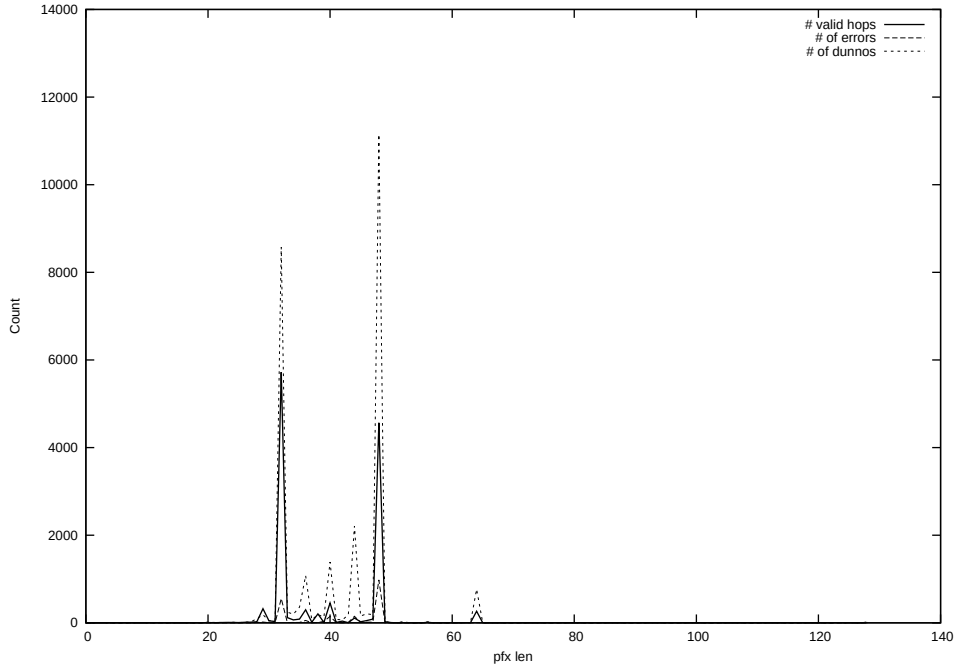
2013-06-19



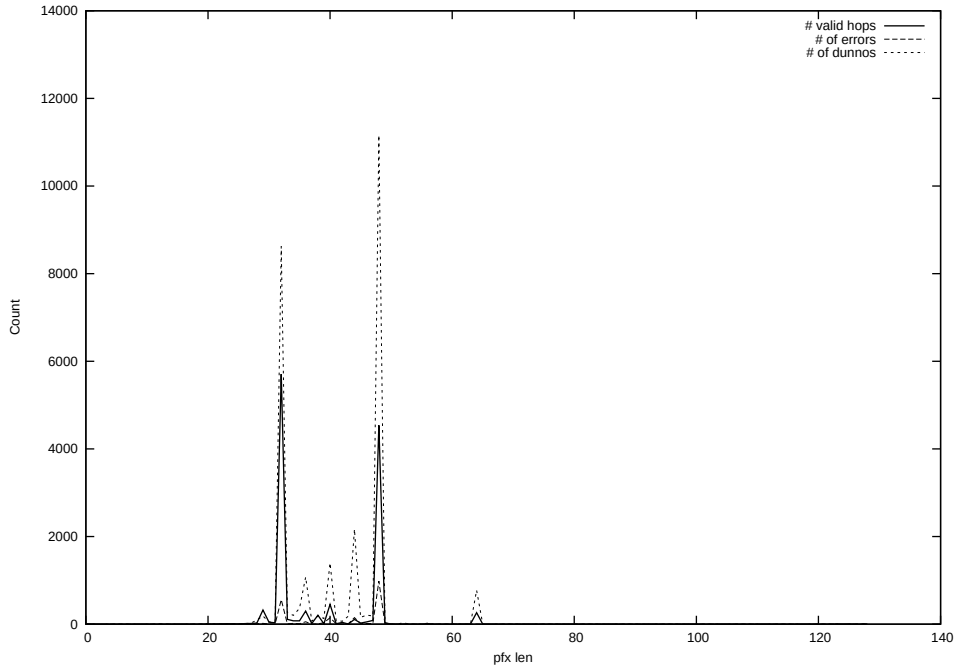
2013-06-20



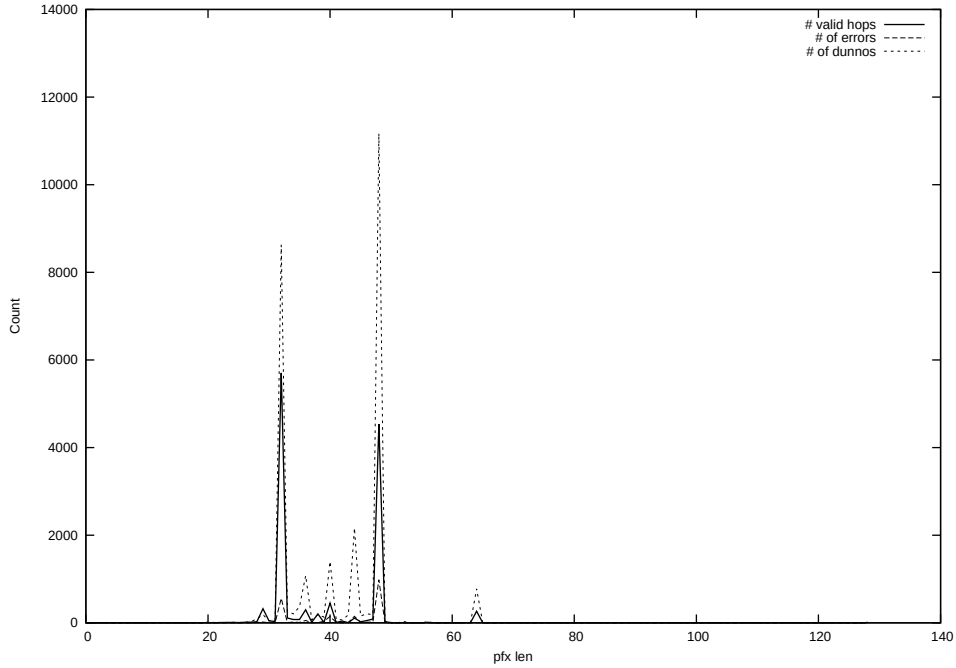
2013-06-21



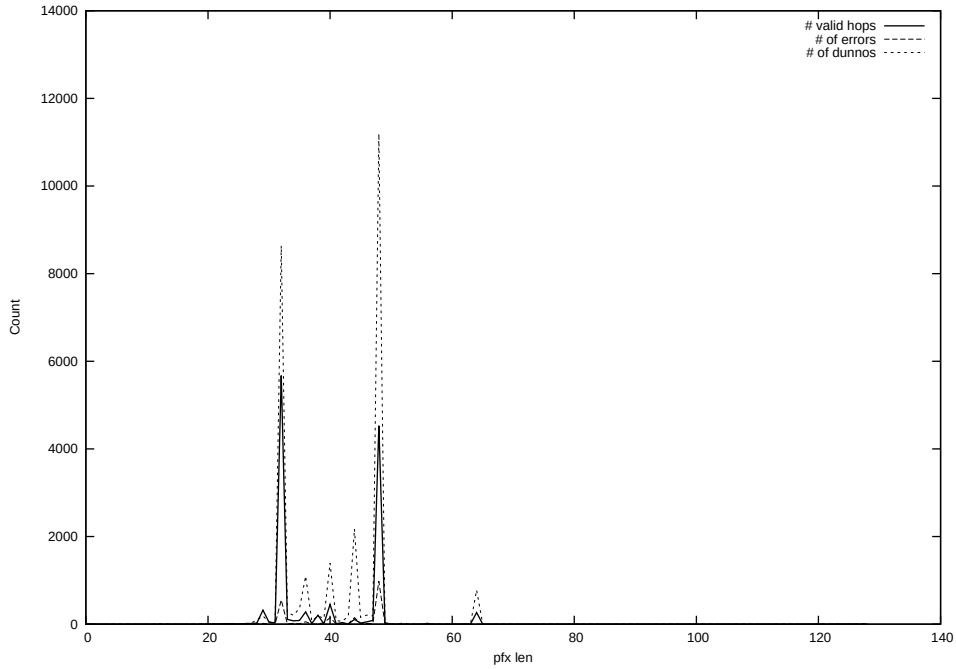
2013-06-22



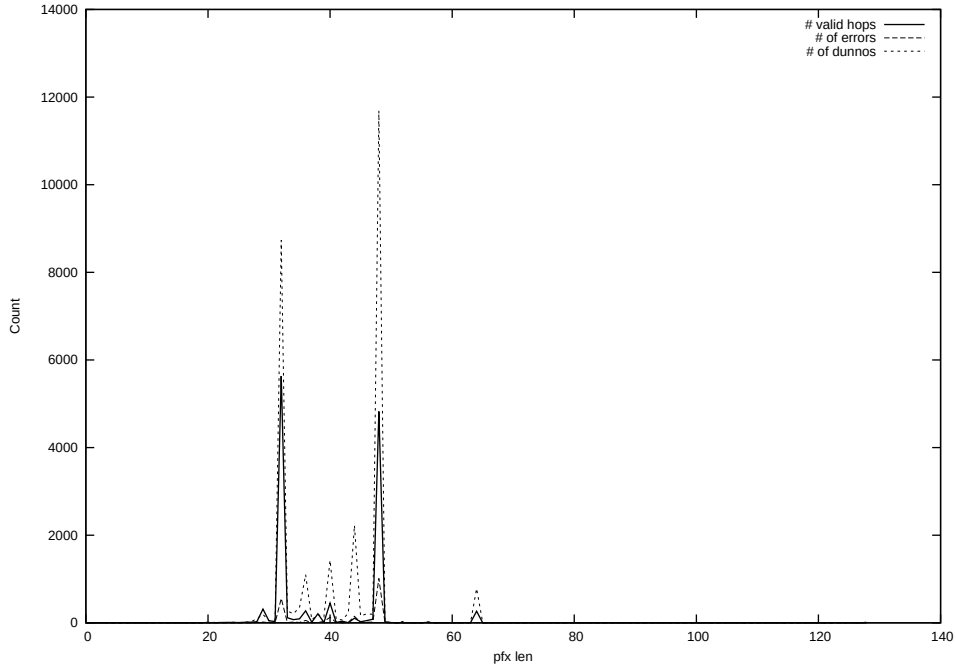
2013-06-23



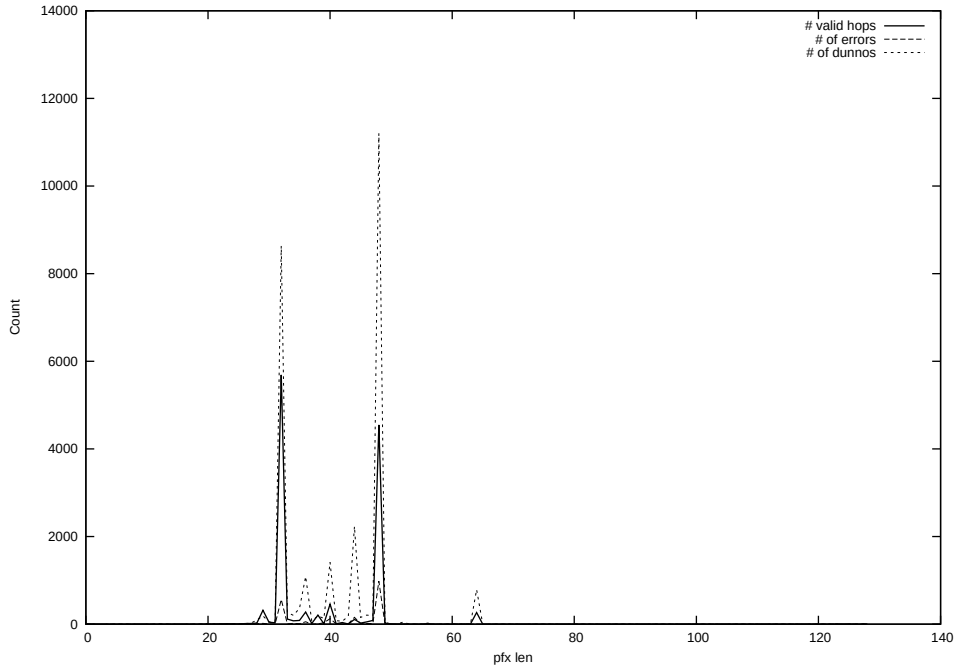
2013-06-24



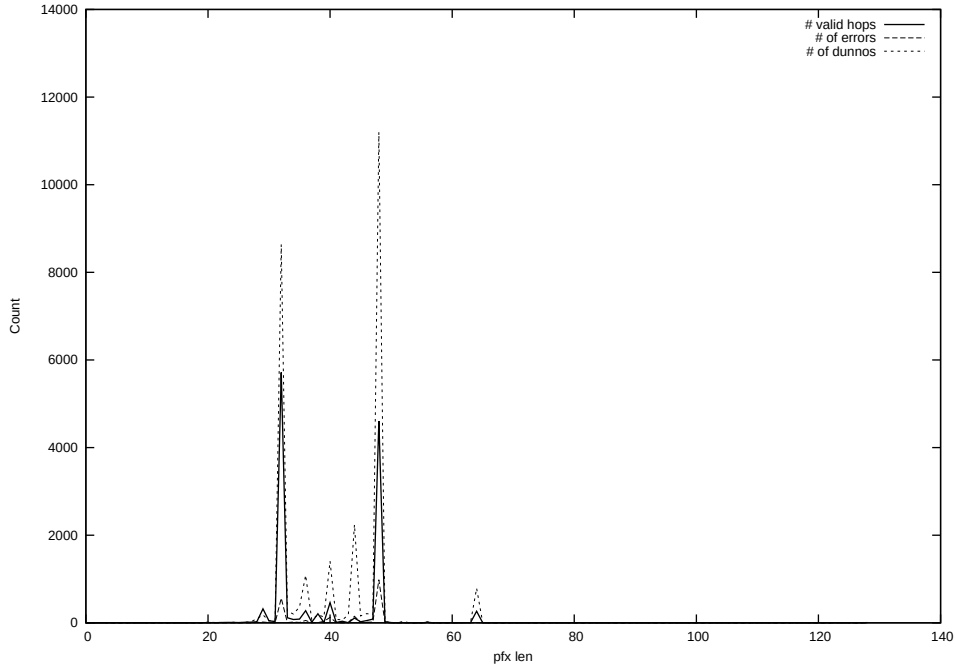
2013-06-25



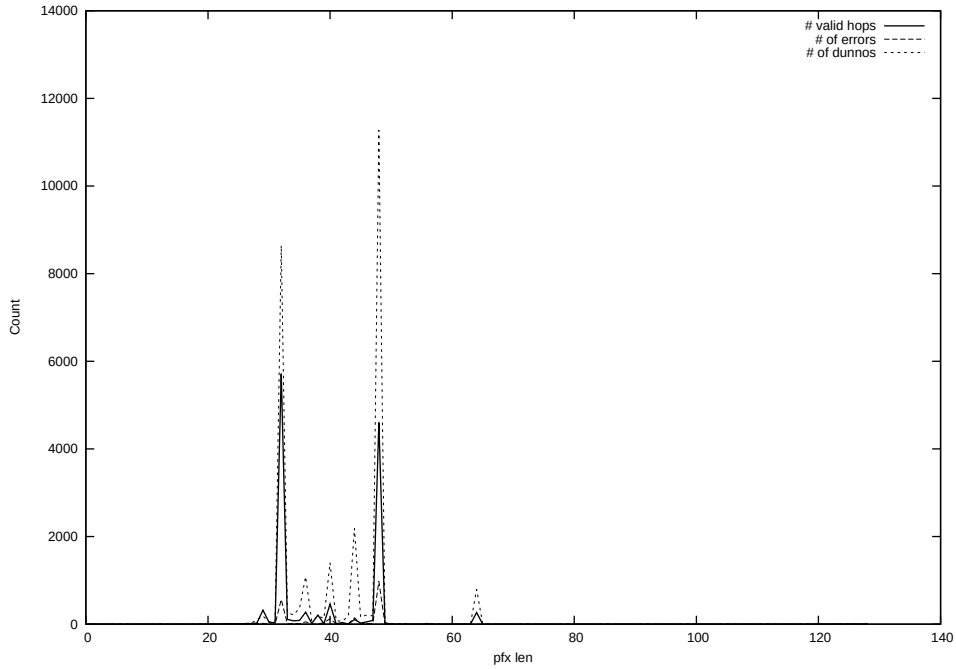
2013-06-26



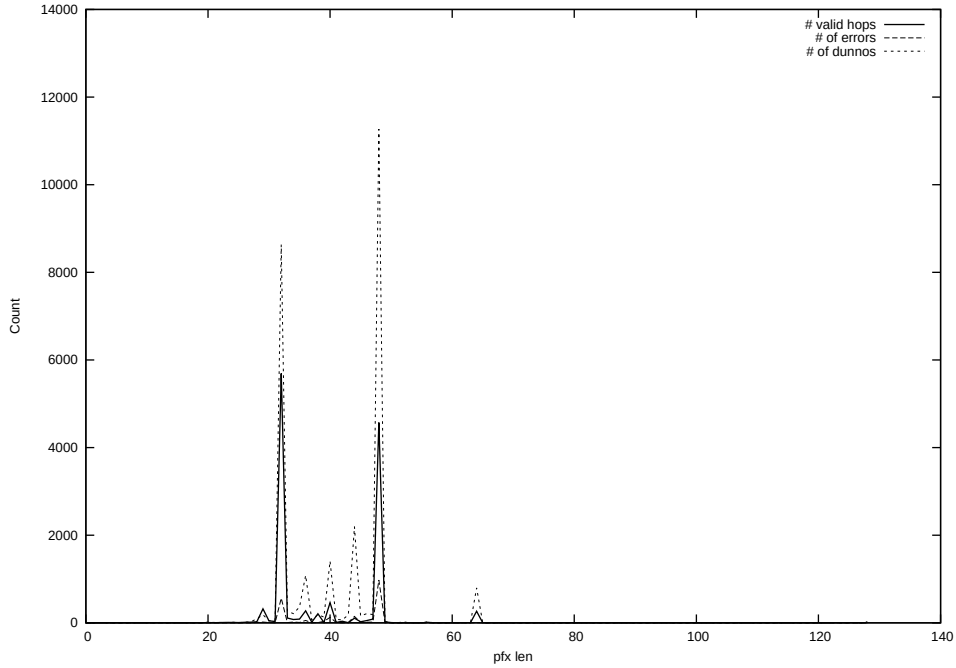
2013-06-27



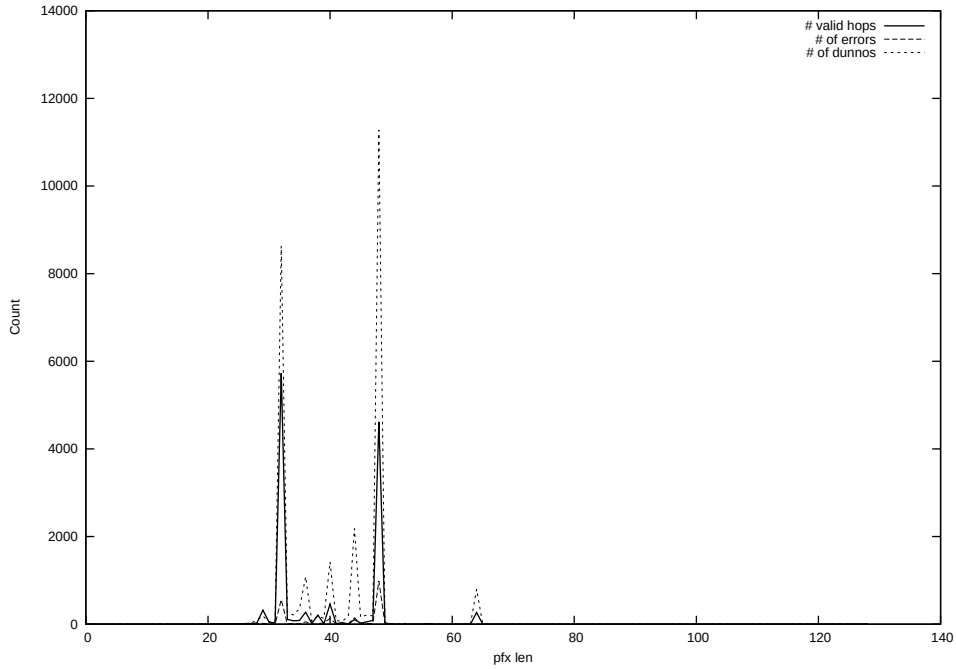
2013-06-28



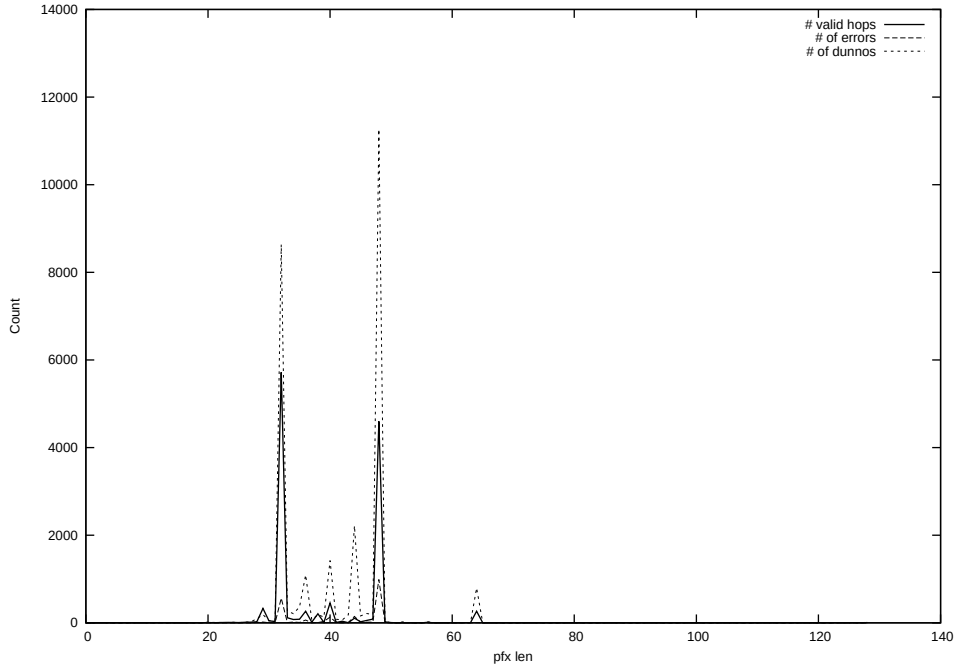
2013-06-29



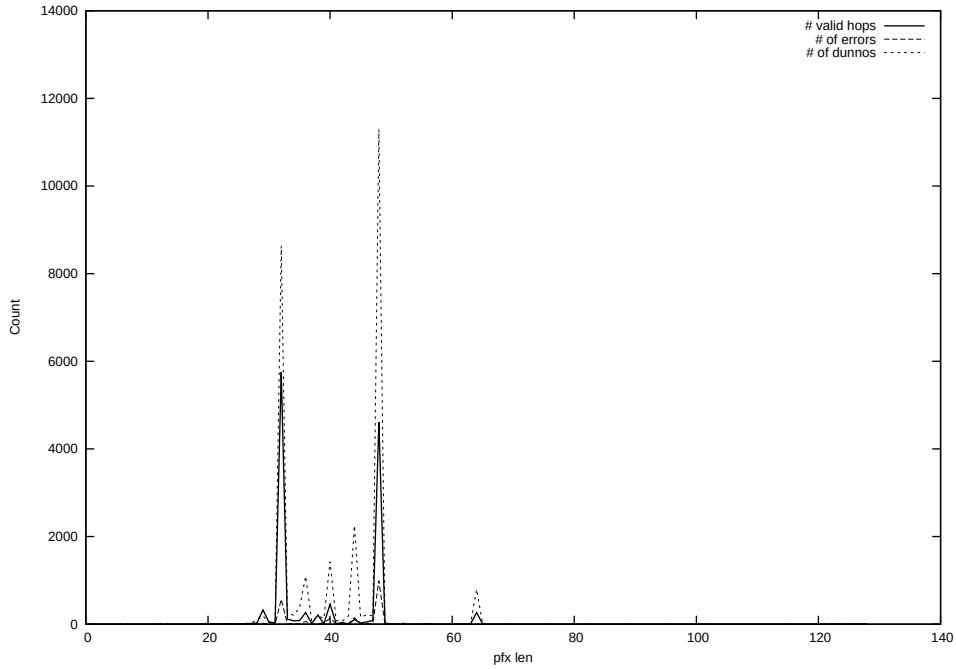
2013-06-30



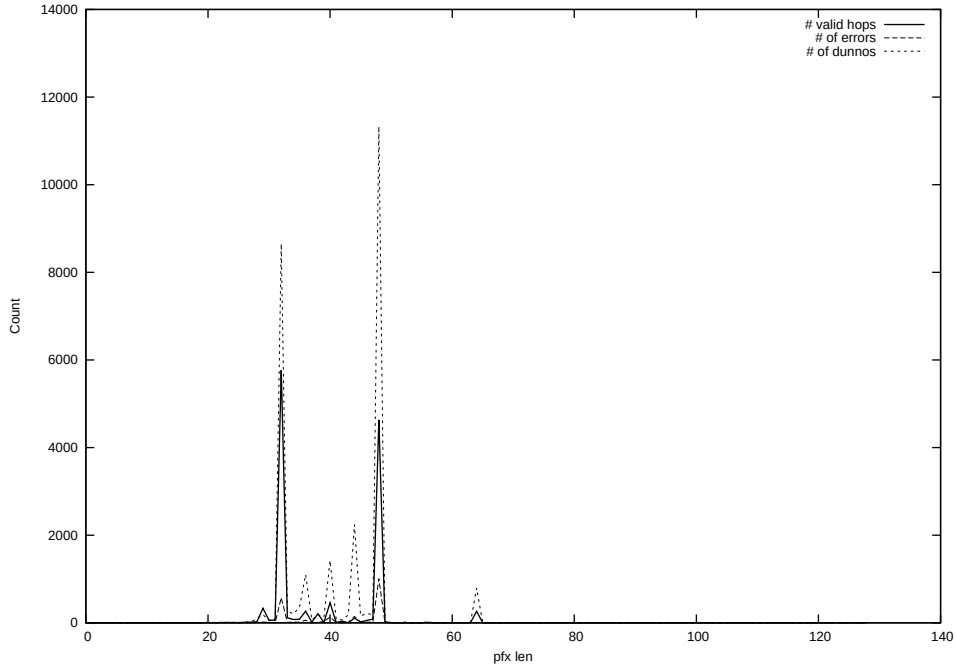
2013-07-01



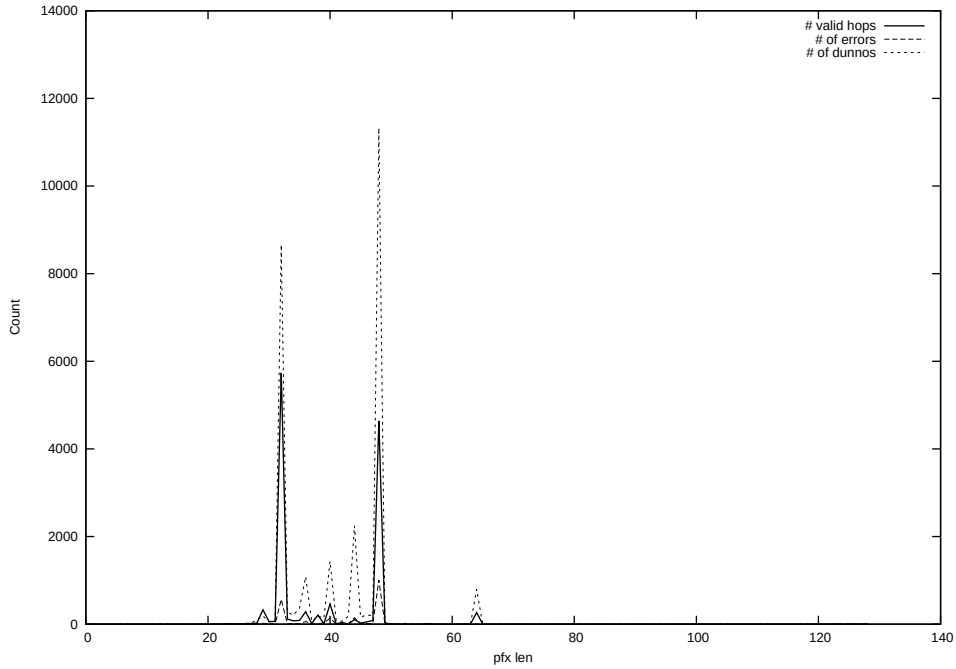
2013-07-02



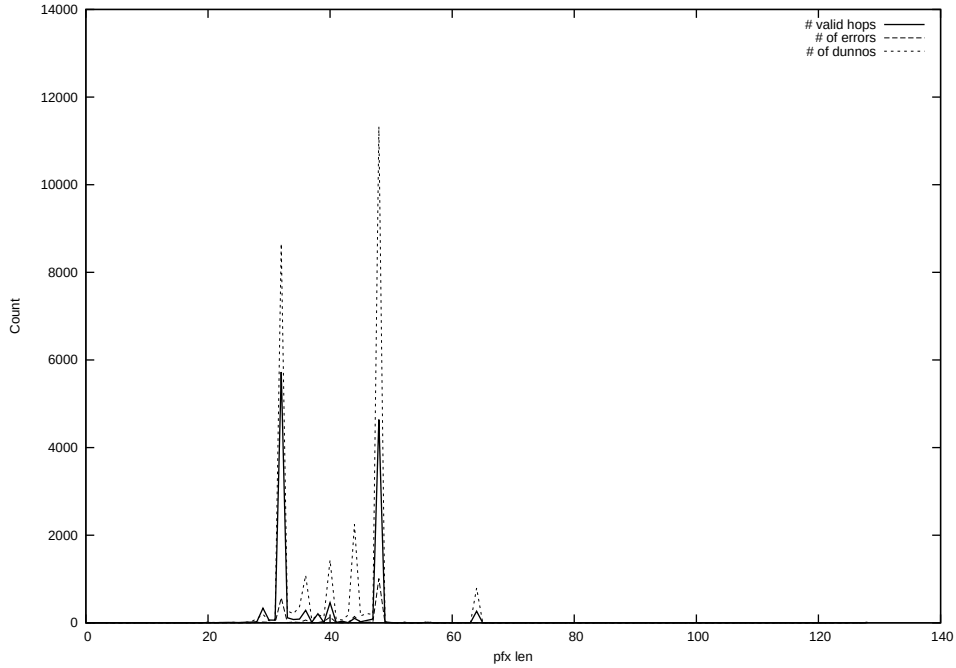
2013-07-03



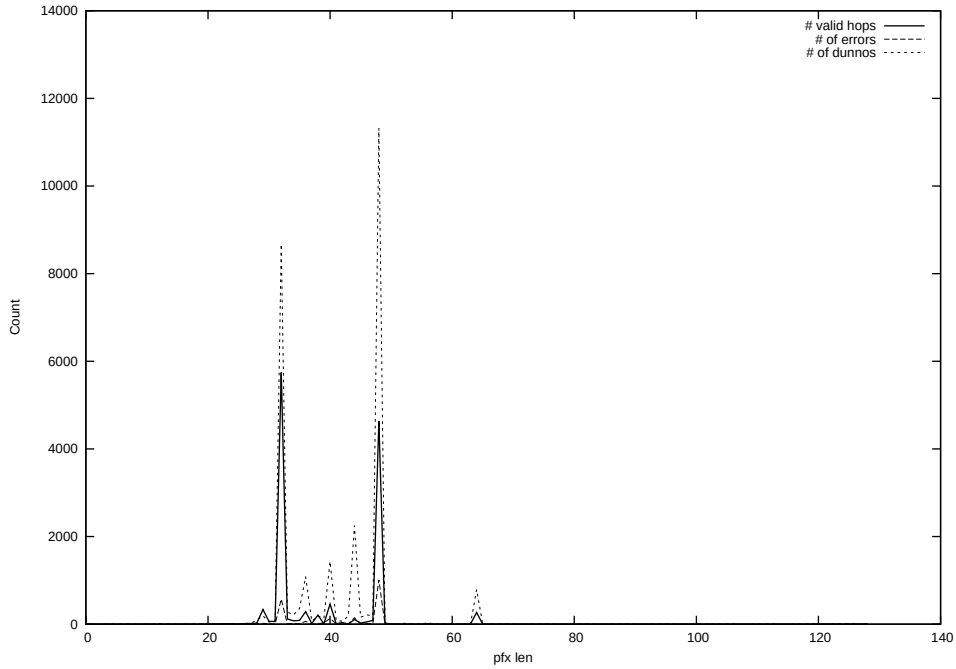
2013-07-04



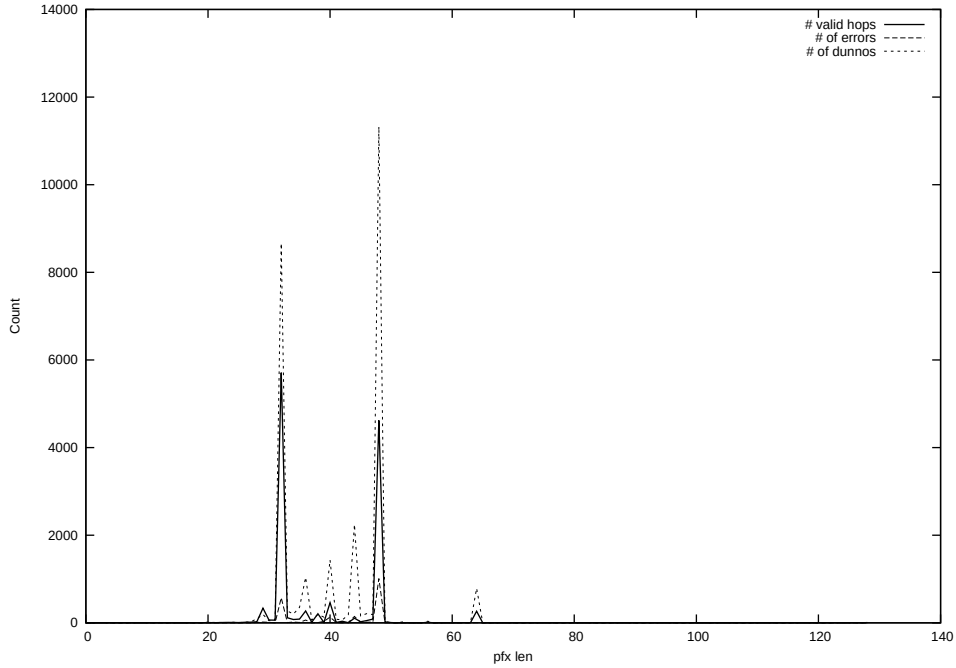
2013-07-05



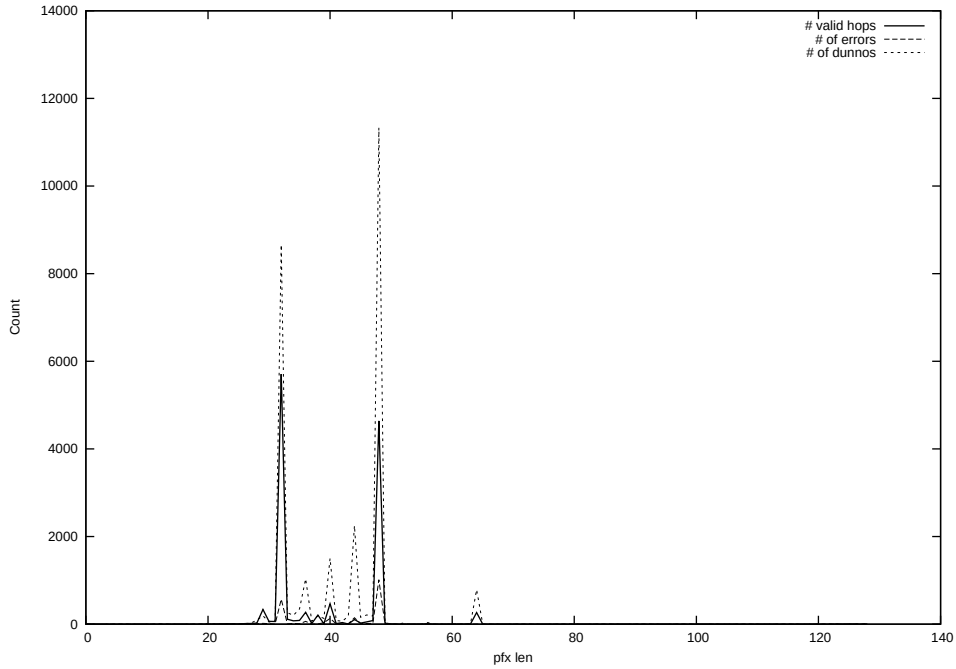
2013-07-06



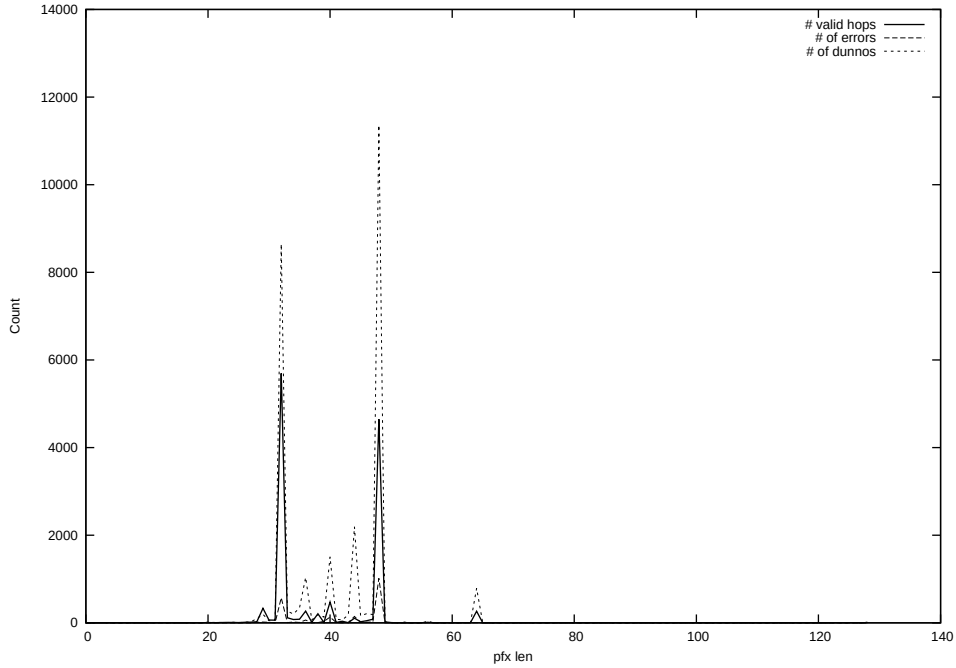
2013-07-07



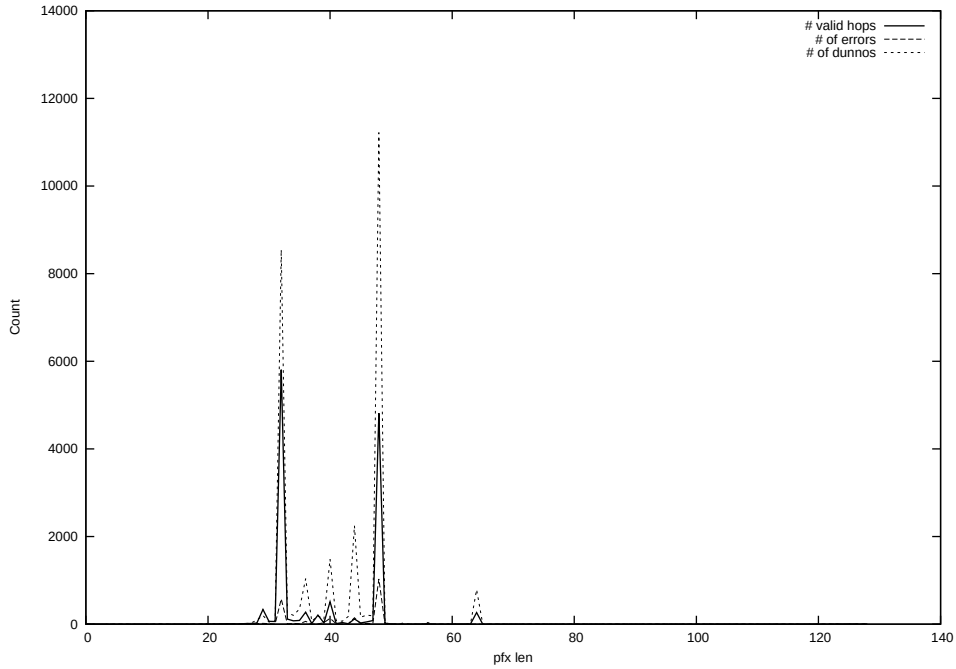
2013-07-08



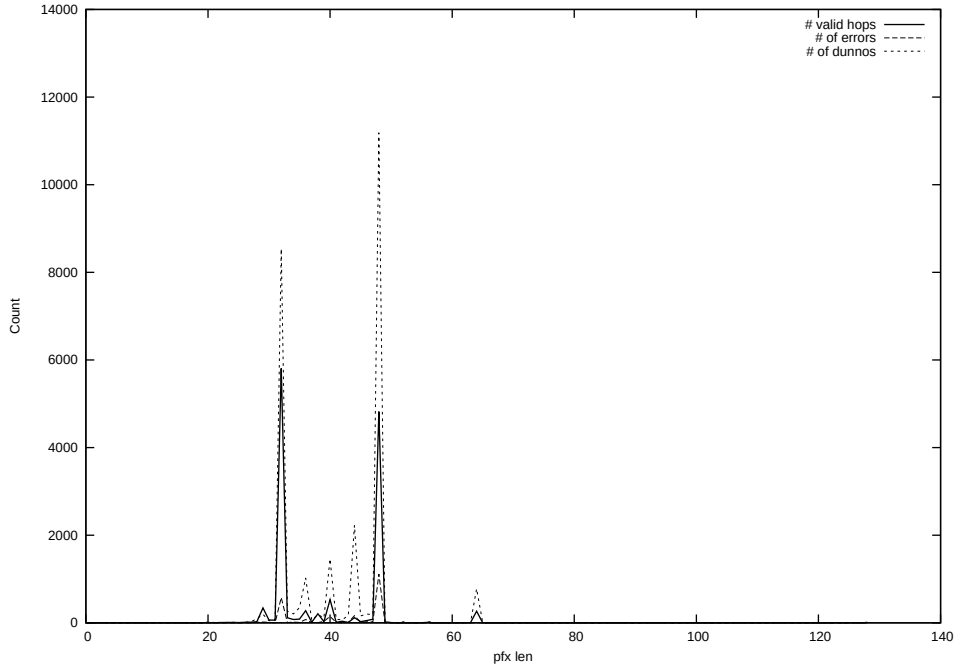
2013-07-09



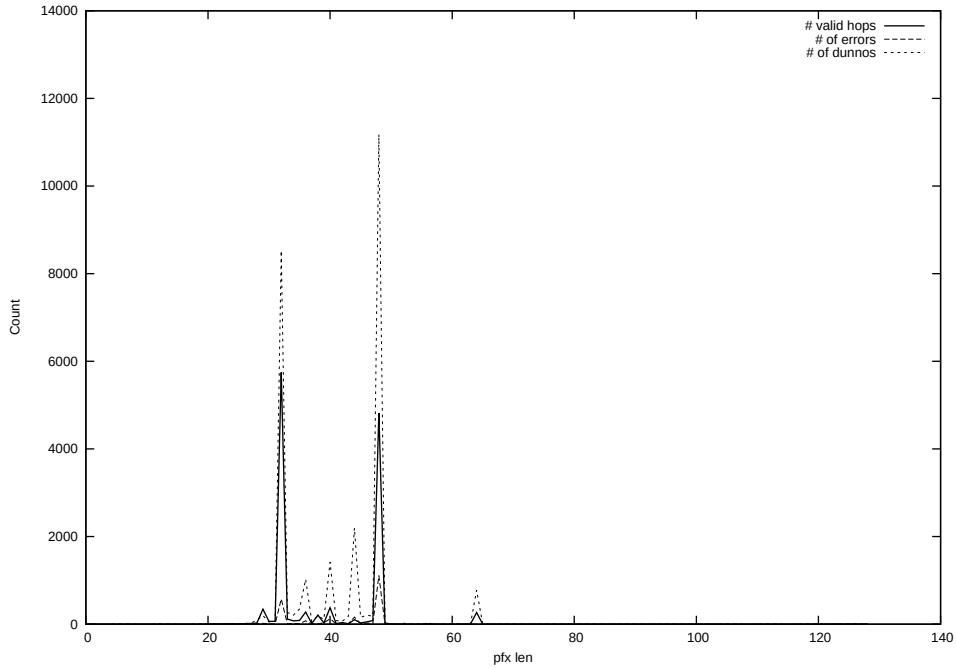
2013-07-10



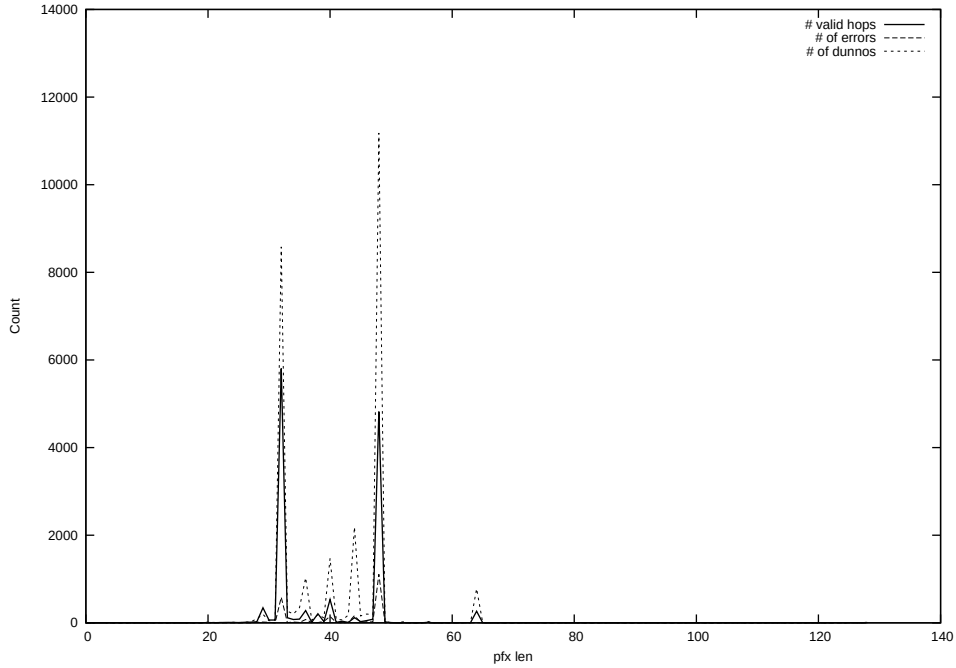
2013-07-11



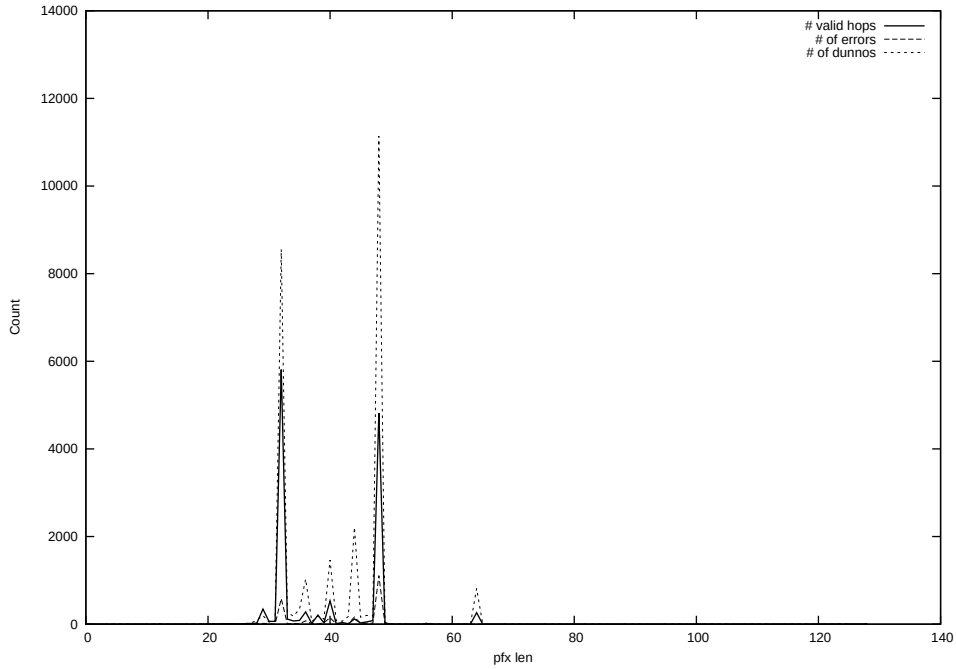
2013-07-12



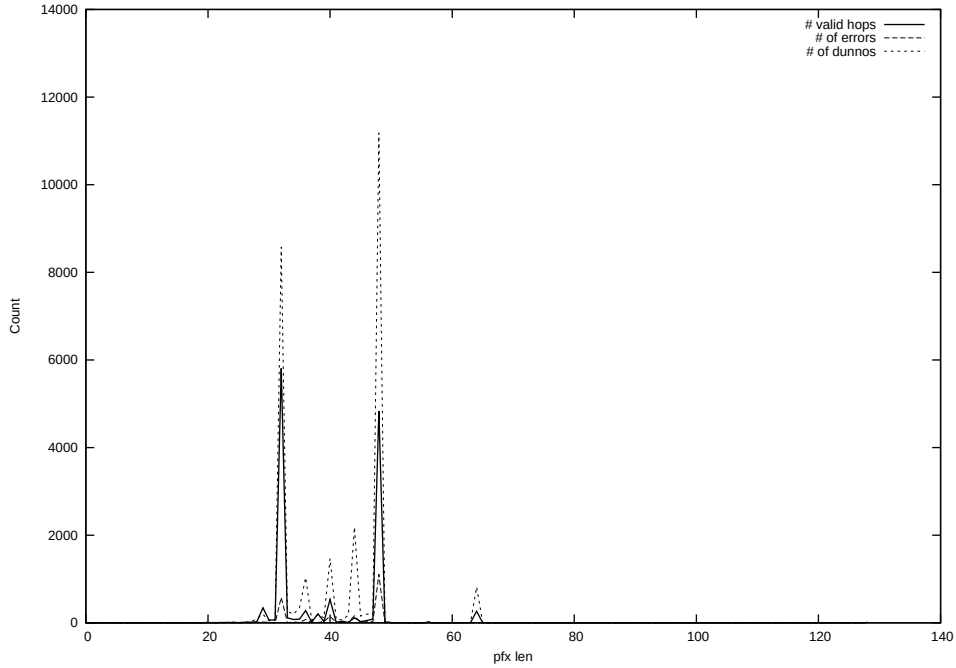
2013-07-13



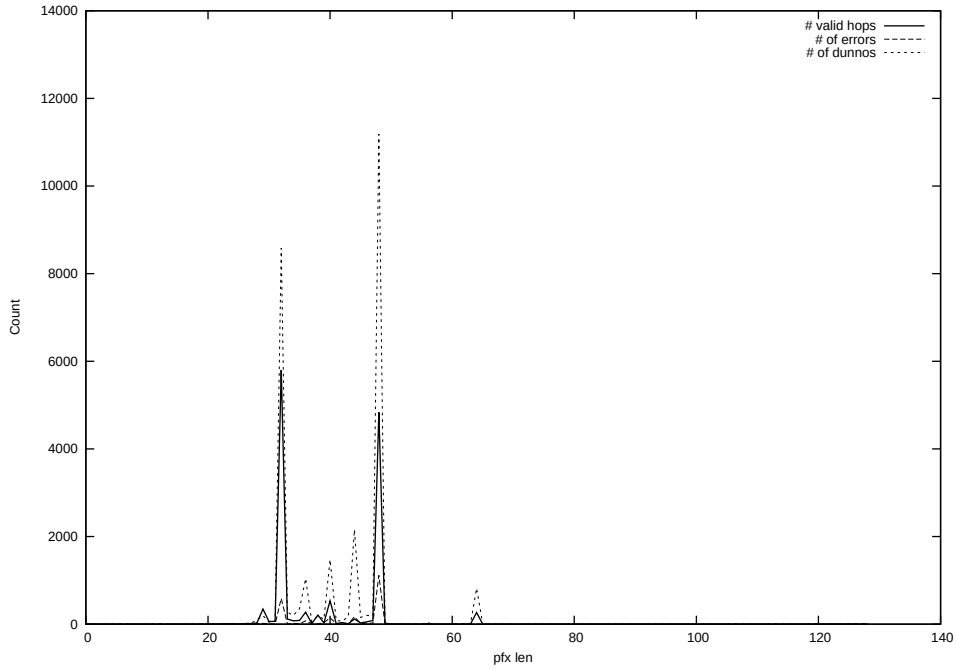
2013-07-14



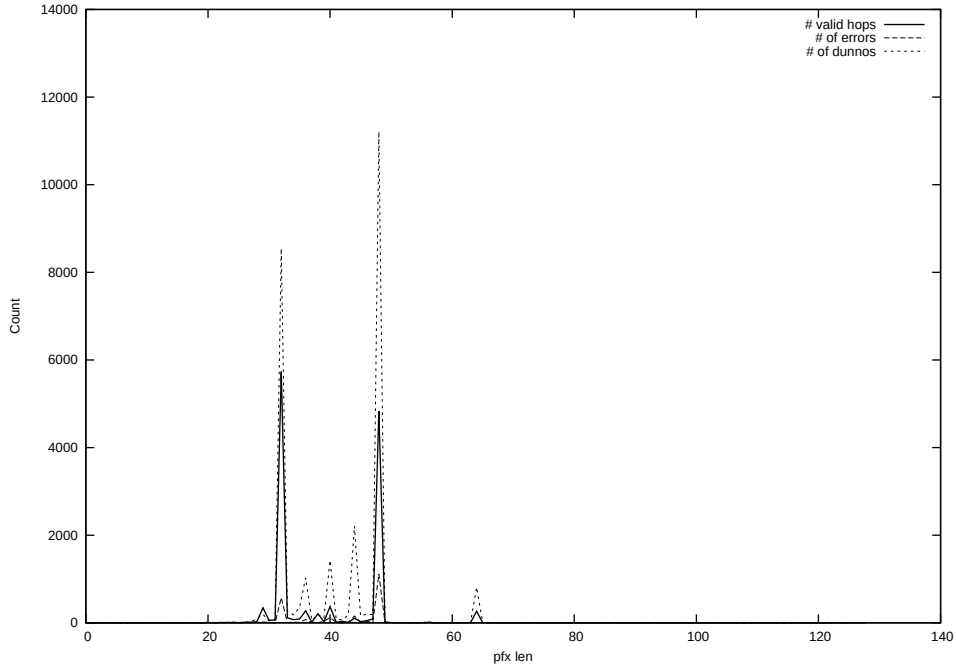
2013-07-15



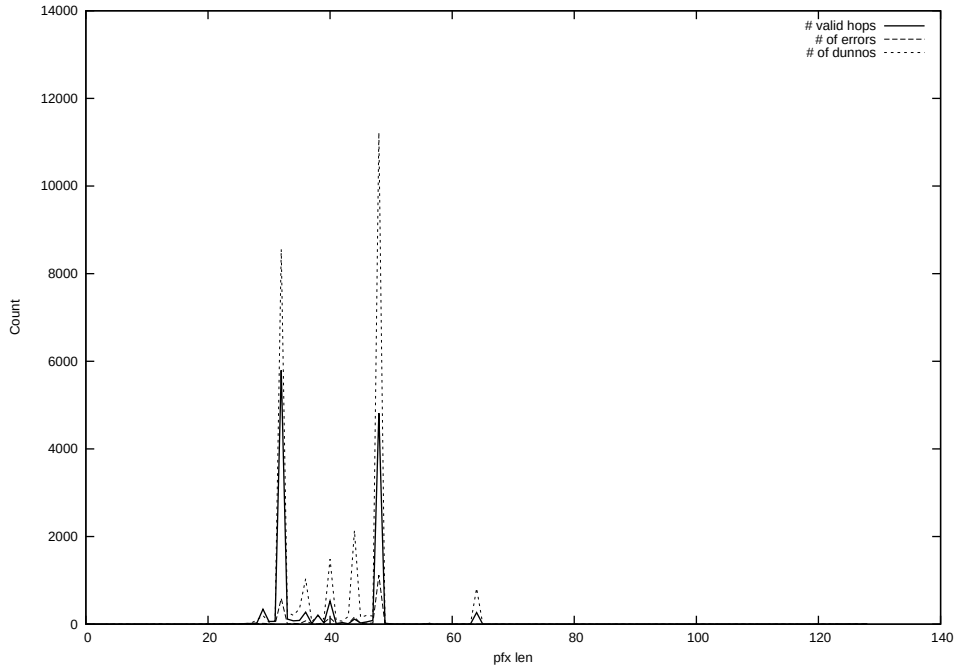
2013-07-16



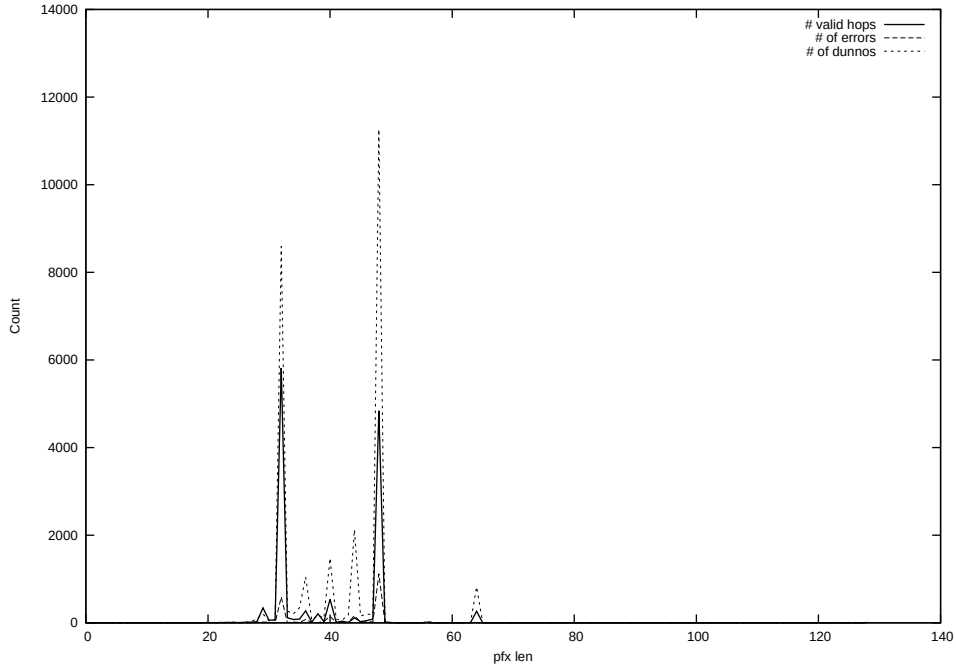
2013-07-17



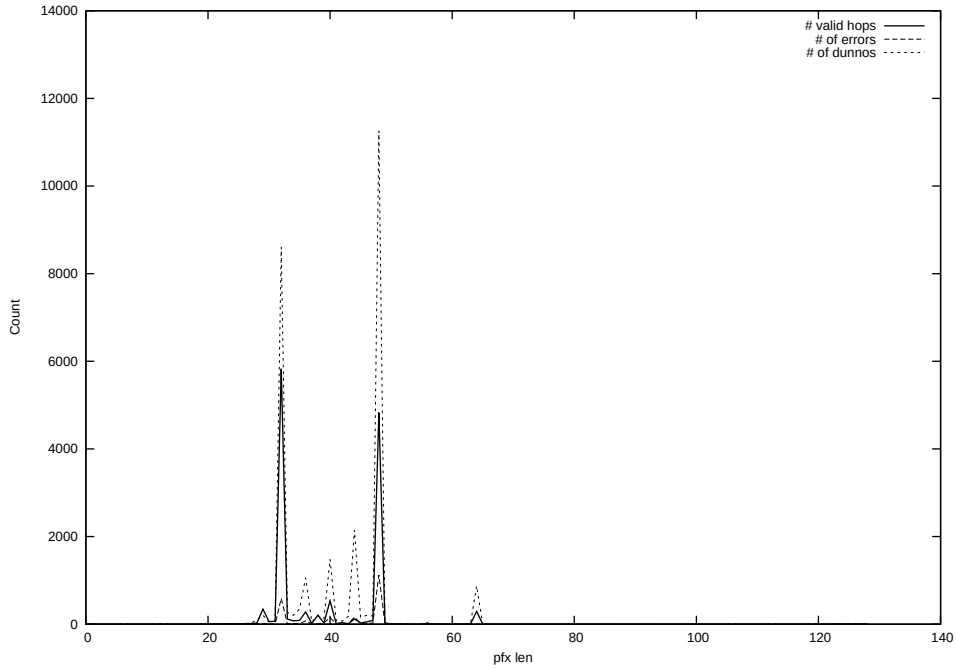
2013-07-18



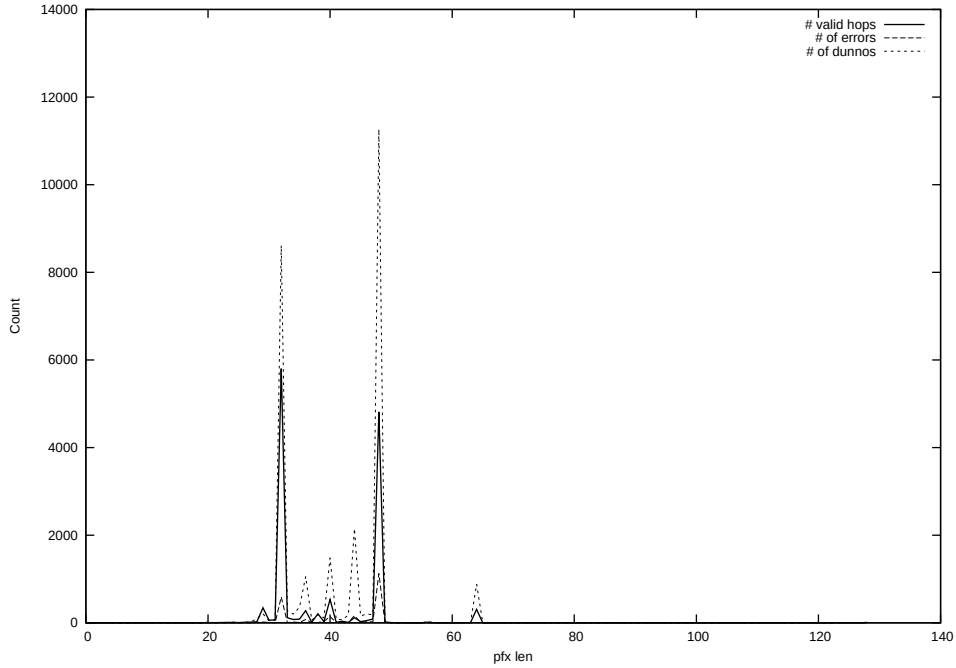
2013-07-19



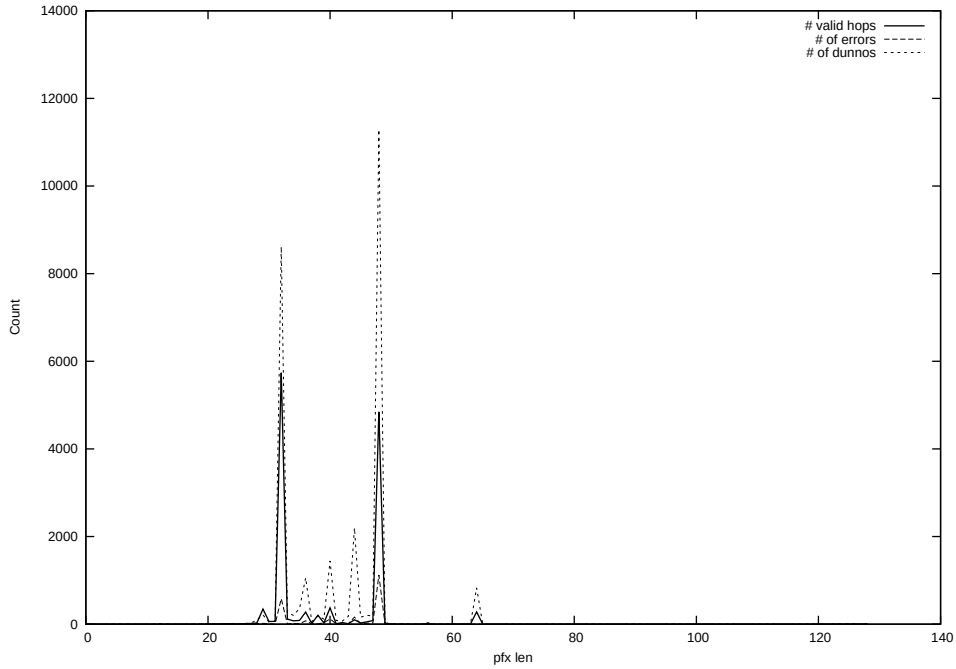
2013-07-20



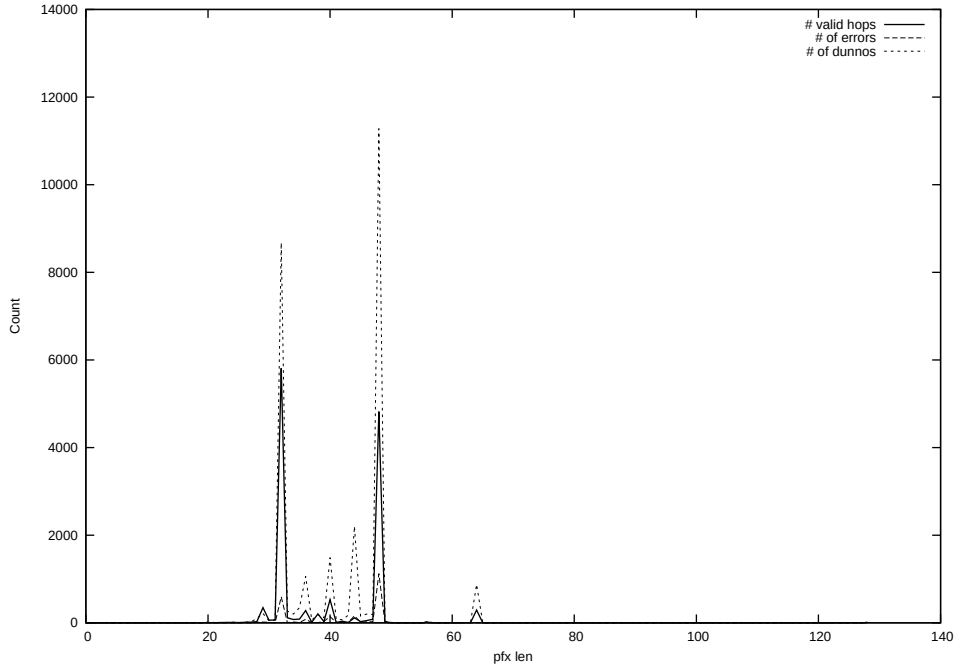
2013-07-21



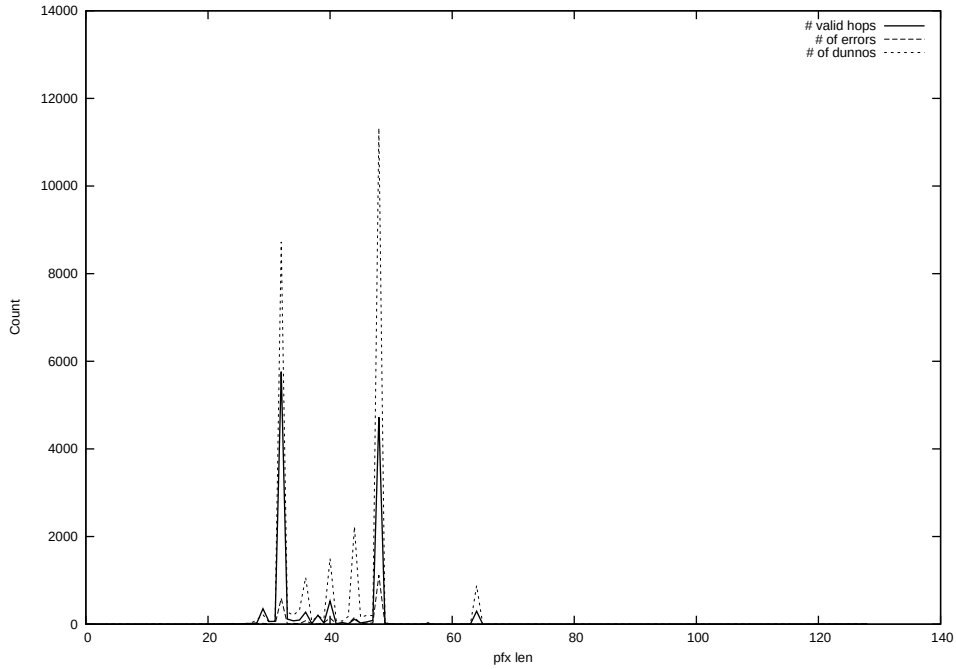
2013-07-22



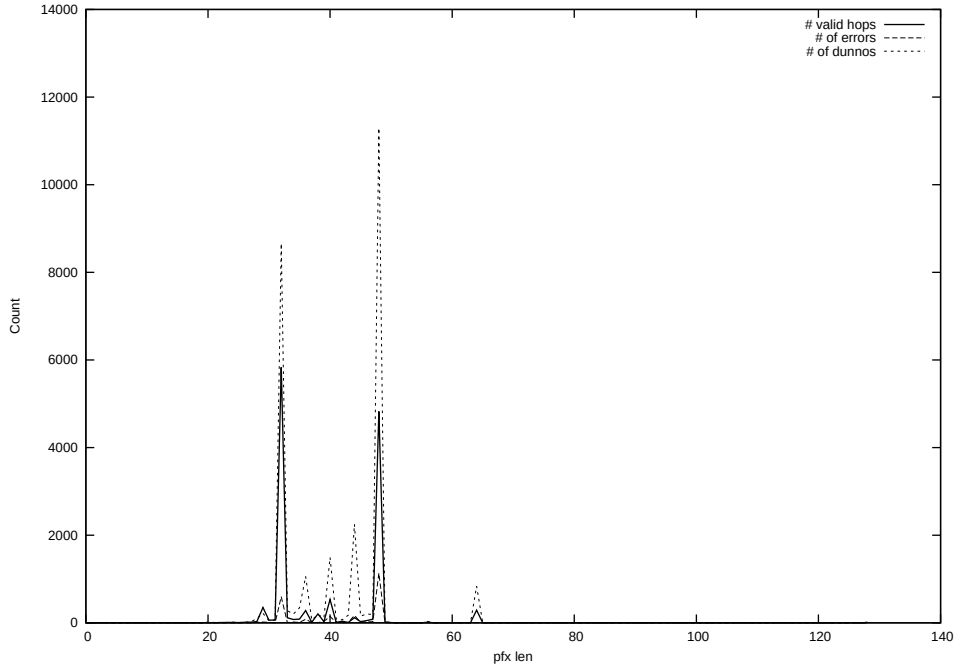
2013-07-23



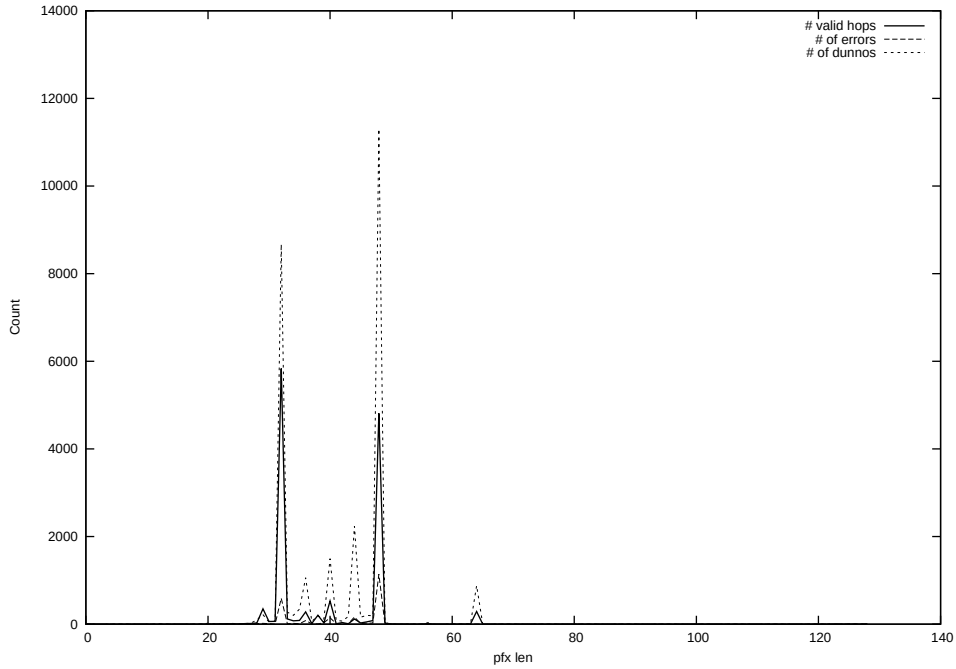
2013-07-24



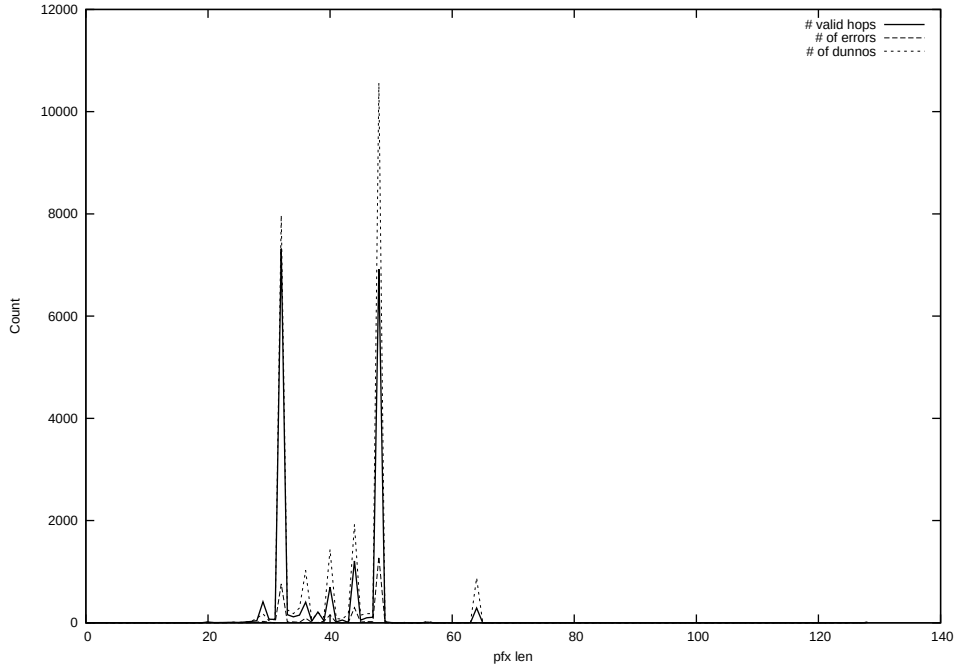
2013-07-25



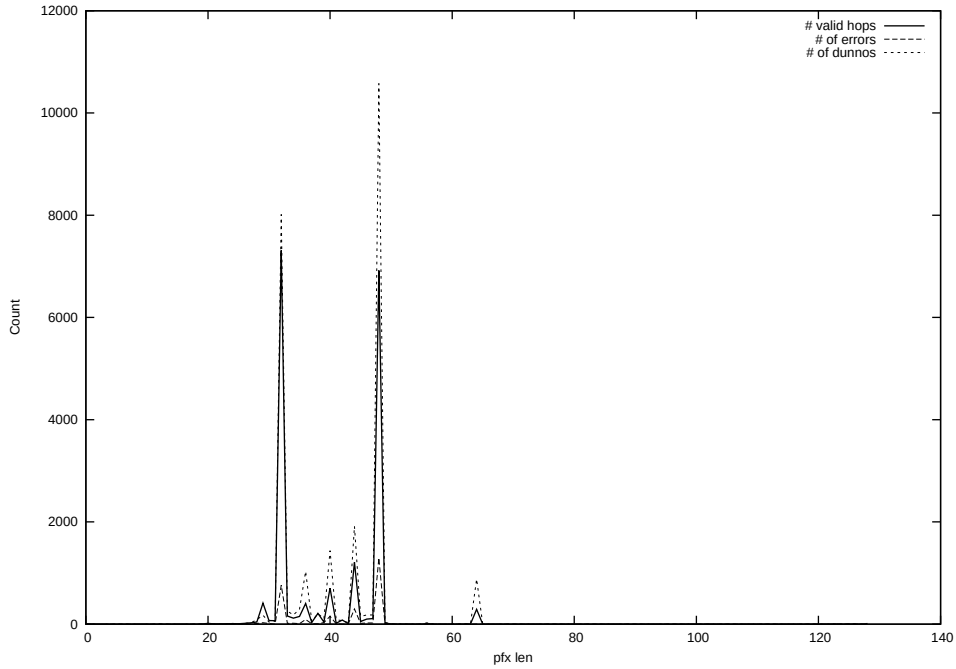
2013-07-26



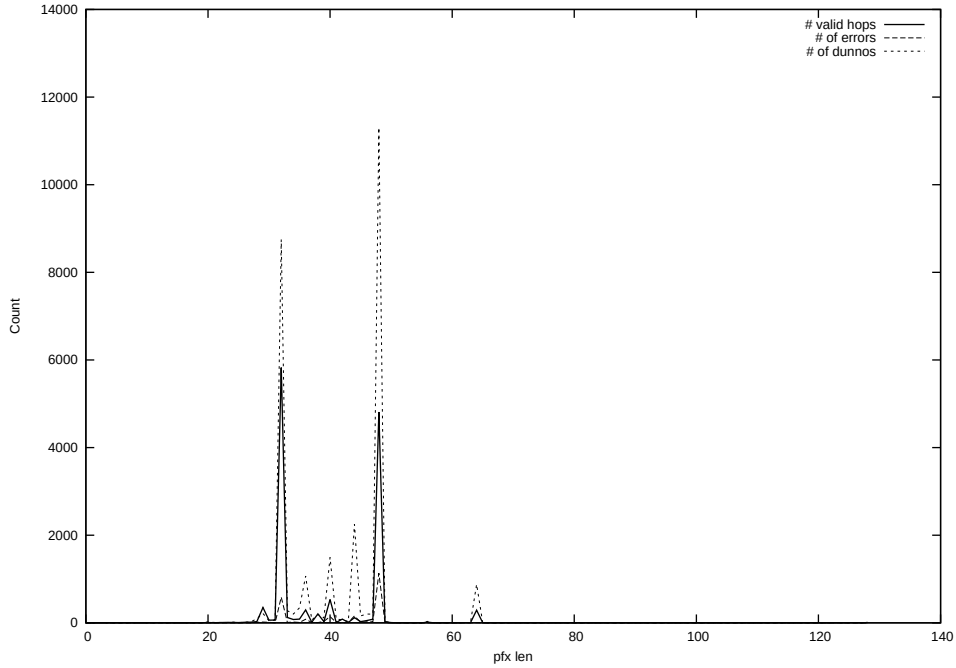
2013-07-27



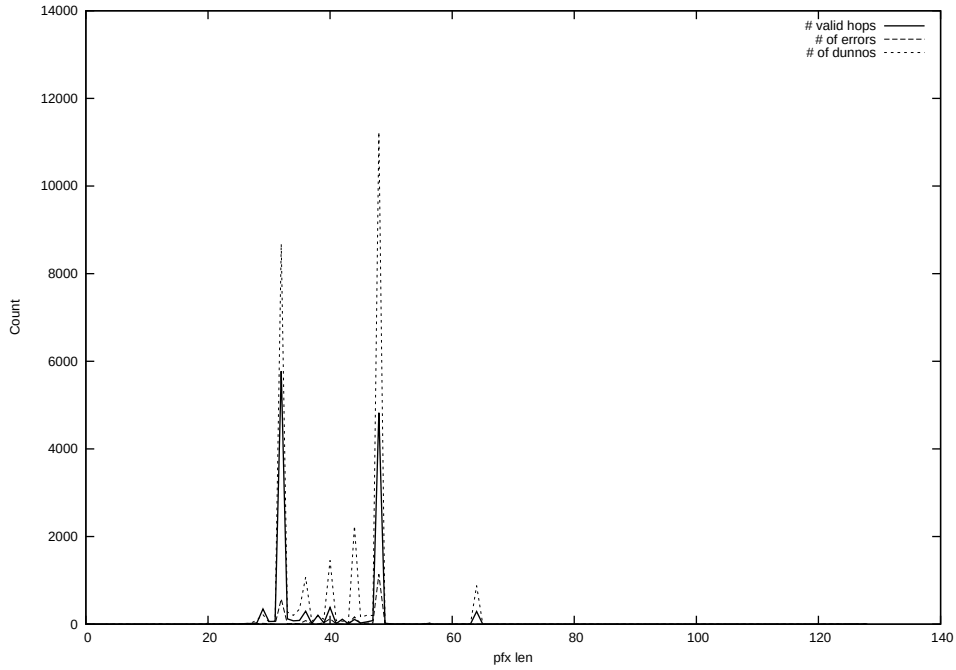
2013-07-28



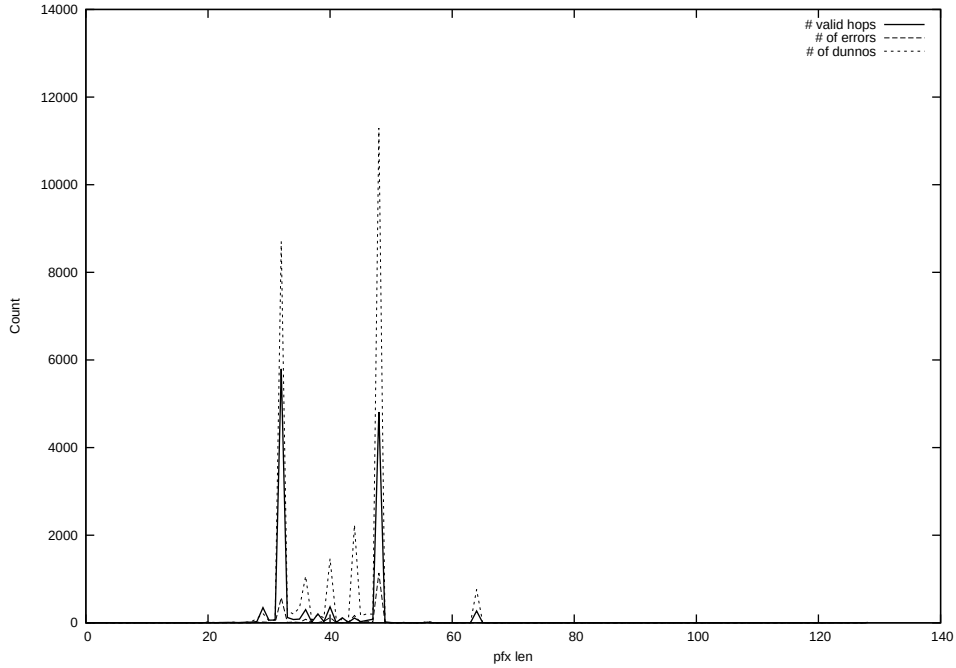
2013-07-29



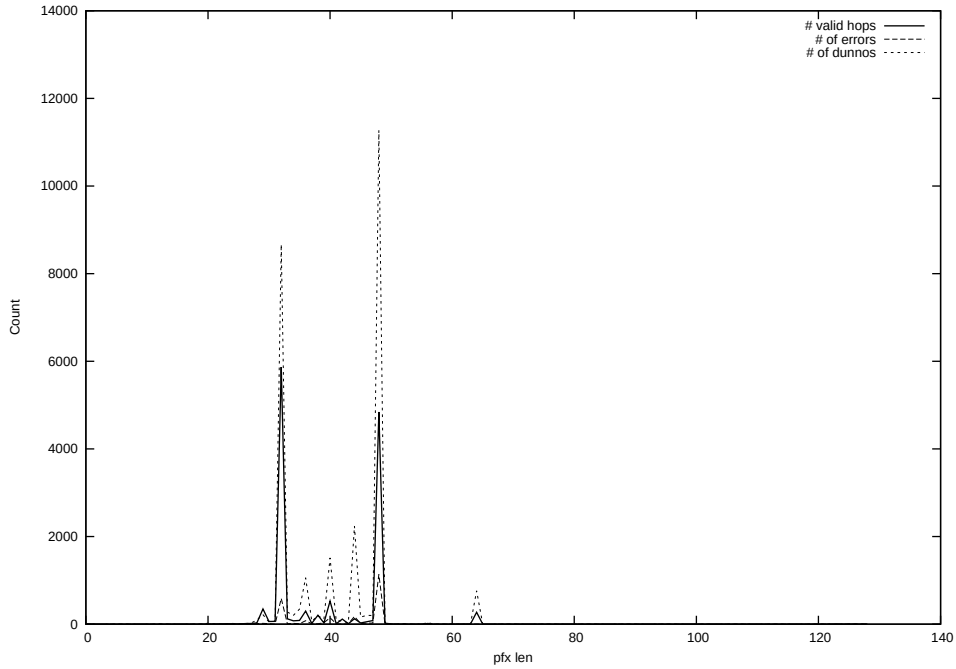
2013-07-30



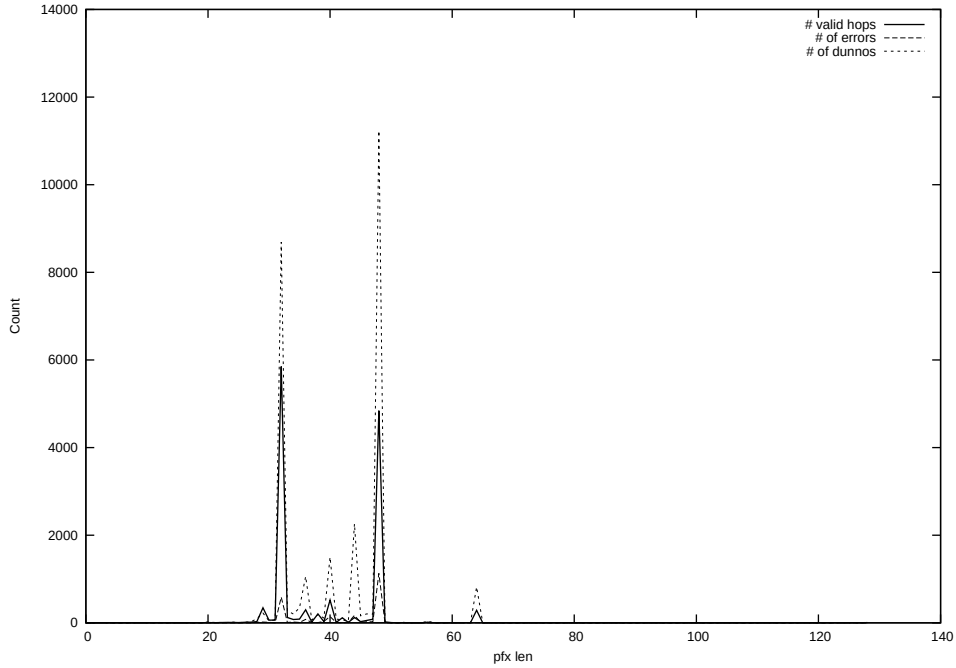
2013-07-31



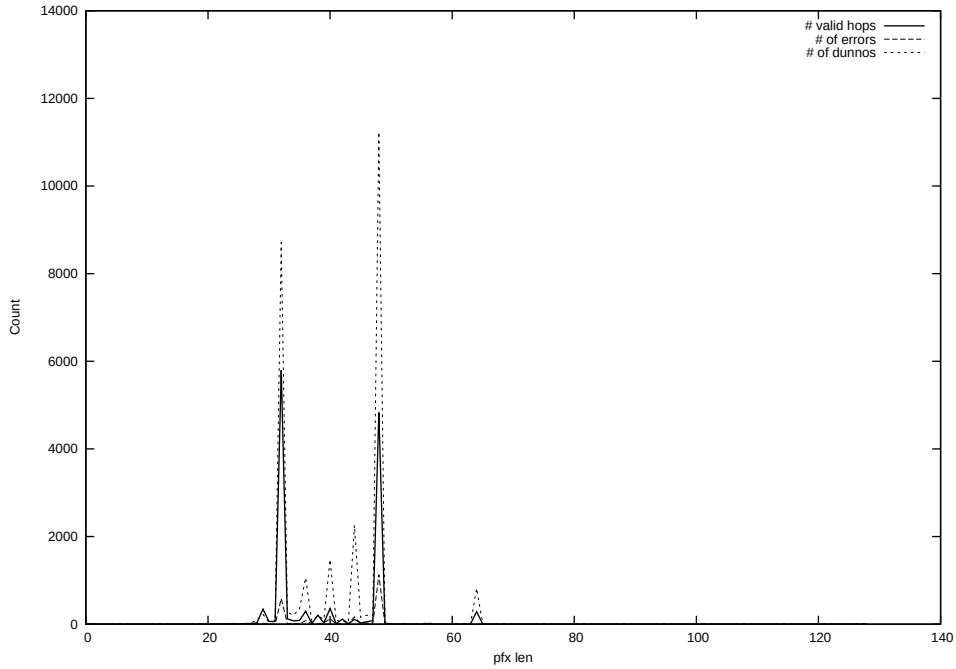
2013-08-01



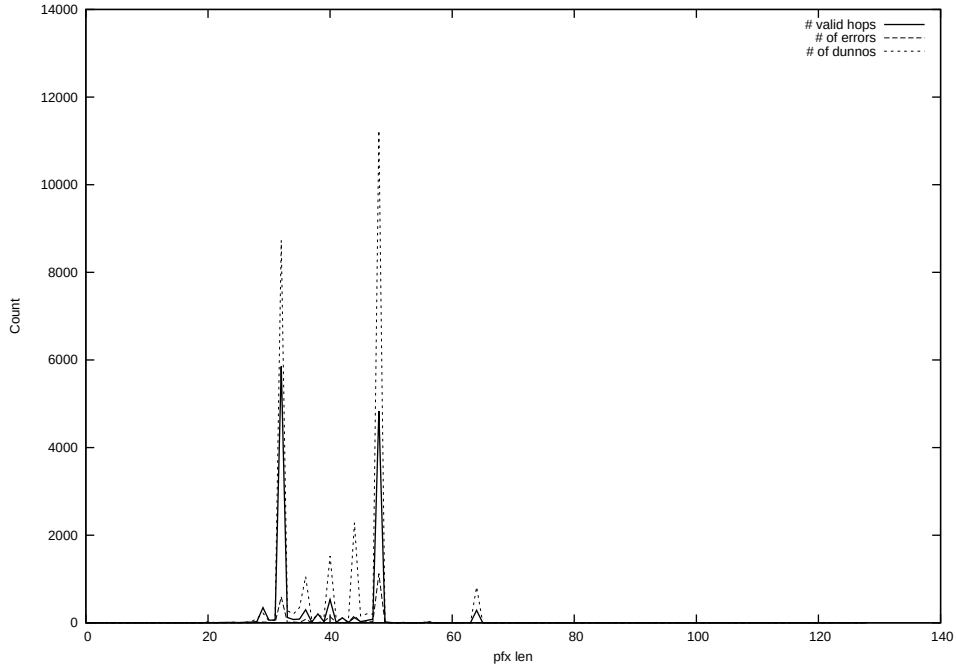
2013-08-02



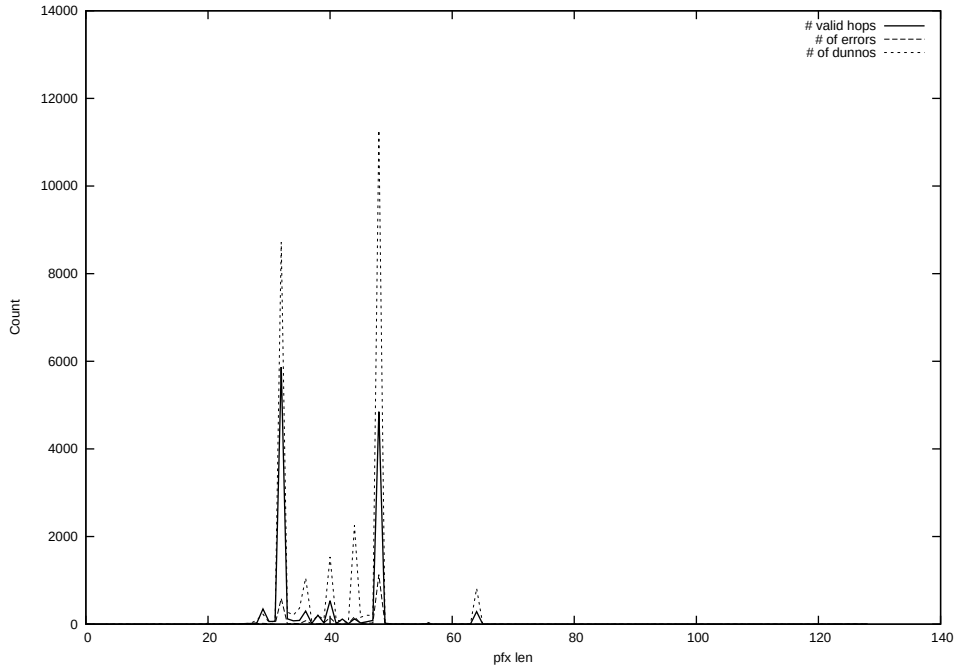
2013-08-03



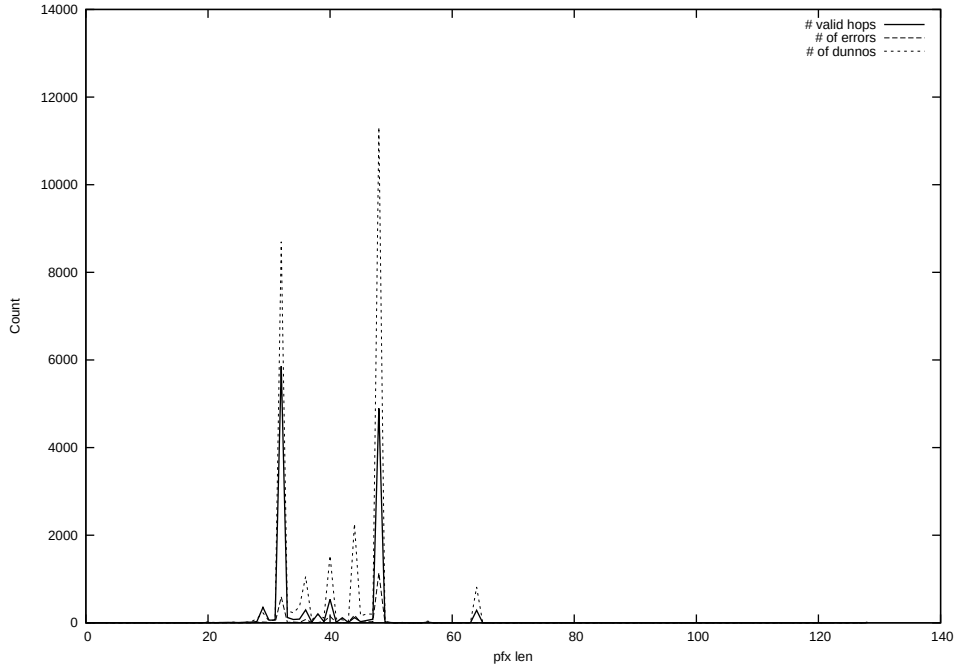
2013-08-04



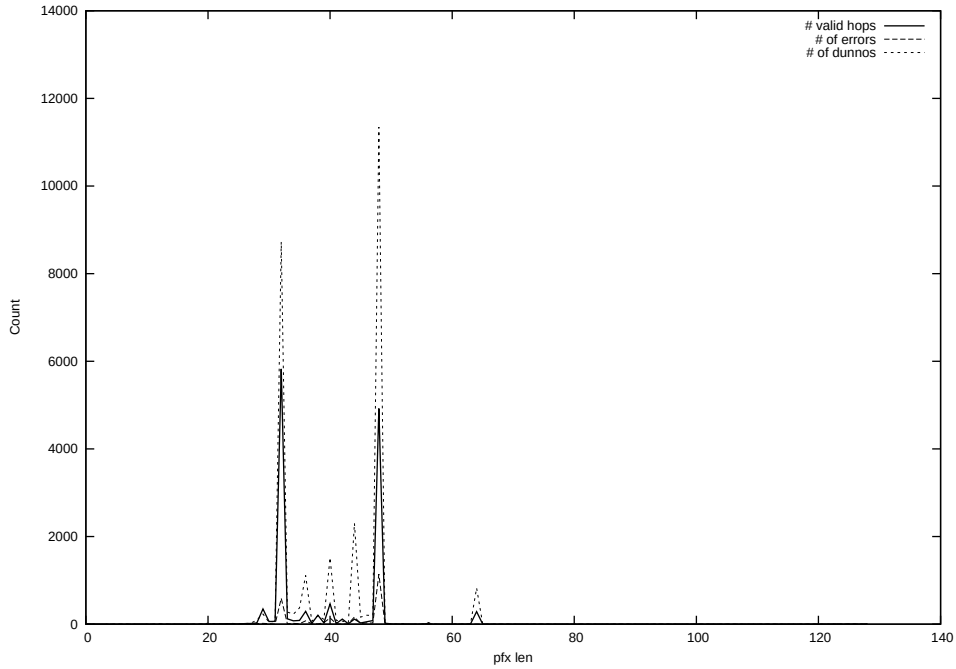
2013-08-05



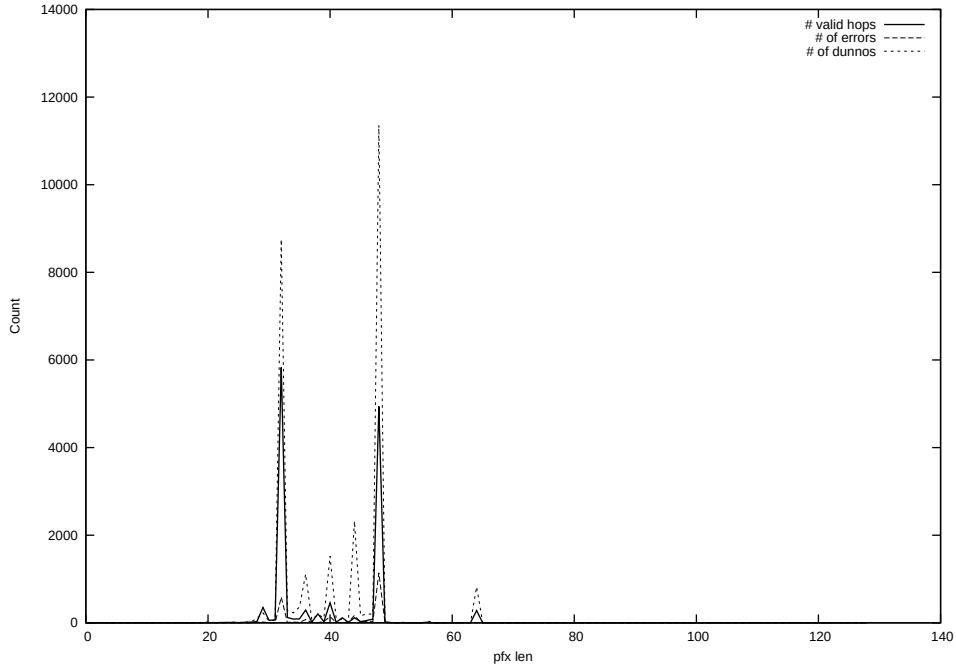
2013-08-06



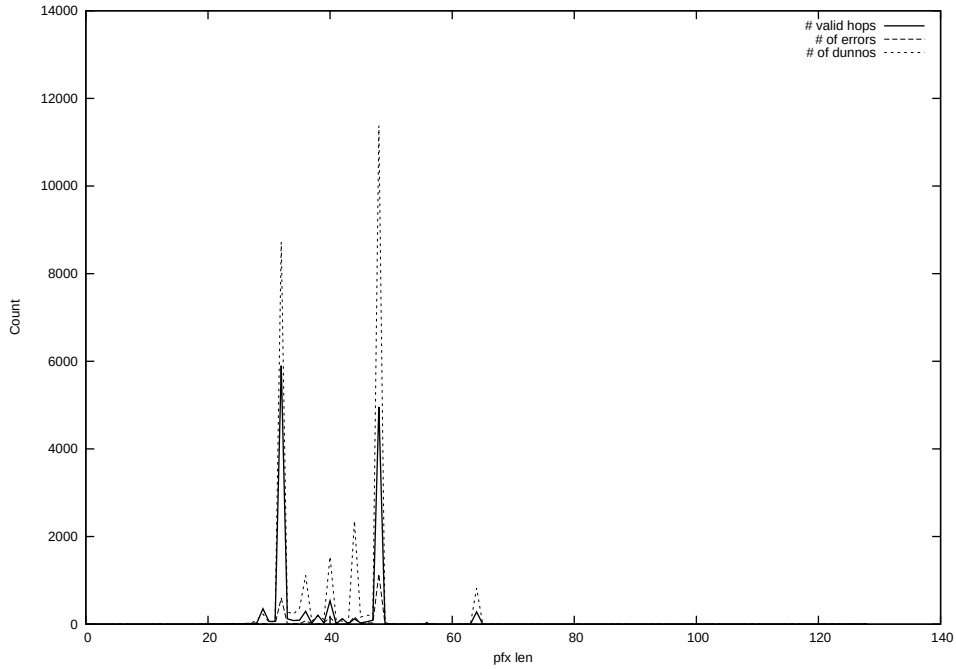
2013-08-07



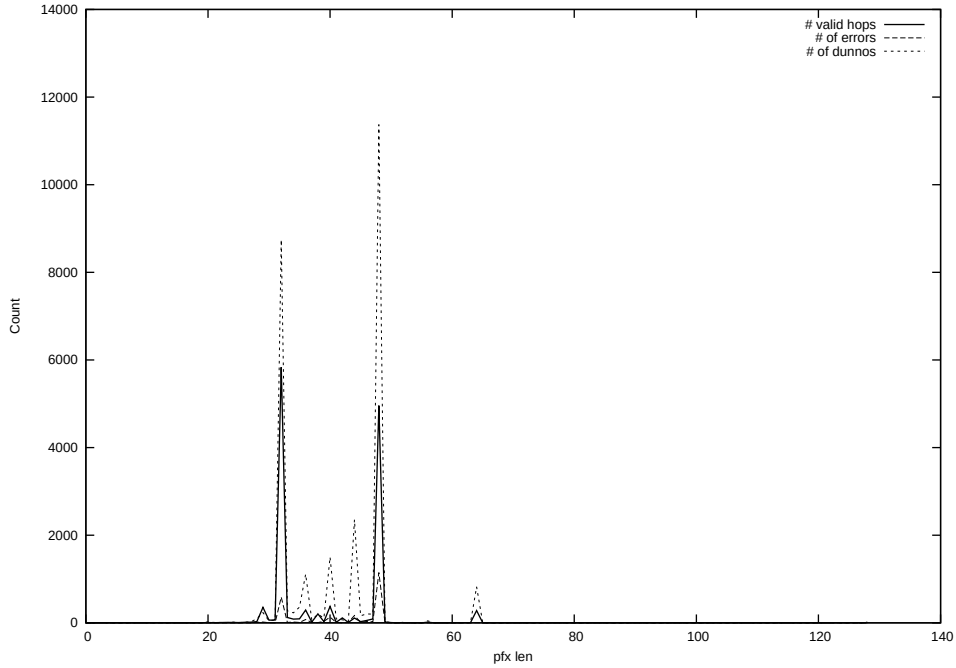
2013-08-08



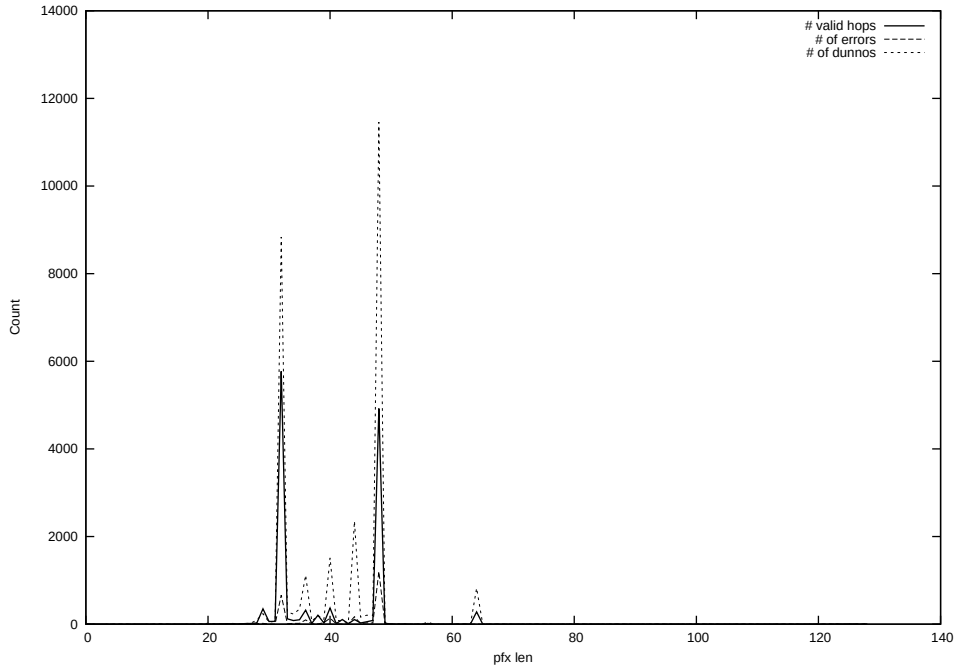
2013-08-09



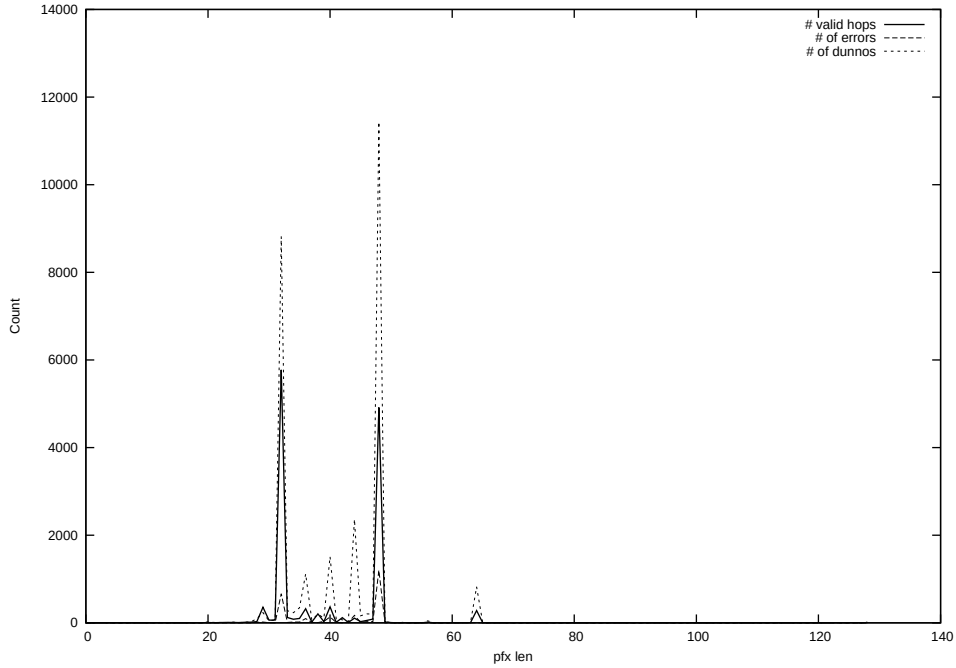
2013-08-10



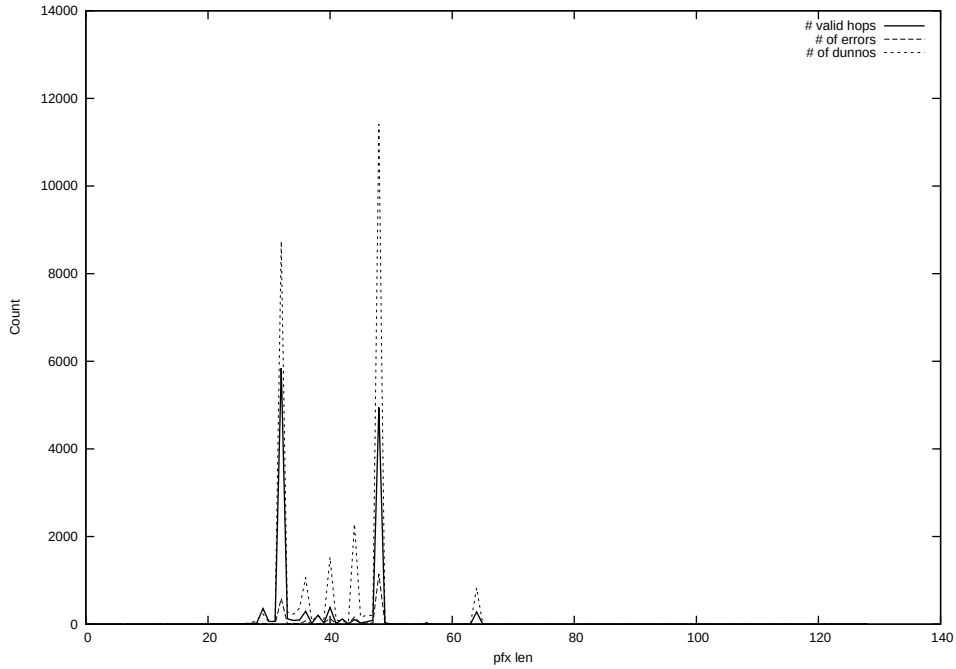
2013-08-11



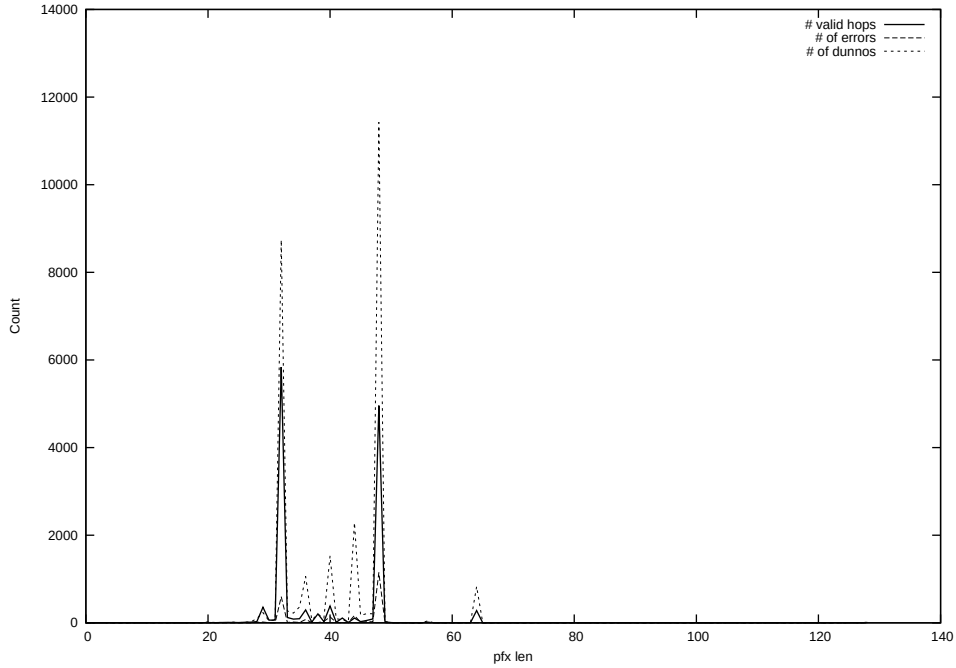
2013-08-12



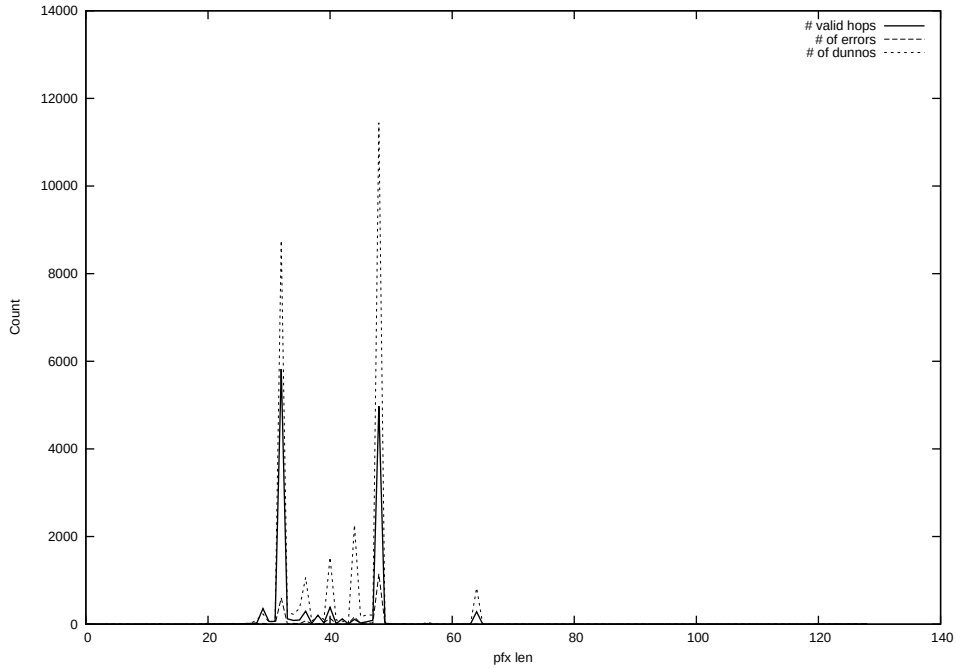
2013-08-13



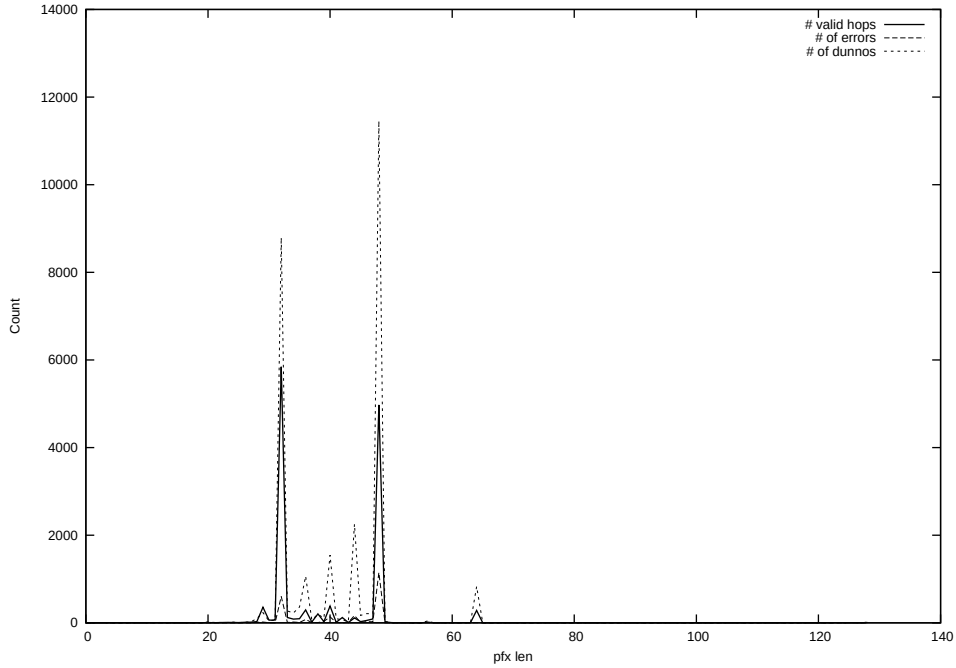
2013-08-15



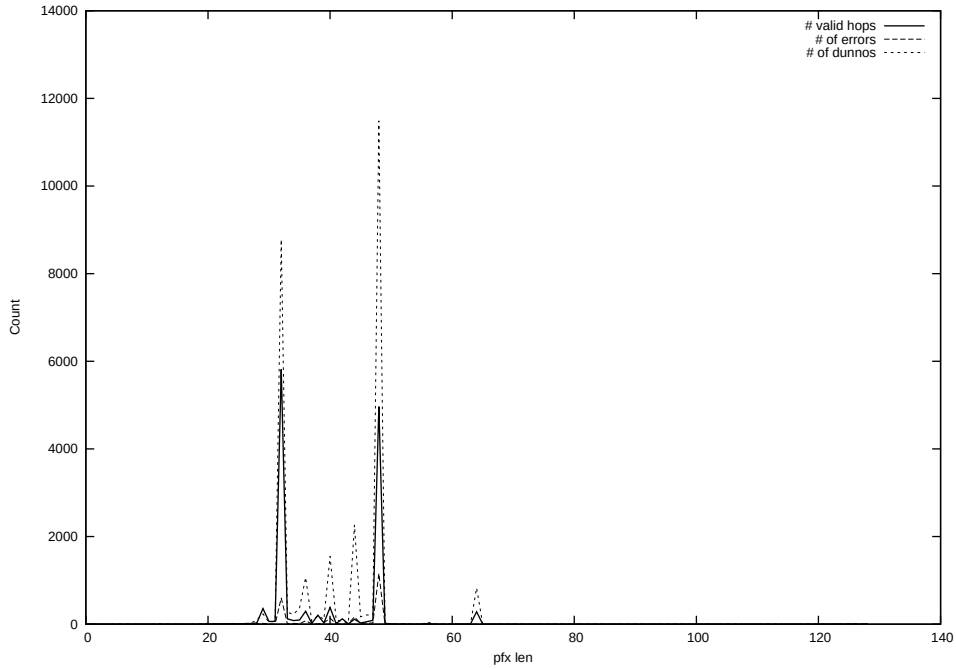
2013-08-16



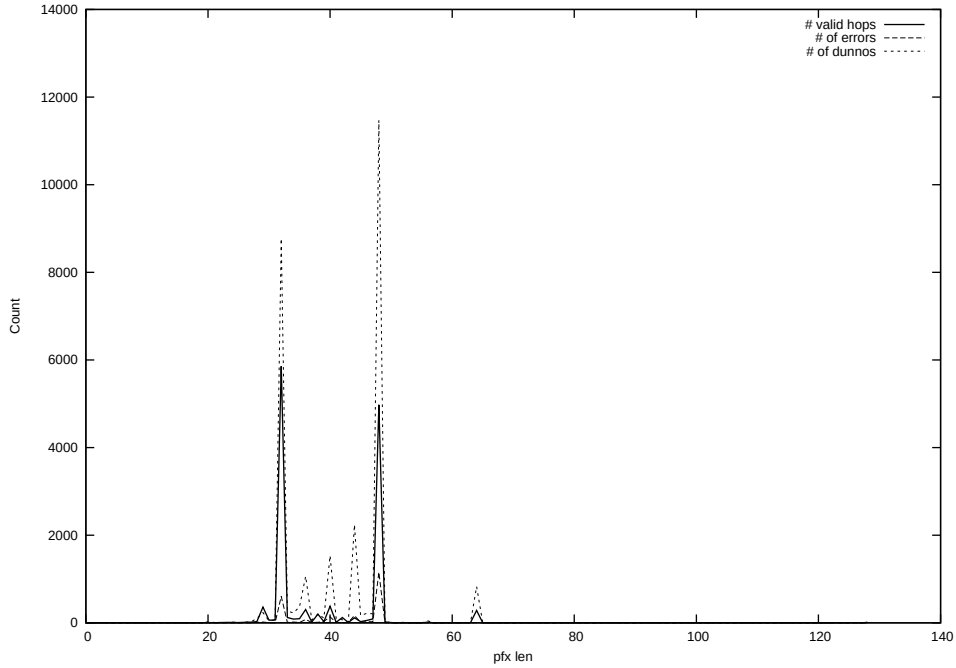
2013-08-17



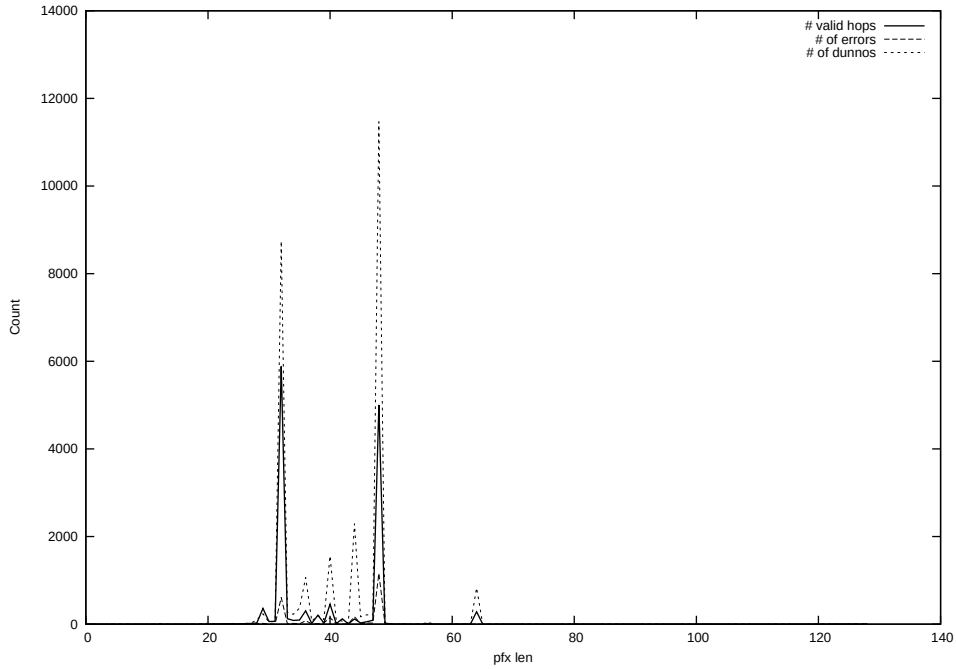
2013-08-18



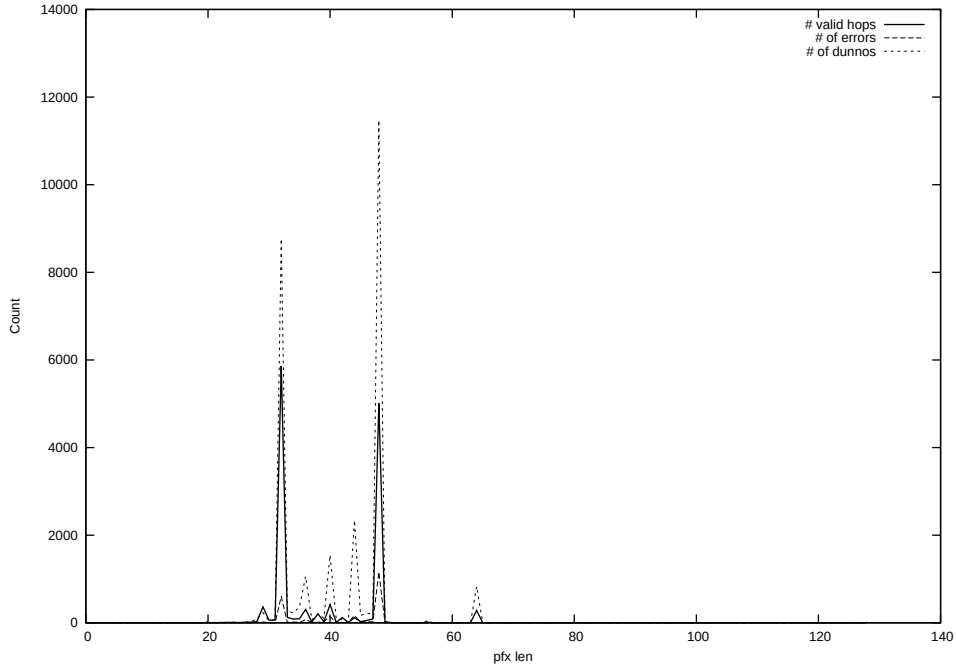
2013-08-19



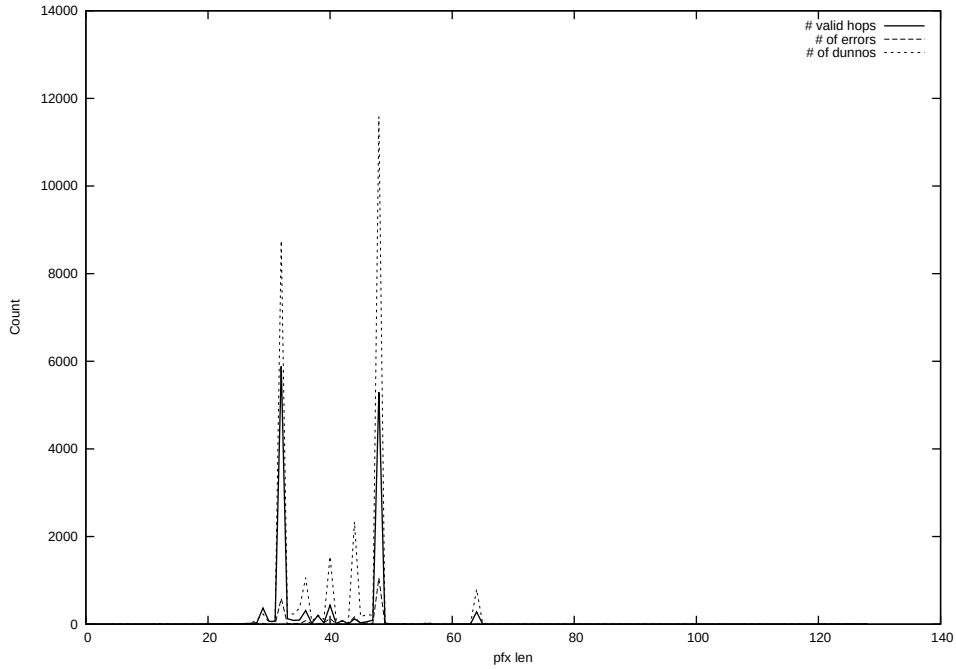
2013-08-20



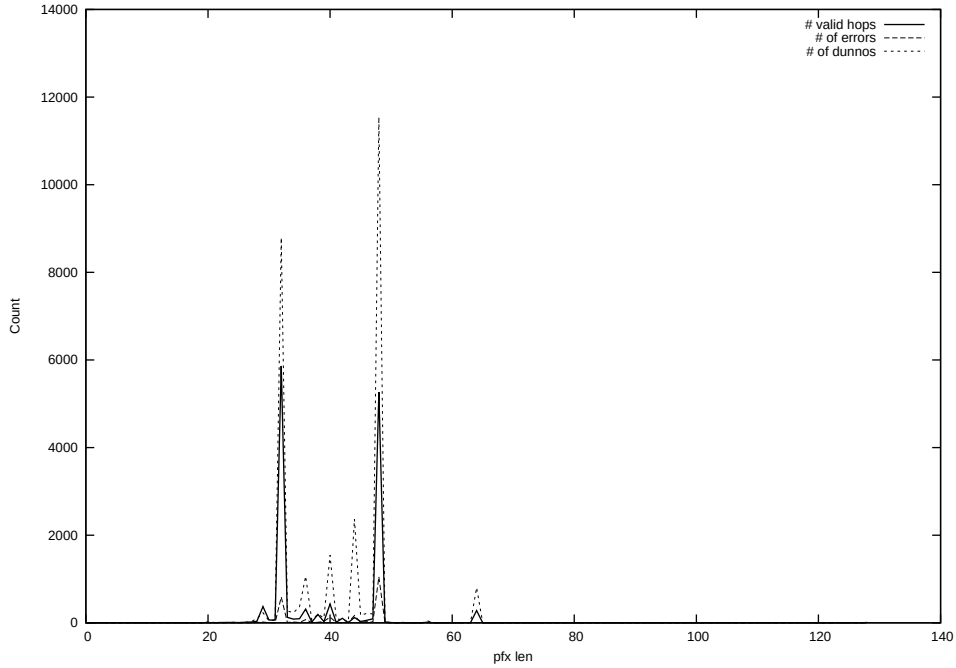
2013-08-21



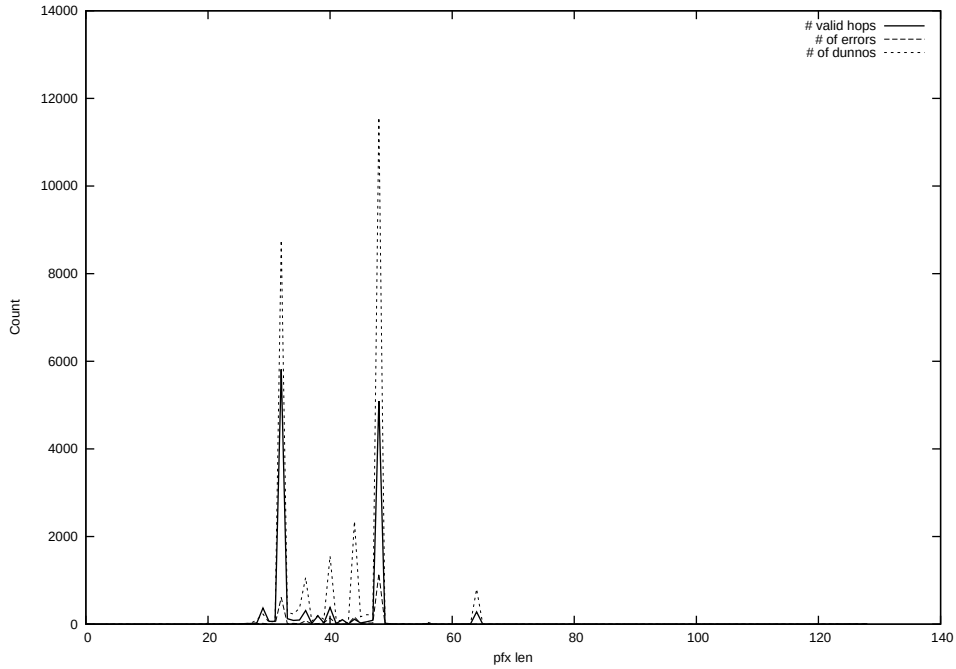
2013-08-22



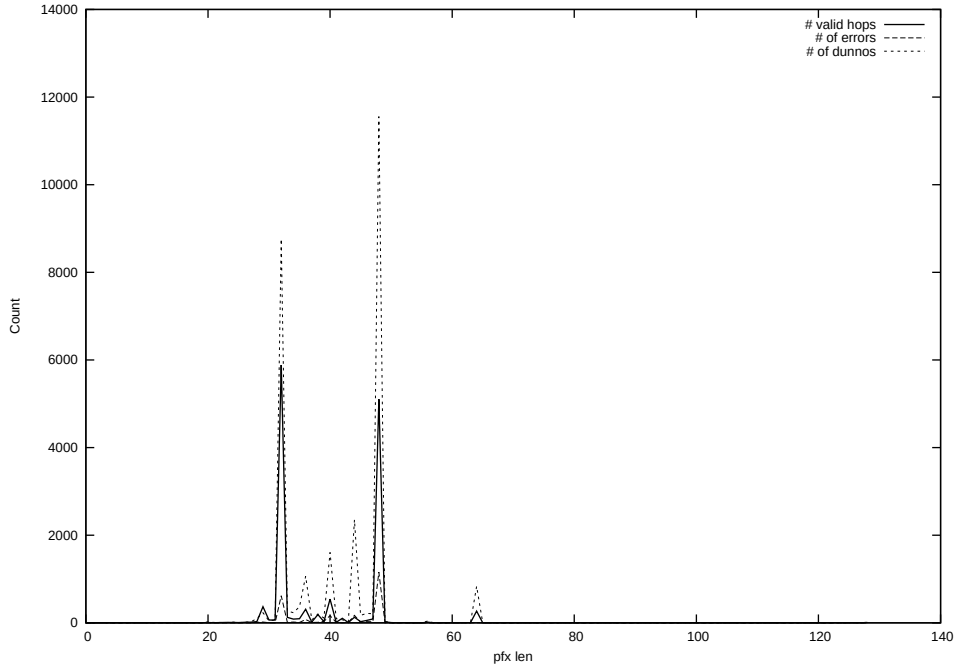
2013-08-23



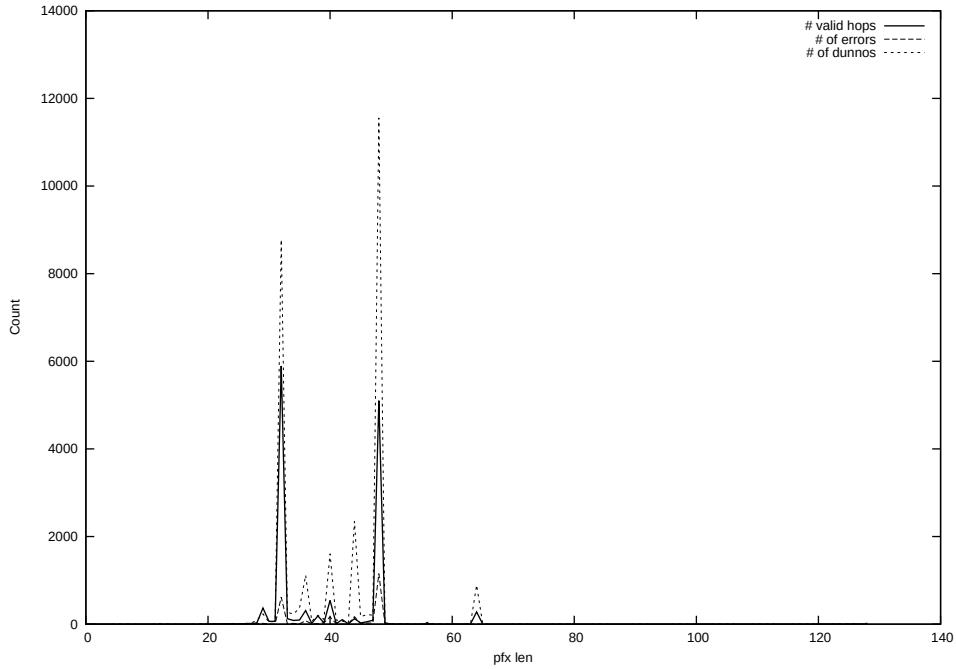
2013-08-24



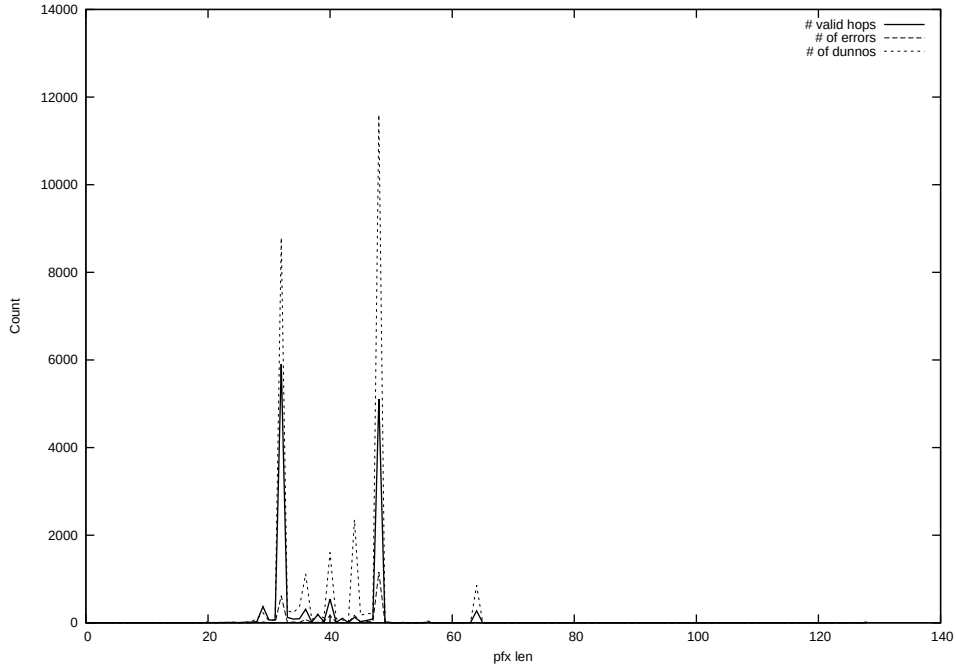
2013-08-25



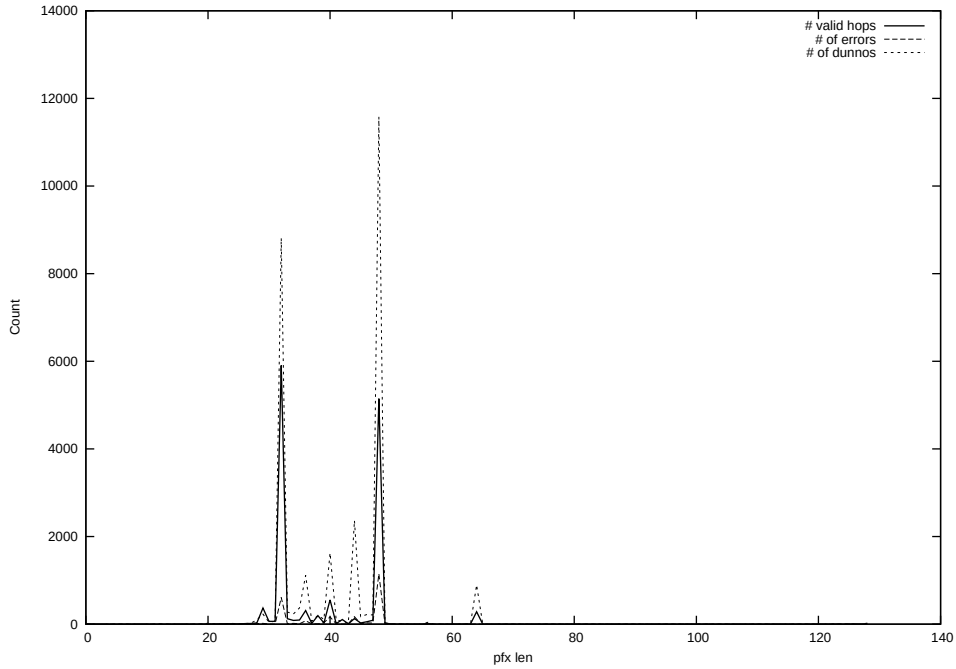
2013-08-26



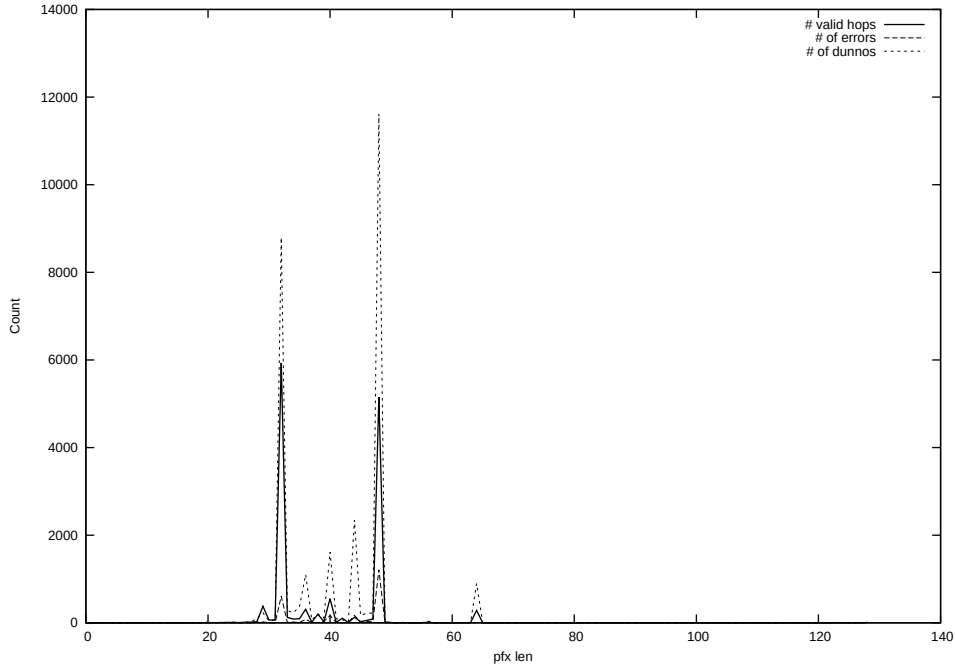
2013-08-27



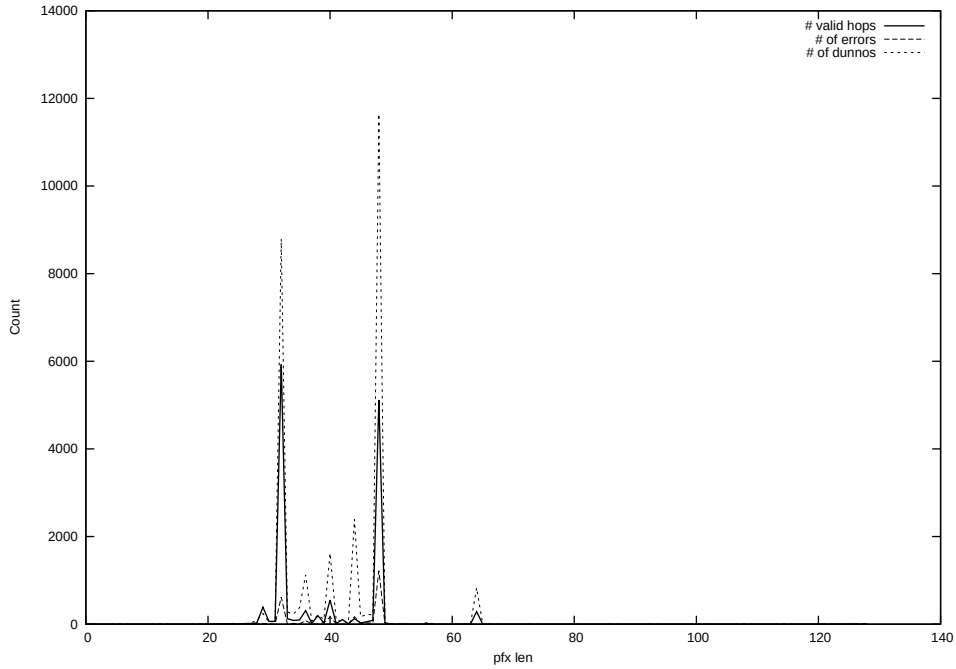
2013-08-28



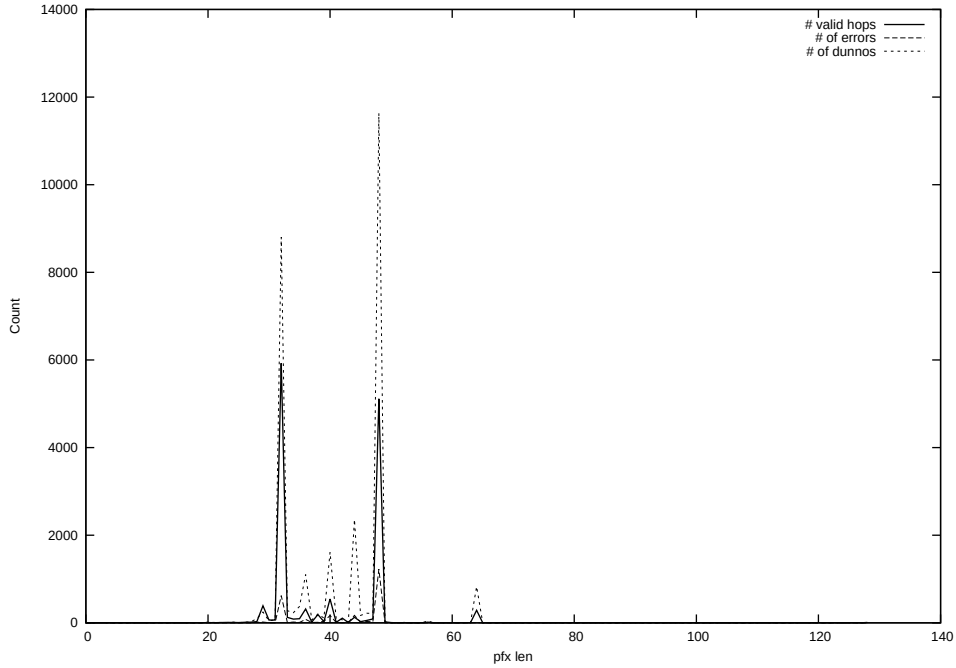
2013-08-29



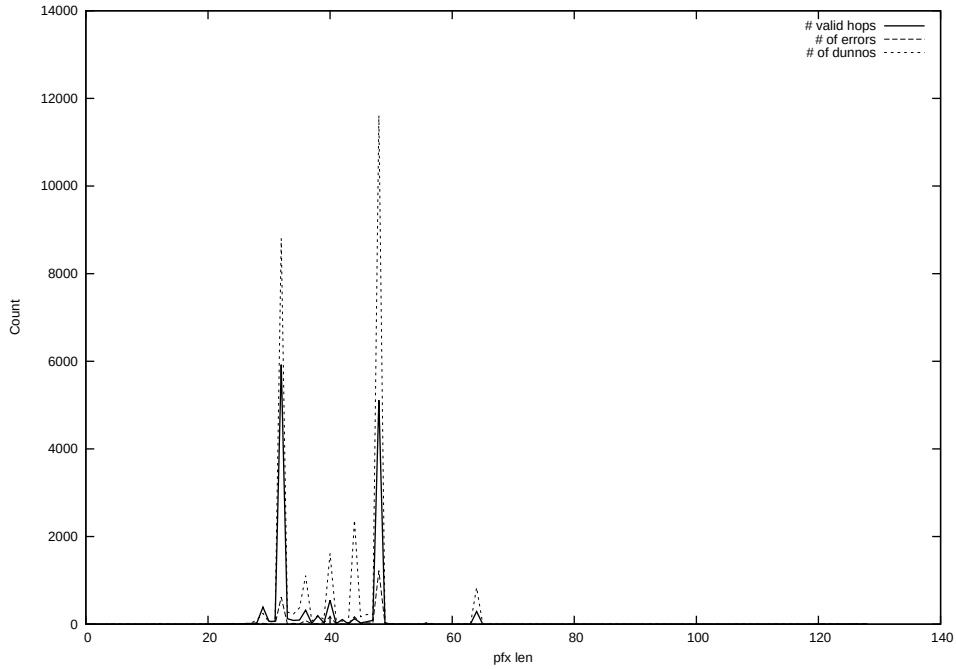
2013-08-30



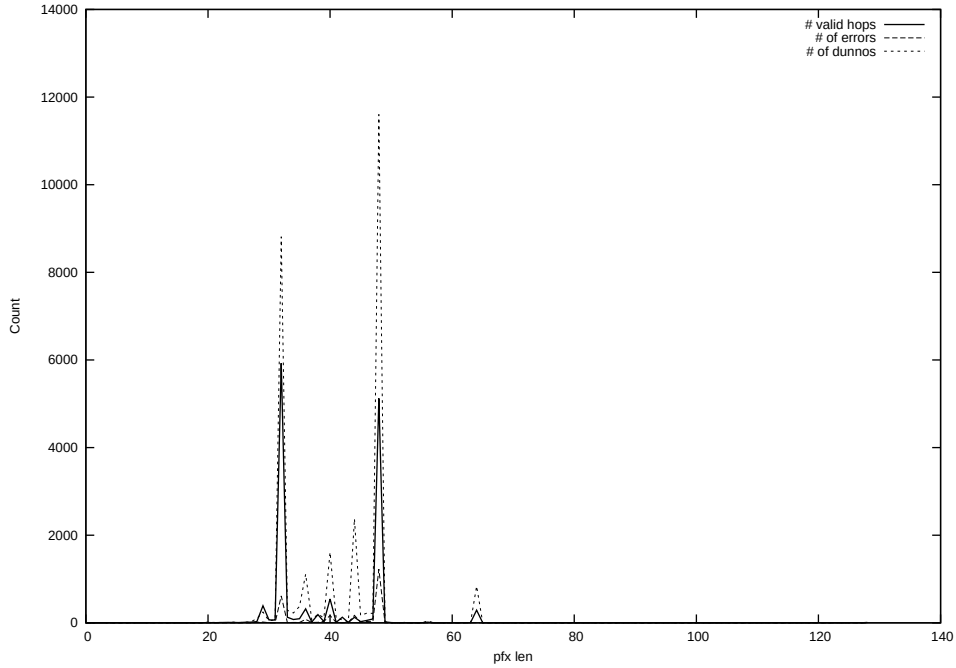
2013-08-31



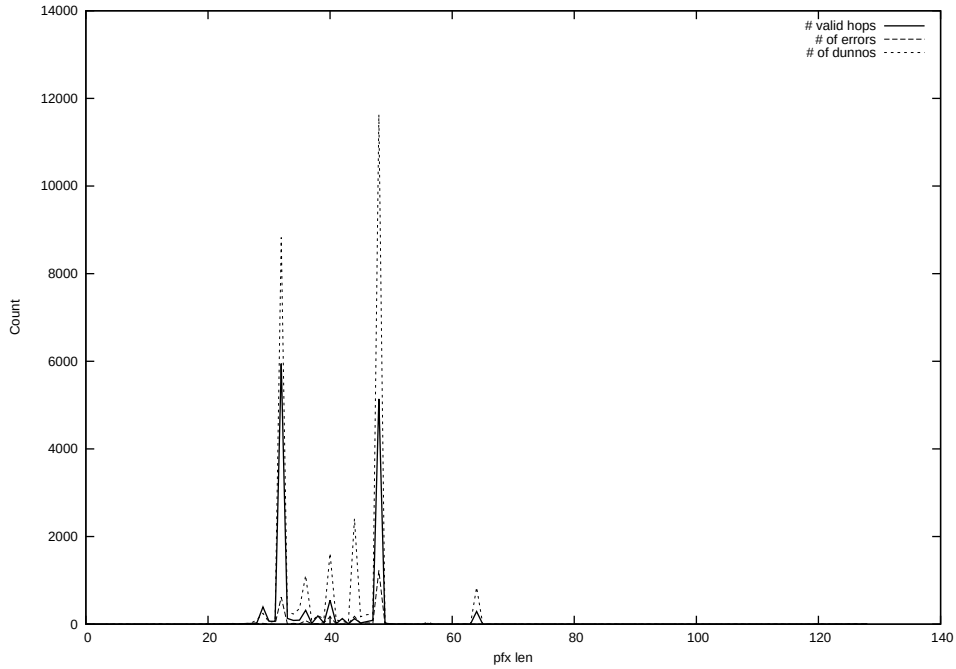
2013-09-01



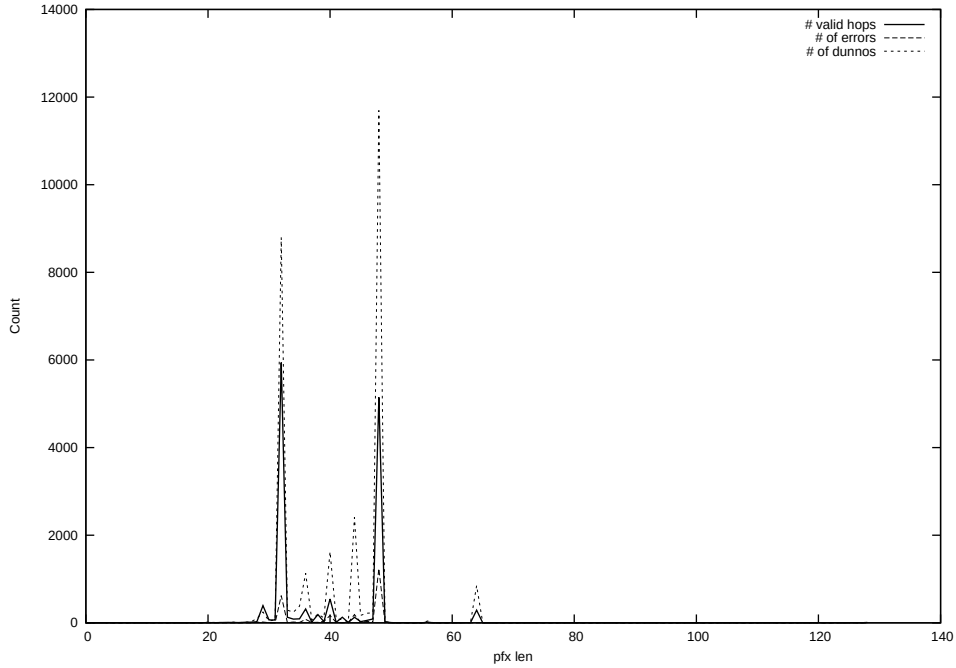
2013-09-02



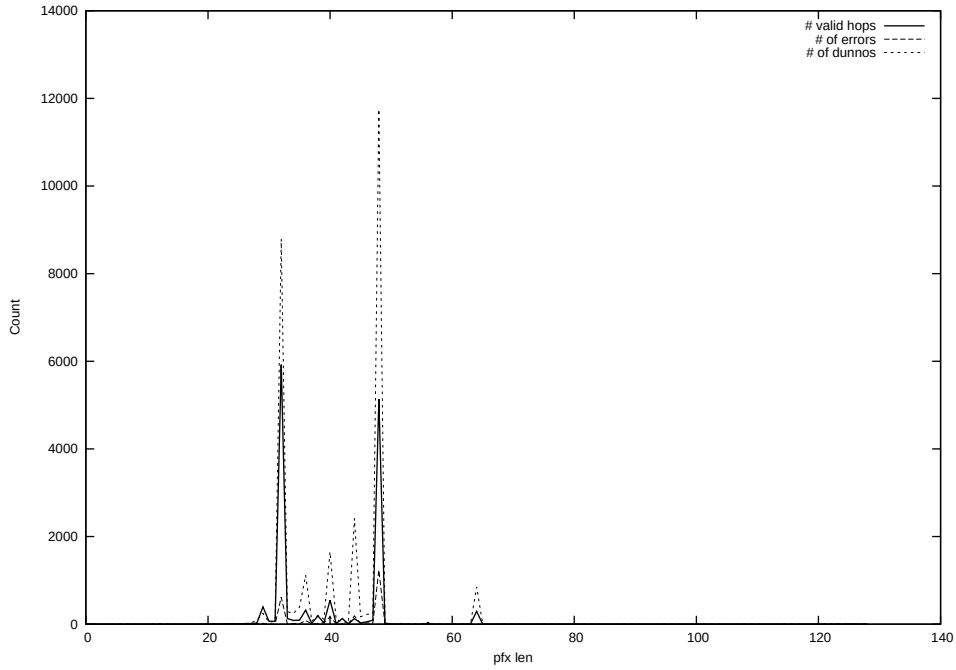
2013-09-03



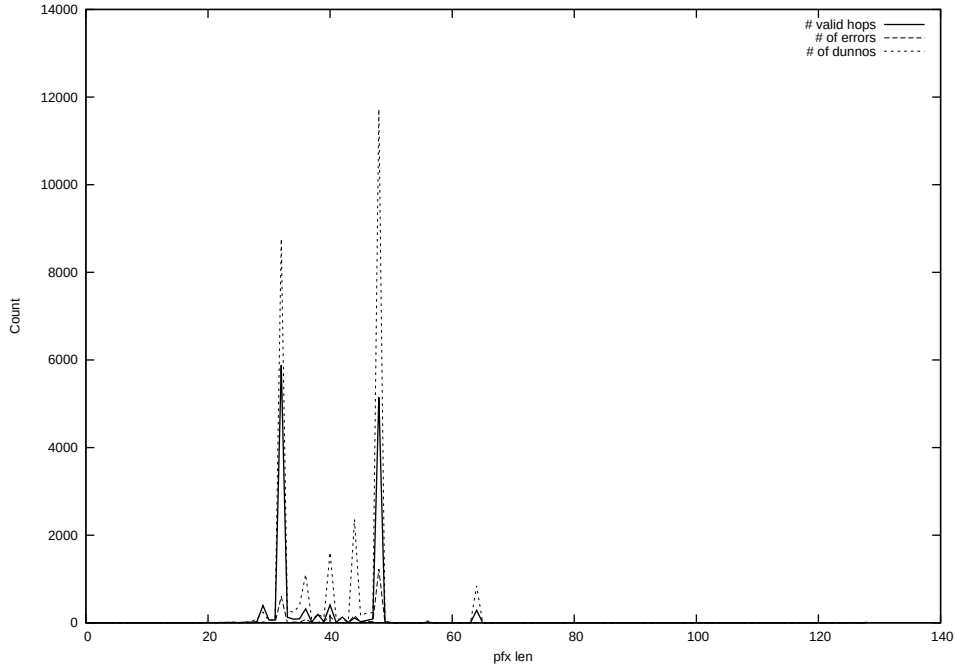
2013-09-04



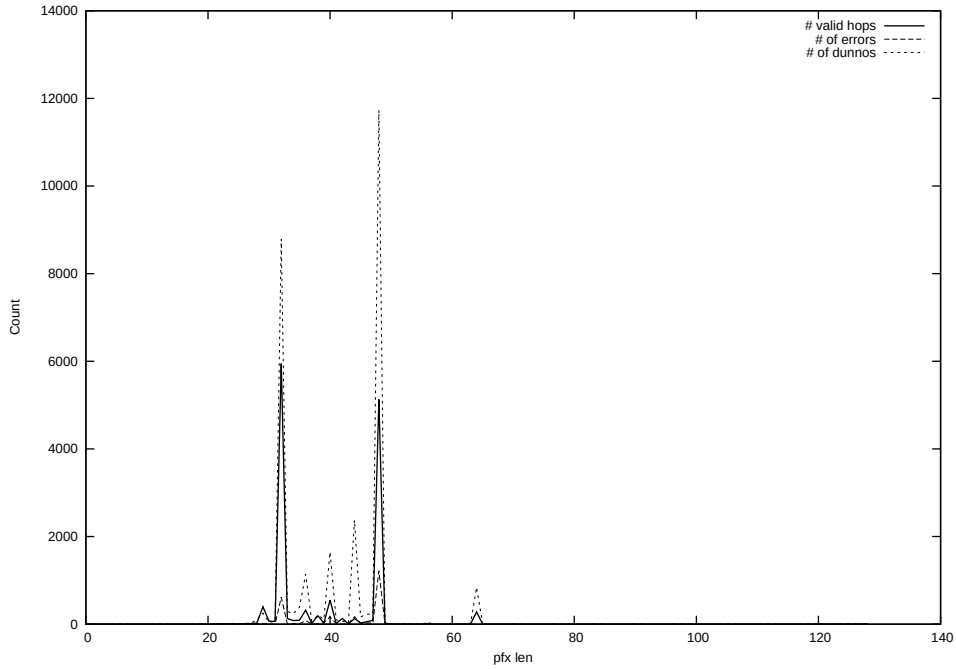
2013-09-05



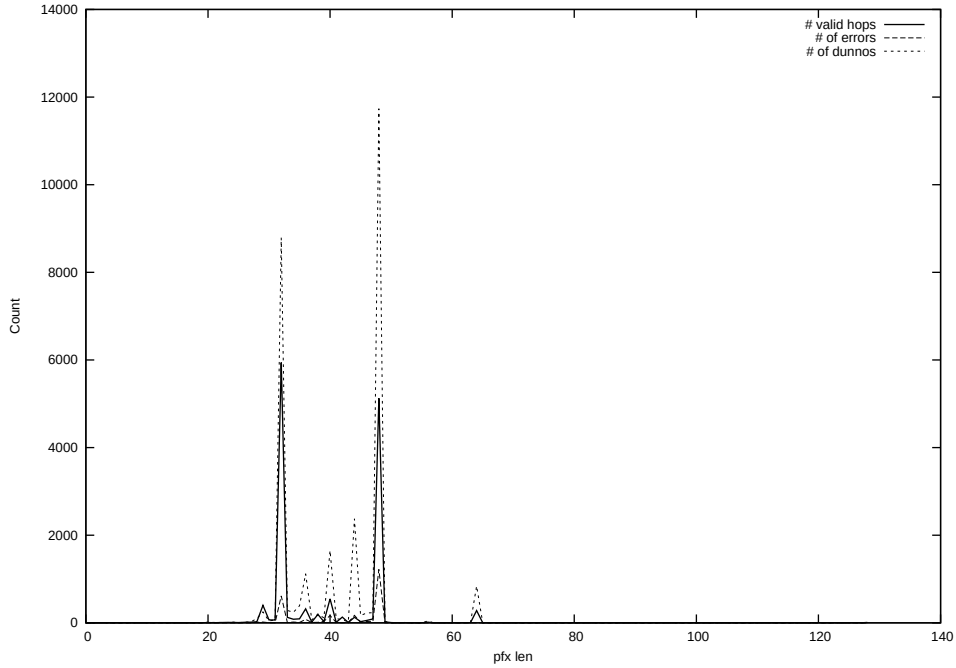
2013-09-06



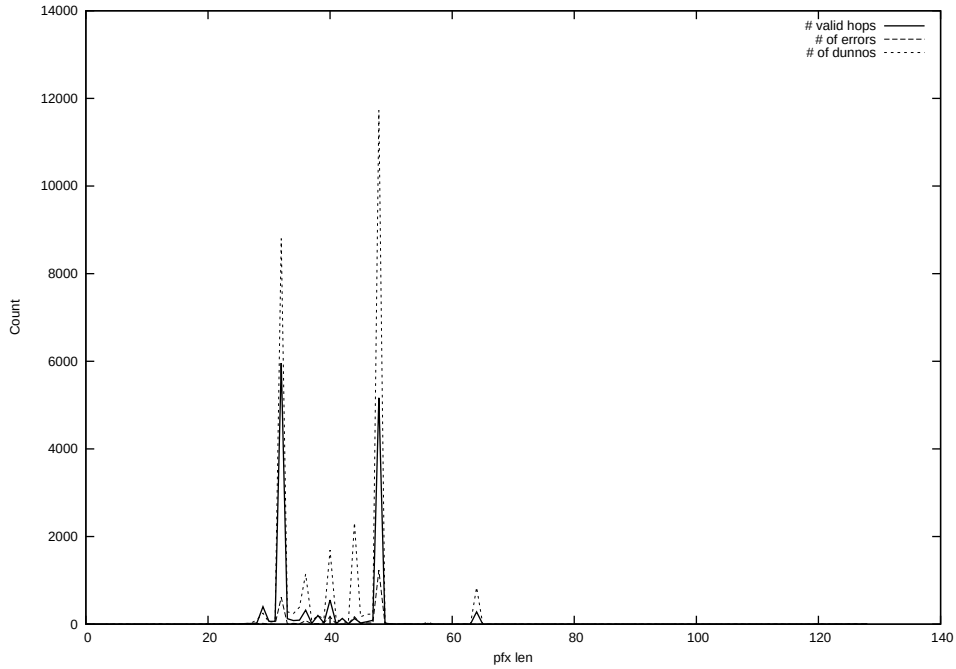
2013-09-07



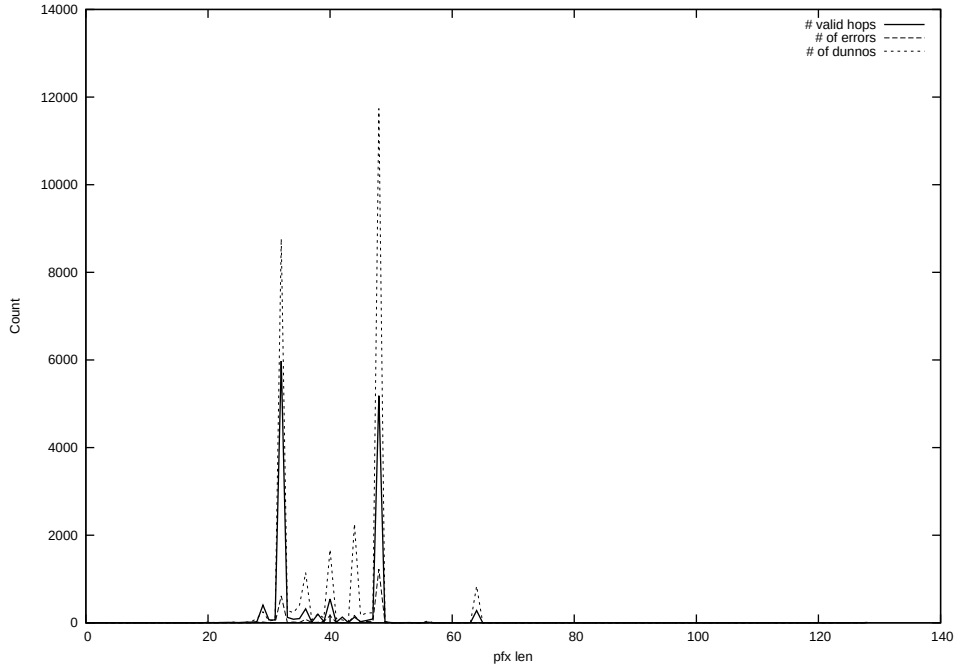
2013-09-08



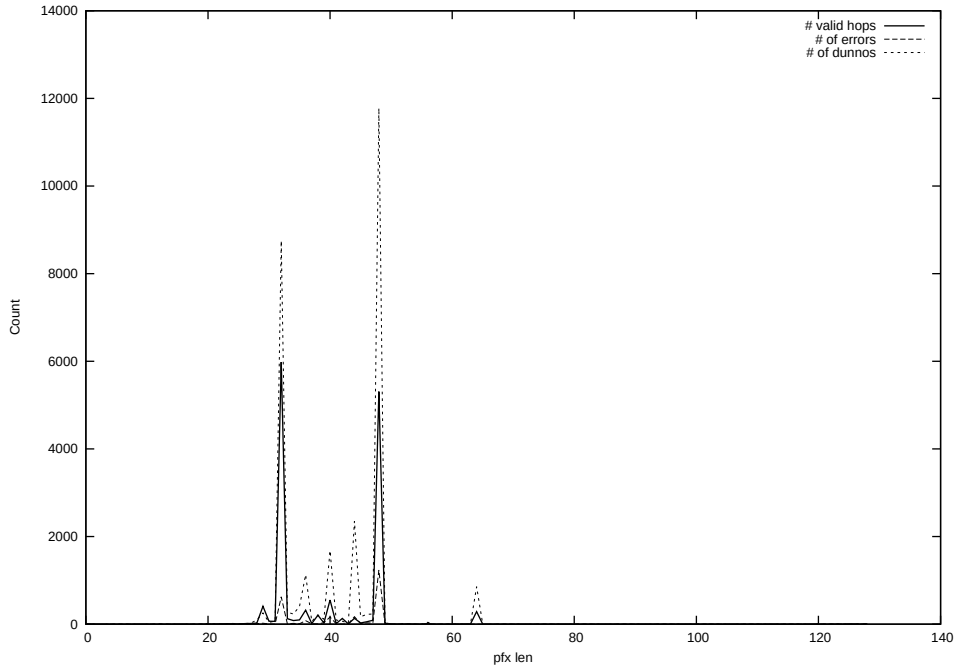
2013-09-09



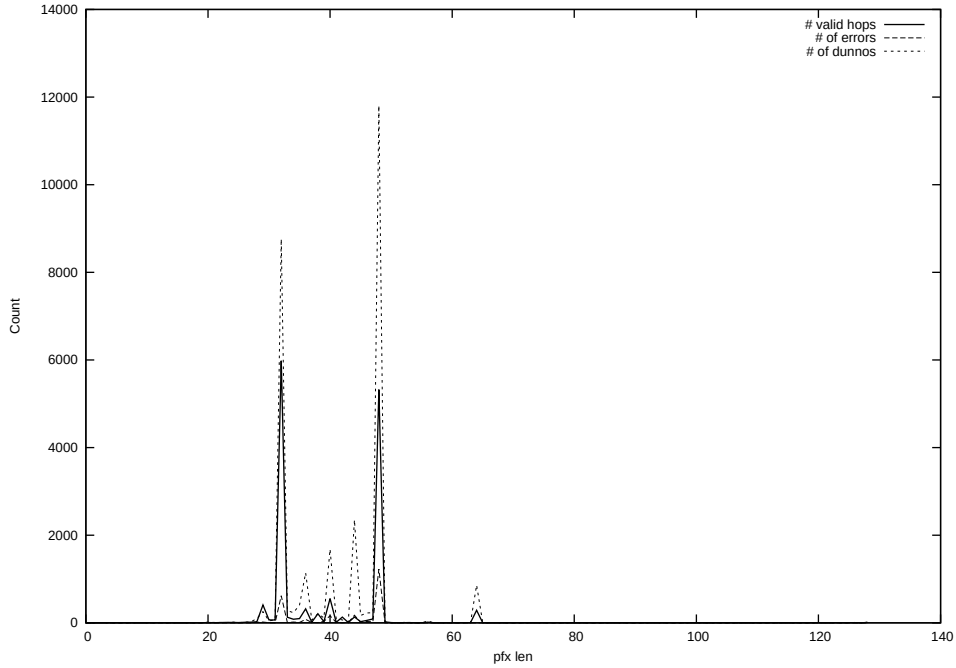
2013-09-10



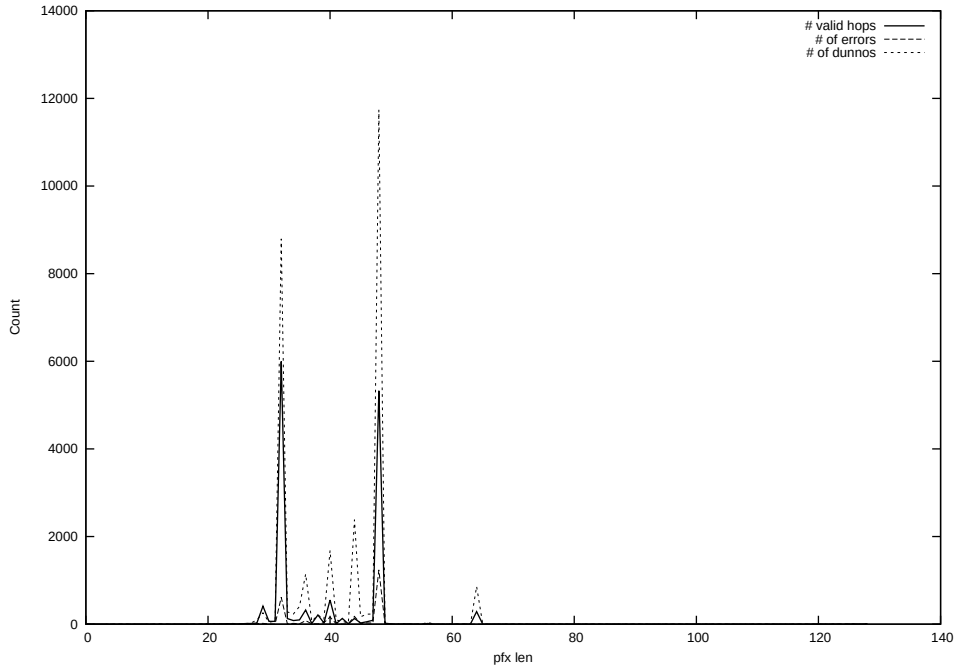
2013-09-11



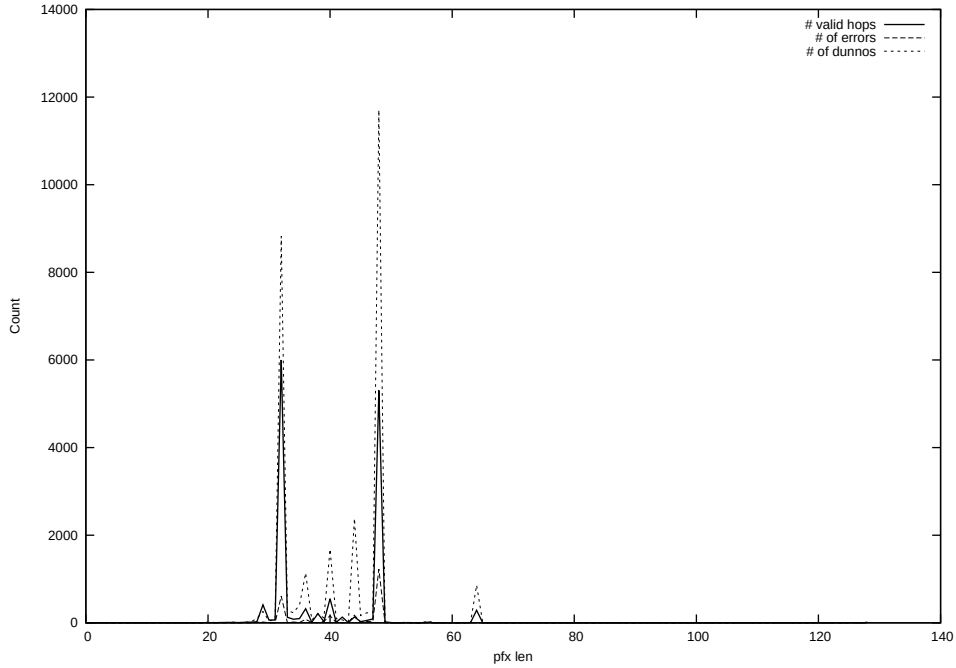
2013-09-12



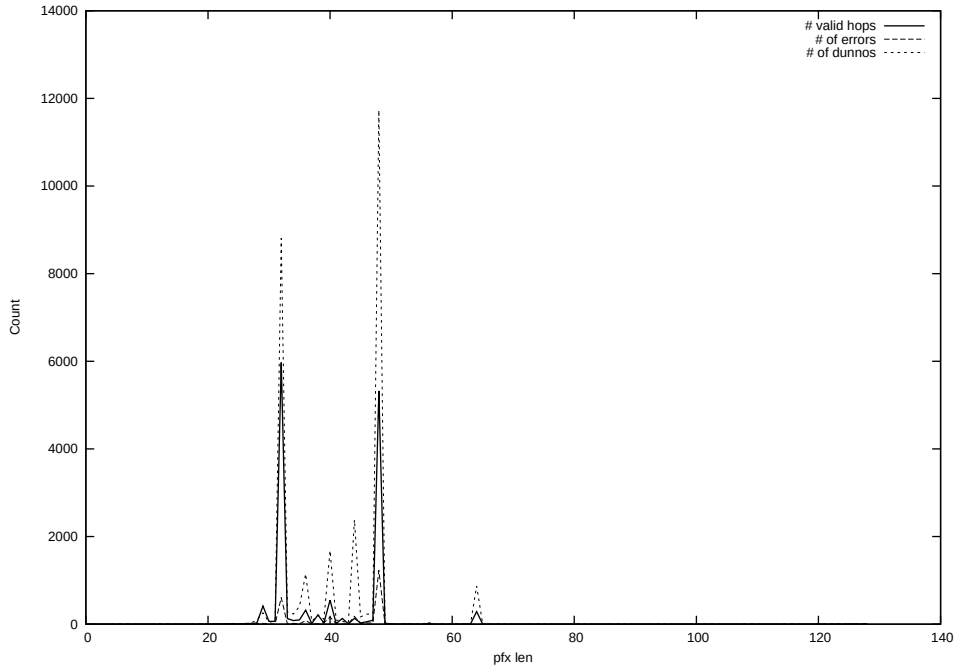
2013-09-13



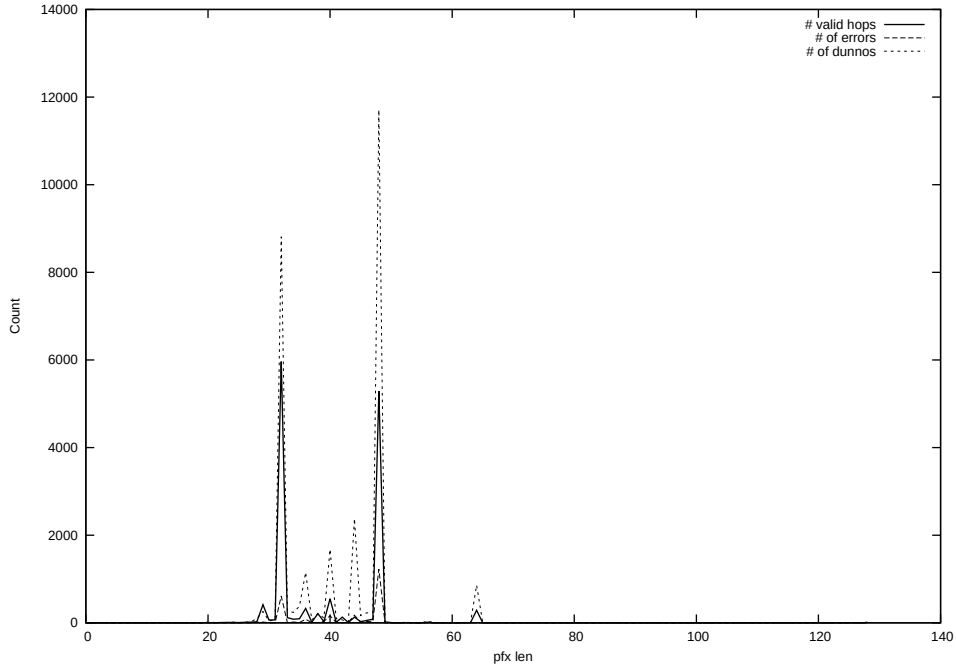
2013-09-14



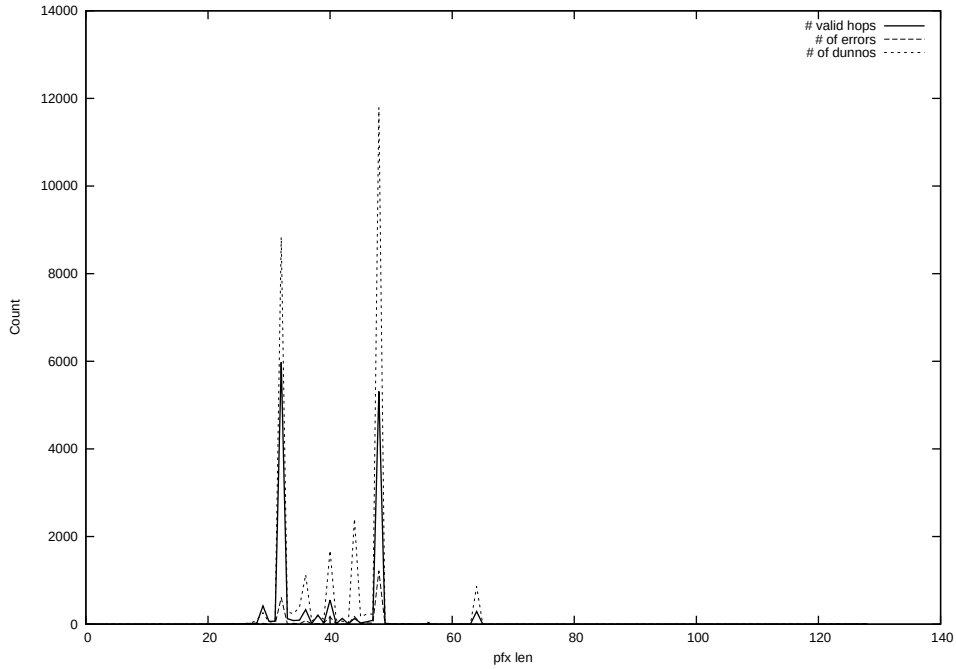
2013-09-15



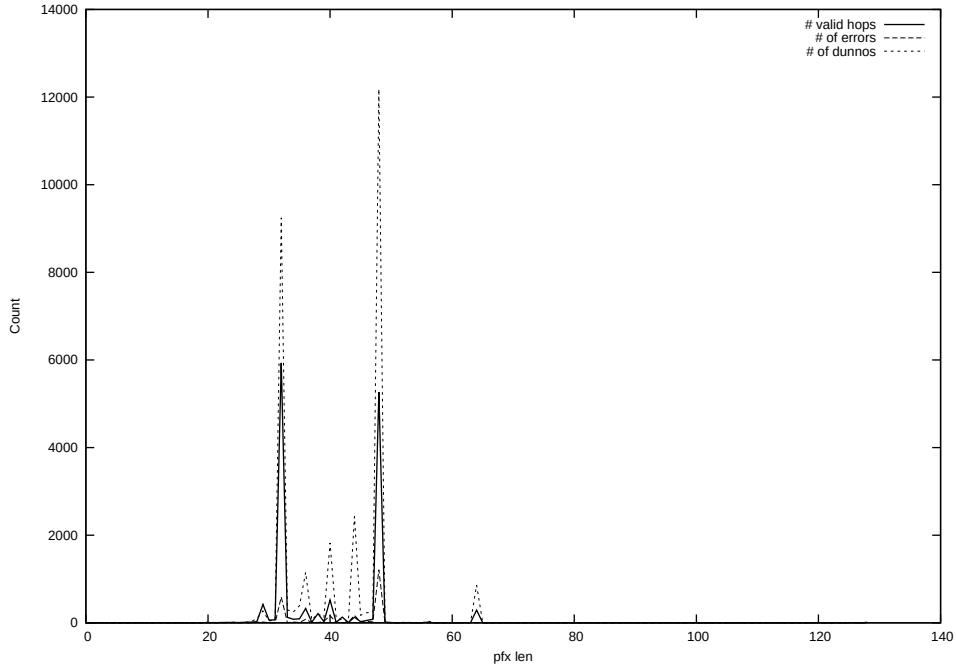
2013-09-16



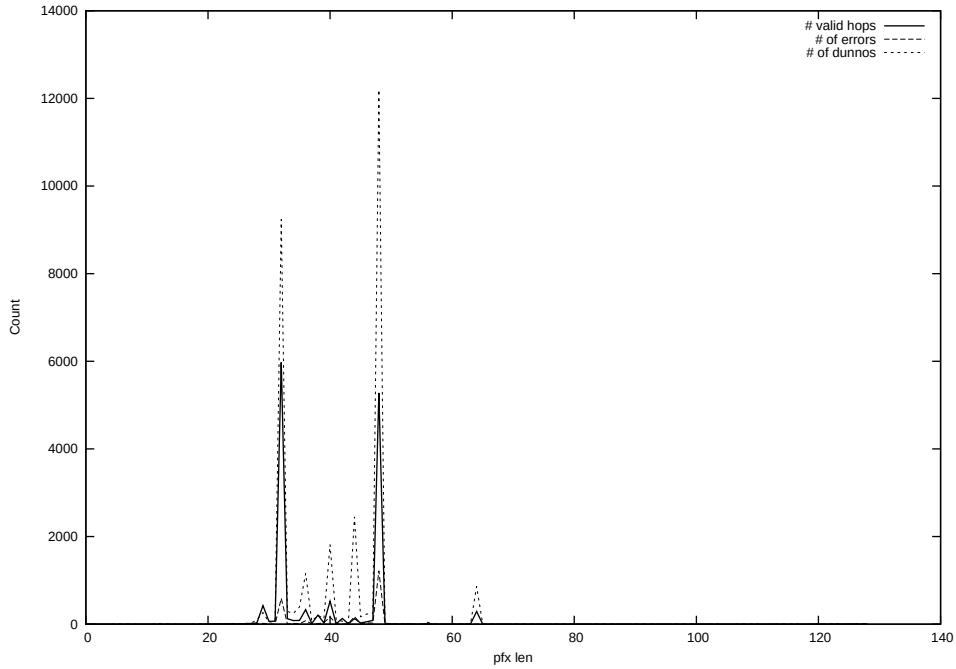
2013-09-17



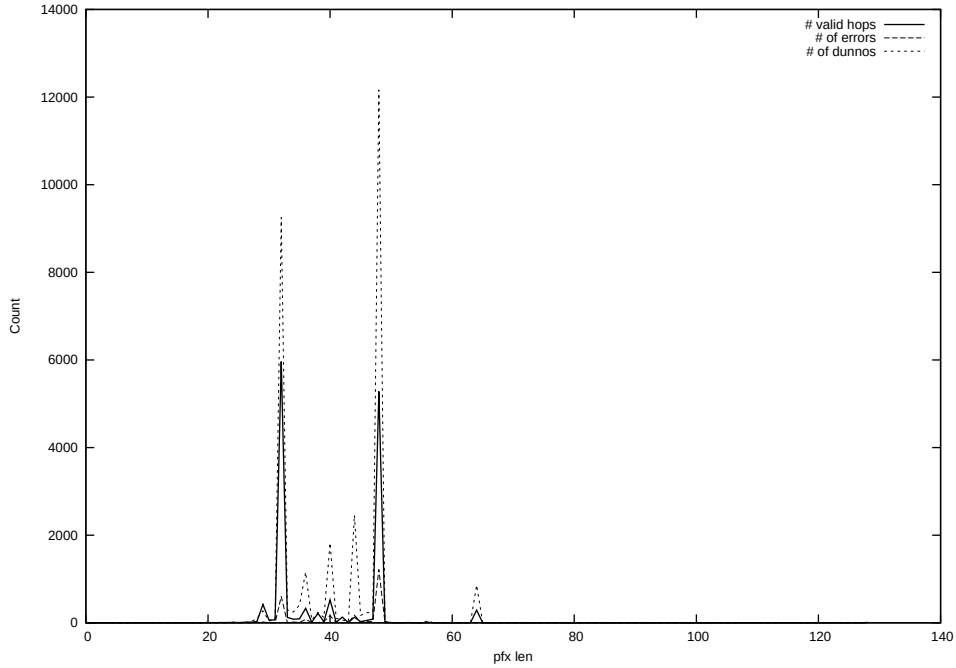
2013-09-18



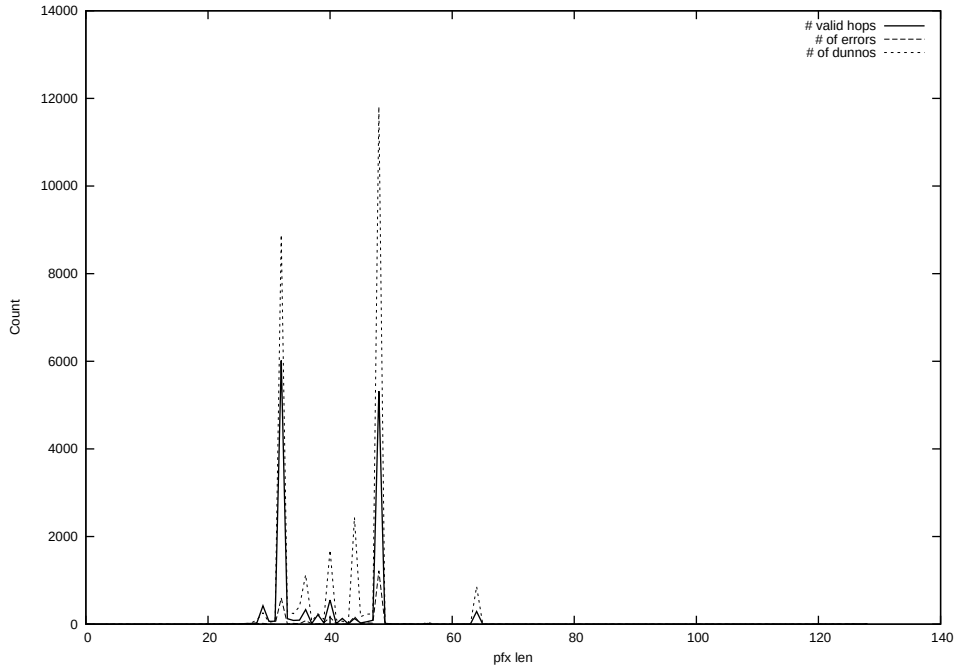
2013-09-19



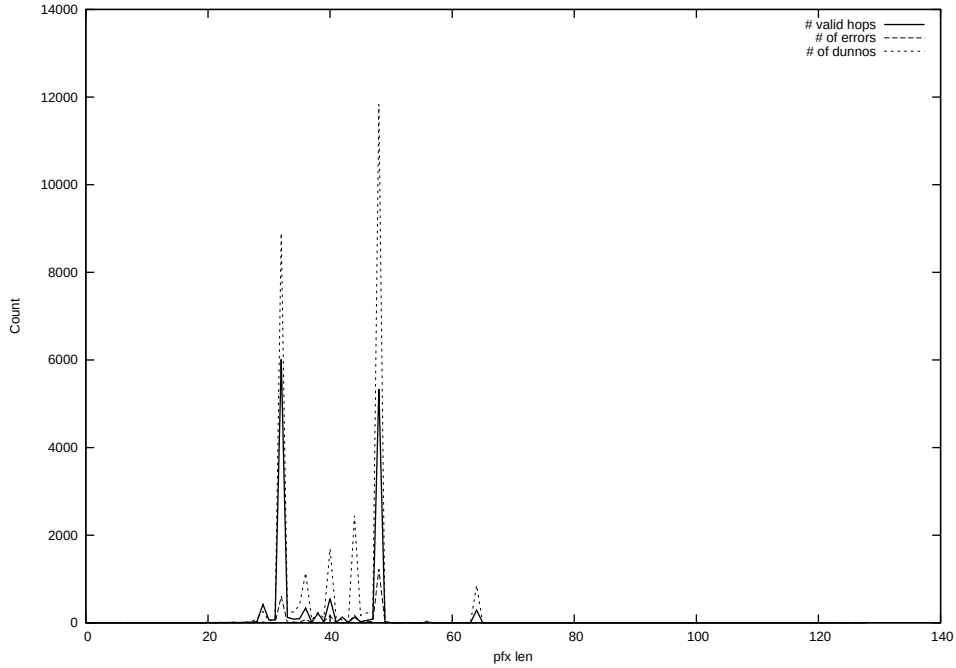
2013-09-20



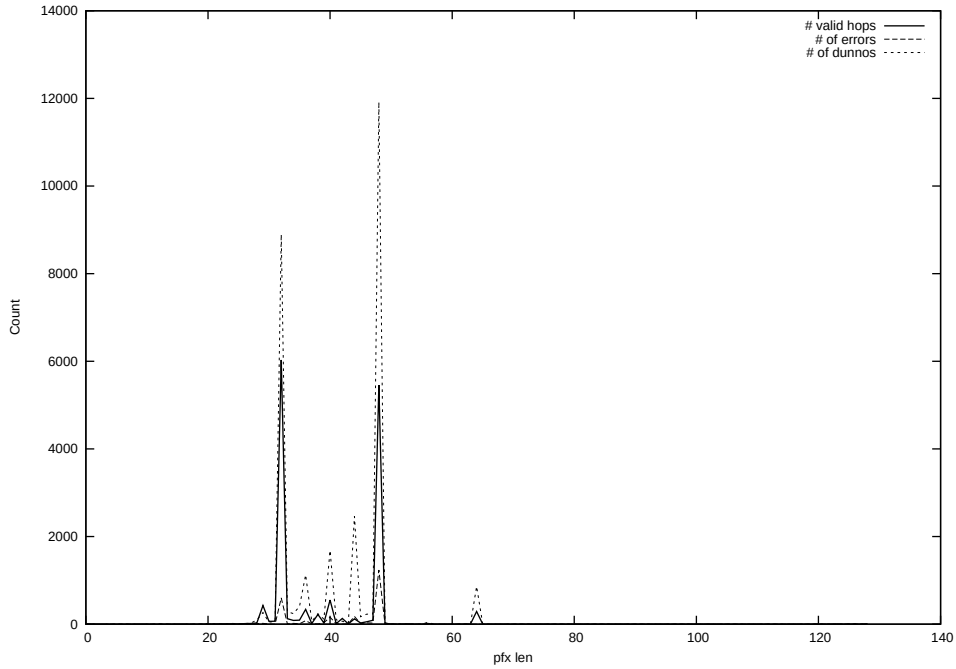
2013-09-21



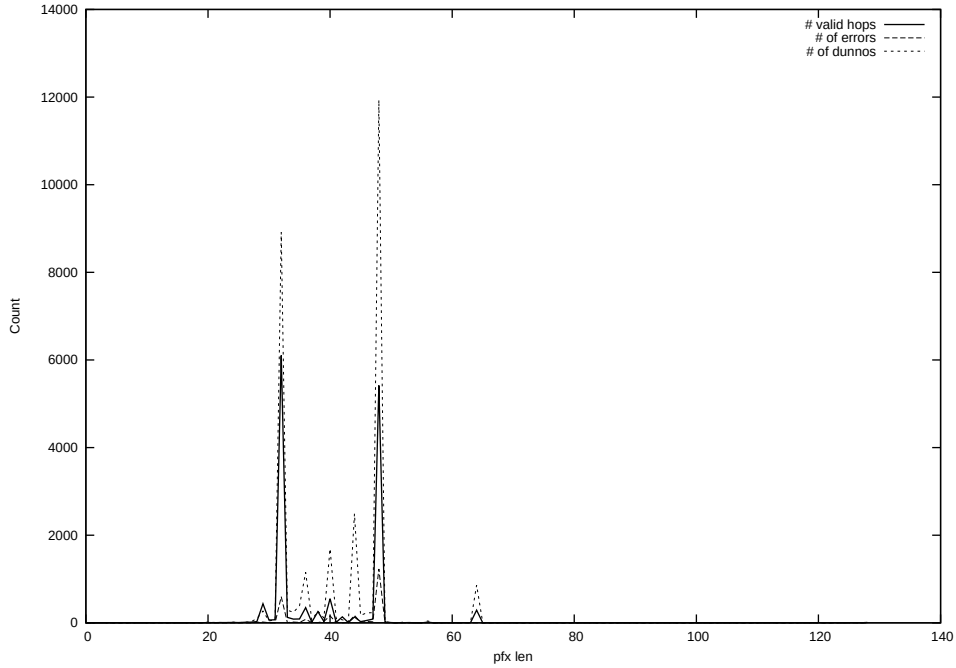
2013-09-22



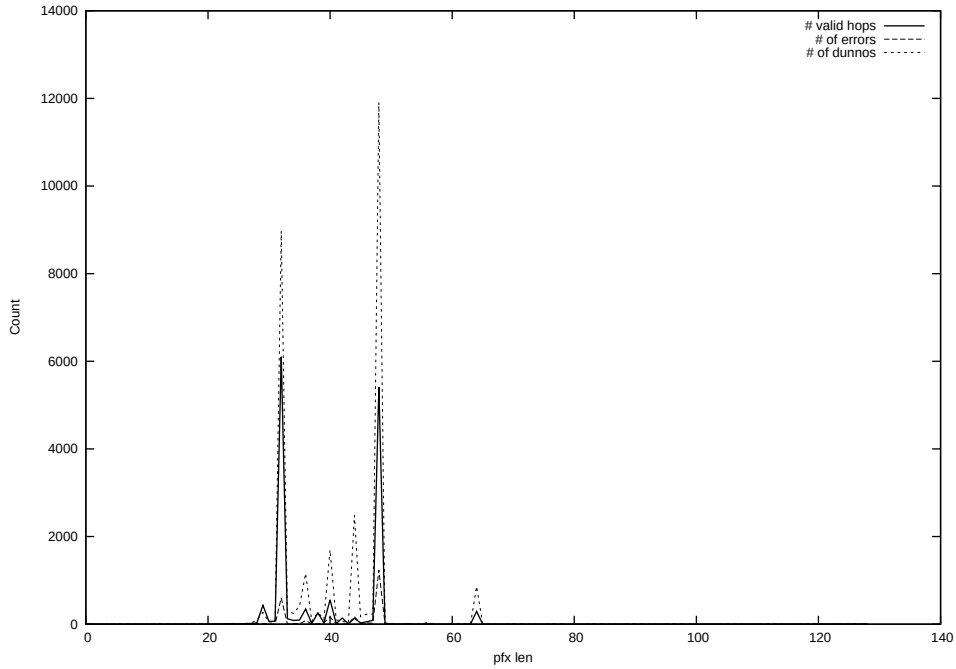
2013-09-23



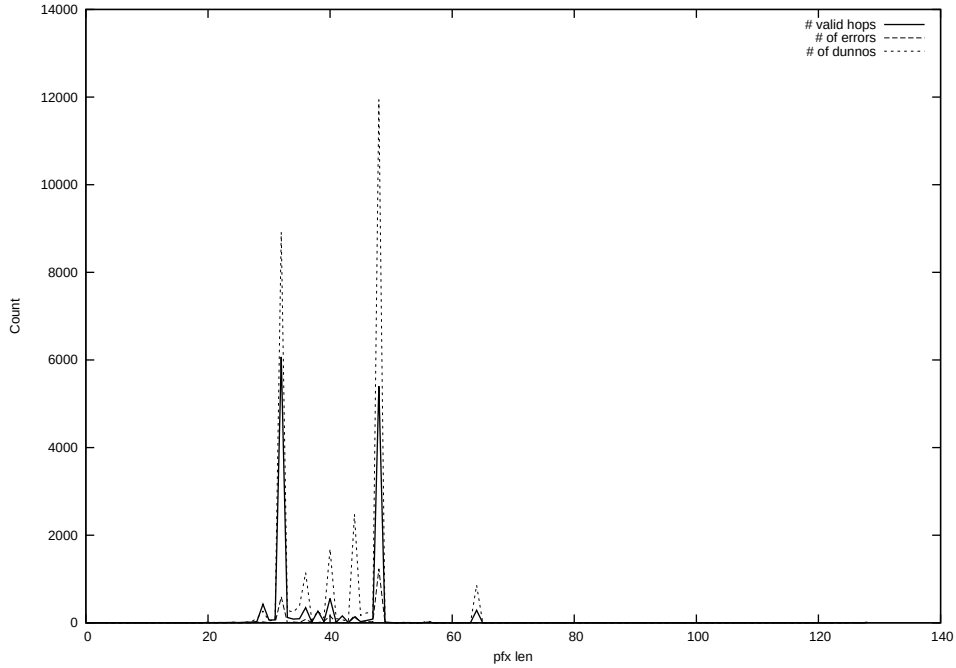
2013-09-24



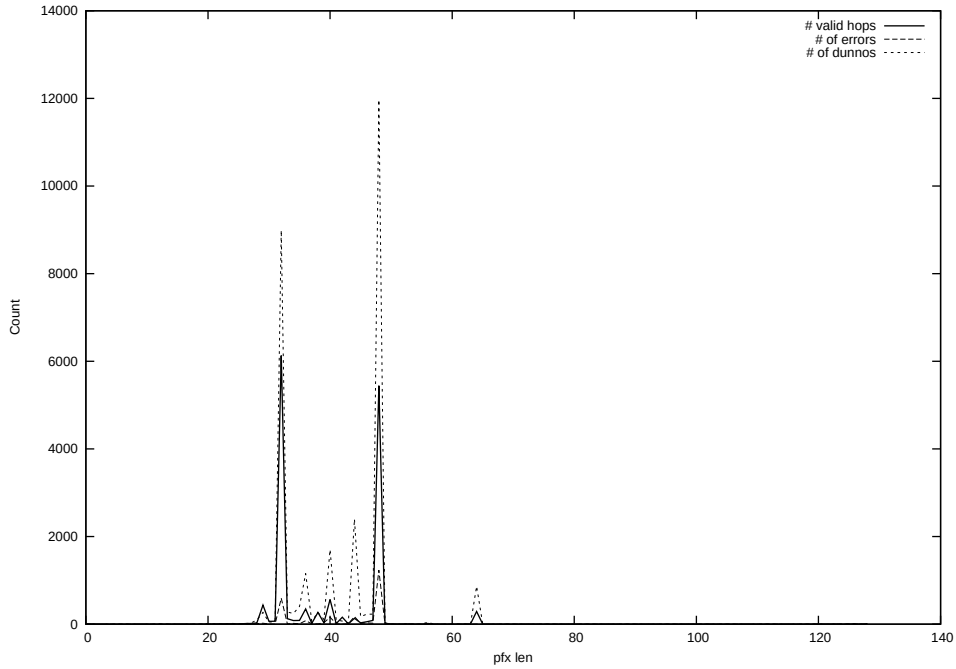
2013-09-25



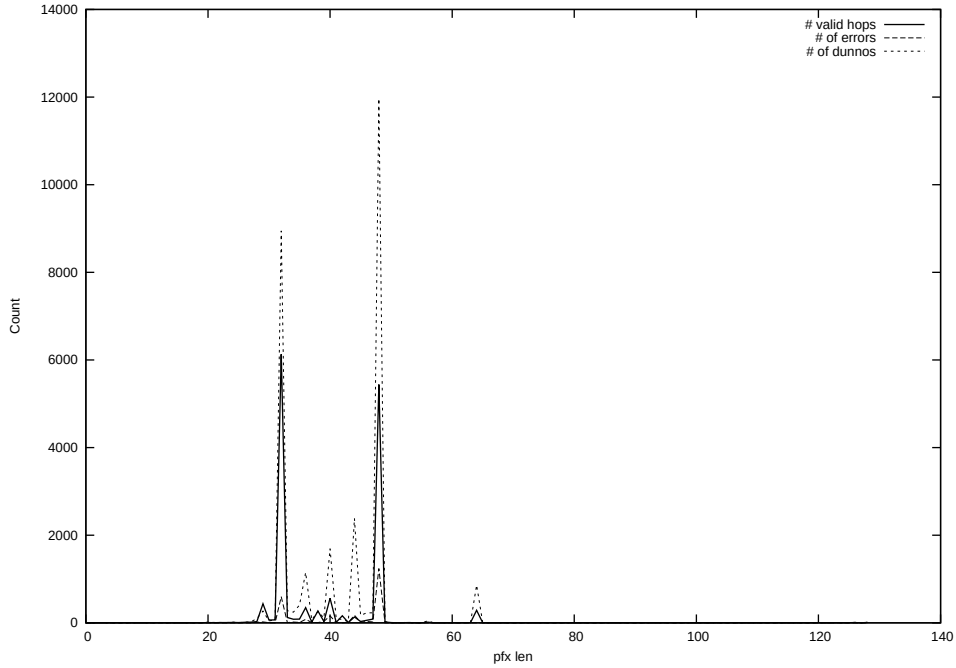
2013-09-26



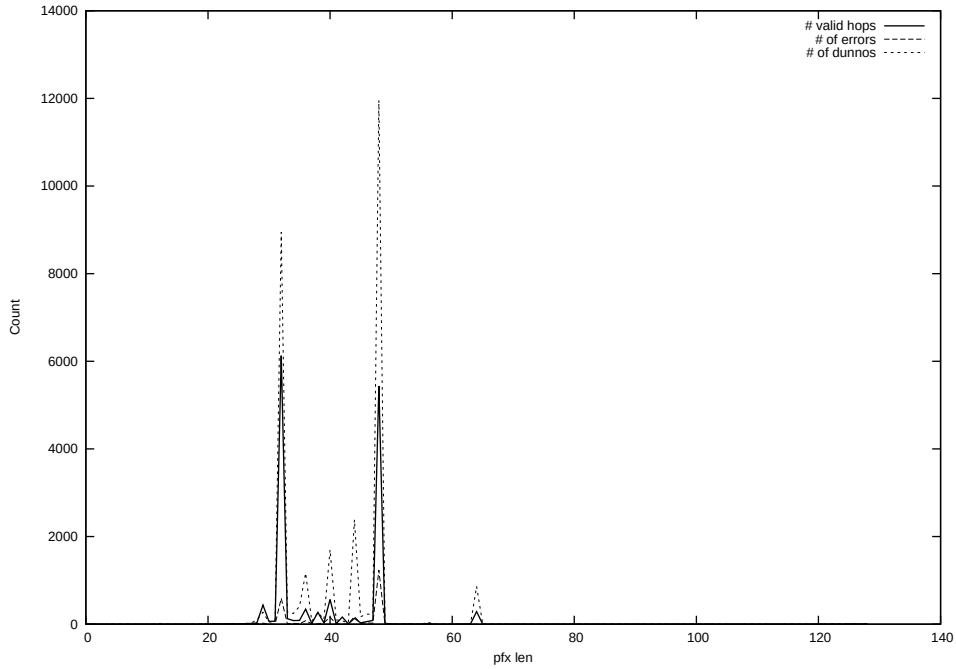
2013-09-27



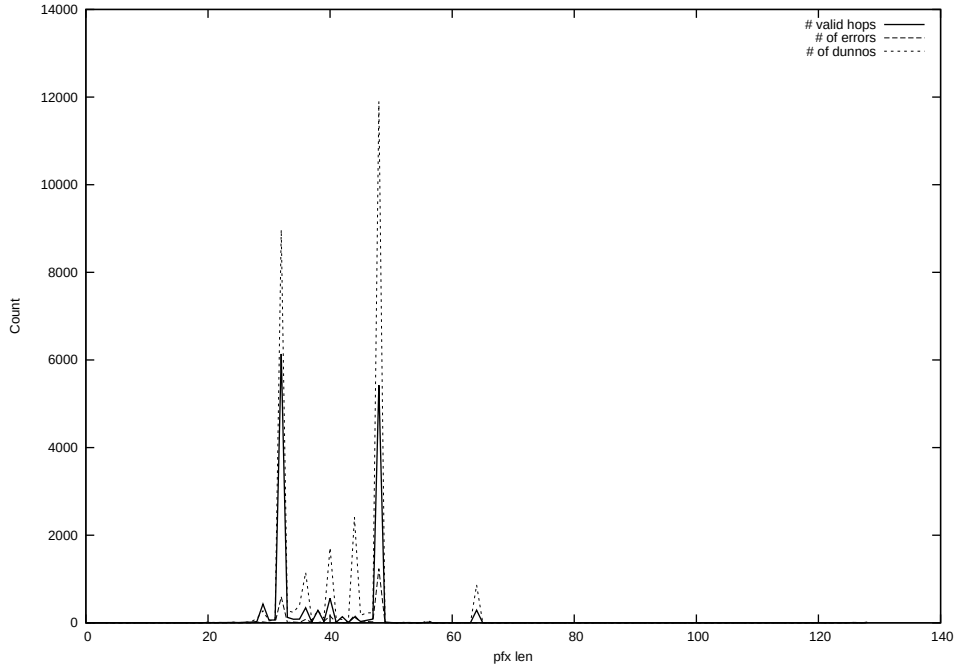
2013-09-28



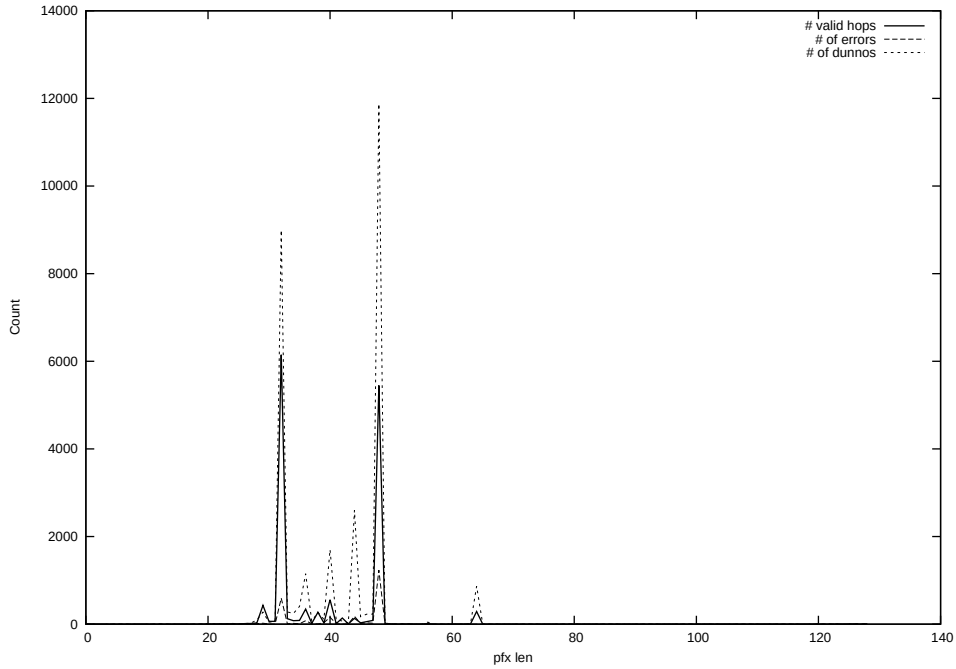
2013-09-29



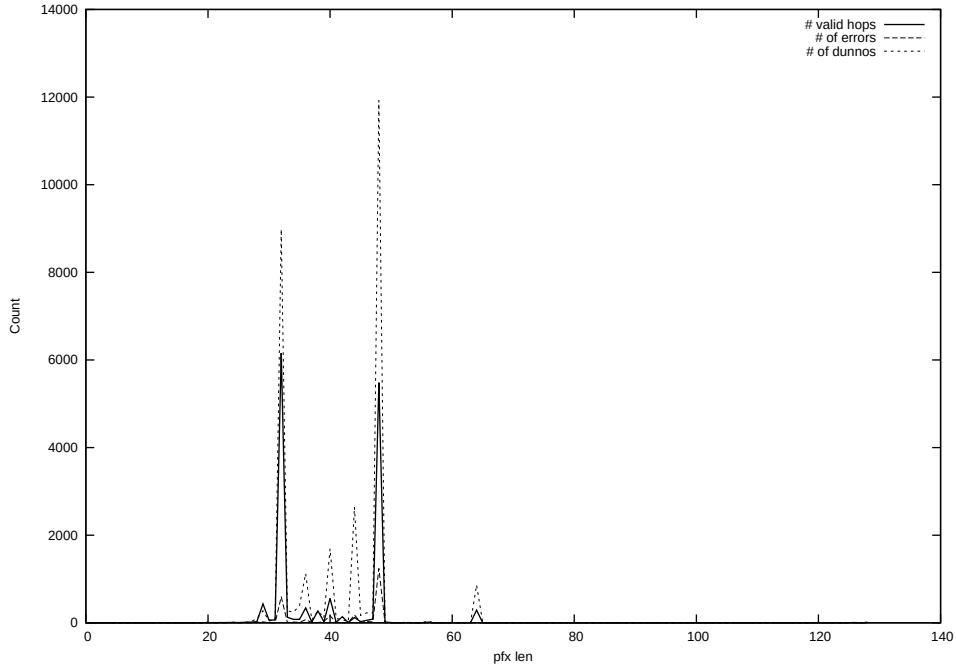
2013-09-30



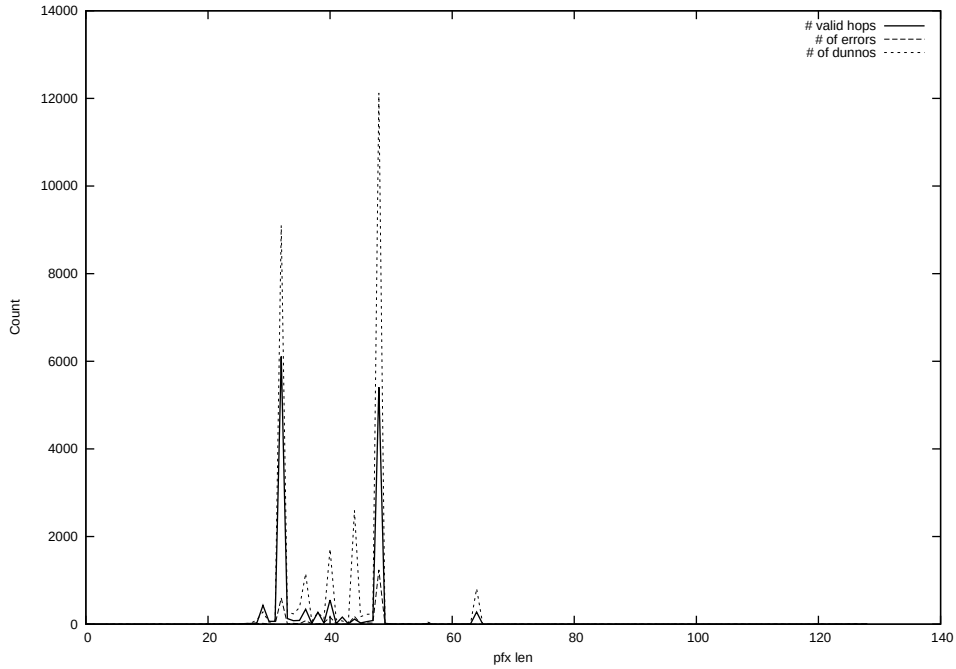
2013-10-01



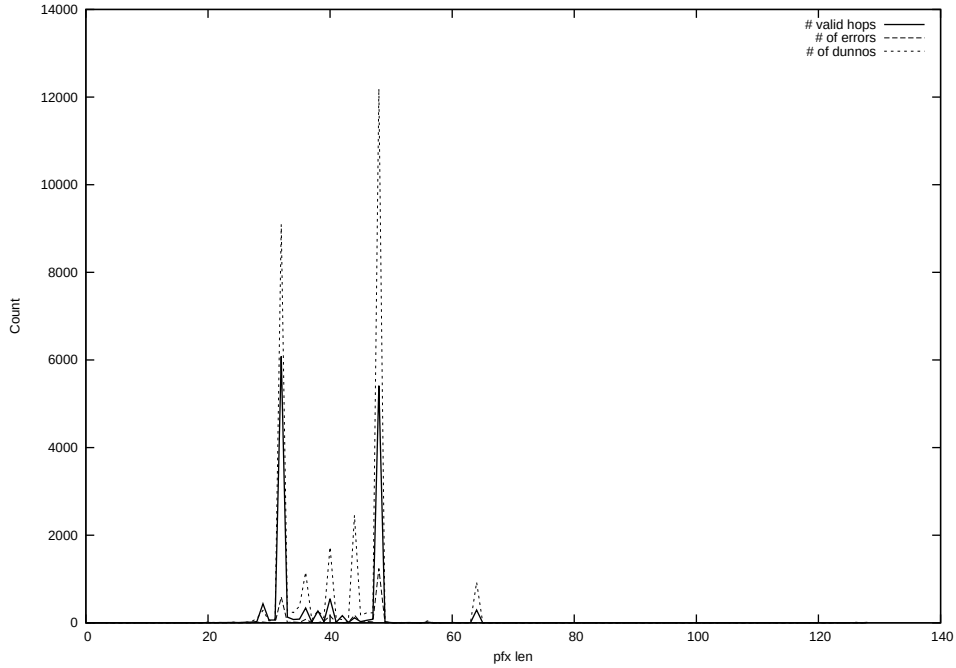
2013-10-02



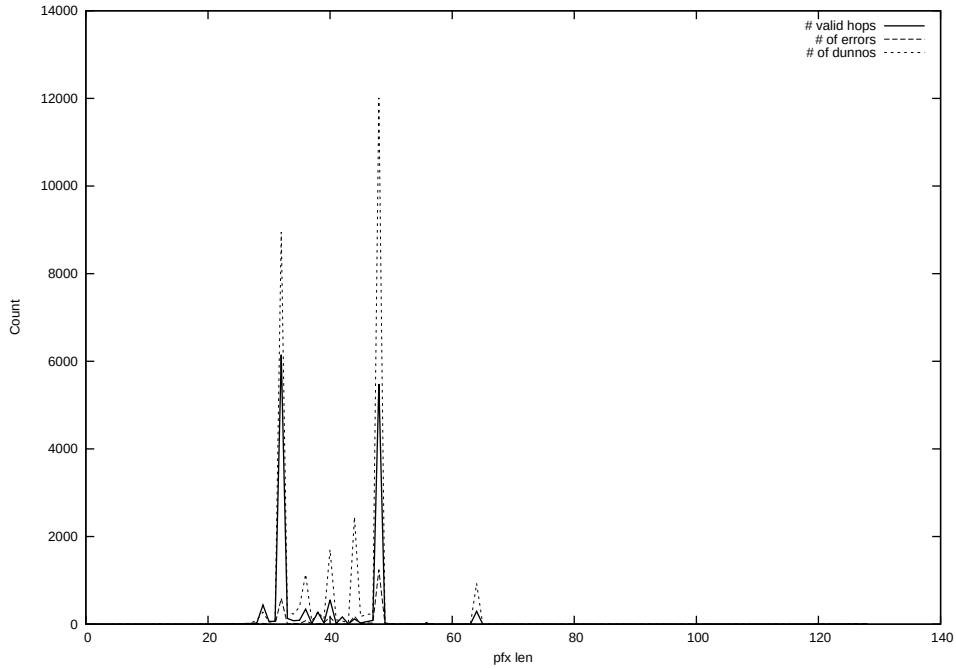
2013-10-03



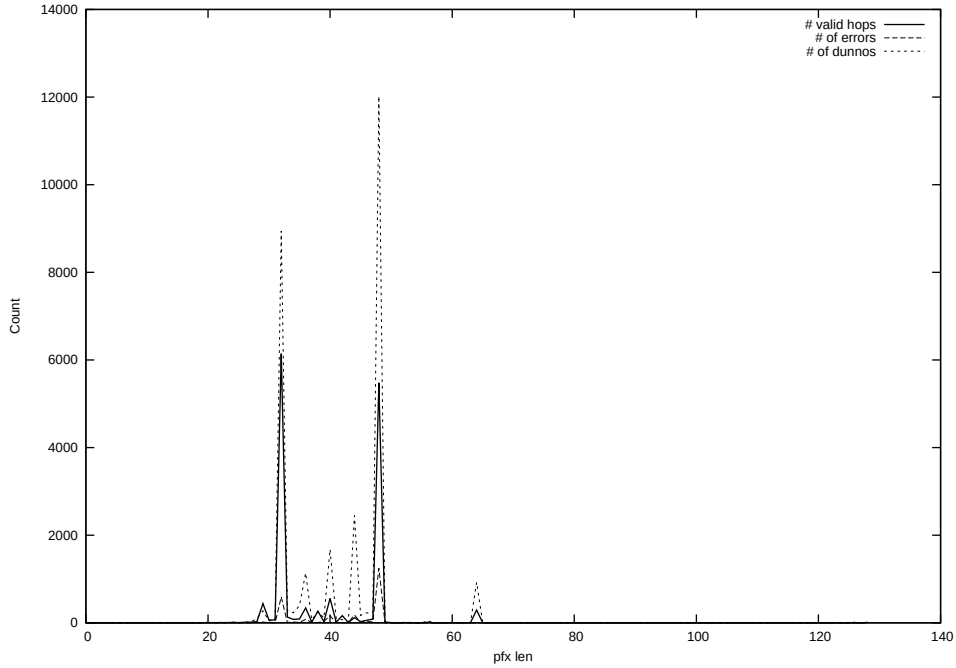
2013-10-04



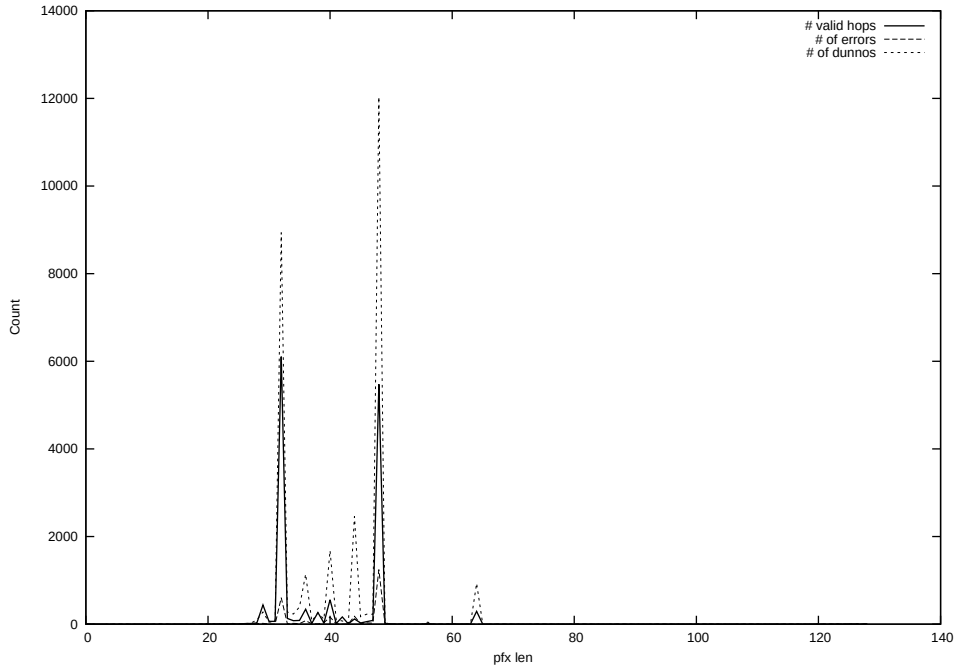
2013-10-05



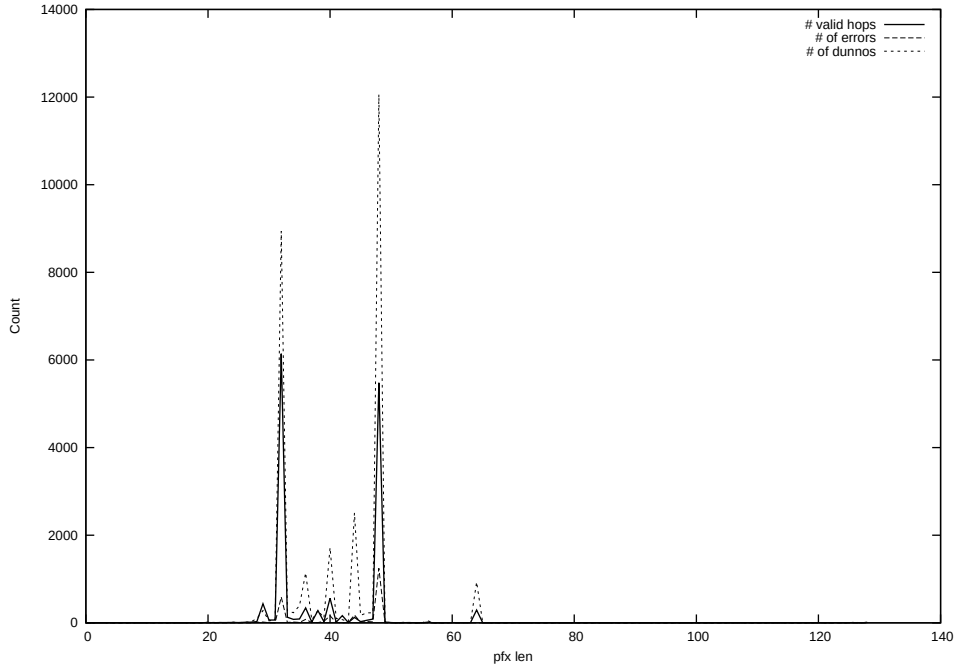
2013-10-06



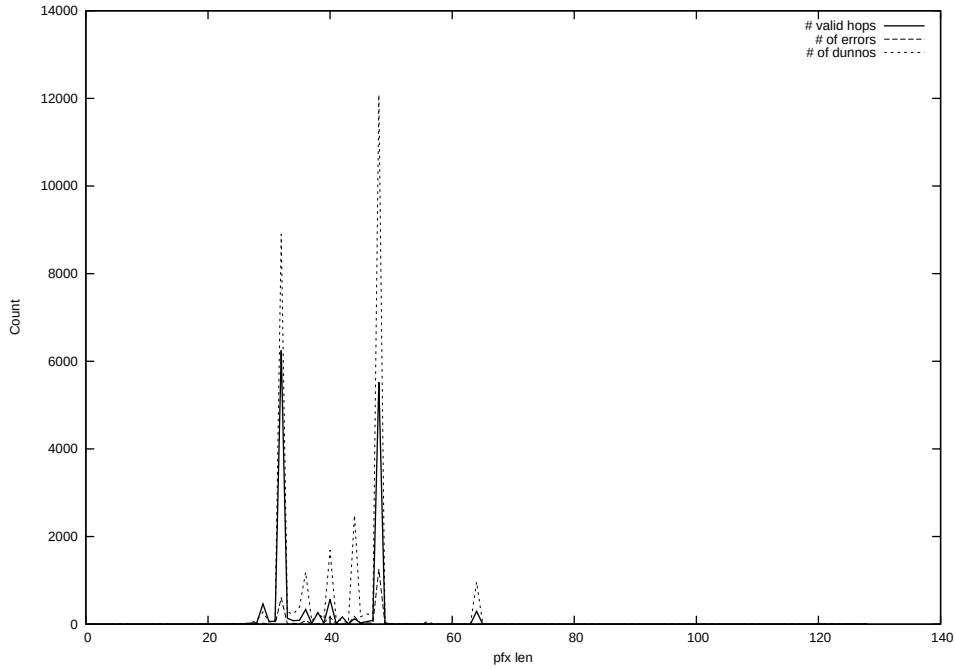
2013-10-07



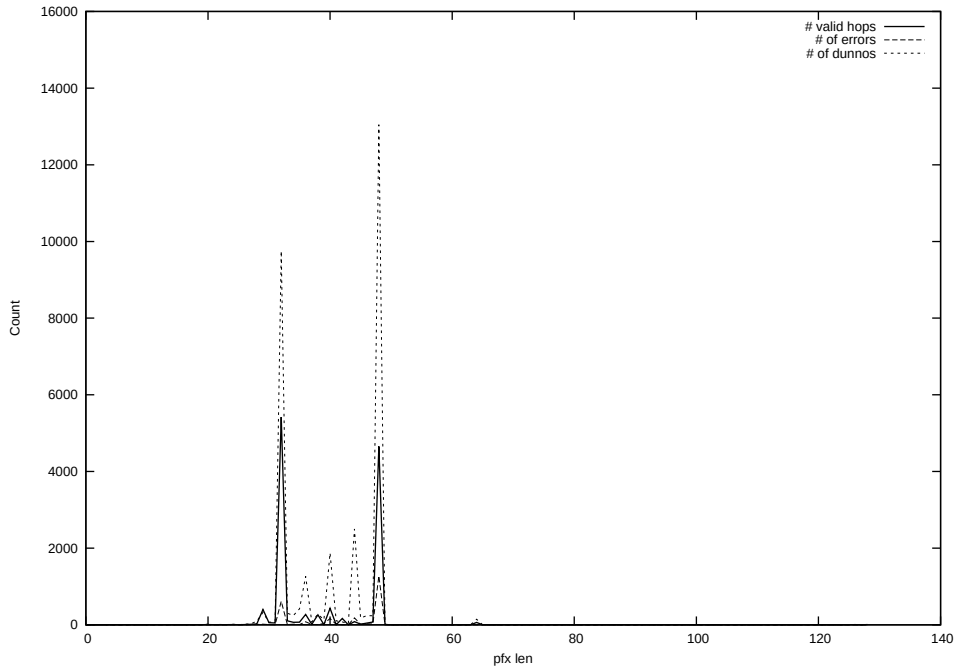
2013-10-08



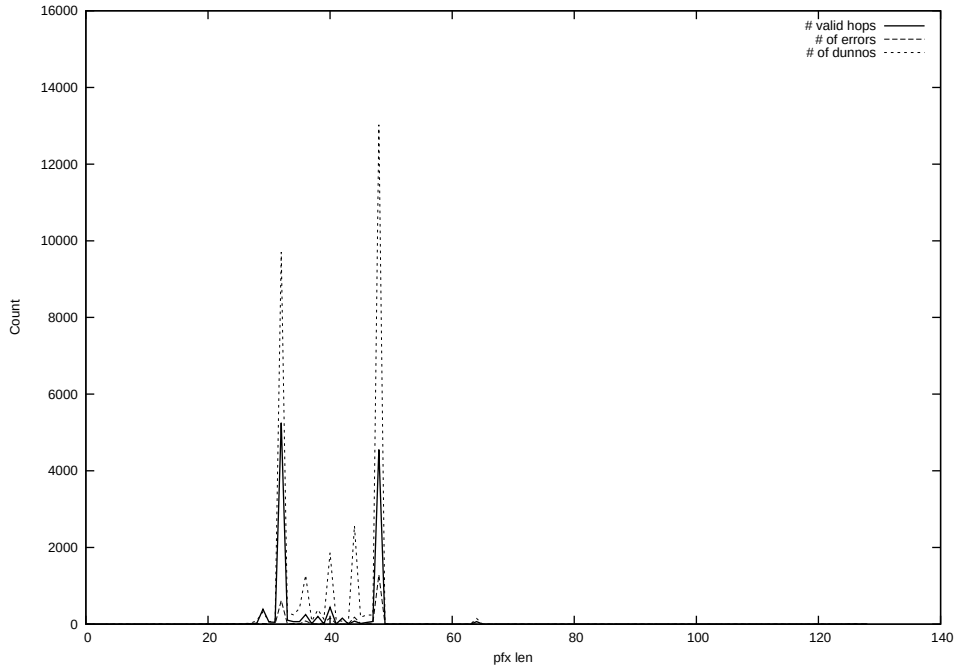
2013-10-09



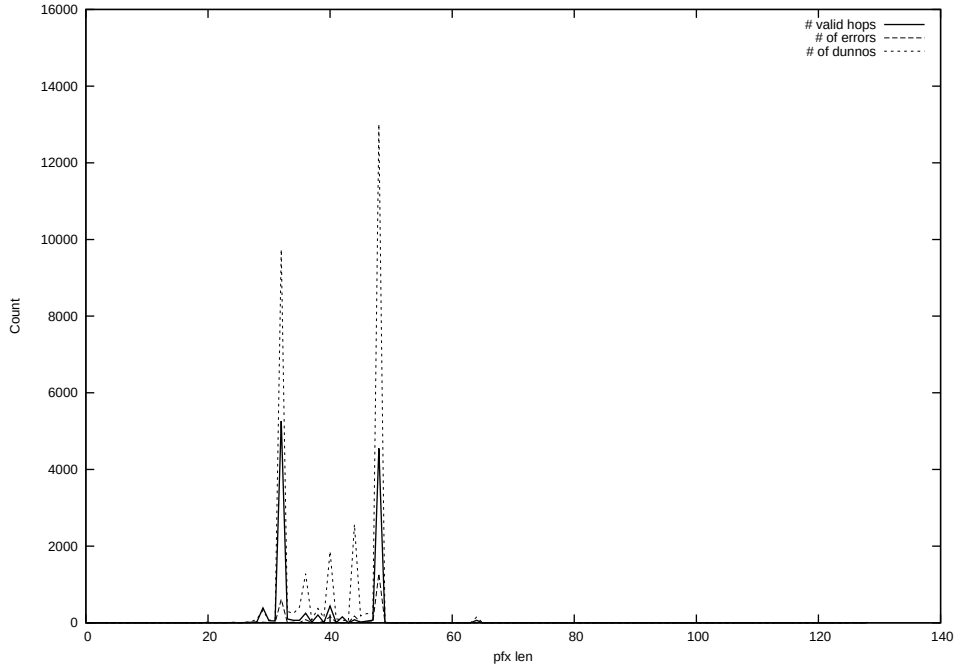
2013-10-10



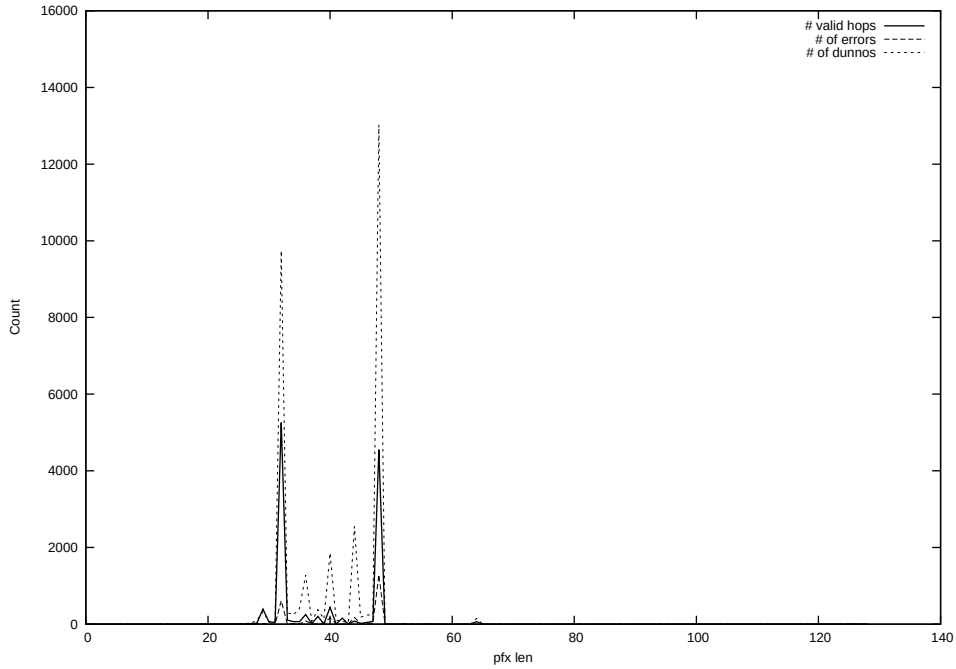
2013-10-11



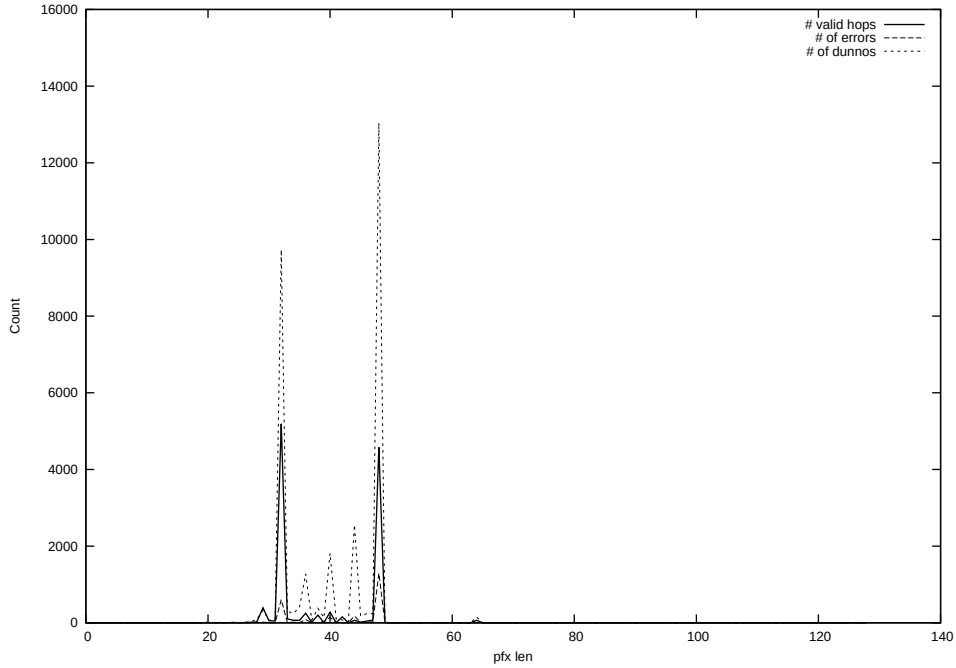
2013-10-12



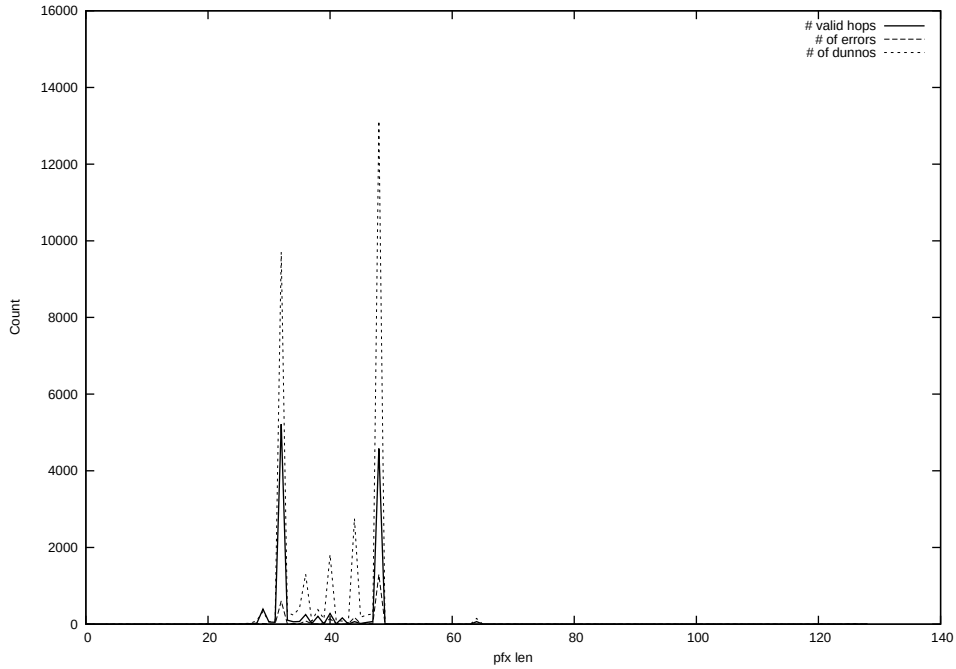
2013-10-13



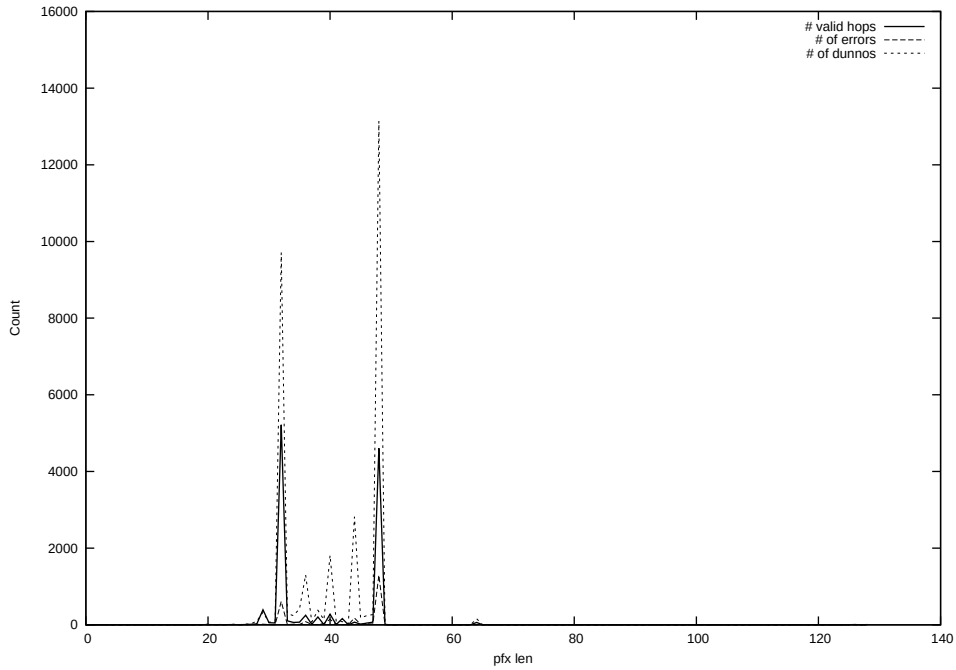
2013-10-14



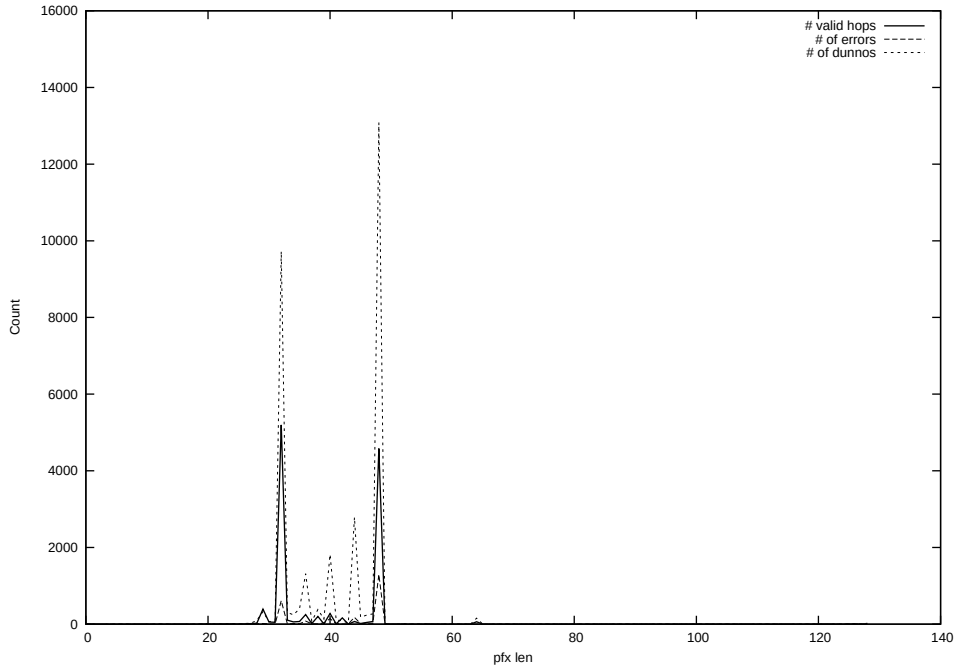
2013-10-15



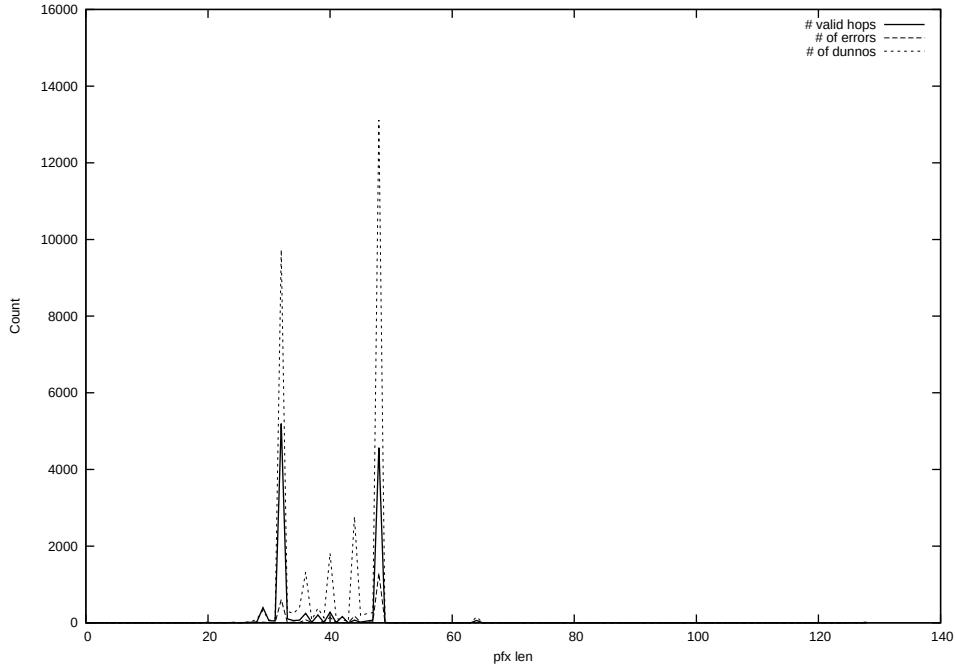
2013-10-16



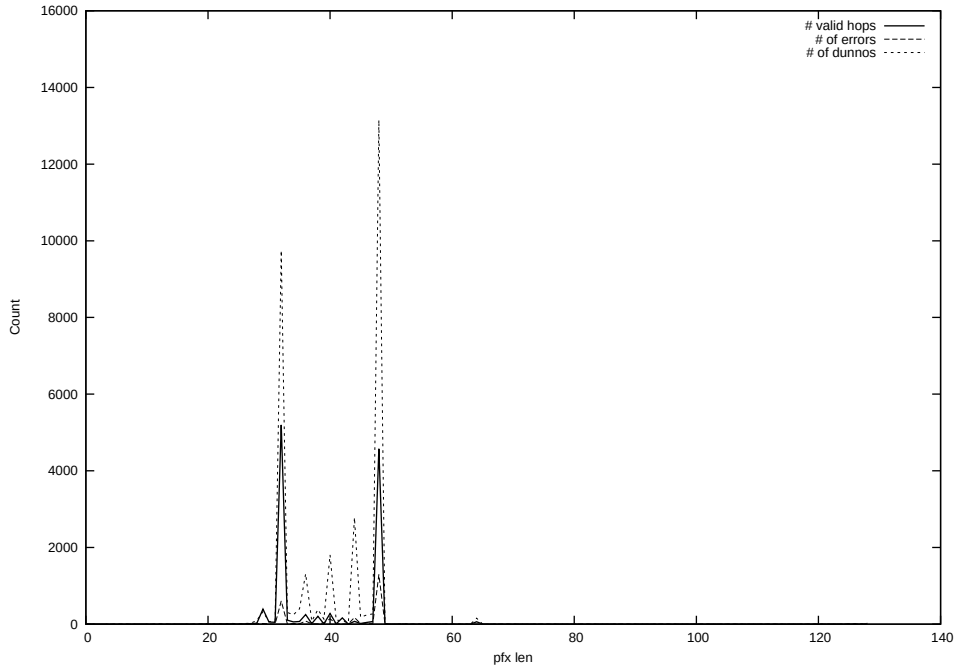
2013-10-17



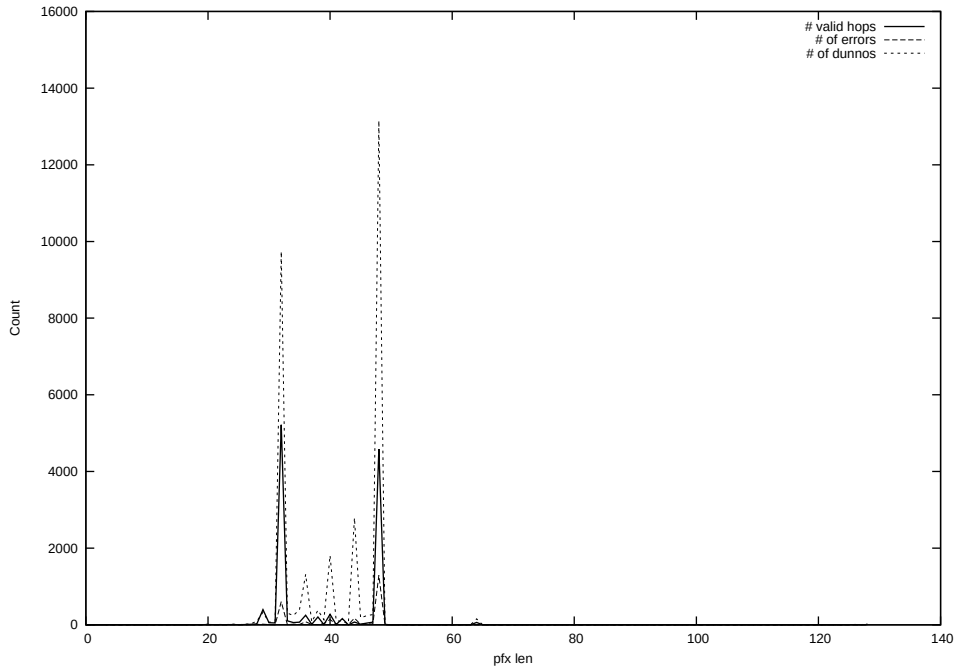
2013-10-18



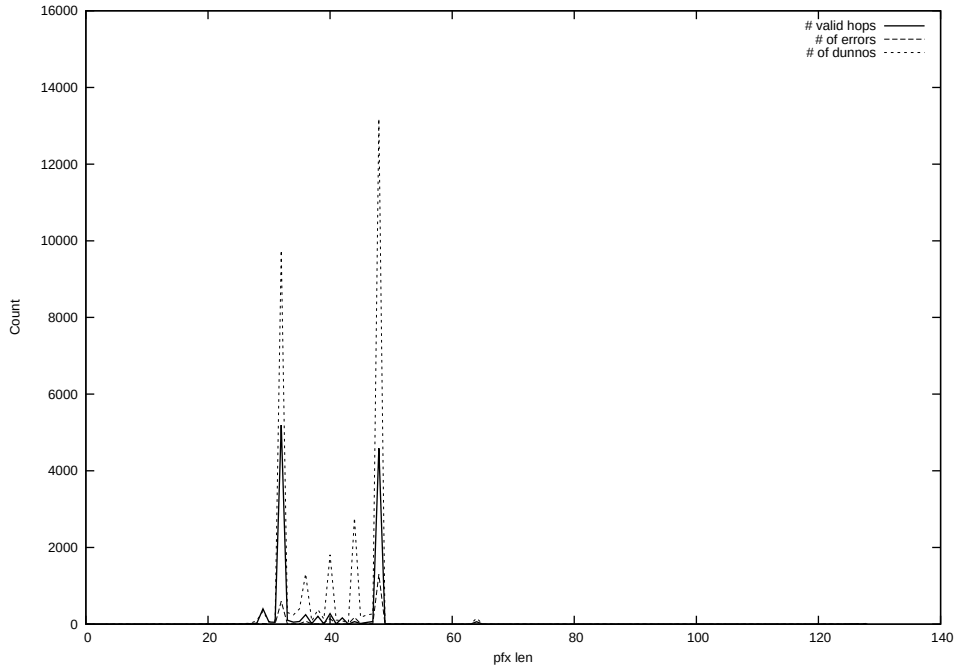
2013-10-19



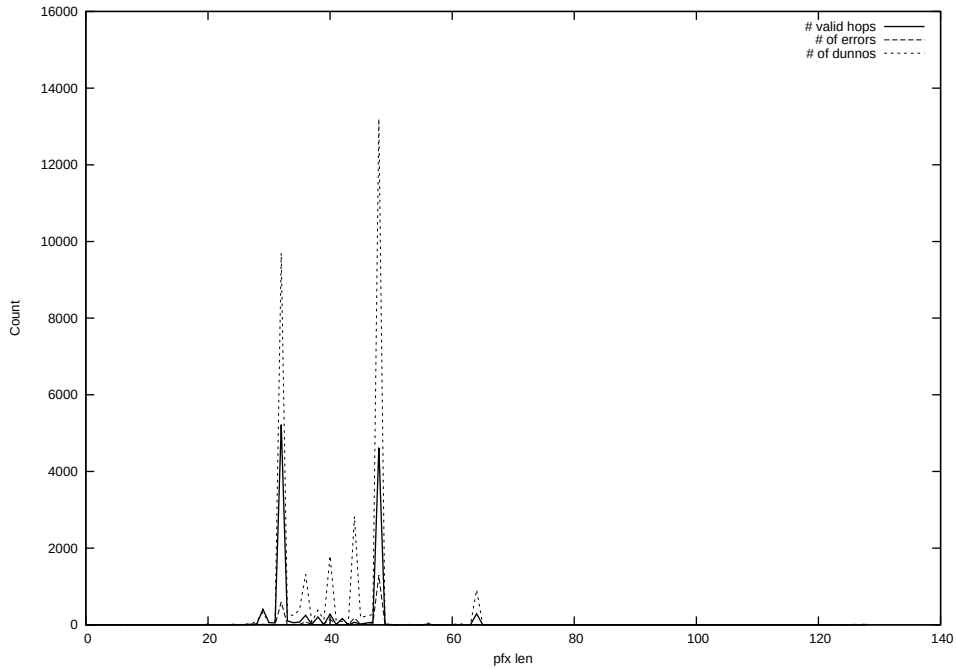
2013-10-20



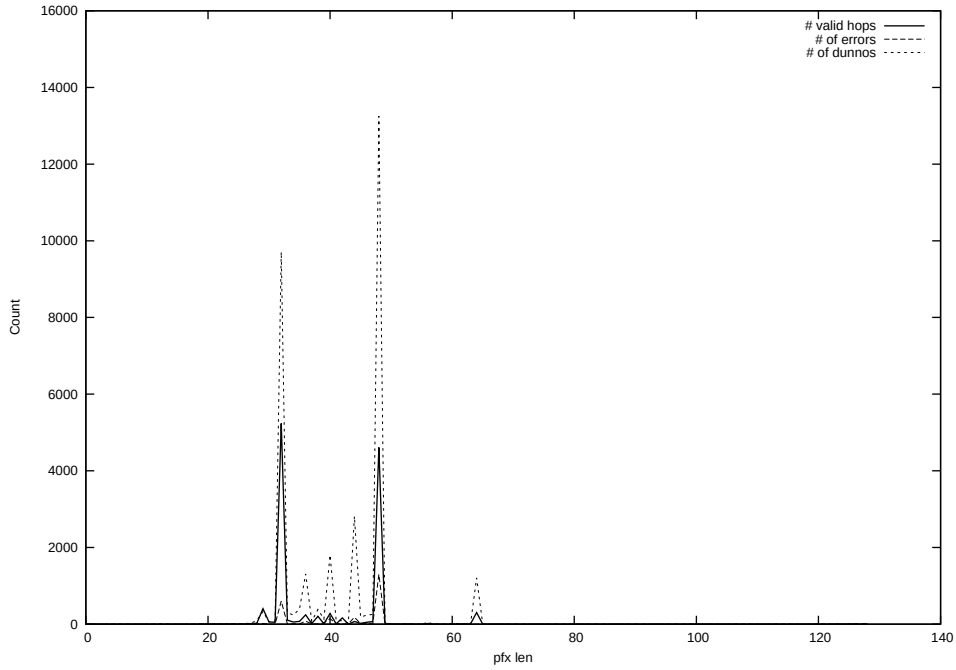
2013-10-21



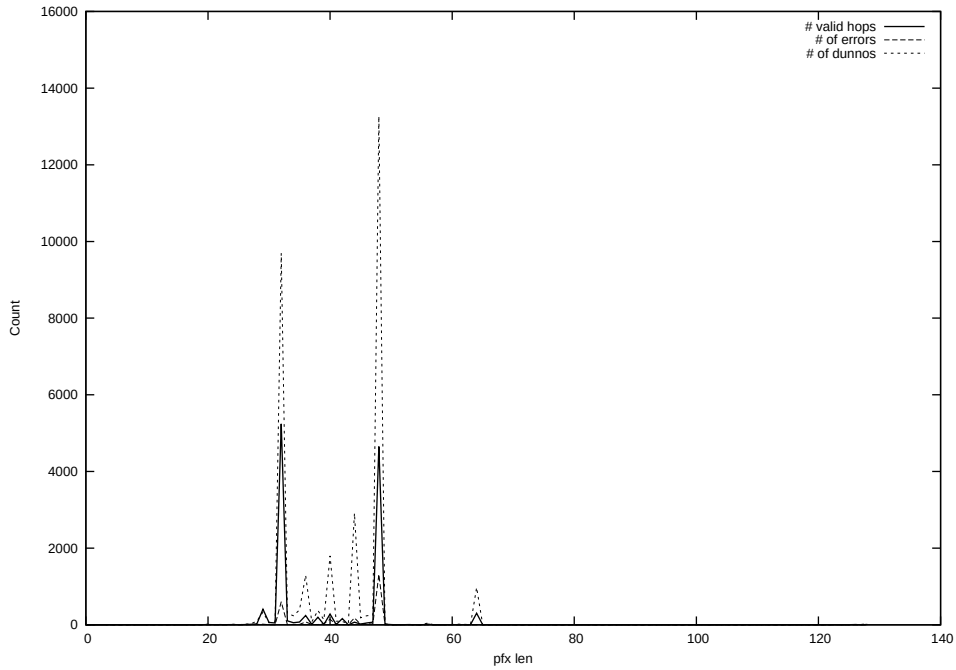
2013-10-22



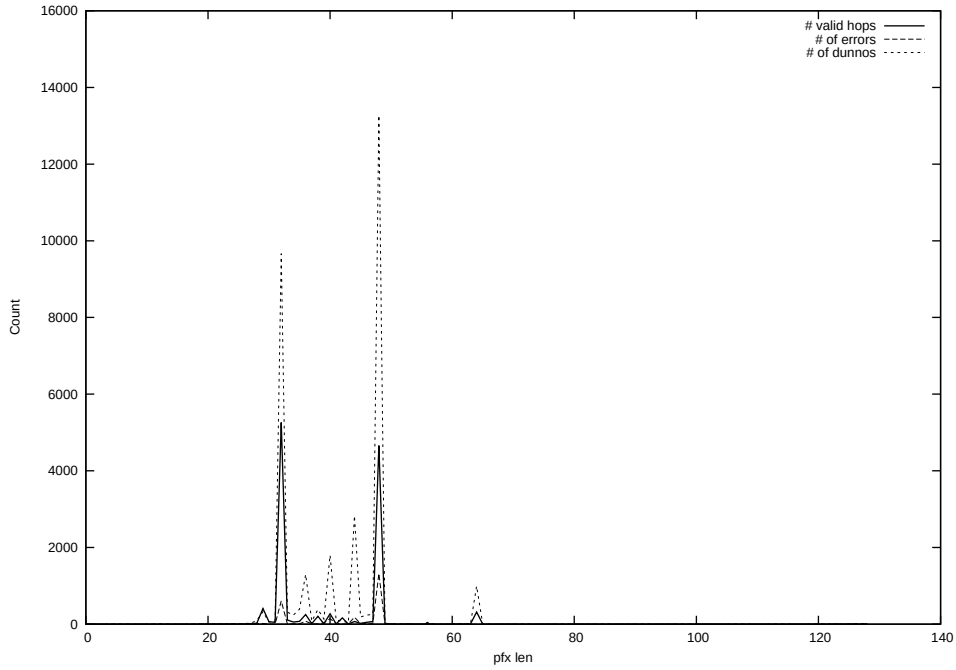
2013-10-23



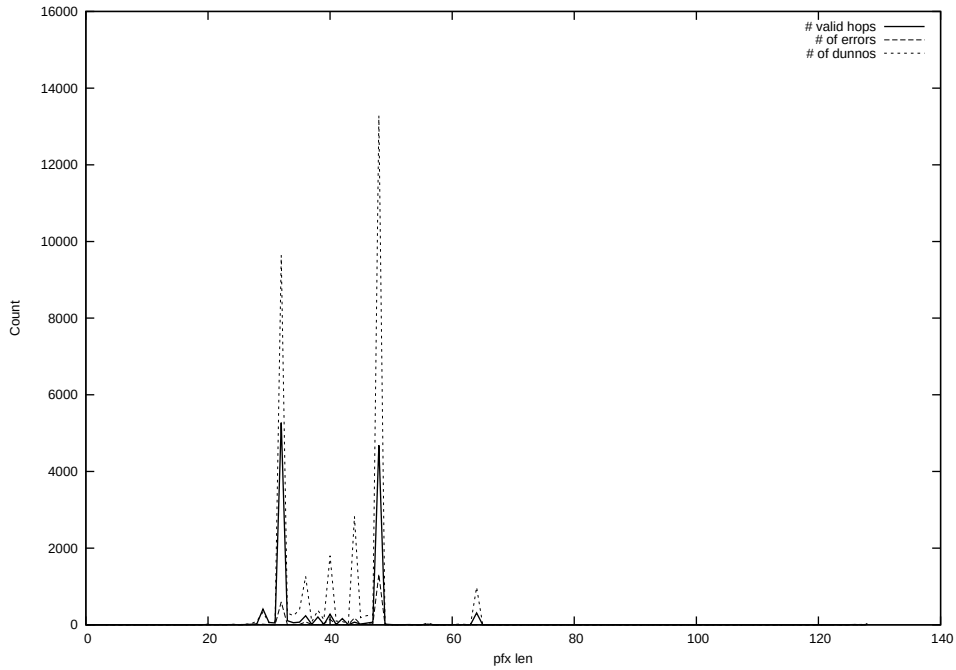
2013-10-24



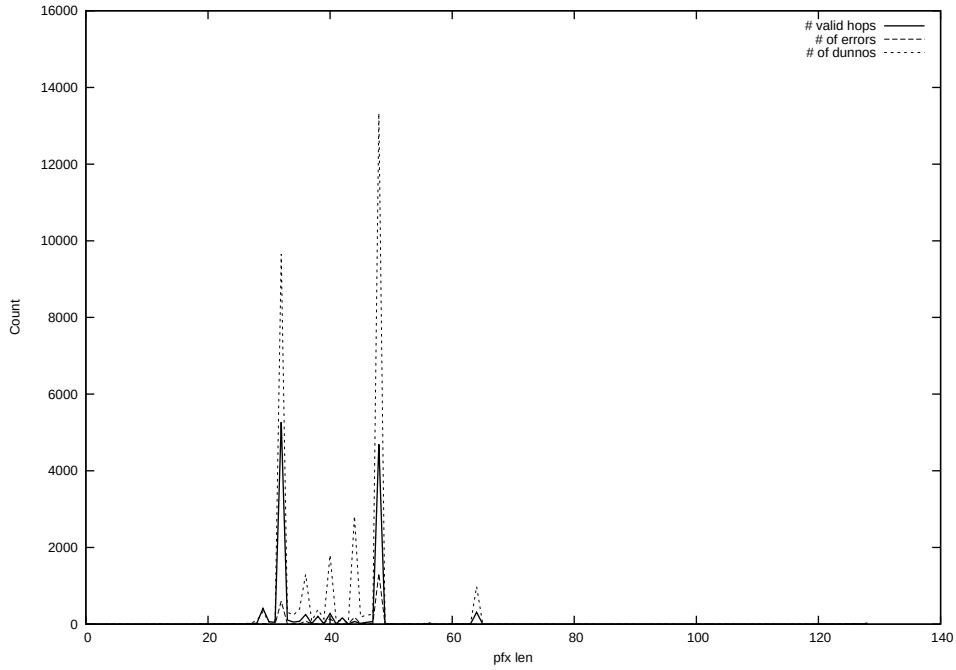
2013-10-25



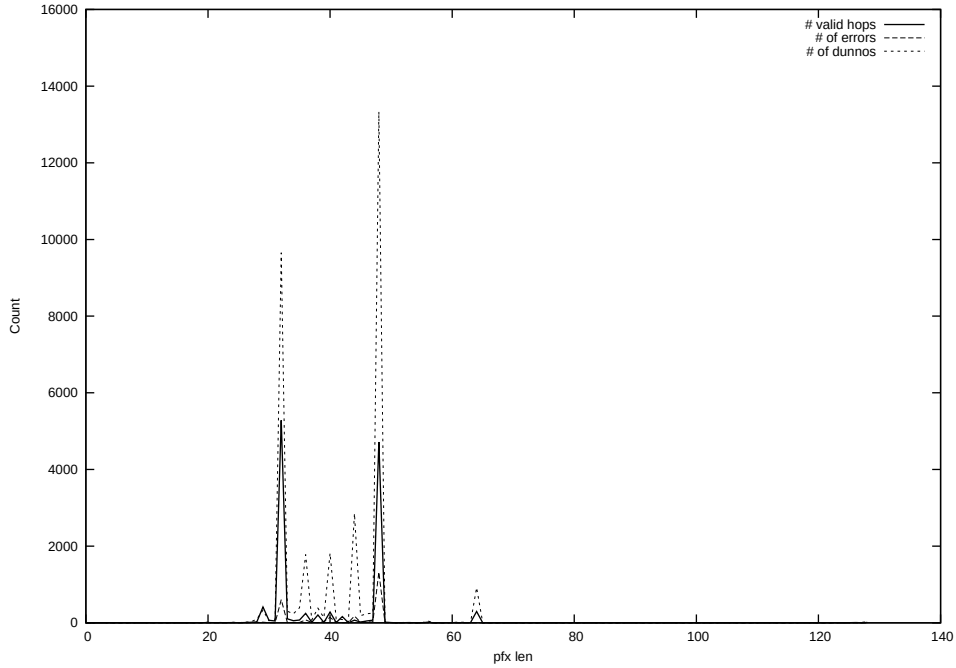
2013-10-26



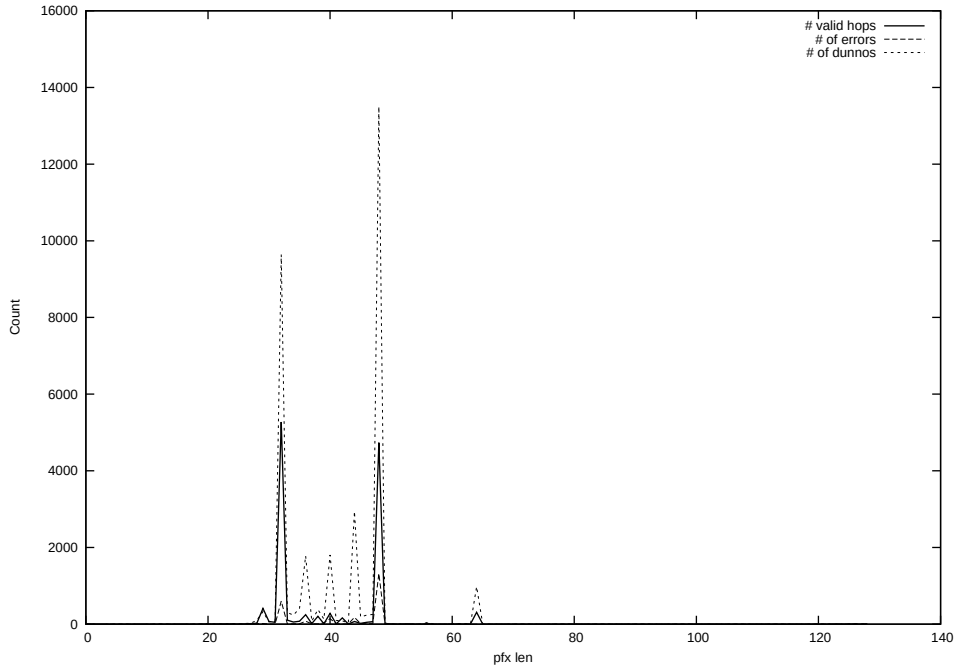
2013-10-27



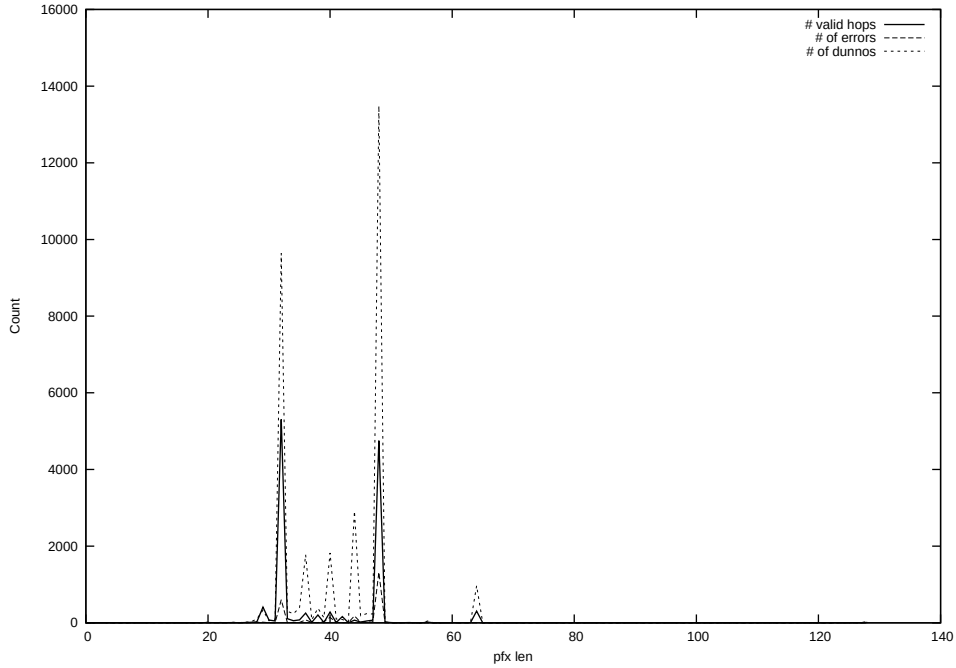
2013-10-28



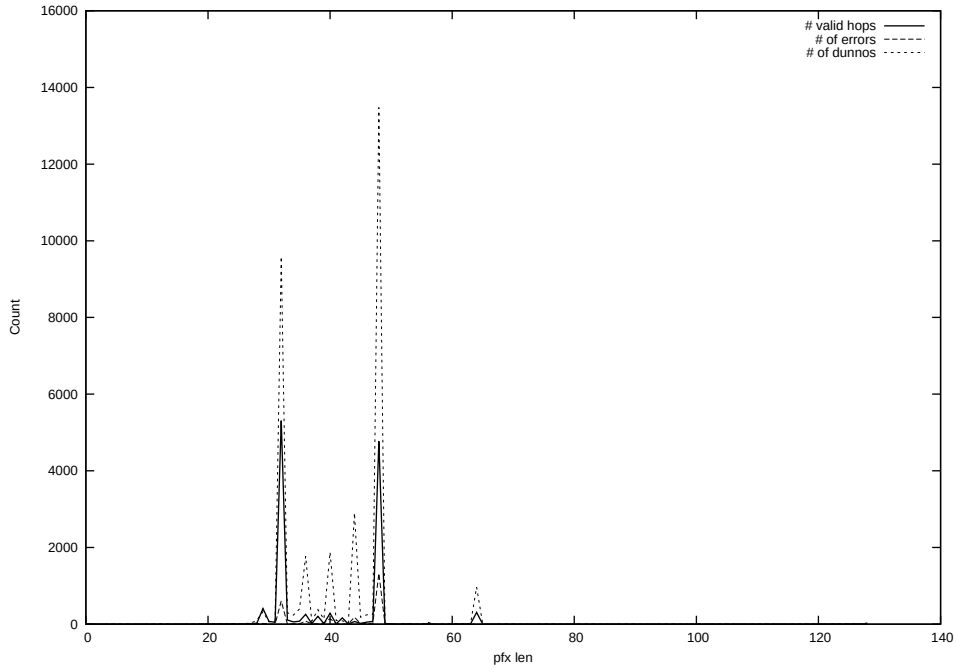
2013-10-29



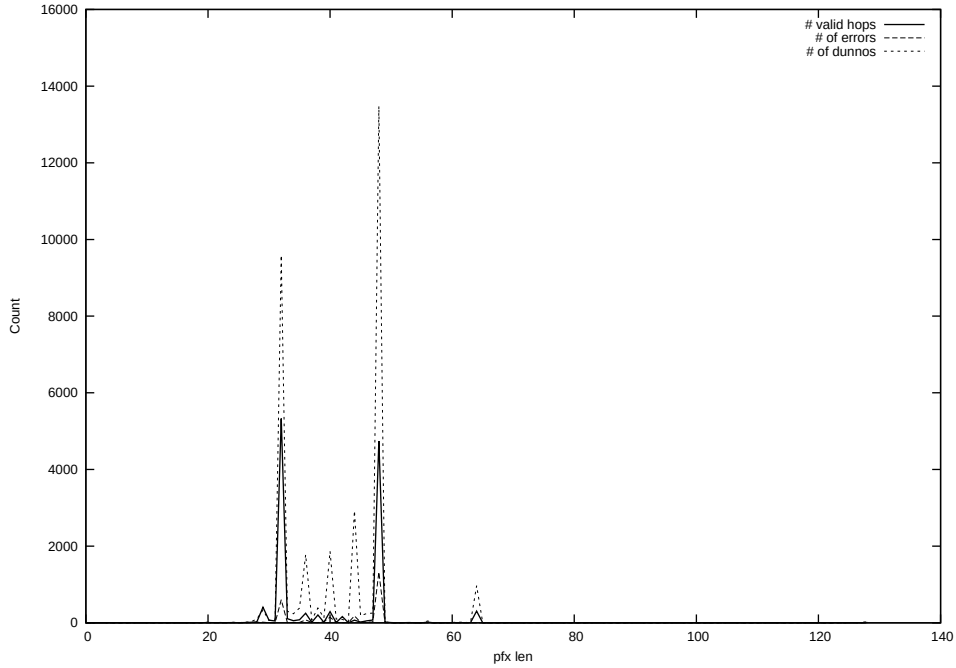
2013-10-30



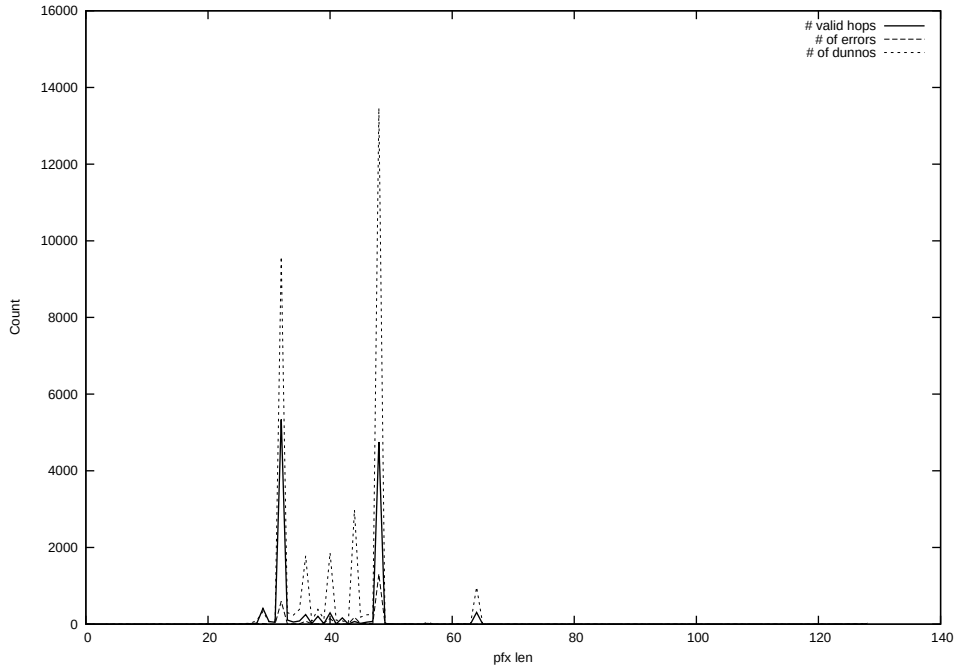
2013-10-31



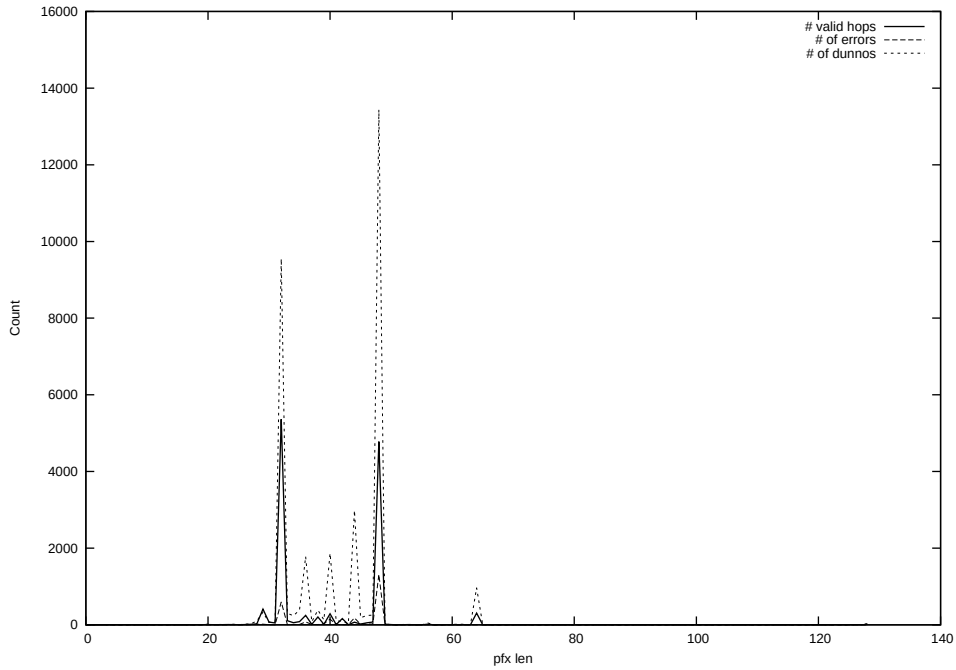
2013-11-01



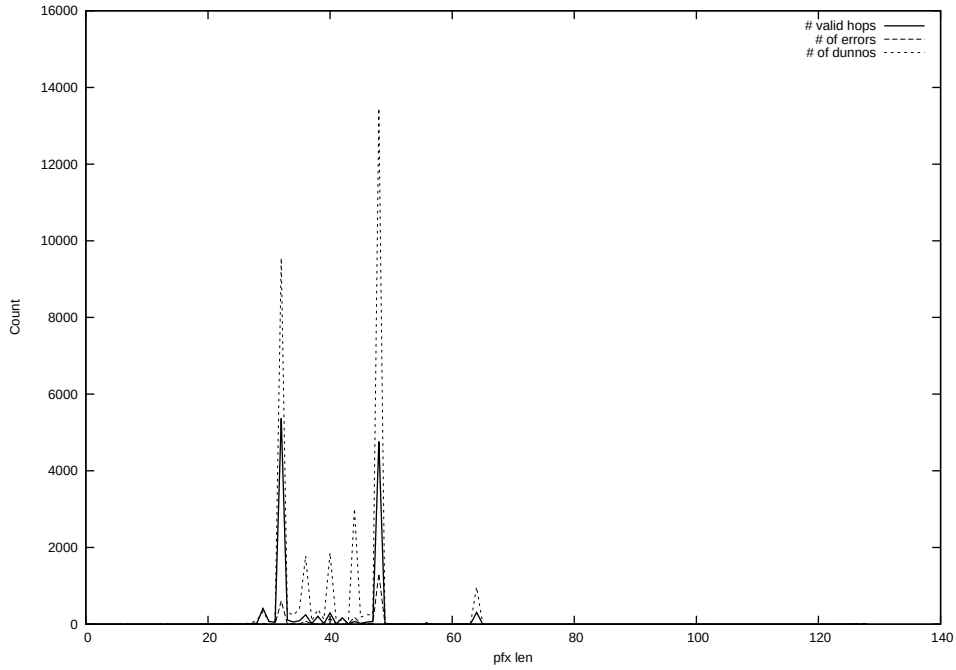
2013-11-02



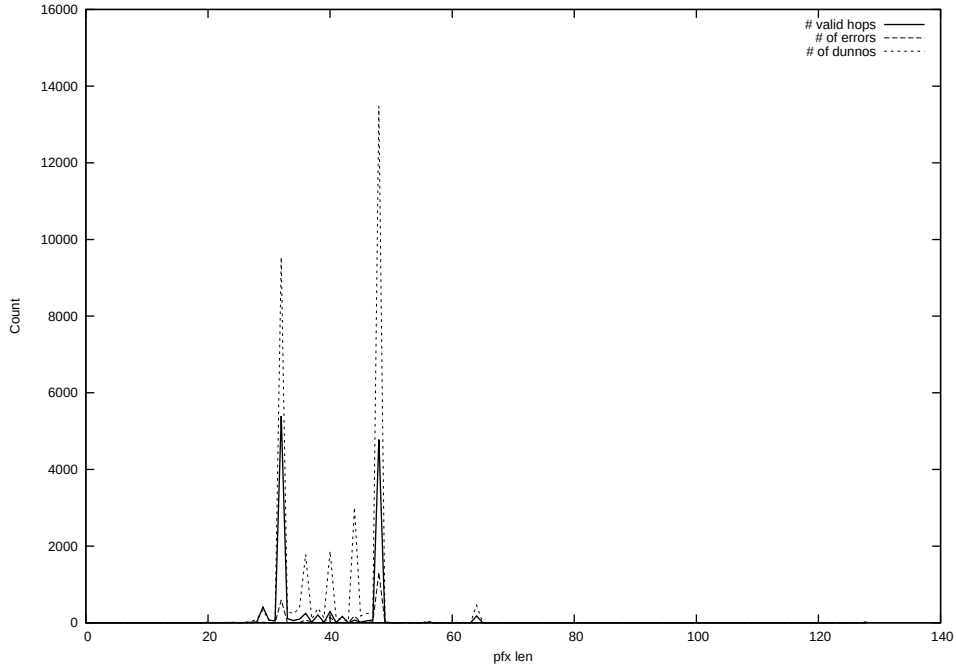
2013-11-03



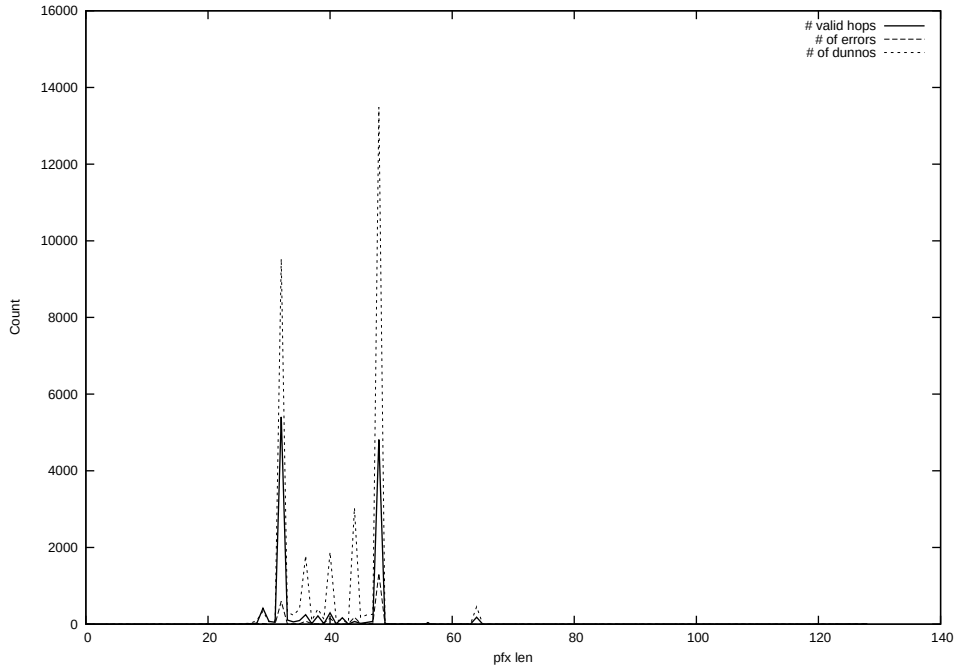
2013-11-04



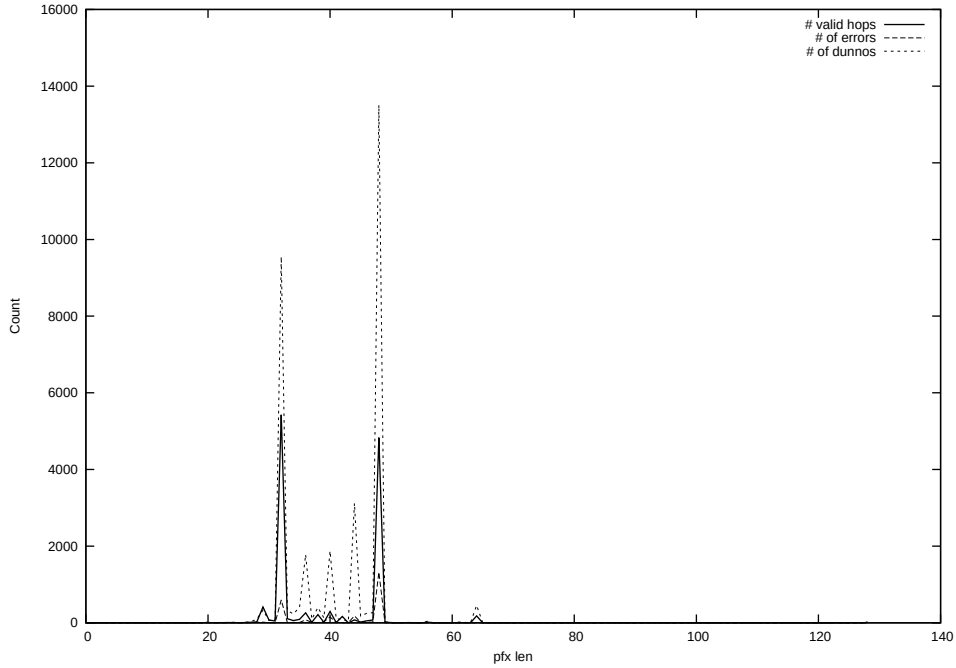
2013-11-05



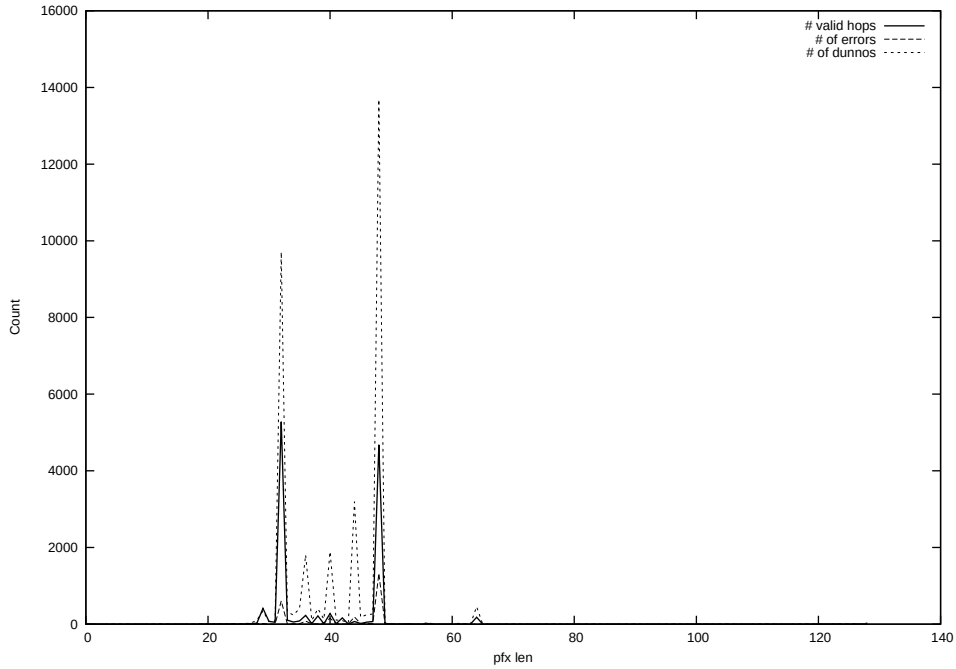
2013-11-06



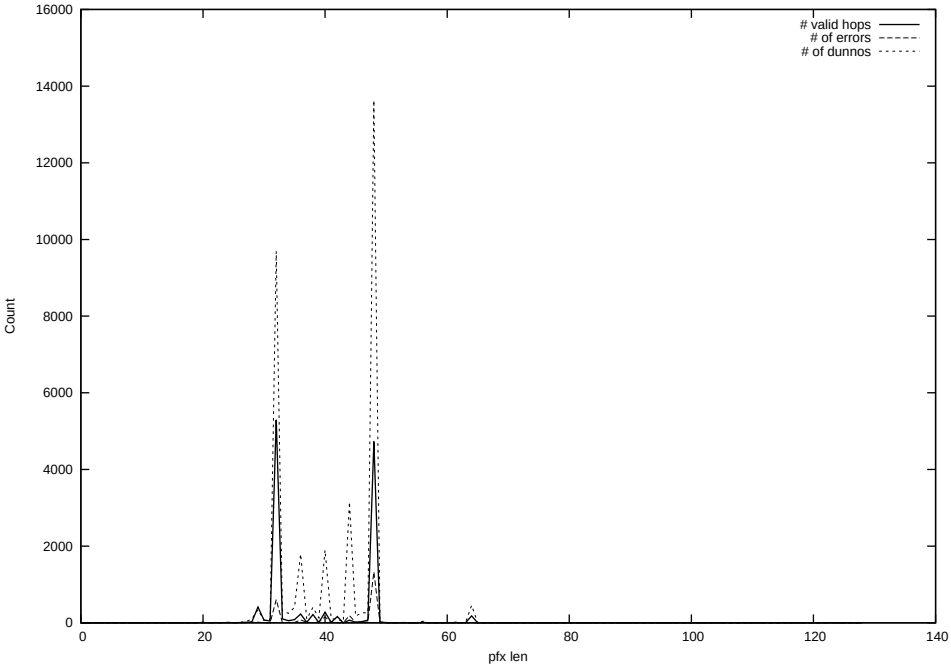
2013-11-07



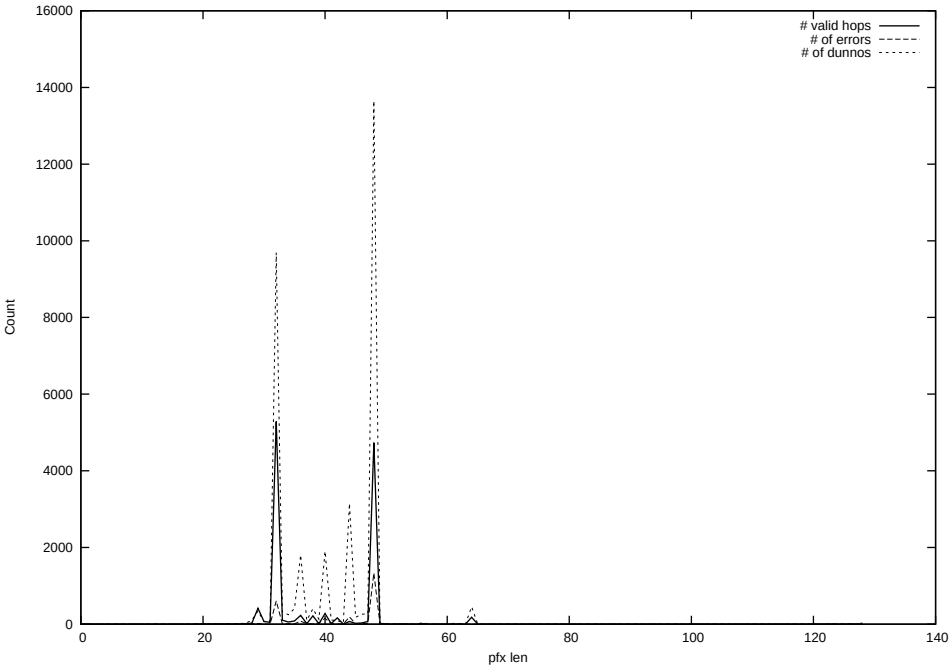
2013-11-08



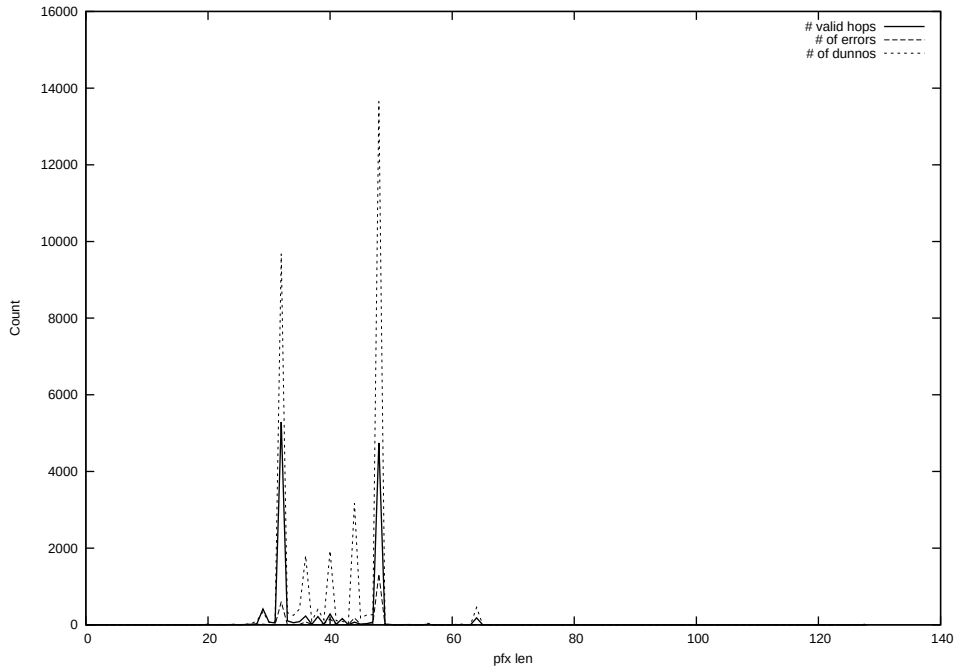
2013-11-09



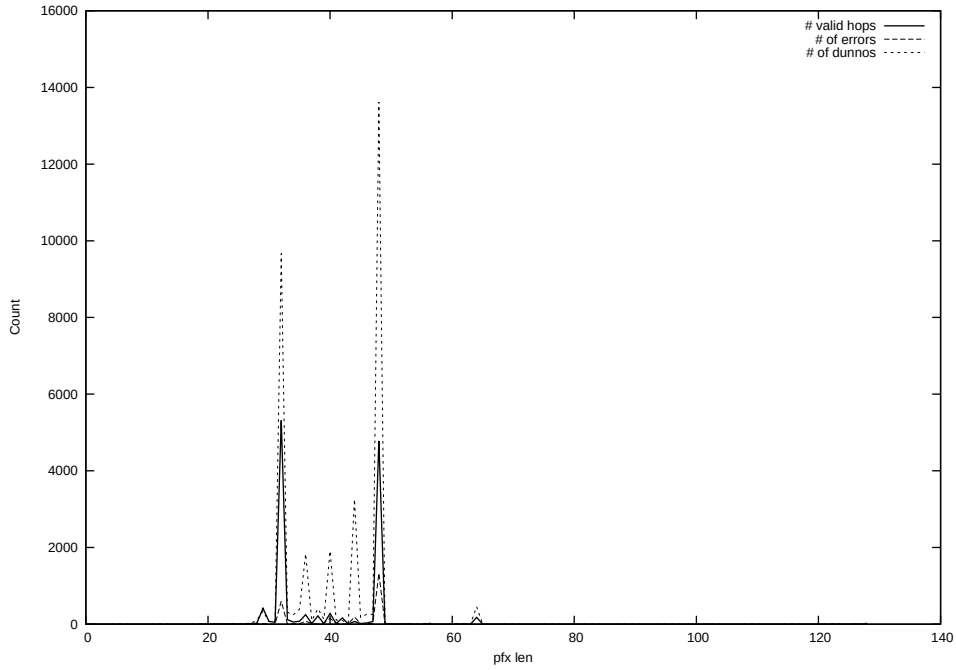
2013-11-10



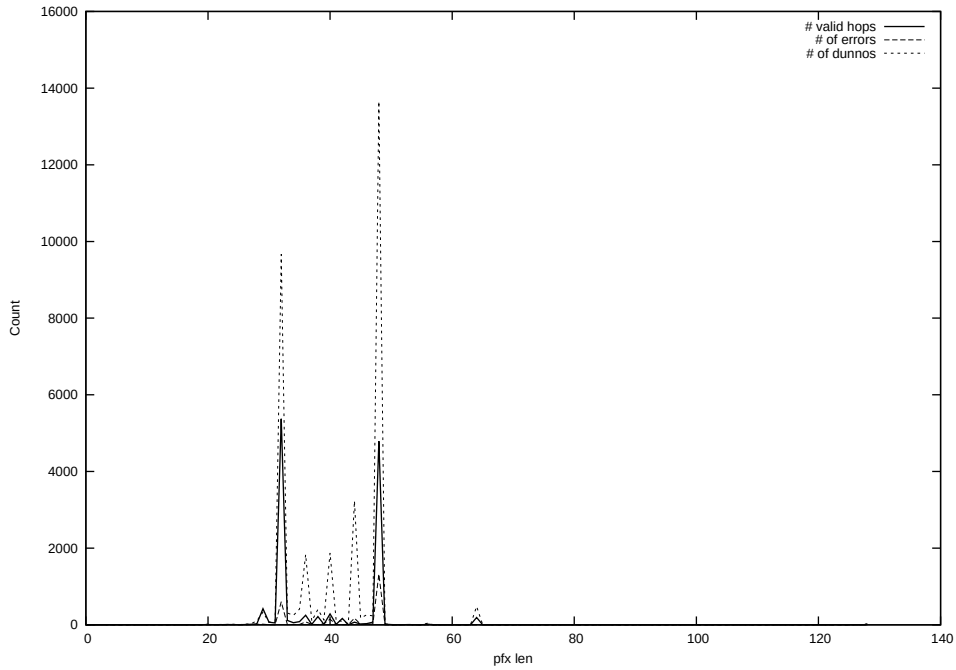
2013-11-11



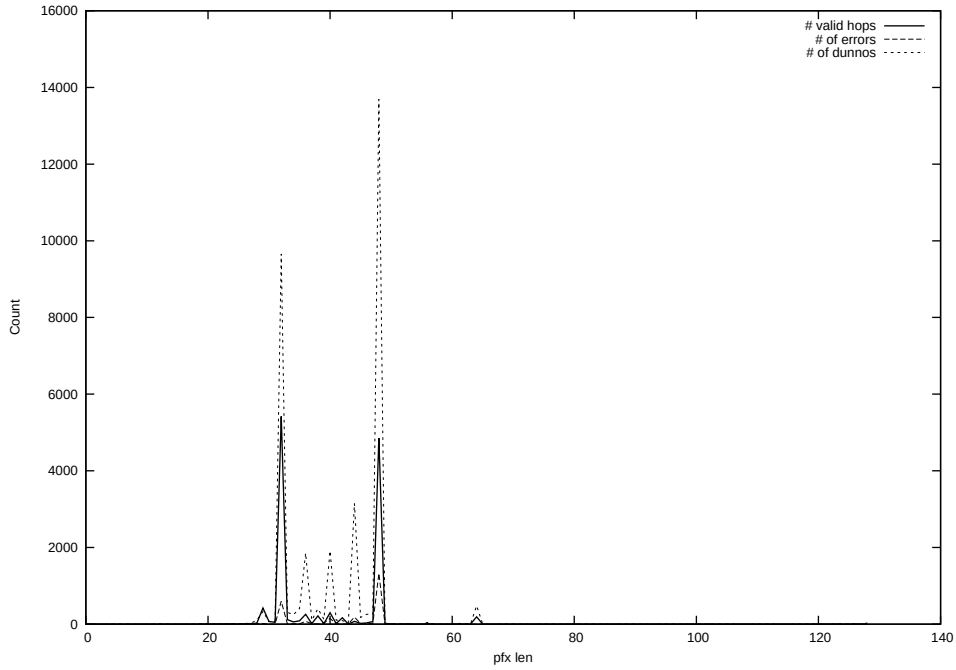
2013-11-12



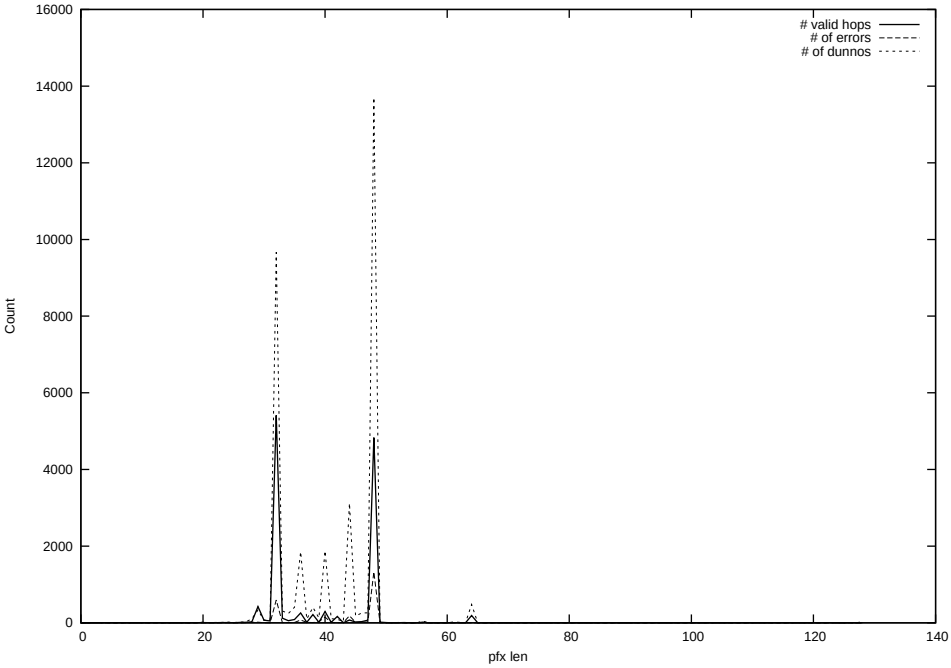
2013-11-13



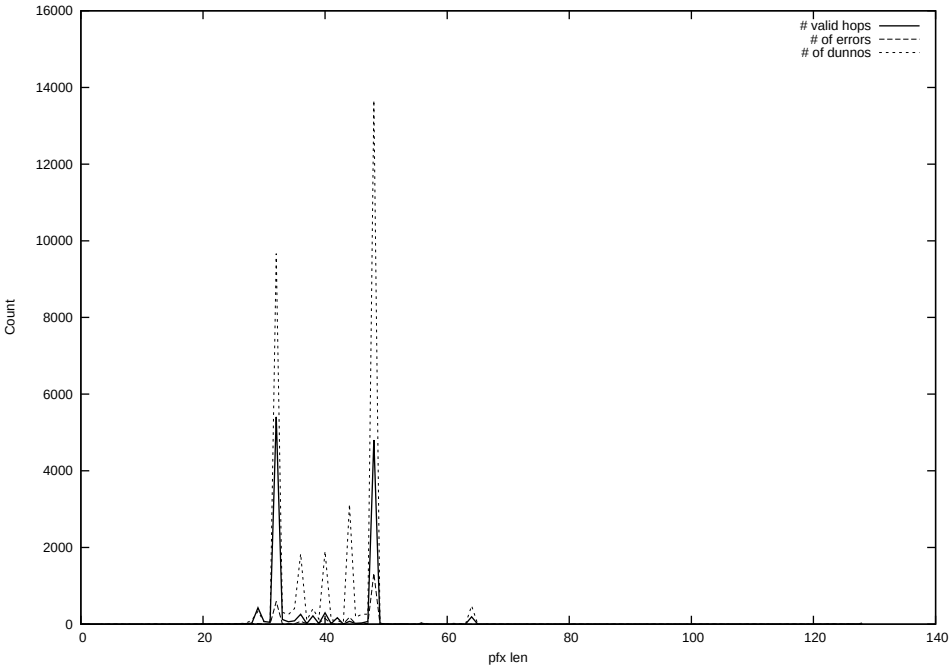
2013-11-14



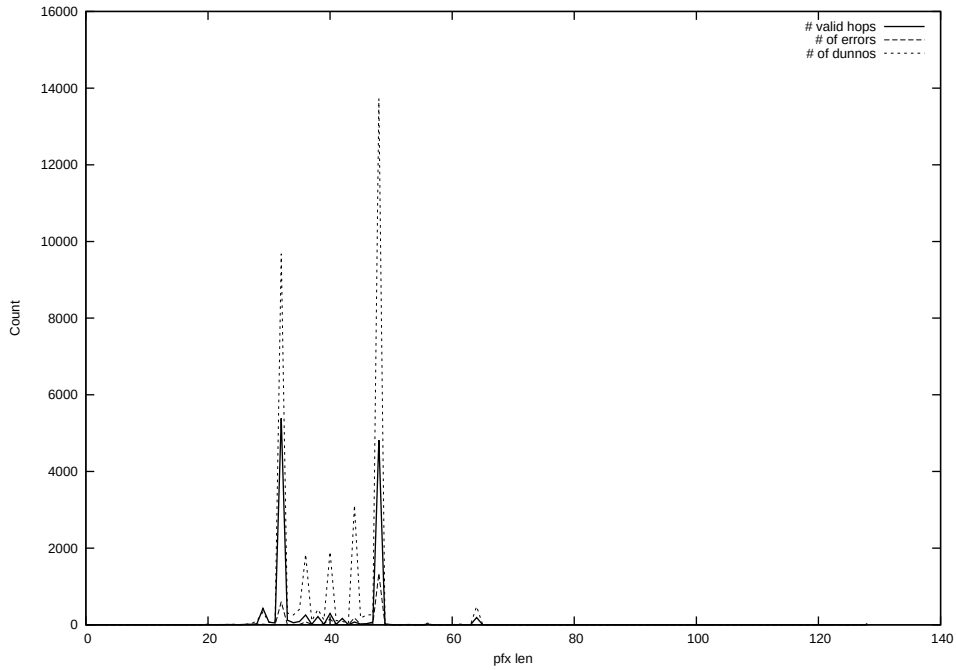
2013-11-15



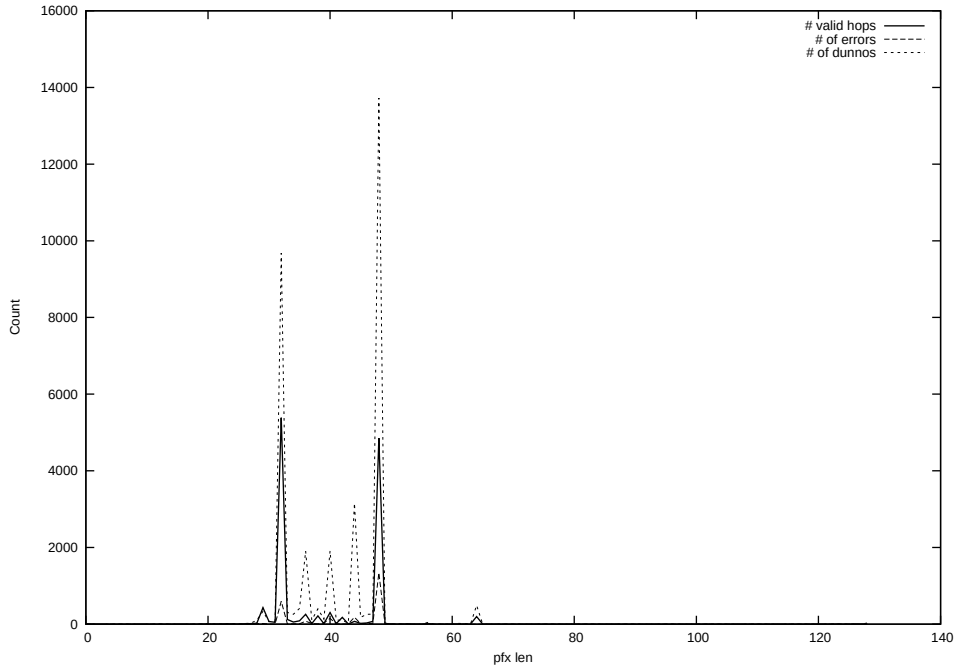
2013-11-16



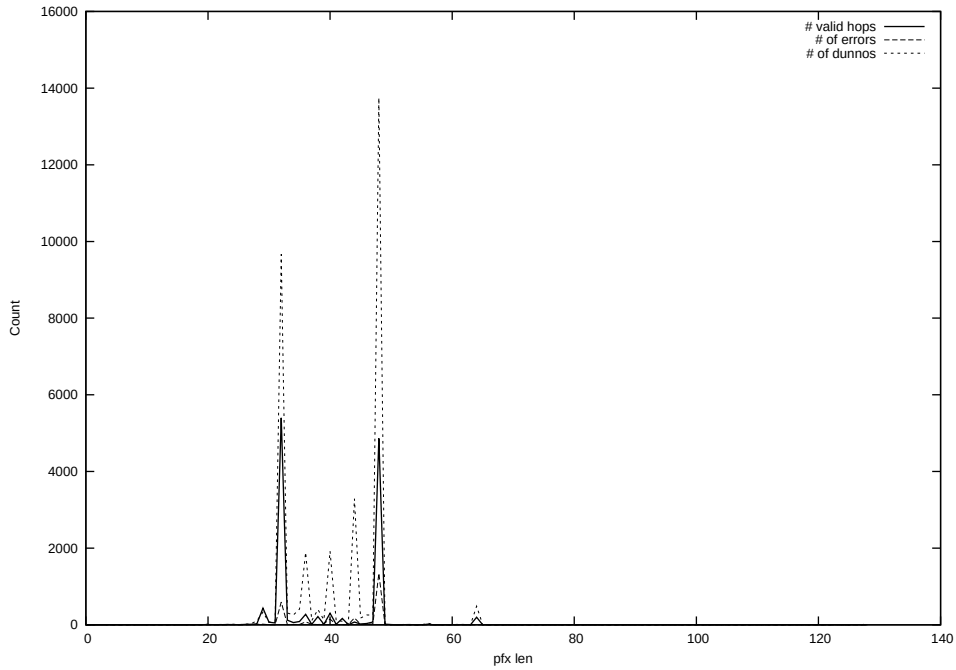
2013-11-17



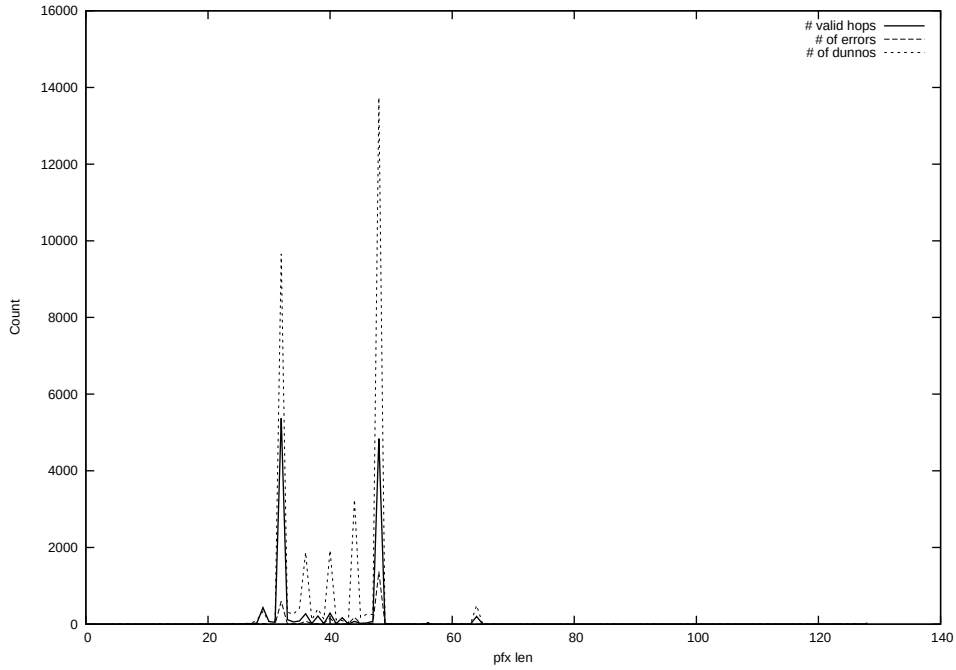
2013-11-18



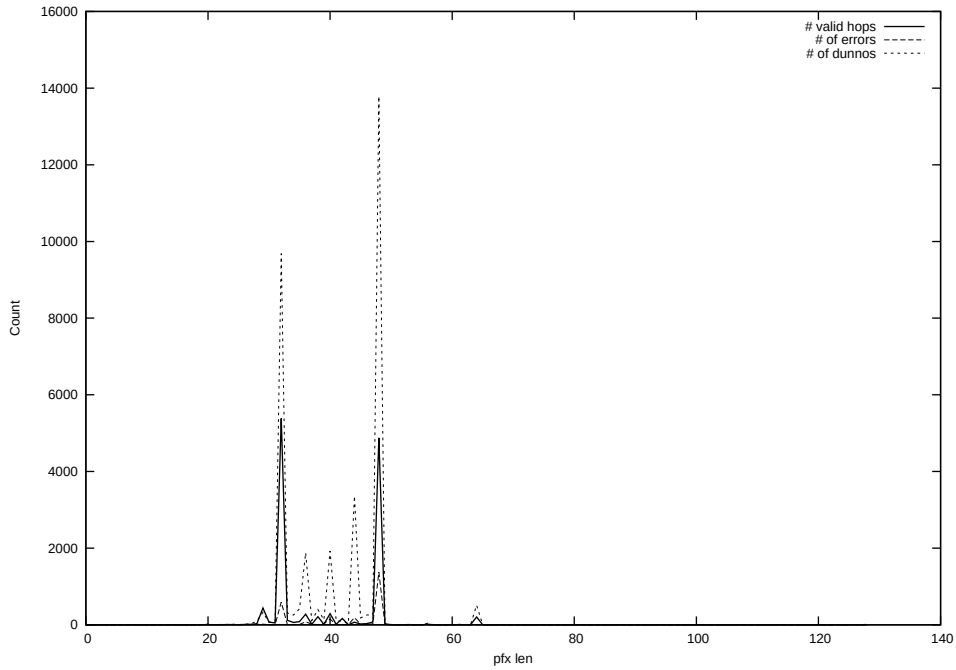
2013-11-19



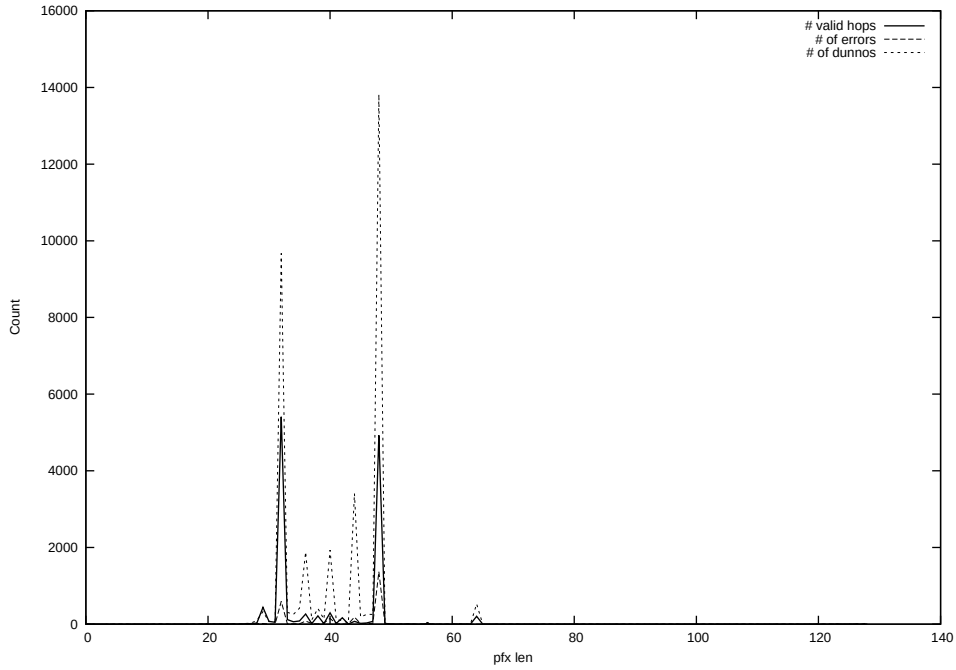
2013-11-20



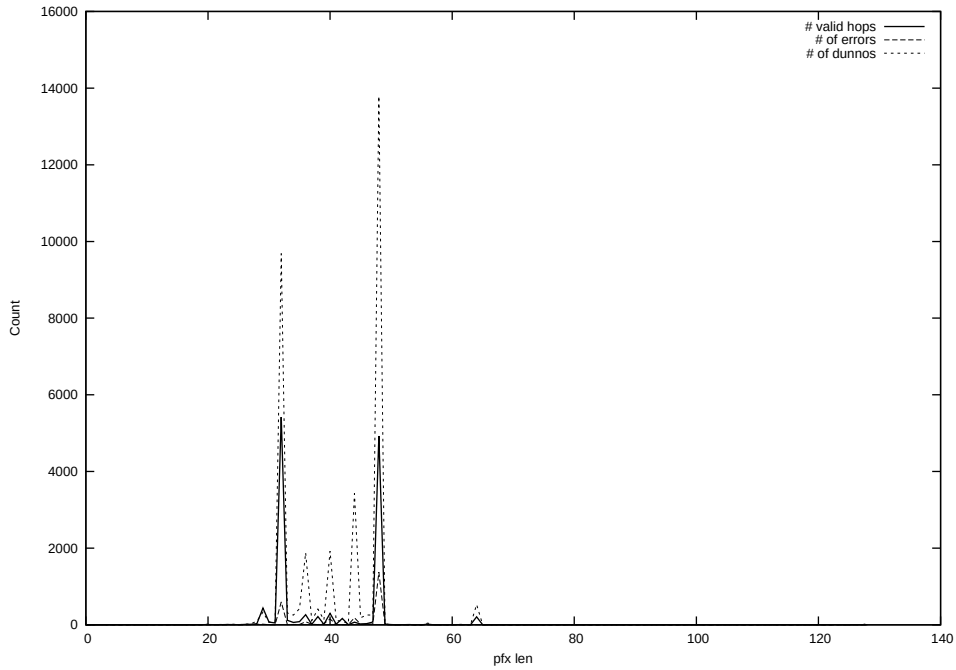
2013-11-21



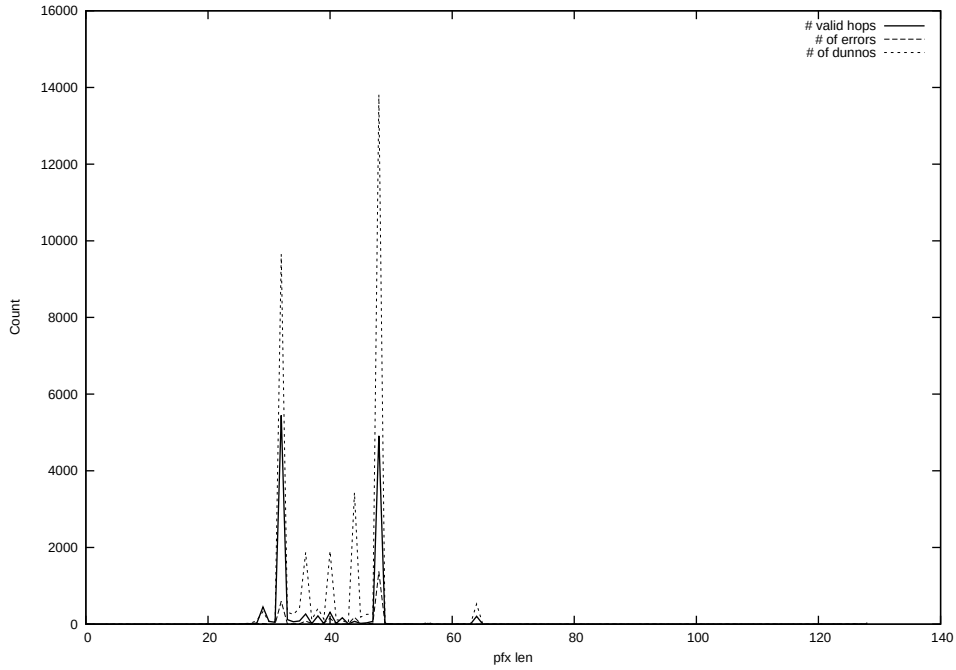
2013-11-22



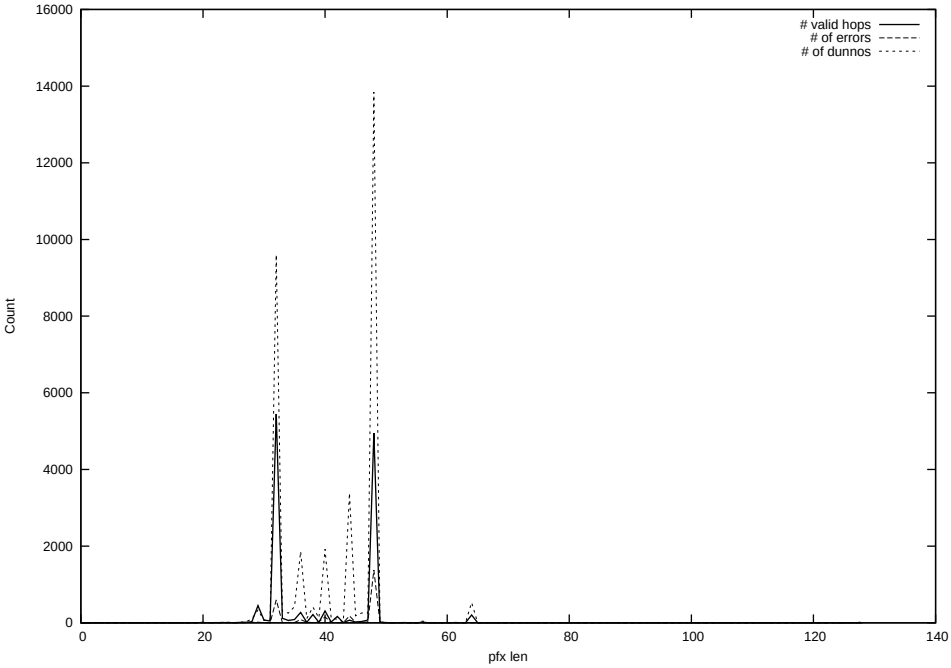
2013-11-23



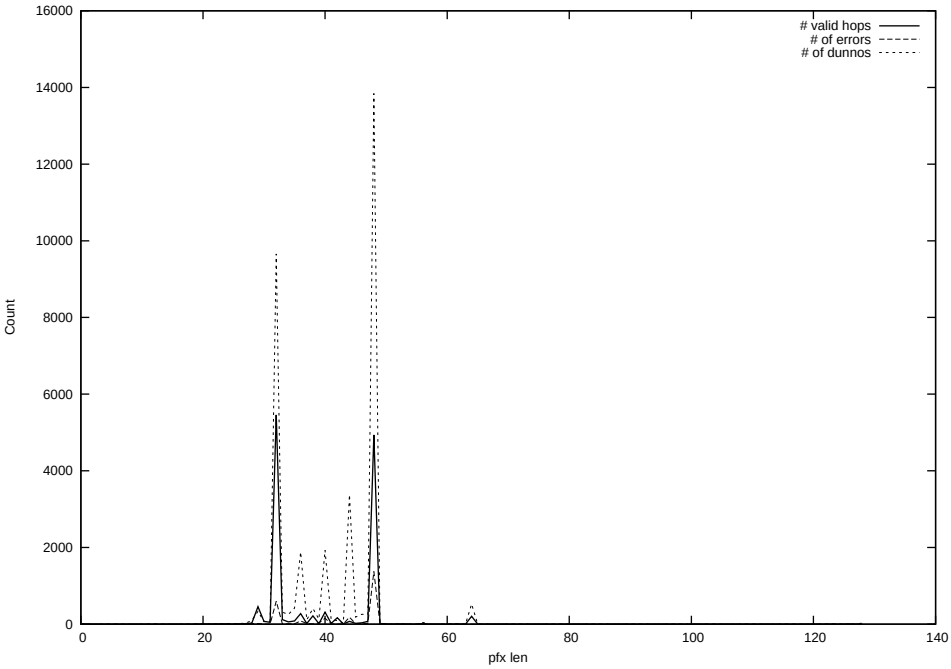
2013-11-24



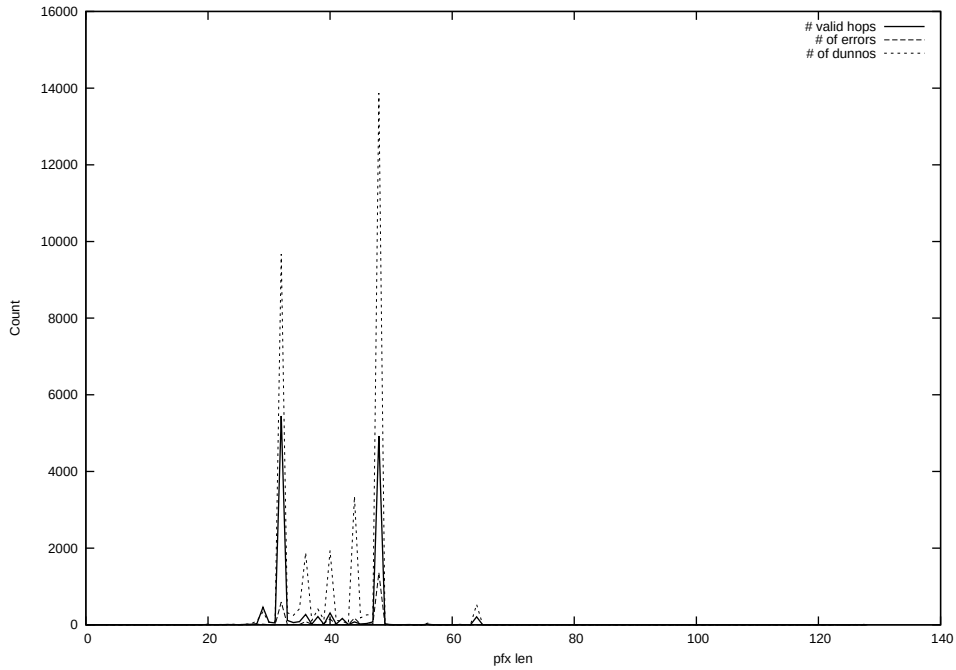
2013-11-25



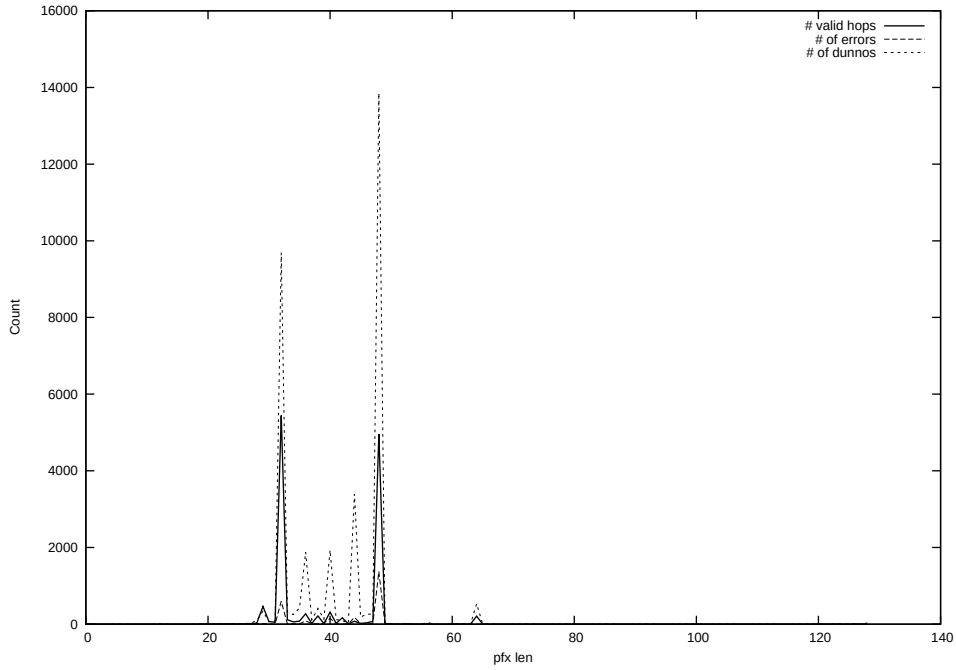
2013-11-26



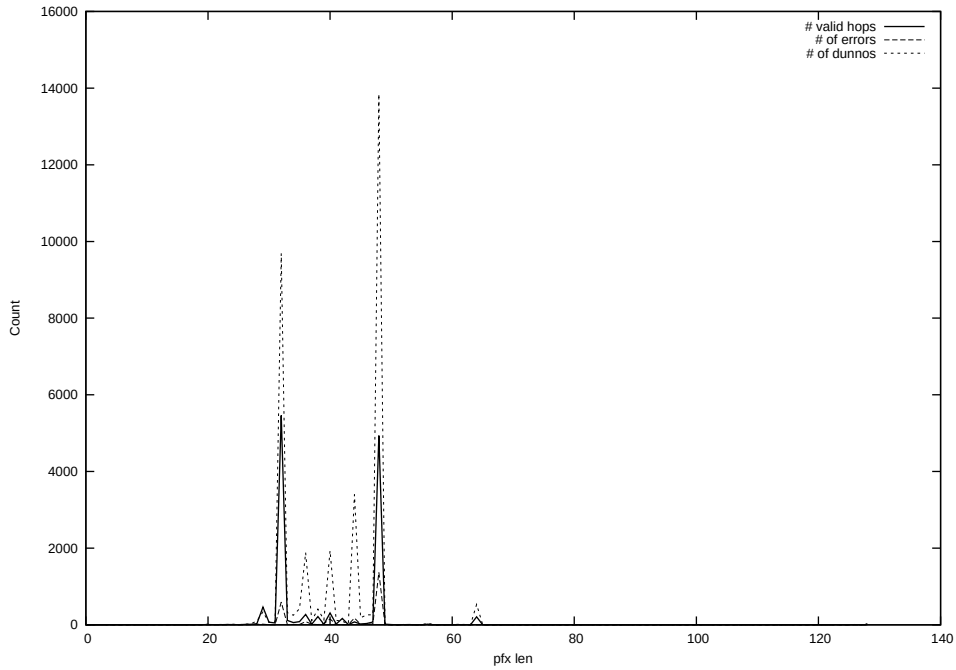
2013-11-27



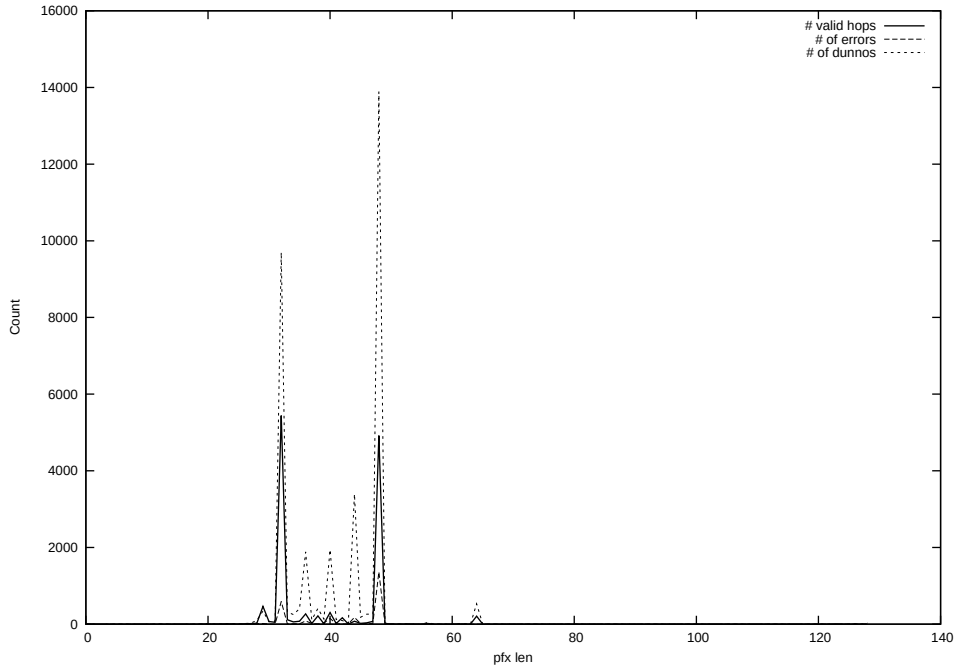
2013-11-28



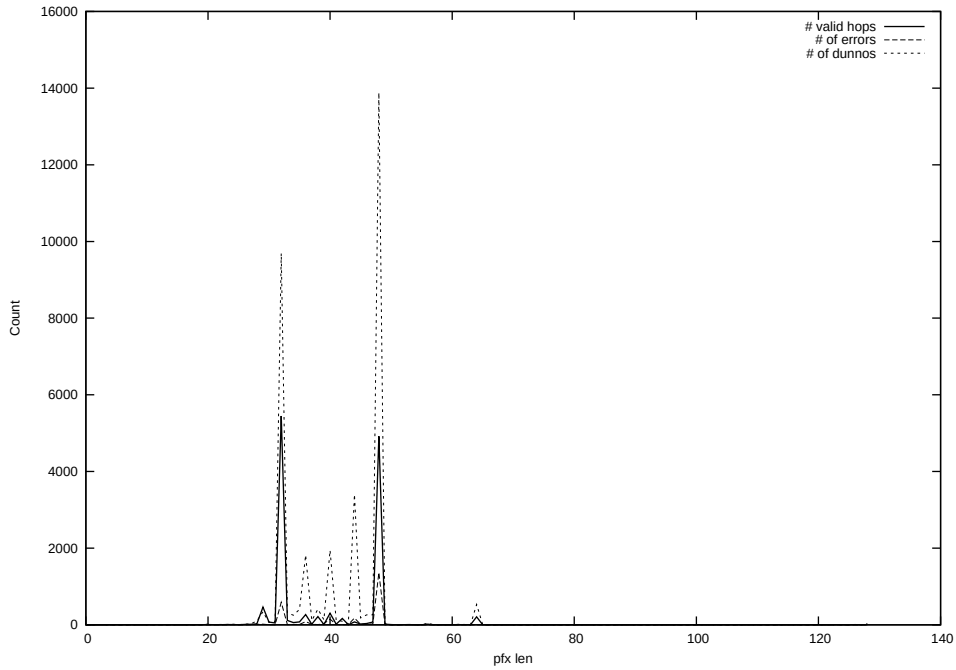
2013-11-29



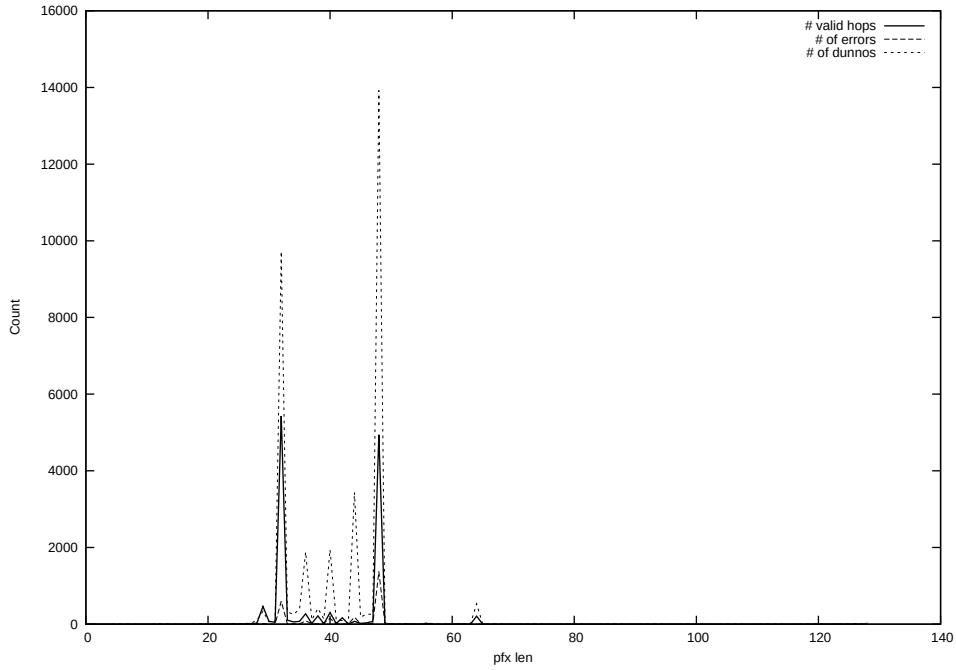
2013-11-30



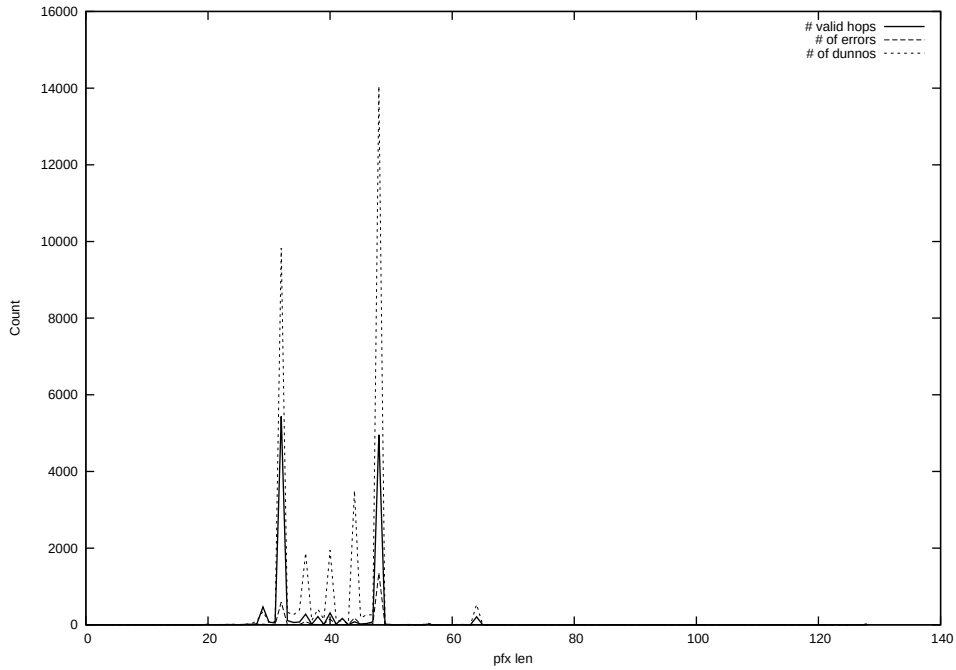
2013-12-01



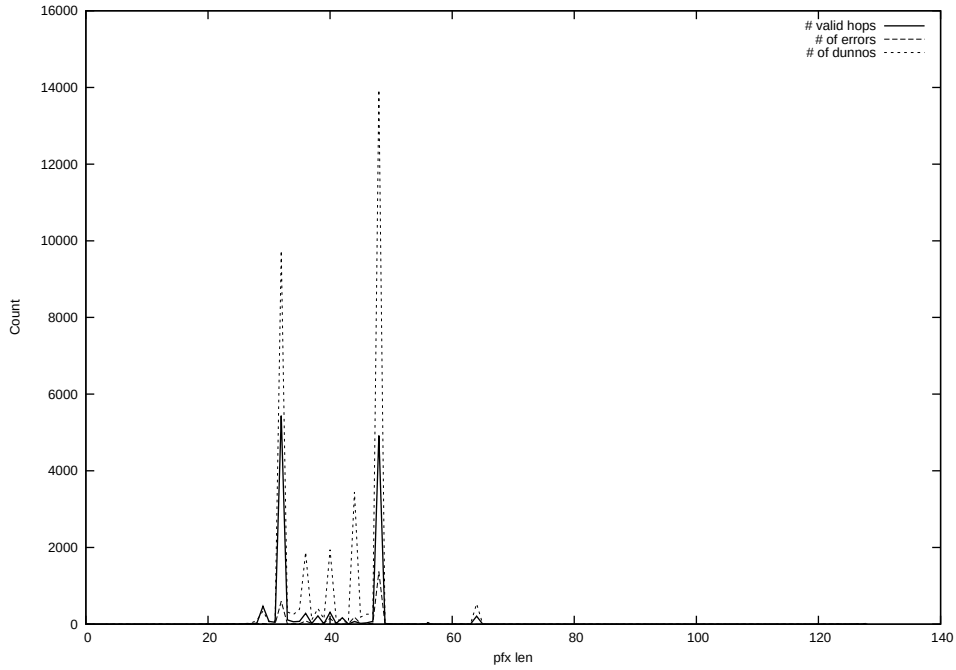
2013-12-02



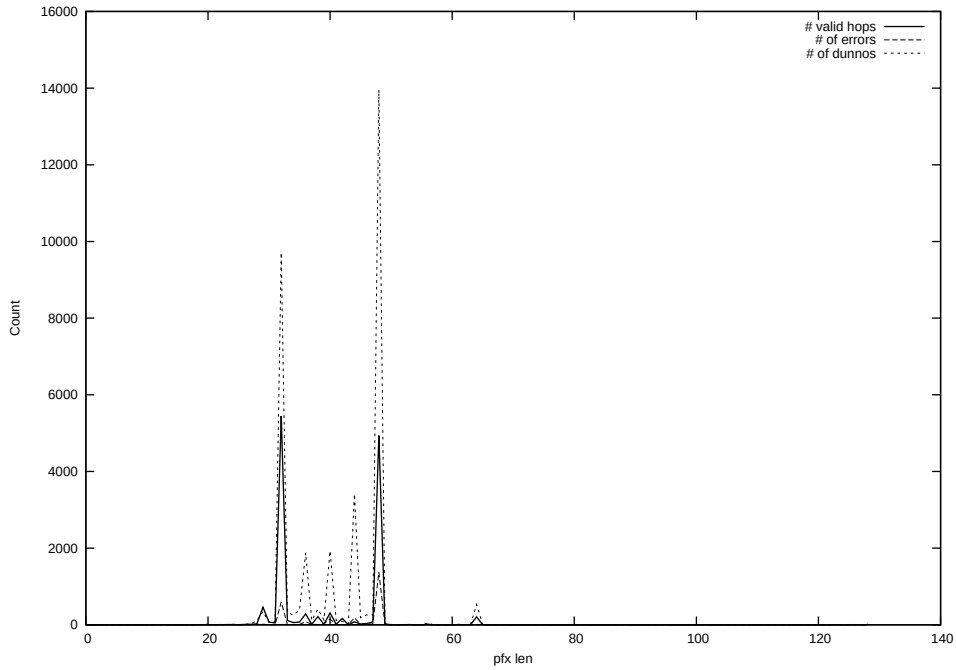
2013-12-03



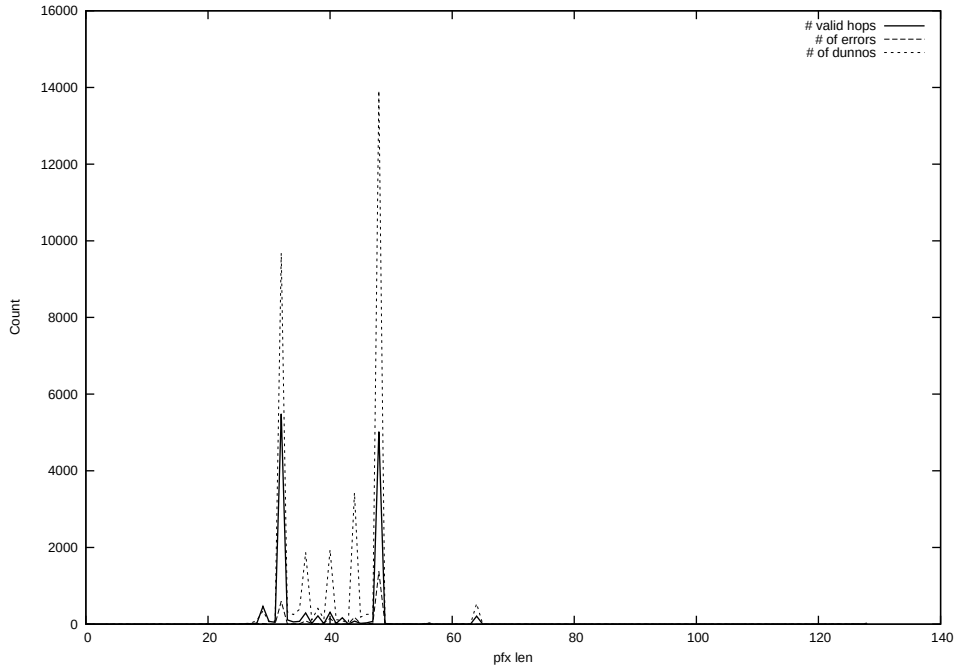
2013-12-04



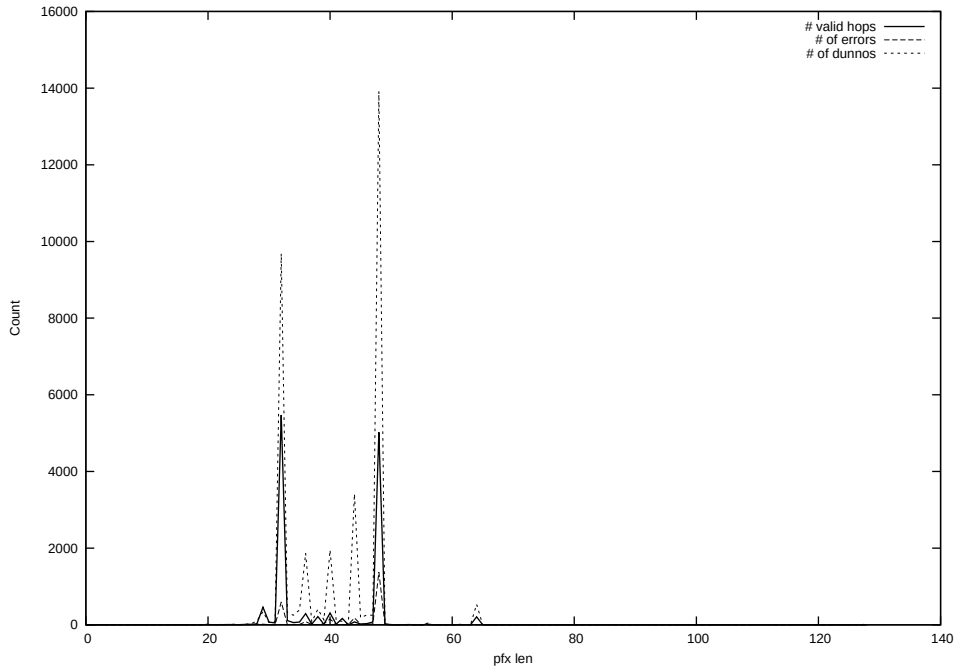
2013-12-05



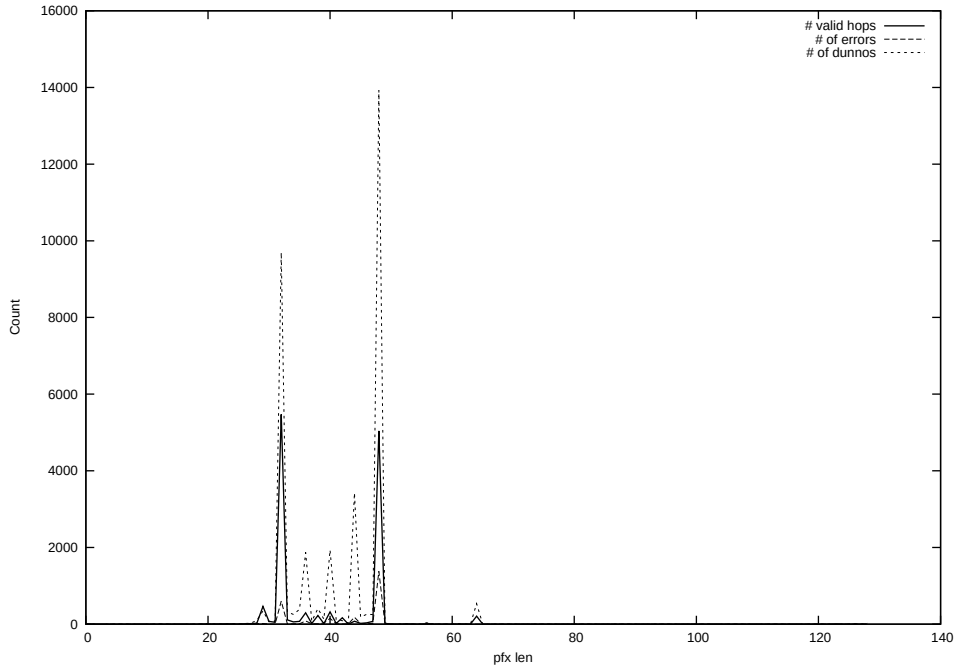
2013-12-06



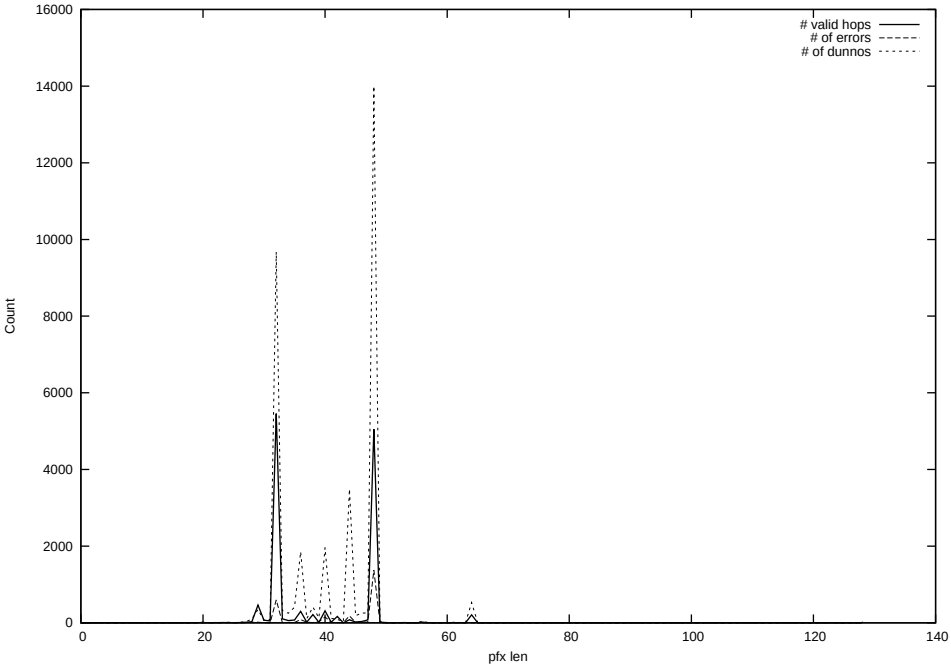
2013-12-07



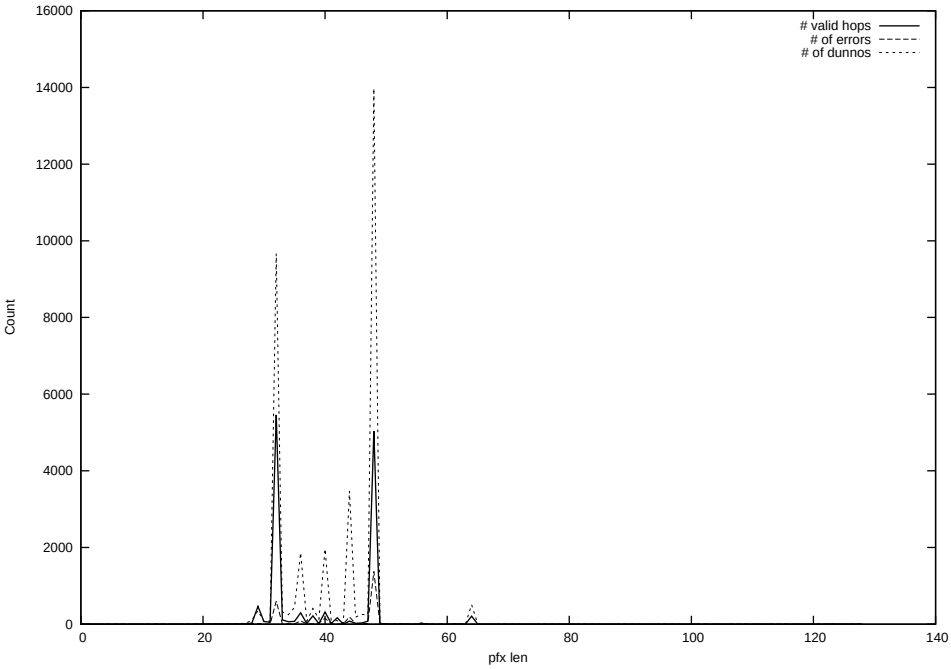
2013-12-08



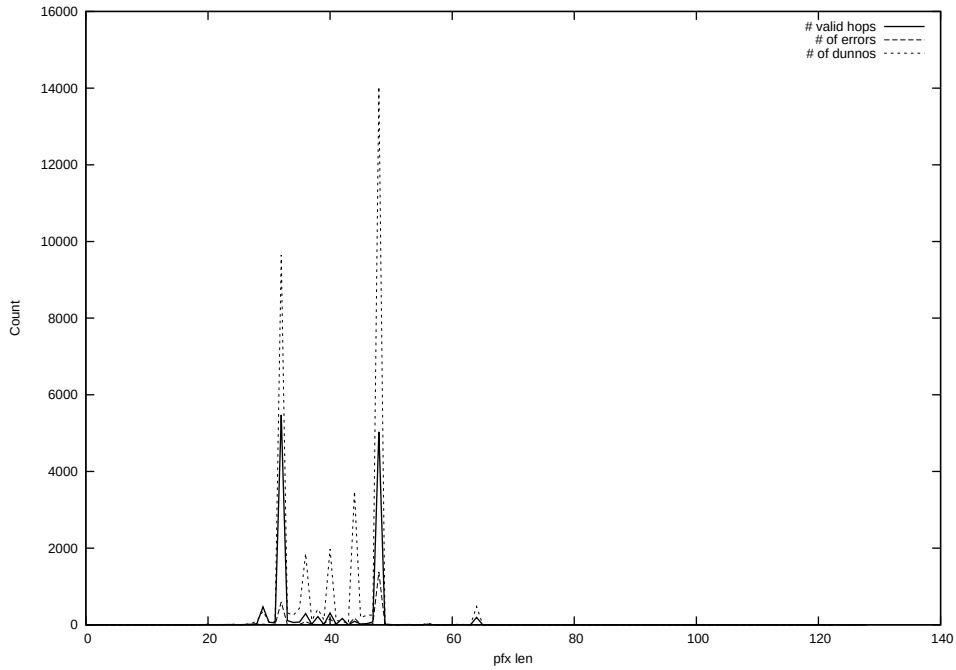
2013-12-09



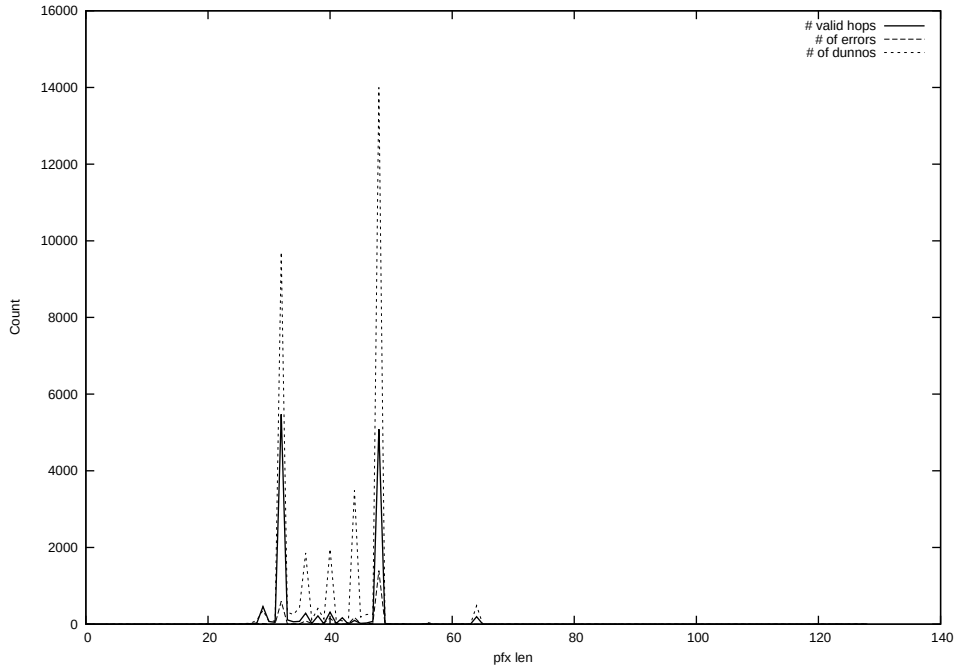
2013-12-10



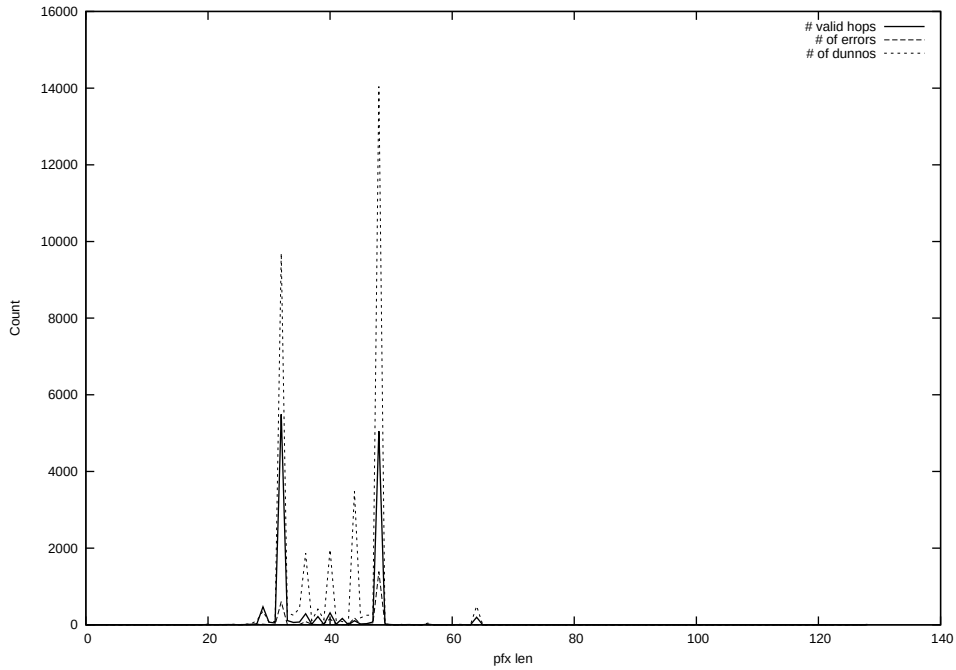
2013-12-11



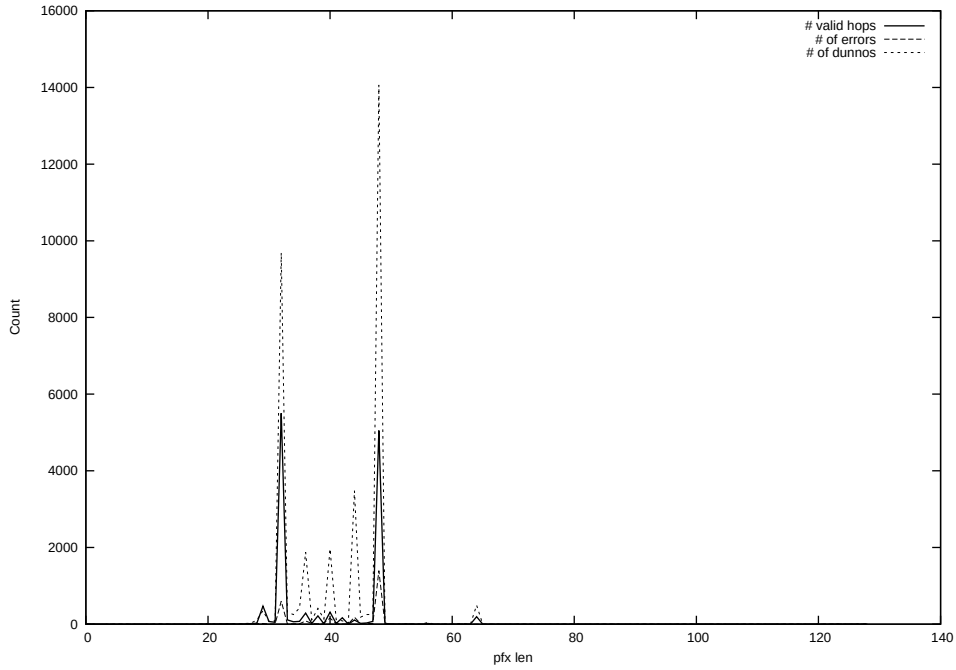
2013-12-12



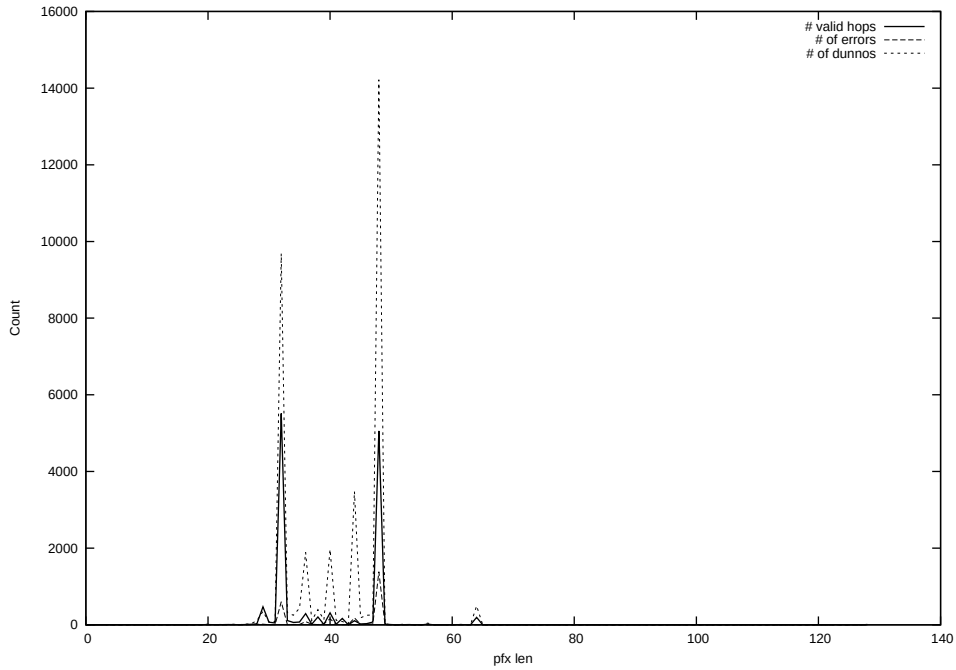
2013-12-13



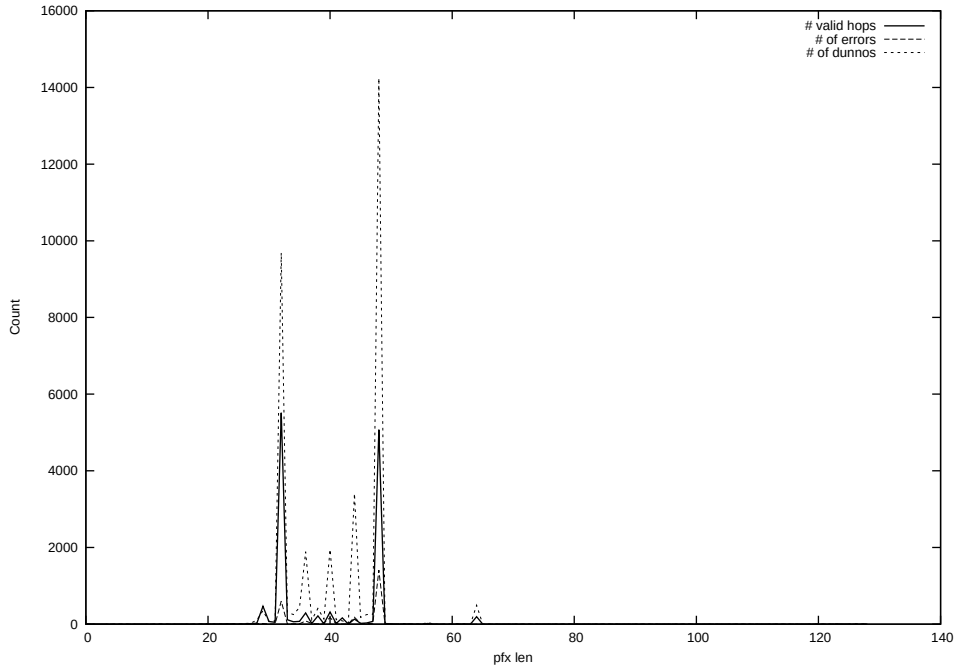
2013-12-14



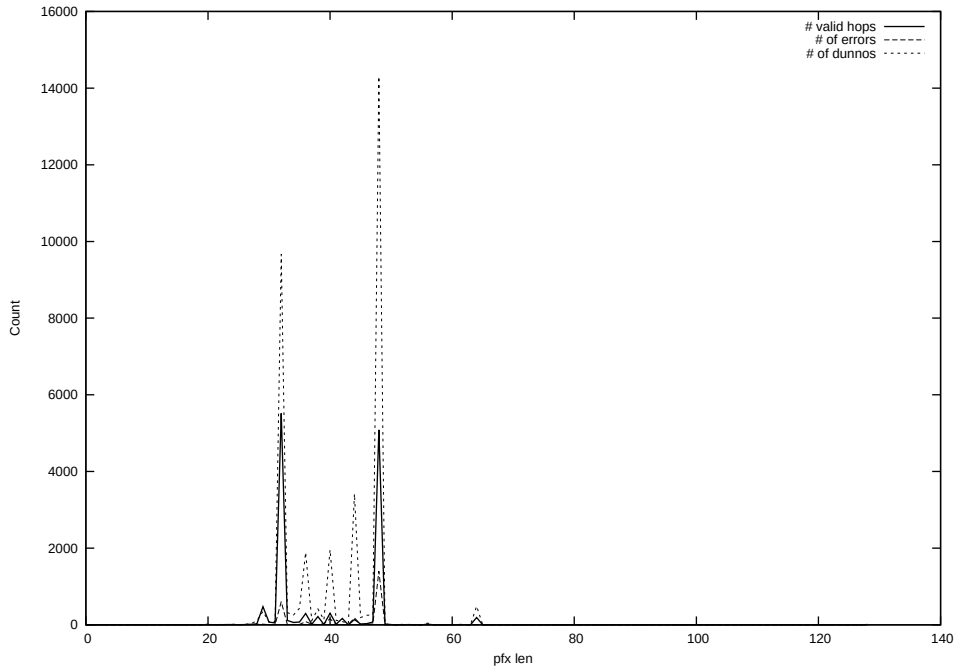
2013-12-15



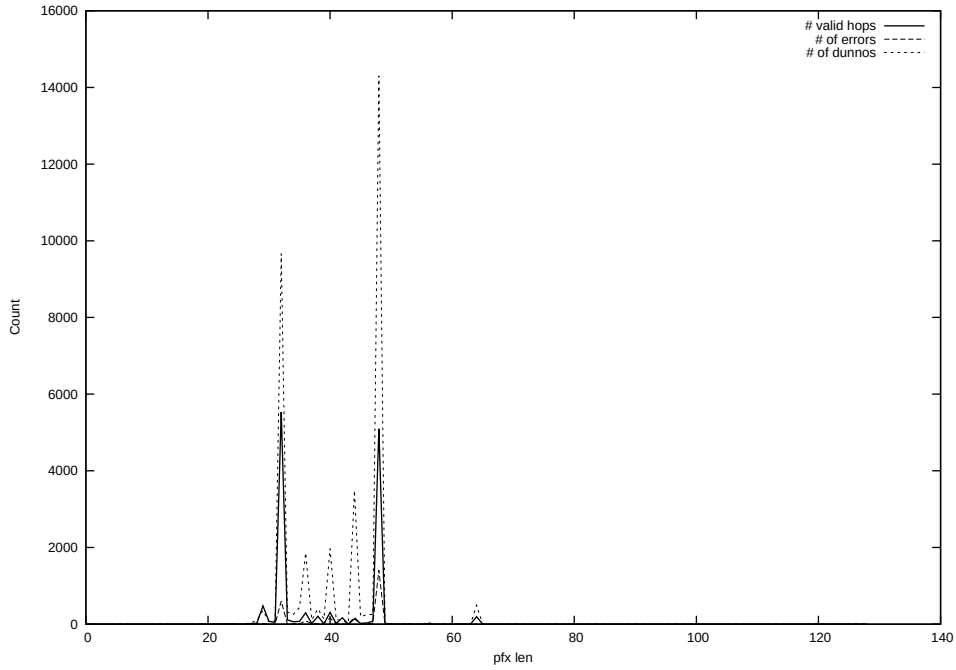
2013-12-16



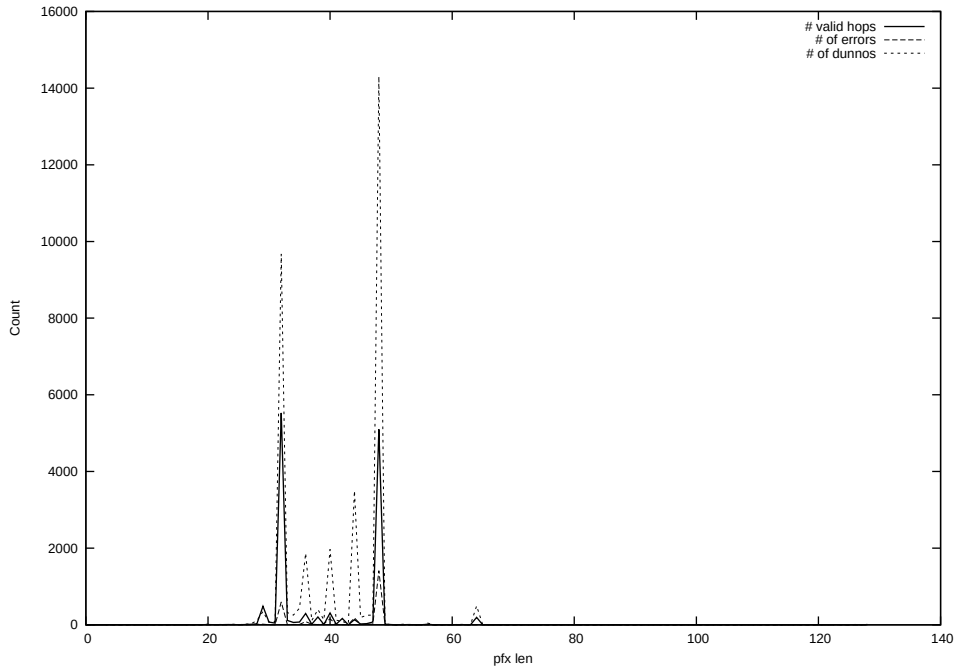
2013-12-17



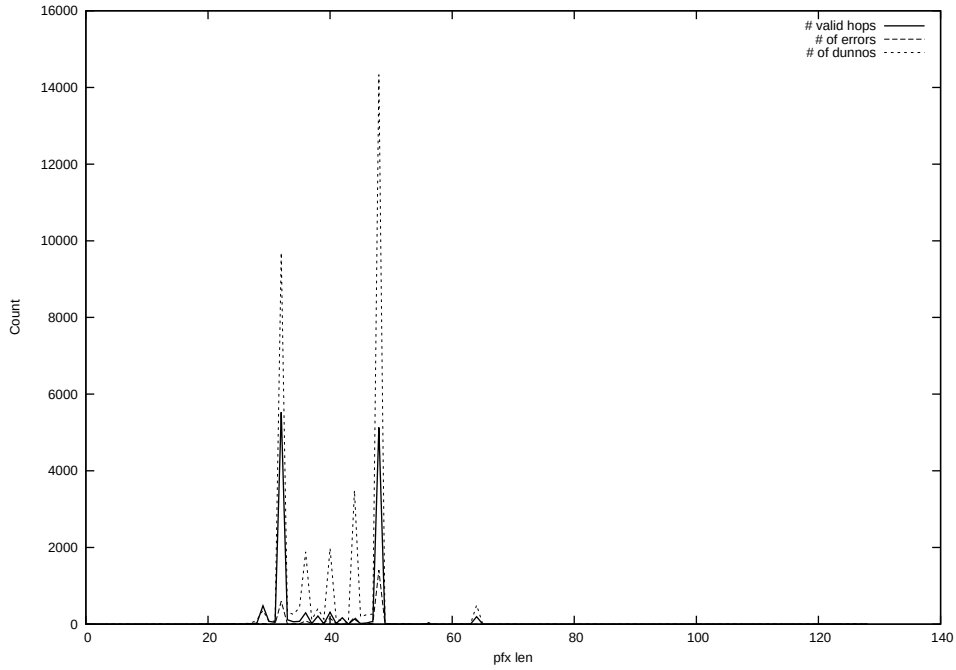
2013-12-18



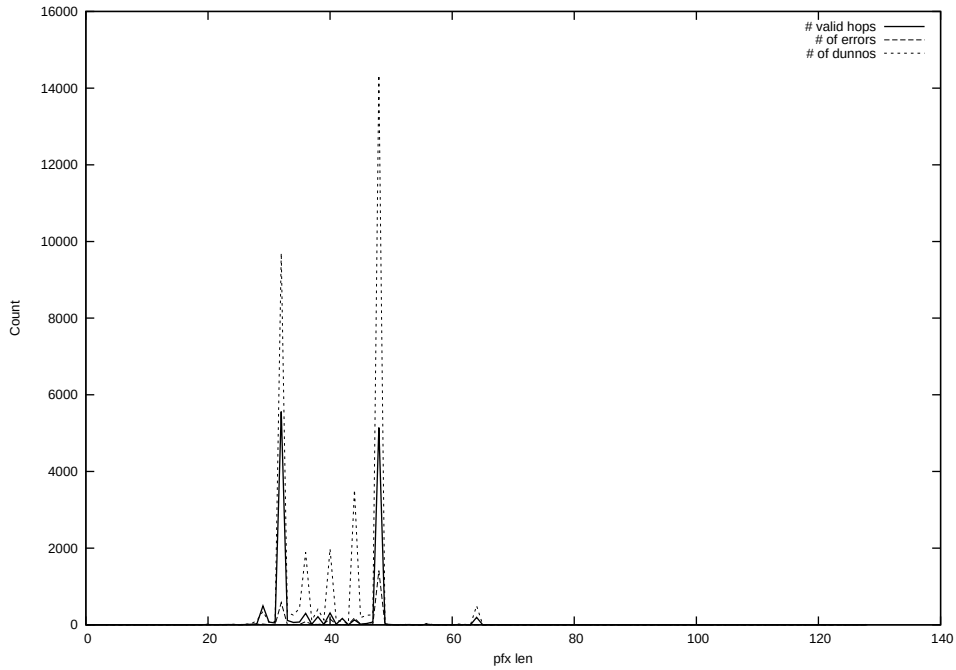
2013-12-19



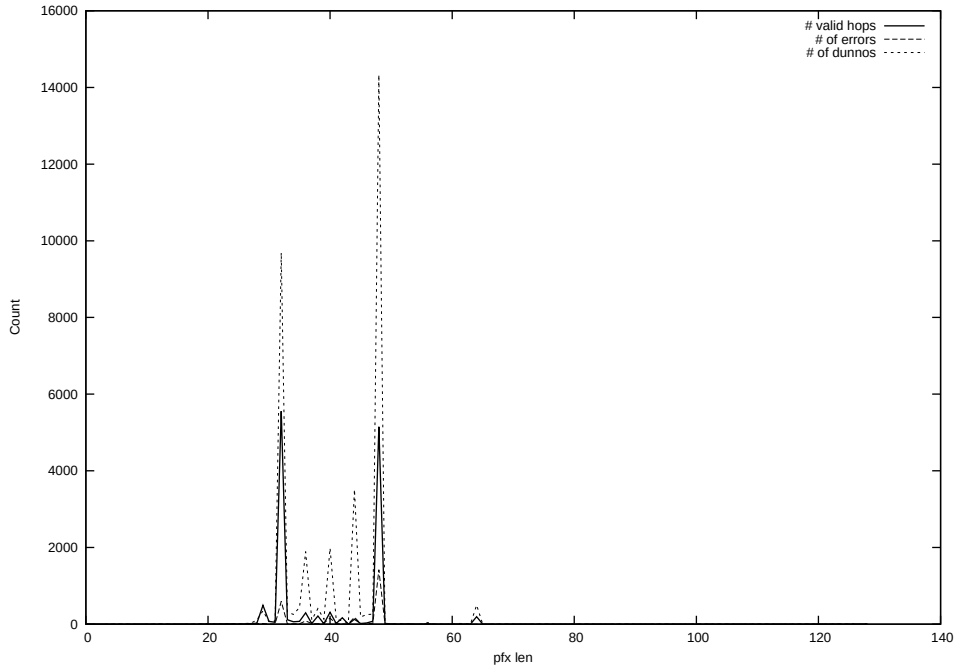
2013-12-20



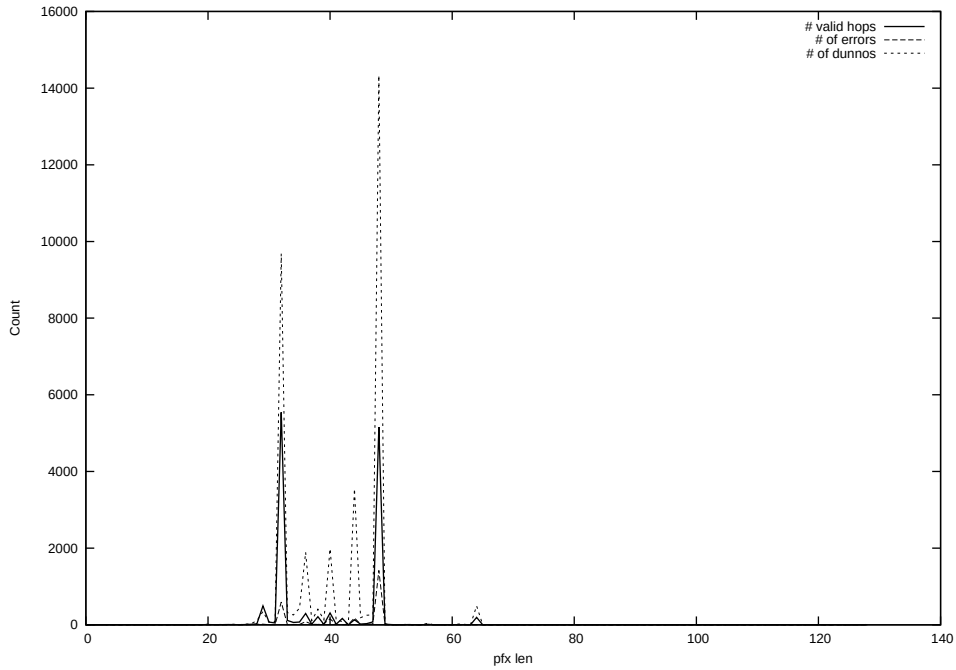
2013-12-21



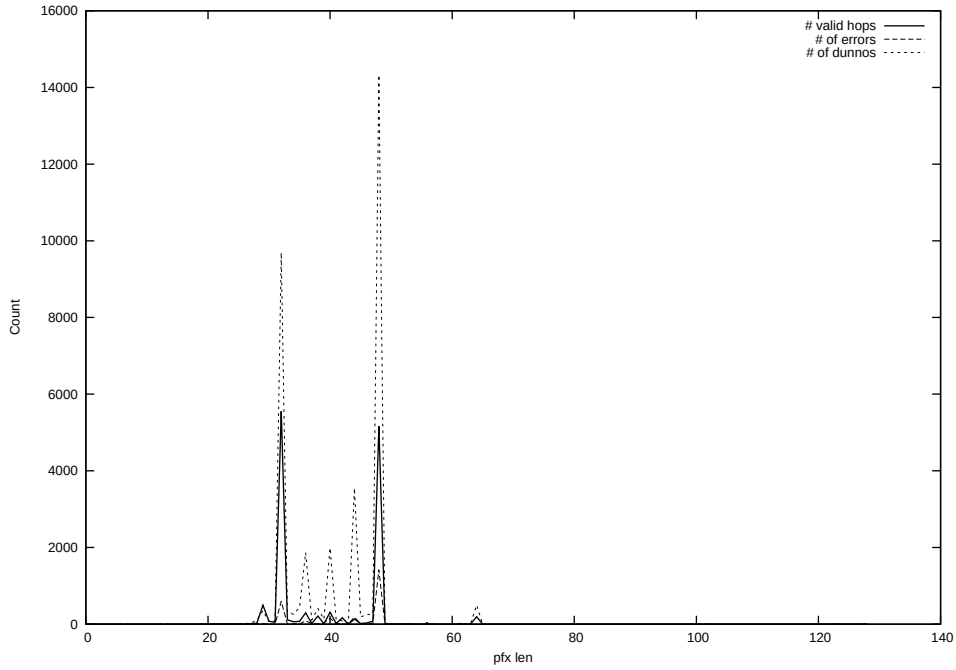
2013-12-22



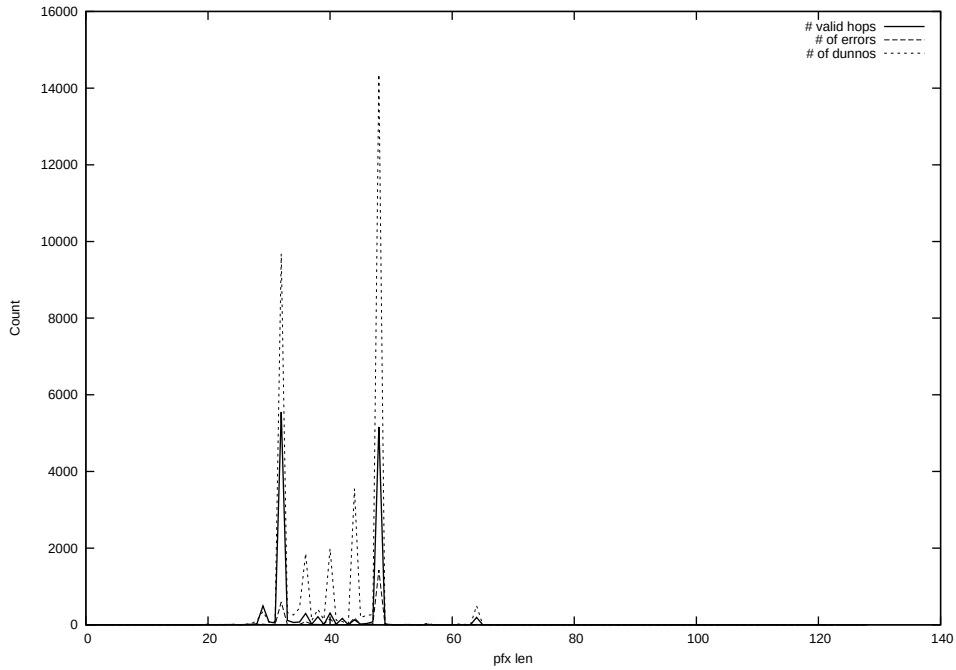
2013-12-23



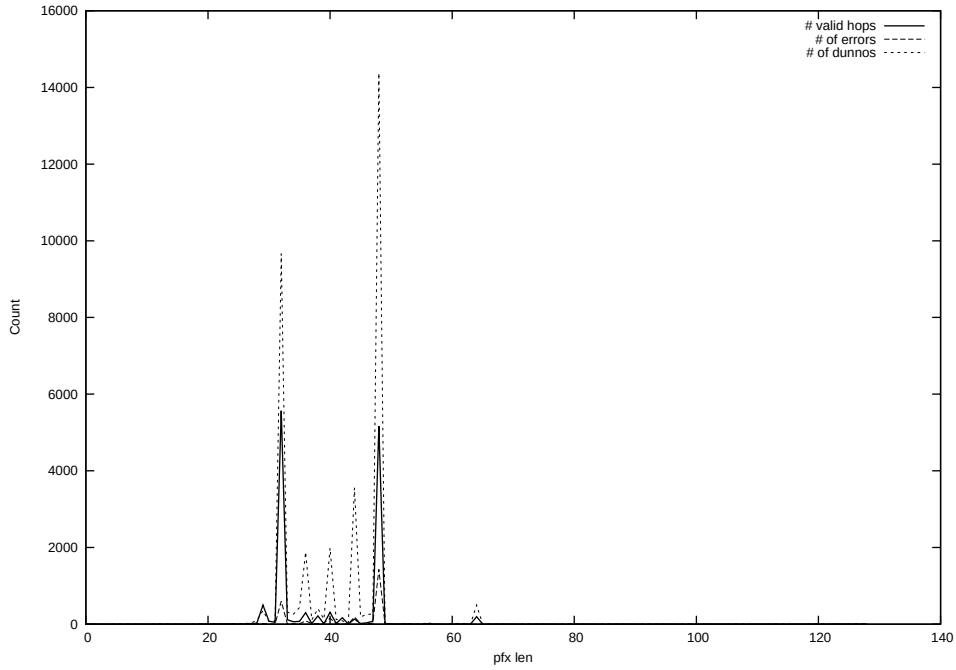
2013-12-24



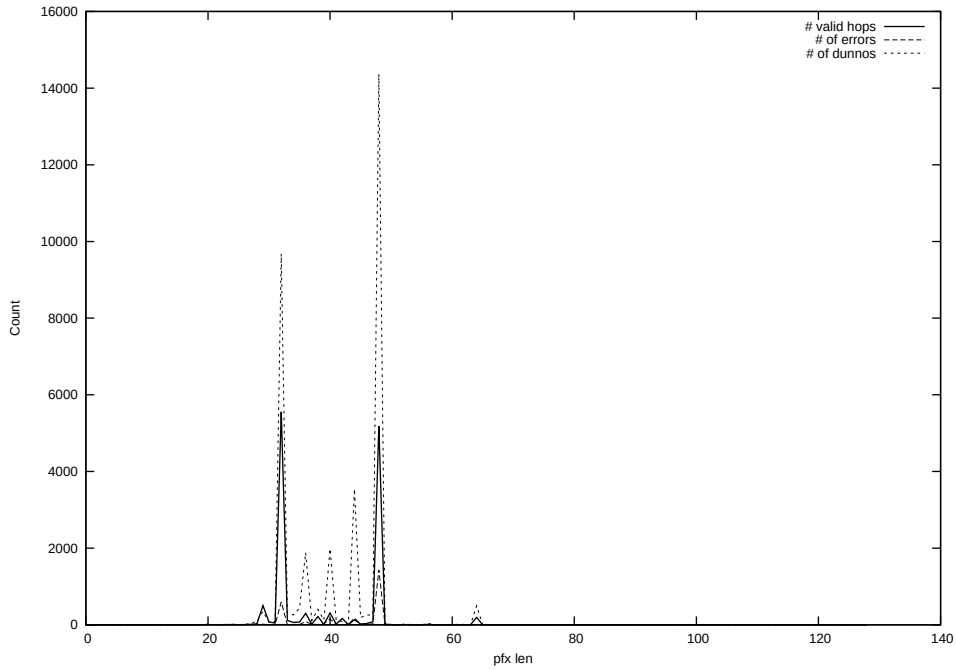
2013-12-25



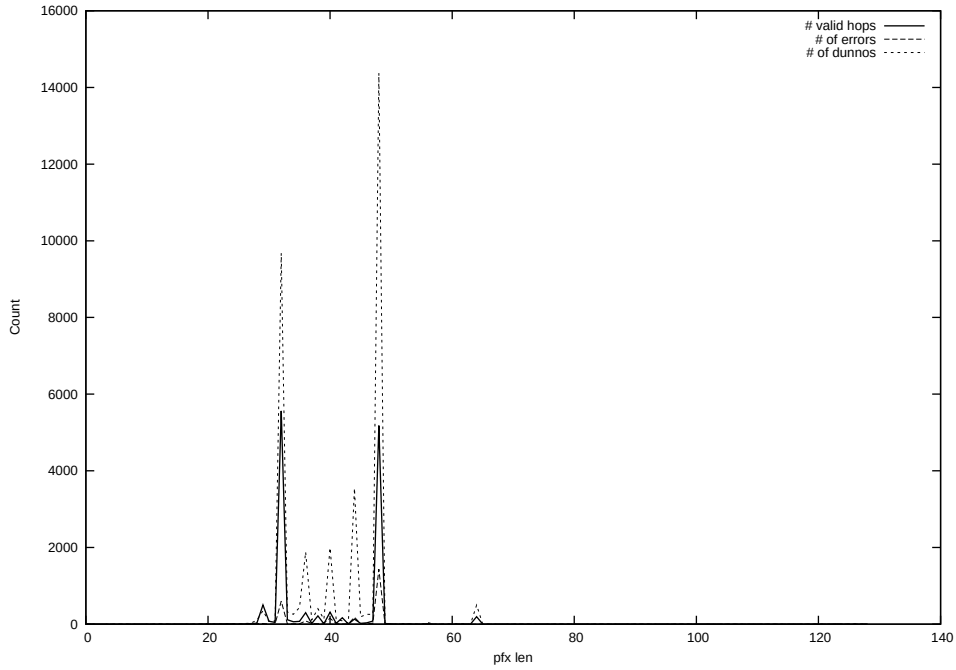
2013-12-26



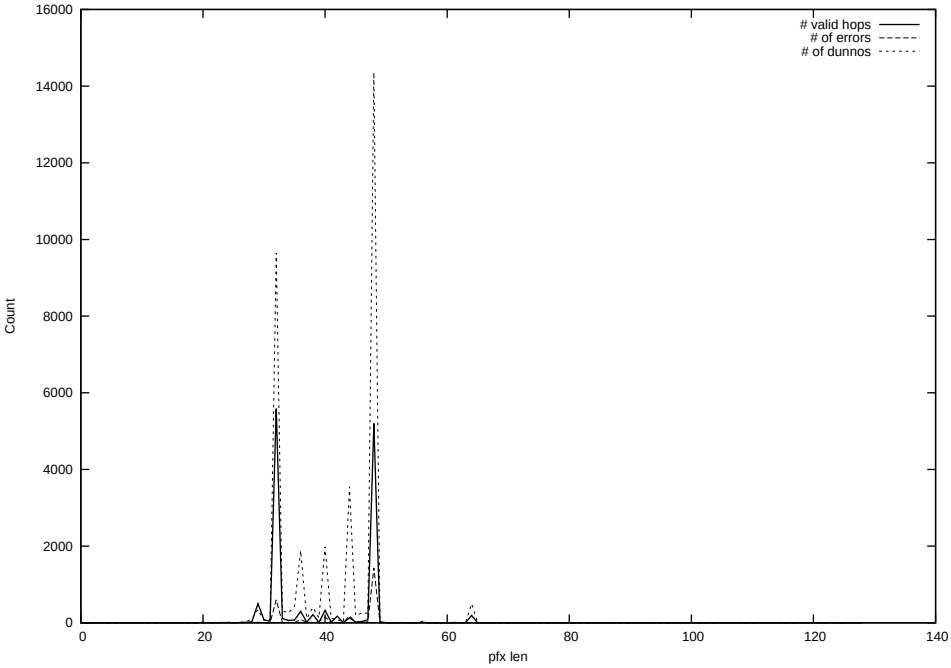
2013-12-27



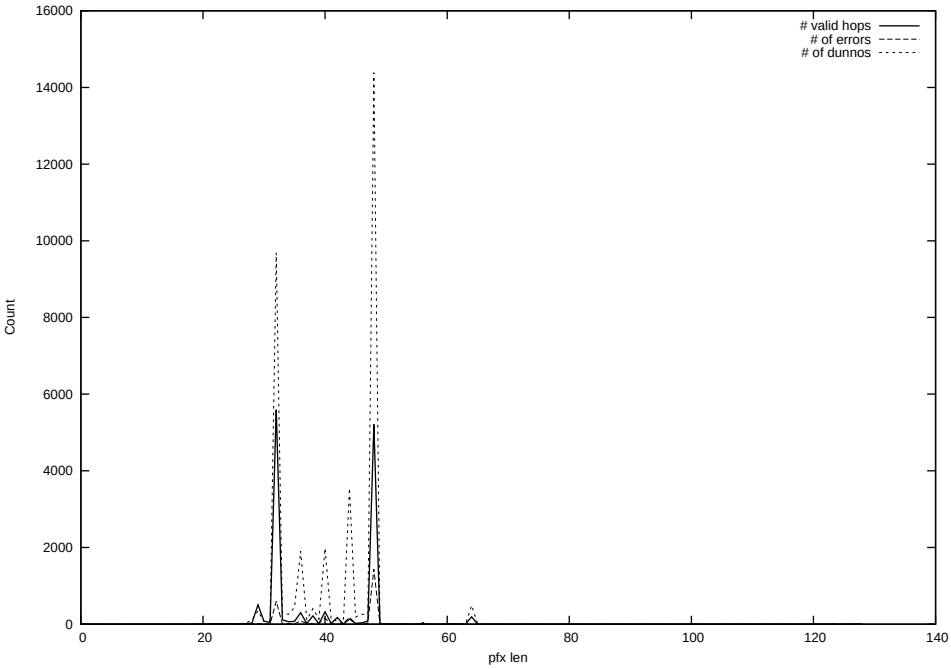
2013-12-28



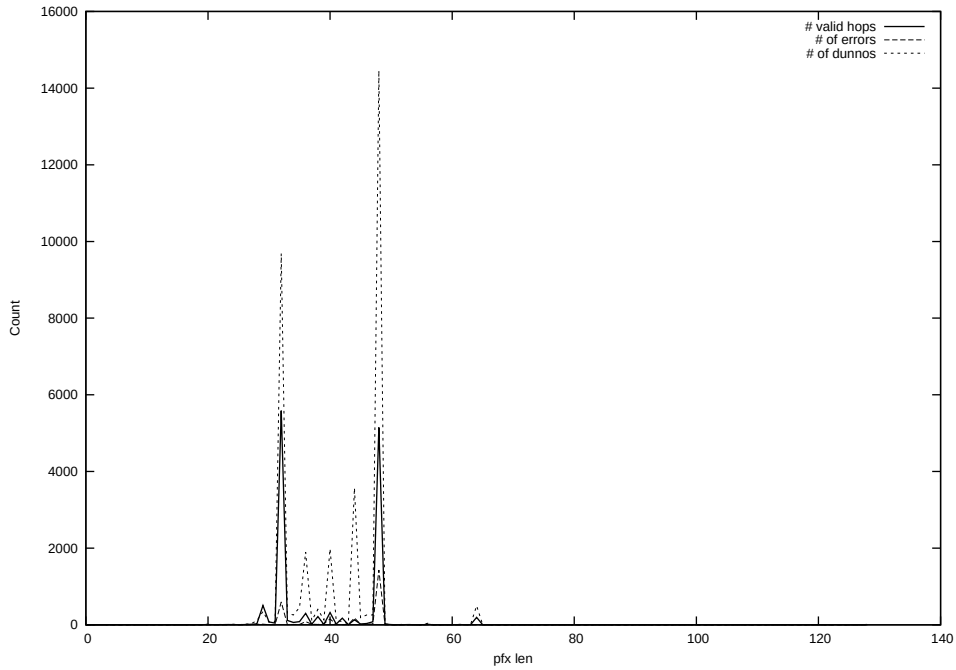
2013-12-29



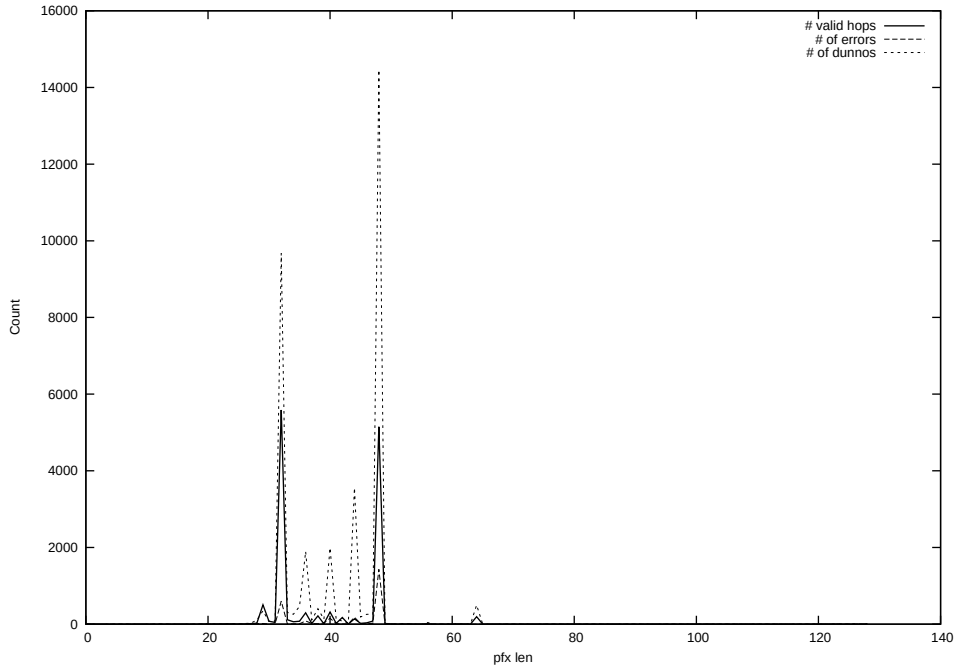
2013-12-30



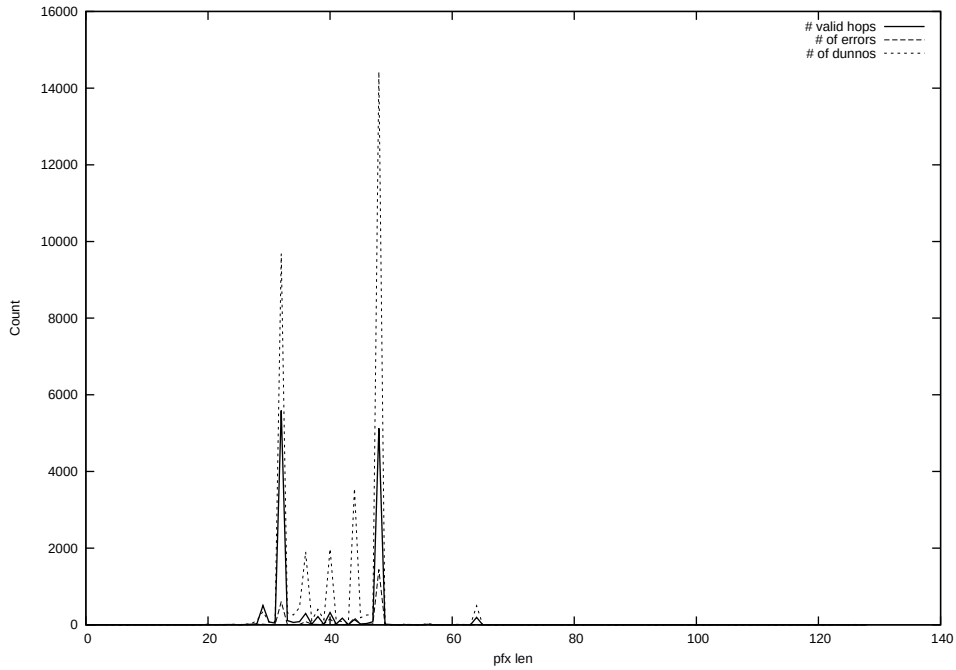
2013-12-31



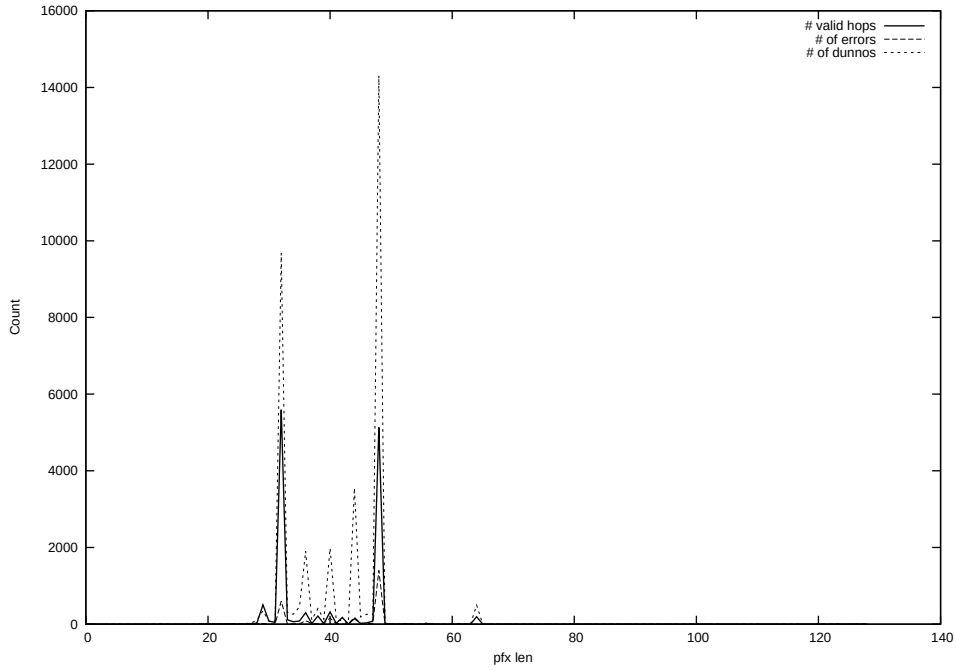
2014-01-01



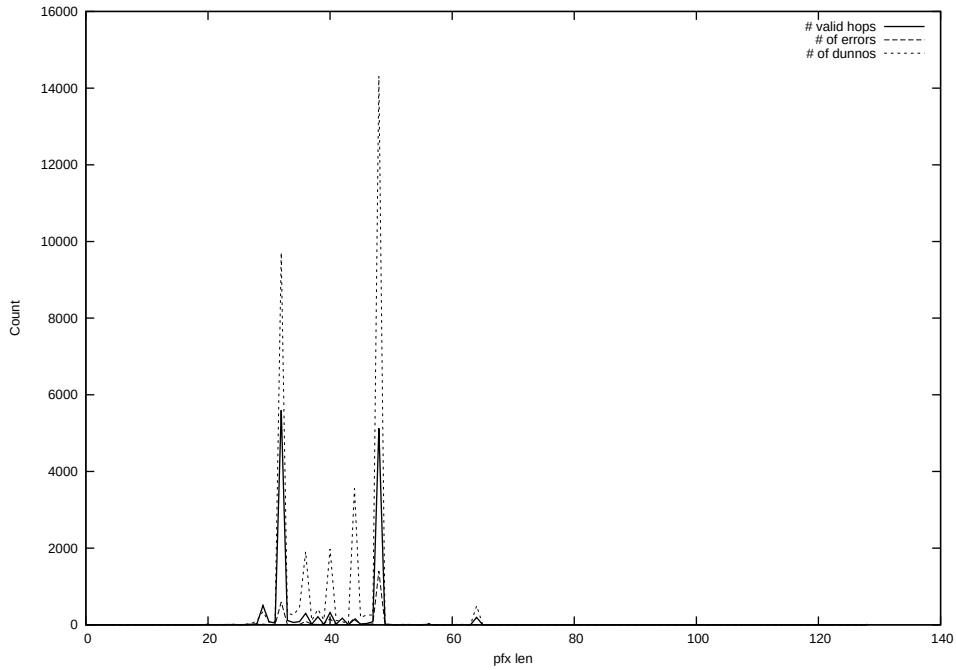
2014-01-02



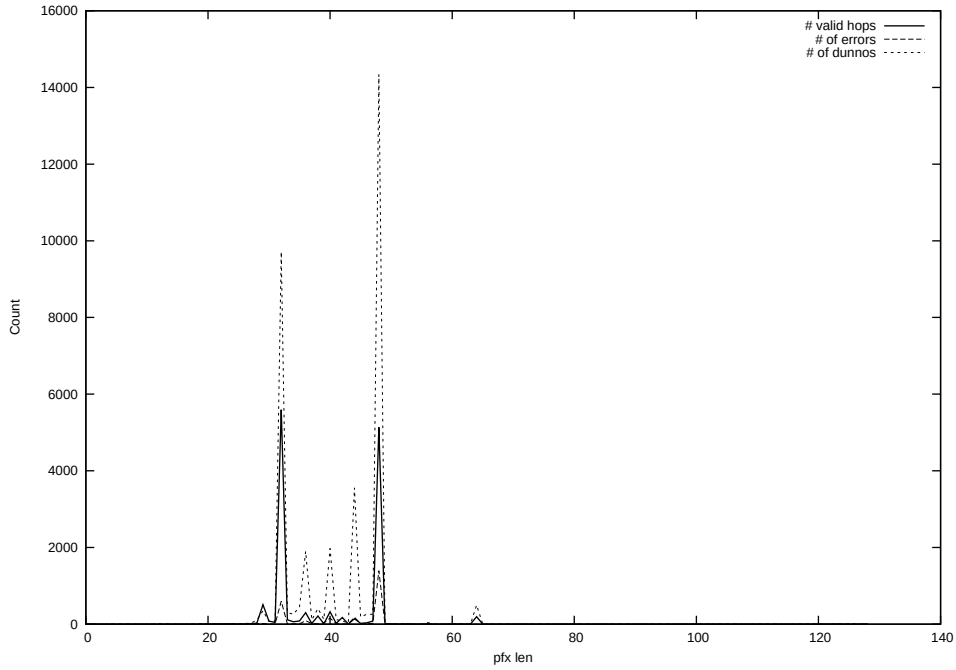
2014-01-03



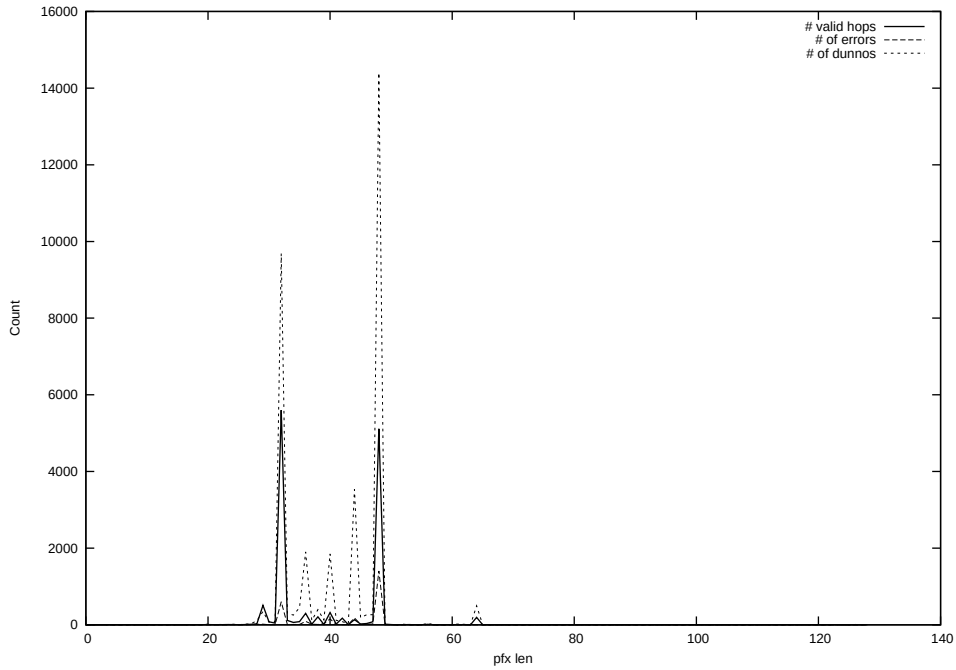
2014-01-04



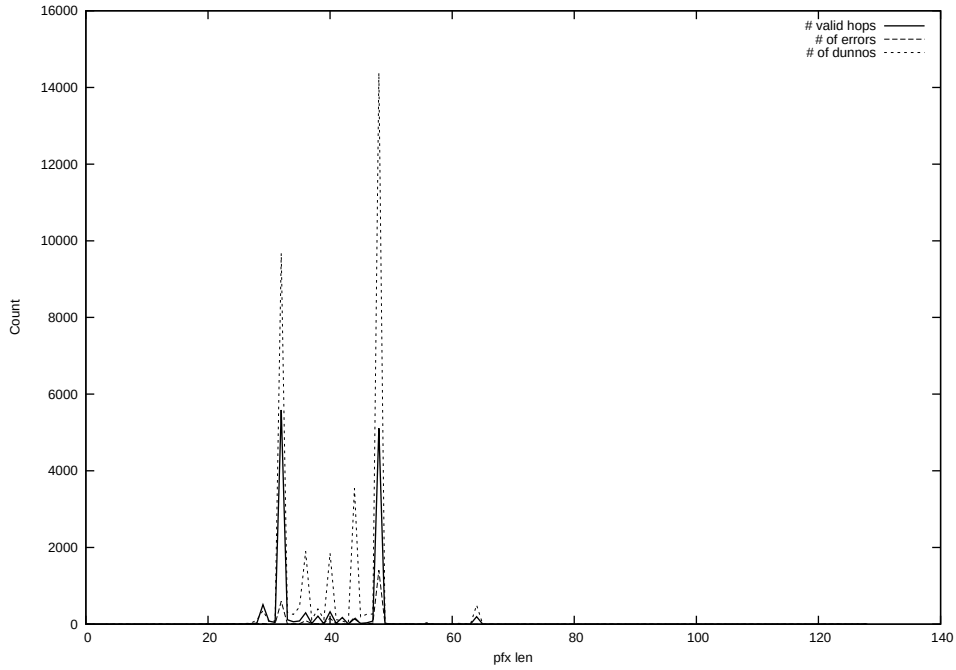
2014-01-05



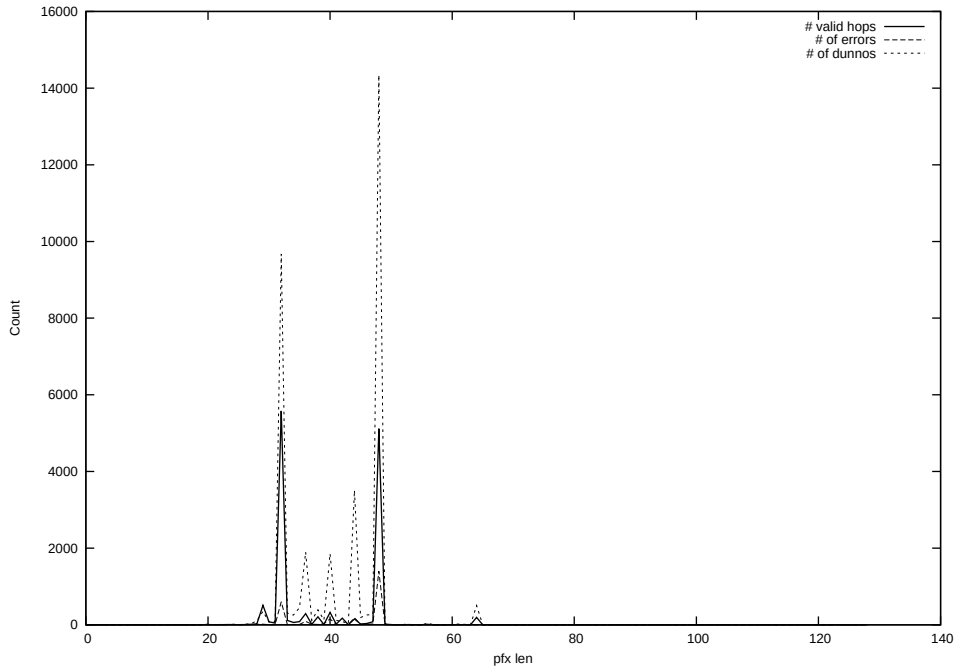
2014-01-06



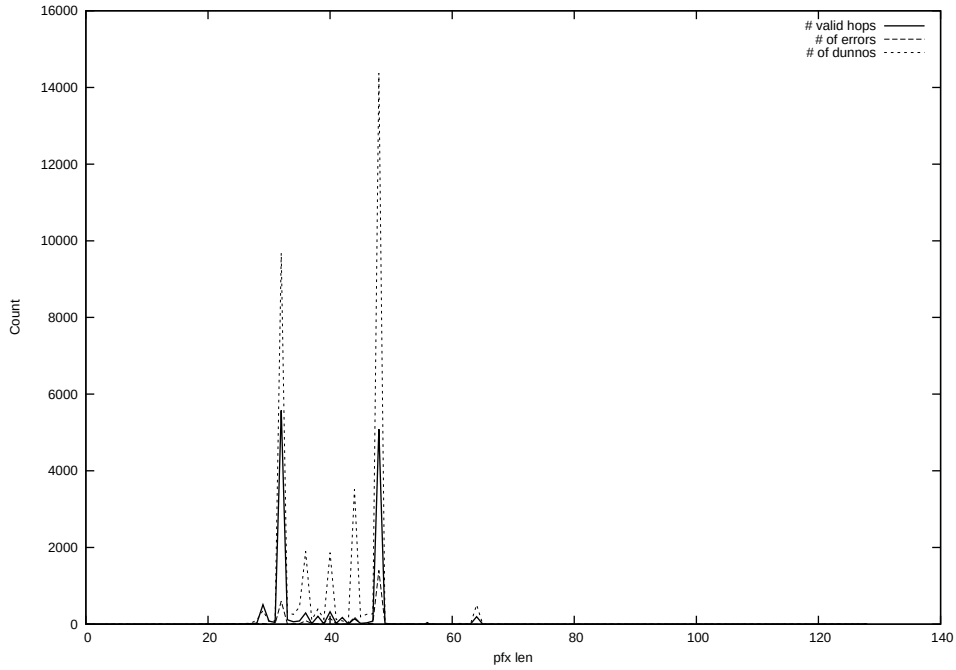
2014-01-07



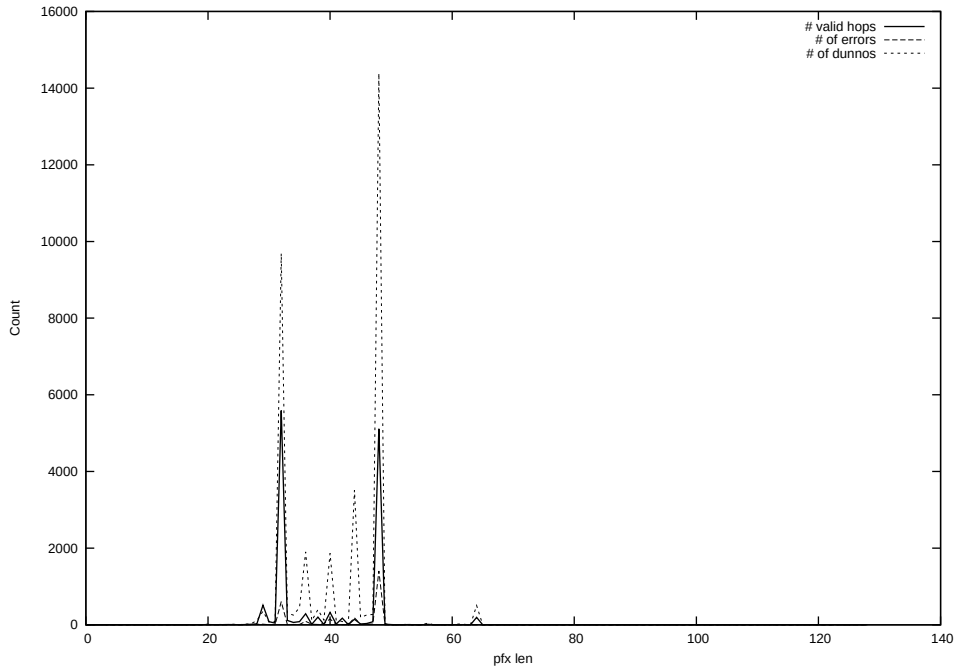
2014-01-08



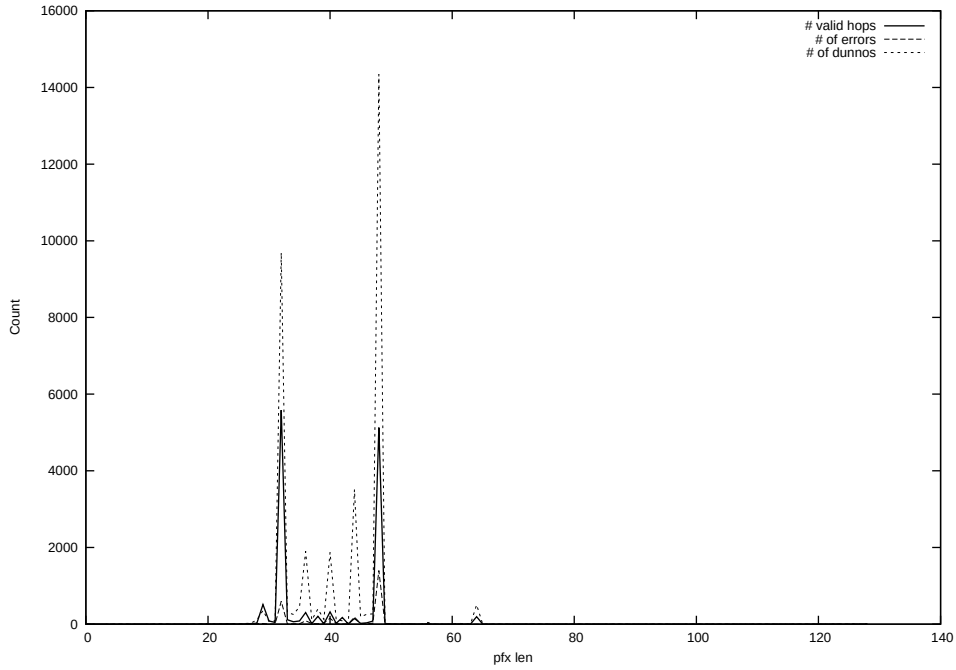
2014-01-09



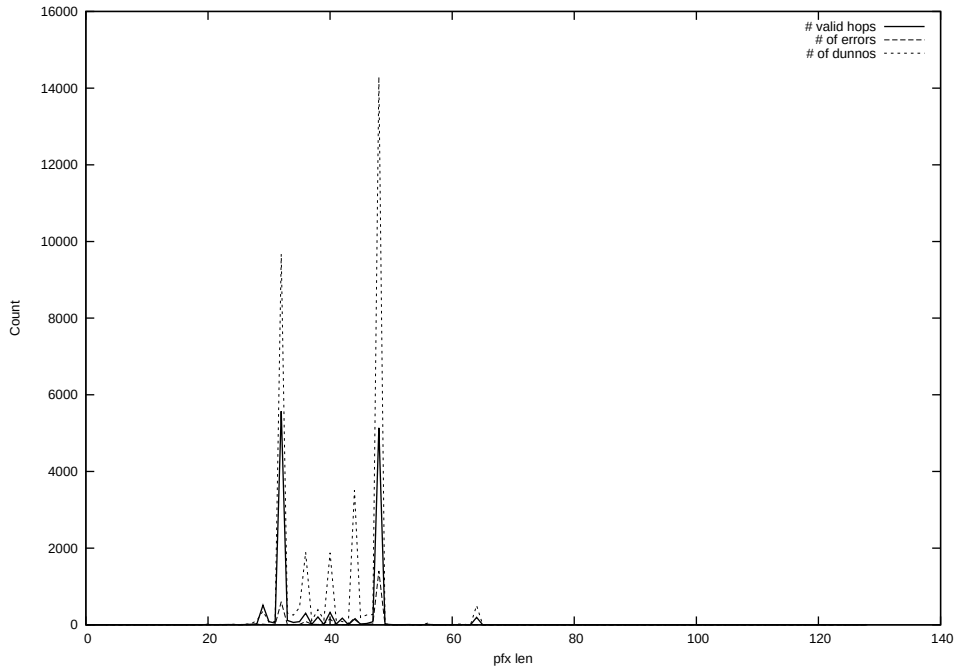
2014-01-10



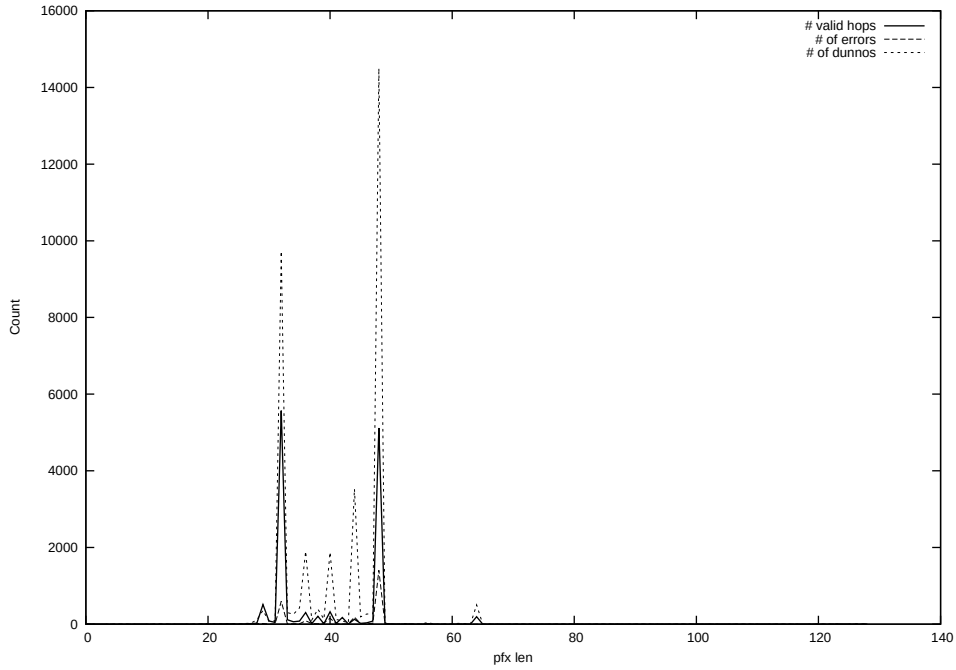
2014-01-11



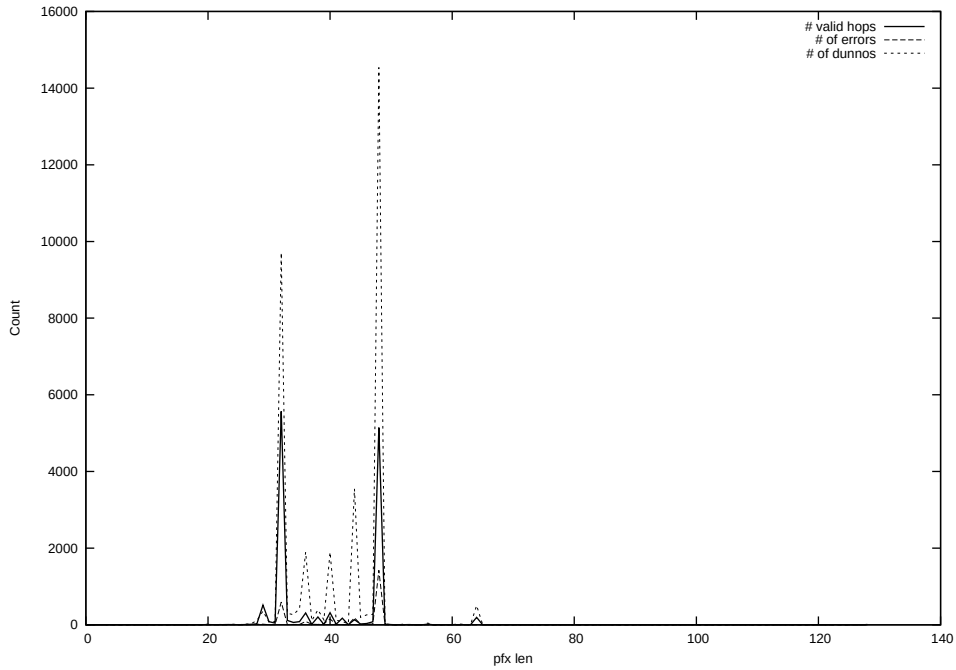
2014-01-12



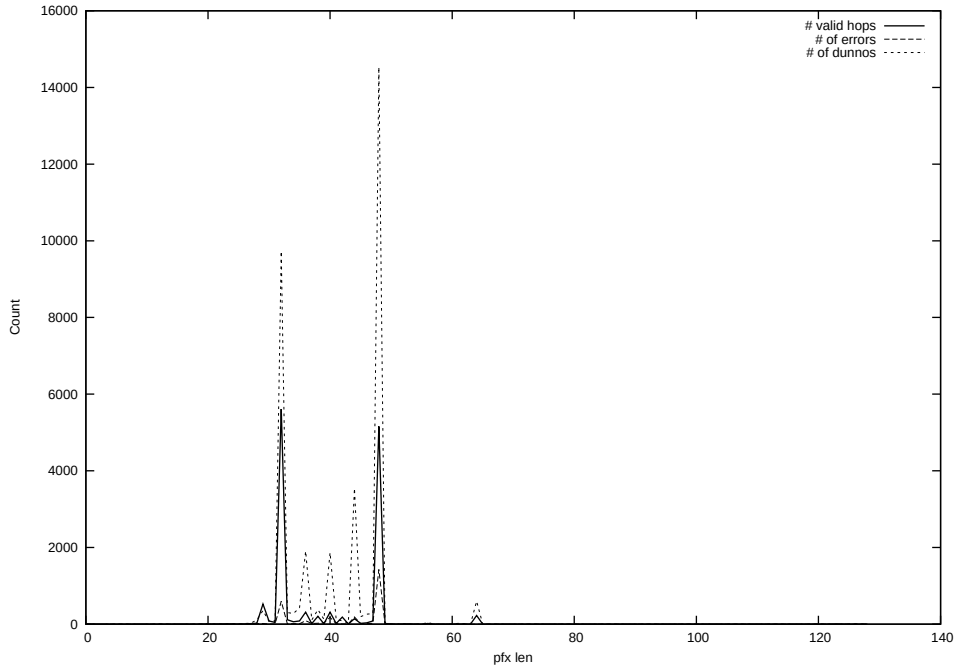
2014-01-13



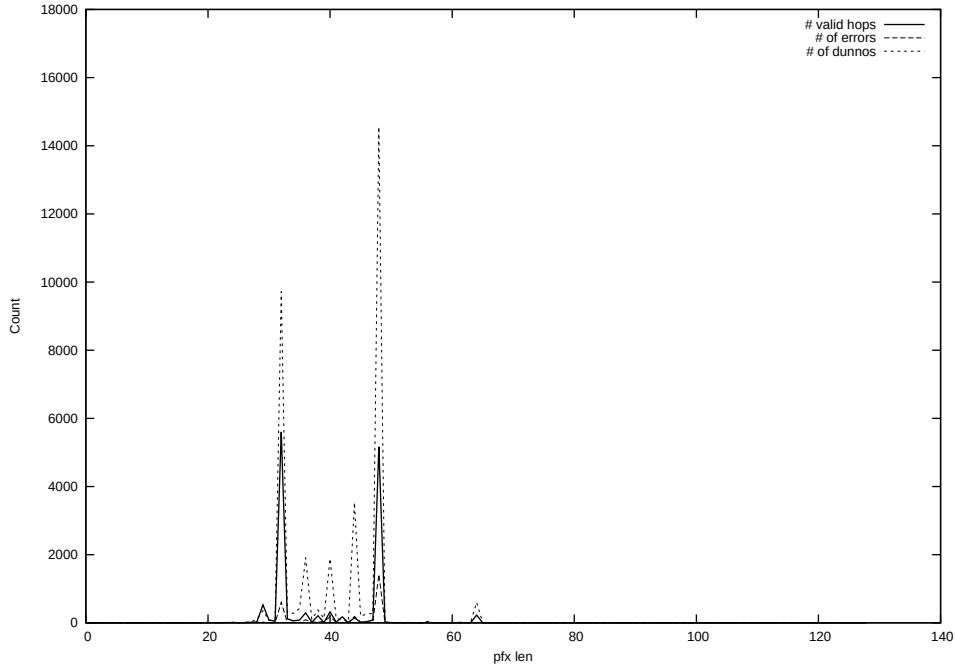
2014-01-14



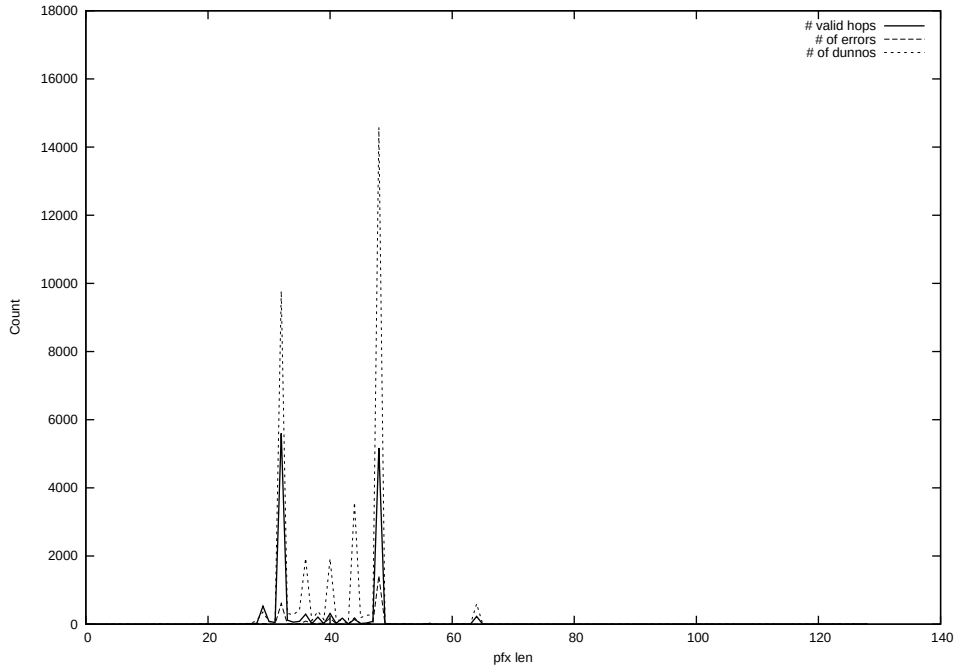
2014-01-15



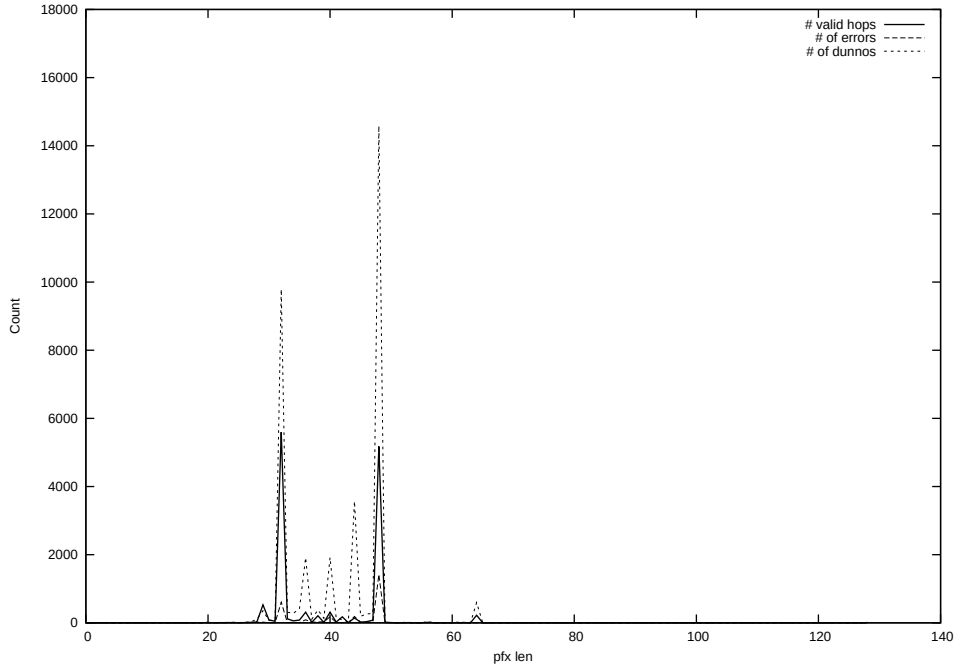
2014-01-16



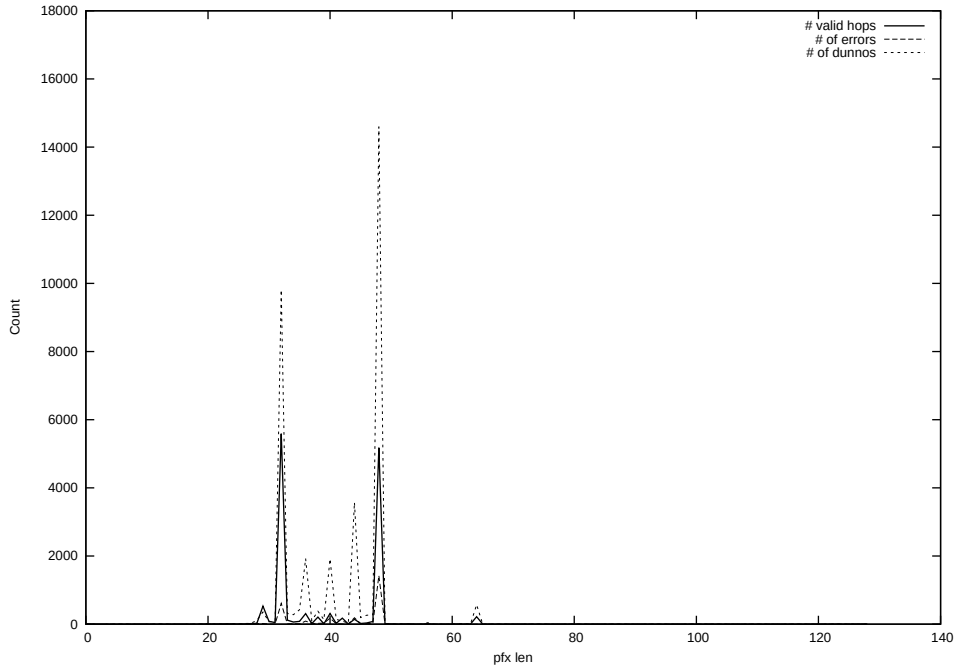
2014-01-17



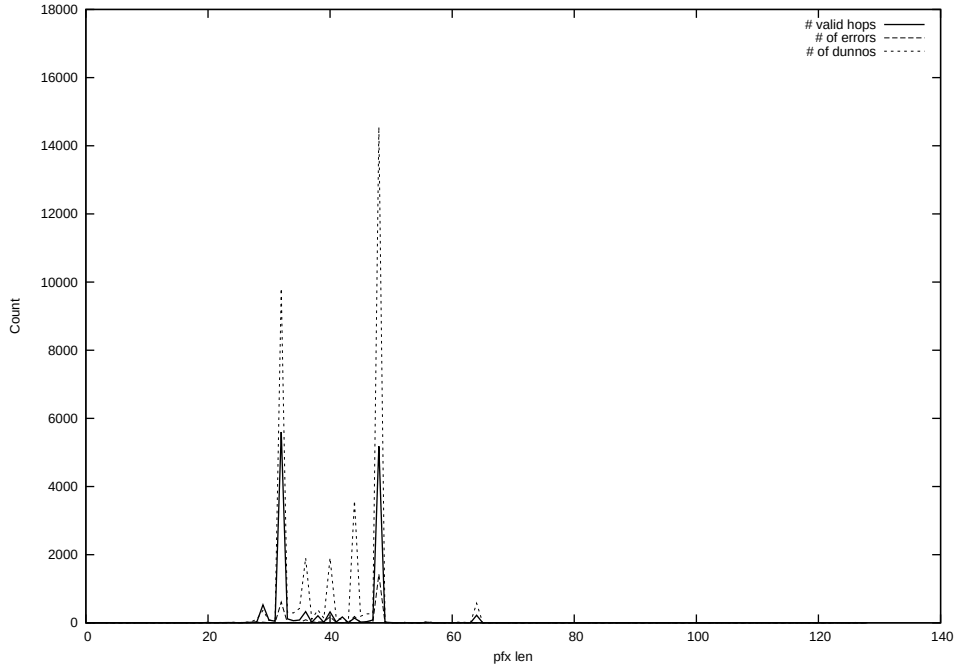
2014-01-18



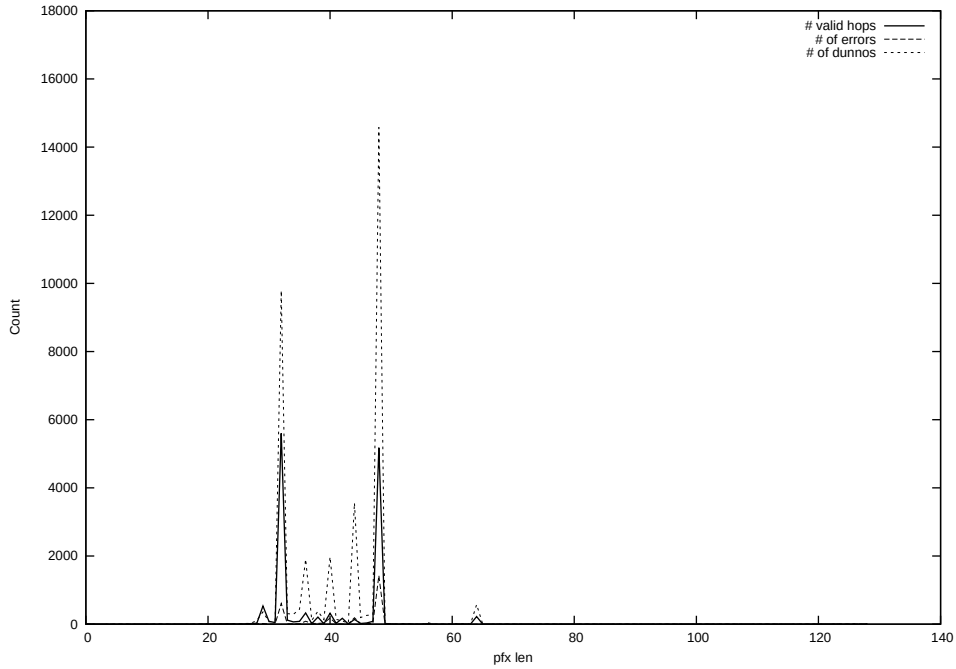
2014-01-19



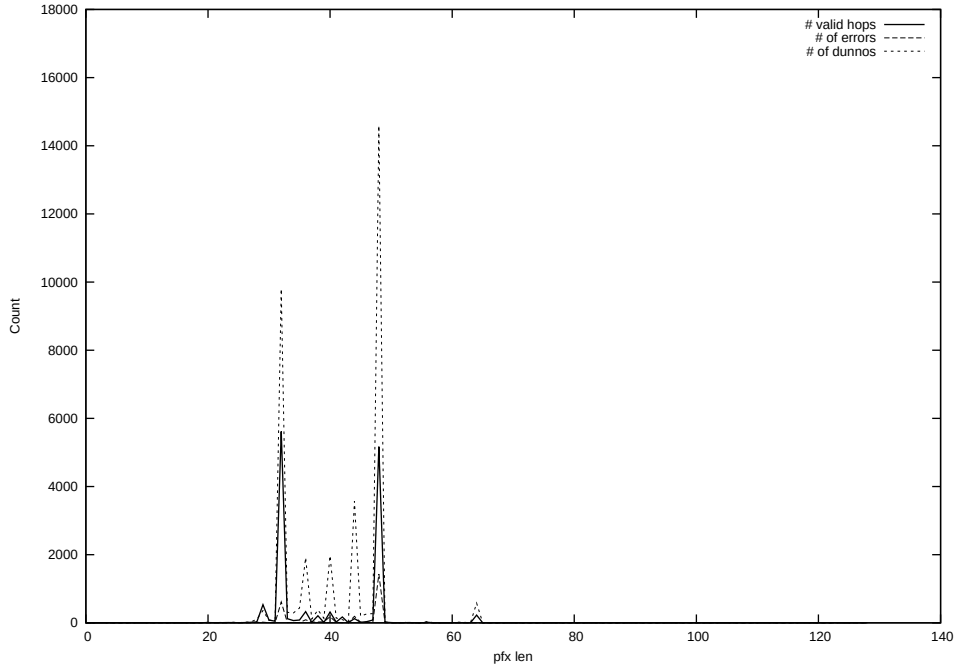
2014-01-20



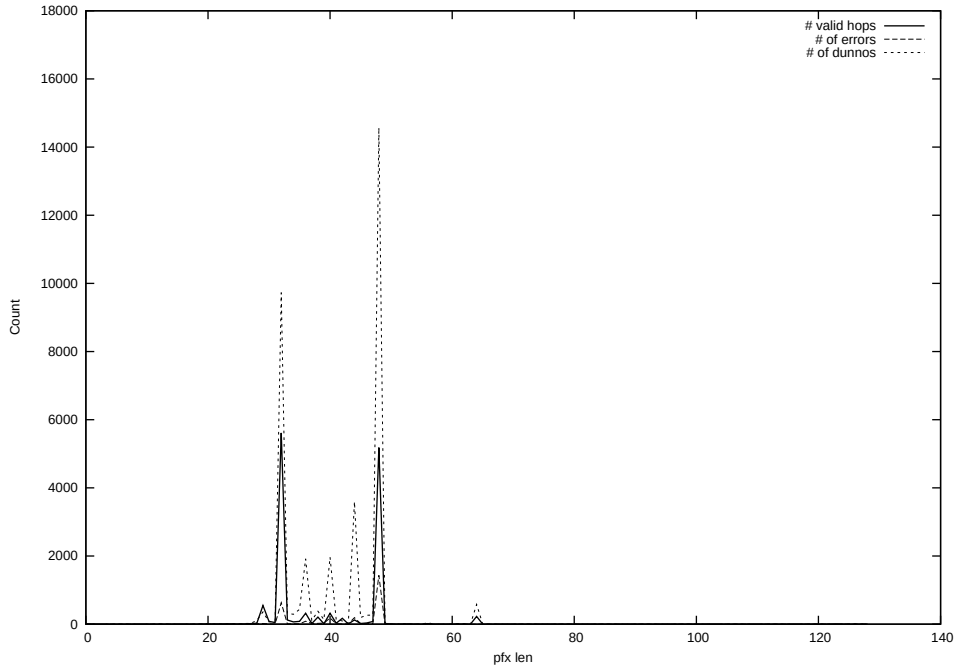
2014-01-21



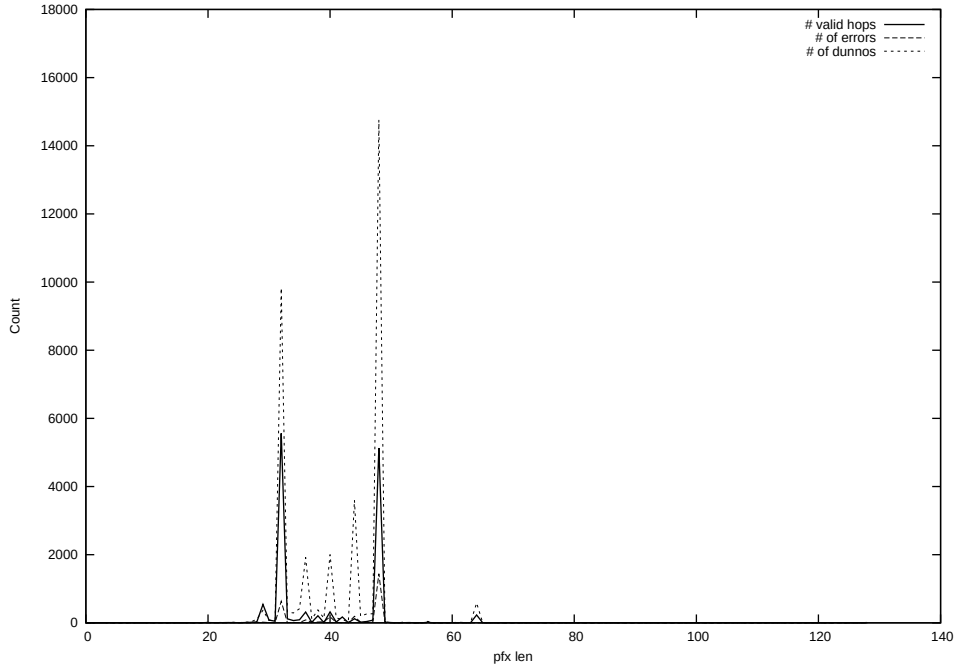
2014-01-22



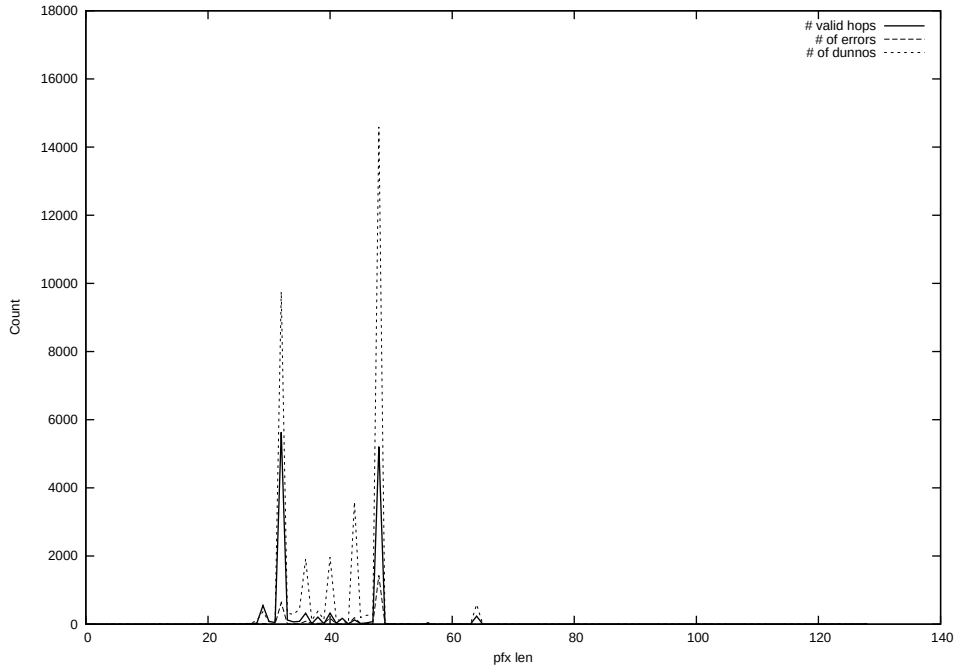
2014-01-23



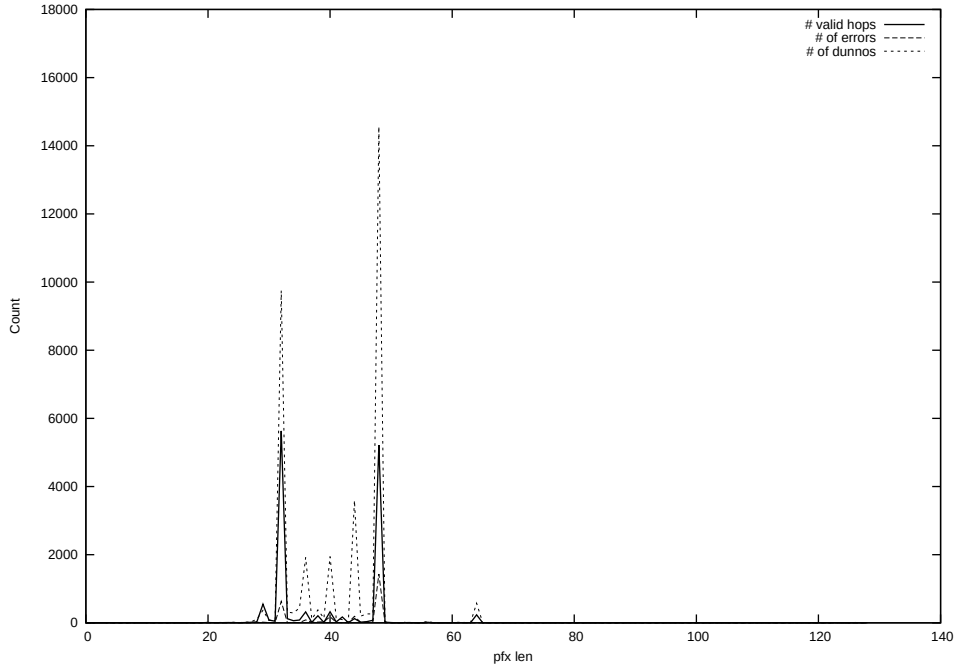
2014-01-24



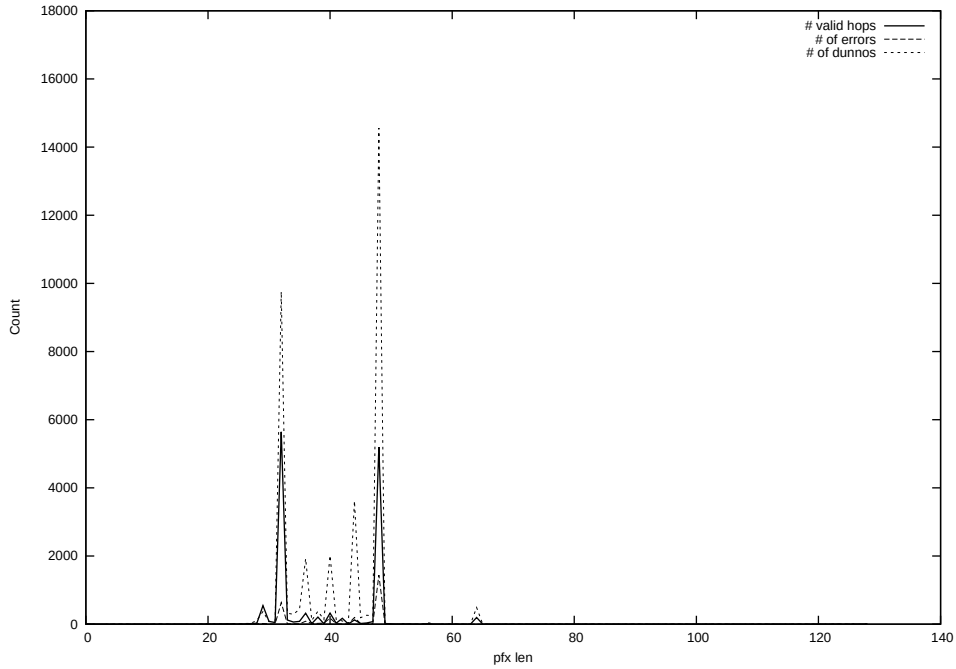
2014-01-25



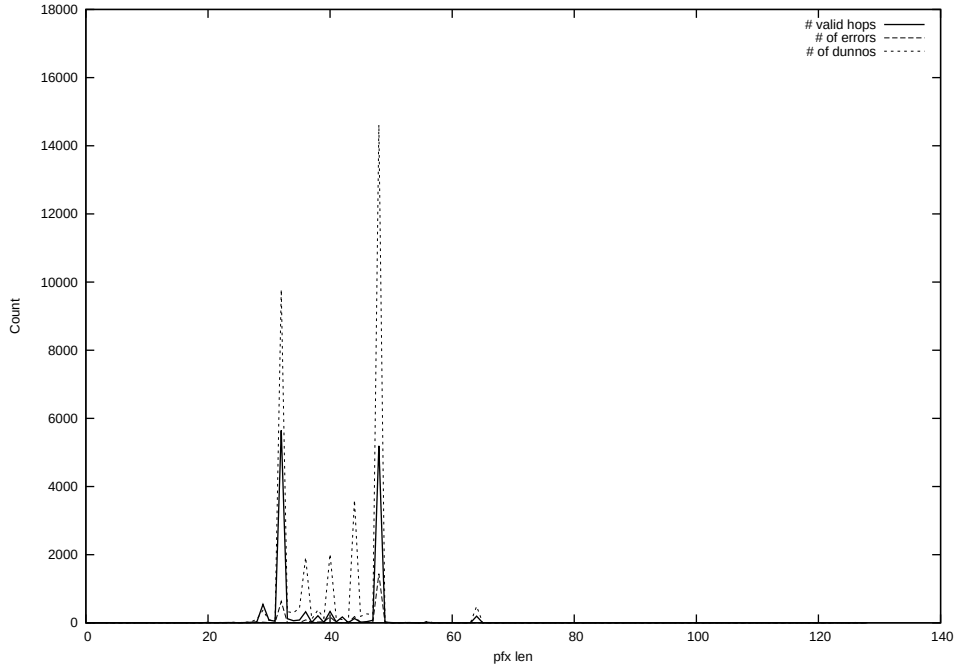
2014-01-26



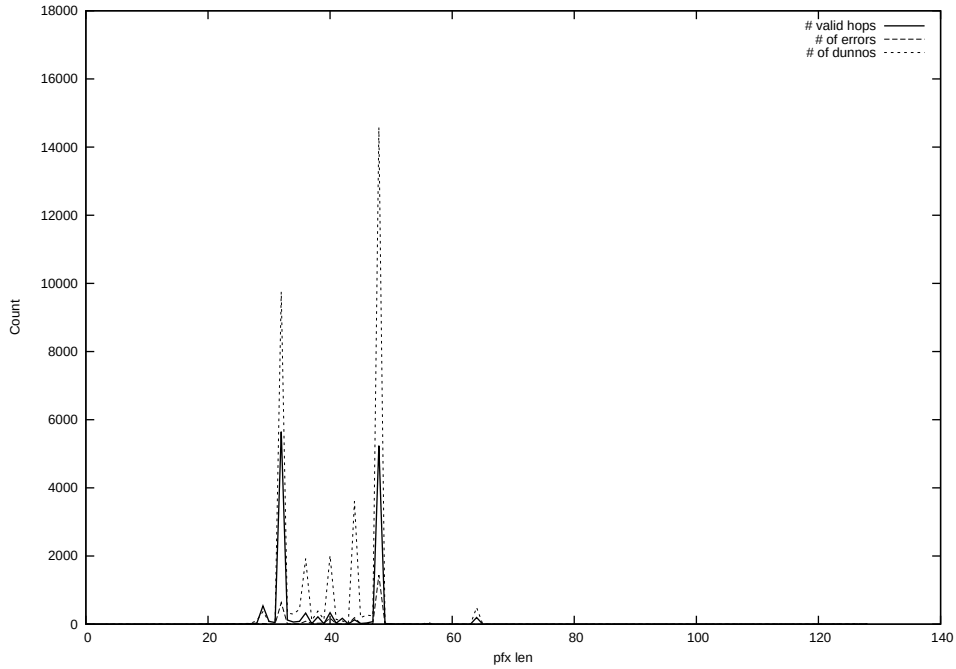
2014-01-27



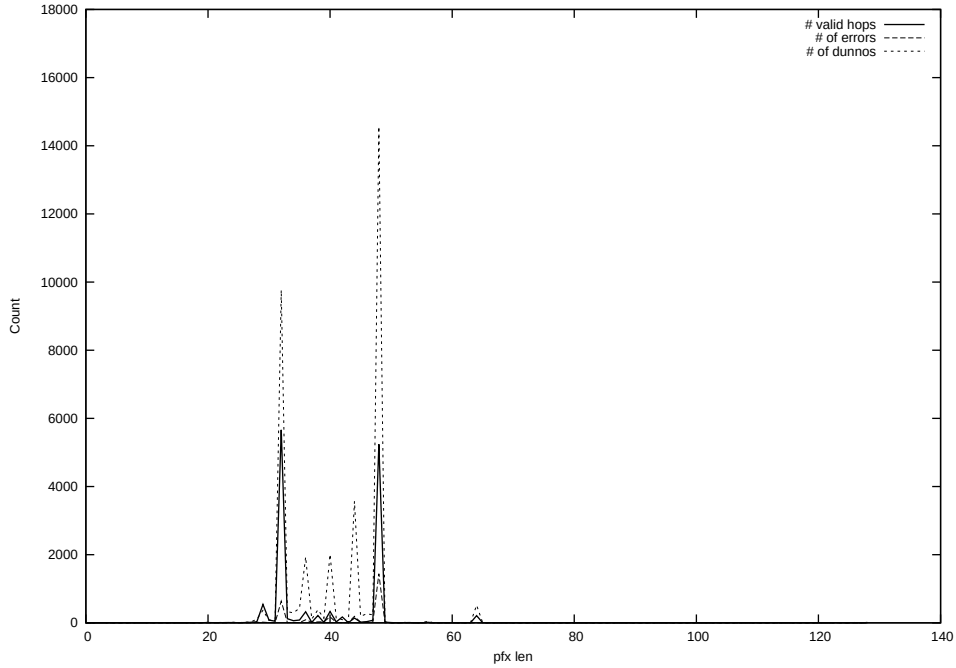
2014-01-28



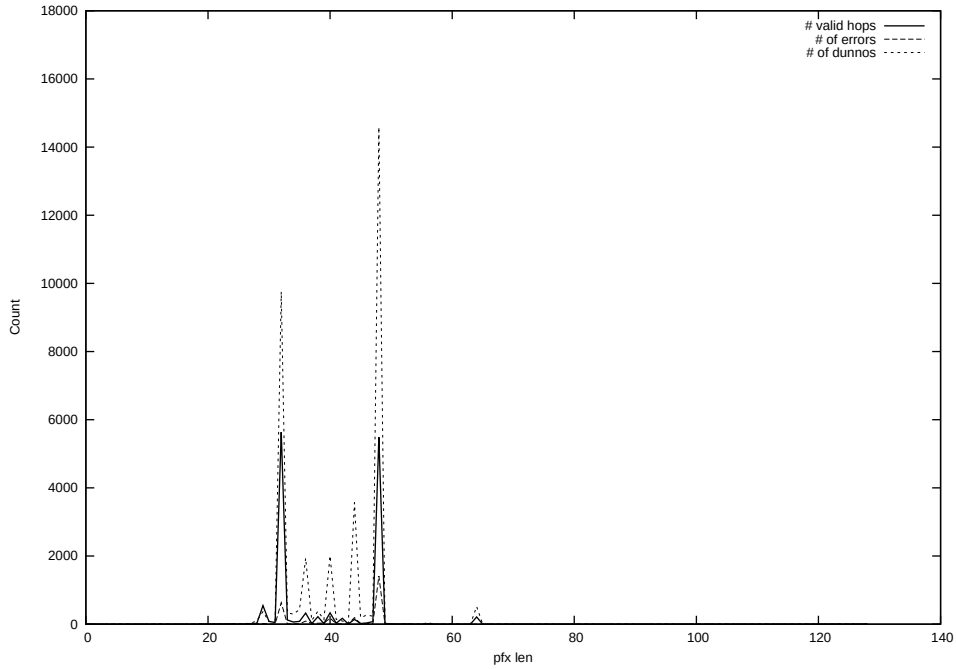
2014-01-29



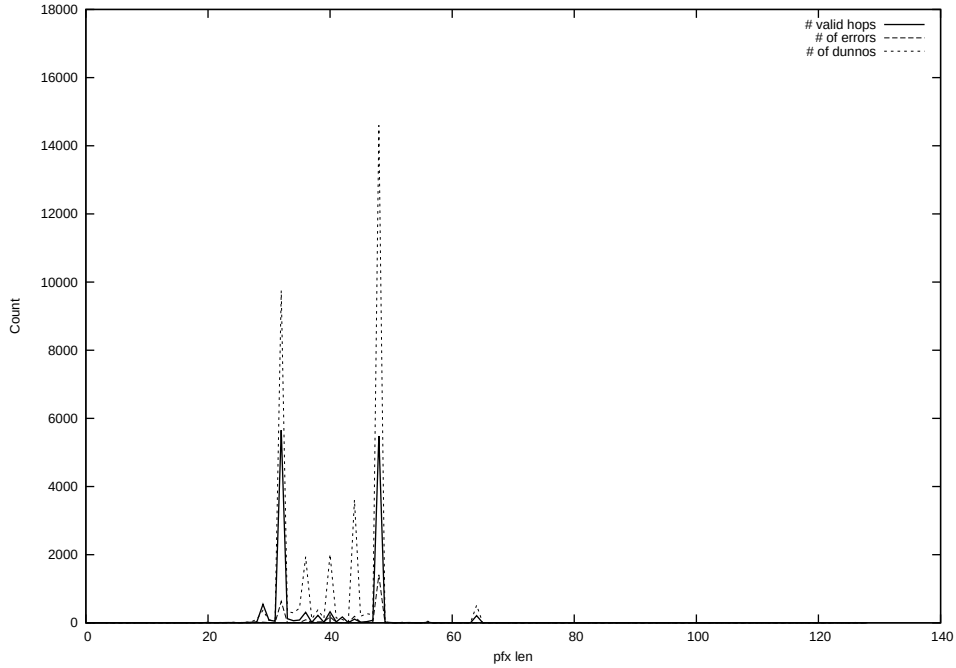
2014-01-30



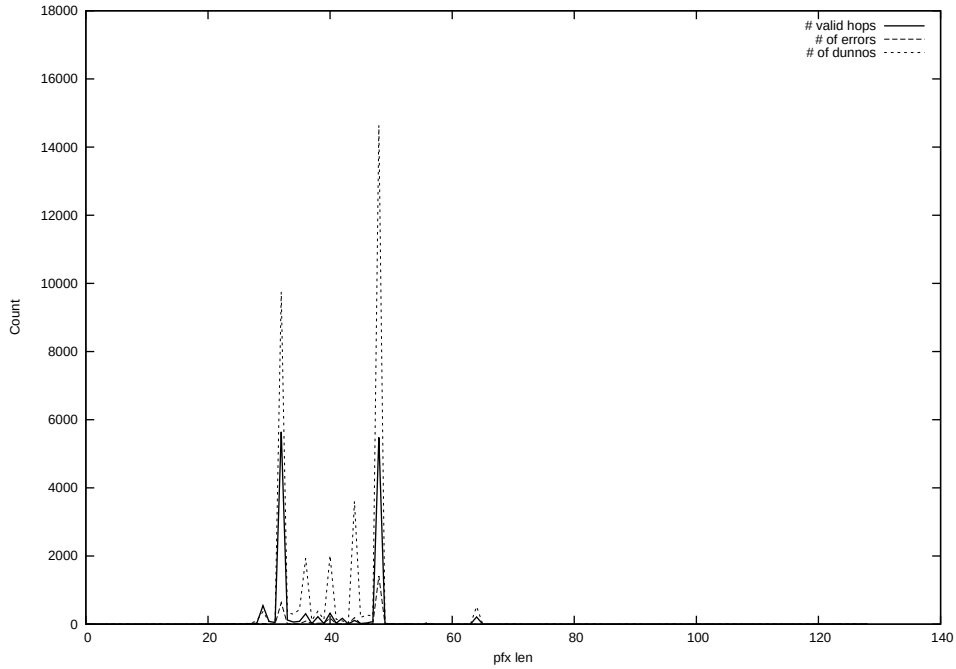
2014-01-31



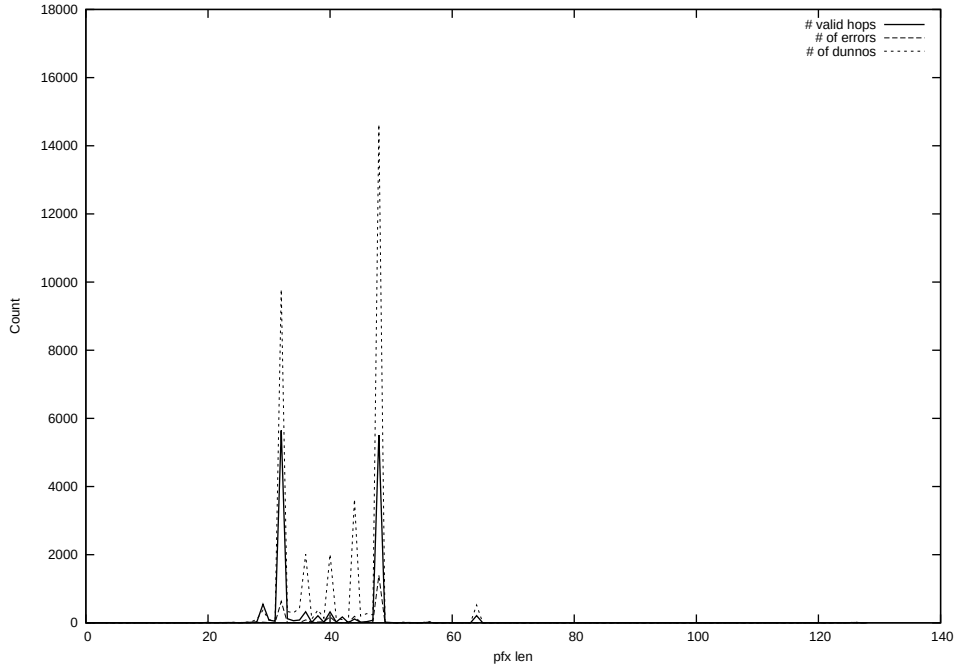
2014-02-01



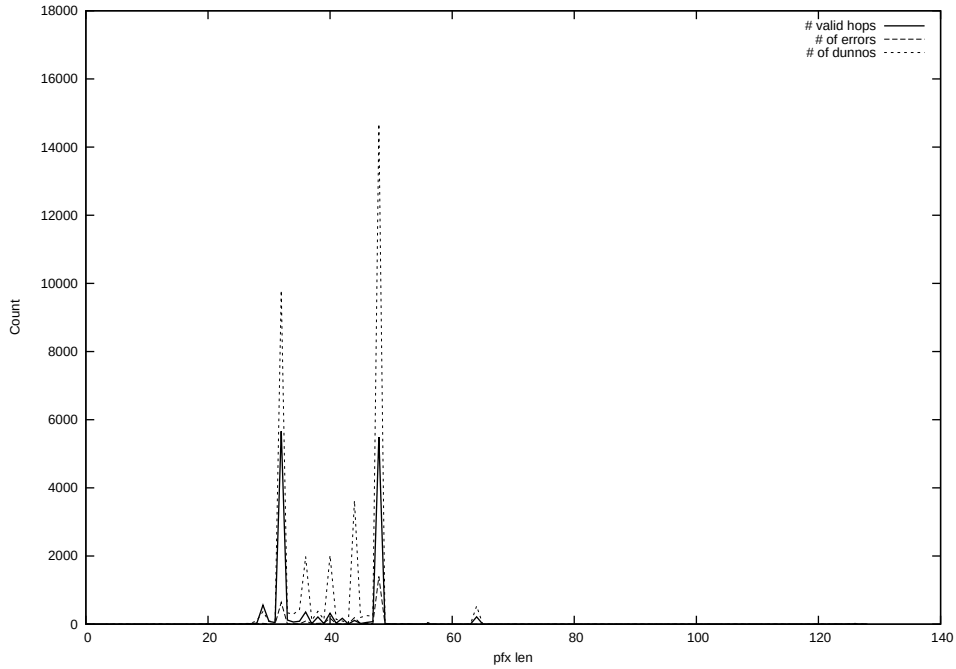
2014-02-02



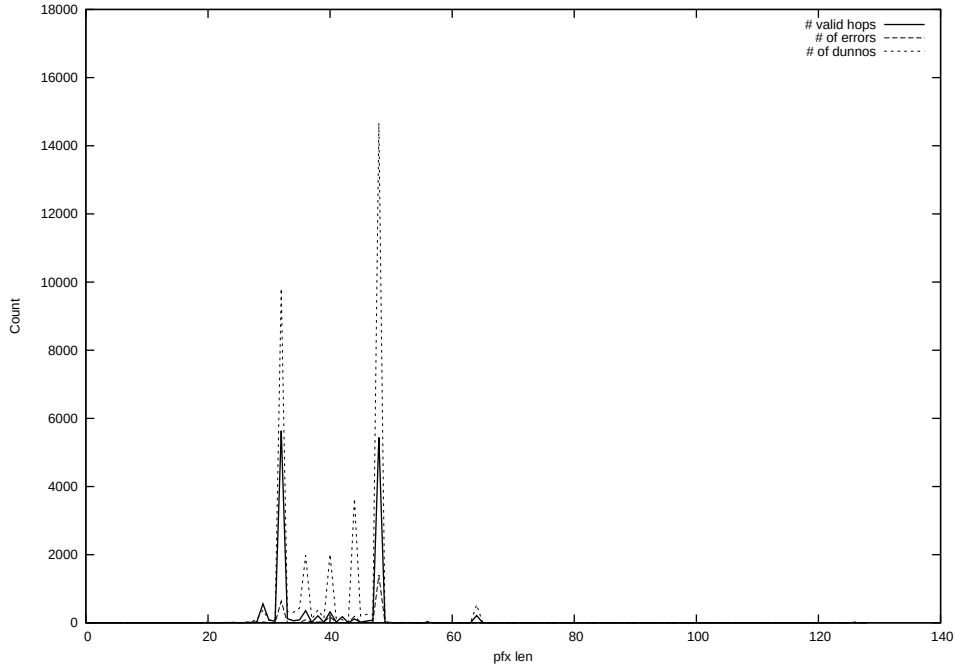
2014-02-03



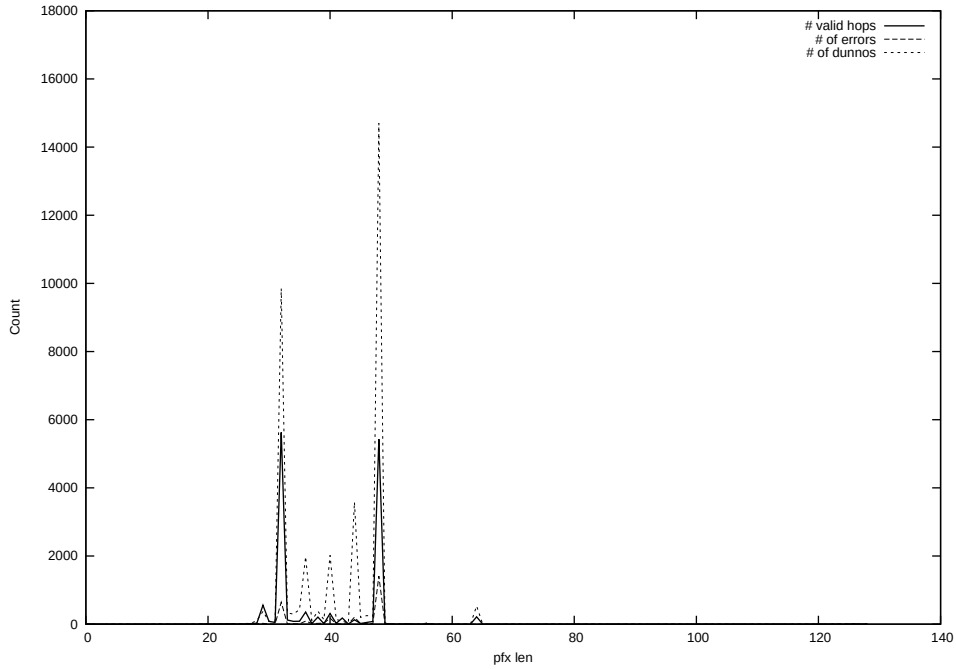
2014-02-04



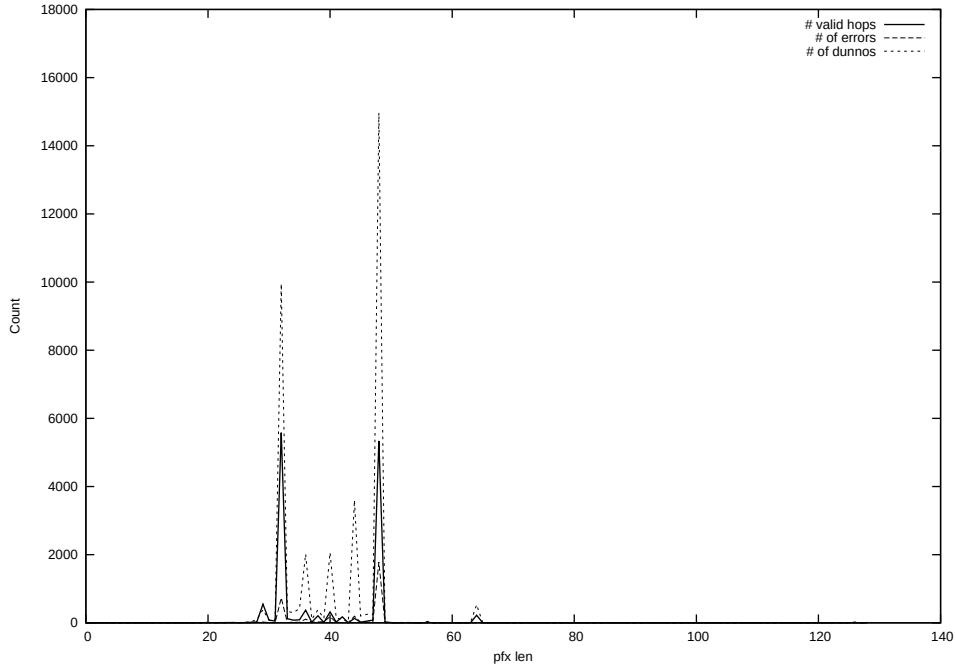
2014-02-05



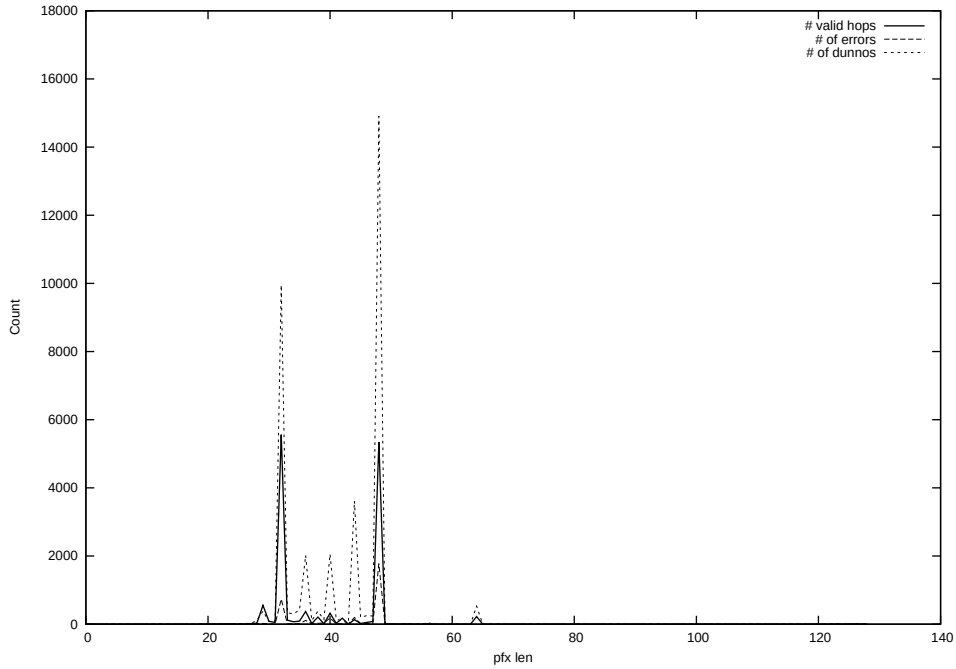
2014-02-06



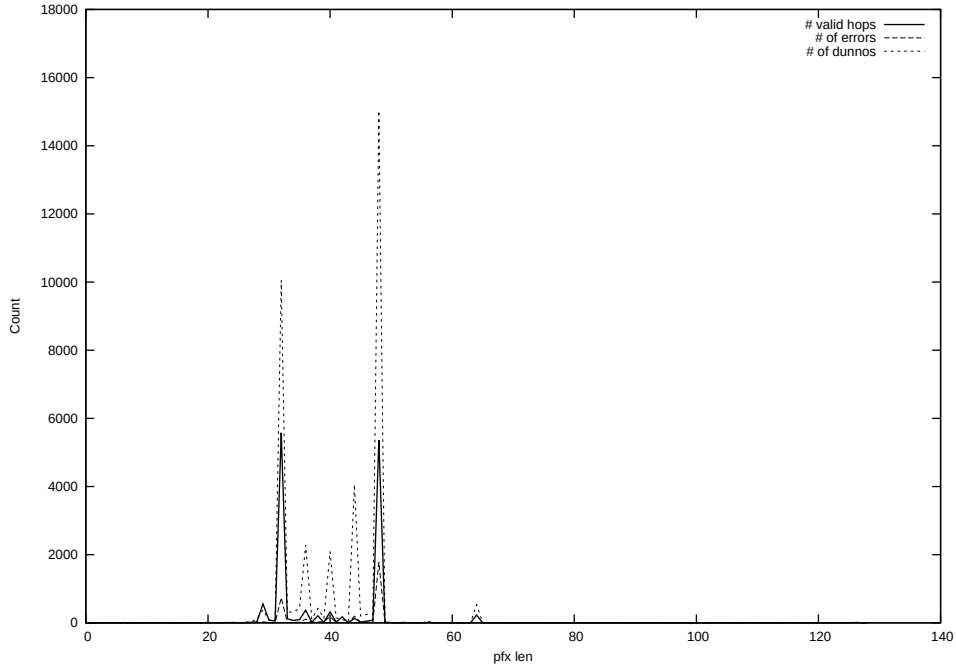
2014-02-07



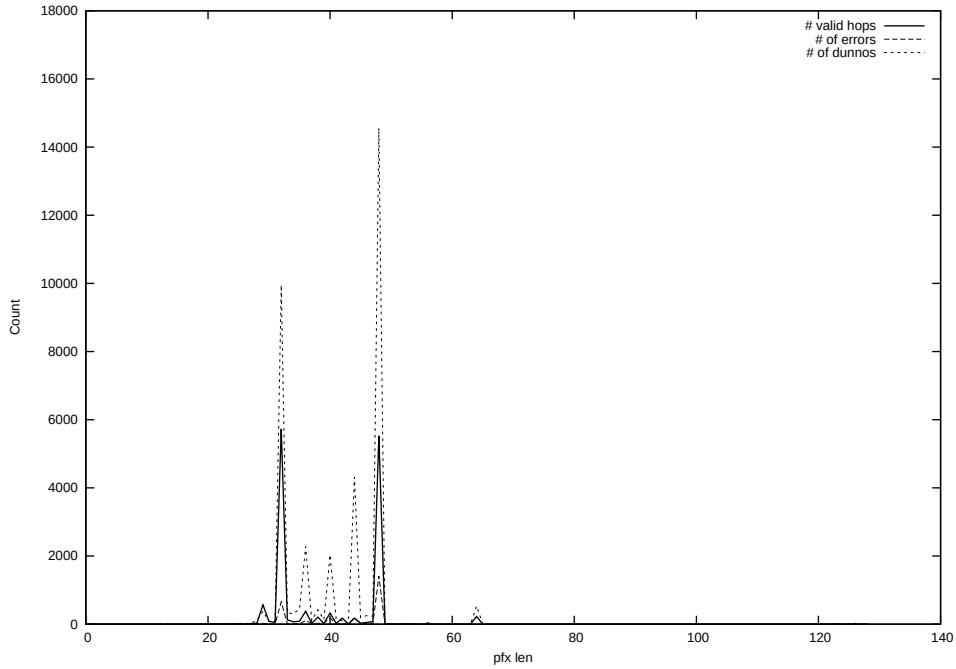
2014-02-08



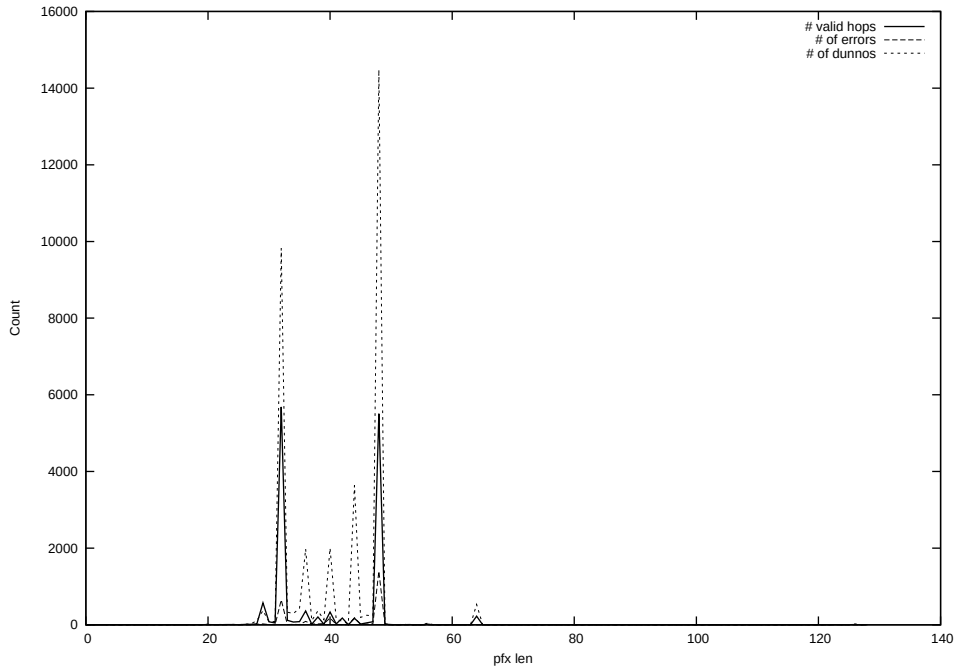
2014-02-09



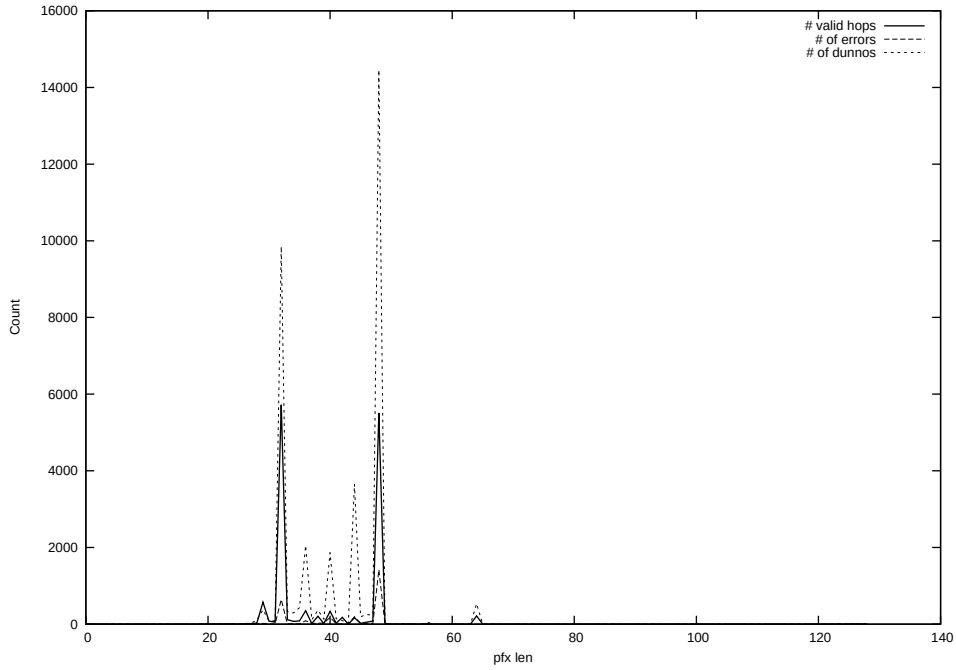
2014-02-10



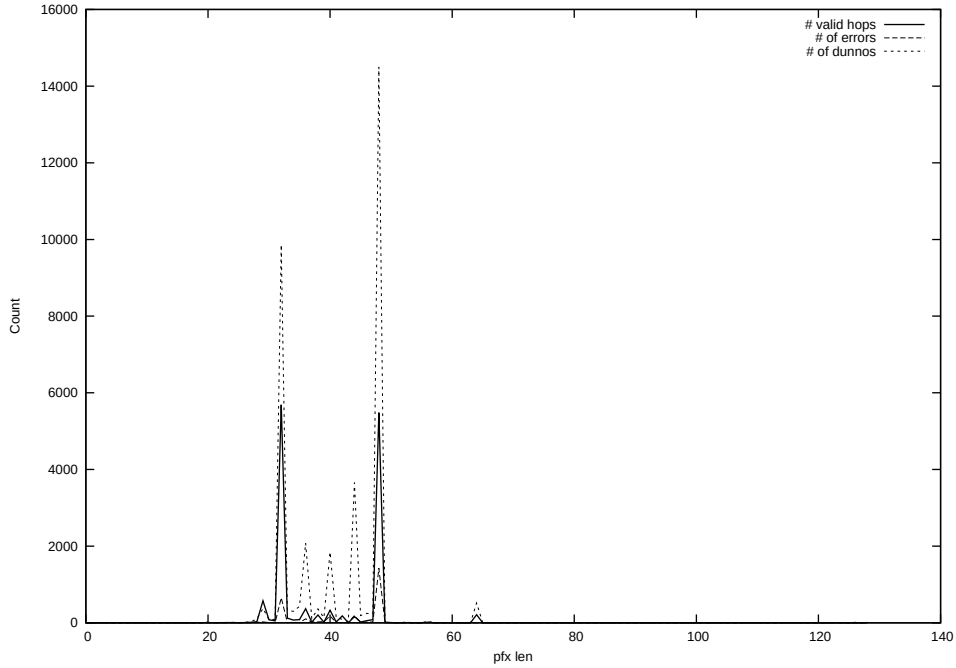
2014-02-11



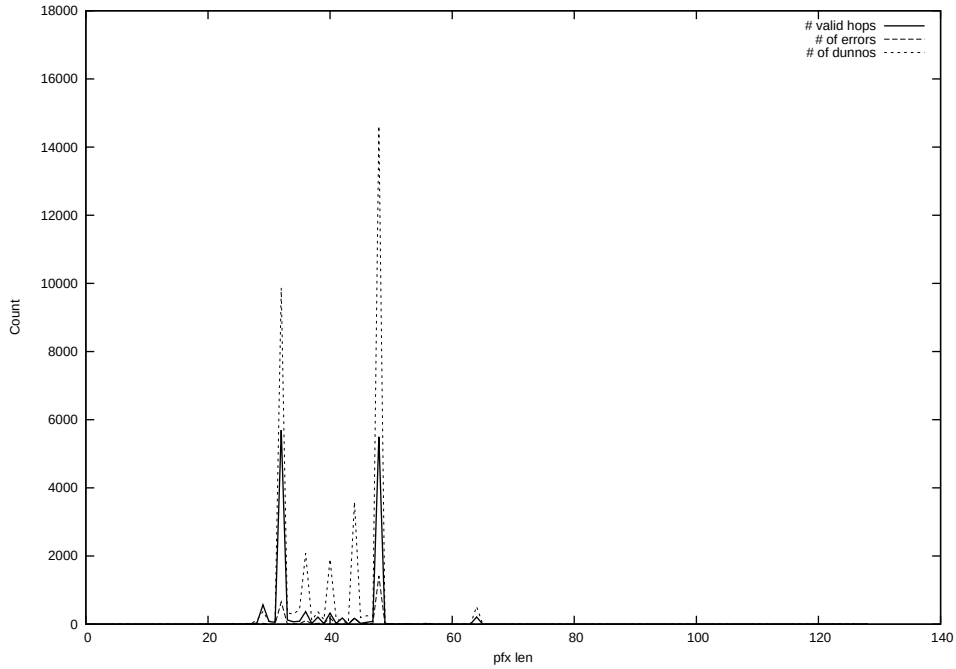
2014-02-12



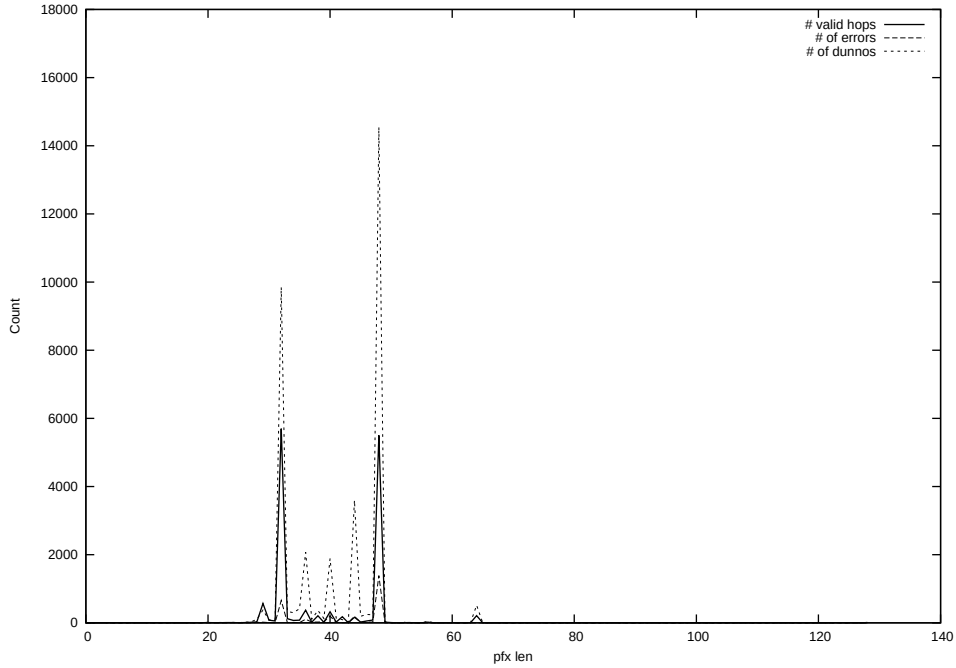
2014-02-13



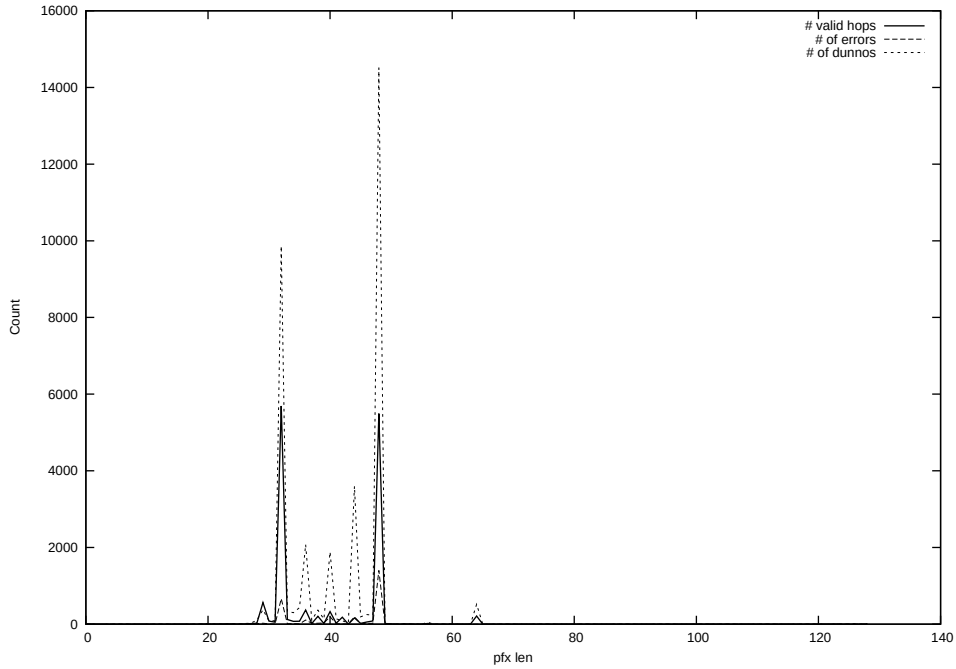
2014-02-14



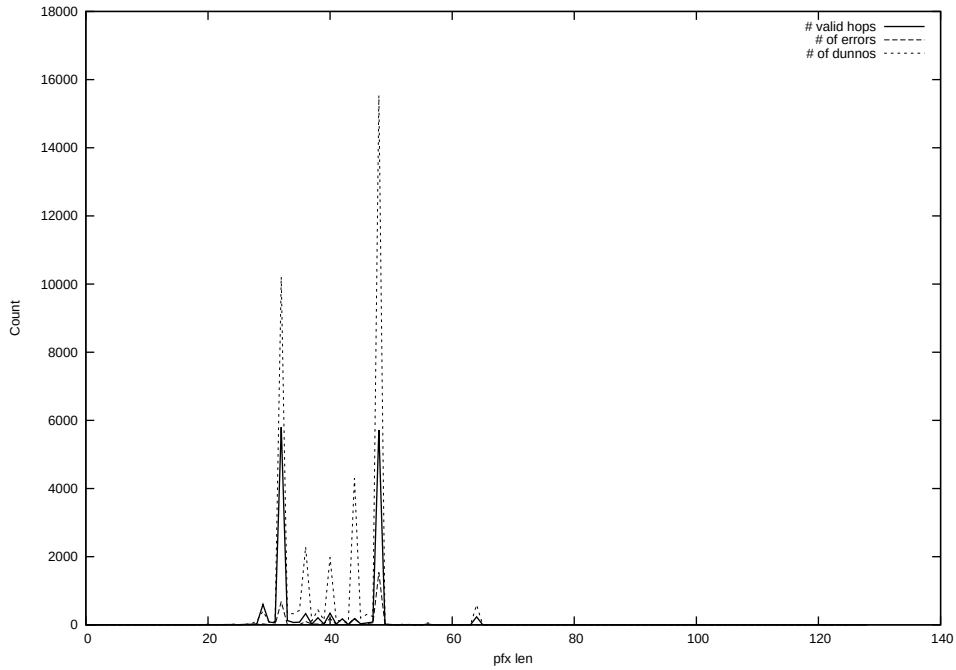
2014-02-15



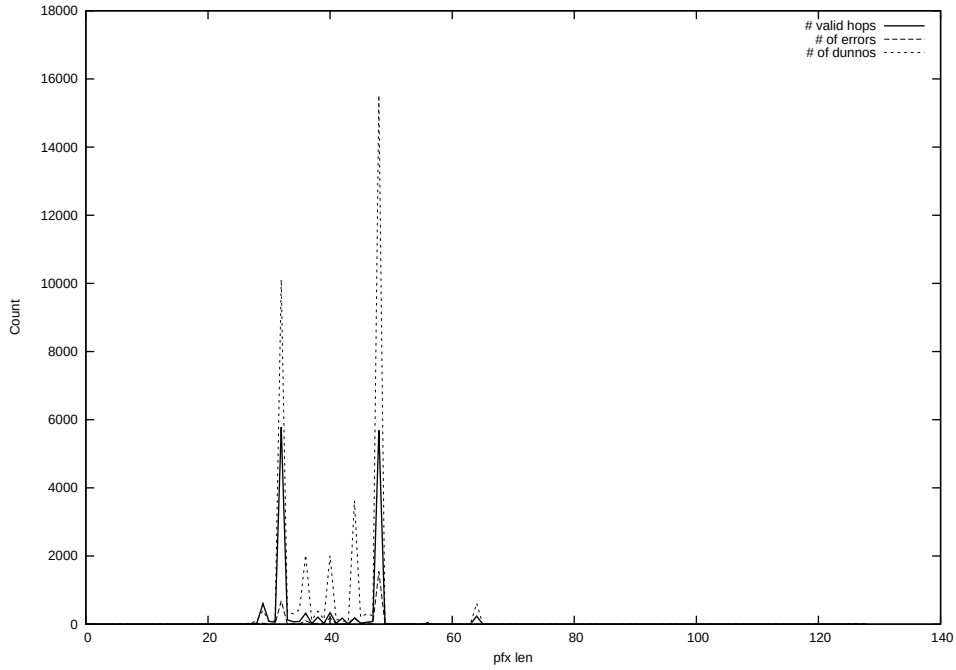
2014-02-16



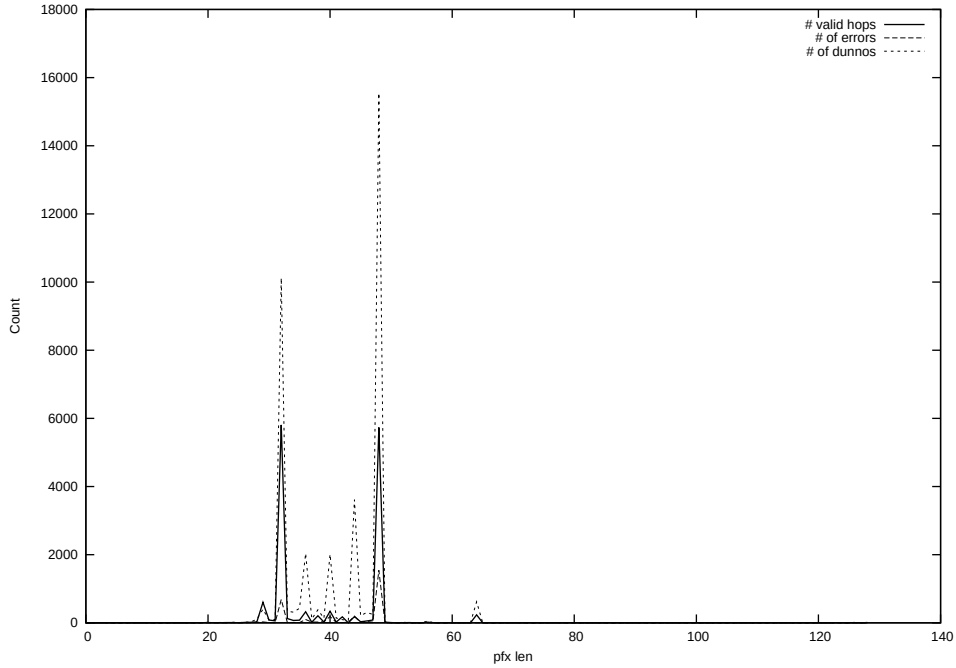
2014-02-17



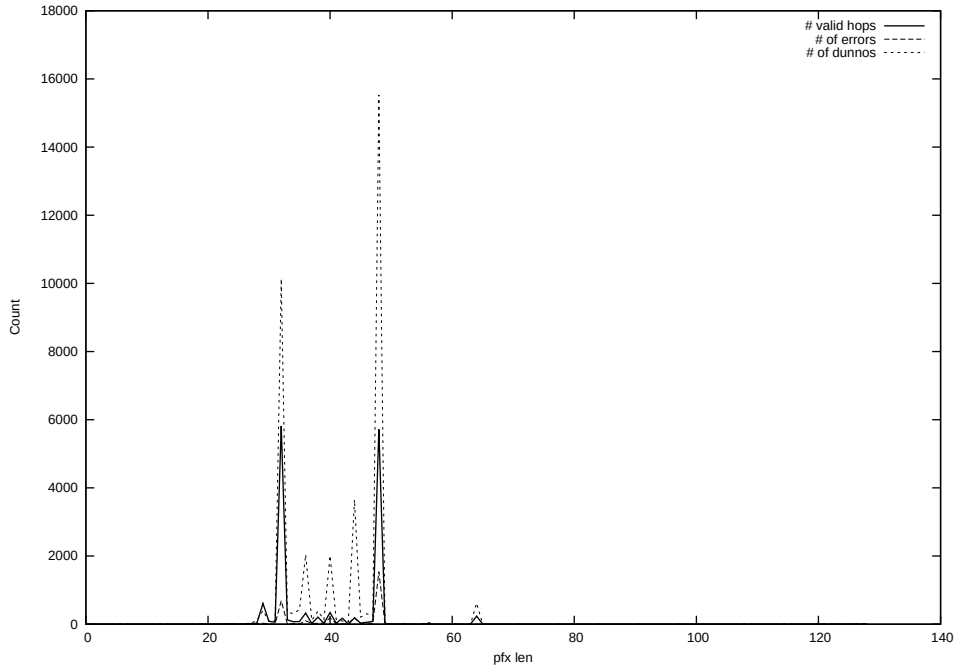
2014-03-20



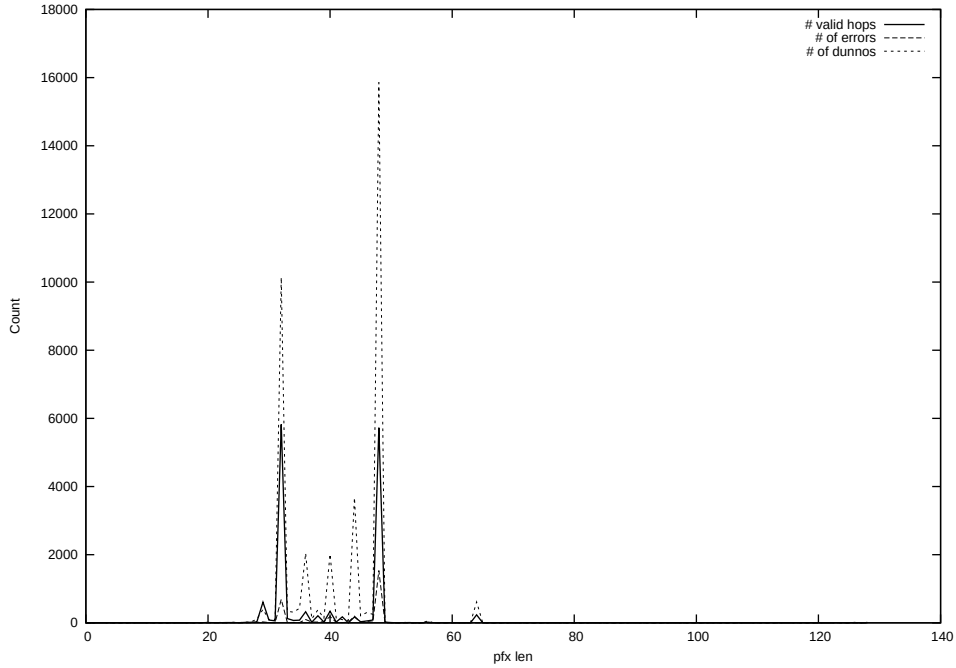
2014-03-21



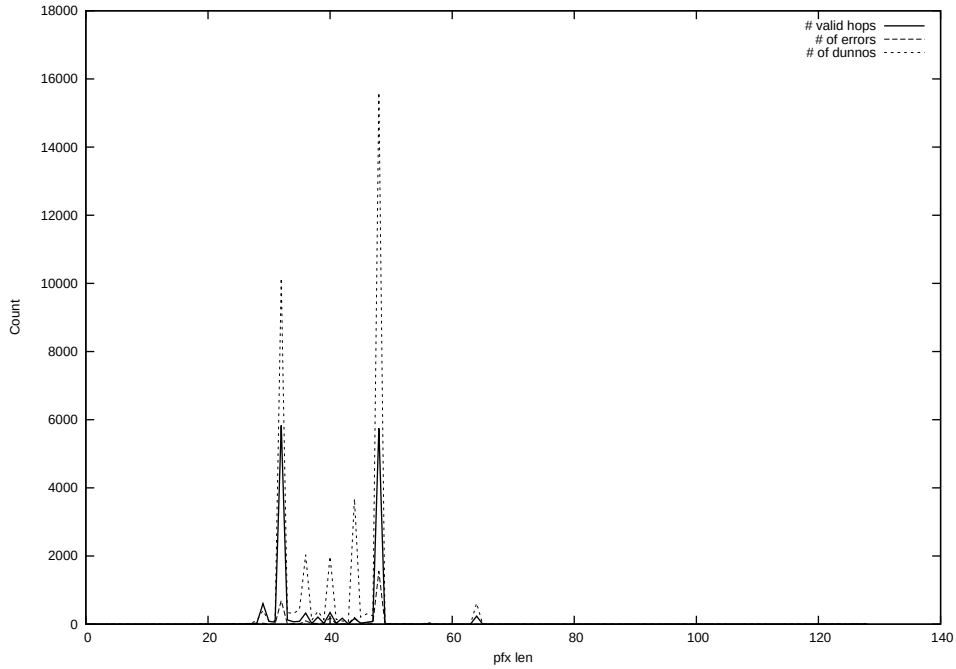
2014-03-22



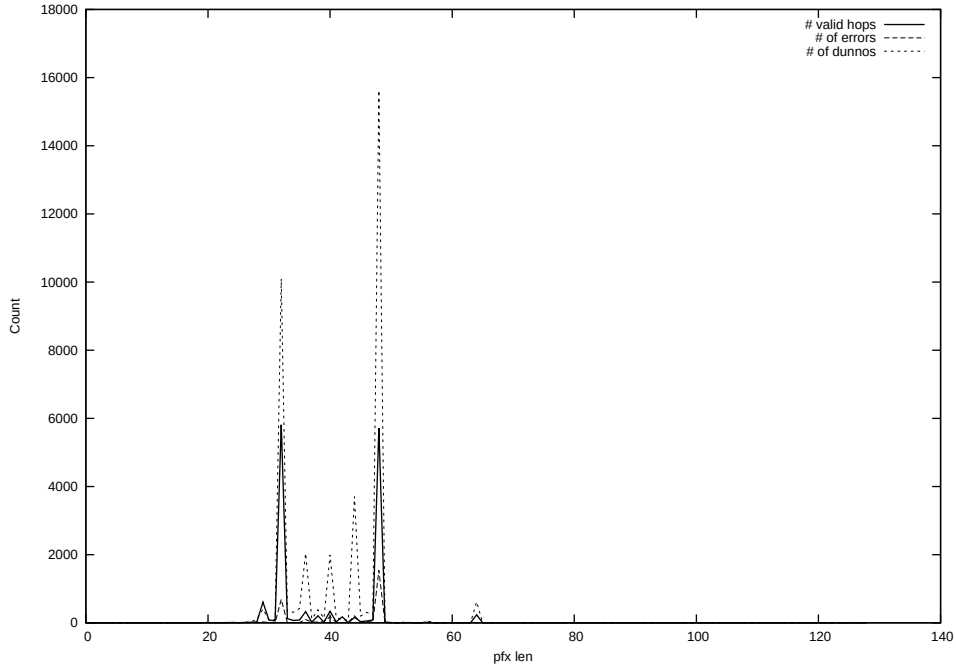
2014-03-23



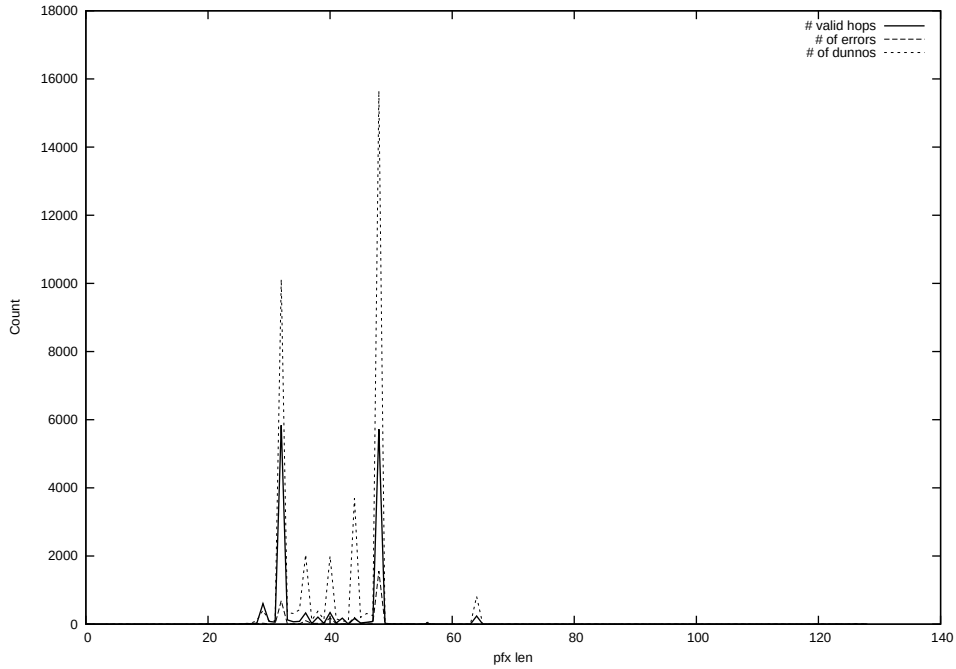
2014-03-24



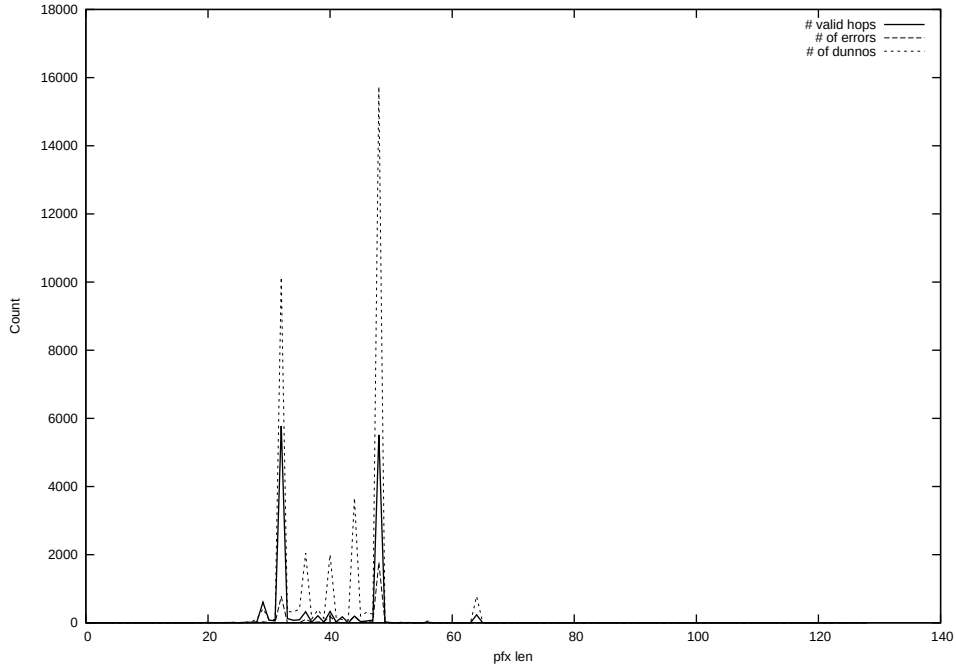
2014-03-25



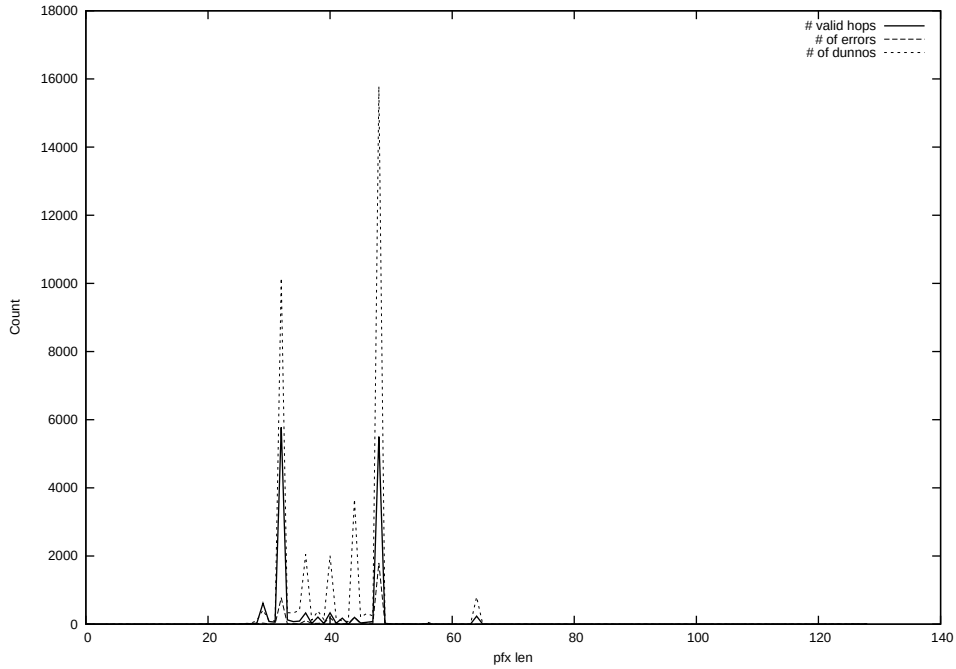
2014-03-26



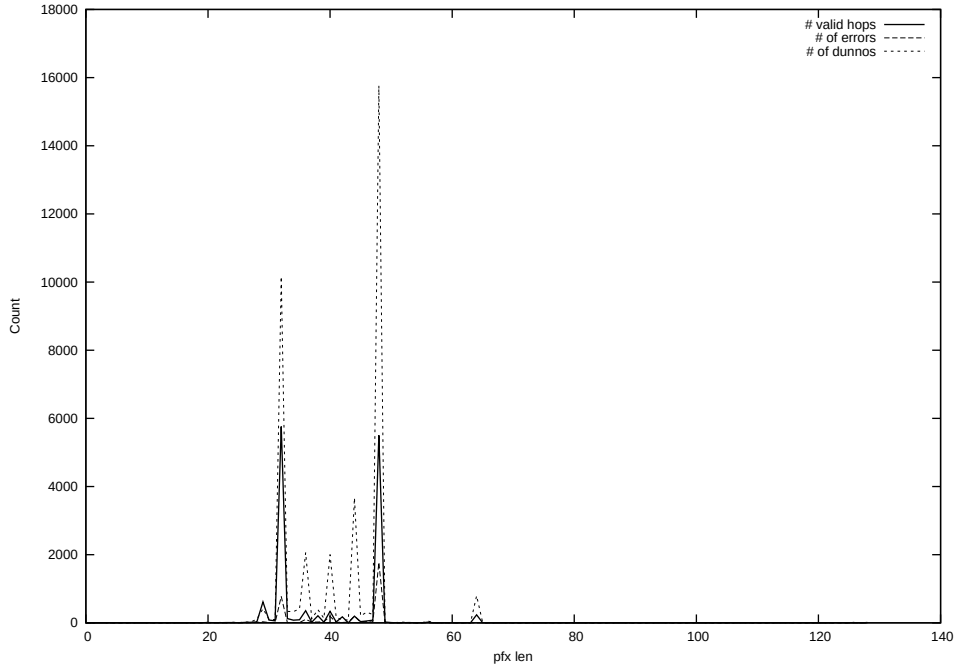
2014-03-27



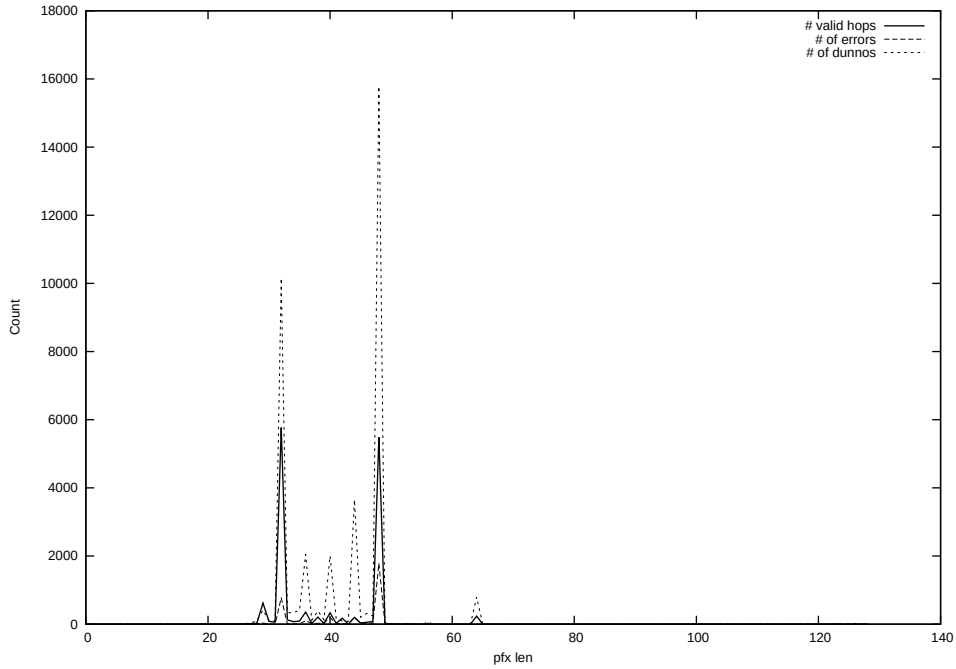
2014-03-28



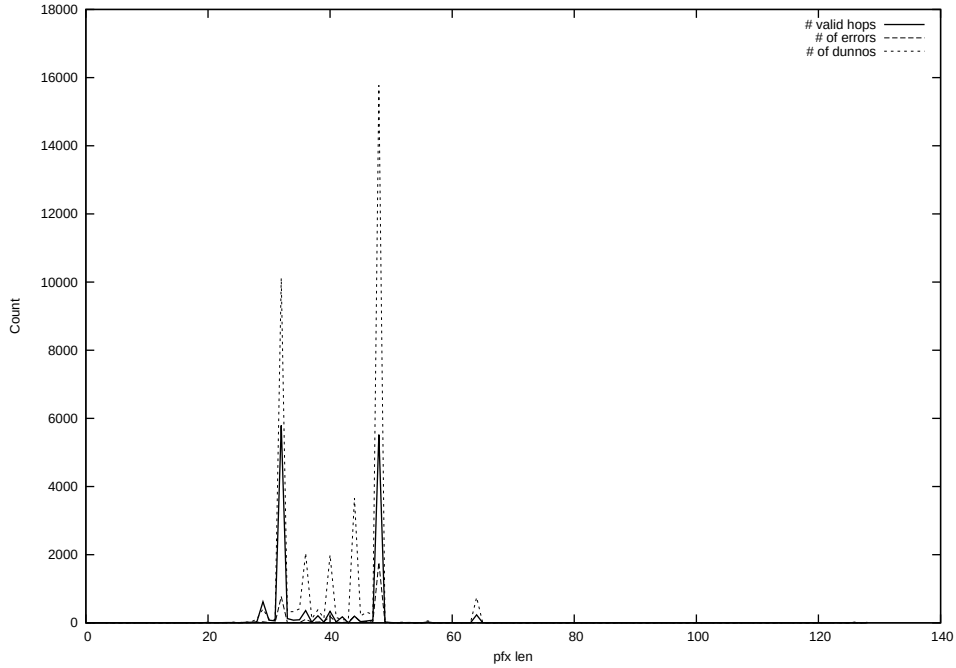
2014-03-29



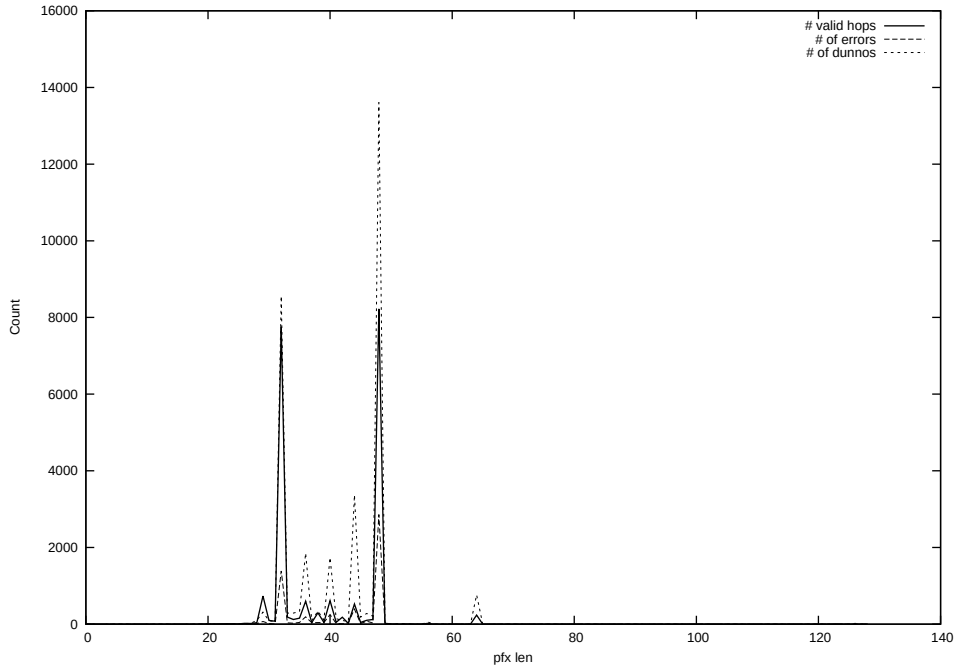
2014-03-30



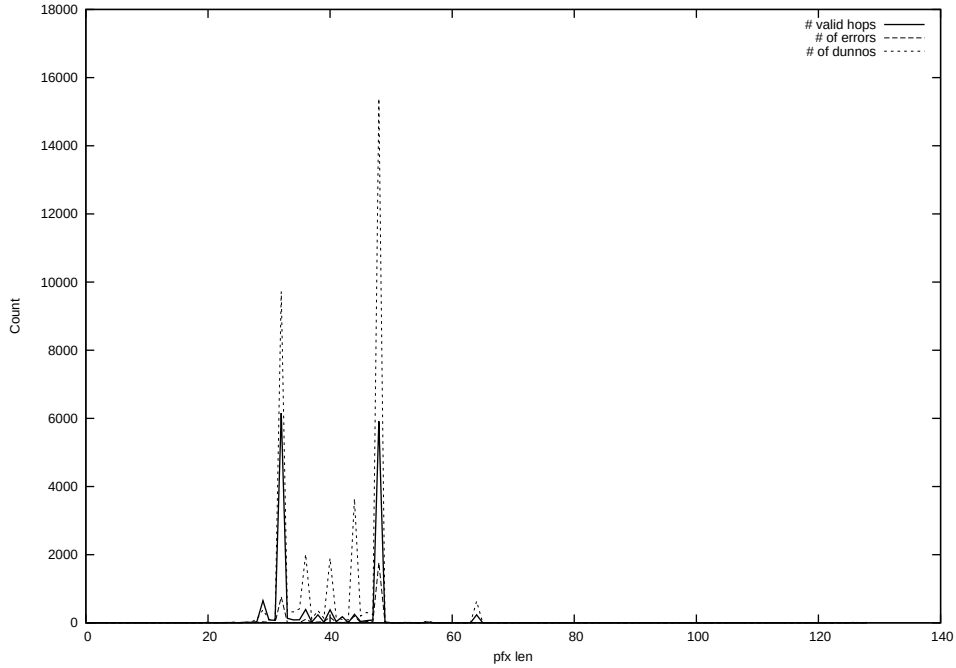
2014-03-31



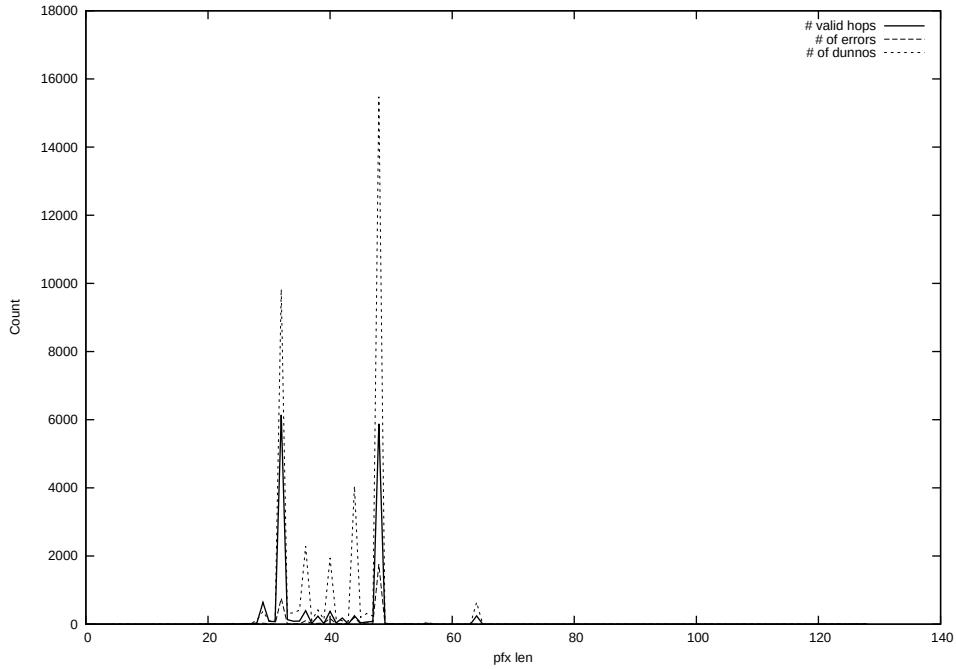
2014-04-01



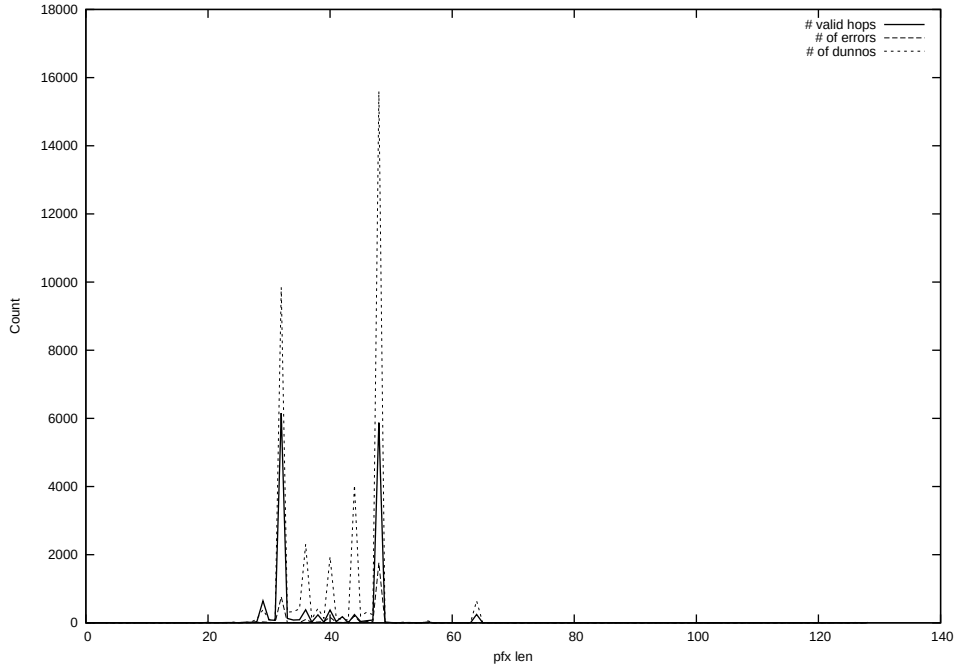
2014-04-02



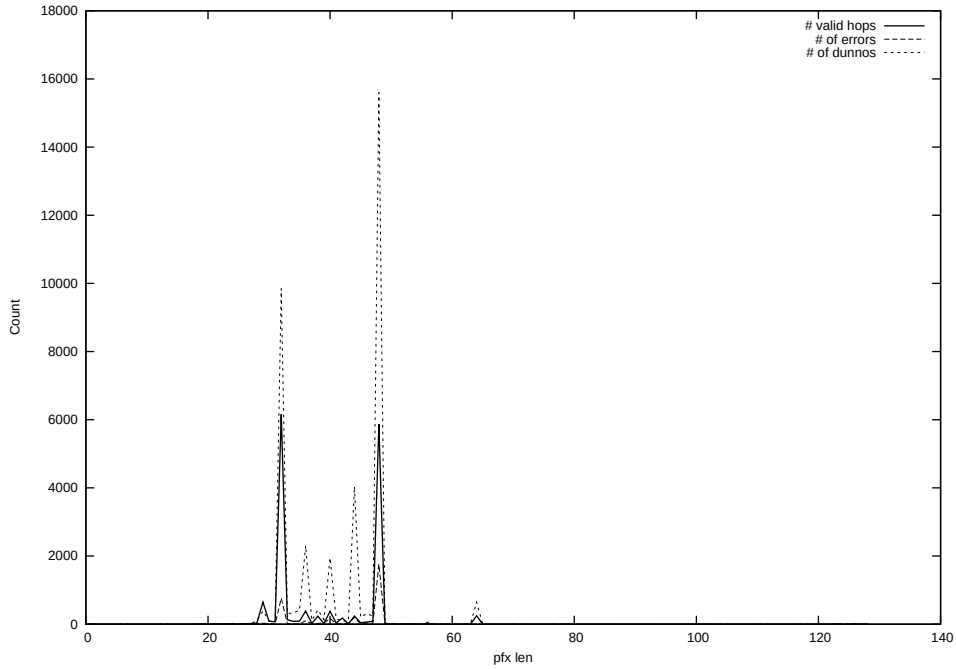
2014-04-03



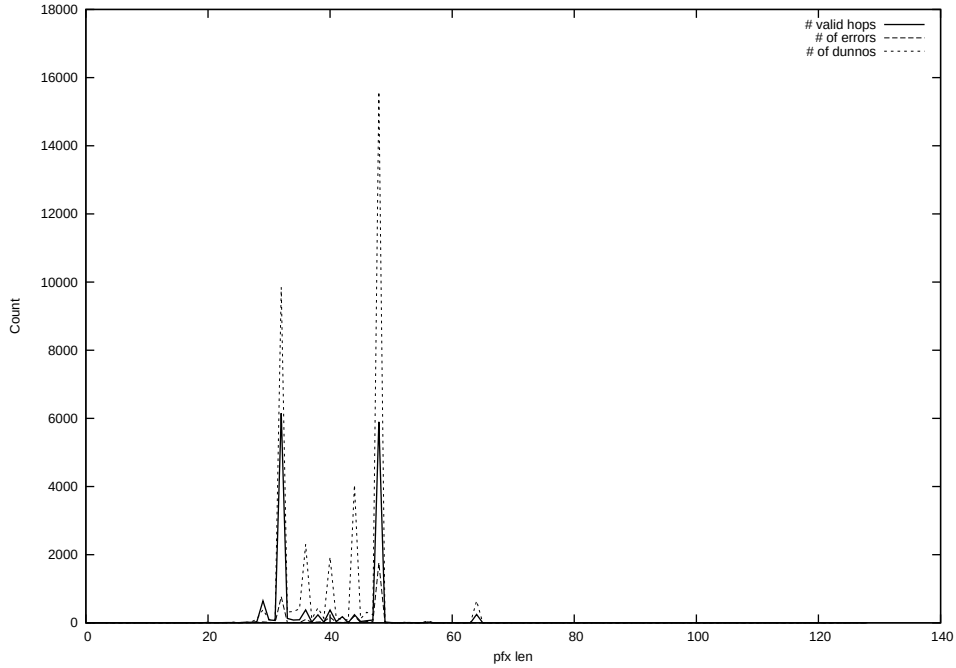
2014-04-04



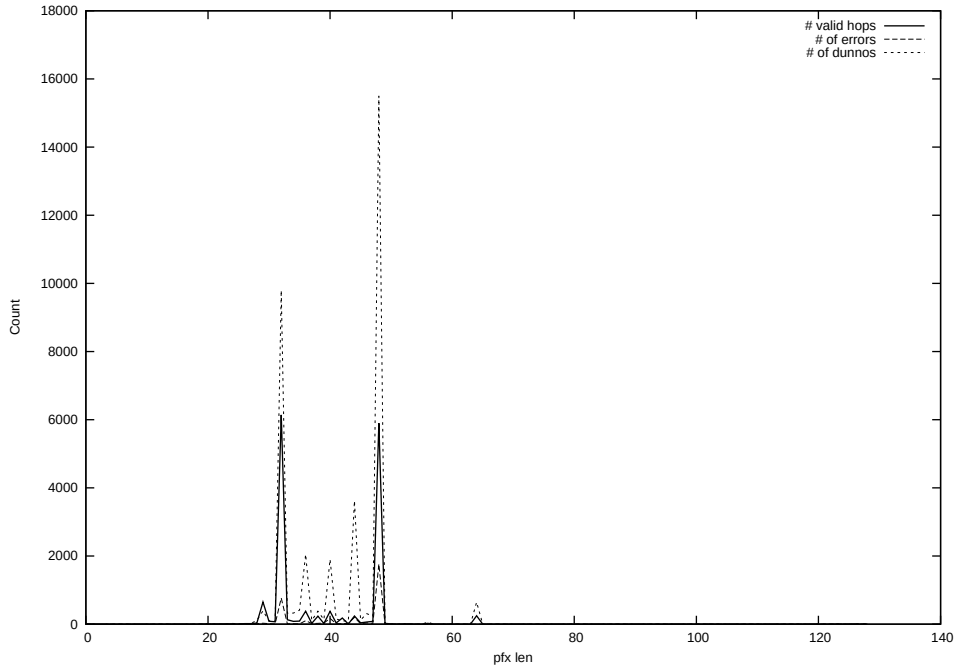
2014-04-05



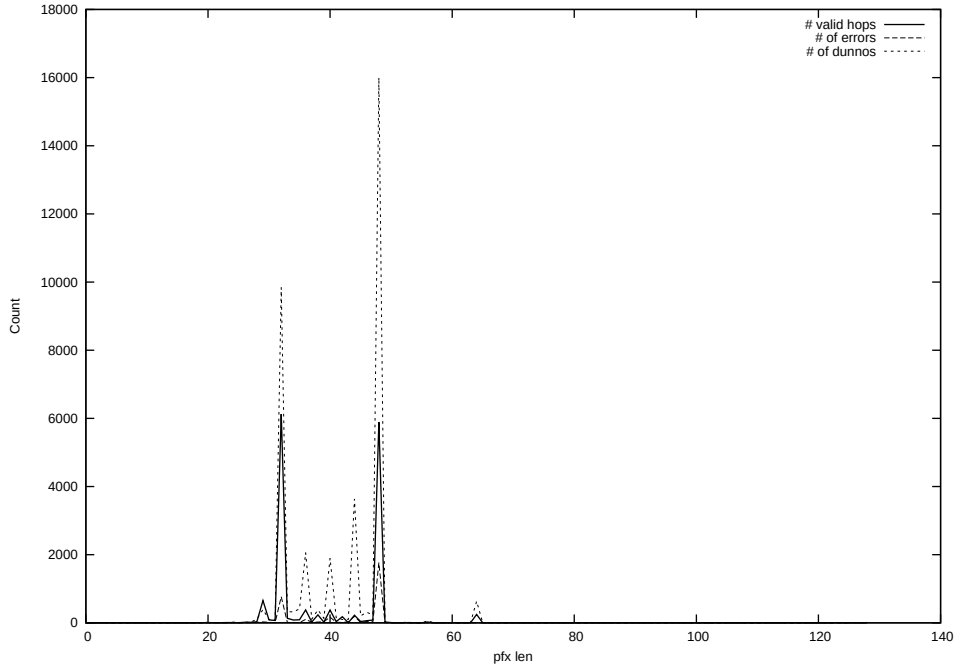
2014-04-06



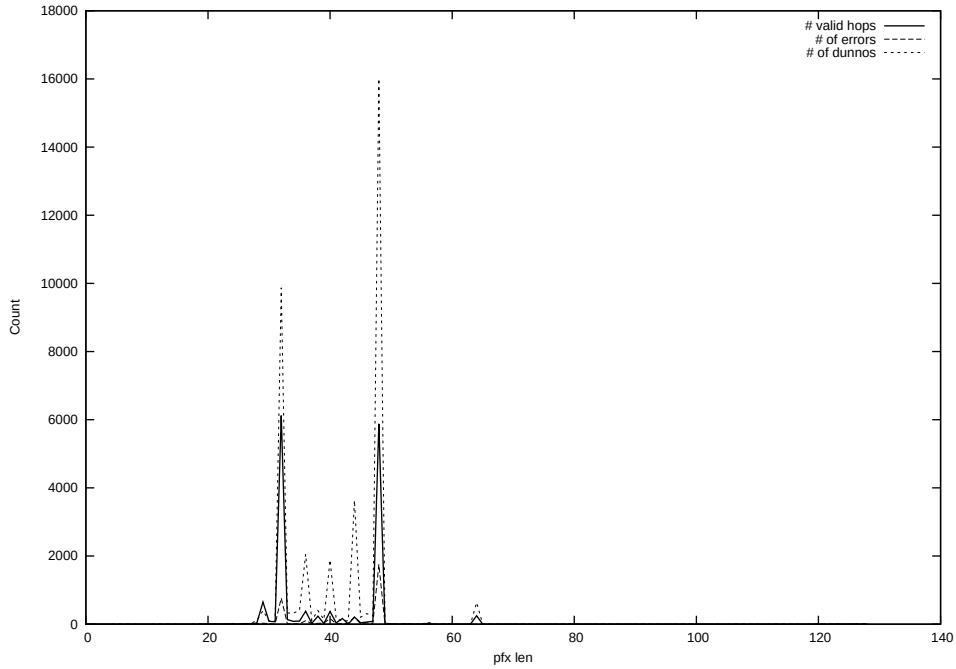
2014-04-07



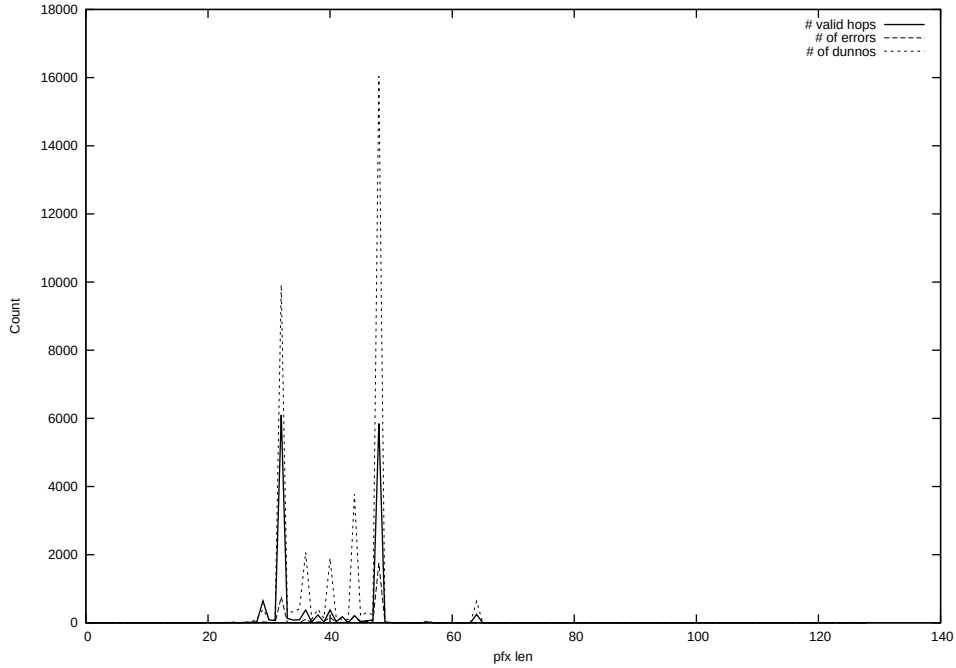
2014-04-08



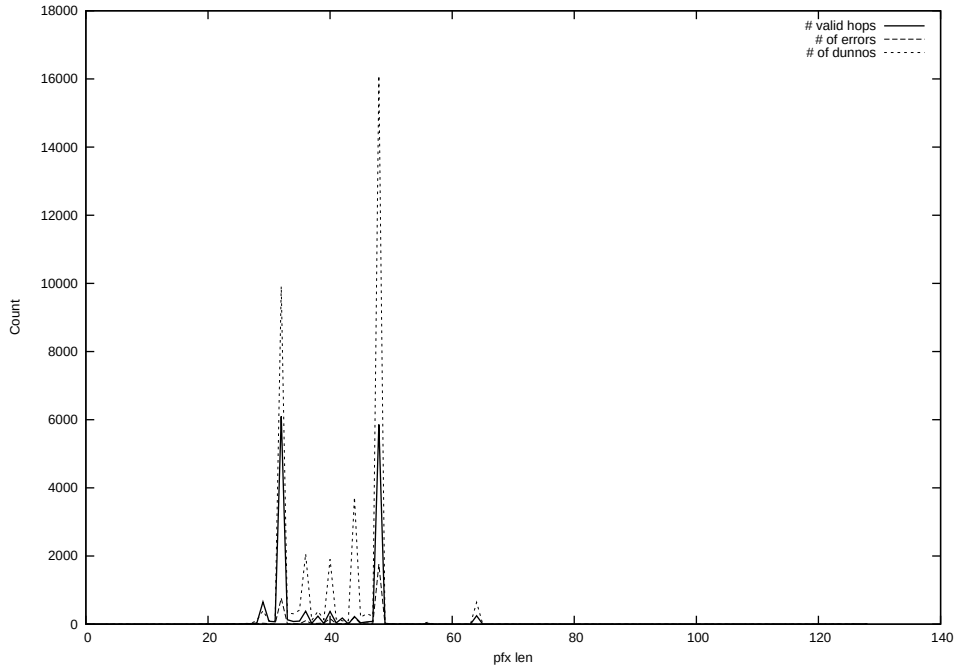
2014-04-09



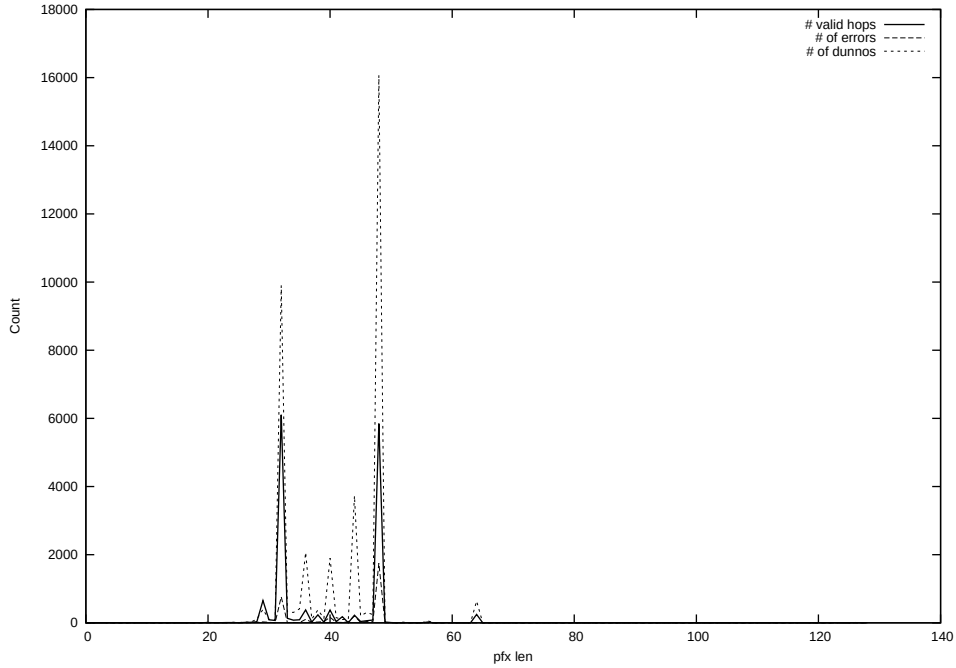
2014-04-10



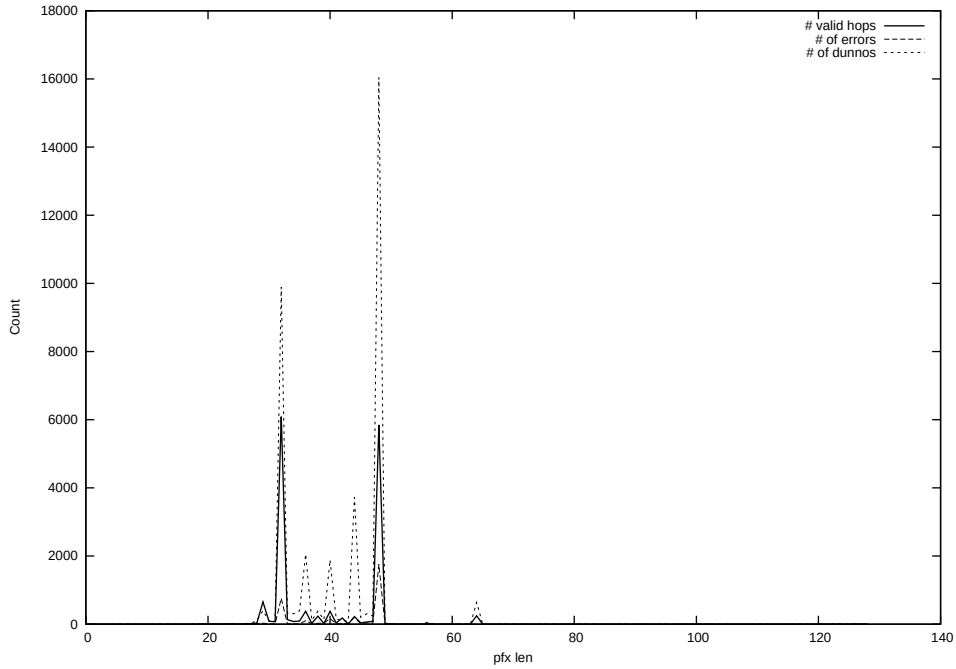
2014-04-11



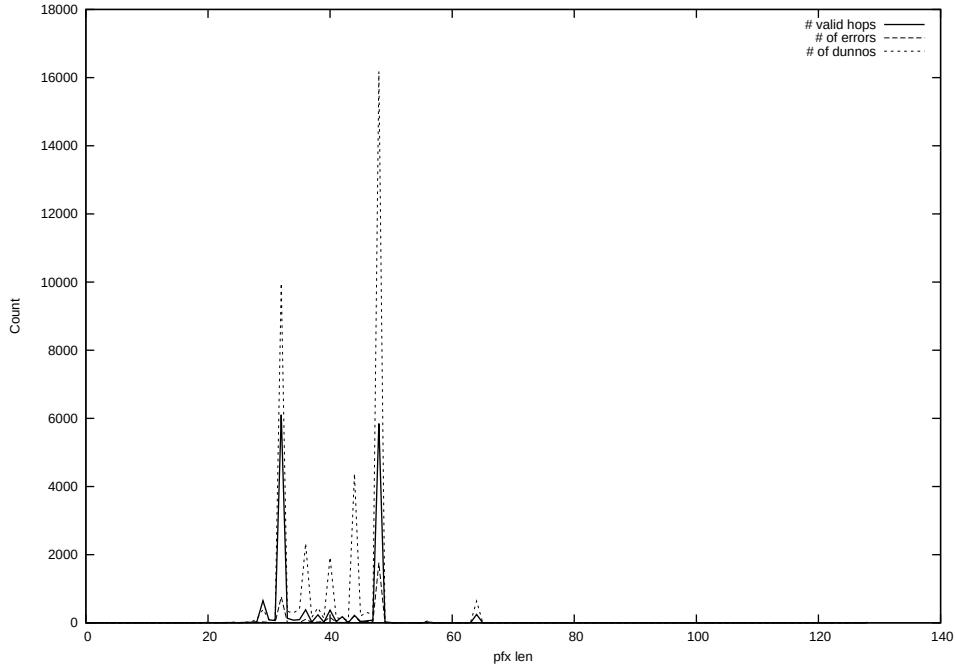
2014-04-12



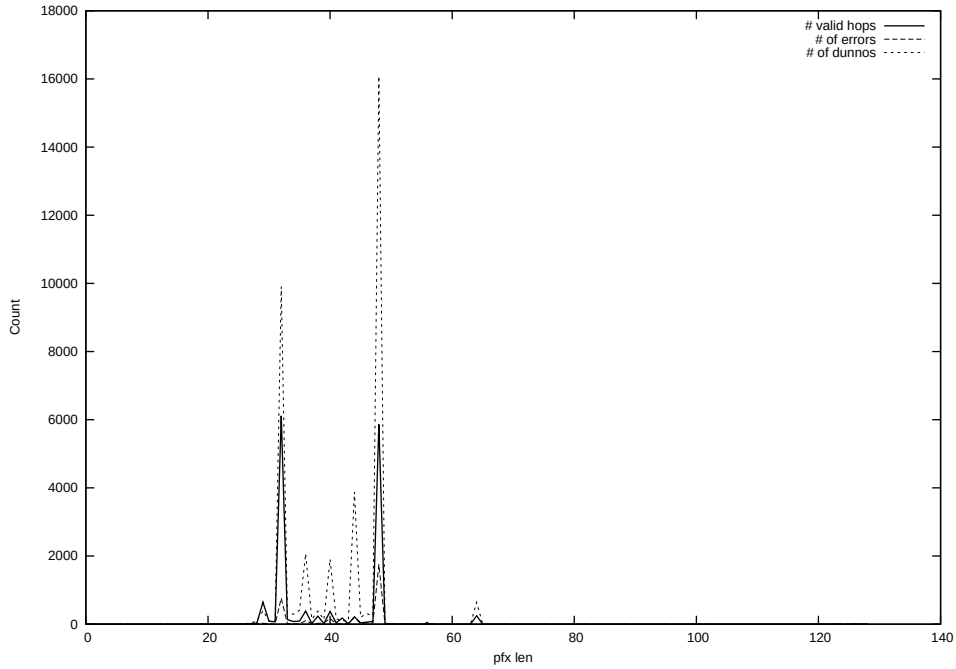
2014-04-13



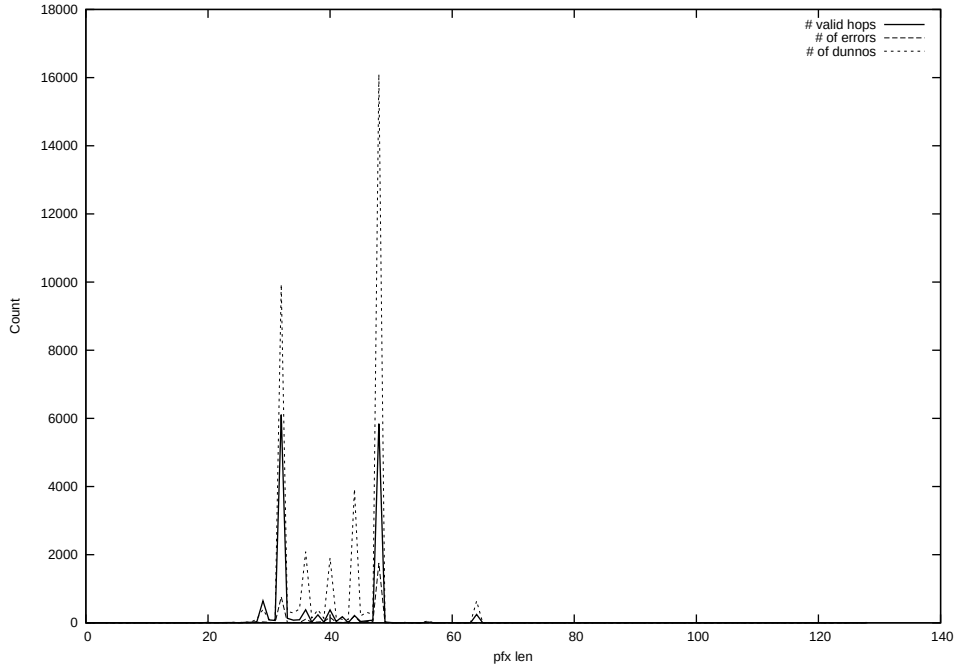
2014-04-14



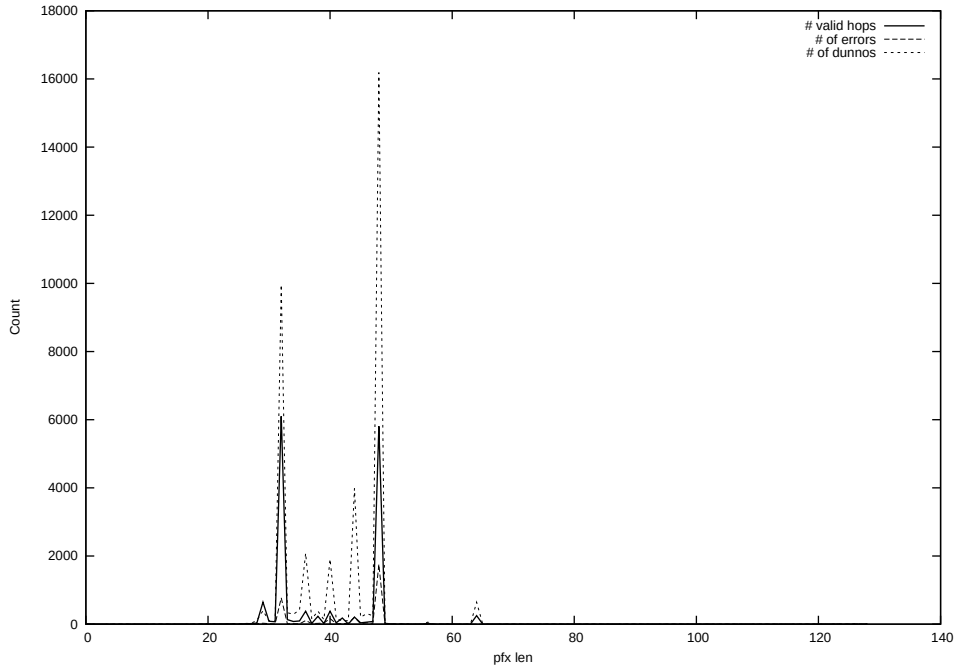
2014-04-15



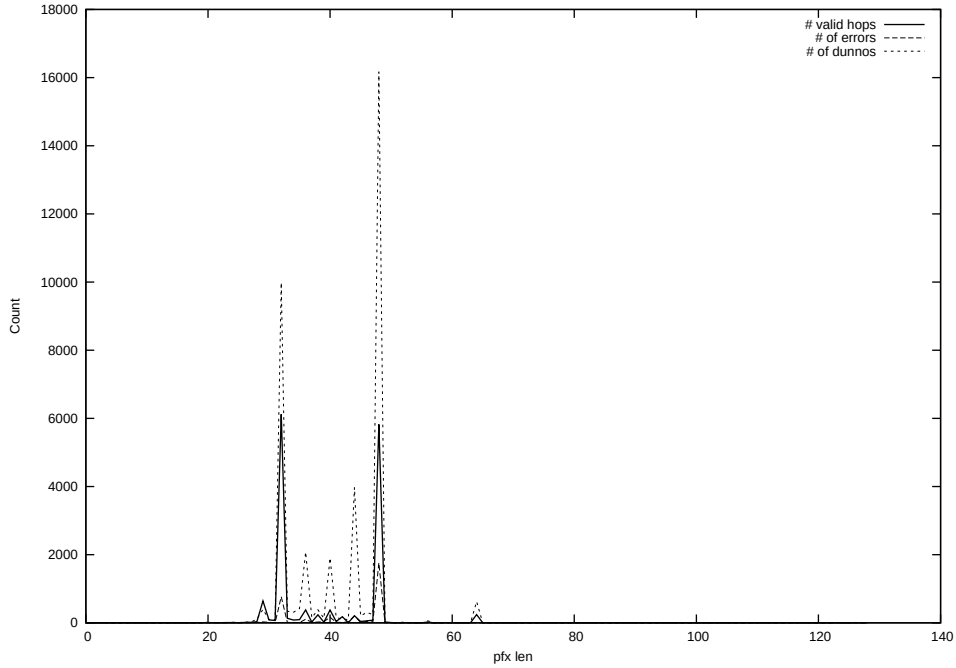
2014-04-16



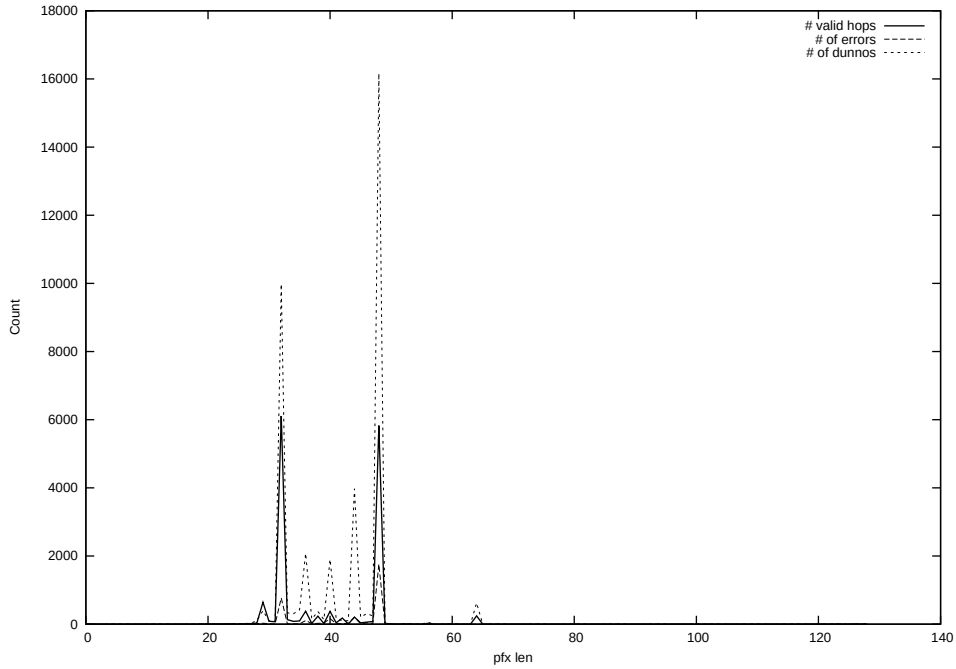
2014-04-17



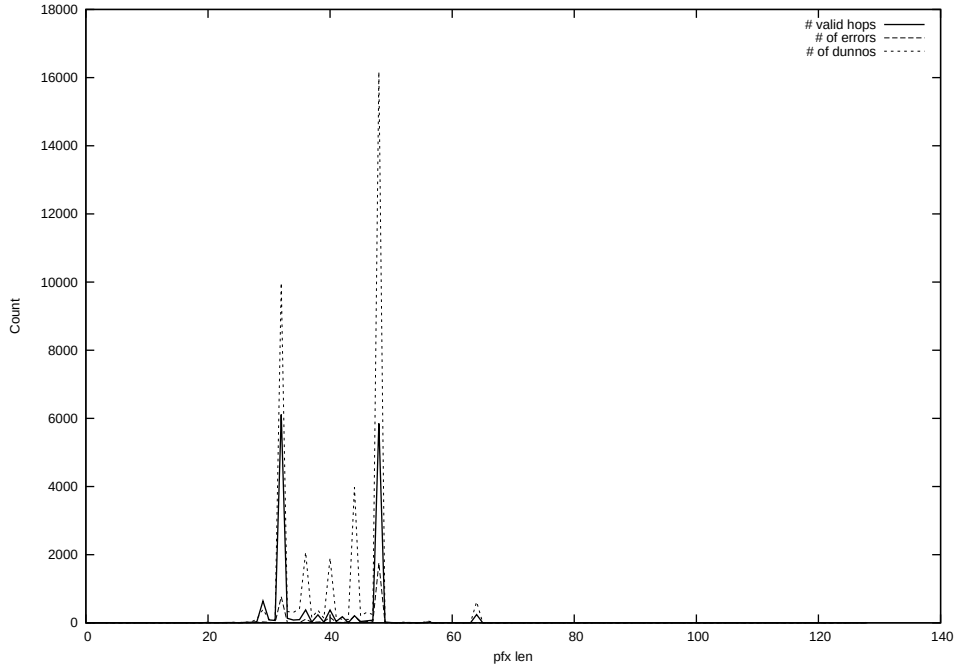
2014-04-18



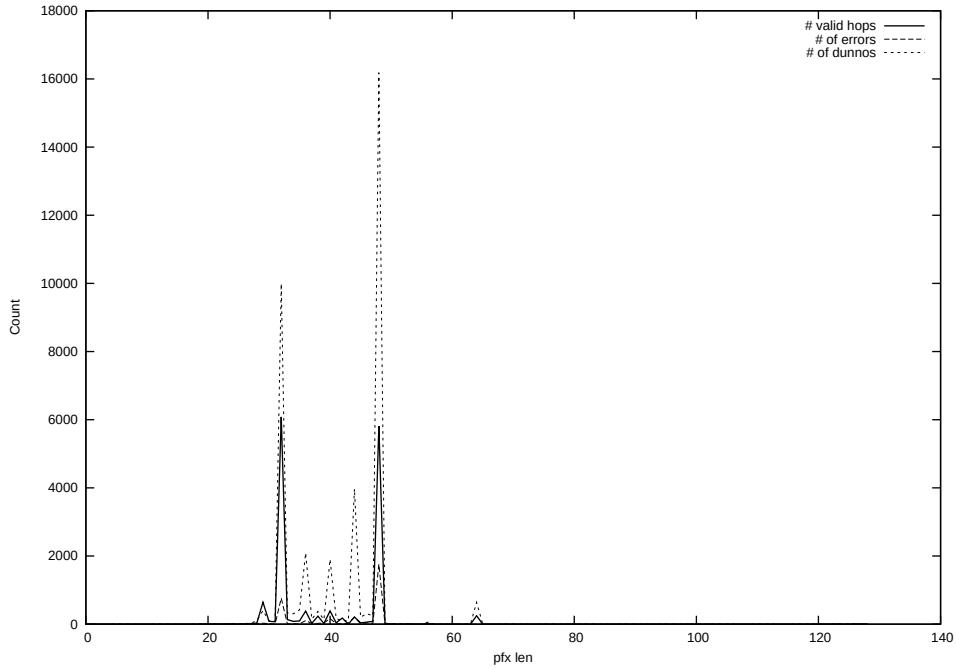
2014-04-19



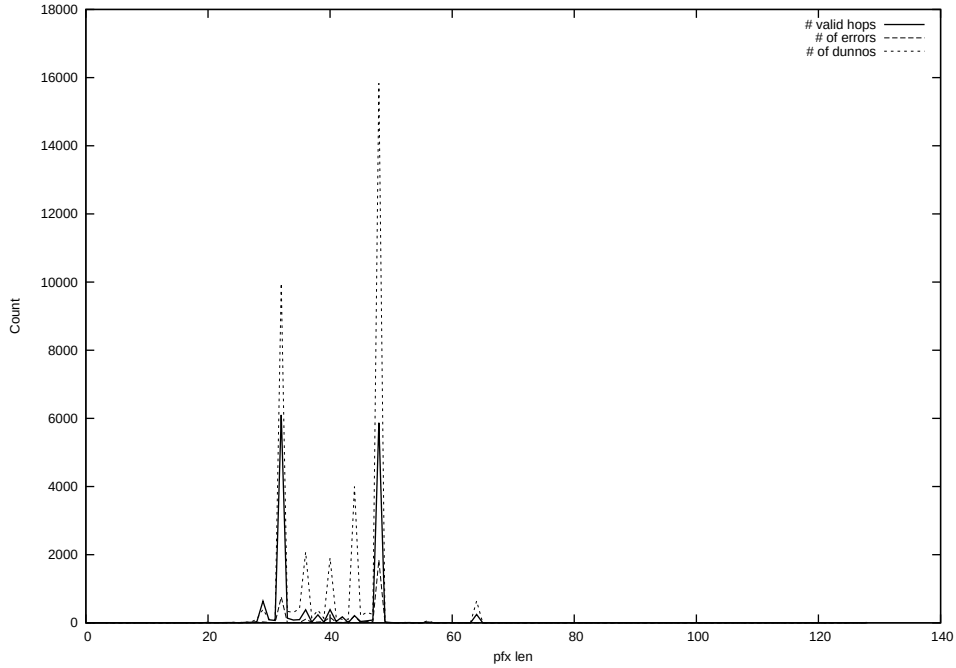
2014-04-20



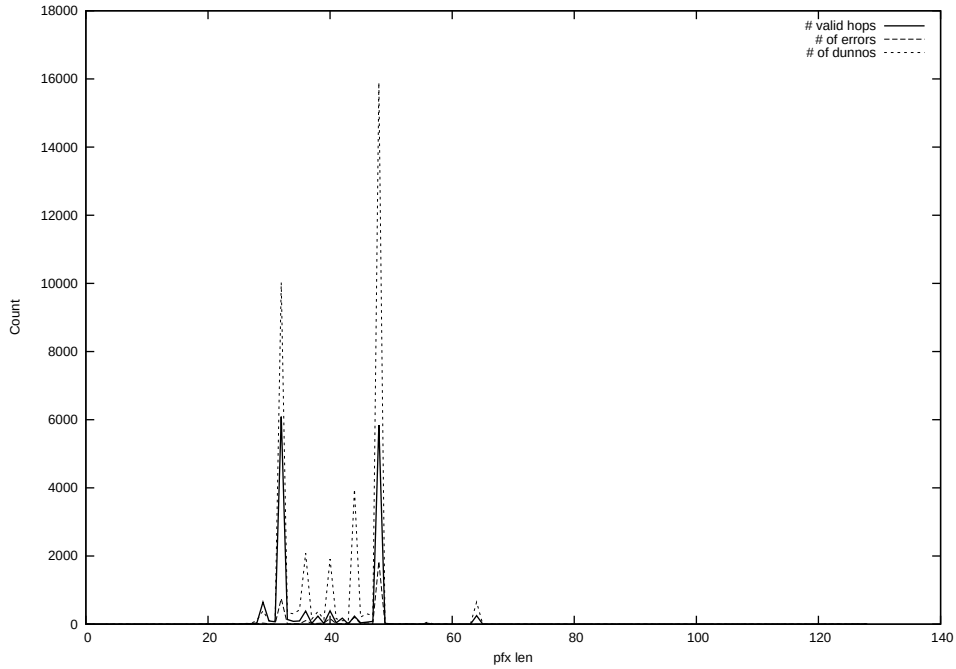
2014-04-21



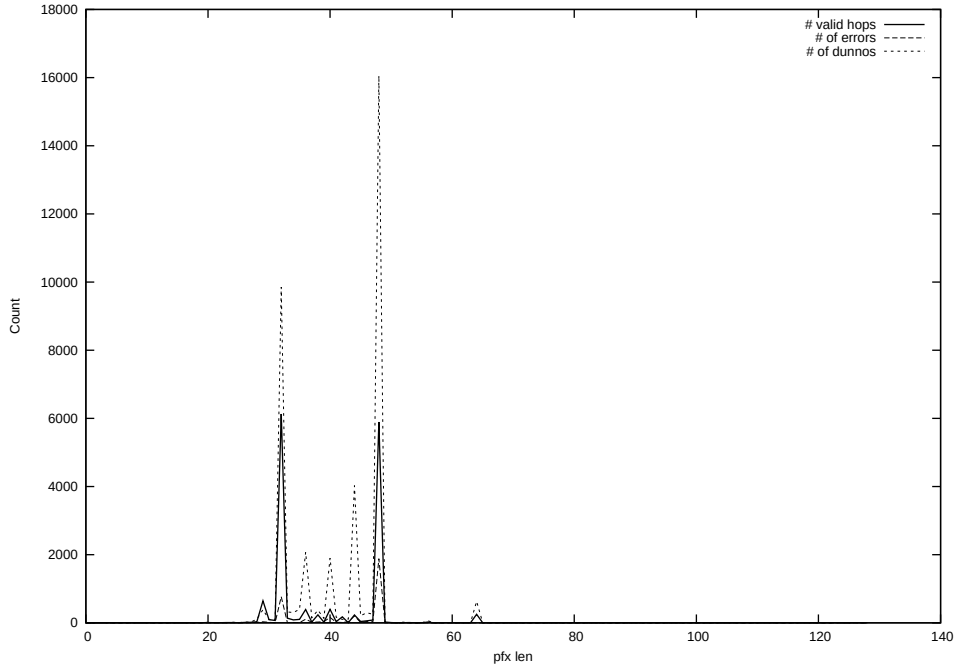
2014-04-22



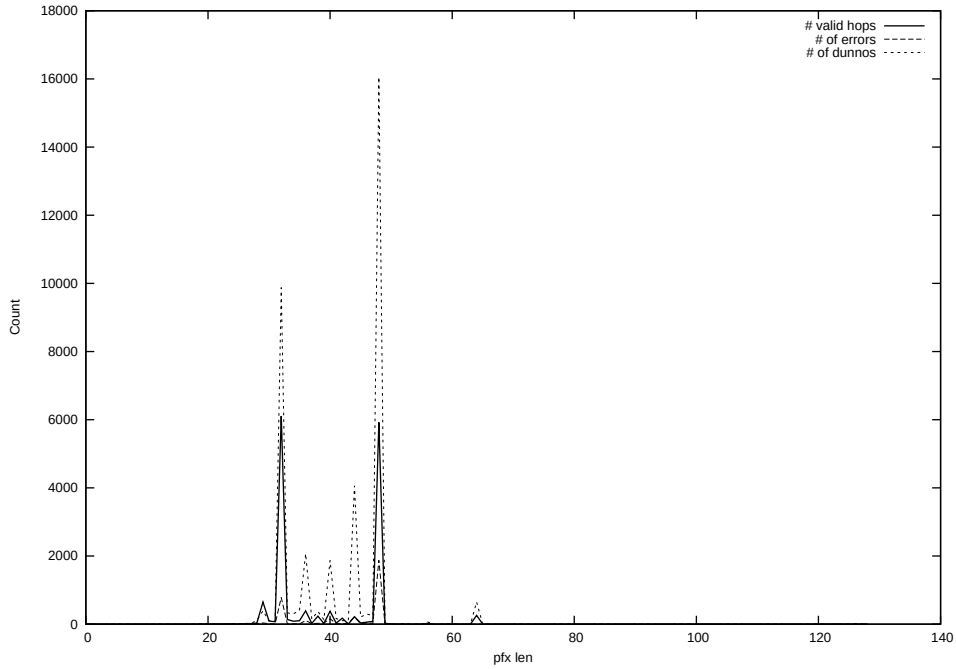
2014-04-23



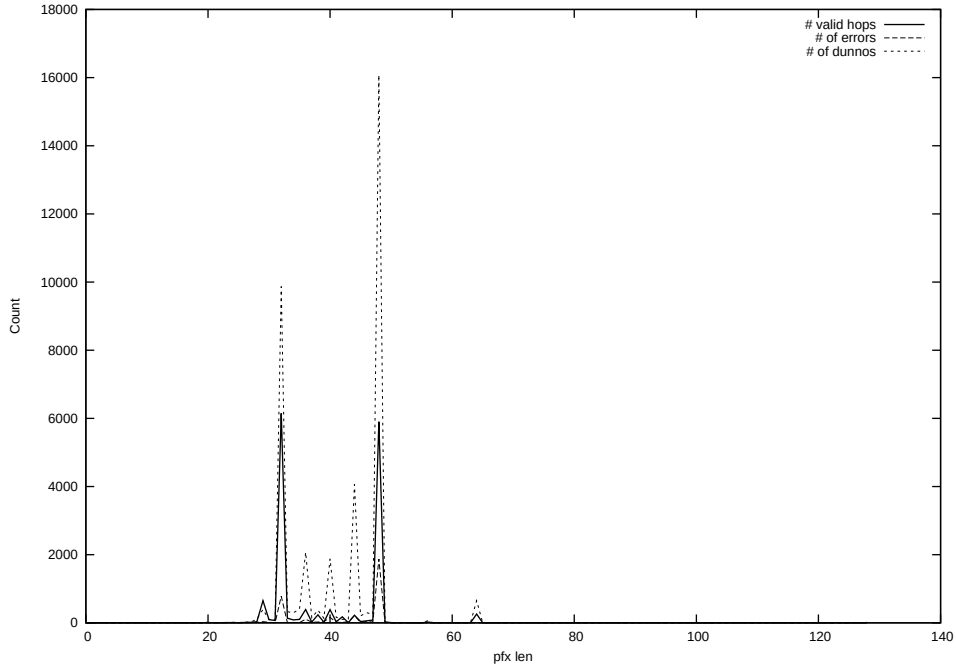
2014-04-24



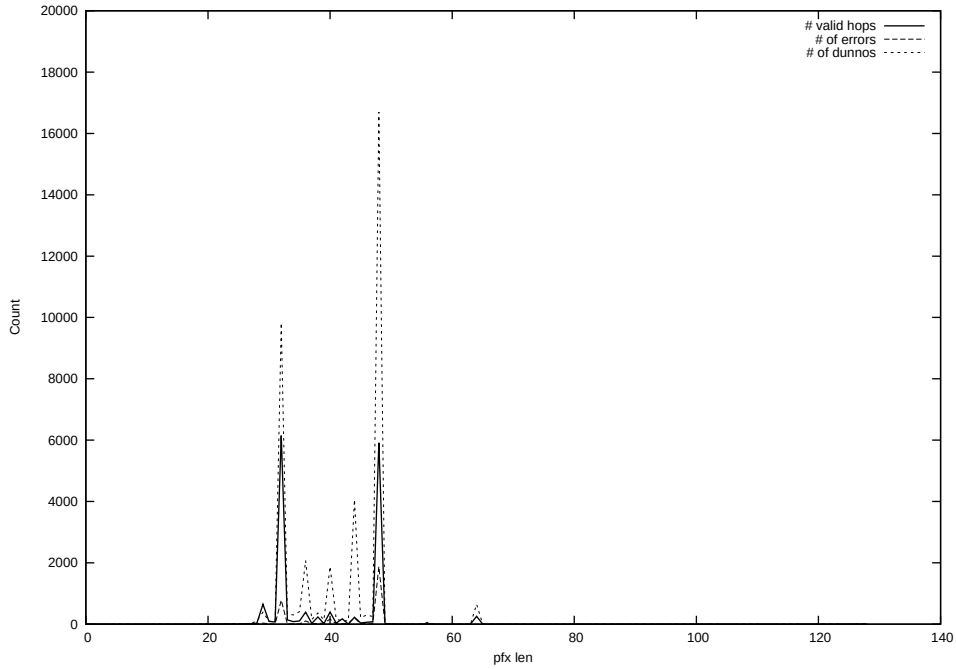
2014-04-25



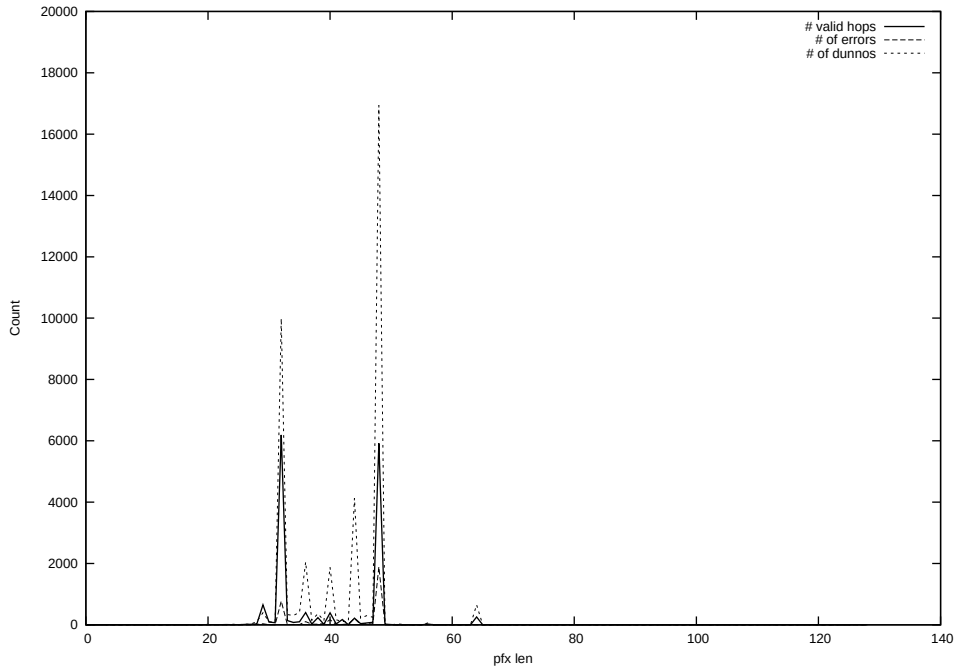
2014-04-26



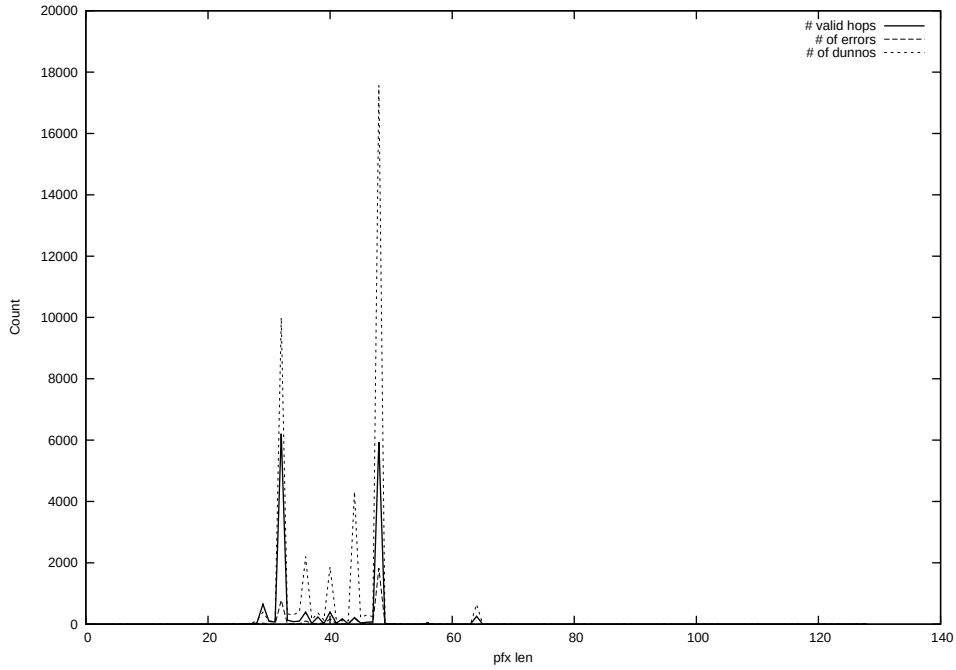
2014-04-27



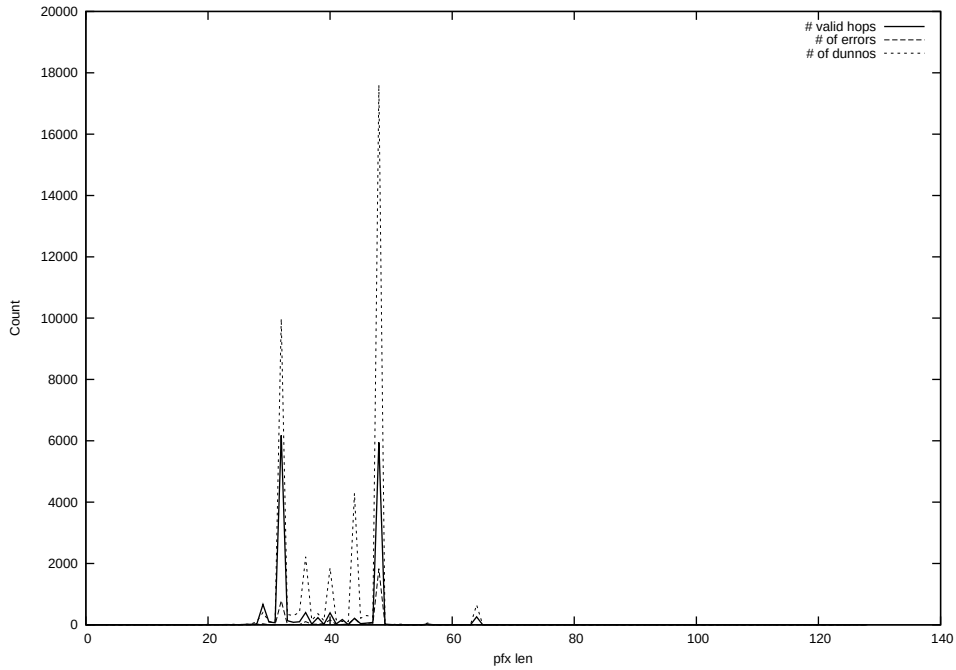
2014-04-28



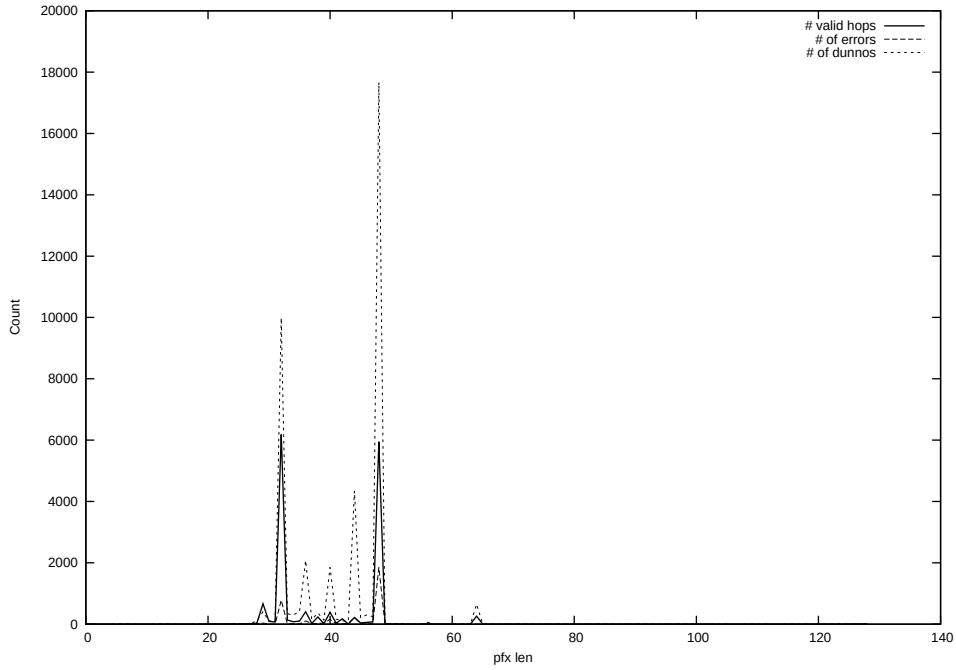
2014-04-29



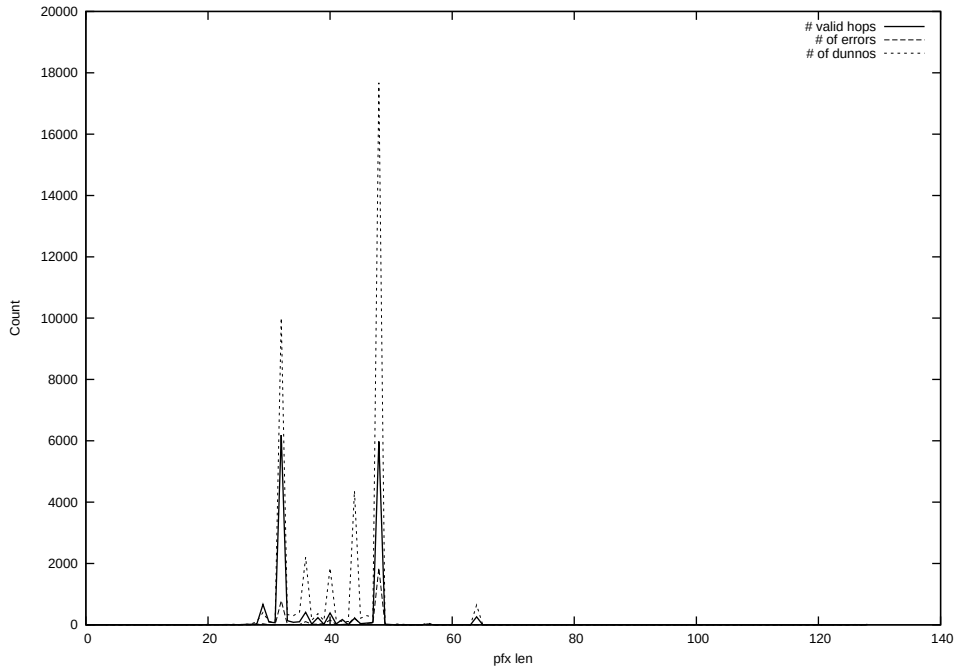
2014-04-30



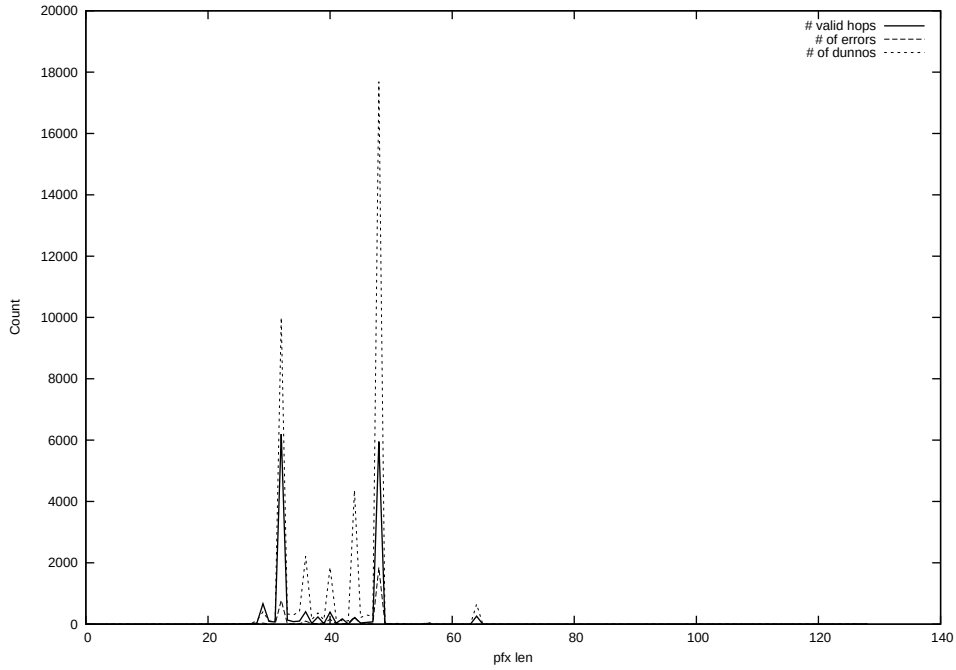
2014-05-01



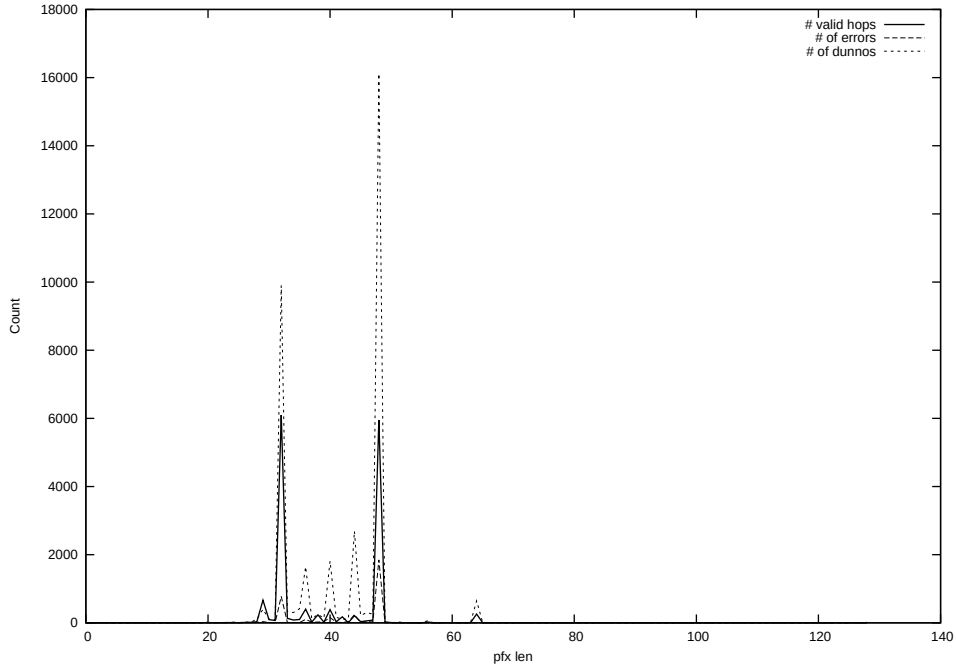
2014-05-02



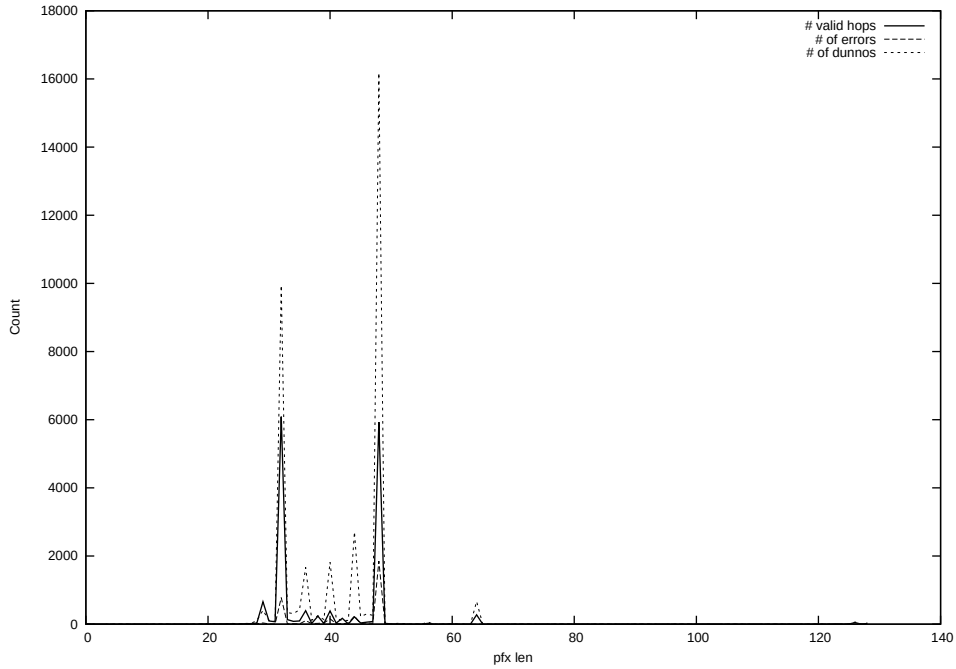
2014-05-03



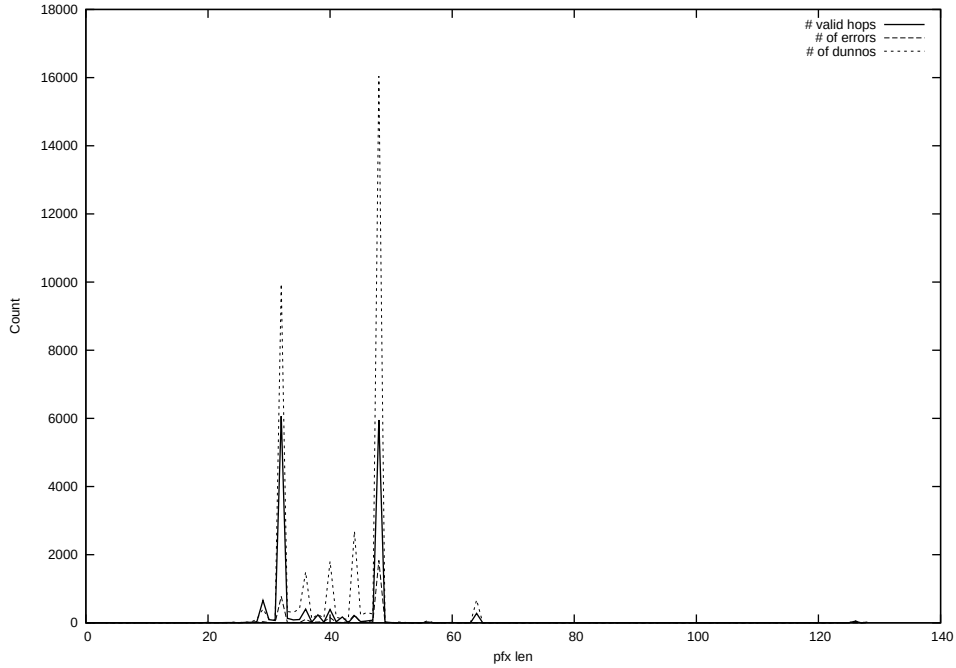
2014-05-04



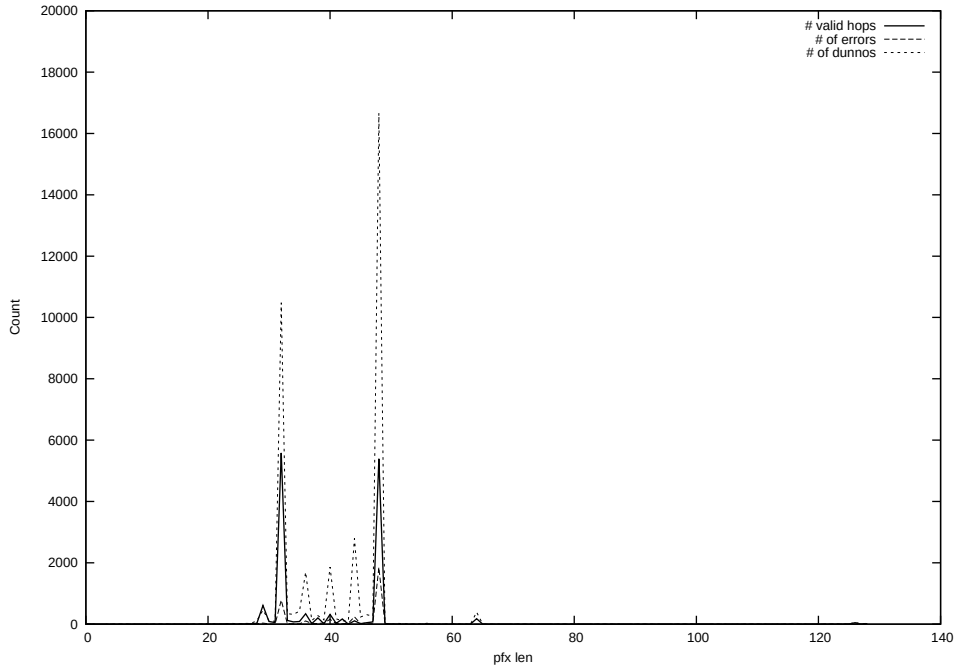
2014-05-05



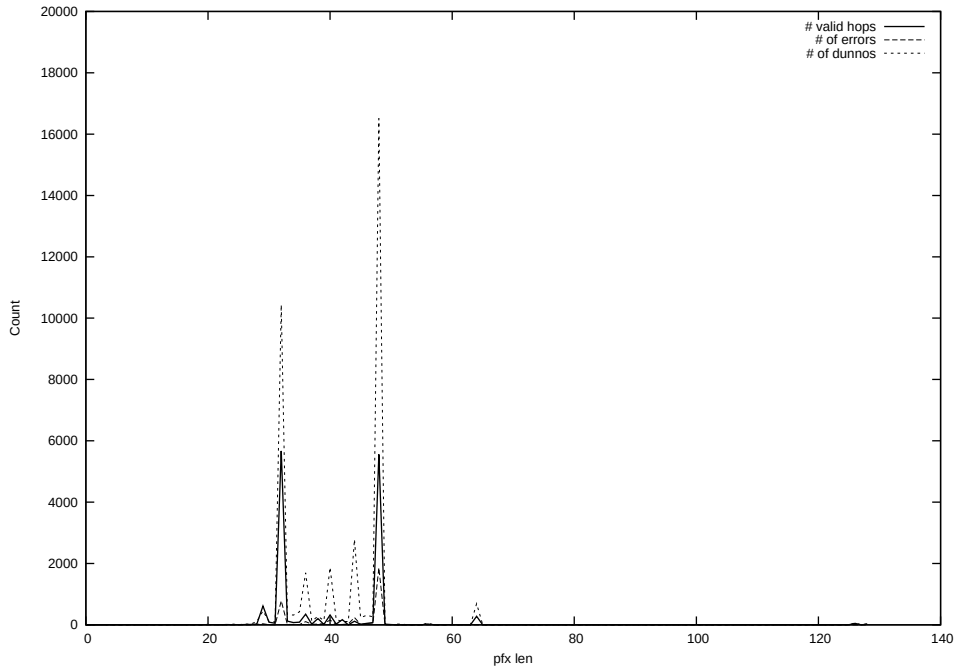
2014-05-06



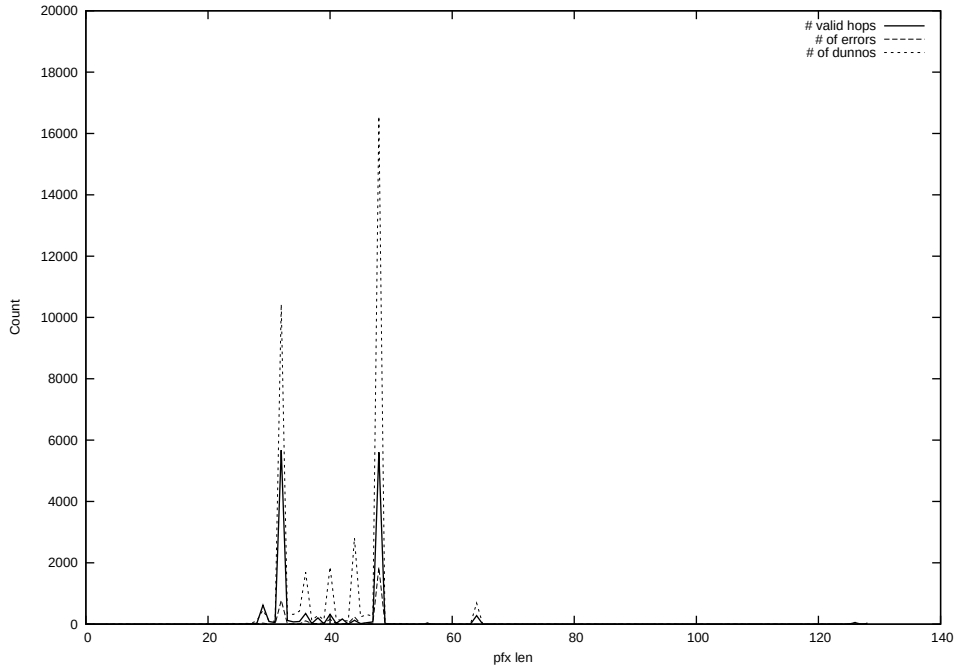
2014-05-07



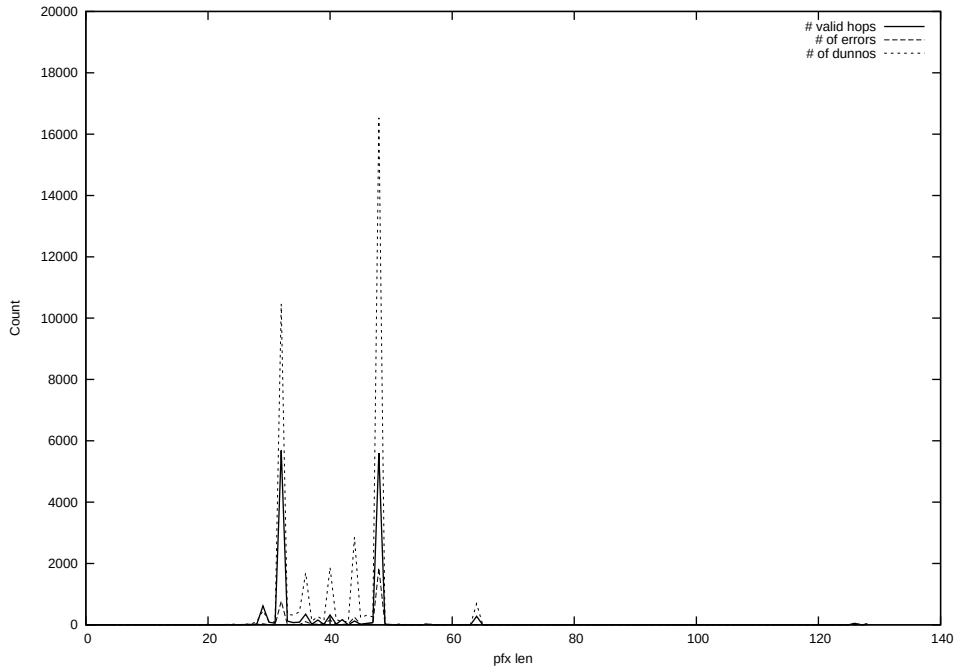
2014-05-08



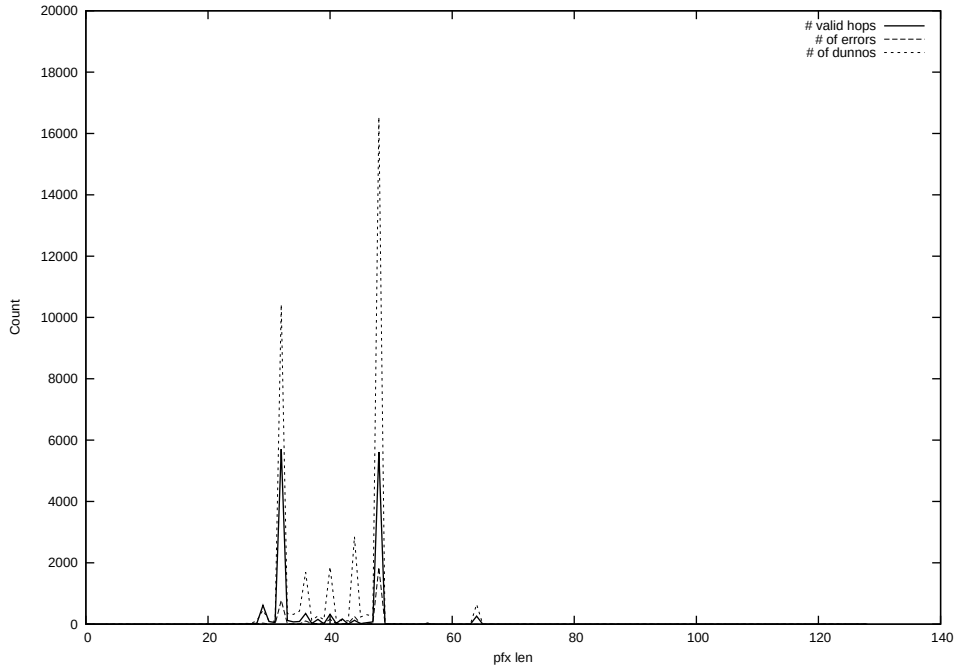
2014-05-09



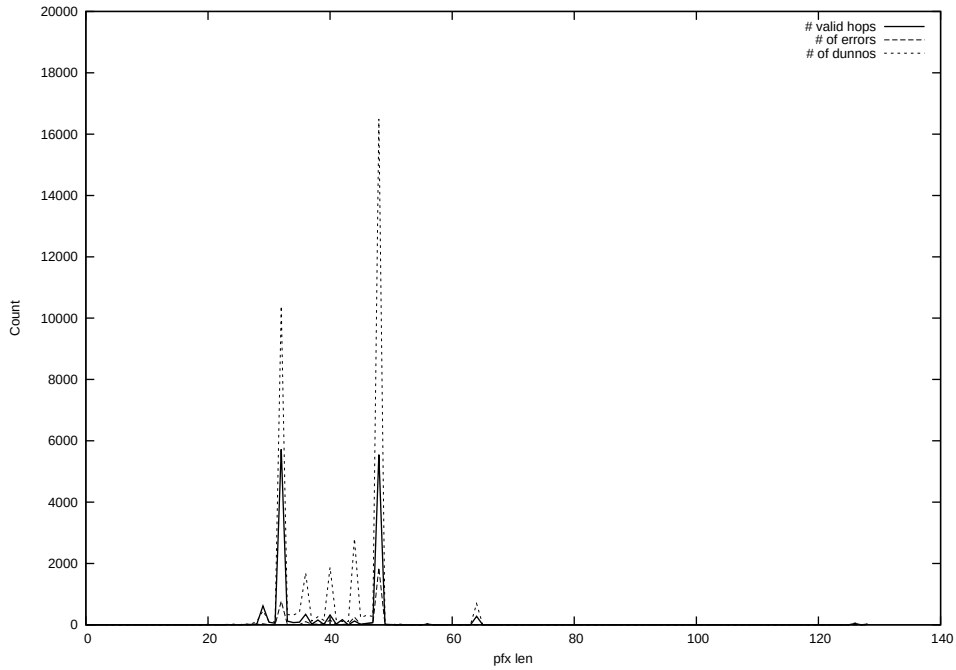
2014-05-10



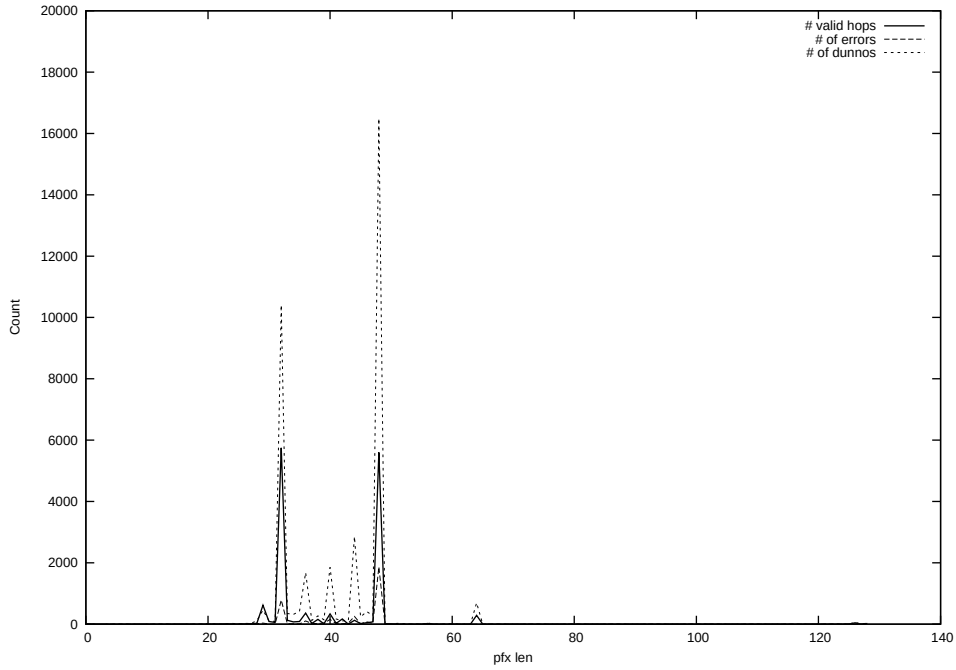
2014-05-11



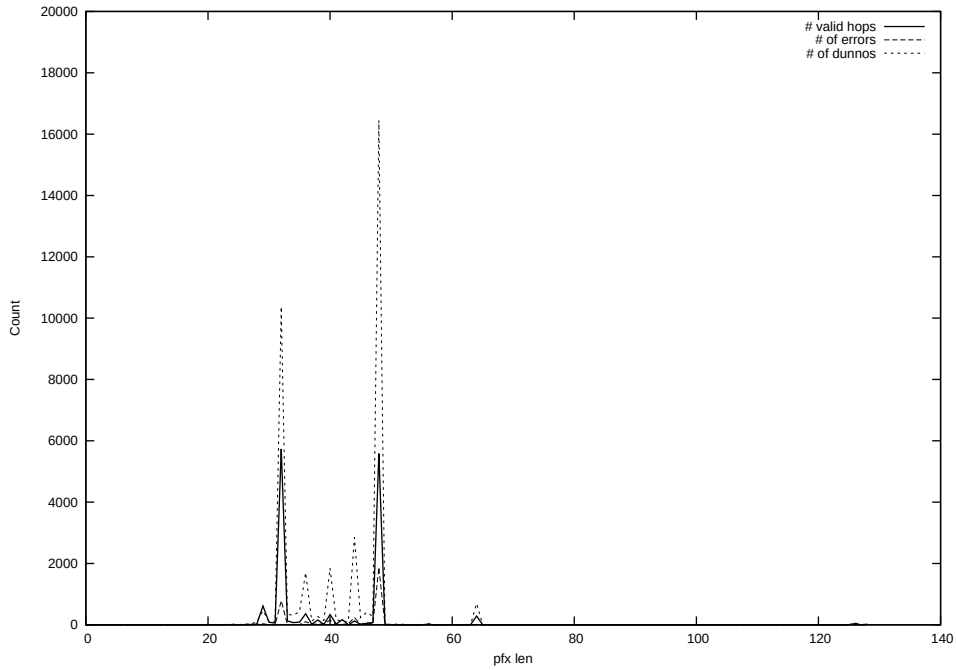
2014-05-12



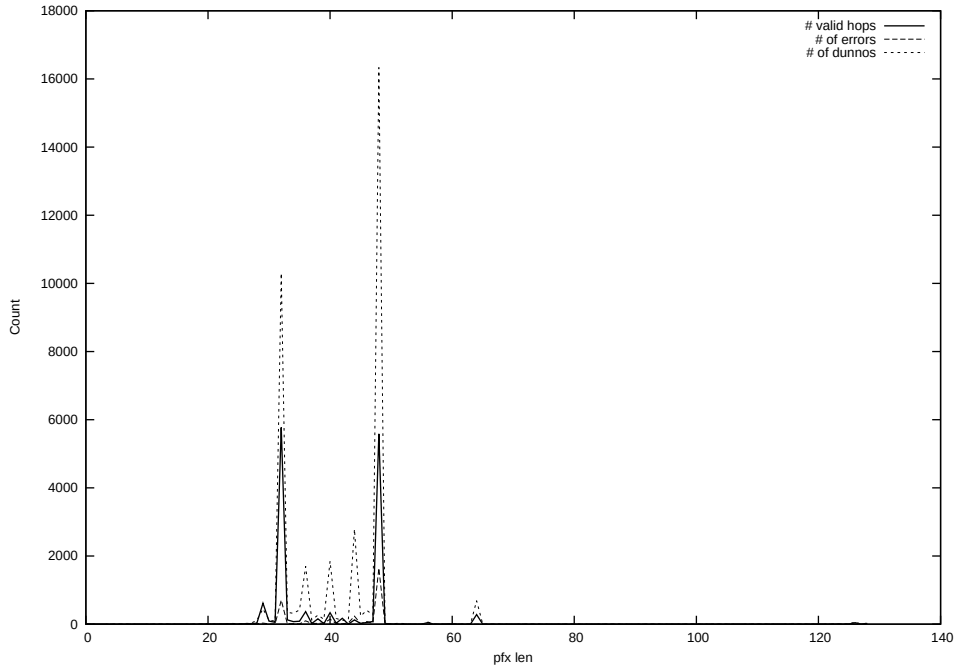
2014-05-13



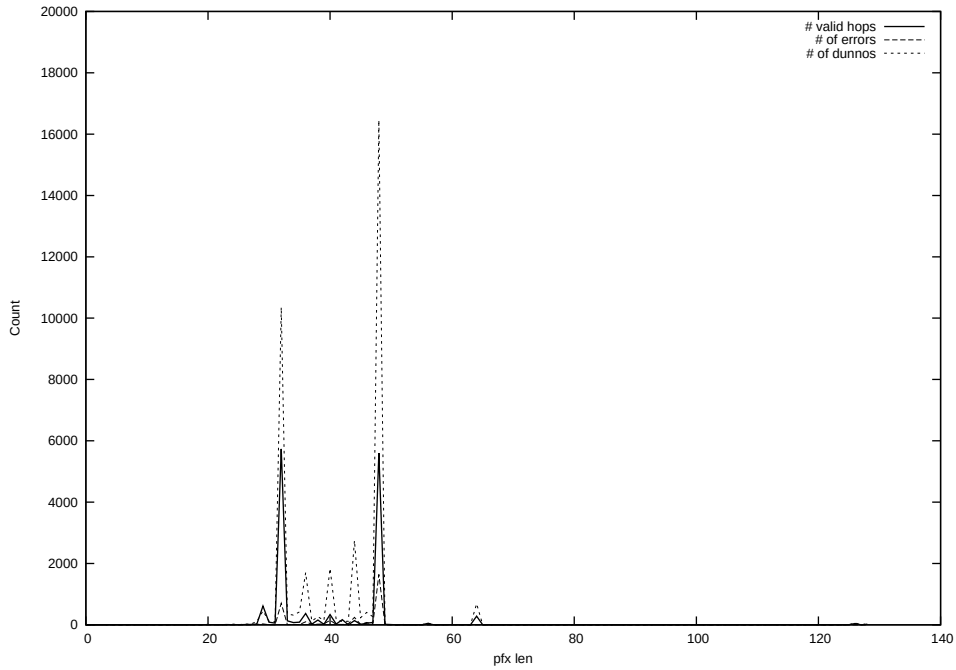
2014-05-14



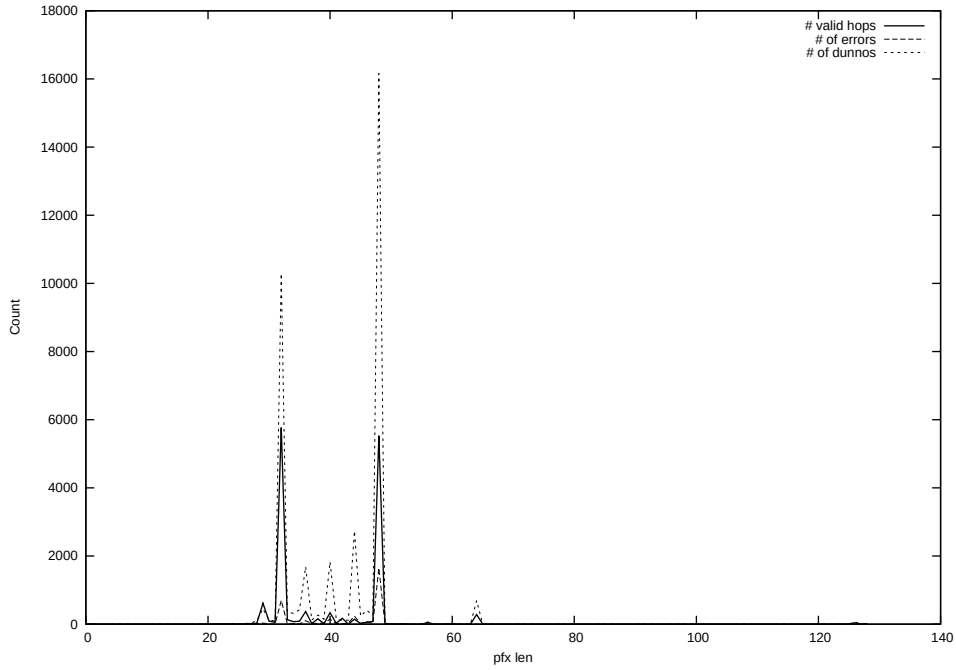
2014-05-15



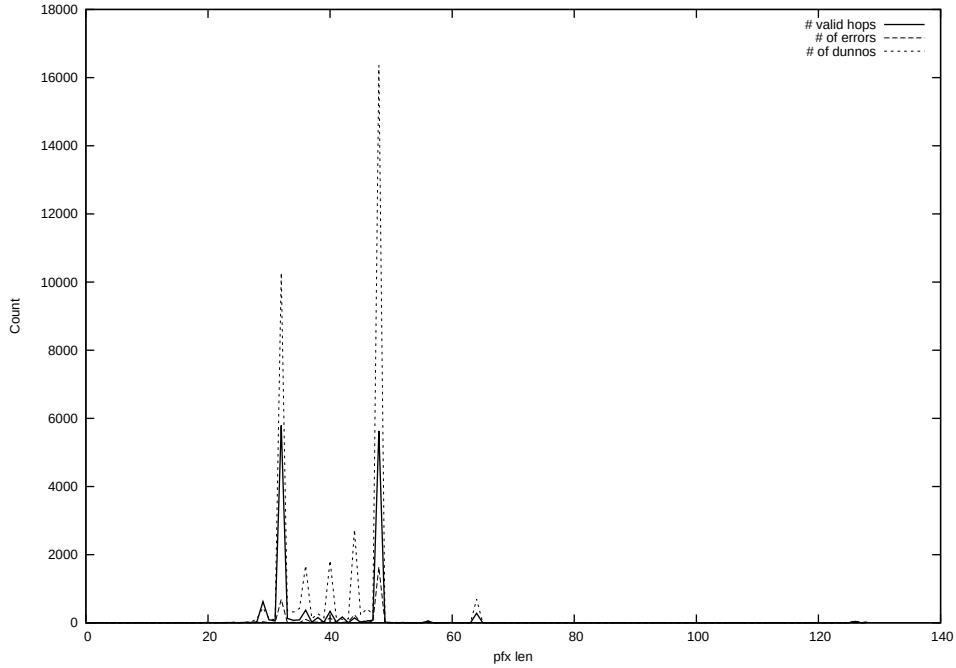
2014-05-16



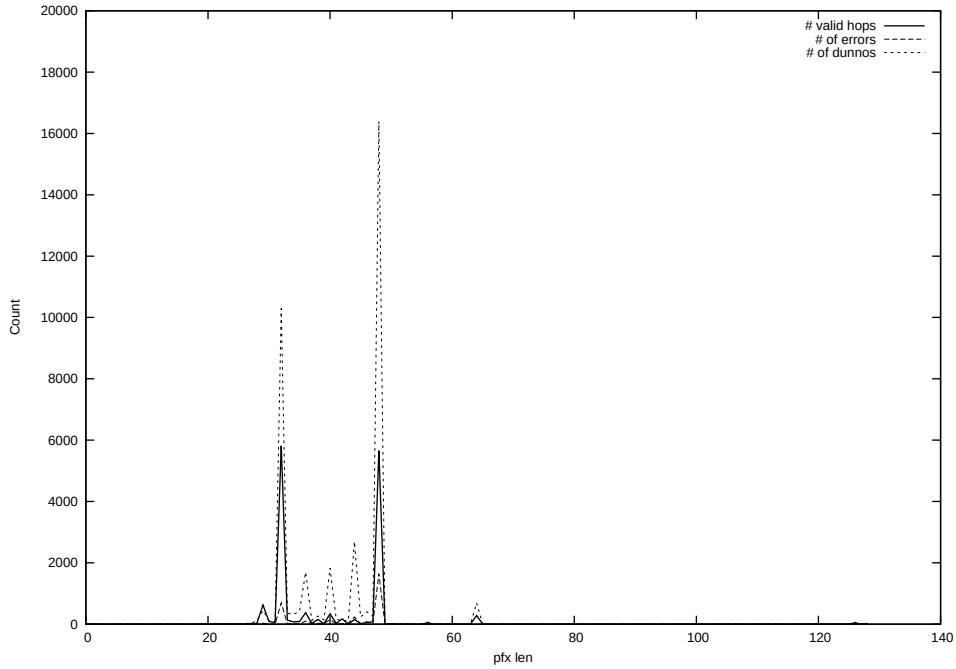
2014-05-17



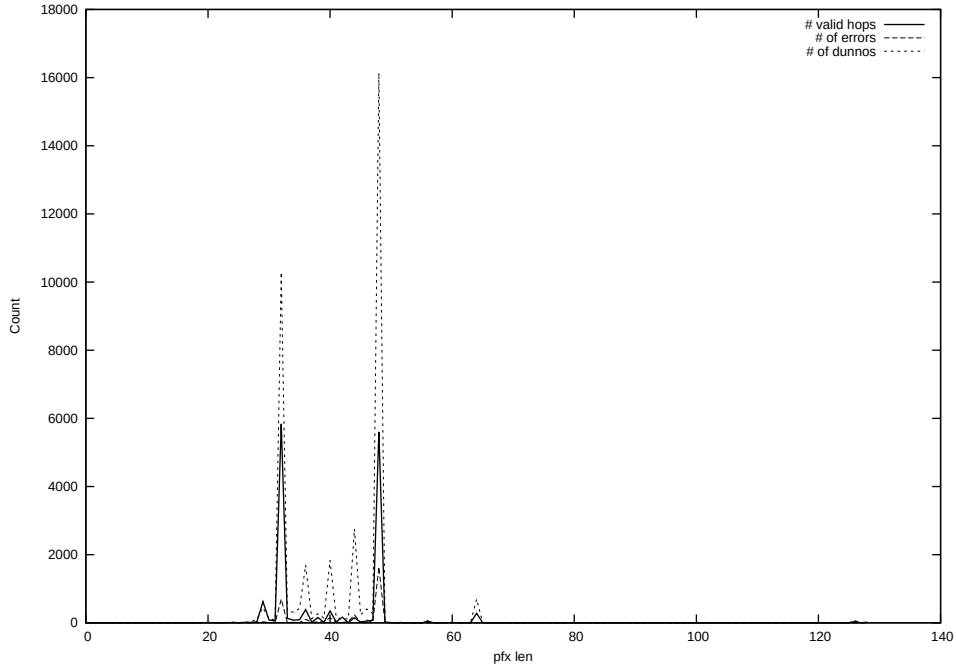
2014-05-18



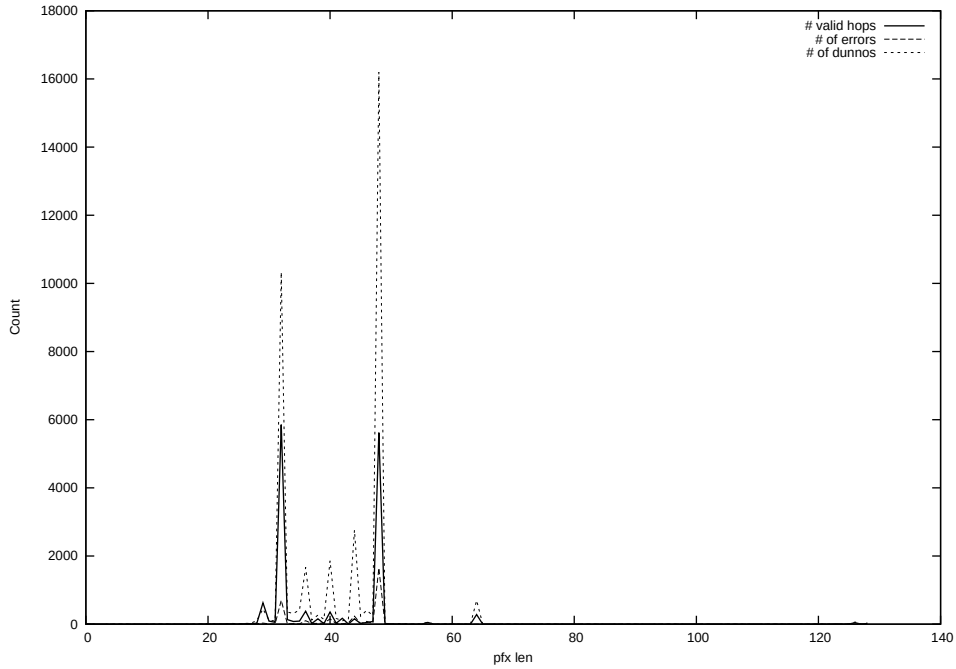
2014-05-19



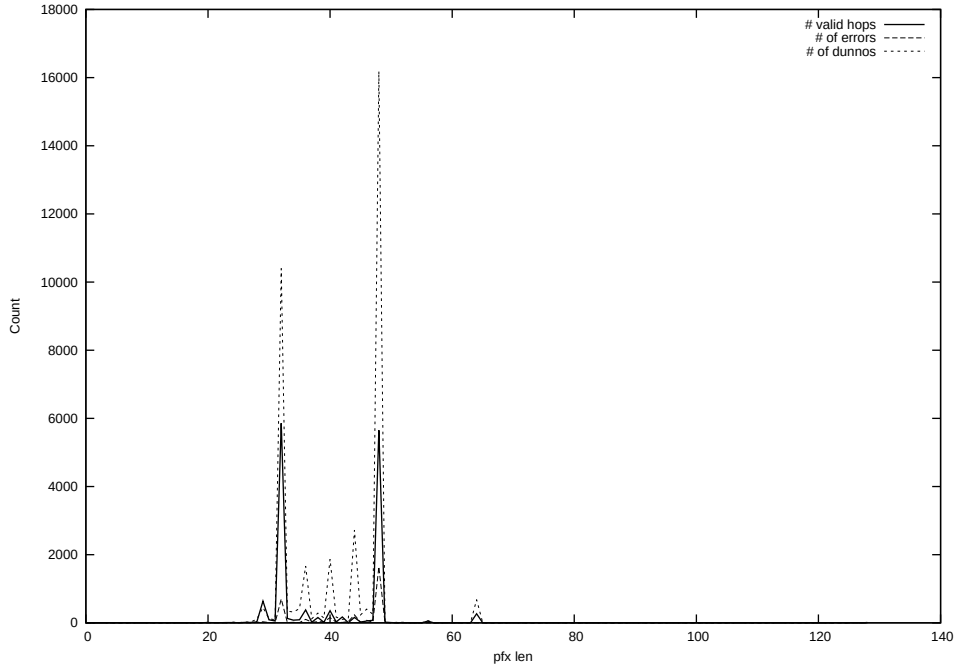
2014-05-20



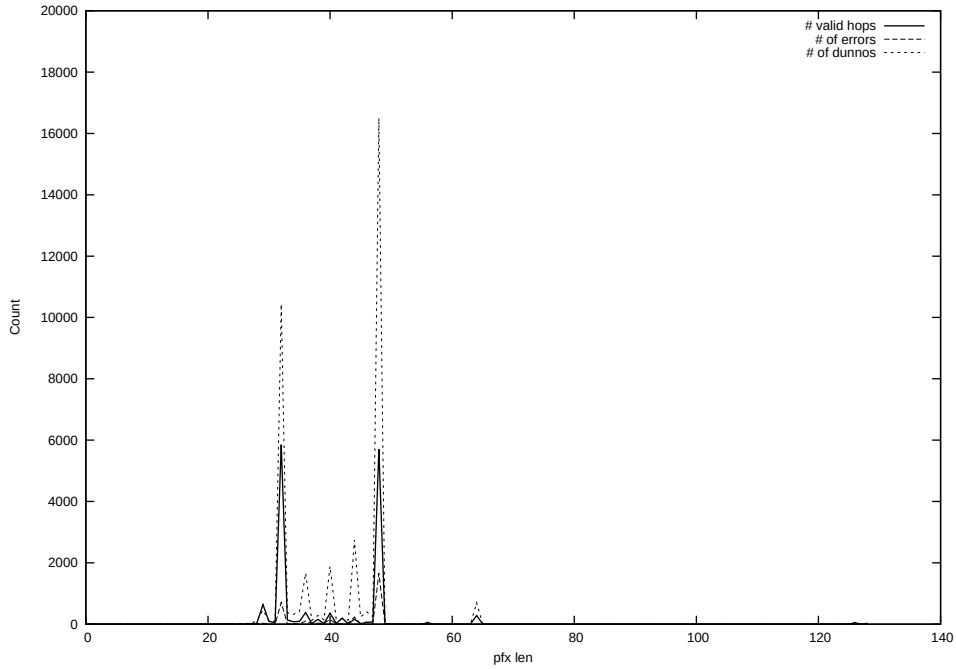
2014-05-21



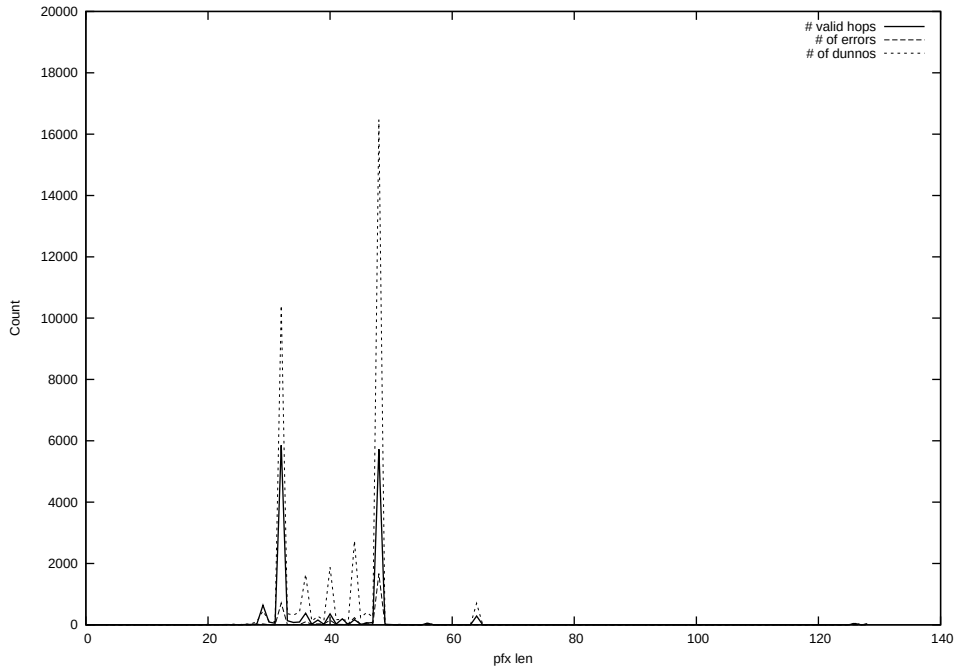
2014-05-22



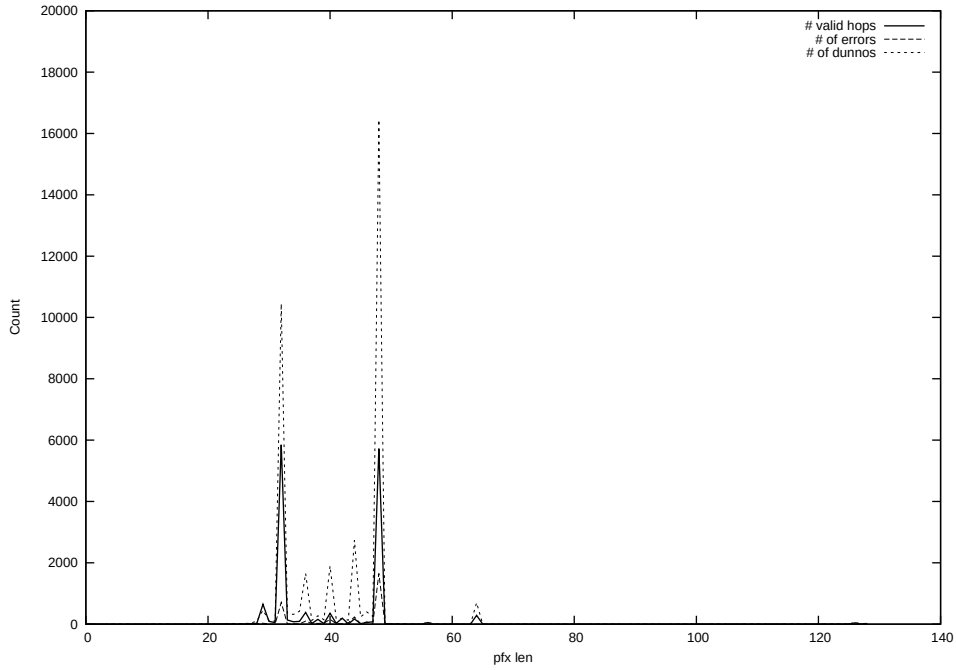
2014-05-23



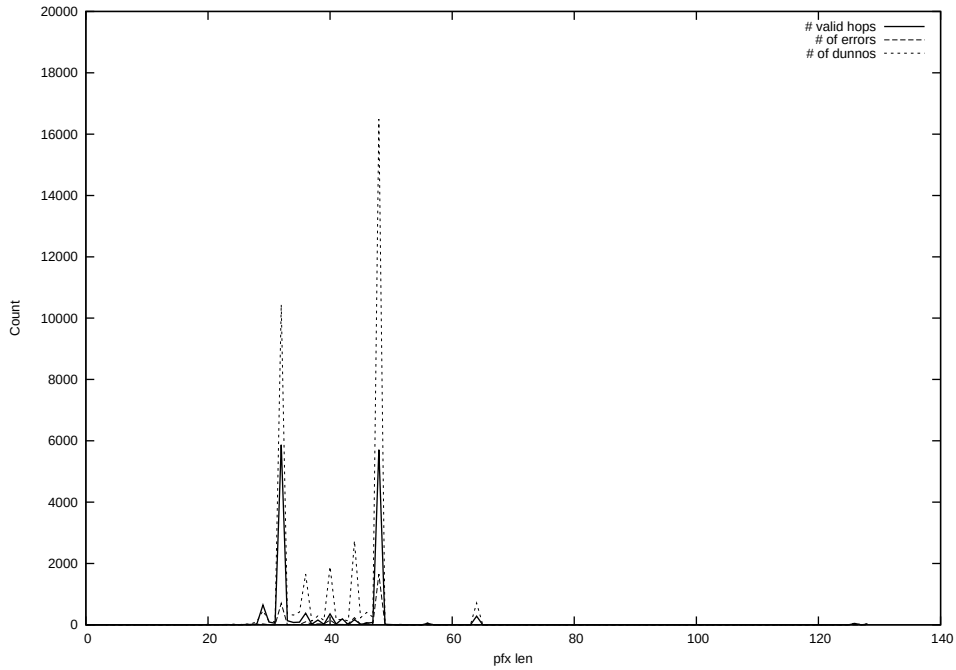
2014-05-24



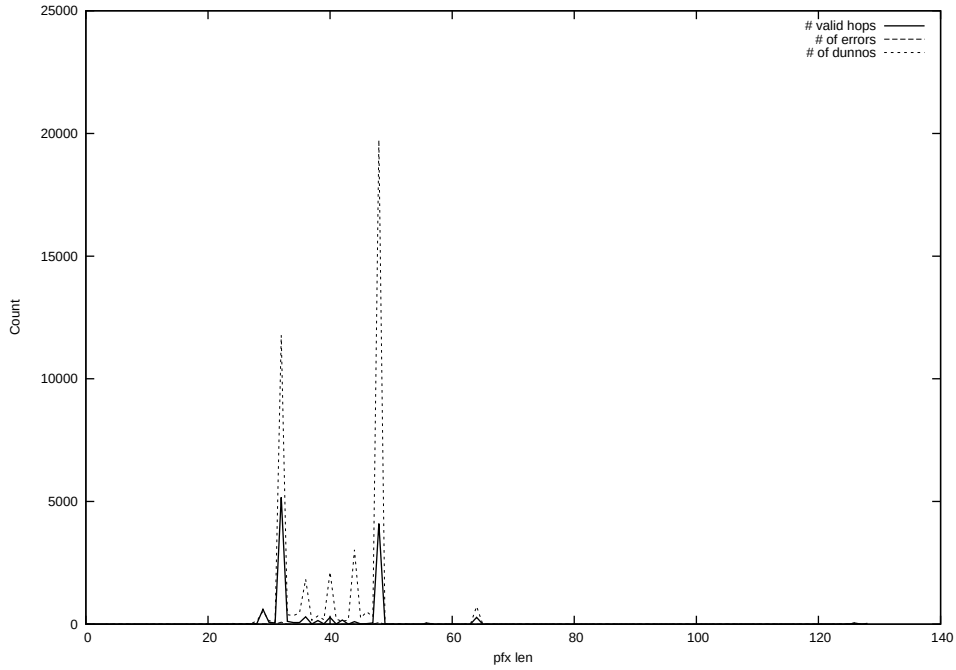
2014-05-25



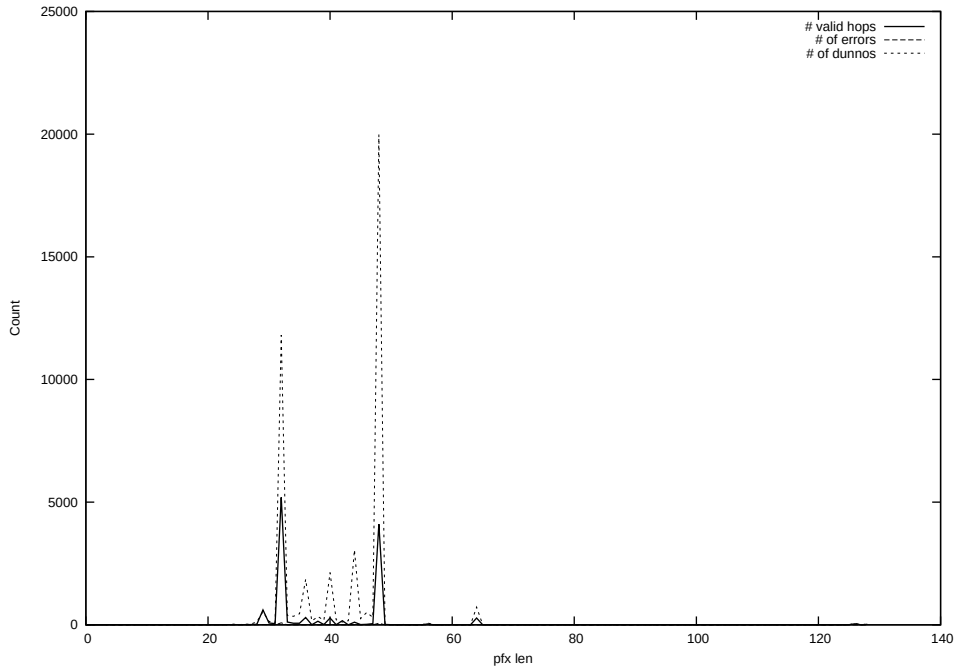
2014-05-26



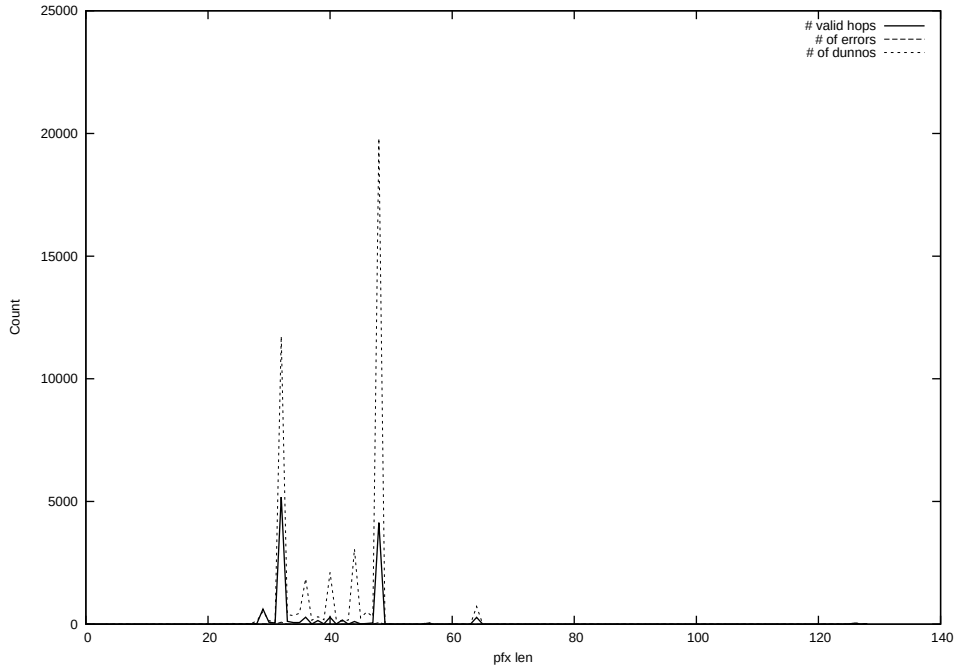
2014-05-27



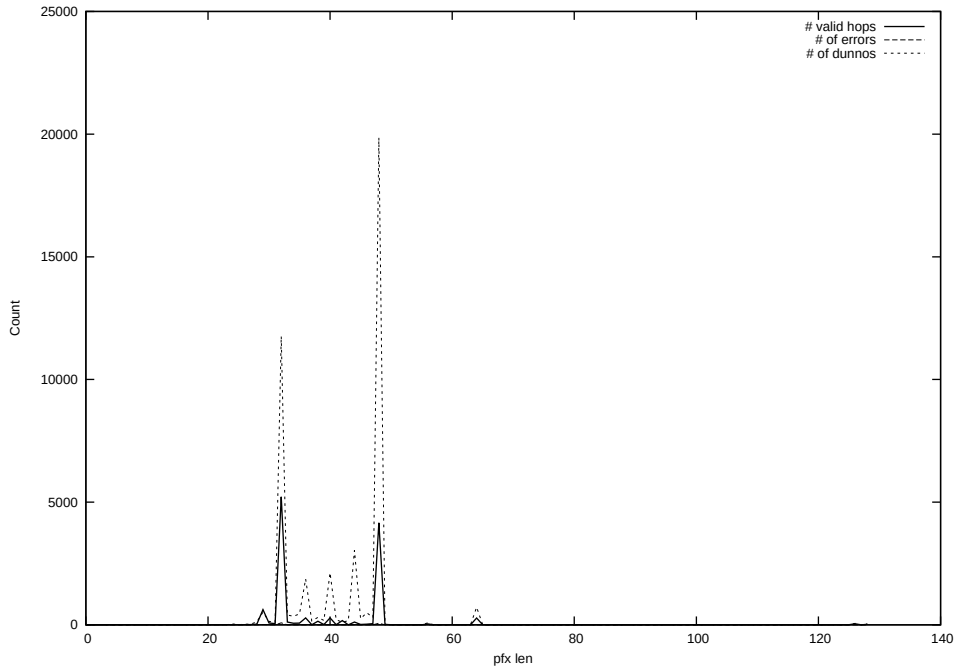
2014-05-28



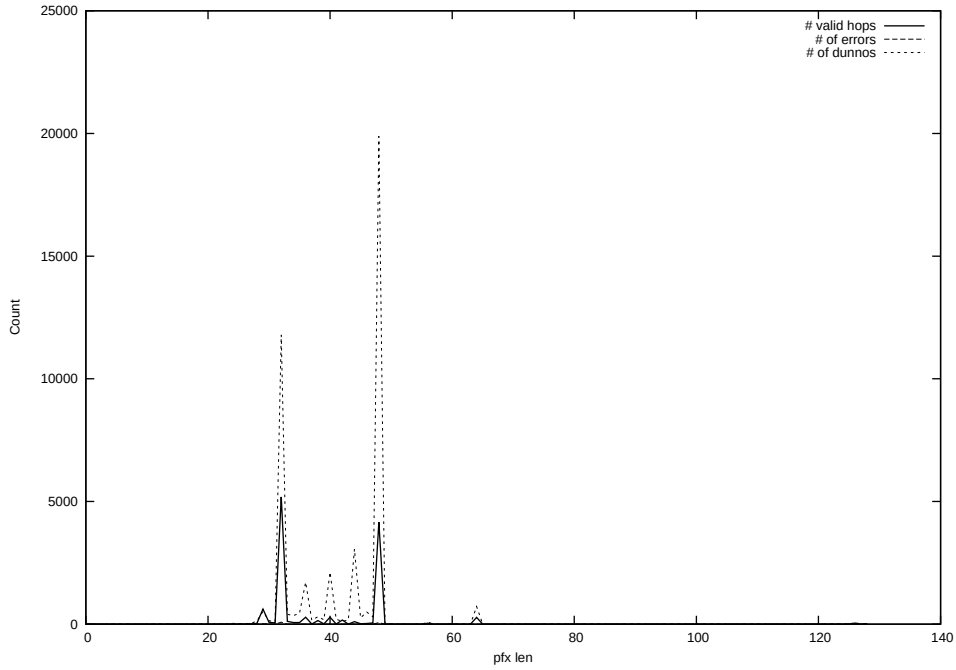
2014-05-29



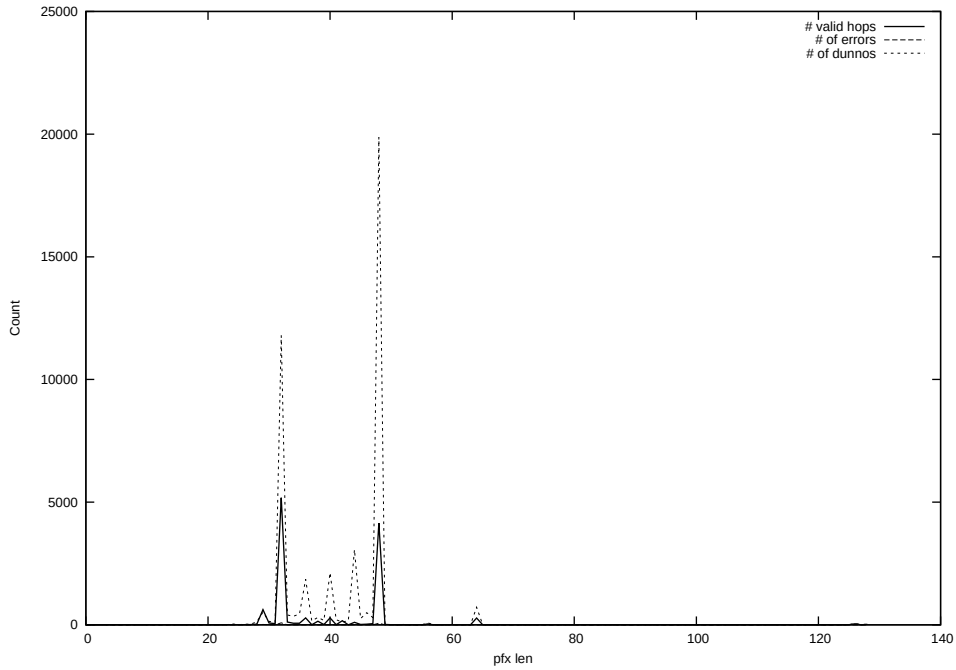
2014-05-30



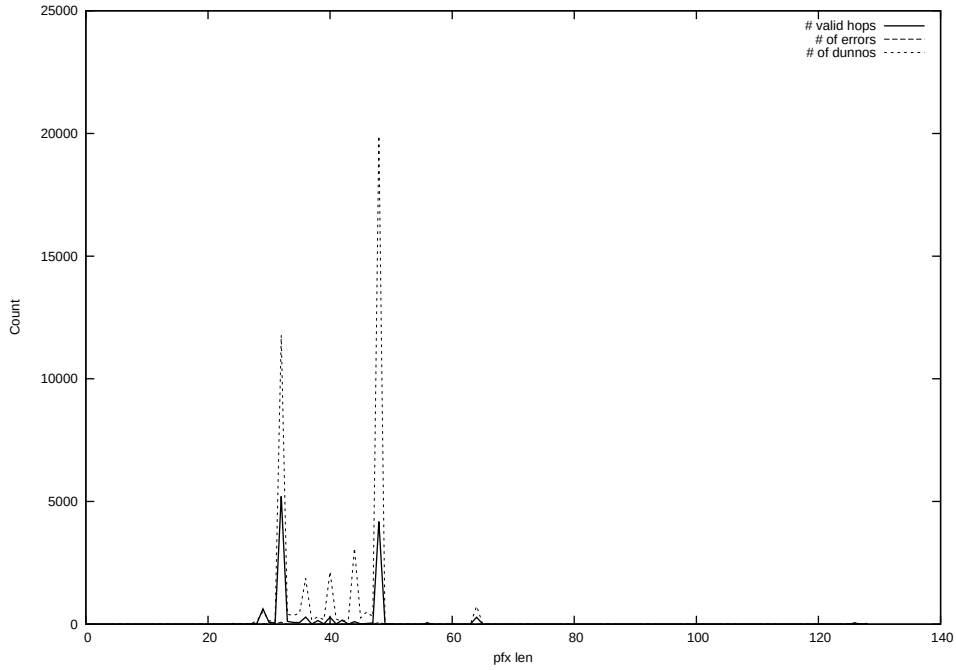
2014-05-31



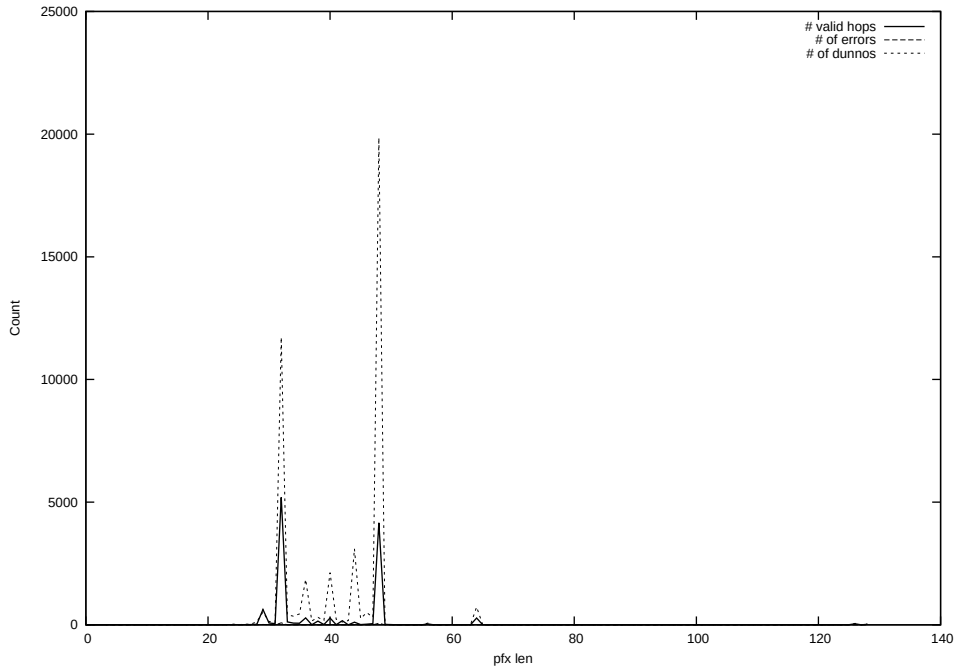
2014-06-01



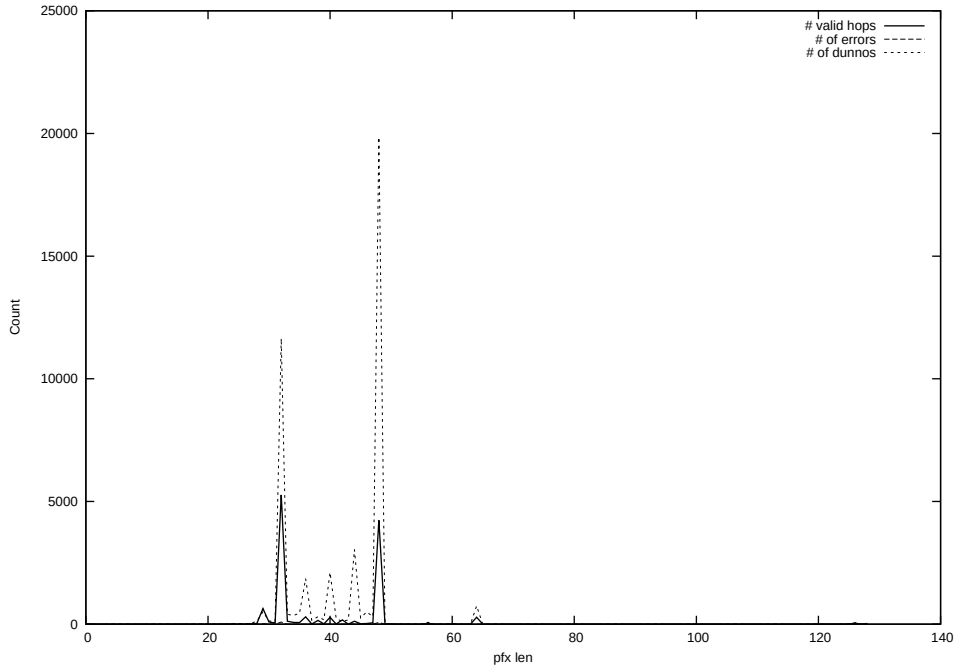
2014-06-02



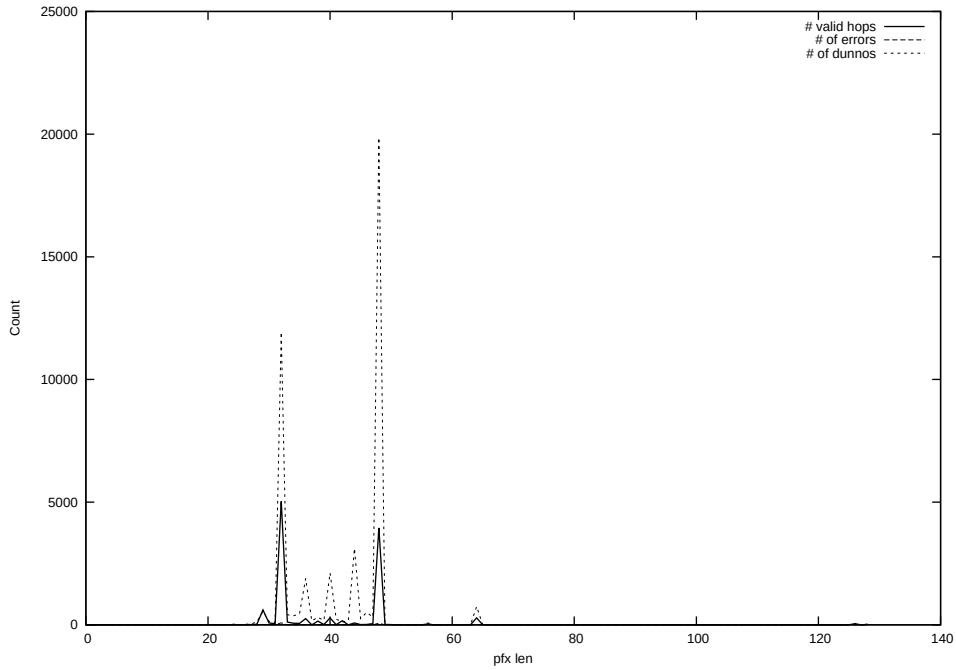
2014-06-03



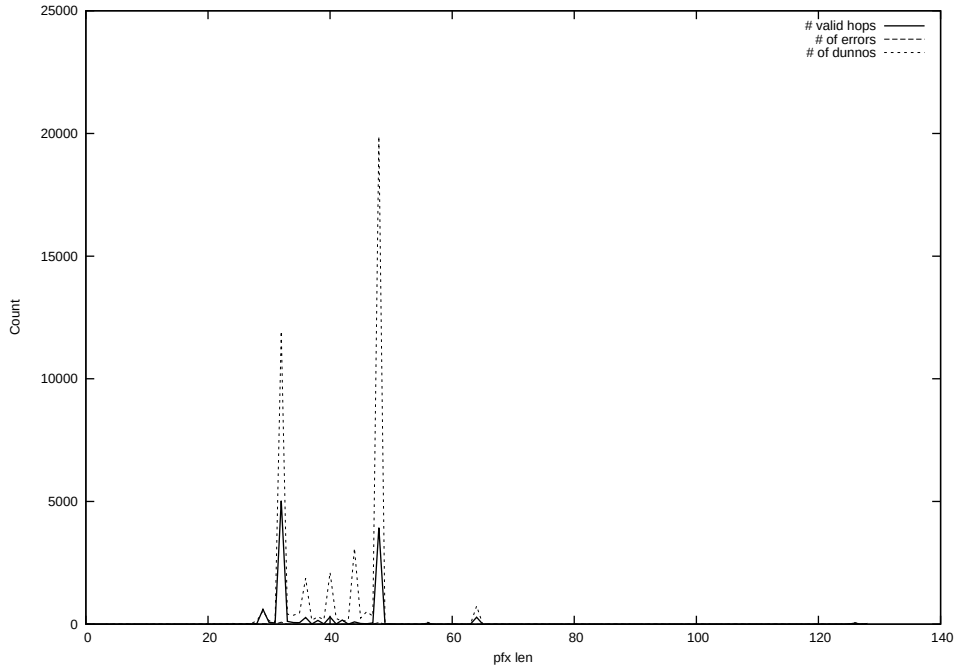
2014-06-04



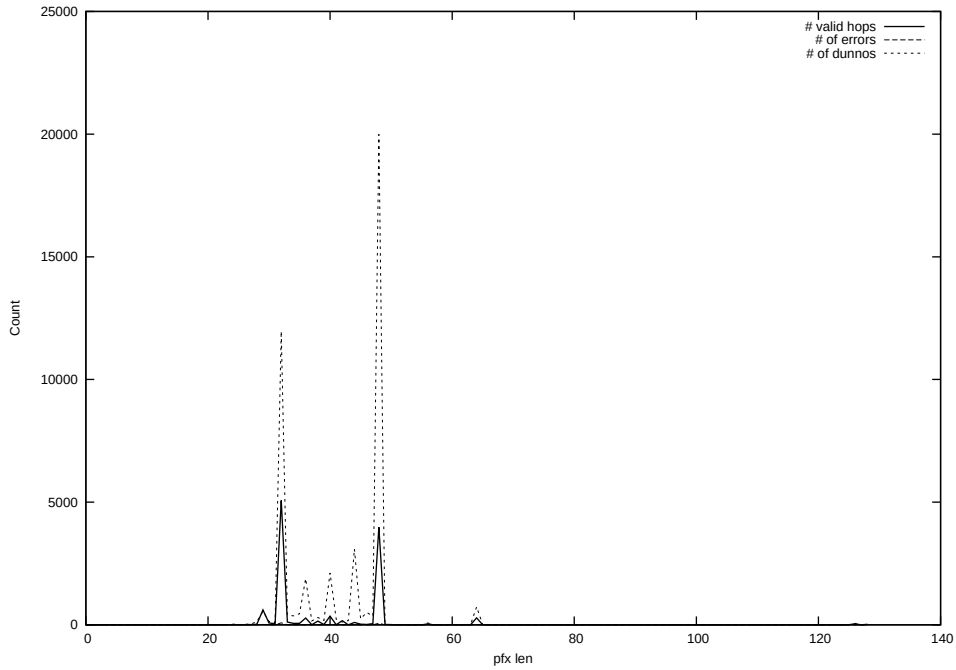
2014-06-05



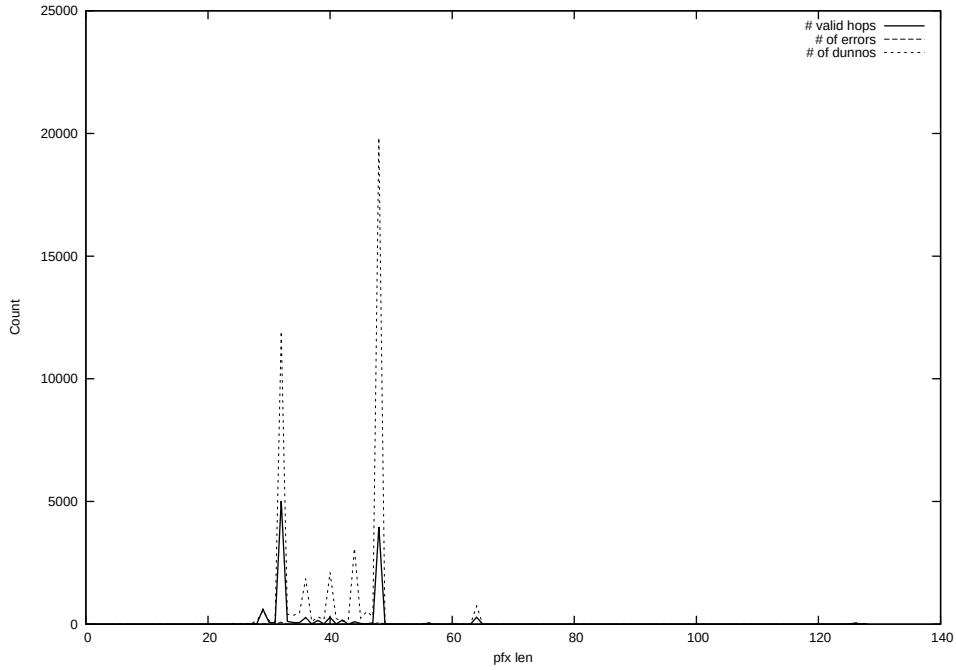
2014-06-06



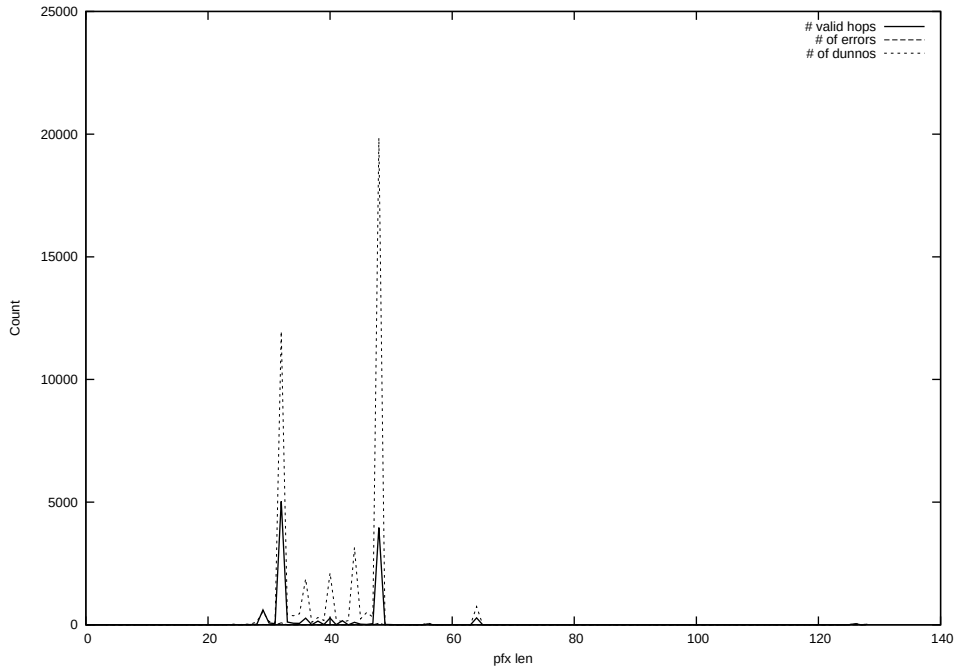
2014-06-07



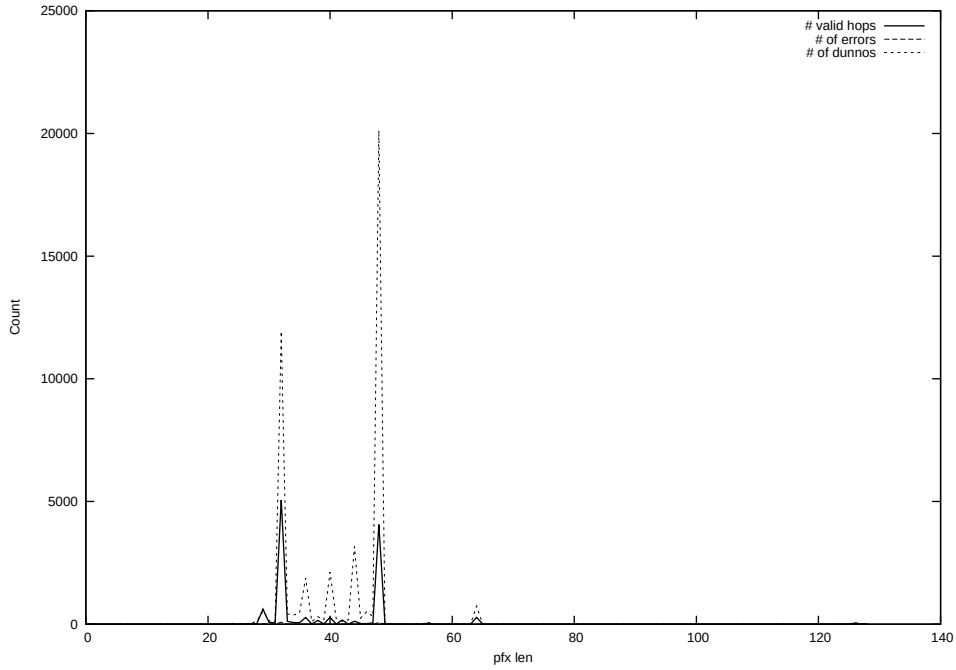
2014-06-08



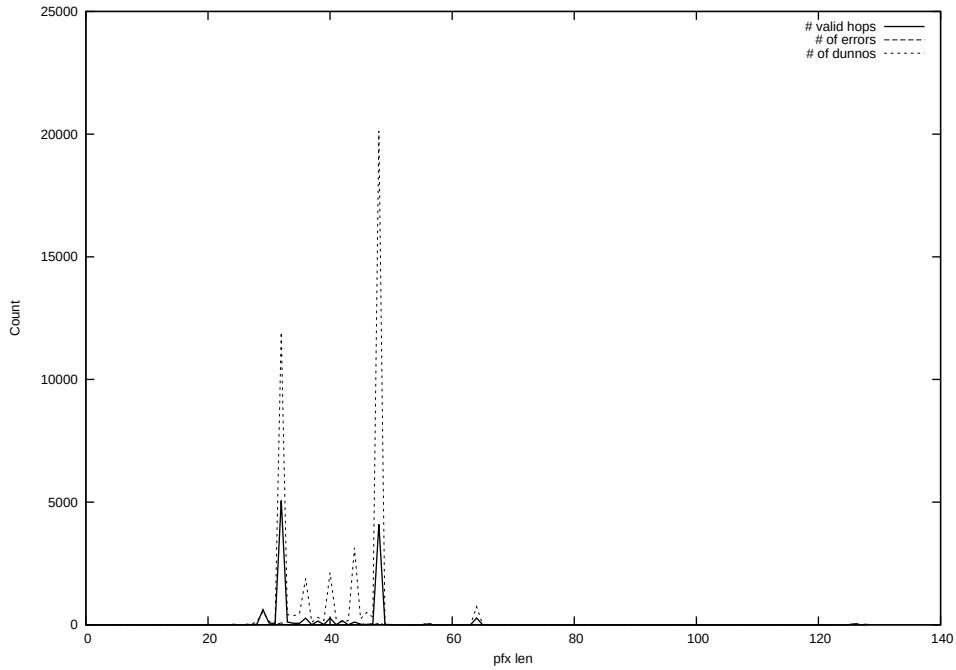
2014-06-09



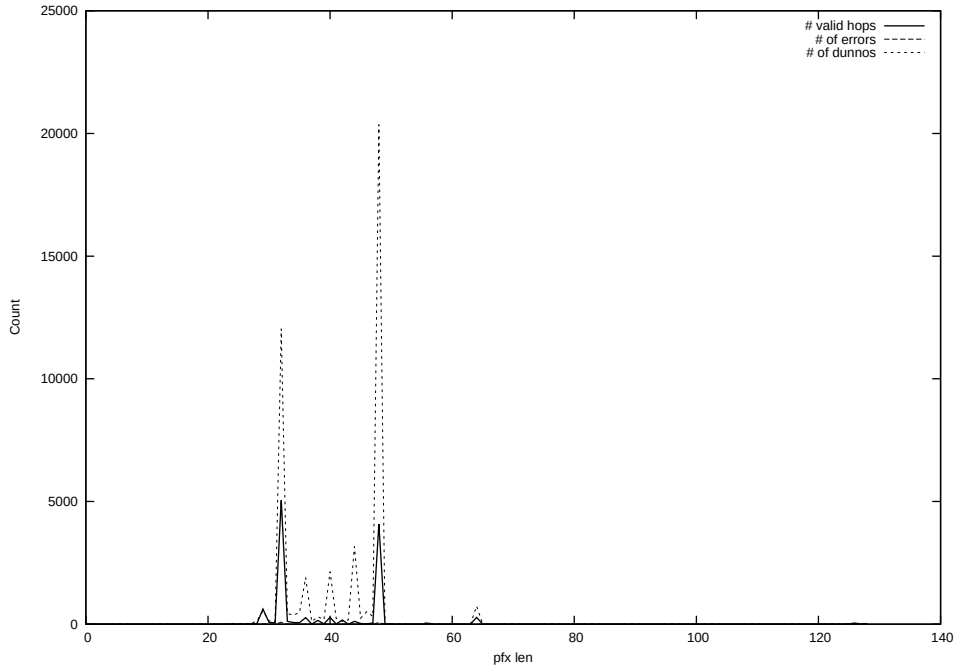
2014-06-10



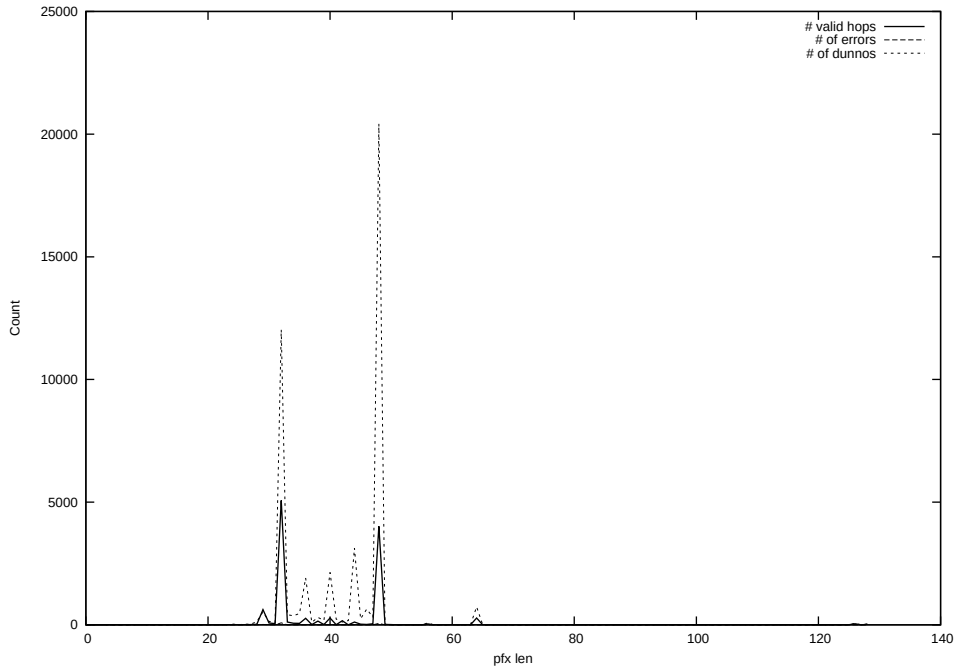
2014-06-11



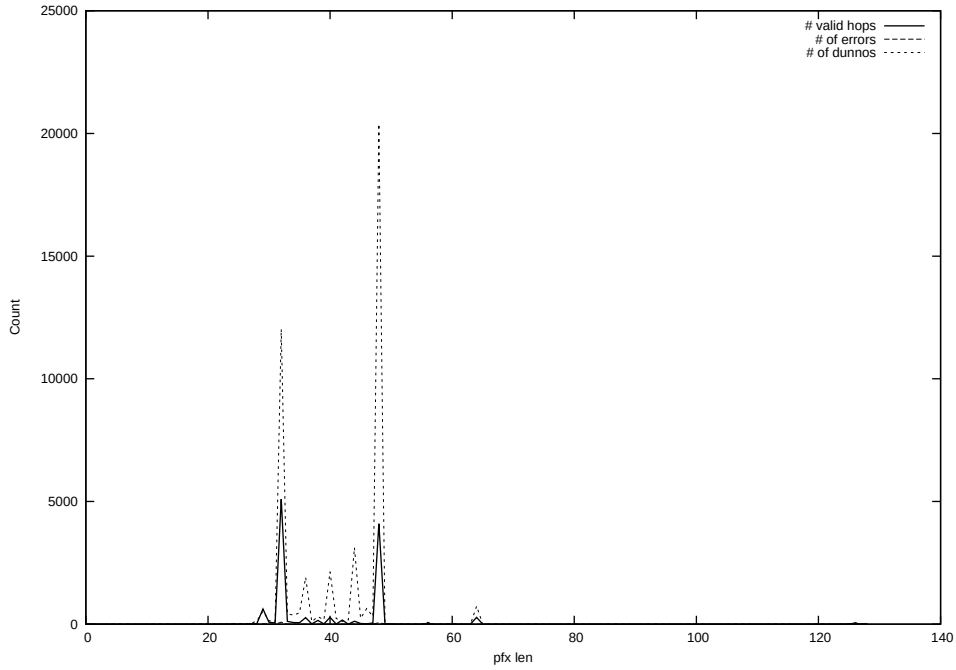
2014-06-12



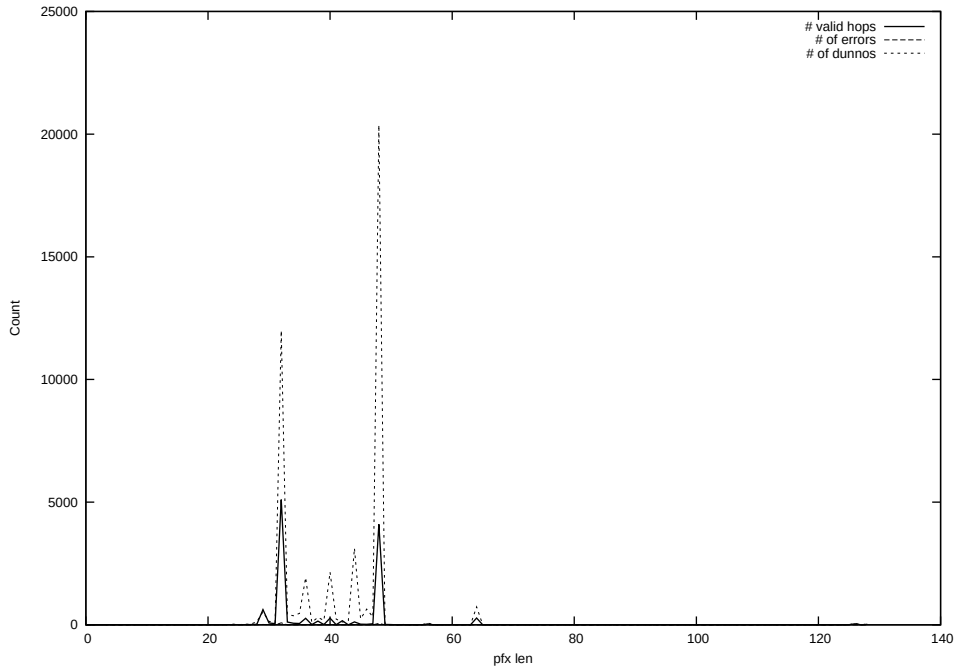
2014-06-13



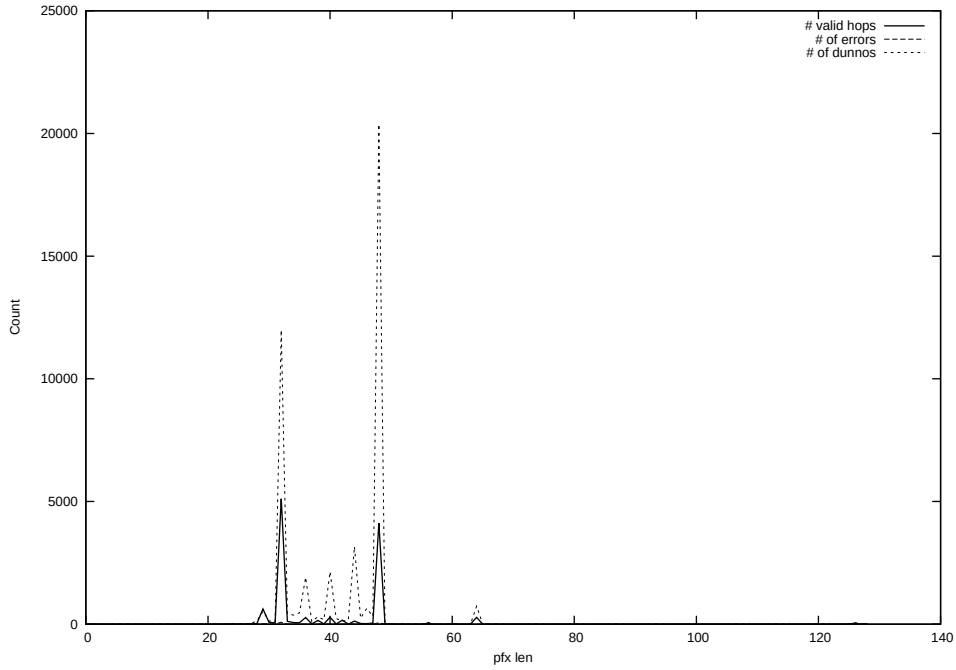
2014-06-14



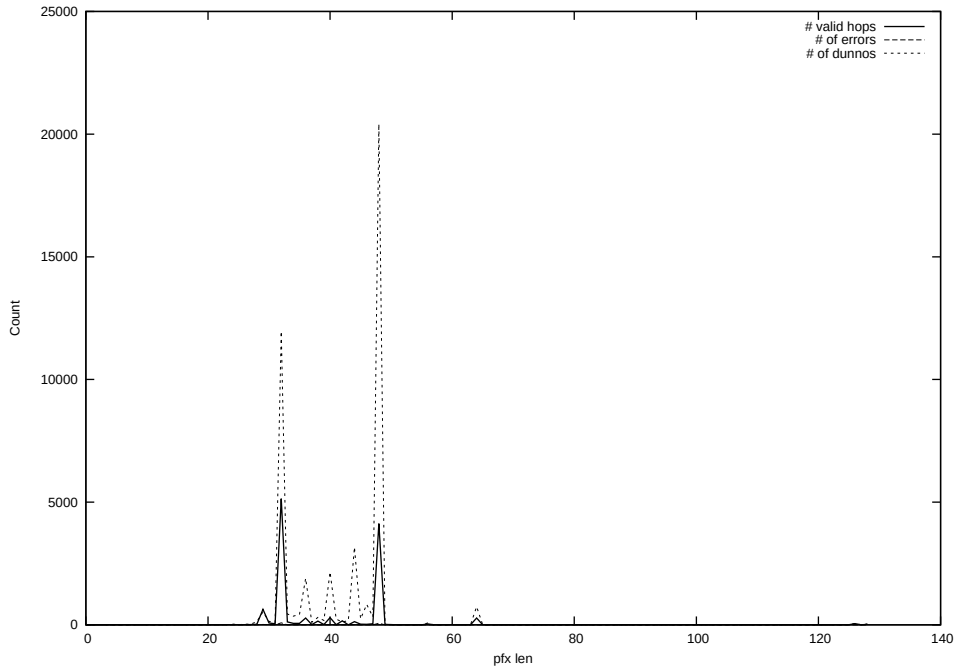
2014-06-15



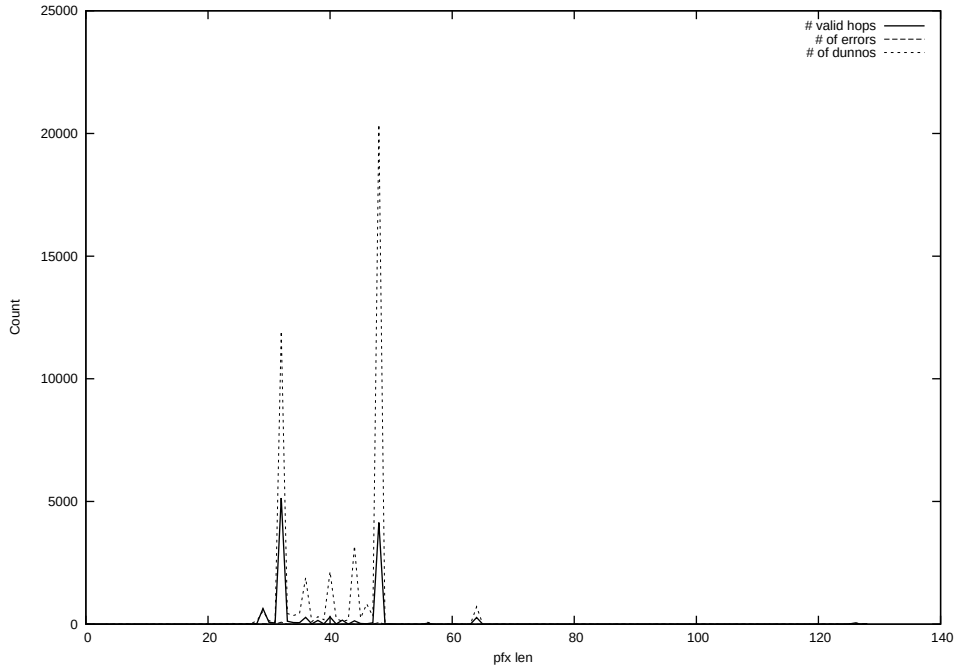
2014-06-16



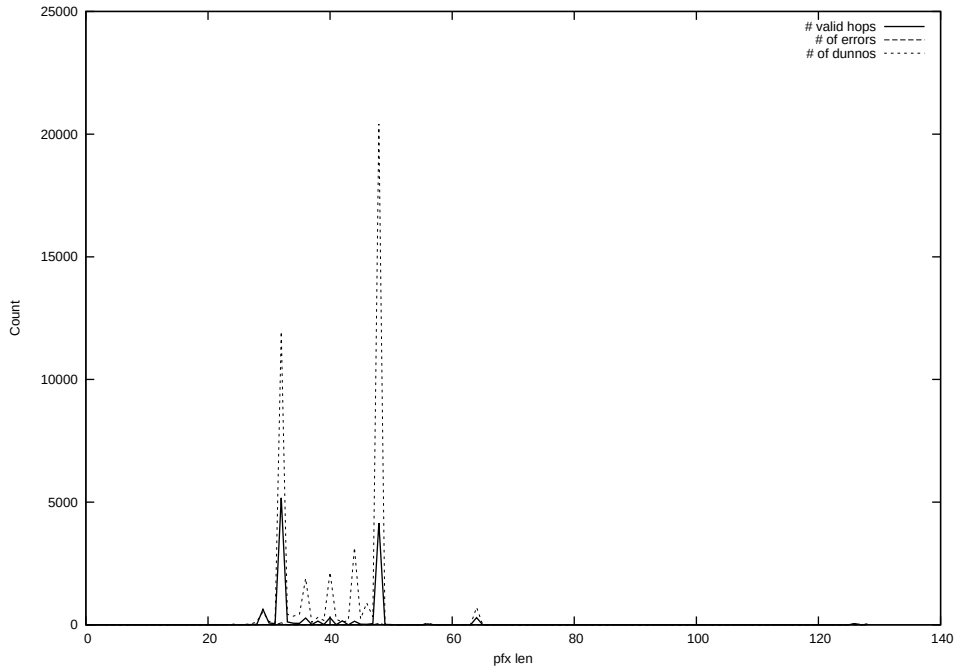
2014-06-17



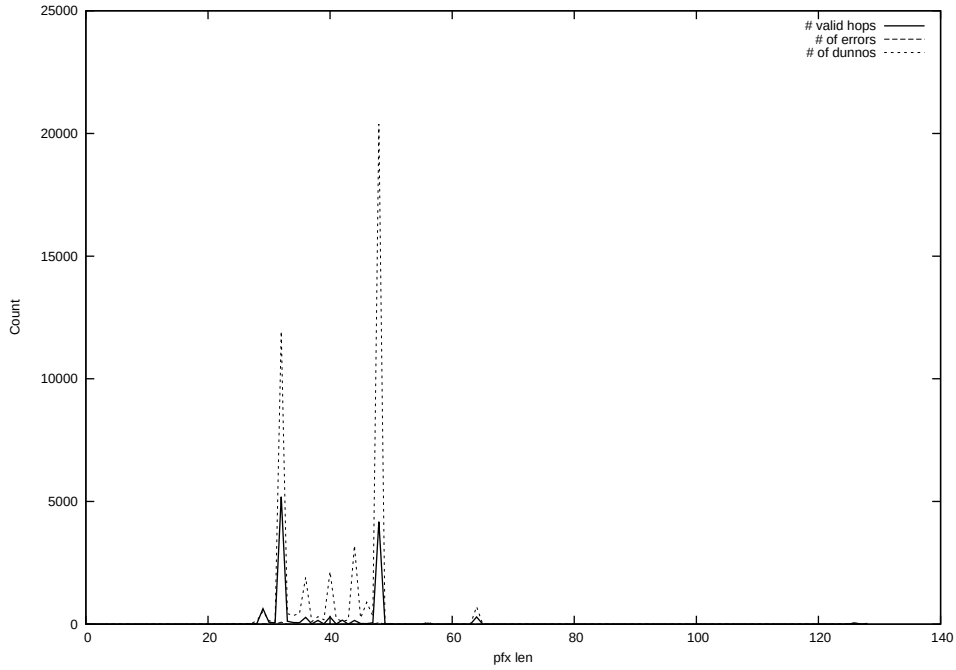
2014-06-18



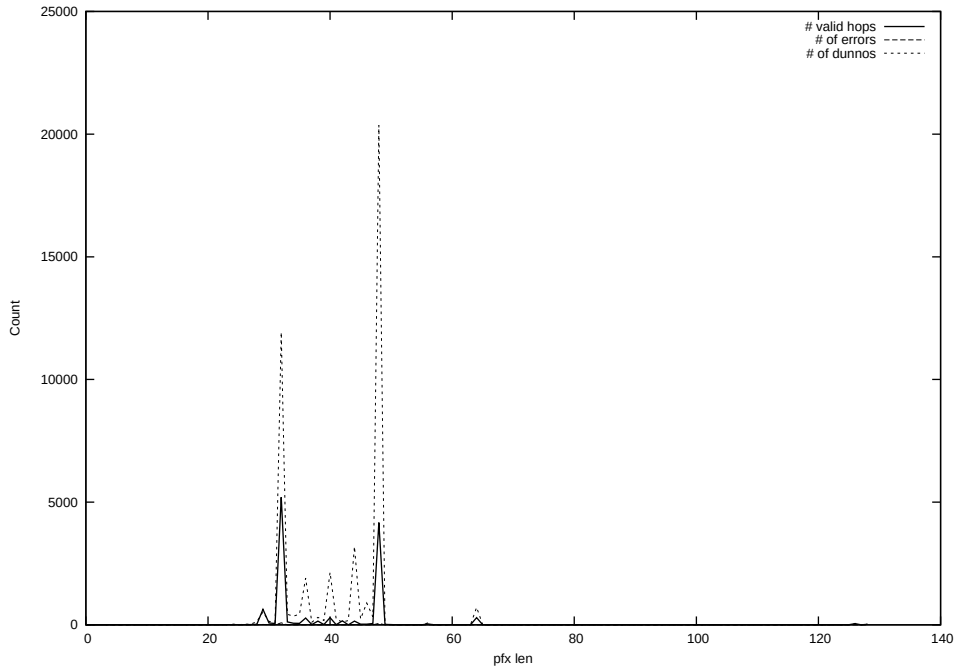
2014-06-19



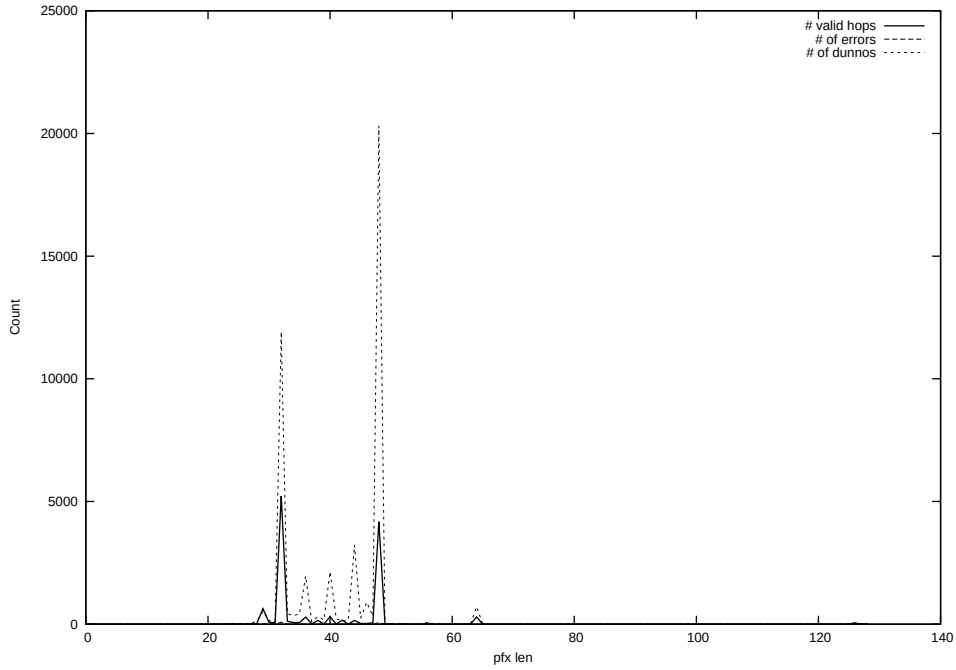
2014-06-20



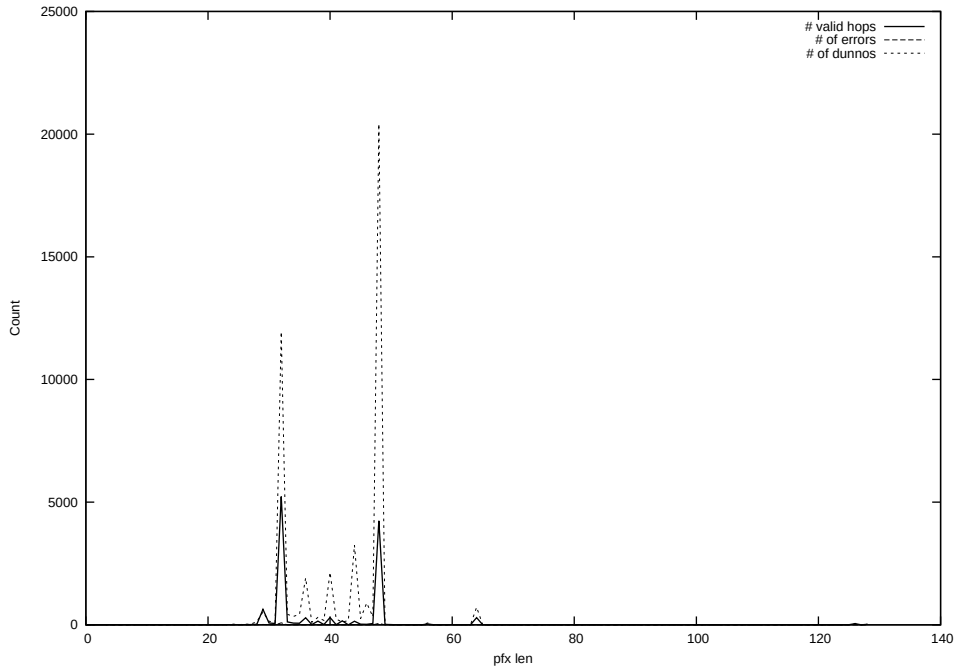
2014-06-21



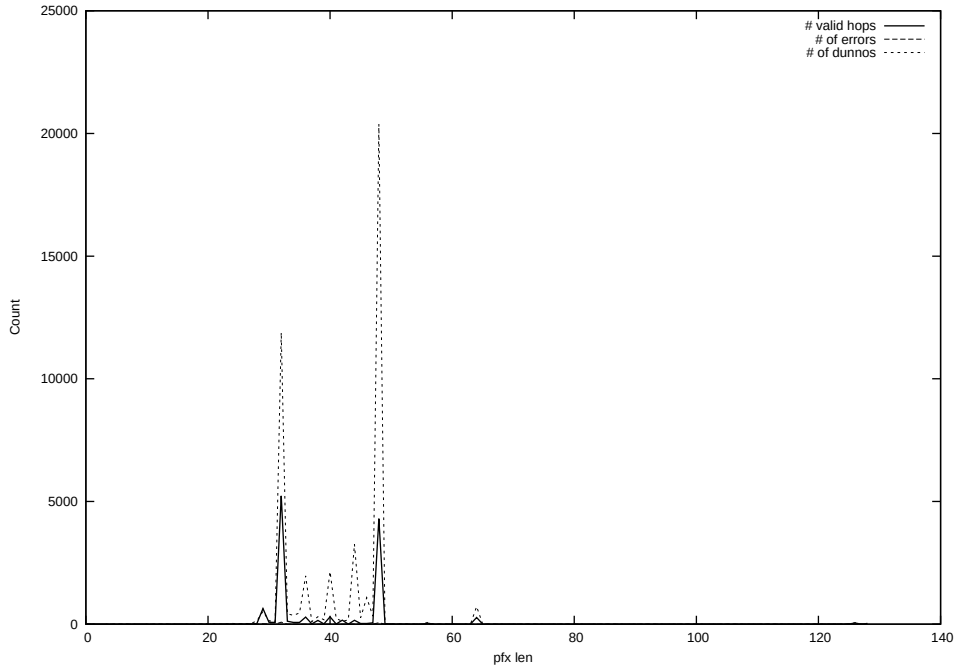
2014-06-22



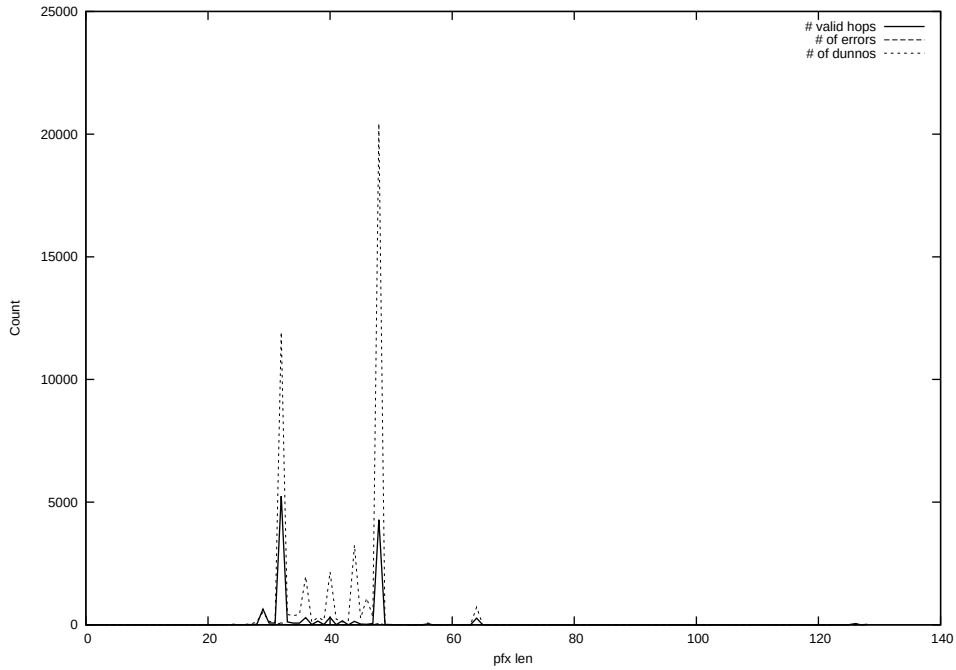
2014-06-23



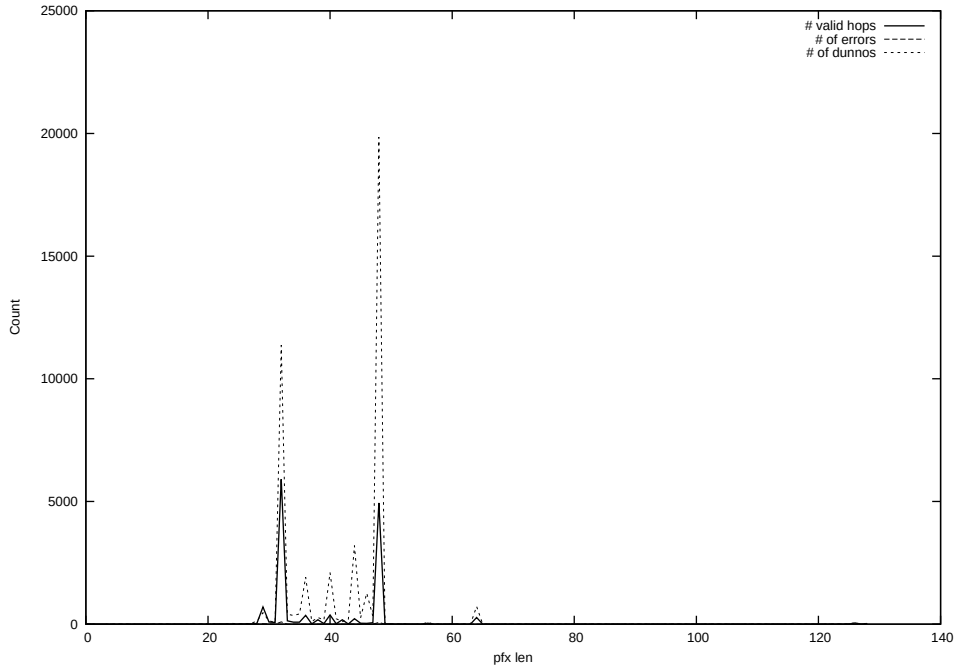
2014-06-24



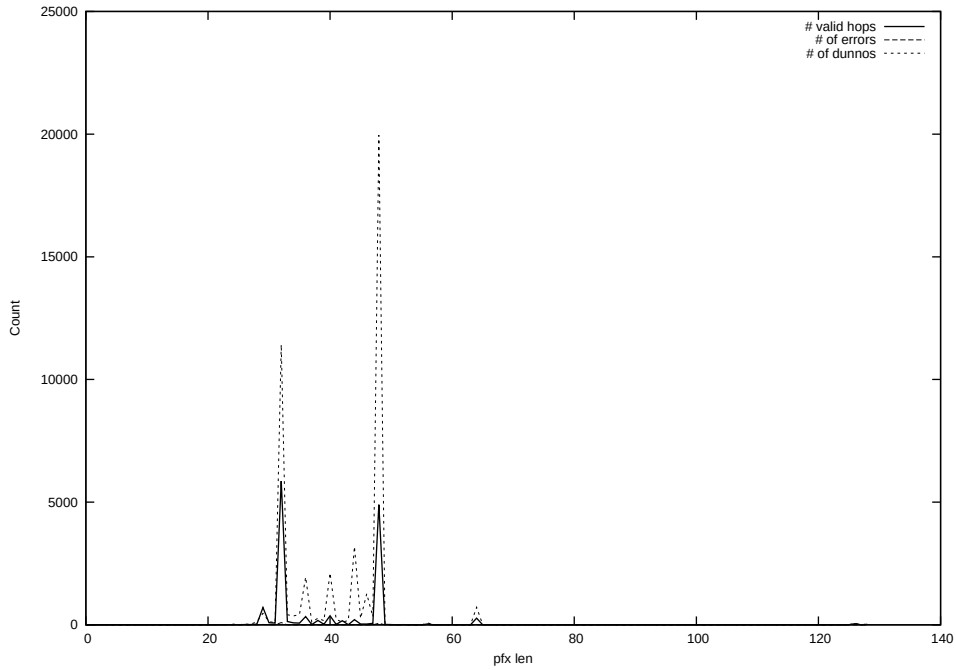
2014-06-25



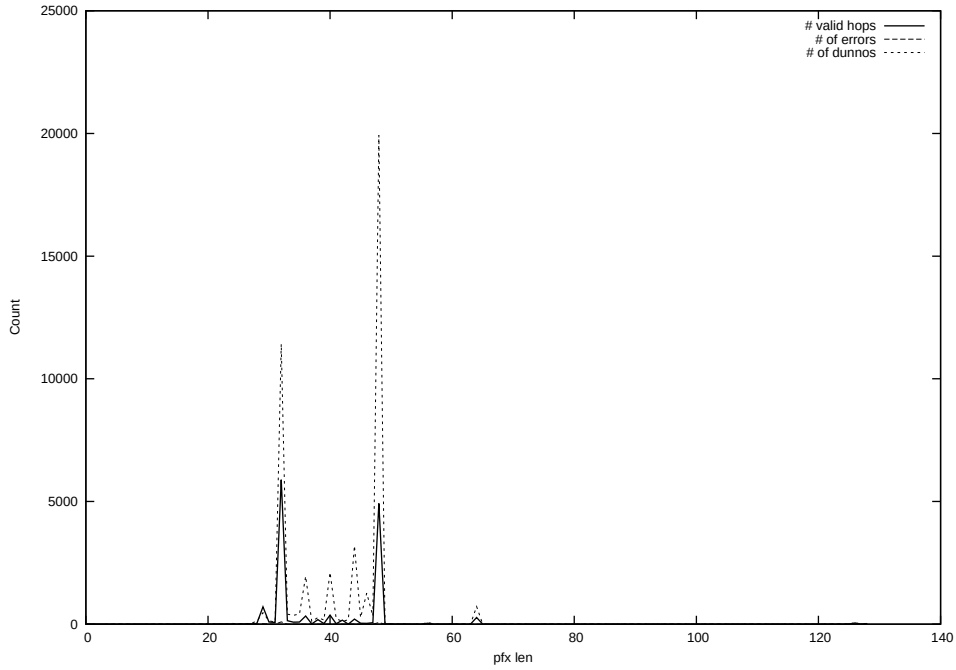
2014-06-26



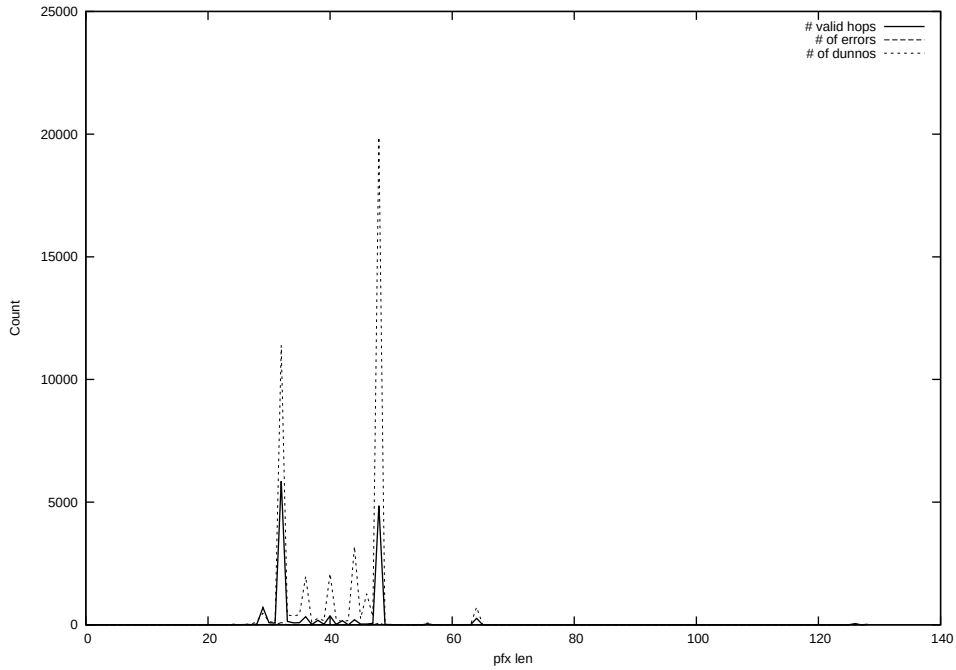
2014-06-27



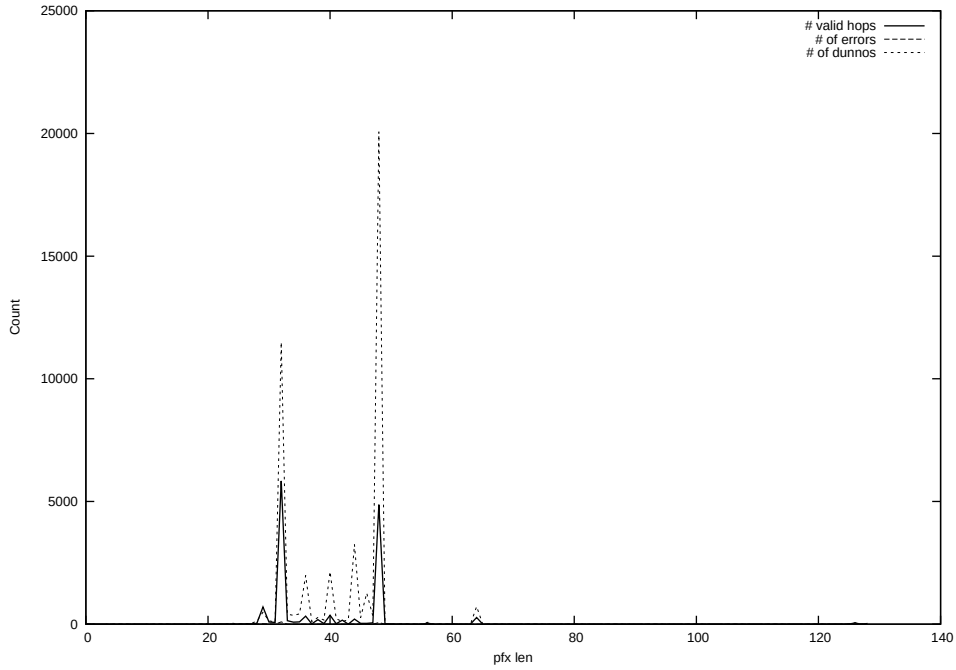
2014-06-28



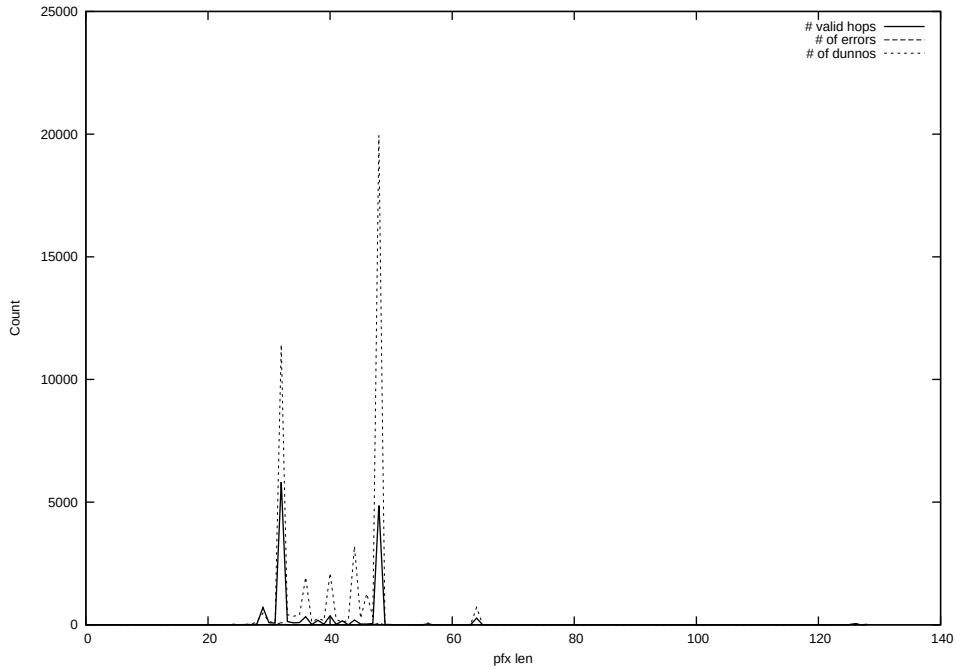
2014-06-29



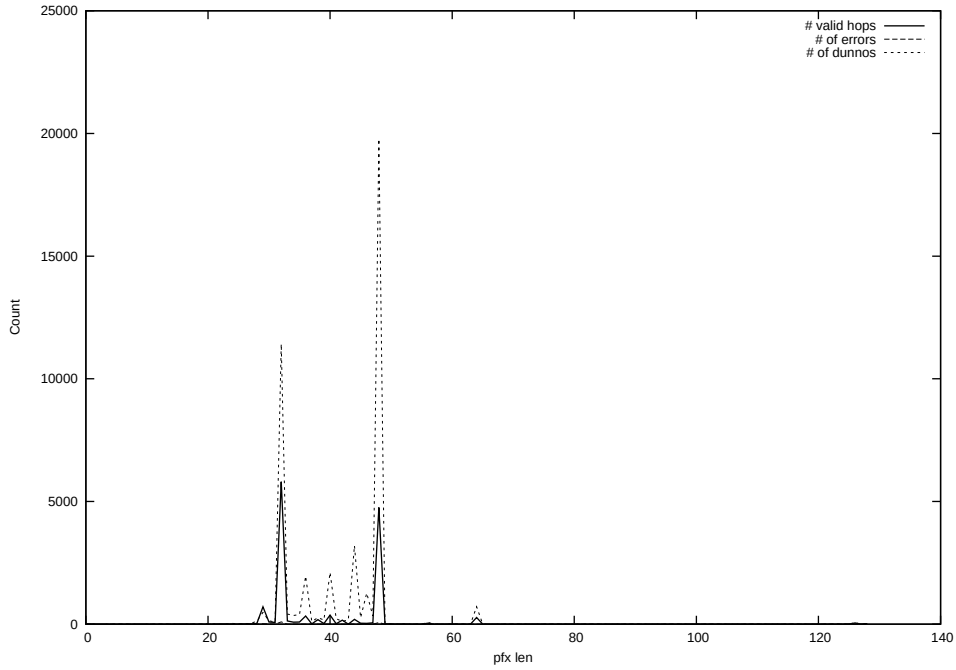
2014-06-30



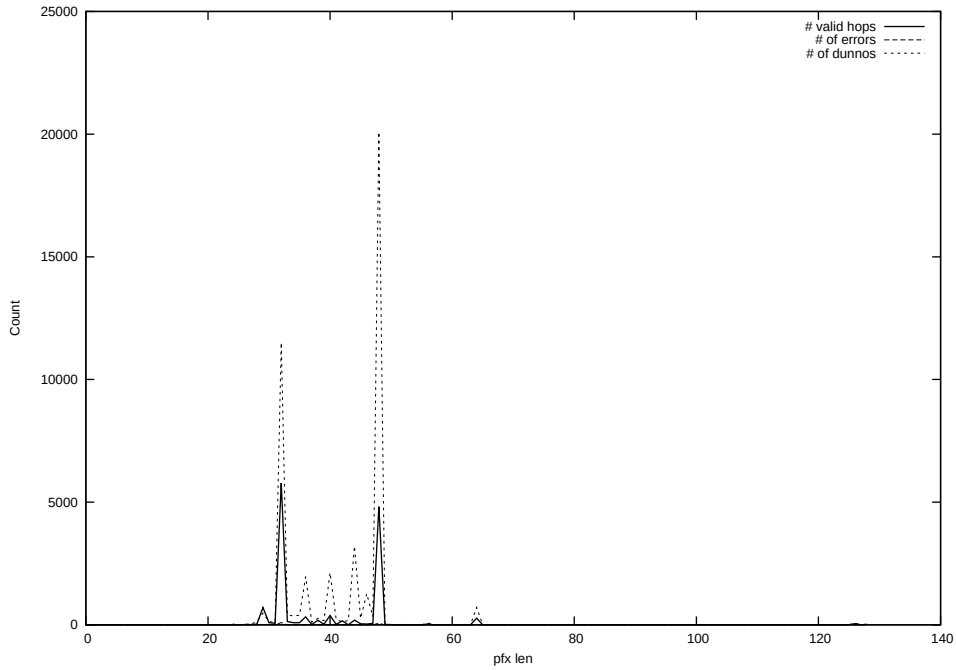
2014-07-01



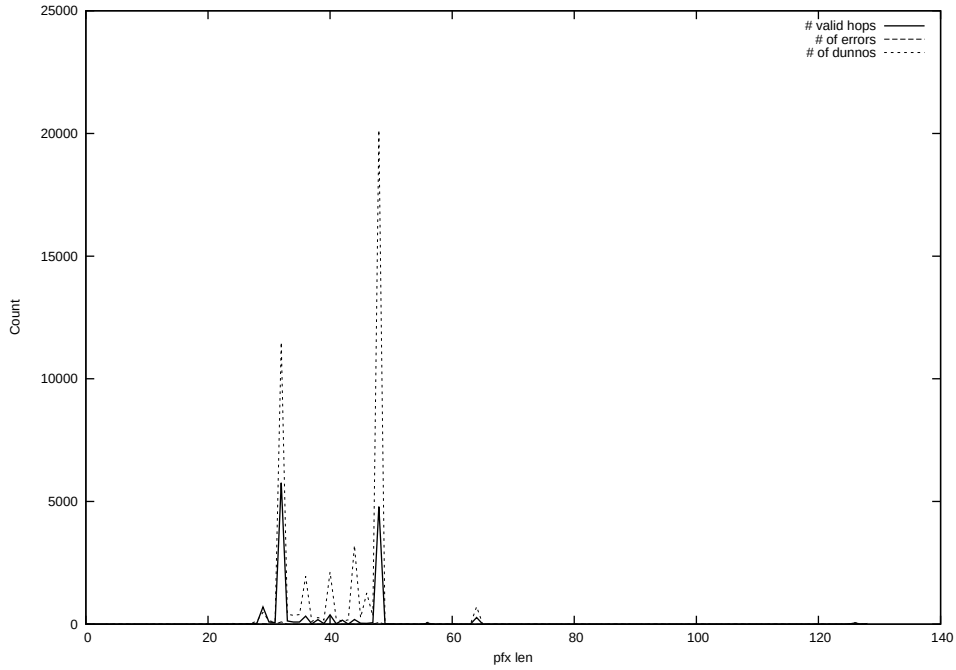
2014-07-02



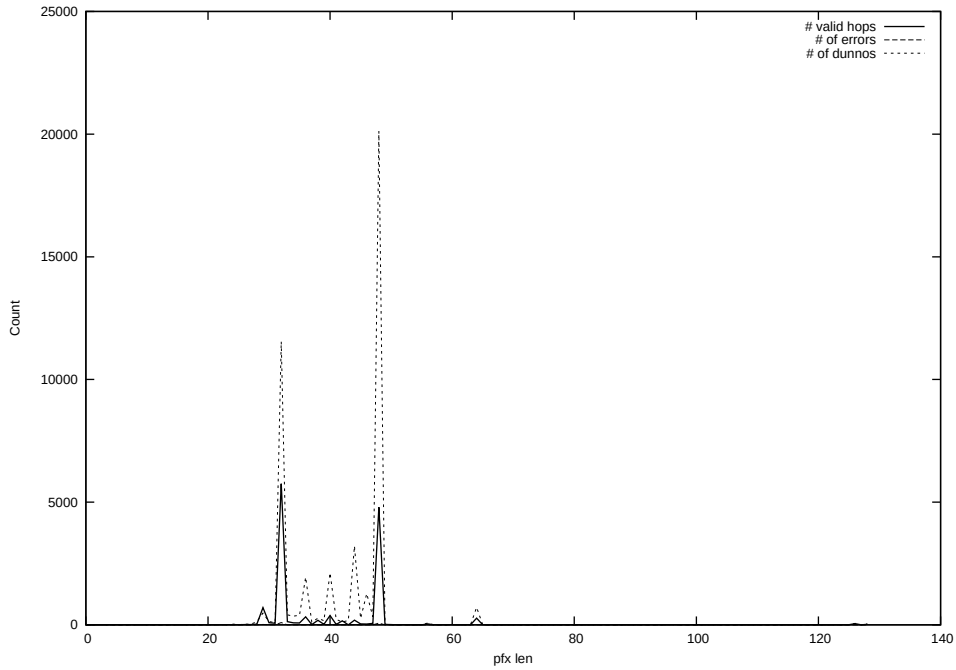
2014-07-03



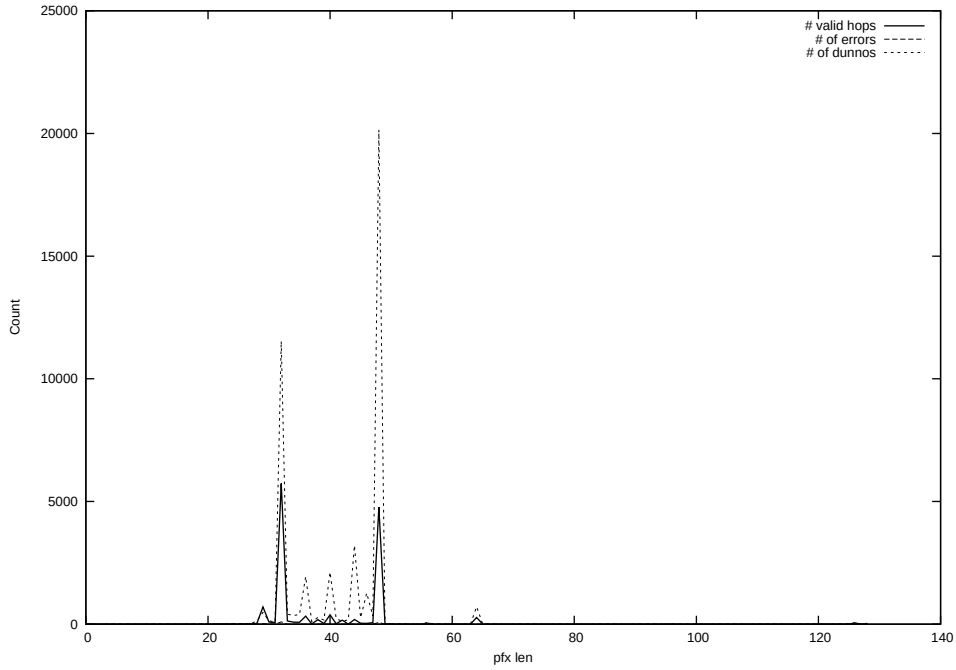
2014-07-04



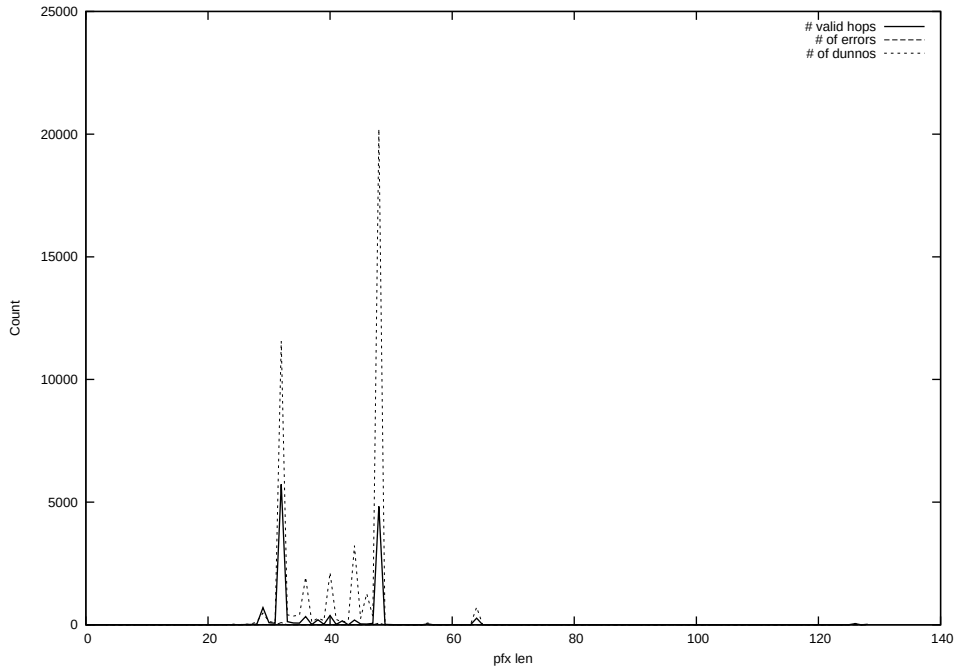
2014-07-05



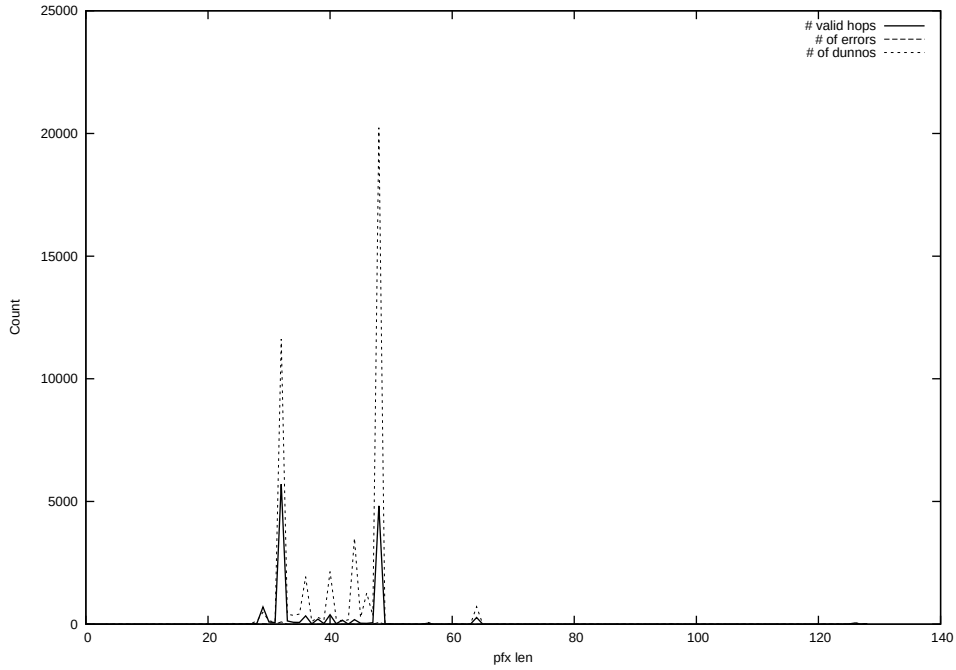
2014-07-06



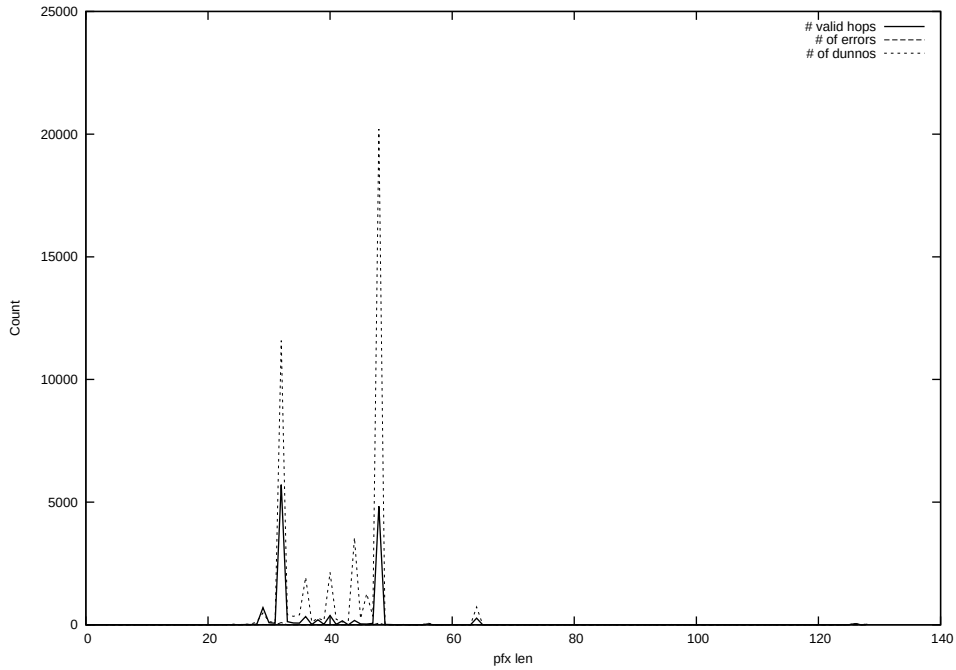
2014-07-07



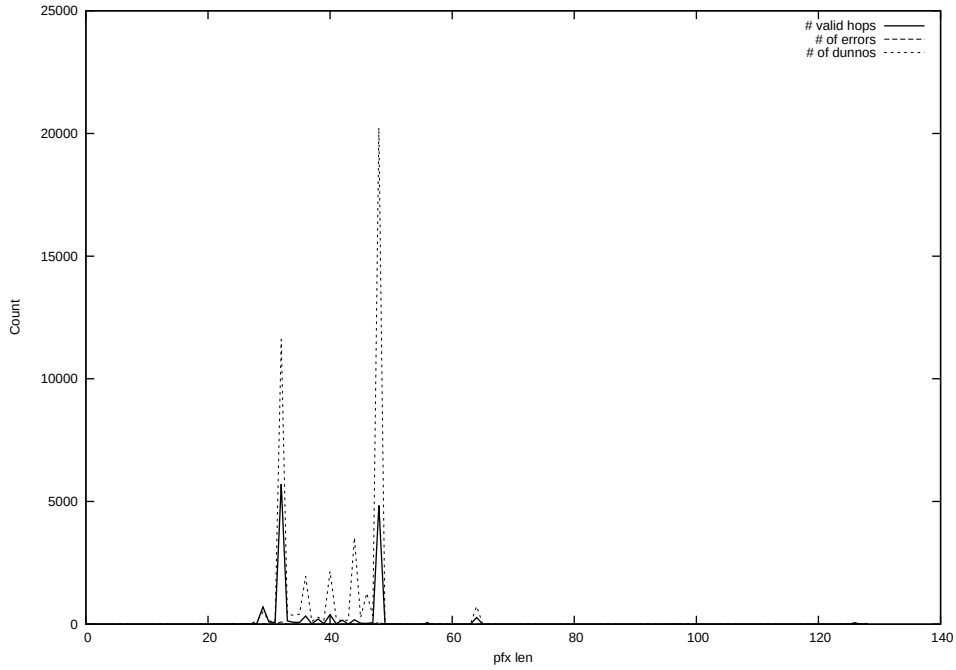
2014-07-08



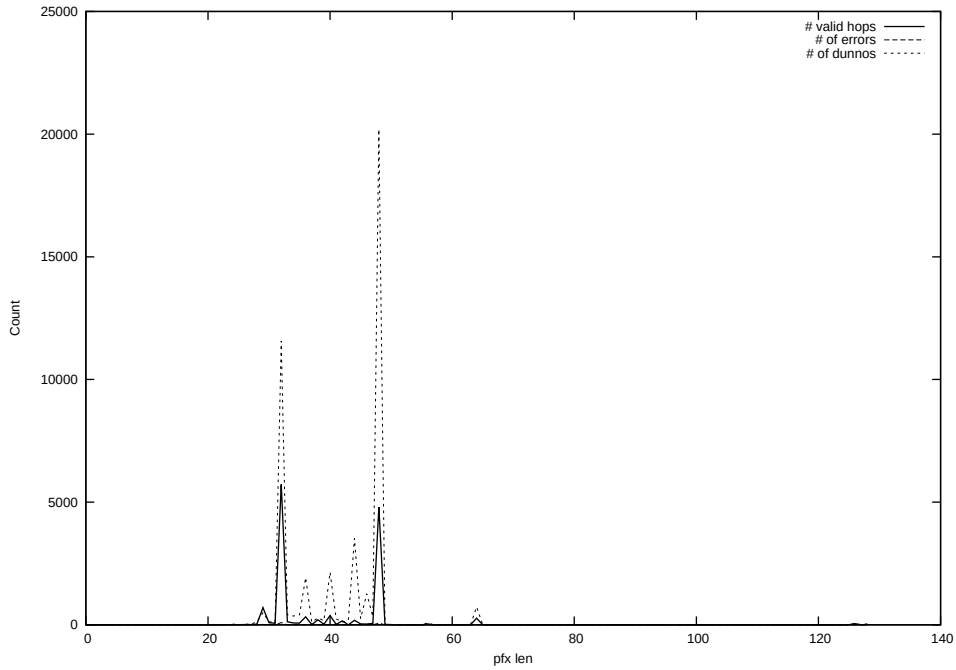
2014-07-09



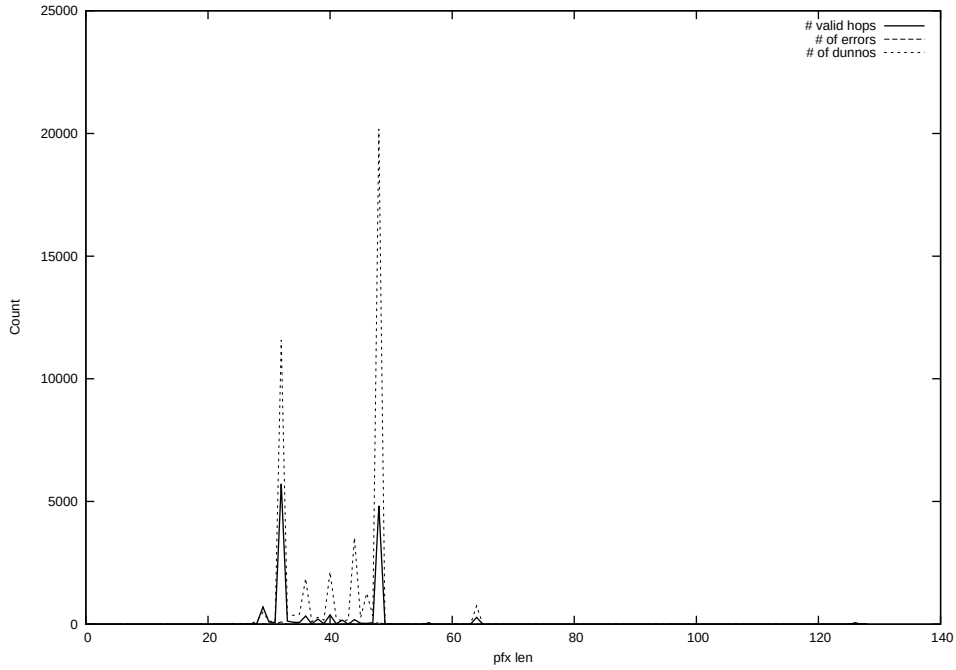
2014-07-10



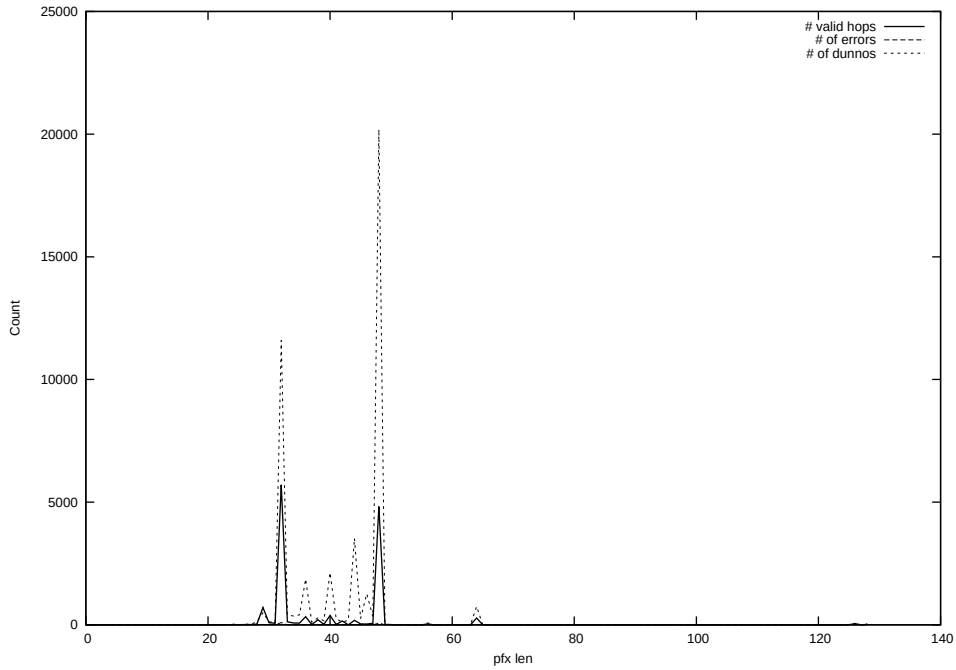
2014-07-11



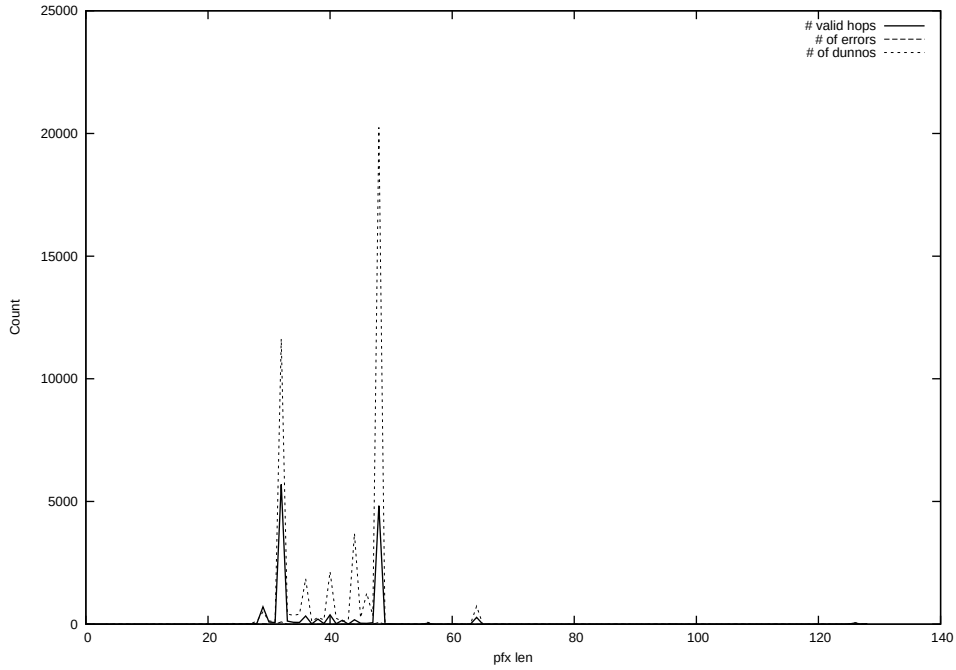
2014-07-12



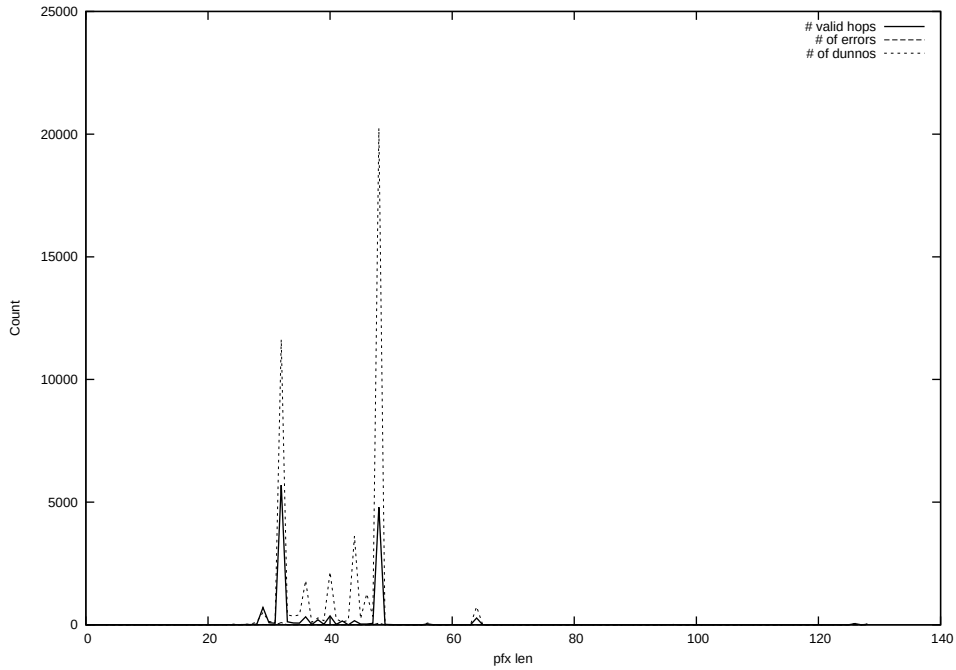
2014-07-13



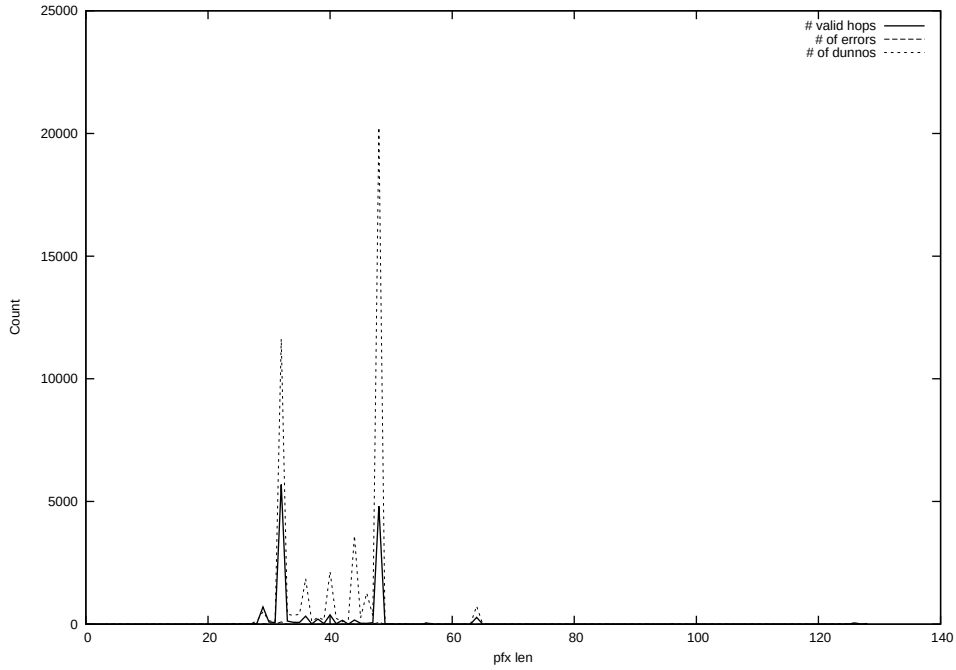
2014-07-14



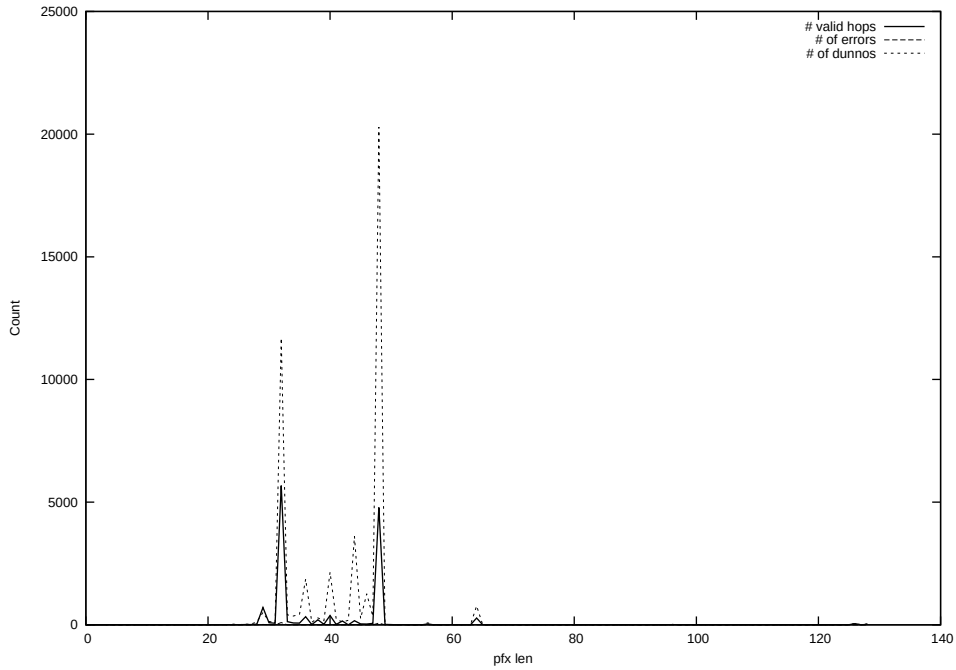
2014-07-15



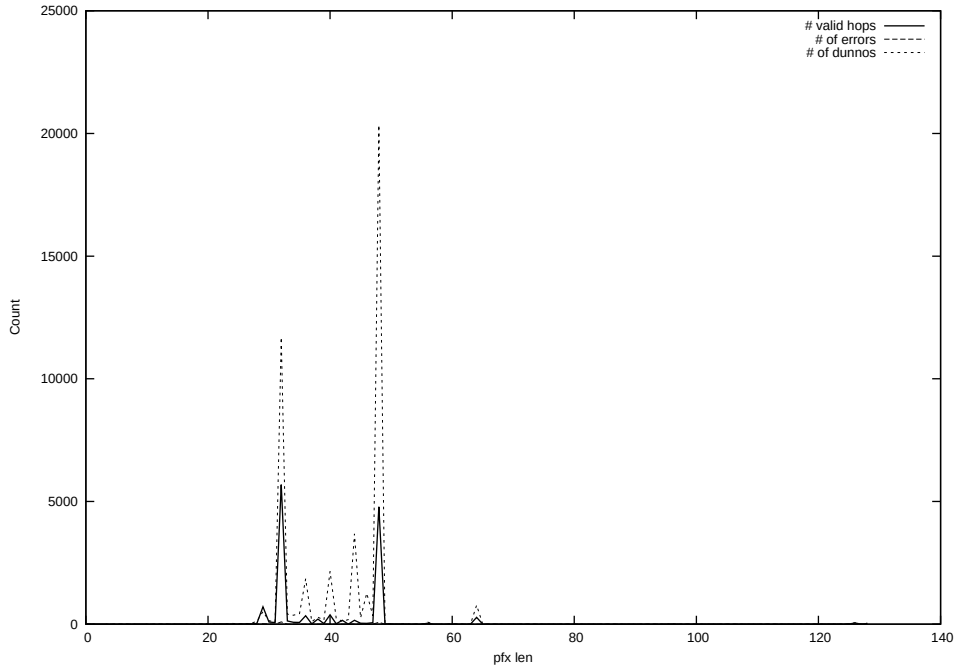
2014-07-16



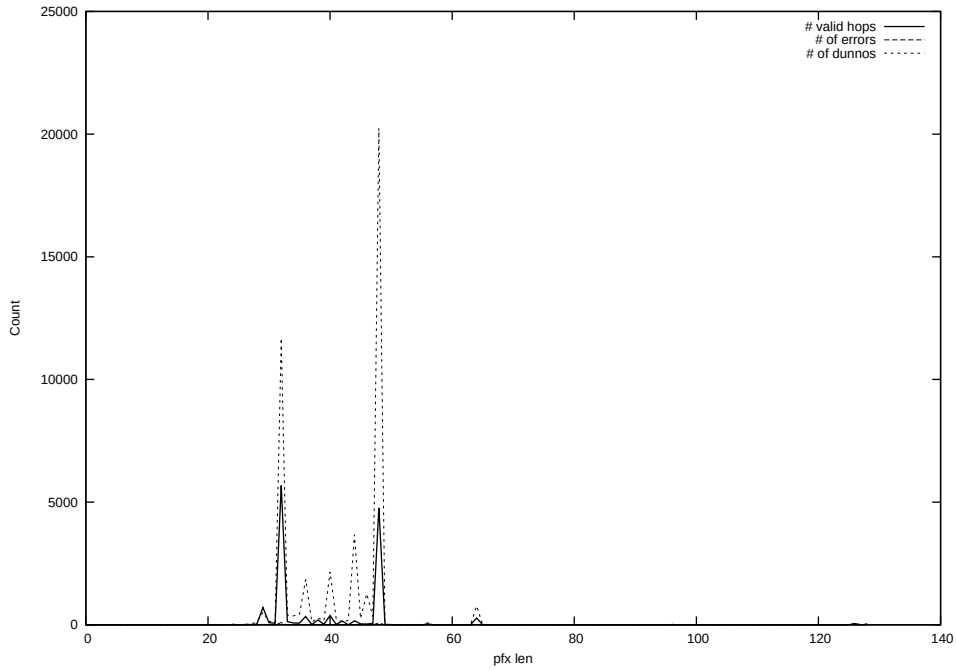
2014-07-17



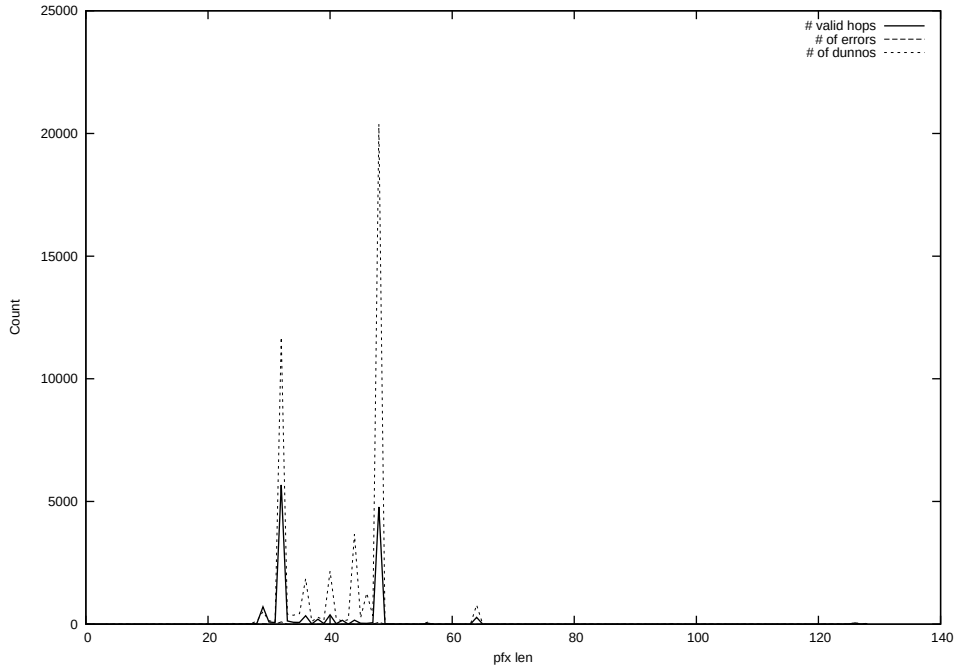
2014-07-18



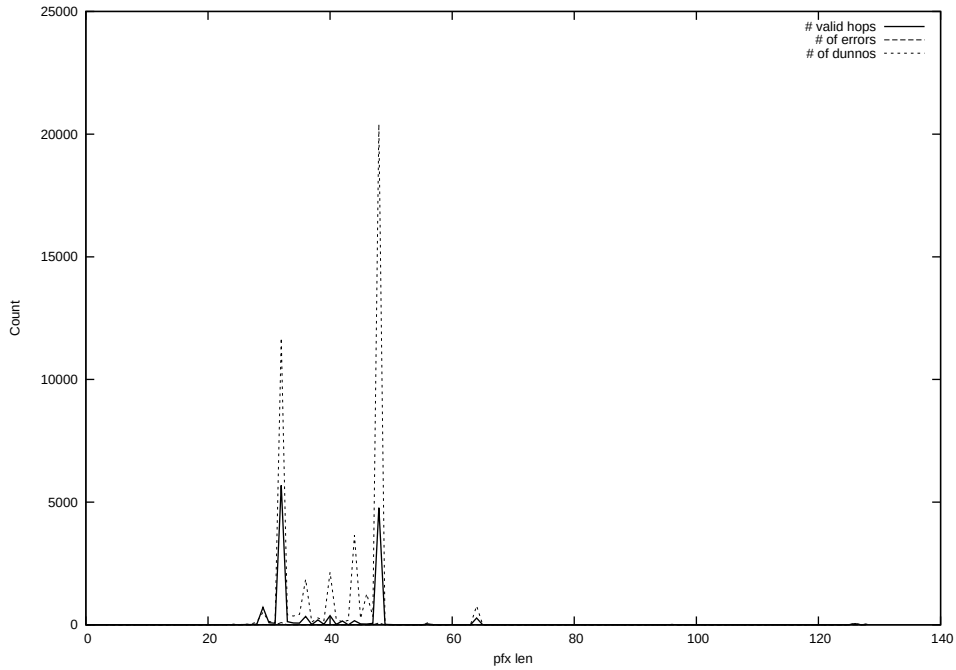
2014-07-19



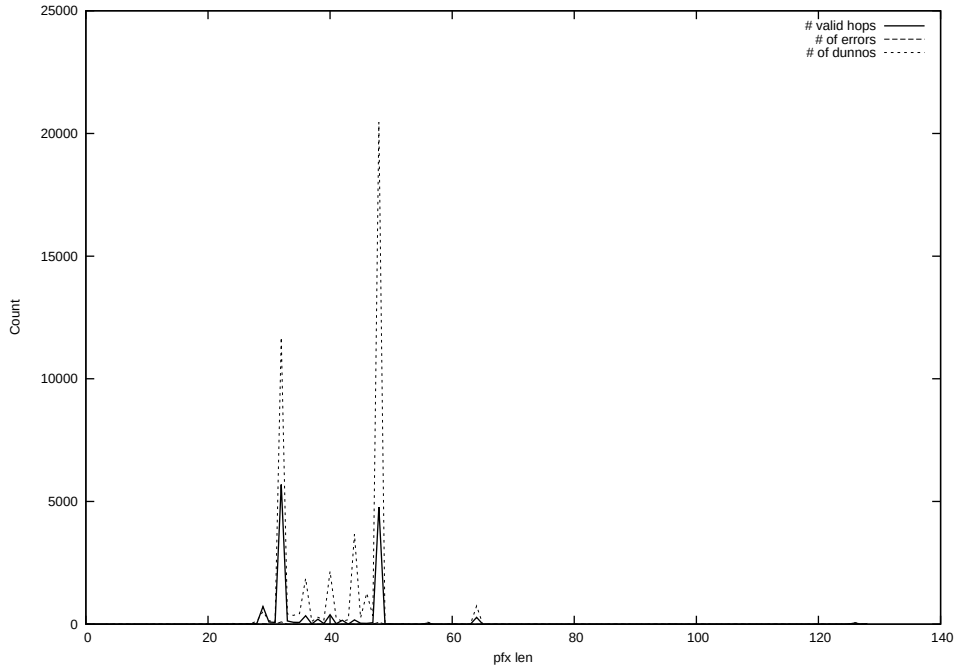
2014-07-20



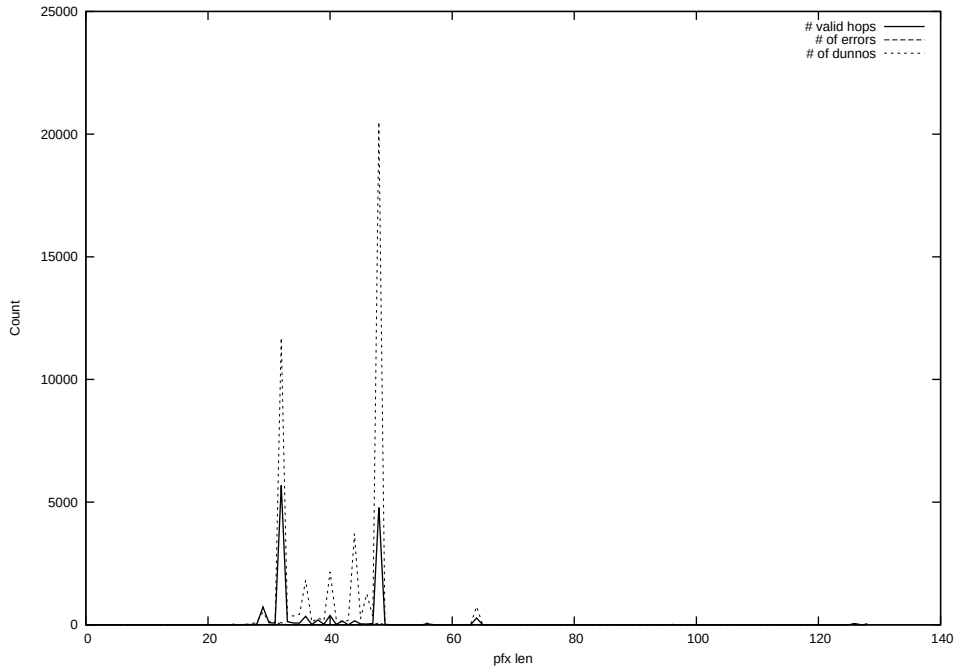
2014-07-21



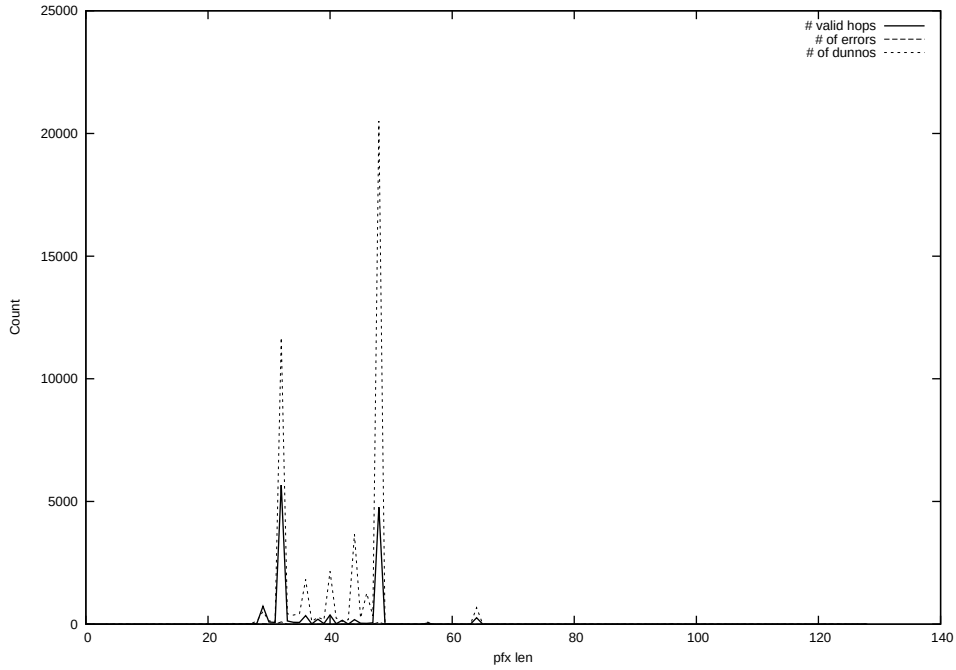
2014-07-22



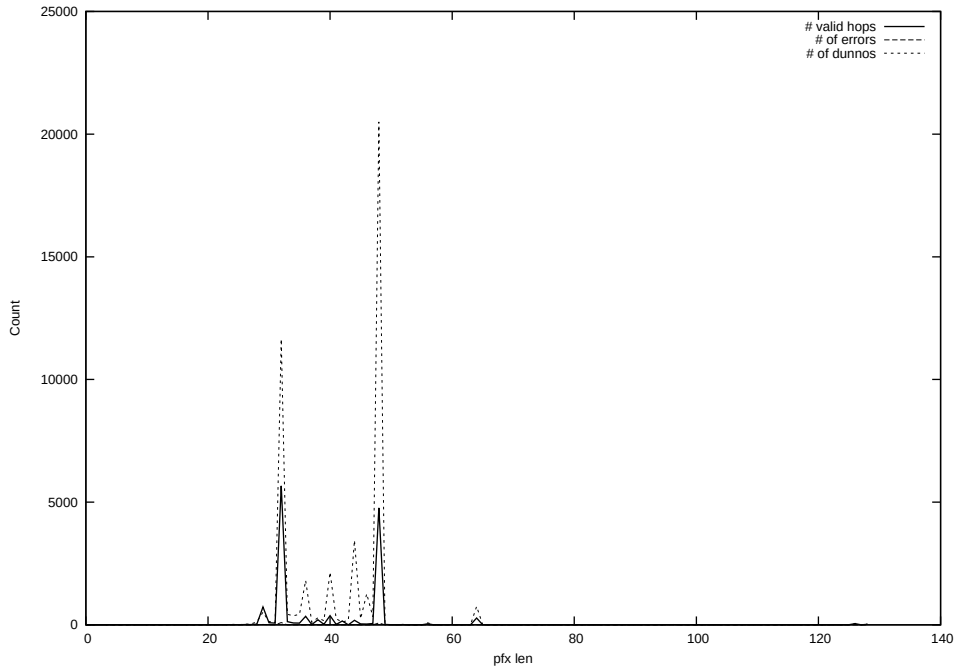
2014-07-23



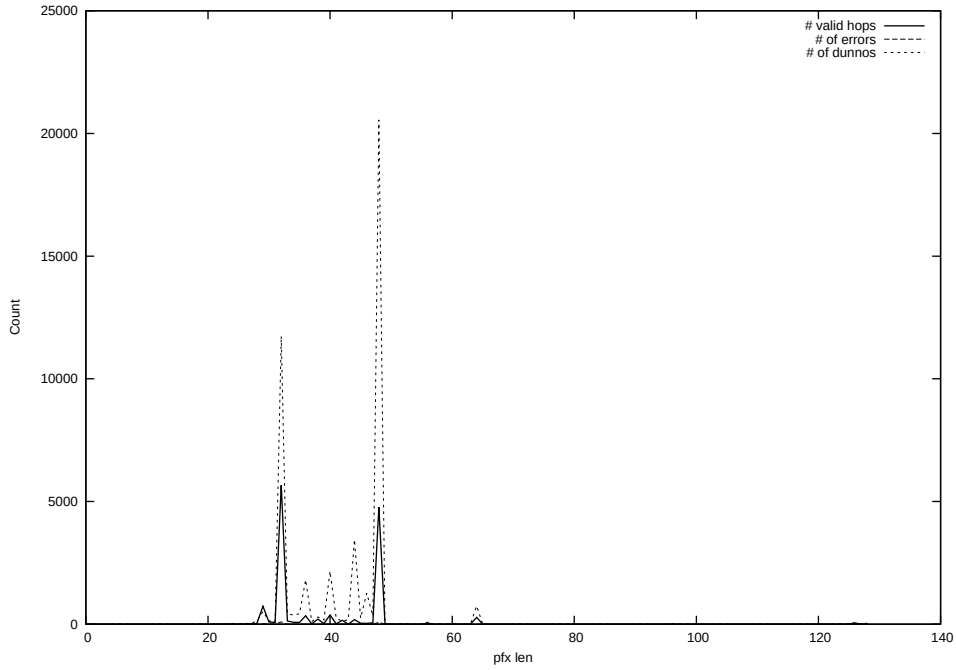
2014-07-24



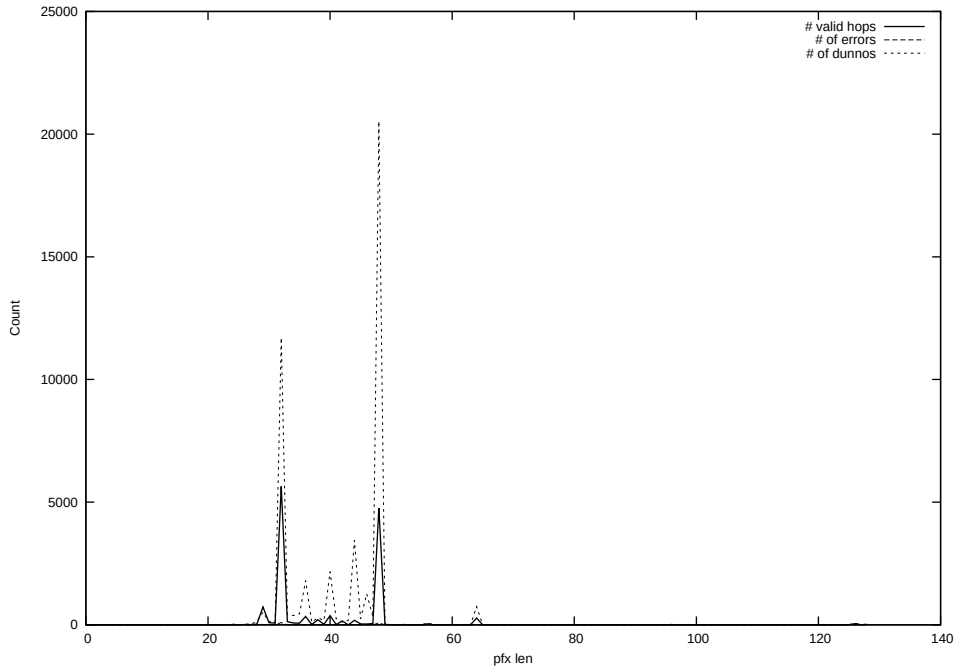
2014-07-25



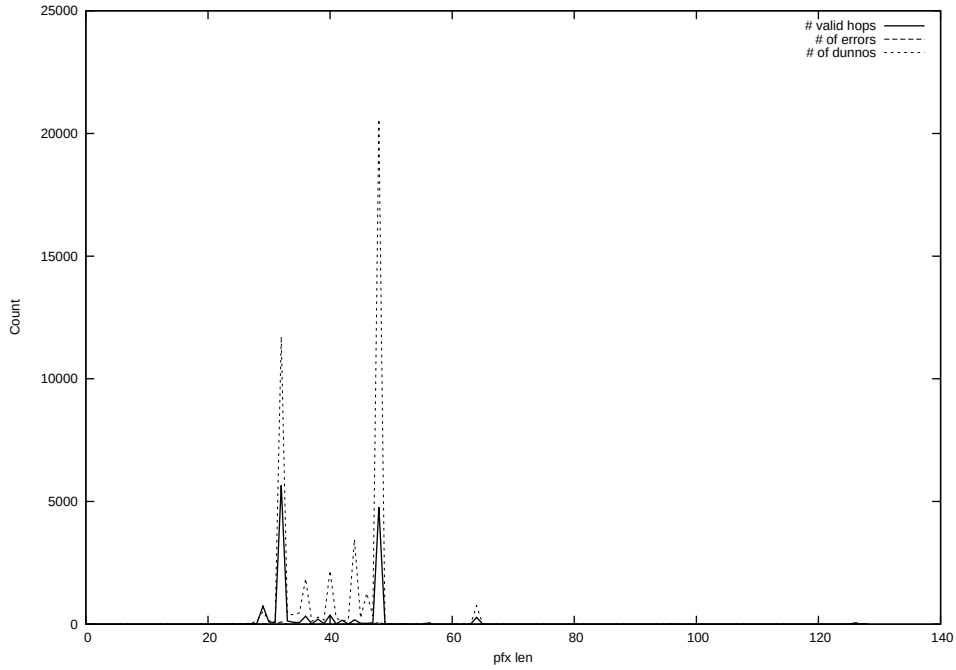
2014-07-26



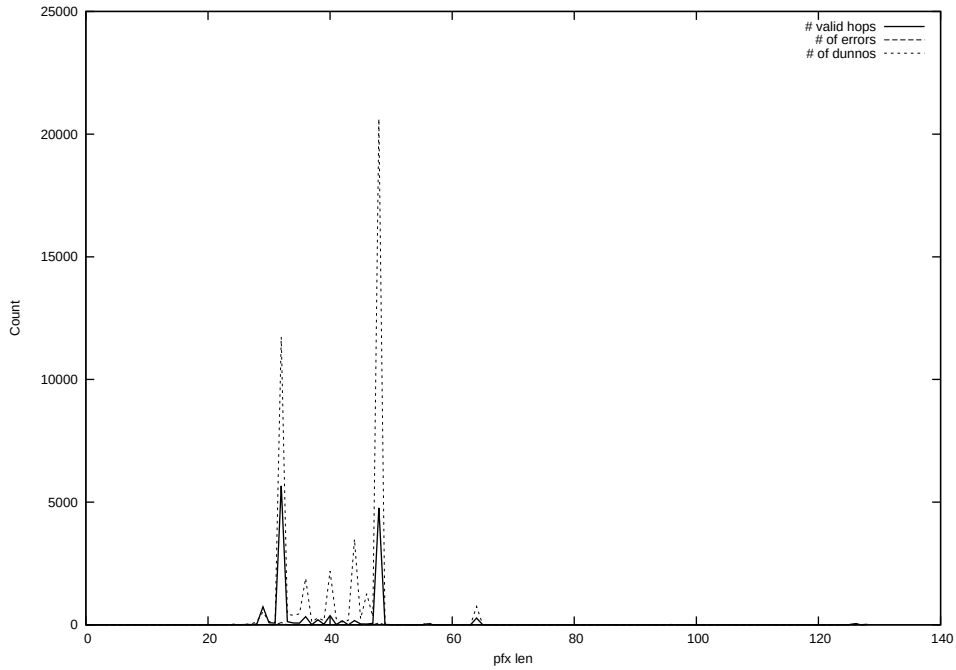
2014-07-27



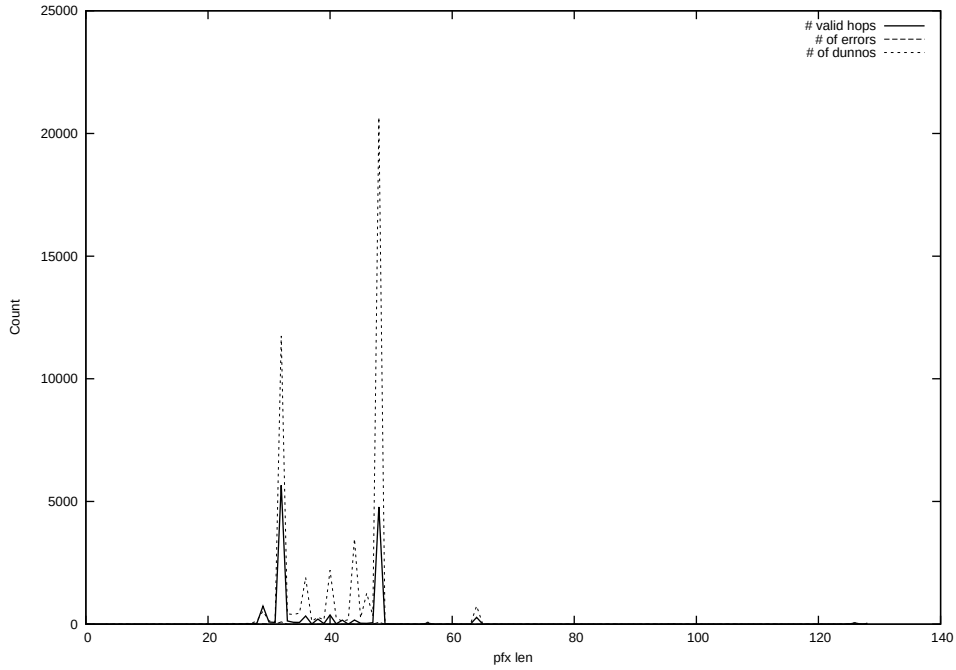
2014-07-28



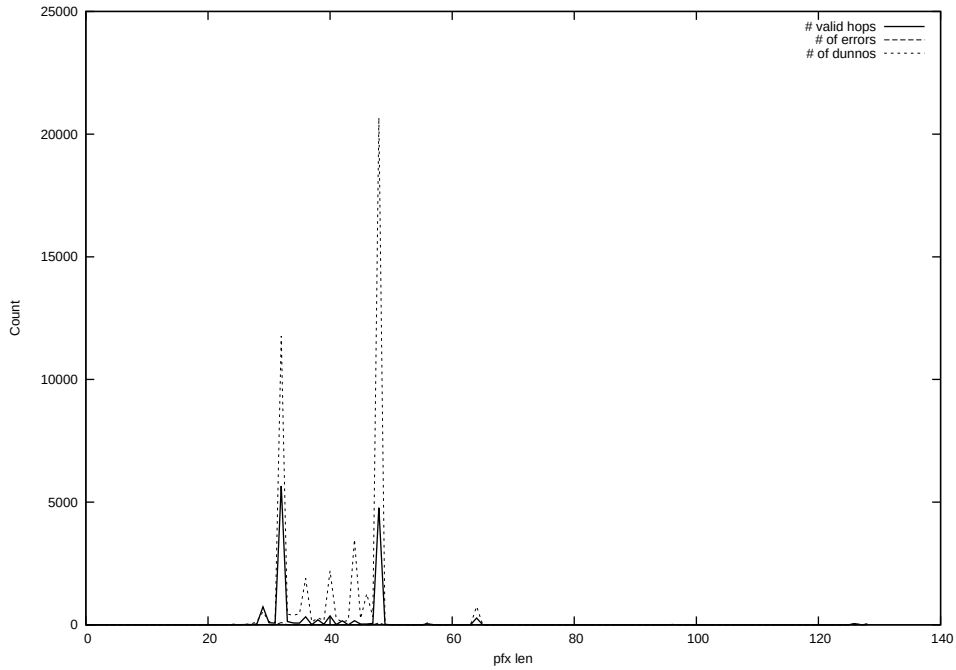
2014-07-29



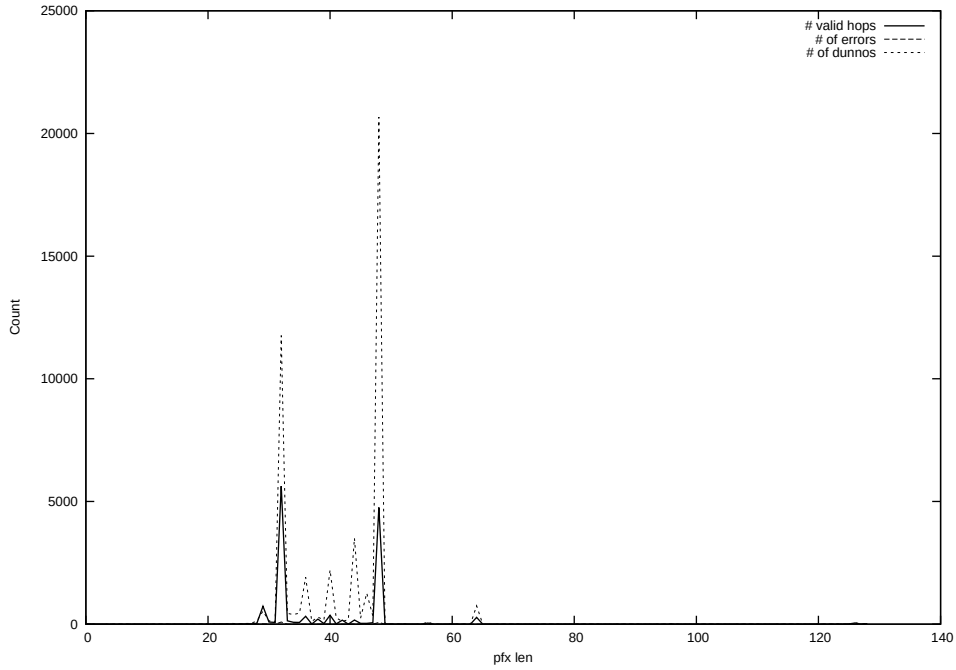
2014-07-30



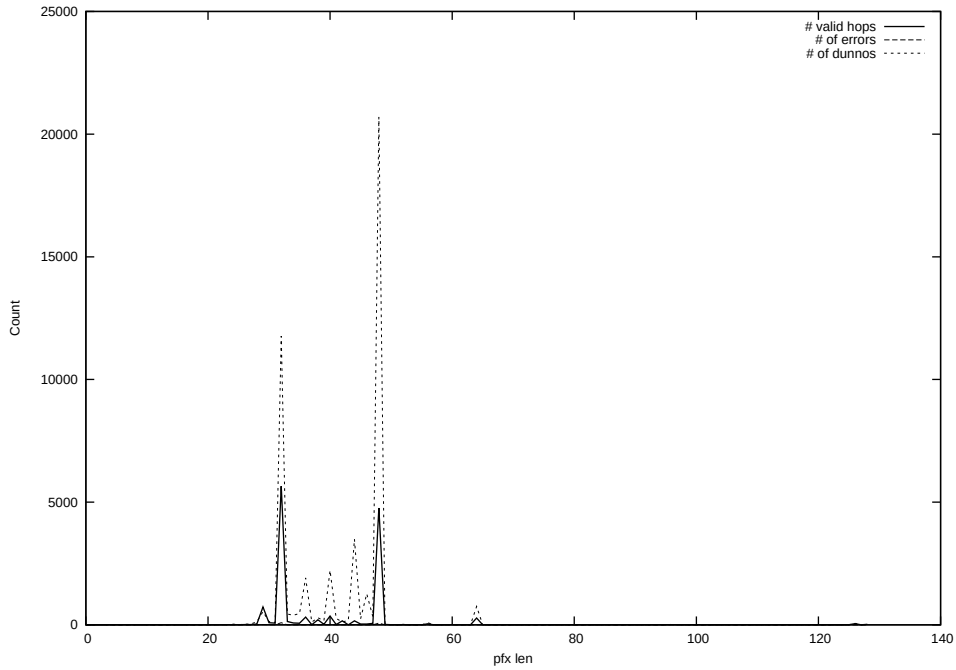
2014-07-31



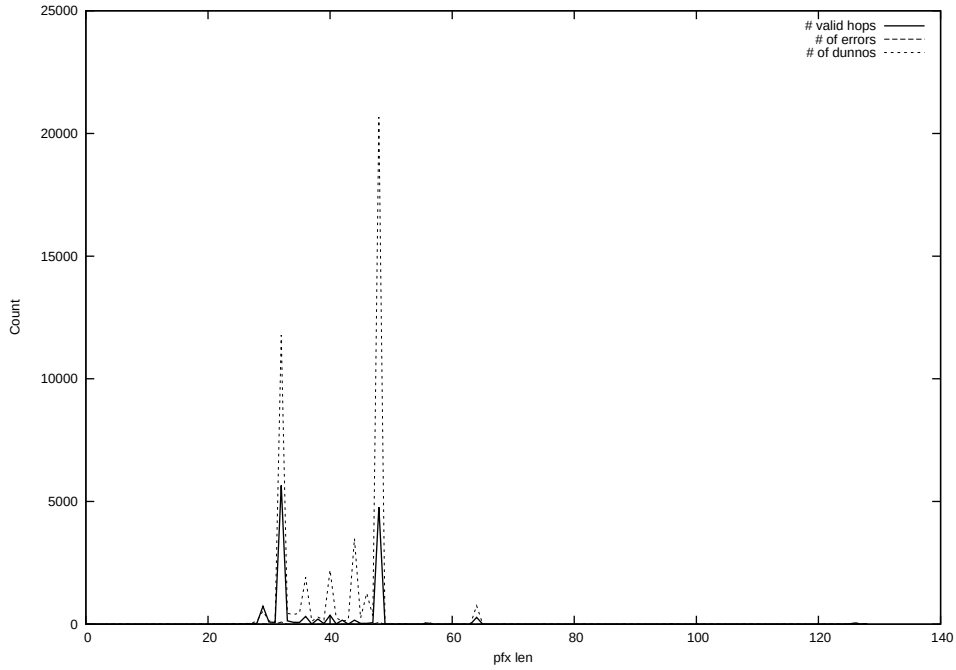
2014-08-01



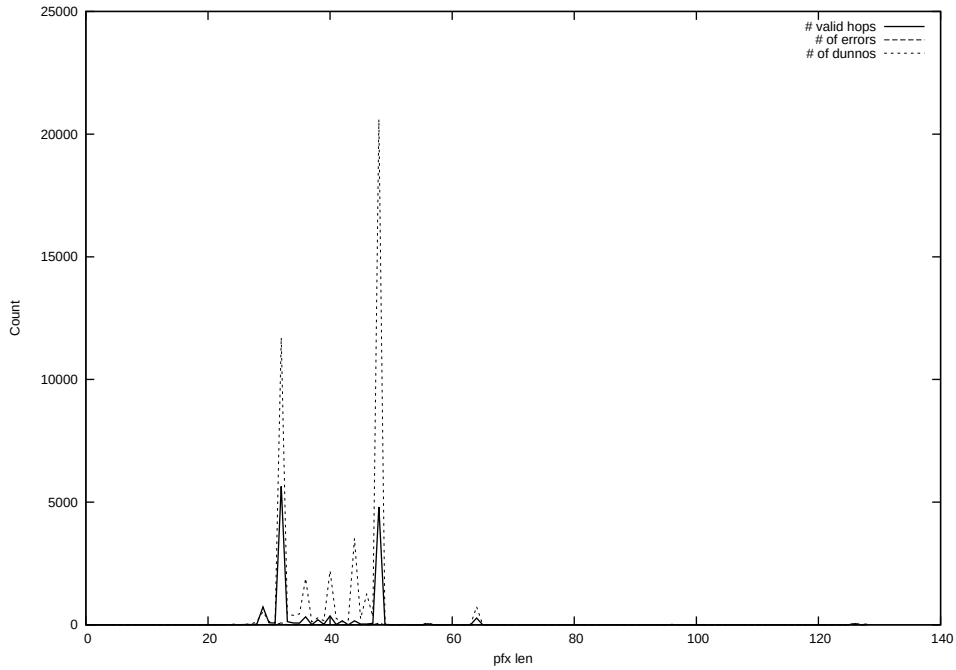
2014-08-02



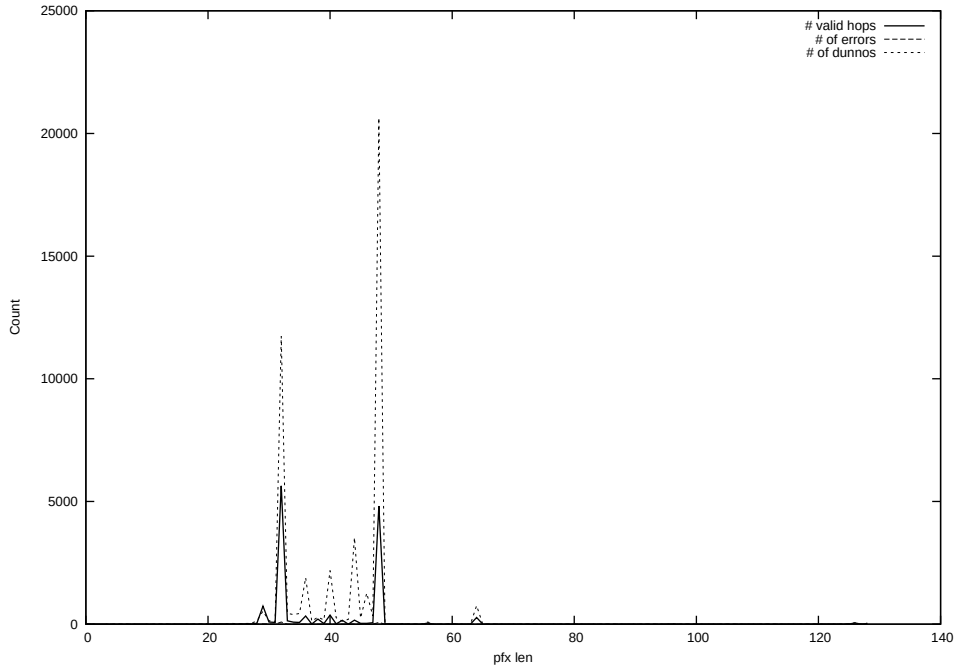
2014-08-03



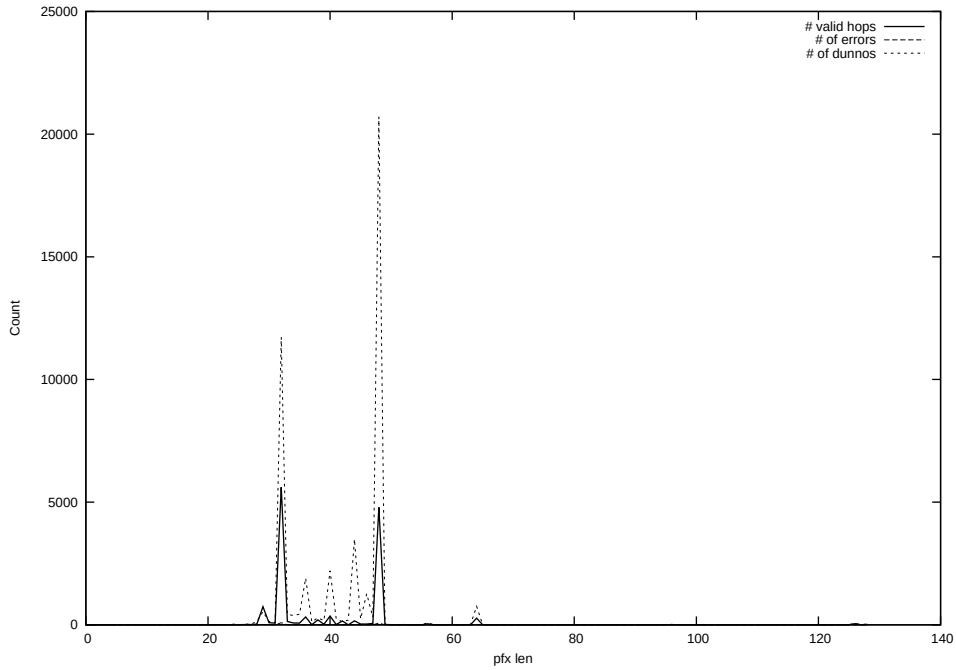
2014-08-04



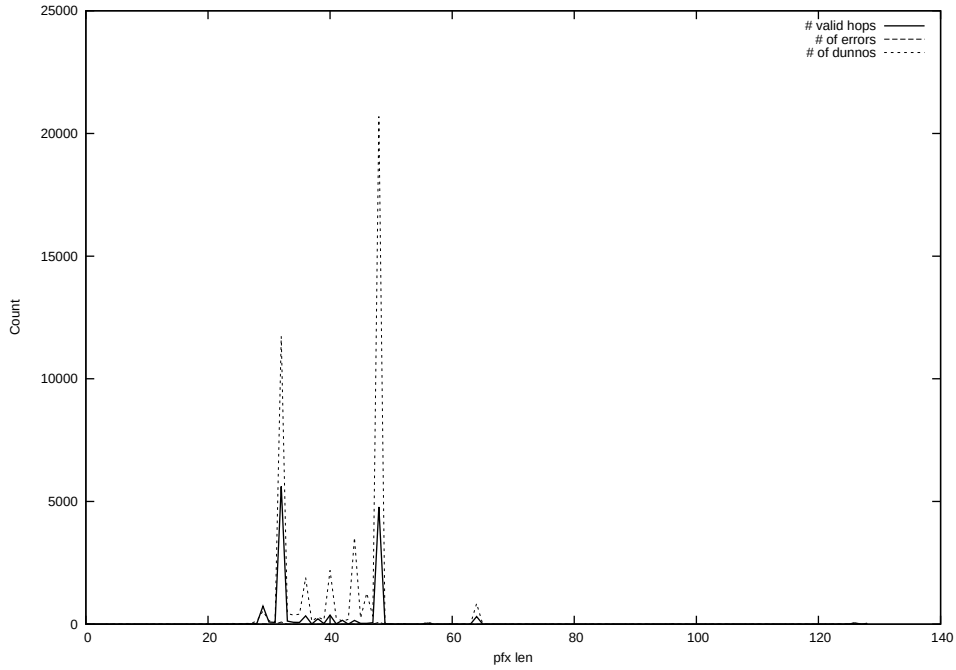
2014-08-05



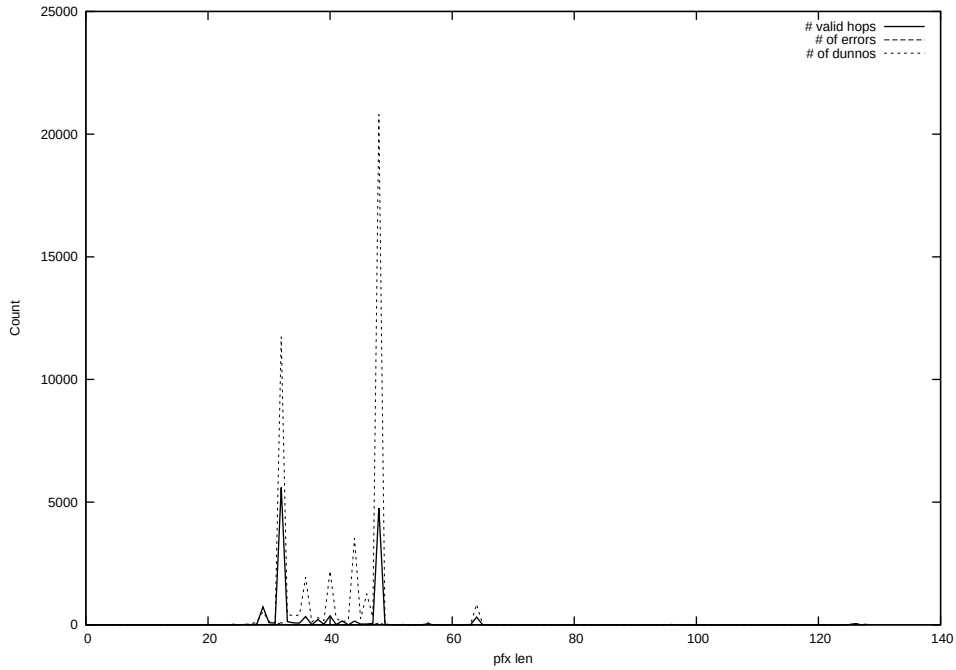
2014-08-06



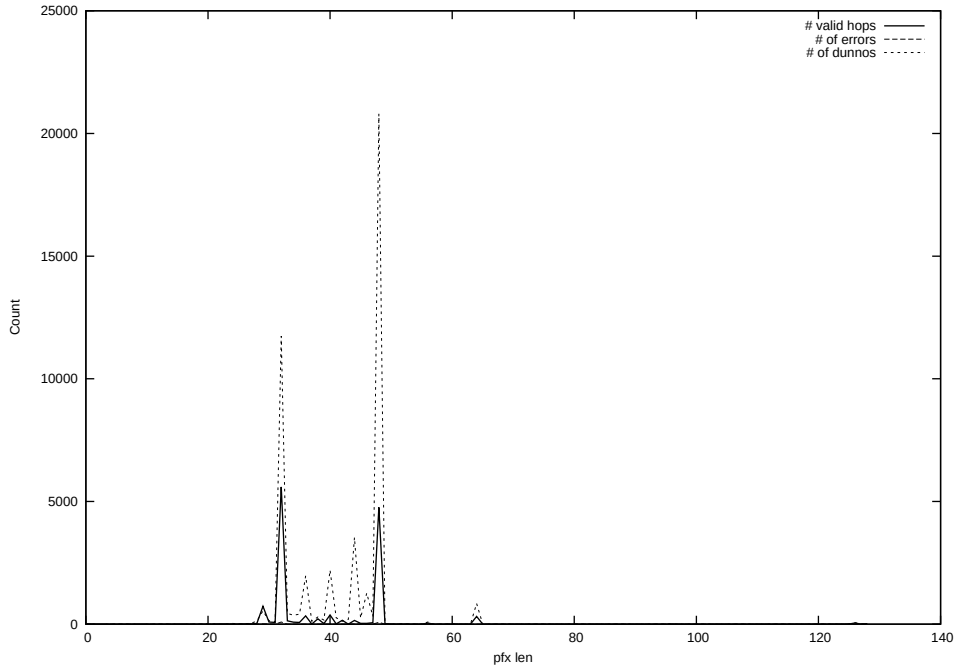
2014-08-07



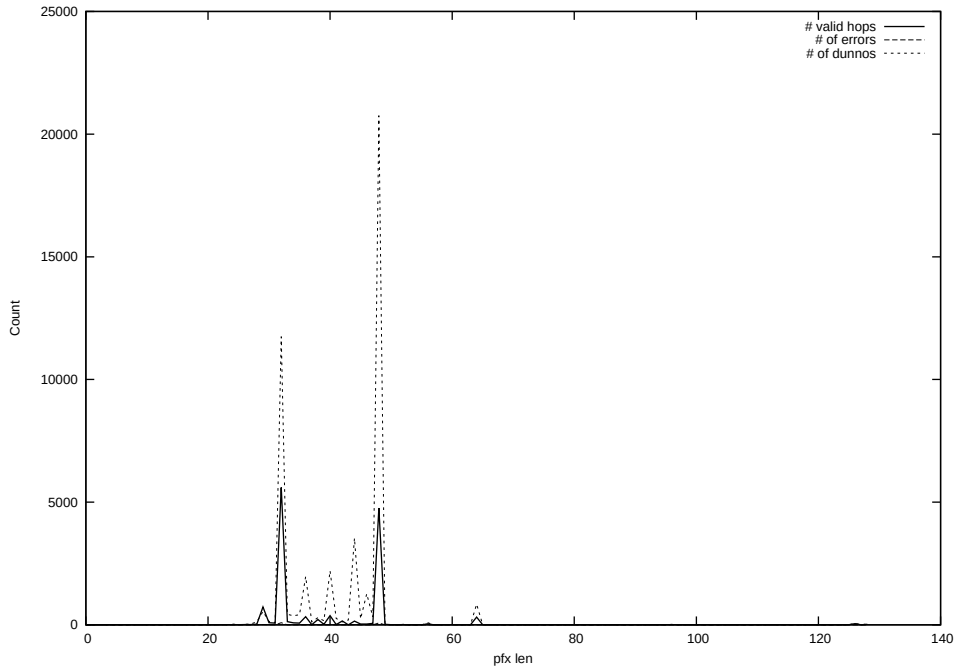
2014-08-08



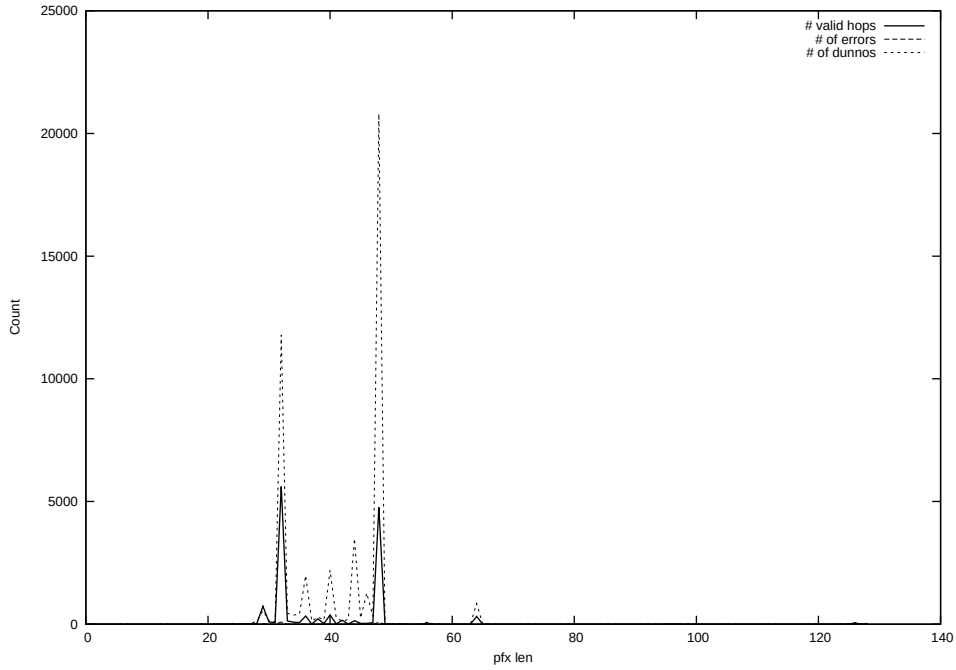
2014-08-09



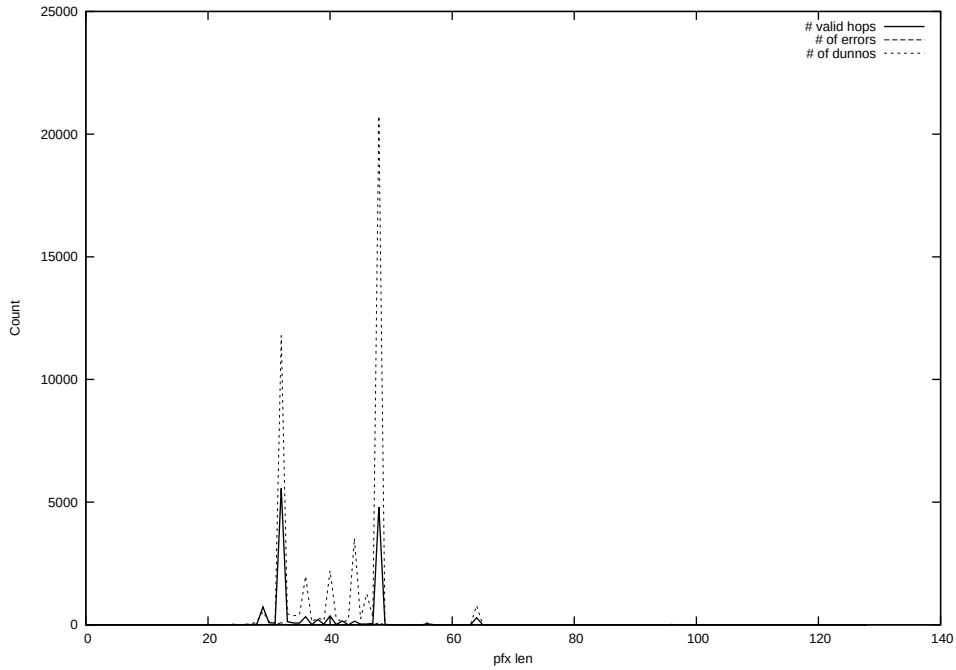
2014-08-10



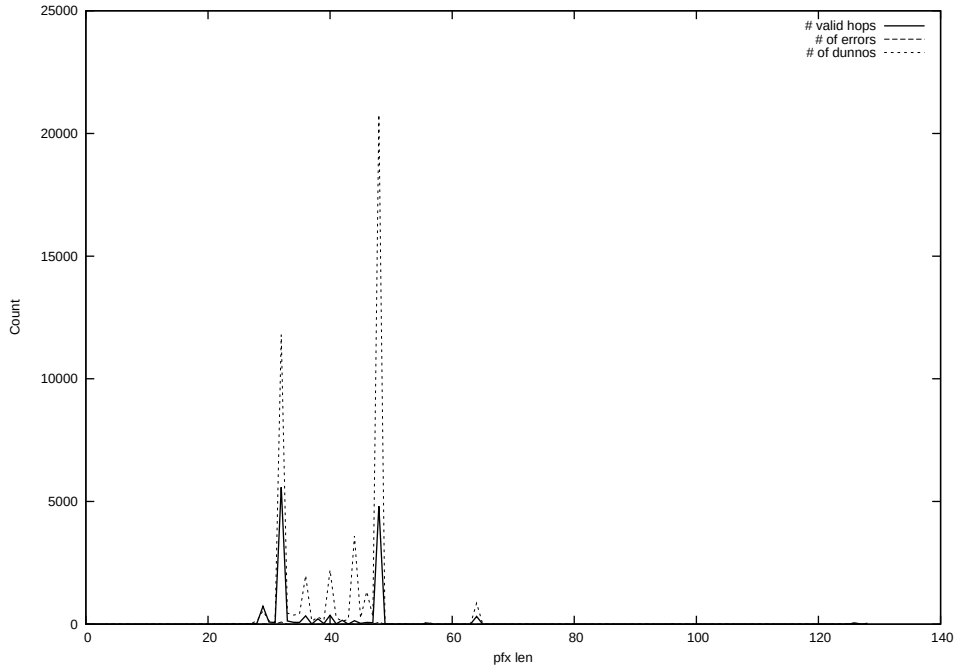
2014-08-11



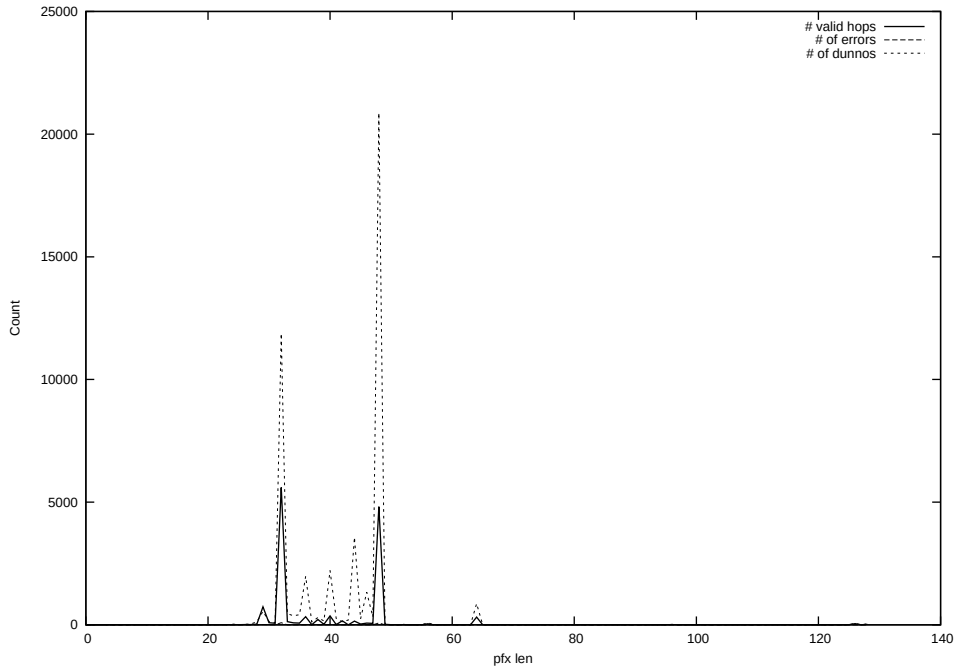
2014-08-12



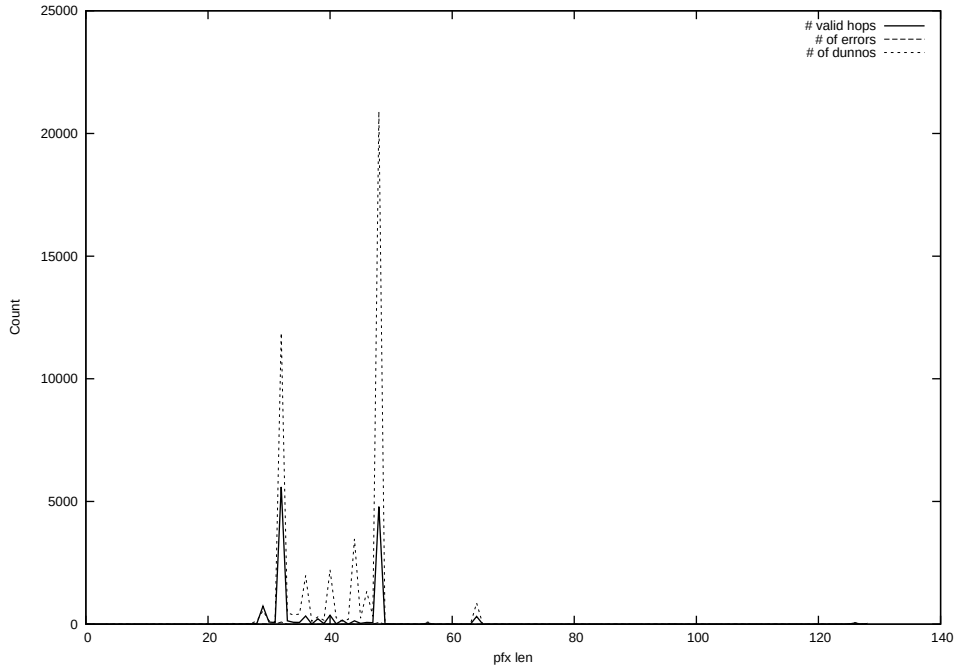
2014-08-13



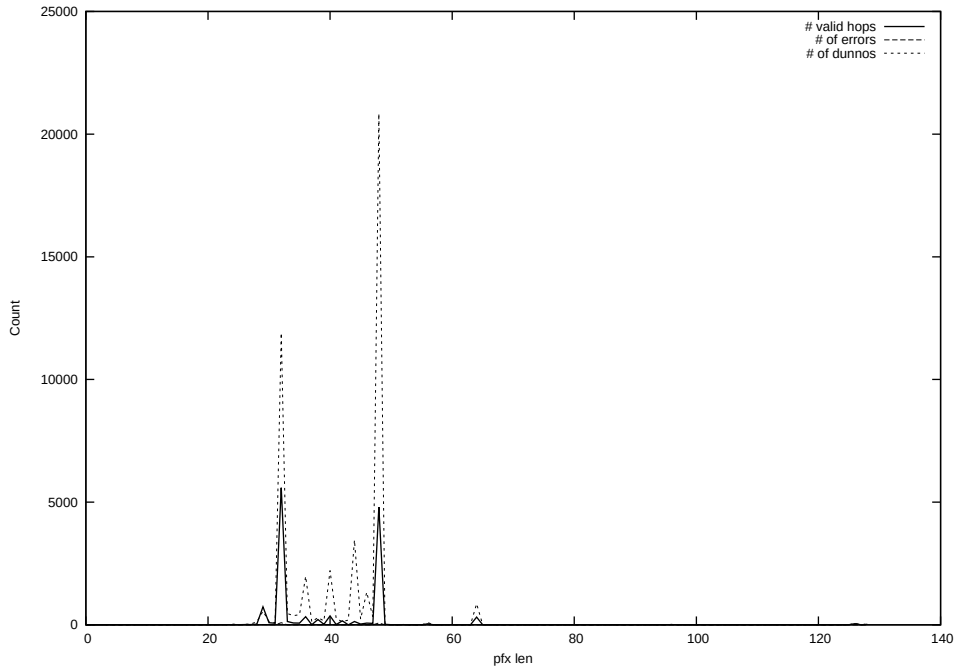
2014-08-14



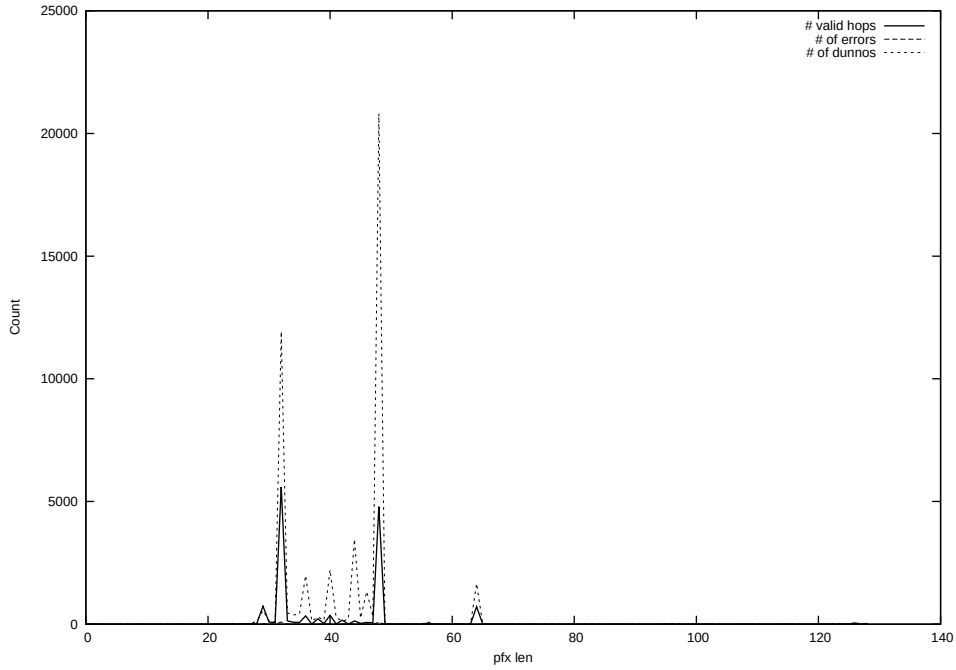
2014-08-15



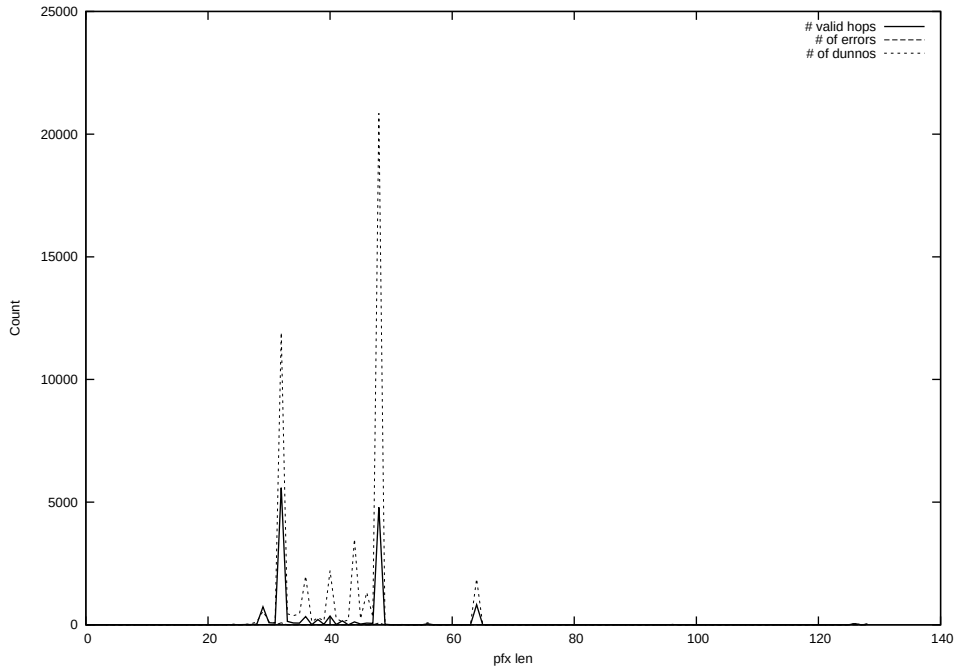
2014-08-16



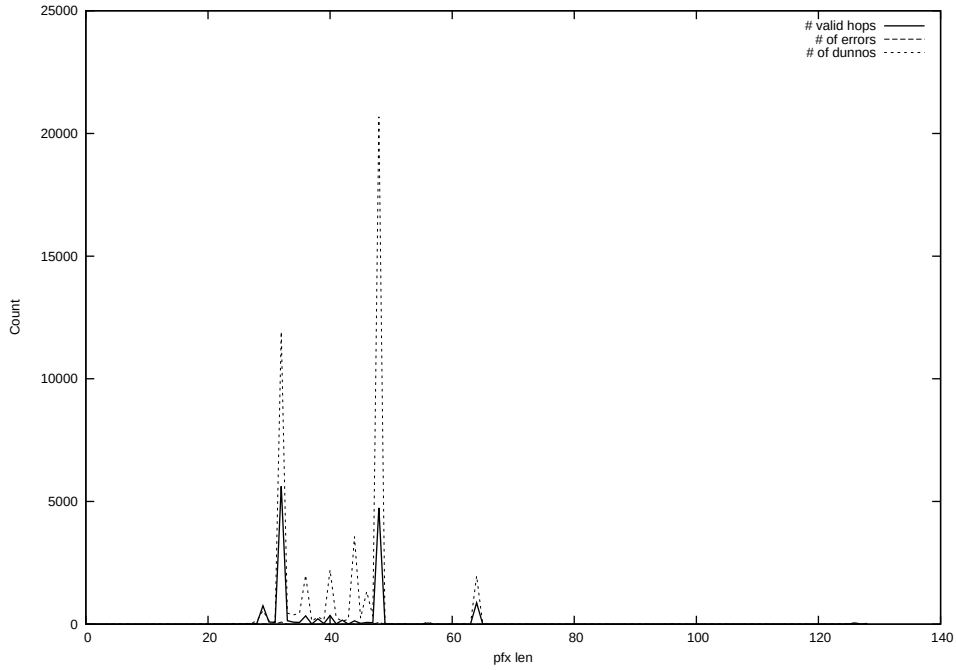
2014-08-17



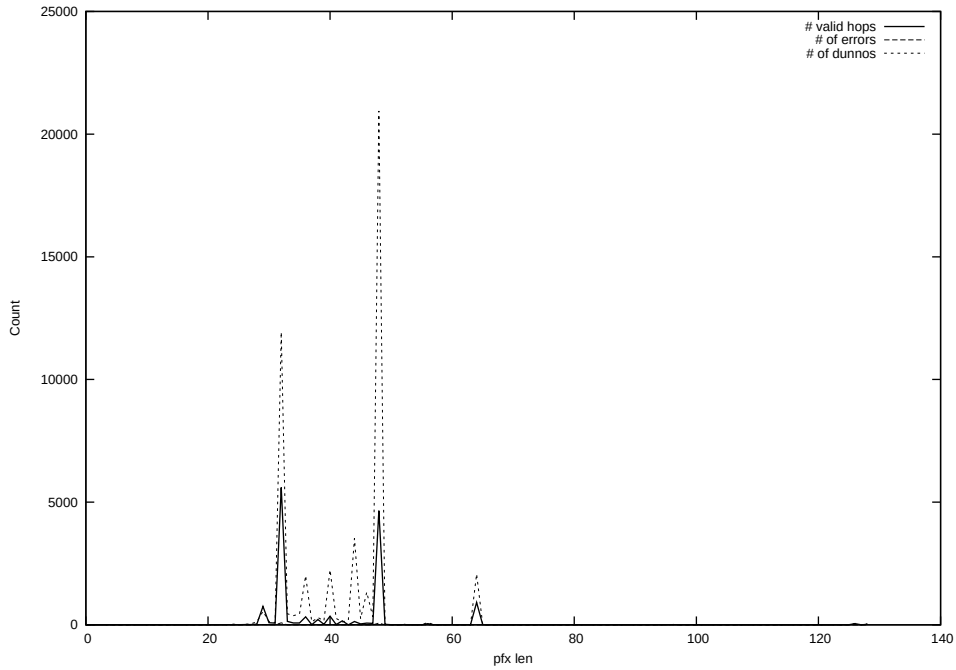
2014-08-18



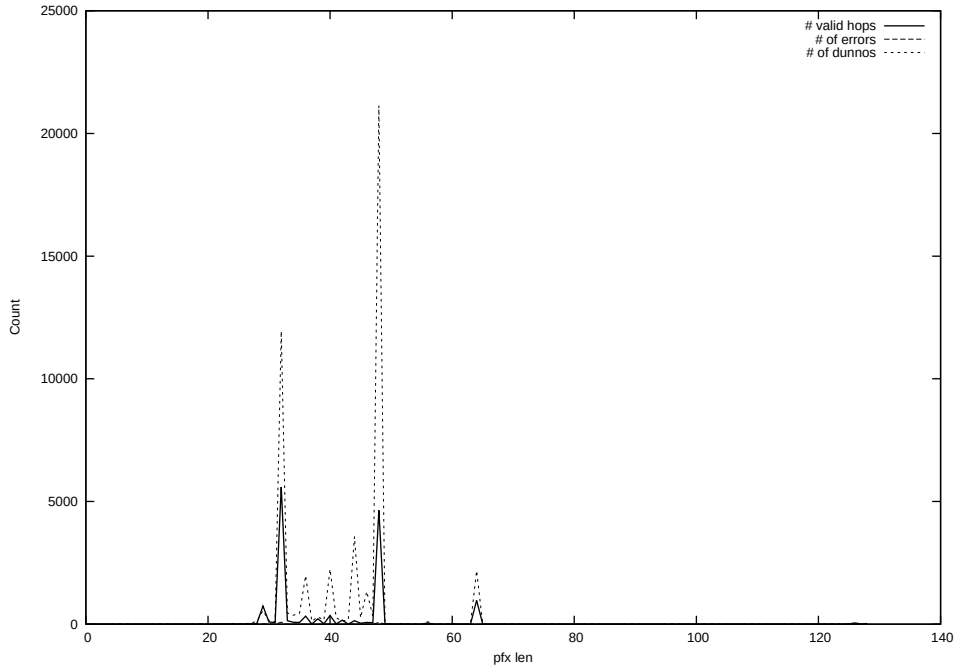
2014-08-19



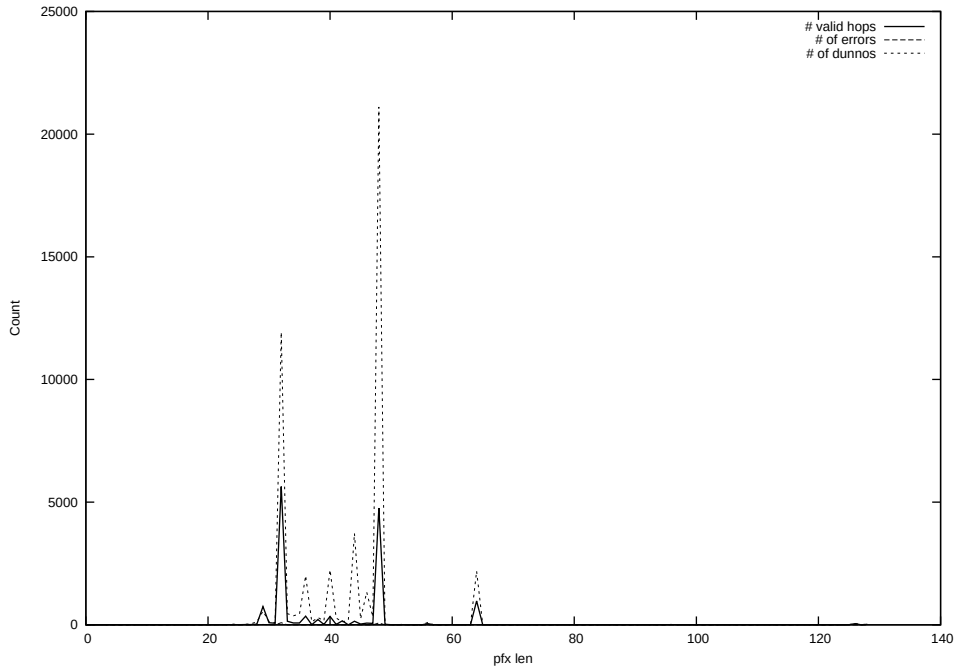
2014-08-20



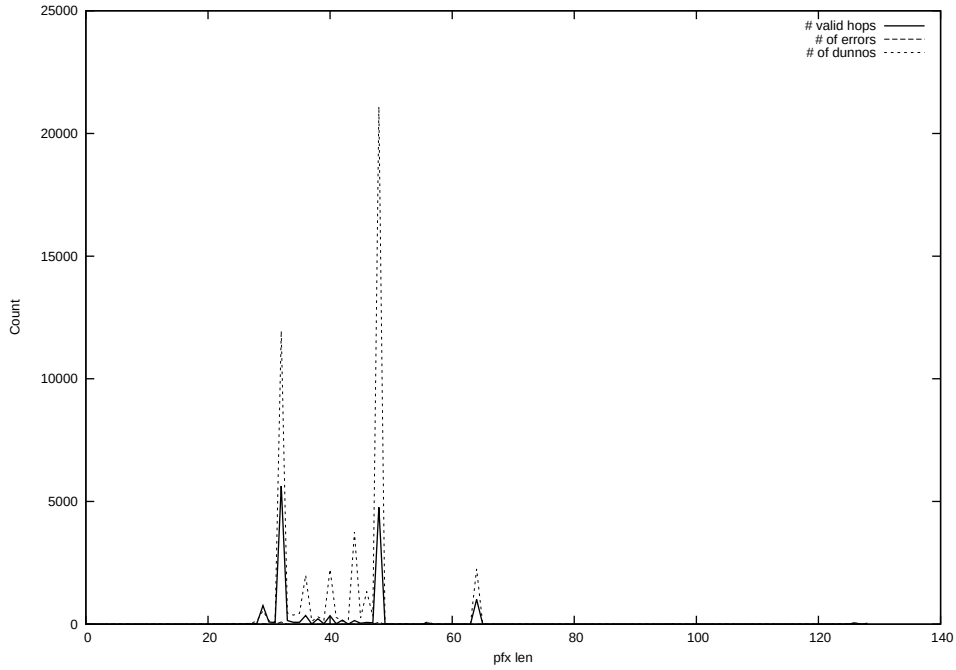
2014-08-21



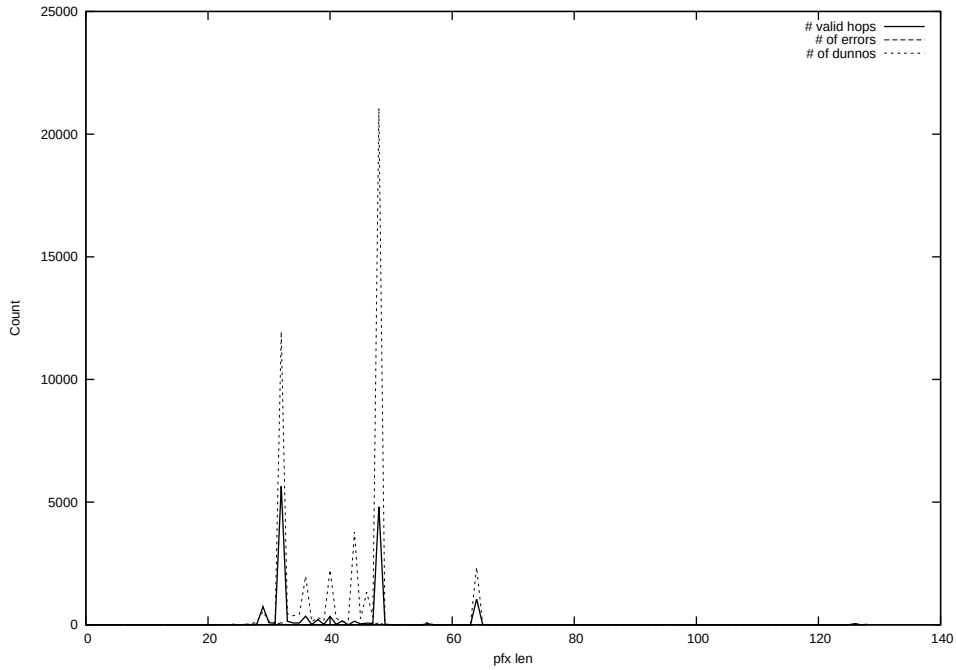
2014-08-22



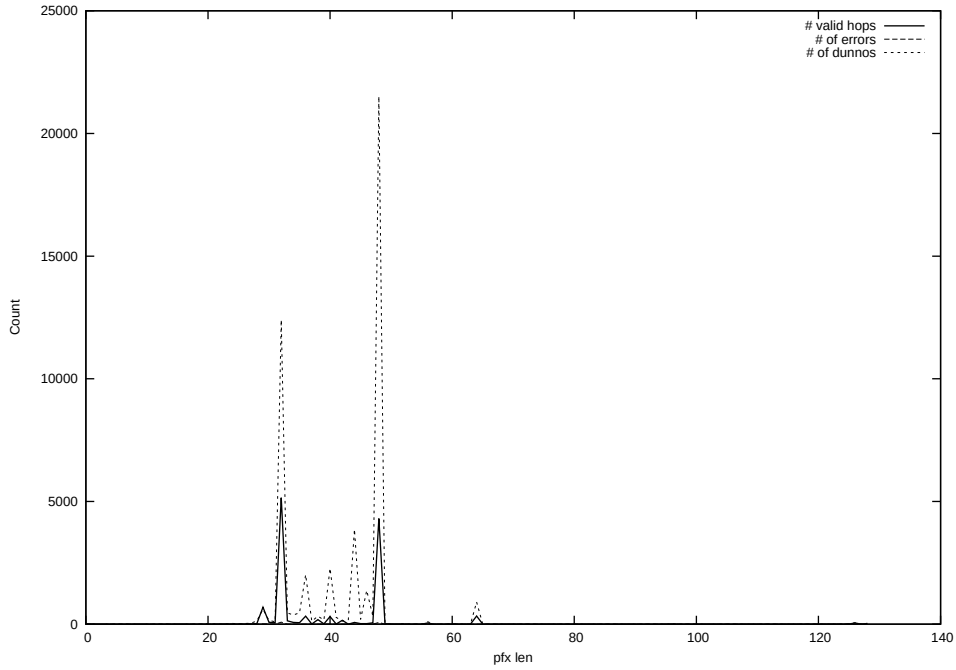
2014-08-23



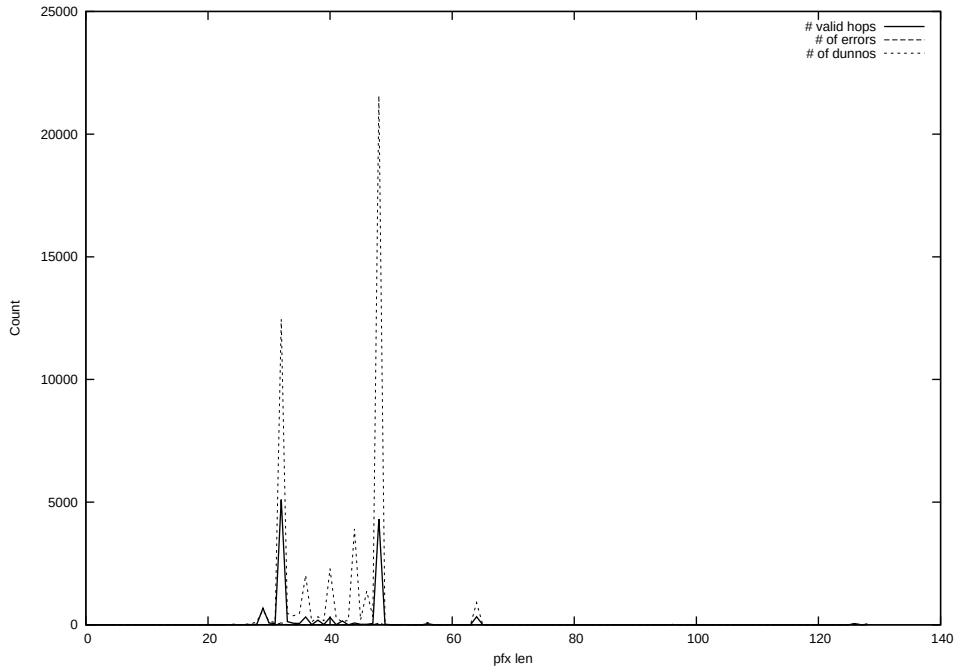
2014-08-24



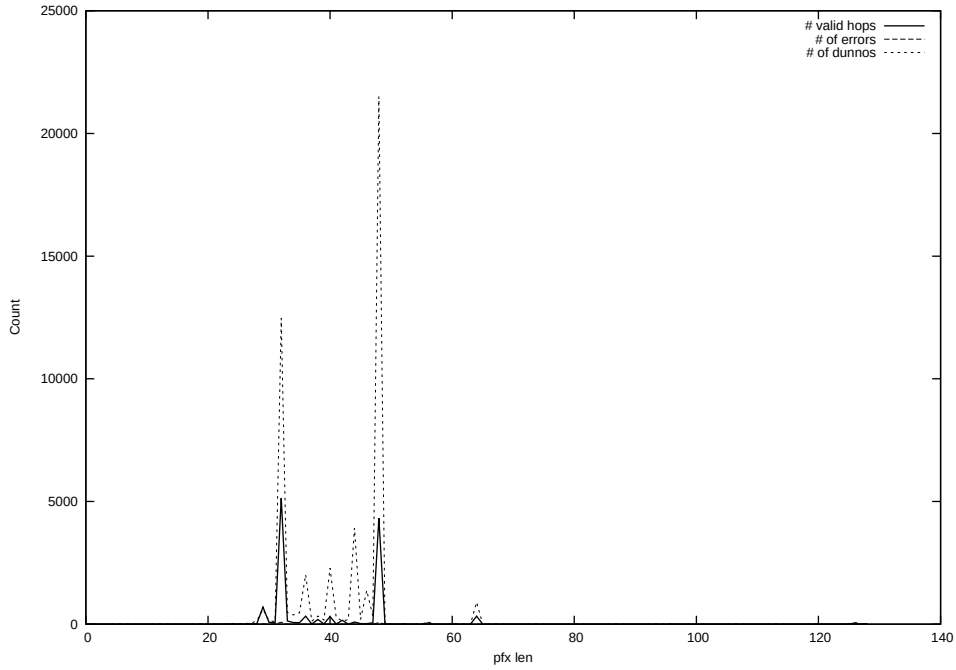
2014-08-25



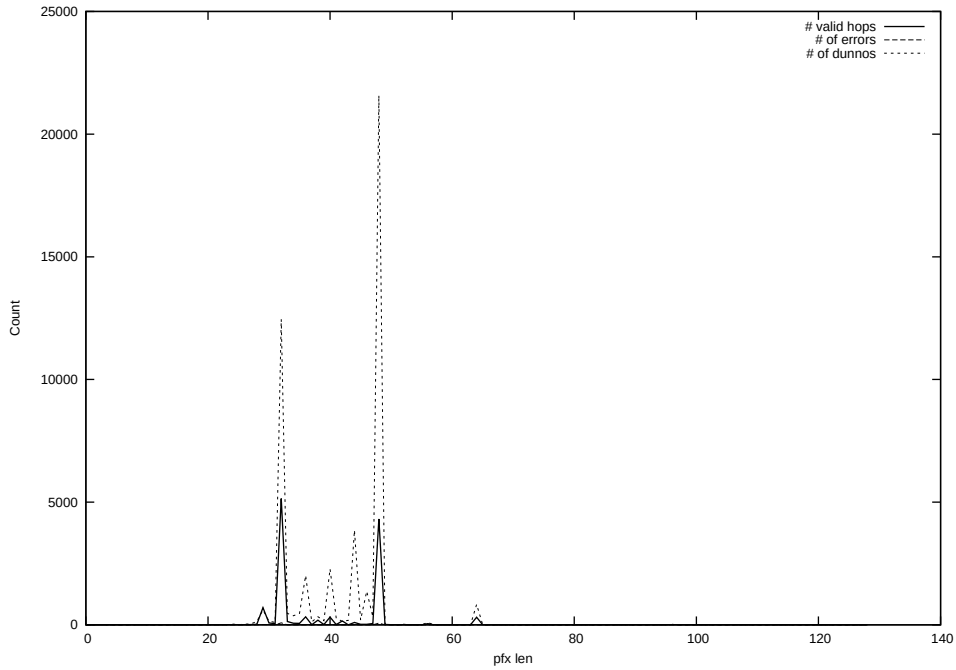
2014-08-26



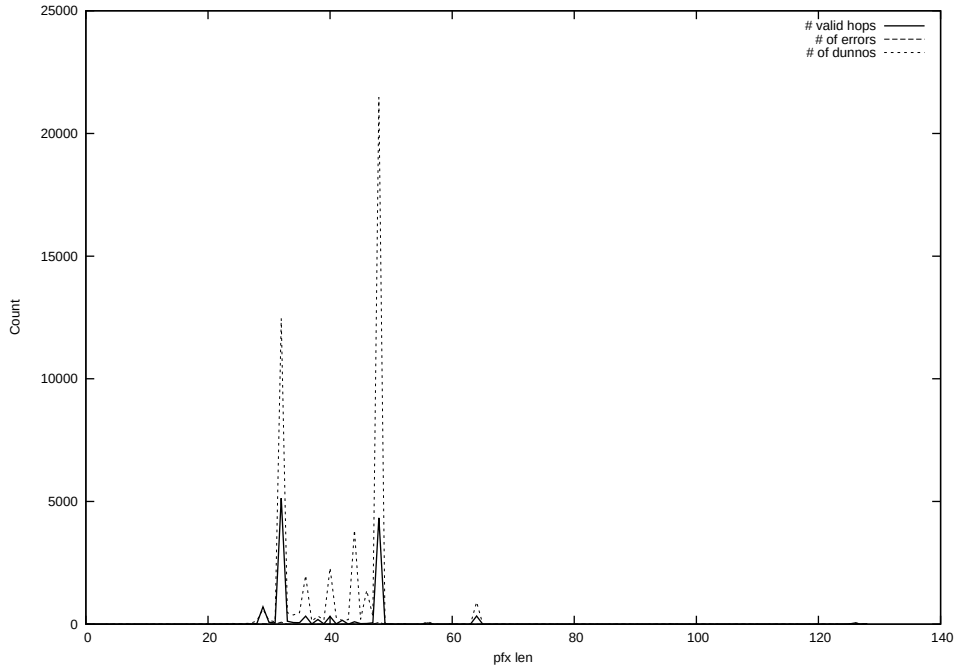
2014-08-27



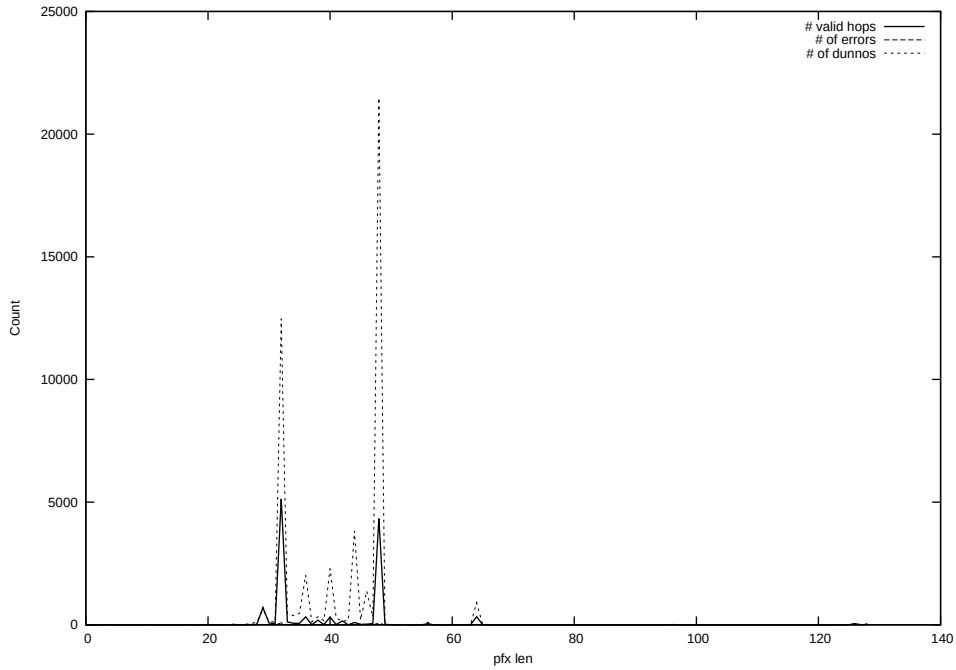
2014-08-28



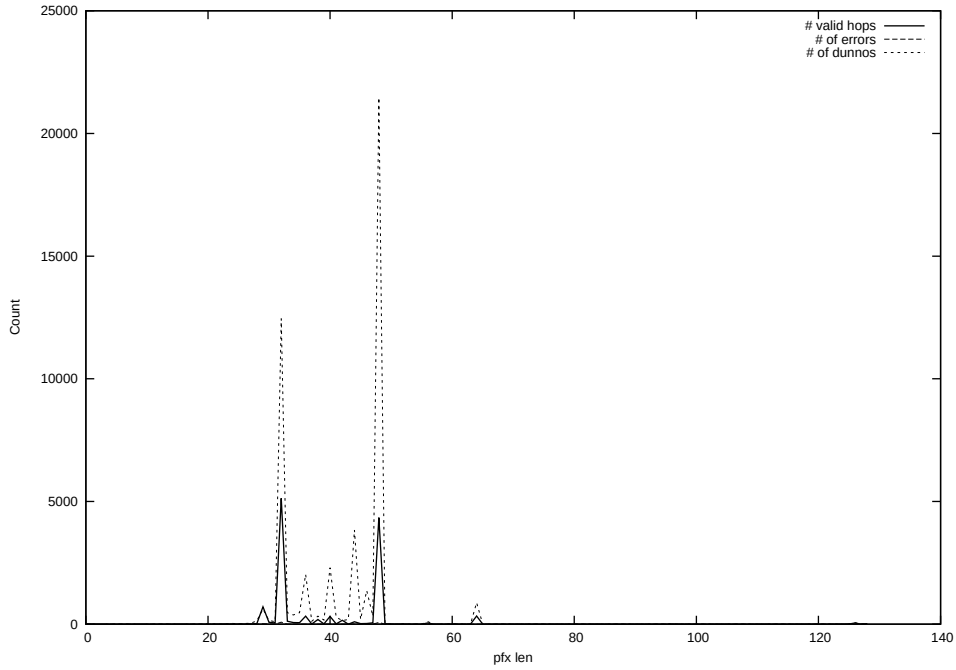
2014-08-29



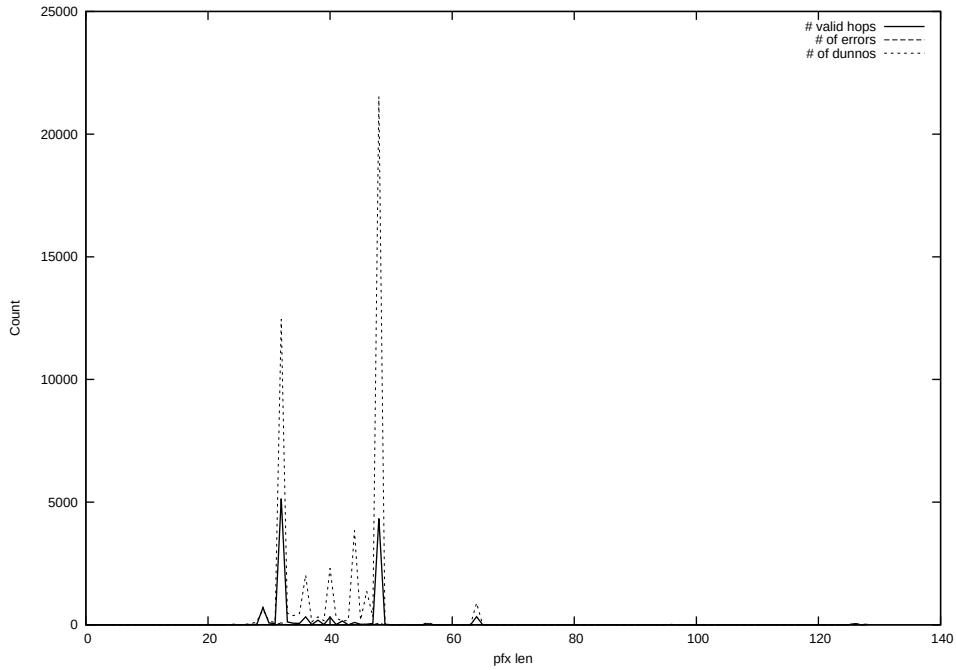
2014-08-30



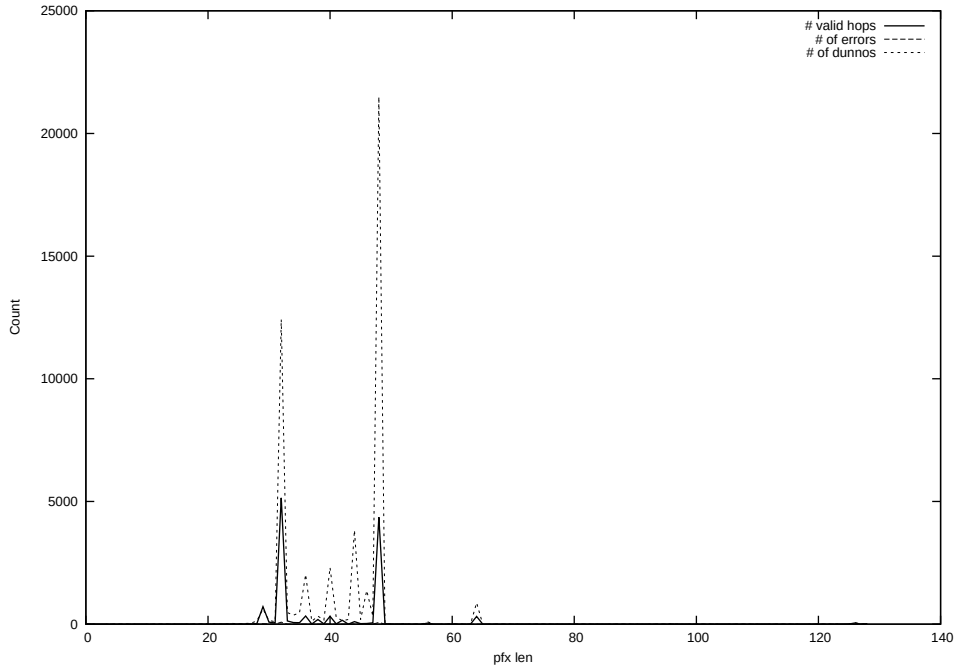
2014-08-31



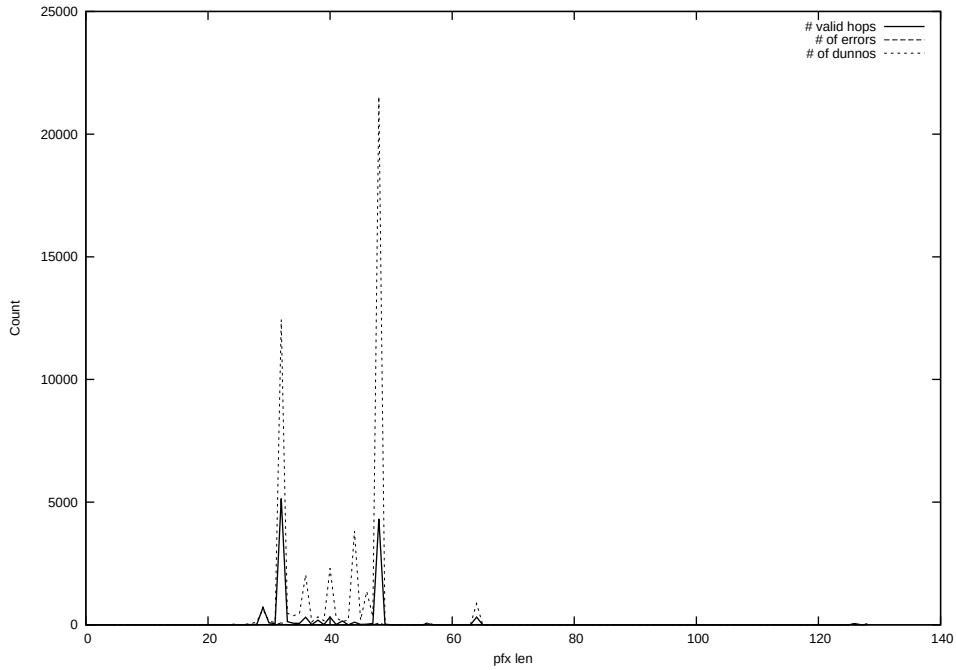
2014-09-01



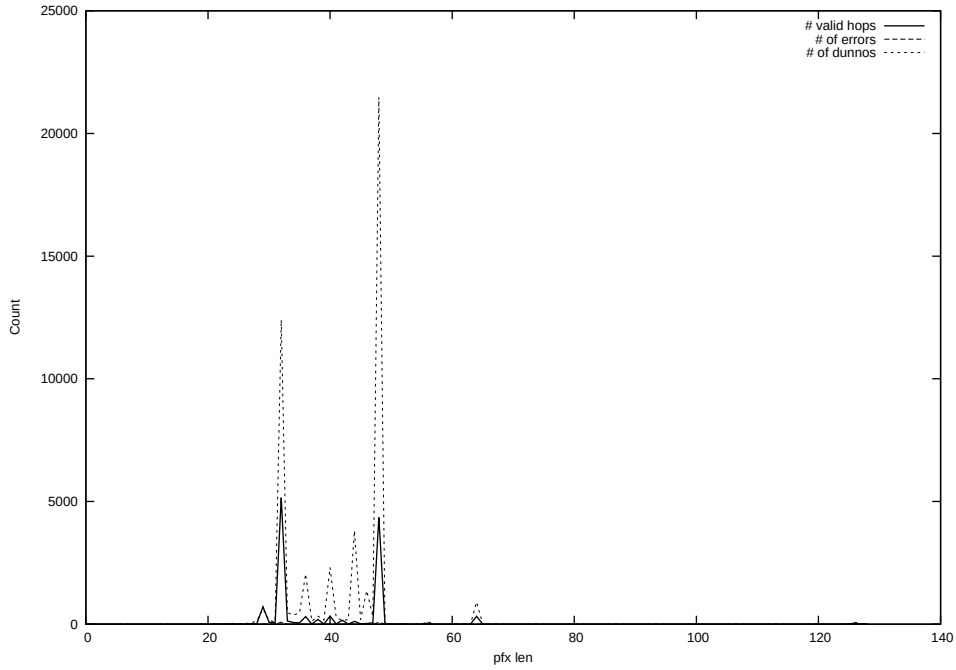
2014-09-02



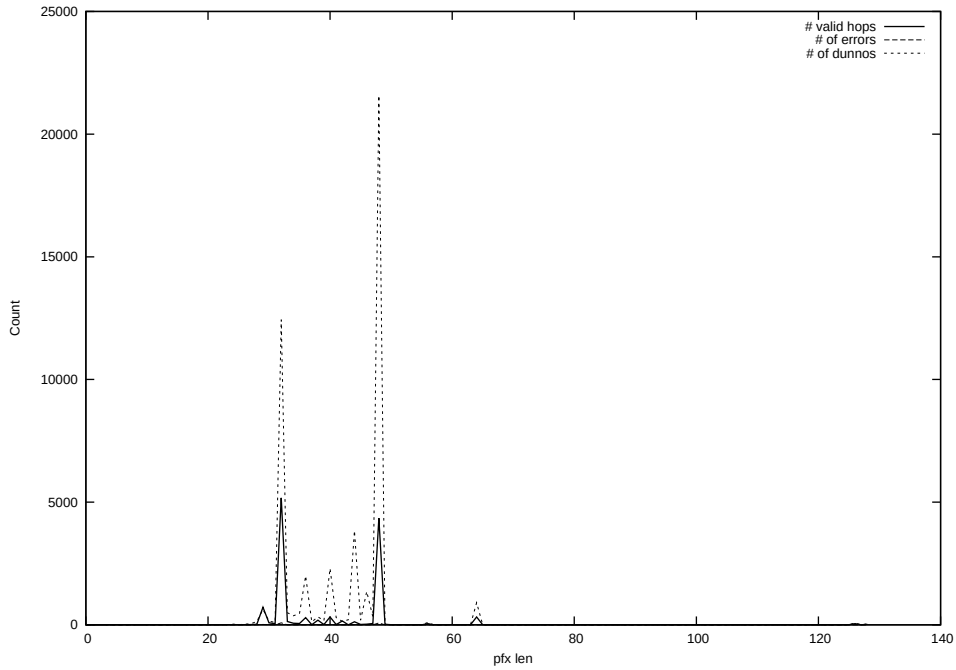
2014-09-03



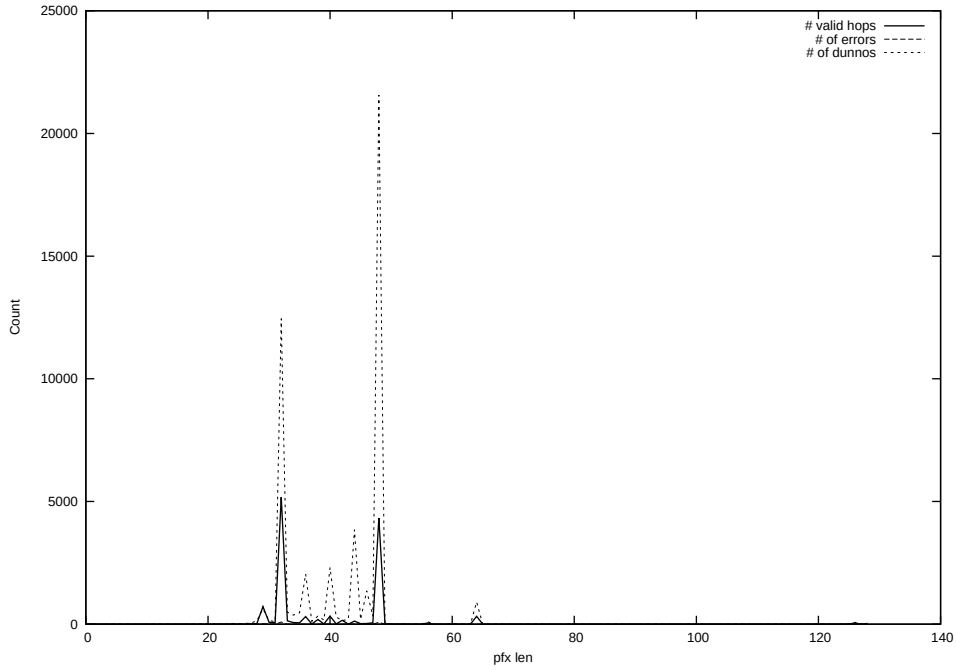
2014-09-04



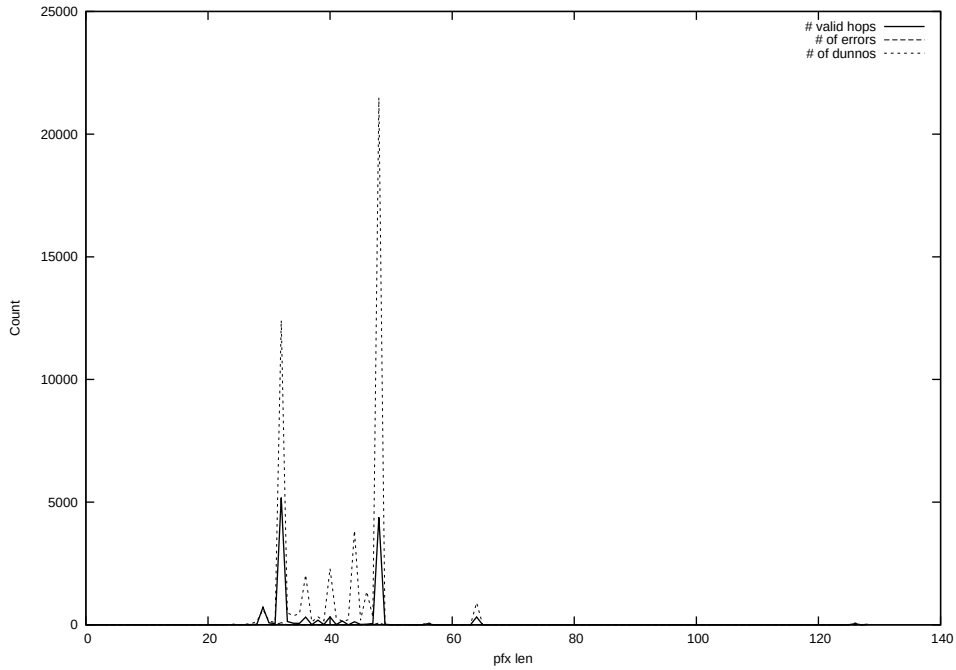
2014-09-05



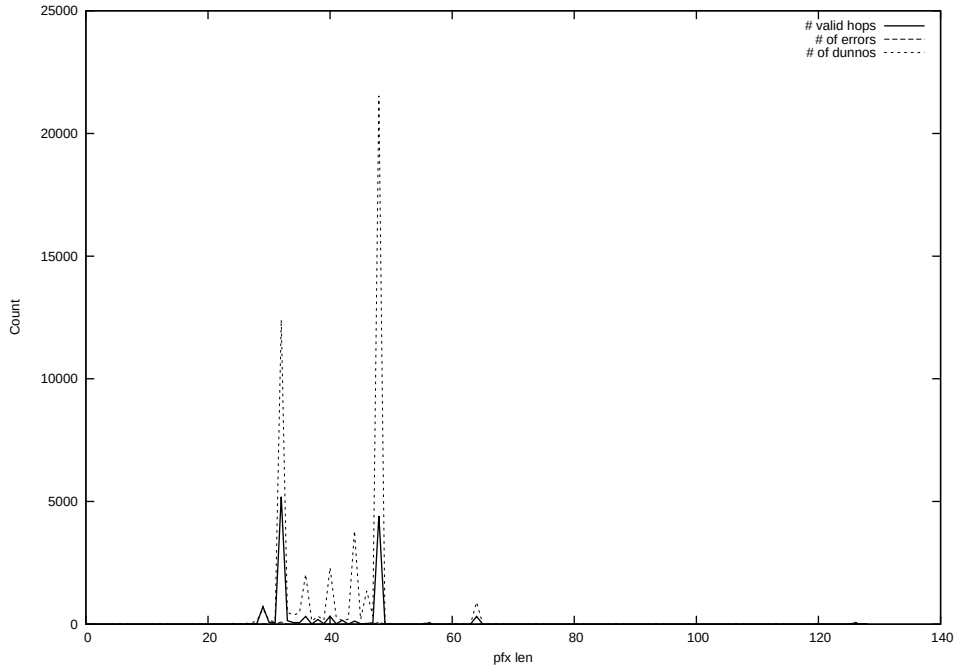
2014-09-06



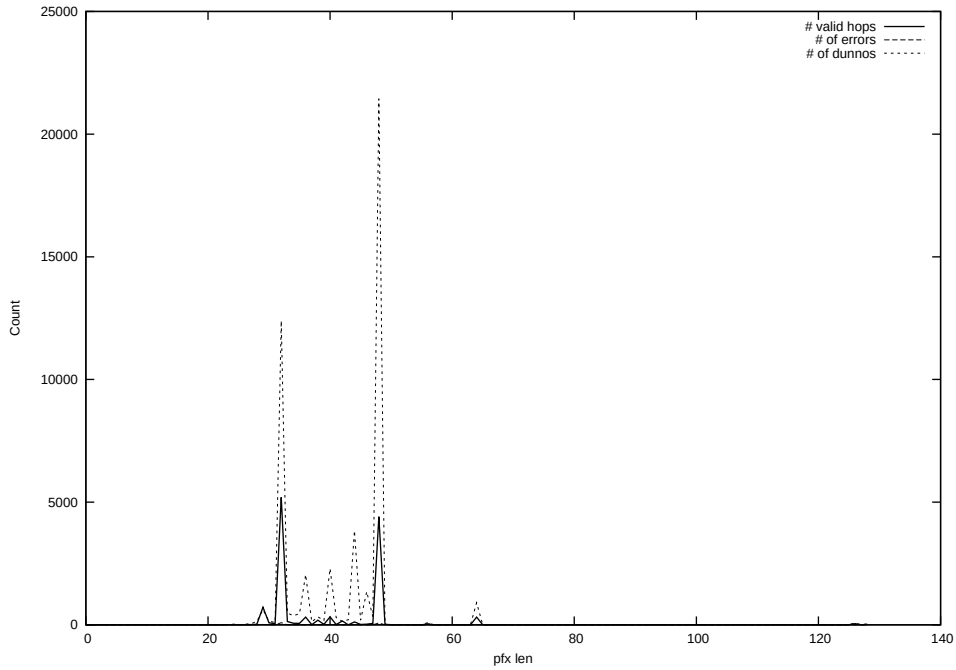
2014-09-07



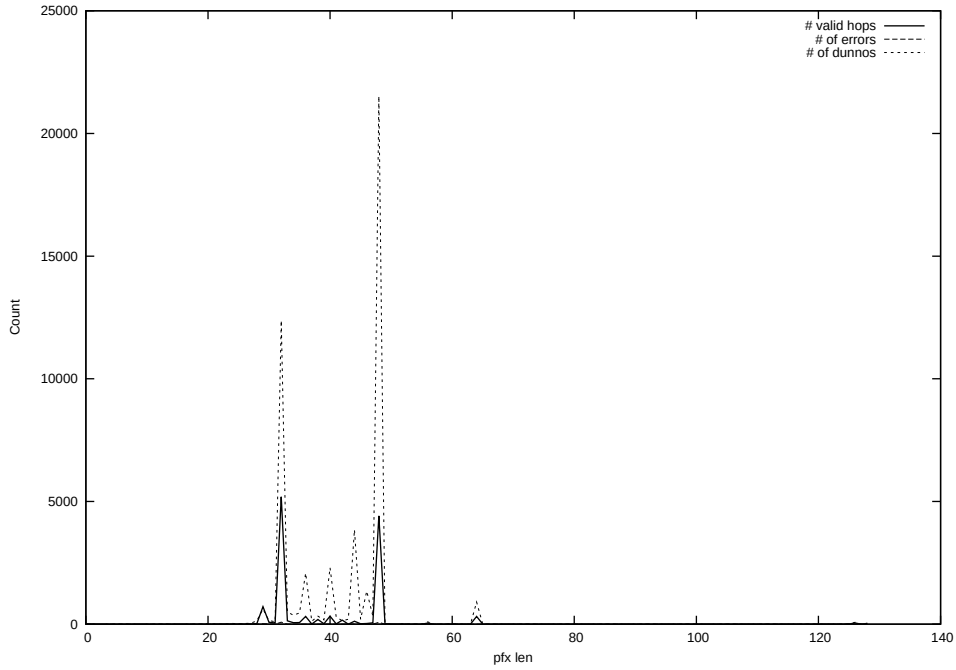
2014-09-08



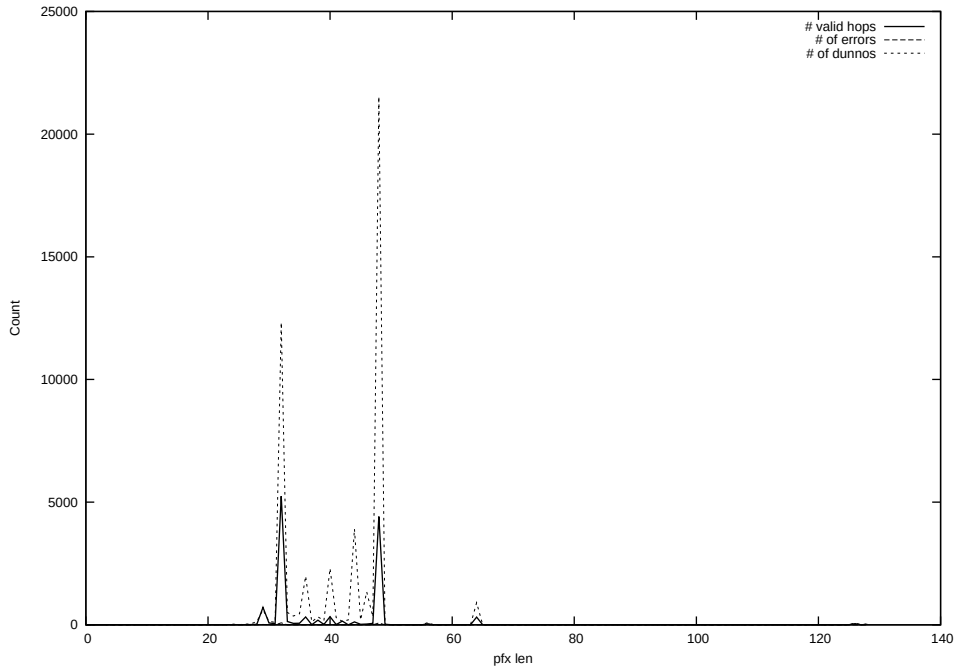
2014-09-09



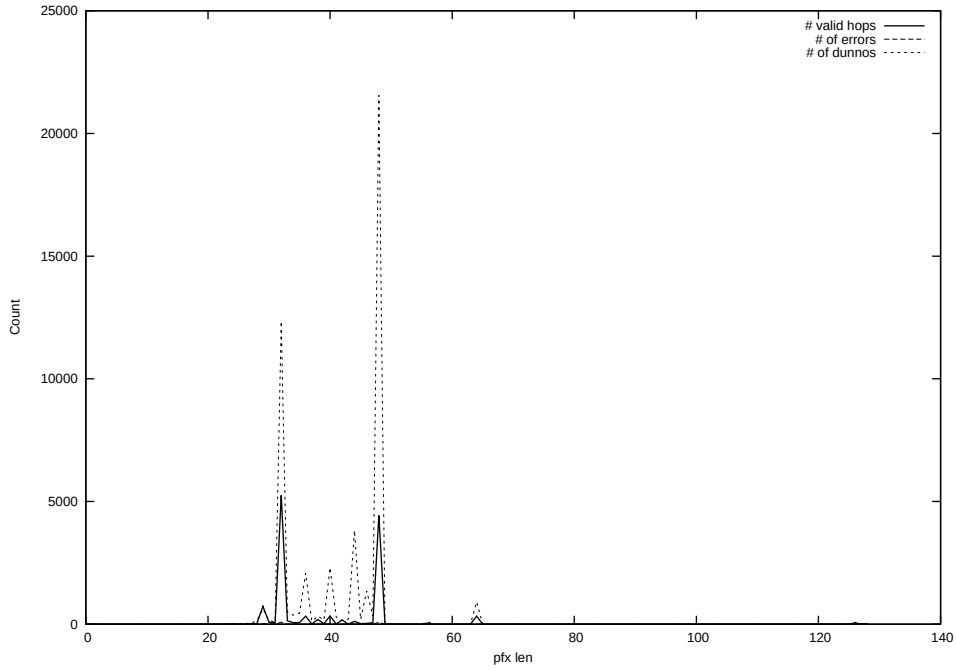
2014-09-10



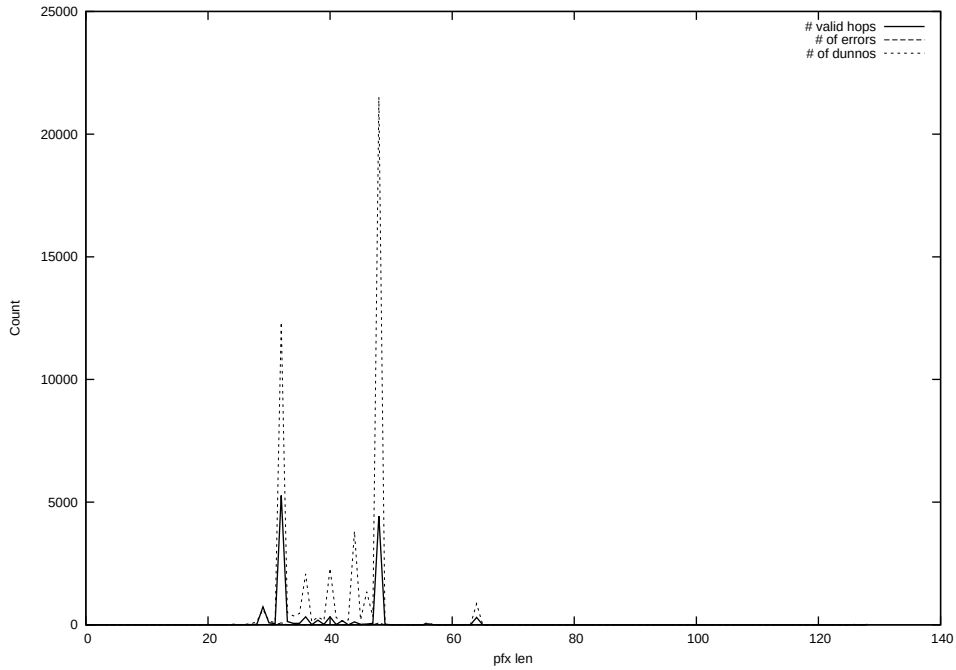
2014-09-11



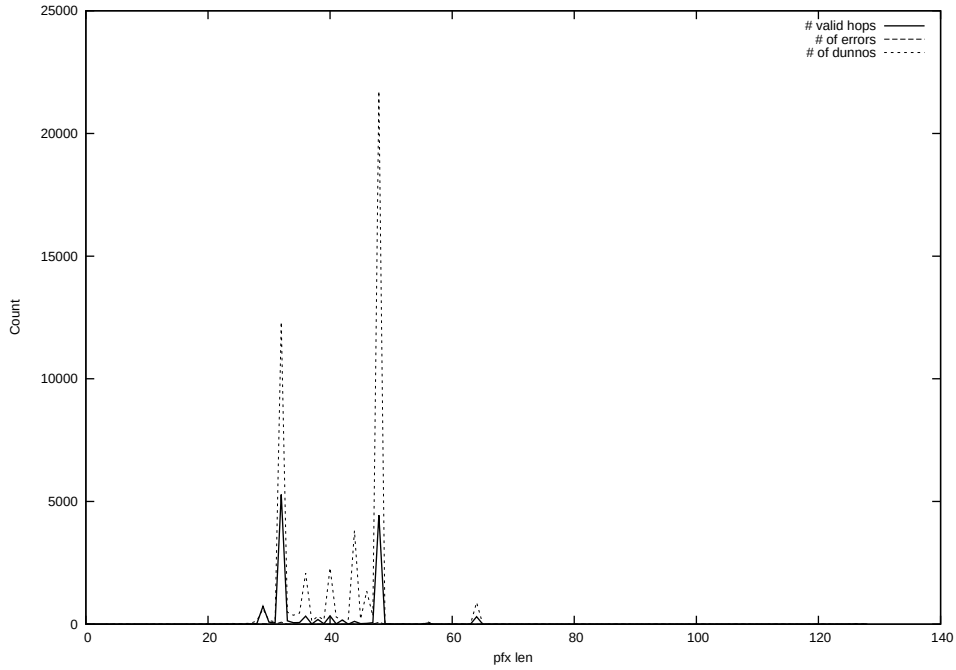
2014-09-12



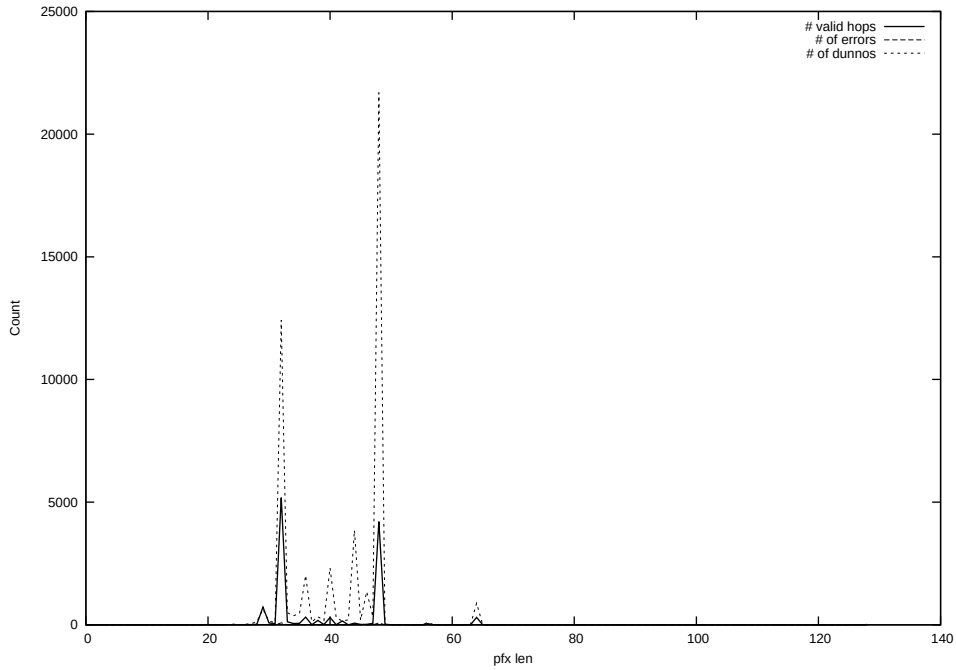
2014-09-13



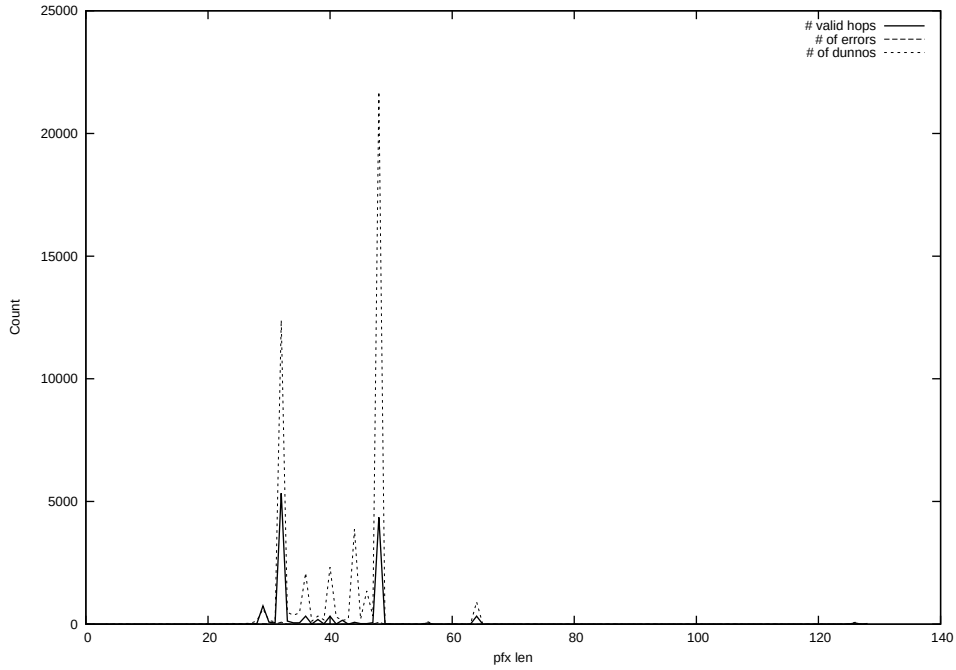
2014-09-14



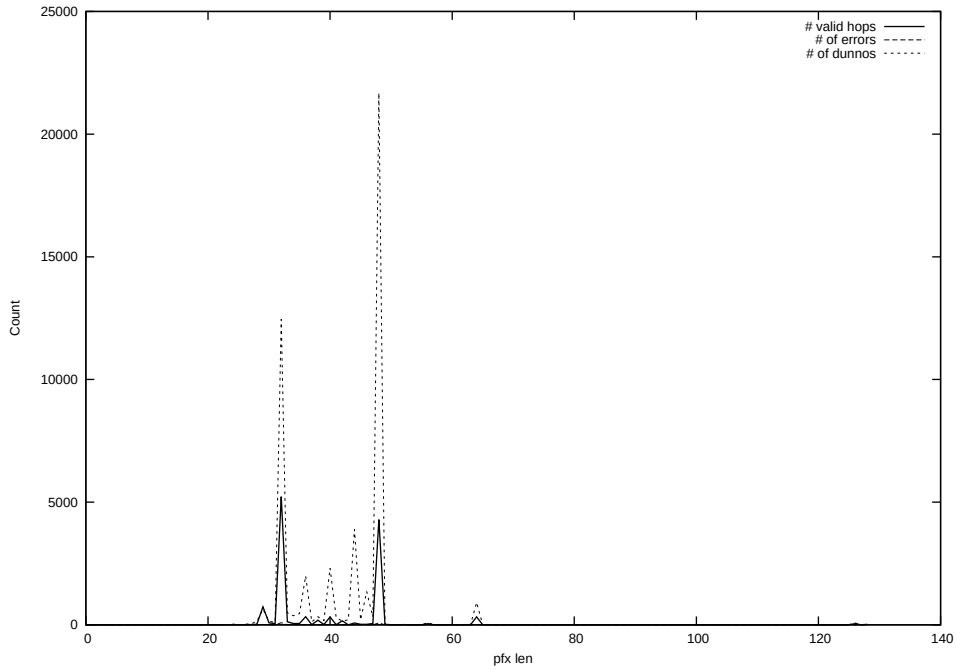
2014-09-15



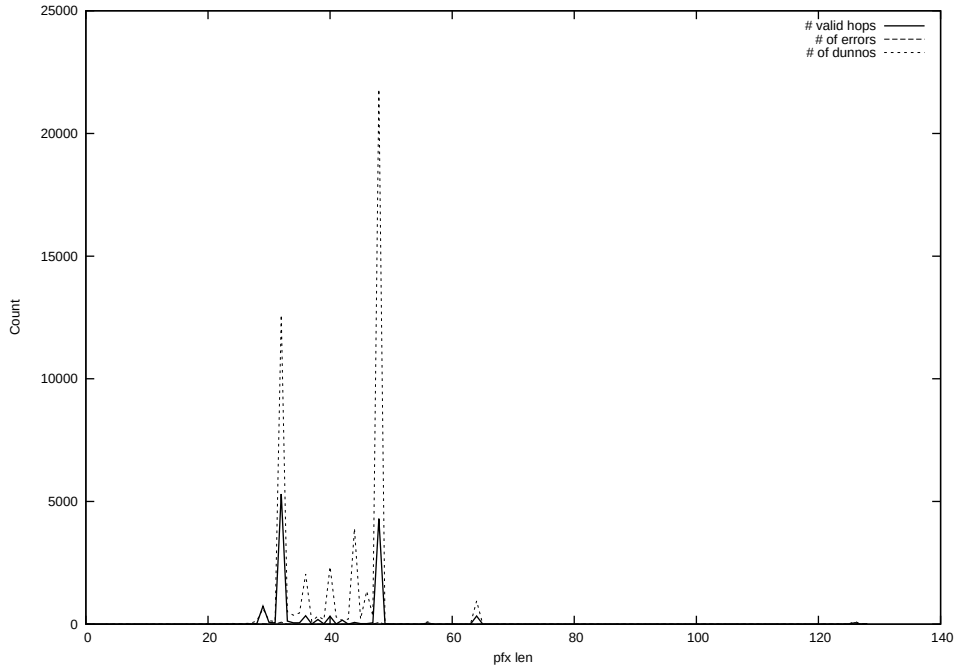
2014-09-16



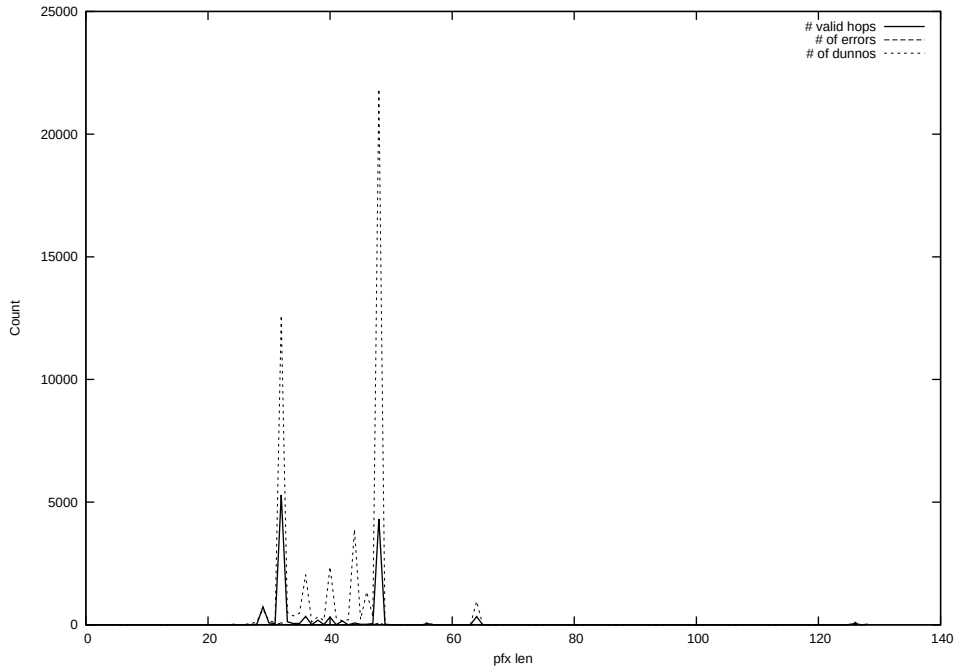
2014-09-17



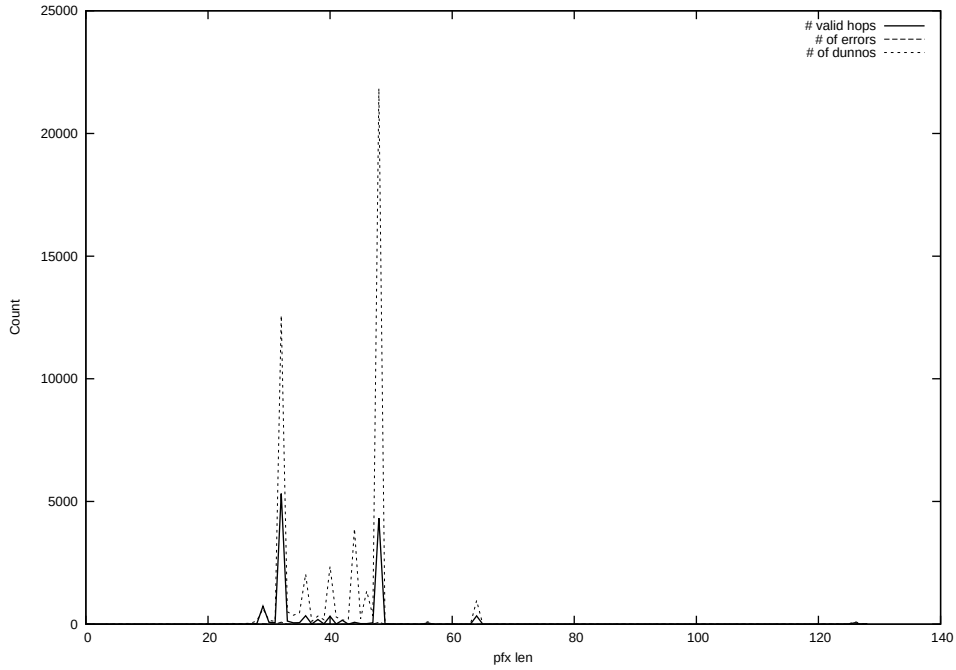
2014-09-18



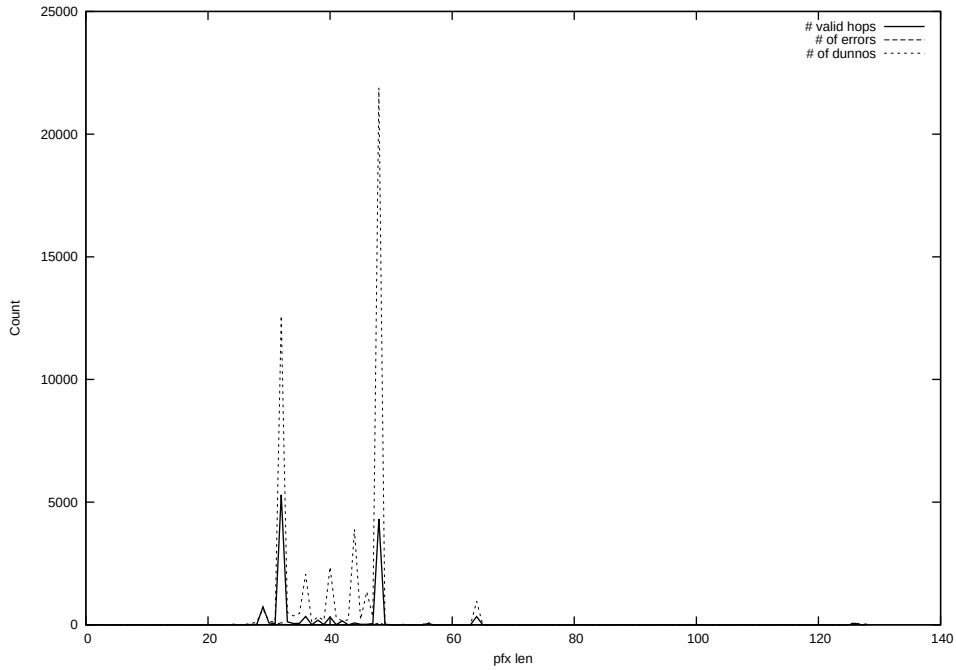
2014-09-19



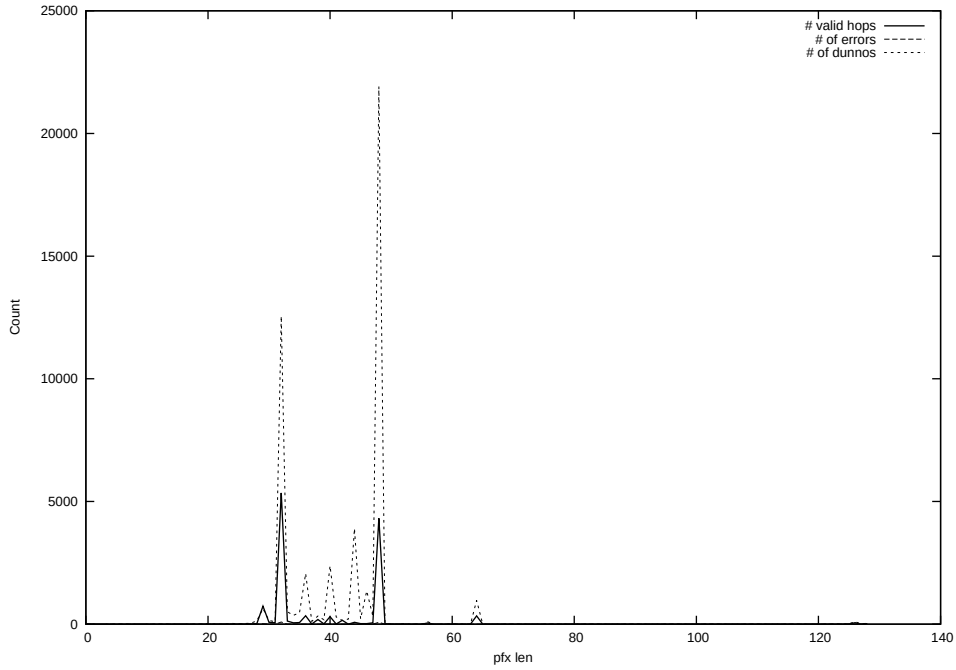
2014-09-20



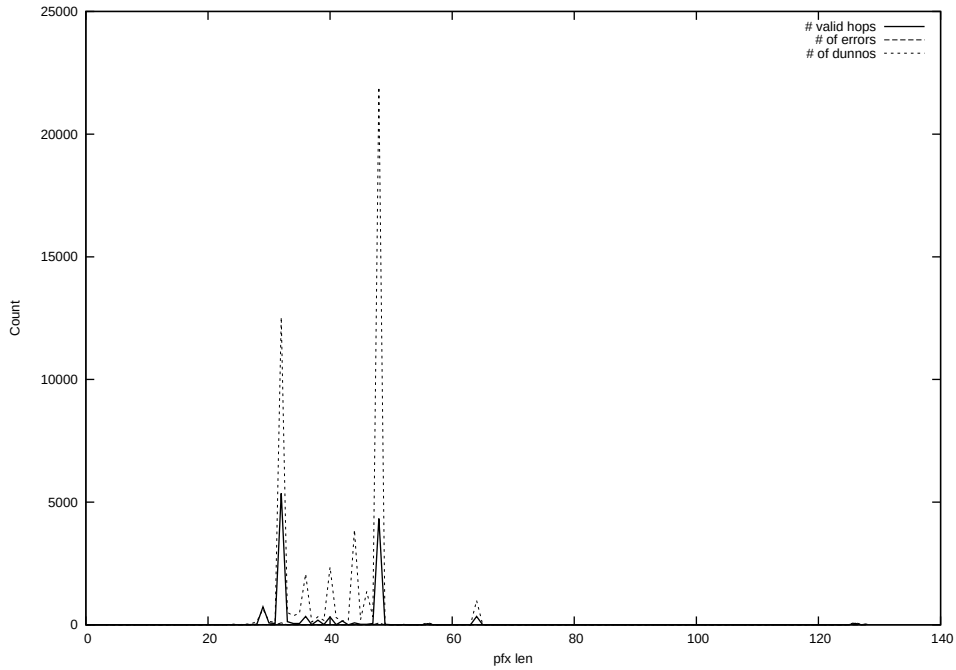
2014-09-21



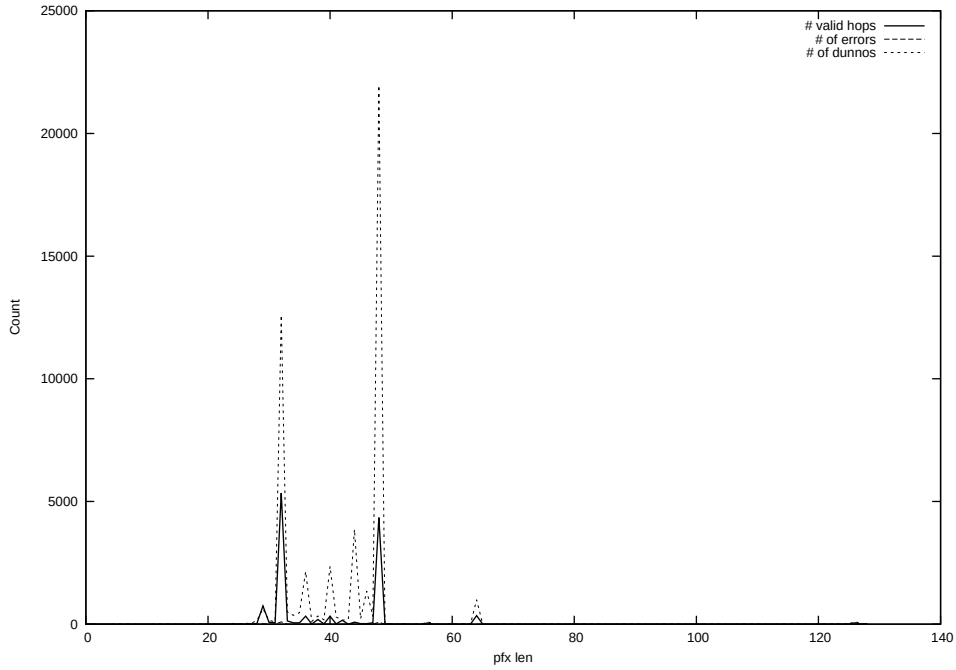
2014-09-22



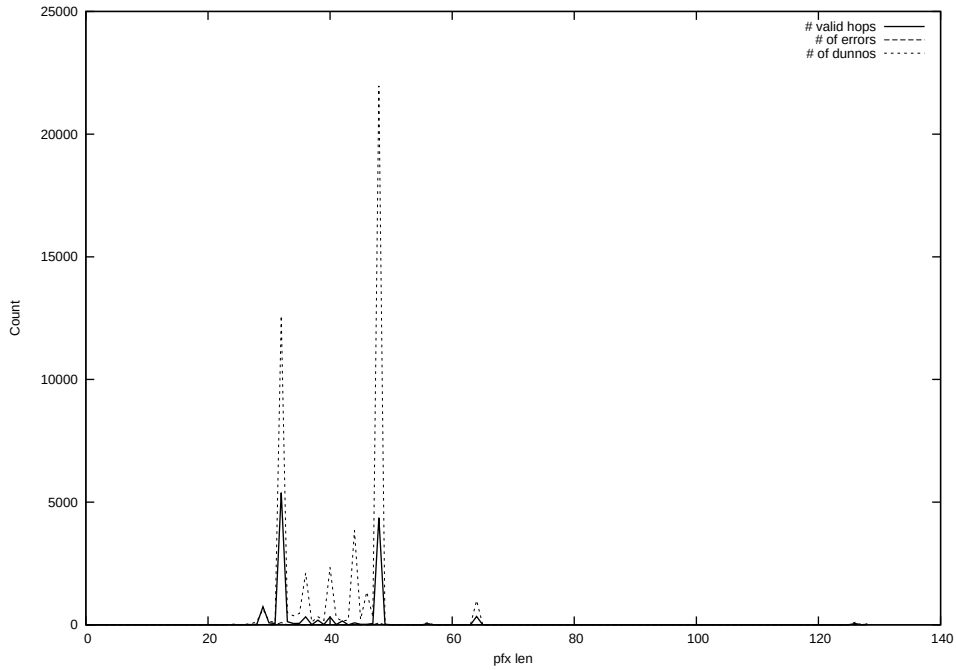
2014-09-23



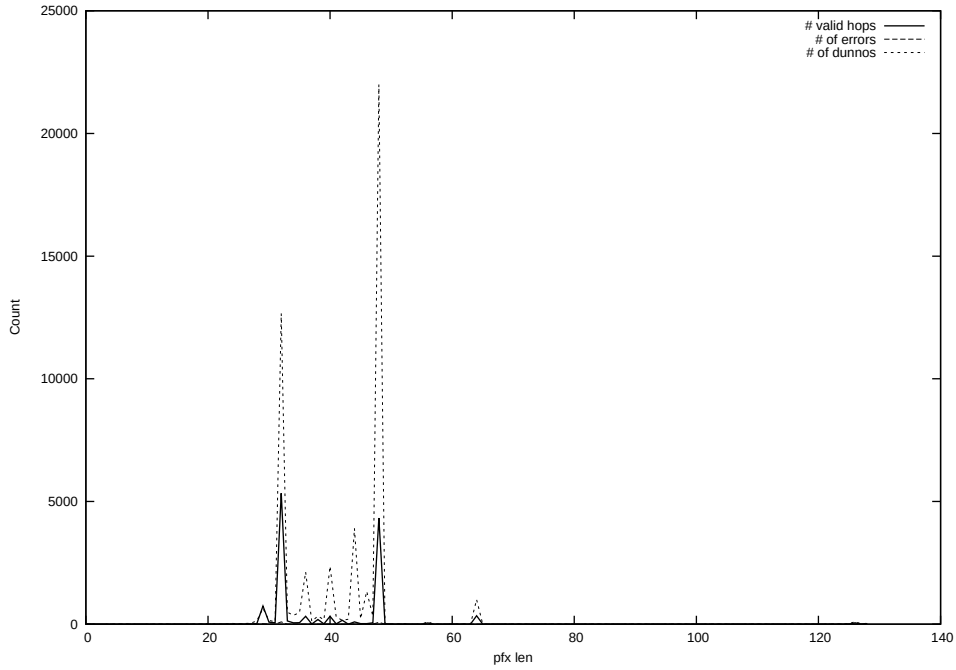
2014-09-24



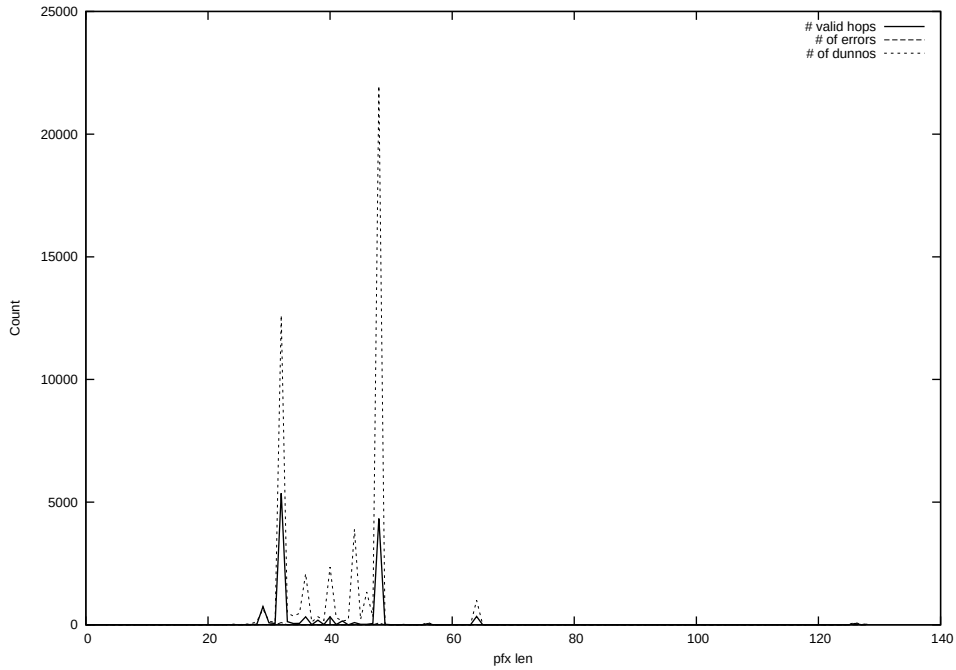
2014-09-25



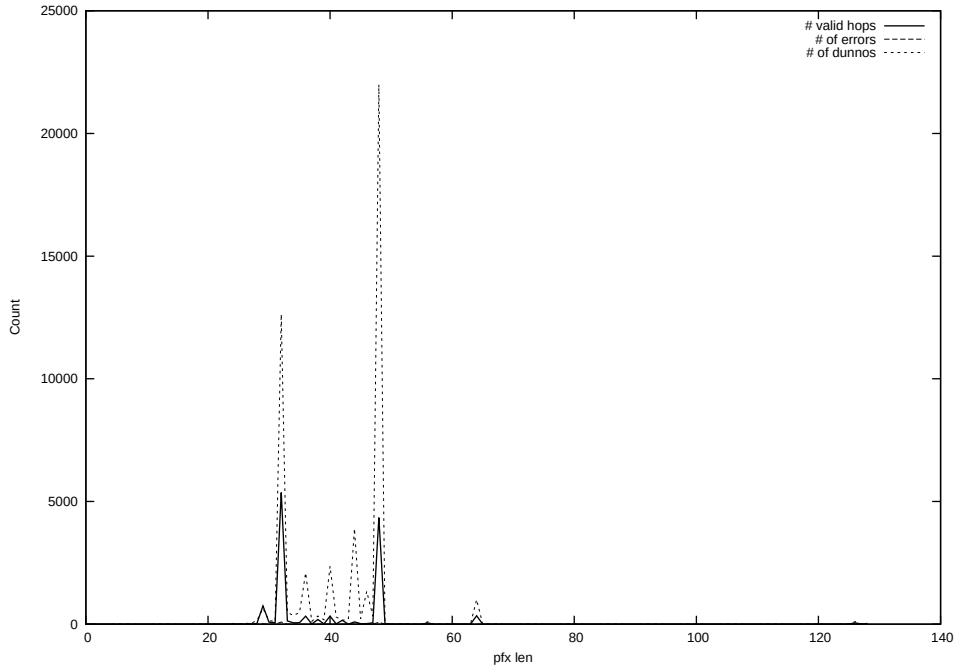
2014-09-26



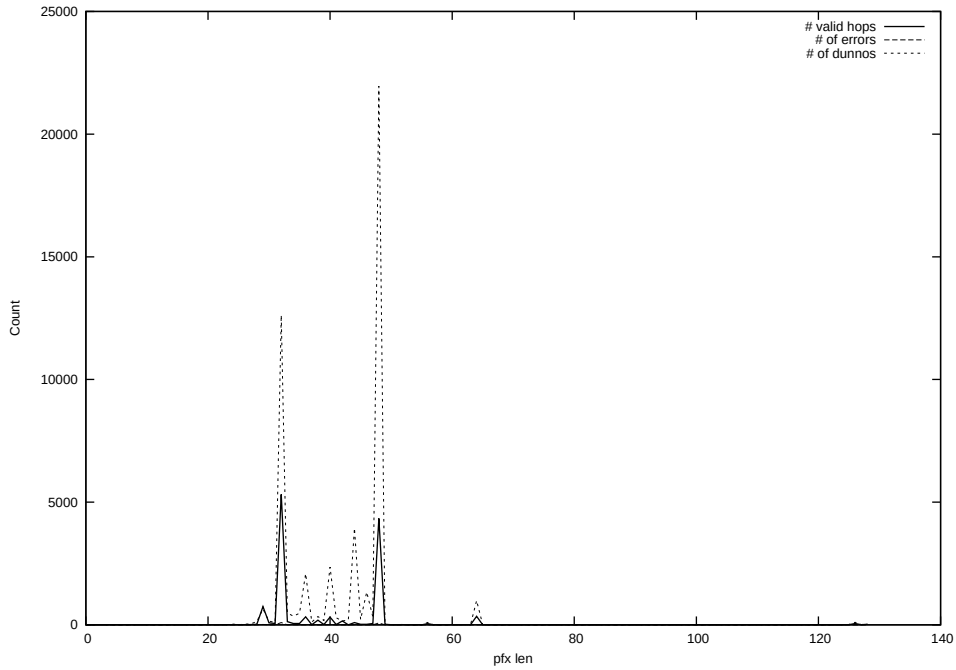
2014-09-27



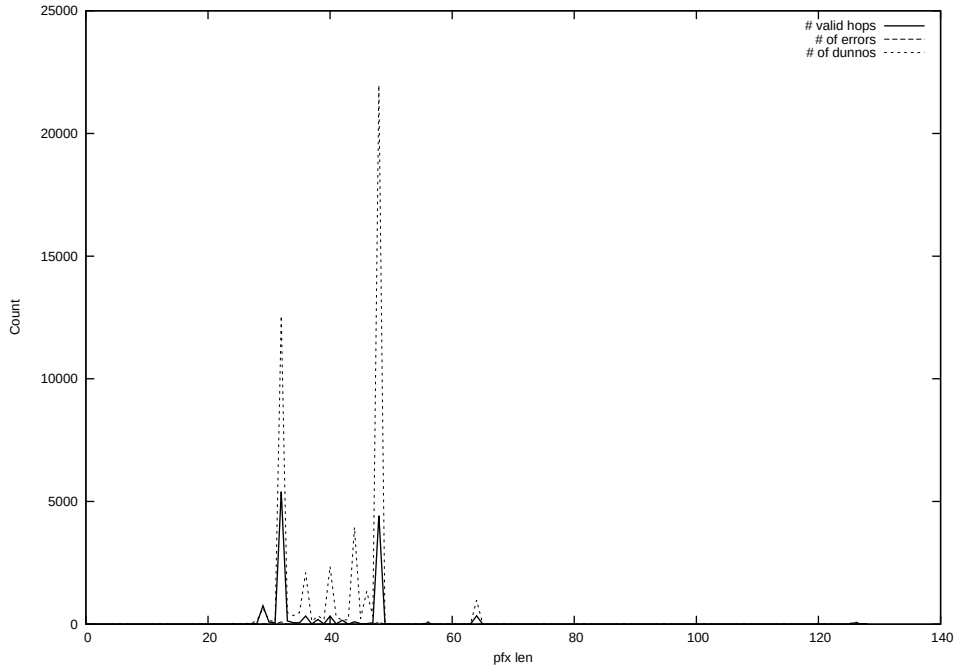
2014-09-28



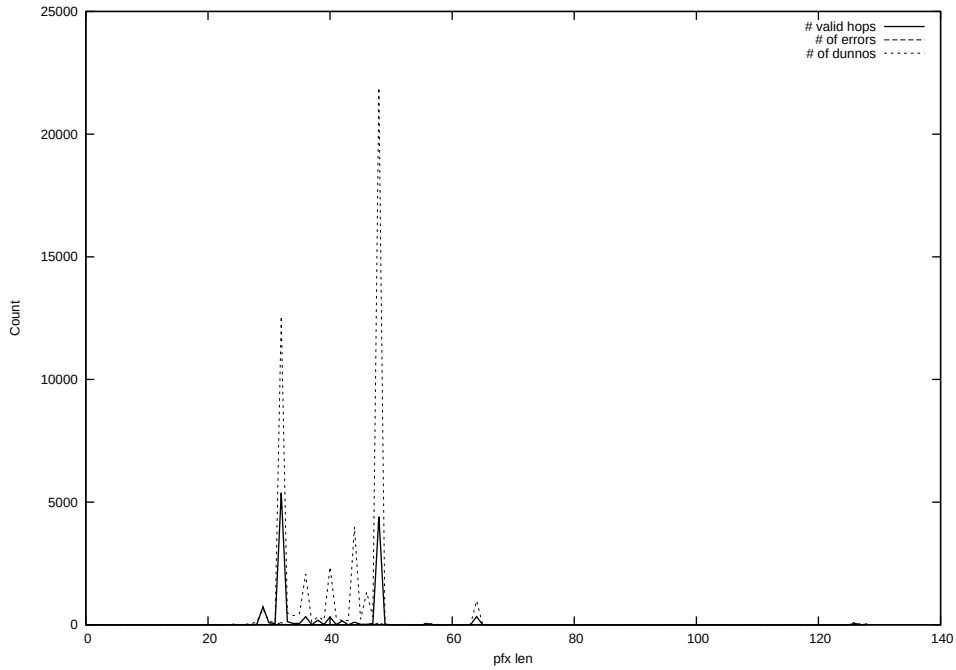
2014-09-29



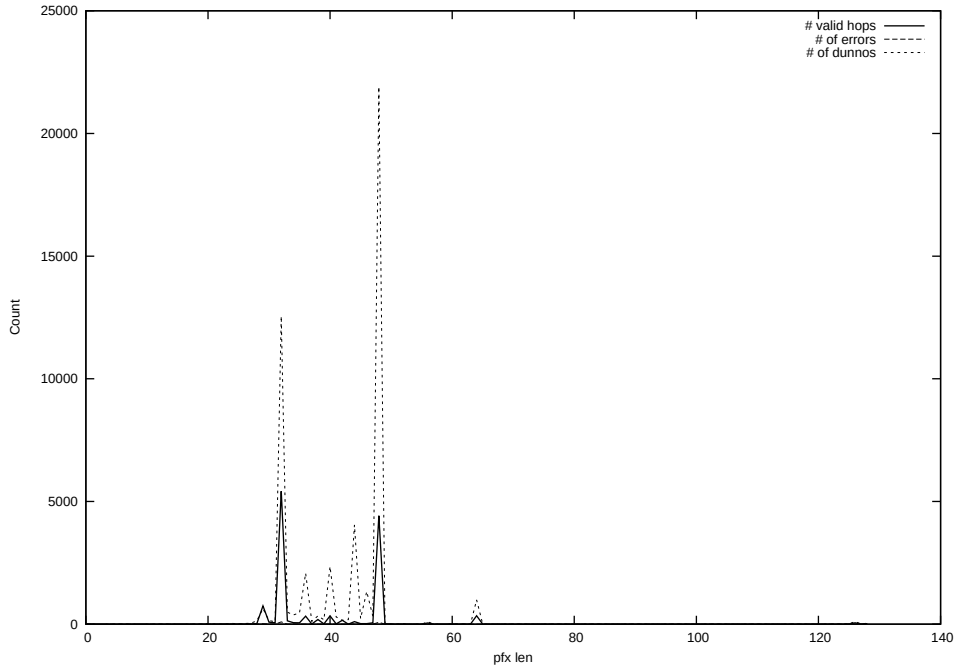
2014-09-30



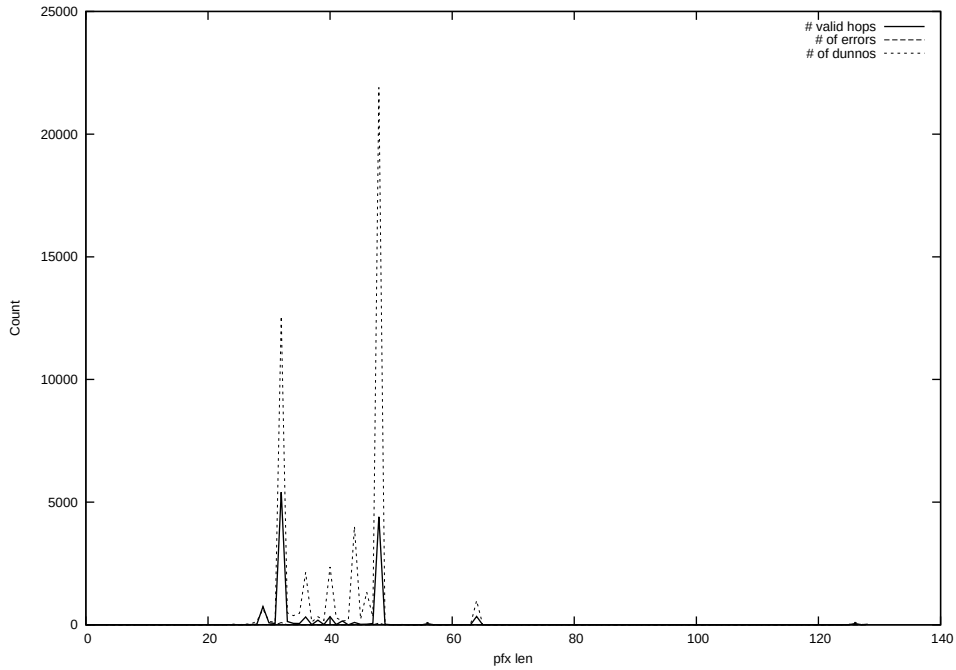
2014-10-01



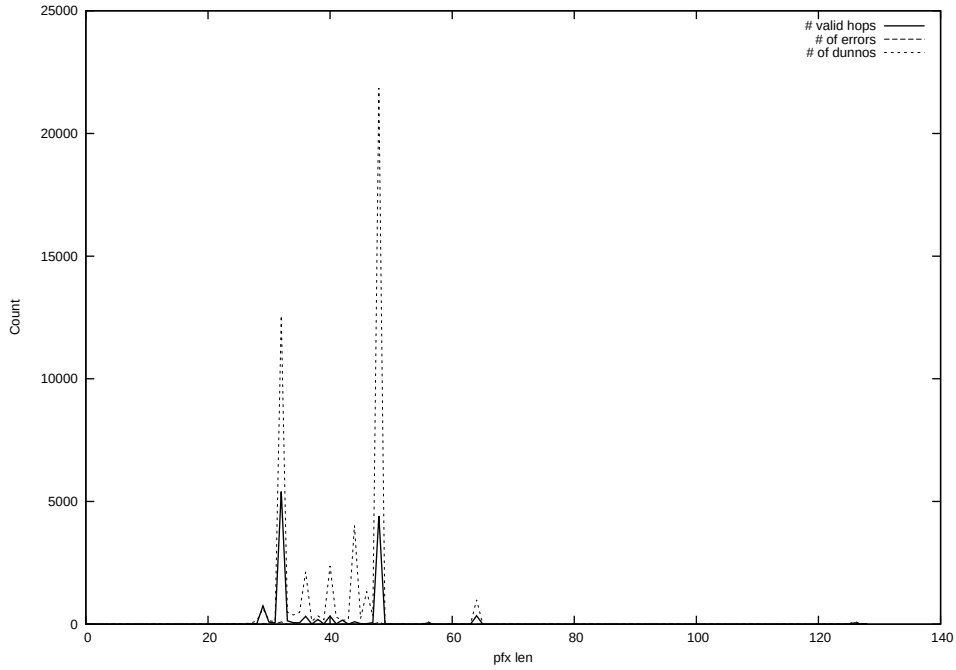
2014-10-02



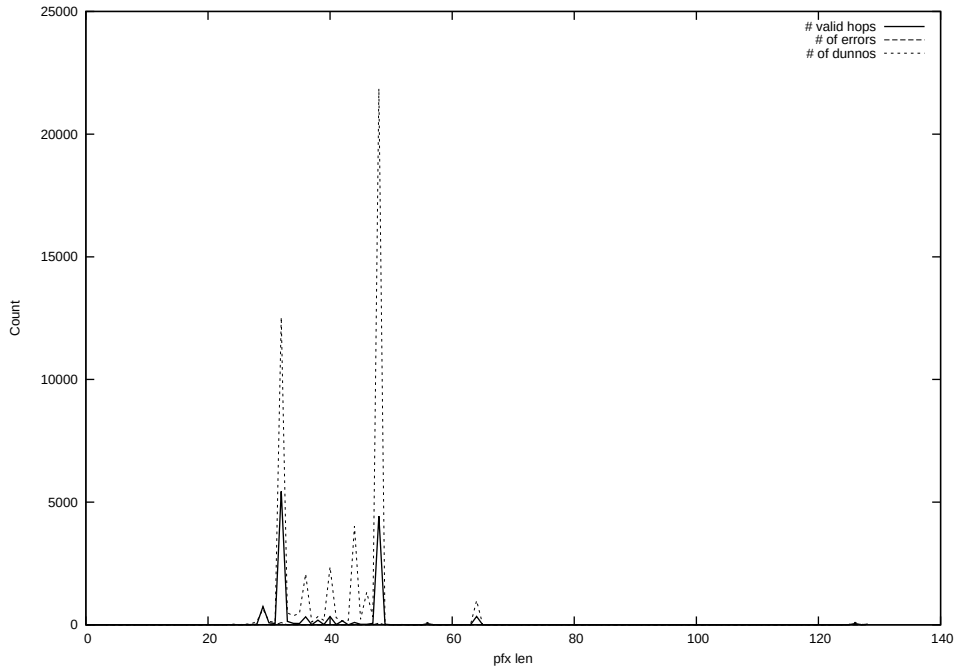
2014-10-03



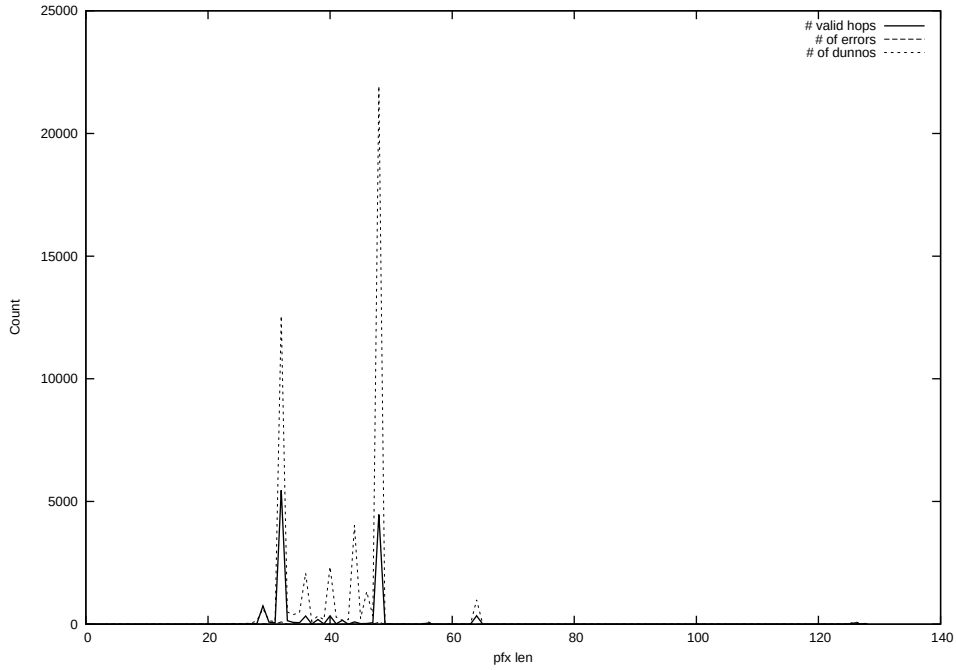
2014-10-04



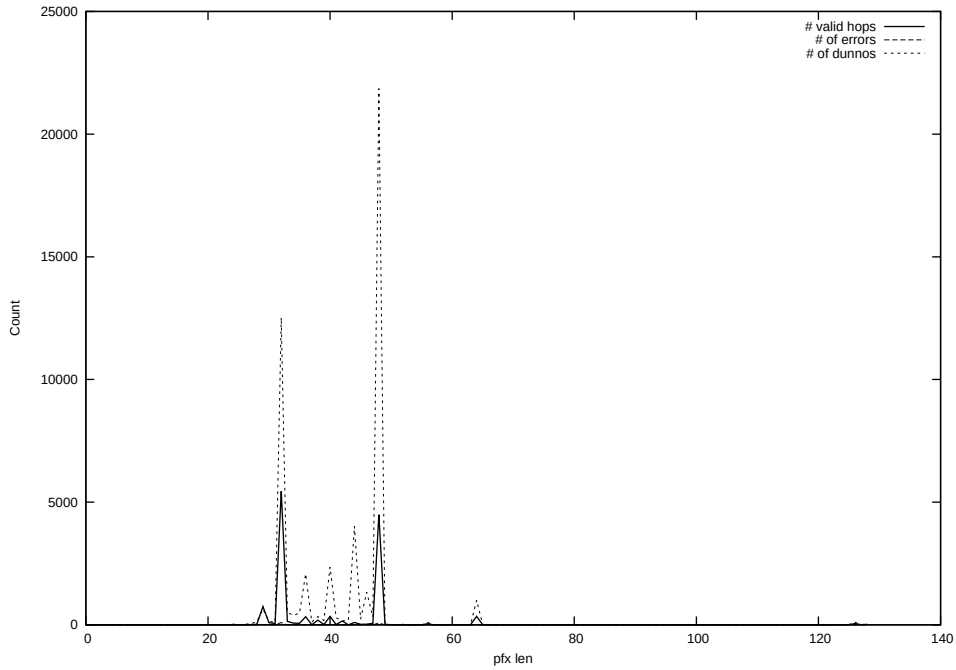
2014-10-05



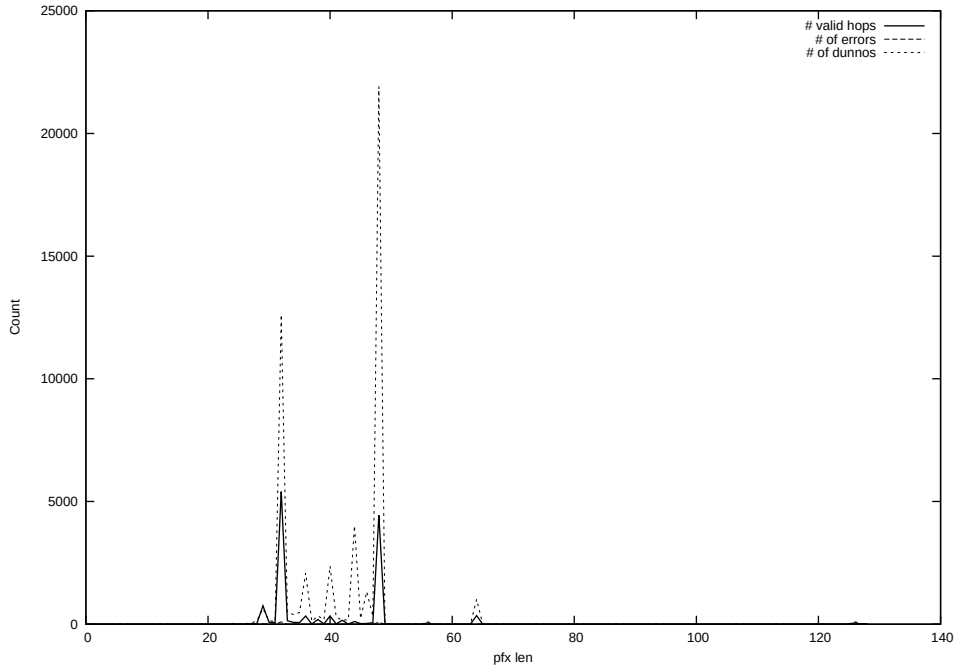
2014-10-06



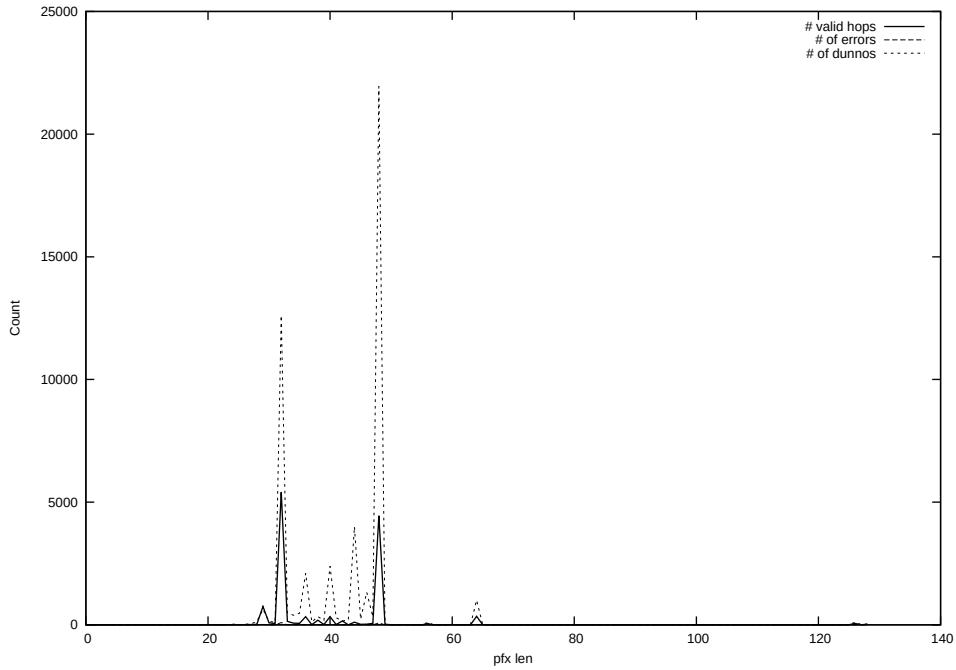
2014-10-07



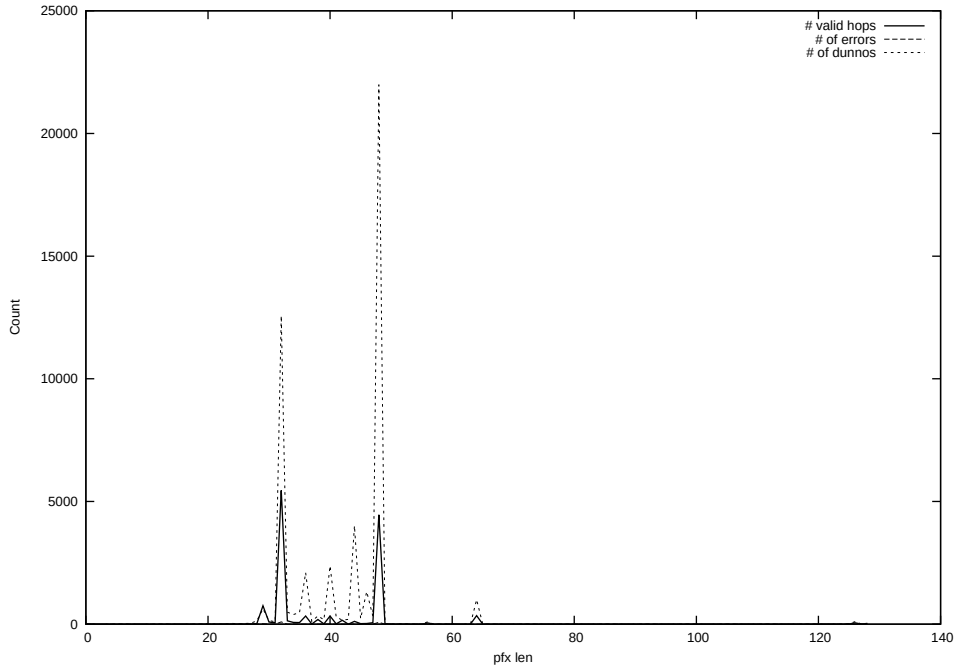
2014-10-08



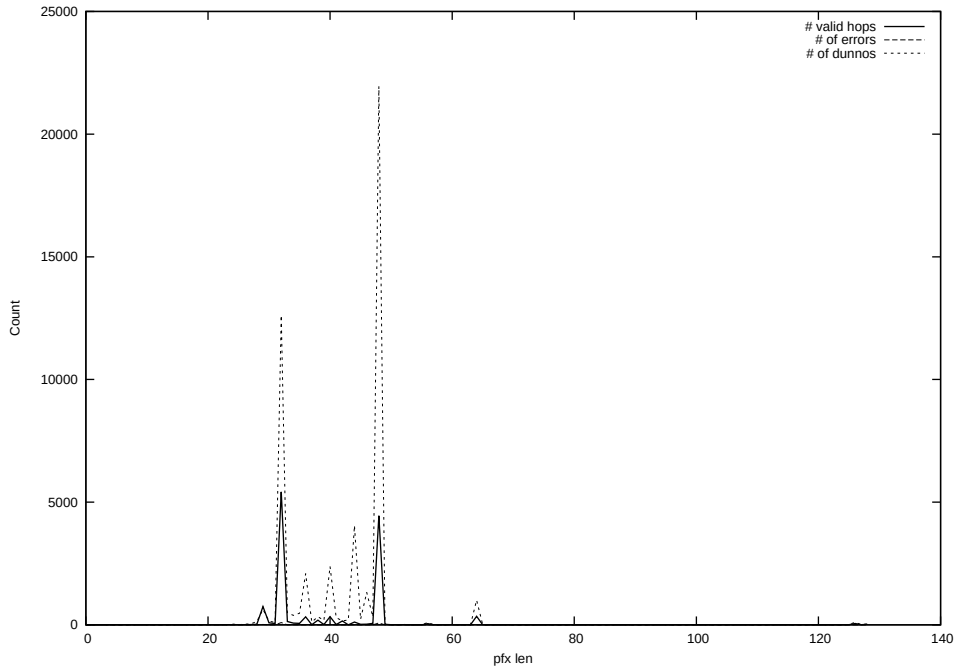
2014-10-09



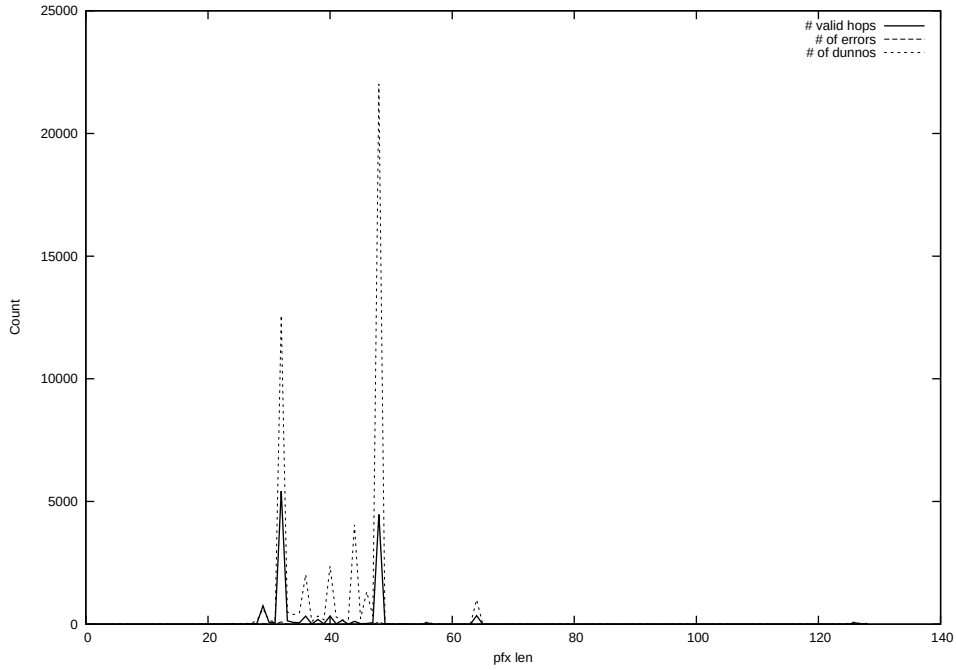
2014-10-10



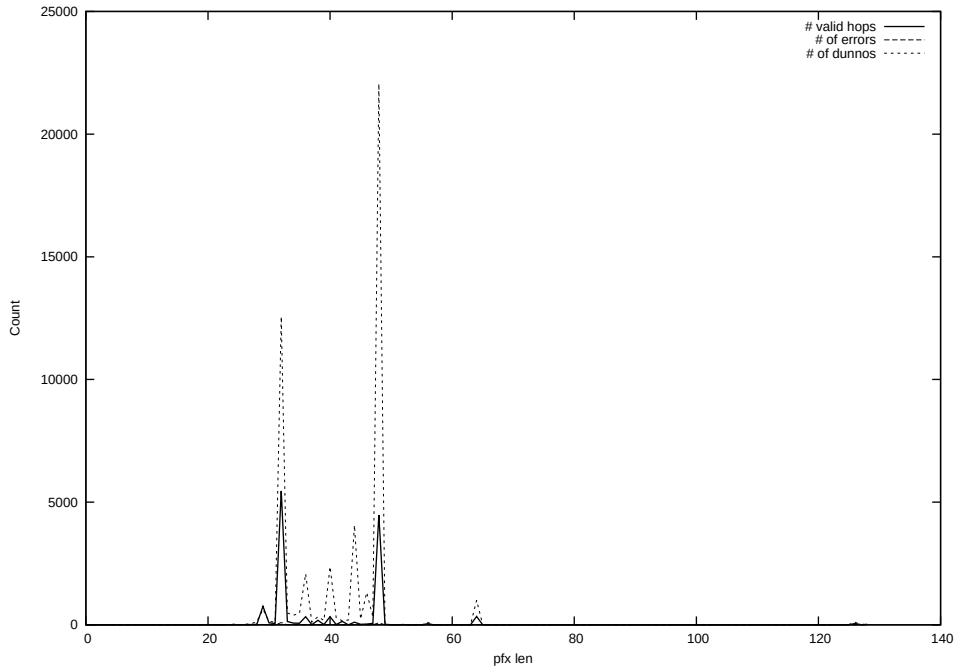
2014-10-11



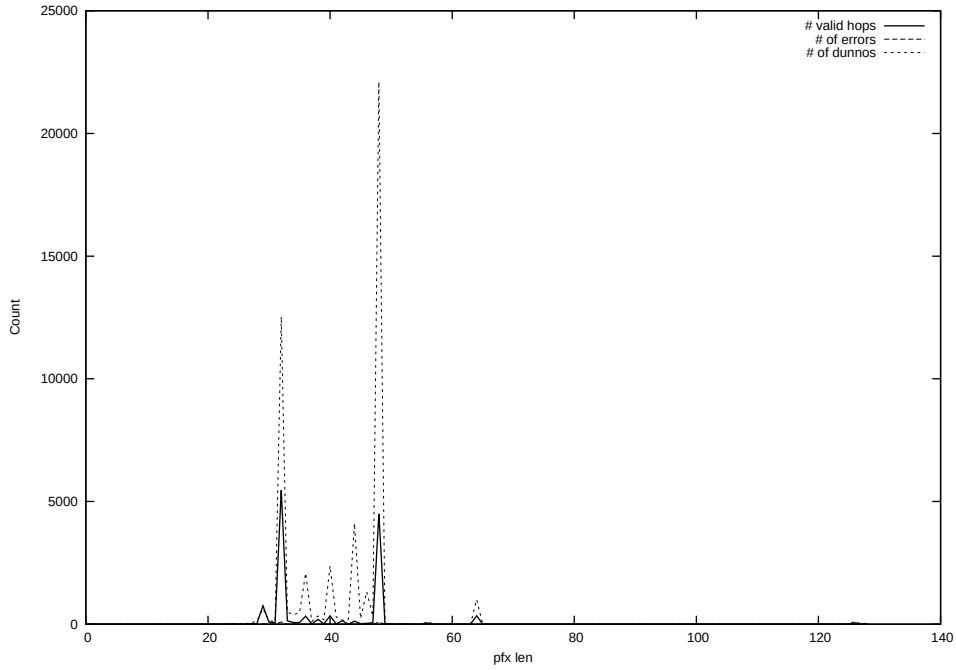
2014-10-12



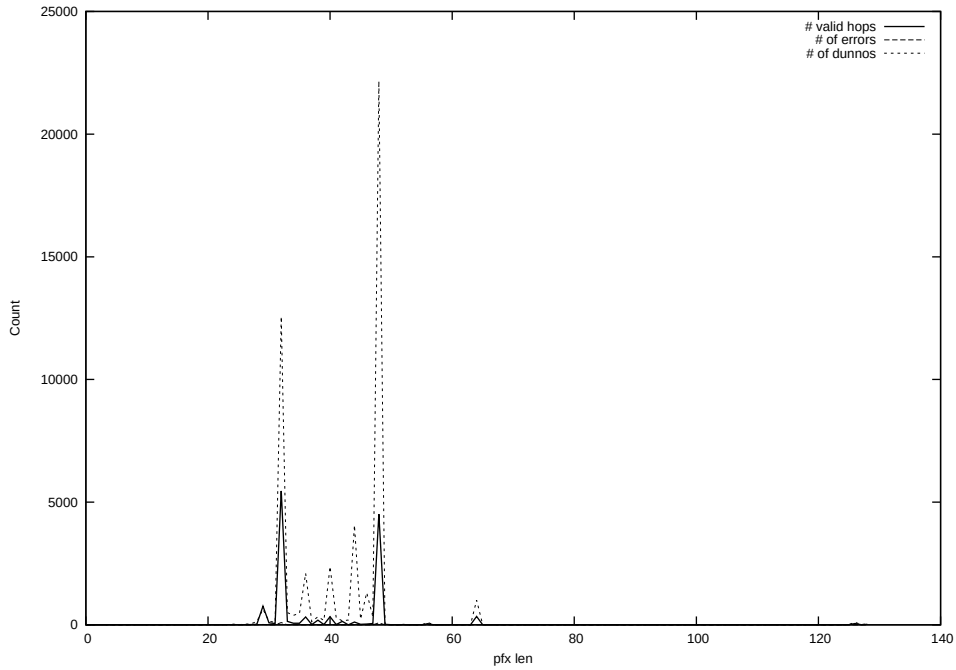
2014-10-13



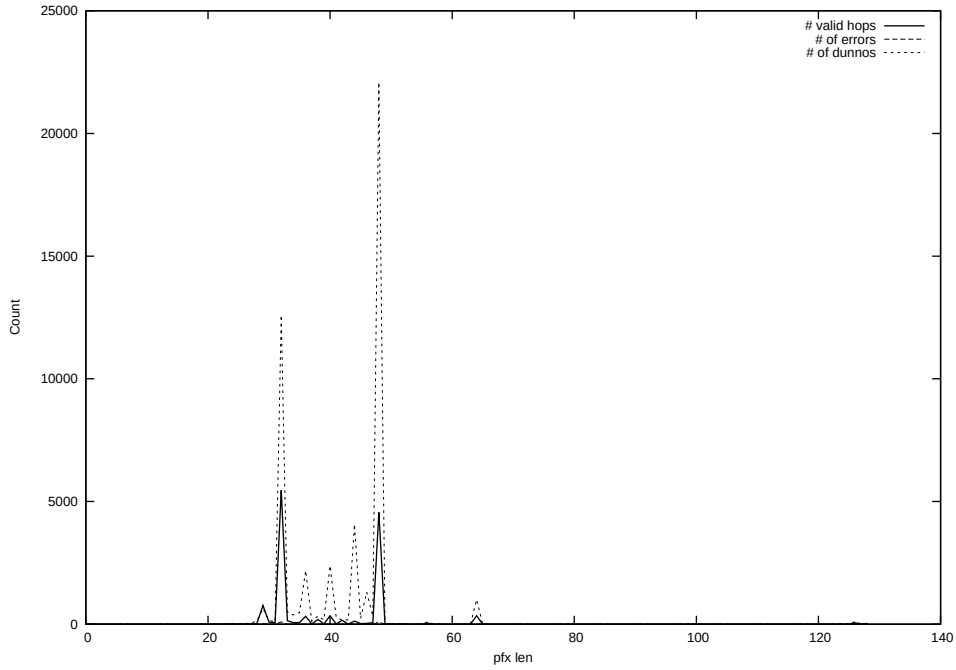
2014-10-14



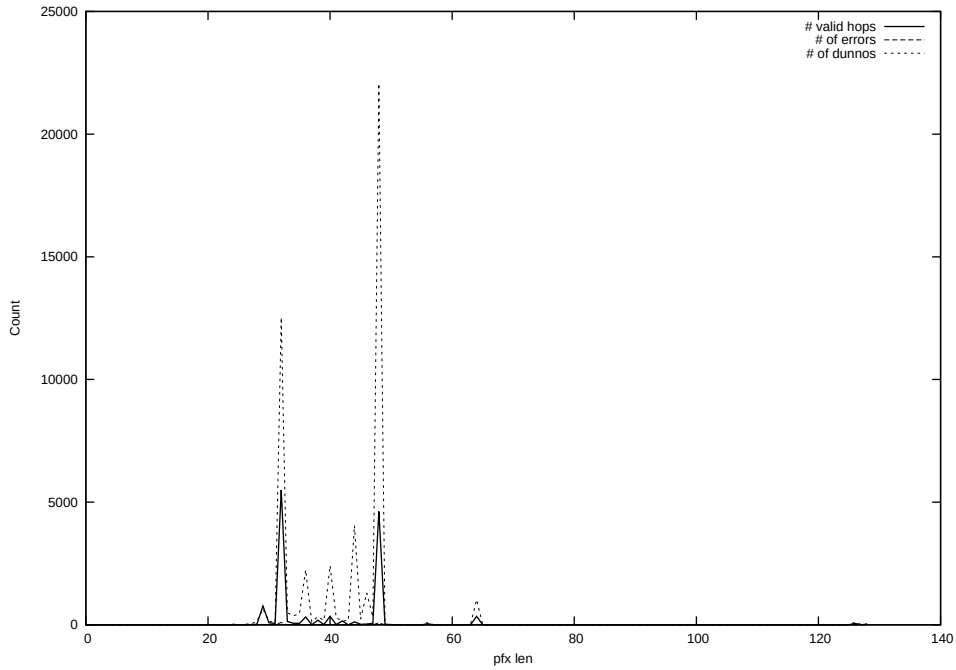
2014-10-15



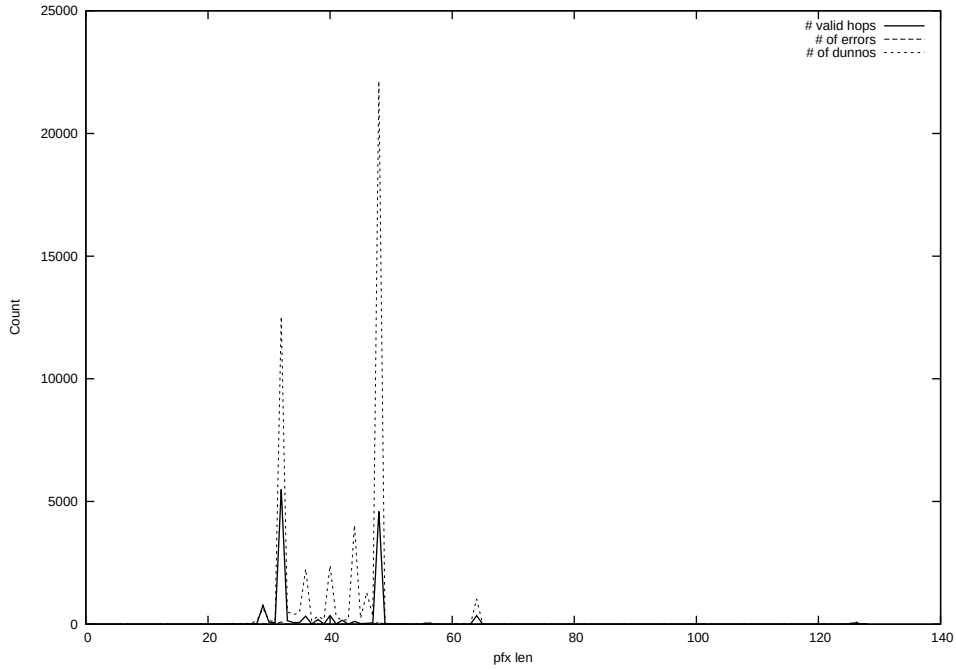
2014-10-16



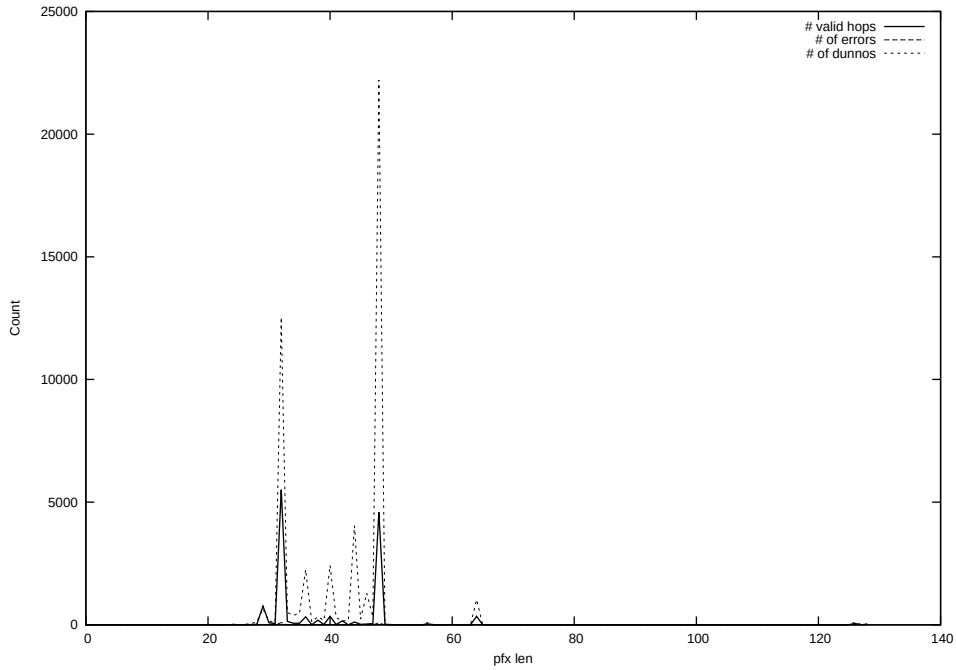
2014-10-17



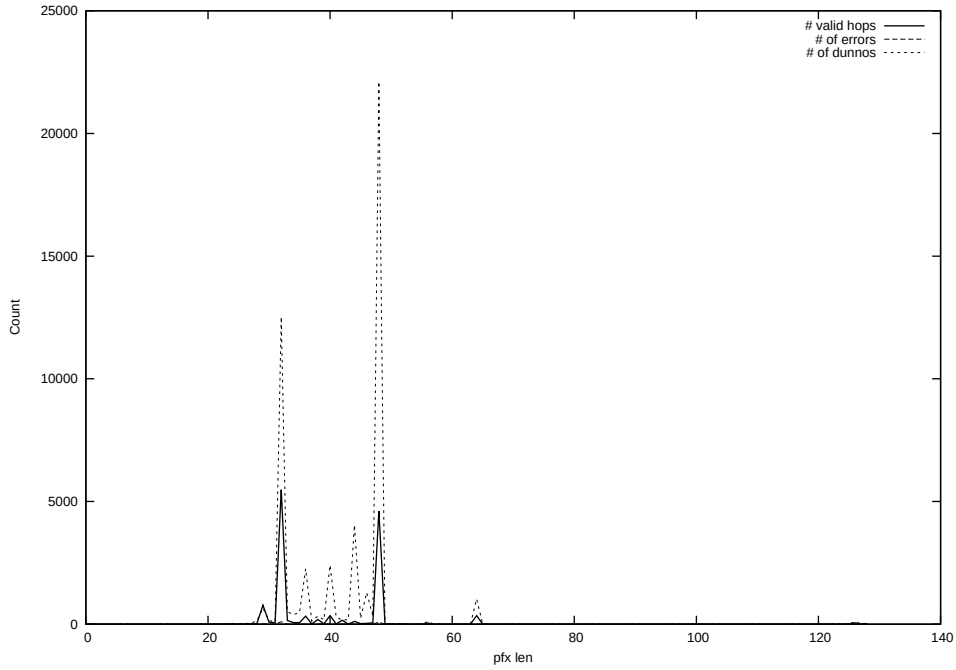
2014-10-18



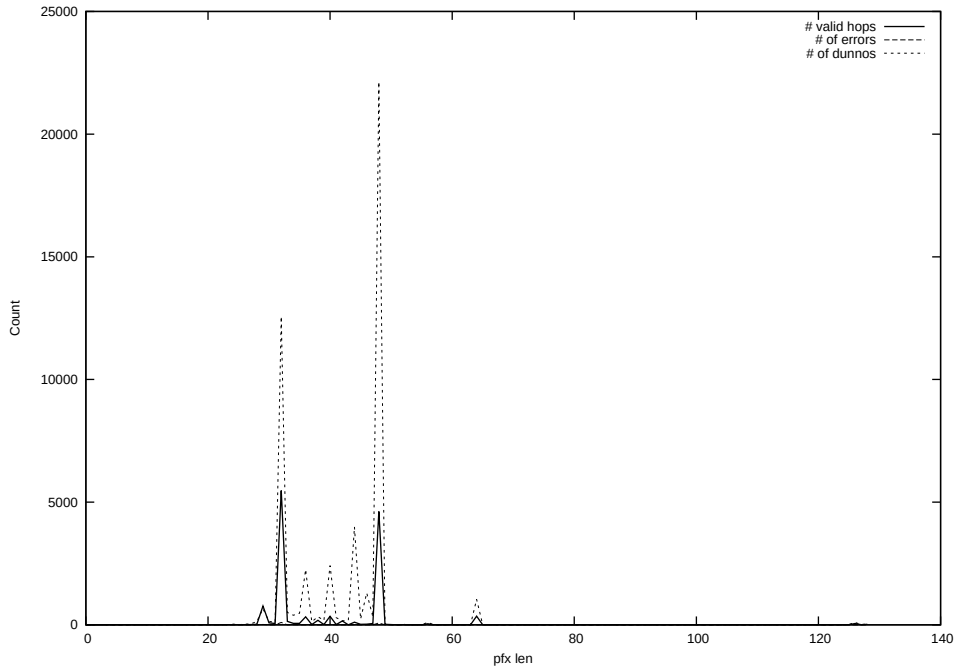
2014-10-19



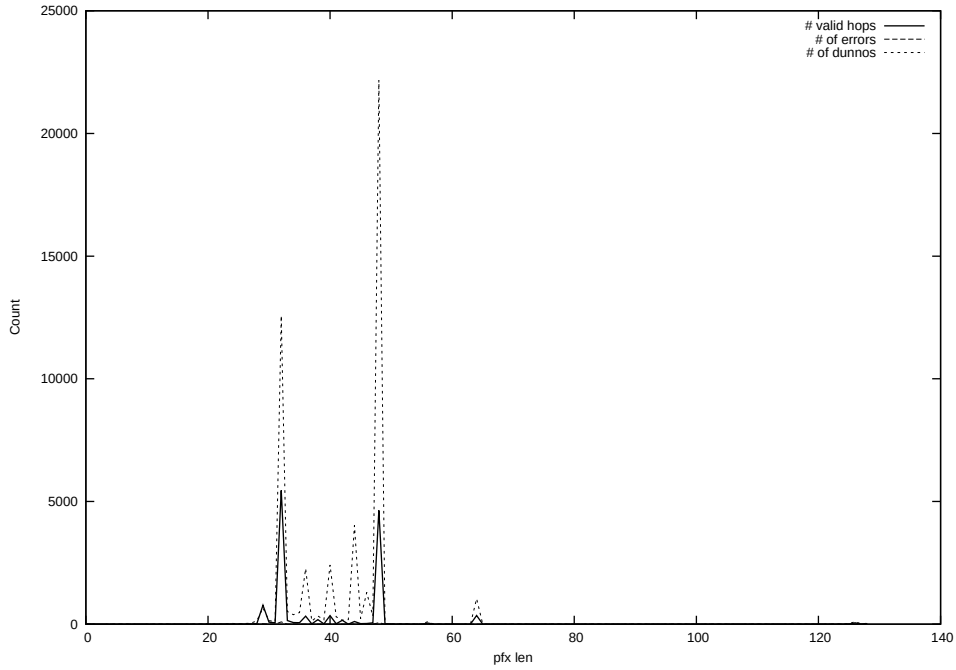
2014-10-20



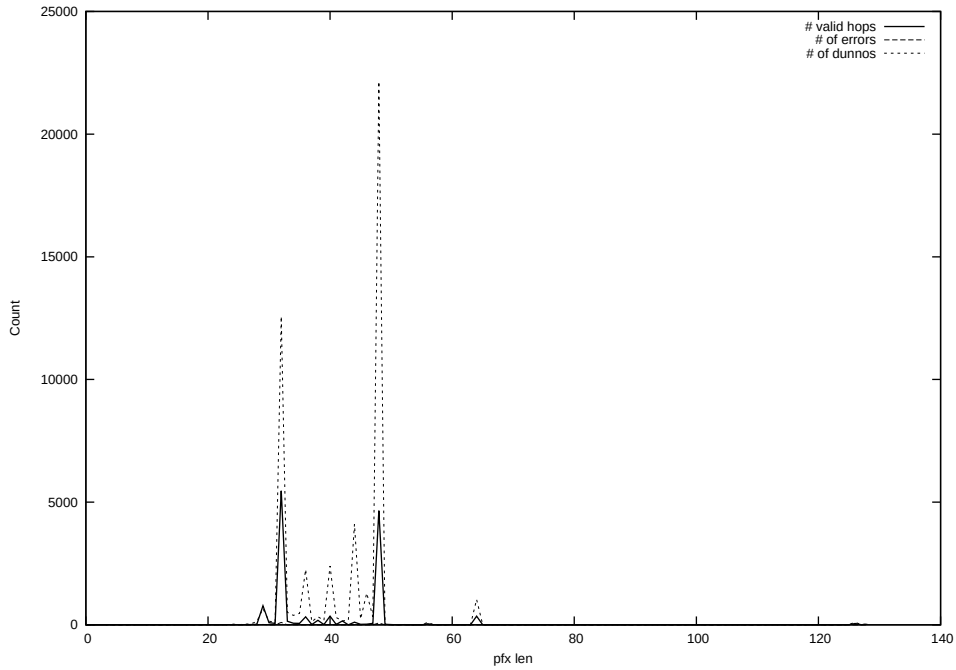
2014-10-21



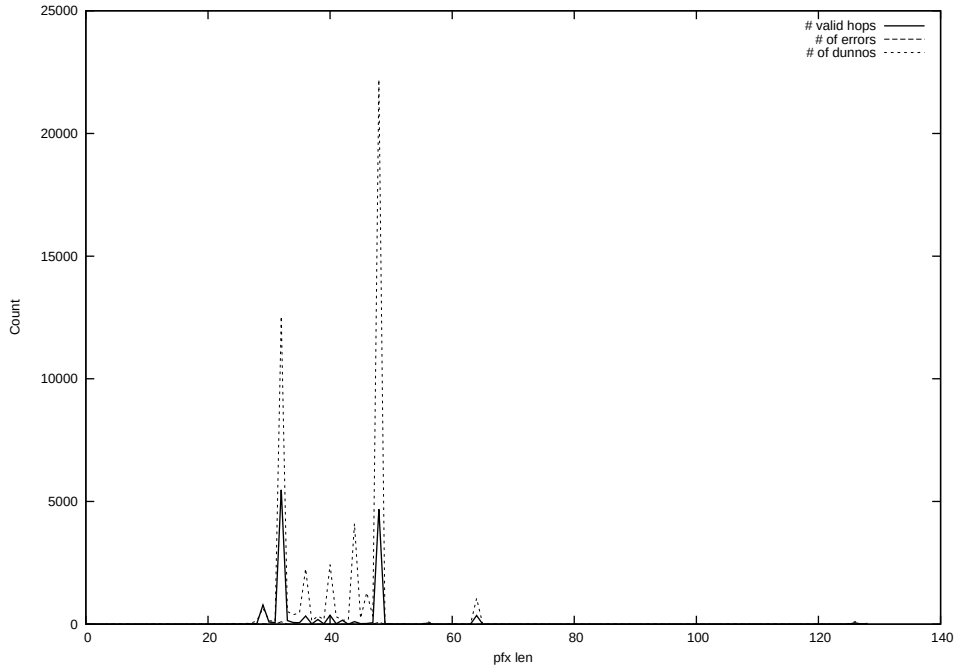
2014-10-22



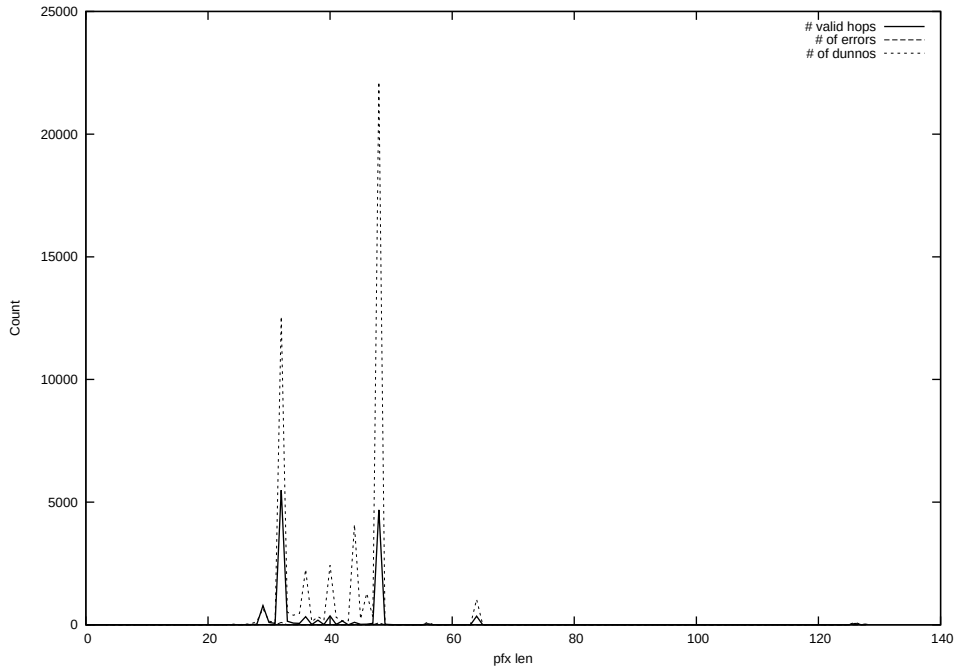
2014-10-23



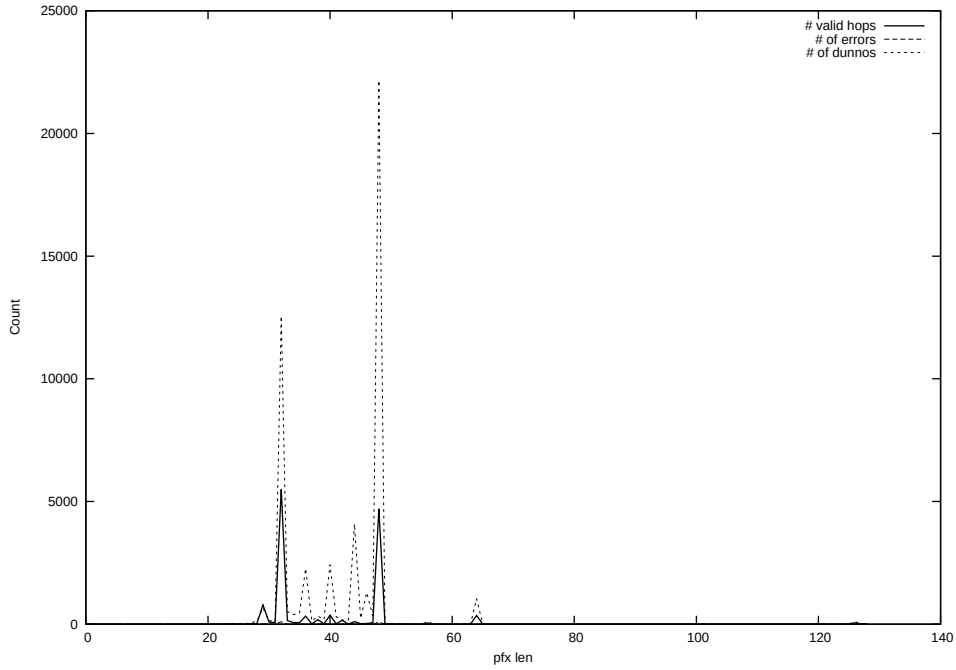
2014-10-24



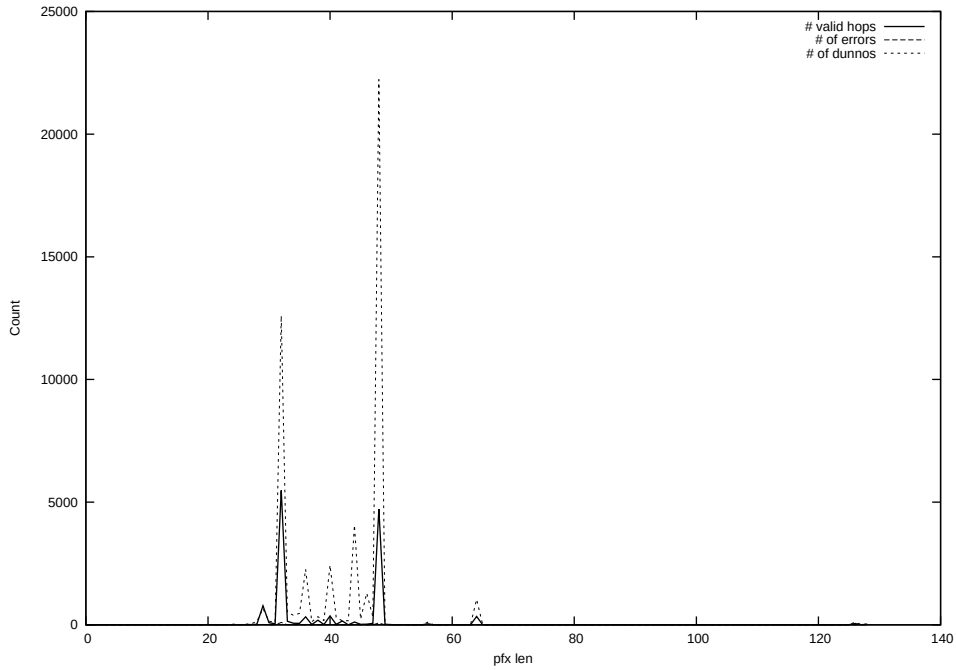
2014-10-25



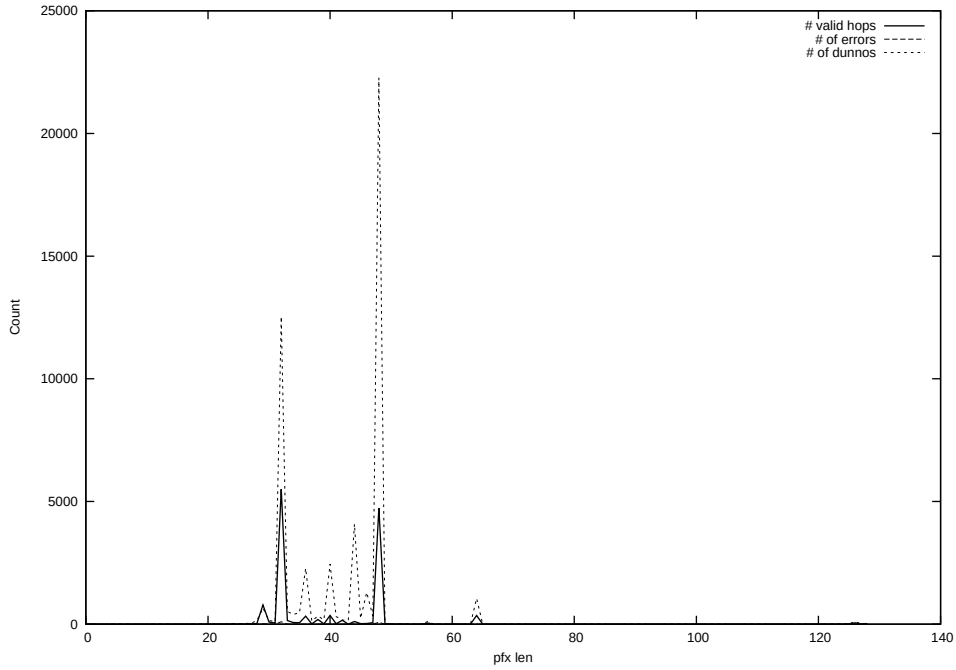
2014-10-26



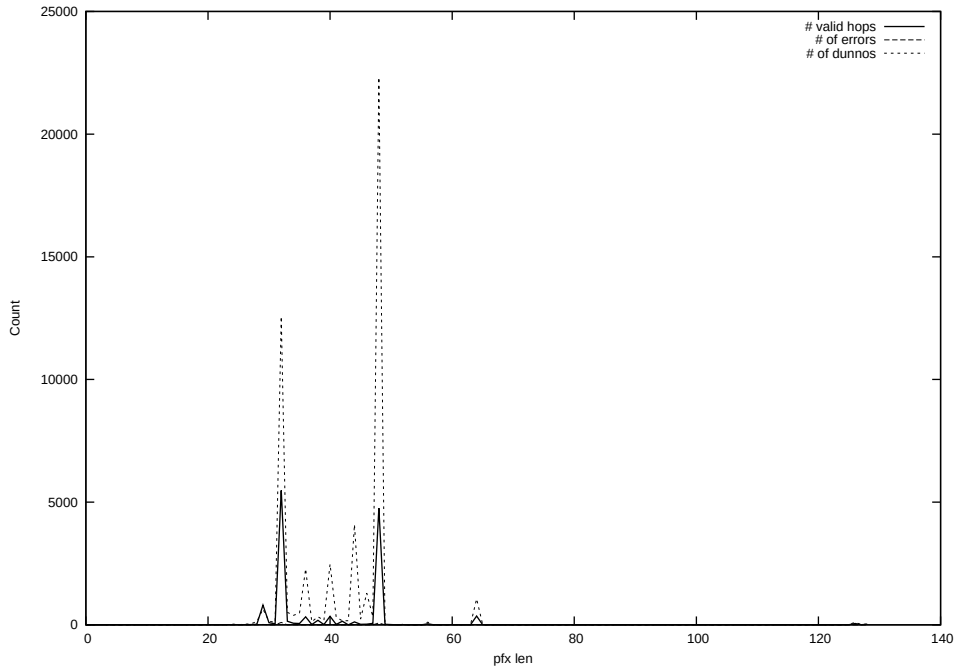
2014-10-27



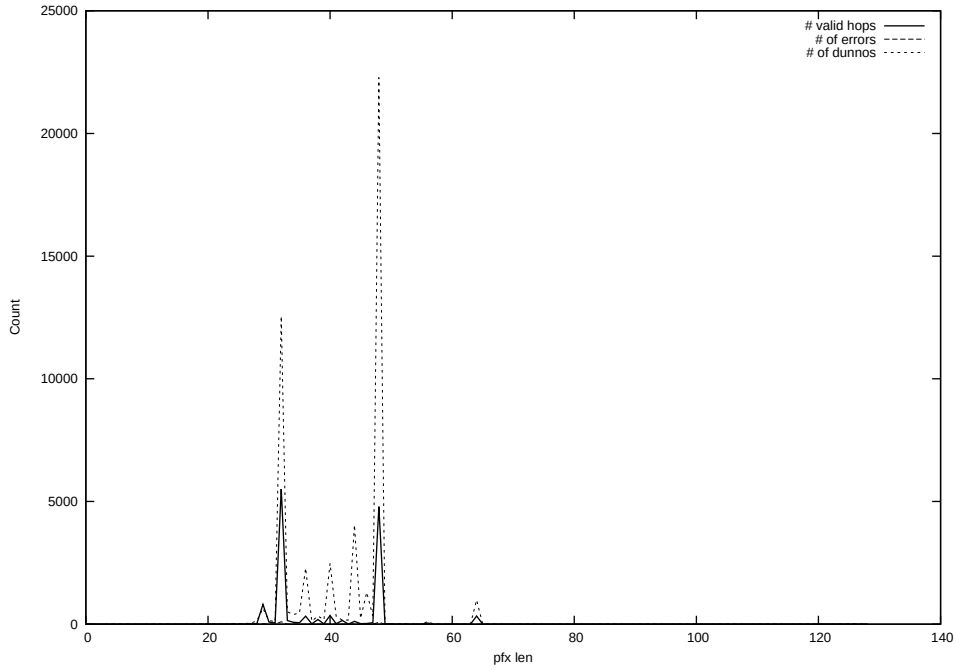
2014-10-28



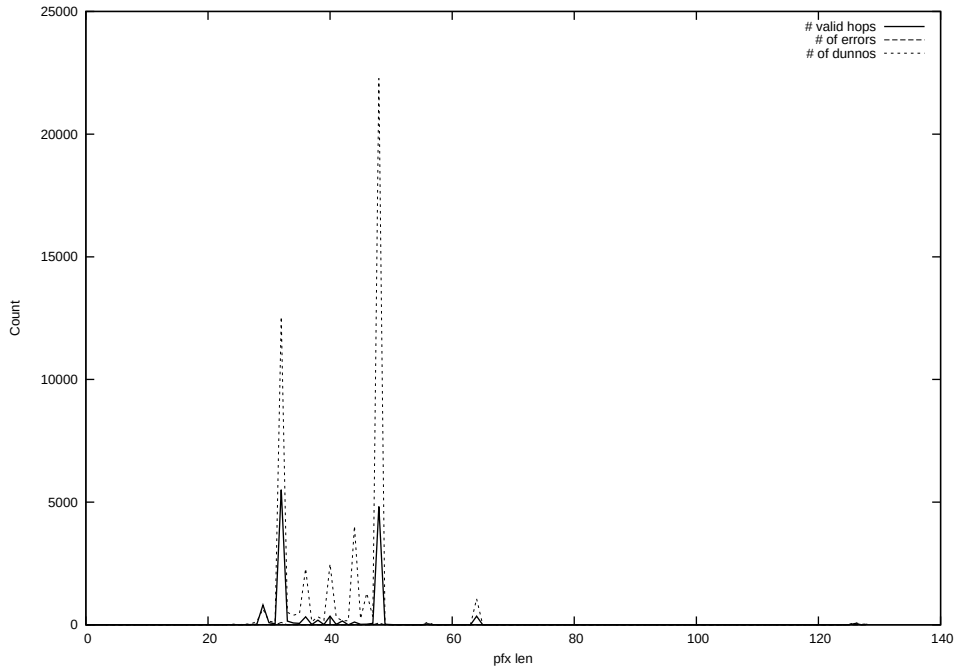
2014-10-29



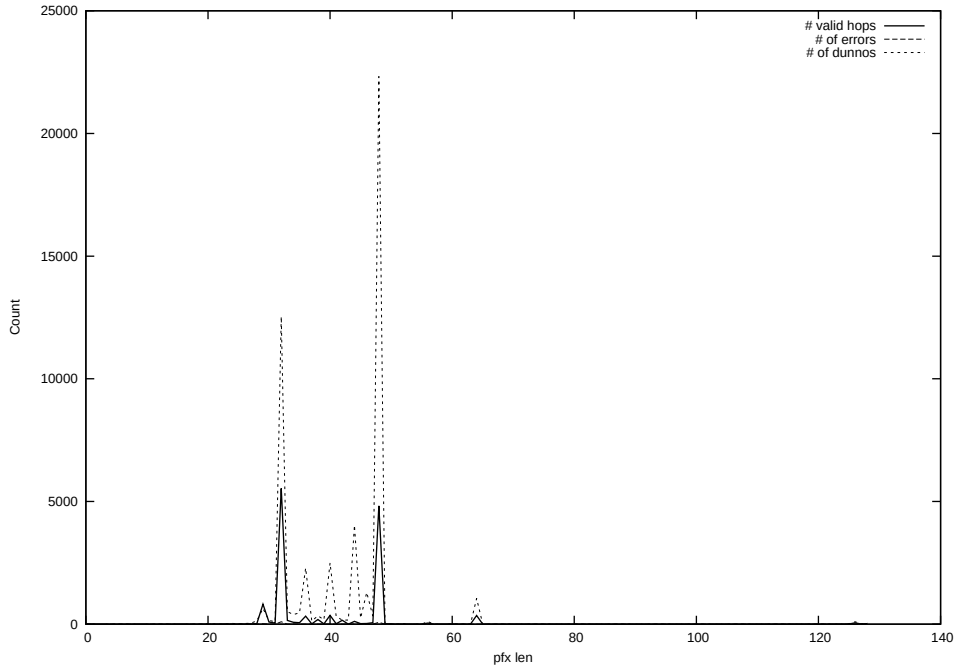
2014-10-30



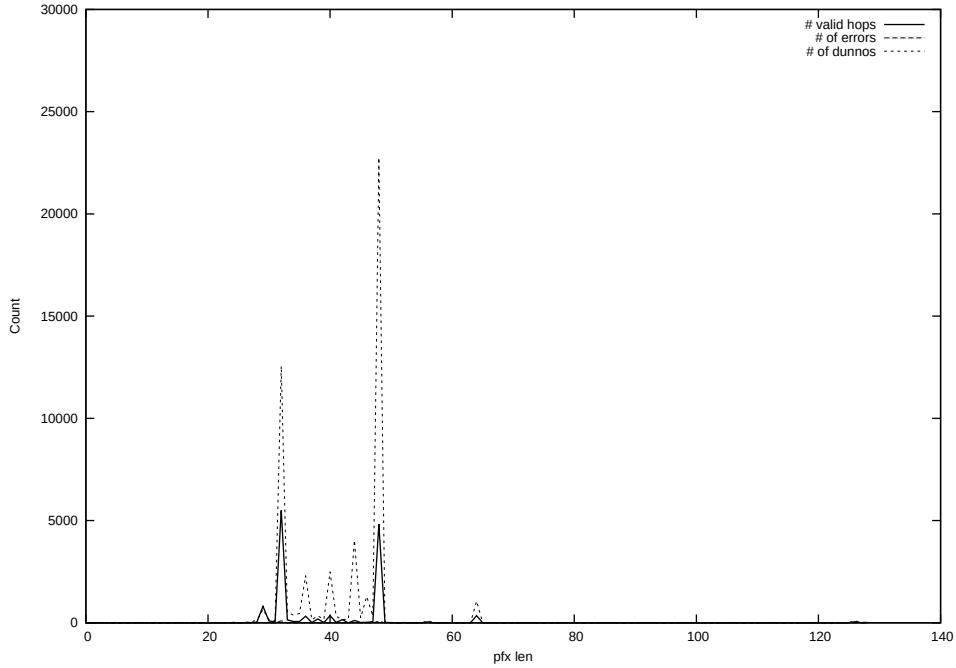
2014-10-31



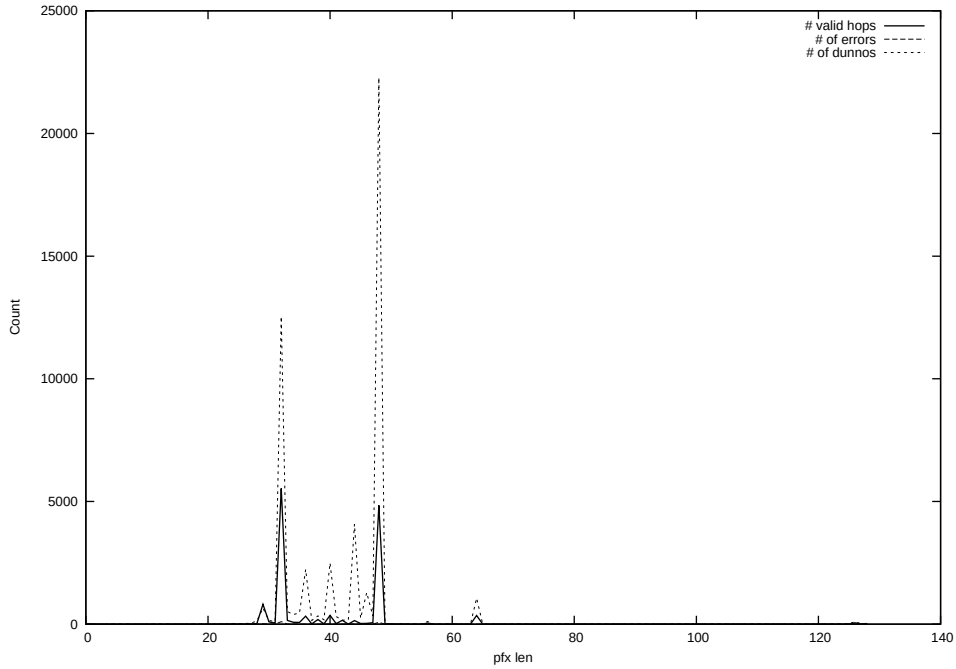
2014-11-02



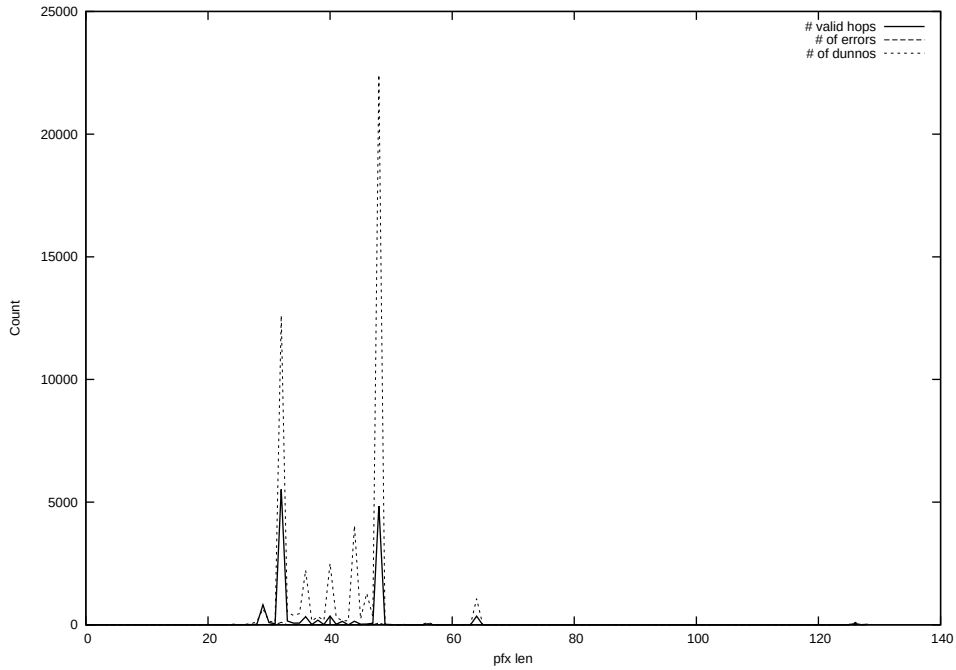
2014-11-03



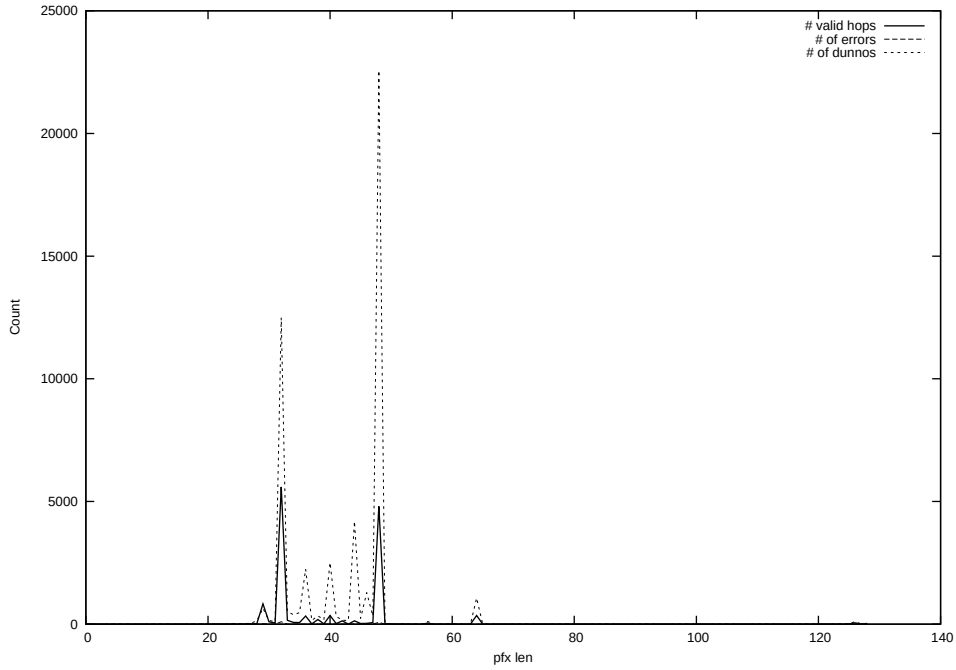
2014-11-04



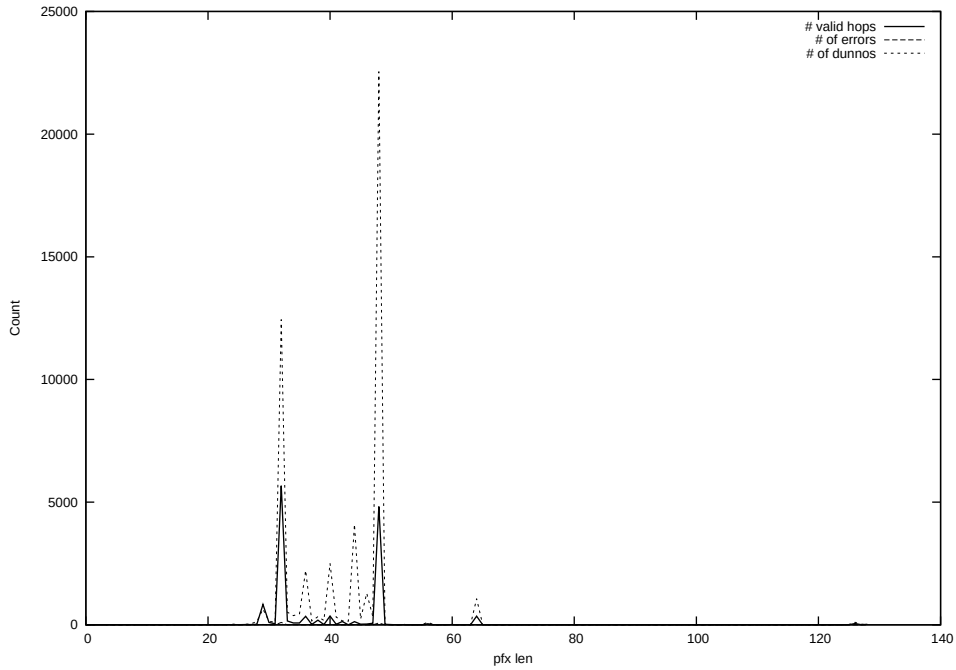
2014-11-05



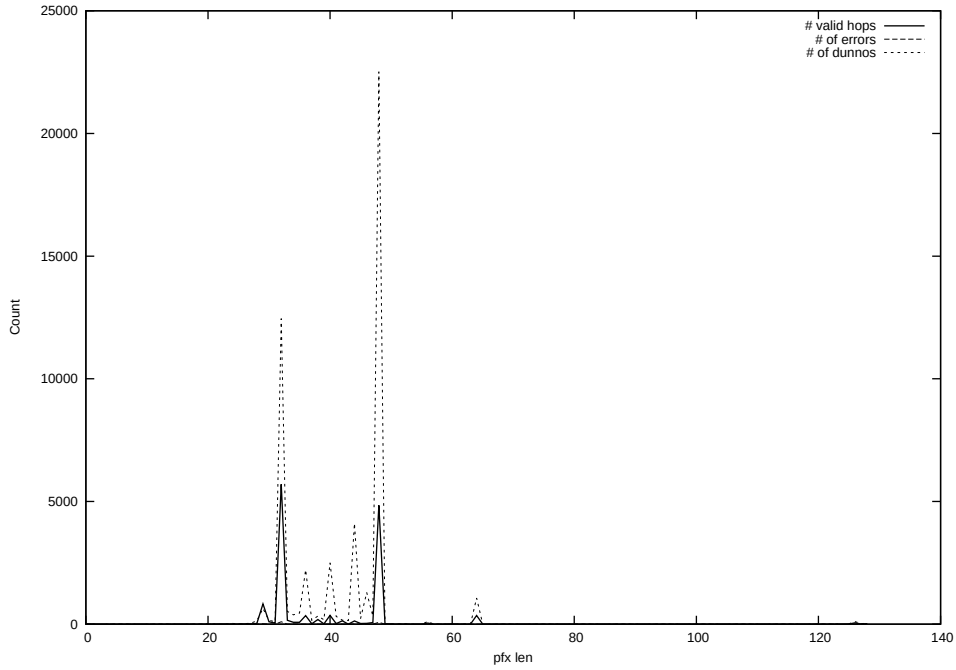
2014-11-06



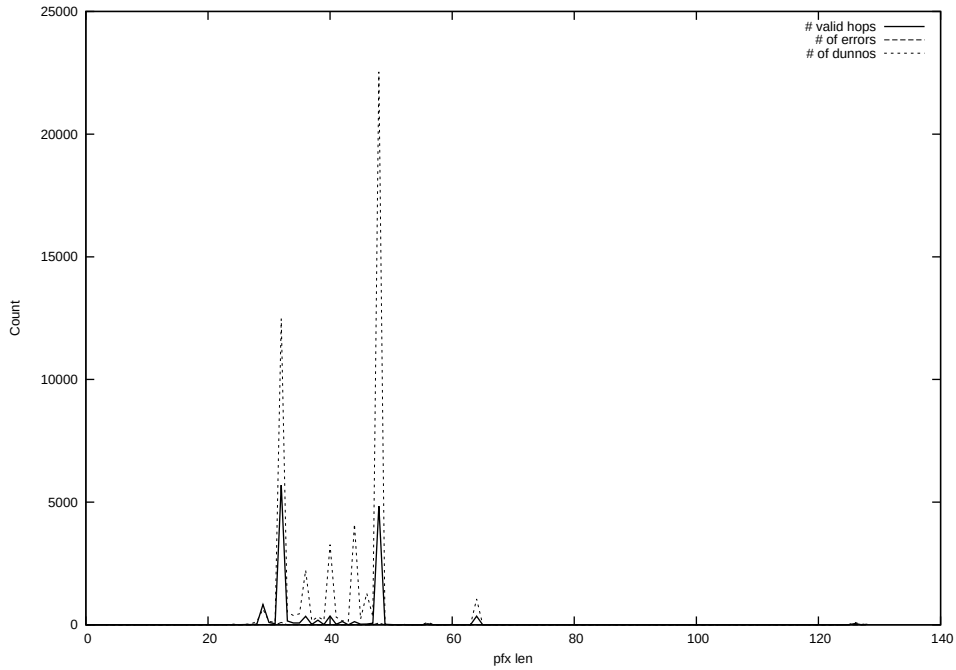
2014-11-07



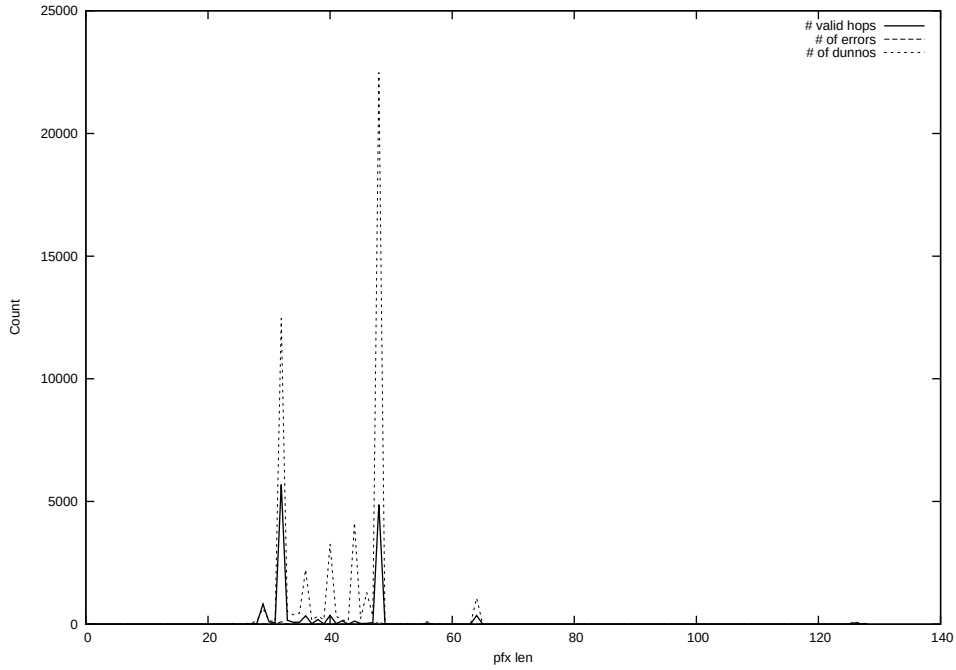
2014-11-08



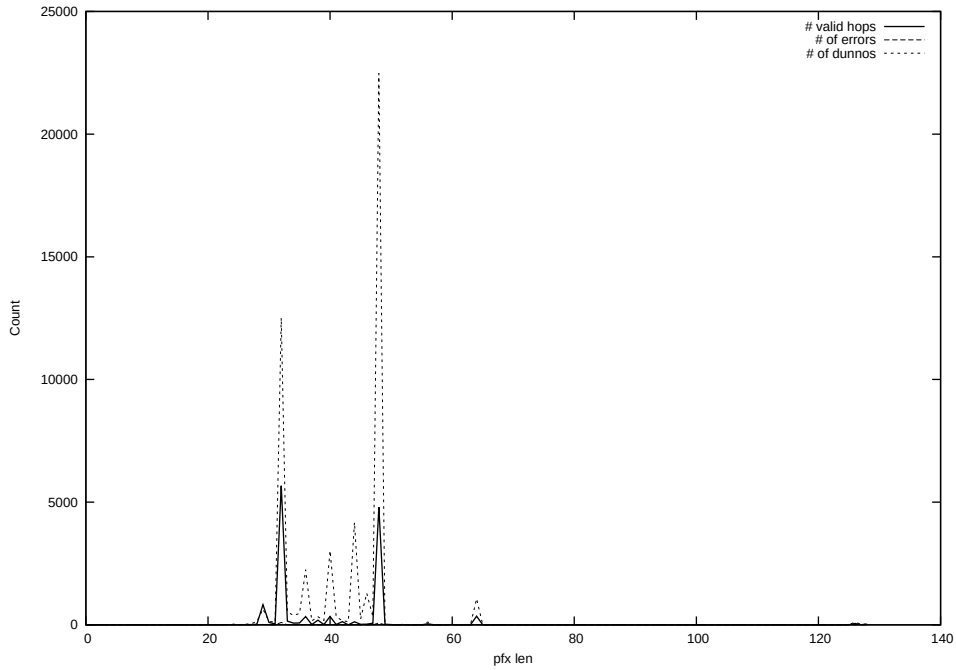
2014-11-09



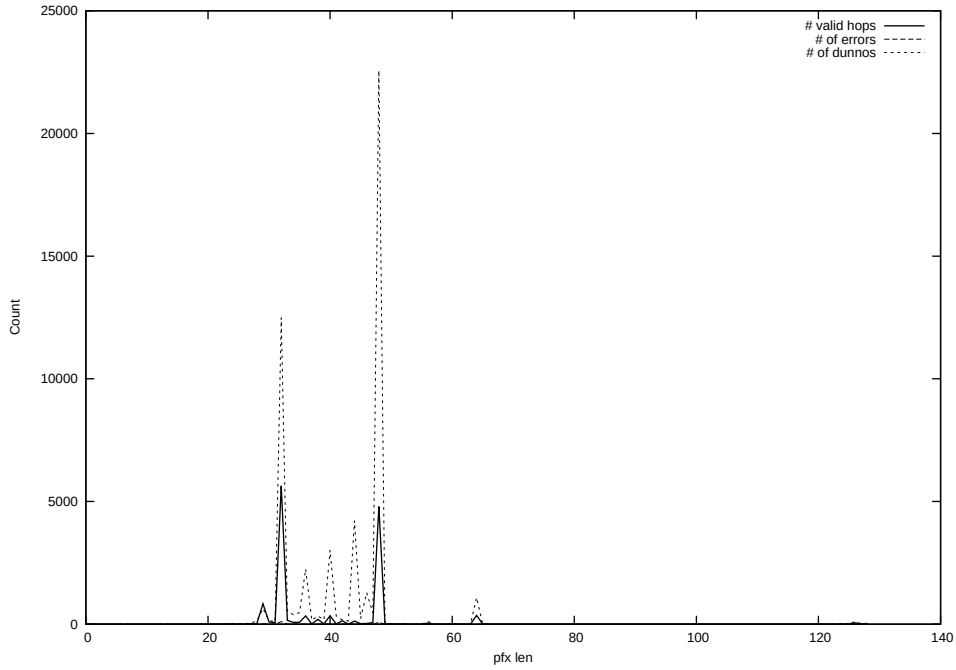
2014-11-10



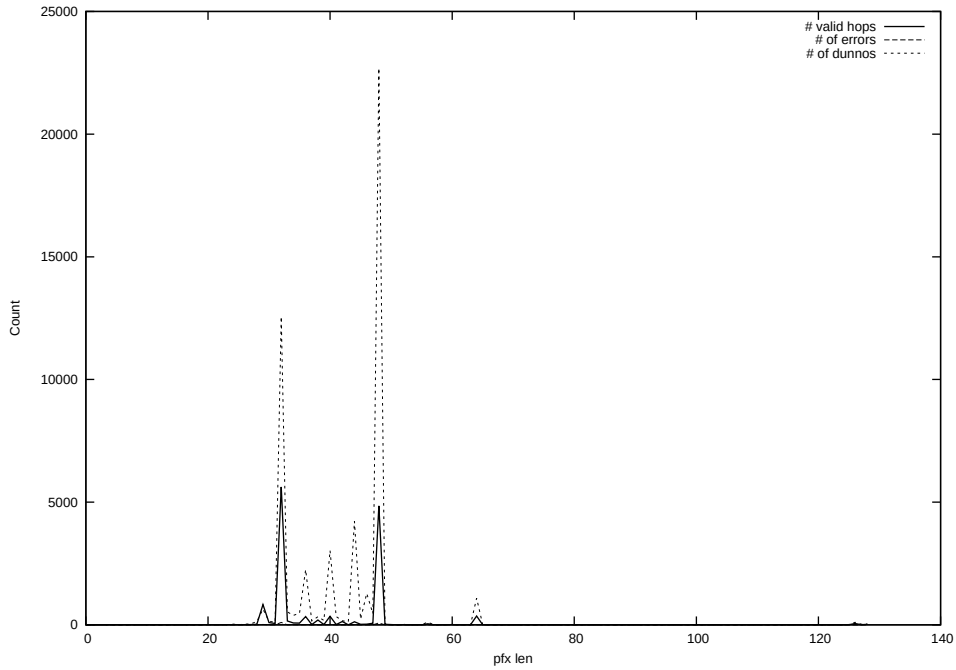
2014-11-11



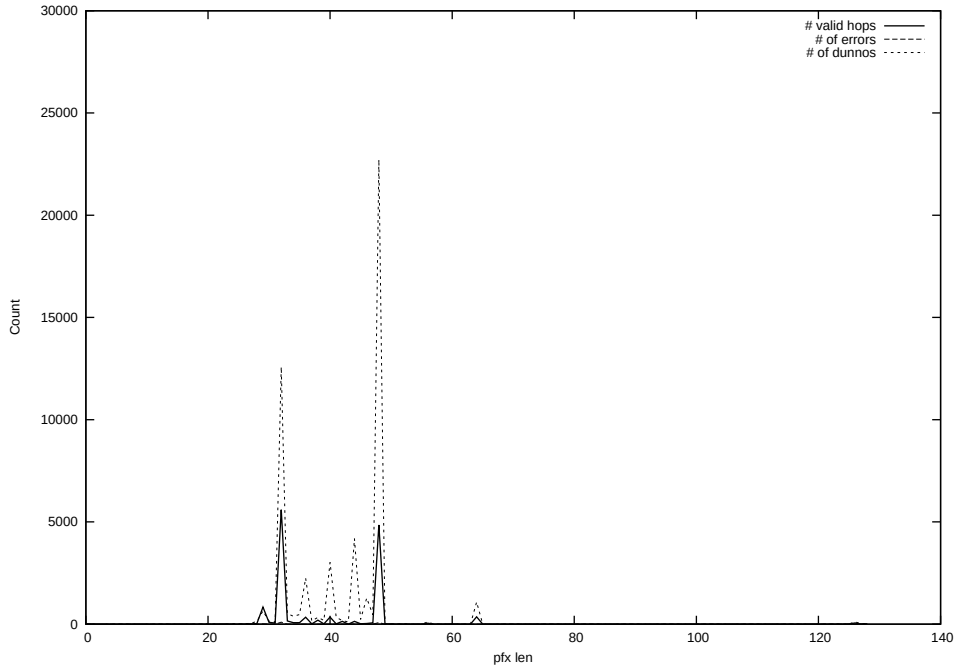
2014-11-12



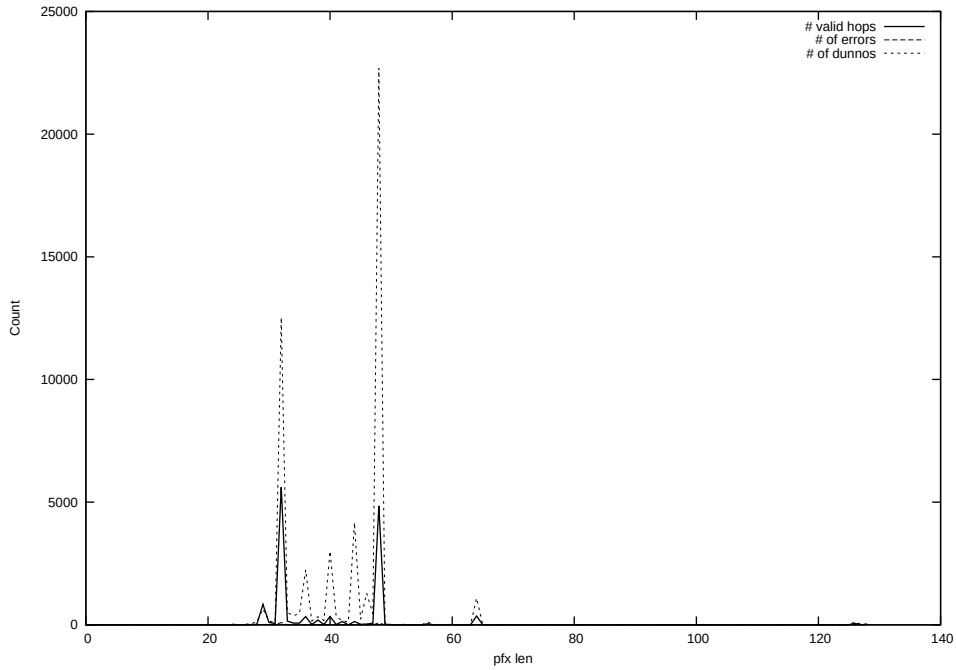
2014-11-13



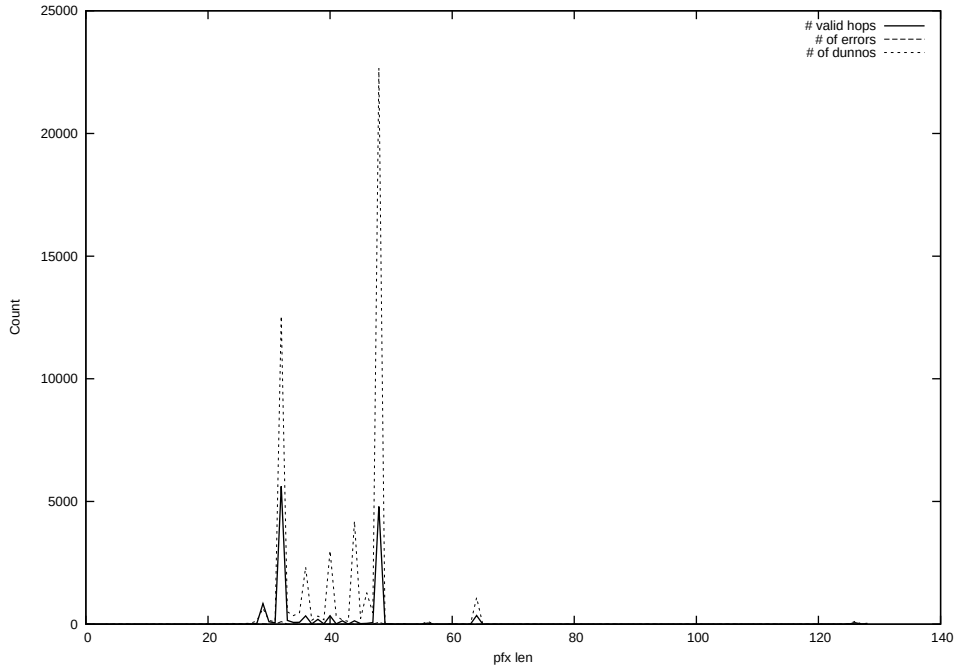
2014-11-14



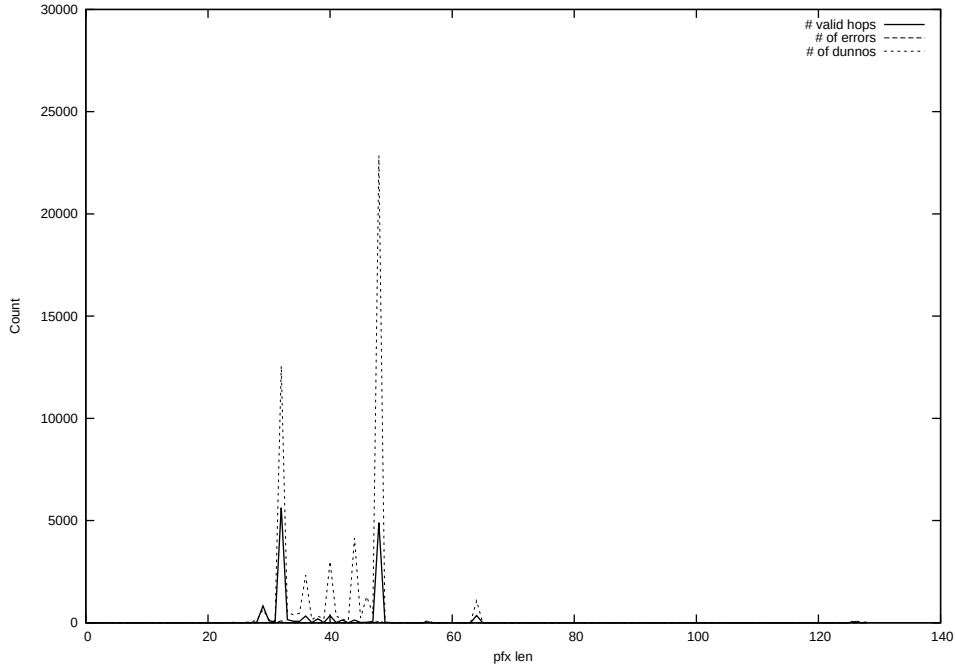
2014-11-15



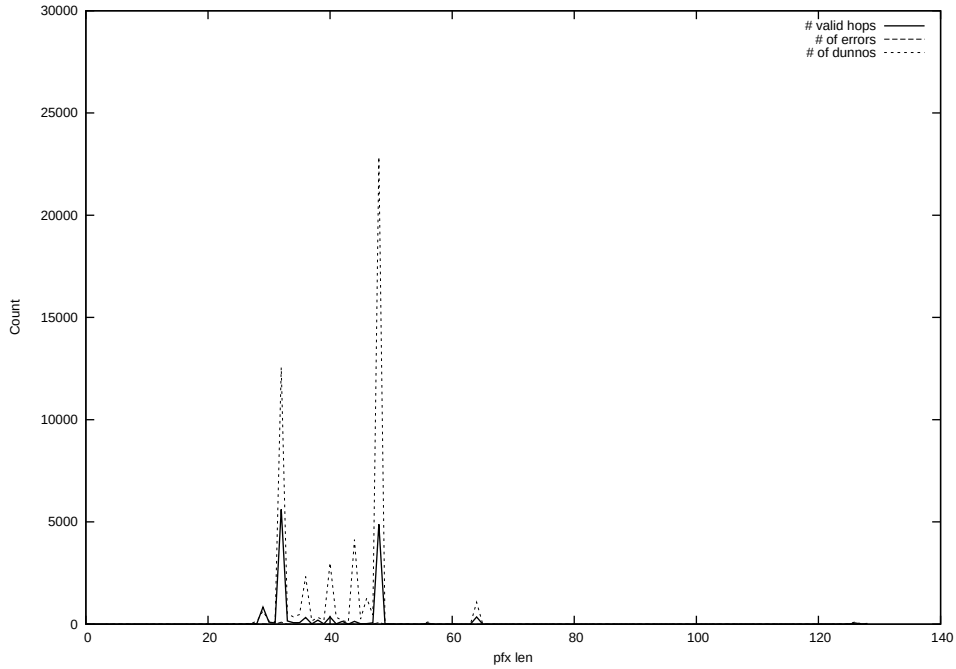
2014-11-16



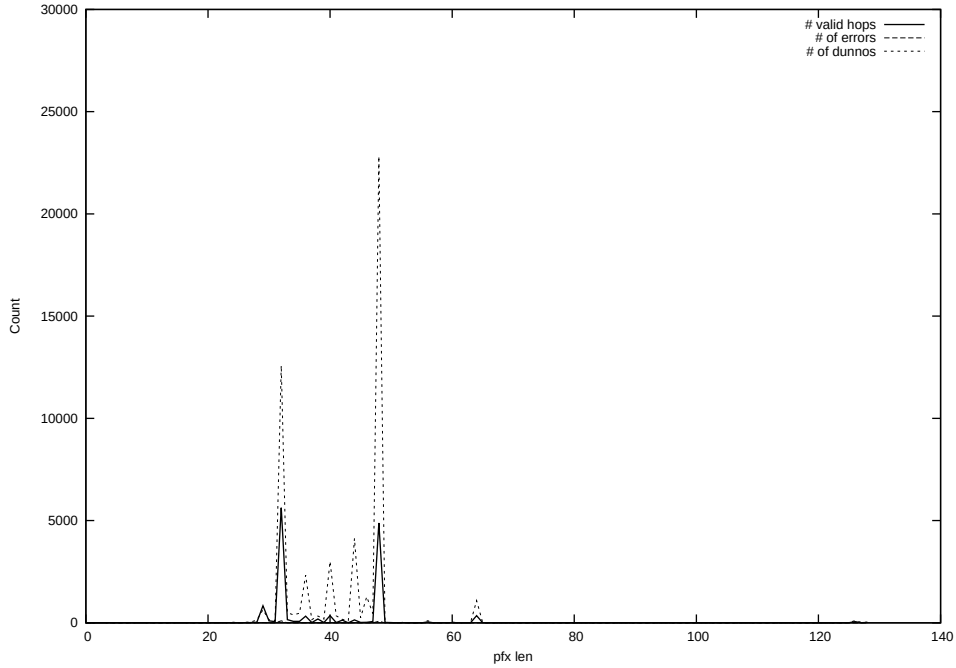
2014-11-17



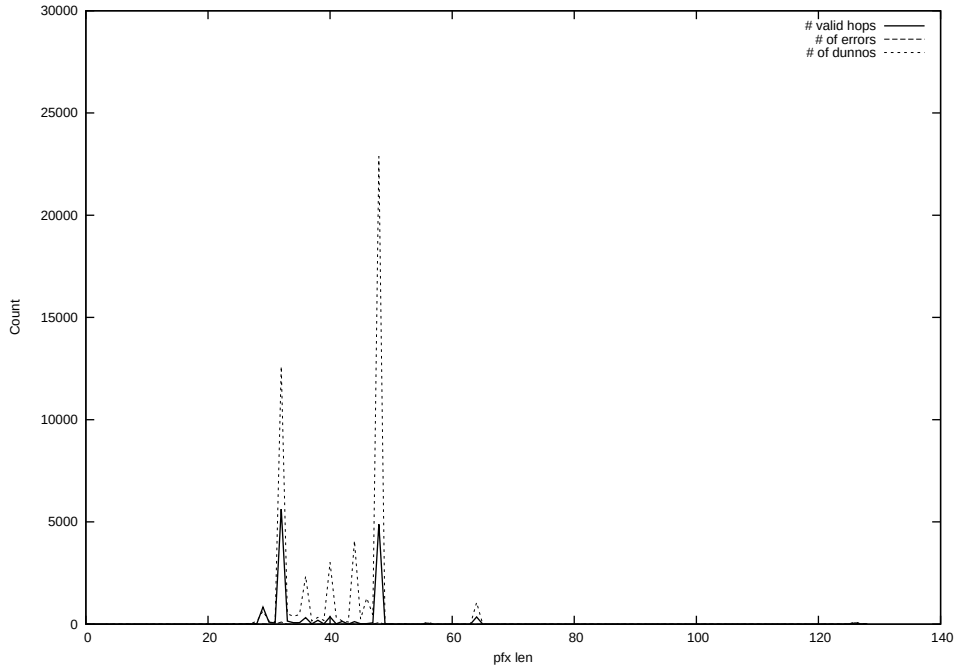
2014-11-18



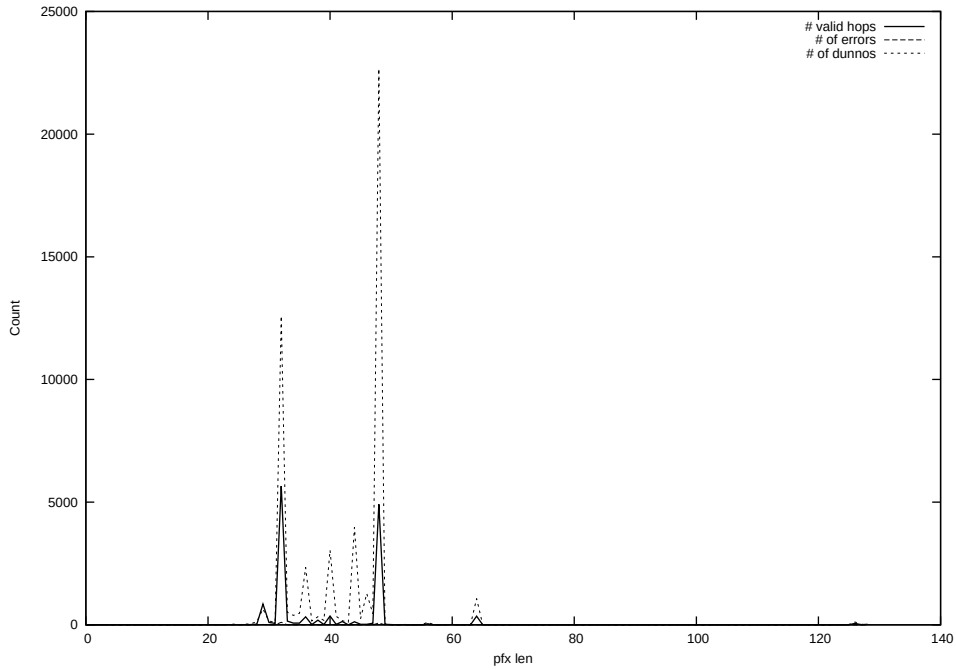
2014-11-19



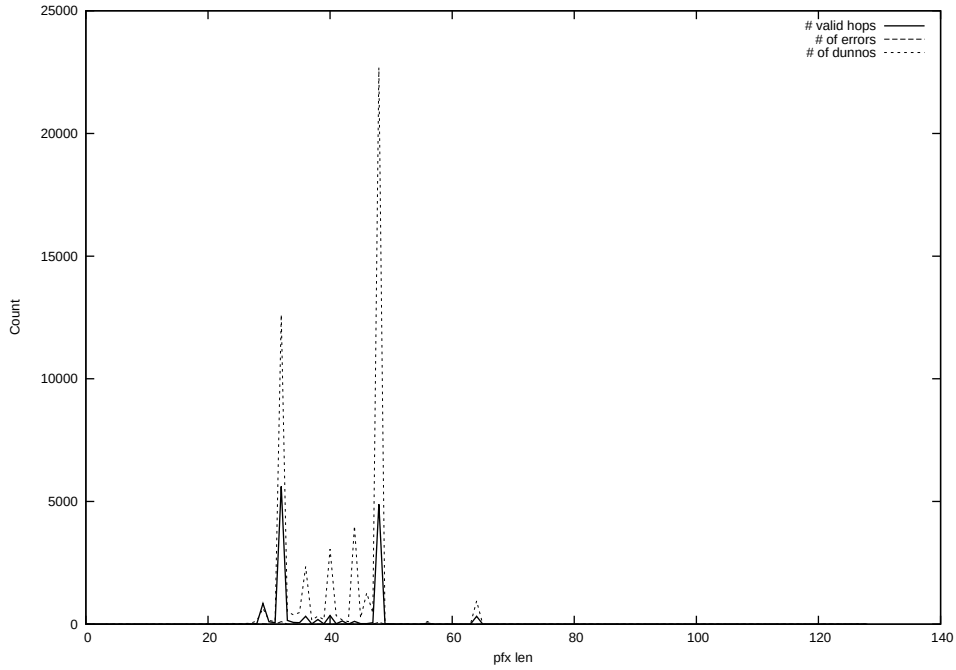
2014-11-20



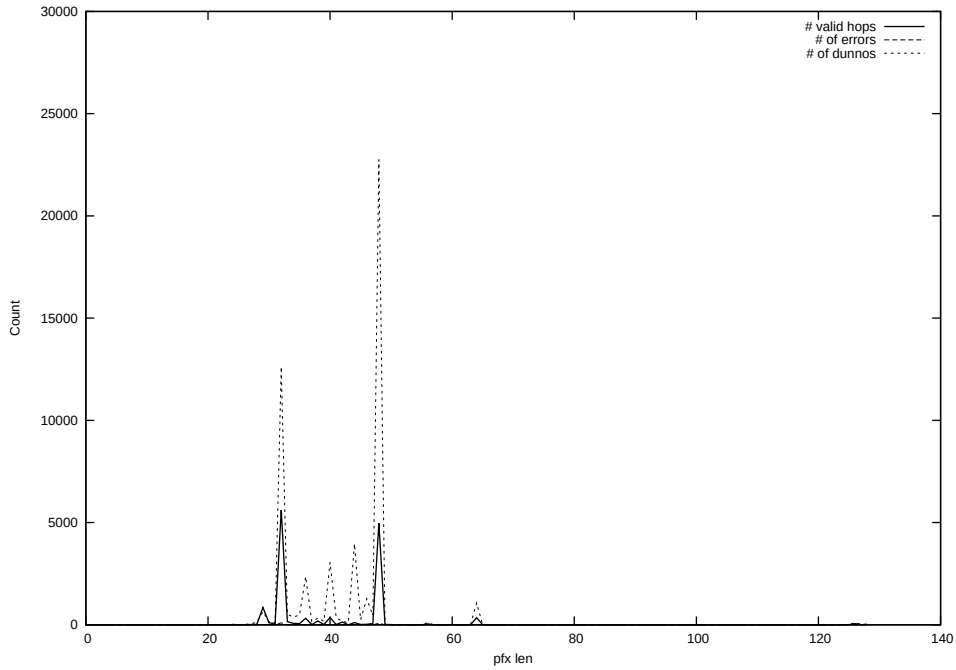
2014-11-21



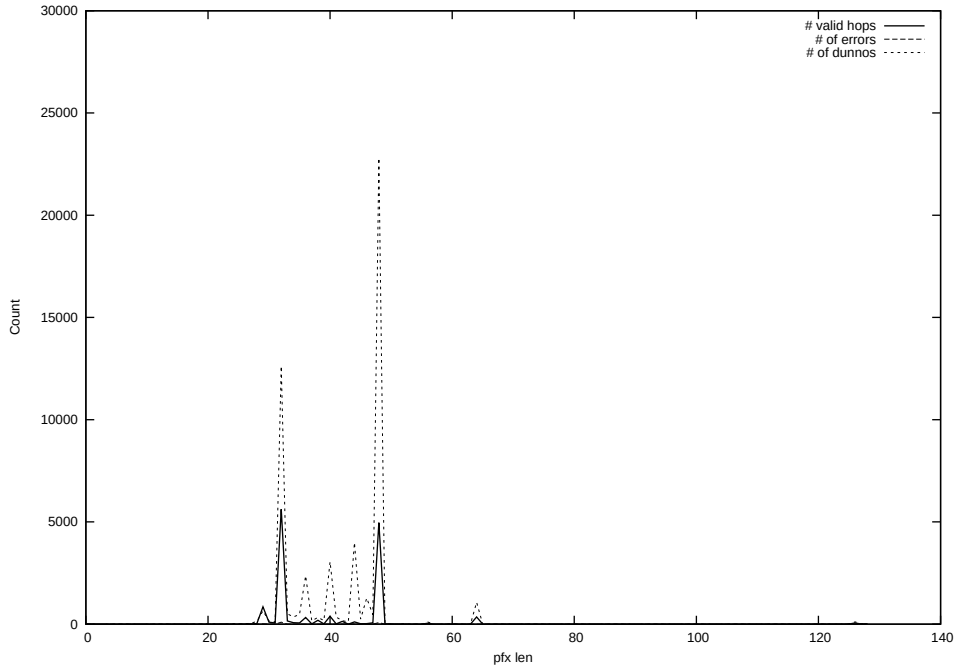
2014-11-22



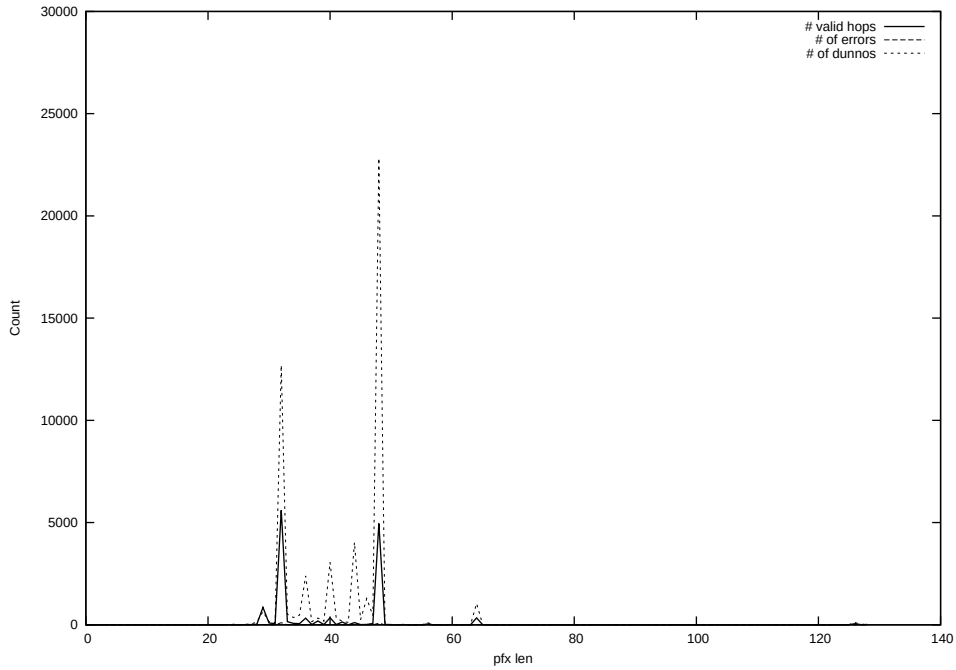
2014-11-23



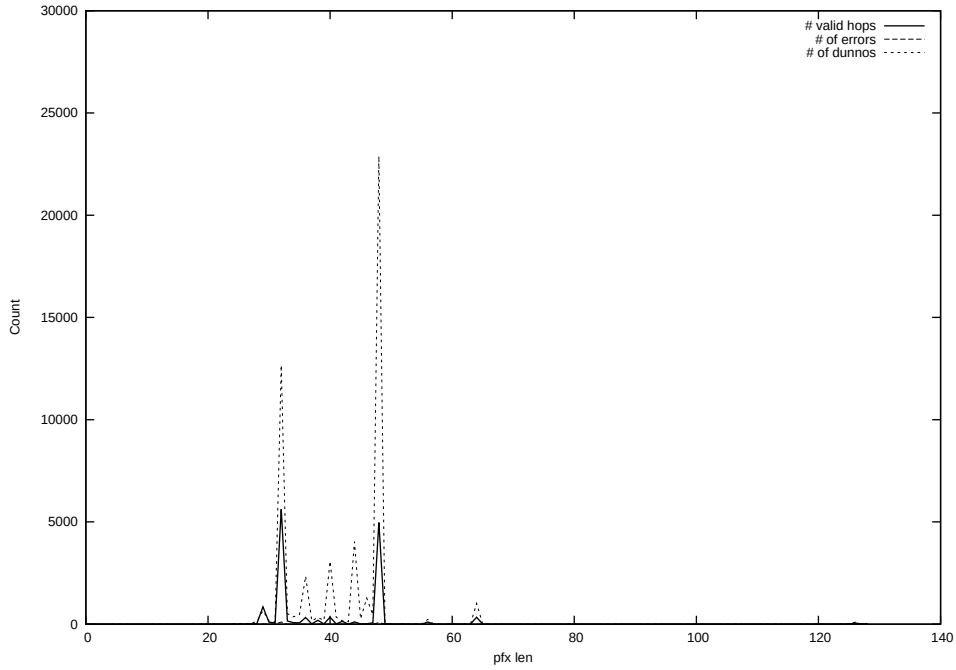
2014-11-24



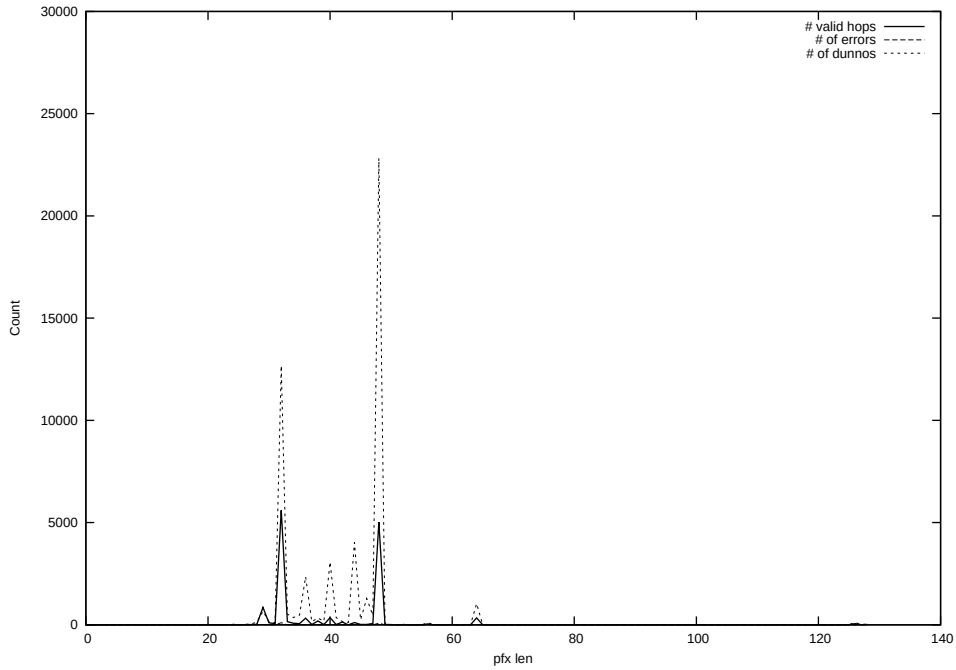
2014-11-25



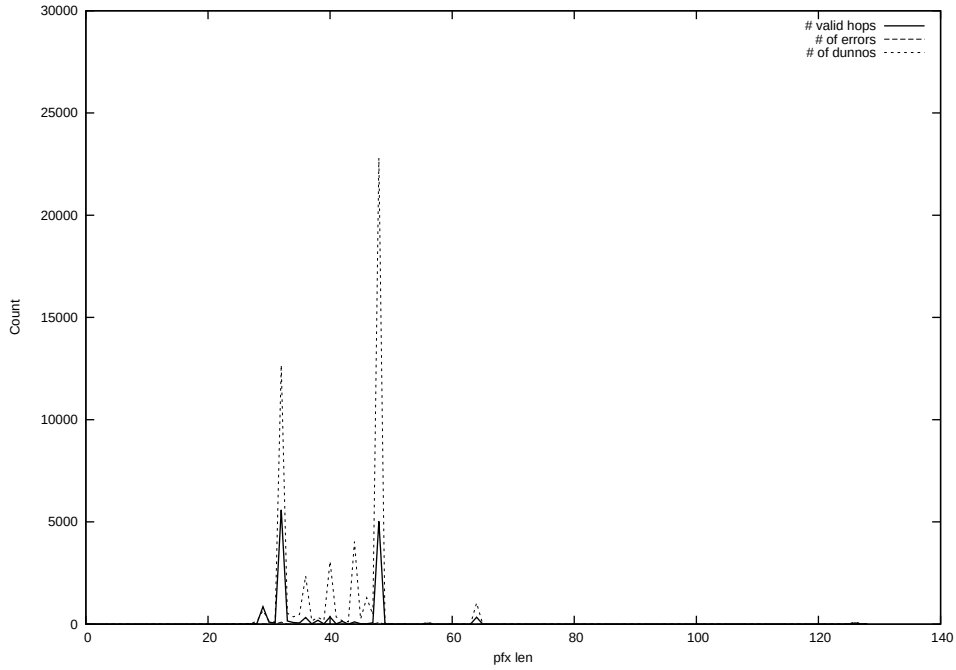
2014-11-26



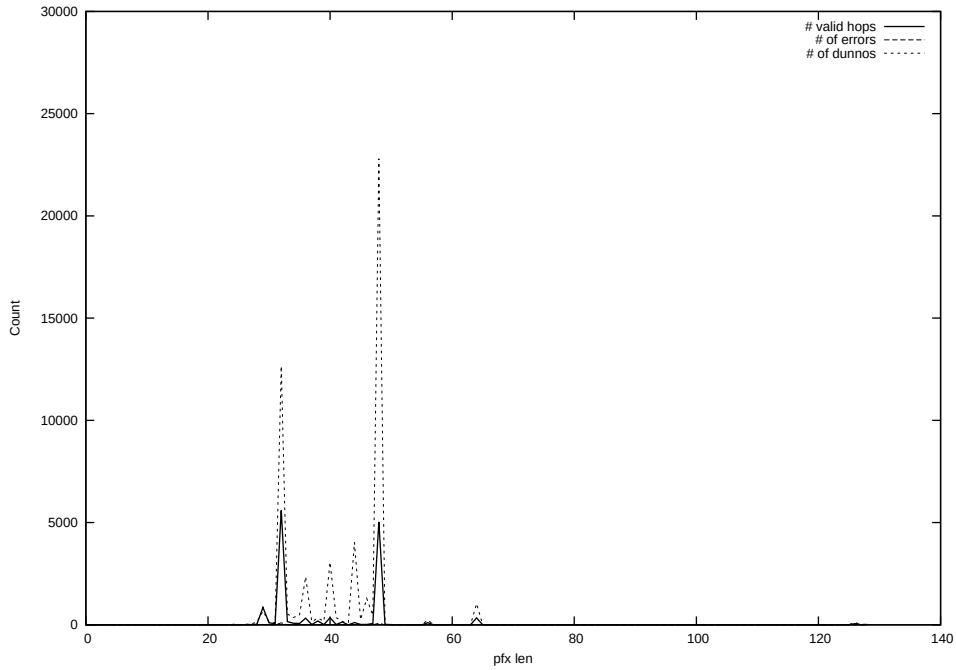
2014-11-27



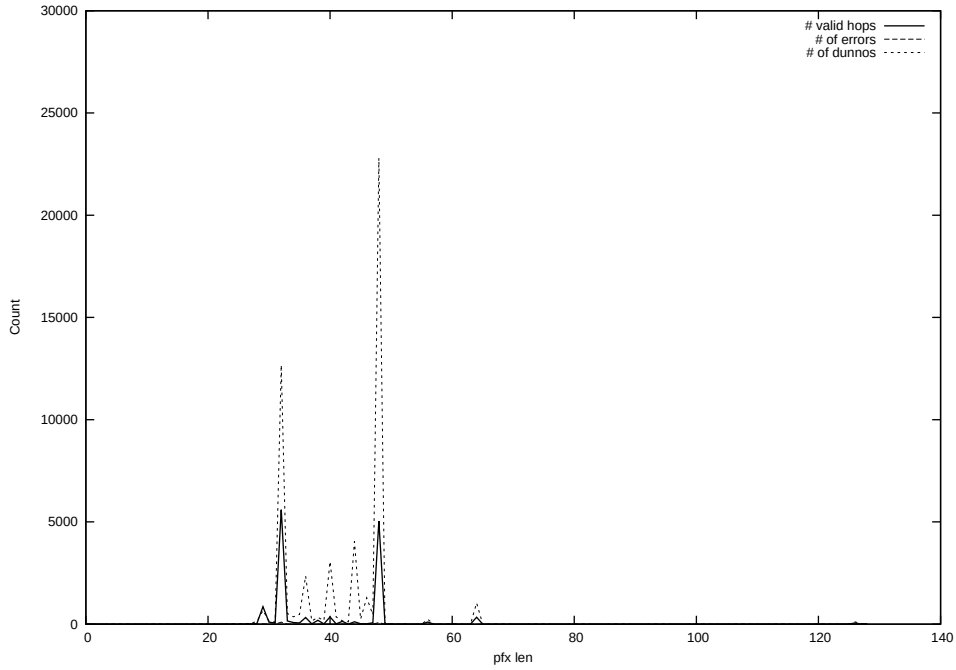
2014-11-28



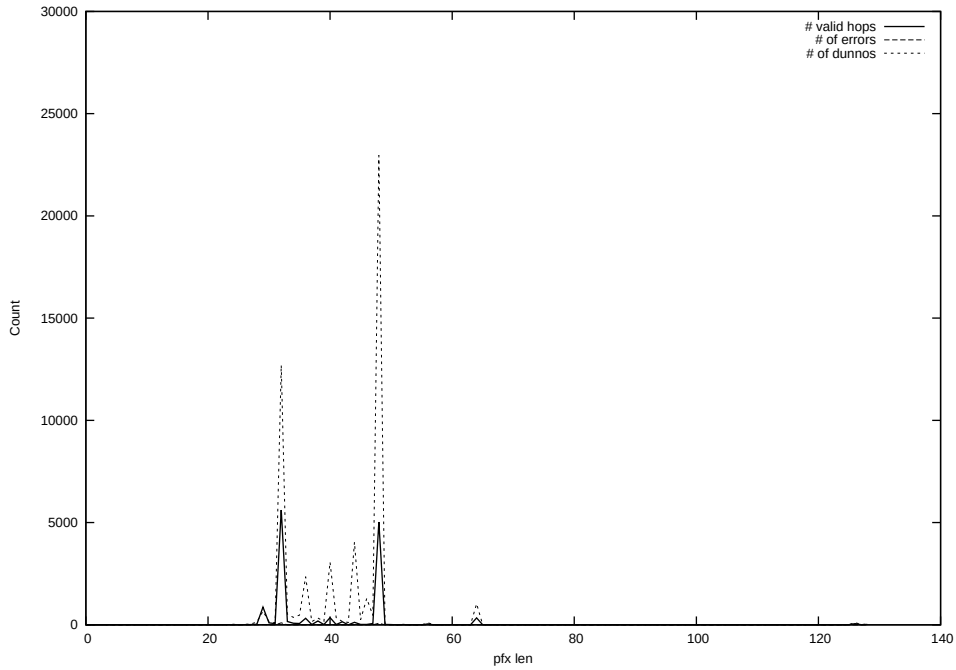
2014-11-29



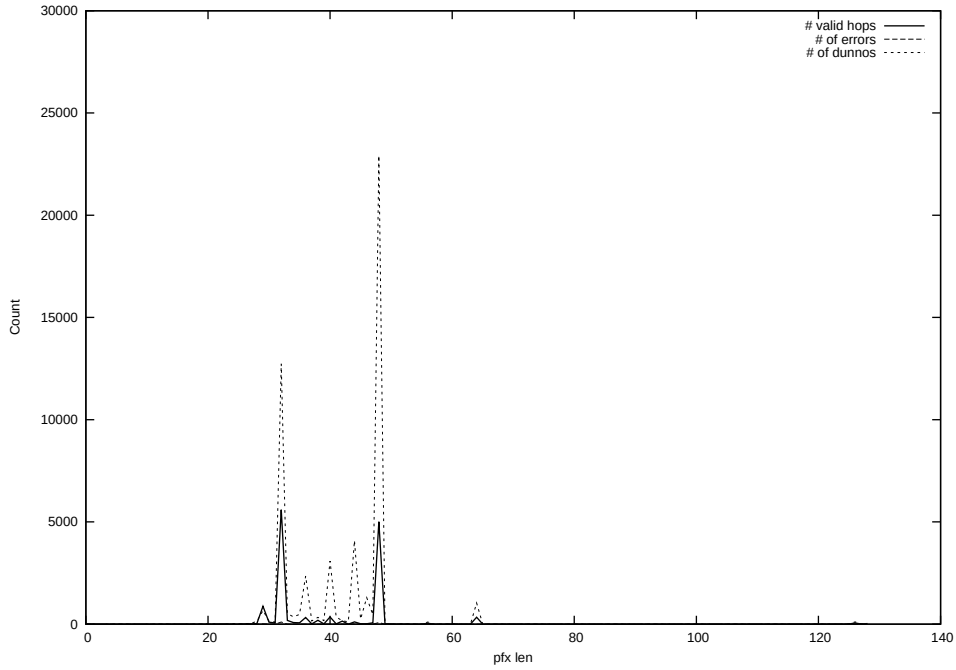
2014-11-30



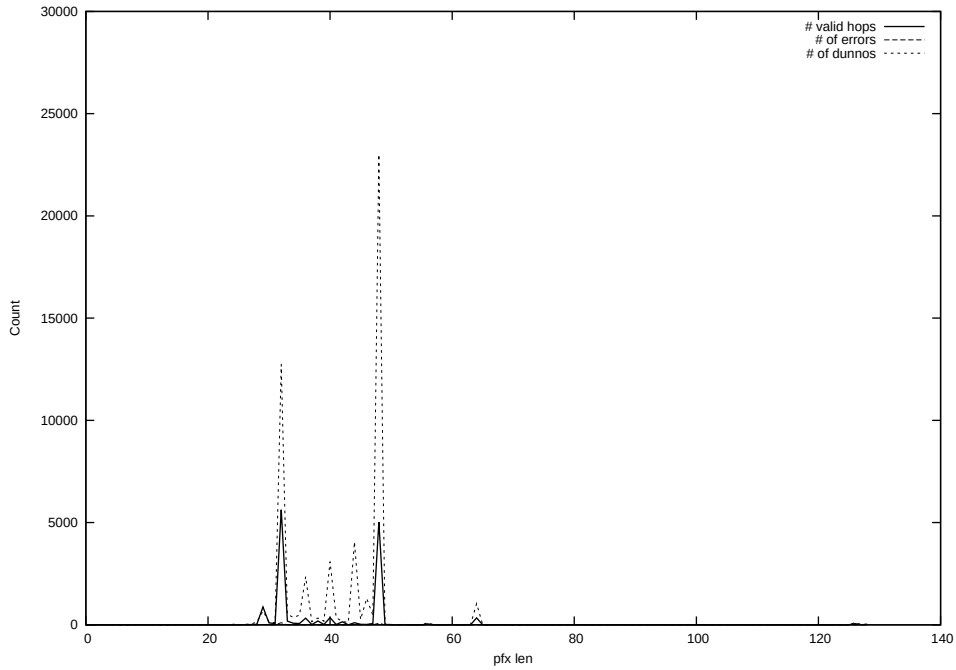
2014-12-01



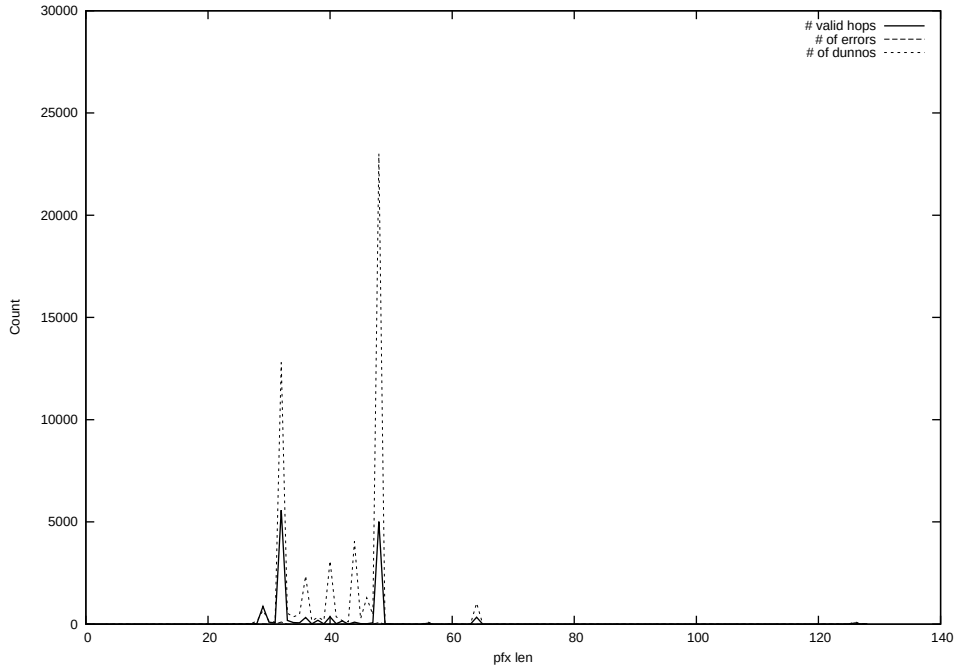
2014-12-02



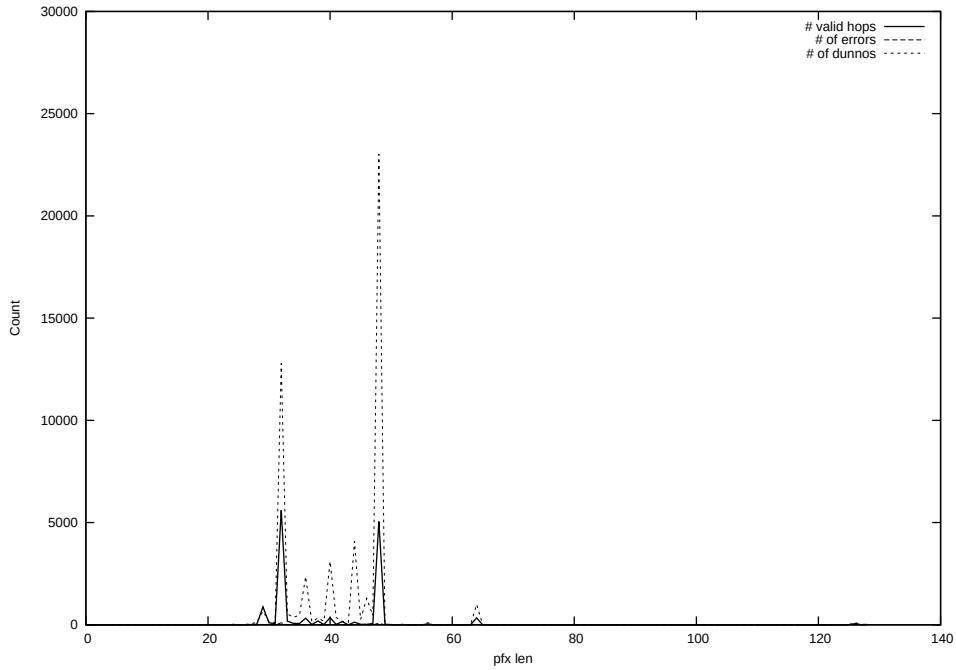
2014-12-03



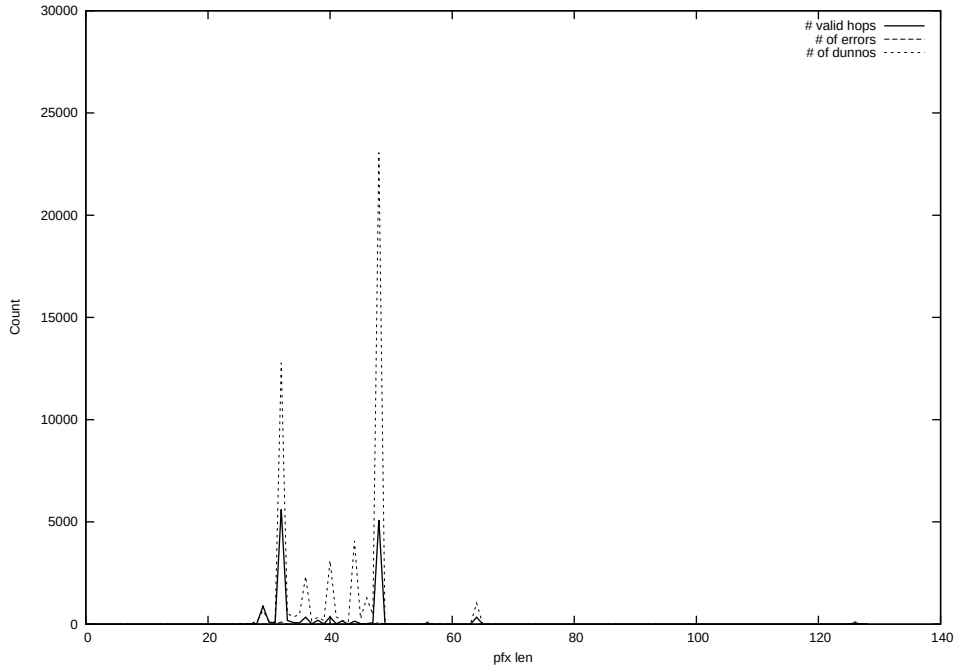
2014-12-04



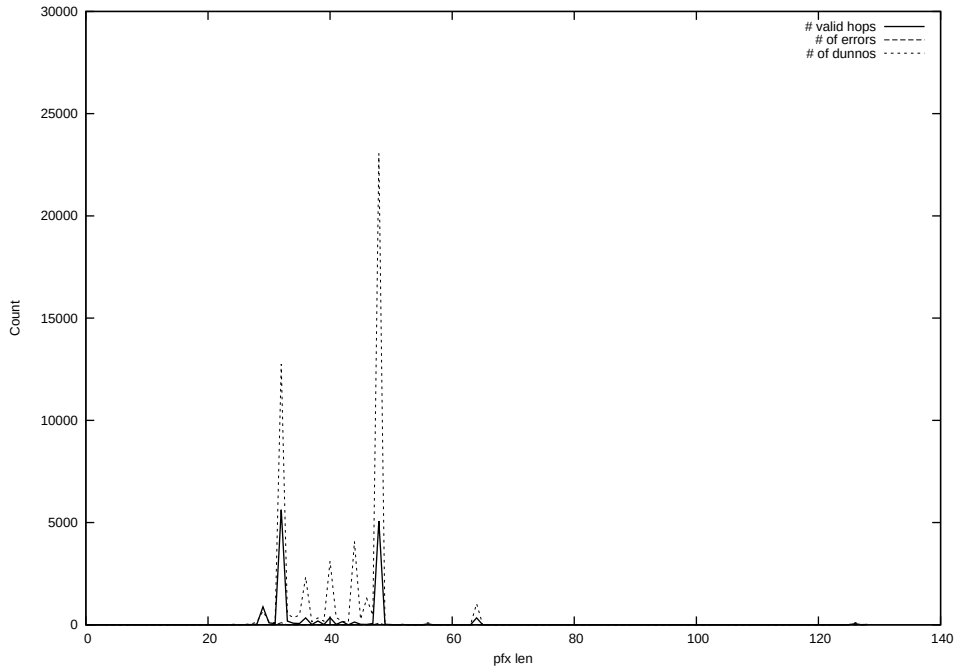
2014-12-05



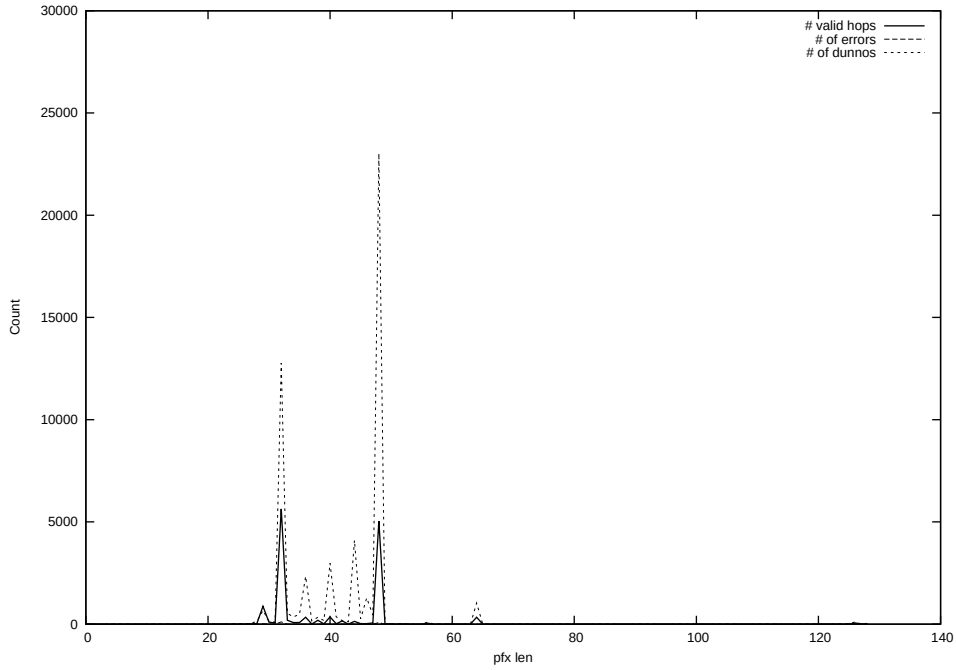
2014-12-06



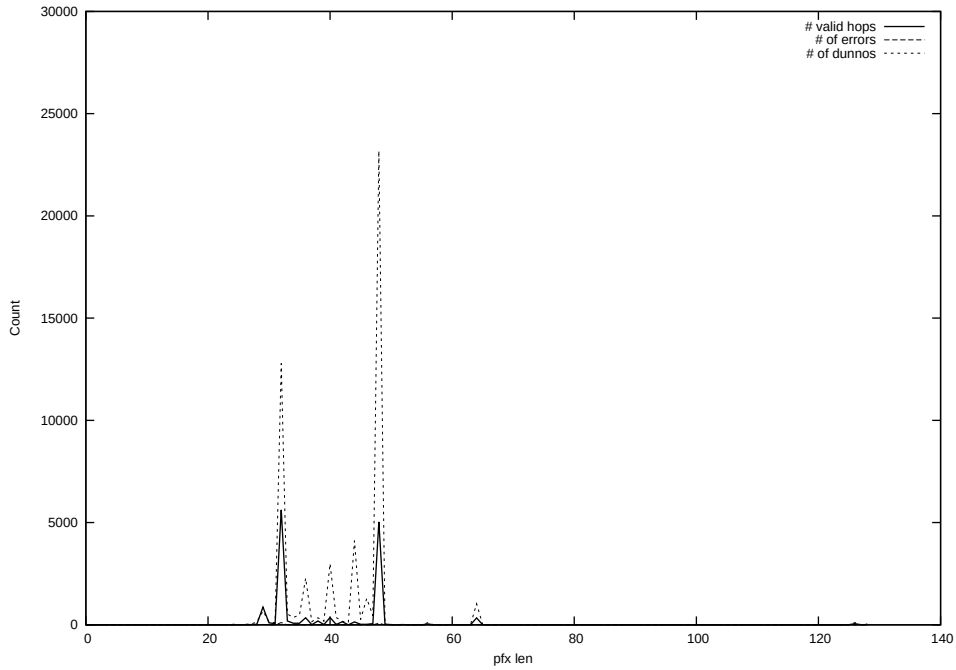
2014-12-07



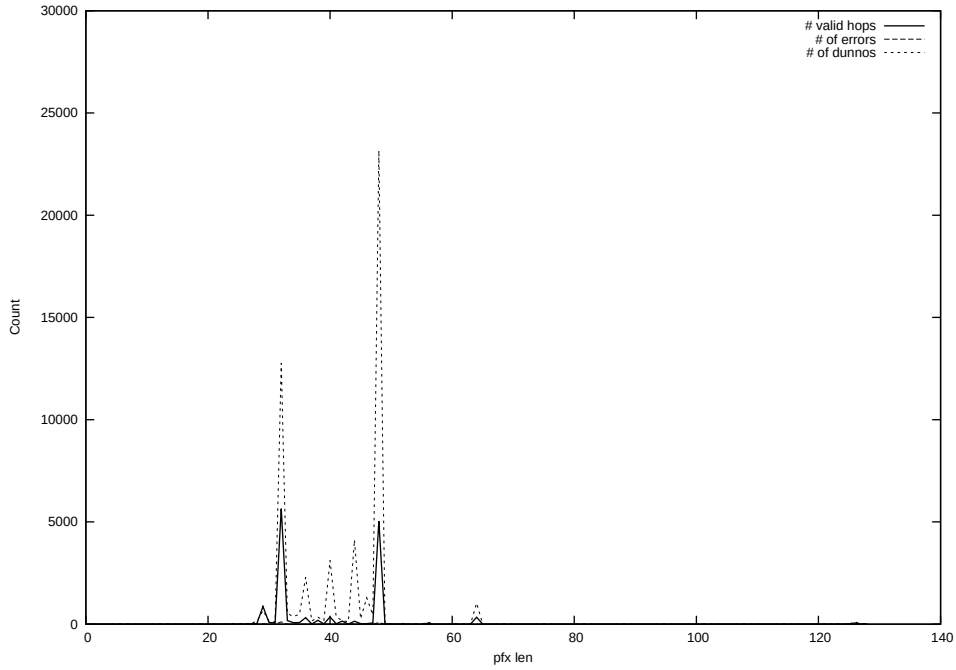
2014-12-08



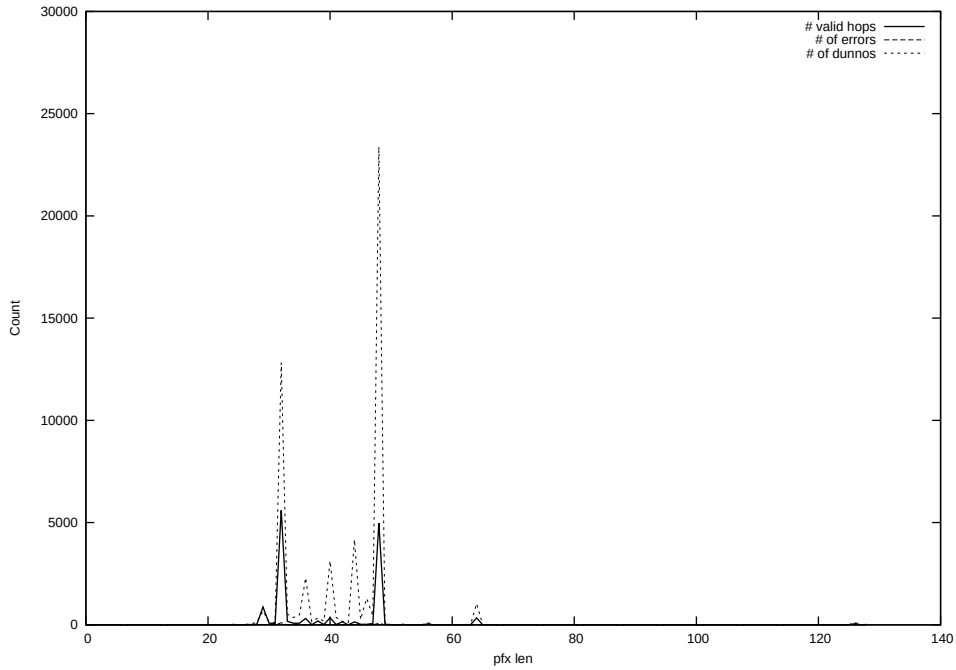
2014-12-09



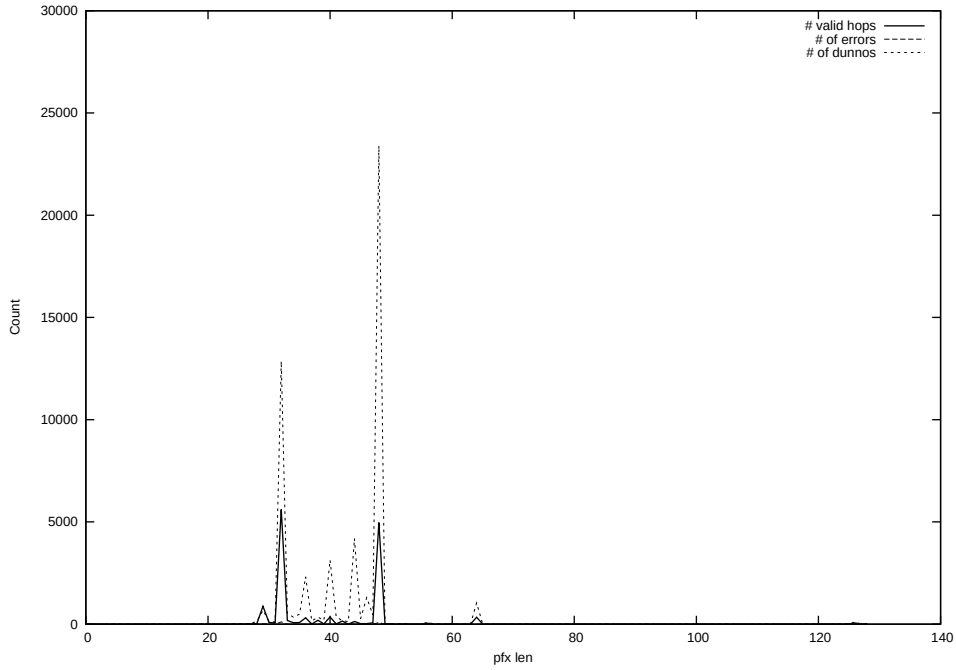
2014-12-10



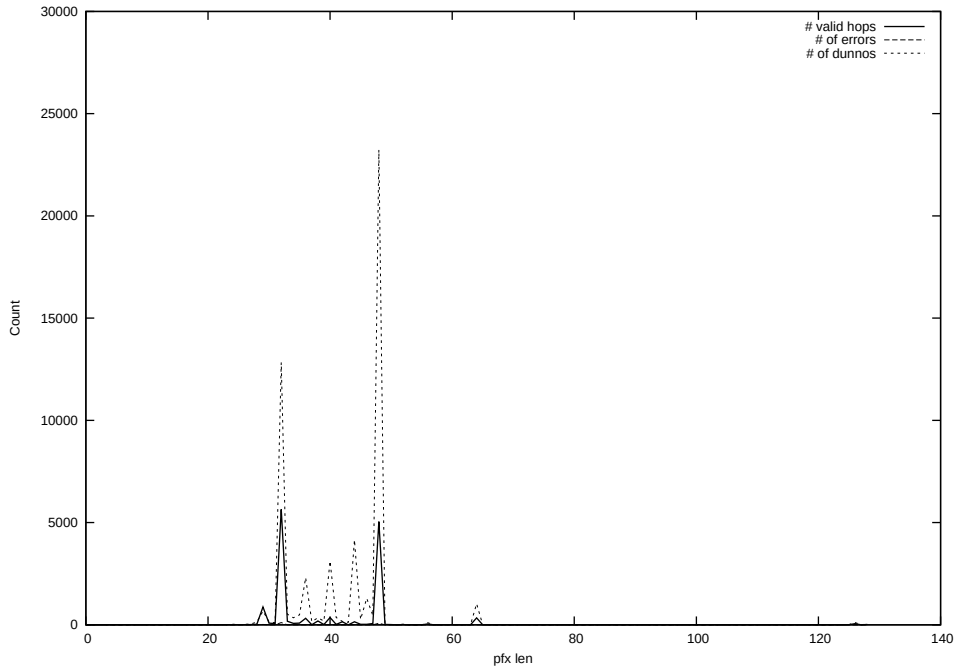
2014-12-11



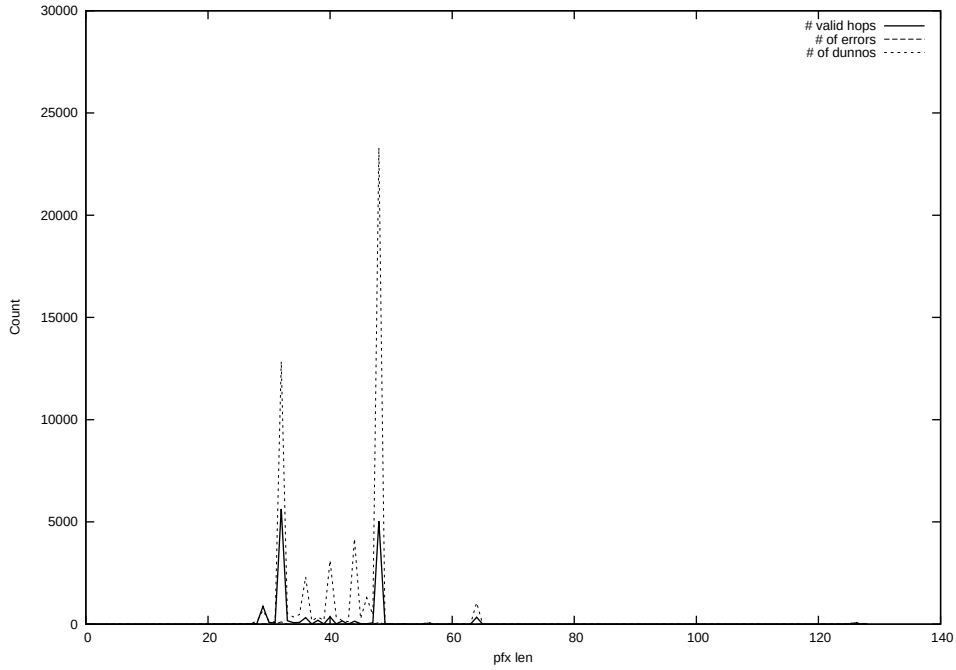
2014-12-12



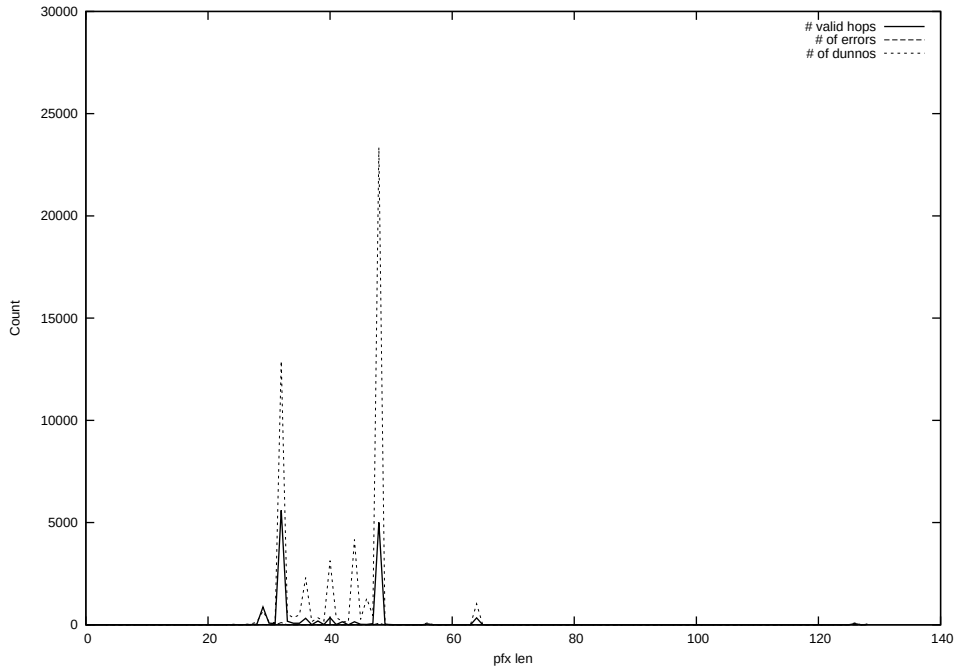
2014-12-13



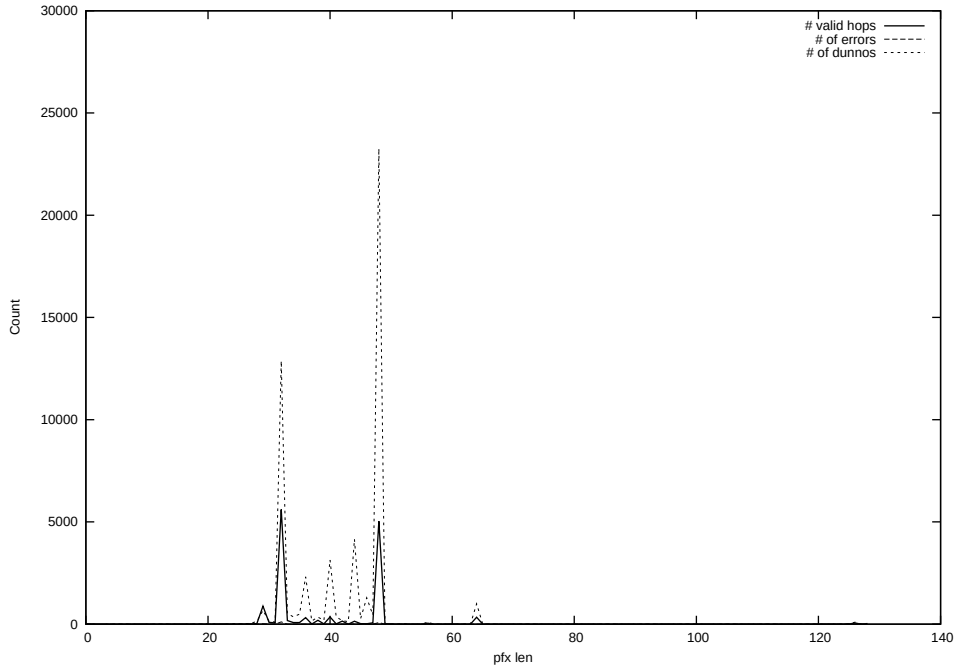
2014-12-14



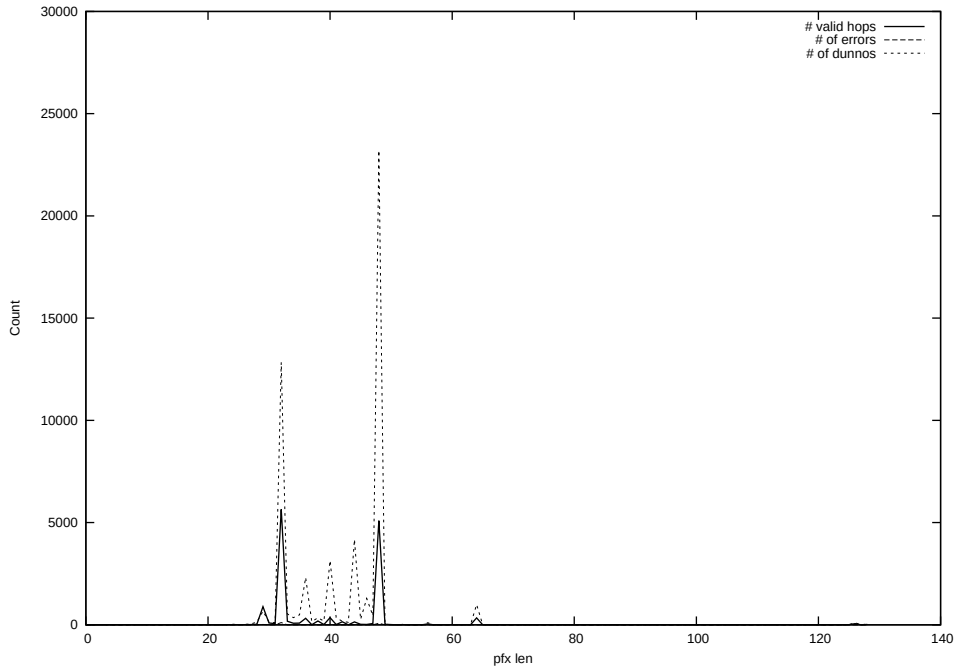
2014-12-15



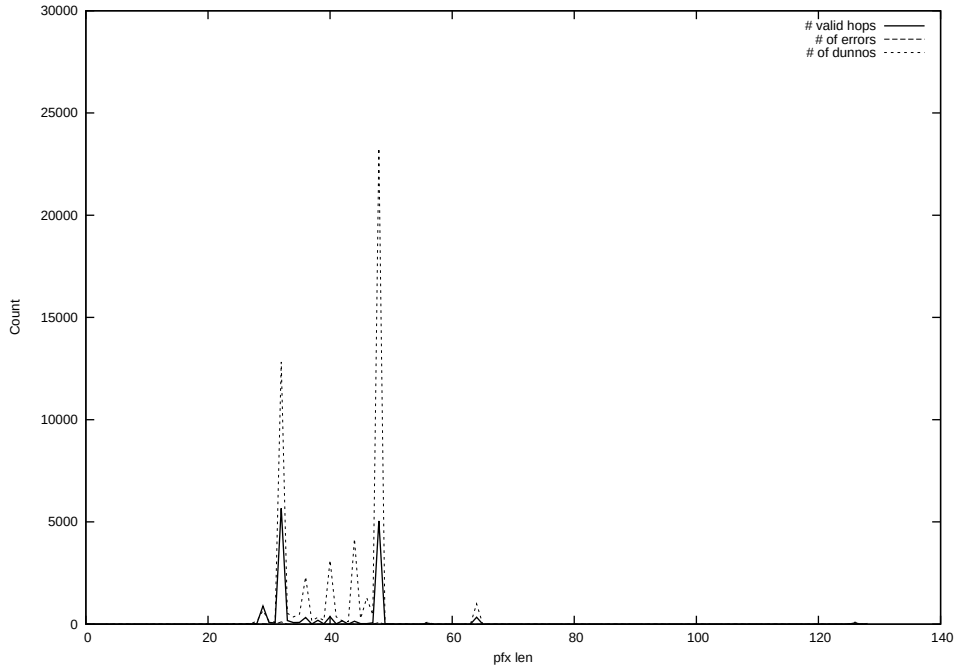
2014-12-16



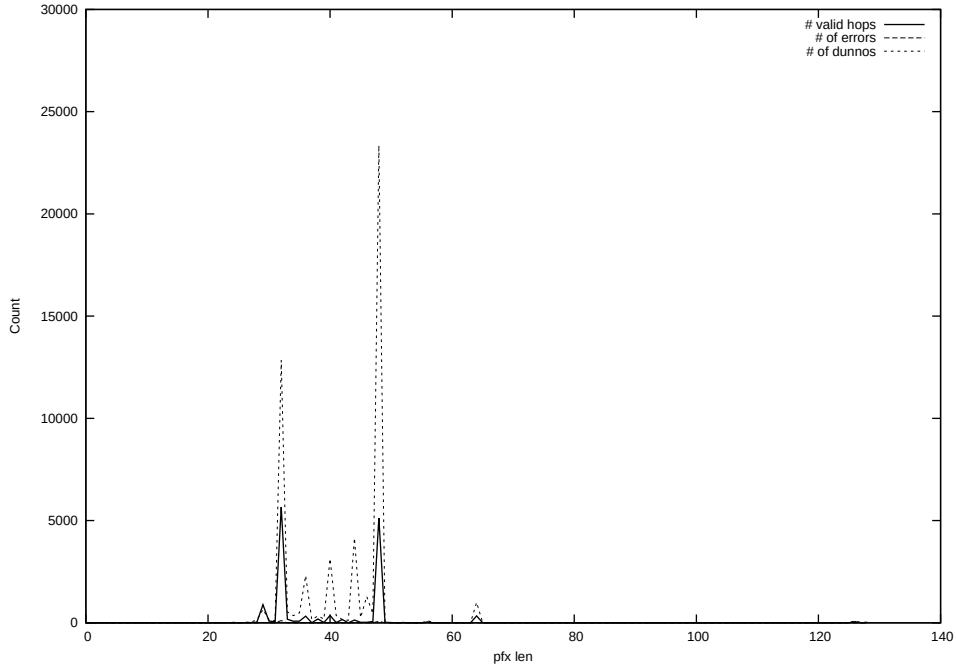
2014-12-17



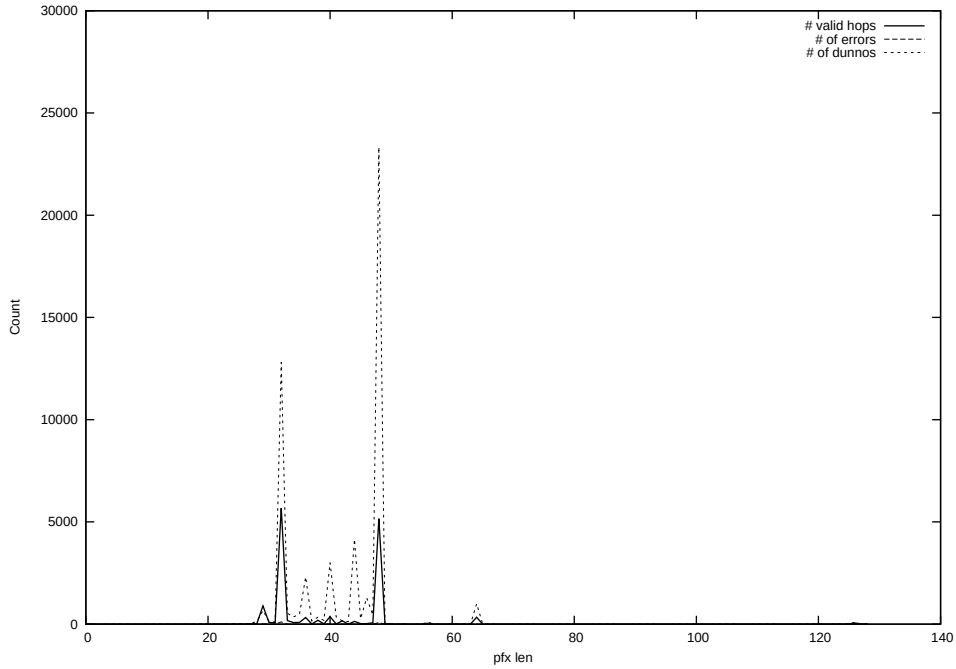
2014-12-18



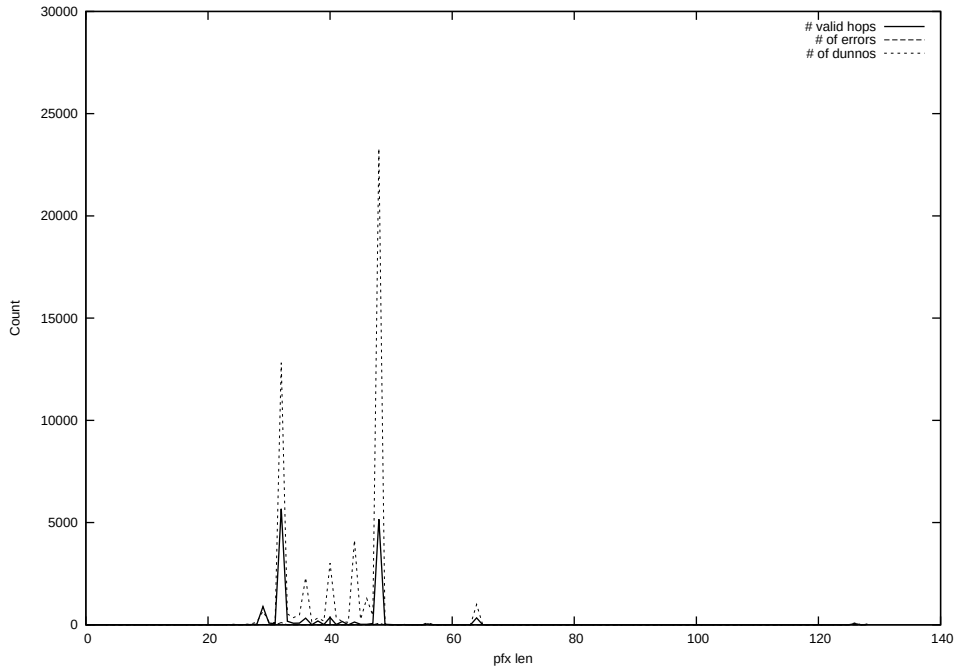
2014-12-19



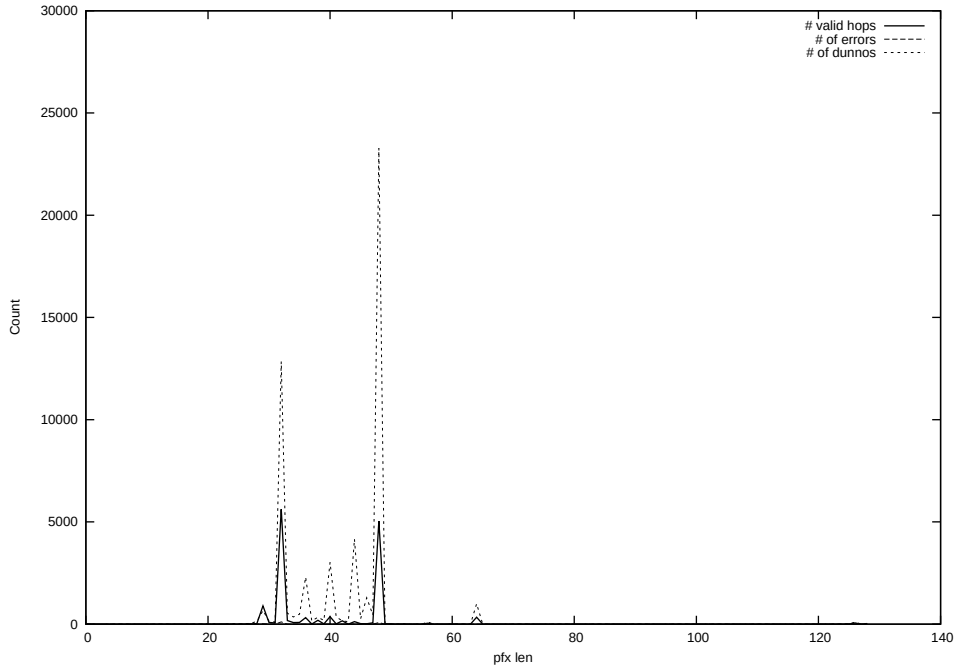
2014-12-20



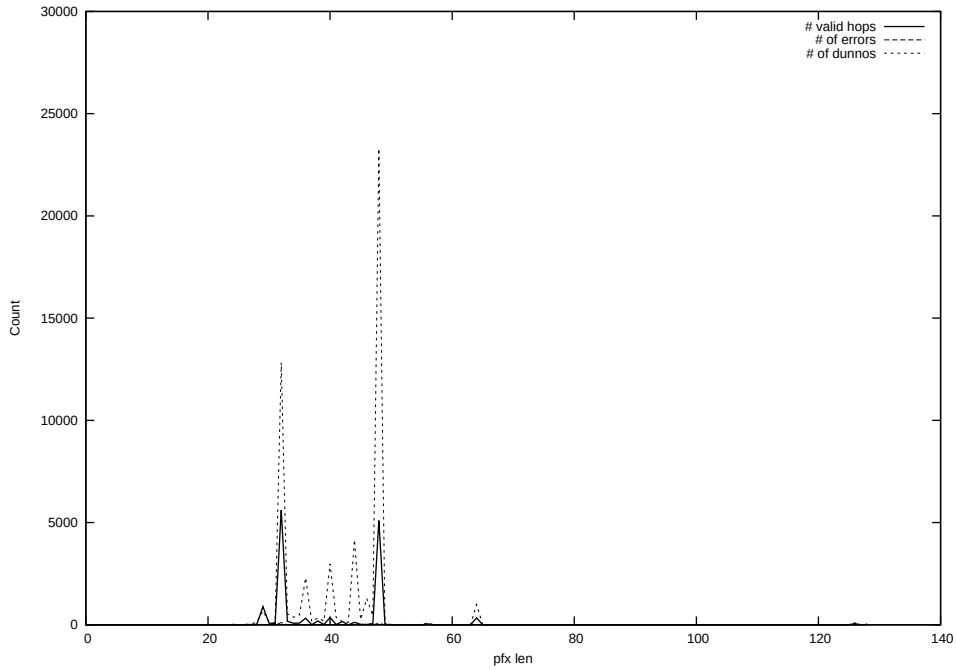
2014-12-21



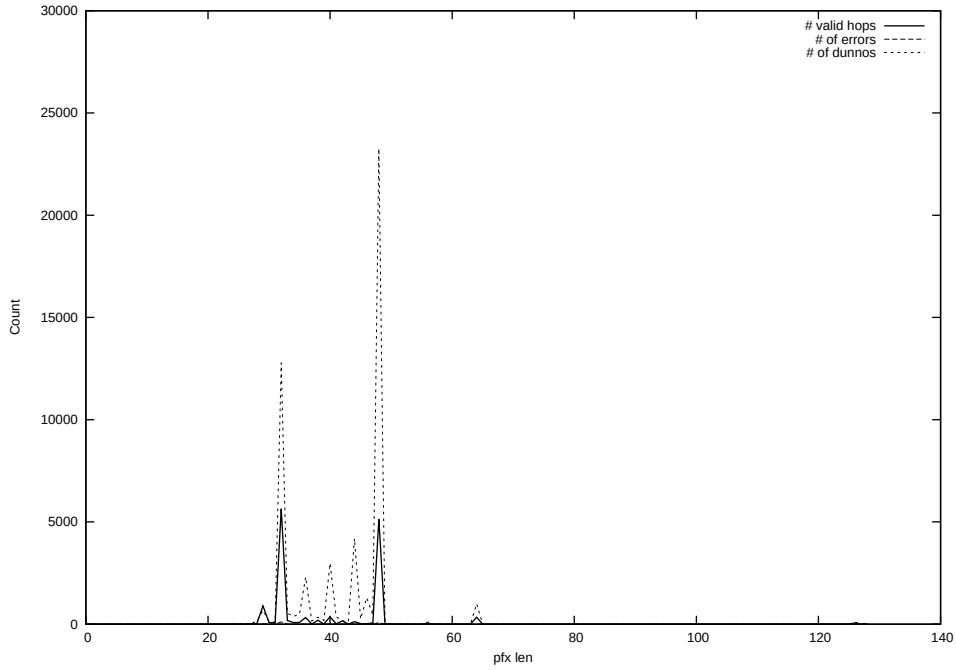
2014-12-22



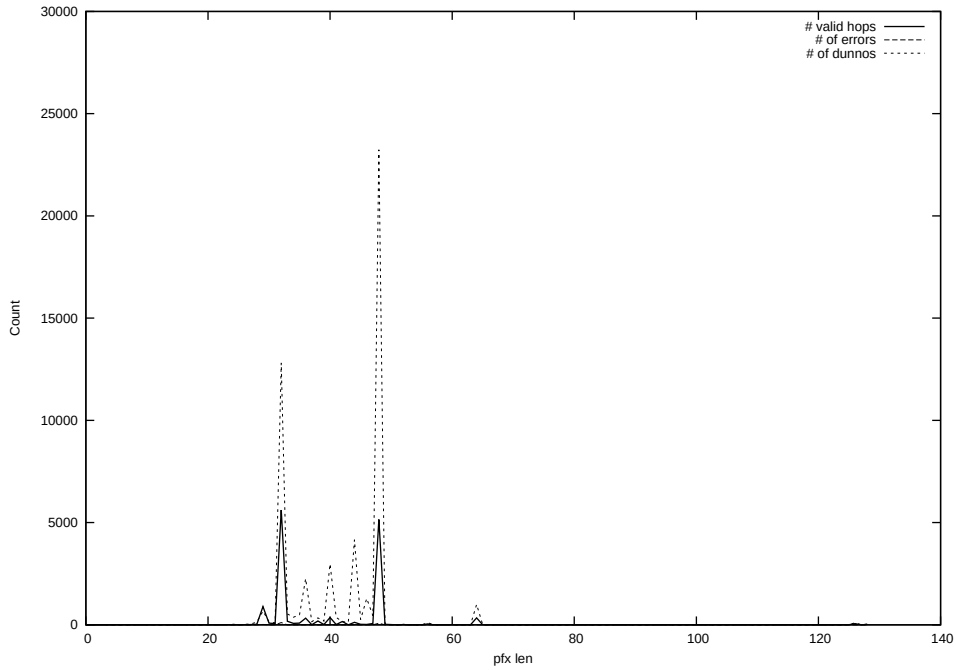
2014-12-23



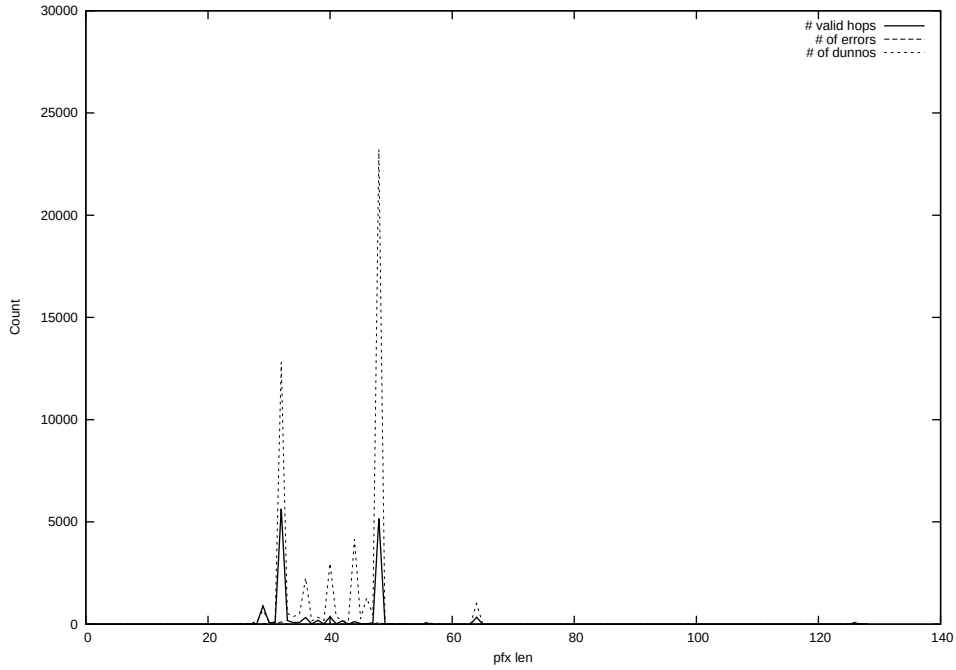
2014-12-24



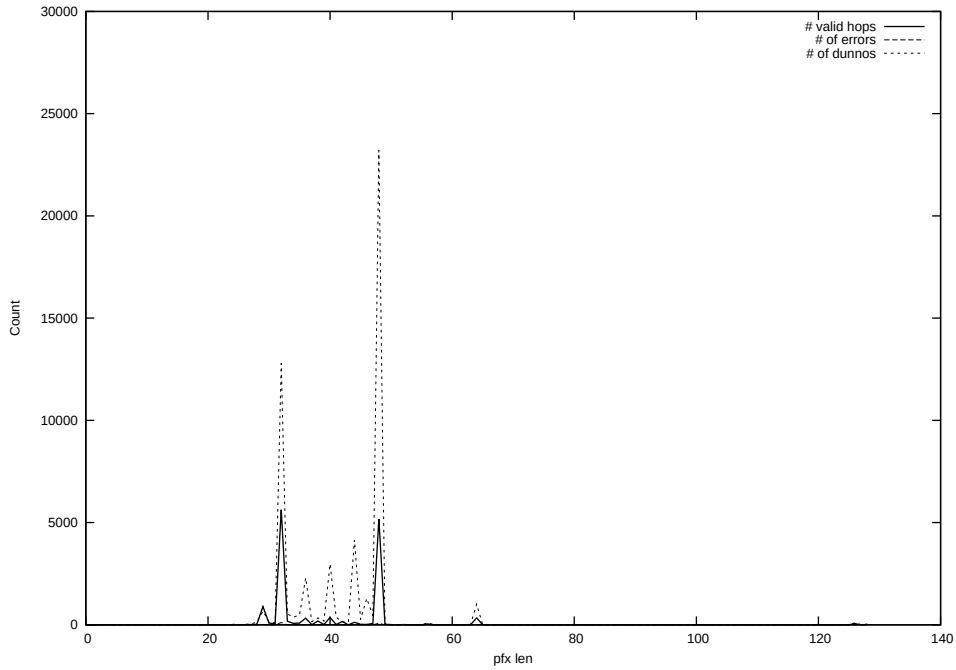
2014-12-25



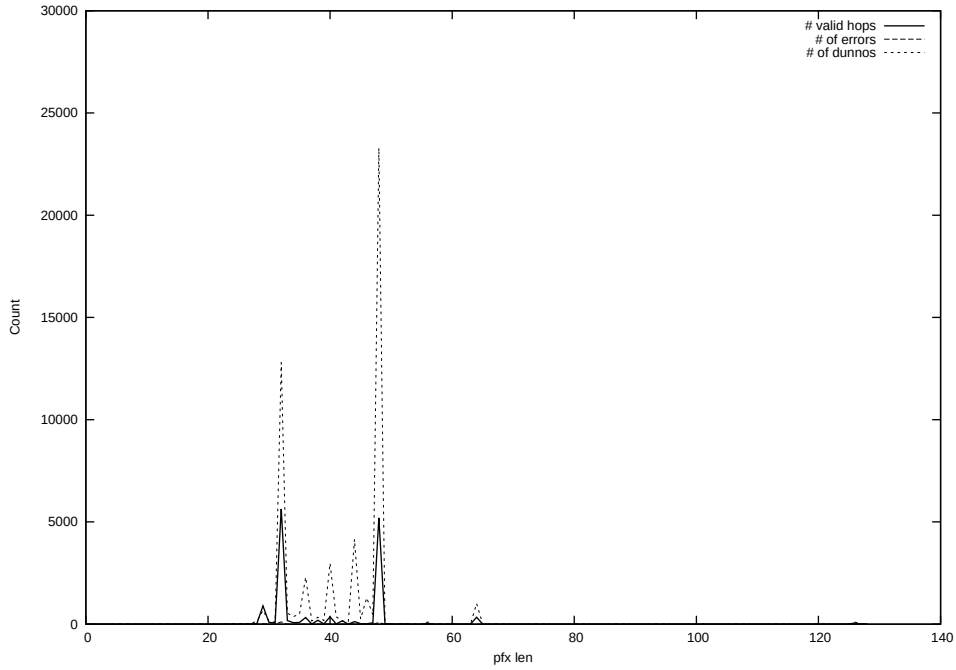
2014-12-26



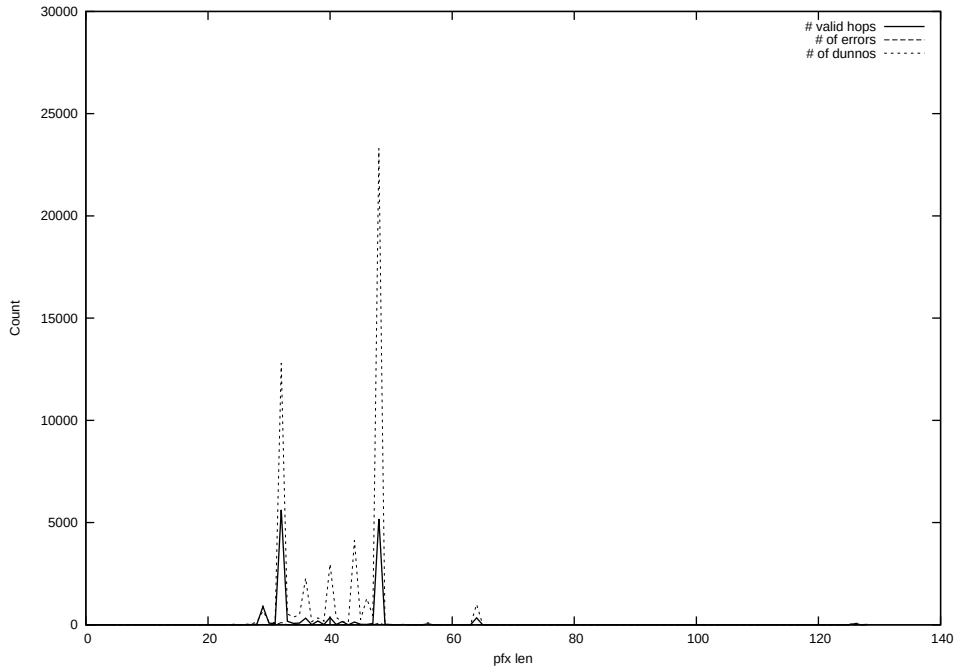
2014-12-27



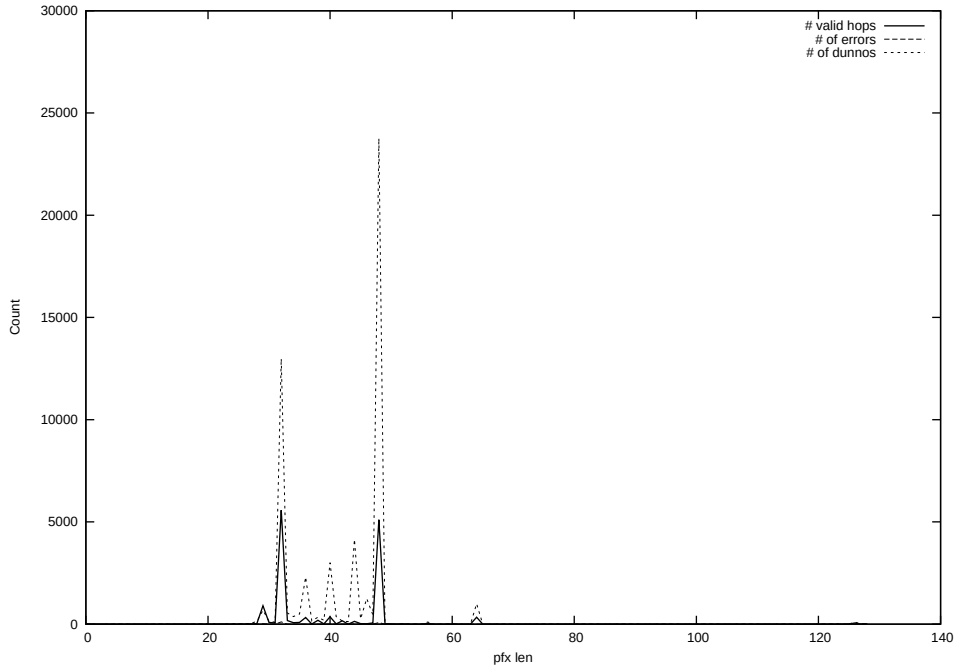
2014-12-28



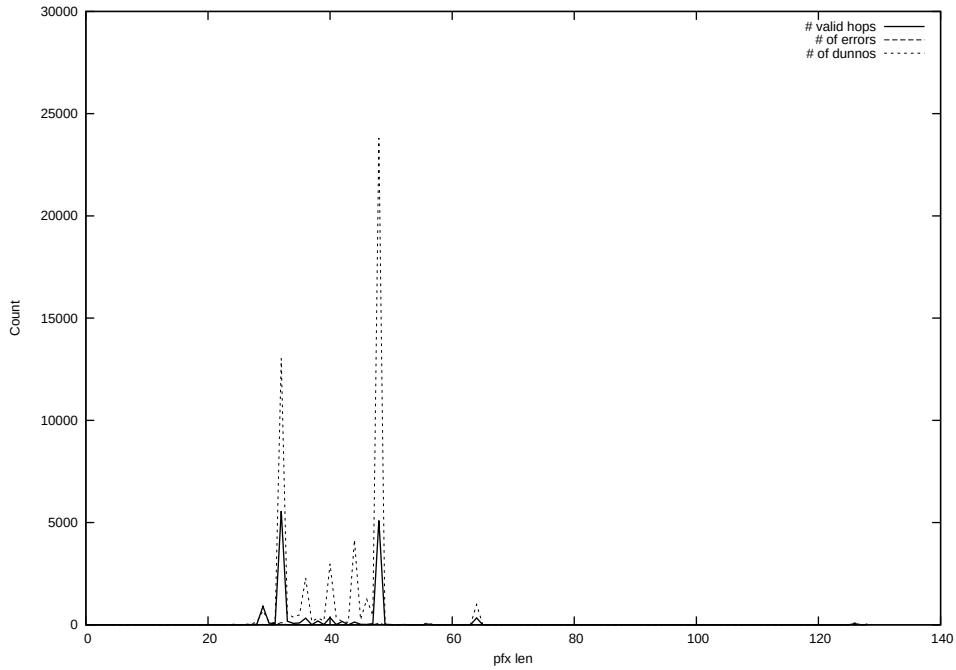
2014-12-29



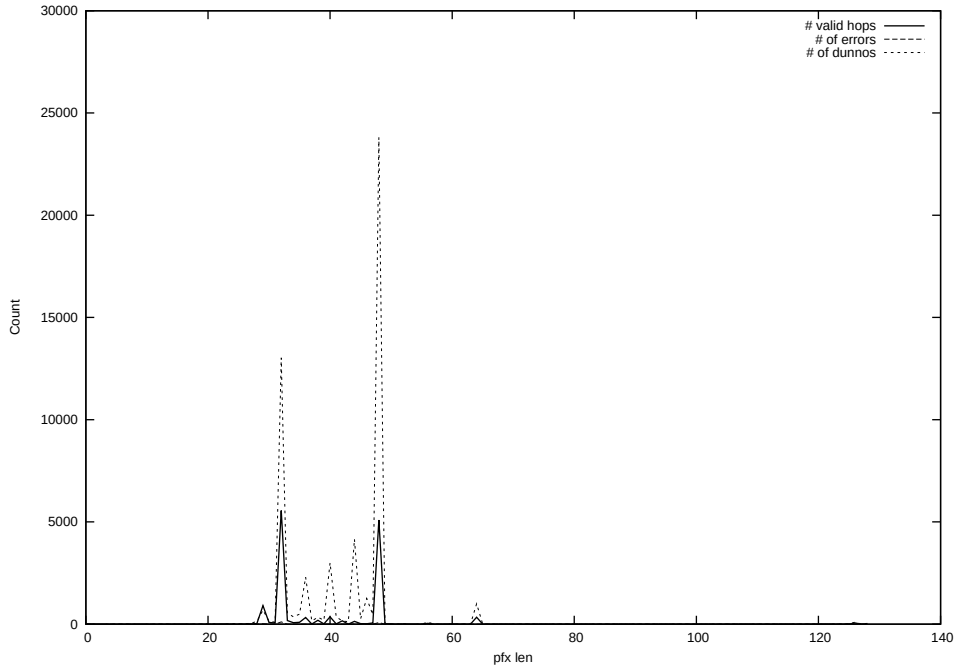
2014-12-30



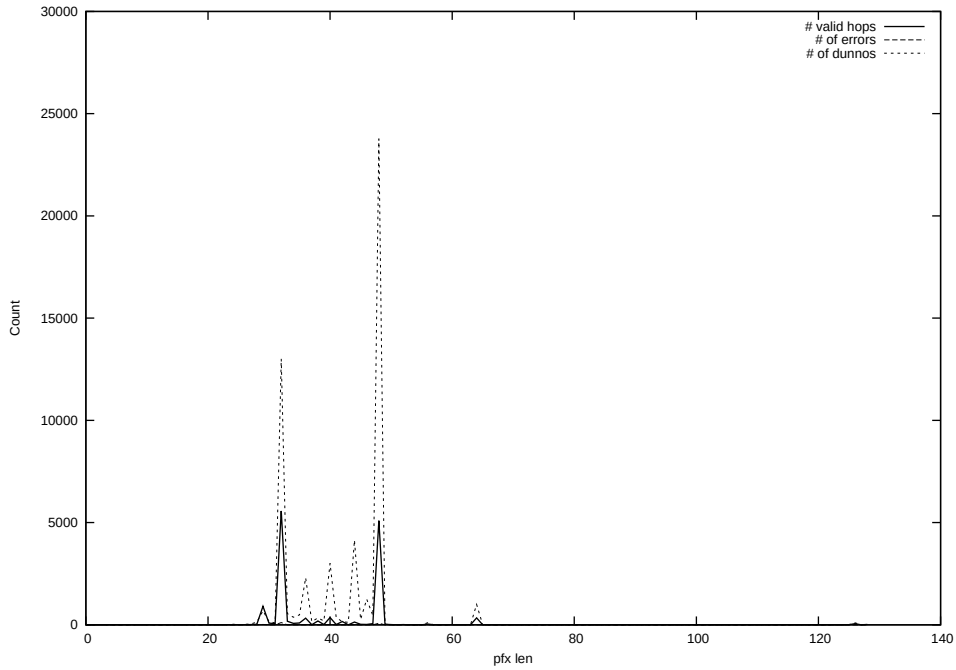
2014-12-31



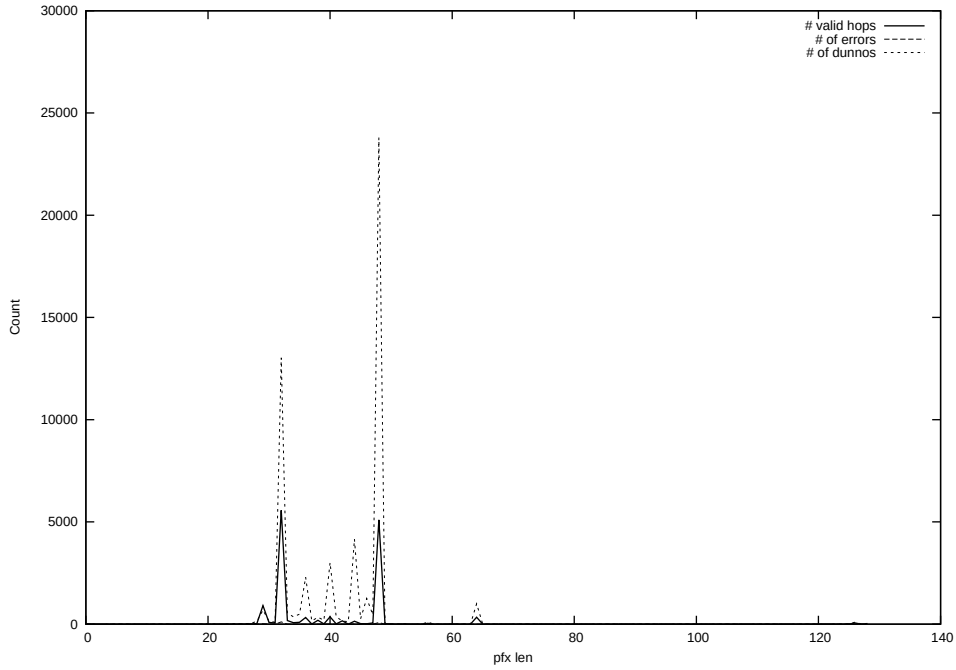
2015-01-01



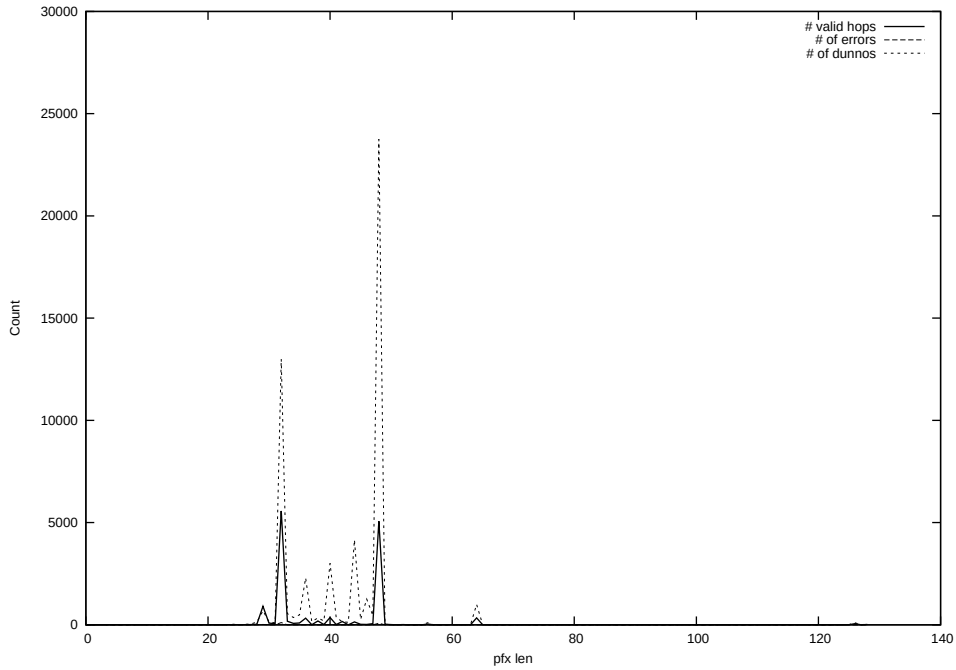
2015-01-02



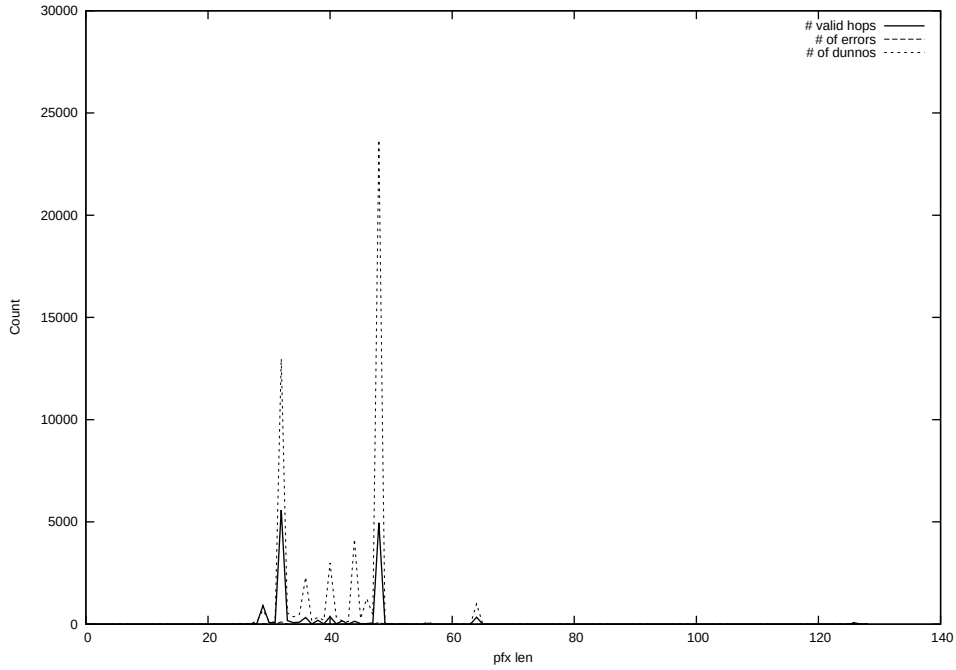
2015-01-03



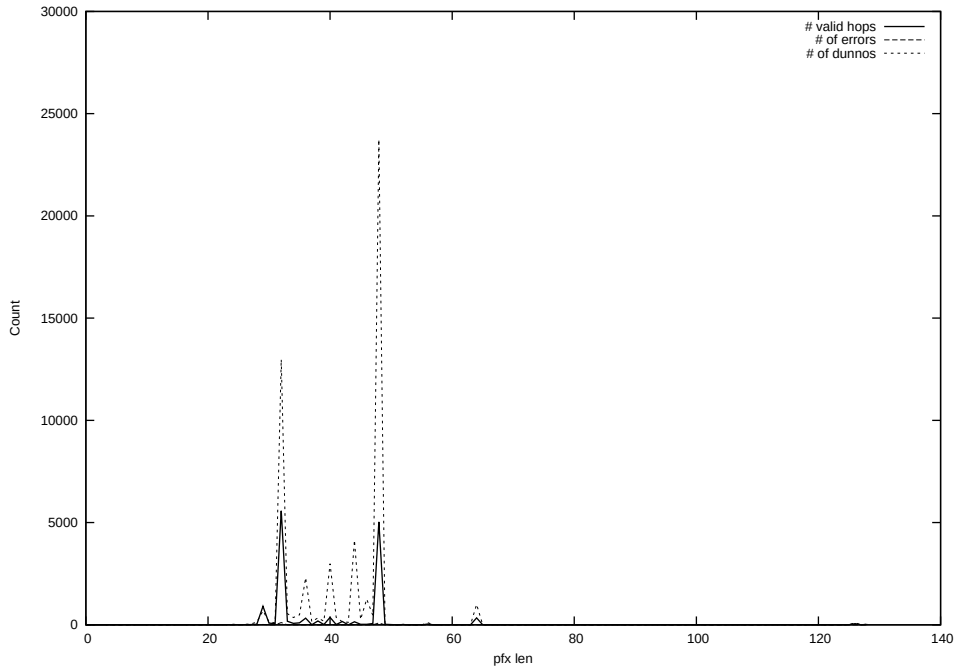
2015-01-04



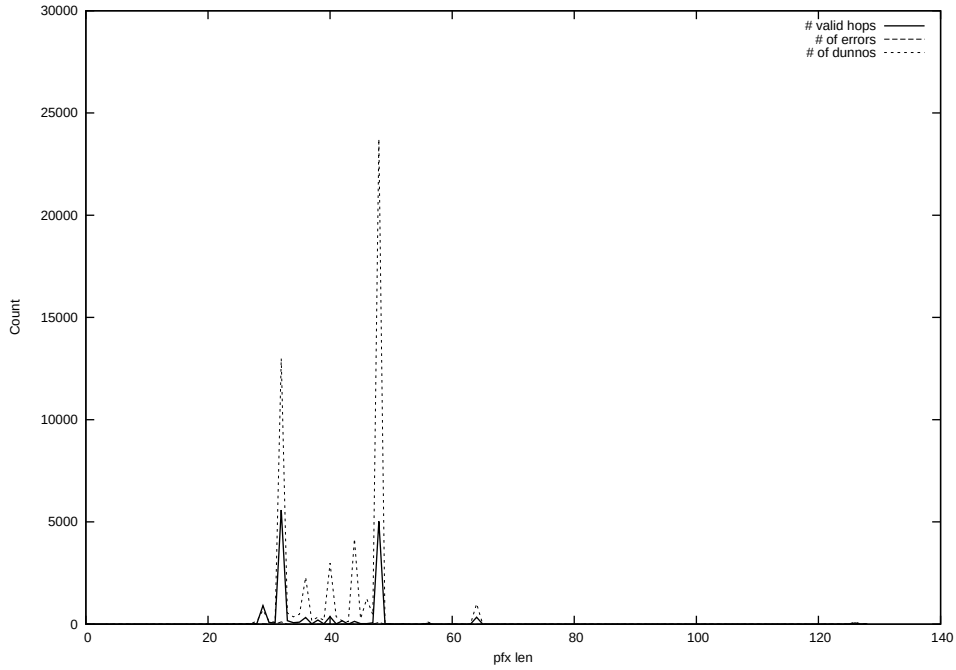
2015-01-05



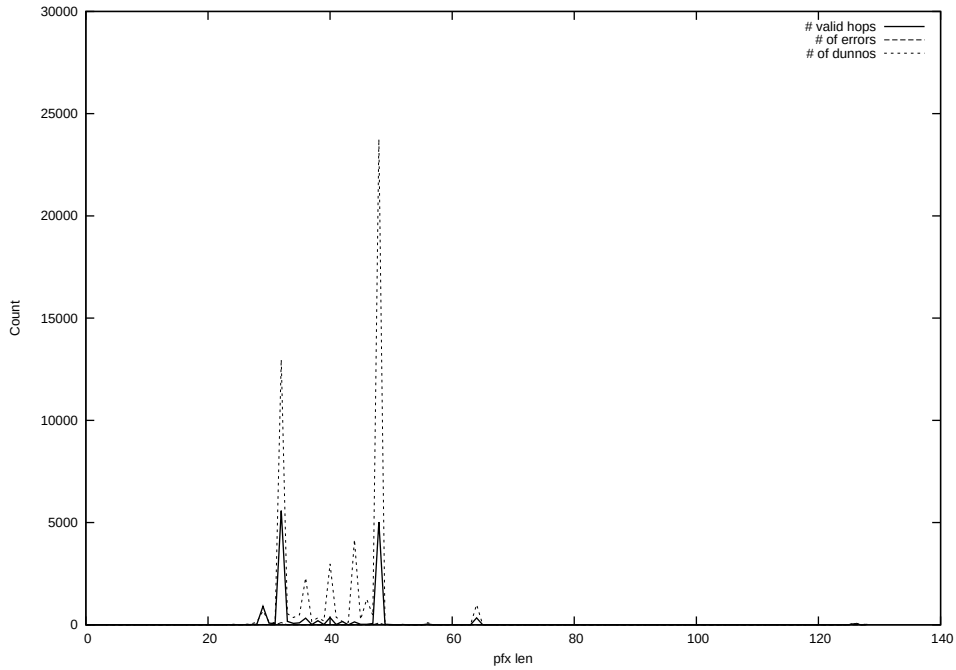
2015-01-06



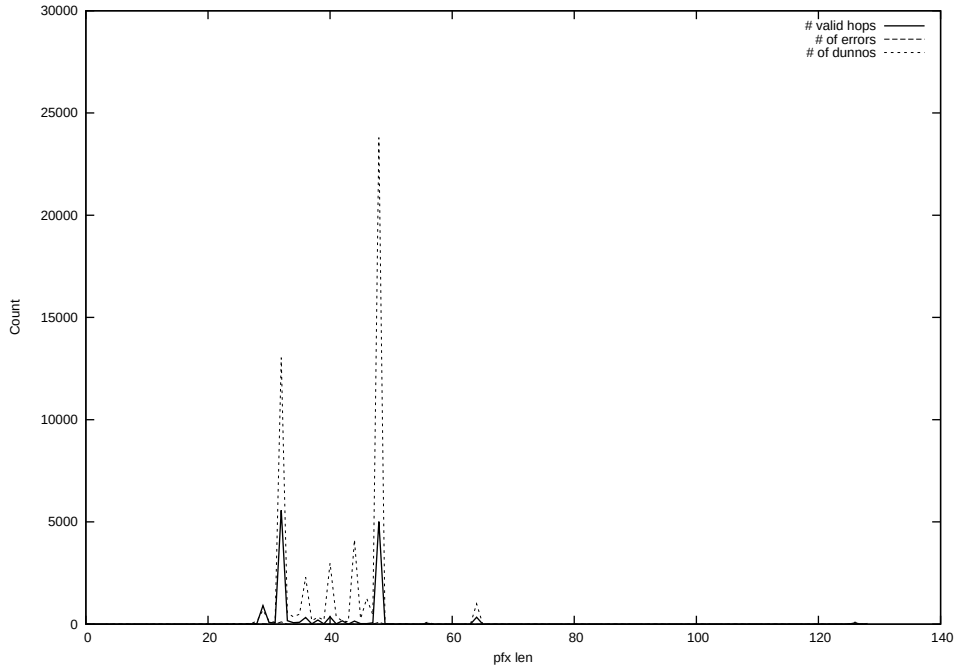
2015-01-07



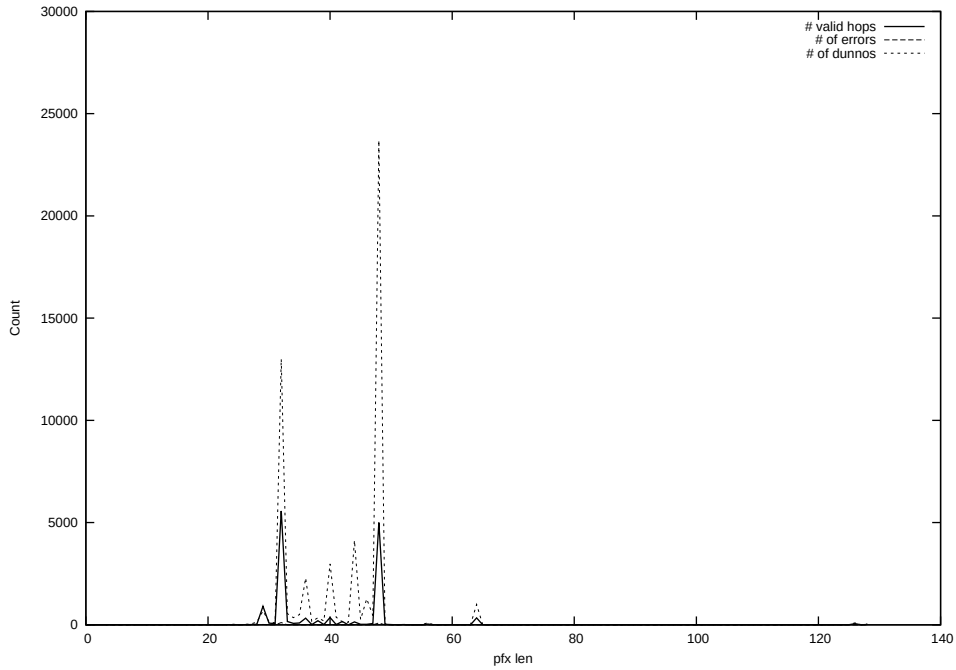
2015-01-08



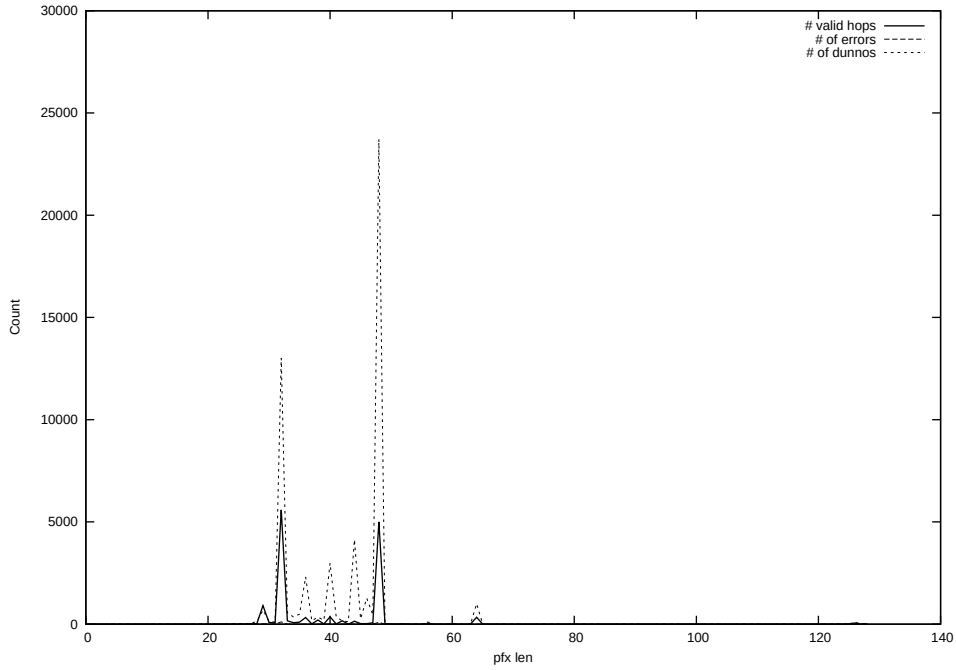
2015-01-09



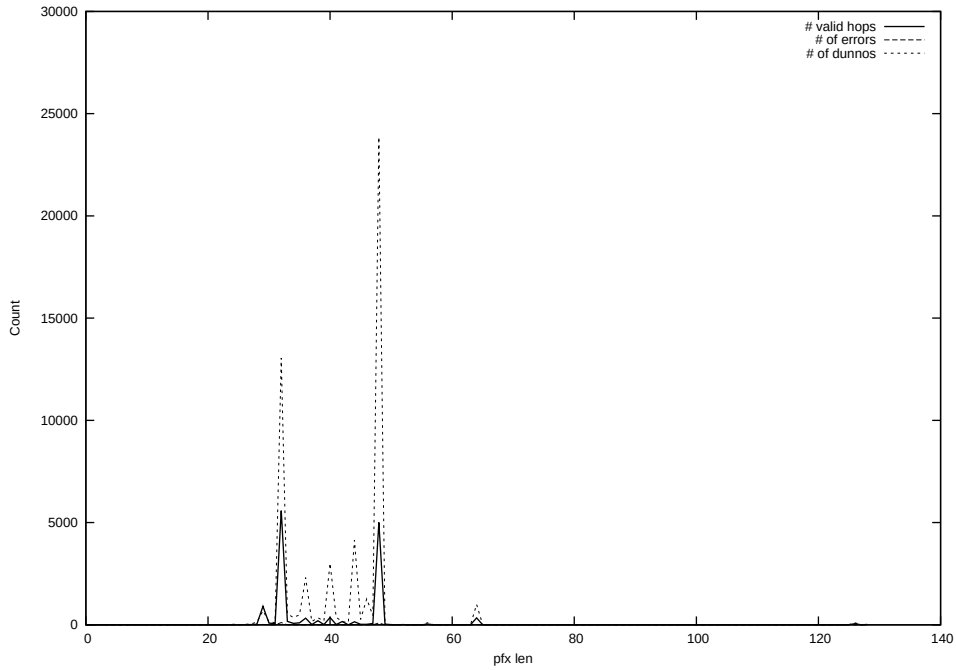
2015-01-10



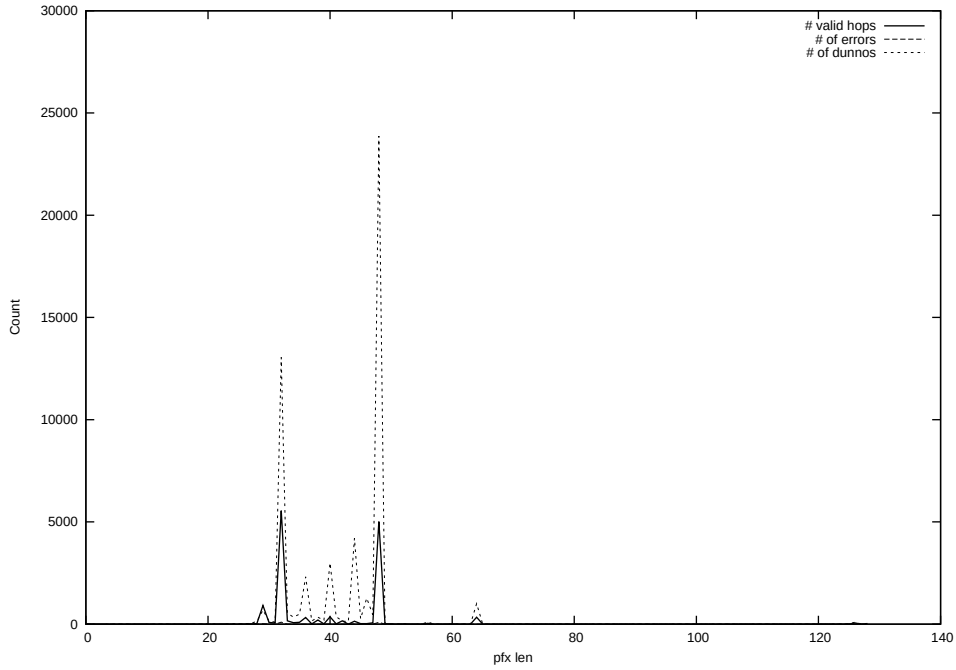
2015-01-11



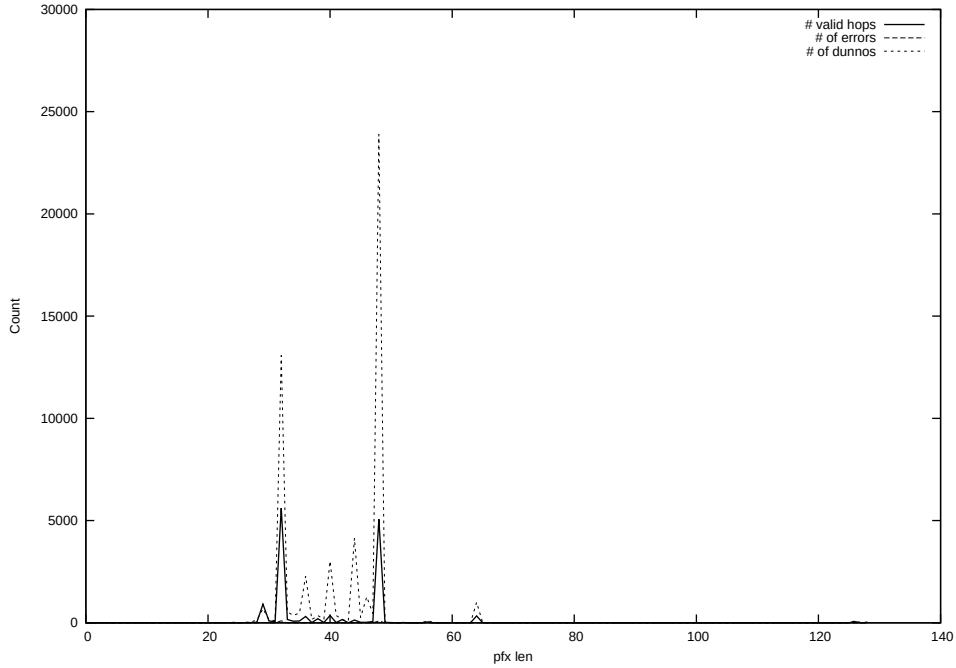
2015-01-12



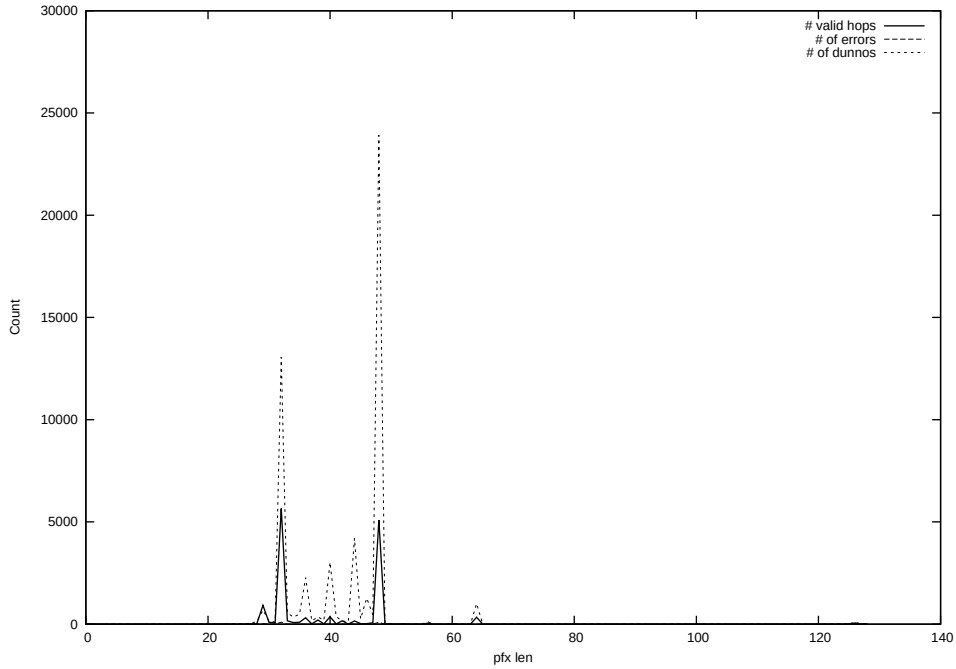
2015-01-13



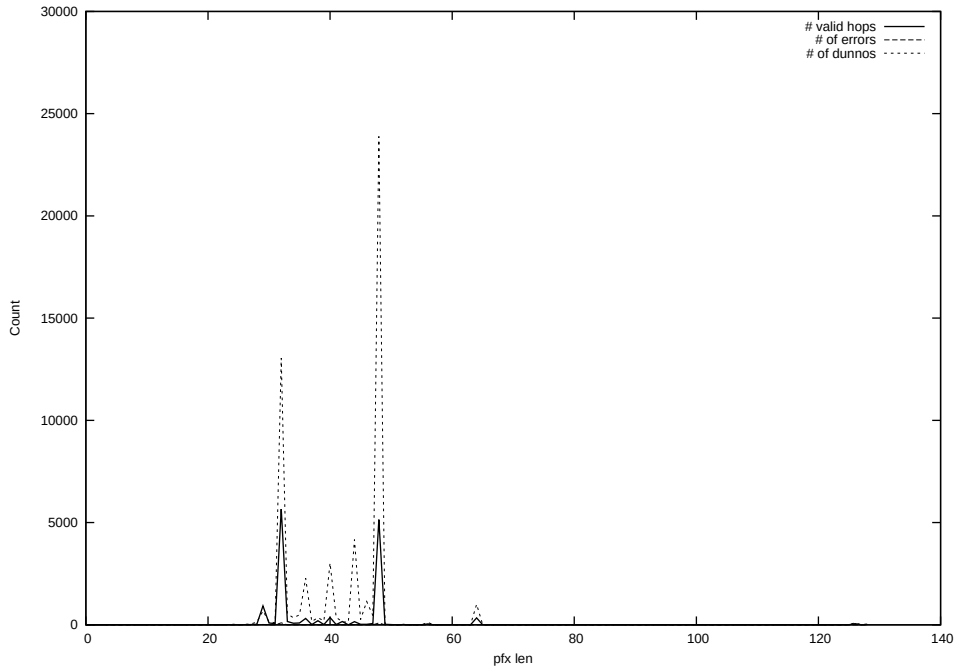
2015-01-14



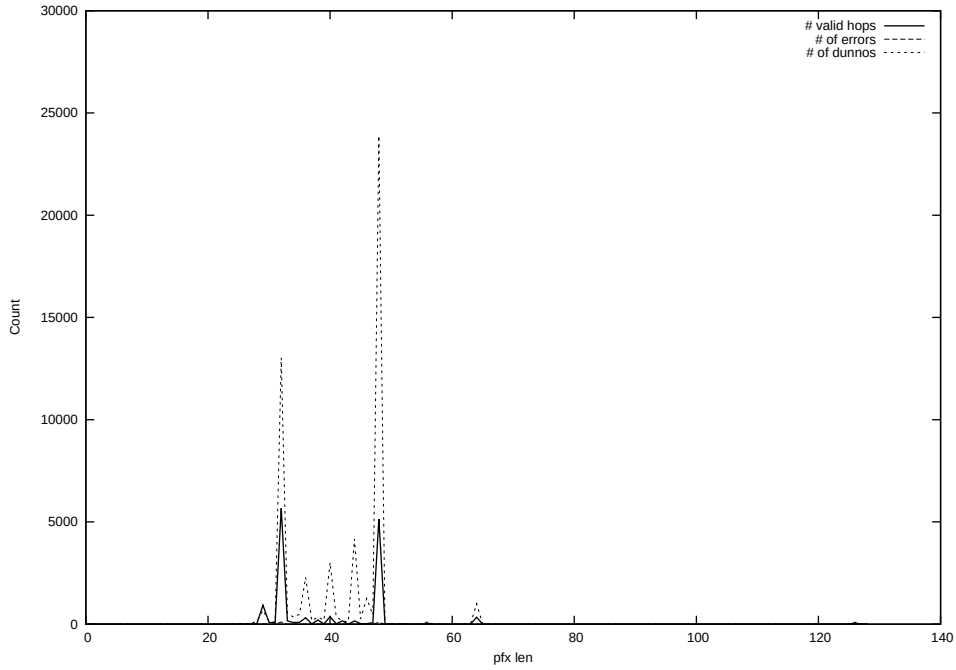
2015-01-15



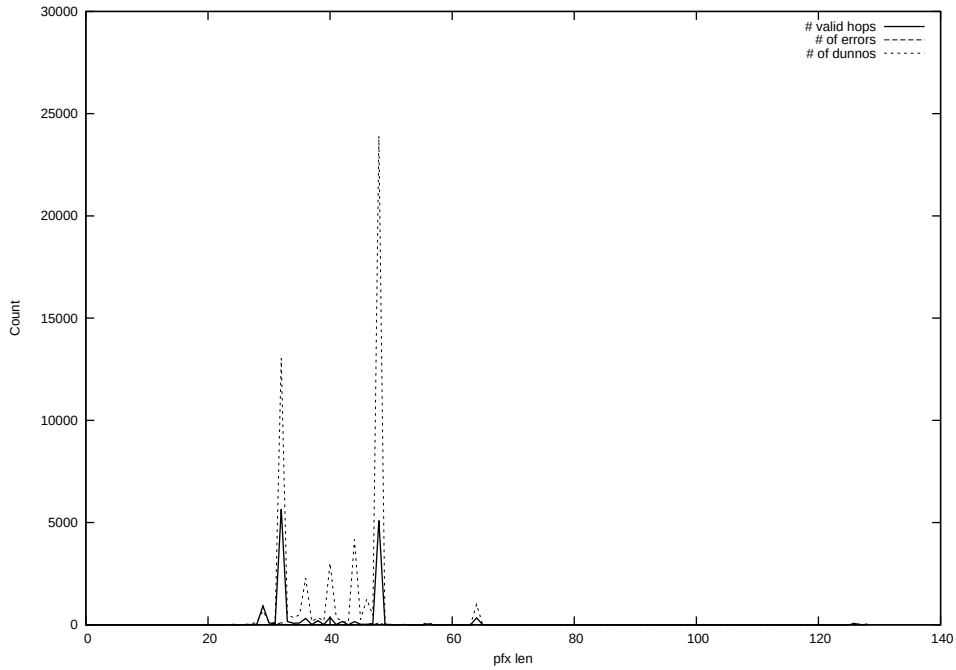
2015-01-16



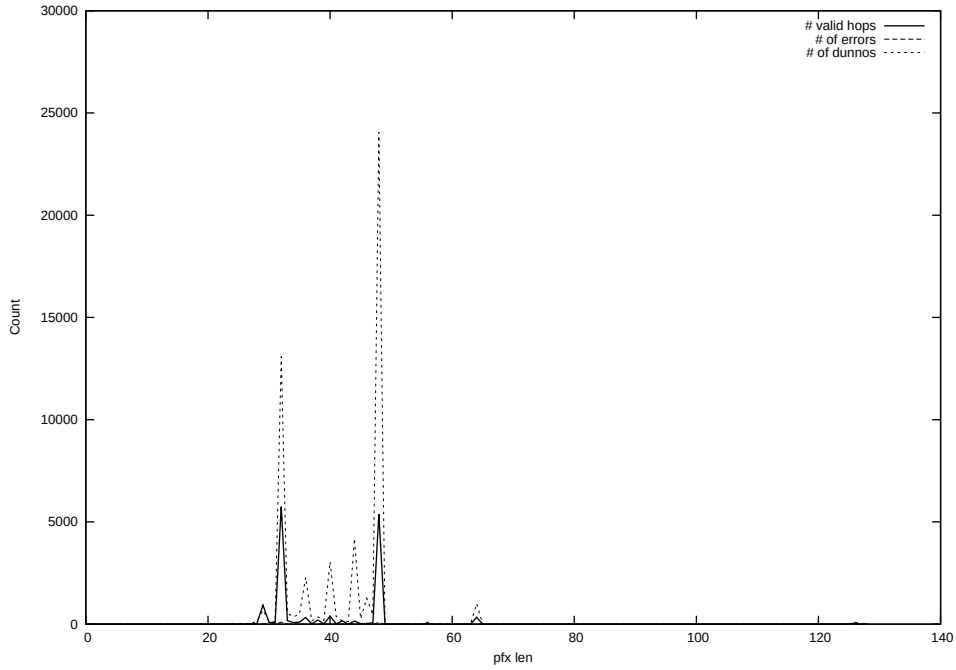
2015-01-17



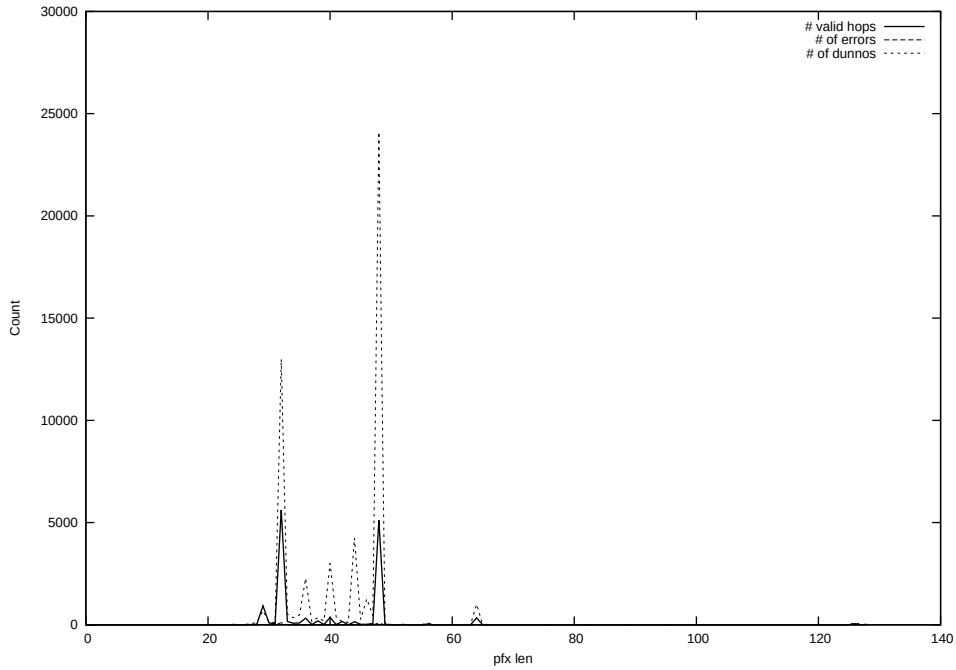
2015-01-18



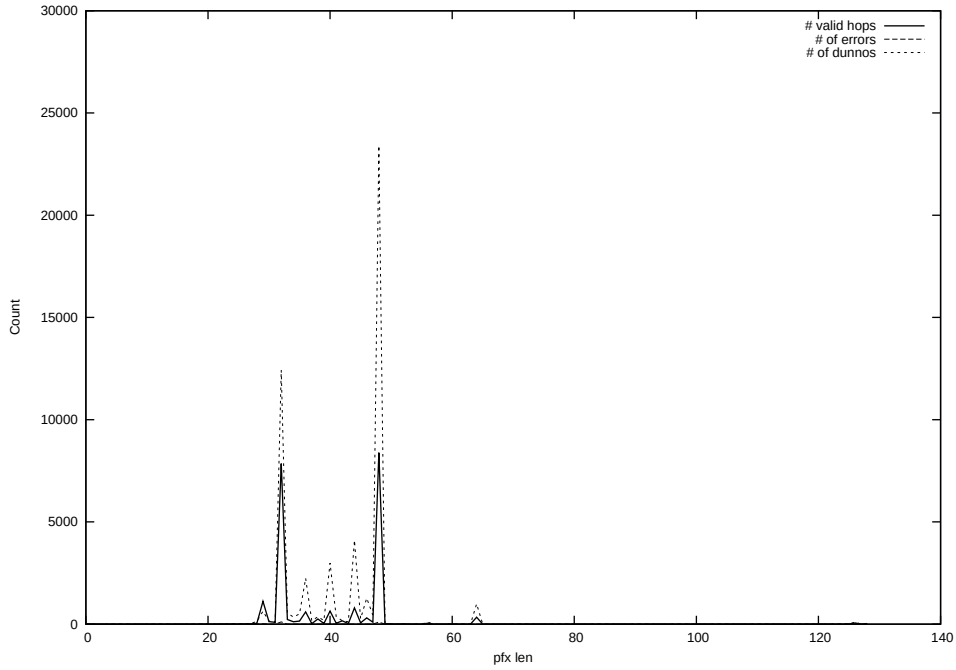
2015-01-19



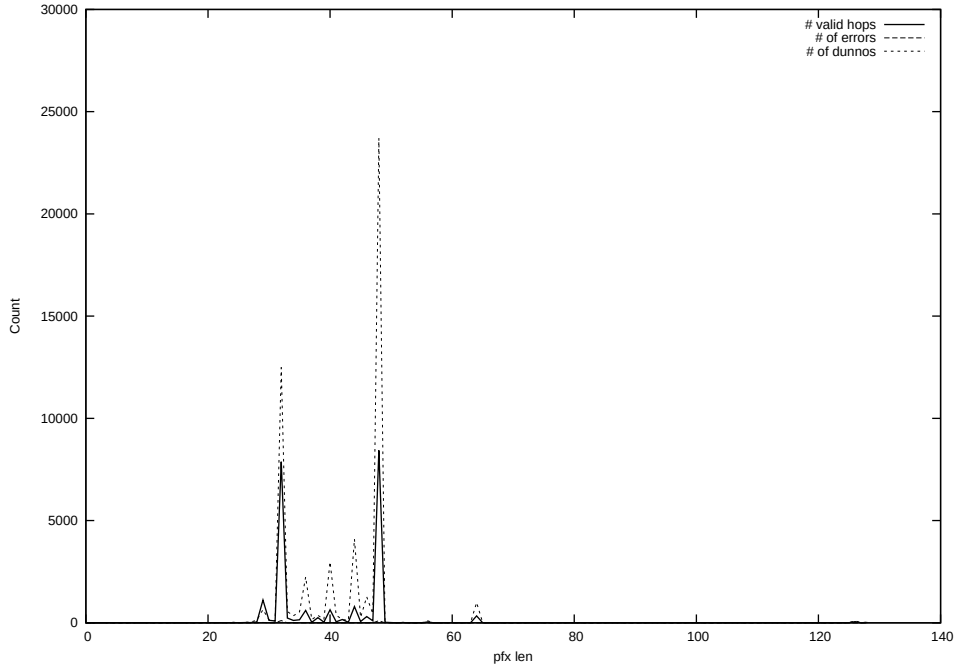
2015-01-20



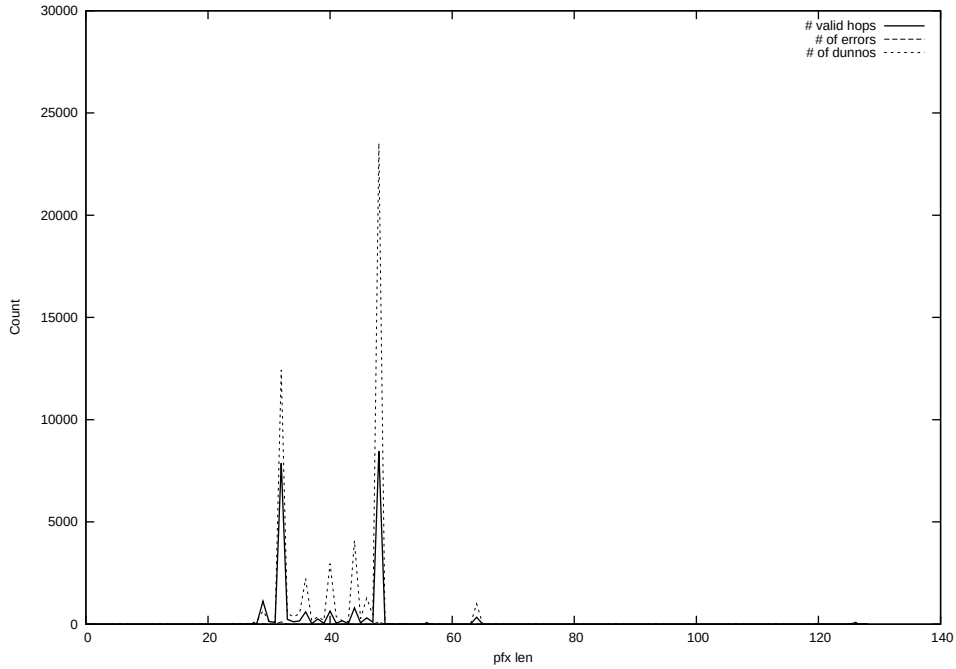
2015-01-21



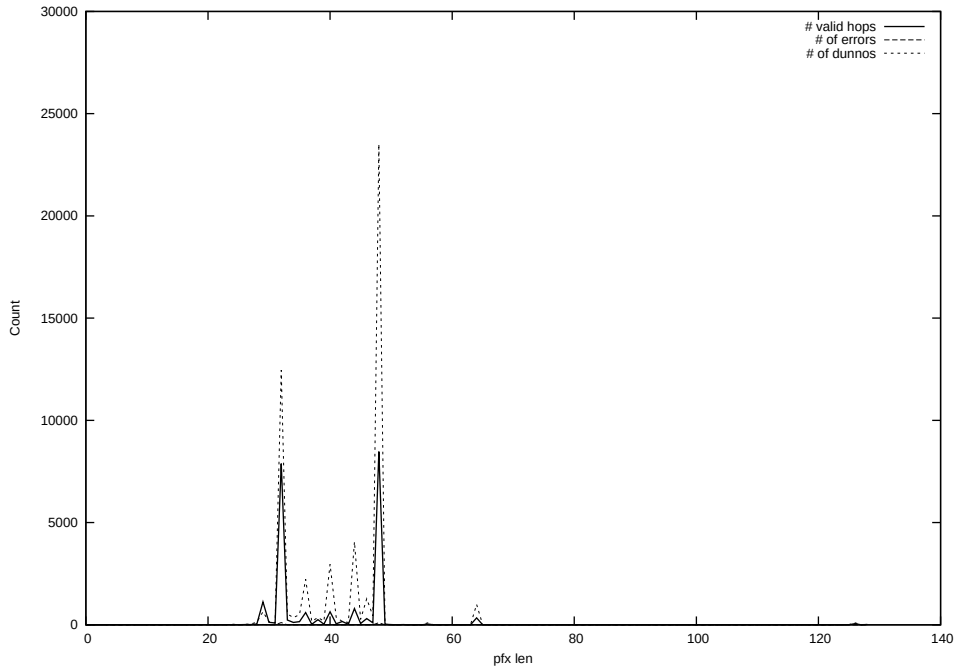
2015-01-22



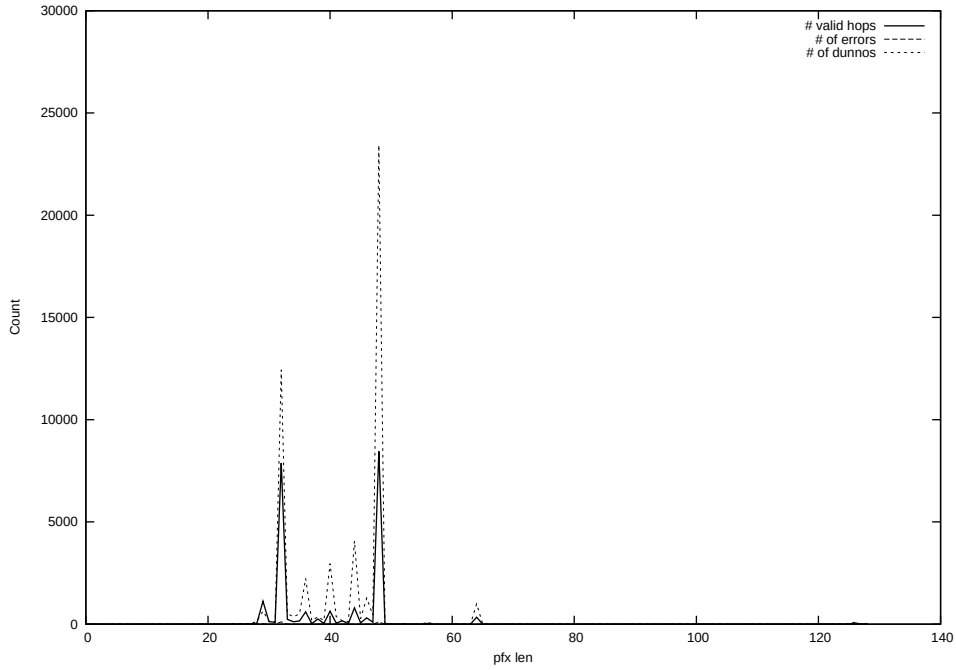
2015-01-23



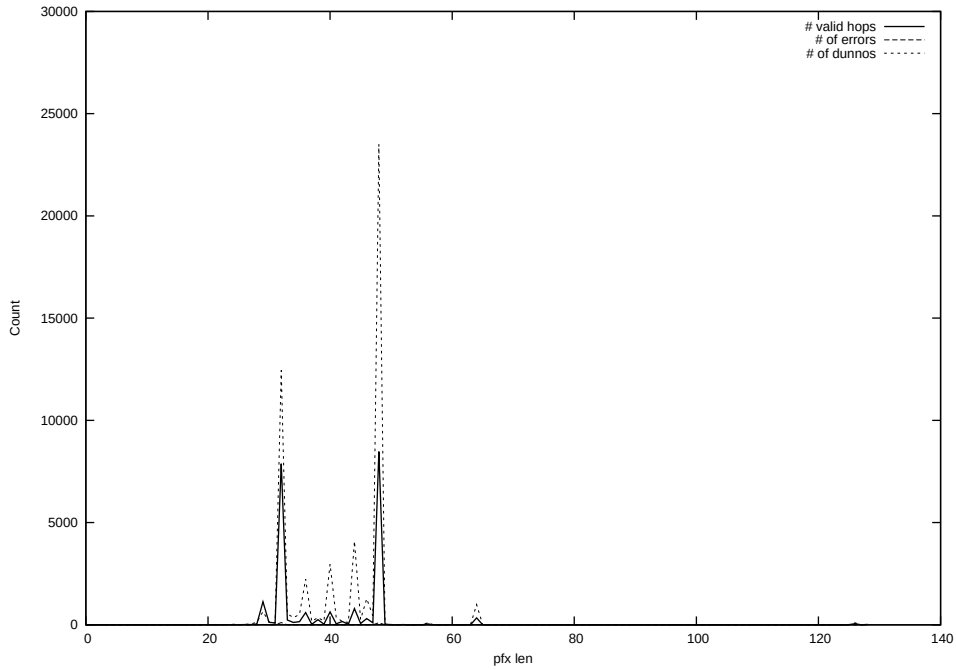
2015-01-24



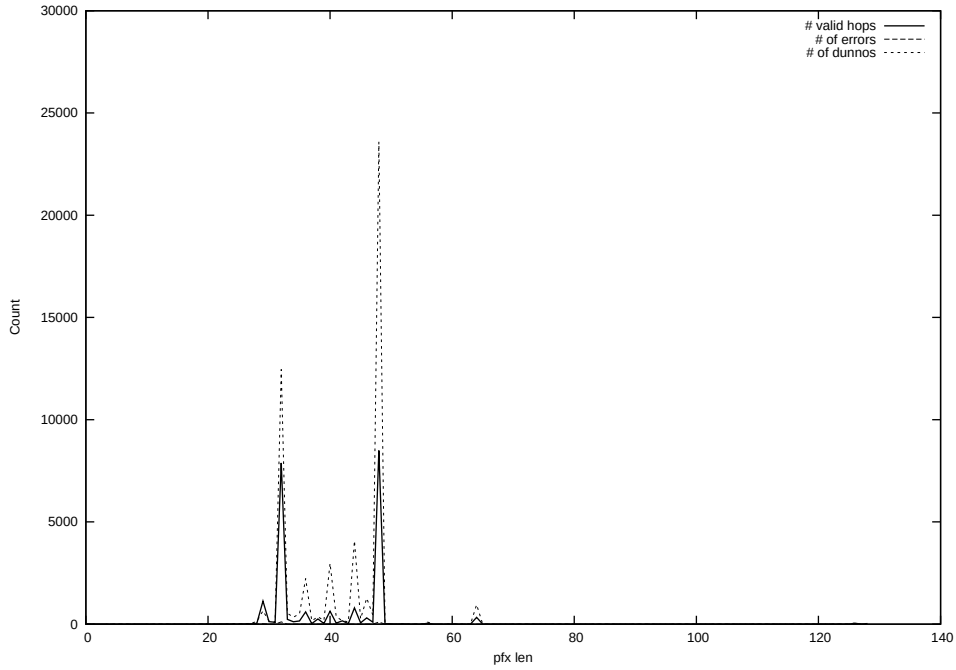
2015-01-25



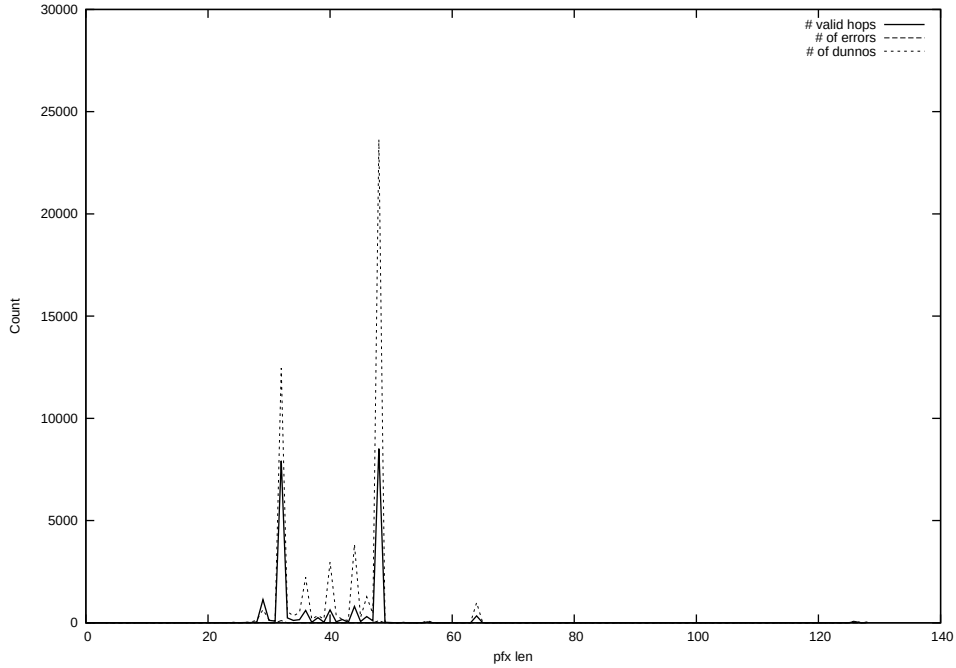
2015-01-26



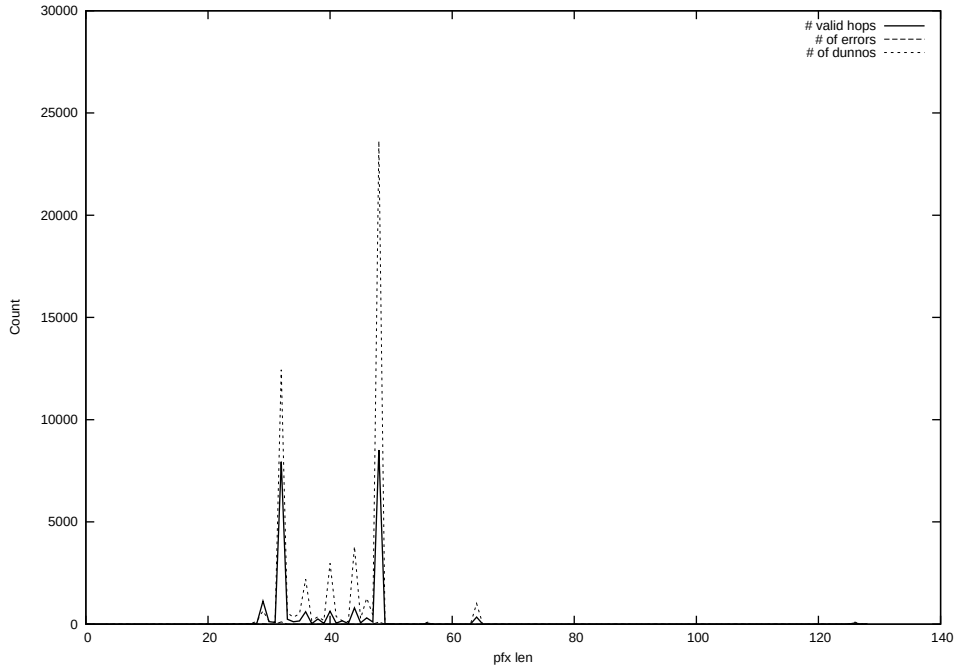
2015-01-27



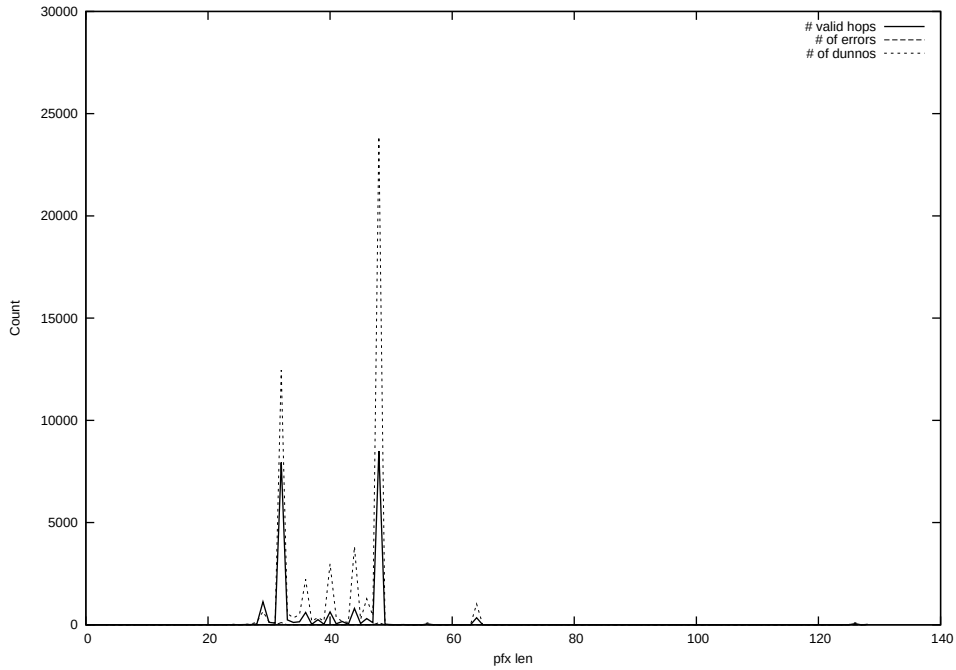
2015-01-28



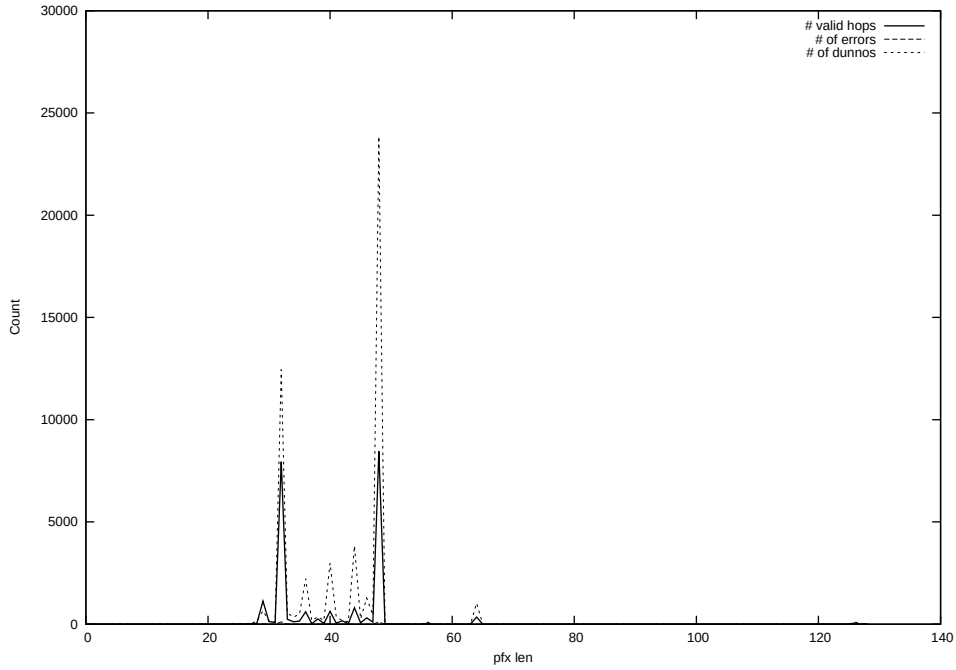
2015-01-29



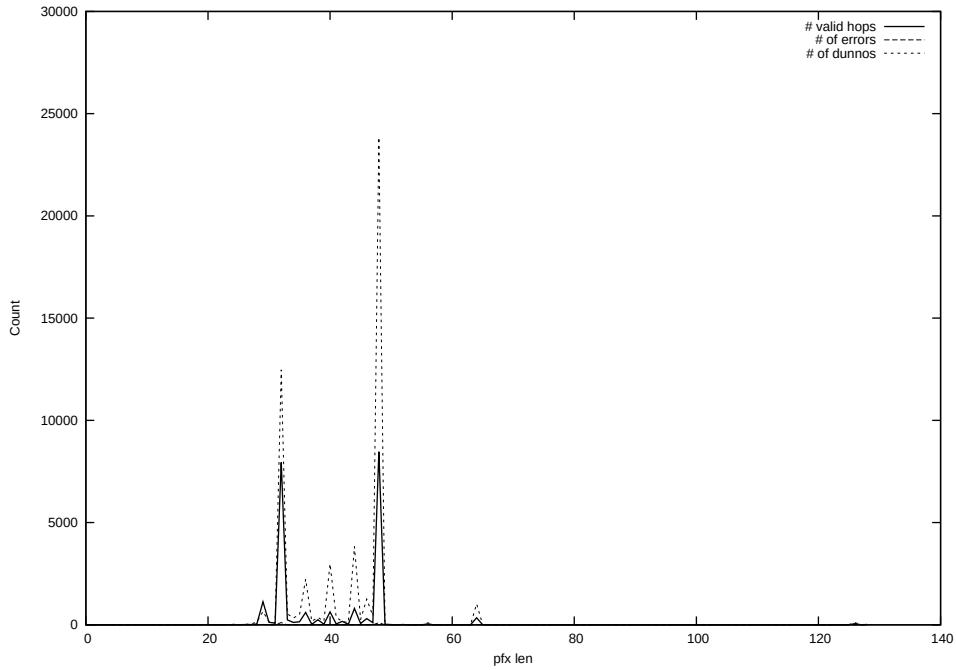
2015-01-30



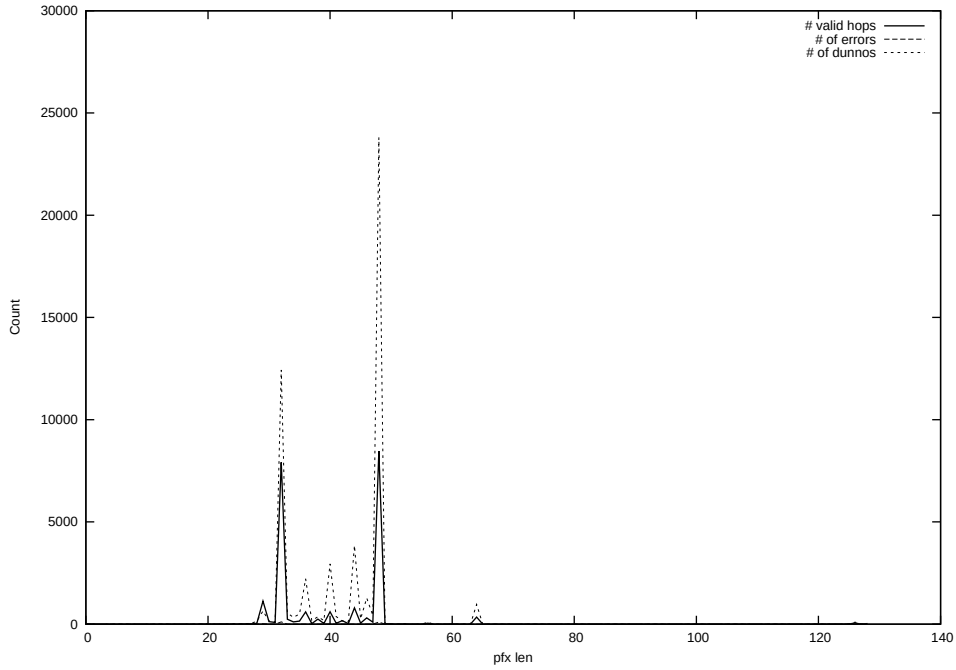
2015-01-31



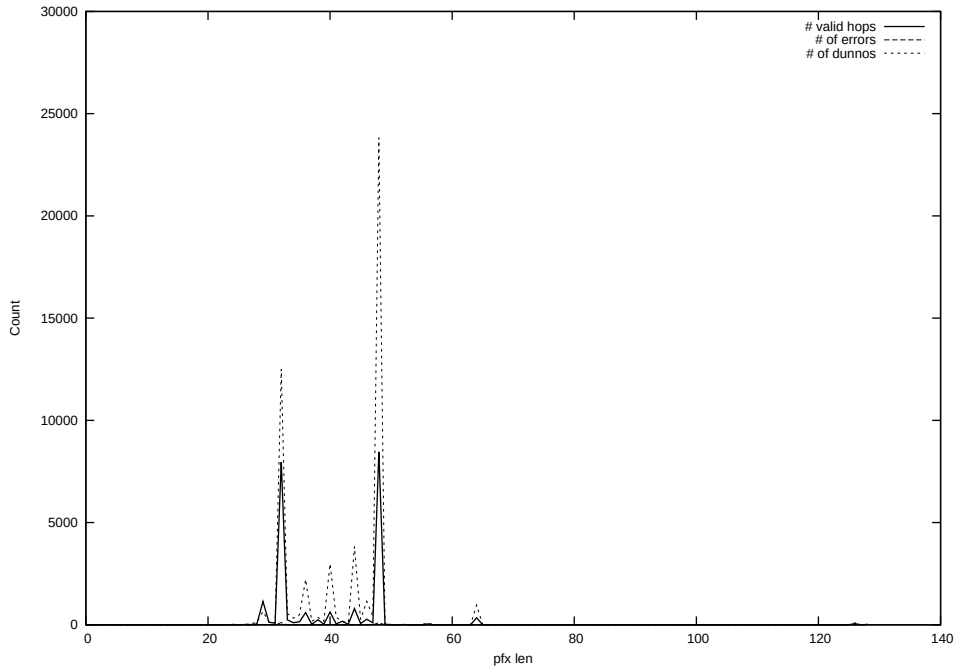
2015-02-01



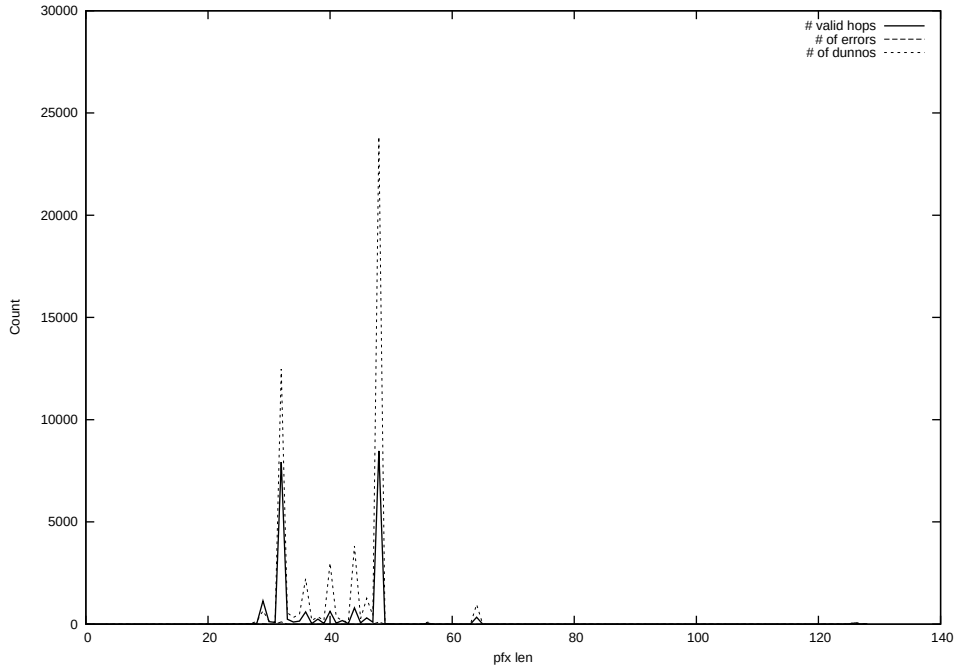
2015-02-02



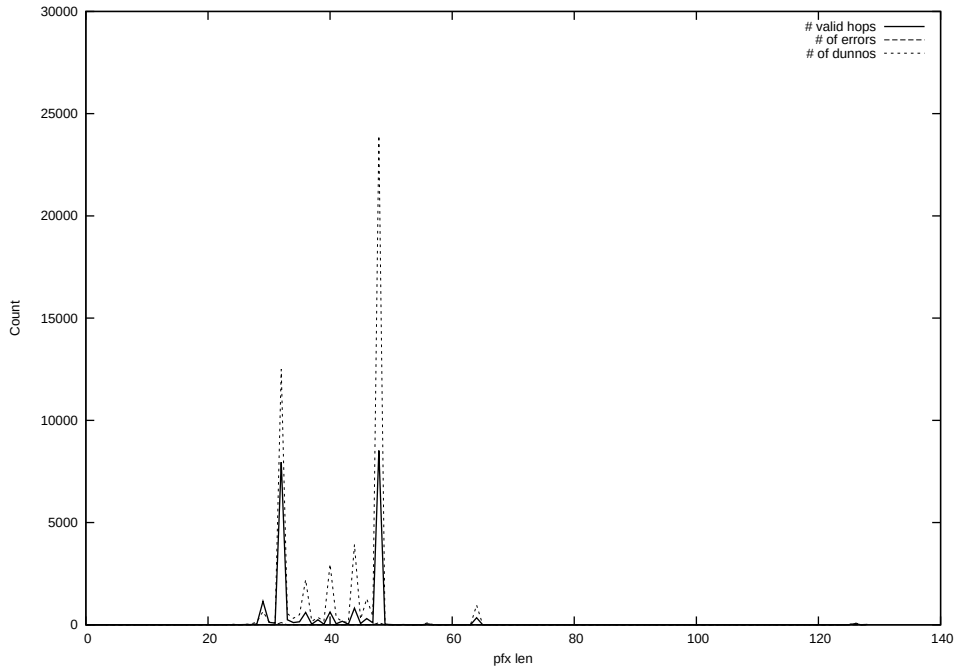
2015-02-03



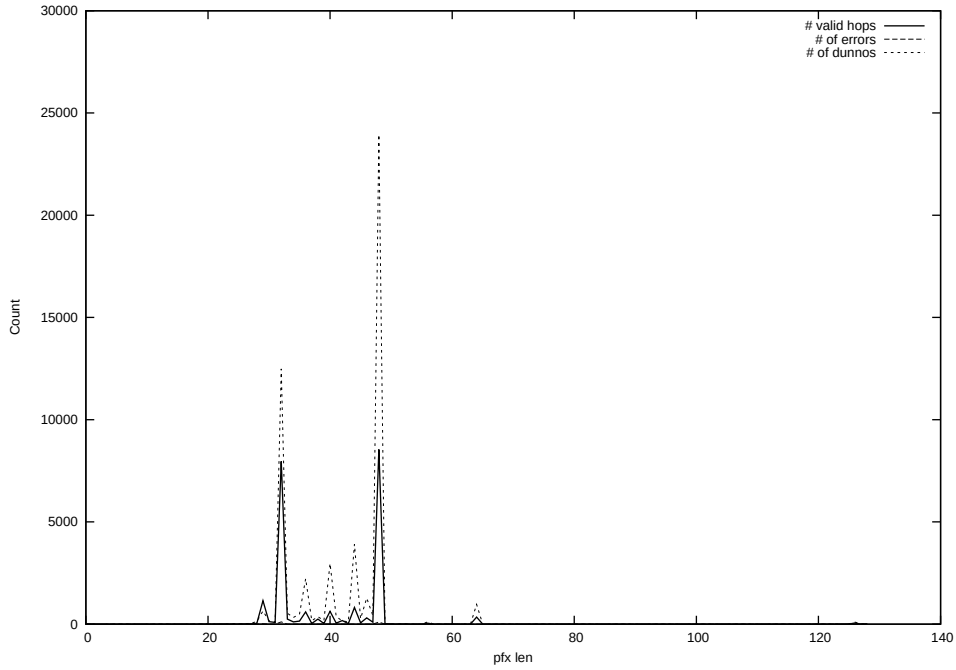
2015-02-04



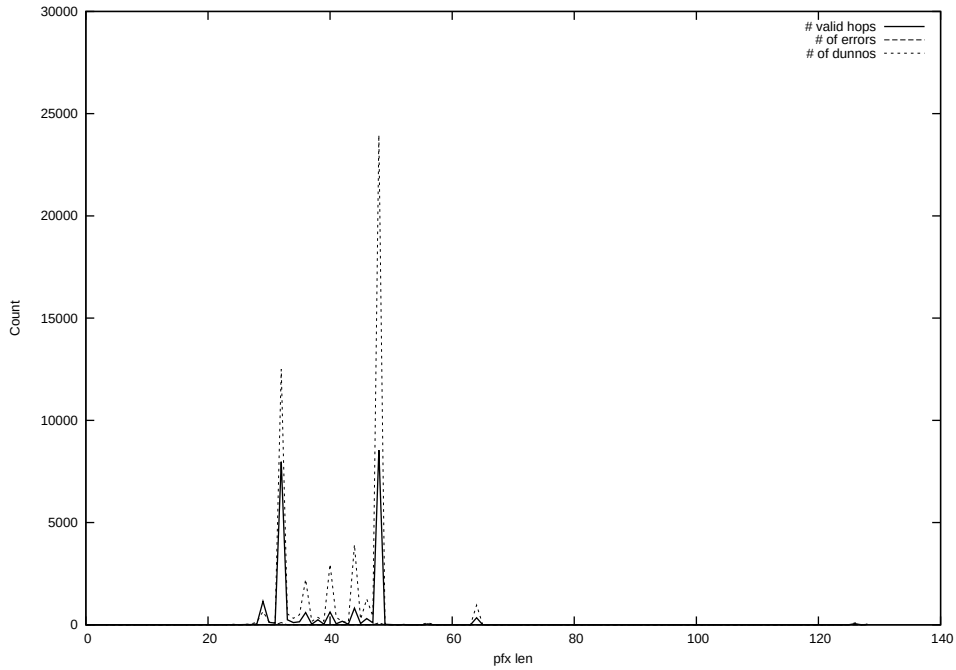
2015-02-05



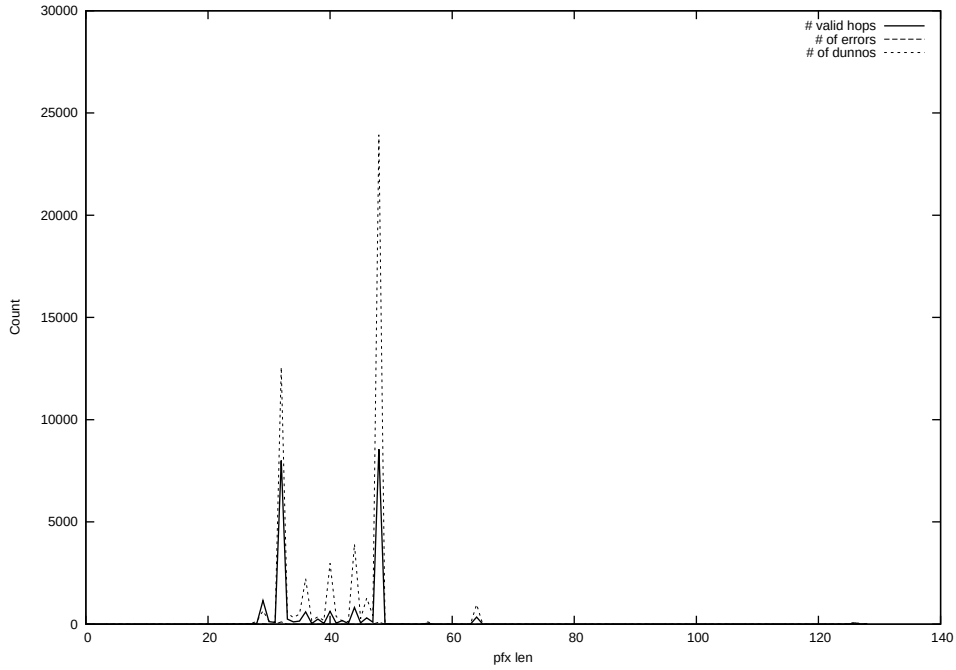
2015-02-06



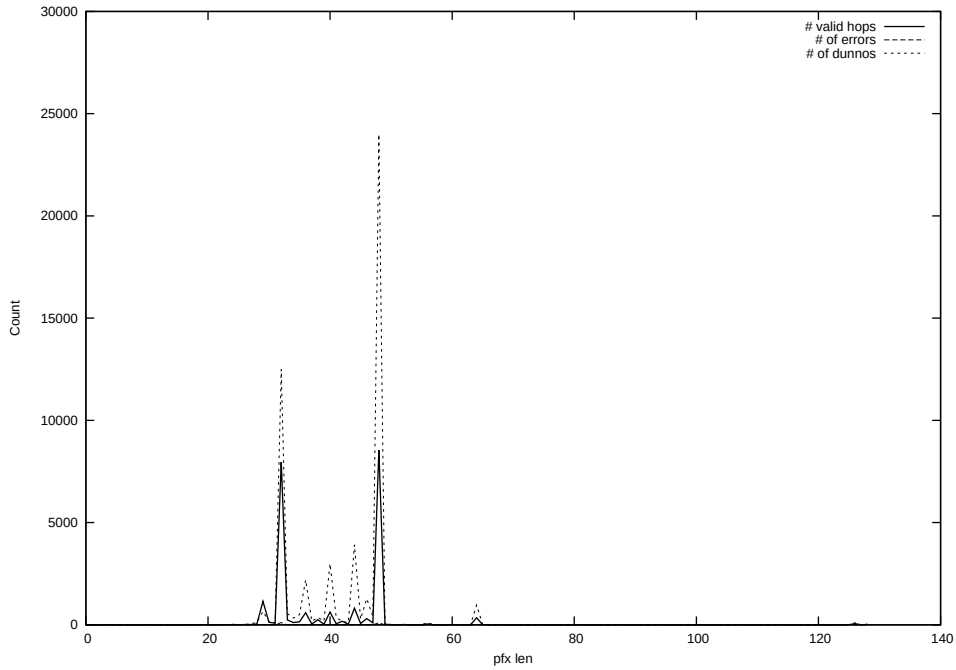
2015-02-07



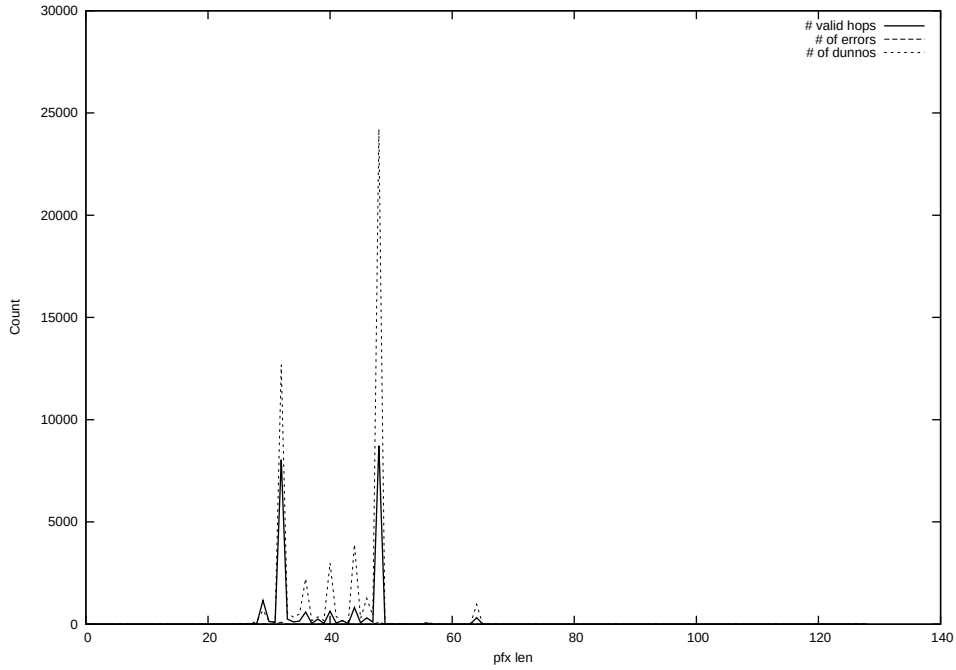
2015-02-08



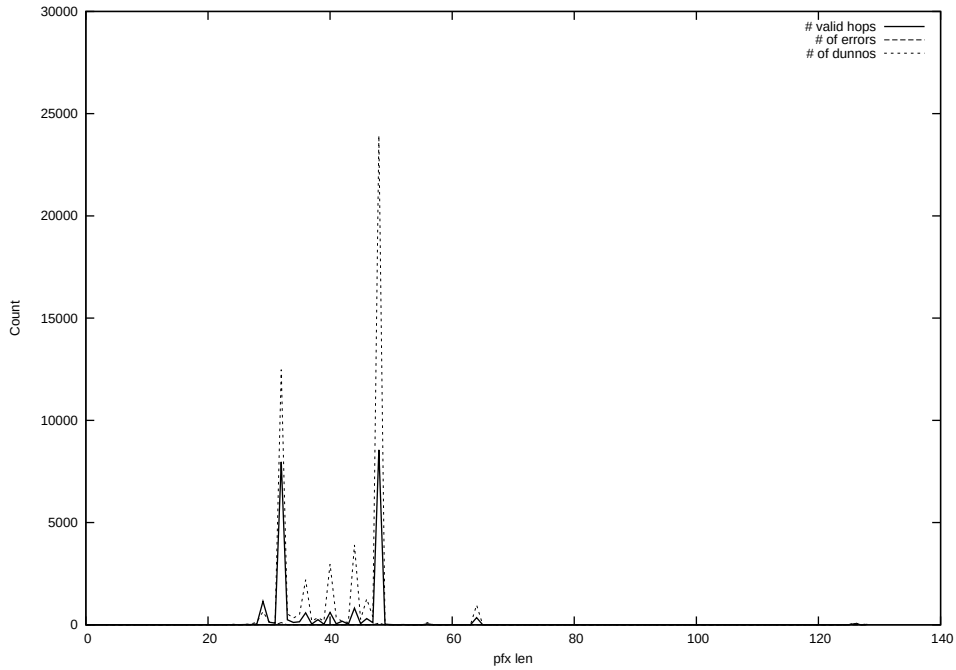
2015-02-09



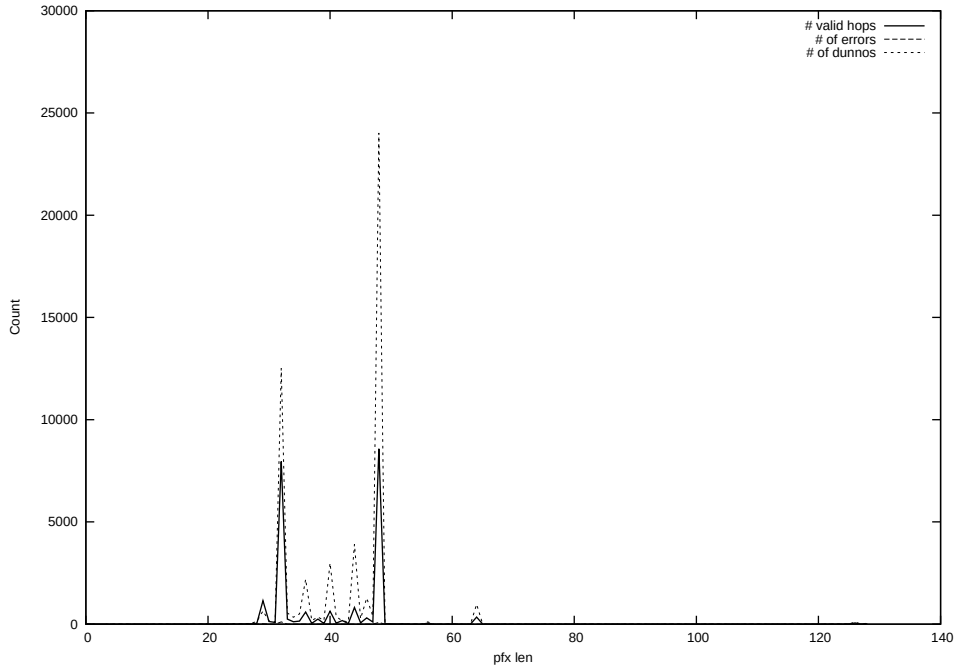
2015-02-10



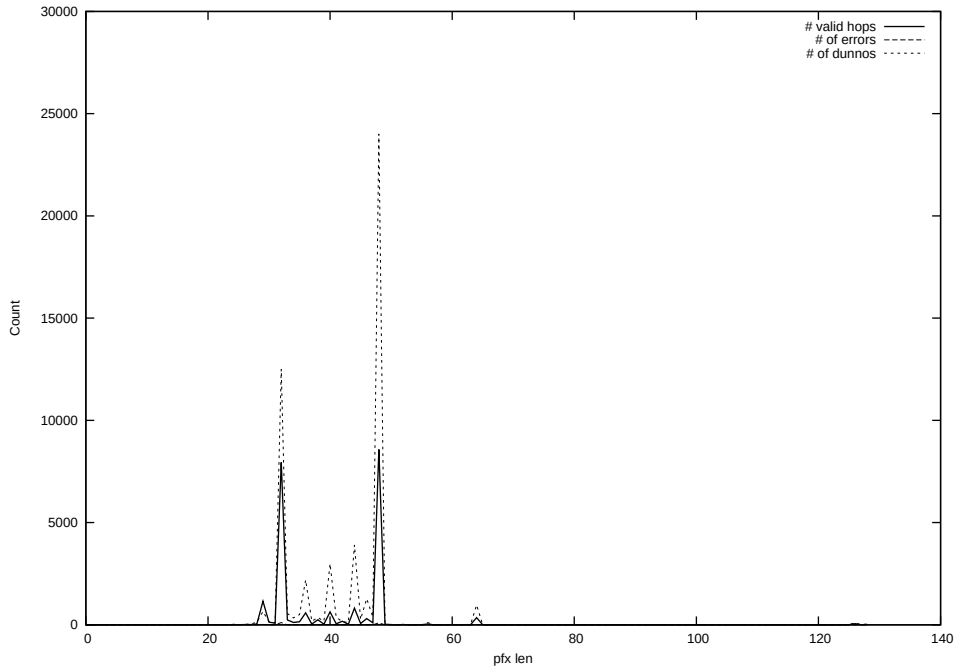
2015-02-11



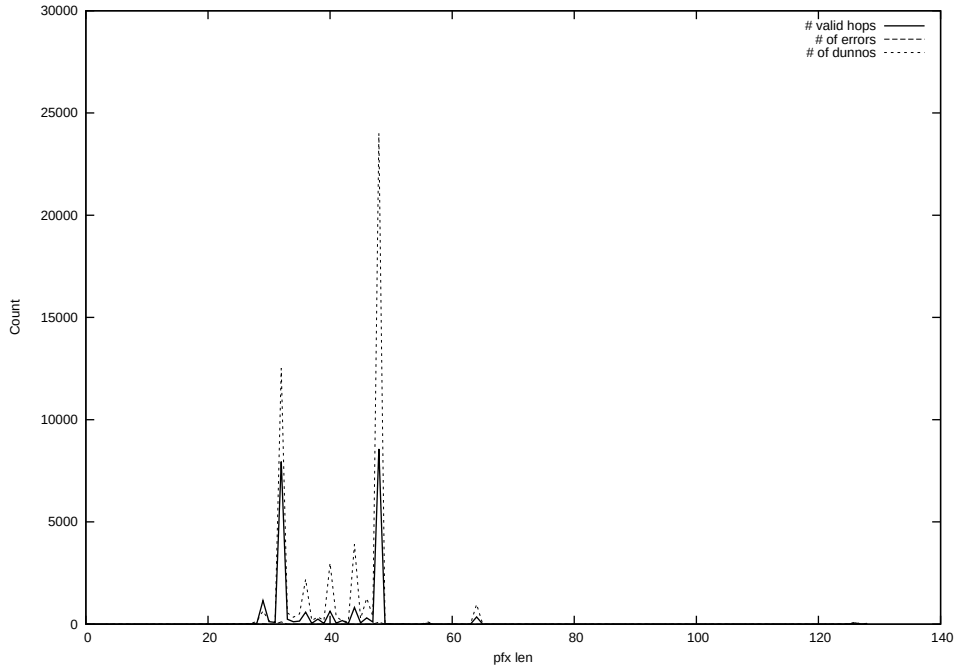
2015-02-12



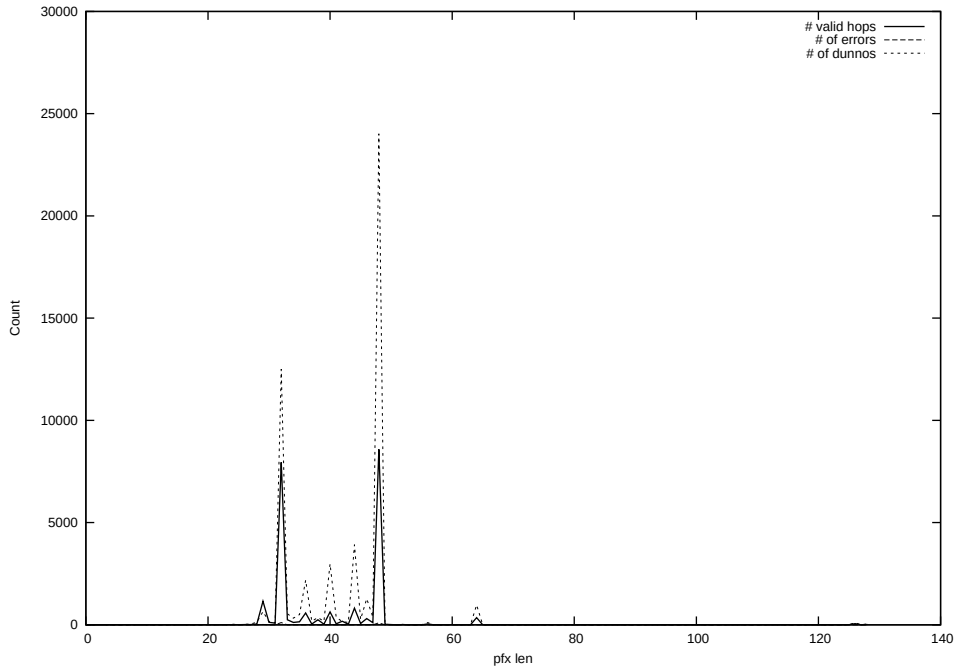
2015-02-13



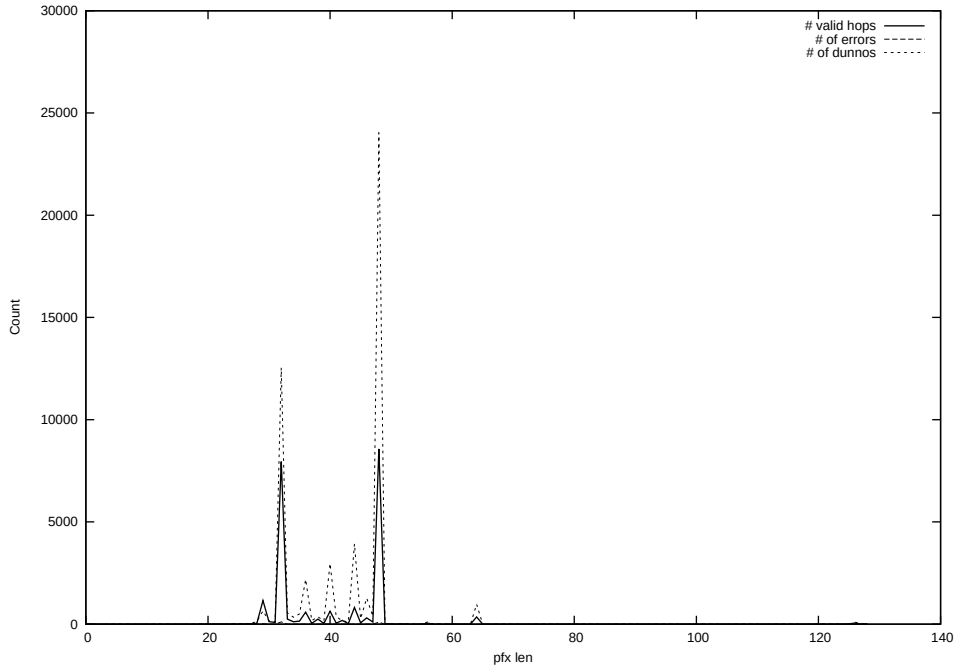
2015-02-14



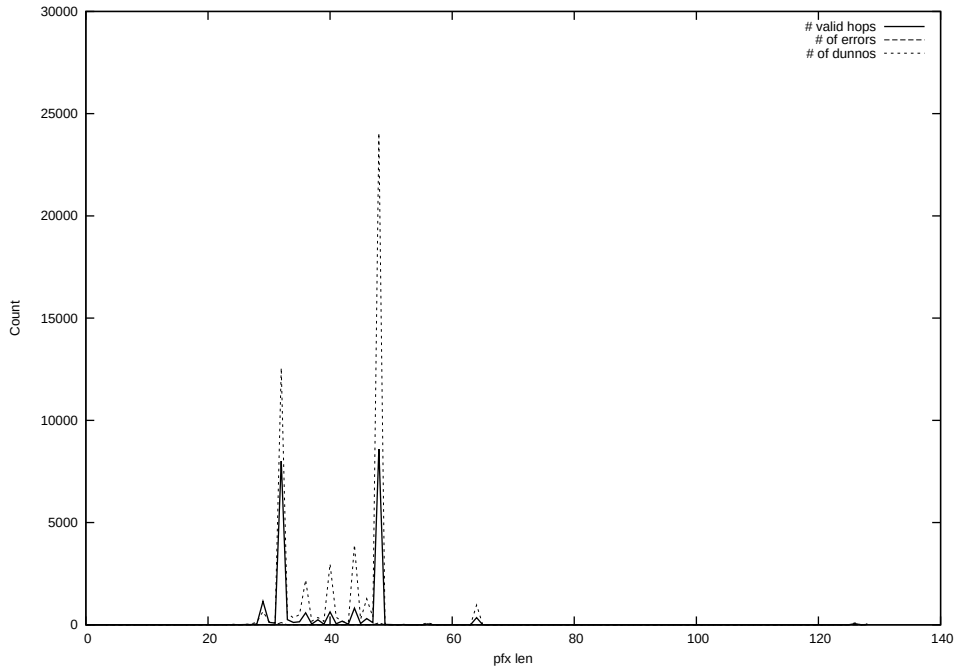
2015-02-15



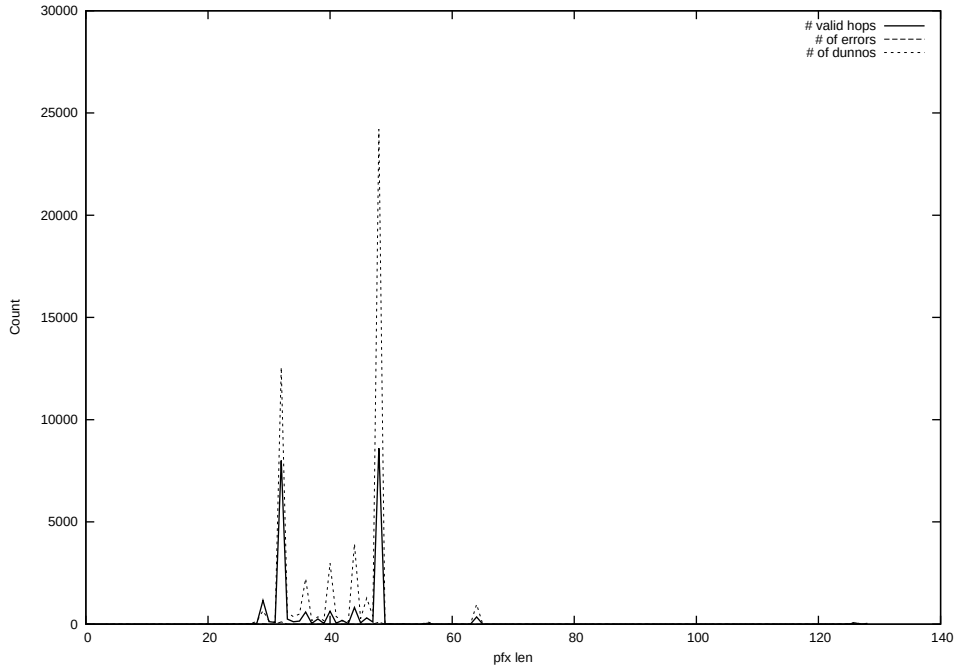
2015-02-16



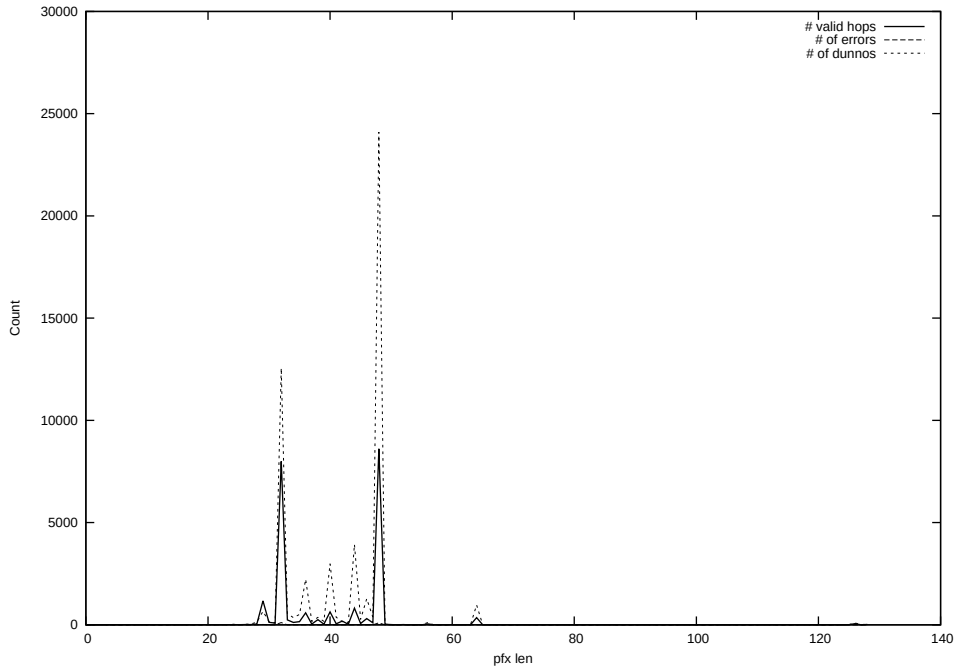
2015-02-17



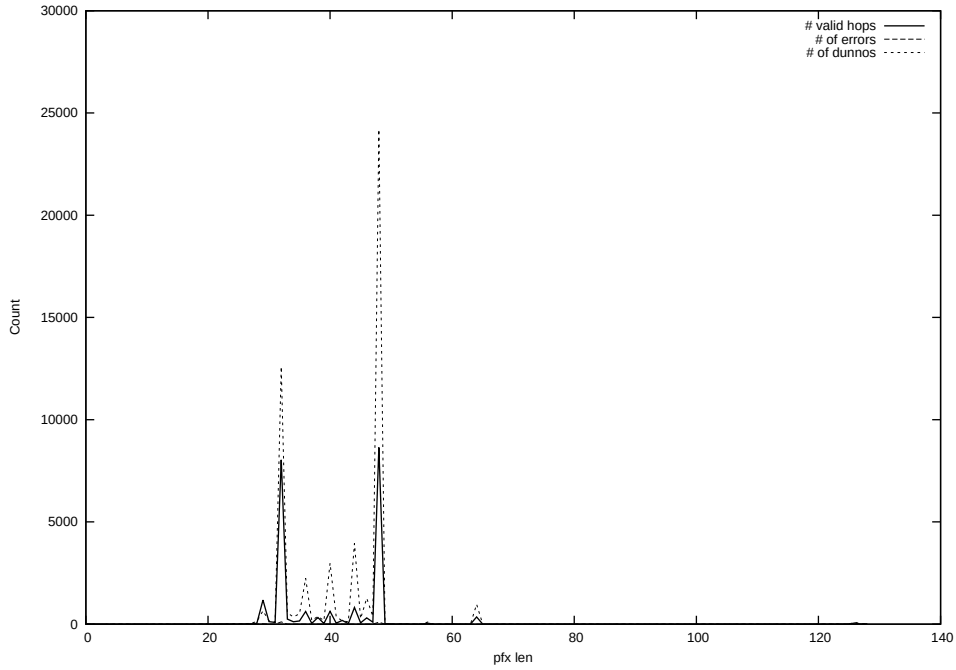
2015-02-18



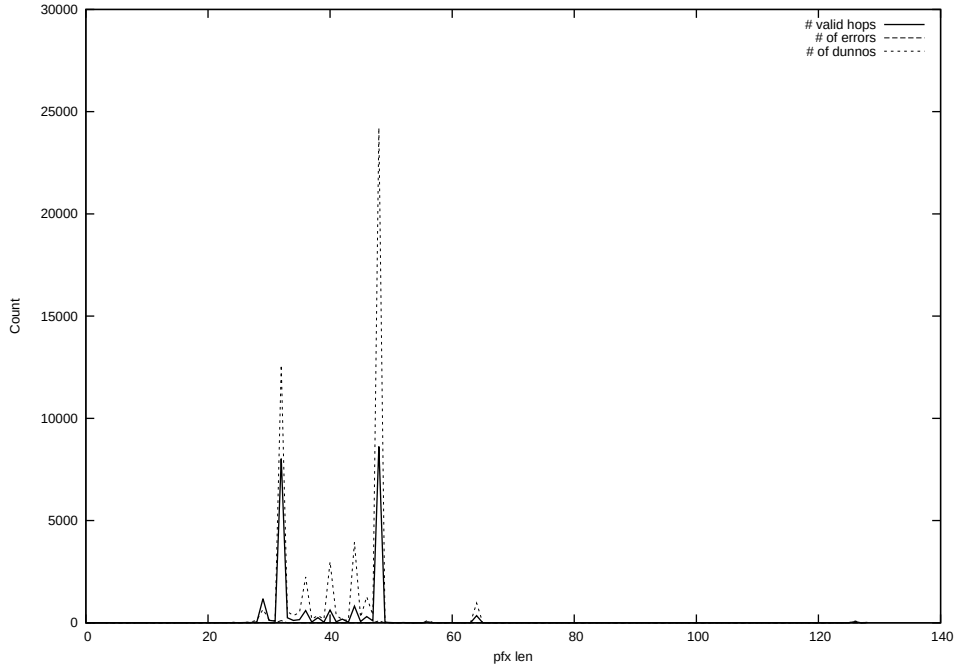
2015-02-19



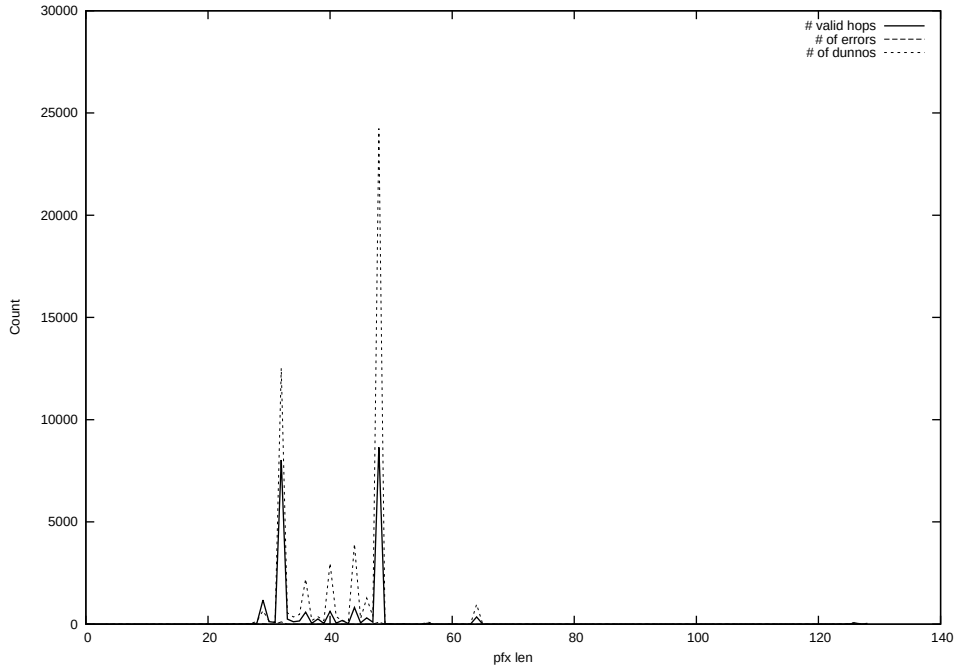
2015-02-20



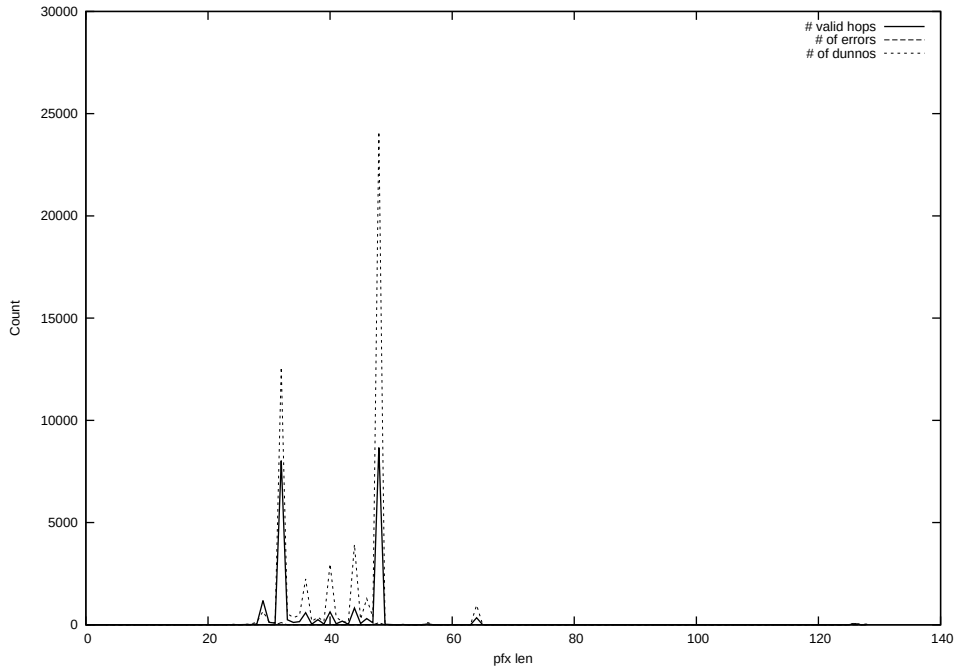
2015-02-21



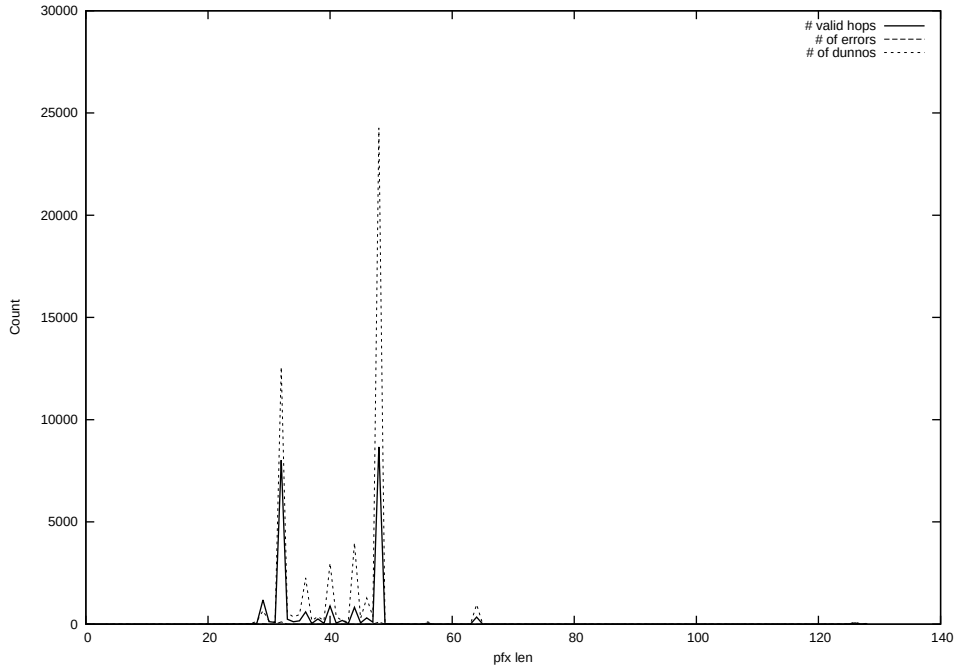
2015-02-22



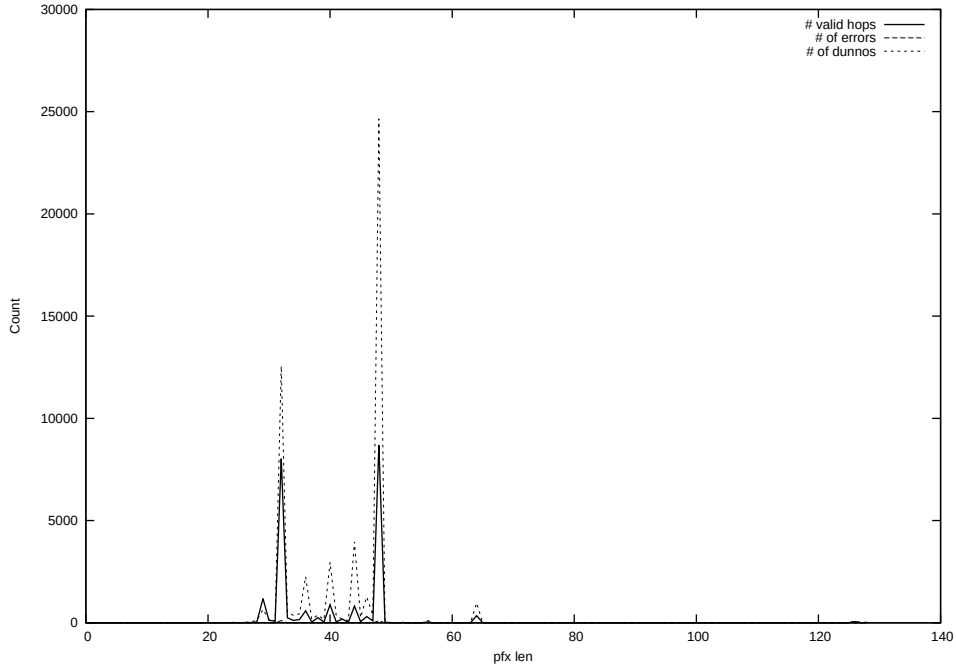
2015-02-23



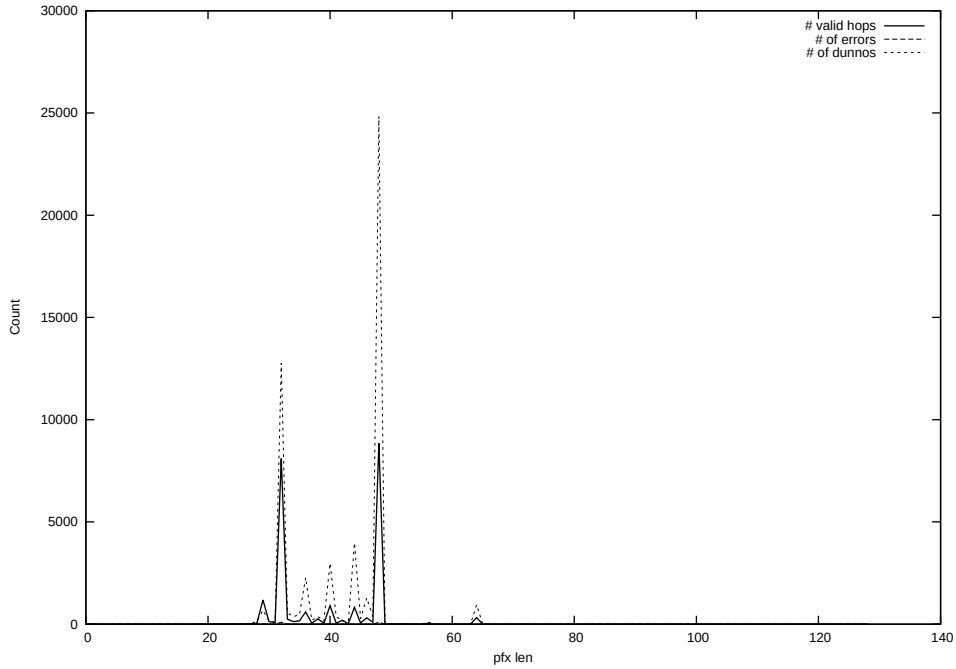
2015-02-24



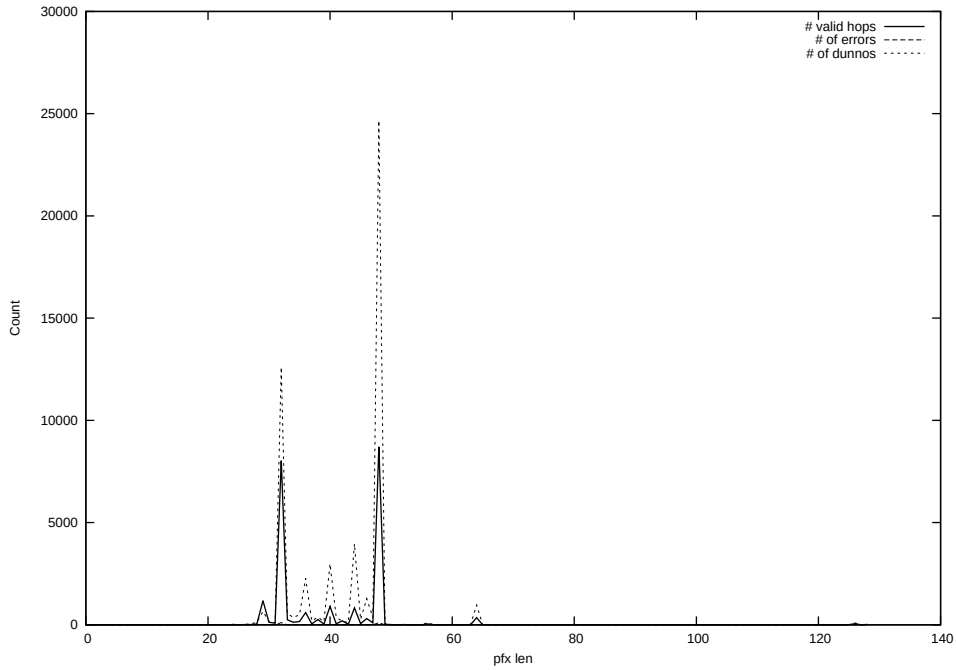
2015-02-25



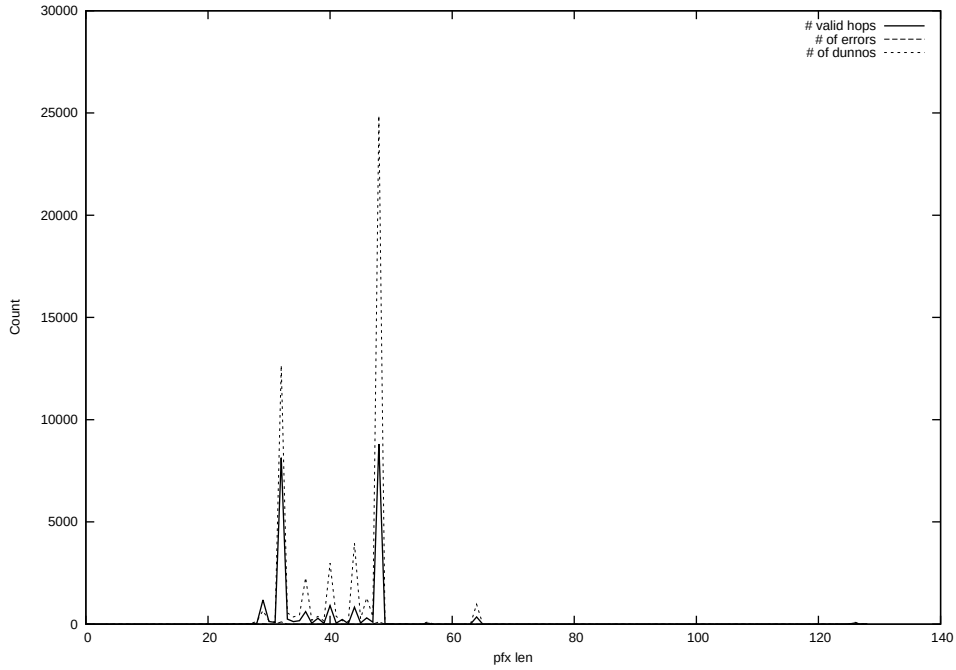
2015-02-26



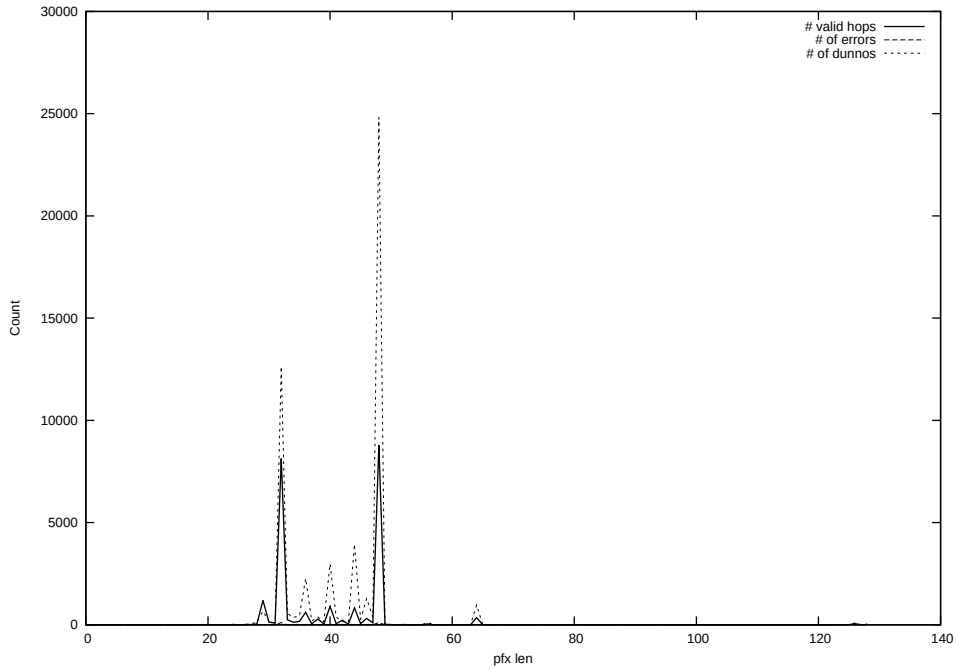
2015-02-27



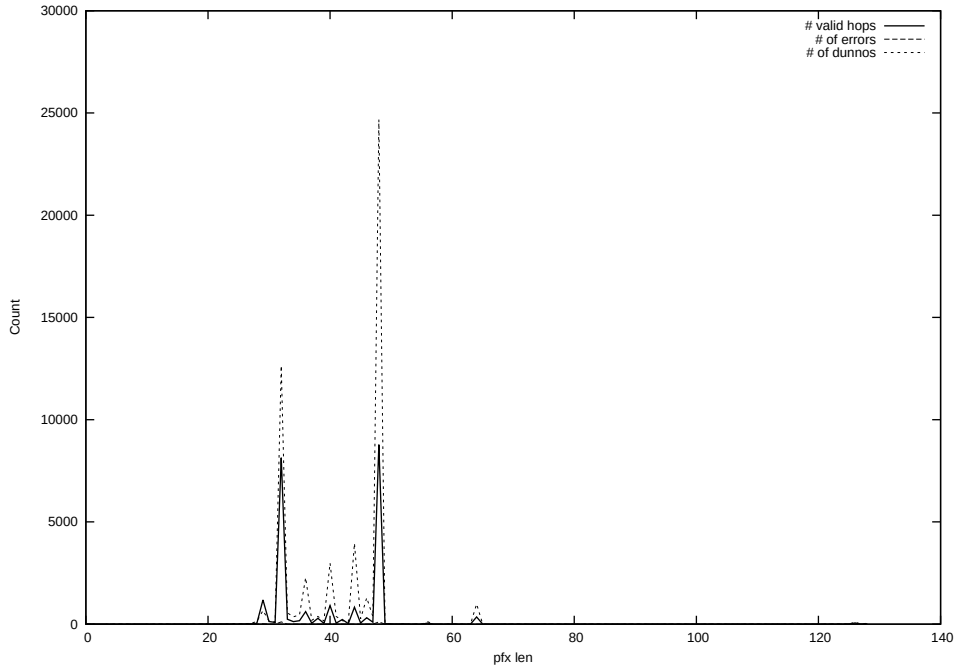
2015-02-28



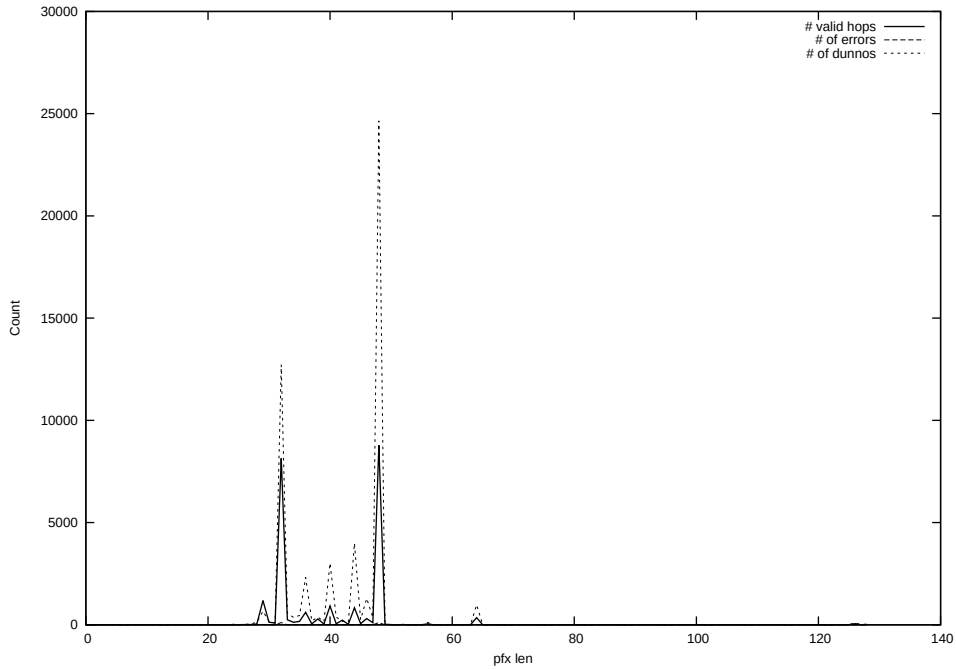
2015-03-01



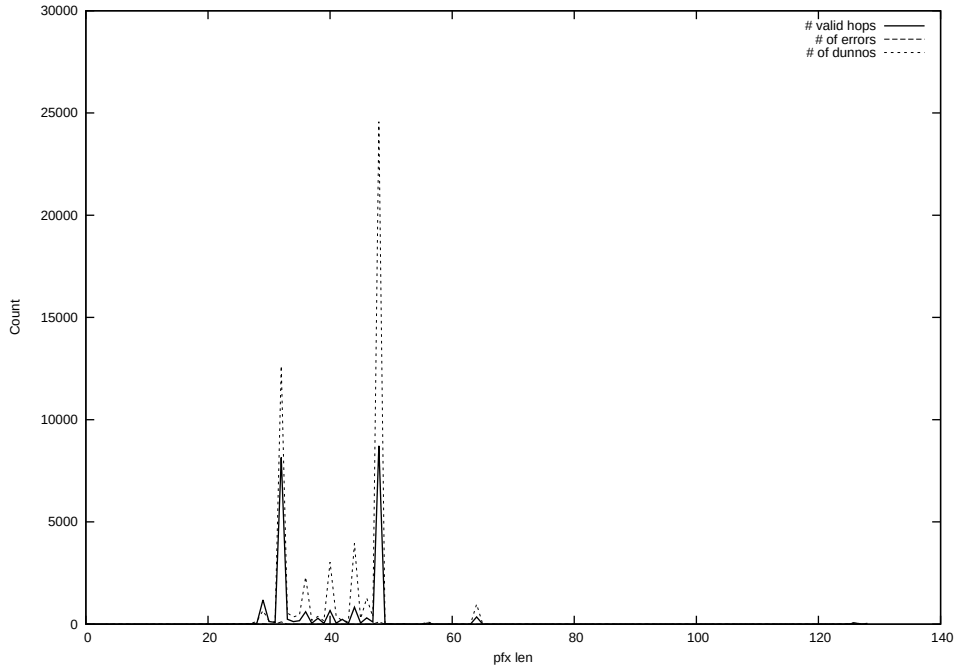
2015-03-02



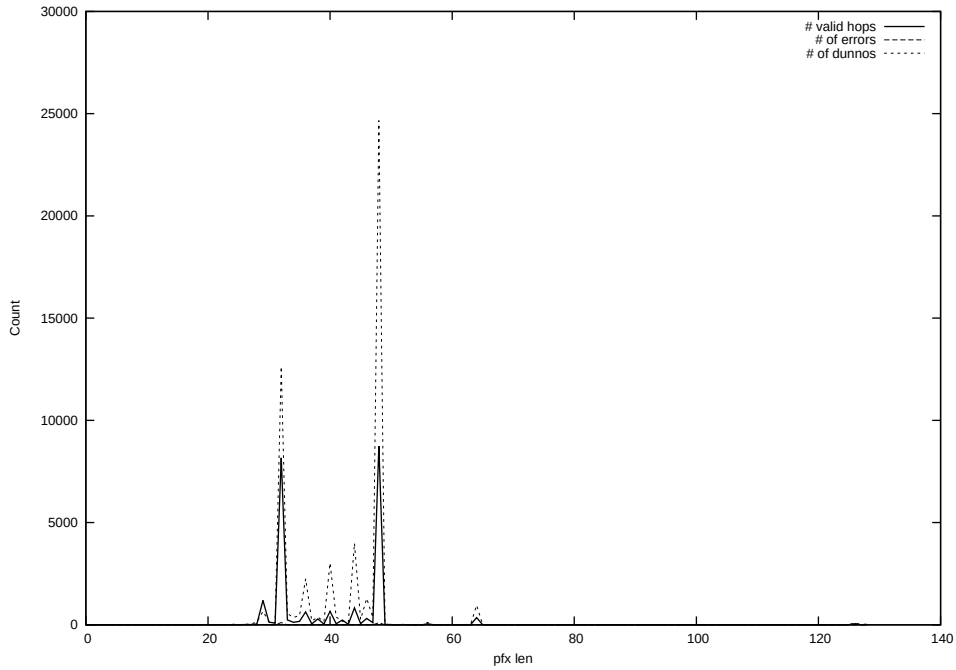
2015-03-03



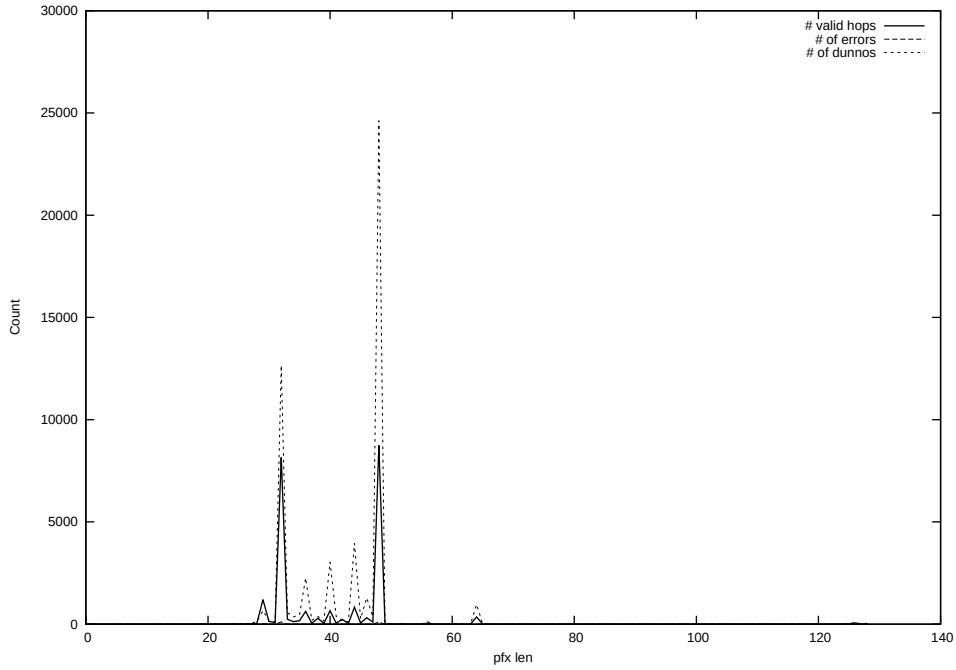
2015-03-04



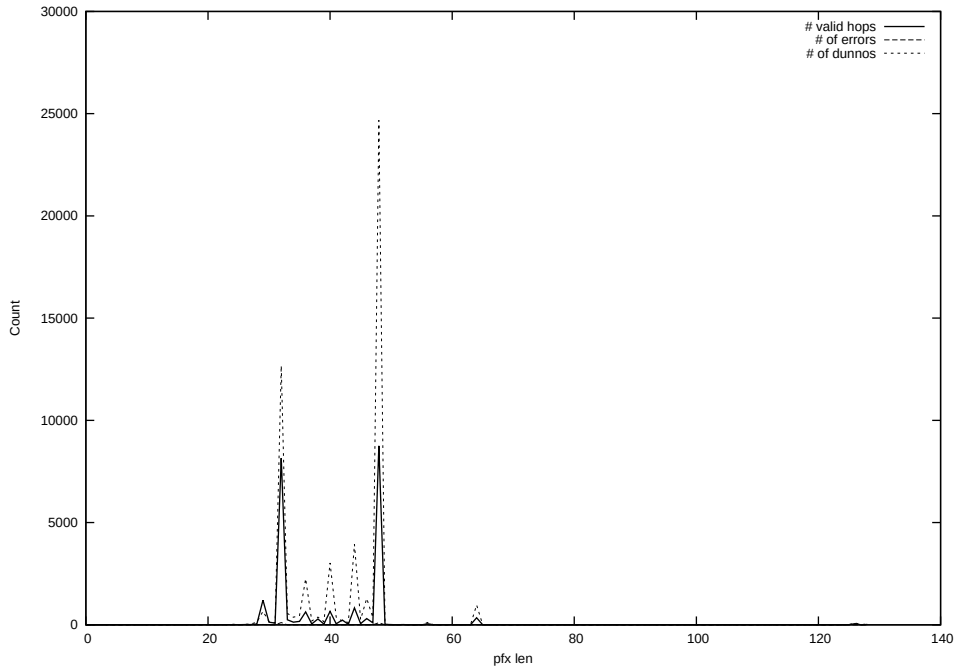
2015-03-05



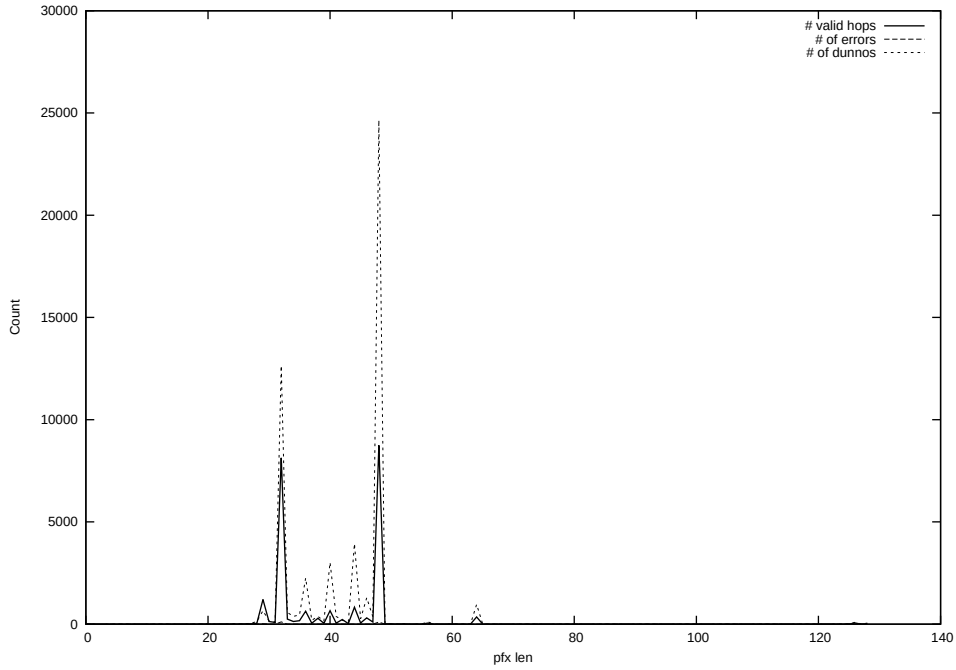
2015-03-06



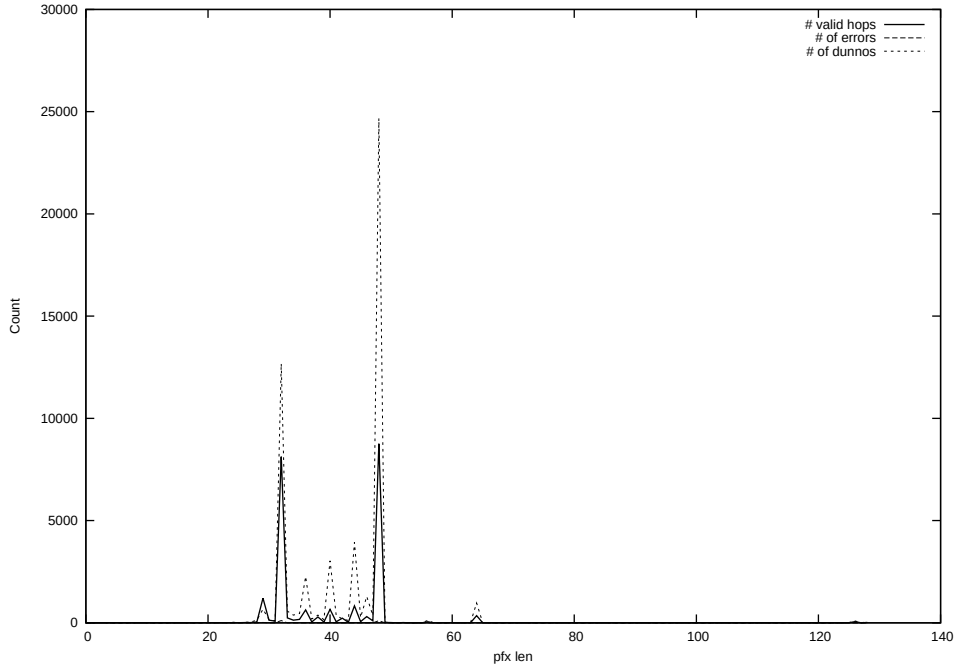
2015-03-07



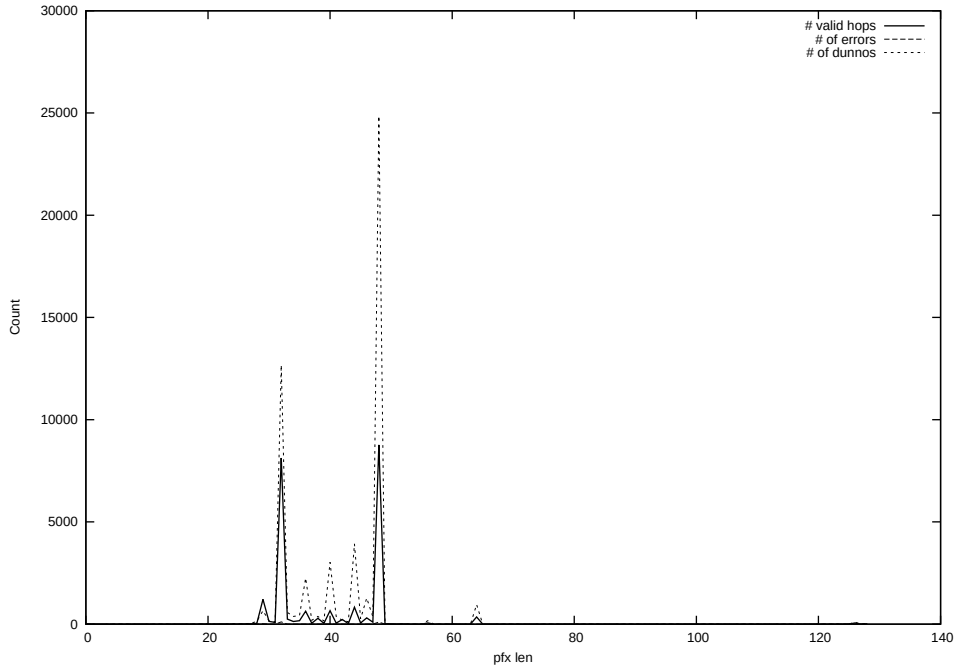
2015-03-08



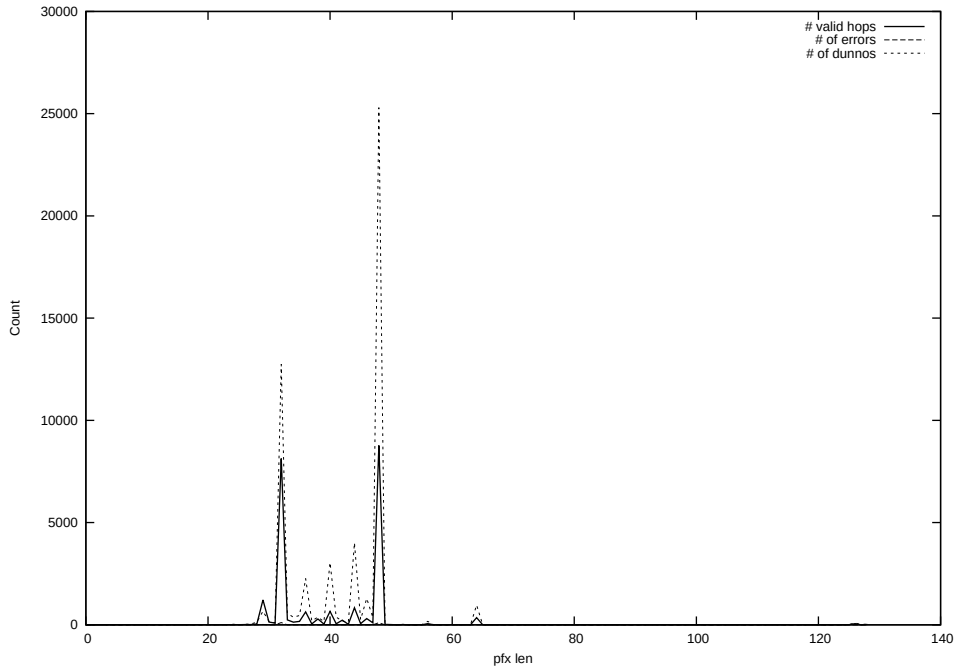
2015-03-09



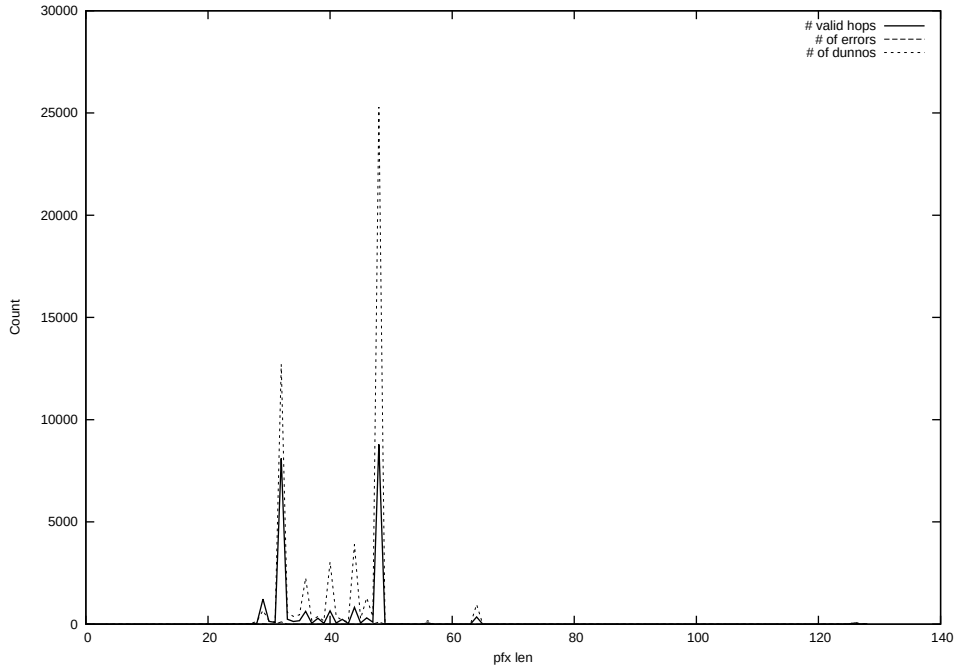
2015-03-10



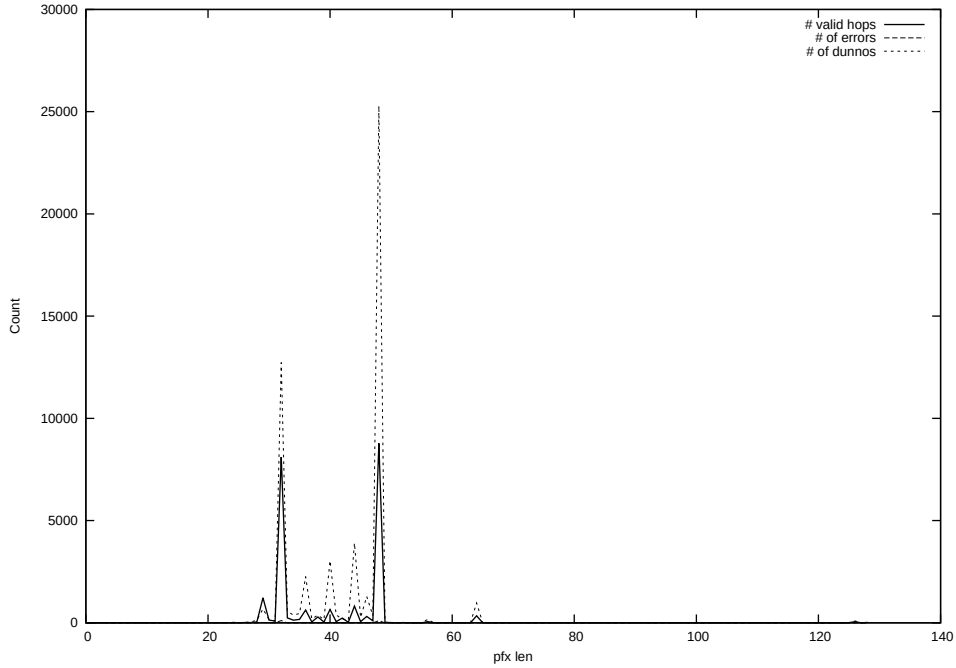
2015-03-11



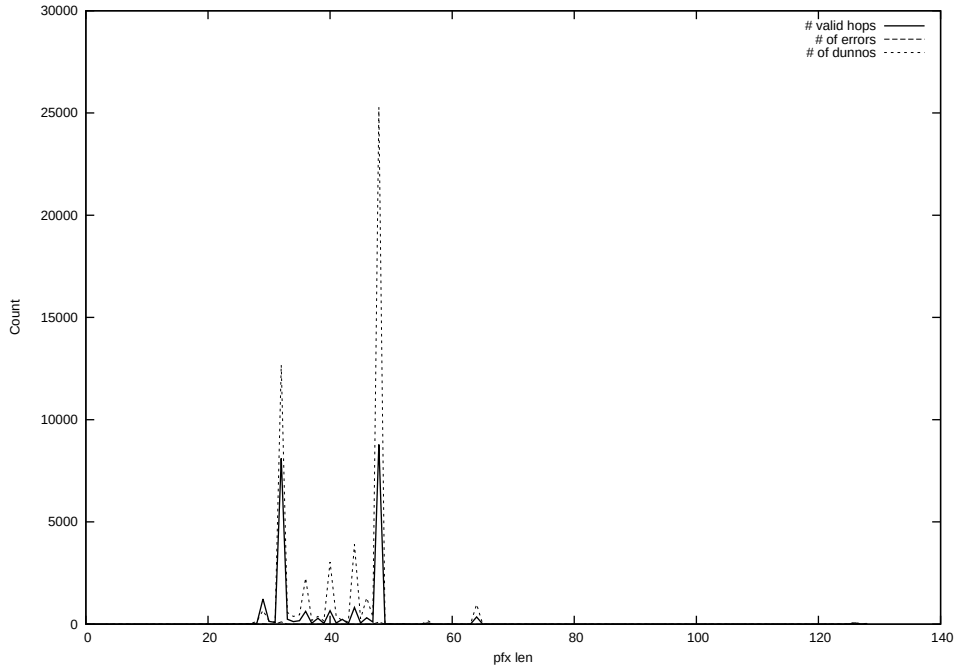
2015-03-12



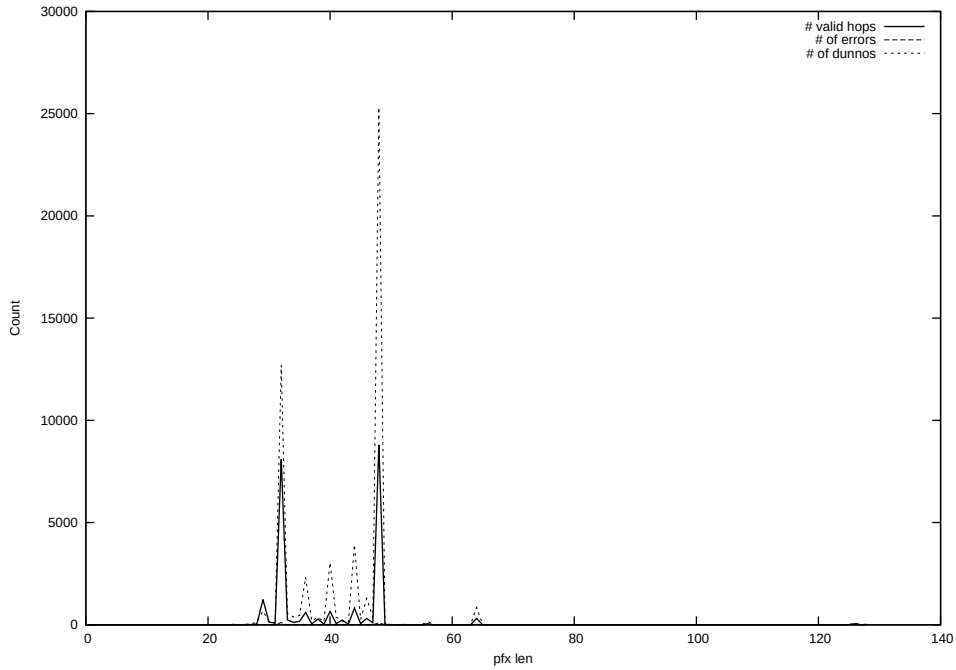
2015-03-13



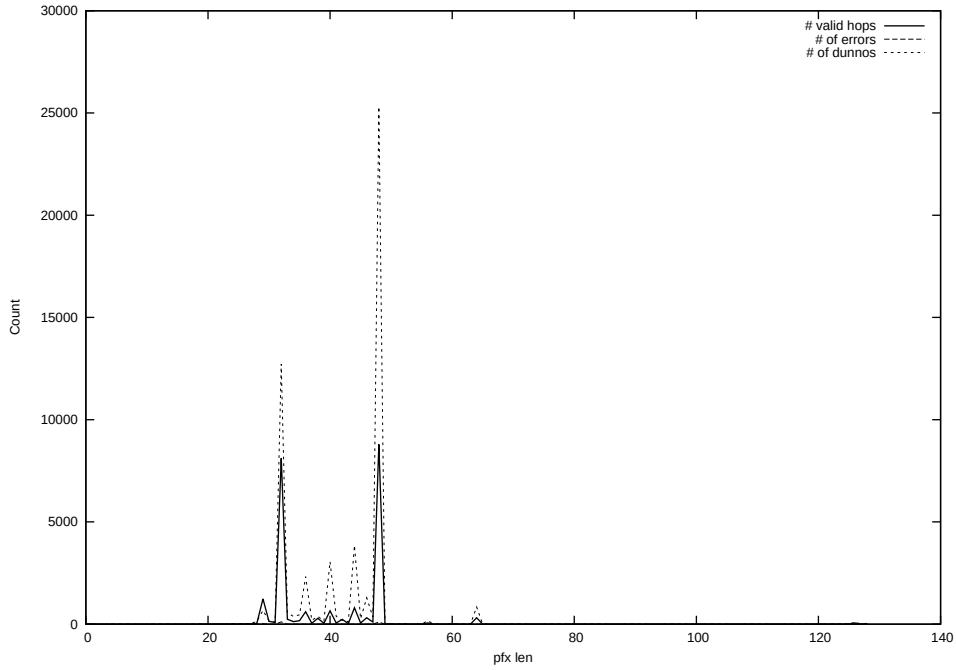
2015-03-14



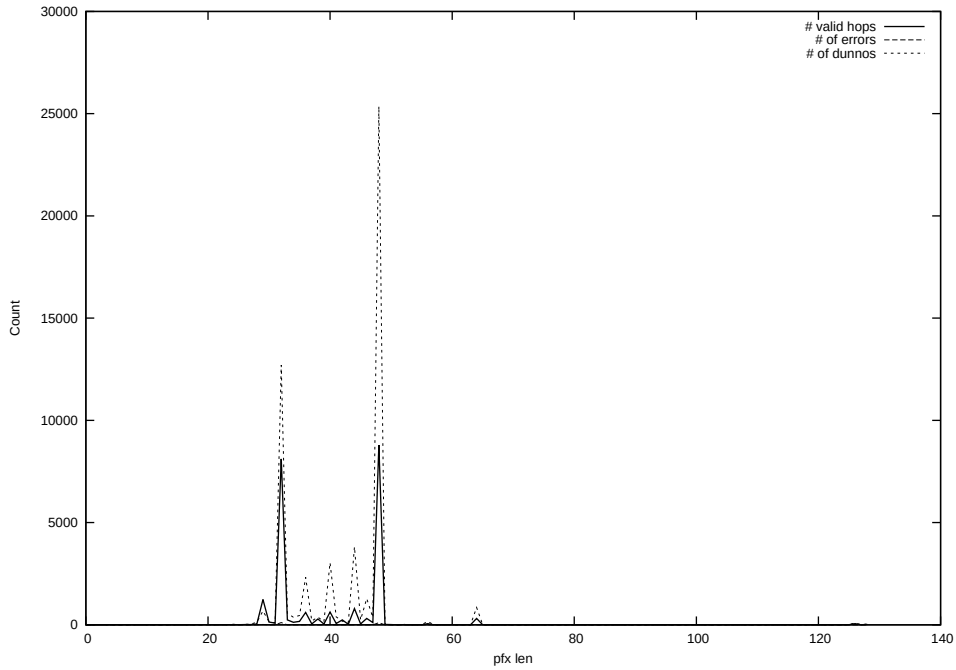
2015-03-15



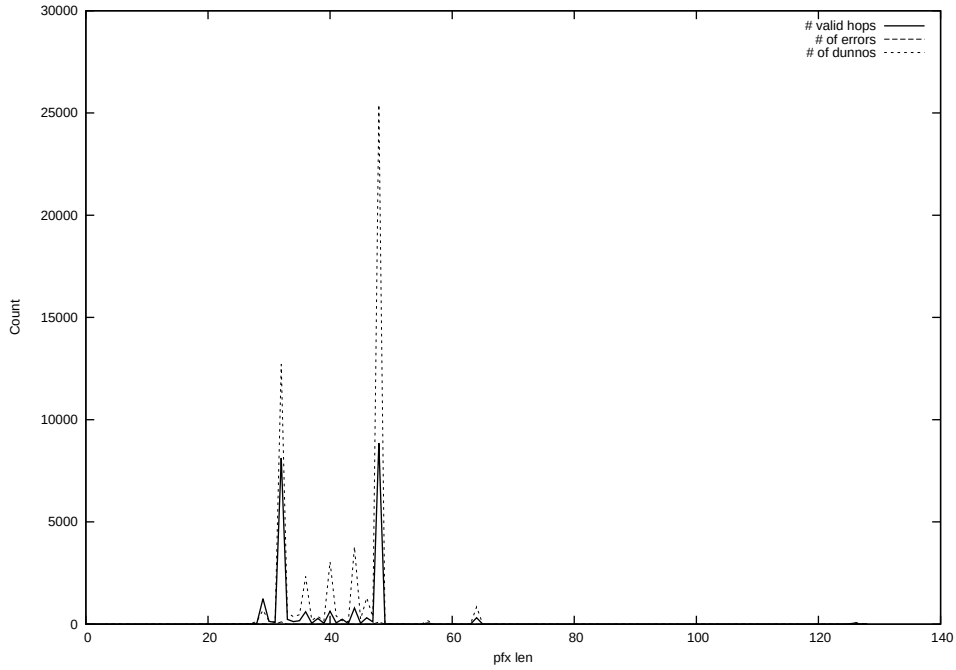
2015-03-16



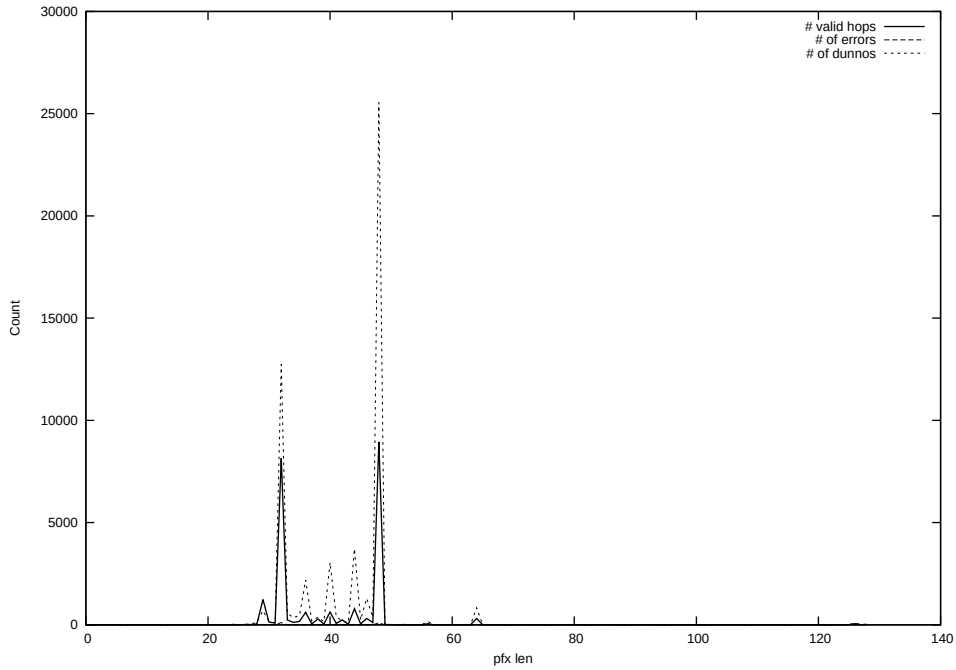
2015-03-17



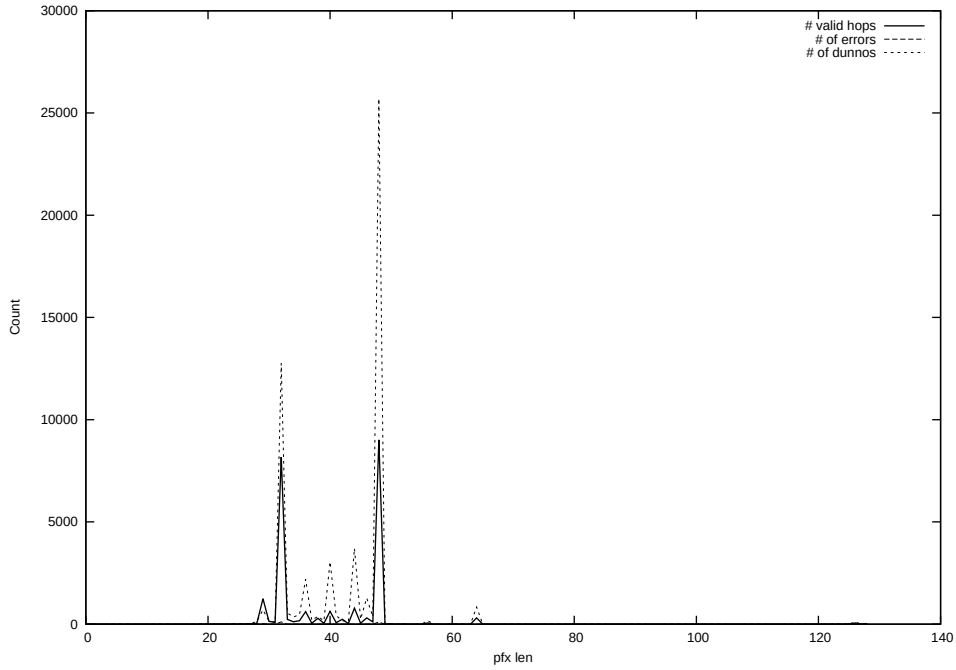
2015-03-18



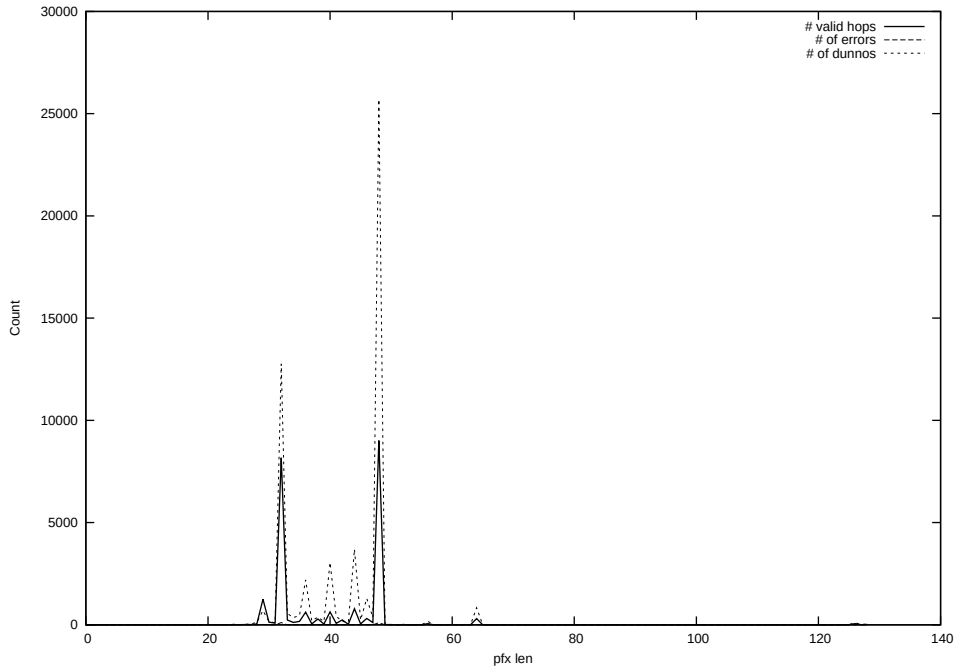
2015-03-19



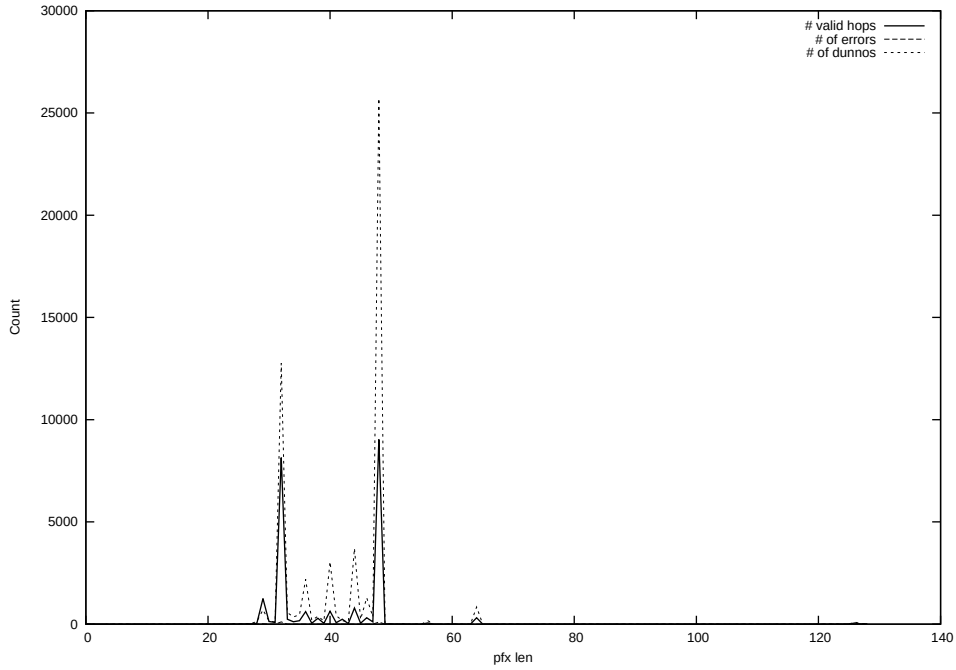
2015-03-20



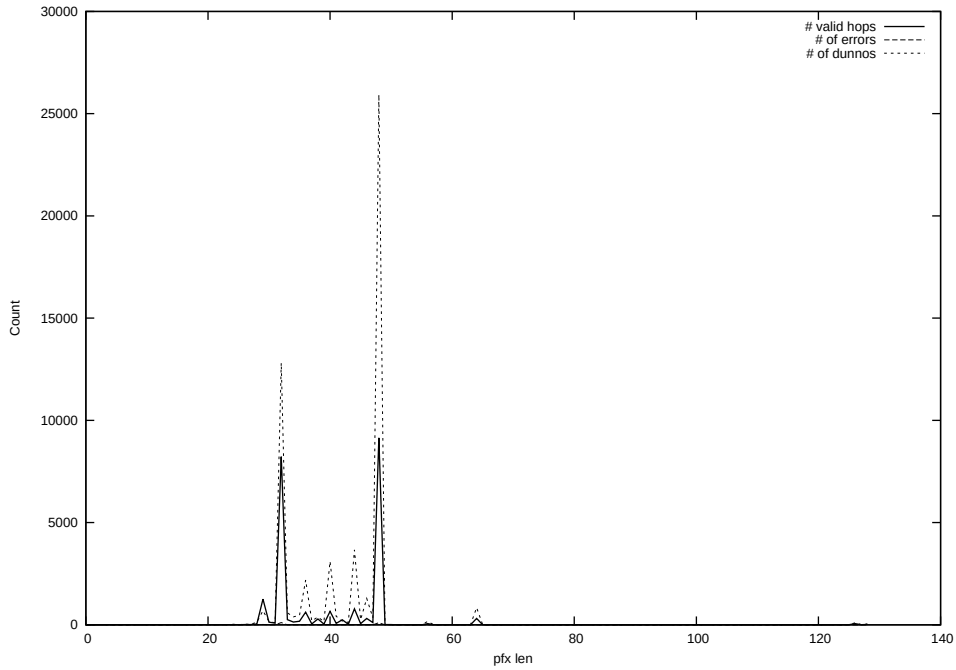
2015-03-21



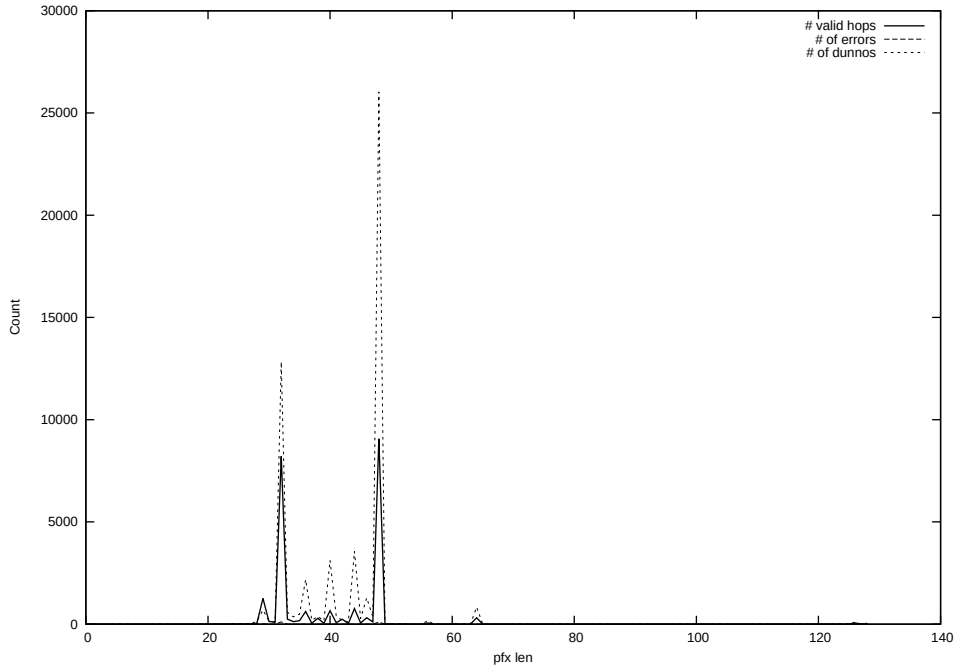
2015-03-22



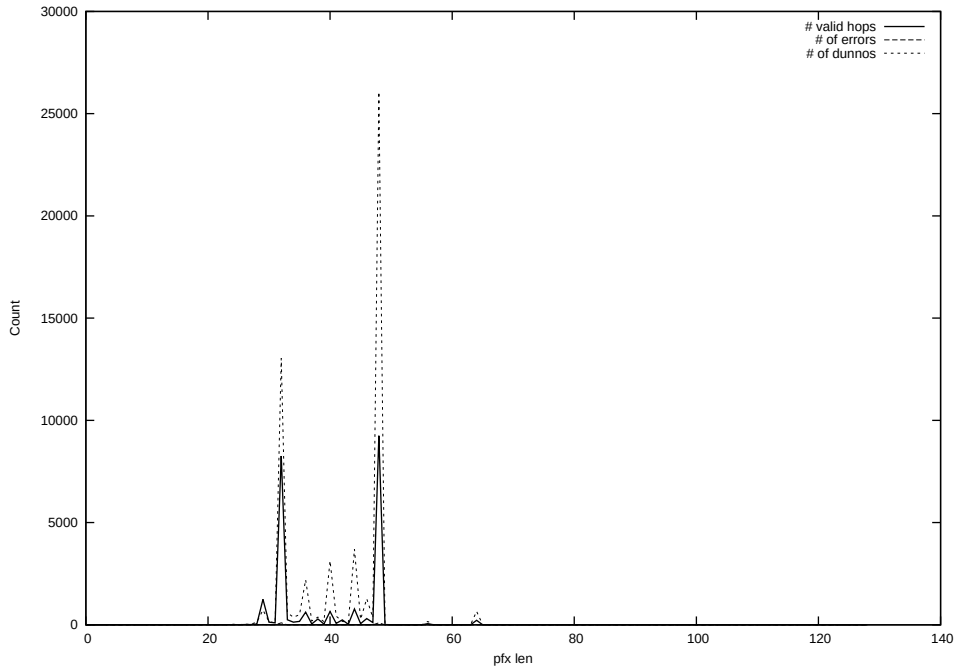
2015-03-23



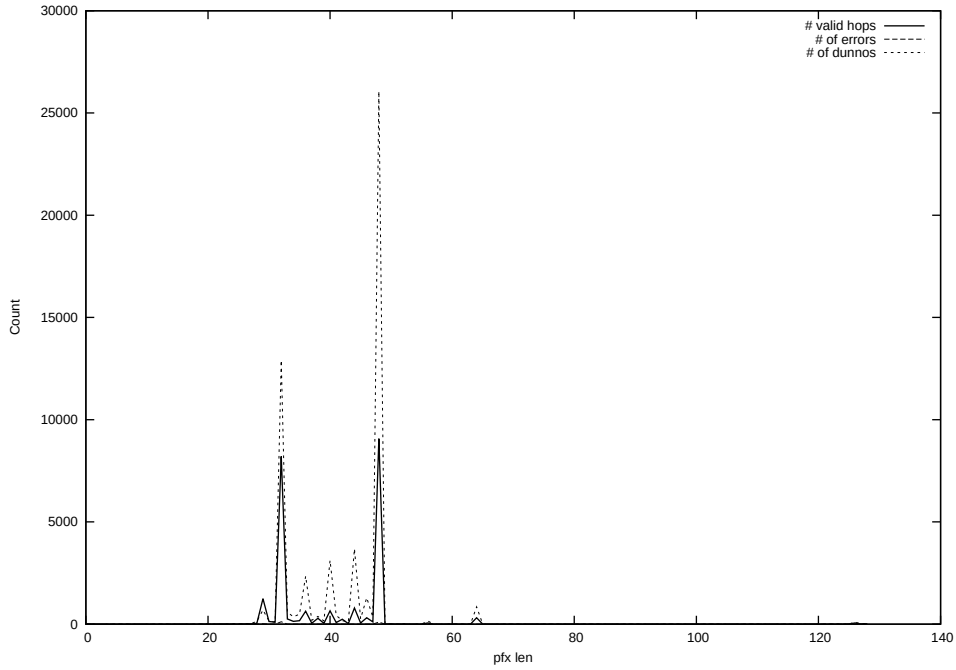
2015-03-24



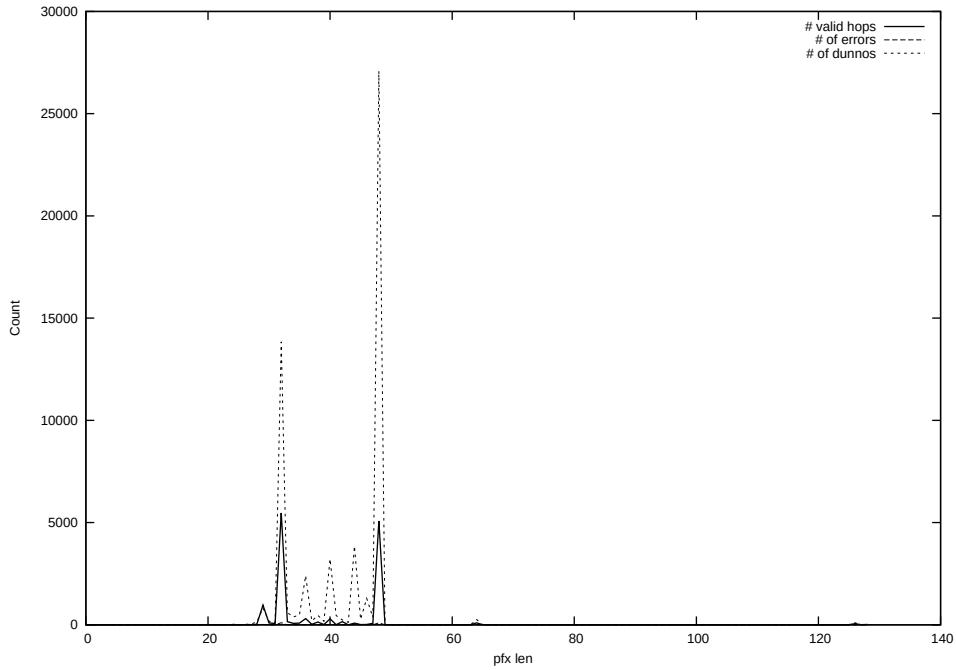
2015-03-25



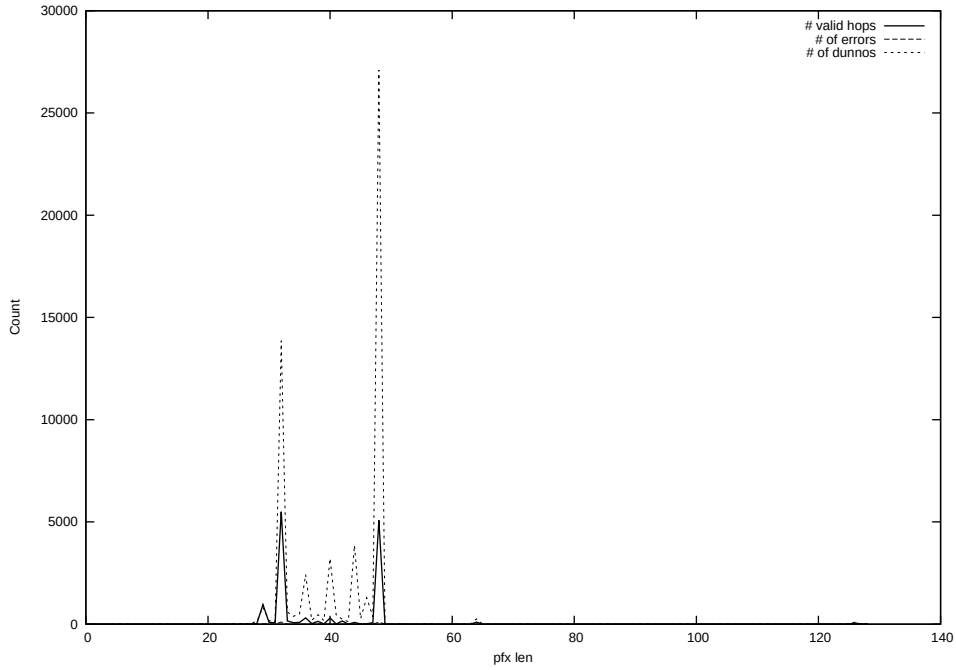
2015-03-26



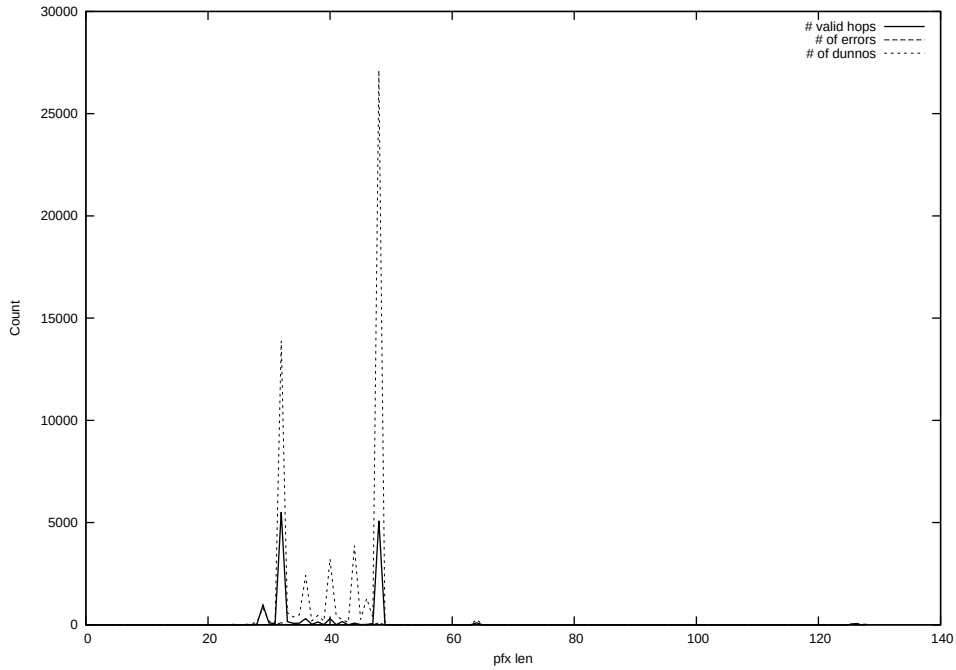
2015-03-27



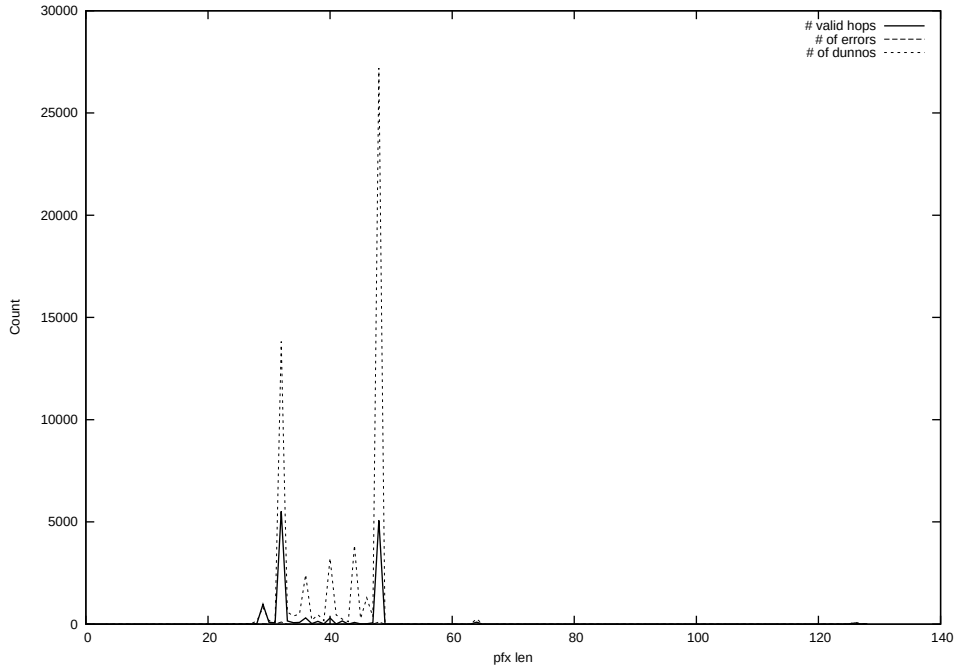
2015-03-28



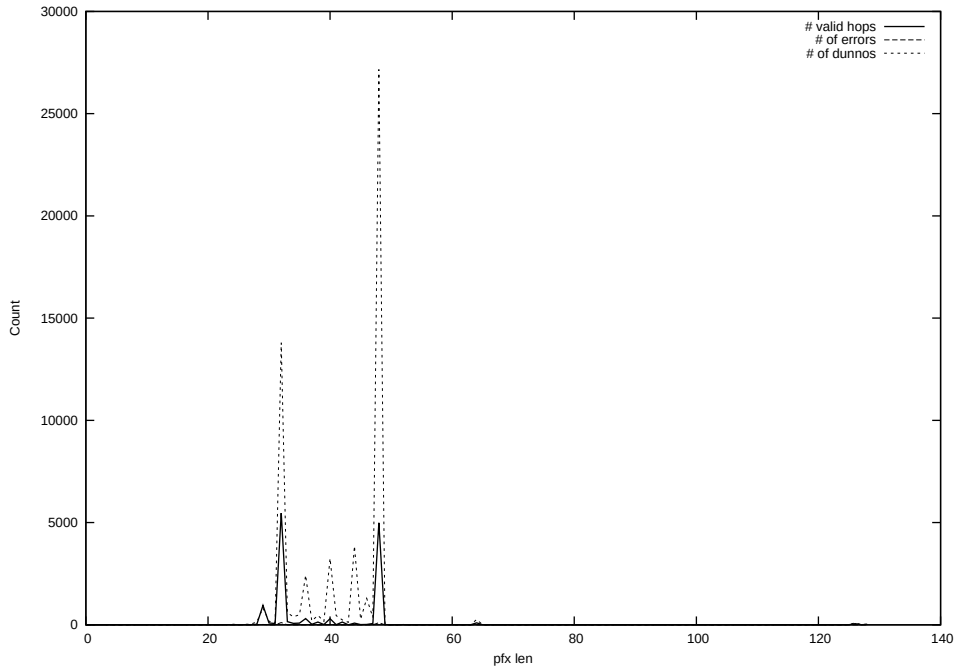
2015-03-29



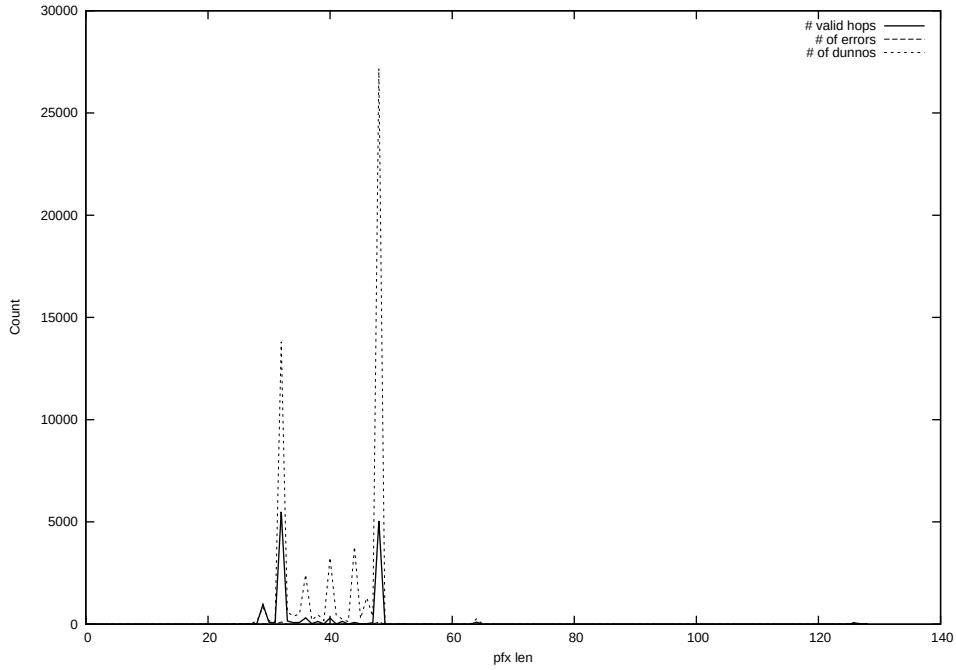
2015-03-30



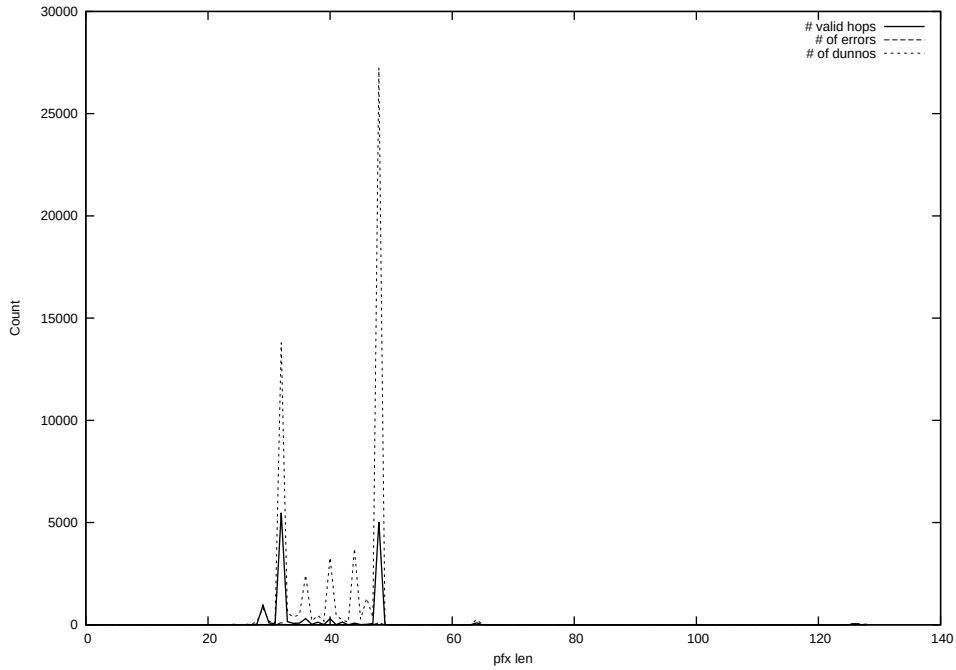
2015-03-31



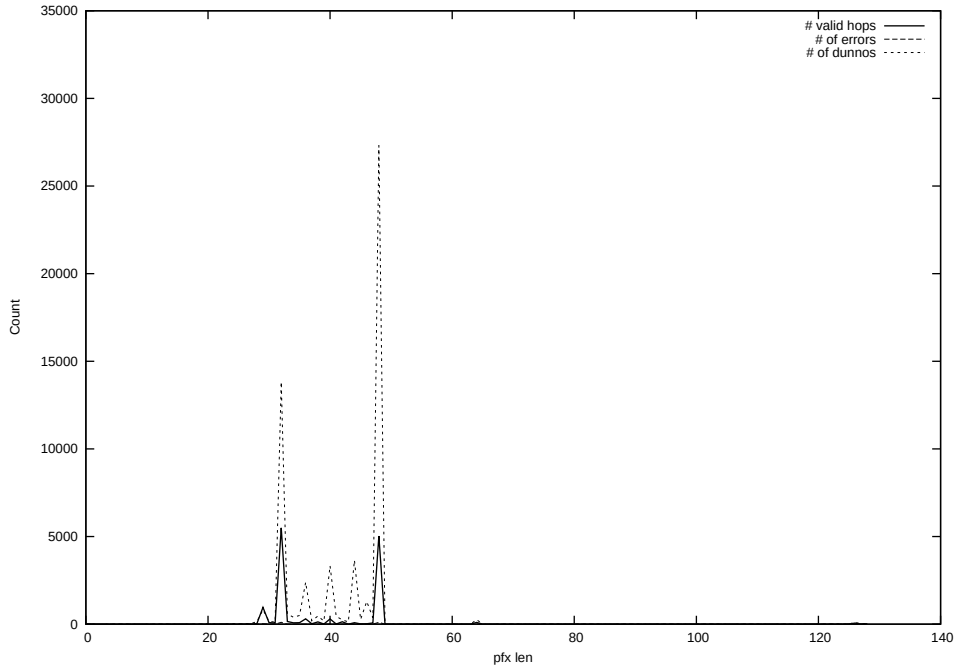
2015-04-01



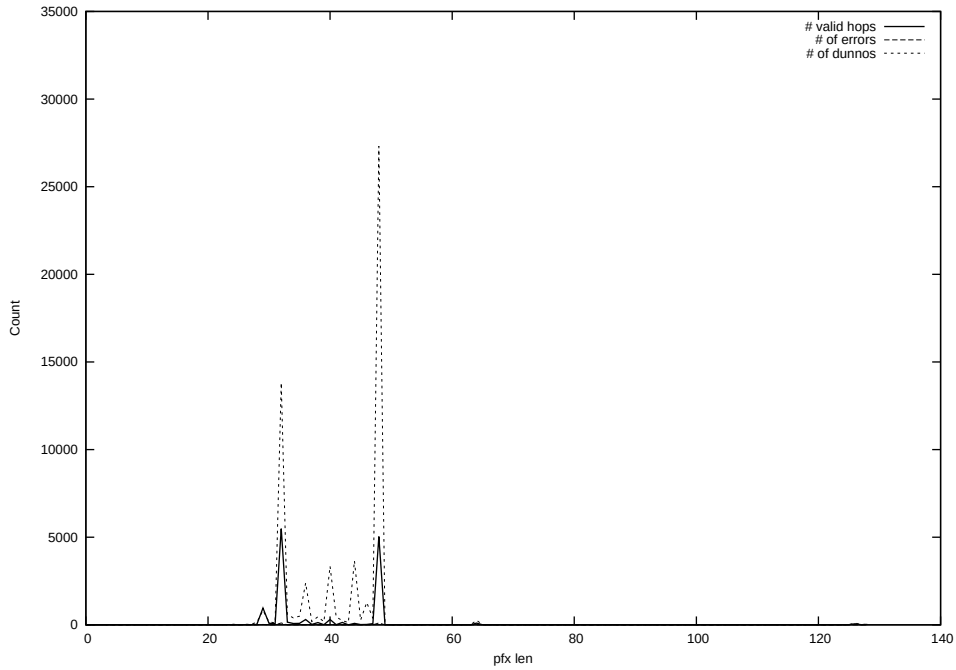
2015-04-02



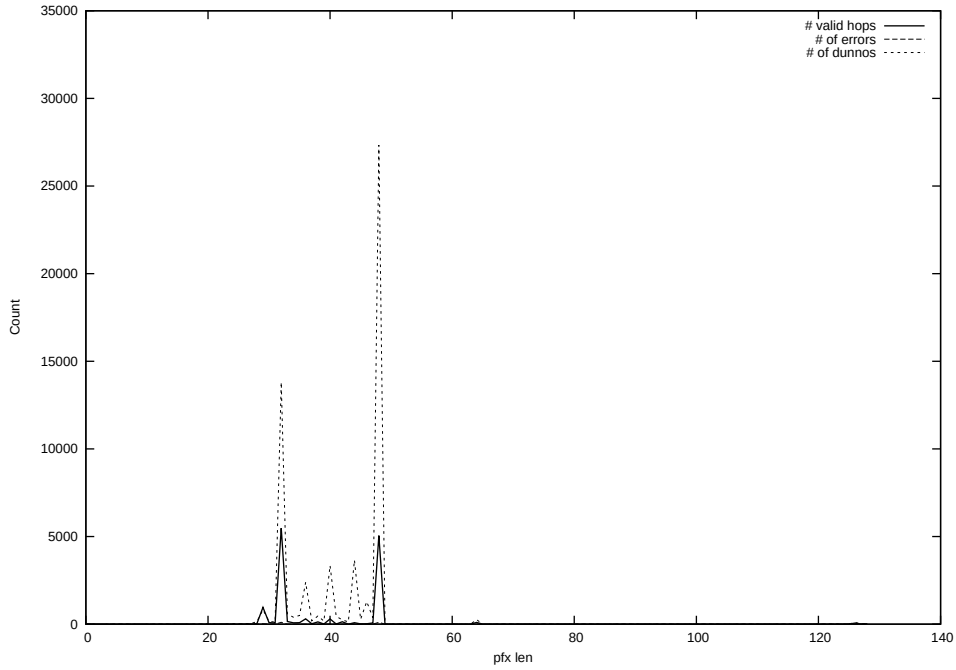
2015-04-03



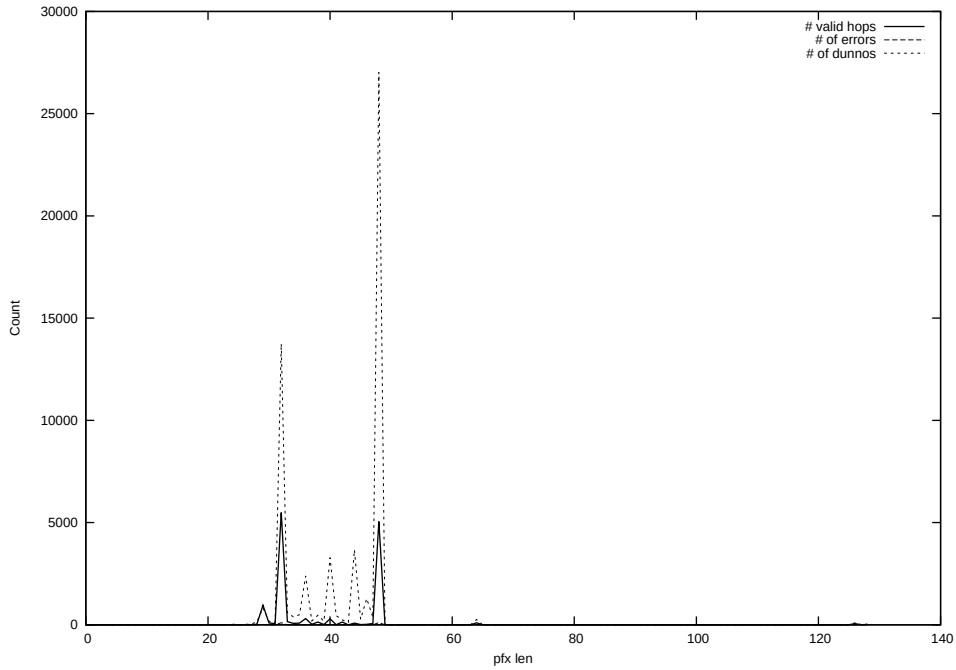
2015-04-04



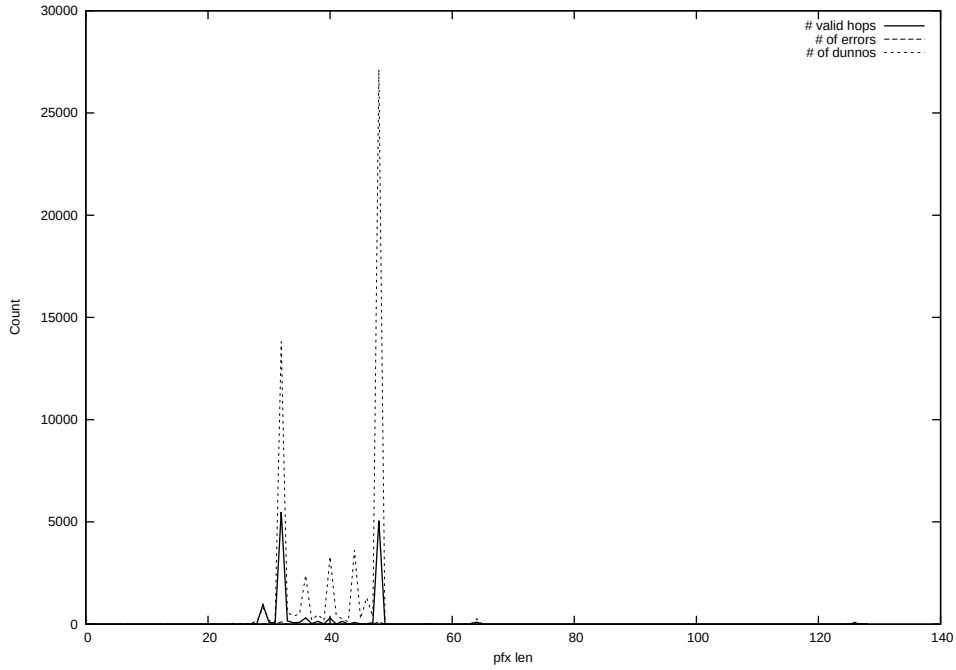
2015-04-05



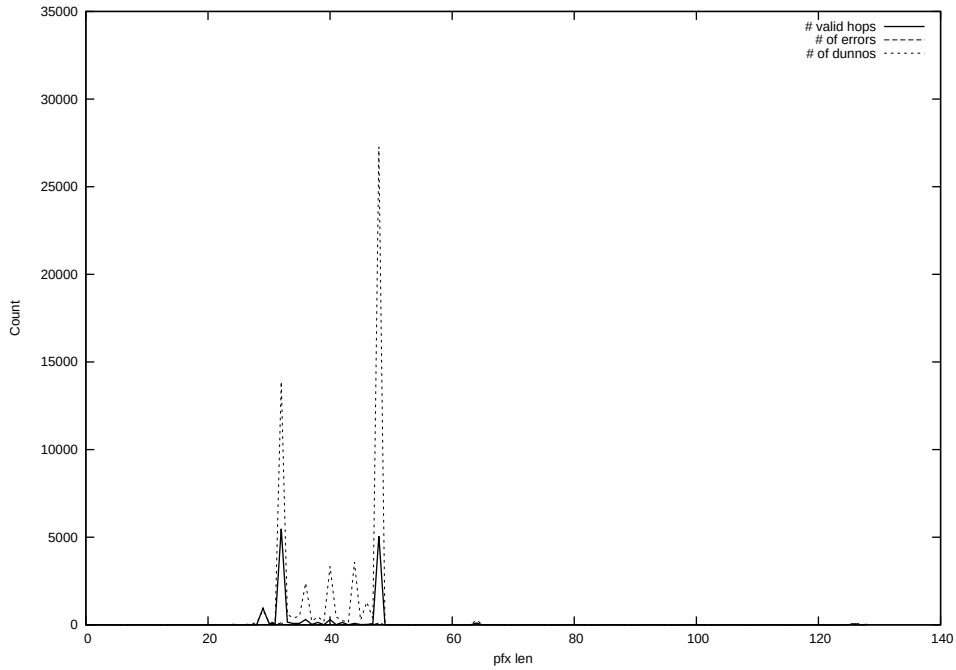
2015-04-06



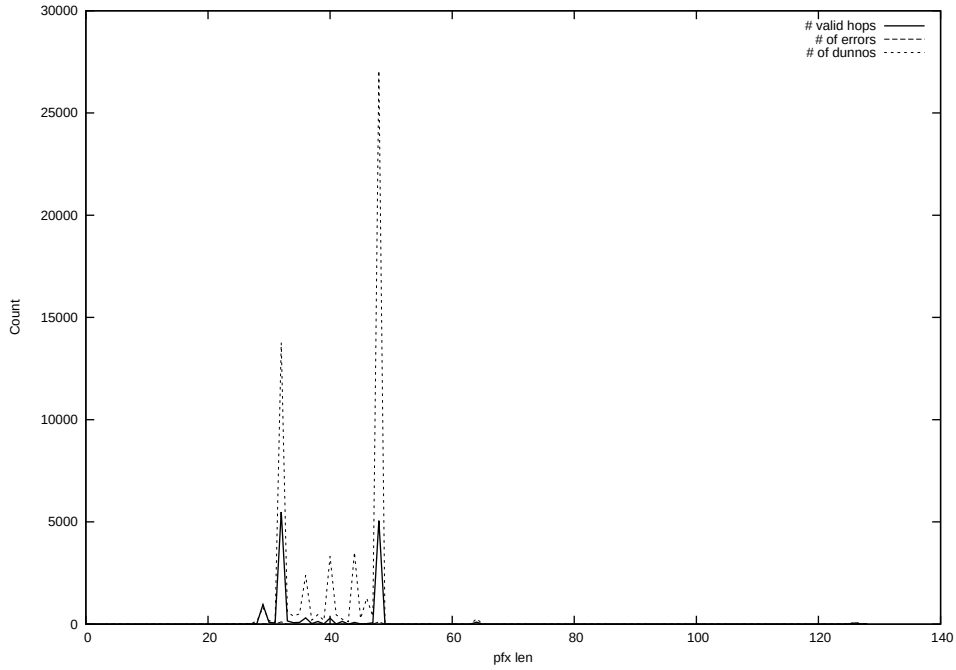
2015-04-07



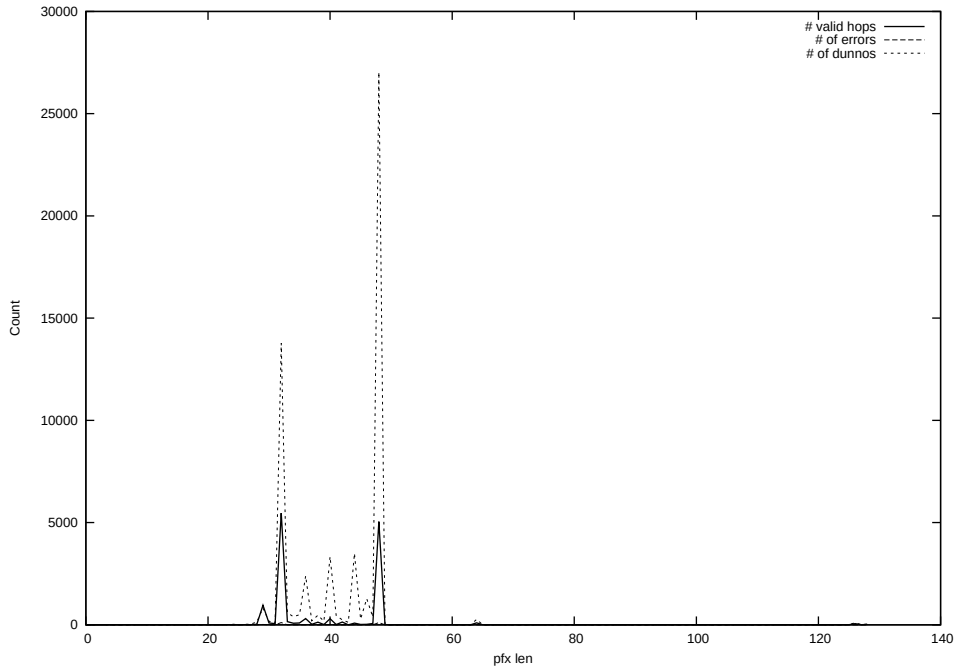
2015-04-08



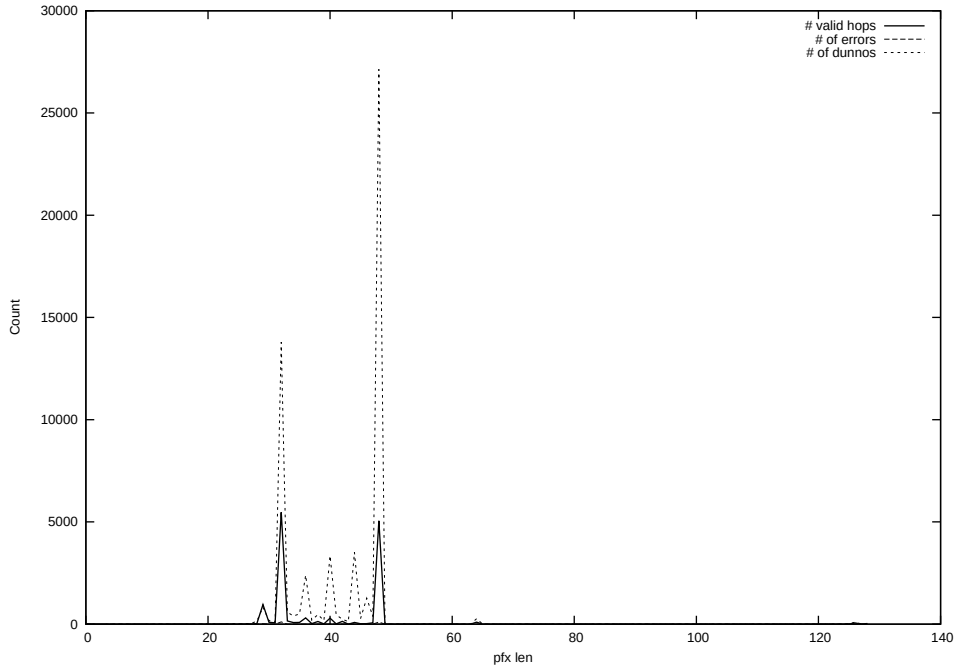
2015-04-09



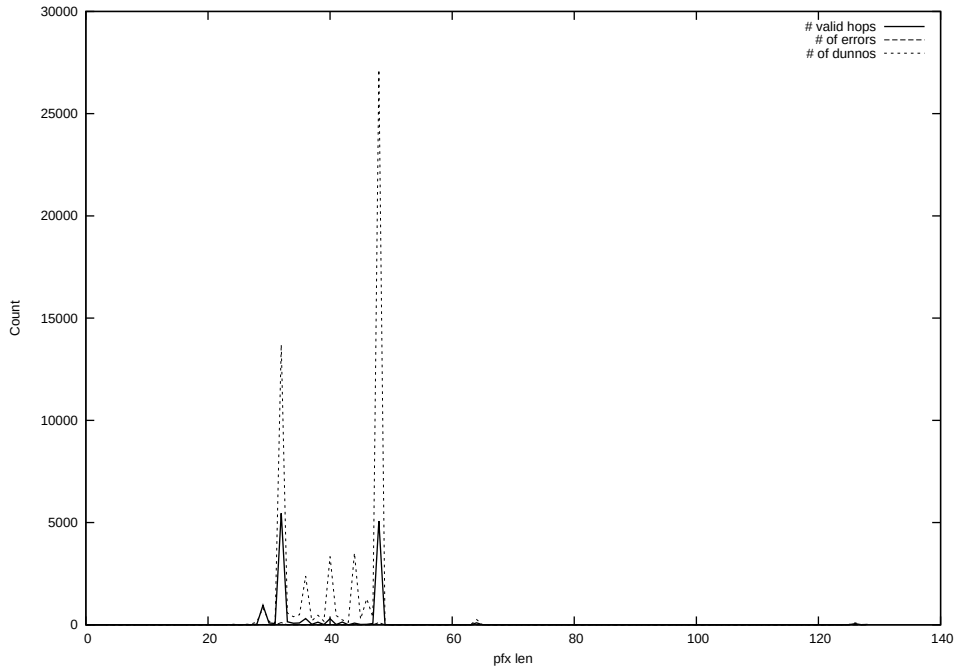
2015-04-10



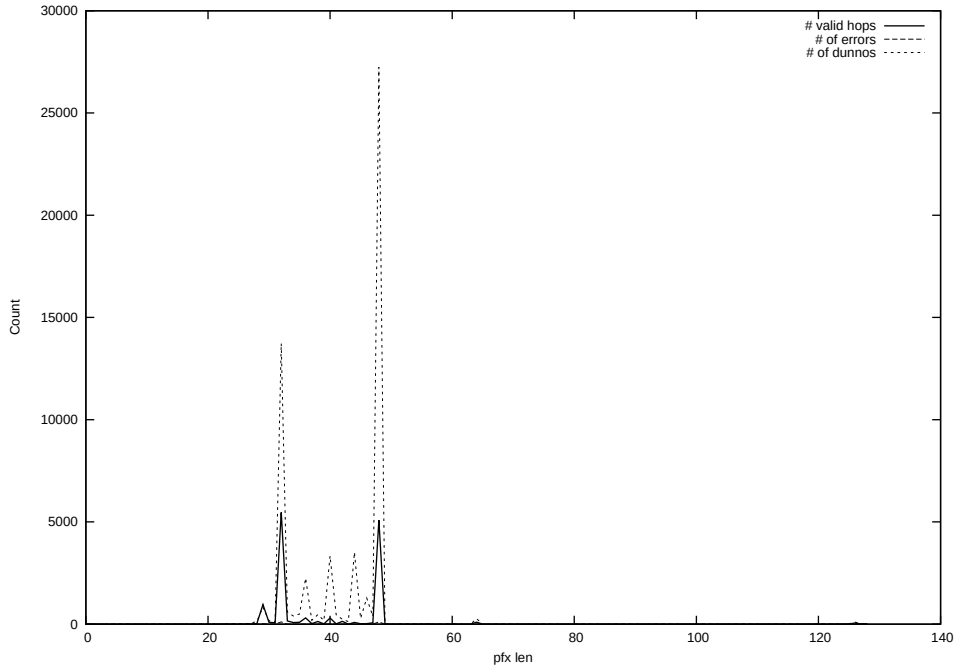
2015-04-11



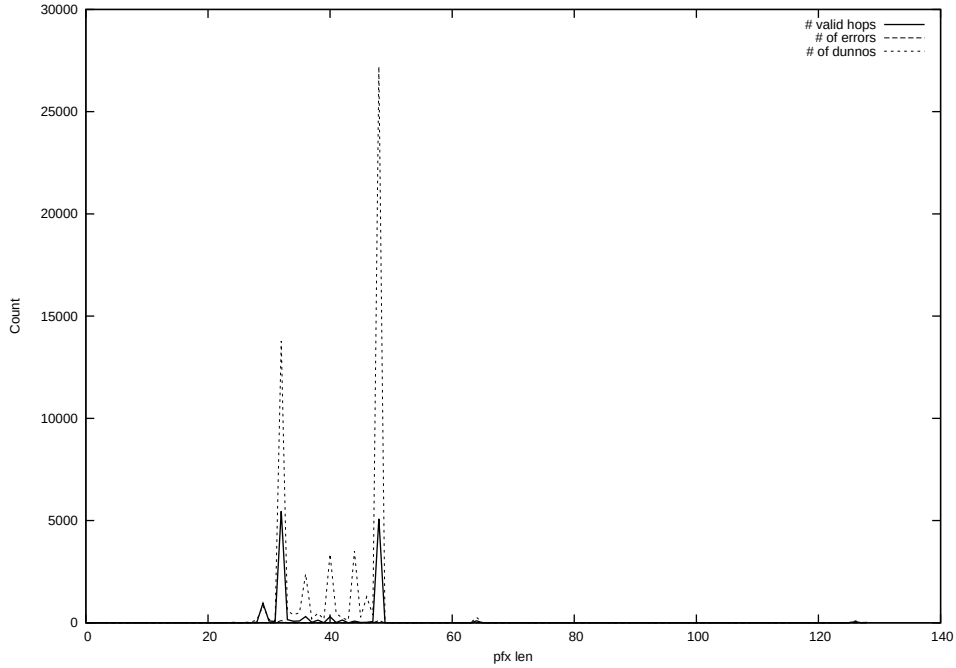
2015-04-12



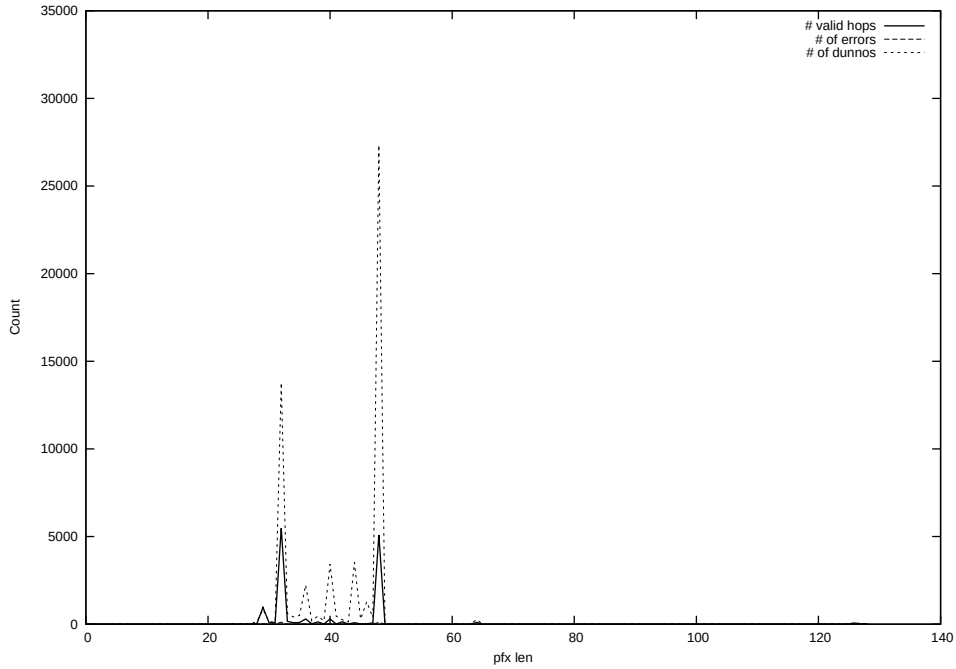
2015-04-13



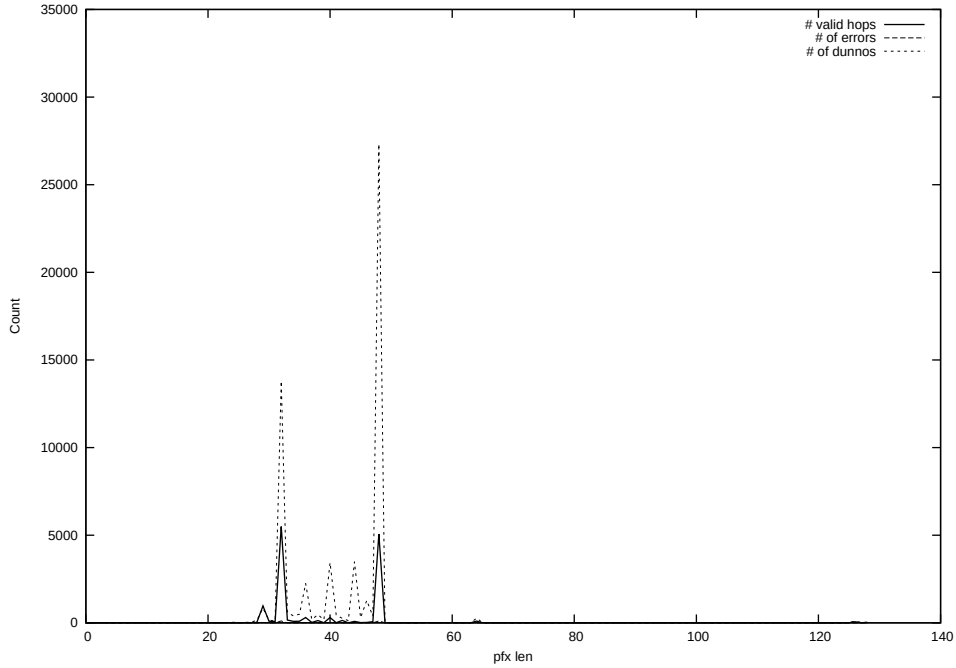
2015-04-14



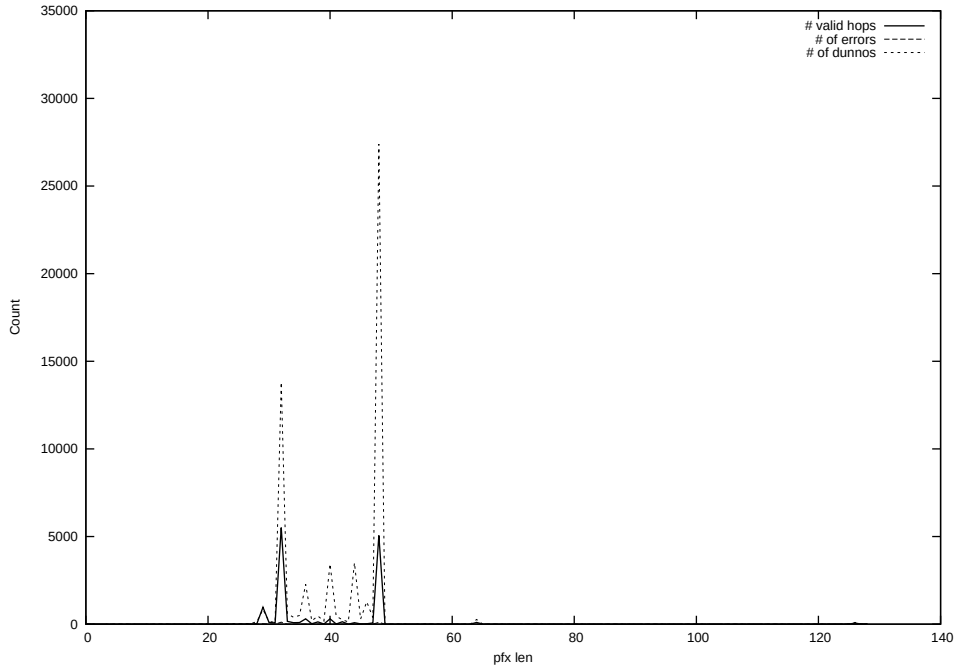
2015-04-15



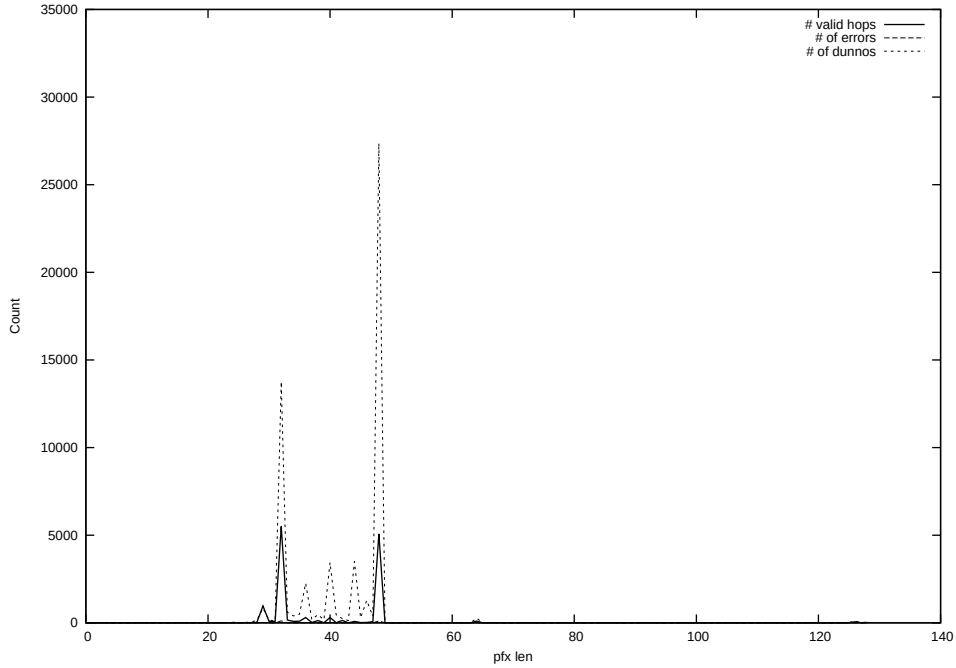
2015-04-16



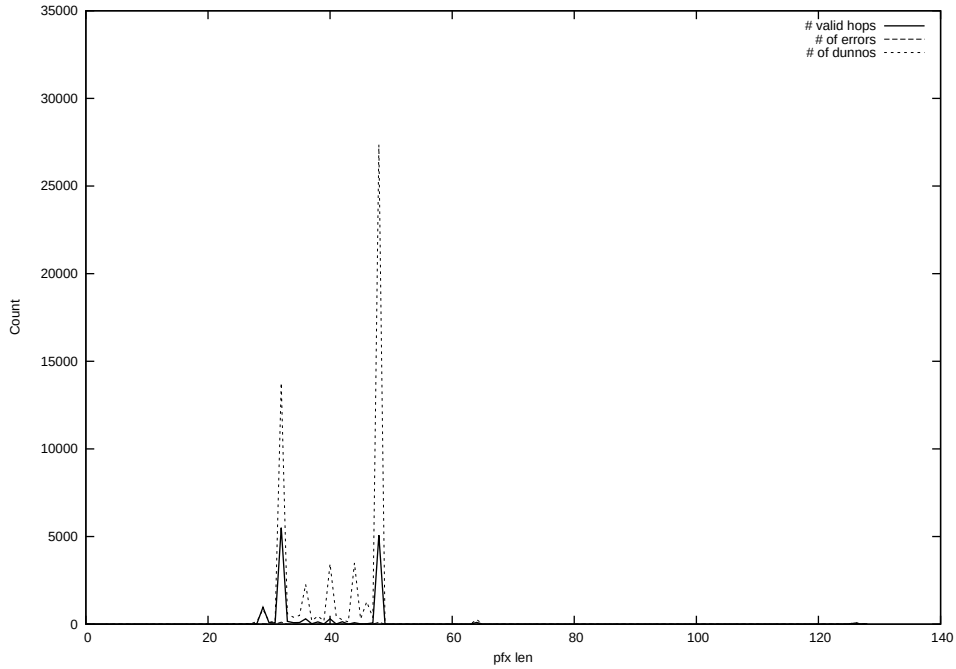
2015-04-17



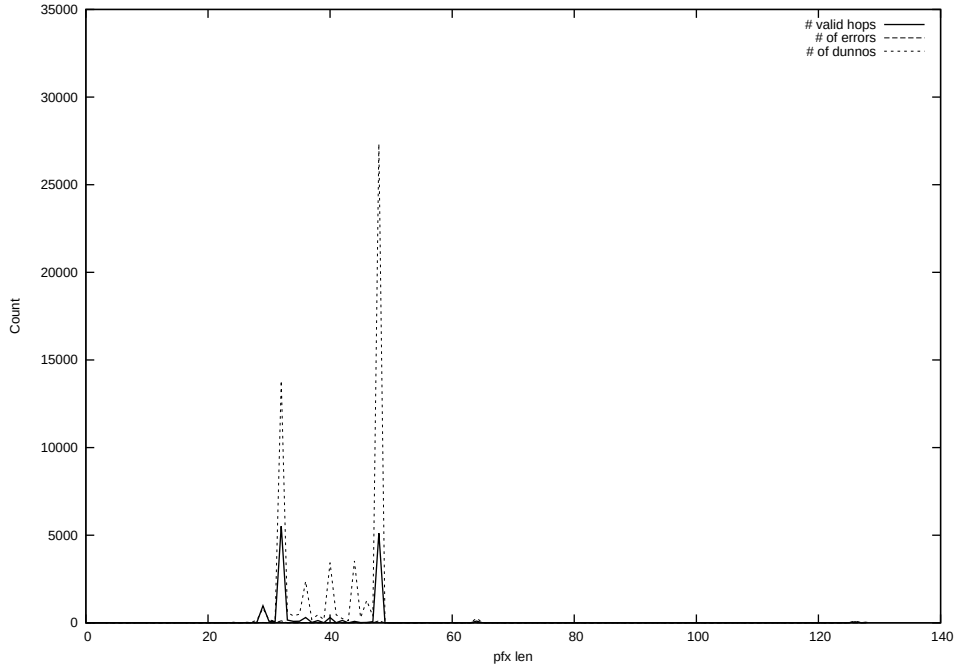
2015-04-18



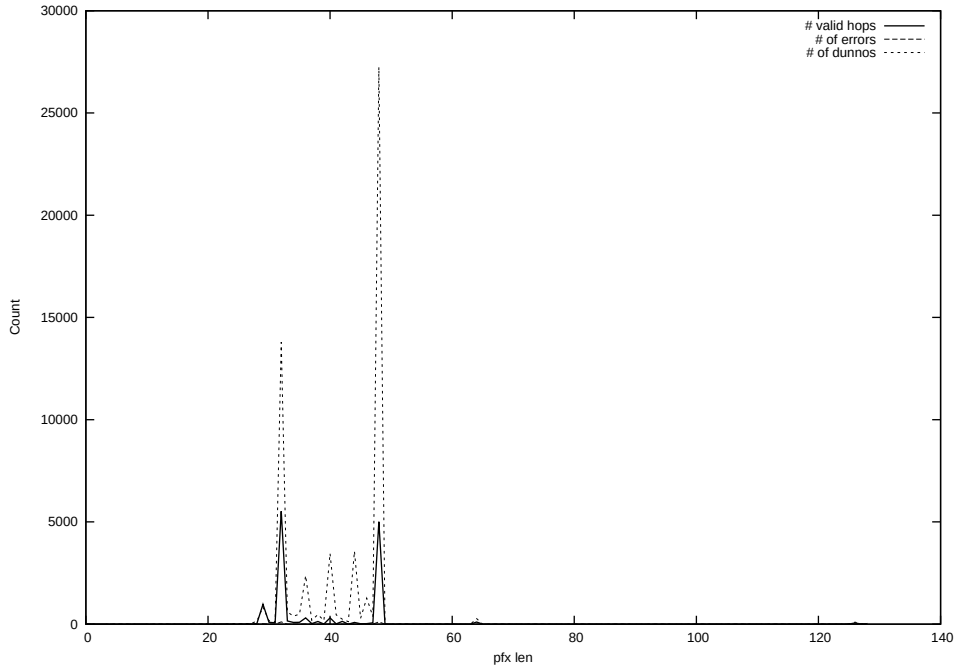
2015-04-19



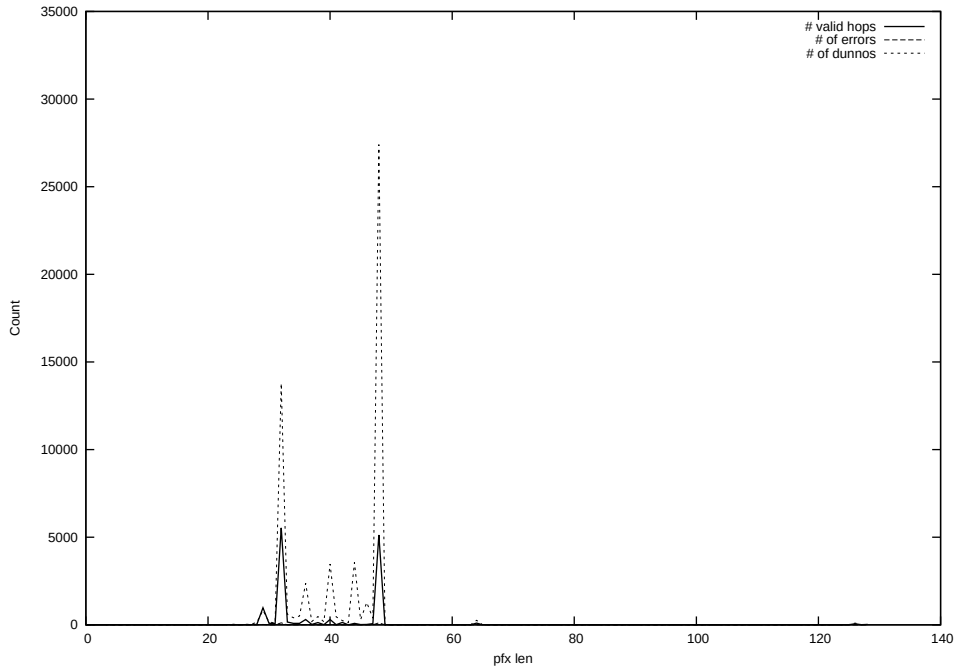
2015-04-20



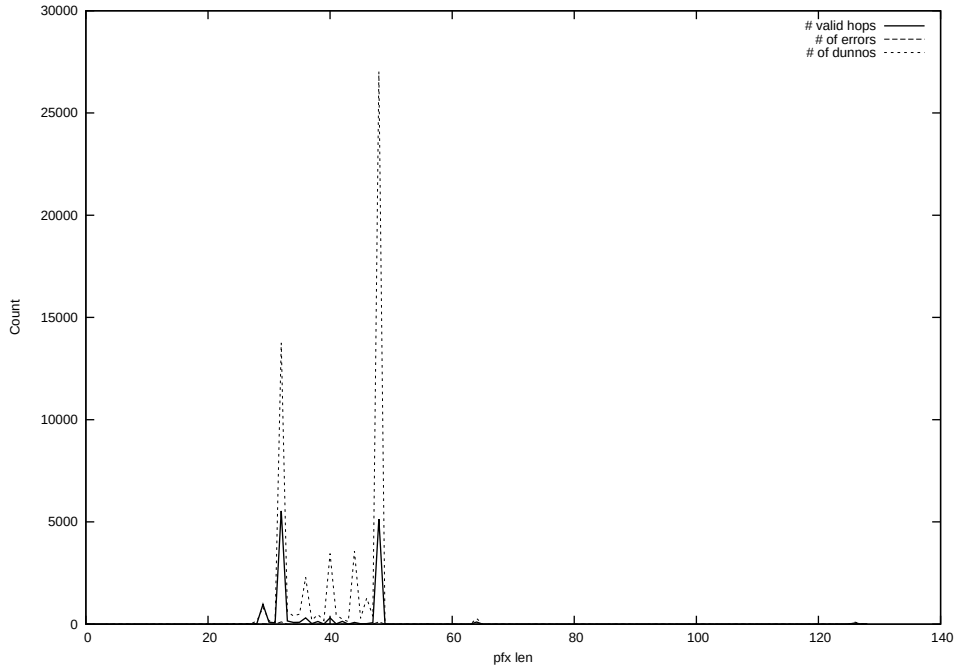
2015-04-21



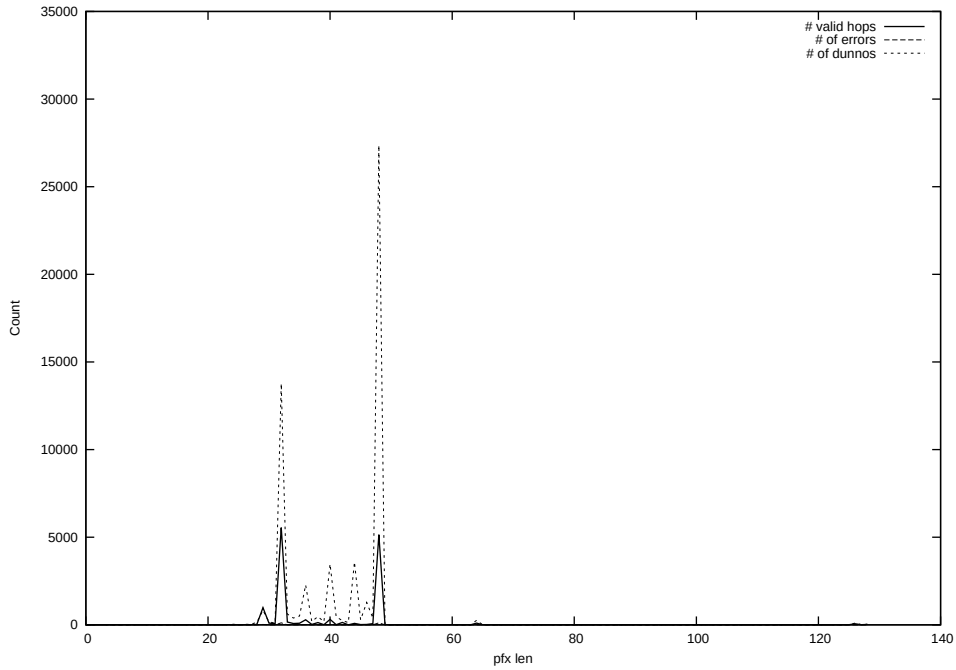
2015-04-22



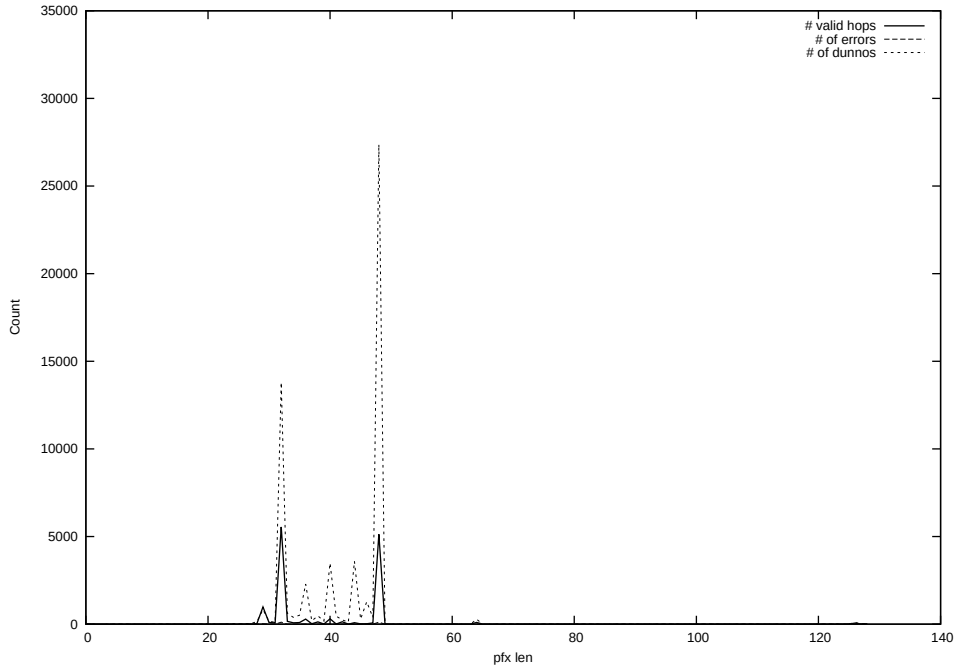
2015-04-23



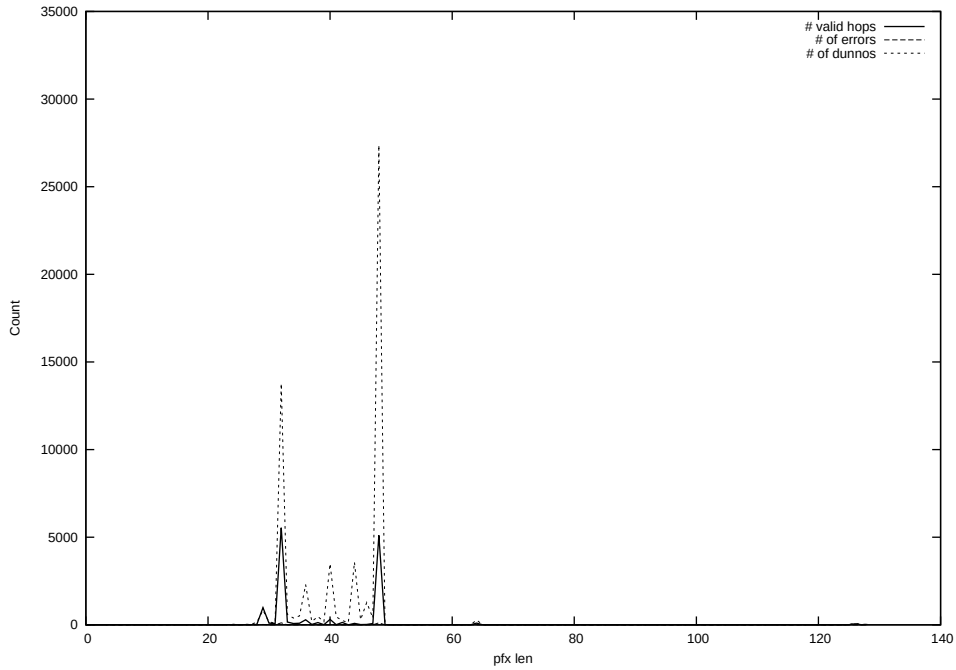
2015-04-24



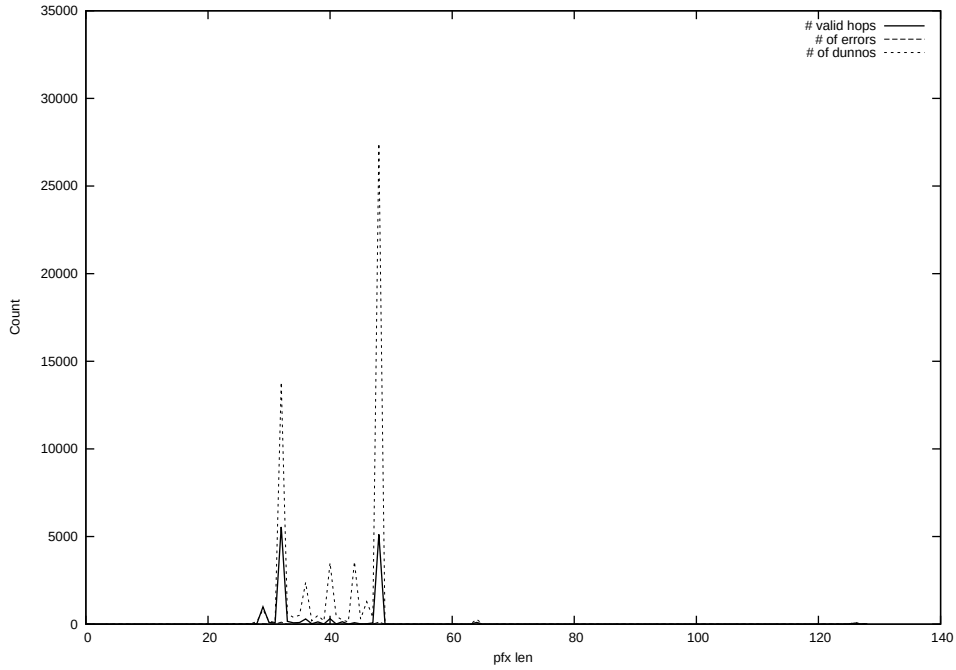
2015-04-25



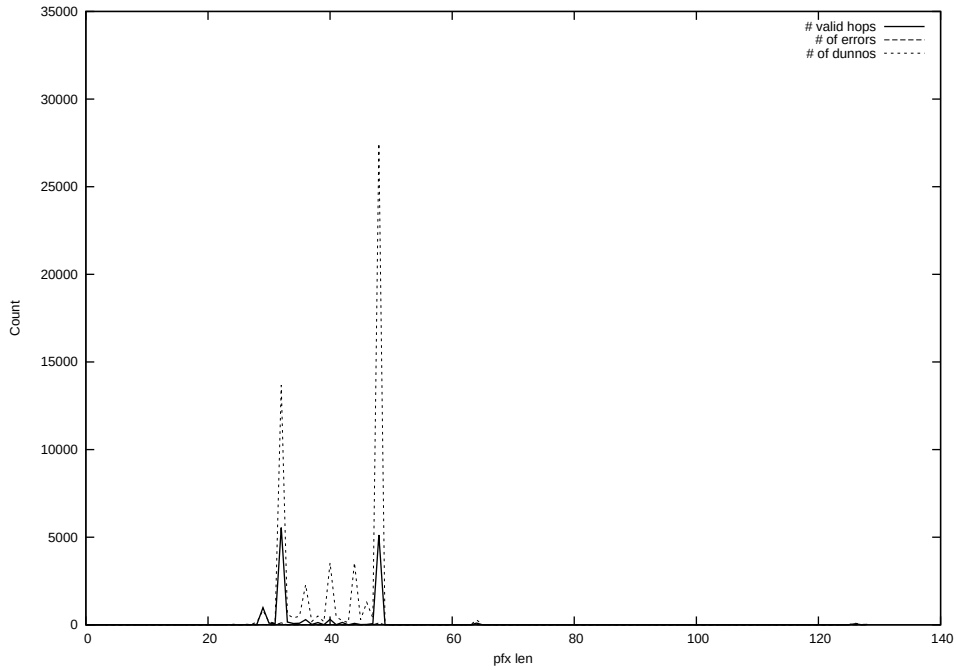
2015-04-26



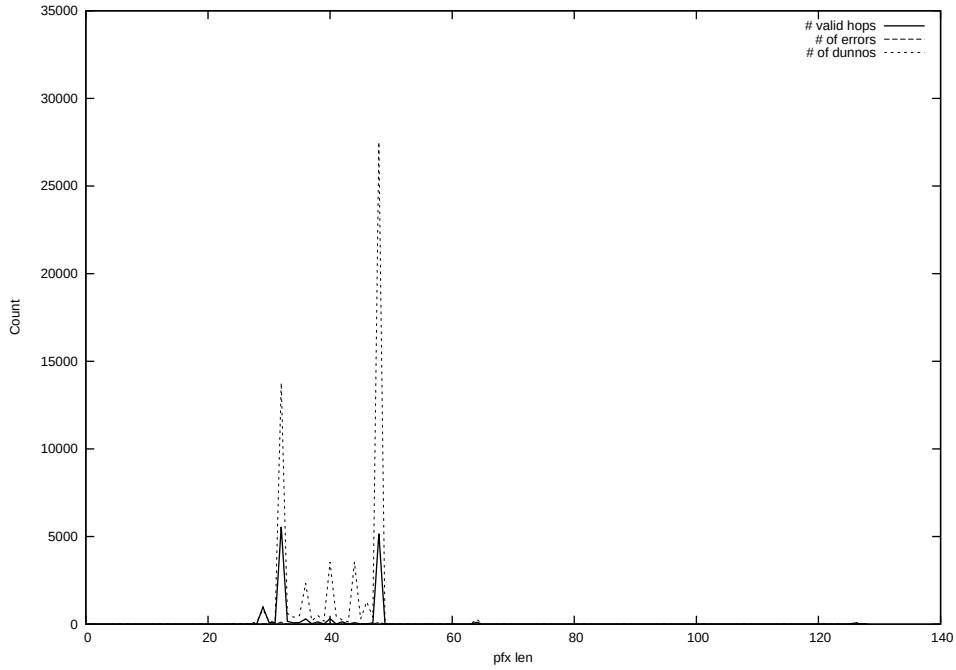
2015-04-27



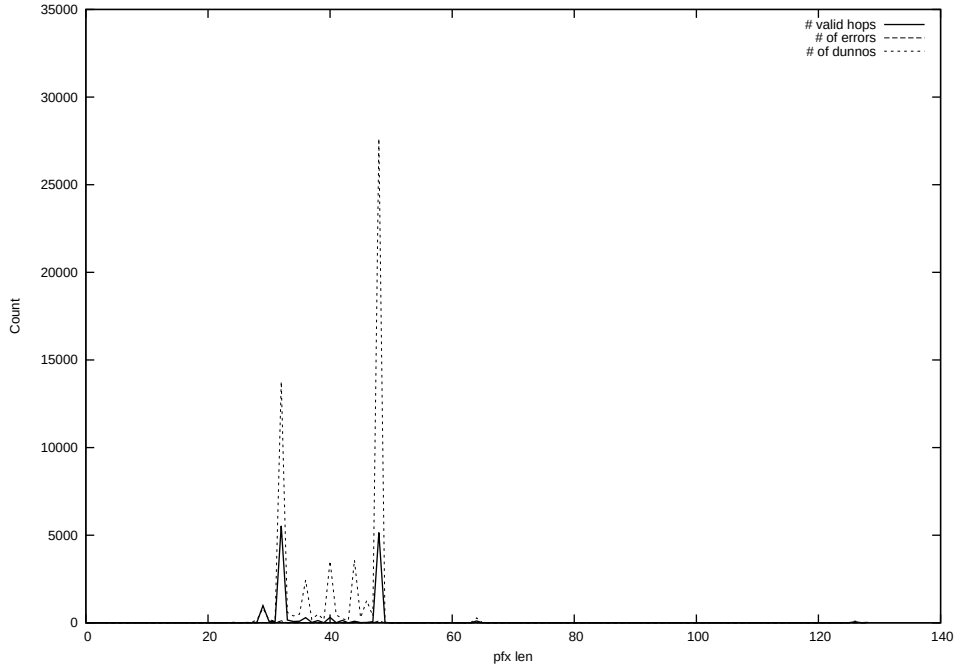
2015-04-28



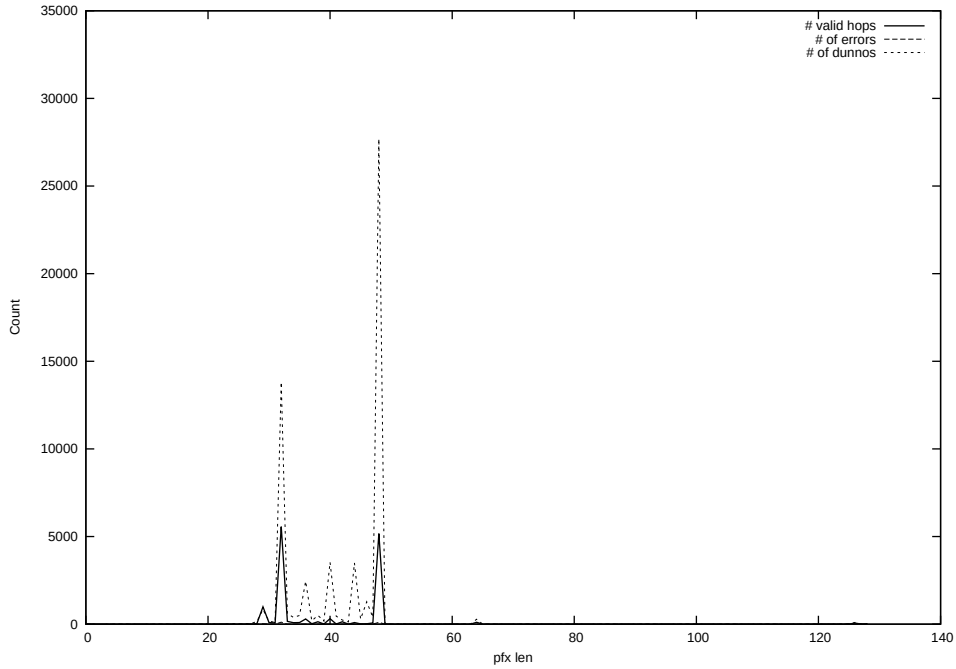
2015-04-29



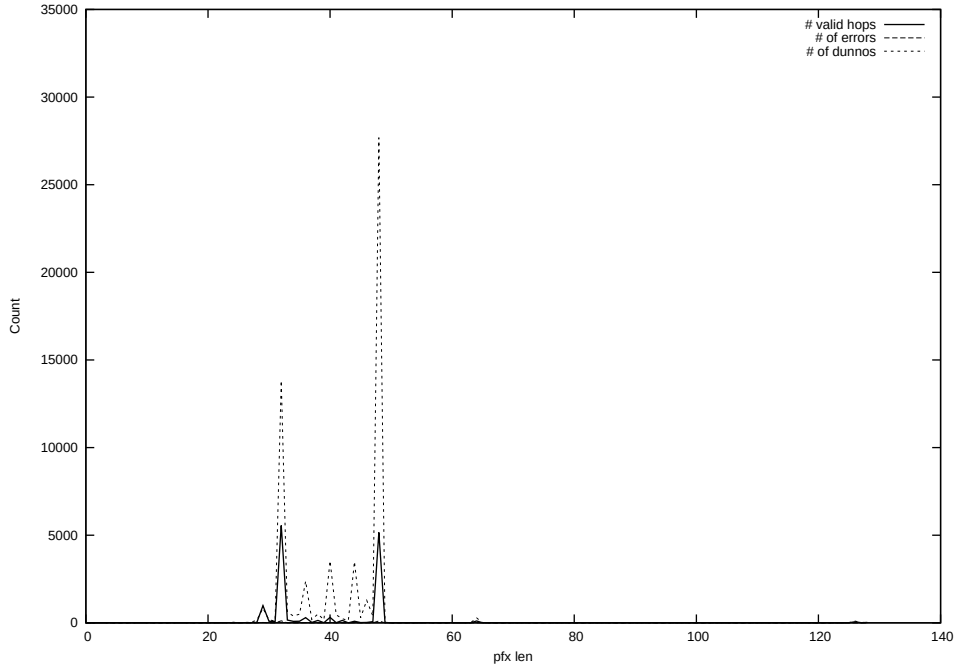
2015-04-30



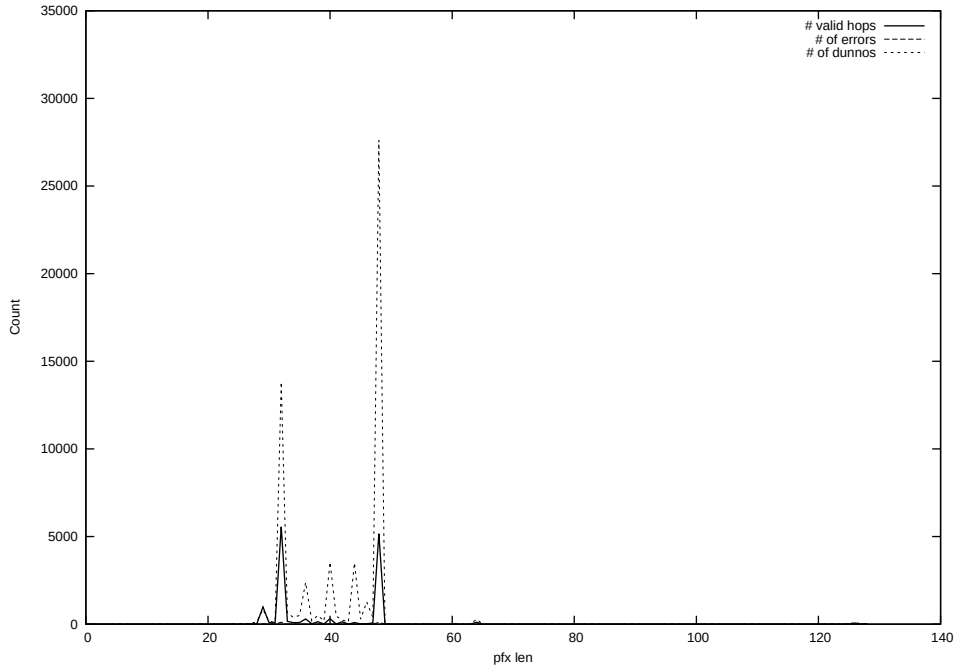
2015-05-01



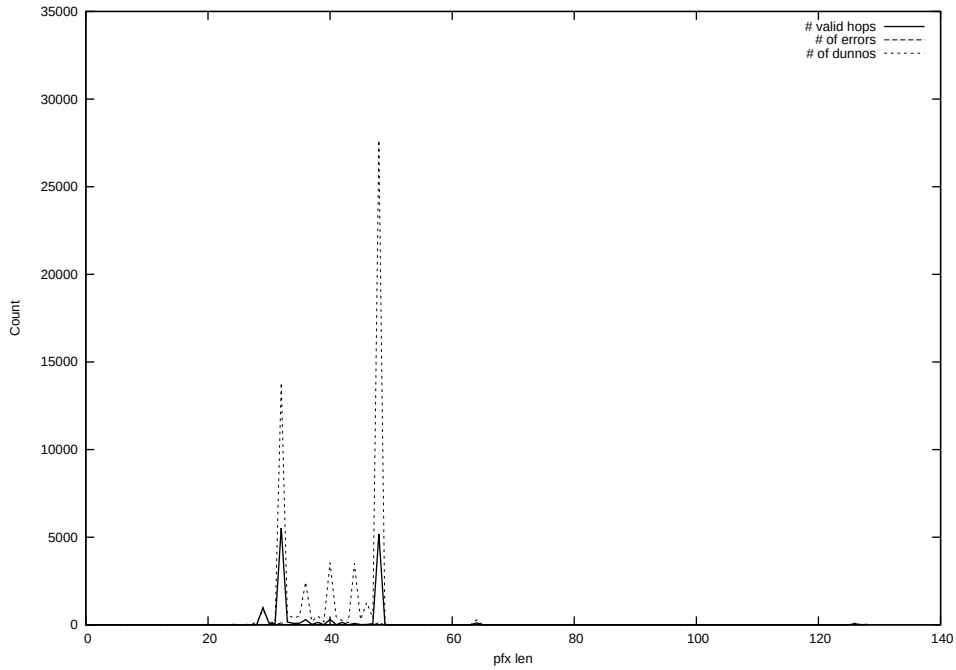
2015-05-02



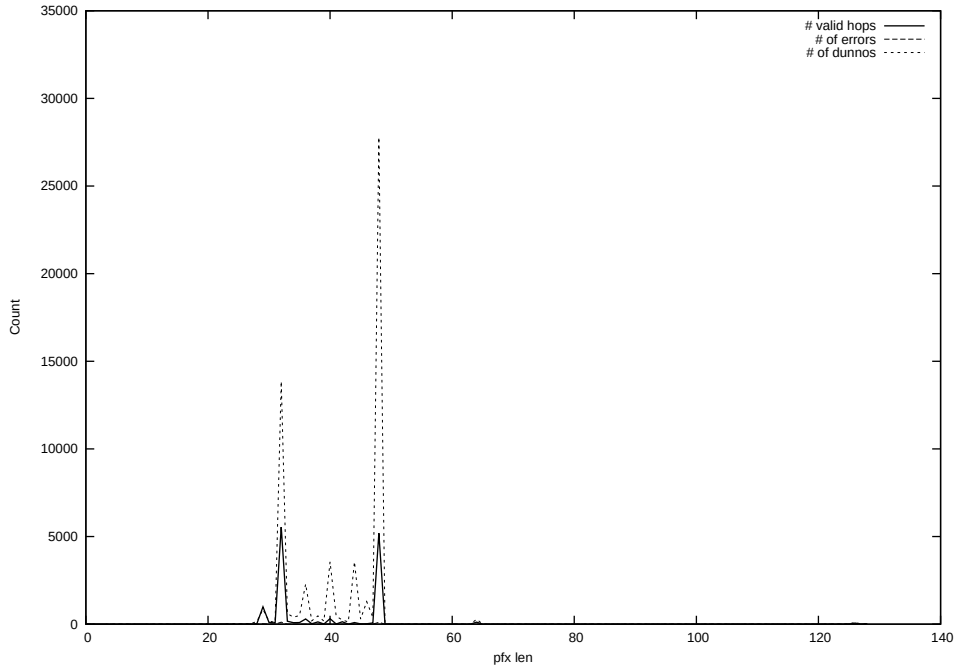
2015-05-03



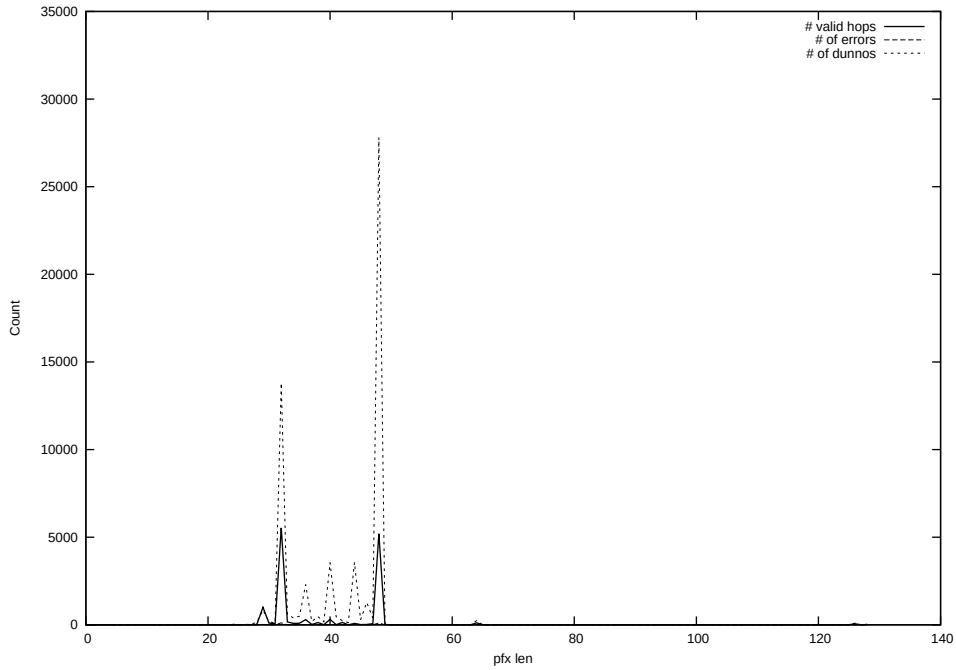
2015-05-04



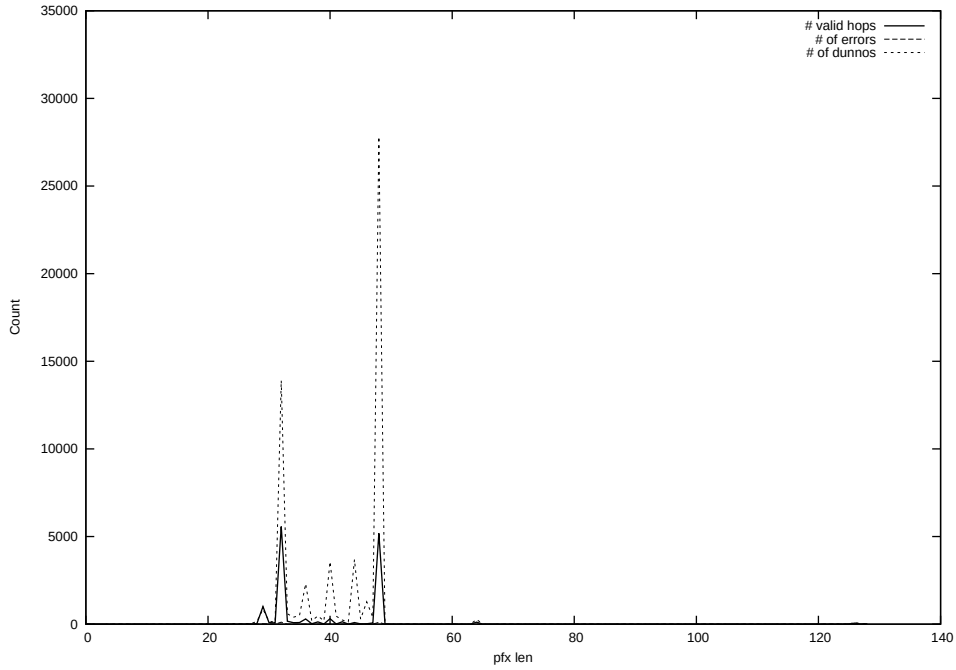
2015-05-05



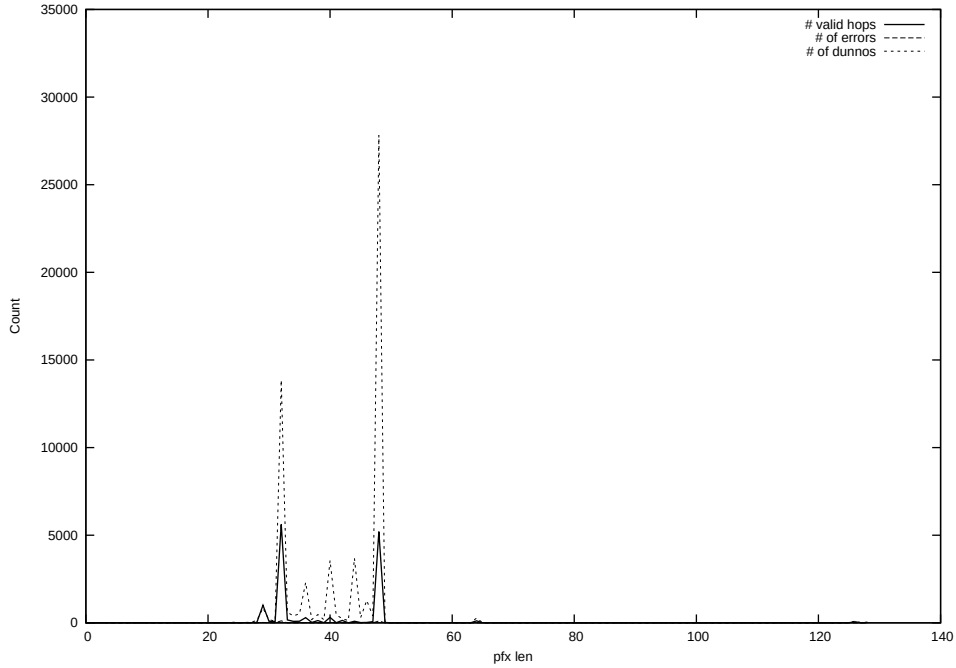
2015-05-06



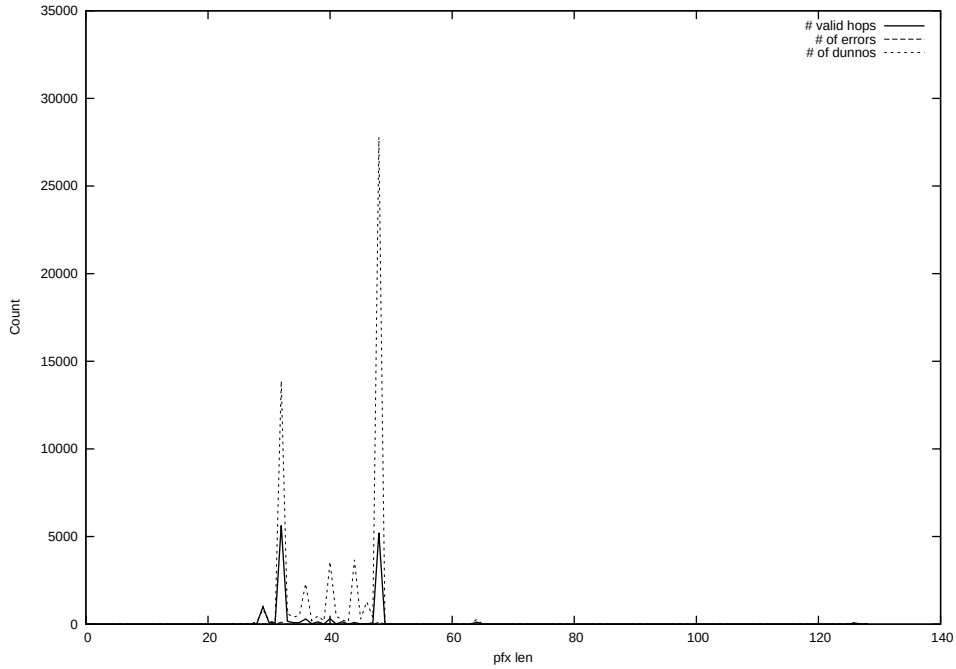
2015-05-07



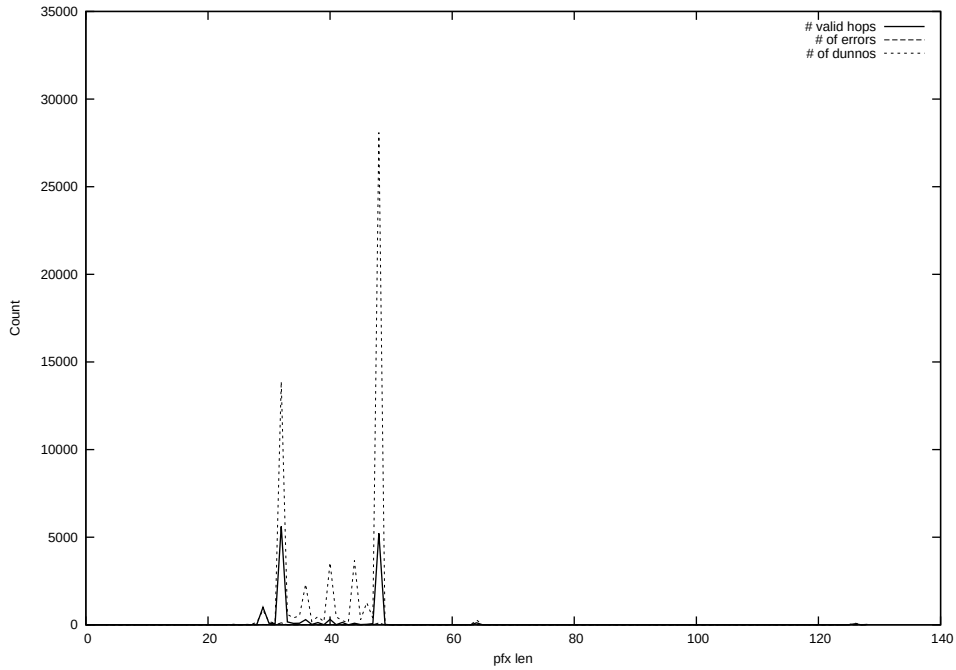
2015-05-08



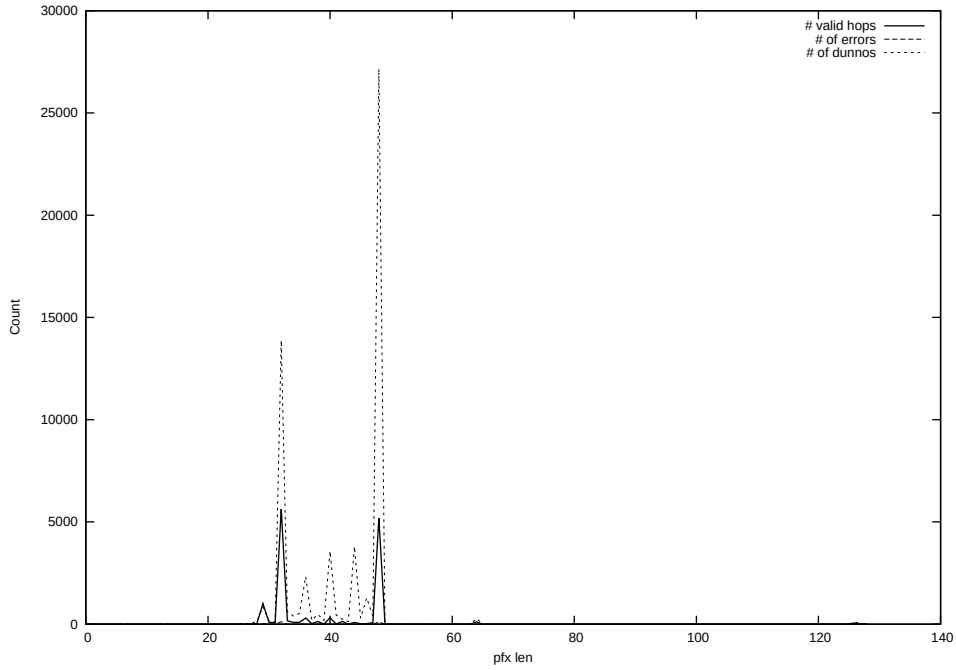
2015-05-09



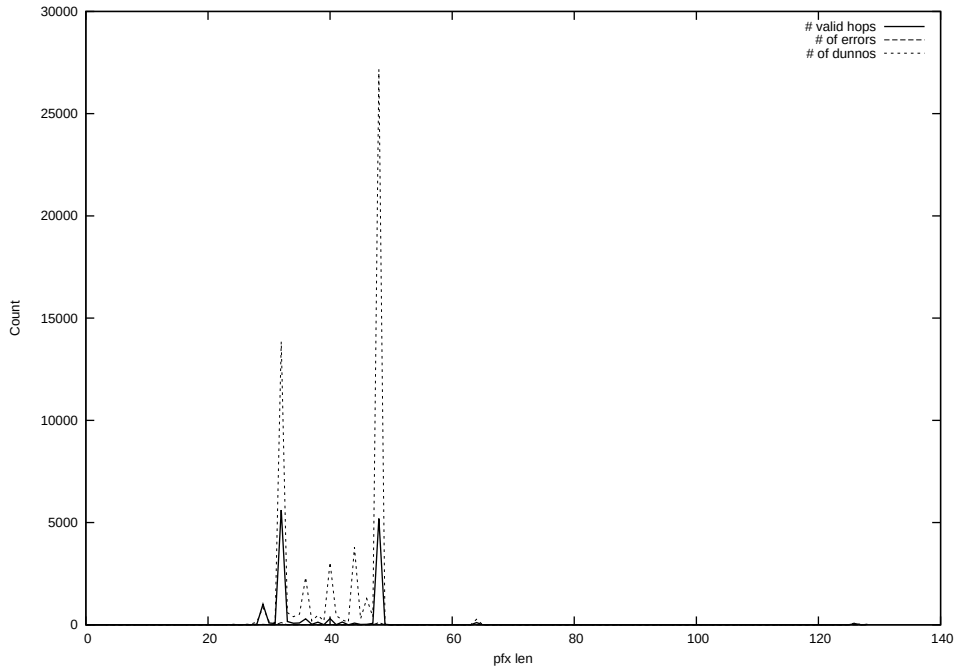
2015-05-10



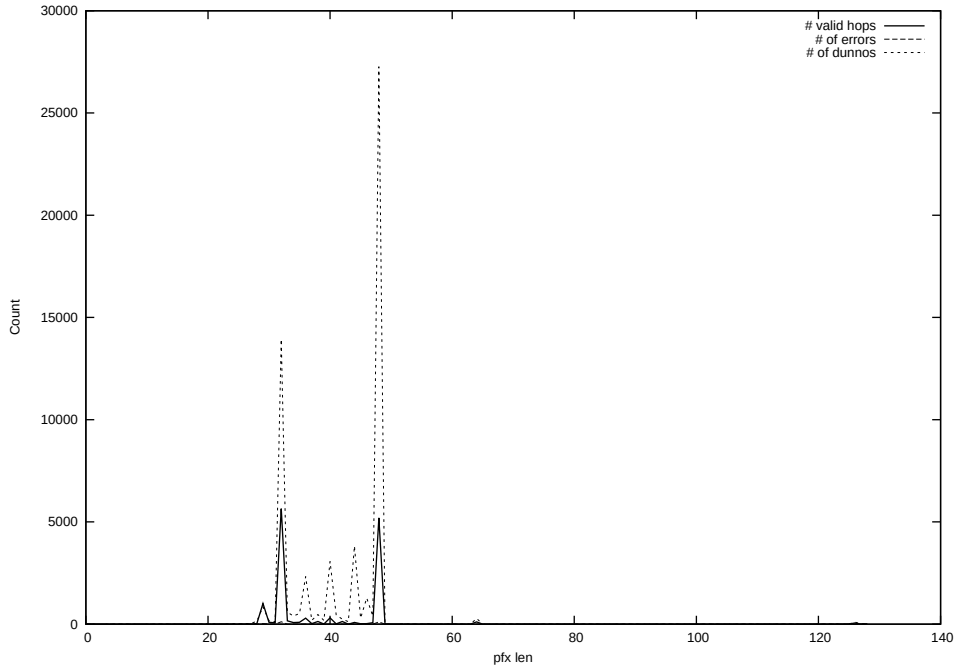
2015-05-11



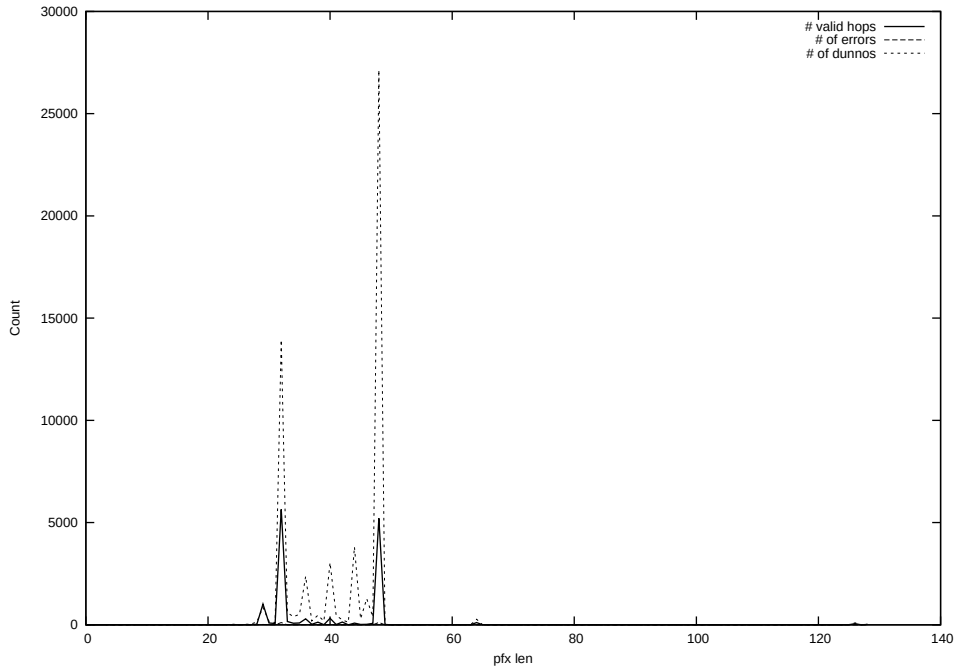
2015-05-12



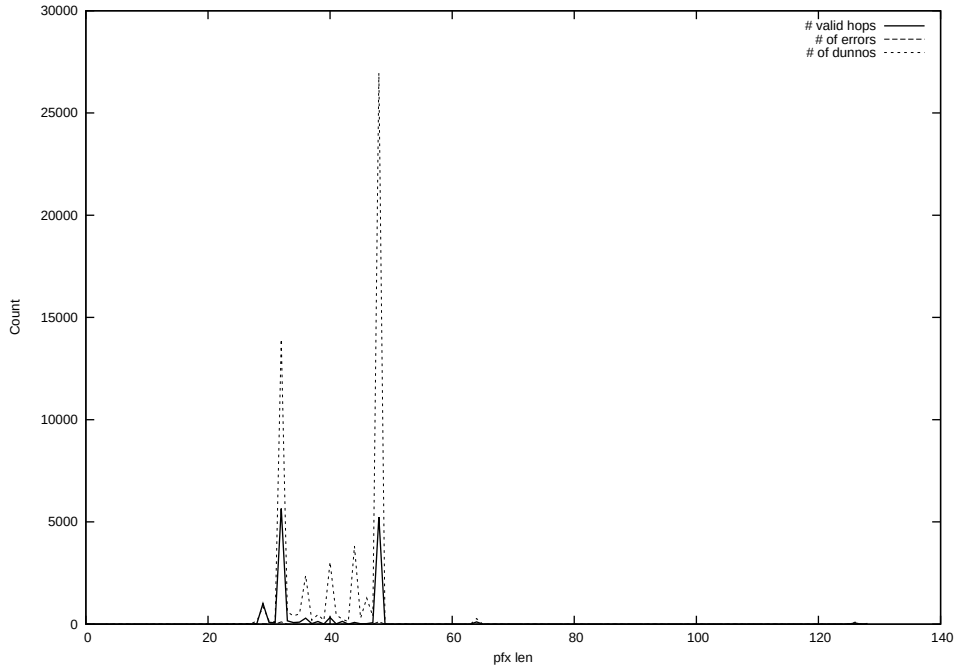
2015-05-13



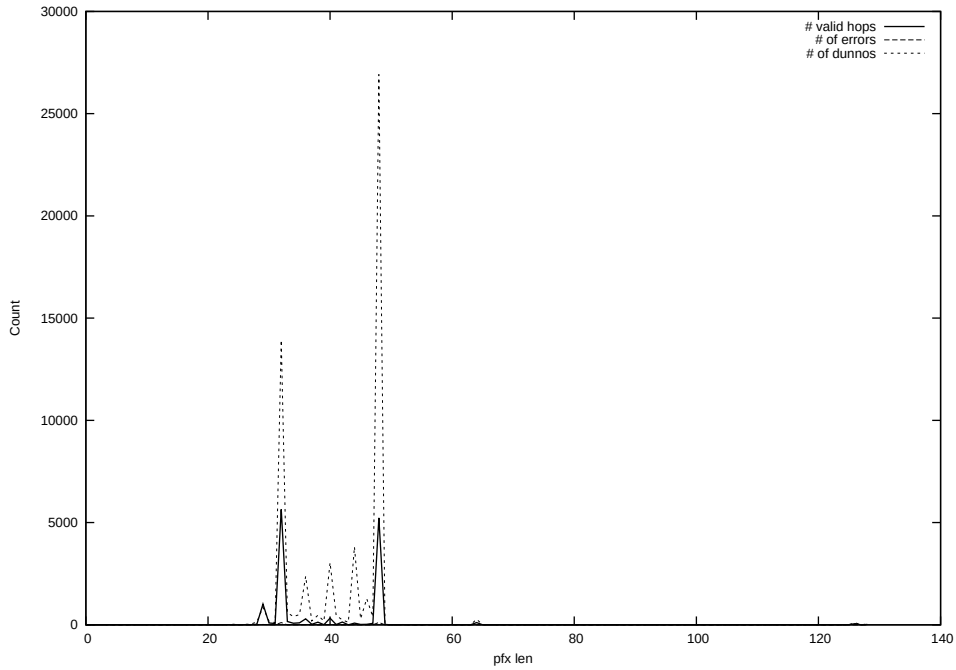
2015-05-14



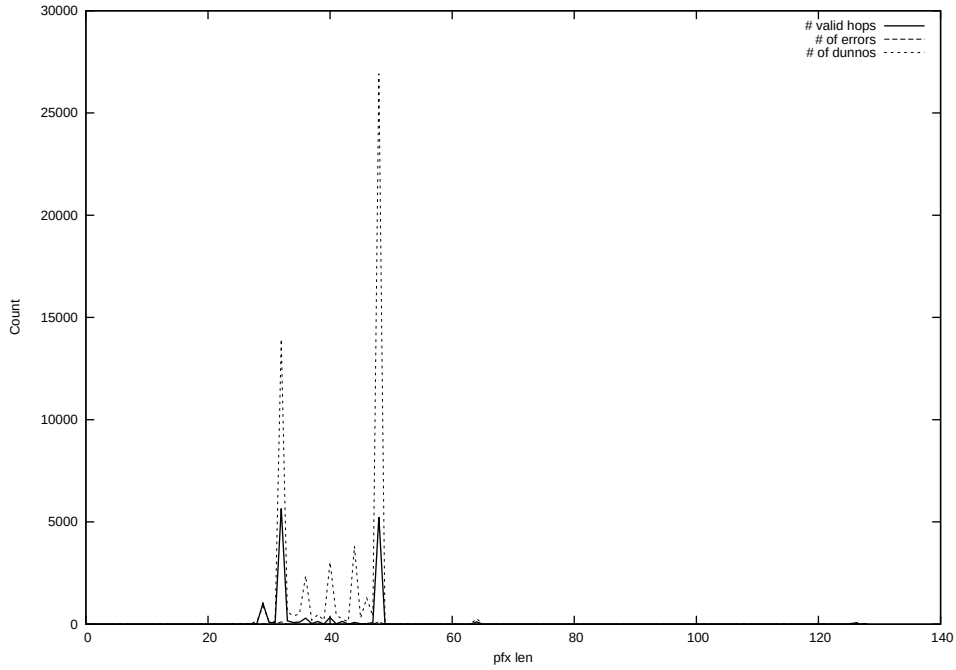
2015-05-15



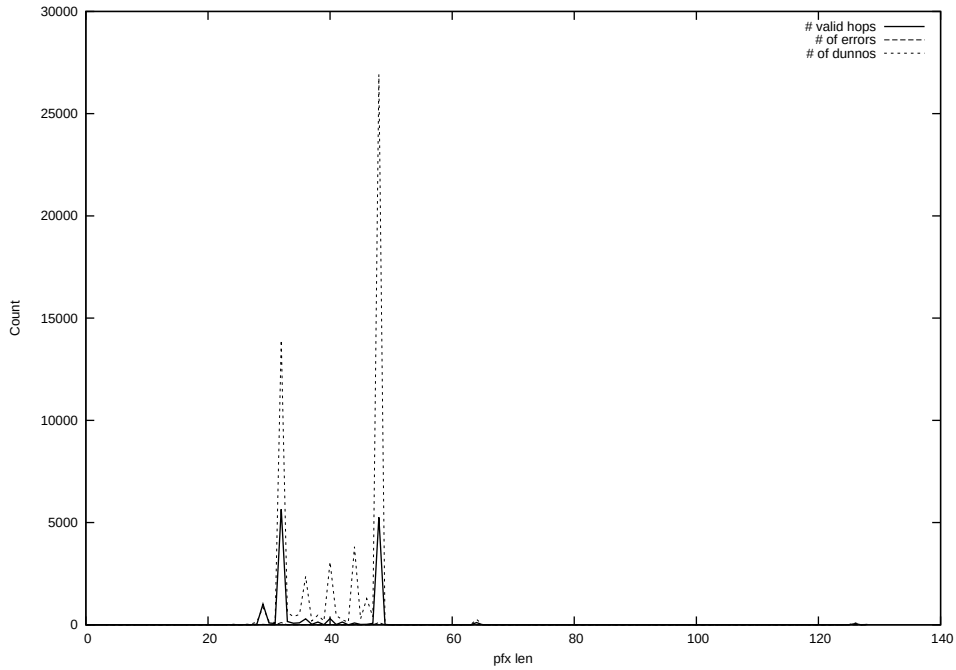
2015-05-16



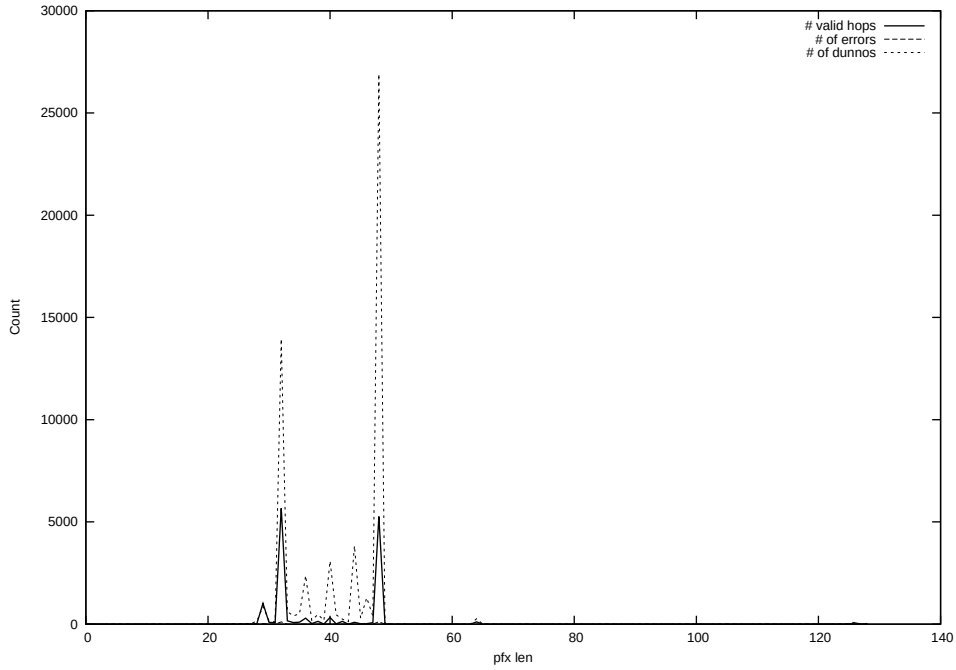
2015-05-17



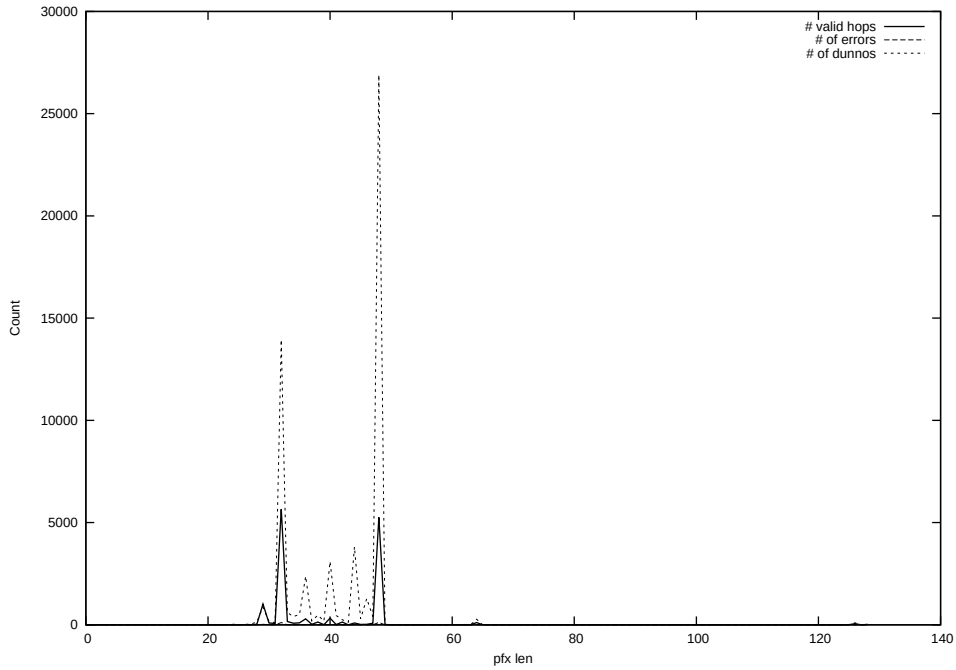
2015-05-18



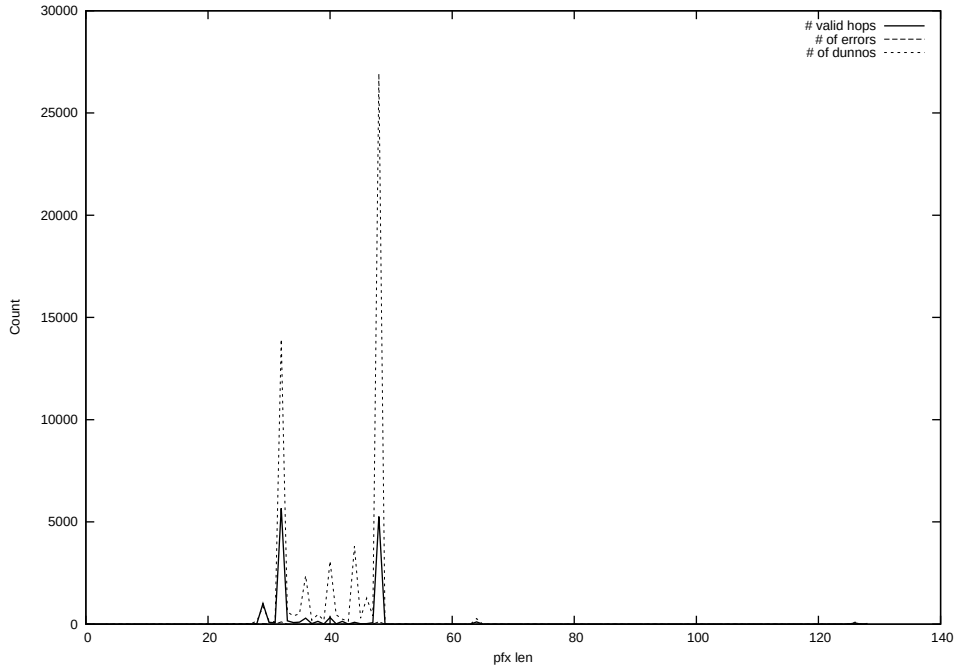
2015-05-19



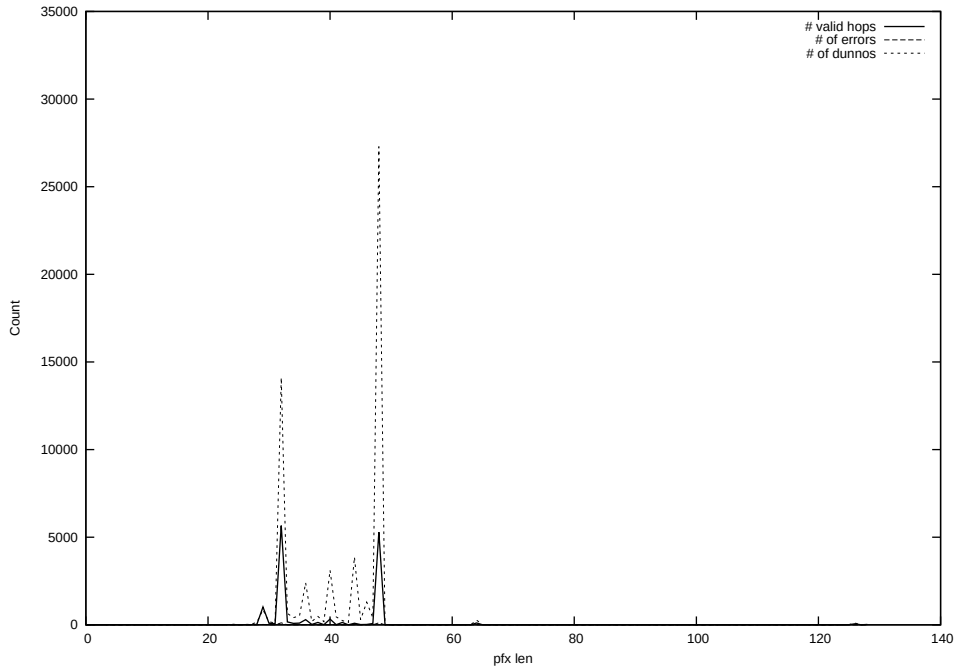
2015-05-20



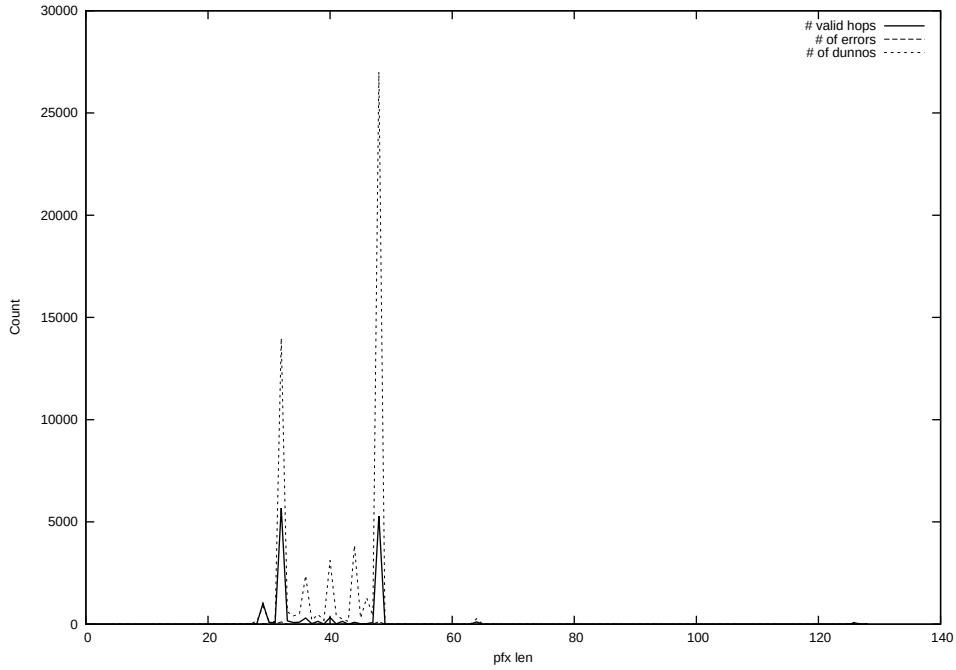
2015-05-21



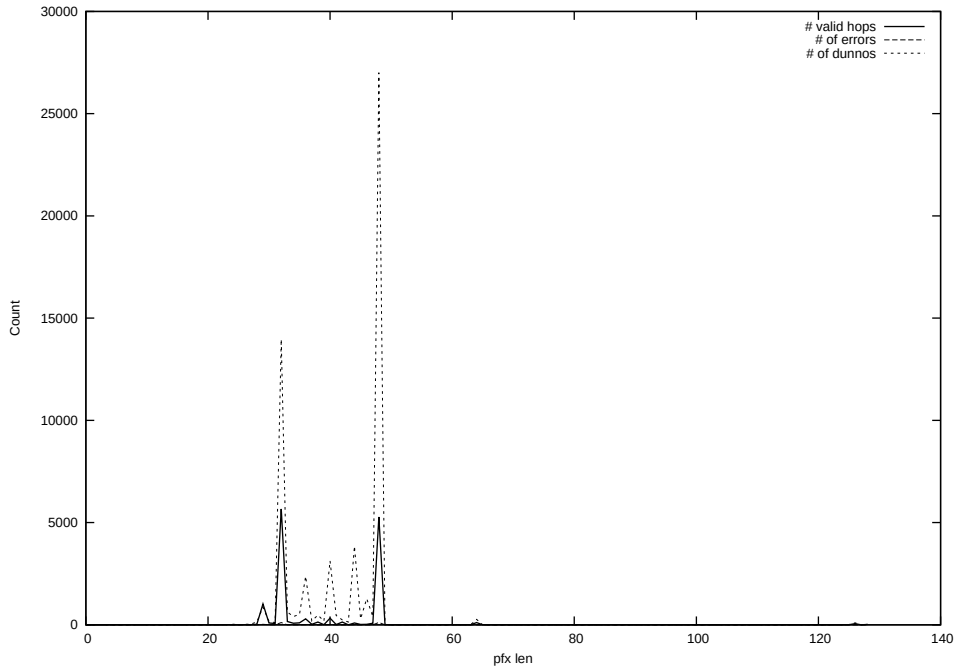
2015-05-22



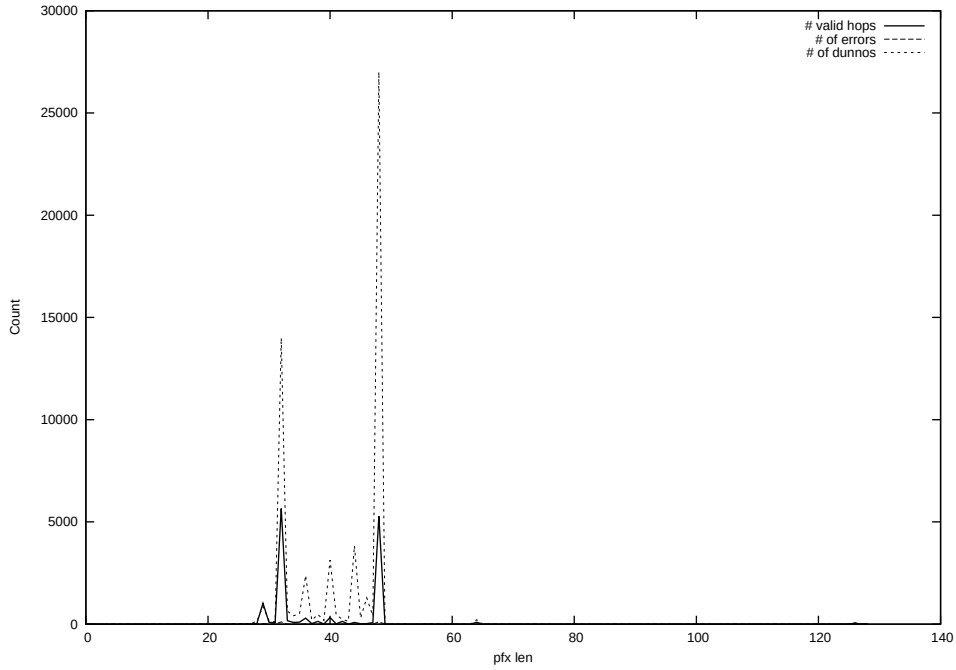
2015-05-23



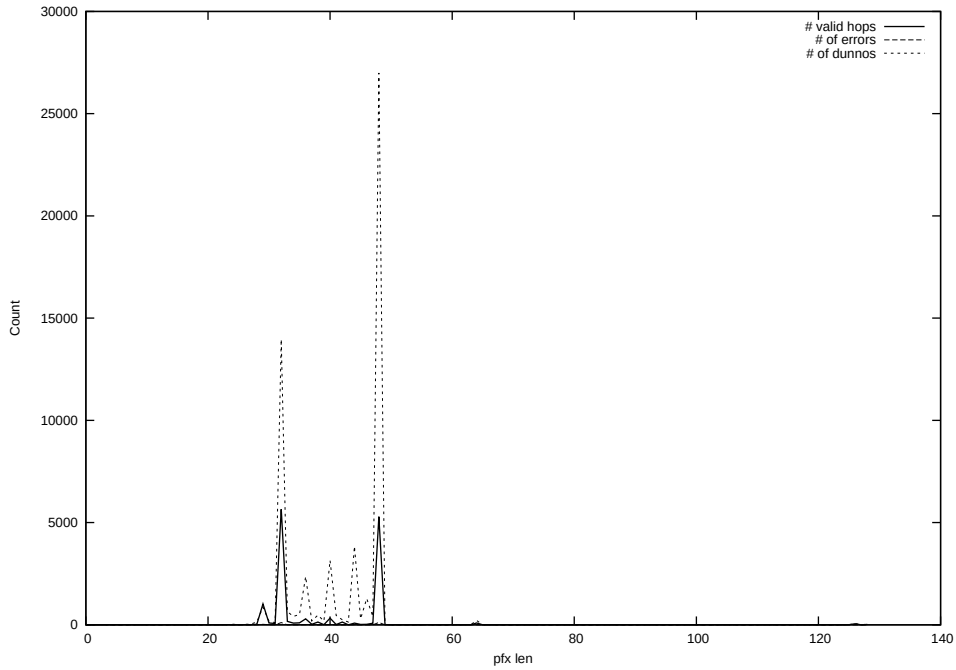
2015-05-24



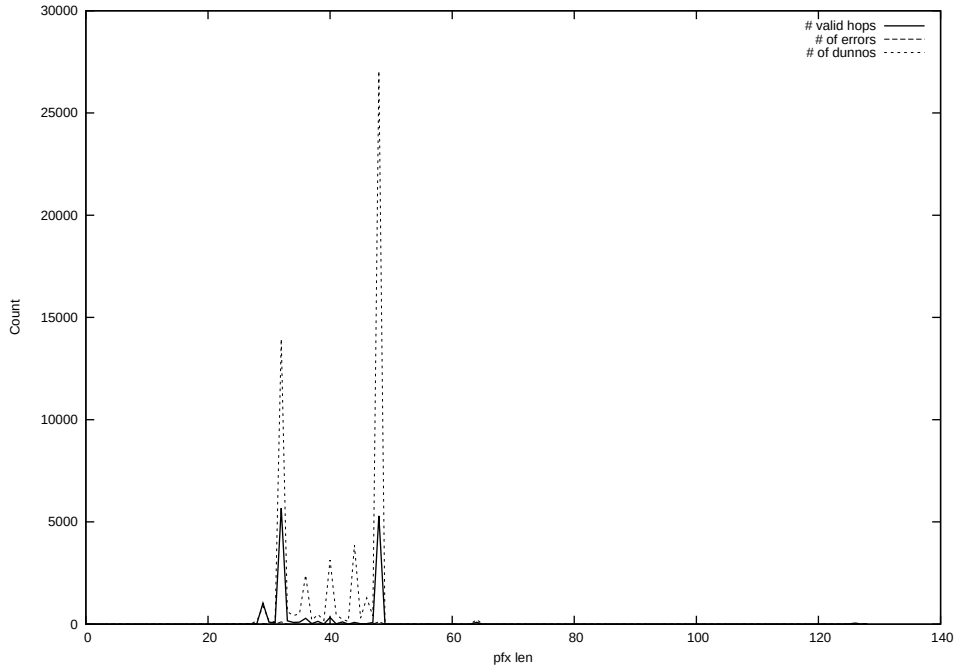
2015-05-25



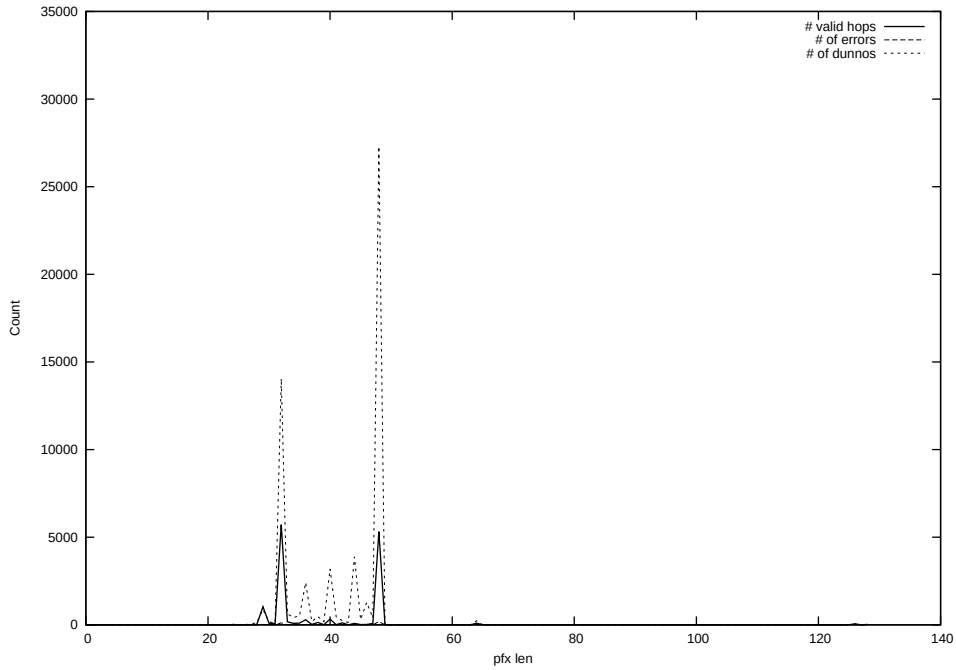
2015-05-26



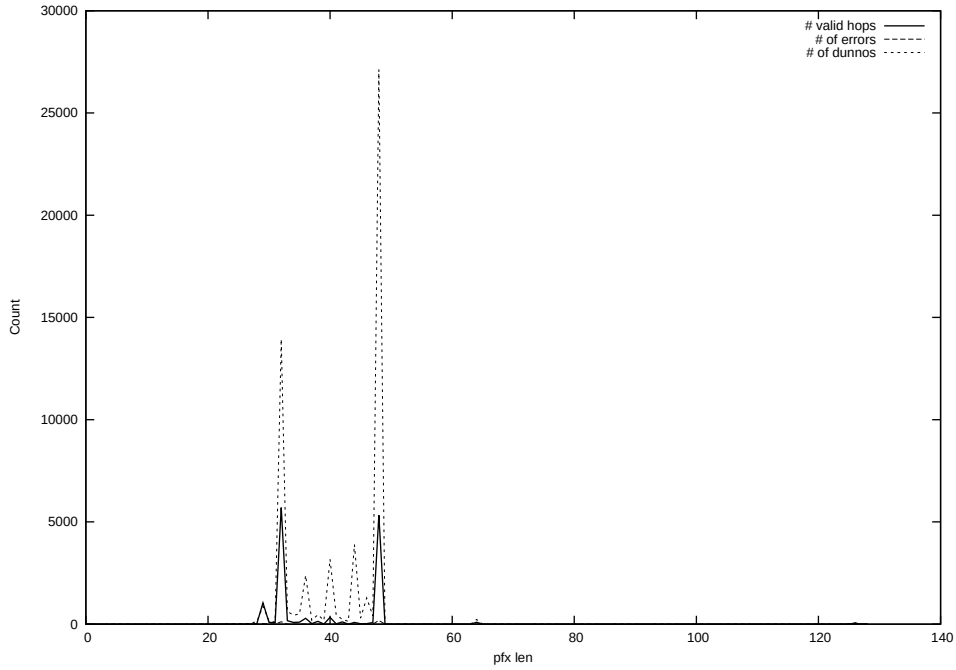
2015-05-27



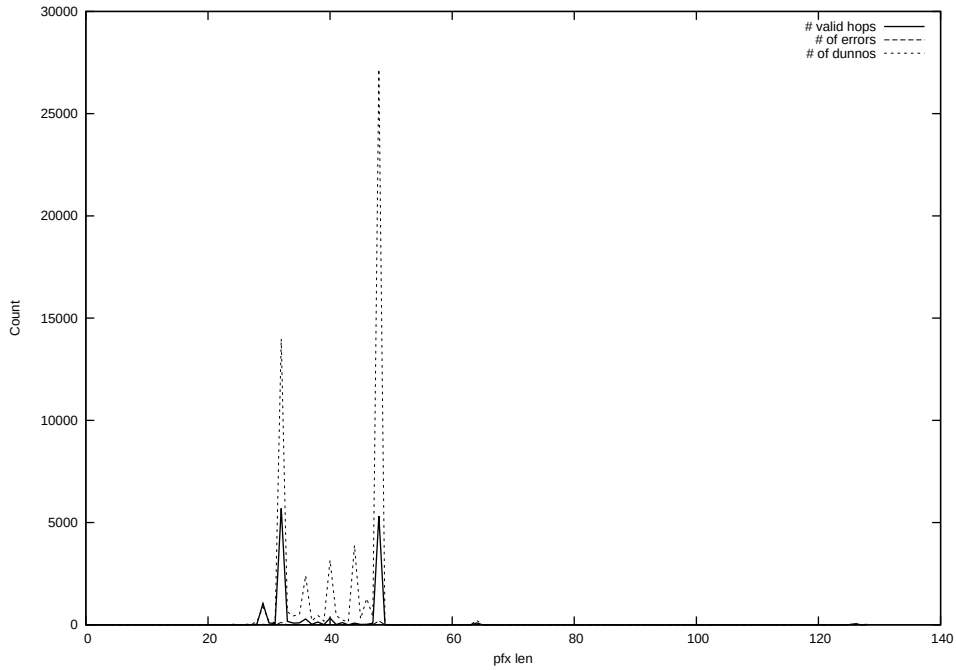
2015-05-28



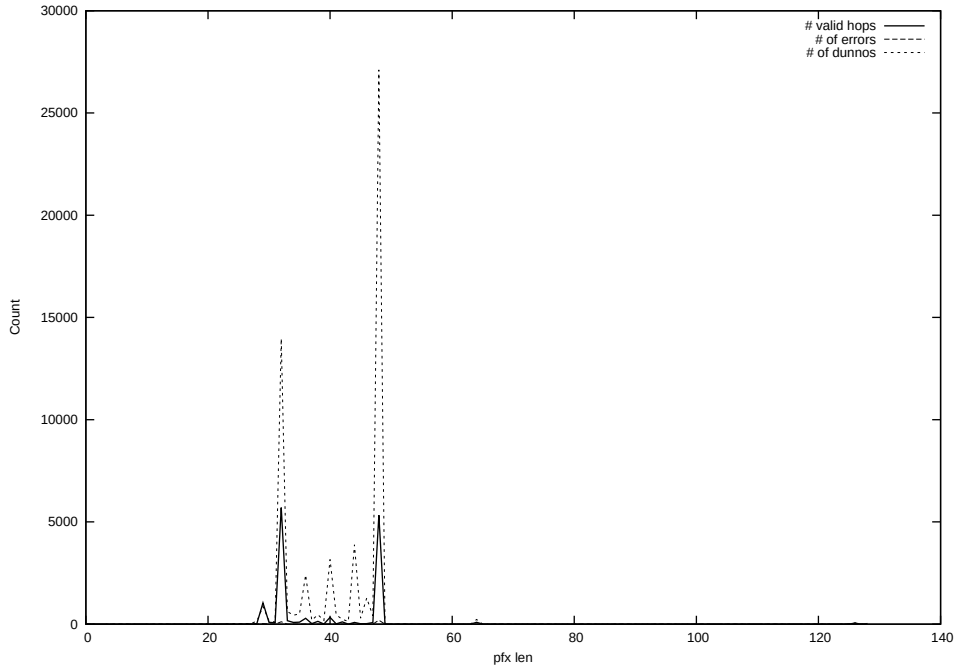
2015-05-29



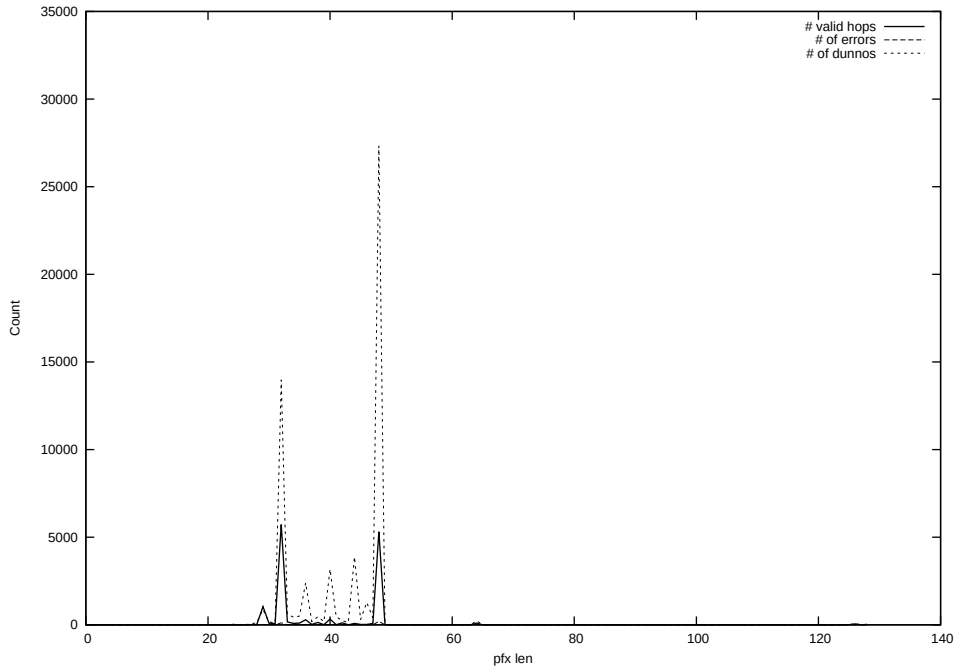
2015-05-30



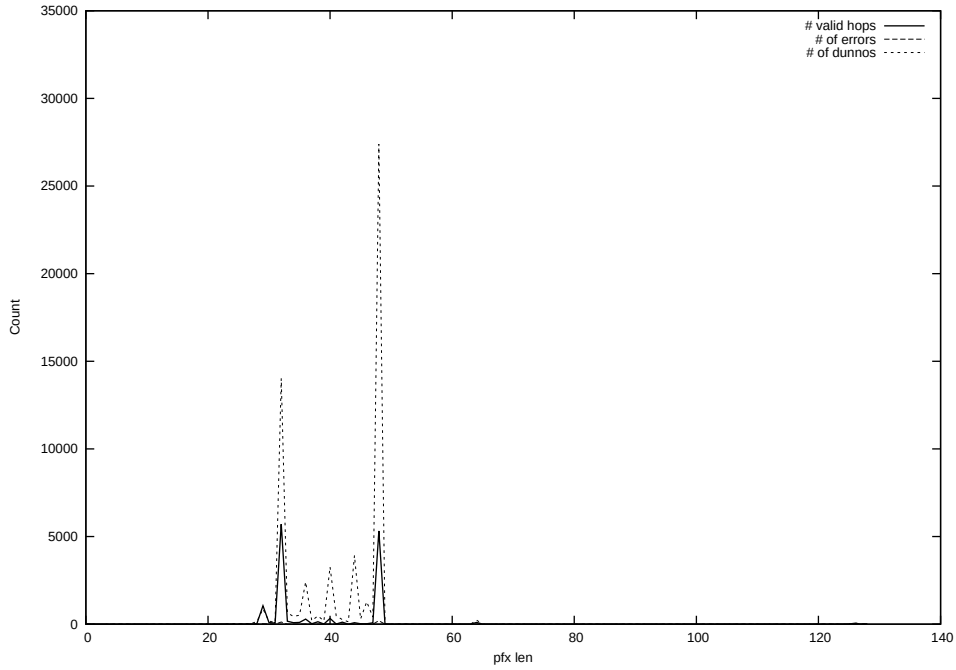
2015-05-31



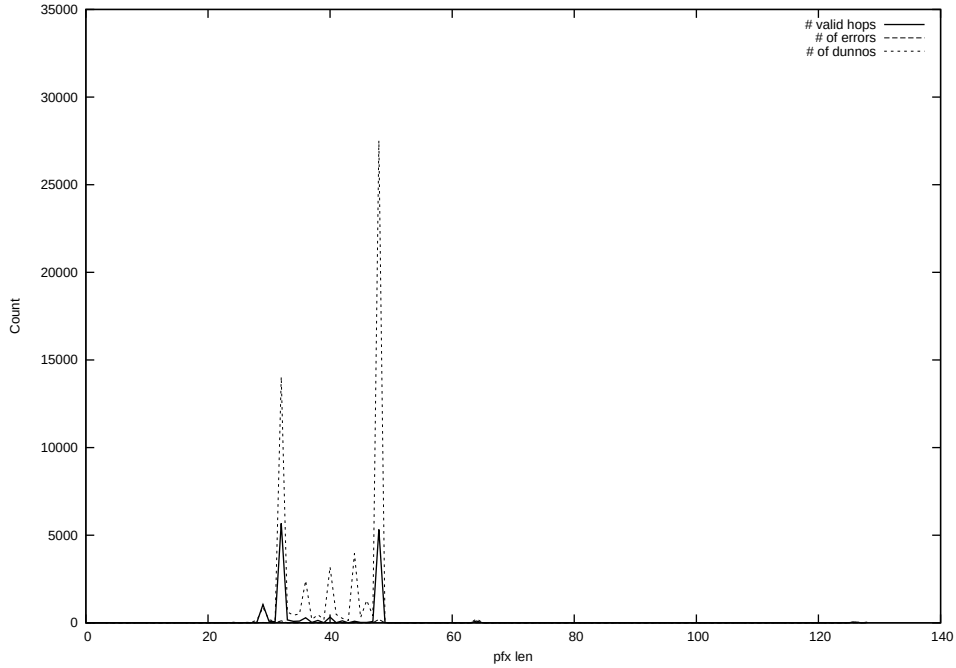
2015-06-01



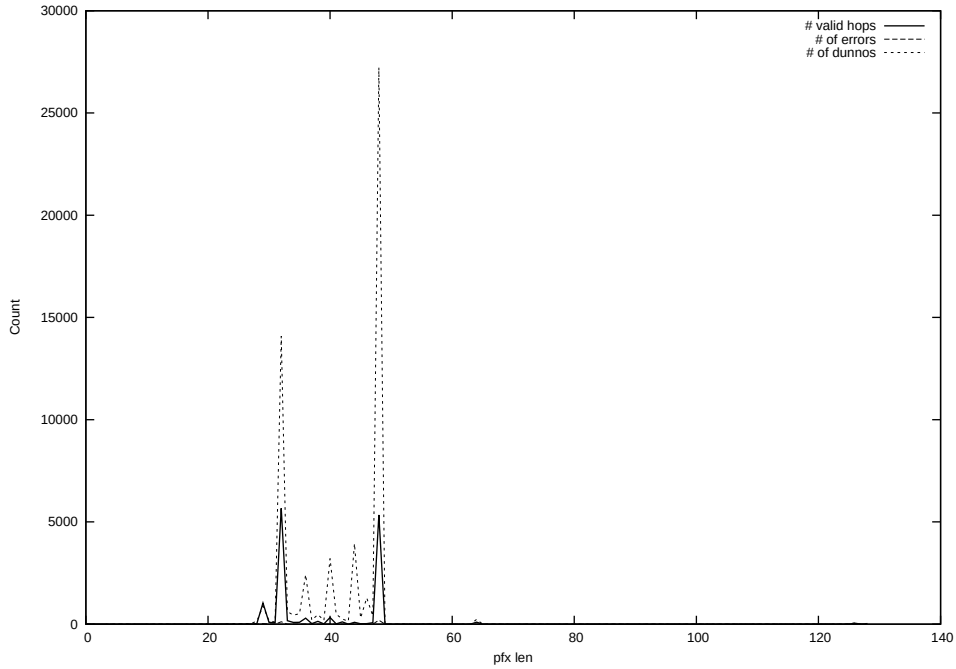
2015-06-02



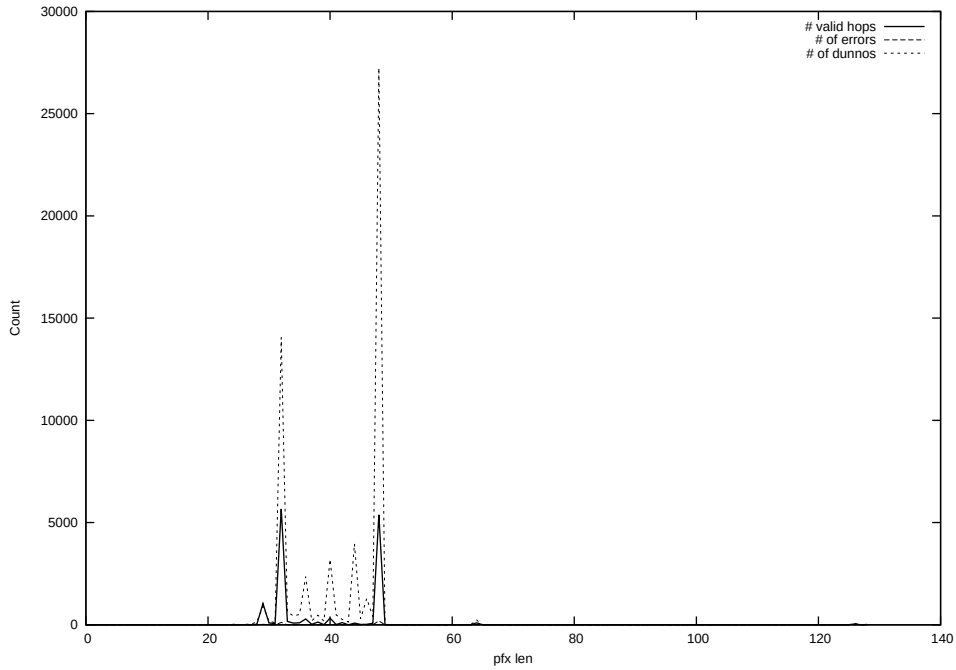
2015-06-03



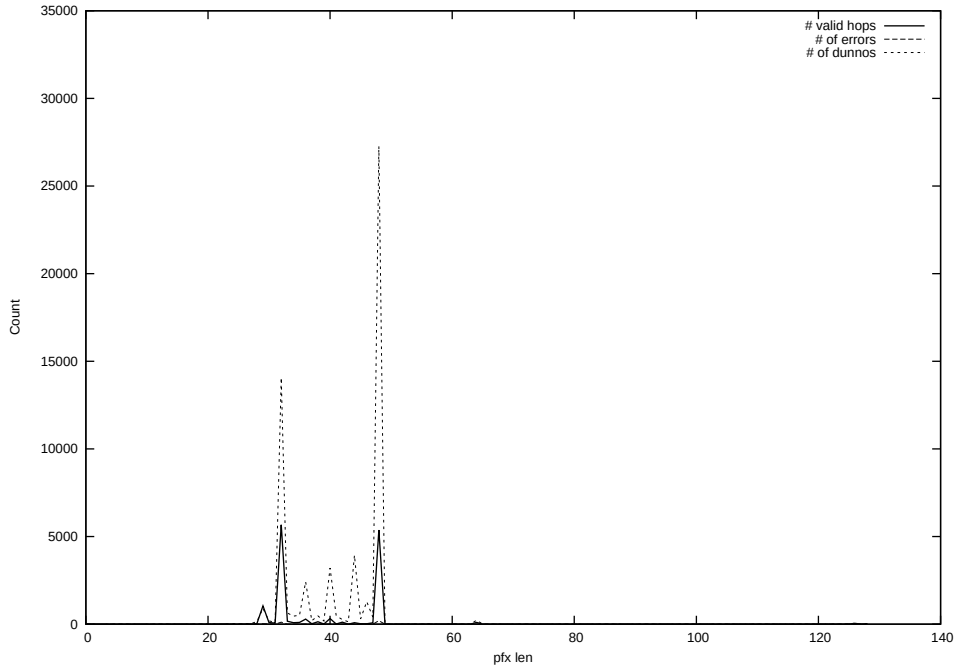
2015-06-04



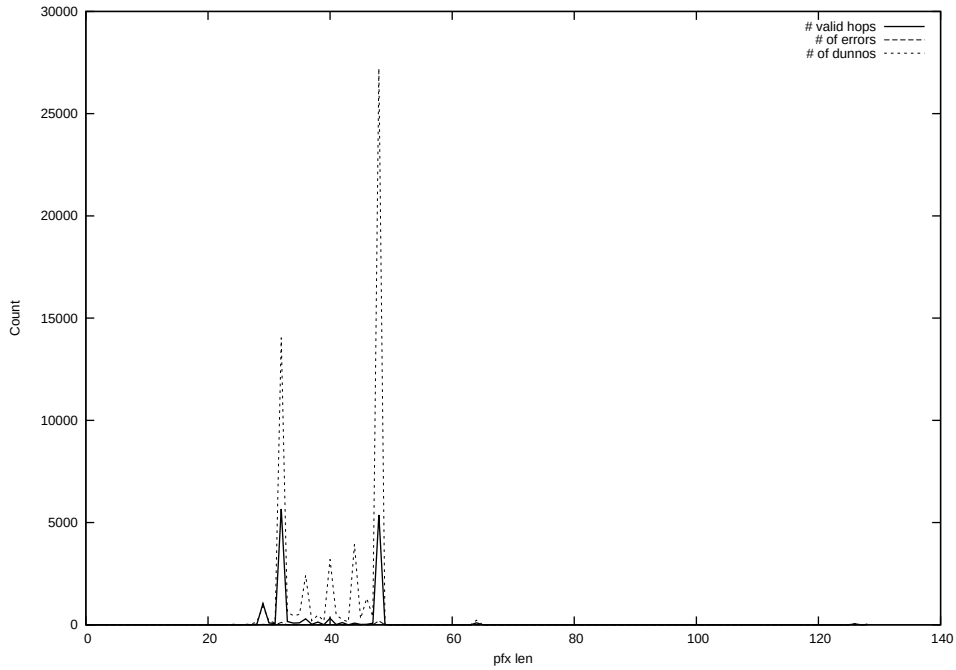
2015-06-05



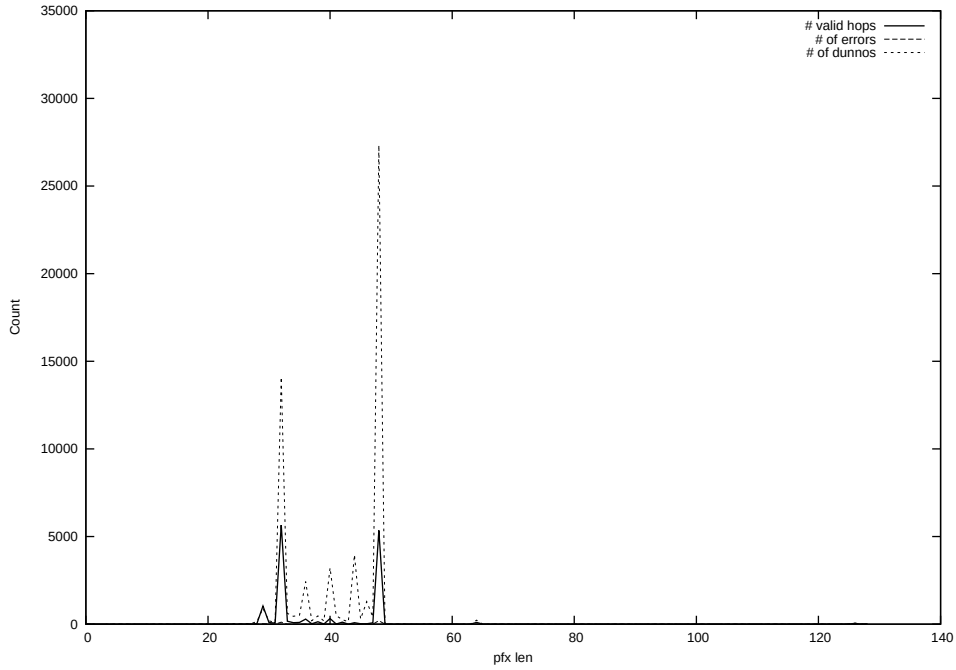
2015-06-06



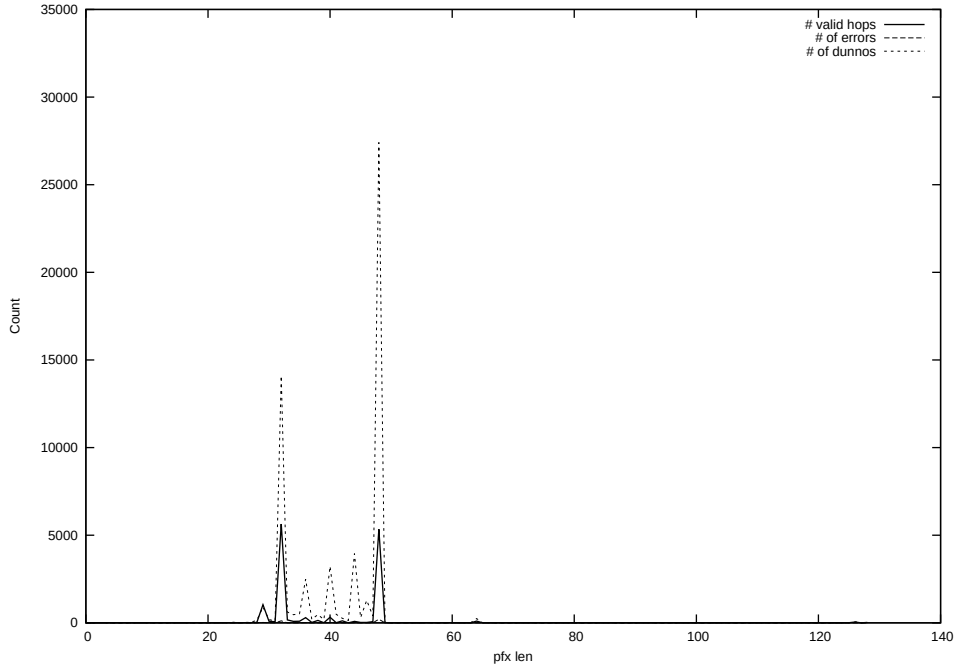
2015-06-07



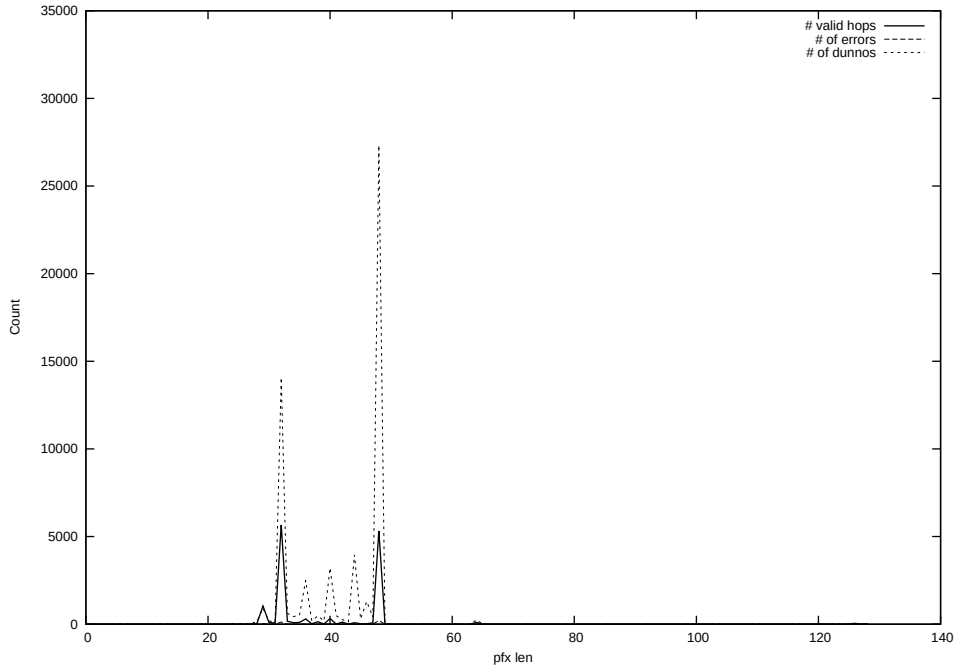
2015-06-08



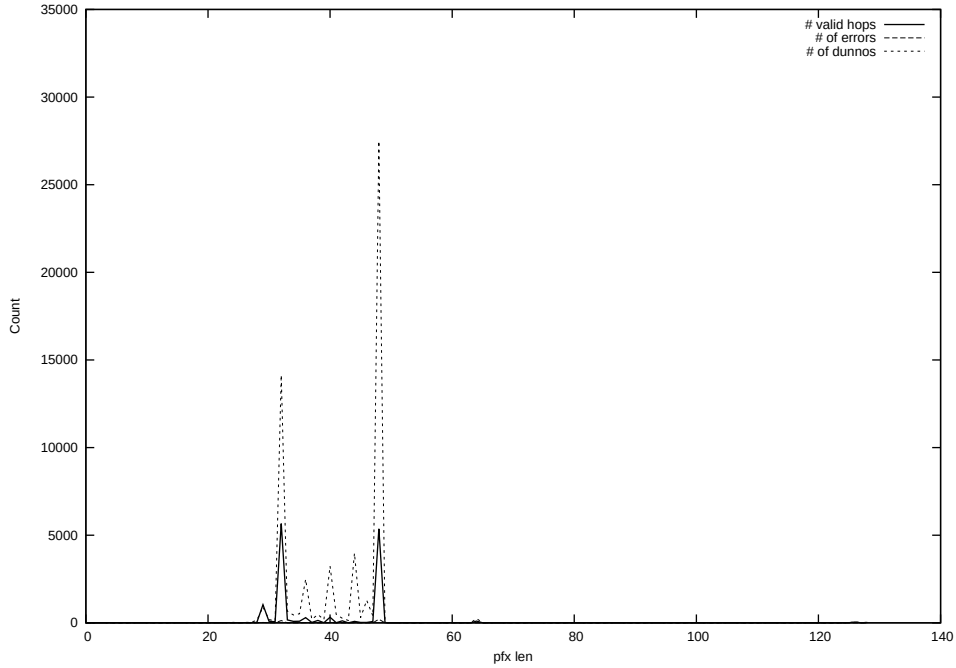
2015-06-09



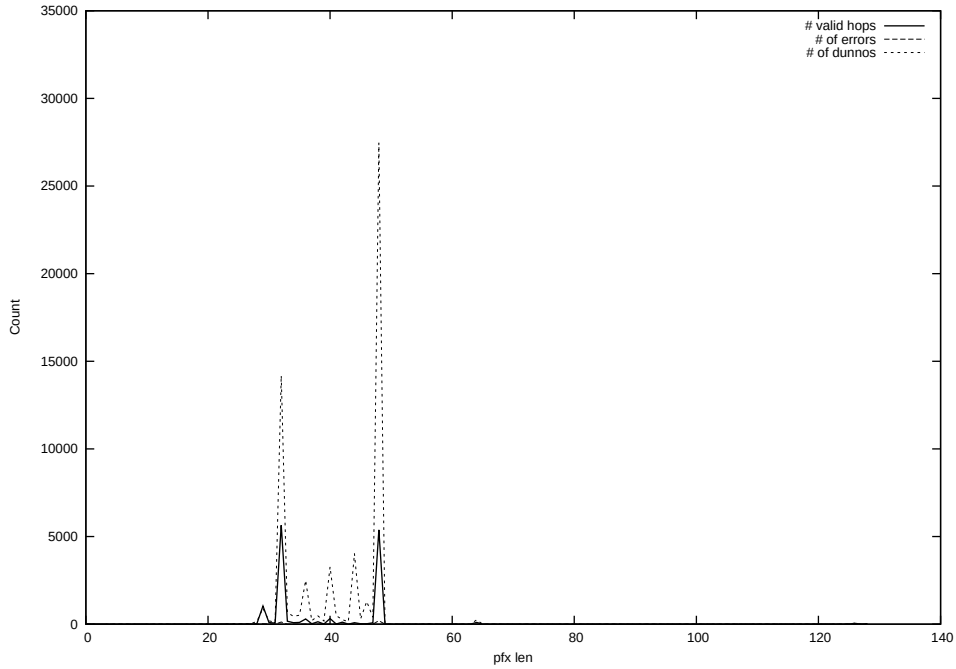
2015-06-10



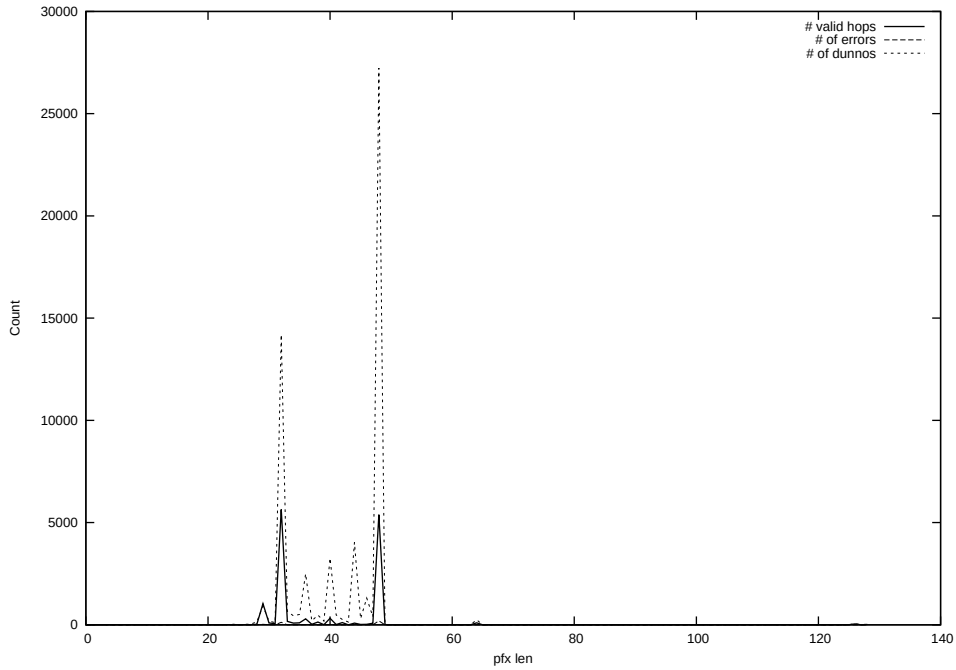
2015-06-11



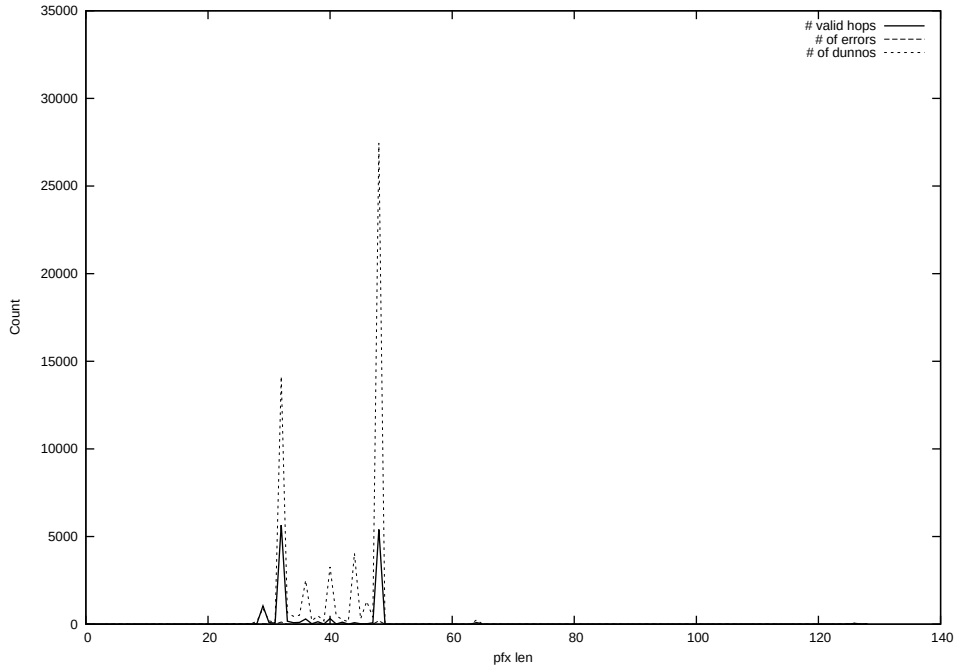
2015-06-12



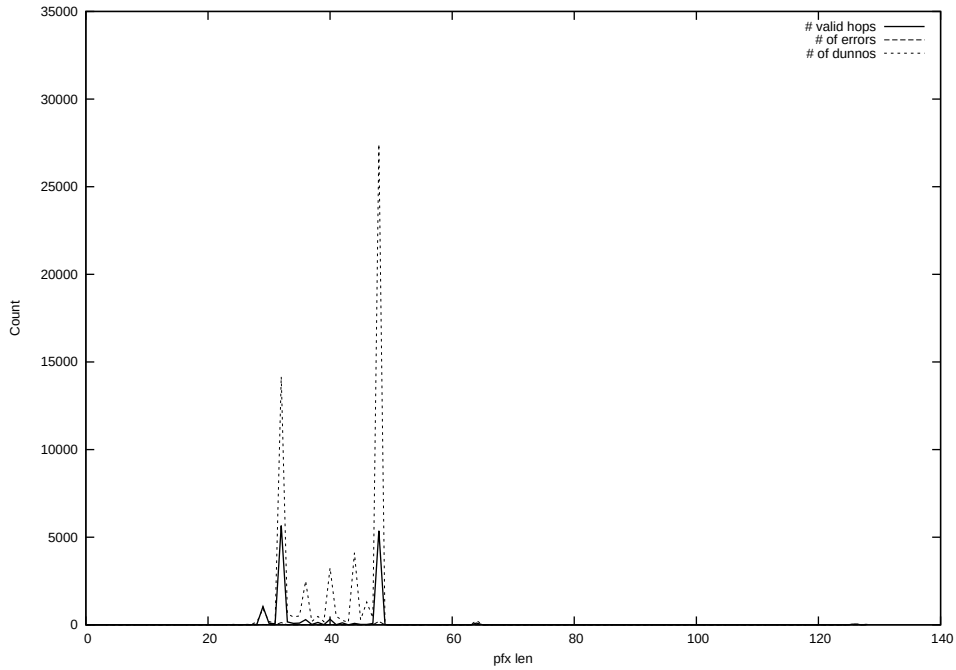
2015-06-13



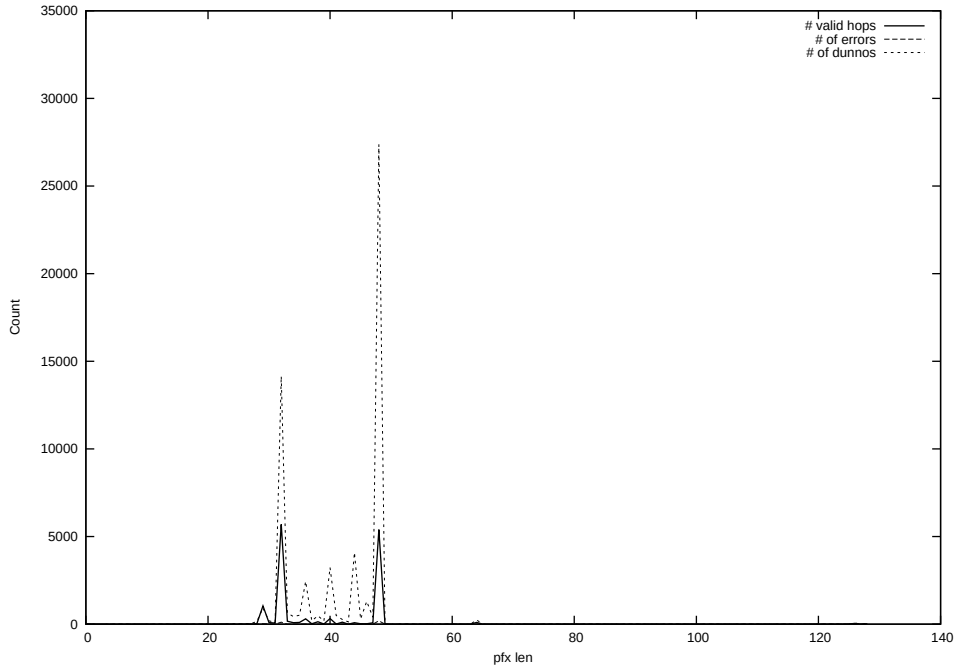
2015-06-14



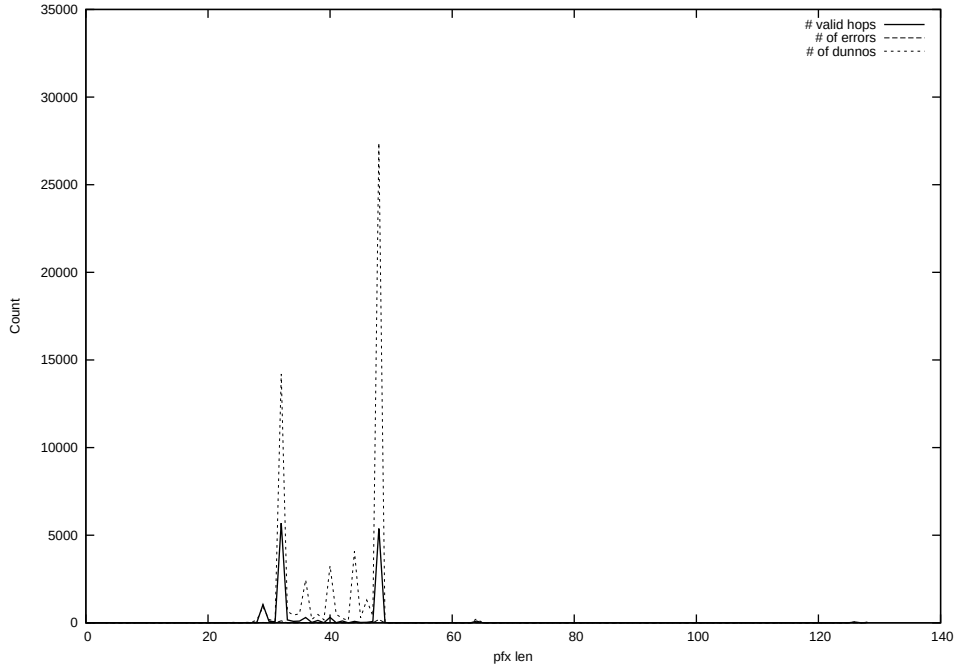
2015-06-15



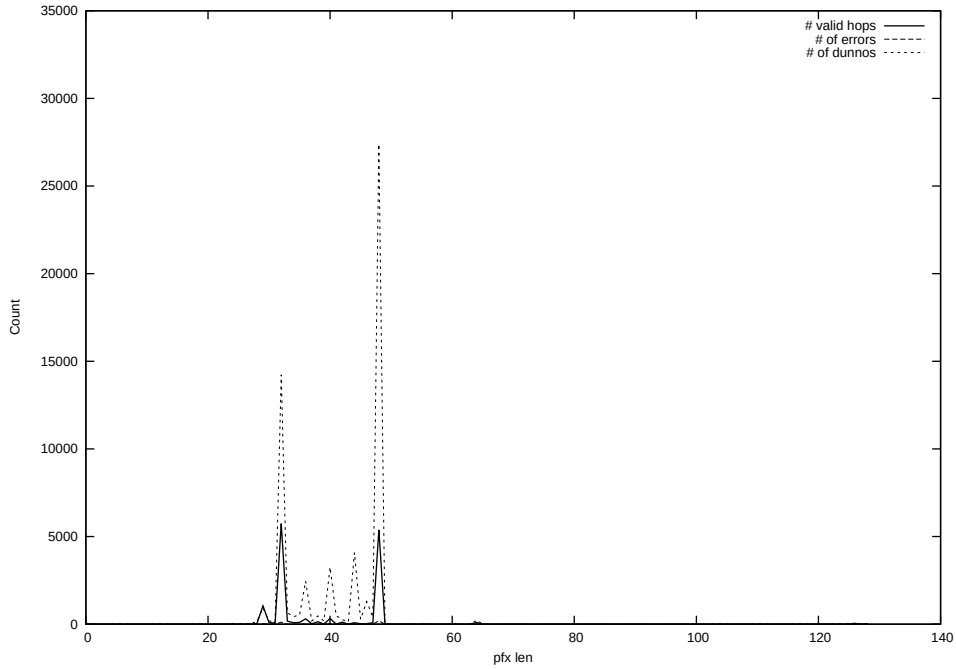
2015-06-16



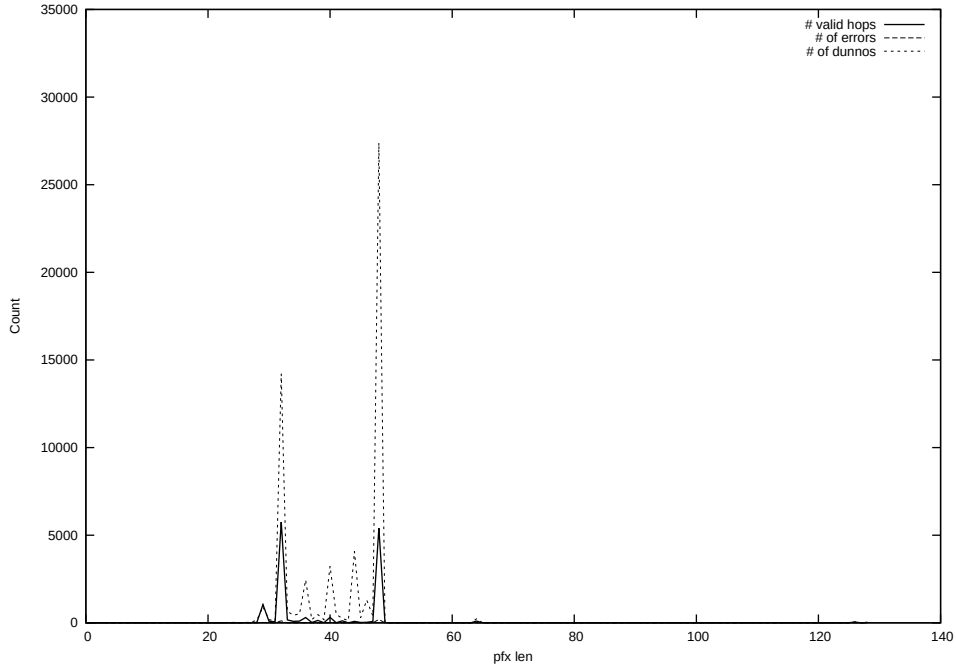
2015-06-17



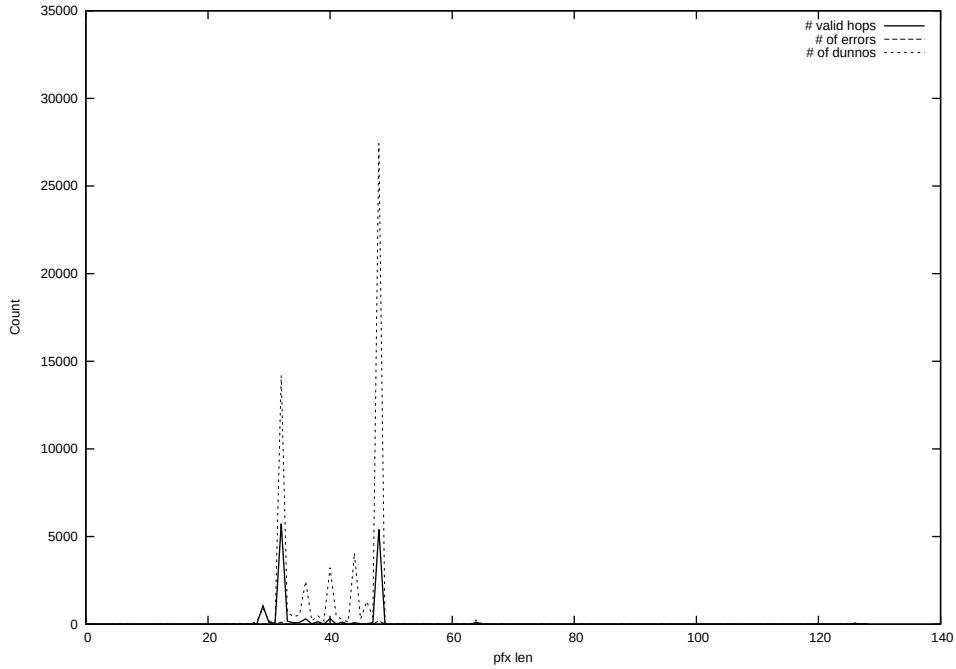
2015-06-18



2015-06-19



2015-06-20



2015-06-21